INFORMATIZACIÓN DEL PROCESO DE GESTIÓN DE LAS AGENDAS PARA LOS SERVICIOS QUE OFRECE EL CENTRO PARA EL CONTROL ESTATAL DE MEDICAMENTOS, EQUIPOS Y DISPOSITIVOS MÉDICOS.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Ernesto A. Redonet Herrera

Tutor: Ing. Yackson Mendoza Romero

"Año 58 de la Revolución"
La Habana, Cuba

Declaración de Autoría

Declaro que Ernesto Antonio Redonet Herrera es el único autor de la presente tesis que tiene por título: Informatización para el proceso de gestión de las agendas para los servicios que ofrece el Centro para el Control Estatal de Medicamentos, Equipos y Dispositivos Médicos. El producto desarrollado, las fuentes y cualquier resultado de la investigación son propiedad del Centro para el Control Estatal de Medicamentos, Equipos y Dispositivos Médicos (CECMED).

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Firma del Autor	Firma del Tutor
Ernesto A. Redonet Herrera	Ing. Yackson Mendoza Romero

Datos de Contacto

Nombre y Apellidos: Ernesto Antonio Redonet Herrera

Correo electrónico: ernestoarh@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Nombre y Apellidos: Ing. Yackson Mendoza Romero

Graduado de Ingeniero en Ciencias Informáticas en el año 2007. Se desempeña como profesor de Algoritmo, Introducción a la Programación 1, Programación 2: Estructuras de Datos y Práctica Profesional 1 y 2 en la Facultad Regional de la UCI “Mártires de Artemisa”. Obtuvo la categoría docente de Profesor Instructor en la Universidad de Pinar del Río. Se ha desempeñado como programador, analista, probador y actualmente Jefe de Grupo de Desarrollo en la Empresa SOFTEL.

Correo electrónico: yackson@softel.cu

AGRADECIMIENTOS

A mis padres.

Por la confianza que han depositado en mí y por la formación que me han inculcado durante toda mi crianza. Por apoyarme en las decisiones que he tomado y alentarme a seguir adelante. A ustedes que son mi razón de ser.

A mi familia.

Por el cariño y la atención que siempre me han brindado. Por ser una parte importante de vida y enseñarme lo importante que es mantener la familia unida.

A mi tutor Yackson.

Por la constancia, la ayuda, la orientación y la exigencia durante la etapa de desarrollo de este trabajo. Sin ustedes llegar hasta aquí habría sido todo un desafío.

DEDICATORIA

A mis padres, a mi familia en general.

Resumen

En la gestión de la información se ha observado una creciente modernización de las tecnologías aplicadas en nuestro país. En la dirección de Preevaluación y Recepción de trámites del Centro para el Control Estatal de Medicamentos, Equipos y Dispositivos Médicos (CECMED), durante el proceso de solicitud de los servicios la elaboración de las agendas de los especialistas se realizaba de forma manual, lo cual resultaba engorroso debido al número de solicitudes de los clientes y a las diferentes vías de recepción. En el actual trabajo se describe la solución a esta problemática a partir de la informatización del proceso de gestión de las agendas para esta dirección, para lo cual se implementó un módulo web en el lenguaje de programación Groovy, con framework Grails, gestor de base de datos PostgreSQL y se utilizó la metodología RUP como guía para el desarrollo del módulo, de igual manera se describieron las características de este módulo, se generaron los artefactos que propone esta metodología y se realizaron pruebas de caja negra para garantizar el correcto funcionamiento del módulo.

PALABRAS CLAVE: módulo, gestión de las agendas, solicitudes, servicios.

SUMMARY

In the information management it has been a growing modernization of technologies applied in our country. In the direction of Screening and Reception steps of the Center for State Control of Drugs, Medical Devices (CECMED) during the application process services the development of schedule of specialists performed manually, which it was cumbersome due to the number of customer requests and receiving different ways. In the current work the solution to this problem from the computerization of the management process of the schedule for this direction, for which a web module was implemented in the Groovy programming language, with framework Grails, database manager described PostgreSQL and RUP methodology was used to guide the development of the module, just as the features of this module is described, artifacts proposing this methodology and black box testing performed to ensure proper operation of the module were generated.

KEYWORDS: module, management schedule, demand, services.

Índice

Índice de figuras

Índice de tablas

Introducción

El Centro para el Control Estatal de Medicamentos, Equipos y Dispositivos Médicos (CECMED) es una Unidad Presupuestada, subordinada al Ministerio de Salud Pública, creada por la Resolución No. 263, de 11 de mayo de 2011 dictada por el Ministro de Salud Pública. Se creó por la fusión de las Unidades Presupuestadas denominadas Buró Regulatorio para la Protección de la Salud, el Centro para el Control Estatal de Calidad de los Medicamentos y el Centro de Control Estatal de Equipos Médicos, todas subordinadas al Ministerio de Salud Pública. Como Autoridad Reguladora de Medicamentos.

Este centro cuenta con varias direcciones y departamentos, en la actualidad gran parte del flujo de trabajo de la dirección de Recepción y Preevaluación de trámites es realizado de forma manual y algunos procesos son gestionados con herramientas informáticas que no cubren todas las necesidades del CECMED, lo que ha generado trabas en el correcto funcionamiento del mismo, por lo cual, se hace imprescindible la informatización de este departamento, estableciéndose con dicho propósito las relaciones de trabajo y de colaboración con la empresa cubana de desarrollo de software “SOFTEL”.

Los especialistas de la dirección de Recepción y Preevaluación de trámites son los encargados de crear y manipular sus agendas, en ellas se escogen los días disponibles para los clientes, quienes solicitan servicios dados por esta institución y están comprendidos por una duración en minutos. Los especialistas deben velar que los clientes no reserven turnos antes o después de la jornada laboral y que no se solapen con otros turnos ya reservados. En caso de que el cliente no pueda ser atendido en el horario reservado, el especialista debe informar este evento a un especialista superior que es el encargado de transferir el turno a otro especialista que atienda el mismo servicio.

Todo este proceso se hace muy complejo porque los especialistas están sobrecargados de tareas y toda esta actividad se realiza de forma manual y la transferencia de turnos por parte del especialista superior se dificulta. Por este motivo surge el **problema a resolver:** ¿Cómo mejorar el proceso de gestión de las agendas para los servicios del CECMED?

A partir del problema enunciado, se define como **objeto de estudio:** proceso de la dirección de Recepción y Preevaluación de trámites, cuyo **campo de acción** es el subproceso de la reservación de turnos de servicios y citas.

El **objetivo general** de esta investigación es desarrollar un módulo que se integre al Sistema de Información del CECMED, y permita perfeccionar la gestión de las agendas de los especialistas del área de Recepción y Preevaluación del CECMED.

La **idea a defender** queda planteada de la siguiente manera: Con el desarrollo del módulo Agenda se garantizará la organización y control de la información de las mismas.

Las **tareas de investigación** están orientadas a:

- ✓ Selección de las herramientas y tecnologías para la implementación del módulo.
- ✓ Levantamiento de requisitos y descripción del proceso de la gestión de agendas mediante la descripción de casos de uso de sistema.
- ✓ Análisis, diseño e implementación del módulo.
- ✓ Integración del módulo implementado.
- ✓ Diseño de los casos de prueba para validar la implementación del módulo Agenda.
- ✓ Corrección de errores detectados.

Para obtener los conocimientos necesarios que hagan posible el cumplimiento del objetivo trazado, se lleva a cabo una investigación en la que se utilizan algunos de los métodos científicos existentes, tanto teóricos como empíricos.

Teóricos:

- ✓ **Histórico lógico:** Se aplicó para constatar teóricamente cómo han evolucionado los sistemas de información, lo que permitió conocer cómo funcionan las tendencias más recientes en la gestión y control de la información y realizar un análisis real para identificar los problemas existentes dentro del proceso actual de la gestión de los medios informáticos.
- ✓ **Analítico-Sintético:** Permitió realizar un estudio sobre los elementos más importantes relacionados con el proceso de gestión de la información y posteriormente sintetizar la información necesaria para la gestión y el control de las agendas.
- ✓ **Modelación:** Se utilizó para representar por medio de diagramas el proceso de gestión y control de las agendas. Lo que permitirá una mayor comprensión de los procesos y lineamientos de desarrollo

a seguir en cada una de las fases, a partir de su utilización para el modelado del sistema, usando el Proceso Racional Unificado (RUP) y el Lenguaje Unificado de Modelación (UML).

Empíricos:

- ✓ **Observación**: Facilita conocer el panorama real de una situación mediante la percepción directa. Con la utilización de este método se determinaron los rasgos imprescindibles para el desarrollo del módulo informático.
- ✓ **Entrevista:** Se realizaron entrevistas a los especialistas del Departamento de Recepción y Preevaluación de trámites, para garantizar que los errores y deficiencias que están presentes en el proceso de gestión actual se erradiquen.

Aporte práctico

Con el módulo informático para el proceso de gestión de las agendas de la dirección de Recepción y Preevaluación de trámites del CECMED, el conjunto de especialistas obtendrán una nueva herramienta para su desempeño diario que contribuye al proceso de gestión de la información de las mismas.

El presente Trabajo de Diploma, está estructurado en 4 capítulos, a continuación se muestra una breve descripción de cada uno de ellos:

Capítulo 1: “Fundamentación teórica”

Se hace una descripción de los conceptos relacionados con los sistemas de información, las tendencias, técnicas, tecnologías, metodologías y el análisis de algunas soluciones existentes a nivel internacional y nacional que son de interés por tener estrechos lazos con el alcance de esta investigación.

Capítulo 2: “Análisis del Sistema”

Breve descripción del problema. Estudio de los principales procesos que se llevan a cabo en la dirección de Recepción y Preevaluación de trámites. Definición de requisitos funcionales y no funcionales descritos a través de diagramas de casos de uso del sistema y descripciones textuales de casos de uso del sistema.

Capítulo 3: “Diseño del Sistema”

Aborda las actividades definidas para el diseño del sistema siguiendo la metodología RUP, se elaboran los diagramas de clases del diseño teniendo presente los patrones a utilizar y se especifica la arquitectura del sistema.

Capítulo 4: “Implementación y Prueba del Sistema”

En este capítulo se definen los diagramas de componentes y despliegue. Además se presentan los resultados de las diferentes pruebas realizadas al sistema implementado, para verificar que cumple con los requisitos y así comprobar sus funcionalidades.

CAPÍTULO 1: Fundamentación teórica

Introducción

En este capítulo se muestran los conceptos básicos relacionados con la gestión de la información, agenda, módulo, y los diferentes recursos que permite aprovechar un sistema con estas características, también se lleva a cabo el análisis y estudio de la selección de herramientas, tecnologías y lenguajes de programación con el de sustentar su utilización en el desarrollo de la solución que se propone.

1.1 Conceptos fundamentales

Agenda

La agenda (del latín agenda, cosas que se han de hacer) es un libro o cuaderno con su parte principal originalmente en blanco, pero que con su uso se irá rellenando con las anotaciones que nos permitan recordar y planificar los diversos eventos previstos para tener tiempo de ocio o ejercicio profesional y los asuntos pendientes de hacer (Gusgus).

Módulo

En programación, un módulo es un software que agrupa un conjunto de subprogramas y estructuras de datos. Los módulos son unidades que pueden ser compiladas por separado y los hace reusables y permite que múltiples programadores trabajen en diferentes módulos en forma simultánea, produciendo ahorro en los tiempos de desarrollo (Jkbw) .

1.2 Componentes Agenda usados en sistemas existentes en el mundo.

Agenda de Zimbra

Zimbra Collaboration Suite (ZCS) es una suite de colaboración que combina herramientas de correo electrónico, calendarios, libreta de direcciones, bloc de notas, etc. Todas estas herramientas se encuentran al utilizar el cliente de web en una misma dirección, una única ventana de autenticación y desde cualquier lugar en el que se encuentre, lo que demuestra la potencia de esta solución sin ni siquiera haber entrado en detalle sobre las distintas herramientas.

La Agenda es una de las características más destacadas de Zimbra, con capacidad para detectar, visualizar y corregir solapamientos. Permite la gestión de recursos (salas de reuniones, proyectores, etc.), programación de reuniones en grupo con delegación del acceso, uso compartido y publicación del

calendario con otros usuarios y suscripción a calendarios remotos en formato iCal (Instituto de Investigaciones Biológicas Clemente Estable, 2013).

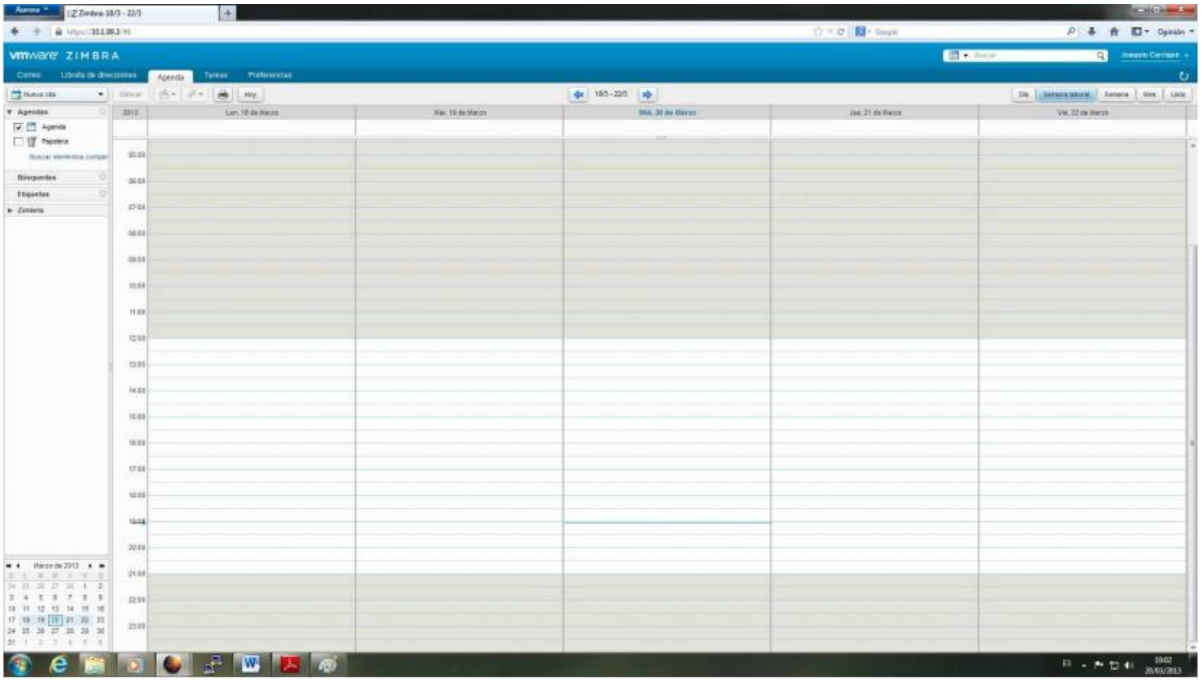

Fig. No 1 Módulo Agenda Zimbra

Agenda de Microsoft Outlook

Outlook es un software que permite enviar, recibir y administrar el correo electrónico, y administra también el calendario y los contacto. También se puede compartir el calendario con familiares y profesionales a través de Internet.

El Calendario de Microsoft Office Outlook es el componente de calendario y agenda de Office Outlook y está completamente integrado con el correo electrónico, los contactos y otras funciones. Con el calendario se puede crear citas y eventos, organizar reuniones, consultar los calendarios de grupos entre otras funcionalidades (Microsoft, 2007).

