Módulo de enrutamiento de mensajes para la plataforma de atención de requisitos

Monografía para optar al título de Ingeniero de Sistemas
Modalidad práctica profesional

Claudia Yicel Caldon Manquillo

Director: Mag. Daniel Eduardo Paz Perafán
Codirector: Mag. Wilson Libardo Pantoja

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Popayán, diciembre de 2022

Tabla de contenido

Índice de tablas

Índice de figuras

Capítulo 1- Introducción

1.1 Planteamiento del problema

La Bolsa de Valores[1] es una institución autorregulada que se encuentra inscrita en la World Federation of Exchanges[2] ubicada en Londres, Inglaterra. La bolsa de valores junto con otras instituciones, empresas e individuos que realizan transacciones de productos financieros, integran lo que se conoce como el mercado bursátil[3], el cual tiene como objetivo prestar a los intermediarios inscritos todos los servicios necesarios para poder realizar eficazmente las transacciones con valores de manera transparente, continua y ordenada [1]. Como beneficios, la bolsa de valores permite que las empresas financien sus proyectos y actividades con la venta de sus activos o títulos, y brindan a los inversionistas la posibilidad de introducir capital en las empresas a través de la compra de los activos o títulos en venta. Por otra parte, cuando se habla de intermediarios se hace referencia a sociedades comisionistas de valores o de bolsa; sociedades fiduciarias; sociedades administradoras de fondos de pensión y cesantías; algunas entidades de naturaleza pública; entidades bancarias; entre otras, las cuales trabajan bajo una serie de normas y controles [2].

La comunicación de los intermediarios implicados en la bolsa de valores está soportada por diversos sistemas software. Algunas comunicaciones se presentan cuando: (i) es ofrecido un valor a través de los sistemas de negociación, (ii) son comprados los valores ofrecidos, (iii) se realiza un registro electrónico de la compra o venta de valores. (iv) se ejecutan consultas de los emisores inscritos en la bolsa de valores, (v) se requiere información de precios y estadísticas del mercado en tiempo real, entre otras comunicaciones que se pueden presentar entre los intermediarios [2].

Una de las empresas que desarrolla aplicaciones software para el mercado bursátil, es PWP Software SAS, una empresa colombiana la cual tiene 10 años de experiencia en este sector. La empresa a través de las aplicaciones que desarrolla brinda la información necesaria para que el inversionista tome decisiones que le ayuden a obtener ganancias y a anticipar los riesgos que se puedan presentar. La empresa cuenta con un grupo de profesionales con amplia experiencia en el mercado bursátil, proveniente de actividades al interior de la Bolsa de Occidente y entidades financieras.

---

[1] Valor: títulos representativos o anotaciones en cuenta de participación en sociedades, de cantidades prestadas, de mercaderías, de depósitos y de fondos monetarios, futuros, opciones, etc., que son objeto de operaciones mercantiles. Tomado de la RAE.

[2] World Federation of Exchanges: es una organización internacional de bolsas y cámaras de compensación de todo el mundo, fundada en 1961 con el propósito de apoyar los mercados de valores organizados y regulados y para satisfacer las necesidades de los mercados de capitales del mundo en el mejor interés de sus usuarios [14].

[3] Mercado bursátil: es la integración de todas aquellas instituciones, empresas o individuos que realizan transacciones de productos financieros, entre ellos se encuentran la bolsa de valores, casas corredoras de bolsa de valores, emisores, inversionistas e instituciones reguladoras de las transacciones que se llevan a cabo en la bolsa de valores [15].

En este orden de ideas, PWP Software SAS busca fortalecer su posición en el mercado bursátil como proveedor de servicios de software, pero presenta inconvenientes para lograr la interoperabilidad entre las diferentes aplicaciones que soportan el mercado bursátil y otras industrias. A través de una serie de entrevistas realizadas a los integrantes de la empresa, se determinaron las siguientes dificultades: (i) la heterogeneidad del sistema operativo, lenguaje de programación y hardware que constituyen los sistemas a interoperar, (ii) sistemas legados (legacy) que no ofrecen una interfaz para su comunicación, (iii) las organizaciones a las cuales pertenecen los sistemas que desean interoperar, manejan una sintaxis y semántica propia de los datos que pueden ser comunicados, sin seguir un estándar o propuesta homogénea, (iv) manejo de un volumen de información elevado lo cual produce sobrecarga en las aplicaciones y entre otras.

En este sentido, para solventar los anteriores inconvenientes, la empresa ha planteado el desarrollo de una plataforma[4] que soporte la distribución masiva de información entre diferentes aplicaciones empresariales y servicios web que pertenecen a diversos intermediarios de la bolsa de valores. La plataforma busca la resolución de requerimientos de interoperabilidad de los clientes y el intercambio de mensajes en tiempo real. Dentro de la empresa la plataforma se ha denominado como Plataforma de Atención de Requerimientos PAR.

PAR es un software distribuido que proporciona una infraestructura para integrar aplicaciones. Esta infraestructura tendrá las siguientes características: (i) permitirá configurar la lógica de enrutamiento de la información de las diferentes aplicaciones asociadas, (ii) ofrecerá una API que permite que aplicaciones escritas en diferentes lenguajes puedan comunicarse. Cuando la empresa PWP Software SAS necesite crear un proyecto software, dentro del cual hay intercambios de información entre diferentes aplicaciones asociadas, los desarrolladores podrán mediante PAR configurar y soportar la comunicación.

Dado que PAR busca integrar diferentes aplicaciones y trabajar con altos volúmenes de información en tiempo real, la velocidad y la transmisión de los datos debe ser fundamental, por lo cual se ha optado por utilizar la tecnología de Apache Kafka. Esta tecnología de código abierto permite transmitir grandes cantidades de datos con baja latencia, soporta la publicación y suscripción a eventos, el almacenamiento y procesamiento en tiempo real, con el fin de favorecer la velocidad, fiabilidad y escalabilidad en el sistema [3][4][5]. En este sentido, durante el desarrollo se priorizó el siguiente atributo de calidad: la interoperabilidad.

[4] Plataforma: la plataforma incluye un conjunto de componentes software y reglas que orientan la interoperabilidad entre diversos sistemas. Adaptado de Kevin Boudreau London Business School [16].

Debido a que la plataforma PAR es muy amplia, debe cumplir con muchos requisitos, con el fin de poder construir y definir los requerimientos de la plataforma PAR que se va a desarrollar, la empresa PWP Software SAS propuso 3 módulos, los cuales son:

- Módulo de enrutamiento de mensajes. Este módulo en términos generales se encargóde ofrecer las siguientes funcionalidades:
  - Permitir la configuración de temas (topics).
  - Ofrecer una interfaz para la conexión de emisores y receptores que desean intercambiar información a través de PAR.
  - Enrutar los mensajes mediante Apache Kafka.

- Módulo de seguridad. Este módulo se presenta en principio, para establecer las políticas de seguridad necesarias para la plataforma PAR y sus componentes, es evidente que el manejo, como el flujo de información es un tema importante en lo que respecta a la seguridad de los datos, por esta razón, se tuvo en consideración lo siguiente:

  - Asegurar que los servidores no sean fácilmente vulnerables o de fácil acceso, para quienes deseen acceder sin autorización a la información.
  - Determinar un almacenamiento seguro de la información suministrada dentro de PAR.
  - Definir protocolos de seguridad para la protección de la conexión a la red.
  - Configurar los tiempos de inactividad dentro de la aplicación, para evitar conexiones indefinidas a la plataforma.
  - Cifrado de la información, para garantizar la confiabilidad y confidencialidad de los datos, como son autenticación e intercambio de mensajes.
  - Establecer el control de acceso a los servicios, funcionalidades y acceso a la información, de acuerdo a los permisos otorgados en la configuración, según los roles indicados.

- Módulo de Despliegue continuo. Este módulo será el encargado de la integración y despliegue de plataformas, está conformado principalmente por las siguientes funcionalidades:

  - Integración con el repositorio remoto GitHub con el fin de almacenar el código fuente.
  - Posibilidad de configurar una plataforma PAR, para que se ejecute en modo pruebas, desarrollo y producción.

- Proporcionar un conjunto de funciones que permitan desplegar una determinada versión de una plataforma PAR, en un determinado ambiente deproducción configurado en Microsoft Azure.

Teniendo en cuenta los módulos definidos, todo lo relacionado con el intercambio de información entre los componentes internos, el presente proyecto se enfocó en el desarrollo del módulo de enrutamiento de mensajes, que permite configurar y externos de la plataforma PAR. Debido a las condiciones del proyecto este se desarrolló tomando elementos de la metodología Scrum, que permitió obtener mejores resultados en la entrega de las funcionalidades del módulo. Los repositorios del front-end y del back-end donde encontramos el código son los siguientes:
https://bitbucket.org/pwpadmin/repo-pwp-par/src/master/
https://bitbucket.org/pwpadmin/repo-pwp-par-ui/src/master/
Los repositorios donde encontramos el código del productor y consumidor son los siguientes:
https://bitbucket.org/ccaldono/repo-pwp-par-consumer/src/master/
https://bitbucket.org/ccaldono/repo-pwp-par-productor/src/master/

1.2 Objetivos

1.2.1 Objetivo general

Desarrollar una aplicación web que permita el enrutamiento de mensajes para la plataformade atención de requisitos de la empresa PWP Software, con el fin de lograr la comunicación entre aplicaciones empresariales y servicios web asociados a la bolsa de valores.

1.2.1 Objetivos específicos

- Identificar las necesidades asociadas al enrutamiento de mensajes, a partir de la aplicación de un conjunto de técnicas para la captura[5] de requisitos funcionales y nofuncionales.

- Diseñar e implementar la aplicación web que permita el enrutamiento de mensajes, considerando los requisitos funcionales y no funcionales identificados.

- Evaluar el cumplimiento de los requisitos de la aplicación web desarrollada medianteun conjunto de pruebas de sistema y usuario.

[5] En esta etapa se realiza el descubrimiento de requisitos (captura), escritura de los requisitos (especificación), resolución de conflictos (negociación), orden de implementación de requisitos (priorización), y se garantiza que las necesidades son especificadas adecuadamente (validación).

1.2 Metodología

Para el desarrollo de la práctica profesional se utilizaron elementos del marco de trabajo Scrum, se plantearon 3 sprints para lograr el cumplimiento de cada uno de los objetivos propuestos. Se definieron cuatro etapas de trabajo con un número determinado de actividades, de las cuales en las últimas dos etapas de evaluación y despliegue no se desarrollaron algunasactividades.

Actividades realizadas:

Identificación de las necesidades:

1. Identificación del contexto general de trabajo de la empresa PWP Software SAS, del mercado bursátil y de la plataforma distribuida Kafka.
2. Identificación de los involucrados en el desarrollo de la plataforma PAR y del módulo de enrutamiento de mensajes.
3. Identificación de las técnicas para capturar los requisitos funcionales y no funcionales.
4. Elicitación[6] de los requisitos funcionales y no funcionales de la plataforma a desarrollar, utilizando un conjunto de técnicas que ayuden a este proceso.

Diseño e implementación del módulo:

5. Capacitación y entendimiento de las diferentes tecnologías para el desarrollo de PAR y el módulo de enrutamiento de mensajes.
6. Diseño de la arquitectura y diseño detallado de la plataforma PAR y del módulo de enrutamiento de mensajes utilizando el modelo C4[7] a partir de los requisitos elicitados.
7. Priorización de las funcionalidades a desarrollar.
8. Planificación de los sprints estableciendo: funcionalidades que se implementarán en cada sprint y fechas de reuniones con el cliente para entrega de los resultados de cada sprint.
9. Implementación del módulo de enrutamiento de mensajes de acuerdo a los requisitos establecidos en la planeación de cada sprint.

[6] Elicitación: Proceso mediante el cual se descubren las necesidades y propiedades de un sistema software, tales como especificación, negociación, validación y verificación de requisitos a partir de la comunicacióncon los usuarios y todos los implicados del sistema.

[7] Modelo c4: es un enfoque de abstracción primaria que permite diseñar la arquitectura de software de un sistema. Mediante un conjunto jerárquico de diagramas para contexto, contenedores, componentes y código (opcional) [17].

10. No se pudo lograr la integración del proyecto dado que los estudiantes de los módulosde seguridad y despliegue continuo se retiraron del proyecto.

Evaluación:

11. Creación de los casos de prueba a partir de los requisitos de la aplicación.
12. Ejecución de los casos de prueba, registrar y analizar resultados, comparar resultados obtenidos con esperados y repetir las actividades de prueba según la acción que se lleve a cabo para cada discrepancia.
13. Realización del informe final de pruebas.

Actividades no realizadas:

Evaluación:

14. Corrección de algunas funcionalidades que no pasaron los casos de prueba.

Despliegue:

15. Configuración del entorno (sistema operativo, paquetes, herramientas de desarrollo yservidor web) para despliegue de los componentes de la aplicación web desarrollada.
16. Despliegue de la aplicación web en el entorno configurado.
17. Realización de pruebas de sistema y usuario en la aplicación web desplegada.

1.4 Estructura de documento

La presente monografía organiza su contenido de la siguiente manera:

- Capítulo 2: esta sección presenta los conceptos teóricos que guiaron la práctica profesional, además de la descripción de las tecnologías usadas para implementar la aplicación.
- Capítulo 3: esta sección presenta las actividades realizadas para levantar los requisitos, especificación de los requisitos mediante historias de usuario y prototiposy finalmente la arquitectura definida para la aplicación.
- Capítulo 4: esta sección presenta las diferentes funcionalidades implementadas de lashistorias de usuario para el rol Administrador.
- Capítulo 5: en esta sección se presentan las conclusiones, lecciones aprendidas y trabajos futuros.

Capítulo 2- Conceptos y Tecnologías aplicadas

A continuación, se definirán los conceptos fundamentales para la comprensión del presente trabajo de grado y herramientas principales que se utilizaron en el desarrollo de la aplicación. Además de algunos atributos de calidad esenciales para el funcionamiento adecuado de las aplicaciones software.

2.1 Herramientas tecnológicas utilizadas

Para el desarrollo del presente proyecto se utilizaron un conjunto de tecnologías para la implementación del front-end, back-end, base de datos, despliegue de la aplicación y pruebas. Las principales tecnologías se describen a continuación:

SPA (single-page application): es una implementación de aplicación web que carga solo un único documento web y, a continuación, actualiza el contenido del cuerpo de ese único documento a través de API de JavaScript como XMLHttpRequest y Fetch cuando se va a mostrar contenido diferente [18].

Angular: es una plataforma de desarrollo de una sola página basada en TypeScript (lenguaje de programación de código abierto), la cual fue usada para el desarrollo front-end de la aplicación. Debido a que angular permite el desarrollo de aplicaciones empresariales e incluye bibliotecas que se integran y se actualizan sin problemas, la empresa PWP Software sugirió utilizarla para el desarrollo de las interfaces gráficas de la aplicación [7].

Apache Kafka: es una plataforma de transmisión de eventos en tiempo real de cualquier aplicación tipo software, es usada en ambientes de trabajo con gran cantidad de volumen de información, garantizando así la distribución eficiente de datos de forma ágil.[5] En Kafka un evento se traduce en el hecho de que "algo sucedió" en el mundo o en su negocio, por lo que la transmisión de eventos es el equivalente al sistema nervioso central del cuerpo humano. Técnicamente hablando, la transmisión de eventos es la práctica de capturar datos en tiempo real de fuentes de eventos como bases de datos, sensores, dispositivos móviles, servicios en la nube y aplicaciones de software. La transferencia de eventos asegura así un flujo continuo y una interpretación de los datos para que la información correcta esté en el lugar correcto, en el momento correcto [12] Kafka al ser una plataforma de transmisión de eventos tiene tres características claves para su debido funcionamiento:

1. Permite publicar (escribir) flujos de eventos y suscribirse (leer) a ellos.
2. Permite almacenar las transmisiones de eventos de forma duradera y confiable durante el tiempo que se desee.

