Universidad de las Ciencias Informáticas

Facultad 2

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO

DE INGENIERO EN CIENCIAS INFORMÁTICAS

Restructuración de la configuración del módulo de Laboratorio del Sistema de Información Hospitalaria del CESIM.

Autor: Dayan Crespo Naranjo

Tutor: Ing. Francisco Rodríguez Torres

Co-Tutor: Ing. Yasnaya Ferras Solórzano

La Habana, Junio 2014

"Año 56 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los 18 días del mes de junio del año 2014:

_____

Dayan Crespo Naranjo

Autor

_____ _____

Ing. Yasnaya Ferras Solorzano

Tutor

Ing. Francisco Rodríguez Torres

Tutor

Datos de Contactos

Ingeniero Francisco Rodríguez Torres: graduado en la Universidad de las Ciencias Informáticas (UCI) en el 2009. Posee la categoría docente de Instructor. Imparte la asignatura de Bases de datos I y II. Pertenece al Departamento Sistemas de Gestión Hospitalaria del centro CESIM. Es el jefe de módulo de Laboratorio del Sistema de Información Hospitalaria del centro CESIM.

Correo electrónico: frtorres@uci.cu

Ingeniero Yasnaya Ferras Solorzano: graduada en la Universidad de las Ciencias Informáticas (UCI) en el 2012. Pertenece al Departamento Sistemas de Gestión Hospitalaria del centro CESIM. Desempeña el rol de analista en el módulo de Epidemiología del Sistema de Información Hospitalaria del centro CESIM.

Correo electrónico: ysolorzano@uci.cu

AGRADECIMIENTOS

Le quiero agradecer primero que todo a mis padres por haberme traído a la vida y darme todo lo que tengo y apoyarme en todo. También agradecer a mi tutor paterno por apoyarme igualmente. A mi familia en general por estar al tanto de todo lo que necesitaba para seguir por el mejor camino.

A mis tutores por guiarme para llegar hasta aquí, les estaré eternamente agradecido por todo lo que han hecho por mí en estos 10 meses de duro trabajo y sacrificio.

Agradecerle a mis amistades a las que han estado cerca y a las que no también, por animarme cuando estaba súper que estresado y saber suavizar el peso de la responsabilidad de mi tesis por unos momentos.

También quiero agradecerle a la Universidad de las Ciencias Informáticas por darme todo lo que estaba a su disposición para cumplir y terminar con mi tesis de título de Ingeniero Ciencias Informáticas.

DEDICATORIA

Mi tesis se la quiero dedicar a mi familia, a mis padres, a mis hermanos y a mi familia en general. También me la quisiera dedicar a mí mismo como recompensa de todo lo bueno que he hecho y para demostrar de que soy capaz de hacer grandes cosas si me lo propongo, aunque sea con mucho trabajo.

RESUMEN

Un Sistema de Información Hospitalaria (Hospital Information System, HIS) centraliza toda la información generada por los distintos servicios del hospital a partir de un mismo paciente. Los resultados generados por los laboratorios clínicos son generalmente plasmados en algún tipo de soporte físico y transmitido manualmente al especialista que los precisa para el diagnóstico. Un Sistema de Información de Laboratorio (Laboratory Information System, LIS) es una tipo de software que recibe, procesa, almacena y gestiona la información generada por los procesos del laboratorio clínico. Estos sistemas deben interactuar con los equipos médicos y otros sistemas de información.

La Universidad de las Ciencias Informáticas tiene un Sistema de Información Hospitalaria que está compuesto por 17 módulos y uno de los módulos es el de Laboratorio. Para el desarrollo de dicho módulo se utilizaron como base de conocimientos los Sistemas de Información de Laboratorio. Este módulo nos brinda la posibilidad de gestionar el flujo de información procesada en un laboratorio clínico de un hospital. Se encuentra divido en dos partes fundamentales: el flujo básico y la configuración. En el flujo básico se realizan todas las actividades del personal del laboratorio clínico, algunas de ellas son la recepción, clasificación y procesamiento de muestras y la entrega de resultados.

El presente trabajo tiene como objetivo: desarrollar la nueva estructura de la configuración del módulo Laboratorio del Sistema de Información Hospitalaria del CESIM que facilite la gestión de los elementos configurables del mismo.

Palabras Claves: configuración, flujo básico, Sistema de Información de Laboratorio, Sistema de Información Hospitalaria.

Índice

Índice de Tablas

INTRODUCCIÓN

INTRODUCCIÓN

Un Sistema de Información Hospitalaria (Hospital Information System, HIS) tiene como propósito la recolección, almacenamiento, procesamiento, interpretación y comunicación de la información de los pacientes.

La Universidad de las Ciencias Informáticas cuenta con el Centro de Informática Médica (CESIM). Este desarrolla un Sistema de Información Hospitalaria que está compuesto por 17 módulos. Estos interconectan las distintas áreas de un hospital como son: Consulta externa, Admisión, Emergencias y Laboratorio. El módulo de Laboratorio brinda la posibilidad de gestionar el flujo de información procesada en un laboratorio clínico de un hospital. Para el desarrollo del mismo se utilizaron como base de conocimientos los Sistemas de Información de Laboratorio (Laboratory Information System, LIS). Un LIS es un tipo de herramienta que recibe, procesa y mantiene la información que se genera en un laboratorio clínico o de investigación. Informan resultados de pruebas analíticas realizadas en muestras procedentes de un paciente con un fin determinado. En el apartado asistencial ayudan en los aspectos preanalíticos (solicitud, cita, obtención de muestras, preparación, transporte y distribución), en los aspectos analíticos (procesamiento de muestras y gestión de equipos) y en los aspectos postanalíticos (edición de informes, distribución, archivos de muestras). (1)

El módulo de Laboratorio del Sistema de Información Hospitalaria del CESIM responde a las exigencias que se presentan durante la gestión de procesos en esta área. Estos procesos gestionan la información necesaria para el correcto funcionamiento del área; estos son la recepción, clasificación y procesamiento de muestras y la entrega de resultados. El módulo se encuentra dividido en dos partes fundamentales: el flujo básico y la configuración. El flujo básico son las diferentes acciones que implican los procesos anteriormente mencionados, estas acciones son: las solicitudes de análisis, resultados de los exámenes y de esta misma forma el registro de las muestras iniciales de los análisis solicitados. El personal del laboratorio clínico utiliza varios elementos en el flujo básico. Entre estos se encuentran las muestras, resultados, exámenes, secciones, pruebas y las solicitudes. Para lograr que los elementos estén listos para su uso en el flujo básico, se cuenta con el proceso de configuración. Es responsable de la gestión y relación de cada uno de estos elementos. La configuración permite que el personal del laboratorio mantenga la misma nomenclatura que utilizaba antes de usar el sistema. Los resultados de las pruebas generadas de forma automática utilizan la misma nomenclatura configurada en el módulo. Abstrae a los

bioanalistas de los nombres poco intuitivos de las pruebas automatizadas. Dichas pruebas son los diversos análisis que contienen o conforman a los exámenes.

El flujo básico del módulo no puede ser utilizado por el personal hasta que no estén gestionados todos los elementos a utilizar. Como regla inviolable los elementos utilizados en el flujo básico no pueden ser eliminados, ni modificados. Los 8 elementos a configurar: las Secciones, los aspectos de solicitud y grupos de aspectos de solicitud asociados a las secciones, los Exámenes , los aspectos de resultados y grupos de aspectos de resultados de los exámenes, los tipos de datos que poseen los aspectos y las unidades de medida de los tipos de datos; tienen una estrecha relación entre sí. Cada uno en el proceso de configuración del módulo se gestiona por separado y existen funcionalidades integradoras que se encargan de relacionarlos. Las funcionalidades que integran no pueden ser utilizadas hasta que no estén configurados todos los elementos previamente. Los errores en los elementos relacionados en la configuración actual no pueden ser corregidos aunque no estén siendo utilizados en el flujo básico. El desacople de las funcionalidades de la configuración provoca que el 90 % de los errores en los elementos se identifiquen en el flujo básico cuando ya están en uso. Debido a que estos errores no pueden ser corregidos se genera una información basura que no puede ser eliminada.

Por lo antes planteado se define como **problema de la investigación**: la estructura actual de la configuración del módulo Laboratorio del Sistema de Información Hospitalaria del CESIM dificulta la gestión de sus elementos configurables, provocando la generación de errores no corregibles e información innecesaria.

El **objeto de estudio** está delimitado por los procesos asociados al funcionamiento de los laboratorios en las instituciones hospitalarias. Por lo anteriormente citado el **campo de acción** queda expresado de la siguiente forma: procesos asociados a la configuración del módulo Laboratorio del Sistema de Información Hospitalaria del CESIM. Para darle solución al problema, se define como **objetivo general**: desarrollar la nueva estructura de la configuración del módulo Laboratorio del Sistema de Información Hospitalaria del CESIM que facilite la gestión de los elementos configurables del mismo.

Tareas de la investigación:

1. Analizar el proceso de configuración del módulo de Laboratorio del Sistema de Información Hospitalaria del CESIM.

2. Asimilar la arquitectura definida por el Departamento de Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones.
3. Generar los artefactos correspondientes a las fases: “Modelado de negocio”, “Diseño” e “Implementación”.
4. Implementar la nueva estructura de configuración manteniendo los elementos configurables que presenta el módulo y lograr una mejor integración entre ellos.

   El desarrollo de la restructuración del módulo de Laboratorio del Sistema de Información Hospitalaria, proporcionará un grupo de beneficios que se muestran a continuación:

   - Se realizará una gestión y relación completa de los exámenes y secciones del módulo.
   - Se disminuirán los errores en las funcionalidades de gestión de elementos en la configuración del módulo.

El presente documento se encuentra estructurado en cuatro capítulos, el primero de ellos, **“FUNDAMENTACIÓN TEÓRICA”**, brinda al lector el estado del arte relacionado con los sistemas de información hospitalaria y lo ubica en el ambiente de desarrollo del módulo de Laboratorio como parte del Sistema de Información Hospitalaria del CESIM. Justificándose las tendencias, tecnologías, metodologías y herramientas que fueron utilizadas para el desarrollo del mismo. Seguidamente el capítulo, **“CARACTERÍSTICAS DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO”**, contiene un marco conceptual asociado a la información que será manipulada por el sistema, llegándose a un acuerdo sobre las funcionalidades, requerimientos deseados y el objeto de automatización.

El tercer capítulo **“ANÁLISIS Y DISEÑO DE LA RESTRUCTURACIÓN DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO”** se centra en la modelación detallada y la construcción de la estructura de la aplicación. En el cuarto y último, **“IMPLEMENTACIÓN DE LA RESTRUCTURACIÓN DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO”**, se implementan las clases y procesos en término de módulo. Se presenta la propuesta de solución para lograr una gestión más eficiente de los procesos y requerimientos pertenecientes al módulo de Laboratorio del Sistema de Información Hospitalaria del CESIM.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

El Sistema Nacional de Salud puede definirse como pequeñas estructuras administrativas designadas a ciertos objetivos específicos en la salud. Este incluye la totalidad de los elementos o componentes del sistema social que se relacionan, en forma directa o indirecta con la salud de la población. Así mismo este está integrado por el conjunto de instituciones y organizaciones que forman parte del sector público. Tiene como finalidad, directa o indirecta, contribuir a mejorar la salud de las personas, las familias y las comunidades, garantizar a toda la población servicios integrales de buena calidad. También dispone de estrategias y programas que permiten la participación de la comunidad en la determinación de necesidades y prioridades, así como en la orientación de los recursos. Establece subsistemas administrativos, eficientes y equitativos, y garantiza por niveles de gestión (central, regional y local) una legislación nacional congruente y organizada. (2)

1.1 Sistemas Automatizados vinculados al campo de acción

En la actualidad se han desarrollado soluciones para resolver los problemas existentes en las instituciones hospitalarias. Los Sistemas de Información de Laboratorio constituyen una herramienta para la actividad de los laboratorios clínicos, es un conjunto de hardware y software que da soporte al personal de un laboratorio. La configuración de los LIS es limitada porque como la mayoría de los sistemas son propietarios y dicha configuración es poco entendible, a los usuarios se les torna engorroso el trabajo. Los autores de estas soluciones son diversos pero todos han hecho el esfuerzo por tratar de lograr la completa automatización de los procesos de gestión hospitalaria. En particular en el área del laboratorio clínico se ha avanzado bastante a nivel mundial.

Entre los sistemas que se han desarrollado están los siguientes LabWare LIMS, Wiener Lab Group, GalenLab, entre otros.

1.1.1 LabWare LIMS

LabWare LIMS es un sistema de información de laboratorio (LIMS) con acceso equivalente cliente/servidor, que se integra sin costuras en cualquier entorno informático gracias a su completa funcionalidad para seguimiento de muestras, certificación de usuarios, gestión de instrumentos, auditoría, y planificación de muestras e informes, así como muchas otras funciones. Como base de datos, se pueden usar todos los productos comerciales, desde ORACLE™ o SQL Server™ hasta DB2™. El sistema no obliga a una versión particular de estos productos. Permite mejorar significativamente la calidad de los

servicios informáticos de su plataforma, disminuyendo al mismo tiempo los costos mensuales de soporte y mantención de su laboratorio. Este software posee beneficios notables porque automatiza y unifica los procesos, así como incrementa la rapidez de los procesos del laboratorio. (3)

1.1.2 Wiener Lab Group

Este LIS es un sistema informático modular, que busca satisfacer los procesos operativos de medianos y grandes laboratorios, respetando los estándares de calidad de la información. Es una herramienta de gestión, que facilita el acceso a la información del laboratorio en forma rápida y simple. Posee una interface amigable para el usuario final, presente una gestión integral del laboratorio, historia clínica en línea. Posee funcionalidades como Tracking (seguimiento) de muestras, Administración de turnos, Carga y Validación de resultados y Recitación de pacientes. (4)

1.1.3 GalenLab

GalenLab es una de las soluciones implementadas por la empresa cubana de desarrollo de software Softel para automatizar los procesos en el área de laboratorio clínico. Está dirigido a la gestión de los medios de diagnóstico, facilitando la solicitud de exámenes, registro y evaluación de los resultados obtenidos, así como la generación de información estadística. Diseñado para ser utilizado por los técnicos, médicos y personal administrativo de medios de diagnóstico para optimizar el trabajo y elevar la eficiencia. Es una aplicación desarrollada en visual Basic, funciona solamente en las estaciones de trabajo que tengan el sistema operativo Windows, utiliza como gestor de base datos SQL; y además para cada estación de trabajo donde se vaya a instalar hay que pagarle a la empresa cubana Softel. (5)

Aunque fueron base de conocimiento dichos sistemas para la creación del módulo de Laboratorio, la configuración de los mismos no satisface los requerimientos del cliente o del Sistema de Información Hospitalaria del CESIM. Estos sistemas requieren de métodos de respaldo y solo son proporcionados por las compañías desarrolladoras de los mismos a un alto costo. Por dichas razones se tomó la decisión de desarrollar una nueva estructura de la configuración de módulo de Laboratorio.