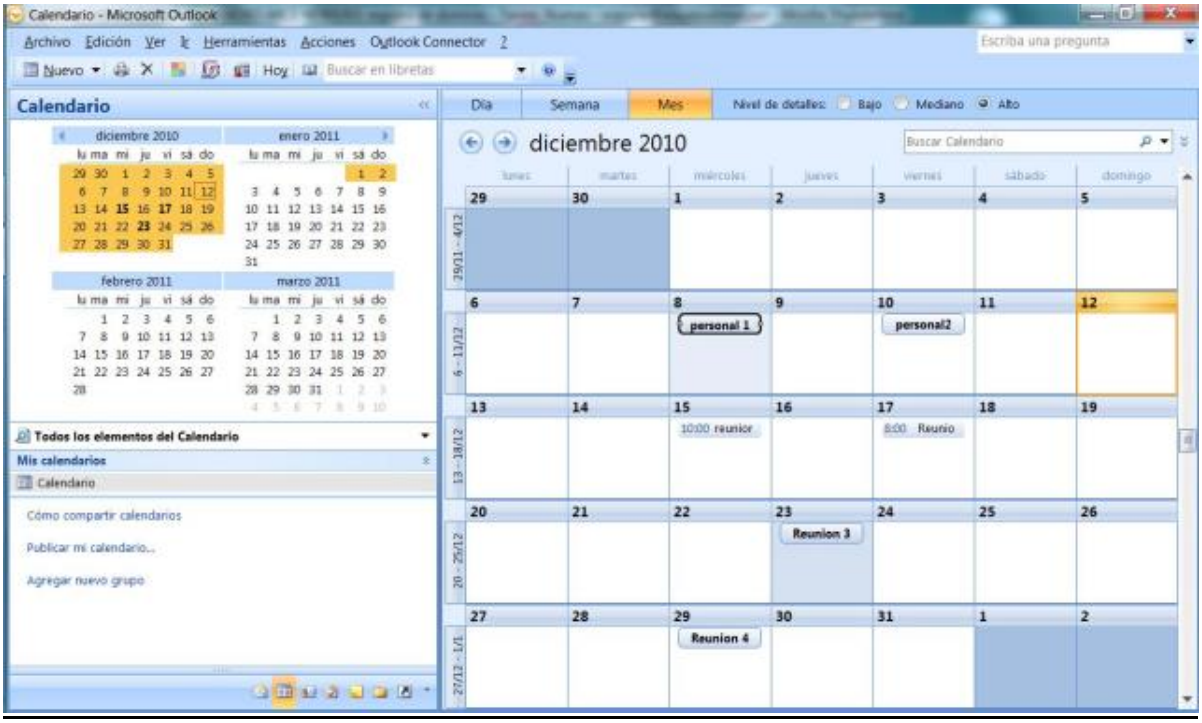

Fig. No 2 Módulo Agenda de Microsoft Outlook

1.3 Estudio de otras aplicaciones existentes en Cuba

Componente Agenda del Sistema Galen Clínicas

Galen Clínicas es una aplicación informática desarrollada por la Empresa Cubana de Soluciones Informáticas (Softel) que tiene una versión que trabaja en entorno web. Esta aplicación funciona en instituciones de salud en Cuba y se usa en los Centros de Diagnóstico Integral (CDI) de Venezuela.

La funcionalidad de este componente es crear espacios de reservación para turnos por consultas y médicos (Cibercuba).

1.4 Metodologías de desarrollo de software

La creciente informatización de los procesos productivos y sociales ha traído consigo que las organizaciones y empresas requieran cada vez más de software confiable y de alta calidad. Es por ello que en los últimos años se han venido publicando estándares, notaciones y metodologías que establecen buenas prácticas para los procesos de desarrollo de software (Z, 2000).

Se debe mencionar que no existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable y las diversas metodologías existentes son aplicables a diferentes proyectos según sus características. Para dar una idea de qué metodología se pueda usar y cuál será la más adaptable al sistema, se hará una breve descripción de las siguientes: RUP, Open UP y XP.

RUP (Rational Unified Process)

El Proceso Unificado Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Divide el proceso en nueve flujos de trabajo y cuatro fases, es iterativo incremental, dirigido por casos de uso y centrado en la arquitectura, enfocado en los riesgos y es utilizado principalmente en grandes proyectos ya que al ser una metodología pesada genera mucha documentación y alarga el tiempo de producción (Kruchten).

Open UP (Open Unified Process)

Open UP posee una gran adaptabilidad a las necesidades de un proyecto en específico y está basada en la potente metodología pesada RUP. Open Up es un proceso unificado ágil y liviano, que aplica un enfoque iterativo e incremental dentro de un ciclo de vida estructurado y contiene un conjunto mínimo de prácticas que ayuda al equipo a ser más efectivo desarrollando software. Abraza una filosofía pragmática y ágil de desarrollo, que se enfoca en la naturaleza colaborativa del desarrollo de software y es libre de pago (Gimson, 2012).

XP (Extreme Programming)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizadas para proyectos de corto plazo, cortó equipo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. XP pone la comprobación como el fundamento del desarrollo, con cada programador escribiendo pruebas cuando escriben su código de producción. Las pruebas se integran en el proceso de integración continua y construcción lo que rinde una plataforma altamente estable para el desarrollo futuro. Se basa en la retroalimentación entre el cliente y el equipo de

desarrollo, buena comunicación entre los participantes y simplicidad en las soluciones implementadas. Se preocupa por el aprendizaje de los desarrolladores, y propicia un buen clima de trabajo (Pérez García, 2006).

Fundamentos de la metodología seleccionada

Debido a las características que tiene el sistema informático de se selecciona la metodología RUP por las siguientes razones:

- ✓ Posee una gran adaptabilidad a las necesidades de un proyecto en específico.
- ✓ Distribuye la carga de trabajo a lo largo del tiempo del proyecto ya que todas las disciplinas colaboran en una iteración.
- ✓ Reduce la complejidad del mantenimiento (extensibilidad y facilidad de cambios).
- ✓ Facilita la construcción de prototipos.

1.5 Lenguaje de modelado

El lenguaje de modelado es un conjunto de signos o modelos que permiten transformar el lenguaje común a un lenguaje más técnico con el objetivo de obtener una mejor comprensión, visualización y descripción del sistema a desarrollar, se utiliza para la creación de diagramas de casos de usos y de clases del diseño, modelos de datos, entre otros.

UML (Unified Modeling Language) 2.0

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código, así como generadores de informes. La especificación de UML no define un proceso estándar, pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos (Rumbaugh, et al., 2007) .

1.6 Herramienta CASE (Computer-Aided Software Engineering)

Las herramientas CASE son una tecnología para automatizar el desarrollo y mantenimiento del software, combinando herramientas de software y metodologías. Estas herramientas deben constituir un conjunto integrado que automatice todas las partes del ciclo de vida y por tanto ahorren trabajo. El uso de Herramientas CASE para la modelación UML proporciona mayor rapidez y entendimiento en el desarrollo de software. Este tipo de aplicaciones se han convertido en ayuda y apoyo imprescindible para los desarrolladores (Hernández González, 2003)**.**

Algunas de estas herramientas son:

Rational Rose

Rational Rose 2000 provee el modelado basado en UML (Unified Modeling Language) para el diseño de aplicaciones basadas en componentes. El UML, del cual fue pionero Rational y fue adoptado oficialmente como una norma por el OMG (Object Management Group), es el lenguaje estándar de la industria para especificar, visualizar, construir, y documentar los artefactos de un sistema de software. Rational Rose 2000 puede hacer ingeniería en reversa de componentes COM, ActiveX y JavaBeans, para derivar las interfaces y determinar las interrelaciones de todos los componentes dentro de un modelo. Rose 2000 en su versión Enterprise provee la capacidad para mezclar y casar múltiples idiomas, como C++, Visual Básico, y Java, dentro del mismo modelo (Quatrani, 2002).

A pesar de ser una buena herramienta presenta inconvenientes, necesita de mucha memoria para poder ser manejado de forma rápida y eficiente. Si no se dispone de un buen rendimiento, presenta dificultades a la hora de editar los diagramas y trabajar con ellos y tiene licencia privativa.

ArgoUML

Es una aplicación de diagramado de UML escrita en Java y publicada bajo la Licencia BSD (Bekerley Software Distribution) open source. Dado que es una aplicación Java, está disponible en cualquier plataforma soportada por Java.

Sin embargo, desde la versión 0.20, ArgoUML está incompleto. No es conforme completamente a los estándares UML y carece de soporte completo para algunos tipos de diagramas de secuencia y los de colaboración (EcuRed).

Visual Paradigm

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. Ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Además, cuenta con una potente funcionalidad para la creación de interfaces de usuarios de las aplicaciones, dándole una peculiaridad sobre el resto de las herramientas de este tipo (Tsang, et al., 2005).

Fundamentación de la Herramienta CASE seleccionada.

La utilización de la herramienta Visual Paradigm es una opción factible por su gran cantidad de funcionalidades y su fácil uso. Visual Paradigm presenta ventajas importantes sobre ArgoUML:

- ✓ Interfaz de usuario muy bien diseñada, fácil de aprender a usar e intuitiva.
- ✓ Instalación fácil y sin costo alguno.
- ✓ Comodidad para la inicialización en la herramienta.

1.7 Sistemas gestores de base de datos (SGBD)

Un Sistema de Gestión de Base de Datos (SGBD) es un software de propósito general que facilita el proceso de definir, construir y manipular la BD para diversas aplicaciones. Pueden ser de propósito general o específico. Actualmente existe una amplia gama de SGBD con características propias, no obstante, todos deben tener en cuenta los siguientes aspectos de manera general: abstracción de la información, la independencia de los datos, redundancia mínima, consistencia en los datos, seguridad, integridad, respaldo y recuperación, tiempo de respuesta y control de concurrencia.

Oracle

Oracle es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos. Es un producto vendido a nivel mundial, aunque su elevado precio hace que sólo se vea en empresas muy grandes y multinacionales. Para desarrollar en Oracle se utiliza PL/SQL un lenguaje de 5ta generación, bastante

potente para tratar y gestionar la base de datos, por norma general se suele utilizar SQL al crear un formulario.

Oracle es considerado como uno de los sistemas de bases de datos más completos, destacando: soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma. Oracle puede ejecutarse en todas las plataformas, soporta tanto funciones como procedimientos almacenados con una integridad referencial declarativa bastante potente.

Esta herramienta ha sido diseñada para controlar y gestionar grandes volúmenes de datos en un único repositorio. Es posible lógicamente atacar a la base de datos a través del SQL plus incorporado en el paquete de programas Oracle para poder realizar consultas, utilizando el lenguaje SQL**.** Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

PostgreSQL 9.2.4.1

Es un Sistema Gestor de Base de Datos que comenzó como un proyecto en la Universidad Berkeley de California. Este proyecto sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto. Está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre. Puede funcionar en múltiples plataformas y a partir de la versión 8.0 también en Windows de forma nativa. Para las versiones anteriores existen versiones binarias para este sistema operativo, pero no tienen respaldo oficial. Sus ventajas más notables son:

- ✓ Aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas.
- ✓ Soporta operadores definidos por el usuario, funciones, métodos de acceso y tipo de datos.
- ✓ Soporta el núcleo SQL99 e incluye características avanzadas tales como las uniones (joines) de SQL92.
- ✓ Soporta la integridad referencial, que se utiliza para asegurar la validez de los datos de base de datos.
- ✓ La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS15 PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.
- ✓ Usa una arquitectura proceso-por-usuario, cliente/servidor PostgreSQL.

MySQL

MySQL o monitor MySQL es un gestor de base de datos sencillo de usar y muy rápido, constituye uno de los motores de base de datos más usados en Internet, es un programa interactivo que permite conectarse a un servidor MySQL, ejecutar algunas consultas, y ver los resultados, puede ser usado también en modo Batch: es decir, se pueden colocar toda una serie de consultas en un archivo, y posteriormente decirle a MySQL que las ejecute.

Puede considerarse un software gratis cuando su uso es para aplicaciones no comerciales ya que muchas de las herramientas que existen para gestionar sus bases de datos son pagadas, como por ejemplo SQLYog, SQL Studio for MySQL o SQL-Front. El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo, así como para integrarse en software para ser distribuido; soporta varios lenguajes de programación como C, C++, java, Perl, PHP, Python.

Fundamentación del SGBD seleccionado

A partir del análisis previo de los tres Sistemas Gestores de Bases de Datos (MYSQL, Oracle y PostgreSQL), se llegó a la conclusión de que el más idóneo para utilizar en este proyecto sería el PostgreSQL. Debido a que soporta una mayor cantidad de peticiones simultáneas de manera correcta, implementa el uso de subconsultas y transacciones, lo que proporciona una mayor eficacia a su funcionamiento, así como la capacidad de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel. Una de las desventajas que presenta el Oracle es que es un software propietario y su licencia es excesivamente cara. Mientras que el PostgreSQL está disponible sin costo alguno.

1.8 Lenguaje de programación

Un lenguaje de programación es un idioma artificial, compuesto por un conjunto de símbolos, reglas semánticas y sintácticas que permiten la comunicación entre una persona y el ordenador (Rodriguez, 2013). Pueden clasificarse en dos grupos principales:

- ✓ Lenguajes de bajo nivel: Se acercan al funcionamiento del computador. El lenguaje representativo es el código máquina. Las instrucciones en este lenguaje están formadas por cadenas binarias (0 y 1). Puede citarse el lenguaje ensamblador el cual trabaja directamente con los registros de la computadora.

- ✓ Lenguajes de alto nivel: Son fáciles de aprender porque están formados por elementos del lenguajes y al usarlos puede dar la sensación de que las computadoras los comprenden.

PHP

PHP es un robusto lenguaje de programación del lado del servidor. Fue diseñado por Rasmus Lerdorf en 1994 como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje. Dado que es un lenguaje abierto dando la posibilidad de modificar el código fuente y añadir nuevas funcionalidades ha tenido una rápida evolución y desarrollo. Actualmente se encuentra en su versión PHP 7.

Una de sus grandes potencialidades es su soporte para una gran cantidad de bases de datos. De las cuales se pueden mencionar InterBase, MySQL, Oracle y PostgreSQL, entre otras. PHP también ofrece la integración con varias bibliotecas externas, que dan al desarrollador la posibilidad de realizar cualquier tarea, desde generar documentos en pdf (Portable Document Format) hasta analizar código XML (Vaswani, 2010).

Java

Java es una tecnología orientada al desarrollo de software con la cual se puede realizar cualquier tipo de programa, nace a inicio de los años noventa, creado por la empresa Sun Microsystems como un lenguaje ideado en sus comienzos para programar electrodomésticos, en sus primeras versiones, se llamó OAK, fue diseñado como un lenguaje orientado a objetos desde el principio, para crear software altamente fiable.

Con el uso de Java se facilita la creación de programas modulares y códigos reutilizables. Este lenguaje se ha convertido en una de las elecciones para ofrecer soluciones en todo el mundo. Es compilado, en la medida en que su código fuente se transforma en una especie de código máquina (bytecodes), semejantes a las instrucciones de ensamblador; Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time) (Chamorro Mínguez, 2011).

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.

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 (IBM).

Fundamentos del lenguaje web utilizado

Como lenguaje utilizado para el desarrollo de la aplicación Web se eligió Groovy ya que es u lenguaje multiplataforma. Es orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. Su capacidad de conexión con la mayoría de bases de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL. Es libre, por lo que se presenta como una alternativa de fácil acceso para todos. Permite aplicar técnicas de programación orientada a objetos y no obliga usar variables tipadas en su codificación.

Otras tecnologías y lenguajes usados

HTML

De las siglas de (*Hyper Text Markup Language)* Lenguaje de Marcas de Hipertexto, es el lenguaje de marcado más utilizado para la construcción de páginas web. Es un lenguaje de composición de documentos y especificación de ligas de hipertexto que define la síntesis y coloca instrucciones especiales que no muestra el navegador, aunque sí le indica como desplegar el contenido del documento, incluyendo texto, imágenes y otros medios soportados.

Su versión 5 brinda más facilidades que sus antecesores, lo que permite el desarrollo de la aplicación con una mayor vistosidad, calidad e integración, este es soportado por las versiones actuales de los navegadores Mozilla Firefox, Chrome, Chromiun, Safari e Internet Explorer.

CCS (Cascading Style Sheets)

Este lenguaje se usa para organizar y controlar la presentación y aspecto de una página Web. Es principalmente utilizado por los diseñadores y programadores de aplicaciones para elegir la multitud de opciones de presentación que este brinda, como colores, tipos y tamaños de letra, etc. Se seleccionó la versión 3 de este lenguaje porque trae muchas características nuevas que enriquecen la apariencia de las

páginas web, como son los bordes redondeados, textos sombreados, sombreados de cajas de texto, múltiples fondos, rotación de textos, animaciones, transiciones, entre otros.

Java Script

Es un lenguaje de tipo script compacto, basado en objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de internet. Como ventaja clave de este lenguaje se puede mencionar que es interpretado por todos los navegadores modernos, debido a que este lenguaje está provisto de una implementación del (*Document Object Model*) Modelo de Objetos del Documento *DOM* estándar diseñado por W3C que incorporan Konqueror, Mozilla Firefox desde su primera versión, Internet Explorer desde su versión 6.0, Netscape Navigator, Opera desde la versión 7 entre otros. El navegador tiene la responsabilidad de interpretar las sentencias.

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.

Bootstrap

Es un framework CSS desarrollado inicialmente (en el año 2011) por Twitter que permite dar forma a un sitio web mediante librerías CSS que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados en cualquier sitio web. Bootstrap es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño. Además, Bootstrap ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías.