3. Permite procesar flujos de eventos a medida que ocurren.

Kafka al ser un sistema distribuido, consta de servidores y clientes. Kafka se ejecuta como un clúster de servidores que se encargan de resolver las peticiones de los clientes y abarcar varios centros de datos, lo que permite estabilidad y tolerancia a fallos. Kafka como cliente permiten escribir aplicaciones distribuidas y microservicios que leen, escriben y procesan flujos de eventos en paralelo y tolerante a fallas.

Elementos principales

En la *figura 2.1* se pueden apreciar los elementos principales de la plataforma Kafka, los cuales están definidos como:

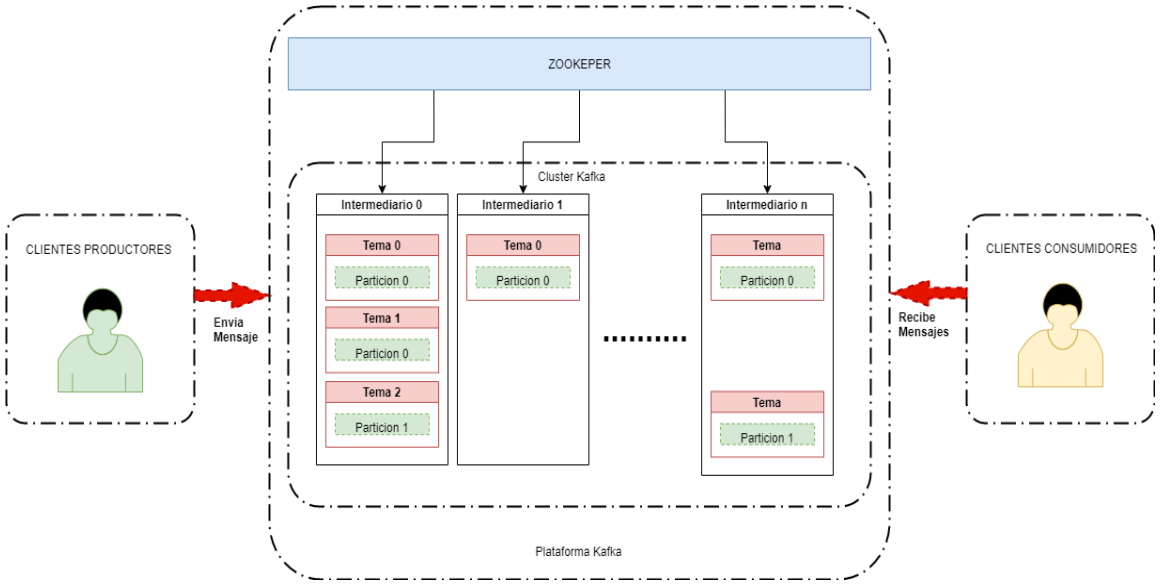

Figura 2.1 Arquitectura general de Kafka, adaptado de [13].

Productores: Aplicaciones clientes que publican (escriben) eventos en Kafka.

Consumidores: Aquellos que se suscriben (leen y procesan) a los eventos de Kafka.

Topics (Temas): Un topic permite una fácil comunicación entre publicadores y suscriptores. Un topic se encarga de almacenar los mensajes escritos por uno o más productores, permitiendo que estos sean leídos por uno o más consumidores.

Records (registros): En Kafka, un evento (llamado también mensaje) de publicación-suscripción se denomina registro (record). Un registro consta de un par clave/valor y metadatos.

Partitions (particiones): En lugar de que todos los registros manejados por el sistema se almacenen en un solo topic, Kafka divide los registros en particiones. Las particiones se pueden considerar como un subconjunto de todos los registros de un topic. Los topics están divididos, lo que significa que un topic se distribuye en varios "depósitos" ubicados en diferentes brokers de Kafka.

Brokers (Intermediario): Un broker es en sí un servidor de Kafka que atiende las solicitudes de los consumidores y almacena los mensajes enviados a los topics. Kafka está diseñado para ejecutarse en varios hosts, con un broker por host. Si un host se desconecta, Kafka hace todo lo posible para asegurarse de que los demás hosts continúen ejecutándose. Todos los brokers de Kafka hablan con Zookeeper el cual se encarga de gestionar los brokers de Kafka, manteniendo así un listado de brokerscon sus respectivos metadatos.

ZooKeeper: Es un software que proporciona un servicio de coordinación de alto rendimiento para las aplicaciones distribuidas. En Kafka proporciona un servicio centralizado para la gestión de la configuración de: registro de cambios (creación de un topic, caída de un broker, etc.) e intercambio de metadatos en los brokers.

Spring Framework: es una plataforma Java de código abierto que principalmente soporta el patrón inyección de dependencias y la programación orientada a aspectos, con los cualeslogran implementar servicios como transacciones, seguridad, logging entre otros, para ser aplicados en múltiples componentes. Los principales módulos del Framework son los siguientes [6]:

- Contenedor de núcleo: está compuesto de los módulos Core, Beans, Context y Expression Language. Proporcionan Inyección de dependencias e inversión de control.
- Web: compuesto por los módulos Web, Web-Servlet, Web-Struts y Web-Portlet. Permite crear controladores Web, tanto de vistas MVC como aplicaciones REST.
- Acceso a datos/integración: compuesto de los módulos JDBC, ORM, OXM, y Transaction. Proporciona abstracciones sobre JDBC; ORMs como JPA, JDO, Hibernate e iBatis; sistemas OXM (Object XML Mappers); Servicio de mensajería de java y transacciones.
- Programación orienta a aspectos (POA): proporciona una implementación de programación orientada a aspectos.
- Instrumentación: proporciona soporte de instrumentación de clases e implementaciones de cargadores de clases para su uso en ciertos servidores de aplicaciones.

- Pruebas de código: admite la prueba de componentes Spring con JUnit o TestNG.

Spring Boot: es el encargado de facilitar el desarrollo de aplicaciones independientes basadas en Spring, funciona como un módulo que simplifica las tareas de selección de dependencias con Maven y el despliegue en el servidor, enfocándose principalmente en la creación y ejecución de aplicaciones en Spring. Spring Boot ofrece soporte embebido a contenedor de Servlet. Un Servlet es una clase de lenguaje de programación Java utilizada para ampliar las capacidades de un servidor.
Se usó Spring Boot porque es una tecnología apoyada en todo el framework de desarrollo Spring, permite el desarrollo de aplicaciones de manera eficiente debido a la configuración que tiene por defecto, está optimizada para altas cargas de trabajo con un mínimo consumo de memoria.

Node.js: es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado enel motor V8 de Google [19].

Java: es una plataforma informática de lenguaje de programación creada por Sun Microsystems en 1995. Java es una plataforma fiable en la que se generan muchos servicios y aplicaciones. Este lenguaje de programación se utilizó para desarrollar el back-end [13].

Maven: es una herramienta de gestión y comprensión de proyectos de software. Basado en el concepto de un modelo de objetos de proyecto (POM), Maven puede administrar la compilación, los informes y la documentación de un proyecto a partir de una pieza central deinformación [14].

MySQL: es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual, licencia pública general y licencia comercial, por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo [20]. MySQL se utilizó para almacenar la información de la aplicación.

Git: es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código [19].

2.2 Conceptos

Aplicaciones empresariales

La aplicación desarrollada se considera una aplicación empresarial debido a que es una colección de componentes que proporciona una funcionalidad empresarial que se puede utilizar internamente, externamente o con otras aplicaciones empresariales [9]. Según [10] entre las características generales de las aplicaciones empresariales se encuentran: la persistencia de datos, manejo de gran cantidad de información, acceso concurrente a datos, uso de gran cantidad de interfaces dado a los diferentes tipos de usuarios, integración con otras aplicaciones empresariales. Dado a lo anterior, las aplicaciones empresariales permiten a las empresas facilitar las tareas, ahorrar costos y generar más ingresos.

Atributos de calidad planteado por la norma ISO 25010

Es esencial que un producto software sigan ciertos atributos de calidad que ayuden a cumplir el objetivo de la empresa y que determine las características de calidad que se tendrán en cuenta al evaluar el producto. Los atributos de calidad apoyados en este proyecto fueron los siguientes: la interoperabilidad, la seguridad y eficiencia de desempeño [11].

Capítulo 3 - Requisitos y arquitectura

En este capítulo se presentan las actividades seguidas para realizar el levantamiento de los requisitos, las técnicas utilizadas, descripción de los aportes de las iteraciones en los diagramas de secuencia a la definición de las historias de usuario. Se presentarán las historias épicas identificadas y las historias de usuario y criterios de aceptación asociados a cada una. Se presentarán los prototipos realizados antes del inicio del desarrollo y la descripción de la arquitectura de la aplicación.

3.1 Levantamiento de requisitos

Para definir los requisitos fueron realizadas las actividades de captura, especificación, negociación, validación y verificación propuestas por la ingeniería de requisitos. En la actividad de captura participaron los stakeholders descritos en la Tabla 1.

Tabla 1 Descripción de los Stakeholders

Nombre	Rol	Características
Irme David Mosquera Bermudez	Stakeholders	• Participar en la revisión de avances del producto. • Interesados en el desarrollo del producto. • Verificar y apoyar las actividades.
Angela Patricia Fernández Pelaez		
Carlos David Mosquera		
Roberto Encarnación		

En la actividad de captura se utilizaron las técnicas de entrevistas, análisis documental y creación de diagramas de secuencias. Las entrevistas planteadas se encuentran en el Anexo A y se construyeron a partir del entendimiento global de las necesidades. En la actividad de especificación se emplearon historias de usuario, criterios de aceptación y prototipos para representar los requisitos.

En la actividad de negociación se encontraron unas discrepancias entre el deseo de los stakeholders, necesidades de la organización y requisitos planteados. La resolución de las discrepancias fue realizada en múltiples reuniones.

En la actividad de verificación de requisitos se utilizó la técnica de revisión por pares, donde los estudiantes mostraron la especificación de los requisitos a los directores para verificar que estuvieran correctos, completos y no ambiguos.

En la actividad de validación se mostraron a los stakeholders los requisitos identificados para verificar que estuvieran alineados a sus deseos y necesidades.

3.2 Rol del Administrador en la aplicación

En la captura de requisitos se identificaron a nivel global 4 roles que interactúan en la aplicación, pero en el módulo de enrutamiento de mensajes solo interactúan 3 roles, Administrador, Consumidor y productor. En la tabla 2 se puede ver los roles.

Tabla 2 Descripción de los Roles del sistema.

Descripción de roles identificados	
Administrador	Realiza las configuraciones de la plataforma PAR.
Desarrollador	Encargado de manejar un entorno con herramientas, ya incluidas, para el desarrollo de aplicaciones.
Consumidor	Software consume la información de un servicio existente en aplicación.
Productor	Software que publica la información en un servicio existente en aplicación

3.3 Historias de usuario

En total se generaron 47 historias de usuario agrupadas en 12 historias épicas que proveen todas las funcionalidades necesarias para el módulo de enrutamiento de mensajes. A partir de los requerimientos identificados, se construyeron las historias épicas, historias de usuarios y criterios de aceptación, que se podrán ver en detalle en el Anexo B.

3.3.1 Historias Épicas

En la tabla 3, muestra 12 historias épicas con las necesidades que se identificaron a partir de la captura de requisitos.

Tabla 3 Descripción de las historias épicas

#	Identificador	Nombre	Rol	Descripción
1	HE-01	Gestión de usuarios	Administrador	Gestionar los usuarios registrados en la aplicación "Plataformas de atención de requisitos-PAR"
2	HE-02	Gestión de plataformas	Administrador	Gestionar las plataformas creadas en la aplicación "Plataforma de atención de requisitos - PAR"
3	HE-03	Establecer rutas de conexión	Administrador	Establece las rutas de conexión.
4	HE-04	Gestión de brokers	Administrador	Gestionar los brokers creados para plataforma de atención de requisitos- PAR
5	HE-05	Gestión de topic	Administrador	Gestionar los topics creados en la Plataforma de atención de requisitos - PAR
6	HE-06	Gestión de servicios	Administrador	Gestionar los servicios creados en la plataforma PAR
7	HE-07	Gestión de Conexiones	Administrador	Gestionar las conexiones de productores o consumidores en la plataforma.
8	HE-08	Gestión de productores/ consumidores	Administrador	Gestionar los productores/consumidores en la aplicación "Plataformas de atención de requisitos-PAR"
9	HE-09	Iniciar sesión	Administrador, Desarrollador Productor, Consumidor	Permitir a los usuarios hacer uso de plataforma PAR.
10	HE-10	Servicios externos	Productor, Consumidor	Acceder a los servicios de la plataforma PAR
11	HE-11	Gestión de permisos	Administrador	Gestionar los permisos en la plataforma de atención de requisitos - PAR

3.3.2 Historias de usuarios

La figura 3.1 se muestra una historia de usuario HU-01 que describe la gestión de una plataforma, para ver el documento completo ir al **Anexo B.**

Figura 3.1. Descripción de HU-01 de Gestión de plataformas

	Enunciado de la Historia				Criterios de Aceptación				
Identificador (ID) de la Historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de Escenario	Criterio de Aceptación (Título)	Contexto	Evento	Resultado / Comportamiento esperado	Observaciones
HU-01	Yo como administrador.	necesito **registrar la información de una plataforma**	con la finalidad de ejecutar funcionalidades requeridas.	1	Agregar una nueva plataforma.	Dado que me encuentro en la vista "Gestión de plataformas" y quiero agregar una nueva plataforma	cuando doy clic en el botón "Agregar plataforma"	El sistema mostrará una ventana modal, la cual permite ingresar los datos básicos de una plataforma	Campo estado presenta las siguientes opciones: activo, inactivo, y el campo Tipo de acceso las siguientes opciones (público, privado).
				2	Nombre ya existe	Dado que me encuentro en la vista "Agregar plataforma" y ya se encuentra registrada una plataforma con el nombre ingresado.	Cuando he terminado de ingresar el nombre	el sistema indicará que la plataforma ya se encuentra registrada.	
				3	Datos incorrectos	Dado que me encuentro en la vista "Agregar plataforma" y he terminado de ingresar los datos en algunos de sus campos.	cuando ingreso caracteres no válidos en los campos del formulario.	el sistema indicará en cuál campo se ha ingresado caracteres no válidos, así mismo inhabilitará el botón de "Agregar".	En el campo, nombre se aceptarán únicamente caracteres alfabéticos. Para el campo versión números reales positivos. Y en los campos estado y rol se deberá seleccionar la opción que corresponda al usuario.
				4	Datos correctos	Dado que me encuentro en la vista "Agregar plataforma"	cuando he ingresado en todos los campos caracteres válidos y además he registrado datos en todos los campos obligatorios.	el sistema habilitará el botón "Agregar".	
				5	Agregar exitoso	Dado que me encuentro en la vista "Agregar plataforma" y el botón "Agregar" se encuentra habilitado.	cuando doy clic en "Agregar"	el sistema mostrará un mensaje indicando que se agregó exitosamente una plataforma	
				6	Ocurrió un error al intentar agregar una plataforma	Dado que he dado clic en "Agregar"	cuando ocurre un error al intentar almacenar la información en la base de datos.	el sistema mostrará un mensaje indicando que ocurrió un error al intentar guardar una nueva plataforma	
				7	Cancelar cambios	Dado que me encuentro agregando una nueva plataforma	cuando doy clic en "Cancelar"	el sistema redirigirá al usuario a la vista de "Gestión de plataformas" sin efectuar ningún cambio.	

3.4 Prototipos

Para la generación de los prototipos fue utilizado Balsamiq Mockups, el cual nos permitió crear vistas de manera interactiva, lo que permitió estilizar mejor los componentes y darles una vista muy cercana a los stakeholders de como luciría finalmente la aplicación.

Gestión de usuarios

Las historias de usuarios HU-01 hasta las HU-06 que se muestran en la tabla 4 estánasociadas al prototipo que se muestran en las figuras 3.2 y 3.3.

Tabla 4 Historias de usuarios de Gestión de usuarios

Historia épica	Historia de usuario	Funcionalidad
HE-01	HU-01	Registrar la información de un nuevo usuario.
	HU-02	Editar la información de un usuario.
	HU-03	Cambiar el estado de un usuario.
	HU-04	Listar los usuarios registrados.
	HU-05	Buscar un usuario registrado.
	HU-06	Ver información de un usuario.

