Universidad de las Ciencias Informáticas
Facultad 6

Título: Sistema Informático para la Red Nacional de Genética Médica (SIGM): Registro Cubano de Enfermedades Genéticas Versión 2.0.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Joan Martínez Ariosa.

Tutores: Ing. Sonia González del Sol

Ing. Yoendris Lacoste Ricardo

Ciudad de la Habana, Cuba

Julio, 2008

DECLARACIÓN DE AUTORÍA

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

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

Joan Martínez Ariosa	Ing. Sonia González del Sol	Ing. Yoendris Lacoste Ricardo
_____	_____	_____
Firma del autor	Firma del tutor	Firma del tutor

Datos de Contacto

Tutor:

Ing. Sonia González del Sol

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Email: sgonzalez@uci.cu

Tutor:

Ing. Yoendris Lacoste Ricardo

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Email: ylacoste@uci.cu

AGRADECIMIENTOS

A mi familia por apoyarme y darme tanto aliento.

A Franklis, a Otto, a Yosuán, a Alelí, a Martha, a Leydi, a Odannys, a Jommy, y a todo el que me aclaró aunque sea una duda.

A mis amigos de la universidad, al Ruso, al Lacho, a Maikel, al Vila, a Alfredo y su novia, a mis compañeros de apartamento, a Martica (Denia), a Anabel, a Yanet por aconsejarme tanto y preocuparse más por mi tesis que yo mismo, a Ingrid por animarme y ayudarme.

A Belquis por levantarme cada vez el ánimo y apoyarme tanto.

Al profe Ballester que me sacó de buen apuro.

A los profes que me dieron varias oportunidades.

A todo el que me ayudó o me dio ánimo de alguna forma.

En fin a los que si y los que no…

DEDICATORIA

A mi papá, que puede que no lo sepa, pero he terminado aquí por él.

A mi hermana que siempre confió en mí y estuvo ahí, por ser mi hermana mayor aunque sea menor que yo.

A mi abuelo Epi, por ser mi todo, y por dejarme ser yo sin siquiera mirarme duro.

A Belquis por estar siempre ahí, aunque ya no haya un nosotros.

A mis tíos Raúl, Rafe, María y mis abuelos Ofelia y Tomás por estar pendientes de mi y apoyarme en todo momento.

A Santica por su apoyo incondicional y por permitirnos ser sus hijos.

A Ernestico, que siga así y no coja el ejemplo de su primo mayor.

A mi familia y mis amigos, que han estado ahí en buenas y malas.

A los amigos que ya no están junto a nosotros: Siul, Leosbel, Jessye por los buenos momentos, y si, también por los malos.

RESUMEN

El Centro Nacional de Genética Médica (CNGM) tiene la necesidad de almacenar y gestionar más fácil y centralizadamente la información de los pacientes que sufren de enfermedades genéticas, para así poder hacer estudios y llegar a conclusiones claves para el tratamiento de dichas enfermedades en nuestra sociedad.

La Universidad de las Ciencias Informáticas, en cooperación con el CNGM se dispuso a crear un sistema informático que esté integrado a la Red Nacional de Salud: el Sistema Informático de Genética Médica (SIGM), que consta de varios registros, entre los que está el Registro Cubano de Enfermedades Genéticas (RECUEGEN). El tema de esta investigación es el diseño y la implementación de dicho registro, donde los especialistas genéticos podrán obtener valores estadísticos o totales de pacientes por distintos criterios, y tendrán la posibilidad de gestionar enfermedades genéticas. Estos procesos se harán de una forma más rápida y segura.

PALABRA CLAVE
Enfermedad genética.

Tabla de Contenidos

INTRODUCCIÓN

El cuidado de la salud de la población cubana es uno de los aspectos más tomados en cuenta por nuestro gobierno, dentro de esta esfera, la atención a los pacientes que presentan padecimientos de genéticos tiene gran prioridad.

¿Qué es una enfermedad genética?

Corresponde a un grupo heterogéneo de afecciones que en su etiología presentan un significativo componente genético. Ello puede ser alguna alteración en un solo gen (monogénicas), en varios genes (multifactoriales) o en muchos genes (cromosomas). La alteración genética puede producir directamente la enfermedad (por ejemplo, el caso de la Hemofilia) o interactuar con factores ambientales (como por ejemplo, la predisposición genética en la etiología de la Hipertensión arterial).

A nivel nacional, es el Centro Nacional de Genética Médica (CNGM) quien rige las investigaciones básicas en el campo de los genes. Fue inaugurado el 5 de agosto del año 2003 por el Comandante en Jefe Fidel Castro Ruz con el objetivo de llevar un mejor control y monitorear el comportamiento de los problemas de salud de origen genético, y actúa como laboratorio de referencia nacional y está concebido en el contexto de una red nacional que incluye los centros provinciales de genética médica. El centro es sede del Programa Nacional de Atención a Discapacitados y Desarrollo de la Genética Médica en Cuba.

El CNGM tiene como principales objetivos:

- Mejorar los niveles de salud de nuestro pueblo y disminuir el impacto de las enfermedades con implicación genética en el cuadro de la morbimortalidad del país y realizar aportes al desarrollo de estas ramas de las ciencias, teniendo en cuenta las potencialidades que se derivan de su integración.
- Coordinar nacionalmente la actividad de asistencia médica en la red de centros y servicios de Genética Médica, así como orientar, evaluar y controlar la ejecución de los distintos subprogramas del Programa de Diagnóstico y Prevención de Enfermedades Genéticas en el país.
- Desarrollar los recursos humanos en los campos de la Genética Médica, la Inmunología, la Bioquímica y otras disciplinas afines para mantener el trabajo de la red, el desarrollo de la genética y los servicios de salud en el país.

- Fungir como centro rector metodológico en la formación de recursos humanos en las especialidades de Genética Clínica y la Inmunología, así como desarrollar recursos técnicos para la educación y formación de una cultura genética en la población.(12)

En Cuba se brinda servicio de consultas genéticas hasta el nivel municipal, los datos recogidos son almacenados de forma aislada en todo el país y no existe un registro centralizado en el cual se almacene toda la información, o sea que hacer un estudio o sacar una estadística nacional demoraría mucho y costaría bastante trabajo.

En el curso 2006 - 2007, la Universidad de Ciencias Informáticas, en conjunto con el CNGM, desarrolló RECUEGEN v1.0, un sistema informático donde los genetistas podrían hacer búsquedas de pacientes por distintos criterios de selección, obtener estadísticas detalladas y hacer estudios de la población, pero dicho sistema no estaba integrado a la arquitectura de salud, lo cual es una característica que debe cumplirse para los sistemas acoplados a la red INFOMED, razón por la cual no se pudo acoplar a la Red de Salud Cubana, faltaron aspectos de búsqueda a la hora de implementar los reportes y el CNGM quiso que también cumpliera con otros requisitos.

Teniendo en cuenta lo presentado anteriormente se ha identificado como **problema científico**: ¿Cómo obtener un producto funcional a partir de los requerimientos identificados para el Registro Cubano de Enfermedades Genéticas integrado a la Red Nacional de Salud?

Este problema se enmarca en el **objeto de estudio**: Proceso de gestión de información de la salud.

El objeto de estudio delimita el **campo de acción**: Proceso de desarrollo de software para la gestión de la información de enfermedades genéticas del Sistema Informático para la Red Nacional de Genética Médica.

Para dar solución al problema se plantea como **objetivo general**: Desarrollar el diseño y la implementación del Registro Cubano de Enfermedades Genéticas del Sistema Informático para la Red Nacional de Genética Médica.

A partir del análisis del objetivo general se derivaron los siguientes **objetivos específicos:**

- Diseñar el Registro Cubano de Enfermedades Genéticas.
- Implementar el Registro Cubano de Enfermedades Genéticas.

Para lograr los objetivos planteados, se definen las siguientes **tareas a cumplir**:

- Estudio del RECUEGEN v1.0, y de otros registros de enfermedades genéticas en Cuba y el mundo.
- Realización de entrevistas con los especialistas del CNGM.
- Estudio de la arquitectura de software establecida para el desarrollo de sistemas informáticos para el Ministerio de Salud Pública (MINSAP).
- Estudio de las tecnologías a utilizar.
- Diseño de las clases del Registro Cubano de Enfermedades Genéticas.
- Elaboración del Modelo de Diseño del Registro Cubano de Enfermedades Genéticas.
- Implementación de las funcionalidades del Registro Cubano de Enfermedades Genéticas.
- Validación del registro a nivel de desarrollador.

Capítulo 1. Fundamentación teórica.

Se describe el estado del arte, se analizan los procesos que son objeto de automatización así con el estudio de las tendencias, herramientas, técnicas y metodologías a usar en el diseño, implementación y desarrollo del sistema. Se describen los roles que se desempeñarán en el proceso de desarrollo del software.

Capítulo 2. Diseño del sistema.

En este capítulo se identifican los requisitos funcionales que debe cumplir el sistema informático. También se presenta el modelo de casos de uso del sistema (MCUS) y el diseño del sistema en el que se presentarán los diagramas de clases del diseño con estereotipos Web, los diagramas de interacción (secuencia), así como el modelo de despliegue.

Capítulo 3. Implementación.

En este capítulo se presenta el modelo de implementación, contiene los diagramas de componentes del registro, se brinda una descripción de los principales métodos implementados y se muestran imágenes de la interfaz del registro.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Estado del arte

En el área de la genética se han logrado avances con la creación de varios sistemas como son el registro de algunos trastornos de tipo genético. Dichos registros permiten a los investigadores estudiar la compleja interrelación entre el estilo de vida, los genes y el medio ambiente. Una vez que haya recolectado y almacenado todas las muestras de sangre y orina y la información genética que se busca, es mucho más fácil el estudio de los diferentes padecimientos y poder prevenirlos.

Registros de Enfermedades Genéticas en el Mundo.

En Internet existen numerosas bases de datos con información genética de diversos tipos. Algunas sólo almacenan información de secuencias de ácido desoxirribonucleico (ADN). Otras bases de datos guardan información acerca de las mutaciones presentes en estas secuencias de ADN, la frecuencia con la que ocurren en distintas poblaciones, etc. La información acerca de la localización de estas secuencias de ADN en cromosomas, así como también la información sobre marcadores cercanos que pueden ser de utilidad para el diagnóstico.

Ejemplo de bases de datos:

OMIM - Online Mendelian Inheritance in Man: Es un catálogo de genes humanos y desórdenes genéticos. Hay referencias históricas y bibliográficas sobre enfermedades, referencias médicas, links a otras bases de datos, etc. Las búsquedas pueden retornar información sobre enfermedades o genes indistintamente.

GeneDis - Human Genetic Disease Database. Bases de datos de enfermedades Genéticas, funciona comparando las secuencias de proteínas ó ADN, por enfermedades o por comparación de proteínas con secuencias de ADN de GeneDis.

GeneClinics ·GeneTests Brinda Información sobre enfermedades genéticas y recursos sobre test genéticos. La aplicación de test genéticos para diagnosis, revisión, información genética de pacientes y familiares con desordenes genéticos, y proporcionar links para relacionarlos con bases de datos.

Estas bases de datos están principalmente enfocadas y dirigidas a guardar información de las enfermedades pero por su estructura, composición, patrones, no para hacer estudios de prevalencia en la población, que es lo que Cuba necesita. Por esta razón, a pesar de no constar con una base de datos que almacene la información de las enfermedades genética ni tampoco con una aplicación que

gestione este tipo de información, no se asumen estas bases de datos como propuestas de solución al problema que se presenta, lo que ha impulsado al CNGM a realizar un Registro Cubano de Enfermedades Genéticas. (1)

Un ejemplo importante de estos registros en el mundo es el primer "biobanco" genético, que cuenta con un estudio realizado en Inglaterra de 500 000 personas entre 40 y 50 años con el objetivo de mantener un registro de la salud, a través del estudio del ADN.

En nuestro país se ha dado prioridad a este tema y se cuenta hoy con registros que facilitan el estudio y la prevención de las enfermedades genéticas, podemos citar dos de ellos:

- Registro genético preventivo automatizado de una enfermedad autosómica dominante (RGP)
- Registro de enfermos con Fibromialgia y/o Síndrome de Fatiga Crónica FM/SFC

En el año 2007, la Universidad de Ciencias Informáticas, en conjunto con el CNGM, desarrolló RECUEGEN v1.0, un sistema informático para gestionar el registro de enfermedades genéticas, dicha versión concluyó con la implementación de un sistema capaz de realizar estadísticas y buscar información sobre los pacientes; pero dicho sistema no estaba integrado a la arquitectura de salud, lo cual es una característica que debe cumplirse para los sistemas acoplados a la red INFOMED, también faltaron aspectos de búsqueda a la hora de implementar los reportes. Por lo que el centro pidió que se realizara la versión 2.0.

1.2. Tendencias y tecnologías en la actualidad.

Para el desarrollo de este sistema se llevó a cabo un estudio para seleccionar cuales serían las herramientas adecuadas para el trabajo, teniendo en cuenta los requisitos y características que debe cumplir el sistema al ser integrado a la red de INFOMED.

1.2.1. Metodología de desarrollo del software

Para la construcción de este sistema el equipo de desarrollo del Registro de Enfermedades Genéticas decidió utilizar como metodología de desarrollo del software Rational Unified Process (RUP).

RUP es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un software. Sin embargo; es más que un simple proceso, es un marco de trabajo genérico que puede especializarse para una gran variedad de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. RUP es una guía de cómo usar UML de la forma más efectiva. Existen tres características claves presentes en RUP, ellas son:

1. Dirigido por casos de uso: Teniendo en cuenta que la razón de ser de un sistema es brindar servicios a los usuarios, define caso de uso como el conjunto de acciones que debe realizar un sistema para dar un resultado de valor a un determinado usuario y los utiliza tanto para especificar los requisitos funcionales del sistema, como para guiar todos los demás pasos de su desarrollo, dígase diseño, implementación y prueba.

2. Centrado en la arquitectura: La arquitectura es una vista del diseño completo con las características más importantes, dejando a un lado los detalles. Esta no solo incluye las necesidades de los usuarios e inversores, sino también otros aspectos técnicos como el hardware, sistema operativo, sistema de gestión de base de datos, protocolos de red; con los que debe coexistir el sistema. En otras palabras, la arquitectura representa la forma del sistema, la cual va madurando en su interacción con los casos de uso hasta llegar a un equilibrio entre funcionalidad y características técnicas.

3. Iterativo e incremental: La alta complejidad de los sistemas actuales hace que sea factible dividir el proceso de desarrollo en varios mini-proyectos. Cada uno de estos mini-proyectos se les denomina iteración y pueden o no representar un incremento en el grado de terminación del producto completo. En cada iteración los desarrolladores seleccionan un grupo de casos de uso, los cuales se diseñan, implementan y prueban. La iteración controlada reduce el riesgo de no sacar al mercado el producto en el calendario previsto, permite mantener la motivación del equipo, pues puede ver avances claros a corto plazo y que el desarrollo pueda adaptarse a los cambios en los requisitos.

Fases e hitos de RUP

RUP divide el proceso de desarrollo en ciclos, teniendo un producto al final de cada ciclo. Cada ciclo se divide en cuatro fases, y cada fase concluye con un hito bien definido.

1. Inicio: Descripción del negocio y alcance del proyecto.

2. Elaboración: Se define la arquitectura del sistema.

3. Construcción: Funcionalidad operativa del software.

4. Transición: Liberación del software (Puede implicar reparación de errores).

"Un proceso de desarrollo de software define quién hace qué, cómo y cuándo. RUP define cuatro elementos, los roles que responden a la pregunta ¿Quién?, las actividades que responden a la pregunta ¿Cómo?, los productos que responden a la pregunta ¿Qué? y los flujos de trabajo de las disciplinas que responde a la pregunta ¿Cuándo?" (9)

Los roles son una definición abstracta de un grupo de responsabilidades que deben llevar a cabo ciertas actividades del proceso y producir algunos documentos. No son individuos, ni son necesariamente equivalentes a títulos profesionales. En su lugar, los roles describen cómo los individuos a los que se les asignan deberán comportarse en el contexto del proyecto.

Una actividad es una unidad de trabajo que una persona que desempeñe un rol puede realizar. Las actividades tienen un objetivo concreto, normalmente expresado en términos de crear o actualizar algún producto.

Un artefacto es la información producida, modificada o usada durante el proceso de desarrollo de software. Los productos son los resultados tangibles del proyecto, las cosas que va creando y usando hasta obtener el producto final.

Un flujo de trabajo es una relación de actividades que producen resultados observables dado por una secuencia de actividades realizadas por los diferentes roles.

- Modelado del negocio: Con este flujo de trabajo se pretende llegar a un mejor entendimiento de la organización donde se va a establecer el producto.

- Requerimientos: Éste es uno de los flujos de trabajo más importantes, porque en él se establece qué tiene que hacer exactamente el sistema que se construya. En esta línea los requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que se especifiquen.

- Análisis y Diseño: El análisis consiste en obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos.

- Implementación: En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y además se deben hacer las pruebas de unidad: cada implementador es responsable de probar las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable.

- Pruebas: Este flujo de trabajo es el encargado de evaluar la calidad del producto que se está desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

- Despliegue: Este flujo de trabajo se desarrolla con mayor intensidad en la fase de transición, ya que el propósito del flujo es asegurar una aceptación y adaptación sin complicaciones del software por parte de los usuarios. Su ejecución inicia en fases anteriores, para preparar el camino, sobre todo con actividades de planificación, en la elaboración del manual de usuario y tutoriales.

- Gestión del proyecto: Es el arte de lograr un balance al gestionar objetivos, riesgos y restricciones para desarrollar un producto que sea acorde a los requisitos de los clientes y los usuarios.

- Configuración y control de cambios: La finalidad de este flujo de trabajo es mantener la integridad de todos los artefactos que se crean en el proceso, así como de mantener información del proceso evolutivo que han seguido.

- Entorno: La finalidad de este flujo de trabajo es dar soporte al proyecto con las adecuadas herramientas, procesos y métodos. Brinda una especificación de las herramientas que se van a necesitar en cada momento, así como definir la instancia concreta del proceso que se va a seguir.

1.2.2. UML (Unified Modeling Lenguaje)

Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema.

Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo.

El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.

UML no es un lenguaje de programación. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguajes de programación, así como construir modelos por ingeniería inversa a partir de programas existentes.

Es un lenguaje de propósito general para el modelado orientado a objetos. UML es también un lenguaje de modelamiento visual que permite una abstracción del sistema y sus componentes.

Existían diversos métodos y técnicas Orientadas a Objetos, con muchos aspectos en común pero utilizando distintas notaciones, se presentaban inconvenientes para el aprendizaje, aplicación, construcción y uso de herramientas, etc., además de pugnas entre enfoques, lo que generó la creación del UML como estándar para el modelamiento de sistemas de software principalmente, pero con posibilidades de ser aplicado a todo tipo de proyectos. (1)

Dentro de los objetivos se encuentran:

- UML es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática.

- UML no pretende ser un método de desarrollo completo. No incluye un proceso de desarrollo paso a paso. UML incluye todos los conceptos que se consideran necesarios para utilizar un proceso moderno iterativo, basado en construir una sólida arquitectura para resolver requisitos dirigidos por casos de uso.

- Ser tan simple como sea posible pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesita construir. UML necesita ser lo suficientemente expresivo para manejar todos los conceptos que se originan en un sistema moderno, tales como la concurrencia y distribución, así como también los mecanismos de la ingeniería de software, como son la encapsulación y componentes.

