"Te Atiendo" Plataforma integral de atención a la población del municipio de Rionegro”

Especialidad: Desarrollo de Aplicaciones de Software Libre

Autor: Juan Gabriel Galeano Arenas
Consultor: Gregorio Robles Martinez
Tutor Externo: Osvaldo Andres Cifuentes Restrepo
Fecha: 09/01/2016

Resumen Ejecutivo

Uno de los pilares fundamentales para el buen desarrollo y funcionamiento de una organización se encuentra en la gestión y direccionamiento de los flujos de información y conocimiento.

En la labor de procurar calidad y pertinencia en la concepción y prestación de los servicios misionales de la entidad, el Municipio de Rionegro decide implementar una plataforma integral de atención a la población del municipio, esto cobra vital importancia en la medida que la gestión de documentos, comunicaciones, acciones y procesos de conocimiento, se soportan en los sistemas informáticos; con la consideración de que dicha gestión debe corresponder a una dinámica que cubra no sólo los procesos administrativos y misionales , sino también los procesos alternos, receptores prioritarios de estos avances.

Uno de los propósitos de la secretaría de gestión y protección social, es la búsqueda de la dirección sus estrategias hacia el fortalecimiento de la plataforma tecnológica institucional respondiendo a los siguientes factores:

Optimización de los procesos administrativos, misionales:

La buena gestión y la eficiencia en los procesos, depende en gran medida de la eficiencia en los procesos administrativos buscando mejorar considerablemente la gestión de:

1. Sistema de gestión de procesos de negocio
2. Control y gestión de la información

Analizando detalladamente los procesos que se quieren sistematizar en la secretaría de la gestión y protección social se encontró lo siguiente:

1. **Sistema de gestión de procesos de negocio**: La secretaria requiere contar con una solución que les permita generar flujos de trabajo, donde se puedan gestionar la productividad entre las personas, sus procesos y los sistemas, que permita tener eficiencia, efectividad, agilidad y control**.**

   Uno de los principales retos de la secretaria es conseguir la flexibilidad y agilidad necesarias para adaptarse a las necesidades del entorno digital e investigación de los diferentes entes públicos y privados que se atienden, gestionando los riesgos operacionales y financieros, incrementando la calidad en los expedientes que se generan y la satisfacción de los clientes.

Este reto no depende solo la tecnología que se utilice, sino del conocimiento, dominio y mejora de los procesos, datos y recursos institucionales.

La gestión de procesos es cada vez más prioritaria en las diferentes empresas, la secretaria busca una agilidad empresarial, que optimice los procesos de negocio, que controle los riesgos operativos, que gestione los recursos y se encamine hacia el cumplimento de los objetivos de la institución.

Esta herramienta debe permitir la identificación, modelización, análisis, ejecución, control y mejora de los procesos de negocio y optimizar los procesos para la mejora continua

Los problemas más grandes que se requieren remediar con la herramienta son:

Las complejas relaciones entre políticas, procedimientos y actividades manuales entre personas sistemas y datos.

La Información relevante para el accionar de la secretaria reside en diferentes lugares, haciendo que el manejo sea difícil.

Poca documentación de los procesos y procedimientos que permitan verificar las interacciones, contradicciones y cumplimiento de los mismos

Muchas tareas manuales, imposibilitando identificar los cuellos de botella en los procesos, dado que es difícil identificar quién debe de realizar las tareas.

Ineficiencia de los procesos por falta de una clara responsabilidad sobre los mismos

Considerable inconsistencia en los tiempos de respuesta de los funcionarios para completar las tareas habituales.

Falta de medidas adecuadas para medir el rendimiento de los funcionarios

2. **Control y Gestión de la información:** Permite guiar la gestión de la secretaria hacia los objetivos del gobierno, dejando en firme un instrumento para evaluar la gestión.

En el presente trabajo de tesis se presenta una arquitectura de software para aplicaciones Web en donde se sigue un proceso de ingeniería de software. En este desarrollo, la arquitectura se descompone mediante distintas vistas o enfoques tales como, la vista lógica, la vista de procesos, la vista de desarrollo, el resultado final del proyecto de software es un producto que toma forma durante su desarrollo, gracias a la intervención de muchos tipos distintos de personas se obtendrá un sistema que supla las necesidades.

Contenido

Índice de graficas

Índice de tablas

Capítulo I

Introducción

Debido al gran éxito que ha tenido la WWW (World Wide Web) el desarrollo de aplicaciones Web ha crecido de forma notable abarcando áreas como comercio electrónico, redes sociales, banca en línea, entretenimiento, juegos, etc. La mayoría de los grandes sistemas de información han tenido que ser trasladados a ambientes Web como parte de su evolución. Los sistemas de comercio que antes atendían pequeñas localidades ahora están encargados de atender a todo el mundo en una organización

Debido a que Internet es un mercado muy demandante nos encontramos con la necesidad de construir aplicaciones Web más complejas y en un tiempo muy reducido. Además de que dichas aplicaciones necesitan cumplir con requisitos de calidad como son rendimiento, usabilidad, escalabilidad, mantenimiento, accesibilidad, etc. Esto conduce a que el desarrollo de aplicaciones Web tenga una probabilidad muy alta de fracasar.

Cada día las entidades públicas requieren el fortalecimiento de las instancias administrativas y su medición con datos cuantitativos (números reales en tiempo real), un punto de un buen gobierno es medir los logros o beneficios que se les ha dado a la población o a un grupo de interés, es un método de articulación eficiente y eficaz que es misional dentro de una administración moderna, dichos procesos son automatizados con sistemas de información (software) que generen registros de cada persona, familia o grupo de interés (barrio, JAC) en todas las dependencias.

Actualmente la secretaría de gestión y protección social viene adelantando esta labor de acompañamiento a diferentes beneficiarios con los programas de protección y/o bienestar social, para registrar dichos beneficios y poder llegar a la población más vulnerable se incorporó el software “TE ATIENDO”, esto ha implicado varios cambios organizativos que han ayudado al proceso misional generando alertas de atención e indicadores de gestión tangibles que son visibles dentro de administración municipal actual ya que muestras en los consejos de gobierno datos reales que son comprobables con consultas dentro del software.

¿Qué es te atiendo?

Es un aplicativo especializado en entidades públicas que permite registrar todos los beneficios que son asignados a una persona, grupo familiar o grupo social. Facilitando así la generación de indicadores de gestión en todas sus dependencias, es personalizado a las necesidades de cada uno de nuestros clientes por medio de los flujos de procesos (Wokflow)

1.1 Motivación y Justificación

La motivación principal de este trabajo de tesis consiste en modelar una arquitectura de software de aplicaciones basadas en la web de un proyecto que se tiene formulado con la alcaldía del Municipio de Rionegro, los modelos actuales de modelado y análisis de software han venido transformándose para ser capaces de representar los

requerimientos del software y que de este modelado sea posible construir el software de forma confiable.

La arquitectura de software es una pieza central del desarrollo de sistemas de software modernos. El objetivo de la arquitectura consiste en desarrollar sistemas de software grandes de forma eficiente, estructurada y con capacidad de reuso. La arquitectura forma parte del proceso de diseño de software el cual también forma parte del proceso de desarrollo de software que comprende, requerimientos, diseño, implementación, prueba y mantenimiento.

El proceso de planificación de todo proceso de software debe hacerse partiendo de una estimación del trabajo a realizar. Para obtener software de calidad es necesario medir el proceso de desarrollo, cuantificar lo que se ha hecho y lo que falta por hacer, estimar el tamaño del proyecto, costos, tiempo de desarrollo, control de calidad, mejora continua y todo lo relacionado al ciclo de vida utilizado en el desarrollo del proyecto.

La aplicación debe de tener un enfoque cuantificable la cual es una tarea compleja que requiere disciplina, conocimiento, y estudio de indicadores adecuados para los diferentes objetivos de la medición y evaluación del desarrollo del proyecto de software con el fin que cumpla con los mayores estándares de calidad y permita en un futuro la escalabilidad de la implementación.

El estudio de la arquitectura de software se centra en la forma en como son diseñados y construidos los sistemas de software. La arquitectura de un sistema de software define la estructura del sistema y de todos los componentes que involucran al sistema así como sus interconexiones.

Los lenguajes orientados a objetos me han permitido definir a los componentes de software elevando su nivel de abstracción, lo cual me ha ayudado a construir componentes más complejos. A partir de este tipo de lenguajes ha sido posible la utilización de patrones arquitecturales.

1.2 Objetivos

Objetivo General

Esta tesis tiene como objetivo principal la propuesta y desarrollo de un sistema de información que permita ingresar información de beneficios otorgados a la población por sus diferentes procesos y programas.

Objetivos específicos

- ✓ Medir a cada dependencia y/o programa en cuanto a número de atenciones o beneficios aplicados a la población.
- ✓ Determinar con exactitud que se la ha entregado a una persona o familia en un periodo de tiempo.
- ✓ Medir costo/beneficio de cada uno de los programas de las dependencias dentro del municipio.
- ✓ Tener información en línea o en tiempo real para generar indicadores de gestión en cualquier momento para cualquier entidad (contraloría, procuraduría, gobernación, etc).

- ✓ Estudiar y analizar patrones de diseño. El objetivo es la integración a arquitecturas web.
- ✓ Utilizar el lenguaje unificado UML para la documentación de la Arquitectura del Software.

1.3 Organización de la tesis

El resto del documento está organizado de la siguiente manera. En el capítulo 2 se presentan los antecedentes teóricos de la arquitectura de software sobre los cuales está fundamentado nuestro trabajo. Después en el capítulo 3 realizaremos un profundo análisis sobre las aplicaciones Web que nos permiten comprender mejor los retos de la tesis y la arquitectura de software que utilizaremos para desarrollar la aplicación “TE ATIENDO”. El capítulo 4 presenta el caso de estudio que utilizaremos para validar el trabajo realizado en los capítulos anteriores, por último en el capítulo 5 presentaremos las conclusiones y proponemos el trabajo futuro de nuestro trabajo.

Capítulo II

Ingeniería del Software

En este capítulo se describen los conceptos relacionados a las arquitecturas de software; así mismo, enfatizamos la necesidad de comprender los requerimientos y planificar el desarrollo para reducir costos, tiempos, y esfuerzos durante y después del desarrollo. Se realizara una pequeña introducción sobre los temas relacionados, con la arquitectura de software y las técnicas utilizadas para el desarrollo de las tesis.

2.1 Modelos de la ingeniería de software

La ingeniería de software es la disciplina de la ingeniería que estudia todos los aspectos relacionados con la producción de software desde la especificación del sistema, hasta la implantación, mantenimiento y la evolución del software [1].

La ingeniería de software cuenta con un conjunto de actividades y resultados asociados para obtener un producto de software, el cual se conoce como proceso de software, un proceso de software considera cuatro actividades fundamentales como especificación, desarrollo, validación y evolución

Cada sistema requiere un tipo de desarrollo diferente que corresponda al domino de su aplicación. Por lo tanto se necesita contar con modelos que permitan describir de forma simplificada los procesos de software. Un modelo de proceso de software es una representación abstracta de un proceso de software, el cual en su mayoría, está basado en modelos de desarrollo de software tales como el enfoque en cascada, el desarrollo interactivo y el desarrollo basado en componentes.

El primer modelo de software publicado fue el modelo en cascada, el cual incluye las actividades fundamentales del proceso de desarrollo de software En este modelo para comenzar con la siguiente actividad se necesita haber terminado o pasado por una de las anteriores.

Una de las desventajas del modelo en cascada es la necesidad de terminar las etapas anteriores para comenzar con la siguiente, dando como resultado un tiempo de desarrollo más largo.

El desarrollo iterativo acorta las etapas, de tal forma que se puede desarrollar una versión reducida del sistema en muy poco tiempo, permitiendo utilizar el sistema y analizar las mejoras para las siguientes versiones. Cada iteración está compuesta de un modelo en cascada reducido y contiene pocos requerimientos. En cada iteración el sistema debe evolucionar hasta convertirse en el sistema que cubra las necesidades del cliente. El enfoque interactivo o permite una interacción directa con todas las personas involucradas con el proyecto (stakeholders), permitiendo entender de mejor forma los requerimientos del sistema. El desarrollo basado en componentes (Figura 2.1) incorpora muchas de las características del modelo iterativo, sigue la metodología de ensamblar componentes y escribir código para hacer que estos componentes

funcionen dentro del proyecto de software. Un componente es un desarrollo de software previamente construido que encapsula una funcionalidad específica y cuenta con interfaces estándar que permiten su uso. Es evolutivo por naturaleza y exige un enfoque de software interactivo para la creación del software. Este modelo permite la reutilización del software a gran escala, abstrae el diseño del sistema permitiendo un rendimiento superior.

Figura 2.1: Ciclo de vida de software basado en componentes

Para la implementación de esta tesis se parte de la arquitectura de software donde se abarcan decisiones importantes sobre:

- La organización del sistema de software
- Los elementos estructurales que compondrán el sistema y sus interfaces, junto con sus comportamientos, tal y como se especifican en las colaboraciones entre estos elementos.
- La composición de los elementos estructurales y del comportamiento en subsistemas progresivamente más grandes.
- El estilo de la arquitectura.

La arquitectura de software está afectada no solo por la estructura y el comportamiento, sino también por el uso, la funcionalidad, el rendimiento, la flexibilidad, la reutilización, la facilidad de comprensión, las restricciones y compromisos económicos y tecnológicos, y la estética.

Para alcanzar un buen desarrollo de la aplicación TE ATIENDO, nos basamos en dos modelos de desarrollo de software el proceso iterativo e incremental y el proceso de reutilización de componentes.

Un proceso de desarrollo de software debe de tener unas secuencias de hitos claramente articulados que proporcionen los criterios que se necesitan para autorizar el paso de la fase a la siguiente dentro del ciclo de vida del producto.

El desarrollo de este software estará dirigido por los casos de uso y centrado en la arquitectura. Estos aspectos tienen un impacto técnico evidente sobre el producto del proceso. El estar dirigido por los casos de uso significa que cada fase en el camino del producto final estará relacionada con lo que los usuarios hacen realmente. El estar centrado en la arquitectura significa que el trabajo de desarrollo se centra en obtener el patrón de la arquitectura que dirigirá la construcción del sistema en las primeras fases, garantizando un proceso continuo no solo para esta primera versión en curso del producto, sino para la vida entera del mismo.

Se toma la iniciativa de un desarrollo interactivo e incremental para obtener un software mejor, para cumplir con los hitos principales y secundarios con los cuales controlamos el desarrollo, adicional permite lo siguiente:

- Tomar las riendas de los riesgos críticos y significativos dese el principio.
- Poner en marcha una arquitectura que guie el desarrollo del software
- Proporcionar un marco de trabajo que gestione de mejor forma los inevitables cambios de los requisitos.
- Construir el sistema a lo largo del tiempo en lugar de hacerlo de una sola vez cerca del final, cuando el cambiar algo se ha vuelto costoso.
- Proporcionar un proceso de desarrollo a través del cual el personal puede trabajar de manera más eficaz.

Uno de los temas a los que más atención le prestamos en el desarrollo de la aplicación es a la atenuación del riesgo “El riesgo es inherente en el empleo de los recursos disponibles para las expectativas futuras” [2]. Lo que necesita la disciplina del software, como escribió Barry Boehm es un modelo de proceso que “cree una aproximación al proceso de desarrollo de software dirigida por los riesgos en lugar de un proceso fundamentalmente dirigido por los documentos o dirigido por el código” [3]. El proceso unificado cumple con estos criterios porque trata los riesgos importantes en las dos primeras fases, inicio y elaboración, y cualquier riesgo restante al principio de la fase de la construcción (Figura 2.2), por orden de importancia. Identifica, gestiona y reduce los riesgos en las primeras fases mediante las iteraciones, lo que permite no poner en riesgo el proyecto y mitigar los riesgos.

Figura 2.2: Atenuación del Riesgo en el desarrollo iterativo

2.2 Las iteraciones sobre el ciclo de vida

Las iteraciones difieren marcadamente en las diferentes fases del ciclo del desarrollo, como consecuencia de que los desafíos, que se afrontan en cada fase son diferentes.

Una iteración es un miniproyecto, un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales, que obtiene como resultado una versión interna.

Se cuenta cinco flujos de trabajo fundamentales: requisitos, análisis, diseño, implementación y prueba que sirven para ayudarnos a describir los flujos de trabajo de la iteración.

Las primeras iteraciones se centran en la comprensión del problema y de la tecnología. En la fase de inicio, las iteraciones se preocupan de producir un análisis del negocio. En la fase de elaboración, las iteraciones se orientan al desarrollo de la línea base de la arquitectura. En la fase de construcción, las iteraciones se dedican a la construcción del producto por medio de una seria de construcciones dentro de cada iteración, que acaban en un producto preparado para su distribución a la comunidad de usuarios. Como se muestra en la figura 2.3, todas las iteraciones siguen el mismo patrón.

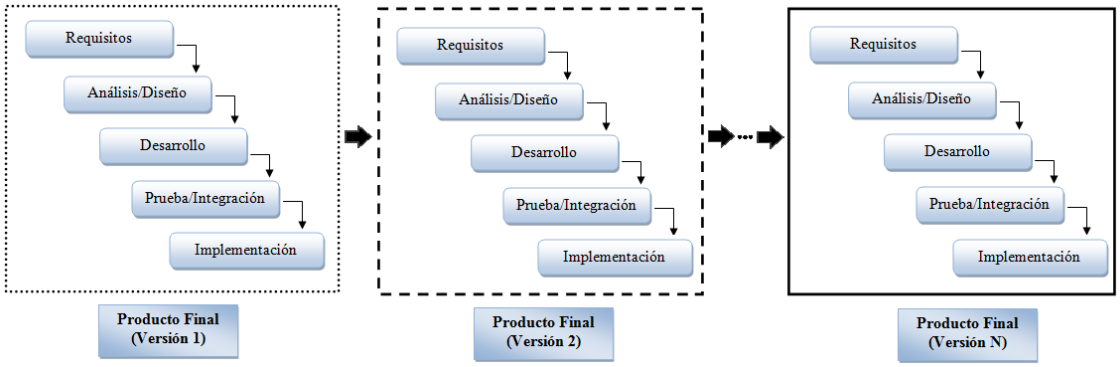


Figura 2.3: Patrones de la iteraciones

Cada iteración se analiza cuando termina. Uno de los objetivos es determinar si han aparecido nuevos requisitos o han cambiado los existentes, afectando las iteraciones siguientes. Durante la planificación de los detalles de la siguiente iteración, se examina como afectaran los riesgos que aún quedan al trabajo en curso, antes de finalizar una iteración, debemos de estar seguros de que no hemos estropeado ninguna otra parte del sistema que funcionaba en anteriores iteraciones.

Al final de una iteración, el conjunto de modelos que representa al sistema queda en un estado concreto. Llamamos a este estado línea base. En un momento dado de la

secuencia de iteraciones, estarán terminados algunos subsistemas, estos contendrán toda la funcionalidad requerida, y estarán implementados y probados. Por tanto un incremento es la diferencia entre dos línea base sucesiva.

Cada una de las fases termina con un hito principal, como se muestra en la figura 2.4 [4]

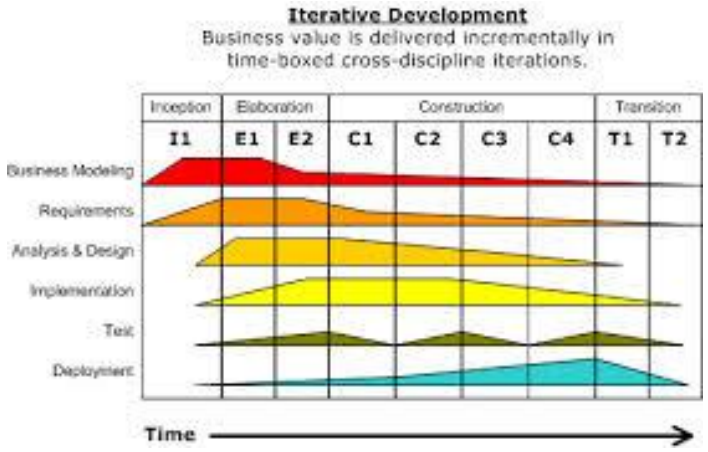


Figura 2.4: Hitos

1. Inicio: objetivos del ciclo de vida
2. Elaboración: arquitectura del ciclo de vida
3. Construcción: funcionalidad operativa inicial
4. Transición: versión del producto.

El objetivo de cada hito principal es garantizar que los diferentes modelos de flujo de trabajo evolucionan de manera equilibrada durante el ciclo de vida del producto.

Los objetivos fundamentales de la *fase de inicio* son el establecimiento del ámbito de lo que debería de hacer el producto, la reducción de los peores riesgos, y la preparación del análisis del negocio inicial, que indique que merece la pena realizar el proyecto desde la perspectiva del negocio.

Los objetivos fundamentales de la *fase la elaboración* son obtener la línea base de arquitectura, capturar la mayoría de los requisitos, y reducir los peores siguientes riesgos, es decir establecer la arquitectura del ciclo de vida. Al final de esta fase, somos capaces de estimar los costes y las fechas y de planificar la des de la construcción en algún detalle. En este momento, deberíamos de ser capaces de hacer nuestra apuesta.

Los objetivos fundamentales de la fase de *construcción* son el desarrollo del sistema entero y la garantía de que el producto puede comenzar una transición a los clientes, es decir que tiene una funcionalidad operativa inicial.

El objetivo principal de la fase de *transición* es garantizar que tenemos un producto preparado para su entrega a la comunidad de usuarios. Durante esta fase se enseña a los usuarios a utilizar el software.

Un proyecto de desarrollo de software puede dividirse aproximadamente en dos trozos, las fases de inicio y elaboración y las fases de construcción y transición. Durante las fases de inicio y elaboración, construimos el análisis del negocio, atenuamos a los peores riesgos, creamos la línea base de la arquitectura, y planificamos el resto del proyecto con una alta precisión.

Después el proyecto pasa a la fase de construcción donde el objetivo es la economía de escala, se desarrolla el grueso de la funcionalidad del sistema construyendo sobre la arquitectura cuya línea base se obtuvo durante la fase de elaboración Se reutiliza el software existente tanto como es posible

2.3 Diseño arquitectónico y tecnologías usadas

La arquitectura se desarrolla mediante las iteraciones, principalmente durante la fase de elaboración. Cada iteración se desarrolla comenzando con los requisitos y siguiendo con el análisis y el diseño, implementación y pruebas, pero centrándonos en los casos de uso relevantes desde el punto de vista de la arquitectura y en otros requisitos. El resultado final de la fase de elaboración es una línea base de la arquitectura, un esqueleto del sistema.

Los casos de uso arquitectónicamente relevantes son aquellos que nos ayudan a mitigar los riesgos más importantes y aquellos que nos ayudan a cubrir todas las funcionalidades más significativas. La implementación, integración y prueba de la línea base de la arquitectura proporciona seguridad al desarrollo

Los casos de uso conducen el desarrollo de la arquitectura, y la arquitectura indica que casos de uso pueden realizasen. Figura 2.5

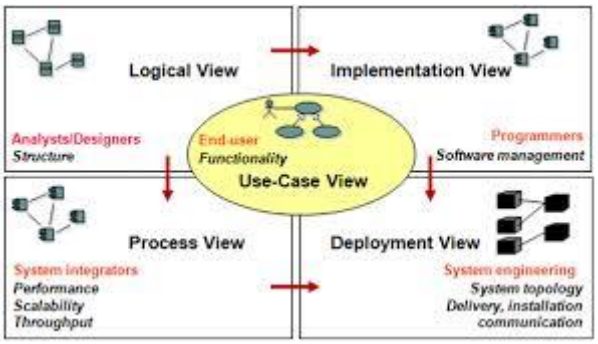


Figura 2.5: Casos de uso y Arquitectura

En la implementación de la arquitectura es indispensable la utilización de patrones de la arquitectura, las ideas del arquitecto Christopher Alexander sobre como los “Lenguajes de patrones” se utilizan para sistematizar principios y practicas importantes en el diseño de edificaciones y comunidades, han inspirado a muchos miembros de la comunidad de la orientación a objetos a definir coleccionar y probar una gran variedad de patrones de software [5] “La comunidad de patrones” define un patrón como “ una solución a un problema de diseño que aparecen con frecuencia”. Muchos de los

diseños de patrones están documentados en libros y en la página web de los desarrolladores, que presentan los patrones utilizando plantillas estándar. Estas plantillas asignan un nombre a un patrón y presentan un resumen de los problemas y las fuerzas que los hacen surgir, una solución en términos de colaboración de clases participantes e interacción entre objetos de esas clases. Según Alexander, sería bueno que los ingenieros de software aprendiesen los nombres y el objetivo de muchos patrones estándar y que los aplicasen para hacer diseños mejores y más comprensibles.

En la ejecución del proyecto de software implementaremos, tecnologías y patrones que facilitaran el desarrollo y la calidad de este dentro de estas encontramos:

Patrones de Diseño en Java

Un patrón de diseño es una abstracción de una solución en un nivel alto. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación. [6]

Muchos diseñadores y arquitectos de software han definido el término de patrón de diseño de varias formas que corresponden al ámbito a la cual se aplican los patrones. Luego, se dividió los patrones en diferentes categorías de acuerdo a su uso.

Los diseñadores de software extendieron la idea de patrones de diseño al proceso de desarrollo de software. Debido a las características que proporcionaron los lenguajes orientados a objetos (como herencia, abstracción y encapsulamiento) les permitieron relacionar entidades de los lenguajes de programación a entidades del mundo real fácilmente, los diseñadores empezaron a aplicar esas características para crear soluciones comunes y reutilizables para problemas frecuentes que exhibían patrones similares.

