

SISTEMA PROTOTIPO DE RECOLECCIÓN Y GESTIÓN DE DATOS PARA LA RED DE
TRANSPORTE PÚBLICO DE PASAJEROS EN LA CIUDAD DE POPAYÁN



Juan Camilo Posso Ponce
Ricardo Andrés Vejarano García

Trabajo de grado en ingeniería electrónica y telecomunicaciones

Director:
Mary Cristina Carrascal Reyes
Msc. Ingeniería telemática

Universidad Del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Popayán, enero de 2019

Juan Camilo Posso Ponce
Ricardo Andrés Vejarano García

Sistema prototipo de recolección y gestión de datos para la
red de transporte público de pasajeros en la ciudad de
Popayán

Trabajo de grado presentado en la Facultad de Ingeniería
Electrónica y Telecomunicaciones de la
Universidad del Cauca para la obtención del
Título de

Ingeniero en:
Electrónica y Telecomunicaciones

Director:
Mary Cristina Carrascal Reyes
Msc. Ingeniería telemática

Popayán
2019

Agradecimientos de Juan Camilo Posso Ponce

Agradezco inicialmente a Dios, a quien pido de sabiduría, discernimiento y paciencia, en cada una de las actividades que realizo, estoy eternamente en deuda con mis padres quienes me han brindado su ayuda y apoyo en todo lo que me propongo, a mi directora Mary Cristina y mi compañero de proyectos y amigo, Ricardo Vejarano, quienes pusieron todo su empeño para hacer posible este trabajo de grado.

Agradezco a cada una de las personas que brindaron un aporte, idea o buenos deseos en este proceso.

Agradecimientos de Ricardo Vejarano

Agradezco a Dios y a mis padres por siempre haberme acompañado en este proceso, por estar ahí en cada paso, a ellos les dedico éste y cada uno de mis triunfos. Del mismo modo, agradezco a mi amigo y compañero de proyecto quién estuvo siempre dispuesto a dar lo mejor en cada etapa y mis amigos, los cuales estuvieron ahí siempre aportando sus mejores ideas y deseos.

Gracias a la Ingeniera Mary Cristina, quien estuvo dispuesta siempre a darnos los mejores aportes, correcciones y dirección del proyecto.

Contenido

Agradecimientos de Juan Camilo Posso Ponce	i
Agradecimientos de Ricardo Vejarano.....	iii
Contenido.....	iv
Lista de Ilustraciones.....	i
Lista de tablas	viii
Lista de ecuaciones	x
Listado de Acrónimos.....	xi
Introducción.....	1
Objetivos.....	3
Objetivo general	3
Objetivos específicos	3
Estado del arte.....	3
Capítulo 1	8
CARACTERIZACIÓN DE LA ZONA.....	8
1.1. Popayán como ciudad.....	8
1.2. Movilidad urbana en Popayán.....	10
1.3. Sistema actual de transporte público colectivo Popayán.	12
1.4. Necesidades a satisfacer del sistema	16
1.5. Descripción del sistema prototipo.....	16
1.6. Requerimientos del sistema	18

1.6.1. Requerimientos funcionales	18
1.6.1.1. Requerimientos funcionales AWGI	18
1.6.1.2. Requerimientos funcionales DMC.....	19
1.6.2. Requerimientos no Funcionales	20
1.6.2.1. Requerimientos no funcionales AWGI	20
1.6.2.2. Requerimientos no funcionales DMC.....	20
Capítulo 2.....	22
DISEÑO	22
2.1. Diagramas del proyecto.....	22
2.1.1. Diagrama de Casos de Uso.....	23
2.1.1.1. Descripción casos de uso Extendidos.....	24
2.2. Diagramas de estados	32
2.2.1. Diagrama de estados DMC.....	33
2.2.2. Diagrama de estados AWGI.....	33
2.3. Diagramas de actividades.....	33
2.3.1. Visualización de datos de conteo en la AWGI.....	34
2.3.2. Gestión de usuarios en la AWGI	34
2.3.2. Conteo de pasajeros	35
2.4. Diagrama de clases.....	36
2.5. Diagrama de componentes.....	36
2.5.1. Diagrama de componentes para el DMC	37
2.5.2. Diagrama de componentes para la AWGI.....	37
2.6. Diagrama de despliegue	38
2.7. Sistema prototipo como aplicación distribuida	38
Capítulo 3.....	39
IMPLEMENTACIÓN.....	39
3.1. Análisis de tecnologías disponibles para el conteo de pasajeros.....	39
3.1.1. Tecnologías para el control de acceso	39
3.1.1.1. NFC	40
3.1.1.2. MIFARE	42
3.1.2. Revisión de dispositivos tecnológicos afines	44

3.1.2.1.	Rapsberry	45
3.1.2.1.1.	Modulo cámara V2 Rapsberry.....	47
3.1.2.1.2.	Kinect.....	47
3.1.2.1.3.	Sensores infrarrojos	49
3.1.2.2.	Smartphones	50
3.1.2.2.1.	Android.....	51
3.1.2.2.2.	iOS	52
3.2.	Evaluación y selección de la tecnología.....	53
3.2.1.	Evaluación y selección de la tecnología	53
3.2.1.1.	Criterios de evaluación.....	53
3.2.1.2.	Tecnología Seleccionada	57
3.3.	Patrón de diseño	57
3.4.	Sistema de control de versiones	58
3.5.	Segmentación por prototipos	58
3.6.	Prototipo 1	59
3.6.1.	Modelamiento	59
3.6.1.1.	Diseño de interfaz AWGI	61
3.6.1.2.	Diseño de interfaz DMC.....	70
3.6.1.3.	Selección de base de datos	71
3.6.2.	Implementación AWGI	73
3.6.2.1.	Tecnología de desarrollo	73
3.6.2.2.	Conexión de Firebase con AWGI	74
3.6.2.3.	Creación de componentes y estructura	75
3.6.2.4.	Routing	75
3.6.2.5.	Persistencia	75
3.6.2.6.	Servicios.....	75
3.6.2.7.	Crud	76
3.6.2.7.1.	Componente formulario	77
3.6.2.7.2.	Componente tabla	79
3.6.2.8.	Perfil de usuario	81
3.6.2.9.	Sección de gráficas	82
3.6.2.9.1.	Búsqueda para un solo día.....	83

3.6.2.9.2.	Búsqueda para un rango de días.....	90
3.6.2.10.	Sección de mapa de calor y gráfica auxiliar	90
3.6.3.	Implementación DMC	94
3.6.3.1.	Tecnología de desarrollo	94
3.6.3.2.	Creación de proyecto y adición de dependencias.....	95
3.6.3.3.	Estructura del proyecto.....	96
3.6.3.4.	Conexión de Firebase con DMC	97
3.6.3.5.	Permisos de archivo manifest	98
3.6.3.6.	Actividad de inicio de sesión	98
3.6.3.6.1.	<i>Layout</i> de inicio de sesión	98
3.6.3.6.2.	Clase Kotlin de inicio de sesión	99
3.6.3.7.	Actividad main.....	101
3.6.3.7.1.	<i>Layout</i> de pantalla main.....	101
3.6.3.7.2.	Clase Kotlin de actividad main	101
3.6.3.8.	Actividad de conteo simulado.....	103
3.6.3.8.1.	<i>Layout</i> de conteo simulado.....	103
3.6.3.8.2.	Clase Kotlin del conteo simulado	104
3.7.	Prototipo 2.....	106
3.7.1.	OpenCV.....	107
3.7.2.	Algoritmo de detección	107
3.7.2.1.	Rasgos de clasificación	108
3.7.2.2.	Imagen integral	109
3.7.2.3.	Organización en cascada de los clasificadores.....	111
3.7.2.4.	Entrenador Clasificador Haar-cascade	112
3.7.3.	Preparación del entorno de desarrollo.....	114
3.7.4.	<i>Layout</i> de la pantalla de conteo	114
3.7.5.	Clase Java de la pantalla de conteo	114
3.7.5.1.	Asignación de tipo de frames a variables de clase Mat	117
3.7.5.2.	Dibujo de líneas delimitadoras de zonas.....	117
3.7.5.3.	Dibujo de variables que indican estado de zonas	118
3.7.5.4.	Detección.....	119
3.7.5.5.	Seguimiento	120

3.7.5.6. Protección contra obstrucciones.....	124
3.8. Prototipo Final	124
3.8.1. Integración	125
3.8.1.1. Migración de clase Java a Kotlin del algoritmo de detección y seguimiento.....	125
3.8.1.2. Integración total de funcionalidad	126
Capítulo 4.....	128
PRUEBAS Y RESULTADOS.....	128
4.1. Prototipo 1	128
4.1.1. Descripción de las pruebas	128
4.1.2. Análisis de resultados	138
4.2. Prototipo 2	139
4.2.1. Descripción de las pruebas	139
4.2.2. Análisis de resultados	145
4.3. Prototipo Final.....	145
4.3.1. Pruebas posicionamiento DMC.....	146
4.3.2. Pruebas funcionales DMC.....	147
4.3.3. Análisis de resultados	153
Capítulo 5.....	155
CONCLUSIONES Y TRABAJOS FUTUROS	155
5.1. Conclusiones	155
5.2. Lecciones Aprendidas.....	157
5.3. Trabajos futuros.....	157
Referencias	159
Anexo A: Cobertura de las empresas del servicio de TPC.....	164
Anexo B: Sistema de control de versiones.....	168
Anexo C: Creación de componentes	169
Anexo D: Routing	172
Anexo E: Persistencia.....	174
Anexo F: Búsqueda por rango de días.....	175
Anexo G: Preparación del entorno de desarrollo.....	180
Compilar SDK.....	180

Agregar módulo OpenCV.....	182
Instalar APK.....	183
Anexo H: Definición del concepto de observable	185
Anexo I: Datos Recolectados Prototipo 1	187
Anexo J: Pruebas ascenso/descenso prototipo 2.....	197
Anexo k: Pruebas Secundarias Finales	199

Lista de Ilustraciones

Ilustración 1 - Distribución de Popayán en Comunas.....	9
Ilustración 2 - Participación de Empresas en el sistema TPC	13
Ilustración 3 - Cobertura espacial a 200 m de las rutas en el área urbana	13
Ilustración 4 - Esquema de alto nivel del sistema prototipo.....	17
Ilustración 5 - Diagrama de casos de uso del proyecto.....	23
Ilustración 6 - Alerta problema de conexión	25
Ilustración 7 - Alerta datos no encontrados	26
Ilustración 8 - Alerta datos incorrectos	27
Ilustración 9 - Alerta usuario existente	27
Ilustración 10 - Alerta bus existente	30
Ilustración 11 - Alerta ruta existente	30
Ilustración 12 - Diagrama de estados DMC.....	33
Ilustración 13 - Diagrama de estados AWGI	33
Ilustración 14 - Diagrama de secuencia para visualizar información de conteo	34
Ilustración 15 - Diagrama de secuencia para operaciones básicas con usuarios	34
Ilustración 16 - Diagrama de secuencia para el conteo.....	35
Ilustración 17 - Diagrama de clases del proyecto.....	36
Ilustración 18 - Diagrama de componentes para el DMC	37
Ilustración 19 - Diagrama de componentes para la AWGI	37
Ilustración 20 - Diagrama de despliegue	¡Error! Marcador no definido.
Ilustración 21 - Pago con móvil NFC	41
Ilustración 22 - Tarjeta MIFARE	42
Ilustración 23 - Características MIFARE	43
Ilustración 24 - Raspberry Pi 3 Modelo B	46
Ilustración 25 - Modulo cámara Raspberry V2.	47
Ilustración 26 - Microsoft Kinect Sensor.	48
Ilustración 27 - Diseño típico de un Smartphone.....	50

Ilustración 28 - Pila de software de Android	51
Ilustración 29- Capas de abstracción IOS	52
Ilustración 30 - Modelo vista controlador	58
Ilustración 31 - Segmentación por prototipos del sistema	59
Ilustración 32 - Evento principal de los módulos con respecto al conteo de pasajeros	60
Ilustración 33 - Botones simulados de ascenso y descenso de pasajeros.....	61
Ilustración 34 - Propuesta en papel en alto nivel.....	62
Ilustración 35 - Inicio de sesión	63
Ilustración 36 - Barra de navegación.....	63
Ilustración 37 - Pagina home.....	64
Ilustración 38 - Página de gráficas	65
Ilustración 39 - Página de mapa.....	66
Ilustración 40 - Página de administradores	68
Ilustración 41 - Paginas de gestión de empleados, conductores rutas y buses.	69
Ilustración 42 - Editar perfil.....	70
Ilustración 43 - Inicio de sesión DMC	71
Ilustración 44 - Información básica pre conteo DM	71
Ilustración 45 - Firebase funcionalidades	72
Ilustración 46 - Arquitectura de Angular	74
Ilustración 47 - Configuración Firebase AWGI	75
Ilustración 48 - Etiqueta Inyectable de un servicio	76
Ilustración 49 - Estructura de componentes para paginas CRUD	76
Ilustración 50 - Inyectando componentes hijos.....	77
Ilustración 51 - NgModel en un input.....	77
Ilustración 52 - Función registrar ruta.....	78
Ilustración 53 - Función insertar ruta en servicio	78
Ilustración 54 - Definición de variable tipo AngularFireList.....	78
Ilustración 55 - Asignación de la variable routesList.....	78
Ilustración 56 - Ruta creada en Firebase	79
Ilustración 57 - Función actualizar ruta en servicio.....	79
Ilustración 58 - ngFor de Angular	79
Ilustración 59 - Clase Routes	80
Ilustración 60 - Función snapshotChanges	80
Ilustración 61 - Función para cambiar contraseña.....	81
Ilustración 62 - Carpeta que contiene fotos de perfil en firebase.....	82
Ilustración 63 - Fotos de perfil con identificador único.....	82
Ilustración 64 - Función para guardar imagen de perfil en firebase.....	82
Ilustración 65 - Selector de búsqueda de ruta	83
Ilustración 66 - Función para obtener buses	84

Ilustración 67 - Lista counter en firebase.....	85
Ilustración 68 - Lista completa de abordajes desde Firebase	85
Ilustración 69 - Función para encontrar índice de coincidencia.....	85
Ilustración 70 - Función para obtener abordajes en fecha específica	86
Ilustración 71 - Jerarquía de counter.....	86
Ilustración 72 - Componente tabla dentro del padre.....	86
Ilustración 73 - Declaración de variables inyectadas en componente	87
Ilustración 74 - Template componente tabla	87
Ilustración 75 - Componente grafica inyectado desde el componente padre	87
Ilustración 76 - Template de componente grafica.....	88
Ilustración 77 - Suscripción a observable para renderizar gráfica.....	88
Ilustración 78 - Creación de vectores para dependencia de graficas 1/2	89
Ilustración 79 - Creación de vectores para dependencia de graficas 2/2	89
Ilustración 80 - Creación de objeto barChartData	90
Ilustración 81 - Dos componentes hijos llamados en el template del componente padre	90
Ilustración 82 - Función en servicio para traer coordenadas de una búsqueda	91
Ilustración 83 - Creación de vectores para grafica de línea	91
Ilustración 84 - Objetos para renderización de gráfica	91
Ilustración 85 - Código fuente de librería heat-map.....	92
Ilustración 86 – Inclusión del código de librería en archivo angular.json.....	92
Ilustración 87 – Inclusión del script del código fuente de la librería en el index de la aplicación	92
Ilustración 88 - Template de componente hijo mapa.....	92
Ilustración 89 - Clase 1 para mapa de calor	92
Ilustración 90 - Clase 2 para mapa de calor	93
Ilustración 91 - Declaración de variables para mapa de calor	93
Ilustración 92 - Definición del centro de mapa de calor.....	93
Ilustración 93 - Opciones generales de mapa de calor	93
Ilustración 94 - Agregar la información para generar mapa de calor.....	94
Ilustración 95 - Creación de objeto legible para librería de mapa de calor	94
Ilustración 96 - Dependencias necesarias para DMC prototipo 1.....	96
Ilustración 97 - Estructura de carpetas del DMC prototipo 1	97
Ilustración 98 - Archivo de configuración de Firebase para aplicación móvil	97
Ilustración 99 - Permisos de cámara y GPS android.....	98
Ilustración 100 - Etiqueta de ConstraintLayout.....	98
Ilustración 101 - Login en ConstraintLayout	99
Ilustración 102 - Función de inicio de sesión DMC prototipo 1.....	99
Ilustración 103 - Extensión para obtener texto de un campo de tipo EditText de Android	100

Ilustración 104 - Clase UserSession DMC	100
Ilustración 105 - Layout Main Activity	101
Ilustración 106 - Inicialización de referencias de base de datos de firebase	101
Ilustración 107 - Función de recuperación de información de conductor	102
Ilustración 108 - Función para verificar existencia de un path en base de datos	102
Ilustración 109 - Función de iniciar conteo DMC	103
Ilustración 110 - Path de conteo de un día	103
Ilustración 111 - Pantalla de conteo simulado	104
Ilustración 112 - Función para obtener valores de conteo total y parcial.....	104
Ilustración 113 - Función para escribir en la base de datos sobre un evento de descenso del vehículo.....	105
Ilustración 114 - Función para escribir en la base de datos sobre un evento de ascenso al vehículo	105
Ilustración 115 - Creación de un request para Localización.....	106
Ilustración 116 - Callback para la localización.....	106
Ilustración 117 - Ejemplos de rasgos de 2 (a y b), 3 (c) y 4 (d) rectángulos utilizados para realizar la detección de objetos.....	108
Ilustración 118 - dos rasgos de clasificación y su correspondencia con partes de un rostro.....	108
Ilustración 119 - Cálculo de la imagen integral. (a) Imagen original (b) Imagen integral (c) Valor del rectángulo utilizando la imagen.....	110
Ilustración 120 - cálculo del valor de un rasgo y del vector W de pesos del rasgo..	111
Ilustración 121 - Cascada de clasificadores.....	111
Ilustración 122- Imagen positiva entrenamiento de clasificador	113
Ilustración 123 - Imagen negativa entrenamiento de clasificador.....	113
Ilustración 124 - Layout de la actividad de captura	114
Ilustración 125 - Declaración de clase OpenCvController	115
Ilustración 126 - Clase BaseLoaderCallback 1/2.....	115
Ilustración 127 - Clase BaseLoaderCallback 2/2.....	116
Ilustración 128 - Permisos de uso de cámara archivo manifest	116
Ilustración 129 - Fragmento onCreate actividad de conteo	116
Ilustración 130 - Función onCreateViewStarted	117
Ilustración 131 - Definiendo valores de variables del tipo Mat.....	117
Ilustración 132 - Dibujo de zonas de decisión	118
Ilustración 133 - Disposición de zonas para seguimiento.....	118
Ilustración 134 – Dibujo de contadores de zonas.....	119
Ilustración 135 - Sección de detección.....	120
Ilustración 136 - Clase PersonCoordinate	120
Ilustración 137 - Ciclo for para el seguimiento.....	121
Ilustración 138 - Dibujo de punto central para detección del perfil izquierdo.....	121

Ilustración 139 - Dibujo del punto central para la detección del perfil derecho.....	121
Ilustración 140 - Variable auxiliar de seguimiento	121
Ilustración 141 - Componentes horizontales y verticales de variables a comparar .	122
Ilustración 142 - Evaluación de distancias	122
Ilustración 143 - Refrescar vector principal en caso de ruido	122
Ilustración 144 - Evaluación de zonas para el perfil izquierdo.....	123
Ilustración 145 - Función para el conteo de descenso de pasajeros.....	123
Ilustración 146 - Script de protección contra obstrucciones	124
Ilustración 147 - Evaluación del vector contours	124
Ilustración 148 - Definición de la clase OpenCVActivity en Kotlin	125
Ilustración 149 - Carga de Haar Cascade Kotlin	125
Ilustración 150 - Sección de detección Kotlin	126
Ilustración 151 - Función para insertar descenso.....	126
Ilustración 152 - Función para insertar ascenso.....	127
Ilustración 153 - Rutas creadas en la AWGI - Pruebas prototipo 1	129
Ilustración 154 - Buses creados en la AWGI - Pruebas prototipo 1.....	129
Ilustración 155 - Conductores creados en la AWGI - Pruebas prototipo	129
Ilustración 156 - Interfaz principal DMC prototipo 1.....	130
Ilustración 157 - Interfaz de simulación de conteo de pasajeros DMC prototipo 1..	130
Ilustración 158 - Mapa ciudad de Buga - Valle del cauca.....	131
Ilustración 159 - Recorrido Ruta 1.....	132
Ilustración 160 - Recorrido ruta 2	132
Ilustración 161 - Captura de datos Conductor 1- bus GXS-326 - Ruta 1.....	133
Ilustración 162 – Conteo Parcial de pasajeros vs Hora – Conductor 1 – Ruta 1	134
Ilustración 163 - Captura de datos Conductor 2- bus DTY-346 - Ruta 2	135
Ilustración 164 - Conteo Parcial de pasajeros vs Hora - Conductor 2 - Ruta 2	135
Ilustración 165 - Grafica rango de días para bus y ruta seleccionada.....	136
Ilustración 166 - Archivos descargadas de tablas AWGI.....	137
Ilustración 167 - Grafica conteo total de pasajeros bus seleccionado por hora.....	137
Ilustración 168 - Interfaz principal.....	138
Ilustración 169 - Prueba detector de rostro frontal y ojos	139
Ilustración 170 - Prueba de detección y tracking.....	141
Ilustración 171 - Prueba detector oreja derecha.....	142
Ilustración 172 - Prueba final 1 prototipo 2.....	143
Ilustración 173- Prueba final 2 prototipo 2.....	143
Ilustración 174 - Prueba final 3 prototipo 2.....	144
Ilustración 175 - Prueba final 4 prototipo 2.....	144
Ilustración 176 - Prueba final 5 prototipo 2.....	145
Ilustración 177 - Dispositivo de alojamiento DMC	146
Ilustración 178 - Dispositivo de alojamiento DMC.....	146

Ilustración 179 - Posición seleccionada DMC.	147
Ilustración 180 - Interfaz login DMC.	148
Ilustración 181- Interfaz principal DMC	148
Ilustración 182 - Prueba final Down 1.....	149
Ilustración 183 - Prueba final Down 2.....	149
Ilustración 184 - Prueba final Down 3.....	150
Ilustración 185 Prueba final Down 4.....	150
Ilustración 186 - Registro en base de datos evento Down	151
Ilustración 187 - Prueba final Up 1.	151
Ilustración 188 - Prueba final Up 2.	152
Ilustración 189 - Prueba final Up 3.	152
Ilustración 190 - Cobertura Empresa Transpubenza	164
Ilustración 191 - Cobertura Empresa Translibertad	165
Ilustración 192 - Cobertura Empresa Sotracauca Metro	166
Ilustración 193- Cobertura Empresa Transtambo.....	167
Ilustración 194 - Estructura de carpetas AWGI	169
Ilustración 195 - Selector nav-bar component.....	170
Ilustración 196 - Template app.component.ts.....	170
Ilustración 197 - Definición de un Behavior Subject	170
Ilustración 198 - Evento de cambio para un Behavior Subject	170
Ilustración 199 - Suscripción a un Behavior Subject	171
Ilustración 200 - Importar módulo de routing	172
Ilustración 201 - Archivo de enrutamiento para el componente de perfil de usuario	172
Ilustración 202 - LocalStorage en Angular	174
Ilustración 203 - Recuperar valor LocalStorage	174
Ilustración 204 - Decisión de búsqueda por rango o diaria	175
Ilustración 205 - Algoritmo para generar array de fechas en rango de días.....	175
Ilustración 206 - Adaptación de función para encontrar índice de coincidencia con iteración.....	176
Ilustración 207 - Servicio para obtener listas de abordajes	176
Ilustración 208 - Función para obtener abordajes por rango de días 1/2	177
Ilustración 209 - Lista de coincidencia dentro de counter.....	177
Ilustración 210 - Última posición de listBoardingsInDay en una iteración	178
Ilustración 211 - Inyectando vectores para gráfica por rango de días	178
Ilustración 212 - Renderización de grafica por rango de días	179
Ilustración 213 - Selección de MinGW Makefiles	181
Ilustración 214 - Generando makefiles.....	181
Ilustración 215 - Comando mingw-32-make	182
Ilustración 216 - Importando módulo OpenCV en Android	182
Ilustración 217 - Agregando dependencia de la librería de OpenCV.....	183

Ilustración 218 - Módulo OpenCV agregado	183
Ilustración 219 - Apk OpenCV Manager	184
Ilustración 220 - Observable o consumidor	185
Ilustración 221 - Métodos de un observer	186
Ilustración 222 - Prueba posición lado izquierdo sin inclinación.....	199
Ilustración 223 - Prueba posición perpendicular.....	200
Ilustración 224 - Prueba posición lado derecho sin inclinación.	200
Ilustración 225 - Prueba posición lado derecho inclinación 20°	201

Lista de tablas

Tabla 1 - Rutas y Flota servicio TPC por empresa.....	12
Tabla 2 - Características de los vehículos.....	14
Tabla 3 - Puntos de despacho y rutas.....	14
Tabla 4 - Flota en operación en día corriente.....	14
Tabla 5 - Caso de uso extendido Iniciar/Detener conteo.....	24
Tabla 6 - Caso de uso extendido visualizar datos.....	25
Tabla 7 - Caso de uso extendido gestionar empleados.....	26
Tabla 8 - Caso de uso extendido gestionar administradores.....	27
Tabla 9 - Caso de uso extendido gestionar conductores.....	28
Tabla 10 - Caso de uso extendido gestionar buses.....	29
Tabla 11 - Caso de use extendido gestionar rutas.....	30
Tabla 12 - Caso de uso extendido gestionar perfil.....	31
Tabla 13 - Caso de use extendido validar datos.....	32
Tabla 14 - Costos tecnologías.....	54
Tabla 15 – criterios de calificación de tecnologías con respecto a la tasa de éxito...	56
Tabla 16 - Puntajes criterios de selección por tecnología.....	57
Tabla 17 - Campos de entrada para formulario de creación de usuarios administradores.....	67
Tabla 18 - Campos requeridos para creación de conductores.....	68
Tabla 19 - Campos requeridos para la creación de rutas.....	69
Tabla 20 - Campos requeridos para la creación de buses.....	69
Tabla 21 - Características plan gratis de uso de Firebase.....	73
Tabla 22 – Componentes de la sección de gráficas.....	82
Tabla 23 - Atributos de la clase bus.....	83
Tabla 24 - Requerimientos para compilar SDK OpenCV para Android.....	180
Tabla 25 -Datos capturados Conductor 1- GXS-326 - Ruta 1.....	187
Tabla 26 -Datos capturados Conductor 2- bus DTY-346 - Ruta 2.....	189

Tabla 27 - Datos rango de dias 24 al 26 de diciembre 2018 - Conductor 1 - bus GXS-326 -Ruta 1.....	193
--	-----

Lista de ecuaciones

Ecuación 1 - Representación integral de una imagen.....	109
Ecuación 2 - Recorrido de una imagen 1	110
Ecuación 3 - Recorrido de una imagen 2	110
Ecuación 4 - Condiciones para la suma de una fila.....	110

Listado de Acrónimos

API	<i>Application Programming Interface</i> , Interfaz de Programación de Aplicaciones.
APK	<i>Android Application Package</i> , Paquete para el Sistema Operativo Android.
AWGI	Aplicación Web para la Gestión de la Información.
BID	Banco Interamericano de Desarrollo.
BSD	<i>Berkeley Software Distribution</i> , Distribución de Software Berkeley.
BTR	Buses de Tránsito Rápido.
CLI	<i>Command Line Interface</i> , Interfaz de Línea de Comandos
COP	<i>Colombian Pesos</i> , Pesos Colombianos.
CSI	<i>Camera Serial Interface</i> , Interfaz Serial de Cámara.
CSV	<i>Comma Separated Values</i> , Valores Separados por Coma.
DMC	Dispositivo Móvil de Captura.
DSI	<i>Display Serial Interface</i> , Interfaz Serial de Pantalla.
ETSI	<i>European Telecommunications Standards Institute</i> , Instituto Europeo de Normas de Telecomunicaciones.
GPIO	<i>General Purpose Input/Output</i> , Entrada/Salida de uso General.
GPS	<i>Global Positioning System</i> , Sistema de Posicionamiento Global.
HSV	<i>Hue Saturation Value</i> , Matriz Saturación Valor.
ISO	<i>International Organization for Standardization</i> , Organización Internacional para la Estandarización.
I ² C	<i>Inter Integrated Circuit</i> , Circuito Inter-Integrado.
LCD	<i>Liquid Cristal Display</i> , Pantalla de Cristal Líquido.
MVC	<i>Model View Controller</i> , Modelo Vista Controlador.
NFC	<i>Near Field Communication</i> , Comunicación de Campo Cercano.
pH	Potencial de Hidrogeniones.

SBC	<i>Single Board Computer</i> , Ordenador de Placa Única.
SETP	Sistema Estratégico de Transporte Público.
SCA	Sistema de Control de Acceso.
SO	Sistema Operativo.
SPA	<i>Single Page Application</i> , Aplicación de Página Única.
SPI	<i>Serial Peripheral interface</i> , Interfaz Periférica Serial.
SDK	<i>Software Development Kit</i> , Kit de Desarrollo de Software.
TFT	<i>Thin-Film Transistor</i> , Transistor de Película Delgada.
TPC	Transporte Público Colectivo.
TPCU	Transporte Público Colectivo Urbano.
UART	<i>Universal Asynchronous Receiver – Transmitter</i> , Receptor – Transmisor Asíncrono Universal.
URL	<i>Uniform Resource identifier</i> , Identificador Uniforme de Recursos.
VGA	<i>Video Graphics Array</i> , Matriz de Gráficos de Video.
WAP	<i>Wireless Application Protocol</i> , Protocolo de Aplicaciones Inalámbricas.
WIFI	<i>Wireless Fidelity</i> , Fidelidad Inalámbrica.

Introducción

El hombre es un ser social por naturaleza, por ello desde el inicio de su evolución se ha congregado con sus iguales para dividir roles y facilitar su supervivencia [1]. A la congregación de personas que viven juntas bajo ciertas reglas o que tienen intereses en común, se le conoce como comunidades. Un tipo de comunidad, son las ciudades, las cuales se configuran como el lugar donde las personas viven, trabajan y se desarrollan personal y laboralmente [2]. Las ciudades en la actualidad no paran de crecer y evolucionar, debido a la ampliación de los tejidos urbanos, tanto en extensión territorial como en densidad poblacional, por esto han surgido diferentes necesidades para sus integrantes, las cuales abarcan desde los aspectos ambientales hasta los socioeconómicos [3].

Las actividades realizadas por las personas que conforman una ciudad; las cuales implican salir de su vivienda; demandan el uso de diferentes medios de desplazamiento como: caminata, transporte mecanizado (bicicleta) o motorizado (autobuses, motocicletas, automóviles, ferrocarriles y metro) [4]. Por ello la movilidad representa uno de los pilares fundamentales dentro de las ciudades, dado a que tiene una incidencia directa tanto en el desarrollo económico de la región, como en la calidad de vida de las personas que la conforman [5].

Las ciudades tienen el reto de convertirse en ciudades sostenibles. Capaces de ofrecer una alta calidad de vida a sus habitantes, facilitándoles el desarrollo y satisfacción de necesidades físicas (salud, seguridad), materiales (vivienda, ingresos, transporte, pertenencias, comida), sociales (de trabajo, familia, relaciones personales, comunidad, responsabilidades), psicológicas y/o emocionales (afecto, autoestima, inteligencia emocional, espiritualidad, religión), de desarrollo (educación, productividad) y ecológicas (calidad del agua, del aire, etc.) [6]. Además, éstas deben centrarse en reducir el impacto al medio ambiente y contar con un gobierno local que tenga capacidad fiscal y administrativa para mantener su crecimiento económico y llevar a cabo sus funciones urbanas con una amplia participación ciudadana [7].

La movilidad es uno de los factores que debe evolucionar a la par del crecimiento de las ciudades si esta quiere ser sostenible en este aspecto, y así evitar dificultades que puedan afectar a sus habitantes. Puntualmente, Popayán tiene una densidad poblacional moderada, sin embargo, se ve afectada por varias de las problemáticas presentes en los grandes centros urbanos, como lo son: la masiva concentración vehicular en horas pico, pocas vías alternas para descongestionar el tráfico automotor, escases de vías peatonales, frecuentes accidentes de tránsito, contaminación ambiental producida por vehículos obsoletos, diferentes problemáticas en el sistema de transporte público, malla vial deteriorada, asimismo; se presentan fenómenos como la guerra del centavo, el mototaxismo, la falta de civismo y cultura ciudadana para respetar las normas y señales de tránsito, escases de ciclo vías, entre otras problemáticas [8].

Desde junio de 2016 y en virtud del reto de convertirse en una ciudad sostenible, la ciudad de Popayán se embarcó en un ejercicio de planificación que conllevó varios meses de trabajo, con el fin de ingresar a la iniciativa Ciudades Emergentes y Sostenibles del BID (Banco Interamericano de Desarrollo) [9], lo que trajo consigo estudios como el Plan Maestro de Acueducto y Alcantarillado, y estudios relacionados con el Plan Maestro del uso de Espacio público. Finalmente, Popayán fue elegido para hacer parte de la iniciativa y trabajar de la mano de Findeter para el diseño y puesta en marcha de proyectos relacionados con mejorar la calidad de vida de sus habitantes, las plazas de mercado, la malla vial, apoyo a procesos de emprendimiento, estrategias para mitigar los efectos del cambio climático, sostenibilidad ambiental urbana, y fiscal de la ciudad, y mejoras en el sistema de transporte público colectivo [10].

Un aspecto que ha influido de forma directa en el reto para alcanzar una movilidad sostenible, es la globalización, entendida como el proceso histórico de integración mundial en los ámbitos político, económico, social, cultural y tecnológico, con lo cual ha convertido al mundo en un lugar cada vez más interconectado [11]. La influencia de este proceso en dicho reto, se debe a los adelantos tecnológicos que conllevó, lo cuales han transformado diversas áreas desde el acceso a la información (internet, medios de comunicación) hasta la forma como se desplazan las personas, por lo cual ha influido de forma directa en la evolución de la movilidad.

El servicio de transporte público colectivo, es una de las formas de desplazamiento más utilizadas actualmente, por ello, representa un factor clave para la movilidad dentro de las ciudades [12], teniendo en cuenta las necesidades de desplazamiento de los habitantes. Con el incremento poblacional, debe brindarse este servicio a mayor número de personas, de forma rápida y efectiva. Es aquí donde los avances tecnológicos traídos por la globalización y la evolución de la tecnología han sido de gran ayuda para encontrar alternativas y herramientas que aporten a la mejora del sistema y a las crecientes necesidades que este presenta, generando beneficios para todos los involucrados con la prestación de este servicio.

Movilidad Futura S.A.S es la empresa encargada de implementar el SETP (Sistema Estratégico de Transporte Público) en la ciudad de Popayán [13], el cual busca por medio de diversas alternativas y un plan de acción definido, solventar las diferentes problemáticas presentes en el actual sistema TPC (Transporte Público Colectivo), por ello, desarrolló en el 2017 diferentes retos con el fin ofrecer una solución innovadora a dichos problemas. Uno de estos retos es un sistema que permita realizar el conteo de pasajeros con alta confiabilidad, precisión y bajo costo para ser utilizado dentro del proyecto SETP [14], ya que, una de las principales problemáticas del sistema es la ausencia de infraestructura y las herramientas tecnológicas necesarias para recolectar información verídica que pueda utilizarse por las empresas que brindan el servicio para llevar un control de éste, con lo cual se puedan identificar las falencias presentes en el servicio y darles solución de forma oportuna.

Buscado además de una solución al reto planteado por Movilidad Futura S.A.S, aplicar los conocimientos adquiridos desde el área de la electrónica y telecomunicaciones para generar

un aporte a la comunidad, se desarrolló para el presente trabajo de grado un prototipo tecnológico que cumpla con los objetivos planteados y que pueda ser incorporado al sistema de transporte público colectivo en la ciudad de Popayán, permitiendo la mejora de este servicio, y teniendo en cuenta las características específicas de la ciudad.

Objetivos

Objetivo general

Desarrollar un sistema prototipo que apoye la gestión del servicio de transporte público colectivo en la ciudad de Popayán por medio del conteo y registro de pasajeros.

Objetivos específicos

- Identificar la tecnología más adecuada para el conteo de pasajeros, teniendo en cuenta las características presentes en la ciudad de Popayán, así como la infraestructura del sistema de transporte público y los datos de interés que se deseen registrar para este servicio.
- Diseñar un sistema prototipo hardware para el conteo y registro de pasajeros.
- Desarrollar una aplicación web para la visualización de los datos de conteo y registro de pasajeros tomados por el prototipo hardware.
- Evaluar la funcionalidad del prototipo y la plataforma web mediante pruebas simulando el flujo de pasajeros.

Estado del arte

A continuación, se realiza un recorrido por los proyectos afines con el conteo de ascenso o descenso de pasajeros, y la gestión de información en sistemas que prestan el servicio de transporte público en el mundo.

Conteo de personas con un sensor RGBD comercial [15]

Revista Iberoamericana de Automática e Informática industrial, España 2014.

El presente trabajo inicialmente describe ciertas técnicas para el conteo de personas en diferentes entornos, haciendo claro énfasis en la utilidad que prestaría para el servicio de transporte público, la toma de datos de entrada y salida de pasajeros. Se toman en cuenta las principales tecnologías que se han usado para abordar el problema hasta la fecha de publicación de la revista: imágenes de los escenarios analizadas con técnicas de visión por computador y el análisis de las firmas que ofrecen diferentes haces de luz en presencia de personas.

El escenario y propuesta del trabajo consiste específicamente en el uso de una cámara Kinect RGB para contar las personas que cruzan una puerta, emulando escenarios de transporte público, sin restringir una serie de parámetros como condiciones de iluminación. A lo largo del desarrollo, se exponen los diferentes algoritmos y ecuaciones que permiten realizar tanto los modelados simples como los basados en combinaciones gaussianas, las cuales son usadas para el análisis, detección y seguimiento.

Una vez evaluados los resultados de la implementación del sistema en el escenario descrito, se llega a que la información de profundidad es una señal válida y sólida para abordar el problema del conteo de personas, con un acierto de un 95% en el movimiento de 258 personas. Los autores también afirman que los resultados son óptimos para entornos donde la afluencia de usuarios no es muy grande, por lo cual la implementación para un sistema de transporte público como el de la ciudad de Popayán puede ser adecuada.

Finalmente, los resultados de las pruebas realizadas a la tecnología son muy claros al encontrar que la tasa de aciertos en el conteo es de más del 85%, es decir, la tecnología es óptima dentro de los parámetros que se prometen, ya que la integración de más factores de análisis, bajan su porcentaje de acierto.

El trabajo analizado, aporta al nuestro, en cuanto al conocer una tecnología que ya se ha usado para el conteo de pasajeros en un ambiente que puede ser el de un autobús. Sin embargo, una limitación que se encuentra radica en que dicho sistema no es capaz por si solo de registrar información que sea valiosa para una empresa de transporte, así que sería necesario analizar más a fondo el módulo de gestión de información.

People counting system using raspberry pi with OpenCV [16]

International Journal for Research in Engineering Application & Management (IJREAM), Vol-02, Issue 01, APR 2016.

Este trabajo de grado describe la importancia del conteo de personas en diferentes entornos, como colegios, universidades, centros comerciales, edificios empresariales, entre muchos otros, resaltando que el conteo de personas permite hacerse una idea más real de lo que de verdad está sucediendo en el ambiente, conocer diferentes variables que ayuden desde la

gestión interna del sistema donde se implementa, hasta en el incremento de ventas o elementos externos que se deseen potenciar.

Esta tesis de maestría se enfoca en desarrollar un sistema para contar personas que pasan por una puerta utilizando un módulo de cámara Pi fijo, montaje realizado en la placa Raspberry Pi y en la herramienta de programación Python vinculada a la aplicación junto a la librería de OpenCV. Los resultados muestran que usar una cámara para contar personas es una buena alternativa a diferencia de otros sensores, sin embargo, se nota también que el sistema necesita muchas mejoras para ser realmente de confianza, ya que su fiabilidad no supera el 75%.

Diseño de un sistema automático para el control y acceso de persona a un autobús intermunicipal de transporte en Colombia [17]

Trabajo de grado Universidad de Boyacá, 2016

Este trabajo busca se diseñó de una manera enfocada a brindar una solución viable al problema que se viene presentando a nivel nacional en el concepto de transporte de pasajeros, ya que aunque el tema se ha tratado en proyectos anteriores, lo que se busca es mostrar una alternativa que resulte económica, entendible y manejable en todos los campos que abarca el tema del transporte en Colombia, esto quiere decir que el sistema que se va a implementar debe ser fácil de manejar por cualquier persona, no obstaculice al momento de permitir el acceso de pasajeros a los autobuses y permita tener un control para evitar fraudes en las tarifas.

El sistema propuesto captura todos los datos por medio de una red de sensores infrarrojos, que comercialmente se conoce como sensores Wall DA-20, estos sensores estarán conectados por una interfaz RS232 a una micro computadora que recogerá los datos, un sistema GPS (Global Positioning System) que capturará los datos de la ubicación de abordaje, además se diseñó un software en Matlab que permitirá realizar todas las funciones.

El proyecto se implementó, consiguiendo un alto grado de confiabilidad, aproximadamente del 92%, sin embargo, se encontró que el sistema ocupaba mucho espacio, dificultando el ingreso de las personas, además la información solo se concibió para la captura del día en curso para evitar fraudes en la recaudación de dineros, para un trabajo futuro se propone, crear un registro de esta información, además, de hacer este sistema menos invasivo para el usuario.

En este trabajo se puede observar el diseño de un sistema para el conteo de pasajeros, sin embargo, no aporta a temas de gestión de información ya que hace un manejo de datos diario de forma local, por otro lado, se encuentra que la fiabilidad del sistema es alta, pero sus costos al tener que implementar una micro computadora, una pantalla, una red de sensores y un GPS, resultan elevados, teniendo en cuenta que para el trabajo aquí propuesto se tiene una restricción en la cual la solución debe ser de bajo costo, alrededor de los 400.000 COP.

Análisis de viabilidad para implementar la tecnología NFC aplicada a medios de pago en sistemas de transporte masivo, en Colombia. [18]

Trabajo presentado a la empresa IPRELENZO SAS en la ciudad de Bogotá D.C, por la ingeniera en electrónica y de soporte técnico, Laura Henao Gámez.

En este documento se expone una posible solución a los problemas en los sistemas de movilidad de las ciudades capitales e intermedias por medio de la implementación de la tecnología NFC [19] como medio de pago, la cual, además de suplir el rendimiento de actuales tecnologías como las tarjetas MIFIRE [20], ofrece beneficios económicos como el ahorro en adquisición de plásticos e infraestructura para adecuar puntos de recarga.

En ciudades capitales como Bogotá o Medellín, tienen implementados sistemas de pago mediante tecnología MIFARE para servicios como Transmilenio, Metro línea, Megabus, Mío. En el contexto de Popayán, estos servicios no están implementados, ya que se cuenta únicamente con el funcionamiento del bus urbano, el cual carece de la infraestructura mencionada y debido a ello, se producen problemas como fraude, pues los recaudadores no llevan un conteo exacto de las personas, generándose así muchas pérdidas, estas problemáticas son las que se buscan solucionar con la implementación del STEP, el cual aún no tiene fija la tecnología que implementará.

Finalmente, en el trabajo se encuentra que la tecnología NFC trae muchas ventajas al ser aplicada en servicios como lo son los sistemas de movilidad, permitiendo gestionar más información que con tecnologías como la tarjeta MIFARE.

Si bien el documento analiza y propone el uso de la tecnología NFC, lo cual genera un aporte, al mostrar una de las posibles tecnologías a usar, se diferencia de este trabajo de grado en que el sistema a desarrollar va a permitir la gestión de la información.

La estructura del presente trabajo de grado se describe a continuación:

CAPÍTULO 1: CARACTERIZACIÓN DE LA ZONA

En este capítulo se indican las características particulares de la ciudad de Popayán donde se desarrolló el proyecto de grado, enfatizando en la movilidad y estado actual del sistema de transporte público colectivo dentro de ésta. Finalmente se realiza un levantamiento de los requerimientos para el desarrollo del sistema prototipo.

CAPÍTULO 2: DISEÑO

En este capítulo se plantean los esquemas y diagramas para el desarrollo del proyecto, con el fin de entender plenamente el comportamiento del mismo y poder iniciar con el proceso de la implementación del sistema.

CAPÍTULO 3: IMPLEMENTACIÓN

En este capítulo se estudian las tecnologías disponibles para la selección de aquella que va a apoyar el conteo de pasajeros. Una vez seleccionada, se realiza el proceso de implementación de todas las funciones que se requieren.

CAPÍTULO 4: PRUEBAS Y RESULTADOS

El capítulo 4 tiene plasmados las pruebas y resultados de los subprototipos y prototipo final del sistema, en él se explican a detalle los escenarios y restricciones que cada uno de los componentes tiene.

CAPÍTULO 5: CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se plasman los logros y objetivos alcanzados, enseñanzas aprendidas y próximas mejoras o trabajos futuros para el proyecto.

Capítulo 1

CARACTERIZACIÓN DE LA ZONA

Para el correcto desarrollo del presente trabajo de grado, se deben conocer las condiciones del entorno para el cual va a ser diseñado el prototipo, en este caso particular la ciudad de Popayán, por esto, en este capítulo se hace un recorrido por las principales características de esta ciudad iniciando desde lo más general como aspectos geográficos y demográfico hasta aspectos tan propios, como su movilidad, y el estado de su sistema TPC.

Finalmente, se realiza una captura de requerimientos teniendo en cuenta la caracterización y los actores que involucran la ciudad.

1.1. Popayán como ciudad.

Popayán es un municipio ubicado al suroeste del país, entre la sierra montañosa del oeste y la cordillera central de Colombia al este, es la capital del Departamento del Cauca y fue fundada el 13 de enero de 1537 por Sebastián de Belalcázar, sin embargo, es hasta el 15 de agosto de 1537 cuando se logra su conquista total, y se realiza su ceremonia de fundación, conservando su nombre indígena "Popayán" [21].

Popayán cuenta con una población aproximada de 282.453 habitantes para el año 2017 y se estima aumente a 288.411 para el año 2020 [22]. Su extensión territorial es de 512 km², con una altura de 1.737 msnm (medidos en la plazuela de la iglesia de San Francisco). Tiene una temperatura media de 18°C a 19 °C durante todo el año, alcanzando temperaturas máximas en los meses de julio, agosto y septiembre en horas del mediodía, hasta de 29 °C y mínimas de 10 °C en horas de la madrugada en verano [23].

La ciudad se encuentra actualmente dividida en 296 barrios agrupados en 9 comunas en el sector urbano, 75 veredas agrupadas en 23 corregimientos y 2 resguardos indígenas en el

sector rural, que se pueden evidenciar claramente por convención de colores en la Ilustración 1. El área metropolitana de Popayán está conformada por sí misma y los municipios de El Tambo, Timbío, Cajibío y Piendamó, que representan una población total de 439.257 habitantes [24]. Aunque la mayoría de la población está asentada en el área urbana, el 95% del total de territorio del municipio hace parte del área rural.

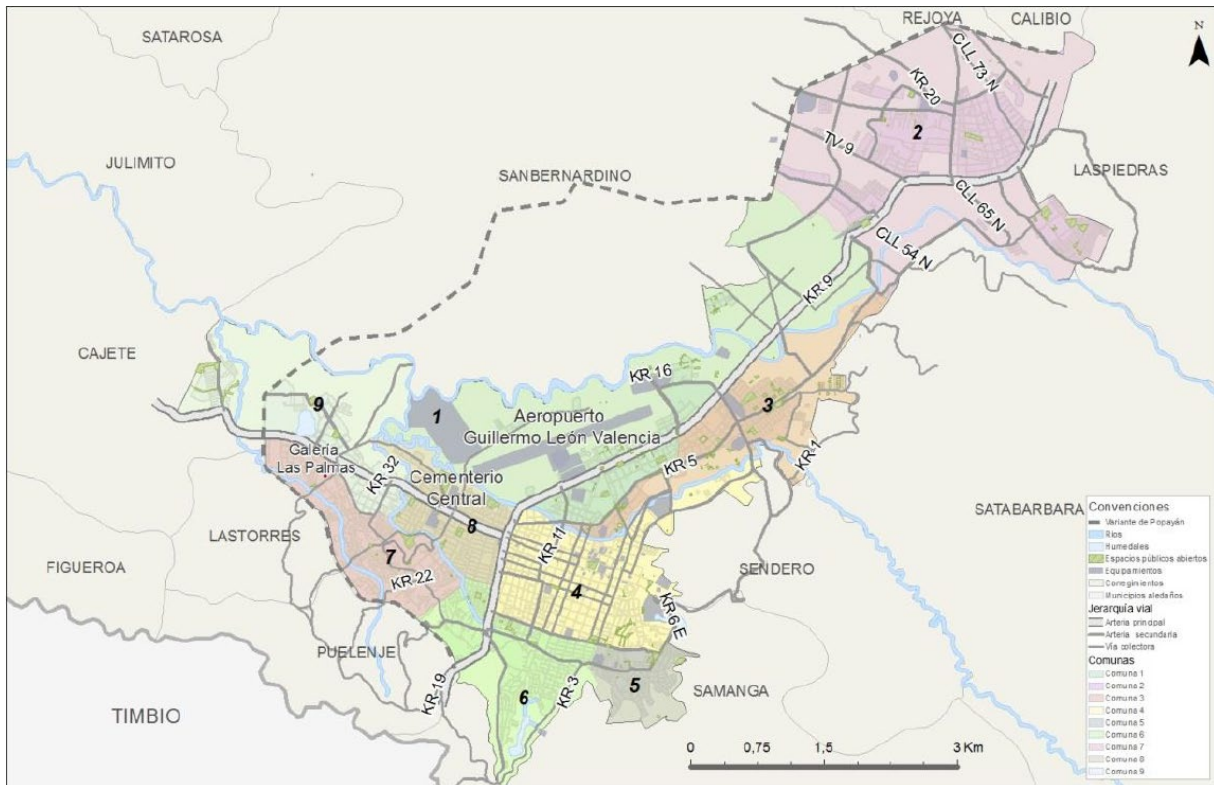


Ilustración 1 - Distribución de Popayán en Comunas
Tomado de: [25]

La comuna 1 está integrada principalmente por predios estrato 5 y algunos de estrato 6, con una densidad poblacional promedio de 4,6 viviendas y 48 habitantes por hectárea, que la convierten en el sector con el menor número de habitantes de la ciudad. Por otro lado, en la comuna 4 está ubicado el centro de la ciudad con 15 viviendas y 121 habitantes por hectárea, esta comuna se caracteriza por agrupar la mayor zona de servicios de la ciudad y el 50% de la oferta de los establecimientos de educación básica, secundaria, media vocacional y universitaria, haciendo que predominen en esta comuna los estratos 3 y 4.

La densidad poblacional más grande se encuentra en las comunas 5, 7 y 8. Por un lado, la comuna 5 tiene un promedio de 248 habitantes por hectárea. En la comuna 8, la mayoría de viviendas pertenecen a los estratos 2 y 3, contando además con un amplio sector comercial y de servicios. Por otro lado, la comuna 7 es la de mayor densidad poblacional de la ciudad con 16 viviendas y 265 habitantes por hectárea, predominando en la zona los estratos

socioeconómicos más bajos (1 y 2). Por último, la comuna 9 tiene una densidad de 197 habitantes por hectárea y como en la 7, prevalecen los estratos socioeconómicos 1 y 2.

Los estratos socioeconómicos más bajos de la ciudad (1 y 2) están generalmente localizados en las periferias del municipio, particularmente en la zona sur y oriental. Los estratos altos (5 y 6) se localizan en el sector occidental y noroccidental de la ciudad, particularmente en el borde occidental de la vía panamericana y en las inmediaciones del aeropuerto Guillermo León Valencia, mientras que los estratos medios (3 y 4) se localizan hacia la zona central y al interior de la zona urbana [25].

La distribución de la población en las diferentes comunas, así como su facilidad de acceso, han generado algunos de los problemas de movilidad dentro de la ciudad. Además, es importante resaltar que, Popayán puede ser considerada una ciudad centralizada donde la mayoría de sus instituciones gubernamentales, económicas y comerciales se encuentran en el centro histórico, que cuenta con aproximadamente 236 manzanas, convirtiéndose en uno de los centros coloniales más grandes del país y de América. Su arquitectura se ha conservado por más de 5 siglos, lo que constituye hoy en día uno de sus mayores atractivos turísticos.

La UNESCO ha otorgado dos importantes reconocimientos a la ciudad, el primero en el año 2009 dado que, desde épocas coloniales, Popayán ha sido un centro religioso importante, reflejándose en las procesiones de semana santa que fueron declaradas Obra Maestra del Patrimonio Oral e inmaterial de la Humanidad. En el segundo reconocimiento, la UNESCO declara a Popayán en 2005 como “Ciudad de la Gastronomía de la UNESCO”, hecho que se ve reflejado en la organización y celebración del festival gastronómico desde el año 2003 por parte de la Corporación Gastronómica de Popayán.

Por esa razón, Popayán se ha convertido en una ciudad de gran atractivo turístico, donde en temporada alta atrae a más de 65.000 personas [26], y debido a que una gran cantidad de actividades se desarrollan en su centro Histórico el cual no ha sufrido grandes cambios en los últimos siglos, se han generado problemas de movilidad en esta zona, así como en la ciudad en general, tanto por el turismo como por el incremento de la población.

1.2. Movilidad urbana en Popayán.

Son consideradas ciudades los lugares donde las personas viven, trabajan y desarrollan una serie de actividades, ya sea dentro o fuera de los hogares. Algunas de estas actividades demandan el uso de diferentes formas de desplazamiento tales como: caminata, transporte mecanizado (bicicleta) o motorizado (autobuses, motocicletas, automóviles, ferrocarriles y metro) [27]. Teniendo en cuenta lo anterior, es evidente que el desplazamiento de la población dentro de la ciudad constituye un factor fundamental para que todos los que la integran puedan realizar sus tareas y cumplir sus respectivos roles dentro de ésta.

Es importante destacar que, el concepto de movilidad es diferente al de transporte. El transporte solo hace referencia al desplazamiento de personas o productos, mientras que la movilidad es un concepto más amplio y complejo, ya que ésta, además del desplazamiento, incorpora variables como la infraestructura, los vehículos y las condiciones sociales, políticas, económicas y culturales de quienes se movilizan. No existe un concepto único y completo de movilidad, la comisión de comunidades europeas, sostiene que movilidad es la capacidad que supone aprovechar al máximo y de forma aislada, el uso de todos los modos de transporte colectivo, (tren, tranvía, metro, autobús y taxi) y los diversos modos de transporte individual (automóvil, bicicleta y marcha a pie), ya que, en combinación, darán como resultado una óptima y sostenible utilización de los [27].

Para el programa de medio ambiente de la obra social caja Madrid, la movilidad no es sino un medio para permitir a los ciudadanos, colectivos y empresas acceder a la multiplicidad de servicios, equipamientos y oportunidades que ofrece la ciudad. Su objetivo es que los ciudadanos puedan alcanzar el destino deseado en condiciones de seguridad, comodidad e igualdad y de la forma más autónoma y rápida posible [27].

Ahora bien, sobre los aspectos sociales, políticos, económicos y culturales que integran el concepto de movilidad, tres grandes condiciones más o menos interrelacionadas se hacen evidentes. Una de ellas relativa a la ampliación del abanico de posibilidades de desplazamientos, lo que significa centrarse en las necesidades de las personas y su accesibilidad; las condiciones relativas a la sostenibilidad ambiental de los medios de transporte, soporte de la movilidad y las condiciones que permiten garantizar la articulación de los distintos modos de transporte que se ofrecen a las personas en una ciudad determinada. La conjunción de estos tres tipos de condiciones permite establecer las diferencias entre transporte y movilidad [27]. Por consiguiente, se construye un concepto de movilidad más general donde ésta no se entiende solamente como la capacidad para desplazarse sino, como el conjunto de variables que afectan positiva o negativamente esta función dentro de un espacio con características específicas.

La sostenibilidad de un sistema se da cuando los recursos consumidos para asegurar las necesidades del presente, no comprometen las necesidades de futuras generaciones; es decir, que los recursos que se consumen son menores a los que se desarrollan o renuevan. Entonces, una ciudad es sostenible cuando suple las necesidades de su población a cabalidad, además de estar en capacidad de seguir sufriendo las de generaciones venideras, teniendo en cuenta el crecimiento de la población y el alza en el consumo de recursos que esto implica. Por esto, para una ciudad alcanzar el reto de ser sostenible, debe contar con una movilidad que lo sea, es decir, la ciudad debe estar en capacidad de movilizar a las personas que la conforman, sufriendo sus diferentes necesidades de forma eficiente, además, de garantizar el continuo y futuro cumplimiento a pesar de la ampliación de los tejidos urbanos, tanto en extensión territorial como en densidad poblacional.

La movilidad urbana en Popayán, presenta grandes retos en su camino a ser sostenible, ya que la ciudad no cuenta con medios masivos de transporte como tren, tranvía, metro o

sistemas BTR (Buses de Transito Rápido). También, la arquitectura colonial en algunos sectores de la ciudad, no permite la ampliación vial, a lo cual se suma que en los últimos años la población se ha incrementado notablemente extendiendo el casco urbano, haciendo que en algunos casos su acceso se dificulte, y algunos medios de transporte no sean una opción.

Otro de los grandes problemas en la movilidad en Popayán, es el incremento en el parque automotor. En Colombia, el Ministerio de Transporte estima en 4.6% el incremento anual del parque automotor [28]; este incremento, junto a la ausencia de nuevas vías o planes de movilidad para mitigar este fenómeno, generan gran congestión vehicular, convirtiéndose en un problema para la movilidad sostenible.

Para el caso particular de Popayán, el sistema TPC transporta diariamente entre el 35% y 40% de la población total de la ciudad [12], es por ello que, solventar los problemas descritos anteriormente, representarían un impacto positivo en la movilidad de la ciudad. En ese sentido, el presente trabajo; busca a partir de dichas problemáticas proponer una alternativa desde la rama de la electrónica y las telecomunicaciones que pueda ser de ayuda para que el sistema TPC brinde un mejor servicio y, por consiguiente, contribuya a la movilidad en la ciudad.

1.3. Sistema actual de transporte público colectivo Popayán.

El sistema TPC de Popayán está compuesto por 45 rutas autorizadas y cuatro empresas transportadoras licenciadas para prestar el servicio urbano y suburbano, cada una de ellas con una participación porcentual como se muestra en la Ilustración 2 y en la Tabla 1.

Tabla 1 - Rutas y Flota servicio TPC por empresa

Empresa	Cantidad de Rutas	Flota Operativa
TRANSTAMBO	6	84
SOTRACAUCA METRO	9	110
TRANSLIBERTAD	10	143
TRANSPUBENZA	15	206
TOTAL	40	543

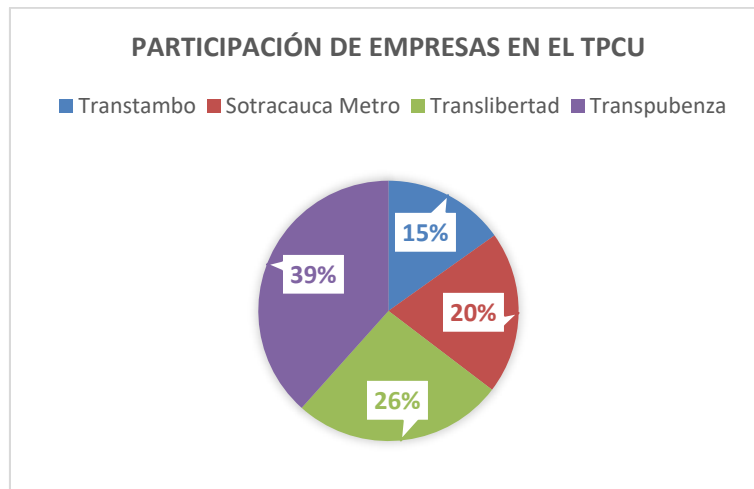


Ilustración 2 - Participación de Empresas en el sistema TPC
 Adaptada de: [12]

De acuerdo a la información del diseño conceptual del año 2008 y, a los datos elaborados por la empresa Movilidad Futura S.A.S con levantamiento de información primaria en el 2013 presente en Tabla 1, la cobertura espacial de las 40 rutas autorizadas en la ciudad de Popayán es de 26.69 Km²; correspondiente al 100%, como se puede observar en la Ilustración 3 [12].

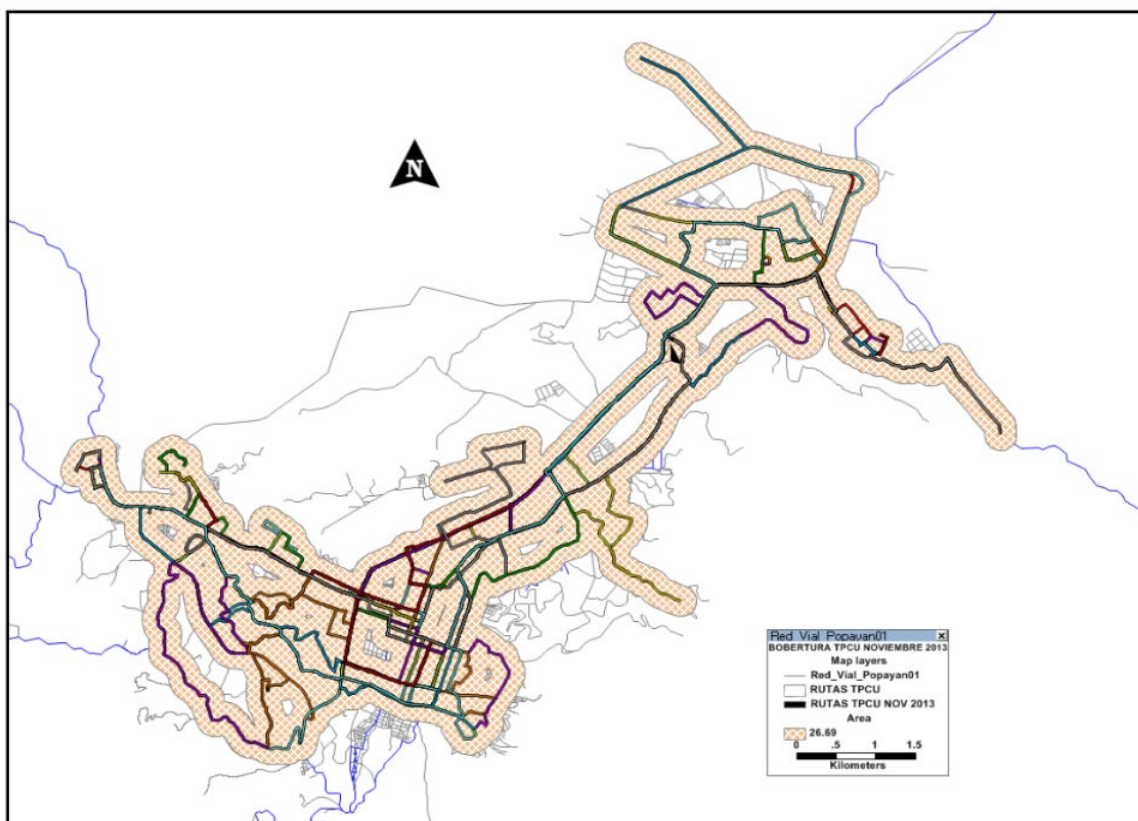


Ilustración 3 - Cobertura espacial a 200 m de las rutas en el área urbana
 Tomado de: [12]

Actualmente, los vehículos de las cuatro empresas tienen capacidades que varían entre 13 y 27 pasajeros sentados, dependiendo del tipo y la distribución interna de la silletería, esto según información suministrada a octubre de 2013 por las cuatro empresas transportadoras. La empresa Movilidad Futura S.A.S condensó dicha información de interés en Tabla 2 y Tabla 3.