Figura 3.2. Prototipo de Gestión de usuarios

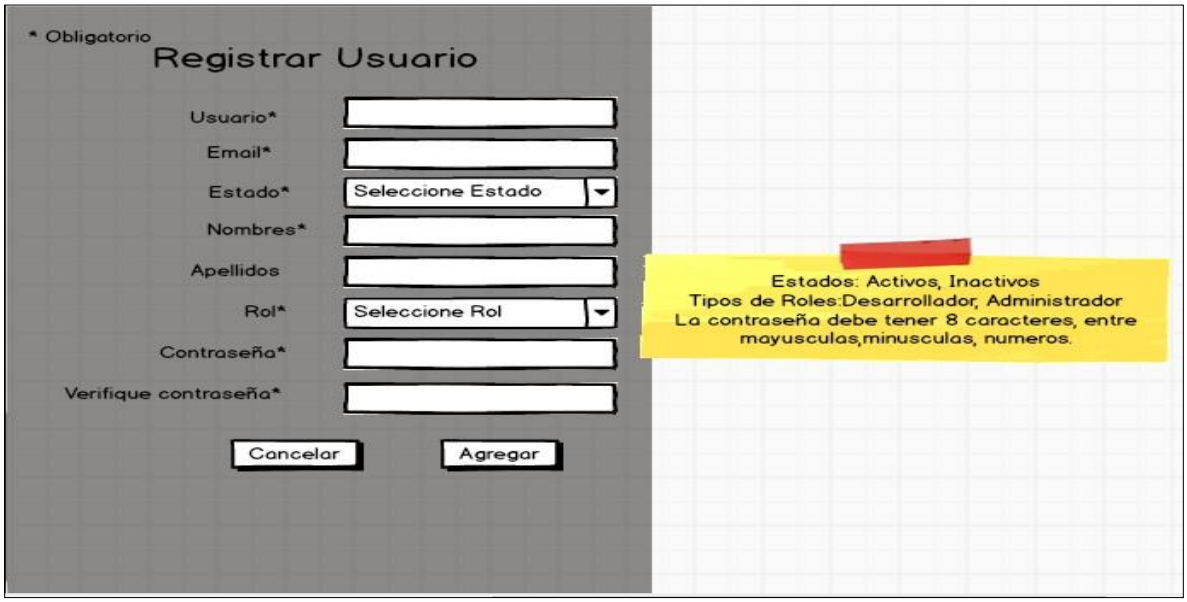

Figura 3.3 Prototipo Registrar usuario

Gestión de Plataformas

Las historias de usuarios HU-01 hasta las HU-06 que se muestran en la tabla 5 estánasociadas al prototipo que se muestran en las figuras 3.4, 3.5 y 3.6.

Tabla 5 Historias de usuarios de Gestión de plataformas

Historia épica	Historia de usuario	Funcionalidad
HE-02	HU-01	Registrar la información de una plataforma
	HU-02	Editar la información de una plataforma
	HU-03	Cambiar el estado de una plataforma
	HU-04	Listar las plataformas registradas.
	HU-05	Buscar una plataforma registrada.
	HU-06	Ver información de una plataforma.
	HU-07	Configurar una plataforma.

Figura 3.4. Prototipo de Gestión de plataformas para Administrador

Figura 3.5 Prototipo agregar plataforma para Administrador.

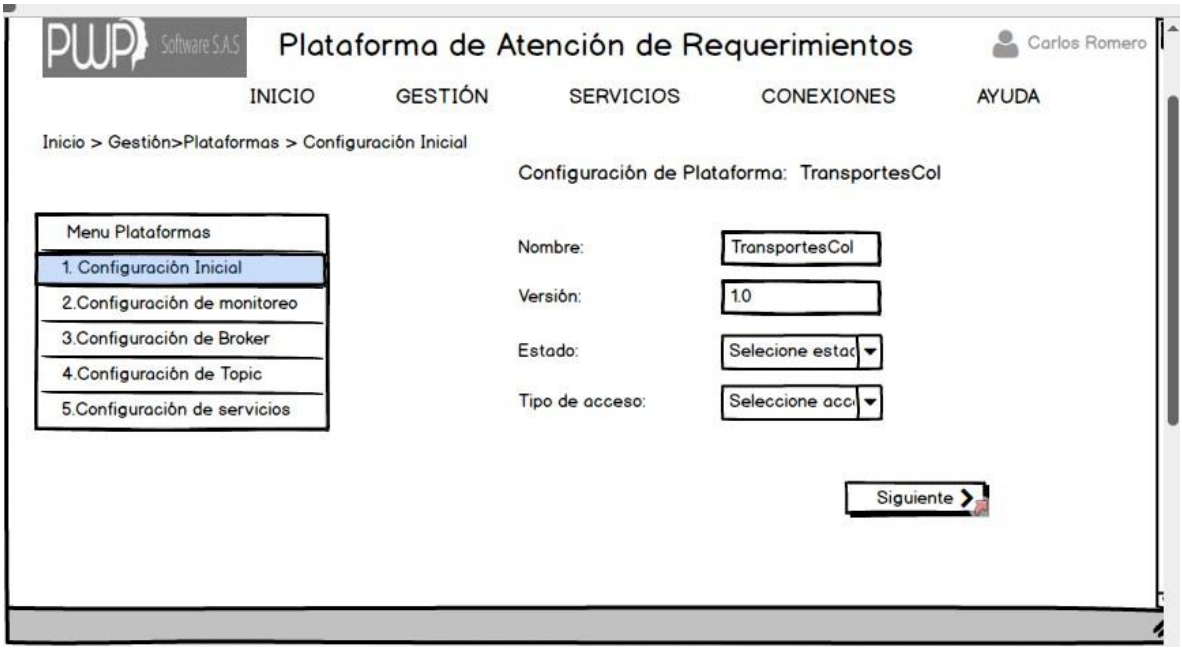

Figura 3.6 Prototipo configuración inicial de una plataforma para Administrador.

Gestión de brokers

Las historias de usuarios HU-01 hasta las HU-06 que se muestran en la tabla 6 están asociadas al prototipo que se muestran en las figuras 3.7 y 3.8.

Tabla 6 Historias de usuarios de Gestión de brokers

Historia épica	Historia de usuario	Funcionalidad
HE-04	HU-01	Registrar la información de un bróker.
	HU-02	Editar la información de un bróker.
	HU-03	Cambiar el estado de un bróker.
	HU-04	Listar los brokers registrados.
	HU-05	Buscar un bróker registrado.
	HU-06	Ver información de un bróker.

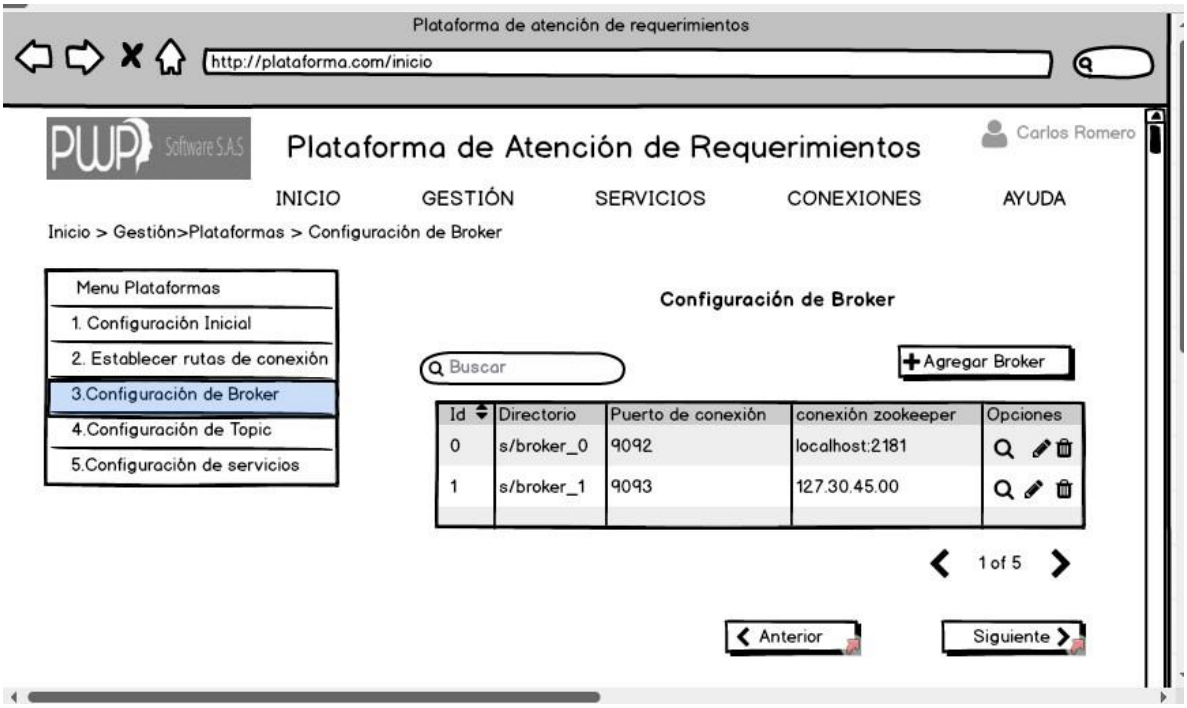

Figura 3.7. Prototipo de configuración de bróker para Administrador.

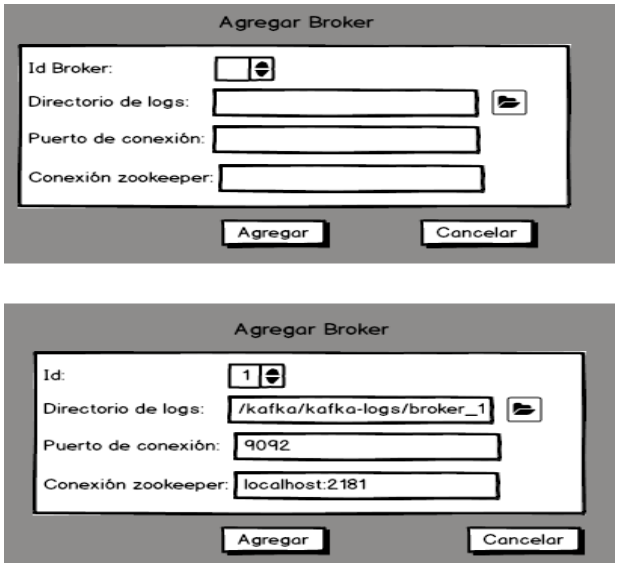

Figura 3.8 Prototipo agregar un nuevo bróker para Administrador.

Gestión de topics

Las historias de usuarios HU-01 hasta las HU-06 que se muestran en la tabla 7 están asociadas al prototipo que se muestran en las figuras 3.9 y 3.10.

Tabla 7 Historias de usuarios de Gestión de topics

Historia épica	Historia de usuario	Funcionalidad
HE-05	HU-01	Registrar la información de un broker.
	HU-02	Editar la información de un broker.
	HU-03	Cambiar el estado de un broker.
	HU-04	Listar los brokers registrados.
	HU-05	Buscar un broker registrado.
	HU-06	Ver información de un broker.

Figura 3.9 Prototipo configuración de topics para Administrador.

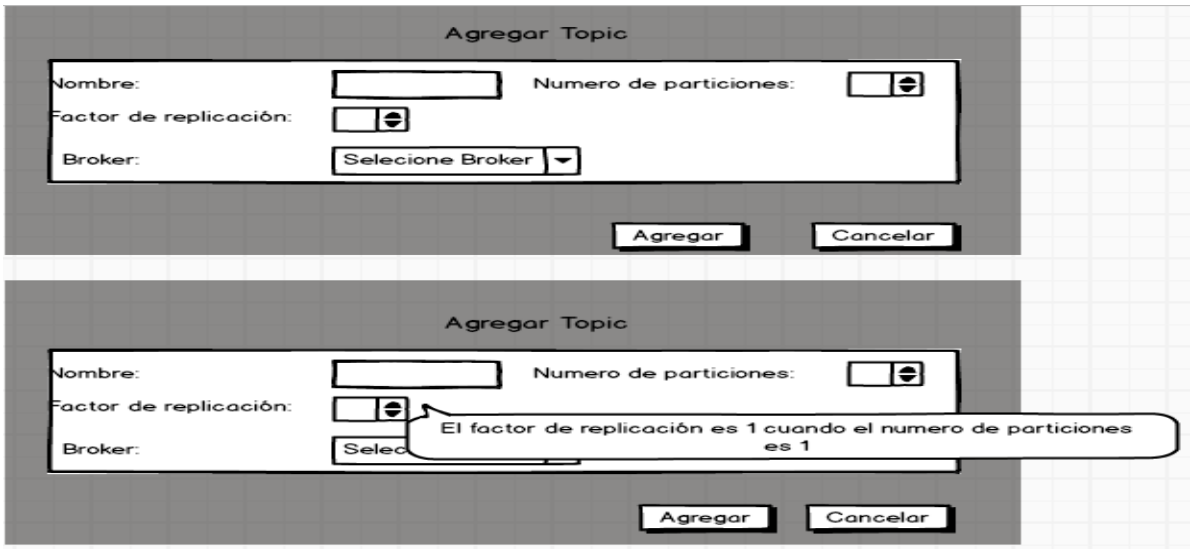

Figura 3.10 Prototipo agregar nuevo topic para Administrador.

Gestión de servicios

Las historias de usuarios HU-01 hasta las HU-06 que se muestran en la tabla 8 están asociadas al prototipo que se muestran en las figuras 3.11 y 3.12.

Tabla 8 Historias de usuarios de Gestión de servicios

Historia épica	Historia de usuario	Funcionalidad
HE-06	HU-01	Registrar la información de un servicio
	HU-02	Editar la información de un servicio
	HU-03	Cambiar el estado de un servicio
	HU-04	Listar los servicios registrados
	HU-05	Buscar un servicio registrado.
	HU-06	Ver información de un servicio.

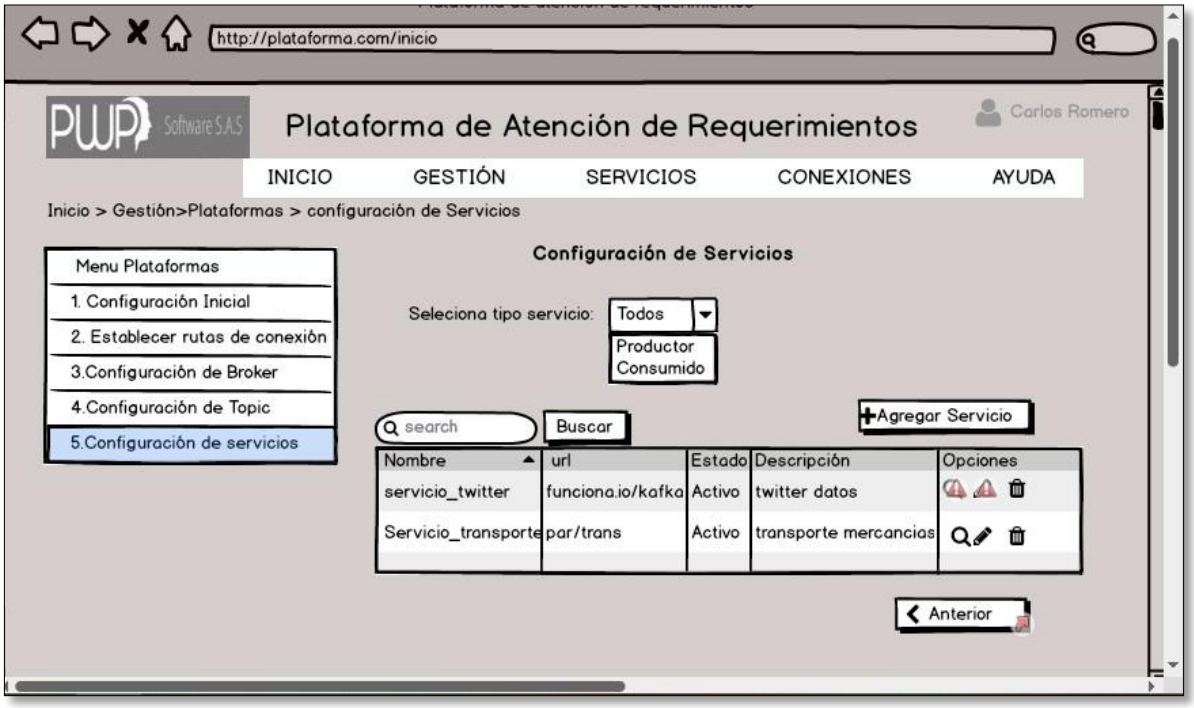

Figura 3.11 Prototipo de configuración de servicios para Administrador.