    - Debe ser un lenguaje universal, como cualquier lenguaje de propósito general. Imponer un estándar mundial. (1)

1.3. Herramientas

La aplicación a desarrollar constituirá un sistema informático más para el ministerio nacional de salud pública (MINSAP). Existe un conjunto de normas que deben cumplir los sistemas informáticos del MINSAP, dentro de las cuales se encuentran hacer uso de las siguientes tecnologías:

1) Servidor HTTP Apache 2.0

2) Lenguaje de programación del lado del servidor PHP 5

3) Sistema gestor de base de datos MySQL 5.0

Por lo cual para en el desarrollo del sistema se utilizarán las tecnologías anteriormente mencionadas. (2)

Se utiliza el Visual Paradigm para modelar en notación UML a través del cual se obtienen los diagramas a utilizar, en el desarrollo del sistema informático contamos con la ayuda del framework Symfony que hace más rápido y simple el trabajo. Como IDE usamos Eclipse y el sistema de control de versiones Subversion. Estas herramientas se definieron en la línea base de la arquitectura para el SIGM.

1.3.1. Herramientas CASES: Visual Paradigm

Las Herramientas CASE (**C**omputer **A**ided **S**oftware **E**ngineering, Ingeniería de Software Asistida por Computadoras) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software. (2)

Una potente herramienta CASE que utiliza como lenguaje de modelación el UML. La herramienta fue desarrollada para una amplia gama de usuarios incluyendo Ingenieros de Software, analistas de sistemas, analistas del negocio y arquitectos de sistemas. Proporciona a los desarrolladores una plataforma con interfaz amigable que les permite diseñar un producto con calidad de forma rápida.

1.3.2. Framework (Symfony)

El framework seleccionado es Symfony, un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Razones por la que fue seleccionado:

1) Gran documentación en varios idiomas (Libro gratuito + Documentación de la API + Wiki).

2) Desarrollo constante.

3) Gran comunidad de usuarios.

4) Flexibilidad (Diseño + configuración + plugins).

5) Reutilización de componentes probados en vez de “inventar de nuevo la rueda”.

1.3.3. Entorno de desarrollo: Eclipse PDT

Eclipse es un IDE para todo tipo de aplicaciones, inicialmente desarrollado por IBM, y actualmente gestionado por la Fundación Eclipse. La característica clave de Eclipse es la extensibilidad. Eclipse es una gran estructura formada por un núcleo y muchos plugins que van conformando la funcionalidad final. La forma en que los plugins interactúan es mediante interfaces o puntos de extensión; así, los nuevos aportes se integran sin dificultad ni conflictos.

Se seleccionó este último IDE para el desarrollo de la herramienta principalmente:

-Permite el desarrollo de aplicaciones utilizando Subversion como sistema de control de versiones.

-Por el potente editor de código que presenta.

-Por brindar la facilidad de agregarles las librerías de los framework a utilizar y una vez agregadas realizar completamiento de código del framework. (2)

1.3.4. Quanta

Quanta es una herramienta de desarrollo web para el entorno de escritorio KDE. Está diseñado para el desarrollo web y rápidamente se está convirtiendo en un editor maduro y con gran variedad de características, entre las que se encuentran:

- Usa KIO para FTP, SSH (con FISH) y soporta otros protocolos.
- Asistentes para creación de tablas, enlaces y páginas en blanco.
- Resaltado de sintaxis de HTML, Javascript, CSS y varios más.
- Contiene un analizador que informa acerca de la correcta creación de nuestras páginas.

- Soporta plugins a través de KParts por defecto son: Konsole, KImageMapEditor, KLinkStatus, Cervisia (CVS) y KFileReplace.

Previsualización: Los documentos pueden ser previsualizados dentro de la aplicación usando el motor KHTML. Es posible preprocesar los documentos a través de un servidor web antes de previsualizar. (7)

1.3.5. Servidores Web: Apache

Un servidor Web es un programa que sirve para atender y responder a las diferentes peticiones de los navegadores, este sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante protocolo HTTP o HTTPS. (1)

En el mundo de los servidores web, Apache ha ganado la mayoría de adeptos, y uno de las principales características por la que este servidor Web ha ganado tanto es por su seguridad y disponibilidad, Apache está disponible para una gran multitud de plataformas:

- FreeBSD, NetBSD, OpenBSD, ...
- GNU/Linux
- Mac OS y Mac OS X Server
- Netware
- Open Step/Match
- UNIX comerciales como AIX (R), Digital UNIX (R), HP-UX (R),IRIX (R), SCO (R), Solaris (R), SunOS (R), UnixWare (R)
- Windows (R)

Además Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor y es posible configurarlo para que ejecute un determinado script cuando ocurra un error en concreto.

Apache está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Estas hacen que a menudo sean necesarias diferentes características o funcionalidades, o que una misma característica o funcionalidad sea implementada de diferente manera para obtener una mayor eficiencia. Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular. Este diseño permite a los administradores de sitios web elegir que características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor. (1)

Apache puede soportar de una forma más fácil y eficiente una amplia variedad de sistemas operativos.

El servidor puede personalizarse mejor para las necesidades de cada sitio web.

Dentro de sus puntos fuertes se encuentran:

- Tiene interfaz con todos los sistemas de autenticación.
- Facilita la integración como "plugins" de los lenguajes de programación de Páginas Web dinámicas más comunes.
- Tiene integración en estándar del protocolo de seguridad SSL.
- Provee interfaz a todas las bases de datos. (1)

1.3.6. Sistema de control de versiones: Subversion

Un sistema de control de versiones es un software que administra el acceso a un conjunto de ficheros y mantiene un historial de cambios realizados. El control de versiones es útil para guardar cualquier documento que cambie con frecuencia, como el código fuente de un programa. Normalmente consiste en una copia maestra en un repositorio central, y un programa cliente con el que cada usuario sincroniza su copia local. Esto permite compartir los cambios sobre un mismo conjunto de ficheros. Además, el repositorio guarda registro de los cambios realizados por cada usuario, y permite volver a un estado anterior en caso de necesidad. (2)

Subversion es un controlador de versiones empleado en la administración de archivos utilizados en el desarrollo de software o contenido. CVS -- considerado su antecesor -- es uno de los controladores de

versiones más utilizados en proyectos de software libre, sin embargo, a pesar de su amplio uso, el mismo diseño de CVS resultó ineficiente para diversos grupos de usuarios, y ante estas inconformidades se dio inicio al proyecto que hoy es conocido como: Subversion, mismo que ha empezado a socavar el dominio de CVS. (3)

1.4. Lenguaje de programación del lado del servidor: PHP

PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas, similar a ASP o el JSP, embebido en páginas HTML y ejecutado en el servidor.

La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. No es un lenguaje de marcas como podría ser HTML, XML o WML. Está más cercano a Java Script o a C, para aquellos que conocen estos lenguajes. (4)

Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas web dinámicas:

- Soporte para una gran cantidad de bases de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, Sybase, MySQL, Informix, entre otras.
- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- Ofrece una solución simple y universal para las paginaciones dinámicas del Web de fácil programación.
- Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.
- Soportado por una gran comunidad de desarrolladores, como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente.

- El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.

Con PHP se puede hacer cualquier cosa que podemos realizar con un script CGI, como el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas.

PHP es un potente lenguaje y el intérprete, tanto incluido en el servidor Web como módulo o ejecutado como un binario CGI, puede acceder a ficheros, ejecutar comandos y abrir comunicaciones de red en el servidor. Todas estas características hacen que lo que se ejecute en el servidor Web sea seguro por defecto.

PHP ha sido diseñado específicamente para ser un lenguaje más seguro para escribir programas CGI, Perl o C y con la correcta selección de las opciones de configuración de tiempo de compilación y ejecución se consigue la exacta combinación de libertad y seguridad que se necesita. Ya que existen diferentes modos de utilizar PHP, existe también una multitud de opciones de configuración que permiten controlar su funcionamiento. Una gran selección de opciones garantiza que se pueda usar PHP para diferentes aplicaciones, pero también significa que existen combinaciones de estas opciones y configuraciones del servidor que producen instalaciones inseguras. (4)

1.5. Lenguaje de programación del lado del Cliente: Java Script

Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado.

Con Java script podemos crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones Java script y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador.

Java script es el siguiente paso, después del HTML, que puede dar un programador de la web que decida mejorar sus páginas y la potencia de sus proyectos. Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez, a veces con ligereza. Incluso las personas que

no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con solo un poco de práctica.

Entre las acciones típicas que se pueden realizar en Java script tenemos dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, java script nos permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que podemos crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo. (1)

Java script es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. Además, Java script pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente.

Con Java script el programador, que se convierte en el verdadero dueño y controlador de cada cosa que ocurre en la página cuando la está visualizando el cliente. (1)

1.6. Sistema gestor de Base de datos: MySQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca.

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. (5)

Las principales características de este gestor de bases de datos son las siguientes:

1. Probado con un amplio rango de compiladores diferentes.
2. Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
3. Soporta gran cantidad de tipos de datos para las columnas.
4. Dispone de API's en gran cantidad de lenguajes (C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, etc.).
5. Gran portabilidad entre sistemas.
6. Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.
7. Proporciona sistemas de almacenamiento transaccional y no transaccional.
8. Un sistema de reserva de memoria muy rápido basado en threads.
9. Joins muy rápidos usando un multi-join de un paso optimizado.
10. Usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice.
11. Soporta hasta 32 índices por tabla.
12. Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.
13. ¡Es libre! (5)

1.7. Patrones de Arquitectura

Se utilizará el clásico patrón Modelo – Vista – Controlador (MVC: Model – View – Controler). El cual es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:

1) **Modelo**: Gestiona el comportamiento y los datos de la aplicación, responde a las peticiones que realizan las vistas sobre su estado y permite su actualización normalmente desde el controlador.

2) **Vista:** Muestra el estado al usuario de la aplicación, redirigiendo las acciones que realiza sobre el interfaz al controlador.

3) **Controlador**: Interpreta las acciones del usuario, accediendo a las operaciones de negocio de la aplicación y modificando a partir de sus resultados el estado del modelo y la navegación entre vistas. (2)

1.8. Roles y artefactos

1.8.1. Rol Diseñador

En la metodología RUP el diseñador es el responsable de diseñar una parte del sistema cumpliendo con las restricciones de los requerimientos, arquitectura y proceso de desarrollo del proyecto, identifica y define las responsabilidades, operaciones, atributos y relaciones de los elementos de diseño

Los artefactos que realiza el diseñador que se obtendrán en el presente trabajo son:

- **Realización de casos de uso del diseño:** Es una colaboración en el modelo de diseño que describe como se realiza un caso de uso especifico, y como se ejecuta en términos de casos de uso del diseño. Una realización de caso de uso del diseño proporciona una traza directa a una realización de caso de uso del análisis en el modelo de análisis.

- **Clases del análisis**: las clases de análisis representan un modelo de concepto tempranamente para cosas en el sistema que tienen responsabilidades y comportamiento.

- **Paquetes de diseño**: Es una colección de clases, relaciones, realizaciones de casos de uso, diagramas y otros paquetes que estén de alguna forma relacionados. Es usado para estructurar el modelo de diseño dividiéndolo en partes más pequeñas.

- **Diagrama de clases:** Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema, esto incluye modelar el vocabulario del sistema, modelar las colaboraciones o modelar esquemas. Los diagramas de clases también son la base para un par de diagramas relacionados: los diagramas de componentes y los diagramas de despliegue. Los diagramas de clases son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa.

- **Clases del diseño**: Una clase es una descripción de un conjunto de objetos que comparten las mismas responsabilidades, las relaciones, las operaciones, atributos, y la semántica. (6)

1.8.2. Rol Implementador

El rol Implementador es responsable de desarrollar y de probar componentes de acuerdo con los estándares adoptados del proyecto para la integración en subsistemas más grandes.

El artefacto que realiza el implementador que se obtendrá en el presente trabajo es:

Elementos de implementación: los elementos de implementación son la parte física de la implementación, incluyen los archivos y directorios. Incluyen ficheros de código (fuentes, binarios o ejecutables), ficheros de datos y de documentación como ficheros de ayuda online. (6)

1.9. Conclusión

Se realizó un estudio detallado de las metodologías, herramientas y tecnologías que se usarán en el proyecto, así como sus principales ventajas y desventajas. Se hizo un profundo análisis acerca de la política de software que están vigente en el Centro de Genética Medica, así como la política que se ha trazado la facultad, se tomó además la decisión de trabajar con herramientas basadas en software libre que faciliten el trabajo y permitan el desarrollo de un proyecto libre de licencias.

Planteadas las premisas anteriores podemos concluir que el estudio de las enfermedades genéticas y el desarrollo de un sistema que sea capaz de gestionar toda información que de ellas se derive son objetivos principales del sistema de salud cubano y en particular de nuestros compañeros de proyecto que trabajan día a día para lograr un sistema con la calidad requerida por tan prestigiosa instalación.

CAPÍTULO 2: DISEÑO DEL SISTEMA

2.1. Introducción.

En este capítulo se identifican los requisitos funcionales que debe cumplir el sistema informático. También se presenta el modelo de casos de usos del sistema (MCUS) y el diseño del sistema en el que se presentarán los diagramas de clases del diseño con estereotipos Web, los diagramas de interacción (secuencia), así como el diseño del modelo de despliegue.

2.2. Características del sistema

2.2.1. Requisitos funcionales:

1. Gestionar reportes estadísticos

1.1. Tasa de Prevalencia
- 1.1.1. Por tiempo
  - 1.1.1.1. Rango de fechas
- 1.1.2. Por rango de edades
- 1.1.3. Región
  - 1.1.3.1. Nacional
  - 1.1.3.2. Por provincia
  - 1.1.3.3. Por municipio
- 1.1.4. Por género
- 1.1.5. Por enfermedad

1.2. Tasa de Letalidad
- 1.2.1. Por tiempo
  - 1.2.1.1. Rango de fechas
- 1.2.2. Por rango de edades
- 1.2.3. Región
  - 1.2.3.1. Nacional
  - 1.2.3.2. Por provincia

1.2.3.3. Por municipio

1.2.4. Por género

1.2.5. Por enfermedad

1.3. Tasa de Morbilidad

1.3.1. Por tiempo

1.3.1.1. Rango de fechas

1.3.2. Por rango de edades

1.3.3. Región

1.3.3.1. Nacional

1.3.3.2. Por provincia

1.3.3.3. Por municipio

1.3.4. Por género

1.3.5. Por enfermedad

1.4. Tasa de Mortalidad

1.4.1. Por tiempo

1.4.1.1. Rango de fechas

1.4.2. Por rango de edades

1.4.3. Región

1.4.3.1. Nacional

1.4.3.2. Por provincia

1.4.3.3. Por municipio

1.4.4. Por género

1.4.5. Por enfermedad

1.5. Mostrar resultados en formato tabla

1.5.1. Imprimir tabla

2. Gestionar reportes de Cálculos Totales

2.1. Por tiempo

2.1.1. Rango de fechas

2.2. Por rango de edades

2.3. Región

2.3.1. Nacional
2.3.2. Por provincia
2.3.3. Por municipio
2.4. Por género
2.5. Por enfermedad
2.6. Mostrar resultados en formato tabla
2.6.1. Imprimir tabla

3. Gestionar enfermedades genéticas

3.1. Buscar y Visualizar Enfermedad.
3.2. Insertar Enfermedad.
3.3. Modificar Enfermedad.
3.4. Eliminar Enfermedad
3.5. Insertar clasificación de enfermedad.
3.6. Buscar y Visualizar clasificación de enfermedad.
3.7. Modificar clasificación de enfermedad.

2.2.2. Requerimientos no funcionales.

El registro debe integrarse como un componente más de SISalud (Sistema de Información para la Salud), para lo cual deberá ser registrado por el mismo, e interactuar con otros componentes consumiendo sus servicios, o brindando los propios, utilizando para ello el componente de seguridad SAAA (Componente de seguridad basado en el modelo de Autenticación, Autorización y Auditoría).
Además, el registro debe cumplir ciertas características que se resumen en la siguiente lista de requisitos no funcionales:

1. *Apariencia o interfaz externa:*

Se deben utilizar imágenes y colores identificados con el negocio del sistema. La interfaz externa debe estar diseñada para verse en cualquier resolución igual o superior a 1024x768.

2. *Usabilidad:*

La aplicación informática debe garantizar un acceso fácil y rápido, contando con un menú que satisfaga las necesidades de los usuarios. Este podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de una computadora y del ambiente web.

3. *Rendimiento:*

Los tiempos de respuestas deben ser generalmente rápidos al igual que la velocidad de procesamiento de la información.

4. *Soporte:*

Se debe asegurar el soporte para los usuarios de manera que se puedan satisfacer sus necesidades a partir de mejoras, una vez puesta en marcha la aplicación. Para ello se crearán una serie de manuales de usuarios y videos tutoriales, y se mantendrá la asistencia a los usuarios.

5. *Seguridad:*

El sistema debe tener un mecanismo propio para gestionar la seguridad a través de niveles de acceso a la información. Los permisos al ejecutar cualquier acción deben estar de acuerdo con el nivel jerárquico de acceso que presente el usuario en cada módulo, el cual es definido por los administradores del sistema.

6. *Software:*

Se requiere para el funcionamiento del sistema disponer de un servidor que cuente con Sistema Operativo Linux, Apache 2.0 y MySQL 5.0 o versiones superiores. Los usuarios del sistema deberán contar con un navegador Internet Explorer 5.5 o Mozilla Firefox 2.0 o superior, para poder acceder a las opciones que brinda el sistema.

7. *Hardware*

Para el desarrollo y ejecución de la aplicación se necesitará:

Para el servidor de aplicación:

• Microprocesador Pentium III a 700 MHz (Recomendado: Pentium IV a 1.7GHz o superior).

• 128 MB de RAM (Recomendado: 512MB o superior).

Para el cliente:

• Microprocesador Pentium a 233 MHz (Recomendado: Pentium a 500MHz o superior)

• 64 MB de RAM (Recomendado: 128 MB RAM o superior)

• 52 MB de espacio de disco duro. (Recomendado: 120MB para la instalación completa del Internet Explorer 5.5)

Conexión al servidor a través de MODEM o tarjeta de red.

Además es necesario contar con una impresora para poder imprimir los diferentes tipos de reportes.

8. *Disponibilidad:* El sistema debe ser capaz de funcionar por si solo en caso de que los servicios de los diferentes componentes de SISalud no estén disponibles. Se debe garantizar el funcionamiento de

la aplicación durante las 24 horas del día y los siete días de la semana, con el menor tiempo posible de recuperación de fallos. Se deben crear copias de respaldo periódicas que puedan restaurar el sistema en caso de fallo crítico o pérdida total de la información.

9. *Requisitos Legales:* Las herramientas y las tecnologías en que estará basada la aplicación informática deberá cumplir con las licencias de software libre.

10. *Persistencia:* La información debe almacenarse en bases de datos con carácter permanente con el objetivo de poder realizar análisis de la misma con el transcurso de los años.

2.2.3. Diagrama de casos de uso del sistema

Este tipo de diagramas representa gráficamente los procesos y su interacción con los actores.

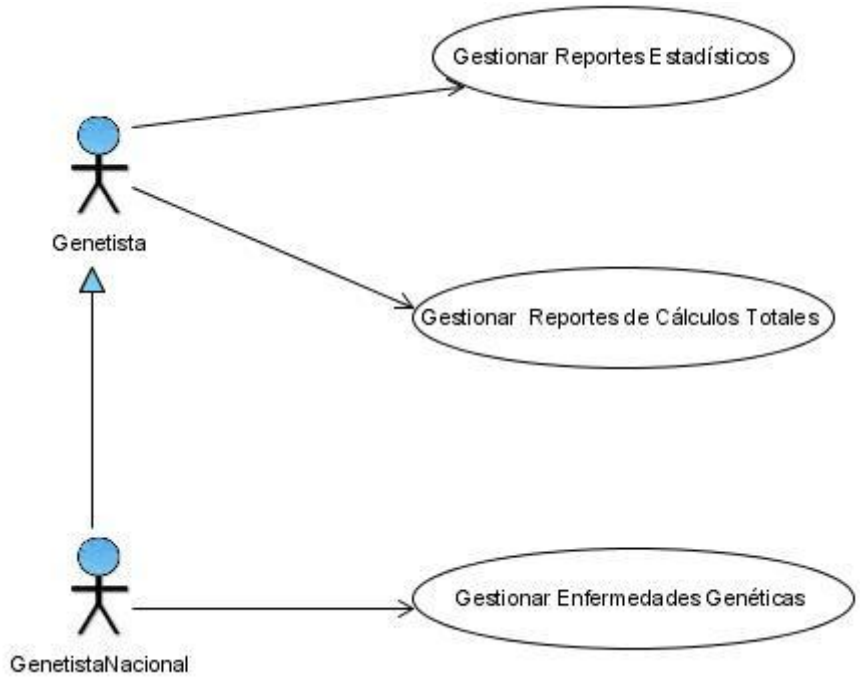

Figura 1. Diagrama de casos de uso del sistema.