Tabla 2 - Características de los vehículos

CARACTERÍSTICAS DE LOS VEHÍCULOS			
Empresa	Cap. Busetones	Cap. Microbús	Marcas
Transpubenza	23 a 27 pasajeros	12 a 19 pasajeros	Chevrolet, Toyota, Dahiatsu, Nissan, NPU, Mitsubishi
Translibertad	27 pasajeros	19 pasajeros	Nissan, Mitsubishi
Sotracauca Metro	27 pasajeros	19 pasajeros	Nissan, Mitsubishi
Transtambo	No Tiene	19 pasajeros	Dahiatsu, Nissan

Tabla 3 - Puntos de despacho y rutas

PUNTOS DE DESPACHO Y RUTAS				
	Número de rutas en operación	Rutas veredales	Frecuencia [minutos]	
			Hora pico	Hora valle
Transpubenza	15	4-8-9-10-11 MB y 2BT	8	8 - 10
Translibertad	10	1 BT - 6BT - 3MB	6	7 - 9
Sotracauca Metro	9	3 - 4 - 5 -7 -8 - 9	6	8 - 10
Transtambo	6	2 - 4 - 5 - 6	8	8 - 13

En el año 2014, la empresa Movilidad Futura S.A.S también realizó un estudio en terminales y paraderos para determinar la flota en operación en el casco urbano un día corriente. La información se encuentra en la Tabla 4.

Tabla 4 - Flota en operación en día corriente

EMPRESA	CANTIDAD DE RUTAS	FLOTA EFECTIVA EN OPERACIÓN URBANA		
		Busetón	Microbús	Total
Transpubenza	15	84	99	183
Translibertad	10	83	58	141
Sotracauca Metro	9	2	75	77
Transtambo	6	0	60	60
TOTAL	40	169	292	461

Finalmente, en el Anexo A, se pueden observar las ilustraciones con las coberturas de las diferentes rutas para cada una de las empresas, las cuales se basan en información entregada por las 4 empresas prestadoras del servicio TPC en la ciudad de Popayán, y construidas por la empresa Movilidad Futura S.A.S en el año 2014, dentro de sus informes de estudio y diagnóstico de la situación actual del sistema TPC en Popayán [12]. Luego de ser cotejadas con información actual obtenida en [29] se encontró que las diferentes rutas en cada una de las empresas han sufrido mínimas adaptaciones o cambios en los últimos 4 años. Además, se expresa que dichas adaptaciones no han sido significativas en los 8 años anteriores a la realización del informe, es decir que, desde el año 2006 en la ciudad de Popayán no se ha realizado un replanteamiento significativo de las rutas, el cual tenga en cuenta desde el crecimiento de la población en ciertas zonas de la ciudad, el crecimiento de algunos sectores urbanísticos e incluso el surgimiento de nuevos.

Según el estudio del ALG-TMB¹ dentro del mismo informe de diagnóstico de la situación actual del TPC [12], la demanda del servicio TPC de la ciudad de Popayán es de aproximadamente 135.000 viajes en un día promedio. Según el diseño conceptual, la demanda diaria en un día hábil se estimó en aproximadamente 127.456 viajes. Este estudio se validó con programación de frecuencia y ocupación visual; por medio del estudio en terminales y el estudio de ascenso y descenso, encuestas origen-destino a bordo de los buses, y a hogares para validar y actualizar dicha información. Los métodos descritos anteriormente resultan engorrosos y pueden presentar una alta tasa de error.

De la anterior información, se puede apreciar cómo en la actualidad el sistema TPC en Popayán carece de infraestructura y herramientas que permitan a las empresas que administran el servicio, realizar un conteo o gestión verídico del abordaje de pasajeros. Dicho esto, se evidencian una serie de problemáticas en el servicio TPC, resultantes de la ausencia de un mecanismo de control y gestión de los datos de abordaje. Uno de ellos radica en la recaudación del dinero generado por la prestación del servicio, ya que al desconocer el número exacto de personas que usan este medio de transporte, no se podrá conocer el valor verídico del dinero recaudado, generando malas prácticas en el recaudo por parte de los conductores. Asimismo, al desconocer el número de personas que usan el servicio con su correspondiente hora y lugar de abordaje, no es posible determinar si el servicio se está brindando de forma eficaz en las diferentes rutas, problema que se evidencia en algunas de estas, donde a determinadas horas los buses pasan a escasos minutos de diferencia con pocos pasajeros, y en horas de mayor congestión resultan insuficientes para la demanda del servicio.

Una de las principales problemáticas radica en que, debido a la ausencia de gestión dentro del servicio, no es posible detectar diferentes inconvenientes presentes en este. Uno de ellos, como se dijo anteriormente; radica en que las rutas no han sido modificadas en los últimos 12 años para adaptarse a las necesidades que han surgido debido a la ampliación del casco urbano en la ciudad de Popayán [12]. Como este, existen otros inconvenientes y debido a la

¹ Consorcio de amplia experiencia en el manejo de transporte público en grandes ciudades como Barcelona.

falta de herramientas que permitan identificarlos, no es posible darles una rápida y eficaz solución, generando así, una mala imagen del servicio donde los usuarios optan por medios alternos de transporte.

1.4. Necesidades a satisfacer del sistema

Para poder encontrar los requerimientos del sistema, se realizó un acercamiento con la empresa Movilidad Futura para determinar las necesidades a satisfacer teniendo en cuenta la caracterización de la zona y los actores que en ella intervienen.

El acercamiento se basó en reuniones y retroalimentación respecto al trabajo a desarrollar. A partir de dichos eventos, se determinaron características específicas que el sistema prototipo debía incluir:

- **Prototipo invisible para el usuario:** Se quiere que la solución no implique ningún contacto con el pasajero, es decir que lo que se implemente dentro del bus no debe requerir que quien se suba o baje haga algo en específico.
- **No requerir reestructuración:** Es de vital importancia que la solución no requiera que deba hacerse una reestructuración en el sistema, es decir que no implique el cambio de infraestructura actual del sistema de transporte público colectivo en la ciudad. Este punto aclara que el prototipo debe ser lo menos invasivo posible y centrarse en lo que sucede dentro de cada uno de los buses.
- **Conteo de pasajeros:** Es uno de los puntos más importantes para lo que se quiere llegar a implementar, ya que es la base del reto propuesto por la empresa.
- **Sistema de gestión:** Se quiere que haya una herramienta que permita además de visualizar datos de abordajes, realizar gestión interna de la empresa, es decir, tener la posibilidad de contar con un registro de conductores, empleados, rutas y buses con las que se cuente, además de poder realizar las operaciones básicas de crear, editar y eliminar cada uno de ellos.

1.5. Descripción del sistema prototipo.

A partir de las necesidades manifestadas por la empresa Movilidad Futura se realiza un esquema de alto nivel, el cual se observa en la Ilustración 4.



Ilustración 4 - Esquema de alto nivel del sistema prototipo

Se identificó que, para la satisfactoria implementación del sistema, es necesario tener en cuenta dos partes importantes, una de las cuales se encargará de llevar el conteo en cada vehículo y otra que sería una herramienta que permita ver los datos tomados y además tener funciones de gestión internas para la empresa.

Entonces, nacen dos módulos en el proyecto, el primero será llamado DMC, que corresponde a las siglas de Dispositivo Móvil de Captura y el segundo que se llamará AWGI, representando Aplicación Web para la Gestión de la Información.

El DMC es el dispositivo de tipo hardware que se va a instalar en cada uno de los buses, él se va a encargar del conteo de ascensos y descensos de pasajeros. Los datos de dicho conteo deben ser llevados a la nube, por lo cual se hace necesario el uso de una base de datos.

La AWGI por otro lado, es una aplicación web que va a permitir la visualización de los datos que sean alojados en la base de datos por el dispositivo de conteo. Además, en ella se van a incluir funcionalidades de gestión para los diferentes activos que la empresa tenga, para ello, se establecen 4 roles de usuarios con características y funciones bien definidas, los cuales se basan en la jerarquización de los permisos, con el fin de restringir las posibilidades de diferentes tipos de usuarios, para tener una mejor gestión:

- **Administrador root:** rol principal dentro de la aplicación web, con todos los permisos y funciones habilitadas. Éste podrá crear, editar y eliminar, cualquier usuario (no-root), así como rutas y buses, además de visualizar, filtrar y descargar toda la información recolectada.
- **Administrador:** rol encargado de la gestión diaria de la aplicación web, cuenta con todas las funciones a excepción de la administración de su propio rango.
- **Empleado:** rol con función única de visualizar, filtrar y descargar los datos recolectados.

De igual forma se establece un tipo de usuario para el DMC, dado que es necesario que un actor se encargue de inicializar dicho dispositivo.

- **Conductor:** rol con acceso a la aplicación móvil por medio del DMC instalado en cada uno de los buses, donde podrán iniciar/detener el conteo de pasajeros.

1.6. Requerimientos del sistema

Una vez establecidas las necesidades con ayuda de la empresa que implementa el SETP y levantado el diagrama de alto nivel del prototipo, se determinan los requerimientos funcionales y no funcionales de forma más específica:

1.6.1. Requerimientos funcionales

Los requerimientos funcionales del sistema estarán divididos en 2 partes, una para la aplicación web AWGI y otra para el módulo hardware-software correspondiente al DMC:

1.6.1.1. Requerimientos funcionales AWGI

- Iniciar sesión validando las credenciales ingresadas.
- Un usuario de tipo Administrador root será el encargado de crear las cuentas de usuario y asignar sus roles, de forma que también es capaz de crear un usuario con sus mismas credenciales.
- La aplicación web va a usar una base de datos que permite el acceso a datos para visualización y gestión.
- Según la credencial del usuario que inicia sesión, se determinará el contenido de la aplicación que estará disponible, de tal forma que, si es un usuario empleado, no va a tener acceso a ninguna función de crear, eliminar o editar (en adelante CRUD) algún

usuario, de manera que solo podrá visualizar los datos proporcionados por el DMC. Si inicia sesión un usuario Administrador, éste va a tener además la posibilidad de hacer un CRUD de empleados y, finalmente, si inicia sesión un usuario Administrador root, va a poder realizar un CRUD de Administradores.

- Visualizar los datos que fueron recolectados por los dispositivos instalados en cada vehículo.
- Filtrar los datos a mostrar a partir de la búsqueda de un vehículo en específico, el cual tendrá asignada una ruta. La información de conteo se podrá buscar para un día en específico o para un rango de días.
- La información de conteo se va a mostrar en gráficas de barras y gráficas de líneas, así como una tabla que tenga todas las componentes del modelo, dicha tabla de información de los datos filtrados se va a poder descargar en un archivo de tipo csv.
- Cada usuario que accede a la AWGI va a poder actualizar su información básica (nombre completo y edad) desde el perfil de usuario.
- Un usuario del sistema podrá recuperar su contraseña por medio de un mensaje de reactivación por correo electrónico.
- Los usuarios de tipo Administrador y Administrador root, van a poder gestionar la asignación de buses a una ruta específica y cada conductor a un bus específico. De modo que diariamente un conductor será habilitado para iniciar sesión en el DMC del bus que le corresponde.
- Todo usuario va a poder cerrar sesión desde un menú desplegable presente en la interfaz.

1.6.1.2. Requerimientos funcionales DMC

- Permitir el inicio de sesión de cada conductor asignado a un respectivo bus, validando sus credenciales.
- Permitir activar el conteo desde la pantalla inicial, luego de validar los datos de acceso del conductor.
- Realizar la desactivación del conteo por medio del cierre de sesión del conductor.
- Detectar el evento de ingreso y salida por parte de un pasajero, de acuerdo al vector de recorrido que este realice y transmitir esta información (entrada/salida, ubicación coordenada, hora, número de personas en vehículo) a la base de datos.

- El dispositivo emitirá una alerta sonora cuando su área de detección sea bloqueada, es decir cuando haya una obstrucción o impedimento para realizar el conteo. Dicha alerta sonora tendrá una duración equivalente a la presencia de la obstrucción.

1.6.2. Requerimientos no Funcionales

De igual forma, los requerimientos no funcionales se seccionarán entre la aplicación web y el DMC.

1.6.2.1. Requerimientos no funcionales AWGI

- La aplicación web debe poseer un diseño *responsive*².
- La aplicación puede utilizarse en navegadores web Chrome, Firefox e Internet Explorer sin necesidad de instalar ningún software adicional.

1.6.2.2. Requerimientos no funcionales DMC

- El prototipo debe contar con un módulo de alimentación que garantice su funcionamiento durante el tiempo en el que se esté ejecutando.
- El sistema debe tener una fiabilidad en el conteo de pasajeros superior al 85%. Este porcentaje fue acordado con Movilidad futura, teniendo en cuenta que el sistema tiene varios puntos que suplir además del conteo y para posibles trabajos futuros se deja una brecha corta en porcentaje de fiabilidad.
- El sistema se dimensiona para condiciones de funcionamiento ideales en los que el sobrecupo sería poco frecuente. El número de asientos promedio en cada bus es de 24. No se garantiza el rendimiento en condiciones de sobrecupo.
- Se considera que las características de todos los buses en cuanto a altura y medidas de puertas de ingreso serán similares para el prototipo diseñado, dado que un cambio radical en dichas medidas implicaría la reconfiguración de ciertos parámetros.
- La tecnología seleccionada para el conteo de pasajeros debe contemplar el uso continuo y diario de más de 18 horas en los criterios de decisión.
- Dadas las extensas jornadas de trabajo del dispositivo, si se llega a una fase de implementación, es importante contar con un sistema de ventilación para mantener la temperatura del mismo.

² Web capaz de adaptarse a cualquier dispositivo donde se visualice.

- El flujo de pasajeros en el bus se debe realizar de forma cotidiana y no debe ser necesario que el usuario haga alguna acción específica para que el DMC efectúe el conteo.

Capítulo 2

DISEÑO

En este capítulo, con base en los requerimientos planteados, se realiza el diagrama de clases y el diagrama de casos de uso, de componentes y de despliegue. Con estos se tiene una primera descripción de la arquitectura del sistema, su comportamiento y sus características.

2.1. Diagramas del proyecto.

Este apartado está dedicado a los diferentes diagramas que ayudaron con el diseño y modelamiento del proyecto, ayudándonos a conocer los diferentes actores, asociaciones, funciones y atributos.

2.1.1. Diagrama de Casos de Uso.

Se identifican 4 actores o roles: administrador root, administrador, empleado y conductor en ese orden de jerarquía específicamente, en la Ilustración 5, se pueden observar las diferentes acciones que estos realizan dentro del sistema, luego de validar sus datos de acceso.

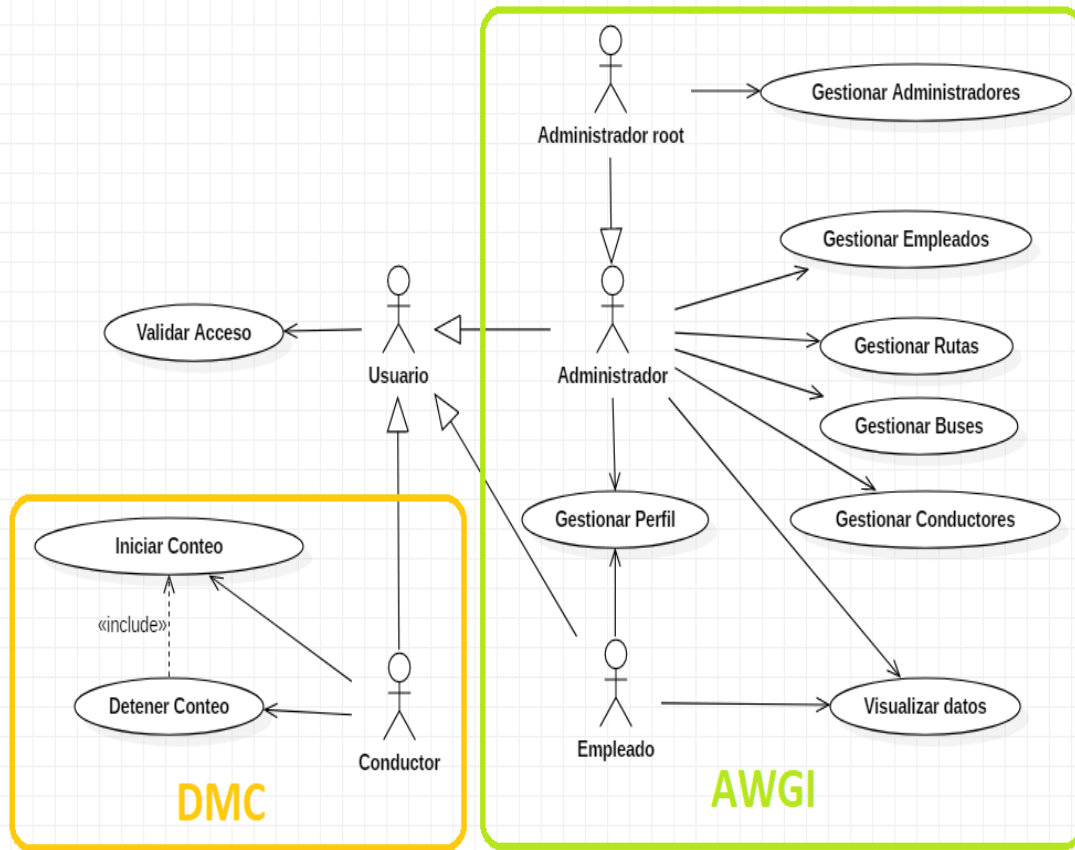


Ilustración 5 - Diagrama de casos de uso del proyecto

Es importante aclarar que, el administrador root es el de mayor jerarquía, puesto que, puede realizar todas las funciones dentro de la AWGI y es el único rol con los permisos de gestionar administradores, este rol hereda del actor Administrador ya que de esta forma éste podrá heredar las funciones que éste tiene, y por su lado sumar la acción de gestionar administradores. Por otro lado, todos los actores del sistema realizan la función de validar acceso debido a que todos heredan de usuario.

2.1.1.1. Descripción casos de uso Extendidos

A continuación, se desglosará en detalle cada uno de los casos de uso del proyecto:

Tabla 5 - Caso de uso extendido Iniciar/Detener conteo

Información General	
Caso de uso:	Iniciar/Detener conteo
Actores:	Conductor
Propósito:	Iniciar y detener el conteo y registro del flujo de pasajeros.
Resumen:	El conductor desde la interfaz principal de la aplicación móvil, en el DMC, iniciará el conteo de pasajeros por medio del botón “Iniciar Conteo”, y de igual manera, el registro de los datos de ascensos y descensos que se realicen desde ese momento. El conteo de pasajeros estará activo hasta el momento que el conductor lo desactive por medio del cierre de su sesión al final de la jornada laboral.
Tipo:	Primario Esencial
Referencias Cruzadas:	Casos de uso: Validar Acceso
Precondiciones	
<ul style="list-style-type: none"> • Los datos del conductor deben encontrarse previamente almacenados en la base de datos del sistema. • El conductor debe iniciar sesión en el dispositivo validando sus credenciales, antes de tener acceso a la opción que permite iniciar el conteo de pasajeros. 	
FLUJO PRINCIPAL	
<ol style="list-style-type: none"> 1. Iniciar el conteo: El conductor pulsa el botón de “Iniciar conteo” desde la interfaz principal de la aplicación móvil en el DMC (E1), la cual se muestra justo después de iniciar sesión en el dispositivo. 2. Cerrar Sesión: Cuando el conductor finalice su jornada laboral y proceda a cerrar su sesión para realizar la desactivación del conteo de pasajeros, despliega la interfaz del menú lateral y selecciona la opción “Cerrar Sesión”, acción que requiere confirmación. 	
FLUJOS DE EXCEPCIÓN	
<ul style="list-style-type: none"> • E1: Problema conexión de red: <ul style="list-style-type: none"> - El sistema despliega el mensaje de error (Ilustración 7) indicando que hubo un error de conexión. 	

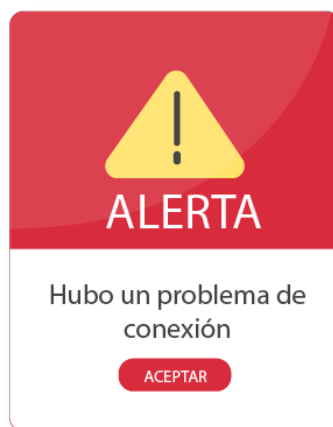


Ilustración 6 - Alerta problema de conexión

Tabla 6 - Caso de uso extendido visualizar datos

Información General	
Caso de uso:	Visualizar datos
Actores:	Administrador root, Administrador, Empleado
Propósito:	Visualizar, filtrar y descargar la información correspondiente a los datos del conteo de pasajeros recolectados en cada uno de los buses.
Resumen:	<p>Los administradores y empleados accederán a la información recolectada en cada uno de los buses, desde las interfaces de “Gráficas” o “Mapa”, en las cuales podrán visualizar los diferentes datos de interés en diferentes presentaciones, dependiendo de los filtros asignados por el usuario.</p> <p>En la interfaz de “Gráficas” se visualizan los gráficos y tablas correspondientes a los datos capturados de conteo, dependiendo de los parámetros de búsqueda, además las tablas con la información filtrada pueden ser descargadas en formato csv.</p> <p>Desde la interfaz “Mapa” se visualizan el mapa correspondiente al bus filtrado en un día específico, en el cual se dibujan los puntos de calor correspondientes a las locaciones donde se produjeron los abordajes.</p>
Tipo:	Primario Esencial
Referencias Cruzadas:	Casos de uso: validar acceso.
Precondiciones	
<ul style="list-style-type: none"> • Los datos de los usuarios deberán estar registrados previamente en la base de datos del sistema. • Los usuarios deben iniciar sesión dentro de la plataforma para acceder a la interfaz donde se visualizan los datos correspondientes al conteo de pasajeros. 	
FLUJO PRINCIPAL	

<ol style="list-style-type: none"> Ingresar a la interfaz: El usuario selecciona la opción “Gráficas” o “Mapa” desde la pantalla principal de la AWGI para ingresar a la interfaz correspondiente donde podrá visualizar la información deseada. Filtrar Datos: dentro de la interfaz seleccionada para visualizar los datos, se deben llenar los campos requeridos para filtrar la búsqueda de los datos correspondientes a lo que se desee visualizar (E1), finalmente al pulsar en el botón “Buscar” se visualiza la información solicitada (S1).
SUBFLUJOS
<ul style="list-style-type: none"> S1: Descargar Grafico Esta opción se dará para descargar las tablas si es el caso.
FLUJOS DE EXCEPCIÓN
<ul style="list-style-type: none"> E1: Datos no encontrados El sistema despliega el mensaje de error (Ilustración 8), informando que no existen datos con los parámetros suministrados.



Ilustración 7 - Alerta datos no encontrados

Tabla 7 - Caso de uso extendido gestionar empleados

Información General	
Caso de uso:	Gestionar Empleados
Actores:	Administrador root, Administrador
Propósito:	Permitir el proceso de creación, lectura, modificación y eliminación de empleados.
Resumen:	Desde la pantalla principal los administradores ingresando a la interfaz “Empleados” podrán gestionar los empleados existentes y los datos correspondientes a estos, o en su defecto crear uno nuevo.
Tipo:	Primario Esencial
Referencias Cruzadas:	Casos de uso: validar acceso.
Precondiciones	
<ul style="list-style-type: none"> Los datos del administrador deberán estar registrado previamente en la base de datos del sistema. El administrador debe iniciar sesión dentro de la plataforma, para poder acceder a la interfaz correspondiente para la gestión de los usuarios. 	
FLUJO PRINCIPAL	
<ol style="list-style-type: none"> Ingresar a la interfaz Empleados: El administrador selecciona la opción “Empleados” en la pantalla principal de la AWGI para ingresar a la interfaz donde se podrán crear, leer, modificar y eliminar los empleados. 	

- 2. Gestionar empleados:** El administrador dentro de la interfaz podrá observar todos los empleados existentes en el sistema, crear uno nuevo (E1, E2), eliminar o modificar uno existente (E1, E2).

FLUJOS DE EXCEPCIÓN

- **E1: Datos incorrectos**
El sistema despliega el mensaje de error (Ilustración 9), informando que los datos ingresados en los formularios tienen no cumplen los requisitos.
- **E2: Usuario Existente**
El sistema despliega el mensaje de error (Ilustración 10), informando que el correo suministrado ya existe.



Ilustración 8 - Alerta datos incorrectos



Ilustración 9 - Alerta usuario existente

Tabla 8 - Caso de uso extendido gestionar administradores

Información General	
Caso de uso:	Gestionar administradores
Actores:	Administrador root
Propósito:	Permitir el proceso de creación, lectura, modificación y eliminación de administradores.
Resumen:	El administrador root por medio de la interfaz “Administradores” podrá gestionar los administradores existentes y los datos correspondientes a estos, o en su defecto crear uno nuevo.
Tipo:	Primario Esencial
Referencias Cruzadas:	Casos de uso: validar acceso.
Precondiciones	
<ul style="list-style-type: none"> • Los datos del administrador root deberán estar registrado previamente en la base de datos del sistema. 	

<ul style="list-style-type: none"> El administrador root debe iniciar sesión dentro de la plataforma, para poder acceder a la interfaz correspondiente para la gestión de los administradores.
FLUJO PRINCIPAL
<ol style="list-style-type: none"> Ingresar a la interfaz Administradores: El administrador root selecciona la opción “Administradores” en la pantalla principal de la AWGI para ingresar a la interfaz donde se podrán crear, leer, modificar y eliminar los administradores. Gestionar administradores: El administrador root dentro de la interfaz podrá observar todos los administradores existentes en el sistema, crear uno nuevo (E1, E2), eliminar o modificar uno existente (E1, E2).
FLUJOS DE EXCEPCIÓN
<ul style="list-style-type: none"> <u>E1: Datos incorrectos</u> El sistema despliega el mensaje de error (Ilustración 9), informando que los datos ingresados en los formularios no cumplen los requisitos. <u>E2: Usuario Existente</u> El sistema despliega el mensaje de error (Ilustración 10), informando que el correo suministrado ya existe.

Tabla 9 - Caso de uso extendido gestionar conductores

Información General	
Caso de uso:	Gestionar conductores
Actores:	Administrador root
Propósito:	Permitir el proceso de creación, lectura, modificación y eliminación de administradores.
Resumen:	El administrador root por medio de la interfaz “Administradores” podrá gestionar los administradores existentes y los datos correspondientes a estos, o en su defecto crear uno nuevo.
Tipo:	Primario Esencial
Referencias Cruzadas:	Casos de uso: validar acceso.
Precondiciones	
<ul style="list-style-type: none"> Los datos del administrador root deberán estar registrados previamente en la base de datos del sistema. El administrador root debe iniciar sesión dentro de la plataforma, para poder acceder a la interfaz correspondiente para la gestión de los administradores. 	
FLUJO PRINCIPAL	
<ol style="list-style-type: none"> Ingresar a la interfaz Conductores: El administrador root selecciona la opción “Conductores” en la pantalla principal de la AWGI para ingresar a la interfaz donde se podrán crear, leer, modificar y eliminar los conductores. Gestionar conductores: El administrador root dentro de la interfaz podrá observar todos los conductores existentes en el sistema, crear uno nuevo (E1, E2), eliminar o modificar uno existente (E1, E2). 	
FLUJOS DE EXCEPCIÓN	
<ul style="list-style-type: none"> <u>E1: Datos incorrectos</u> 	

<p>El sistema despliega el mensaje de error (Ilustración 9), informando que los datos ingresados en los formularios no cumplen los requisitos.</p> <ul style="list-style-type: none"> <p><u>E2: Usuario Existente</u> El sistema despliega el mensaje de error (Ilustración 10), informando que el correo suministrado ya existe.</p>

Tabla 10 - Caso de uso extendido gestionar buses

Información General	
Caso de uso:	Gestionar buses
Actores:	Administrador root, Administrador
Propósito:	Permitir el proceso de creación, lectura, modificación y eliminación de Buses.
Resumen:	Los Administradores desde la interfaz “Buses” podrán gestionar la información de los buses existentes, o en su defecto crear uno nuevo.
Tipo:	Primario Esencial
Referencias Cruzadas:	Casos de uso: validar acceso.
Precondiciones	
<ul style="list-style-type: none"> Los datos del administrador deberán estar registrados previamente en la base de datos del sistema. El administrador debe iniciar sesión dentro de la plataforma, para poder acceder a la interfaz correspondiente para la gestión de buses. 	
FLUJO PRINCIPAL	
<ol style="list-style-type: none"> <p>Ingresar a la interfaz Buses: El administrador selecciona la opción “Buses” en la pantalla principal de la AWGI para ingresar a la interfaz donde se podrán crear, leer, modificar y eliminar los buses.</p> <p>Gestionar buses: El administrador dentro de la interfaz podrá observar todos los buses existentes en el sistema, crear uno nuevo (E1, E2), eliminar o modificar uno existente (E1, E2).</p> 	
FLUJOS DE EXCEPCIÓN	
<ul style="list-style-type: none"> <p><u>E1: Datos incorrectos</u> El sistema despliega el mensaje de error (Ilustración 9), informando que los datos ingresados en los formularios no cumplen los requisitos.</p> <p><u>E2: Bus Existente</u> El sistema despliega el mensaje de error (Ilustración 11), informando que el bus ya existe (placa).</p> 	

Tabla 11 - Caso de use extendido gestionar rutas

Información General	
Caso de uso:	Gestionar rutas
Actores:	Administrador root, administrador
Propósito:	Permitir el proceso de creación, lectura, modificación y eliminación de las rutas de servicio que brinda la empresa.
Resumen:	Los administradores desde la interfaz “Rutas” podrán gestionar la información de las diferentes rutas con las que cuenta la empresa, o en su defecto crear una nueva.
Tipo:	Primario Esencial
Referencias Cruzadas:	Casos de uso: Validar Acceso
Precondiciones	
<ul style="list-style-type: none"> Los datos del administrador deberán estar registrados previamente en la base de datos del sistema. El administrador debe iniciar sesión dentro de la plataforma, para poder acceder a la interfaz correspondiente para la gestión de las rutas de servicio. 	
FLUJO PRINCIPAL	
<ol style="list-style-type: none"> Ingresar a la interfaz Rutas: El administrador selecciona la opción “Rutas” en la pantalla principal de la AWGI para ingresar a la interfaz donde se podrán crear, leer, modificar y eliminar las rutas. Gestionar rutas: El administrador dentro de la interfaz podrá observar todas las rutas existentes en el sistema, crear una nueva (E1, E2), eliminar o modificar una existente (E1, E2). 	
FLUJOS DE EXCEPCIÓN	
<ul style="list-style-type: none"> <u>E1: Datos incorrectos</u> El sistema despliega el mensaje de error (Ilustración 9), informando que los datos ingresados en los formularios no cumplen los requisitos. <u>E2: Ruta Existente</u> El sistema despliega el mensaje de error (Ilustración 12), informando que la ruta ya existe (nombre de ruta). 	



Ilustración 10 - Alerta bus existente

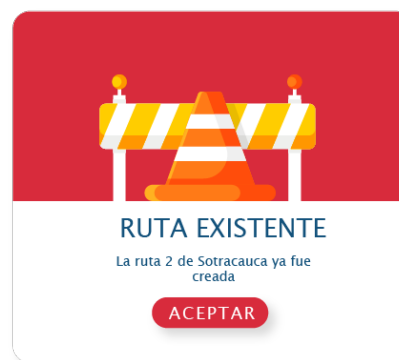


Ilustración 11 - Alerta ruta existente

Tabla 12 - Caso de uso extendido gestionar perfil

Información General	
Caso de uso:	Gestionar Perfil
Actores:	Administrador root, Administrador, Empleado
Propósito:	Gestionar la información propia del perfil usuario.
Resumen:	Los usuarios que ingresen a la AWGI, luego de validar sus datos de inicio de sesión, podrán acceder a la interfaz "Perfil", en la cual se encontrará su información personal y desde allí podrán realizar cambios a dicha información.
Tipo:	Primario Esencial
Precondiciones	
<ul style="list-style-type: none"> • Los datos de los usuarios deberán estar registrados previamente en la base de datos del sistema, para poder verificar sus credenciales. • El usuario debe iniciar sesión dentro de la plataforma, para poder acceder a la interfaz correspondiente para la gestión de su perfil. 	
FLUJO PRINCIPAL	
<ol style="list-style-type: none"> 1. Ingresar a la interfaz Perfil: Los usuarios deben seleccionar la opción "Perfil" desde la pantalla principal de la AWGI para ingresar a la interfaz donde se podrán gestionar su información personal. 2. Editar Información Personal: Dentro de la interfaz de "Perfil" los usuarios encontraran un formulario deshabilitado el cual contendrá sus datos personales, para editarlos; en la parte inferior se encontrará un botón "Editar perfil" el cual al ser pulsado permitirá la edición de dicho formulario. 3. Guardar cambios Perfil: Luego de pulsar el botón de "Editar perfil" se habilitará el botón "Guardar cambios" al editar la información deseada y pulsar dicho botón (E1), se actualizará la información personal. 	
SUBFLUJOS	
<ul style="list-style-type: none"> • <u>S1: Cambiar contraseña</u> <u>Esta opción permitirá realizar el cambio de contraseña para el usuario.</u> • <u>S2: Cambiar Imagen de perfil</u> <u>Esta opción se tendrá para ingresar una imagen la cual se establecerá como imagen de perfil.</u> 	
FLUJOS DE EXCEPCIÓN	
<ul style="list-style-type: none"> • <u>E1: Datos incorrectos</u> El sistema despliega el mensaje de error (Ilustración 9), informando que los datos ingresados en los formularios no cumplen los requisitos. 	

Tabla 13 - Caso de use extendido validar datos

Información General	
Caso de uso:	Validar datos
Actores:	Administrador root, Administrador, Empleado, Conductor
Propósito:	Validar las credenciales de los usuarios y restringir el acceso.
Resumen:	<p>Cualquier usuario desde la interfaz de “Login” debe ingresar sus credenciales para poder ingresar al sistema, y realizar las funciones que le correspondan.</p> <p>Los administradores root, administradores y empleados deben acceder desde al AWGI, los conductores desde el DMC</p>
Tipo:	Primario Esencial
Precondiciones	
	<ul style="list-style-type: none"> Los datos de los usuarios deberán estar registrados previamente en la base de datos del sistema, para poder verificar sus credenciales.
FLUJO PRINCIPAL	
	<p>1. Ingresar a la interfaz Principal: los usuarios deben ingresar sus datos de acceso en la interfaz “Login” ya sea en la AWGI o en el DMC dependiendo del rol y luego pulsar en el botón de “iniciar sesión” para validar sus datos y poder acceder al sistema (E1, E2).</p>
FLUJOS DE EXCEPCIÓN	
	<ul style="list-style-type: none"> <u>E1: Problema conexión de red:</u> El sistema despliega el mensaje de error (Ilustración 7) indicando que hubo un error de conexión. <u>E1: Datos incorrectos</u> El sistema despliega el mensaje de error (Ilustración 9), informando que los datos ingresados en los formularios tienen no cumplen los requisitos.

2.2. Diagramas de estados

A continuación, se ilustran los diagramas de máquina de estados, los cuales proporcionan la descripción de los diferentes estados que los principales objetos van a tener a lo largo de su ejecución.

2.2.1. Diagrama de estados DMC.

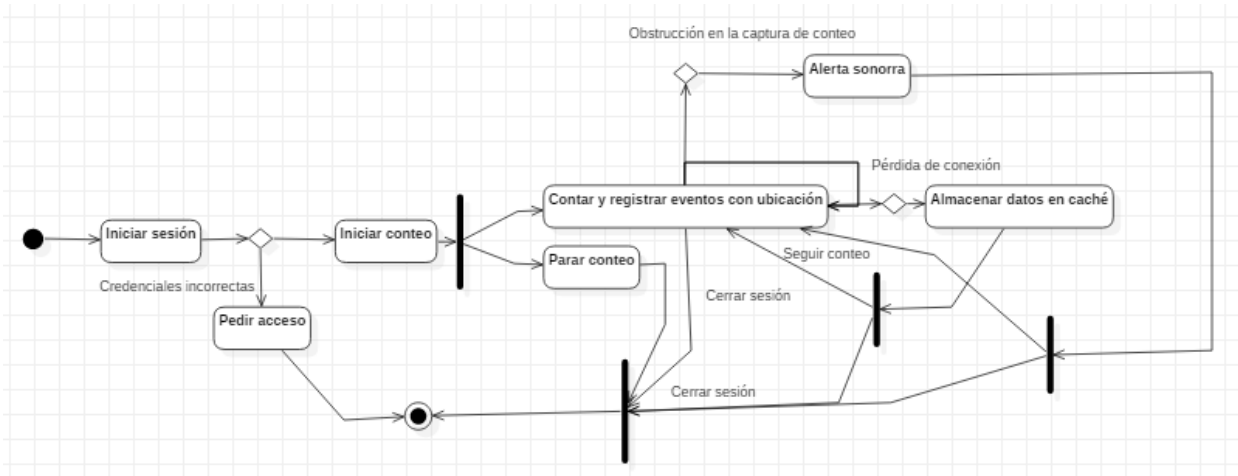


Ilustración 12 - Diagrama de estados DMC

2.2.2. Diagrama de estados AWGI.

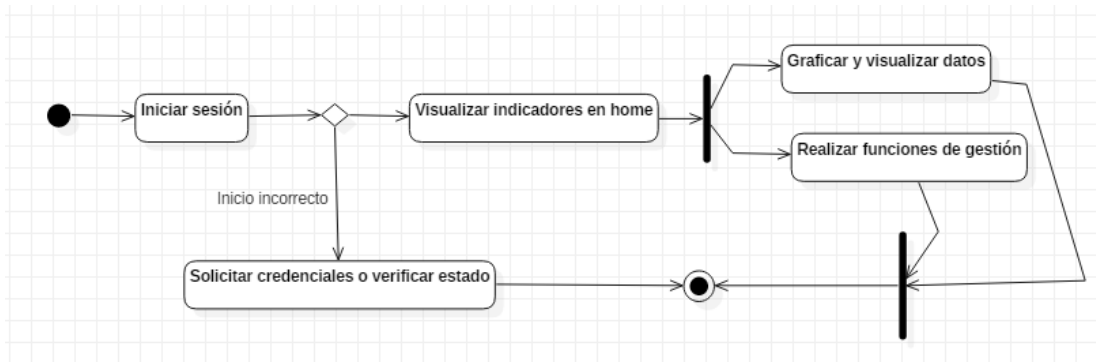


Ilustración 13 - Diagrama de estados AWGI

2.3. Diagramas de actividades

Los diagramas de estados muestran el flujo de los actores con respecto a los principales casos de uso.

2.3.1. Visualización de datos de conteo en la AWGI

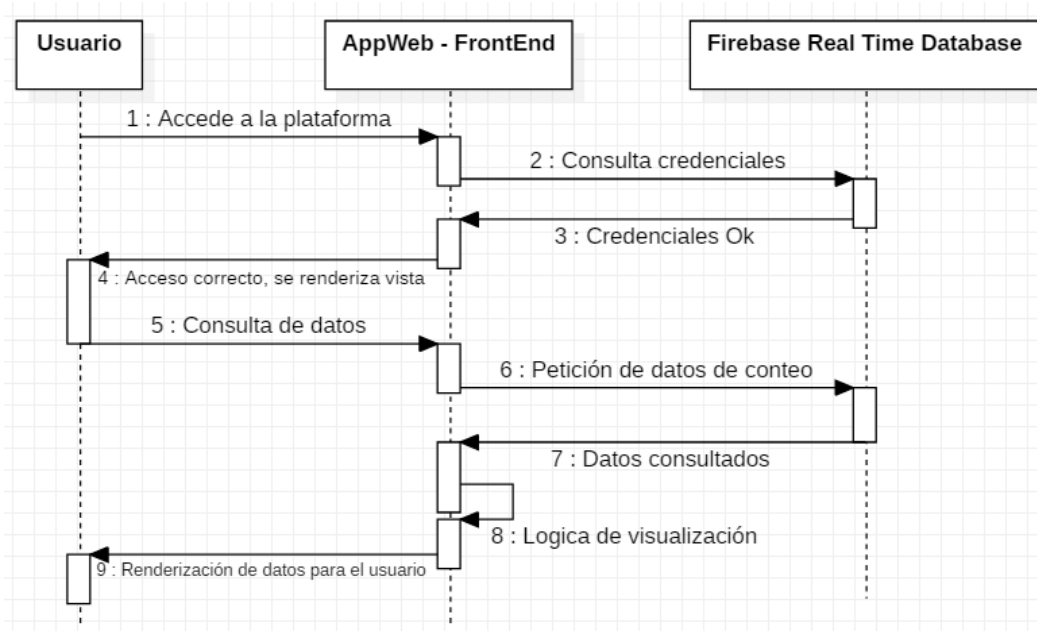


Ilustración 14 - Diagrama de secuencia para visualizar información de conteo

2.3.2. Gestión de usuarios en la AWGI

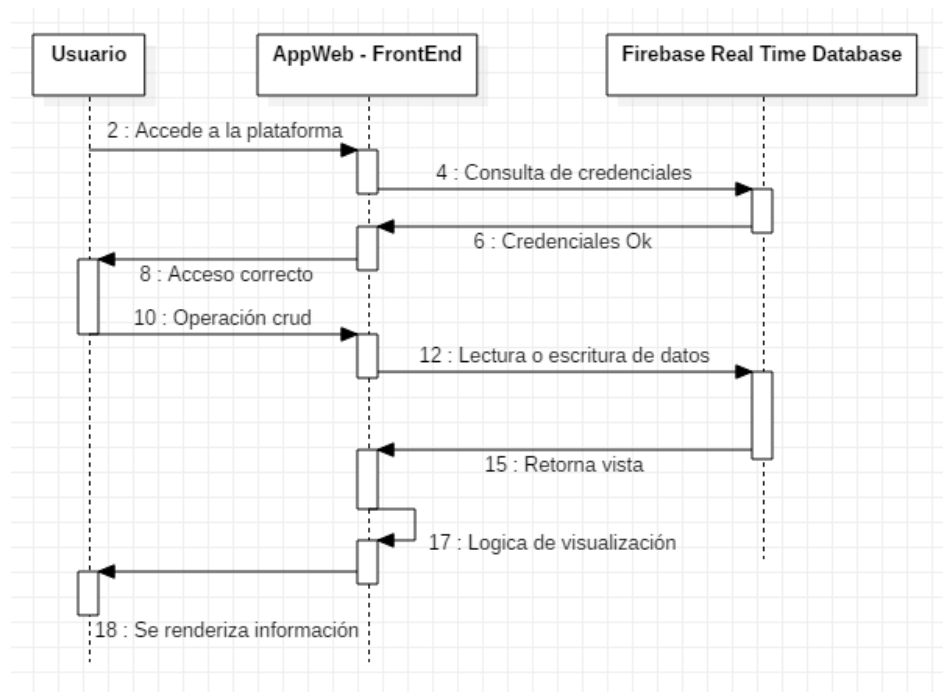


Ilustración 15 - Diagrama de secuencia para operaciones básicas con usuarios

2.3.2. Conteo de pasajeros

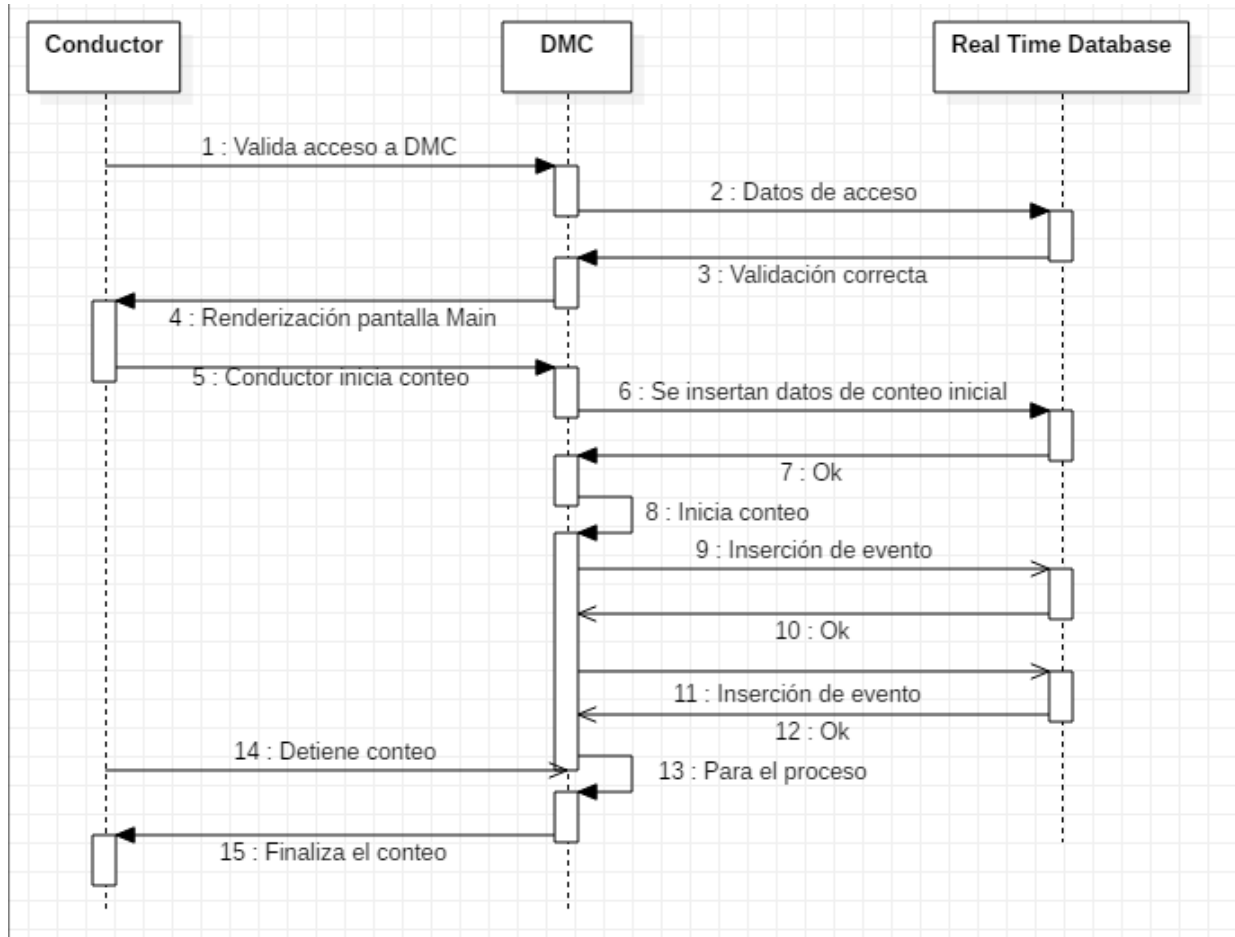


Ilustración 16 - Diagrama de secuencia para el conteo

2.4. Diagrama de clases.

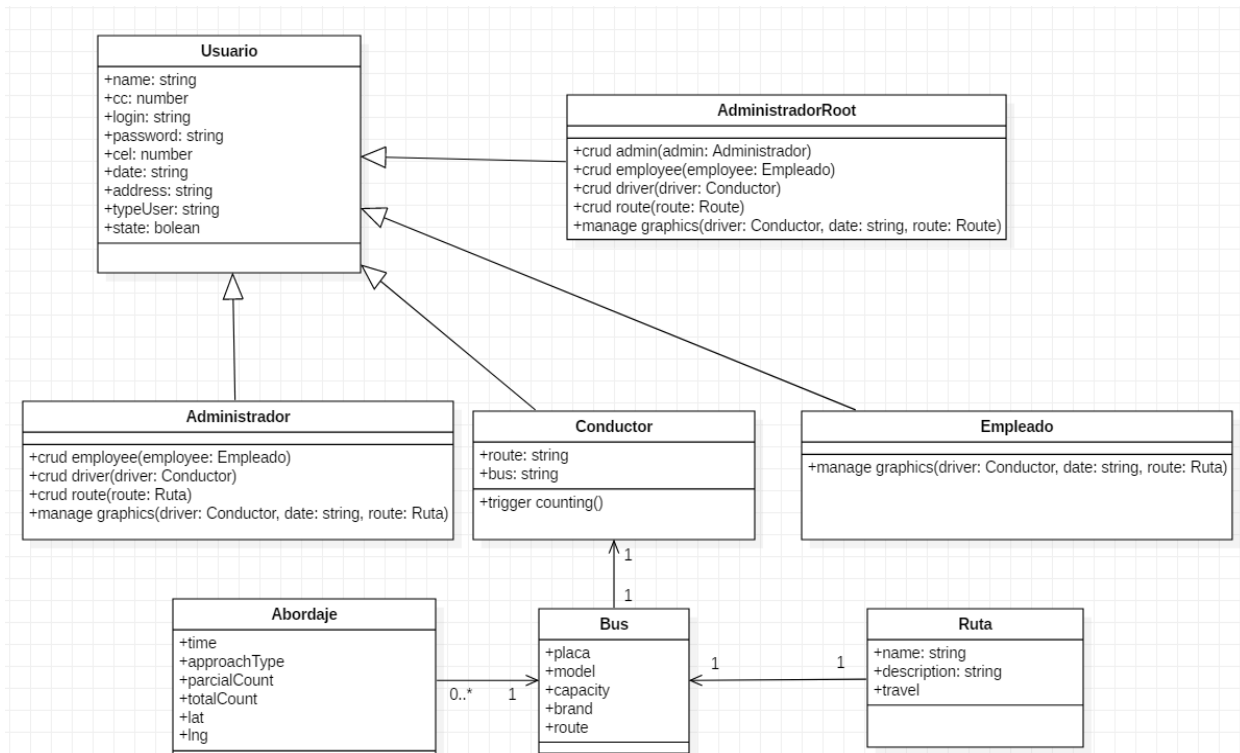


Ilustración 17 - Diagrama de clases del proyecto.

En la Ilustración 6, se observa el modelo utilizado para el proyecto en el cual se tiene una superclase (Usuario), la cual hereda a cada uno de los diferentes roles o tipo de usuarios que se tienen (Administrador root, Administrador, Conductor, Empleado), donde a su vez, cada uno de estos cuenta con funciones o acciones respectivas. La clase conductor está directamente relacionada a la clase bus, ya que cada uno de estos tendrá asignado solo un bus, a su vez los diferentes buses pertenecen a una única ruta, por último, la clase Abordaje corresponde a cada uno de los abordajes de pasajeros que se presenta en el bus en cuestión.

2.5. Diagrama de componentes

Los Diagramas de Componentes ilustran las piezas del software, controladores embebidos, etc. que conformarán un sistema. Un diagrama de Componentes tiene un nivel más alto de abstracción que un diagrama de clase.

2.5.1. Diagrama de componentes para el DMC

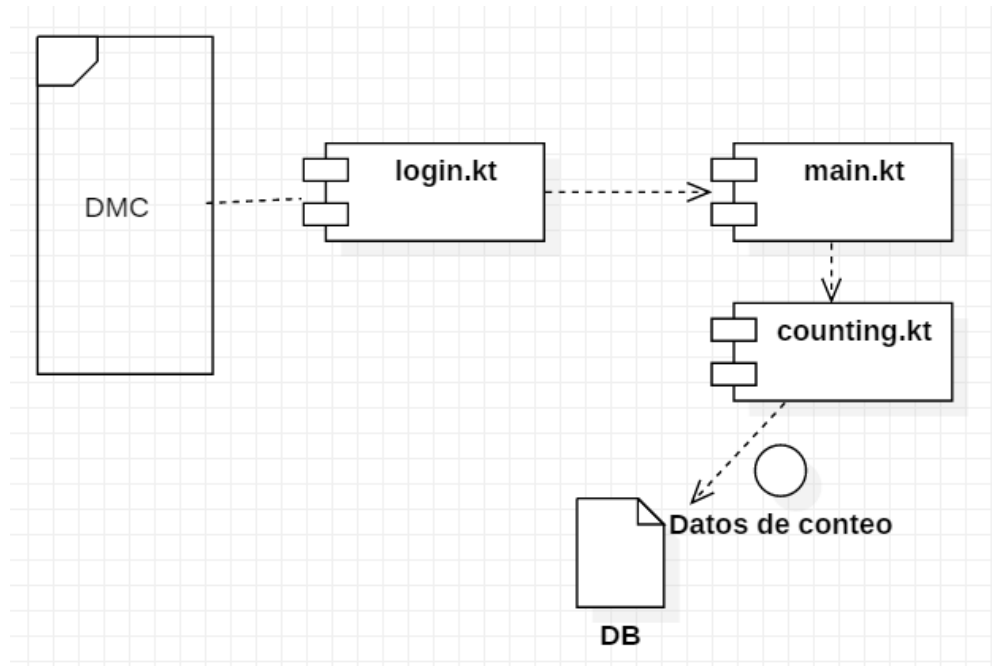


Ilustración 18 - Diagrama de componentes para el DMC

2.5.2. Diagrama de componentes para la AWGI

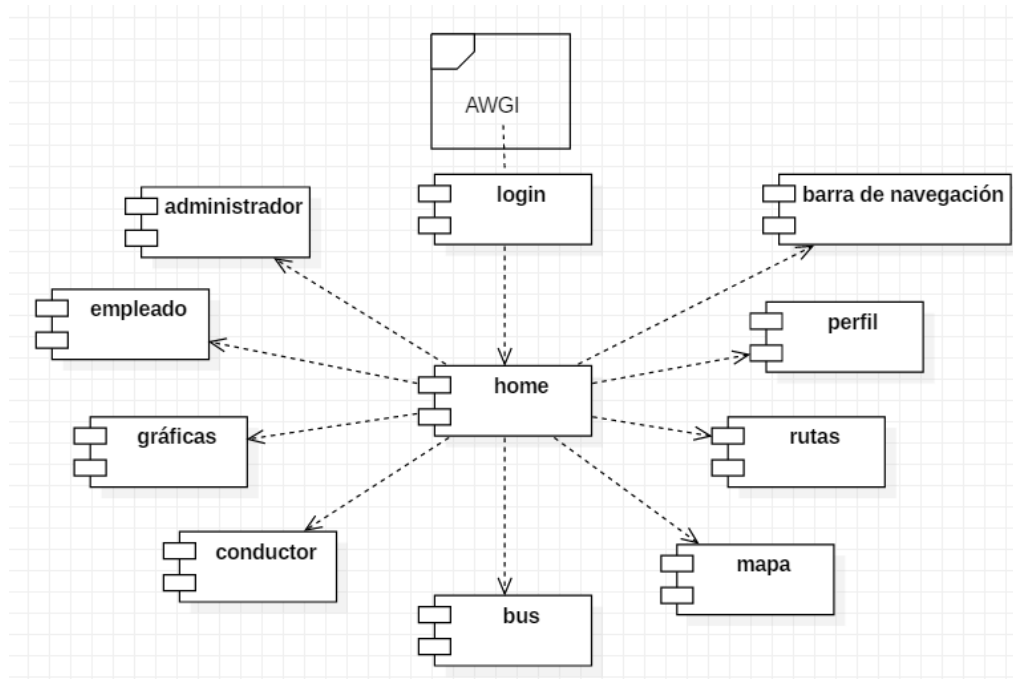


Ilustración 19 - Diagrama de componentes para la AWGI

2.6. Diagrama de despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos.

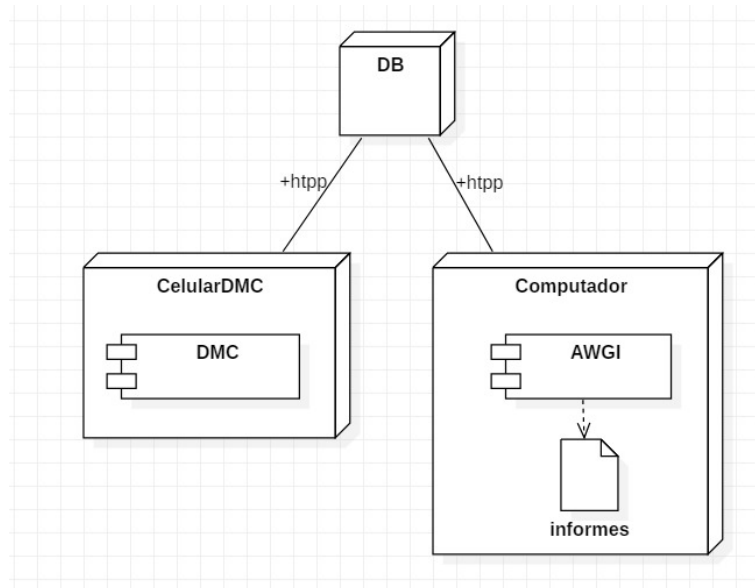


Ilustración 20 - Diagrama de despliegue

2.7. Sistema prototipo como aplicación distribuida

Una aplicación distribuida es una aplicación con distintos componentes que se ejecutan en entornos separados, normalmente en diferentes plataformas conectadas a través de una red.

La distribución habla de que las partes o componentes se encuentran en entornos separados, entonces para realizar la separación física es necesario entender la separación lógica.

Para el desarrollo del prototipo, se implementa una arquitectura del tipo Cliente/Servidor. Los elementos se distribuyen de la siguiente manera:

- La interfaz de usuario que va a tener el DMC, por medio del cual los conductores van a poder poner en marcha el conteo.
- Los servidores web en los cuales se alojará la base de datos para el acceso a datos.
- Interfaz de usuario para la AWGI, a partir de la cual los usuarios tendrán acceso a todas las funcionalidades

Capítulo 3

IMPLEMENTACIÓN

Inicialmente en este capítulo, basándose en los requerimientos del sistema y en los esquemas de diseño trazados, se continúa con el análisis de las principales tecnologías disponibles en el mercado que permitan cumplir con los requerimientos y objetivos del proyecto, así como los criterios de selección usados en la selección de las mismas.

Para la implementación del sistema propuesto en el presente trabajo de grado, se definen dos subprototipos que permitirán el escalamiento del problema y la interacción con las tecnologías involucradas, para finalmente hacer una integración y generar un prototipo final que cumpla con los requerimientos descritos en el presente documento.

3.1. Análisis de tecnologías disponibles para el conteo de pasajeros

Las tecnologías que se describen a continuación, son las que presentan mayor disponibilidad en el mercado, y permiten desarrollar el proyecto con las características y requerimientos expuestos anteriormente, es decir, son las más adecuadas para el desarrollo del DMC el cual será el encargado de la captura física de datos de abordajes.

3.1.1. Tecnologías para el control de acceso

Actualmente, en diversos escenarios es de vital importancia disponer de un sistema de control de acceso (SCA), el cual se define como un sistema de seguridad y monitoreo que limite el libre ingreso de personas a un lugar determinado. Todo esto con el fin de lograr un entorno más seguro, además de prevenir fraudes, daños y pérdidas.

Por tal razón, cada vez se dedica más tiempo, dinero y esfuerzo para desarrollar directivas y controles de seguridad apropiados. Sin embargo, esto requiere analizar las necesidades específicas del entorno en el cual se quiere implementar un SCA, y así determinar el nivel de seguridad y control requerido, por lo tanto, es necesario crear un plan de seguridad, el cual debe ser cuidadosamente diseñado, revisado y actualizado constantemente [13]. El control de acceso ha evolucionado a la par de la tecnología y las necesidades de su entorno, con lo cual han surgido diferentes tipos de SCA, uno de ellos es el control de acceso electrónico el cual surgió para resolver algunos problemas relacionados con el uso de llaves, cerraduras y cerrojos mecánicos, debido a que estos presentaban problemas de seguridad, ya que las llaves podían ser copiadas fácilmente, además de carecer de un control y gestión de su utilización [14]. El control de acceso electrónico a su vez ha hecho uso de diferentes tecnologías, cada una de ellas con sus características específicas las cuales se detallan a continuación:

3.1.1.1. NFC

NFC (*Near Field Communication*) es una tecnología de comunicación inalámbrica de corto alcance y alta frecuencia que permite el intercambio bidireccional de datos entre dispositivos a una distancia corta, menor a 10 cm, además, ejecuta la comunicación inalámbrica entre dos dispositivos habilitados para NFC [30]. Su desarrollo data del año 2002 y los promotores fueron Phillips y Sony, principalmente para conseguir compatibilidad con sus tecnologías, Mifare y FeliCa respectivamente. A finales del año 2003, fue aprobada como el estándar ISO 180926 (*International Organization for Standardization*).

NFC-*fórum* define NFC como una tecnología basada en estándares de conectividad que permite soluciones tales como; controles de acceso, pagos electrónicos, transporte, publicidad, marketing, acceso a sitios web, identificación, etc.

La tecnología NFC puede operar en dos modos distintos:

- **Modo pasivo:** Un campo electromagnético es generado por el dispositivo iniciador mientras que el dispositivo de destino se comunica con él modulando la señal recibida.
- **Modo activo:** En este modo, tanto el dispositivo iniciador como el destino se comunican generando su propio campo electromagnético.

Dentro de la tecnología de NFC, se encuentran los dispositivos NFC que son los componentes que actúan en la comunicación. Particularmente existen tres tipos, el teléfono móvil con NFC presente en muchos modelos de teléfonos actualmente, el lector de NFC que realiza la transferencia de datos con un componente de NFC, por ejemplo, en un punto de venta de almacén se puede realizar pagos en un lector NFC sin contacto, y por último, la etiqueta pasiva (ya que no tiene una fuente de alimentación) NFC que comienza una comunicación una vez entra en el campo de proximidad establecido por los estándares.

Cuando dos dispositivos con NFC se aproximan cerca a los 10 cm para que sus campos magnéticos entren en contacto, se produce un acoplamiento por inducción magnética para transferir energía y datos entre ellos. Este acoplamiento magnético es la gran diferencia entre NFC y otros dispositivos como Bluetooth.

NFC es una tecnología que se ha incluido en los móviles de última generación y facilita el intercambio de datos entre diferentes terminales a la vez que se pueden emular, un ejemplo de esto son las tarjetas bancarias en el propio teléfono como se puede evidenciar en la Ilustración 21.



*Ilustración 21 - Pago con móvil NFC
Tomado de [31]*

Algunos de los beneficios que la tecnología ofrece para los consumidores y empresas son:

- **Intuitivo:** Las interacciones con la tecnología requieren solo un toque.
- **Versátil:** Ideal para una amplia gama de industrias, entornos y usos.
- **Basada en estándares abiertos:** Las capas subyacentes a la tecnología siguen estándares universales implementando las normas ISO, ECMA y ETSI (*European Telecommunications Standards Institute*).
- **Tecnologías habilitadas:** NFC facilita la configuración rápida y sencilla de las tecnologías inalámbricas como Bluetooth o Wi-Fi (*Wireless Fidelity*).
- **Intrínsecamente seguro:** Debido a que, el modo de comunicación es de corto alcance, dificultando la captación de los datos dentro de la misma.
- **Seguridad:** La tecnología ha incorporado capacidades para soportar aplicaciones seguras.

3.1.1.2. MIFARE

MIFARE es una tecnología de tarjetas inteligentes sin contacto, la cual es utilizada ampliamente en el control de acceso a diferentes espacios como: edificios, oficinas, sectores e incluso en sistemas de pago para el transporte público, entre otras aplicaciones [32].

Estas tarjetas incluyen un chip y una antena en su interior como se observa en la Ilustración 22, de forma que, permite a la tarjeta comunicarse con el receptor o lector de radiofrecuencia a una distancia de entre 2,5 cm y 10 cm. Cuentan además con una capacidad de memoria que oscila, dependiendo del modelo, entre 1Kb y 8Kb; dicha memoria sirve para registrar la información de usuario, números de serie, claves, etc. [33].



Ilustración 22 - Tarjeta MIFARE

Tomado de: [34]

La tecnología MIFARE fue lanzada al mercado por la compañía *NXP Semiconductors*, la cual ha logrado que sus tarjetas sean ampliamente distribuidas, logrando ser una de las tecnologías más extendidas en el mundo. La tecnología MIFARE se basa en el estándar ISO 14443 Tipo A; una de las características más importantes de esta tecnología, es su rápida evolución en aspectos de memoria, seguridad y servicios, En la Ilustración 23 se pueden observar con más detalle las diferentes referencias de esta tecnología y sus características.

