DESARROLLO DEL SAP COMO PASANTÍA PARA *AXON GROUP*.

Diego Fernando Perdomo Samboní
Ever Andres Molano Hurtado

Universidad del Cauca
Facultad de Ciencias naturales, exactas y de la educación
Popayán, Mayo de 2014

DESARROLLO DEL SAP COMO PASANTÍA PARA *AXON GROUP*.

Trabajo de Pasantía como requisito para optar al título de Ingeniero Físico.

Diego Fernando Perdomo Samboní
Ever Andres Molano Hurtado

Director: Mg. Luis Fernando Echeverri.

Universidad del Cauca
Facultad de Ciencias naturales, exactas y de la educación
Popayán, Mayo de 2014

Nota de Aceptación

Director Mg. Luis Fernando Echeverri

Jurado Ing. Eduardo Andrés Cañola Sotelo

Jurado Dr. Wilfrand Pérez Urbano

Fecha de sustentación: 2 de julio de 2014

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ACRÓNIMOS

SAP: Sistemas, Aplicaciones y Productos para los Negocios (*System, Aplicattion and Product – Bussines Suite*).

ERP: Planeación de Recursos Empresariales (*Enterprise Resource Planning*).

CRM: Gestión de Relación con el Cliente (*Customer Relationship Management*).

PLM: Gestión del Ciclo de Vida del Producto (*Product Lifecycle Management*)
.

SCM: Gestión de Cadena de Suministro (*Supplier Chain Management*).

SRM: Gestión de Relación con el Proveedor (*Supplier Relationship Management*).

SCADA: Sistema de Supervisión, Control y Adquisición de Datos (*Supervisory Control and Data Acquisition*).

IDE's: Entornos de Desarrollo Integrados (*Integrated Development Environment).*

LAN: Redes de Area Local (*Local Area Networks*)

WAN: Redes de Área Extendida (*Wide* Area *Netwrok*).

P2P: Punto a Punto (*Peer To Peer*).

SOA: Arquitectura Orientada a Servicios del Cliente (*Service Oriented Architecture*)

GPL: Licencia Pública General (General Public License)

J2EE: Entorno Empresarial Java (*Java Enviroment Enterprise*)

DM: Modelo de Datos (*Data Model*)

DDL: Lenguaje de Definición de Datos (*Data Definition Language)*

DML: Lenguaje de Manipulación de Datos (*Data Manipulation Language*)

QL: Lenguaje de Consulta (*Query Language*)

DBMS: Sistemas de Gestión de Base de Datos (*Database Management System*)

RUP: Proceso Unificado Racional (*Rational Unified Process*)

XP: Programación Extrema (*Xtreme Programming*)

API: Interfaces de Programación de Aplicaciones (*Application Programming Interface)*

DAO: Objeto de Acceso de Datos (*Data Acces Objet*)

BO: Objeto de Negocios (*Business Object*)

MB: *Bean* Administrado (*Managed Bean*)

HTML: Lenguaje Marcado de Hipertexto en línea *web* (*HyperText Markup Language*)

HTTP: Protocolo de transferencia de hipertexto en línea *web* (*Hyper Text Transfer Protocol*)

JBPM: *Bites* Por Minuto en *Java* (*Java Bites By Minute*)

JPA: persistencia de una API en *Java* (*Java persistense* API)

EJBs: API's Empresariales en *Java* (*Enterprise java beans*)

JDBC: Controlador de acceso a Base de Datos *Java* (*Java Data Base Controller*)

JAR: Archivo Java (*Java ARchive*)

WAR: Archivo Web (*Web ARchive*)

CDI: Inyección de Dependencia y Contextos **(***Contexts and Dependency Injection*)

JSF: Servidor de interfaces de Java (*Java Server Faces*)

JAX-WS: Api de Java para XML y Servicios Web (*Java* API *for XML- Web Services*)

JAX-RS: Api de Java para Servicios Web inactivos (*Java* API *for RESTful Web Services*)

XML: Lenguaje de Marcas Extensible (*Extensible Markup Languague*)

INTRODUCCIÓN 1

La última década a sido escenario de un gran avance en las tecnologías de desarrollo *software*, esto ha fomentado un crecimiento en las aplicaciones y servicios que se utilizan en el diario vivir; desde la comunicación personal hasta el seguimiento y programación de tareas laborales y cotidianas. Un factor determinante en este avance ha sido el crecimiento exponencial del mercado debido a la demanda por parte de las compañías, de productos que automaticen este tipo de actividades básicas para los empleados; a fin de mejorar la eficiencia, eficacia y el rendimiento de los trabajadores en sus empresas, permitiéndole al empleador identificar las falencias y aciertos en la construcción de sus líneas de negocios.
La empresa de hoy en comparación a sus predecesoras, ha tenido un gran desarrollo en el área de tecnología demandando cada día más calidad y recursos tecnológicos con el propósito de mejorar las prestaciones que brinda. Es así como las tareas en las diferentes áreas de una compañía se han especializado, hasta el punto de convertirse en una tarea caótica para los directores de compañía identificar el fin de cada uno de los procesos de una empresa, afectando la unicidad en general.

Es por ello que la tecnología de vanguardia de la mano con la proliferación y el refinamiento del desarrollo *web*, supone nuevas oportunidades en la evolución y crecimiento de las empresas, permitiendo que las nuevas tecnologías jueguen un papel preponderante a nivel de automatización y sofisticación en sus distintas necesidades, haciendo de estas un eje transversal a todas las áreas que conforman una empresa, brindando un sustento y una alternativa a muchos procesos, en base al desarrollo de *software*.

Con esta premisa, se propone una solución que introduzca las características más relevantes de algunas tecnologías de desarrollo *software*, que permitan la gestión de las tareas, asignaciones, procesos y proyectos de los recursos humanos de las áreas que posee una empresa de cualquier índole para su correcto funcionamiento.

El objetivo de esta pasantía se centra en determinar las consideraciones necesarias que permitan realizar una arquitectura para una *suite* empresarial, Sistemas, Aplicaciones y Productos para los Negocios (SAP, *System, Aplicattion and Product – Bussines Suite*), que mediante la creación de algunos sub-módulos permita verificar la funcionabilidad en el desempeño de las diferentes tareas y labores que tiene y desarrolla la empresa *Axon Group*.

Este documento se divide en cuatro capítulos estructurados de la siguiente manera:

Capítulo 1. Presenta de manera general los diferentes servicios que tiene la empresa en el mercado de la automatización industrial.

Capítulo 2. Define los conceptos teóricos, así como las características y aspectos más importantes utilizados en el desarrollo de *software*.

Capítulo 3. Describe de manera general la realización de la arquitectura la cual se valida con la realización y funcionamiento de los módulos como trabajo de pasantía en la empresa *Axón Group*.

Capítulo 4. Se exponen las conclusiones del trabajo de pasantía y se presentan una serie de recomendaciones para su continuación en el desarrollo de Sistemas, Aplicaciones y Productos para los Negocios (SAP, *System, Aplicattion and Product – Bussines Suite*) e investigaciones en este campo tratado; finalmente se genera una serie de ideas que pueden contribuir a realizar una profundización en esta área.

1. AXON GROUP

Axon Group es una empresa de automatización industrial especializada en el diseño de productos *software*, que buscan mejorar y facilitar aplicaciones en áreas del sector eléctrico e industrial [1]. Principalmente dentro del abanico de aplicaciones se ofrece el sistema *scada*, la pasarela de comunicaciones, Un sistema de administración de bodegas de datos, el simulador de protocolos, diseño, ejecución y evaluación de proyectos de control y protección entre otros; además ofrece capacitación mediante cursos de formación en protocolos y *software* de tele-control [2].

La empresa con sede en Colombia ha tenido un alcance que supera los límites nacionales, estableciéndose en países latinoamericanos como Chile, Bolivia, Argentina entre otros; y dentro de las áreas que conforman la empresa, se encuentran ingenieros de la Universidad del Cauca en la rama de electrónica.

Este trabajo de pasantía, determina la arquitectura para el abanico de Sistemas, Aplicaciones y productos para los Negocios (SAP, *System, Aplicattion and Product – Bussines Suite*) y su validación con un módulo de Planeación de Recursos Empresariales (ERP, *Enterprise Resource Planning*) desarrollando un submódulo de gestión del capital humano, a fin de realizar un sistema piloto tipo SAP en una versión Beta[1] que permita la gestión de contenidos personalizados con extensibilidad para algunas labores de gestión empresarial.

La arquitectura SAP ERP, permite mejorar la calidad de la gestión de contenidos relacionados con las labores de producción, logística, distribución, inventario, envíos, facturas y contabilidad de las empresas. Este módulo a su vez se compone en submódulos.

A continuación se presentan los productos y servicios de *Axon Group*.

1.1. SAP

SAP es un conjunto de programas que permiten a las empresas ejecutar y optimizar distintos aspectos como el área de ventas, finanzas, operaciones bancarias, compras, fabricación, inventarios y relaciones con los clientes. Ofreciendo la posibilidad de realizar procesos específicos de la empresa o crear módulos independientes para funcionar con otro *software* de SAP o de otros proveedores. Se puede utilizar en cualquier sector empresarial [3].

SAP Business Suite está dividido en 5 módulos:

[1] Versión Beta: Representa la primera versión completa de un programa informático o de un producto, que posiblemente sea inestable pero útil para realizar una inspección previa técnica.

- SAP CRM (*Customer Relationship Management*).
- SAP ERP (*Enterprise Resource Planning*).
- SAP PLM (*Product Lifecycle Management*).
- SAP SCM (*Supply Chain Management*).
- SAP SRM (*Supplier Relationship Management*).

Proporcionando soporte para las siguientes áreas empresariales:

- Finanzas
- Fabricación
- Aprovisionamiento
- Desarrollo de productos
- *Marketing*
- Ventas
- Servicios
- Recursos Humanos
- Gestión de la cadena de suministro
- Gestión de tecnologías de la información

1.1.1. SAP ERP

La planificación de recursos empresariales ERP, da soporte a las funciones esenciales de los procesos y operaciones de la empresa, de análisis empresarial, contabilidad financiera e interna, gestión del capital humano, gestión de operaciones, gestión de servicios corporativos y autoservicios [4], [5]. A su vez, este producto se subdivide en:

1.1.1.1. SAP ERP FINANZAS

Este submódulo permite cumplir con los estándares de generación de informes financieros, flujo de caja y gestión de riesgos [5].

1.1.1.2. SAP ERP GESTIÓN DEL CAPITAL HUMANO

El submódulo de gestión del capital humano, permite optimizar los procesos de selección y motivación de los empleados, haciendo énfasis en los proyectos en los cuales laboran, las horas invertidas, el tiempo a favor, los beneficios recibidos entre otras opciones [5].

1.1.1.3. SAP ERP OPERACIONES

En este submódulo se mejoran las operaciones para reducir costos, aumentar ingresos, maximizar la rentabilidad y la atención al cliente [5].

1.2. PRODUCTOS Y SERVICIOS

La suite de *Axon Group* se divide en productos de tipo *software* (Aplicaciones) y en servicios de mantenimiento, soporte técnico, soluciones y atención a problemas y eventos en tiempo real [1 – 5].

1.2.1. *AXON BUILDER IHM/SCADA*

Axon Builder es el sistema de Supervisión, Control y Adquisición de Datos (SCADA, *Supervisory Control and Data Acquisition*) de la empresa, es de alta confiabilidad, eficiente, robusto y fácil de usar para el desarrollo de aplicaciones que el cliente necesite, también tiene soporte instantáneo sin retraso alguno [1 – 5].Sus características más relevantes son:

- Variedad en protocolos de comunicación.
- Arquitectura Cliente/Servidor con múltiples clientes de control y visualización.
- Múltiple redundancia.
- Conexión a diferentes sistemas de base de datos.
- Reportes personalizados.
- Alarmas y Eventos.
- Tendencias en tiempo real e histórico.
- Un poderoso módulo de *scripts*.
- Creación de diferentes niveles de seguridad.
- Herramienta completa de dibujo.
- Librerías para el sector eléctrico e industrial.
- Filtros y herramientas para imprimir y exportar datos a pdf o csv.

Figura 1.1. Logo Axon Builder Scada. Tomado de [1].

1.2.2. *AXON BUILDER SERVER*

Es una herramienta que se utiliza para configurar aplicaciones requeridas por el cliente, tiene como característica principal parametrizar señales de entrada, comandos y medidas mediante un método de topología para su mapeo, todo esto permite múltiple redundancia en aplicaciones que tengan un alto grado de robustez, debido a su arquitectura. *Axon Builder Server* permite configurar eventos, avisos y alarma entre otros [1 – 5].

1.2.3. *AXON* COMUNICACIONES

Herramienta que unifica los equipos al sistema SCADA. Posee varios protocolos propios (protocolos nativos sin instaladores adicionales) que facilitan esta tarea sin tener que hacer uso de más equipos [1 – 5]. Entre estos protocolos se encuentra:

- IEC 61850.
- MODBUS (TCP Y SERIAL).
- IEC 60870-5-101.
- IEC 60870-5-104.
- DNP3.
- SEL.
- OPC.

1.2.4. *AXON BUILDER* CLIENTE

Esta herramienta proporciona la configuración que desee el cliente, permitiéndole hacer, insertar y configurar dibujos y sonidos, con el fin de tener un diseño propio que cumpla con sus especificaciones. Esta herramienta posee librerías específicas del sector eléctrico, con las que se pueden generar animaciones de colores, esconder objetos, rotar, mover y manipular. Por tanto facilita la generación de un diseño en forma completa e integral para cada necesidad del usuario [1 – 5].
Además, este producto hace uso de otras herramientas del sistema SCADA para visualizar datos, alarmas, entre otras cosas.

1.2.4.1. LICENCIA DE USO

Se ofrece una licencia con acceso *hardware* dependiendo del tamaño del proyecto para agilizar el manejo en el desarrollo *software*, la ampliación o movimiento de sus licencias de uso, como se muestra a continuación:

Características	Configuración
Numero de Tags	Desde 1024 hasta 100.000 Tags
Número de clientes	Desde 1 hasta 20 clientes Aplica también para cliente web
Redundancia	Con o sin redundancia
Protocolos de comunicación	Puede seleccionar entre: IEC 61850, IEC 61870-5-101/104/103, DNP3, SEL, OPC Por defecto, protocolo Modbus

Tabla 1.1. Configuraciones de *AXON BUILDER CLIENTE*. Tomado de [1].

1.2.5. *AXON EXCHANGE GATEWAY* DE COMUNICACIONES

Es una pasarela de comunicaciones que permite la conversión de información de un protocolo a otro, unifica y guarda datos de distintos protocolos de comunicación que se tramiten a otros sistemas. Este producto posibilita la unificación, integración y automatización de procesos industriales y subestaciones eléctricas [1 – 5].

1.2.5.1. Características y Protocolos

El sistema de comunicaciones *Axon Exchange* fue creado para integrar, manejar y administrar la información de forma rápida, sencilla y eficiente, en diferentes Entornos de Desarrollo Integrados (IED's, *Integrated Development Environment*), y remitirla al sistema SCADA Local o remoto, a través de la utilización de los siguientes protocolos:

- IEC 61850 (Cliente).
- DNP3 LAN/WAN (Maestro/Esclavo).
- DNP3 serial (Maestro/Esclavo).
- IEC 60870-5-104 (Maestro/Esclavo).
- IEC 60870-5-101 (Maestro/Esclavo).
- IEC 60870-5-103 (Maestro).
- Modbus (Maestro/Esclavo).
- SEL (Maestro).

Figura 1.2. Logo *Axon Exchange Gateway De Comunicaciones*. Tomado de [1].

Utilizando herramientas de programación sencilla se pueden reunir los diferentes equipos en tiempos cortos. Este programa es creado para correr en *Windows* XP/7/*Embedded*.

1.2.5.2. HERRAMIENTAS DE CONFIGURACIÓN Y *RUNTIME*[2]

Axon Exchange es un sistema que tiene herramientas necesarias y útiles para mantenimiento de sus equipos garantizando un correcto y eficiente funcionamiento, también facilitan y posibilitan diagnosticar la conexión de todos los IED's integrados (maestro o esclavo), lo que da lugar a la marcha del servicio del sistema automatizado [1 – 5].

1.2.5.3. *CONFIGURATION*

Esta herramienta permite configurar los distintos IED's existentes en el sistema de comunicaciones (*gateway*). Haciendo simple y rápida la configuración, minimizando los costos de implementación para un sistema de automatización.

Tiene una herramienta denominada árbol de protocolos maestros/esclavos, a su lado derecho se ven desplegadas características que configuran todos y cada uno de los IED's, que lo conforman. Aquí se pueden configurar: Parámetros de conexión, señales de entrada/salida con direcciones y lógicas simples [1 – 5].

1.2.5.4. *RUNTIME*

Esta herramienta inicia el funcionamiento del sistema de pasarela de comunicaciones, mostrado que equipos se encuentran en línea.

La ventana posee tres iconos en la parte superior: *State*, es una función que permite evidenciar si se esta en línea (estado de conexión), *Viewe*r, visualiza como se encuentran las variables que se han configurado en la pasarela de comunicaciones y por ultimo se encuentra el ícono *Trace*, es la función que se usa para analizar el sistema en cada una de sus tramas de cada conexión.

- *LOG*

  Esta función proporciona el diagnóstico del sistema *Axon Exchange* y permite el correcto funcionamiento de las conexiones que se encuentran predeterminadas por el diseño en el sistema SCADA.

- *WEB SERVER*

  Es una función de la herramienta *Axón Echange* que permite observar los datos en forma instantánea, desde un equipo cualquiera el cual posea acceso a la misma red de los equipos configurados en la pasarela de comunicación, con su respectivo intervalo de tiempo, calidad, nombre y descripción de la señal, además desde esta función se puede solicitar los datos, facilitando la toma de pruebas con otros

[2] *Runtime*: es el intervalo de tiempo en el cual un programa se ejecuta.

niveles de control. También posee filtros que mejoran el manejo de las variables que sean necesarias diagnosticar.