1.2 Tecnologías utilizadas en el proceso de desarrollo.

Las tecnologías utilizadas para dar solución al problema planteado son las definidas por el proyecto de Sistema de Gestión Hospitalaria del CESIM para la elaboración de soluciones informáticas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

La arquitectura en capas tiene como objetivo primordial la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

- **Capa de presentación:** Reúne todos los aspectos del software que tiene que ver con las interfaces y la interacción con los diferentes tipos de usuarios humanos. Estos aspectos típicamente incluyen el manejo y aspecto de las ventanas, el formato de los reportes, gráficos y elementos multimedia en general. (6)
- **Capa de negocio:** Agrupa los aspectos del software que tienen que automatizar o apoyar los procesos de negocio que llevan a cabo los usuarios. Estos aspectos típicamente incluyen las tareas que forman parte de los procesos, las reglas y restricciones que aplican. Esta capa también recibe el nombre de la capa de la Lógica de la Aplicación. (6)
- **Capa de datos:** Contiene los aspectos del software que tienen que ver con el manejo de los datos persistentes, por lo que también se le denomina capa de las Bases de Datos. (6)

1.2.1 JavaScript

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación. Se usa principalmente en páginas web, con una sintaxis semejante a la del lenguaje de programación Java y C.

Es un lenguaje orientado a objetos, ya que dispone de herencia y puede ser interpretado por la mayoría de los navegadores. Tiene la ventaja de ser incorporado en cualquier página web, y puede ser ejecutado sin la necesidad de instalar otro programa para ser visualizado. (7)

1.2.2 XML (Extensible Markup Language)

XML es un metalenguaje extensible de etiquetas. Permite definir la gramática de lenguajes específicos. Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

1.2.3 AJAX

Ajax (JavaScript asíncrono y XML) es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad,

velocidad y usabilidad en las aplicaciones. Es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. (8)

1.2.4 Java

Java es un lenguaje de programación que ofrece la potencia del diseño orientado a objetos con una sintaxis fácilmente accesible y un entorno robusto y agradable. Es distribuido, multiplataforma, compilado, seguro y posee una arquitectura neutral. Proporciona un conjunto de clases potente y flexible. Este lenguaje reduce en un 50% los errores más comunes de programación con lenguajes como C y C++ al eliminar muchas de las características de éstos, entre las que destacan: aritmética de punteros, no existen referencias, registros, definición de tipos, macros, necesidad de liberar memoria, entre otras. Soporta sincronización de múltiples hilos de ejecución a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. (7)

1.2.5 POJOs (Plain Old Java Object)

POJO es el acrónimo de Plain Old Java Object son las siglas utilizadas por programadores java para enfatizar el uso de clases simples y que no dependen de un framework en especial.

1.2.6 JBoss Seam como Framework para desarrollo de aplicaciones Web

JBoss Seam es un framework open source que integra JavaScript Asíncrono y XML (Ajax), Businnes Process Management (BPM), la capa de presentación (JSF) con la capa de negocios y persistencia (EJB). Fue desarrollado con el fin de unir diferentes tecnologías y estándares de Java en un solo framework, a la vez que añade algunas funcionalidades no contempladas por ellos. Además de otras características y ventajas permite eliminar la complejidad de la arquitectura y los niveles del API, ensamblar aplicaciones web complejas con simples anotaciones POJOs sin importar el tipo de componente. Las aplicaciones Seam son conceptualmente simples y requieren significativamente menos código (en Java y en XML) para obtener las mismas funcionalidades. Añade herramientas útiles para el desarrollo de aplicaciones web y está basado en estándares ampliamente utilizados y probados (escalables, portables y reusables). (9)

1.2.7 RichFaces 3.2 como librería de componentes Web

RichFaces es una rica biblioteca de componentes para JSF y un avanzado marco o framework para integrar fácilmente capacidades AJAX, en el desarrollo de aplicaciones. Los componentes RichFaces vienen listos para su uso, por lo que los desarrolladores pueden aprovechar de inmediato las

características de los componentes para crear aplicaciones web más fiables y rápidas, contribuyendo a la mejora de la experiencia del usuario. Esta incluye un fuerte apoyo para el cambio de temas de aplicaciones JSF; también aprovecha al máximo los beneficios de EJB incluyendo el ciclo de vida, la validación y las facilidades de conversión, junto con la gestión de los recursos estáticos y dinámicos.

1.2.8 Framework Hibernate

Hibernate es una capa de persistencia objeto/relacional y un generador de sentencias SQL. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. Permite expresar consultas en una extensión de SQL (HQL), así como en SQL nativo o utilizando criterios orientado a objetos. Brinda filtros para trabajar con datos históricos, regionales o condicionados por permisos. Puede ser usado para desarrollar y distribuir aplicaciones de forma gratuita.

1.2.9 PostgreSQL

PostgreSQL 9.1.1 es un sistema de gestión de bases de datos objeto-relacionales perteneciente al ámbito del software libre que destaca por su robustez, escalabilidad y cumplimiento de los estándares SQL y, en algunos aspectos, está diseñado para que sea extensible por los usuarios. Cuenta con versiones para una amplia gama de sistemas operativos, entre ellos: Linux, Windows, Mac OS X, Solaris y otros más. Este gestor permite la realización de transacciones seguras (ACID), vistas, uniones, claves extranjeras, procedimientos almacenados, copias de seguridad en línea, replicación asíncrona, transacciones anidadas, optimizador de consultas.

1.2.10 Servidor de Aplicaciones JBoss

JBoss es un servidor de aplicaciones Java 2 Platform Enterprise Edition (J2EE) de código abierto, implementado en Java puro sin coste adicional. JBoss implementa todo el paquete de servicios de J2EE. Al estar basado en Java, puede ser utilizado en cualquier sistema operativo que lo soporte. Es el primer servidor de aplicaciones de código abierto, preparado para la producción siendo muy confiable a nivel de empresa, ofreciendo una plataforma de alto rendimiento para aplicaciones javas y aplicaciones web, combinando una arquitectura incrustable y orientada a servicios. Dicho servidor brinda también una flexibilidad consistente y un soporte completo para Java Management Extensions (JMX).

1.2.11 Proceso Unificado de Desarrollo (Rational Unified Process, RUP)

Es un proceso que define como realizar de forma correcta el desarrollo de software. Trae integrado los mejores elementos de las metodologías anteriores. Es una plataforma elaborada de ingeniería de

procesos que permite a los equipos definir, configurar, personalizar y practicar un proceso coherente. RUP es una recopilación de prácticas de ingeniería de software que se están mejorando continuamente de forma regular para reflejar los cambios en las prácticas de la industria. RUP proporciona una buena base de arquitectura y una gran cantidad de material con las que construir una definición de proceso, lo que permite configurar y ampliar dicha base como desee; esto ahorrará mucho tiempo y esfuerzo que de otra manera tendría que aplicar para crear dicha definición de proceso desde cero. (10)

1.2.12 Lenguaje Unificado de Modelado (Unified Modeling Language, UML)

Lenguaje Unificado de Modelado es un conjunto de herramientas que permiten modelar (analizar y diseñar) sistemas orientados a objetos. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Brinda la posibilidad de realizar diferentes tipos de diagramas, los cuales agrupados jerárquicamente serían: diagramas de estructura (enfatizan en los elementos que deben existir en el sistema modelado), diagramas de comportamiento (enfatizan en lo que debe suceder en el sistema modelado), y los diagramas de interacción (son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado).

1.3 Herramientas

Con los elementos expuestos anteriormente es posible definir las herramientas que conforman el ambiente de desarrollo del módulo de Laboratorio para el Sistema de Información Hospitalaria del CESIM. Las herramientas utilizadas para dar solución al problema planteado son las definidas por el proyecto de Sistemas de Gestión Hospitalaria del CESIM para la elaboración de soluciones informáticas.

Como herramienta CASE (Computer-Aided Software Engineering) se utiliza Visual Paradigm 8.0, esta es una herramienta UML profesional que permite desarrollar rápido y con calidad; además soporta la versión 2.1 de UML y permite la incorporación de nuevas anotaciones así como de nuevas formas y símbolos.

Como gestor de bases de datos se utilizó PostgreSQL 9.1.1, se hace uso de pgAdmin III, esta herramienta es una aplicación gráfica, siendo la más completa y popular con licencia open source.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Se seleccionó JbossDeveloper Studio para crear entornos integrados de desarrollo (IDE). Este IDE permite la agregación de nuevas funcionalidades al editor, por medio de nuevos módulos ('plugins'). Esto posibilita el empleo de JBoss Tools que es una colección de plugins, añadidos al IDE para proveer de una serie de funciones y facilidades que abstraen al usuario del funcionamiento interno. Posee un editor gráfico para la configuración de archivos Seam. Soporta la realización de pruebas de integración de Seam desde el Eclipse.

CAPÍTULO 2. CARACTERÍSTICAS DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

En cada uno de los procesos que se realizan en el módulo de Laboratorio se maneja un cúmulo de información y se realizan las actividades para garantizar el cumplimiento de los objetivos planteados. Recibir y clasificar muestra como proceso, tiene como objetivo recibir, preparar y enviar las muestras a cada una de las secciones que se encargarán de su análisis. El proceso comienza cuando el paciente arriba al área de recepción y clasificación de muestras, luego de finalizada la extracción por parte de los bioanalistas (también puede ser el ayudante del área), con la solicitud de análisis, que contiene entre otros datos los análisis solicitados, a quién corresponden los análisis y la persona que los solicitó; la boleta de reporte de los resultados que contiene los datos del paciente y además trae las muestras a recibir y clasificar. El proceso de clasificar muestra tiene como objetivo procesar las muestras enviadas a cada una de las secciones del laboratorio y emitir los resultados correspondientes. El proceso se realiza en cada una de las secciones del área de laboratorio. Entregar resultado es otro de los procesos que se encarga de gestionar la entrega de los resultados recibidos en el área de cada una de las secciones encargadas del procesamiento y obtención de los mismos.

La estrecha relación entre los elementos de los procesos antes mencionados es de gran importancia en la configuración del módulo de Laboratorio. Las Observaciones (antes aspectos de solicitud) que se encuentran agrupadas en los grupos de Observaciones, representan los apuntes del médico en las secciones que trabajan. Las Pruebas (antes aspectos de resultados) son los análisis que se realizan en los exámenes del laboratorio clínico. La configuración es la encargada de tener estos elementos relacionados y listos para su uso en el flujo básico del módulo de Laboratorio.

Con motivo de la poca estructuración de los procesos del negocio y para poder comprender el contexto en el cual se desarrolla el sistema, se muestra el Modelo de Dominio, donde se exponen conceptos relacionados con el módulo de Laboratorio, su configuración y los vínculos entre estas definiciones. Por otra parte, se enumeran los requerimientos funcionales y no funcionales, agrupándose los primeros en casos de uso, con el objetivo de estructurar el Diagrama de Casos de Uso del módulo.

CAPÍTULO 2. CARACTERÍSTICAS DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

2.1 Modelo de Dominio

El Modelo de Dominio o Modelo Conceptual es una representación visual de los conceptos u objetos que se manejan en el dominio del sistema. Los objetos o conceptos incluidos en el Modelo de Dominio no describen clases u objetos del software; sino entidades o conceptos del mundo real que están asociadas al problema en cuestión. Dicho modelo podrá ser utilizado como una base de las abstracciones relevantes en el proceso de reconstrucción del sistema.

Se proporciona una breve descripción de los conceptos encontrados en el ámbito del problema, con el fin de facilitar una mejor comprensión del Diagrama del Modelo de Dominio, recordando que dichos conceptos estarán, en gran escala, asociados a los conceptos actuales existentes en el proceso de configuración del módulo de Laboratorio. Estos son:

Sección: permite realizar las operaciones básicas sobre las secciones que se desean configurar en el Laboratorio.

Examen: la interfaz permite realizar las operaciones básicas sobre los exámenes que se desean configurar en el Laboratorio.

Tipo de muestra: permite realizar las operaciones básicas sobre los tipos de muestra.

Unidad de Medida: permite gestionar las distintas unidades de medida a utilizar en el Laboratorio.

Aspecto de solicitud: permite realizar las operaciones básicas sobre los aspectos de solicitud que se desean configurar en el Laboratorio.

Grupo de aspecto de Solicitud: permite realizar las operaciones básicas sobre los grupos de aspectos de solicitud que se desean configurar en el Laboratorio.

Aspectos de resultados: permite realizar las operaciones básicas sobre los aspectos de resultados que se desean configurar en el Laboratorio.

Grupo de aspecto de resultados: permite realizar las operaciones básicas sobre los grupos de aspectos de resultados que se desean configurar en el Laboratorio.

CAPÍTULO 2. CARACTERÍSTICAS DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

Diagrama del modelo de Dominio

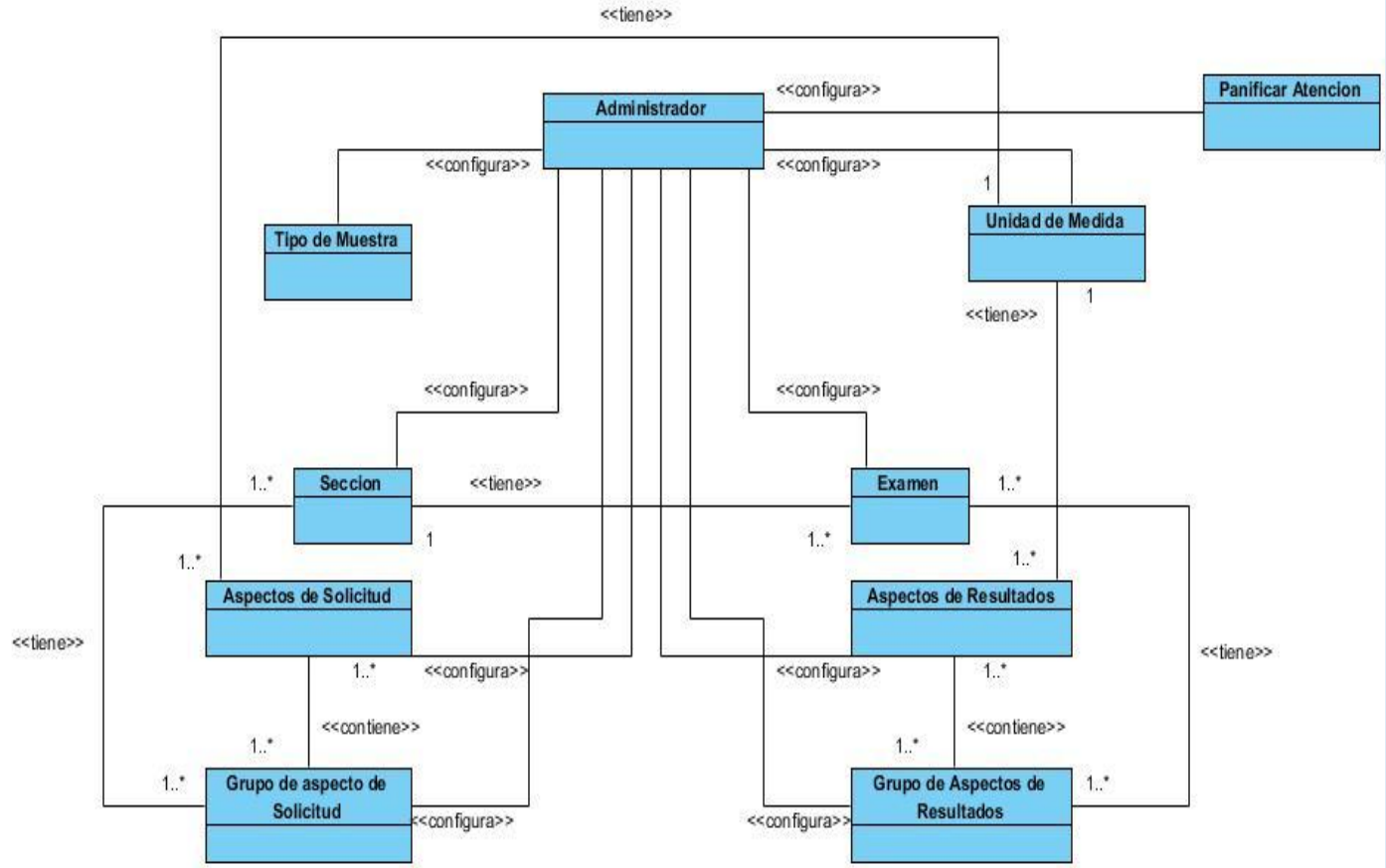


Figura 1. Diagrama de Modelo de Dominio

2.2 Especificación de requerimientos de software

La Ingeniería de requerimientos comprende las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta los diversos requerimientos de los inversores, que pueden entrar en conflicto entre ellos. Tiene como propósito definir y validar los requerimientos con los que debe cumplir el software; así como provee a los desarrolladores del sistema un mejor entendimiento de los requerimientos del sistema.