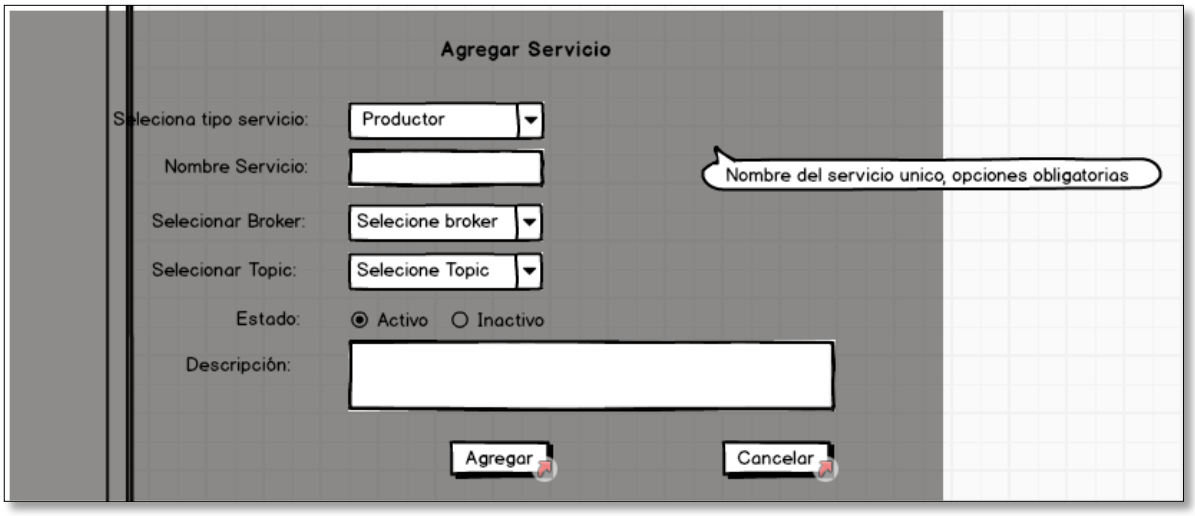

Figura 3.12 Prototipo de agregar un nuevo servicio para Administrador.

Gestión de productores y consumidores

Las historias de usuarios HU-01 hasta las HU-06 que se muestran en la tabla 9 están asociadas al prototipo que se muestran en las figuras 3.13 y 3.14.

Tabla 9 Historias de usuarios de Gestión de productores/consumidores

Historia épica	Historia de usuario	Funcionalidad
HE-08	HU-01	Registrar la información de un productor/consumidor
	HU-02	Editar la información de un productor/consumidor
	HU-03	Cambiar el estado de productor/consumidor
	HU-04	Listar los productores/consumidores registrados
	HU-05	Buscar un productor/consumidor registrado.
	HU-06	Ver información de un productor/consumidor

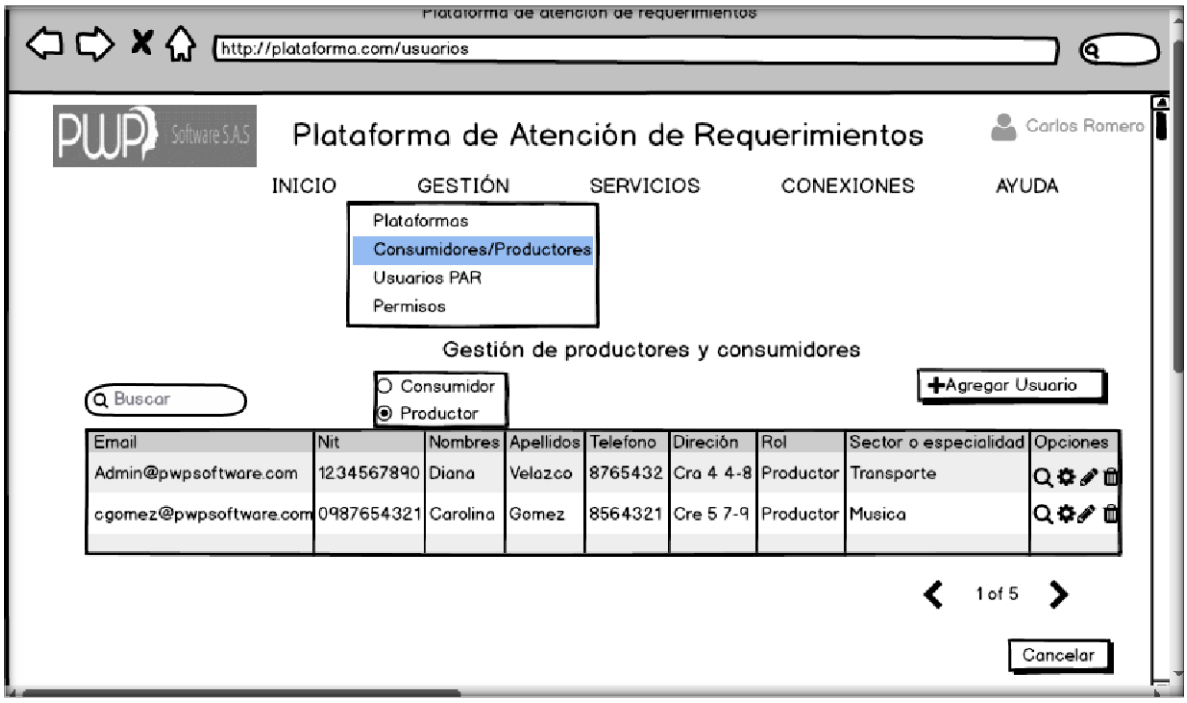

Figura 3.13 Prototipo de Gestión de productores y consumidores para Administrador.

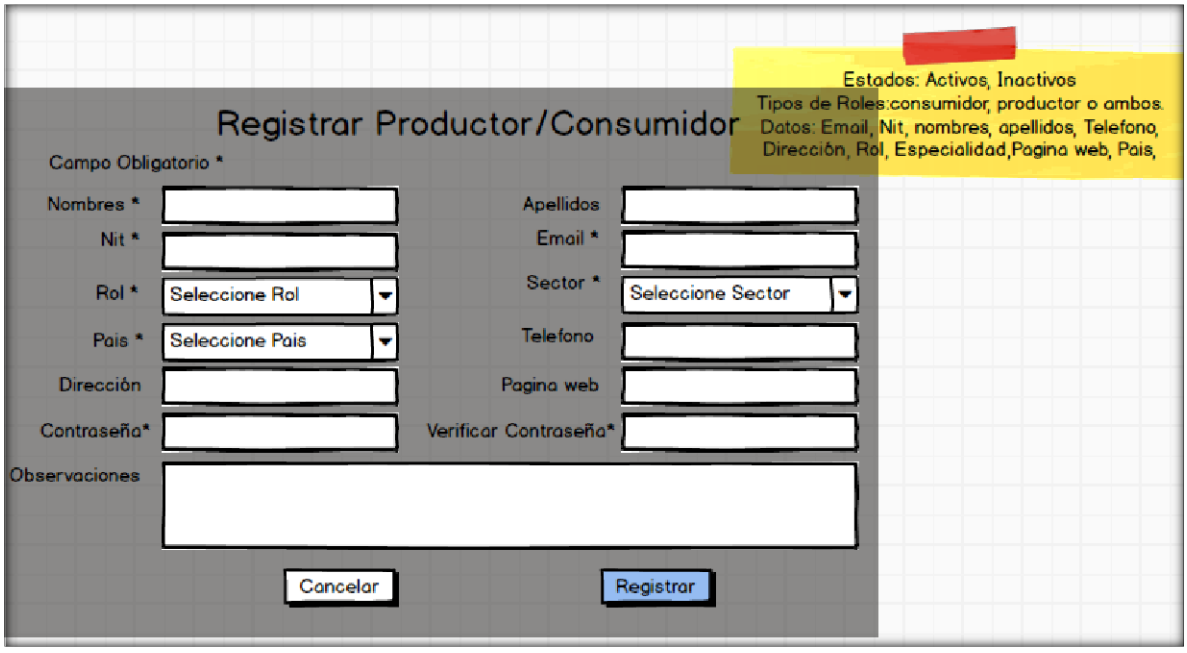

Figura 3.14 Prototipo de registrar un nuevo productor/consumidor para Administrador.

Gestión de permisos

Las historias de usuarios HU-01 hasta las HU-06 que se muestran en la tabla 10 están asociadas al prototipo que se muestran en la figura 3.15.

Tabla 10 Historias de usuarios de Gestión de permisos

Historia épica	Historia de usuario	Funcionalidad
HE-11	HU-01	Registrar la información de un permiso
	HU-02	Editar la información de un permiso
	HU-03	Cambiar el estado de un permiso
	HU-04	Listar los permisos registrados
	HU-05	Buscar un permiso registrado.
	HU-06	Ver información de un permiso.

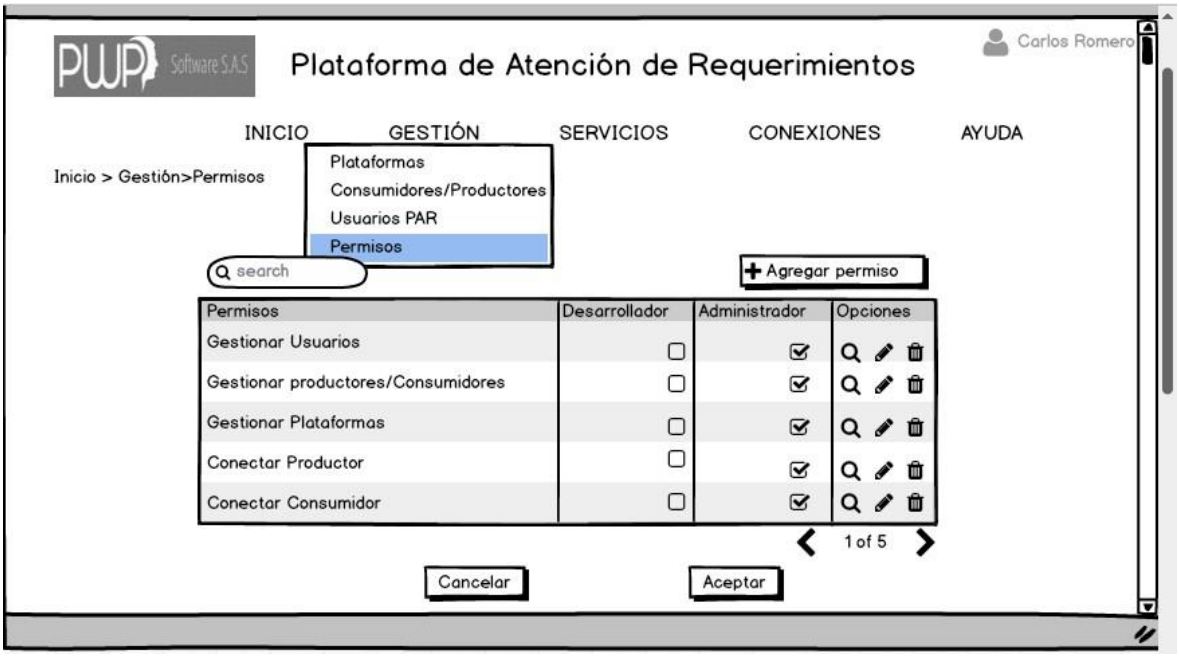

Figura 3.15 Prototipo de Gestión de permisos para Administrador.

3.5 Requisitos no funcionales

Stack tecnológico:

- El código fuente será gestionado por GIT versión 2.22.0 o superior.
- Front-end:
    - El front-end debe ser desarrollado en el framework con nombre Angular, versión 12 o superior.

    - El front-end debe ser desarrollado con el lenguaje de programación connombre TypeScript.
- Back-end:
    - El back-end debe ser desarrollado en el framework con nombre Spring.
    - El back-end debe ser desarrollado con el lenguaje de programación con nombre Java, versión 14 o superior.

Independencia lógica de negocio, medio de persistencia y presentación:

- La lógica de negocio del producto debe ser independiente de la presentación con la cual interactúa el usuario, de tal manera que cuando se agreguen nuevos tipos de presentación, o se modifique la tecnología de la presentación, sea posible utilizar la lógica de negocio del producto sin estar obligado a modificarla.
- La lógica de negocio del producto software debe ser independiente del medio de persistencia, de tal manera que cuando se cambie el medio de persistencia o la tecnología del medio de persistencia, la lógica de negocio del producto no cambie.

Usabilidad:

- Toda funcionalidad del sistema y transacción de negocio debe responder al usuarioen menos de 5 segundos.

Interoperabilidad:

- Intercambio de información entre sistemas. Se buscó que los productores no conocieran a los consumidores a través de la plataforma mediante el uso de apache Kafka.

3.6 Diseño

La arquitectura de la aplicación se definió en la parte final del levantamiento de requisitos, esta arquitectura se planteó desde diferentes perspectivas mediante el modelo C4. La arquitectura se planteó haciendo uso de 4 diagramas considerando los requisitos funcionalesy no funcionales.

Nivel 1: Diagrama de contexto del sistema

Se define como un modelo de alto nivel que muestra las interacciones de la aplicación con personas (administrador, desarrollador) y otros sistemas que intervienen en el proceso de intercambio de información. Este modelo se visualiza en tiempo de ejecución.

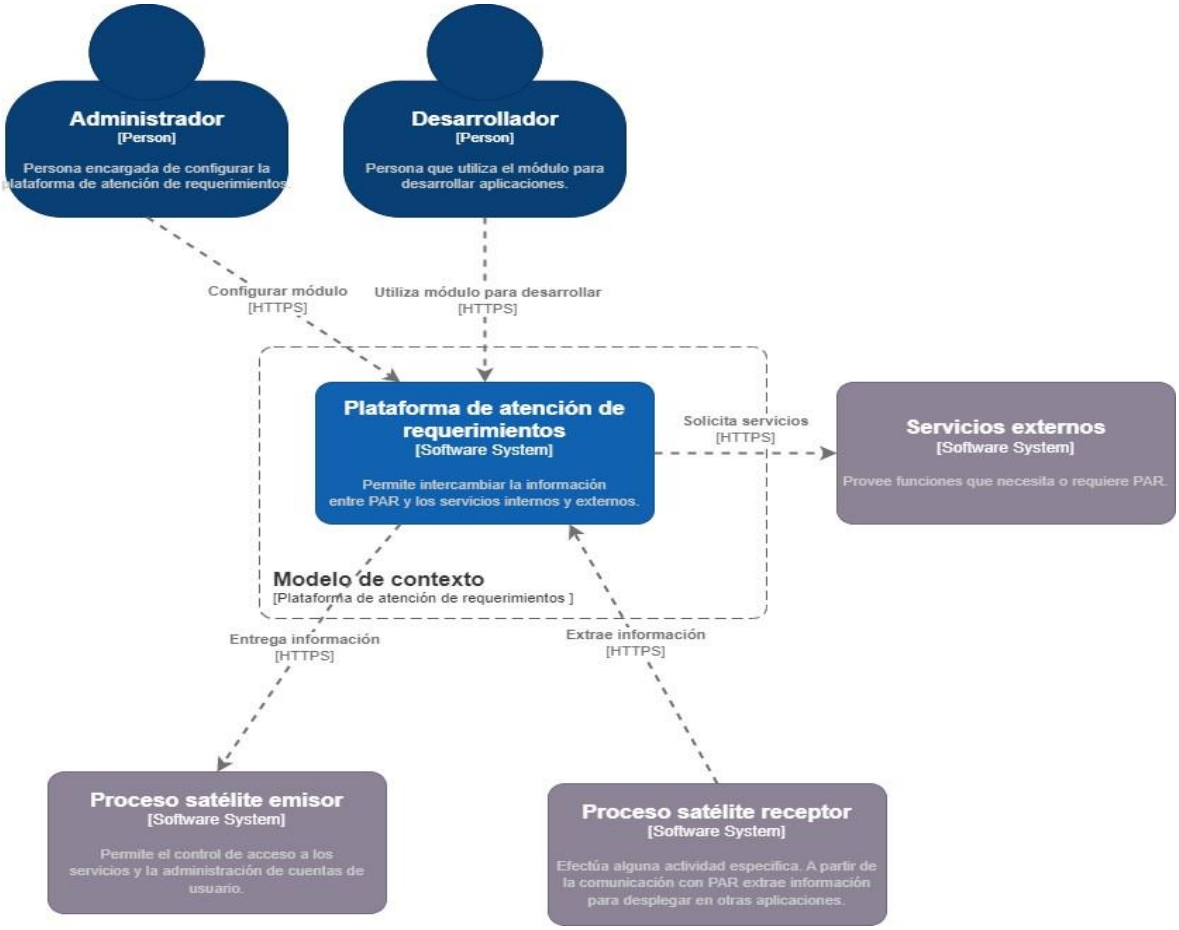

Figura 3.16 Diagrama de contexto del sistema.