Fue por los años 1994, que apareció el libro "Design Patterns: Elements of Reusable Object Oriented Sofware" escrito por los ahora famosos Gang of Four (GoF, que en español es la pandilla de los cuatro) formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Ellos recopilaron y documentaron 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos.

Groovy Lenguaje

Groovy es un lenguaje de programación orientado a objetos implementado sobre la plataforma Java. Tiene características similares a Python, Ruby, Perl y Smalltalk. La especificación JSR 241 se encarga de su estandarización para una futura inclusión como componente oficial de la plataforma Java.[7]

Groovy usa una sintaxis muy parecida a Java, comparte el mismo modelo de objetos, de hilos y de seguridad. Desde Groovy se puede acceder directamente a todas las API existentes en Java. El bytecode generado en el proceso de compilación es totalmente compatible con el generado por el lenguaje Java para la Java Virtual Machine (JVM), por tanto puede usarse directamente en cualquier aplicación Java. Todo lo anterior

unido a que la mayor parte de código escrito en Java es totalmente válido en Groovy hacen que este lenguaje sea de muy fácil adopción para programadores Java; la curva de aprendizaje se reduce mucho en comparación con otros lenguajes que generan bytecode para la JVM, tales como Jython o JRuby. Groovy puede usarse también de manera dinámica como un lenguaje de scripting.

Groovy 1.0 apareció el 2 de enero de 2007. Después de varias versiones beta y otras tantas candidatas a release, el 7 de diciembre de 2007 apareció la versión Groovy 1.1 que finalmente fue renombrada a Groovy 1.5 con el fin de notar la gran cantidad de cambios que ha sufrido con respecto a la versión 1.0. En diciembre de 2009 se publicó la versión 1.7.

Spring-framework

Spring es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.2 [8]

La primera versión fue escrita por Rod Johnson, quien lo lanzó junto a la publicación de su libro Expert One-on-One J2EE Design and Development (Wrox Press, octubre 2002). El framework fue lanzado inicialmente bajo la licencia Apache 2.0 en junio de 2003. El primer gran lanzamiento fue la versión 1.0, que apareció en marzo de 2004 y fue seguida por otros hitos en septiembre de 2004 y marzo de 2005. La versión 1.2.6 de Spring Framework obtuvo reconocimientos Jolt Awards y Jax Innovation Awards en 2006.3 4 Spring Framework 2.0 fue lanzada en 2006, la versión 2.5 en noviembre de 2007, Spring 3.0 en diciembre de 2009, y Spring 3.1 dos años más tarde. El inicio del desarrollo de la versión 4.0 fue anunciado en enero de 2013. La versión actual es 4.1.1.

Si bien las características fundamentales de Spring Framework pueden ser usadas en cualquier aplicación desarrollada en Java, existen variadas extensiones para la construcción de aplicaciones web sobre la plataforma Java EE. A pesar de que no impone ningún modelo de programación en particular, este framework se ha vuelto popular en la comunidad al ser considerado una alternativa, sustituto, e incluso un complemento al modelo EJB (Enterprise JavaBean).

Spring Data JPA

Spring Data es un proyecto de SpringSource cuyo propósito es unificar y facilitar el acceso a distintos tipos de tecnologías de persitencia, tanto a bases de datos relacionales como a las del tipo NoSQL.[9]

Spring ya proporcionaba soporte para JDBC, Hibernate, JPA, JDO o MyIbatis, simplificando la implementación de la capa de acceso a datos, unificando la configuración y creando una jerarquía de excepciones común para todas ellas.

Y ahora, Spring Data viene a cubrir el soporte necesario para distintas tecnologías de bases de datos NoSQL y, además, integra las tecnologías de acceso a datos tradicionales, simplificando el trabajo a la hora de crear las implementaciones concretas.

Con cada tipo de tecnología de persistencia los DAOs (Data Access Objects) ofrecen las funcionalidades típicas de un CRUD (Create-Read-Update-Delete) para objetos de dominio propios, métodos de búsqueda, ordenación y paginación. Spring Data proporciona interfaces genéricas para estos aspectos (CrudRepository, PagingAndSortingRepository) e implementaciones específicas para cada tipo de tecnología de persistencia

MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.1 MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.[10]

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

JQuery

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC. jQuery es la biblioteca de JavaScript más utilizada. [11]

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Factory

Consiste en utilizar una clase constructora (al estilo del Abstract Factory) abstracta con unos cuantos métodos definidos y otro(s) abstracto(s): el dedicado a la construcción de objetos de un subtipo de un tipo determinado. Es una simplificación del Abstract

Factory, en la que la clase abstracta tiene métodos concretos que usan algunos de los abstractos; según usemos una u otra hija de esta clase abstracta, tendremos uno u otro comportamiento.[12]

Singleton

El patrón de diseño singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.[13]

Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

El patrón singleton se implementa creando en nuestra clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor (con atributos como protegido o privado).

La instrumentación del patrón puede ser delicada en programas con múltiples hilos de ejecución. Si dos hilos de ejecución intentan crear la instancia al mismo tiempo y esta no existe todavía, sólo uno de ellos debe lograr crear el objeto. La solución clásica para este problema es utilizar exclusión mutua en el método de creación de la clase que implementa el patrón.

Las situaciones más habituales de aplicación de este patrón son aquellas en las que dicha clase controla el acceso a un recurso físico único (como puede ser el ratón o un archivo abierto en modo exclusivo) o cuando cierto tipo de datos debe estar disponible para todos los demás objetos de la aplicación.

El patrón singleton provee una única instancia global gracias a que:

La propia clase es responsable de crear la única instancia.

- ✓ Permite el acceso global a dicha instancia mediante un método de clase.
- ✓ Declara el constructor de clase como privado para que no sea instanciable directamente.

Builder

Como Patrón de diseño, el patrón builder (Constructor) es usado para permitir la creación de una variedad de objetos complejos desde un objeto fuente (Producto), el objeto fuente se compone de una variedad de partes que contribuyen individualmente a la creación de cada objeto complejo a través de un conjunto de llamadas a interfaces comunes de la clase Abstract Builder.[14]

A menudo, el patrón builder construye el patrón Composite, un patrón estructural.

Adapter

El patrón Adapter (Adaptador) se utiliza para transformar una interfaz en otra, de tal modo que una clase que no pudiera utilizar la primera, haga uso de ella a través de la segunda.[15]

MVC

El modelo–vista–controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento [16]

Ajax

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.[17]

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página, aunque existe la posibilidad de configurar las peticiones como síncronas de tal forma que la interactividad de la página se detiene hasta la espera de la respuesta por parte del servidor.

JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

Java Server

JavaServer Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java.[18]

Git

Git (pronunciado "guit") es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o front end como Cogito o StGIT. Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad

plena. Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux.[19]

El mantenimiento del software Git está actualmente (2009) supervisado por Junio Hamano, quien recibe contribuciones al código de alrededor de 280 programadores.

Maven

Maven es una herramienta de software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002. Es similar en funcionalidad a Apache Ant (y en menor medida a PEAR de PHP y CPAN de Perl), pero tiene un modelo de configuración de construcción más simple, basado en un formato XML. Estuvo integrado inicialmente dentro del proyecto Jakarta pero ahora ya es un proyecto de nivel superior de la Apache Software Foundation.[20]

Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.

Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores. Este repositorio y su sucesor reorganizado, el repositorio Maven 2, pugnan por ser el mecanismo de facto de distribución de aplicaciones en Java, pero su adopción ha sido muy lenta. Maven provee soporte no sólo para obtener archivos de su repositorio, sino también para subir artefactos al repositorio al final de la construcción de la aplicación, dejándola al acceso de todos los usuarios. Una caché local de artefactos actúa como la primera fuente para sincronizar la salida de los proyectos a un sistema local.

Maven está construido usando una arquitectura basada en plugins que permite que utilice cualquier aplicación controlable a través de la entrada estándar. En teoría, esto podría permitir a cualquiera escribir plugins para su interfaz con herramientas como compiladores, herramientas de pruebas unitarias, etcétera, para cualquier otro lenguaje. En realidad, el soporte y uso de lenguajes distintos de Java es mínimo. Actualmente existe un plugin para .Net Framework y es mantenido, y un plugin nativo para C/C++ fue alguna vez mantenido por Maven.

Jenkins

Jenkins es un software de Integración continua open source escrito en Java. Está basado en el proyecto Hudson y es, dependiendo de la visión, un fork del proyecto o simplemente un cambio de nombre.[21]

Jenkins proporciona integración continua para el desarrollo de software. Es un sistema corriendo en un servidor que es un contenedor de servlets, como Apache Tomcat. Soporta herramientas de control de versiones como CVS, Subversion, Git, Mercurial, Perforce y Clearcase y puede ejecutar proyectos basados en Apache Ant y Apache

Maven, así como scripts de shell y programas batch de Windows. El desarrollador principal es Kohsuke Kawaguchi. Liberado bajo licencia MIT, Jenkins es software libre.

La estrategia para desarrollar un producto de software en pasos pequeños manejables:

- Planificar un poco
- Especificar, Diseñar, e implementar un poco
- Integrar, probar, y ejecutar un poco cada iteración

Entre cada paso, obtenemos retroalimentación que nos permite ajustar nuestros objetivos para el siguiente paso. Después se da el siguiente paso, y después otro. Cuando se han dado todos los pasos que habíamos definido y planificado, tenemos un producto desarrollado que podemos distribuir a nuestro cliente y usuario.

Un proyecto de desarrollo de software transforma una “delta” (un cambio) de los requisitos de usuario, con el método iterativo e incremental esta adaptación de los cambios se realiza poco a poco. Cada iteración tiene todo lo que tiene un proyecto de desarrollo de software, planificación, desarrollo de una serie de flujos de trabajo (requisitos, análisis y diseño, implementación y prueba), y una preparación para la entrega.

El ciclo de vida iterativo produce resultados tangibles en forma de versiones internas, y cada una de ellas aporta un incremento y demuestra la reducción de los riesgos con los que se relaciona. Estas versiones se presentan al cliente y usuario, en cuyo caso proporcionan una retroalimentación valiosa para la validación del trabajo.

La consecución de una arquitectura robusta en sí misma es el resultado de las iteraciones en las primeras fases. En la fase de inicio, encontramos una arquitectura esencial que satisface los requisitos clave, evita los riesgos críticos y resuelve los problemas de desarrollo principales. En la fase de elaboración establecemos la línea base de la arquitectura que guiara el resto de desarrollo.

Eclipse

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus.[22]

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia pública general de GNU (GNU GPL).

Típicamente los patrones de software se implementan de una forma muy directa en lenguajes orientados a objetos.

Consideramos los patrones de diseño como colaboraciones entre clases e instancias, con su comportamiento explicado en los diagramas de colaboración. Utilizaremos plantillas de colaboración ya que entendemos que las soluciones son bastante generales. Utilizaremos herencia, extensión, implementaciones y otros mecanismos para especializar el patrón.

Los patrones de las arquitecturas se utilizan de una forma parecida pero se centran en estructuras e interrelaciones, entre subsistemas e incluso entre sistemas.

Un patrón que utilizamos para darle orden al desarrollo es el Layers, este tipo de patrón define como organizar el modelo del diseño en capas, lo cual quiere decir que los componentes de una capa solo pueden hacer referencia a componentes a capas inmediatamente inferiores. Este patrón es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas bajas no son conscientes de ningún detalle o interfaz de la superiores. Además nos ayudan identificar que puede reutilizarse, y proporciona una estructura que nos ayuda a tomar decisiones sobre que partes comprar o reutilizar y que partes construir.

Un sistema con una arquitectura en capas pone a los subsistemas de aplicación individuales en lo más alto. Esto se construye a partir de subsistemas en las capas más bajas, como son los marcos de trabajo y las bibliotecas de clases. Observemos la figura 2.6 y la figura 2.7. La capa general de aplicación contiene los subsistemas que no son específicos de una sola aplicación, sino que pueden ser utilizados por muchas aplicaciones diferentes dentro del mismo dominio o negocio. La arquitectura de las dos capas inferiores puede establecerse sin considerar los casos de uso debido a que no son dependientes del negocio.

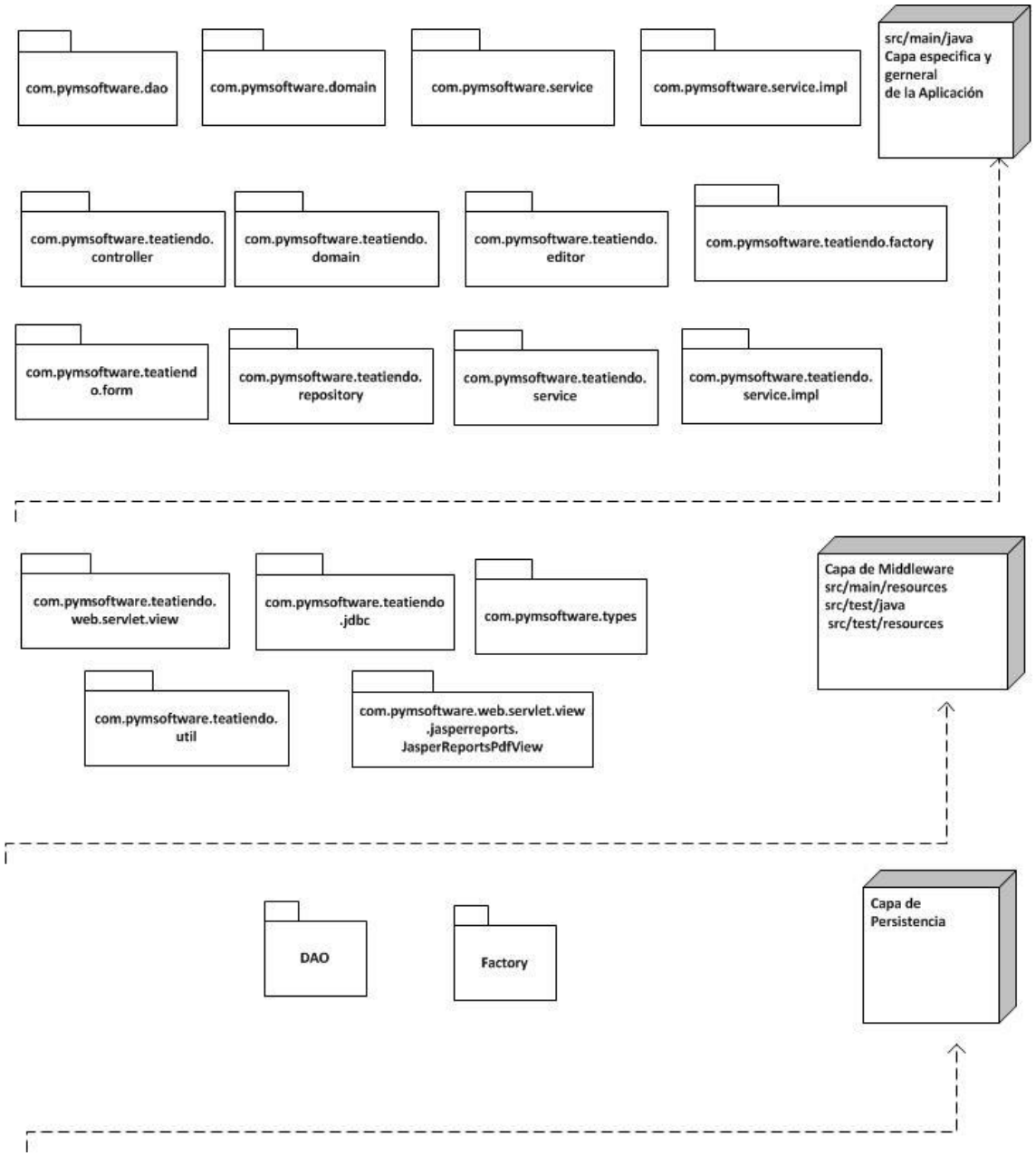


Figura 2.6: Vista de arquitectura de diseño del proyecto Te Atiendo

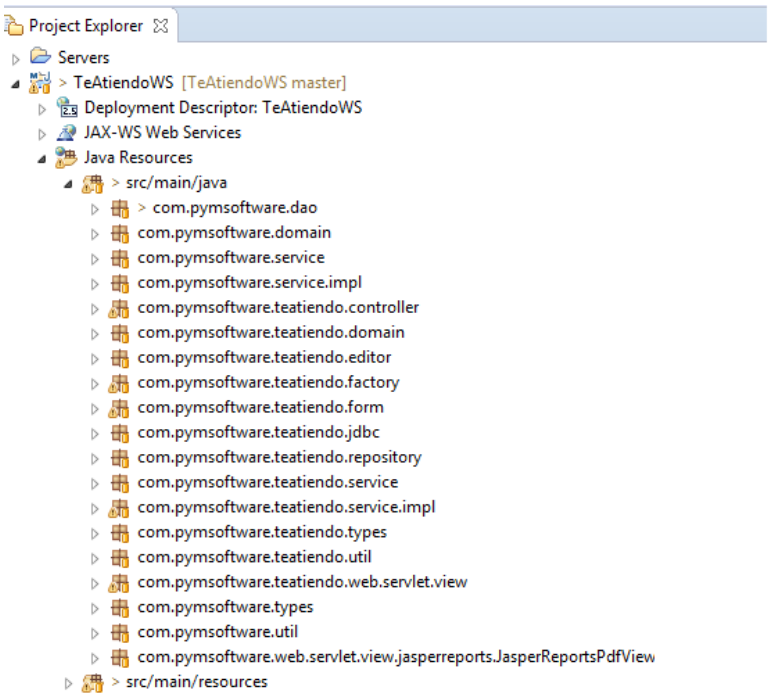


Figura 2.7: Patron Layer Te atiendo

Una capa es un conjunto de subsistemas que comparten el mismo grado de generalidad y de volatilidad en las interfaces: las capas inferiores son de aplicación general a varias aplicaciones y deben de poseer interfaces más estables, mientras que las capas más altas son más dependientes de la aplicación y pueden tener interfaces menos estables. Debido a que las capas inferiores cambian con menor frecuencia, los desarrolladores que trabajan en las capas superiores pueden construir sobre capas inferiores estables subsistemas en diferentes capas pueden reutilizar casos de uso, clases, interfaces, colaboradores y otros componentes de las capas inferiores. Podemos aplicar sobre un mismo sistema muchos patrones de arquitectura. Los patrones que estructuran el modelo de despliegue es decir (client/ server, Three-tier, o Peer to Peer) pueden combinarse con el patrón Layers, lo cual nos ayuda a estructurar el modelo del diseño. Los patrones que tratan estructuras en diferentes modelos son a menudo independientes unos de otros. Incluso lo patrones que tratan el mismo modelo suelen poder combinarse bien mutuamente.

2.4 Descripción de la Arquitectura

La línea base de la arquitectura no solo se usa para desarrollar una arquitectura, sino también para especificar los requisitos del sistema en un nivel que permita el desarrollo de un plan detallado, el modelo de casos de esta línea base pueden contener también más casos de uso aparte de los interesantes desde el punto de vista de la arquitectura

La descripción de la arquitectura debe mantenerse actualiza a lo largo de la vida del sistema para reflejar los cambios y las adiciones que son relevantes para la arquitectura. Estos cambios son normalmente secundarios y pueden incluir:

- La identificación de nuevas clases abstractas e interfaces
- La adición de nueva funcionalidad a los subsistemas existentes
- La actualización a nuevas versiones de los componentes reutilizables
- La reordenación de la estructura de procesos

La descripción de la arquitectura también incluye requisitos significativos para la arquitectura que no están descritos por medio de los casos de uso. Estos otros requisitos son no funcionales y se especifican como requisitos adicionales, como aquellos relaticos a la seguridad, la infraestructura, e importantes restricciones acerca de la distribución y la concurrencia

La descripción de la arquitectura subraya los temas de diseño más importantes y los expone para ser considerados y para obtener la opinión de los otros. Después se deben de tratar de analizar y resolver esos temas

2.4 Resumen

En resumen un clico de vida se compone de una secuencia de iteraciones. Algunas de ellas especialmente las primeras, nos ayudan a comprender los riesgos, determinar la viabilidad, construir el núcleo interno del software, y realizar el análisis del negocio.

Las iteraciones ayudan a la dirección a planificar, a organizar, a hacer seguimiento y a controlar el proyecto. Las iteraciones se organizan dentro de las cuatro fases, cada una de ellas con necesidades concretas de personal, financiación, planificación, y con sus propios criterios de comienzo y de fin.

La ingeniería de software y las técnicas de modelado ayudan a construir grandes sistemas de software, con componentes reutilizables que permiten la escalabilidad y al optimización del tiempo de desarrollo.

El desarrollo iterativo e incremental requiere no solo una nueva forma de gestionar los proyectos, sino también de herramientas que soporten este nuevo método. Es prácticamente imposible tratar con todos los artefactos del sistema que están sujetos a cambios concurrentemente en cada construcción y en cada incremento sin el apoyo de las herramientas.

Capítulo 3

Arquitectura de software para aplicaciones Web

Dentro del desarrollo de aplicaciones web han surgido una gran variedad de metodologías basadas en UML, que abordan con éxito la especificación de la navegación y la presentación que las aplicaciones web exigen.

Actualmente en un mercado tan competitivo como es interne, nos encontramos con la necesidad de desarrollar las aplicaciones web cada vez más complejas y en el menor tiempo posible. Además dichas aplicaciones deben de cumplir con requisitos de calidad como lo son el rendimiento, la usabilidad, la escalabilidad, el mantenimiento para poder satisfacer a los usuarios.

En los últimos años, dentro de la ingeniería de software ha nacido un nuevo paradigma como es Web Engineering, el cual han definido como una colección de metodología, cada una específica con mayor o menor acierto la problemática que un desarrollo web exige.[23]

Para la implementación, se sigue un proceso de desarrollo que implica realizar inicialmente un proceso de recogida de requisitos del usuario, en el cual obtendremos tantos requisitos funcionales, como no funcionales. Estos últimos expresados a partir de un conjunto de atributos de calidad claramente clasificados. A partir de dichos requisitos, se efectúa la definición de un conjunto de modelos expresados mediante UML y siguiendo el estándar MDA (Model –Driven Architecture) , las diferentes vistas en las que comprenderá la aplicación web, cada uno de los cuales se encargara de recoger una vista diferente de la aplicación. Además cada uno de los modelos se encuentran relacionados entre sí, ya sea mediante la trazabilidad de cada una de las vitas o mediante los procesos de mapeo de los modelos que se encuentran en diferentes niveles de abstracción

En los siguientes puntos vamos a especificar las diferentes vistas en las que dividimos la aplicación web, se justificara el uso de la arquitectura de software, para entrar a explicar cada una de las fases en las que se compone las vistas de arquitectura

3.1 Definición de arquitectura web mediante MDA

Utilizando la definición de OMG [24] define una aproximación para la especificación de sistema cuyo principio es la separación de la funcionalidad del sistema de los aspectos dependientes de la plataforma de destino. Esto lo realiza mediante una arquitectura de modelos que provee un conjunto de guías para estructurar las especificaciones expresadas como modelos.

Como vemos en la definición de MDA, existen dos aspectos diferenciados, la separación con la plataforma destino y por otro lado es la aportación de una arquitectura de modelos, la cual está definida a partir de los conceptos de vistas,

refinamiento y abstracción que nos van a permitir definir un conjunto de modelos interrelacionados con diferentes vistas y en diferentes niveles de abstracción que nos permitirán realizar un seguimiento durante todo el ciclo de vida

Las principales ventajas que nos aporta la aproximación a MDA son:

- Integración de sistemas
- Utilización de estándares como UML que nos proporcionan la posibilidad de intercambiar nuestros modelos a cualquier herramienta que sea compatible con UML
- Reducción de costos durante todo el ciclo de vida de la aplicación
- Rápida inclusión de los beneficios de las tecnologías emergentes en los sistemas existentes

3.2 Arquitectura Lógica

Dicha Arquitectura se encarga de expresar cuales son los componentes lógicos (Subsistemas, módulos o componentes de software) que participan en nuestra solución y la relación entre ellos. Dicha arquitectura se obtiene a partir de los requisitos funcionales y no funcionales definidos en la primera fase.

Para conseguir relacionar cada uno de los requisitos no funcionales recogidos en la fase de requerimientos con el diseño arquitectónico especificado en la vistas de arquitectura, vamos a utilizar por un lado los patrones, definidos a diferentes niveles de abstracción, que nos van a indicar cuales son los requisitos que cumplen con cada uno de los casos y por otro cuales son las operaciones unitarias (lógica) que se están realizando al aplicarse.

Existen muchas técnicas y lenguajes para especificar la arquitectura lógica de un sistema. Nosotros utilizaremos una nomenclatura basada en UML, nuestro sistema se divide en 3 modelos cada uno correspondiendo a 3 fases dentro del diseño de la arquitectura lógica, cada una de estas fases va estar situada en un nivel de abstracción distinto, que nos va permitir recoger diferentes aspectos de la arquitectura y poder captar todos los requisitos especificados por el usuario.

1. **Modelo de subsistemas:** Denominado tradicionalmente como diseño estructural, define cuales son los subsistemas que van a construir nuestra aplicación
2. **Modelo de configuración de componentes web**: Consiste en refinar cada uno de los subsistemas descomponiéndolos en cada uno de los componentes abstractos propios de la aplicación web, expresándose de forma independiente de dominio y de plataforma
3. **Modelo de integración de componentes Web** .Consiste en definir cuáles son los componentes o módulos concretos una vez estos apareados con el dominio del problema.