IDE (Integrated Development Environment)

Un Entorno de Desarrollo Integrado es un conjunto de herramientas de desarrollo de software para programadores. Generalmente están compuestos por un editor de código, un compilador, un depurador y un constructor de interfaz gráfica de usuario. Son creados para el desarrollo de aplicaciones en un solo lenguaje de programación, sin embargo, hay algunos en los que se puede desarrollar en más de un lenguaje. Cuando el lenguaje es Orientado a Objetos, incluyen navegador de clases, inspector de objetos y diagrama de jerarquía de clases (Carrancá 2014).

Netbeans 8.0

Es un entorno Integrado de Desarrollo gratuito, de código abierto para desarrolladores de software. Contiene herramientas para crear aplicaciones profesionales para el escritorio, la empresa, la web y equipos móviles con el lenguaje Java, C/C++, y Ruby. Es fácil de instalar y de usar y se ejecuta en varias plataformas incluyendo Windows, Linux y Mac OS X y Solaris. La versión de Netbeans IDE 8.0, provee mejoras en el rendimiento, menor consumo de memoria y mejor respuesta cuando se trabajan con proyectos grandes. Algunas de las tecnologías y lenguajes soportadas por esta versión son: Ajax, C/C++, Databases, Debugger, Desktop, Editor, Groovy, GUI Builder, Java EE, Java FX, Java ME, Java SE, JavaScript, Mobile, PHP, Profiler, Python, Refactor, REST, Rich Client Platform, Ruby, SOA, SOAP, UML, Web, WSDL, XML, Hibernate, Spring y Struts.

Visual Studio.net

Es un entorno integrado de programación para sistemas Windows, que soporta los lenguajes de programación Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic.NET. Los programas desarrollados en .NET no se compilan en lenguaje máquina cómo C++, sino que se compilan en un lenguaje intermedio llamado Microsoft Intermediate Language (MSIL) como se conoce por sus siglas en ingles.

Este lenguaje Incluye tipos genéricos, similares en muchos aspectos a las plantillas de C++. Con esto se consigue encontrar muchos más errores en la compilación en vez de en tiempo de ejecución, incitando a usar comprobaciones estrictas en áreas donde antes no era posible. Incluye un diseñador de implantación, que permite que el diseño de la aplicación sea validado antes de su implantación. También se incluye un entorno para publicación web y Tests de carga para comprobar el rendimiento de los programas bajo varias condiciones de carga.

IntelliJ IDEA 14.0.1

Es un ambiente de desarrollo integrado (IDE) para el desarrollo de programas informáticos. Es desarrollado por JetBrains (Anteriormente conocido como IntelliJ), y está disponible en dos ediciones: community edition y edición comercial. IntelliJ IDEA no está basada en Eclipse como MyEclipse u Oracle Enterprise Pack para Eclipse.

Fundamentos del IDE seleccionado

Para el desarrollo del módulo se eligió el entorno de desarrollo IntelliJ IDEA, el cual posee ventajas sobre NetBeans, entre estas se encuentran:

- ✓ Excelente integración con el lenguaje de programación seleccionado, haciendo vínculos a sus numerosas clases, interfaces, vistas, etc.
- ✓ El consumo de memoria se puede regular a tres niveles mediante la acción perfil de inspección.
- ✓ Numerosas acciones para la implementación o manteamiento del código.
- ✓ Posee opciones de vistas para la visualidad del proyecto que se esté desarrollando.
- ✓ Sincronización a diferentes bases de datos sin importar el tipo de driver.
- ✓ Auto-guardado cuando el software pierde el foco.

Máquina virtual JVM

La máquina virtual de Java se denomina al procesador o entorno virtual que se utiliza para interpretar los bytecodes de los binarios de Java, ya que este se desarrolló para correr en cualquier plataforma sin recompilar los binarios. Esta máquina virtual ha tenido la característica de ser un entorno de ejecución pesado en términos de recursos del procesador y memoria, que, por medio de una administración rigurosa del sistema operativo, podrían llegar a ser insuficientes y las aplicaciones ejecutarse de manera muy lenta. Esto no es cierto en la actualidad, ya que existen alternativas a la JVM provista por Sun Microsystems que permiten una velocidad comparable a una aplicación compilada en C++ nativa en la arquitectura.

Servidor de Aplicaciones Web

Tomcat v7

Tomcat es una implementación completamente funcional de los estándares de JSP (Java Server Pages) y Java Servlet, es el servidor Web más utilizado a la hora de trabajar con el lenguaje Java en aplicaciones web, puede especificarse también como el manejador de las peticiones de JSP recibidas por servidores Web populares, como el servidor Apache HTTP de la Fundación de software de Apache o el servidor Microsoft Internet Information Server (IIS).
Tomcat puede utilizarse como un contenedor solitario (principalmente para desarrollo y depuración) o como plugin para un servidor web existente (en este caso está integrado como un plugin al framework Grails). Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

JDK7

El Kit de desarrollo conocido como JDK (Java Development Kit) provee de un compilador, un mecanismo para comprimir un proyecto en un solo archivo de tipo JAR (que es compatible con ZIP) y un entorno de ejecución para nuestros binarios.

Marco de Trabajo o Framework

Un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Proporciona una estructura que fuerza al desarrollo de código más legible. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio y permite separar en capas la aplicación.

Grails 2.3.7

Grails es un framework libre para desarrollo web, de código abierto y que utiliza la JVM, está construido sobre cinco pilares fundamentales:

- ✓ Groovy (lenguaje de programación) para la creación de propiedades y métodos dinámicos en los objetos de la aplicación.
- ✓ Spring (framework para el desarrollo de aplicaciones) para los flujos de trabajo e inyecciones de dependencias.
- ✓ Hibernate (framework de persistencia) para el Mapeo objeto-relacional.
- ✓ SiteMesh (framework de aplicación) que gestiona la creación de las vistas.
- ✓ Ant como framework para gestionar la compilación de un proyecto.

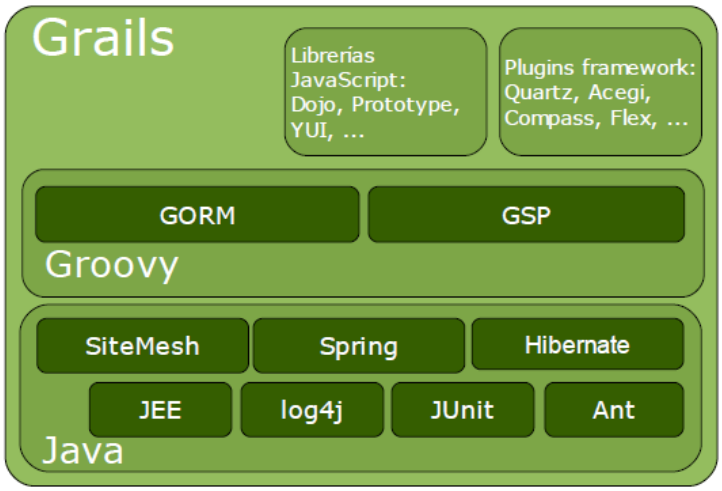

Fig. No 3 Diagrama arquitectura Grails

La idea de Grails, es crear un marco de desarrollo que favorezca la productividad al crear aplicaciones web, integrando factores de configuración comunes a la mayoría de los escenarios siguiendo paradigmas tales como; Convención sobre Configuración. cuyo objetivo fundamental es disminuir el número de decisiones que el desarrollador debe tomar, ganando en sencillez pero no pierde por ello la flexibilidad y No te Repitas, ya que su finalidad es promover la disminución de la duplicación.

Algunas de las características fundamentales de Grails son:

- **Productividad:** Posee características fundamentales que intentan incrementar su productividad en comparación con los marcos de trabajo Java tradicionales, es un marco de trabajo preparado para funcionar desde el primer momento, y con funcionalidades disponibles a través de métodos dinámicos.
- **Persistencia:** GORM (Grails Object Relational Mapping) es el medio a través del cual los datos del modelo de dominio se hacen persistentes utilizando métodos dinámicos.

Conclusiones parciales

✓ Del estudio de las herramientas que se expusieron anteriormente, se concluye que las siguientes cuentan con un gran potencial que tributa a que el producto final sea estable. Estas herramientas apoyan el desarrollo general de la aplicación, brindando facilidades a los desarrolladores en cuanto a rapidez, facilidad de uso y variedad de funciones, sin dejar de mencionar que son herramientas libres.

- ✓ Se tiene también como propuesta a utilizar la metodología de desarrollo RUP con notación UML debido a sus características y organización del trabajo. Herramienta CASE de Modelado UML: Visual Paradigm for UML 8.0.

- ✓ Partiendo de la experiencia adquirida de otros módulos Agenda integrados en diversos sistemas, se determinó que era necesario desarrollar un módulo Agenda para el sistema de Información del CECMED, puesto que las funcionalidades de los ya existentes no cumplían con los requisitos funcionales.

CAPÍTULO 2: Descripción del sistema

Introducción

Después de seleccionadas las herramientas y metodología a utilizar en el desarrollo del sistema informático; se realiza el levantamiento de los requisitos, con el fin de dar una solución factible al problema que se investiga. Se definen los requisitos funcionales y no funcionales, el modelo de dominio, el diagrama de casos de uso de sistema, el patrón de casos de uso de sistema a utilizar, actores, entidades que intervienen y la relación que existe entre los artefactos.

2.1 Descripción del Sistema Propuesto

Tras haber culminado el estudio basado en la problemática existente se proponer desarrollar el módulo de gestión de las agendas, el cual llegará a informatizar los procesos actuales que se llevan a cabo en la gestión de la información de la dirección de Recepción y Preevaluación de trámites. Al módulo podrán acceder solos los usuarios que se encuentren registrados en el sistema y su actividad en este estará definida por el tipo de rol que tengan dichos usuarios.

2.1.1 Actor del negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos con los que el negocio interactúa y desempeña un rol determinado dentro del negocio para beneficiarse de sus resultados. Representa un tipo particular de usuario del negocio más que un usuario físico, ya que varios usuarios físicos pueden realizar el mismo papel en relación al negocio, por otro lado un mismo usuario puede actuar como diferentes actores.

Tabla 1 Actor del negocio

Actor	Justificación
Cliente	Es la persona que solicita servicios atendidos por especialistas del Departamento de Preevaluación y Recepción de trámites a través de reservas de turnos en la agenda de los especialistas.

2.1.2 Trabajador del negocio

Un trabajador del negocio representa a personas o sistemas dentro del negocio que son los que realizan las actividades que están comprendidas dentro de un caso de uso. Los trabajadores están dentro de la frontera del negocio, son los candidatos a convertirse en un futuro en usuarios del sistema que se quiere construir.

Tabla 2 Trabajadores del negocio

Trabajador	Justificación
Especialista	Es la persona encargada de atender las solicitudes de servicios por los clientes a partir de las reservas de turnos en la agenda del especialista.
Especialista superior	Es la persona encargada de la transferencia de turnos de una agenda de un especialista hacia otro.

2.2 Diagrama de caso de uso del negocio

El diagrama de casos de uso representa la interacción entre actores del negocio y los casos de uso del negocio, incluye las relaciones entre actores y entre casos de uso.

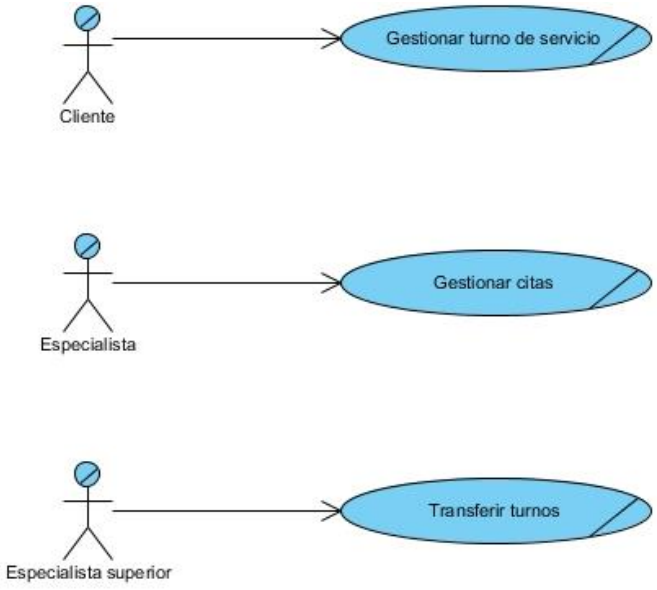

Fig. No 4 Diagrama de caso de uso del negocio

Realización de los casos de uso del negocio

Los casos de uso del negocio consisten en secuencias de actividades que, en conjunto, producen algo para el actor del negocio. La realización de los casos de uso ayuda a entender en detalle cómo se suceden las actividades a partir de la descripción de los flujos de procesos. Brinda una noción de cómo funcionan los procesos dentro del negocio.

2.3 Descripción del CUN Reservar turno

Tabla 3 Caso de uso del negocio reservar turno

Caso de Uso:	Reservar turno
Actores:	Cliente
Trabajadores:	Especialista
Resumen:	El cliente solicita reservar un turno en la agenda de un especialista del CECMED que atiende un servicio determinado. El turno está comprendido por fecha/horario. La vía de recepción del cliente puede ser mediante correo electrónico, vía telefónica o presencial. Si la combinación fecha/horario se encuentra registrada en la agenda, se le debe informar al Cliente que ya existe un turno en la agenda.
Posibles mejoras:	Se recomienda informatizar el proceso de reservas de turno en la agenda del especialista para agilizar dicha actividad.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1- El Cliente solicita reservar un turno para un servicio específico en la agenda del Especialista.	2- El Especialista revisa su agenda y verifica que la fecha y horario para el turno solicitado se encuentren servibles, en caso de que la fecha y horario estén disponibles el Especialista registra el turno en su agenda.

	3- Se le notifica al cliente que el turno pedido quedó registrado en la agenda.
Flujos Alternos	
Flujo alterno 1: Si la solicitud para reservar el turno fue enviada mediante correo electrónico	
Acción del Actor	**Respuesta del Negocio**
	2- El Especialista revisa el correo electrónico y verifica que el cliente haya escrito correctamente la fecha y horario.
	3- Continúa en el Paso 2 del Flujo Normal de Eventos.
Flujo alterno 2: Si la solicitud para reservar el turno se hace mediante vía telefónica.	
Acción del Actor	**Respuesta del Negocio**
	2- El Especialista atiende la llamada telefónica y escucha atentamente la solicitud por parte del Cliente.
	3. Continúa en el paso 3 del Flujo Normal de eventos.
Flujo alterno 3: Si la solicitud para reservar el turno es presencial.	
Acción del Actor	**Respuesta del Negocio**
	2- El Especialista
	3. Continúa en el paso 3 del Flujo Normal de eventos.

2.4 Modelo dominio

Un Modelo Dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, en la tarea construcción del modelo de dominio, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física. En el siguiente modelo se evidencia cómo funciona el entorno en el cual está enmarcado el modulo y representa los conceptos fundamentales del mundo.