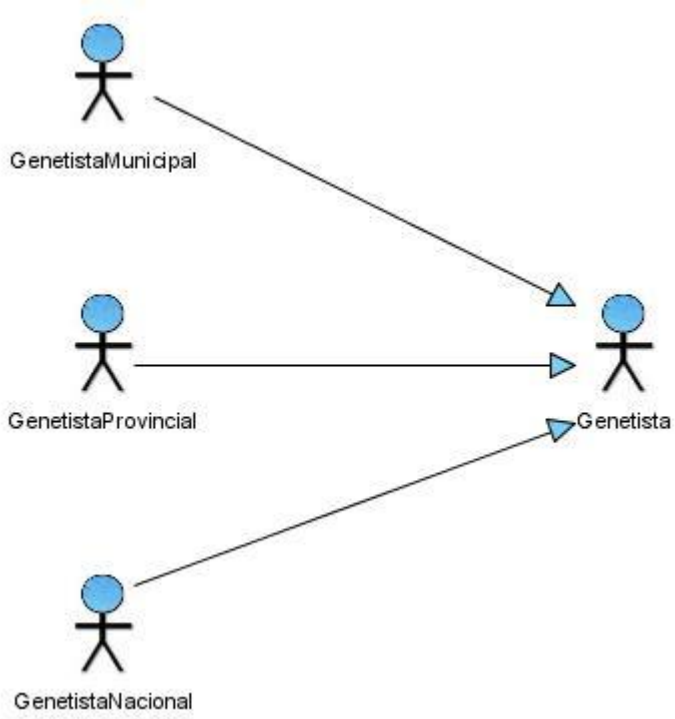

Figura 2. Diagrama de actores del sistema

Actores del sistema.

Actor	Descripción
Genetista	Genetista que desea gestionar reportes tanto estadísticos como de cálculos totales o gestionar enfermedades.
Genetista Nacional	Genetista que tiene la posibilidad de gestionar reportes tanto estadísticos como de cálculos totales o de gestionar enfermedades.
Genetista Provincial	Genetista que tiene la posibilidad de gestionar reportes estadísticos y de cálculos totales.
Genetista Municipal	Genetista que tiene la posibilidad de gestionar reportes estadísticos y de cálculos totales.

Tabla 1: Actores del sistema

2.2.4. Breve descripción de los Casos de Uso del Sistema

Caso de uso Gestionar Reportes Estadístico.

Caso de Uso:	**Gestionar Reporte Estadístico**
Actores:	**Genetista(inicia)**
Resumen:	El caso de uso se inicia cuando el genetista entra al sistema y le pide realizar una estadística. El sistema muestra una interfaz donde el genetista debe escoger el estadígrafo que va a calcular ya sea Tasa de Prevalencia, Tasa de Letalidad, Tasa de Morbilidad, Tasa de Mortalidad, y los parámetros por los que se va a regir que pueden ser fecha de consulta, edad, género, enfermedad genética, municipio o provincia. El genetista establece todas sus opciones y el sistema realiza los cálculos necesarios y muestra la respuesta en el formato tabla y da la opción de imprimir el resultado.
Referencia:	**R1, R1.1- R1.5**

Caso de uso Gestionar Reportes de Cálculos Totales.

Caso de Uso:	**Gestionar Reportes de Cálculos Totales**
Actores:	**Genetista(inicia)**
Resumen:	El caso de uso se inicia cuando el Genetista entra al sistema y va a la sección Calcular totales. El sistema muestra una interfaz donde brinda la opción de escoger los parámetros por los que se va a regir el cálculo. El genetista escoge los parámetros. El sistema realiza el cálculo, muestra los totales en una tabla dependiendo del nivel que se ha escogido (nacional, provincial o municipal) y da la posibilidad de imprimirlos.
Referencia:	**R2, R2.1- R2.6**

Caso de uso Gestionar Enfermedades Genéticas.

Caso de Uso:	**Gestionar Enfermedades Genéticas.**
Actores:	**Genetista Nacional(inicia)**
Resumen:	El caso de uso se inicia cuando el genetista quiere realizar alguna de las siguientes operaciones relacionadas con la gestión de enfermedades genéticas: • Insertar una enfermedad. • Buscar y visualizar enfermedad. • Modificar una enfermedad. • Eliminar una enfermedad. • Insertar una clasificación de enfermedad. • Buscar y visualizar una clasificación de enfermedad. • Modificar una clasificación de enfermedad. El sistema le 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 solicitada.
Referencia:	**R3, R3.1- R3.7**

2.2.5. Descripción detallada de los casos de uso del sistema.

(ver Anexo 1)

2.3. Pautas del Diseño.

Para lograr una agradable apariencia y facilitar el uso del software se definieron algunas pautas de diseño como son:

1. Los formularios que sean creados deben estar centrados, con un ancho de un 90 % y con bordes de valor 0.
2. La primera fila del formulario debe contener el nombre de la operación que se pretende realizar con el formulario.

3. Cuando se le solicita datos al usuario se hará a través de una tabla de dos columnas. En la columna izquierda se ubicará el nombre del dato solicitado al usuario, alineado a la derecha y en la columna de la derecha el componente adecuado para la solicitud.
4. Los botones para efectuar operaciones sobre el formulario se ubicarán en la parte inferior derecha del formulario. Se posicionarán de izquierda a derecha teniendo en cuenta el peso de la operación que representan.
5. Para representar campos que deben ser de entrada obligatoria se colocará al lado derecho del componente en el cual el usuario entrará los datos un asterisco.
6. Los mensajes de error ocurridos durante la validación del formulario se mostrarán en la parte superior del campo validado en el cual ocurrió el error.

Componentes	Estilos
TextBox	Entradaplana
ComboBox	Entradaplana
Botones	Sbttn
ListBox	Entradaplana
NombredeCampos	class="nombrecampo"

Estilos en los componentes web.

En este epígrafe se han plasmado algunas de las pautas de diseño, para una mayor información de las mismas se puede encontrar una descripción mas detallada en el documento de las pautas del diseño definido por el grupo de arquitectura del Sistema Informático de Genética Médica. (8)

2.4. Patrones aplicados

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).

Los patrones son soluciones de sentido común que deberían formar parte del conocimiento de un diseñador experto. Además facilitan la comunicación entre diseñadores, pues establecen un marco de referencia (terminología, justificación). (10)

2.4.1. Patrones GRASP aplicados

- Experto

  Este es uno de los patrones que se utilizan al trabajar con el Framework Symfony, un ejemplo de esto es la inclusión de Propel para mapear la base de datos. Este genera las clases para la gestión de las entidades con las responsabilidades debidamente asignadas según el patrón Experto, pues cada una de estas clases cuenta con un conjunto de funcionalidades relacionadas directamente con la entidad que representan.

- Creador

  La clase egActions es la que contiene las acciones del módulo enfermedades genéticas y es la encargada de ejecutar las mismas. En dichas acciones se crean los objetos de las clases que representan las entidades, por lo cual la clase egActions es “creador” de dichas entidades, por ejemplo, en la acción executeInsertarEnfermedad se evidencia el uso de este patrón mediante la creación de las instancias de las clases entidades que contienen los datos de la enfermedad a insertar (egClasificacionEnfermedadGenetica).

- Alta cohesión

Una de las características principales del Framework Symfony es la organización del trabajo en el mismo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión. Por ejemplo, la clase egActions contiene varias funcionalidades estrechamente relacionadas entre ellas, teniendo un sentido común y un propósito único, siendo estas las encargadas de controlar las acciones de las plantillas. Esto hace posible que el software sea flexible a cambios sustanciales con efecto mínimo.

- Bajo acoplamiento

Este patrón está evidenciado en el Framework Symfony ya que dentro de la capa modelo las clases de abstracción de datos son las más reutilizables y no tienen asociaciones con las clases de la capa vista ni con el controlador.

- Controlador:

Symfony es un framework basado en el patrón arquitectónico Modelo-Vista-Controlador, que define una capa específica para los controladores, que son el núcleo del mismo. Este patrón se puede observar en las clases sfFrontController, sfWebFrontController, sfContex propias de Symfony. Symfony también aplica el patrón “Front Controller” (Controlador frontal) y tiene una estructura bien organizada

de sus controladores que parte desde el la página “indexRMSuccess.php” y se cumplimenta en la clase egActions. Cada clase de esta capa tiene su responsabilidad y es única, por ejemplo, hay controladores que se encargan de la seguridad del sistema trabajando con ficheros .yml.

2.4.2. Patrones GoF aplicados

Patrón Singleton

Este patrón se aplica por ejemplo en la clase sfRouting, ya que es muy utilizada porque es la encargada de enrutar todas las peticiones que se hagan al registro. El singleton sfRouting también define otros métodos que pueden llegar a ser muy útiles para el manejo manual de las rutas como son: el hasRoutes() y el getRoutesByName().

Patrón Command

Este patrón se pone de manifiesto en el método dispatch() de la clase sfWebFrontController, que es la encargada de determinar cual módulo y acción usar en dependencia de la petición del usuario.

Patrón Decorador

En este método de la clase abstracta sfView padre de todas las vistas, tienen cada una un decorador para permitir añadir funcionalidades a las vistas dinámicamente.

Este patrón se observa en el archivo denominado layout.php que contiene el Layout (plantilla global) de todos las páginas del registro. El mismo almacena el código HTML que es común a todas las páginas del registro, para no tener que repetirlo en cada página, por lo que el Layout decora la plantilla.

2.4.3. Patrón de arquitectura aplicado

El framework Symfony utiliza el patrón arquitectónico MVC por lo que obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por este patrón. El código de la presentación se guarda en la vista, el código de manipulación de datos se guarda en el modelo y la lógica de procesamiento de las peticiones constituye el controlador. Aplicar el patrón MVC a una aplicación resulta bastante útil además de restrictivo.

La implementación que realiza Symfony de la arquitectura MVC incluye varias clases como son:

- sfController es la clase del controlador. Se encarga de decodificar la petición y transferirla a la acción correspondiente.

- sfRequest almacena todos los elementos que forman la petición (parámetros, cookies, cabeceras, etc.)
- sfResponse contiene las cabeceras de la respuesta y los contenidos. El contenido de este objeto se transforma en la respuesta HTML que se envía al usuario.
- El singleton de contexto (que se obtiene mediante sfContext::getInstance()) almacena una referencia a todos los objetos que forman el núcleo de Symfony y puede ser accedido desde cualquier punto de la aplicación.

Symfony toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. (11)

2.5. Modelo de diseño.

2.5.1. Clases del diseño.

Nombre de la clase: egActions
Nombre del método: executeRealizarEstadistica()
Descripción: Muestra la interfaz que permite escoger los criterios para el reporte estadístico.
Nombre del método: executeEstadistica()
Descripción: Calcula la estadística y muestra los resultados en una tabla.
Nombre del método: executeCalcularTotales()
Descripción: Muestra la interfaz que permite escoger los criterios para el reporte de calcular totales.
Nombre del método: executeTotales()
Descripción: Calcula los totales según los datos proporcionados y muestra los resultados en una tabla.
Nombre del método: executeBuscarEnfermedades()
Descripción: Muestra la interfaz que permite escoger los criterios para buscar una enfermedad.
Nombre del método: executeMostrarEnfermedades
Descripción: Busca enfermedad.

Nombre del método: executeInsertarEnfermedad
Descripción: Inserta una nueva enfermedad.
Nombre del método: executeInsertada()
Descripción: Muestra un mensaje cuando se ha insertado correctamente una enfermedad.
Nombre del método: executeYaInsertada()
Descripción: Muestra un mensaje cuando ya ha sido insertada anteriormente la enfermedad.
Nombre del método: executeUpdate
Descripción: Modifica una enfermedad.
Nombre del método: executeUpdated()
Descripción: Muestra un mensaje cuando se han modificado correctamente los datos de una enfermedad.
Nombre del método: executeEliminarEnfermedad()
Descripción: Elimina una enfermedad.

2.5.2. Diagramas de clases Web del diseño.

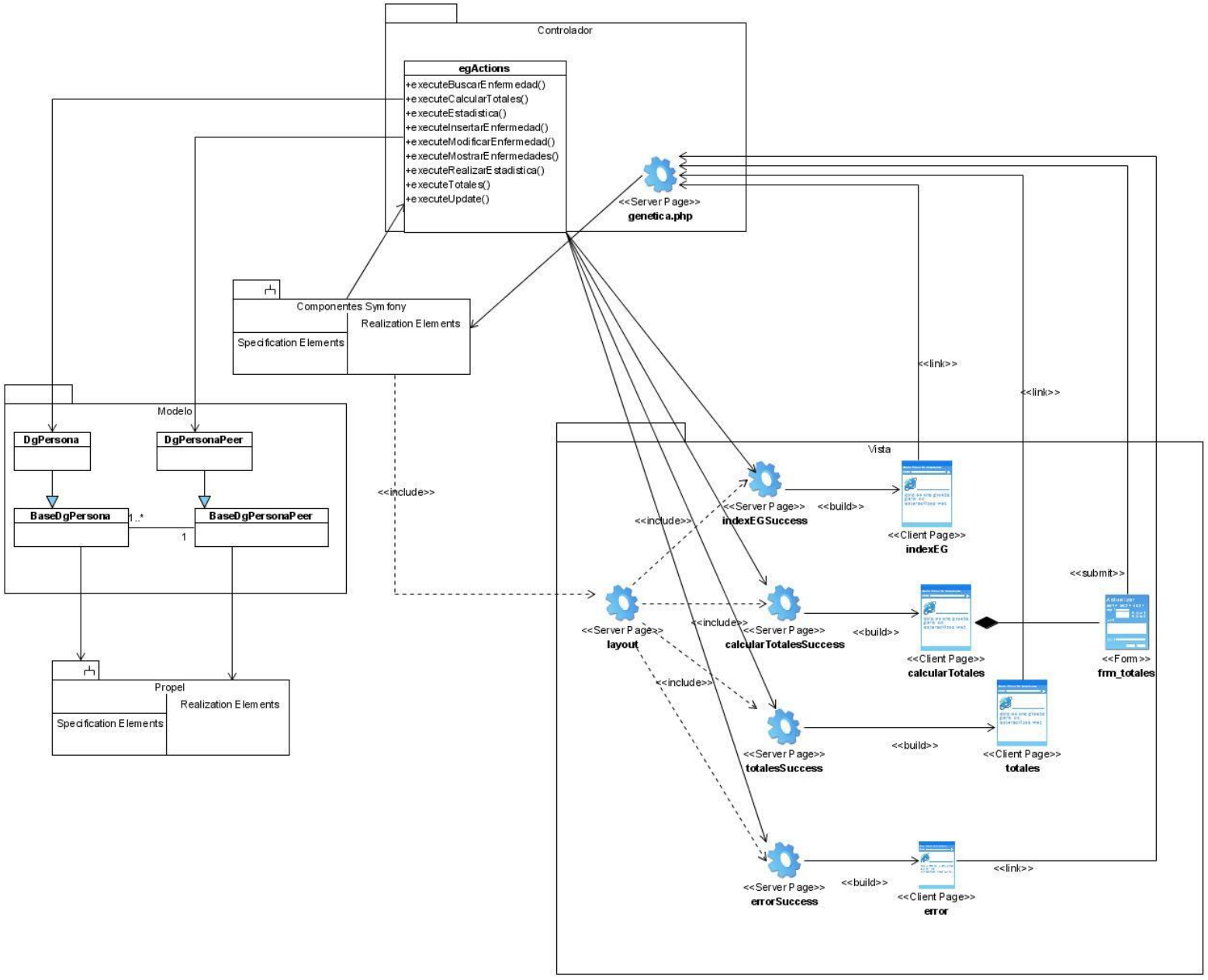

Figura 3. Diagrama de clases del diseño caso de uso: Gestionar reporte de cálculos totales.

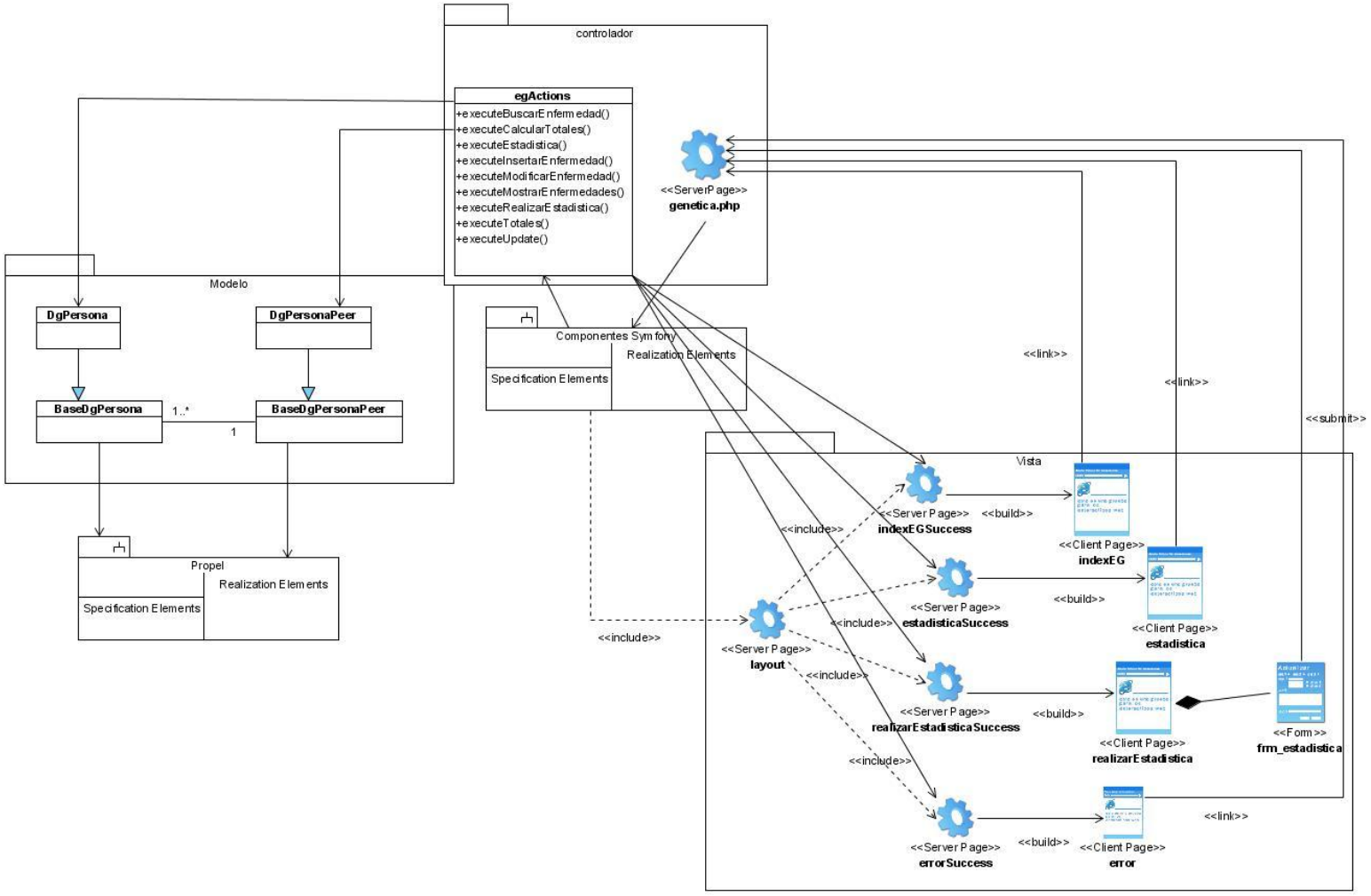

Figura 4. Diagrama de clases del diseño caso de uso: Gestionar reporte estadístico.