1.2.6. *AXON TEST* SIMULADOR Y HERRAMIENTA DE ANÁLISIS DE PROTOCOLOS

Axon Test es otro producto de la empresa *Axon Group*, es una herramienta que simula protocolos, analiza y muestra de manera específica cada fragmento de la trama tomada por los equipos conectados al sistema, tiene filtros que se han implementado con la característica de visualizar únicamente las señales que sean necesarias, implementados por rango de direcciones o por tipo de dato (*Single, Double, Step Position, Bitstring, Normalized, Scaled, Short Float*). Además tiene una función de insertar colores en la ventana que analiza las tramas o en la tabla datos recibidos, de modo que al recibir nuevas señales es resaltada para poder visualizar cual fue la señal que se cambio de manera rápida [1 – 5].

Esta herramienta también posibilita la generación de un mapeo (Dirección – Descripción) o simplemente una copia desde *Excel*, de modo que cuando obtengan las señales se pueda ver con la especificación de cada punto, permitiendo investigar cada recepción emitida por todas las señales.

Protocolos utilizados:

- DNP3 LAN/WAN (Maestro/Esclavo).
- DNP3 serial (Maestro/Esclavo).
- IEC 60870-5-104 (Maestro/Esclavo).
- IEC 60870-5-101 (Maestro/Esclavo).
- IEC 60870-5-103 (Maestro).
- Modbus (Maestro/Esclavo).
- SEL (Maestro).

También es posible simular protocolos de forma gratuita, esta versión ya tiene una licencia, la cual cuenta además con:

- Uso de múltiples protocolos a la vez
- Conexión como monitor
- Sin publicidad

- Sin espera al iniciar

1.2.7. *AXON MANAGER* GESTIÓN Y ANÁLISIS DE DATOS HISTÓRICOS

Es un programa que gestiona el almacenamiento de los datos en forma instantánea, tiene compatibilidad con motores de bases de datos comerciales tales como: *MySQL Serve, Oracle*, *Interbase, PostgreSQL, MySql*, Entre Otros. *Axon Manager* posee interfaces gráficas de fácil uso y útiles que permiten una accesibilidad completa al sistema de información histórica (HIS) de la información guardada en su totalidad, de una manera muy manejable y configurable. Además permite obtener copias de seguridad de datos en un soporte externo magnético u óptico [1 – 5].

Figura 1.3. Logo *Axon Manager* Gestión Y Análisis De Datos Históricos. Tomado de [1].

La síntesis de los productos y servicios ofrecidos por *Axon Group* es un proceso que debe realizarse con la intención de identificar las distintas arquitecturas utilizadas en ellos, ya que se presentan las características más relevantes y necesarias que deben presentar las aplicaciones de la empresa.

El siguiente capítulo describirá los distintos niveles de la arquitectura y las técnicas de desarrollo que permitirán elaborar la arquitectura que mejor se adapte a las necesidades del SAP ERP.

2. CARACTERÍSTICAS, ELEMENTOS Y MODELOS QUE INCIDEN EN LA ARQUITECTURA SAP ERP PARA *AXON GROUP*

2.1 ARQUITECTURA DE *SOFTWARE*

Toda arquitectura debe describir diversos aspectos del *software* [6]. Generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas, por lo cual es de gran relevancia destacar que cada uno de ellos constituye una descripción parcial de una misma arquitectura y es deseable que exista cierto solapamiento entre ellos. Esto es necesario a fin de lograr que todas las vistas sean coherentes entre sí, dado que cada vista permite describir la arquitectura [7].

A lo largo del tiempo cada paradigma de desarrollo utiliza diferente número y tipo de vistas o modelos para describir una arquitectura, no obstante, existen al menos tres vistas absolutamente fundamentales en cualquier arquitectura:

- **visión estática:** Describe qué componentes tiene la arquitectura.
- **visión funcional:** Describe qué hace cada componente.
- **visión dinámica:** Describe cómo se comportan los componentes a lo largo del tiempo y cómo interactúan entre sí [6], [7] y [8].

Generalmente, no es necesario inventar una nueva arquitectura de *software* para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Así, las arquitecturas más universales son:

2.1.1. ARQUITECTURA MONOLÍTICA

Es un modelo en la que el *software* se estructura en grupos funcionales muy acoplados, involucrando los aspectos referidos a la presentación, procesamiento y almacenamiento de la información; implementándose dentro de un solo componente de *software* **¡Error! Argumento de modificador desconocido.**[8][9].

Históricamente, representan la estructura de los primeros *softwares,* constituidos fundamentalmente por un solo programa compuesto por un conjunto de funciones entrelazadas de tal forma que cada una puede llamar a cualquier otra. Las características fundamentales de este tipo de aplicaciones son:

- Construcción del programa final a base de componentes compilados al mismo tiempo.

- Carecen de protecciones y privilegios al entrar a rutinas que manejan diferentes aspectos de los recursos de la computadora, como memoria, disco, etc.

- Generalmente están hechos a medida, por lo que son eficientes y rápidos en su ejecución y gestión, pero como consecuencia carecen de flexibilidad para soportar diferentes ambientes de trabajo o tipos de aplicaciones.

- No hay distribución, tanto a nivel físico como a nivel lógico.

- Estructura interna indefinida. Los niveles de funcionalidad no están bien separados. Una aplicación monolítica puede componerse de uno o más niveles; elaborada bajo el supuesto de que será ejecutada en un único equipo de cómputo. Pero este tipo de aplicación también puede ser implementada en un computador central o *mainframe* y desde una terminal de servicio se puede monitorear y recibir los resultados que la aplicación monolítica devuelva.

La siguiente figura ilustra una arquitectura monolítica:

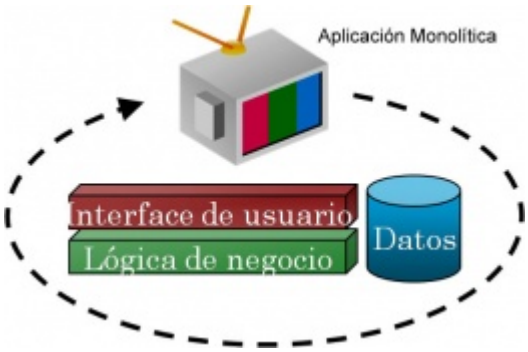

Figura 2.1 Arquitectura Monolítica. Adaptado de [9].

Ventajas:

- Eficiencia, ya que se producen pocos cambios en el contexto.

Desventajas:

- Difícil de depurar.
- Difícil de ampliar.
- Difícil de distribuir.
- Difícil de implantar.

Ejemplos: Tipos de *software* al cual aplica este Estilo Arquitectónico:

- *Firmware*[3]
- *Software* base de Sistema Operativo (Servicios, demonios, *Kernell*[4]).
- Pequeñas utilidades específicas (Desde virus, *keyloggers*, *sniffers,* entre otros.*)*

2.1.2. ARQUITECTURA CLIENTE-SERVIDOR

Es un modelo de aplicación distribuida en el cual las tareas se reparten entre los proveedores de recursos o servicios, conocidos como servidores, y los demandantes, llamados clientes. En este sentido un cliente realiza peticiones a otro programa y el servidor es quien le da respuesta [6 – 9].

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de suministradores incluyen los servidores *web*, los servidores de archivo, los servidores de correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes maquinas aumentando así el grado de distribución del sistema.

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico [10].

- Cliente:

Es el que inicia un requerimiento de servicio esperando recibir la respuesta del servidor; teniendo por tanto un papel activo en la comunicación (dispositivo maestro o amo). El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de

[3] Firmware: es un bloque de instrucciones de máquina para propósitos específicos, grabado en una memoria, normalmente de lectura/escritura.
[4] Kernell: Es el principal responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora.

redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente. Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

- Servidor**:**

Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente, desempeñando entonces un papel pasivo en la comunicación (dispositivo esclavo). Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente, conectándose al usuario final a través de Redes de Área Local (*LAN*, Local Area *Network*) o Redes de Área extendida (*WAN*, *Wide* Area *Netwrok*), para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc. Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado) y no es frecuente que interactúen directamente con los usuarios finales.

Ventajas:

- Centralización del control**:** los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P).

- Escalabilidad**:** se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).

- Fácil mantenimiento**:** al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, siendo este proceso transparente para el clientes sin verse afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.

- Gran Difusión**:** Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma cliente-servidor que aseguran la fiabilidad en las transacciones, la amigabilidad de la interfaz y la facilidad de empleo.

Desventajas:

- La congestión del tráfico ha sido siempre un problema en la arquitectura cliente servidor. Cuando una gran cantidad de clientes envían peticiones simultáneas al mismo servidor, puede ser que cause muchos problemas para éste (a mayor número de clientes, más problemas para el servidor). Por otro lado, en las redes P2P como cada nodo en la red hace las veces de servidor, el ancho de banda mejora en cuanto se conecten más nodos.

- En la arquitectura cliente servidor no se tiene la robustez de una red P2P, es por ello que cuando un servidor está abajo, las peticiones de los clientes no pueden ser satisfechas. En la mayor parte de redes P2P, los recursos están generalmente distribuidos en varios nodos de la red y aunque algunos salgan o abandonen la descarga; otros pueden todavía acabar de descargar consiguiendo datos del resto de los nodos en la red.

- El *software* y el *hardware* de un servidor son generalmente muy determinantes. Un *hardware* promedio de un equipo personal es posible que sirva a cierta cantidad de clientes. Normalmente se necesita *software* y *hardware* específico, sobre todo en el lado del servidor, para satisfacer el trabajo, aumentando el costo de implementación.

- El cliente no dispone de los recursos que puedan existir en el servidor. Por ejemplo, si la aplicación es *Web*, es imposible escribir en el disco duro del cliente o imprimir directamente sobre las impresoras sin sacar antes la ventana previa de impresión de los navegadores.

2.1.2.1. ELEMENTOS DE LA ARQUITECTURA CLIENTE/SERVIDOR

Con el objetivo de definir y delimitar el modelo de referencia de una arquitectura Cliente/Servidor, es importante identificar los componentes que permitan articular dicha arquitectura, considerando que toda aplicación de un sistema de información está caracterizada por tres componentes básicos [10], [11]:

- Presentación/Captación de Información.

- Procesos.

- Almacenamiento de la Información.

Los cuales suelen distribuirse tal como se ilustra en la figura:

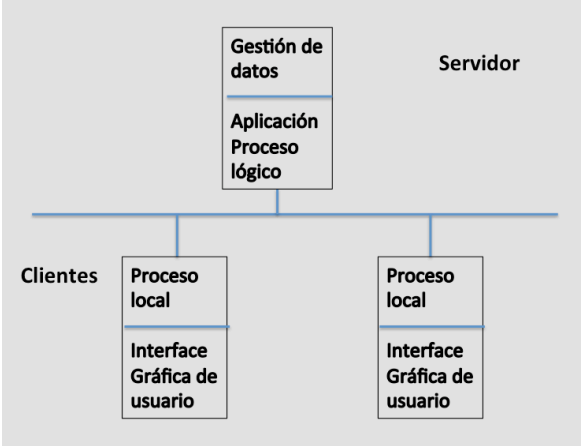

Figura 2.2 Distribución de una aplicación. Tomado de [11].

Y se integran en una arquitectura Cliente/Servidor en base a los elementos que caracterizan dicha arquitectura, es decir:

- Puestos de Trabajo.
- Comunicaciones.
- Servidores.

Tal como se presenta en la figura:

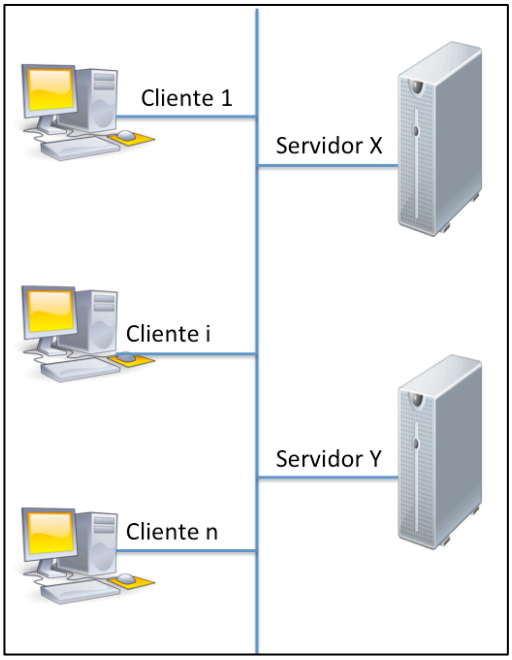

Figura 2.3 Arquitectura Cliente Servidor. Tomado de [11].

De estos elementos se destaca el puesto de trabajo o cliente final, quien representa estaciones de trabajo o computadoras personales conectadas a una red, que le permite acceder y gestionar una serie de recursos [6]. También es resaltable que el puesto de trabajo basado en un equipo personal conectado a una red, favorece la flexibilidad y el dinamismo en las organizaciones. Entre otras razones, porque permite modificar la ubicación de los puestos de trabajo, dadas las ventajas de la red [12].

Los Servidores suministran una serie de servicios como Bases de Datos, Archivos, Comunicaciones.), según la especialización y los requerimientos de los servicios que debe suministrar; entre estos servicios es posible encontrar:

- *Mainframes*[5].
- Miniordenadores.
- Especializados (Dispositivos de Red, Imagen, etc.).

Es relevante considerar una característica de los diferentes servicios, los cuales según el caso, pueden ser suministrados por un único servidor o por varios servidores especializados; ejemplo de ello son los servidores de las comunicaciones en sus dos vertientes:

- Infraestructura de redes.

Componentes *Hardware* y *Software* que garantizan la conexión física y la transferencia de datos entre los distintos equipos de la red.

- Infraestructura de comunicaciones.

Componentes *Hardware* y *Software* que permiten la comunicación y su gestión, entre los clientes y los servidores.

2.1.2.2. CARACTERISTICAS DE LA ARQUITECTURA CLIENTE/SERVIDOR

En el modelo CLIENTE/SERVIDOR se pueden encontrar las siguientes características [13]:

- El cliente y el servidor pueden actuar como una sola entidad y también como entidades separadas, realizando actividades o tareas independientes.

[5] Mainframes: es una computadora grande, potente y costosa usada principalmente por una gran compañía para el procesamiento de una gran cantidad de datos.

- Las funciones de cliente y servidor pueden estar en plataformas separadas, o en la misma plataforma.

- Un servidor da servicio a múltiples clientes en forma concurrente.

- Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los clientes o de los servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.

- La interrelación entre el *hardware* y el *software* están basados en una infraestructura solida, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.

- Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones clientes. Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.

- las funciones Cliente/Servidor pueden ser dinámicas; es decir, un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red.

- Designa un modelo de construcción de sistemas informáticos de carácter distribuido.

2.1.2.3. Tipos De Clientes

- Cliente Liviano

Es una computadora cliente o un *software* de cliente que depende primariamente del servidor central para las tareas de procesamiento, y se enfoca principalmente en transportar la entrada y la salida entre el usuario y el servidor remoto. En este caso el servidor es rápidamente saturado por la carga de procesamiento de las peticiones de los clientes, generando una gran circulación de datos de interfaz en la red.

- Cliente Pesado

Un cliente pesado es una máquina que realiza tanto procesamiento como sea posible y transmite solamente los datos para las comunicaciones y el almacenamiento al servidor; realizando la mayor carga de trabajo en el cliente. En este caso no se realiza centralización de la gestión de la Base de Datos, por lo que existe una gran circulación de datos inútiles en la red.

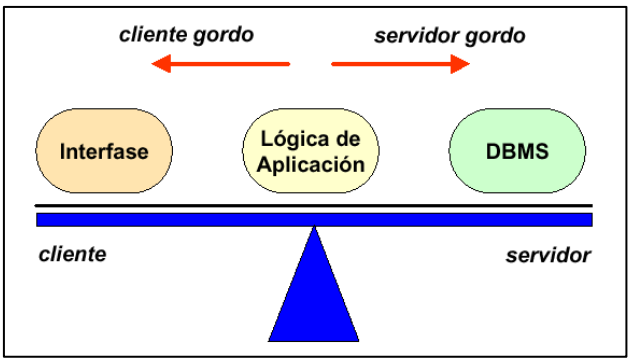

Figura 2.4.Cliente Pesado. Adaptado de [11].

2.1.2.4. Tipos De Servidor:

- Servidores de archivos.

Servidor donde se almacena archivos y aplicaciones de productividad como por ejemplo procesadores de texto, hojas de cálculo, etc.

- Servidores de bases de datos.

Servidor donde se almacenan las bases de datos, tablas, índices, siendo este uno de los servidores que más carga de peticiones tiene por parte de los clientes.

- Servidores de transacciones.

Servidor que cumple o procesa todas las transacciones, encargándose de validar en primera instancia, para generar posteriormente una solicitud al servidor de bases de datos.

- Servidores de *Groupware*.

Un servidor *groupware* es un *software* diseñado para permitir colaborar a los usuarios, sin importar la localización, vía Internet o vía Intranet corporativo y trabajar juntos en una atmósfera virtual.

- Servidores de objetos

Contienen objetos que deben estar fuera del servidor de base de datos. Estos objetos pueden ser videos, imágenes, objetos multimedia en general.

- Servidores *Web*

Se usan como una forma inteligente para comunicación entre empresas a través de Internet. Este servidor permite transacciones con el acondicionamiento de un *browser* específico.

2.1.3. ARQUITECTURA CLIENTE-SERVIDOR DE TRES NIVELES

Es una arquitectura cliente-servidor donde el procesamiento se encuentra alojado en tres niveles, dividiendo las funciones de manera explícita para cada uno de ellos, como se presenta a continuación [6 -13]:

- Nivel de presentación también conocido como interfaz de usuario.
- Nivel de cálculo donde se encuentra el modelado del negocio.
- Nivel de almacenamiento el cual permite el acceso a datos.

Cada una de las capa solo tiene relación con su capa adyacente, permitiendo centralizar la gestión de la reglas del negocio en un único lugar, sin tener que duplicarlas en cada aplicación. En una arquitectura de 3 capas los clientes solicitan y envían información a la aplicación centralizada, pero no al gestor de base de datos en el servidor [9].