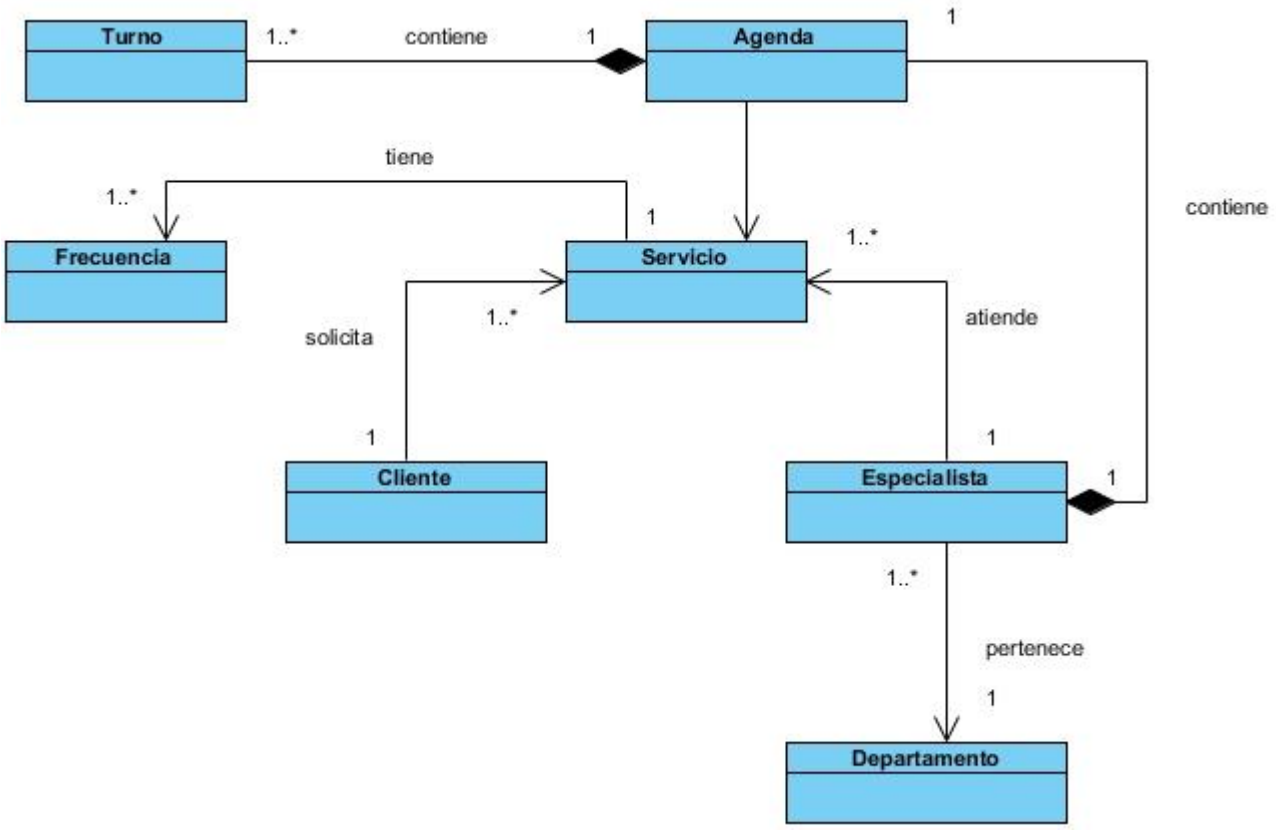

Fig. No 5 Diagrama de dominio

2.5 Descripción de las entidades

En la siguiente tabla se muestra la descripción de cada clase usada en el diagrama de clases.

Tabla 4 Descripción de las entidades

Entidad	Descripción
Agenda	Entidad que representa la agenda un especialista.
Turno	Entidad que representa el turno el cual se define por la combinación de (día, horario); donde el horario lo define hora de inicio y fin.
Frecuencia	Entidad que representa la frecuencia que tiene un servicio, se define por (cantidad de días, horario)

Cliente	Entidad que representa el usuario que solicita un servicio.
Especialista	Entidad que representa el usuario que atiende al menos un servicio.
Servicio	Entidad que representa el tipo de servicio.
Departamento	Entidad que representa el departamento donde trabajan los especialistas.

Diagrama de actividades

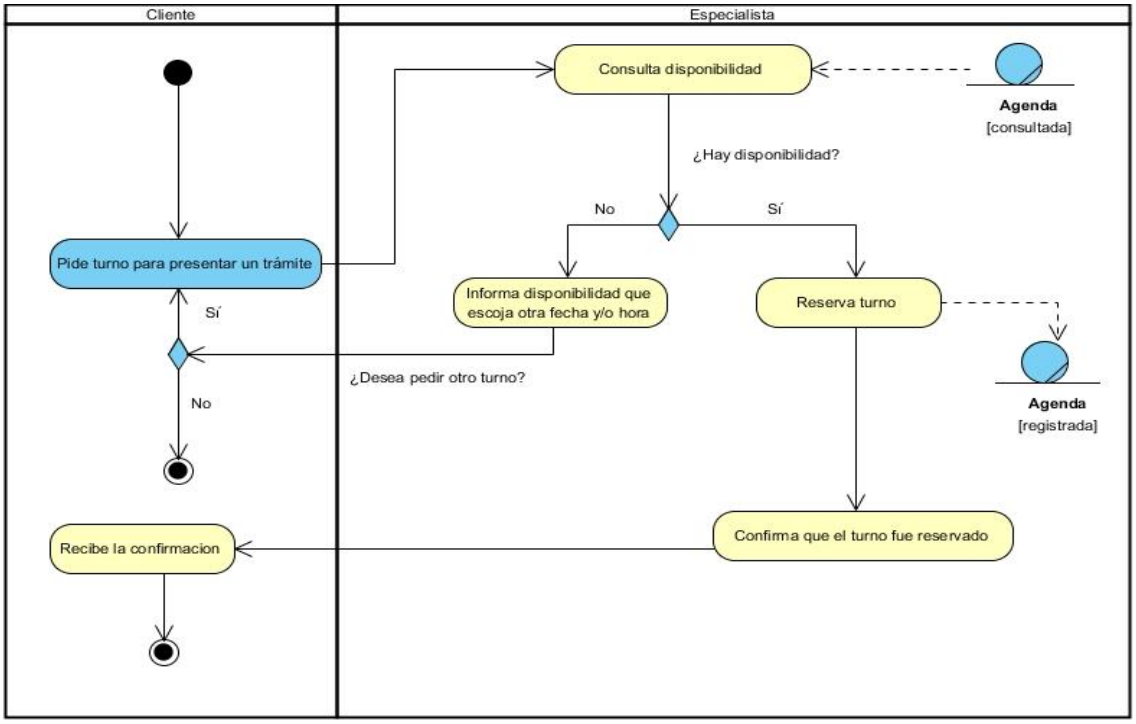

Fig. No 6 Diagrama de actividades Reservar turno

Diagrama de objetos del negocio

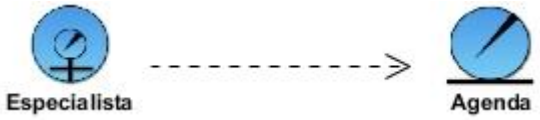

Fig. No 7 Diagrama de objetos del negocio

2.6 Especificación de requisitos

Un requisito es:

- ✓ Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- ✓ Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
- ✓ Una representación documentada de una condición o capacidad como en las dos anteriores.

Los requisitos se pueden clasificar en: funcionales y no funcionales. Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Los requisitos no funcionales son propiedades o cualidades que el producto debe tener, las características que hacen al producto atractivo, usable, rápido o confiable.

Estrategia de captura de requisitos utilizada

La captura de requisitos es la actividad mediante la cual el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema. Por la complejidad que esto implica, se empleó una técnica que permitió hacer este proceso de una forma más eficiente y precisa.

2.7 Técnicas de captura de requisitos

Entrevistas

Resultan una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Básicamente, la estructura de la entrevista abarca tres pasos: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados (protocolo de la entrevista).

Tormenta de ideas

Es una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la acumulación de ideas e información sin evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy numeroso (máximo 10 personas), una de ellas debe asumir

el rol de moderador de la sesión, pero sin carácter de controlador. Como técnica de captura de requisitos es sencilla de usar y de aplicar, puesto que no requiere tanto trabajo en grupo como éste. Además suele ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del mismo, por lo que suele aplicarse en los primeros encuentros.

Para realizar y guiar el levantamiento de requisitos se utilizó la técnica de la entrevista al cliente o usuario final, lo que ayudó a definir lo que éste desea. Otra técnica que se utilizó fue la de los casos de uso, la que consiste en llegar al requerimiento basándose en las funciones del sistema.

2.8 Lista de requisitos

Requisitos Funcionales del Sistema.

Los requisitos que a continuación se detallan responden a las diferentes funcionalidades que tendrá el sistema:

RNF1 Buscar agendas creadas por el siguiente parámetro: nombre de la agenda, el cual coincide con el nombre y apellidos del especialista.

RNF2 Listar Agendas, de cada una de ellas se muestran los siguientes datos: Nombre, Fecha de Inicio, Fecha de Fin, Estado.

RNF3 Crear agenda para cada uno de los especialistas que pertenezcan al departamento de Recepción y Preevaluación de Trámites, de cada una de ellas se solicitan los siguientes datos:

- Especialista, se corresponde con el Nombre y Apellidos del especialista el cual tiene que ser usuario del sistema, tener el rol de especialista o especialista superior asignado y pertenecer al departamento de Recepción y Pre evaluación. Dato obligatorio.
- Período Agendable, define la fecha de inicio y la fecha de fin de cada una de las agendas. La fecha de inicio no puede ser anterior a la fecha actual, y la fecha de fin no puede ser anterior a la fecha de inicio, Datos obligatorios.
- Disponibilidad, define Horario, Frecuencia, Servicio(s), donde:

  Horario: hora de inicio de atención y hora de fin de atención, donde la hora de inicio no puede ser superior a la hora de fin. Dato obligatorio.

  Frecuencia: Diaria, Semanal, Mensual, Anual. Dato obligatorio.

  Servicio(s): Lista de servicios que se seleccionan a partir de los servicios asociados al especialista al cual se le creará la agenda.

RNF4 Editar agenda, permite cambiar la configuración de una agenda. Se podrán editar todos los datos excepto el nombre de la agenda. La misma no podrá ser editada si la operación afecta los turnos de servicios reservados.

RNF5 Eliminar Agenda, se permitirá eliminar una agenda solo si la misma no tiene turnos de servicios reservados.

RNF6 Activar agenda. Solo se aplica a las agendas que se encuentren desactivadas. Una vez activadas se muestran en el sistema para su uso. De la misma se cambia el dato estado a: Activado.

RNF7 Desactivar una agenda. La misma deja de estar disponible, no se pueden agendar citas en la misma. De la misma se cambia el dato estado a: Desactivado.

RNF8 Mostrar Agenda. Se mostrarán por días los turnos o citas reservados.

RNF9 Agendar citas, permite registrar una cita atendiendo a la disponibilidad del especialista, de cada una de las citas se solicitan los siguientes datos:

- Tipo de Actividad, las actividades se clasifican en: Vacaciones, Feriados, Reuniones, Eventos, Otros.
    - Frecuencia, la frecuencia a seleccionar se muestra en función del tipo de actividad:
        - En la fecha de inicio hasta la fecha de fin, Desde: dd/mm/aa Hasta: dd/mm/aa, donde la fecha de inicio no puede ser anterior a la fecha de inicio de la agenda, la fecha de inicio no puede ser superior a la fecha de fin.
        - En la Fecha: dd/mm/aa, la fecha de inicio no puede ser anterior a la fecha de inicio de la agenda.
        - En la Fecha, de hora de inicio a hora de fin: En la fecha dd/mm/aa, de hh:mm a hh: mm, la fecha de inicio no puede ser anterior a la fecha de inicio de la agenda, la fecha de inicio no puede ser superior a la fecha de fin y la hora de inicio no puede ser anterior a la hora de inicio definida en la agenda, la hora de inicio no puede ser mayor a la hora de fin.

RNF10 El sistema no puede permitir que las citas o turnos reservados se solapen.

RNF11 Editar una cita de la agenda. Se pueden editar todos los datos de la cita siempre y cuando se respete el RF9 y RF10.

RNF12 Eliminar una cita de la agenda.

RNF13 Reservar turno para un servicio en la agenda de un especialista. De los mismos se solicitan

i. Servicio, Fecha, Hora de Inicio, Hora de Fin.

RNF14 Calcular la Hora de Fin del turno en base al tiempo de duración en minutos del servicio el cual se define en la creación del servicio: hora de fin = hora de inicio del turno + minutos

RNF15 El sistema muestra todas las agendas de los especialistas que atienden el servicio solicitado.

RNF16 Los turnos reservados no se puede solapar con otro turno o cita registrada en el sistema.

RNF17 El horario de los turnos reservados no puede estar fuera del horario de atención definido en la agenda para el servicio seleccionado.

RNF18 Cambiar turno para un servicio en la agenda del especialista.

RNF19 Eliminar turno para un servicio en la agenda del especialista.

RNF20 Transferir turno de un especialista a otro siempre y cuando los dos atiendan el mismo servicio.

Requisitos No Funcionales del Sistema.

Los requisitos que a continuación se expresan no todos son específicos para el módulo de agenda que se desarrollará, algunos responden al Sistema de Información del CECMED al cual el módulo se integrará.

Seguridad

RNF1 El usuario debe autenticarse para acceder a las funcionalidades desarrolladas.

RNF2 La agenda solo podrá ser configurada por aquellos usuarios que se encuentren autenticados en el sistema y tengan el rol Especialista Superior. Las acciones de configuración son: Crear, Editar, Activar, Desactivar, Mostrar Agenda y Agendar Citas.

RNF3 Para gestionar un turno de servicio en el sistema es necesario estar autenticados en el módulo de Recepción y Preevaluación de Solicitudes, y además tener el rol Cliente, Especialista y Especialista Superior

RNF4 Las agendas de los especialistas solo estarán disponibles para operar sobre ellas si están activadas en el sistema.

Usabilidad

RNF5 A la hora de seleccionar la fecha de la cita o turno el usuario seleccionará en un componente de tipo calendario los días disponibles para ese servicio, los cuales fueron previamente definidos en la configuración de la agenda. Los días no disponibles se inhabilitarán y se mostrarán marcados con algún recurso visual de manera tal que ante la selección del usuario no les permitirá acceder al mismo.

RNF6 Cuando el usuario vaya a seleccionar la fecha de inicio y fin del periodo agendable, el componente permitirá moverse por los años a partir del mes y año actual.

RNF7 El componente para seleccionar el servicio en la configuración de la agenda solo mostrará los servicios que atiende el especialista al cual se le crea la agenda, y permitirá realizar selección múltiple y deshabilitar los servicios seleccionados.

RNF8 Al crear una agenda el componente de fecha no le permitrá seleccionar como fecha de inicio del periodo agendable una fecha antes de la fecha actual, para ello en el componente, además de no permitirle realizar la acción.

RNF9 El componente utilizado para mostrar la hora de inicio y fin de la cita o turno comenzará a partir de la hora de inicio del horario establecido en la configuración para ese día, y se incrementará en 15 min hasta la hora de fin, establecida en la configuración para ese día.

RNF10 Los componentes de selección simple o múltiple no solo permitirán seleccionar sino filtrar en función del patrón de texto introducido.

Soporte

RNF12 Se necesita un servidor de bases de datos que soporte volúmenes de datos (PostgreSQL).

RNF13 Se elaborará un paquete de instalación.

Portabilidad

RNF14 El sistema será multiplataforma (Linux o Windows).

RNF15 Garantizar el acceso controlado a la información. Este debe influir sobre cómo se presentan las interfaces para cada usuario dependiendo del nivel de acceso a la información.

Hardware

RNF16 Contar con servidor profesional para desplegar la aplicación, aunque se podrá desplegar la aplicación en una MV que se encuentre corriendo sobre un servidor físico de tales características.

RNF17 Contar con al menos 8 GB de memoria RAM.

RNF18 Contar con al menos 500 GB de capacidad en HDD

2.9 Justificación de los actores del sistema

Un actor del sistema representa cualquier persona, entidad o sistema externo al mismo, que interactúe con él para intercambiar información o como receptor pasivo de esta. Pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado.

Cada trabajador del negocio (inclusive si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema.

2.10 Actores y Casos de uso del sistema

Tabla 5 Actores y Casos de uso del sistema

Actor	Descripción
Cliente	Los usuarios con el rol Cliente podrán gestionar turnos de servicio en las agendas de los especialistas que atienden el servicio que los mismos solicitan, en este caso podrán crear, cambiar y eliminar el turno.
Especialista	Los usuarios con el rol Especialista podrán gestionar turnos de servicio a nombre de los usuarios Clientes, en las agendas de los especialistas que atienden el servicio que los mismos solicitan, en este caso podrán crear, cambiar y eliminar el turno.
Especialista Superior	Los usuarios con el rol Especialista Superior podrán gestionar turnos de servicio a nombre de los usuarios Clientes, en las agendas de los especialistas que atienden el servicio que los mismos solicitan, en este caso podrán crear, cambiar y eliminar el turno. Además podrán transferir los turnos de una agenda a otra para el mismo servicio. Además gestionará las agendas de los especialistas, sus citas y se le permitirá activar y desactivar las agendas creadas.

Diagrama de caso de uso del sistema.