Una especificación de requerimientos del software es una descripción completa del comportamiento del sistema a desarrollar. Incluye un conjunto de casos de uso que describen todas las interacciones que se prevén que los usuarios tendrán con el software. También contiene requerimientos no funcionales (o

suplementarios). Los requerimientos no funcionales son los requerimientos que imponen restricciones al diseño o funcionamiento del sistema (tal como requerimientos de funcionamiento, estándares de calidad, o requerimientos del diseño).

Los requerimientos se dividen en dos:

- Funcionales: son las funcionalidades que realiza el software.
- No funcionales: son los "recursos" para que trabaje el sistema de información (redes, tecnología).

2.2.1 Requerimientos funcionales

Un Requerimiento funcional define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica. Son complementados por los requerimientos no funcionales, que se enfocan en cambio en el diseño o la implementación. Como se define en la ingeniería de requerimientos, los requerimientos funcionales establecen los comportamientos del sistema.

Teniendo en cuenta las actividades a automatizar es posible definir los siguientes requisitos funcionales:

RF1: Gestionar sección

RF2: Gestionar exámenes

RF3: Crear observación

RF4: Modificar observación

RF5: Eliminar observación

RF6: Crear grupo observación

RF7: Modificar grupo observación

RF8: Eliminar grupo observación

RF9: Crear prueba

RF10: Modificar prueba

RF11: Eliminar prueba

RF12: Crear grupo prueba

RF13: Modificar grupo prueba

RF14: Eliminar grupo prueba

RF15: Crear tipo de dato

RF16: Modificar tipo de dato

RF17: Crear unidad de medida

RF18: Modificar unidad de medida

2.2.2 Requerimientos no funcionales

Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad y estándares. Un requerimiento no funcional es, en la ingeniería de sistemas y la ingeniería de software, un requerimiento que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requerimientos funcionales.

2.2.1.2.1 Eficiencia

El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria. Para ello se potenciará como regla guardar en la memoria caché datos y recursos de alta demanda.

El sistema respetará buenas prácticas de programación para incrementar el rendimiento en operaciones costosas para la máquina virtual como la creación de objetos. Se deberá usar siempre que sea posible el patrón Singleton, destruir referencias que ya no estén siendo usadas, optimizar el trabajo con cadenas, entre otras buenas prácticas que ayudan a mejorar el rendimiento.

2.2.1.2.3 Requerimientos de rendimiento

El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria. El sistema respetará buenas prácticas de programación para incrementar el rendimiento en operaciones costosas para la máquina virtual como la creación de objetos.

2.2.1.2.4 Requisitos de soporte

Se permitirá la creación de usuarios, otorgamiento de privilegios y roles, asignación de perfiles y activación de permisos por direcciones IP.

Se permitirá administración remota, monitoreo del funcionamiento del sistema en los centros hospitalarios y detección de fallas de comunicación.

Se permitirá realizar copias de seguridad de la base de datos hacia otro dispositivo de almacenamiento externo, además de recuperar la base de datos a partir de los respaldos realizados.

Se permitirá el chequeo de las operaciones y acceso de los usuarios al sistema. Se permitirá establecer parámetros de configuración del sistema y actualización de nomencladores.

2.2.1.2.5 Requisitos de hardware

2.2.1.2.5.1 Servidores

- La solución estará conformada, fundamentalmente, por servidores de alta capacidad de procesamiento y redundancia, que permitan garantizar movilidad y residencia de la información y las aplicaciones bajo esquemas seguros y confiables.
- Servidores de Base de datos: Servidor con cualquier tecnología o sistema operativo que soporte a PostgreSQL Server 8.4 o superior en los servidores de base de datos de cada hospital, y Oracle 10g o superior para los servidores de base de datos del centro de datos.
- Servidores de Aplicaciones: Servidor con cualquier tecnología o sistema operativo que soporte el Java Runtime Environment (JRE) 1.6.0_24 o superior y al JBoss AS 4.2.2.GA.

2.2.1.2.5.2 Requisitos de software

El sistema debe correr en sistemas operativos Windows, Unix y Linux, utilizando la plataforma JAVA (Java Virtual Machine), JBoss AS y PostgreSql.

2.2.1.2.5.3 Requisitos de diseño

La capa de presentación contendrá todas las vistas y la lógica de la presentación. El flujo web se manejará de forma declarativa y basándose en definiciones de procesos del negocio.

La capa del negocio mantendrá el estado de las conversaciones y procesos del negocio que concurrentemente pueden estar siendo ejecutados por cada usuario.

La capa de acceso a datos contendrá las entidades y los objetos de acceso a datos correspondientes a las mismas. El acceso a datos está basado en el estándar JPA y particularmente en la implementación del motor de persistencia Hibernate.

2.2.1.2.6 Requisitos de interfaz

2.2.1.2.6.1 Interfaces de usuario

Las ventanas del sistema contendrán claro y bien estructurados los datos, además de permitir la interpretación correcta de la información.

La interfaz contará con teclas de función y menús desplegables que faciliten y aceleren su utilización.

La entrada de datos incorrecta será detectada claramente e informada al usuario.

Todos los textos y mensajes en pantalla aparecerán en idioma español.

2.2.1.2.7 Fiabilidad

En los servidores de los hospitales y en el centro de datos nacional se garantizará una arquitectura de máxima disponibilidad, tanto de servidores de aplicación como de base de datos. El sistema permitirá la implementación de políticas de respaldo a toda la información, evitando pérdidas en caso de desastres ajenos al sistema. Para esta implementación de las políticas de respaldo existen dos variantes:

Variante 1: El respaldo es responsabilidad del cliente. En este caso el equipo de desarrollo debe especificar por escrito al cliente los elementos a respaldar, ejemplo: base de datos, repositorio de documentos clínicos.

Variante 2: El equipo de desarrollo establece el uso de la herramienta REKO, y garantizará copias espejos de la base de datos y el repositorio de documentos clínicos.

Para el cifrado y firmado de los documentos que se intercambian se usan las librerías de Java Cryptography Extension, con la ayuda de estas librerías y el componente de criptografía del sistema se generan las firmas digitales XAdES de cada usuario cumpliendo con las especificaciones definidas por el

CAPÍTULO 2. CARACTERÍSTICAS DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

European Telecommunications Standards Institute (ETSI). Los documentos CDA se generan en formato XML, luego estos se firman utilizando el mecanismo anteriormente descrito.

Se mantendrá seguridad y control a nivel de usuario, garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo a la función que realizan. Las contraseñas podrán cambiarse solo por el propio usuario o por el administrador del sistema.

Se registrarán todas las acciones que se realizan, llevando el control de las actividades de cada usuario en todo momento. De conjunto con esto se establecerán mecanismos de control y verificación para los procesos susceptibles de fraude. Los mecanismos serán capaces de informar al personal autorizado sobre posibles irregularidades que den indicios sobre la introducción de información falseada.

El sistema implementará un mecanismo de auditoría para el registro de todos los accesos efectuados por los usuarios, proporcionando un registro de actividades (bitácora) de cada usuario en el sistema.

Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la BD, independientemente de que para el sistema, este elemento ya no exista.

El sistema permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

2.3 Modelo de casos de uso del sistema

El modelo de casos de uso del sistema es un artefacto de ingeniería de software que describe, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario, permitiendo de esta forma el establecimiento de un acuerdo entre clientes y desarrolladores sobre las condiciones y requerimientos que debe cumplir el sistema. Este modelo está formado por actores, casos de usos y las relaciones que se establecen entre estos, es decir representa gráficamente a los procesos y su interacción con los actores y constituye una entrada de gran valor para las siguientes fases de construcción de un software.

2.3.1 Definición de actores

Un actor es cualquier entidad externa al sistema que demanda una funcionalidad de este; puede ser también una abstracción de un software que interactúa de alguna manera con el mismo. Representa un rol que desempeñan una o varias personas, un equipo o un sistema automatizado. De esta misma forma

intercambian información con el sistema. Toda persona, grupo o fenómeno que coincida con una o varias de estas categorías es candidato para ser un actor.

Actor	Descripción
Usuario	Usuario global que se autentica en la aplicación, el sistema lo valida y le asigna un rol con la posibilidad de interactuar con el sistema.
Administrador	Realiza la configuración del módulo.

Tabla 1.Descripción de actores

2.3.2 Vista global de actores del sistema

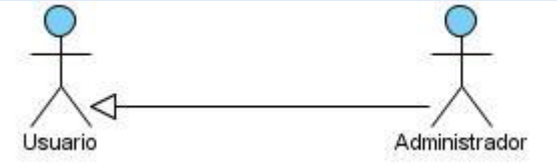
Vista global de actores del sistema


Figura 2. Vista global de actores del sistema

CAPÍTULO 2. CARACTERÍSTICAS DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

2.3.2 Diagrama de Casos de Uso del Sistema

Diagrama de Casos de Uso del Sistema

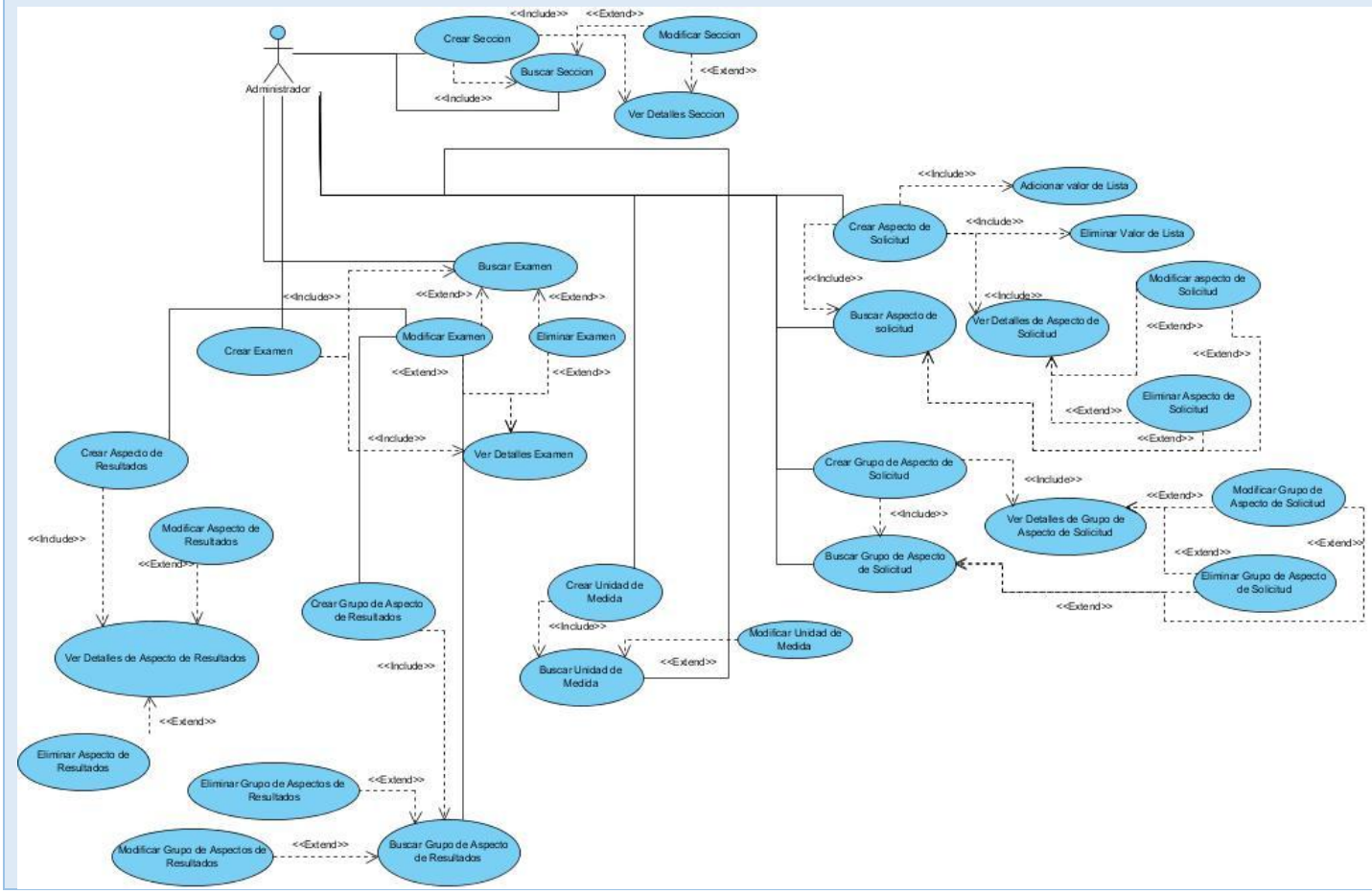


Figura 3. Diagrama de Casos de Uso del Sistema

2.3.3 Descripción de Casos de Uso

2.3.3.1 Caso de uso Crear sección.

CASO DE USO:	Crear sección
Resumen:	El caso de uso inicia cuando el actor accede a la opción Crear Sección, el sistema muestra los campos a llenar. El actor entra los nuevos datos y acepta. El sistema valida la información y crea la nueva sección, regresa a la vista anterior, el caso de uso termina.
Complejidad:	Alta

Prioridad:	Alta
Precondiciones:	
REFERENCIAS	
Actores:	Administrador
Requisitos:	**RF1**
Entidades:	Sección
Casos de Uso:	

FLUJO NORMAL DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el actor accede a la opción Crear Sección.	
	2. Brinda la posibilidad de introducir los criterios para crear sección: • Nombre • Placa y permite: • Crear sección. El sistema brinda las opciones “Aceptar”, “Cancelar”. Ver **Alternativa 1:** “Cancelar”
3. Introduce los nuevos datos y selecciona la opción “Aceptar”.	
	4. Valida los datos introducidos. Si existen datos incompletos o incorrectos, ver **Alternativa 2**: “Datos incorrectos o incompletos”.
	5. Adiciona la nueva sección.