Cada uno de estos modelos tiene una dependencia de forma que el modelo de configuración de componentes gurda las restricciones establecidas por el modelo de

subsistemas y el modelo de integración ha de mantener las relaciones establecidas con el modelo de configuración

3.3 Modelo de subsistemas

Denominado tradicionalmente como el diseño estructural tiene que ver con el diseño de los macrocomponentes de nuestra aplicación Esta fase se localiza en el nivel de abstracción más alto en el diseño arquitectónico, consiste en definir claramente cuáles van a ser los subsistemas que van a construir nuestra aplicación Cada uno de los subsistemas van a estar identificados con una capa lógica de la aplicación. Dicha separación en capas es un principio que nos va a ayudar a reducir la complejidad de los sistemas, basándonos en el patrón " Layering approach"[25].

La reducción de las dependencias entre los módulos y el tratamiento local de los cambios, la portabilidad y la reutilización son aspectos bien conocidos de una estratificación por capas. Como lo muestra la figura 3.1

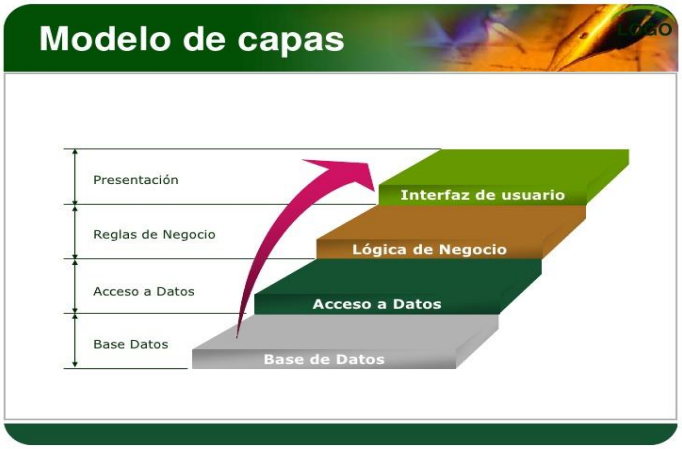


Figura 3.1: Refinamiento del modelo en capas

Nuestra aplicación Te Atiendo, se estructura en 3 capas fundamentales

1. Interfaz de usuario
2. Lógica de Negocio
3. Persistencia

Interfaz de usuario: Capa encargada de dotar de la funcionalidad necesaria para la interacción entre el usuario y la aplicación, el patrón de distribución que divide ambas capas se denomina " Presentación distribuida", y divide el refinamiento en dos sub capas:

- **Presentación**: se encarga de dotar únicamente la funcionalidad de aspecto, y recibir las peticiones del usuario que en algunos casos serán válidas
- **Control de usuario:** Se encarga de recibir las peticiones del usuario, gestionando la navegación del interfaz de usuario, y re direccionando las peticiones a la lógica del negocio, la cual procera la petición y devolverá un resultado al control de usuario.

Lógica del negocio: Esta es la parte del sistema que resuelve las reglas del negocio especificadas para el dominio del problema Si aplicamos el patrón de distribución “Núcleo de aplicación distribuido” podemos definirla en dos subcapas:

- Procesamiento de control: Esta se encarga de atender las peticiones que se reciben del cliente, convirtiéndola en procesos que se realizara del forma transaccional no, de forma síncrona o asíncrona.
- Objetos del Negocio: En dicho sistema residen los sistemas que contienen la representación de los objetos definidos en el domino del problema.

Persistencia*:* Es la parte del sistema encargada de dotar la persistencia al sistema de información, está marcada por la aplicación del patrón “Base de Datos Remota” se distribuye en dos subcapas:

- **Acceso a datos:** El subsistema se encarga de contener aquellos elementos que permiten el acceso a los datos físicos desde la capa lógica del negocio.
- **Capa fascia de datos**: Determina cuales van a ser las fuentes físicas de datos que van a contener nuestro sistema.

En la figura 3.2 se muestra el diagrama de diseño estructural de Te Atiendo

Figura 3.2: Subsistemas Te Atiendo

Una vez hemos divido el sistema en los subsistemas que lo constituyen, el siguiente paso es ver cuáles son los componentes que lo componen. Dichos componentes deben definirse respeto al esquema definid, ya que las dependencias entre los diferentes subsistemas, nos indicaran si los componentes de estos podrán relacionarse o no.

3.4 Modelos de configuración de componentes Web

En este se encuentran los componentes que contiene cada subsistema para construir la arquitectura estática de la aplicación, atreves de una colección de componentes de dominio web abstractos vinculados entre sí a través de diferentes tipos de relaciones de dependencia. Cada uno de los componentes web va a estar identificado por un tipo propio de subsistema en el que está ubicado, y además tendrá asociada una cardinalidad sobre la población de dicho componente.

A este nivel de abstracción vamos a poder identificar otra serie de elementos de" reutilización como lo son los "patrones de arquitectura" que nos van a permitir relacionar el modelo con los requisitos no funcionales, los patrones serán aplicados a los componentes de las diferentes capas de la aplicación web. [25] [26]

- Las propiedades principales de dicho modelo son:
- Se define una filosofía de elementos localizados en el dominio de las aplicaciones web.
- Proporciona un nivel de abstracción que recoge los patrones arquitectónicos y frameworks, que posibilitan recoger nuevos requisitos de calidad.
- Nos aporta la posibilidad de reutilizar dicho modelo para más de un sistema, independiente del dominio.
- Independencia de lenguaje y plataforma y de esta manara poder especificar un abanico más alto de implementaciones.
- Definición y formalización de elementos a través del metamodelo WebSA (Web Software Arquitectura) en un UML, lo que nos proporciona una trazabilidad de los elementos que se incorporan.

Un componente a este nivel de abstracción va a proporcionar una tarea concreta e identificada dentro de la arquitectura de la aplicación web

Cada uno de los componentes de la aplicación cuenta con constructores de diseño modular los cuales tienen inmersos unas propiedades y elementos que permiten la interacción entre los demás subsistemas. Su calificación se divide en:

Componente abstracto: Representa la abstracción de uno o más componentes del software con una misma funcionalidad o tarea dentro de la aplicación web Además cada componente va a tener una relación de dependencia con uno o más componentes que estarán restringidas en función del tipo y del modelo del subsistema definido en la fase anterior.

Conector abstracto: Relación de dependencia establecida entre dos componentes del sistema Esta relación expresa una dependencia entre los dos componentes, dependencia que va venir expresada por el tipo de conector. Además al igual que los elementos, se va a definir una cardinalidad de despliegue que va a generar los elementos conectores que cumplan con la condición de despliegue.

3.4.1. Clasificación de los tipos de componentes

Para hacer la clasificación de los tipos de componentes vamos a identificar cuáles de ellos pueden definirse en cada una de las capas identificadas en el modelo de subsistemas.

3.4.1.1 Subsistemas de interfaz de usuario

Capa encargada de dotar la funcionalidad necesaria para la interacción entre el usuario y la aplicación.

Los elementos que podemos identificar en cada una de las subcapas son:

a. *Subsistemas de presentación*: Su principal función es la de mostrar e interactuar con el usuario de la aplicación. En esta capa va aparecer los siguientes tipos de elementos:

Explorador: Cualquier explorador HTML estándar, capaz de interpretar las páginas de presentación, estilo y funcionalidad, y mostrársela al usuario.

Página Cliente: Contiene el contenido necesario para para presentar la información al usuario, recibir la entrada de datos, validarlos si es necesario y además realizar las invocaciones al servidor, puede contener referencias a otros componentes cliente, estilo y funcional, en la aplicación te atiendo la podemos observar en la figura 3.3

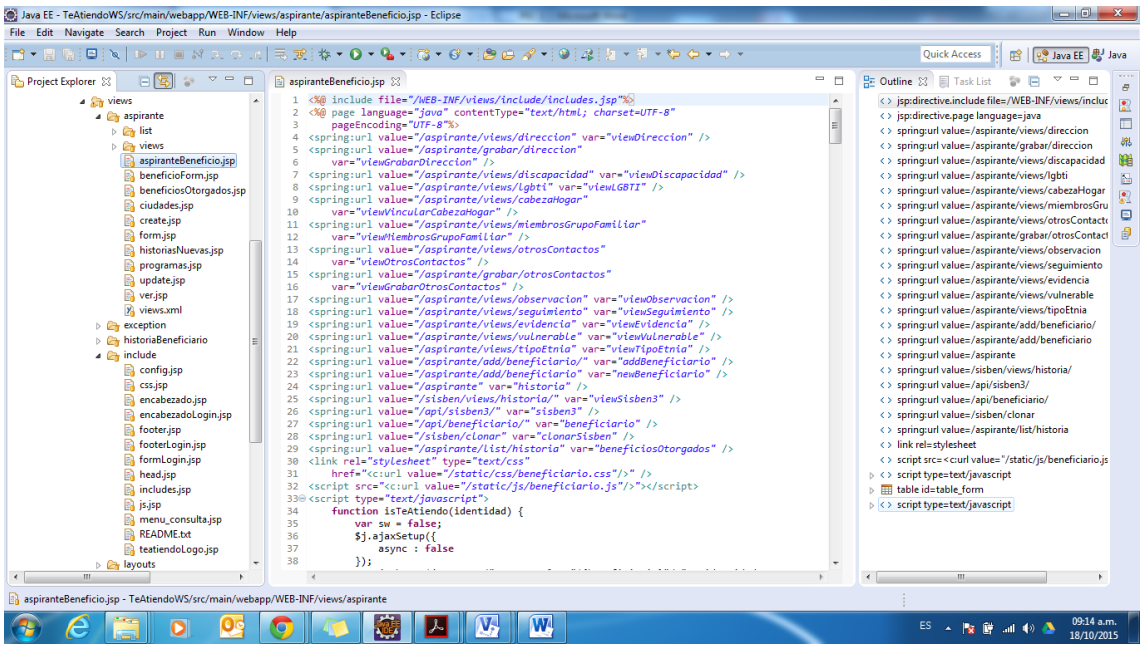


Figura 3.3: Subsistema webapp Te Atiendo

Página Estilo: Componente que únicamente mantiene información referente al aspecto del interfaz del usuario, puede ser referenciado por un componente cliente, la podemos observar en la figura 3.4

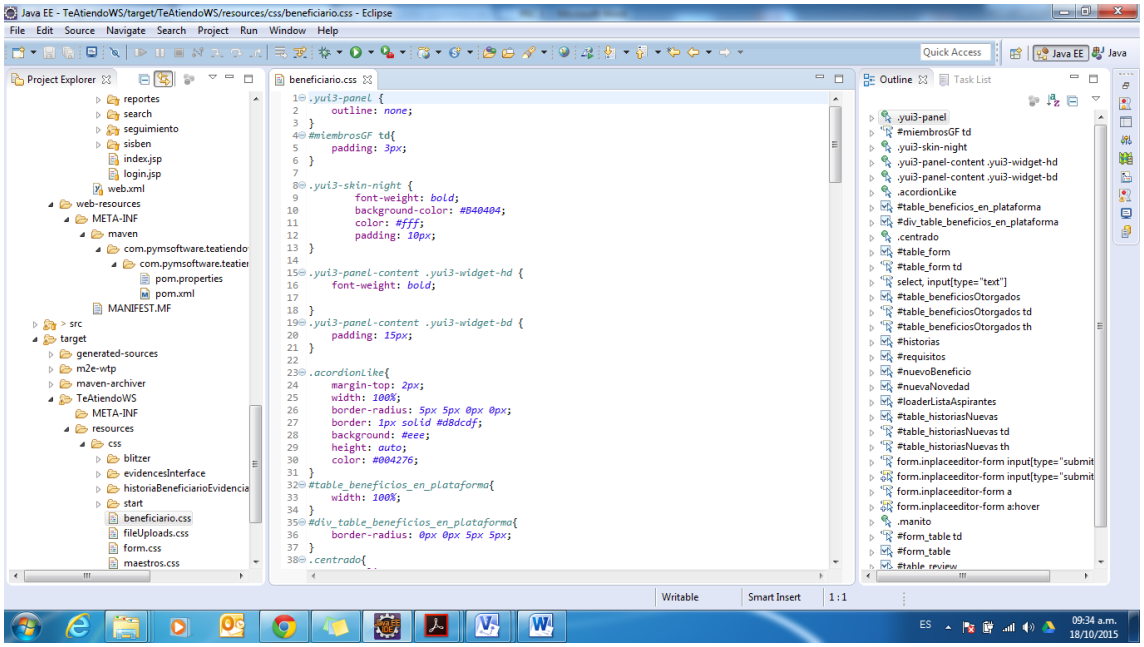


Figura 3.4: Paginas de estilo css

Página Funcional: Mantiene funcionalidad para la interacción del usuario con el interfaz de la aplicación. Dicha funcionalidad puede ser validación, presentación, búsqueda, etc., en el caso de la aplicación “Te atiendo se realiza con JavaScript.

Almacén: Mantiene un mecanismo de almacenamiento localizado en la capa del cliente, que le va a permitir recuperar información entre sesiones.

b. *Sistemas de control de dialogo*: El subsistema de control de dialogo o capa de presentación se encarga de realizar el procesamiento de la funcionalidad de presentación, tanto a lo que se refiere a su navegación, a su presentación y en las invocaciones que se realizan a la capa lógica. Basándonos en la separación funcional de procesamiento, entrada y salida que hace el patrón MVC.

Los principales identificados son:

Página Servidora: Componente que realiza el procesamiento de la página web mediante scripts ejecutados en la parte servidora, normalmente por un engine que ejecuta el código de la página y le devuelve el contenido al servidor web, dicho componente contiene la información referente a la presentación y/o lógica en el servidor y genera uno o más componentes de la capa de interfaz de usuario.

Control: componente que se encarga de recibir las peticiones realizadas por el usuario y establece la reacción de la interfaz. La funcionalidad que aporta es la de redireccionar las peticiones de servicios a la lógica de negocio o componente, modelo, y la de redireccionar las peticiones de navegación a las páginas servidoras.

Modelo: Componente que representa los datos y la funcionalidad del dominio de la capa de presentación. Está definido mediante el diagrama de clases de la vista conceptual.

Vista: Componente que obtiene la información del modelo, y muestra la información al usuario Figura 3.5

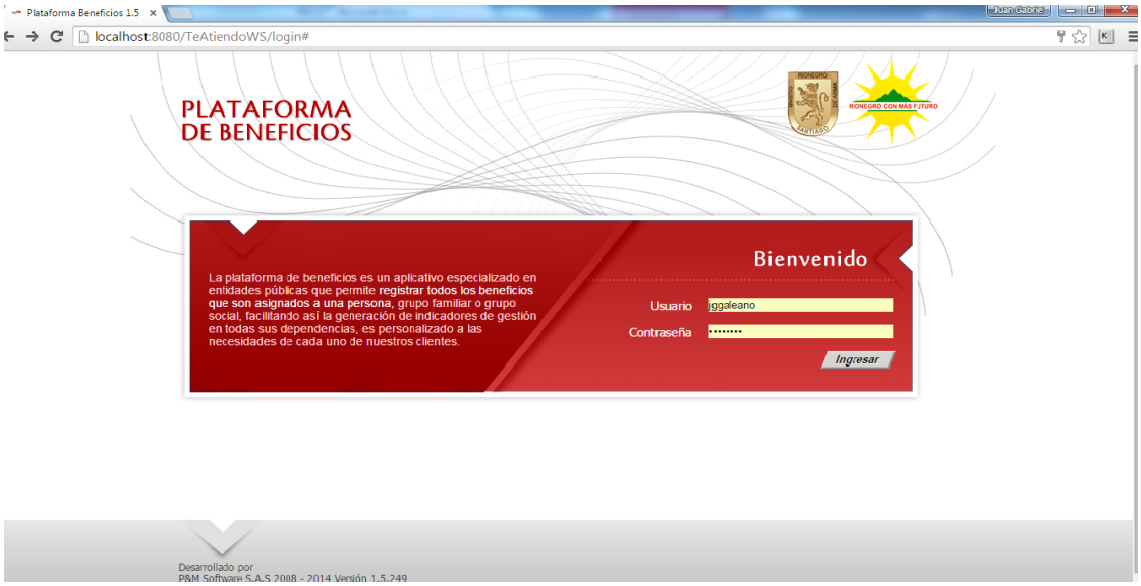


Figura 3.5: Vista

Sesión: Componente que mantiene la información durante la sesión de un determinado usuario. Figura 3.6

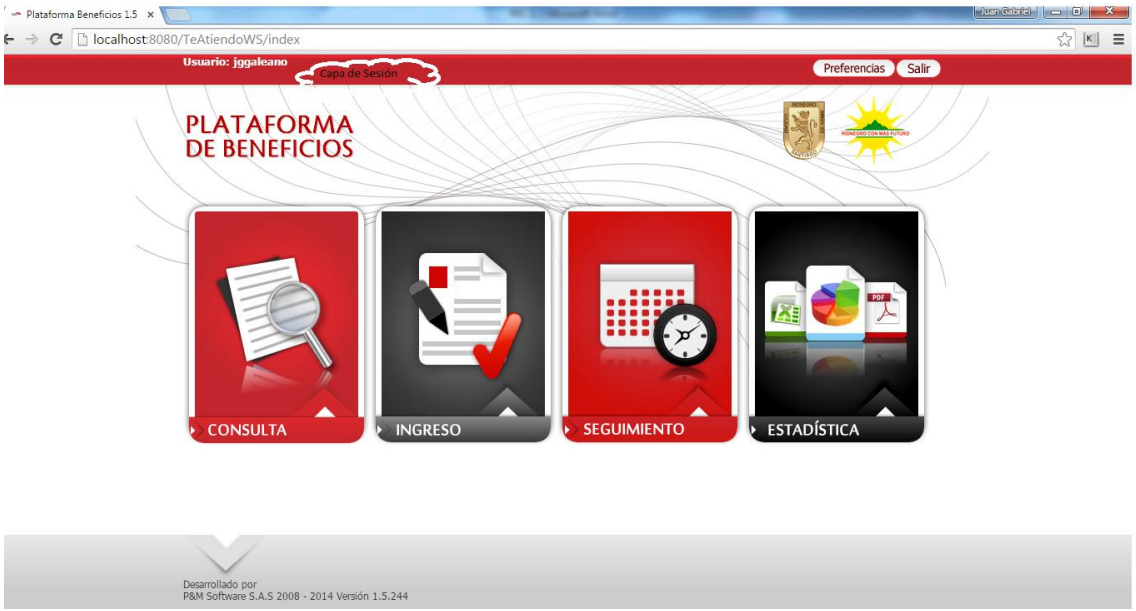


Figura 3.6: Sesión

Aplicación: Componente que se mantiene común para todos los usuarios

Cliente: Componente que mantiene la información para un determinado cliente.

Legado: Componente que realiza una invocación a la capa de control para que realice una tarea específica.

Almacén: Componente que va a mantener la información de forma persistente localizada en la capa de control o presentación. Esta capa almacena información referente a configuración, idioma, personalización. En el caso de la aplicación “Te Atiendo”, se pueden encontrar en los archivos (Properties, xml).

3.4.1.2 Subsistemas de lógica de negocio

Resuelve las reglas de negocio específicas para el dominio del problema, se divide en dos subcapas, procesos de negocio y los objetos del negocio. Ambas partes aparecen especificadas por la vista conceptual de sistemas, y se va a distribuir en un conjunto de componentes que en función de los requisitos de la calidad van a sugerir la utilización de computación distribuida. Los componentes identificadosen este subsistema están basados en abstracciones estándar EDOC(Enterprise Distributed Object Computing) [27] , los cuales nos van a permitir obtener el refinamiento a los componentes

a) Subsistema de Procesamiento de Control: Esta parte del sistema se encarga de atender las peticiones que se reciben del cliente, convirtiéndolas en procesos que se realizará de forma transaccional o no, de forma síncrona o asíncrona. Aquí utilizaremos las siguientes combinaciones de componentes:

Sesión sin estado: Componente que no mantiene estado de un determinado cliente, es decir, se crea un hilo de ejecución nuevo por cada invocación que se realice. Lo que le otorga la posibilidad de generar n hilos muy ligeros y atender a muchos clientes simultáneamente. Se utiliza cuando el número de usuarios de la aplicación es elevado.

Sesión con estado: Componente que mantiene estado de un determinado cliente durante toda su interacción. Lo que le obliga a establecer una instancia por cada cliente. Resultando más útil cuando se exigen altas prestaciones, a un número de usuarios no muy elevado.

Sesión Asíncronos: Componente que recibe invocaciones en modo asíncrono de una lista de distribución o de una pila de llamadas.

Componente por lotes: Componente que realiza un conjunto de eventos no transaccionales.

Componente servidor legado: Componente que realiza su tarea desde un sistema legado.

Funcional: Componente que contiene una colección de funciones accesibles por el resto de elementos del subsistema.

b). Subsistema de objetos de negocio: En dicho subsistema residen los elementos que contienen la representación de los objetos definidos en el dominio del problema, aparecen definidos en la vista conceptual. Dichos componentes pueden ser de estos tipos:

Entidad distribuida: Componente que representa una instancia de una o más clases del modelo. Además, mantiene la información del modelo en memoria de manera que previo un proceso de carga, se accede a esta información y posteriormente se actualiza dicha información en la capa de persistencia. Esto permite el número de accesos a base de datos con la ventaja a nivel de rendimiento que supone.

Entidad local: Componente que representa una instancia de una o más clases del modelo, pero que necesita acceder a la persistencia para realizar cualquier operación. Es utilizada cuando no es necesario optimizar los accesos a BD. CAD (Componente de Acceso a Datos): Componente cuya funcionalidad es la de acceder a los datos independizando a las componentes del dominio del tipo de persistencia que el sistema utiliza. Está basado en el patrón de diseño DAO de Gamma.

Funcional: Componente que contiene una colección de funciones accesibles por el resto de elementos del subsistema.

Legado: Representa una instancia del modelo o de otro sistema que es mantenida por un sistema legado.

4.4.1.3. Subsistemas de persistencia

Parte del sistema encargada del dotarle de persistencia al sistema de información. Dada la división marcada por la aplicación del patrón Base de Datos Remota, los componentes en cada una de las subcapas van a estar enfocados por un lado al acceso y por el otro a la gestión de dichos datos.

a) Subsistema de acceso a datos: Este subsistema se encarga de contener aquellos elementos que permiten el acceso a los datos físicos desde la capa de lógica de negocio. Además provee de la funcionalidad necesaria para poder otorgar una serie de mejoras de rendimiento y escalabilidad como son la utilización de un pool de conexiones o la de disponer de bases de datos remotas.

Pool: Componente que consiste en una colección de recursos que nos permiten reutilizar dichos recursos para un conjunto de usuarios elevados, y además reducir drásticamente el tiempo necesario para crear y destruir cada conexión física.

Fuente de Datos: Componente que nos proporciona la conexión lógica con la fuente de datos. Va a mantener una serie de propiedades como son la posibilidad de admitir transacciones, el usuario y password, etc.

CAD (Componente Acceso Datos): Componente que nos proporciona la posibilidad de realizar una conexión física con la fuente de datos. Dicho puente puede admitir conexiones remotas (TCP/IP...), ser estándar (JDBC, ODBC.), admitir BDOO o BDR o Legacy.

b) *Subsistema físico de Datos*: Este subsistema determina cuales van a ser las fuentes físicas de datos que va a contener nuestro sistema. Dichas fuentes son de diversa naturaleza, y pueden estar distribuidas de forma local o remota dependiendo del tipo de puente utilizado. Los elementos que nos podemos encontrar son los siguientes:

Almacén: Gestor donde almacenamos la información de nuestro sistema de información. Puede ser de diferentes tipos: Relacional: Base de Datos Relacional, compuesta por un conjunto de tablas relacionadas y mediante un lenguaje de consulta SQL que nos facilita el acceso a los datos.

Orientado Objetos: Base de Datos Orientada a Objetos, más sofisticada que la relacional puesto que no solo permite el tratamiento de los datos, sino que proporcionan un lenguaje propietario que posibilita la definición de lógica de negocio a este nivel.

Ficheros Planos: Base de datos que gestiona ficheros físicos, ya sea con un formato propio o estándar (XML).

Legada: Sistema legado que nos proporciona datos de un sistema existente y sobre el cual vamos a acceder para el funcionamiento del sistema.

Funcional: Funciones localizadas a un nivel dependiente del gestor de base de datos, pueden tratarse tanto de procedimientos almacenados en el lenguaje del propio gestor o en XML.

3.4.1.4 Conectores

Existen diferentes tipos de conectores cada uno de los cuales indica una relación diferente entre componentes o módulos:

Invocación Remota: Llamada que realiza un componente a otro elemento del diagrama de forma remota, es decir, soporta una distribución en nodos físicos diferentes (HTTP,TCP/IP,SMTP).

Invocación Remota Segura: Llamada que realiza un componente a otro elemento del diagrama de forma remota y además utilizando un protocolo seguro (p.e. HTTPS, IMAP/SSL, SMTP/SSL, etc.)

Invocación local: Llamada que realiza un componente a otro elemento del diagrama de forma local, es decir, dichos elementos han de residir en el mismo subsistema.

Construir (build): Identifica que componente página servidora es responsable de la creación de uno o más componentes página cliente, o funcional o estilo. Es una relación direccional puesto que solo los componentes página servidora tiene conocimiento de los componentes cliente y no al contrario.

Contiene (Include): Relación que se establece entre dos componentes en el que el primer componente contiene al segundo. Relación definida por [28] para vincular una página témplate con las páginas que referencia.

En la figura 3.7 se definen los elementos que componen el diseño modular, utilizando la notación UML para la aplicación Te Atiendo.