En la figura 3.16, se muestran los diferentes sistemas con los que interactúa la aplicación de atención de requerimientos y que se describen a continuación:

Plataforma de atención de requerimientos: como un sistema software que proporciona una infraestructura que permite el intercambio de la información entre sistemas mediante el uso de apache Kafka.

Servicios externos: se encarga de proporcionar la información que requiere o le interesa a la aplicación.

Procesos satélites emisores: son procesos que envían información asociada a la bolsa de valoresa la plataforma.

Procesos satélites receptores: son procesos que reciben información proveniente de la plataforma.

Entre las personas que interactúan con el sistema, está el administrador, que es el encargado de realizar las configuraciones con la que aplicación se va a ejecutar.

Nivel 2: Diagrama del contenedor

En este nivel 2 o diagrama de contenedores, se muestra el sistema de la plataforma de atención de requerimientos con los contenedores que lo conforman: Interfaz (GUI), Seguridad, Bases de datos, Módulo de despliegue continuo, Módulo de enrutamiento de mensajes, plataforma Kafka y Archivo de texto kafka como se observa en la figura 3.17

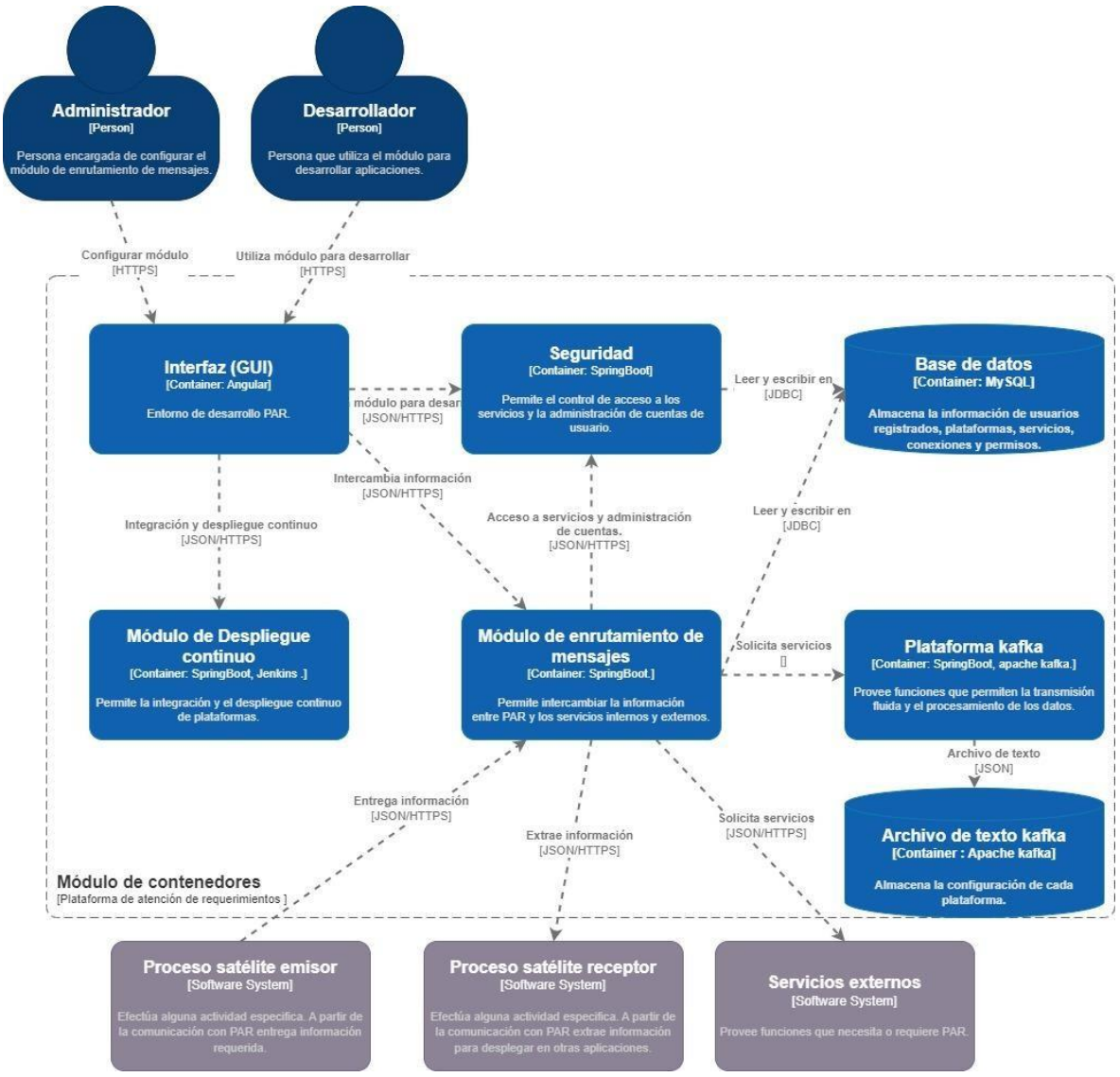

Figura 3.17 Diagrama de contenedores.

Interfaz (GUI): Este contenedor representa una interface de alto nivel, la cual se encarga de realizar la comunicación con todos los servicios del sistema.

Seguridad: Este contenedor se encarga de establecer las políticas de seguridad necesarias para la plataforma, como permitir el control de acceso a los servicios, encriptar y desencriptar los mensajes que se transportan sobre la red.

Bases de datos: Almacena la información de usuarios, plataformas, servicios, conexiones y permisos.

Módulo de despliegue continuo: Permite la integración y el despliegue continuo de la plataforma.

Módulo de enrutamiento de mensajes: Permite intercambio de información entre diferentes sistemas.

Plataforma Kafka: permite publicar, almacenar y procesar los flujos de registros y suscribirse a ellos en tiempo real.

Archivo de texto Kafka: Almacena la información relacionada con la configuración de la plataforma Kafka.

En esta práctica profesional solo se abordaron los siguientes contenedores: Interfaz GUI, Módulo de enrutamiento de mensajes, plataforma apache Kafka, Base de datos y archivo de texto Kafka, en otras prácticas profesionales se piensan abordar los otros contenedores.

Nivel 3: Diagrama de componentes

En el presente diagrama se expande el contenedor del Módulo de enrutamiento de mensajes,que es sobre el cual se enfocó el desarrollo de esta práctica profesional, en la figura 3.18 se observa los componentes que contiene.

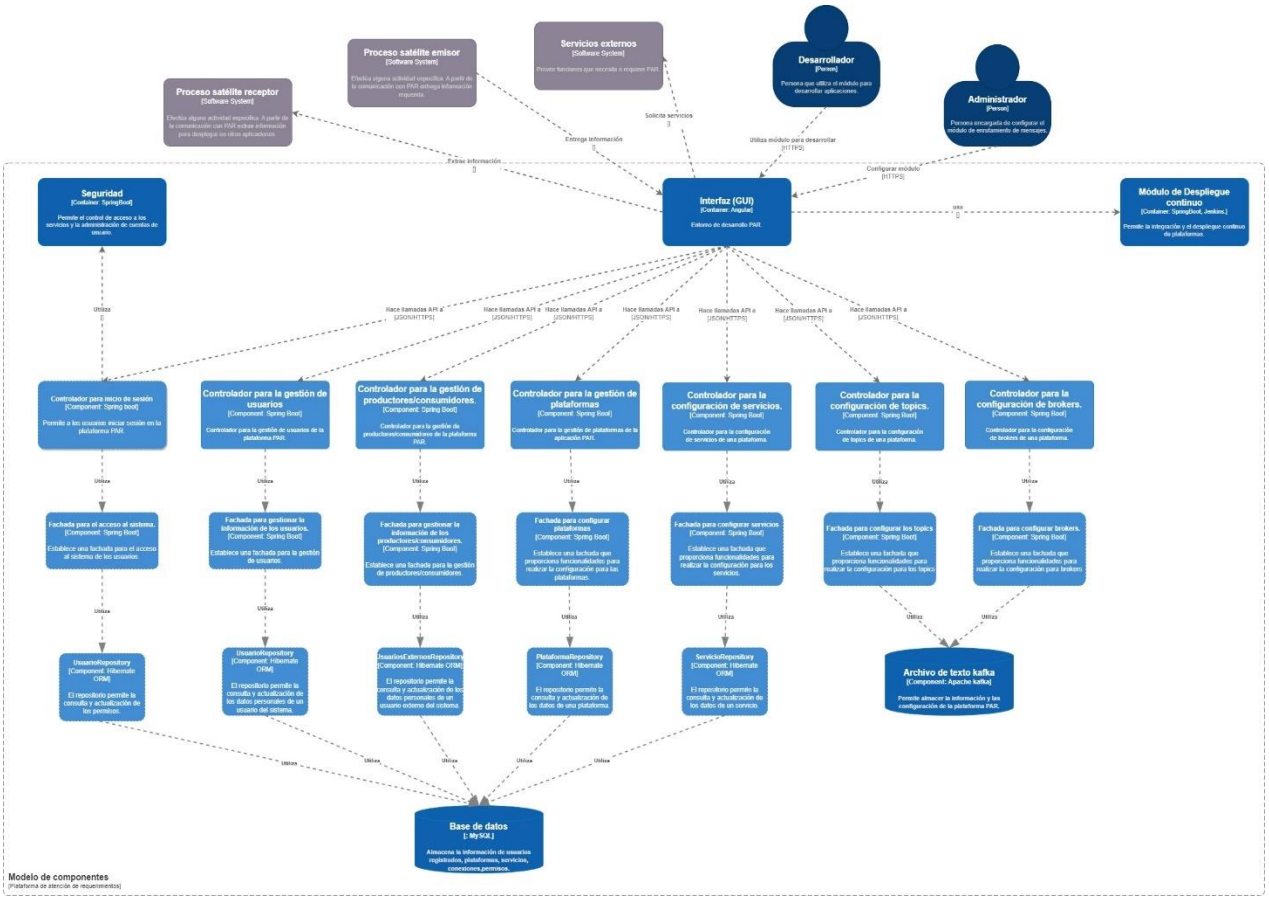

Figura 3.18 Diagrama de componentes.

El modelo de componentes se utilizó los patrones de diseño: fachada, repositorio, inyección de dependencias, y el MVC.

- Fachada: es un patrón de diseño estructural que proporciona un acceso a un parte específica de la funcionalidad de un sistema. En la figura 3.19, se observa una interfaz que define los métodos para la gestión de una plataforma.

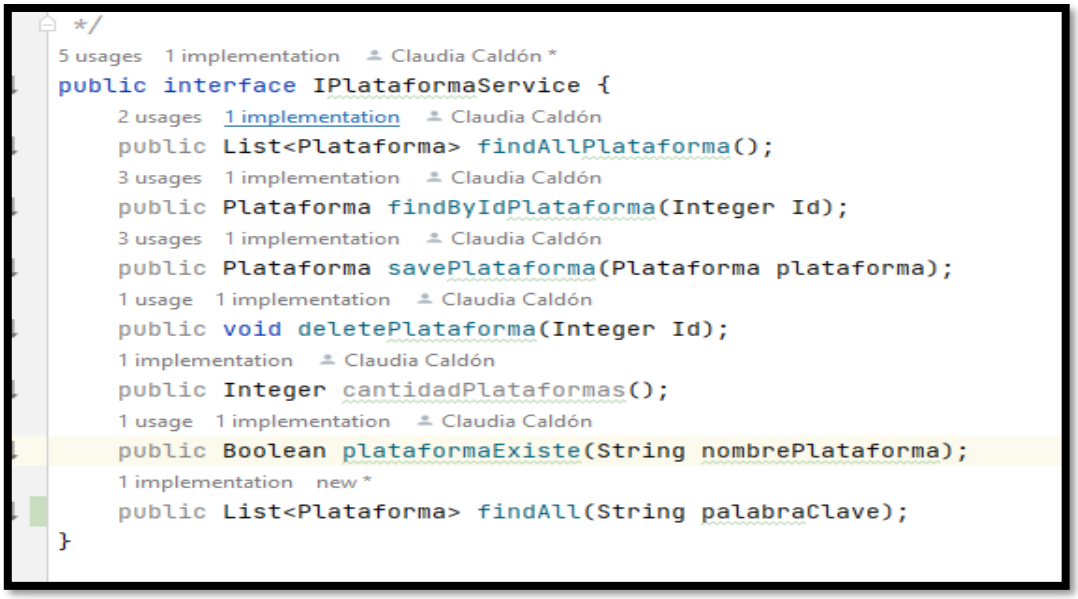

Figura 3.19 Representación del patrón fachada.

- Repositorio: este patrón actúa como intermediario entre la lógica del negocio y la lógica de acceso a los datos. En la figura 3.20, se observa una definición de una interfaz repositorio que extiende de CrudRepository el cual recibe como parámetros una entidad (Plataforma) y el tipo de parámetro del ID permitiendo acceso a las funciones CRUD.

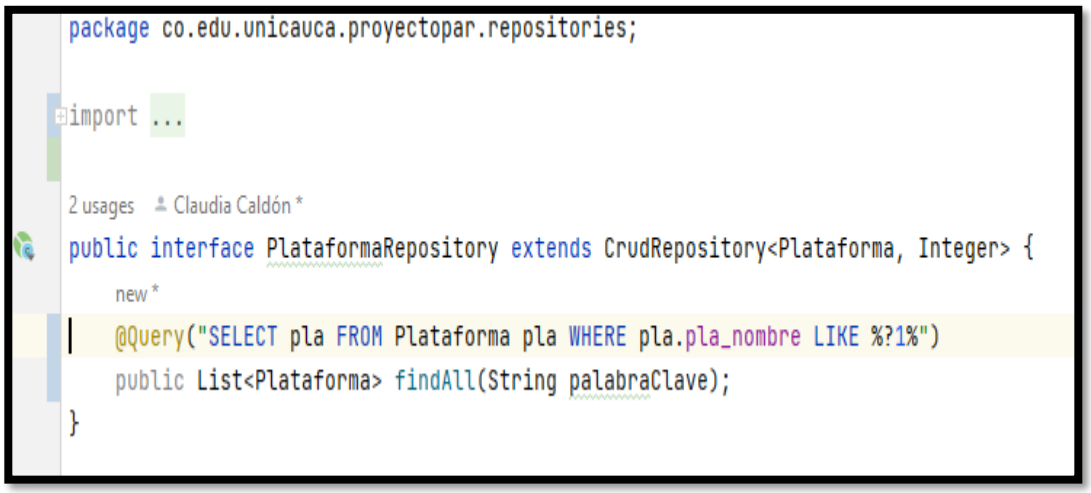

Figura 3.20 Interfaz repositorio para plataforma.

