Universidad de las Ciencias Informáticas

Facultad 6

Título: Sistema de Gestión de los Datos Clínicos del Proyecto Mapeo Cerebral Humano Cubano 2.0

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es):

Guillermo León Hernández

Alejandro Claudio Pérez Romeu

Tutor:

Ing. Maikel Laurencio Giralt

Ciudad de La Habana

"Año 52 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos 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 ____ días del mes de _____ del año _____.

Autor(es):

Guillermo León Hernández

_____

Firma del Autor

Tutor:

Ing. Maikel Laurencio Giralt

_____

Firma del Tutor

Alejandro Claudio Pérez Romeu

_____

Firma del Autor

"Solamente aquel que construye el futuro tiene derecho a juzgar el pasado".

Friedrich Nietzsche

DATOS DE CONTACTO

Tutor: Ing. Maikel Laurencio Giralt

Ingeniero en Ciencias Informáticas

E-mail: mlaurencio@uci.cu

AGRADECIMIENTOS

A mis padres que siempre insistieron y lucharon para que terminara esta carrera.

Especialmente a mi madre que es ejemplo de constancia y desinterés en todo lo que hace.

A mis hermanos, especialmente Alexander que a pesar de la lejanía siempre estuvo ahí.

A los amigos de casa por los consejos que al final fueron escuchados.

A los amigos de aquí por sencillamente todo.

A mis compañeros Yosbel, Rodolfo, Jesús que entre todos conformamos una comunidad discutiendo como haríamos para resolver los distintos problemas de nuestras tesis con Spring.

Al tutor: Ing. Maikel Laurencio Giralt cuyos trabajos anteriores nos sirvieron de guía y sus revisiones finales acabaron por perfilar este documento.

A todos los que de una forma u otra contribuyeron a la realización de esta tesis así como a todos los que conspiraron para que yo llegara a este punto en la carrera:

```
for(int i = 0; i< 1000000; i++){
    Out.println ("Gracias por todo");
}
```

DEDICATORIA

A mi familia completa la que está y la que no, son los que contribuyeron a convertirme en ingeniero, en especial a mi hermano Alexander que sirvió de ejemplo en muchas de las cosas que hice. Ojalá hubieran sido más hermanazo.

A mis amigos de afuera y adentro; a la Universidad de las Ciencias Informáticas.

RESUMEN

El Centro de Neurociencia de Cuba lleva a cabo el desarrollo del proyecto Mapeo Cerebral Humano Cubano (MCHC) en conjunto con la Universidad de las Ciencias Informáticas (UCI). En este se realizan un grupo de entrevistas y pruebas clínicas a los sujetos, que conforman el Examen Clínico, cuyo objetivo es determinar la normalidad neuropsiquiátrica del sujeto que formará parte del estudio.

Para dar solución a esta problemática se desarrolló el Sistema para la Gestión de los Datos Clínicos del Proyecto Mapeo Cerebral Humano Cubano (MCHC), aplicación web que permite gestionar con mayor agilidad y correctamente toda la información obtenida en el Examen Clínico, haciendo uso de instrumentos avalados internacionalmente que dan validez al criterio del diagnóstico arrojado por el examen. La aplicación cuenta con una interfaz que facilita la gestión de todos los exámenes clínicos necesarios para incluir un sujeto o persona en el estudio, permitiendo realizar búsquedas, obtener reportes de forma automatizada.

Palabras claves: neuropsiquiátrica, interfaz, gestión.

Introducción

Descubrir las partes que constituyen el complejo cerebro humano, conocer su funcionamiento y detectar sus anomalías es el reto de los investigadores de las Neurociencias. Uno de los métodos más novedosos aplicado con éxito es el de los mapeos cerebrales.

Para escoger los sujetos factibles a incluir en el estudio que realiza el Centro de Neurociencias deben realizarse un conjunto de exámenes, como por ejemplo: examen psicométrico, examen físico neurológico, cuestionario de manualidad, cuestionario de normalidad, cuestionario inicial, electroencefalograma y entrevista MINI neuropsiquiátrica. La aplicación de estos instrumentos avalados internacionalmente permite realizar un pesquisaje neuropsiquiátrico en la muestra del estudio para determinar si los sujetos son funcionalmente sanos.

En el año 2007 se desarrolló como parte de la solución del proyecto Mapeo Cerebral Humano Cubano (MCHC), la aplicación: Examen Clínico 1.0 que gestiona estos exámenes. La aplicación fue desarrollada en lenguaje Personal Home Page (PHP) utilizando una herramienta de mapeo de clases denominada Propel y una arquitectura Modelo Vista Controlador (MVC). (1) Esta aplicación cuenta con un conjunto de inconvenientes: la seguridad fue implementada a partir de sesiones, sin prevenir graves problemas de seguridad que suceden cuando un usuario no autorizado obtiene el identificador de una sesión válida, esta acción resulta fácil ya que estos datos se guardan en los temporales. Además, al acceder a un sitio externo desde la aplicación se envía la sesión con su id. Otro problema es que cualquier usuario con permisos para leer archivos del servidor puede ver el contenido de las sesiones.

Al desarrollarse la aplicación se utilizaron dos librerías javascript: jscalendar y tabpanel, que permite mostrar un popup para escoger la fecha y simular los paneles tabulares de las aplicaciones de escritorio, respectivamente. Estas no son la mejor opción, pues sobrecargan las páginas con scripts cuando existen otras con mejores prestaciones, todas las funcionalidades necesarias incluidas y un mejor diseño gráfico como son jQuery (2) y Ext.js (3). La aplicación utiliza AJAX (1) en funcionalidades que requieren cargar datos dinámicos sin recargar la página, sin embargo, en gran parte de la aplicación se recurrió a técnicas tales como: enviar a un controlador y de este redireccionar a la misma página que es cargada por el navegador, comprometiendo el rendimiento del servidor cuando solo se

actualizó un dato. Existen grandes bloques de código PHP embebido en interfaces generando código "HTML" estático con funciones PHP como "ECHO".

Aunque la aplicación cumplió su función nunca se pudo integrar al resto de los módulos por incompatibilidades, pues estos estaban implementados en Java.

A partir de la situación problémica anteriormente planteada surge el siguiente **problema científico**: ¿cómo integrar el módulo Sistema de Gestión de los Datos Clínicos del proyecto Mapeo Cerebral Humano Cubano al resto de los módulos? El problema planteado se enmarca en el **objeto de estudio** aplicaciones web y **campo de acción,** las aplicaciones web de gestión en la plataforma J2EE.

Como objetivo general se propone desarrollar una aplicación con las funcionalidades del Sistema de Gestión de los Datos Clínicos del proyecto Mapeo Cerebral Humano Cubano versión 1.0, que resuelva los problemas de esta aplicación para lograr la integración con el resto de los módulos.

Del objetivo general se derivan los siguientes **objetivos específicos**:

- Realizar el análisis de las aplicaciones web.
- Realizar una investigación de las técnicas a utilizar para resolver los problemas de la versión 1.0 de Sistema de Gestión de los Datos Clínicos.
- Diseñar de la aplicación del Sistema de Gestión de los Datos Clínicos versión 2.0.
- Implementar la aplicación web para gestionar las funcionalidades del Sistema de Gestión de los Datos Clínicos versión 2.0
- Validar y probar algunos escenarios importantes del sistema.

Para dar cumplimiento a los objetivos específicos se definen las siguientes tareas:

- Análisis del estado del arte de las aplicaciones web con plataforma J2EE.
- Investigación del Sistema de Gestión de los Datos Clínicos versión 1.0
- Estudio y selección de las herramientas a utilizar para desarrollar la aplicación web.
- Obtención de los diagramas de clases del diseño.
- Elaboración de los diagramas de componentes.
- Implementación de cada funcionalidad del Sistema de Gestión de los Datos Clínicos versión 2.0.

- Validar y probar las funcionalidades: Insertar examen de manualidad, Generar reportes y Autenticar usuario.

Capítulo 1: Fundamentación teórica.

Comprende el estudio del estado del arte al mencionar las aplicaciones que la preceden, se presentan y justifican las tecnologías, software y metodologías utilizados en la solución del problema planteado.

Capítulo 2: Características del sistema.

Contiene la descripción general de cómo debe funcionar el sistema y los procesos a automatizar. Se especifican los requisitos funcionales y no funcionales, planteándose los casos de uso y las relaciones con los actores, así como sus descripciones textuales.

Capítulo 3: Diseño del sistema.

Muestra los diagramas de clases del diseño, los patrones utilizados y el modelo de datos.

Capítulo 4: Implementación y Prueba.

Incluye los artefactos correspondientes al flujo de trabajo de implementación como son el diagrama de componentes y el modelo de despliegue. Se describen algunos fragmentos representativos del código fuente de la aplicación y algunas de sus principales interfaces.

Capítulo 1. Fundamentación Teórica

Introducción

En este capítulo se caracteriza el objeto de estudio. Se estudian las tendencias actuales de la utilización de aplicaciones similares a nivel nacional e internacional. Se describen las herramientas, tecnologías y metodologías utilizadas para dar solución al problema, con el objetivo de justificar su empleo en el desarrollo de la presente investigación.

1.1 Aplicaciones web

En la actualidad existe una gran variedad de sitios web. En su interior predomina la información estática y la interacción con el usuario es limitada a búsquedas y algún formulario para registrarse. Entre los sitios web se encuentran: portales de empresas y periódicos online. Cuando el usuario interactúa con el sitio al punto de protagonizar la interacción surge la aplicación web. (4)

Una aplicación web es un software de manejo de información dinámica donde los usuarios están interesados en realizar acciones. El usuario de alguna forma manipula información regularmente de forma persistente, en otras palabras, el usuario espera realizar verdaderamente algo, no se centra en contemplar su contenido. (5)pág. 12

El término"aplicación web" surge cuando el contenido dinámico fue introducido en los sitios web. Una aplicación web es un sitio web cuyo contenido es generado dinámicamente antes de ser enviado al navegador. En la actualidad se desarrollan aplicaciones cada vez más complejas, tanto, que el término requiere un nuevo nombre: RIA (en inglés Rich Internet Application) (6) pág. 13. Una RIA tiene como objetivo parecerse en sus funcionalidades e interfaces a una aplicación de escritorio. Estas tienen la ventaja de poder usar todos los recursos del sistema donde se ejecutan, al contrario de las aplicaciones y sitios web, las cuales solo tienen acceso a los recursos que ofrece el navegador. Para suplir esta desventaja han surgido y surgen infinidad de tecnologías. Ejemplos de ellas son: Flex (7), un grupo de tecnologías de la compañía Adobe (7) , basadas en su producto insignia: Flash (8) y Gears (9), una extensión para Firefox (10) e Internet Explorer (11) que viene nativa en Google Chrome (12) de la compañía Google. Sin embargo, estas tecnologías tienen un gran inconveniente, pues no son soportadas nativamente por los sistemas de la mayoría de los usuarios, de ahí que gran parte de las

RIA en internet utilicen frameworks basados en javascript (13) y Ajax (14) como ExtJs (5), Dojo (15), jQuery (16) y Prototype (17). A continuación se presenta una gráfica comparativa del uso de estos framework obtenida del sitio oficial de Kyle Hayes (18).

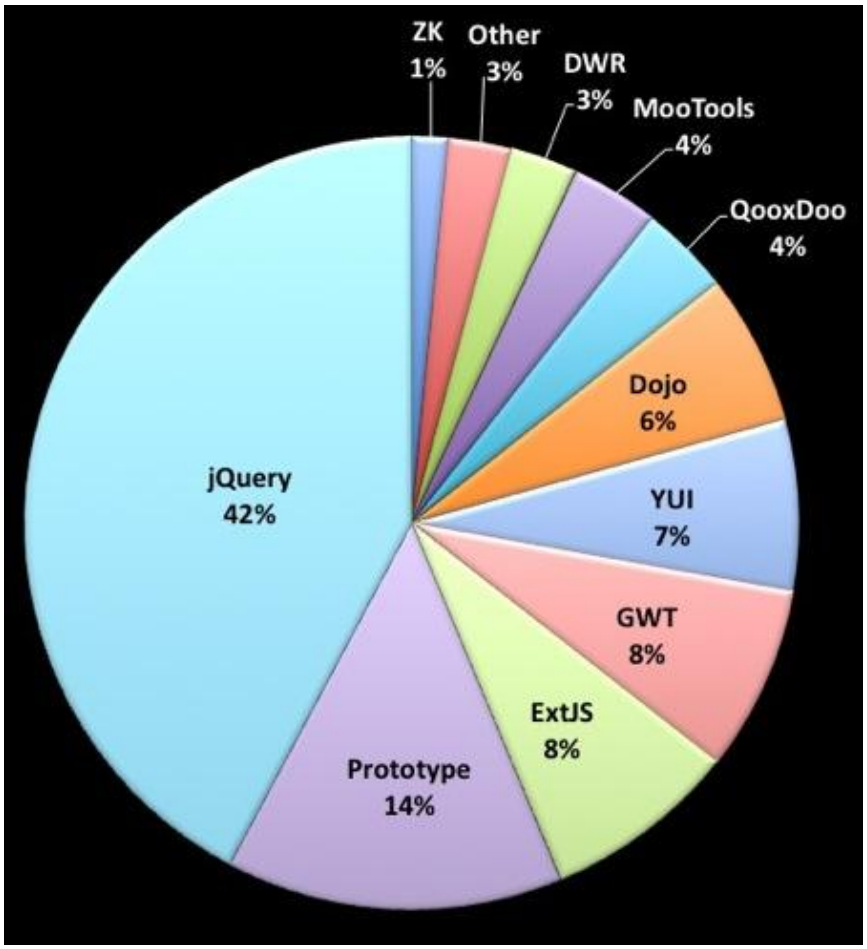

Figura 1 Comparación de frameworks y tecnologías (18)

Detrás de cada aplicación ya sea escritorio, web o RIA, existe una plataforma de desarrollo y un lenguaje de programación. Java (19) ofrece una de las mejores opciones en este campo: la plataforma J2EE de la que se ofrece un pequeño resumen.

1.2 J2EE

La plataforma ***Java 2 Enterprise Edition*** (J2EE) define el estándar para el desarrollo de aplicaciones empresariales multinivel. J2EE simplifica las aplicaciones empresariales basándolas en componentes estandarizados y modulares, proporcionando un conjunto completo de servicios a esos componentes. La plataforma J2EE aprovecha muchas características de la plataforma ***Java 2 Standard Edition*** (J2SE), como la portabilidad; el API JDBC para acceder a la base de datos, la tecnología CORBA para la interacción con los recursos existentes de la empresa, y un modelo de seguridad que protege los datos incluso en aplicaciones de Internet. Sobre esta base, la plataforma ***Java 2 Enterprise Edition*** incluye soporte completo para componentes: Enterprise JavaBeans, Java Servlets, Java Server Pages y la tecnología XML. El estándar J2EE incluye las especificaciones completas y las pruebas de cumplimiento para asegurar la portabilidad de aplicaciones a través de la amplia gama de sistemas empresariales existentes capaces de soportar la plataforma J2EE. Además, la especificación J2EE también se encarga de interoperabilidad para servicios web a través del apoyo para la WS-I Basic Profile. (20)

El lenguaje Java es un gran éxito en muchos aspectos, pues ha logrado traer estandarización donde no la había, tiene en estos momentos una gran industria y una gran comunidad internacional que lo soporta. Hace varios años se convirtió en el lenguaje líder para el desarrollo de aplicaciones empresariales, pues introdujo técnicas para la optimización de servidores donde logró atender varias peticiones sin levantar un proceso por cada una en el servidor, técnicas que después fueron adoptadas por el resto de los desarrolladores de servidores. Actualmente muchos lenguajes alternativos como PHP (21) y Python (22) han ganado protagonismo en el mundo de las aplicaciones web. Esto se ha debido a que Java es complejo, por lo que su desarrollo es costoso en tiempo, pero aun sigue siendo el preferido para realizar aplicaciones que requieren mucho procesamiento en el servidor o una alta seguridad, como bancos y sitios de comercio electrónico. Java y la plataforma J2EE son usados actualmente en aplicaciones empresariales. Algunos sitios importantes a nivel internacional que utilizan java en algunos de sus componentes son: Amazon (23), de comercio electrónico, Gmail (24), de correo electrónico; algunas redes sociales como Hi5 (25) y Facebook (26) además de Google (27) que incluso desarrolla tecnologías que usan el lenguaje como GWT (28). En su totalidad lo utilizan sitios de comercio electrónico como EBay (29). La mayoría de estos sitios utilizan java en las transacciones

bancarias que realizan como parte de su función comercial; en el caso de las redes sociales en applets con funcionalidades para compartir información, por ejemplo la aplicación para gestionar las imágenes de Facebook. En Cuba se pueden mencionar aplicaciones como la del proyecto MINPAL y el módulo de MRI del proyecto Mapeo Cerebral Humano Cubano, ambas desarrolladas en la Universidad de las Ciencias Informáticas (UCI). (30)

1.2 Gestión de Información

La gestión de la información es un proceso que incluye operaciones como: extracción, manipulación, tratamiento, depuración, conservación, acceso y/o colaboración de la información adquirida por una organización a través de diferentes fuentes, gestiona el acceso y derechos de los usuarios sobre la misma. Esta gestión involucra la captura, organización, recuperación, publicación y distribución de la información para percibir y generar conocimiento. (31)

Actualmente los gestores de información juegan un papel importante en el funcionamiento de numerosas instituciones de salud, los servicios o cualquier otra rama donde incursione el ser humano. Esto se debe a la gran cantidad de información que genera cualquier proceso. No resulta factible gestionar esta información de forma manual en enormes volúmenes de documentos, lo cual ha provocado el surgimiento de numerosos gestores de información, que permiten llevar a cabo este proceso con mayor rapidez y exactitud.