	6. Muestra la interfaz correspondiente a Ver Detalles de la sección creada.
	7. El caso de uso termina.
[Prototipo de Interfaz]	
FLUJOS ALTERNOS	
Alternativa 1. "Cancelar operación."	**Respuesta del Sistema**
1. **Acción del Actor**	
Selecciona la opción de Cancelar operación.	2. Cierra la vista actual.
	3. El caso de uso termina.
Alternativa 2: "Existen datos incompletos o incorrectos"	**Respuesta del Sistema**
Acción del Actor	1. Muestra un indicador sobre los campos incompletos o incorrectos.
	2. Regresa al paso 3 del **Flujo Normal de Eventos.**
Poscondiciones	Se gestiona una nueva sección.

Tabla 2.Caso de uso Crear Sección

2.3.3.2 Caso de uso Crear examen

CASO DE USO:	Crear examen
Resumen:	El caso de uso inicia cuando el actor accede a la opción Crear examen, el sistema muestra los campos a llenar. El actor entra los nuevos datos y acepta. El sistema valida la información y si no existen errores crea el nuevo examen, regresa a la vista anterior, el caso de uso termina.
Complejidad:	Alta
Prioridad:	Alta
Precondiciones:	No tiene
REFERENCIAS	

CAPÍTULO 2. CARACTERÍSTICAS DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

Actores:	Administrador
Requisitos:	**RF2**
Entidades:	Examen
Casos de Uso:	

FLUJO NORMAL DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el actor accede a la opción Crear examen	
	2. Brinda la posibilidad de introducir los criterios para crear examen: • Nombre • Sección • Requisitos y permite: • Crear examen. El sistema brinda las opciones “Aceptar”, “Cancelar”. Ver **Alternativa 1:** “Cancelar”
3. Introduce los nuevos datos y selecciona la opción “Aceptar”.	
	4. Valida los datos introducidos. Si existen datos incompletos o incorrectos, ver **Alternativa 2**: “Datos incorrectos o incompletos”.
	5. Adiciona el nuevo examen.
	6. Muestra la interfaz correspondiente a Ver detalles del examen creado.
	7. El caso de uso termina.
[Prototipo de Interfaz]	

FLUJOS ALTERNOS	
Alternativa 1. "Cancelar operación."	**Respuesta del Sistema**
Acción del Actor	
1. Selecciona la opción de Cancelar operación.	2. Cierra la vista actual.
	3. El caso de uso termina.
Alternativa 2: "Existen datos incompletos o incorrectos"	**Respuesta del Sistema**
Acción del Actor	1. Muestra un indicador sobre los campos incompletos o incorrectos.
	2. Regresa al paso 3 del **Flujo Normal de Eventos.**
[Prototipo de Interfaz]	
Poscondiciones	Se crea un examen.

Tabla 3.Caso de Uso Gestionar Examen

2.3.3.3 Caso de uso Crear aspecto de solicitud

CASO DE USO:	Crear aspecto de solicitud
Resumen:	El caso de uso inicia cuando el actor accede a la opción Crear aspecto de solicitud, el sistema muestra los campos a llenar. El actor entra los nuevos datos y acepta. El sistema valida la información y si no existen errores crea el nuevo aspecto de solicitud, regresa a la vista anterior, el caso de uso termina.
Complejidad:	Alta
Prioridad:	Media
Precondiciones:	No tiene
REFERENCIAS	
Actores:	Administrador
Requisitos:	**RF1**

Entidades:	Aspecto de Solicitud
Casos de Uso:	

FLUJO NORMAL DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el actor accede a la opción Crear aspecto de solicitud.	
	2. Brinda la posibilidad de introducir los criterios para crear un aspecto de solicitud: • Nombre • Tipo de dato y permite: • Crear aspecto de solicitud. El sistema brinda las opciones “Aceptar”, “Cancelar”. Ver **Alternativa 1:** “Cancelar”
3. Introduce los nuevos datos y selecciona la opción “Aceptar”.	
	4. Valida los datos introducidos. Si existen datos incompletos o incorrectos, ver **Alternativa 2**: “Datos incorrectos o incompletos”.
	5. Adiciona el nuevo aspecto de solicitud.
	6. Muestra la interfaz correspondiente a los detalles del aspecto de solicitud creado.
	7. El caso de uso termina.
[Prototipo de Interfaz]	
FLUJOS ALTERNOS	

Alternativa 1. "Cancelar operación."	**Respuesta del Sistema**
Acción del Actor	
1. Selecciona la opción de Cancelar operación.	2. Cierra la vista actual.
	3. El caso de uso termina.
Alternativa 2: "Existen datos incompletos o incorrectos"	**Respuesta del Sistema**
Acción del Actor	1. Muestra un indicador sobre los campos incompletos o incorrectos.
	2. Regresa al paso 3 del **Flujo Normal de Eventos.**
[Prototipo de Interfaz]	
Poscondiciones	Se crea un aspecto de solicitud.

Tabla 4.Caso de Uso Gestionar Aspecto Solicitud

Conclusiones

En este capítulo se obtuvo el modelo de dominio, donde se abarcaron las definiciones asociadas al sistema y las relaciones definidas entre ellas, lo que posibilita un mejor entendimiento del problema y una rápida identificación de las funcionalidades y propiedades con las que debe contar la nueva estructura de la configuración del módulo. Se enunciaron los requisitos funcionales y no funcionales y se definió el modelo de casos de uso del sistema con su diagrama y especificaciones, con el fin de lograr el desarrollo de una aplicación que responda satisfactoriamente a la situación problémica planteada.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA RESTRUCTURACIÓN DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

En el presente capítulo se realiza una descripción detallada del sistema propuesto y se explica la arquitectura que presenta el mismo. Para la realización de las actividades correspondientes a esta etapa se utilizan como entrada los artefactos que han sido obtenidos en los flujos de trabajo anteriores y refinados por los que se han llevado a cabo posteriormente. El diseño debe implementar los requisitos explícitos contenidos en el modelo de análisis, y debe tener en cuenta los requisitos implícitos que desea el cliente.Se presentan los principales patrones de diseño que se ponen en práctica y los diagramas de clases del diseño e interacción que forman parte del modelo de diseño. Se ofrece una breve descripción de las clases u operaciones necesarias para el funcionamiento de la restructuración de la configuración del módulo de Laboratorio.

3.1 Descripción de la arquitectura y fundamentación

La arquitectura de software es un conjunto de patrones y abstracciones coherentes que proporcionan un marco de referencia necesario para guiar la construcción de un software dentro de un sistema informático. Es considerada el diseño de mayor nivel de la estructura de un sistema. Permite a los programadores, diseñadores, ingenieros y analistas trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado. (11)

Para el desarrollo de las funcionalidades de la aplicación y teniendo en cuenta las herramientas, tecnologías y metodologías propuestas, se define como arquitectura la implementación del patrón de diseño de arquitectura Modelo Vista Controlador (MVC). Este patrón es el utilizado en el Departamento de Gestión Hospitalaria del CESIM y permite la separación de los datos de una aplicación, la interfaz de usuario y la lógica de control, en tres componentes. Estos componentes son: el modelo, para la administración de los datos, la vista, que muestra la información del modelo al usuario y el controlador, que gestiona las entradas del usuario. Con este patrón se logra realizar un diseño que desacople la vista del modelo y permita la reusabilidad de los componentes.

El modelo de datos se encarga del almacenamiento de los datos, su estructura y las relaciones entre los mismos. Este se evidencia en la relación de los elementos en la configuración del módulo de Laboratorio como por ejemplo: las secciones con sus respectivas observaciones. Para la administración de los datos se utiliza el framework Hibernate, lo cual tiene como ventajas el control de la accesibilidad a la información

almacenada, así como la realización de rápidas consultas. Mantiene referencias a los objetos cargados y devuelve las mismas cada vez que se solicita un objeto ya cargado. Permite mapear las relaciones entre referencias entre objetos y claves entre tablas.

El controlador está compuesto por las clases controladoras de los procesos identificados y modelados, además de las controladoras personalizadas y autogeneradas. Entre esas clases están la controladora de los exámenes y la controladora de las pruebas asociadas a los exámenes. Es en estás clases que se manejan las reglas que regulan el flujo de la información. Acceden al modelo de datos a medida que se producen los eventos en las vistas que requieren de una determinada información.

Las vistas se identifican con las interfaces con las que el usuario interactúa que muestran la información manejada por el modelo. En las nuevas funcionalidades de la configuración del módulo de Laboratorio se tienen ejemplos de la utilización de las vistas mencionadas: la interfaz de Modificar observación y Crear sección. Es a través de ellas que se introducen los datos requeridos para la ejecución de una u otra funcionalidad, además de que también se visualiza el resultado de las operaciones realizadas que son de interés para el usuario.

La capa de negocio está constituida por clases controladoras que se encargan de definir la lógica del negocio del módulo, así como del manejo y validación de los datos capturados en la capa de presentación.

Por último, la capa de acceso a datos tiene la principal función de cargar, modificar, eliminar y persistir la información existente en la base de datos una vez que sea validada, ejemplo en la funcionalidad de Gestionar exámenes, la creación de una nueva prueba de un examen existente en el sistema. Este proceso se logra gracias al uso de componentes Hibernate por los que se encuentra constituida dicha capa. Estos componentes logran abstraer al desarrollador del gestor de base de datos utilizado, mediante el mapeo de tablas. Esto da la posibilidad de llevar las consultas a un lenguaje de objetos.

3.2 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de la implementación. En la realización del mismo y con el objetivo de mejorar la calidad de los diagramas correspondientes a esta disciplina fueron aplicados patrones de diseño.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA RESTRUCTURACIÓN DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces, son la especificación del problema y la solución del mismo. Un patrón de diseño es una solución estándar para un problema común. Deben ser totalmente reusables sin importar el contexto en que se presente el problema a solucionar.

Los patrones GRASP (Patrones de los Principios Generales para Asignar Responsabilidades) tuvieron un gran uso en el diseño realizado. Permitiendo asignar a cada clase las tareas posibles a realizar en correspondencia con la información manejada, así mismo dándole a la misma el poder instanciar otras clases en dependencia de la responsabilidad asignada; manifestando los patrones experto y creador. El uso de los mismos contribuyó a la conservación del encapsulamiento, puesto que para lograr que cada objeto realice alguna tarea se baste con la información que maneja explícitamente. En el diseño realizado está presente el bajo acoplamiento y alta cohesión, lo cual hace posible que las clases colaboren entre ellas y que además sean reutilizables.

En el modelo de diseño se realizan una serie de diagramas, como son los diagramas de clases y de interacción. Los diagramas de interacción reflejan a los objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos. En el modelo del diseño del sistema propuesto se identificaron una serie de subsistemas que cada uno de ellos contiene una serie de elementos que por las características que presentan lo dotan de una gran importancia.

3.2.1Diagrama de Paquetes

En la confección del diseño se define una estructura de paquetes que permite dividir el sistema en fragmentos manejables para su implementación. Se emplea además el criterio de empaquetamiento por procesos y por clases, siguiendo la estructura de procesos definidos en el sistema. Los paquetes que hacen referencia a procesos se componen por subcarpetas que responden a las realizaciones de los casos de uso y los cuales contienen un diagrama de clases y los respectivos diagramas de secuencia.

Existe un paquete repositorio de clases que contiene tres subpaquetes: el de las vistas, el de las sesiones y el paquete de las entidades.

El paquete de las vistas incluye los contenidos web referentes a las páginas clientes y los formularios que la componen. En este paquete se agruparon las vistas de la nueva estructura de la configuración del

módulo de Laboratorio, entre las que se encuentra: Gestionar_Examenes, Gestionar_Observaciones, ModificarGestionPrueba, y Modificar_Observacion_Solicitud.

En el de sesiones se agrupan las clases controladoras autogeneradas por el entorno de desarrollo y las clases controladoras personalizadas.

El subpaquete de las entidades contiene las entidades autogeneradas y las personalizadas. Las entidades autogeneradas son obtenidas mediante el mapeo de la base de datos, ejemplo: examen_lab, seccion_laboratorio y estructura_resultado_lab. Las entidades personalizadas son las entidades autogeneradas que se personalizan para lograr un mejor funcionamiento, ejemplo: Examen_Laboratorio_List_Custom, Seccion_Laboratorio_List_Custom.

Diagrama de Paquetes

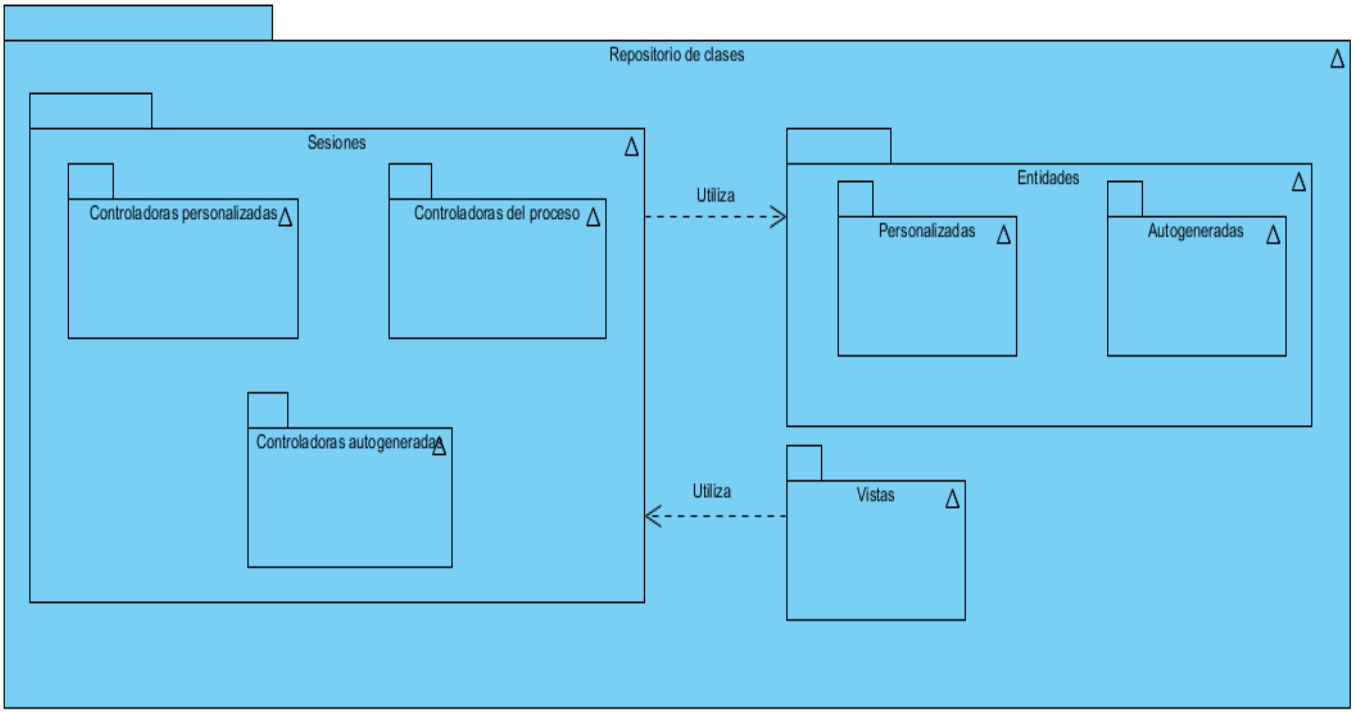


Figura 4.Diagrama de Paquetes del Sistema

Descripción de cada paquete:

Repositorio de clases: Contiene todas las clases definidas en el diseño de acuerdo con las tecnologías que serán usadas en la implementación del sistema.