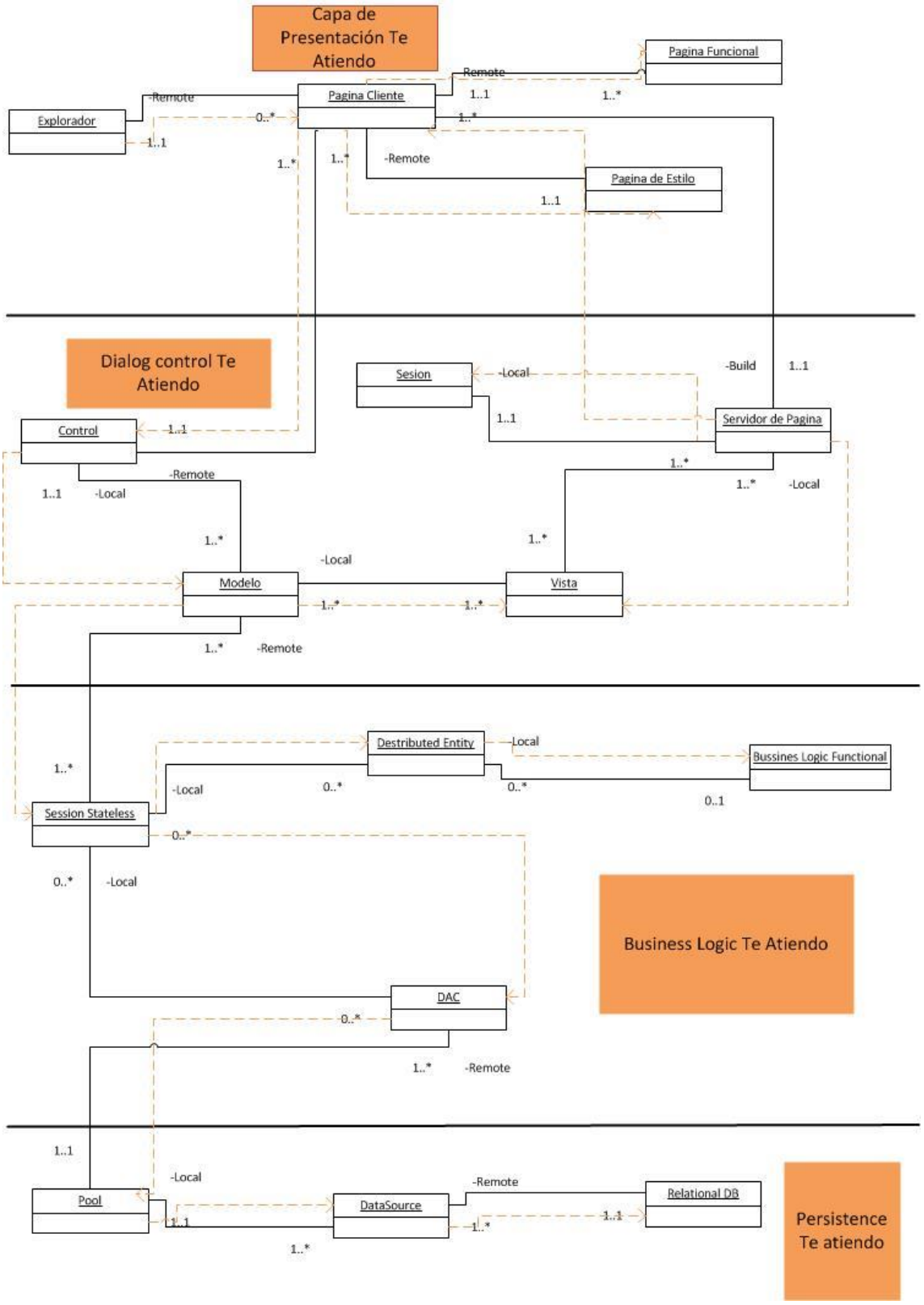


Figura 3.7: Modelo de composición de componentes de la aplicación Te Atiendo

3.5 Diagrama de integración de componentes

Una vez hemos identificado en los modelos anteriores la arquitectura de la aplicación, se hace necesario vincular la parte no funcional recogida a través de los diagramas de arquitectura anteriores con la parte funcional de la aplicación, recogida a través de los diagramas de procedentes de las fases de análisis., en la cual se puede recoger todos los requisitos que una aplicación exige.

Entre los componentes hay varios que deben de invocar objetos de otros, esta vinculación de los componentes nos va a proporcionar un mapeo por defecto que dará como resultado un modelo sin refinar de la integración de la arquitectura y funcionalidad expresada

El diagrama de integración presenta las siguientes propiedades:

1. Recoge requisitos que implican tanto aspectos funcionales como no funcionales.

2. Es dependiente de aplicación, ya que representa los componentes del dominio de las aplicaciones web para una aplicación en concreto.

3. Es independiente de plataforma de manera que dicho modelo puede servir para generar código sobre diversas implementaciones. Es según la especificación MDA un modelo PIM (Platform-Independent-Model).

4. Reduce drásticamente el tiempo de modelado ya que se nos proporciona un modelo por defecto gracias al mapping que debemos refinar.

5. Respecta las restricciones establecidas por los modelos de arquitectura abstractos.

3.5.1 Constructores

a) Componente Concreto (CCAW)

Unidad más pequeña dentro del modelo de integración, y representa a un componente software dentro del dominio una aplicación web. Es la instanciación de un componente abstracto del tipo definido en el modelo de composición de componentes y recoge la funcionalidad completa que dicho componente aportará al sistema.

A diferencia de los componentes abstractos definidos en el modelo MCC donde los componentes de diferentes tipos únicamente realizaban tareas identificadas como comunes en el dominio de las aplicaciones web, un componente de este nivel contiene toda la funcionalidad del componente software.

Además presenta la característica fundamental de ser independiente de dispositivo, es decir, soporta n implementaciones, tantas como plataformas sean capaces de reproducir su funcionalidad, lo que nos otorga un mayor nivel de reutilización, y por otro lado, dota de mayor portabilidad a un sistema.

Propiedades:

a) **Tipo**: Al igual que en nivel abstracto dicho tipo sigue representando la tarea que dicho componente software realiza en la aplicación web. Un componente dentro del dominio de una aplicación web podría ser desde una página, a un componente de servidor, a un componente de control, etc.

b) **Servicios**: Son aquellas operaciones ofertadas por el componente definido por el tipo del componente y por el dominio de la aplicación. Dichos servicios son obtenidos a través del mapeo con la parte funcional del sistema.

c) **Interfaz:** Representa un subconjunto de los servicios públicos definidos por el componente dentro del dominio de la aplicación, que dicho componente oferta al resto del sistema para poder ser invocados. Se reduce así el nivel de dependencia entre un componente y otro cuando la relación se establece a través de su interfaz y no del mismo componente.

b) Módulo (MAW):

Es el conjunto de uno o más elementos concretos del mismo tipo dentro de una aplicación web, dichos elementos son módulos, componentes y conectores. Su funcionalidad es la de agrupar la funcionalidad de un conjunto de elementos reduciendo así la complejidad de modelado.

Se genera a partir del proceso de refinamiento de un componente abstracto cuya cardinalidad es mayor que 1, que al realizarse la integración da lugar a un conjunto de componentes del mismo tipo, los cuales serán contenidos por dicho módulo.

c) Conector Concreto (CCWA)

Relación establecida entre dos componentes o módulos concretos del sistema. Es la instanciación de una relación de dependencia definida en el modelo abstracto de arquitectura. Influyendo en el mapeo la cardinalidad definida en el conector abstracto y las relaciones establecidas en el metamodelo entre los tipos de los componentes abstractos y la parte funcional, obteniendo así un conector perteneciente al del dominio de la aplicación.

Por último indicar que dicha relación cuando representa una invocación, se establece con un determinado servicio público del componente destino, ya sea sobre el mismo componente o sobre su interfaz lo que indicaría un nivel de acoplamiento menor.

Propiedades:

Tipo: Indica el tipo de relación establecida entre los dos componentes o módulos concretos. Cada uno de estos tipos va a indicar si se trata de una tarea de invocación, construcción (build), si es remota o local (admite llamadas entre dos nodos diferentes de la arquitectura física), etc.

Servicio: En aquellas relaciones del tipo invocación local o remota que representan invocaciones de uno a otro componente dicho conector puede identificar el servicio que invoca en el componente destino.

3.5.2 Proceso de Refinamiento

En la mayoría de los casos se hace necesario un proceso de refinamiento en el cual se capturen aquellos requisitos de usuario que no hemos podido recoger en fases anteriores.

Los procedimientos más típicos de refinamiento que podremos realizar con este modelo son los siguientes:

1. Integración de funcionalidad en componentes: Integramos la funcionalidad que estaba repartida en 2 o más componentes, en un solo componente reduciendo así el número de invocaciones entre componentes.

2. División de funcionalidad de componentes: Dividimos la funcionalidad localizada en un solo componente y la repartimos en 2 o más componentes.

3. Cambios de invocación de componentes: Puede venir consecuencia de otros cambios de refinamiento, o puede derivarse de un cambio de la arquitectura en algunos casos particulares del dominio del problema. Por ejemplo, si queremos

3.6 Uso de SCRUM

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

Scrum se basa en:

1. El desarrollo incremental de los requisitos del proyecto en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas, si así se necesita).

2. La priorización de los requisitos por valor para el cliente y coste de desarrollo en cada iteración.

3. El control empírico del proyecto. Por un lado, al final de cada iteración se demuestra al cliente el resultado real obtenido, de manera que pueda tomar las decisiones necesarias en función de lo que observa y del contexto del proyecto en ese momento. Por otro lado, el equipo se sincroniza diariamente y realiza las adaptaciones necesarias.

4. La potenciación del equipo, que se compromete a entregar unos requisitos y para ello se le otorga la autoridad necesaria para organizar su trabajo.

5. La sistematización de la colaboración y la comunicación tanto entre el equipo y como con el cliente.

6. El timeboxing de las actividades del proyecto, para ayudar a la toma de decisiones y conseguir resultados.

7. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

El proceso

En Scrum un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

El proceso parte de la lista de objetivos/requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas.

Las actividades que se llevan a cabo en Scrum son las siguientes:

Planificación de la iteración

El primer día de la iteración se realiza la reunión de planificación de la iteración.

Tiene dos partes:

Selección de requisitos (4 horas máximo). El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos más prioritarios que se compromete a completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.

Planificación de la iteración (4 horas máximo). El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos a que se ha comprometido. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se autoasignan las tareas.

Ejecución de la iteración

Cada día el equipo realiza una reunión de sincronización (15 minutos máximos). Cada miembro del equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido. En la reunión cada miembro del equipo responde a tres preguntas:

¿Qué he hecho desde la última reunión de sincronización?

¿Qué voy a hacer a partir de este momento?

¿Qué impedimentos tengo o voy a tener?

Durante la iteración el Facilitador (Scrum Master) se encarga de que el equipo pueda cumplir con su compromiso y de que no se merme su productividad.

Elimina los obstáculos que el equipo no puede resolver por sí mismo.

Protege al equipo de interrupciones externas que puedan afectar su compromiso o su productividad.

Durante la iteración, el cliente junto con el equipo refinan la lista de requisitos (para prepararlos para las siguientes iteraciones) y, si es necesario, cambian o replanifican los objetivos del proyecto para maximizar la utilidad de lo que se desarrolla y el retorno de inversión.

Inspección y adaptación

El último día de la iteración se realiza la reunión de revisión de la iteración. Tiene dos partes:

Demostración (4 horas máximo). El equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, replanificando el proyecto.

Retrospectiva (4 horas máximo). El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El Facilitador se encargará de ir eliminando los obstáculos identificados.

3.7 Resumen

En este capítulo se presentó una arquitectura de software basadas en la web WebSA (Web Software Architecture), la arquitectura es el resultado de un estudio de estilos arquitectónicos, patrones de arquitectura, y patrones de diseño, tomando como base

los diferentes modelos que hay para el desarrollo en nuestro caso nos basamos en un modelo cliente servidor con un patrón de división en subsistemas lógicos

El diseño de la aplicación Te Atiendo comienza dividiendo la arquitectura en los subsistemas vistos lo que nos permite, utilizar componentes y patrones de desarrollo ya implementados, además de utilizar Scrum para la gestión de proyecto lo que nos permito concéntranos en los tiempos de las iteraciones de cada módulo logrando con esto una mayor competitividad, flexibilidad y productividad en el desarrollo del proyecto.

Se utilizó el patrón de diseño multicapa para definir las capas de la arquitectura a nivel lógico, se utilizó el patrón Modelo Vista Controlador para separar correctamente las capas de la presentación, la lógica del negocio y los datos, se validó que las capas estuvieran alineadas con el dominio web, el cual nos permitió corregir los errores del diseño y modificar estos en la implementación del código fuente. Se describieron los diagramas UML de la aplicación “Te Atiendo” el cual permite una fácil modulación e interpretación.

Capítulo 4

Caso de estudio

Para comenzar con nuestro desarrollo de la aplicación te atiendo debemos de comprender muy bien el sistema, para lo cual partiremos del papel que juegan los requisitos en un modelado de caso de usos que se desarrollaran a lo largo de la implementación de la aplicación, donde cada día estaremos mirando las iteraciones, basándonos en uso de Scrum, donde las iteraciones añadirán nuevos casos de usos y/o añadirán detalle a las descripciones de los casos de uso existentes.

Durante la fase de inicio, se identificaran la mayoría de los casos de uso para delimitar el sistema y el alcance del proyecto y para detallar los más importantes

Durante la fase de elaboración, se capturan la mayoría de los requisitos restantes para poder estimar el tamaño del esfuerzo del desarrollo

Durante la fase de construcción se capturan, se refinan e implementan los requisitos restantes.

4.1 El desarrollo de la aplicación web

Se conoce que el desarrollo de aplicaciones Web es diferente debido a que los requerimientos (funcionales y no funcionales) y el dominio de estas aplicaciones se enfocan en el dominio Web. Para el desarrollo de aplicaciones Web se requiere definir claramente los alcances de la aplicación y considerar todos los posibles problemas a los que nos podemos enfrentar. Esta tarea resulta complicada debido a que cada persona involucrada en el desarrollo puede tener una idea distinta sobre la funcionalidad de la aplicación.

Para el desarrollo de Te Atiendo utilizamos una búsqueda de casos de uso a partir de un modelo de negocio como entrada, donde se crea un modelo de casos de uso tentativo que nos ayudara a definir el dominio de la aplicación web, el modelado del dominio se realiza en reuniones organizadas por los analistas y las personas que utilizaran la aplicación

El objetivo del modelado del domino es comprender y describir las clases más importantes dentro del contexto del sistema, este modelado se almacena como definiciones en un glosario de términos.

El glosario y el modelo del dominio ayudan a los usuarios, clientes, desarrolladores y a otros interesados a utilizar un vocabulario común. La terminología común es necesaria para compartir el conocimiento con los otros., el modelado de dominio debe contribuir a una comprensión del problema que se supone que el sistema resuelve en relación a su contexto.

A continuación describimos el primer planteamiento después de algunas reuniones realizadas con los implicados en el proyecto " Te Atiendo" , donde el sistema utilizara internet para registrar información de beneficiarios, historia de beneficios, certificación de beneficios y registro y filtro: el sistema ayudara a las dependencias de la secretaría de gestión y protección social a realizar control y trazabilidad de todos los beneficios, registrar todos los beneficios que son asignados a una persona, grupo familiar o grupo social. Facilitando así la generación de indicadores de gestión en todas sus dependencias. Figura 4.1

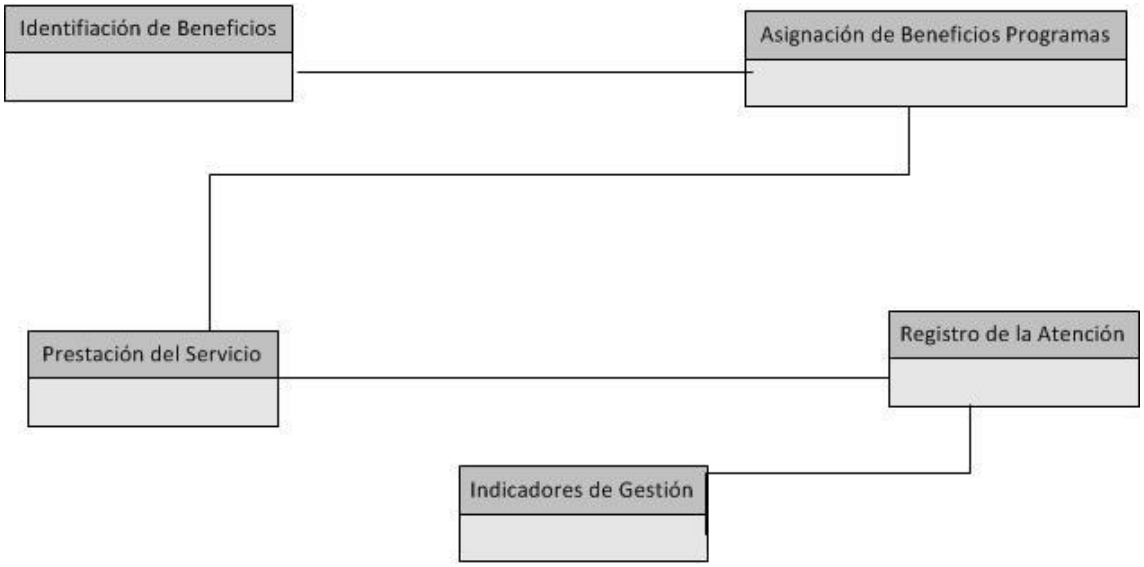


Figura 4.1: Workflow Te Atiendo

Para la implementación del Workflow planteado a los integrantes del proyecto, se procede a realizar el modelo de domino que permite capturar los tipos mas importantes de objetos del contexto del sistema. Los objetos del dominio representan las cosas que existen en los eventos que suceden en el entorno en que trabajara el sistema.

Muchos de los dominios o clases se obtienen de las especificaciones de requisitos y las entrevistas con los usuarios o actores que manejan el proceso. Las clases del domino aparecerán en tres formas típicas:

1. Objetos del negocio que representan cosas que manipulan en el negocio

2. Objetos del mundo real y conceptos de los que el sistema debe de hacer un seguimiento

3. Sucesos que ocurrirán o han ocurrido

Este diagrama muestra a los clientes, usuarios, revisores y a otros desarrolladores las clases de dominio y como se relacionan unas con las otras mediante asociaciones.

En la figura 4.1.se muestra el modelo dominio de la aplicación que corresponde al primer esbozo del 10% de lo que contiene el aplicativo “Te atiendo. En este se muestra lo siguiente:

Unos beneficios que son entregados por la Alcaldía de Rionegro, a unos aspirantes, donde se registra la información en una historia de beneficiarios, se debe de realizar seguimiento a la historia de beneficiarios, para al fin obtener estadísticas e informes reales del impacto que tienen las secretarias.

Para terminar el modelado de dominio se deben de realizar reuniones periódicas con los involucrados en el proyecto con el fin de determinar los requisitos adicionales que son fundamentales requisitos no funcionales que no pueden asociarse a ningún caso de uso en concreto, en cambio cada uno de estos requisitos, tiene impacto en varios casos de uso o en ninguno.Ejemplo de estos son el rendimiento, las interfaces y los requisitos de diseño físico, asi como las restricciones arquitectónicas y de implementación.

Un *requisito de Interfaz*, especifica la interfaz con un elemento externo con el cual debe de interactuar el sistema o que establece restricciones condicionantes en formatos, tiempos u otros factores de la relevancia en esa interacción.

Un *requisito físico* , especifica una característica física que debe de poseer un sistema, como su material, forma, tamaño o peso, por ejemplo, puede utilizarse este tipo de requisito para representar requisitos hardware como las configuraciones físicas.

Una *restricción de diseño*, limita el diseño de un sistema, como lo hacen las restricciones de extensibilidad y mantenibilidad, o las restricciones relativas a la reutilización de sistemas heredados o partes esenciales de los mismos.

Una restricción de la implementación, especifica o limita la codificación o construcción de un sistema. Son ejemplos los estándares requeridos, las normas de codificación, los lenguajes de programación, políticas para la integridad de la base de datos, limitaciones de recursos y entornos operativos.

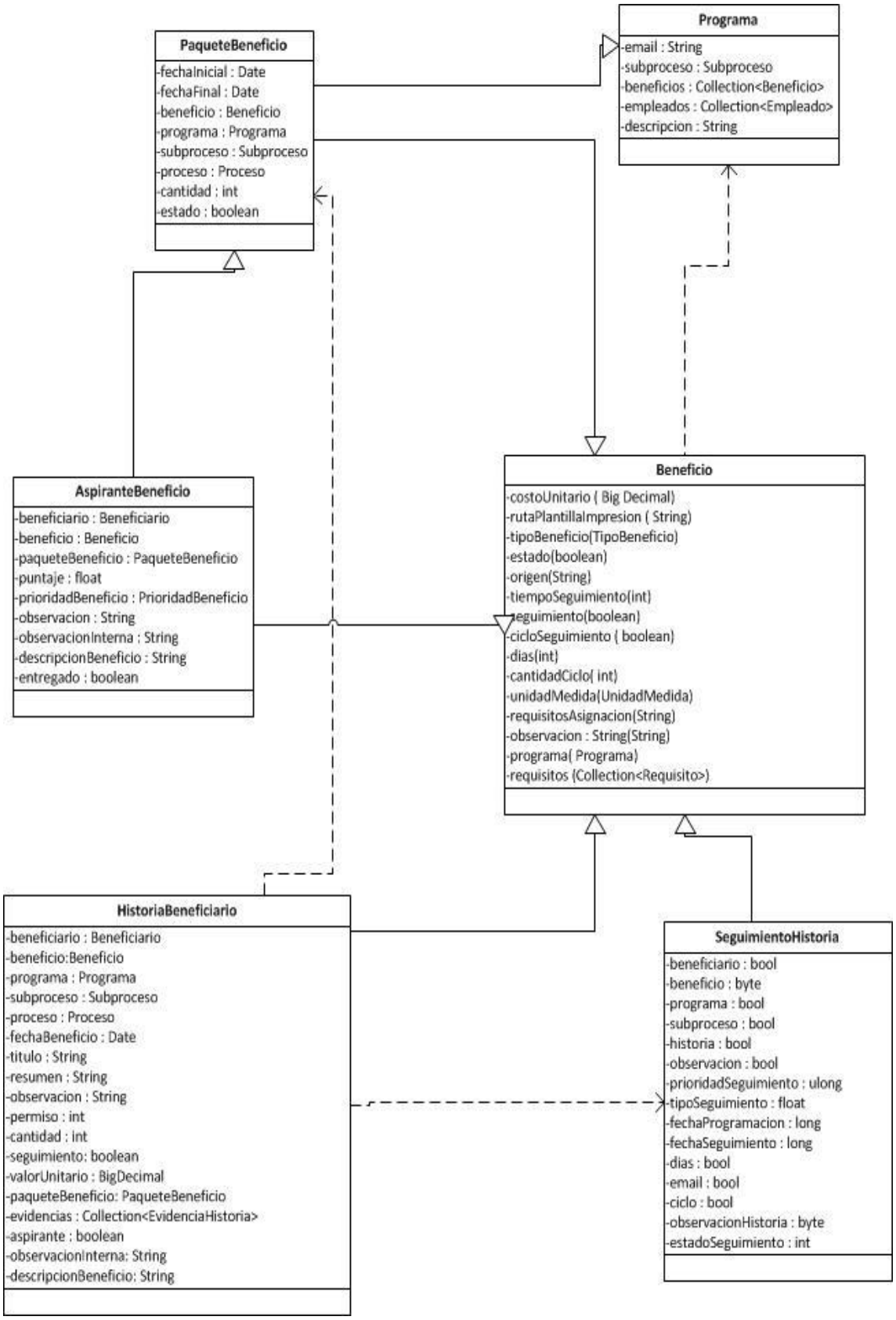


Figura 4.2: Modelo dominio Te Atiendo

4.2 Planificación del proyecto

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Ámbito	**0,5 días**	**mar 29/09/15**	**mar 29/09/15**	
Determinar el ámbito	1 hora	mar 29/09/15	mar 29/09/15	
Definir los recursos preliminares	2 horas	mar 29/09/15	mar 29/09/15	3
Acta de constitución (1 reunión)	1 hora	mar 29/09/15	mar 29/09/15	4
Documento de ámbito y acta de constitución	0 horas	mar 29/09/15	mar 29/09/15	5
Análisis y requisitos de software	**2,5 días**	**mar 29/09/15**	**jue 01/10/15**	
Realizar análisis de necesidades	8 horas	mar 29/09/15	mie 30/09/15	6
Borrador de especificaciones preliminares del software	4 horas	mie 30/09/15	mie 30/09/15	8
Desarrollar cronograma preliminar	2 horas	mie 30/09/15	mié 30/09/15	9
Revisar especificaciones preliminares y cronograma preliminar	1 hora	mié 30/09/15	mié 30/09/15	10
Incorporar comentarios a las especificaciones del software	1 hora	mié 30/09/15	mié 30/09/15	11
Calcular los tiempos y fechas de entrega (hitos)	1 hora	jue 01/10/15	jue 01/10/15	12
Obtener aprobaciones para continuar (concepto, fechas, presupuestos)	1 hora	jue 01/10/15	jue 01/10/15	13
Obtener recursos necesarios (2 y 3 reunión)	2 horas	jue 01/10/15	jue 01/10/15	14
Documento de análisis	0 días	jue 01/10/15	jue 01/10/15	15
Diseño	**5 días**	**jue 01/10/15**	**mié 07/10/15**	
Revisar especificaciones preliminares del software	8 horas	jue 01/10/15	vie 02/10/15	16
Desarrollar especificaciones de funcionamiento (DLD)	16 horas	vie 02/10/15	lun 05/10/15	18
Revisar especificaciones de funcionamiento (DLD Review)	8 horas	lun 05/10/15	mar 06/10/15	19
Incorporar comentarios a las especificaciones de funcionamiento (DLD)	4 horas	mar 06/10/15	mar 06/10/15	20
Documento de Diseño bajo nivel (DLD)	0 horas	mar 06/10/15	mar 06/10/15	21

