Universidad Nacional Autónoma de México

Facultad de Ingeniería

Reporte de experiencia profesional

Que para obtener el título de Ingeniero en Computación presenta el alumno:

Angel Enrique González Sánchez

Asesor de tesis: M.C. Alejandro Velázquez Mena

Ciudad Universitaria Diciembre 2013

CONTENIDO

INTRODUCCIÓN

Este reporte describe las actividades profesionales realizadas por el Pasante Angel Enrique González Sánchez durante un periodo de cuatro años donde aplicó los conocimientos adquiridos en la carrera de Ingeniería en Computación en el entorno laboral.

Este proyecto consiste en la descripción de todos los desarrollos de software realizados para la empresa Cuallisys S.A. de C.V. desde junio de 2010 hasta marzo del 2013. También describe las actividades realizadas por el pasante en cada uno de los proyectos.

Se realiza una descripción breve de cada proyecto acompañado de los diagramas de contexto de cada uno, también se listan las tecnologías utilizadas y los casos de uso en los que se tuvo participación.

Se toma uno de los proyectos en el cual se haya tenido más participación a lo largo del todo el ciclo de desarrollo y en el que se haya tenido más responsabilidad al implementar la solución propuesta para llevar la descripción mas a profundidad, se describe el problema, se documenta brevemente la solución, se muestran los diagramas de contexto y los diagramas de clases, algunos diagramas de estados y se describen las actividades realizadas para el proyecto.

Se espera que este reporte demuestre que se cuenta con la experiencia necesaria para recibir la titulación mediante esta modalidad de reporte de experiencia profesional.

El objetivo del presente reporte es mostrar que la formación como Ingeniero en Computación que me otorgo la Facultad de Ingeniería de la Universidad Nacional Autónoma de México me permiten desenvolverme con total seguridad en el ámbito laboral.

La facultad de Ingeniería me doto de las herramientas necesarias para hacer frente a todos los retos con los que se encuentra uno en la vida laboral, me enseño a ser analítico, dedicado, estudioso y proactivo.

Todo el plan de estudios está conformado de tal forma que al egresar de la universidad contemos con todas las bases para entender los problemas del mundo real, si bien es cierto que uno quisiera una enseñanza mas dirigida hacia las herramientas y soluciones que se están usando e implementando en el mundo entero, con la información que nos provee la Facultad estamos preparados y con los fundamentos suficientes para aprender nuevas herramientas o proponer nuevas soluciones.

En este reporte doy detalle de todas las actividades que he realizado profesionalmente desde que me encuentro laborando en Cuallisys S.A. de C.V., también doy un pequeño resumen de en donde se encuentra posicionada la empresa en la que laboro y doy una ligera introducción de la metodología de desarrollo que he estado utilizando.

Espero que con todo esto sea suficiente para demostrar que la Universidad Nacional Autónoma de México tiene en mí un orgulloso egresado de sus aulas y espero devolver algún día algo de lo mucho que me entrego la Universidad.

CAPÍTULO 1. ORGANIGRAMA

Laboro en Cuallisys S.A. de C.V. desde Junio del 2010, anteriormente trabaje en Softtek S.C., Televisa S.A. de C.V. y el Centro Nacional de Prevención de Desastres, en todos me he dedicado al desarrollo de sistemas.

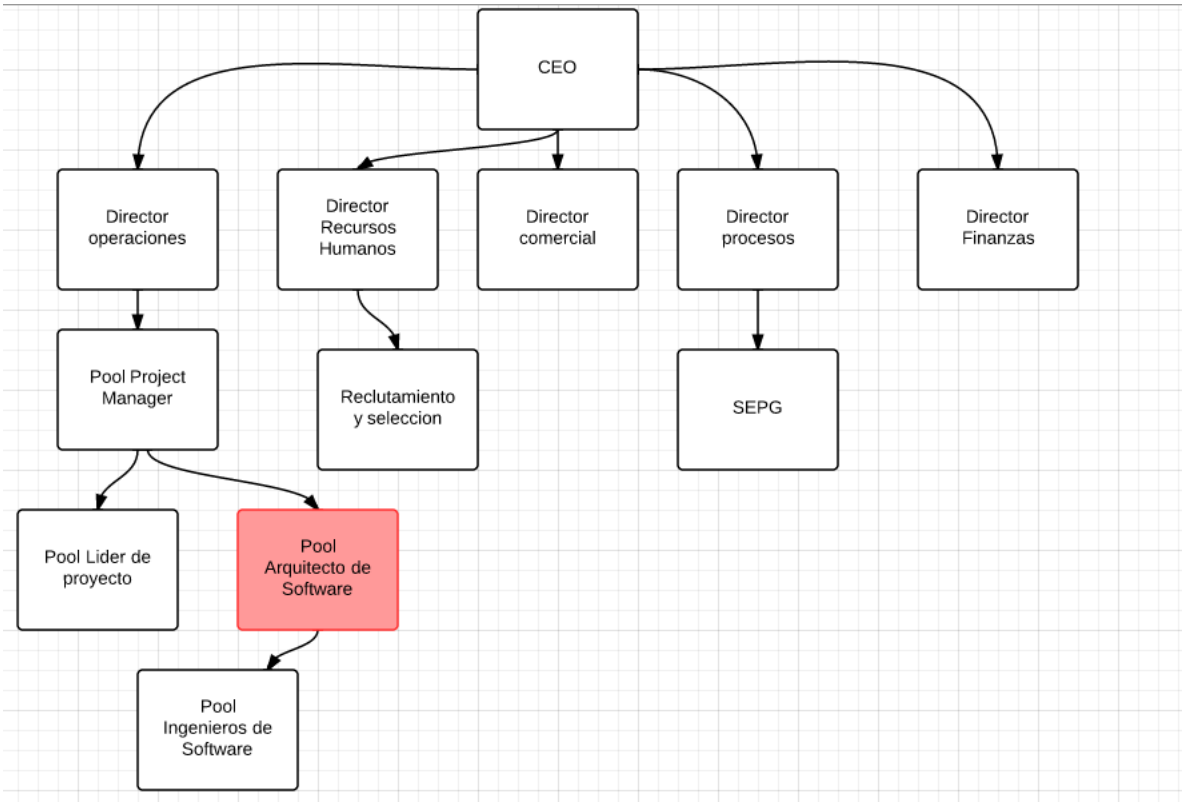
Cuallisys es una empresa 100% mexicana constituida en 2010 que se dedica al desarrollo de software a la medida. Para desarrollar con calidad mundial emplean las metodologías del Personal Software Process y del Team Software Process desarrolladas en la Universidad Carnegie Mellon (en Pittsburgh, Pennsylvania, EU) por Watts S. Humphrey, lo que permite incrementar la madurez, no solamente de los procesos de desarrollo, sino incluir el mejoramiento del elemento básico que da sustento a la empresa: las personas.

Gracias a que llevan el registro de sus métricas (tiempo, tamaño y defectos) han podido generar una base de datos histórica que les permite realizar estimaciones de proyectos con una desviación al plan de más menos 15% y tener una densidad de defectos de 0.36 defectos por cada mil líneas de código.

En Cuallisys me desempeño como Arquitecto de Software, puesto en el cual he logrado poner a prueba todas mis capacidades permitiéndome resolver los diferentes problemas que he afrontado, he participado en varios proyectos y he utilizado diferentes tecnologías en cada uno de ellos.

He estado a cargo de diferentes roles TSP en todos mis equipos de desarrollo así como también he estado involucrado en diferentes fases de los proyectos.

Dentro del organigrama de la empresa los arquitectos son los líderes técnicos del proyecto por lo que le reportan directamente junto con los líderes administrativos al gerente de proyectos, figura 1.



Capítulo 1. Figura 1.

CAPÍTULO 2. DESCRIPCIÓN DE PROYECTOS

Hoy en día muchas empresas están conscientes de sus necesidades en cuanto a desarrollo de software, saben que el contar con una herramienta de software que les ayude a hacer más eficiente su trabajo puede ser la ventaja competitiva frente a sus rivales de negocio, es por esto que se acercan al software comercial buscando alguno que cubra la mayoría de sus necesidades, pero muchas veces debido a sus reglas de negocio tan especificas o flujos de trabajo tan singulares dicho software comercial les queda muy corto en sus pretensiones, en estos casos la mejor solución es construir un sistema de software a la medida.

El software a la medida es un producto de Ingeniería de software desarrollado específicamente para un cliente en particular, teniendo en cuenta sus reglas de negocio y particularidades en sus flujos de trabajo.

Cuallisys como empresa consultora en tecnologías de la información se especializa en ofrecer este tipo de soluciones al sector empresarial, teniendo entre sus clientes a bancos, aseguradoras y dependencias del gobierno.

Dentro de Cuallisys he participado en diferentes proyectos, en cada uno he aprendido diferentes tecnologías y he desempeñado diferentes roles, en todos he demostrado que tengo las capacidades necesarias para cumplir con lo que se me ha asignado.

Cuallisys utiliza una metodología para el desarrollo de software llamada Personal Software Process, dicha metodología evoluciona en el desarrollo en equipo llamándose Team Software Process, dicha metodología nos compromete como desarrolladores a cumplir con ciertos roles dentro del equipo.

Todas las tecnologías implican una curva de aprendizaje, es por eso que las asignaciones en los proyectos van aumentando en complejidad, esto se puede ver conforme se van mostrando los proyectos en los que he participado.

A continuación voy a explicar los proyectos en los que he participado, un diagrama de contexto, las tecnologías utilizadas y mi participación dentro del mismo, al final voy a describir un proyecto mas a profundidad ya que es el primero en el que juego un rol de arquitecto y eso me permitió tener una visión global de todo el proyecto.

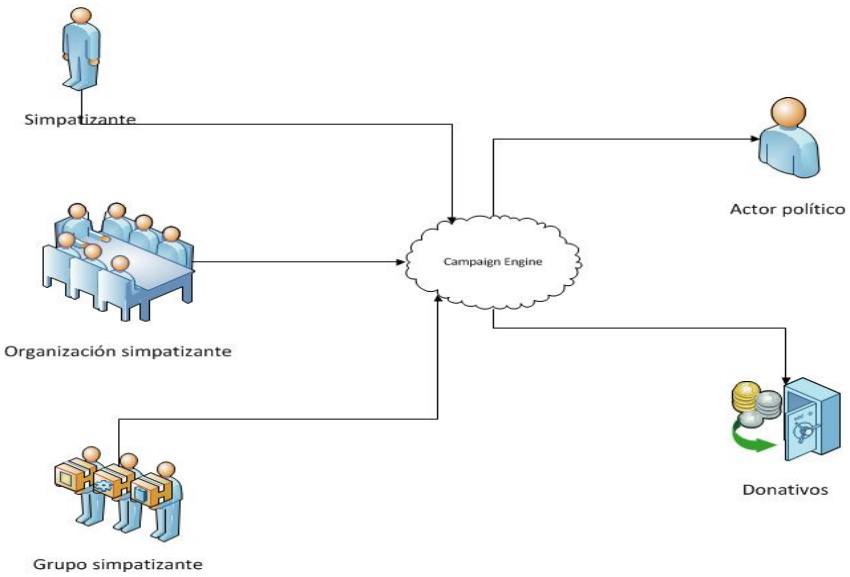
También en la descripción de cada proyecto incluyo el rol TSP asignado y las tareas realizadas como parte de ese rol.

Proyecto Campaign Engine para Media Mezcla LLC

La empresa Media Mezcla LLC, con sede en Estados Unidos y dueña de plataformas para la administración de campañas políticas, solicitó a Cuallisys el desarrollo de funcionalidad extra para una plataforma que sirve como medio de comunicación entre un candidato político y sus votantes, dicha plataforma es Campaign Engine.

El sistema de Campaign Engine es una plataforma dirigida a los diferentes actores políticos que deseen tener un acercamiento con sus simpatizantes. Figura 2. El objetivo del sistema es servir como un canal de comunicación entre el candidato político y sus votantes, entre las acciones que permite realizar se encuentran:

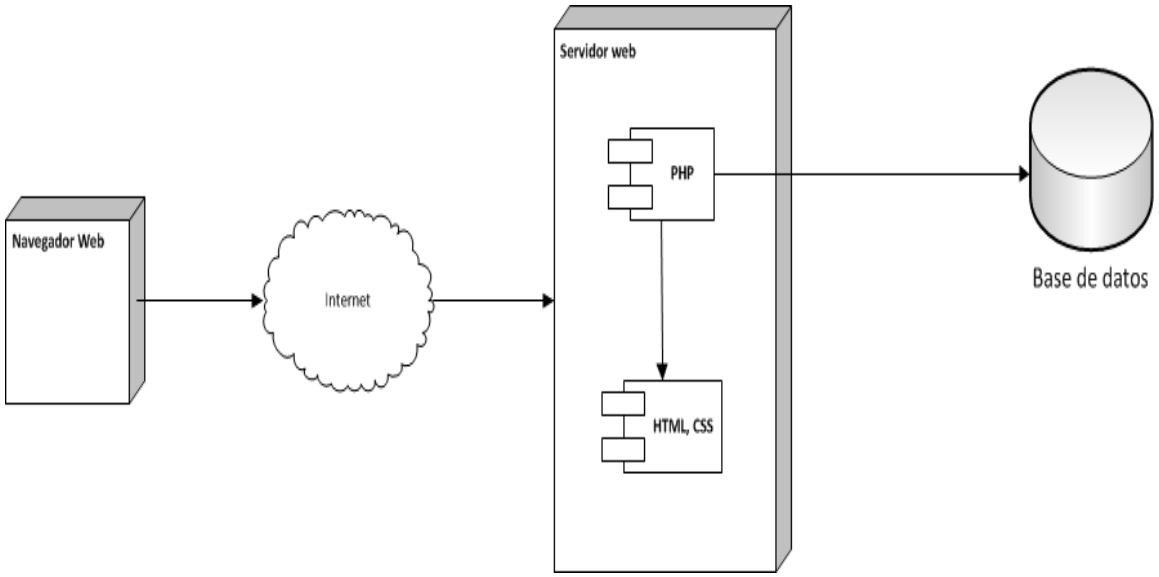
- Realizar contribuciones en línea.
- Estar enterado de eventos.
- Enviar email a todos los suscriptores.
- Facilidad en la creación y edición del contenido.
- Administración de los voluntarios.
- Obtener reportes del tráfico del sitio, envío de correos y recaudación de fondos.
- Total control de los datos almacenados.
- Interfaz fácil de usar.



Capítulo 2. Figura 2.

Por requerimientos del cliente este proyecto debía de cumplir con una arquitectura LAMP (Linux, Apache, MySQL, PHP), la parte visual de la aplicación se creó utilizando HTML, hojas de estilo CSS y JavaScript, utilizando la librería JQuery para hacer más dinámica la parte web agregando algunas validaciones en los

formularios y la carga de algunas imágenes del sitio. por lo cual la arquitectura propuesta es la mostrada en la figura 3:



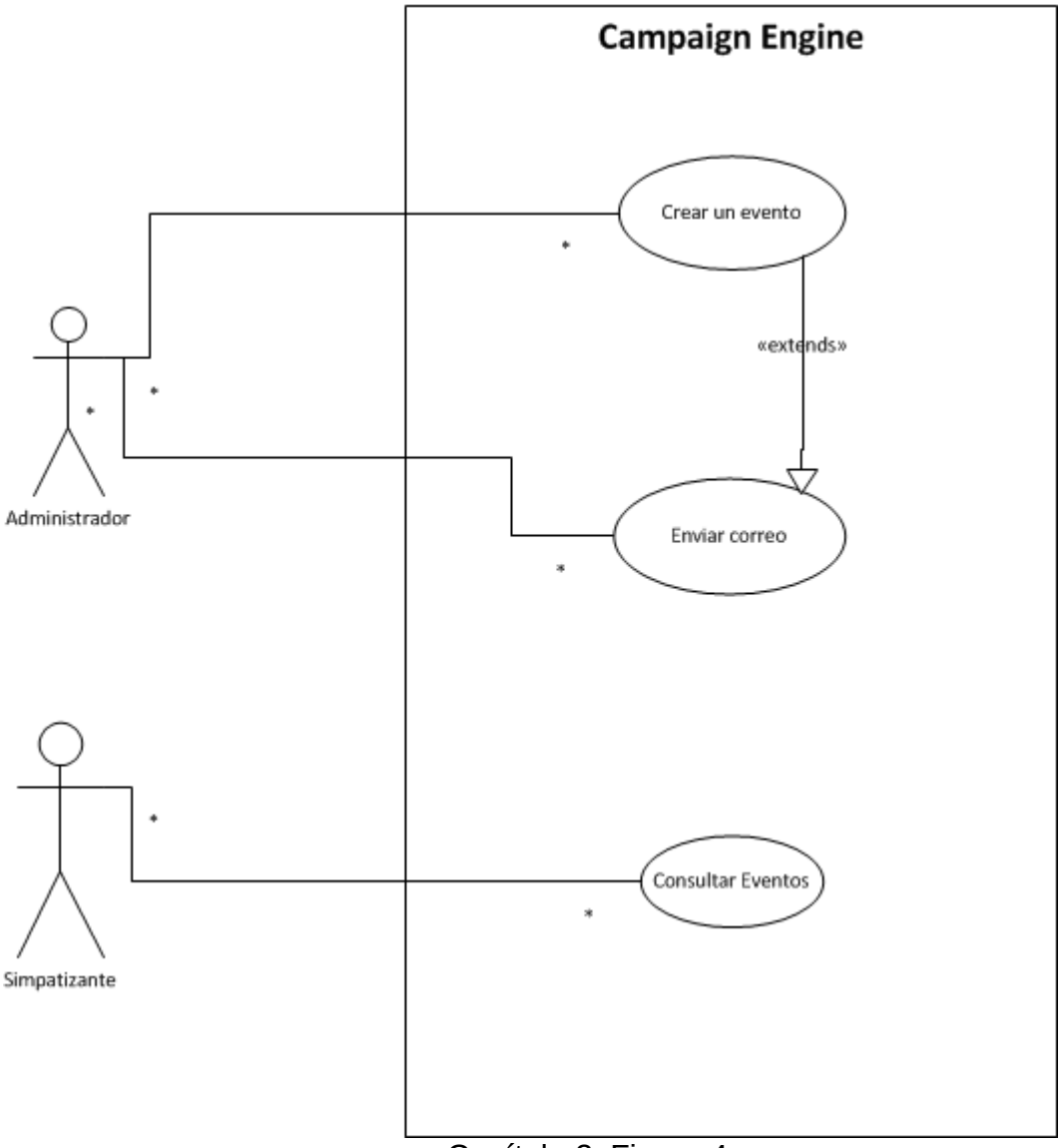
Capítulo 2. Figura 3.