Al ser el proyecto Mapeo Cerebral Humano de carácter internacional, todos los países que lo integran poseen aplicaciones para gestionar la organización de esta información, es decir, los resultados de los exámenes, las imágenes obtenidas en electroencefalogramas y las estadísticas del proyecto. En Cuba se desarrolló por este mismo proyecto una primera versión de una aplicación para realizar la gestión.

Para poner en marcha el módulo Examen Clínico del proyecto MCHC es necesario el procesamiento de un gran volumen de información: entrevistas a pacientes, test psicológicos y resultados de pruebas. Para facilitar la gestión de esta información, es necesario desarrollar un sistema que sea capaz de realizar las acciones requeridas. Se utilizó para el desarrollo de la aplicación OpenUP, una metodología ágil que ha demostrado buenos resultados en otros módulos del proyecto y de acuerdo a la línea base de la arquitectura ha sido la escogida para el desarrollo del producto.

1.3 Metodología de Desarrollo de Software

Open Unified Process (OpenUP) es un proceso de desarrollo unificado, basado en RUP, desarrollado por IBM y reconocido mundialmente como uno de los procesos de desarrollo de software de mayor calidad. Se basa en los principios de adaptación, importancia a los involucrados e interesados en los resultados del proyecto, colaboración, valor a la iteración, y calidad continua. (32)

OpenUP/Basic permite un abordaje ágil al proceso de desarrollo de software, con sólo proveer un conjunto simplificado de contenidos, fundamentalmente relacionados con orientación, productos de trabajo, roles y tareas.

OpenUP/Basic es un proceso interactivo de desarrollo de software simplificado, completo y extensible. Utilizado para pequeños equipos de desarrollo que valoran los beneficios de la colaboración y de los involucrados con el resultado del proyecto por encima de formalidades innecesarias.

OpenUP está caracterizado por cuatro principios básicos interrelacionados:

- Prioridades compitiendo por un balance para maximizar el valor para los *stakeholders*.
- Colaboración para alinear los intereses y un entendimiento compartido.
- Evolucionar para obtener continuamente retroalimentación y mejora.
- Enfoque en articular la arquitectura. (32)

OpenUP/Basic se centra en articular la arquitectura para facilitar la colaboración técnica, reducir el riesgo y minimizar el sobreesfuerzo de desarrollo.

OpenUP/Basic procura un equilibrio entre las necesidades de los involucrados con los resultados del proyecto y los costos técnicos, con el fin de maximizar el valor de los involucrados y las guías del proceso de desarrollo.

OpenUP/Basic desarrolla un ciclo de vida interactivo que mitiga el riesgo a tiempo y ofrece demostrar resultados en curso al cliente del proyecto. OpenUP/Basic está basada en RUP y utiliza como lenguaje de modelado UML.

1.4 Lenguaje de Modelado

Se utiliza el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) debido a que la metodología de desarrollo seleccionada lo propone. UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Permite visualizar, especificar, construir y documentar un sistema de software. Ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Está respaldado por el OMG (en inglés Object Management Group). (33)

1.5 Lenguajes de Programación

Siguiendo la línea base de la arquitectura del proyecto MCHC y específicamente en el módulo Examen Clínico, se seleccionan el lenguaje de programación Java y la tecnología JSP, debido a la portabilidad que ofrecen. A continuación se describen estas tecnologías.

1.5.1 Java Server Page (JSP)

Java Server Pages TM (JSP) es un conjunto de tecnologías que permiten la generación dinámica de páginas web combinando código Java (scriptlets) con un lenguaje de marcas como HTML o XML, para generar el contenido de la página.

Como parte de la familia de la tecnología Java, con JSP se pueden desarrollar aplicaciones web independientes de la plataforma. Una característica importante es que permite separar la interfaz del usuario de la generación del contenido dinámico, dando lugar a procesos de desarrollo más rápidos y eficientes.

Adicionalmente, se puede acceder directamente a componentes Java Beans o Enterprise Java Beans (EJB), instanciándolos y estableciendo sus propiedades e invocando sus métodos directamente desde la página JSP. Esto permite desarrollar aplicaciones n-capas donde se separan en lo posible los datos, la lógica del negocio y la lógica de presentación, encapsulando, generalmente en beans, el acceso a los datos. La tecnología JSP es una extensión de la tecnología Servlets, los cuales son aplicaciones 100% Java que corren en el servidor: se crea e inicia un Servlet, se procesan las peticiones recibidas y por último se destruye. Este diseño explica por qué un Servlet remplaza perfectamente a un CGI

(Common Gateway Interface por sus siglas en inglés, es la tecnología clásica que implementan todo tipo de servidores web, por razones de portabilidad se acostumbra a utilizar lenguajes de script como PHP), ya que el servlet se carga una sola vez y está residente en memoria mientras son procesadas las peticiones recibidas generándose así las respuestas a los usuarios.

Cada vez que un cliente solicita al servidor web una página JSP, este pasa la petición al motor de JSP el cual verifica si la página no se ha ejecutado antes o fue modificada después de la última compilación, tras lo cual la compila, convirtiéndola en Servlet, la ejecuta y devuelve los resultados al cliente en formato HTML. La especificación JSP es el producto de una colaboración amplia de varias de las industrias líderes en el desarrollo de software, liderados por Stanford University Network (Sun). Sun hizo la especificación de JSP disponible libremente para la comunidad de desarrollo de software, con la idea de que todos los servidores web soporten JSP, compartiendo la característica de la tecnología Java "Write Once, Run Anywhere" (Escríbelo una vez, córrelo donde quieras). Es conveniente resaltar, que la tecnología JSP es un componente clave de la plataforma Java 2 Enterprise Edition (J2EE) propuesta por Sun Microsystems. (34)

En resumen, las tecnologías JSP y Servlets son una alternativa importante para la programación de web de contenido dinámico que permiten:

- Independencia de la plataforma.
- Rendimiento mejorado.
- Separación de la lógica de la aplicación de la presentación de los datos.
- Uso de componentes (Java Beans).
- Facilidad de administración y uso.
- El respaldo importante de la tecnología sólida Java TM. (35)

1.5.2 Java

Java es un lenguaje orientado a objetos con características muy importantes:

- Es un lenguaje que se compila, generando ficheros de clases compiladas, aunque estas clases compiladas, son en realidad interpretadas por la máquina virtual de Java. Siendo esta la que mantiene el control sobre las clases que se estén ejecutando.

- Es un lenguaje multiplataforma: el mismo código Java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual.

- Es un lenguaje seguro: La máquina virtual, al ejecutar el código Java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.

Gracias al API de Java se puede ampliar el lenguaje para que sea capaz de, por ejemplo, comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas y crear aplicaciones visuales al estilo Windows. (36)

Las características que más interesan del lenguaje al proyecto son las funcionalidades de procesamiento en paralelo que provee. Aunque estas no se utilizan en la aplicación de Examen Clínico, sí son aplicadas en otros módulos del proyecto como el de MRI, que utiliza una grid desarrollada usando estas características. Para lograr la compatibilidad con los restantes módulos se desarrolla esta aplicación.

Java es un lenguaje robusto que verifica su código al mismo tiempo que lo escribe, y una vez más antes de ejecutarse, de manera que se consigue un alto margen de codificación sin errores. Se realiza un descubrimiento de la mayor parte de los errores durante el tiempo de compilación, ya que Java es estricto en cuanto a tipos y declaraciones, y así la rigidez y falta de flexibilidad se convierten en eficacia. Respecto a la gestión de memoria, Java libera al programador del compromiso de tener que controlar especialmente la asignación que de esta hace a sus necesidades específicas. Este lenguaje posee una gestión avanzada de memoria llamada recolector de basura (en Inglés Garbage Collector), y un manejo de excepciones orientado a objetos integrados. (37)

1.6 Servidor de Aplicaciones

Apache Tomcat es el contenedor de servlets, sobre el cual pueden ejecutarse Servlets de Java incluido JSP, principales medios que se utilizarán para la confección de la aplicación.

Tomcat es gratis, fácil de instalar, se ejecuta en máquinas más pequeñas y es compatible con las API más recientes de Java. Puede descargarse, instalarse y probarse en menos de una hora. Ocupa muy poco espacio, teniendo su código binario (toda clase de Java) un tamaño total de apenas dos megabytes, de modo que no es raro que se ejecute tan deprisa. Otra ventaja es que Tomcat es muy fiable. Innumerables empresas lo utilizan (aunque es imposible saber cuántas debido a la falta de licencia comercial). Citando a Linux Torvalds acerca del código libre, "…dado un número suficiente de ojos, todos los errores son irrelevantes…". Lo que significa es que si el número de usuarios es lo bastante grande, siempre habrá alguien capaz de arreglar lo que los demás pueden pensar que es un error muy complejo. Dicho de otra forma, la solidez de Tomcat se basa en que miles de desarrolladores contribuyen con código. Este pone a disposición de todo el mundo las últimas actualizaciones de Java. Además, como puede ejecutarse utilizando la JVM que se quiera, puede utilizarse JDK 1.6. Tiene seguridad de nivel de aplicación, una aplicación de administración intuitiva basada en web, expresiones regulares compatibles con JDK 1.2 y 1.3 y mejor escalabilidad y rendimiento. (38)

1.7 Gestor de Bases de Datos

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) que ha sido desarrollado de varias formas desde la década de 1980. Sus principales características se detallan a continuación.

Funciones

Bloques de código ejecutados en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos brinda, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación, orientación a objetos o la programación funcional.

PostgreSQL soporta funciones que retornan "filas", donde la salida puede tratarse como un conjunto de valores que pueden ser tratados igual a una fila retornada por una consulta (en inglés Query).

Las funciones pueden ser definidas para ejecutarse con los derechos del usuario ejecutor o con los derechos de un usuario previamente definido. El concepto de funciones, en otros Sistemas de Gestión de Base de Datos (DBMS por sus siglas en inglés), son muchas veces referidas como "procedimientos almacenados" (en inglés stored procedures).

Características:

- **Alta concurrencia.**

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

- **Amplia variedad de tipos nativos.**

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indeseables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS. (39)

¿Por qué usar PostgreSQL?

El proyecto MCHC gestionará un gran volumen de información por lo que siguiendo la línea base de la arquitectura se escoge para el desarrollo del sistema al SGBD PostgreSQL por ser un sistema manejador de bases de datos, diseñado para administrar grandes cantidades de datos. Es considerado como una de las bases de datos de código abierto (Open Source) más avanzada del mundo. PostgreSQL se ha preocupado por ser una solución real a los complejos problemas del mundo empresarial y a la vez mantener la eficiencia al consultar los datos. Con ese fin, se han desarrollado y añadido a PostgreSQL las más interesantes y útiles características que antes sólo podían hallarse en sistemas manejadores de bases de datos comerciales como Oracle, DB2 o Sybase. Tiene además un alto rendimiento, estable y capacitado para lidiar con grandes volúmenes de datos. Estas características de PostgreSQL son las que garantizan la integridad de los datos, la velocidad de acceso y consultas a la base de datos.

1.8 Frameworks de Aplicación

El framework puede ser definido como un conjunto de clases (generalmente abstractas), que colaboran entre ellas, para proporcionar un diseño abstracto y solucionar un conjunto de problemas característicos de determinados tipos de aplicaciones. Este captura las decisiones de diseño comunes a un tipo de aplicación, estableciendo un modelo común a todas ellas, asignando responsabilidades y formando colaboraciones entre las clases que forman el modelo.

Un framework de aplicación encapsula una capa de funcionalidad horizontal que puede ser aplicada en la construcción de una gran variedad de programas.

Frameworks de Dominio

Un framework de dominio implementa una capa de funcionalidad vertical, correspondiéndose con un dominio de aplicación o una línea de producto. Su evolución deberá ser también la más rápida, pues deben adaptarse a las áreas de negocio para las que están diseñados.

Ventajas en la utilización de Frameworks

Se logra buen nivel de reutilización (no solo a nivel de código), lo que implica: reducción en el tiempo de desarrollo de nuevos aplicativos, reducción del costo de mantenimiento y mayor nivel de confiabilidad (comparado con escribir código nuevo). En la medida que hay reutilización y el framework se estabiliza gana estandarización y consistencia. Es posible encapsular estándares y mejores prácticas de una compañía en un framework, logrando consistencia y aseguramiento de uso de dichos estándares y mejores prácticas.

Dificultades con los Frameworks

Los frameworks no son reutilizables por sí solos, y cuando se diseña e implementan soluciones con la reutilización en mente, el tiempo y los costos en general, son mayores. Esto debe ser visto como una inversión, ya que en la medida que el framework se reutilice, aparecerán los beneficios. Requieren de programadores más experimentados para su desarrollo, que una aplicación común, así como de buena documentación y entrenamiento para los desarrolladores que lo utilizan. (40)

1.8.1 Spring

Spring es un Framework de aplicación desarrollado por la compañía Interface 21, para aplicaciones escritas en el lenguaje de programación Java. Fue creado gracias a la colaboración de grandes programadores, entre ellos se encuentran como principales partícipes y líderes de este proyecto Rod Johnson y Jürgen Höller. Estos dos desarrolladores, además de otros colaboradores juntando todas sus experiencias en el desarrollo de aplicaciones J2EE (Java 2 Enterprise Editions), incluyendo EJB(Enterprise JavaBeans), Servlets y JSP, lograron combinar estas y otras herramientas en un solo paquete, para brindar una estructura más sólida y un mejor soporte para este tipo de aplicaciones.

Se considera a Spring un framework *lightweight*, es decir liviano o ligero, ya que no es una aplicación que requiere de muchos recursos para su ejecución. El framework completo puede ser distribuido en un archivo de alrededor de 1MB, lo cual representa muy poco espacio, y para la cantidad de servicios que ofrece es relativamente insignificantes su tamaño. Este se encuentra actualmente en su versión 3.0.0, y ha adquiriendo gran auge y popularidad. Una de las características que ayuda a este éxito se debe a que es una aplicación de código abierto, además de que no tiene ningún costo ni se necesita una licencia para utilizarlo, por tanto, da la libertad a muchas empresas y desarrolladores de incursionar en la utilización de esta aplicación. Spring fue creado basado en los siguientes principios:

- El buen diseño es más importante que la tecnología subyacente.
- Los JavaBeans ligados de una manera más libre entre interfaces es un buen modelo.
- El código debe ser fácil de probar. (41)

Spring permite además configurar las clases en un archivo XML y definir en él las dependencias. De esta forma la aplicación se vuelve muy modular y a la vez no adquiere dependencias con Spring. Cuenta con plantillas de utilidades para Hibernate, Ibatis y JDBC, así como la integración con Struts,

JSF y otros frameworks. Es uno de los proyectos más sorprendentes en el panorama actual de Java, en el grado en que ayuda a que los diferentes componentes que forman una aplicación trabajen entre sí, pero no establece apenas dependencias consigo mismo. Esta es la primera característica de este framework. Sería posible retirarlo sin prácticamente cambiar líneas de código.

Lo único que sería necesario es, lógicamente, añadir la funcionalidad que provee, ya sea con otro framework principal o con el código del propio desarrollador. A nivel de soporte de la comunidad, Spring es uno de los proyectos con mayor actividad, con desarrollo dentro y fuera del framework.

Anteriormente se dijo que Spring es un conjunto de librerías o módulos, compuesto por varios frameworks más pequeños:

- Contenedor de inversión de control: la configuración de los componentes de la aplicación y el ciclo de vida es manipulado básicamente por objetos Java comunes.
- Programación orientada a aspectos: el trabajo con las funcionalidades que no pueden ser implementadas mediante la programación orientada a objetos sin hacer sacrificios de rendimiento y estabilidad.
- Framework de acceso a datos: trabaja con sistemas de administración objeto-relacionales que proveen soluciones para los desafíos técnicos reusables en multitudes de ambientes basados en Java.
- Manipulación de transacciones: armonización de varios APIs manipuladores de transacciones orquestados por objetos Java.
- Modelo-Vista-Controlador: Estructura de controladores que gestionan pedidos hechos en la vista utilizando objetos del modelo.
- Framework de acceso remoto: soporte para aplicaciones basadas en el protocolo HTTP, tales como RMI, CORBA y Servicios Web o Web Services (SOAP). (41)
- Spring Security: fundado en 2003 y mantenido activamente por SpringSource, hoy en día se utiliza para asegurar entornos exigentes. Numerosas agencias gubernamentales, aplicaciones militares y bancos centrales lo utilizan. Es liberado bajo una licencia Apache 2.0 por lo que con seguridad se puede utilizar en la aplicación. Spring Security es fácil de aprender, implementar y

administrar. Contiene un espacio de nombres de seguridad que establece directrices para la mayoría de las operaciones comunes, lo que permite garantizar la seguridad de aplicaciones completamente, en tan sólo unas pocas líneas de XML.

