Universidad de Ciencias Informáticas

Facultad 6

TÍTULO: "ALASMEDIGEN: REGISTRO GENÉTICO PREVENTIVO DE FAMILIAS CON ENFERMEDADES COMUNES".

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

Autores: Ángel Manuel García Vidal
Randy Ibarrola Suárez

Tutores: Ing. Yoendris Lacoste Ricardo
Lic. Tomás Gámez Acosta

Ciudad de La Habana, Cuba.
18 Junio, 2009.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de esta 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 junio del año 2009.

Ángel Manuel García Vidal

_____

Firma del autor

Randy Ibarrola Suárez

_____

Firma del autor

Ing. Yoendris Lacoste Ricardo

_____

Firma del tutor

Lic. Tomás Gámez Acosta

_____

Firma del tutor

Agradecimientos

Randy Ibarrola Suárez

Al término de esta etapa de mi vida, quiero expresar un profundo agradecimiento a quienes con su ayuda, apoyo y comprensión me alentaron a logar esta hermosa realidad.

A mis padres, quienes me han heredado el tesoro más valioso que puede dársele a un hijo: amor. A quienes sin escatimar esfuerzo alguno, han sacrificado gran parte de su vida para formarme y educarme. A quienes la ilusión de su vida ha sido convertirme en persona de provecho. A quienes nunca podré pagar todos sus desvelos ni aún con las riquezas más grandes del mundo.

A mis hermanos, por ser mis mejores amigos, por darme toda su confianza y apoyo incondicional

A Yanelis, el amor y la mujer de mi vida, por tantas horas de entrega, dedicación y comprensión.

A mi tía, a mi prima.

A mis cuñadas Sandra y Marlenis

A mis suegros, que siempre me han acogido como un hijo más.

A los viejos amigos de Matanzas,

A los nuevos amigos de la Universidad,

A toda mi gente del edificio 71, por todos los buenos momentos.

A los pibes del futbol, con los que compartí muchas tardes.

A todo el antiguo grupo 4, por demostrarme que son los mejores compañeros.

A todo el proyecto de Genética Médica, a mi tutor Yoendris Lacoste y al profesor Andrés Ballester.

A mi compañero de tesis y amigo Angel.

A todos los que de una manera u otra influyeron en que pudiera alcanzar este logro.

Ángel Manuel García Vidal

Ya han pasado cinco largos años desde que llegué por primera vez a esta maravillosa universidad. Hoy tengo la satisfacción y la dicha de poder decir que dejé mi pequeña huella en esta escuela que me dio la posibilidad de formarme como profesional y brindarme todos los conocimientos que he adquirido. Han sido muchos los que han contribuido a mi formación y educación, hoy a todos ellos quiero expresarles mis más profundos agradecimientos.

A mis padres, por educarme y darme todo su amor, por dedicarme cada minuto de sus vidas.

A mi hermana, por cuidarme y por quererme tanto, por ser tan buena conmigo, por ser mi hermana.

A Alelí, por quererme tanto, por ser mi guía en estos 5 años, por compartir tantas cosas juntos, y las que nos faltan

A mis suegros y demás familiares de mi novia, que me han acogido con cariño y me han brindado todo su apoyo.

A mis amistades del barrio y de siempre, a los amigos del pre, a los de aquí de la universidad, a todos lo que siempre han estado, en las buenas y en las malas, a esos con los que comparto diariamente, por ser mis amigos, muchas gracias.

A todas las personas que han contribuido a mi formación individual, estudiantil y profesional, a todos los que ha apostado su granito de arena por mí, espero no defraudarlos, muchísimas gracias.

Dedicatoria

A nuestros padres, por su confianza, apoyo y amor incondicional.

A todos nuestros familiares.

A nuestras novias.

A nuestros amigos.

RESUMEN

El objetivo fundamental del Centro Nacional de Genética Médica es desarrollar proyectos de investigación e innovación, en el campo de la Genética Médica, Inmunología y disciplinas afines. Además de evaluar e introducir nuevas tecnologías para el diagnóstico, tratamiento y asesoramiento en relación con las enfermedades genéticas y desarrollar investigaciones en familias con enfermedades genéticas multifactoriales, en la búsqueda de los genes que contribuyen o están relacionados con su aparición, bajo los principios éticos establecidos. En los últimos años se ha logrado no solo secuenciar el llamado alfabeto de la vida, así como los genes causantes de un número alto de enfermedades genéticas, sino también hemos aprendido como las enfermedades más frecuentes en las diferentes poblaciones del mundo presentan factores de riesgo genéticos, asociados a su origen. Varios son los ejemplos: hipertensión arterial, diabetes mellitus, esquizofrenia, diferentes variantes del cáncer, trastorno bipolar, depresión, demencias, coronariopatías, entre otras. Uno de los campos que engloban estas investigaciones son los de los factores genéticos asociados al origen de las enfermedades comunes. Para hacer posibles estos estudios es vital la recolección, almacenamiento y actualización de la información. Como solución a esta problemática se decide desarrollar por parte de la Universidad de las Ciencias Informáticas el Registro Genético Preventivo de Familias con Enfermedades Comunes, integrado al Sistema Informático para la red Nacional de Genética Médica: alasMEDIGEN, dando la posibilidad a todos los genetistas del país poder llevar a cabo los estudios previamente mencionados, desde la red nacional de salud con mayor rapidez y eficacia.

Palabras claves: enfermedades comunes, registro, sistema informático, hipertensión arterial, diabetes mellitus, esquizofrenia, diferentes variantes del cáncer, trastorno bipolar, depresión, demencias, coronariopatías.

TABLA DE CONTENIDOS

Índice

Introducción.

Desde el mismo comienzo de la Revolución cubana, en 1959, una de las directrices principales del estado fue mejorar el sistema de salud pública, que se encontraba en una situación precaria. Como parte de esta estrategia se crea el Ministerio de Salud Pública (MINSAP), con el objetivo de organizar toda una nueva infraestructura de centros tanto para la atención a la población como para el desarrollo de las investigaciones científicas en el campo de la medicina. Una de esas instituciones creadas fue el Centro Nacional de Genética Médica (CNGM), el cual tiene dentro de sus funciones principales las investigaciones básicas en el campo de la Genética Médica, la Inmunología, la Bioquímica y otras disciplinas afines dirigidas a la obtención de nuevos conocimientos, evaluación y desarrollo de nuevas tecnologías, productos y procedimientos de trabajo, con el fin de mejorar los niveles de salud de nuestro pueblo y disminuir el impacto de las enfermedades con implicación genética, así como desarrollar recursos técnicos para la educación y formación de una cultura genética en la población.

El CNGM es el centro de referencia nacional para el Programa Cubano de Diagnóstico Manejo y Prevención de Enfermedades Genéticas y Defectos Congénitos, el cual está compuesto por diferentes registros, estos almacenan todo el volumen de información necesaria para poder realizar los estudios pertinentes, tales como: el Registro Cubano de Malformaciones Congénitas, el Registro Cubano de Discapacitados, el Registro Cubano de Gemelos, el Registro Cubano de las Personas con Retraso Mental, entre Registro Cubano de Historias Clínicas, Registro Cubano de Enfermedades Genéticas y de más reciente creación el Registro Genético Preventivo de Familias con Enfermedades Comunes, siendo este ultimo esencial para determinar los factores de riesgos genéticos asociados al origen de enfermedades comunes. Ahora, los procesos actuales de almacenamiento y transferencia de datos sobre las enfermedades comunes desde la provincias hasta el Centro Nacional de Genética Médica no se realizan con eficiencia, debido al alto riesgo que tienen de perderse o deteriorarse toda esta información archivada en papeles y tampoco existe una estructura que haga fluir los datos con rapidez, para el estudio del comportamiento de muchas enfermedades.

Es por esto que surge el siguiente problema científico **¿Cómo contribuir a la gestión de los estudios de factores de riesgos genéticos asociados al origen de las enfermedades comunes?**

Derivándose como objeto de estudio de nuestra investigación **El proceso de gestión de información de la salud** enmarcado en el campo de acción: **Proceso de gestión de la**

información de los factores de riesgos genéticos asociados al origen de las enfermedades comunes de la población cubana.

Surgiendo como objetivo general: **Desarrollar el módulo Registro Genético Preventivo de Familias con Enfermedades Comunes.**

Para lo que se proponen los siguientes objetivos específicos:

- Identificar las funcionalidades que tendrá el módulo Registro Genético Preventivo de Familias con Enfermedades Comunes.
- Diseñar el módulo Registro Genético Preventivo de Familias con Enfermedades Comunes.
- Implementar el módulo Registro Genético Preventivo de Familias con Enfermedades Comunes.

Para solucionar el problema científico planteado y lograr con éxito el cumplimiento de los objetivos, se realizaron las siguientes tareas:

- Estudio de los sistemas de gestión de enfermedades comunes en la población.
- Análisis de la arquitectura establecida para alasMEDIGEN.
- Modelado de los procesos de negocio del módulo.
- Levantamiento de los requisitos del módulo.
- Diseño de las clases del módulo.
- Implementación de los componentes del módulo.

El desarrollo del Registro Genético Preventivo de Familias con Enfermedades Comunes posibilitará que los especialistas gestionen la información de las personas referente a las enfermedades comunes de nuestro país de una manera segura, rápida, eficiente y eficaz. El mismo estará integrado al Sistema Informático para la Red Nacional de Genética Médica, alasMEDIGEN.

El presente trabajo de diploma se estructura en Introducción, cuatro capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía y anexos.

En el **Capítulo 1: Fundamentación Teórica,** se hace una explicación de los sistemas de gestión de enfermedades de la población como el que se piensa desarrollar, la metodología, tecnología, herramientas, lenguajes y patrones de diseño a utilizar para la confección del mismo.

En el **Capitulo 2: Características del Sistema,** se abunda acerca del negocio en que esta enmarcada la investigación, se definen actores y trabajadores, y los casos de usos del mismo. Igualmente se definen los requerimientos del sistema, actores, casos de usos y sus diagramas, y los requerimientos funcionales y no funcionales del sistema. Además de un análisis de la arquitectura establecida para alasMEDIGEN.

En el **Capitulo 3: Diseño,** se indaga en todo lo referente a esta fase de desarrollo, mostrándose todos los detalles que respectan al diseño, las clases y los diagramas propios de esta fase: diagramas de clases, diagramas de interacción y despliegue. También se tocan aspectos como la arquitectura y patrones utilizados por el framework de desarrollo: Symfony, validaciones y prototipos de interfaz.

En el **Capitulo 4: Implementación,** se caracteriza la implementación del sistema, se analizan los estilos de codificación mostrándose una solución a los requisitos especificados. También se hacen referencias a los diagramas de componentes, código fuente de las clases más importantes y las interfaces utilizadas en la aplicación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

En el presente capitulo se hace una breve investigación acerca de los sistemas de gestión de enfermedades en la población, tanto a nivel nacional como internacional y las normativas para la creación de software para la salud cubana. También se abunda sobre el Sistema Informático para la Red Nacional de Genética Médica (SIGM), así como el objeto de estudio, el campo de acción, la metodología a utilizar, roles que se desarrollan, artefactos y patrones de diseño usados.

1.1 Sistemas de gestión de enfermedades en la población.

El uso de la informática como herramienta de ayuda a la medicina es una realidad en auge. El manejo de la información es algo integrado en la práctica clínica. Médicos y pacientes interactúan en una compleja matriz de información. El médico es un "manipulador" de la información en el sentido que adquiere, procesa, almacena, revisa y aplica en relación a la historia y evolución del paciente, a la realización de protocolos de diagnósticos y terapéuticos, al establecimiento de patrones poblacionales de enfermedad, al funcionamiento del sistema sanitario y al amplio almacén de literatura médica publicada. Pocas cosas en la relación médico paciente y en el trabajo sanitario en general, no están relacionadas de alguna forma con la obtención, el procesamiento y la aplicación de la información, junto a tareas de comunicación, como por ejemplo, obtención y registro de la información procedente del paciente, de su historia, de la consulta con otros especialistas, de la literatura médica, la selección de los procedimientos de diagnóstico o terapéuticos, la interpretación de los datos de laboratorio, o la recolección de datos con fines de investigación. Por estas características de la medicina, se van utilizando de forma cada vez más profusa, las ventajas de la informática en un entorno caracterizado por el aumento del número y la complejidad de las especialidades médicas, mayor disponibilidad y prestaciones de los ordenadores, precios más asequibles, mayor familiaridad médico - máquina, necesidad de guardar y transmitir gran cantidad de información y la presión socioeconómica que demanda una mayor eficacia en la gestión de los recursos.

Como parte de esta vinculación entre la medicina y la informática se han creado una gran variedad de sistemas de gestión de información de enfermedades de la población, teniendo como objetivo principal el almacenamiento y manipulación de los datos necesarios para realizar diferentes estudios con el fin de determinar las causas que propician tales enfermedades y cómo prevenirlas,

aprovechándose las ventajas que brinda el campo de la computación, al ganarse en organización, integridad y rapidez de procesamiento de dicha información. Los sistemas de gestión de información de salud existentes en la actualidad se centran en la gestión hospitalaria, automatización, bibliotecas virtuales y telemedicina, entre ellos se encuentran: Concerto Portal, eWorkflow Modules, Rhapsody Integration Engine, todos ellos forman parte de los programas informáticos para la asistencia sanitaria, así como para optimizar el intercambio de datos médicos de la empresa Orion Health, una de las prestigiosas en este campo; Axon, es una familia de productos especialmente diseñada para la gestión de todo tipo de Servicios médicos por Medigest Consultores S.L, cubre desde los aspectos clínicos hasta los administrativos; GalenusPro, software altamente especializado para la gestión integral avanzada de consultas y datos clínicos.

Todas estas aplicaciones tienen como características comunes que son desarrollados por empresas bien consolidadas en el mundo de la informática medica además de ser software que incluye un amplio soporte y por tanto se encarece tanto su adquisición y su mantenimiento.

1.2 Normativas en la creación de sistemas informáticos para la Salud Pública en Cuba.

Desde el mismo comienzo de la revolución cubana la máxima dirección del país y el Ministerio de Salud Pública han mostrado la voluntad de incorporar los avances de la informática y las nuevas tecnologías de la información y las comunicaciones a los procesos relacionados con la salud, con el objetivo de mejorar el servicio y la atención a la población.

Durante las últimas dos décadas un grupo de instituciones cubanas han desarrollado sistemas encaminados a alcanzar algún nivel de informatización de la salud, los cuales debido a que no tenían una definición generalizable, integración y los recursos tecnológicos para su despliegue en el Sistema Nacional de Salud(SNS) terminaban siendo proyectos aislados, es por eso que en 1997 se implementa una primera estrategia de informatización con la finalidad de coordinar esfuerzos para el desarrollo de este proceso en el SNS, dicha estrategia está caracterizada por el conjunto de métodos, técnicas, procederes y actividades administrativas dirigidas al manejo de la información de la salud, cuyo principal beneficiado será el paciente.

En estos momentos se trabaja integradamente en el desarrollo de un grupo de aplicaciones en la cual participan la Dirección Nacional de Informática del MINSAP, INFOMED, las Direcciones Nacionales del Ministerio de Salud Pública, Softel y la Universidad de las Ciencias Informáticas.

Ahora en esta nueva etapa se han definido por parte del MINSAP un grupo de estándares, premisas, herramientas y requisitos que garanticen la continuidad de los productos. Destacan entre estas normativas que todos los proyectos deben responder a las políticas y principios socialistas, todos los productos y servicios se integraran a la ciber-infraestructura del sector y se realizarán en lo fundamental sobre sistemas abiertos, arquitectura orientada a servicios, utilizando software libre y de calidad.

Los proyectos deben ser basados en el principio de software libre y acorde a la infraestructura, mantener las capas de la presentación, negocio y datos en tres servidores dedicados, por último el diseño e implementación retoman el concepto de la arquitectura orientada a servicios y el despliegue en tres capas. En el caso de la presentación, esta contendrá la interfaz de usuario y negocio de la validación de formularios de entrada de datos, la capa del negocio que contendrá los servicios que dan cumplimiento a los requisitos funcionales del sistema y la capa de datos que contiene la Base de Datos, todos los componentes siguen también principios de alta cohesión y bajo acoplamiento. Se desarrollará una arquitectura para el SIGM, teniendo en cuenta estos elementos. [1]

1.3 alasMEDIGEN, Sistema Informático para la red nacional de Genética Médica.

Con el objetivo de lograr un producto funcional mediante la integración de los proyectos independientes del CNGM acorde a la arquitectura de salud, surge el SIGM en el año 2007, donde cada registro de dicho centro representa una parte fundamental dentro del nuevo sistema que los integra, compartiendo el mismo tipo de arquitectura en su diseño, así como la Base de Datos que utilizan para el almacenamiento y gestión de la información. Este año con la aparición de nuevos registros del CNGM, debido a los amplios estudios en el campo de la medicina de carácter genético que se llevan a cabo en nuestro país por parte de esta institución, entre los que se encuentra el Registro Genético Preventivo de Familias con Enfermedades Comunes, se hace viable la creación de una nueva versión del sistema titulado: alasMEDIGEN Sistema Informático para la red nacional de Genética Médica, constituido por los siete registros anteriores y la informatización de dos nuevos registros.(ver Anexo 1)

1.4 Metodologías.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Son como un libro de recetas de cocina, en el que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.

Para el desarrollo de alasMEDIGEN se definió usar la metodología: Rational Unified Process (RUP), que constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. RUP es un proceso de software genérico que puede ser utilizado para una gran cantidad de tipos de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de competencia y diferentes tamaños de proyectos.

Provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de muy alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible. [2,3]

RUP divide en 4 fases del desarrollo del software:

Inicio: El Objetivo en esta etapa es determinar la visión del proyecto.
Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.
Construcción: En esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.
Transición: El objetivo es llegar a obtener la liberación del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. [4-6]

Las actividades en RUP se han agrupado en 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los 3 últimos como flujos de apoyo, siendo los siguientes: [4,7]

Modelo del Negocio.
Requerimientos.
Análisis y Diseño.
Implementación.
Pruebas.
Instalación.
Administración del proyecto.
Administración de configuración y cambios.
Ambiente.

(Ver Anexo 2)

El Proceso Unificado tiene 3 características principales:

1. Guiado/Manejado por casos de uso.
2. Centrado en arquitectura.
3. Iterativo e Incremental.

1.5 Lenguaje Unificado de Modelado (UML).

UML (Unified Modeling Languaje) o Lenguaje Unificado de Modelado es un lenguaje gráfico para especificar, construir, visualizar y documentar las partes o artefactos *(información que se utiliza o produce mediante un proceso de software).* UML es un lenguaje muy detallado y uniforme para el diseño Orientado a Objetos, incrementando las potencialidades de éxito en el desarrollo de los mismos. [8, 9].

Sus principales características son:

• Lenguaje unificado para la modelación de sistemas.

• Tecnología orientada a objetos.

• El cliente participa en todas las etapas del proyecto.

• Corrección de errores viables en todas las etapas.

• Aplicable para tratar asuntos de escala inherentes a sistemas complejos de misión crítica, tiempo real y cliente/servidor.

1.6 Tecnologías y herramientas a utilizar.

Para el diseño e implementación del Registro Genético Preventivo de Familias con Enfermedades Comunes es necesario cumplir con ciertas características que posibiliten su incorporación al Sistema Informático para la red nacional de Genética Médica, alasMEDIGEN. Para ello es imprescindible el uso de determinadas tecnologías y herramientas que faciliten el proceso de desarrollo de software.

Para ello se tiene PHP 5.0, como lenguaje de programación; MySQL 5.0, como sistema gestor de base de datos; Apache 2.0, como servidor de aplicaciones; Visual Paradigm 6.0, como herramienta case; Symfony 1.0.9, como framework para agilizar y hacer más fácil el proceso de desarrollo de software; Subversion 1.4.5, como sistema de control de versiones, para administrar las diferentes versiones del producto; Eclipse 3.0, como entorno de desarrollo.

1.7 Roles y artefactos.

Se mostrarán los roles y los artefactos que se generan en las dos dimensiones que propone RUP de acuerdo al ciclo de negocio, sistema, diseño e implementación correspondiente: Los flujos de trabajo de negocio, requerimientos, diseño e implementación en las fases de inicio, elaboración y construcción. [4,10]

Fase de inicio: se enfoca hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos. Durante la fase de inicio las iteraciones hacen mayor énfasis en actividades de modelado del negocio y de requerimientos.

Fase de elaboración: las iteraciones se orientan al desarrollo de la línea base de la arquitectura, abarcan más los flujos de trabajo de modelo de negocios (refinamiento), requerimientos, análisis, diseño y una parte de implementación orientado a la línea base de la arquitectura.

Fase construcción: se lleva a cabo la construcción del producto por medio de una serie de iteraciones. Para cada iteración se selecciona algunos Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan tantas iteraciones hasta que se termine la implementación de la nueva versión del producto.

Al emplear la metodología basada en RUP, el modelado del negocio plantea la identificación de los procesos del negocio y su completo análisis, el cual servirá como base para la identificación de probables candidatos a sistemas informáticos que soporten el negocio total o parcialmente.

1.7.1 Rol Arquitecto.

El Arquitecto dirige el desarrollo de la arquitectura de software del sistema, que incluye la promoción y la creación de soporte para las decisiones técnicas clave que restringen el diseño global y la implementación para el proyecto. El arquitecto de software tiene la responsabilidad global de dirigir las principales decisiones técnicas, expresadas como la arquitectura de software. Esto habitualmente incluye la identificación y la documentación de los aspectos arquitectónicamente significativos del sistema, que incluye las “vistas” de requisitos, diseño, implementación y despliegue del sistema.

También es responsable de proporcionar el fundamento de estas decisiones, equilibrando las preocupaciones de los diferentes interesados, reduciendo los riesgos técnicos, y garantizando que las decisiones se comunican, y validan con eficacia, y que se acatan.

Desde el punto de vista de la experiencia, el arquitecto de software también necesita compaginar las capacidades de Diseñador. Sin embargo, a diferencia del diseñador, el arquitecto de software:

- tiende a ser generalista en lugar de especialista, conoce muchas tecnologías a un alto nivel en lugar de pocas tecnologías a nivel de detalle
- toma decisiones técnicas más amplias y, por lo tanto, un amplio conocimiento y experiencia, así como habilidades de comunicación y liderazgo, son esenciales.

1.7.2 Rol Diseñador

El diseñador es el responsable de diseñar una parte del sistema informático 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

Encontramos entre los artefactos que se obtendrán realizados por el diseñador:

Realización de casos de uso del diseño: Es una colaboración en el modelo de diseño que describe como se realiza un caso de uso específico, 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.

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.

1.7.3 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.

Encontramos entre los artefactos que elabora el implementador:

Elementos de implementación: parte física de la implementación, incluyen los archivos y directorios. Incluyen ficheros de código (fuentes, binarios o ejecutables) y ficheros de datos.

El rol del implementador es responsable de los componentes de desarrollo y de prueba, de acuerdo con los estándares aprobados por el proyecto, para la integración en subsistemas más grandes. Cuando los componentes de prueba, como controladores o fragmentos para simulación, deben crearse para dar soporte a las pruebas, el implementador también es responsable del desarrollo y las pruebas de los componentes de prueba y los subsistemas correspondientes.

Las habilidades y conocimientos apropiados para el implementador incluyen:

- Conocimiento del sistema o aplicación que se somete a prueba
- Familiaridad con las herramientas de prueba y de automatización de prueba
- Habilidades de programación.

(Ver Anexo 3)

1.8 Patrones de Diseño.

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. [11,12]

1.8.1 Patrones GRASP.

Los patrones GRASP son patrones generales de software para asignación de responsabilidades a objetos, expresados en forma de patrones, es el acrónimo de "General Responsibility Assignment Software Patterns". Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y aplicable. [13,14]. De estos patrones se encuentran: el patrón creador, el experto, el controlador, bajo acoplamiento, alta cohesión, entre otros, algunos de ellos los veremos con más detenimiento en capítulos posteriores. [12-14]

Creador: Este patrón se encarga de asignarle a la clase B la responsabilidad de crear una instancia de clase A. B es un creador de los objetos A.

Experto: Este se encarga de asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Controlador: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase.

Alta Cohesión: Este patrón se encarga de asignar una responsabilidad de modo que la cohesión siga siendo alta. Una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas. Una clase de alta cohesión posee un número relativamente pequeño de métodos, con funcionalidad altamente relacionada y no realiza mucho trabajo. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande. El patrón Alta Cohesión presenta semejanzas con el mundo real, si alguna persona se le asignan demasiadas responsabilidades no será eficiente en ninguna de ellas.

Bajo Acoplamiento: Este patrón se encarga de asignar una responsabilidad para mantener bajo acoplamiento. Las clases deben comunicarse con un número pequeño de clases tanto como sea posible.

1.8.2 Patrones GOF.

Los patrones GOF se clasifican en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento. [15,16]

- **Creacionales**: Patrones creacionales tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
- **Estructurales**: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.
- **Comportamiento**: Los patrones de comportamiento nos ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

En el segundo nivel, se clasifican los patrones en 2 ámbitos: Clases y objetos. Es así que, tenemos 6 tipos de patrones:

Creacionales

- **Creacional de la Clase**

Los patrones creacionales de Clases usan la herencia como un mecanismo para lograr la instanciación de la Clase. Por ejemplo el método Factoría.

- **Creacional del objeto**

Los patrones creacionales de objetos son más escalables y dinámicos comparados con los patrones creacionales de Clases. Por ejemplo la Factoría abstracta y el patrón Singleton.

Estructurales

- **Estructural de la Clase**

Los patrones estructurales de Clases usan la herencia para proporcionar interfaces más útiles combinando la funcionalidad de múltiples Clases. Por ejemplo el patrón Adaptador (Clase).