Tecnología utilizada

- Lenguaje de programación PHP
- Manejador de base de datos MySQL
- Servidor Web Apache

Actividades realizadas

En este proyecto tuve asignado el módulo de eventos el cual abarcaba los casos de uso mostrados en la figura 4.



Capítulo 2. Figura 4.

La funcionalidad del módulo abarcaba la administración de eventos por parte del administrador así como la consulta por parte de los simpatizantes, también incluía el envío de correos a los interesados con la información del evento.

El desarrollo consistió en crear el módulo en todas las capas, es decir se modelaron los eventos en la base de datos, se crearon las vistas correspondientes a la administración y la consulta de los eventos, se crearon las clases encargadas de procesar las peticiones relacionadas así como toda la lógica necesaria para el envío de correos.

Aunque éste fue mi primer proyecto en la empresa no era mi primer proyecto con esa tecnología, anteriormente cuando estuve trabajando en Televisa ya había desarrollado con una arquitectura LAMP (Linux, Apache, MySQL, PHP), por tal motivo la curva de aprendizaje fue corta y me enfoque más en entender la lógica de lo que ya se tenia del proyecto.

Al ser mi primer proyecto como miembro de un equipo TSP no tuve rol asignado, con lo cual solo me limite a cumplir con mis reportes, registrar mis métricas e informarlas puntualmente al rol de calidad y planeación del equipo.

Cuando entré a este proyecto ya tenían algunos meses trabajando en el por lo cual el principal problema fue el entender el trabajo que ya se llevaba y el acoplarme a utilizar la metodología de la empresa, pero con el apoyo del equipo eso no fue problema, logre cumplir con mis asignaciones, aunque no me quedé a la culminación del proyecto debido a una nueva asignación.

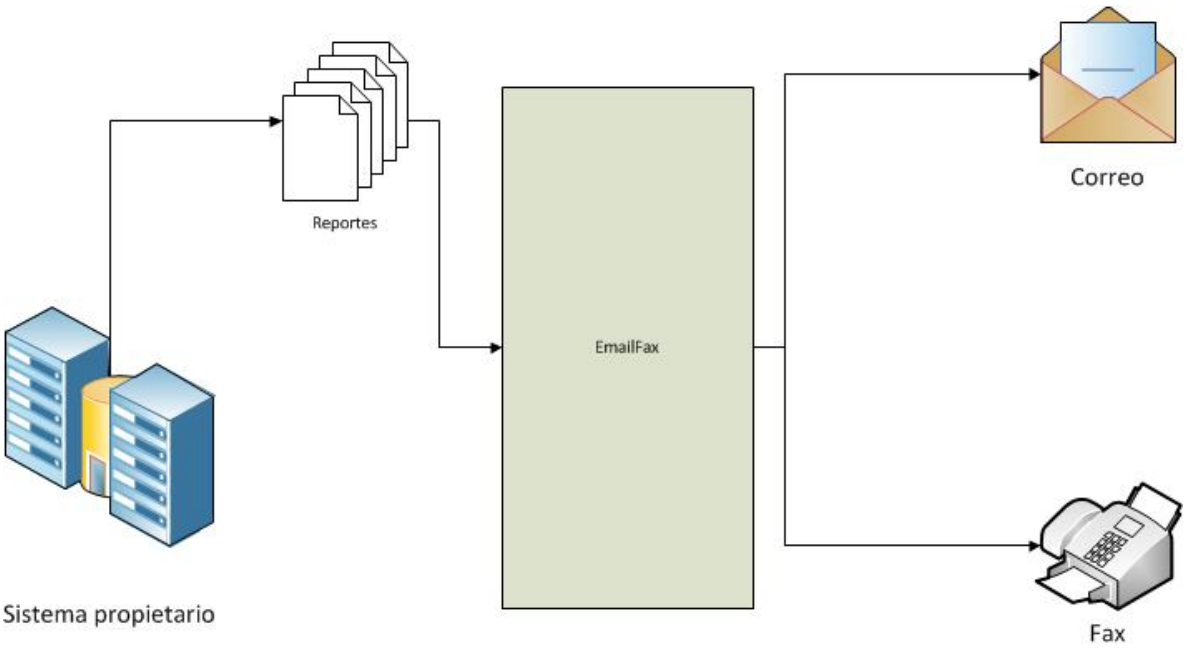
Proyecto para el envío de correos y fax para la empresa G&D

La empresa Giesecke & Devrient (G&D), empresa ubicada en Ontario Canada y dedicada a servicios financieros y de infraestructura solicitó a Cuallisys el desarrollo de un componente que envíe por correo y por fax unos reportes generados por su sistema propietario.

Entre los servicios de la empresa G&D están los servicios bancarios, en su flujo de negocios ellos generan reportes, lo que buscan es un componente de software que al momento de generar un reporte lo mande por correo electrónico y por fax. Figura 5.

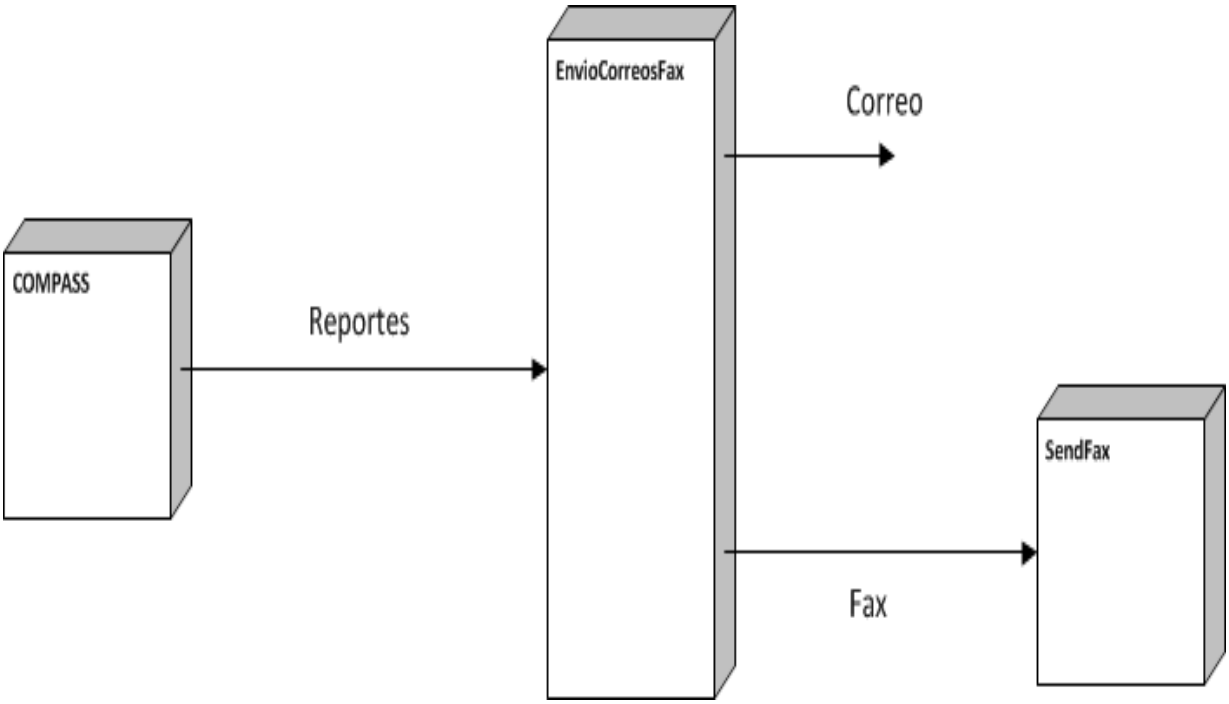
Los requerimientos que debe de cumplir el sistema son:

- Enviar correo de forma automática al momento de generar el reporte.
- Enviar fax de forma automática al momento de generar el reporte.



Capítulo 2. Figura 5.

Debido a la plataforma utilizada por el cliente, la tecnología de este proyecto fue Visual Basic 6 y un programa llamado SendFax el cual es la interfaz con el fax, dicho programa de SendFax corría como un programa ejecutable de Windows y lo único que necesitaba de nuestro sistema era que copiáramos el reporte generado en una carpeta específica del sistema. La arquitectura propuesta fue la siguiente (figura 6):



Capítulo 2. Figura 6

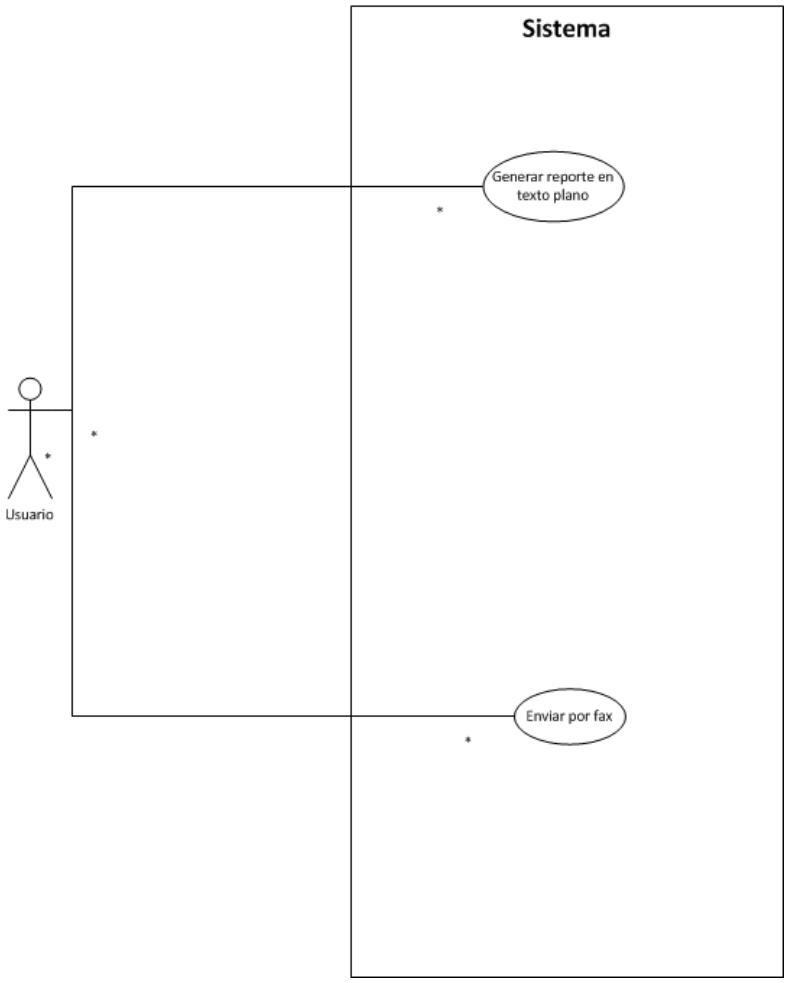
Tecnología utilizada

- Visual Basic 6.

Actividades realizadas

En este proyecto tuve asignado la creación del reporte en un formato de texto plano, para que pudiera ser enviado por correo o por fax, así como la

comunicación del sistema con el programa SendFax para el envío del reporte por fax, tal como se puede ver en la figura 7.



Capítulo 2. Figura 7

Mis actividades consistieron en recuperar el reporte desde el sistema Compass para después ir leyendo los resultados y generar un archivo en texto plano con la información del reporte y en un formato legible.

Ya con el reporte en texto plano lo siguiente fue realizar la comunicación con el sistema SendFax de acuerdo a lo indicado en su documentación.

En este proyecto tuve mi primer rol TSP el cual fue el rol de pruebas, dentro de las tareas de dicho rol me dedique a verificar que se cumplieran los casos de pruebas de cada componente y que se registraran correctamente los resultados generados.

Este proyecto se realizó con un cliente extranjero, razón por la cual una de las dificultades del mismo fue la barrera del idioma ya que en la fase de toma de requerimientos algunas cosas no quedaban claras al momento de expresarlas. Otro problema también fue la poca documentación que tenían de su componente desarrollado SendFax y de las interfaces que tenia.

Participe en este proyecto de principio a fin, logrando así tener mi primer proyecto exitoso en la empresa ya que el anterior solo participe brevemente.

Proyecto para la Secretaría de Marina

La Secretaria de Marina solicitó a Cuallisys el desarrollo de un sistema que incorpore la funcionalidad de todos los sistemas con los que cuentan en este momento y que les sirva como herramienta para optimizar su trabajo.

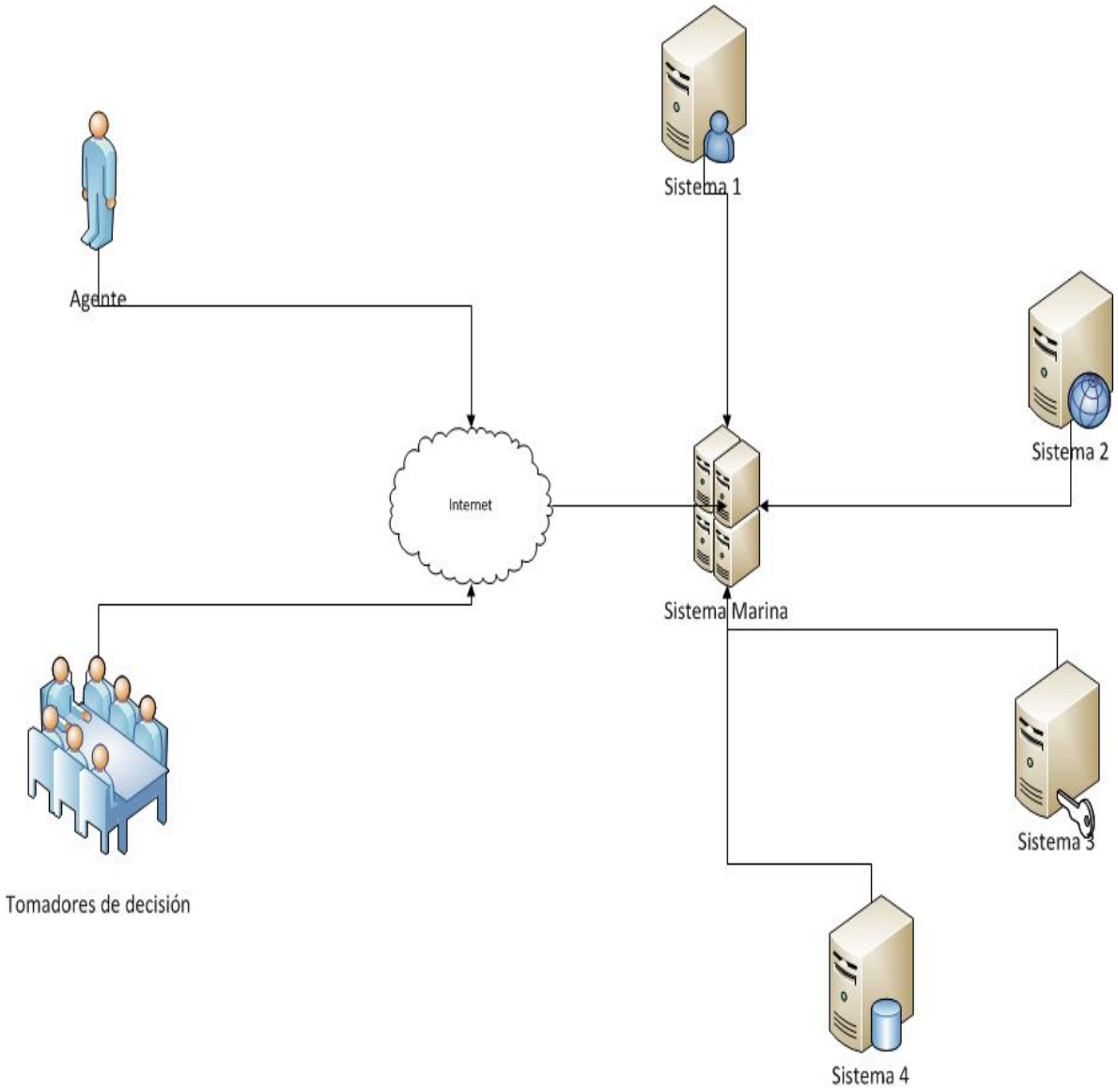
Los sistemas actuales de la SEMAR trabajan de forma separada lo que ocasiona que exista duplicidad en la información o que no se tenga la información actualizada en todos los sistemas, además que el consultar la información es un proceso muy tardado debido a que se tiene que pedir en cada una de las plataformas con las que se cuenta.