- **Inyección de dependencias:** Patrón de diseño orientado a objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase la que cree dichos objetos. En la figura 3.21 se puede observar un caso de inyección de dependencias por campo, que se realiza añadiendo la anotación @Autowired en el campo de la clase que

queremos su dependencia.

Figura 3.21 Inyección de dependencias basada en campos.

- **MVC:** es un patrón de arquitectura de software que separa los datos de la aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

  Modelo: Contine la representación de los datos que maneja el sistema y la lógica de negocio. En la figura 3.22 se observa la clase plataforma que pertenece al modelo.

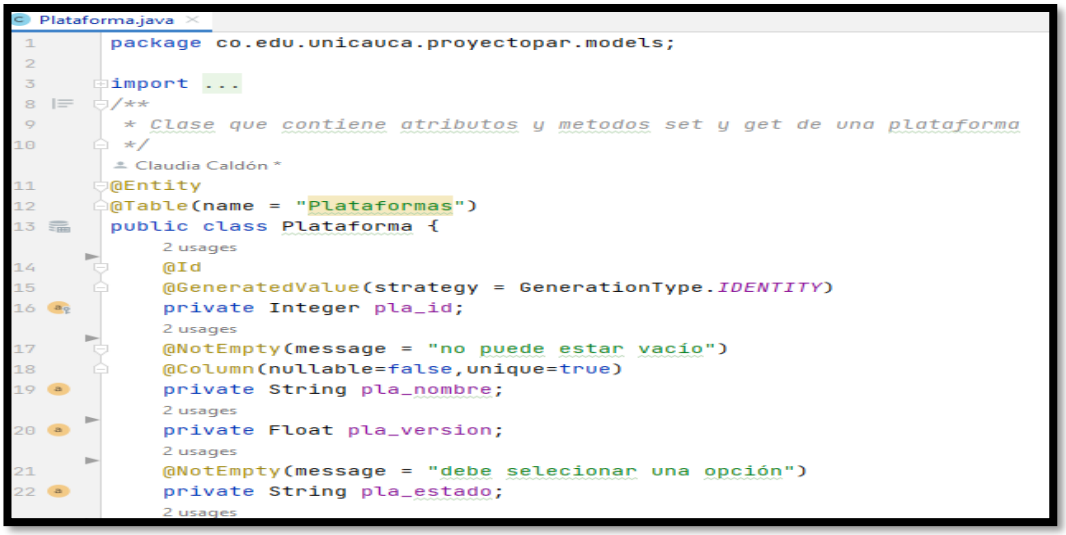

Figura 3.22 Representa el modelo

Vista: Interfaz gráfica que se le presenta al usuario. En la figura 3.23 se observa parte

del código HTML desarrollado en Angular.

```
<div class="form-outline col-sm-4" >
  <input type="search"  class="form-control " name ="campoDeBusqueda" [(ngMod
</div>
<button type="button" class="btn btn-primary btn-sm"(click)="this.buscar()" >
  <i class="bi bi-search"></i>
</button>
</div>

<div *ngIf="plataformas?.length==0" class="alert alert-info">
   No hay registros en la base de datos!
</div>

 <table class= "table table-bordered table-striped" *ngIf="plataformas.length>0">
     <thead>
         <tr>
             <th class="hidden">Id</th>
             <th>Nombre</th>
             <th>Version</th>
             <th>Estado</th>
             <th>Fecha de creación</th>
             <th>Tipo de acceso</th>
             <th>Opciones</th>
         </tr>
```

Figura 3.23 Código HTML

Controlador: Actúa como intermediario entre el modelo y la vista, recibe las órdenes del usuario, solicita los datos al modelo y se los comunica a la vista. En sprint boot una clase se denota como controlador utilizando la anotación @Restcontroller como se observa en la figura 3.24.

```
package co.edu.unicauca.proyectopar.controllers;

import ...
/**
 * Este clase controladora contiene operaciones crud para una plataforma.
 */
 Claudia Caldón +1 *
@CrossOrigin(origins = "http://localhost:4200",  maxAge = 3600)
@RequestMapping("/api")
@RestController
public class PlataformaRestController {
    10 usages
    @Autowired
    private IPlataformaService plataformaService;
    7 usages
    @Autowired
    private IArchivoService archivoService;
    /**
     * lista todas las plataformas
     * @return Lista de plataforma
     */
     Claudia Caldón
    @GetMapping("/plataformas")
    public List<Plataforma> listarplataformas() {...}
```

Figura 3.24 Clase controladora

La siguiente figura 3.25 muestra la estructura que se elaboró en el back end para la aplicación:

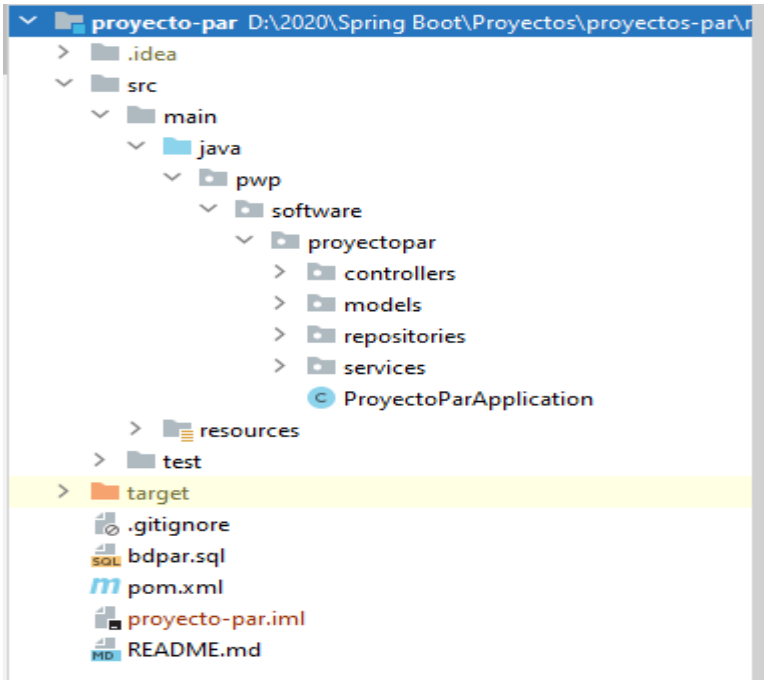

Figura 3.25 Estructura del back-end

A continuación, se describen cada uno de los paquetes que la componen:

Controllers: Paquete donde se crean las clases que se encargan de procesar las peticiones y las respuestas HTTP.
Models: Paquete donde se crean las clases que representan el modelo de datos.
Repositories: Paquete donde se crean las clases que establecen la comunicación con la base de datos.
Services: Paquete donde se crean las clases que tiene como fin hacerla la implementación de los métodos que se definan para la aplicación.

La siguiente figura 3.26 se muestra la estructura que se elaboró en el front-end para la aplicación:

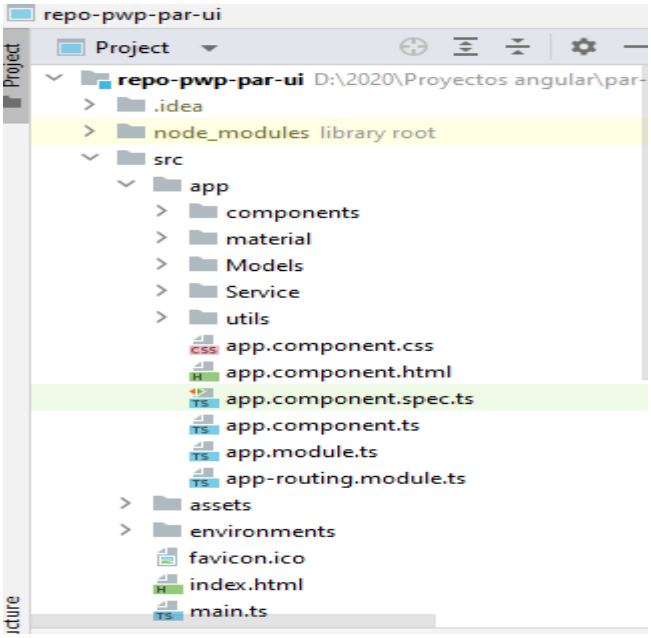

Figura 3.26 Estructura del front-end

A continuación, se describe los paquetes que lo componen:

Components: Paquete donde se crea el HTML que se va a mostrar.
Material: Paquete que contiene la clase donde se declaran los componentes para utilizar Angular Material
Models: Contiene las interfaces que representan cada una de las entidades de la aplicación.
Services: Paquete donde se encuentran las clases que van a hacer llamada a los end –point o Api rest.
Utils: Paquete que contine funciones o ayudas comunes (correo único, password iguales).

Nivel 4: El código

En el diagrama de la figura 3.19 se muestra la representación de los datos por medio de tablas relacionadas, conformando así una base de datos. En total, el modelo está constituido por 9 tablas.

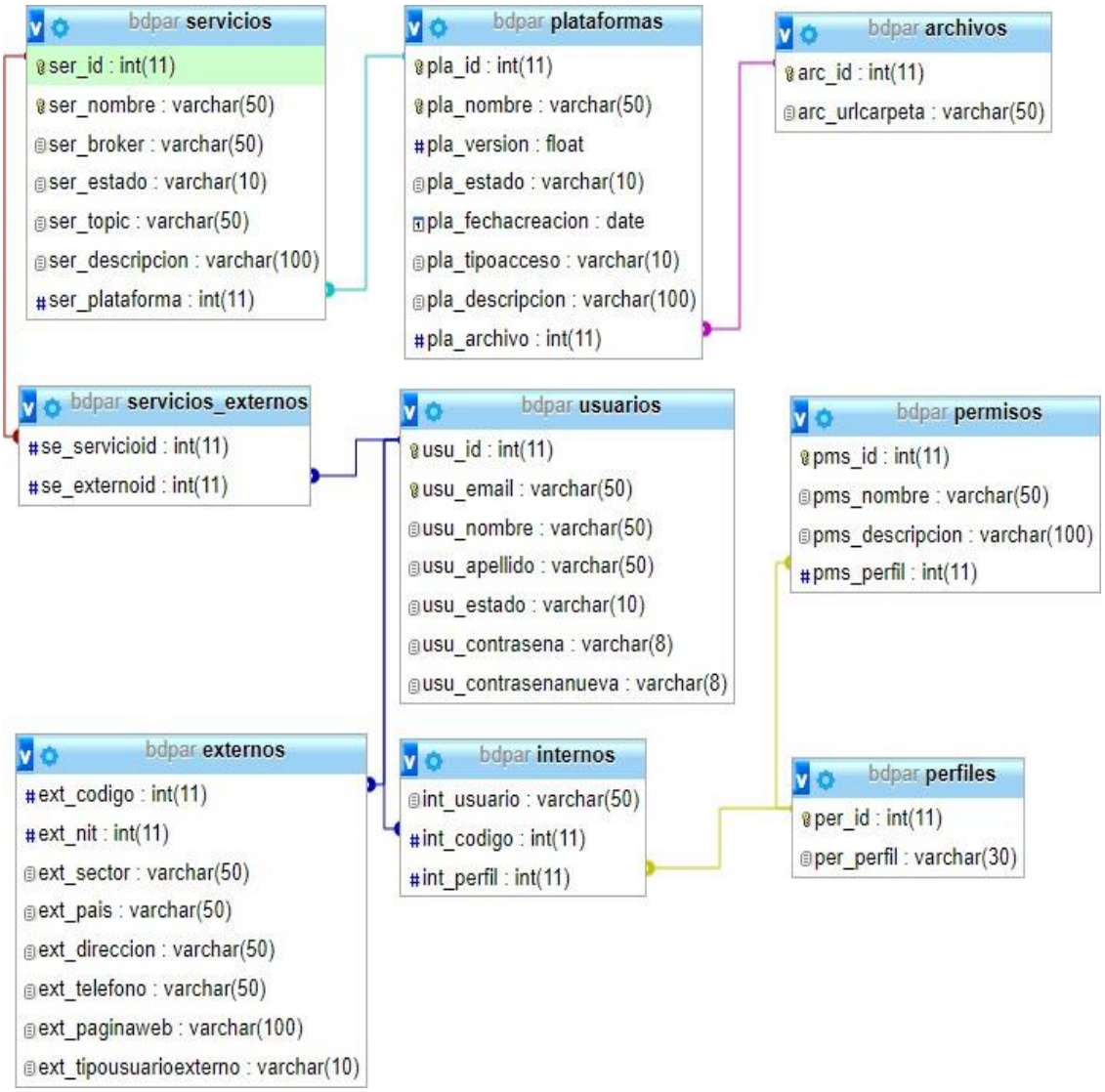

Figura 3.27 Diagrama relacional

Capítulo 4 - Implementación y funcionalidades

En este capítulo se presentan las actividades seguidas para desarrollar la aplicación, descripción de las funcionalidades principales del módulo de la aplicación desarrollada.

4.1 Componentes implementados

A continuación, mostramos la implementación de las funcionalidades principales del módulode enrutamiento de mensajes, asociados al Rol Administrador.

Funcionalidades del rol administrador

Menú principal

En la Figura 4.1 se muestra el menú principal para el Administrador, donde se marcan las opciones a la que puede acceder:

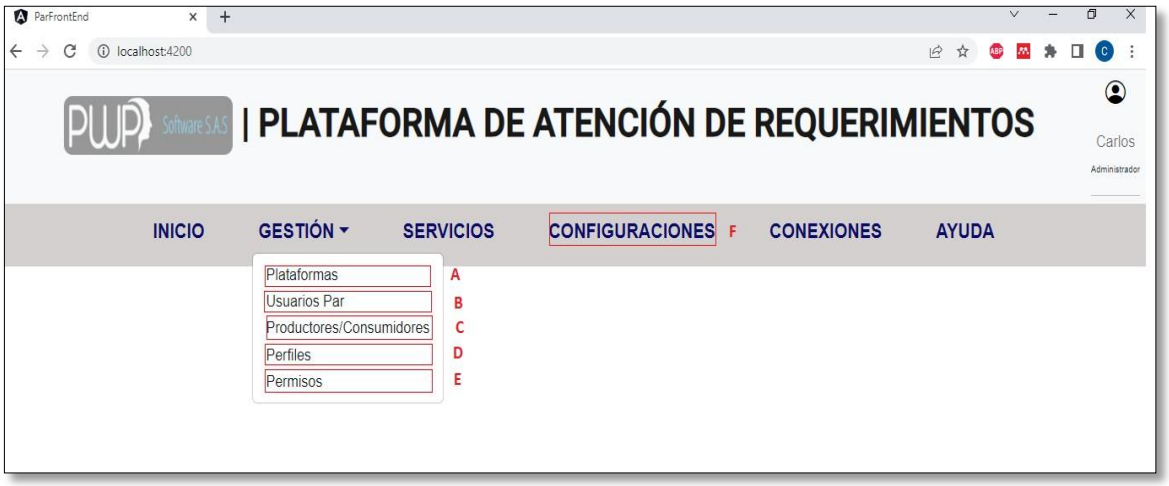

Figura 4.1 Menú principal para Administrador

A. Gestión de plataformas.
B. Gestión de usuarios.
C. Gestión de productores y consumidores.
D. Perfiles
E. Gestión de Permisos
F. Configuración de la plataforma.

Gestión de plataformas

En la Figura 4.2 se muestra la implementación de la vista que permite al Administrador realizar la Gestión de plataformas. En esta vista el Administrador realiza las siguientes funciones:

A. Agregar una nueva plataforma.
B. Buscar una plataforma por nombre.
C. Listar todas las plataformas registradas en la Base de datos.
D. Ver la información de una plataforma.
E. Configurar una plataforma, permite adicionar los servicios que va a prestar laplataforma.
F. Editar la información de una plataforma.
G. Activar o desactivar una plataforma cuando se quiera dar de baja.