	Memoria	Organización	Tipo memoria	UID	Baudrate	Claves de acceso	Seguridad	Generador número aleatorio
Mifare Classic 1K	1.024 bytes	16 sectores con 64 bytes	EEPROM 100.000 cidos	4 bytes (NUID) 7 bytes (UID)	106 kbits/segundo	2 claves por sector	Crypto1	Sí
Mifare Classic 4K	4.096 bytes	32 sectores con 64 bytes y 8 sectores con 256 bytes	EEPROM 100.000 cidos	4 bytes (NUID) 7 bytes (UID)	106 kbits/segundo	2 claves por sector	Crypto1	Sí
Mifare Ultralight	64 byte (32 bits OTP)	16 páginas con 4 bytes	EEPROM 10.000 cidos	7 bytes (UID)	106 kbits/segundo	No	No	No
Mifare Ultralight C	192 byte (32 bits OTP)	48 páginas con 4 bytes	EEPROM 10.000 cidos	7 bytes (UID)	106 kbits/segundo	1 clave	Autenticación DES-3DES	Sí
Mifare Plus S 2K	2.048 bytes	32 sectores con 64 bytes	EEPROM 200.000 cidos	4 bytes (NUID) 7 bytes (UID)	Desde 106 a 848 kbits/segundo	2 claves por sector	Crypto1 o AES	Sí
Mifare Plus S 4K	4.096 bytes	32 sectores con 64 bytes y 8 sectores con 256 bytes	EEPROM 200.000 cidos	4 bytes (NUID) 7 bytes (UID)	Desde 106 a 848 kbits/segundo	2 claves por sector	Crypto1 o AES	Sí
Mifare Plus X 2K	2.048 bytes	32 sectores con 64 bytes	EEPROM 200.000 cidos	4 bytes (NUID) 7 bytes (UID)	Desde 106 a 848 kbits/segundo	2 claves por sector	Crypto1 o AES	Sí
Mifare Plus X 4K	4.096 bytes	32 sectores con 64 bytes y 8 sectores con 256 bytes	EEPROM 200.000 cidos	4 bytes (NUID) 7 bytes (UID)	Desde 106 a 848 kbits/segundo	2 claves por sector	Crypto1 o AES	Sí
Mifare DESFire EV1 2K	2.048 bytes	Sistema de archivos	EEPROM 500.000 cidos	7 bytes (UID)	Desde 106 a 848 kbits/segundo	14 claves por aplicación	DES, 3DES, AES 128	Sí
Mifare DESFire EV1 4K	4.096 bytes	Sistema de archivos	EEPROM 500.000 cidos	7 bytes (UID)	Desde 106 a 848 kbits/segundo	14 claves por aplicación	DES, 3DES, AES 128	Sí
Mifare DESFire EV1 8K	8.192 bytes	Sistema de archivos	EEPROM 500.000 cidos	7 bytes (UID)	Desde 106 a 848 kbits/segundo	14 claves por aplicación	DES, 3DES, AES 128	Sí

Ilustración 23 - Características MIFARE
Tomado de: [35]

Como se ha dicho MIFARE es una tecnología que se ha extendido ampliamente y hoy día se encuentra en diferentes medios, prestando variados servicios siendo utilizada por diferentes empresas y mercados para realizar micro-pagos, un ejemplo de esto son los sistemas de transportes quienes han acogido esta tecnología para realizar el proceso de *ticketing*³, y así controlar el pago e ingreso al servicio, algunas de las principales características de esta tecnología son:

- **Seguridad:** cifrado de la información contenida en el chip.
- **Rapidez:** lectura y escritura de datos entre los 80 y 106 Kbits/s.
- **Facilidad:** con 10 cm de cercanía el lector puede captar los datos.

³ Proceso de compra de ticket o recarga de tarjeta para acceder al servicio de transporte público.

- **Fiabilidad:** el sistema lee una sola tarjeta cada vez, aunque se acerquen varias.
- **Prevención del fraude:** los lectores están en la capacidad de detectar tarjetas falsificadas.
- **Resistencia:** las tarjetas cuentan con una vida operativa alrededor de los 10 años.
- **Versatilidad:** la información contenida en el chip puede ser reescribir hasta 100.000 veces.
- **Personalización:** las tarjetas actuales son fabricadas con un material plástico (PVC), el cual está disponible en varios colores y permite la imprenta de cualquier imagen, logotipo o contenido sobre su superficie.
- **Compatibilidad:** los sistemas actuales cuentan con la capacidad de ser actualizados, y así estar en capacidad de leer futuras generaciones de tarjetas MIFARE, de esta forma no es necesario reemplazarlos.
- **Precio:** el costo de producir una tarjeta MIFARE es bajo en comparación con otras tecnologías, alrededor de 5 dólares.

3.1.2. Revisión de dispositivos tecnológicos afines

Los dispositivos tecnológicos están conformados por un determinado número de elementos más sencillos o básicos conocidos como dispositivos electrónicos, los cuales realizan una acción o conjunto de acciones de forma individuales que en conjunto buscan desarrollar una función o actividad más compleja [36], un ejemplo de esto, son los *smartphones* que avanzan vertiginosamente en la funciones que pueden realizar y en su capacidad de procesamiento, todo esto gracias a la evolución individual en los diferentes dispositivos electrónicos que los conforman, debido a los avance en la tecnología, una de las ramas más importante de dispositivos electrónicos es la de los sensores, los cuales se describen a continuación.

Un sensor es un dispositivo con la capacidad de transformar magnitudes físicas o químicas, llamadas variables de instrumentación, en magnitudes eléctricas por medio de un transductor, el cual a diferencia del sensor no se encuentra todo el tiempo en contacto con las variables de instrumentación que pueden ser: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, pH, etc. Una magnitud eléctrica obtenida puede ser resistencia, capacidad, tensión o corriente eléctrica, entre otras [36].

En la práctica, los sensores se usan para diferentes fines, desde los que son instalados en los vehículos para detectar el sobrepaso de los límites permitidos de velocidad, hasta los sensores que se instalan en la entrada de las viviendas y reaccionan con el movimiento, de forma que, si una persona se acerca al dispositivo, éste emite una señal y se enciende una alarma [37].

Para efectos del presente trabajo de grado, se enuncian a continuación algunos dispositivos tecnológicos los cuales haciendo uso de sus características y de elementos como sensores hacen viable su implementación en el conteo de pasajeros.

3.1.2.1. Raspberry

Raspberry Pi es una computadora de placa reducida, computador de placa única o reducida también conocido como SBC (Single Board Computer), el cual tiene un tamaño reducido, un bajo costo y un consumo energético considerablemente bajo, sus primeros modelos fueron lanzados al mercado en abril de 2012 buscando mejorar las habilidades de programación y comprensión de hardware al nivel preuniversitario, gracias a su pequeño tamaño y precio accesible, fueron adoptados rápidamente por entusiastas y fabricantes de herramientas electrónicas, los cuales requerían más que un microcontrolador básico, lo cual ocasionó su rápida expansión a diferentes áreas y aplicaciones [38].

Además de un microordenador, Raspberry incorpora diversas funciones de electrónica a sus placas, las cuales cuentan con pines para: GPIO (General Purpose Input/Output), comunicación UART (Universal Asynchronous Receiver-Transmitter) y SPI (Serial Peripheral Interface), I²C (Inter-Integrated Circuit), entre otros. Estas funciones hacen que pueda ser empleado en proyectos de electrónica y robótica interactuando con diferentes sensores y actuadores [39]. Este tipo de microordenadores se ejecutan generalmente con sistemas operativos basados en distribuciones Linux y están íntimamente relacionados con el software libre. No obstante, el desarrollo en sí de Raspberry no es de hardware libre.

Es importante resaltar que las Raspberry no son equiparables con un computador moderno y convencional, aunque los últimos modelos incorporen procesadores muy superiores a las primeras versiones; la capacidad de procesamiento de estos microordenadores es comparable de un *smartphone* moderno.

La empresa Raspberry Pi Foundation, además de desarrollar y comercializar microordenadores, ha creado un mercado de diferentes módulos adaptables a sus placas (teclados, pantallas LCD (Liquid Cristal Display), pantallas TFT (Thin-Film Transistor), cámaras, diversos sensores, entre otros), los cuales brindan mucha facilidad a la hora de realizar desarrollos, ya que estos cuentan con bajos costos, son fácilmente ensamblables y configurables, esto último, como se mencionó anteriormente, Raspberry está ligado con el software libre, con lo cual se pueden encontrar diversas fuentes que brindan código para la configuración de los módulos, además, las placas cuentan su propio compilador y lenguaje de programación, los cuales son intuitivos y con amplia documentación en la web.

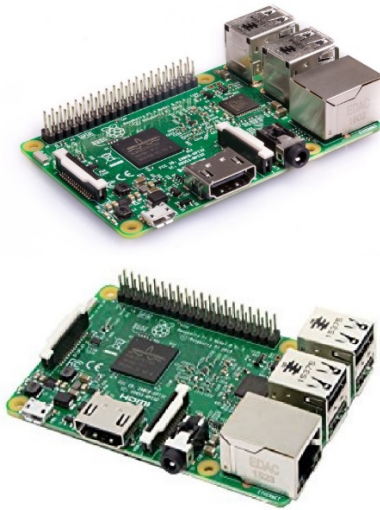


Ilustración 24 - Raspberry Pi 3 Modelo B
Tomado de: [40]

Uno de los modelos principales de Raspberry debido a su capacidad de procesamiento y su bajo costo es Pi 3 modelo b, Ilustración 24, el cual cuenta con características tales como:

- CPU: Quad-Core Cortex A7 a 900MHZ
- GPU: Video Core IV de doble núcleo
- RAM: 1GB DDR2
- Puertos:
 - 4 - USB 2.0
 - 1 - 40 GPIO pin
 - 1 - HDMI 1.4
 - 1 - Ethernet
 - 1 - Combo audio/mic
 - 1 - Interfaz de cámara CSI (Camera Serial Interface)
 - 1 - Interfaz de Pantalla DSI (Display Serial Interface)
 - 1 - Micro SD
 - 1 - Núcleo Grafico 3D
 - Módulo Bluetooth
 - Módulo de WI-FI en la banda de 2.4GHz

Gracias a las características de la Rapsberry pi 3 modelo B, y a su capacidad de integrar o adaptar gran variedad de periféricos y módulos externos propios de Rapsberry o de otro tipo, los cuales son de fácil acceso en el mercado, ha llevado a sea utilizada en diversos proyectos y gran variedad de áreas.

Uno de estos periféricos son los sensores ópticos, los cuales permiten la detección de eventos, o personas, lo cual es de gran interés para el presente trabajo, a continuación, se describen algunos de estos:

3.1.2.1.1. **Módulo** cámara V2 Rapsberry

En la Ilustración 25 Se puede observar este módulo de alta calidad de 8 megapíxeles con sensor de imagen Sony IMX219 diseñado a medida para Raspberry Pi, con una lente de foco fijo, con capacidad de tomar imágenes estáticas de 3280x2464 pixeles, además de capturar vídeo de 1080p30, 720p60. Se conecta a la placa de Raspberry Pi a través de uno de los conectores en la parte superior de la placa y utiliza la interfaz dedicada CSI, diseñada especialmente para las interfaces con cámaras.

El módulo de cámara V2 de Raspberry es uno de los más utilizados en proyectos que requieran procesamiento de imágenes, un ejemplo de esto es *Footfall* [41] el cual es un proyecto donde se implementa un contador de personas utilizando el módulo de cámara V2, este proyecto es de código abierto, sin embargo, dado las referencias del mismo su fiabilidad en baja.

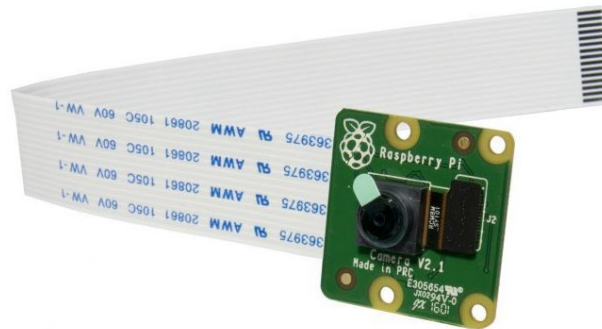


Ilustración 25 - Modulo cámara Raspberry V2.
Tomado de: [42]

3.1.2.1.2. **Kinect**

Kinect fue creado por Alex Kipman y desarrollado por Microsoft para la videoconsola Xbox 360, originalmente para revolucionar la forma en que las personas juegan y como experimentan el entretenimiento, ya que este dispositivo permite que los usuarios puedan interactuar con los juegos mediante su cuerpo de forma natural.

La comprensión del lenguaje corporal humano es la clave de esta tecnología, el estudio de esta siempre ha sido un campo de investigación activo en la visión por computadora, el cual es extremadamente difícil empleando cámaras de video. El sensor Kinect permite que la

computadora detecte directamente la tercera dimensión de las personas y el entorno, además de ser capaz de entender cuando los usuarios hablan, sabe quiénes son, cuando caminan hacia él y algo que es particularmente importante, el Kinect puede interpretar los movimientos y traducirlos a un formato que los desarrolladores pueden personalizar según las necesidades [43].

El Kinect contiene principalmente un sensor de profundidad, una cámara a color y una matriz de cuatro micrófonos que proporcionan capacidades de captura de movimiento 3D, reconocimiento facial y reconocimiento de voz de cuerpo completo como se puede observar en la Ilustración 26.

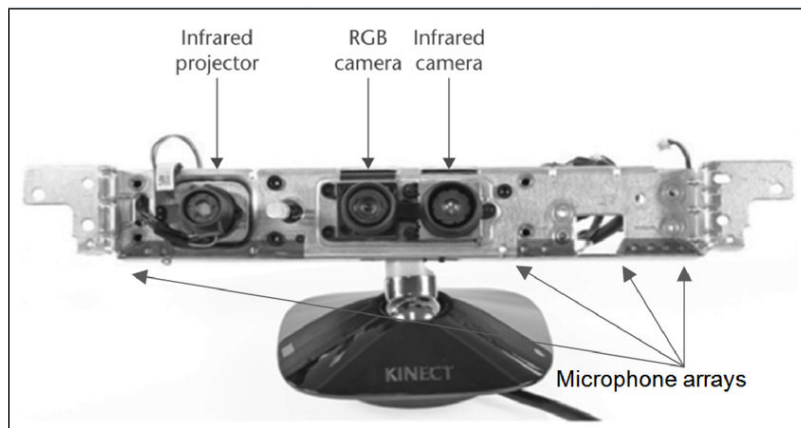


Ilustración 26 - Microsoft Kinect Sensor.
Tomado de: [44]

Gracias a toda la información que captura este dispositivo, los desarrolladores de software pueden hacer uso de él para programar toda una serie de aplicativos cuyo activo principal es la interacción con los elementos “virtuales” a través de los distintos movimientos del cuerpo humano. Es ahí donde entra en juego el SDK (*Software Development Kit*) para Kinect, que Microsoft ha puesto al alcance de los desarrolladores de todo el mundo.

El SDK es una librería que facilita diferentes funciones que ayudan a interactuar con el dispositivo Kinect. Por ejemplo, cuando es detectado el esqueleto humano, se brinda información detallada de la posición exacta en el plano (X, Y, Z) de todas y cada una de las articulaciones en las que divide el esqueleto humano. Es gracias a esa información que se pueden desarrollar aplicaciones que funcionan con la interacción del cuerpo humano, sin necesidad de ratones ni *touchpads*.

La primera versión del Kinect (Kinect 1.0), fue presentada en 2009 con una cámara VGA (*Video Graphics Array*) por defecto a 640x480 píxeles, con la posibilidad de trabajar a 1280x1024 a costa de una menor tasa de refresco. El microchip de la primera versión deja gran parte del procesamiento a la consola. Para el año 2013 fue anunciado el [45] lanzamiento de la versión 2.0, la cual se diferencia de su predecesor incorporando una cámara *time-of-flight* de alta resolución que permite al Kinect capturar más detalles con gran precisión y mayor resolución,

estimando distancias de cuerpos calculando el tiempo transcurrido entre la emisión y la recepción de un haz de luz infrarrojo. La versión 2.0 permite reproducir una escena con tres veces más fidelidad que la versión 1.0.

Acorde a los argumentos presentados anteriormente, Kinect es un sensor que puede ser utilizado para múltiples propósitos y en diversas áreas, debido a su facultad de conectividad y desarrollo de software en múltiples plataformas, existen diversos trabajos documentados que utilizan el Kinect para el conteo de personas, haciendo uso de placas Raspberry u otro tipo de computadoras, tal es el caso de un proyecto de fin de carrera titulado “Estudio sobre el uso de sensores RGB-D para el conteo de personas y sus caracterización” [46], en donde se presenta un contador de personas automático que utiliza técnicas de visión por computador para realizar dicho conteo utilizando secuencias de imágenes recogidas por un sensor Kinect, así como un estudio sobre la caracterización del sexo de las personas capturadas.

3.1.2.1.3. Sensores infrarrojos

En el mercado existen diferentes dispositivos que hacen uso de sensores infrarrojos para contabilizar un evento del paso o interrupción en su haz de luz, y generar una acción a partir de esto, en estos dispositivos se pueden encontrar una gran variedad de funciones, complejidad en su diseño y costos, Rapsberry Pi 3B es uno de estos dispositivos y están disponibles diversos proyectos que hacen uso de estos sensores.

Profundizando más en esta tecnología se tiene que los sensores infrarrojos son de medición de distancia, se basan en un sistema de emisión/recepción de radiación lumínica en el espectro de los infrarrojos. Este tipo de sensor presenta el inconveniente de ser sensible a la luz ambiente como consecuencia de que los rayos de sol también emiten en el espectro de luz infrarroja. Por este motivo, son sensores que se utilizan habitualmente en entornos con iluminación artificial de forma predominante (interiores) [47]. Para el caso particular del conteo de personas que ingresan a un autobús, los sensores infrarrojos se montan lateralmente en la puerta de entrada, el dispositivo envía un haz de infrarrojo directo y cuenta a la persona que se mueve a través de la puerta una vez que ha traspasado el haz. Las ventajas más importantes que se pueden encontrar con estos dispositivos respecto a otras tecnologías para el conteo de personas son su coste y simplicidad, sin embargo los sensores infrarrojos son menos precisos ya que no pueden reconocer la dirección de movimiento, por lo tanto, cuentan a las personas de igual manera si entran o salen del vehículo, generalmente los sistemas que emplean este tipo de sensores se ven obligados a dividir el conteo en dos para poder estimar el número de personas que viajaron.

Otro aspecto para tener en cuenta de estos dispositivos, es su baja fiabilidad, ya que tal tecnología no puede diferenciar entre la entrada de una persona o un grupo de personas al mismo tiempo, de tal forma que, si dos personas pasan por el umbral, la cuenta va a ser de un pasajero, influyendo directamente en el conteo real.

3.1.2.2. Smartphones

En general, un dispositivo móvil se puede definir como un aparato de dimensiones pequeñas, que cuenta con alguna capacidad de procesamiento, además tiene conexión permanente o intermitente a una red, una memoria limitada, y ha sido diseñado específicamente para cumplir una función, pero puede llevar a cabo otras funciones más generales. Ahora bien, un teléfono móvil es un dispositivo inalámbrico electrónico basado en la tecnología de ondas de radio, que tiene la misma funcionalidad que cualquier teléfono de línea fija. Su principal característica es la portabilidad, ya que la realización de llamadas no es dependiente de ningún terminal fijo y no requiere ningún tipo de cableado para llevar a cabo la conexión a la red telefónica. Aunque la función principal de un teléfono móvil es la comunicación de voz, el rápido desarrollo ha incorporado funciones adicionales como mensajería instantánea, agenda, juegos, cámara fotográfica, agenda, acceso a internet, reproducción de video, entre otras [48].

Inicialmente los teléfonos móviles sólo permitían realizar llamadas de voz y enviar mensajes de texto, al tiempo que la tecnología fue avanzando se incluyeron nuevas aplicaciones como el acceso WAP (*Wireless Application Protocol*) permitiendo que estos dispositivos prestaran nuevas funcionalidades, y que estas evolucionaran rápidamente, en pocos años se dio el cambio del teléfono móvil que solo permitía hacer llamadas, al *smartphone* que cuenta con un gran número de funciones y servicios [49]. Dicho lo anterior, se define un *smartphone* como un dispositivo electrónico que funciona como un teléfono móvil, pero con características similares a las de un ordenador personal. Los *smartphone* tienen un diseño particular como se observa en la Ilustración 27, el cual se propagó con rapidez desde el año 2007, una característica importante de casi todos los teléfonos inteligentes es que permiten la instalación de programas para incrementar el procesamiento de datos y la conectividad, estas aplicaciones pueden ser desarrolladas por el fabricante del dispositivo, por el operador o por un tercero.



Ilustración 27 - Diseño típico de un Smartphone

El auge presente en el desarrollo de aplicaciones móviles y en el mercado de estas sigue creciendo exponencialmente, llegando incluso a liderar en sectores que hasta hace poco eran liderados por otros productos. Con toda esta diversificación en las funciones que pueden realizar las aplicaciones móviles y debido al avance tecnológico en los dispositivos móviles que permite grandes y rápidas mejoras en la capacidad de procesamiento, almacenamiento y sus periféricos, la aplicaciones móviles se han abierto a un gran campo de acción [50].

Es importante resaltar que todo *smartphone* cuenta con un sistema operativo (SO), él cual es un software que proporciona un acceso sencillo y seguro al hardware, ocultando al usuario detalles de la implementación particular. El uso de un determinado SO va a determinar las capacidades multimedia de los dispositivos y la forma de éstas interactuar con el usuario, además de la forma como se desarrollan las aplicaciones para estos, actualmente en el mercado, existe son 2 grandes SO los cuales abarcan 99.6% de este para el año 2018 [51]. A continuación, se realizará una breve descripción de estos 2 sistemas operativos:

3.1.2.2.1. Android

Android es una pila de software de código abierto basado en Linux que posee el 81,7% del mercado y fue creado para una variedad amplia de dispositivos [51]. En la Ilustración 28 se muestran los componentes principales de la plataforma.

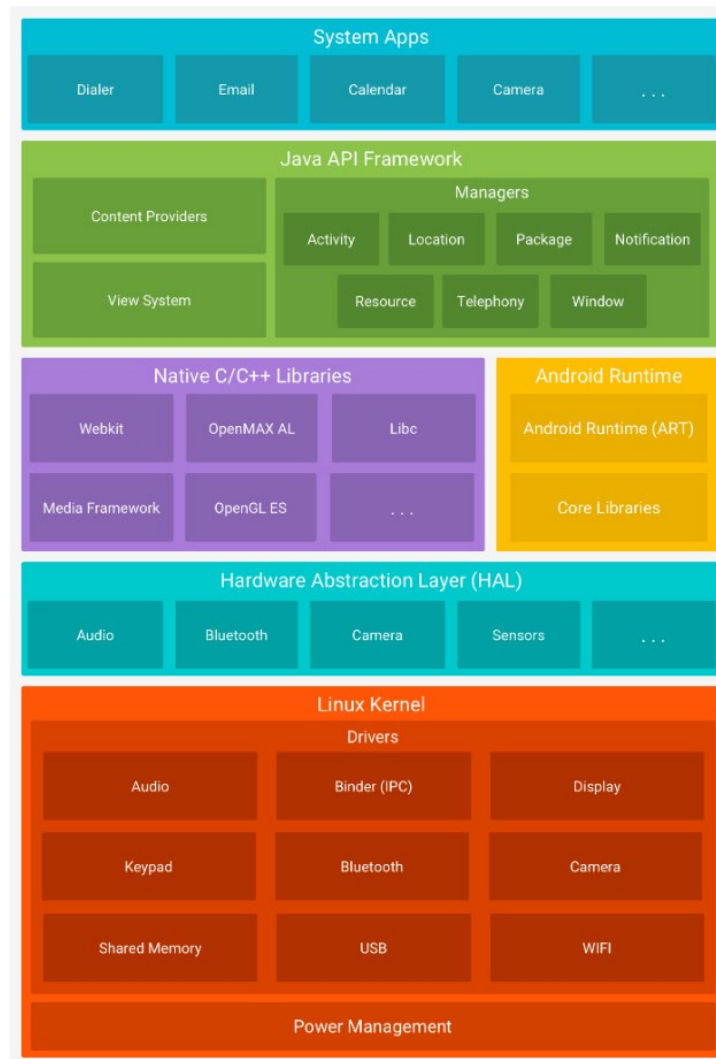


Ilustración 28 - Pila de software de Android
Tomado de: [52]

La base de la plataforma es el kernel, cuyo uso permite que Android aproveche funciones de seguridad, claves y al mismo tiempo permite a los fabricantes de dispositivos desarrollar controladores de hardware para un kernel conocido. La capa de abstracción de hardware consiste en varios módulos de biblioteca y cada uno de estos implementa una interfaz para un tipo específico de componente hardware, como el módulo de la cámara o del bluetooth. Muchos componentes y servicios centrales del sistema Android, se basan en código nativo que requiere bibliotecas nativas escritas en C y C++. La plataforma Android proporciona la API (*Application Programming Interface*) del framework de Java para exponer la funcionalidad de algunas de estas bibliotecas nativas a las apps.

Todo el conjunto de funciones del sistema operativo Android está disponible mediante API escritas en el lenguaje Java. Las herramientas de Android SDK compilan el código de los desarrolladores, junto con los archivos de recursos y datos, en un APK (Android Application Package), que es un archivo de almacenamiento con el sufijo (.apk). Dicho archivo incluye todos los contenidos de una aplicación de Android y es el archivo que usan los dispositivos con ese sistema operativo para instalar aplicaciones.

3.1.2.2.2. iOS

iOS es un sistema operativo móvil desarrollado por la compañía Apple Inc, originalmente fue diseñado para el iPhone (iPhone OS), pero debido a su rápida y creciente acogida se extendió a otra categoría de dispositivos como el iPod touch y el iPad. Actualmente es el segundo sistema operativo móvil más utilizado, solo superado por Android, obteniendo una cuota de mercado de aproximadamente 17,9% al año 2018 [51]. La última versión de este sistema operativo es el iOS 11, éste se deriva de su contraparte para equipos de escritorio MacOS, el cual es su vez, está basado en Darwin BSD (Berkeley Software Distribution), y por lo cual es un sistema operativo Tipo Unix.

iOS cuenta con cuatro capas de abstracción: la capa del núcleo del sistema operativo, la capa de "Servicios Principales", la capa de "Medios" y la capa de "Cocoa Touch". Como se puede observar en la Ilustración 29. [53]

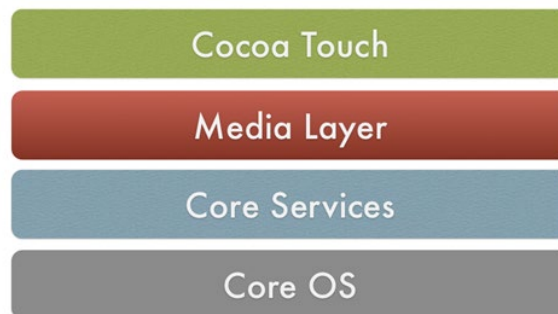


Ilustración 29- Capas de abstracción IOS
Tomado de: [53]

La arquitectura de iOS es una arquitectura en capas, donde cada capa cumple una tarea determinada. La capa Core OS contiene las características de bajo nivel sobre las que se basan la mayoría de las demás tecnologías (BSD, *Sockets*, *Security*, *Power management*, etc.), la capa Core Services proporciona una abstracción sobre los servicios proporcionados en la capa anterior (*Address Book*, *File Access*, *SQLite*, *Core Location*, *URL Utilities*, etc.), la siguiente capa (Media layer) proporciona acceso a los servicios multimedia (audio, video, PNG, OpenGL, etc.), finalmente la capa Cocoa Touch proporciona una capa de abstracción adicional e incluye todo el Core de librerías de programación (eventos *multi-touch*, acelerómetro, localización, etc.). Con esto se puede observar que las capas inferiores brindan los servicios básicos en los que se basa toda aplicación y la capa de nivel superior proporciona gráficos sofisticados y servicios relacionados con la interfaz.

3.2. Evaluación y selección de la tecnología

Luego de describir las diferentes tecnologías con las cuales se podría desarrollar el prototipo DMC el cual como se mencionó anteriormente es el componente hardware-software encargado de contabilizar el ascenso y descenso de pasajeros en un bus correspondiente, se procede a evaluar cada una de estas para determinar la más conveniente para cumplir con los requerimientos y expectativas del proyecto.

3.2.1. Evaluación y selección de la tecnología

A la hora de evaluar y seleccionar la tecnología más adecuada para el desarrollo del proyecto se deben tener en cuenta los aspectos dados por la empresa Movilidad Futura S.A.S desde la captura de requerimientos, ya que son de importancia no solo para esta empresa, sino, para diferentes empresas que brindan el servicio TPC.

Dicho lo anterior las tecnologías NFC y MIFARE serán descartadas como posibles tecnologías para la implementación del proyecto, ya que no cumplen con las características de estar desligadas al cobro y ser transparentes al usuario, aspectos importantes ya que la empresa Movilidad Futura S.A.S, expuso que la ciudad de Popayán no tiene las capacidades actualmente de implementar un sistema TPC del tipo BTR, el cual necesita estaciones o centrales donde las personas carguen sus tarjetas (NFC, MIFARE) para realizar el pago de sus tiquetes, además, las estaciones de abordaje deben contar algún tipo de control en su entrada para realizar el cobro .

3.2.1.1. Criterios de evaluación

Se deben elegir diferentes criterios frente a los cuales se evaluará cada una de las tecnologías, a continuación, se describirán los seleccionados dentro del proyecto, a los cuales se les asignará una calificación de 1, 2 o 3 puntos (bajo, medio o alto) dependiendo de cómo se comporte la tecnología frente al criterio a evaluar.

- **Costos:** hace referencia al valor que tendrá el producir cada uno de los prototipos hardware, basándose exclusivamente en los componentes de cada una de las tecnologías. Es importante resaltar que solo se tendrá el costo de la tecnología, ya que al ser un sistema prototipo no se analiza su precio de implementación real dentro del sistema TPC.

La calificación se dará tomando como base un valor de \$ 240.000 COP (Pesos Colombianos) el cual, fue un valor propuesto por la empresa Movilidad Futura S.A.S en una de las reuniones como precio de bajo costo, explicando que este precio es el valor promedio en el que oscilan diferentes herramientas o dispositivos externos (hardware) que se adaptan actualmente a los vehículos como GPS, cámaras, etc. También la empresa manifestó que un precio considerado alto es el que pasara de los 600.000 COP.

En la Tabla 14, se puede observar los costos específicos para producir el DMC con cada una de las tecnologías y su calificación, ya que todas las tecnologías analizadas no están por debajo del precio determinado como bajo, y de igual forma ninguna está por encima, se califican con una puntuación media.

Tabla 14 - Costos tecnologías

TECNOLOGÍA		COSTOS	Calificación
Rapsberry PI-3B	Modulo Cámara V2	\$ 280.000	Medio (2 puntos)
	Kinect	\$ 260.000	Medio (2 puntos)
Smartphone (Huawei Y6 II - 2018)		\$ 350.000	Medio (2 puntos)

Se seleccionó el Huawei Y6 II – 2018 luego de realizar una búsqueda entre dispositivos móviles de gama media, y hacer una comparativa entre prestaciones y su valor comercial.

- **Escalabilidad:** Entendiendo este criterio como la propiedad de un sistema de aumentar la capacidad de trabajo o el tamaño del mismo sin comprometer el funcionamiento y calidad normales de este [54]. Dentro del proyecto se evaluará la capacidad que tiene cada una de las tecnologías, de actualizarse o implementar mejoras, de forma que no se afecte el sistema ya existente.

Este criterio se evaluó en cada una de las tecnologías analizando su capacidad de procesamiento y la resolución de los sensores ópticos, ya que estos 2 aspectos, son los que permiten realizar avances mejorar en los algoritmos desarrollados, y en las funciones del prototipo sin incurrir en cambios en el hardware, con respecto a las siguientes características para cada tecnología se les dio una calificación:

- **Huawei Y6 II – 2018:**

- 16 megapíxeles
- 2 GB - RAM
- 1.1 GHz

- **Rapsberry Pi 3 con Kinect:**
 - 16 megapíxeles
 - Sensor de profundidad
 - 900 MHz
 - 1 GB – RAM

- **Rapsberry Pi 3 con modulo cámara V2:**
 - 8 megapíxeles
 - Sensor de profundidad
 - 900 MHz
 - 1 GB – RAM

Se encontró en diferente documentación que el correcto funcionamiento del análisis de imágenes, proceso que se utilizará para detectar a los pasajeros, es alrededor de los 12 megapíxeles con una capacidad de procesamiento de 1 GHz y 2 GB de RAM [55], basados en esto se calificaron las tecnologías, encontrando que la única tecnología que cuenta con estas características es la de *smartphone*.

Es importante resaltar, que el módulo de cámara propio de Raspberry tiene una resolución menor a la sugerida para un correcto funcionamiento, referente al análisis de imágenes, además su capacidad de procesamiento está limitada por las características de la placa Pi 3 con lo cual esta tecnología tiene una baja escalabilidad.

Por otro lado, la tecnología Kinect tiene una buena resolución, pero su procesamiento es limitado por las características de la placa Raspberry, de forma que los algoritmos podrán mejorarse hasta el punto que esta lo permita.

Los *smartphones* por otro lado son dispositivos que se actualizan cada día, y debido a su continuo y rápido avance se lanzan cada constantemente al mercado dispositivos con mejores prestaciones a menores precios. El dispositivo móvil seleccionado cuenta con las mejores prestaciones de las 3 tecnologías, permitiendo que a futuro se refinan sus algoritmos sin tener mayores implicaciones, y al desarrollar una aplicación móvil nativa, esta en un futuro podrá ser instalada en dispositivos más avanzados, sin tener que cambiar los algoritmos diseñados, por todo esto se encuentra que los smartphones son la tecnología con mejores prestaciones frente a la escalabilidad.

- **Implementación:** Dentro de este criterio se evaluará la dificultad de implementar cada tecnología para el desarrollo del proyecto, ya que cada una requiere un diseño e

implementación hardware y software diferente. La calificación se asignará como alta cuando la tecnología sea más simple de implementar.

En primer lugar, se encontró que implementar la tecnología Kinect resulta en una ardua tarea, ya que para diseñar algoritmos que accedan al uso de sus recursos no es una tarea sencilla debido a que la empresa Microsoft no brinda una documentación abierta y amplia para el uso de esta herramienta. Por lo cual implementar el proyecto con esta tecnología resulta en un reto mayor con respecto a la elaboración de los algoritmos de detección, razón por la cual se le calificó como baja en este criterio.

El módulo de cámara de Rapsberry al ser desarrollado por la misma empresa que la placa en la que se implementara tiene una conectividad muy sencilla, además existe amplia documentación de proyectos realizados con esta tecnología, y la empresa desarrolladora de esta tecnología brinda algoritmos para la detección y análisis de imágenes que facilitan en gran medida el desarrollo de algoritmos, por lo cual esta tecnología resulta con la mayor calificación frente al criterio en cuestión.

Por último, desarrollar una aplicación móvil nativa para un teléfono inteligente resulta en una tarea con dificultad media, ya que tiene una amplia documentación, pero el desarrollo de este tipo de aplicaciones implica la integración con diferentes librerías al proyecto que permitan hacer uso de los recursos del dispositivo.

- **Fiabilidad:** entendiéndola como la probabilidad de que un sistema, aparato o dispositivo cumpla una determinada función bajo ciertas condiciones durante un tiempo determinado [56]. Este criterio se evaluará en cada tecnología con la tasa de éxito con respecto al correcto conteo de pasajeros.

Ya que las 3 tecnologías, utilizarán el análisis de imágenes para la detección, y en base a los proyectos similares analizados con anterioridad, en los cuales se utilizan estas tecnologías, se encontró que su fiabilidad es muy similar, con valores alrededor del 80% y en base a la Tabla 15 se encuentra que su fiabilidad es media.

Tabla 15 – criterios de calificación de tecnologías con respecto a la tasa de éxito

Calificación	Tasa de éxito (TE)
Alta	TE > 90%
Media	75 < TE < 90
Baja	TE < 75%

- **Recolección de datos:** Este criterio hace referencia a la capacidad que tendrá cada tecnología por sí sola, sin hacer uso de dispositivos externos, para recolectar diferentes datos de interés.

Las tecnologías de Kinect y módulo de cámara V2 de Rapsberry, por sí solas son solo sensores de captura, no cuentan con ningún nivel de procesamiento, por lo cual la

captura de datos de estas tecnologías será únicamente el correspondiente a la detección de personas y como se dijo anteriormente es de vital importancia capturar información de conductores, rutas y buses asignados a estos, por lo cual estas tecnologías tienen una calificación baja en este criterio

Por otro lado, el *smartphone* permite una interacción directa y sencilla con los conductores y por medio de una validación de sesión se pueden capturar los datos antes mencionados, brindando una gran ventaja a esta tecnología frente este criterio.

3.2.1.2. Tecnología Seleccionada

Dados los criterios anteriormente mencionados, se elabora la Tabla 16, con el fin de obtener un puntaje total y seleccionar la tecnología adecuada para el desarrollo del proyecto. Con base en dicha tabla, se concluye que la tecnología a utilizar en el presente trabajo son los smartphones, puesto que presentan el mejor promedio de desempeño.

Tabla 16 - Puntajes criterios de selección por tecnología.

CRITERIOS	TECNOLOGÍAS		
	Rapsberry		Smartphones
	Modulo Cámara V2	Kinect	
Costos	Medio	Medio	Medio
Escalabilidad	Bajo	Medio	Alto
Implementación	Alto	Bajo	Medio
Fiabilidad	Medio	Medio	Medio
Recolección de datos	Medio	Medio	Alto
Puntaje Total	10	9	12

Alto = 3 puntos; Medio = 2 puntos; Bajo = 1 punto.

3.3. Patrón de diseño

El patrón de desarrollo usado para la realización de la capa de acceso a datos es el MVC mostrado en la Ilustración 30. Su selección se basa en la premisa de que a partir de él se han hecho adaptaciones como el modelo MVP o MVVM, los cuales se desarrollan a partir de su estructura, pero la separación entre los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos, modelo, vista y controlador [57] son suficientes para las necesidades del desarrollo. Además, frameworks de desarrollo como Angular, .Net, Django, lo incorporan.

El modelo, contiene la representación de los datos que maneja el sistema, su lógica de negocio y métodos de persistencia. La interfaz de usuario o vista es la encargada de presentar

la información al usuario, cada vista define un procedimiento de modificación que se activa por los cambios del mecanismo de propagación. Cuando el procedimiento de modificación es llamado, la vista devuelve todos los actuales para desplegar desde el modelo y ubicarlos en la pantalla. Finalmente, el controlador actúa como intermediario entre el modelo y la vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

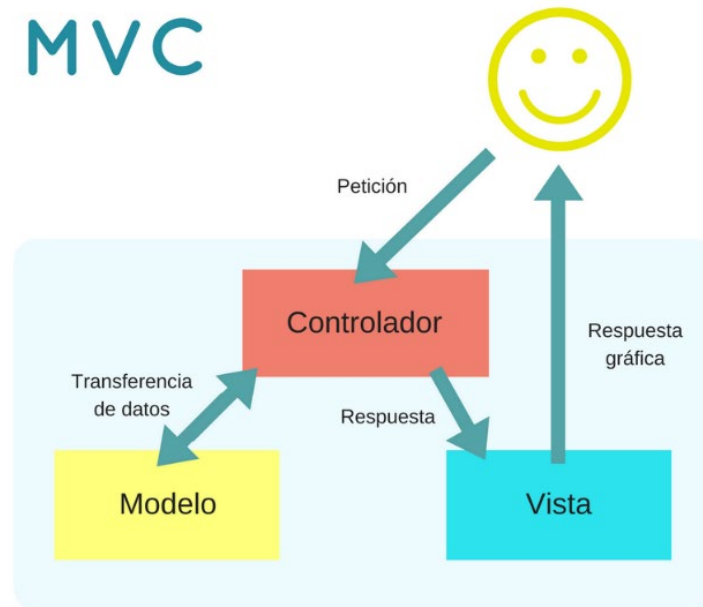


Ilustración 30 - Modelo vista controlador

3.4. Sistema de control de versiones

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que se pueden recuperar las versiones más específicas en algún punto del proceso de desarrollo [58].

Para el presente trabajo de grado, se realiza un control de versiones local y remoto, de modo que los dos integrantes del grupo de desarrollo pueden implementar las tareas respectivas manteniendo un orden en el desarrollo y generando un respaldo que garantiza puntos de recuperación del código en caso de cualquier incidencia. El sistema de control de versiones remoto que se usa es GitHub [59], los principales comandos del proceso de control de versiones se encuentran en el Anexo B.

3.5. Segmentación por prototipos

El sistema prototipo es segmentado en base a la realización de prototipos, tal y como se observa en la Ilustración 31.

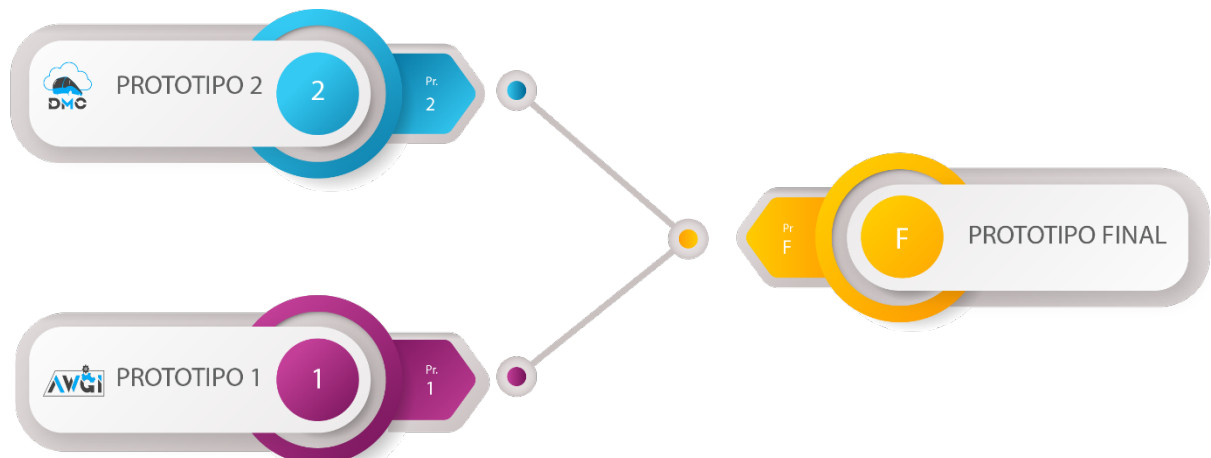


Ilustración 31 - Segmentación por prototipos del sistema

De donde el prototipo 1 corresponde a la implementación de la aplicación web, con las funciones de visualización y gestión. Además, se implementa la primera versión de la aplicación móvil, la cual contará con las funcionalidades de autenticación, visualización de rutas asignadas y la posibilidad de iniciar una actividad de conteo simulado.

Por otro lado, el prototipo 2, se basa en la implementación de una aplicación móvil centrada únicamente en el conteo del flujo de pasajeros por medio de análisis de imágenes. Esta implementación reemplazará la actividad del conteo simulado del prototipo 1, generando el prototipo final.

3.6. Prototipo 1

El prototipo 1 le proporciona al sistema un mapeo de funcionamiento a partir del cual se pueden construir los diseños, navegaciones entre pantallas y lógica de administración, con eventos simulados. La finalidad de realizar por separado este prototipo con respecto a la lógica de conteo de personas, es reducir la complejidad de desarrollo evitando incidencias o errores de la librería que se escoja para realizar dicho conteo en el DMC, por lo tanto, los eventos simulados descritos con anterioridad corresponden a los eventos de ascenso y descenso del bus.

3.6.1. Modelamiento

Para tener claridad del flujo principal que siguen los módulos AWGI y DMC con respecto a la captura de datos de abordaje de pasajeros, se elabora el esquema de la Ilustración 32.

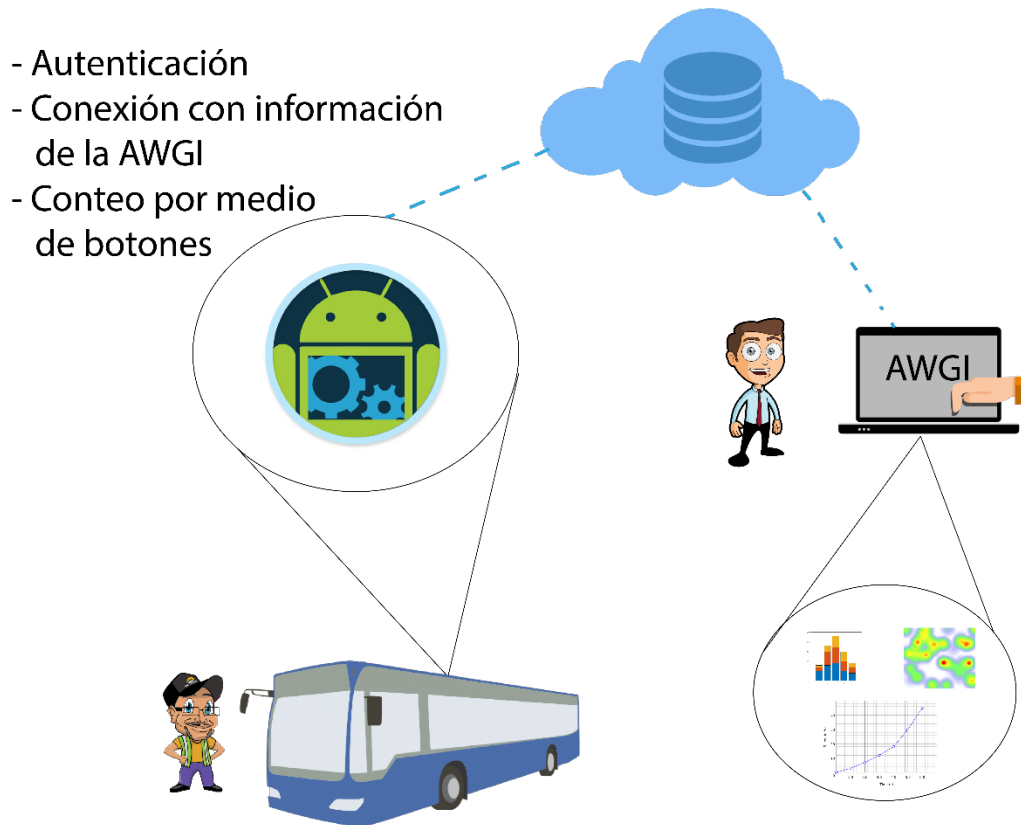


Ilustración 32 - Evento principal de los módulos con respecto al conteo de pasajeros

Gracias a este esquema se entiende de una forma más clara el modelo base para la implementación del prototipo 1, donde encontramos que el DMC se basará en la implementación de una aplicación móvil en Android, la cual tendrá tres pantallas básicas, la primera permitirá realizar la autenticación del conductor, la segunda vista le indica al conductor la fecha y ruta que tiene asignada para el día en específico y finalmente la última vista corresponde a una simulación de flujo de abordajes provisional, la cual se muestra en la Ilustración 33. Dicho conteo de abordajes y descenso simulado estará acompañado por la determinación de la latitud y longitud del evento, datos que también serán enviados a la base de datos.

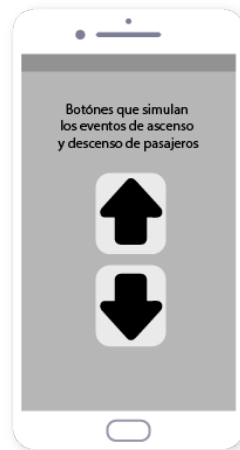


Ilustración 33 - Botones simulados de ascenso y descenso de pasajeros

Una vez que se procesa el conteo por la aplicación móvil, los eventos son guardados en una base de datos, la cual es accesible por la aplicación web, de tal forma que los usuarios que accedan al AWGI, pueden visualizar la información.

A continuación, se presenta el proceso de diseño de la interfaz, y el planteamiento de las vistas que serán usadas en los módulos AWGI y DMC. Es necesario tener claridad en la navegación y contenido de cada una de las pantallas para facilitar el proceso de implementación. Una vez las interfaces son definidas en ambos módulos, se procede con la implementación, en la cual se definen las tecnologías de desarrollo y la forma técnica en la que se comunican los módulos con el sistema.

3.6.1.1. Diseño de interfaz AWGI

La AWGI va a ser utilizada por empleados delegados del SETP. En ella se pueden hacer gestiones diarias de información referente a buses, rutas y conductores, con un flujo de asignación basado en formularios y tablas de visualización.

Es importante tener contacto con empresas o personas que potencialmente harían uso de una herramienta que se está desarrollando. Para este caso en particular, se tuvo el apoyo de una empresa de transporte público de la ciudad de Buga, llamada Soluciones Empresariales Posso y Asociados S.A.S. Dado que el prototipo es diseñado para la ciudad de Popayán, se analizaron aspectos claves que permitirían tomar como viable la retroalimentación otorgada por la empresa mencionada, en la cual se encontró que la ciudad de Buga, para fines de movilidad y transporte público, tiene aspectos limitantes igual que en Popayán, aspectos entre los cuales se encuentra que el sistema actual no es de tipo BRT y se producen problemáticas similares, una de ellas es el desconocimiento del conteo de pasajeros y la falta de gestión de información interna de las empresas. Entonces, dada la congruencia, se acepta la ayuda y colaboración por parte de la empresa.

Para tener un diseño en la interfaz web que se ajuste más a las necesidades finales de quién va a hacer uso de la aplicación, se realizó una propuesta en papel al más alto nivel que fue

discutida en una reunión presencial con la empresa de Buga. En la Ilustración 34, se muestra el diseño mostrado en dicha reunión.

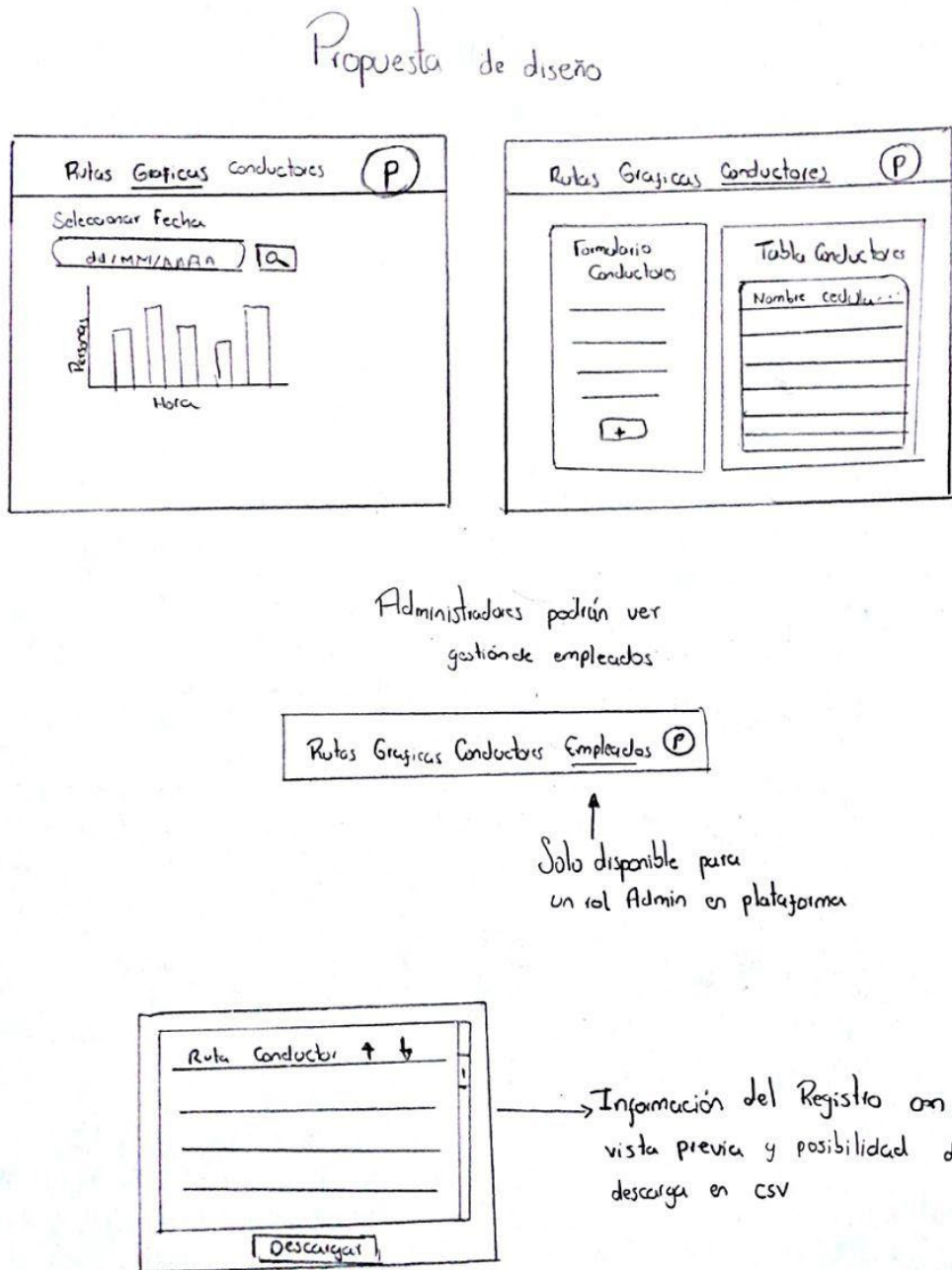


Ilustración 34 - Propuesta en papel en alto nivel

La intención del diseño presentado, fue mostrar las funciones básicas que tendría la aplicación web y la forma en que los empleados interactuarían con ella. El sistema de navegación conocido como *navbar*, la gráfica de visualización de información, tablas de descarga y formularios de administración de información (conductores, empleados y buses) fueron los

principales puntos que se abordaron. A lo largo de la reunión, hubo retroalimentación y por ende sugerencias que fueron adoptadas en el diseño final que se mostrará junto con la descripción detallada de cada pantalla.

La primera interfaz que se define es la de inicio de sesión, en la cual es necesario tener credenciales válidas de acceso de correo electrónico y contraseña para ingresar a la aplicación. Dado que no es una aplicación pública, se requieren credenciales válidas para la visualización de la información. El diseño final del inicio de sesión se observa en la Ilustración 35.

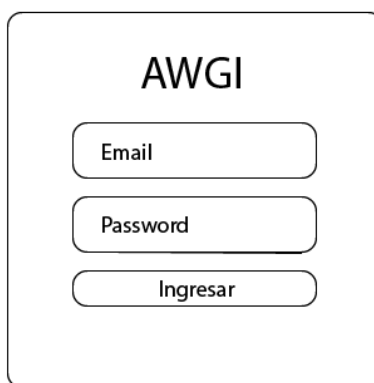


Ilustración 35 - Inicio de sesión

Cabe resaltar que en la vista de inicio de sesión no se cuenta con opciones de reinicio de contraseña o registro, dado que es meramente administrativo, entonces los casos particulares deben ser tratados directamente con el administrador root o administradores de la plataforma. El registro de los usuarios que pueden acceder a la plataforma viene ligado al administrador root que tiene las credenciales absolutas en el sistema, a partir de él se jerarquizan los demás roles.

Una vez el usuario ingresa a la plataforma se encuentra con un esquema de navegación que le da acceso a cada una de las características particulares de su rol, esto hace referencia a que la información presentada en dicho esquema, es dinámico y depende exclusivamente del rol de quien inicia la sesión. La barra de navegación que contiene todos los contenidos disponibles y aparece para el usuario administrador root, se presenta en la Ilustración 36.



Ilustración 36 - Barra de navegación

Se muestra a continuación, en la Ilustración 37, la pantalla que corresponde al *home* o primera pantalla luego de ingresar a la plataforma.

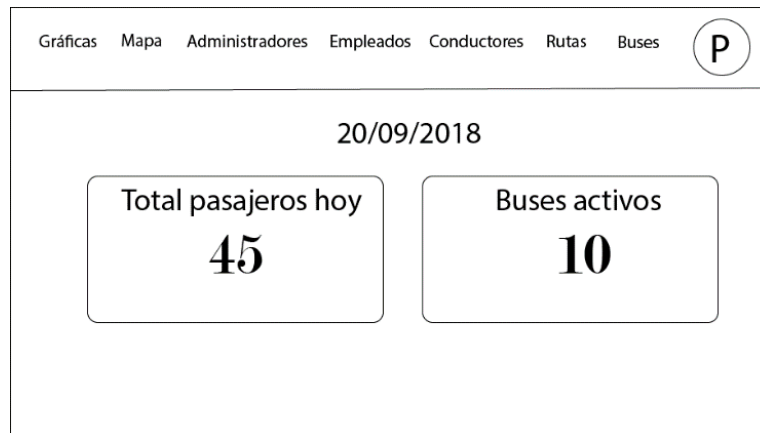


Ilustración 37 - Pagina home

La información mostrada en la página de *home* es dinámica y muestra la cantidad de pasajeros que han abordado los buses registrados en el sistema, así como la cantidad de vehículos activos.

Las dos pantallas que más incidencia tienen para el propósito del presente trabajo de grado son la de gráficas presentadas en la Ilustración 38 y mapa en la Ilustración 39, las cuales son accesibles para todos los usuarios, incluidos los de menor jerarquía en el sistema.

En la pantalla de gráficas, se puede visualizar la información proporcionada por el DMC a lo largo de un día o en un rango de fechas. El primer filtro de búsqueda es seleccionar la ruta, posteriormente se habilita un segundo campo del tipo *selector*⁴ en el cual se recuperan los buses específicos que pertenecen a la ruta, finalmente se encuentran disponibles dos campos de tipo *date*. Dado que la búsqueda puede hacerse en un solo día, basta con indicar solo la fecha de inicio para que el sistema interprete la consulta, de lo contrario, al indicarse también la fecha final, será filtrada la información por rango de días. Cuando la búsqueda es para un solo día, la gráfica será número de pasajeros vs hora del día (cuenta total y parcial en dos barras contiguas), en cambio, cuando la búsqueda se realiza por rango de fechas, el resultado será número de pasajeros vs día.

⁴ Tipo de entrada de datos que despliega una lista de opciones, generalmente visto en formularios.

Gráficas Mapa Administradores Empleados Conductores Rutas Buses P

Ruta: Bus: Fecha Inicio: Fecha Final:

■ Pasajeros Totales ■ Pasajeros en Vehículo

Hora del día	Pasajeros Totales	Pasajeros en Vehículo
1	~15	~8
2	~20	~15
3	~25	~10

Conteo de pasajeros

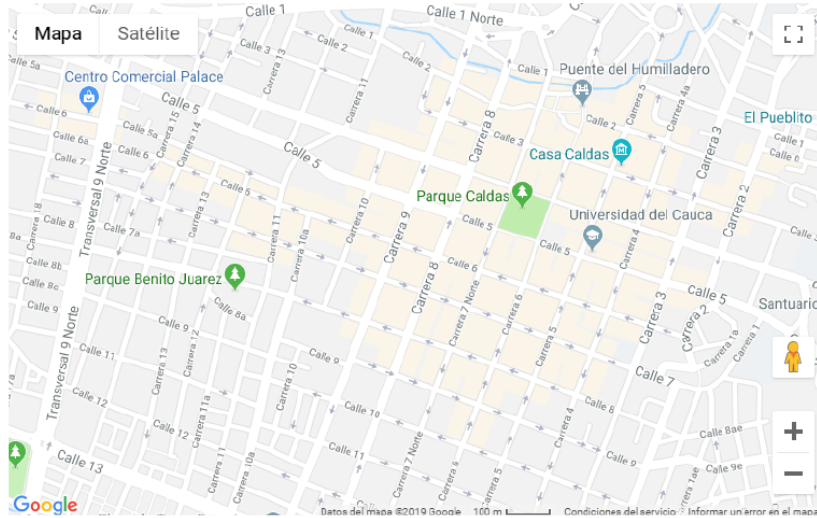
Fecha	Hora	Abordaje	Conteo Parcial	Conteo Total	Latitud	Longitud

Descarga

Ilustración 38 - Página de gráficas

Gráficas Mapa Administradores Empleados Conductores Rutas Buses P

Ruta: Bus: Fecha:



Gráfica de eventos

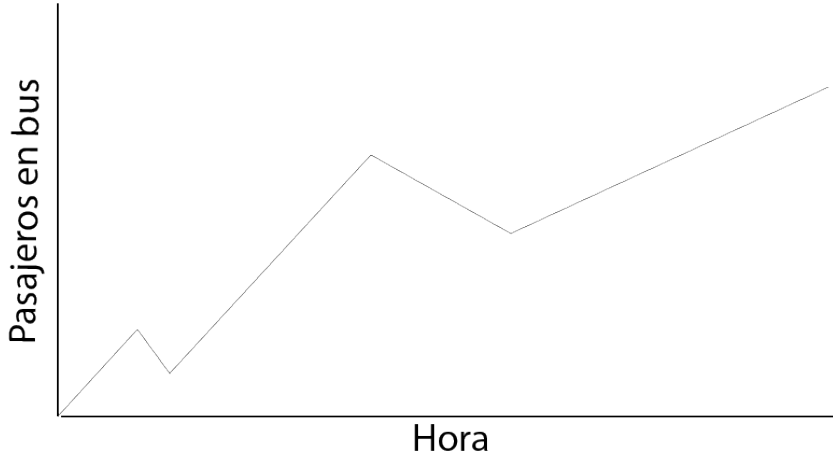


Ilustración 39 - Página de mapa

La página de mapas está pensada para un seguimiento diario más específico. La vista se divide en dos secciones, una de ellas corresponde a un mapa de calor, el cual va a mostrar las zonas con más intensidad en eventos, de forma tal que los colores oscuros generados van a corresponder a las zonas en las que más eventos de subida o bajada de pasajeros se registran. La segunda sección corresponde a una gráfica que muestra la variación de pasajeros con respecto a los eventos de subida y bajada. Dada la naturaleza del seguimiento, la búsqueda se hace de igual manera que en la página de graficas, con la diferencia que no se permiten hacer consultas por rango de fecha.

La siguiente sección de ilustraciones, tiene un esquema muy parecido en su composición, dado que según las recomendaciones de la empresa Posso & Asociados S.A.S, se hace mejor para ellos tratar la información de administración por bloques con la inclusión de un formulario rápido junto con una tabla de visualización.

La pagina de administradores en la Ilustración 40 permite crear usuarios con dicho rol. Los campos definidos para el formulario, son definidos en la Tabla 17:

Tabla 17 - Campos de entrada para formulario de creación de usuarios administradores

NOMBRE DEL CAMPO	TIPO DE ENTRADA
Cédula de ciudadanía	<i>Integer</i>
Nombres	<i>String</i>
Correo Electrónico	<i>String</i>
Contraseña	<i>String</i>
Fecha de nacimiento	<i>String</i>
Dirección	<i>String</i>
Número de celular	<i>Integer</i>

La vista de la tabla de información permite cambiar el estado de un administrador en particular, siendo una variable booleana la que determina si está activo o inactivo por medio de un botón tipo *switch*. Así mismo, dos botones permiten en orden respectivo editar y eliminar un usuario.

Dado que no toda la información del usuario tiene el mismo peso, se le da prioridad en visualización al nombre y la cedula, permitiendo acceder a la información completa mediante un click en el botón de editar usuario. Los datos del administrador en particular serán desplegados sobre las mismas entradas del formulario de creación, cambiando de estado el botón de crear administrador por editar administrador.

La página de empleados tiene la misma composición que la de administradores, la diferencia radica en que solamente el usuario administrador root tiene acceso a la primera. Entonces, desde la pagina de empleados, un usuario administrador está habilitado para crear empleados.