Por estas características fue el elegido para resolver los problemas de seguridad en la aplicación subsanando así los problemas de la anterior.

1.8.2 Hibernate

Es una herramienta libre, utilizada para el mapeo objeto-relacional en Java y que facilita el mapeo de atributos entre una Base de Datos (BD) relacional tradicional y el modelo orientado a objetos de una aplicación. Esta potente herramienta convierte los tipos de datos utilizados por Java, en los definidos por SQL (Structured Query Language) y viceversa. Es capaz de generar sentencias SQL para la persistencia y recuperación de los objetos en la BD, liberando al programador de ciertas responsabilidades y reduciendo significativamente el tiempo de desarrollo del software.

Puede ser utilizado tanto en la web como en aplicaciones convencionales. Esto no solo brinda la inversión de tiempo en Hibernate de cara al futuro, sino que, de ser útil, hace a los desarrolladores totalmente independientes del mismo.

1.9 Herramienta CASE

CASE es una sigla y en su traducción al Español significa Ingeniería de Software Asistida por Computación. Se puede definir a las herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software: Investigación preliminar, Análisis, Diseño, Implementación e Instalación. (42)

Las herramientas CASE aseguran que se alcanzan las tres C´s:

- Consistencia
- Complitud
- Cumplimiento de los estándares.

Mejora de la productividad y de la calidad, mediante un entorno interactivo. Aceleración del proceso de desarrollo, favoreciendo el proceso de prototípico (espiral). Automatiza e integra las tareas de las distintas etapas del ciclo de vida. Asistencia en la gestión de proyectos software. Mejora de la calidad del software (automatización comprobación de errores). Automatiza la generación de documentación.

Existen herramientas CASE de trabajos visuales como DBDesigner 4, Rational Rose, Embarcadero ER/Studio, GNU Ferret, MagicDraw, Umbrello, ArgoUML y Visual Paradigm que permiten realizar el modelado del desarrollo de los proyectos. (43)

Visual Paradigm

Es una herramienta CASE profesional para el desarrollo de aplicaciones, que integra el diseño visual, la generación de código fuente, bases de datos y documentación, abarcando todo el ciclo de vida del software. Provee mecanismos para estimar las consecuencias de los cambios y su impacto en los diagramas de análisis, así como la integración con múltiples herramientas de desarrollo. Esta fue la herramienta CASE seleccionada por la línea base de la arquitectura para el desarrollo de la aplicación.

Conclusiones

En el capítulo, fueron descritos de forma breve los principales conceptos y procesos presentes en el módulo. Se proponen además, un conjunto de herramientas para dar solución al problema planteado, entre las que se pueden mencionar a PostgreSQL como gestor de bases de datos y Java como lenguaje de desarrollo. La propuesta, se realizó de acuerdo a la arquitectura definida en el proyecto y especificando las diferentes tecnologías a utilizar en el desarrollo del sistema integrado.

Capítulo 2: Características del Sistema

Introducción

En este capítulo se describen los conceptos de mayor importancia en el entorno donde estará el sistema. Se elabora una lista de requerimientos funcionales y no funcionales que se deben tener en cuenta para la implementación de la herramienta. Se identifican los actores, casos de uso y las relaciones existentes entre ellos. Se identifica el modelo de objetos.

2.1 Requisitos Funcionales

RF 1 Gestionar Cuestionario de Manualidad

RF 1.2 Insertar datos del Cuestionario de Manualidad.

RF 1.3 Obtener automáticamente la manualidad.

RF 1.4 Buscar datos del Cuestionario de Manualidad.

RF 1.5 Visualizar datos del Cuestionario de Manualidad.

RF 1.6 Actualizar datos del Cuestionario de Manualidad.

RF 1.7 Eliminar un Cuestionario de Manualidad.

RF 2 Gestionar Examen Físico-Neurológico

RF 2.1 Insertar datos del Examen Físico-Neurológico.

RF 2.2 Buscar datos del Examen Físico-Neurológico.

RF 2.3 Visualizar datos del Examen Físico-Neurológico.

RF 2.4 Actualizar datos del Examen Físico-Neurológico.

RF 2.5 Eliminar un Examen Físico-Neurológico.

RF 3 Gestionar Examen Psicométrico

RF 3.1 Insertar datos del Examen Psicométrico.

RF 3.2 Buscar datos del Examen Psicométrico.

RF 3.3 Visualizar datos del Examen Psicométrico.

RF 3.4 Actualizar datos del Examen Psicométrico.

RF 3.5 Eliminar un Examen Psicométrico.

RF 3.6 Visualizar los resultados del Examen Psicométrico.

RF 3.7 Graficar automáticamente los resultados del Examen Psicométrico.

RF 4 Gestionar Cuestionario de Normalidad

RF 4.1 Insertar datos personales del Cuestionario de Normalidad.

RF 4.2 Insertar datos clínicos del Cuestionario de Normalidad.

RF 4.3 Buscar datos personales del Cuestionario de Normalidad.

RF 4.4 Visualizar datos personales del Cuestionario de Normalidad.

RF 4.5 Buscar datos clínicos del Cuestionario de Normalidad.

RF 4.6 Visualizar datos clínicos del Cuestionario de Normalidad.

RF 4.7 Actualizar datos personales del Cuestionario de Normalidad.

RF 4.8 Actualizar datos clínicos del Cuestionario de Normalidad.

RF 4.9 Eliminar datos clínicos del Cuestionario de Normalidad.

RF 5 Gestionar Mini Entrevista Neuropsiquiátrica

RF 5.1 Insertar datos de la Mini Entrevista Neuropsiquiátrica.

RF 5.1.1 Obtener automáticamente los diagnósticos de la Mini Entrevista Neuropsiquiátrica.

RF 5.2 Buscar datos de la Mini Entrevista Neuropsiquiátrica.

RF 5.3 Visualizar datos de la Mini Entrevista Neuropsiquiátrica.

RF 5.4 Actualizar datos de la Mini Entrevista Neuropsiquiátrica.

RF 5.5 Eliminar una Mini Entrevista Neuropsiquiátrica.

RF 6 Generar reportes automáticamente del Examen Clínico

RF 6.1 Generar reporte de la cantidad de sujetos con al menos un examen registrado.

RF 6.2 Generar reporte de la cantidad de sujetos excluidos.

RF 6.3 Generar reporte de la cantidad de sujetos no excluidos.

RF 6.4 Generar reporte de la cantidad de sujetos encuestados en cada examen.

RF 6.5 Generar reporte de la cantidad de exámenes realizados por policlínico.

RF 6.6 Generar reporte de la cantidad y por ciento de sujetos excluidos por cada causa de exclusión.

RF 7 Autenticar Usuario

RF 7.1 Comparar nombre y contraseña del usuario con los existentes.

RF 8 Gestionar sujeto

RF 8.1 Insertar datos del sujeto.

RF 8.2 Modificar datos del sujeto.

RF 8.3 Buscar datos de un sujeto.

RF 8.4 Buscar exámenes del sujeto.

RF 8.5 Visualizar datos del sujeto.

RF 9 Gestionar encuestador

RF 9.1 Insertar datos del encuestador.

RF 9.2 Buscar datos del encuestador.

RF 9.3 Visualizar datos del encuestador.

RF 9.4 Modificar datos del encuestador.

RF 9.5 Eliminar un encuestador.

RF 10 Gestionar usuario

RF 10.1 Insertar datos del usuario.

RF 10.2 Buscar datos del usuario.

RF 10.3 Visualizar datos del usuario.

RF 10.4 Modificar datos del usuario.

RF 10.5 Eliminar un usuario.

2.2 Requerimientos no funcionales identificados por la arquitectura.

Apariencia o interfaz externa

El sistema informático contará con un diseño de interfaz sencillo, de fácil entendimiento. Cada una de las páginas de la aplicación cuenta solo con la información requerida para el usuario.

Seguridad

El sistema debe contar con varios niveles de acceso para permitir el trabajo organizado. En el módulo Examen Clínico solo tendrá acceso el técnico y el administrador. Se debe garantizar que la información sensible sólo pueda ser vista por los usuarios con el nivel de acceso adecuado y que las funcionalidades del sistema se muestren de acuerdo al usuario que esté activo.

Usabilidad

El sistema permitirá un acceso fácil y rápido. Puede ser usado por todas aquellas personas que cuenten con conocimientos básicos del manejo de una computadora.

Confiabilidad

El sistema será usado y administrado solamente por trabajadores del Centro de Neurociencias. El mismo validará la entrada de datos para evitar entradas inadecuadas.

Soporte

Una vez instalado el sistema se impartirá un curso de asesoramiento y familiarización con el mismo. Además se hará un seguimiento del funcionamiento del software.

Portabilidad

El sistema es adaptable a diferentes entornos como GNU/Linux o Windows sin necesidad de requerir herramientas adicionales. Además es compatible con las diferentes versiones de los navegadores existentes en los entornos anteriores.

Hardware

Se requiere de una PC cliente de al menos 256 MB de RAM. El servidor debe tener las siguientes características: capacidad de disco duro superior a 80.0 GB, microprocesador superior a 2.0 GHz y 4.0 GB de RAM.

Software

Se deberá disponer, para instalar la aplicación, del sistema operativo Windows 98 o superior, o cualquier distribución de Linux. Las computadoras clientes de los usuarios accederán al sistema usando Internet Explorer o Mozilla Firefox. Para el servidor de la aplicación el sistema operativo recomendado es GNU/Linux. Se debe instalar un servidor web Apache Tomcat y para la base de datos del sistema se usará PostgreSQL.

Restricciones en el diseño y la implementación

La lógica de presentación constituirá una capa independiente de la lógica de negocio, centrando su función en la interfaz de usuario y validaciones de los datos de entrada. Se usará el lenguaje de programación Java y el gestor de bases de datos PostgreSQL.

Legales

Se estarán usando herramientas de software libre, licencia GNU/GPL y herramientas de las que se posea licencia.

2.3 Actores del sistema

Pueden ser las personas, sistemas o hardware externo que se relacionan o interactúan con el sistema. (44) En la tabla se muestran los actores del sistema identificados durante la investigación.

Actor	Descripción
Administrador	Es el encargado de gestionar los usuarios, sujetos y sus exámenes.
Técnico	Es quien inserta y gestiona los sujetos en la aplicación, junto a sus exámenes.

2.4 Diagrama de Casos de Uso del Sistema

Un diagrama de casos de usos del sistema representa gráficamente a los procesos y su interacción con los actores del sistema.

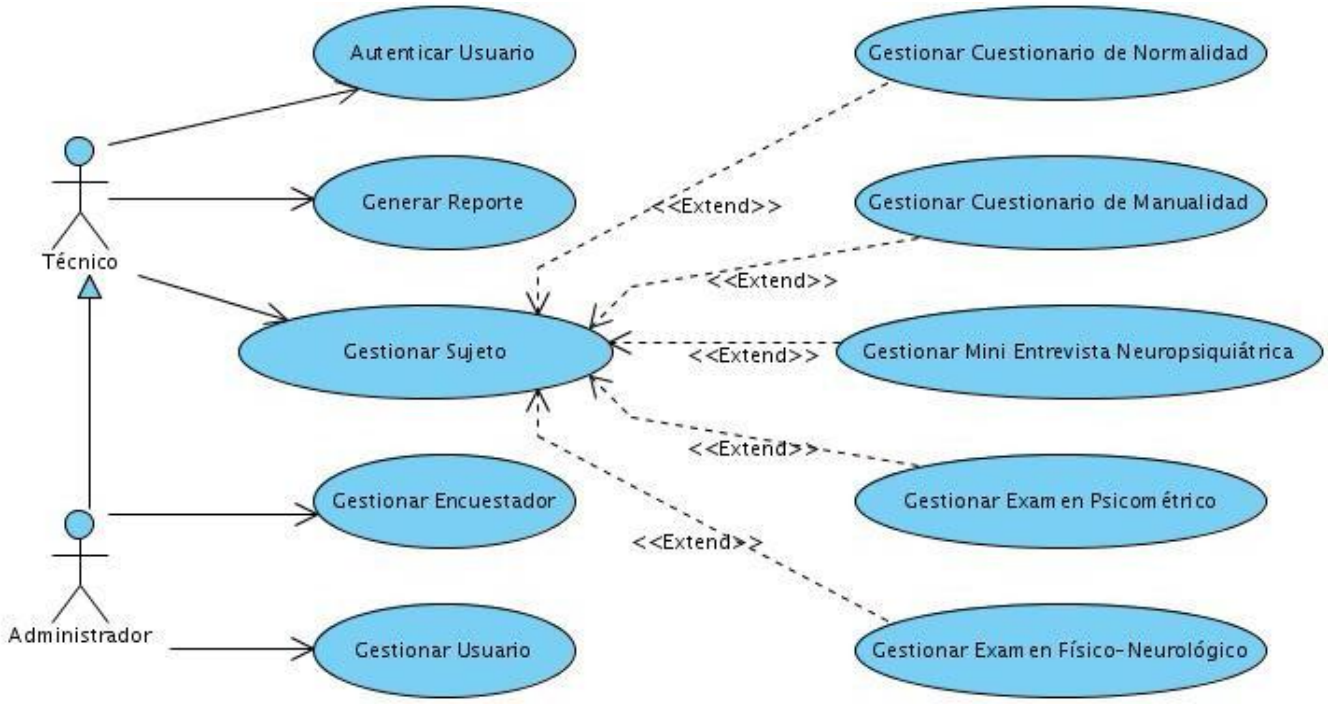

Figura 2 Modelo de Casos de Usos del Sistema

En el diagrama se muestran los casos identificados en el sistema así como los actores que los inician. Obsérvese que la gestión de los exámenes comienza en todos los casos por la gestión del sujeto. Se identificaron dos roles, el administrador gestiona sujetos y usuarios. El técnico no tiene acceso a estas páginas en la aplicación.

2.5 Descripción textual de los casos de usos del sistema

2.5.1 Descripción Textual Caso de Uso Gestionar Sujeto

Nombre del caso de uso	Gestionar Sujeto

Actores	Técnico
Propósito	Llevar a cabo una serie de operaciones con los sujetos presentes en el estudio como son: insertar, buscar, visualizar, un sujeto, insertarlo o modificarlo.
Resumen	El caso de uso se inicia cuando el técnico o el administrador va a realizar alguna de las siguiente operaciones: 1. Insertar Sujeto. 2. Buscar Sujeto. El sistema muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación
Referencias	RF 8
Precondiciones	Que el técnico o el administrador estén autenticados en la aplicación.
Pos condiciones	El sistema inserta un nuevo sujeto o muestra los datos de un sujeto solicitado, al cual se le podrán actualizar sus datos.
Requerimientos especiales	-

Flujo normal de los eventos	
Acción del actor	Respuesta del sistema
1. El técnico, quiere realizar una de las siguientes operaciones: 1.1 Insertar Sujeto. 1.2 Buscar Sujeto	2. El sistema, en dependencia de la operación que se solicita realizar, hace lo siguiente: Si decide Insertar los datos referentes al Sujeto, ir a la sección: Insertar Sujeto. Si decide Buscar un Sujeto ya existente, ir a la sección:

	Buscar Sujeto.
Sección "Insertar Sujeto"	
Acción del actor	Respuesta del sistema
	1. El sistema muestra la interfaz correspondiente a la inserción de un sujeto.
2. Inserta los datos correspondientes: Datos del Sujeto Código Nombre Apellido(1) Apellido (2) CI Policlínico Teléfono Sexo Manualidad(Derecho, Zurdo, Ambidextro) Color de Piel(Blanca, Negra, Mulata, Otras) Peso (lbs) Talla (cms) Temperatura (°C) Nivel de Escolaridad Vencido Ingreso Per Cápita (suma total	3. El sistema verifica que todos los campos estén correctos y que los siguientes datos tengan valor. Datos del Sujeto Código Nombre Apellido(1) Apellido (2) CI Policlínico Edad Dirección Actual Provincia Municipio Policlínico

de los salarios dividido por número de personas en la vivienda) Fecha última extracción de Sangre Status Social Comentario Datos de Nacimiento Fecha Edad Provincia Municipio Dirección Actual Dirección Provincia Municipio	
Indica insertar los datos	
	4. El sistema verifica existencia del sujeto.
	5. El sistema registra al sujeto dando la posibilidad de insertar sus exámenes y ver sus datos personales.
6. El técnico marca una de las siguientes opciones: Exámenes no Realizados Examen psicométrico Cuestionario de normalidad	7. El sistema en correspondencia con la operación escogida por el técnico realiza lo siguiente: Si selecciona en los Examen sin realizar Examen psicométrico: Ir al Caso de Uso “Gestionar Examen psicométrico” , a la Sección