Figura 4.2 Gestión de plataformas

Para agregar una nueva plataforma se debe presionar el botón “agregar nueva plataforma” correspondiente al ítem A, el cual desplegará la vista para llenar los datos correspondientes a esta, como se muestra en la Figura 4.3. El usuario debe tener en cuenta que todos los campos son obligatorios, una vez esté completa la información puede presionar el botón “Agregar” o “Cancelar” en caso de no crear el usuario.

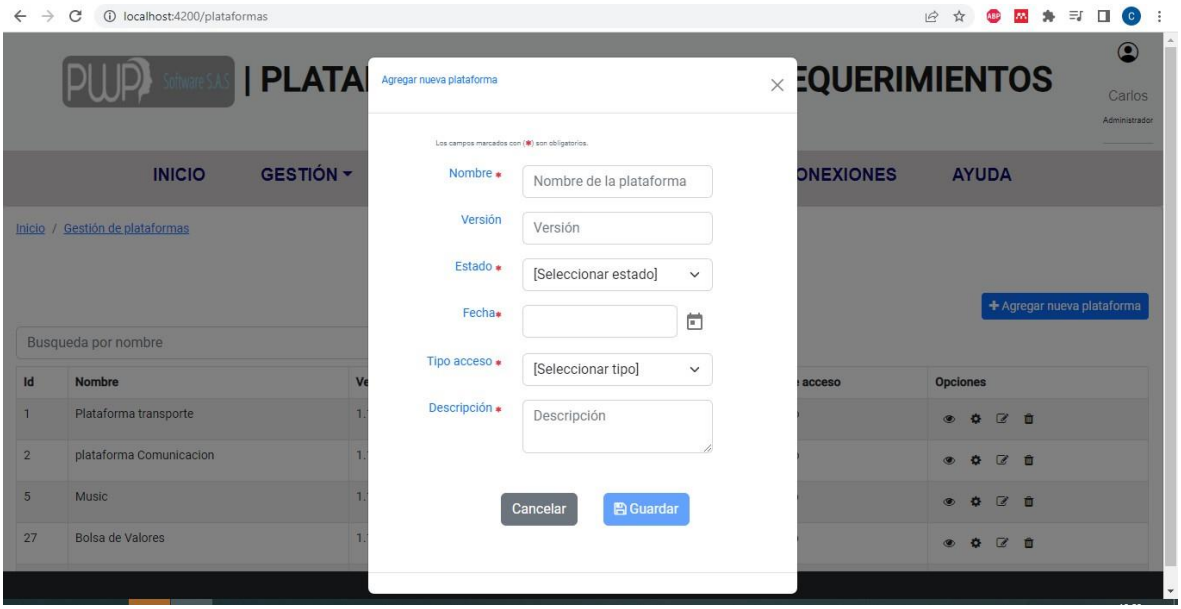

Figura 4.3 Agregar nueva plataforma.

Para configurar una plataforma, el sistema desplegará una vista, como se muestra en laFigura 4.4**.** Donde muestra una vista con sub-menu con las siguientes opciones:

A. Menú principal, que permite regresar a la vista principal de gestión de plataformas
B. Configuración Inicial, muestra la información básica de la plataforma que se va a configurar.
C. Configuración de servicios, es la opción donde se maneja la gestión de servicios que cumple con las siguientes funcionalidades, ver figura 4.5.

    1. Agregar un nuevo servicio a la plataforma
    2. Listar los servicios pertenecientes a la plataforma.
    3. Ver la información de un servicio
    4. Editar la información de un servicio.
    5. Configurar un servicio, permite adicionar el productor o consumidor con el cual interactúa la aplicación.

Figura 4.4. Menú configurar plataforma

Figura 4.5 Menú de servicios

Para la opción 5 configurar un servicio, el sistema desplegará una vista, como se muestra en la Figura 4.6**,** para que se pueda:

A. adicionar un nuevo productor o consumidor al servicio que se esté configurando.
B. Quitar un productor o consumidor al servicio que se esté configurando.
C. Listar todos los productores y consumidores que contiene el servicio.

Figura 4.6. Menú de servicios

Para agregar un nuevo productor/consumidor a un servicio se debe presionar el botón “Agregar nuevo productor o consumidor”, y el sistema desplegará una vista, como se muestra en la Figura 4,7. Con los productores y consumidores que puede adicionar al servicio.

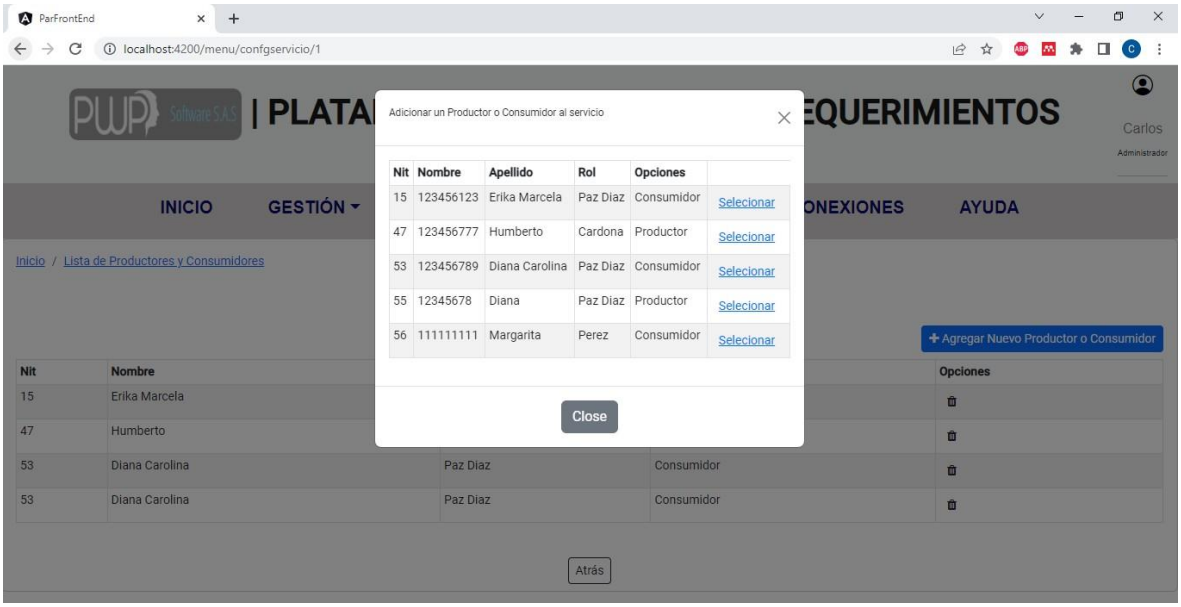

Figura 4.7 Lista de productor y consumidores para seleccionar

Gestión de usuarios

En la Figura 4.8 se muestra la implementación de la vista que permite al Administrador realizar la Gestión de usuarios. En esta vista el Administrador realiza las siguientes funciones:

A. Agregar un nuevo usuario de rol Administrador o Desarrollador.
B. Buscar un usuario por nombre, apellido o usuario.
C. Listar los usuarios registrados
D. Ver la información de un usuario
E. Editar la información de un usuario Registrado.
F. Activar o desactivar un usuario cuando se quiera dar de baja.

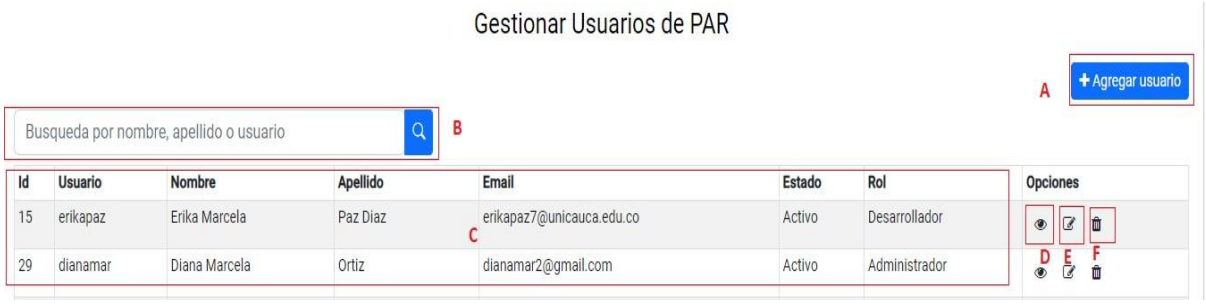

Figura 4.8 Gestión de usuarios

Para agregar un usuario se debe presionar el botón “agregar usuario” correspondiente al ítem A

el cual desplegará la vista para llenar los datos correspondientes a esta, como se muestra en la Figura 4,9. El usuario debe tener en cuenta que todos los campos son obligatorios, una vez esté completa la información puede presionar el botón “Agregar” o “Cancelar” en caso de no crear el usuario.

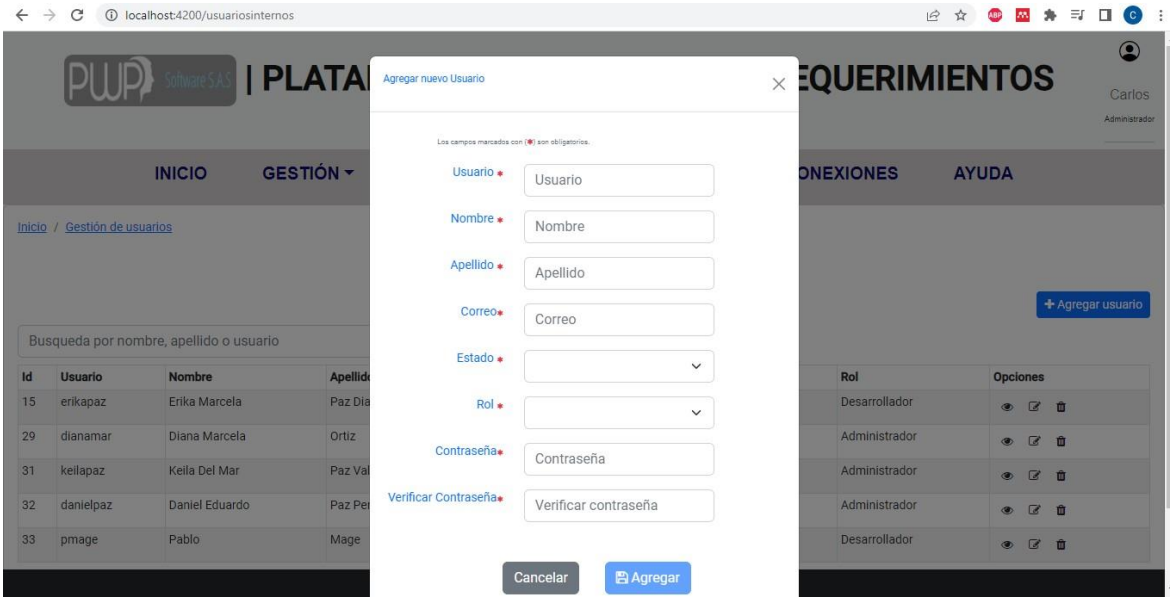

Figura 4.9 Agregar nuevo usuario.

Gestión de productores y consumidores

En la Figura 4.10 se muestra la implementación de la vista que permite al Administrador realizar la Gestión de productores y consumidores. En esta vista el Administrador realiza las siguientes funciones:

A. Agregar un nuevo productor o consumidor
B. Buscar un productor o consumidor por nombre, apellido o correo.
C. Listar los productores o consumidores registrados
D. Ver la información.
E. Editar la información de productores o consumidores registrados.
F. Activar o desactivar un usuario cuando se quiera dar de baja.

Figura 4.10 Gestión de productores y consumidores.

Para agregar nuevos productores o consumidores se debe presionar el botón “agregar productor/consumidor” el cual desplegará la vista para llenar los datos correspondientes a esta, como se muestra en la Figura 4.11. El usuario debe tener en cuenta que todos los campos son obligatorios, una vez esté completa la información puede presionar el botón “Agregar” o “Cancelar” en caso de no crear el usuario.

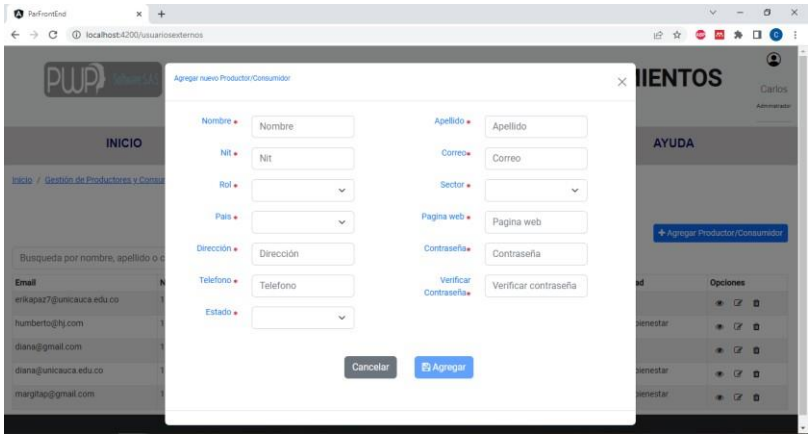

Figura 4.11 Agregar nuevo productores y consumidores.

Gestión de permisos

En la Figura 4.12 se muestra la implementación de la vista que permite al Administrador realizar la Gestión de productores y consumidores. En esta vista el Administrador realiza las siguientes funciones:

A. Agregar un nuevo permiso
B. Listar los permisos registrados
C. Editar la información de un permiso registrado.
D. Eliminar un permiso.

Figura 4.12 Gestionar de permisos

Configuraciones

En la Figura 4.13 se muestra los submenús de la opción configuraciones. En esta vista el Administrador puede realizar las siguientes funciones:

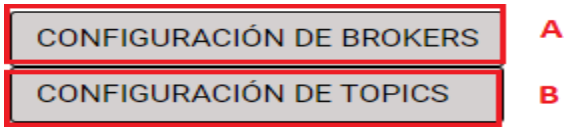

Figura 4.13 *submenús de la opción configuraciones.*

Gestión de brokers

En la Figura 4.14 se muestra la implementación de la vista que permite al Administrador realizar la Gestión de brokers. En esta vista el Administrador realiza las siguientes funciones:

A. Agregar un nuevo broker
B. Listar los brokers registrados

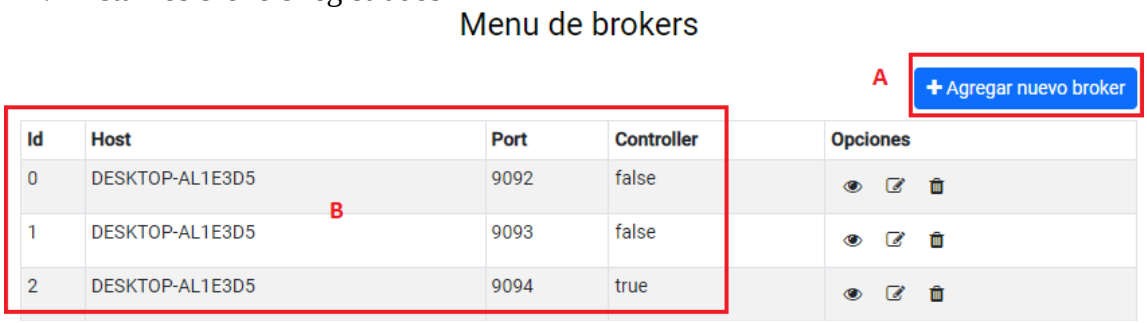

Figura 4.14 Menú de gestionar broker.