Lo que se busca con este proyecto es agrupar toda la información en una sola base de datos y conjuntar toda la funcionalidad de los sistemas con los que se cuenta actualmente para hacer más eficiente la consulta de información y dotar a los analistas de mejores herramientas para desarrollar su trabajo.

Otra de las ventajas que se obtienen al centralizar la información es el generar reportes que ayuden a los altos mandos a mejores tomas de decisiones.

Las acciones que debe realizar el sistema son:

- Consolidación de la información de todos los sistemas que actualmente utilizan en la SEMAR.
- Búsqueda de información en diversas fuentes.
- Generación de reportes.
- Administración de información.



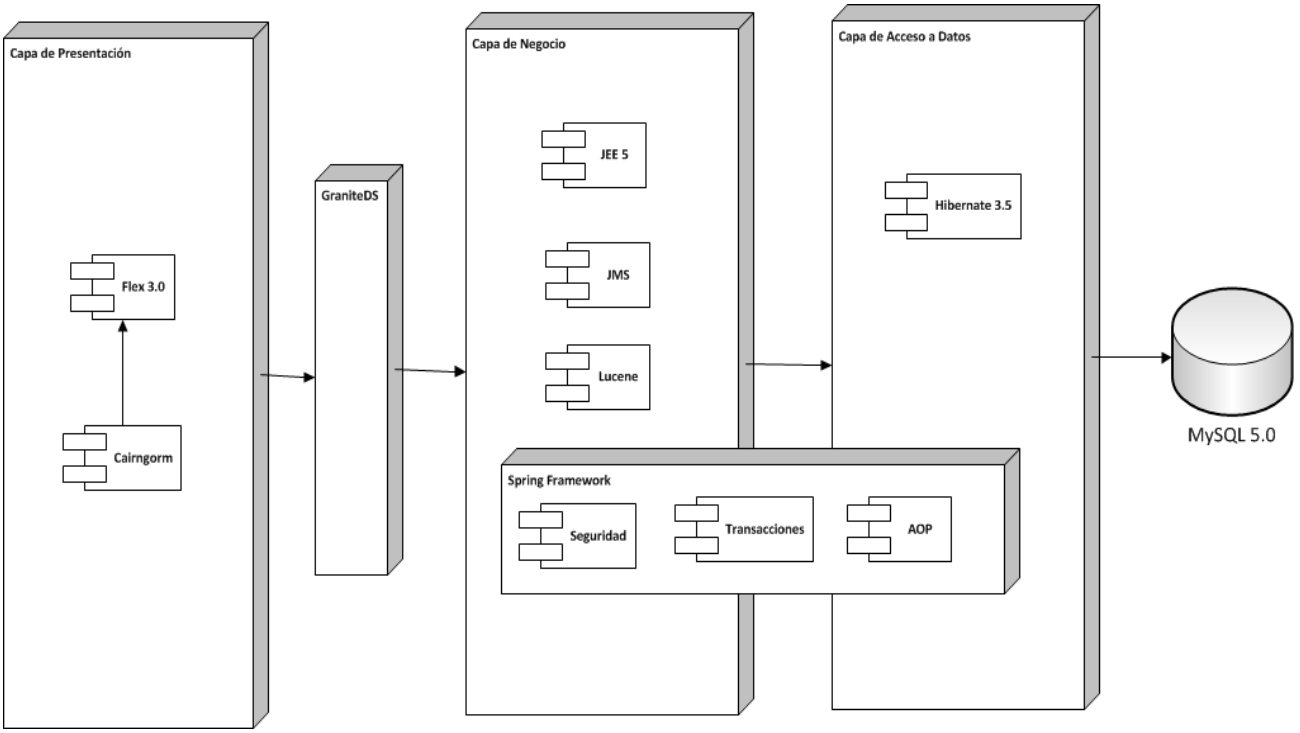
Capítulo 2. Figura 8

Durante la etapa de preventa del proyecto se le mostro al cliente un pequeño demo utilizando la tecnología Flex de Adobe para crear la interfaz de usuario del sistema, razón por la cual dicha tecnología formo parte de los requerimientos de la aplicación, para la parte del backend se opto por tecnología JEE por su robustez en seguridad, las librerías de integración con Flex y las librerías open source que existían para generar la solución propuesta.

Como marco de desarrollo se utilizó Spring con sus modulo score y security, se utilizo Hibernate como herramienta de Mapeo Objeto-Relacional además de un proyecto de hibérnate llamado Hibernate Search el cual permite realizar búsquedas del tipo Full Text sobre las entidades persistidas en el sistema, todo esto utilizando un proyecto de Apache llamado Lucene el cual permite analizar dichos índices en las entidades.

Como herramienta para permitir la comunicación entre los servicios expuestos en Java y los servicios en Action Script de Flex se utilizó GraniteDS, la cual toma los objetos de Java y genera sus contrapartes en Action Script de tal forma que pareciera que se trabaja en el mismo nivel de dominio en ambos lados de la aplicación.

La arquitectura propuesta para este proyecto fue la siguiente:



Capítulo 2. Figura 9

Tecnología utilizada

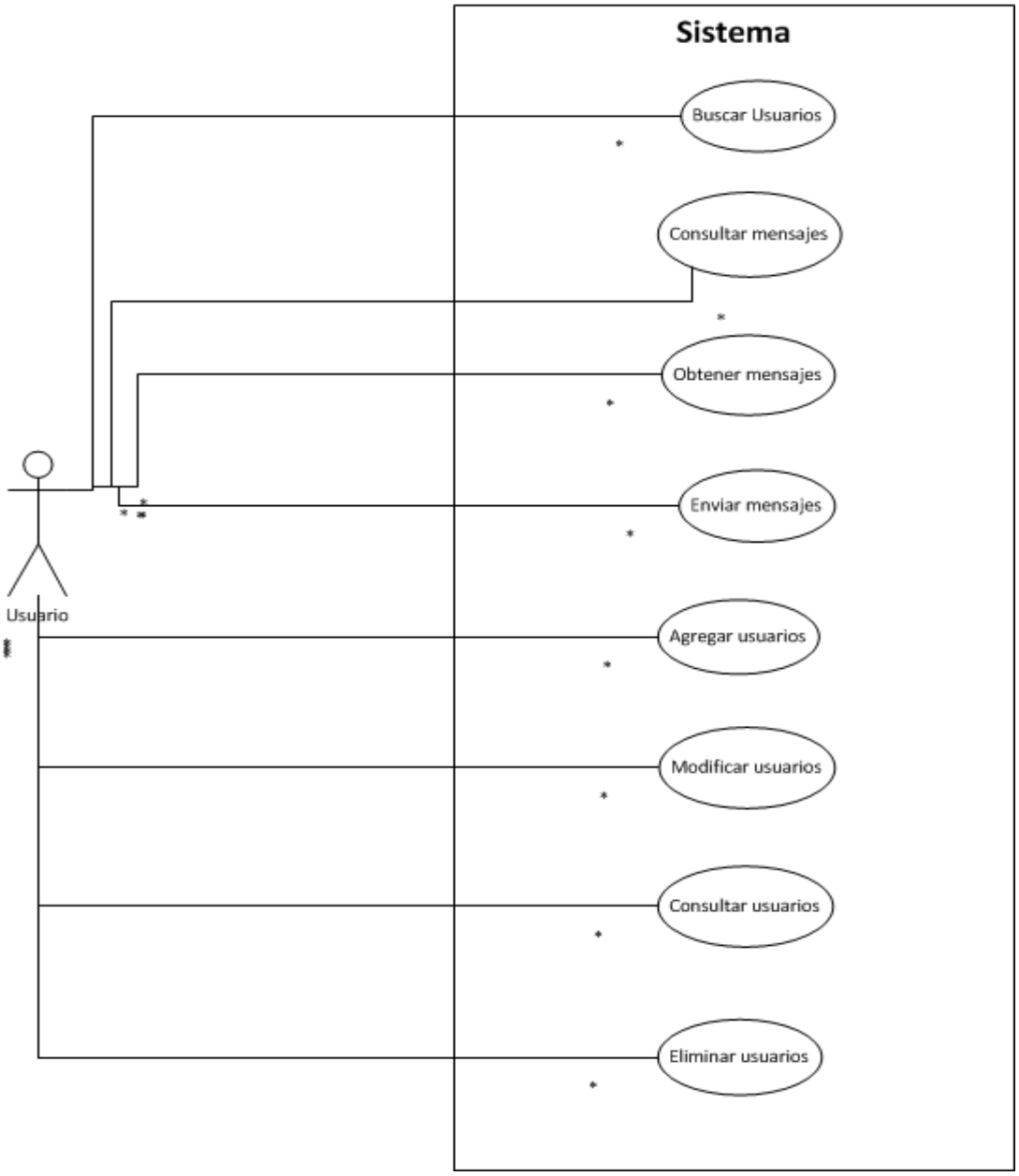
- Lenguaje de programación JEE 5.0
- Lenguaje de programación Adobe Flex 3.0
- Lenguaje de programación AspectJ para la implementación de interceptores con AOP.
- Servidor de aplicaciones Jboss 5.0
- Manejador de base de datos MySQL 5.0

- Framework de persistencia Hibernate 3.0
- Framework de desarrollo Spring 3.0
- Solución de integración entre Flex y JEE Granite Data Services
- Framework de desarrollo en Flex Cairngorm
- Motor de búsquedas fulltext Lucene
- Servicio de mensajes JMS

Actividades realizadas

Las actividades que tuve asignadas para este proyecto fueron la construcción de los siguientes módulos.

Módulo en backend del buscador de usuarios.
Módulo en backend de mensajes.
Módulo en frontend de la administración de usuarios.



Capítulo 2. Figura 10

El buscador de usuarios se realizo utilizando índices fulltext generados con la herramienta de Lucene, la cual permite generar índices de las entidades que sean anotadas para tal fin, de esta forma utilizando hibernate search se pueden obtener dichas entidades al buscarlas sobre los índices generados.

El módulo de mensajes consistió en la creación de las colas de mensajes de jms y del cliente que consumía dichas colas en el front con Flex.

El módulo de administración de usuarios del front contemplaba la creación, edición, consulta y eliminación de los usuarios en el sistema, para esto se utilizó el framework Cairngorm para crear los servicios en el front y Granite Data Services para consumir los servicios del back.

La primer complejidad del sistema fue el front-end utilizado ya que al no ser una tecnología java la interacción entre capas no se hace de forma natural, es necesario meter un componente intermedio para que se puedan comunicar, en este caso fue GraniteDS el utilizado, el cual para clases simples funciona correctamente pero empieza a tener problemas al utilizar clases más complejas, cosa que se pudo solventar al crear nuestras propias clases mapeadas.

Otro problema con el que nos encontramos fue al utilizar una tecnología asíncrona como Flex, ya que el flujo de eventos no se comportaba como lo esperábamos, por esta razón utilizamos componentes de action script llamados watchers, los cuales notifican cuando un componente sufre un cambio o termina un llamado asíncrono.

Para este proyecto yo no tenía experiencia utilizando un Framework para java ni un ORM, además de algunos conceptos como transacciones y aspectos, además la arquitectura del sistema cumplía con varios patrones de diseño, los cuales yo desconocía, creo que este proyecto ha sido el mas motivante y desafiante al que me he enfrentado en mi experiencia laboral ya que tuve que estudiar e investigar conceptos y soluciones que no conocía, además aprendí mucho de los arquitectos del proyecto los cuales siempre fueron una guía para lograr el éxito del mismo.

Lo primero que hice fue estudiar patrones de diseño, básicamente los utilizados en el proyecto (mvc, observer, factory, singleton, decorator, controller, view helper, etc.). Ya con algún conocimiento de las bases me dediqué a estudiar sobre el framework utilizado (Spring 2.5) y los módulos que contiene, el uso de aspectos para hacer transaccionales los métodos y también sobre el ORM (hibernate) para entender el mapeo de las entidades.

Cuando finalmente comencé a desarrollar la vista de mis componentes los otros integrantes del equipo ya llevaban desarrollado algunos componentes entonces me basé mucho en lo que ya se tenía.

Este proyecto tuvo aparte otras implicaciones que alargaron la duración del mismo, el más grave fue que la fase de requerimientos no se cerró sino hasta 5 meses después de lo planeado, todo esto porque no se delimito al usuario la funcionalidad del sistema y también por una mala estrategia para la obtención de requerimientos.

Para la obtención de requerimientos se utilizó la técnica del story board pero de una forma poco eficiente ya que el formato de requerimientos utilizado por la empresa era poco eficiente y las pantallas que presentábamos las teníamos que

realizar con un software que nos tomaba mucho tiempo, en lugar de dibujarlas a mano.

Otro problema fue la burocracia con el cliente, para que alguna pantalla fuera aprobada tenía que pasar por la validación de 4 personas y por lo menos dos de ellas no siempre estaban disponibles.

En este proyecto tuve asignado el rol de planeación en el cual me encargaba de apoyar al líder de proyecto en la administración de las fases del proyecto, balancear cargas, asignar tareas, administrar juntas e informar al área de procesos nuestro estatus como equipo. Al principio del proyecto me costó mucho trabajo cumplir con mi rol TSP debido a la nula experiencia que tenia, pero con la ayuda del área de procesos lo pude ir aprendiendo.
Finalmente considero que tuve un desempeño sobresaliente representando mi rol TSP ya que en los últimos ciclos ya no necesitaba supervisión del área de procesos y las tareas fluían rápidamente.

Finalmente aprendí mucho de este proyecto, tanto de las cosas buenas como las malas, aprendí a utilizar un framework, una herramienta de ORM, un lenguaje de front-end, patrones de diseño, herramientas para indexar información, aprendí a desempeñar un rol de planeación TSP, aprendí el impacto que tiene una mala estimación, una mala estrategia de toma de requerimientos, tuve contacto directo con el cliente y conocí los problemas a los que se enfrenta uno al querer explicar algo al cliente sin caer en un lenguaje demasiado técnico.

CAPÍTULO 3. PORTAL FACTURA

Planteamiento del problema

El Sistema de Administración Tributaria estableció que a partir del 1 de enero del 2011 los contribuyentes deberán expedir documentos digitales como comprobantes por las actividades que realicen.

Una factura electrónica, también llamada comprobante fiscal digital, e-factura o efactura, es un documento electrónico que cumple con los requisitos legal y reglamentariamente exigibles a las facturas tradicionales garantizando, entre otras cosas, la autenticidad de su origen y la integridad de su contenido.

La factura electrónica es, por tanto, la versión electrónica de las facturas tradicionales en papel y debe ser funcional y legalmente equivalente a estas últimas. Por su propia naturaleza, las facturas electrónicas pueden almacenarse, gestionarse e intercambiarse por medios electrónicos o digitales.

Contexto de la solución

La factura electrónica es un tipo de factura que se diferencia de la factura en papel por la forma de gestión informática y el envío mediante un sistema de comunicaciones que conjuntamente permiten garantizar la autenticidad y la integridad del documento electrónico.

Una factura electrónica se construye en 2 fases:

1. Se crea la factura tal y como se ha hecho siempre y se almacena en un archivo de datos.
2. Posteriormente se procede a su firma con un certificado digital o electrónico propiedad del emisor que cifra el contenido de factura y añade el sello digital a la misma.