- **Estructural de Objetos**

Los patrones estructurales de objetos crean objetos complejos agregando objetos individuales para construir grandes estructuras. La composición del patrón estructural del objeto puede ser cambiado

en tiempo de ejecución, el cual nos da flexibilidad adicional sobre los patrones estructurales de Clases. Por ejemplo el Adaptador (Objeto), Facade, Bridge, Composite.

Comportamiento

- **Comportamiento de Clase**

Los patrones de comportamiento de Clases usan la herencia para distribuir el comportamiento entre Clases. Por ejemplo Interpreter.

- **Comportamiento de Objeto**

Los patrones de comportamiento de objetos nos permite analizar los patrones de comunicación entre objetos interconectados, como objetos incluidos en un objeto complejo. Ejemplo Iterator, Observer, Visitor.

1.9 Conclusiones.

En este capítulo se realizo una breve explicación de los sistemas de gestión de enfermedades en la población y las normativas para la creación de los mismos en Cuba. Se realizo un estudio de lo que constituye el Sistema Informático para la red nacional de Genética Médica: alasMEDIGEN, así como las herramientas y metodologías utilizadas en el proceso de desarrollo de software. Siguiendo la metodología RUP se definen bien explicados los roles y principales artefactos generados correspondientes a los flujos de trabajos y fases del ciclo de vida del software y la aplicación de los patrones de diseño, GRASP y GOF.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

En este capítulo se muestra lo referente a los flujos de Modelado del Negocio y Levantamiento de Requisitos para el Registro Genético Preventivo de Familias con Enfermedades Comunes. Se hace una breve explicación del negocio actual para llegar a comprender bien el objetivo del sistema, donde se especifican los actores y trabajadores del negocio así como los artefactos generados y los casos de usos identificados. Se representan los diferentes diagramas en estos flujos de trabajo como son: Diagrama de casos de uso del negocio, diagrama de actividades del negocio, diagrama de objeto del negocio y la descripción textual de los casos de usos del negocio. Se especifican los actores del sistema, casos de usos del sistema, diagramas de casos de usos del sistema, así como la descripción textual y ampliada de los casos de usos del sistema. Posteriormente se detallan los requisitos funcionales y no funcionales y se hace un análisis de la arquitectura establecida para alasMEDIGEN.

2.1 Modelado de Negocio.

Un sistema, por pequeño que sea, generalmente es complicado. Por eso se necesita dividirlo en piezas si se pretende comprenderlo y gestionar su complejidad. Esas piezas se pueden representar a través de modelos que permitan abstraer sus características esenciales.

Una técnica para la especificación de los requisitos más importantes del sistema, que da soporte al negocio, es el modelo del negocio, con lo cual se refuerza la idea de que sea el propio negocio lo que determine los requisitos.

De ahí, que en el campo del software también resulte útil la creación de modelos que organicen y presenten los detalles importantes de problemas reales que se vinculan con el sistema informático a construir. Estos modelos deben cumplir una serie de propiedades, entre ellas la de ser coherentes y relacionados. Uno de los modelos útiles previo al desarrollo de un software es el modelo del negocio. [17,18]

Los objetivos del modelado del negocio son:

- Comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema.

- Comprender los problemas actuales de la organización e identificar las mejoras potenciales.

- Asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización.

- Derivar los requerimientos del sistema que va a soportar la organización.

2.1.1 Definición de negocio actual.

En el año 2004 la red de centros de genética médica del país dio inicio a una investigación en todo el territorio nacional para construir el registro cubano de familias con 13 enfermedades de alta prevalencia en nuestra población: asma bronquial, diabetes mellitus, hipertensión arterial, depresión, esquizofrenia, trastorno afectivo bipolar, cáncer de mama, cáncer de colon, cáncer de próstata, cardiopatía isquémica, enfermedad de Alzheimer, enfermedad de Parkinson y adicción al alcohol. Al finalizar el año 2007, el registro de familias con enfermedades comunes, tiene incorporadas más de 43 mil familias con más de 115 mil miembros afectados.

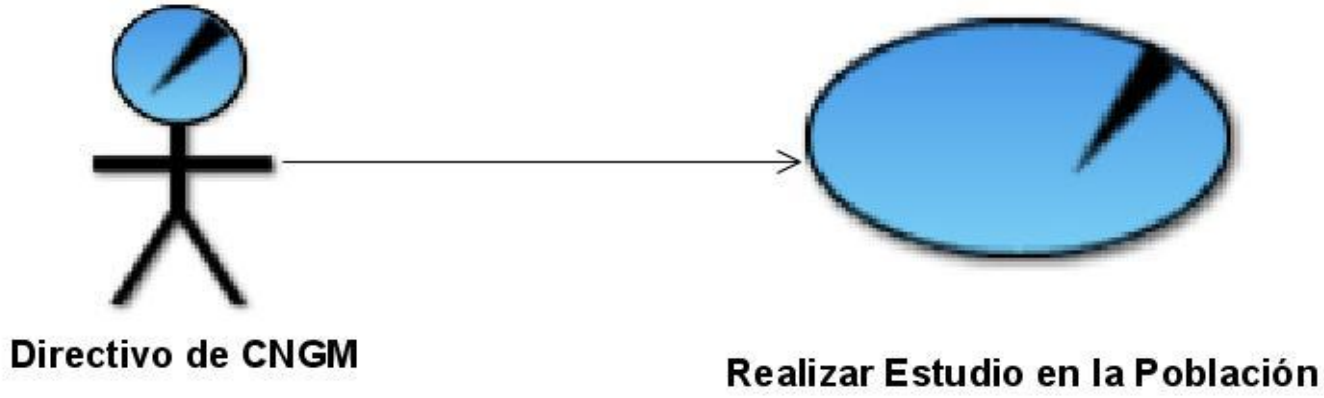
Esta valiosa información es utilizada sistemáticamente por los genetistas del CNGM en toda la red nacional de genética en proyectos de investigación, con el objetivo de realizar un estudio poblacional en nuestro país, archivando determinada información psicosocial de las familias cubanas para así caracterizar la distribución, comportamiento y factores de riesgo asociados al origen de esas enfermedades en las diferentes regiones del país y con ello llevar a cabo medidas pertinentes para su prevención, tratamiento y disminución.

2.1.2 Actores y trabajadores del negocio.

Actor	Descripción
Directivo del CNGM	Es la persona interesada en que se recoja la información psicosocial relacionada con los pacientes de manera correcta para posteriormente realizar los estudios pertinentes. Además está interesado en obtener los informes estadísticos posteriores al estudio.

Trabajador	Descripción
Genetista	Es el encargado de visitar los pacientes, y llenar el instrumento. Es el especialista encargado de revisar todos los instrumentos y visitar los casos de causa genética y los no precisados. Además de elaborar los informes estadísticos del estudio.
Médico de familia	Es el encargado de visitar los pacientes, y llenar el instrumento.
Trabajador Social	Es el encargado de realizar el censo de las personas que presentan algún tipo de enfermedad común.
Entrevistador	Es una generalización del Genetista y el médico de la familia, o sea es la persona que visita al paciente.

2.1.3 Casos de usos del negocio.

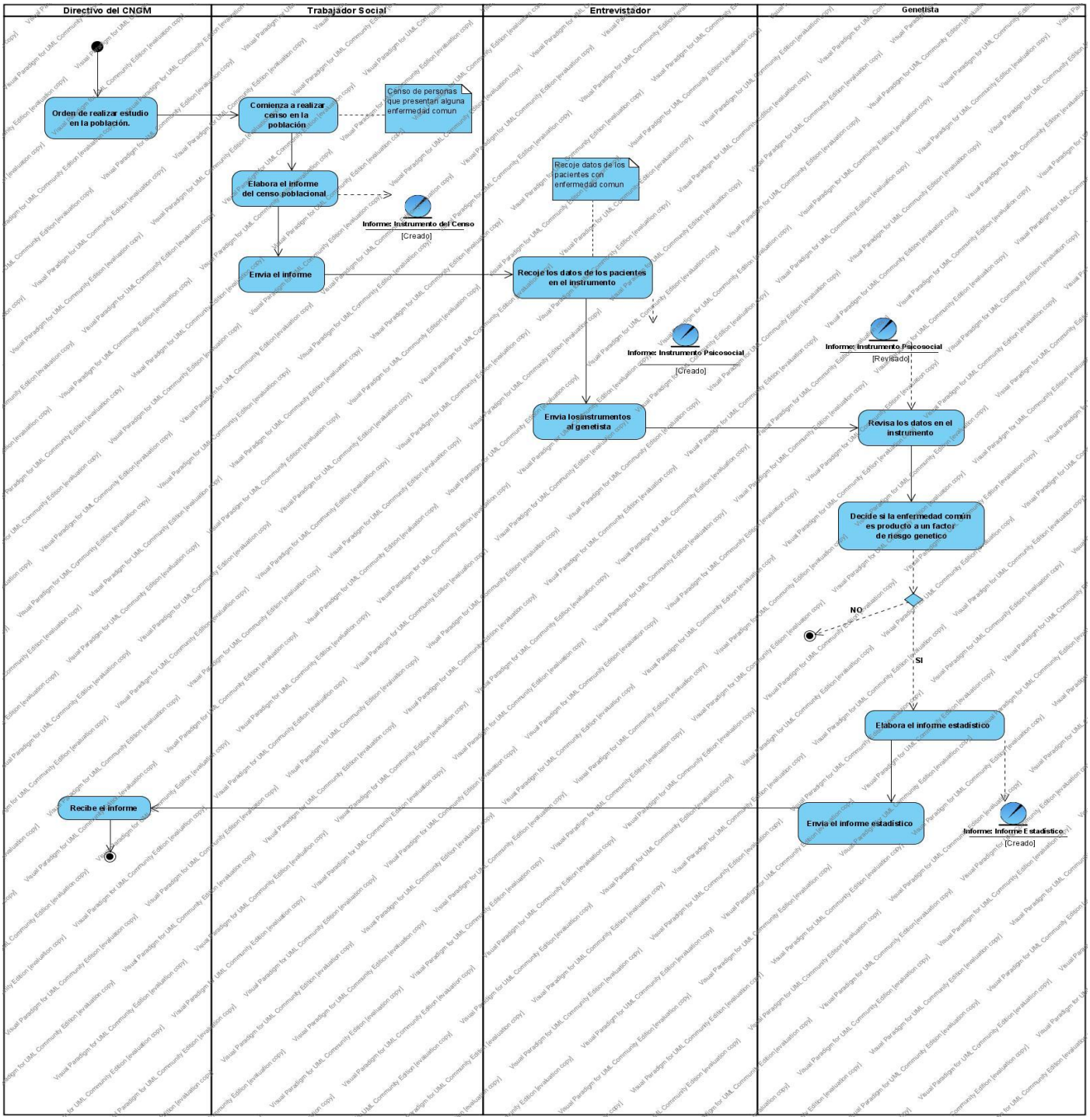


2.1.4 Descripción textual de casos de uso del negocio.

Caso de uso del negocio	Realizar Estudio en la Población
Actores	Directivo del CNGM (Inicia).
Trabajadores	Genetista, Médico de familia, Trabajador Social, Entrevistador.
Resumen	El caso de uso se inicia cuando el Directivo del CNGM le da la orden a los centros nacionales y municipales de realizar el estudio, el trabajador social realiza el censo poblacional, el entrevistador visita al paciente, se recogen los datos en el instrumento brindados por este y emite el diagnóstico. El genetista revisa los instrumentos, y los archiva y elabora el informe estadístico del estudio para entregárselo al Directivo del CNGM para dar

	conclusión así al caso de uso.
Casos de uso asociados	
Acción del actor	**Respuesta del proceso de negocio**
1. El Directivo del CNGM da la orden de que se realice el estudio y se comience el censo.	1.1 El Trabajador Social realiza el censo de todas las personas que presentan algún tipo de enfermedad común. 1.2 El Trabajador Social elabora el informe del censo. 1.3 El Trabajador Social envía el informe al entrevistador. 1.4 El entrevistador recoge los datos del paciente en el instrumento. 1.5 El entrevistador envía los instrumentos al genetista. 1.6 El genetista revisa los datos recogidos en el instrumento. 1.7 El genetista decide si la enfermedad común es producto de un factor de riesgo genético. 1.8 El genetista elabora el informe estadístico 1.9 El genetista envía el informe estadístico al directivo del CNGM.
1.10 El Directivo del CNGM recibe informe del estudio realizado.	

2.1.5 Diagrama de actividades del negocio.



2.1.6 Reglas del negocio.

- Un instrumento es una planilla donde se archivan todos los datos referentes a un paciente que padece de una determinada enfermedad común.

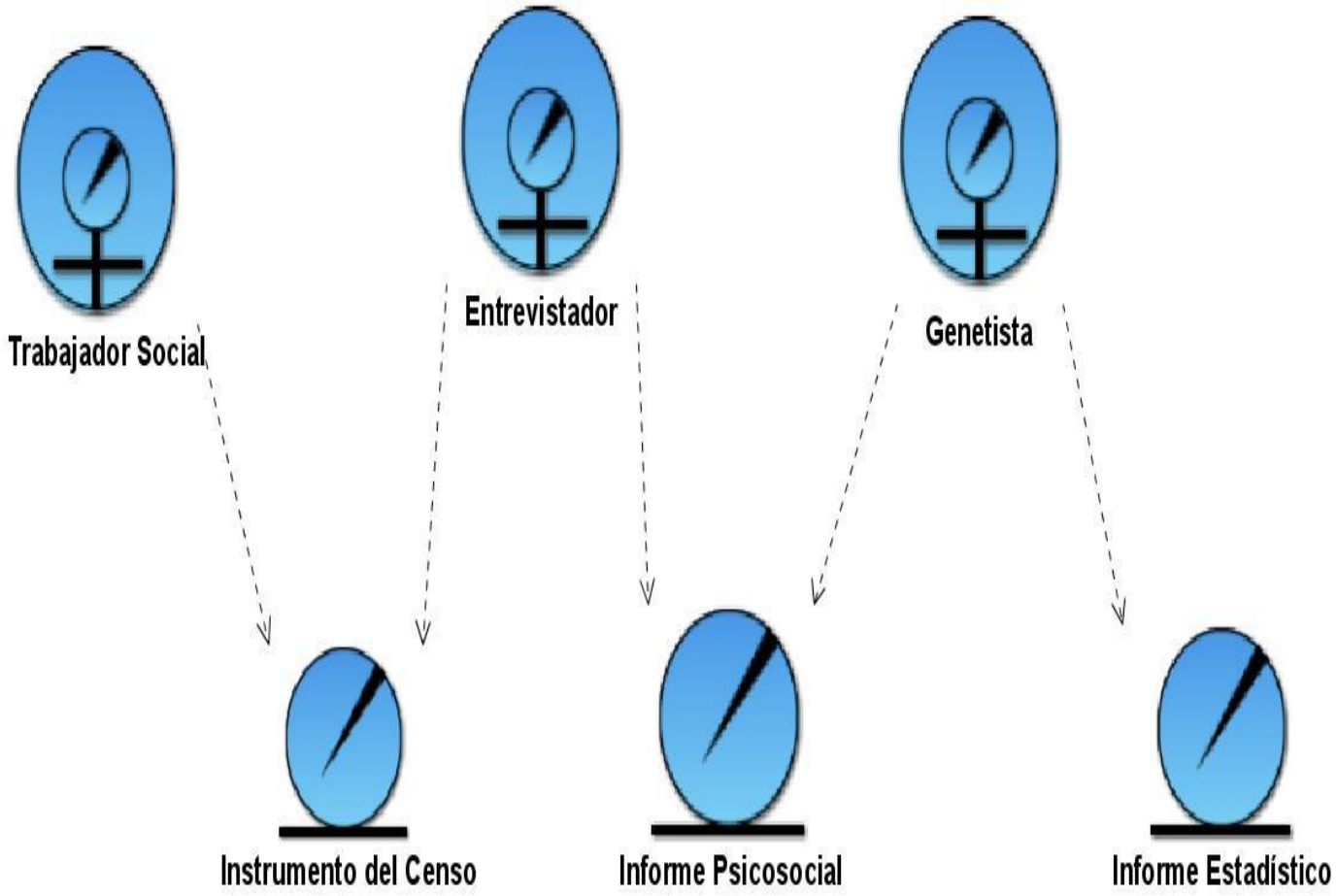
- Los genetistas son médicos especialistas y se dividen en tres grupos fundamentales: genetista municipal, genetista provincial y genetista nacional. El genetista provincial y municipal pueden atender al paciente y llenarle el instrumento que pertenece solo a ese único paciente, el cual será utilizado para los estudios correspondientes. Los genetistas municipales, provinciales y nacionales pueden cambiar y consultar alguna información en el instrumento del paciente que se encuentren bajo su responsabilidad y hacer los posteriores estudios para arribar a conclusiones en beneficio de todos los que padecen enfermedades comunes con factores de riesgo genético.

- Los genetistas pueden tener dos roles: revisores y editores los cuales tienen diferentes permisos de acuerdo al trabajo que desempeñen.

- Aunque la persona a la cual se le recogieron los datos haya fallecido, nunca se efectuará la eliminación física de un instrumento, ya que los datos que arroja son importantes a la hora de la realización de estudios para determinar las causas que originan las diferentes enfermedades comunes con factores de riego genético, así como cuáles son las que se presentan con mayor frecuencia en nuestra sociedad y cualquier otra investigación que sea de interés en el centro.

- La persona con enfermedades comunes va a ser atendido por el genetista de su municipio o provincia, el cual se encarga de recoger los datos solicitados en los instrumentos, para luego someterlo a un estudio de manera integral por un equipo que conduce el centro de genética.

2.1.7 Modelo de objetos del negocio.



2.2 Levantamiento de Requisitos.

La IEEE Standard Glossary of Software Engineering Terminology define un requerimiento como condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. [19]

Todas las ideas que los clientes, usuarios y miembros del equipo de proyecto tengan acerca de lo que debe hacer el sistema, deben ser analizadas como candidatas a requisitos.

En el flujo de trabajo de Levantamiento de Requisitos se persiguen los siguientes objetivos:

1. Definir el ámbito del sistema.
2. Definir una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario.
3. Establecer y mantener un acuerdo entre clientes y otros involucrados sobre lo que el sistema debería hacer.
4. Proveer a los desarrolladores un mejor entendimiento de los requerimientos del sistema.
5. Proveer una base para estimar recursos y tiempo de desarrollo del sistema.
6. Proveer una base para la planeación de los contenidos técnicos de las iteraciones.

2.2.1 Requerimientos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Constituyen las características que hacen al producto atractivo, usable, rápido o confiable al usuario.

El sistema 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).

Requisito de 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.

Requisito de Rendimiento

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

Requisito de 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.

Requisito de 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.

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.

Requisito de 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.

Requisito de 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.

Requisito de 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.

Requisito de 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 a1.7GHz 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.

Requisitos Legales.

Las herramientas y las tecnologías en que estará basada la aplicación informática deberán cumplir con las licencias de software libre.

2.2.2 Requerimientos funcionales.

- RF1 Insertar datos complementarios del paciente que posee enfermedades comunes.

El genetista municipal, provincial o nacional busca un paciente y lo seleccionan. Se muestra un formulario en el cual aparecen los datos generales del paciente y debajo los campos donde se introducen los datos complementarios del paciente.

- RF2 Modificar los datos complementarios del paciente.

El genetista municipal, provincial o nacional busca un paciente y lo selecciona. Se muestra un formulario donde se muestran los datos generales del paciente y debajo los campos donde se pueden modificar los datos complementarios insertados con anterioridad.

- RF3 Visualizar el árbol genealógico del paciente que se le van a insertar los datos complementarios.

El genetista municipal, provincial o nacional busca un paciente y lo selecciona. Se muestra un formulario donde se tienen todos los pacientes, se selecciona el paciente del cual se va a visualizar la imagen del árbol genealógico, la cual debió ser insertada por el Registro Cubano de Historias Clínicas de alasMEDIGEN.

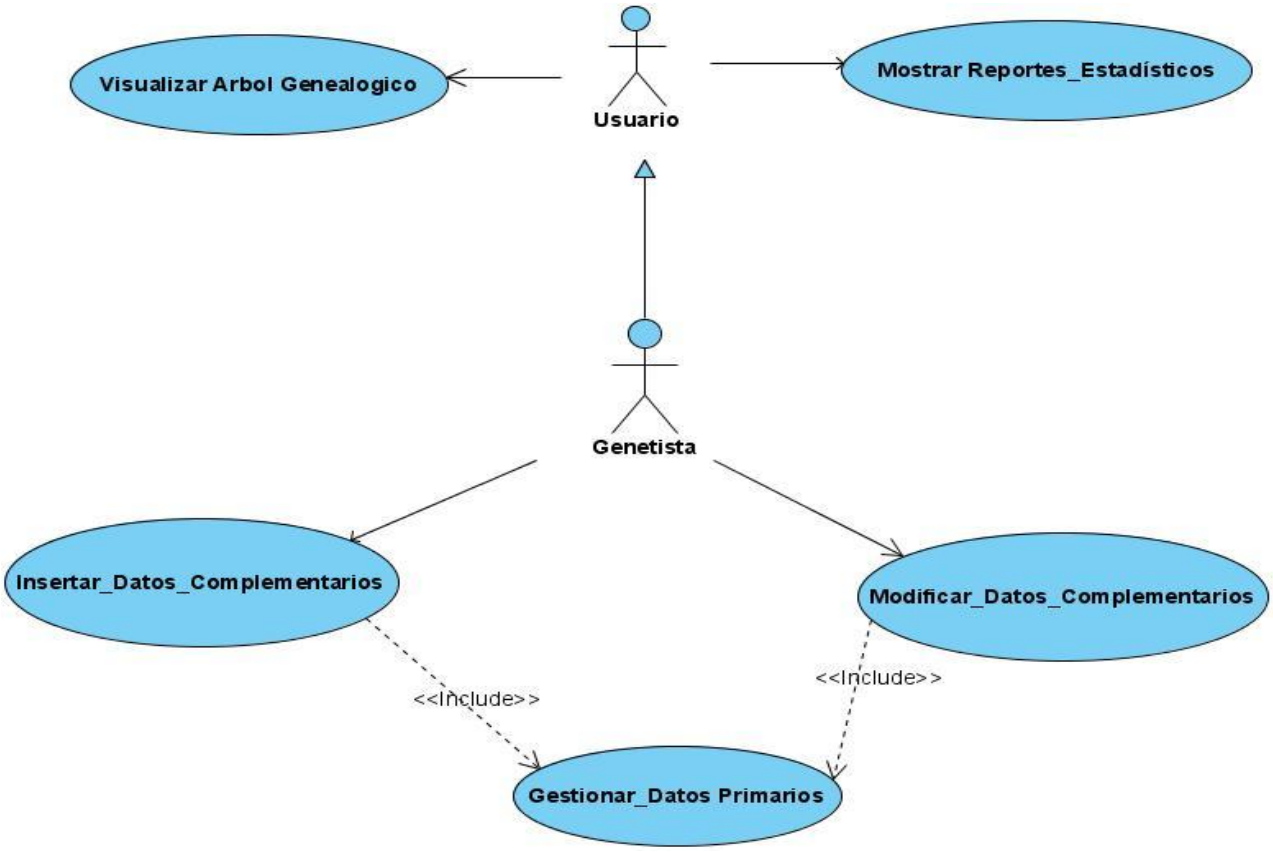
- RF4 Generar reportes.

El usuario, municipal, provincial o nacional va a generar reportes de los datos del paciente. El sistema le muestra la interfaz correspondiente y finaliza cuando se emite el resultado de la operación solicitada.

Reportes:

- RF4.1 Representar el nivel de incidencia de las enfermedades comunes con factor de riesgo genético en un mapa o gráfica, teniendo en cuenta el nombre de la enfermedad, la región y los apellidos.
- RF4.2 Representar el nivel de incidencia de las enfermedades comunes con factor de riesgo genético en las regiones, desde el nivel de área de salud hasta el nivel nacional en un mapa o gráfica.

2.2.3 Diagrama de casos de usos del sistema.



2.2.4 Descripción textual de casos de usos del sistema (Ver Anexo 5).

	Insertar Datos Complementarios del Paciente
Actores:	**Genetista** (Inicia el caso de uso)
Resumen:	El caso de uso se inicia cuando el genetista desea insertar los datos complementarios del paciente en el sistema, dicho paciente (persona que padece alguna enfermedad común con posibilidad de tener un factor de riesgo genético) ofrece los datos que se van a incluir en la aplicación y el genetista procede a la inserción de los mismos.
Precondiciones:	El genetista tiene que estar registrado para poder acceder a esta parte del sistema. Debe estar disponible la información a la que tiene acceso el usuario (los consejos populares, la provincia, etc.).
Referencias	RF1
Prioridad	Crítica
Flujo Normal de Eventos	
Sección “”	
Acción del Actor	**Respuesta del sistema**
1- El genetista selecciona la opción de insertar los datos complementarios de una paciente.	2- Se llama al Caso de uso incluido **“Gestionar Datos Primarios”** para buscar a la persona a la cual se le van a insertar los datos complementarios.
3- El genetista selecciona al paciente al cual le va a llenar los datos complementarios.	4- El sistema le brinda una interfaz que muestra los datos primarios del paciente y los criterios para la inserción de los datos complementarios**.**
5- El genetista registra los datos complementarios del paciente.	6- El sistema verifica que no se haya quedado ningún campo obligatorio vacío o campos con datos incorrectos.