Entidades: Contiene las entidades autogeneradas y personalizadas de la base de datos.

Entidades autogeneradas: Entidades autogeneradas de la base de datos. (Generadas utilizando el ORM Hibernate).

Entidades personalizadas: Entidades autogeneradas que se personalizan.

Sesiones: Contiene las controladoras autogeneradas y personalizadas de la base de datos, y además las propias del proceso.

Controladoras autogeneradas: Clases controladoras autogeneradas por el entorno de desarrollo.

Controladoras personalizadas: Clases controladoras autogeneradas que son personalizadas.

Controladoras del proceso: Clases controladoras de cada uno de los procesos automatizados.

El criterio de agrupamiento utilizado en los paquetes del proceso fue por casos de uso, aunque la mayoría de ellos comparten en gran medida las entidades asociadas a los casos de uso contenidos en cada uno de ellos.

3.2.2 Diagramas de clases del diseño

Para modelar los diagramas de clases del diseño se emplea el Lenguaje Unificado de Modelado, el cual provee los elementos necesarios para representar estereotipos web. Estos elementos tienen como principales componentes las clases “server page “, “client page” y “form”, para la representación de las clases contenedoras del código de las páginas servidoras, clientes y los formularios respectivamente.

Las clases contenedoras del código servidor se encargan de generar el código de las páginas clientes, las páginas clientes componen los formularios (es lo que hace que entre las páginas clientes y sus formularios exista la relación de agregación), a través de los cuales se muestran e introducen los datos necesarios, los cuales son enviados hacia la página servidora que construyó la página cliente asociada. Entre las distintas páginas clientes pueden existir vínculos y redireccionamientos. Las páginas servidoras pueden construir varias páginas clientes, pero esta puede ser generada por una única página servidora, aunque la misma necesite incluir una u otras clases de su tipo.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA RESTRUCTURACIÓN DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

Para representar las clases previamente mencionadas se procedió usando la siguiente nomenclatura, Formulario_<Nombre_Formulario>, PC_<Nombre_ClaseCliente>, PS_<Nombre_ClaseServidora> y <NombreClaseControladora>, para los formularios, páginas clientes, clases servidoras y controladoras respectivamente ubicadas en los diferentes niveles de la aplicación.

Diagrama de Clases de Diseño Gestionar Observación Solicitud

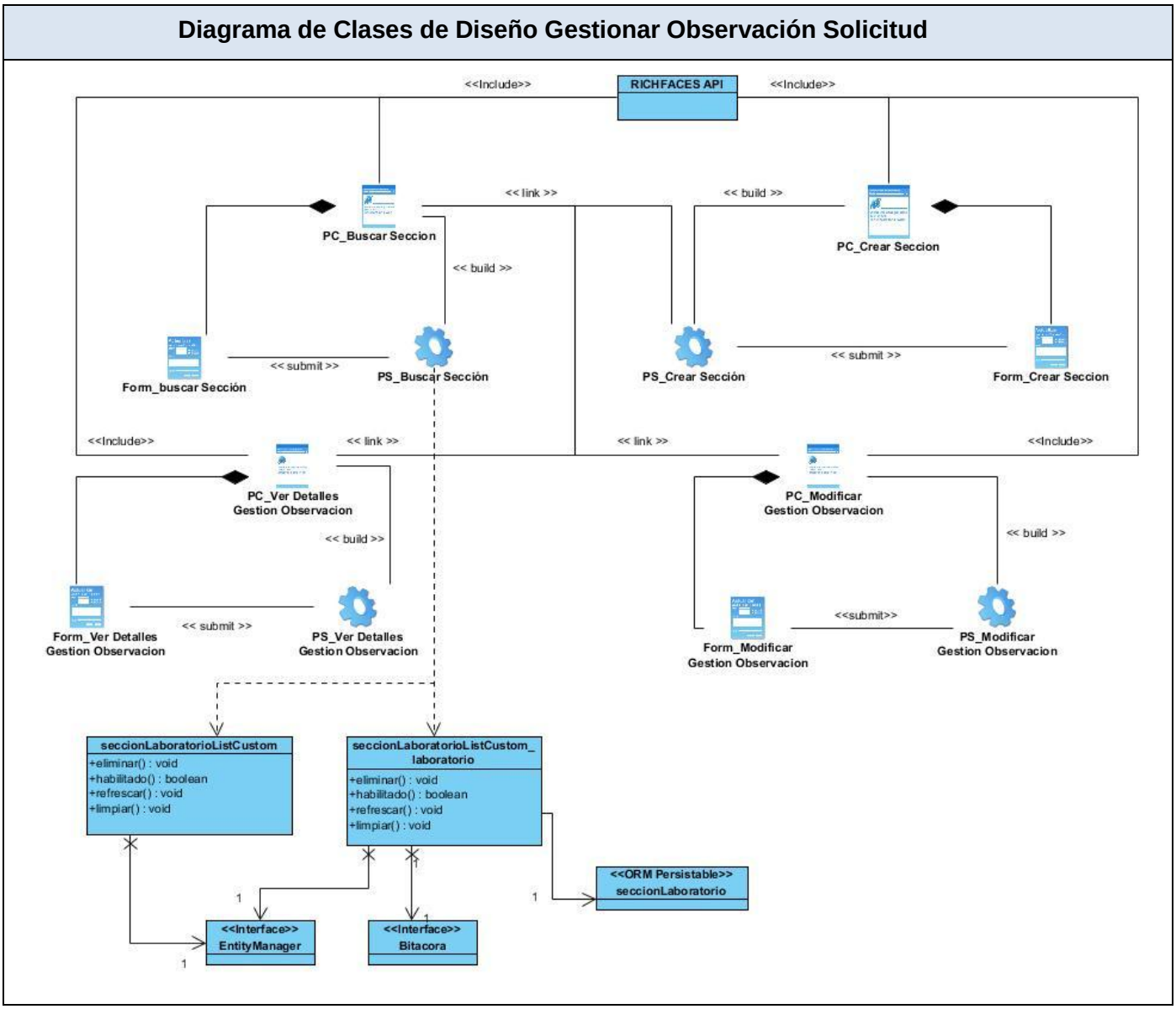


Figura 5. Diagrama de Clases de Diseño Gestionar Observación Solicitud

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA RESTRUCTURACIÓN DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

Diagrama de Clases de Diseño Aspecto de Solicitud

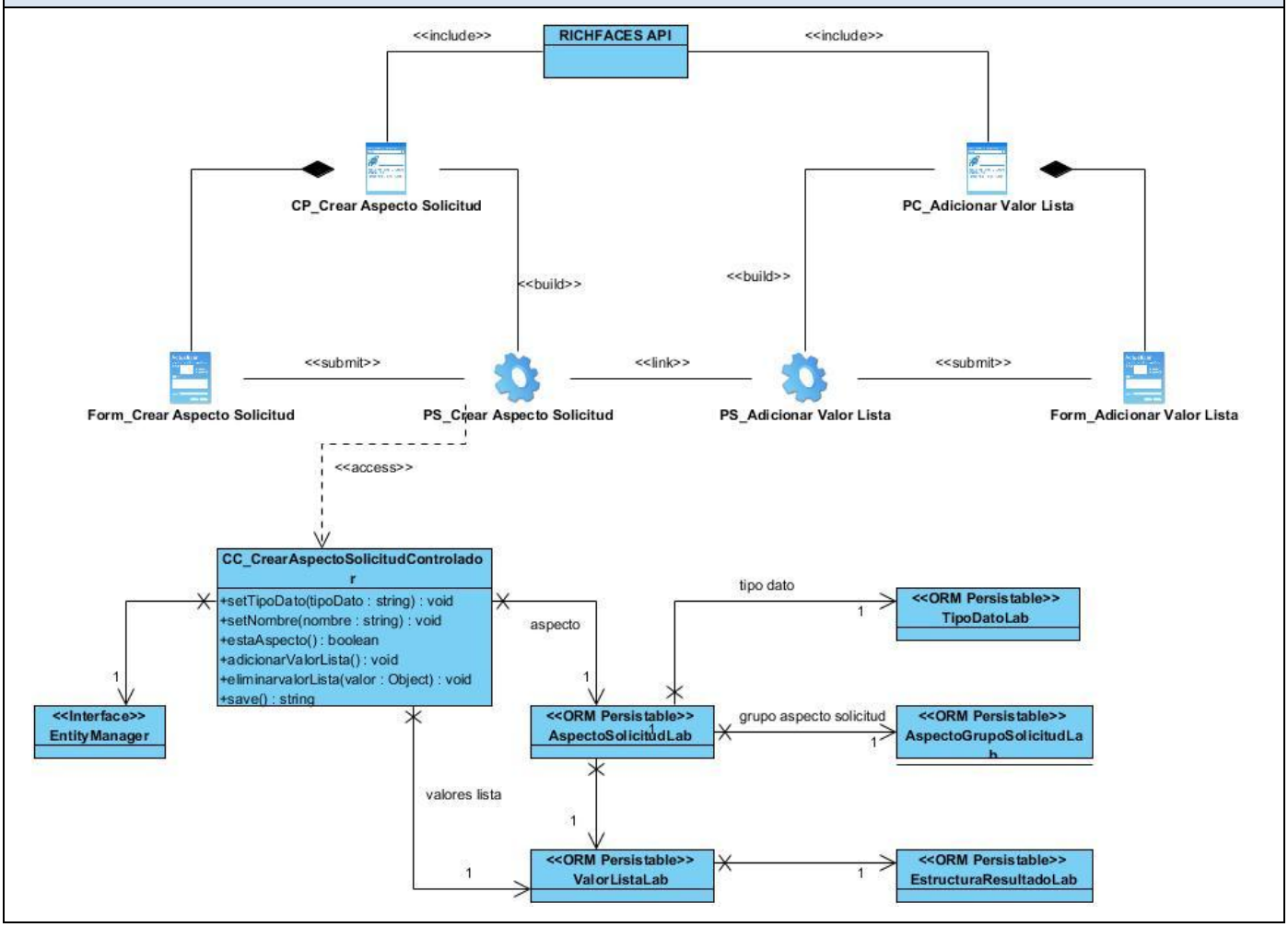


Figura 6. Diagrama de Clases de Diseño Crear Aspecto de Solicitud

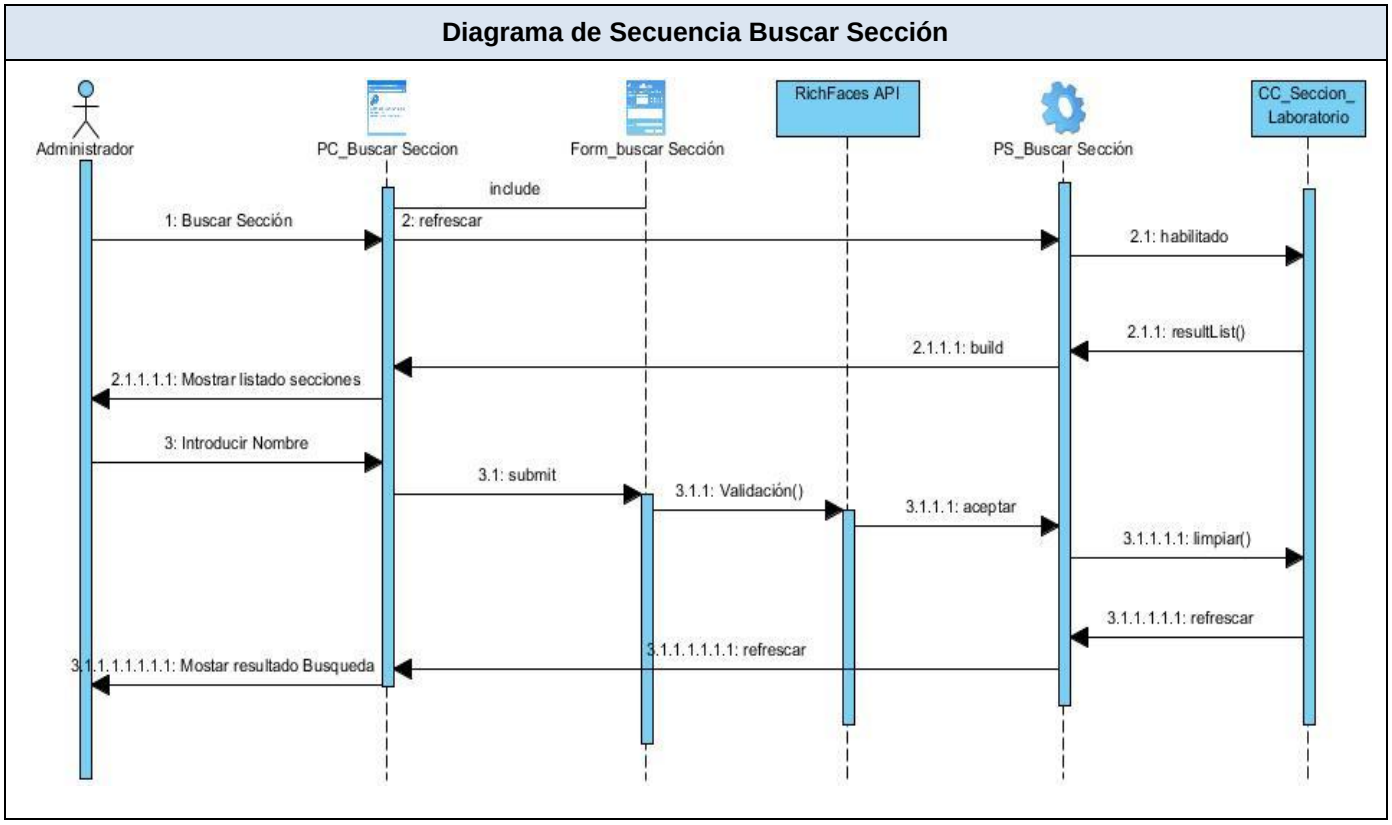


Figura 7. Diagrama de Secuencia: Buscar Sección.

3.2.3 Descripción de las clases y sus atributos

A continuación se realiza la descripción de las principales clases que han sido identificadas en el diseño para su futura implementación, con el objetivo de lograr una comprensión más amplia del sistema en cuestión.

Nombre: CrearSeccion	
Tipo de clase: Controladora.	
Atributo	**Tipo**
nombre	String
Id	Long
antibiograma	Boolean
cid	Long
seccion	SeccionLaboratorio
activeModule	IActiveModule
Para cada responsabilidad:	
Nombre:	ExisteSeccion(): boolean
Descripción:	Comprueba si la sección que se va a crear está ya en la base de datos del sistema. Retorna true

	o false en dependencia de la acción realizada.
Nombre:	Save(): string
Descripción:	Guarda la nueva sección creada en la base de datos del sistema. Retorna un string con un mensaje que indica si la sección se guardó correctamente u ocurrió algún error en el proceso de guardado.