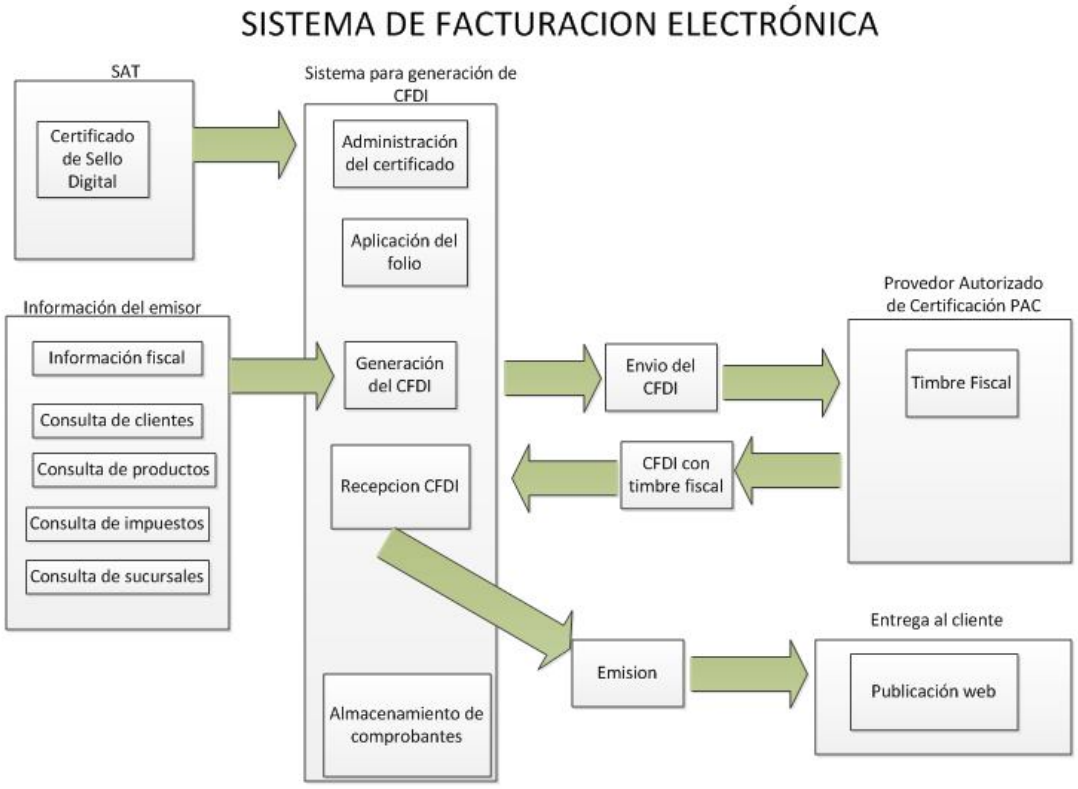
Al terminar obtenemos una factura que nos garantiza:

- que la persona física o jurídica que firmó la factura es quien dice ser (autenticidad) y
- que el contenido de la factura no ha sido alterado (integridad).

El emisor envía la factura al receptor mediante medios electrónicos, Si bien se dedican muchos esfuerzos para unificar los formatos de factura electrónica, actualmente está sometida a distintas normativas y tiene diferentes requisitos legales exigidos por las autoridades tributarias de cada país, de forma que no siempre es posible el uso de la factura electrónica, especialmente en las relaciones con empresas extranjeras que tienen normativas distintas a la del propio país.

Los requisitos legales respecto al contenido mercantil de las facturas electrónicas son exactamente las mismas que regulan las tradicionales facturas en papel. Los requisitos legales en relación con la forma imponen determinado tratamiento en aras de garantizar la integridad y la autenticidad y ciertos formatos que faciliten la interoperabilidad.

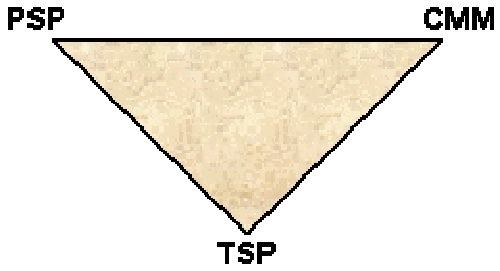
La factura electrónica permite que instituciones, empresas y profesionales dejen atrás las facturas en papel y las remplacen por la versión electrónica del documento tributario. Tiene exactamente la misma validez y funcionalidad tributaria que la factura tradicional en papel. Todo el ciclo de la facturación puede ser administrado en forma electrónica.



Capítulo 3. Figura 11

Metodología empleada

El desarrollo de software en Cuallisys se basa en el seguimiento de las metodologías PSP y TSP, las cuales son a su vez dos de los tres vértices en los cuales descansa el proceso de mejora que trabaja sobre tres niveles de la organización como se muestra en la figura 12.



Capítulo 3. Figura 12

CMM se enfoca a nivel organizacional.
TSP se enfoca a un proceso de grupos de trabajo.
PSP se enfoca a un nivel personal.

PSP (Personal Software Process)

Personal Software Process es un proceso diseñado para ayudar a los ingenieros de software a controlar, manejar y mejorar su trabajo. PSP está basado en una motivación: La calidad del software depende del trabajo de cada uno de los ingenieros de software. Debido a que los costos de personal constituyen el 70% del costo del desarrollo de software, las capacidades y hábitos de trabajo de los ingenieros determinan en gran manera los resultados del desarrollo del software.

Basado en prácticas encontradas en CMM, el PSP puede ser usado por ingenieros para estructurar y disciplinar el desarrollo de software. El ingeniero de software podrá planear mejor el trabajo, conocer con precisión el desempeño, medir la calidad del producto y mejorar las técnicas.

PSP puede ser aplicado en:

- Desarrollo de programas.
- Definición de requerimientos.
- Documentación.
- Pruebas de sistema.

- Mantenimiento de sistemas.

El Proceso Personal de Software está alineado y diseñado para emplearse en organizaciones con modelos de procesos CMMI o ISO 15504. Fue propuesto por Watts Humphrey en 1995 y estaba dirigido a estudiantes. A partir de 1997 con el lanzamiento del libro "An introduction to the Personal Software Process" se dirige a Ingenieros junior.

El PSP provee a los ingenieros de disciplina para la mejora de los procesos de desarrollo personal de software. El PSP ayuda a los ingenieros a:

- Mejorar sus estimaciones y la planeación de sus tareas.
- Hacer compromisos que puedan cumplir.
- Gestionar la calidad de sus proyectos.
- Reducir el número de defectos en su trabajo.

El objetivo del PSP es ayudar a los desarrolladores a generar productos con calidad, sin defectos y en tiempo.

Estructura

El estudio de PSP sigue un enfoque de mejora evolutiva, un ingeniero aprendiendo a integrar el PSP en su proceso inicia en un primer nivel PSP0, y conforme avanza el proceso madura hasta PSP2.1

Todos los niveles cuentan con scripts, checklist y templates para guiar a los ingenieros en su estudio.

Proceso

El inicio del PSP es el documento de requerimientos liberado.

PSP0, PSP0.1 (Introduce la disciplina del proceso y mediciones).
PSP0 tiene tres fases: Planeación, desarrollo (diseño, codificación, pruebas) y post mortem. Una línea base es establecida de acuerdo a las métricas actuales: Tiempo que toma programar, defectos insertados y removidos, tamaño del programa. En el postmortem el ingeniero se asegura que todos los datos del proyecto han sido apropiadamente registrados y analizados. PSP0.1 Avanza en el proceso agregando un estándar de código, una medida del tamaño y el desarrollo de un plan de mejora personal. En el plan de mejora personal el ingeniero registra ideas para mejorar en su propio proceso.

PSP1, PSP1.1 (Introduce a la estimación y planeación).
Basándose a partir de la línea base de datos obtenidos en PSP0 y PSP0.1, el ingeniero estima que tan grande será el nuevo programa y prepara un reporte de pruebas (PSP1). Los datos históricos de los proyectos previos son usados para estimar el tiempo total. Cada nuevo proyecto registrará un tiempo actual. Esta información es usada para planear y estimar las tareas (PSP1.1).

PSP2, PSP2.1 (Introduce administración de la calidad y el diseño).
PSP2 agrega dos nuevas fases: Revisión del diseño y Revisión del código. La prevención y remoción de defectos es el principal objetivo de PSP2. Los ingenieros aprenden a evaluar y mejorar su proceso midiendo que tan grande se pueden hacer las tareas de acuerdo al número de defectos insertados en cada fase del desarrollo. Los ingenieros construyen y utilizan un checklist para las revisiones de diseño y código. PSP2.1 introduce especificaciones del diseño y técnicas de análisis.

PSP3 es la evolución a TSP

El PSP tiene cuatro métricas principales:

- Tamaño - Medición del tamaño de un producto, pudieran ser Líneas de código (LOC).
- Esfuerzo - Tiempo requerido para completar una tarea, usualmente medido en minutos.
- Calidad - El número de defectos en el producto.
- Planeación - Medición del progreso del proyecto, se lleva un seguimiento de las fechas de término planeadas contra las fechas reales.

TSP (Team Software Process)

Team Software Process (TSP) es un marco para el desarrollo de software que pone igual énfasis en el proceso, producto y trabajo en equipo. Al igual que PSP, TSP fue propuesto por Watts Humphrey.

TSP se basa en PSP, y se fundamenta en que el software, en su mayoría es desarrollado por equipos, por lo que los ingenieros deben primero saber controlar su trabajo, y después saber trabajar en equipo. TSP le enseña a los ingenieros a construir equipos auto dirigidos y desempeñarse como un miembro efectivo del equipo. También muestra a los administradores como guiar y soportar estos equipos.

Estrategia TSP.

- Proveer un proceso sencillo basado en PSP.

- Desarrollar productos en varios ciclos. Ciclo de TSP: Lanzamiento, Estrategia, Plan, Requerimientos, Diseño, Implementación, Pruebas, Postmortem.
- Establecer medidas estándares para calidad y desempeño.
- Proveer definiciones de roles, y evaluaciones de rol y de equipo.
- Requiere disciplina de proceso.
- Provee guía para manejo de problemas de trabajo en equipo.

Un equipo TSP consiste en desarrolladores entrenados en PSP que participan en áreas de responsabilidad dentro del proyecto, lo que conlleva a que el proyecto sea administrado por el propio equipo. Usando sus habilidades en PSP el equipo realiza los planes, las estimaciones y controla la calidad. Esto les ayuda a conocer el seguimiento de su plan y producir software de gran calidad.

El Team Software Process junto con el Personal Software Process ayudan a los ingenieros a:

- Asegurar la calidad en sus productos de software.
- Mejorar los procesos administrativos en la organización.

Grupos de ingenieros usan TSP para aplicar conceptos de integración de equipos en el desarrollo de sistemas intensivos de software. Un proceso de lanzamiento guía al equipo y los administradores para:

- Establecer objetivos.
- Definir los roles del equipo.
- Identificar los riesgos.
- Producir un plan del equipo.

Después del lanzamiento el TSP define un marco de trabajo para la administración, seguimiento y reporte del avance del plan del equipo. TSP ayuda a la organización a establecer prácticas ingenieriles maduras y disciplinadas que producen software seguro y confiable.

El uso de cualquier proceso de desarrollo se debe implementar por niveles, en el caso de PSP y TSP los niveles de madurez que alcanza y que es lo que se logra en cada uno son los siguientes:

- Nivel 1 - inicial:
    - Seguimiento y control de proyectos.
    - Planeación de los proyectos.
- Nivel 2 - repetible:
    - Revisión entre colegas.
    - Ingeniería del producto de software.
    - Manejo integrado del software.

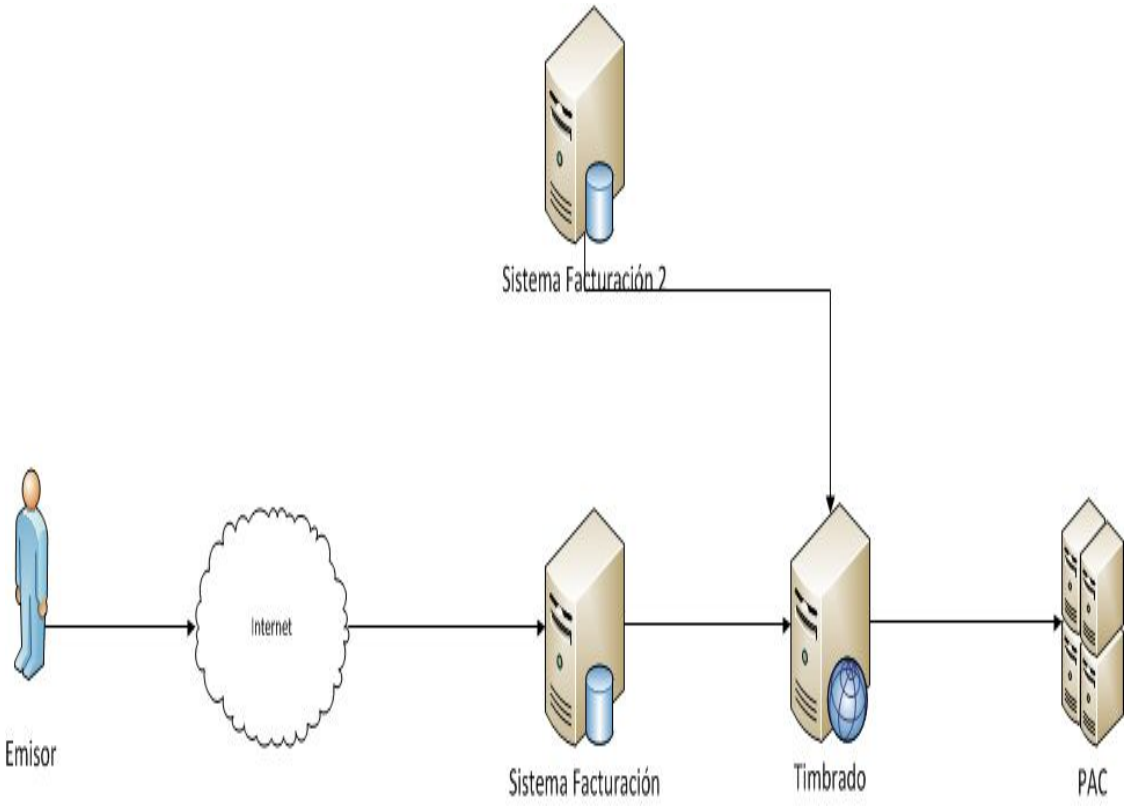
    - Definición del proceso de software.
    - Foco del proceso de software.
- Nivel 3 - Definido:
    - Control de calidad.
    - Administración cuantitativa del proyecto.
- Nivel 4 - Controlado:
    - Administración de los cambios del proceso.
    - Administración del cambio tecnológico.
    - Prevención de defectos.