	7- El sistema emite un mensaje diciendo que se ha insertado los datos correctamente.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del sistema**
	6- Se emite un mensaje para que se llene los campos obligatorios o se corrijan los datos erróneos.
Prototipo de Interfaz	
Poscondiciones	Quedará registrada toda la información de las personas que presentan alguna enfermedad común con factores de riesgos genéticos.

Caso de Uso:	**Modificar Datos Complementarios del Paciente**
Actores:	Genetista (Inicia el caso de uso)
Resumen:	El caso de uso inicia cuando el genetista necesita modificar los datos complementarios de un paciente. El genetista busca el paciente e introduce los datos que desea modificar.
Precondiciones:	El genetista debe estar registrado con los permisos pertinentes para poder acceder a esta parte del sistema. La información debe estar disponible.
Referencias	RF2
Prioridad	Secundario
Flujo Normal de Eventos	
Sección “”	
Acción del Actor	**Respuesta del Sistema**

1- El genetista selecciona la opción de modificar los datos complementarios de un paciente con alguna enfermedad común con factor de riesgo genético.	2- Se llama al Caso de uso incluido **"Gestionar Datos Primarios"** para buscar a la persona a la cual se van a modificar los datos complementarios.
3- El genetista selecciona al paciente al cual le va a modificar los datos complementarios.	4- El sistema le brinda una interfaz que muestra los datos primarios y complementarios del paciente a modificar.
5- El genetista registra los nuevos datos complementarios del paciente seleccionado y presiona el botón modificar.	6- El sistema verifica que no se haya quedado ningún campo obligatorio vacío o campos con datos incorrectos.
	7-El sistema modifica los datos complementarios del paciente, que fueron cambiados.
	8- El sistema emite un mensaje diciendo que se han modificado los datos correctamente.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	6- Se emite un mensaje para que se llene los campos obligatorios o se corrijan los datos erróneos.
Prototipo de Interfaz	
Poscondiciones	Deben guardarse las modificaciones en los datos del paciente.

2.2.5 Análisis de la arquitectura establecida para alasMEDIGEN.

Como parte de la estrategia para la informatización del Sistema Nacional de Salud Pública en Cuba, el grupo de arquitectura MINSAP-MIC ha establecido un grupo de normas y tecnologías para el desarrollo de aplicaciones informáticas destinadas a este sector. Estas premisas se han establecido con el objetivo de que se garantice la continuidad y sostenibilidad de los productos que se obtengan, y contribuir con su integración como pauta fundamental. Dentro de estas normas, se establece que cualquier aplicación informática desarrollada para la salud en Cuba, debe cumplir los siguientes requerimientos:

- Utilizar lenguaje de programación del lado del servidor PHP5.
- Funcionar sobre servidor Linux, distribución Debian (Sarge).
- Funcionar sobre Servidor Web Apache 2.0.
- Utilizar como sistema gestor de base datos MySQL5.0.
- Consumir o brindar servicios web para interactuar con otros sistemas.

Los sistemas desarrollados deben poder desplegarse en los servidores disponibles en Infomed.

Además las aplicaciones deben interactuar entre sí suministrando y/o consumiendo servicios web, según el esquema de seguridad desarrollado por Softel. Este mecanismo está implementado en el componente de seguridad (SAAA), que garantiza los tres elementos básicos de la seguridad:

- Confidencialidad: Que la información sea revelada sólo a los usuarios autorizados, en la forma y tiempo determinado.
- Integridad: Que la información sea modificada (incluyendo su creación y borrado) sólo por personal autorizado.
- Disponibilidad: Que la información sea utilizable cuándo y cómo lo requieran los usuarios autorizados.

El Sistema de Información para la Salud (SISalud) es un sistema desarrollado en el marco de la informatización del Sistema Nacional de Salud Pública en Cuba. Este presenta una arquitectura orientada a servicios (SOA) y un desarrollo basado en componentes.

2.3 Conclusiones.

Se definieron los flujos de trabajo de Modelado de Negocio y Levantamiento de Requisitos necesarios para lograr a un buen entendimiento con el cliente y propiciar la definición de las funcionalidades que presenta RECUEC. Se hizo una breve descripción del negocio en el que se enmarca el sistema con el propósito de llegar a comprender mejor las acciones que se automatizaron. Se representaron los diferentes diagramas acorde con los flujos de trabajo mencionados para una mejor comprensión gráfica y visual del sistema, y se especificaron los requisitos funcionales y no funcionales que caracterizan al Sistema Informático para la red nacional de Genética Médica.

CAPÍTULO 3: DISEÑO.

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida, y crear un plano del modelo de implementación. Durante la fase de construcción, cuando la arquitectura es estable y los requisitos están bien entendidos, el centro de atención se desplaza a la implementación. [20]

En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen.

En este capítulo encontramos las clases del diseño necesarias para la el Registro Genético Preventivo de Familias con Enfermedades Comunes, se encuentra una explicación de los patrones aplicados por el framework de desarrollo Symfony así como lo referente a la arquitectura Modelo-Vista-Controlador (MVC) implementada. Se hace una representación del modelo de la base de datos y un prototipo de interfaz de usuario. Se explica cómo Symfony lleva a cabo la seguridad del sistema y las validaciones de los formularios.

3.1 Patrones GRASP implementados.

Creador

Este patrón es el encargado de crear las instancias de los objetos para acceder a las entidades que brindan la información necesaria para realizar cualquier acción. En la clase ecActions encontramos las funciones definidas para el Registro Genético Preventivo de Familias con Enfermedades Comunes. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase ecActions es “creador” de dichas entidades. Un ejemplo de su uso es en la función CargarDatosComplementarios() donde se crean instancias de diferentes clases como: EcPaciente, EcSustituto, EcNocupacionPrincipal ,entre otras con el objetivo de manipular determinada información que poseen estas entidades, convirtiendo a ecAction en creadora de dichas entidades.

Experto

Este patrón es utilizado en el modelo, puesto que Propel es la librería externa que utiliza Symfony para realizar su ORM (object-relational mapping), o mapeo de objetos a bases de datos. El mismo garantiza acceder a la base de datos como si fuera orientada a objetos a partir de su estructura relacional. El framework Symfony divide el modelo en dos capas, la primera de acceso a datos y la segunda de abstracción de datos donde las clases generadas contienen todos los atributos y funcionalidades únicas para acceder a los datos correspondientes y modificarlos, es decir que cada una de estas clases es la encargada de brindar las funcionalidades de acceso a sus datos ya que son ellas mismas las que poseen dicha información.

Alta Cohesión

Symfony permite una excelente organización del trabajo en cuanto a la estructura y responsabilidades bien definidas para todos sus componentes, posibilitando que cada uno se centre en las tareas asignadas. Un ejemplo clave es el controlador, el cual delega en sus diferentes componentes, funciones para el manejo de los eventos del sistema estrechamente relacionados entre sí y donde la clase ecActions solo se encarga de definir las diferentes acciones a realizar delegando a cada una de ellas un objetivo especifico. Es decir cada parte se encarga únicamente de sus responsabilidades y se relacionan entre ellas para satisfacer necesidades del sistema en general manteniendo una alta cohesión.

Bajo acoplamiento

Este patrón permite que no existan dependencias innecesarias entre algunas partes del sistema que no tengan porque compartir o utilizar información que no necesiten. En la implementación MVC de Symfony, esto se evidencia en el modelo donde se manifiesta el uso de este patrón, ya que las clases de abstracción a la base de datos solo se relacionan con las de acceso a datos, los cambios que puedan realizarse en ellas tienen muy poca repercusión en el resto de los componentes asociados, dándole un alto grado de reutilización. Además no propicia que un cambio en una clase determinada repercuta en otra que no se relacione con esta.

Controlador

Este patrón es muy importante, ya observamos otros patrones capaces de crear instancias de otras clases, capaces de brindan determinada información que poseen solo ellos y otro que permiten que cada parte del sistema tenga sus propias responsabilidad independientes. Pero para lograr la

unidad entre todas las partes y que el sistema funcione como un todo es necesario este patrón controlador que se encargue de unir todas las partes. Todas las peticiones al sistema son manejadas por un controlador frontal que se divide en varios componentes encargados de la seguridad, validaciones, configuración y enrutamiento, de igual modo se definen las acciones a ejecutar por el sistema en la clase ecActions, parte fundamental de dicho controlador, encargada de realizar todas las acciones del sistema.

3.2 Tipos de patrones GOF que implementa Symfony

Creacionales:

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En la Clase sfRouting hay una llamada a sfRouting::getInstance(), que garantiza que exista una única instancia, creándola en caso que no exista o retornando la existente.

Prototype (Prototipo): Tiene como finalidad crear nuevos objetos duplicándolos, clonando una instancia previamente creada. Es usado para abstraer la lógica que decide que tipos de objetos se utilizarán, de la lógica que luego usarán esos objetos en su ejecución.

Cada clase base del modelo contiene una definición para el método Copy() y CopyInto(), para realizar la clonación de objetos.

En la categoría Estructurales:

Decorator (Envoltorio): Añade funcionalidad a una clase, dinámicamente. Por ejemplo un archivo layont.php, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este comportamiento es una implementación de este patrón de diseño.

En la categoría de comportamiento:

Command (Comando): Manipular las peticiones de los usuarios y enviarlas a un objeto encargado de darle respuesta a las mismas.

Dentro del controlador existe una clase denominada sfRouting que contiene un método llamado Parser, el cual identifica o desglosa la dirección URL que recibe como parámetro en módulo, acción.

Template Method (Método Plantilla): Define una estructura algorítmica en la súper clase, delegando la implementación a las subclases. Es decir, define una serie de pasos, en donde los pasos serán redefinidos en las subclases (polimorfismo). En Symfony se implementan de tipo abstract (abstractas) para poder permitir redefinir sus métodos como por ejemplo las clases bases generadas en el modelo.

Otros

Controlador Frontal ("Front Controller"): Todas las peticiones Web son manejadas por un controlador frontal, que es el punto de entrada único para toda la aplicación en un entorno determinado.

Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción definida en la clase ecActions y el nombre de un módulo con la URL entrado por el usuario, mientras otros componentes se encargan de la seguridad y validación mediante el uso de los archivos YML. Es decir se origina un flujo de eventos a partir del controlador frontal y se van delegando responsabilidades a cada componente hasta terminar en el ecActions.

Registry (Registro): Mecanismos para almacenar datos globales.

En la clase sfConfig se crea como mecanismo para almacenar las variables globales de configuración para una aplicación de symfony.

3.3 Arquitectura MVC que utiliza Symfony.

Modelo Vista Controlador (MVC) 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. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista. [21]

Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.

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

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

Específicamente para el framework de desarrollo utilizado, Symfony, las características de estas partes que componen el patrón MVC es la siguiente: [22] (Ver Anexo 4.)

El modelo: Solo se encarga del acceso a los datos almacenados en el gestor de base de datos. Ha sido dividido en dos capas, la capa de acceso a los datos y en la capa de abstracción de la base de datos. De esta forma, las funciones que acceden a los datos no utilizan sentencias ni consultas que dependen de una base de datos, sino que utilizan otras funciones para realizar las consultas. Así, si se cambia de sistema gestor de bases de datos, solamente es necesario actualizar la capa de abstracción de la base de datos.

La vista: Las páginas web suelen contener elementos que se muestran de forma idéntica a lo largo de toda la aplicación: cabeceras de la página, el layout genérico, el pie de página y la navegación global. En la mayor parte de las veces sólo cambia el interior de la página. Por este motivo, la vista se separa en un layout y en una plantilla. Normalmente, el layout es global en toda la aplicación o al menos en un grupo de páginas. La plantilla sólo se encarga de visualizar las variables definidas en el

controlador. Para que estos componentes interaccionen entre sí correctamente, es necesario añadir cierto código, código que será añadido a través de la lógica de la vista.

El controlador: Una parte importante de su trabajo es común a todos los controladores de la aplicación. Entre las tareas comunes se encuentran el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares. Por este motivo, el controlador se ha dividido en un controlador frontal, que se encarga de realizar las tareas comunes y las acciones, que incluyen el código específico del controlador de cada página.

3.4 Diagramas de clases del diseño Web.

Después de conocer la arquitectura MVC que utiliza Symfony y sus características podemos entender más fácilmente los diferentes diagramas de diseño del Registro Genético Preventivo de Familias con Enfermedades Comunes para los principales casos de uso.

Diagrama de clases del Diseño. Insertar Datos Complementarios.

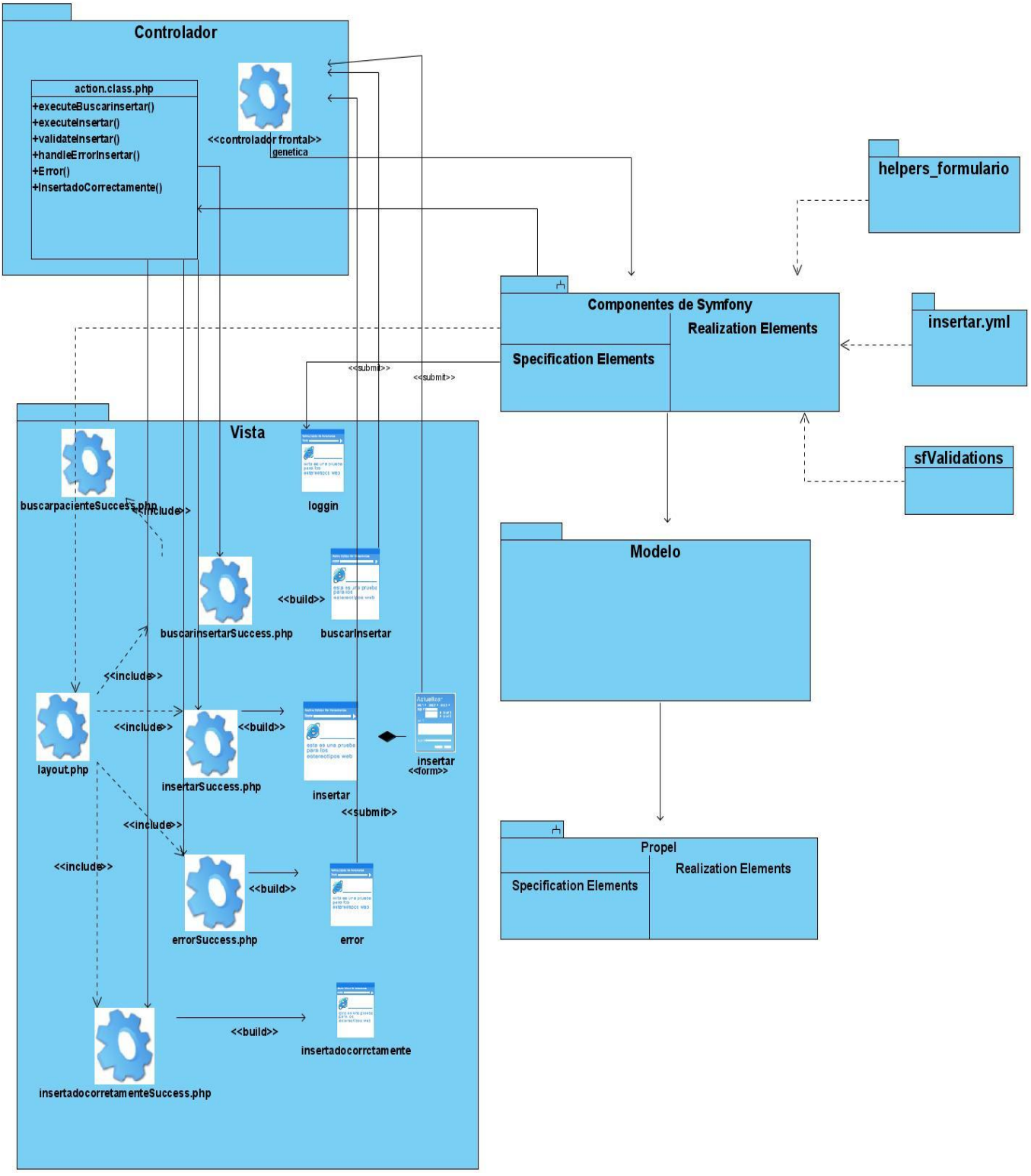


Diagrama de Clases del Diseño. Modificar Datos Complementarios.

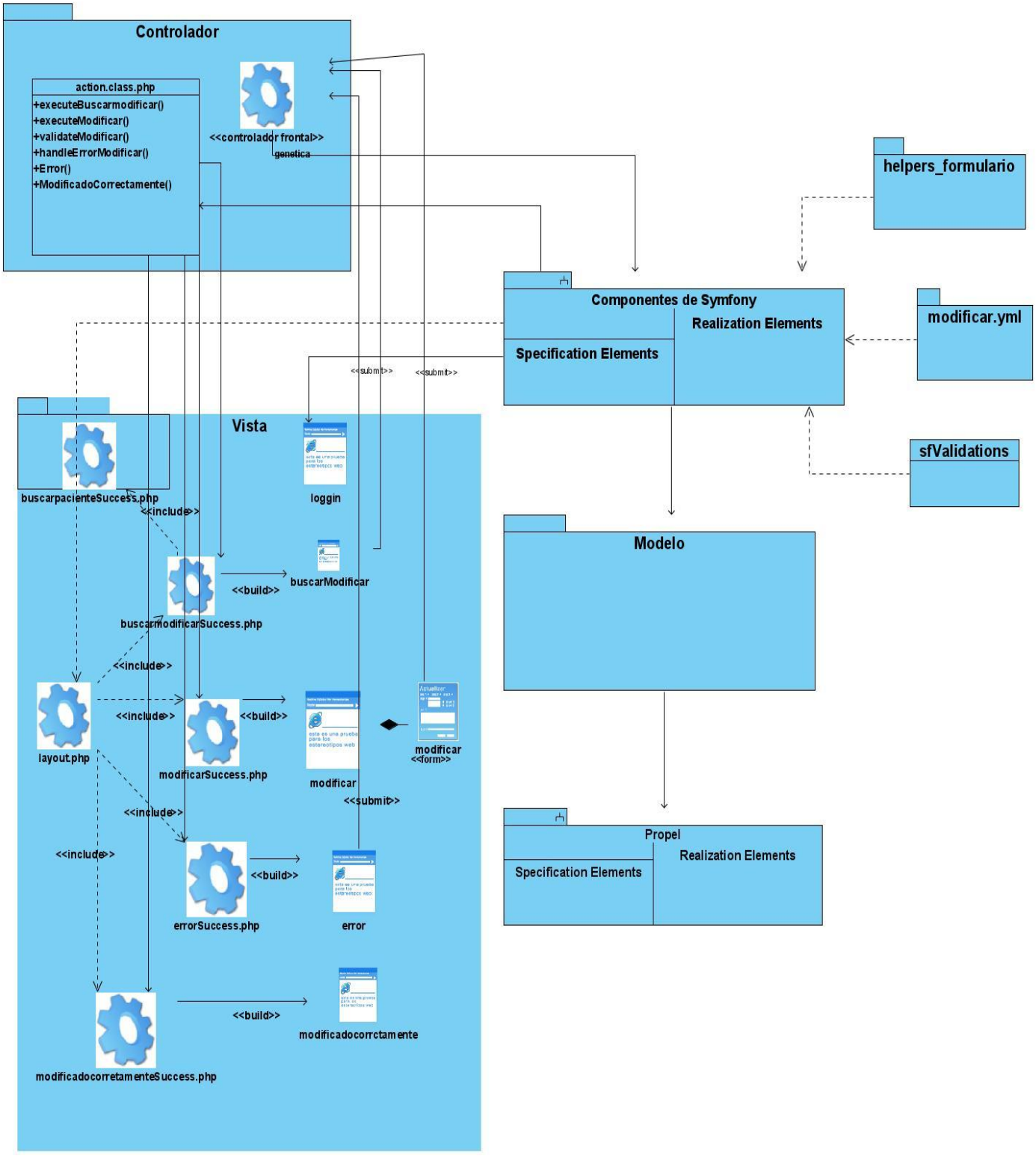


Diagrama de Clases del Diseño. Visualizar Árbol Genealógico.

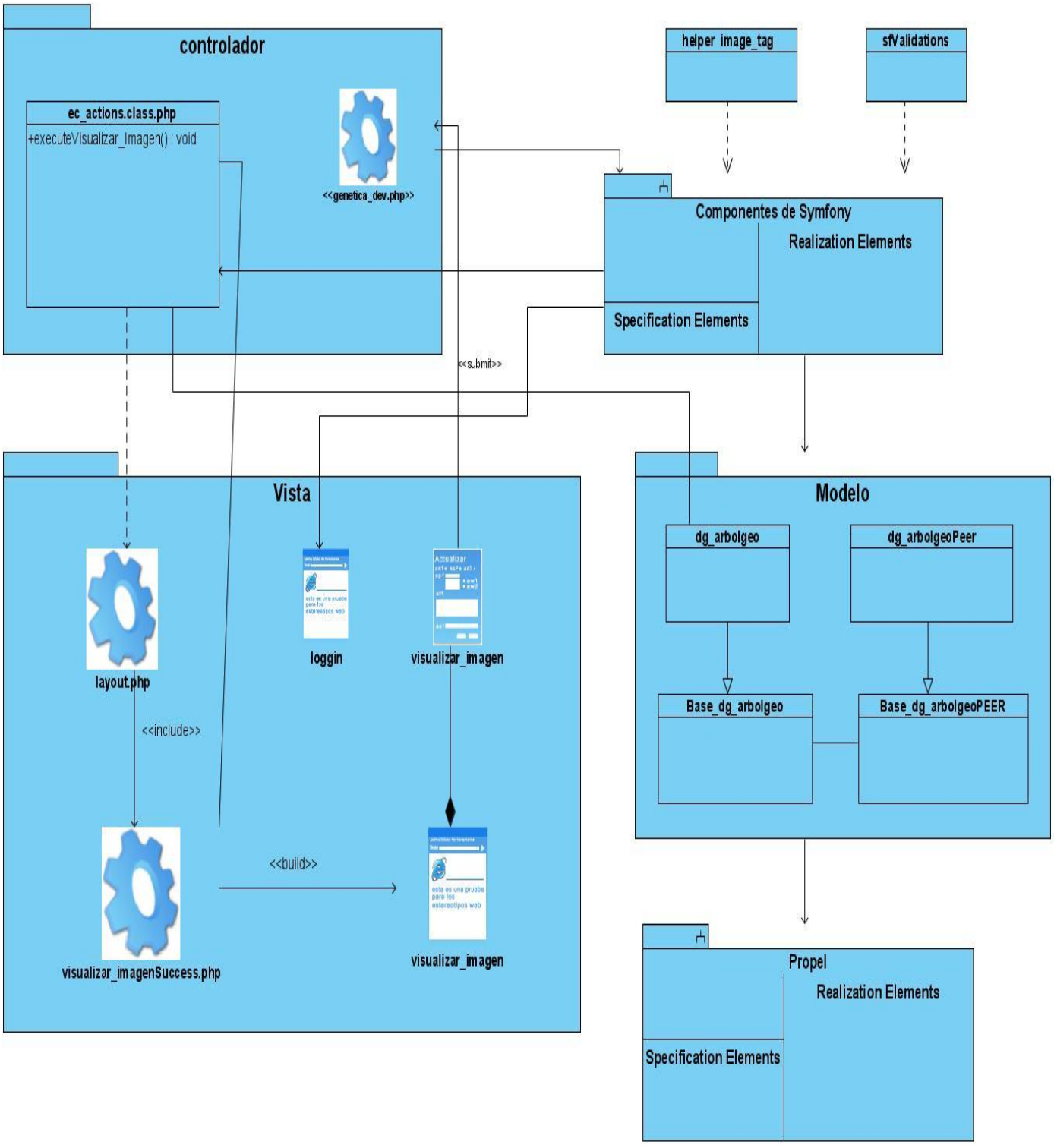
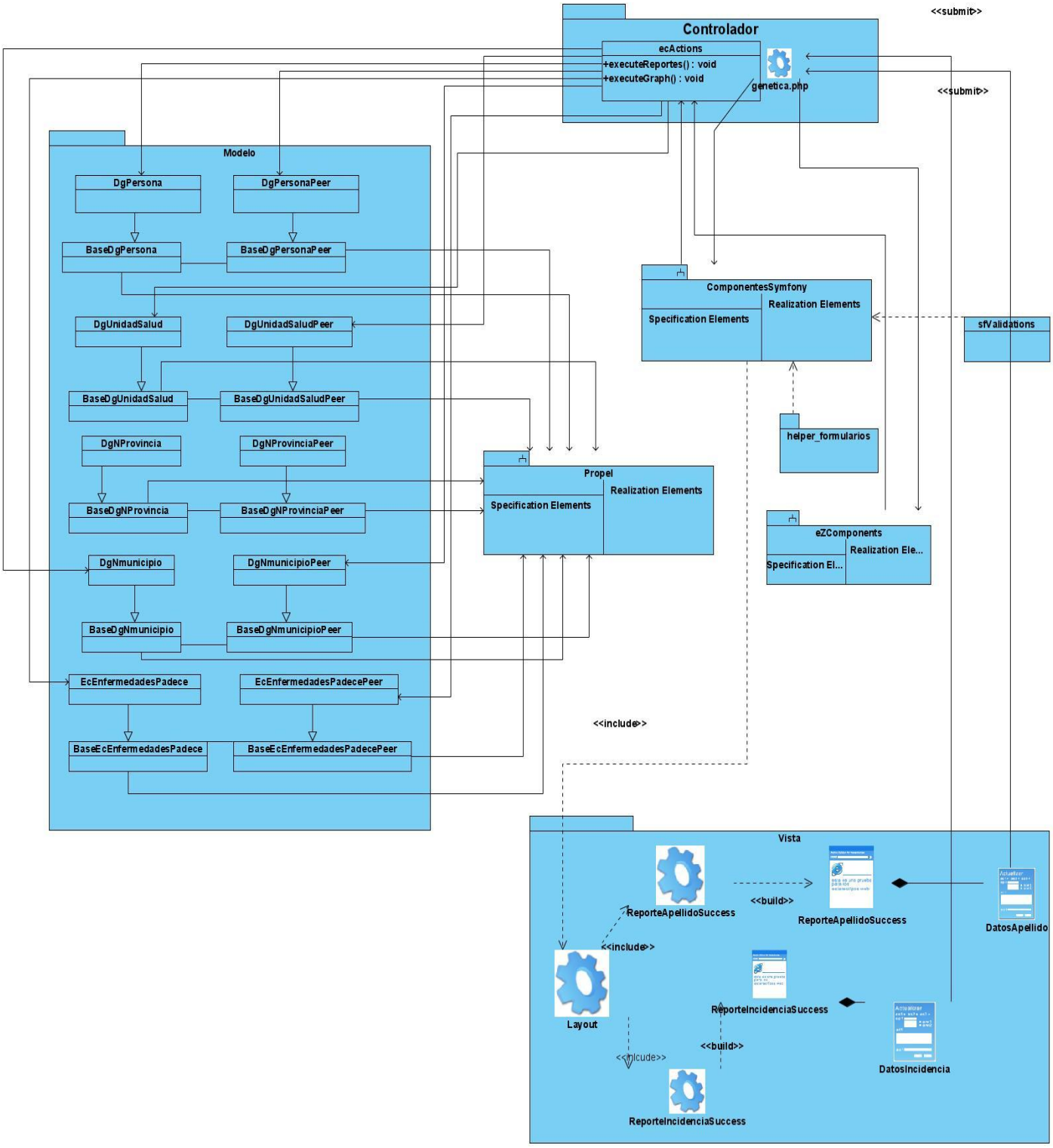


Diagrama de Clases del Diseño. Generar Reportes Estadísticos.