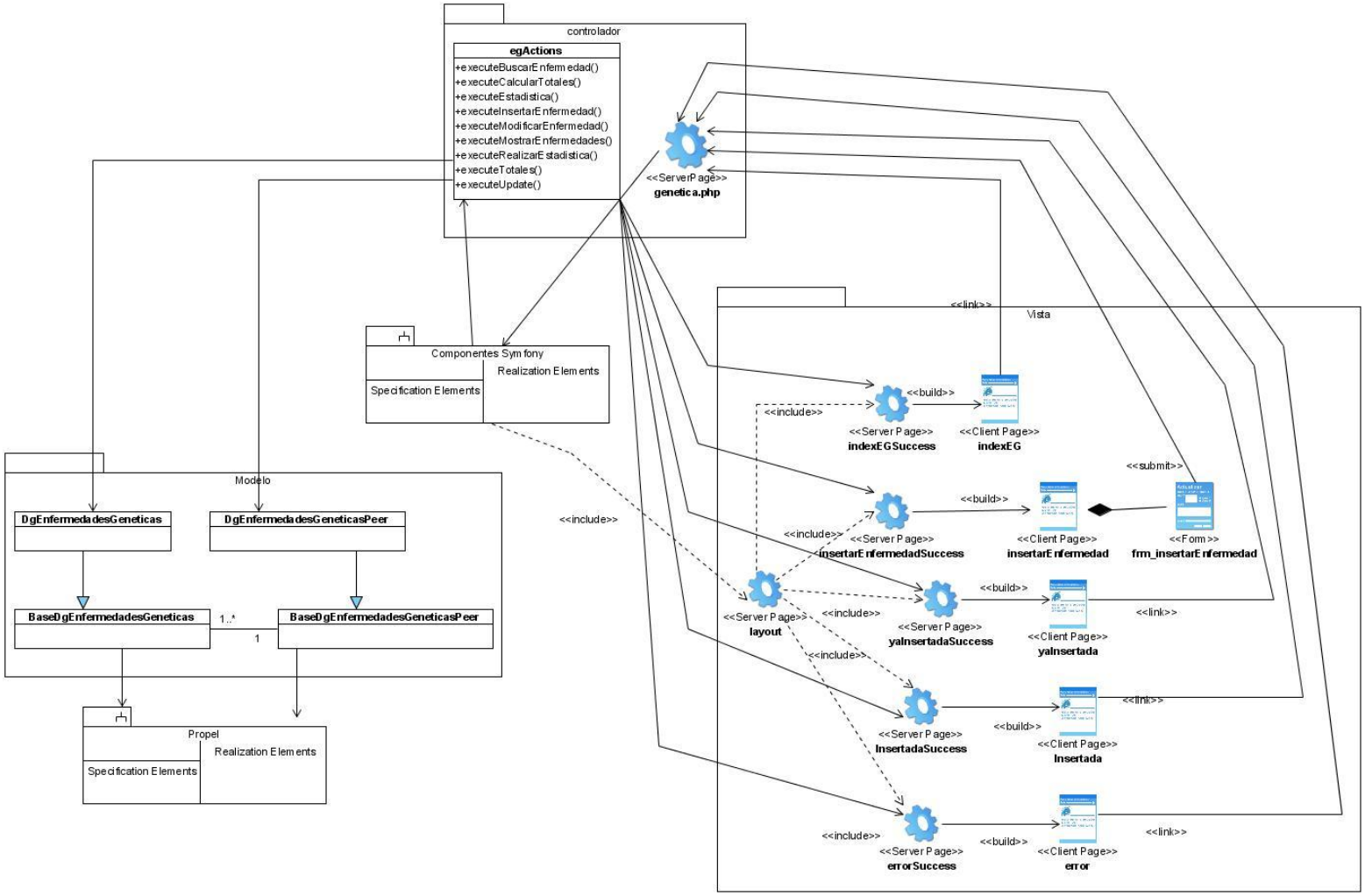

Figura 5. Diagrama de clases del diseño caso de uso: Gestionar enfermedades: Sección Insertar enfermedad.

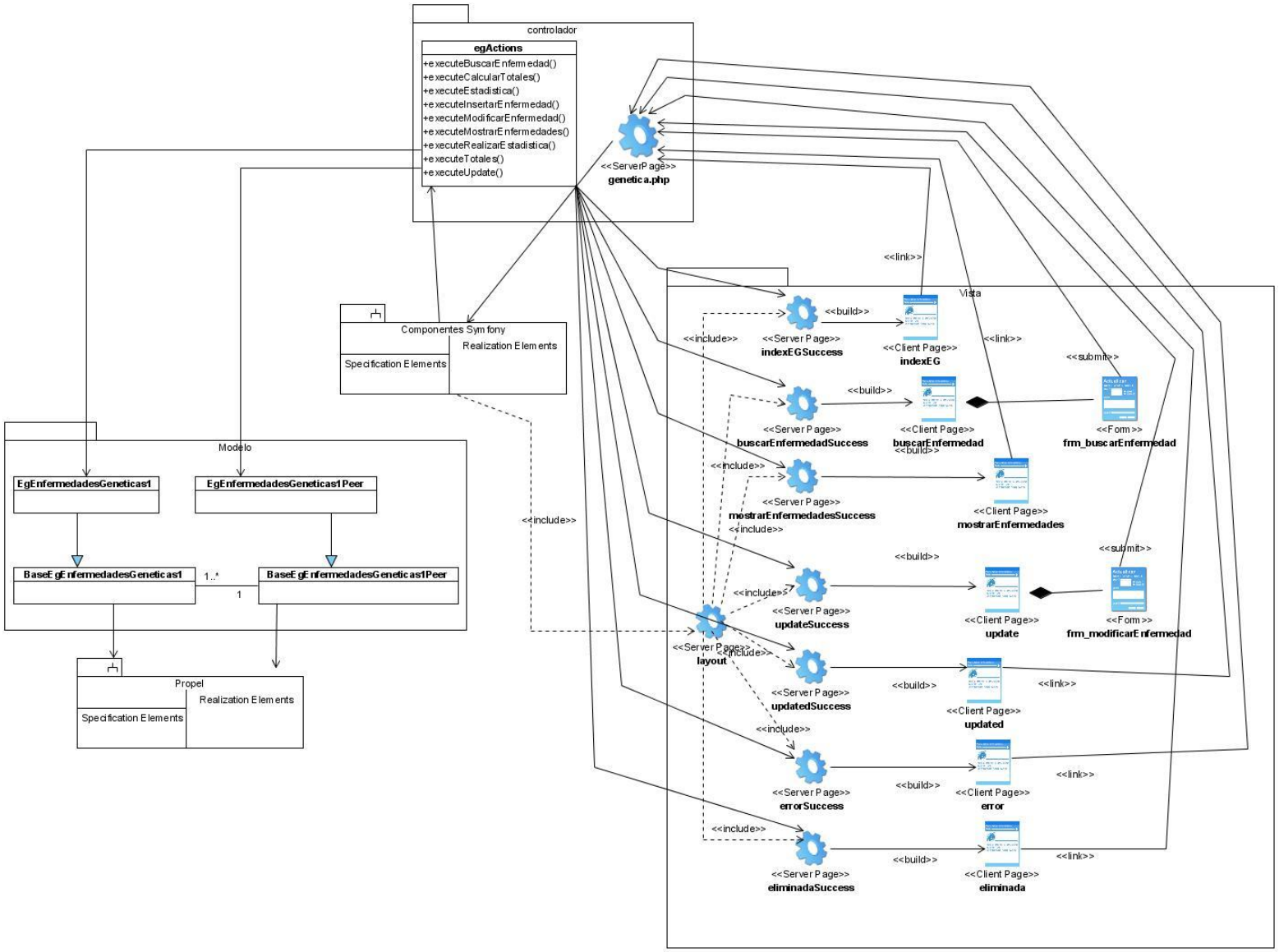

Figura 6. Diagrama de clases del diseño caso de uso: Gestionar enfermedades: Sección Editar enfermedad.

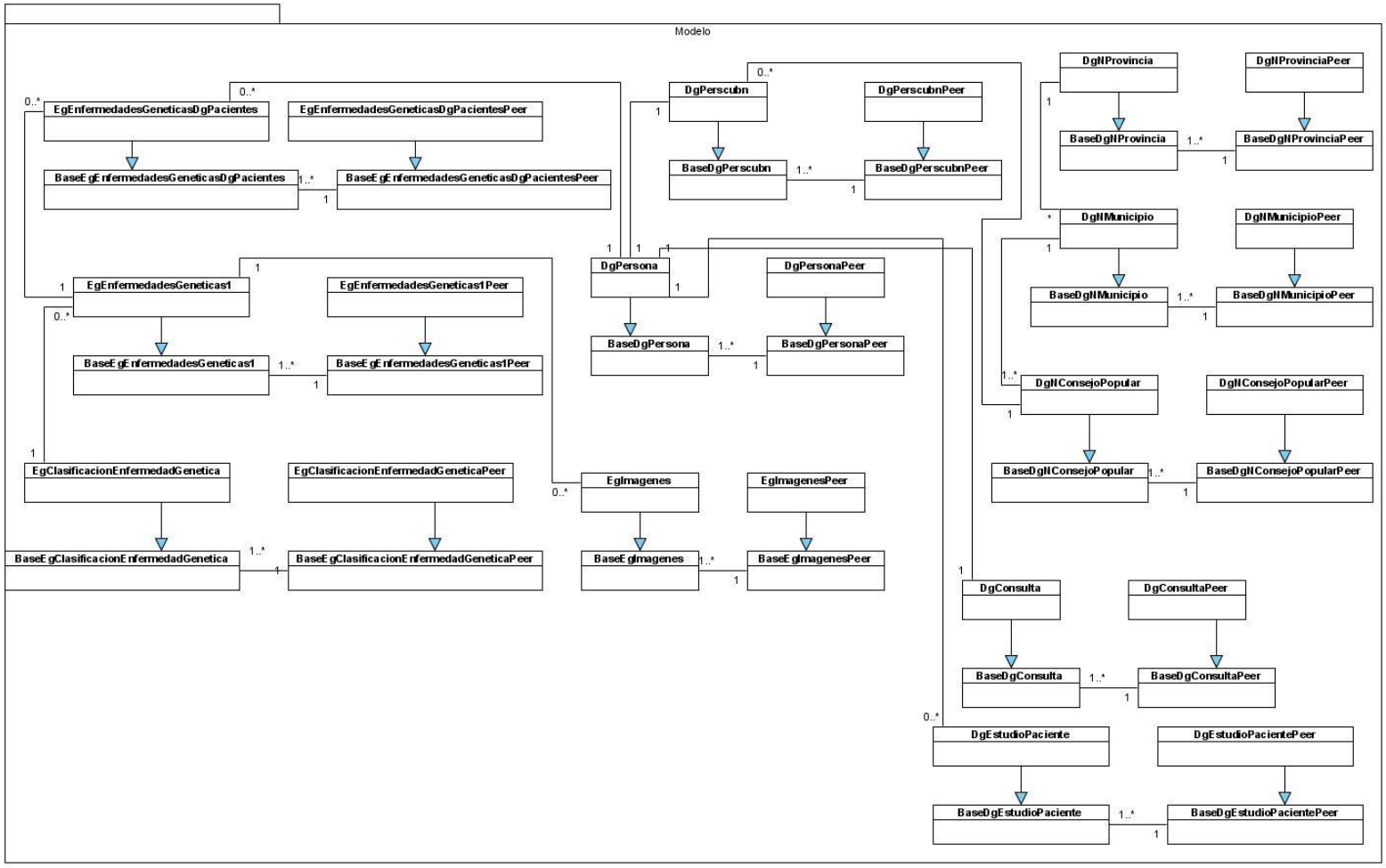

Figura 7. Paquete del modelo de datos.

2.5.3. Diagramas de Interacción (Secuencia).

El diagrama de interacción del sistema es el encargado de presentar de manera gráfica la secuencia de acciones en un caso de uso ya que el objetivo principal que se persigue es encontrar secuencias de interacciones detalladas y ordenadas. Estos se pueden representar por diagramas de Colaboración y/o Diagramas de Secuencia.

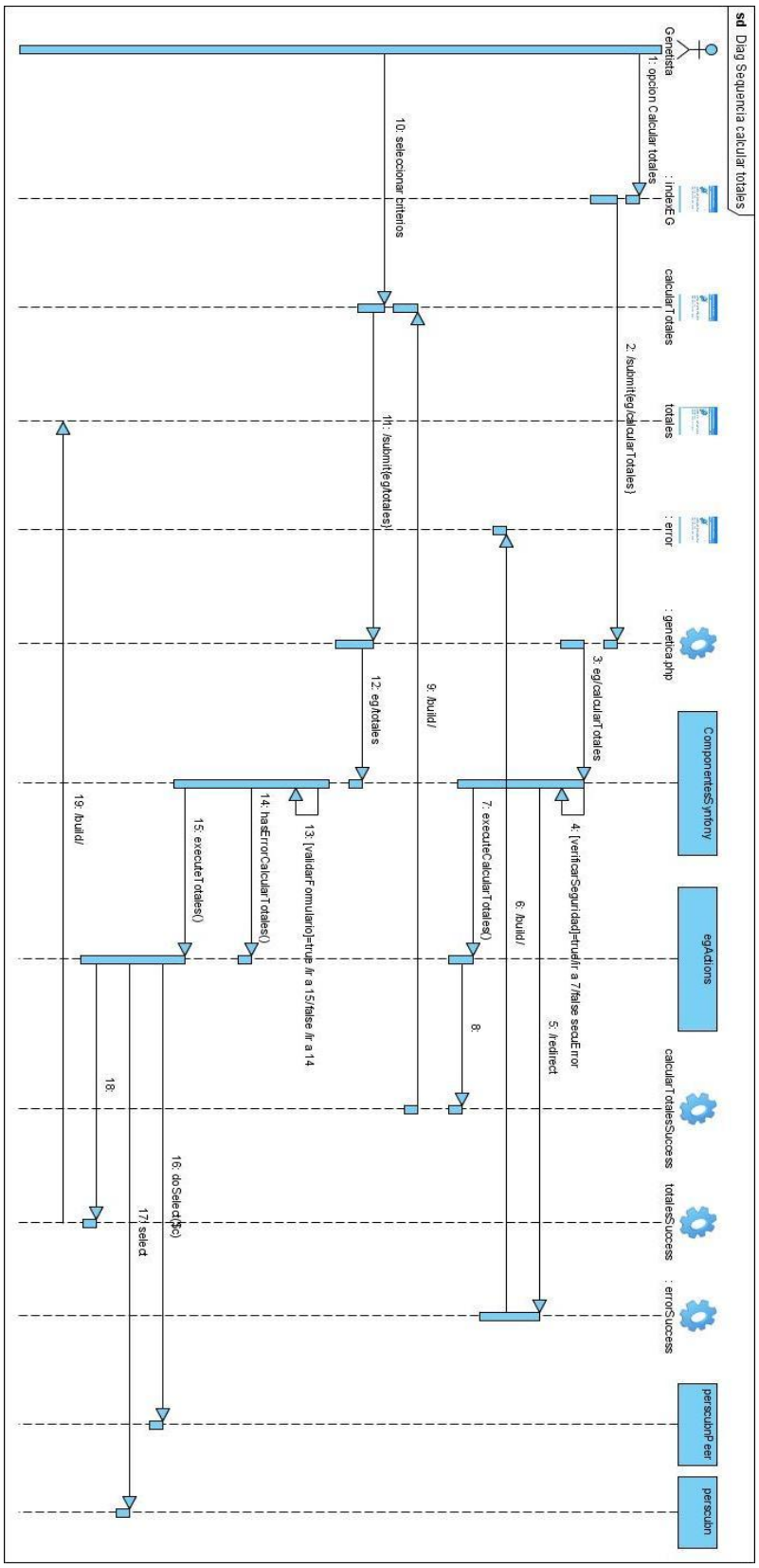

Figura 8. Diagrama de secuencia caso de uso: Gestionar reporte de cálculos totales.

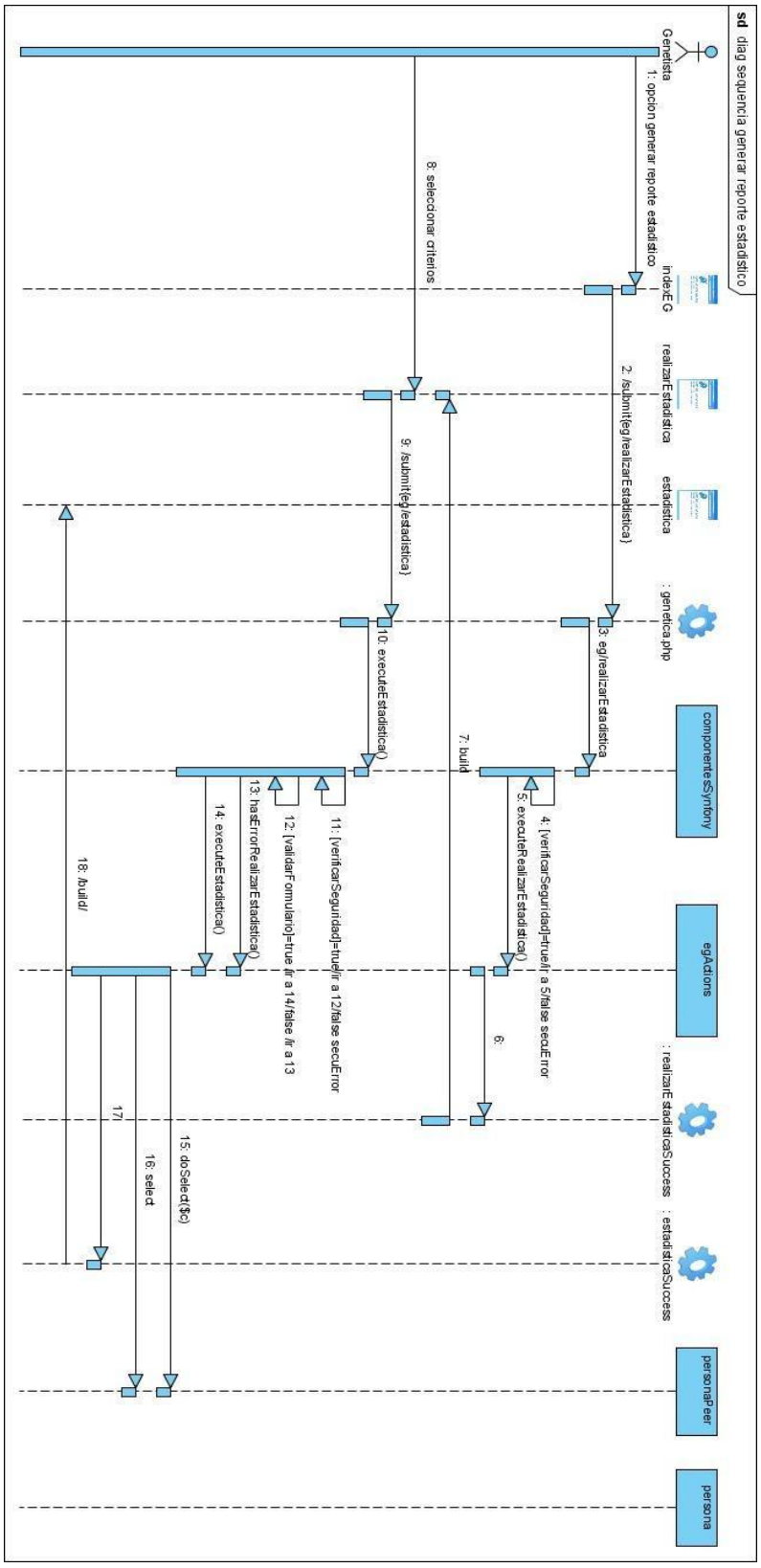

Figura 9. Diagrama de secuencia caso de uso: Gestionar reporte estadístico.