(Datos Personales) MINI Entrevista Neuropsiquiátrica Internacional Cuestionario de normalidad (Datos Clínicos) Cuestionario de manualidad Examen Físico-Neurológico Mostrar Datos del Sujeto.	Insertar Examen psicométrico. Cuestionario de normalidad (Datos Personales): Ir al Caso de Uso "Gestionar Cuestionario de normalidad", a la Sección Insertar Cuestionario de normalidad. MINI Entrevista Neuropsiquiátrica Internacional: Ir al Caso de Uso "Gestionar MINI Entrevista Neuropsiquiátrica Internacional" a la Sección Insertar MINI Entrevista Neuropsiquiátrica Internacional. Cuestionario de Normalidad (Datos Clínicos): Ir al Caso de Uso "Gestionar Cuestionario de Normalidad", a la Sección Insertar Cuestionario de Normalidad. Cuestionario de Manualidad: Ir al Caso de Uso "Gestionar Cuestionario de Manualidad" a la Sección Insertar Cuestionario de Manualidad. Examen Físico-Neurológico: Ir al Caso de Uso "Gestionar Examen Físico-Neurológico" a la Sección Insertar Examen Físico-Neurológico. Si selecciona Mostrar Datos del Sujeto Ir a la Sección. Visualizar Datos del Sujeto.
Flujo alterno de la Sección Insertar Sujeto	
2.1 Si el técnico no desea insertar el sujeto sale de la sección.	4.1 Si el campo de código está sin valor o no está correcto se activa un mensaje de alerta notificando el error y se ejecuta a la acción 2 de la Sección Insertar Sujeto.
6.1 Si el técnico no desea realizar	5.1 Si el sujeto existe el sistema muestra un mensaje de

ninguna de las operaciones sale de la sección.	alerta. Se ejecuta la acción 2 de la Sección Insertar Sujeto.
Sección Buscar Sujeto	
	1. El sistema muestra la interfaz correspondiente a la búsqueda del sujeto.
2. El técnico selecciona el tipo de búsqueda e inserta el dato a buscar. Código del sujeto.	3. El sistema verifica que el parámetro seleccionado tenga valor y que esté correcto.
	4. El sistema realiza la búsqueda del sujeto que cumpla con el parámetro indicado.
	5. El sistema muestra el sujeto dando la posibilidad de ver los exámenes realizados, los exámenes no realizados y sus datos personales.
6. El técnico marca una de las siguientes opciones: Exámenes Realizados Examen psicométrico Cuestionario de Normalidad(Datos Personales) MINI Entrevista Neuropsiquiátrica Internacional Cuestionario de Normalidad(Datos Clínicos) Cuestionario de Manualidad Examen Físico-Neurológico	7. El sistema en correspondencia con la operación escogida por el técnico realiza lo siguiente: Si selecciona en los exámenes realizados. Examen psicométrico: Ir al Caso de Uso Gestionar Examen psicométrico a la Sección Buscar Examen psicométrico Cuestionario de normalidad(Datos Personales): Ir al Caso de Uso Gestionar Cuestionario de Normalidad a la Sección Buscar Cuestionario de Normalidad MINI Entrevista Neuropsiquiátrica Internacional: Ir al Caso de Uso Gestionar MINI Entrevista Neuropsiquiátrica Internacional a la Sección Buscar MINI Entrevista Neuropsiquiátrica Internacional Cuestionario de Normalidad (Datos Clínicos): Ir al Caso de

Exámenes no Realizados Examen psicométrico Cuestionario de Normalidad(Datos Personales) MINI Entrevista Neuropsiquiátrica Internacional Cuestionario de Normalidad(Datos Clínicos) Cuestionario de Manualidad Examen Físico-Neurológico	Uso Gestionar Cuestionario de Normalidad a la Sección Buscar Cuestionario de Normalidad Cuestionario de Manualidad: Ir al Caso de Uso Gestionar Cuestionario de Manualidad a la Sección Buscar Cuestionario de Manualidad. Examen Físico-Neurológico: Ir al Caso de Uso Gestionar Examen Físico-Neurológico a la Sección Buscar Examen Físico-Neurológico. Si selecciona en los Exámenes sin realizar Examen psicométrico: Ir al Caso de Uso Gestionar Examen psicométrico , a la Sección Insertar Examen psicométrico. Cuestionario de normalidad (Datos Personales): Ir al Caso de Uso Gestionar Cuestionario de normalidad, a la Sección Insertar Cuestionario de normalidad. MINI Entrevista Neuropsiquiátrica Internacional: Ir al Caso de Uso Gestionar MINI Entrevista Neuropsiquiátrica Internacional a la Sección Insertar MINI Entrevista Neuropsiquiátrica Internacional. Cuestionario de normalidad (Datos Clínicos): Ir al Caso de Uso Gestionar Cuestionario de Normalidad, a la Sección Insertar Cuestionario de Normalidad. Cuestionario de Manualidad: Ir al Caso de Uso Gestionar Cuestionario de Manualidad a la Sección Insertar Cuestionario de Manualidad. Examen Físico-Neurológico: Ir al Caso de Uso

	Gestionar Examen Físico-Neurológico a la Sección Insertar Examen Físico-Neurológico. Si selecciona Mostrar Datos del Sujeto se ejecuta la Sección Visualizar Datos del Sujeto
Flujo Alterno 1 de la Sección Buscar Sujeto	
2.1 El técnico selecciona el tipo de búsqueda e inserta el dato a buscar. Nombre del sujeto.	2.2 El sistema verifica que el parámetro seleccionado tenga valor y que esté correcto.
	2.3 El sistema realiza la búsqueda del sujeto que cumpla con el parámetro indicado.
	2.4 El sistema muestra el o los sujeto encontrados dando la posibilidad de seleccionar un sujeto para ver los exámenes realizados, los exámenes no realizados y sus datos personales.
	2.6 Se ejecuta la acción 5 de la Sección Buscar Sujeto.
2.5 El técnico selecciona el sujeto e indica ver los exámenes realizados, los exámenes no realizados y sus datos personales.	4.1 Si el campo de código está sin valor o no está correcto se activa un mensaje de alerta notificando el error y se ejecuta a la acción 2 de la Sección Buscar Sujeto.
Flujo Alterno 2 de la Sección Buscar Sujeto	
	2.3.1 Si el campo de código está sin valor o no está correcto se activa un mensaje de alerta notificando el error y se ejecuta a la acción 2 de la Sección Buscar Sujeto.

	2.4.1 Si el sujeto no existe el sistema muestra un mensaje de alerta. Se ejecuta la acción 2 de la Sección Buscar Sujeto.
Sección Visualizar Datos del Sujeto	
	1. El sistema muestra los datos del sujeto dando la posibilidad de que estos puedan ser modificados.
2. El técnico realiza los cambios necesarios y/o actualizaciones pertinentes. Código Nombre 1er Apellido 2do Apellido CI Datos de nacimiento Fecha Edad actual Provincia Municipio Dirección actual Dirección Provincia Municipio Policlínico Teléfono Datos personales	3. El sistema verifica que los campos necesarios tengan valor y que todos estén correctos. Campos necesarios: *Datos del Sujeto* Código Nombre Apellido(1) Apellido (2) CI Policlínico Edad Dirección Actual Provincia Municipio Policlínico

Sexo Manualidad Referida Color de la piel Peso Talla Temperatura Nivel de escolaridad vencido Ingreso per cápita Fecha de última extracción de Sangre Estatus Social Comentario	
Flujo Alterno de la Sección Visualizar Datos del Sujeto	
2.1 El técnico no desea modificar los datos del sujeto y sale de la sección.	4.1 Si los campos necesarios no tienen valor o existen campos con errores se muestra un mensaje de alerta. Se ejecuta la acción 2 de la Sección Visualizar Datos del Sujeto.

2.5.2 DESCRIPCIÓN TEXTUAL CASO DE USO AUTENTICAR USUARIO

Nombre del caso de Uso	Autenticar Usuario
Actores	Técnico
Propósito	Entrar al sistema para realizar cierta operación.
Resumen	El Caso de Uso se inicia cuando el usuario introduce los datos que se le piden para acceder a la aplicación, estos se verifican y finaliza dándole los permisos según los roles y habilitándole la entrada. El caso de uso termina cuando el

	usuario entra al sistema
Referencias	RF 7.1
Precondiciones	
Pos condiciones	Se habilitan las funcionalidades según lo privilegios.
Requerimientos Especiales	-

Estos son dos de los casos de uso representativos del proyecto, el resto se pueden encontrar en los anexos de este documento.

Conclusiones.

Se presentaron los conceptos más importantes dentro del contexto del sistema. Se identificaron 56 requisitos funcionales, agrupados en 10 casos de uso. Se representaron los casos de uso y los actores del sistema y se describieron los casos de uso.

Capítulo 3: Diseño del Sistema

Introducción

En este capítulo se confeccionan una serie de artefactos correspondientes al flujo de trabajo de diseño, donde se describe cómo se va a implementar el sistema, entre ellos se encuentran los diagramas de clases del diseño por cada caso de uso así como los diagramas de secuencia correspondientes. Se describe la arquitectura utilizada y los principales patrones arquitectónicos.

3.1 Patrones de arquitectura y patrones de diseño

Christopher Alexander expresó: "Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma". (45)

Los patrones ayudan a construir sobre la experiencia colectiva de ingenieros de software experimentados. Estos capturan la experiencia existente y que ha demostrado ser exitosa en el desarrollo de software, y ayudan a promover las buenas prácticas de diseño. Cada patrón aborda un problema específico y recurrente en el diseño o implementación de un sistema de software. (46)

El objetivo principal que persiguen los patrones es crear un lenguaje común entre los diseñadores y desarrolladores para comunicar experiencias sobre los problemas y sus soluciones.

Durante el diseño del sistema se utilizó el patrón Modelo-Vista-Controlador o Separación Modelo-Vista como también se le conoce. Este patrón de arquitectura separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Modelo: Representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.

Vista: Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

La solución que aporta este patrón es definir las clases de dominio (modelo) para que no tengan acoplamiento ni visibilidad directa respecto a las clases ventana (vista) y para que los datos de la aplicación y de la funcionalidad se conserven en la clase de dominio, no en las de ventana. (47)

Entre las razones por las cuales se emplea MVC se pueden citar:

- Permitir desarrollar independientemente el modelo y las capas de interfaz para el usuario.
- Permitir conectar fácilmente otras vistas a la capa actual del dominio, sin que esto la afecte.
- Permitir vistas simultáneas y múltiples del mismo objeto modelo.
- Permitir ejecutar la capa del modelo independiente de la capa de interfaz para el usuario.
- Permitir transportar fácilmente la capa de modelo a otro esquema de interfaz para el usuario. (47)

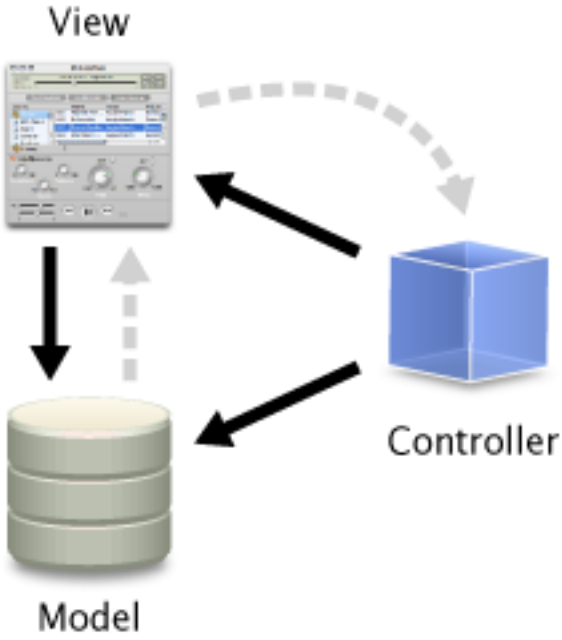

Figura 3 Patrón Modelo Vista Controlador

Ventajas: Brinda soporte de vistas múltiples, es decir, que la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente, debido a que la vista se halla separada del modelo y no existe dependencia directa entre ellos.

3.1.2 Patrones de diseño

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.

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Entre los patrones de diseño definidos por Spring y utilizados en el desarrollo de la aplicación se encuentran:

Service Locator

La localización de los servicios se realiza mediante inyección utilizando el contenedor IoC de Spring.

Front Controller

Todas las peticiones pasan a través del controlador frontal de Spring DispatcherServlet.

Composite Entity

Implementado por Hibernate siendo totalmente transparente. Mediante la definición del Mapping de Hibernate se realiza la composición de la entidad.

Data Acces Object

Todo el acceso a datos se realiza a través de los DAOs. La arquitectura impide el acceso a la propia sesión de Hibernate, si no se realiza desde uno de ellos. (48)

3.1.3 Patrones de asignación de responsabilidades (GRASP)

Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería del software. Todo lo contrario; intentan codificar el conocimiento, las expresiones y los principios ya existentes: cuanto más trillados y generalizados, tanto mejor. En consecuencia, los patrones GRASP no introducen ideas novedosas; son una mera codificación de los principios básicos más usados.

Los patrones son parejas de problema/solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (47)

GRASP es un acrónimo que significa "General Responsibility Assignment Software Patterns" (patrones generales de software para asignar responsabilidades) y la aplicación de sus principios son de vital importancia si se quiere diseñar eficazmente el software orientado a objetos.

Existen 5 patrones GRASP básicos, el Experto, Creador, Alta Cohesión, Bajo Acoplamiento y Controlador.

Experto

Consiste en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Beneficios: Con la aplicación de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.

El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase "sencillas" y más cohesivas que son fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

En la figura se muestra la clase Datos. Esta contiene la información necesaria para procesar los resultados del examen psicométrico, además de los métodos para hacerlo; por lo que se evidencia el uso del patrón Experto.

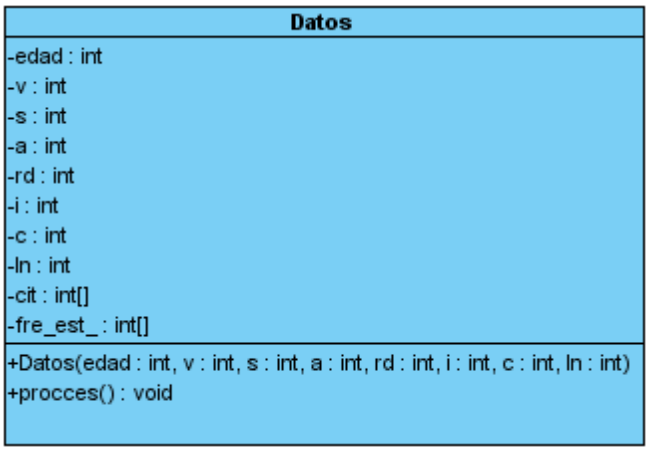

Figura 4 Ejemplo Patrón Experto

Clase del paquete "*proccesPsicometrico*" que se encarga del procesamiento de los datos del examen, esta clase es experto pues no necesita interactuar con ninguna otra para procesar el examen y brindar la información.

Creador

La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella.

El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización. El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental del mismo es encontrar un creador que se debe conectar con el objeto

producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. Asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A o B utiliza especialmente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).
- B es un creador de los objetos A.
- Si existe más de una opción, prefiera la clase B que agregue o contenga la clase A.

A continuación se muestra un ejemplo del patrón creador en la forma de relación que se establece en dos de las clases del sistema.

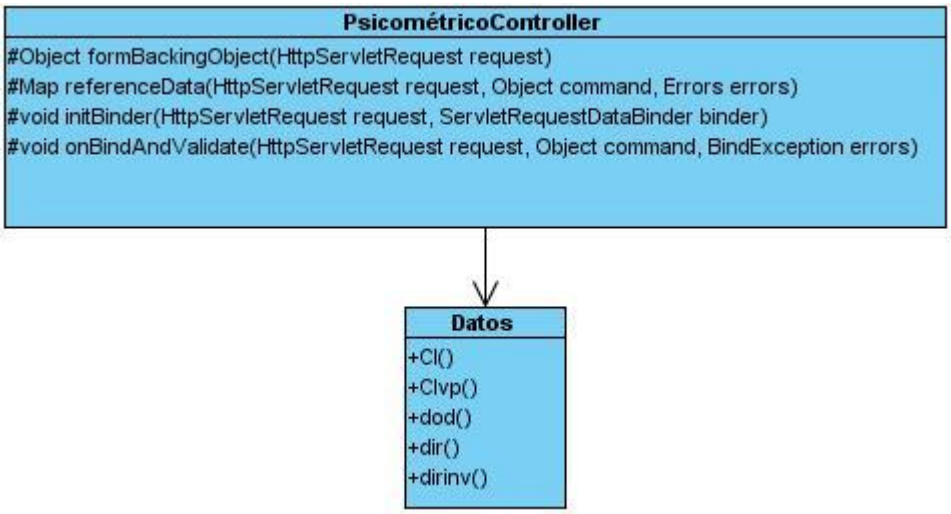

Figura 5 Ejemplo Patrón Creador

El controlador de Psicométrico construye una instancia de Datos, clase encargada de contener y calcular los datos para la generación de los resultados del Examen Psicométrico.