3.5 Diagrama de interacción.

Los diagramas de secuencia reflejan la sucesión temporal de los eventos que se producen en las diferentes funcionalidades.

Diagrama de secuencia. Insertar Datos Complementarios.

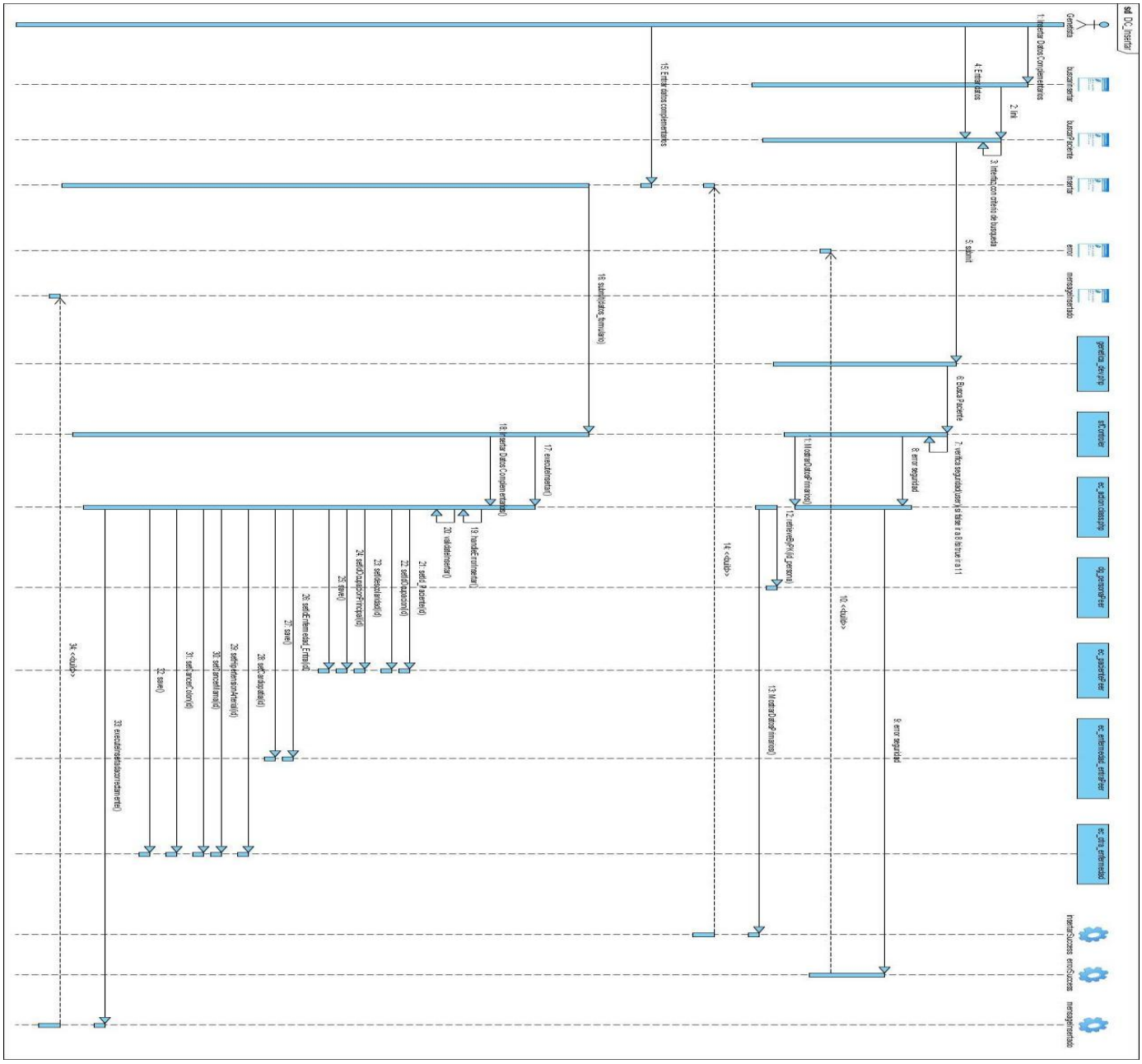


Diagrama de Secuencia. Modificar Datos Complementario.

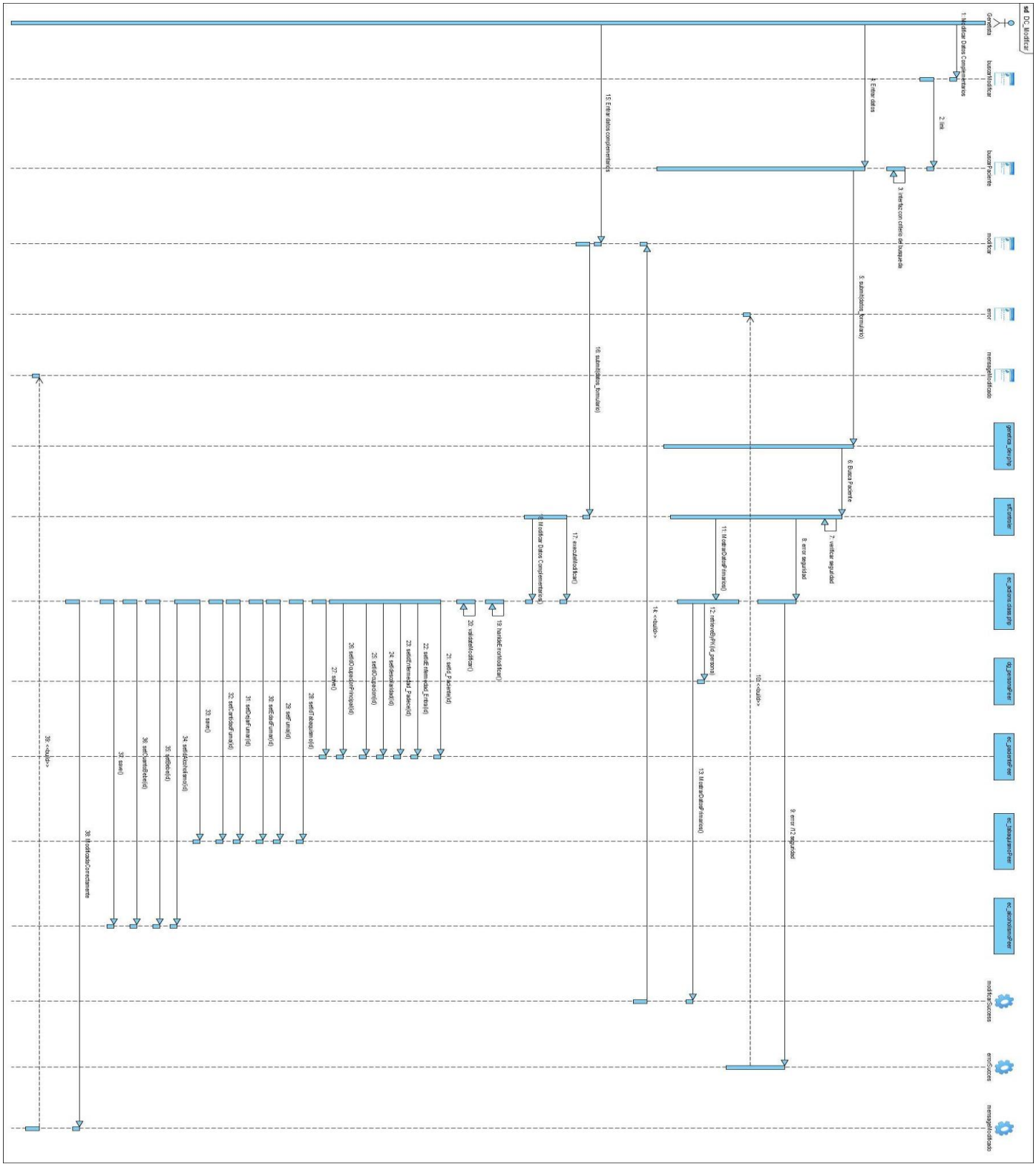


Diagrama de Secuencia. Visualizar Árbol Genealógico.

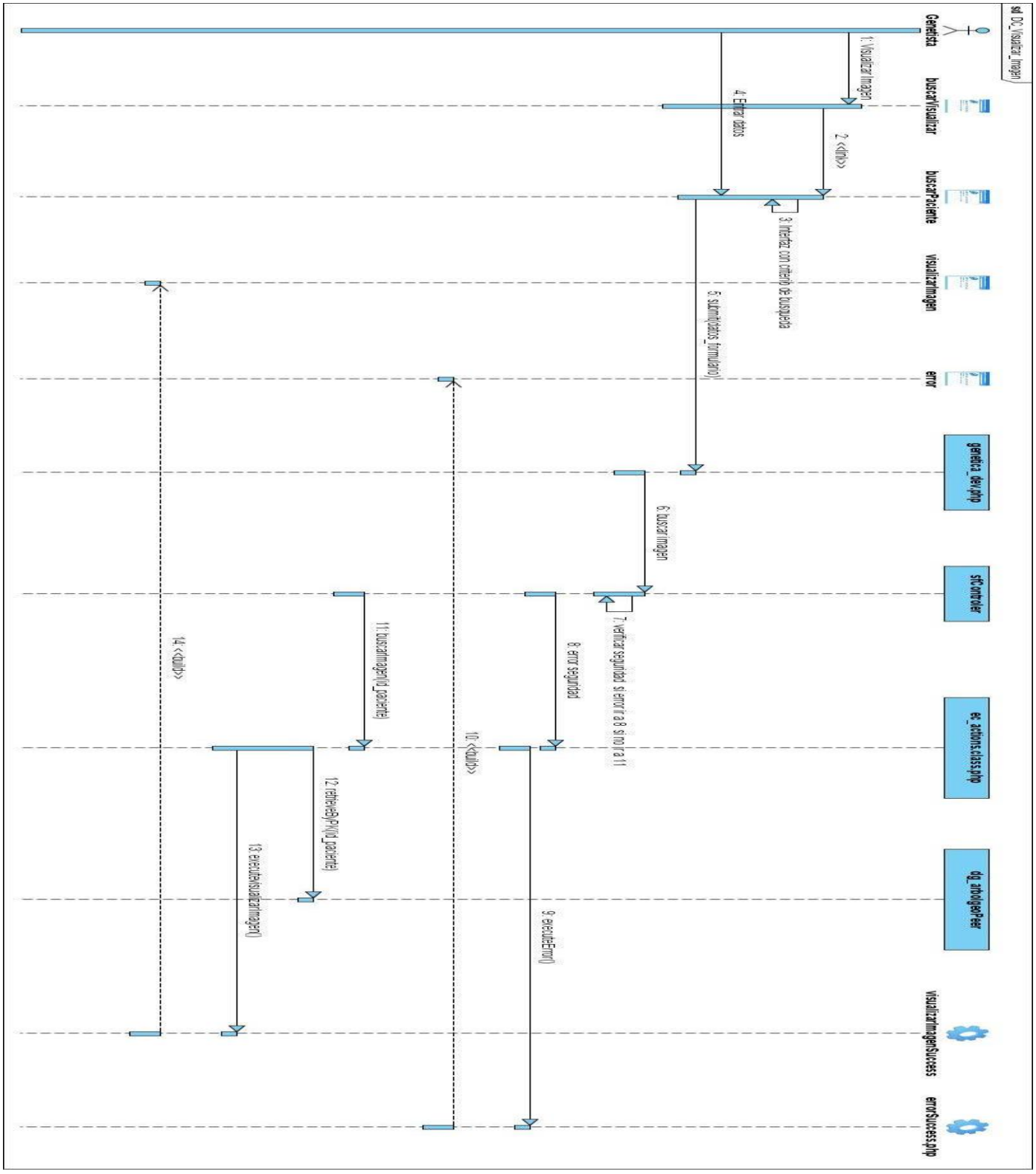
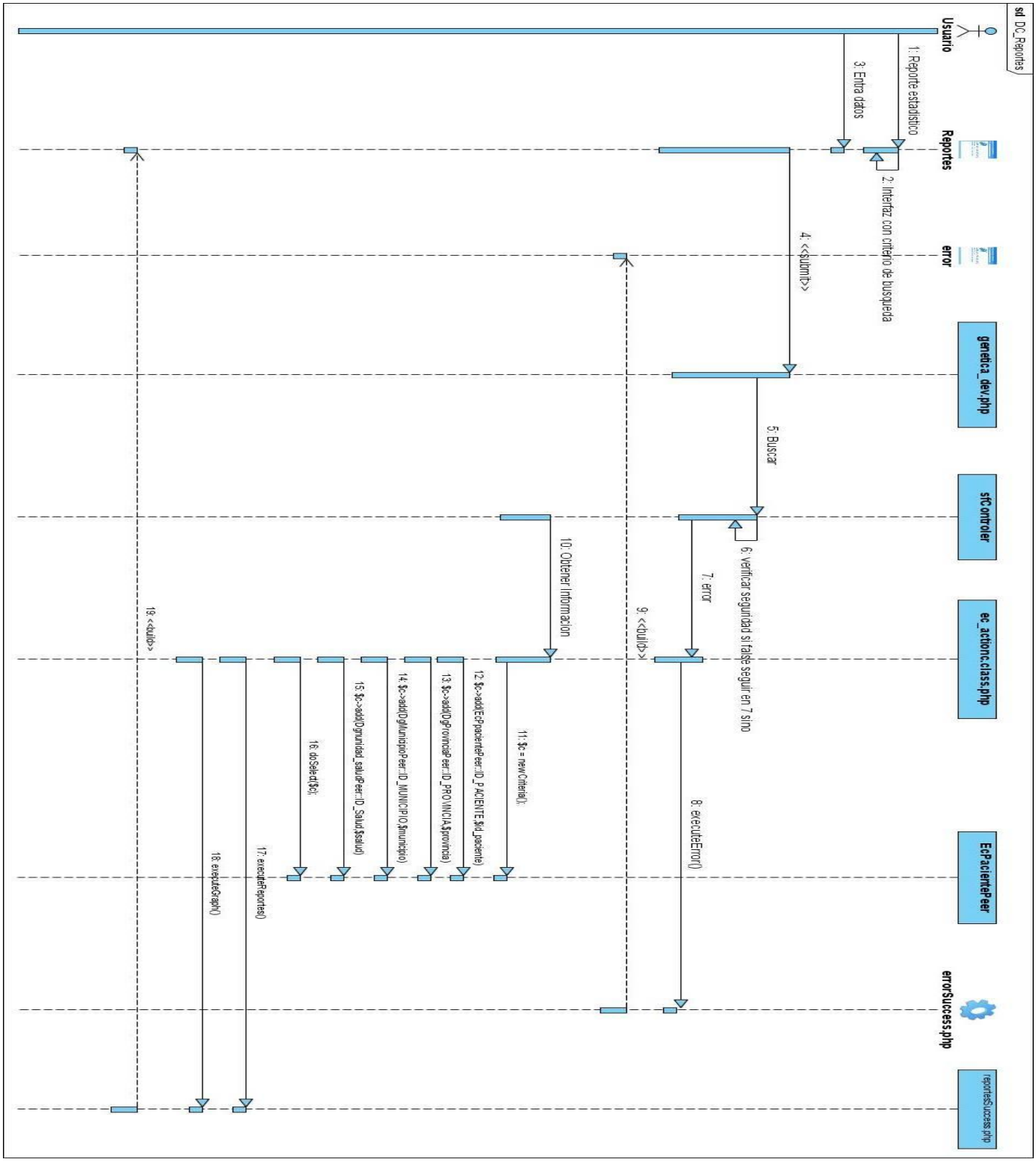


Diagrama de Secuencia. Generar Reportes Estadísticos.



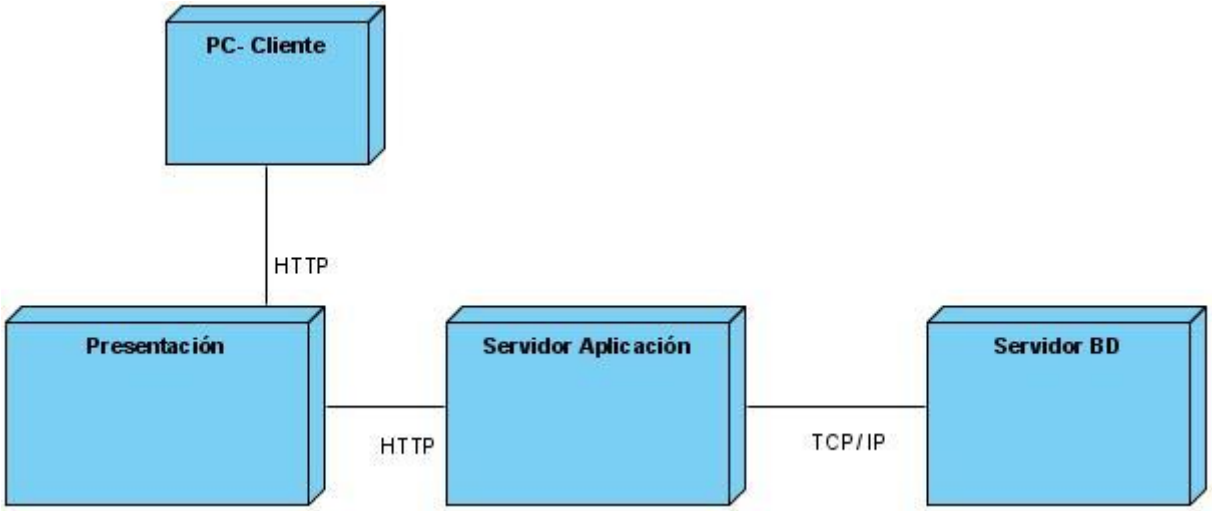
3.6 Diagrama de Despliegue.

El modelo de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los links de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. Consiste en:

- **Nodos:** Elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos
- **Dispositivos**: Nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela
- **Conectores**: Expresa el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

En el siguiente diagrama se representan los diferentes nodos en los que se desplegará el sistema:

Diagrama de Despliegue.



3.7 Diseño de la BD. Modelo de clases persistentes.

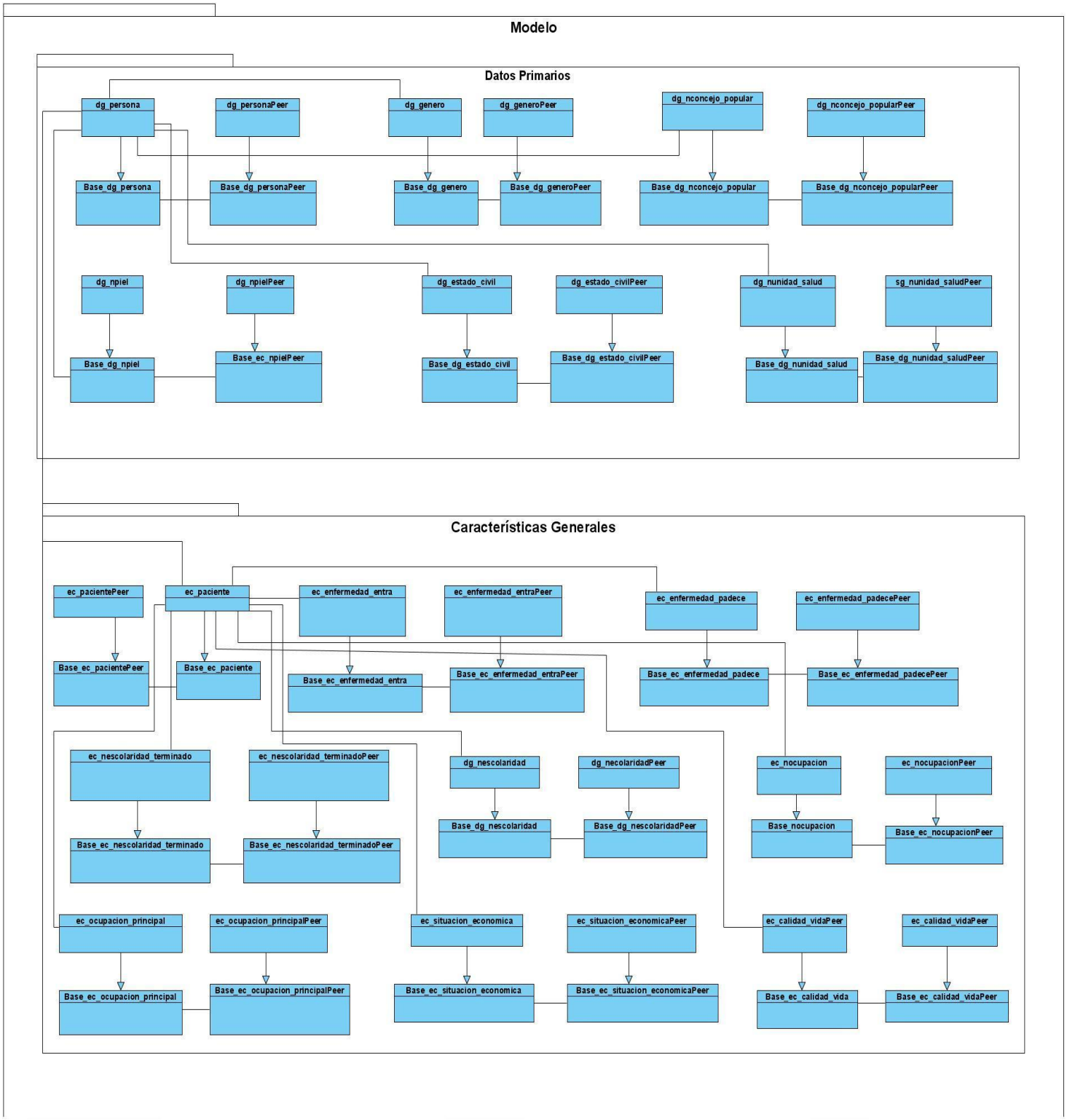
alasMEDIGEN se caracteriza por poseer una base de datos central para los diferentes módulos que lo integran. Por lo general cada uno de estos módulos o registros utilizan un conjunto de tablas que archivan información necesaria solo para ese registro, aunque todos necesitan de un conjunto de datos generales de todos los pacientes que son recogidos en la sección de Datos Primarios por ser datos comunes para todos los pacientes de cualquier registro o alguna que otra información que necesiten de otro módulo.