Tabla 5.**Descripción de Clase Controladora CrearSección.**

Nombre: CrearExamen	
Tipo de clase: Controladora.	
Atributo	**Tipo**
nombre	String
seccion	String
requisitos	String
id	Long
cid	Long
Examen	ExamenLab
activeModule	IActiveModule
Para cada responsabilidad:	
Nombre:	ExisteExamen(): boolean
Descripción:	Comprueba si el examen que se va a crear está ya en la base de datos del sistema. Retorna true o false en dependencia de la acción realizada.
Nombre:	Save(): string
Descripción:	Guarda el nuevo examen creado en la base de datos del sistema. Retorna un string con un mensaje que indica si el examen se guardó correctamente u ocurrió algún error en el proceso de guardado.

Tabla 6.**Descripción de Clase Controladora CrearExamen.**

Nombre: CrearAspectoSolicitud	
Tipo de clase: Controladora.	
Atributo	**Tipo**
nombre	String
Id	Long
tipoDato	String
cid	Long
aspectoSolicitud	AspectoSolicitud
valoresLista	List<ValorListaLab>
tipoDatoCodigo	String
activeModule	IActiveModule
Para cada responsabilidad:	
Nombre:	setTipoDato(String tipoDato): void
Descripción:	Verifica si el tipo de dato del aspecto de solicitud tiene algún valor.
Nombre:	setNombre(String nombre): void
Descripción:	Posibilita cambiar el nombre del aspecto de solicitud seleccionado.

Nombre:	estaAspecto(): boolean
Descripción:	Verifica si está el aspecto de solicitud en cuestión en la base de datos del sistema. Retorna true o false en dependencia del resultado de la consulta.
Nombre:	adicionarValorLista(): void
Descripción:	Adiciona un nuevo valor a la lista de valores de los aspectos de solicitud.
Nombre:	eliminarvalorLista(Object valor): void
Descripción:	Elimina el valor de la lista de valores del aspecto de solicitud.
Nombre:	save(): string
Descripción:	Guarda el aspecto de solicitud en la base de datos del sistema. Retorna un string con un mensaje que indica si el aspecto se guardó correctamente u ocurrió algún error en el proceso de guardado.

Tabla 7.Descripción de Clase Controladora CrearAspectoSolicitud

Conclusiones

Con el desarrollo de este capítulo se definieron las clases necesarias para el correcto funcionamiento de la configuración del módulo de Laboratorio. De igual manera fueron descritos los atributos y métodos que deben tener las clases, los cuales servirán de apoyo al desarrollador durante la fase de implementación. Se elaboraron los diagramas de clases de diseño y los diagramas de secuencia correspondientes a cada funcionalidad.

CAPÍTULO 4. IMPLEMENTACIÓN DE LA RESTRUCTURACIÓN DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

En este capítulo se introduce el flujo de trabajo de implementación, partiendo de los resultados obtenidos en el diseño. Primeramente, se detalla el Modelo de Datos, que refleja la estructura donde se almacena toda la información requerida en el sistema. Luego se muestra el Modelo de Implementación, que está compuesto por el diagrama de despliegue y el diagrama de componentes. Estos diagramas, describen los componentes a construir, su organización y dependencias entre los nodos físicos en los que funcionará la aplicación. También se exponen aspectos referentes a la seguridad del sistema, las estrategias de codificación, así como la forma en que se tratarán los errores.

4.1 Modelo de Datos

El Modelo de Datos es la traducción del análisis de requisitos al esquema conceptual, mediante una representación gráfica de las entidades y sus relaciones. Es usado para definir el mapeo entre las clases del diseño y las estructuras de datos. Está compuesto por entidades, atributos y sus relaciones.

Las entidades son objetos de los que el sistema necesita guardar información.

Los atributos son las características asociadas a una entidad. Estos pueden ser clasificados en obligatorios, opcionales, claves foráneas y claves primarias (estas se dividen en simples y compuestas).

Las relaciones, por su parte, muestran la forma en que dos entidades se asocian. Se representan mediante una línea que une a las dos entidades implicadas y manifiestan dos características principales: la cardinalidad y la obligatoriedad.

La cardinalidad se refiere al número de ocurrencias de una entidad con respecto a la otra. La entidad de la que sale la relación tendrá tantas ocurrencias como indique el número asociado a la entidad a donde llega la relación señalando con una flecha el sentido de entrada, de no mostrarse el número de la cardinalidad se asume que la ocurrencia es de solamente una vez.

La obligatoriedad determina que ante la existencia de una entidad puede haber ocurrencias de otras relacionadas con esta. Si la ocurrencia es obligatoria se representa mediante una línea continua, en caso contrario, se realiza a través de una línea discontinua.

La anterior descripción de los componentes del modelo de datos, proporcionará un mejor entendimiento del diagrama que se presenta a continuación:

CAPÍTULO 4. IMPLEMENTACIÓN DE LA RESTRUCTURACIÓN DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

DIAGRAMA DE ENTIDAD RELACIÓN

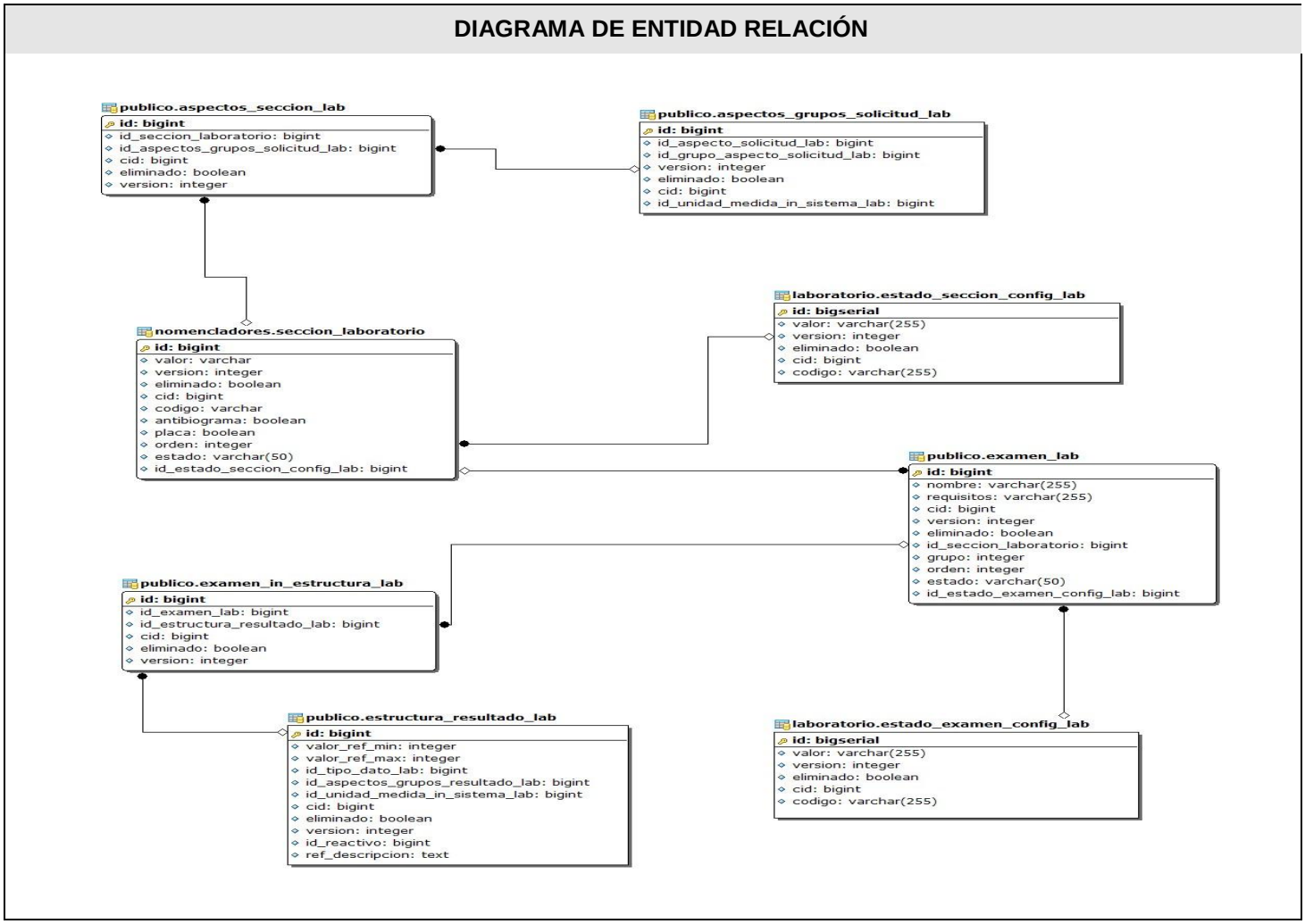


Figura 8. Diagrama entidad relación

4.1.1 Descripción de las tablas

Nombre: examen_lab		
Descripción: Todos los exámenes que se realizan en el laboratorio.		
Atributo	**Tipo**	**Descripción**
nombre	String	Nombre de examen.
requisitos	String	Los requisitos que tiene que cumplir el paciente para ir a hacerse el examen.
id	Long	Identificador de la entidad único y autoincremental.
cid	Integer	Identificador único de la conversación.
version	Integer	Versión del objeto.
eliminado	boolean	Si la información está eliminada.

Tabla 8.Descripción examen_lab

Nombre: estructura_resultado_lab		
Descripción: Las pruebas que contienen un examen en el laboratorio clínico		
Atributo	**Tipo**	**Descripción**
Ref_descripcion	String	Referencia que posee el examen relacionado
codigo	String	Valor que identifica la entidad y no cambia.
valor_ref_min	Integer	Valor mínimo que puede tener este aspecto.
valor_ref_max	Integer	Valor máximo que puede tener este aspecto.
id	Long	Identificador de la entidad único y autoincremental.
cid	Integer	Identificador único de la conversación.
version	Integer	Versión del objeto.
eliminado	boolean	Si la información está eliminada.

Tabla 9.Descripción estructura_resultado_lab

Nombre: examen_in_estructura_lab		
Descripción: Registra el valor de cada aspecto de resultado.		
Atributo	**Tipo**	**Descripción**
cid	Integer	Identificador único de la conversación.
version	Integer	Versión del objeto.
eliminado	boolean	Si la información está eliminada.

Tabla 10. Descripción examen_in_estructura_lab

Nombre: estado_examen_config_lab		
Descripción: Los estados de los exámenes.		
Atributo	**Tipo**	**Descripción**
Valor	String	El valor del estado de los exámenes.
codigo	String	Valor que identifica la entidad y no cambia.
id	Long	Identificador de la entidad único y autoincremental.
cid	Integer	Identificador único de la conversación.
version	Integer	Versión del objeto.
eliminado	boolean	Si la información está eliminada.

Tabla 11. Descripción estado_examen_config_lab

Nombre: seccion_laboratorio		
Descripción: Las secciones del laboratorio.		
Atributo	**Tipo**	**Descripción**
Valor	String	Nombre de las secciones.
id	Long	Identificador de la entidad único y autoincremental.
cid	Integer	Identificador único de la conversación.
version	Integer	Versión del objeto.
eliminado	boolean	Si la información está eliminada.

Tabla 12. Descripción Tabla seccion_laboratorio

Nombre: aspectos_seccion_lab		
Descripción: Los aspectos asociados a las secciones del laboratorio.		
Atributo	**Tipo**	**Descripción**
id	Long	Identificador de la entidad único y autoincremental.
cid	Integer	Identificador único de la conversación.
version	Integer	Versión del objeto.
eliminado	boolean	Si la información está eliminada.

Tabla 13.Descripción aspectos_seccion_lab

Nombre: aspectos_grupos_solicitud_lab		
Descripción: Los aspectos asociados a los aspectos sección.		
Atributo	**Tipo**	**Descripción**
id	Long	Identificador de la entidad único y autoincremental.
cid	Integer	Identificador único de la conversación.
version	Integer	Versión del objeto.
eliminado	boolean	Si la información está eliminada.

Tabla 14.Descripción aspectos_grupos_solicitud_lab

Nombre: estado_seccion_config_lab		
Descripción: Los estados de las secciones.		
Atributo	**Tipo**	**Descripción**
Valor	String	El valor del estado de las secciones.
codigo	String	Valor que identifica la entidad y no cambia.
id	Long	Identificador de la entidad único y autoincremental.
cid	Integer	Identificador único de la conversación.
version	Integer	Versión del objeto.
eliminado	boolean	Si la información está eliminada.

Tabla 15.Descripción estado_seccion_config_lab

CAPÍTULO 4. IMPLEMENTACIÓN DE LA RESTRUCTURACIÓN DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

4.2 Implementación

El Modelo de Implementación consiste en obtener una visión general de lo que tiene que ser implementado, y una estructura para cada iteración con los componentes y subsistemas a implementar durante esa iteración, así como el testeo que se ha de realizar sobre ellos. Esta vista presenta como objetivo determinar la organización del código, planificar las integraciones de sistemas necesarias en cada iteración e implementar las clases y subsistemas definidos durante el diseño.

4.2.1 Diagrama de despliegue

El diagrama de despliegue muestra la configuración de los elementos de hardware (nodos) y muestra como los elementos y artefactos del software se trazan en esos nodos. Es un grafo de nodos unidos por conexiones que modela la arquitectura en tiempo de ejecución de un sistema. Permite describir la vista de despliegue estática de un sistema.