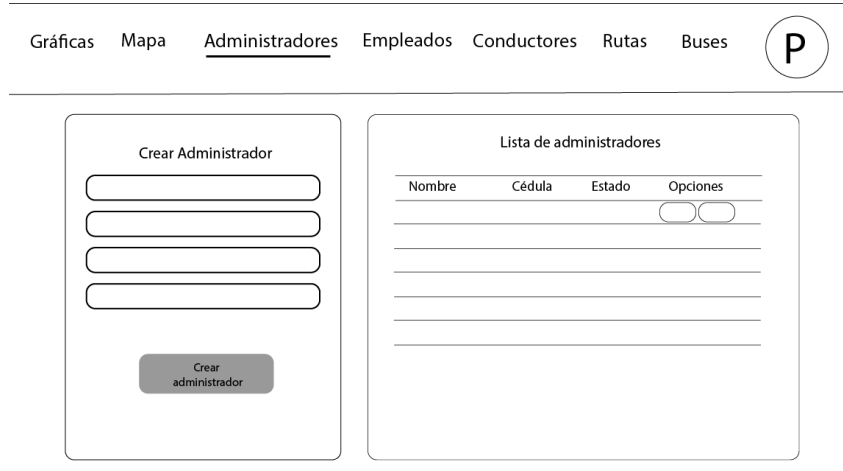


Ilustración 40 - Página de administradores

Las entradas definidas para la página de gestión de conductores son las mismos usadas con anterioridad, con la salvedad de que se agregan dos nuevos campos, los cuales se muestran en la

Tabla 18

Tabla 18 - Campos requeridos para creación de conductores

NOMBRE DEL CAMPO	TIPO DE ENTRADA
Ruta del conductor	String
Bus del conductor	String

La información definida para los campos de creación de rutas y buses están expuestas en la Tabla 19 y la Tabla 20. Cabe resaltar que la información de creación de rutas pretende definir un contexto de la misma, permitiendo definir una descripción de la misma y un recorrido con los puntos clave por donde el vehículo pasa.

Tabla 19 - Campos requeridos para la creación de rutas

NOMBRE DEL CAMPO	TIPO DE ENTRADA
Nombre de la ruta	String
Descripción de la ruta	String
Recorrido de la ruta	String

Tabla 20 - Campos requeridos para la creación de buses

NOMBRE DEL CAMPO	TIPO DE ENTRADA
Placa	String
Modelo	String
Marca	String
Capacidad	Integer
Ruta	String

Las páginas de gestión de empleados, conductores, rutas y buses se observan en la Ilustración 41.



Ilustración 41 - Páginas de gestión de empleados, conductores rutas y buses.

Finalmente, se diseña una interfaz de edición de perfil, la cual está disponible para todos los usuarios que tienen credenciales de acceso a la AWGI. El usuario va a ser capaz de editar su información principal (incluida imagen de perfil) y contraseña. La Ilustración 42, muestra el diseño final de la edición de perfil.

Gráficas Mapa Administradores Empleados Conductores Rutas Buses **P**

Actualizar perfil

P Cambiar

Actualizar Cambiar contraseña

Ilustración 42 - Editar perfil

3.6.1.2. Diseño de interfaz DMC

El dispositivo móvil de conteo está basado en tres pantallas. La primera es la de inicio de sesión (Ilustración 43), en la cual el conductor debe validar su acceso. Inmediatamente es validado, se muestra la información básica del día en la segunda pantalla (Ilustración 44), en la cual se visualiza la fecha, ruta y nombre del conductor. Finalmente, la tercera pantalla corresponde a la vista de la cámara del teléfono en la cual se implementa la lógica de conteo. Esta pantalla no tiene diseño de interfaz dado que en ella se observa la captura de video de la cámara y no tiene piezas gráficas.



Ilustración 43 - Inicio de sesión DMC



Ilustración 44 - Información básica pre conteo DM

3.6.1.3. Selección de base de datos

El primer punto a decidir sobre qué base de datos usar, fue seleccionar cuál tipo se usaría, entendiendo los tipos como bases de datos relacionales y bases de datos no relacionales.

El principio de las bases de datos relacionales se basa en la organización de la información en trozos pequeños, que se relacionen entre ellos mediante la relación de identificadores, mientras que las bases de datos no relacionales no tienen un identificador que sirva de relación entre un conjunto de datos y otro.

Con ambos tipos de base de datos se puede construir un sistema con casi cualquier especificación, sin embargo, gracias a los siguientes aspectos, para el presente trabajo se seleccionó el uso de bases de datos no relacionales:

- Son especializadas en aplicaciones del tipo *big data*, es decir que se almacenan datos masivos.
- Son sistemas económicos y menos complejos que las bases de datos de tipo relacional.
- Utilizan modelos de datos más flexibles que los ofrecidos por el modelo relacional con las operaciones SQL.

Una vez seleccionado el tipo de base de datos, se plantearon dos posibilidades de uso gracias al entendimiento de los requerimientos y a experiencias de trabajo anteriores.

Por un lado, se estudió MongoDB, la cual es una base de datos orientada a documentos y de código abierto, generalmente usada en el *full stack*⁵ de Node.js y por otro lado está Firebase, una herramienta de google que provee de una base de datos en tiempo real.

⁵ Aspectos front-end y back-end basados en Node.js

Ambas bases de datos se ajustan a las necesidades del proyecto y tienen una implementación similar, pero hubo aspectos que sumaron peso a Firebase, razón por la cual fue la base de datos escogida:

- **Seguridad:** Firebase es una herramienta que provee de reglas de seguridad bien definidas y controladas para el acceso a los datos.
- **Back-end en la nube:** Al ser una herramienta de google, Firebase provee de un back-end en la nube sin la necesidad de ser implementarla en un servidor externo, de esta manera se tiene acceso a datos en tiempo real en todo momento.
- **Sin conexión:** Si se genera una pérdida de conexión, los datos se mantienen temporalmente en memoria caché, una vez se reestablezca la comunicación, los datos se sincronizan automáticamente.
- **Una sola base de datos:** Firebase brinda la posibilidad de tener una sola base de datos por aplicación para todas las plataformas, las cuales incluyen aplicaciones android, iOS y de web con SDKs nativos.
- **Funcionalidades adicionales:** La base de datos en tiempo real es una funcionalidad añadida de todas las herramientas que provee Firebase, de modo que muchas utilidades agilizan el desarrollo del proyecto.

Algunas de las funcionalidades se observan en la Ilustración 45.

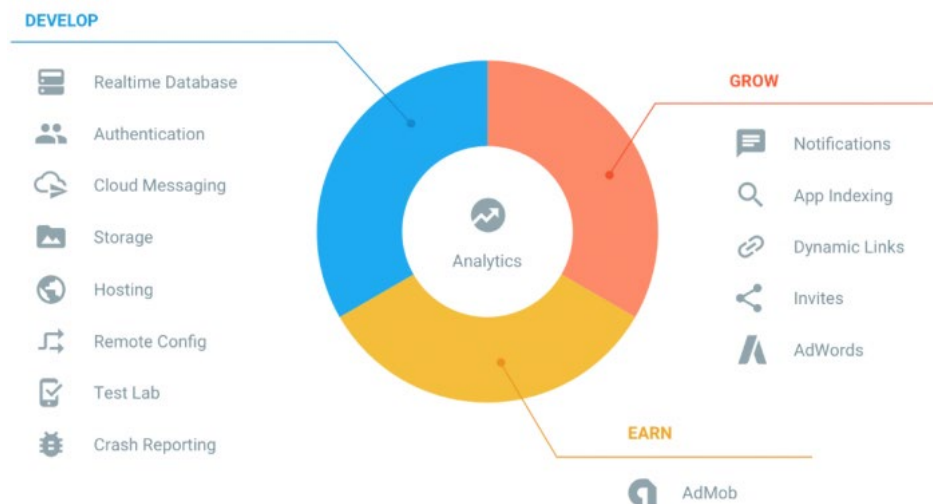


Ilustración 45 - Firebase funcionalidades

Tomado de: [60]

Las herramientas usadas para el prototipo 1 son *Realtime Database*, *Authentication* y *Storage*. La base de datos en tiempo real de Firebase permite almacenar todos los datos que la aplicación requiera. Tiene una compatibilidad importante con React⁶ y su patrón reactivo que

⁶ Es una librería de JavaScript focalizada en el desarrollo de interfaces de usuario.

permite actualizar los datos en los componentes automáticamente. Los datos se almacenan en formato JSON y se pueden agregar reglas para permitir *requests* con token o solo desde una URL por ejemplo [61]. La herramienta de autenticación simplifica las gestiones de sesiones y existe adaptabilidad para diferentes proveedores como Google, Facebook, Twitter o Github. Finalmente, el servicio de almacenamiento permite guardar archivos para la lógica dinámica de la aplicación o subir documentos estáticos para el uso de la misma.

Dado que existen diferentes planes de facturación para el uso de Firebase, en la Tabla 21 se muestran las limitaciones a nivel de prestaciones que Google ofrece con el *Plan Spark* (Gratis).

Tabla 21 - Características plan gratis de uso de Firebase

FUNCIONALIDAD	PLAN GRATIS
A/B Testing, Analytics, App Indexing, Authentication (except Phone Auth), Cloud Messaging (FCM), Crashlytics, Dynamic Links, Invites, Performance Monitoring, Predictions, and Remote Config.	Incluidas
Conexiones simultáneas	100
GB almacenados	1 GB
GB descargados	10 GB/mes
GB almacenados	5 GB
GB descargados	1 GB/día
Operaciones de carga	20,000/día
Operaciones de descarga	50,000/día

 Base de datos en tiempo real

 Almacenamiento

3.6.2. Implementación AWGI

En esta sección se muestra el proceso de implementación de la aplicación web, empezando por la selección de la tecnología y las configuraciones básicas con la plataforma Firebase, para finalmente hacer las pruebas respectivas del prototipo 1.

3.6.2.1. Tecnología de desarrollo

La tecnología seleccionada para el desarrollo de la AWGI es Angular, un *framework* JavaScript, gratuito y de código abierto, creado para facilitar la creación de aplicaciones web modernas de tipo SPA⁷.

Angular comenzó con la primera versión de AngularJS que estaba escrito en JavaScript puro. Así mismo, ofrecía un concepto muy novedoso como lo era el *two-way data binding*, que era la forma de interactuar entre la vista y el modelo, aquello que se conoce como *view model*.

⁷ Tipo de aplicación web donde todas las pantallas las muestra en la misma página, sin recargar el navegador

Gracias a dicho concepto, se podía interactuar con las propiedades del modelo y se actualizaban tanto desde la vista como desde el modelo.

Ese concepto se ha mantenido hasta la actualidad en las últimas versiones de Angular, su arquitectura evolucionó y se reescribió como se observa en la Ilustración 46. Cada módulo de Angular se forma por componentes totalmente independientes unos de otros.

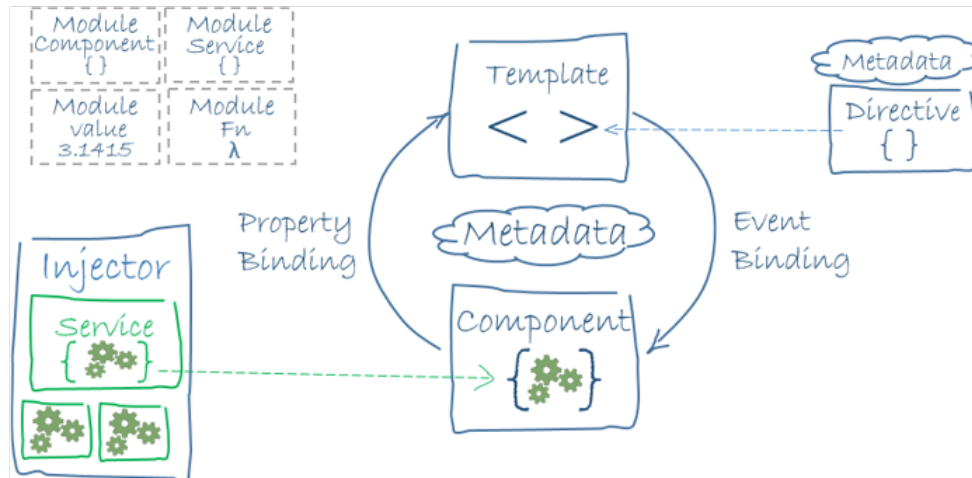


Ilustración 46 - Arquitectura de Angular
Tomado de: [62]

En las últimas versiones se incluye Angular CLI, un intérprete de la línea de comandos que permite crear módulos, clases y servicios de forma más rápida.

Las principales ventajas que decantaron su uso se describen a continuación:

- Es un framework respaldado por Google y tiene una creciente comunidad.
- Facilita labores de *testing*.
- TypeScript, el lenguaje principal del framework, proporciona detección temprana de errores y tipado fuerte de clases, métodos, objetos y APIs JavaScript ya existentes.
- Es un framework muy estable.
- La documentación de soporte con Firebase es muy amplia.

3.6.2.2. Conexión de Firebase con AWGI

Cuando se inicia un nuevo proyecto con Firebase, este genera un token único que permite la conexión, en este caso de una aplicación web. En la Ilustración 47, se observa la fracción de

código, invocada desde el módulo de aplicación en los *imports*, la cual inicializa la conexión con Firebase.

```
export const environment = {
  production: false,
  firebaseConfig: {
    apiKey: 'AIzaSyAUoKdGWAspCtdJBd0Qt830FyBn3p9Q4h8',
    authDomain: 'hegi-f634b.firebaseio.com',
    databaseURL: 'https://hegi-f634b.firebaseio.com',
    projectId: 'hegi-f634b',
    storageBucket: 'hegi-f634b.appspot.com',
    messagingSenderId: '794033922816'
  }
};
```

Ilustración 47 - Configuración Firebase AWGI

3.6.2.3. Creación de componentes y estructura

Es necesario generar una estructura de proyecto bien definida y organizada según el modelo de desarrollo MVC, con el fin de acceder a todos los recursos y vistas de una manera coherente y eficiente. El proceso de creación de los componentes se encuentra detallado en el Anexo C.

3.6.2.4. Routing

La creación de un módulo de *routing* permite gestionar la renderización de *templates* y la implementación de su lógica a partir de una estructura de *paths*⁸ bien definida. La implementación completa de dicho módulo está definida en el Anexo D.

3.6.2.5. Persistencia

Para facilitar la interacción del usuario con el sistema, se implementa un control de sesión, por medio del cual, si una persona ya ingresó a la plataforma en un navegador determinado y ésta cierra la pestaña, gracias a la implementación de la persistencia, cuando ingrese de nuevo a la plataforma, no tendrá necesidad de pasar por la vista de *login*. La implementación de la persistencia se encuentra detallada en el Anexo E.

3.6.2.6. Servicios

A medida que los objetos son más y más complejos en los componentes, es normal que el código vaya aumentando, por lo tanto, es importante tener presente que la organización del

⁸ Dirección que resuelve el navegador, se visualiza en la URL.

código en piezas pequeñas de responsabilidad reducida es esencial. Además, cuando el proyecto se hace más grande, es normal que muchos componentes necesiten acceder a los mismos datos y hacer operaciones similares con ellos. Por lo anterior, nace el concepto de servicio, que básicamente es un proveedor de datos, el cual mantiene lógica de acceso y operativa relacionada con el negocio y las operaciones que se hacen con los datos dentro de una aplicación. Los servicios son consumidos por los componentes, los cuales delegan en ellos la responsabilidad de acceder a la información y la realización de operaciones con los datos [63].

Con el fin de mantener la organización en el proyecto, los servicios creados llevan el nombre del componente que lo va a consumir. Cada uno de estos servicios debe ser declarado en el módulo de Angular [64], en la sección *providers*. La distinción principal de un servicio es la etiqueta *@Injectable* que se observa en la Ilustración 48.

```
@Injectable({  
  providedIn: 'root'  
})
```

Ilustración 48 - Etiqueta Injectable de un servicio

3.6.2.7. Crud

Crud es el acrónimo de “Crear, Leer, Actualizar y Borrar”, en esta sección se describen estas acciones para las vistas que tienen un patrón de diseño basado en formulario y tabla (administradores, empleados, conductores, rutas y buses).

La estructura de cada una de estas vistas está basada en un componente padre y dos componentes hijos, como se observa en la Ilustración 49, donde el componente padre es aquel que se llama desde el *routing* de la aplicación, pero este se comporta de forma similar a lo observado por el *template* del *Application Module*, permitiendo inyectar los selectores de los dos componentes hijos, formulario y tabla de visualización.

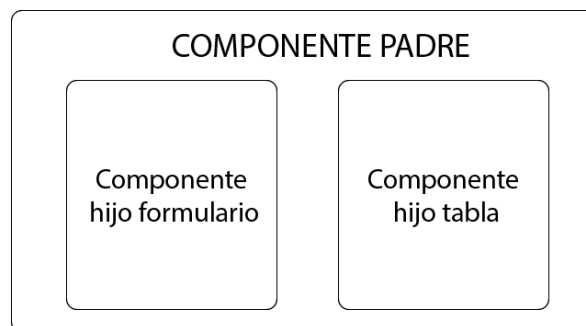


Ilustración 49 - Estructura de componentes para páginas CRUD

Lo anteriormente descrito, se observa programáticamente en la Ilustración 50, en donde los dos sub selectores *app-form-admin* y *app-list-admin* son inyectados en el *template* del componente padre.

```
<div class="row">
  <div class="col-md-5 mx-auto mt-3 mb-4">
    <div class="card">
      <app-form-admin></app-form-admin>
    </div>
  </div>
  <div class="col-md-6 mx-auto mt-3 mb-4">
    <div class="card">
      <app-list-admin></app-list-admin>
    </div>
  </div>
</div>
```

Ilustración 50 - Inyectando componentes hijos

Hecho lo anterior, la lógica de cada uno de los hijos es manejada desde sus propios componentes y servicios.

3.6.2.7.1. Componente formulario

El archivo html del componente hijo formulario, de aquí en adelante *template*, se basa en campos de entrada y botón de acción. Cada uno de los campos está asociado en forma de puente al componente por medio del NgModel⁹, permitiendo la comunicación “*template* ⇔ *component*” como se muestra en la Ilustración 51, donde se observa un *input* que tiene el parámetro “[ngModel]”, el cual indica que es en doble vía es decir, cualquier cambio en el componente se observará en el *template* y viceversa. Y así la variable de clase *UserProfile.address* puede ser procesada dentro del componente.

```
<div class="form-group">
  <input type="text" class="form-control text-center"
    name="address" #address="ngModel" [(ngModel)]="UserProfile.address"
    placeholder="Dirección">
</div>
```

Ilustración 51 - NgModel en un input

Ahora bien, el archivo TypeScript del componente hijo formulario, en adelante, componente, tiene como su principal función, el registrar o actualizar la información ingresada a los campos del *template*. Para mostrar el flujo principal del componente, se describe un fragmento del código empleado para la página de rutas.

⁹ ngModel viene de lo que se conoce como el modelo en el MVC. El modelo trabaja con los datos, así que el enlace creado con ngModel permite que desde la vista pueda usar un dato.

```

createRoute(routeForm?: NgForm) {
  this.routeService.insertRoute(routeForm.value)
    .then(res => {
      console.log(res);
      this.toastr.success('Operacion Exitosa', 'Ruta Creado');
    }, err => {
      console.log('error', err);
      this.toastr.warning('Operacion fallida', 'No se creó la ruta');
    });
}

```

Ilustración 52 - Función registrar ruta

La función *createRoute* de la Ilustración 52 tiene como parámetros de entrada, los campos del formulario, los cuales son enviados a la función *insertRoute*, Ilustración 53, del *routeService* (instancia creada para acceder al servicio).

```

insertRoute(route: Routes) {
  return this.routesList.push({
    name: route.name,
    travel: route.travel,
    description: route.description,
  });
}

```

Ilustración 53 - Función insertar ruta en servicio

En la función *insertRoute* se hace el push de datos a la base de datos de firebase por medio de una variable del tipo *AngularFireList*¹⁰, la cual brinda acceso a la lista con inicio en la rama 'routes'. La definición y asignación de dicha variable se puede observar en la Ilustración 54 y en la Ilustración 55.

```

routesList: AngularFireList<any>;

```

Ilustración 54 - Definición de variable tipo AngularFireList

```

this.routesList = this.afDatabase.list('routes');

```

Ilustración 55 - Asignación de la variable routesList

Una vez el *push* se ha realizado con éxito, se visualizan los datos en la base de datos, tal y como se observa en la Ilustración 56.

¹⁰ Propiedad que permite crear una instancia del tipo lista para la base de datos de Firebase

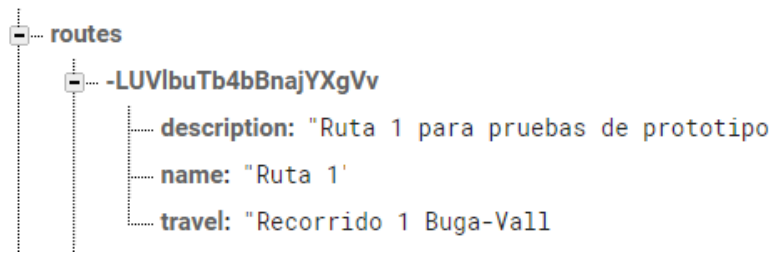


Ilustración 56 - Ruta creada en Firebase

El procedimiento para realizar una actualización de datos tiene un flujo similar, pero difiere en la función que realiza el servicio, puesto que en vez de usar la función *push*, Ilustración 53, se usa la función *update*, Ilustración 57 y en ella se mandan los nuevos datos.

```

updateRoute(route: Routes) {
  this.routesList.update(route.$key, {
    name: route.name,
    travel: route.travel,
    description: route.description,
  });
}
    
```

Ilustración 57 - Función actualizar ruta en servicio

3.6.2.7.2. Componente tabla

El *template* del componente tabla está basado en la etiqueta *table* de html, con la particularidad que el contenido mostrado a partir de una consulta a la base de datos se muestra usando la característica “*ngFor”¹¹ de Angular, tal y como se observa en la Ilustración 58.

```

<tbody>
  <tr *ngFor="let route of routesList">
    <td>{{route.name}}</td>
    <td>{{route.description}}</td>
    <td>{{route.travel}}</td>
    <td>
      <a class="btn btn-color text-white" (click)="onEdit(route)">
        <i class="far fa-edit"></i>
      </a>
      <a class="btn btn-danger text-white" (click)="onDelete(route.$key)">
        <i class="fas fa-trash-alt"></i>
      </a>
    </td>
  </tr>
</tbody>
    
```

Ilustración 58 - ngFor de Angular

El *ngFor* se asemeja a un ciclo *for* con la particularidad que se implementa en el documento html. En este caso en particular, se asigna la variable *route* para cada ciclo del objeto completo

¹¹ Directiva encargada de presentar una lista de elementos en pantalla

llamado `routesList`, dicho objeto es creado en el *component* a partir de una clase llamada *Routes*, Ilustración 59.

```
import { DriverHegi } from './driver.model';

export class Routes {

  constructor($key = '',
    name = '',
    travel = '',
    description = '',
    drivers = [],
  ) {
    this.$key = $key;
    this.name = name;
    this.travel = travel;
    this.description = description;
    this.drivers = drivers;
  }
  $key?: string;
  name?: string;
  travel?: string;
  description?: string;
  drivers?: DriverHegi[];
}
```

Ilustración 59 - Clase Routes

El primer paso para obtener todas las rutas que se encuentran creadas en la base de datos es utilizar un servicio que retorne una variable del tipo *AngularFireList* la cual trae la rama principal “*routes*”, Ilustración 55. Ahora bien, se procede a crear una función en el *ngOnInit*¹² del *component*, Ilustración 60 y mediante la función *snapshotChanges*¹³ se notificará por cada cambio que ocurra dentro de la lista, lo cual permite que la información se esté actualizando en el mismo momento que es creada, eliminada o actualizada.

```
ngOnInit() {
  return this.routeService.getRoutes()
    .snapshotChanges().subscribe(item => {
      this.routesList = [];
      item.forEach(element => {
        const x = element.payload.toJSON();
        x['$key'] = element.key;
        this.routesList.push(x as Routes);
      });
    });
}
```

Ilustración 60 - Función snapshotChanges

¹² Este método del ciclo de vida se llama inmediatamente después de la creación del componente.

¹³ Es un estado actual de una colección de Firebase, devuelve un observable de datos.

Dentro de la función `snapshotChanges`, el *length* o tamaño de *ítem* es equivalente a la cantidad de rutas que se encuentren bajo la rama de *“routes”*, por ende el *forEach* permite recorrerlo y asignar cada iteración a un *array* del tipo `Routes` llamado *routerList*, *array* que es usado por el *template* para imprimir los resultados en la tabla, Ilustración 58.

3.6.2.8. Perfil de usuario

La estructura del componente de perfil tiene tres componentes hijos, los cuales cumplen la función de actualizar información, cambiar contraseña y cambiar imagen de perfil.

El *template* de actualizar información está basado en formularios con campos de entrada en los cuales se traen los datos existentes del perfil de usuario que esté con la sesión activa, permitiendo modificarlos y posteriormente guardar los cambios. Por su parte, el componente gestiona la lectura y escritura en la base de datos con las funciones ya mostradas, como *snapshotChanges*.

El cambio de contraseña se realiza mediante una función de *firebase* en la cual es necesario crear una instancia del usuario, y asignarle a una constante el objeto de la instancia cuando invoca una función llamada *auth()* aplicada al usuario actual, tal y como se observa en la Ilustración 61.

```
changeAuthPassword(newPass) {  
  const user = firebase.auth().currentUser;  
  return user.updatePassword(newPass);  
}
```

Ilustración 61 - Función para cambiar contraseña

Para la gestión imagen de perfil, se hace uso de una funcionalidad de *Firebase*, el *storage*, Ilustración 45. Dicha funcionalidad como se mencionó, permite almacenar archivos en el mismo entorno de *Firebase*.

El manejo de usuarios en base de datos está basado de un identificador único denominado *UID* de usuario, en adelante *keyUser*. Entonces, las fotos de perfil van a ser nombradas por su identificador correspondiente, Ilustración 63, dentro de una carpeta previamente creada en *Firebase* llamada *imagesProfile*, Ilustración 62.

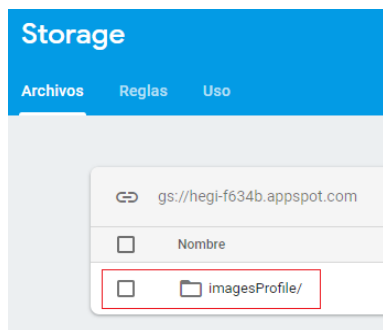


Ilustración 62 - Carpeta que contiene fotos de perfil en firebase

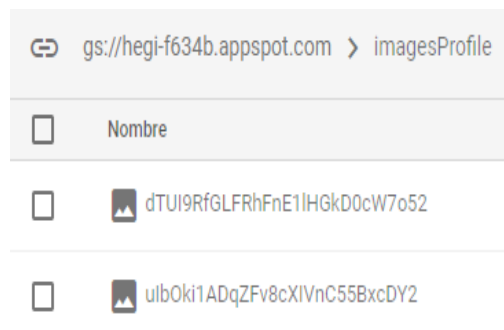


Ilustración 63 - Fotos de perfil con identificador único

Lo primero que debe hacerse es obtener el `keyUser` de la sesión activa, para lo cual, se debe recuperar la instancia de Firebase y consumir la función `auth()`, tal y como se observa en la Ilustración 61, una vez hecho esto, se puede acceder al `keyUser` de la forma `"user.uid"`.

Una vez se tiene el identificador único de usuario, hace falta consumir una función dentro del servicio que inserte la imagen que haya sido seleccionada desde el `template`, tal y como se observa en la Ilustración 64.

```
saveImageProfile(keyUser, img) {
  const storageRef = firebase.storage().ref(`imagesProfile/${keyUser}`);
  return storageRef.putString(img, 'base64', { contentType: 'image/png' });
}
```

Ilustración 64 - Función para guardar imagen de perfil en firebase

Se observa en la ilustración anterior que primero se crea una referencia en la dirección y nombre que va a tener el recurso (en este caso una imagen) para luego hacer uso de la función `putString`.

3.6.2.9. Sección de gráficas

La vista de la Ilustración 38 del apartado de diseño corresponde a la sección que se describe en esta parte de la implementación. El componente graficas tiene 3 hijos: tabla, gráfica día y gráfica por rango. Para facilidad de comprensión, en la Tabla 22 se describen los componentes involucrados y la forma en que serán mencionados en el presente documento:

Tabla 22 – Componentes de la sección de gráficas

NOMBRE DEL COMPONENTE	IDENTIFICADOR
GraphsComponent	Padre
TablesComponent	Tabla
ChartsComponent	Grafica
ChartRangeComponent	Grafica de rango

3.6.2.9.1. Búsqueda para un solo día

Lo primero que el usuario hace en esta vista es seleccionar la ruta que quiere buscar, la cual está asociada al fragmento de código de la Ilustración 65.

```
<div class="form-group mx-auto col-md-3">
  <label>Ruta:</label>
  <select (ngModelChange)="onChangeSl($event)" name="route"
    class="form-control text-center" required
    #route="ngModel" [ngModel]="">
    <option [value]="noValue">Seleccione Ruta</option>
    <option *ngFor="let route of routesList" [value]="route.name">
      | {{route.name}}
    </option>
  </select>
</div>
```

Ilustración 65 - Selector de búsqueda de ruta

Las opciones que se cargan en dicha entrada, vienen de una consulta de todas las rutas disponibles, dicha consulta guarda su resultado en la variable *routesList*. Cuando un usuario selecciona la ruta de preferencia, es llamada la función *onChangeSl* con el parámetro *\$event*, la cual a su vez llama una nueva función llamada *getBuses*, de parámetro *route*, tal y como se observa en la Ilustración 66.

A continuación, se consume el servicio del componente por medio de la función *getBuses*, la cual retorna una lista de un objeto del tipo *AngularFireList* con una rama principal llamada 'buses'. Luego de esto, se llena el objeto *busesList[]* que es del tipo *Bus*, cuyos atributos se observan en la Tabla 23.

Tabla 23 - Atributos de la clase bus

NOMBRE	TIPO
<i>\$key</i>	<i>String</i>
<i>plaque</i>	<i>String</i>
<i>model</i>	<i>String</i>
<i>Brand</i>	<i>String</i>
<i>capacity</i>	<i>Number</i>
<i>route</i>	<i>String</i>

```
onChangeSl(route: string) {
  this.getBuses(route);
  this.flagSearch = false;
}

getBuses(route: string) {
  this.flagRoute = true;
  return this.busService.getBuses()
    .snapshotChanges().subscribe(item => {
      this.busesList = [];
      this.busesInRoute = [];
      item.forEach(element => {
        const x = element.payload.toJSON();
        x['$key'] = element.key;
        this.busesList.push(x as Bus);
      });
      for (let i = 0; i < this.busesList.length; i++) {
        if (this.busesList[i].route === route) {
          this.busesInRoute.push(this.busesList[i]);
        }
      }
    });
}
```

Ilustración 66 - Función para obtener buses

Una vez se tiene el objeto completo *busesList*, se hace un filtro para evaluar todos aquellos que correspondan a la ruta ingresada por el usuario. Lo anterior se hace por medio de un ciclo *for*, que iguala la variable *busesList[iteración].route* con *route* (variable que llegó como parámetro) y posteriormente si hay coincidencia, se hace un *push* al objeto *busesInRoute*. Entonces, *busesInRoute* es el arreglo que va al segundo selector para que, una vez elegida la ruta, se seleccione el bus en particular al cual le quiere realizar la búsqueda.

A partir de lo anterior, seleccionada la fecha de interés, es llamada la función *createGraph*. En ella, lo que se hace principalmente es evaluar la existencia de una segunda fecha, la cual indicaría que es una búsqueda por rango. Una vez se identifica que la búsqueda corresponde a un día en específico son llamadas tres funciones que se estudian a continuación.

Lo primero que se hace es recuperar la lista de conteo de pasajeros, generada por el DMC, la cual se visualiza en la base de datos de firebase tal y como se observa en la Ilustración 67. Ahora bien, por medio de un objeto de tipo *AngularFireList*, asignando la referencia '*counter*' se recupera la lista completa en la función *getCounterInDay*, y posteriormente se le asigna a un objeto llamado *counterList*, Ilustración 68. Creado el arreglo, se llama la segunda función, *existDate*, la cual tiene como finalidad, encontrar el índice del objeto previamente almacenado en el cual coinciden los valores de fecha y placa del vehículo que se está buscando, Ilustración 69. De no encontrarse coincidencia, se notifica al usuario por medio de un mensaje de tipo alerta que no se encuentran resultados, en caso contrario, se llama la función *getApproach*, con parámetro *x*, Ilustración 70, donde *x* es el índice encontrado en la segunda función.

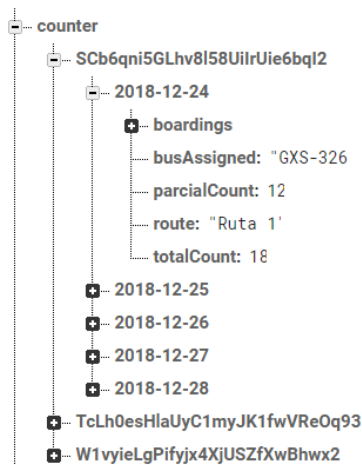


Ilustración 67 - Lista counter en firebase

```

getCounterInDay(plaque: string) {
  if (plaque !== 'noPlaque') {
    this.graphsService.getCounters()
      .snapshotChanges().subscribe(item => {
        this.counterList = [];
        item.forEach(element => {
          const x = element.payload.toJSON();
          x['$key'] = element.key;
          this.counterList.push(x);
        });
        if (this.counterList && this.counterExistDate === 0) {
          this.existDate();
          this.counterExistDate = 1;
        }
      });
  } else {
    window.alert('Seleccione un bus');
  }
}

```

Ilustración 68 - Lista completa de abordajes desde Firebase

```

existDate() {
  let coin = 0;
  for (let x = 0; x < this.counterList.length; x++) {
    if (this.counterList[x]['${this.dateI}']) {
      if (this.plaque === this.counterList[x]['${this.dateI}']['busAssigned']) {
        coin = 1;
        this.getApproach(x);
        x = this.counterList.length - 1;
      }
    }
  }
  if (coin === 0) {
    window.alert('No se encuentran resultados');
    this.graphsService.chartSubjetc.next(true);
  }
}

```

Ilustración 69 - Función para encontrar índice de coincidencia

```

getApproach(indice: number) {
  const key = this.counterList[`${indice}`][`${key}`];
  setTimeout(() => {
    this.graphsService.getBoardings(key, this.dateI)
      .snapshotChanges().subscribe(item => {
        this.boardingsList = [];
        item.forEach(element => {
          const x = element.payload.toJSON();
          x[`${key}`] = element.key;
          this.listBoardingsInDay.push(x as Approach);
          this.datesForTable.push(this.dateI);
        });
        this.graphsService.chartSubjetc.next(true);
      });
  }, 1000);
}

```

Ilustración 70 - Función para obtener abordajes en fecha específica

Lo primero que se hace dentro de la función *getApproach*, es recuperar la llave de inserción que corresponde al índice. Para entender el concepto de llave de inserción, se puede entender analizando que la Ilustración 67 muestra que dentro de la rama *counter*, existen tres sub listas con identificadores únicos, dichos identificadores únicos son la llave de inserción.

Una vez hecho lo anterior, se consume la función *getBoardings* en el servicio, la cual tiene dos parámetros, la llave de inserción y la fecha proveniente del template. Finalmente, de esta función se obtienen dos variables, la primera es *listBoardingsInDay* y la segunda es *datesForTable*.

Como se observa en la Ilustración 70, *datesForTable* es un array que contiene únicamente la fecha de búsqueda, esto se debe a que la jerarquía de dicha fecha está arriba de la información que contiene el arreglo *listBoardingsInDay*, Ilustración 71, y para efectos de la tabla del template, es necesario contar con dicha variable.

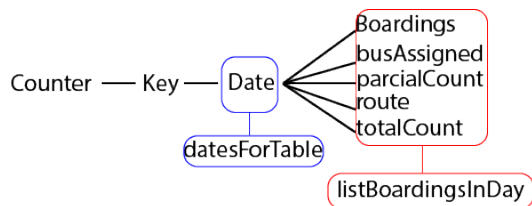


Ilustración 71 - Jerarquía de counter

Ya se cuenta con las variables necesarias para realizar las vistas en el *template*, el cual, para la búsqueda emplea los componentes tabla y gráfica.

```

<app-tables [listBoardingsInDay]="listBoardingsInDay"
[datesForTable]="datesForTable">
</app-tables>

```

Ilustración 72 - Componente tabla dentro del padre

En la Ilustración 72 se observa cómo se inyecta el componente tabla con las dos variables obtenidas previamente, las cuales deben ser declaradas en el componente tabla en forma de *inputs*, tal y como se observa en la Ilustración 73.

```
export class TablesComponent implements OnInit {  
  
  @Input() listBoardingsInDay: Approach[];  
  @Input() datesForTable = [];
```

Ilustración 73 - Declaración de variables inyectadas en componente

Una vez que las dos variables existen en el padre, son inyectadas automáticamente en el componente tabla, cuyo *template* es el que se observa en la Ilustración 74.

```
<tbody>  
  <tr *ngFor="let approach of listBoardingsInDay; let i = index">  
    <td>{{datesForTable[i]}}</td>  
    <td>{{approach.time}}</td>  
    <td *ngIf="approach.approachType === true">Ingreso</td>  
    <td *ngIf="approach.approachType === false">Salida</td>  
    <td>{{approach.parcialCount}}</td>  
    <td>{{approach.totalCount}}</td>  
    <td>{{approach.lat}}</td>  
    <td>{{approach.lng}}</td>  
  </tr>  
</tbody>
```

Ilustración 74 - Template componente tabla

La visualización de la gráfica está a cargo del componente gráfica descrito en la Tabla 22, el cual de forma similar al de tablas, se inyecta desde el componente padre, Ilustración 75.

```
<app-charts *ngIf="flagChart == true"  
  [listBoardingsInDay]="listBoardingsInDay">  
</app-charts>
```

Ilustración 75 - Componente grafica inyectado desde el componente padre

Para dicha visualización, se hace uso de la dependencia *ng2-charts*, cuyo código fuente está en GitHub [65]. La dependencia debe importarse en el módulo de la aplicación y su forma de llamarse desde el template del componente respectivo es como se muestra en la Ilustración 76.


```

<div style="display: block">
  <canvas *ngIf="flag == true" baseChart
    [datasets]="barChartData"
    [labels]="barChartLabels"
    [options]="barChartOptions"
    [legend]="barChartLegend"
    [chartType]="barChartType"
    (chartHover)="chartHovered($event)"
    (chartClick)="chartClicked($event)"></canvas>
</div>

```

Ilustración 76 - Template de componente grafica

Los parámetros más importantes dentro de la etiqueta canvas son *datasets*, *labels* y *options*. De donde *datasets* corresponde a los datos que van a ser graficados, *labels*, son las etiquetas del eje horizontal de la gráfica y *options* corresponde a los ajustes de forma de la dependencia.

Una particularidad del uso de esta dependencia es que la renderización ocurre únicamente cuando el componente padre es cargado, de modo que cuando el usuario realiza la búsqueda y las variables son obtenidas, aunque a la componente gráfica sea inyectada la variable *listBoardingsInDay*, se debe agregar un observable [Anexo H] del tipo *BehaviorSubject*, en el servicio, para que cuando el evento de búsqueda sea efectuado en el componente padre, este genere un cambio en dicho observable y mediante una suscripción, Ilustración 77 permanente en el constructor del componente gráfica, se va a saber cuándo debe volver a renderizarse el elemento canvas, Ilustración 76.

```

constructor(public graphsService: GraphsService) {
  this.graphsService.chartSubjetc.subscribe((message: boolean) => {
    if (message === true) {
      setTimeout(() => {
        if (this.listBoardingsInDay) {
          this.flag = false;
          this.totalPassegers = [];
          this.parcialPassegers = [];
          this.determinateArrays();
        }
      }, 500);
    } else {
      // this.flag = true;
    }
  });
}

```

Ilustración 77 - Suscripción a observable para renderizar gráfica

Entonces, el parámetro *barChartData*, Ilustración 76, se inicializa vacío cuando se carga el componente, y una vez la suscripción indique un evento de búsqueda, es llamada la función *determinateArrays*, la cual se encarga de gestionar dos funciones, una de las cuales renderiza la vista con un *delay* (en Angular *setTimeout*) de medio segundo para asegurar la correcta carga de datos, mientras que la otra función, Ilustración 78 e Ilustración 79, se encarga de

manipular la variable que fue inyectada y crear los vectores necesarios para convertirse en la entrada de datos de la dependencia de gráficas.

```
fillVector() {
  let totalCount = 0;
  let parcialCount = 0;
  for (let x = 0; x < this.listBoardingsInDay.length; x++) {
    if (this.listBoardingsInDay[x].approachType === true) {
      // tslint:disable-next-line:no-unused-expression
      this.intHour = this.listBoardingsInDay[x].time.split(':')[0];
      switch (this.intHour) {
        case '05':
          totalCount = totalCount + 1;
          parcialCount = parcialCount + 1;
          this.parcialPassegers[0] = parcialCount;
          this.totalPassegers[0] = totalCount;
          break;

```

Ilustración 78 - Creación de vectores para dependencia de graficas 1/2

```

} else {
  parcialCount = parcialCount - 1;
  if (x === this.listBoardingsInDay.length - 1) {
    switch (this.intHour) {
      case '05':
        this.parcialPassegers[0] = parcialCount;
        break;

```

Ilustración 79 - Creación de vectores para dependencia de graficas 2/2

La función *fillVector*, recorre la variable de entrada *listBoardingsInDay*, donde inicialmente se verifica si el tipo de ascenso en la posición *x* del ciclo *for* es *true* (ascenso) o *false* (descenso), de modo que el primer condicional corresponde a un evento de ascenso. Lo primero que se toma es el vector en posición cero del *split(':')*¹⁴ de la hora de la muestra, la cual viene en formato de HH:MM:SS, de modo que la variable *intHour* tiene la hora de abordaje, a partir de la cual se hace una función *switch* evaluando las posibles 19 horas de muestras, ya que las rutas empiezan su recorrido alrededor de las 5 am hasta la media noche. En cada *case* de la función *switch*, se evalúa el total y parcial de pasajeros, aumentando en uno cada vez que se da una coincidencia, se debe tener en cuenta que las posiciones de los vectores *parcialPassegers* y *totalPassegers* corresponden a las posibles 19 horas de muestra, de tal forma que la posición cero (0) de cualquiera de los dos vectores, corresponde a las 5 de la mañana. La Ilustración 79, muestra la codificación cuando el evento es de descenso, entonces, la cuenta parcial disminuye en uno y esta se iguala a la posición respectiva de la muestra.

Cuando los arreglos son terminados, se crean los objetos de entrada que la dependencia *ng2-charts* necesita, tal y como se observa en la Ilustración 80.

¹⁴ Función que sirve para dividir una cadena en partes utilizando un carácter delimitador, en este caso el carácter dos puntos.

```
barChartDataFunc(): any {
  return [
    { data: this.totalPassegers, label: 'Pasajeros totales' },
    { data: this.parcialPassegers, label: 'Pasajeros en vehiculo' }
  ];
}
```

Ilustración 80 - Creación de objeto *barChartData*

3.6.2.9.2. Búsqueda para un rango de días

El algoritmo para la búsqueda en un rango de días aumenta su complejidad, pero sigue un flujo similar al estudiado en la sección anterior. Son añadidas nuevas funciones y se renderiza un componente de gráfica diferente. Todo el proceso de codificación está descrito en el Anexo F.

3.6.2.10. Sección de mapa de calor y gráfica auxiliar

La sección que se implementa a continuación, corresponde a la interfaz de la Ilustración 39, la cual está dividida principalmente en dos componentes hijos, mapa y gráfica; ambos necesitan de la inyección de una variable del tipo *Approach* en su invocación hecha en el template del componente padre de esta sección, Ilustración 81.

```
<div class="col-md-8 mx-auto mt-3 mb-4">
  <app-map-ubication [coordinates]="coordinates">
  </app-map-ubication>
</div>

<div class="col-md-8 mx-auto mt-3 mb-4">
  <app-graph-line [coordinates]="coordinates">
  </app-graph-line>
</div>
```

Ilustración 81 - Dos componentes hijos llamados en el template del componente padre

Coordinates es en este caso la variable inyectada a ambos componentes; para obtenerla se sigue una codificación parecida a la estudiada en la sección de gráficas, dado que el usuario sigue el mismo flujo de búsqueda donde primero se selecciona la ruta de interés, que activa una función, permitiéndole mostrar los buses disponibles para dicha ruta y únicamente dispone de una entrada de tipo *date*, inhabilitando el rango de días en esta sección.

Entonces, primero se traen todas las listas con raíz "*counter*", luego se encuentra una coincidencia entre la fecha y el bus, la cual genera un índice que a su vez permite realizar la consulta exacta con los parámetros de llave de inserción y fecha para poder consumir la función del servicio que se muestra en la Ilustración 82.

```
getCoordinates(key, date) {  
  return this.afDatabase.list(`counter/${key}/${date}/boardings`);  
}
```

Ilustración 82 - Función en servicio para traer coordenadas de una búsqueda

El componente de gráfica se comporta de manera similar a los componentes que consumen la misma dependencia ya descritos en el documento, de forma que la notificación de búsqueda se da por medio de un observable de tipo *BehaviorSubject*. La gráfica de este componente a diferencia de las anteriores es del tipo línea, por ende, necesita la existencia de 2 vectores para mapear la información, ambos son construidos, Ilustración 83 a partir de la variable inyectada.

```
determinateArrays() {  
  for (let x = 0 ; x < this.coordinates.length; x++) {  
    this.parcialCountArray.push(this.coordinates[x].parcialCount);  
    this.hourArray.push(this.coordinates[x].time);  
  }  
  this.changeChart();  
}
```

Ilustración 83 - Creación de vectores para grafica de línea

Cuando los vectores son creados, se realiza la creación de los dos objetos necesarios para la correcta renderización de la gráfica, tal y como se muestra en la Ilustración 84.

```
lineChartDataFunc(): any {  
  return [  
    { data: this.parcialCountArray, label: 'Pasajeros en bus' }  
  ];  
}  
  
lineChartLabelsFunc(): any {  
  return this.hourArray;  
}
```

Ilustración 84 - Objetos para renderización de gráfica

Finalmente, para la implementación del componente de mapa, fue necesario integrar una librería de JavaScript e integrarla al proyecto de la aplicación en Angular, para lo cual el código fuente de dicha librería se agrega a la carpeta *assets*, Ilustración 85.

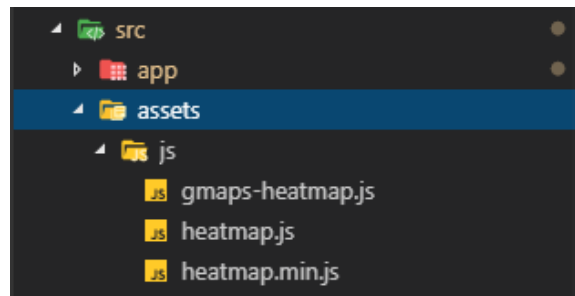


Ilustración 85 - Código fuente de librería heat-map

Luego, es necesario agregar la referencia al archivo *angular.json* (Ilustración 86), y al *index* de la aplicación (Ilustración 87) para que los recursos sean correctamente cargados.

```

    "scripts": [
      "node_modules/jquery/dist/jquery.js",
      "node_modules/popper.js/dist/umd/popper.js",
      "node_modules/bootstrap/dist/js/bootstrap.js",
      "src/assets/js/gmaps-heatmap.js",
      "src/assets/js/heatmap.js"
    ]
  }

```

Ilustración 86 – Inclusión del código de librería en archivo angular.json

```

<script type="text/javascript" src="assets/js/gmaps-heatmap.js"></script>
<script type="text/javascript" src="assets/js/heatmap.js"></script>

```

Ilustración 87 – Inclusión del script del código fuente de la librería en el index de la aplicación

El template de dicho componente se muestra en la Ilustración 88.

```

<div class="heatmap" id="map-canvas"
  style="width: 100%;height: 500px;cursor:pointer;position:relative">
  </div>

```

Ilustración 88 - Template de componente hijo mapa

Fue necesario realizar un modelamiento de dos clases para emular el objeto con el que la librería realiza la creación del mapa de calor, Ilustración 89 e Ilustración 90.

```

export class DataC {
  constructor(
    data = [],
  ) {
    this.data = data;
  }
  data?: Coordinates[];
}

```

Ilustración 89 - Clase 1 para mapa de calor

```
export class Coordinates {  
  constructor(  
    lat = 0,  
    lng = 0,  
  ) {  
    this.lat = lat;  
    this.lng = lng;  
  }  
  lat?: number;  
  lng?: number;  
}
```

Ilustración 90 - Clase 2 para mapa de calor

Entonces, en el componente se declara una variable de tipo *DataC*, una de tipo *Approach* inyectada en el padre y los valores centrales de inicialización de latitud y longitud para el mapa, Ilustración 91.

```
export class MapUbicationComponent implements OnInit {  
  @Input() coordinates: Approach[];  
  coordinatesMap: DataC = new DataC();  
  latC = 3.8945442;  
  lngC = -76.3001605;
```

Ilustración 91 - Declaración de variables para mapa de calor

La inicialización del mapa depende de ciertos objetos, el primero de ellos define el centro del mapa por medio de una instancia propia del servicio de google Ilustración 92.

```
const myLatLng = new google.maps.LatLng(this.latC, this.lngC);
```

Ilustración 92 - Definición del centro de mapa de calor

A continuación, se definen las opciones generales del mapa, entre las que se encuentra el zoom predeterminado, la posibilidad de navegar en él y otras configuraciones con sus valores por defecto Ilustración 93.

```
const myOptions = {  
  zoom: 14,  
  center: myLatLng,  
  gestureHandling: 'greedy',  
  mapTypeId: google.maps.MapTypeId.ROADMAP,  
  disableDefaultUI: false,  
  draggable: true,  
  navigationControl: true,  
  mapTypeControl: true,  
  disableDoubleClickZoom: false,  
  zoomControl: true,  
  scaleControl: true  
};
```

Ilustración 93 - Opciones generales de mapa de calor

De forma similar se realizan configuraciones previas y la asociación del *div* del template con el controlador, para finalmente asignar una variable y generar el mapa, Ilustración 94.

```
heatmapoverlay.setData(this.coordinatesMap);
```

Ilustración 94 - Agregar la información para generar mapa de calor

La asignación de valores del objeto *coordinatesMap* se visualiza en la Ilustración 95, lo primero que se observa es la limpieza del mismo objeto, dado que cada vez que la función es invocada, la información debe vaciarse. Luego, se reasignan los valores centrales de latitud y longitud para asegurarse que el centro del mapa se genere cerca de las muestras provenientes de la variable inyectada por el padre; finalmente, dentro de un ciclo con el tamaño de la longitud de la variable *coordinates*, se crea un objeto temporal en la constante *tempCoord* que almacena la latitud y longitud del recorrido, para luego asignar esa constante el objeto *coordinatesMap* en su parámetro *data*. Una vez el ciclo termina, se invoca la función *initMap* para la renderización del mapa.

```
createObject() {  
  this.coordinatesMap = new DataC();  
  this.latC = this.coordinates[0].lat;  
  this.lngC = this.coordinates[0].lng;  
  for (let y = 0; y < this.coordinates.length; y++) {  
    const tempCoord = {  
      'lng': this.coordinates[y].lng,  
      'lat': this.coordinates[y].lat,  
    };  
    this.coordinatesMap.data.push(tempCoord);  
    if (y === this.coordinates.length - 1) {  
      this.initMap();  
    }  
  }  
}
```

Ilustración 95 - Creación de objeto legible para librería de mapa de calor

3.6.3. Implementación DMC

Esta sección analiza la implementación del dispositivo móvil de captura. En primera medida, se describe la tecnología de desarrollo para posteriormente describir las partes más importantes de la implementación en este módulo del dispositivo.

3.6.3.1. Tecnología de desarrollo

El desarrollo de aplicaciones móviles puede ser visto desde 3 grandes flancos: aplicaciones nativas, aplicaciones Web App y Web App nativas.

Las aplicaciones nativas [66] son las que se desarrollan de forma específica para un determinado sistema operativo llamado *Software Development Kit* o *SDK*. Cada una de las plataformas tiene un sistema operativo diferente, particularmente para Android, las aplicaciones se desarrollan en lenguaje Java o Kotlin. La principal ventaja con respecto a los otros tipos de aplicaciones es la posibilidad de acceder a todas las características del hardware del móvil como la cámara, GPS, entre otras. Por tanto, la experiencia del usuario es considerablemente mejor.

Una aplicación Web App [67] se desarrolla con lenguajes conocidos como HTML Javascript y CSS. Una de las ventajas con respecto a las aplicaciones nativas es la posibilidad de programar independientemente del sistema operativo en el que la aplicación va a ser usada, característica que hace que no se tengan que crear distintas aplicaciones. Estas aplicaciones se ejecutan dentro del propio navegador web del dispositivo a través de la URL y no necesitan de instalación.

Finalmente, las aplicaciones Web App nativas [68] también conocidas como aplicaciones híbridas, son la combinación de las mejores características de las dos mencionadas anteriormente. Pueden ser desarrolladas en lenguajes de las Web App, por lo que permite su uso en diferentes plataformas, pero también dan la posibilidad de acceder a gran parte de las características del hardware del dispositivo, un ejemplo de framework de este tipo de aplicaciones es Ionic.

Dado que para el presente trabajo de grado se necesita un rendimiento óptimo que permita un correcto aprovechamiento de los recursos hardware del dispositivo para soportar los algoritmos de procesamiento de imágenes, se opta por realizar una aplicación nativa Android Studio como entorno de desarrollo. Así mismo, se selecciona a Kotlin como lenguaje de programación, puesto que presenta un 100% de interoperabilidad con Java y tiene funcionalidades modernas como las llamadas seguras, en las cuales se evitan errores de nulos, además fue catalogado como lenguaje oficial para desarrollo de aplicaciones nativas Android [69].

3.6.3.2. Creación de proyecto y adición de dependencias

La creación del proyecto requiere seguir la secuencia de opciones que Android Studio ofrece, en el cual se detalla el nombre de la aplicación, localización del proyecto, entre otras configuraciones básicas, las cuales se pueden encontrar en tutoriales básicos de desarrollo en Android [70]. Los detalles más importantes a tener en cuenta son el soporte con Kotlin y la elección de una versión de Android adecuada, en este caso, dado que no se requieren funcionalidades modernas, se elige una compatibilidad con la versión 4.0.3, la cual abarca cerca de un 100% de dispositivos.


```
//Circle imageview
implementation 'de.hdodenhof:circleimageview:2.2.0'

//Circe Indicaroe
implementation 'me.relex:circleindicator:1.2.2@aar'

implementation 'com.google.android.gms:play-services-analytics:16.0.3'
//Firebase
implementation 'com.google.firebase:firebase-core:16.0.3'
implementation 'com.google.firebase:firebase-auth:16.0.3'
implementation 'com.google.firebase:firebase-database:16.0.2'

testImplementation 'junit:junit:4.12'
androidTestImplementation 'com.android.support.test:runner:1.0.2'
androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'

//Blurry
implementation 'jp.wasabeef:blurry:2.1.1'

// Material
// implementation 'com.google.android.material:material:1.0.0'

// Google Location
implementation 'com.google.android.gms:play-services-location:15.0.1'
```

Ilustración 96 - Dependencias necesarias para DMC prototipo 1

Las primeras tres librerías más la de “*Blurry*” son importadas para la visualización del perfil de usuario, en el cual se tiene por el diseño mostrado en la Ilustración 44 que la imagen de perfil es circular y, con la dependencia *Blurry*, se le puede agregar un efecto de distorsión a una imagen para efectos de *background* en el diseño. Luego, son importadas las dependencias necesarias para la integración del proyecto con firebase y la localización de google, Ilustración 96.

3.6.3.3. Estructura del proyecto

Dentro del paquete principal de desarrollo, se crea una estructura organizada con los componentes necesarios para la interacción de todas las partes de la aplicación, tal y como se observa en la Ilustración 97. El primer paquete, corresponde a *data*, en donde se encuentran los modelos, las preferencias y los *singleton* usados. La carpeta *ui* o interfaz de usuario contiene los archivos de las clases que corresponden a las *activities* o pantallas, mientras que en la carpeta *util*, tiene clases auxiliares de configuración y extensiones usadas. Finalmente, la clase *App* es un archivo que se llama desde el *manifest*, el cual genera instancias únicas que sean accesibles para toda la aplicación de ciertas entidades necesarias que se estudian más adelante.

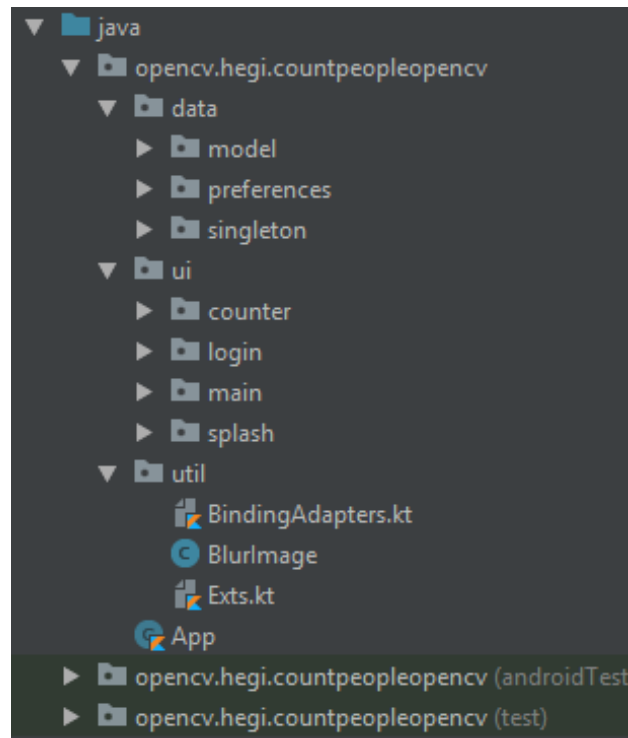


Ilustración 97 - Estructura de carpetas del DMC prototipo 1

3.6.3.4. Conexión de Firebase con DMC

La conexión por parte de Android Studio con Firebase es muy parecida a como se hace para un proyecto en Angular. Se debe ingresar el nombre del paquete de la aplicación móvil, e inmediatamente después, es generado un archivo llamado `google-services.json`, el cual se agrega en el proyecto a nivel de la carpeta `src`, tal como se observa en la Ilustración 98.

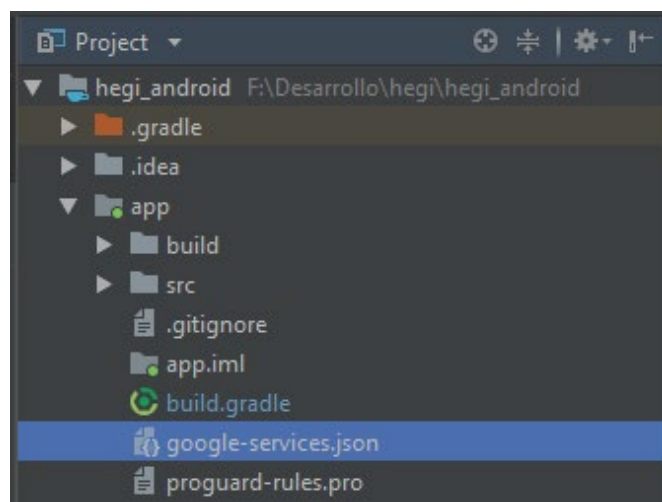


Ilustración 98 - Archivo de configuración de Firebase para aplicación móvil

3.6.3.5. Permisos de archivo manifest

El *AndroidManifest.xml* es un archivo de configuración donde se pueden aplicar las configuraciones básicas de la aplicación. Dicho archivo está situado en la raíz del proyecto.

La accesibilidad a ciertos componentes *hardware* del dispositivo requieren de permisos, los cuales se declaran en el documento de configuración *manifest*. En el caso particular de este prototipo, se requieren dos permisos, uno que corresponde al acceso a internet y otro que permita el uso del GPS, la declaración de dichos permisos se observa en la Ilustración 99.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Ilustración 99 - Permisos de cámara y GPS android

3.6.3.6. Actividad de inicio de sesión

En Android, las actividades se dividen en su clase Java o Kotlin que contiene la lógica y su Layout, el cual es un archivo XML que genera el diseño de la pantalla.

3.6.3.6.1. Layout de inicio de sesión

Para el diseño de las interfaces, se hace uso de una nueva tecnología de Android Studio llamada *ConstraintLayout*, la cual permite crear interfaces de usuario en distintos tamaños y resoluciones con una jerarquía de vista plana. Todas las vistas se establecen según las relaciones entre las vistas de hermanos y el diseño de los padres.

La forma de indicarle a Android que se hace uso de *ConstraintLayout*, es incluir en la etiqueta de inicio su declaración, tal y como se muestra en la Ilustración 100.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
```

Ilustración 100 - Etiqueta de ConstraintLayout

El diseño de la interfaz puede hacerse de forma programática o por medio del entorno de edición que provee Android. Para fines prácticos se muestra la vista en su composición gráfica, Ilustración 101, aunque se debe resaltar que todo en cuanto al diseño gráfico se refiere, genera su respectivo código. Toda la aplicación se diseña en horizontal, programáticamente hablando, en orientación *Landscape*, dado que el reconocimiento y conteo se efectúa en esta orientación, entonces se hace una adaptación al diseño presentado.

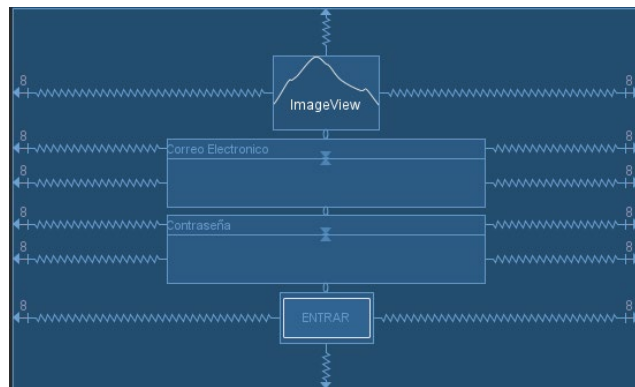


Ilustración 101 - Login en ConstraintLayout

Básicamente el *Layout* consta de tres bloques principales, la imagen de logo, los campos de entrada y el botón, todos ellos centrados y relativos entre sí.

3.6.3.6.2. Clase Kotlin de inicio de sesión

La función principal que se encuentra definida en la actividad es la que se muestra en la Ilustración 102

```

@SuppressLint( ..value: "CheckResult")
override fun onResume() {
    super.onResume()

    loginAcceptBtn.clicks()
        .subscribe {
            username = loginUsernameEdt.text()
            password = loginPasswordEdt.text()
            if (!TextUtils.isEmpty(username) && !TextUtils.isEmpty(password)) {
                mProgressBar!!.setMessage("Autenticando...")
                mProgressBar!!.show()

                mAuth!!.signInWithEmailAndPassword(username!!, password!!)
                    .addOnCompleteListener(this) { task ->
                        mProgressBar!!.hide()
                        if (task.isSuccessful) {
                            // Success Login
                            UserSession.isLogged = true
                            startActivity<MainActivity>()
                        } else {
                            // Error in Login
                            toast("Usuario o contraseña incorrectos")
                        }
                    }
            } else {
                toast("Campos requeridos")
            }
        }
}

```

Ilustración 102 - Función de inicio de sesión DMC prototipo 1

La función está ubicada dentro del *onResume*¹⁵ del ciclo de vida y depende del identificador que se le asignó al botón de inicio de sesión en el *Layout*, en este caso *loginAcceptBtn*. Cuando el evento de *submit* se detecta, se guardan en dos variables los campos de correo electrónico y contraseña digitados por el usuario. En este punto, cabe resaltar que se hace uso de extensiones para retornar dichos campos, lo cual mejora el rendimiento y evita redundancia de código. La función *loginUsernameEdt.text* se vale de la extensión mostrada en la Ilustración 103, en la cual se reemplaza una función completa por la definición *EditText.text*.