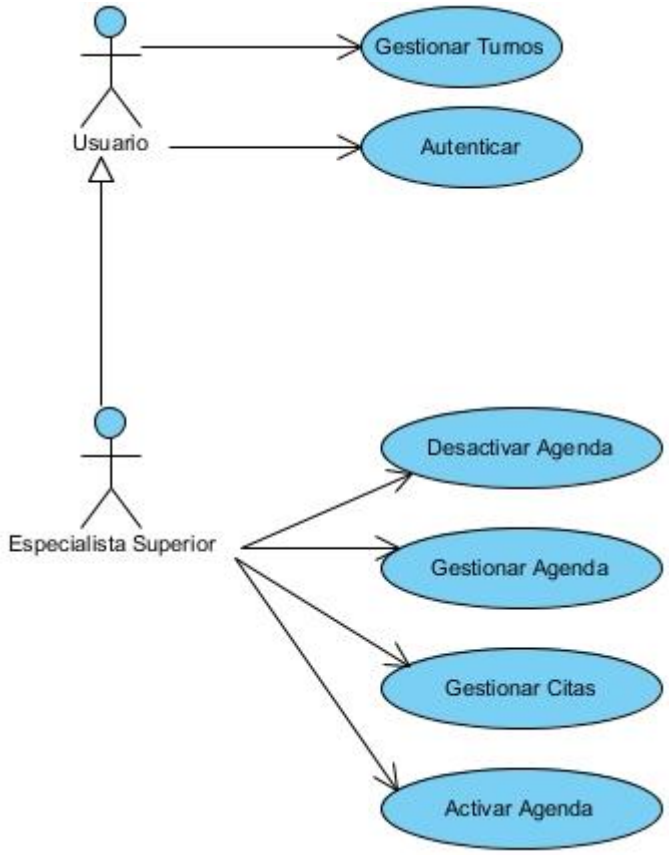

Fig. No 8 Diagrama caso de uso del sistema

2.11 Descripción de Casos de Uso del Sistema.

Caso de Uso Gestionar Turno.

Tabla 6 Caso de uso del sistema gestionar turno

Caso de Uso Gestionar Turno
Actor(es): Especialista, Especialista Superior, Cliente.
Descripción: Permite a los usuarios con el rol Especialista, Especialista Superior, Cliente reservar turnos para los servicios que ofrece el CECMED, los mismos no pueden solaparse con otros turnos o citas reservadas en el sistema, así como tampoco podrá reservarse fuera del horario definido en la agenda para la atención del turno. Además permitirá a los usuarios cambiar un turno o eliminarlo.
Referencias:

Sección: Reservar Turno **Propósito:** Reservar turno para un servicio. **Prioridad:** Alta **Pre Condiciones:** El siguiente flujo se ejecuta una vez el usuario con el rol cliente, especialista o especialista superior se encuentra autenticado en el sistema y va a realizar una Solicitud de Servicio. . El caso de uso gestionar turno – sección Reservar Turno es un caso de uso incluido del Caso de Uso Gestionar Solicitud.	
Flujo Normal.	
1. Desde la solicitud selecciona la acción Consultar Agenda.	2. El sistema permite obtener la agenda de un especialista dado el servicio seleccionado, por defecto muestra la agenda de uno los especialistas que atienden el servicio seleccionado, Un especialista puede tener más de una agenda por lo cual se mostrará al usuario la agenda que contenga en el periodo agendable la fecha actual y se encuentre activa, por defecto se muestra en el componente la disponibilidad para el mes en curso, aunque puede seleccionar semana o día, además se muestra un formulario para seleccionar el especialista, en el componente se muestran las agendas de todos los especialistas que tengan agendas creadas para el servicio seleccionado, además se captura la fecha del turno, hora de inicio y un campo deshabilitado donde se mostrará la fecha de fin del turno.
3. El usuario selecciona los datos Fecha y Hora del Turno	4. El sistema calcula la Hora del turno atendiendo a la duración definida para el servicio para la etapa la pre evaluación (minutos)
5. El usuario selecciona la acción Reservar Turno.	6. El sistema chequea que se cumplan los requisitos establecidos para reservar un turno: • Los datos obligatorios no se encuentren en blanco,

	• la Hora de Inicio y la Hora de Fin del turno se encuentre dentro del horario definido para la fecha seleccionada. • la Hora de Inicio y la Hora de Fin no se encuentren dentro del horario definido para los turnos o citas registradas en la agenda. • la Fecha se encuentra dentro de la fecha definida en el periodo agendable de la agenda
	7. Se guardan los siguientes datos: Fecha, Hora de Inicio, Hora de Fin, y además devuelve hacia la solicitud, la FF, HI del turno y EE que atenderá la solicitud. Finaliza el Caso de Uso.
Flujo Alterno	
FA 1. En el caso que no se llenen los campos obligatorios.	
	9. Mostrar mensaje indicando que hay campos obligatorios vacios. "Verifique que no hayan campos vacios". Continua em el paso 3 del flujo normal.
FA 2. La cita se solapa con otra cita o turno	
	9. Mostrar mensaje indicando que cita se solapa com una cita o turno registrado en el sistema. "Seleccione otra fecha o horario, existen turnos o citas en la fecha y hora seleccionadas" Continua em el paso 3 del flujo normal.
FA 3. La Fecha no se encuentra dentro de la fecha de inicio y fin del periodo agendable definido para la agenda.	
	9. Mostrar mensaje indicando que la Fecha seleccionada no se encuentran dentro de la fecha de inicio y fin del periodo agendable definido para la agenda. Mensaje: "La Fecha seleccionada deben estar comprendidas entre Fecha de Inicio del periodo

	agendable Fecha de Fin del periodo agendable". Continua em el paso 3 del flujo normal.
FA 4. La Hora Inicio y Fin de la cita no se encuentran dentro de la Hora de inicio y fin definida para el día en la agenda	
	9. Mostrar mensaje indicando que la Hora de Inicio y Fin del turno no se encuentran dentro de la fecha de inicio y fin configurada en la agenda para el día seleccionado. Mensaje: "La Hora de Inicio y Fin deben estar comprendidas entre Hora de Inicio y Fin definida para la fecha. Continua em el paso 3 del flujo normal.
Sección: Eliminar Turno **Propósito:** Eliminar turno para un servicio. **Prioridad: Alta** **Pre Condiciones:** El siguiente flujo se ejecuta una vez el usuario con el rol cliente, especialista o especialista superior se encuentra autenticado en el sistema, y existe un turno reservado para la solicitud, se encuentra con la solicitud abierta en pantalla. El caso de uso gestionar turno – sección eliminar turno es un caso de uso incluido del Caso de Uso Gestionar Solicitud.	
Flujo Básico	
1. Selecciona la acción Cancelar Solicitud	3. Muestra mensaje pidiendo confirmación para cancelar la solicitud.
2. Selecciona la acción Sí.	4. Chequea que los datos: Nombre de la Agenda, Fecha y Hora no se envíen vacíos.
	5. Busca el turno.
	6. Elimina el turno.
Sección: Cambiar Turno **Propósito:** Editar turno para un servicio. **Prioridad:** Alta	

Pre Condiciones: El siguiente flujo se ejecuta una vez el usuario con el rol cliente, especialista o especialista se encuentra autenticado en el sistema, tiene un turno reservado para una solicitud, se encuentra con la solicitud abierta en pantalla, selecciona la acción Editar Solicitud y selecciona la acción consultar agenda. El caso de uso gestionar turno – sección cambiar turno es un caso de uso incluido del Caso de Uso Gestionar Solicitud.	
1. Desde la solicitud selecciona la acción Consultar Agenda.	2. El sistema permite obtener la agenda de un especialista dado el servicio seleccionado, por defecto muestra la agenda de uno los especialistas que atienden el servicio seleccionado, Un especialista puede tener más de una agenda por lo cual se mostrará al usuario la agenda que contenga en el periodo agendable la fecha actual y se encuentre activa, por defecto se muestra en el componente la disponibilidad para el mes en curso, aunque puede seleccionar semana o día, además se muestra un formulario para seleccionar el especialista, en el componente se muestran las agendas de todos los especialistas que tengan agendas creadas para el servicio seleccionado, además se captura la fecha del turno, hora de inicio y un campo deshabilitado donde se mostrará la fecha de fin del turno.
3. El usuario selecciona los datos Fecha y Hora del Turno	4. El sistema calcula la Hora del turno atendiendo a la duración definida para el servicio para la etapa la pre evaluación (minutos)
5. El usuario selecciona la acción Reservar Turno.	6. Chequea si ya la solicitud tiene un turno reservado.
	7. El sistema chequea que se cumplan los requisitos establecidos para reservar un turno: • Los datos obligatorios no se encuentren en blanco, • la Hora de Inicio y la Hora de Fin del turno se encuentre dentro del horario definido para la fecha seleccionada. • la Hora de Inicio y la Hora de Fin no se encuentren dentro del horario definido

	para los turnos o citas registradas en la agenda. la Fecha se encuentra dentro de la fecha definida en el periodo agendable de la agenda
	8. Se edita del turno los siguientes datos: Fecha, Hora de Inicio, Hora de Fin, y además devuelve hacia la solicitud, la FF, HI del turno y EE que atenderá la solicitud. Finaliza el Caso de Uso.
Flujo Alterno	
FA 1. En el caso que no se llenen los campos obligatorios.	
	9. Mostrar mensaje indicando que hay campos obligatorios vacios. "Verifique que no hayan campos vacios". Continua em el paso 3 del flujo normal.
FA 2. La cita se solapa con otra cita o turno	
	9. Mostrar mensaje indicando que cita se solapa com una cita o turno registrado en el sistema. "Seleccione otra fecha o horario, existen turnos o citas en la fecha y hora seleccionadas" Continua em el paso 3 del flujo normal.
FA 3. La Fecha no se encuentra dentro de la fecha de inicio y fin del periodo agendable definido para la agenda.	
	9.Mostrar mensaje indicando que la Fecha seleccionada no se encuentran dentro de la fecha de inicio y fin del periodo agendable definido para la agenda. Mensaje: "La Fecha seleccionada deben estar comprendidas entre Fecha de Inicio del periodo agendable Fecha de Fin del periodo agendable". Continua em el paso 3 del flujo normal.
FA 4. La Hora Inicio y Fin de la cita no se encuentran dentro de la Hora de inicio y fin definida para el día en la agenda	

	9. Mostrar mensaje indicando que la Hora de Inicio y Fin del turno no se encuentran dentro de la fecha de inicio y fin configurada en la agenda para el día seleccionado. Mensaje: “La Hora de Inicio y Fin deben estar comprendidas entre Hora de Inicio y Fin definida para la fecha. Continua em el paso 3 del flujo normal.

2.12 Técnicas de validación de requisitos empleadas

El objetivo de la validación de requisitos es descubrir problemas en la redacción de requisitos antes de comprometer recursos en su implementación. Los requisitos deben revisarse para descubrir omisiones, conflictos y ambigüedades.

Para la validación de los requisitos se emplearon varias técnicas:

Revisiones (Reviews)

Es la fórmula más empleada para validación. Un grupo de personas (incluyendo usuarios) se ocupan de revisar el documento de requisitos. Consta de tres fases:

1. Búsqueda de problemas.
2. Reunión.
3. Acuerdos.

2.13 Patrones de casos de usos.

Un patrón se define como una solución probada con éxito que aparece una y otra vez ante determinado tipo de problema en un contexto dado. Los patrones se definen por un nombre, un problema, una solución y las consecuencias de su aplicación; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias, y considera los puntos fuertes y compromisos.

Inclusión

Este patrón se utiliza en la aplicación web en los casos de usos gestionar agendas y gestionar citas ya que no siempre se tienen que usar todas las funcionalidades del gestionar.

CRUD (Creating, Reading, Updating, Deleting)

Este patrón se utiliza en la aplicación web, ya que los casos de uso crear agenda, actualizar agenda, mostrar agenda y borrar agenda conforman el caso de uso gestionar agendas.

CRUD Parcial

Este patrón se utiliza en la aplicación web, debido a que el caso de uso gestionar turno está compuesto por las funcionalidades crear turno, mostrar y editar turno.

Conclusiones parciales

- ✓ Al concluir el capítulo se identificaron los requisitos funcionales con los cuales se llegó a una definición más formal de lo que el sistema debe hacer y ayudó a establecer un convenio entre desarrolladores y clientes de las características funcionales del módulo.

- ✓ A partir de la utilización de los patrones de casos de uso se determinaron los casos de uso y la relación con los actores del sistema, esta relación se especificó en detalle en el diagrama de casos de uso del sistema.

- ✓ Para hacer una valoración del trabajo se emplearon técnicas de validación de requisitos, lo que dio como resultado la aprobación del cliente de los requerimientos identificados y aseguró la calidad de los mismos.

CAPÍTULO 3: Diseño del sistema

Introducción

En este capítulo se hace un estudio del diseño del sistema. Se representa la realización de casos de uso del diseño a través de los diagramas de clase del diseño y de interacción correspondientes a cada uno. A medida que se representan los diagramas se hace alusión a las técnicas empleadas para obtener mejores resultados en la solución, específicamente los patrones de diseño empleados en la obtención de las clases del diseño y la estructuración de los diagramas. En el capítulo se abordan además los detalles del diseño de la base de datos mediante la representación del Modelo de Datos y la descripción de las tablas, incluyendo el Modelo de Despliegue. Además se describe el proceso de tratamiento de errores.

3.1 Patrones

Es una descripción de un problema y la solución, a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias, y considera los puntos fuertes y compromisos.

3.1.1 Patrones Arquitectónicos

Centrados en la arquitectura del sistema. Definen una estructura fundamental sobre la organización del sistema. Proveen un conjunto predefinido de subsistemas, cuáles son sus responsabilidades y como se interrelacionan.

Arquitectura Cliente/Servidor

En la aplicación informática se hace uso de la arquitectura Cliente/Servidor ya que todos los clientes están conectados a través del uso de la red. El servidor debe ser lo suficientemente potente debido a que todas las gestiones que se realizan se concentran en él, mientras que el cliente puede cumplir con especificaciones más ligeras debido a que solo va a realizar peticiones, esperar y recibir respuesta del servidor. En dicha aplicación se utilizó como servidor de aplicaciones *Tomcat*, servidor de BD *PostgreSQL* y como cliente los navegadores web. La centralización del control, la escalabilidad, el fácil mantenimiento y la seguridad son algunas de las ventajas que provee este modelo, por lo que permitió aprovechar para beneficio del sistema propuesto.

Modelo Vista Controlador (Modelo 2)

Modelo-Vista-Controlador (MVC) es una arquitectura de diseño de software que separa los datos de la aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos de forma que las modificaciones al componente de la vista pueden ser hechas con un mínimo impacto en el componente del modelo de datos, en este caso se utilizó el framework Grails que , donde el Modelo va a constar de objetos groovy entre los que se encuentran los Agenda y Turno, la Vista contiene las páginas GSP , el código que provee datos dinámicos a las misma y el Controlador se implementa siempre en clases con sufijo Controller por ejemplo FrontController el cual recibe peticiones y ejecuta un conjunto de pasos para servir una respuesta. Este modelo permitió normalizar y estandarizar el desarrollo de la aplicación, además de brindar una API muy bien definida; que cualquiera que decida utilizarla, podrá reemplazar el modelo, la vista o el controlador, sin demasiada dificultad, mientras que la conexión entre el modelo y sus vistas es dinámica y se produce en tiempo de ejecución, no en tiempo de compilación.

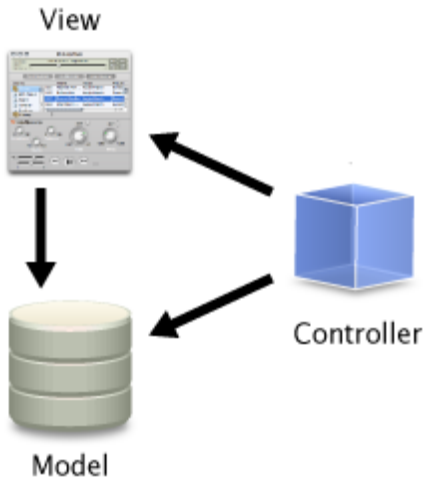

Fig. No 9 Arquitectura Modelo Vista Controlador

Arquitectura de Grails

La arquitectura de Grails está concebida por 3 capas lógicas principales: Web Layer, Service Layer y Data Layer, su interacción se lleva a cabo mediante interfaces que definen funcionalidades que la misma debe brindar, también llamadas fachadas cuya función fundamental es asegurar que el acoplamiento sea el más bajo posible y la abstracción del funcionamiento de la capa inferior.

Grails implementa el patrón (Modelo-Vista-Controlador, MVC) en la que se separa la lógica empresarial de la presentación de la aplicación, lo cual le permite cambiar fácilmente el aspecto de su aplicación sin modificar su comportamiento.

Web Layer: La capa web se compone de dos partes principales: vistas y controladores, teniendo en Controllers las clases controladoras y en View las Groovy Server Pages o más conocidas como GSP. En la Lógica de Presentación se manejará todo el flujo web utilizando la implementación del patrón MVC que brinda Grails mediante Spring MVC. Esto permite cambiar fácilmente el aspecto de la aplicación, sin modificar su comportamiento. La capa de presentación se compone principalmente de: modelo, vistas, controladores.