Controlador

Consiste en asignar las responsabilidades del manejo de eventos a una clase intermedia entre la interfaz que produce los eventos y el sistema que debe responder contribuyendo de esta forma a desacoplar el sistema. Hay dos formas de implementar como un controlador del sistema en general que maneja todos los eventos y como controladores independientes que atienden distintos casos de uso.

Beneficios

Mayor potencial de los componentes reutilizables. Garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la interfaz.

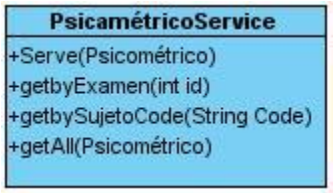

Figura 6 Ejemplo Patrón Creador

El controlador PsicométricoService controla las operaciones de persistencia sobre el objeto Psicométrico.

Bajo Acoplamiento

Consiste en asignar responsabilidad para mantener bajo acoplamiento. Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. El uso de la inyección de dependencias en el framework Spring esta dirigido a potenciar este patrón. Las clases no instancian directamente a otras clases sino que reciban la instancia que necesitan desde el controlador frontal.

Beneficios:

No se afectan por cambios de otros componentes fáciles de entender por separado y fáciles de reutilizar.

Alta Cohesión

Consiste en asignar una responsabilidad de modo que la cohesión siga siendo alta. Grady Booch señala que se da una alta cohesión funcional cuando los elementos de un componente (clase, por ejemplo) "colaboran para producir algún comportamiento bien definido". (49) Se encuentra en el uso de un controlador para atender una petición específica de una forma bien definida.

Beneficios:

- Mejoran la claridad y la facilidad con que se entiende el diseño.
- Se simplifica el mantenimiento y las mejoras en funcionalidad.
- A menudo se genera un bajo acoplamiento.
- La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico.

3.2 Diagramas de clases del diseño

Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones (incluyendo herencia, agregación, asociación, etc.). Los diagramas de clases son el pilar básico del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido (diseño). El diagrama de clases de más alto nivel (main class diagram), será lógicamente un dibujo de los paquetes que componen el sistema. A su vez cada paquete tendrá un main class diagram que muestra las clases del paquete. Debido al uso del framework Spring se introduce para el diseño el patrón arquitectónico Modelo-Vista-Controlador (MVC), es por esta razón que los diagramas de clases del diseño correspondientes representan las interacciones entre las clases del sistema, cumpliendo con el patrón arquitectónico definido. Este diseño está compuesto por tres paquetes (Modelo, Vista y Controlador), donde se encuentran las clases controladoras, páginas clientes y clases entidades. Las peticiones realizadas por los usuarios son atendidas por el controlador springmvc-servlet quien delega a otro componente de Spring el procesamiento de la solicitud; quien determina cuál será el controlador que recibirá la petición del usuario. El controlador definido se encarga de recuperar la información necesaria mediante comunicaciones que establece con las diferentes clases como las que prestan servicios, las clases de

acceso a datos y las clases entidades contenidas en el modelo. Luego este notifica al controlador springmvc-servlet para que construya las páginas clientes con los datos requeridos. (50)

En la figura 7 se muestra el diagrama de clases del diseño de Gestionar Examen Psicométrico. El usuario accede por un enlace en la página CP_Exámenes al DispatcherServlet del subsistema Spring, que se encarga de redireccionar al controlador de la Examen PsicométricoController, que construye la vista Psicométrico con los datos del modelo obtenidos a través del PsicométricoService. Si el usuario decide guardar, entonces el mismo controlador se encarga de persistir los cambios. Si el usuario decide consultar resultados, entonces el propio controlador se encarga de construir la vista resultados, que hace uso del PsicométricoImgController que utiliza el subsistema JfreeChart, que es una librería para la generación de gráficas con licencia GNU/GPL.

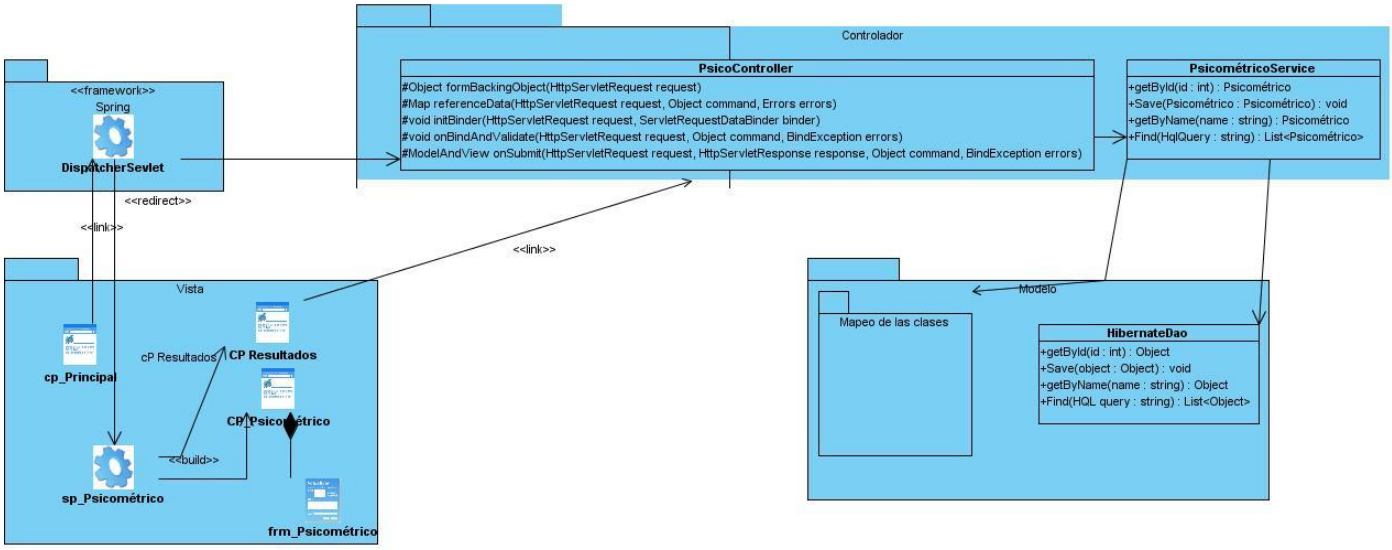

Figura 7 Diagrama de Clases del Diseño: CU Gestionar Psicométrico

En la figura 8 se muestra el diagrama de clases del diseño de Gestionar Sujeto. El usuario accede por un enlace en la página CP_Principal al DispatcherServlet del subsistema Spring, que se encarga de redireccionar al controlador del gestionar Sujeto SujetoController, que construye la vista Sujeto con los datos del modelo obtenidos a través del SujetoService. Si el usuario decide guardar, entonces el mismo controlador se encarga de persistir los cambios.

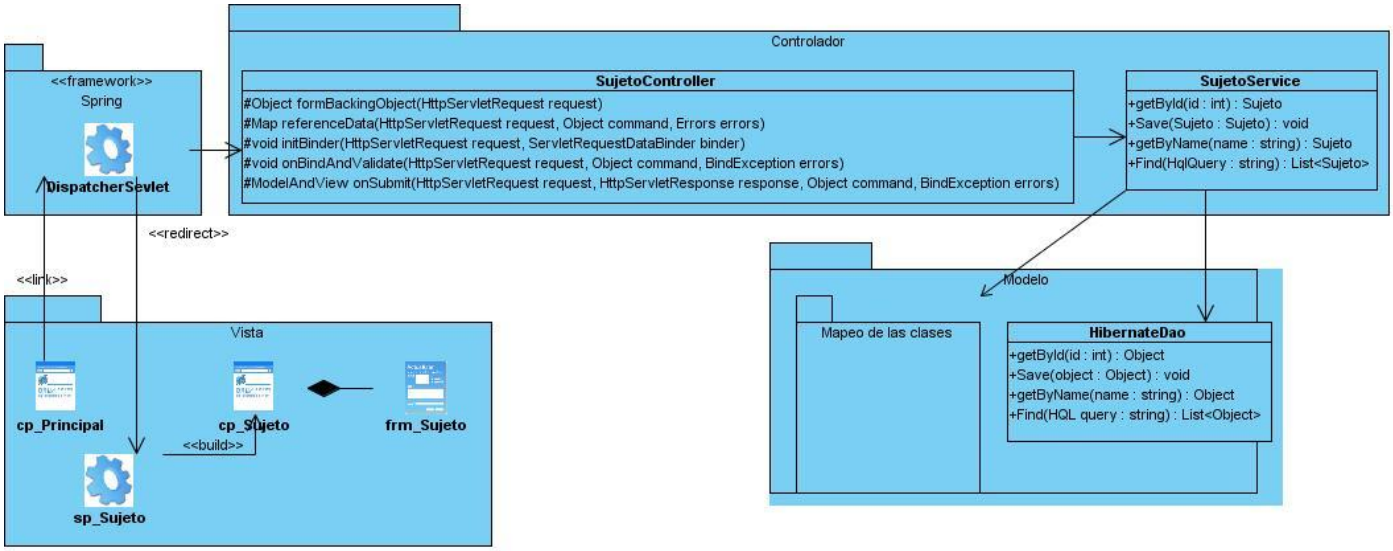

Figura 8 Diagrama de Clases del Diseño: CU Gestionar Sujeto

En la figura 15 se muestra el diagrama de clases del diseño de Autenticar Usuario. El usuario entra los datos en la interfaz de entrada que son enviados al controlador AutenticarController, este comprueba los datos utilizando UsuarioService y redirecciona a la entrada del sistema: Entrada o interfaz inicio: Inicio, según sea el caso.

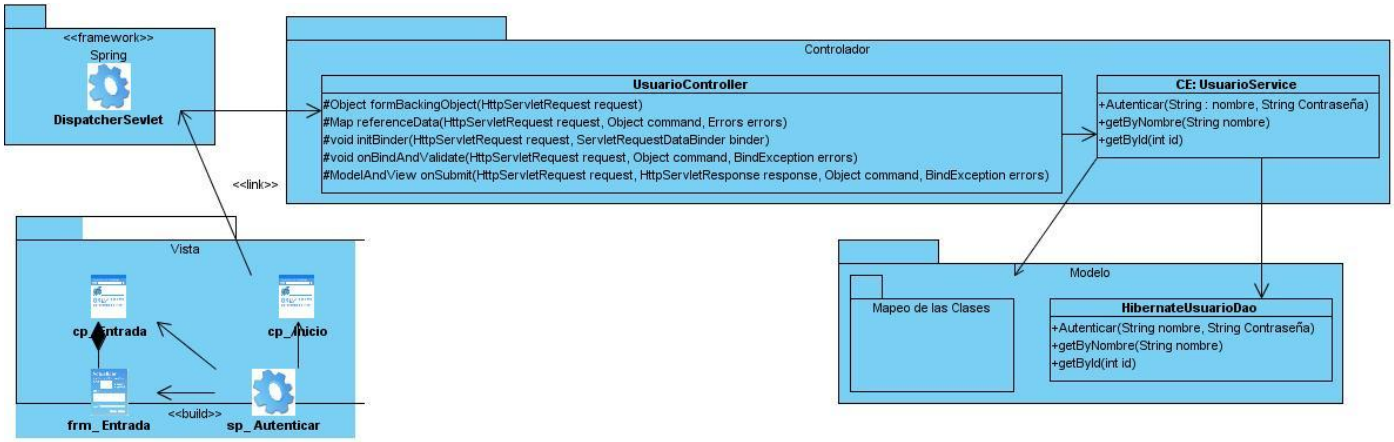

Figura 9 Diagrama de Clases del Diseño: CU Autenticar Usuario

3.3 Diagrama de secuencia

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en secuencia temporal. Muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados

entre los objetos, para llevar a cabo la funcionalidad descrita por el escenario. En aplicaciones grandes además de los objetos se muestran también los componentes y casos de uso.

En la figura 10 se muestra el diagrama de secuencia de autenticar usuario. El usuario entra los datos. El controlador se encarga de enviar a la interfaz correspondiente según el resultado de la autenticación.

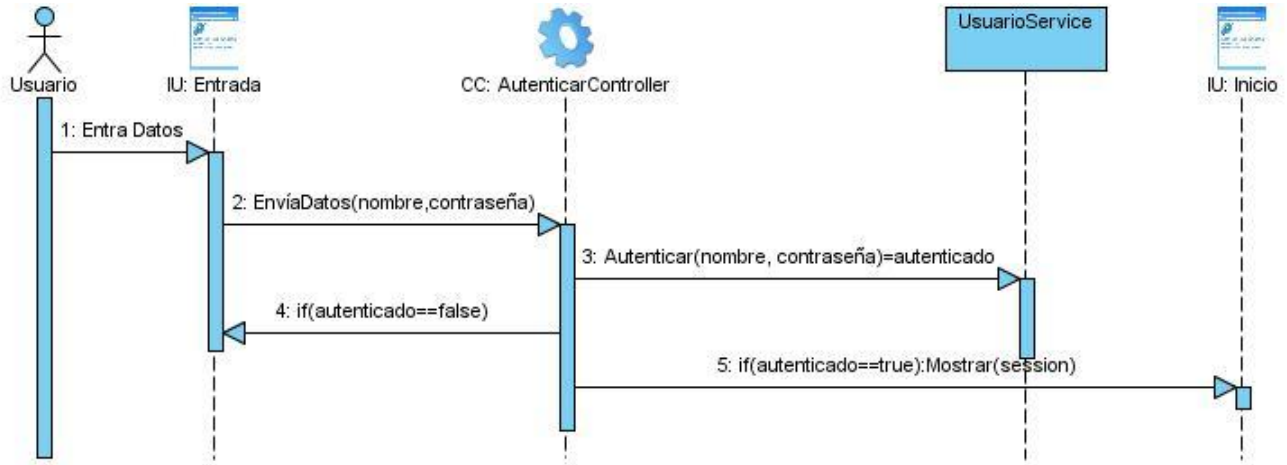

Figura 10 Diagrama de Secuencia: CU Autenticar Usuario

El resto de los diagramas de secuencia pueden ser consultados en los anexos de este documento.

3.4 Modelo de datos

Para almacenar la información generada en el proceso de examen clínico se diseñó una Base de Datos. La misma fue diseñada en el DBDesigner ForK, herramienta que permite el diseño y exportación a diferentes gestores de bases de datos tales como: PostgreSQL, MySQL, Oracle y SQL Server.

La Base de Datos, compuesta por cincuenta tablas, está normalizada hasta la tercera forma normal. La tabla principal, denominada Sujeto, contiene los datos personales (nombre, apellidos, edad, sexo, policlínico, área de salud, municipio y provincia de residencia) del sujeto que pertenece al estudio. A cada sujeto se le realizan una serie de exámenes existiendo así, una tabla Examen de la que heredan los diferentes tipos de exámenes. Cada examen tiene un encuestador (persona que lo realiza) y un responsable (usuario que lo inserta en el sistema). Todos los exámenes tienen asociadas una serie de tablas donde se almacenan los datos obtenidos en cada uno. Existen tablas que son necesarias para almacenar otros datos del sujeto como son: causas, causas específicas, en las que se guardan las

causas de exclusión del estudio y los sujetos excluidos. Para consultar más información referente a la Base de Datos, remitirse al Expediente de Proyecto de la versión 1.0 del módulo de Examen Clínico, ya que en esta versión solo se construye el diagrama de clases con el ORM Hibernate .

3.5 Modelo de despliegue

Es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuyen las funcionalidades entre los nodos de cómputo. (51)

La aplicación estará distribuida en los servidores de CNEURO, de la siguiente manera: Un servidor web para publicar el sistema, otro servidor para la base de datos, a la aplicación se tendrá acceso desde cualquier máquina cliente del centro. En la figura 22 se presenta el diagrama de despliegue.

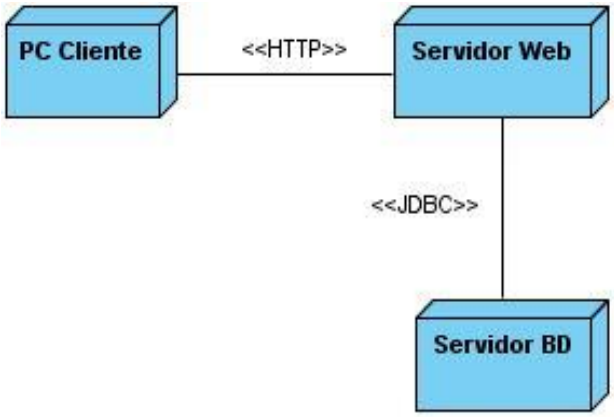

Figura 22 Diagrama de Despliegue

Conclusiones

Se realizaron los diagramas de clase de los casos de uso, así como sus realizaciones del diseño. Se ejemplificaron y fundamentaron varios de los patrones utilizados. Se muestra el modelo de datos definido.

Capítulo 4: Implementación y Prueba

Introducción

En este capítulo se describe cómo fue implementado el sistema a través de los diagramas de componentes y breves explicaciones de estos. Se muestra un conjunto de interfaces del sistema.