Desarrollar prototipo diseño grafico	8 horas	vie 02/10/15	vie 02/10/15	18
Revisar prototipo diseño grafico	4 horas	lun 05/10/15	lun 05/10/15	23
Prototipo diseño gráfico (PDG)	0 horas	lun 05/10/15	lun 05/10/15	24
Desarrollar Modelo Entidad Relación (MER)	4 horas	vie 02/10/15	vie 02/10/15	18
Revisión de modelo entidad relación (MER)	2 horas	vie 02/10/15	vie 02/10/15	26
Documento MER	0 horas	vie 02/10/15	vie 02/10/15	27
Obtener aprobación para continuar (4 y 5 reunión)	4 horas	mié 07/10/15	mié 07/10/15	25;22;28
Diseño completado	0 horas	mié 07/10/15	mié 07/10/15	29
Desarrollo	**18,25 días**	**mié 07/10/15**	**mar 27/10/15**	
Revisar especificación de funcionamiento	8 horas	mié 07/10/15	jue 08/10/15	30
Ingreso - Login – Menú	8 horas	jue 08/10/15	jue 08/10/15	32
Permisos - Security - Auditoria	4 horas	vie 09/10/15	vie 09/10/15	33
Refactorización TeAtiendo	80 horas	vie 09/10/15	mié 21/10/15	34
Evidencia	**8 días**	**jue 08/10/15**	**vie 16/10/15**	
Archivos y fotos	40 horas	jue 08/10/15	mié 14/10/15	32
Otros detalles (post-modificar)	24 horas	mié 14/10/15	vie 16/10/15	37
Seguimiento a beneficiarios	**5 días**	**vie 16/10/15**	**jue 22/10/15**	**36**
Agenda y seguimiento	40 horas	vie 16/10/15	jue 22/10/15	38
Pruebas desarrollo (primaria)	32 horas	jue 22/10/15	mar 27/10/15	35;40
Reunión de avance (6 reunión)	2 horas	mar 27/10/15	mar 27/10/15	41
Prototipo Funcional	0 horas	mar 27/10/15	mar 27/10/15	41
Pruebas	**1,25 días**	**mié 28/10/15**	**mié 28/10/15**	
Actividad #1	8 horas	mié 28/10/15	mié 28/10/15	42
Actividad #2	2 horas	mié 28/10/15	mié 28/10/15	45
Capacitación	**0,5 días**	**mié 28/10/15**	**mié 28/10/15**	
Actividad #1	4 horas	mié	mié	42

		28/10/15	28/10/15	
Documentación	**0,25 días**	**jue 29/10/15**	**jue 29/10/15**	
Modulo #1	2 horas	jue 29/10/15	jue 29/10/15	41;45;46
Piloto	**8 días**	**jue 29/10/15**	**vie 06/11/15**	
Secretaria Familia	16 horas	jue 29/10/15	vie 30/10/15	44;48
Retroalimentación piloto	8 horas	vie 30/10/15	lun 02/11/15	52
Piloto otras dependencias Familia	16 horas	lun 02/11/15	mar 03/11/15	53
Retroalimentación piloto otras dependencias	8 horas	mié 04/11/15	mié 04/11/15	54
Piloto otras secretarias	8 horas	mié 04/11/15	jue 05/11/15	55
Retroalimentación otras secretarias	8 horas	jue 05/11/15	vie 06/11/15	56
Implementación	**2 días**	**vie 06/11/15**	**lun 09/11/15**	
Configuración servidor SQL	4 horas	vie 06/11/15	vie 06/11/15	57
Configuración Servidor WEB	4 horas	vie 06/11/15	lun 09/11/15	59
Software y build	8 horas	lun 09/11/15	lun 09/11/15	60
Revisión posterior a la implementación (PM)	**10,5 días**	**mar 10/11/15**	**vie 20/11/15**	
Documentar la experiencia adquirida (postmortem)	4 horas	mar 10/11/15	mar 10/11/15	58
Mantenimiento de software y postventa	80 horas	mar 10/11/15	vie 20/11/15	63
Revisión posterior a la implementación	0 horas	vie 20/11/15	vie 20/11/15	64

Tabla 4.1: Cronograma de actividades

4.3 Captura de Requisitos

Las aplicaciones web han sido una herramienta clave para el desarrollo de las organizaciones, son el pilar fundamental en la administración de grandes cantidades de usuarios y por consiguiente de datos.

Cada día las entidades públicas requieren el fortalecimiento de las instancias administrativas y su medición con datos cuantitativos (números reales en tiempo real), un punto de un buen gobierno es medir los logros o beneficios que se les ha dado a la población o a un grupo de interés, es un método de articulación eficiente y eficaz que es misional dentro de una administración moderna, dichos procesos son

automatizados con sistemas de información (software) que generen registros de cada persona, familia o grupo de interés (barrio, JAC) en todas las dependencias.

Actualmente la secretaria de gestión y protección social viene adelantando esta labor de acompañamiento a diferentes beneficiarios con los programas de protección y/o bienestar social, para registrar dichos beneficios y poder llegar a la población más vulnerable se incorporara el software “TE ATIENDO”, esto implicara varios cambios organizativos que ha ayudaran al proceso misional generando alertas de atención e indicadores de gestión tangibles que son visibles dentro de administración municipal actual ya que muestran en los consejos de gobierno datos reales que son comprobables con consultas dentro del software.

El caso de estudio planteado de la tesis consiste en desarrollar una arquitectura para una aplicación web para el municipio de Rionegro que contenga los siguientes módulos.

- **Beneficiario** (Persona, Familia, Empresa o Grupo especial – Educación, Salud u otros).Registra la información de un beneficiario para así llevar su historial que les ha aplicado, información clave como nombres, dirección, teléfono, barrio, nivel socioeconómico, situación especial. Se puede agrupar por el tipo de familia y su cabeza de hogar.

- **Historial de beneficios**: Caracteriza a todas las personas llevando un registro de todos los procesos o acontecimiento que se les ha dado, a esto se le llama historial de beneficios: Ejemplo: Encuestas, Ayudas, Peticiones, Atenciones en cualquier dependencia del municipio.

- **Entidad, Dependencias, Programa, Beneficio:** Módulos que permite configurar a qué nivel de registro se llega en el municipio, discriminando el programa que pertenece un beneficio que será aplicado a la atención de la población.
- **Certificaciones de beneficios:** Se requiere dar a los beneficiarios cartas o certificaciones en diferentes estilos del beneficio dado. Ejemplos como en la casa campesina o visitas de asistencia.

- **Filtros y Búsquedas:** Módulo de búsquedas y/o filtro de información de los beneficiarios y el historial, este llevaría a una agilidad en consultas de información

- **Seguridad:** Un sistema de permisos y roles que permitan a los usuarios tener módulos independientes y opciones diferentes dependiendo su rol dentro del municipio**.**

- **Integración con otros sistemas**: Actualmente la información está dispersa en diferentes aplicativos, se requiere en modo CONSULTA saber datos básicos de otros sistemas. Ejemplo: SISBEN.

- **Carga de archivos planos:** En ciertos programas (adulto mayor, mana) los beneficios son diarios y/o mensuales y se sabe con antelación quienes son los participantes de ese programa, es necesario ingresar la información continuamente pero NO ES FUNCIONAL ingresarla manualmente, se requiere un módulo que automáticamente ingrese esta información sin pasar por una persona.

- **Informes de estándares de gestión**

1. Inversión en un programa
2. Inversión de un beneficiario y/o un grupo de beneficiarios.
3. Cantidad de beneficios ingresados por el personal adscrito a una secretaria**.**

4.4 Obtención de Requerimientos

La primera etapa para la construcción de un sistema de software es la obtención de requerimientos. Los requerimientos se dividen en dos grupos (1) los requerimientos funcionales y (2) los requerimientos no funcionales. En esta etapa se muestran cada una de las necesidades que se han detectado, de cada uno de los usuarios involucrados en el sistema.

En esta sección se en listan cada uno de los requerimientos de la aplicación de forma natural, como el cliente lo expreso, para después mostrarlo formalmente. Cada uno de estos requerimientos es analizado y expresados de forma técnica y en términos del sistema, para conocer cuáles son las características que el sistema debe de tener.

El esfuerzo principal en la fase de requisitos es desarrollar un modelo del sistema que se va a construir, y la utilización de los casos de uso es una forma adecuada de crear ese modelo: esto es debido a que los requisitos funcionales se estructuran de forma natural mediante casos de uso y que la mayoría de los otros requisitos no funcionales son específicos de un solo caso de uso, y pueden tratarse en el contexto de ese caso de uso.

Los casos de uso proporcionan un medio intuitivo y sistemático para capturar los requisitos funcionales con un énfasis especial en el valor añadido para cada usuario individual o para cada sistema externo. Mediante la utilización de los casos de uso, los analistas se ven obligados a pensar en términos de quienes son los usuarios y que necesidades u objetivos de la empresa pueden cumplir.

El modelo casos de uso permite que los desarrolladores de software y los clientes lleguen a un acuerdo sobre los requisitos, es decir sobre las condiciones y posibilidades que debe de cumplir el sistema. El modelo de casos de uso sirve como acuerdo entre clientes y desarrolladores, y proporcionan la entrada fundamental para el análisis, el diseño y las pruebas.

Un modelo de casos de uso es un modelo del sistema que contiene actores, casos de uso y sus relaciones.

UML nos permite presentar el modelo de diagramas que muestran los actores y los casos de uso desde diferentes puntos de vista y con diferentes propósitos

4.4.1 Requerimientos del Usuario

A continuación se detallan cada uno de los requerimientos del usuario de TE ATIENDO, dichos requerimientos son expresados en el lenguaje natural, de forma que puedan ser comprendidos fácilmente por los clientes o usuarios del sistema.

4.4.2 Requerimientos del Sistema

En esta sección se describirá mediante tablas cada uno de los servicios y restricciones que el sistema debe de cumplir para satisfacer los requerimientos del usuario de TE ATIENDO. Esta sección está orientada al personal encargado de su desarrollo.

#	Requerimientos del usuario
1	El sistema Te Atiendo deberá de autenticar a cada uno de los usuarios , mediante su login y password
2	El Sistema Te Atiendo deberá de permitir el uso de servicios de acuerdo a los permisos de cada usuario o grupo
3	El sistema Te Atiendo deberá de contar con unos módulos maestros que permitan alimentar otros módulos del software estos archivos son. (Proceso, Subproceso, Programa, Beneficio, Requisito, Empleado, Barrio, Paquete Beneficios, Novedad Discapacidad, Centro de atención)
4	El sistema Te Atiendo deberá permitir la carga de archivos masivos (Sisben, Beneficiarios, Beneficios Otorgados)
5	El sistema Te Atiendo deberá permitir Ingresar, eliminar, modificar y actualizar Aspirantes
6	El sistema Te Atiendo deberá permitir Ingresar, eliminar, modificar y actualizar Beneficios
7	El sistema Te Atiendo deberá permitir Ingresar, eliminar, modificar y actualizar Personas en Situación de discapacidad
8	El sistema Te Atiendo deberá permitir realizar seguimiento a los beneficiarios.
9	El sistema Te Atiendo deberá permitir realizar consultas según el usuario sobre los beneficios otorgados
10	El sistema Te Atiendo deberá permitir generar reportes en Excel, PDF y Graficos.
11	El Sistema Te Atiendo deberá contar con interfaces amigables e intuitivas para facilitar el uso
12	El Sistema Te Atiendo deberá de garantizar la disponibilidad e integridad de la información en todo momento así como el acceso a ella desde cualquier parte del mundo.

Tabla 4.2: Requerimientos usuario TE ATIENDO

En las siguientes tablas, se identificaran los requerimientos del sistema que corresponden a cada requerimiento de los usuarios. Todos los requerimientos están descritos en lenguaje natural

Requerimiento del usuario	
1	El sistema Te Atiendo deberá de autenticar a cada uno de los usuarios , mediante su login y password
Requerimientos del Sistema	
1.1	El sistema Te Atiendo deberá contar con una base de datos de usuarios donde se almacenen los usuarios que tienen derecho al uso del sistema, contendrá información básica, así como el proceso y los programas de los que hará parte y de los privilegios con los que cuenta.
1.2	El sistema Te Atiendo deberá incluir una interfaz de identificación, para que el usuario introduzca sus datos.
1.3	El sistema Te Atiendo deberá verificar que el login y el password sean válidos comparándolos con los existentes en la base de datos de usuarios.
1.4	Deacurdo con el tipo de usuario, según el grupo, el proceso y el programa Te Atiendo deberá permitir el acceso a los diferentes servicios del sistema.
1.5	La tabla usuario debe de cumplir con auditoria de registros

Tabla 4.3: Requerimientos del sistema correspondiente al requerimiento del usuario 1

Requerimiento del usuario	
2	El Sistema Te Atiendo deberá de permitir el uso de servicios de acuerdo a los permisos de cada usuario o grupo
Requerimientos del Sistema	
2.1	El sistema Te Atiendo deberá mostrar un menú con todos los servicios disponibles para el usuario
2.2	El sistema Te Atiendo deberá restringir al usuario el acceso no autorizado a los servicios de acuerdo al tipo de usuario.
2.3	El sistema Te Atiendo deberá contar con un menú de preferencias donde se encuentren las configuraciones del sistema y la información del usuario.

Tabla 4.4: Requerimientos del sistema correspondiente al requerimiento del usuario 2

Requerimiento del usuario	
3	El sistema Te Atiendo deberá de contar con unos módulos maestros que permitan alimentar otros módulos del software estos archivos son. (Proceso, Subproceso, Programa, Beneficio, Requisito, Empleado, Barrio, Paquete Beneficios, Novedad Discapacidad, Centro de atención)
Requerimientos del Sistema	
3.1	El sistema Te Atiendo deberá contar con unas tablas de: procesos, subproceso, Programa, Beneficio, Requisito, Empleado, Barrio, Paquete Beneficios, Novedad Discapacidad, Centro de Atención en donde se almacenen datos principales que serán utilizados por los demás módulos
3.2	El sistema Te Atiendo deberá incluir una interfaz, para que el usuario introduzca los datos solicitados por cada tabla
3.3	El sistema Te Atiendo guardara en la base de datos los nuevos registros asociados con las tablas.
3.4	El sistema Te Atiendo deberá proporcionar una interfaz que les permita a los usuarios crear, modificar eliminar y ver registros.
3.5	La tablas deben de cumplir con auditoria de registros

Tabla 4.5: Requerimientos del sistema correspondiente al requerimiento del usuario 3

Requerimiento del usuario	
4	El sistema Te Atiendo deberá permitir la carga de archivos masivos (Sisben, Beneficiarios, Beneficios Otorgados)
Requerimientos del Sistema	
4.1	Si se trata de cargar información de sisben, Beneficiarios y Beneficios otorgados, el sistema Te Atiendo proporcionara una interfaz con los campos necesarios para la carga masiva
4.2	El sistema Te Atiendo guardara en la base de datos los nuevos registros asociados con las tablas.
4.3	La tablas deben de cumplir con auditoria de registros

Tabla 4.6: Requerimientos del sistema correspondiente al requerimiento del usuario 4

Requerimiento del usuario	
5	El sistema Te Atiendo deberá permitir Ingresar, eliminar, modificar y actualizar Aspirantes
Requerimientos del Sistema	
5.1	Si se trata de un registro nuevo Aspirante, El sistema Te Atiendo proporcionara una interfaz con los campos necesarios para el registro.
5.2	El Sistema Te Atiendo guardara en la base de datos los nuevos registros asociados con el Aspirante
5.3	Si se trata de eliminar un Aspirante, el sistema Te Atiendo proporcionara una interfaz para capturar los datos del Aspirante a ser eliminado
5.4	El Sistema Te Atiendo validara que el identificador del Aspirante pertenezca a un usuario registrado
5.5	El Sistema Te Atiendo eliminara los registros de la base de datos relacionados con el usuario.
5.6	Si se trata de una modificación de la información del Aspirante, el sistema Te Atiendo deberá de proporcionar una interfaz en la que se pueda acceder a la información del Aspirante, brindándole la modificación de dichos datos.
5.7	El sistema Te Atiendo verificara que los nuevos datos sean correctos, en caso contrario se mostrara un aviso al usuario con los datos incorrectos.
5.8	El sistema Te Atiendo actualizara los registros de la Base de datos con la información proporcionada por el usuario.
5.9	La tablas deben de cumplir con auditoria de registros

Tabla 4.7: Requerimientos del sistema correspondiente al requerimiento del usuario 5

Requerimiento del usuario	
6	El sistema Te Atiendo deberá permitir Ingresar, eliminar, modificar y actualizar Beneficios
Requerimientos del Sistema	
6.1	Si se trata de un registro nuevo Beneficio El sistema Te Atiendo proporcionara una interfaz con los campos necesarios para el registro.
6.2	El Sistema Te Atiendo guardara en la base de datos los nuevos registros asociados con el Aspirante y el Beneficio
6.3	Si se trata de eliminar un Beneficio, el sistema Te Atiendo proporcionara una interfaz para capturar los datos del Aspirante y el Beneficio a ser eliminado

6.4	El Sistema Te Atiendo validara que el identificador del Beneficio pertenezca a un usuario registrado
6.5	El Sistema Te Atiendo eliminara los registros de la base de datos relacionados con el Beneficio
6.6	Si se trata de una modificación de la información del Beneficio, el sistema Te Atiendo deberá de proporcionar una interfaz en la que se pueda acceder a la información del Beneficio, brindándole la modificación de dichos datos.
6.7	El sistema Te Atiendo verificara que los nuevos datos sean correctos, en caso contrario se mostrara un aviso al usuario con los datos incorrectos.
6.8	El sistema Te Atiendo actualizara los registros de la Base de datos con la información proporcionada por el usuario.
6.9	La tablas deben de cumplir con auditoria de registros

Tabla 4.8: Requerimientos del sistema correspondiente al requerimiento del usuario 6

Requerimiento del usuario	
7	El sistema Te Atiendo deberá permitir Ingresar, eliminar, modificar y actualizar Personas en Situación de discapacidad
Requerimientos del Sistema	
7.1	Si se trata de un registro nuevo de Situación de discapacidad. El sistema Te Atiendo proporcionara una interfaz con los campos necesarios para el registro.
7.2	El Sistema Te Atiendo guardara en la base de datos los nuevos registros asociados con el Aspirante y la Situación de discapacidad
7.3	Si se trata de eliminar un Situación Discapacidad, el sistema Te Atiendo proporcionara una interfaz para capturar los datos del Aspirante y la Situación de discapacidad o a ser eliminada
7.4	El Sistema Te Atiendo validara que el identificador dela Situación de la Discapacidad pertenezca a un usuario registrado
7.5	El Sistema Te Atiendo eliminara los registros de la base de datos relacionados con la Situación de Discapacidad
7.6	Si se trata de una modificación de la información de la Situación de la Discapacidad, el sistema Te Atiendo deberá de proporcionar una interfaz en la que se pueda acceder a la información de la situación de discapacidad, brindándole la modificación de dichos datos.
7.7	El sistema Te Atiendo verificara que los nuevos datos sean correctos, en caso contrario se mostrara un aviso al usuario con los datos incorrectos.
7.8	El sistema Te Atiendo actualizara los registros de la Base de datos con la información proporcionada por el usuario.
7.9	La tablas deben de cumplir con auditoria de registros

Tabla 4.9: Requerimientos del sistema correspondiente al requerimiento del usuario 7

Requerimiento del usuario	
8	El sistema Te Atiendo deberá permitir realizar seguimiento a los beneficiarios
Requerimientos del Sistema	
8.1	El sistema Te Atiendo debe de proporcionar una interfaz que permita llevar una agenda con los beneficios que estén pendientes de atender por parte del grupo al que se le asigna.

8.2	Si se trata de un registro nuevo. El sistema Te Atiendo proporcionara una interfaz con los campos necesarios para el registro el cual llevara una historia de cada una de las acciones que se realicen, por cada una de las personas.
8.3	El Sistema Te Atiendo guardara en la base de datos los nuevos registros asociados con el Aspirante
8.4	Si se trata de eliminar un registro, el sistema Te Atiendo proporcionara una interfaz para capturar los datos del Aspirante a ser eliminado
8.5	El Sistema Te Atiendo validara que el identificador del Aspirante pertenezca a un usuario registrado
8.6	El Sistema Te Atiendo eliminara los registros de la base de datos relacionados con la Historia del Aspirante.
8.7	Si se trata de una modificación de la información de la Historia del Aspirante, el sistema Te Atiendo deberá de proporcionar una interfaz en la que se pueda acceder a la información de la historia, brindándole la modificación de dichos datos.
8.8	El sistema Te Atiendo verificara que los nuevos datos sean correctos, en caso contrario se mostrara un aviso al usuario con los datos incorrectos.
8.9	El sistema Te Atiendo actualizara los registros de la Base de datos con la información proporcionada por el usuario.
8.10	La tablas deben de cumplir con auditoria de registros

10Tabla 4.10: Requerimientos del sistema correspondiente al requerimiento del usuario 8

Requerimiento del usuario	
9	El sistema Te Atiendo deberá permitir realizar consultas según el usuario sobre los beneficios otorgados
Requerimientos del Sistema	
9.1	El sistema Te Atiendo debe de proporcionar una interfaz que permita revisar los beneficios otorgados al usuario
9.2	Si se trata de un registro nuevo. El sistema Te Atiendo proporcionara una interfaz con los campos necesarios para el registro el cual llevara una historia de cada una de las acciones que se realicen, por cada una de las personas.
9.3	El Sistema Te Atiendo guardara en la base de datos los nuevos registros asociados con el Aspirante
9.4	Si se trata de eliminar un registro, el sistema Te Atiendo proporcionara una interfaz para capturar los datos del Beneficio a ser eliminado
9.5	El Sistema Te Atiendo validara que el identificador del Aspirante pertenezca a un usuario registrado
9.6	El Sistema Te Atiendo eliminara los registros de la base de datos relacionados con la Historia del Aspirante.
9.7	Si se trata de una modificación de la información de la Historia del Aspirante, el sistema Te Atiendo deberá de proporcionar una interfaz en la que se pueda acceder a la información de la historia, brindándole la modificación de dichos datos y permitiendo subir evidencias.
9.8	El sistema Te Atiendo verificara que los nuevos datos sean correctos, en caso contrario se mostrara un aviso al usuario con los datos incorrectos.
9.9	El sistema Te Atiendo actualizara los registros de la Base de datos con la información proporcionada por el usuario.
9.10	La tablas deben de cumplir con auditoria de registros

Tabla 4.11: Requerimientos del sistema correspondiente al requerimiento del usuario 9

Requerimiento del usuario	
10	El sistema Te Atiendo deberá permitir generar reportes en Excel, PDF y Gráficos.
Requerimientos del Sistema	
10.1	El sistema Te Atiendo debe de proporcionar una interfaz que permita generar reportes
10.2	El sistema Te Atiendo debe de proporcionar Reportes en Excel de personas discapacitadas con apoderado
10.3	El sistema Te Atiendo debe de proporcionar Reportes en Excel de personas discapacitadas sin apoderado
10.4	El sistema Te Atiendo debe de proporcionar Reportes en Excel de personas inscritas en un paquete
10.5	El sistema Te Atiendo debe de proporcionar Reportes Gráficos por fechas vs beneficios
10.6	El sistema Te Atiendo debe de proporcionar Reportes Gráficos por beneficios vs programas
10.7	El sistema Te Atiendo debe de proporcionar Reportes Gráficos por número de beneficios
10.8	El sistema Te Atiendo debe de proporcionar Reportes en PDF por beneficios por número de identidad del beneficiario
10.9	El sistema Te Atiendo debe de proporcionar Reportes en PDF por cantidad de beneficios por número de identidad del beneficiario
10.10	El sistema Te Atiendo debe de proporcionar Reportes en PDF por beneficios por responsable
10.11	El sistema Te Atiendo debe de proporcionar Reportes en PDF por beneficios por programa
10.12	El sistema Te Atiendo debe de proporcionar Reportes en PDF por beneficios no Asignados
10.13	El sistema Te Atiendo debe de proporcionar Reportes en PDF por aspirantes de un paquete beneficio
10.14	El sistema Te Atiendo debe de proporcionar Reportes en PDF por Beneficiarios con seguimiento
10.15	El sistema Te Atiendo debe de proporcionar Reportes en PDF por Beneficios entregados a personas (nativos o no nativos)
10.16	El sistema Te Atiendo debe de proporcionar Reportes en PDF por requisitos por beneficio
10.17	El sistema Te Atiendo debe de proporcionar Reportes en PDF por personas de mayor a menor número de beneficios
10.18	El sistema Te Atiendo debe de proporcionar Reportes en PDF de certificados de Historias-Usuarios

Tabla 4.12: Requerimientos del sistema correspondiente al requerimiento del usuario 10

Requerimiento del usuario	
11	El Sistema Te Atiendo deberá contar con interfaces amigables e intuitivas para

	facilitar el uso
Requerimientos del Sistema	
11.1	El sistema Te Atiendo deberá contar con interfaces interactivas con el usuario a base de ventanas, botones, listas, y menús
11.2	Las Interfaces de Te Atiendo deberán ser sencillas y claras para el usuario, con procedimientos cortos e integrales.