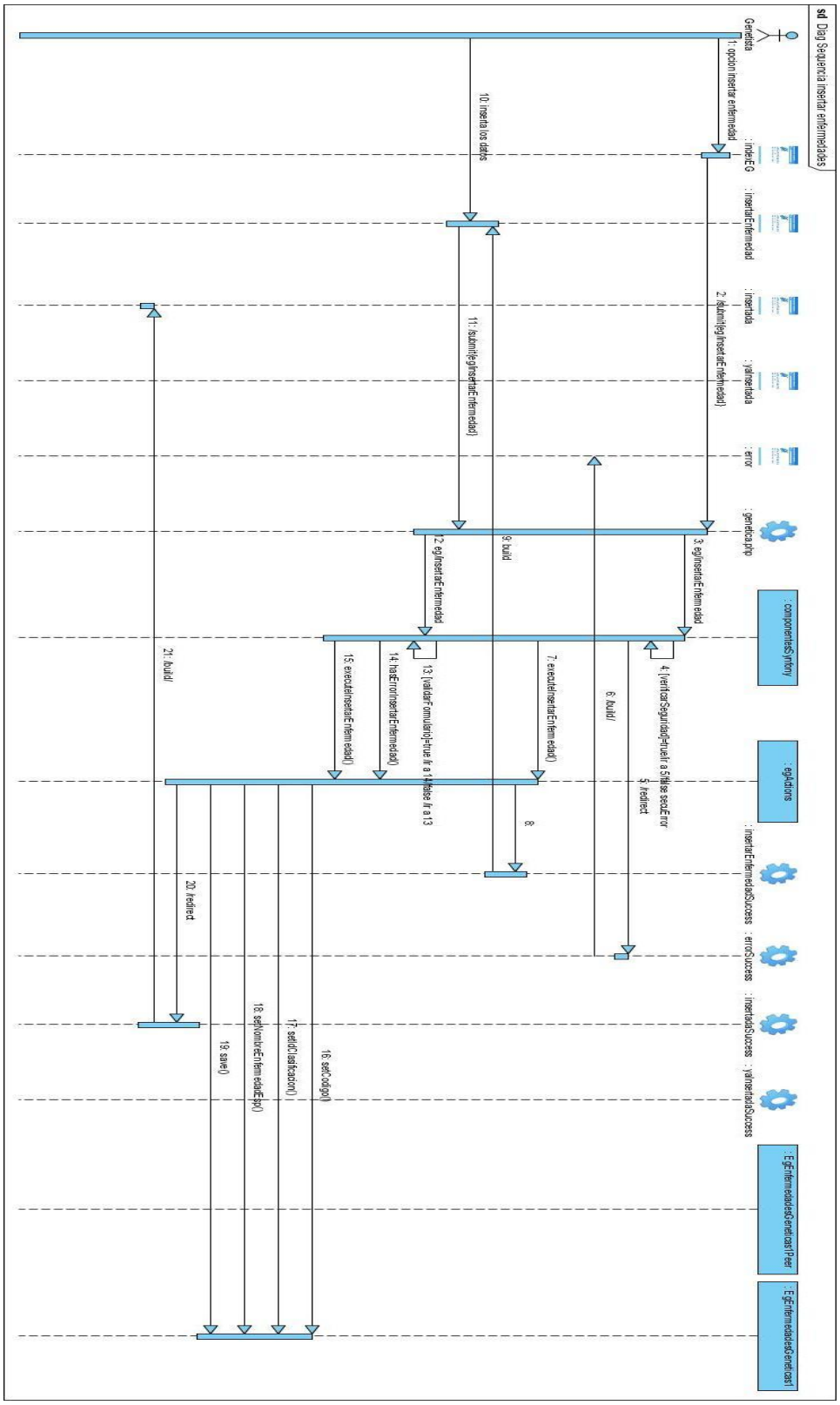

Figura 10. Diagrama de secuencia caso de uso: Gestionar Enfermedades (Insertar Enfermedad).

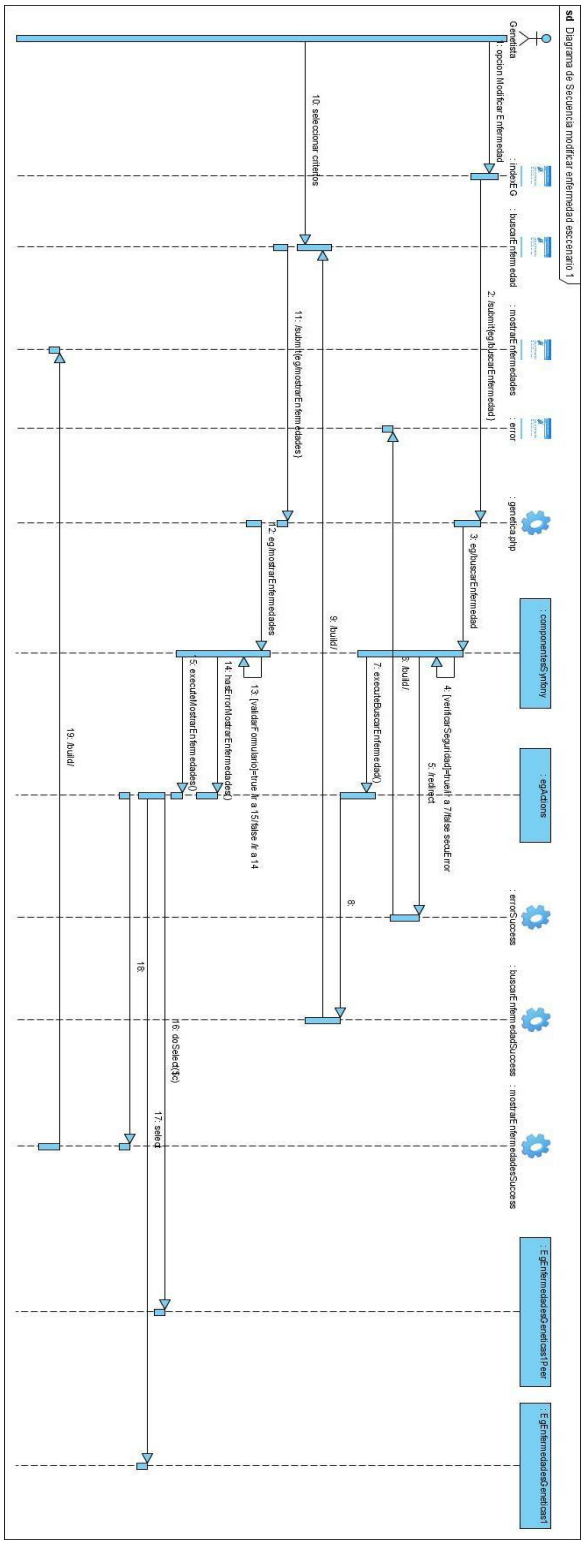

Figura 11. Diagrama de secuencia caso de uso: Gestionar enfermedades (modificar Enfermedad escenario 1).

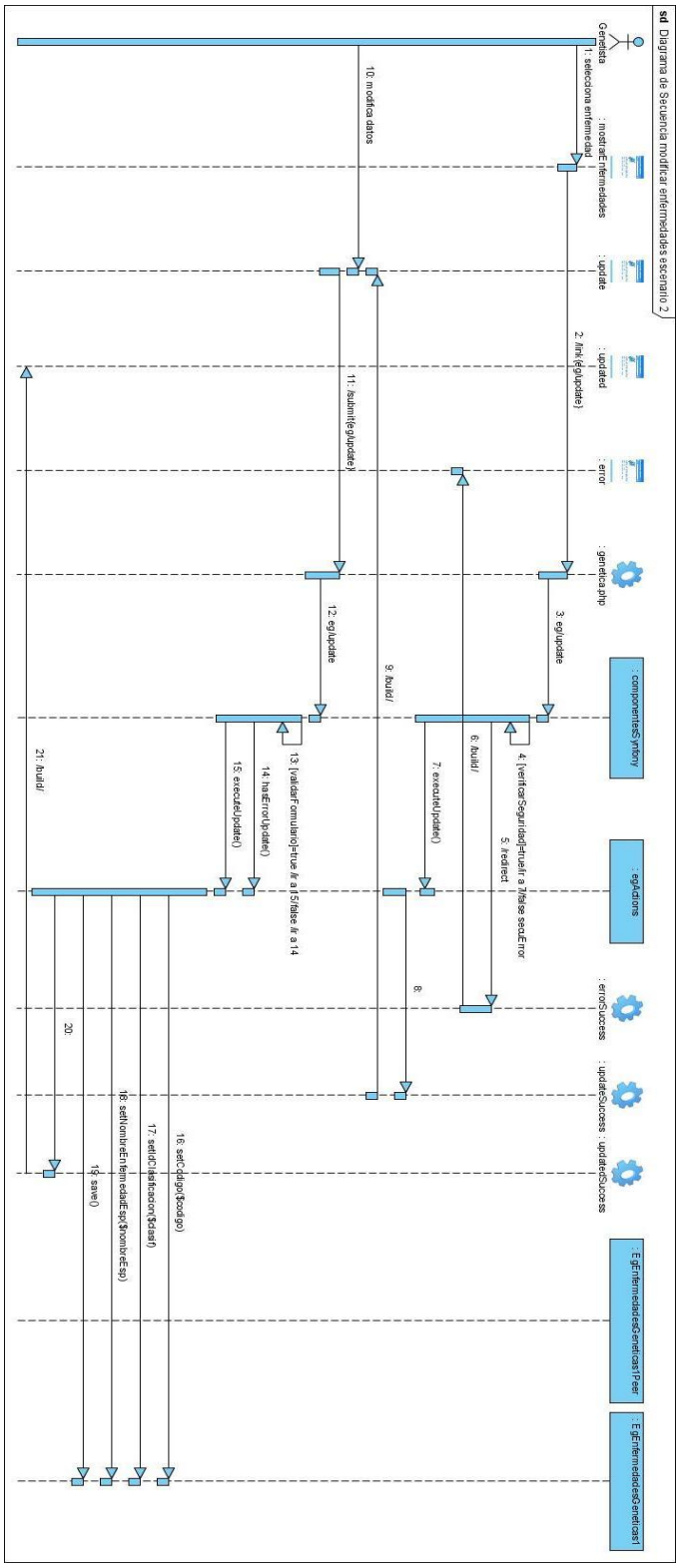

Figura 12. Diagrama de secuencia caso de uso: Gestionar enfermedades (modificar Enfermedad escenario 1).

2.6. Modelo de despliegue.

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de computo. El modelo de despliegue se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño.

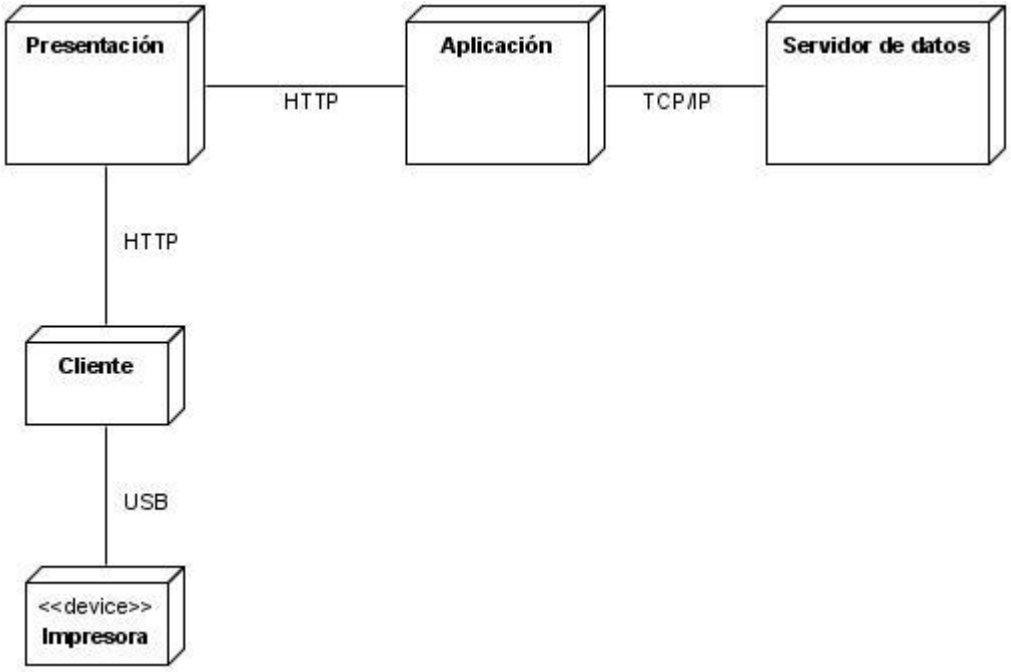

Figura 13. Modelo de despliegue.

2.7. Tratamiento de errores.

El tratamiento de errores es uno de los procesos más importantes en la creación de cualquier sistema informático debido que es la cara del sistema frente a las situaciones adversas que se puedan presentar. Con un buen tratamiento de errores se puede garantizar el correcto funcionamiento del sistema así como la satisfacción de los usuarios del mismo.

Una vez que el usuario introduce los datos, estos serán validados en el lado del servidor mediante archivos YAML (.yml) y en caso de haber algún error, se mostraran mensajes que le indican donde ha ocurrido dicho error, luego, cuando los datos sean correctos, irán a la base de datos del sistema sin peligro alguno de que queden datos incompletos.

2.8. Seguridad

El registro también restringe el acceso a ejecutar las diferentes acciones a los usuarios, estos tienen determinados privilegios y necesitan autenticarse antes de comenzar a usar el sistema informático.
Antes de ser ejecutada, cada acción pasa por un filtro especial que verifica si el usuario actual tiene privilegios de acceder a la acción requerida. En Symfony, los privilegios están compuestos por dos partes:

▪ Las acciones seguras requieren que los usuarios estén autenticados.

▪ Las credenciales son privilegios de seguridad agrupados bajo un nombre y que permiten organizar la seguridad en grupos.

Para restringir el acceso a una acción se crea y se edita un archivo de configuración YAML llamado 'security.yml en el directorio config/ del módulo. En este archivo, se pueden especificar los requerimientos de seguridad que los usuarios deberán satisfacer para cada acción o para todas (all) las acciones. (11)

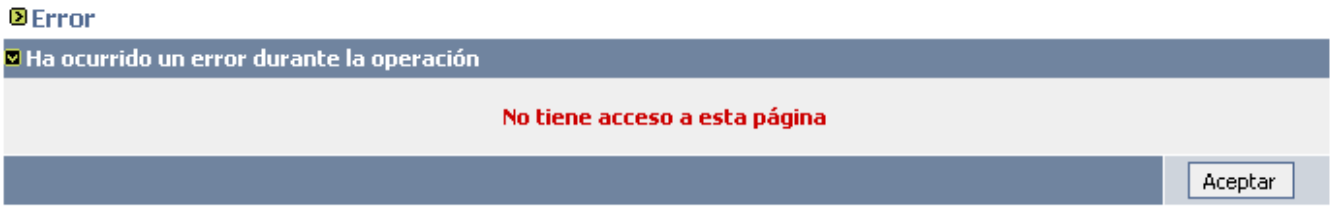

Figura 14. Página de error que se muestra cuando se intenta a acceder sin el nivel requerido a una página.

2.9. Conclusiones

En este capítulo se presentaron los requisitos tanto funcionales como no funcionales del registro, se identificaron los casos de uso y actores del sistema, así como sus descripciones, se representaron los diagramas de clases del diseño y los de interacción, también se habla sobre los patrones de diseño y los de arquitectura que se aplicaron. Por ultimo se presentaron el tratamiento de errores y el tema de la seguridad.

CAPÍTULO 3: IMPLEMENTACIÓN

3.1. Introducción

En este capítulo se presenta el modelo de implementación, contiene los diagramas de componentes del registro y se brinda una descripción de los principales métodos implementados así como se muestran imágenes de la interfaz del registro.

3.2. Modelo de Implementación

El modelo de implementación describe los componentes a construir y su organización y dependencia entre nodos físicos en los que funcionará el sistema. En este se generan los diagramas de despliegue y de componentes. Un componente es una parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces y proporciona la realización de los mismos, típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios, etc.).

3.2.1. Diagrama de Componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación.

El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, especificando:

• Los subsistemas de implementación y sus dependencias a la hora de importar código.

• Organizar los subsistemas de implementación en capas.

También se utilizan para mostrar las dependencias de compilación de los ficheros de código, relaciones de derivación entre ficheros de código fuente y ficheros que son resultados de la compilación, dependencias entre elementos de implementación y los correspondientes elementos de diseños que son implementados.

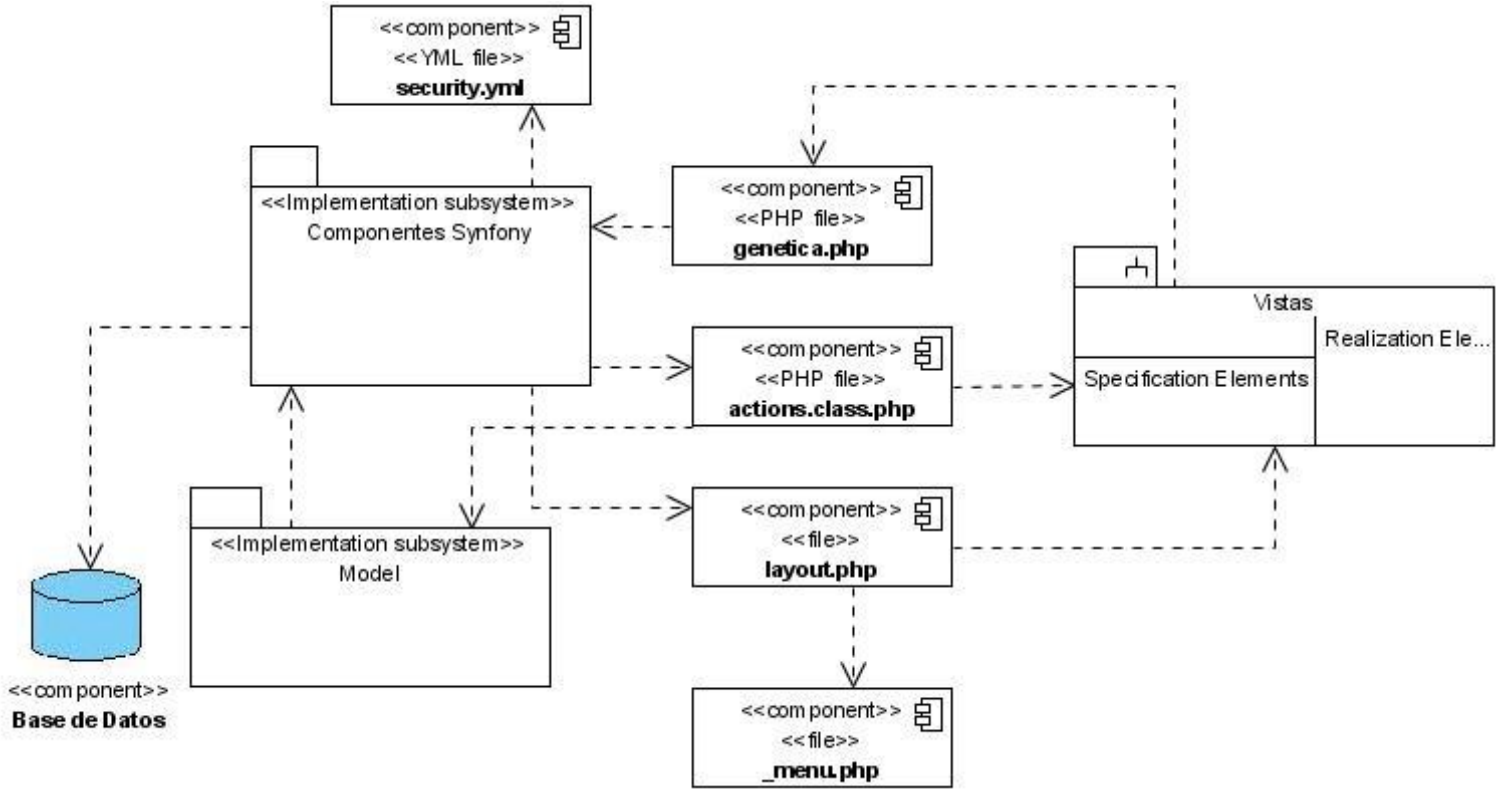

Figura 15. Diagrama de componentes del registro.