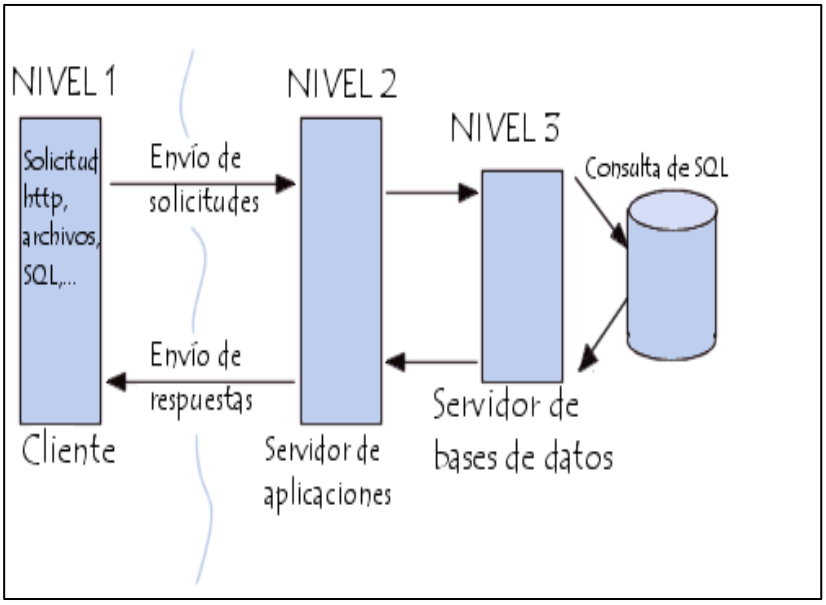

Figura 2.5. Arquitectura de tres niveles. Adaptado de [13].

Ventajas:

- Los componentes de la aplicación pueden ser desarrollados en cualquier lenguaje, posibilitando que el grupo de desarrolladores no se centre en el uso de un solo lenguaje.
- Los componentes están centralizados permitiendo un uso sencillo, un desarrollo flexible y mantenimiento simple.
- Cada componente de la aplicación pueden estar esparcido en múltiples servidores permitiendo una mayor escalabilidad.
- Los problemas de limitación para las conexiones a las bases de datos se minimizan ya que la base de datos solo es vista desde la capa intermedia y no desde todos los clientes.
- No es necesario que las conexiones y los manejadores de las bases de datos estén cargadas en los clientes.
- Permite un mayor grado de flexibilidad.
- Brinda un nivel de seguridad mayor, debido a que este tema se puede definir independientemente para cada servicio y en cada nivel.
- Mejor rendimiento, ya que las tareas se comparten entre servidores.

Desventajas:

- Este tipo de arquitectura pone más carga en la red, debido a una mayor cantidad de tráfico de la red.
- Es mucho más difícil programar y probar el *software*, en comparación con la arquitectura de dos niveles debido a que tienen que comunicarse más dispositivos para terminar la transacción de un usuario.

2.1.4. ARQUITECTURA DE MÚLTIPLES NIVELES

En la arquitectura en 3 niveles cada servidor por nivel o capa realiza una tarea especializada para brindar un servicio, permitiendo que un servidor pueda utilizar los servicios de otros servidores para proporcionar su propio servicio. Por consiguiente, la arquitectura en 3 niveles es potencialmente una arquitectura en N-niveles.

Ventajas:

- La ventaja fundamental de una arquitectura de n-capas comparado con una arquitectura de dos niveles radica en la separación orientada al exterior del proceso, permitiendo mejorar el balance de la carga en procesamiento en los diversos servidores; siendo aún más escalable, que en un nivel de 2 capas.

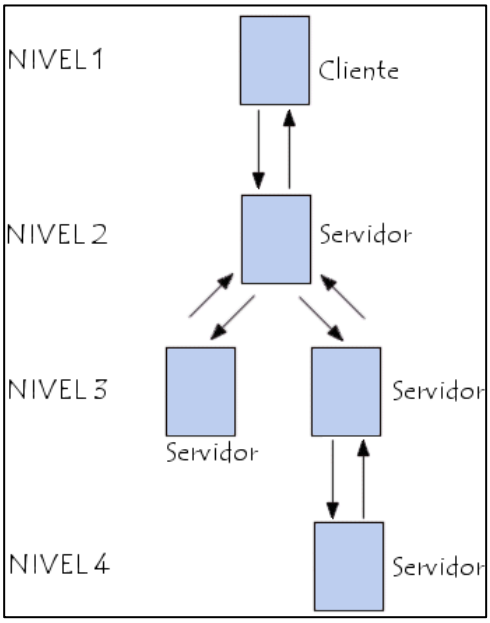

Figura 2.6. Arquitectura de Múltiples niveles. Adaptado de [9].

2.1.5. ARQUITECTURA ORIENTADA A SERVICIOS DEL CLIENTE (SOA, *Service Oriented Architecture*)

La arquitectura SOA, es un concepto de arquitectura de *software* que define la utilización de servicios para dar soporte a los requisitos del negocio. Esta permite la creación de sistemas de información altamente escalables que reflejan el negocio de la organización y a su vez brindan una forma definida de exposición e invocación de servicios, facilitando la interacción entre distintos sistemas propios o de terceros [7], [9] y [12].

Esta arquitectura define las siguientes capas de *software*:

- Aplicaciones básicas: Sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad.

- De exposición de funcionalidades: donde las funcionalidades de la capa aplicativa son expuestas en forma de servicios.

- De integración de servicios: donde facilitan el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración.

- De composición de procesos: definen el proceso en términos del negocio y sus necesidades, y que varía en función del negocio.

- De entrega: donde los servicios son desplegados a los usuarios finales.

La metodología de modelado y diseño para aplicaciones SOA se conoce como análisis y diseño orientado a servicios; por tanto, la arquitectura SOA es un marco de trabajo para el desarrollo e implementación de *software*. Un proyecto SOA tiene éxito si se crean servicios comunes que son solicitados por clientes o mediadores para implementar los procesos de negocio. Cuando se habla de una arquitectura orientada a servicios se hace referencia a un grupo de servicios residentes en Internet o en una intranet, usando servicios *web* [9]. Existen diversos estándares relacionados a los servicios *web* incluyéndose los siguientes:

- XML
- HTTP
- SOAP
- REST
- WSDL
- UDDI

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la utilización de Servicios *Web* (empleando SOAP y WSDL) en su implementación, no obstante se puede implementar SOA utilizando cualquier tecnología basada en servicios. En ese sentido las arquitecturas SOA están formadas por servicios de aplicación débilmente acoplados y altamente interoperables; los cuales para comunicarse entre sí, se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación. La definición de la interfaz encapsula (oculta) las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo (como Plataforma *Java* o *Microsoft* .Net). Con esta arquitectura, se pretende que los componentes de *software* desarrollados sean muy reutilizables, ya que la interfaz se define siguiendo un estándar [8].

Ventajas:

- Mejora en los tiempos de realización de cambios en procesos.
- Facilidad para evolucionar a modelos de negocios basados en tercerización.
- Facilidad para abordar modelos de negocios basados en colaboración con otros entes (socios, proveedores).
- Poder para reemplazar elementos de la capa aplicativa SOA sin interrumpir en el proceso de negocio.
- Facilidad para la integración de tecnologías diferentes.
- Reduce el nivel de acoplamiento.
- Clara definición de roles de desarrollo.
- Definición de seguridad más clara.
- Fácil testeo.
- Mejora la mantención.
- Favorece la reutilización.
- Favorece el desarrollo en paralelo.
- Permite una escalabilidad sencilla.

- Permite un mapeo directo entre los procesos y los sistemas.
- Permite un monitoreo preciso.
- Permite la interoperabilidad.

Desventajas:

- SOA depende de la implementación de estándares. Sin estándares, la comunicación entre aplicaciones requiere de mucho tiempo y código.
- SOA no es para aplicaciones con alto nivel de transferencia de datos, aplicaciones que no requieren de implementación del tipo *request/response* y para aplicaciones que tienen un corto periodo de vida.
- Incrementalmente se hace difícil y costoso el ser capaz de cumplir con los protocolos y hablar con un servicio.
- Implica conocer los procesos del negocio, clasificarlos, extraer las funciones que son comunes a ellos, estandarizarlas y formar con ellas capas de servicios que serán requeridas por cualquier proceso de negocio.
- En la medida en que un servicio de negocio, vaya siendo incorporado en la definición de los procesos de negocio, dicho servicio aumentara su nivel de criticidad. Con lo cual cada que se requiera efectuar una actualización en dicho servicio (por ejemplo, un cambio en el código, una interfaz nueva, etc.), deberá evaluarse previamente el impacto y tener mucho cuidado con su implementación. Sin embargo, parte de la problemática anterior, puede ser solventada en virtud a un buen diseño del servicio.

2.2. LENGUAJES DE PROGRAMACIÓN

2.2.1. JAVA

Es un lenguaje de programación que actualmente pertenece a la compañía *Oracle*, maneja gran similitud con lenguajes como C y C++, es de bajo nivel ya que sus instrucciones ejercen menos control directo sobre el *hardware*. A nivel global *Java* maneja un estándar para todas las aplicaciones de red, como móviles, juegos y robustos *software* para empresas [14].

Es un lenguaje que puede ser probado, ajustado y ampliado, posee un diseño que posibilita un alto rendimiento en el desarrollo de aplicaciones portables en distintas plataformas [15].

Otras características de *Java*, es su gran uso para optimizar las comunicaciones y la intervención del usuario final y su reducción de costos por ser de uso libre [14].

Ventajas:

- Es libre y gratuito.
- Permite Escribir *software* en una plataforma y ejecutarla virtualmente en otra.
- Es modular y un mismo código puede tener varios usos *software* (distintas aplicaciones, mismo código).
- Es posible crear programas que se puedan ejecutar en un explorador y acceder a servicios *web* disponibles.
- Permite crear aplicaciones o servicios con un gran nivel de personalización.
- Permite el diseño de aplicaciones potentes y eficaces para teléfonos móviles, procesadores remotos, productos de consumo y prácticamente cualquier otro dispositivo electrónico.
- Funciona en varios sistemas operativos como *Windows*, *Linux*, *Mac Os* entre otros.

Desventajas:

- Su sintaxis es poco entendible.
- Se debe tener una noción previa de programación orientada a objetos para su manejo.
- Corre en una maquina virtual, lo que puede ocasionar que sea más lento que otros lenguajes.

2.2.2. C#

Lenguaje orientado a objetos desarrollado por *Microsoft*, el cual pertenece también a su plataforma .Net, permitiendo estandarizarla. C# se diseña bajo la ejecución de aplicaciones en un entorno virtual, su escritura elemental deriva de C/C++ y maneja modelos de objetos en la plataforma .Net; siendo análogo al que contiene *Java* [16].

C# permite programar de forma independiente ya que su plataforma .Net esta estandarizada; brindando a su compilador la capacidad para el desarrollo en diferentes plataformas de *Microsoft* [17].

Ventajas:

- C# tiene una gama más grande y definida en tipos de datos en comparación a los que se encuentran en C, C++ o *Java*.
- No es necesario tener en cuenta archivos de cabecera .h.
- No tiene en cuenta el orden en que se definen las clases y las funciones.
- Puede minimizar errores que escapan de la fase de prueba.
- Es orientado a objetos.
- Soporta sobrecarga de operadores.

Desventajas:

- Es necesario tener una versión reciente de *Visual Studio* .Net.
- Es muy pesado ya que debe tener alrededor de 4 *Giga Bytes* de espacio libre solo para instalación.
- Es necesario tener conocimientos previos de lenguajes de programación, para consultar algún tutorial más explícito sobre la programación en C#, además de tener que contar con una conexión a Internet.
- C# solo se puede implementar en entornos *Windows*.

2.2.3. PHYTON

Lenguaje, que contrario a uno de compilación, se diseño para que sea ejecutado por un intérprete, se caracteriza por enfatizar en una escritura clara que permite que haya un código comprensible. Es un lenguaje de alto nivel, lo que implica mayor claridad en su código. Contrasta con la programación declarativa ya que le indica a la maquina como

debe hacer la tarea, también permite variar el estilo de programación e incluso tolera un lenguaje orientado a objetos [18] y [19].

Ventajas:

- Es un lenguaje de alto nivel.
- Es fácil de entender la sintaxis a nivel de usuario.
- Es de código abierto sin costo.
- Es portable, con poca posibilidad de cambios en ejecución en diferentes maquinas.
- La posibilidad de error es menor.
- Es modular.
- Es orientado a objetos.
- Ya que tiene tipo dinámico no interesa el número de declaración de variables.
- Aplicaciones hechas en *Virtual Box* pueden ejecutarse en *Linux* gracias al proyecto mono.

Desventajas:

- Compatible con la Licencia Pública General (GPL, *General Public License*) de GNU a partir de la versión 2.1.1, anteriores no lo es.
- Por ser lenguaje de alto nivel es necesario traducirlo a lenguaje maquina lo que demanda tiempo considerable.
- Aplicaciones hechas para *Linux* aun no son 100% compatibles.
- Es más lento en lenguaje compilado o ensamblador.
- Su sintaxis está basada en acomodar espacios de identacion.
- No está incluido en una empresa de *software* por ser relativamente nuevo lo que no genera un buen soporte.

2.2.4. RUBY

En programación se le llama lenguaje dinámico, es simple, además de tener un código libre y enfocado a una gran productividad, teniendo una sintaxis clara y de forma natural. Es un programa en el que se busca incorporar partes importantes de otros lenguajes de programación, obteniendo así un lenguaje robusto de fácil manejo y un gran alcance en distintos desarrollos de proyectos [20].

Ruby es creado especialmente hacia la programación orientada a objetos y tiene como característica la posibilidad de incluir métodos junto con variables a instanciar para todos sus datos sin importar su tipo [21].

Ventajas:

- Es un lenguaje de fácil manejo.
- Es orientado a objetos.
- Es un lenguaje flexible que permite la manipulación por parte del usuario.
- No restringe al desarrollador en su sintaxis.
- Posee herencia simple y es modular.
- Se le posibilita la carga de bibliotecas de extensión en forma dinámica dependiendo del sistema operativo.
- Maneja excepciones, como *Java* y *Python*, facilitando el manejo de errores.

Desventajas:

- Es un lenguaje de programación nuevo con respecto a los ya existentes en el mercado como por ejemplo, *Java*.
- No tiene un gran soporte comparado con otros lenguajes como por ejemplo C# y PHP.
- Es lento en comparación con los ya existentes en el mercado.
- Debido a que es un lenguaje muy reciente es necesario aprender de nuevo sus sintaxis, lo que implica que la comunidad de programadores optan por quedarse con lo ya conocido y con un gran soporte como por ejemplo *Java*.

2.2.5. PHP

Es un lenguaje programación que se diseño exclusivamente para desarrollo *web*, aunque es utilizado también como un lenguaje de propósito general. Los servidores para interpretar el código PHP tienen que procesar un modulo de PHP del cual se obtiene una página *web*.También es posible que los comandos de PHP sean embebidos por HTML de forma directa del documento inicial o base, en vez de traer un archivo de afuera para el respectivo proceso de datos [22], [23] y [24]. Es un lenguaje de programación libre de alto nivel y ejecutado por el servidor.

Ventajas:

- Es Multiplataforma.
- Utiliza muchos módulos a los que llama extensiones.
- Posee Manejo de excepciones.
- Tiene Biblioteca nativa de funciones.
- Permite técnicas de programación orientada a objetos.
- Amplia documentación en su página oficial.
- Destacada conectividad con *MySQL*.
- Es libre.
- Trae consigo iteradores de datos.

Desventajas:

- Promueve creación de código desordenado y con un mantenimiento complejo.
- Es muy difícil de optimizar.
- Diseñado especialmente hacia un modo de realizar aplicaciones *Web* que es problemático y obsoleto.
-
- Es difícil de controlar su seguridad si no se tiene un buen dominio.
- No tiene separación de capas.
- No es compatible con GPL de GNU.

2.3. SERVIDORES DE APLICACIONES

2.3.1. *JBOSS*

Es un servidor principalmente para aplicaciones de Entorno Empresarial *Java* (J2EE, *Java Enviroment Enterprise*), de código libre implementado en *Java* básico. Como pertenece a *Java,* J*Boss* es utilizado en distintos sistemas operativos en el que tenga disponibilidad la máquina virtual de *Java* [25] y [26].

También contiene otros proyectos como: *JBoss* para Programación Orientada a Aspectos (*JBoss* AOP, *Jboss Aspect Oriented Programming*) el cual se diseña para trabajar con este tipo de programación, permitiendo adicionar de una forma fácil servicios empresariales (transacciones, seguridad, persistencia) a clases *Java* más sencillas [25] y [26].

Dentro de este servidor de aplicaciones se encuentra *Hibernate*, servicio de conexión con la base de datos. Posibilitando que los desarrolladores creen clases de persistencia mediante la utilización de lenguaje *Java* con la utilización de todos los paradigmas de orientación a objetos como asociación, herencia, polimorfismo, composición y todo Java en su totalidad [25].

JBoss ofrece una IDE eclipse para el *JBoss* AS, así la depuración y muchas acciones que van con el desarrollo de aplicaciones son realizadas desde eclipse. *JBoss* JBPM es una plataforma para lenguajes de procesos ejecutables, encargándose de gestionarlos procesos de negocio y todo lo que sean servicios [26].

JBoss tiene como características principales:

- Los productos licenciados con código libre no tienen costo adicional.
- Cumple los estándares.
- Es de alta confiabilidad empresarialmente hablando.
- Principalmente se orienta hacia a una arquitectura de servicios SOA.
- Permite su manipulación sin perder estabilidad.

2.3.2. APACHE

Conocido como Apache *Tomcat*, es un servidor *web* que se implementa en código libre con tecnologías como *Java Servlet* y *Java Server Pages*, se desarrollan con especificaciones de este servidor con entorno muy abierto y a su vez participativo [27].

En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y *Tomcat* es usado como servidor *web* autónomo en entornos con alto nivel de tráfico y alta disponibilidad [28].

Apache tiene como características principales:

- Esta escrito en *Java*.
- Funciona en cualquier sistema operativo que disponga de la máquina virtual *Java*.
- Posee funciones básicas HTTP.
- Altamente configurable
- Bases de datos de autenticación y negociado de contenido.
- Es modular de código abierto, multi-plataforma, extensible y con gran cantidad de soporte de inconvenientes via internet.

2.3.3. *GLASSFISH*

Este servidor de aplicaciones es de uso libre, con distribución bajo licencia *GNU* GPL. Tiene una versión de forma comercial la cual se llama *Oracle Glassfish Enterprise Server* [29].