4.1 Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean estos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. (52)

A continuación se muestran los diagramas de componentes correspondientes a los casos de uso: Autenticar Usuario, Generar Reportes, GestionarCuestionarioManualidad, Gestionar Psicométrico y Gestionar Sujeto.

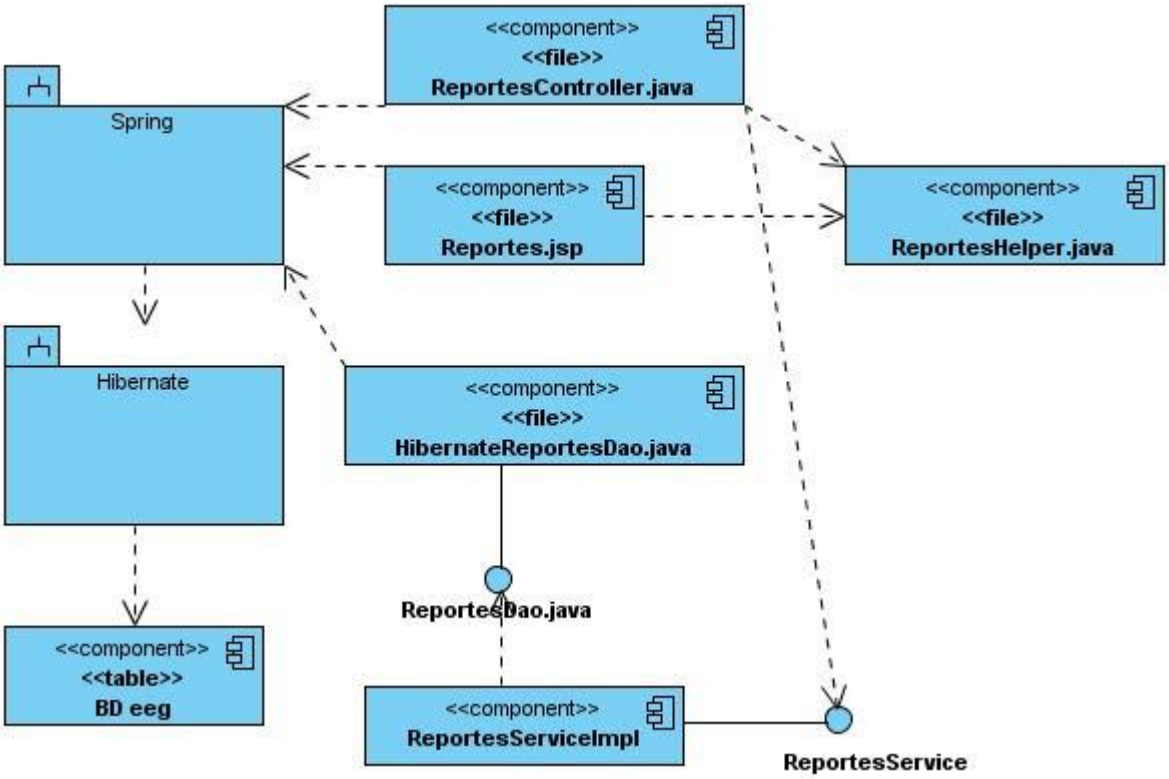

Figura 24 Diagrama de componentes Caso de Uso Generar Reportes

Se muestran los componentes y relaciones de dependencia para el caso de uso Generar Reportes.

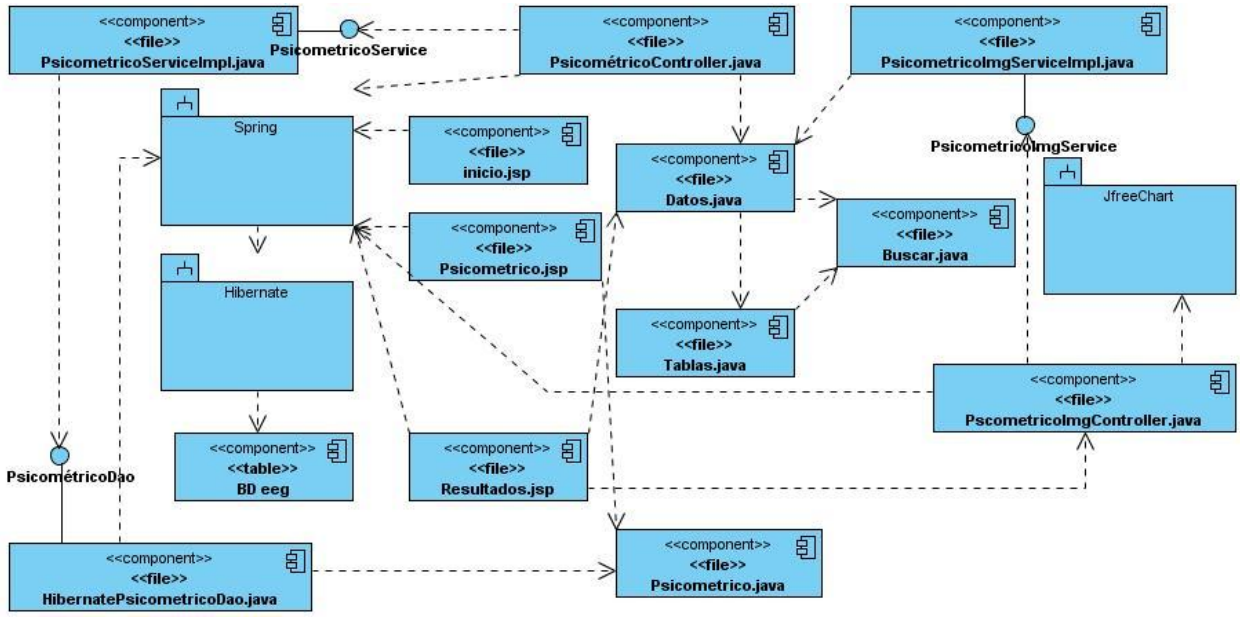

Figura 26 Diagrama de componentes Caso de Uso Gestionar Examen Psicométrico

Se muestran los componentes y relaciones de dependencia para el caso de uso Gestionar Examen Psicométrico. Obsérvese el subsistema JfreeChart, una librería de licencia GNU/GPL para la generación de gráficas.

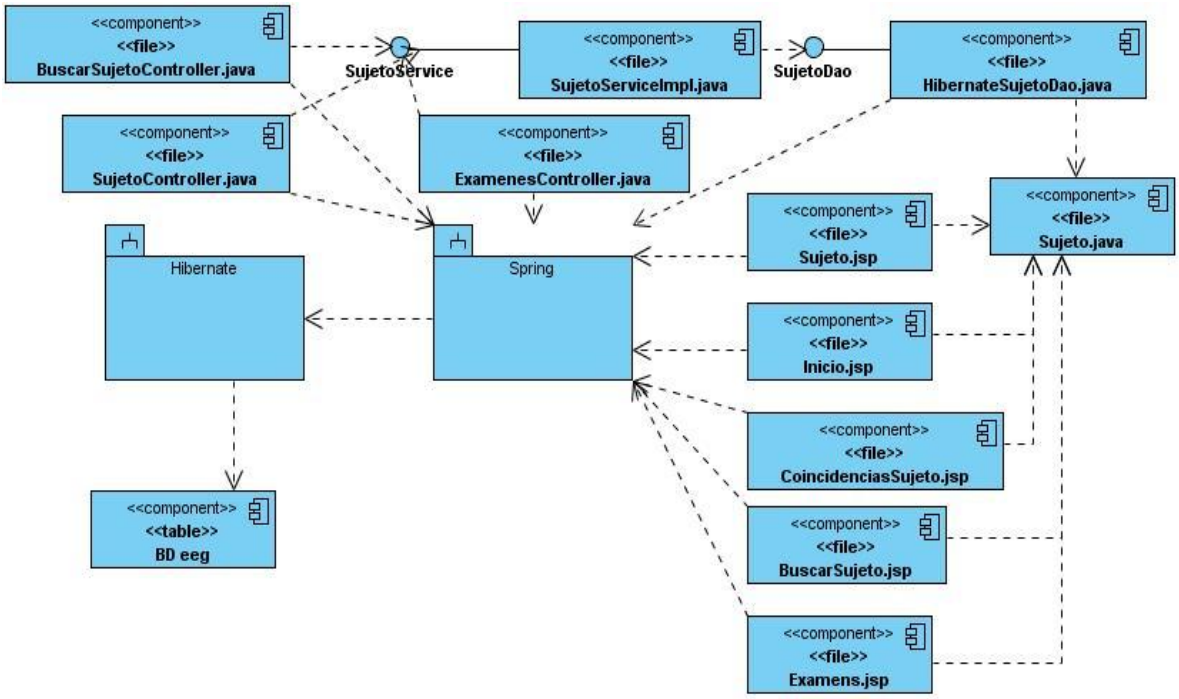

Figura 27 Diagrama de componentes Caso de Uso Gestionar Sujeto

Se muestran los componentes y relaciones de dependencia para el caso de uso Gestionar Sujeto.

4.2 Código fuente

El código fuente de un sistema informático es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para así poder ejecutar las funcionalidades requeridas por el software. El código fuente varía según el lenguaje de programación.


```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  <bean name="/login.html" class="org.springframework.web.servlet.mvc.UrlFilenameViewController"/>

  <bean name="/Autenticar.html" class="controllers.AutenticarController">
  <property name="usuario">
        <ref bean="UsuarioService"/>
    </property>
  </bean>
  <bean name="/Welcome.html" class="org.springframework.web.servlet.mvc.UrlFilenameViewController"/>
  <bean name="/IngresarUsuario.html
    <property name="usuario">
        <ref bean="UsuarioService"/
    </property>
    <property name="rol">
          <ref bean="RolService" />
      </property>
  </bean>
  <bean name="/ModificarUsuario.html" class="controllers.ModificarUsuarioController">
    <property name="usuario">
        <ref bean="UsuarioService"/>
    </property>
    <property name="rol">
          <ref bean="RolService" />
      </property>
  </bean>
  <bean name="/IngresarEncuestador.html" class="controllers.IngresarEncuestadorController">
    <property name="encuestador">
        <ref bean="EncuestadorService"/>
    </property>
  </bean>
```

Figura 28 Fragmento del fichero de configuración del servlet de Spring donde se inicializa cada sub-servlet. Obsérvese el mapeo de cada url a un controlador y el patrón de Inyección de Dependencias (IOC: inversión of control en inglés)

4.3 Casos de prueba

Las pruebas, son una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos específicos, los resultados obtenidos son observados, registrados y finalmente se realiza una evaluación.

Para desarrollar las pruebas al sistema se decidió utilizar el método de caja negra, que se refiere a las pruebas llevadas a cabo sobre la interfaz del software. Es decir, a través de ellas se pretende demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. En general, este método se centra principalmente en los requisitos funcionales del software. (53)

Escenario	Variable1 campo usuario	Variable2 campo contraseña	Respuesta del Sistema	Resultado de la prueba	Flujo central
Autenticar usuario correctamente	V	V	El sistema comprueba el usuario y la contraseña, asigna un rol y muestra la pantalla de inicio.	El sistema muestra la pantalla de inicio con los enlaces a las paginas a las que tiene acceso el rol	1. Introducir los valores correctamente en todos los campos. 2. Presionar el botón **entrar**.
Autenticar usuario incorrectamente.	I	I	El sistema muestra un mensaje de error.	El sistema muestra un mensaje de error.	1. Introducir los valores incorrectos en los campos. 2. Presionar el botón **entrar**.
	V	I			
	I	V			

Figura 31 Prueba de Caja Negra CU Autenticar Usuario

scenario	Variable1 campo Fecha de Inserción	Variable2 campo Fecha de aplicación	Variable2 campo Encuestador	Variable4 Todos los campos de características seleccionados	Respuesta del Sistema	Resultado de la prueba	Flujo centr
ısertar uestionario lanualidad orrectamente	V	V	V	V	El sistema guarda el cuestionario asociado al sujeto y muestra la interfaz de sujeto-exámenes.	El sistema guarda el cuestionario asociado al sujeto y muestra la interfaz de sujeto-exámenes.	1. Introduci correctame campos. 2. Presionar
ısertar uestionario lanualidad orrectamente	F	V	V	V	El sistema señala en la página los campos inválidos.	El sistema señala en la página los campos inválidos.	1. Introduci incorrectam campos. 2. Presionar
	V	F	V	V			
	V	V	F	V			
	V	V	V	F			
	F	F	V	V			
	F	V	F	V			
	F	V	V	F			
	V	F	F	V			
	V	F	V	F			
	V	V	F	F			

Figura 32 Prueba de Caja Negra: CU: Insertar Cuestionario de Manualidad

Escenario	Variables	Respuesta del Sistema	Resultado de la prueba	Flujo central
Generar Reportes	-	El sistema muestra la interfaz de tablas con reportes.	El sistema muestra la interfaz de tablas con reportes.	ir al enlace **Reportes**.

Figura 33 Prueba de Caja Negra: CU: Generar Reportes

4.3 Pantallas de la aplicación

A continuación se muestran algunas de las pantallas que el equipo de desarrollo considera que tienen gran importancia en el funcionamiento del sistema.