```
import ...  
  
fun EditText.text(): String = text.toString()
```

Ilustración 103 - Extensión para obtener texto de un campo de tipo *EditText* de Android

Luego de guardar las dos variables y verificando que ambas existan, se consume una función de firebase a partir de una instancia del tipo *FirebaseAuth*, la cual tiene acceso a *signInWithEmailAndPassword*; función que verifica a partir de dos parámetros de entrada, si las credenciales escritas por el usuario son correctas o no.

Si los valores son correctos, se produce una navegación a la actividad llamada *MainActivity*. Es de resaltar que antes de realizar dicha navegación, se está asignando una variable en valor booleano *true* para efectos de persistencia de sesión, dicha variable pertenece a la clase *UserSession*, clase que se inicializa en el documento App mencionado con anterioridad. El código de la clase *UserSession* se muestra en la Ilustración 104. En ella se observa que se hace uso del tipo *SharedPreferences*, el cual se asemeja a la persistencia de la AWGI con su *LocalStorage*.

```
import ...  
  
object UserSession {  
  
    private const val USER_LOGGED = "user_logged"  
  
    private lateinit var preferences: SharedPreferences  
  
    var isLoggedIn: Boolean  
        get() = preferences.getBoolean(USER_LOGGED, false)  
        set(value) { preferences.edit().putBoolean(USER_LOGGED, value).apply() }  
  
    fun init(context: Context) {  
        preferences = context.getSharedPreferences("UserPreferences", Activity.MODE_PRIVATE)  
    }  
}
```

Ilustración 104 - Clase *UserSession* DMC

¹⁵ Método que se llama cuando la actividad va a empezar a interactuar con el usuario después de estar en un estado de pausa.

Finalmente, de ser incorrectos los datos proporcionados, se muestra al usuario una alerta del tipo *toast* en la que se le notifica el evento.

3.6.3.7. Actividad main

Esta actividad le muestra al usuario la información pre conteo de fecha y ruta, así como información básica del conductor asignado.

3.6.3.7.1. Layout de pantalla main

Se muestra a continuación, en la Ilustración 105 , el diseño en editor de la página main. En dicha ilustración se aprecian los componentes de nombre, imagen, fecha ruta y un botón de inicio de conteo.

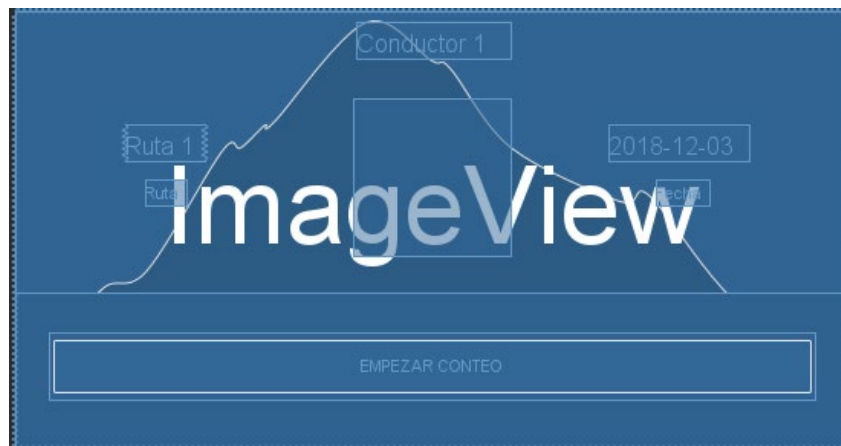


Ilustración 105 - Layout Main Activity

3.6.3.7.2. Clase Kotlin de actividad main

Al igual que en Angular, es necesario declarar referencias que apunten a las listas de la base de datos de firebase, la inicialización en la clase Kotlin se aprecia en la Ilustración 106.

```
private fun initialise() {  
    mDatabaseReference = DBConnection.db.reference.child( pathString: "driver")  
    mDatabaseReferenceCounter = DBConnection.db.reference.child( pathString: "counter")  
    mDatabaseReferenceCountePassegers = DBConnection.db.reference.child( pathString: "countPassegers")  
}
```

Ilustración 106 - Inicialización de referencias de base de datos de firebase

La función que recupera los datos mostrados en el Layout se muestra en la Ilustración 107, en la cual primero se crea una instancia del usuario actual, y a partir de la referencia inicializada con anterioridad, se realiza la consulta de los parámetros requeridos por medio de la función *onDataChange()*, función que notifica cada cambio en la base de datos y comunica directamente a la aplicación para su visualización.

```

val mUser : FirebaseUser? = DBConnection.mAuth!!.currentUser
val mUserReference : DatabaseReference = mDatabaseReference!!.child(mUser!!.uid)
mUserReference.addValueEventListener(object : ValueEventListener {
    override fun onDataChange(snapshot: DataSnapshot) {
        mainTextViewUserName.text = snapshot.child( path: "name").value as String
        driverRoute.text = snapshot.child( path: "route").value as String
        routeDriver = snapshot.child( path: "route").value as String
        busAssigned = snapshot.child( path: "busAssigned").value as String
    }
    override fun onCancelled(databaseError: DatabaseError) {}
})

```

Ilustración 107 - Función de recuperación de información de conductor

Otra función que es llamada antes de realizarse cualquier evento en el *Layout*, es la encargada de verificar que el *path* de conteo de la fecha actual existe o no, Ilustración 110, existencia que se almacena en una variable en forma de bandera como true o false, Ilustración 108, de modo que si es *true*, indica que el *path* ya existe, lo que indica que ya se han realizado conteos en el día, mientras que si el *path* no existe, al momento de que se pulse el botón empezar conteo, se creará en base de datos, Ilustración 109.

```

private fun checkExitPath() {
    // This function checks if uid/date exist
    val mUser : FirebaseUser? = DBConnection.mAuth?.currentUser
    val mUserReference2 : DatabaseReference? =
        mDatabaseReferenceCounter?.child( pathString: mUser?.uid + "/" + currentDate)
    mUserReference2?.addValueEventListener(object : ValueEventListener {
        override fun onDataChange(snapshot: DataSnapshot) {
            val ghj : Any? = snapshot.value
            if(snapshot.value != null) {
                existDateInDatabase = true
                Log.i( tag: "counter", msg: "Existe")
            } else {
                existDateInDatabase = false
                Log.i( tag: "counter", msg: "No Existe")
            }
        }
        override fun onCancelled(databaseError: DatabaseError) {}
    })
}

```

Ilustración 108 - Función para verificar existencia de un path en base de datos

```

MainBtnStartCount.clicks()
  .subscribe{
    if (existDateInDatabase && existCountInDatabase) {
      startActivity<OpenCVActivity>()
    } else {
      if(!existDateInDatabase) {
        val daily = Daily(routeDriver, busAssigned, totalCount: 0, parcialCount: 0)
        val mUser :FirebaseUser? = DBConnection.mAuth.currentUser
        val mUserReference :DatabaseReference = mDatabaseReferenceCounter!!.child(pathString: mUser!!.uid+ "/" +currentDate)
        mUserReference.setValue(daily)
      }

      if(!existCountInDatabase) {
        val countP = CountPassegers( total: 0)
        val mUserReference2 :DatabaseReference = mDatabaseReferenceCountePassegers!!.child(currentDate)
        mUserReference2.setValue(countP)
      }

      startActivity<OpenCVActivity>()
    }
  }
}
    
```

Ilustración 109 - Función de iniciar conteo DMC

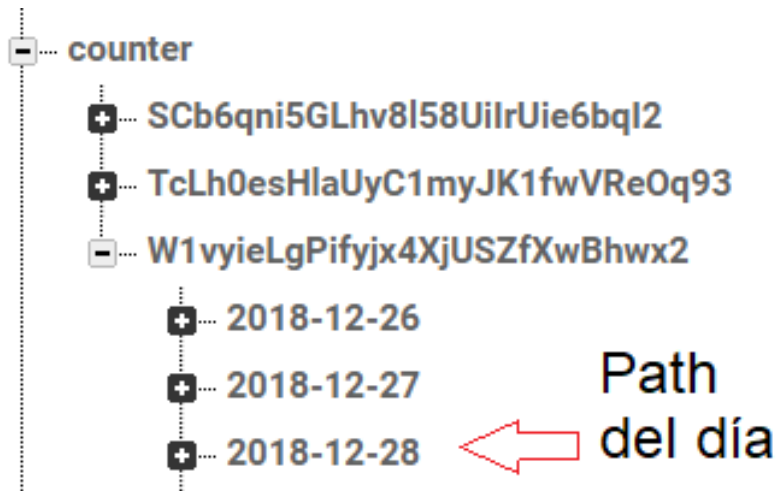


Ilustración 110 - Path de conteo de un día

3.6.3.8. Actividad de conteo simulado

Esta actividad representa la simulación de los eventos de abordaje y descenso de pasajeros de forma manual, por medio de dos botones.

3.6.3.8.1. Layout de conteo simulado

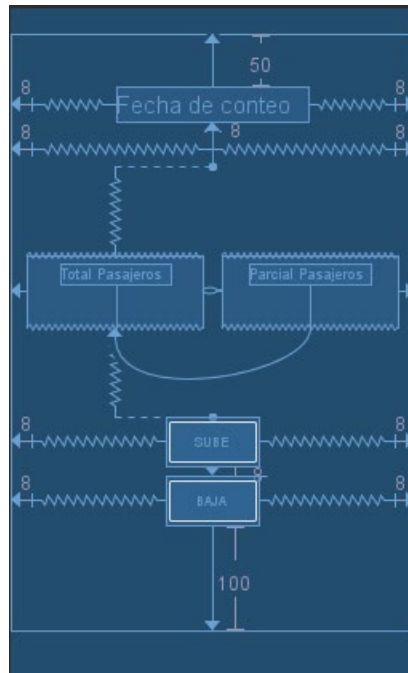


Ilustración 111 - Pantalla de conteo simulado

El *Layout* de la Ilustración 111 presenta dos bloques principales, las dos vistas rectangulares en forma de tarjetas que muestran la cantidad de pasajeros parciales junto con los pasajeros totales del conteo realizado en un día y por otro lado, los dos botones que simulan ascenso y descenso del vehículo. Se observa también en la parte superior, una vista de texto que muestra la fecha actual del conteo.

3.6.3.8.2. Clase Kotlin del conteo simulado

Lo primero que se obtiene es la cuenta parcial y total de abordajes del día, la función encargada de recuperar ambos valores es la mostrada en la Ilustración 112

```

val mUser : FirebaseUser? = DBConnection.mAuth.currentUser
val mUserReference : DatabaseReference? = mDatabaseReferenceCounter?.child( pathString: mUser!!.uid + "/" + currentDate)
mUserReference?.addValueEventListener(object : ValueEventListener {
    override fun onDataChange(snapshot: DataSnapshot) {
        totalCount = snapshot.value( path: "totalCount").toString().toInt()
        parcialCount = snapshot.value( path: "parcialCount").toString().toInt()
        Log.d( tag: "VARIABLET", totalCount.toString())
    }

    override fun onCancelled(databaseError: DatabaseError) {}
})
    
```

Ilustración 112 - Función para obtener valores de conteo total y parcial

Cuando se simula el evento de descenso, la función que se llama es *writeDown*, indicada en la Ilustración 113, en ella se crea una variable del tipo *Boardings*, la cual comparte los mismos atributos de la clase *Approach* de la AWGI. Seguidamente y con ayuda de las referencias de

las instancias de base de datos, se hacen dos inserciones, la primera, corresponde al número actualizado de cuenta parcial y la segunda es el objeto completo de registro del evento.

```

fun writeDown() {
    currentHour = Date().formatHour()
    var downCount = Boardings(currentHour, approachType: false, parcialCount: parcialCount - 1, totalCount, latitude, longitude)
    val mUser : FirebaseUser? = DBConnection.mAuth.currentUser
    val mUserReference : DatabaseReference = mDatabaseReferenceCounter!!.child( pathString: mUser!!.uid + "/" + currentDate + "/boardings")
    val mCountReference : DatabaseReference = mDatabaseReferenceCounter!!.child( pathString: mUser.uid + "/" + currentDate)
    val tempCount : DatabaseReference = mCountReference.child( pathString: "parcialCount")
    tempCount.setValue(parcialCount - 1)
    mUserReference.push().setValue(downCount)
}

```

Ilustración 113 - Función para escribir en la base de datos sobre un evento de descenso del vehículo

El evento de ascenso tiene ciertas particularidades que le añaden funcionalidad con respecto al descenso, la primera de ellas corresponde a la alerta por sobrecupo, que se activa en forma preventiva cuando el nivel máximo de pasajeros (característica presente en el momento de crear un bus en la AWGI) alcanza su valor, activando un sonido preventivo, de ahí en adelante cada que ingrese un usuario será disparada una alarma más aguda para indicar que se está por encima de los límites permitidos. A diferencia del evento de descenso, aquí se generan 4 inserciones, la primera corresponde al valor parcial actualizado de pasajeros, el segundo es el valor total de abordajes del día, el tercero corresponde al total de abordajes, de todo el sistema de vehículos y la cuarta es el objeto completo del registro del evento.

```

fun writeUp() {
    currentHour = Date().formatHour()
    if(parcialCount + 1 == maxCapacity) {
        var player : MediaPlayer! = MediaPlayer.create(applicationContext, R.raw.alerprev)
        player.start()
    }

    if(parcialCount + 1 > maxCapacity) {
        var player2 : MediaPlayer! = MediaPlayer.create(applicationContext, R.raw.alert)
        player2.start()
    }

    var upCount = Boardings(currentHour, approachType: true, parcialCount: parcialCount + 1, totalCount: totalCount + 1, latitude, longitude)
    val mUser : FirebaseUser? = DBConnection.mAuth.currentUser
    val mUserReference : DatabaseReference = mDatabaseReferenceCounter!!.child( pathString: mUser!!.uid + "/" + currentDate + "/boardings")
    val mCountReference : DatabaseReference = mDatabaseReferenceCounter!!.child( pathString: mUser.uid + "/" + currentDate)
    val mCountReferenceTotalPasseggers : DatabaseReference = mDatabaseReferenceCounter!!.child(currentDate)
    val tempCountParcial : DatabaseReference = mCountReference.child( pathString: "parcialCount")
    val tempCountTotal : DatabaseReference = mCountReference.child( pathString: "totalCount")
    val tempCountTotalPasseggers : DatabaseReference = mCountReferenceTotalPasseggers.child( pathString: "total")
    tempCountParcial.setValue(parcialCount + 1)
    tempCountTotalPasseggers.setValue(totalCountAllPasseggers + 1)
    tempCountTotal.setValue(totalCount + 1)
    mUserReference.push().setValue(upCount)
}

```

Ilustración 114 - Función para escribir en la base de datos sobre un evento de ascenso al vehículo

Dado que una de las características del sistema es la representación en mapa de calor de los eventos de los pasajeros, es necesario contar con un servicio que indique la geolocalización del dispositivo para que las coordenadas de latitud y longitud sean registradas en el objeto de conteo. Para ello, se usa un cliente de tipo *LocationServices*, el cual se inicializa con unas

opciones por defecto, entre las cuales está el tiempo de repetición de la búsqueda de localización, que para este caso en particular se usa de 5 segundos, Ilustración 115.

```
// La variable locationRequest tiene los parámetros de localización de prioridad
// y de tiempo de repetición de la captura de lat y lng
fun createLocationRequest() {
    locationRequest = LocationRequest.create()?.apply { this: LocationRequest
        interval = 5000
        fastestInterval = 5000
        priority = LocationRequest.PRIORITY_HIGH_ACCURACY
    }!!
}
```

Ilustración 115 - Creación de un request para Localización

La función *onLocationResult*, Ilustración 116, se ejecuta cada 5 segundos y actualiza el valor de latitud y longitud que se ingresa en el objeto de conteo, de tal forma que cada abordaje tiene las coordenadas actualizadas en su inserción en base de datos.

```
// Se inicializa la instancia de FusedLocationProviderClient
fusedLocationClient = LocationServices.getFusedLocationProviderClient(this)
// Este callback se ejecuta cada 5 segundos y reasigna las variables
// latitud y longitud para insertar en base de datos
locationCallback = object : LocationCallback() {
    override fun onLocationResult(locationResult: LocationResult?) {
        locationResult?.return
        for (location: Location! in locationResult.locations) {
            longitude = location.longitude
            latitude = location.latitude
            // toast(location.latitude.toString() + location.longitude.toString())
        }
    }
}
```

Ilustración 116 - Callback para la localización

3.7. Prototipo 2

El segundo prototipo desarrollado en el presente trabajo es el que busca implementar la tecnología seleccionada para el conteo de pasajeros, con lo cual, se centró en realizar la función del reconocimiento y conteo del flujo de pasajeros, esta tarea se desarrolló utilizando los diferentes periféricos con los que cuenta el DMC, desarrollando así una aplicación móvil que por medio de la cámara y el análisis de imágenes, estuviera en la capacidad de reconocer el evento de ascenso o descenso de una persona.

A continuación, se profundizará en las librerías, recursos y métodos utilizados para lograr alcanzar los objetivos del presente prototipo.

3.7.1. OpenCV

OpenCV por sus siglas en inglés *Open Source Computer Vision*, es una librería de código abierto desarrollada por Intel, especializada en visión artificial y *machine learning*. Cuenta con más de 2500 algoritmos, los cuales permiten identificar objetos, caras, clasificar acciones humanas en video, hacer seguimiento de objetos, extraer modelos 3D, entre otras funcionalidades.

OpenCV está escrito en C++, cuenta con interfaces en C++, C, Python, Java y MATLAB interfaces. Debido a que cuenta con una licencia BSD, su uso es gratuito de forma que su código puede ser modificado o emplearse de forma regular.

3.7.2. Algoritmo de detección

Con el fin de realizar el conteo de pasajeros que ascienden y descienden de un bus específico, es fundamental la correcta detección del evento deseado, para esto, la principal tarea, y la que requiere más atención es la precisa identificación en la forma del objeto que se desea, en este caso el del rostro de una persona y el movimiento que este realiza, para esto existen varios métodos o algoritmos.

Uno de los algoritmos de detección de objetos más ampliamente utilizado, debido a su licencia tipo BSD y sus los beneficios que brindan sus algoritmos propios de detección, reconocimiento y *tracking*, es el propuesto por P. Viola y M. Jones [71]. Este algoritmo, basado en etapas de clasificación de complejidad creciente, es considerado como uno de los mejores en cuanto a tiempo de ejecución y efectividad de detección [72] [73], por lo cual es el que se implementara dentro del presente trabajo de grado.

La detección de un objeto, y más específicamente de un rostro en una imagen puede ser una tarea de alto costo computacional en función del algoritmo que se utilice. Una forma de implementar un algoritmo de detección de objetos consiste en ir recorriendo la imagen mediante una ventana de un determinado tamaño, la cual puede contener un objeto (candidato). Un aspecto a considerar es la forma de evaluar si la ventana contiene o no el objeto. Después de recorrer la imagen, la ventana puede ser escalada y repetir el proceso con el objetivo de detectar posibles objetos de mayor tamaño que no fueron detectados con tamaños de ventana inferiores [74].

El algoritmo de detección de rostros propuesto por P. Viola y M. Jones, basado en la utilización de ventanas deslizantes, es ampliamente utilizado, no sólo en tareas de detección de rostros sino también de detección de diversos objetos en general.

Las características fundamentales en las cuales se basa este algoritmo son:

- Utilización de rasgos de clasificación.
- Utilización de una imagen integral.
- Organización en cascada de los clasificadores.

3.7.2.1. Rasgos de clasificación

Los rasgos de clasificación son las formas geométricas utilizadas por el algoritmo para detectar zonas de una imagen (o ventana) que pueda contener partes de un rostro. Los rasgos de clasificación utilizados por el algoritmo Viola-Jones son estructuras simples compuestas por dos, tres o cuatro rectángulos grises y blancos, como los que se pueden observar en la Ilustración 117, cabe resaltar que los rasgos pueden tener 6, 8 o 9 puntos significativos, correspondientes a las esquinas de cada rectángulo.

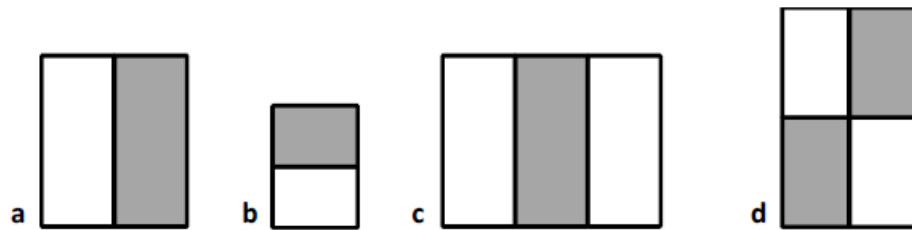


Ilustración 117 - Ejemplos de rasgos de 2 (a y b), 3 (c) y 4 (d) rectángulos utilizados para realizar la detección de objetos.

Tomado de: [75]

Estas estructuras simples pueden ser asociadas a partes comunes de un rostro, como las correspondientes a los ojos, la nariz, la frente, el pelo, etc. En la Ilustración 118 se observa la relación de dos posibles rasgos de clasificación con partes de un rostro.

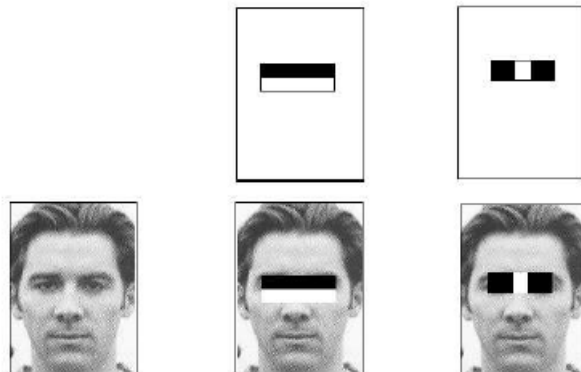


Ilustración 118 - dos rasgos de clasificación y su correspondencia con partes de un rostro.

Tomado de: [75]

Como el número de rasgos que se pueden corresponder con partes de un rostro en una imagen de un determinado tamaño es considerablemente elevado, Viola y Jones utilizaron un algoritmo de entrenamiento conocido como AdaBoost [76], el cual se aplica sobre un gran conjunto de imágenes con y sin rostros, para seleccionar aquellos rasgos que mejor distinguen las zonas de una imagen que contienen un rostro [71].

Es importante resaltar, que los rasgos no sólo se caracterizan por su forma, sino también por su tamaño y posición dentro de una ventana, así como por la posible contribución de los mismos a la detección de un rostro u objeto. Para ello es necesario calcular el valor del rasgo, parámetro que se obtiene de la diferencia entre las intensidades de los puntos de las zonas blancas y grises de un rasgo. Si el valor de un rasgo sobrepasa un determinado umbral (umbral de clasificación), se considera que contribuye con un determinado valor *alpha* a la detección de un rostro. Todo este conjunto de parámetros (y otros que se expondrán a continuación) forman parte de lo que se conoce como un clasificador [71].

Es importante aclarar que el cálculo del valor del rasgo puede ser un proceso computacionalmente costoso en tiempo si fuese preciso recorrer todos los puntos del rasgo para su evaluación, por esto, la segunda característica del algoritmo de Viola-Jones reduce este problema considerablemente.

3.7.2.2. Imagen integral

Una contribución fundamental del algoritmo de Viola-Jones consiste en acelerar el cálculo del valor del rasgo al trabajar no con la imagen original sino con una imagen integral, esta no es más que una transformación de la imagen original en donde cada punto de la misma toma el valor de la suma de todos los puntos que están ubicados por encima y a su izquierda. Así, se puede definir una imagen integral ii según la Ecuación 1:

$$ii(x, y) = \sum_{x' < x, y' < y} i(x', y')$$

Ecuación 1 – Representación integral de una imagen.

En donde $ii(x, y)$ es el punto de la imagen integral en las coordenadas (x, y) e $i(x, y)$ es el punto de la imagen original en las mismas coordenadas. En la Ilustración 119 en parte (a) se observa una imagen original, mientras que la parte (b) la imagen integral correspondiente.

La imagen integral se puede calcular en un solo recorrido de la imagen original realizando las operaciones descritas en las Ecuación 2 y Ecuación 3:

$$s(x, y) = s(x, y - 1) + i(x, y)$$

Ecuación 2 - Recorrido de una imagen 1

$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

Ecuación 3 - Recorrido de una imagen 2

En donde $s(x, y)$ es la suma de toda una fila considerando que:

$$s(x, -1) = 0, ii(-1, y) = 0.$$

Ecuación 4 - Condiciones para la suma de una fila

A partir de la imagen integral se puede calcular rápidamente la suma de todos los puntos contenidos en un rectángulo cualquiera de la imagen utilizando sólo los cuatro valores asociados a sus esquinas, tal como se ilustra en la Ilustración 119 parte (c). Esta característica permite que el cálculo de la suma de los puntos contenidos en un rectángulo de tamaño arbitrario pueda ser realizado en un tiempo constante utilizando sólo cuatro operaciones, por lo cual, el cálculo del valor de un rasgo se reduce considerablemente.

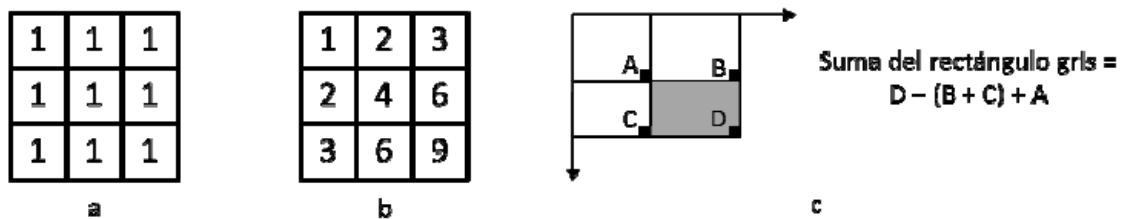


Ilustración 119 - Cálculo de la imagen integral. (a) Imagen original (b) Imagen integral (c) Valor del rectángulo utilizando la imagen.

Tomado de: [74]

A partir de la imagen integral es simple la determinación del valor de un rasgo, conociendo los valores de la imagen integral en los puntos significativos de un rasgo y las características del rasgo, la determinación del valor de un rasgo se reduce a simples operaciones de multiplicación (por coeficientes constantes) y suma. La Ilustración 120 se puede observar este proceso, en donde se muestra un rasgo formado por dos rectángulos y se detalla la determinación de su valor.

Nótese que los coeficientes por los cuales es necesario multiplicar los valores de la imagen integral de los seis puntos significativos de este rasgo son constantes. Al conjunto de estas constantes (6, 8 o 9 valores según el rasgo) se denomina vector de pesos del rasgo (W) y es otro de los parámetros que forman parte de un clasificador.

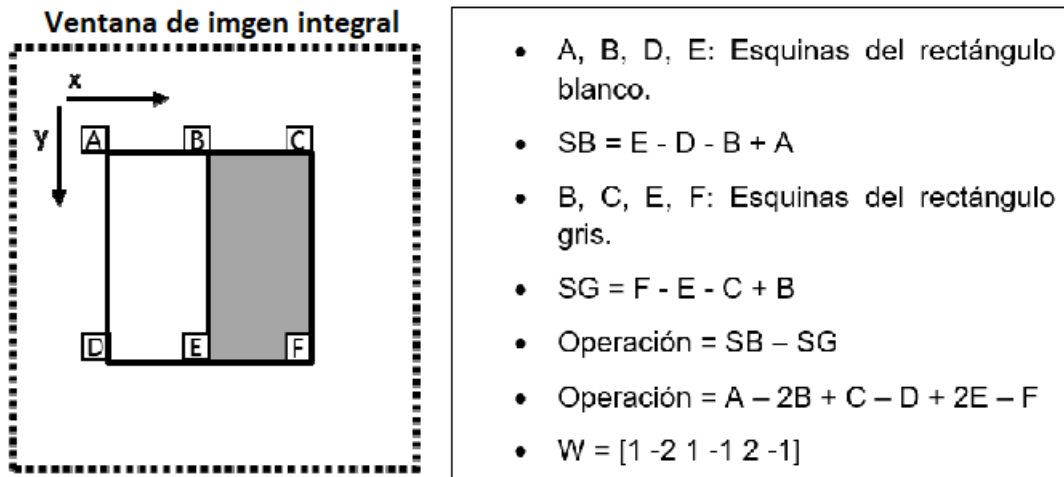


Ilustración 120 - cálculo del valor de un rasgo y del vector W de pesos del rasgo.

Adaptado de: [74]

3.7.2.3. Organización en cascada de los clasificadores

La tercera característica del algoritmo Viola-Jones es la utilización de una combinación en cascada de grupos de clasificadores cada vez más complejos (con mayor número de rasgos). Las primeras etapas de la cascada poseen pocos clasificadores, pero permiten descartar rápidamente aquellas ventanas que no contienen rostros o el objeto deseado, concentrando el esfuerzo computacional en las etapas siguientes en aquellas con mayor probabilidad de contener el objeto deseado.

En la Ilustración 121 se observa una posible distribución de clasificadores en una cascada de cuatro etapas. La cantidad de etapas y de clasificadores en cada etapa dependen de la base de datos de clasificadores utilizada para el proceso de detección de rostros [77].

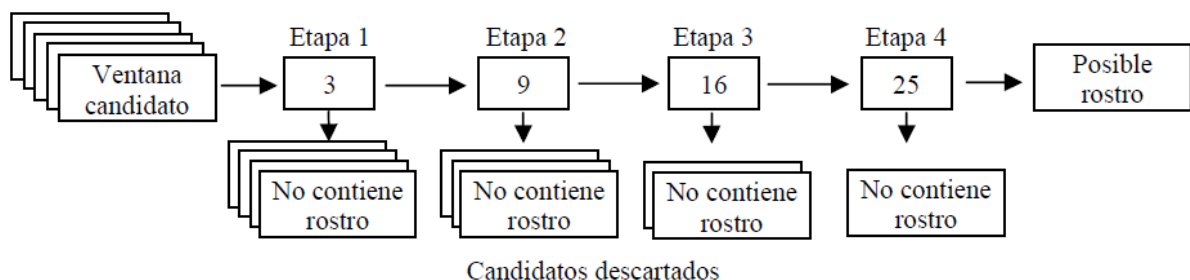


Ilustración 121 - Cascada de clasificadores.

En la primera etapa, para cada una de las ventanas de la imagen se calculan sus tres clasificadores y se suman los aportes α de los mismos. Si este valor es inferior al umbral

de la etapa, se descarta que la ventana contenga un rostro y se procesa la siguiente. Si es superior, la ventana se procesa con los nueve clasificadores de la segunda etapa, y así sucesivamente hasta que la ventana sea descartada en alguna de las etapas o, si las supera todas, sea identificada como un posible rostro. Los valores del umbral de cada etapa se obtienen también del proceso de entrenamiento [74].

Una vez expuestos los aspectos fundamentales en que se basa el algoritmo de detección de rostros de Viola-Jones se puede resumir el mismo. Se comienza por procesar una ventana evaluando los clasificadores de las etapas. Si la ventana no es descartada en alguna de las etapas, se marca como un posible rostro. El proceso continúa desplazando la ventana (en sentido horizontal o vertical) según un *factor de escaneo* determinado. Una vez que se haya recorrido toda la imagen con un tamaño de ventana determinado, se incrementa el tamaño de la ventana (según un *factor de escalado* determinado) y se repite el procesamiento de las etapas y de barrido de la imagen. Los valores de los factores de escaneo y de escalado son obtenidos también durante el proceso de entrenamiento [71].

3.7.2.4. Entrenador Clasificador Haar-cascade

La librería OpenCV cuenta con varios clasificadores en cascada ya entrenados, los cuales sirven para la detección de rostros, nariz, boca, o algunos objetos específicos, sin embargo, es limitada la lista de estos, por lo cual, la librería también cuenta con la funcionalidad de entrenar un clasificador propio, el cual detecte específicamente para lo que fue entrenado.

Dicho lo anterior, para el presente trabajo de grado donde el evento a detectar son personas que ascienden o descienden de un bus, se probaron algunos de clasificadores ya entrenados (orejas, rostro, nariz), con los cuales, debido a la posición en la que ingresan o salen las personas, no se logra una correcta detección, el clasificador de perfil fue la opción más certera, sin embargo, no lo suficiente, debido a que solo detecta rostros completamente de perfil sin ningún tipo de inclinación ni ángulo.

En consecuencia, se optó por entrenar un clasificador que a diferencia del anterior no solo detectara el rostro completamente de perfil, si no también, variaciones de este, dadas por las posiciones en la que ingresa una persona a un bus.

Para el entrenamiento del clasificador, dentro de la documentación de OpenCV se encuentra el procedimiento a seguir [78], para el cual se deben disponer de múltiples imágenes en las cuales se encuentre presente de forma clara el objeto que se desea detectar (positivas), además de múltiples imágenes en las que este no se encontraba (negativas).

Para el entrenamiento del clasificador utilizado en el presente trabajo de grado, se utilizaron un total de 514 imágenes positivas, las cuales se obtuvieron de diferente sitio web que

brindaban imágenes con personas de perfil en diferentes ángulos e inclinaciones como se observa en la Ilustración 122, además, se obtuvieron múltiples imágenes de personas que decidieron colaborar con el proyecto permitiendo capturar en fotos su perfil con las características requeridas.



Ilustración 122- Imagen positiva entrenamiento de clasificador
Tomado de: [79]

Para las imágenes negativas, se utilizó el banco de imágenes negativas ofrecidas por OpenCV para el entrenamiento de clasificadores, el cual cuenta con 1500 imágenes negativas en las cuales se encuentran diferentes espacios vacíos de personas como se puede observar en la Ilustración 123.



Ilustración 123 - Imagen negativa entrenamiento de clasificador
Tomado de: [78]

Posteriormente a la recolección de imágenes tanto positivas como negativas, se realiza el entrenar el clasificador deseado siguiendo el proceso descrito en la documentación de OpenCV, haciendo uso de los ejecutables `opencv_createsamples.exe` y `opencv_traincascade.exe`, ofrecidos por la librería, los cuales permiten realizar el trabajo de entrenamiento de manera sencilla, el primero prepara el conjunto de imágenes positivas y el otro genera el clasificador en cascada.

3.7.3. Preparación del entorno de desarrollo

Es necesario preparar el entorno de desarrollo para poder desarrollar una aplicación con la librería de OpenCV. Por ende, se debe realizar la compilación del SDK de OpenCV para Android, posteriormente agregar el resultado de la compilación al proyecto en forma de módulo y finalmente, instalar una aplicación generada por el SDK en el dispositivo en el cual se va a implementar la aplicación. Todo el proceso está detallado en el Anexo G.

3.7.4. Layout de la pantalla de conteo

El diseño de la pantalla de conteo solo implementa un elemento en su *layout*, el cual corresponde a una clase de tipo puente entre OpenCV y la *Java Camera*¹⁶, Ilustración 124.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".OpenCVActivity">

    <org.opencv.android.JavaCameraView
        android:id="@+id/fd_activity_surface_view"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Ilustración 124 - Layout de la actividad de captura

3.7.5. Clase Java de la pantalla de conteo

¹⁶ Instancia de Java para acceder a la cámara

La declaración de la clase que implementa el conteo de pasajeros extiende la clase *Activity* e implementa la interfaz *CameraBridgeViewBase.CvCameraViewListener2*, Ilustración 125, dicha interfaz, implementa las funciones *onCameraFrame*, *onCameraViewStarted*, *onCameraViewStopped*, las cuales son estudiadas más adelante.

```
public class OpenCvController extends Activity implements CameraBridgeViewBase.CvCameraViewListener2
```

Ilustración 125 - Declaración de clase *OpenCvController*

Lo primero que la actividad hace es revisar que exista el *OpenCV Manager* en el dispositivo, e intentar cargar el archivo *Haar Cascade*. La clase *BaseLoaderCallback*, Ilustración 126 e Ilustración 127, con su función *onManagerConnected*, verifica la existencia del *OpenCV Manager*.

Cuando se verifica la existencia de la aplicación en el dispositivo de prueba, se realiza el cargue del archivo Haar, en un bloque *try, catch*. En él, lo primero que se hace es asignar a las variables de tipo *file* e *inputstream*, los recursos disponibles en la carpeta *raw*, para luego, cargar la variable *mJavaDetector* (del tipo *CascadeClassifier*). Si luego del proceso descrito, *mJavaDetector* es vacío, indica un error en la carga del archivo, de lo contrario se sigue con el flujo.

Finalmente, por medio de la interfaz *mOpenCvCameraView*, se activa el parámetro de visualización de FPS y se asigna a 0 o 1 el index de la cámara, donde 0 corresponde a la cámara trasera del dispositivo y 1 corresponde a la cámara frontal del mismo.

```
private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback( AppContext: this) {
    @Override
    public void onManagerConnected(int status) {
        switch (status) {
            case LoaderCallbackInterface.SUCCESS: {
                Log.i(TAG, msg: "OpenCV loaded successfully");

                try {
                    // Carga de Haarcascade Profileface haarcascade_profileface.xml
                    InputStream is = getResources().openRawResource(R.raw.haarcascade_profileface);
                    File cascadeDir = getDir( name: "cascade", Context.MODE_PRIVATE);
                    mCascadeFile = new File(cascadeDir, child: "haarcascade_profileface.xml");
                    FileOutputStream os = new FileOutputStream(mCascadeFile);

                    byte[] buffer = new byte[1024];
                    int bytesRead;
                    while ((bytesRead = is.read(buffer)) != -1) {
                        os.write(buffer, off: 0, bytesRead);
                    }
                    is.close();
                    os.close();

                    mJavaDetector = new CascadeClassifier(mCascadeFile.getAbsolutePath());
                    if (mJavaDetector.empty()) {
                        Log.e(TAG, msg: "Failed to load cascade classifier");
                        mJavaDetector = null;
                    } else
                        Log.i(TAG, msg: "Loaded cascade classifier from " + mCascadeFile.getAbsolutePath());
                    cascadeDir.delete();
                }
            }
        }
    }
}
```

Ilustración 126 - Clase *BaseLoaderCallback* 1/2

```

        cascadeDir.delete();
    } catch (IOException e) {
        e.printStackTrace();
        Log.e(TAG, "Failed to load cascade. Exception thrown: " + e);
    }
    mOpenCvCameraView.enableFpsMeter();
    mOpenCvCameraView.setCameraIndex(1);
    mOpenCvCameraView.enableView();
}
break;
default: {
    super.onManagerConnected(status);
}
break;
}
}
};

```

Ilustración 127 - Clase BaseLoaderCallback 2/2

Dado que se hace uso de una característica hardware como lo es la cámara, es necesario agregar dicho permiso en el archivo *manifest*, Ilustración 128.

```

<uses-permission android:name="android.permission.CAMERA"/>

<uses-feature android:name="android.hardware.camera" android:required="false"/>
<uses-feature android:name="android.hardware.camera.flash" android:required="false"/>
<uses-feature android:name="android.hardware.camera.autofocus" android:required="false"/>
<uses-feature android:name="android.hardware.camera.front" android:required="false"/>
<uses-feature android:name="android.hardware.camera.front.autofocus" android:required="false"/>

```

Ilustración 128 - Permisos de uso de cámara archivo manifest

En la función *onCreate*¹⁷, se asigna la instancia de cámara a *mOpenCvCameraView*, Ilustración 129.

```

setContentView(R.layout.activity_open_cv);

mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.fd_activity_surface_view);
mOpenCvCameraView.setCvCameraViewListener(this);

```

Ilustración 129 - Fragmento onCreate actividad de conteo

Luego, en la función *onCameraViewStarted*, se inicializan las variables de tipo *Mat*, que corresponden a una clase de matriz de n-dimensiones, las cuales se utilizan en el script principal de detección, Ilustración 130.

¹⁷ Método llamado al crear una actividad.

```
public void onCameraViewStarted(int width, int height) {  
    widthResolution = width;  
    heightResolution = height;  
    limitZones = widthResolution / 8;  
    mGray = new Mat();  
    mRgba = new Mat();  
    mRgbaTest = new Mat();  
    mRgba2 = new Mat();  
    mGray2 = new Mat();  
}
```

Ilustración 130 - Función *onCameraViewStarted*

3.7.5.1. Asignación de tipo de frames a variables de clase *Mat*

El script principal de detección se ejecuta en la función *onCameraFrame*, la cual es llamada cada que un *frame*¹⁸ es detectado por la instancia de la cámara. Lo primero que se hace es definir el valor de las variables de tipo *Mat*, con respecto a cada imagen que entra en la función, Ilustración 131.

```
mRgba = inputFrame.rgba();  
mGray = inputFrame.gray();  
Core.flip(mGray, mGray2, flipCode: 1);  
Core.flip(mRgba, mRgba2, flipCode: 1);
```

Ilustración 131 - Definiendo valores de variables del tipo *Mat*

Se puede observar que a *mRgba* se le asigna al valor de vector del tipo *rgba* de la captura de la cámara, mientras que *mGray*, se asigna a un array de tipo escala de grises. Por otro lado, la función *flip* del *core* de OpenCV asigna a la variable *mGray2* la misma escala de grises de la variable *mGray*, pero invertida, es decir con efecto espejo, al igual que sucede con la variable *mRgba2* y *mRgba*, las cuales comparten escala *rgba* pero con un efecto espejo en sus capturas.

3.7.5.2. Dibujo de líneas delimitadoras de zonas

OpenCV brinda la posibilidad de dibujar en la pantalla que corresponda a la instancia definida en el Layout. En este caso se dibujan las zonas para el estudio del seguimiento de la detección, Ilustración 132. La variable *limitZones*, corresponde al cálculo del tamaño completo de la pantalla dividido entre 8, y la disposición de las 7 zonas junto con los límites de decisión se observa en la Ilustración 133.

¹⁸ Imágenes instantáneas de la captura de la cámara

```

int x1 = (limitZones * 1) / 2;
int y1 = heightResolution;
int x2 = (limitZones * 5) / 2;
int y2 = heightResolution;

// Se dibujan las zonas de decisión
Imgproc.line(mRgba, new Point( * limitZones * 1, y: 0), new Point( * limitZones * 1, y1), new Scalar(255, 255, 0), thickness: 1);
Imgproc.line(mRgba, new Point( * limitZones * 2, y: 0), new Point( * limitZones * 2, y1), new Scalar(255, 255, 0), thickness: 1);
Imgproc.line(mRgba, new Point( * limitZones * 3, y: 0), new Point( * limitZones * 3, y1), new Scalar(255, 255, 0), thickness: 1);
Imgproc.line(mRgba, new Point( * limitZones * 4, y: 0), new Point( * limitZones * 4, y1), new Scalar(255, 255, 0), thickness: 1);
Imgproc.line(mRgba, new Point( * limitZones * 5, y: 0), new Point( * limitZones * 5, y1), new Scalar(255, 255, 0), thickness: 1);
Imgproc.line(mRgba, new Point( * limitZones * 6, y: 0), new Point( * limitZones * 6, y1), new Scalar(255, 255, 0), thickness: 1);
Imgproc.line(mRgba, new Point( * limitZones * 7, y: 0), new Point( * limitZones * 7, y1), new Scalar(255, 255, 0), thickness: 1);
// Se dibujan los dos limites de conteo
Imgproc.line(mRgba, new Point(x1, y: 0), new Point(x1, y1), new Scalar(255, 0, 0), thickness: 3);
Imgproc.line(mRgba, new Point(x2, y: 0), new Point(x2, y2), new Scalar(0, 255, 0), thickness: 3);
    
```

Ilustración 132 - Dibujo de zonas de decisión

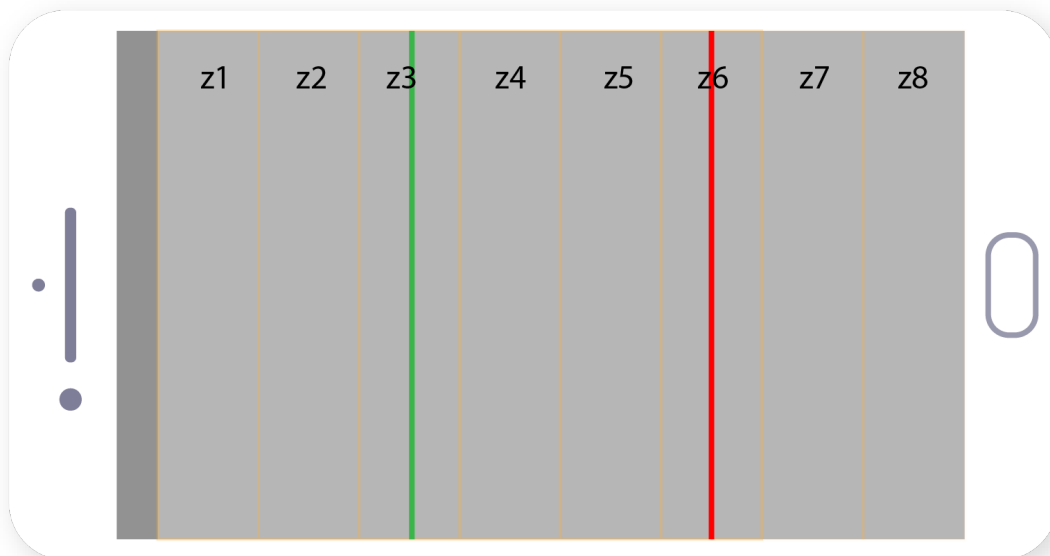


Ilustración 133 - Disposición de zonas para seguimiento

3.7.5.3. Dibujo de variables que indican estado de zonas

Para tener un control de la las zonas dibujadas en la sección anterior, es necesario imprimir en pantalla el estado de cada una de ellas, donde el cero representa que no ha sido activada, es decir que el objeto no ha pasado por ahí y el 1 indica el paso del objeto en seguimiento por dicha zona, Ilustración 134. Las variables que representan cada zona son *zone1-zone8*.

```
// =====CONTADORES DE ZONAS =====//
Imgproc.putText(mRgba, text: "Zona1: " + zone1,new Point( x 20, y 170),Core.FONT_HERSHEY_DUPLEX, fontScale: 1, new Scalar(0, 0, 255,255));
Imgproc.putText(mRgba, text: "Zona2: " + zone2,new Point( x 20, y 210),Core.FONT_HERSHEY_DUPLEX, fontScale: 1, new Scalar(0, 0, 255,255));
Imgproc.putText(mRgba, text: "Zona3: " + zone3,new Point( x 20, y 250),Core.FONT_HERSHEY_DUPLEX, fontScale: 1, new Scalar(0, 0, 255,255));
Imgproc.putText(mRgba, text: "Zona4: " + zone4,new Point( x 20, y 290),Core.FONT_HERSHEY_DUPLEX, fontScale: 1, new Scalar(0, 0, 255,255));
Imgproc.putText(mRgba, text: "Zona5: " + zone5,new Point( x 20, y 330),Core.FONT_HERSHEY_DUPLEX, fontScale: 1, new Scalar(0, 0, 255,255));
Imgproc.putText(mRgba, text: "Zona6: " + zone6,new Point( x 20, y 370),Core.FONT_HERSHEY_DUPLEX, fontScale: 1, new Scalar(0, 0, 255,255));
Imgproc.putText(mRgba, text: "Zona7: " + zone7,new Point( x 20, y 410),Core.FONT_HERSHEY_DUPLEX, fontScale: 1, new Scalar(0, 0, 255,255));
Imgproc.putText(mRgba, text: "Zona8: " + zone8,new Point( x 20, y 450),Core.FONT_HERSHEY_DUPLEX, fontScale: 1, new Scalar(0, 0, 255,255));
Imgproc.putText(mRgba, text: "WIDTH: " + widthRec, new Point( x 20, y 490),Core.FONT_HERSHEY_DUPLEX, fontScale: 1, new Scalar(0, 0, 255,255));
Imgproc.putText(mRgba, text: "REFRESH: " + counterRefresh,new Point( x 20, y 530),Core.FONT_HERSHEY_DUPLEX, fontScale: 1, new Scalar(0, 0, 255,255));
//===== FIN DE LA SECCIÓN DBT =====//
```

Ilustración 134 – Dibujo de contadores de zonas

3.7.5.4. Detección

Le sección de detección permite agregar componentes a un array de coincidencias con respecto al Haar que se cargó en el principio del código.

Primero se crean dos variables del tipo *MatOfRect*. El tipo *Mat* almacena el *frame* detectado, mientras que el tipo *MatOfRect* contiene todos los rectángulos captados por el detector.

La eficiencia de la aplicación se ve radicalmente comprometida con la agregación de más de un detector del tipo Haar, es por eso que se trabaja con aquella Cascada que reconoce el perfil izquierdo de la cabeza de una persona. Lo anterior explica el uso de dos variables del tipo *Mat* que tienen asignadas las capturas inversas del *frame*, por ende, la detección en realidad se hace solamente del perfil izquierdo, pero cuando esta se hace con la variable en espejo, el procesamiento corresponde a una detección del perfil derecho.

Así pues, luego de declarar las variables que van a contener los rectángulos de detección, se elaboran dos bloques de código para la detección izquierda y derecha, en donde el único parámetro que cambia es *mGray* y *mGray2*. Cuando se produce una detección en el perfil izquierdo, se agrega una componente al vector *faces*, mientras que cuando se produce una detección del perfil derecho se agrega la componente al vector *faces2*.

Finalmente, se declaran dos variables del tipo *Rect*, las cuales almacenan los vectores en las variables *faces* y *faces2* por separado. Dicho tipo de variable guarda las coordenadas de detección en un tipo de arreglo 2D. Todo el proceso de detección descrito en esta sección se muestra en la Ilustración 135.


```

MatOfRect faces = new MatOfRect();
MatOfRect faces2 = new MatOfRect();

// ===== SECCIÓN DT (detección) =====//

// SE DETECTA EL PERFIL IZQUIERDO POR MEDIO DEL INPUTFRAME mGray
if (mDetectorType == JAVA_DETECTOR) {
    if (mJavaDetector != null)
        Log.e("detector1", msg: "Entra a detector1");
        mJavaDetector.detectMultiScale(mGray, faces, scaleFactor: 1.1, minNeighbors: 2, flags: 2,
            new Size(mAbsoluteFaceSize, mAbsoluteFaceSize), new Size());
    } else {
        Log.e(TAG, msg: "Detection method is not selected!");
    }

// SE DETECTA EL PERFIL DERECHO POR MEDIO DEL INPUTFRAME mGray2
if (mDetectorType == JAVA_DETECTOR) {
    if (mJavaDetector != null)
        mJavaDetector.detectMultiScale(mGray2, faces2, scaleFactor: 1.1, minNeighbors: 2, flags: 2,
            new Size(mAbsoluteFaceSize, mAbsoluteFaceSize), new Size());
    } else {
        Log.e(TAG, msg: "Detection method is not selected!");
    }

//===== FIN DE LA SECCIÓN DT =====//

// SE ALMACENAN LAS DETECCIÓNES EN UN ARRAY DE TIPO Rect[]
Rect[] facesArray = faces.toArray(); // IZQUIERDO
Rect[] facesArray2 = faces2.toArray(); // DERECHO

```

Ilustración 135 - Sección de detección

3.7.5.5. Seguimiento

Una vez realizada la detección, es necesario realizar el seguimiento del objeto detectado, lo cual incluye tener en cuenta la mitigación de posible ruido y eventos que no corresponden al evento real. Es necesario implementar una clase que contenga los parámetros de componente horizontal y componente vertical, Ilustración 136.

```

class PersonCoordinate {
    private int horizontal;
    private int vertical;

    public PersonCoordinate()
    {
        horizontal = 0;
        vertical = 0;
    }
    public PersonCoordinate(int x , int y)
    {
        horizontal = x;
        vertical = y;
    }

    public int getHorizontal() { return horizontal; }

    //Método para establecer la edad del animal
    public void setHorizontal(int setHorizontal) { horizontal = setHorizontal; }

    //Método para obtener el nombre del animal
    public int getVertical() { return vertical; }

    public void setVertical(int setVertical) { vertical = setVertical; }
}

```

Ilustración 136 - Clase PersonCoordinate

El seguimiento se realiza dentro de un for del tamaño del vector *facesArray* o *facesArray2*. Primero, se dibuja un rectángulo con las coordenadas de esquina superior izquierda (*tl*) y esquina inferior derecha (*br*), Ilustración 137.

```
//===== CICLO PARA EL PERFIL IZQUIERDO =====
for (int i = 0; i < facesArray.length; i++) {
    widthRec = facesArray[i].width;
    Imgproc.rectangle(mRgba, facesArray[i].tl(), facesArray[i].br(),
        FACE_RECT_COLOR, thickness: 3);
}
```

Ilustración 137 - Ciclo for para el seguimiento

Luego, se realiza el cálculo del punto central por medio de componentes en el eje vertical, horizontal y el tamaño del rectángulo detectado. Para el dibujo del punto central de la detección del perfil izquierdo se realiza un cálculo sin modificar valores, de la forma mostrada en la Ilustración 138, mientras que la detección del punto central para el perfil derecho debe ser modificada puesto que existe una inversión en la captura por la función *flip* aplicada, entonces, se debe hacer un cálculo del valor absoluto de dichos puntos para que el seguimiento de la imagen concuerde con el efecto en espejo, Ilustración 139.

```
xCenter = (facesArray[i].x + facesArray[i].width + facesArray[i].x) / 2;
yCenter = (facesArray[i].y + facesArray[i].y + facesArray[i].height) / 2;
Point center = new Point(xCenter, yCenter);

Imgproc.circle(mRgba, center, radius: 10, new Scalar(255, 0, 0, 255), thickness: 3);
```

Ilustración 138 - Dibujo de punto central para detección del perfil izquierdo

```
int xx = widthResolution - facesArray2[i].x;
double xbien = Math.abs(xx); // El valor absoluto permite el correcto seguimiento de los frames
Point tlAbs = new Point(Math.abs(widthResolution - facesArray2[i].tl().x), facesArray2[i].tl().y);
Point brAbs = new Point(Math.abs(widthResolution - facesArray2[i].br().x), facesArray2[i].br().y);
Imgproc.rectangle(mRgba, tlAbs, brAbs,
    FACE_RECT_COLOR, thickness: 3);
xCenter = xbien - (facesArray2[i].width) / 2;
yCenter = (facesArray2[i].y + facesArray2[i].y + facesArray2[i].height) / 2;
Point center = new Point(xCenter, yCenter);
```

Ilustración 139 - Dibujo del punto central para la detección del perfil derecho

A continuación, se crea una variable auxiliar del tipo *PersonCoordinate*, la cual almacena el componente horizontal y vertical de la detección actual, Ilustración 140.

```
// La variable myPersonCoordinate guarda las coordenadas X,Y por cada ciclo de detección
PersonCoordinate myPersonCoordinate = new PersonCoordinate();
myPersonCoordinate.setHorizontal(facesArray[i].x);
myPersonCoordinate.setVertical(facesArray[i].y);
```

Ilustración 140 - Variable auxiliar de seguimiento

La primera vez que se produce una detección, se hace una agregación al vector principal, llamado *PersonTestCoordinates*, que es del mismo tipo de la variable auxiliar de seguimiento, en el siguiente ciclo, se realiza la comparación del último componente ya cargado en el vector principal y el componente de la variable auxiliar que corresponde a la detección actual. Dicha comparación toma los componentes en el eje horizontal y vertical de las dos tomas. Es decir, los componentes vertical y horizontal del vector principal *PersonTestCoordinates[tamaño - 1].x*, *PersonTestCoordinates[tamaño - 1].y* respectivamente, se comparan con las componentes vertical y horizontal de la variable auxiliar *myPersonCoordinate.x* *myPersonCoordinate.y*, Ilustración 141.

```
int lastPosition = personTestCoordinates.size() - 1; // Se encuentra el último índice del array personTestCoordinates
int lastVertical = personTestCoordinates.get(lastPosition).getVertical(); // Último valor en la coordenada Y
int lastHorizontal = personTestCoordinates.get(lastPosition).getHorizontal(); // Último valor en la coordenada X

int actualVertical = myPersonCoordinate.getVertical(); // Valor en la coordenada X de la toma actual
int actualHorizontal = myPersonCoordinate.getHorizontal(); // Valor en la coordenada Y de la toma actual
```

Ilustración 141 - Componentes horizontales y verticales de variables a comparar

La comparación se hace por medio de una distancia definida en la variable *distance* mediante pruebas de eficiencia, en la cual ambos pares de componentes deben encontrarse a una separación mínima, Ilustración 142.

```
// Se verifica que la detección actual está cerca por lo menos a una distancia de la variable distance
int distance = 350;
if (actualVertical >= lastVertical - distance && actualVertical <= lastVertical + distance &&
    actualHorizontal >= lastHorizontal - distance && actualHorizontal <= lastHorizontal + distance) {
```

Ilustración 142 - Evaluación de distancias

Si la condición se cumple, entonces se evalúa en qué punto central está el rectángulo de detección y se activa la zona correspondiente, Ilustración 144, además se agrega la variable auxiliar a un nuevo componente del vector principal y se repite la iteración, donde la actual toma se convierte en el último componente a evaluar. De forma contraria, si no se cumple la condición se aumenta un contador por cada frame errado, de manera que, si dicha variable se aumenta en un valor de 5, el sistema interpreta que la detección fue de ruido, por lo que la iteración vuelve a comenzar y el vector principal es vaciado, Ilustración 143.

```
} else {
    counterRefresh++; // Si la detección no está cerca, un contador para refrescar aumenta
    if (counterRefresh > 5) {
        personTestCoordinates.clear();
        counterRefresh = 0;
        counterFrames = 0;
        clearZones();
    }
}
```

Ilustración 143 - Refrescar vector principal en caso de ruido

```

if (actualHorizontal > limitZones * 7) {
    zone8 = 1;
}
if (actualHorizontal > limitZones * 6 && actualHorizontal < limitZones * 7) {
    zone7 = 1;
}
if (actualHorizontal > limitZones * 5 && actualHorizontal < limitZones * 6) {
    zone6 = 1;
}
if (actualHorizontal > limitZones * 4 && actualHorizontal < limitZones * 5) {
    zone5 = 1;
}
if (actualHorizontal > limitZones * 3 && actualHorizontal < limitZones * 4) {
    zone4 = 1;
}
if (actualHorizontal > limitZones * 2 && actualHorizontal < limitZones * 3) {
    zone3 = 1;
}
if (actualHorizontal > limitZones && actualHorizontal < limitZones * 2) {
    zone2 = 1;
}
if (actualHorizontal < limitZones) {
    zone1 = 1;
}
counterFrames++; // Se aumenta el contador de frames una vez se determina la zona que se aumenta
personTestCoordinates.add(myPersonCoordinate); // Se agrega la coordenada al array personTestCoordinates

```

Ilustración 144 - Evaluación de zonas para el perfil izquierdo

Las zonas de decisión dependen del lado de la detección, entonces, la línea que se encuentra en la mitad de la zona 6 de la Ilustración 133 corresponde a la zona de evaluación para el perfil derecho, mientras que la línea que está en la mitad de la zona 3 de la misma ilustración corresponde a la zona de evaluación del perfil izquierdo. Cuando son activadas dichas zonas, se llaman las funciones de evaluar ascenso o descenso según corresponda.

Para evaluar si hubo ascenso o descenso, en cada función se hace un promedio de las zonas de interés para cada evento, por ejemplo, para la zona de evaluación del perfil derecho, se toman en cuenta de la zona 1 a la 6. Dicho promedio debe ser mayor a 3 para garantizar que hubo un flujo de movimiento en dirección al evento al que se está aspirando. Cuando es correcto, se limpia el vector principal, las zonas, se aumenta el conteo de descenso en 1 y el proceso vuelve a repetirse, Ilustración 145.

```

public void evaluateDownPassager() {

    int average = (zone1 + zone2 + zone3 + zone4 + zone5 + zone6);
    if (average >= 3) {
        counterDown++;
        personTestCoordinates.clear();
        counterFrames = 0;
        counterRefresh = 0;
        clearZones();
    }
}

```

Ilustración 145 - Función para el conteo de descenso de pasajeros

3.7.5.6. Protección contra obstrucciones

El primer paso, consiste en convertir la imagen a HSV, esto hace que la selección de objetos basados en colores es un poco más simple, la función *cvtColor* se encarga la conversión y en este caso, la salida corresponde a una nueva variable del tipo *Mat*, *mRgbaTest*.

A continuación, con la función *inRange*, se hace un proceso para seleccionar todos los pixeles dentro de un rango de color, lo cual proporciona valores de umbral superior e inferior a dicha función, con lo que se genera una máscara de imagen binaria donde los pixeles de primer plano, están dentro del rango especificado.

La función *dilate*, ayuda a eliminar el ruido en la máscara ya que agrega pixeles a los límites de los objetos de la imagen.

Finalmente, la función *findContours* calcula los contornos de imágenes binarias, cada una de estas capturas se almacena en el vector *contours*. El proceso descrito en esta sección se muestra en la Ilustración 146.

```

Imgproc.cvtColor(mRgba, mRgbaTest, Imgproc.COLOR_RGB2HSV, dstCn: 1);
Scalar lowerThreshold = new Scalar(120, 100, 100); // Blue color - lower hsv values
Scalar upperThreshold = new Scalar(179, 255, 255); // Blue color - higher hsv values
Core.inRange(mRgbaTest, lowerThreshold, upperThreshold, mRgbaTest);
Imgproc.dilate(mRgbaTest, mRgbaTest, new Mat());
List<MatOfPoint> contours = new ArrayList<>();
Mat hierarchy = new Mat();
Imgproc.findContours(mRgbaTest, contours, hierarchy, Imgproc.RETR_LIST, Imgproc.CHAIN_APPROX_SIMPLE);
Log.d( tag: "contornos", String.valueOf(contours.size()));
MediaPlayer player = MediaPlayer.create( context: this, R.raw.camera_alert);

```

Ilustración 146 - Script de protección contra obstrucciones

El algoritmo mostrado determina la cantidad de contornos que encuentra en la captura de cada frame, de modo que cuando el vector *contours* tiene un tamaño igual a cero, se determina que existe obstrucción, activando una alerta sonora hasta que sea despejada la zona de la cámara y el vector tenga un tamaño mayor, Ilustración 147.

```

if (contours.size() > 0) {
    player.stop();
}

```

Ilustración 147 - Evaluación del vector contours

3.8. Prototipo Final

El último prototipo desarrollado busca dar por cumplidos todos los objetivos, metas y requerimientos propuestos en el presente trabajo de grado. El principal reto en la implementación de éste, correspondió a la unificación de los 2 prototipos anteriores, es decir, implementar la detección, seguimiento y conteo del flujo de personas sobre el prototipo 1, con lo cual se lograría una aplicación móvil final en la cual, al presentarse el evento de ascenso o

descenso de una persona, se registraría en la base de datos, con la información de interés de la captura, la información del bus y conductor de la cuenta de usuario activa en el DMC. Adicionalmente, toda la información recolectada estará disponible desde la AWGI con las funcionalidades que esta tiene.

3.8.1. Integración

La implementación del prototipo 3 consiste en la integración de los primeros dos prototipos, de modo que es necesario agregar el algoritmo de detección y seguimiento con OpenCV a la actividad actual de conteo simulado del DMC, por ende, serán reemplazados los botones que efectuaban dicha simulación. Para lo anterior es necesario convertir la actividad que está en lenguaje Java a lenguaje Kotlin para su integración. Una vez integrada la librería y el algoritmo desarrollado, se conecta el conteo con las variables propias de Firebase para que los eventos se vean reflejados en la AWGI.

3.8.1.1. Migración de clase Java a Kotlin del algoritmo de detección y seguimiento

Una vez implementado el algoritmo en lenguaje Java, se procede con la migración a Kotlin para la integración con todo el entorno del prototipo 1 del DMC.

Definición de la clase, Ilustración 148.