Glassfish se basa en código proporcionado por *Sun* y *Oracle*, el cual brinda un módulo de persistencia llamado *TopLink*. *Glassfish* a su vez lo contiene el servidor *Sun Java System Application Server* de *Oracle* y que usa un componente adicional llamado *Grizzly* el cual es usado por *Java Nio*, esto con el fin de permitir escalabilidad y velocidad [30].

Glassfish tiene como características principales:

- Código abierto.
- Fácil instalación.
- Soporte completo con *Java* EE 5.
- Integración total con *Netbeans*.

- Mucha documentación sobre uso, administración y desarrollo.

- Consola de administración sencilla y amigable.

2.4. MODELO DE DATOS

El modelo de datos está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades y de relaciones entre esos objetos; que permiten describir la información relevante de cierta parte de la realidad mediante un conjunto de representaciones gráficas [31].

Un modelo de datos es un lenguaje quese divide en los siguientes sub-lenguajes:

- Un Lenguaje de definición de datos (DDL, *Data Definition Language*), orientado a describir de una forma abstracta las estructuras de datos y las restricciones de integridad.

- Un lenguaje de manipulación de datos (DML, *Data Manipulation Language*), orientado a describir las operaciones de manipulación de los datos. Que también es conocido como lenguaje de consulta (QL, *Query Language*).

2.4.1. Modelos de Datos Conceptuales

Es el modelo orientado a la descripción de estructuras de datos y restricciones de integridad, usado principalmente durante la etapa de análisis de un problema dado y orientado a representar los elementos que intervienen en ese problema y sus relaciones [32].

2.4.1.1. Modelo Entidad-Relación.

2.4.1.1.1. Entidad

Se entiende entidad como un objeto del mundo real que es posible distinguir del resto de objetos; del cual interesan algunas propiedades. Teniendo una existencia independiente tanto física (concreta) como conceptualmente (abstracta), es decir, diferenciándose unívocamente de otro objeto o cosa, incluso siendo del mismo tipo, o una misma entidad [33].

Una instancia de una entidad representa un objeto particular de la entidad; por lo cual todas las entidades deben cumplir las siguientes reglas:

- Los nombres de las entidades deben ser únicos
  En un modelo de datos, no debe haber dos entidades con el mismo nombre. Esto permite especificar precisamente qué entidades son de interés sin ambigüedad.

- Cada entidad debe tener una llave primaria
  Cada entidad debe tener una llave primaria y sólo una llave primaria ya sea simple o compuesta.

Existen dos tipos de entidades:

- Regular o fuerte. Tiene existencia por sí misma en el universo del discurso, independientemente de cualquier otra entidad.

- Débil. Depende de alguna entidad existente en el universo del discurso. Al desaparecer la relación, desaparecerá la entidad débil vinculada a la misma. Por lo cual una entidad débil no puede ser identificada solamente por sus atributos.

2.4.1.1.2. Atributos

Los atributos son cualidades, características de interés o hechos que definen o identifican a una entidad; permitiendo a través de estas propiedades particulares describir a cada entidad dentro de un conjunto de entidades. Estos, portan toda la información adicional o extensiva de una entidad dentro del universo del discurso. Además, a nivel semántico todos los atributos son sustantivos o adjetivos calificativos, pero no todos los sustantivos o adjetivos calificativos son atributos [31], [32] y [33].

Para decidir si un objeto debe representarse como entidad o atributo, el diseñador debe considerar si el objeto describe a otro objeto y si tiene valores para sus instancias. En ese caso, es mejor representar el objeto como un atributo; sin embargo, si es difícil identificar valores, el objeto podría ser una entidad. También el diseñador es quien se encarga de definir la cantidad de atributos que se utilizan o implementan para una entidad, seleccionando los que considere más relevantes [34].

Todas las entidades deben cumplir las siguientes reglas:

- Los nombres de los atributos deben ser únicos en la entidad
  Cada atributo en una entidad debe tener un nombre único. Sin embargo, los atributos en diferentes entidades pueden compartir el mismo nombre.

- Los datos de los atributos deben ser atómicos
  La atomicidad a nivel de modelado significa la capacidad para que un objeto sea indivisible; es decir, si un atributo puede descomponerse en partes más pequeñas,

debe ser redefinido como dos o más atributos. La excepción a esta regla se aplica avalores de fecha y hora que generalmente son tratados como un tipo de dato.

a) Dominio de un atributo

El conjunto de valores permitidos para cada atributo se llama dominio del atributo y es posible tener diferentes atributos con el mismo dominio. Este hace referencia al tipo de datos que será almacenado o a restricciones en los valores que el atributo puede tomar (cadenas de caracteres, números, solo dos letras, solo números mayores que cero, solo números enteros, etcetera.) [35].

b) Tipos de Atributos:

- Atributo Simple. Un atributo simple es un atributo que tiene un solo componente, que no se puede dividir en partes más pequeñas que tengan un significado propio.

- Atributo Compuesto. Un atributo compuesto es un atributo con varios componentes, en el cual cada componente tiene un significado por sí mismo. Un grupo de atributos se representa mediante un atributo compuesto cuando tienen afinidad en cuanto a su significado, o en cuanto a su uso.

- Atributo con Valores único. Atributo que tiene un solo valor para cada ocurrencia en un tipo de entidad o la relación a la que pertenece.

- Atributo Multi-valorado. Es un tipo de atributo que tiene varios valores para cada ocurrencia (instancia) de la entidad o relación a la que pertenece. Es decir; puede tener un número máximo y un número mínimo de valores.

- Atributo Obligatorio / Opcional. Atributo que siempre debe tomar algún valor para cada instancia de la entidad o la relación.

- Atributo Derivado. Un atributo derivado es aquel que representa un valor que se puede obtener a partir del valor de uno o varios atributos, que no necesariamente deben pertenecer a la misma entidad o relación.

- Cuando algún atributo correspondiente a una entidad no tiene un valor determinado, recibe el valor nulo, bien sea porque no se conoce, porque no existe o porque no se sabe nada al respecto del mismo.

2.4.1.1.3. Relación

Una relación es una asociación entre diferentes entidades, describiendo cierta dependencia entre estas o permitiendo la asociación de las mismas. Por lo que una relación tiene sentido al expresar las entidades que relaciona [32 - 35].

Cada relación tiene un nombre que describe su función y al igual que las entidades existen tipos de relaciones:

- Regular o fuerte. Todas las entidades que relaciona son fuertes.
- Débil. Al menos una de las entidades que relaciona es una entidad débil.

2.4.1.1.4. Grado de una Relación

Una relación puede asociar dos o más entidades. El número de entidades que asocia en una relación es lo que determina el grado de la misma. Las entidades que están involucradas en una determinada relación se denominan entidades participantes [31].

- Unaria. Relación con la misma entidad (grado 1); también se conoce como relación reflexiva donde la misma entidad participa más de una vez en la relación con distintos papeles. El nombre de estos papeles es importante para determinar la función de cada participación [32].
- Binaria. Relación entre dos entidades (grado 2), es la más frecuente.
- Ternaria.Relación entre tres entidades (grado 3).
- N-aria. Relación entre n entidades, es muy poco frecuente.
- Caso especial circular o cíclica. Relación entre tres entidades A, B y C en la cual A se relaciona con B, B se relaciona con C y C se relaciona con A.

2.4.1.1.5. Restricciones

Son reglas del universo del discurso a las que las entidades y sus relaciones se deben adaptar, evitando así problemas de consistencia durante el proceso de creación de modelos de entidad de relación [33].

2.4.1.1.6. Correspondencia de cardinalidades

Cardinalidad en una relación es el número de veces que una entidad aparece asociada a otra entidad. Es decir; la cantidad de ocurrencias (instancias) de una entidad que se

pueden asociar a un elemento de la otra entidad a través de una relación. Para el modelo ER es de mayor relevancia la cardinalidad mínima y la cardinalidad máxima de una relación, ya que esta información permite deducir qué tipo de relación se establece y las posibles restricciones que se deben aplicar [34].

- Cardinalidad Mínima. Es el mínimo número de asociaciones que una instancia de una entidad puede presentar en una relación conocida con otra entidad.

- Cardinalidad Máxima. Es el máximo número de asociaciones que una instancia de una entidad puede presentar en una relación conocida con otra entidad.

En cualquier caso las cardinalidades mínima y máxima podrán ser cero (0), uno (1) o muchos (N), es así como las posibles cardinalidades son:

- Cero a uno (0,1): Una instancia (A1, A2, A3, A4) de la entidad A puede relacionarse con una instancia (B1, B2, B3, B4) de la entidad B o no tener relación con alguna instancia de B. Pero una instancia de la entidad B solo puede relacionarse con una instancia de A.

- Uno a uno (1,1): Una instancia de la entidad A se relaciona únicamente con una instancia de la entidad B y viceversa.

- Cero a varios (0, N): Una instancia de la entidad A puede relacionarse con muchas instancias de la entidad B o no tener relación con alguna instancia de B. Pero una instancia de la entidad B se relaciona únicamente con una instancia de A.

- Uno a Varios (1, N): Una instancia de la entidad A puede relacionarse con muchas instancias de la entidad B. Pero una instancia de la entidad B se relaciona únicamente con una instancia de A.

- Varios a varios (N, N): Una instancia de la entidad A puede relacionarse con muchas instancias de la entidad B y viceversa.

2.4.1.1.7. Restricciones de participación (Cardinalidad Mínima)

Dentro del modelado es posible encontrar casos en los cuales no todos los miembros de una entidad participen en la relación y viceversa [35]. A estos tipos de participación se les conoce como:

- Participación Total:
  Si cada miembro de una entidad debe participar en la relación.

- Participación Parcial

Cuando algún miembro (al menos uno) de la entidad no participa en la relación.

2.4.1.1.8. Restricciones de integridad

Dentro del modelo también se presentan restricciones sobre los valores que puede tomar cada atributo dentro de una entidad, basado en la representación del modelo del negocio [32].

- Verificación (*Check*). Comprueba que en todas las operaciones de actualización sobre una ocurrencia de la entidad se cumpla un predicado que se establece al momento del diseño del atributo; de no ser así se rechaza la operación.

- Aserción (*Assertion*). Trabaja de forma similar que la verificación, con excepción de que esta validación actúa sobre un conjunto de relaciones

- Valores nulos. Se puede definir un valor nulo como una señal usada para representar información desconocida, inaplicable, inexistente, no válida, no proporcionada, indefinida, etc.

- Valores Obligatorios. Se puede definir valores que no pueden ser nulos, a fin de representar información de carácter importante para el negocio.

2.4.1.1.9. Claves

Es un subconjunto del conjunto de atributos comunes en una colección de entidades, que permite identificar de forma única a cada instancia de las entidades pertenecientes a dicha colección. Asimismo, permiten distinguir entre sí las relaciones de un conjunto de relaciones [30 - 35].

Dentro de los conjuntos de entidades existen los siguientes tipos de claves:

- Clave primaria simple. Una clave primaria simple es un atributo que identifican de forma única a una entidad. Esto significa que siempre permite distinguir una instancia de la entidad de otra. Para identificar una clave primaria se requiere considerar el significado de los atributos, una noción semántica, antes de decidir si existen extensiones únicas. Las claves primarias representan una restricción que previene que dos entidades tengan el mismo valor para estos atributos. Una característica importante de las claves primarias, es que sus atributos no pueden tener valores nulos o duplicados.

- Clave primaria compuesta. Una clave primaria compuesta es un conjunto de atributos que por sí solos no son clave primaria, pero al agruparse identifican de

forma única a una entidad, teniendo las mismas propiedades que una clave primaria simple.

2.4.2. Modelos de Datos Lógicos

Orientado más a las operaciones que a la descripción de la realidad, es implementado en gran medida en los Sistemas de Gestión de Base de Datos (DBMS, *Data Base Management System*).

2.4.2.1. Modelo relacional

Es un modelo de datos que basado en la lógica de predicados y en la teoría de conjuntos realiza la gestión de una base de datos; siendo el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente.

Su idea fundamental es el uso de relaciones entre entidades, considerándose en forma lógica como conjuntos de datos llamados tuplas (Registros), pensando en cada relación como si fuese una tabla que está compuesta por registros (cada fila de la tabla sería un registro o tupla) y columnas (campos).

En este modelo todos los datos son almacenados en relaciones y como cada relación es un conjunto de datos, el orden en el que éstos se almacenen no tiene relevancia, a diferencia de otros modelos donde el orden de los datos altera el resultado final. Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar por un usuario no experto; puesto que la información puede ser recuperada o almacenada por medio de consultas que ofrecen una amplia flexibilidad y poder para administrar la información.

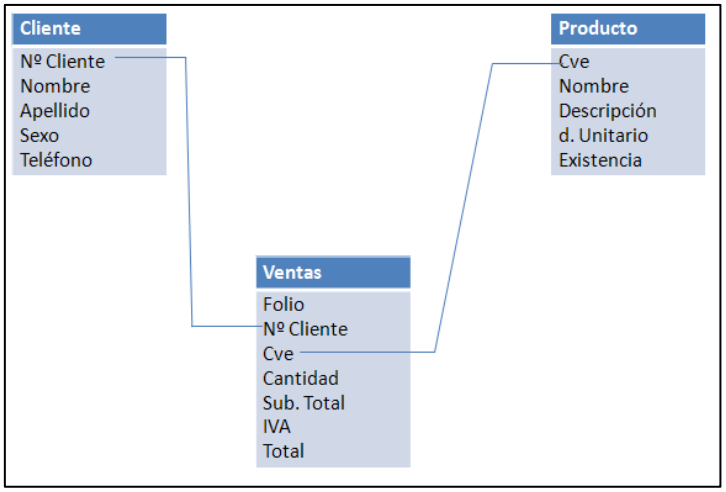

Figura 2.7. Modelo Relacional. Adaptado de [33].

2.5. SISTEMAS OPERATIVOS

Un sistema operativo es el *software* básico de un equipo que permite incluir programas y una interfaz para la interacción del usuario con el *hardware* del sistema y las aplicaciones de este**.**

2.5.1. *WINDOWS SERVER*

Es el nombre que se le da al conjunto de servidores con sistemas operativos *Windows server*, pertenecientes a la empresa *Microsoft* [36]. Estos sistemas operativos *Windows Server* permiten tener una base consistente y soportar la carga por el trabajo del servidor y los requerimientos de aplicación de forma conjunta y al mismo tiempo, además de tener una fácil implementación y administración. Ahorran tiempo y minimizan los costos gracias a la innovación en herramientas virtuales junto con los recursos *web* y el constante mejoramiento de administración e integración de *Windows*, ofreciendo una plataforma con el propósito de tener datos dinámicos de forma central, en la nube o en su centro de datos [37].

Ventajas:

- Proporciona infraestructura integral que permite mejorar el aseguramiento de información.
- Permite fiabilidad, disponibilidad, y escalabilidad para que un cliente no final pueda garantizar servicios de red a un cliente final o usuario.
- Permite q los usuarios tengan autonomía de trabajo.
- Tiene herramientas flexibles que permiten ajustar el diseño del cliente.
- Tiene servidores *web* integrados, servidores de aplicación para desarrollo e implementación, servidores de transmisión que permiten un trabajo instantáneo de forma dinámica en la *web*.
- Ofrece herramientas de conexión *web* con aplicaciones del diseño del cliente.

Desventajas:

- Es de uso bajo licencia de alto costo.

- En su historial existen referencias de servidores con fallos de seguridad debido a su popularidad.
- Es difícil de configurar.
- El *hardware* y el pago del personal especializado en él, manejan un alto costo.

2.5.2. *LINUX*.

Es propiamente un núcleo (*Kernell*) de sistema operativo ya que brinda la interfaz entre *hardware* y cliente su nombre real es *GNU/Linux*, pero también permite trabajar proyectos del GNU en su entorno, es por eso que los desarrolladores lo conocen como *Linux*, además es reconocido por permitir desarrollo de *software* abierto, bajo licencia libre GPL de GNU, dejando realizar modificaciones, insertar especificaciones y aplicaciones con ciertos propósitos; siendo esto, conocido como distribuciones [38], [39].

Ventajas:

- Es gratuito aunque tiene algunas versiones que manejan un muy bajo costo.
- Su código es libre lo que permite modificaciones a medida.
- Es muy rápido debido a que tienen muchos clientes desarrolladores integrantes a nivel mundial que le dan soporte y con una gran eficiencia como consecuencia.
- Es seguro ya que los virus solo corren si el usuario accede a ellos, estando por lo general libre de virus.
- Gracias al gran soporte que tiene a nivel mundial siempre está en constante actualización.

Desventajas:

- No hay disponibilidad de forma continua de instaladores.
- Algunos desarrolladores hacen programas de uso único bajo licencia y con costo alguno.
- Existen aplicaciones de forma privativaque hasta el momento no tienen un alto rendimiento en *Linux*.

2.5.3. *RED HAT*

Es una distribución de GNU/Linux conocida como *Red Hat Linux*, sus inicios se remontan al año de 1994, era una versión en paquete de lo que ya tenía *Linux* con aplicaciones y soporte adicional, incluyendo toda la documentación por parte de *Linux*; en 2006 adquirió la compañía el dueño del servidor *JBoss* que hoy en día es de los mas importantes debido a su aplicación de código libre J2EE [40], **¡Error! No se encuentra el origen de la referencia.**.

Ventajas:

- Es gran soporte empresarial a manera preventiva principalmente.
- Tiene un rendimiento alto.
- Es muy seguro.
- Tiene una gran escalabilidad y disponibilidad.
- Es una distribución de *GNU /Linux*.

Desventajas:

- Aunque es seguro su costo es alto.
- Maneja un costo elevado para licencia, sin contar el costo adicional para las actualizaciones.
- Debido a su gran costo y gran demanda de sistemas operativos de *Microsoft* con *Windows server* principalmente, tiene poca acogida en la comunidad mundial de desarrolladores *software*.
- Tiene una gran competencia con *Microsoft* pues el sistema operativo *Windows* tiene el mercado mundial.

2.5.4. CENTOS

Es un sistema operativo de código abierto que se fusiono con *Red Hat* para tener una gran plataforma de código libre [42].

Ventajas:

- Es una distribución de *Linux* y es gratuito.
- Debido a su similitud con el sistema comercial y a su distribución de *Red Hat* y *Linux*, es muy estable.
- Es muy rápido al correr programas simples y básicos evitando bloqueos.
- Maneja una gran confiabilidad ya que su actulizacion tiene un lapso de tiempo largo debido a que es distribución de *Red Hat* y *Linux*.