Solución propuesta

Se solicitó crear un sistema que fuera un portal web para la creación de comprobantes fiscales digitales que cumplan con las disposiciones del SAT y ayude a los emisores de factura a administrar sus clientes y productos con el fin de generar sus comprobantes de forma rápida y sencilla.
Los requerimientos que debe de cumplir el sistema son:

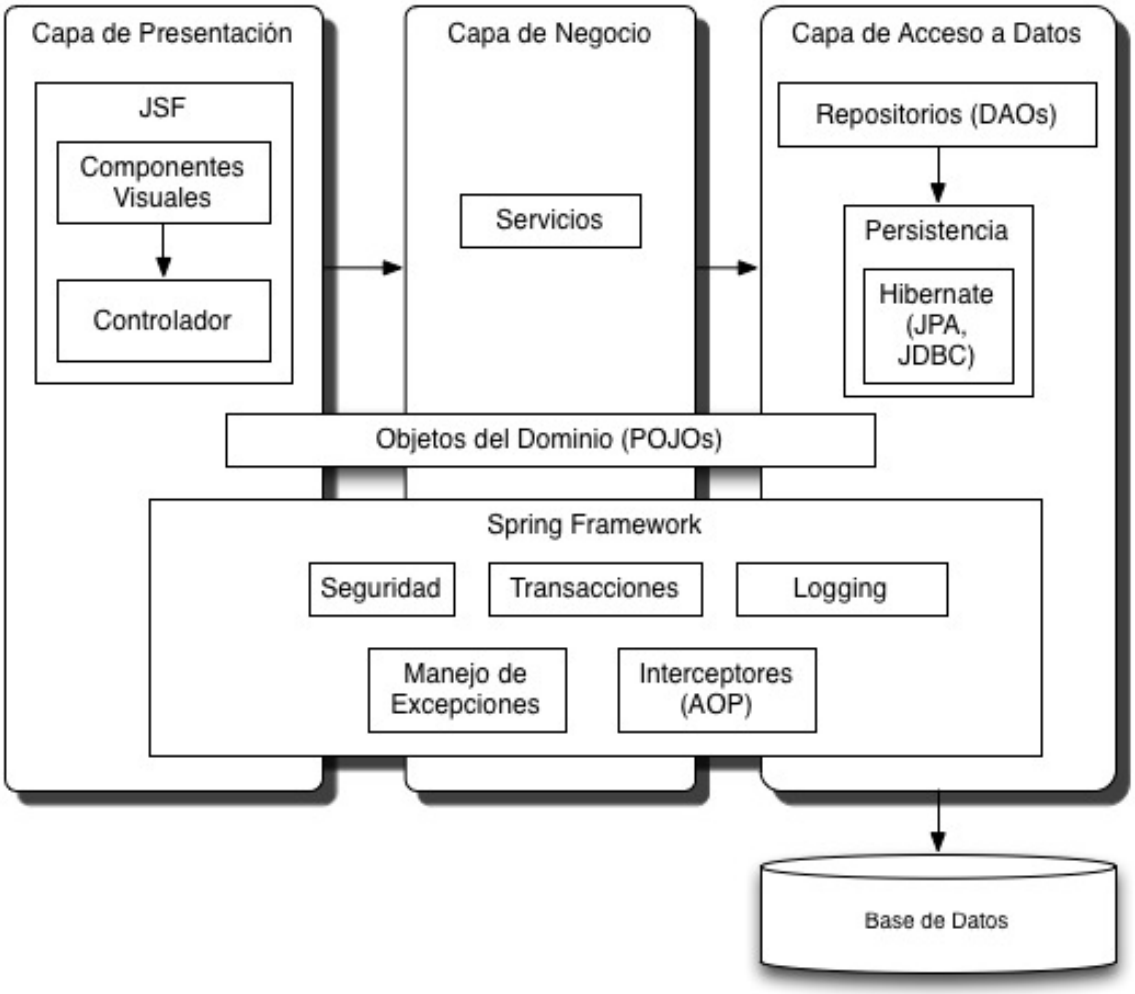
- Los usuarios ingresan al sistema por medio de un login y una contraseña.
- Los usuarios podrán recuperar su contraseña para ingresar al sistema.
- El usuario puede visualizar su estado, esto incluye operaciones disponibles, carga del logo, carga de certificados, carga de folios.
- El sistema debe mostrar como parte de la ayuda el manual de usuario del sistema.
- Los usuarios podrán consultar en el sistema la guía para obtener la guía para generar la firma electrónica avanzada (FIEL) y el certificado de sello digital (CSD).
- El administrador puede crear, modificar y consultar emisores, puede generar nuevas credenciales o agregarles saldo.
- El sistema debe ser capaz de administrar paquetes de las distintas operaciones que se pueden realizar en la empresa.
- El sistema deberá permitir administrar las plantillas utilizadas para la generación de las representaciones impresas de los comprobantes fiscales digitales.
- El usuario puede administrar los detalles de su cuenta, datos fiscales, datos de contacto, dirección y el logo.
- El sistema debe administrar los certificados del usuario proporcionados por el SAT.

- El sistema deberá permitir administrar los folios personalizados con los que contarán los comprobantes fiscales digitales.
- El sistema deberá permitir administrar clientes del emisor.
- El sistema debe permitir la administración de las sucursales que manejen las empresas de los usuarios.
- El sistema deberá permitir administrar los productos de los emisores, los cuales se deberán utilizar para la generación de los comprobantes fiscales digitales.
- El sistema deberá permitir administrar los usuarios del emisor, se deben poder asignar perfiles a los usuarios así como generar nuevas credenciales.
- Se debe establecer una plantilla default para la generación de comprobantes.
- El sistema deberá permitir la expedición de comprobantes fiscales digitales incluyendo la generación de su representación impresa.
- El sistema deberá permitir la expedición de comprobantes CBB incluyendo la generación de su representación impresa.
- El sistema debe administrar los comprobantes que generó el usuario del mismo.
- El sistema deberá permitir generar reportes acerca de los comprobantes generados.
- El sistema debe permitir cerrar la sesión del usuario.



Capítulo 3. Figura 13

La arquitectura propuesta para el sistema fue la siguiente:



Capítulo 3. Figura 14

Para el desarrollo de este sistema, el cual será una aplicación Web, se seguirá el patrón de diseño Modelo Vista Controlador (MVC), el cual permite separar los datos de la aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos.

La tecnología a utilizar para implementar la interfaz de usuario será JavaServer Faces (JSF) dada la experiencia del equipo de desarrollo en esta herramienta. Con el uso de esta tecnología se asegura el uso de un estándar reconocido en el medio para desarrollo de interfaces gráficas dentro de aplicaciones empresariales.

Como gestor de base de datos se utilizará MySQL por ser un gestor robusto y simple que no impone restricciones de licencia para su uso dentro del desarrollo de este sistema. Junto con esta herramienta se usará Hibernate, una herramienta que facilita el almacenamiento y la obtención de los objetos de dominio

implementados en Java a través de un mapeo Objeto/Relacional (ORM). Ambas tecnologías constituyen la capa de acceso a datos de nuestro sistema.

La capa de servicios será implementada utilizando Spring Framework principalmente por la versatilidad que provee para implementar patrones como la inyección de dependencias (DI) para interconectar todos servicios que serán implementados dentro del sistema y la facilidad para exponerlos a la capa de vista. Otros aspectos a implementar a través del uso de esta tecnología son la seguridad de la aplicación (Spring Security), el manejo de transacciones y el uso de la programación orientada a aspectos (AOP) para implementar cuestiones como el logging de información.

Tecnología utilizada

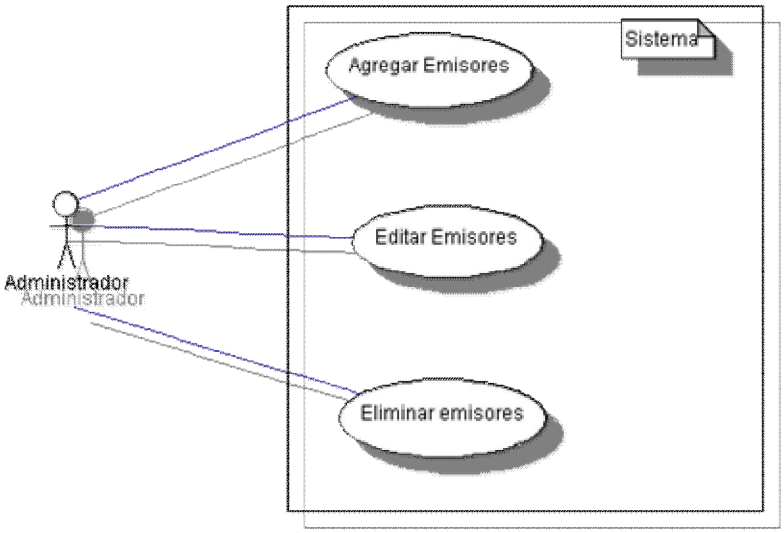
- Lenguaje de programación JEE 5.0
- Framework para interfaces de usuario Java Server Faces 2.2
- Librería de reportes Jasper Reports 4.0
- Librería de componentes JSF Primefaces 2.2
- Lenguaje de programación AspectJ para la implementación de interceptores con AOP.
- Servidor de aplicaciones Jboss 5.0
- Manejador de base de datos MySQL 5.0
- Framework de persistencia Hibernate 3.6
- Framework de desarrollo Spring 3.0

Actividades realizadas

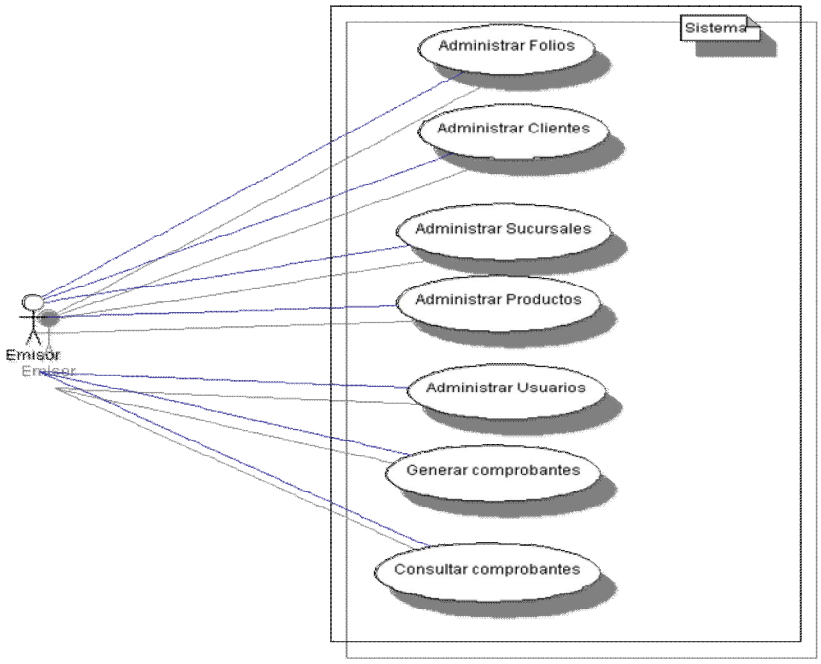
Las actividades que tuve asignadas en este proyecto fueron las siguientes:

- Definir la arquitectura del sistema.
- Crear la instalación del sistema.
- Administración del código.
- Definir la estructura de la base del sistema.
- Modelado de las entidades base del sistema.
- Creación de los servicios de Emisores.
- Creación de los servicios de Folios.
- Creación de los servicios de Clientes.
- Creación de los servicios de Sucursales.
- Creación de los servicios de Consulta.
- Creación componente de vista Comprobantes.
- Creación componente de vista Folios.

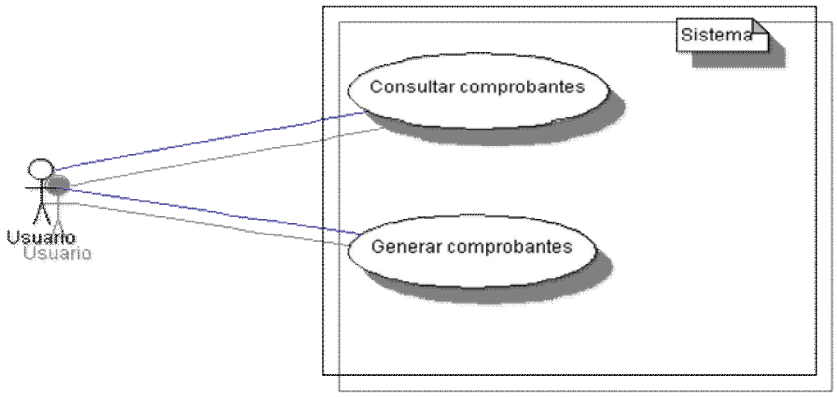
- Creación componente de vista Productos.
- Creación componente de vista Consulta.
- Creación componente de vista Sucursales.
- Creación componente de vista Usuarios.



Capítulo 3. Figura 15



Capítulo 3. Figura 16



Capítulo 3. Figura 17

La definición de la arquitectura surgió del análisis de los requerimientos, de la experiencia de proyectos anteriores y de asesorías con los arquitectos de dichos proyectos, se quería que la comunicación del back con el front fuera de forma natural por lo que se optó por utilizar JavaServerFaces como tecnología para la interfaz de usuario.

Después de investigar librerías de JSF a utilizar se decidió el uso de PrimeFaces por el número de componentes que tiene la facilidad de uso, de tal forma que la curva de aprendizaje no fuera tan pronunciada.

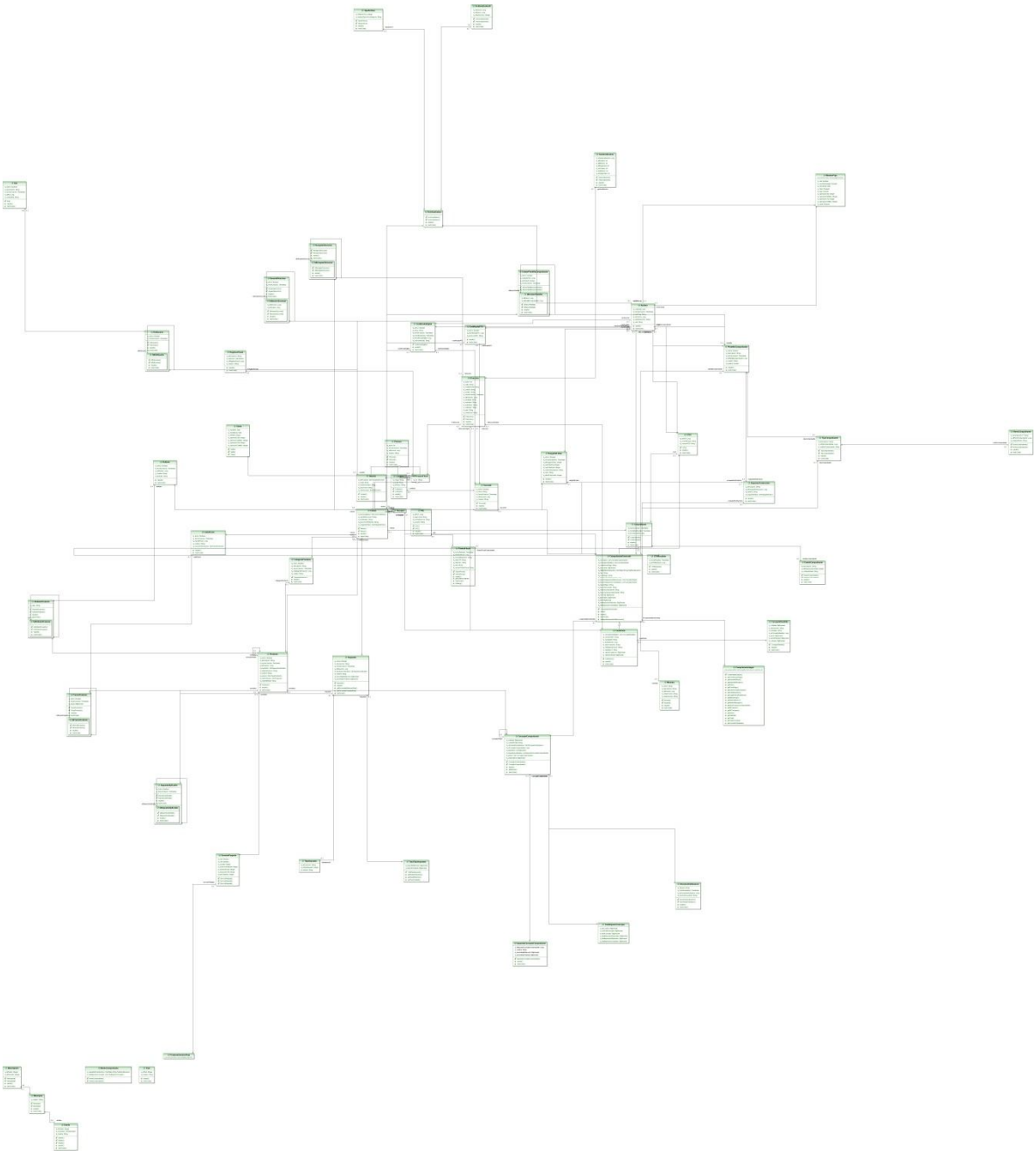
La instalación del sistema consistió en la puesta en marcha del servidor de aplicaciones (Jboss 6), la configuración del framework de Spring 3, la conexión con Hibernate 3.2 y PrimeFaces 2.2.

Dentro de la instalación del sistema se contempló la creación de los proyectos de Maven de la aplicación, esto para llevar a cabo la administración del código y no depender de ningún IDE en el desarrollo.

Dentro de la administración del código se contempló la creación de los repositorios de subversión y la definición de las políticas de administración de código. Dichas políticas se basaron en las mejores prácticas en la administración de versiones con Subversion.