Tabla 4.13: Requerimientos del sistema correspondiente al requerimiento del usuario 11

Requerimiento del usuario	
12	El Sistema Te Atiendo deberá de garantizar la disponibilidad e integridad de la información en todo momento así como el acceso a ella desde cualquier parte del mundo.
Requerimientos del Sistema	
12.1	El sistema Te Atiendo deberá contar con una capa de administración de datos, para evitar cualquier anomalía en el manejo de la base de datos.
12.2	El sistema Te Atiendo deberá contara con un mecanismo de acceso que permita múltiples usuarios conectados en un mismo momento, dotándolo de capacidad para atender peticiones de los usuarios
12.3	El sistema Te Atiendo debe de ser compatible con la mayoría de navegadores Web para el acceso externo e interno.

Tabla 4.14: Requerimientos del sistema correspondiente al requerimiento del usuario 12

4.4.3 Especificación de los requerimientos

La especificación de los requerimientos es la forma de detallar cada uno de los requerimientos para su aclaración y fácil rastreo en caso de detectar errores o ambigüedades en el diseño o implementación.

A continuación se detallan los requerimientos del proyecto Te Atiendo.

Requerimiento # 1
Función: Identificar Usuarios
Descripción: El sistema identificara varios usuarios, teniendo como usuario principal al administrador.
Entradas: Login y Password
Salidas: Aseso al Sistema
Requerimientos: El usuario debe de estar registrado en la base de datos
Precondiciones: Que existan registros de los usuarios**,** los cuales deben de estar asociados a un grupo y contar con uno de los siguientes permisos (Admin,DBO, SuperDBO, User; SuperUser, Viewer)
Postcondiciones: Uso de los servicios del sistema disponibles para el tipo de usuario.
Efectos Colaterales: Ninguno

Tabla 4.15: Especificación de requerimiento # 1

Requerimiento # 2 **Función:** Uso de Servicios **Descripción:** El sistema permitirá el uso de diferentes servicios con los que el sistema cuenta, distinguiendo el tipo de usuario. Les mostrara un menú con los servicios que el usuario está autorizado a usar. **Entradas:** Tipo usuario por grupo **Salidas:** Listado de servicios disponibles para el usuario **Requerimientos**: El usuario debe de estar registrado en la base de datos y estar asociado a un grupo **Precondiciones:** Que el usuario este validado en el sistema **Postcondiciones**: Listado y uso de los servicios del sistema **Efectos Colaterales:** Restringir el uso de los servicios a usuarios no autorizados.

Tabla 4.16: Especificación de requerimiento # 2

Requerimiento # 3 **Función:** Archivos Maestros **Descripción:** Esta función permitirá, agregar, modificar, eliminar y buscar la información de las tablas maestras como lo son Proceso, Subproceso, Programa, Beneficio, Requisito, Empleado, Barrio, Paquete Beneficios, Novedad Discapacidad, Centro de atención, en la base de datos del sistema. Además de verificar que la información proporcionada sea correcta ya que estará asociada con otros módulos de la aplicación **Entradas:** Id y descripción **Salidas:** Agregar, modificar, eliminar y buscar registros de los archivos maestros. **Requerimientos**: El usuario debe de estar registrado en el sistema **Precondiciones:** : En caso de agregar un registro se debe de contar con toda la información, en caso de modificar no permitir modificar el ID y que el registro si este en la base de datos, en caso de eliminar verificar que el registro si este en la base de datos **Postcondiciones**: En caso de un nuevo registro, el registro debe de ser agregado a la base de datos del sistema para algunas tablas debe de validar que se cumpla con los requerimientos de datos previos ya que la información depende de otra tabla, en caso de eliminarse, el registro debe de ser eliminado de la base de datos del sistema y en caso de medicación, se actualizara la información relacionada en los registros de las tablas. **Efectos Colaterales:** Si no se cumple con los requisitos para agregar, modificar, eliminar y actualizar un nuevo registro del aspirante se capturara el error en el sistema y no se podrá llevar a cabo la operación.

Tabla 4.17: Especificación de requerimiento # 3

Requerimiento # 4

Función: Cargar Registros
Descripción: Esta función permitirá, cargar, validar y adicionar registros a las tablas de Sisben, Beneficiarios, Beneficios otorgados
Entradas: Id y descripción
Salidas: Agregar Registros a tablas.
Requerimientos: El usuario debe de estar registrado en el sistema
Precondiciones: En caso de agregar un registro se debe de contar con toda la información el archivo debe ser delimitado por puntos y comas (;),además de contar con la extensión .csv.
Postcondiciones: En caso de un nuevo registro, el registro debe de ser agregado a la base de datos del sistema para algunas tablas debe de validar que se cumpla con los requerimientos de datos previos ya que la información depende de otra tabla, se actualizara la información relacionada en los registros de las tablas.
Efectos Colaterales: Si no se cumple con los requisitos para agregar un registro se producirá un error en el sistema.

Tabla 4.18: Especificación de requerimiento # 4

Requerimiento # 5
Función: Agregar, Modificar, Eliminar y Actualizar Aspirantes
Descripción: Esta función permitirá, agregar, modificar, eliminar y actualizar la información de cada uno de los aspirantes en la base de datos del sistema Además de verificar que la información proporcionada sea correcta ya que estará asociada con otros módulos de la aplicación y es la información base para construir la Historia de Beneficios.
Entradas: Id aspirante
Salidas: Agregar, modificar, eliminar y actualizar registros de los aspirantes
Requerimientos: El usuario debe de estar registrado en el sistema
Precondiciones: : En caso de agregar un registro se debe de contar con toda la información, en caso de modificar no permitir modificar el ID y que el registro si este en la base de datos, en caso de eliminar verificar que el registro si este en la base de datos
Postcondiciones: En caso de un nuevo registro, el registro debe de ser agregado a la base de datos del sistema para algunas tablas debe de validar que se cumpla con los requerimientos de datos previos ya que la información depende de otra tabla, en caso de eliminarse, el registro debe de ser eliminado de la base de datos del sistema y en caso de medicación, se actualizara la información relacionada en los registros de las tablas.
Efectos Colaterales: Si no se cumple con los requisitos para agregar, modificar, eliminar y actualizar un nuevo registro del aspirante se capturara el error en el sistema y no se podrá llevar a cabo la operación.

Tabla 4.19: Especificación de requerimiento # 5

Requerimiento # 6
Función: Ingresar, eliminar, modificar y actualizar Beneficios
Descripción: Esta función permitirá, agregar, modificar, eliminar y actualizar la información de cada uno de los beneficios en la base de datos del sistema Además de verificar que la información proporcionada sea correcta ya que estará asociada con

otros módulos de la aplicación y es la información base para construir la Historia de Beneficios. **Entradas:** Id beneficio **Salidas:** Agregar, modificar, eliminar y actualizar registros de los aspirantes **Requerimientos**: El usuario debe de estar registrado en el sistema **Precondiciones:** : En caso de agregar un registro se debe de contar con toda la información, en caso de modificar no permitir modificar el ID y que el registro si este en la base de datos, en caso de eliminar verificar que el registro si este en la base de datos **Postcondiciones**: En caso de un nuevo registro, el registro debe de ser agregado a la base de datos del sistema para algunas tablas debe de validar que se cumpla con los requerimientos de datos previos ya que la información depende de otra tabla, en caso de eliminarse, el registro debe de ser eliminado de la base de datos del sistema y en caso de medicación, se actualizara la información relacionada en los registros de las tablas. **Efectos Colaterales:** Si no se cumple con los requisitos para agregar, modificar, eliminar y actualizar un nuevo registro del beneficio se capturara el error en el sistema y no se podrá llevar a cabo la operación.

Tabla 4.20: Especificación de requerimiento # 6

Requerimiento # 7 **Función:** Ingresar, eliminar, modificar y actualizar Personas en Situación de discapacidad **Descripción:** Esta función permitirá, agregar, modificar, eliminar y actualizar la información de cada uno de los Aspirantes en situación de discapacidad en la base de datos del sistema Además de verificar que la información proporcionada sea correcta ya que estará asociada con otros módulos de la aplicación y es la información base para construir la Historia de Beneficios. **Entradas:** Id beneficiario, id aspirante **Salidas:** Agregar, modificar, eliminar y actualizar registros de los aspirantes **Requerimientos**: El usuario debe de estar registrado en el sistema **Precondiciones:** : En caso de agregar un registro se debe de contar con toda la información, en caso de modificar no permitir modificar el ID y que el registro si este en la base de datos, en caso de eliminar verificar que el registro si este en la base de datos **Postcondiciones**: En caso de un nuevo registro, el registro debe de ser agregado a la base de datos del sistema para algunas tablas debe de validar que se cumpla con los requerimientos de datos previos ya que la información depende de otra tabla, en caso de eliminarse, el registro debe de ser eliminado de la base de datos del sistema y en caso de medicación, se actualizara la información relacionada en los registros de las tablas. **Efectos Colaterales:** Si no se cumple con los requisitos para agregar, modificar, eliminar y actualizar un nuevo registro del aspirante se capturara el error en el sistema y no se podrá llevar a cabo la operación.

Tabla 4.21: Especificación de requerimiento # 7

Requerimiento # 8

Función: realizar seguimiento a los beneficiarios
Descripción: Esta función permitirá realizar seguimiento a las historia del beneficiario. Adicional, agregar, modificar, eliminar y actualizar la información de cada uno de los beneficiarios en la base de datos del sistema, debe verificar que la información proporcionada sea correcta ya que estará asociada con otros módulos de la aplicación
Entradas: id beneficiario
Salidas: consultar**,** agregar, modificar, eliminar y actualizar registros de los aspirantes
Requerimientos: El usuario debe de estar registrado en el sistema
Precondiciones: : En caso de agregar un registro se debe de contar con toda la información, en caso de modificar no permitir modificar el ID y que el registro si este en la base de datos, en caso de eliminar verificar que el registro si este en la base de datos
Postcondiciones: En caso de un nuevo registro, el registro debe de ser agregado a la base de datos del sistema para algunas tablas debe de validar que se cumpla con los requerimientos de datos previos ya que la información depende de otra tabla, en caso de eliminarse, el registro debe de ser eliminado de la base de datos del sistema y en caso de medicación, se actualizara la información relacionada en los registros de las tablas.
Efectos Colaterales: Si no se cumple con los requisitos para agregar, modificar, eliminar y actualizar un nuevo registro del aspirante se capturara el error en el sistema y no se podrá llevar a cabo la operación.

Tabla 4.22: Especificación de requerimiento # 8

Requerimiento # 9
Función: consultas según el usuario sobre los beneficios otorgados
Descripción: Esta función permitirá realizar seguimiento a las historia del beneficiario vs los beneficios otorgados Adicional, agregar, modificar, eliminar y actualizar la información de cada uno de los beneficiarios en la base de datos del sistema, debe verificar que la información proporcionada sea correcta ya que estará asociada con otros módulos de la aplicación
Entradas: id beneficiario
Salidas: consultar**,** agregar, modificar, eliminar y actualizar registros de los aspirantes
Requerimientos: El usuario debe de estar registrado en el sistema
Precondiciones: : En caso de agregar un registro se debe de contar con toda la información, en caso de modificar no permitir modificar el ID y que el registro si este en la base de datos, en caso de eliminar verificar que el registro si este en la base de datos
Postcondiciones: En caso de un nuevo registro, el registro debe de ser agregado a la base de datos del sistema para algunas tablas debe de validar que se cumpla con los requerimientos de datos previos ya que la información depende de otra tabla, en caso de eliminarse, el registro debe de ser eliminado de la base de datos del sistema y en caso de medicación, se actualizara la información relacionada en los registros de las tablas.
Efectos Colaterales: Si no se cumple con los requisitos para agregar, modificar, eliminar y actualizar un nuevo registro del beneficiario se capturara el error en el sistema y no se podrá llevar a cabo la operación.

Tabla 4.23: Especificación de requerimiento # 9

Requerimiento # 10 **Función:** Generar reportes en Excel, PDF y Gráficos. **Descripción:** Esta función permitirá generar diferentes reportes según los criterios de búsqueda **Entradas:** id beneficiario **Salidas:** consultar y reporte **Requerimientos**: El usuario debe de estar registrado en el sistema y que la base de datos contenga información **Precondiciones:** Que exista registros en la base de datos y validar los criterios por los que se está generando la consulta para el reporte. **Postcondiciones**: Que exista registros en la base de datos y que las tablas estén asociadas **Efectos Colaterales:** Si no existen registros en la base de datos. No realizara la consulta ya que no generara información

Tabla 4.24: Especificación de requerimiento # 10

4.4.4 Requerimientos no funcionales

Los requerimientos no funcionales son aquellos que no se refieren directamente a las funciones específicas que entrega el sistema. Estos requerimientos definen las restricciones que tendrá el sistema durante su desarrollo y operación.

#	Descripción
1	Requerimientos de Hardware, procesador i386 o superior, memoria RAM de 2 gigas o superior , Disco duro de 40 GB o superior (Espacio Requerido por el software 100MB)
2	Requerimientos del Software, Windows 7 o superior, IE 9.0 o superior o cualquier navegador exceptuando IE6
3	El tiempo de respuesta de sistema a cualquier transacción no debe de sobre pasar los 15 segundos
4	Debe de contar con conexión de Banda Ancha hacia internet.
5	La tasa de fallas para que el sistema sea aceptable debe ser del 2%
6	El funcionamiento del software a través de internet o la intranet debe ser independiente del tipo de usuario al que se atienda (PC, Desktop, Ipad, celular) y el navegador utilizado por este último.
7	Cada página del software debe de hacer validaciones de entrada de datos del lado del cliente, para garantizar la integridad de la información contenida en la base de datos.
8	La base de datos debe de contar con auditoria de datos.

Tabla 4.25: Especificación de Requerimientos no funcionales

4.5 Análisis y Diseño

Ayudan a modelar el comportamiento del sistema. El comportamiento del sistema va a estar definido por la interacción con el usuario u otros sistemas. Después de tener la especificación de cada uno de los requerimientos del lenguaje natural, es necesario traducir los requerimientos a un modelo que permita el desarrollo. Para una descripción grafica utilizaremos los diagramas de casos de uso y la secuencia del lenguaje del modelo unificado

En la figura 4.3 se muestra el diagrama de casos de uso del sistema, con sus respectivos actores directos, los actores representan el tipo de usuario que va a tener acceso al sistema. En nuestro caso consideramos dos tipos principales: Administrador y Proceso

Se muestran 3 casos de usos para el administrador y las acciones que son válidas para este tipo de usuario. Para TE ATIENDO, el administrador es el encargado de la administración de usuarios (Ingresar, modificar, actualizar y eliminar) Es el usuario que se le entrega a la Alcadia de Rionegro, oficina de sistemas ya que llevara el control del sistema.

Utilizaremos diagrama de secuencia para mostrar la interacción de un conjunto de objetos en la aplicación atreves del tiempo y se modelara para cada caso de uso. El diagrama de secuencia contiene detalles de implementación, incluyendo los objetos, clases, mensajes intercambiados entre los objetos. En estos diagramas se examina la descripción de un caso de uso para determinar que objetos son necesarios para la implementación. Un diagrama de secuencia muestra los objetos que intervienen con líneas discontinuas verticales y los mensajes entre los objetos como flechas horizontales.

Cada diagrama de secuencia muestra los procedimientos que el sistema debe de seguir para poder lograr una determinada tarea.

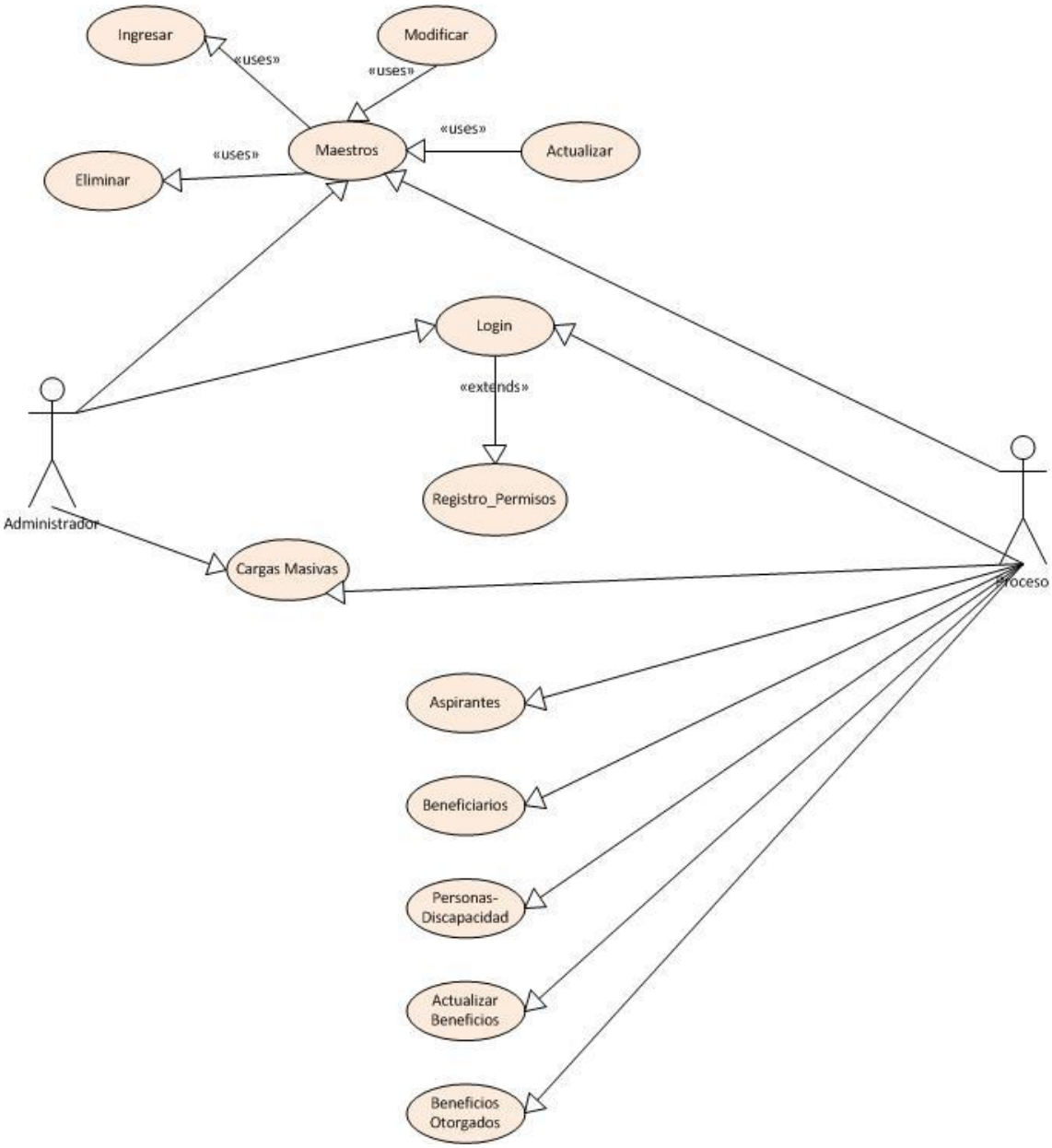


Figura 4.3: Casos de usos de Te Atiendo

La figura 4.4 muestra los diagramas de secuencia asociados al usuario administrador describiendo cada uno de los casos de uso, cada diagrama de secuencia muestra los procedimientos que el sistema debe de seguir para poder lograr una determinada tarea

La figura 4.5, 4,6 y 4,7 muestran o describen el procedimiento para poder ingresar, modificar y eliminar un registro en el sistema, el resultado de tal operación es que el usuario pueda interactuar con el sistema y sus módulos.

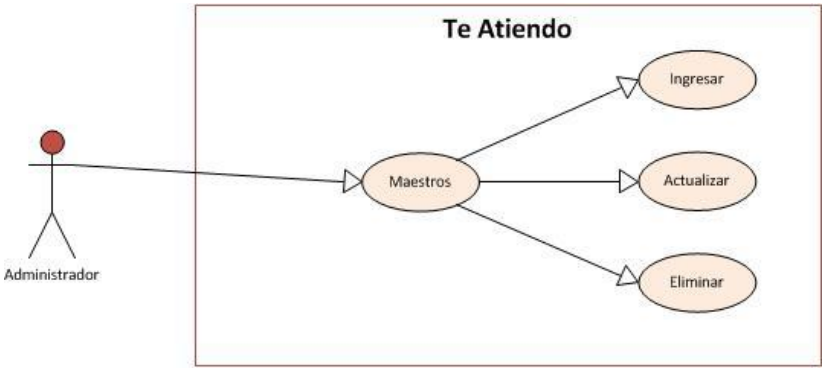


Figura 4.4: Caso de uso del actor Administrador

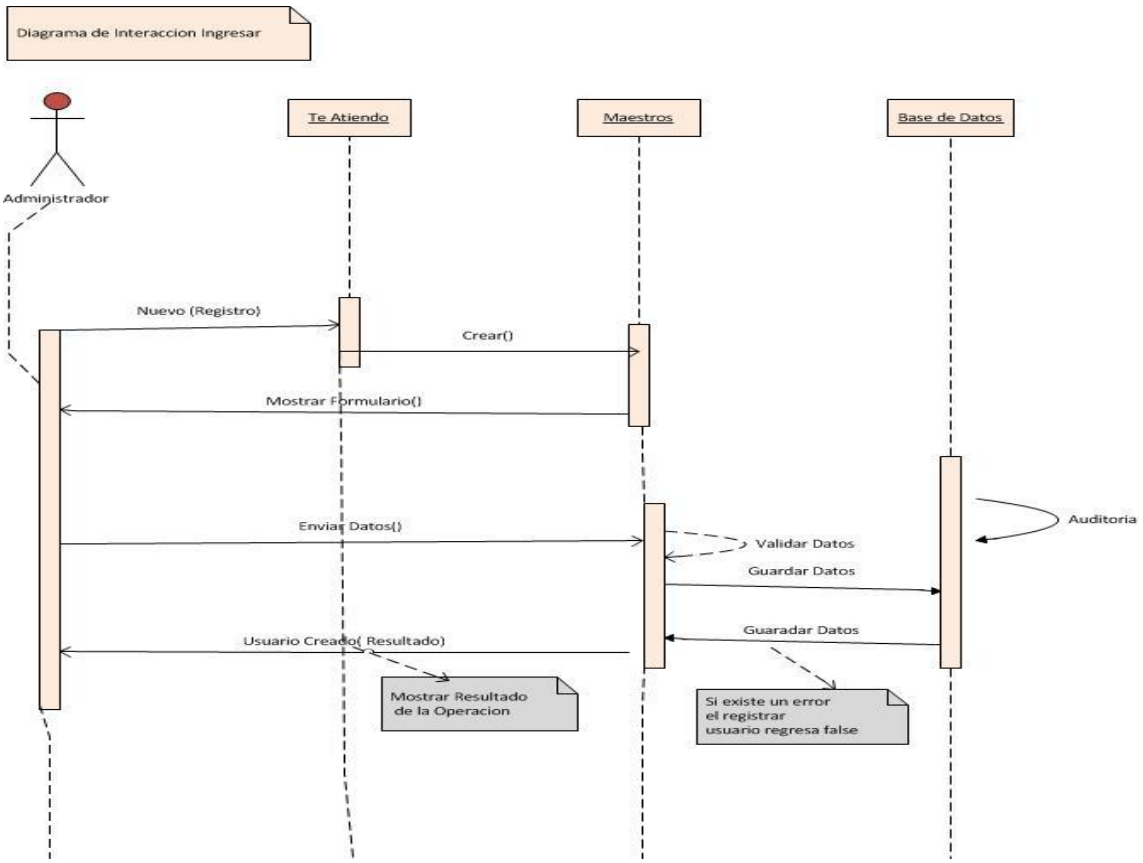


Figura 4.5: Diagrama de secuencia de Ingreso en archivos maestros

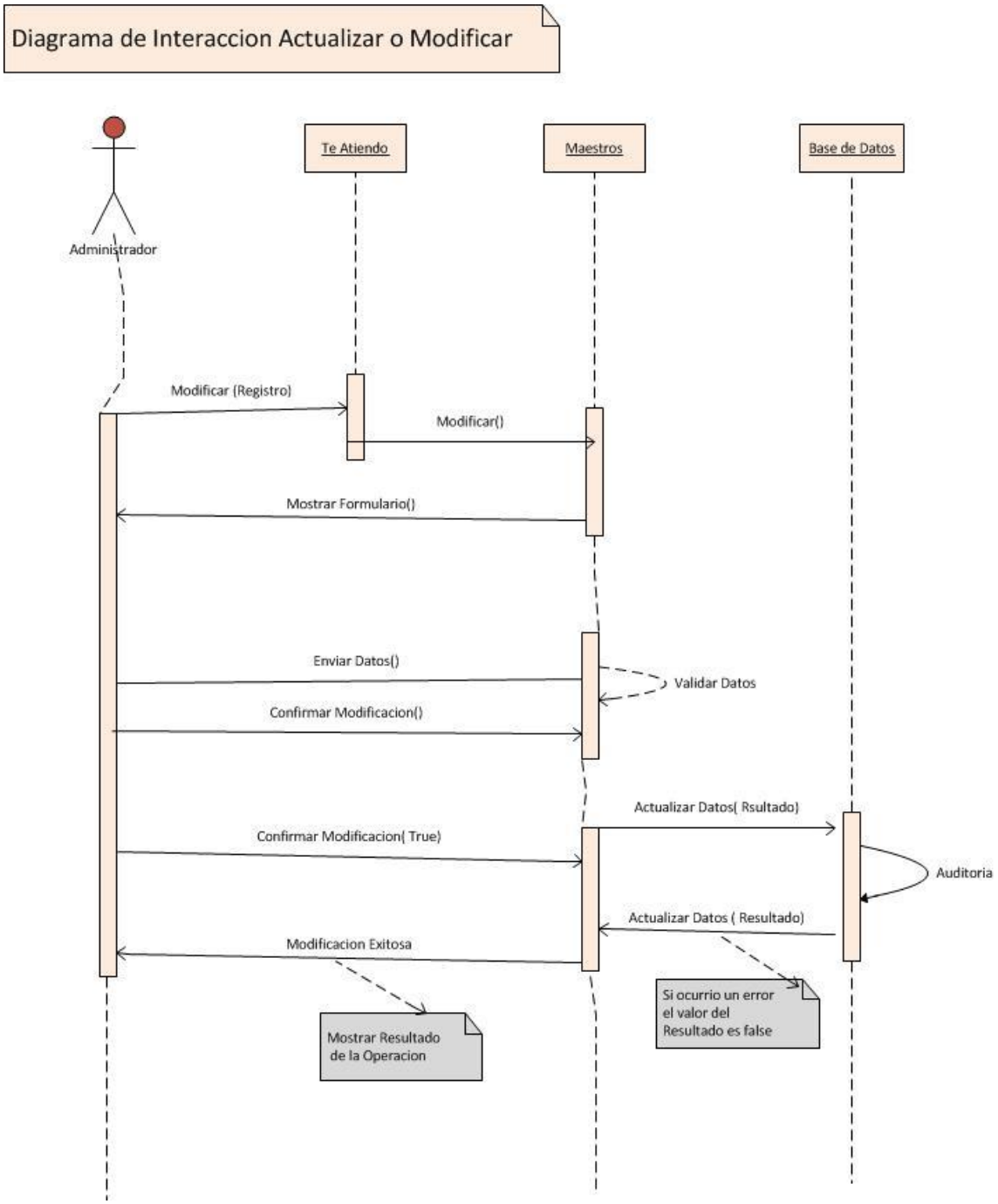


Figura 4.6: Diagrama de secuencia de modificación o actualización en archivos maestros