Desventajas:

- Ya que Centos corre programas básicos y estables no es muy funcional en comparación con otras versiones de programas en *Linux* con gran compatibilidad.
- No intercambia archivos de texto.
- Maneja mucha dificultad en la migración de código e informacion.

2.6. METODOLOGIA DE DESARROLLO

Esta técnica de ingeniería se refiere al uso de métodos basados en la estructura y forma del desarrollo *software* junto con la planificación, logrando así obtener un grado de control sobre éste; usualmente se realiza mediante *frameworks*[6]que a su vez tienen su propia marca que los distingue dependiendo de la compañía que los suministra [43].

[6] Frameworks: es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos.

2.6.1. PROCESO UNIFICADO RACIONAL (RUP, *RATIONAL UNIFIED PROCESS*)

Esta metodología se encarga de asignar tareas y funciones con grado de responsabilidad para el desarrollo *software* en una empresa, necesita de una cantidad considerable de programadores ya que aquí se define los parámetros de actividades y artefactos en totalidad del proyecto, midiendo la velocidad de las iteraciones de acuerdo a lo establecido inicialmente como por ejemplo fechas, tareas, horario. Es una metodología que antes de poner en marcha las características indicadas identifica y prueba una arquitectura solida, es por eso que se le llama a RUP una metodología de desarrollo general; teniendo como característica principal su división en 4 fases las cuales son: Inicio, elaboración, construcción y transmisión que se basan en asignar tiempos, iteración de fases, planificar el proyecto e hitos históricos [44].

Ventajas:

- Define disciplinas de modelado, análisis y gestión en cada fase.
- Su levantamiento de requerimientos es amplio y exhaustivo.
- Permite la detección de problemas en fases finales.
- Debido a su énfasis en la arquitectura minimiza cambios en el proceso de desarrollo.

Desventajas:

- Por su grado de complejidad no es recomendable para pequeños proyectos.
- Es un método que requiere gran capacidad de memoria y procesamiento.
-
- Su costo es elevado en proyectos pequeños.

2.6.2. PROGRAMACIÓN EXTREMA (XP, *XTREME PROGRAMMING*)

Esta metodología de desarrollo nace bajo la necesidad de reducir costos y aumentar la velocidad de realización de un proyecto, ya que requiere pequeños grupos de

programadores, se prueba en cada etapa de desarrollo, debido a que tiene iteraciones cortas permite mayor avance en el proyecto, basándose en variables como: Costo, tiempo, calidad y alcance que le permite durante el proceso de desarrollo la codificación el testeo y el diseño continuo [45].

Ventajas:

- Es un método muy rápido para el desarrollo de proyectos.
- Disminuye costos en el desarrollo *software*.
- Sus iteraciones son cortas con respecto de las de otros métodos.
- Su rediseño es continuo mejorando su simplicidad.

Desventajas:

- Su aplicación se limita para proyectos complejos con alto grado de iteración.
- Su rediseño continuo se puede volver un problema para grandes desarrollos.
- Se hace más difícil el cumplimiento de requerimientos del cliente por el rápido inicio en el desarrollo y el constante cambio y rediseño en casos de proyectos de gran tamaño y extensión.

2.6.3. SCRUM

Es una metodología que se basa en los procesos de trabajo grupal para optimizar los resultados en un proyecto determinado. Su principal característica es la entrega de manera parcial en corto plazo del producto final bajo iteraciones cortas, esto con el fin de inspeccionar si el cliente esta recibiendo el resultado adecuado de los requerimientos que solicito. Es un método muy eficiente respecto a la identificación de problemas debido a su construcción incremental **¡Error! No se encuentra el origen de la referencia.**.

Ventajas:

- Identifica eficientemente ineficiencias en el desarrollo del proyecto.
- Maneja iteraciones cortas.

- Acorta el tiempo de entrega.
- Es un método adaptable.

Desventajas:

- Su utilización se ve reducida par proyectos de corto plazo.
- Maneja un costo elevado en caso de fallo debido a la gran utilización de mano de obra capacitada.

2.7. ENTORNO DE DESARROLLO IDE

Es un *software* creado para permitir el desarrollo en distintos lenguajes si este lo permite, dependiendo de la clase de entorno que se esté manejando, su función es suministrar un ambiente idóneo para el programador, siendo así un programa de aplicación con iteraciones tales como depuradores, compiladores, editores de código y diferentes clases de constructores que faciliten la interfaz grafica **¡Error! No se encuentra el origen de la referencia.**.

2.7.1. ECLIPSE

Es un entorno de desarrollo con una interfaz sencilla y de fácil uso, se utiliza prioritariamente en aplicaciones de cliente enriquecido, es decir aplicaciones grandes y con extensos requerimientos con variaciones *web*, permitiendo lenguajes de programación en C++ y *Java* principalmente orientado a objetos [47].

Ventajas:

- Su entorno es modular.
- Es de uso libre.
- Su modularidad le permite utilizar otros lenguajes de programación.
- Es abierto y extensible.

- Posee una gran cantidad de *frameworks* que suministran herramientas de manipulación *software* y diseño grafico principalmente necesarias, para que el programador realice un desarrollo optimo.

Desventajas:

- Tiene un alto consumo de recurso del sistema.
- Su soporte nos es continuo.

2.7.2. NETBEAN

Es un entorno de desarrollo de uso libre, basado en el lenguaje de programación *Java* puesto que se diseño en base a este; es usado como *framework* debido a su plataforma de aplicación, el cual ha tenido notable mejoramiento en su rapidez y creación de nuevas librerías en sus últimas versiones, permitiendo dar soporte al lenguaje Ruby en aspectos de administración y depuración principalmente [49].

Ventajas:

- Es de uso libre.
- Permite soluciones rápidas.
- Es liviano.
- Constantemente esta ampliando su soporte abriéndose a nuevos lenguajes como Ruby Y PHP.
- Su plataforma permite cualquier tipo de aplicación.

Desventajas:

- Esta plataforma no tiene suficientes *plugins*.
- Aunque se está ampliando su soporte no es suficiente para grandes proyectos.

- No edita código HTML.

2.7.3. DELPHI

Es un en entorno que se especializa principalmente en programación de interfaz visual, no necesita tiempo de ejecución lo que lo hace mas rápido, también tiene herramientas que compilan sistemas operativos como *Windows* y *Linux*. Trabaja con programación de propósito general **¡Error! No se encuentra el origen de la referencia.**.

Ventajas:

- Tiene librerías suficientes que permiten el acceso a las Interfaces De Programación De Aplicaciones (API[7], *Application Programming Interface*) de *Windows*.
- Es rápido.

Desventajas:

- No es libre.
- Su soporte es limitado.
- No es multiplataforma como los entornos *Java*.

2.8. CARACTERÍSTICAS A CONSIDERAR PARA LA ARQUITECTURA SAP ERP

La tabla 2.1 resume las características que se identifican para realizar la arquitectura, teniendo en cuenta costos, licencias, eficiencia, rendimiento y otros aspectos que permitan llevar a consecución la propuesta de arquitectura para el desarrollo del módulo SAP ERP para *Axon Group*.

[7] API: es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Consideraciones Arquitectura	Tecnología	Característica
Arquitectura	Arquitectura de Múltiples Niveles	• Escalibilidad. • Balance de carga en Procesamiento. • Modularidad. • Separación de Capas.
Lenguaje de Programación	JAVA	• Libre y Gratuito, Permitiendo un Facíl acceso. • Modular. • Multi-plataforma. • Amplia Documentación.
Servidor de Aplicaciones	JBOSS AS GLASSFISH	• Libre y Gratuito, Permitiendo un Facíl acceso. • Modular. • Entorno empresarial, para grandes aplicaciones. • Facil instalación • Integración con Eclipse.
Base de Datos	PostgresSQL	• Libre y Gratuito, Permitiendo un Facíl acceso. • Robusta. • Intuitiva. • Integración con otras tecnologías. • Escalable. • Facíl proceso de migración.
Sistema Operativo	Linux	• Libre y Gratuito, Permitiendo un Facíl acceso. • Robusta. • Integración con otras tecnologías.
Metodología de desarrollo	XP	• Especial para grupos pequeños de trabajo. • Rápida. • Iteraciones cortas.
IDE	Eclipse	• Libre y Gratuito, Permitiendo un Facíl acceso. • Integración con otras tecnologías. • Integración con JBOSS

		AS. • Modular. • Extensible. • *Frameworks* conocidos.

Tabla 2.1. Consideraciones de Arquitectura. Por los Autores.

En este capítulo se abordó la descripción de las características más importantes de las tecnologías que permiten abordar el desarrollo de la arquitectura SAP ERP, a fin de identificar y seleccionar las tecnologías y técnicas que mejor se adapten a las condiciones del entorno de trabajo buscando integrar las buenas practicas de ingeniería, pero con las robustez suficiente para dar soporte a la demanda del mercado SAP; esto para plantear la arquitectura de desarrollo para una aplicación SAP ERP.

El siguiente capítulo plantea la arquitectura de desarrollo de *software* que permitió realizar un módulo SAP ERP para la empresa *Axon Group*, con la facilidad de extenderse a más módulos SAP permitiendo así tener la escalabilidad que toda aplicación de hoy en día debe presentar.

3. ARQUITECTURA DE DESARROLLO SAP ERP PARA AXON GROUP

3.1. ESPECIFICACIONES DE ARQUITECTURA

El proyecto para un modulo SAP ERP, se propuso sobre una arquitectura empresarial de *Java*, utilizando el patrón de múltiples niveles o capas, que garantiza fácil mantenimiento y separación de responsabilidades. La versión empresarial de *Java* brinda un API distinta para cada capa o nivel de una aplicación empresarial (como lo son capa de presentación, la capa de negocio y la capa de datos).

A continuación se identifica cada una de las capas de una aplicación multicapas.

- Capa de Presentación.
  - Capa Cilente. En esta capa es donde el cliente interactúa por medio de un navegador *Web*, un cliente móvil (Teléfono inteligente), una aplicación de escritorio, entre otros.
  - Capa Web. Esta capa puede estar ubicada en un servidor *web* y para este proyecto se utiliza *Java Server Faces*, teniendo la posibilidad de seleccionar entre Icefaces o PrimeFaces.
- Capa de Negocio.

  En esta capa se encuentra la tecnología de *Java*, *Enterprise Java Beans* (EJBs).

- Capa de Datos.

  Aquí se aloja la tecnología Java *Persistence* API (JPA), la cual permite la comunicación con la base de datos para leer y almacenar información en ella.

La figura 3.1. Ilustra la interacción entra las capas en el modelo de arquitectura de Múltiples niveles.

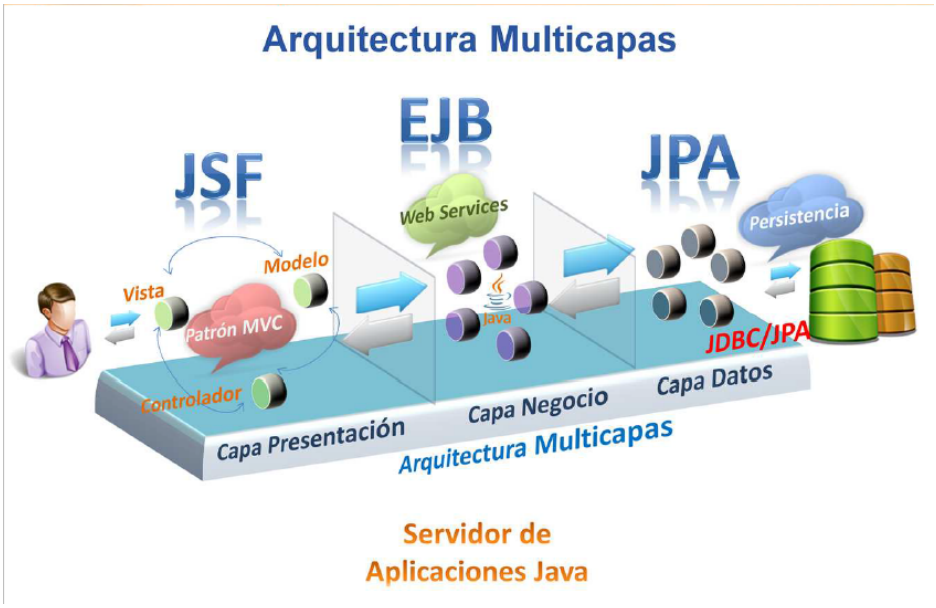

Figura 3.1. Arquitectura de Múltiples Capas. Por los Autores.

Es así que con las tecnologías disponibles en *Java* se planteo un diseño para la aplicación de cinco capas, utilizando 5 API's para ello. La figura 3.2. Indica cada uno de los niveles o capas que se utilizaron para la arquitectura propuesta para el desarrollo del SAP ERP, dentro de la teoría de la arquitectura multicapas.

En la construcción de la arquitectura fue necesario identificar las API's adecuadas para este proceso, por ello se estimó utilizar las tecnologías de *Java* en su versión estable, las cuales no son la ultima versión liberada, pero se encuentran catalogadas como tecnología nueva.

Esto debido a que la utilización de tecnología en versión de liberación o beta presenta problemas de inestabilidad, incompatibilidad y de fidelidad con respecto a otras tecnologías que no son *Java*, evitando con esto problemas al momento de implementación. Además, las versiones estables, permiten en lapsos de tiempos muy cortos actualizar a las versiones de liberación, cuando estas últimas se encuentren totalmente estables, haciendo a la arquitectura realizada escalable y migrable a nuevas tecnologías, pero con la premisa de garantizar siempre su uso.

Las API's utilizadas para la arquitectura se identifican en la tabla 3.1, destacando la tecnología, el estándar de factor y su versión.

3.2. PRESENTACIÓN (MB)

En el proceso de construcción de la arquitectura nivel a nivel, se inició con la capa de presentación, en la cual dos *framework* de trabajo se destacan sobre los demás; *Primefaces* e *Icefaces*, ambas tecnologías se utilizaron y se mezclaron dentro del código, permitiendo el diseño y programación de eventos, de una manera rápida, práctica, eficiente y altamente escalable; logrando que los módulos sean extensibles y adaptables a nuevas necesidades que surgan para la empresa.

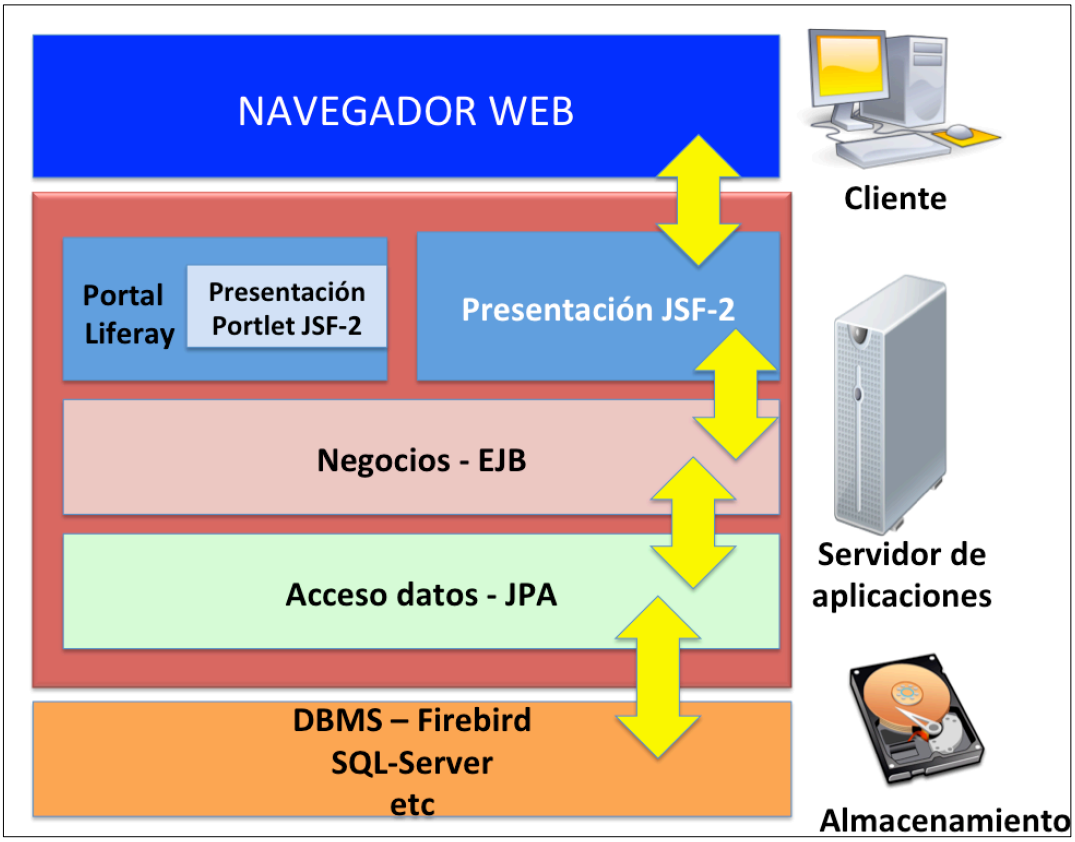

Figura 3.2. Arquitectura propuesta de Múltiples Capas para el SAP ERP. Por los Autores.

Capa	Tecnología	Estándar	Implementación
Persistencia	JPA 2.0	JSR 317 JEE6	Hibernate 4.1
Integración de servicios y Negocio	EJB 3.1	JSR 318 JEE6	
	CDI 1.1	JSR 229 JEE6	
	JAX-WS 2.0	JSR 224	Apache CXF 2.4
	JAX-RS 1.0	JSR 311	Resteasy 2.2.2.GA
Presentación	JSF 2.1	JSR 314	Mojarra
	FaceletsComponents		PrimeFaces 3.5, IceFace 3.0
	ManagedBeans		
Control de proyecto	Apache Maven		3.0.4
Servidor de aplicaciones	JBOSS AS, Glassfish 3.1		EAP 6.1.0.Alpha

Tabla 3.1. API's por nivel o capa de la Arquitectura. Por los Autores.

Así, con esta capa se generó la interfaz de interacción entre el usuario final del sistema y la arquitectura, encargándose de recibir todas las peticiones que vengan del usuario y su accionar al momento de relacionarse con la arquitectura a través de dispositivos con acceso *web*. Esta capa no contiene lógica de negocio, para ello usa la siguiente capa en la estratificación, permitiendo a la arquitectura separar tareas para ser más robusta.