Modelo: Una de las actividades fundamentales llevadas a cabo por los controladores en Grails es obtener los datos que serán mostrados en la vista, está definido por clases de dominio de Groovy las cuales se mapean en la base de datos utilizada y permiten el manejo y almacenamiento de los datos.

Vista: La tecnología JSP es utilizada por Grails para la interacción con el usuario, basada en una implementación mediante GSP, el mismo permite a los desarrolladores mezclar las etiquetas de lenguajes de marcas tradicionales como HTML con código Java para producir vistas dinámicas. Las vistas son los recursos que junto al modelo generado por los controladores le permiten al cliente visualizar la información.

Controlador: Un controlador de Grails es una clase responsable por el manejo de los pedidos provenientes de la aplicación, es el encargado de mantener el flujo de comunicación entre las vistas y el modelo, están definidos por los Groovy controllers o controladores Groovy.

Service Layer: La capa de servicios es la encargada de encapsular toda la lógica de la aplicación en fachadas de negocio que son utilizadas por los controladores en la capa de presentación, dejándoles el manejo de flujo de solicitudes con redirecciones. Sus clases radicarán, según la arquitectura propuesta por Grails, en el paquete Services.

Data Layer: La capa de datos es la encargada de manejar los objetos de acceso a datos separándolos del mecanismo de persistencia utilizado, mediante las interfaces que exponen las operaciones de persistencia. Grails para evitar a los programadores tener que trabajar directamente con el sistema gestor de base de datos y tablas, permite trabajar con objetos en su lugar. Utiliza Hibernate, la biblioteca más popular para Java, como una herramienta de Mapeo Objeto-Relacional (ORM). Sin embargo dada la naturaleza dinámica de Grails y la adopción del convenio sobre la configuración, se crea sobre una versión superior de la implementación de Hibernate llamado Grails Mapeo Objeto-Relacional (GORM) que simplifica el trabajo con Hibernate y elimina cualquier configuración externa.

3.2 Patrones de diseño

Relacionados con el diseño de los objetos y frameworks de pequeña y mediana escala. Aplicables al diseño de una solución para conectar los elementos de gran escala que se definen mediante los patrones de arquitectura, y durante el trabajo de diseño detallado para cualquier aspecto del diseño local. También se conocen como patrones de micro-arquitectura.

Patrones de diseño GRASP

Los patrones de diseño son descripciones de clases y objetos relacionados, que están particularizados para resolver un problema de diseño general en un determinado contexto.

Experto: Este patrón es muy utilizado para asignar responsabilidades. Una clase contiene la información necesaria para llevar a cabo sus funcionalidades. Permite conservar el encapsulamiento, debido a que los objetos se valen de su propia información para cumplir lo que se le pide. Proporciona que las clases cuenten con la funcionalidad requerida, brindando así una alta cohesión.

En el módulo se aplica a clases como **AgendaController.groovy** y **TurnoController.groovy.**

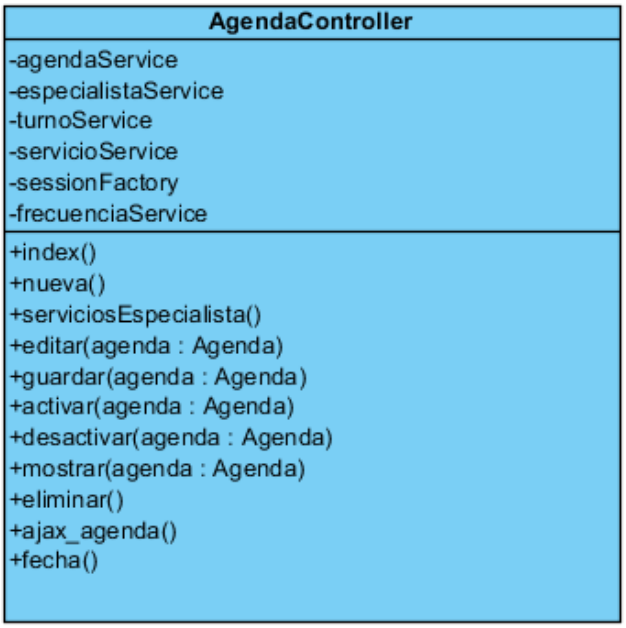

Fig. No 10 Ejemplo de clase que usa patrón Experto

Creador: Este patrón establece la creación de instancias entre clases. Brinda un soporte a bajo acoplamiento. Guía la asignación de responsabilidades relacionada con la creación de objetos, tarea frecuente en la programación orientada a objeto.

En el módulo se aplica a clases como **TurnoService.groovy** y **AgendaService.groovy.**

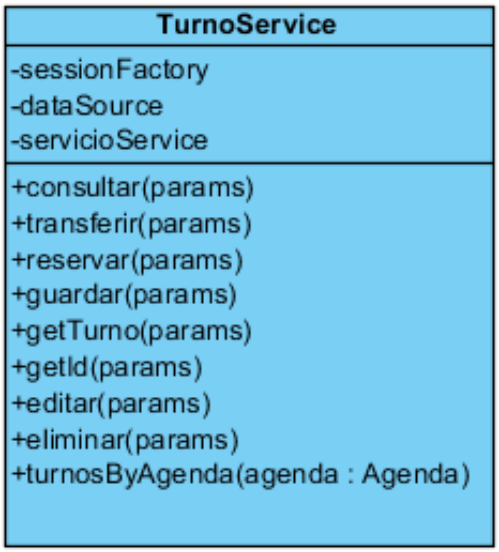

Fig. No 11 Ejemplo de clase que usa patrón Creador

Alta Cohesión: Este patrón se encarga de guiar el diseño. Permite que las clases del diseño realicen las funcionalidades necesarias para cumplir con su responsabilidad. Mejora la claridad y facilidad para entender el diseño. Busca soluciones para asignar los métodos a las clases de forma coherente, completa y relacionada, permitiendo el cambio, poniendo toda la información que se necesita controlar a la vista en el mismo fichero. Fomenta la reutilización.

Se aplica en el sistema en clases como **ServicioService.groovy, EspecialistaService.groovy, FrecuenciaService.groovy**

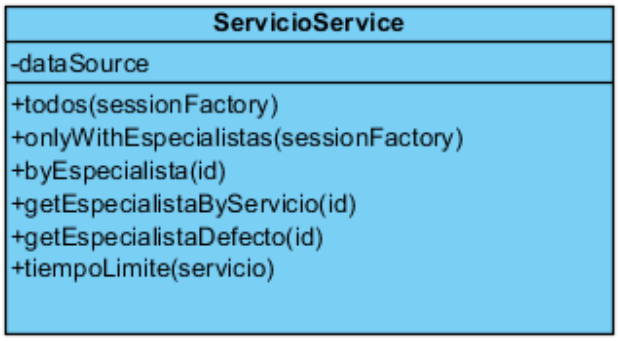

Fig. No 12 Ejemplo de clase que usa patrón Alta Cohesión

Bajo acoplamiento: Es imprescindible utilizarlo cuando se vaya a diseñar. Permite la reutilización y el diseño de clases más independientes, minimiza el impacto de los cambios.

Se aplica en el sistema en clases como **TurnoService.groovy, Frecuencia.groovy, Agenda.groovy.**

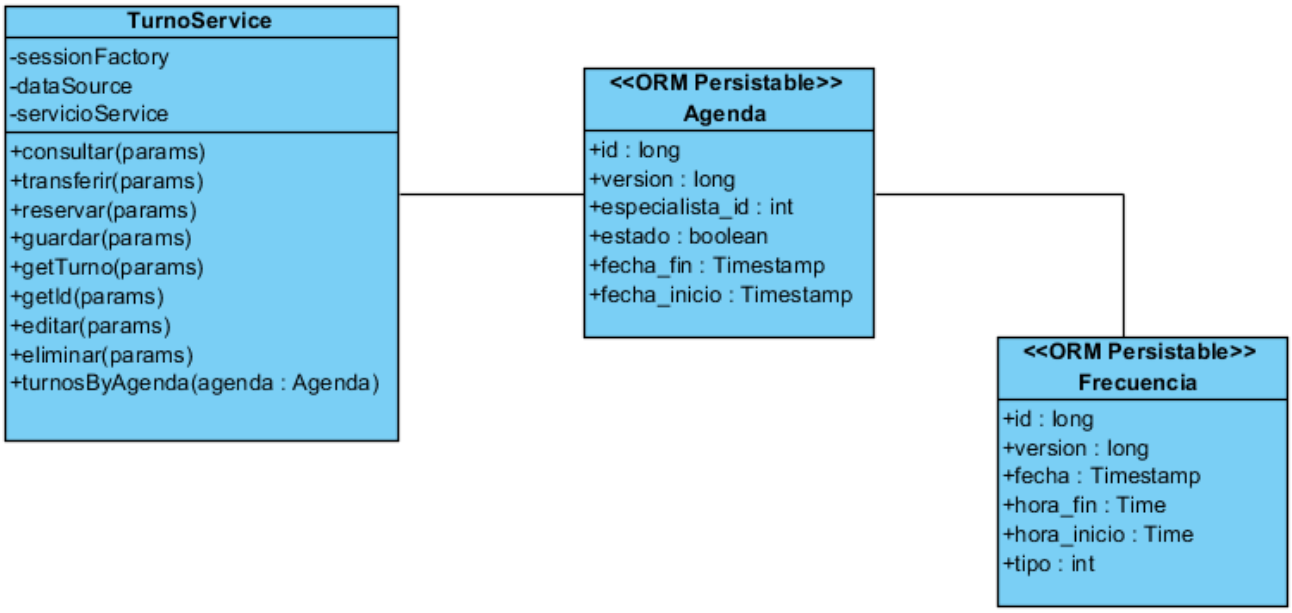

Fig. No 13 Ejemplo de clase que usa patrón Bajo acoplamiento

Patrón de diseño GOF

Se utilizaron además los patrones GoF (Banda de los Cuatro, del inglés *Gang-of-Four*) los cuales se clasifican en dependencia del propósito para los que hayan sido definidos: creación, estructurales y de comportamiento (Guerrero, et al., 2013).

Singleton: El objetivo de este patrón es asegurarse de que de una clase solo existe una instancia y que esta es accesible, o mejor dicho, ofrecer un punto de acceso a ella.

Se aplica este tipo de patrón a todas las clases de servicio, ya que Groovy así lo implementa. Las clases son **AgendaService.groovy, EspecialistaService.groovy, FrecuenciaService.groovy, ServicioService.groovy y TurnoService.groovy.**

3.3 Diagrama de clases del diseño

Un diagrama de clases proporciona una perspectiva estática que representa el diseño estructural del sistema mostrando un conjunto de clases, sus atributos y las relaciones entre ellos.

Los diagramas de clases son utilizados durante el proceso de diseño de los sistemas, donde se establece el diseño conceptual de la información que contendrá el sistema.

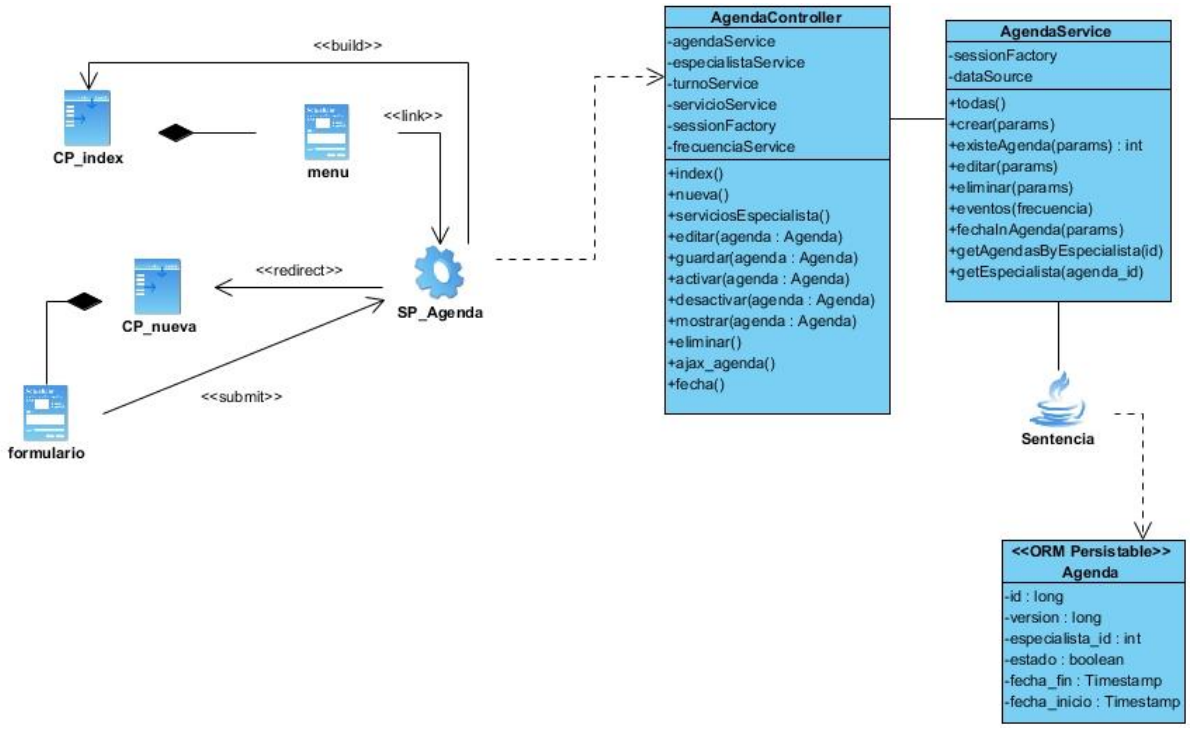

Fig. No 14 Diagrama de clase del diseño de caso de uso crear agenda

3.4 Diagrama de secuencias

El diagrama de secuencia de UML muestra la forma en que los objetos se comunican entre sí al transcurrir el tiempo. Los diagramas están compuestos por objetos con su línea de vida, mensajes intercambiados entre objetos de una secuencia ordenada y una línea de vida activa, se utilizan con frecuencia para validar los casos de uso.

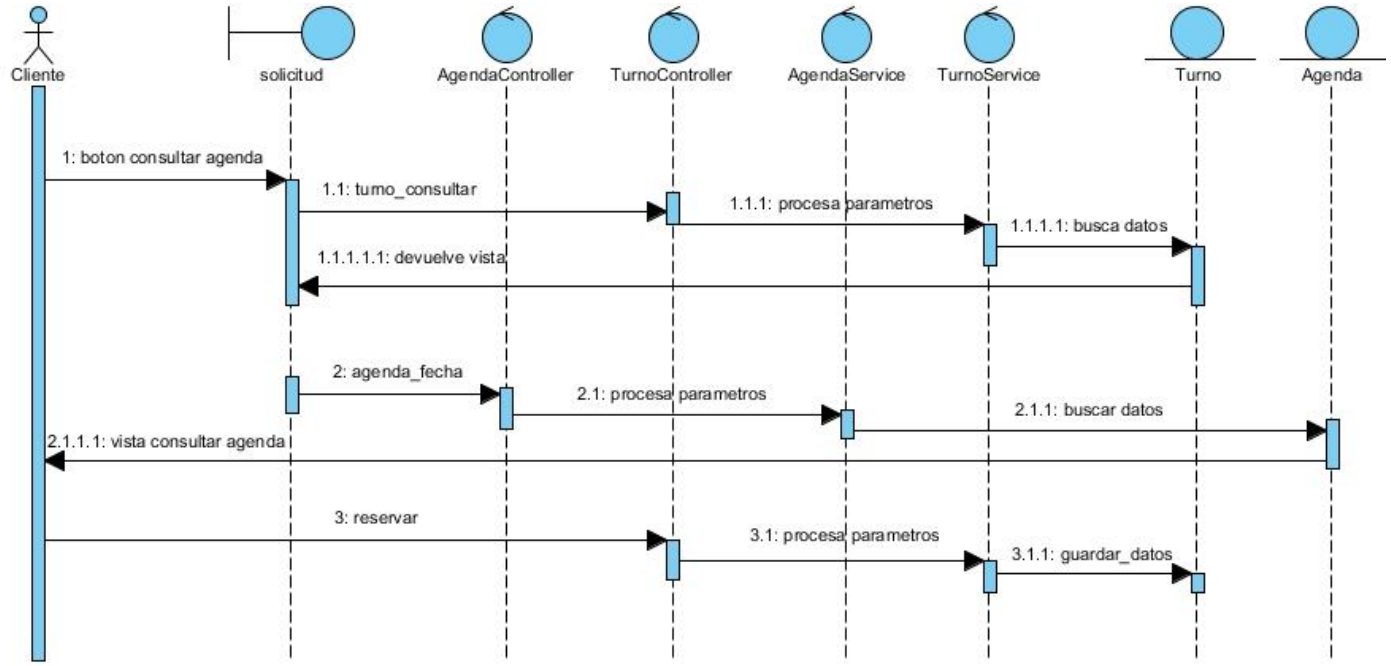

Fig. No 15 Diagrama de secuencias del caso de uso reservar turno

3.5 Diseño de base de datos

El diseño de una base de datos es de suma importancia ya que de ello dependerá que nuestros datos estén correctamente actualizados y la información siempre sea exacta. Con un buen diseño de base de datos se puede obtener reportes efectivos y eficientes. Permitirá definir la estructura de los datos que debe tener la base de datos del sistema.