3.3. Estilo de código

1. Todas las etiquetas php deben ser completas (<?php ?>)... no reducidas (<? ?>).
2. Todas las variables deberían ser inicializadas o, al menos, comprobada su existencia utilizando isset() antes de ser utilizadas.
3. **El identado** del texto debe ser siempre de 4 espacios. No utilices los tabulador (todos los editores no interpretan el tab de la misma manera).
4. **Los nombres de las variables y funciones** tienen que ser siempre fáciles de leer, procurando que sean palabras en minúsculas con significado claro. Si realmente necesita más de una palabra, póngalas juntas, poniendo la inicial de cada palabra en mayúscula siempre que no sea la primera, pero procure mantenerlas tan breves como sea posible. Utilice nombres en plural para arreglos o matrices de objetos. Ejemplos:
5. **Los bloques de código** siempre deben estar encerrados por llaves (incluso si solo constan de una línea) y correctamente identados (a 4 espacios).

6. **Las cadenas** tienen que ser definidas utilizando comillas simples siempre que sea posible, para obtener un mejor rendimiento.
7. **Todas** las funciones y clases deben estar comentariadas. **Los comentarios** deben ser añadidos de forma que resulten prácticos, para explicar el flujo del código y el propósito de las funciones o variables.

En los comentarios de las funciones debe aparecer el autor de la función, el objetivo de la misma, y una descripción de cada uno de los parámetros que se le pasen. (13)

3.4. Código fuente de los principales métodos y su descripción.

```
 //accion ke hace la consulta para mostrar los resultados de la estadistica...
public function executeEstadistica()
    {
    $c = new Criteria();
    //para el sexo...
    $this->sexo = $this->getRequestParameter('sexo');
    if($this->sexo == '1' || $this->sexo == '2')
    {
        $c->add(DgPersonaPeer::ID_GENERO, "$this->sexo");
    }
    $this->region = $this->getRequestParameter('region');

    //para si es por provincia...
    if($this->region == 3)
    {
        $this->idProvincia = $this->getRequestParameter('provincia');
        $c->add(DgNprovinciaPeer::ID_PROVINCIA, "$this->idProvincia");
        $c->addJoin(DgNprovinciaPeer::ID_PROVINCIA,DgNmunicipioPeer::ID_PROVINCIA);
        $c->addJoin(DgNmunicipioPeer::ID_MUNICIPIO, DgNconsejoPopularPeer::ID_MUNICIPIO);
        $c->addJoin(DgNconsejoPopularPeer::ID_CONSEJO, DgPersCubnPeer::ID_CONSEJO);
        $c->addJoin(DgPersonaPeer::ID_PERSONA, DgPersCubnPeer::ID_PERSONA);
    }

    //para si es por municipio...
    elseif($this->region == 4)
    {
        $this->idMunicipio = $this->getRequestParameter('municipio');
        $c->add(DgNmunicipioPeer::ID_MUNICIPIO, "$this->idMunicipio");
        $c->addJoin(DgNmunicipioPeer::ID_MUNICIPIO, DgNconsejoPopularPeer::ID_MUNICIPIO);
        $c->addJoin(DgNconsejoPopularPeer::ID_CONSEJO, DgPersCubnPeer::ID_CONSEJO);
        $c->addJoin(DgPersonaPeer::ID_PERSONA, DgPersCubnPeer::ID_PERSONA);
```

```
}

//para el rango de edades...
$this->edad1 = $this->getRequestParameter('edad1');
$this->edad2 = $this->getRequestParameter('edad2');
if($this->edad1 != 'edad' && $this->edad2 != 'edad')
{
    $c->add(DgPersonaPeer::EDAD, $this->edad2, Criteria::LESS_THAN);
    $c->add(DgPersonaPeer::EDAD, $this->edad1, Criteria::GREATER_THAN);
}

//para enfermedad...
$this->idEnfermedad = $this->getRequestParameter('enfermedad');
$c->add(DgPersonaEgEnfermedadesGeneticasPeer::CODIGO, "$this->idEnfermedad");
$c->addJoin(DgPersonaEgEnfermedadesGeneticasPeer::ID_PERSONA,DgPersonaPeer::ID_PERSONA);

//para el tiempo...
$this->tiempo = $this->getRequestParameter('tiempo');
if($this->tiempo == 1)
{
    $this->anno=$this->getRequestParameter('anno');
    $iAnno = new DateTime();
    $fAnno = new DateTime();
    $fAnno->setDate("$this->anno-12-31");
    $iAnno->setDate("$this->anno-1-1");
    $c->add(DgConsultaPeer::FECHA_CONSULTA,$fAnno,CRITERIA::LESS_EQUAL);
    $c->add(DgConsultaPeer::FECHA_CONSULTA,$iAnno,CRITERIA::GREATER_EQUAL);
    $c->addJoin(DgConsultaPeer::ID_PERSONA,DgPersonaPeer::ID_PERSONA);
}
if($this->tiempo == 2)
{
    $this->annoI = $this->getRequestParameter('annoInicial');
    $this->annoF = $this->getRequestParameter('annoFin');
    $iAnno = new DateTime();
    $fAnno = new DateTime();
    $fAnno->setDate($this->annoF-12-31);
    $iAnno->setDate($this->annoI-1-1);
    $c->add(DgConsultaPeer::FECHA_CONSULTA,$iAnno,Criteria::GREATER_EQUAL);
    $c->add(DgConsultaPeer::FECHA_CONSULTA,$fAnno,Criteria::LESS_EQUAL);
    $c->addJoin(DgConsultaPeer::ID_PERSONA,DgPersonaPeer::ID_PERSONA);
}
    $pager = new sfPropelPager('DgPersona', sfConfig::get('pagnum'));
    $pager->setCriteria($c);
    $pager->setPage($this->getRequestParameter('page', 1));
    $pager->init();
    $this->pager = $pager;
    $this->personas = DgPersonaPeer::doSelect($c);
```

Figura 16. Código del método: estadística.

Este método permite hacer una estadística en cuanto a enfermedad a nivel nacional, provincial, o municipal, se toman los datos que vienen del formulario de realizarEstadistica y se hace la consulta

para buscar las personas que cumplen con esas características y se muestra una tabla con los pacientes encontrados y el resultado de la estadística si se han escogido bien los criterios, si no se escogieron bien los criterios se mostrará un mensaje avisando qué criterio falta o está mal puesto.

```
//hace la consulta para buscar la enfermedad
public function executeMostrarEnfermedades()
{
    $this->nombreEnfermedadEsp = $this->getRequestParameter('nombreEnfermedadEsp');
    $this->nombreEnfermedadIng = $this->getRequestParameter('nombreEnfermedadIng');
    $this->codigoEnfermedad = $this->getRequestParameter('codigoEnfermedad');
    $this->clasif = $this->getRequestParameter('clasificacion');
    $this->clasificaciones = EgClasificacionEnfermedadGeneticaPeer::doSelect(new Criteria());
    $c  = new Criteria();
    if($this->nombreEnfermedadEsp != NULL)
    {
        $c->add(EgEnfermedadesGeneticas1Peer::NOMBRE_ENFERMEDAD_ESP, "%$this->nombreEnfermedadEsp%", Cr
    }
    if($this->nombreEnfermedadIng != NULL)
    {
        $c->add(EgEnfermedadesGeneticas1Peer::NOMBRE_ENFERMEDAD_ENG, "%$this->nombreEnfermedadIng%", Cr
    }
    if($this->codigoEnfermedad != NULL)
    {
        $c->add(EgEnfermedadesGeneticas1Peer::CODIGO, "$this->codigoEnfermedad%", Criteria::LIKE);
    }
    if($this->clasif != " ")
    {
        //$c->add(EgEnfermedadesGeneticas1Peer::ID_CLASIFICACION, "$this->clasif");
    }
    $pager = new sfPropelPager('EgEnfermedadesGeneticas1', sfConfig::get('pagnum'));
    $pager->setCriteria($c);
    $pager->setPage($this->getRequestParameter('page', 1));
    $pager->init();
    $this->pager = $pager;
    $this->enfermedades = EgEnfermedadesGeneticas1Peer::doSelect($c);
```

Figura 17. Código del método: mostrarEnfermedades.

Este método es el encargado de hacer la consulta de acuerdo a los datos que vienen del formulario de buscarEnfermedad y buscar las enfermedades que cumplen con esas características y si se ha escogido al menos un criterio, se muestra una tabla con las encontradas y da la posibilidad de modificarlas o eliminarlas, si no, se mostrará un mensaje avisándolo.

```
//insertar una enfermedad...
public function executeInsertarEnfermedad()
{
    $codigo = $this->getRequestParameter('codigoEnfermedad');
    $crit = new Criteria();
    $crit->add(EgEnfermedadesGeneticas1Peer::CODIGO, "$codigo");
    $dime = EgEnfermedadesGeneticas1Peer::doCount($crit);
    if($this->getRequestParameter('but')=='cancelar')
    {
        return sfView::SUCCESS;
    }
    if($dime > 0)
    {
        $this->redirect('eg/yaInsertada');
    }
    else
    {
        $enf = new EgEnfermedadesGeneticas1();
        $clasificacion = $this->getRequestParameter('clasificacion');
        $nombreEsp = $this->getRequestParameter('nombreEnfermedadEsp');
        $nombreIng = $this->getRequestParameter('nombreEnfermedadIng');
        $desc = $this->getRequestParameter('descripcion');
        if($desc != NULL)
        {
            $enf->setDescripcion($desc);
        }
        $enf->setCodigo($codigo);
        $enf->setNombreEnfermedadEsp($nombreEsp);
        $enf->setNombreEnfermedadEng($nombreIng);
        $enf->setIdClasificacion($clasificacion);
        $enf->save();
```

Figura 18. Código del método: insertarEnfermedad.

En este método se inserta una enfermedad, se toman los datos del formulario y se busca el código en la base de datos, si no se encuentra y los datos están correctos se procede a insertar la enfermedad en la base de datos y se redirecciona a una pagina que muestra: “La enfermedad ha sido insertada correctamente”; si ya se encuentra, entonces se redirecciona a la página que muestra: “La enfermedad ya ha sido insertada antes”.

```
//modificar una enfermedad...
public function executeUpdate()
{
    //para cargar los datos de la enfermedad
    $enf = EgEnfermedadesGeneticas1Peer::retrieveByPK($this->getRequestParameter('id'));
    $this->nombreEsp = $enf->getNombreEnfermedadEsp();
    $this->nombreIng = $enf->getNombreEnfermedadEng();
    $this->codigo = $enf->getCodigo();
    $this->clasif = $enf->getIdClasificacion();
    $this->desc = $enf->getDescripcion();

    //modificar
    $nClasif = $this->getRequestParameter('clasificacion');
    $nCodigo = $this->getRequestParameter('codigoEnfermedad');
    $nNombreEsp = $this->getRequestParameter('nombreEnfermedadEsp');
    $nNombreIng = $this->getRequestParameter('nombreEnfermedadIng');
    $nDesc = $this->getRequestParameter('descripcion');
    $enf->setCodigo($nCodigo);
    $enf->setIdClasificacion($nClasif);
    $enf->setNombreEnfermedadEsp($nNombreEsp);
    $enf->setNombreEnfermedadEng($nNombreIng);
    if($nDesc != NULL)
    {
        $enf->setDescripcion($nDesc);
    }
    $enf->Save();

    $this->redirect('eg/updated');
}
```

Figura 19. Código del método: update.

En este método se modifican los datos de una enfermedad, toma el id de la enfermedad, busca en la base de datos, y muestra la información de esta en el formulario, luego se toman los datos del formulario nuevamente y si están correctos se procede a modificar la enfermedad en la base de datos, entonces se redirecciona a la página que muestra: “Los datos de la enfermedad han sido modificados satisfactoriamente”.

```
public function executeTotales()
{
    $c = new Criteria();
    //para el sexo...
    $this->sexo = $this->getRequestParameter('sexo');
    if($this->sexo == '1' || $this->sexo == '2')
    {
        $c->add(DgPersonaPeer::ID_GENERO, "$this->sexo");
    }
    $this->region = $this->getRequestParameter('region');

    //para si es nacional...
    if($this->region == 2)
    {
        $pr = new Criteria();
        $this->provs= DgNprovinciaPeer::doselect($pr);
    }

    //para si es por provincia...
    if($this->region == 3)
    {
        $this->idProvincia = $this->getRequestParameter('provincia');
        $mu = new Criteria();
        $this->provincia = DgNprovinciaPeer::retrieveByPK($this->idProvincia);
        $mu->add(DgNmunicipioPeer::ID_PROVINCIA,"$this->idProvincia");
        $this->muns = DgNmunicipioPeer::doselect($mu);
        $c->add(DgNprovinciaPeer::ID_PROVINCIA, "$this->idProvincia");
        $c->addJoin(DgNprovinciaPeer::ID_PROVINCIA,DgNmunicipioPeer::ID_PROVINCIA);
        $c->addJoin(DgNmunicipioPeer::ID_MUNICIPIO, DgNconsejoPopularPeer::ID_MUNICIPIO);
        $c->addJoin(DgNconsejoPopularPeer::ID_CONSEJO, DgPersCubnPeer::ID_CONSEJO);
    }
```

```
//para si es por municipio...
elseif($this->region == 4)
{
    $con = new Criteria();
    $this->idMunicipio = $this->getRequestParameter('municipio');
    $this->idProvincia = $this->getRequestParameter('provincia');
    $this->provincia = DgNprovinciaPeer::retrieveByPK($this->idProvincia);
    $this->municipio = DgNmunicipioPeer::retrieveByPK($this->idMunicipio);
    $con->add(DgNconsejoPopularPeer::ID_MUNICIPIO,"$this->idMunicipio");
    $this->consejos = DgNconsejoPopularPeer::doselect($con);
    $c->add(DgNmunicipioPeer::ID_MUNICIPIO, "$this->idMunicipio");
    $c->addJoin(DgNmunicipioPeer::ID_MUNICIPIO, DgNconsejoPopularPeer::ID_MUNICIPIO);
    $c->addJoin(DgNconsejoPopularPeer::ID_CONSEJO, DgPersCubnPeer::ID_CONSEJO);
}
//para el rango de edades...
$this->edad1 = $this->getRequestParameter('edad1');
$this->edad2 = $this->getRequestParameter('edad2');
if($this->edad1 != 'edad' && $this->edad2 != 'edad')
{
    $c->add(DgPersonaPeer::EDAD, "$this->edad2", Criteria::LESS_THAN);
    $c->add(DgPersonaPeer::EDAD, "$this->edad1", Criteria::GREATER_THAN);
}

//para enfermedad...
$this->idEnfermedad = $this->getRequestParameter('enfermedad');
$c->add(DgPersonaEgEnfermedadesGeneticasPeer::CODIGO, "$this->idEnfermedad");
$c->addJoin(DgPersonaEgEnfermedadesGeneticasPeer::ID_PERSONA,DgPersonaPeer::ID_PERSONA);

//para el tiempo...
$this->tiempo = $this->getRequestParameter('tiempo');
if($this->tiempo == "meses")
```

```
    {
        $this->anno=$this->getRequestParameter('anno');
        $iAnno = new DateTime();
        $fAnno = new DateTime();
        $fAnno->setDate($this->anno-12-31);
        $iAnno->setDate($this->anno-1-1);
        $c->add(DgConsultaPeer::FECHA_CONSULTA,$fAnno,CRITERIA::LESS_EQUAL);
        $c->add(DgConsultaPeer::FECHA_CONSULTA,$iAnno,CRITERIA::GREATER_EQUAL);
        $c->addJoin(DgConsultaPeer::ID_PERSONA,DgPersonaPeer::ID_PERSONA);
    }
    if($this->tiempo == "annos")
    {
        $this->annoI = $this->getRequestParameter('annoInicial');
        $this->annoF = $this->getRequestParameter('annoFin');
        $iAnno = new DateTime();
        $fAnno = new DateTime();
        $fAnno->setDate($this->annoF-12-31);
        $iAnno->setDate($this->annoI-1-1);
        $c->add(DgConsultaPeer::FECHA_CONSULTA,$iAnno,Criteria::GREATER_EQUAL);
        $c->add(DgConsultaPeer::FECHA_CONSULTA,$fAnno,Criteria::LESS_EQUAL);
        $c->addJoin(DgConsultaPeer::ID_PERSONA,DgPersonaPeer::ID_PERSONA);
    }
    $c->addJoin(DgPersonaPeer::ID_PERSONA,DgPersCubnPeer::ID_PERSONA);
    $this->totales = DgPersCubnPeer::doSelect($c);
}
```

Figura 20. Código del método: totales.

Este método permite hacer calcular totales en cuanto a enfermedad a nivel nacional, provincial, o municipal, se toman los datos que vienen del formulario de calcularTotales y se hace la consulta para buscar la cantidad de personas que cumplen con esas características y se muestra una tabla con los totales encontrados si se han escogido bien los criterios, si no se escogieron bien los criterios se mostrará un mensaje avisando qué criterio falta o está mal puesto.