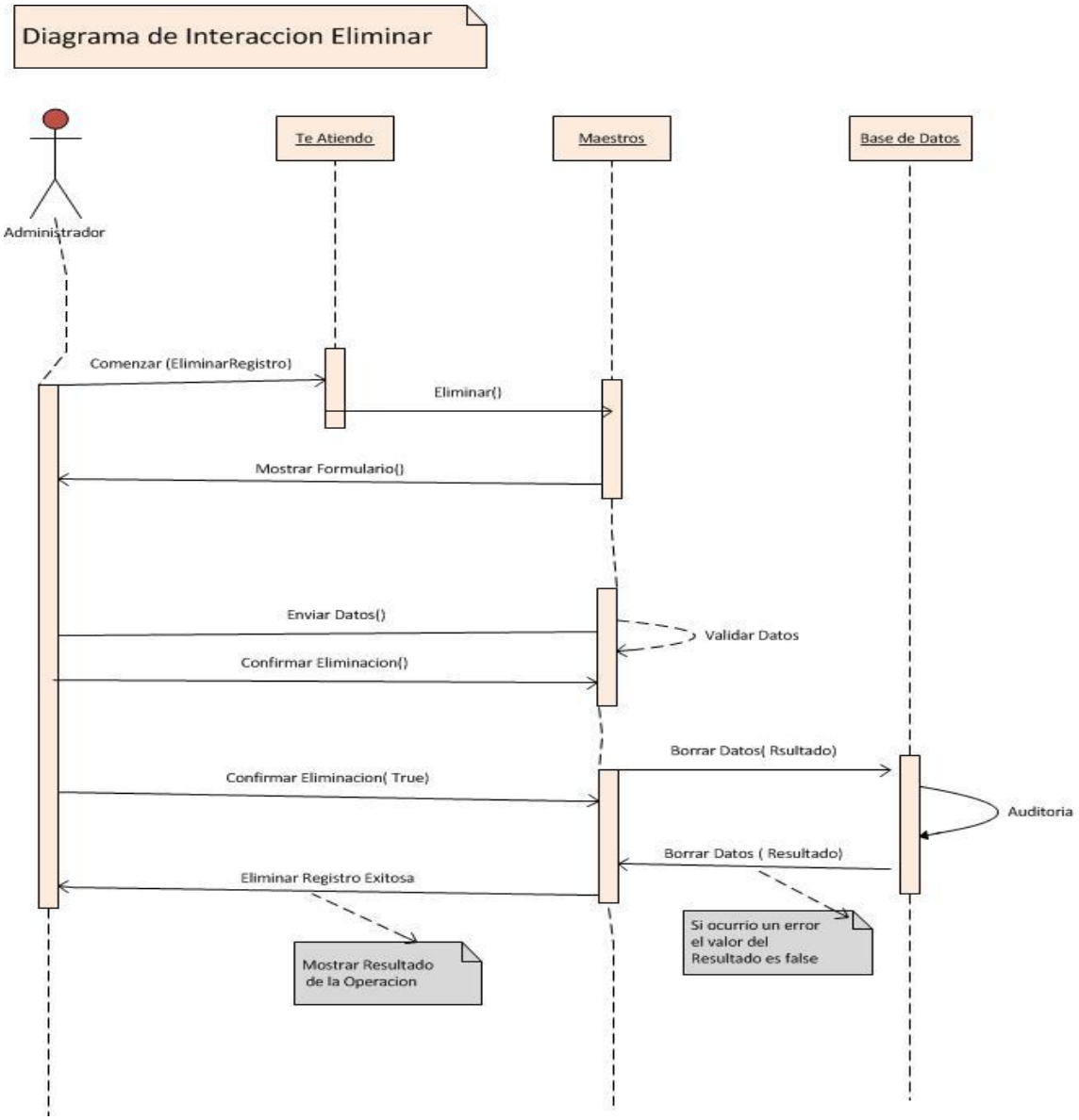


Figura 4.7: Diagrama de secuencia de eliminar archivos maestros

En las siguientes figuras se describirán las secuencias principales del actor Proceso, el cual según el subproceso y el programa puede interactuar con los diferentes módulos del sistema.

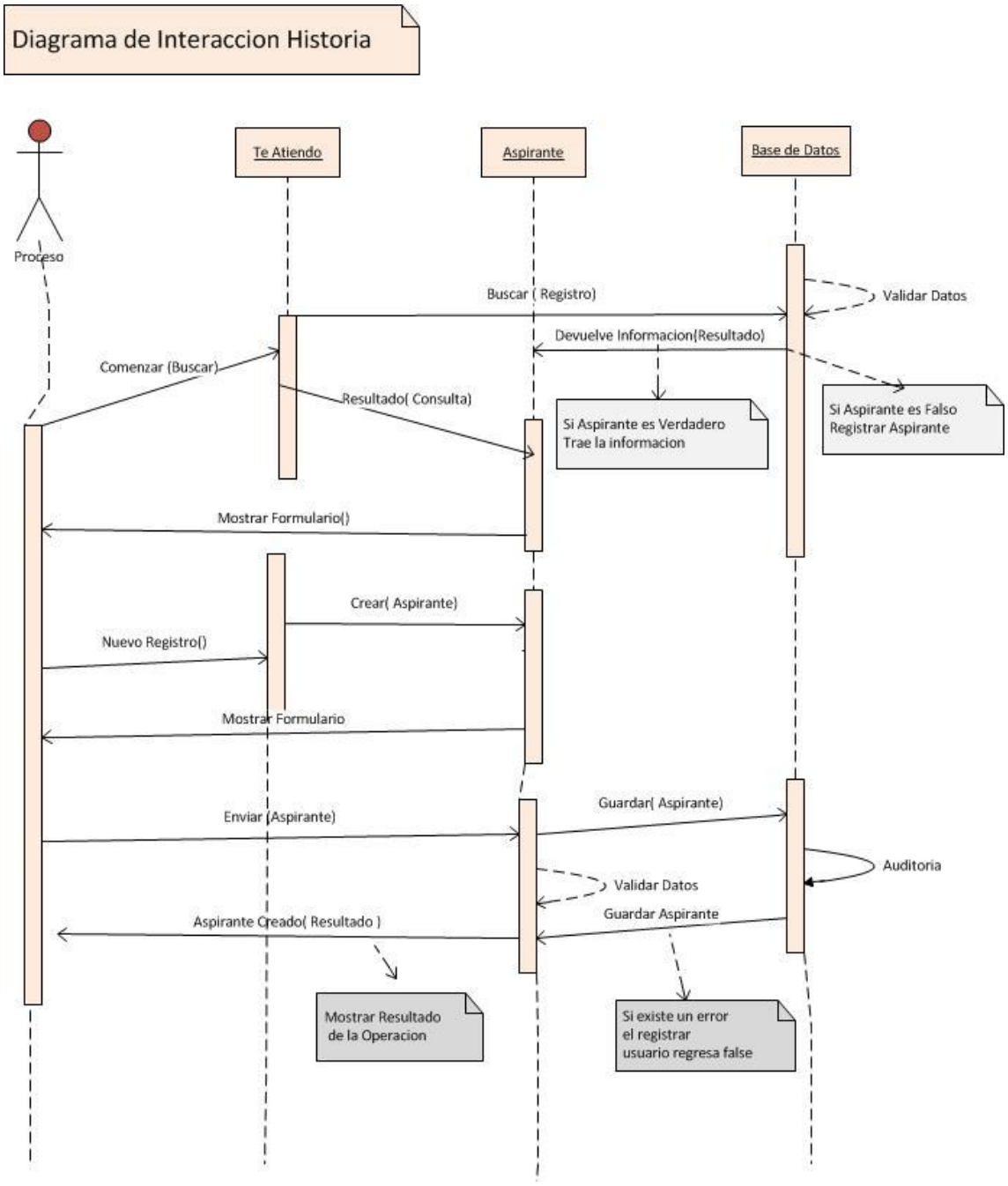


Figura 4.8: Diagrama de secuencia de Historia Aspirante

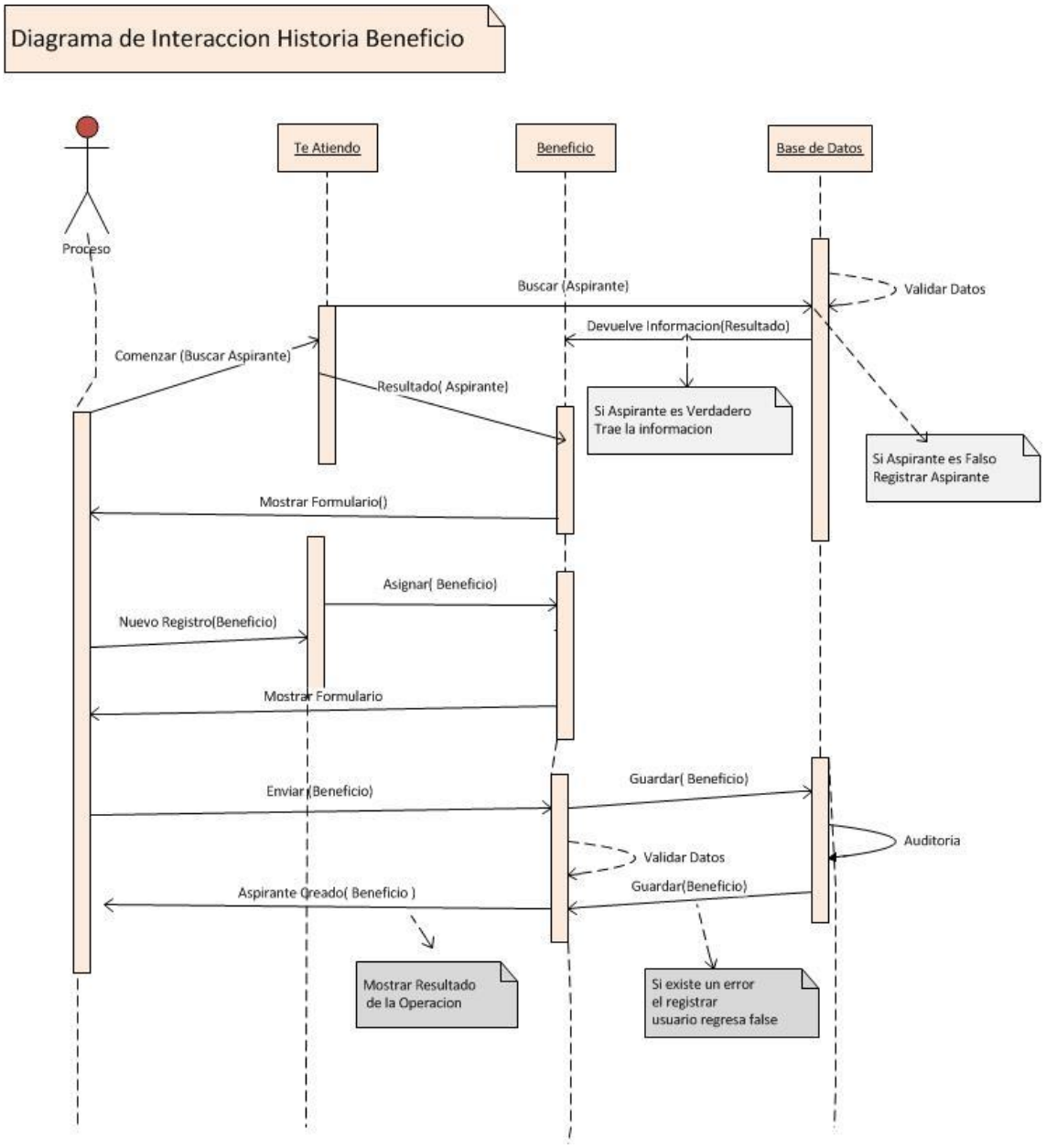


Figura 4.9: Diagrama de secuencia de Historia Aspirante Beneficio

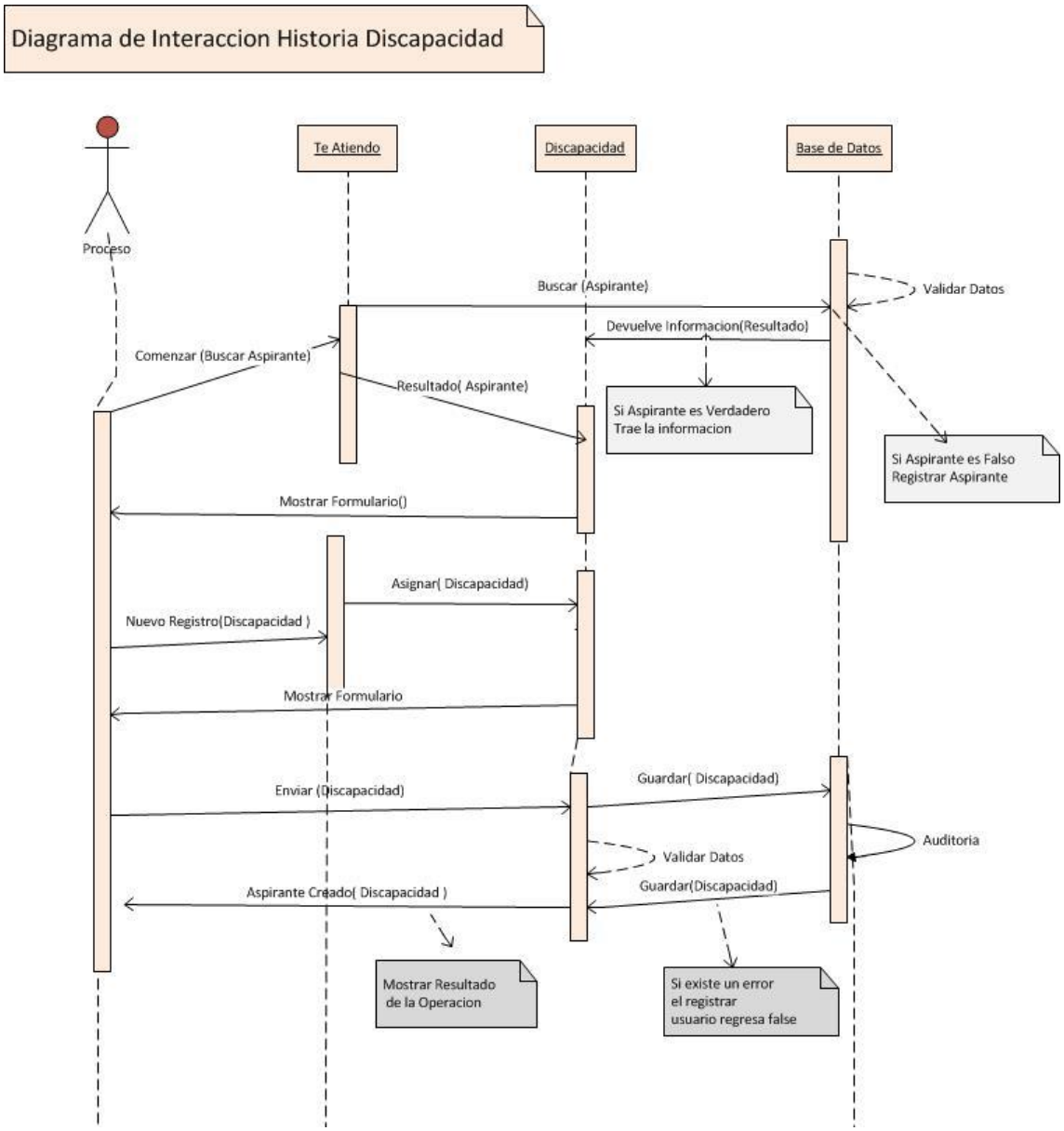


Figura 4.10: Diagrama de secuencia de Historia Aspirante Discapacidad

Base de datos de la Aplicación

La aplicación Te Atiendo cuenta con 31 tablas, el diseño se envía anexo al trabajo.

1. ASPIRANTE_BENEFICIOS (16 columnas)
2. BARRIOS (6 columnas)
3. BENEFICIARIOS (79 columnas)
4. BENEFICIO_REQUISITOS (2 columnas)
5. BENEFICIOS (19 columnas)
6. CENTROS_ATENCION (5 columnas)
7. CIUDAD (7 columnas)
8. DEPARTAMENTO (6 columnas)
9. DOCUMENTOS_ASPIRANTE (10 columnas)
10. EMPLEADOS (18 columnas)

11. EVIDENCIA_HISTORIAS (10 columnas)
12. GRUPO_PERMISOS (2 columnas)
13. GRUPOS (5 columnas)
14. HISTORIA_BENEFICIARIOS (23 columnas)
15. LOGAUDITORIA (8 columnas)
16. MOTIVOSFALLECIMIENTO (5 columnas)
17. NOVEDADES_DISCAPACIDAD (5 columnas)
18. NOVEDADESSUBSIDIO_DISCAPACIDAD (12 columnas)
19. PAQUETE_BENEFICIOS (13 columnas)
20. PERMISOS (5 columnas)
21. POBLACION_DISCAPACITADA (63 columnas)
22. PROCESOS (7 columnas)
23. PROGRAMA_EMPLEADOS (2 columnas)
24. PROGRAMAS (7 columnas)
25. REQUISITOS (5 columnas)
26. SEGUIMIENTO_HISTORIAS (21 columnas)
27. SISBEN3 (44 columnas)
28. SUBPROCESOS (8 columnas)
29. USUARIO_GRUPOS (2 columnas)
30. USUARIOS (8 columnas)
31. VEREDA (6 columnas)

En la siguiente grafica se podrán observar las tablas implementadas

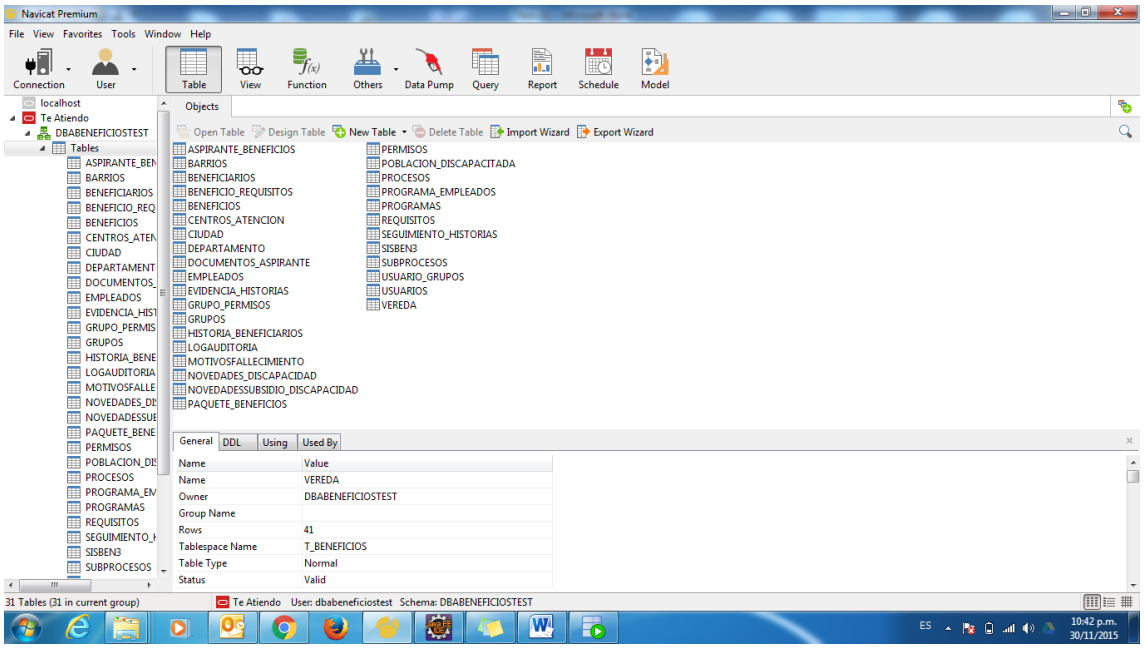


Figura 4.11: Diseño Base de Datos

Diseño de frontera más representativo

A continuación se detallan las pantallas más representativas de la aplicación, las demás se entregan como anexo a este proyecto.

Ingreso al sistema

Figura 4.12. Pantalla de ingreso de Plataforma de Beneficios.

Usuario: Digite el usuario que inscribió

Clave: Digite la clave que ingreso.

Nota: El primer usuario se hace por base de datos(Entregado por la empresa desarrolladora),ya que no es posible hacer un registro previo al momento de ingresar al sistema.

Menú principal

Figura 4.13. Pantalla del menú principal

Consulta: Seleccione esta opción si desea ingresar a consultar los datos de las personas a las que se les ha otorgado beneficios por el usuario que está en sesión.
Ingreso: Seleccione esta opción si desea ingresar un beneficiario al sistema.

Seguimiento: Seleccione esta opción si desea ver las personas que tienen seguimiento en la asignación de beneficios.

Estadística: Seleccione esta opción si desea generar reporte

Figura: 4:14. Pantalla consulta de beneficiarios por usuario

Beneficiarios por usuario. Muestra los últimos beneficiarios a los que se les ha otorgado beneficios por parte del usuario que se encuentra en sesión.

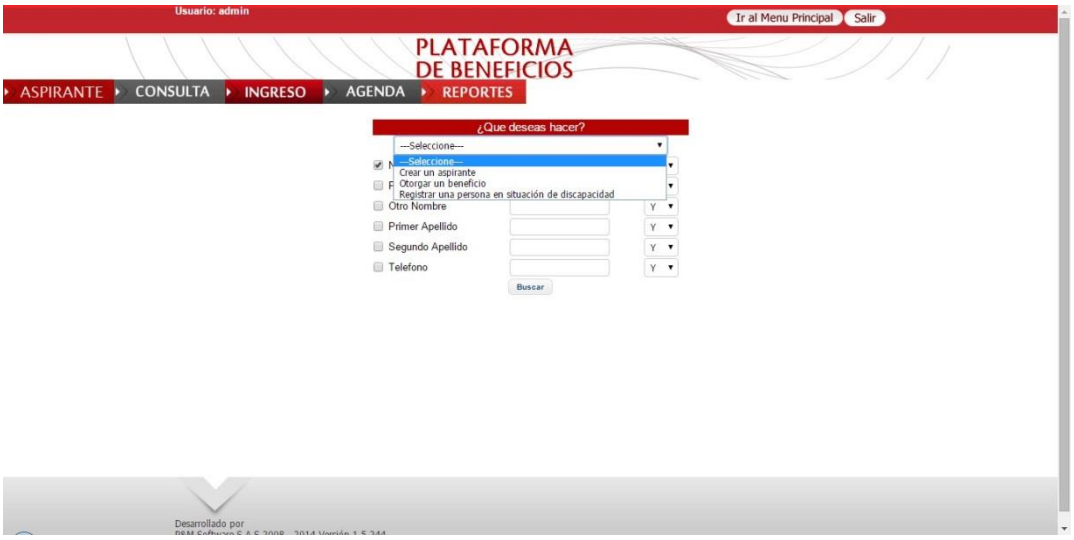


Figura: 4.15. Pantalla de búsqueda de beneficiarios.

¿Que deseas hacer?: En esta lista hay tres opciones en las cuales puede seleccionar una, dependiendo de lo que desea hacer. Además se puede hacer varios filtros de búsqueda teniendo diferentes campos en los cuales puede ingresar los datos a buscar.

Crear un Aspirante: Si selecciona esta opción ingresara una persona que va a aspirar a un beneficio.

Otorgar un Beneficio: Si selecciona esta opción es porque se le va a dar un beneficio a una persona.

Registrar una persona en situación de discapacidad: Si selecciona esta opción hará un registro de una persona en situación de discapacidad que aspira a un subsidio de discapacidad.