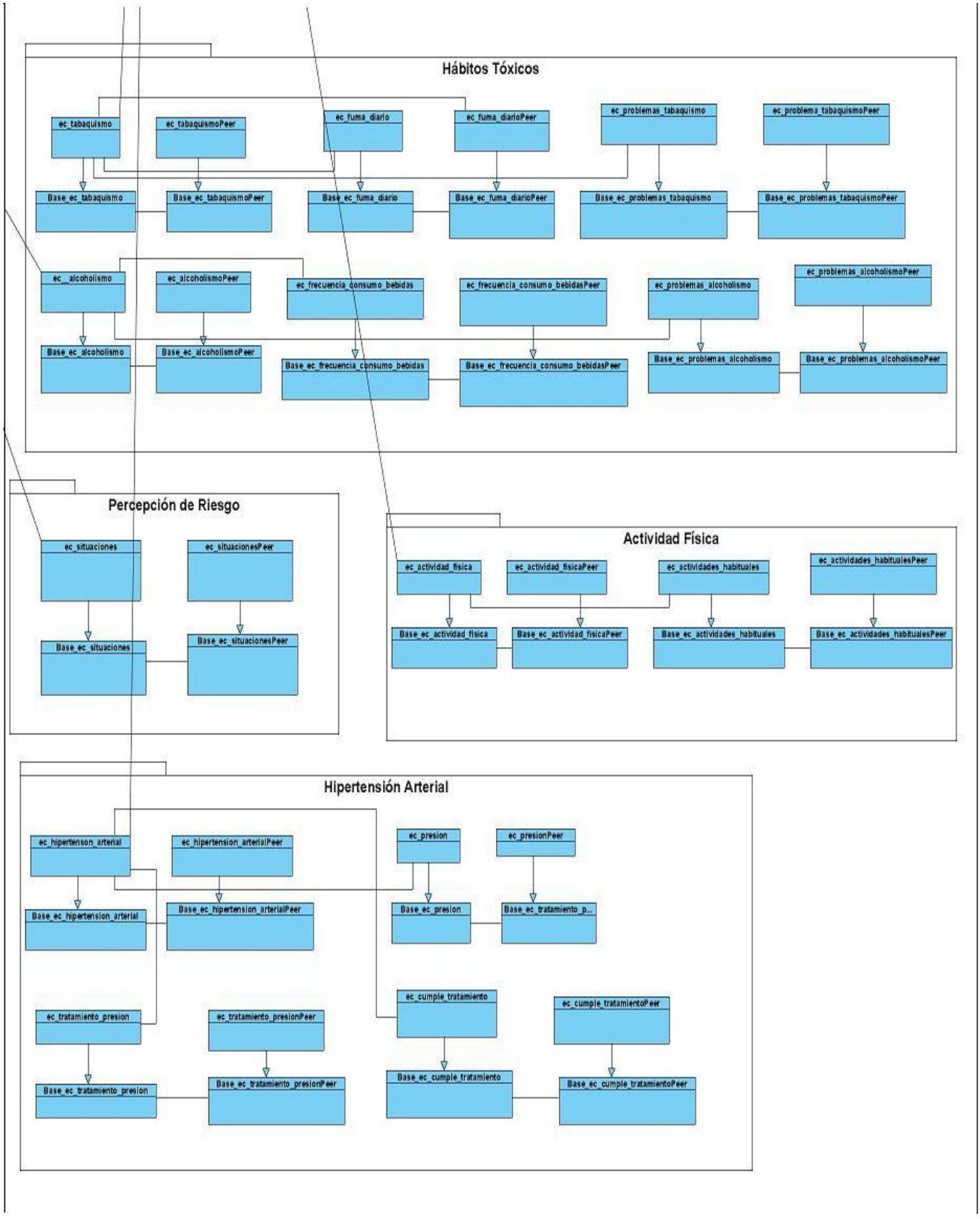
Según el patrón MVC que implementa Symfony las clases persistentes se encuentran en el modelo y por cada clase que exista se crean 3 clases más, generadas por el framework, siendo dos de ellas de acceso a los datos y las otras dos pertenecientes a la capa de atracción de la base de datos, BasenombreClase(), BasenombreClasePeer(), nombreClase(), nombreClasePeer(). Esto se explica de una manera sencilla. Puede ser necesario añadir métodos y propiedades personalizadas en los objetos del modelo (piensa por ejemplo en el método getNombreCompleto ()). También es posible que a medida que el proyecto se esté desarrollando, se añadan tablas o columnas. Además, cada vez que se modifica el archivo schema.yml se deben regenerar las clases del modelo de objetos mediante el comando propel-build-model. Si se añaden los métodos personalizados en las clases que se generan, se borrarían cada vez que se vuelvan a generar esas clases. Las clases con nombre Base del directorio lib/model/om/ son las que se generan directamente a partir del esquema. Nunca se deberían modificar esas clases, porque cada vez que se genera el modelo, se sobrescriben todas las clases antiguas, con las nuevas generadas. Por otra parte, las clases de objetos propias que están en el directorio lib/model heredan de las clases con nombre Base. Estas clases no se modifican cuando se ejecuta la tarea propel-build-model, por lo que son las clases en las que se añaden los métodos propios, métodos que el programador puede adicionar a medida que se necesiten funcionalidades nuevas.

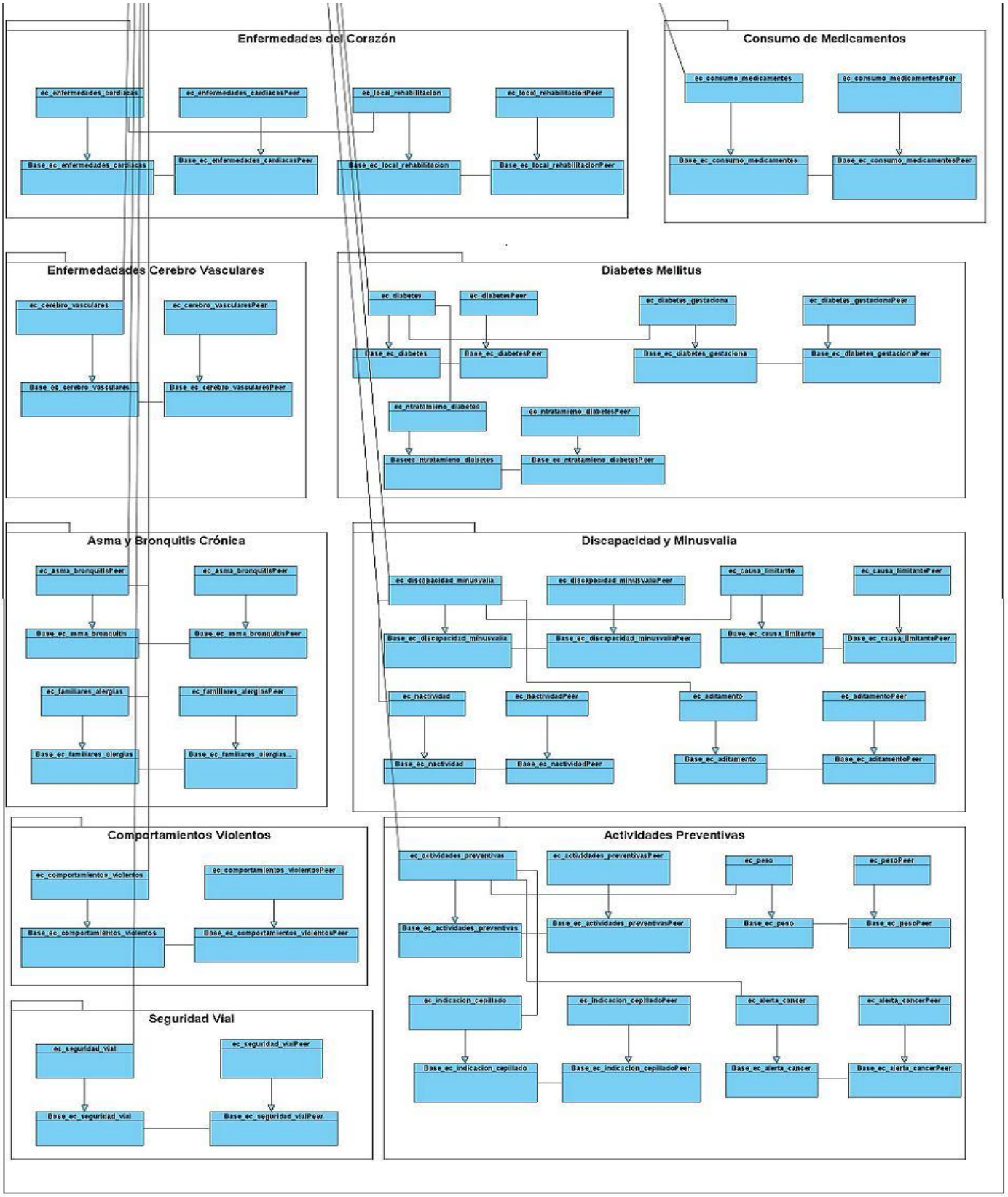
El Registro Genético Preventivo de Familias con Enfermedades Comunes incorpora un cúmulo de nuevas clases a la base de datos del sistema general, permitiendo poder almacenar la información pertinente a los datos complementarios del paciente, un ejemplo de ellas serían EcPaciente, EcTabaquismo, EcNEnfermedadesComunes, EcActividadFisica.

A continuación se hace una representación del modelo para el Registro Genético Preventivo de Familias con Enfermedades Comunes separado en paquetes por las diferentes secciones que presentan los datos:

Paquete de Modelo.


Modelo
Datos Primarios
dg_persona
dg_personaPeer
dg_genero
dg_generoPeer
dg_nconcejo_popular
dg_nconcejo_popularPeer
Base_dg_persona
Base_dg_personaPeer
Base_dg_genero
Base_dg_generoPeer
Base_dg_nconcejo_popular
Base_dg_nconcejo_popularPeer
dg_npiel
dg_npielPeer
dg_estado_civil
dg_estado_civilPeer
dg_nunidad_salud
sg_nunidad_saludPeer
Base_dg_npiel
Base_ec_npielPeer
Base_dg_estado_civil
Base_dg_estado_civilPeer
Base_dg_nunidad_salud
Base_dg_nunidad_saludPeer
Características Generales
ec_pacientePeer
ec_paciente
ec_enfermedad_entra
ec_enfermedad_entraPeer
ec_enfermedad_padece
ec_enfermedad_padecePeer
Base_ec_pacientePeer
Base_ec_paciente
Base_ec_enfermedad_entra
Base_ec_enfermedad_entraPeer
Base_ec_enfermedad_padece
Base_ec_enfermedad_padecePeer
ec_nescolaridad_terminado
ec_nescolaridad_terminadoPeer
dg_nescolaridad
dg_necolaridadPeer
ec_nocupacion
ec_nocupacionPeer
Base_ec_nescolaridad_terminado
Base_ec_nescolaridad_terminadoPeer
Base_dg_nescolaridad
Base_dg_nescolaridadPeer
Base_nocupacion
Base_ec_nocupacionPeer
ec_ocupacion_principal
ec_ocupacion_principalPeer
ec_situacion_economica
ec_situacion_economicaPeer
ec_calidad_vidaPeer
ec_calidad_vidaPeer
Base_ec_ocupacion_principal
Base_ec_ocupacion_principalPeer
Base_ec_situacion_economica
Base_ec_situacion_economicaPeer
Base_ec_calidad_vida
Base_ec_calidad_vidaPeer


Hábitos Tóxicos
ec_tabaquismo
ec_tabaquismoPeer
ec_fuma_diario
ec_fuma_diarioPeer
ec_problemas_tabaquismo
ec_problema_tabaquismoPeer
Base_ec_tabaquismo
Base_ec_tabaquismoPeer
Base_ec_fuma_diario
Base_ec_fuma_diarioPeer
Base_ec_problemas_tabaquismo
Base_ec_problemas_tabaquismoPeer
ec_alcoholismo
ec_alcoholismoPeer
ec_frecuencia_consumo_bebidas
ec_frecuencia_consumo_bebidasPeer
ec_problemas_alcoholismo
ec_problemas_alcoholismoPeer
Base_ec_alcoholismo
Base_ec_alcoholismoPeer
Base_ec_frecuencia_consumo_bebidas
Base_ec_frecuencia_consumo_bebidasPeer
Base_ec_problemas_alcoholismo
Base_ec_problemas_alcoholismoPeer
Percepción de Riesgo
ec_situaciones
ec_situacionesPeer
Base_ec_situaciones
Base_ec_situacionesPeer
Actividad Física
ec_actividad_fisica
ec_actividad_fisicaPeer
ec_actividades_habituales
ec_actividades_habitualesPeer
Base_ec_actividad_fisica
Base_ec_actividad_fisicaPeer
Base_ec_actividades_habituales
Base_ec_actividades_habitualesPeer
Hipertensión Arterial
ec_hipertenson_arterial
ec_hipertension_arterialPeer
ec_presion
ec_presionPeer
Base_ec_hipertension_arterial
Base_ec_hipertension_arterialPeer
Base_ec_presion
Base_ec_tratamiento_p...
ec_tratamiento_presion
ec_tratamiento_presionPeer
ec_cumple_tratamiento
ec_cumple_tratamientoPeer
Base_ec_tratamiento_presion
Base_ec_tratamiento_presionPeer
Base_ec_cumple_tratamiento
Base_ec_cumple_tratamientoPeer


Enfermedades del Corazón
ec_enfermedades_cardiacas
ec_enfermedades_cardiacasPeer
ec_local_rehabilitacion
ec_local_rehabilitacionPeer
Base_ec_enfermedades_cardiacas
Base_ec_enfermedades_cardiacasPeer
Base_ec_local_rehabilitacion
Base_ec_local_rehabilitacionPeer
Consumo de Medicamentos
ec_consumo_medicamentes
ec_consumo_medicamentesPeer
Base_ec_consumo_medicamentes
Base_ec_consumo_medicamentesPeer
Enfermedadades Cerebro Vasculares
ec_cerebro_vasculares
ec_cerebro_vascularesPeer
Base_ec_cerebro_vasculares
Base_ec_cerebro_vascularesPeer
Diabetes Mellitus
ec_diabetes
ec_diabetesPeer
ec_diabetes_gestaciona
ec_diabetes_gestacionaPeer
Base_ec_diabetes
Base_ec_diabetesPeer
Base_ec_diabetes_gestaciona
Base_ec_diabetes_gestacionaPeer
ec_ntratamiento_diabetes
ec_ntratamiento_diabetesPeer
Baseec_ntratamiento_diabetes
Base_ec_ntratamiento_diabetesPeer
Asma y Bronquitis Crónica
ec_asma_bronquitisPeer
ec_asma_bronquitisPeer
Base_ec_asma_bronquitis
Base_ec_asma_bronquitisPeer
ec_familiares_alergias
ec_familiares_alergiasPeer
Base_ec_familiares_alergias
Base_ec_familiares_alergias...
Discapacidad y Minusvalia
ec_discapacidad_minusvalia
ec_discapacidad_minusvaliaPeer
ec_causa_limitante
ec_causa_limitantePeer
Base_ec_discapacidad_minusvalia
Base_ec_discapacidad_minusvaliaPeer
Base_ec_causa_limitante
Base_ec_causa_limitantePeer
ec_nactividad
ec_nactividadPeer
ec_aditamento
ec_aditamentoPeer
Base_ec_nactividad
Base_ec_nactividadPeer
Base_ec_aditamento
Base_ec_aditamentoPeer
Comportamientos Violentos
ec_comportamientos_violentos
ec_comportamientos_violentosPeer
Base_ec_comportamientos_violentos
Base_ec_comportamientos_violentosPeer
Actividades Preventivas
ec_actividades_preventivas
ec_actividades_preventivasPeer
ec_peso
ec_pesoPeer
Base_ec_actividades_preventivas
Base_ec_actividades_preventivasPeer
Base_ec_peso
Base_ec_pesoPeer
ec_indicacion_cepillado
ec_indicacion_cepilladoPeer
ec_alerta_cancer
ec_alerta_cancerPeer
Base_ec_indicacion_cepillado
Base_ec_indicacion_cepilladoPeer
Base_ec_alerta_cancer
Base_ec_alerta_cancerPeer
Seguridad Vial
ec_seguridad_vial
ec_seguridad_vialPeer
Base_ec_seguridad_vial
Base_ec_seguridad_vialPeer

3.8 Seguridad.

El sistema garantiza la seguridad mediante la autenticación de usuarios y la gestión de credenciales, simplificando la creación de secciones restringidas y la gestión de la seguridad de usuario, ya que antes de ser ejecutada cada acción pasa por un filtro especial que verifica si el usuario tiene privilegios de acceder a la misma. El controlador frontal se encarga del manejo de la seguridad, siendo un punto de entrada único para toda la aplicación. Así mismo se apoya en el plugin sfGuardPlugin, diseñado específicamente para manejar todo el proceso de seguridad de la aplicación, proporcionando funciones muy efectivas para la gestión de usuarios, grupos y permisos, es por eso que no se hace necesario para los desarrolladores la implementación de un nuevo módulo de seguridad.

Esos privilegios de usuarios son:

▪ 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 llamado ‘”security.yml” donde se especifican los requerimientos de seguridad que los usuarios deberán satisfacer para cada acción o para todas.

Por ejemplo:

▪ Si el usuario está autenticado y tiene las credenciales apropiadas, entonces la acción se ejecuta.

▪ Si el usuario no está autenticado, es redireccionado a la acción de login.

▪ Si el usuario está autenticado, pero no posee las credenciales apropiadas (es decir, que no están definidas en el security.yml asociado a la página en cuestión), será redirigido a la acción segura por defecto.

Ejemplo de archivo de configuración de seguridad ”security.yml”:

```
all:
  is_secure: on

buscarinsertar:
  credentials: [[ nacEditor_EnfermedadesC, provEditor_EnfermedadesC, munEditor_EnfermedadesC]]

insertar:
  credentials: [[ nacEditor_EnfermedadesC, provEditor_EnfermedadesC, munEditor_EnfermedadesC]]

mensaje:
  credentials: [[ nacEditor_EnfermedadesC, provEditor_EnfermedadesC, munEditor_EnfermedadesC]]

modificada:
  credentials: [[ nacEditor_EnfermedadesC, provEditor_EnfermedadesC, munEditor_EnfermedadesC]]

buscarmodificar:
  credentials: [[ nacEditor_EnfermedadesC, provEditor_EnfermedadesC, munEditor_EnfermedadesC]]

modificar:
  credentials: [[ nacEditor_EnfermedadesC, provEditor_EnfermedadesC, munEditor_EnfermedadesC]]
```

En el ejemplo anterior se define para las acciones “buscarinsertar”, “insertar”, “mensaje”, “modificada“, ”buscarmodificar” y “modificar”, las credenciales que poseen un usuario: “nacEditor_EnfermedadesC”, “provEditor_EnfermedadesC” y “munEditor_EnfermedadesC”. En este ejemplo se tiene para cada acción las mismas credenciales y en caso de no tener alguna de esas credenciales el usuario no tendrá acceso a estas acciones.

3.9 Pautas del Diseño.

Se utilizarán las pautas de diseño definidas para el SIGM.

Los formularios que sean creados deben estar centrados y tener bordes con valores 1

La primera fila del formulario debe contener el nombre de la operación que se pretende realizar con el formulario. El estilo a utilizar para dicha fila es " hOperation".

Por ejemplo si el formulario tuviera la finalidad de buscar un paciente, el encabezado sería el siguiente:

<td class="hOperation">Buscar Paciente </td>

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 alineada a la izquierda. El estilo a utilizar en la columna izquierda es “nombrecampo”, el de la derecha es “valorcampo” y el de los componentes “entradaplana”.

Por ejemplo, se le solicita al usuario su nombre.

```
<tr>
  <td align="right" class="nombrecampo">Nombre</td>
     <td  class="valorcampo">
        <?php echo input_tag(‘nombre’ , ’ ’ , ’ class=”entradaplana” ’)?>
     </td>
  </tr>
```

Los botones para efectuar operaciones sobre el formulario se ubicaran en la parte inferior derecha del formulario y utilizarán el estilo “sbttn”. Se posicionarán de izquierda a derecha teniendo en cuenta el peso de la operación que representan.

Ejemplo un formulario que los botones de las operaciones son enviar y cancelar.

```
<tr align="right">
    <td>
       <!--  Botón enviar se ubica más a la derecha que el botón cancelar -->
       <?php echo submit_tag('enviar','class=sbttn')?>
       <?php button_to('cancelar',' ','class=sbttn')?>
    </td>
</tr>
```

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 con el estilo “entradaobligatoria”

Por ejemplo el campo nombre debe ser obligatorio

```
<tr>
   <td class="nombrecampo" width="40%" height="20" align="right">nombre</td>
   <td  class="valorcampo">
      <?php echo input_tag(‘nombre’ , ’ ’ , ’ class=”entradaplana” ’)?>
       <span class="entradaobligatoria"> *</span>
```

```
</td>
</tr>
```

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.

3.10 Prototipo de Interfaz.

Para el SIGM se elaboró un prototipo de interfaz, planificando un diseño orientado a las necesidades de sus futuros usuarios y al ambiente en el que será desplegado. Se ha tenido en cuenta el orden de los elementos que aparecen en el instrumento, acorde al orden de prioridad de los elementos que lo componen.

Descripción de los elementos que componen las interfaces del prototipo diseñado para el SIGM.

Sección 1:

Menú de Operaciones: presenta todas las operaciones correspondientes a cada modulo clasificadas en categorías.

Sección 2:

Área de visualización de datos: presenta la información correspondiente a cada operación invocada en el Menú de Operaciones.

3.11 Validación del sistema.

El sistema constará con diferentes tipos de validaciones con el fin de facilitar la entrada de datos, la legibilidad de los campos, y la integridad de la información que se gestione a través de los diferentes formularios.

El framework Symfony brinda en sus librerías un conjunto de componentes de gran importancia para el desarrollo de aplicaciones web de gran envergadura.

En el tema de la validación de formularios cuenta con archivos de configuración “yml “, parecidos a los que vimos en el subcapítulo de seguridad (security.yml).

Ejemplo de “yml” de validación:

```
fields:
 amparo:
    required:
    msg: Campo Obligatorio
 ingresot:
    required:
     msg: Campo Obligatorio
    sfNumberValidator:
     nan_error: Por favor, introduce un n&uacute;mero entero
     min: 0
     min_error: El valor debe ser como mínimo 0
```

La etiqueta “fields” significa los campos que se van a validar: “amparo” e “ingresot”. El primer campo requiere entrada obligatoria y el segundo entrada obligatoria y usa la clase sfNumberValidator para acotar que el valor entrado sea entero y mayor que cero. Esta es una manera de hacer una validación rápida y efectiva. También se usaron otras clases que contribuyeron a restringir la entrada

de datos, como fue la clase sfStringValidator, que es utilizada para controlar la entrada de cadenas de texto válidos.

También de la misma manera que cada acción tiene un template por defecto y una función "handleError" por la cual se filtran los datos del formulario antes de ser enviados a la acción y procesados hacia el modelo, existe también una función "valídate", en la cual se restringe la entrada de datos erróneos en el template que le corresponde a esa función.

Por ejemplo:

La acción executeNombreAccion(), le corresponde el template nombreAccionSuccess.php y las funciones handleErrorNombreAccion() y validateNombreAccion(). Así ante un envió de datos desde el formulario del template, estos pasan por la función handleErrorNombreAccion() , que se encarga de filtrar los datos validándolos según el archivo nombreAccion.yml que le corresponde según lo visto anteriormente y luego pasa por la función validateNombreAccion() llevándose a cabo otro tipo de validaciones definidas por los analistas o programadores y luego enviando los datos hacia la acción executeNombreAccion() para que sean procesados los datos y enviados mediante el modelo a la base de datos, en caso de detectarse algún error en las validaciones se recibirá el template nuevamente con los mensajes de error respectivos a cada campo erróneo capturados en el helper <?php echo form_error('nombrecampo')?>.

En el sistema se utilizan también validaciones en javascript con el objetivo de validar las diferentes variantes que puede tener el formulario dependiendo de la complejidad de las mismas, tales son los casos de deshabilitar y habilitar o mostrar y esconder determinados campos teniendo en cuenta las diferentes opciones del formulario, con este framework esto se hace implementando las funciones javascript en el directorio del proyecto "web\js" y usándolas en cada formulario independientemente.

Ejemplo:

Se crea el archivo de validación javascript "validacion" en el directorio "web\js" del proyecto y luego se incluye este archivo en el template que requiera de las funciones que contenga este archivo con el helper de Symfony "<?php use_helper('validacion')?>".

3.12 Conclusiones.

Mediante la modelación del diseño del sistema se describe la estructura que deriva la implementación. Se especificaron los patrones de arquitectura y de diseño implementados en el sistema con el uso de las características que ofrece el framework de desarrollo Symfony y se obtuvieron los diagramas de clases del diseño, diagramas de secuencia, así como el prototipo de interfaz y la distribución del sistema mediante el modelo de despliegue. Se explicaron los mecanismos de seguridad desarrollados y los métodos de validación utilizados. Quedó conformado el diseño del sistema como base fundamental para la implementación.

CAPÍTULO 4: IMPLEMENTACIÓN.

En este capítulo se muestran las características de la implementación del sistema en términos de componentes, manteniendo como objetivo principal desarrollar la arquitectura acorde al diseño realizado. Se analizan y aplican los estilos de codificación definidos para alasMEDIGEN, se muestra el código fuente de las clases principales así como interfaces de la aplicación. Se incluye la validación a nivel de desarrollador.

4.1 Diagrama de Componentes.

Diagrama de Componente. Insertar Datos Complementarios.