La aplicación se generó utilizando *facelets* con componentes de *Prime Faces* o *Ice Faces,* realizando procesos de conversión de una clase *Java* a un servicio web con notaciones. Para ello fue necesario desarrollar dos elementos a este nivel:

- Páginas de vista con facelets (xhtml).
- Clase MB (*Managed Bean*) (con anotaciones de CDI).

Luego, La comunicación entre esta capa y la capa de Negocio quien es la siguiente capa en la jerarquía de capas, se realizó utilizando el Modelo de Datos.

3.3. NEGOCIO (BO)

El siguiente paso o nivel es la capa de negocio, la cual contiene toda la lógica de programación que permitió realizar las reglas que caracterizaron y parametrizaron el SAP-ERP, sin embargo, esta capa no interactúa directamente con el usuario o cliente de la aplicación, solo es llamada desde la capa de presentación o nivel superior a esta; del mismo modo, el nivel de negocio tampoco interactúa directamente con el almacenamiento de información en la base de datos, ya que esta tarea es asignada al nivel inferior o siguiente en la arquitectura a través de la capa de acceso a datos, también conocida como persistencia.

Las tecnologías JAX-WS y JAX-RS, resultaron determinantes para el desarrollo de la misma, aplicando ambas para la propuesta, pero vinculando junto con estas, las tecnologías EJB y CDI. Por ultimo fue necesario tener la interfaz para permitir el desacople entre las capas de negocio y presentación, con el fin de poder realizar modificaciones en las implementaciones de una manera transparente al usuario que interactúa con el sistema.

En esta capa fueron construidos dos componentes:

- Interfaz del BO (*Business Object)*
- Clase BOEJB (Implementación del BO) en un subpaquete EJB

La comunicación entre esta capa y la capa de acceso a datos se hace utilizando el Modelo de Datos (DM).

3.4. ACCESO A DATOS (DAO)

En esta capa se manejó el acceso a la capa de almacenamiento que es la última capa de la arquitectura. Adicional al proceso de acceder a una base de datos, en esta capa se puedo realizar el acceso a un sistema externo a través de un servicio *web* emulando ser un cliente de un servicio web que se consume.

En este nivel fue necesaria la generación de dos componentes:

- Interfaz del DAO (*Data AccesObject*)
- Clase DAO de tipo Implementacion (Implementación del DAO; esta puede ser de diferentes tipos: JPA, JDBC, WS *Client*, etc.)

La comunicación con la base de datos se realizó por medio de un Controlador de Acceso a Base de Datos *Java* (JDBC, *Java Data Base Controller*), complemento que permite la conexión entre el entorno de desarrollo y la base de datos. Además fue necesario configurar el *Data Source* en el servidor *Web* de aplicaciones.

3.5. MODELO DE DATOS (DM).

El modelo de datos es la representación del sistema a través de objetos de *Java*.
La comunicación entre las capas dentro del servidor de aplicaciones se realizó utilizando el DM. Estas son clases que contienen una representación de la información del sistema (ej. Usuario, Consulta, etc.).

Para hacer el mapeo entre estos objetos y una base de datos relacional se utilizó el API de JPA, que es nativo de *Java*.

Generando dos componentes para esta capa:

- Las clases *Java Beans* (propiedades privadas, con métodos *get* / *set*)
- Las clases que contienen las anotaciones de JPA que permiten realizar la persistencia en la capa de Acceso a Datos (*javax.persistence*)

3.6. SERVIDOR DE APLICACIONES Y CONTROL DEL PROYECTO.

Para el caso del servidor de aplicaciones se utilizaron las tecnologías *Jboss SA* y *Glassfish*, las cuales presentan acoplamiento con el entorno de desarrollo *Eclipse Indigo*, que además contiene todos los complementos requeridos para el desarrollo del proyecto.

En este caso la arquitectura se creó con el apoyo de la tecnología de gestión y construcción de proyectos *software Maven* para la aplicación web; realizando la creación de una proyecto estándar sobre esta tecnología, el cual se encuentra empaquetado en carpetas especiales con extensión WAR para la parte del servidor de aplicaciones *Jboss* y con extensión JAR para el caso de *Glassfish*.

Al final, *Maven* posibilitó la generación de una estructura en carpetas que organiza todas las capas de la arquitectura de manera modular y separadas unas de otras acorde a las tecnologías tratadas en los apartados, 3.2. a 3.5. facilitando incluir o dejar por fuera determinada funcionalidad o módulo a solicitud del empresa acorde con sus necesidades, por medio de una dependencia registrada en un archivo de configuración dentro de las carpetas del proyecto *Maven* que contiene la aplicación *web*. Permitiendo la escalabilidad de la arquitectura, pensando en la integración de otros módulos o submódulos del SAP ERP.

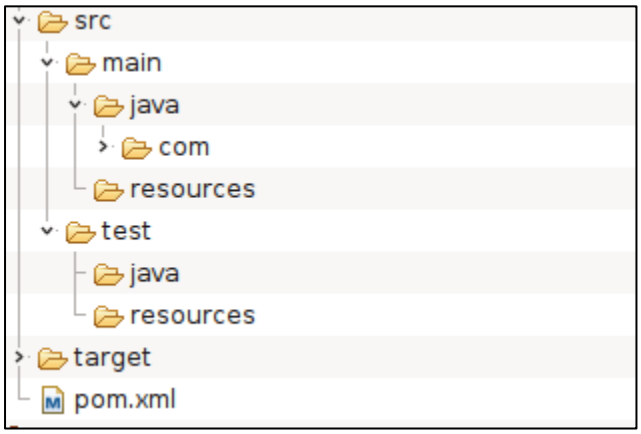

Figura 3.3. Proyecto *Maven,* separado en carpetas para la arquitectura. Por los Autores.

En el anexo A y B, se encuentra el manual de usuario de la aplicación implementada en las tecnologías indicadas en el capitulo 2 y soportada sobre la arquitectura propuesta en el capitulo 3.

En este capítulo se estableció la arquitectura de múltiples niveles o capas para el módulo SAP-ERP, en base a las técnicas y tecnologías identificadas en el capitulo dos, en el siguiente capítulo se presentan las conclusiones a las que se llegaron después del desarrollo de la proyecto de pasantía, las recomendaciones y los trabajos futuros que se pueden implementar en base a este tema.

4. CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

En este capítulo se presentan las conclusiones, recomendaciones y los trabajos futuros, resultado de la realización de esta pasantía. Las conclusiones se basan en los objetivos propuestos y en su desarrollo; las recomendaciones se dan acerca de la metodología de trabajo; los trabajos futuros con respecto a las distintas tecnologías empleadas, las arquitecturas de desarrollo y sus resultados.

4.1. CONCLUSIONES

4.1.1 Respecto a las tecnologías de desarrollo de *software*

- Es fundamental tener en cuenta el proceso de identificación de las características de las tecnologías de vanguardia, para plantear una adecuada arquitectura *software* que brinde las bondades requeridas por el cliente.

- La determinación y/o identificación de técnicas, elementos, tecnologías y características de los componentes *software*, fue esencial a fin de realizar la selección adecuada de la arquitetura del proyecto.

- El uso del servidor de aplicaciones *Jboss* brindó a la aplicación un nivel empresarial que posibilita a futuro el manejo de grandes volúmenes de tráfico y de usuarios, con la ventaja de no necesitar potentes recursos *hardware*, ajustándose a la pequeña y mediana empresa.

- Se seleccionó el lenguaje de programación *Java* en el presente trabajo, permitió utilizar multiples tecnologías libres, de acceso fácil y gratuito en las distintas capas de la arquitectura multinivel propuesta, además, el uso de la tecnología *Maven*

brindo una organización modular a nivel de *software* que brindara a futuro la oportunidad de escalar el tamaño de la aplicación.

- La metodología de desarrollo seleccionada permitio abordar un proyecto de gran envergadura, con pocos recursos monetarios, en especie y en tiempo, demostrando que un proyecto de *software* debe abordarse siguiendo una pautas de desarrollo, como la metodología XP.

- EL sistema operativo *Linux* es un aliado impresionante a la hora de desarrollar con pocas prestaciones de recursos *hardware* y sin la necesidad de pagar por licencias. Logrando tener un proyecto con el minimo de recursos dado su característica de pasantía.

4.1.2 Respecto a la arquitectura de *software*

- En este proyecto se identificó que la arquitectura de multiples niveles aumenta la escalabilidad de una aplicación; permitiendo un gran nivel de modularidad, además es posible lograr una arquitectura como esta, haciendo uso de tecnologías y técnicas libres y gratuitas.

- La arquitectura multinivel permite migrar de tecnologías en todos los niveles de la misma, sin afectar el resultado final de la aplicación debido a la separación por capas. Siendo además, transparente para el usuario final todos estos aspectos técnicos.

- La arquitectura propuesta y desarrollada permite llegar a distintas clases de usuarios finales, posibilitando que estos accedan a través de dispostivos móviles, de escritorio, u otros que tengan acceso a la red de internet.

- La arquitectura brinda la posibilidad de soportar otras áreas o procesos del SAP, permitiendo mantener estable un aplicativo.

4.2. RECOMENDACIONES

- Para abordar el análisis de problemas de *software*, mediante el uso de tecnologías de vanguardia, es vital la selección de una metodología de desarrollo de *software*, pues, esto permitirá abordar la solución al problema de forma clara y estratégica, permitiendo obtener unos resultados coherentes y satisfactorios.

- Para la selección de una arquitectura, es necesario realizar una exploración de sus características, las ventajas y sus defectos, la configuración, el acceso a la misma

entre otras, a fin de determinar si esta cumple con los requisitos planteados, evitando que se presenten inconvenientes durante la fase de desarrollo de la aplicación.

- Cuando se trabaje con proyectos de *software* en una arquitectura de multiples niveles, es recomendable mantener el proyecto estructurado y organizado, complementado con comentarios que describan el significado de las líneas. Por tanto utilizar un API como *Maven* permite mantener el desarrollo ordenado.

- Es recomendable utilizar las tecnologías y API's, en su versión de liberación estable, puesto que las versiones de prueba o que recientemente aparecen en el mercado son inestables y aun presentan problemas de compatibilidad con otras herramientas necesarias para el desarrollo de un proyecto software.

4.3. TRABAJOS FUTUROS

- La arquitectura considero el uso de *postgressql*, sin embargo, se plantea una arquitectura que haga uso de *Oracle* como motor de base de datos, puesto que este presenta mejores características y prestaciones a la hora de trabajar con empresa de mediano y gran tamaño.

- La arquitectura utilizó la versión Eclipse indigo que se encuentra en su etapa final de liberación, se propone la migración a la versión Eclipse Kepler para integrarla con nuevas versiones de *maven* y de *Glassfish.*

- La arquitecura se propuso para realizar el SAP ERP gestión de recursos humanos, sin embargo, aun no se abarca todas las ramas del SAP, por lo cual se propone integrar un SAP en la arquitectura ya creada.

REFERENCIAS

[1] "Axon Group", Axon Group, 2014. http://www.axongroup.com.co/.

[2] "AxonGroup", Productos y Servicios, 2014.http://www.axongroup.com.co/axon_productos.php.

[3] "SAP Business Suite", SAP Business Suite, 2014. https://help.sap.com/businesssuite

[4] "SAP ERP 6.0", SAP ERP 6.0, 2014. https://help.sap.com/erp/

[5] "SAP ERP", enterprise resource planning, 2014. http://en.wikipedia.org/wiki/SAP_ERP

[6] "Software architect", Software architect, 2014. http://en.wikipedia.org/wiki/Software_architect

[7] "Software architectural model", Software architectural model, 2014. http://en.wikipedia.org/wiki/Software_Architectural_Model

[8] C. Hofmeister, R. L. Nord, D. Soni, "Describing Software Architecture with UML," *Siemens Corporate Research, Princeton, New Jersey, USA,* Sept. 1999.

[9] L. Bass, P. Clements, R. Kazman, "Software Architecture in Practice", 2nd Edition, Addison Wesley, 2003.

[10] Ken Schwaber and Jeff Sutherland. Scrum.http://www.scrum.org/scrumguides/

[11] "Manifesto for Agile Software Development", Agile manifesto.http://agilemanifesto.org

[12] "R Redmine", Redmine Issue Tracker. Website, 2006 - 2010. http://www.redmine.org/

[13] "Arquitecto de software vs agil", Businness World TI, 2012.http://businessworldti.wordpress.com/2012/11/15/arquitecto-de-software-vs-agil/

[14] "Java y Tu", Java Website, 2014. http://www.java.com/es/

[15] "The Java Tutorials", Oracle, 2014. http://docs.oracle.com/javase/tutorial/

[16] "Visual C# resources", C# Website, 2014. http://msdn.microsoft.com/en-us/vstudio/hh341490.aspx

[17] "CSharp Language Center", mongoDB, 2014.http://docs.mongodb.org/ecosystem/drivers/csharp/

[18] "Python", Python Website, 2014. https://www.python.org

[19] "Python (programming language)", Wikipedia Website, 2014. . http://en.wikipedia.org/wiki/Python_%28programming_language%29

[20] "Ruby es…" Ruby Website, 2014. https://www.ruby-lang.org/es/

[21] "Ruby (programming language)", Wikipedia Website, 2014. http://en.wikipedia.org/wiki/Ruby_%28programming_language%29

[22] "PHP", PHP Website, 2014. http://www.php.net/

[23] "PHP 5 Introduction", W3Schools.com, 2014. http://www.w3schools.com/php/php_intro.asp

[24] "PHP", Wikipedia Website, 2014. http://en.wikipedia.org/wiki/PHP

[25] "Jboss Community" Jboss Website, 2014. https://www.jboss.org/overview/

[26] "Red Hat Jboss Middleware", Red Hat Website, 2014. http://www.redhat.com/products/jbossenterprisemiddleware/web-server/

[27] "The Apache Software Foundation", Apache Website, 2014. http://www.apache.org/

[28] "Servidor HTTP Apache", Wikipedia Website, 2014. http://es.wikipedia.org/wiki/Servidor_HTTP_Apache

[29] "GlassFish Server", GlassFishWebiste, 2014. https://glassfish.java.net/es/

[30] "GlassFish", Wikipedia Website, 2014. http://en.wikipedia.org/wiki/GlassFish

[31] "Data Model", Wikipedia Website, 2014. http://en.wikipedia.org/wiki/Data_model

[32] "Data Modeling", Agil data Website, 2014. http://www.agiledata.org/essays/dataModeling101.html

[33] The Data Description Language EAST Specification (CCSD0010). Recommendation for Space Data System Standards, CCSDS 644.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, November 2000.

[34] Data Entity Dictionary Specification Language (DEDSL)—Abstract Syntax (CCSD0011). Recommendation for Space Data System Standards, CCSDS 647.1-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, June 2001.

[35] Reference Model for Open Storage Systems Interconnection—Mass Storage System Reference Model. Version 5. Project 1244. New York: IEEE, 1994. http://www.ssswg.org/public_documents.html

[36] "Microsoft Businness para pequeñas y medianas empresas", Microsoft Website, 2014. http://www.microsoft.com/business/es-xl/Soluciones/Paginas/servidores-muevetusfichas.aspx

[37] "What's New in Windows Server 2012 R2 in the TechNet Library at", Microsoft Website, 2014.http://technet.microsoft.com/en-us/evalcenter/bb291020.aspx

[38] "Linux.org" Linux Website, 2014. http://www.linux.org/

[39] "Sobre linux" Linux Website habla hispana, 2014. http://www.linux-es.org/

[40] "redhat", redhatWebiste Colombia, 2014. http://co.redhat.com/

[41] "Red Hat Enterprise Linux", Red Hat Website, 2014. http://co.redhat.com/products/enterprise-linux/

[42] "New Look New CentOS", CentOS Website, 2014. http://www.centos.org/

[43] L. Night, T. Steinbach, V. Kellen "System Development Methodologies for Web Enabled E-Business: A Customization Paradigm," Noviembre 2001.http://www.kellen.net/SysDev.htm

[44] "IBM Rational Unified Process (RUP)", IBM Website, 2014. http://www-01.ibm.com/software/rational/rup/

[45] "Extreme Programming: A Gentle Introduction", XP Website, 2014.www.extremeprogramming.org

[46] "Modelo original de Scrum para desarrollo de software", Scrum Manager Website, 2014. http://www.scrummanager.net/bok/index.php?title=Modelo_original_de_Scrum_para_desarrollo_de_software

[47] "Integrated development environment", Wikipedia Website, 2014. http://en.wikipedia.org/wiki/Integrated_development_environment

[48] "Eclipse", Eclipse Website, 2014. https://www.eclipse.org/

[49] "Net Beans IDE", Net Beans Website, 2014. https://netbeans.org/

[50] "Delphi IDE Tools and Plug-Ins", Delphi Website, 2014. http://delphi.about.com/od/toppicks/tp/aatpaddin.htm

DESARROLLO DEL SAP COMO PASANTÍA PARA AXON GROUP

ANEXOS

Diego Fernando Perdomo Samboní
Ever Andres Molano Hurtado

Universidad del Cauca
Facultad de Ciencias naturales, exactas y de la educación
Popayán, Mayo de 2014

DESARROLLO DEL SAP COMO PASANTÍA PARA AXON GROUP

ANEXOS

Diego Fernando Perdomo Samboní
Ever Andres Molano Hurtado

Director: Mg. Luis Fernando Echeverri.

Universidad del Cauca
Facultad de Ciencias naturales, exactas y de la educación
Popayán, Mayo de 2014

TABLA DE CONTENIDO

LISTA DE FIGURAS

ANEXO A. MANUAL DE USUARIO SUB-MÓDULO SAP- ERP GESTIÓN DE RECURSOS HUMANOS

Se crearon dos roles para acceder a la aplicación, el primer rol tiene los privilegios de un administrador *web,* siendo posible crear, editar y eliminar usuarios, además de otorgar toda clase de permisos sobre el módulo de gestión de recursos humanos del SAP ERP. El otro rol es un usuario estándar de la aplicación el cual solo tiene acceso a consultar información sobre su hoja de tiempo y a realizar actualizaciones sobre esta. Para ello, el anfitrión que interactúa con la aplicación debe dar el usuario y la clave suministrada por el administrador web de la aplicación, quien es el encargado de generar toda cuenta de usuario de la aplicación, y luego dar clic en el botón *Login*.