```
class OpenCVActivity : Activity(), CameraBridgeViewBase.CvCameraViewListener2 {
```

Ilustración 148 - Definición de la clase OpenCVActivity en Kotlin

El bloque de carga del archivo Haar, queda como sigue, Ilustración 149.

```
try {
    // Carga de Haarcascade_Profileface haarcascade_profileface.xml
    val `is`: InputStream = resources.openRawResource(R.raw.haarcascade_profileface)
    val cascadeDir: File? = getDir( name: "cascade", Context.MODE_PRIVATE)
    mCascadeFile = File(cascadeDir, child: "haarcascade_profileface.xml")
    val os = FileOutputStream(mCascadeFile)

    val buffer = ByteArray( size: 1024)
    var bytesRead: Int = 0
    while (`is`.read(buffer).also { bytesRead = it } >= 0) {
        os.write(buffer, off: 0, bytesRead)
    }
    `is`.close()
    os.close()

    mJavaDetector = CascadeClassifier(mCascadeFile!!.absolutePath)
    if (mJavaDetector!!.empty()) {
        Log.e(TAG, msg: "Failed to load cascade classifier")
        mJavaDetector = null
    } else {
        Log.i(TAG, msg: "Loaded cascade classifier from " + mCascadeFile!!.absolutePath)
        cascadeDir.delete()
    }
} catch (e: IOException) {
    e.printStackTrace()
    Log.e(TAG, msg: "Failed to load cascade. Exception thrown: $e")
}
```

Ilustración 149 - Carga de Haar Cascade Kotlin

La sección de dibujo no sufre cambios pues la función *Imgproc* se comporta de la misma manera que en Java.

Por otro lado, la detección queda definida en Kotlin como se observa en la Ilustración 150.

```

val faces = MatOfRect()
val faces2 = MatOfRect()

// ===== SECCIÓN DT (detección) =====//
// SE DETECTA EL PERFIL IZQUIERDO POR MEDIO DEL INPUTFRAME mGray
if (mDetectorType == JAVA_DETECTOR) {
    if (mJavaDetector != null)
        mJavaDetector!!.detectMultiScale(mGray, faces, scaleFactor 1.1, minNeighbors: 2, flags: 2,
            Size(mAbsoluteFaceSize.toDouble(), mAbsoluteFaceSize.toDouble()), Size())
    } else {
        Log.e(TAG, msg: "Detection method is not selected!")
    }
}

// SE DETECTA EL PERFIL DERECHO POR MEDIO DEL INPUTFRAME mGray2
if (mDetectorType == JAVA_DETECTOR) {
    if (mJavaDetector != null)
        mJavaDetector!!.detectMultiScale(mGray2, faces2, scaleFactor 1.1, minNeighbors: 2, flags: 2,
            Size(mAbsoluteFaceSize.toDouble(), mAbsoluteFaceSize.toDouble()), Size())
    } else {
        Log.e(TAG, msg: "Detection method is not selected!")
    }
}

//===== FIN DE LA SECCIÓN DT =====//

// SE ALMACENAN LAS DETECCIÓNES EN UN ARRAY DE TIPO Rect[]
val facesArray : Array<out Rect!> = faces.toArray() // IZQUIERDO
val facesArray2 : Array<out Rect!> = faces2.toArray() // DERECHO

```

Ilustración 150 - Sección de detección Kotlin

Las funciones complementarias del algoritmo sufren leves cambios en declaración de variables y asignación de las mismas, pero la estructura no se modifica.

3.8.1.2. Integración total de funcionalidad

Resta conectar las lecturas de conteo total y parcial con el algoritmo de detección. Dado que en el prototipo 1 del DMC ya se cuenta con dichas consultas, basta con reemplazar la inserción en base de datos al momento de gestionar el evento, entonces, una vez se llama la función que evalúa ascenso o descenso del prototipo dos, se llaman las funciones *writeDown* y *writeUp* respectivamente, las cuales se observan en la Ilustración 151 e Ilustración 152.

```

fun writeDown() {
    currentHour = Date().formatHour()
    var downCount = Boardings(currentHour, approachType: false, parcialCount: parcialCount - 1, totalCount, latitude, longitude)
    val mUser : FirebaseUser? = DBConnection.mAuth.currentUser
    val mUserReference : DatabaseReference = mDatabaseReferenceCounter!!.child( pathString: mUser!!.uid + "/" + currentDate + "/boardings")
    val mCountReference : DatabaseReference = mDatabaseReferenceCounter!!.child( pathString: mUser.uid + "/" + currentDate)
    val tempCount : DatabaseReference = mCountReference.child( pathString: "parcialCount")
    tempCount.setValue(parcialCount - 1)
    mUserReference.push().setValue(downCount)
}

```

Ilustración 151 - Función para insertar descenso

```
fun writeUp() {
    currentHour = Date().formatHour()
    if (parcialCount + 1 == maxCapacity) {
        var player : MediaPlayer! = MediaPlayer.create(applicationContext, R.raw.alert)
        player.start()
    }

    if (parcialCount + 1 > maxCapacity) {
        var player2 : MediaPlayer! = MediaPlayer.create(applicationContext, R.raw.alert)
        player2.start()
    }

    var upCount = Boardings(currentHour, approachType: true, parcialCount: parcialCount + 1, totalCount: totalCount + 1, latitude, longitude)
    val mUser : FirebaseAuth? = DBConnection.mAuth.currentUser
    val mUserReference : DatabaseReference = mDatabaseReferenceCounter!!.child( pathString: mUser!!.uid + "/" + currentDate + "/boardings")
    val mCountReference : DatabaseReference = mDatabaseReferenceCounter!!.child( pathString: mUser.uid + "/" + currentDate)
    val mCountReferenceTotalPasseggers : DatabaseReference = mDatabaseReferenceCounter!!.child(currentDate)
    val tempCountParcial : DatabaseReference = mCountReference.child( pathString: "parcialCount")
    val tempCountTotal : DatabaseReference = mCountReference.child( pathString: "totalCount")
    val tempCountTotalPasseggers : DatabaseReference = mCountReferenceTotalPasseggers.child( pathString: "total")
    tempCountParcial.setValue(parcialCount + 1)
    tempCountTotalPasseggers.setValue(totalCountAllPasseggers + 1)
    tempCountTotal.setValue(totalCount + 1)
    mUserReference.push().setValue(upCount)
}
```

Ilustración 152 - Función para insertar ascenso

Capítulo 4

PRUEBAS Y RESULTADOS

En este capítulo se plasman los resultados producto de las diferentes pruebas realizadas a cada uno de los prototipos desarrollados dentro del proyecto, mostrando sus características, procesos de evaluación, efectividad y falencias, exponiendo las capacidades reales de cada prototipo.

4.1. Prototipo 1

En las diferentes pruebas realizadas al prototipo inicial, se evaluó el funcionamiento de la AWGI, desde la gestión de usuarios, rutas y buses, hasta el despliegue de datos capturados por el DMC en campo. Por otra parte, con respecto al DMC se evaluó su funcionamiento para la captura de datos de interés (día, hora, tipo de evento, ruta, conductor, coordenadas) al simular el evento ascenso o descenso.

4.1.1. Descripción de las pruebas

El primer paso para comprobar el correcto funcionamiento de la AWGI es iniciar sesión en esta con el administrador root, el cual fue creado directamente desde la base de datos (Firebase). Al validar sus datos y comprobar el correcto funcionamiento de la conexión entre la aplicación web y la base de datos, desde el perfil del administrador root se verifica el correcto funcionamiento en la gestión de las rutas Ilustración 153, buses Ilustración 154 y usuarios (conductores) Ilustración 155, necesarios para realizar las diferentes pruebas y captura de datos.

Se debe tener en cuenta que el primer elemento a crear son las rutas, ya que, al crear un bus, a este se le debe asignar una ruta existente, y a su vez a cada conductor se le asigna una ruta y un bus perteneciente a ella.

Lista de Rutas

Nombre	Descripción	Recorrido	Opciones
Ruta 1	Ruta 1 para pruebas de prototipo 1	Recorrido 1 Buga-Valle	 
Ruta 2	Ruta 2 para pruebas de prototipo 1	Recorrido 2 Buga-Valle	 

Ilustración 153 - Rutas creadas en la AWGI - Pruebas prototipo 1

Lista de Buses





Placa	Modelo	Marca	Capacidad	Ruta	Opciones
GXS-326	2010	Hyundai	19	Ruta 1	 
DTY-346	2008	Toyota	20	Ruta 2	 

Ilustración 154 - Buses creados en la AWGI - Pruebas prototipo 1

Lista de Conductores

Nombre	Cédula	Ruta	Bus	Estado	Opciones
Conductor 1	10000000	Ruta 1	GXS-326		 
Conductor 2	11122555	Ruta 2	DTY-346		 

Ilustración 155 - Conductores creados en la AWGI - Pruebas prototipo

Luego de crear el conductor 1 y conductor 2, asignarle una ruta y bus correspondiente a cada uno, se ingresó con estos perfiles desde el DMC, validando sus datos, comprobando así el correcto funcionamiento en la conexión de la aplicación móvil y la base de datos, al iniciar sesión, desde la interfaz principal Ilustración 156, se ejecutó la acción de iniciar el conteo de pasajeros.



Ilustración 156 - Interfaz principal DMC prototipo 1

Una vez iniciado el conteo, dentro de la interfaz de simulación de conteo de pasajeros Ilustración 157, se procedió a la captura de datos, la cual se realizó en la ciudad de Buga, Valle del Cauca (Ilustración 158), de donde es originaria la empresa prestadora de servicio de transporte la cual permitió realizar las pruebas de los prototipos de este proyecto de grado, además es de importancia resaltar que dada la fase actual del sistema prototipo y de la tecnología con la que se implementa, la cual no tiene contacto directo con los usuarios, no se tienen grandes implicaciones que afecten el sistema con respecto a la ciudad de Popayán donde se planteó el desarrollo del sistema.

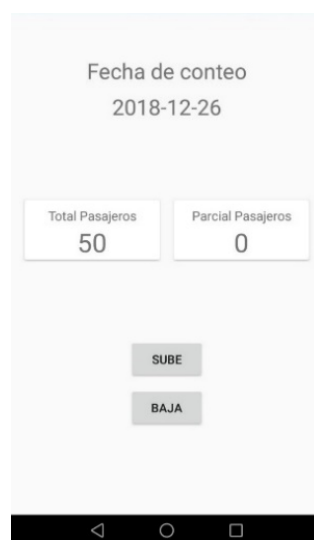


Ilustración 157 - Interfaz de simulación de conteo de pasajeros DMC prototipo 1

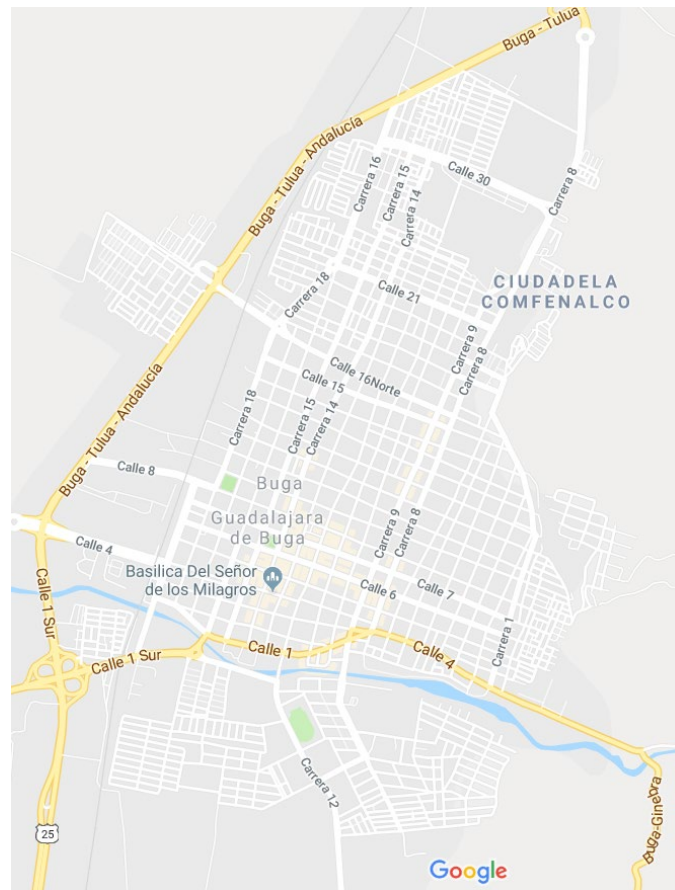


Ilustración 158 - Mapa ciudad de Buga - Valle del cauca

Para realizar la recolección de datos y posterior evaluación del funcionamiento del sistema se trazaron 2 rutas Ilustración 159, Ilustración 160, para los conductores 1 y 2 y su bus correspondiente.

Para cada ruta se realizó el desplazamiento en campo siguiendo el recorrido fijado, dicho recorrido se realizó el día 26 de diciembre de 2018, con el perfil del conductor correspondiente, y simulando el ascenso y descenso de pasajeros en diferentes posiciones del recorrido.

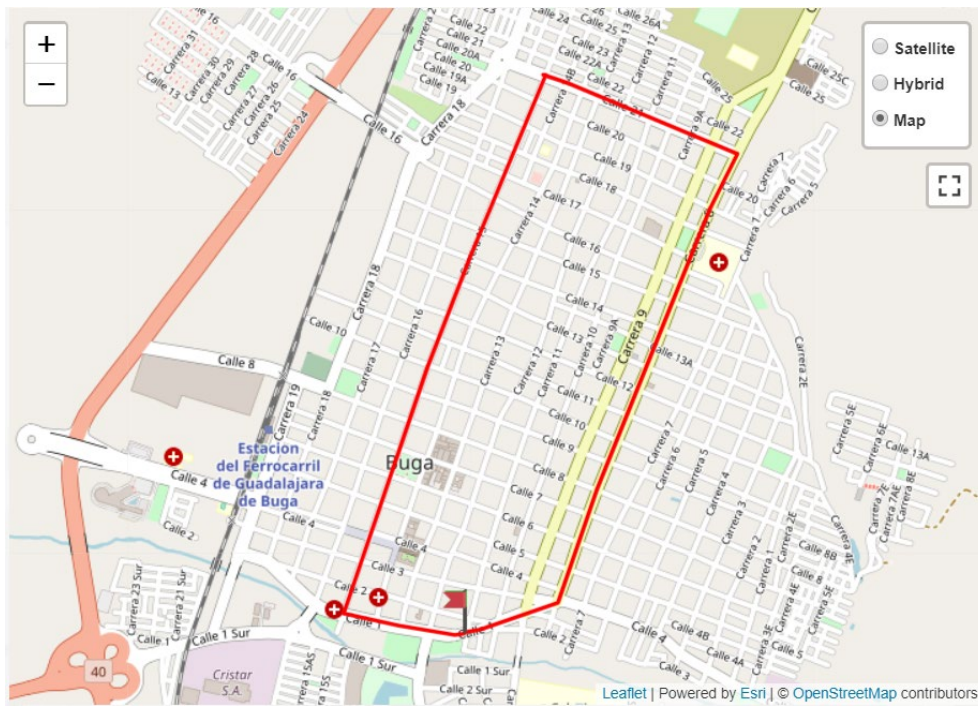


Ilustración 159 - Recorrido Ruta 1

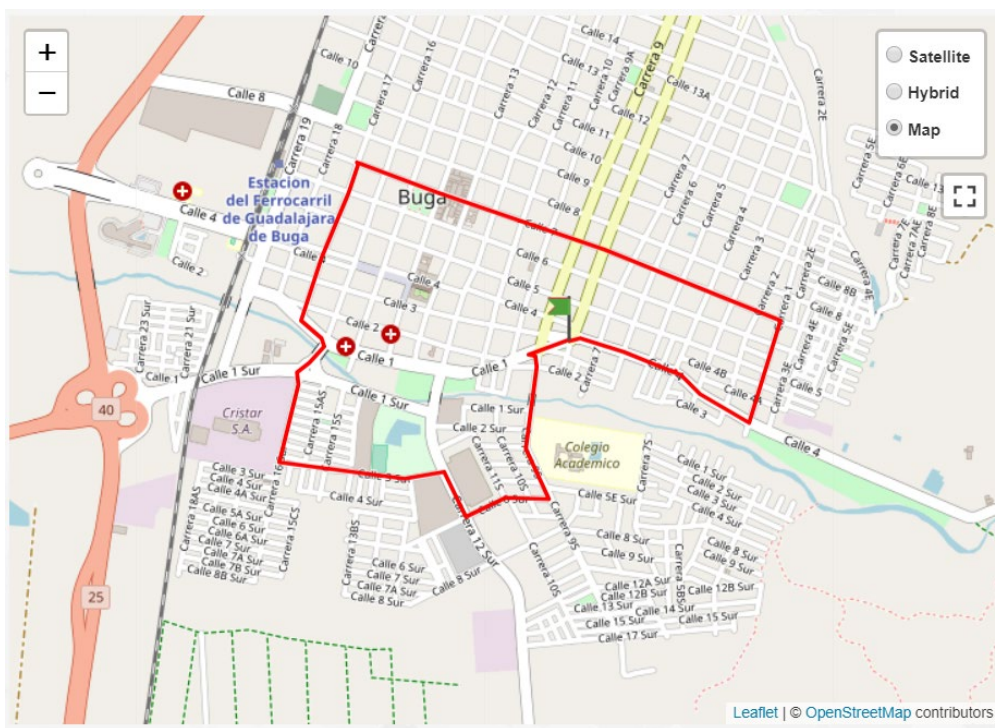


Ilustración 160 - Recorrido ruta 2

El recorrido 1, correspondiente al conductor 1, bus GXS-326 y ruta 1, se capturó un total de 50 ascensos y 50 descensos, para un total de 100 eventos registrados, repartidos por el total del recorrido. Para comprobar todos los datos registrados desde la interfaz de “Graficas” de AWGI se realizó la búsqueda en la fecha en la que se realizó la toma de datos, obteniendo una tabla con todos los datos registrados en la base de datos, Tabla 25- Anexo I, comprobando que se registró el 100% de los datos cada uno en el lugar correspondiente. En la Ilustración 161, se observa otra de las funciones del prototipo y una de las herramientas más importantes del sistema, correspondiente a un mapa de calor en donde se visualizan cada uno de los lugares donde se registró un evento de ascenso o descenso, esto mediante la interfaz “mapa” e ingresando la ruta, bus y fecha de los datos que se desean visualizar.

Este mapa mostrar los puntos de mayor afluencia, llamados puntos de calor, con los cuales se pueden detectar los lugares en los cuales hay mayor uso del servicio y donde este se congestiona más. Se puede comprobar además que hay una alta precisión en la localización comparando la Ilustración 159 con la Ilustración 161.

Ruta:	Bus:	Fecha:	<input type="button" value="Buscar"/>
<input type="text" value="Ruta 1"/>	<input type="text" value="GXS-326"/>	<input type="text" value="26/12/2018"/>	

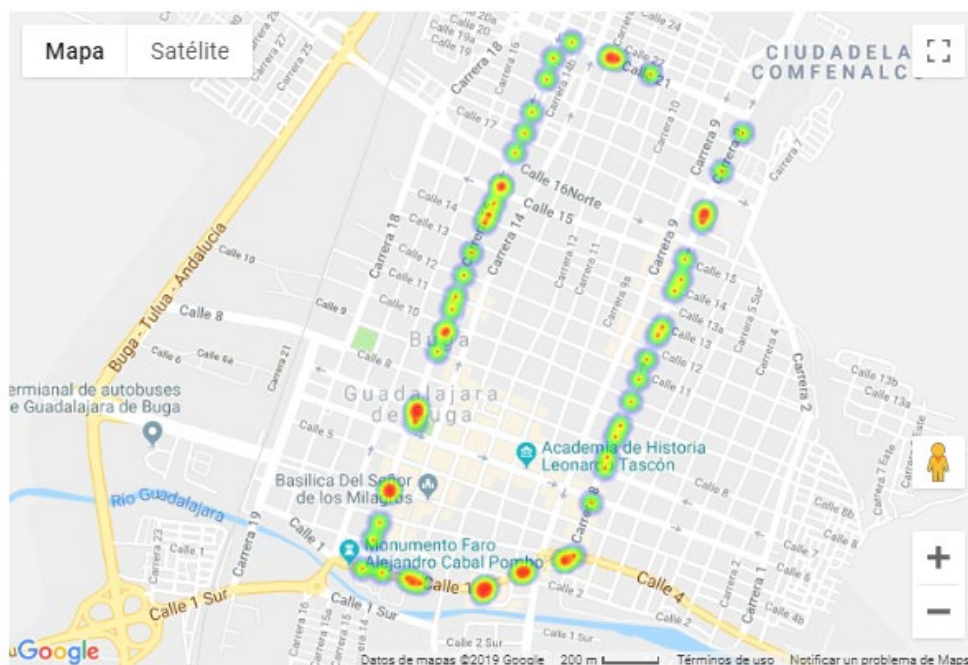


Ilustración 161 - Captura de datos Conductor 1- bus GXS-326 - Ruta 1

Otra de las gráficas que se puede ver desde la AWGI se observa en la Ilustración 162 donde se evidencia el comportamiento de la cantidad de personas dentro del bus en cada momento del día en los cuales se tienen datos de ascensos o descensos. Dicha cantidad se compara con el Contador parcial de los datos capturados para este recorrido Tabla 25 – Anexo H, y se comprueba un 100% de concordancia.

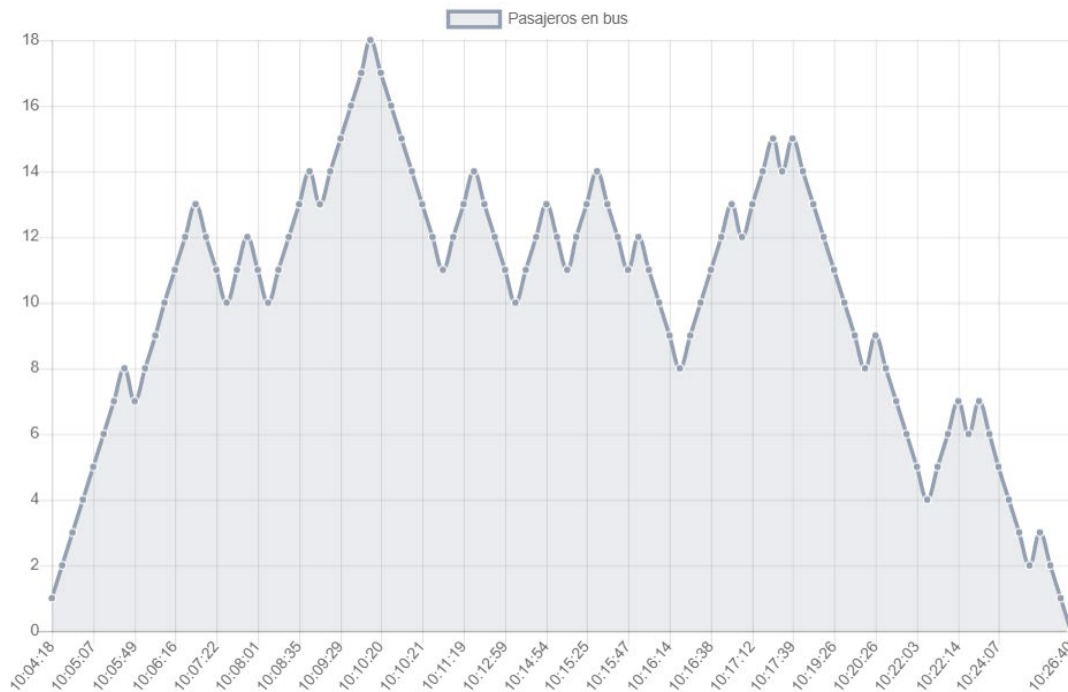


Ilustración 162 – Conteo Parcial de pasajeros vs Hora – Conductor 1 – Ruta 1

Se realizó el mismo proceso para el segundo recorrido con el conductor 2 – ruta 2, donde se capturaron 73 eventos de ascenso y 73 eventos de descenso, para un total de 146 eventos registrados. Al visualizar desde la AWGI los datos registrados, Tabla 26 – Anexo H, se obtiene un 100% de relación entre los datos capturados y los registrados, comprobando nuevamente el correcto funcionamiento del prototipo 1.

Igualmente se comprueban las locaciones de los datos capturados y los puntos de calor del recorrido realizado como se observa en la Ilustración 163, comprobando nuevamente la precisión de los datos capturados con respecto a la posición donde estos fueron registrados. De igual forma se corrobora el conteo parcial en este recorrido durante el tiempo que se realizó (Ilustración 164).

MAPA

Ruta: Bus: Fecha:

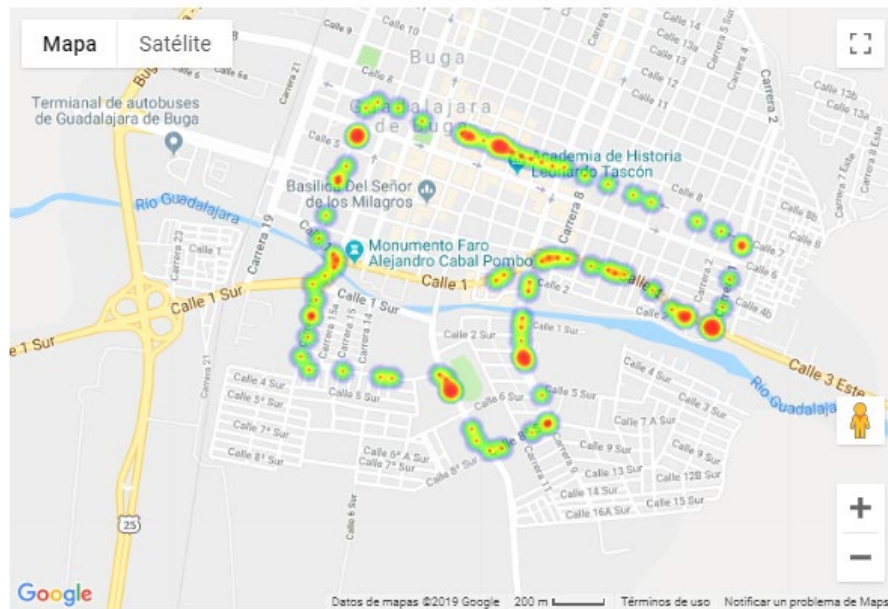


Ilustración 163 - Captura de datos Conductor 2- bus DTY-346 - Ruta 2

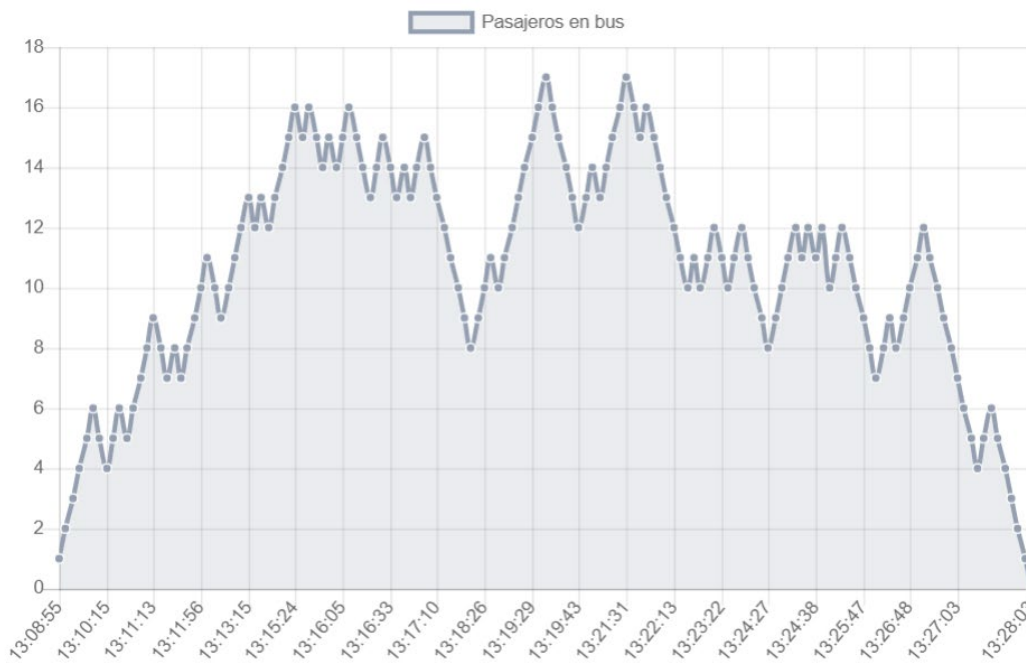


Ilustración 164 - Conteo Parcial de pasajeros vs Hora - Conductor 2 - Ruta 2

Además de la búsqueda de datos de abordaje para un solo día, se implementó la búsqueda para un rango de días, donde se obtienen todos los datos de abordajes del rango de días seleccionado, en la Ilustración 165 se puede observar la gráfica obtenida con la búsqueda para las presentes pruebas, en concreto se buscan todos los datos entre el 24 y 26 de diciembre de 2018 para el bus GSX-326 perteneciente a la ruta 1, se evidencia el correcto funcionamiento de esta función, conociendo que para el día 26 de diciembre el sistema transportó 50 pasajeros como se ha constatado en las anteriores pruebas.

Esta funcionalidad les permite a las empresas prestadoras del servicio TPC tener un registro exacto del número de pasajeros que movilizaron en cada uno de los días que desee, y hacer una comparativa entre ellos, brindando una herramienta para obtener información de interés del servicio.

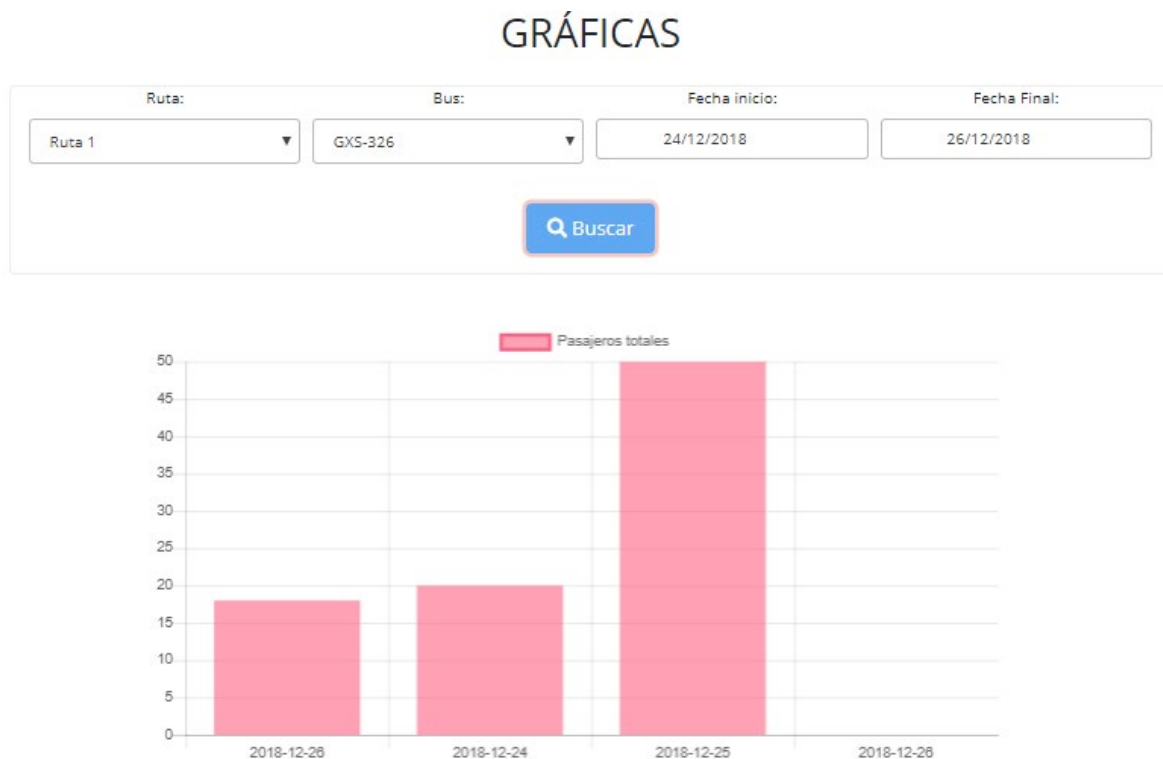


Ilustración 165 - Grafica rango de días para bus y ruta seleccionada.

Tanto para la búsqueda de datos de abordaje en un día específico como para un rango de días, se evaluó, además, la función para descargar de las tablas con los datos consultados (Ilustración 166). Para comprobar el correcto funcionamiento de esta función, dichas tablas se plasmaron dentro de la Tabla 27 - Anexo H, corroborando los

datos registrados y la descarga. Los archivos son descargados en formato csv y se les asigna por nombre los parámetros consultados, facilitando así su reconocimiento.

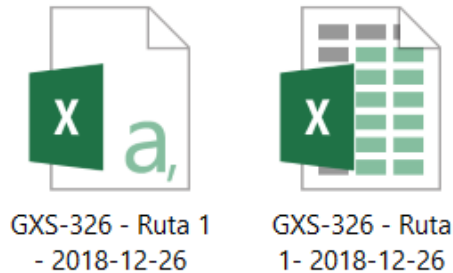


Ilustración 166 - Archivos descargadas de tablas AWGI.

Se implementó además, dentro de la interfaz “Gráficas” en la AWGI, una gráfica que muestra el conteo total de pasajeros movilizados por el bus seleccionado para cada hora del día elegido (Ilustración 167), cabe resaltar que esta gráfica permite observar el incremento hora a hora de pasajeros permitiéndole conocer a la empresa prestadora del servicio TPC las horas con mayor afluencia información de vital importancia que en conjunto con los mapas de calor, brinda a la empresa las herramientas para poder conocer el estado del servicio prestado.

GRÁFICAS

Formulario de búsqueda para la gráfica:

Ruta: Bus: Fecha inicio: Fecha Final:



Ilustración 167 - Grafica conteo total de pasajeros bus seleccionado por hora.

Por último, en la interfaz principal “Home” se implementó un apartado donde los usuarios que ingresen podrán observar el número total de pasajeros movilizados y el total de buses activos para el día en curso (Ilustración 168), con lo cual tendrán un rápido acceso a información actualizada de cómo se registra el uso del servicio durante el día.

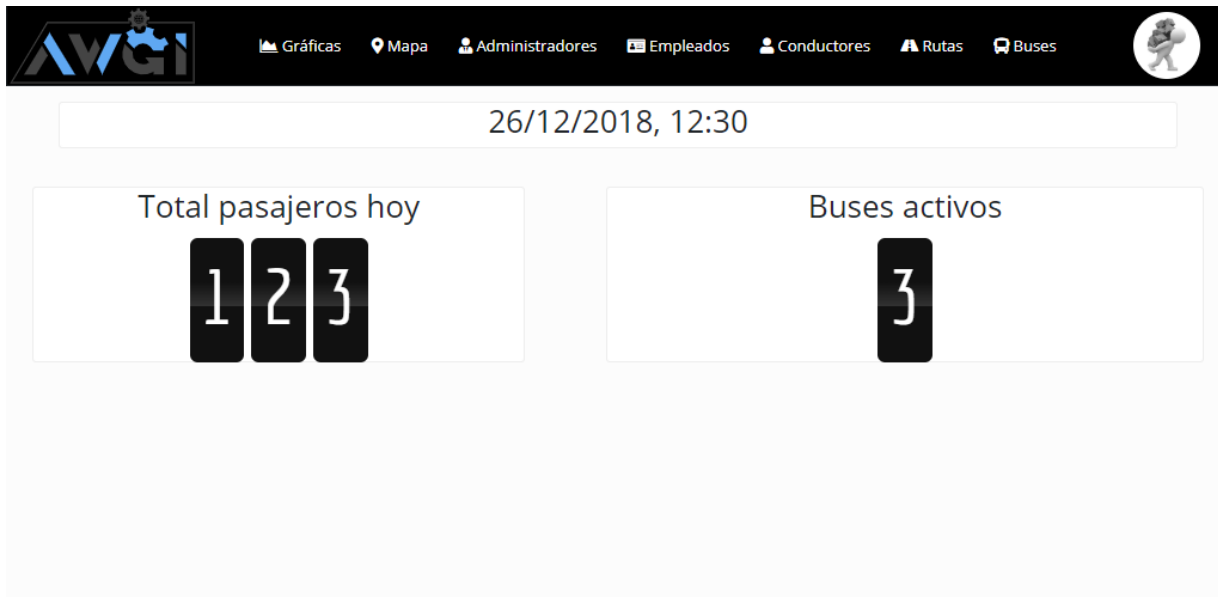


Ilustración 168 - Interfaz principal.

4.1.2. Análisis de resultados

Se comprueba que el prototipo 1, realizó las tareas esperadas con un margen de error del 0%, debido a que presentó una conexión a internet estable durante el tiempo que se capturaron los datos (aproximadamente 4 horas) y que la base de datos utilizada (Firebase) es de gran velocidad (tiempo real) y eficiencia. Con lo cual se tiene que tanto el módulo DMC y AWGI realizaron las funciones esperadas, permitiendo la captura, registro y posterior visualización de datos de interés correspondientes a la simulación del flujo de pasajeros en un recorrido real con un éxito del 100% para las condiciones bajo las que se realizaron las pruebas.

A partir de las pruebas realizadas, se encuentra que el principal factor que puede influir en que se presenten determinados errores es una conexión inestable a internet por parte del DMC, la cual se evidenciaría con inserciones erróneas o posibles datos perdidos.

También se evidencia que las funciones brindadas por el proyecto para la gestión de los datos recolectados, es de gran ayuda y permite obtener información de gran interés

de forma rápida y actualizada, con lo cual las empresas prestadoras del servicio TPC podrán contar con herramientas para realizar una mejor gestión de su servicio.

4.2. Prototipo 2

Las pruebas realizadas para el prototipo 2 fueron encaminadas a comprobar el funcionamiento de los clasificadores en cascada utilizando el método Haar, por medio de los diferentes detectores suministrados por la librería de OpenCV, de tal forma que se pudiese comprobar su viabilidad dentro del proyecto.

4.2.1. Descripción de las pruebas

El primer detector utilizado corresponde al “haarcascade_frontalface.xml” [80] el cual, como su nombre lo indica busca detectar el rostro humano de forma completamente frontal, este detector se implementó, junto con un detector de ojos, los cuales funcionaban correctamente como se puede observar en la Ilustración 169, comprobando la versatilidad de los clasificadores en cascada para detectar rostros y diferentes objetos. Estos detectores estaban en la capacidad de detectar rápidamente el objeto deseado así que, si este realiza un movimiento, veloz mente vuelve a ser detectado en la nueva posición.

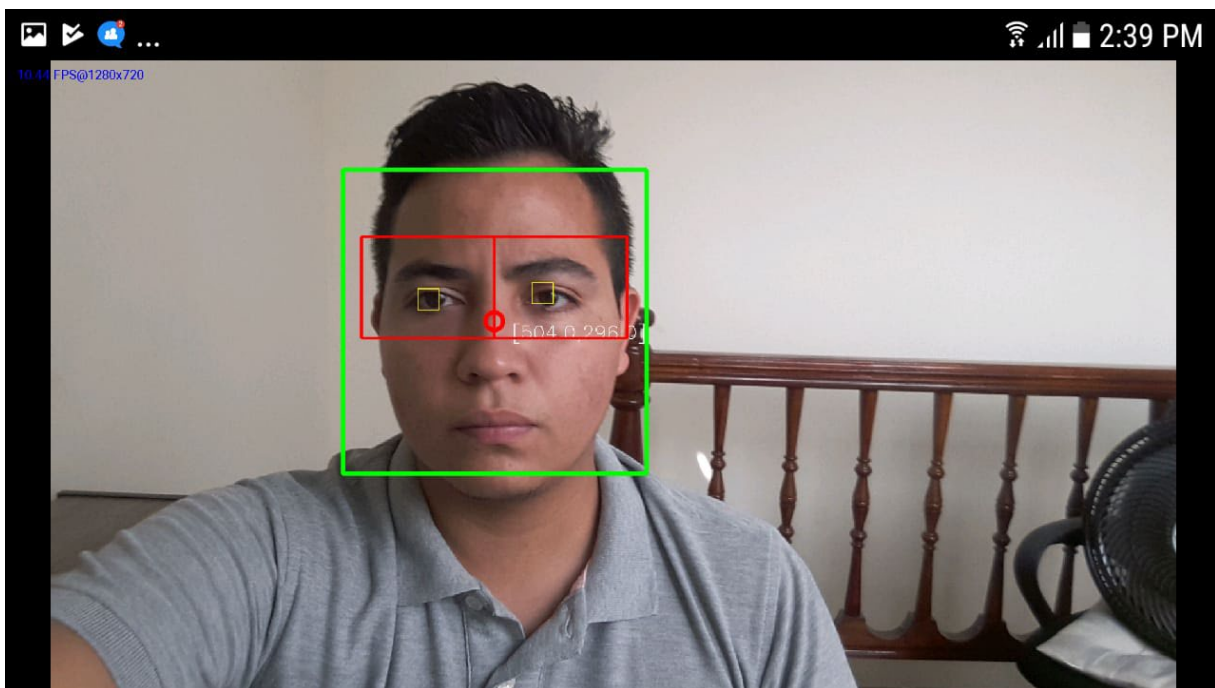


Ilustración 169 - Prueba detector de rostro frontal y ojos

Se realizaron pruebas de detección en el ambiente simulado de un espacio con luz normal a diferentes horas del día, con una intensidad lumínica como se describe a continuación:

- Entre 8 – 11 am ~ 105_{lx}¹⁹
- Entre 3 – 5 pm ~ 70_{lx}
- Entre 6 – 8pm (con luz blanca artificial) ~ 85_{lx}

Datos a partir de los cuales se pudo observar la versatilidad del detector, además de su capacidad para detectar múltiples rostros y su alta velocidad de procesamiento, se procedió a realizar las pruebas de *Tracking*, es decir, del seguimiento del objeto detectado, en las cuales como se mencionó anteriormente se evaluarán las nuevas opciones de detección con relación a las anteriores para conocer la dirección en que se desplaza el objeto, utilizando las 8 diferentes zonas implementadas, las cuales se activan cuando el objeto es detectado dentro de ellas, y luego de pasado el umbral de decisión, ya sea de ascenso (*Up*) o descenso (*Down*), se contabilizara dicho evento.

En la Ilustración 170, se puede evidenciar lo mencionado anteriormente, la detección del rostro actual está en la zona 6, si este se desplaza hacia la izquierda se podrían detectar las zonas 5, 4 y 3, y luego de pasar la línea de color verde se incrementaría el contador *Up*, simulando un evento de ascenso. También en la Ilustración 170 podemos observar que el contador *Down*, se encuentra en 1, esto es porque anteriormente se realizó el proceso inverso, donde la detección se dio en las diferentes zonas de izquierda a derecha, terminando el rostro en la posición mostrada.

¹⁹ El lux (símbolo lx) es la unidad derivada del Sistema Internacional de Unidades para la iluminancia o nivel de iluminación. Equivale a un lumen /m².

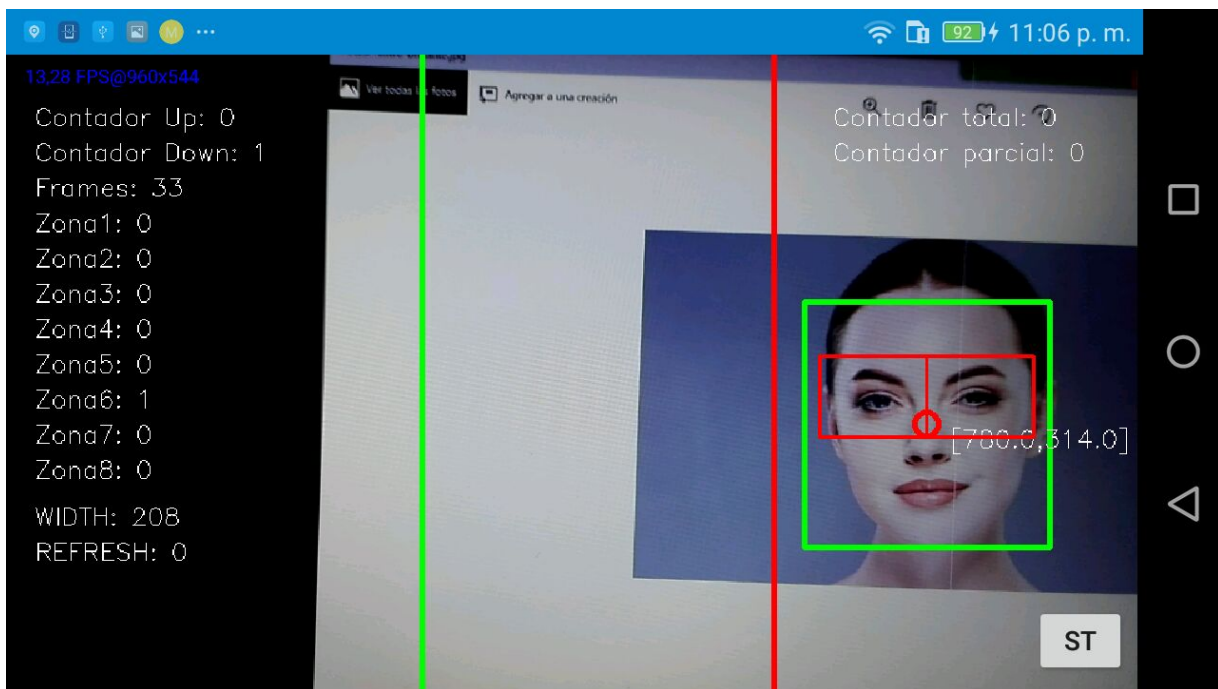


Ilustración 170 - Prueba de detección y tracking

Luego de realizar diferentes pruebas de detección y seguimiento, se procedió a simular el proceso de ascenso y descenso en un ambiente ideal, intentado tener el mayor acercamiento posible al ambiente donde se implementará finalmente el prototipo. Para determinar la mejor posición en la cual debía ubicarse el DMC en el bus con respecto al ascenso y descenso de pasajeros se evaluaron 3 localizaciones, la primera fue colocar el dispositivo en la parte de arriba para detectar únicamente la parte superior de la cabeza del pasajero, pero hubo muchos problemas de rango de visión en los que se cubría mucho rango de visión. Así mismo, se hicieron pruebas de detección ubicando el dispositivo justo en frente al abordaje de cada usuario, pero dada la lejanía y la variación de figuras de cuerpos humanos, se determinó entonces que, debido a la forma como ingresan o descienden los pasajeros al bus, y buscando ser lo más transparente al usuario, además, de un ángulo de captura superior a 90 grados, en el cual se eviten obstrucciones, se determinó que el DMC debería estar posicionado de forma lateral al ascenso o descenso.

Dicho lo anterior se encontró que tanto el detector de rostro frontal como el de ojos, pese a su alta fiabilidad, no podrían ser de mucha utilidad dentro del proyecto ya que estos no pueden detectar el rostro de perfil o de costado, ya sea izquierdo o derecho, por lo cual se procedió a buscar diferentes alternativas, una de ellas era detectar la oreja del pasajero, ya que al pasar frente al DMC, la oreja de la persona sería un elemento que siempre estaría presente.

Basándose en lo anteriormente dicho, se utilizaron 2 detectores, uno para oreja izquierda y otro para derecha, en la Ilustración 171 se puede observar la detección de la oreja derecha utilizando uno de los clasificadores de la librería de OpenCV, esto junto al algoritmo diseñado

para el *tracking* de los elementos detectados, con lo cual se aumentan los contadores dependiendo de las zonas detectadas y del límite de decisión rebasado.

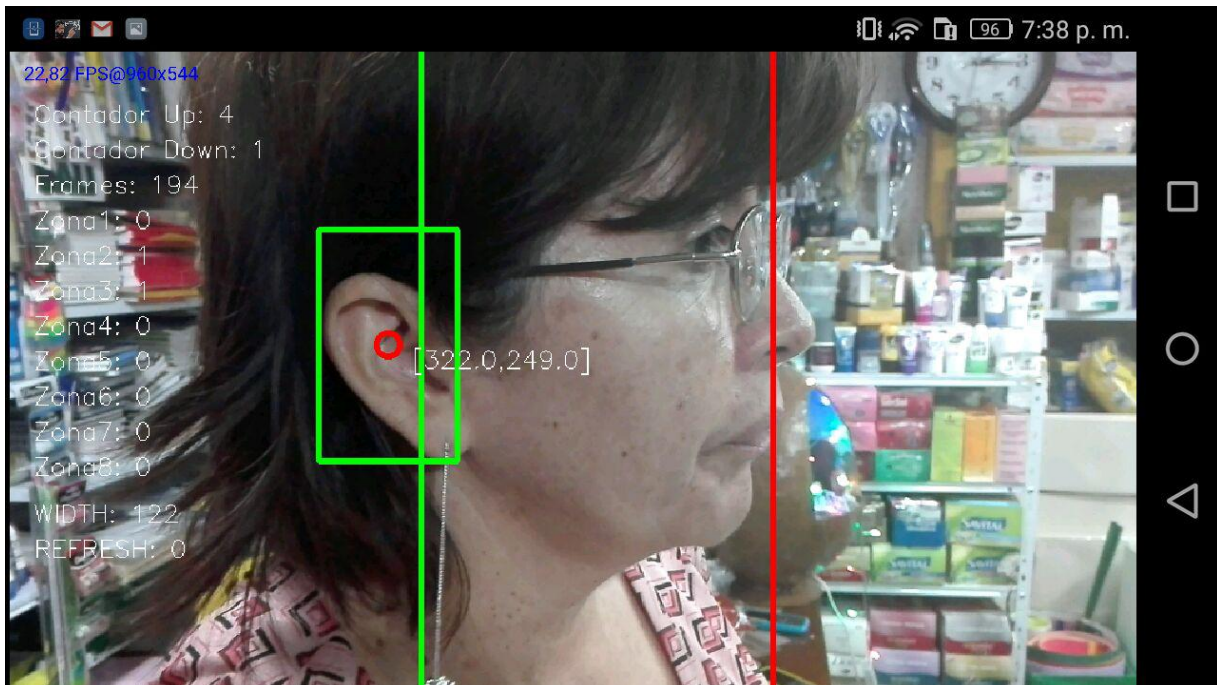


Ilustración 171 - Prueba detector oreja derecha

Sin embargo, al utilizar los detectores de orejas la fiabilidad del prototipo redujo de forma significativa, ya que el éxito de la detección de los clasificadores Haar-cascade, como se dijo anteriormente, dependen del número de características que se puedan guardar del objeto a detectar, y en un elemento como las orejas es mucho más limitado que en la del rostro humano.

Intentado tener un mayor número de características en el objeto detectado se procedió a utilizar el clasificador del rostro de perfil, con el cual, se obtuvieron mejores resultados, sin embargo, no permitía la detección de personas que pasaban con un tipo de inclinación o ángulo, por lo cual se encontró que la mejor alternativa era crear un detector con las características deseadas, para lo cual, se entrenó un clasificador propio para detectar los rasgos deseados.

Finalmente, con el clasificador entrenado y el algoritmo de seguimiento se implementó el prototipo 2 y su funcionamiento se evaluó como se describe a continuación:

- En la Ilustración 172 se puede evidenciar la detección correcta del perfil de una persona que se moviliza en dirección derecha-izquierda, en la cual se puede notar además, el rápido procesamiento para la detección de la imagen, el cual se evidencia en los aproximadamente 4 FPS que logra la aplicación, ya que la persona en cuestión se

encuentra en movimiento, en dicha ilustración se puede observar también que la zona 6 está activa ya que es donde se detectó el perfil de la persona.

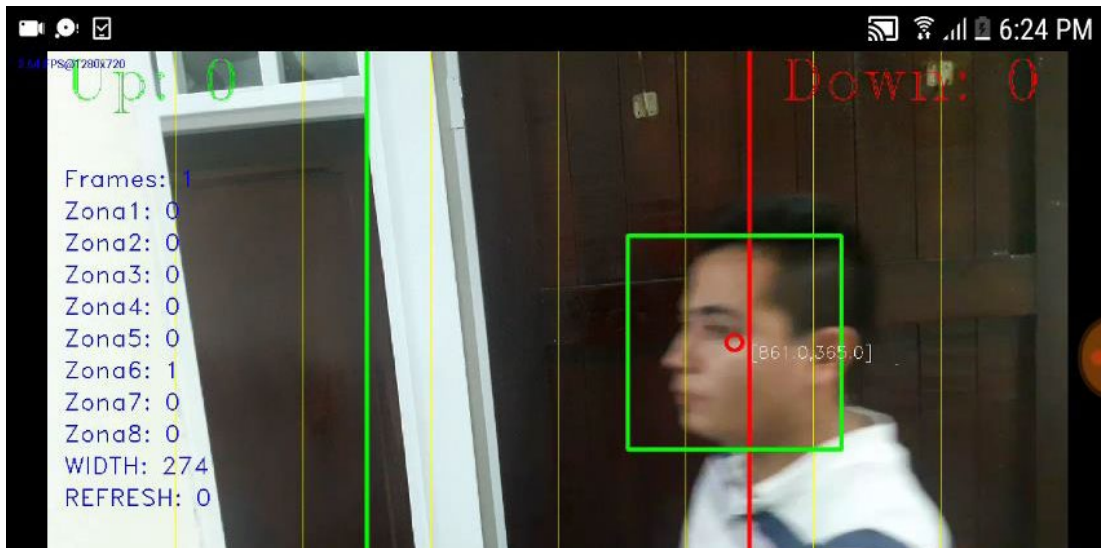


Ilustración 172 - Prueba final 1 prototipo 2

- En la Ilustración 173 se puede constatar el correcto funcionamiento del algoritmo de seguimiento, ya que al realizar el desplazamiento, las diferentes zonas donde fue detectado el perfil se activan, realizando el seguimiento del objeto detectado, con lo cual posteriormente como se observa en la Ilustración 174 al pasar la zona de decisión se incrementa el contador correspondiente, en el caso específico el evento de ascenso (Up), y las zonas son reiniciadas para un posterior conteo.

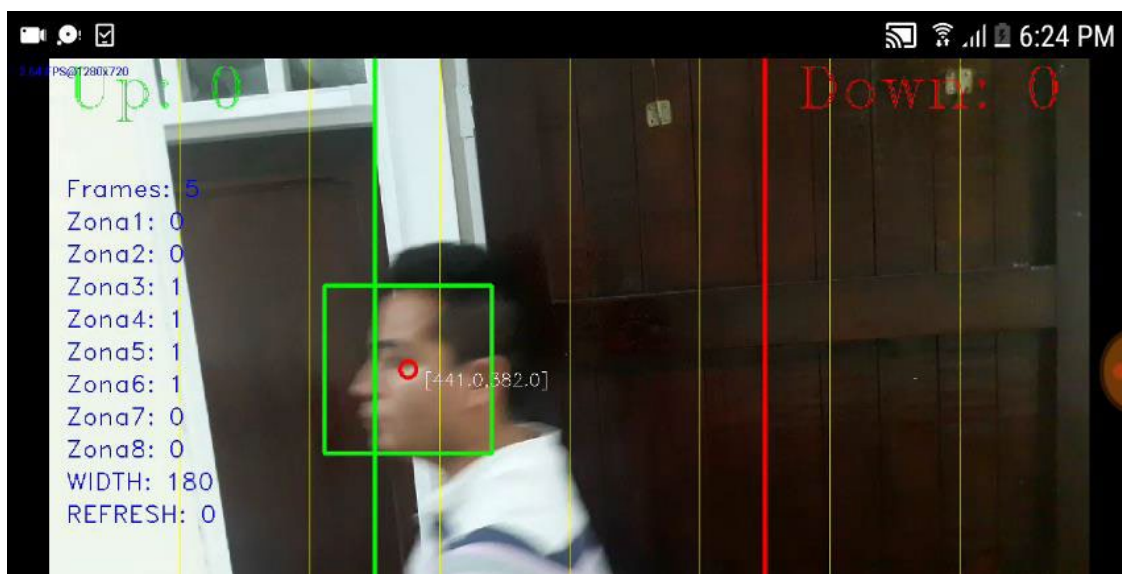


Ilustración 173- Prueba final 2 prototipo 2



Ilustración 174 - Prueba final 3 prototipo 2

- De igual forma, en las Ilustración 175 e Ilustración 176, se evidencia el proceso contrario, donde la persona en cuestión se desplaza de izquierda-derecha, activando las zonas correspondientes, para que posteriormente se registre el evento simulado de descenso.

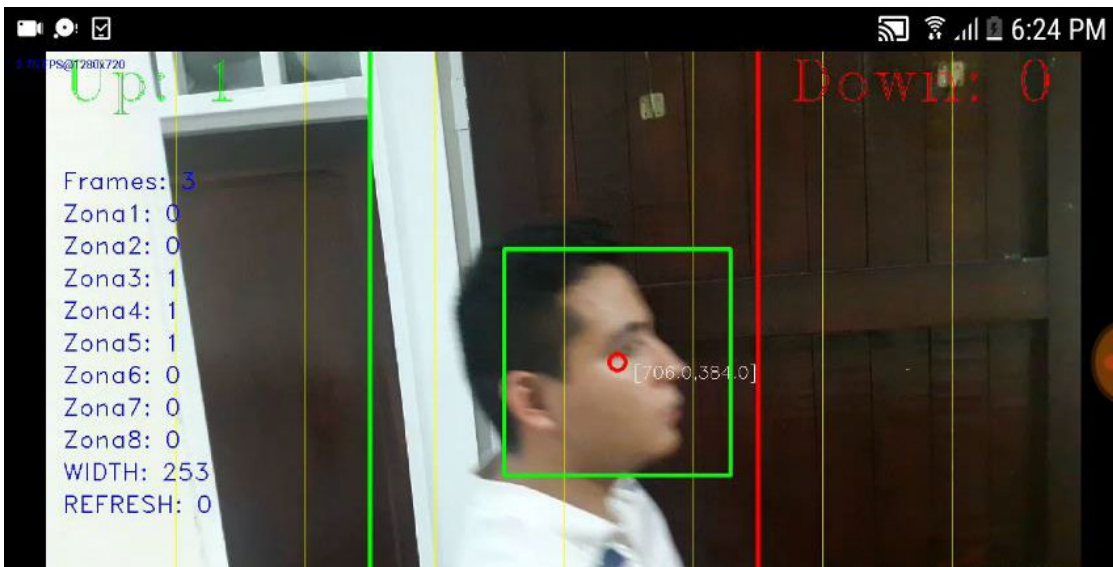


Ilustración 175 - Prueba final 4 prototipo 2

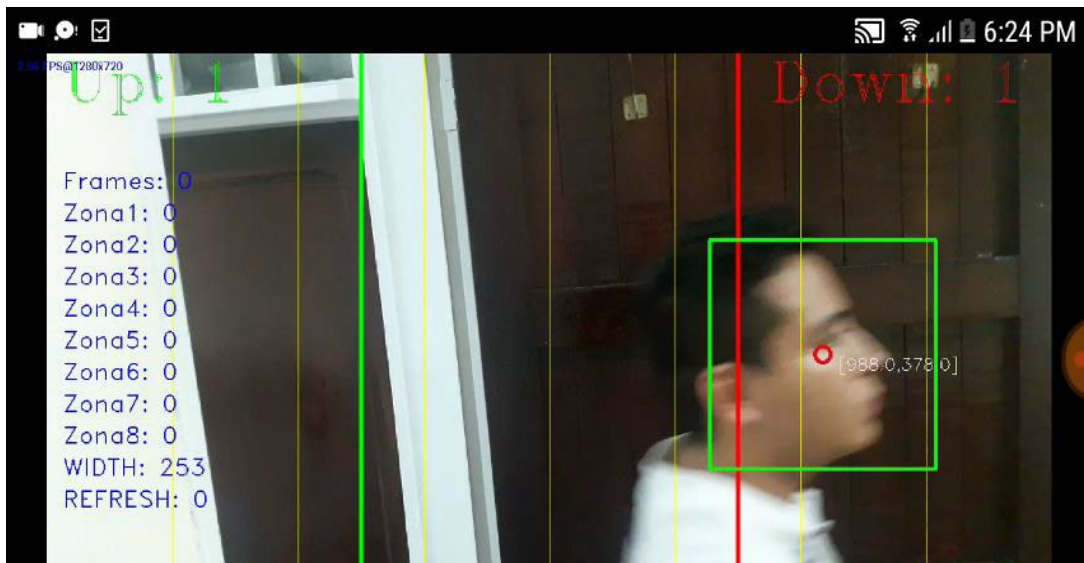


Ilustración 176 - Prueba final 5 prototipo 2.

Con lo anterior, se comprueba la correcta funcionalidad del prototipo 2, el cual busca realizar la detección, seguimiento y conteo del evento específico del conteo del flujo de personas por medio de una aplicación móvil. Además, se evidencia su alto grado de fiabilidad en el ambiente simulado e ideal en el cual se realizaron 100 conteos entre ascensos y descensos, con diferentes personas, en los cuales se obtuvo un 94% de éxito, la tabla de datos de soporte se encuentra disponible en el Anexo J.

4.2.2. Análisis de resultados

El prototipo 2 cumplió con su objetivo de implementar un método para el conteo del flujo de pasajeros utilizando la tecnología seleccionada, es decir el desarrollo de una aplicación móvil con la capacidad de detectar, seguir y realizar el posterior conteo del flujo de personas en un lugar específico, haciendo uso de la librería OpenCV y de los detectores Haar-cascade; dicha aplicación, demostró un alto grado de fiabilidad en el ambiente simulado, alcanzando un 94% de éxito con 100 pruebas realizadas entre ascenso y descensos con diferentes personas.

4.3. Prototipo Final

Para el prototipo final se realizaron diferentes tipos de pruebas, tanto para encontrar la posición más adecuada para la captura, como para comprobar la funcionalidad del prototipo en el ambiente real.

4.3.1. Pruebas posicionamiento DMC

Las principales características que se buscaron suplir para las pruebas de posicionamiento, fueron las siguientes:

- **Ángulo de visión:** Tener al menos 90 grados de visibilidad de la cámara del dispositivo
- **Intensidad lumínica:** Dadas las pruebas realizadas con antelación, se busca una ubicación que ofrezca al menos $\sim 70_{lx}$

Para encontrar el lugar y el ángulo más adecuado se realizaron múltiples pruebas utilizando un dispositivo trípode que permite alojar el DMC y variar su posición con 3 grados de libertad como se puede observar en la Ilustración 177 e Ilustración 178 permitiendo así realizar diferentes pruebas en un mismo lugar del bus, además, dicho dispositivo también se localizó en diferentes sitios para encontrar en cuál el éxito de captura fuera más alto por medio de simulaciones de abordajes.



Ilustración 177 - Dispositivo de alojamiento DMC



Ilustración 178 - Dispositivo de alojamiento DMC.

Luego de numerosas pruebas las cuales se encuentran plasmadas en el Anexo K, se encontró que la mejor posición de captura es con un ángulo de inclinación entre 15° y 25° paralelo a la trayectoria descrita por los pasajeros que ascienden, ubicando el DMC al lado derecho de la entrada (cabina de conducción), a aproximadamente 72cm del paso de personas, y a una altura del suelo del bus de 140 cm, como se observa en la Ilustración 179.



Ilustración 179 - Posición seleccionada DMC.

4.3.2. Pruebas funcionales DMC

Luego de seleccionada la posición final en la que será situado el DMC, se analizan las detecciones realizadas desde este sitio, puesto que pese a ser el lugar más apropiado, presenta algunos problemas de luminosidad y pérdida en el rango de detección, con respecto al entorno ideal en el que se habían realizado pruebas, dado que el vehículo está constantemente cambiando sus condiciones dependiendo del lugar por el cual esté pasando, de forma que en ocasiones hubo picos inferiores de intensidad lumínica de hasta 40_{lx} .

Con base en lo anteriormente dicho, las zonas de captura y detección fueron movidas a la izquierda ligeramente, ya que en la zona extrema derecha se presentan problemas de luminosidad, dada la sombra interna que se produce y congestión, debido a que está enfocada directamente al interior del bus. Además, las zonas de decisión y contadores de *Down* y *Up* debieron ser intercambiadas, ya que para el prototipo anterior se planteó situar el DMC en la parte izquierda con respecto al ingreso de pasajeros al bus.

El primer paso para efectuar las pruebas del prototipo final, consistieron en validar los datos del conductor del vehículo donde se realizarán dichas pruebas, desde la interfaz login del DMC (Ilustración 180), para este caso en particular corresponde a la información del conductor 1, asignado al bus de placa GXS-326 perteneciente a la ruta 1, y desde la interfaz principal se inició el conteo de pasajeros como se puede observar en la Ilustración 181.



Ilustración 180 - Interfaz login DMC.

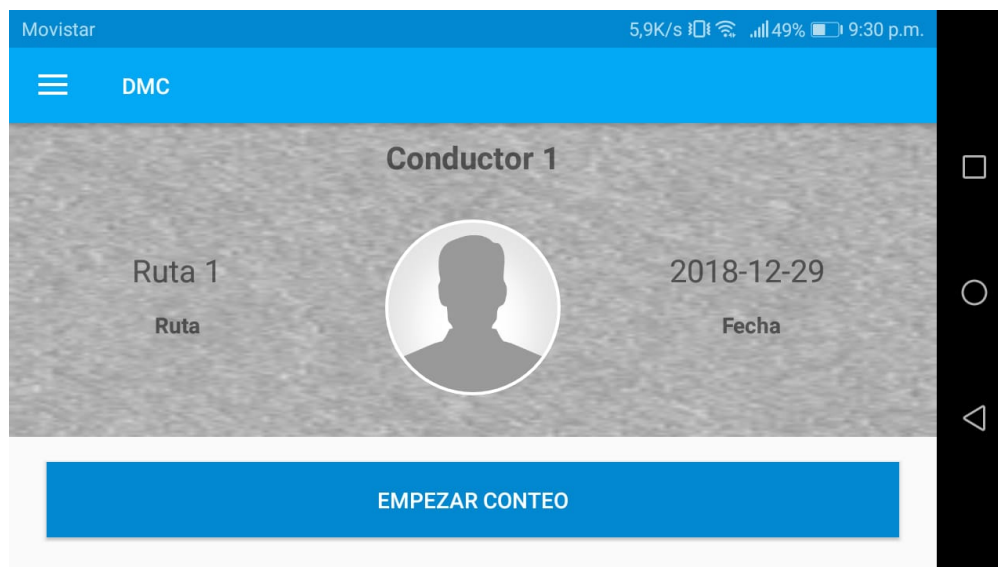


Ilustración 181- Interfaz principal DMC

Luego de iniciar el conteo e ingresar a la interfaz de detección y seguimiento, Ilustración 182, se puede comprobar que las primeras 7 zonas de captura se desplazaron a la izquierda levemente y la zona 8 ocupó la porción de pantalla restante. En dicha ilustración también se evidencia la detección dentro de la zona 8 y por consiguiente la activación de dicha zona, con lo cual se comprueba la correcta y efectiva integración entre el primer prototipo al constatar la validación de datos de sesión y el prototipo 2 con la detección del perfil.

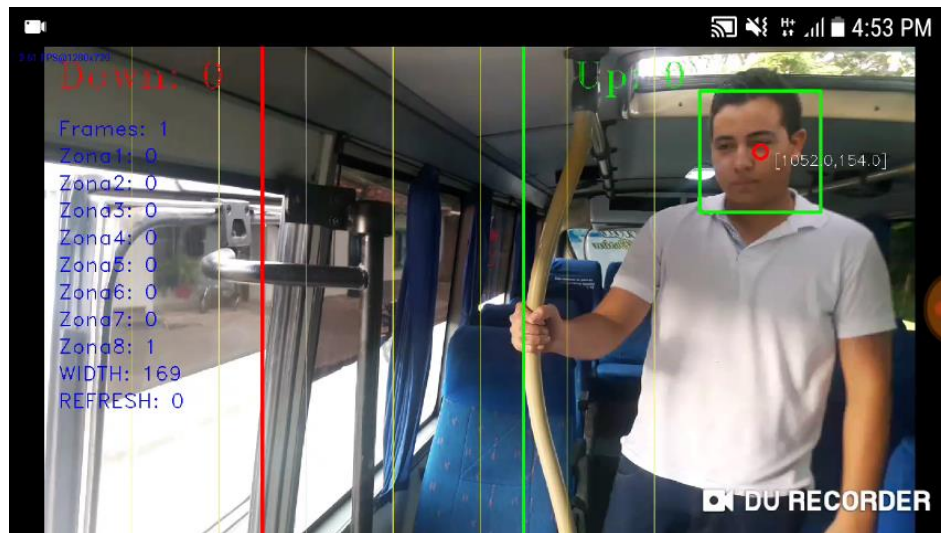


Ilustración 182 - Prueba final Down 1.

En la Ilustración 183, podemos observar el algoritmo de seguimiento en funcionamiento, donde la detección se ha desplazado hasta la zona 6, siguiendo el recorrido derecha-izquierda, en la Ilustración 184 se puede observar el recorrido más avanzado y próximo a contabilizarse en el contador Down.

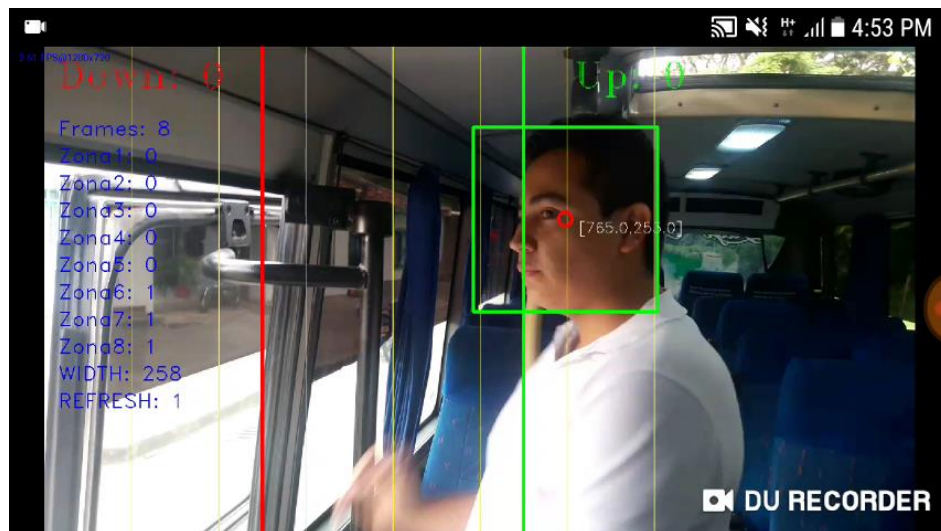


Ilustración 183 - Prueba final Down 2.

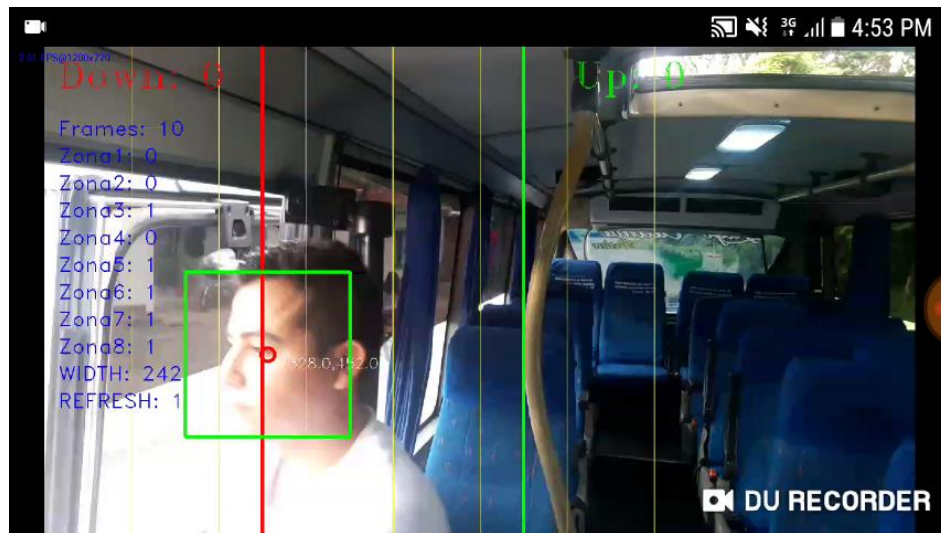


Ilustración 184 - Prueba final Down 3

Finalmente, en la Ilustración 185, al pasar la zona de decisión para el evento de descenso del vehículo se aumenta el contador Down y dicho evento es registrado en la base de datos como se observa en la Ilustración 186.

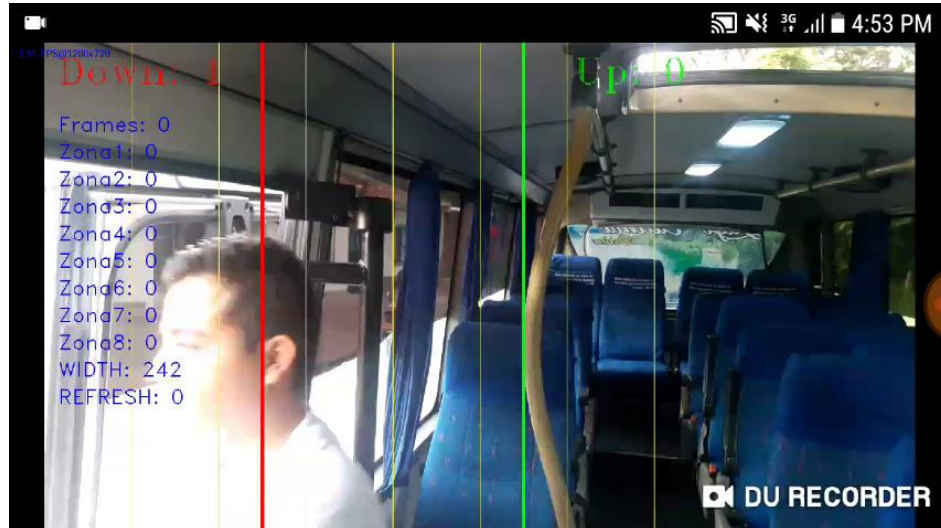


Ilustración 185 Prueba final Down 4