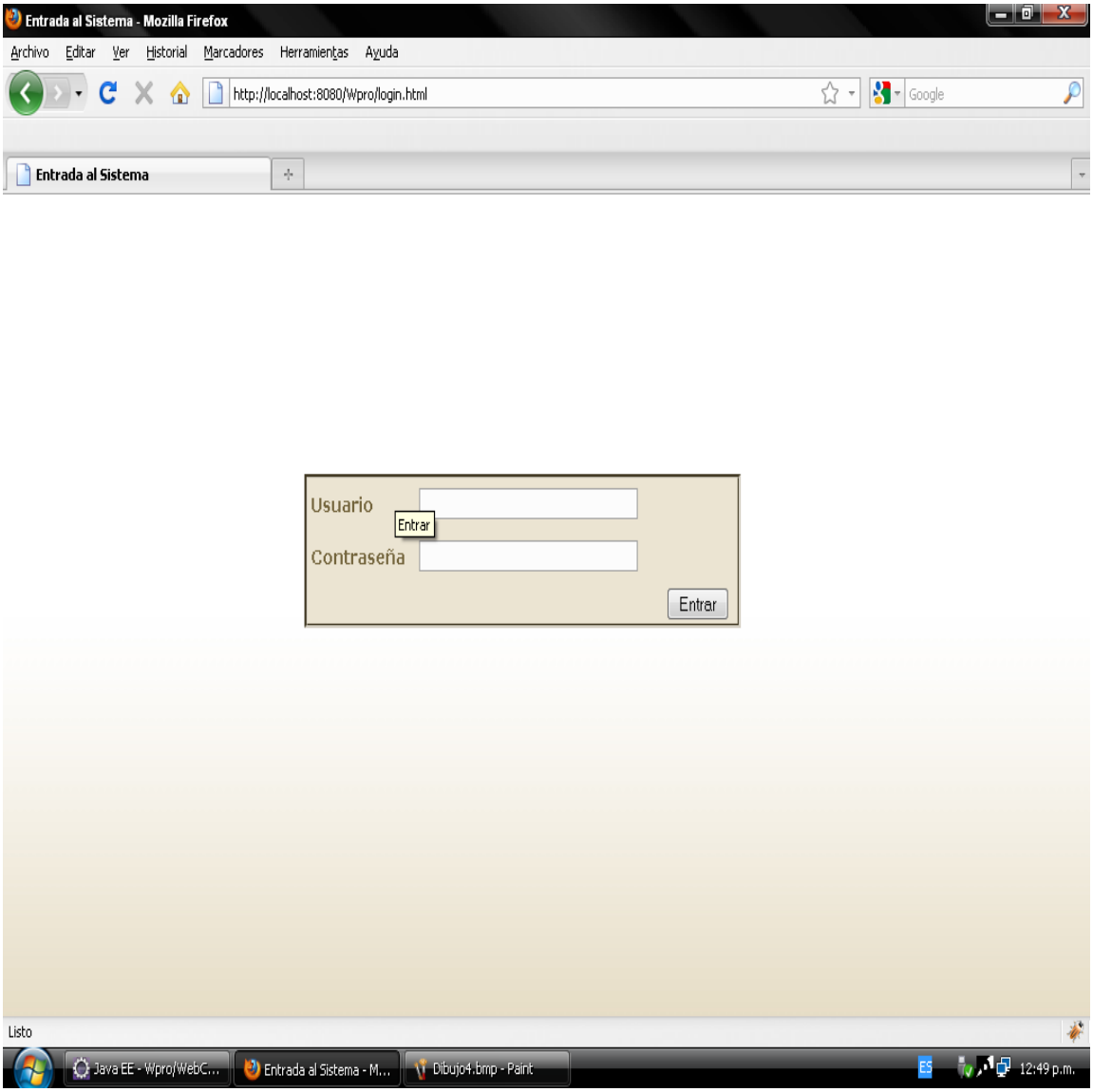

Figura 34 Página para autentificarse en el sistema

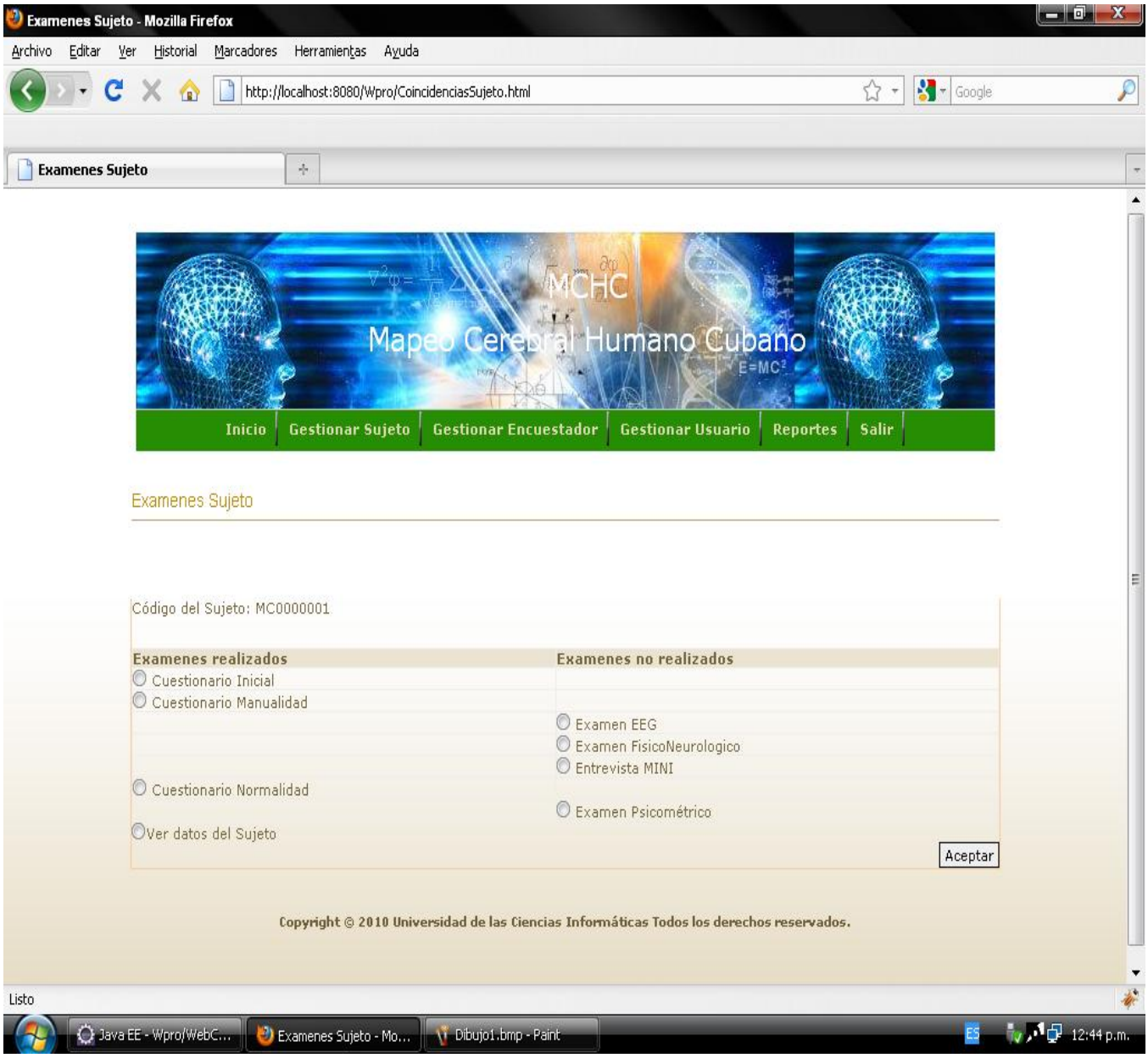

Figura 36 Página de Resultados de las Coincidencias de los Sujetos

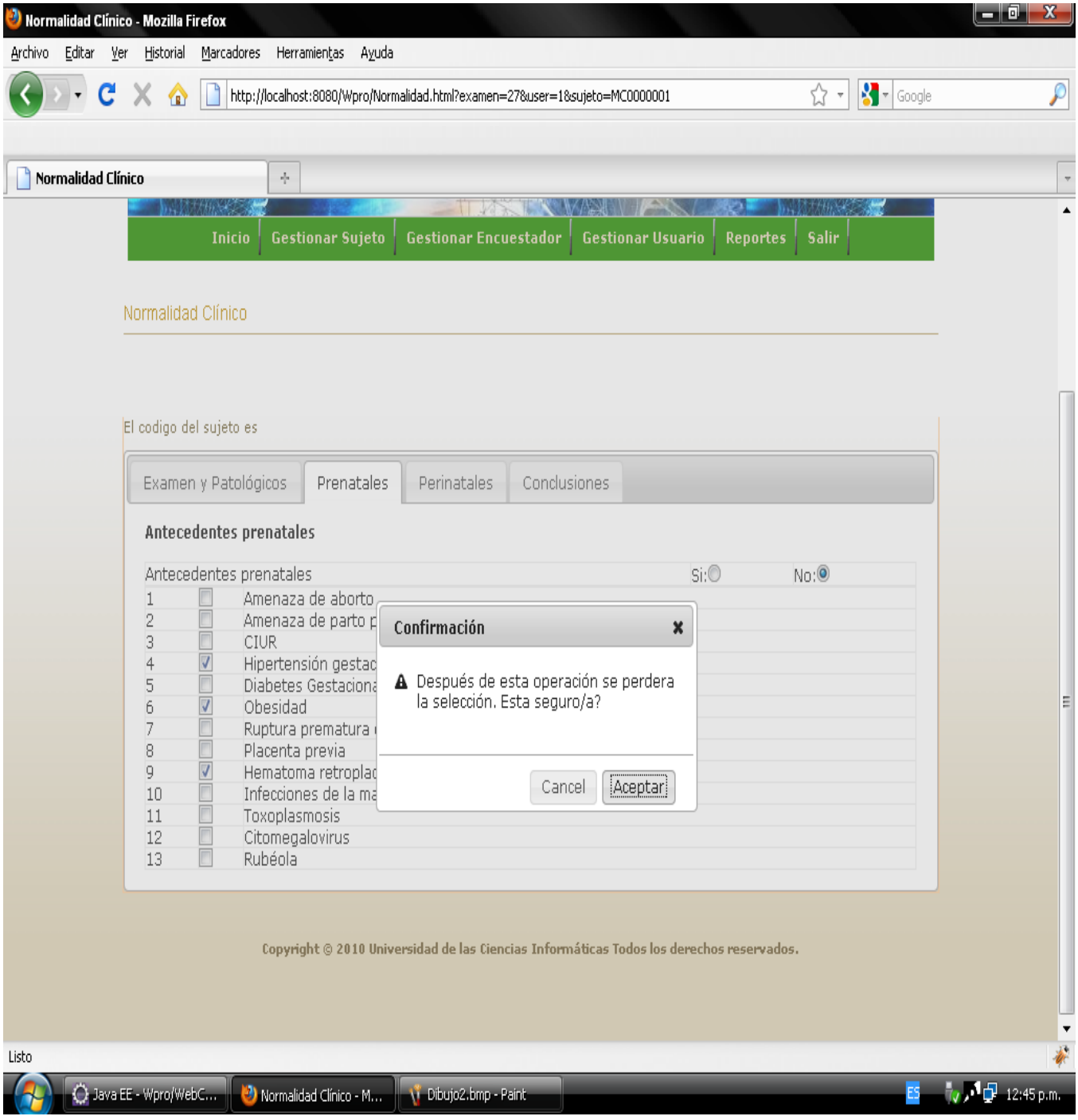

Figura 37 Página del Examen de Normalidad

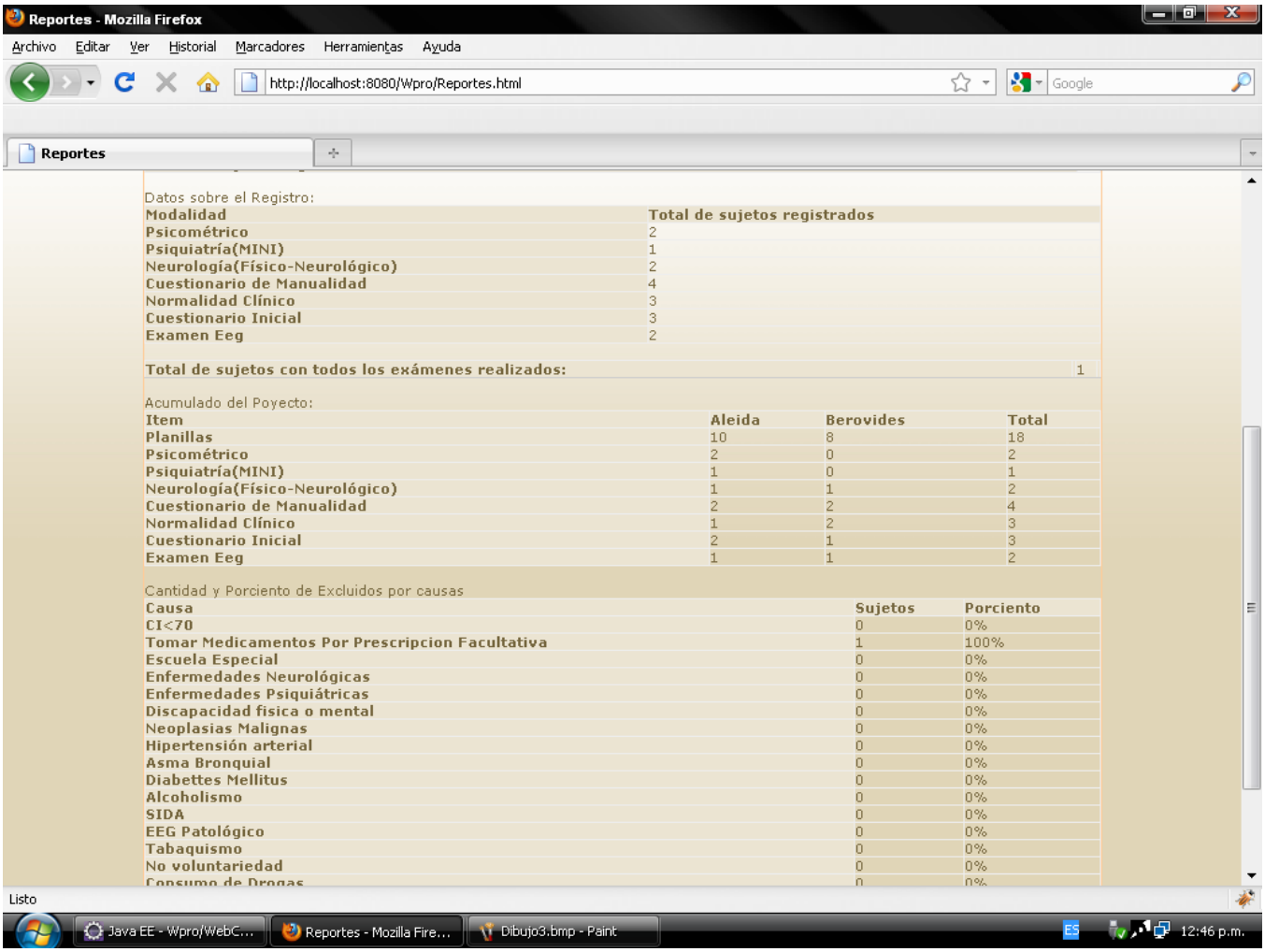

Figura 38 Página de Resultados de los Reportes

Conclusiones

Se obtuvo como resultado la confección de los diagramas de componentes y sus descripciones, logrando mostrar las organizaciones y dependencias lógicas entre los componentes del sistema desarrollado. Se mostraron y describieron algunas pantallas funcionales para evidenciar las principales características visuales del sistema, presentándose además, ejemplos de pruebas de caja negra realizadas, las cuales verifican el cumplimento de algunos de los requerimientos funcionales identificados.

Conclusiones generales

Se investigaron las tendencias en el uso de la plataforma J2EE en aplicaciones web y se mencionaron sitios que hacen uso de esta.

Se realizó el diseño de clases del sistema utilizando los frameworks definidos por la arquitectura.

Como resultado se obtuvo la versión 2.0 de la aplicación del módulo Examen Clínico del proyecto Mapeo Cerebral Humano Cubano.

Recomendaciones

- Se recomienda para el uso de este software utilizar navegadores que soporten javascript como Internet Explorer 6.0 y Mozilla Firefox preferiblemente este último. En otros navegadores con soporte para javascript comprobados como Google Chrome, la aplicación funcionará pero se observará ligeramente distinta.
- Mantener la navegabilidad que ofrece la aplicación evitando en lo posible la utilización de los controles atrás y adelante en los navegadores pues puede conducir a errores de sobre-inserción de datos.
- Completar la BD con los policlínicos por municipio de todo el país cuando el estudio lo requiera. El sistema responde a un pedido de estos policlínicos como “no disponible” hasta tanto no se inserten en la base de datos, además deberá cambiarse ligeramente la implementación de la interfaz de reportes que por acuerdo con el cliente en la versión anterior solo presenta los dos policlínicos existentes en la BD.

Referencias Bibliográficas

1. **Armas Montero, Lázaro y Laurencio Giralt, Maikel.** *Sistema de Gestión de la Información del Proyecto Mapeo Cerebral Humano Cubano: Módulo Examen Clínico.* 2007.

2. *JQuery.* [En línea] [Citado el: 7 de Marzo de 2010.] http://jquery.com/.

3. *Ext JS.* [En línea] [Citado el: 7 de Marzo de 2010.] http://www.extjs.com.

4. *Rich Internet Aplication.* [En línea] [Citado el: 8 de Marzo de 2010.] www.adobe.com/.../rich_internet_apps/.

5. *ExtJs.* [En línea] [Citado el: 18 de Febrero de 2010.] http://www.extjs.com/deploy/dev/examples/.

6. **Zammetti, Frank.** *Practical Ext JS Projects with Gears.* 2009. 978-1-4302-1924-8.

7. *Adobe.* [En línea] [Citado el: 4 de Marzo de 2010.] http://www.adobe.com/es/products/flex/?promoid=BPBJI.

8. *Adobe.* [En línea] [Citado el: 4 de Marzo de 2010.] http://www.adobe.com/.

9. *Google.* [En línea] [Citado el: 5 de Marzo de 2010.] http://gears.google.com/.

10. *Firefox*. [En línea] [Citado el: 5 de Marzo de 2010.] http://www.firefox.org.

11. *Internet Explorer.* [En línea] [Citado el: 5 de Marzo de 2010.] http://www.microsoft.com/spain/windows/internet-explorer/.

12. *Google Chrome.* [En línea] [Citado el: 5 de Marzo de 2010.] http://chrome.blogspot.com/.

13. **Morrison, Michael.** *Hadfirst Javascript*. s.l. : O' Reilly, 2008. 978-0-596-52774-7.

14. **Heilmann, Christian.** *Beginning JavaScript with DOM Scripting and Ajax*. s.l. : Apress, 2006. 978-1-59059-680-7.

15. *Dojo.* [En línea] [Citado el: 18 de Marzo de 2010.] www.dojotoolkit.org.

16. *jQuery:* The Write Less, Do More, JavaScript Library. [En línea] [Citado el: 12 de Mayo de 2010.] http://jquery.com/.

17. *Prototype JavaScript framework: Easy Ajax and DOM manipulation for dynamic web applications.* [En línea] [Citado el: 15 de Marzo de 2010.] http://www.prototypejs.org/.

18. Kyle Hayes. [En línea] [Citado el: 12 de Mayo de 2010.] http://www.kylehayes.info.

19. *Java*. [En línea] [Citado el: 4 de Marzo de 2010.] http://www.java.com/es/.

20. *Sun Java*. [En línea] Sun MicroSystems. [Citado el: 12 de Mayo de 2010.] http://java.sun.com/j2ee/overview.html.

21. *PHP*. [En línea] [Citado el: 6 de Marzo de 2010.] http://php.net/index.php.

22. *Python*. [En línea] [Citado el: 6 de Marzo de 2010.] http://www.python.org/.

23. *Amazon*. [En línea] [Citado el: 6 de Marzo de 2010.] http://www.amazon.com/Getting-StartED-Dojo-Kyle-Hayes/dp/1430225211/ref=ntt_at_ep_dpi_1/179-9236748-5366801.

24. *Gamil*. [En línea] [Citado el: 6 de Marzo de 2010.] http://www.mail.google.com/mail/?hl=es.

25. *hi5 | Social Entertainment*. [En línea] [Citado el: 12 de Marzo de 2010.] http://www.hi5.com/.

26. *Facebook*. [En línea] [Citado el: 12 de Marzo de 2010.] http://www.facebook.com.

27. *Google*. [En línea] [Citado el: 12 de Marzo de 2010.] http://www.google.com.

28. *GWT*. [En línea] [Citado el: 12 de Marzo de 2010.] http://code.google.com/webtoolkit.