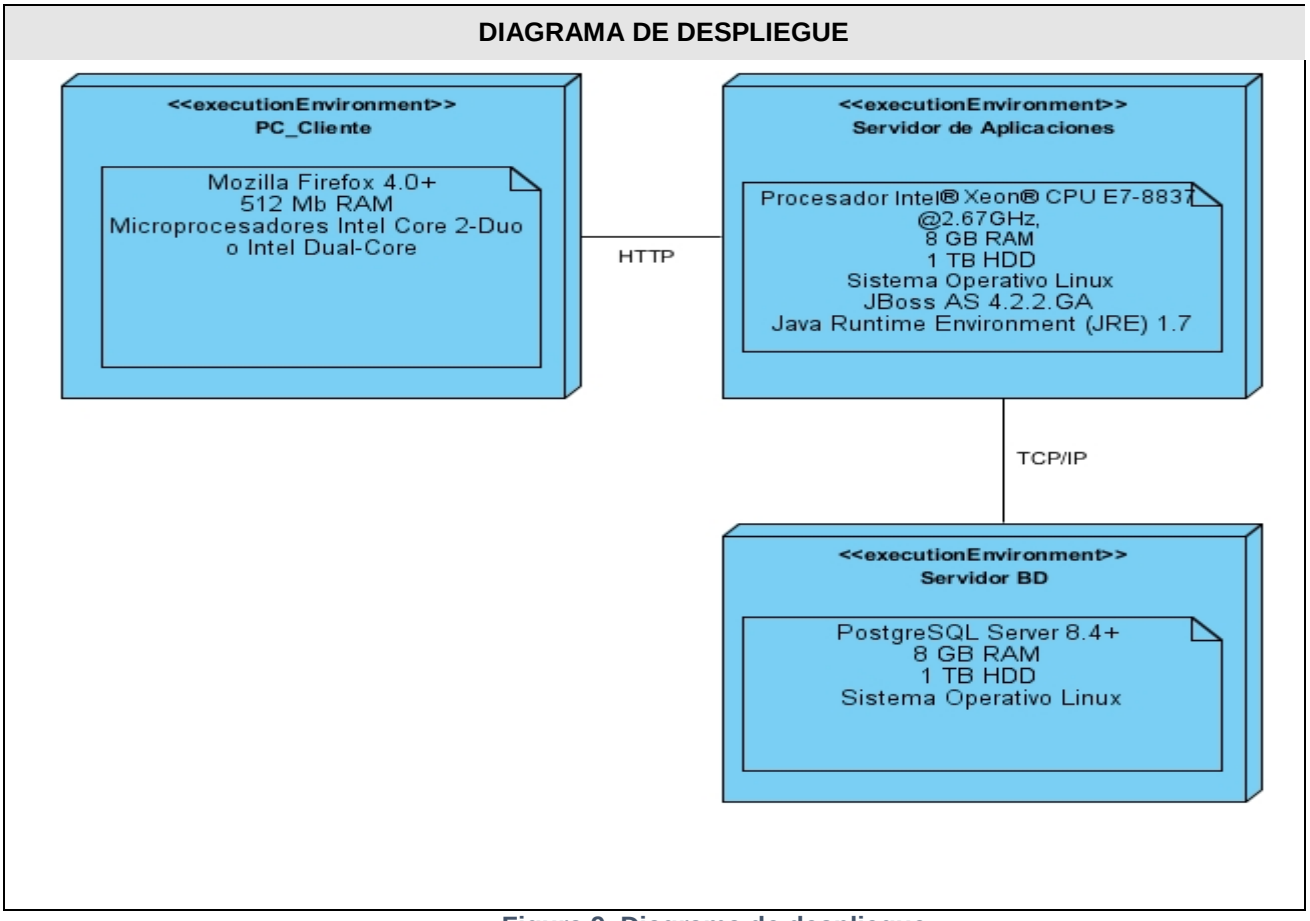


Figura 9. Diagrama de despliegue

4.2.2 Diagrama de componentes

El diagrama de componentes muestra el conjunto de componentes y sus relaciones. Permite describir la vista de implementación estática de un sistema. Un componente se corresponde con una o más clases, interfaces o colaboraciones.

DIAGRAMA DE COMPONENTES

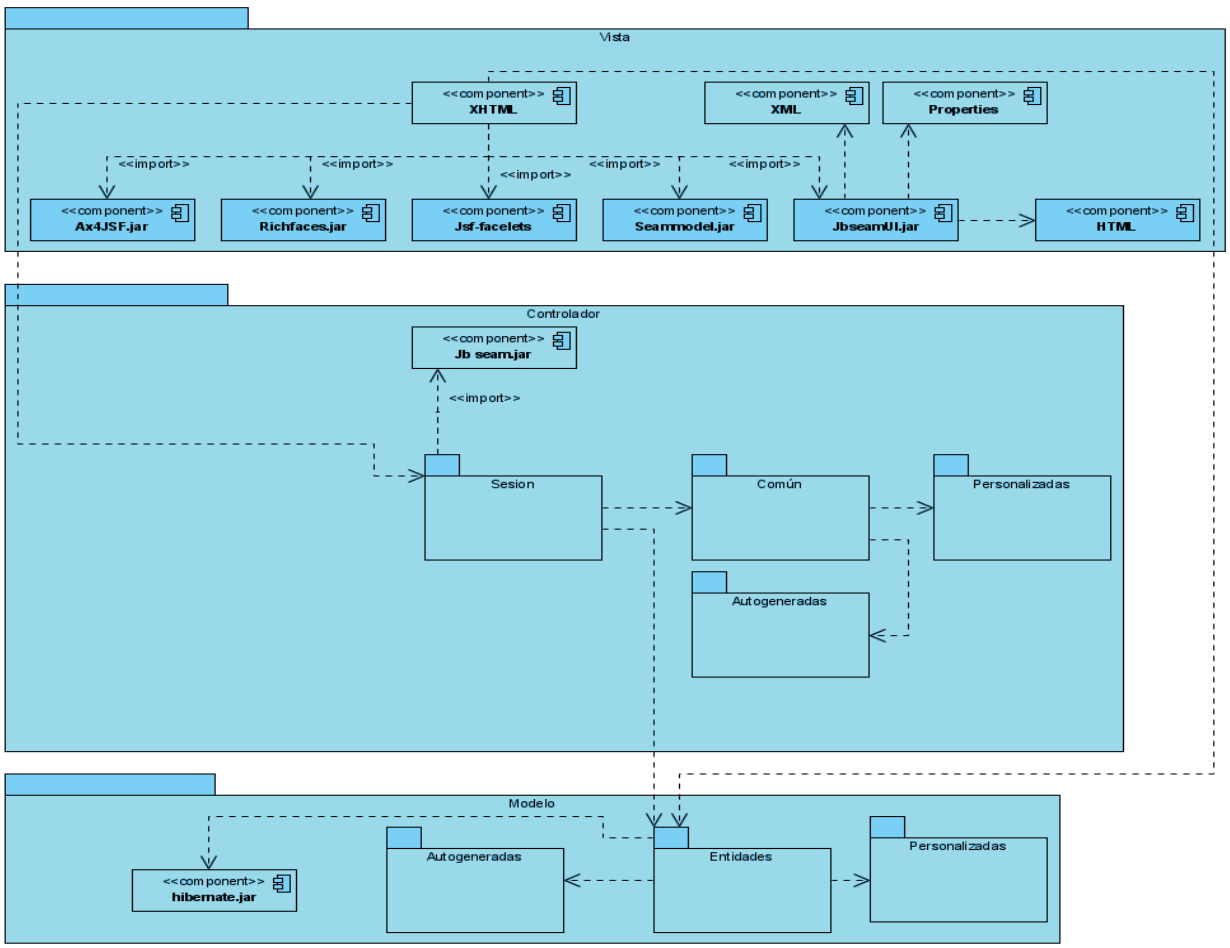


Figura 10. Diagrama de componentes

CAPÍTULO 4. IMPLEMENTACIÓN DE LA RESTRUCTURACIÓN DE LA CONFIGURACIÓN DEL MÓDULO DE LABORATORIO

4.3 Tratamiento de errores

El tratamiento de errores es un aspecto que se debe realizar sin importar el tipo de aplicación. Cuando no se tienen en cuenta los errores que puede dar el sistema y por consiguiente no se tratan los mismos es imposible comunicarle al usuario la causa y motivo del error, y además a veces es necesario el reinicio del sistema por lo que en ocasiones se pierden los datos que no hayan sido guardados por el mismo. Es por esto que es tan importante desarrollar esta tarea, aunque a veces se le reste importancia o se considere que no es necesario.

En el sistema se identifican cada uno de los puntos en los que puede darse el error, los mismos son capturados y gestionados a través de un archivo de configuración, en el cual se definen cada uno de los posibles errores y como deben ser mostrados los mismos de manera entendible para el usuario, además este manejo de errores se realiza en dependencia del tipo de error lanzado.

Otros elementos generales del tratamiento de errores son los siguientes:

- ✓ La corrección de errores en la introducción de datos será clara y fácil de realizar.
- ✓ La entrada de datos incorrecta será detectada claramente e informada al usuario.

Para el control de las demás excepciones es utilizado un componente del framework Seam. Un buen control de excepciones garantiza una mayor calidad de las funcionalidades brindadas; sobre este aspecto el lenguaje de programación empleado posibilita un manejo acorde de estos elementos, tratando los casos no básicos por separado, lo cual permite especificar qué acciones realizar en dependencia del tipo de excepción.

4.4 Seguridad

La seguridad es un elemento primordial en todo tipo de aplicación, pues aquí se define como se va a gestionar el acceso de los usuarios al sistema, así como los permisos efectivos para cada uno de ellos. Específicamente en el sistema propuesto en lo que se refiere a la autorización a los distintos recursos y elementos del sistema será basada en reglas y estará presente en todas las capas de la aplicación, esto posibilita que si hay algún cambio con respecto a esto solamente requiera la modificación de un único fichero.

El acceso al sistema será a nivel de usuarios y contraseñas, permitiéndole a los mismos realizar las acciones en correspondencia con los privilegios asignados por el administrador. Estas contraseñas solo pueden ser alteradas por el usuario en cuestión o por el administrador en caso excepcional. En las

estaciones de trabajo sólo se podrán ejecutar las aplicaciones definidas para la estación en la que se vaya a desplegar el sistema. Esto dota al sistema de un segundo nivel de seguridad que lo hace más fiable.

4.5 Estrategias de codificación. Estándares y estilos a utilizar

Un estándar de codificación comprende todos los aspectos de la generación de código, de tal manera que sea práctico y entendible para todos los programadores. Por lo general, incluye pautas sobre cómo nombrar variables y constantes, dónde ubicar comentarios, cómo poner entre paréntesis, forma de guión, entre otras. No detecta los errores existentes, más bien evita la ocurrencia de estos, lo que permite obtener un código de alta calidad.

4.5.1 Indentación del código fuente

El indentado debe ser de dos espacios por bloque de código. No se debe usar el tabulador; ya que este puede variar según la computadora o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deben estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción. Para el inicio y fin de bloque se recomienda dejar dos espacios en blanco desde la instrucción anterior.

4.5.2 Variables

El nombre empleado para las variables, debe permitir que con sólo leerlo se conozca el propósito de la misma.

El nombre que se le da a las variables debe comenzar con la primera letra en minúscula e identificará el tipo de datos al que se refiere. En caso de que sea un nombre compuesto, la segunda palabra, comenzará con letra inicial mayúscula. Ejemplo: adicionarObservacion

4.5.3 Separadores, líneas, espacios en blanco y márgenes

Ubicación de comentarios: se recomienda comentar al inicio de cada clase o función de forma que se especifique el objetivo de la misma así como los parámetros que usa (declarar tipos de datos, y objetivo del parámetro) entre otras cosas.

Líneas en blanco: se recomienda dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función.

Espacios en blanco: se recomienda usar espacios en blanco entre operadores lógicos y aritméticos para lograr una mayor legibilidad del código. Ejemplo: usuario = nombreUsuario. No se debe usar espacio en blanco después del corchete abierto y antes del cerrado de un arreglo, luego del paréntesis abierto y antes del cerrado o antes de un punto y coma.

4.5.4 Clases y objetos

El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.

Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing. Ejemplo: GestionarExamenes (). Para el caso de las instancias se comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.

El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula y estará en correspondencia al tipo de dato al que se refiere, en caso de que sea un nombre compuesto, la segunda palabra comenzará con mayúscula.

Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que hacen las mismas. Ejemplo: Salvar (). Si son funciones que obtienen un dato se emplea el prefijo “get” y si fijan algún valor se emplea el prefijo “set”.

Conclusiones

Una vez concluido este capítulo se logró dar solución a los requisitos funcionales y no funcionales especificados anteriormente. Para ello se estableció el modelo de datos del sistema, el diagrama de despliegue, el diagrama de componentes y se realizó una descripción de las tablas de la base de datos. Durante el proceso de codificación se cumplió con los estándares y estilos definidos, lo que permitió obtener una aplicación entendible para todos los programadores y fácil de mantener.

CONCLUSIONES

Conclusiones

- La arquitectura asimilada provee un ambiente robusto, seguro y flexible para el desarrollo del sistema, permitiendo el uso de recursos novedosos en las aplicaciones web de gestión.
- El estudio de los Sistemas de Información de Laboratorio existentes no responden las necesidades identificadas del módulo de Laboratorio del Sistema de Información Hospitalaria del CESIM.
- La implementación de la restructuración de la configuración del módulo de Laboratorio del Sistema de Información Hospitalaria del CESIM, permite un mejor manejo de la información e integración de los elementos en el módulo de Laboratorio.
- La realización de una restructuración de la configuración del módulo dota a los procesos realizados de una mayor integridad y eficiencia.
- Se disminuyó el desacople de las funcionalidades de la configuración del módulo de Laboratorio del Sistema de Información Hospitalaria del CESIM.

RECOMENDACIONES

Recomendaciones:

- Agregar una funcionalidad que permita que un examen que esté en uso pueda quedar no disponible.
- Permitir observaciones generales válidas para todas las secciones.
- Permitir que se puedan crear observaciones sin que estén asociadas a ningún grupo.

REFERENCIAS BIBLIOGRAFICAS

Referencias bibliográficas

1. Gil, Antonio López. Sistemas de información del laboratorio clínico Cap. 4. Pamplona, España : Sociedad Española de Informática de la Salud, 2004.

2. DÍAZ, MIGUEL E. MARÍN. Fundamentos del Sistema de Salud Pública en Cuba para estudiantes de Informática. p 3. La Habana. Cuba : s.n., 2006.

3. CHEMEUROPE.COM. [En línea] Marzo de 2009. http://www.chemie.de/products/es/52345/.

4. Poceiro, Juan. Wiener lab. LIS : Informática para Laboratorios. Buenos Aires,Argentina : s.n., 2010.

5. Softel. Manual de usuario del Galen Lab. [En línea] Marzo de 2009. http://www.softel.cu/productos.htm.

6. Pelaéz, Juan. Arquitectura basada en capas. [En línea] 26 de Mayo de 2009. http://jtentor.com.ar/post/Arquitectura-de-N-Capas-y-N-Niveles.aspx.

7. Instituto de Física Aplicada del CSIC. ¿Qué es JavaScript? [En línea] Abril de 2009. http://www.iec.csic.es/criptonomicon/java/quesjava.html.

8. Wenz, Christian. Programming ASP.NET AJAX. California, E.E.U.U : s.n., 2007.

9. Nusairat, Joseph Faisal. Beginning JBoss Seam: From Novice to Professional. New York, E.E.U.U : s.n., 2008.

10. Rational Unified Process (RUP). Proven best practices for software and systems delivery. [En línea] Enero de 2010. http://www-01.ibm.com/software/rational/rup/.

11. Yera, Angel Cabo. Estudio Científico de las Arquitectura de Software. Madrid, España : Vision Libros, 2010.

12. [En línea] [Citado el: 5 de enero de 2014.] http://www.access.gpo.gov/cgibin/cfrassemble.cgi?title=200521.

13. [En línea] [Citado el: 5 de enero de 2014.] http: / /www.InfoScienceToday.org.

14. [En línea] [Citado el: 5 de enero de 2014.] http://cap.org.

15. "Microbiología Clínica", en Escolar y Carnicero (Coord.), VI Informe SEIS, El sistema integrado de información clínica. s.l. : Ed. Sociedad Española de Informática de la Salud.Pamplona.

16. López, A. "El laboratorio general: mecanización y gestión", en Escolar y Carnicero (Coord.), VI Informe SEIS, El sistema integrado de información clínica. s.l. : Ed. Sociedad Española de Informática de la Salud. Pamplona., 2004.

17. "Tendencias actuales en los sistemas de información del laboratorio clínico",Todo Hospital, vol. 7. 2000.

18. "La intranet como soporte del sistema de calidad del laboratorio" en Gestión y calidad total en el laboratorio clínico. . s.l. : Ed. Mapfre. Madrid., 1999.

19. Yasser Manuel Garbey Bermudes, Francisco Rodríguez Torres. Módulo Laboratorio del Sistema de Información Hospitalaria alas HIS. Junio 2009.