```

2018-12-29
├── boardings
│   ├── approachType: true
│   ├── lat: 3.896126
│   ├── lng: -76.30380
│   ├── parcialCount: 1
│   ├── time: "4:53:39"
│   └── toalCount: 1
├── busAssidned: "GXS-326"
├── parcialCount: 1
└── route: "Ruta 1"

```

Ilustración 186 - Registro en base de datos evento Down

También en la Ilustración 187, se puede observar el evento contrario, es decir el reconocimiento de un perfil derecho para una persona que ingresara al bus, se observa que se realiza el mismo proceso de seguimiento, activando las zonas 3, 4, 5 y 6 como se observa en la Ilustración 188, para cuando se pase el umbral registrar el evento de ascenso.

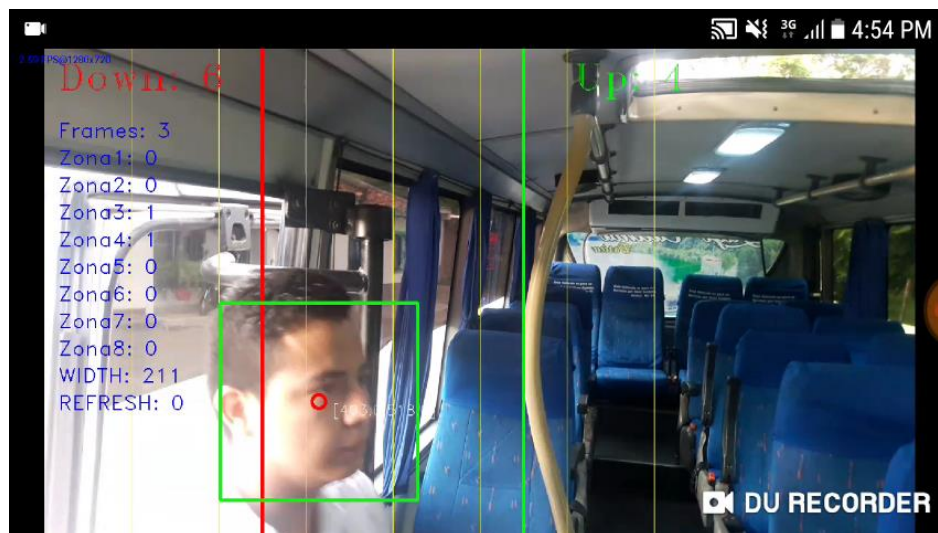


Ilustración 187 - Prueba final Up 1.

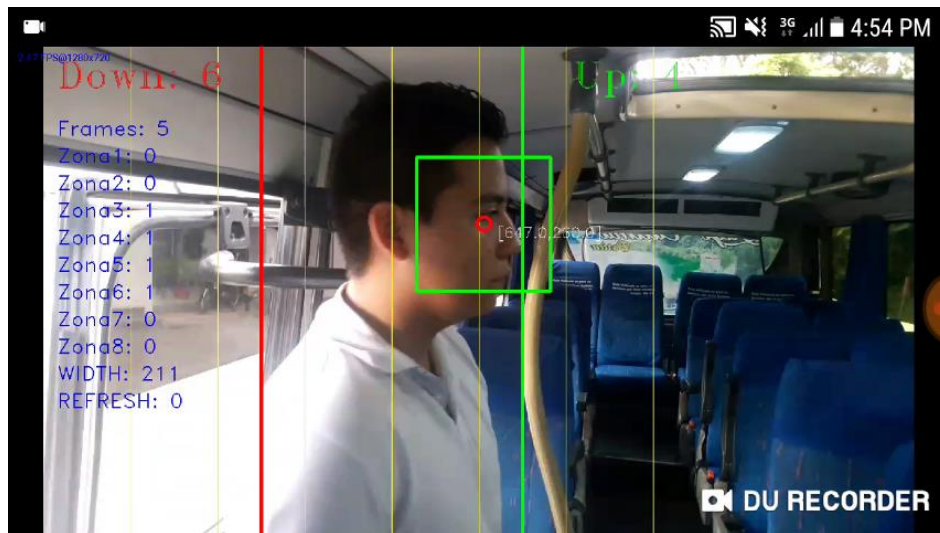


Ilustración 188 - Prueba final Up 2.

Se escogió el ascenso número 5 al bus en dicha prueba, ya que como se observa en Ilustración 189 al pasar el umbral, se detectan 2 objetos, uno correspondiente al perfil y otro ruido detectado en el mismo instante, donde se observa una de las facultades del algoritmo diseñado, ya que el seguimiento solo se realizó al objeto que se detectó en las diferentes zonas, por esto solo este registró el evento ascenso de dicho objeto.

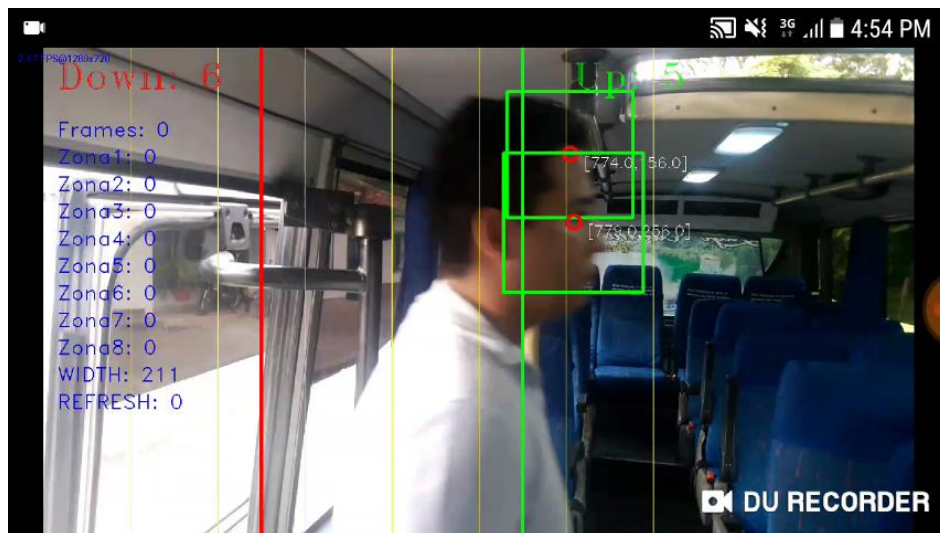


Ilustración 189 - Prueba final Up 3.

Para las anteriores pruebas se realizaron 52 ascensos y 52 descensos, con 3 personas diferentes, el día 29 de diciembre de 2018 entre las 4 pm y las 6 pm, obteniendo un éxito de captura de 39 eventos de ascenso y 46 eventos de descenso, equivalente al 75% y 88,46%

correspondientemente de los eventos realizados, es decir, un total de éxito del prototipo del 81,73%

4.3.3. Análisis de resultados

En el prototipo final se comprueba la correcta integración entre los 2 prototipos anteriores, combinando el registro y conexión a base de datos, junto con la visualización de datos capturados desde la AWGI, a partir de las capturas de eventos del flujo de pasajeros mediante la detección del perfil de una persona cuando esta se desplaza frente a la cámara del DMC.

Es importante resaltar que el 81,73% de fiabilidad demostrado por el prototipo final, con respecto a las pruebas realizadas en el prototipo 2 que arrojaron un 94%, se ven altamente influenciadas por el cambio de luz y rango de captura entre el ambiente ideal y el real. Para tener una fiabilidad más alta, considerando las condiciones del entorno, se infiere que controlar las condiciones lumínicas dentro del vehículo determinaría un ascenso a picos de hasta 95%, esto quiere decir que integrando ciertas conversiones en las capas de los fotogramas para no depender solo de la luz o controlando de forma constante la iluminación en la zona donde está ubicado el dispositivo y garantizar al menos 70_{lx} se puede llegar a obtener una fiabilidad de hasta un 95%, dado que una vez implementadas dichas soluciones tendrían que hacerse evaluaciones y casos de estudio más rigurosos para tener en cuenta todas las incidencias, como por ejemplo el paso de muchas personas al mismo tiempo, condiciones de lluvia extrema o situaciones en las que el vehículo no pueda proveer de la luz adecuada.

Cabe resaltar que la disponibilidad de los buses para la realización de las pruebas en el entorno real fue muy limitada, lo que no permitió indagar sobre el efecto que podría tener un reacondicionamiento de las condiciones lumínicas del vehículo en el desempeño del sistema, ni realizar pruebas en horas de la noche con luz artificial.

Capítulo 5

CONCLUSIONES Y TRABAJOS FUTUROS

5.1. Conclusiones

- Al evaluar las principales tecnologías que pueden brindar una solución a las problemáticas planteadas dentro del servicio TPC en la ciudad de Popayán, se seleccionó el desarrollo de una aplicación móvil nativa implementada para *Smartphones* con sistema operativo Android, ya que esta alternativa no requiere ninguna intervención por parte del usuario, ni un amplio montaje dentro de los buses.
- El sistema prototipo desarrollado es altamente escalable ya que constantemente se están lanzando dispositivos móviles, con más capacidad de procesamiento y mejores especificaciones, a menor costo, con lo cual se puede mejorar los algoritmos diseñados, implementando métodos y funciones más complejas, tales como la inclusión de algoritmos de *machine learning*.
- El método de clasificadores Haar-cascade para el análisis de imágenes y reconocimiento, presentó una solución efectiva a la hora de detectar un objeto específico, debido a que realizó el reconocimiento con un costo moderado de procesamiento, en el cual se alcanzaban umbrales de FPS aceptables, en el rango de 4.
- Luego de diversas pruebas, se comprobó que el mejor método para detectar un objeto muy específico como el perfil de una persona que está ascendiendo o descendiendo de un bus, es utilizar las herramientas brindadas por OpenCV para entrenar un

clasificador propio, que detecte el objeto específico deseado, y no intentar adaptar un de los clasificadores brindado por la librería, ya que estos son para objetos muy específicos y limitados.

- El rendimiento de la aplicación móvil se ve altamente reducido si se utiliza más de una cascada de clasificadores, dado que los FPS tuvieron una caída a 0,5, ocasionando errores en detección, por esto se utilizó uno solo, haciendo uso de un algoritmo que se vale del efecto espejo de los *frames* para detectar ambos lados de perfiles,
- Se encontró que la ubicación óptima para el DMC dentro del bus en el que se realizaron las pruebas, tanto por rango de captura, con un ángulo de 90 grados, limosidad con un promedio de 70_{lx} y éxito de detección, es en el lado derecho con respecto al ingreso de pasajeros, con una inclinación de 20° paralelo a la inclinación dada por las escaleras del bus para el ascenso o descenso, a una altura aproximada de 145 cm del piso y a una distancia del paso de pasajeros de 70 cm.
- El prototipo 1 logró una fiabilidad del 100% ya que no se presentaron errores en su funcionamiento durante todas las pruebas que se le realizaron, cumpliendo ampliamente con las expectativas deseadas.
- El prototipo 2 luego de 100 pruebas de funcionamiento en un ambiente ideal y simulado, demostró un 94% de fiabilidad en la detección y seguimiento de personas, con lo cual se comprobó el grado de éxito de la aplicación y algoritmo diseñados, además de la eficacia de los detectores Haar-cascade de OpenCV; con lo cual se cumplen las expectativas para la aplicación desarrollada, teniendo en cuenta que el control de la intensidad lumínica es el aspecto más crítico a tener en cuenta, el cual determina en gran escala la fiabilidad del conteo por reconocimiento de imagen.
- La fiabilidad del 81,73% en la detección, y el 100% de registro de los datos capturados por parte del DMC, es aceptable, dada la variabilidad de la intensidad lumínica en el bus ya que el análisis de imágenes, detección y *tracking* de objetos son campos muy amplios que implican un alto grado de desarrollo y estudio para su correcto uso.

5.2. Lecciones Aprendidas

Del desarrollo del presente trabajo de grado, surgieron las siguientes lecciones aprendidas:

- Cuando se realiza un proyecto que involucre análisis de imágenes, se recomienda realizar las pruebas, teniendo en cuenta todos los factores que acarrea la implementación en el entorno real, ya que, de no ser tenidos en cuenta, es posible que se presente un retraso en el desarrollo.
- La selección de una tecnología teniendo como uno de sus criterios de selección una amplia documentación, evita el estancamiento en implementación, ya que se encuentran salidas viables a cada uno de los retos.

5.3. Trabajos futuros

Como parte de los trabajos futuros, se plantean:

- Estudiar los conceptos de *Machine Learning* para mejorar el proceso de detección del objeto deseado.
- Refinar el algoritmo de seguimiento para incrementar la fiabilidad del conteo.
- Adaptar el prototipo actual a otros ambientes en los cuales se desee realizar el conteo de personas y tener una herramienta que permita la gestión de los datos capturados. Por ejemplo, el entorno de los aeropuertos es un ambiente en el cual es de importancia conocer el flujo de pasajeros y tener la opción de ver dicha información en una herramienta Web. Una de las ventajas es que como el anterior, es un ambiente mucho más controlado que un bus de servicio público, en donde incluso el movimiento del mismo afecta en la detección.
- Evaluar la viabilidad de implementar el concepto matriz origen destino, en el cuál, se obtiene la localización de ingreso y salida de una persona en específico. Es decir,

tener el conocimiento de quién se subió y en qué momento se bajó; esta información estadística sería de inmensa ayuda para cualquier tipo de sistema que requiera el conteo de flujo de personas.

- A partir de los datos capturados por el DMC, agregar a la AWGI, nuevas funcionalidades que otorguen mayor cantidad de filtros y formas de visualizar la información. Por ejemplo, tener la posibilidad de filtrar por días específicos, agregar graficas cruzadas con información permitiendo la visualización directa de estadísticas.

Referencias

- [1] «Revista Unimar Número 56». [En línea]. Disponible en: <http://www.umariana.edu.co/RevistaUnimar/publicaciones/RevistaUnimar56.html#/48/>. [Accedido: 14-ene-2019].
- [2] R.- ASALE, «comunidad», *Diccionario de la lengua española*. [En línea]. Disponible en: <http://dle.rae.es/?id=A5NKSVv>. [Accedido: 07-mar-2018].
- [3] J. A. Lupano y R. Sánchez, «Políticas de movilidad urbana e infraestructura urbana de transporte», dic. 2008.
- [4] Eduardo Alcántara Vasconcellos, «Análisis de la movilidad urbana Espacio, medio ambiente y equidad». CAF, 2010.
- [5] Steer Davies Gleave, «Informe 4 Diagnostico Parte II», *Scribd*, 2018. [En línea]. Disponible en: <https://es.scribd.com/document/347342477/150706-Informe-4-Diagnostico-Parte-II>. [Accedido: 28-may-2017].
- [6] M. A. Cuervo-Arango, «La calidad de vida. Juicios de satisfacción y felicidad como indicadores actitudinales de bienestar», *Rev. Psicol. Soc.*, vol. 8, n.º 1, pp. 101-110, ene. 1993.
- [7] «¿Qué es una Ciudad Sostenible?» [En línea]. Disponible en: https://www.findeter.gov.co//publicaciones/_que_es_una_ciudad_sostenible_publicacion. [Accedido: 07-mar-2018].
- [8] «POPAYAN CITY: Movilidad urbana». [En línea]. Disponible en: <http://popayancity.blogspot.com.co/2008/06/movilidad-urbana.html>. [Accedido: 07-mar-2018].
- [9] «Popayán inició proceso para transformarse en una ciudad Sostenible y Competitiva con apoyo del BID», *Radio Super Popayán*, 09-ago-2016. .
- [10] «Popayán será una ciudad sostenible y competitiva». [En línea]. Disponible en: <http://www.popayan.gov.co/ciudadanos/sala-de-prensa/noticias/Popay%C3%A1n-ser%C3%A1-una-ciudad-sostenible-y-competitiva>. [Accedido: 14-sep-2018].
- [11] «DLE: globalización - Diccionario de la lengua española - Edición del Tricentenario». [En línea]. Disponible en: <http://dle.rae.es/?id=JFCXg0Z>. [Accedido: 07-mar-2018].
- [12] Movilidad Futura S.A.S, «Estudio y Diagnóstico de la situación Actual del Transporte Público Colectivo». 2014.
- [13] «El SETP, un proyecto de sostenibilidad ambiental para Popayán | Movilidad Futura S.A.S - Sistema Estratégico de Transporte Público de Popayán». [En línea]. Disponible en: http://movilidadfutura.gov.co//debe_conocer/el-setp-un-proyecto-de-sosteni/. [Accedido: 14-ene-2019].
- [14] «Desafíos y Retos Movilidad Futura», *StarTic*. [En línea]. Disponible en: <http://www.unicauca.edu.co/starttic/index.php/retos-movilidad/>. [Accedido: 31-may-2017].
- [15] «Conteo de personas con un sensor RGBD comercial - ScienceDirect». [En línea]. Disponible en:

- <http://www.sciencedirect.com/science/article/pii/S1697791214000363>. [Accedido: 08-may-2017].
- [16] B. Hemangi y K. Nikhita, «PEOPLE COUNTING SYSTEM USING RASPBERRY PI WITH OPENCV», vol. 02, n.º 01, p. 5, 2016.
- [17] Andrei L. Yepes, «Diseño de un sistema automático para el control y acceso de persona a un autobús intermunicipal de transporte en Colombia». [En línea]. Disponible en: <http://biblioteca.usbbog.edu.co:8080/Biblioteca/BDigital/38480.pdf>. [Accedido: 05-feb-2019].
- [18] H. Gámez y L. Teresa, «Análisis de viabilidad para implementar la tecnología NFC aplicada a medios de pago en sistemas de transporte masivo, en Colombia», jul. 2014.
- [19] J. Penalva, «NFC: qué es y para qué sirve», *Xataka*, 25-ene-2011. [En línea]. Disponible en: <https://www.xataka.com/moviles/nfc-que-es-y-para-que-sirve>. [Accedido: 28-may-2017].
- [20] F. D. Garcia *et al.*, «Dismantling MIFARE Classic», en *Computer Security - ESORICS 2008*, 2008, pp. 97-114.
- [21] Alcaldía de Popayán, «Historia de Popayán». [En línea]. Disponible en: <http://www.popayan.gov.co/ciudadanos/popayan/historia>. [Accedido: 14-ene-2019].
- [22] DANE, «Proyecciones de población». [En línea]. Disponible en: <https://www.dane.gov.co/index.php/estadisticas-por-tema/demografia-y-poblacion/proyecciones-de-poblacion>. [Accedido: 14-ene-2019].
- [23] «Nuestra geografía». [En línea]. Disponible en: <http://www.popayan.gov.co/ciudadanos/popayan/nuestra-geografia>. [Accedido: 03-oct-2018].
- [24] «Territorios». [En línea]. Disponible en: <http://www.popayan.gov.co/ciudadanos/popayan/territorios>. [Accedido: 03-oct-2018].
- [25] Steer Davies Gleave, «Plan de Movilidad para el municipio de Popayán, Informe 3: Diagnóstico Parte I». 2015.
- [26] «Turistas a Popayán en Semana Santa», *Radio Super Popayán*, 05-abr-2018. .
- [27] C. Dangond Gibsone, J.-F. Jolly, A. Monteoliva Vilches, y F. Rojas Parra, «Algunas reflexiones sobre la movilidad urbana en Colombia desde la perspectiva del desarrollo humano», *Pap. Polit.*, vol. 16, n.º 2, pp. 485-514, dic. 2011.
- [28] Ministerio de Transporte, «Parque Automotriz Colombia 2018». 14-ene-2019.
- [29] «Popayán, Colombia: Tu Guía de Transporte Público | Moovit». [En línea]. Disponible en: https://moovitapp.com/index/es-419/transporte_p%C3%BAblico-Popay%C3%A1n-3463. [Accedido: 14-ene-2019].
- [30] V. Coskun, K. Ok, y B. Ozdenizci, *Near Field Communication (NFC): From Theory to Practice*. John Wiley & Sons, 2011.
- [31] «BBVA-Wallet-1920x0-c-f.jpg (1920x779)». [En línea]. Disponible en: <https://www.bbva.com/wp-content/uploads/2017/04/BBVA-Wallet-1920x0-c-f.jpg>. [Accedido: 14-ene-2019].
- [32] F. D. Garcia *et al.*, «Dismantling MIFARE Classic», presentado en Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security, 2008, pp. 97-114.

-
- [33] «Acerca de MIFARE». [En línea]. Disponible en: <https://www.mifare.net/es/acerca-de-mifare/>. [Accedido: 14-ene-2019].
- [34] «4k Smart Card», *indiamart.com*. [En línea]. Disponible en: <https://www.indiamart.com/proddetail/4k-smart-card-11632637962.html>. [Accedido: 14-ene-2019].
- [35] «Conocimiento | FQ Ingeniería Electrónica». [En línea]. Disponible en: <https://www.fqingenieria.com/es/conocimiento/tipos-de-chips-mifarer-52>. [Accedido: 14-ene-2019].
- [36] S. M. Sze y K. K. Ng, *Physics of Semiconductor Devices*. John Wiley & Sons, 2006.
- [37] «Definición de sensor», *Definiciones Tecnológicas*. [En línea]. Disponible en: <https://definicion.de/sensor/>. [Accedido: 14-ene-2019].
- [38] «What is a Raspberry Pi?», *OpenSource.com*. [En línea]. Disponible en: <https://opensource.com/resources/raspberry-pi>. [Accedido: 14-ene-2019].
- [39] Raspberry Pi (Trading) Ltd, «DATASHEET Raspberry Pi 3». 2015.
- [40] «Raspberry Pi 3 Model B», *Raspberry Pi*. [En línea]. Disponible en: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. [Accedido: 14-ene-2019].
- [41] «GitHub - WatershedArts/Footfall: Application that allows you to monitor the traffic in and out of your building, using the RPi Camera and openFrameworks». [En línea]. Disponible en: <https://github.com/WatershedArts/Footfall>. [Accedido: 14-ene-2019].
- [42] «Raspberry Pi Camera Module V2 :: Solarbotics». [En línea]. Disponible en: <https://solarbotics.com/product/52090/>. [Accedido: 14-ene-2019].
- [43] «Microsoft Kinect Sensor and Its Effect - IEEE Journals & Magazine». [En línea]. Disponible en: <https://ieeexplore.ieee.org/document/6190806>. [Accedido: 14-ene-2019].
- [44] «Figure 1. Kinect hardware (Zhang, 2012)», *ResearchGate*. [En línea]. Disponible en: https://www.researchgate.net/figure/Kinect-hardware-Zhang-2012_fig1_265126750. [Accedido: 14-ene-2019].
- [45] «La evolución de Kinect y la importancia real de Microsoft Research». [En línea]. Disponible en: <https://www.xatakawindows.com/xbox/la-evolucion-de-kinect-y-la-importancia-de-microsoft-research>. [Accedido: 31-oct-2018].
- [46] E. Quesada Díaz, «Estudio sobre el uso de sensores RGB-D para el conteo de personas y su caracterización», 2015.
- [47] *GUÍA PRÁCTICA DE SENSORES*. Creaciones Copyright SL, 2010.
- [48] Arturo Baz Alonso, María Álvarez Rodríguez, Rosana García Baniello, y Irene Ferreira Artime, «Dispositivos Móviles». .
- [49] «¿Qué fue del WAP?» [En línea]. Disponible en: <https://www.xatakamovil.com/conectividad/que-fue-del-wap>. [Accedido: 14-ene-2019].
- [50] «El negocio de las aplicaciones móviles continúa imparables: un 60% más de apps descargadas y el doble de ingresos durante 2017». [En línea]. Disponible en: <https://www.xatakamovil.com/aplicaciones/el-negocio-de-las-aplicaciones-moviles-continua-imparables-un-60-mas-de-apps-descargadas-y-el-doble-de-ingresos-durante-2017>. [Accedido: 14-ene-2019].

-
- [51] «El 99.6% del mercado móvil le pertenece a Android y iOS - PCMag Latam». [En línea]. Disponible en: <https://latam.pcmag.com/sistemas-operativos-moviles/18490/el-996-del-mercado-movil-le-pertenece-a-android-y-ios>. [Accedido: 14-ene-2019].
- [52] «Arquitectura de la plataforma | Android Developers». [En línea]. Disponible en: <https://developer.android.com/guide/platform/?hl=es-419>. [Accedido: 14-ene-2019].
- [53] Intellipaat, «iOS Architecture - iOS Tutorial | Intellipaat.com», *Intellipaat*. [En línea]. Disponible en: <https://intellipaat.com/tutorial/ios-tutorial/ios-architecture/>. [Accedido: 14-ene-2019].
- [54] P. L. Castro, «¿A qué se refieren con eso de escalabilidad?», *aboutespanol*. [En línea]. Disponible en: <https://www.aboutespanol.com/que-es-escalabilidad-157635>. [Accedido: 14-ene-2019].
- [55] J. Giacomantone *et al.*, «Generación de características, análisis de imágenes y reconocimiento de patrones con restricciones temporales», presentado en XVII Workshop de Investigadores en Ciencias de la Computación (Salta, 2015), 2015.
- [56] «fiabilidad | Definición de fiabilidad - Diccionario de la lengua española - Edición del Tricentenario». [En línea]. Disponible en: <https://dle.rae.es/srv/fetch?id=Hpsj999>. [Accedido: 14-ene-2019].
- [57] «Modelo vista controlador (MVC)». [En línea]. Disponible en: <https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html>. [Accedido: 14-ene-2019].
- [58] «Git - Acerca del control de versiones». [En línea]. Disponible en: <https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones>. [Accedido: 14-ene-2019].
- [59] «Build software better, together», *GitHub*. [En línea]. Disponible en: <https://github.com>. [Accedido: 14-ene-2019].
- [60] «¿Qué es Firebase? La mejorada plataforma de desarrollo de Google», *El Androide Libre*, 19-may-2016. .
- [61] «¿Qué es Firebase de Google? | OpenWebinars». [En línea]. Disponible en: <https://openwebinars.net/blog/que-es-firebase-de-google/>. [Accedido: 14-ene-2019].
- [62] «Angular - Architecture overview». [En línea]. Disponible en: <https://angular.io/guide/architecture>. [Accedido: 14-ene-2019].
- [63] «Angular - Services». [En línea]. Disponible en: <https://angular.io/tutorial/toh-pt4>. [Accedido: 14-ene-2019].
- [64] «Angular - Introduction to modules». [En línea]. Disponible en: <https://angular.io/guide/architecture-modules>. [Accedido: 14-ene-2019].
- [65] *Beautiful charts for Angular based on Chart.js. Contribute to valor-software/ng2-charts development by creating an account on GitHub*. Valor Software, 2019.
- [66] «Qué es una App Nativa | Blog de Tecnología Qode Apps». [En línea]. Disponible en: <https://www.qode.pro/blog/que-es-una-app-nativa/>. [Accedido: 14-ene-2019].
- [67] «¿Qué es una app nativa y una web app?» [En línea]. Disponible en: https://www.tendencias21.net/Que-es-una-app-nativa-y-una-web-app_a33476.html. [Accedido: 14-ene-2019].

-
- [68] «Aplicaciones híbridas: qué son y cómo se desarrollan y crean», *APLICACIONES MÓVILES*, 10-jun-2013. .
- [69] «Crear un proyecto», *Android Developers*. [En línea]. Disponible en: <https://developer.android.com/studio/projects/create-project?hl=es-419>. [Accedido: 05-feb-2019].
- [70] T. Rodríguez, «Kotlin ya es un lenguaje oficial en Android: ¿qué implicaciones tiene y por qué es tan importante?», *Xataka Android*, 19-may-2017. [En línea]. Disponible en: <https://www.xatakandroid.com/programacion-android/kotlin-ya-es-un-lenguaje-oficial-en-android-que-implicaciones-tiene-y-por-que-es-tan-importante>. [Accedido: 05-feb-2019].
- [71] P. Viola y M. Jones, «Rapid object detection using a boosted cascade of simple features», en *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, vol. 1, pp. I-I.
- [72] «Detecting faces in images: a survey - IEEE Journals & Magazine». [En línea]. Disponible en: <https://ieeexplore.ieee.org/abstract/document/982883>. [Accedido: 14-ene-2019].
- [73] B. Mathew, A. Davis, y R. Evans, «A characterization of visual feature recognition», en *2003 IEEE International Conference on Communications (Cat. No.03CH37441)*, 2003, pp. 3-11.
- [74] «Impacto de la memoria cache en la aceleración de la ejecución de algoritmo de detección de rostros en sistemas empotrados». [En línea]. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59282012000200008. [Accedido: 14-ene-2019].
- [75] «OpenCV: Face Detection using Haar Cascades». [En línea]. Disponible en: https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html. [Accedido: 14-ene-2019].
- [76] Y. Freund y R. E. Schapire, «A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting», *J. Comput. Syst. Sci.*, vol. 55, n.º 1, pp. 119-139, ago. 1997.
- [77] «Cascade Classification — OpenCV 2.4.13.7 documentation». [En línea]. Disponible en: https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html#cascadeclassifier. [Accedido: 14-ene-2019].
- [78] «Cascade Classifier Training — OpenCV 2.4.13.7 documentation». [En línea]. Disponible en: https://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html. [Accedido: 14-ene-2019].
- [79] Ú. modificación: null, «A well dressed young man looking up at copy space», *iStock*. [En línea]. Disponible en: <https://www.istockphoto.com/es/foto/mirando-at-us-ideas-gm175181327-21889626>. [Accedido: 14-ene-2019].
- [80] *Open Source Computer Vision Library. Contribute to opencv/opencv development by creating an account on GitHub*. OpenCV, 2019.
- [81] «Angular - Routing & Navigation». [En línea]. Disponible en: <https://angular.io/guide/router>. [Accedido: 14-ene-2019].

Anexo A: Cobertura de las empresas del servicio de TPC

A continuación, se muestran las coberturas de las 4 empresas prestadoras del servicio de transporte público colectivo de la ciudad de Popayán.

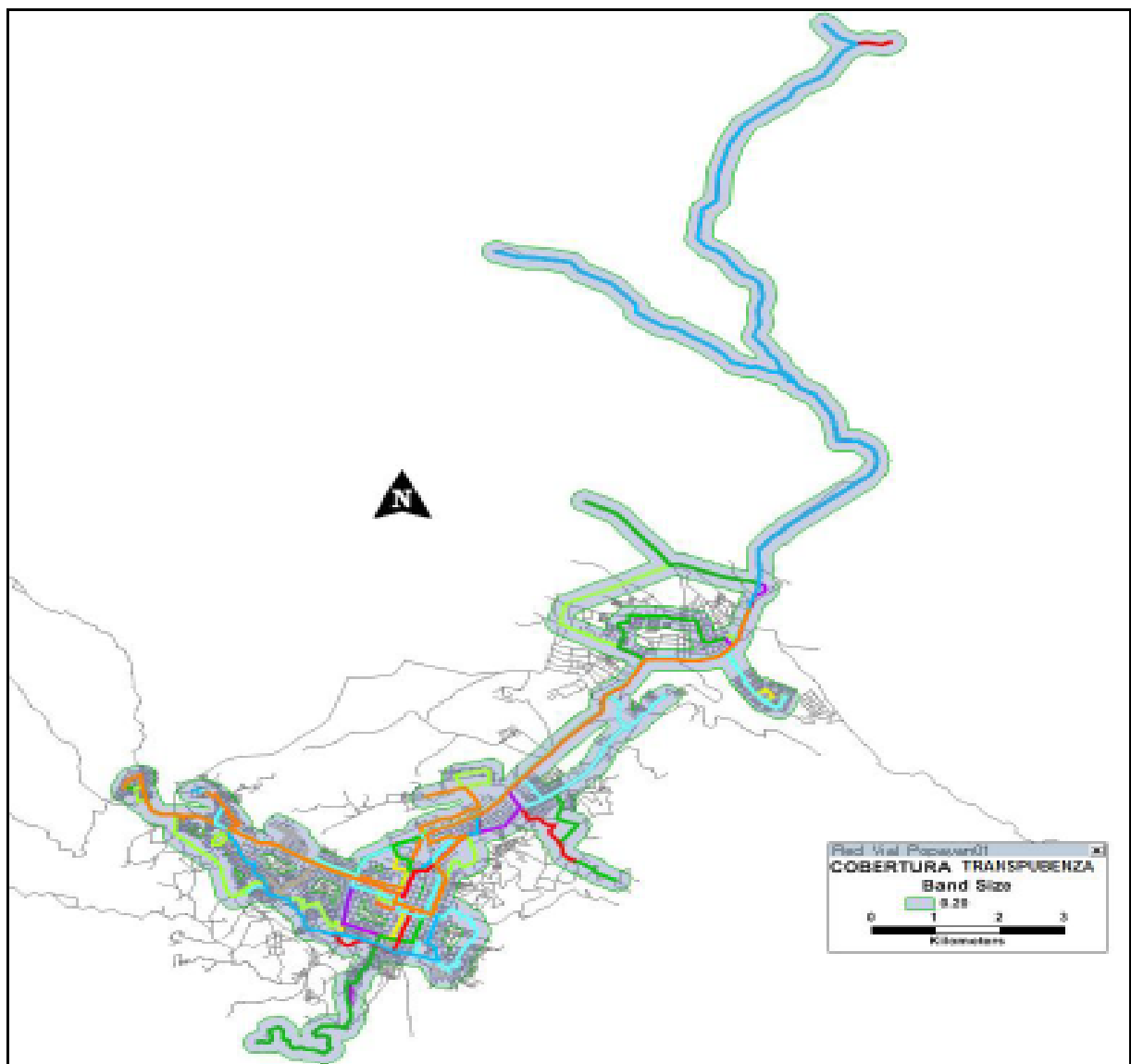


Ilustración 190 - Cobertura Empresa Transpubenza
Tomado de: [12]

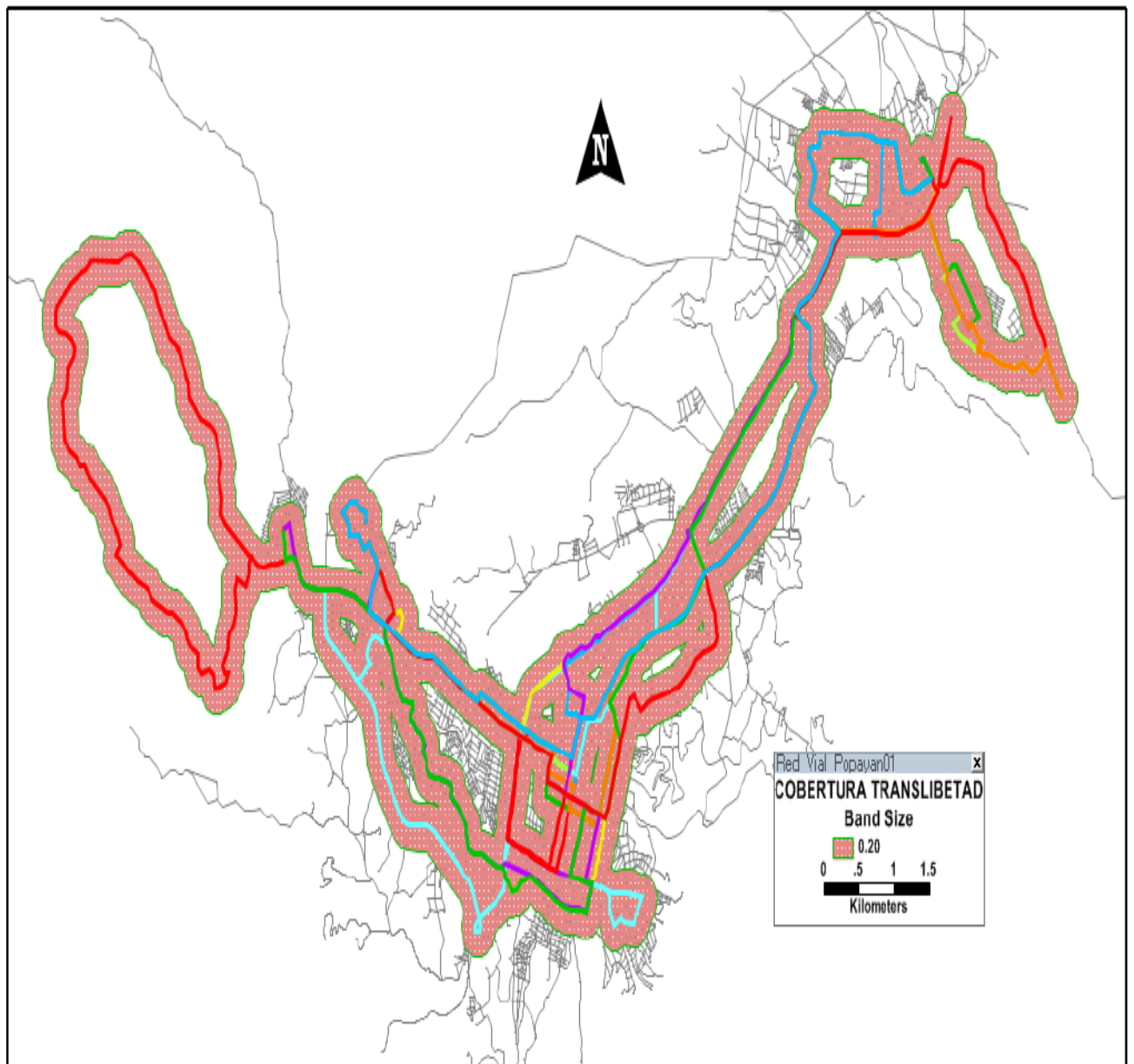


Ilustración 191 - Cobertura Empresa Translibertad
Tomado de: [12]

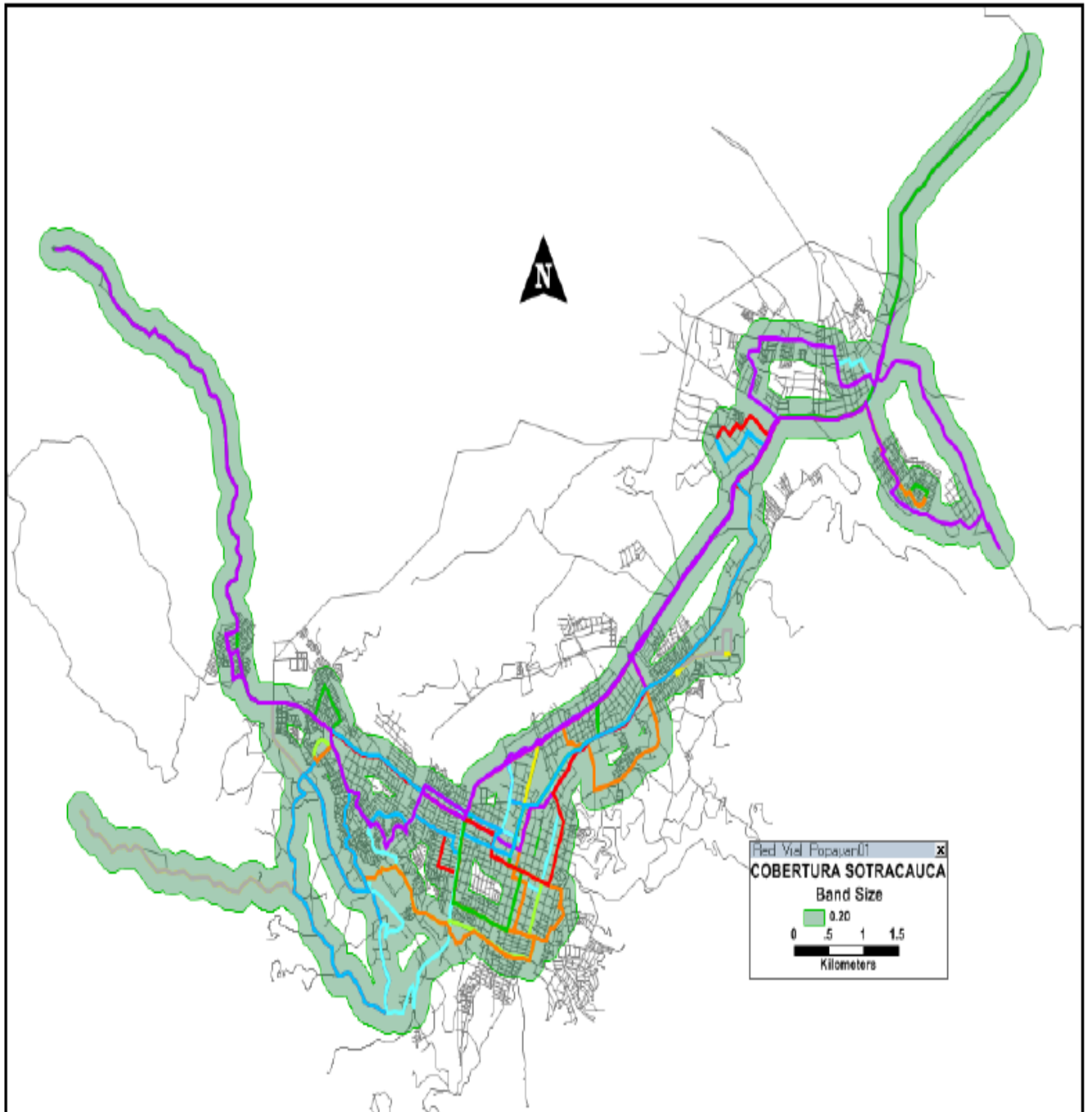


Ilustración 192 - Cobertura Empresa Sotracauca Metro
Tomado de: [12]

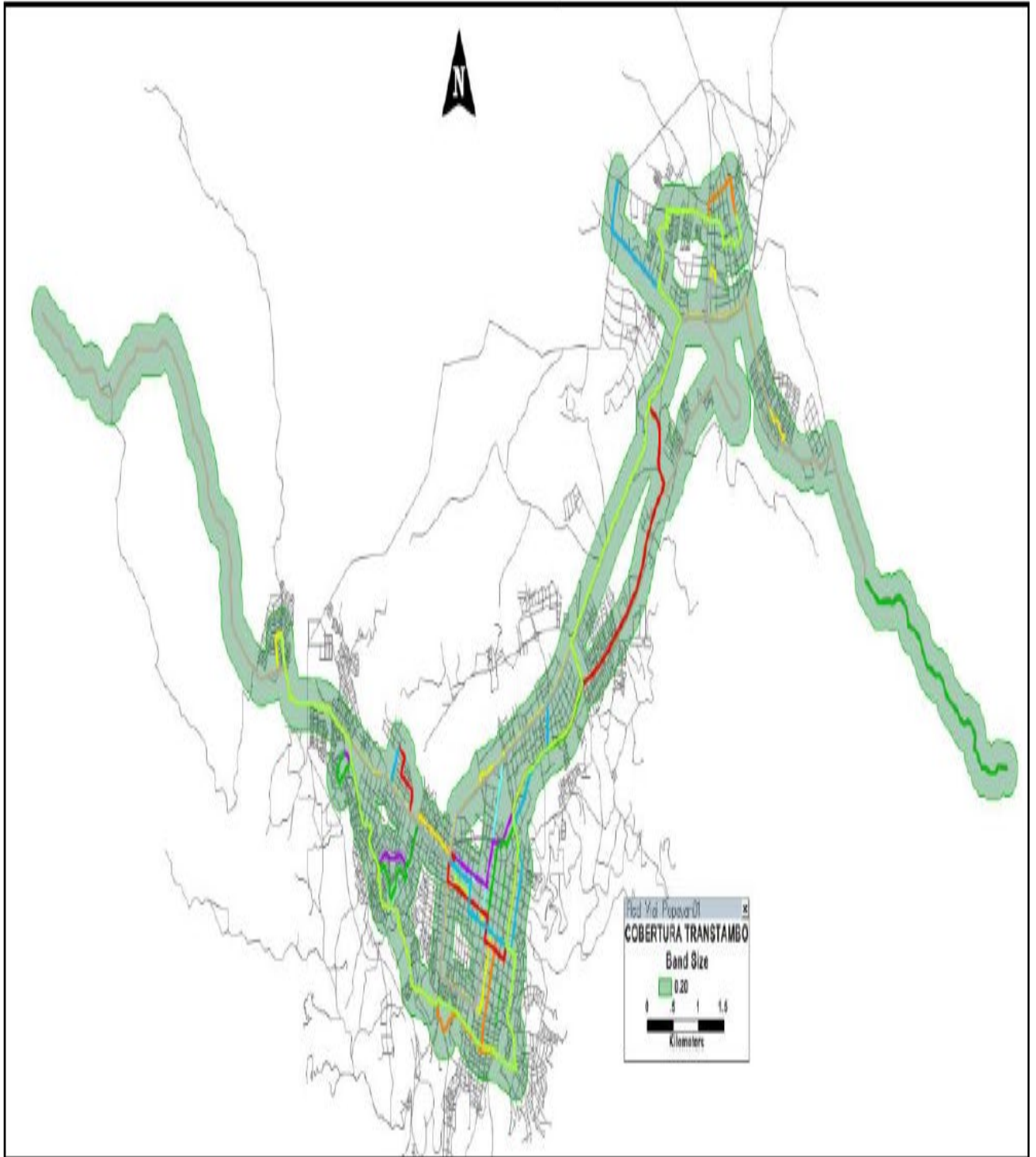


Ilustración 193- Cobertura Empresa Transtambo
Tomado de: [12]

Anexo B: Sistema de control de versiones

Los principales comandos usados para el control de versiones se describen a continuación:

- git init: comando usado para crear un nuevo repositorio git.
- git add: Usado para agregar archivos a seguimiento.
- git clone: Se usa para “descargar” un repositorio remoto en una carpeta local
- git commit: Este comando cambia la cabecera, es decir guarda un punto de recuperación del proyecto.
- git status: Muestra la lista de archivos que se han cambiado junto con los archivos que están por ser añadidos o comprometidos.
- git remote: Se usa para conectar un repositorio local con uno remoto.
- git pull: Este comando fusiona los cambios que se han hecho en el repositorio local trabajado.
- git push: Sube los cambios del repositorio local al repositorio remoto.
- git reset: Resetea la cabecera del repositorio, eliminando los commits posteriores al punto sobre el cual se está ejecutando el comando.

El proyecto se puede encontrar en los siguientes enlaces directos a sus repositorios:

Aplicación Web (AWGI):

<https://bitbucket.org/Jucapo/hegi/src/master/>

Aplicación Móvil (DMC):

https://github.com/ricardovejarano/mobile_aplication_hegi

Anexo C: Creación de componentes

Una vez generado el proyecto mediante la CLI de Angular, se procede con la instalación de las dependencias requeridas por medio del comando *npm install*. Dicho comando debe ser ejecutado en *PowerShell*²⁰, ubicados dentro de la carpeta del proyecto creado. Cada que una nueva dependencia es agregada al proyecto, es necesario re lanzar el comando *npm install*.

En la Ilustración 194 se observa la totalidad de componentes creados, en donde todas las carpetas dentro de 'src/app/components', corresponden a todas las pantallas descritas en el apartado de diseño.

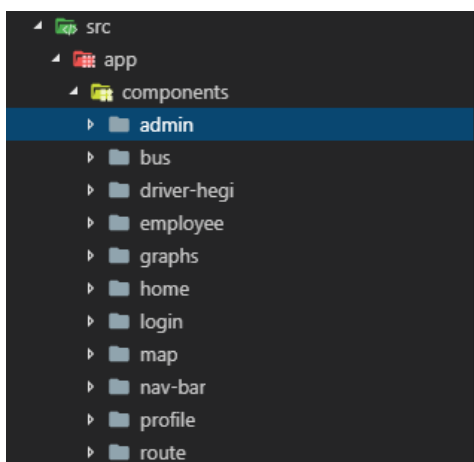


Ilustración 194 - Estructura de carpetas AWGI

Cada componente generado consta de tres archivos con el mismo nombre de carpeta y de extensiones *.css*, *.html*, *.ts*. El archivo con extensión *.css* corresponde a la hoja de estilos del componente, el *.html* es el archivo que contiene el lenguaje de marcas de hipertexto y finalmente el *.ts* es el archivo que gestiona las plantillas del *html*.

La carpeta *nav-bar* corresponde a la barra de navegación y, dado que es un componente que se hace presente en prácticamente todas las vistas (a excepción de *login*), es necesario

²⁰ Interfaz de consola con posibilidad de escritura y unión de comandos por medio de instrucciones

reusar su vista incluyendo su *selector*, Ilustración 195, en el archivo *app.component.ts*, Ilustración 196.

```
@Component({
  selector: 'app-nav-bar',
  templateUrl: './nav-bar.component.html',
  styleUrls: ['./nav-bar.component.css']
})
```

Ilustración 195 - Selector nav-bar component

```
@Component({
  selector: 'app-root',
  template: `
    <app-nav-bar></app-nav-bar>
    <router-outlet></router-outlet>
  `,
  styleUrls: ['./app.component.css']
})
```

Ilustración 196 - Template app.component.ts

Con lo anterior, cada una de las vistas tendría incluida tanto la página de *nav-bar* como la que se inyecte por medio del *router-outlet*.

Para hacer la distinción de la vista del *login* y ocultar ahí el *nav-bar*, se hace uso de un observable del tipo *BehaviorSubject* puesto que la renderización de los componentes ocurre solo una vez al ser inicializado el archivo *app.component.ts* y es necesaria una posterior suscripción a dicho observable para indicarle al template que debe o no aparecer. La definición, evento de cambio y suscripción de un observable del tipo *Behavior Subject* en Angular se muestra en la Ilustración 197, Ilustración 198 e Ilustración 199.

```
activated = new BehaviorSubject<boolean>(false);
```

Ilustración 197 - Definición de un Behavior Subject

```
this.activated.next(true);
```

Ilustración 198 - Evento de cambio para un Behavior Subject

```
this.homeService.activated.subscribe((res: boolean) => {
  if (res) {
    setTimeout(() => {
      this.valval = this.homeService.totalCounPassegers;
      this.temp = this.valval;
      this.assss(this.temp);
      this.homeService.activated.next(false);
    }, 1000);
  }
});
```

Ilustración 199 - Suscripción a un Behavior Subject

Anexo D: Routing

El *routing* o enrutador de Angular interpreta una URL del navegador como una instrucción para navegar a una vista generada por el cliente. Se pueden incluso pasar parámetros junto con el componente de vista. El servicio de enrutamiento es de uso opcional en Angular y no se encuentra disponible desde las librerías de *core* de Angular sino, desde el paquete *@angular/router*, Ilustración 200.

```
import { RouterModule, Routes } from '@angular/router';
```

Ilustración 200 - Importar módulo de routing

Cuando se agrega el módulo de routing, la aplicación debe tener una instancia única para el servicio de enrutamiento. Cuando cambia la URL del navegador, ese enrutador busca un equivalente en su instancia, el cual determina el componente a mostrar [81]. Cada uno de los componentes debe tener un archivo de la clase routing en la cual se indica principalmente cual es el *path* de URL que va a concordar y cuál es el componente que va a ser mostrado, por ejemplo, en la Ilustración 201, se muestra el contenido del *routing* en el componente “*profile*”.

```
import { Routes } from '@angular/router';
import { ProfileComponent } from './profile.component';
import { AuthGuard } from '../guards/auth.guard';
import { ChagePasswordComponent } from './chage-password/chage-password.component';
import { UpdateInfoComponent } from './update-info/update-info.component';
import { ImageProfileComponent } from './image-profile/image-profile.component';

export const profileRoute: Routes = [
  {path: 'profile', component: ProfileComponent, children: [
    {path: '', component: UpdateInfoComponent },
    {path: 'change-pass', component: ChagePasswordComponent },
    {path: 'change-image-profile', component: ImageProfileComponent },
  ]}, canActivate: [AuthGuard]]
];
```

Ilustración 201 - Archivo de enrutamiento para el componente de perfil de usuario

Una de las ventajas de usar el enrutamiento de Angular es que se identifican claramente las jerarquías en las rutas de navegación de la aplicación. Para el caso mostrado en la anterior ilustración, se observa que el *path* principal es ‘*profile*’, el cual llama el componente *ProfileComponent*. Cabe resaltar que cuando un componente tiene *paths* del tipo *children*, este debe permitir la inyección de componentes incluyendo la etiqueta *router-outlet* en su

template. Así, cuando se navegue al *path* 'profile', este redirecciona al componente *ProfileComponent*, el cual tiene un *router-outlet* de inyección, por lo cual es inmediatamente redirigido al primer componente con *path* vacío, en este caso *UpdateInfoComponent*.

Anexo E: Persistencia

La persistencia se basa en el almacenamiento de objetos del tipo llave valor, cada uno de ellos puede ser recuperado en cualquier componente de la aplicación, para el caso del presente documento el objeto de persistencia que ayuda con el control de la sesión, cambia de estado en el flujo de *login* → *home* al iniciar la sesión y cuando se cierra sesión.

Para esto, se hace uso de la utilidad *LocalStorage*, en angular se usa como se muestra en la Ilustración 202.

```
localStorage.setItem('logged', 'true');
```

Ilustración 202 - LocalStorage en Angular

Donde *logged* es la llave o identificador con la cual se va a llamar luego dicho objeto y *true* es el valor. Para recuperar el objeto de la ilustración anterior, se debe igualar una variable a la invocación de la Ilustración 203.

```
localStorage.getItem('logged')
```

Ilustración 203 - Recuperar valor LocalStorage

Entonces, *logged* verifica el valor de verdad indicando al sistema la vista que tiene que presentar, donde un “estado” de *false* dirige al *login*, en caso contrario a *home*, dicha evaluación se realiza en el *AppModule*.

En el momento en el que el usuario cierra su sesión, el item de llave *'logged'*, toma el valor de verdad *false*.

Anexo F: Búsqueda por rango de días

El proceso de búsqueda para un rango de fechas tiene el mismo flujo que el estudiado en la sección de búsqueda para un solo día. Aquí, se añade la selección de una fecha final, la cual, al no ser nula en la función de evaluación, Ilustración 204, determina el flujo de codificación por rango de fecha.