29. *EBay*. [En línea] [Citado el: 12 de Marzo de 2010.] http://www.ebay.com.

30. *Java Hispano*. [En línea] [Citado el: 7 de Marzo de 2010.] http://www.javahispano.org/contenidos/es/por_que_los_grandes_sitios_no_estan_construidos_con_java/.

31. **Curto, Josep.** *Information Management*. [En línea] [Citado el: 20 de 4 de 2009.] http://informationmanagement.wordpress.com/2006/11/28/%C2%BFque-es-la-gestion-de-la-%20informacion-3-de-4/.

32. *Roadmap:OpenUp/Basic*. [En línea] [Citado el: 21 de 4 de 2009.] http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/roadmaps/openup_basic_roadmap,_vEruwN-rEdqiM_wFaqLjNg.html#.

33. **Ferré Grau, Xavier.** *Clikear.com*. [En línea] [Citado el: 5 de 4 de 2010.] http://www.clikear.com/manuales/uml/index.aspx.

34. **Tio, Miguel Angel.** *JSP: MANUAL INICIALIZACION.* [En línea] [Citado el: 5 de 4 de 2010.] http://usuarios.multimania.es/migutio/tio/jsp.html.

35. **Yanez, William.** *Java Server Pages TM.* [En línea] [Citado el: 5 de 4 de 2010.] http://mipagina.cantv.net/williamyanez/jsp/default.htm..

36. **Exes.** *Java.* [En línea] [Citado el: 5 de 4 de 2010.] http://www.mailxmail.com/curso-java/caracteristicas-lenguaje-java-1.

37. **Manzanedo del Campo, Miguel Ángel.** *CARACTERÍSTICAS DE JAVA.* [En línea] [Citado el: 7 de 4 de 2010.] http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_3.htm.

38. **Don Denoncourt.** *Elección del servidor de aplicaciones web* . [En línea] [Citado el: 4 de 4 de 2010.] http://www.help400.es/asp/scripts/nwart.asp?Num=131&Pag=10&Tip=T.

39. **Quiñones, Ernesto.** *INTRODUCCION A POSTGRESQL.* [En línea] [Citado el: 4 de 5 de 2010.] http://postgresql.org.pe/articles/introduccion_a_postgresql.pdf.

40. **Fernando Telechea, Germán Matosas.** *Frameworks de Aplicación.* [En línea] [Citado el: 4 de 5 de 2010.] www.asiap.org/jiap/presentaciones/bull.pps.

41. **Sánchez Rico, Mario Alfredo.** *Spring.* [En línea] [Citado el: 4 de 5 de 2010.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo3.pdf.

42. **Tuesta Pereyra, Martín.** *Herramientas CASE.* [En línea] [Citado el: 4 de 5 de 2010.] http://www.unapiquitos.edu.pe/intranet/pagsphp/docentes/archivos/Capitulo%205%20-%20Herramientas%20CASE.doc?PHPSESSID=a8bf8196ddc53f256dff70e017d71fb1.

43. **Manrique, Jorge.** *Introducción a las Herramientas Case.* [En línea] [Citado el: 4 de 4 de 2010.] www.iesjorgemanrique.es/hda/descargas/tema1.ppt.

44. *Desarrollo del Software.* [En línea] [Citado el: 8 de Marzo de 2010.] http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/solucTallerEventos.pdf.

45. **Alexander, Cristopher.** *A Pattern Language.*

46. **León, Welicki.** *Patrones y Antipatrones.* [En línea] [Citado el: 4 de 5 de 2010.] http://msdn.microsoft.com/es-es/library/bb972242.aspx#M2.

47. **Larman, Craig.** *UML y Patrones.*

48. [En línea] [Citado el: 8 de Marzo de 2010.] http://www.jarq.googlecode.com/svn/trunk/docs/JArq-DocumentoDiseño.doc.

49. **Booch, Grady.** [En línea] [Citado el: 8 de Marzo de 2010.] http://www.casadellibro.com/libros/booch-grady/booch32grady.

50. **Visconti , Marcello; Astudillo, Hernán ;.** [En línea] [Citado el: 5 de Abril de 2010.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/10-DisenoOO.pdf.

51. **Sánchez, Omar.** *Conceptos.* [En línea] [Citado el: 8 de Marzo de 2010.] http://omarsanchez.net/conceptomod.aspx.

52. **Vilas Fernández, Ana.** *Introducción a UML.* [En línea] [Citado el: 5 de 5 de 2010.] http://wwwgris.det.uvigo.es/~avilas/UML/UML.html.

53. *Pruebas.* [En línea] [Citado el: 8 de Marzo de 2010.] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm.

54. **Kyle Hayes.** *Kyle Hayes | Proficiency by Derivation.* [En línea] [Citado el: 20 de Mayo de 2010.] http://www.kylehayes.info/2009/03/29/survey-results-javascript-frameworks/.

55. *Roadmap:OpenUP/Basic.* [En línea] [Citado el: 24 de 3 de 2010.] http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/roadmaps/openup_basic_roadmap,_vEruwN-rEdqiM_wFaqLjNg.html#.

56. *Lenguajes de Programación.* [En línea] [Citado el: 5 de 4 de 2010.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

57. *Spring Security Reference Documentation 2.0.x*. 2005.

Bibliografía

Alvarez, Sara. *desarrolloweb.com*. [En línea] Guiarte Multimedia, 31 de Julio de 2007. http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

Armas Montero, Lázaro y Laurencio Giralt, Maikel. *Sistema de Gestión de la Información del Proyecto Mapeo Cerebral Humano Cubano: Módulo Examen Clínico.* s.l. : Universidad de las Ciencias Informáticas, 2007.

Cruzado Nuño, Ignacio. *Guía de iniciación al lenguaje Java.* [En línea] Versión 2.0, Noviembre de 1999. http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/java20.pdf.

González, Carlos D. *Curso Base de Datos PostgreSQL, SQL avanzado y PHP.* [En línea] Marzo de 2010. http://www.usabilidadweb.com.ar/postgre.php.

Hall, Marty y Brown, Larry. *Core Servlets and JavaServer Pages.* s.l. : Prentice Hall, 2004. ISBN: 3827266459.

Harrop, Rob y Machacek, Jan. *Pro Spring.* s.l. : Apress, 2005. ISBN: 1590594614.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software.* Madrid: Addison Wesley, 2000. ISBN: 8478290362.

Larman, Craig. *UML y Patrones.* México: Prentice Hall, 1999. ISBN: 9701702611.

Lou, Ricard. Programación en Castellano. *Catálogo de Patrones de Diseño J2EE. Y II: Capas de Negocio y de Integración.* [En línea] 14 de Diciembre de 2003. http://www.programacion.com/articulo/catalogo_de_patrones_de_diseno_j2ee_y_ii:_capas_de_negocio_y_de_integracion_243/8.

1. **Armas Montero, Lázaro y Laurencio Giralt, Maikel.** *Sistema de Gestión de la Información del Proyecto Mapeo Cerebral Humano Cubano: Módulo Examen Clínico.* 2007.

2. *JQuery.* [En línea] [Citado el: 7 de Marzo de 2010.] http://jquery.com/.

3. *Ext JS.* [En línea] [Citado el: 7 de Marzo de 2010.] http://www.extjs.com.

4. *Rich Internet Aplication.* [En línea] [Citado el: 8 de Marzo de 2010.] www.adobe.com/.../rich_internet_apps/.

5. *ExtJs.* [Online] [Cited: Febrero 18, 2010.] http://www.extjs.com/deploy/dev/examples/.

6. **Zammetti, Frank.** *Practical Ext JS Projects with Gears.* 2009. 978-1-4302-1924-8.

7. *Adobe.* [Online] [Cited: Marzo 4, 2010.] http://www.adobe.com/es/products/flex/?promoid=BPBJI.

8. *Adobe.* [En línea] [Citado el: 4 de Marzo de 2010.] http://www.adobe.com/.

9. *Google.* [En línea] [Citado el: 5 de Marzo de 2010.] http://gears.google.com/.

10. *Firefox*. [Online] [Cited: Marzo 5, 2010.] http://www.firefox.org.

11. *Internet Explorer.* [En línea] [Citado el: 5 de Marzo de 2010.] http://www.microsoft.com/spain/windows/internet-explorer/.

12. *Google Chrome.* [En línea] [Citado el: 5 de Marzo de 2010.] http://chrome.blogspot.com/.

13. **Morrison, Michael.** *Hadfirst Javascript*. s.l. : O' Reilly, 2008. 978-0-596-52774-7.

14. **Heilmann, Christian.** *Beginning JavaScript with DOM Scripting and Ajax*. *s.l.* : Apress, 2006. 978-1-59059-680-7.

15. *Dojo.* [Online] [Cited: Marzo 18, 2010.] www.dojotoolkit.org.

16. *jQuery: The Write Less, Do More, JavaScript Library.* [Online] [Cited: Mayo 12, 2010.] http://jquery.com/.

17. *Prototype JavaScript framework: Easy Ajax and DOM manipulation for dynamic web applications.* [Online] [Cited: Marzo 15, 2010.] http://www.prototypejs.org/.

18. Kyle Hayes. [Online] [Cited: Mayo 12, 2010.] http://www.kylehayes.info.

19. *Java.* [En línea] [Citado el: 4 de Marzo de 2010.] http://www.java.com/es/.

20. *Sun Java.* [Online] Sun MicroSystems. [Cited: Mayo 12, 2010.] http://java.sun.com/j2ee/overview.html.

21. *PHP*. [En línea] [Citado el: 6 de Marzo de 2010.] http://php.net/index.php.

22. *Python.* [En línea] [Citado el: 6 de Marzo de 2010.] http://www.python.org/.

23. *Amazon.* [Online] [Cited: Marzo 6, 2010.] http://www.amazon.com/Getting-StartED-Dojo-Kyle-Hayes/dp/1430225211/ref=ntt_at_ep_dpi_1/179-9236748-5366801.

24. *Gamil.* [Online] [Cited: Marzo 6, 2010.] http://www.mail.google.com/mail/?hl=es.

25. *hi5 | Social Entertainment.* [Online] [Cited: Marzo 12, 2010.] http://www.hi5.com/.

26. *Facebook*. [Online] [Cited: Marzo 12, 2010.] http://www.facebook.com.

27. *Google.* [Online] [Cited: Marzo 12, 2010.] http://www.google.com.

28. *GWT.* [Online] [Cited: Marzo 12, 2010.] http://code.google.com/webtoolkit.

29. *EBay.* [Online] [Cited: Marzo 12, 2010.] http://www.ebay.com.

30. *Java Hispano.* [En línea] [Citado el: 7 de Marzo de 2010.] http://www.javahispano.org/contenidos/es/por_que_los_grandes_sitios_no_estan_construidos_con_java/.

31. **Curto, Josep.** *Information Management.* [En línea] [Citado el: 20 de 4 de 2009.] http://informationmanagement.wordpress.com/2006/11/28/%C2%BFque-es-la-gestion-de-la-%20informacion-3-de-4/.

32. *Roadmap:OpenUp/Basic.* [En línea] [Citado el: 21 de 4 de 2009.] http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/roadmaps/openup_basic_roadmap,_vEruwN-rEdqiM_wFaqLjNg.html#.

33. **Ferré Grau, Xavier.** *Clikear.com.* [En línea] [Citado el: 5 de 4 de 2010.] http://www.clikear.com/manuales/uml/index.aspx.

34. **Tio, Miguel Angel.** *JSP: MANUAL INICIALIZACION.* [En línea] [Citado el: 5 de 4 de 2010.] http://usuarios.multimania.es/migutio/tio/jsp.html.

35. **Yanez, William.** *Java Server Pages TM.* [En línea] [Citado el: 5 de 4 de 2010.] http://mipagina.cantv.net/williamyanez/jsp/default.htm..

36. **Exes.** *Java.* [Online] [Cited: 4 5, 2010.] http://www.mailxmail.com/curso-java/caracteristicas-lenguaje-java-1.

37. **Manzanedo del Campo, Miguel Ángel.** *CARACTERÍSTICAS DE JAVA.* [En línea] [Citado el: 7 de 4 de 2010.] http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_3.htm.

38. **Don Denoncourt.** *Elección del servidor de aplicaciones web* . [En línea] [Citado el: 4 de 4 de 2010.] http://www.help400.es/asp/scripts/nwart.asp?Num=131&Pag=10&Tip=T.

39. **Quiñones, Ernesto.** *INTRODUCCION A POSTGRESQL.* [En línea] [Citado el: 4 de 5 de 2010.] http://postgresql.org.pe/articles/introduccion_a_postgresql.pdf.

40. **Fernando Telechea, Germán Matosas.** *Frameworks de Aplicación.* [En línea] [Citado el: 4 de 5 de 2010.] www.asiap.org/jiap/presentaciones/bull.pps.

41. **Sánchez Rico, Mario Alfredo.** *Spring.* [Online] [Cited: 5 4, 2010.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo3.pdf.

42. **Tuesta Pereyra, Martín.** *Herramientas CASE.* [En línea] [Citado el: 4 de 5 de 2010.] http://www.unapiquitos.edu.pe/intranet/pagsphp/docentes/archivos/Capitulo%205%20-%20Herramientas%20CASE.doc?PHPSESSID=a8bf8196ddc53f256dff70e017d71fb1.

43. **Manrique, Jorge.** *Introducción a las Herramientas Case.* [En línea] [Citado el: 4 de 4 de 2010.] www.iesjorgemanrique.es/hda/descargas/tema1.ppt.

44. *Desarrollo del Software.* [En línea] [Citado el: 8 de Marzo de 2010.] http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/solucTallerEventos.pdf.

45. **Alexander, Cristopher.** *A Pattern Language.*

46. **León, Welicki.** *Patrones y Antipatrones.* [Online] [Cited: 5 4, 2010.] http://msdn.microsoft.com/es-es/library/bb972242.aspx#M2.

47. **Larman, Craig.** *UML y Patrones.*

48. [En línea] [Citado el: 8 de Marzo de 2010.] http://www.jarq.googlecode.com/svn/trunk/docs/JArq-DocumentoDiseño.doc.

49. **Booch, Grady.** [En línea] [Citado el: 8 de Marzo de 2010.] http://www.casadellibro.com/libros/booch-grady/booch32grady.

50. **Visconti , Marcello; Astudillo, Hernán ;.** [En línea] [Citado el: 5 de Abril de 2010.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/10-DisenoOO.pdf.

51. **Sánchez, Omar.** *Conceptos.* [En línea] [Citado el: 8 de Marzo de 2010.] http://omarsanchez.net/conceptomod.aspx.

52. **Vilas Fernández, Ana.** *Introducción a UML.* [En línea] [Citado el: 5 de 5 de 2010.] http://wwwgris.det.uvigo.es/~avilas/UML/UML.html.

53. *Pruebas.* [En línea] [Citado el: 8 de Marzo de 2010.] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm.

54. **Kyle Hayes.** *Kyle Hayes | Proficiency by Derivation.* [Online] [Cited: Mayo 20, 2010.] http://www.kylehayes.info/2009/03/29/survey-results-javascript-frameworks/.

55. *Roadmap:OpenUP/Basic.* [En línea] [Citado el: 24 de 3 de 2010.] http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/roadmaps/openup_basic_roadmap,_vEruwN-rEdqiM_wFaqLjNg.html#.

56. *Lenguajes de Programación.* [En línea] [Citado el: 5 de 4 de 2010.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

57. *Spring Security Reference Documentation 2.0.x*. 2005.

Glosario De Términos

Neuropsiquiátrica: El término "neuropsiquiatría" se refiere fundamentalmente a disciplinas clínicas que se entrecruzan al compartir la creencia de que los síntomas mentales se producen en zonas cerebrales alteradas. También se utiliza para dejar sentada una posición profesional con relación a puntos de vista rivales sobre los trastornos mentales, tales como el psicoanálisis. Finalmente, crea un espacio social y económico donde investigadores que piensan de una manera similar se congregan, sin peligro, para usufructuar sus ideas a la moda.

Stakeholders: Aquellas “partes interesadas” en el proyecto o personas que de alguna manera afectan y tienen relación con el proyecto; pueden ser patrocinadores, proveedores, la comunidad vecina, etc. Es importante su identificación, la gestión de la comunicación con éstos y visualizarlos como clientes externos o internos, según corresponda, determinando qué esperan de nosotros y que esperamos de ellos. Es recomendable aplicar una política calidad de atención a los stakeholders.

Framework: Estructura de soporte definida, en la cual un proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting, para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

JAVA: Lenguaje de programación orientado a objetos desarrollado por Sun Microsystems.

MVC: Modelo, vista, controlador. Patrón de Arquitectura.

PostgreSQL: Sistema de gestión de bases de datos liberado bajo la licencia BSD para software libre.

DBMS: Database Management System. Sistema de Administración de Bases de Datos.

Javascript: Lenguaje interpretado por la mayoría de los navegadores, es utilizado para interactuar con la estructura de la página. Sirve a de base a muchos frameworks que enriquecen la interfaz de aplicaciones web.

Jquery: librería que funciona sobre javascript para interactuar con el DOM.