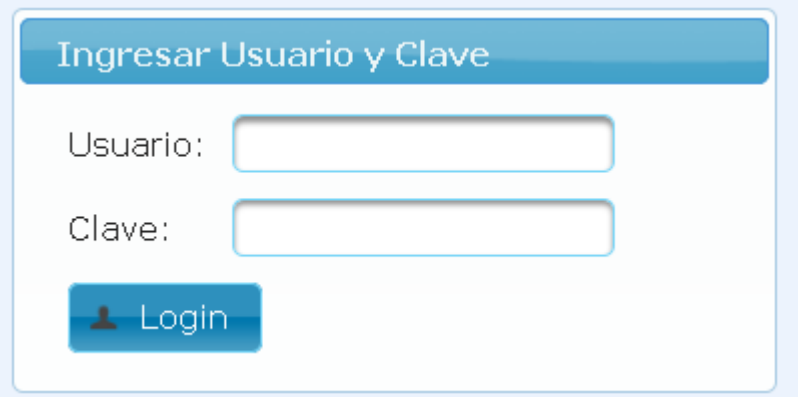

Figura A.1. Inicio de sesión Módulo de gestión de recursos humanos SAP ERP.

Es así, como se despliega un cuadro de texto que da la bienvenida al usuario en cuestión.

Figura A.2. Pantalla de bienvenida a usuarios.

Cuando se accede a través de la cuenta con rol o permisos de administrador, el sistema muestra dos sub-módulos del SAP ERP gestión de recursos humanos, el primero es “Seguridad” y el siguiente es “Gestión de Tiempo”; como se presenta a continuación.

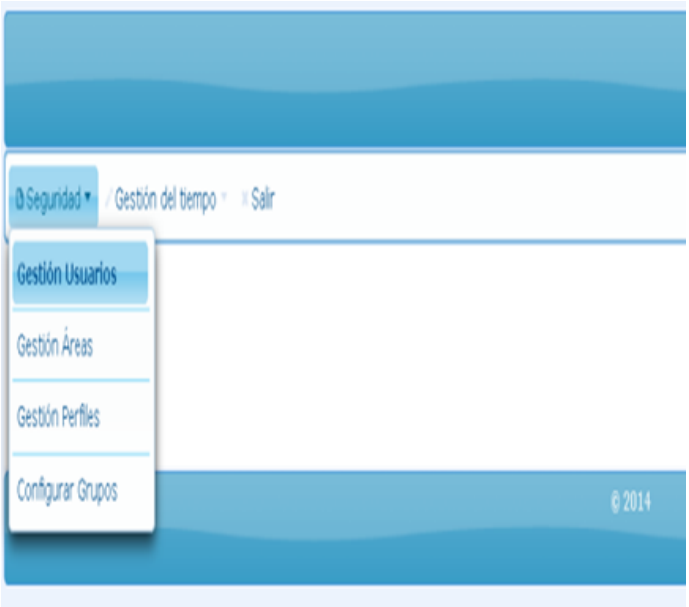

Figura A.3. Interfaz del módulo SAP ERP gestión de recursos humanos, con el sub-modulo de seguridad desplegado.

Estando sobre el submódulo seguridad, se proporciona la posibilidad de seleccionar una de cuatro opciones o pestañas:

- Gestión de Usuarios
  Esta opción permite crear, editar, consultar y eliminar un usuario del sistema.
- Gestión de Áreas
  Esta opción proporciona la creación, edición y actualización de las áreas que conforman una empresa.
- Gestión de perfiles
  Esta opción permite crear, editar, eliminar y consultar los distintos roles que se utilizan para la aplicación.
- Configurar Grupos.
  En esta opción se le permite al administrador agregar un usuario del sistema a las distintas áreas del sistema.

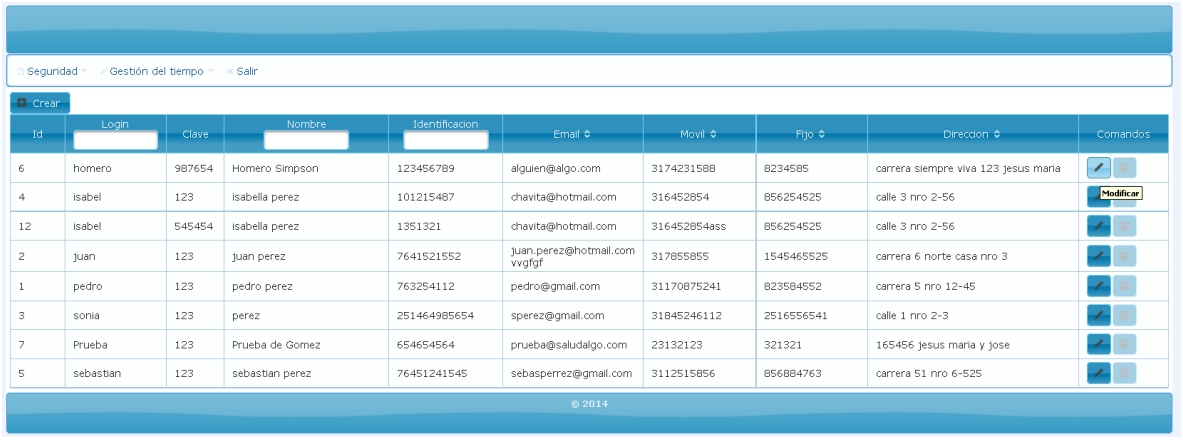

Figura A.4. Pestaña Gestión de Usuarios.

Al seleccionar el botón Crear ubicado en la parte superior izquierda de la pantalla, el sistema abre una ventana emergente, que posibilita la creación de un perfil para un usuario que accederá luego de ello al sistema. Los campos en la ventana identificados con el asterisco son obligatorios para avanzar con el proceso.

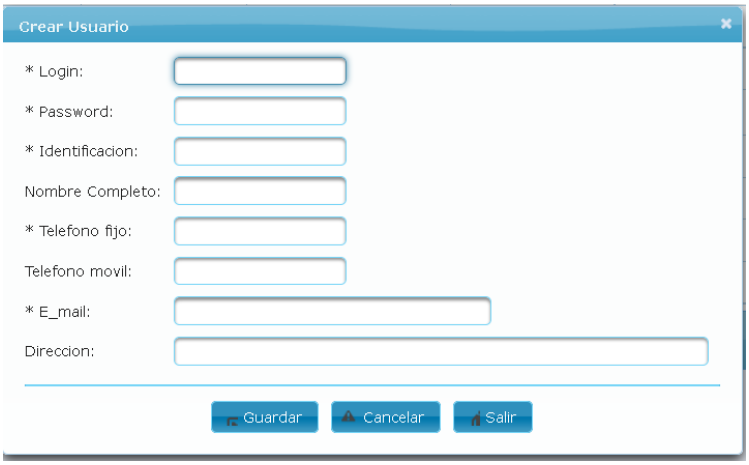

Figura A.5. Ventana emergente Crear Usuario.

El botón que posee una etiqueta con la forma de un lápiz, representa la acción de modificar un usuario, permitiendo actualizar cualquier dato de un usuario del sistema.

Figura A.6. Ventana emergente Modificar Usuario

Al seleccionar el botón Crear ubicado en la parte superior izquierda de la pantalla, el sistema abre una ventana emergente, que posibilita la creación de área que hace parte del núcleo de la empresa. El botón que posee una etiqueta con la forma de un lápiz, representa la acción de modificar un área, permitiendo actualizar cualquier dato de esta.

Figura A.7. Pestaña Gestión de Áreas.

Figura A.8. Pestaña Gestión de perfiles.

Al seleccionar el botón Crear ubicado en la parte superior izquierda de la pantalla, el sistema abre una ventana emergente, que posibilita la creación de un rol o perfil para acceder a la aplicación.

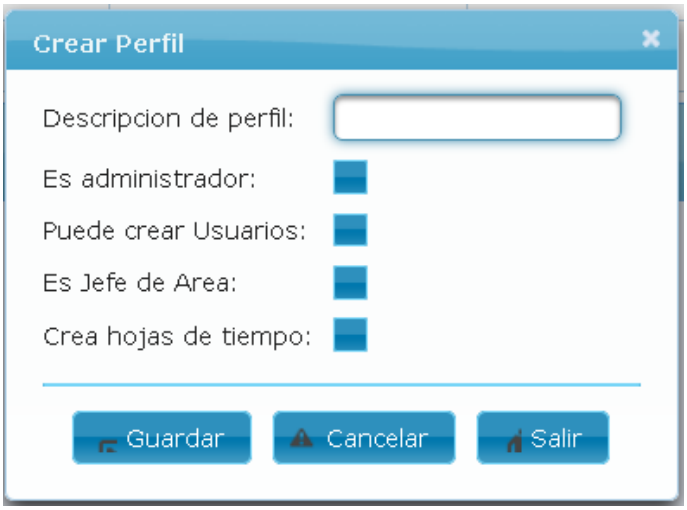

Figura A.9. Ventana emergente Crear Perfil.

El botón que posee una etiqueta con la forma de un lápiz, representa la acción de modificar un perfil, permitiendo actualizar cualquier dato de esta.

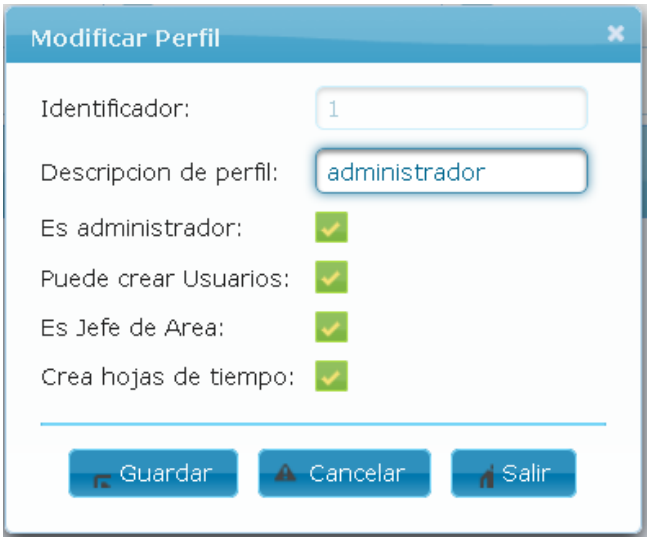

Figura A.10. Ventana emergente Modificar Perfil.

Pestaña configurar grupo, donde se observan todos los usuarios del sistema.

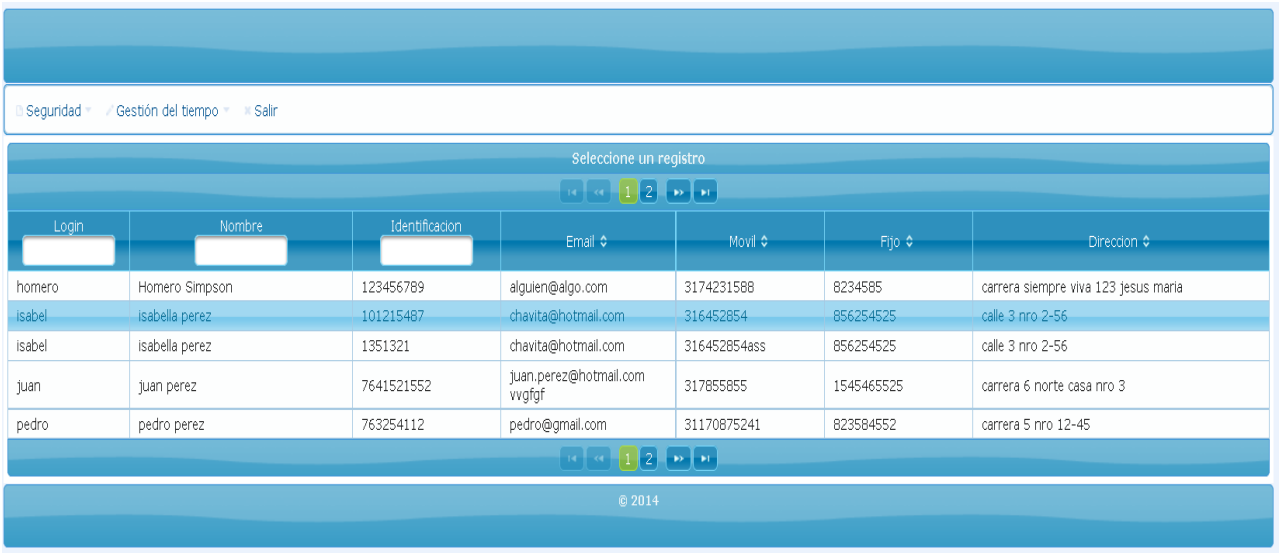

Figura A.11. Pestaña Configurar Grupos.

En la interfaz de Configurar grupo, se selecciona el usuario del sistema y se aprecia la información relevante de este.

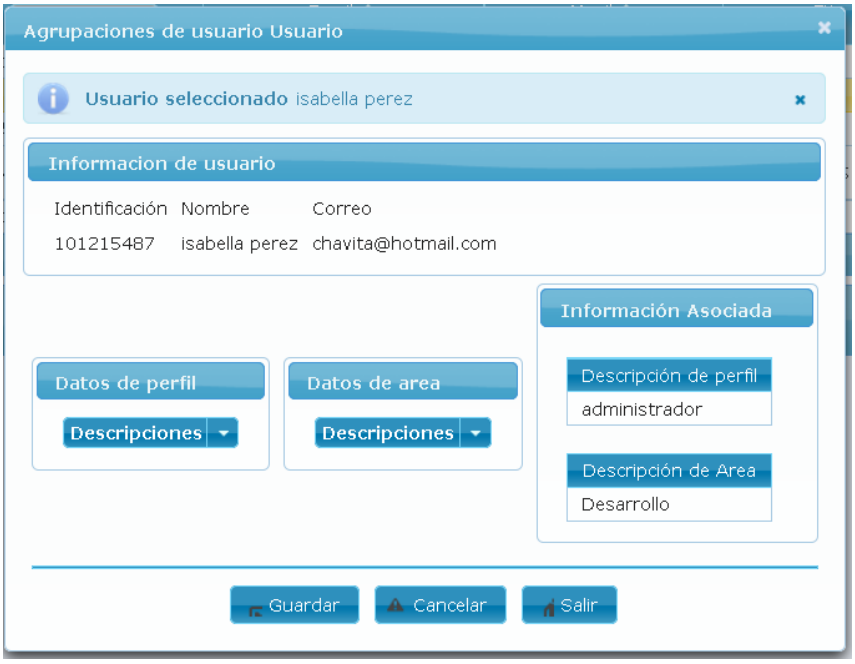

Figura A.12. Información de usuario que pertenece a un grupo.

Después se dirige al módulo de Gestión de tiempo, en el cual se despliegan las siguientes pestañas:

Figura A.13. Interfaz del módulo SAP ERP gestión de recursos humanos, con el sub-modulo de Gestión de tiempo desplegado.

- Administración de Permisos
  Esta opción permite crear, editar, consultar y eliminar los tipos de permisos que una persona de la empresa puede tomar, además indicando si requiere de la firma del jefe superior o no.

- Administrar plantillas de tiempo
  Esta opción proporciona la creación, edición y selección o búsqueda de información de las plantillas de tiempo, que están representadas en periodos, que pueden abarcar desde semanas hasta años.
- Administrar proyectos
  Esta opción permite crear, editar, eliminar y consultar los distintos proyectos en los cuales puede estar asignado el personal de una empresa, diviendo el tiempo de asignación a los mismos.
- Administrar tipos de actividad
  Esta opción permite crear, editar, eliminar y consultar las actividades que lleva a cabo cada empleado dentro de cada área de la empresa, específicamente para cada proyecto o actividad especial.
- Detalle del registro
  Esta opción permite verificar las horas detalladas por registro dentro de las hojas de tiempo, teniendo como base el calendario anual.
- Gestión del tiempo – Usuario
  Este permite, observar la información de manera dinámica del empleado, revisando sus tiempos por día y las asignaciones realizadas.
- Registro de Actividades
  Permite el registro de las actividades por proyecto, área y empleado.

Figura A.14. Pestaña Administración de Permisos.

Al seleccionar el botón Crear ubicado en la parte superior izquierda de la pantalla, el sistema abre una ventana emergente, que posibilita la creación de un tipo de permiso que es otorgado en la empresa, permitiendo además la selección entre el requerimiento de la firma del jefe inmediato o no.

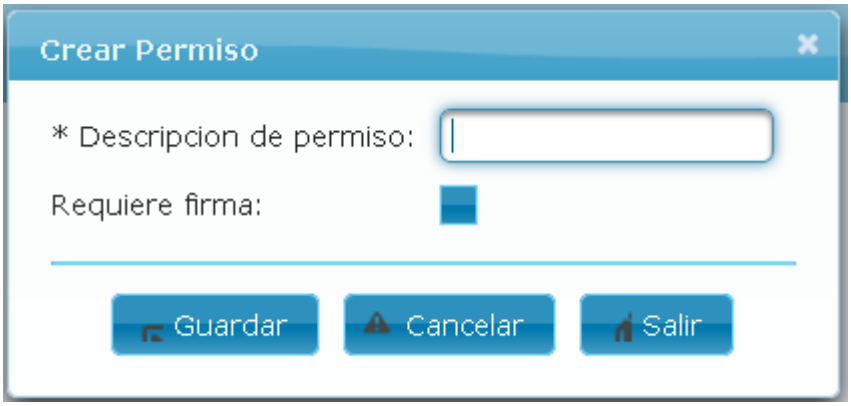

Figura A.15. Ventana emergente Crear permiso.

El botón que posee una etiqueta con la forma de un lápiz, representa la acción de modificar un permiso, permitiendo actualizar cualquier dato de esta.