Gestión de topics

En la Figura 4.15 se muestra la implementación de la vista que permite al Administrador realizar la Gestión de topics. En esta vista el Administrador realiza las siguientes funciones:

A. Agregar un nuevo topic
B. Listar los topics registrados

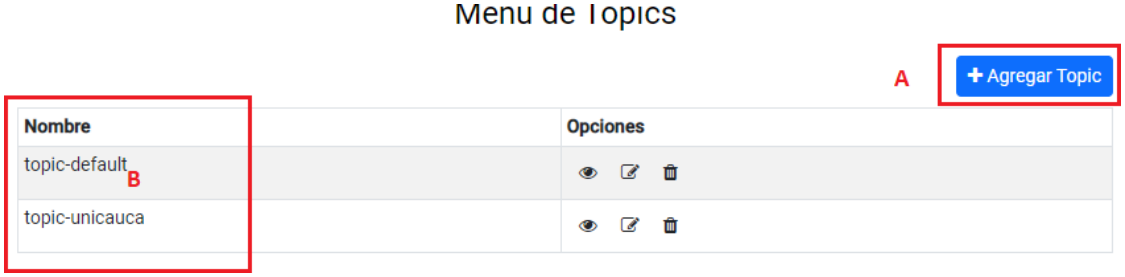

Figura 4.15 Menú de gestionar topics

Aplicación productora para probar que se envía la información a la plataforma

En la Figura 4.16 se muestra la implementación de la vista que permite a un Productor realizar él envió de mensajes hacia la aplicación de atención de requerimientos. En esta vista se debe ingresar el ID del servicio al cual se asociará la información, el nit que identifica al productor y por último el mensaje a enviar. La funcionalidad puede llevar a cabo el productor es la siguiente.

A. Publicar

Figura 4.16 Publicar un mensaje.

Aplicación consumidora para probar que se recibe la información a la plataforma

En la Figura 4.17 se muestra la implementación de la vista que permite a un Consumidor capturar el mensaje de la aplicación de atención de requerimientos. En esta vista se debe ingresar el ID del servicio de donde corresponde la información, y el nit que identifica al consumidor. Una vez llenos los campos se habilita los botones. Las funcionalidades del consumidor son las siguientes:

A. Consumir, permite que un consumidor se conecte a la aplicación de atención de requerimientos, solicitando la información del ID del servicio que ingreso.
B. Refrescar, permite actualizar la lista los mensajes que se envían desde elproductor.

Figura 4.17 Consumir mensaje.

Pruebas

Pruebas funcionales:

Se ejecutaron una serie 44 casos de pruebas sobre las funcionalidades de la aplicación, estas pruebas se ejecutaron en una sola versión y falta realizar ajustes a las fallas encontradas en estas. Se aprobaron 30 casos y se corrigieron 2 de los 14 que no se aprobaron. Las pruebas funcionales ejecutadas se encuentran en el Anexo C. En la siguiente tabla se muestra el formato utilizado para realizar los casos de prueba.

Tabla 11 Formato casos de prueba.

<Nombre caso prueba>	**<Código del CP>** **¿Prueba de despliegue?**	**Si/No**
Descripción: **<Descripción del caso de prueba>**		
Prerrequisitos **<Enumerar los prerrequisitos para la prueba>**		
Pasos: **<Pasos generales para la prueba, basados en los escenarios de los casos de uso, si existen.>**		
Resultado esperado: **<Resultado esperado de la prueba>**		
Resultado obtenido: **<Resultado obtenido de la ejecución del caso de prueba>**		

En la Tabla 12, se muestra una descripción de cantidad de pruebas aprobadas, no aprobadas y corregidas de los casos de prueba realizados por cada historia épica y unos casos relacionados con la producción y consumo de mensajes.

Tabla 12 Informe de casos de prueba

Historias épicas	# de pruebas	Aprobados	No Aprobados	Corregidos
Gestión de usuarios	12	10	2	2
Gestión de plataformas	10	8	2	0
Gestión de brokers	4	2	2	0
Gestión de topics	4	2	2	0
Gestión de productores / Consumidores	10	6	4	0
Producir y consumir un mensaje.	4	2	2	0

Pruebas del rol Administrador

Gestión de usuarios

Casos de prueba aprobados

- Agregar un usuario ingresando entradas correctas para todos los campos obligatorios.
- Agregar un usuario con entradas incorrectas y sin llenar algunos camposobligatorios.
- Cambiar un dato de un campo obligatorio con resultado exitoso.
- Cancelar cambios en el momento de editar o agregar un usuario.
- Cambiar de estado un usuario
- Cancelar cambios en el momento de Activar o Inactivar un usuario.
- Listar todos los usuarios registrados.
- Ver la información de un usuario

Casos de prueba No aprobados

- Agregar usuario con un correo ya existente en la base de datos.
- Buscar un usuario registrado.

Casos de prueba corregidos

- Ingresar un correo existen en la base de datos
- Realizar la búsqueda de un usuario por su nombre, apellido o usuario.

Gestión de Plataformas

Casos de prueba aprobados

- Agregar una plataforma ingresando entradas correctas para todos los campos obligatorios.
- Agregar una plataforma con entradas incorrectas y sin llenar algunos campos obligatorios.

- Cambiar un dato de un campo obligatorio con resultado exitoso.
- Cancelar cambios en el momento de editar o agregar una plataforma.
- Cambiar de estado una plataforma
- Cancelar cambios en el momento de Activar o Inactivar una plataforma.
- Listar todas las plataformas registradas.
- Ver la información de una plataforma

Casos de prueba No aprobados

- Agregar una plataforma con un nombre ya existente en la base de datos
- Buscar una plataforma registrada.

Casos de prueba corregidos

No se efectuó ninguna corrección de los casos no aprobados.

Gestión de brokers

Casos de prueba aprobados

- Agregar un nuevo broker ingresando entradas correctas para todos los campos obligatorios.
- Agregar un nuevo broker con entradas incorrectas y sin llenar algunos campos obligatorios.
- Listar todos brokers registrados.

Casos de prueba No aprobados

- Agregar un broker con un puerto ya existente

Casos de prueba corregidos

No se efectuó ninguna corrección de los casos no aprobados.

Gestión de topics

Casos de prueba aprobados

- Agregar un nuevo topic ingresando entradas correctas para todos los campos obligatorios.
- Listar todos topics registrados.

Casos de prueba No aprobados

- Agregar un nuevo broker con entradas incorrectas y sin llenar algunos campos obligatorios.
- Agregar un topic con un nombre ya existente

Casos de prueba corregidos

No se efectuó ninguna corrección de los casos no aprobados.

Gestión de productores y consumidores

Casos de prueba aprobados

- Agregar un productor/consumidor ingresando entradas correctas para todos los campos obligatorios.
- Agregar un productor con entradas incorrectas y sin llenar algunos campos obligatorios.
- Cambiar un dato de un campo obligatorio con resultado exitoso.
- Cancelar cambios en el momento de editar o agregar un productor/consumidor.
- Buscar un productor/consumidor registrado.
- Ver la información de un productor/consumidor

Casos de prueba No aprobados

- Agregar productor/consumidor con un correo ya existente en la base de datos.
- Cambiar de estado un productor/consumidor.
- Cancelar cambios en el momento de Activar o Inactivar un usuario.
- Listar todos los productores/consumidores registrados.

Casos de prueba corregidos

No se efectuó ninguna corrección de los casos no aprobados.

Publicar o consumir un mensaje

Casos de prueba aprobados

- Acceder exitosamente a un servicio PAR- publicar
- Acceder exitosamente a un servicio PAR- consumidor

Casos de prueba No aprobados

- Ingresar datos incorrectos al acceder a un servicio PAR- publicar
- Ingresar datos incorrectos al acceder a un servicio PAR-consumir

Casos de prueba corregidos

No se efectuó ninguna corrección de los casos no aprobados.

Capítulo 5 – Conclusiones

5.1 Conclusiones

Durante la realización de esta práctica profesional, quedan de conclusión las siguientes:

- La práctica profesional permitió, mediante el desarrollo de una aplicación para el módulo de enrutamiento de mensajes, la comunicación con otras aplicaciones, que interactúan enviando o consumiendo mensajes, mediante el uso de la tecnología apache Kafka.
- Para probar el funcionamiento de la aplicación para el módulo de enrutamiento de mensajes fueron desarrolladas dos aplicaciones más, la primera productora, que envía mensajes asociados al mercado bursátil a la plataforma, y la segunda consumidora, a la cual se le reenvían los mensajes.
- Las técnicas utilizadas para captura de requisitos funcionales y no funcionales facilitaron el proceso de identificar las necesidades asociadas al módulo de enrutamiento de mensajes, y facilitaron el levantamiento de las Historias épicas, Historias de usuario y criterios de aceptación.
- Al evaluar el cumplimiento de algunos requisitos de la aplicación para el módulo de enrutamiento de mensajes, se evidencia que permite que varios productores envíen información a la plataforma, la cual la redirige de manera genérica a varios consumidores.
- La tecnología apache Kafka proporcionar un modelo de mensajería de publicación/suscripción que permite la interoperabilidad entre sistemas que usan diferentes lenguajes de programación, sistemas operativos, y protocolos de comunicación.
- El uso de elementos de la metodología scrum permitió obtener mejores resultados en la entrega de las funcionalidades del módulo de enrutamiento de mensajes, al permitir entregas parciales y regulares del producto final.

5.2 Lecciones aprendidas

Durante la realización de esta práctica profesional, algunas de las lecciones aprendidas son las siguientes:

- Cuando se requiera desarrollar un módulo que necesite de la integración con otros, se debe tener como prioridad comunicación entre los desarrolladores, para que al final el producto final no se ve perjudicado.
- De un buen levantamiento de requisitos funcionales y no funcionales, asegura que el producto final sea el esperado por los clientes.
- Desarrollar componentes que permitan que cualquier productor y consumidor se comunique de manera genérica con Kafka, requirió una gran inversión de tiempo en la implementación.
- Durante el desarrollo de prácticas profesionales, se requiere en la planificación establecer un espacio para la capacitación del estudiante en las tecnologías a utilizar en el proyecto.

5.3 Conocimientos aplicados, conocimientos y habilidades adquiridas

El desarrollo de práctica profesional permitió profundizar en temas vistos durante la formación académica y adquirir nuevos conocimientos y habilidades que facilitaron la realización del proyecto. Entre los conocimientos obtenidos en la carrera que fueron aplicados tenemos: conocimientos en técnicas que facilitan la captura de requisitos funcionales y no funcionales, elaboración de diagramas de secuencia, diseño de la arquitectura de un sistema software; desarrollo de servicios REST, prototipado de interfaces gráficas, especificación de requisitos mediante historias de usuario y criterios de aceptación y manejo de la metodología Scrum.

Entre los conocimientos nuevos adquiridos tenemos los siguientes: diseño de la arquitectura de un sistema software mediante el uso del Modelo C4 y desarrollo de sistemas software usando tecnologías como apache Kafka, Sprint Boot y Angular.

Durante el periodo de la práctica, también fueron adquiridas un conjunto de habilidades blandas, entienda como habilidad blanda a las habilidades interpersonales, que se atribuyen completamente a la personalidad, que le ayudan a desenvolverse eficazmente en su trabajo, y a tener buena relación con sus compañeros. Las principales habilidades adquiridas fueron: el trabajo en equipo, ya que al compartir ideas con otras personas permitió mejorar la comprensión y dar solución a las dificultades; capacidad para aprender nuevos conceptos, tecnologías y técnicas relacionadas a la carrera; buena actitud para manejar las situaciones de la mejor manera; compromiso y persistencia dado que se afrontaron adversidades que se presentaron en el desarrollo de práctica; y por último, flexibilidad y tolerancia al cambio al afrontar las crisis presentadas al momento de negociar los requisitos y crear componentes software genéricos.

5.4 Dificultades durante la practica

Dado que es importante resaltar los logros obtenidos, también es necesario compartir los motivos que causaron dificultades en el proceso, con el fin de conocerlas y aplicar medidas correctivas.

- Fue necesario emplear más tiempo del esperado durante la captura de requisitos debido a que la empresa no tenía claros los requisitos del sistema y las implicaciones de nuevos requisitos en el código fuente.

- El retiro de un stakeholder llevo a que se perdieran los lazos de comunicación y el seguimiento en las actividades que se llevaban a cabo.

- El equipo de trabajo en cargado del módulo de seguridad y módulo de despliegue continuo abandonaron el proyecto y, por lo tanto, no fue posible llevar a cabo la integración de esos módulos con el módulo de enrutamiento de mensajes.

- Crear un código genérico que permita que un productor envié la información a la plataforma mediante servicios REST y que el consumidor la consuma mediante servicios REST.

5.5 Aspecto a mejorar en las prácticas profesionales

Teniendo en cuenta que la práctica profesional se realizó de manera virtual y no remunerada sería bueno que las empresas faciliten el acceso a internet de alta velocidad, brinde capacitación en temáticas o en herramientas tecnológicas que se requieran en el desarrollo de la práctica, y provea el licenciamiento para software requerido. Otro aspecto a mejorar, que durante la creación de anteproyecto delimitar el alcance de la propuesta a 6 meses.

5.6 Trabajos futuros

Quedan como trabajo futuro a partir de la práctica profesional realizada:

- Se deben realizar pruebas por parte de los usuarios de la aplicación más exhaustivas para identificar oportunidades de mejora para la aplicación.
- Se requiere que este proyecto tenga una siguiente fase en la cual se puedan construir los módulos de despliegue continuo y módulo de seguridad para que se pueda llevar a cabo la integración.

Referencias Bibliográficas

[1] J. Giraldo Jaramillo, “El mercado de la bolsa de valores: motivaciones de inversión en un segmento de mercado,” Universidad Autónoma de Occidente, 2020.

[2] E. Bash, “ABC del Inversionista,” *PhD Propos.*, vol. 1, p. 24, 2015.

[3] S. Salvador Rodríguez, “Obtención y representación de datos en tiempo real procedentes de un sistema IoT sobre un visor geográfico,” Universidad de Vigo, 2020.

[4] “Apache Kafka.” [Online]. Available: https://kafka.apache.org/. [Accessed: 15-Feb-2021].

[5] H. F. González, J. C. Salavarrieta, and F. Soto, “Desarrollo de una solución informática para el análisis de datos del sistema de pagos de seguridad social para el sector financiero colombiano,” *Ingenio Magno*, vol. 11, no. 1, p. 16, 2020.

[6] D. M. Martín, “Evaluación de spring MVC,” Universidad de Alcalá, 2014.

[7] “Angular.” [Online]. Available: https://angular.io/. [Accessed: 24-Feb-2021].

[8] “Servicios de informática en la nube | Microsoft Azure.” [Online]. Available: https://azure.microsoft.com/es-es/. [Accessed: 26-Feb-2021].

[9] IBM Knowledge Center, “Aplicaciones empresariales,” 2020. [Online]. Available: https://www.ibm.com/support/knowledgecenter/es/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/UserGuide/c_cmdb_business_apps.html. [Accessed: 15-Feb-2021].

[10] H. F. Sarasty, “Documentación y análisis de los principalaes frameworks de arquitectura de software en aplicaciones empresariales,” Universidad Nacional de LaPlata, 2015.

[11] “ISO 25000.” [Online]. Available: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&limitstart=0. [Accessed: 15-Feb-2021].

[12] Apache Software Foundation, “Apache Kafka - Introducción.” [Online]. Available: https://kafka.apache.org/documentation/#introduction. [Accessed: 15-Feb-2021].

[13] B. Stopford, *Designing Event-Diven Systems*, vol. 1, no. 1. 1005 Gravenstein Highway North, Sebastopol, 2018.

[14] WFE - The World Federation of Exchanges, “Welcome to the future of markets | The World Federation of Exchanges.” [Online]. Available: https://www.world-exchanges.org/. [Accessed: 15-Mar-2021].

[15] D. A. Beltrán Morales, “Estrategias de inversión para fomentar las negociaciones bursátiles de empresas medianas en la bolsa de valores de Colombia,” Universidad Piloto de Colombia, 2020.

[16] System Innovation, “Platform Technologies,” 2017.

[17] S. Brown, “El modelo C4 para visualizar la arquitectura de software.” [Online]. Available: https://c4model.com/. [Accessed: 27-Feb-2021].

[18] Mozilla developer, “SPA (aplicación de una sola página).” https://developer.mozilla.org/en-US/docs/Glossary/SPA [Accessed: 20-Ago-2022].

[19] OpenJS Foundation, “Node.js.” https://nodejs.org/es/about/ [Accessed: 20-Ago-2022].

[20] Oracle, “MySQL.” https://www.mysql.com/ [Accessed: 20- Jun-2022].