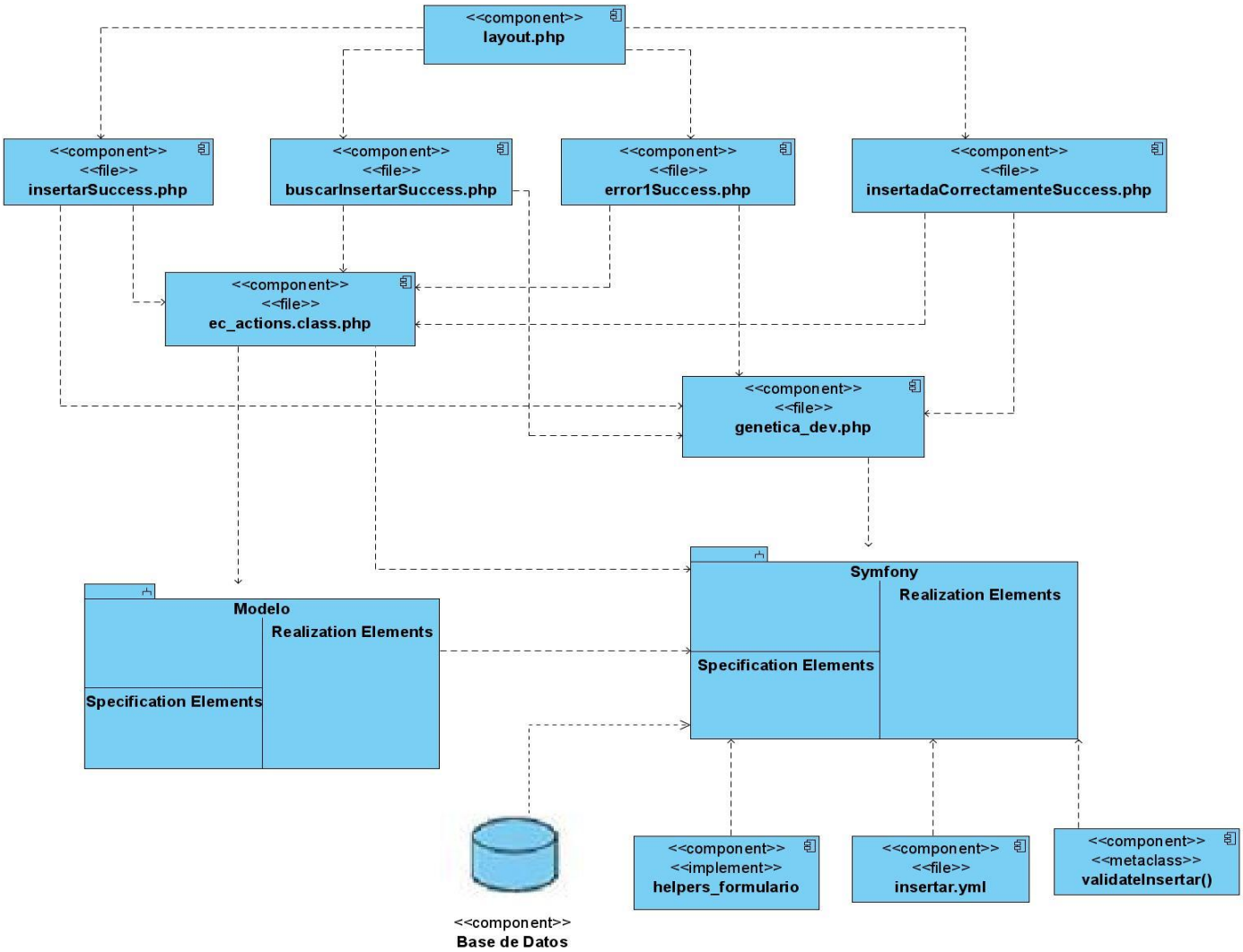


Diagrama de Componente. Modificar Datos Complementarios.

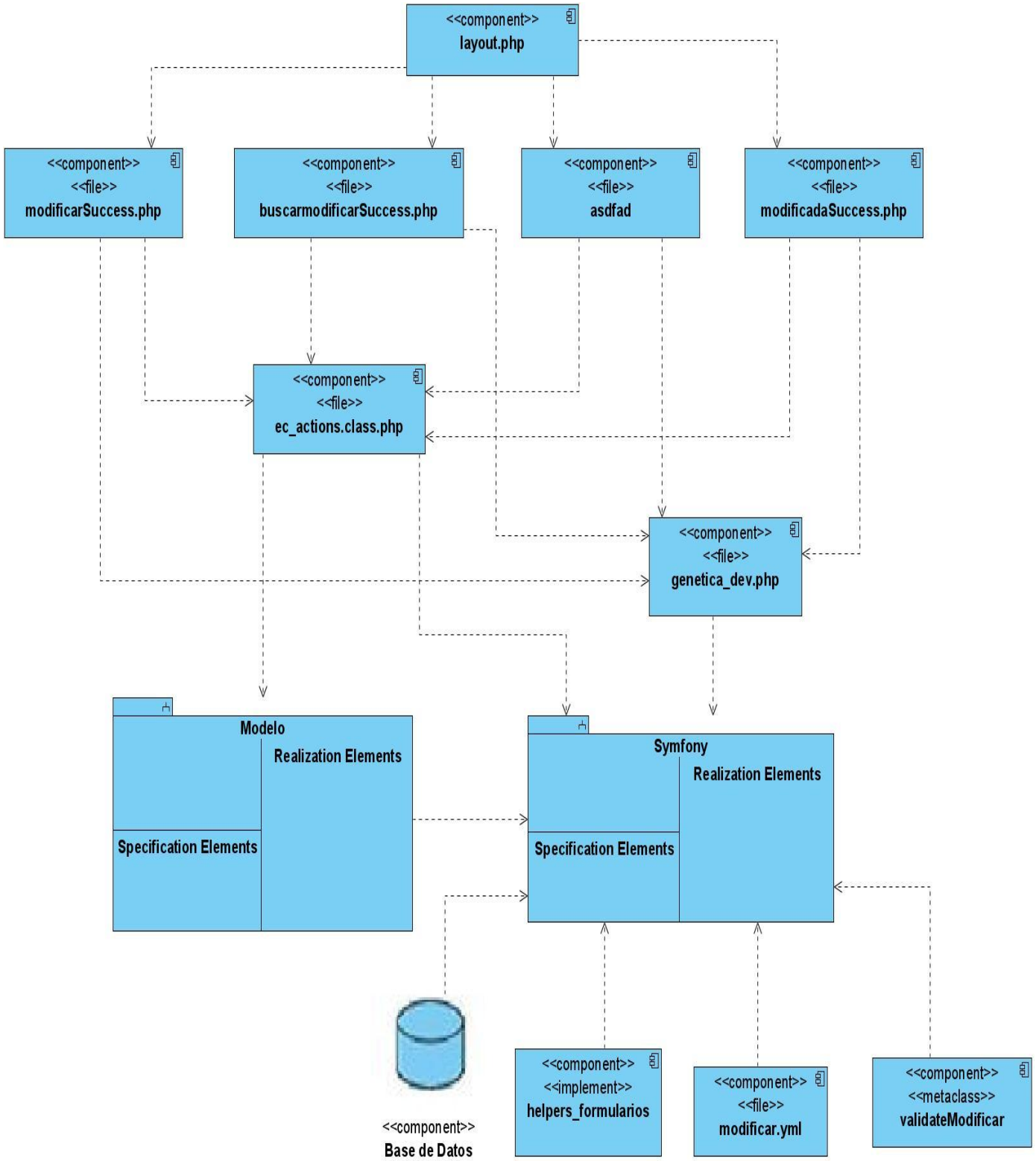


Diagrama de Componente. Visualizar Árbol Genealógico.

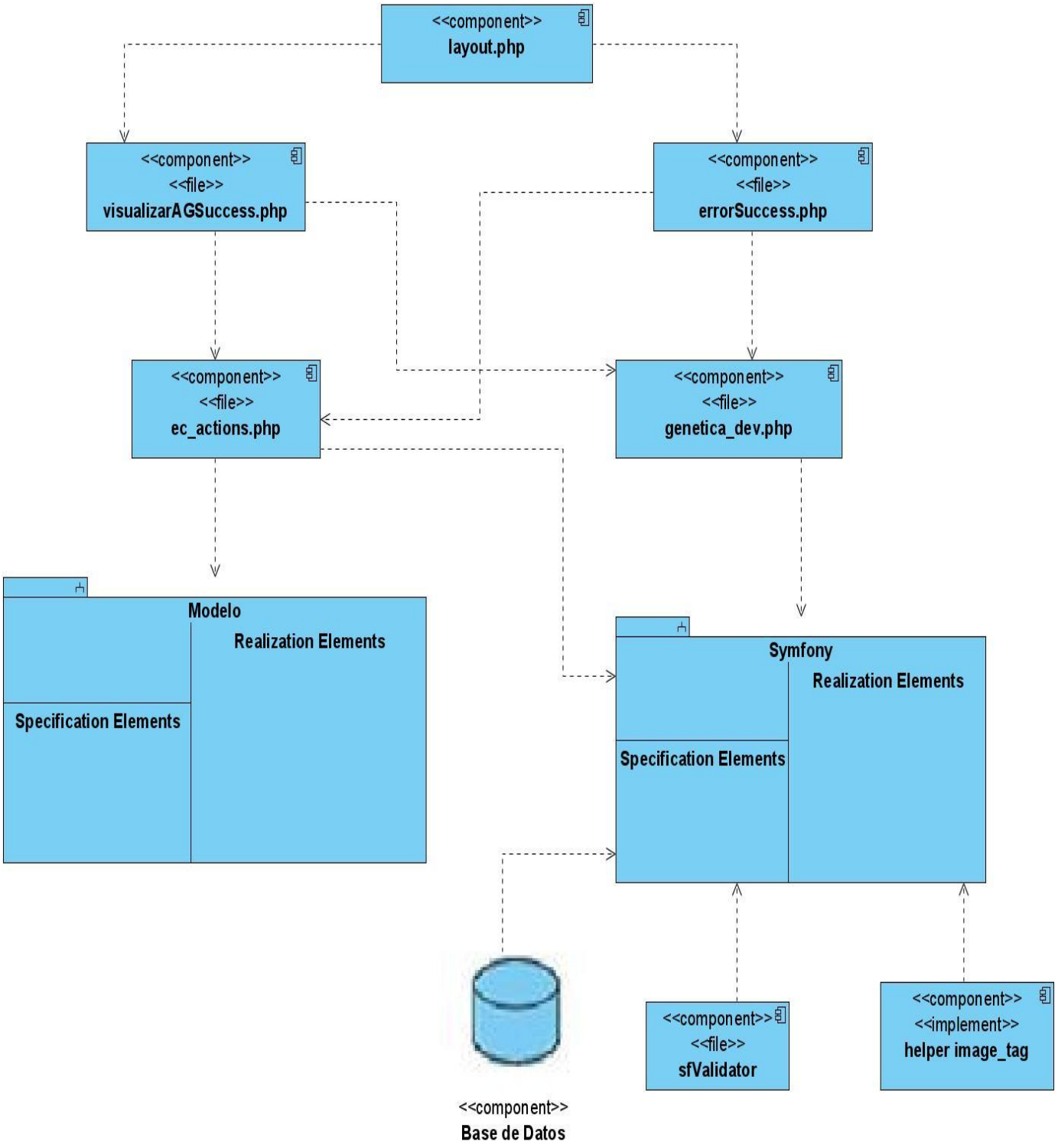
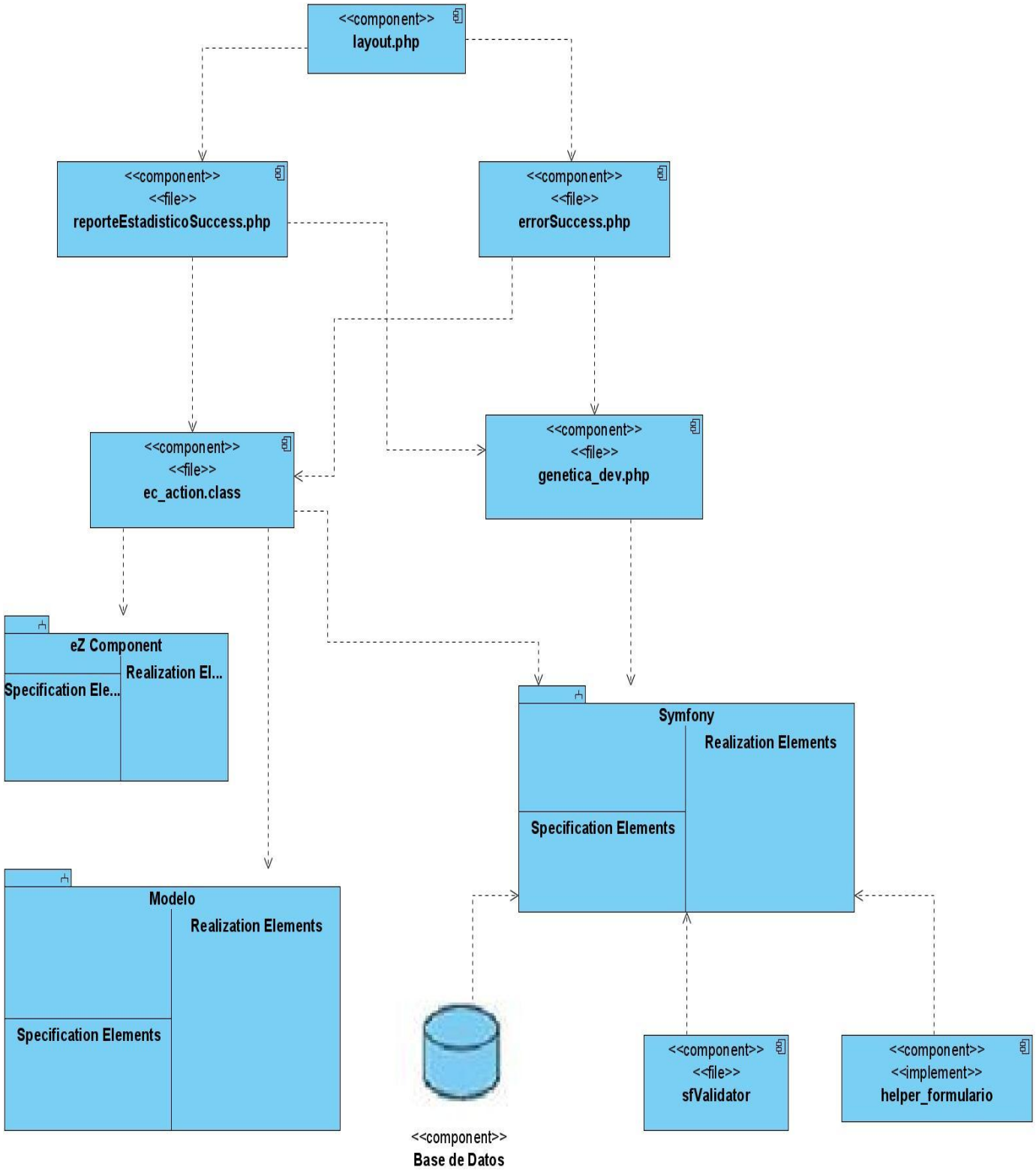


Diagrama de Componente. Generar Reportes Estadísticos.



4.2 Estilos de codificación definidos para alasMEDIGEN.

Los estilos de codificación son un aspecto de gran importancia aunque algunos programadores no les prestan mucha atención. Siempre es necesario definir estilos o estándares de codificación para el desarrollo de cualquier sistema con el objetivo de hacer que el código fuente sea más legible y así propiciar un mayor entendimiento de la programación realizada y una mejor revisión.

Para la codificación dentro de la implementación de los componentes, se definen los siguientes estilos:

Todas las etiquetas php deben ser completas (<?php ?>)... no reducidas (<? ?>).

Todas las variables deberían ser inicializadas o, al menos, comprobada su existencia utilizando isset() antes de ser utilizadas.

El sangrado del texto debe ser siempre de 4 espacios. No utilizar los tabulador (todos los editores no interpretan el tab de la misma manera).

Los nombres de las variables y funciones tienen que ser siempre fáciles de leer, procurando que sean palabras en minúsculas con significado claro análogo a la información que almacenan o la acción que realizan. Si realmente se necesita más de una palabra, deben ponerse juntas, poniendo la inicial de cada palabra en mayúscula siempre que no sea la primera, procurando mantenerlas tan breves como sea posible. Utilizar nombres en plural para arreglos o matrices de objetos.

Ejemplos:

Etiquetas php:

```
<?php echo input_tag('nombre',' ',' class = "entradaplana"')?>
<?php echo radiobutton_tag('sexo', '1' ,false , ' class = "entradaplana"')?>
<?php echo label_for('nombre','Nombre: ',' class = "nombrecampo"')?>
<?php echo checkbox_tag(' diabetico ',' si ',' class = "nombrecampo"')?>
```

Variables:

```
$nombre;
$edad;
$nombreCompleto;
$fechaNacimiento;
$arrayPersonas;
```

Los bloques de código siempre deben estar encerrados por llaves (incluso si solo constan de una línea).

Las cadenas tienen que ser definidas utilizando comillas simples siempre que sea posible, para obtener un mejor rendimiento.

Todas las funciones y clases deben tener comentarios. 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.

Ejemplo:

```
/*  El objetivo de esta función es……
        *nombre del autor
        *paramtro1  es la edad de la persona
        *parámetro2 es el sexo de la persona
*/
public function cargarDatos($edad,$sexo)
{
/*  bloque de instrucciones……
}

/*  El objetivo de esta función es …….
        *nombre del autor
        *paramtro1  es la evaluación del estudiante
        *parámetro2 es el promedio del estudiante
*/
public function Aval($eval,$prom)
{
/*  bloque de instrucciones……
}
```

4.3 Código fuente de las principales clases.

Se presenta el código fuente de las principales clases que soportan la información necesaria del Registro Genético Preventivo de Familias con Enfermedades Comunes.

La clase ecActions es una clase controladora puesto que contiene todas las funcionalidades necesarias para manipular la información gestionada en cada una de los templates donde se ingresan los datos o se muestran los diferentes reportes.

Clase ecActions

```
class ecActions extends sfActions
{
    * Executes index action

  public function executeEjemplo()

  private function MostrarDatosPrimarios()
  public function executeIndex()

  public function executeInsertar()
  public function handleErrorInsertar()

  private function CargarDatosComplementarios()
  public function executeBuscarInsertar()
  public function executeBuscarModificar()

  public function executeModificar()
  public function  handleErrorModificar()

  private  function privilegiosInsert()
  private function VerificarPrivilegios()

  public function handleErrorInsertarPaciente()
  private function InsertarDatosEC()

  public function validateInsertar()

  public function executeReporteIncidencia()

  public function executeMunicipios()

  public function executeGraph()
}
```

4.4 Librerías y Plugins Gráficos usados en los reportes.

Una tarea ardua para cualquier programador web es el uso de esquemas o gráficas para representar diferentes situaciones con el fin de propiciar una interfaz más amigable para el usuario. Para dar solución a este problema se han creado una gran variedad de librerías y plugins, los cuales están a disposición de cualquier usuario. Entre estos componentes podemos mencionar las librerías Open Flash Chart, jpGraph , pChart y los plugins especialmente creados para toda la comunidad de desarrolladores de Symfony , dwJpgraphPlugin, sfCsvPlugin, dbFusionChartPlugin, todos ellos son de marcada y probada calidad pero sobre todos ellos destaca por su alcance y nivel de profesionalidad la librería eZ Components , desarrollada por el grupo eZ Systems.

eZ Components.

Es una biblioteca de componentes independientes de alta calidad que ayudan en el desarrollo de aplicaciones centradas en la Web, ofreciendo un conjunto de componentes independientes, fáciles de usar y diseñados para reducir errores y ahorrar el tiempo de desarrollo, precisamente esa independencia le permite al desarrollador escoger explícitamente la parte de la librería a utilizar, así mismo es muy flexible y no dicta como organizar la aplicación en desarrollo, además todos los componentes de la API se han diseñado de manera similar, estableciéndose una alta coherencia entre los mismos. Otro aspecto importante a su favor, es que posee una amplia y basta documentación, disponible en su sitio web: http://ezcomponents.org/docs , dando la posibilidad así a todos los desarrolladores de aprovechar al máximo la librería. Igualmente otra ventaja de su uso es que posee una licencia abierta, por lo cual goza de amplia popularidad, existiendo una gran cantidad de contribuyentes para el mejoramiento de la biblioteca, dándose mantenimiento periódicamente a la misma por parte del equipo de desarrolladores de eZ Components.

Gráficas con Ez Components

La biblioteca Ez Components, incluye entre sus tantos componentes, uno dedicado especialmente para el manejo, edición y creación de gráficos, llamado Graph. El mismo controla toda la visualización de información estática, generando gráficos de barras, tartas y líneas, con vistas en 2d y 3d, con detalles y estilos altamente profesionales, capaces de brindar de manera exacta información a través de imágenes (ver Anexo 6). Una de las principales virtudes de este componente es que el formato de salida de los gráficos es de extensión SVG. Los archivos SVG tienen todas las ventajas

asociadas a un formato vectorial, su tamaño es muy compacto, el texto que incluyen es editable, lo que supone una ventaja sobre los formatos GIF o JPG, pues puede ser incluso indexado por los buscadores web. Soporta hojas de estilos (CSS). Admite efectos como sonidos, efectos visuales al hacer click o mover el mouse. Esta escrito completamente en XML y brinda una gran compatibilidad con los navegadores.

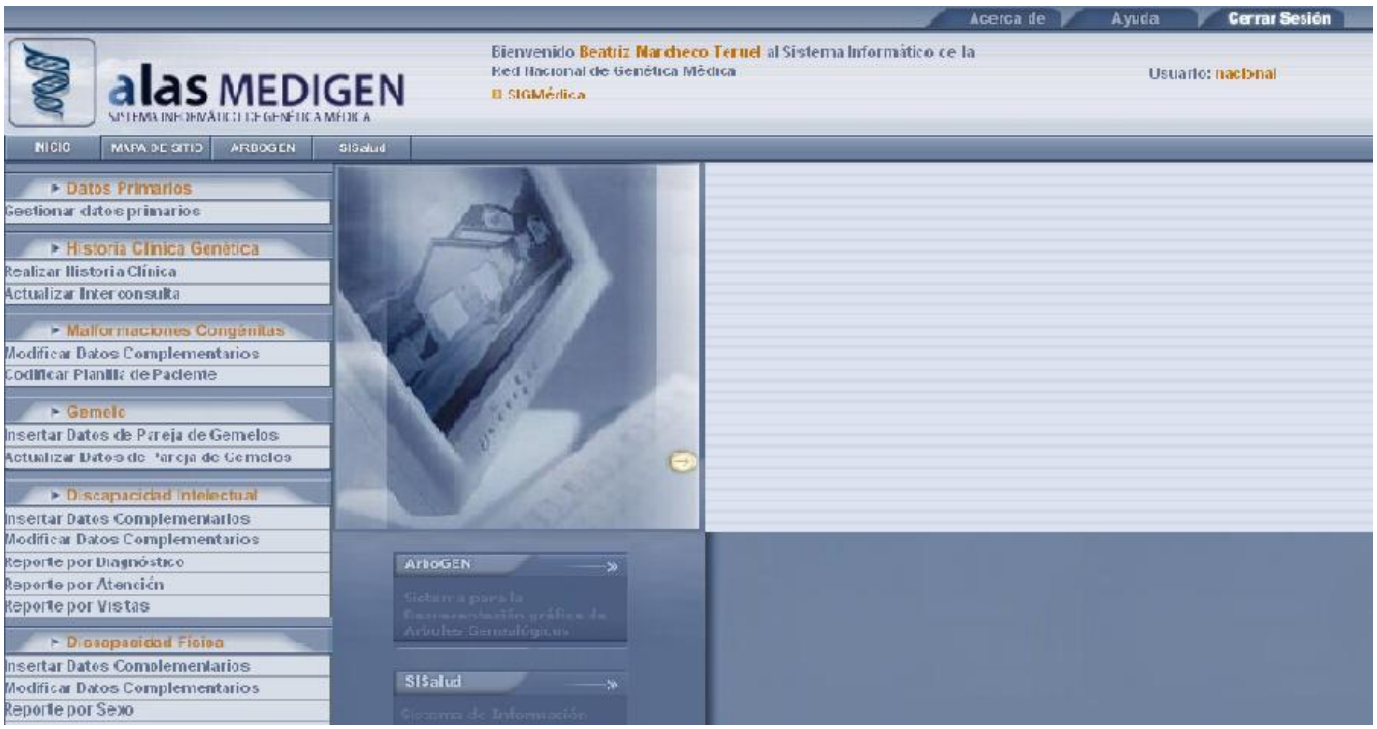
eZ Components y Symfony

Esta librería está diseñada para aplicaciones desarrolladas en PHP, contando con una perfecta compatibilidad entre ellas, así mismo lo hace con los frameworks existentes para este lenguaje, Zend Framework, CakePHP y Symfony. Por todas las características previamente explicadas se decide el uso de esta potente librería para el diseño e implementación de las funcionalidades necesarias del sistema.

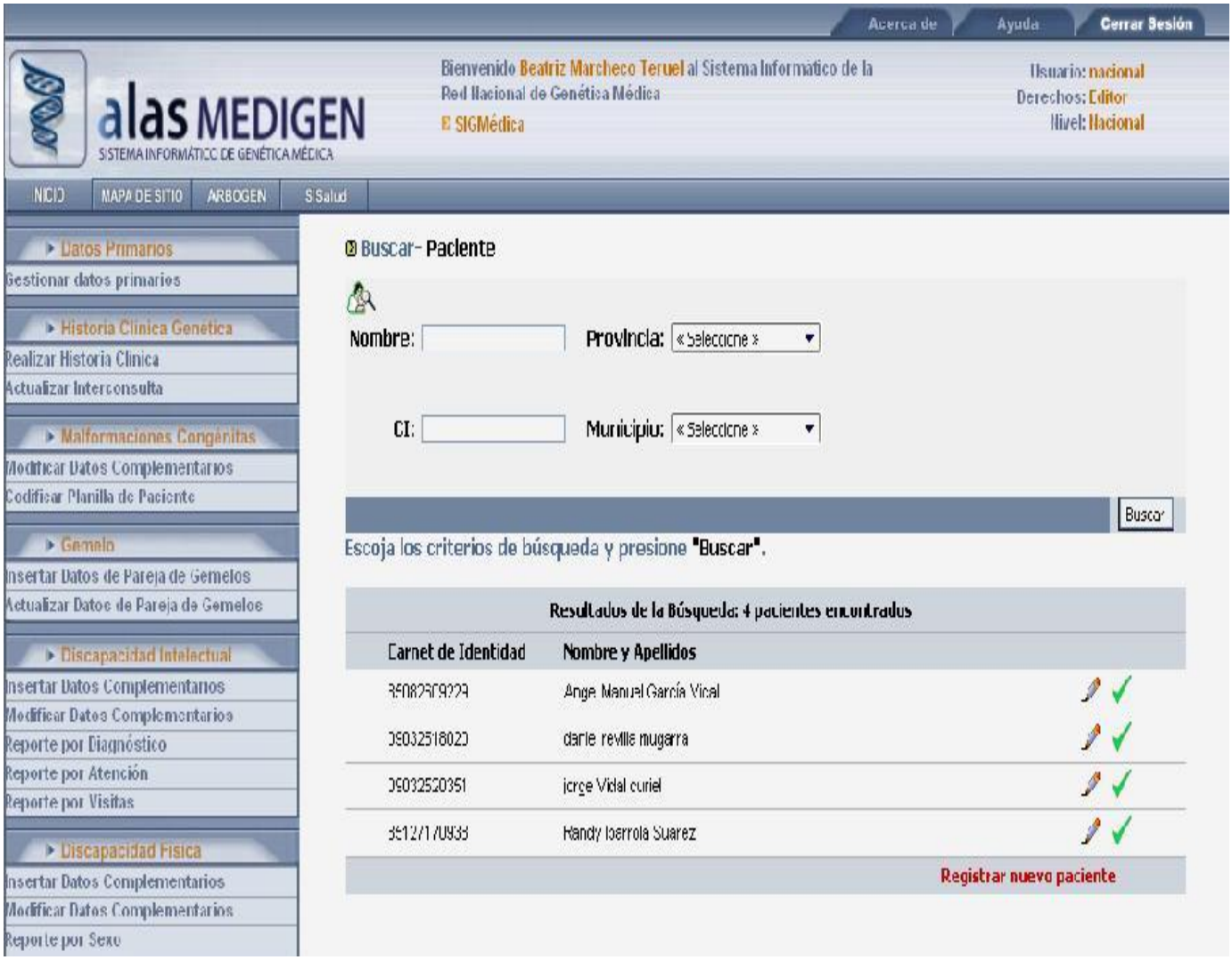
4.5 Interfaces de la aplicación (Ver Anexo 7).