3.5. Interfaz

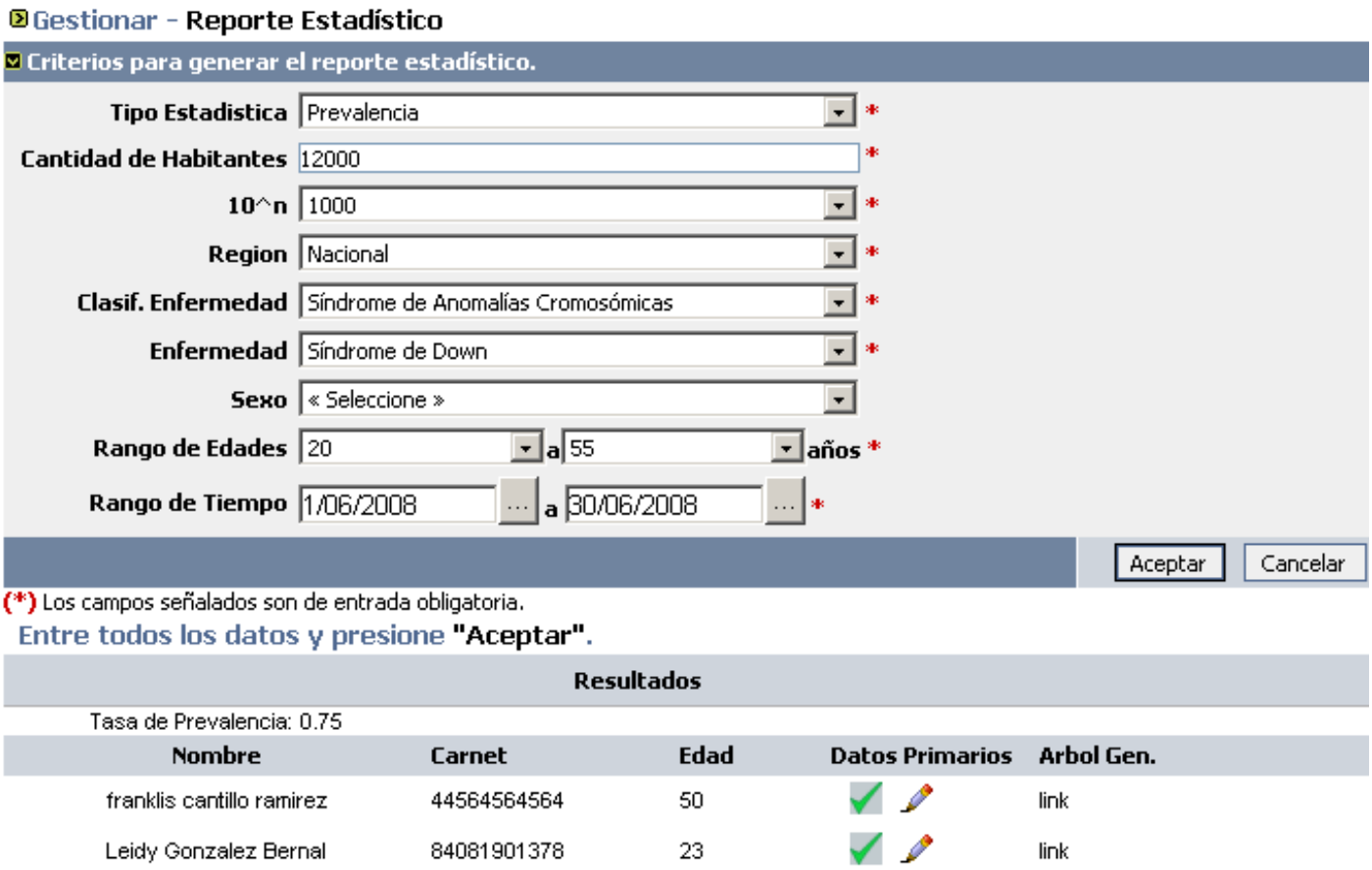

Figura 21. Interfaz Gestionar Reporte Estadístico.

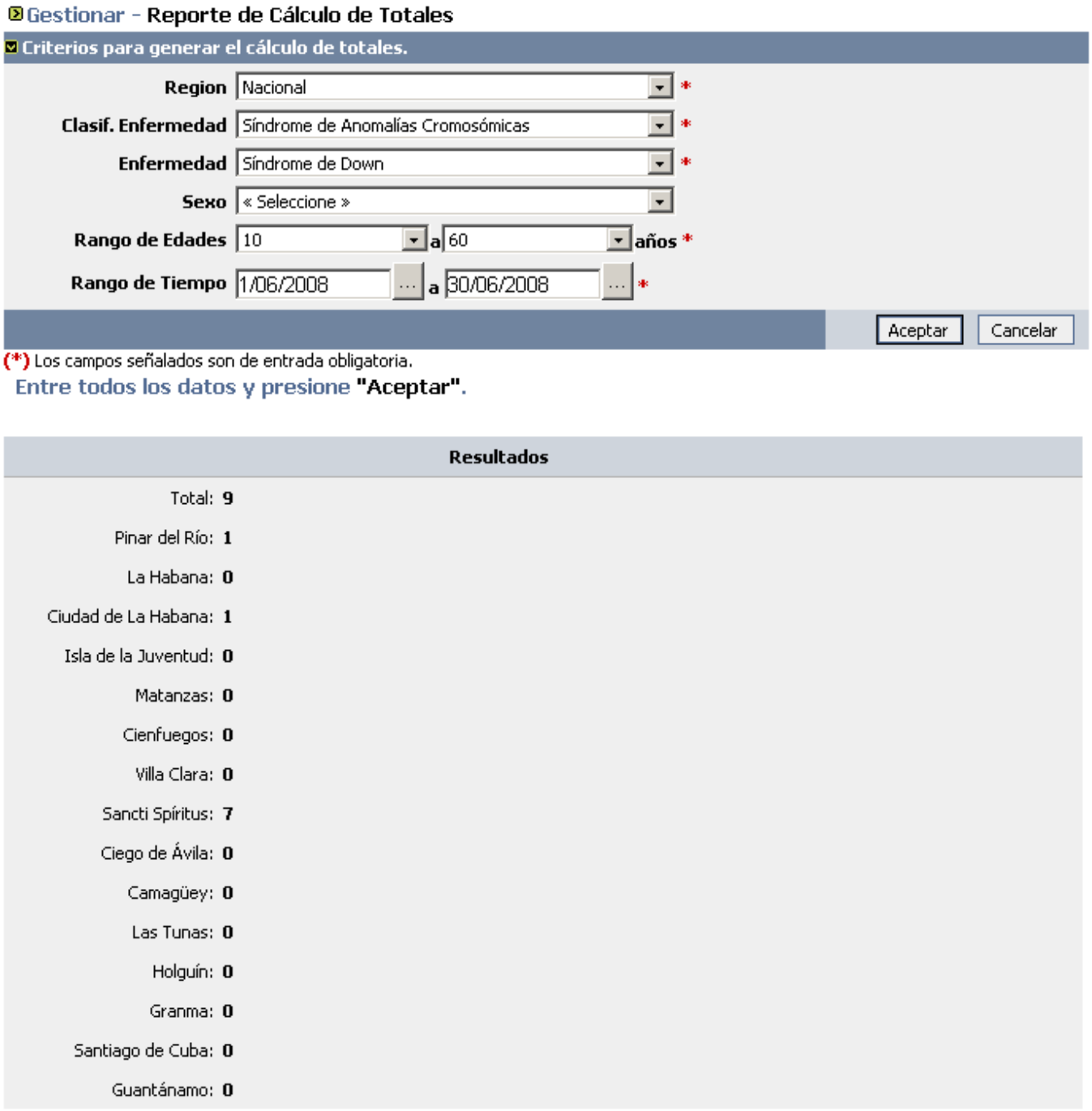

Figura 22. Interfaz Gestionar Reporte de cálculos totales.

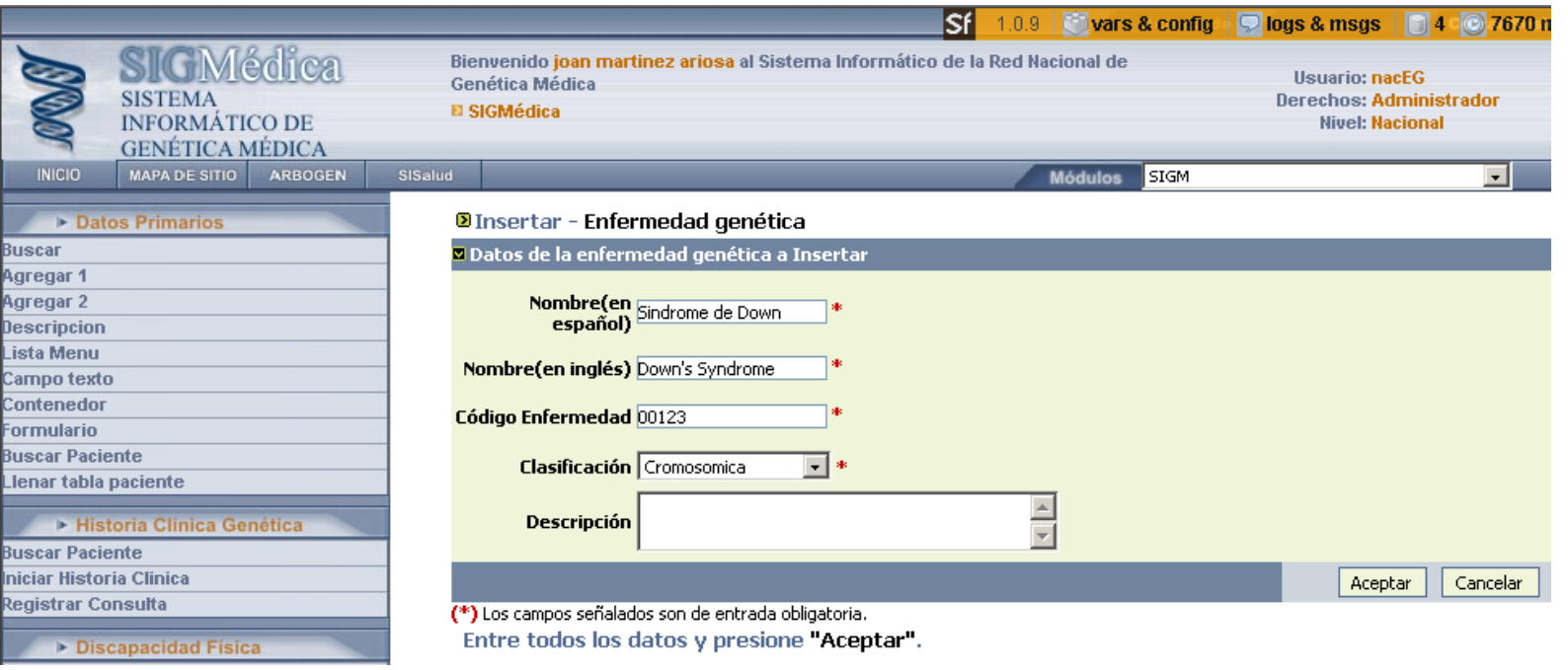

Figura 23. Interfaz Insertar enfermedad.

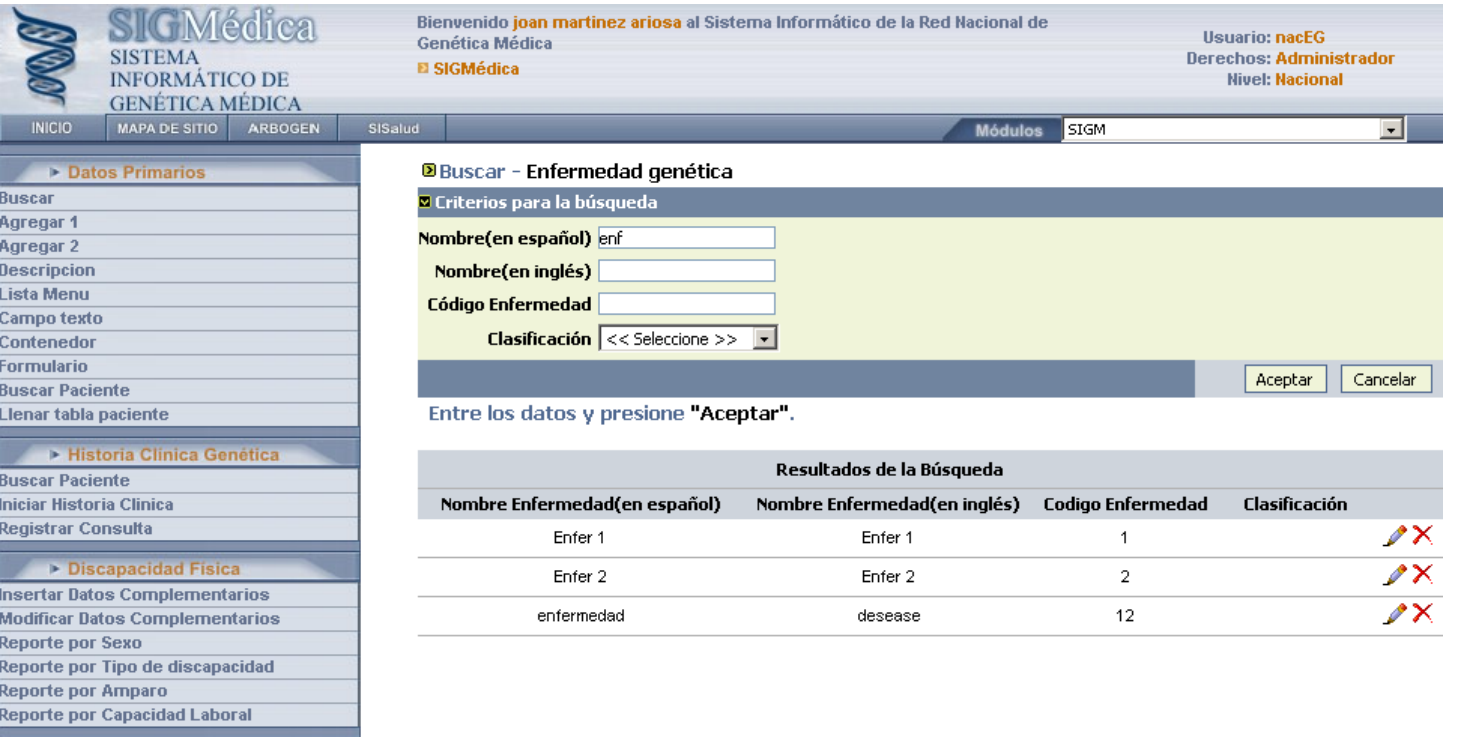

Figura 24. Interfaz Buscar enfermedad.

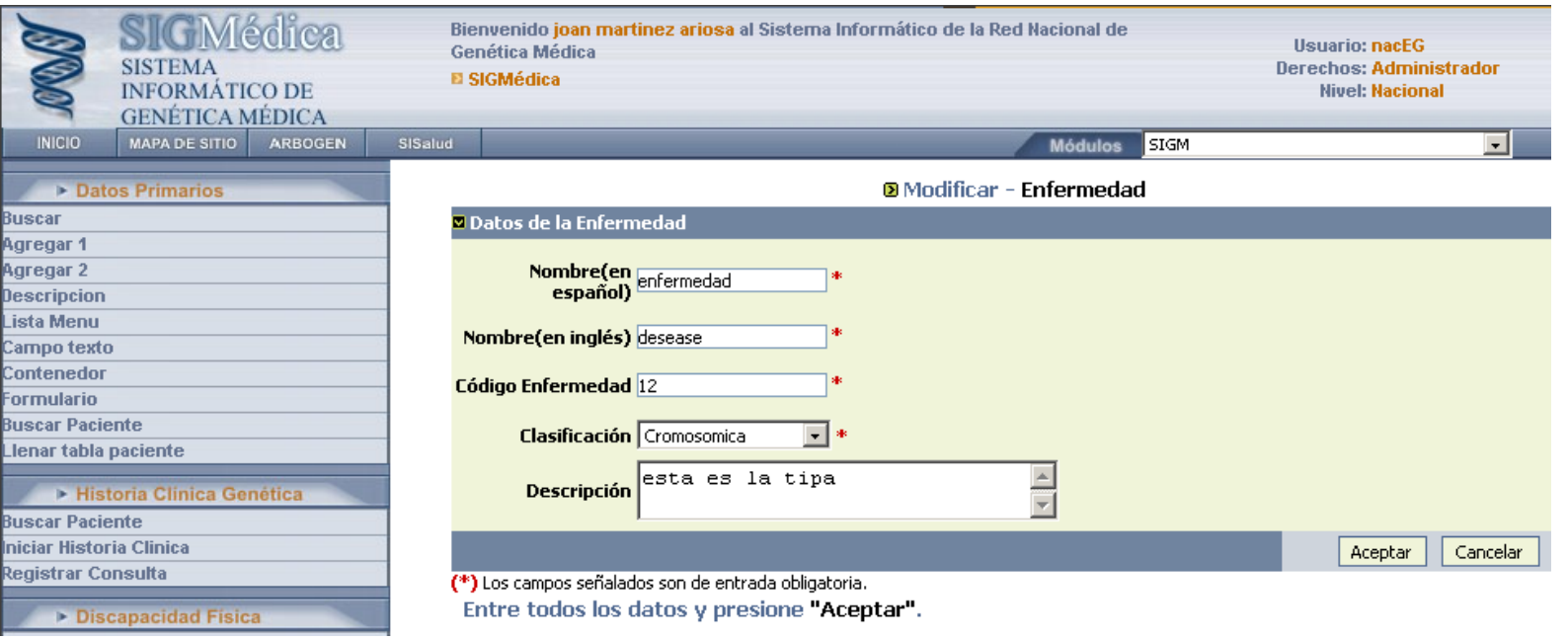

Figura 25. Interfaz Modificar enfermedad.

3.6. Conclusiones.

En este capítulo se obtuvieron como resultado los diagramas de componentes de los casos de uso, se presentaron fragmentos del código fuente con una breve descripción de los métodos implementados y se definió la interfaz del registro, mostrándose imágenes de esta.

CONCLUSIONES

A partir del estudio de la versión anterior de RECUEGEN se identificaron aspectos positivos y deficiencias, que sirvieron de guía para esta investigación.

Se utilizaron los patrones de diseño mejorando así el diseño de las clases y la implementación del registro.

El producto funcional obtenido a partir de la implementación de las clases del diseño permitió integrar el RECUEGEN al SIGM, dando solución al problema científico.

El Registro Cubano de Enfermedades Genéticas versión 2.0 constituye un importante aporte al desarrollo de la genética en Cuba ya que le permite a los genetistas gestionar las enfermedades, y obtener valores estadísticos y totales desde cualquier lugar del país.

RECOMENDACIONES

Al término de esta investigación se recomienda:

Continuar profundizando el estudio de las enfermedades genéticas y de tipos de estadísticas para agregar funcionalidades y características al registro.

Continuar implementando próximas versiones del Registro Cubano de Enfermedades Genéticas con el CNGM y la UCI, para satisfacer todas las necesidades existentes hoy en el Centro.

REFERENCIAS BIBLIOGRÁFICAS

1. Quintana Trujillo, Yudel Juan y Espinosa Núñez, Joel. *Sistema Informatizado para la gestión de las Enfermedades Genéticas*. Universidad de Ciencias Informáticas. Ciudad Habana, 2007.

2. Perodin Sánchez, Yusdenis. *Línea base de la arquitectura del Sistema de Información de Genética Médica.* Facultad de Bioinformática, Universidad de Ciencias Informáticas. Ciudad Habana, 2008.

3. *Guía de Subversion: ¿Qué es? ¿Qué hace?* 2005 http://www.osmosislatina.com/subversion/basico.htm. [En línea]

4. Hinostroza, R.R. *Características de PHP*. 2006.

5. *Las principales características de MySQL.* 2007. http://dev.mysql.com/doc/refman/5.0/es/features.html. [En línea]

6. Larman, Craig. *Uml y patrones, introducción al análisis y diseño orientado a objetos*. México: Prentice Hall, 1999.

7. *Quanta Plus*. [Citado el: 9 de diciembre de 2007.] http://quanta.kdewebdev.org/ [En línea]

8. Sánchez Perodín, Yusdenis. *Pautas de diseño del Sistema de Información de Genética Médica.* Facultad de Bioinformática, Universidad de Ciencias Informáticas. Ciudad Habana, 2008.

9. Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia. *Rational Unified Process (RUP).* https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20a%20RUP.doc [En línea]

10. Lagos Torres, Manuel. *Introducción al diseño con patrones.* 2002 http://www.elrincondelprogramador.com/default.asp?id=29&pag=articulos/leer.asp [En línea]

11. Potencier, Fabien y Zaninotto, François. *Symfony, la guía definitiva*. 2008.

12. *Centro Nacional de Genética Médica.* http://www.sld.cu/sitios/genetica/temas.php?idv=7561 [En línea]

13. Sánchez Perodín, Yusdenis. *Estilo de código del Sistema de Información de Genética Médica.* Facultad de Bioinformática, Universidad de Ciencias Informáticas. Ciudad Habana, 2008.

BIBLIOGRAFÍA

1. Quintana Trujillo, Yudel Juan y Espinosa Núñez, Joel. *Sistema Informatizado para la gestión de las Enfermedades Genéticas*. Universidad de Ciencias Informáticas. Ciudad Habana, 2007.

2. Perodin Sánchez, Yusdenis. *Línea base de la arquitectura del Sistema de Información de Genética Médica.* Facultad de Bioinformática, Universidad de Ciencias Informáticas. Ciudad Habana, 2008.