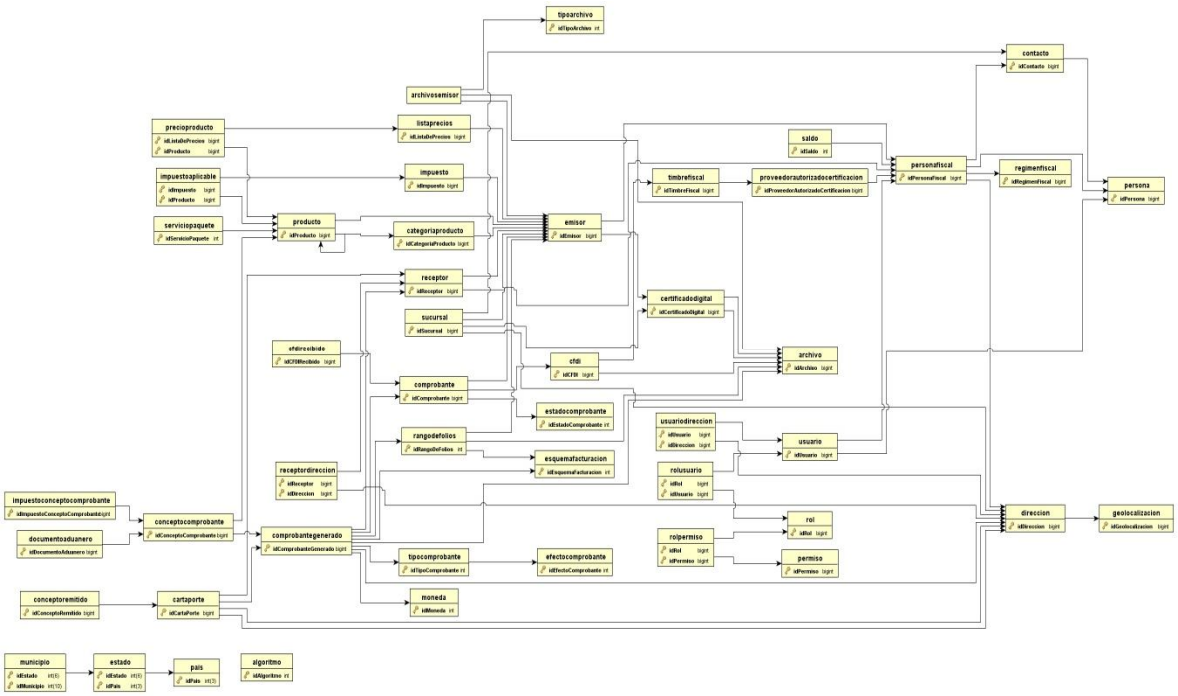
Se modelaron todas las entidades base del sistema de acuerdo al anexo 20 publicado por el SAT y a los requerimientos del cliente. Se mapearon las entidades a la base de datos utilizando Hibernate como ORM.

Después del análisis del anexo 20 se genero el diagrama de clases del sistema de la figura 18.



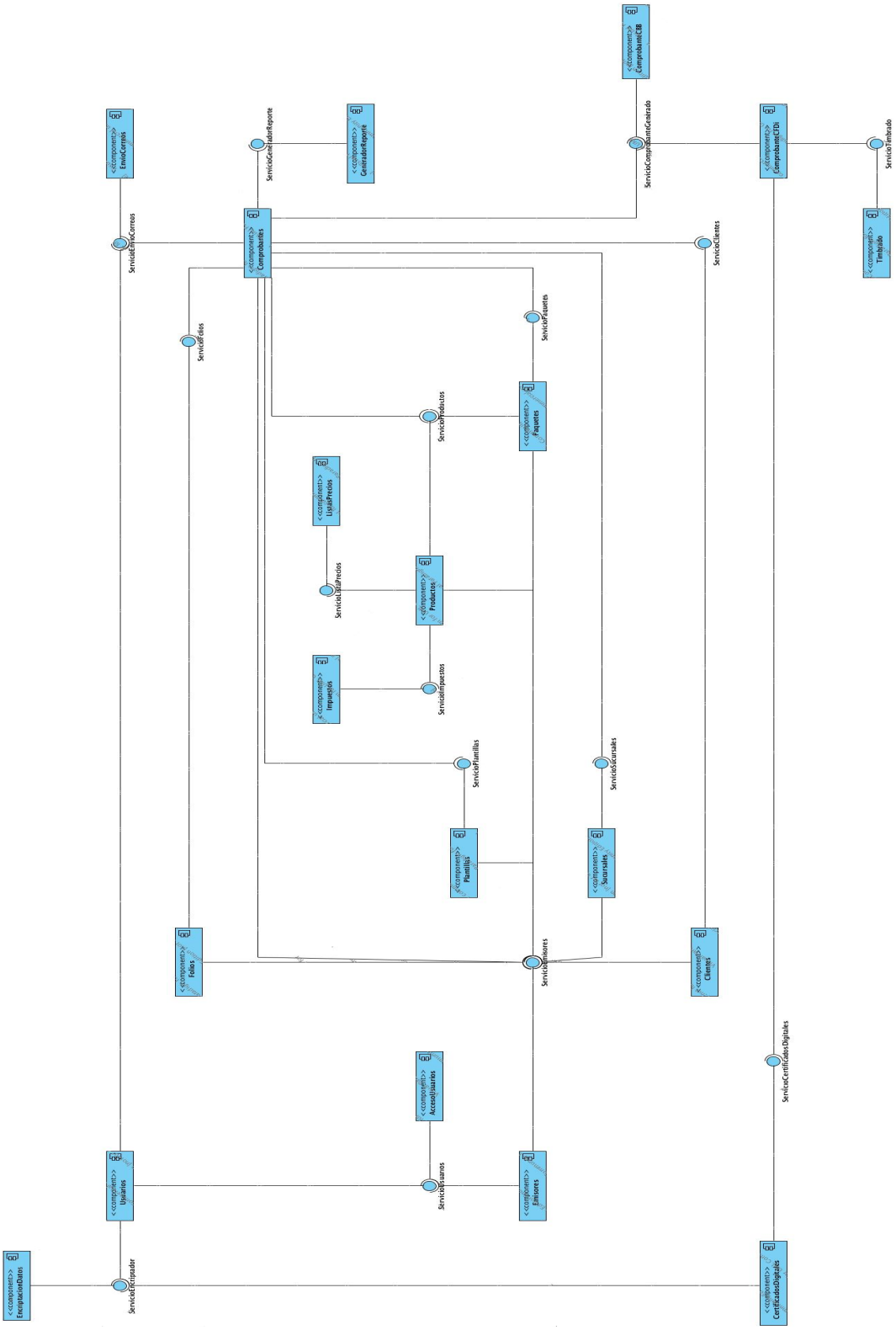
Capítulo 3. Figura 18

Junto con el diagrama de clases también se obtuvo el diagrama entidad relación del sistema mostrado en la figura 19.



Capítulo 3. Figura 19

Con los diagramas de clases y de entidad relación se definieron los módulos del sistema junto con su correspondiente diagrama. Figura 20. Dado que el sistema no fue concebido en su totalidad sino que se pensó que su crecimiento fuera gradual se pensó en la creación de los componentes como un conjunto de servicios que consumen otros servicios, es por esto que el diagrama solo refleja la funcionalidad mas importante en el sistema y la interacción por medio de interfaces entre los diferentes componentes.



Capítulo3. Figura 20

Los módulos definidos y su funcionalidad fue la siguiente:

- Usuarios. Este módulo es el encargado de llevar a cabo la administración de usuarios del sistema así como la asignación de sus permisos.

- Folios. Este módulo es el encargado de llevar a cabo las operaciones para la administración de los folios fiscales tramitados ante el SAT y de los folios para control interno de los emisores.

- Acceso Usuarios. Este módulo es el encargado de las operaciones de acceso al sistema.

- Encriptación de Datos. Este módulo es el encargado de cifrar y descifrar la información en la base de datos.

- Emisores. Este módulo es el encargado de llevar a cabo las operaciones para la administración de los emisores.

- Certificados digitales. Este módulo es el encargado de la administración de los certificados digitales del emisor y sus sucursales.
- Clientes. Este módulo contiene las operaciones necesarias para llevar a cabo la administración de los clientes del emisor.

- Sucursales. Este módulo contiene las operaciones necesarias para llevar a cabo la administración de las sucursales del emisor.

- Plantillas. Este módulo es el encargado de administrar las plantillas utilizadas en las representaciones impresas de los comprobantes fiscales.

- Impuestos. Este módulo es el encargado de las operaciones para la administración de los impuestos en el sistema, también define el cálculo de los impuestos aplicados a los productos.

- Productos. Este módulo es el encargado de la administración de los productos que factura el emisor.

- Listas de precios. Este módulo es el encargado de administrar las listas de precios del emisor.

- Paquetes. Este módulo contiene las operaciones para la administración de los paquetes de operaciones en el sistema, define que tipo de operaciones y cuantas, también administra el saldo del emisor.

- Comprobantes. Este módulo es el encargado de administrar los comprobantes en el sistema en forma genérica.

- Generador de reportes. Este módulo es el encargado para la generación de reportes en el sistema.

- Envío de correos. Este módulo es el encargado del envío de correos en el sistema.

- Comprobante CFDi. Este módulo contiene las operaciones específicas necesarias para manejar la lógica que atañe a los comprobantes CFDi.

- Comprobante CBB. Este módulo contiene las operaciones específicas necesarias para manejar la lógica que atañe a los comprobantes CBB.

- Timbrado. Este módulo es el encargado de generar las conexiones con el PAC para llevar a cabo el timbrado de los comprobantes.

La creación de los servicios contempló desde la creación del DAO necesario hasta la creación de los servicios, para los DAO's se utilizó un DAO Genérico que por medio de genéricos se acoplaba a cualquier entidad del sistema.

La creación de la vista incluye los archivos xhtml necesarios para crear la vista, el controlador de la vista y los Backing Beans necesarios que modelen los componentes de la vista.

Conclusiones y resultados

Los resultados obtenidos en el proyecto fueron satisfactorios, el sistema ya se encuentra en producción con 200 usuarios registrados hasta enero del 2013 y teniendo 17 ingresos nuevos en promedio cada mes.

El promedio de facturas diarias elaboradas en el sistema es de 56, teniendo una disponibilidad promedio de 99.7%, reservándonos el .3% restante para tareas de mantenimiento y actualización del sistema.

En diciembre del 2011 el SAT público cambios para la versión 3.2 de la factura digital, dada la arquitectura implementada en nuestro sistema dicha actualización se pudo realizar sin mayores contratiempos y completamente transparente al usuario final.

Datos obtenidos por el área de procesos y calidad al analizar nuestras métricas arrojan que el proyecto salió en el tiempo estimado, con un error de 4.3%, nuestro índice de remoción de defectos fue de 80% y la productividad obtenida fue de 17 LOC/h para JEE y 12 LOC/h para JSF.

El principal problema en este proyecto fue que no se contaba en un inicio con una entidad experta en la materia que nos asesorara en cuestiones fiscales y contables, razón por la cual en algunos momentos caímos en el retrabajo de varios módulos ya que al no contar con la definición exacta del problema tuvimos que corregir en el camino.

Además el SAT no cuenta con un área especializada en la resolución de dudas sobre la factura electrónica, solo unos documentos sirven como guía para el desarrollo.

Como arquitecto del proyecto mi rol en el equipo TSP fue el administrador del diseño, mi trabajo consistió en verificar que el diseño se respetara conforme se iba avanzando en la etapa de requerimientos, fui el responsable de que la matriz de trazabilidad fuera consistente a lo largo del proyecto y también fui el responsable de la generación de los documentos técnicos como el documento de arquitectura y el documento de requerimientos.

CONCLUSIONES

Gracias a la formación que me dio la Facultad de Ingeniería he logrado desenvolverme profesionalmente en mi ámbito laboral.

Cuando se me han presentado retos he logrado afrontarlos gracias a la experiencia adquirida pero también a esos conocimientos básicos que recibí durante mi instancia en la Universidad.

También he logrado comprender que la formación de la Universidad en el aspecto social y humanista me ha dotado de herramientas poderosas en mi interacción con las personas.

Agradezco enormemente a la Universidad Nacional Autónoma de México y a la Facultad de Ingeniería el haberme dado las herramientas necesarias para poder afrontar los retos profesionales que se me han presentado.

Las ventajas competitivas de las empresas ya no se construyen solo con dinero, actualmente las ventajas más poderosas son contar con gente comprometida, con pasión en lo que hace y capaces de afrontar cualquier reto que se les presente, es este talento de las personas el que se adquiere desde las aulas, desde la formación académica y es lo que ha logrado la Facultad de Ingeniería en muchos de nosotros.

REFERENCIAS

HUMPHREY, Watts S.
Introduction to the Personal Software Process.
1st Edition, Addison-Wesley Professional, Reading, M.A. 1996.

HUMPHREY, Watts S.
PSP, A self-Improvement Process for Software Engineers.
1st Edition, Addison-Wesley Professional. SEI Series in Software Engineering. 2005.

MINTER, Dave.
Beginning Spring 2: From Novice to Professional (Beginning: From Novice to Professional).
1st Edition, Apress. 2007.

BAUER, Christian; KING, Gavin.
Java Persistence with Hibernate.
Revised Edition, Manning Publications. 2006.

GONCALVES, Antonio.
Beginning Java EE 6 with GlassFish 3 (Expert's Voice in Java Technology).
2nd Edition, Apress. 2010.

NIXON, Robert.
Learning PHP, MySQL, and JavaScript: A Step-By-Step Guide to Creating Dynamic Websites (Animal Guide).
1st Edition, O'Reilly Media, Inc. 2009.

Team Software Process. Overview. Software Engineering Institute. Carnegie Mellon.
http://www.sei.cmu.edu/tsp
Consulta 20 de marzo de 2012.

PSP Personal Software Process. Cuerpo Académico de Tecnologías de Aprendizaje e Ingeniería de Software. Centro de Ciencias Básicas, Universidad Autónoma de Aguascalientes.
http://ingsw.ccbas.uaa.mx/sitio/images/material/psp.htm
Consulta 20 de marzo de 2012.

Spring Framework. SpringSource Community.
http://www.springsource.org
Consulta 20 de marzo de 2012.

PrimeFaces. Prime Technology.
http://primefaces.org
Consulta 20 de marzo de 2012.

JBoss Application Server. JBoss Community.
http://www.jboss.org
Consulta 27 de abril de 2012.

Hibernate. JBoss Community.
http://www.hibernate.org
Consulta 27 de abril de 2012.

Buenas Prácticas de Gestión de Versiones con Subversion. Tecsisa.
http://blogs.tecsisa.com/articulos-tecnicos/buenas-practicas-de-gestion-de-versiones-con-subversion
Consulta 14 de mayo de 2012.

Productos UModel. Diagramas de perfil UML. Altova®.
http://www.altova.com/es/umodel/profile-diagrams.html
Consulta 20 de noviembre de 2012.

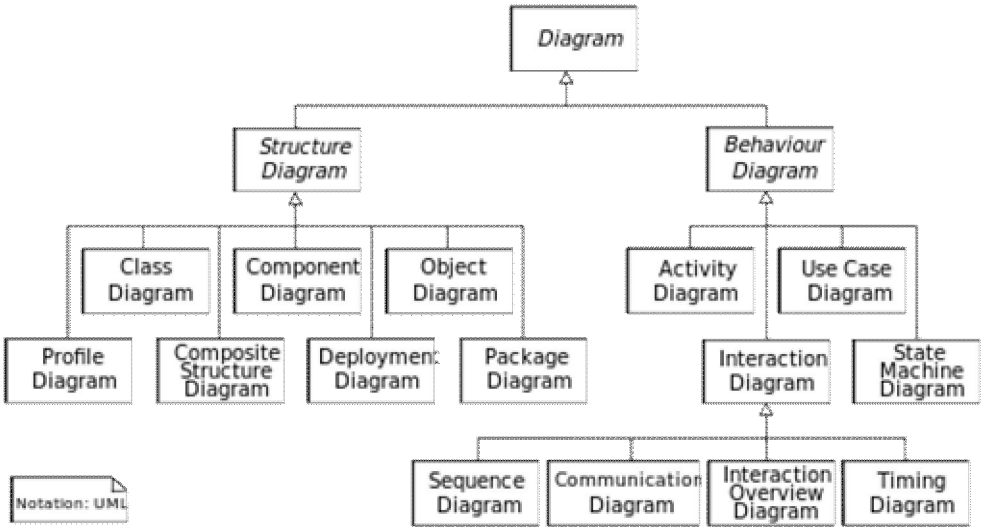
The Java Language Environment. Design Goals of the Java TM Programming Language. Sun Microsystems, Inc.
http://www.oracle.com/technetwork/java/intro-141325.html
Consulta: 6 de octubre de 2012.

ANEXOS

UML

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Diagramas
UML define 14 tipos de diagramas divididos en 2 categorías. Siete diagramas representan información de la estructura del sistema y los otros siete representan tipos generales de comportamiento incluidos cuatro que representan diferentes aspectos de interacciones.



Anexos. Figura 21

UML no restringe elemento a ciertos tipos de diagramas, en general, cualquier elemento UML en cualquier tipo de diagrama.

- *Diagrama de clases:* Describe la estructura de un sistema mostrando las clases del sistema, sus atributos y las relaciones entre las clases.

- *Diagrama de componentes:* Describe como un sistema de software se divide en componentes y muestra las dependencias entre estos.

- *Diagrama de estructura compuesta:* Describe la estructura interna de una clase y las colaboraciones que esta estructura hace posibles.

- *Diagrama de despliegue:* Describe el hardware usado en la implementación de un sistema y los ambientes de ejecución y artefactos desplegados en el hardware.