Interfaz de entrada al sistema.

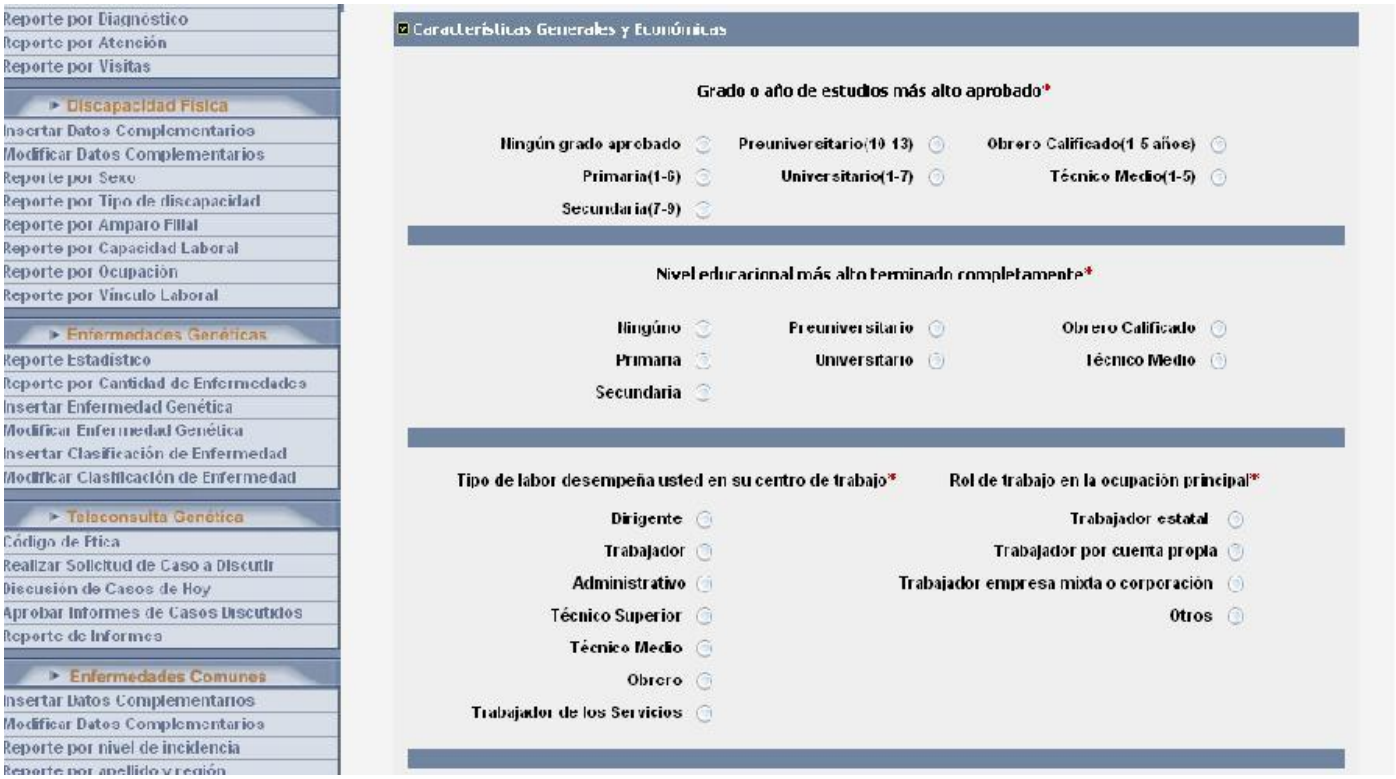
Interfaz principal.



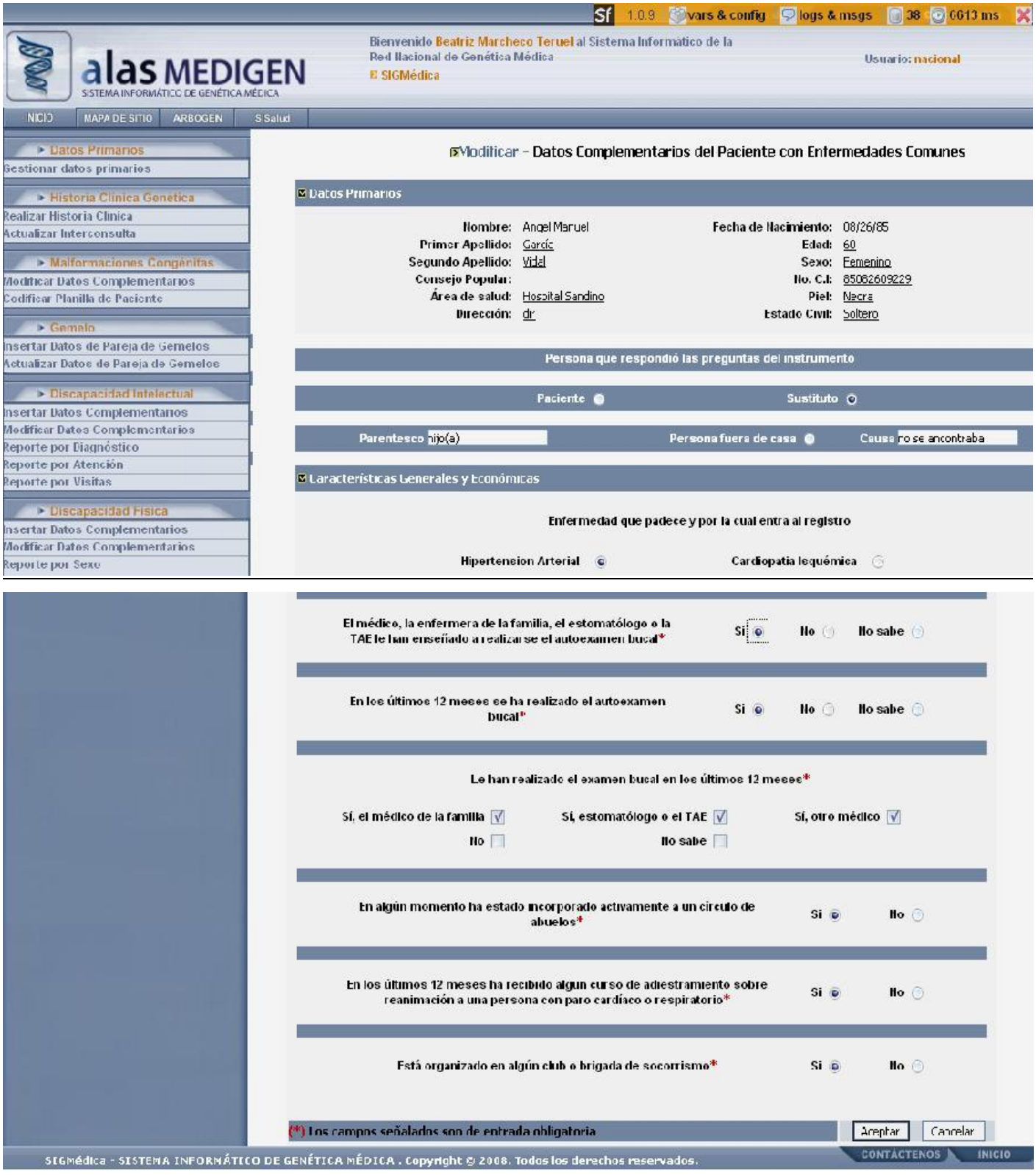
Interfaz de búsqueda. Datos Primarios.



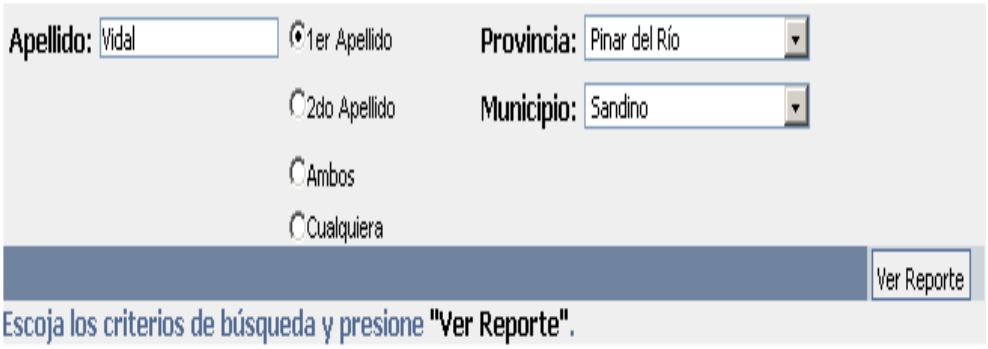
Interfaz para Insertar Datos Complementarios.

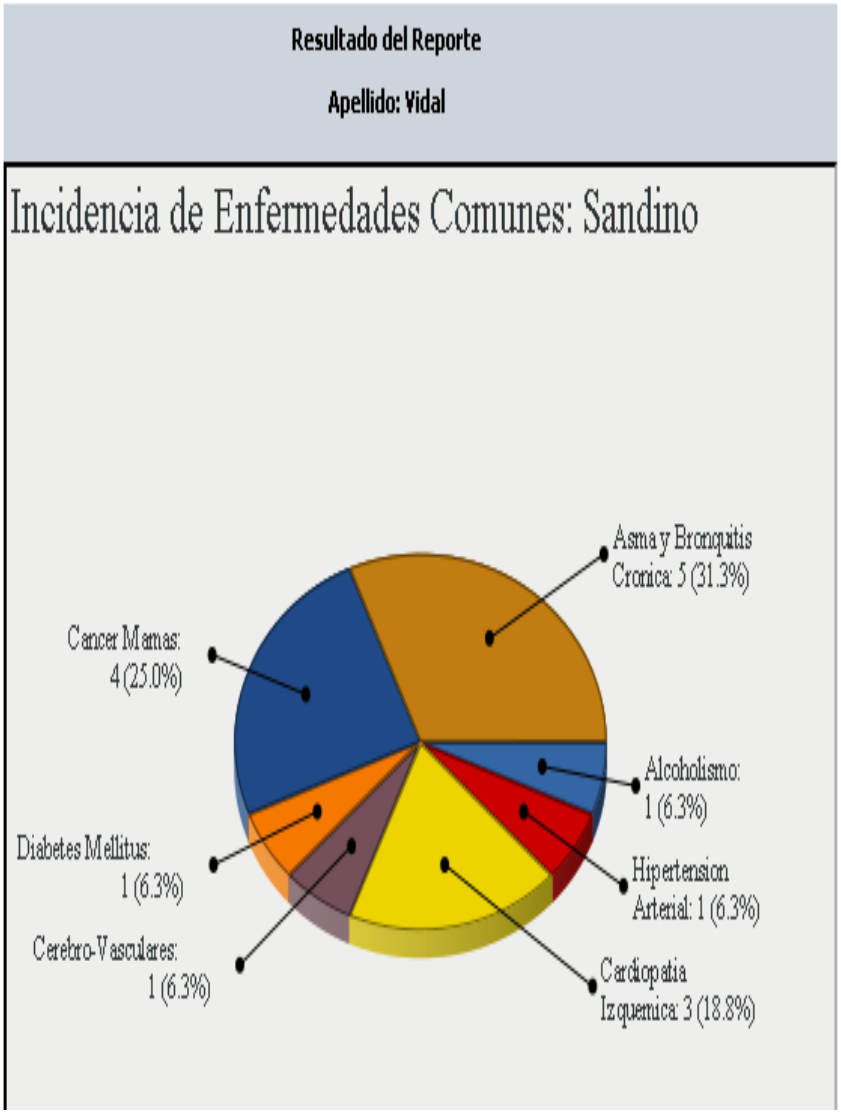


Interfaz para Modificar Datos Complementarios.



Interfaz de Reportes Estadísticos. Reporte de incidencia de enfermedad por región.





4.6 Validación del sistema a nivel de desarrollador.

Durante la implementación del Registro Genético Preventivo de Familias con Enfermedades Comunes, se le realizó a la aplicación web diferentes comprobaciones con el objetivo de probar que el sistema cumple con sus funcionalidades específicas; se intentó encontrar fallos en la aplicación web, corregirlos y así lograr que las funciones del software sean operativas, que las entradas se acepten de forma adecuada, se produzca una salida correcta y que la integridad de la información se mantenga. Para ello se explotaron al máximo las entradas de datos válidos y erróneos con el objetivo de encontrar vulnerabilidades en el sistema y percibir cuán estrictos eran los métodos de validación empleados. Las comprobaciones realizadas se centraron en lo que se espera del Sistema, se chequeó de forma periódica su comportamiento durante la ejecución, se verificó sistemáticamente que cada una de las funcionalidades necesarias para el cumplimiento de los requerimientos del sistema hicieran lo esperado, para lograr de ese modo satisfacer las necesidades del cliente. Se llevaron a cabo tres tipos de validaciones fundamentalmente: haciendo uso de los componentes de Symfony se utilizaron validaciones a través de la función “valídate()”, por archivos “yml” y haciendo uso de la clase sfValidation, sfNumberValidator, sfStringValidator, para realizar validaciones más especificas debido a la complejidad y extensión de los formularios de entrada de datos, además de las validaciones en javascript. Las principales validaciones utilizadas estuvieron encaminadas a evitar que el usuario, producto a la extensión de los formularios y complejidad, fuera a dejar campos vacíos o sin seleccionar, introducir datos incorrectos o fuera de los rangos establecidos, y así propiciar que la entrada de información al sistema fuera correcta y permita realizar con efectividad los estudios genéticos necesarios para el CNGM.

4.7 Conclusiones.

Dicho capítulo propició comprender la implementación del sistema en términos de componentes, se presentaron los diagramas de componentes de implementación de cada una de las funcionalidades, así como una explicación de las clases fundamentales y el objetivo de las principales funciones que encapsula cada una de ellas. Se plantearon los estilos de codificación necesarios para realizar una programación legible y con claridad. Se mostraron las interfaces principales con las que constará el sistema para un mayor entendimiento de la aplicación y se explicaron las diferentes validaciones que se realizaron a lo largo del proceso de desarrollo del software.

Conclusiones.

Conociendo el funcionamiento del Registro Genético Preventivo de Familias con Enfermedades Comunes del Centro Nacional de Genética Médica y la importancia que tiene para disminuir el impacto de las enfermedades comunes con factores de riesgos genéticos asociado, en la población cubana y partiendo de los requisitos funcionales y no funcionales obtenidos para el sistema informático, se realizo un diseño previo del sistema, mediante el cual se analizaron los diferentes artefactos que sirvieron de base para una posterior implementación.

Con la implementación del registro se obtuvo un producto funcional acorde con los requerimientos y las necesidades del cliente, que posibilitará una mejor realización de los estudios genéticos en todo el país para así caracterizar la distribución, comportamiento y factores de riesgo asociados al origen de esas enfermedades en las diferentes regiones del país y con ello llevar a cabo medidas pertinentes para su prevención, tratamiento y disminución.

Recomendaciones.

- Llevar a cabo el despliegue del Registro Genético Preventivo de Familias con Enfermedades Comunes del Sistema Informático de la red nacional de Genética Médica.

- Desarrollar reportes estadísticos de alta complejidad incorporando cálculos estadísticos y matemáticos para determinar factores genéticos más especializados.

- Brindar la posibilidad de impresión de los resultados arrojados por los reportes estadísticos del Registro Genético Preventivo de Familias con Enfermedades Comunes.

- Posibilitar incorporar al sistema el fichero de configuración de la aplicación alasArbogen correspondiente al árbol genealógico de los pacientes.

REFERENCIAS BIBLIOGRÁFICAS.

[1]. **Sourd, Frank, Pompa**. *Arquitectura, normas y tecnologías para el desarrollo de aplicaciones Informáticas para la salud*. 2007.

[2]. **Bussines Consulting Services.** [En línea] 11 de Marzo de 2008. [Citado el: 30 de Octubre de 2008.] http://www.sowre.es/wps/portal/enlace?WCM_GLOBAL_CONTEXT=/WebPublica/Servicios/BusinessConsulting/Ingenieria+del+Software

[3] **Metodologia XP Vs. Metodologia Rup.** [Citado el: 30 de Octubre 2008.] http://metodologiaxpvsmetodologiarup.blogspot.com/

[4] **Jacobson Ivar, Booch Grady, Rumbaugh James**. *El Proceso Unificado de Desarrollo de Software.* España.2000.

[5] **Informatizate**. [En línea] 27 de Noviembre de 2002. [Citado el: 1 de Noviembre de 2008.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

[6] **Molpeceres, Alberto.** *Procesos de desarrollo: RUP, XP y FDD.* 2002.

[7] **Valverde, Pedro Hernández.** *RUP y su relación con las técnicas y métodos de la ingeniería de usuabilidad del software.* 2005.

[8] **Alarcón, Raúl.** *Diseño Orientado a Objetos con UML.* 2000.

[9] **Rumbaugh J., Jacobson I., Booch G**.*"El Lenguaje Unificado de Modelado. Manual de Referencia"* - Editorial Addison-Wesley. 2000.

[10] **Rational Unified Process**.. **Ayuda del RUP. Suite del Rational 2003.**

[11] **Díaz, Moisés Daniel** .*Diseño de aplicaciones Internet usando los Patrones de diseño*.2003

[12] **Larman C**. *UML y Patrones*. 2000.

[13] **El Mundo Informático** [Citado 8 de Diciembre de 2008] http://jorgesaavedra.wordpress.com/category/patrones-grasp/

[14] **Gamma Eric, Hel Richard*, Johnson Ralph, Vlissides John***. *Design Patterns: Elements of Reusable Object-Oriented Software. Cuarta Edición (Enero, 2004)*

[15] **Ciberaula.Java** [Citado 11 de Enero de 2009.] http://java.ciberaula.com/articulo/diseno_patrones_j2ee/

[16] **Acevedo Muñoz, Sergio.** Desarrollo de capas de abstracción para mejorar la eficiencia en la construcción de aplicaciones web. Enero 2008.

[17] **Fernández Pérez, Yudid.** Enfoque administrativo de la Ingeniería del Software. 2007.

[18] **Hernández González, Anaisa.** Identificación de Procesos de Negocio**.** Diciembre 2004.

[19] **IEEE Std 1233a-1998.** Guía para el desarrollo de Especificaciones de Requerimientos de Sistemas.
1998.
[20] **Jiménez López, Rafael.** Análisis y diseño de un framework para el modelado estadístico. 2003
[21] **Open Sourse Plataform**. [Citado 8 de Febrero de 2008.]
http://www.openpopuli.com/manual/estructura_mvc.php
[22] **Zaninotto François, Potencier Fabien**. *Symfony la guía definitiva* http://www.librosweb.es .2007.

Bibliografía.

1- **Zaninotto François, Potencier Fabien**. *Symfony la guía definitiva* http://www.librosweb.es (8/12/2007).

2- **Jacobson Ivar, Booch Grady, Rumbaugh James**. *El Proceso Unificado de Desarrollo de Software.* España. 2000.

3- **Rumbaugh J., Jacobson I., Booch G**., *"El Lenguaje Unificado de Modelado. Manual de Referencia"* - Editorial Addison-Wesley - 2000.

4- **Schach Stephe** . *Ingeniería del Software Orientada a Objetos* (6ªED.)México. 2006. http://bases.bireme.br/cgi-bin/wxislind.exe/iah/online/

5- **Rational Unified Process.. Ayuda del RUP. Suite del Rational.** 2003.

6- **Larman C**. *"UML y Patrones"* - Segunda Edición - Editorial Prentice-Hall .2003.

7- **Gutiérrez Gallardo Juan Diego**. *Desarrollo web con PHP 5 y MySQL*. 2004

8- **Puerto, Roberto**. *Avanzando hacia la sociedad de la información*. GIGA no.1/2002.

9- **Úbeda Luís**. *Vía a la sociedad del conocimiento*. (GIGA no. 4/2005.)

10- **Softel.** Documento sobre la arquitectura de software para los componentes a emplear por el sistema de información para la salud. Softel, 2006.

11- **Pompa Frank**. *Guía para el desarrollo de módulos para el Registro Informatizado de Salud (RIS)*.2005

12- **Delgado Ariel, Vidal María**. *Informática en la Salud Pública Cubana*. 2005.

13- **Rodríguez, José A**. *Tutorial de PHP y MySQL*.

http://es.tldp.org/Manuales-LuCAS/manual_PHP/manual_PHP (20/01/2007)

14- **Larman C**. *UML y Patrones*. 2000.

15-. **Pressman, Roger.** *Ingeniería de software. Un enfoque práctico*. McGraw.Hill/Interamericana de España. 2002.

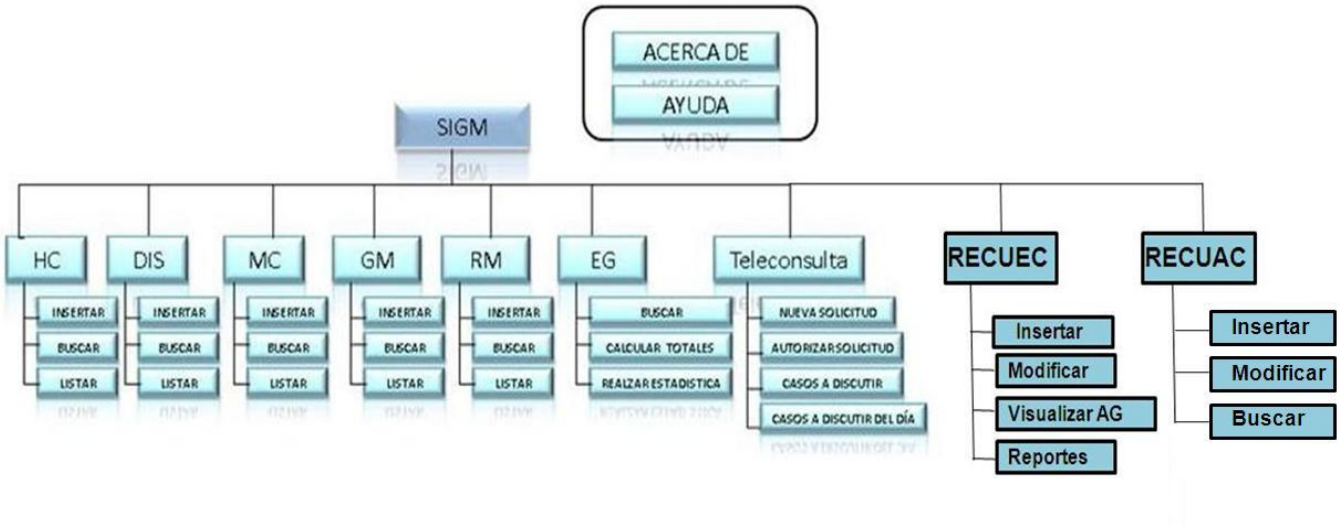
16-. **Collazo Avila, Irina**. Sistema Informático para la Gestión de Información del Registro Cubano de Discapacitados. 2007.

17- **Sánchez Perodín, Yusdenis**. Sistema de Información Genética Línea base de la arquitectura. 2007.