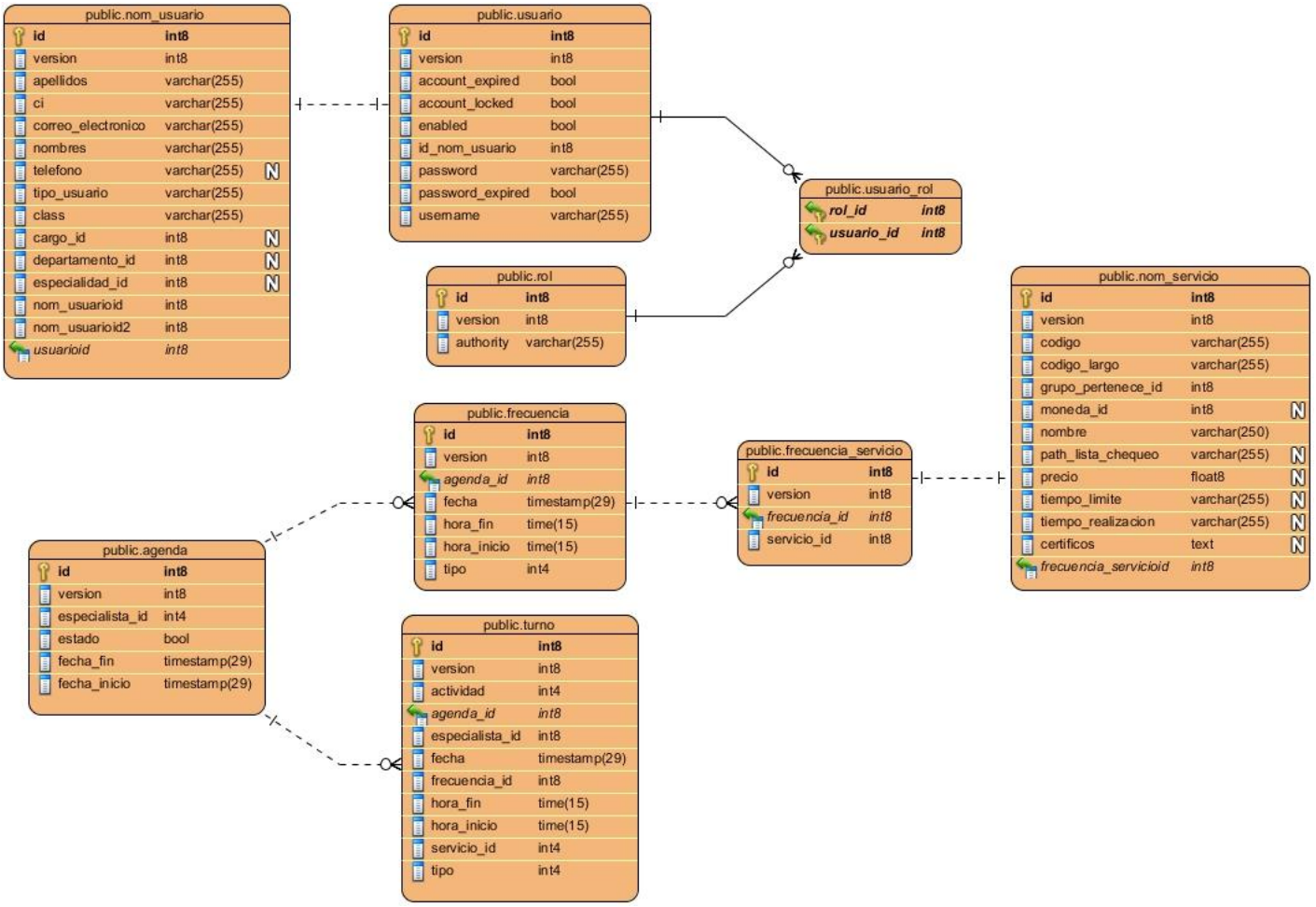

Fig. No 16 Diagrama de modelo de datos

3.6 Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software, así como la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software. Son capaces de describir la arquitectura física del sistema durante la ejecución, en términos de: procesadores dispositivos componentes de software además de la topología del sistema.

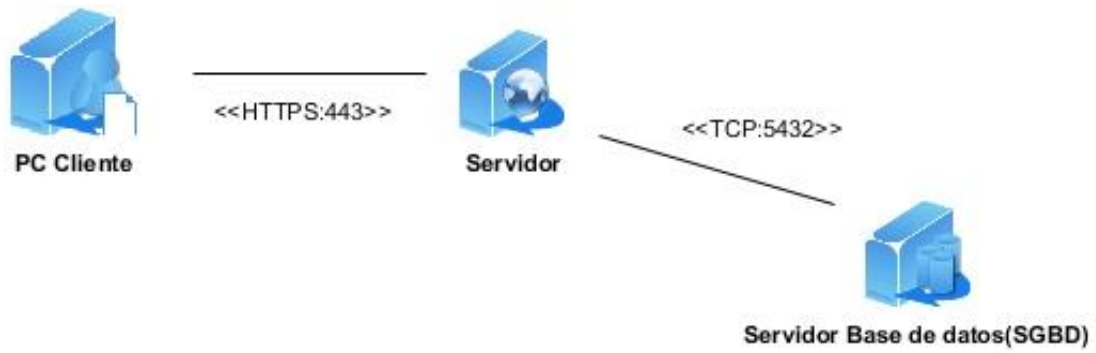

Fig. No 17 Diagrama de despliegue

Nodo: Un nodo es un elemento de hardware o software. Se representa por una terminal

Descripción de los nodos.

PC Cliente: Representa los ordenadores que van a permitir a los usuarios conectarse con el sistema.

Servidor: Representa el servidor Tomcat donde se encuentra desplegada la aplicación.

Servidor Base de datos: Representa el servidor donde se encuentra instalado el gestor de base de datos PostgreSQL con la BD del sistema.

Conclusiones parciales

Al concluir el presente capítulo se realizó el análisis de la arquitectura del sistema propuesta por el marco de trabajo Grails, se han generado artefactos tales como los diagramas de clases del diseño elaborados para darle solución al sistema, así como el modelo de datos perteneciente al mismo. Se hizo referencia a los patrones seleccionados para el diseño e implementación del sistema.

CAPÍTULO 4: Implementación y Resultados esperados

Introducción

Al concluir el diseño del sistema, se realizan las pruebas de caja negra con el propósito de comprobar y documentar cómo se comporta este módulo cuándo se miden sus principales funcionalidades, de acuerdo a las especificaciones descritas en los casos de uso del sistema.

4.1 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, bibliotecas cargadas dinámicamente, entre otros. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.

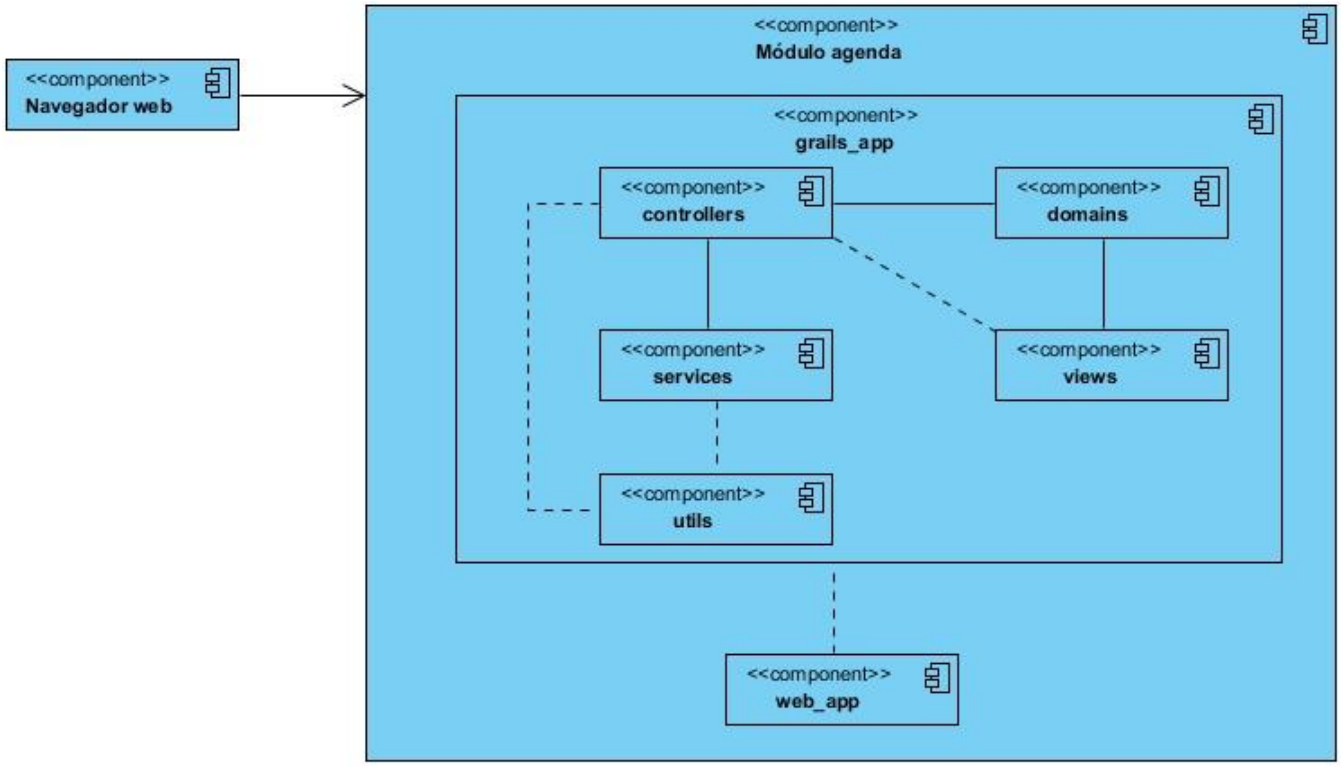

Fig. No 18 Diagrama de componentes

Descripción de los componentes del diagrama de componentes de la aplicación web.

Tabla 7 Compontes del diagrama de componente de la aplicación web

Componente	Descripción
Navegador web	Indica que se va a acceder a la aplicación web.
Módulo agenda	Componente principal el cual representa la aplicación.
Grails_app	Contiene todos los archivos ejecutados en el lado servidor.
Controllers	Componente que contiene todos los controladores del módulo.
Domains	Componente que contiene todos los dominios del módulo.
Services	Indica donde se van a implementar los servicios del módulo.
Views	Contiene las vistas del módulo.
Utils	Contiene las clases útiles que se utilizarán.
Web_app	Componente que contiene todos los recursos que se ejecutaran en el lado cliente.

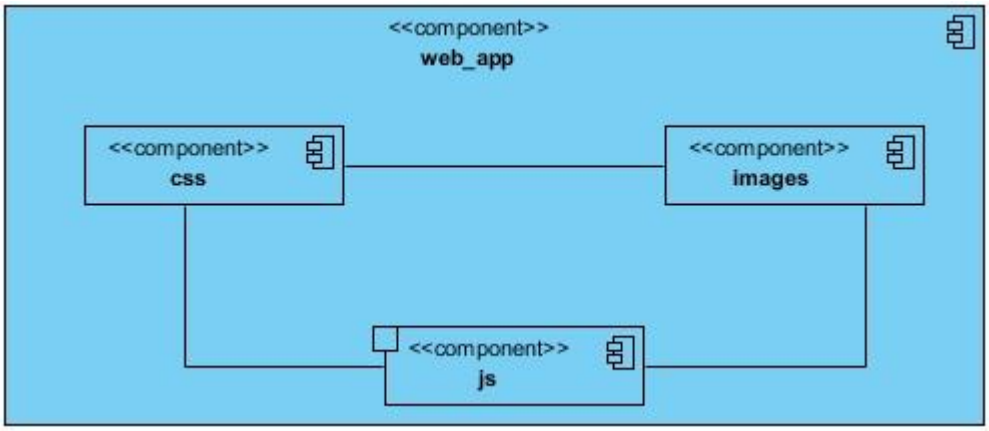

Fig. No 19 Diagrama de componente del componente web_app

Descripción de los componentes del diagrama de componente web_app.

Tabla 8 Componentes del diagrama de componente web_app

Componente	Descripción
css	Contiene todas las hojas de estilos del módulo.
images	Contiene todas las imágenes que se van a utilizar en el módulo.
js	Contiene todos los archivos javascript.

4.2 Estándares de Codificación

Un estándar de codificación, son reglas que se siguen para la escritura del código fuente. Para una mejor comprensión del código, se definieron una serie de estándares basados en diversas reglas.

Estilos para la capitalización

Clases: Para todo nombre de clase, la primera letra debe de ser mayúscula, si son varias palabras se debe de intercalar entre mayúsculas y minúsculas, este mecanismo de nombre es llamado ***camelCase***. Por ejemplo: **AgendaController**, **AgendaService** y **FrecuenciaService**.

```
@Transactional
class TurnoService {

    def sessionFactory
    def dataSource
    def servicioService
```

Fig. No 20 Ejemplo de camelCase para una clase

Métodos: Para los métodos de clases, la primera letra debe ser minúscula, si son varias palabras se debe de intercalar entre minúsculas y mayúsculas, para el caso de los métodos de clases aplica el mecanismo de ***camelCase*.** Por ejemplo, **fechaIn**, **existeAgenda** y **getEspecialista**.

```
def todas(){
    def agendas = Agenda.list(order:'asc')
}


//metodo para saber si existe una agenda existe
Integer existeAgenda(params)
{
    def existe = Agenda.findAll(" From Agenda Where ((fechaInicio >= :inicio
    especialista_id = :espec "
        ,[inicio: new Date(params.fechaInicio),
          fin: new Date(params.fechaFin),
          espec: params.especialista.toInteger()])

    return existe.size()
}
```

Fig. No 21 Ejemplo de camelCase para un método

Variables: Para las variables, se aplica el caso de los métodos, donde la primera letra es minúscula, y las demás se deben de guiar por el mecanismo de ***camelCase***, deben ser cortos y descriptivos en sí mismo. Por ejemplo: **turnoService**, **sessionFactory** y **dataSource**.

```
def sessionFactory
def dataSource
def servicioService
```

Fig. No 22 Ejemplo de camelCase para las variables

4.3 Reglas de codificación

- ✓ No usar caracteres simples para el nombre de las variables i, n, s, entre otros. Una excepción de esta regla son las variables dentro de los ciclos.
- ✓ No usar nombres de variables que coincidan con palabras reservadas.
- ✓ Los comentarios deben estar en el mismo nivel del código.
- ✓ Las llaves se deben poner al mismo nivel del código que las contiene.
- ✓ Usar una línea en blanco para separar agrupaciones lógicas del código.

- ✓ Debe dejarse una y solo una línea en blanco entre cada método dentro de las clases.
- ✓ Las llaves deben ser utilizadas sobre líneas separadas y no sobre la misma línea como en ***switch***.

4.4 Pruebas

La construcción de un *software* tiene como objetivo satisfacer una necesidad planteada por un cliente. Pero ¿cómo se puede saber si el producto construido se corresponde exactamente con lo que el cliente les pidió y funciona correctamente? Por ese motivo es que se hace necesario llevar a cabo, en paralelo al proceso de desarrollo, un proceso de evaluación o comprobación de los distintos productos o modelos que se van generando (Juristo, 2005).