- *Diagrama de objetos:* Muestra una vista completa o parcial de las instancias específicas de una clase en un momento particular del sistema.

- *Diagrama de paquetes:* Describe como un sistema esta dividido en agrupaciones lógicas y las dependencias entre dichas agrupaciones.

- *Diagrama de perfil:* Funciona a nivel de metamodelos, permiten definir estereotipos, definiciones de etiquetas y restricciones personalizadas en un diagrama UML especial. La personalización de estos estereotipos, etiquetas y restricciones se definen en un perfil, que después se aplica a un paquete.

- *Diagrama de actividades:* Describe paso por paso los flujos de trabajo de negocio y operacionales de los componentes de un sistema. Un diagrama de actividades muestra el flujo de control general.

- *Diagrama de estados:* Describe los estados y sus transiciones de un sistema.

- *Diagrama de casos de uso:* Describe la funcionalidad de un sistema en términos de actores, sus objetivos representados como casos de uso y cualquier dependencia entre esos casos.

- *Diagrama de comunicación:* Modela las interacciones entre objetos o partes en términos de una secuencia de mensajes. Los diagramas de comunicación representan una combinación de información tomada desde el diagrama de clases, secuencia, y diagrama de casos de uso describiendo tanto la estructura estática como el comportamiento dinámico de un sistema.

- *Diagrama global de interacciones:* Muestra una vista global en donde los nodos representan diagramas de comunicación.

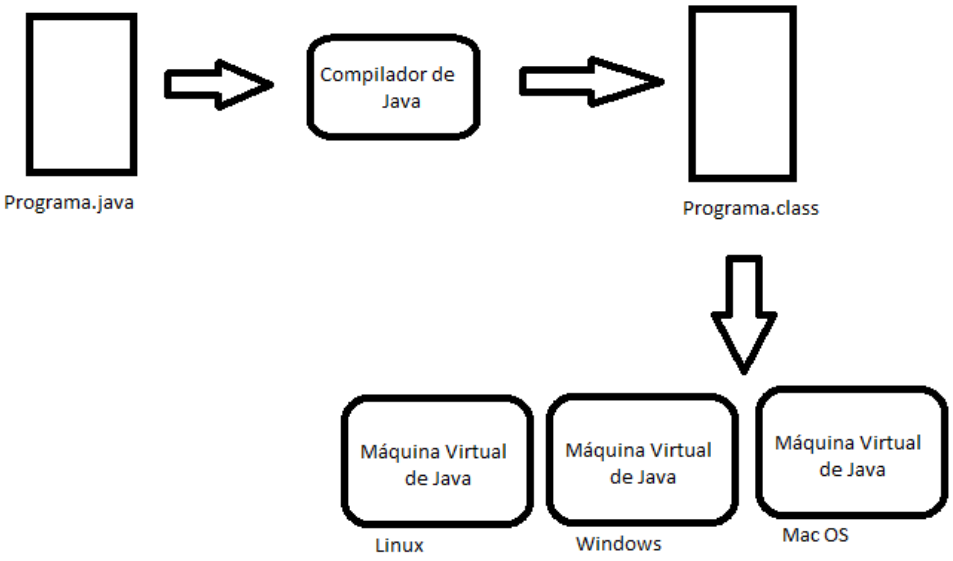
- *Diagramas de secuencia:* Muestra como los objetos se comunican entre si en términos de una secuencia de mensajes. También indica el ciclo de vida de los objetos relativos a esos mensajes.

- *Diagrama de tiempos:* Tipo específico de un diagrama de interacciones que se enfoca en las restricciones de tiempo de los mensajes enviados entre objetos.

JAVA

Java es un lenguaje de programación de alto nivel orientado a objetos, desarrollado por James Gosling en 1995 para Sun Microsystems (comprada por Oracle Corporation). El lenguaje en sí mismo toma mucha de su sintaxis de C, Cobol y Visual Basic, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. La memoria es gestionada mediante un recolector de basura.

Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador Java también es posible.



Anexos. Figura 22

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems desde 1991 y liberadas en 1995.

El lenguaje Java se creó con cinco objetivos principales:

1. Debería usar el paradigma de la programación orientada a objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debería incluir por defecto soporte para trabajo en red.
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Objetivos de diseño del lenguaje:

Simple, orientado a objetos y familiar. Java es un lenguaje que puede ser utilizado sin un programa extenso de estudio, es un lenguaje orientado a objetos y es un lenguaje que le resulta familiar a los desarrolladores al parecerse a C++.

Robusto y seguro. Debe ser diseñado para generar software confiable, debe contener formas de comprobar el código en tiempo de compilación y en tiempo de ejecución. Estas características guían para que los desarrolladores tengan hábitos confiables de programación. La forma de gestionar la memoria es simple, de esta forma el sistema detectará errores rápidamente.

La tecnología está diseñada para funcionar en entornos distribuidos, lo que significa que la seguridad es importante. Está diseñado con características de seguridad integradas al lenguaje y al sistema en tiempo de ejecución. Permite construir aplicaciones que no pueden ser invadidas desde el exterior.

Arquitectura neutral y portátil. La tecnología está diseñada para soportar aplicaciones que se van a implementar en entornos de red heterogéneos, de tal forma que las aplicaciones serán capaces de ejecutarse en una variedad de arquitecturas de hardware.

El compilador Java genera bytecodes (una arquitectura neutral con un formato intermedio diseñado para transportar código eficientemente a múltiples plataformas de hardware y software. El intérprete natural de Java resuelve el

problema de la distribución binaria y el problema de las versiones, de esta forma el mismo lenguaje de bytes del lenguaje puede correr en cualquier plataforma.

La tecnología Java se basa en especificaciones, define tamaños de sus datos y el comportamiento de sus operadores aritméticos, de esta forma sus programas son los mismos en todas las plataformas ya que no existen incompatibilidades en los tipos de datos.

La arquitectura neutral y el lenguaje portátil de la tecnología Java es conocido como Java Virtual Machine.

Alto rendimiento. El rendimiento siempre es una consideración. La plataforma Java logra un rendimiento superior al adoptar un esquema en el cual el intérprete puede correr a toda velocidad sin la necesidad de comprobar el ambiente en tiempo de ejecución. El recolector de basura se ejecuta como un proceso en segundo plano, lo que garantiza una alta probabilidad de que la memoria está disponible cuando se solicita.

Interpretado, capacidad de generar procesos y dinámico. El intérprete de Java puede ejecutar bytecodes directamente en cualquier maquina donde el intérprete y el sistema de ejecución ha sido portado. La capacidad multihilos de Java permite construir aplicaciones con varios subprocesos simultáneos.

Mientras que el compilador de Java es estricto en su comprobación de tiempo de ejecución, el lenguaje y el sistema de ejecución son dinámicos en sus fases de enlace, esto es que las clases solo son vinculadas cuando es necesario, así los módulos de código pueden estar enlazados a una gran variedad de fuentes.

JEE

Java Platform, Enterprise Edition es una plataforma de programación (parte de la plataforma Java) para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

Java EE tiene varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets (siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios web. Ello permite al desarrollador crear una Aplicación Empresarial portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los

componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel.

Las empresas cada vez requieren aplicaciones más y más complejas, que tengan alta disponibilidad, que sean escalables, seguras, transaccionales, interoperables y distribuidas, JEE provee de un API para el soporte de cada una de estas áreas.

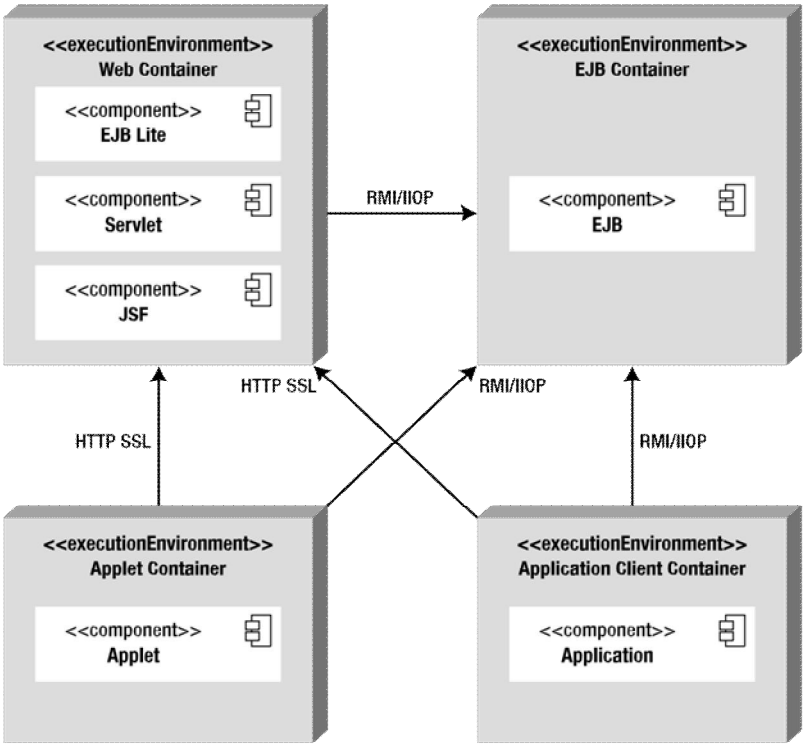
JEE es un conjunto de especificaciones pensadas para aplicaciones empresariales, puede verse como una extensión de Java SE para ayudar en el desarrollo de aplicaciones distribuidas, robustas, poderosas y siempre disponibles.

Estándares.
JEE está basado en estándares. Esto significa un conjunto de especificaciones en el mismo paquete. JEE provee estándares abiertos que son implementados por soluciones comerciales o de código abierto, de esta forma una aplicaciones puede correr en diferentes servidores de aplicaciones con muy pocos cambios.

Arquitectura.
JEE es un conjunto de especificaciones implementadas por diferentes contenedores. Un contenedor es un ambiente de ejecución JEE que provee ciertos servicios a los componentes que albergan. Estos componentes usan contratos bien definidos para comunicarse con la infraestructura de Java EE y con otros componentes.

Java EE es un súper conjunto de la plataforma Java SE, lo que significa que las librerías de Java SE pueden ser utilizadas por cualquier componente de JEE.

La figura muestra la relación lógica entre los componentes estándar de JEE, las flechas representan los protocolos usados por un contenedor para acceder a otro.



Anexos. Figura 23

Componentes.

El ambiente de ejecución de Java EE define cuatro tipos de componentes que una implementación debe soportar.

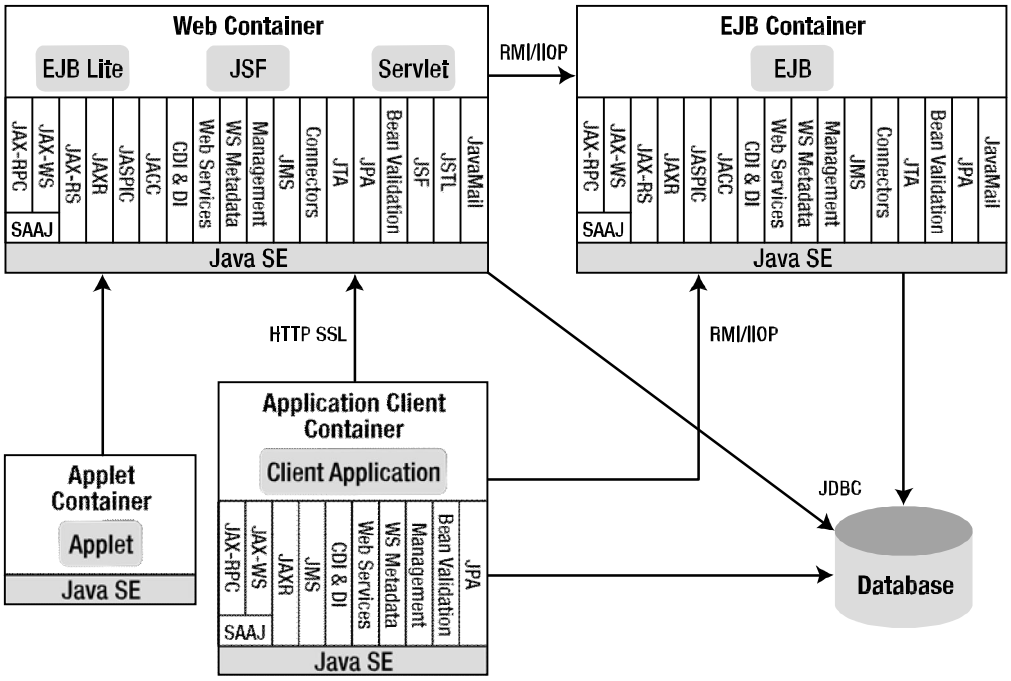
- *Applets:* Aplicaciones gráficas que son ejecutadas en un navegador web. Utilizan librerías de Swing Enriquecidas para proveer poderosas interfaces de usuario.

- *Applications:* Son programas ejecutados en el cliente. Son típicas interfaces de usuario o procesos en batch que tienen acceso a todas las facilidades de JEE.

- *Web applications:* Son ejecutadas en un contenedor Web y responden a peticiones HTTP desde el cliente web.

- *Enterprise applications*: Son ejecutadas en un contenedor EJB. Los EJB (Enterprise Java Beans) son componentes administrados por el contenedor para procesar lógica de negocio transaccional. Pueden ser accedidas local o remotamente.

Contenedores y servicios.
La infraestructura de Java EE esta particionada en dominios lógicos llamados contenedores. Cada contenedor tiene un rol específico, soporta un conjunto de librerías y ofrece servicios a los componentes. En la imagen se muestran los servicios que ofrece cada contenedor.



Anexos. Figura 24

Java Enterprise Edition ofrece los siguientes servicios:

- *Java Transaction API (JTA):* Este servicio ofrece una API de transacciones utilizado por el contenedor y la aplicación. También proporciona una interfaz entre el gestor de transacciones y un administrador de recursos en el nivel de la interfaz del proveedor del servicio.

- *Java Persistence API (JPA):* API estándar para el mapeo Objeto-Relacional. Con el Java Persistence Query Language(JPQL Lenguaje de consulta independiente de Java) se pueden realizar consultas sobre objetos almacenados en la capa de base de datos.

- *Validation*: La validación de Beans provee la declaración de restricciones y validaciones a nivel de clase.

- *Java Message Service (JMS):* Este servicio permite a los componentes comunicarse asíncronamente a través de mensajes.

- *Java Naming and Directory Interface (JNDI):* Esta API, incluida en Java SE, es utilizada para al sistema de nombres y directorios. Las aplicaciones la usan para asociar nombres a objetos y después encontrar esos objetos en un directorio.

- *JavaMail:* Muchas aplicaciones requieren la habilidad de mandar e-mails, lo cual puede ser implementado a través del uso del API de JavaMail.

- *JavaBeans Activation Framework (JAF):* Esta API, incluida en Java SE, provee un marco de trabajo para el manejo de datos en diferentes tipos MIME (extensiones multipropósito de correo de internet).

- *Java API for XML Processing (JAXP):* Provee el soporte para el parseo de documentos con API's SAX y DOM, así como XSTL.

- *Java EE Connector Architecture (JCA):* Permite conectar componentes Java EE con Sistemas de Información Empresarial (EIS). Estos pueden ser bases de datos, ERP, o computadoras centrales.

- *Security Services:* Java Authentication and Authorization Service (JAAS) habilita servicios para autenticar y reforzar el control de acceso a los usuarios. The Java Authorization Service Provider Contract for Containers (JACC) define el contrato entre el servidor de aplicaciones JEE y un servicio proveedor de autorizaciones.

- *Web Services:* Java EE provee soporte para Servicios Web del tipo SOAP o RESTful. Java API for XML Web Services (JAX-WS) provee soporte para Servicios Web usando el protocolo SOAP/HTTP. Java API for RESTful Web Services (JAX-RS) provee soporte para Servicios Web usando el estilo RESTful.

- *Dependency Injection:* Algunos recursos pueden ser inyectados a los componentes administrados por un contenedor. Java EE utiliza las especificaciones CDI (Context and Dependency Injection) asi como DI (Dependency Injection for Java).