- *Identidad:* En este campo ingresa el número de identificación de la persona que desea buscar.
- *Primer Nombre:* En este campo ingresa el primer nombre de la persona que desea buscar.
- *Otro Nombre:* En este campo ingresa el segundo nombre de la persona que desea buscar.
- *Primer Apellido:* En este campo ingresa el primer apellido de la persona que desea buscar.
- *Segundo Apellido:* En este campo ingresa el segundo apellido de la persona que desea buscar. *Teléfono:* En este campo ingresa el

Figura 4.16. Pantalla consulta de seguimientos de los beneficiarios.

Seguimiento de Beneficiarios. Aquí se muestra los próximos seguimientos que se deben hacer a los beneficiarios, que se les otorgó un beneficio que debe llevar un seguimiento. En esta lista se muestra la fecha seguimiento en el que se asignó el seguimiento y la fecha programación, el cual se revisara si lo que se le entrego al beneficiario lo está utilizando como debe ser.

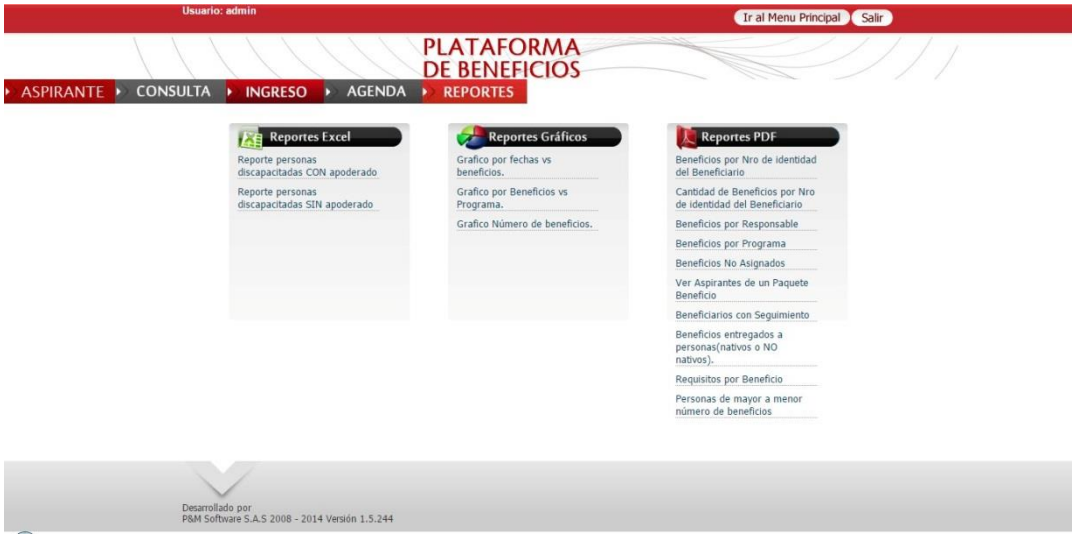


Figura 4.17 Pantalla de reportes

Paquete de análisis

Para la aplicación Te Atiendo, se realizaron los siguientes paquetes de análisis.

El contenido de cada una de las carpetas es el siguiente utilizando Maven.

src/main/java: Contiene el código fuente con nuestras clases Java incluida la estructura de paquetes

src/main/resources :Contiene ficheros de recursos como imagenes ficheros properties etc

src/test/java: Contiene el código fuente con nuestras clases Java para realizar test

src/test/resource: Contiene ficheros de recursos como imágenes ficheros properties etc. para nuestros test

target :Contiene los desplegables que generamos con Maven jar,war,ear etc

pom.xml Es el fichero encargado de definir el concepto de Artefacto.

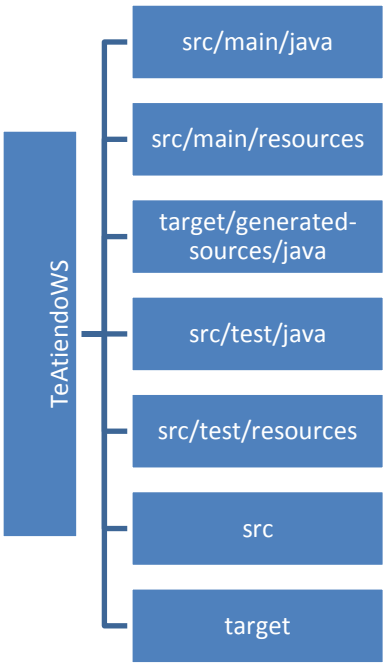


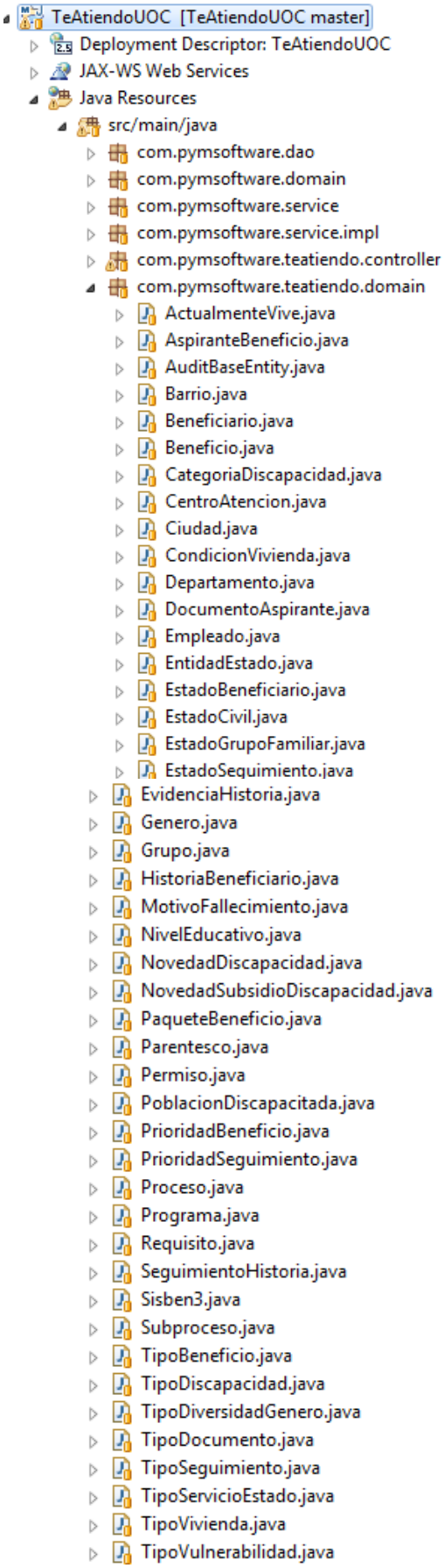
Figura 4.18: Paquete de Análisis TE ATIENDO

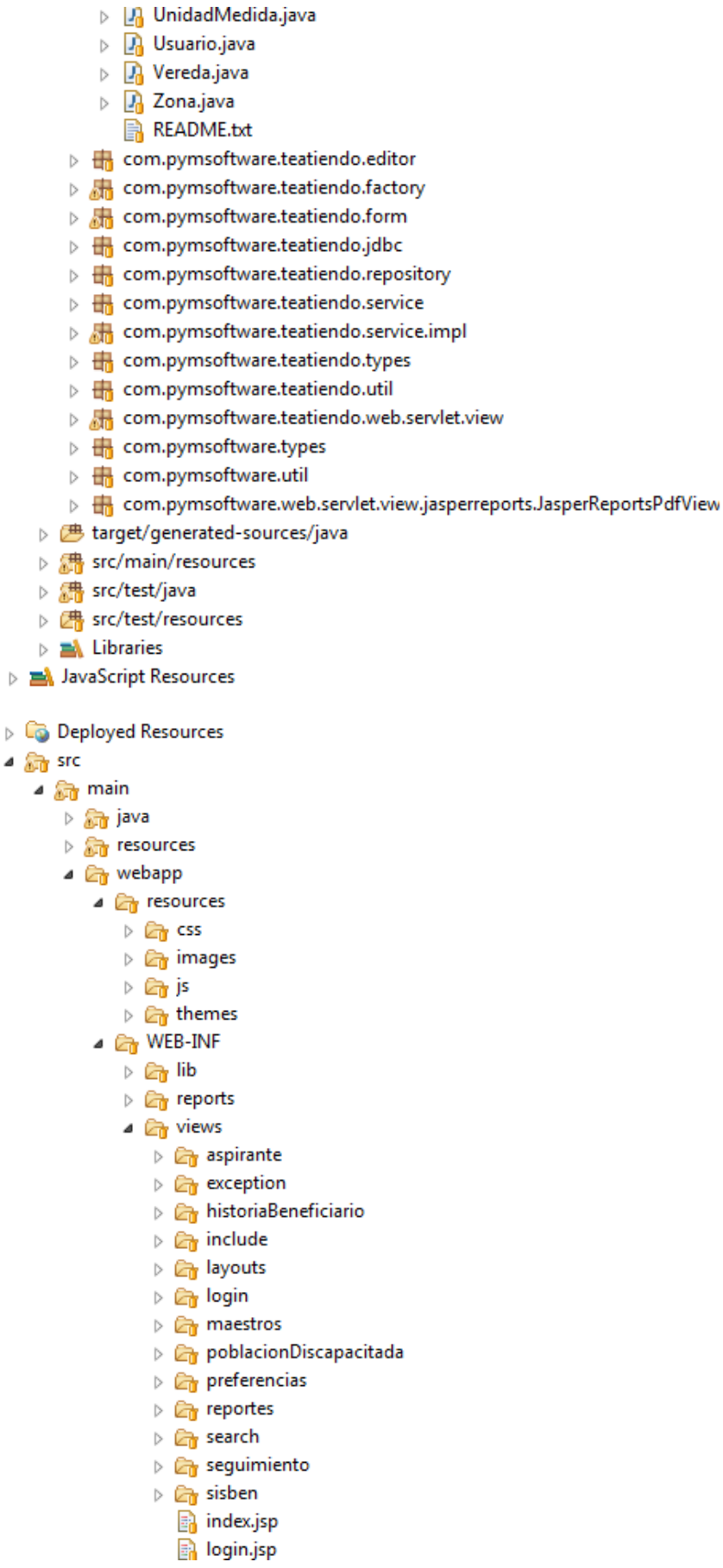
Especificación de las clases de análisis

Del caso de estudio y de los casos de uso se pueden ver las clases siguientes.

com.pymsoftware.teatiendo.domain, el análisis y la implementación se anexan al trabajo final donde se podrán detallar el uso de los patrones en la implementación del software, donde se muestran todos los componentes de la aplicación y la relación que existe entre ellos,. Esta vista permite tener el panorama real de lo que se desarrollo
En este se podrán evidenciar los componentes que pertenecen al nivel del cliente, los del servidor web, el servidor de aplicaciones, sistemas heredados y de persistencia.
En los patrones que se pueden identificar de forma rápida, se está usando un patrón de diseño COMPOSITE VIEW en la capa de presentación se usa el patrón de diseño MVC para separar la presentación de la lógica del negocio

En la capa de negocio se usaran los patrones de diseño VALUE LIST ITERATOR, para la paginación, SESSION FAÇADE, para acceder al bean de la entidad para encapsular la complejidad de las interacciones entre los objetos de negocio que participan en un flujo de trabajo. El Session Facade maneja los objetos de negocio y proporciona una capa de acceso a servicios de grano grueso uniforme a los clientes. VALUE OBJECT, para minimizar la comunicación con el servidor.
En la capa de datos se usará el patrón de diseño DAO (el cual usa el patrón FACTORY para su implementación) para encapsular código, en principio, no portable, como es el de acceso a diferentes SGBD.


TeAtiendoUOC [TeAtiendoUOC master]
Deployment Descriptor: TeAtiendoUOC
JAX-WS Web Services
Java Resources
src/main/java
com.pymsoftware.dao
com.pymsoftware.domain
com.pymsoftware.service
com.pymsoftware.service.impl
com.pymsoftware.teatiendo.controller
com.pymsoftware.teatiendo.domain
ActualmenteVive.java
AspiranteBeneficio.java
AuditBaseEntity.java
Barrio.java
Beneficiario.java
Beneficio.java
CategoriaDiscapacidad.java
CentroAtencion.java
Ciudad.java
CondicionVivienda.java
Departamento.java
DocumentoAspirante.java
Empleado.java
EntidadEstado.java
EstadoBeneficiario.java
EstadoCivil.java
EstadoGrupoFamiliar.java
EstadoSeguimiento.java
EvidenciaHistoria.java
Genero.java
Grupo.java
HistoriaBeneficiario.java
MotivoFallecimiento.java
NivelEducativo.java
NovedadDiscapacidad.java
NovedadSubsidioDiscapacidad.java
PaqueteBeneficio.java
Parentesco.java
Permiso.java
PoblacionDiscapacitada.java
PrioridadBeneficio.java
PrioridadSeguimiento.java
Proceso.java
Programa.java
Requisito.java
SeguimientoHistoria.java
Sisben3.java
Subproceso.java
TipoBeneficio.java
TipoDiscapacidad.java
TipoDiversidadGenero.java
TipoDocumento.java
TipoSeguimiento.java
TipoServicioEstado.java
TipoVivienda.java
TipoVulnerabilidad.java


UnidadMedida.java
Usuario.java
Vereda.java
Zona.java
README.txt
com.pymsoftware.teatiendo.editor
com.pymsoftware.teatiendo.factory
com.pymsoftware.teatiendo.form
com.pymsoftware.teatiendo.jdbc
com.pymsoftware.teatiendo.repository
com.pymsoftware.teatiendo.service
com.pymsoftware.teatiendo.service.impl
com.pymsoftware.teatiendo.types
com.pymsoftware.teatiendo.util
com.pymsoftware.teatiendo.web.servlet.view
com.pymsoftware.types
com.pymsoftware.util
com.pymsoftware.web.servlet.view.jasperreports.JasperReportsPdfView
target/generated-sources/java
src/main/resources
src/test/java
src/test/resources
Libraries
JavaScript Resources
Deployed Resources
src
main
java
resources
webapp
resources
css
images
js
themes
WEB-INF
lib
reports
views
aspirante
exception
historiaBeneficiario
include
layouts
login
maestros
poblacionDiscapacitada
preferencias
reportes
search
seguimiento
sisben
index.jsp
login.jsp

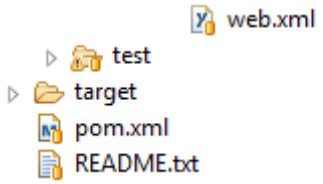


Figura 4.19: Layar de paquetes de Análisis Te Atiendo

4.6. Implementación

Para la implementación de la aplicación se utilizaron varios frameworks y patrones(*Groovy Lenguaje, Spring-framework, Spring Data JPA , MySQL , JQuery, Factory, Singleton, Builder, Adapter, MVC, Ajax, Java Server, Git, Maven, Jenkins, Eclipse, Layers* **)** que nos permitieron lo siguiente:

- Arquitectura en capas que nos permite usar lo que se necesita prescindiendo de lo que no se necesita.
- Permiten centrarnos en la programación de objetos planos java, eso permite realizar pruebas e integración continuas.
- Siendo código abierto, no hay condiciones de proveedor
- La inyección de dependencias y la inversión de control hacen la conectividad a base de datos más sencilla.
- Se gestiona y contiene la configuración y ciclo de vida de la aplicación de los objetos
- Se proporciona una capa abstracta genérica para llevar acabo la gestión de transacciones evitar problemas de bajo nivel.
- El desarrollo rápido de aplicaciones. Los componentes incluidos en un framework constituyen una capa que libera al programador de la escritura de código de bajo nivel.
- La reutilización de componentes software.
- El uso y la programación de componentes que siguen una política de diseño uniforme. Un framework orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que resulta en bibliotecas más fáciles de usar.
- Minimizar el código requerido por las aplicaciones para acceder a los datos, a menos código, mayor facilidad de mantenimiento y menor acoplamiento
- Homogenizar la capa de acceso a los datos para así abstraer detalles específicos de los stores, de las aplicaciones cliente, de esta manera se reduce el impacto mover los datos de un store a otro.
- Puede ser utilizada en cualquier plataforma y navegador
- Menor transferencia de datos cliente/servidor
- Optimización de recursos (tiempo de operaciones)
- Portabilidad y usabilidad (permite realizar una petición de datos al servidor y recibirla sin necesidad de cargar la página entera)

- Centralización de creación de objetos, todo lo haríamos a través de una sola clase por lo que podríamos definir los mismos métodos y acciones para todos los objetos de una aplicación.
- Un sistema debe ser independiente de cómo se crean, componen y representan sus productos.
- Un sistema debe ser configurado con una familia de productos entre varias.
- Una familia de objetos producto relacionados está diseñada para ser usada conjuntamente y es necesario hacer cumplir esa restricción.
- Se quiere proporcionar una biblioteca de clases de productos y sólo se quiere revelar sus interfaces, no sus implementaciones.

4.7 Resumen

En este caso de estudio se desarrolló la arquitectura de software para una aplicación Web que permite registrar todos los beneficios que son asignados a una persona, grupo familiar o grupo social. Facilitando así la generación de indicadores de gestión en todas sus dependencias, es personalizado a las necesidades de cada uno de nuestros clientes por medio de flujos de procesos (workflow). La primera etapa del proceso de ingeniera de software fue la obtención y análisis de los requerimientos, obteniendo la vista de escenarios.

El modelo para esta vista fueron los diagramas de casos de uso y de secuencia respectivamente.

La vista de desarrollo detalla cada una de las capas y de los componentes llegando hasta las clases y objetos del sistema. La vista de procesos muestra interoperabilidad del sistema con el usuario para la realización de una tarea específica Para analizar, la vista física se muestra a la aplicación desplegada en los componentes físicos

Se obtiene la documentación de la arquitectura con los requerimientos de la aplicación

Capítulo 5

Conclusiones y trabajo futuro

En este capítulo se resumen todas las ideas principales mencionadas a lo largo de la tesis para dar una conclusión acerca del trabajo realizado, también se hace una evaluación de los logros obtenidos y se menciona el trabajo que queda por hacer en la aplicación Te Atiendo y la integración con otras secretarias del municipio de Rionegro y el uso del software en el sector público.

5.1 Conclusiones

La principal contribución de nuestro trabajo consiste en el desarrollo de una arquitectura de software para aplicaciones basadas en Web siguiendo un modelo de ingeniería de software. Otra contribución es la especiación de dicha arquitectura, cada una de las vistas fueron documentadas utilizando el lenguaje UML.

Para el desarrollo de este trabajo de tesis se propuso el uso del modelo en cascada, para definir la arquitectura de software, debido a que describe cada una de las etapas para el desarrollo de productos de software. Nos enfocamos en la etapa de diseño del modelo en RUP. De acuerdo a las interacciones que nos salieron de los requerimientos de la aplicación, lo que nos permito trabajar metodologías agiles como Scrum para un desarrollo de la aplicación descentralizado y en un ambiente controlado, donde se iban integrando los diferentes módulos desarrollados y probados.

Para controlar las versiones se utilizó el GIT el cual nos permitía comparar los cambios que realizábamos en los diferentes componentes del software permitiéndonos tener control sobre el código y al final aceptando los cambios de la aplicación.

Para proponer una arquitectura inicial analizamos la capa lógica de una aplicación web, esto nos permitió identificar las capas y niveles iniciales de la arquitectura. Con base a este análisis presentamos una arquitectura web con integración de patrones y frameworks que nos ayudan a simplificar y tener control sobre el código desarrollo de la aplicación.

El proceso de diseño fue acompañado por el patrón de diseño multicapa y Layer, estos nos permitieron agregar componentes e interfaces a la arquitectura siguiendo una buena distribución y buen acoplamiento. El resultado del proceso de diseño es una arquitectura que incluye todos los componentes e interfaces necesarios para el funcionamiento de la aplicación Te Atiendo de forma robusta.

Podemos concluir que los objetivos planteados fueron alcanzados, para lograrlos fue necesario repasar varios módulos de los estudios realizados en la maestría y la lectura de varios apartados de libros que me permitieron interpretar formalmente el proyecto y

ver la relación entre java, patrones y UML, esto se ha conseguido a medida que el alcance del proyecto se aumentada, una de las principales dificultades que se presentaron es que los clientes no saben el alcance del proyecto y constantemente están solicitando cambios y algunos exigían cambiar muchas líneas de código.

Fue una experiencia enriquecedora debido a que he perfeccionado los conocimientos en metodologías usadas para el desarrollo del software integrando RUP, UML, en el paradigma de la programación orientada a objetos y el uso de metodologías agiles las cuales permitieron lo siguiente:

- Prepararnos para los cambios del proyecto
- Impuestas para el desarrollo de los módulos, incentivando el trabajo en equipo y logrando la escalabilidad y trazabilidad del software
- Procesos menos controlados, con pocos principios lo que permite un desarrollo descentralizado
- El cliente es parte del equipo de desarrollo
- Menos énfasis en la arquitectura del software

El uso de patrones fue de gran ayuda ya que permiten reusar información con un alto grado de abstracción, enriquece las heurísticas existentes, ya que brinda orientación en los procesos de construcción y una fuente de de confirmación en el escenario de desarrollo como lo es MAVEN, de este modo se mejoran los escenarios producidos y se reduce el tiempo de desarrollo.

5.2 Trabajo Futro

En cuento a los trabajos futuros que se tienen en la aplicación son muchos ya que a medida que se van sumando nuevas dependencias de la Alcaldia de Rionegro, el alcance del proyecto se va volviendo más grande, una de las apuestas que estamos apuntando es a montar la aplicación en los servidores de Amazon lo cual nos permitirá tener mayor seguridad, escalabilidad y fiabilidad de la aplicación.

Para lograr lo antes expuesto la Alcaldia para el año 2016 debe de destinar presupuesto por el rubro de tecnología, para poder seguir realizándole mejoras a la aplicación, desarrollando nuevos módulos, como el de reportes gráficos ya que al principio se tenían contemplados, pero con los cambios que han salido el presupuesto asignado se ha agotado y se ha quedado el modulo pendiente por desarrollar.

Otros de los puntos relevantes que están pendientes es incorporar el módulo de finanzas donde la Alcaldia pueda cuantificar, cuanto invierte en cada una de las dependencias, permitiéndole presentar informes completos y llevar una sana contabilidad a la hora de rendir sus cuentas a las diferentes entidades que la regulan.

6. Referencias

[1]J.Conallen. Modeling web application architectures with uml, Communications of the ACM, 42(10):63{70, 1999.

[2] Peter F. Drucker, Managment: Tasks, Responsabilities, Practices, New York : Haper & Row, 1973

[3] Barry W. Boehm " A spiral model of software development and enhancement ", Computer, May 1988, pp 61-72. (Reprinted in Barry W. Boehm, Tutorial : Software Risk Management. IEEE Computer Society Press, Los Alamitos, CA, 1989.

[4] Barry Boehm, " Anchoring the software process", IEEE Software , July 1996, pp 73-82.

[5] Christopher Alexander, Sara Ishikawa, Muray Silverstein, with Max Jacobsen, Ingrid Fiksdahi-King, Shlomo Angel, A Pattern Language: Towns, Buildings, Construction, New York: Oxford University Press, 1977.

[6] Tutorial patrones de Java, tutorialspoint simpleasy learning disponible en: http://www.tutorialspoint.com/design_pattern/index.htm

[7] Documentación de Groovy disponible en:
http://www.groovy-lang.org/single-page-documentation.html

[8] Documentación de Spring framework disponible en: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/

[9] Documentación de Spring- Data JPA disponible en:
http://docs.spring.io/spring-data/data-jpa/docs/current/reference/html/

[10] Documentación de MySql disponible en:
http://dev.mysql.com/doc/

[11] Documentación de JQuery disponible en:
https://jquery.com/

12] Documentación de Factory disponible en:
https://es.wikipedia.org/wiki/Factory_Method_(patr%C3%B3n_de_dise%C3%B1o)

[13] Javaworl, artículo 2073352 disponible en:
http://www.javaworld.com/article/2073352/core-java/simply-singleton.html

[14]Tutorial patrones de Java, tutorialspoint simpleasy learning disponible en: http://www.tutorialspoint.com/design_pattern/builder_pattern.htm

[15] Tutorial patrones de Java, tutorialspoint simpleasy learning disponible en: http://www.tutorialspoint.com/design_pattern/adapter_pattern.htm

[16] Oracle, techetwork articulo java 142890 disponible en: http://www.oracle.com/technetwork/articles/javase/index-142890.html

[17] Documentación de Ajax disponible en:
http://api.jquery.com/jquery.ajax/

[18] Oracle, techetwork articulo java 13995 disponible en:
http://www.oracle.com/technetwork/java/index-jsp-135995.html

[19] Documentación de git disponible en:
https://git-scm.com/

[20]Apache maven Project disponible en:
https://maven.apache.org/

[21]Documentación Jenkins disponible en:
https://jenkins-ci.org/

[22] Documentación Eclipse disponible en:
https://eclipse.org/

[23] Bass, L., Klein, M., Bachmann, F. “Quality Attribute Design Primitives” CMU/SEI-2000-TN-017, Carnegie Mellon, Pittsburgh, December, 2000

[24] Model-Driven Architecture (MDA) Home Page:
http://www.omg.org/mda/index.htm

[25] Frank Buschman, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal: Pattern-Oriented Software Architecture A System of Patterns; John Wiley & Sons Ltd. Chichester, England, 1996

[26] Admed E Hassan, R.C. Holt. Architecture Recovery of Web Applications. ICSE’02, May 2002.

[27] UML Profile for Enterprise Distributed Object Computing Specification. Febrary 2002.

[28] Jim Conallen. Building Web Applications with UML. Adisón Wesley Longman. December 1999.

[29] WebSA (Web Software Architecture) home page: http://www.dlsi.ua.es/~santi/papers/websatr.pdf

7. Licencia

GNU Free Documentation License

Version 1.3, 3 November 2008

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that

could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will

remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.
If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS
You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment

to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a

principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.