```

if (graphForm.value.dateF === null || graphForm.value.dateF === '') {
  this.nameCsvFile = this.plaque + ' - ' + this.route + ' - '
    + graphForm.value.dateI;
  this.getCounterInDay(this.plaque);
  this.flagChart = true;
  this.flagChart2 = false;
} else {
  this.getArrayDays(graphForm.value.dateI, graphForm.value.dateF);
  setTimeout(() => {
    this.getCounterInRange(this.plaque);
    this.nameCsvFile = this.plaque + ' - ' + this.route
      + ' - ' + graphForm.value.dateI + ' - ' + graphForm.value.dateI;
  }, 1000);
}

```

Ilustración 204 - Decisión de búsqueda por rango o diaria

Lo primero que se debe hacer es generar un *array* que contenga todos los días que correspondan al rango de fechas ingresadas en el *template* con el formato aaaa-mm-dd, ya que con dicho formato el DMC realiza las inserciones. Los parámetros que se necesitan para la realización del algoritmo que construya dicho *array*, Ilustración 205, son la fecha de inicio y la fecha final.

```

getArrayDays(start, end) {
  this.arrayDays = [];
  const one_day = 1000 * 60 * 60 * 24;
  this.startDate = new Date(start).getTime();
  this.endDate = new Date(end).getTime();
  for (let i = this.startDate; i <= this.endDate; i = i + one_day) {
    // tslint:disable-next-line:radix
    const date123 = new Date(parseInt(i.toString())).toISOString();
    this.allDate = date123.split('T');
    this.arrayDays.push(this.allDate[0]);
  }
}

```

Ilustración 205 - Algoritmo para generar array de fechas en rango de días

En primer lugar, se crea la constante *one_day* con el valor de un día en milisegundos y se calculan también los valores en milisegundos de las dos fechas de entrada a la función. Luego, se crea un ciclo *for* que empieza en el valor calculado para la fecha inicial, y termina en la fecha final (en milisegundos), dicho ciclo aumente de día en día. En cada iteración del ciclo, refresca una constante que contiene la conversión necesaria para generar cada uno de los días que pertenecen al rango, de forma que se va llenando *arrayDays*.

Una vez es llenado el array del rango de días, se procede a consumir la misma función de la búsqueda por día, en donde se encuentran todos los abordajes bajo la lista “*counter*”. A continuación, se realiza una adaptación de la función que se visualiza en la Ilustración 69, ya que ahora no se tiene un día en específico sino un array de varios de ellos, por lo cual es necesario realizar una iteración dentro del ciclo *for* ya existente. La función adaptada se puede ver en la Ilustración 206.

```
existDateRange() {
  let coin = 0;
  for (let x = 0; x < this.counterList.length; x++) {
    for (let y = 0; y < this.arrayDays.length; y++) {
      if (this.counterList[x][`${this.arrayDays[y}`}]) {
        if (this.plaque === this.counterList[x][`${this.arrayDays[y}`}][`busAssigned`]) {
          coin = 1;
          this.getApproachInRange(x);
          x = this.counterList.length - 1;
        }
      }
    }
  }
  if (coin === 0) {
    window.alert('No se encuentran resultados');
    this.graphsService.chartSubjetc.next(true);
  }
}
```

Ilustración 206 - Adaptación de función para encontrar índice de coincidencia con iteración

Se observa que ahora dentro del ciclo del vector que contiene las listas de abordajes, se hace una nueva iteración con el vector encontrado con el algoritmo de fechas por rango, de manera que cuando se encuentre una coincidencia, se guarda el índice de la lista y se consume la función *getApproachInRange(x)*, donde *x* es el índice de coincidencia encontrado.

La función que va a traer cada una de las listas de abordajes por días, necesita consumir un servicio, el cual a partir de la llave de inserción y fecha unitaria, retorna una lista de abordaje correspondiente al día, Ilustración 207.

```
getBoardings(key: string, date: string) {
  return this.afDatabase.list(`counter/${key}/${date}/boardings`);
}
```

Ilustración 207 - Servicio para obtener listas de abordajes

```

getApproachInRange(indice: number) {
  console.log('ArrayDates: ', this.arrayDays);

  for (let y = 0; y < this.arrayDays.length; y++) {
    const key = this.counterList[`${indice}`][`${y}`];
    setTimeout(() => {
      this.graphsService.getBoardings(key, this.arrayDays[y])
        .snapshotChanges().subscribe(item => {
          this.boardingsList = [];
          item.forEach(element => {
            const x = element.payload.toJSON();
            x['key'] = element.key;
            this.listBoardingsInDay.push(x as Approach);
            this.datesForTable.push(this.arrayDays[y]);
          });
          if (this.listBoardingsInDay[this.listBoardingsInDay.length - 1]) {
            this.arrayTemp.push(this.listBoardingsInDay
              [this.listBoardingsInDay.length - 1]['totalCount']);
          }
          this.graphsService.chartSubjetc.next(true);
        });
    }, 500);
  }
}

```

Ilustración 208 - Función para obtener abordajes por rango de días 1/2

En la Ilustración 208 se observa que la codificación empieza con un ciclo *for* del tamaño del vector de días, y a continuación, se recupera la llave de inserción (la cual no varía) que es enviada como parámetro a la función del servicio junto con la iteración de un día dentro del vector de rango de días. Para entender las variables que participan en el fragmento de código descrito, se muestra su estructura en la base de datos de firebase mediante la Ilustración 209.

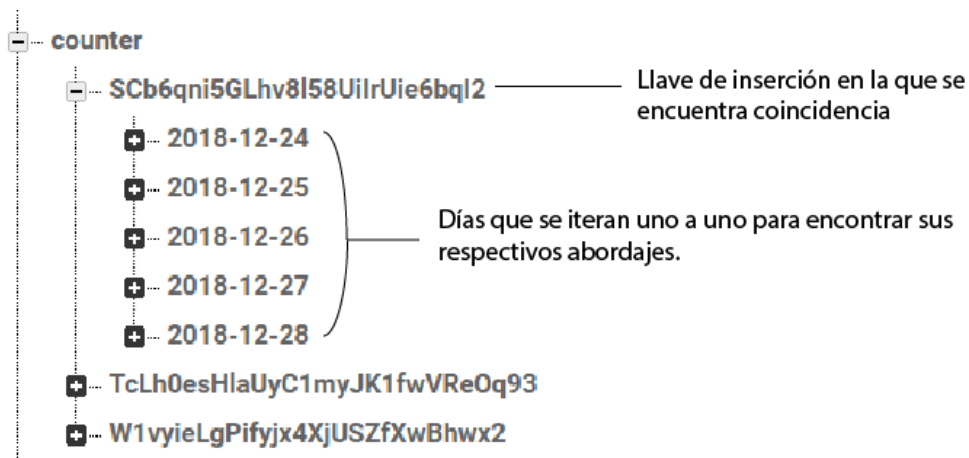


Ilustración 209 - Lista de coincidencia dentro de counter

Al igual que en la búsqueda por día, las variables *listBoardingsInDay* y *datesForTable*, son necesarias para la muestra de la información en la tabla del *template*. Al final del *forEach* de la suscripción se evalúa si en dicha iteración existe *listBoardingsInDay* y se asigna su parámetro *totalCount* en la última posición al vector temporal *arrayTemp*. Lo anterior se realiza de esa manera puesto que la última posición tiene la cantidad total de abordajes, tal y como se muestra en la Ilustración 210.

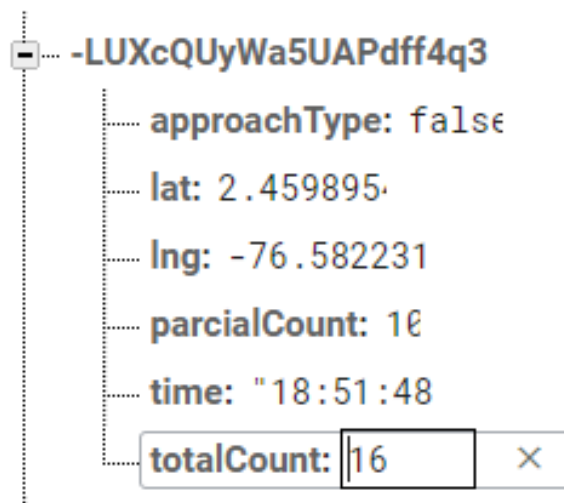


Ilustración 210 - Última posición de `listBoardingsInDay` en una iteración

Finalmente, es necesario organizar los dos vectores necesarios para la renderización de la gráfica por rango de días, la cual imprime total de pasajeros vs día en el que existe registro. Entonces, se inyectan como se muestra en la Ilustración 211.

```
<app-chart-range *ngIf="flagChart2 == true"
  [arrayDaysExist]="arrayDaysExist"
  [arrayTotalCounterDays]="arrayTotalCounterDays">
</app-chart-range>
```

Ilustración 211 - Inyectando vectores para gráfica por rango de días

El primer vector, `arrayDaysExist`, contiene aquellos días en los que hay un registro de actividad, el cual se obtiene de filtrar los resultados de la suscripción y comparar la fecha de búsqueda, entonces una vez se llena la lista de abordajes, se confirma la existencia de la fecha, por lo cual se agrega el día respectivo al vector. De forma similar, el vector `arrayTotalCounterDays`, contiene la cuenta total de aquellos días que tienen registro.

El componente de la gráfica de rango debe ser notificado de la búsqueda por rango de fecha, para que sean cargados los dos vectores y además sea renderizada la etiqueta `canvas` de la dependencia de gráficas. Dicho proceso se realiza nuevamente con la inclusión de un observable de tipo `BehaviorSubject`, el cual cambia de estado en el componente padre, entonces la suscripción del componente grafica de rango permite activar las funciones para el llenado de objetos y posterior dibujo de la gráfica, tal y como se observa en la Ilustración 212.

```
constructor(public graphsService: GraphsService) {
  this.graphsService.chartSubjetc2.subscribe((message: boolean) => {
    if (message === true) {
      setTimeout(() => {
        if (this.arrayDaysExist) {
          this.flag = false;
          this.barChartData = this.barChartDataFunc();
          this.barChartLabels = this.barChartLabelFunc();
          setTimeout(() => {
            this.flag = true;
            this.graphsService.chartSubjetc2.next(false);
          }, 500);
        }
      }, 500);
    } else {
      // this.flag = true;
    }
  });
}
```

Ilustración 212 - Renderización de grafica por rango de días

Anexo G: Preparación del entorno de desarrollo

Compilar SDK

Los requerimientos mínimos para que un ordenador pueda compilar el SDK de la librería de OpenCV se basan en:

- 2Gb de Memoria Ram
- 2Ghz velocidad de procesamiento
- Sistema operativo Windows de 32 o 64 bits.

El primer paso es realizar la descarga de las siguientes herramientas:

Tabla 24 - Requerimientos para compilar SDK OpenCV para Android

Nombre de la herramienta	Versión
Código fuente OpenCV	3.4.3
CMake	3.13.3
MinGW	5.3.0-3
Apache Ant	1.9.13
Java JDK	8u202
Android NDK	R16b
Python	2.7

De donde CMake es requerido para generar el proyecto, MinGW se encargará de construir un sistema de compilación basado en los proyectos GNU, compilando y enlazando código que se ejecutará en las plataformas con arquitectura Win32. Apache Ant, Java JDK y Android NDK son herramientas necesarias dentro de la generación interna del código. Finalmente, Python se encarga de interpretar varios de los scripts en la compilación.

El primer paso, luego de descargar e instalar las herramientas es abrir CMake e indicarle la carpeta en la que está alojado el código fuente de OpenCV y la carpeta que contendrá la salida, seguidamente se presiona el botón de configurar que aparece en la ventana, el cual abre la ventana de la Ilustración 213. Se debe seleccionar la opción MinGW Makefiles junto con *Specify toolchain file for cross-compiling*.

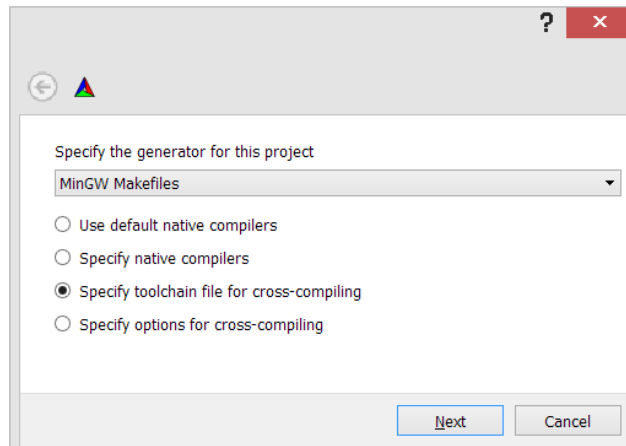


Ilustración 213 - Selección de MinGW Makefiles

La ventana que sigue, pide seleccionar el archivo *android.toolchain.cmake*, el cual se encuentra en la carpeta *../opencv-3.4.3/platform/android/*.

Si el proceso sale correcto, se debe observar una salida en ventana como en la Ilustración 214.

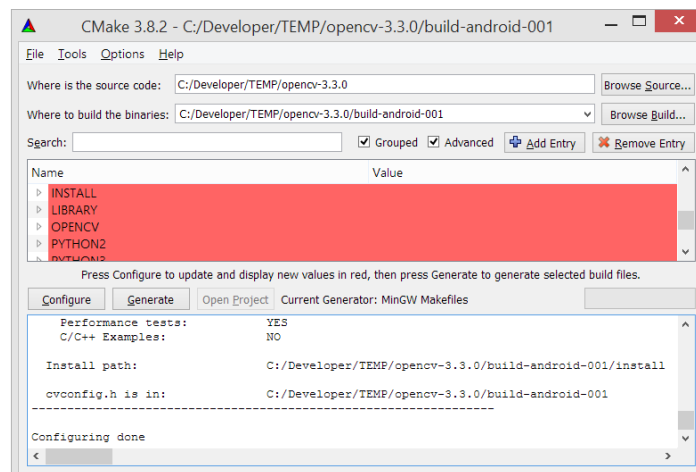


Ilustración 214 - Generando makefiles

Luego, ubicados en la carpeta donde fueron instalados los archivos, se debe ejecutar el comando *mingw-32-make*, el cual tiene una salida como la mostrada en la Ilustración 215.

```

Scanning dependencies of target libcpufeatures
[ 0%] Building C object 3rdparty/cpufeatures/CMakeFiles/libcpufeatures.dir/cpu-features.c.o
[ 0%] Linking C static library ..\lib\armeabi-v7a\libcpufeatures.a
[ 0%] Built target libcpufeatures
Scanning dependencies of target libtiff
[ 0%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_aux.c.o
[ 0%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_close.c.o
[ 0%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_codec.c.o
[ 0%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_color.c.o
[ 0%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_compress.c.o
[ 0%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_dir.c.o
[ 0%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_dirinfo.c.o
[ 0%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_dirread.c.o
[ 0%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_dirwrite.c.o
[ 0%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_dumpmode.c.o
[ 1%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_error.c.o
[ 1%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_extension.c.o
[ 1%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_fax3.c.o
[ 1%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_fax3sm.c.o
[ 1%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_flush.c.o
[ 1%] Building C object 3rdparty/libtiff/CMakeFiles/libtiff.dir/tif_getimage.c.o

```

Ilustración 215 - Comando mingw-32-make

Finalmente, se ejecuta el comando *mingw32-make install* el cual almacena los archivos resultantes del SDK en la dirección especificada el comenzar el proceso.

Agregar módulo OpenCV

Para poder usar las funcionalidades de la librería de OpenCV es necesario agregar el módulo a partir del SDK generado en el apartado anterior. Lo primero que se hace es crear un proyecto en blanco con un flujo normal, para esta fase del prototipo, no se agrega soporte Kotlin, ya que la librería usa funcionalidades Java, por ende, el cambio a soporte Kotlin se hace en el prototipo 3.

El primer paso es importar el módulo, para lo cual es necesario ir a *File -> New -> Import Module*. A continuación, se abre una ventana que pide indicar la dirección del SDK, tal y como se observa en la Ilustración 216.

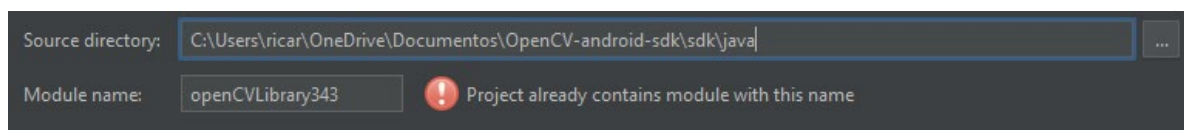


Ilustración 216 - Importando módulo OpenCV en Android

Luego, es necesario ir a la estructura de proyecto, seleccionar el módulo de la aplicación y en la sección dependencias, agregar OpenCV, Ilustración 217.

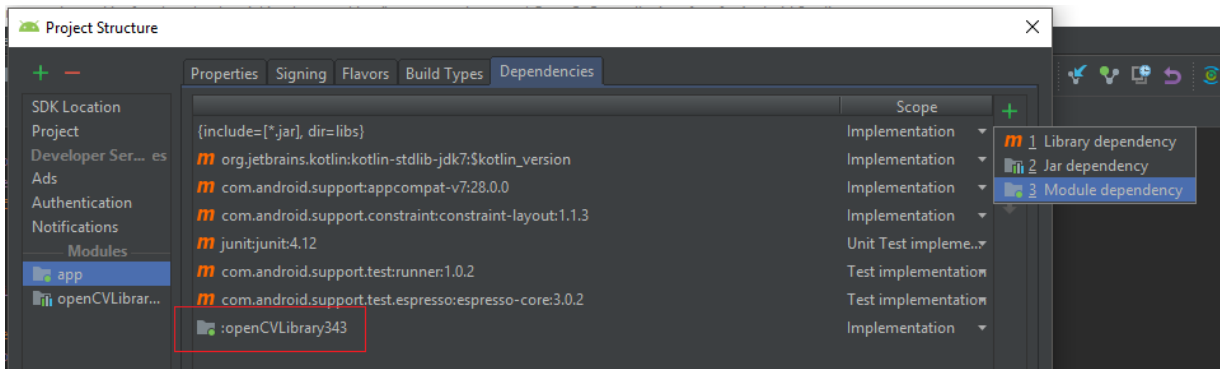


Ilustración 217 - Agregando dependencia de la librería de OpenCV

Una vez hecho lo anterior de forma satisfactoria, se observa que en la estructura Android aparece la carpeta correspondiente al módulo, Ilustración 218.

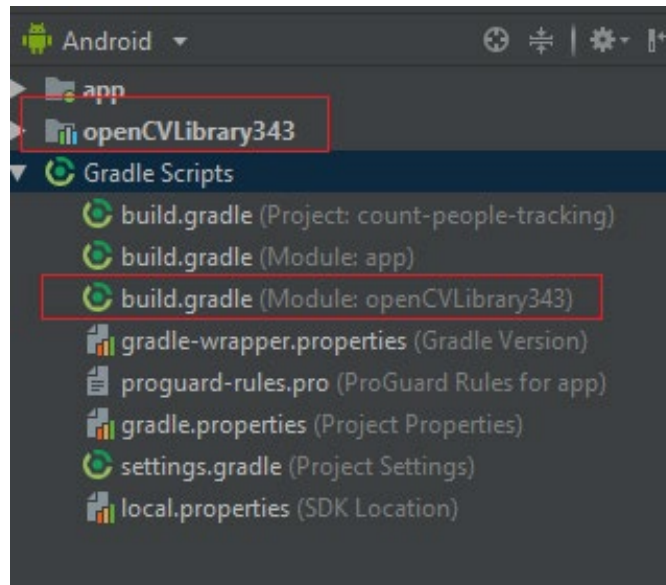


Ilustración 218 - Módulo OpenCV agregado

Instalar APK

Para que un dispositivo móvil pueda consumir los recursos del módulo OpenCV, es necesario instalar en él, una aplicación llamada *OpenCV Manager*, la cual viene en la carpeta instalada del SDK, Ilustración 219.

equipo > Documentos > OpenCV-android-sdk > apk

Nombre









-  OpenCV_3.4.3_Manager_3.43_arm64-v8a.apk
-  OpenCV_3.4.3_Manager_3.43_armeabi.apk
-  OpenCV_3.4.3_Manager_3.43_armeabi-v7a.apk
-  OpenCV_3.4.3_Manager_3.43_mips.apk
-  OpenCV_3.4.3_Manager_3.43_mips64.apk
-  OpenCV_3.4.3_Manager_3.43_x86.apk
-  OpenCV_3.4.3_Manager_3.43_x86_64.apk
-  readme

Ilustración 219 - Apk OpenCV Manager

Anexo H: Definición del concepto de observable

Un observable es un concepto propio de la programación reactiva, la cual está orientada al manejo de streams de datos asíncronos y la propagación del cambio.

Los streams son un flujo de datos, que han estado ligados a operaciones como lectura/escritura de ficheros o queries a base de datos. En la programación reactiva, sea cual sea el origen de la información, va a ser tratada como un Stream. Una de las máximas en programación reactiva es que todo es un Stream, desde vectores, rangos de números, promesas, etc.

El patrón observer juega un papel fundamental y explica a la perfección el concepto de reactivo. Dicho patrón define un productor de información, que va a ser un stream y que en programación reactiva está representado como un observable o consumidor, Ilustración 220.

```
//Observable
const myObservable$ = Rx.Observable.from([1,2,3]);

//Observer
const myObserver = {
  next: x => console.log(`Observer next value: ${x}`),
  error: err => console.error(`Observer error value: ${x}`),
  complete: () => console.log(`Observer complete notification`),
};

myObservable$.subscribe(myObserver);
```

Ilustración 220 - Observable o consumidor

El Observer es un objeto que dispone de 3 métodos para recibir información acerca del Observable. Cada uno de estos métodos cumple una función y a través de cada uno de ellos se reciben distintos tipos de notificaciones del Observable.

El Observer en sí mismo no devuelve ningún valor hasta que se active la comunicación entre ambas partes. Este mecanismo es la subscripción y nada se ejecutará hasta que se establezca una subscripción. El método subscribe activa el Observable y habilita al

Observer a recibir notificaciones del stream. No obstante, es posible establecer una subscripción pasando directamente funciones como argumentos, ya que internamente se asignará cada uno de estos callbacks a los respectivos métodos del Observer, siguiendo el orden en el que son pasados, siendo el primero de ellos next, el segundo error y el tercero complete, Ilustración 221.

```
const arrayStream$ = Rx.Observable.from([1,2,3]);  
  
arrayStream$  
  .subscribe(  
    next => console.log(next),  
    err => console.log(err),  
    () => console.log('completed!')  
  );
```

Ilustración 221 - Métodos de un observer

Este recurso se utilizó en el proyecto para acceder a todos los datos consultados a la base de datos, de modo que en los servicios se crean y en los componentes se ejecuta la función subscribe para acceder a ellos cuando existan y posteriormente poder usarlos.

Anexo I: Datos Recolectados Prototipo 1

A continuación se muestran las tablas que evidencian las pruebas realizadas por el prototipo 1, en donde se realizaron simulaciones de ascenso y descenso de pasajeros, incluyendo la ubicación de dicho evento (latitud y longitud).

Tabla 25 -Datos capturados Conductor 1- GXS-326 - Ruta 1

Fecha	Hora	Abordaje	C. Parcial	C. Total	Latitud	Longitud
2018-12-26	8:04:18	Ingreso	1	1	3,8945442	-76,3001605
2018-12-26	08:04:19	Ingreso	2	2	3,8945442	-76,3001605
2018-12-26	08:04:19	Ingreso	3	3	3,8945442	-76,3001605
2018-12-26	08:04:28	Ingreso	4	4	3,894575	-76,3000647
2018-12-26	08:05:07	Ingreso	5	5	3,8950342	-76,2989599
2018-12-26	08:05:09	Ingreso	6	6	3,8950518	-76,2989319
2018-12-26	08:05:30	Ingreso	7	7	3,8950832	-76,2987827
2018-12-26	08:05:31	Ingreso	8	8	3,8950832	-76,2987827
2018-12-26	08:05:49	Salida	7	8	3,8954196	-76,2975534
2018-12-26	08:05:56	Ingreso	8	9	3,8954196	-76,2975534
2018-12-26	08:06:00	Ingreso	9	10	3,8954286	-76,2975269
2018-12-26	08:06:15	Ingreso	10	11	3,8955877	-76,2972317
2018-12-26	08:06:16	Ingreso	11	12	3,8955877	-76,2972317
2018-12-26	08:06:17	Ingreso	12	13	3,8955877	-76,2972317
2018-12-26	08:06:48	Ingreso	13	14	3,8973355	-76,2966639
2018-12-26	08:07:21	Salida	12	14	3,8984307	-76,2962129
2018-12-26	08:07:22	Salida	11	14	3,8984307	-76,2962129
2018-12-26	08:07:24	Salida	10	14	3,8987957	-76,2961631
2018-12-26	09:07:37	Ingreso	11	15	3,8994582	-76,2958854
2018-12-26	09:07:44	Ingreso	12	16	3,899832	-76,2957388
2018-12-26	09:08:01	Salida	11	16	3,9006186	-76,2953657
2018-12-26	09:08:14	Salida	10	16	3,9013386	-76,2950959
2018-12-26	09:08:34	Ingreso	11	17	3,9019918	-76,2948434
2018-12-26	09:08:35	Ingreso	12	18	3,9019918	-76,2948434
2018-12-26	09:08:35	Ingreso	13	19	3,9019918	-76,2948434

2018-12-26	09:08:36	Ingreso	14	20	3,9019918	-76,2948434
2018-12-26	09:09:07	Salida	13	20	3,9027386	-76,2945038
2018-12-26	09:09:15	Ingreso	14	21	3,9030527	-76,2943491
2018-12-26	09:09:29	Ingreso	15	22	3,904202	-76,2939495
2018-12-26	09:09:39	Ingreso	16	23	3,9046023	-76,2937376
2018-12-26	09:09:49	Ingreso	17	24	3,9052174	-76,2936193
2018-12-26	09:10:14	Ingreso	18	25	3,906455	-76,2930375
2018-12-26	09:10:20	Salida	17	25	3,906734	-76,2929785
2018-12-26	09:10:20	Salida	16	25	3,906734	-76,2929785
2018-12-26	09:10:21	Salida	15	25	3,906734	-76,2929785
2018-12-26	09:10:21	Salida	14	25	3,906734	-76,2929785
2018-12-26	09:10:21	Salida	13	25	3,906734	-76,2929785
2018-12-26	09:10:22	Salida	12	25	3,9067762	-76,2929568
2018-12-26	10:10:22	Salida	11	25	3,9067762	-76,2929568
2018-12-26	10:11:12	Ingreso	12	26	3,9081073	-76,292426
2018-12-26	10:11:19	Ingreso	13	27	3,9093666	-76,2917311
2018-12-26	10:12:26	Ingreso	14	28	3,9112884	-76,2947112
2018-12-26	10:12:52	Salida	13	28	3,9117502	-76,2958016
2018-12-26	10:12:56	Salida	12	28	3,9117977	-76,2960553
2018-12-26	10:12:59	Salida	11	28	3,9117977	-76,2960553
2018-12-26	10:13:01	Salida	10	28	3,9118176	-76,2960668
2018-12-26	10:13:04	Ingreso	11	29	3,9118176	-76,2960668
2018-12-26	10:13:59	Ingreso	12	30	3,912286	-76,2972851
2018-12-26	10:14:54	Ingreso	13	31	3,9117765	-76,2979031
2018-12-26	10:15:03	Salida	12	31	3,9111209	-76,2981025
2018-12-26	10:15:05	Salida	11	31	3,9111209	-76,2981025
2018-12-26	10:15:23	Ingreso	12	32	3,9100482	-76,2985744
2018-12-26	10:15:25	Ingreso	13	33	3,9100482	-76,2985744
2018-12-26	10:15:30	Ingreso	14	34	3,9093637	-76,2988525
2018-12-26	10:15:44	Salida	13	34	3,9087015	-76,2991662
2018-12-26	10:15:46	Salida	12	34	3,9087015	-76,2991662
2018-12-26	10:15:47	Salida	11	34	3,9087015	-76,2991662
2018-12-26	10:16:08	Ingreso	12	35	3,9076448	-76,2995963
2018-12-26	10:16:09	Salida	11	35	3,9076448	-76,2995963
2018-12-26	10:16:12	Salida	10	35	3,9076448	-76,2995963
2018-12-26	10:16:14	Salida	9	35	3,9076352	-76,2996013
2018-12-26	11:16:16	Salida	8	35	3,9076352	-76,2996013
2018-12-26	11:16:32	Ingreso	9	36	3,9076029	-76,2996048
2018-12-26	11:16:38	Ingreso	10	37	3,9070719	-76,2998343
2018-12-26	11:16:38	Ingreso	11	38	3,9066834	-76,2999903
2018-12-26	11:16:51	Ingreso	12	39	3,9064177	-76,3001448
2018-12-26	11:16:58	Ingreso	13	40	3,9054519	-76,3005613
2018-12-26	11:17:05	Salida	12	40	3,9054519	-76,3005613

2018-12-26	11:17:12	Ingreso	13	41	3,9047304	-76,3008089
2018-12-26	11:17:18	Ingreso	14	42	3,9040351	-76,3011024
2018-12-26	11:17:27	Ingreso	15	43	3,9036119	-76,3011916
2018-12-26	11:17:32	Salida	14	43	3,9029458	-76,3013697
2018-12-26	11:17:39	Ingreso	15	44	3,9028233	-76,3014479
2018-12-26	11:17:52	Salida	14	44	3,9022616	-76,3016697
2018-12-26	11:19:12	Salida	13	44	3,9004214	-76,3023256
2018-12-26	11:19:19	Salida	12	44	3,9004214	-76,3023256
2018-12-26	11:19:26	Salida	11	44	3,900332	-76,3023581
2018-12-26	11:19:30	Salida	10	44	3,9003012	-76,3023596
2018-12-26	11:19:34	Salida	9	44	3,9003012	-76,3023596
2018-12-26	11:19:38	Salida	8	44	3,9002867	-76,3023596
2018-12-26	11:20:26	Ingreso	9	45	3,8999187	-76,3024162
2018-12-26	11:21:53	Salida	8	45	3,8977851	-76,3032189
2018-12-26	11:21:58	Salida	7	45	3,8977212	-76,3032472
2018-12-26	11:22:00	Salida	6	45	3,8977212	-76,3032472
2018-12-26	12:22:03	Salida	5	45	3,8977212	-76,3032472
2018-12-26	12:22:06	Salida	4	45	3,8976716	-76,3032855
2018-12-26	12:22:10	Ingreso	5	46	3,8976716	-76,3032855
2018-12-26	12:22:12	Ingreso	6	47	3,8976716	-76,3032855
2018-12-26	12:22:14	Ingreso	7	48	3,8976716	-76,3032855
2018-12-26	12:22:42	Salida	6	48	3,896687	-76,3035504
2018-12-26	12:22:56	Ingreso	7	49	3,8961264	-76,3038033
2018-12-26	12:23:46	Salida	6	49	3,8951842	-76,3041298
2018-12-26	12:24:07	Salida	5	49	3,8950277	-76,3034972
2018-12-26	12:24:24	Salida	4	49	3,8948625	-76,3026613
2018-12-26	12:24:32	Salida	3	49	3,8947576	-76,3025034
2018-12-26	12:24:37	Salida	2	49	3,8947576	-76,3025034
2018-12-26	12:24:53	Ingreso	3	50	3,8947137	-76,3022223
2018-12-26	12:26:35	Salida	2	50	3,8944709	-76,3001883
2018-12-26	12:26:38	Salida	1	50	3,8943718	-76,3002397
2018-12-26	12:26:40	Salida	0	50	3,8943718	-76,3002397

Tabla 26 -Datos capturados Conductor 2- bus DTY-346 - Ruta 2

Fecha	Hora	Abordaje	C. Parcial	C. Total	Latitud	Longitud
2018-12-26	13:08:55	Ingreso	1	1	3,894702	-76,2995444
2018-12-26	13:08:55	Ingreso	2	2	3,894702	-76,2995444
2018-12-26	13:08:57	Ingreso	3	3	3,8948814	-76,2992322
2018-12-26	13:09:35	Ingreso	4	4	3,8954964	-76,2973338
2018-12-26	13:09:43	Ingreso	5	5	3,8954741	-76,2969589
2018-12-26	13:09:56	Ingreso	6	6	3,8951961	-76,2961359
2018-12-26	13:10:08	Salida	5	6	3,8950771	-76,2955427

2018-12-26	13:10:15	Salida	4	6	3,8949591	-76,295296
2018-12-26	13:10:23	Ingreso	5	7	3,8948811	-76,2949273
2018-12-26	13:10:44	Ingreso	6	8	3,8943496	-76,2939349
2018-12-26	13:10:57	Salida	5	8	3,8936508	-76,293257
2018-12-26	13:11:05	Ingreso	6	9	3,8936508	-76,293257
2018-12-26	13:11:07	Ingreso	7	10	3,8933551	-76,2927854
2018-12-26	13:11:10	Ingreso	8	11	3,8933551	-76,2927854
2018-12-26	13:11:13	Ingreso	9	12	3,8934136	-76,2929008
2018-12-26	13:11:23	Salida	8	12	3,8934577	-76,2927142
2018-12-26	13:11:28	Salida	7	12	3,8934577	-76,2927142
2018-12-26	13:11:38	Ingreso	8	13	3,8930711	-76,2919756
2018-12-26	13:11:46	Salida	7	13	3,8929741	-76,2918142
2018-12-26	13:11:54	Ingreso	8	14	3,8929389	-76,2917478
2018-12-26	13:11:56	Ingreso	9	15	3,8929389	-76,2917478
2018-12-26	13:11:56	Ingreso	10	16	3,8929389	-76,2917478
2018-12-26	13:11:59	Ingreso	11	17	3,8929835	-76,2918373
2018-12-26	13:12:28	Salida	10	17	3,8931643	-76,2917619
2018-12-26	13:12:38	Salida	9	17	3,8937414	-76,2914872
2018-12-26	13:12:49	Ingreso	10	18	3,894745	-76,2912165
2018-12-26	13:13:09	Ingreso	11	19	3,8958659	-76,2907979
2018-12-26	13:13:13	Ingreso	12	20	3,8958659	-76,2907979
2018-12-26	13:13:15	Ingreso	13	21	3,8958659	-76,2907979
2018-12-26	13:13:21	Salida	12	21	3,8959843	-76,2907926
2018-12-26	13:13:54	Ingreso	13	22	3,8964611	-76,2914926
2018-12-26	13:14:04	Salida	12	22	3,8967884	-76,2923509
2018-12-26	13:14:29	Ingreso	13	23	3,8974857	-76,2939978
2018-12-26	13:14:37	Ingreso	14	24	3,897738	-76,2946344
2018-12-26	13:14:51	Ingreso	15	25	3,8981095	-76,2955402
2018-12-26	13:15:24	Ingreso	16	26	3,8984062	-76,2963444
2018-12-26	13:15:45	Salida	15	26	3,8987177	-76,2971074
2018-12-26	13:15:47	Ingreso	16	27	3,8987177	-76,2971074
2018-12-26	13:15:54	Salida	15	27	3,8987678	-76,2975222
2018-12-26	13:15:56	Salida	14	27	3,8987678	-76,2975222
2018-12-26	13:16:02	Ingreso	15	28	3,8988805	-76,2979031
2018-12-26	13:16:04	Salida	14	28	3,8988805	-76,2979031
2018-12-26	13:16:05	Ingreso	15	29	3,8990265	-76,2982875
2018-12-26	13:16:06	Ingreso	16	30	3,8990265	-76,2982875
2018-12-26	13:16:07	Salida	15	30	3,8990265	-76,2982875
2018-12-26	13:16:09	Salida	14	30	3,8990265	-76,2982875
2018-12-26	13:16:10	Salida	13	30	3,8991045	-76,2986216
2018-12-26	13:16:21	Ingreso	14	31	3,8992506	-76,298992
2018-12-26	13:16:25	Ingreso	15	32	3,8994062	-76,2992173
2018-12-26	13:16:33	Salida	14	32	3,8994772	-76,2994142

2018-12-26	13:16:35	Salida	13	32	3,8994772	-76,2994142
2018-12-26	13:16:39	Ingreso	14	33	3,8995124	-76,2994488
2018-12-26	13:16:42	Salida	13	33	3,8995124	-76,2994488
2018-12-26	13:16:43	Ingreso	14	34	3,8995124	-76,2994488
2018-12-26	13:16:44	Ingreso	15	35	3,8995212	-76,2994593
2018-12-26	13:17:07	Salida	14	35	3,8994072	-76,2994905
2018-12-26	13:17:10	Salida	13	35	3,8994072	-76,2994905
2018-12-26	13:17:16	Salida	12	35	3,8996691	-76,2999523
2018-12-26	13:17:24	Salida	11	35	3,8997946	-76,3003836
2018-12-26	13:17:29	Salida	10	35	3,8998096	-76,3005404
2018-12-26	13:17:36	Salida	9	35	3,8998969	-76,3007503
2018-12-26	13:17:40	Salida	8	35	3,8998969	-76,3007503
2018-12-26	13:18:25	Ingreso	9	36	3,900345	-76,3021463
2018-12-26	13:18:26	Ingreso	10	37	3,900345	-76,3021463
2018-12-26	13:18:27	Ingreso	11	38	3,900345	-76,3021463
2018-12-26	13:18:40	Salida	10	38	3,9008224	-76,3030154
2018-12-26	13:18:51	Ingreso	11	39	3,9010155	-76,3037235
2018-12-26	13:18:53	Ingreso	12	40	3,9010155	-76,3037235
2018-12-26	13:19:05	Ingreso	13	41	3,9008178	-76,3041614
2018-12-26	13:19:24	Ingreso	14	42	3,8998929	-76,3044348
2018-12-26	13:19:29	Ingreso	15	43	3,8998929	-76,3044348
2018-12-26	13:19:31	Ingreso	16	44	3,8998366	-76,3044517
2018-12-26	13:19:33	Ingreso	17	45	3,8998366	-76,3044517
2018-12-26	13:19:35	Salida	16	45	3,8997806	-76,3044903
2018-12-26	13:19:37	Salida	15	45	3,8997806	-76,3044903
2018-12-26	13:19:39	Salida	14	45	3,8997806	-76,3044903
2018-12-26	13:19:40	Salida	13	45	3,8997603	-76,3045102
2018-12-26	13:19:43	Salida	12	45	3,8997603	-76,3045102
2018-12-26	13:20:19	Ingreso	13	46	3,8987625	-76,3048607
2018-12-26	13:20:24	Ingreso	14	47	3,8983279	-76,3051665
2018-12-26	13:20:35	Salida	13	47	3,898202	-76,3051473
2018-12-26	13:20:55	Ingreso	14	48	3,8969814	-76,3055931
2018-12-26	13:21:09	Ingreso	15	49	3,8961715	-76,305837
2018-12-26	13:21:24	Ingreso	16	50	3,8956347	-76,3053129
2018-12-26	13:21:31	Ingreso	17	51	3,8954057	-76,3052007
2018-12-26	13:21:38	Salida	16	51	3,8951492	-76,3052869
2018-12-26	13:21:43	Salida	15	51	3,8948357	-76,3056493
2018-12-26	13:21:51	Ingreso	16	52	3,8945782	-76,3060731
2018-12-26	13:22:06	Salida	15	52	3,8940062	-76,3059391
2018-12-26	13:22:09	Salida	14	52	3,8934468	-76,3060961
2018-12-26	13:22:11	Salida	13	52	3,8934468	-76,3060961
2018-12-26	13:22:13	Salida	12	52	3,8934468	-76,3060961
2018-12-26	13:22:15	Salida	11	52	3,8934468	-76,3060961

2018-12-26	13:22:17	Salida	10	52	3,8934468	-76,3060961
2018-12-26	13:22:18	Ingreso	11	53	3,8934206	-76,3060996
2018-12-26	13:22:43	Salida	10	53	3,8926528	-76,3062316
2018-12-26	13:22:54	Ingreso	11	54	3,8919674	-76,3064827
2018-12-26	13:23:04	Ingreso	12	55	3,8915093	-76,306172
2018-12-26	13:23:22	Salida	11	55	3,8914937	-76,3049787
2018-12-26	13:23:38	Salida	10	55	3,8912771	-76,3036483
2018-12-26	13:23:48	Ingreso	11	56	3,8912274	-76,303232
2018-12-26	13:24:03	Ingreso	12	57	3,8913354	-76,3015577
2018-12-26	13:24:15	Salida	11	57	3,8911861	-76,3012965
2018-12-26	13:24:23	Salida	10	57	3,8907588	-76,3011646
2018-12-26	13:24:25	Salida	9	57	3,8907588	-76,3011646
2018-12-26	13:24:27	Salida	8	57	3,8907693	-76,3011709
2018-12-26	13:24:28	Ingreso	9	58	3,8907693	-76,3011709
2018-12-26	13:24:29	Ingreso	10	59	3,8907693	-76,3011709
2018-12-26	13:24:31	Ingreso	11	60	3,8907693	-76,3011709
2018-12-26	13:24:37	Ingreso	12	61	3,890782	-76,3011121
2018-12-26	13:24:37	Salida	11	61	3,890782	-76,3011121
2018-12-26	13:24:38	Ingreso	12	62	3,890782	-76,3011121
2018-12-26	13:24:38	Salida	11	62	3,890782	-76,3011121
2018-12-26	13:24:43	Ingreso	12	63	3,8907445	-76,3010904
2018-12-26	13:24:43	Salida	10	62	3,8907445	-76,3010904
2018-12-26	13:25:08	Ingreso	11	64	3,8894284	-76,3003868
2018-12-26	13:25:09	Ingreso	12	65	3,8889992	-76,3002205
2018-12-26	13:25:23	Salida	11	65	3,8886407	-76,2997335
2018-12-26	13:25:32	Salida	10	65	3,8887471	-76,29933
2018-12-26	13:25:47	Salida	9	65	3,8892998	-76,2982453
2018-12-26	13:25:50	Salida	8	65	3,8892998	-76,2982453
2018-12-26	13:25:57	Salida	7	65	3,8895777	-76,2977523
2018-12-26	13:26:01	Ingreso	8	66	3,8895777	-76,2977523
2018-12-26	13:26:15	Ingreso	9	67	3,8896405	-76,2976638
2018-12-26	13:26:34	Salida	8	67	3,8906192	-76,2979166
2018-12-26	13:26:46	Ingreso	9	68	3,8917619	-76,2984449
2018-12-26	13:26:48	Ingreso	10	69	3,8919657	-76,2985479
2018-12-26	13:26:49	Ingreso	11	70	3,8919657	-76,2985479
2018-12-26	13:26:51	Ingreso	12	71	3,8919657	-76,2985479
2018-12-26	13:26:52	Salida	11	71	3,8919657	-76,2985479
2018-12-26	13:26:53	Salida	10	71	3,8919803	-76,2985519
2018-12-26	13:26:54	Salida	9	71	3,8919803	-76,2985519
2018-12-26	13:26:55	Salida	8	71	3,8919803	-76,2985519
2018-12-26	13:27:03	Salida	7	71	3,8920038	-76,2985931
2018-12-26	13:27:03	Salida	6	71	3,8920038	-76,2985931
2018-12-26	13:27:16	Salida	5	71	3,89254	-76,2986385

2018-12-26	13:27:17	Salida	4	71	3,8930155	-76,298642
2018-12-26	13:27:19	Ingreso	5	72	3,8930155	-76,298642
2018-12-26	13:27:20	Ingreso	6	73	3,8930155	-76,298642
2018-12-26	13:27:25	Salida	5	73	3,8933651	-76,2986113
2018-12-26	13:27:37	Salida	4	73	3,8943047	-76,2983718
2018-12-26	13:27:39	Salida	3	73	3,894626	-76,2983504
2018-12-26	13:27:40	Salida	2	73	3,894626	-76,2983504
2018-12-26	13:28:00	Salida	1	73	3,895372	-76,2978986
2018-12-26	13:28:03	Salida	0	73	3,8954878	-76,297667

Tabla 27 - Datos rango de días 24 al 26 de diciembre 2018 - Conductor 1 - bus GXS-326 -Ruta 1

Fecha	Hora	Abordaje	C. Parcial	C. Total	Latitud	Longitud
2018-12-24	18:46:58	Ingreso	1	1	2,4598954	-76,5822318
2018-12-24	18:49:23	Ingreso	2	2	2,4598954	-76,5822318
2018-12-24	18:49:30	Ingreso	3	3	2,4598954	-76,5822318
2018-12-24	18:49:33	Salida	2	3	2,4598954	-76,5822318
2018-12-24	18:49:40	Ingreso	3	4	2,4598954	-76,5822318
2018-12-24	18:49:53	Ingreso	4	5	2,4598954	-76,5822318
2018-12-24	18:50:00	Ingreso	5	6	2,4598954	-76,5822318
2018-12-24	18:50:07	Ingreso	6	7	2,4598954	-76,5822318
2018-12-24	18:50:11	Ingreso	7	8	2,4598954	-76,5822318
2018-12-24	18:50:14	Salida	6	8	2,4598954	-76,5822318
2018-12-24	18:50:16	Salida	5	8	2,4598954	-76,5822318
2018-12-24	18:50:18	Salida	4	8	2,4598954	-76,5822318
2018-12-24	18:50:21	Ingreso	5	9	2,4598954	-76,5822318
2018-12-24	18:50:23	Ingreso	6	10	2,4598954	-76,5822318
2018-12-24	18:50:30	Ingreso	7	11	2,4598954	-76,5822318
2018-12-24	18:50:35	Ingreso	8	12	2,4598954	-76,5822318
2018-12-24	18:50:53	Ingreso	9	13	2,4598954	-76,5822318
2018-12-24	18:50:59	Salida	8	13	2,4598954	-76,5822318
2018-12-24	18:51:17	Ingreso	9	14	2,4598954	-76,5822318
2018-12-24	18:51:30	Ingreso	10	15	2,4598954	-76,5822318
2018-12-24	18:51:41	Ingreso	11	16	2,4598954	-76,5822318
2018-12-24	18:51:48	Salida	10	16	2,4598954	-76,5822318
2018-12-24	18:51:49	Ingreso	11	17	2,4598954	-76,5822318
2018-12-24	18:52:00	Ingreso	12	18	2,4598954	-76,5822318
2018-12-25	16:38:09	Ingreso	1	1	2,4598966	-76,5822317
2018-12-25	16:38:31	Ingreso	2	2	2,4598692	-76,5822461
2018-12-25	16:38:36	Ingreso	3	3	2,4598754	-76,5822403
2018-12-25	16:38:40	Salida	2	3	2,459879	-76,582246
2018-12-25	16:38:52	Ingreso	3	4	2,4598719	-76,5822536
2018-12-25	16:38:58	Ingreso	4	5	2,4598738	-76,5822451

2018-12-25	16:39:04	Salida	3	5	2,4598557	-76,5821978
2018-12-25	16:39:08	Ingreso	4	6	2,4598379	-76,5821071
2018-12-25	16:39:14	Ingreso	5	7	2,4598241	-76,5820447
2018-12-25	16:39:21	Ingreso	6	8	2,4598178	-76,5820198
2018-12-25	16:39:22	Ingreso	7	9	2,4598178	-76,5820198
2018-12-25	16:39:22	Ingreso	8	10	2,4598178	-76,5820198
2018-12-25	16:39:23	Salida	7	10	2,4598178	-76,5820198
2018-12-25	16:39:26	Ingreso	8	11	2,4598103	-76,5819397
2018-12-25	16:39:31	Ingreso	9	12	2,4598171	-76,5818798
2018-12-25	16:39:32	Ingreso	10	13	2,4598171	-76,5818798
2018-12-25	16:39:36	Ingreso	11	14	2,4598198	-76,5818007
2018-12-25	16:39:41	Ingreso	12	15	2,4598132	-76,5817185
2018-12-25	16:39:42	Ingreso	13	16	2,4598132	-76,5817185
2018-12-25	16:39:45	Ingreso	14	17	2,4598132	-76,5817185
2018-12-25	16:39:49	Ingreso	15	18	2,4598053	-76,5816449
2018-12-25	16:39:50	Ingreso	16	19	2,4598053	-76,5816449
2018-12-25	16:39:51	Salida	15	19	2,4598053	-76,5816449
2018-12-25	17:04:10	Ingreso	16	20	2,4598914	-76,5822335
2018-12-26	8:04:18	Ingreso	1	1	3,8945442	-76,3001605
2018-12-26	08:04:19	Ingreso	2	2	3,8945442	-76,3001605
2018-12-26	08:04:19	Ingreso	3	3	3,8945442	-76,3001605
2018-12-26	08:04:28	Ingreso	4	4	3,894575	-76,3000647
2018-12-26	08:05:07	Ingreso	5	5	3,8950342	-76,2989599
2018-12-26	08:05:09	Ingreso	6	6	3,8950518	-76,2989319
2018-12-26	08:05:30	Ingreso	7	7	3,8950832	-76,2987827
2018-12-26	08:05:31	Ingreso	8	8	3,8950832	-76,2987827
2018-12-26	08:05:49	Salida	7	8	3,8954196	-76,2975534
2018-12-26	08:05:56	Ingreso	8	9	3,8954196	-76,2975534
2018-12-26	08:06:00	Ingreso	9	10	3,8954286	-76,2975269
2018-12-26	08:06:15	Ingreso	10	11	3,8955877	-76,2972317
2018-12-26	08:06:16	Ingreso	11	12	3,8955877	-76,2972317
2018-12-26	08:06:17	Ingreso	12	13	3,8955877	-76,2972317
2018-12-26	08:06:48	Ingreso	13	14	3,8973355	-76,2966639
2018-12-26	08:07:21	Salida	12	14	3,8984307	-76,2962129
2018-12-26	08:07:22	Salida	11	14	3,8984307	-76,2962129
2018-12-26	08:07:24	Salida	10	14	3,8987957	-76,2961631
2018-12-26	09:07:37	Ingreso	11	15	3,8994582	-76,2958854
2018-12-26	09:07:44	Ingreso	12	16	3,899832	-76,2957388
2018-12-26	09:08:01	Salida	11	16	3,9006186	-76,2953657
2018-12-26	09:08:14	Salida	10	16	3,9013386	-76,2950959
2018-12-26	09:08:34	Ingreso	11	17	3,9019918	-76,2948434
2018-12-26	09:08:35	Ingreso	12	18	3,9019918	-76,2948434
2018-12-26	09:08:35	Ingreso	13	19	3,9019918	-76,2948434

2018-12-26	09:08:36	Ingreso	14	20	3,9019918	-76,2948434
2018-12-26	09:09:07	Salida	13	20	3,9027386	-76,2945038
2018-12-26	09:09:15	Ingreso	14	21	3,9030527	-76,2943491
2018-12-26	09:09:29	Ingreso	15	22	3,904202	-76,2939495
2018-12-26	09:09:39	Ingreso	16	23	3,9046023	-76,2937376
2018-12-26	09:09:49	Ingreso	17	24	3,9052174	-76,2936193
2018-12-26	09:10:14	Ingreso	18	25	3,906455	-76,2930375
2018-12-26	09:10:20	Salida	17	25	3,906734	-76,2929785
2018-12-26	09:10:20	Salida	16	25	3,906734	-76,2929785
2018-12-26	09:10:21	Salida	15	25	3,906734	-76,2929785
2018-12-26	09:10:21	Salida	14	25	3,906734	-76,2929785
2018-12-26	09:10:21	Salida	13	25	3,906734	-76,2929785
2018-12-26	09:10:22	Salida	12	25	3,9067762	-76,2929568
2018-12-26	10:10:22	Salida	11	25	3,9067762	-76,2929568
2018-12-26	10:11:12	Ingreso	12	26	3,9081073	-76,292426
2018-12-26	10:11:19	Ingreso	13	27	3,9093666	-76,2917311
2018-12-26	10:12:26	Ingreso	14	28	3,9112884	-76,2947112
2018-12-26	10:12:52	Salida	13	28	3,9117502	-76,2958016
2018-12-26	10:12:56	Salida	12	28	3,9117977	-76,2960553
2018-12-26	10:12:59	Salida	11	28	3,9117977	-76,2960553
2018-12-26	10:13:01	Salida	10	28	3,9118176	-76,2960668
2018-12-26	10:13:04	Ingreso	11	29	3,9118176	-76,2960668
2018-12-26	10:13:59	Ingreso	12	30	3,912286	-76,2972851
2018-12-26	10:14:54	Ingreso	13	31	3,9117765	-76,2979031
2018-12-26	10:15:03	Salida	12	31	3,9111209	-76,2981025
2018-12-26	10:15:05	Salida	11	31	3,9111209	-76,2981025
2018-12-26	10:15:23	Ingreso	12	32	3,9100482	-76,2985744
2018-12-26	10:15:25	Ingreso	13	33	3,9100482	-76,2985744
2018-12-26	10:15:30	Ingreso	14	34	3,9093637	-76,2988525
2018-12-26	10:15:44	Salida	13	34	3,9087015	-76,2991662
2018-12-26	10:15:46	Salida	12	34	3,9087015	-76,2991662
2018-12-26	10:15:47	Salida	11	34	3,9087015	-76,2991662
2018-12-26	10:16:08	Ingreso	12	35	3,9076448	-76,2995963
2018-12-26	10:16:09	Salida	11	35	3,9076448	-76,2995963
2018-12-26	10:16:12	Salida	10	35	3,9076448	-76,2995963
2018-12-26	10:16:14	Salida	9	35	3,9076352	-76,2996013
2018-12-26	11:16:16	Salida	8	35	3,9076352	-76,2996013
2018-12-26	11:16:32	Ingreso	9	36	3,9076029	-76,2996048
2018-12-26	11:16:38	Ingreso	10	37	3,9070719	-76,2998343
2018-12-26	11:16:38	Ingreso	11	38	3,9066834	-76,2999903
2018-12-26	11:16:51	Ingreso	12	39	3,9064177	-76,3001448
2018-12-26	11:16:58	Ingreso	13	40	3,9054519	-76,3005613
2018-12-26	11:17:05	Salida	12	40	3,9054519	-76,3005613

2018-12-26	11:17:12	Ingreso	13	41	3,9047304	-76,3008089
2018-12-26	11:17:18	Ingreso	14	42	3,9040351	-76,3011024
2018-12-26	11:17:27	Ingreso	15	43	3,9036119	-76,3011916
2018-12-26	11:17:32	Salida	14	43	3,9029458	-76,3013697
2018-12-26	11:17:39	Ingreso	15	44	3,9028233	-76,3014479
2018-12-26	11:17:52	Salida	14	44	3,9022616	-76,3016697
2018-12-26	11:19:12	Salida	13	44	3,9004214	-76,3023256
2018-12-26	11:19:19	Salida	12	44	3,9004214	-76,3023256
2018-12-26	11:19:26	Salida	11	44	3,900332	-76,3023581
2018-12-26	11:19:30	Salida	10	44	3,9003012	-76,3023596
2018-12-26	11:19:34	Salida	9	44	3,9003012	-76,3023596
2018-12-26	11:19:38	Salida	8	44	3,9002867	-76,3023596
2018-12-26	11:20:26	Ingreso	9	45	3,8999187	-76,3024162
2018-12-26	11:21:53	Salida	8	45	3,8977851	-76,3032189
2018-12-26	11:21:58	Salida	7	45	3,8977212	-76,3032472
2018-12-26	11:22:00	Salida	6	45	3,8977212	-76,3032472
2018-12-26	12:22:03	Salida	5	45	3,8977212	-76,3032472
2018-12-26	12:22:06	Salida	4	45	3,8976716	-76,3032855
2018-12-26	12:22:10	Ingreso	5	46	3,8976716	-76,3032855
2018-12-26	12:22:12	Ingreso	6	47	3,8976716	-76,3032855
2018-12-26	12:22:14	Ingreso	7	48	3,8976716	-76,3032855
2018-12-26	12:22:42	Salida	6	48	3,896687	-76,3035504
2018-12-26	12:22:56	Ingreso	7	49	3,8961264	-76,3038033
2018-12-26	12:23:46	Salida	6	49	3,8951842	-76,3041298
2018-12-26	12:24:07	Salida	5	49	3,8950277	-76,3034972
2018-12-26	12:24:24	Salida	4	49	3,8948625	-76,3026613
2018-12-26	12:24:32	Salida	3	49	3,8947576	-76,3025034
2018-12-26	12:24:37	Salida	2	49	3,8947576	-76,3025034
2018-12-26	12:24:53	Ingreso	3	50	3,8947137	-76,3022223
2018-12-26	12:26:35	Salida	2	50	3,8944709	-76,3001883
2018-12-26	12:26:38	Salida	1	50	3,8943718	-76,3002397
2018-12-26	12:26:40	Salida	0	50	3,8943718	-76,3002397

Anexo J: Pruebas ascenso/descenso prototipo 2

En las siguientes tablas, se encuentran las pruebas del prototipo 2, en donde bajo un ambiente simulado se emularon eventos de ascenso y descenso de pasajeros, dichos eventos fueron captados en base a los algoritmos implementados de la librería de análisis de imágenes.

Número de muestra	Tipo de Flujo	Correcto
1	Ascenso	SI
2	Descenso	SI
3	Ascenso	SI
4	Descenso	SI
5	Ascenso	SI
6	Descenso	SI
7	Ascenso	SI
8	Descenso	NO
9	Ascenso	SI
10	Descenso	SI
11	Ascenso	SI
12	Descenso	SI
13	Ascenso	SI
14	Descenso	SI
15	Ascenso	SI
16	Descenso	SI
17	Ascenso	SI
18	Descenso	SI
19	Ascenso	SI
20	Descenso	SI
21	Ascenso	SI
22	Descenso	SI
23	Ascenso	SI
24	Descenso	SI
25	Ascenso	SI

26	Descenso	SI
27	Ascenso	SI
28	Descenso	SI
29	Ascenso	SI
30	Descenso	NO
31	Ascenso	SI
32	Descenso	SI
33	Ascenso	SI
34	Descenso	SI
35	Ascenso	SI
36	Descenso	SI
37	Ascenso	SI
38	Descenso	SI
39	Ascenso	SI
40	Descenso	SI
41	Ascenso	SI
42	Descenso	SI
43	Ascenso	SI
44	Descenso	SI
45	Ascenso	SI
46	Descenso	SI
47	Ascenso	SI
48	Descenso	SI
49	Ascenso	SI
50	Descenso	SI
51	Ascenso	NO
52	Descenso	SI

53	Ascenso	SI
54	Descenso	SI
55	Ascenso	SI
56	Descenso	SI
57	Ascenso	SI
58	Descenso	SI
59	Ascenso	SI
60	Descenso	SI
61	Ascenso	SI
62	Descenso	SI
63	Ascenso	NO
64	Descenso	SI
65	Ascenso	SI
66	Descenso	SI
67	Ascenso	SI
68	Descenso	SI
69	Ascenso	SI
70	Descenso	SI
71	Ascenso	SI
72	Descenso	SI
73	Ascenso	SI
74	Descenso	SI
75	Ascenso	SI
76	Descenso	SI

77	Ascenso	SI
78	Descenso	SI
79	Ascenso	SI
80	Descenso	NO
81	Ascenso	SI
82	Descenso	SI
83	Ascenso	SI
84	Descenso	SI
85	Ascenso	SI
86	Descenso	NO
87	Ascenso	SI
88	Descenso	SI
89	Ascenso	SI
90	Descenso	SI
91	Ascenso	SI
92	Descenso	SI
93	Ascenso	SI
94	Descenso	SI
95	Ascenso	SI
96	Descenso	SI
97	Ascenso	SI
98	Descenso	SI
99	Ascenso	SI
100	Descenso	SI

Anexo k: Pruebas Secundarias Finales

Se realizaron pruebas en diferentes posiciones e inclinaciones, simulando el flujo de pasajeros para así comprobar el nivel de detección y el éxito en la captura de los eventos correspondientes, inicialmente, el prototipo 2 había sido diseñado para localizarse en la posición izquierda al ingresar al bus como se observa en la Ilustración 222, en esta no se tiene ningún grado de inclinación es decir, la captura se realiza completamente horizontal a la altura en la que se posiciona el DMC, al realizar el proceso de captura se obtuvo un muy bajo grado de reconocimiento, debido a factores de luminosidad, y de rango de captura, ya que solo en 3 de las 8 zonas era visible el perfil de la persona .



Ilustración 222 - Prueba posición lado izquierdo sin inclinación.

Posteriormente, se procedió a realizar pruebas de forma perpendicular al paso de personas como se puede observar en la Ilustración 223, ya que en este lugar se utilizaran las 8 zonas, sin embargo debido a la altura del bus donde se realizaron las pruebas, y que la mayoría de buses prestadores del servicio TPC son muy similares, se encontró que al ingresar una persona, la cabeza de esta ocupaba el 70% de la zona de captura, por lo cual, no se podría realizar ninguna acción de reconocimiento y seguimiento.



Ilustración 223 - Prueba posición perpendicular.

Posteriormente se ubicó el DMC en el lado derecho del ingreso de pasajeros, como se observa en la Ilustración 224, donde se obtuvo una mejora considerable en la detección del perfil de las personas, sin embargo, se comprobó que la zona da captura debido a los cambios de altura del bus, ya sea por las escaleras para descender o porque la altura del interior es de 175cm, no permite una captura en toda la zona de visibilidad, debido a que en las zonas donde se observa el interior del bus zonas 7 y 8 la iluminación es limitada y presentan más congestión, se determinó que sería las zonas a excluir en la captura.

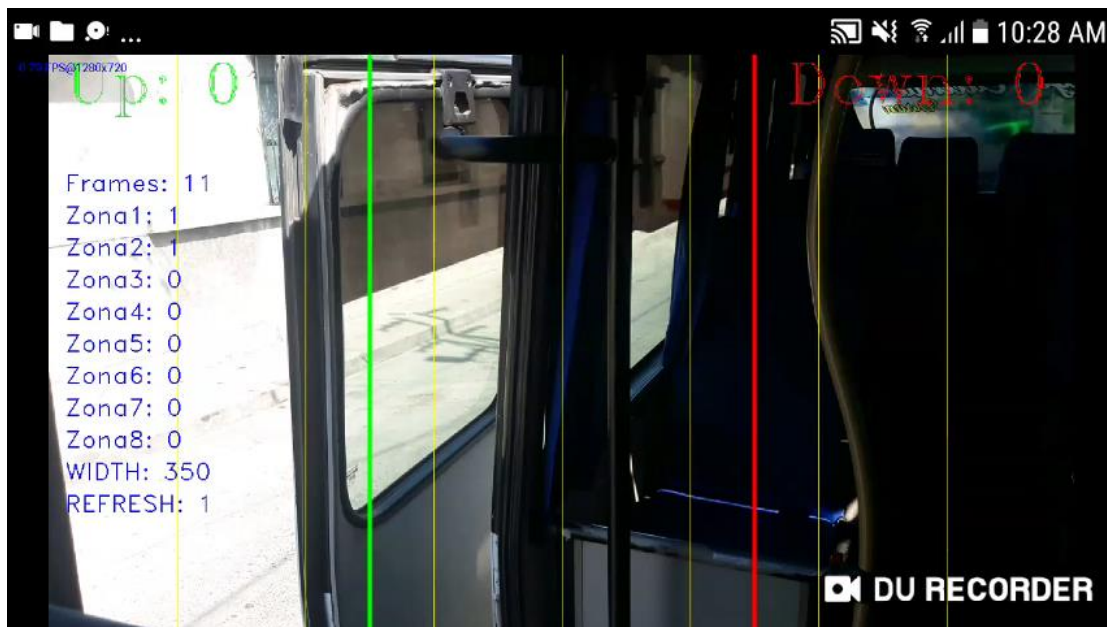


Ilustración 224 - Prueba posición lado derecho sin inclinación.

Por último, se encontró que dando una inclinación de entre 20° y 30° como se observa en la Ilustración 225, es decir posicionando el DMC en paralelo a la trayectoria que describen los pasajeros al ascender o descender del vehículo debido al desnivel por las escaleras, la captura aumentaba considerablemente, ya que se contaba con mayor zona de detección para el perfil de la persona.



Ilustración 225 - Prueba posición lado derecho inclinación 20°.