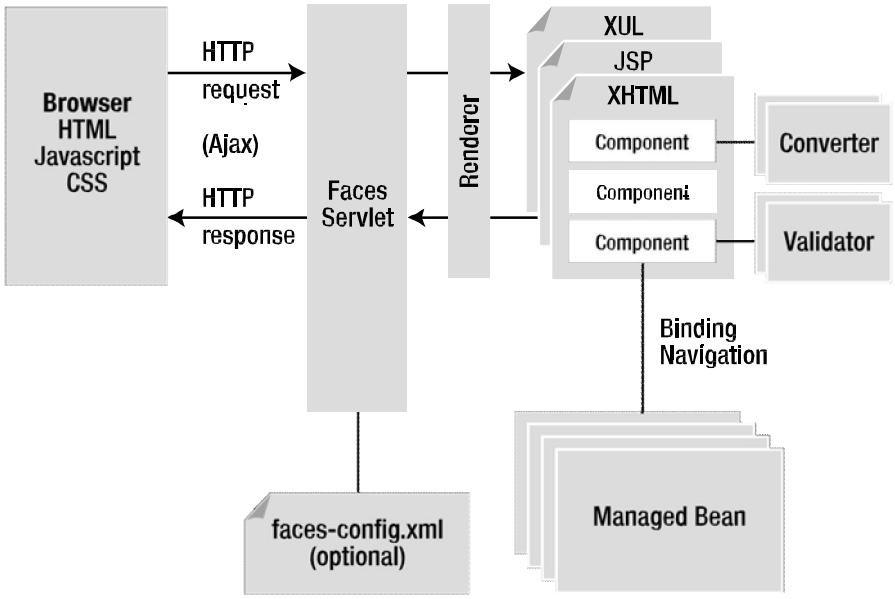
- *Management:* Java EE define API's para administrar contenedores y servidores.

- *Deployment:* La especificación de despliegue de JEE define un contrato entre las herramientas de despliegue y los productos JEE para estandarizar el despliegue de las aplicaciones.

JSF

Es una tecnología y marco de desarrollo para aplicaciones java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones JEE.

Las aplicaciones JSF son aplicaciones Web que interceptan las peticiones HTTP a través del Servlet Faces y generan HTML. La arquitectura de JSF puede verse a muy alto nivel como en la figura.



Anexos. Figura 25

La figura muestra varias piezas importantes de JSF que hacen de su arquitectura rica y flexible:

- *FacesServlet:* Es el servlet principal para cada aplicación y puede ser opcionalmente configurado por el archivo faces-config.xml.

- *Pages and Components:* JSF permite múltiples Lenguajes Descriptores de Páginas (PDL en inglés) como JSP o Facelets.

- *Renderers:* Son los responsables de mostrar los componentes y trasladar las entradas del usuario a las propiedades de los componentes.

- *Converters:* Estos convierten los valores de los componentes (Date, Boolean, etc) en y desde cadenas (String).

- *Validators:* Son los responsables de asegurar que el valor ingresado por el usuario es válido.

- *Managed Bean and navigation:* La lógica del negocio se encuentra en los beans administrados, los cuales controlan también la navegación entre páginas.

PrimeFaces

PrimeFaces es una librería de componentes visuales para Java Server Faces de código abierto que cuenta con gran cantidad de componentes que facilitan la creación de aplicaciones web.

PrimeFaces es de origen turco, desarrollada por Prime Technology bajo la licencia de Apache License V2.

Entre las características con las que cuenta están:

- Amplio conjunto de componentes enriquecidos (Editores, autocompletar, paneles, gráficas, etc.).

- Soporte de Ajax con despliegue parcial lo que permite controlar cuales componentes de la página se actualizan.

- JQuery como librería de JavaScript.

- Temas prediseñados o herramienta de creación de temas ThemeRoller.

- Componentes para el desarrollo de aplicaciones móviles.

- Compatible con otras librerías de componentes de JSF.

- Open Source.

Spring

Spring es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java.

El framework fue lanzado inicialmente bajo Apache 2.0 License en junio de 2003. El primer gran lanzamiento fue la versión 1.0, que apareció en marzo de 2004 y fue seguida por otros lanzamientos en septiembre de 2004 y marzo de 2005.

A pesar de que Spring Framework no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores en Java al ser considerado como una alternativa y sustituto del modelo de Enterprise JavaBeans. Por su diseño el framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria.

Mientras que las características fundamentales de este framework pueden emplearse en cualquier aplicación hecha en Java, existen muchas extensiones y mejoras para construir aplicaciones basadas en web por encima de la plataforma empresarial de Java (Java Enterprise Plataform).

Spring se compone de varios módulos:

- Inversion of Control container: Configuración de los componentes de la aplicación y la administración del ciclo de vida de los objetos de Java a través del patrón de diseño de Inyección de dependencias, el cual establece que se suministran objetos a una clase en lugar de ser la propia clase quien crea el objeto.
- Aspect-oriented programming: Habilita la implementación de rutinas comunes.
- Data access: Trabaja con sistemas manejadores de bases de datos relacionales usando la librería JDBC y herramientas de mapeo Objeto-Relacional y con NoSQL.
- Transaction management: Unifica varias librerías para el manejo de transacciones y coordina transacciones entre los objetos de Java.
- Model-view-controller: Contiene su propio framework que contiene extensiones y components para crear aplicaciones y servicios web.
- Remote Access framework: Soporta exportación e importación de objetos Java a través de la red utilizando diferentes protocolos.
- Convention-over-configuration: Una solución de rápido desarrollo de aplicaciones empresariales basadas en Spring utilizando el módulo Spring Roo.
- Batch processing: Un framework para procesar un alto volumen de información, incluye varios componentes de apoyo.
- Authentication and authorization: Procesos de seguridad configurables que soportan varios estándares, protocolos, herramientas y prácticas de seguridad.

- Remote Management: Configuración de la administración de los objetos java en forma local o remota.
- Messaging: Configuración de escuchas de mensajes para un fácil consumo de mensajes utilizando la librería JMS.
- Testing: Soporta clases para escribir pruebas unitarias y pruebas de integración.

Hibernate

Hibernate es una herramienta de Mapeo Objeto/Relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos o anotaciones en las entidades que permiten establecer estas relaciones.

Caracteristicas:

- Mapeo: Hibernate permite el mapeo de Objetos a modelos relacionales a través del uso de archivos de configuración o con anotaciones directamente en la clase.

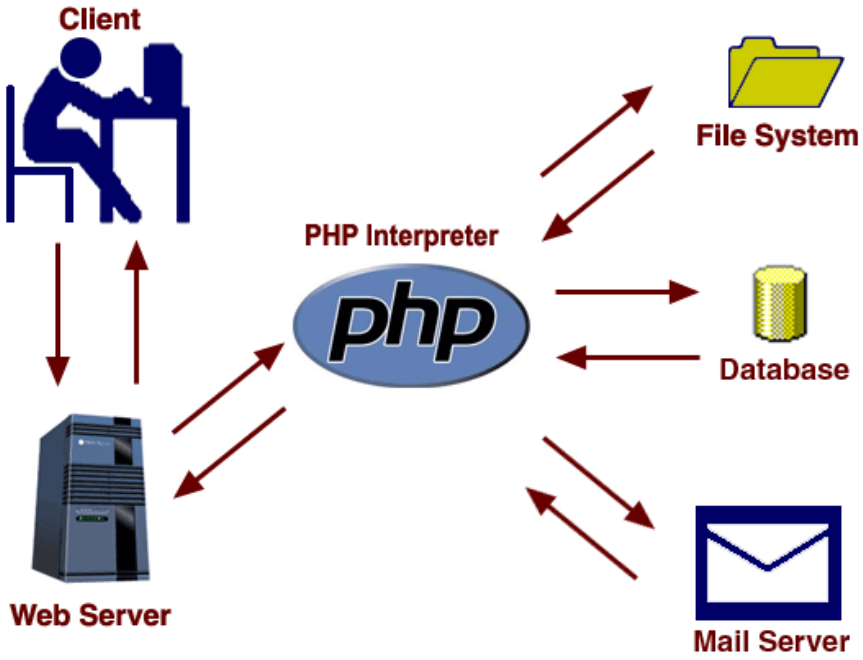
- Persistencia. Hibernate se ocupa de ayudar a una aplicación a lograr la persistencia. Persistencia significa que los datos de una aplicación sobrevivan al proceso de solicitud. En términos de Java se busca que el estado de los objetos viva más allá del alcance de la máquina virtual para que ese mismo estado esté disponible más adelante.

- Hibernate Query Language: Hibernate implementa un lenguaje inspirado en SQL que permite consultas sobre objetos de forma similar a las consultas SQL.

PHP

PHP (Hipertext PreProcesor) es un lenguaje de script de propósito general muy utilizado en el desarrollo de aplicaciones web con contenido dinámico. Desarrollado por Rasmus Lerdorf en 1994, fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML.

La diferencia con los lenguajes del lado del cliente como JavaScript es que el código es ejecutado en el servidor, generando código HTML que es enviado al cliente. El cliente recibirá los resultados de la ejecución del script pero de forma transparente, sin conocer el código tras bambalinas. Ver figura 25.



Anexos. Figura 26

Existen 3 áreas principales en las cuales PHP es usado.

- *Programación del lado del servidor.* Este es el campo tradicional y principal de PHP. Se necesitan tres cosas para realizar esta función, un parser de PHP, un servidor web y un navegador Web. Se necesita estar corriendo el servidor web, con un parser o instalación de PHP, de esta forma se puede acceder al resultado del programa PHP con un navegador Web el cual navega por las páginas Web que sirve el Servidor Web.

- *Programación por línea de comandos.* Se pueden crear programas en PHP que corran sin servidor web o navegador, en este caso solo se necesita el parser de PHP.

- *Aplicaciones de escritorio.* Usando una extensión de PHP (PHP-GTK) es posible crear aplicaciones de escritorio, aunque PHP no es el mejor lenguaje para realizar este tipo de desarrollos.

Características.

- Puede ser usado en los principales Sistemas Operativos.
- PHP es soportado por la mayoría de los Servidores Web actuales.
- PHP se puede usar de forma procedural, con programación orientada a objetos (OOP) o una combinación de ambos.

- PHP fue pensado para ser un lenguaje de fácil aprendizaje.
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- PHP no se limita a generar solamente HTML, también puede generar imágenes, PDF's, películas Flash, texto, XML.
- PHP soporta un amplio rango de bases de datos, así como la forma en que se conecta a ellas, puede ser con extensiones específicas de la base (MySQL), usando una capa de abstracción(PDO) o un estándar de conexión como ODBC.
- PHP se puede conectar con otros servicios usando protocolos como LDAP, IMAP, HTTP, COM, POP3 y otros.
- PHP puede abrir sockets de conexión de redes para utilizar otros protocolos.
- Múltiples extensiones que aportan mucha funcionalidad a PHP.

Visual Basic 6

Es un lenguaje de programación dirigido por eventos de tercera generación desarrollado por Microsoft lanzado en su primera versión en 1991.

Visual Basic contiene un Entorno de Desarrollo Integrado o IDE que integra un editor de textos para la edición del código fuente, un depurador, un compilador y un editor de Interfaces Graficas de Usuario.

Visual Basic deriva del lenguaje BASIC y fue diseñado para ser relativamente fácil de aprender y usar.

Características.

Los compiladores de Visual Basic generan código que requiere una o más librerías de enlace dinámico para que funcione, conocidas comúnmente como DLL (sigla en inglés de dynamic-link library); en algunos casos reside en el archivo llamado MSVBVMxy.DLL (siglas de "MicroSoft Visual Basic Virtual Machine x.y", donde x.y es la versión) y en otros en VBRUNXXX.DLL ("Visual Basic Runtime X.XX"). Estas bibliotecas DLL proveen las funciones básicas implementadas en el lenguaje, conteniendo rutinas en código ejecutable que son cargadas *bajo demanda* en tiempo de ejecución. Además de las esenciales, existe un gran número de bibliotecas del tipo DLL con variedad de funciones, tales como las que facilitan el acceso a la mayoría de las funciones del sistema operativo o las que proveen medios para la integración con otras aplicaciones.

Dentro del mismo Entorno de desarrollo integrado (IDE) de Visual Basic se puede ejecutar el programa que esté desarrollándose, es decir en modo intérprete (en

realidad pseudo-compila el programa muy rápidamente y luego lo ejecuta, simulando la función de un intérprete puro). Desde ese entorno también se puede generar el archivo en código ejecutable (exe); ese programa así generado en disco puede luego ser ejecutado sin requerir del ambiente de programación (incluso en modo stand alone), aunque sí será necesario que las librerías DLL requeridas por la aplicación desarrollada se encuentren también instaladas en el sistema para posibilitar su ejecución.

El propio Visual Basic provee soporte para empaquetado y distribución; es decir, permite generar un *módulo instalador* que contiene al programa ejecutable y las bibliotecas DLL necesarias para su ejecución. Con ese módulo la aplicación desarrollada se distribuye y puede ser instalada en cualquier equipo (que tenga un sistema operativo compatible).

Así como bibliotecas DLL, hay numerosas aplicaciones desarrolladas por terceros que permiten disponer de variadas y múltiples funciones, incluso mejoras para el propio Visual Basic; las hay también para el empaquetado y distribución, y hasta para otorgar mayor funcionalidad al entorno de programación (IDE).

Maven

Crear una aplicación Java requiere diferentes operaciones:

- Generar el código y sus recursos.
- Compilar las clases generadas y sus pruebas.
- Ejecutar las pruebas.
- Empaquetar todo el código con todas sus librerías en un único paquete.

Maven es una herramienta de software para la gestión y construcción de proyectos Java que nos ayuda a optimizar todas las tareas inherentes a la construcción de una aplicación. Maven utiliza un archivo descriptor del proyecto (POM) para configurar el proyecto, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos.

Funcionamiento.

El motor incluido en su núcleo puede dinámicamente descargar las dependencias o plugins del proyecto desde un repositorio, dicho repositorio puede ser público o privado.

La filosofía general de Maven es la estandarización de las construcciones generadas con el fin de utilizar modelos ya existentes para la construcción de software.

Maven está construido alrededor de la idea de la reutilización de la lógica de construcción. Como los proyectos generalmente se construyen en patrones similares, una elección lógica podría ser reutilizar los procesos de construcción.

Ciclo de vida y tareas de construcción.

Maven define un ciclo de vida para la construcción de aplicaciones las cuales tienen que pasar las siguientes fases:

1. process-resources
2. compile
3. process-test-resources
4. test-compile
5. test
6. package
7. install
8. deploy

Cuando se ejecuta un comando Maven llamando una de las fases, Maven irá verificando todas las fases del ciclo de vida desde la primer fase hasta la que fue llamada.

MySQL

Una base de datos es una colección estructurada de registros o información organizada de tal forma que la información pueda ser buscada fácilmente y obtenida rápidamente.

Para acceder y administrar esta información se utilizan sistemas manejadores de bases de datos y MySQL es probablemente el manejador de base de datos más popular, las razones van desde su buen rendimiento, su gran escalabilidad y que se distribuye bajo la licencia GNU GPL de software libre.

Un manejador de base de datos (en inglés *database management system*, abreviado DBMS) es un software dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Los distintos objetivos con los que debe de cumplir un manejador de base de datos son los siguientes:

- Abstracción de la información. Los DBMS ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si

una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios *niveles de abstracción*.

- Independencia. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- Consistencia. En aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea. Por otra parte, la base de datos representa una realidad determinada que tiene ciertas condiciones, por ejemplo que los menores de edad no pueden tener licencia de conducir. El sistema no debería aceptar datos de un conductor menor de edad. En los DBMS existen herramientas que facilitan la programación de este tipo de condiciones.
- Seguridad. La información almacenada en una base de datos puede llegar a tener un gran valor. Los DBMS deben garantizar que esta información se encuentra segura de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- Manejo de transacciones. Una transacción es una unidad de ejecución en un programa que accede y posiblemente actualiza varios elementos de datos. Los DBMS proveen mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.
- Tiempo de respuesta. Lógicamente, es deseable minimizar el tiempo que el DBMS demora en proporcionar la información solicitada y en almacenar los cambios realizados.

Algunas de las características disponibles en MySQL se encuentran:

- Amplio subconjunto del lenguaje ANSI SQL 99. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Procedimientos almacenados.
- Manejo de triggers (o disparadores).
- Recorrido de registros por medio de cursores.
- Vistas actualizables.
- Información del esquema de las tablas.
- Modo estricto el cual asegura que MySQL no trunca o modifica los datos cuando un valor no compatible es insertado.
- Soporte para transacciones distribuidas.

- Motores de almacenamiento independientes.
- Motores de almacenamiento en cluster.
- Soporte SSL.
- Consulta en Cache.
- SELECT’s anidados.
- Soporte para replicación con un maestro por esclavo o muchos esclavos por maestro.
- Indexación y búsqueda Full-Text en el motor MyISAM.
- Librería de base de datos embebida.
- Soporte UNICODE.
- Propiedades ACID en las transacciones al usar motores de almacenamiento que las soporten.
- Arquitectura Shared-nothing en la clusterización a través de MySQL Cluster.
- Respaldos en caliente bajo ciertas condiciones.
- Múltiples motores de almacenamiento permitiendo elegir el más eficaz para cada tabla de la aplicación.
- Agrupación de commits, reuniendo múltiples transacciones de varias conexiones para aumentar el número de confirmaciones por segundo.