3. *Guía de Subversion: ¿Qué es? ¿Qué hace?* 2005 http://www.osmosislatina.com/subversion/basico.htm. [En línea]

4. Hinostroza, R.R. *Características de PHP*. 2006.

5. *Las principales características de MySQL.* 2007. http://dev.mysql.com/doc/refman/5.0/es/features.html. [En línea]

6. Larman, Craig. *Uml y patrones, introducción al análisis y diseño orientado a objetos*. México: Prentice Hall, 1999.

7. *Quanta Plus*. [Citado el: 9 de diciembre de 2007.] http://quanta.kdewebdev.org/ [En línea]

8. Sánchez Perodín, Yusdenis. *Pautas de diseño del Sistema de Información de Genética Médica.* Facultad de Bioinformática, Universidad de Ciencias Informáticas. Ciudad Habana, 2008.

9. Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia. *Rational Unified Process (RUP).* https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20a%20RUP.doc [En línea]

10. Lagos Torres, Manuel. *Introducción al diseño con patrones.* 2002 http://www.elrincondelprogramador.com/default.asp?id=29&pag=articulos/leer.asp [En línea]

11. Potencier, Fabien y Zaninotto, François. *Symfony, la guía definitiva*. 2008.

12. *Centro Nacional de Genética Médica.* http://www.sld.cu/sitios/genetica/temas.php?idv=7561 [En línea]

13. Sánchez Perodín, Yusdenis. *Estilo de código del Sistema de Información de Genética Médica*. Facultad de Bioinformática, Universidad de Ciencias Informáticas. Ciudad Habana, 2008.

14. *OMIM* - Online Mendelian Inheritance in Man.

15. *Medical Genetics Information Resource* (database online), University of Washington, Seattle. 1993-2007, 2007.

ANEXOS

Anexo 1 Descripción detallada de los casos de uso del sistema.

CU-1	**Gestionar Enfermedades Genéticas.**
Actores	Genetista Nacional (Inicia).
Propósito	Insertar, buscar y visualizar, modificar o eliminar una enfermedad genética, insertar, buscar y visualizar, o modificar una clasificación de enfermedad genética.
Referencia:	R3, R3.1- R3.7
Precondiciones	El genetista debe estar autenticado.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El genetista desea realizar una de las siguientes operaciones: • Insertar una enfermedad. • Buscar y visualizar una enfermedad. • Modificar una enfermedad. • Eliminar una enfermedad. • Insertar una clasificación de enfermedad. • Buscar y visualizar una clasificación de enfermedad. • Modificar una clasificación de enfermedad.	2. El sistema, en dependencia de la operación que solicita realizar, hace lo siguiente: • Si decide Insertar Enfermedad, ir a sección **"Insertar Enfermedad"**. • Si decide buscar y visualizar, modificar una Enfermedad o eliminarla, ir a la sección **"Editar Enfermedad"**. • Si decide Insertar una clasificación de enfermedad, ir a la sección **"Insertar Clasificación".** • Si decide buscar y visualizar una clasificación de enfermedad o modificar una clasificación de enfermedad, ir a la sección **"Modificar Clasificación".**
Sección "Editar Enfermedad"	
Acción del Actor	**Respuesta del Sistema**
	1. Muestra la interfaz correspondiente para la búsqueda de enfermedades.

2. Provee los datos al campo: - Nombre de la Enfermedad en español. - Nombre de la Enfermedad en inglés. - Código. - Clasificación. Solicita realizar la búsqueda.	4. Realiza la búsqueda, muestra las enfermedades, y le da al usuario la posibilidad de modificar o eliminar.
5. Escoge la opción de modificar	6. muestra la interfaz correspondiente para modificar una enfermedad.
7. Provee los datos al campo: - Nombre de la Enfermedad en español. - Nombre de la Enfermedad en inglés - Código. - Clasificación. - Descripción. Solicita modificar la enfermedad.	8. Actualiza la enfermedad en la base de datos si los campos están correctos. 9. Muestra una interfaz donde se informa que los datos han sido modificados correctamente.
Flujos Alternos	
5.1 Escoge eliminar una enfermedad. 6.1 Provee datos incorrectos o deja campos obligatorios vacios.	4.1 Si no existe la enfermedad muestra un mensaje **"No hay resultados acordes a los datos proporcionados"**. El sistema da la posibilidad de ir a la Sección "Insertar Enfermedad". 6.1 Elimina la enfermedad y muestra una nueva página avisando que se ha eliminado correctamente dicha enfermedad. 7.1 Muestra un mensaje señalando los campos que están incorrectos.
Sección "Insertar Enfermedad"	

Acción del Actor	**Respuesta del Sistema**
	1. Muestra la interfaz correspondiente para la inserción de una enfermedad.
2. Provee los datos al campo: - Nombre de la Enfermedad en español. - Nombre de la Enfermedad en inglés - Código. - Clasificación. - Descripción. Solicita insertar enfermedad.	3. Inserta la enfermedad en la Base de Datos si los campos están correctos. 4. Muestra una interfaz donde se informa que los datos han sido insertados correctamente.
Flujos Alternos	
2.1 Provee datos incorrectos o deja campos obligatorios vacios.	3.1 Muestra un mensaje señalando los campos que están incorrectos.
Prioridad:	Crítico
Sección “Insertar Clasificación”	
Acción del Actor	**Respuesta del Sistema**
	2. Muestra la interfaz correspondiente para la inserción de una clasificación.
3. Provee los datos al campo: - Nombre de clasificación. - Código de clasificación. - Descripción. Solicita insertar clasificación.	5. Inserta la clasificación en la Base de Datos si los campos están correctos. 6. Muestra una interfaz donde se informa que los datos han sido insertados correctamente.
Flujos Alternos	
2.1 Provee datos incorrectos o deja campos obligatorios vacios.	3.1 Muestra un mensaje señalando los campos que están incorrectos.
Prioridad:	**Crítico**
Sección “Modificar Clasificación”	

Acción del Actor	**Respuesta del Sistema**
	2. Muestra la interfaz correspondiente para la búsqueda de clasificación.
3. Provee los datos al campo: - Nombre de clasificación. - Código de clasificación. Solicita realizar la búsqueda.	4. Realiza la búsqueda, muestra las clasificaciones, y le da al usuario la posibilidad de modificar.
5. Escoge la opción de modificar	6. muestra la interfaz correspondiente para modificar una clasificación.
7. Provee los datos al campo: - Nombre de clasificación. - Código de clasificación. - Descripción. Solicita modificar la enfermedad.	8. Actualiza la clasificación en la base de datos si los campos están correctos. 9. Muestra una interfaz donde se informa que los datos han sido modificados correctamente.
Flujos Alternos	
7.1 Provee datos incorrectos o deja campos obligatorios vacios.	4.1 Si no existe la enfermedad muestra un mensaje **"No hay resultados acordes a los datos proporcionados"**. El sistema da la posibilidad de ir a la Sección "Insertar Clasificación". 8.1 Muestra un mensaje señalando los campos que están incorrectos.

Tabla 1: Descripción del caso de uso del sistema Gestionar enfermedades genéticas.

CU-2	**Gestionar Reporte Estadístico.**
Actores	Genetista (Inicia).
Propósito	Hacer un cálculo estadístico.
Resumen	El caso de uso se inicia cuando el genetista entra al sistema y le pide realizar una estadística. El sistema muestra una interfaz

	donde el genetista debe escoger el estadígrafo que va a calcular ya sea Tasa de Prevalencia, Tasa de Letalidad, Tasa de Morbilidad, Tasa de Mortalidad, y los parámetro por los que se va a regir que puede ser tiempo, edad, género, enfermedad, municipio o provincia. El genetista establece todas sus opciones y el sistema realiza los cálculos necesarios, muestra la respuesta en el formato tabla y da la opción de imprimir el resultado.
Referencias	R1, R1.1- R1.5
Precondiciones	El genetista debe estar autenticado.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El genetista desea realizar una estadística.	2. El sistema, en dependencia del nivel de acceso que tenga el genetista, muestra la interfaz para realizar el cálculo de la estadística.
4. Selecciona el tipo de estadística y los criterios para esta.	5. El sistema muestra una tabla con el resultado de la estadística y con los pacientes que cumplen con esas características.
Flujos Alternos	
3.1 Provee datos incorrectos o deja campos obligatorios vacios.	4.1 Muestra un mensaje señalando los campos que están incorrectos o vacios. 4.2 Muestra el mensaje: No existen resultados acordes a los criterios seleccionados.
Prioridad:	

Tabla 2: Descripción del caso de uso del sistema Gestionar reporte estadístico.

CU-3	**Gestionar Reporte de cálculos totales.**
Actores	Genetista (Inicia).
Propósito	Calcular totales.
Resumen	El caso de uso se inicia cuando el Genetista entra al sistema y va a la sección Calcular totales. El sistema muestra una interfaz donde brinda la opción de escoger los parámetros por los que se va a regir el cálculo. El genetista escoge los parámetros. El sistema realiza el cálculo, muestra los totales en una tabla dependiendo del nivel que se ha escogido (nacional, provincial o municipal) y da la posibilidad de imprimirlos.
Referencias	R2, R2.1- R2.6
Precondiciones	El genetista debe estar autenticado.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El genetista desea realizar un cálculo de totales.	2. El sistema, en dependencia del nivel de acceso que tenga el genetista, muestra la interfaz para realizar el cálculo de totales.
6. Selecciona los criterios para el cálculo.	7. El sistema muestra los totales en una tabla dependiendo del nivel que se ha escogido (nacional, provincial o municipal).
Flujos Alternos	
3.1 Deja campos obligatorios vacios.	4.1 Muestra un mensaje señalando los campos que están incorrectos o vacios. 4.2 Muestra el mensaje: No existen resultados acordes a los criterios seleccionados.
Prioridad:	

Anexo 2 Imágenes del registro.

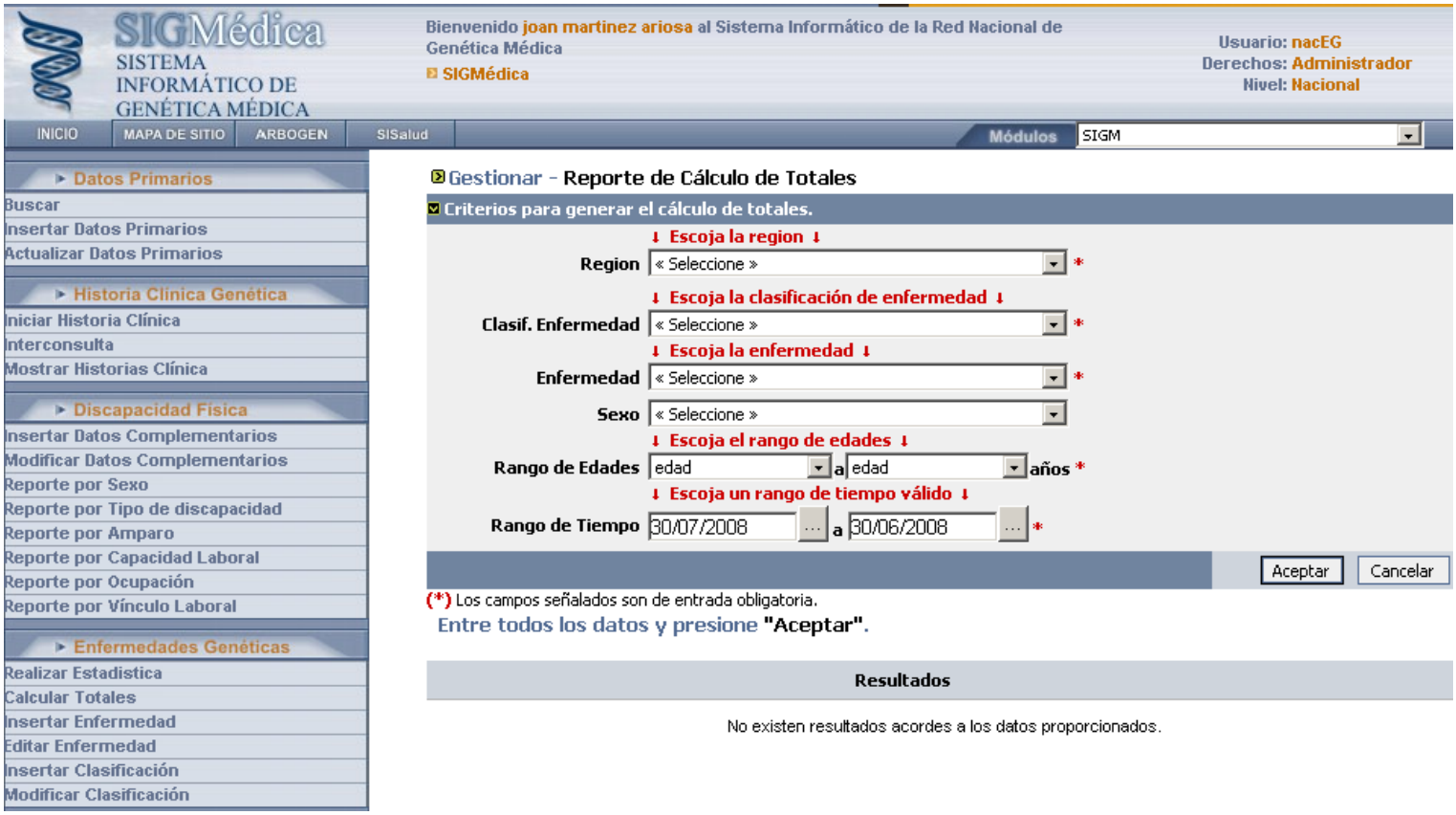

Figura 26. En la parte superior: Layout, a la izquierda: menú del SIGM (dividido en los submenús de cada registro), a la derecha: Interfaz del caso de uso gestionar reportes de cálculos totales (con avisos de validación).

Resultados de la Búsqueda			
Nombre Enfermedad (en español)	**Codigo Enfermedad**	**Clasificación**	
Síndrome de Down	1	Síndrome de Anomalías Cromosómicas	
Síndrome de la Trisomía 18	2	Síndrome de Anomalías Cromosómicas	
Síndrome de la Trisomía 13	3	Síndrome de Anomalías Cromosómicas	
Síndrome de la Trisomía 8	4	Síndrome de Anomalías Cromosómicas	
Síndrome de la Trisomía 9 en mosaico	5	Síndrome de Anomalías Cromosómicas	
Síndrome de la triploidía y Síndrome de mixoploidía/triploide	6	Síndrome de Anomalías Cromosómicas	
Síndrome de deleción 3p	7	Síndrome de Anomalías Cromosómicas	
Síndrome de duplicación3q	8	Síndrome de Anomalías Cromosómicas	
Síndrome de deleción 4p	9	Síndrome de Anomalías Cromosómicas	
Síndrome de deleción 4q	10	Síndrome de Anomalías Cromosómicas	

« < 1 - 2 - 3 > »

Figura 27. Resultado de una búsqueda en la sección Editar Enfermedad.

GLOSARIO

Apache: Es un software (libre) servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

API: Application Programming Interface (en español: Interfaz de Programación de Aplicaciones).

Base de datos: Conjunto de datos organizados entre los cuales existe una correlación y que están almacenados con criterios independientes de los programas que los utilizan. La filosofía de las bases de datos es la de almacenar grandes cantidades de datos de una manera no redundante y que permita las posibles consultas de acuerdo a los derechos de acceso.

Caso de uso: Fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.

CGI: Common Gateway Interface, importante tecnología de la World Wide Web que permite a un cliente (explorador web) solicitar datos de un programa ejecutado en un servidor web.

CNGM: Son las siglas del Centro Nacional de Genética Médica.

CPU: Central Processing Unit (en español: Unidad Central de Procesamiento), es la denominación oficial del procesador de una computadora.

Copyright: Derecho de autor, es un conjunto de normas y principios que regulan los derechos morales y patrimoniales que la ley concede a los autores.

CSS: Cascading Style Sheets (en español: Hojas de Estilo en Cascada). Son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML.

Etiología: Estudio de las causas de las enfermedades. Parte de la medicina que tiene por objeto el estudio de las causas de enfermedades.

Formulario: Parte de una página Web que el usuario completa y devuelve al servidor para su procesamiento.

FTP: File Transfer Protocol. Es un protocolo de transferencia de archivos entre sistemas conectados a una red basado en la arquitectura cliente-servidor.

Genética: Rama de las ciencias biológicas que trata de comprender cómo la herencia biológica es transmitida de una generación a la siguiente, y cómo se efectúa el desarrollo de las características que controlan estos procesos.

GoF: Gang of Four, conjunto de patrones de diseño.

GPL: General Public License (en español: Licencia Pública General), es una licencia creada por la Free Software Foundation y orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre.

GRASP: General Responsibility Assignment Software Patterns (en español: Patrones de software de asignación de responsabilidades a objetos).

Herramienta CASE: (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador). Aplicación informática destinada a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero.

HTML: Acrónimo inglés de Hyper Text Markup Language (lenguaje de marcación de hipertexto), es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo Explorer o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos.

HTTP: Protocolo usado para la transferencia de documentos a través de la World Wide Web. Estas transferencias requieren un programa cliente http en un extremo de la comunicación y un servidor http en el otro.

Infomed: Red de Salud de Cuba.

Interfaz: Es la parte de un programa informático que permite a éste comunicarse con el usuario o con otras aplicaciones permitiendo el flujo de información.

KDE: K Desktop Environment o Entorno de Escritorio K, entorno de escritorio e infraestructura de desarrollo para sistemas Unix/Linux.

Macs: Computadoras personales diseñadas, desarrolladas, construidas y comercializadas por Apple Inc.

MySQL: Es un sistema gestor de base de datos, multihilo y multiusuario, software libre en un esquema de licenciamiento dual.

MVC: Patrón arquitectónico Modelo Vista Controlador.

Navegador: Es una aplicación software que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web de todo el mundo a través de Internet. Esta red de documentos es denominada World Wide Web (WWW) o Telaraña Mundial. Los navegadores actuales permiten mostrar y/o ejecutar: gráficos, secuencias de vídeo, sonido, animaciones y programas diversos además del texto y los hipervínculos o enlaces.

ODBC: Open Database Connectivity, estándar de acceso a Bases de Datos desarrollado por Microsoft.

PC: Personal Computer (en español: computadora personal).

PDF: Portable Document Format, (en español: Formato de Documento Portátil).

PDT: PHP Development Tool.

Plugin: Es un programa de ordenador que interactúa con otro programa para aportarle una función o utilidad, generalmente muy específica.

Protocolo: Conjunto de reglas que controlan la secuencia de mensajes que ocurren durante una comunicación entre entidades que forman una red. En este contexto, las entidades de las cuales se habla son programas de computadora o automatismos de otro tipo, tales y como dispositivos electrónicos capaces de interactuar en una red.

Reporte: Informe detallado sobre alguna información, o sobre el estado de la información.

SAAA: Componente de seguridad basado en el modelo de Autenticación, Autorización y Auditoría.

Servidor: Una computadora que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes. Algunos servicios habituales son los servicios de archivos, que permiten a los usuarios almacenar y acceder a los archivos de un ordenador y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final.

Servidor Web: Programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para transferir los llamados hipertextos, páginas web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos.

SGBD: Sistema Gestor de Base de Datos. Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez.

SIGM: Sistema Informático de Genética Médica.

SISalud: Sistema de Información para la Salud.

Software: Es la parte lógica del ordenador, esto es, el conjunto de programas que puede ejecutar el hardware para la realización de las tareas de computación a las que se destina. Es el conjunto de instrucciones que permite la utilización del equipo.

Versión: Término que nombra las actualizaciones de un producto, se utiliza cuando se saca al mercado, o bien, cuando las modificaciones que se hacen del antiguo son muy numerosas o de gran alcance.

XML: Extensible Markup Language (en español: Lenguaje de Marcas Extensible). Es un metalenguaje extensible de etiquetas.

YAML: Es un formato para serializar datos que son fáciles de procesar por las máquinas, fácil de leer para las personas y fácil de interactuar con los lenguajes script.

1

1

2