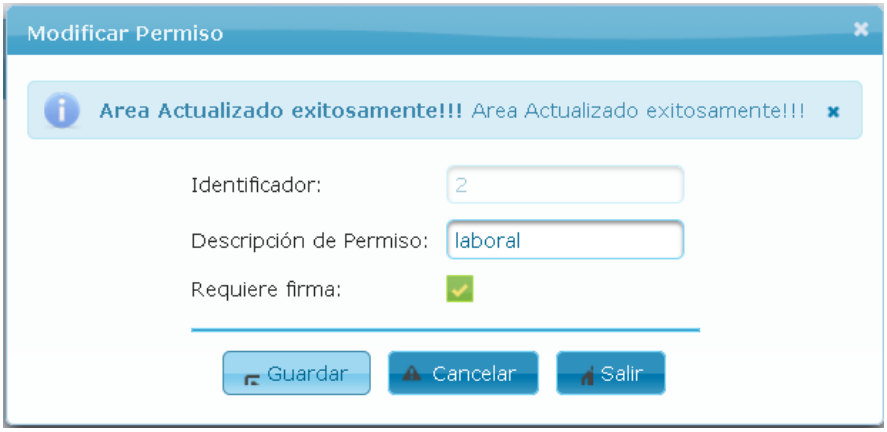

Figura A.16. Ventana emergente Modificar permiso.

Al seleccionar el botón Crear ubicado en la parte superior izquierda de la pantalla, el sistema abre una ventana emergente, que posibilita la creación de una plantilla de tiempo, que puede abarcar desde semana hasta años.

Figura A.17. Pestaña Administración plantillas de tiempo y ventana emergente crear hoja de tiempo.

Al dar clic sobre el ícono de la derecha que contiene la imagen de un almanaque, se despliega un calendario que permite la selección de la fecha de inicio y el fin de la hoja de tiempo.

El botón que posee una etiqueta con la forma de un lápiz, representa la acción de modificar una plantilla de tiempo, permitiendo actualizar cualquier dato de esta.

Figura A.18. Ventana emergente crear hoja de tiempo con el calendario desplegado.

Figura A.19. Ventana emergente Modificar hoja de tiempo.

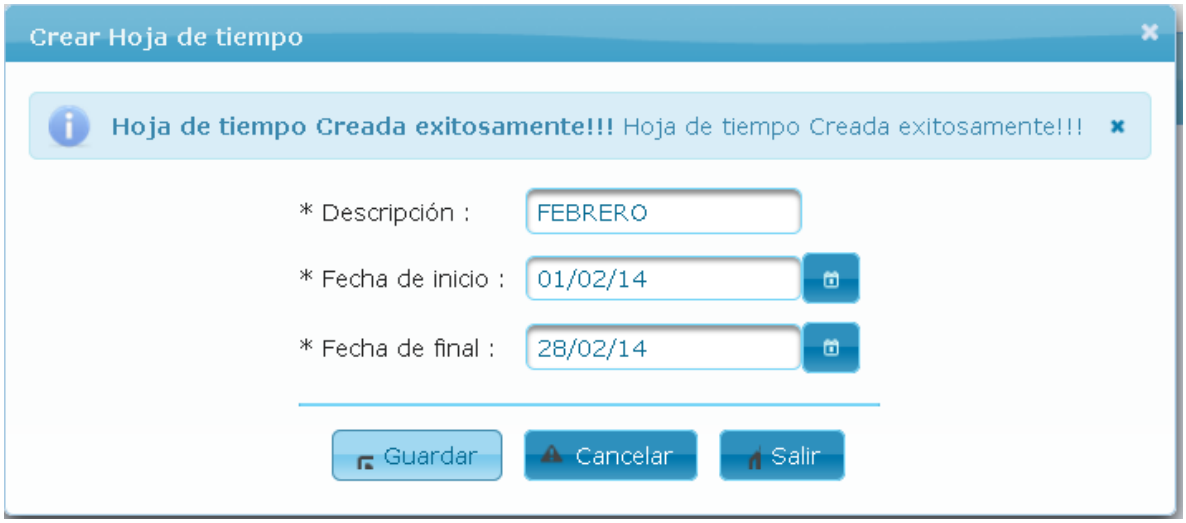

Figura A.20. Ventana emergente Crear Hoja de tiempo con mensaje de éxito.

Figura A.21. Pestaña Administrar proyectos.

Al seleccionar el botón Crear ubicado en la parte superior izquierda de la pantalla, el sistema abre una ventana emergente, que posibilita la creación de una proyecto que la empresa este realizando.

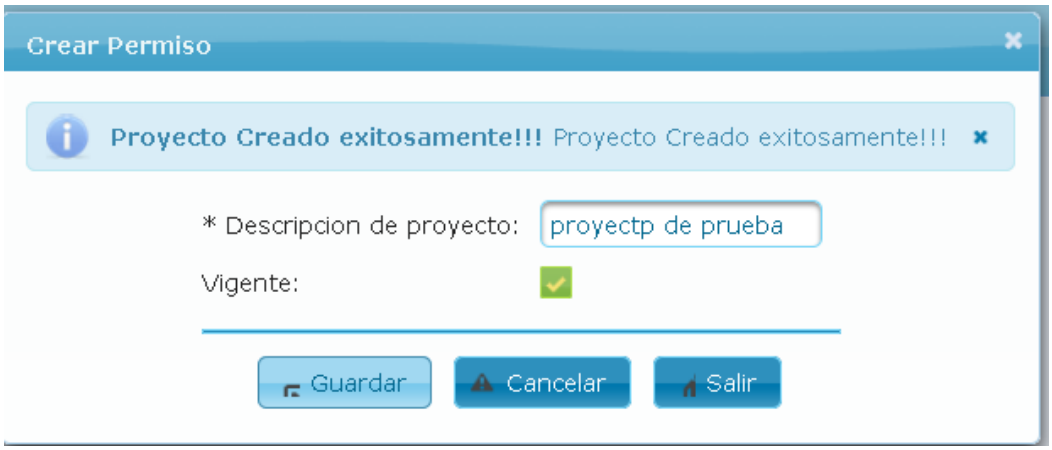

Figura A.22. Ventana emergente Crear proyecto.

El botón que posee una etiqueta con la forma de un lápiz, representa la acción de modificar un proyecto, permitiendo actualizar cualquier dato de esta.

Figura A.23. Ventana emergente Modificar proyecto.

Figura A.24. Pestaña Administrar tipo de actividad y ventana emergente crear tipo de actividad.

Al seleccionar el botón Crear ubicado en la parte superior izquierda de la pantalla, el sistema abre una ventana emergente, que posibilita la creación de una actividad destinada a un proyecto, dentro de la temática de la empresa.

El botón que posee una etiqueta con la forma de un lápiz, representa la acción de modificar una actividad, permitiendo actualizar cualquier dato de esta.

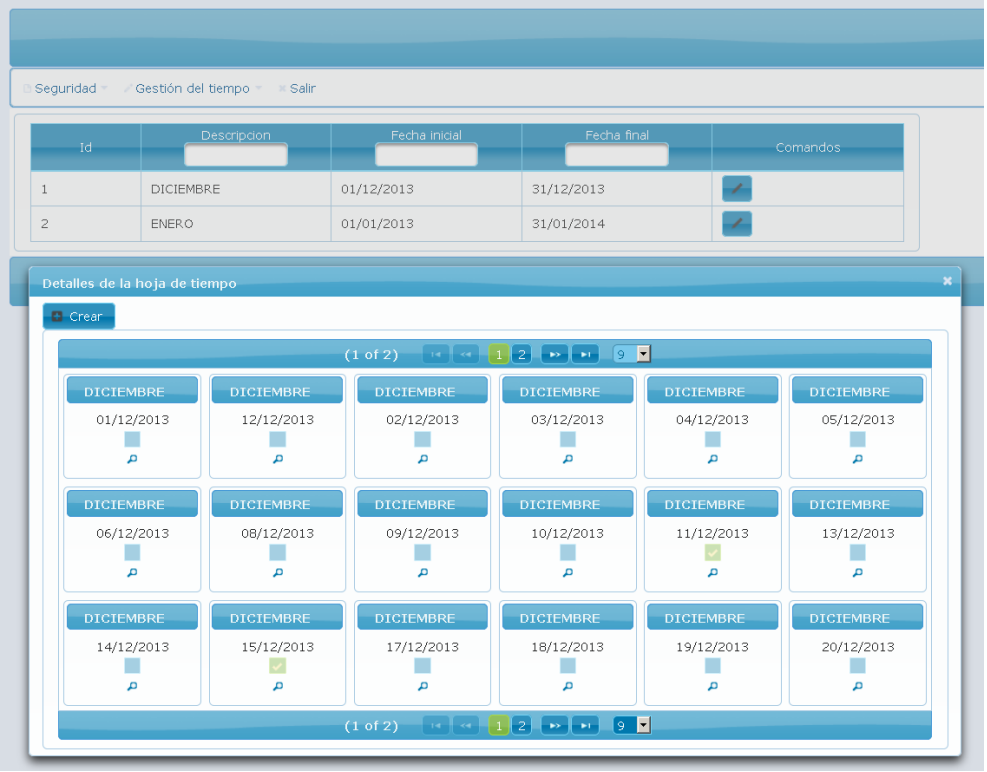

Figura A.25. Pestaña Detalle del registro.

Al seleccionar el botón Crear ubicado en la parte superior izquierda de la pantalla, el sistema abre una ventana emergente, que posibilita la creación de un detalle para el día seleccionado como por ejemplo, festivo entrega, etc.

Figura A.26. Ventana emergente detalle.

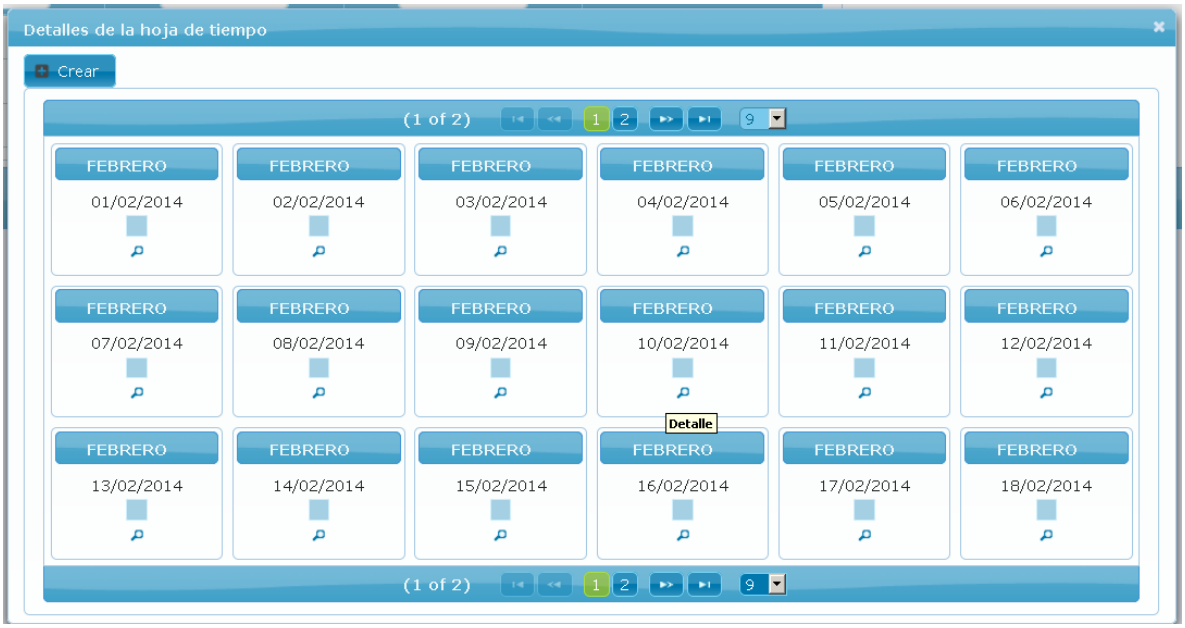

Figura A.27. Ventana emergente detalles de la hoja de tiempo.

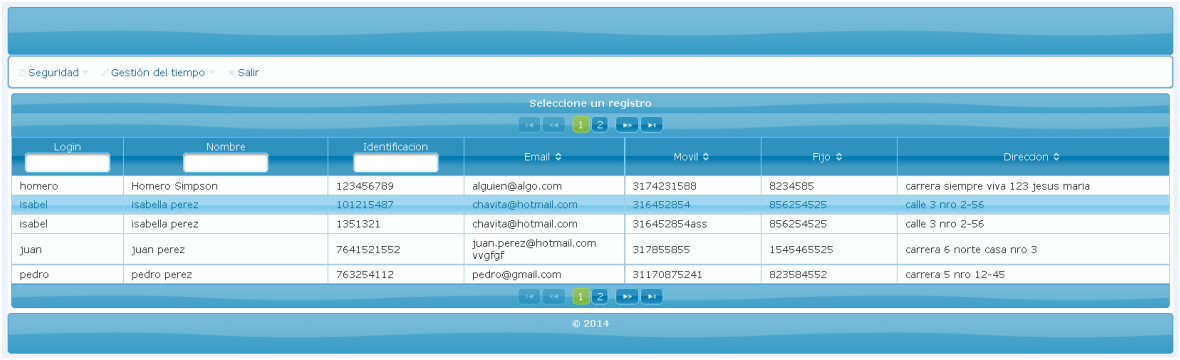

Figura A.28. Pestaña Gestión del tiempo.

Al seleccionar cualquiera de los usuarios el sistema abre una ventana emergente, que permite avistar toda la información de un usuario del sistema.

El botón que posee una etiqueta con la forma de un lápiz, representa la acción de modificar una actividad, permitiendo actualizar cualquier dato de esta.

Figura A.29. Ventana emergente Hojas de tiempo asignadas al usuario.

Figura A.30. Pestaña registro de actividades.

Al dar clic en la hoja de tiempo asignada al usuario se despliega la ventana emergente de registro de actividades, para poder realizar el proceso de inserción de la información del trabajador. Además se puede apreciar todas las actividades registradas para un usuario.

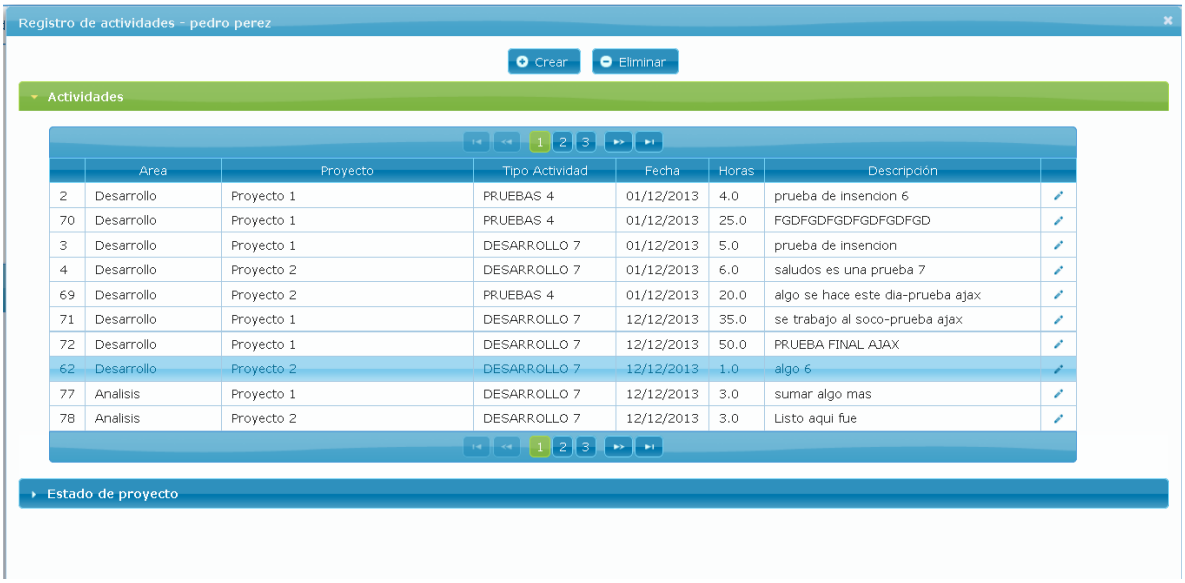

Figura A.31. Ventana emergente registro de actividades.

Al dar clic en el botón crear aparece una ventana emergente que permite agregar una nueva actividad.

Figura A.32. Ventana emergente actividad nueva.

Por ultimo al seleccionar la opción estado del proyecto se puede apreciar las horas que ha trabajado un empleado por proyecto.

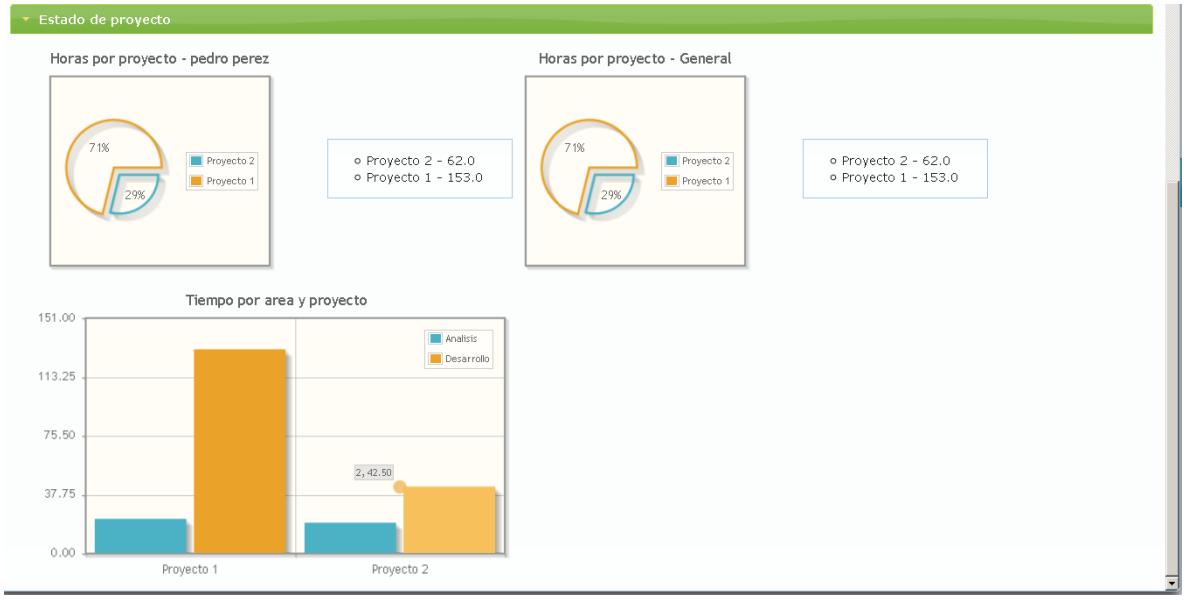

Figura A.33. Ventana emergente estado del proyecto.