Para dicho proceso se pueden usar distintos tipos de técnicas. En general, éstas se agrupan en dos categorías: Técnicas de Evaluación Estáticas y de Evaluación Dinámicas. Las pruebas de *software* son los procesos que permiten verificar y revelar la calidad de un producto. Son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa. Básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas. Las pruebas de *software* se integran dentro de las diferentes fases del ciclo de vida del proyecto dentro de la Ingeniería de *software*. Para comprobar que la aplicación cumple con los requisitos funcionales definidos, es necesario realizarle pruebas antes de dar por terminado su proceso de implementación. El propósito de éstas es simular una carga de producción real y observar cómo se comporta el sistema bajo cargas intensivas. Esto permite solucionar los problemas de rendimiento, antes de poner la aplicación en marcha (Juristo, 2005).

4.5 Niveles y técnicas de pruebas

El proceso de evaluación de un *software* debe permitir comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el *software* en su conjunto (Juristo, 2005).

Para ello se cuenta con tres pasos fundamentales que se implementan de manera secuencial, los mismos son representados en la figura.

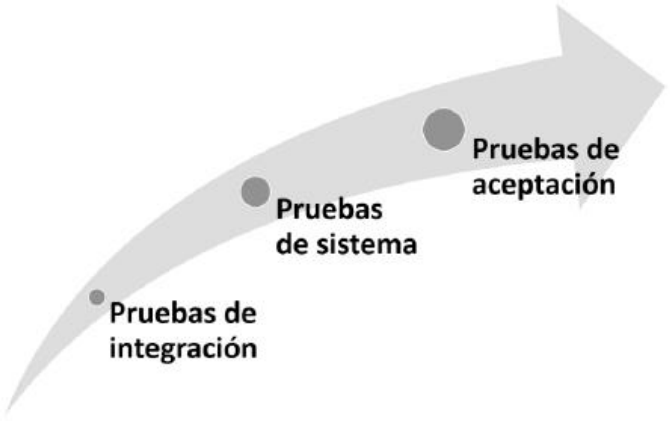

Fig. No 23 Niveles de prueba

Pruebas de integración

Las pruebas de integración se centran en probar la coherencia semántica entre los diferentes módulos, tanto de semántica estática (se importan los módulos adecuados; se llama correctamente a los procedimientos proporcionados por cada módulo), como de semántica dinámica (un módulo recibe de otro lo que esperaba). Normalmente estas pruebas se van realizando por etapas, englobando progresivamente más y más módulos en cada prueba. Las pruebas de integración se pueden empezar en cuanto se tengan unos pocos módulos, aunque no terminarán hasta disponer de la totalidad. En un diseño descendente (*top-down*) se empieza a probar por los módulos más generales; mientras que en un diseño ascendente se empieza a probar por los módulos de base (Pressman, 2010).

Pruebas del sistema

Este tipo de pruebas tiene como propósito ejercitar profundamente el módulo para verificar que se han integrado adecuadamente todos los elementos del sistema (hardware, otro *software*, etc.) y que se realizan las funciones adecuadas. Concretamente se debe comprobar que:

- ✓ Se cumplen los requisitos funcionales establecidos.
- ✓ El funcionamiento y rendimiento de las interfaces de hardware, software y de usuario.
- ✓ La adecuación de la documentación de usuario (Juristo, 2005).

Pruebas de aceptación

En las pruebas de aceptación su objetivo es la evaluación del producto y la realización de una revisión de la documentación final. Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el

sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable de cara al cliente; sino una vez pasada todas las pruebas de integración por parte del desarrollador.

La experiencia muestra que aún después del más cuidadoso proceso de pruebas por parte del desarrollador, quedan una serie de errores que sólo aparecen cuando el cliente se pone a usarlo. Muchos desarrolladores ejercitan unas técnicas denominadas "pruebas alfa" y "pruebas beta". Las pruebas alfa consisten en invitar al cliente a que venga al entorno de desarrollo a probar el sistema. Se trabaja en un entorno controlado y el cliente siempre tiene un experto a mano para ayudarle a usar el sistema y para analizar los resultados. Las pruebas beta vienen después de las pruebas alfa, y se desarrollan en el entorno del cliente, un entorno que está fuera de control. Aquí el cliente se queda a solas con el producto y trata de encontrarle fallos (reales o imaginarios) de los que informa al desarrollador.

4.6 Métodos de prueba

Prueba de caja blanca

Método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al emplear los métodos de prueba de caja blanca, el ingeniero del software podrá derivar casos de prueba que:

- ✓ Garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez.
- ✓ Ejerciten los lados verdadero y falso de todas las decisiones lógicas.
- ✓ Ejecuten todos los bucles en sus límites y dentro de sus límites operacionales.
- ✓ Ejerciten estructuras de datos internos para asegurar su validez.

Prueba de Caja Negra

Se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

La prueba de caja negra no es una alternativa a las técnicas de prueba de la caja blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la caja blanca. Muchos autores consideran que estas pruebas permiten encontrar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

- ✓ **Técnica de la Partición de Equivalencia:** esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del módulo.
- ✓ **Técnica del Análisis de Valores Límites:** esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- ✓ **Técnica de Grafos de Causa-Efecto:** es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Dentro del método de Caja Negra la técnica de la *Partición de Equivalencia* es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el módulo, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar (Pressman, 2010)

La técnica utilizada en la presente investigación se denomina Prueba de Partición Equivalente, la cual divide el dominio de entrada de un programa en clases de datos, a partir de las cuales deriva los casos de prueba. Cada una de estas clases de equivalencia representa a un conjunto de estados válidos o inválidos para las condiciones de entrada.

Estrategia de prueba

Según Pressman una estrategia de prueba del *software*: integra los métodos de diseño de caso de pruebas en una serie bien planeada de pasos que desembocará en la eficaz construcción del *software*. La estrategia proporciona un mapa que describe los pasos, además de cuánto esfuerzo, tiempo y recursos consumirán (Pressman, 2010).

Durante el proceso de desarrollo del módulo se llevaron a cabo diferentes pruebas mediante el uso del método de caja negra, proyectando resultados visibles. Se emplearon además, pruebas de estrés y de aceptación, las que se describen más adelante. Para los efectos de esta investigación se propone la siguiente estrategia de prueba:

Tabla 9 Estrategia de prueba aplicada a la solución

Nivel de prueba	Tipo de prueba	Método de prueba	Técnica de prueba
Sistema	Funcional	Caja negra	Particiones de equivalencia

4.7 Aplicación y resultado de las pruebas

Pruebas de integración

Como en la investigación la solución informática que se propone es un módulo que forma parte del Sistema de Información del Centro para el control estatal de medicamentos, equipos y dispositivos médicos , es necesario probar la integración entre las agrupaciones funcionales, además su integración con los módulos de los que se obtiene información.

Prueba de carga

Mediante esta prueba se ve la reacción del sistema con picos de mayor volumen en un tiempo limitado, puede ser en situaciones que nunca ocurran o que puedan eventualmente ocurrir. Para realizar las pruebas de carga se utilizó la herramienta *Jmeter* que arrojó valiosos datos sobre el rendimiento del sistema.

En la siguiente figura se hace referencia al tiempo que demora el sistema en responder a la simulación de cinco peticiones realizadas a la funcionalidad de registrar una solicitud sin reservación.

# Muestras	Media	Mediana	90% Line	Min	Max	% Error	Rendimiento	Kb/sec
20	1163	1055	1531	808	1587	0,00%	8,1	44,6
20	1345	1253	1773	798	2227	0,00%	6,7	36,8
40	1319	1280	1540	897	2090	0,00%	7,8	42,8
100	1319	1320	1661	601	1872	0,00%	10,3	56,4
120	1205	1179	1624	437	2133	0,00%	11,3	62,3
300	1265	1260	1652	437	2227	0,00%	14,9	81,7

Fig. No 24 Tabla de datos ofrecida por la herramienta Jmeter

Análisis de resultados

Los resultados de la figura 24 se pueden interpretar de la siguiente forma:

- ✓ #Muestras: cantidad de hilos utilizados para la URL.
- ✓ Media: tiempo promedio en milisegundos para un conjunto de resultados.
- ✓ *Mediana:* valor en tiempo del percentil 50.
- ✓ *Línea de 90%:* máximo tiempo utilizado por el 90% de la muestra.
- ✓ *Min:* tiempo mínimo de la muestra de una determinada URL.
- ✓ *Max:* tiempo máximo de la muestra de una determinada URL.
- ✓ *%Error:* porcentaje de requerimientos con errores.
- ✓ *Rendimiento:* rendimiento medido en los requerimiento por segundo / minuto / hora.
- ✓ *Kb/sec:* rendimiento medido en kilobytes por segundo. El tiempo es en milisegundos.

Se obtuvo el tiempo promedio para acceder a una página mediante el siguiente cálculo:

Tiempo Total= #Muestra * Media.

Dio como resultado 1,265 segundos, realizándose un total de cincuenta requerimientos al servidor. El tiempo promedio total requerido por cada hilo se alcanzó mediante la fórmula:

((Tiempo Total/1000)/60)/cantidad de hilos.

El mismo fue de 0,02108 minutos. Se puede asumir que el resultado obtenido es satisfactorio, lo que significa que el sistema es capaz de soportar mucho más carga de la que se deberá enfrentar en el peor de los casos, ya que será usado por un pequeño número de usuarios.

4.8 Casos de prueba

Un caso de prueba es una serie de pruebas de entrada, condiciones de ejecución y resultados esperados desarrollados para un objetivo en particular, tal como ejecutar una ruta particular de un programa o verificar el cumplimiento con un requerimiento en específico.

Un caso de prueba bien diseñado tiene gran posibilidad de llegar a resultados más fiables y eficientes, mejorar el rendimiento del sistema, y reducir los costos en tres categorías: productividad (menos tiempo para escribir y mantener los casos), capacidad de prueba (menos tiempo para ejecutarlos) programar la fiabilidad (estimaciones más fiables y efectivas) (O. 2010).

Los casos de prueba de la caja negra pretenden demostrar que:

- ✓ Las funciones del módulo son operativas.
- ✓ La entrada se acepta de forma correcta.
- ✓ Se produce una salida correcta.

Tabla 10 Caso de prueba Crear agenda

Crear Agenda				
Escenario	**Entrada**	**Resultados esperados**	**Resultados obtenidos**	**Condiciones**
Campo servicios vacío	El actor deja vacío el campo y presiona el botón agregar frecuencia de la página de crear agenda	Debe salir la notificación “Debe seleccionar al menos un servicio.”	El sistema muestra el error satisfactoriamente.	El campo debe estar vacío
Tabla frecuencia vacía	El actor deja vacía la tabla de frecuencias y presiona el botón	Debe salir la notificación “No hay frecuencias seleccionadas para esta agenda. Por	El sistema muestra la notificación satisfactoriamente.	Debe estar vacía la tabla de frecuencias.

	aceptar de la página de crear agenda	favor seleccione alguna.”		
Campo frecuencia con valor Semanal	El actor no selecciona ninguna casilla de selección de días de la semana y presiona el botón agregar frecuencia de la página de crear agenda	Debe salir la notificación “Asegúrese de marcar al menos un día de la semana”	El sistema muestra la notificación satisfactoriamente.	Campo frecuencia con valor Semanal y campo de servicios con al menos un servicio seleccionado
Agregar frecuencia	El actor presiona el botón agregar frecuencia de la página de crear agenda.	Debe salir la notificación “Este día ya se encuentra seleccionado” y mostrar una lista de los días que se encuentren ya agregados	El sistema muestra la notificación satisfactoriamente.	Debe estar ya el día o los días agregados en la tabla frecuencias.
Período agendable	El actor entra las fechas en el campo Desde y Hasta	Debe salir una notificación “El periodo agendable pertenece a otra agenda. Especifique otro”	El sistema muestra la notificación satisfactoriamente.	Una de las fechas del rango agendable debe existir en otra agenda guardada en BD.

Conclusiones parciales

Tras haber realizado el capítulo se puede concluir que los estándares de código propuestos facilitan el trabajo del programador a la hora de implementar el módulo. La propuesta de solución estará orientada a objetos debido al uso de las técnicas de programación y herramientas utilizadas. Además que la estrategia de prueba propuesta posibilitó la detección de errores tanto del código como de la interfaz del módulo, contribuyendo a elevar la calidad de la propuesta de solución.

CONCLUSIONES

Una vez finalizada la implementación del módulo Agenda, el autor considera que se cumplieron los objetivos trazados concluyendo que:

- ✓ Mediante el uso de las tecnologías y lenguaje de programación seleccionados para el desarrollo de la aplicación, se obtuvo un producto con un diseño simple e intuitivo que responde a las necesidades alcanzadas a través de la captura de los requisitos funcionales y no funcionales. La organización de la información ofrece al usuario final una fácil interacción con el sistema y se ajusta a la definición de los requerimientos no funcionales.
- ✓ El estudio de los módulos existentes, demuestra que los mismos no brindan solución al problema de la investigación, por lo que fue necesario desarrollar un módulo que se ajustara al negocio estudiado.
- ✓ La implementación del componente Agenda proporcionó una herramienta que evidenció la contribución realizada a la gestión de las agendas del departamento de Recepción y Preevaluación de trámites, así como las pruebas realizadas a esta determinaron la robustez y eficacia del sistema diseñado.

RECOMENDACIONES

Con los resultados obtenidos en la investigación y la experiencia adquirida en el desarrollo de la aplicación, se recomienda:

- ✓ El sistema final podrá ser utilizado en otras entidades que tengan características o brinden un servicio similar al del sistema de Información del CECMED, colaborando de esta manera a la informatización del proceso de gestión de agendas.
- ✓ El especialista pueda visualizar los turnos y citas reservadas en su agenda sin realizar modificación en esta.
- ✓ Cuando el especialista superior transfiera un turno, se le haga llegar una notificación por correo electrónico al cliente.

REFERENCIAS BIBLIOGRÁFICAS

Chamorro Mínguez, José Manuel. 2011. *Aplicación Web para la gestión y planificación de competiciones de tenis.* Leganés : s.n., 2011.

Cibercuba. Cibercuba. [En línea] [Citado el: 2 de Marzo de 2016.] https://www.cibercuba.com/noticias/2016-05-28-u2397-galen-clinica-aplicacion-informatica-desarrollada-en-cuba-para-ayudar-la.

EcuRed. EcuRed. [En línea] [Citado el: 8 de Marzo de 2016.] http://www.ecured.cu/ArgoUML.

Gimson, Loraine. 2012. *Metodologías ágiles y desarrollo basado en conocimiento .* 2012.

Gusgus. Wikipedia. [En línea] [Citado el: 10 de Marzo de 2016.] https://es.wikipedia.org/wiki/Agenda_personal.

Hernández González, Anaissa . 2003. *na Herramienta Cases para el Diseño y la Generacion de la Estructura Estática de La Base de datos. Revista Investigación Operacional. Centro de Estudios de Ingeniería y Sistemas (CEIS).* 2003.

IBM. IBM. [En línea] [Citado el: 3 de Marzo de 2016.] http://www.ibm.com/support/knowledgecenter/es/SSWSR9_11.0.0/com.ibm.pim.app.doc/code/java/pim_con_javasoldev.html.

Instituto de Investigaciones Biológicas Clemente Estable. 2013. *Manual básico para usuarios de la interfase web del servicio de correo electrónico en el IIBCE (Zimbra Collaboration Suite 8).* Uruguay : s.n., 2013. 2.

Jkbw. Wikipedia. [En línea] [Citado el: 10 de Marzo de 2016.] https://es.wikipedia.org/wiki/M%C3%B3dulo_%28inform%C3%A1tica%29.

Kruchten, P. *The Rational Unified Process An Introduction.*

Microsoft. 2007. Support Office. [En línea] 2007. [Citado el: 1 de Marzo de 2006.] https://support.office.com/es-es/article/Introducci%C3%B3n-al-Calendario-de-Outlook-6fdbea97-7315-4de7-b9a3-9a50429971e3.

Pérez García, Alejandro. 2006. *METODOLOGÍAS PARA EL DESARROLLO DE SISTEMAS. CASOS DE ESTUDIO: MÉTRICA II Y MERISE .* México : s.n., 2006.

Quatrani, Terry . 2002. *Visual Modeling with Rational Rose 2002 and UML.* 2002.

Rodriguez, A. 2013. *Lenguaje de Programación.* 2013.

Rumbaugh, James y Booch, Grady. 2007. *El Lenguaje Unificado de Modelado A.* 2007.

Tsang y C. H. K. 2005. *Object-Oriented Technology. Pittsford.* NY : s.n., 2005.

Vaswani, V. 2010. *Fundamentos de PHP.* 2010.

Z, Cataldi. 2000. *Metodología de diseño, desarrollo y evaluación de software educativo. Tesis de Magíster en Informática. (Versión resumida). Facultad de Informática. Universidad Nacional de La Plata.* Argentina : s.n., 2000. ISBN 960-34-0204-2.