BIBLIOGRAFÍA

Bibliografía

1. Gil, Antonio López. Sistemas de información del laboratorio clínico Cap. 4. Pamplona, España : Sociedad Española de Informática de la Salud, 2004.

2. DÍAZ, MIGUEL E. MARÍN. Fundamentos del Sistema de Salud Pública en Cuba para estudiantes de Informática. p 3. La Habana. Cuba : s.n., 2006.

3. CHEMEUROPE.COM. [En línea] Marzo de 2009. http://www.chemie.de/products/es/52345/.

4. Poceiro, Juan. Wiener lab. LIS : Informática para Laboratorios. Buenos Aires,Argentina : s.n., 2010.

5. Softel. Manual de usuario del Galen Lab. [En línea] Marzo de 2009. http://www.softel.cu/productos.htm.

6. Pelaéz, Juan. Arquitectura basada en capas. [En línea] 26 de Mayo de 2009. http://jtentor.com.ar/post/Arquitectura-de-N-Capas-y-N-Niveles.aspx.

7. Instituto de Física Aplicada del CSIC. ¿Qué es JavaScript? [En línea] Abril de 2009. http://www.iec.csic.es/criptonomicon/java/quesjava.html.

8. Wenz, Christian. Programming ASP.NET AJAX. California, E.E.U.U : s.n., 2007.

9. Nusairat, Joseph Faisal. Beginning JBoss Seam: From Novice to Professional. New York, E.E.U.U : s.n., 2008.

10. Rational Unified Process (RUP). Proven best practices for software and systems delivery. [En línea] Enero de 2010. http://www-01.ibm.com/software/rational/rup/.

11. Yera, Angel Cabo. Estudio Científico de las Arquitectura de Software. Madrid, España : Vision Libros, 2010.

12. [En línea] [Citado el: 5 de enero de 2014.] http://www.access.gpo.gov/cgibin/cfrassemble.cgi?title=200521.

13. [En línea] [Citado el: 5 de enero de 2014.] http: / /www.InfoScienceToday.org.

14. [En línea] [Citado el: 5 de enero de 2014.] http://cap.org.

15. "Microbiología Clínica", en Escolar y Carnicero (Coord.), VI Informe SEIS, El sistema integrado de información clínica. s.l. : Ed. Sociedad Española de Informática de la Salud.Pamplona.

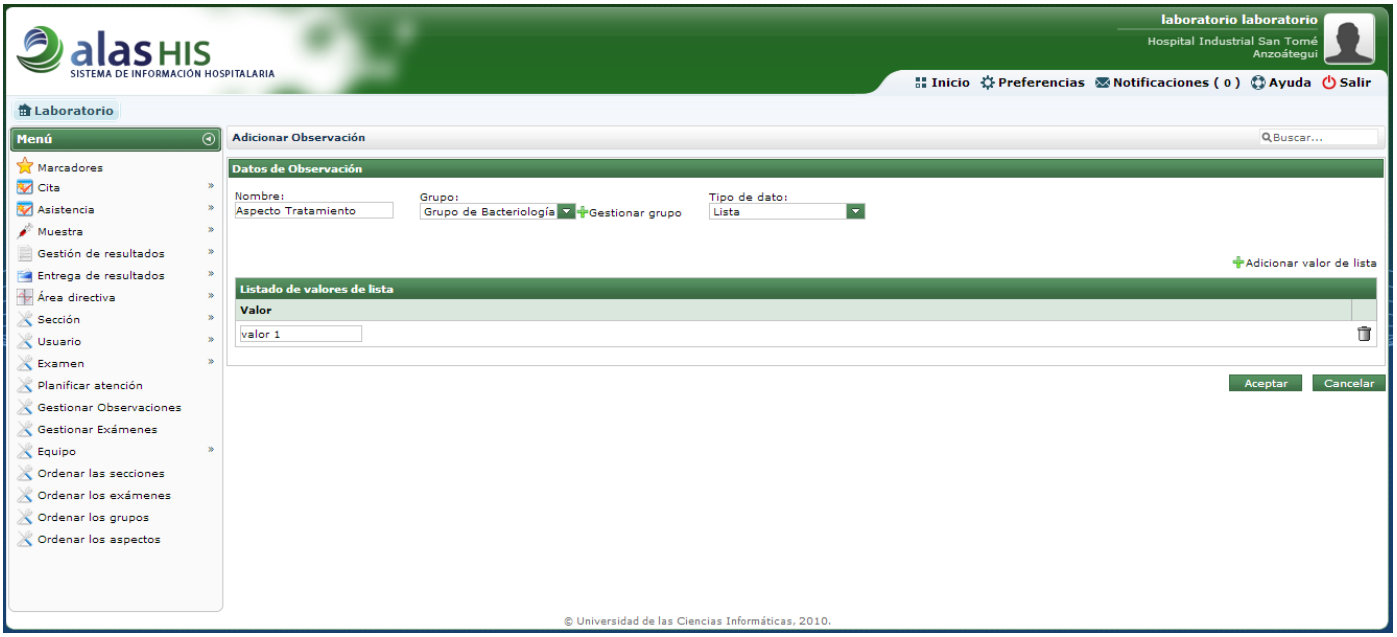
16. López, A. "El laboratorio general: mecanización y gestión", en Escolar y Carnicero (Coord.), VI Informe SEIS, El sistema integrado de información clínica. s.l. : Ed. Sociedad Española de Informática de la Salud. Pamplona., 2004.

17. "Tendencias actuales en los sistemas de información del laboratorio clínico",Todo Hospital, vol. 7. 2000.

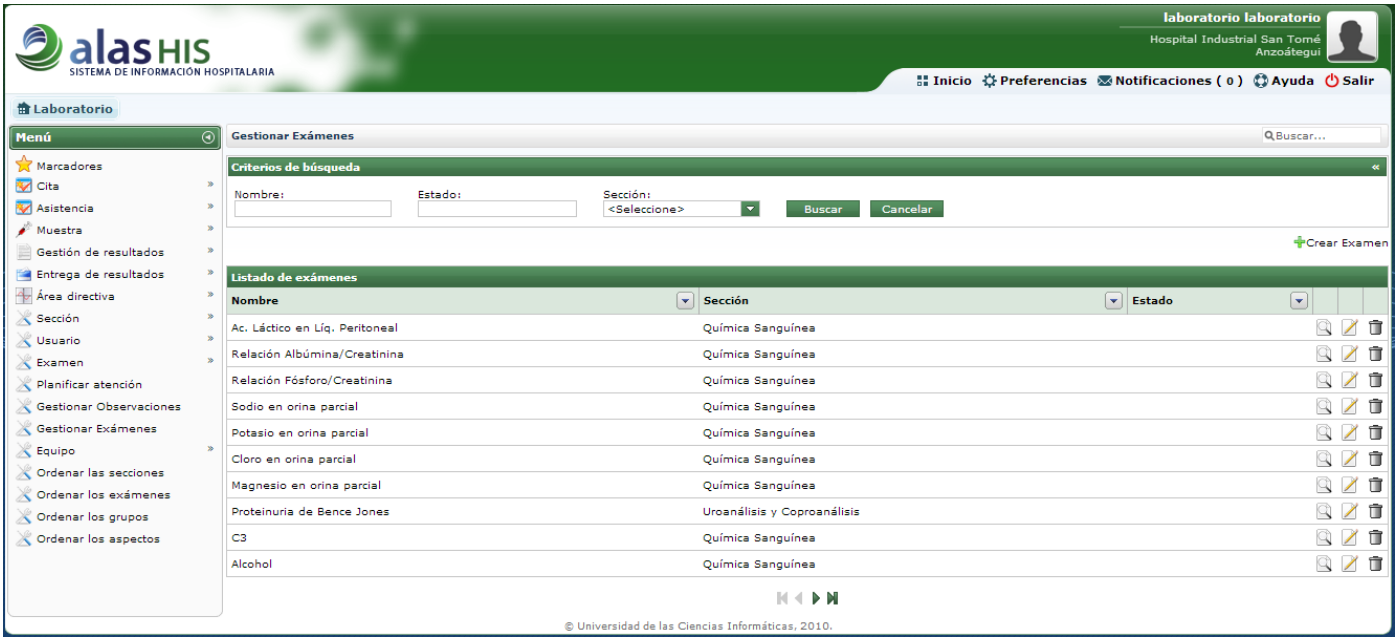
18. "La intranet como soporte del sistema de calidad del laboratorio" en Gestión y calidad total en el laboratorio clínico. . s.l. : Ed. Mapfre. Madrid., 1999.

19. Yasser Manuel Garbey Bermudes, Francisco Rodríguez Torres. Módulo Laboratorio del Sistema de Información Hospitalaria alas HIS. Junio 2009.

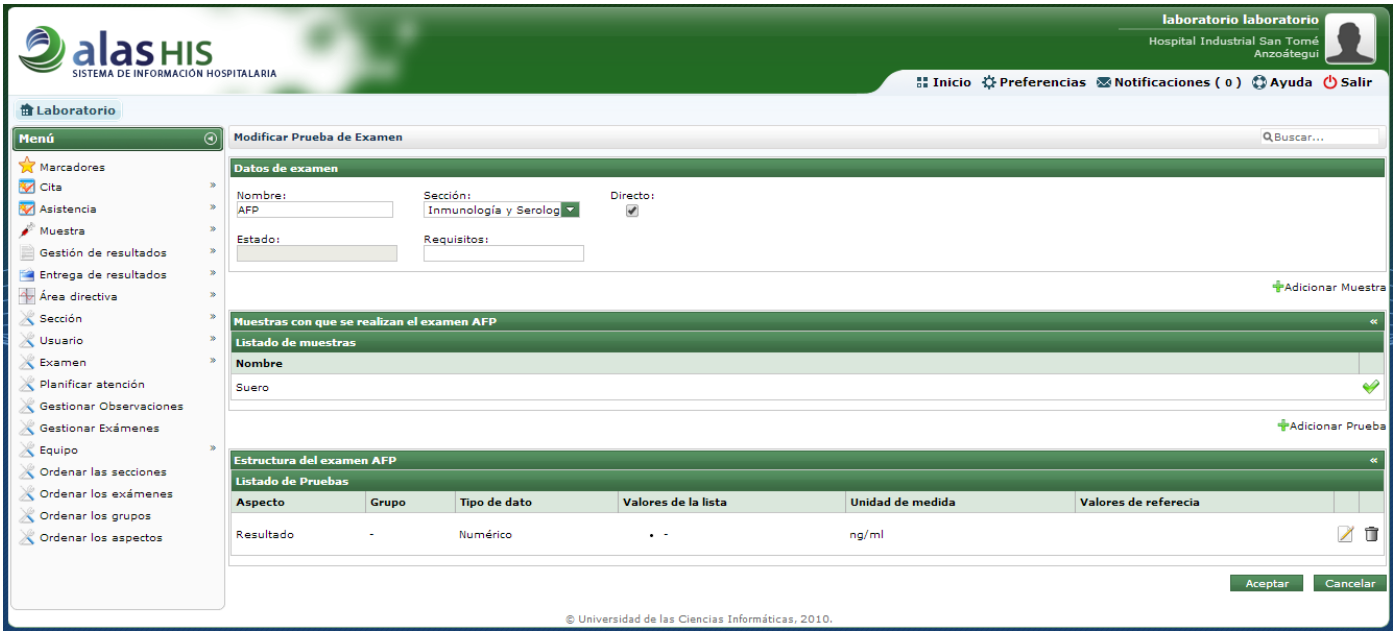
ANEXOS



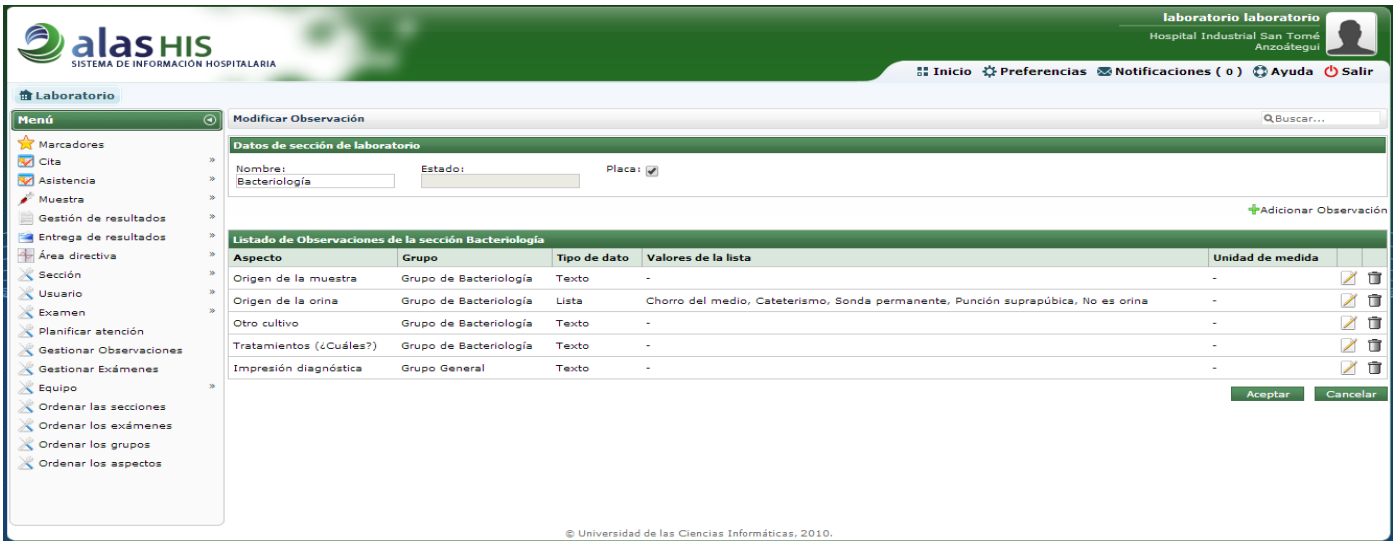
Anexo 1. Imagen de la funcionalidad Adicionar Observación de la aplicación



Anexo 2. Imagen de la funcionalidad Buscar Exámenes



Anexo 3. Imagen de la funcionalidad Modificar Prueba de Examen de la aplicación



Anexo 4. Imagen de la funcionalidad Modificar Observación de la aplicación

GLOSARIO DE TÉRMINOS

Glosario de Términos

Bioanalista: Persona que labora en las distintas áreas del laboratorio.

Examen: Parámetro analítico solicitado por los profesionales de la salud para apoyar o descartar diagnóstico.

Sección: Parámetro en la cual el bioanalista podrá registrar los resultados de los exámenes del paciente.

Framework: Estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado.

Muestra: Pequeña cantidad de una parte del cuerpo humano que le es extraída al paciente en dependencia de los análisis a realizar, por ejemplo, en caso de un análisis de HIV se le extrae sangre.

Tipo de Muestra: Clasificación de las muestras teniendo en cuenta su lugar de origen y sus características individuales. Ejemplo: Esputo, Sangre, Orina, Esperma, Tejido.

Autoanalizadores: Máquina de un laboratorio clínico que mide las diferentes sustancias químicas y otras características en un número de muestras biológicas, con una asistencia humana mínima.