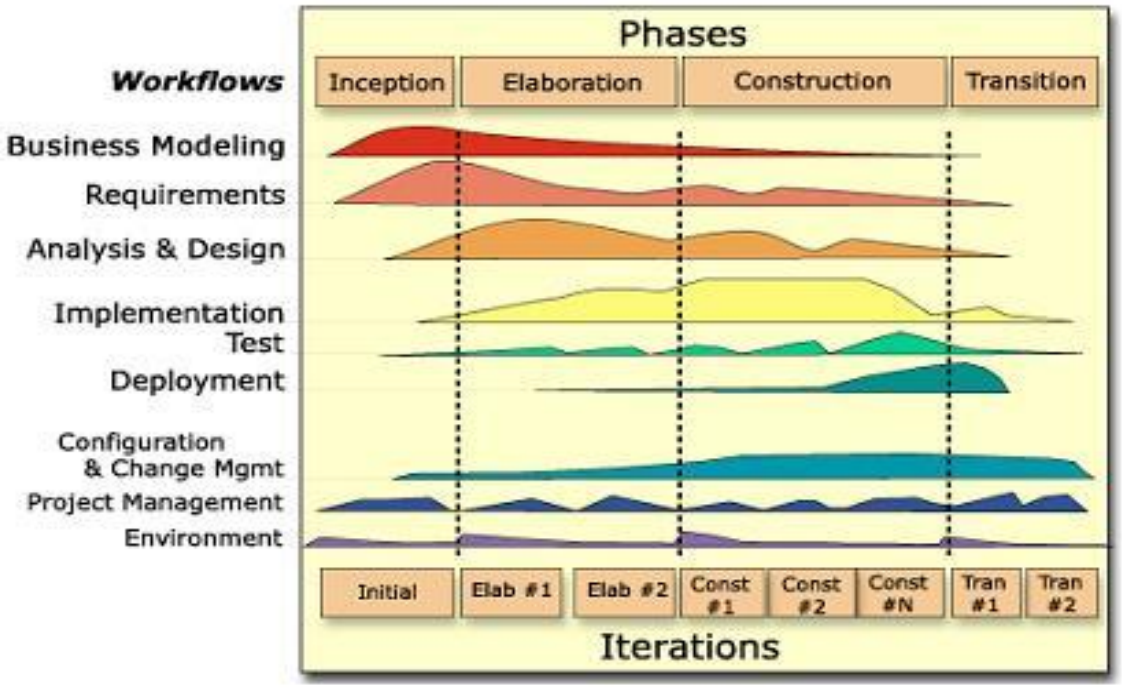
18- **Fernández Pérez, Yudid.** Enfoque administrativo de la Ingeniería del Software. 2007

Índice de Anexos.

Anexo 1. Mapa de alasMEDIGEN.

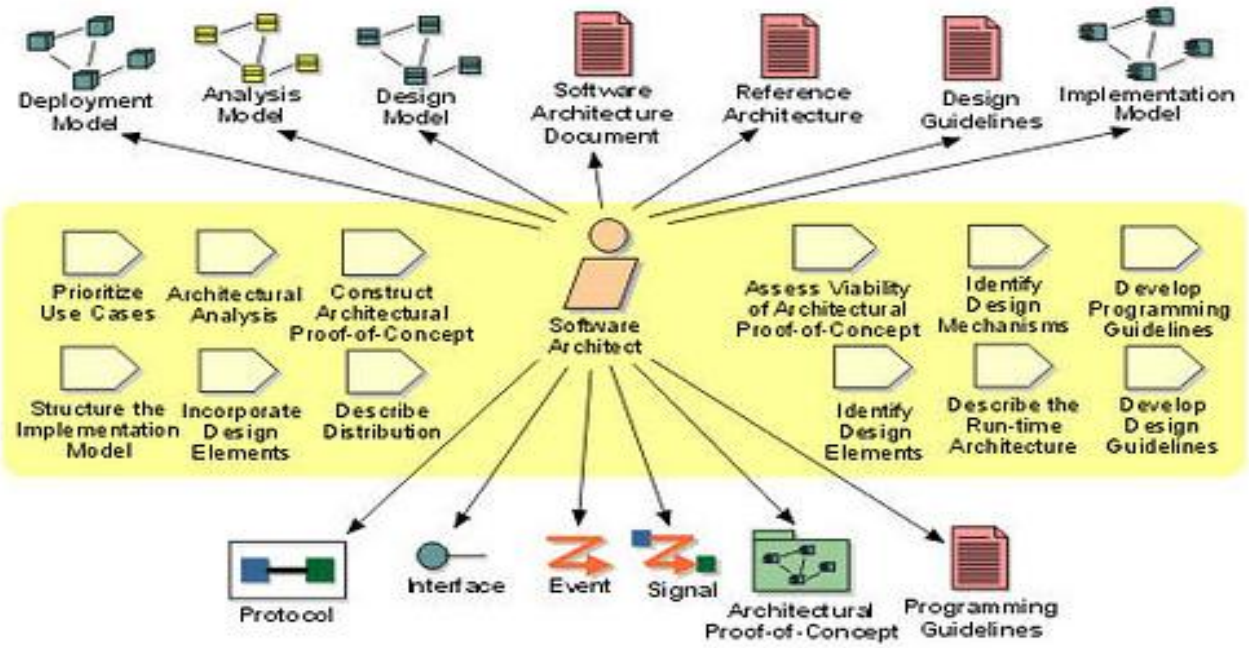


Anexo 2. Fases y Flujos de Trabajos de RUP.

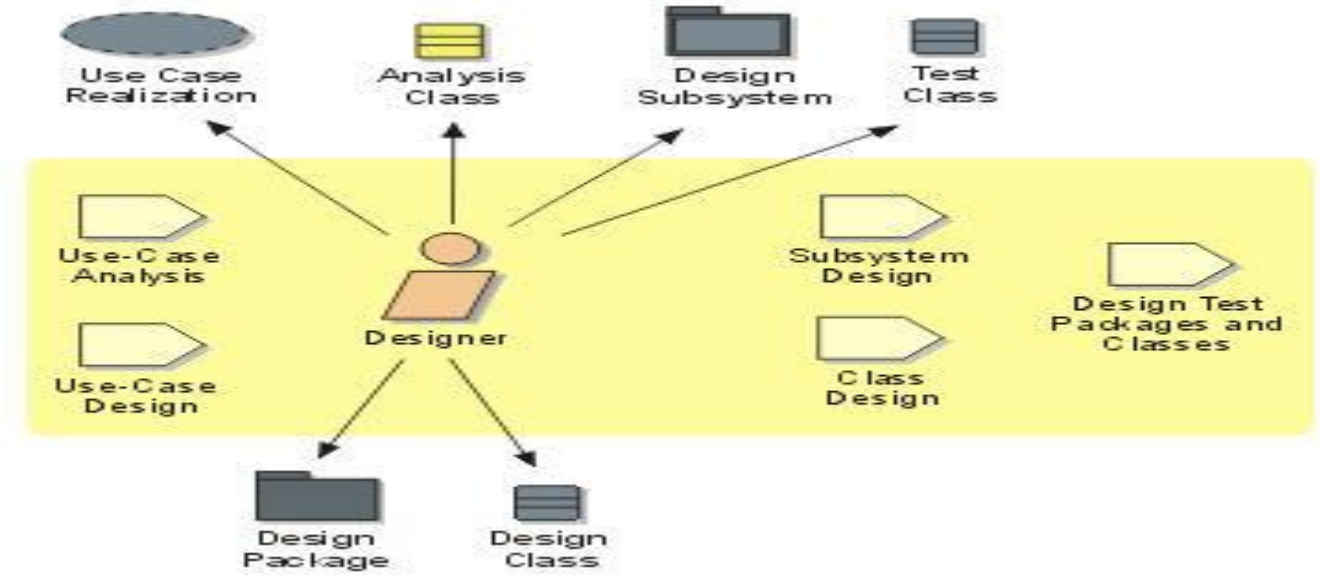


Anexo 3. Roles y Artefactos.

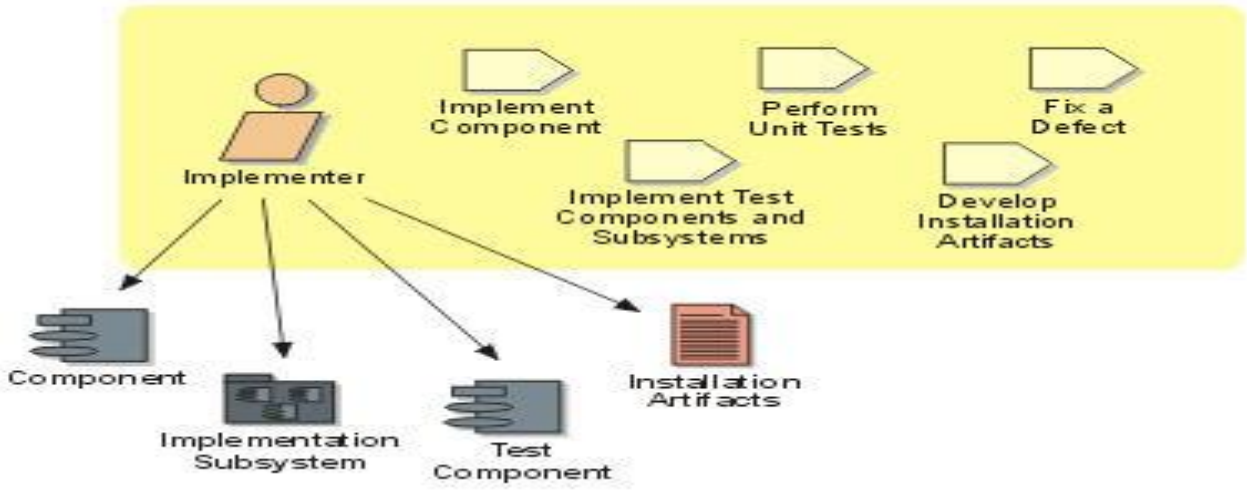
Rol de Arquitecto.



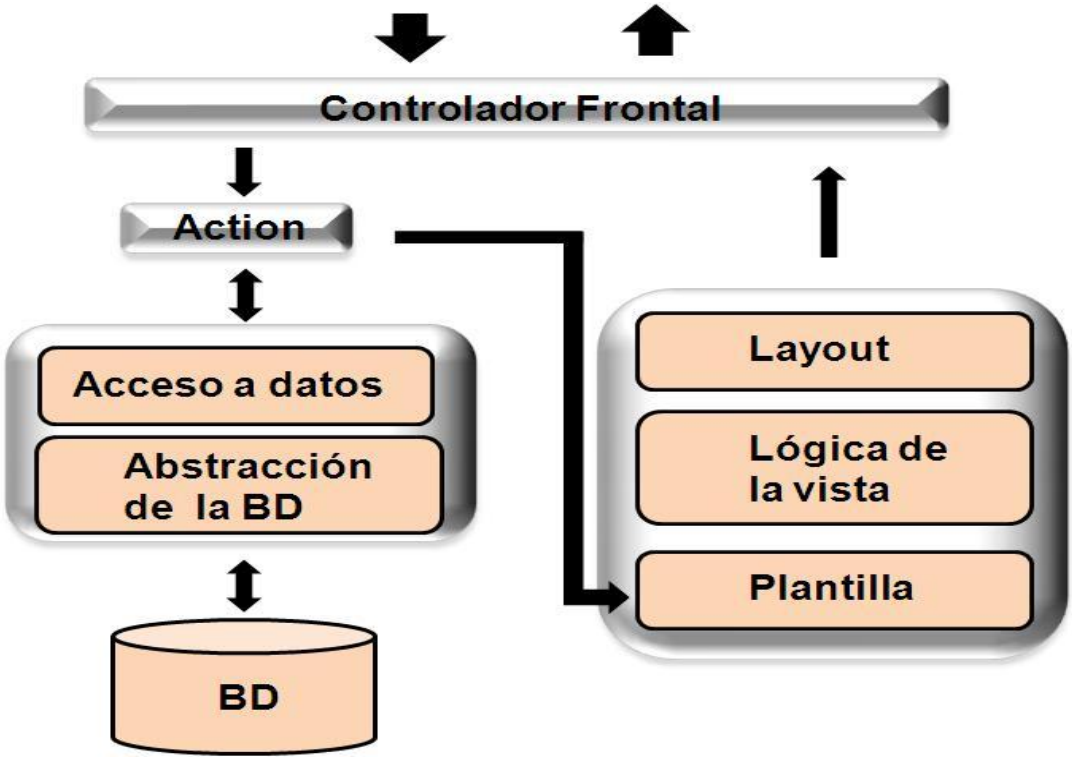
Rol de Diseñador.



Rol de Implementador.



Anexo 4. Arquitectura MVC que utiliza Symfony.



Anexo 5. Descripción de Casos de Uso.

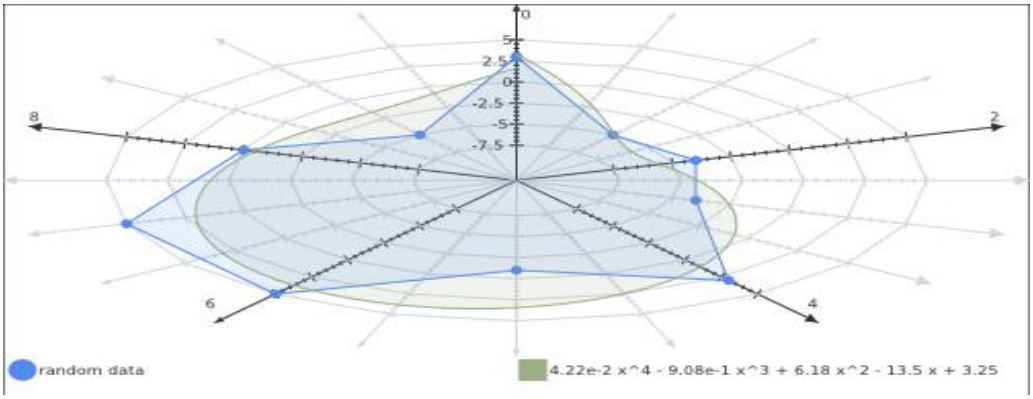
	Visualizar Árbol Genealógico del Paciente.
Actores:	**Usuario** (Inicia el caso de uso)
Resumen:	El caso de uso se inicia cuando el usuario desea consultar el archivo del árbol genealógico del paciente (imagen.jpg). Se muestra un interfaz con los pacientes del registro y el usuario selecciona un paciente. Posteriormente el sistema muestra una interfaz con la imagen del árbol genealógico del paciente seleccionado.
Precondiciones:	El usuario tiene que estar registrado para poder acceder a esta parte del sistema. Debe estar disponible la información a la que tiene acceso el usuario (imagen.jpg).
Referencias	RF3
Prioridad	Secundario
Flujo Normal de Eventos	
Sección “”	
Acción del Actor	**Respuesta del sistema**
1- El usuario selecciona la opción de visualizar árbol genealógico del paciente.	2- El sistema muestra una interfaz donde se muestran los campos necesarios para realizar la búsqueda de pacientes.
3- El usuario entra los datos acordes a los criterios de búsqueda.	4- El sistema le brinda una interfaz que muestra todos los pacientes que cumplen con dicho criterio de búsqueda.
5- El usuario selecciona el paciente del cual desea ver el árbol genealógico.	6- El sistema muestra una interfaz en la cual se visualiza la imagen del árbol genealógico de dicho paciente.
Prototipo de Interfaz	
Flujos Alternos	

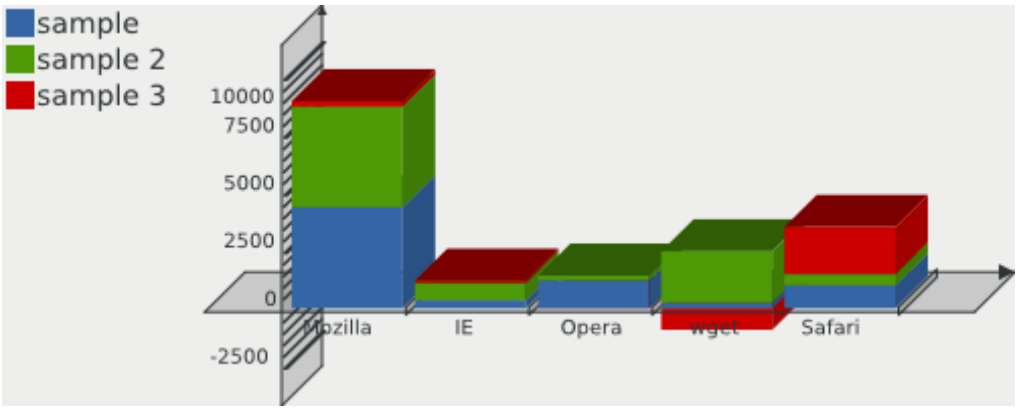
Acción del Actor	Respuesta del sistema
	6- El sistema muestra un mensaje en caso que la imagen de dicho paciente no este disponible.

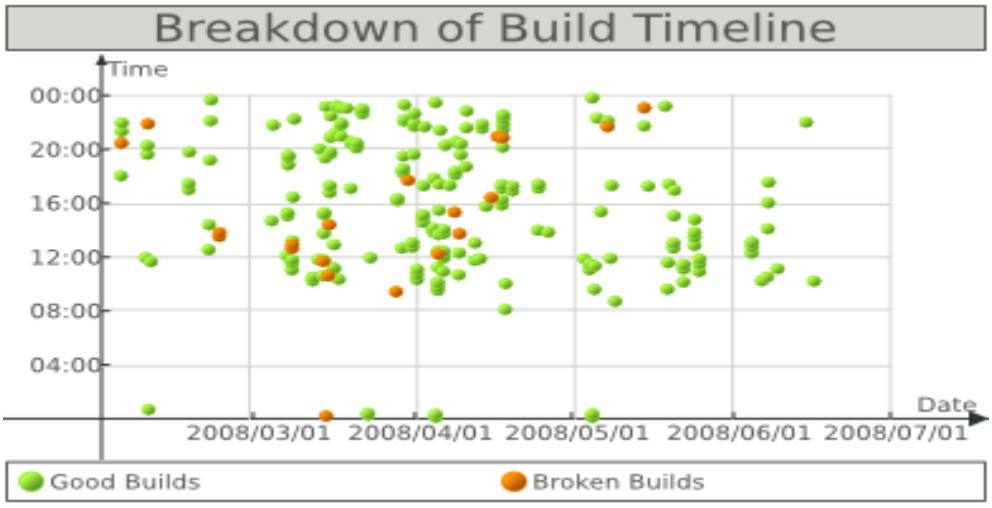
Caso de Uso:	**Mostrar Reportes Estadísticos**
Actores:	Usuario(Inicia el caso de uso)
Resumen:	El caso de uso inicia cuando el Usuario necesita conocer la cantidad de pacientes que existen en dependencia de un criterio determinado, puede ser por incidencia de enfermedades desde área de salud hasta nivel nacional o por el apellido del paciente (primer apellido, segundo apellido, ambos, o cualquiera) y el nombre de la enfermedad, para ello selecciona en el menú el reporte y el sistema se encarga de mostrarle las estadísticas y los cálculos correspondientes.
Precondiciones:	Los usuarios deben estar registrados con los permisos pertinentes para poder acceder a esta parte del sistema. La información debe estar disponible.
Referencias	RF4, RF4.1, RF4.2
Prioridad	Secundaria
Flujo Normal de Eventos	
Sección “”	
Acción del Actor	**Respuesta del Negocio**
1- El usuario selecciona la opción Reporte Estadístico por un determinado criterio (Incidencia, Apellidos).	2- El sistema muestra la interfaz de obtener reporte estadístico correspondiente**.**
3- El usuario escoge el tipo de criterio de acuerdo al reporte seleccionado y el nivel al que desea obtenerlo (nacional, provincial o área de salud).	4- El sistema busca y muestra las estadísticas y los cálculos correspondientes en dependencia de los criterios establecidos.

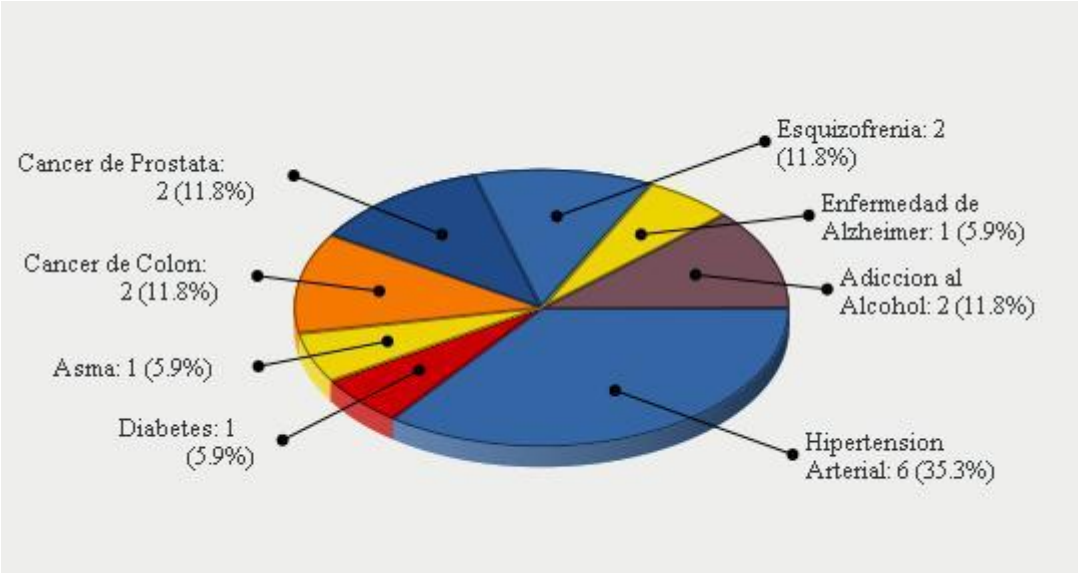
Prototipo de Interfaz		
Flujos Alternos		
Acción del Actor		**Respuesta del Negocio**
Prototipo de Interfaz		
Poscondiciones	Se muestra la información referente al reporte estadístico seleccionado.	

Anexo 6. Gráficas de la Librería ezComponents.






Breakdown of Build Timeline
Time
00:00
20:00
16:00
12:00
08:00
04:00
Date
2008/03/01
2008/04/01
2008/05/01
2008/06/01
2008/07/01
Good Builds
Broken Builds


Esquizofrenia: 2 (11.8%)
Enfermedad de Alzheimer: 1 (5.9%)
Adiccion al Alcohol: 2 (11.8%)
Hipertension Arterial: 6 (35.3%)
Diabetes: 1 (5.9%)
Asma: 1 (5.9%)
Cancer de Colon: 2 (11.8%)
Cancer de Prostata: 2 (11.8%)

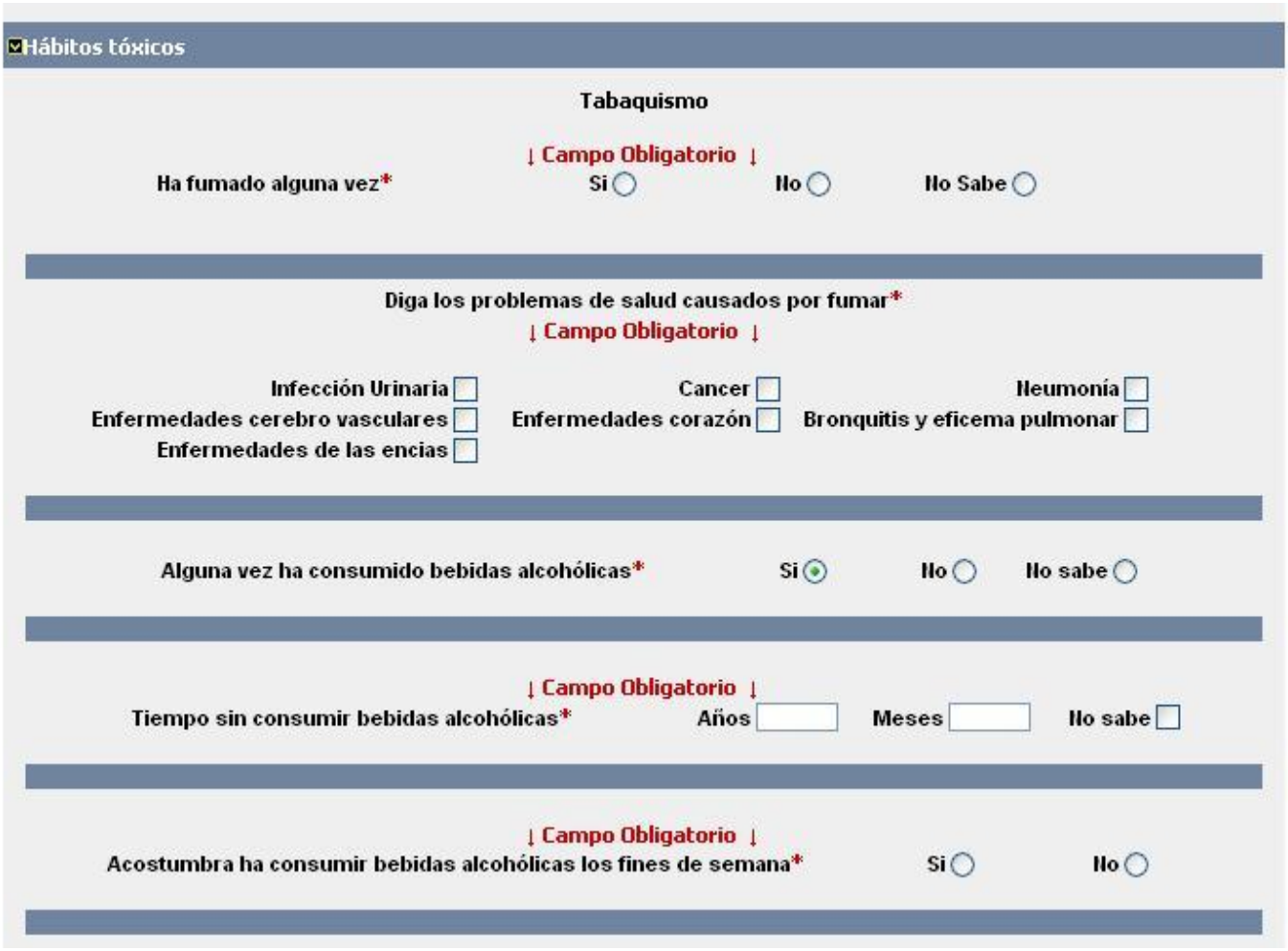
Anexo 7. Interfaces de la aplicación.

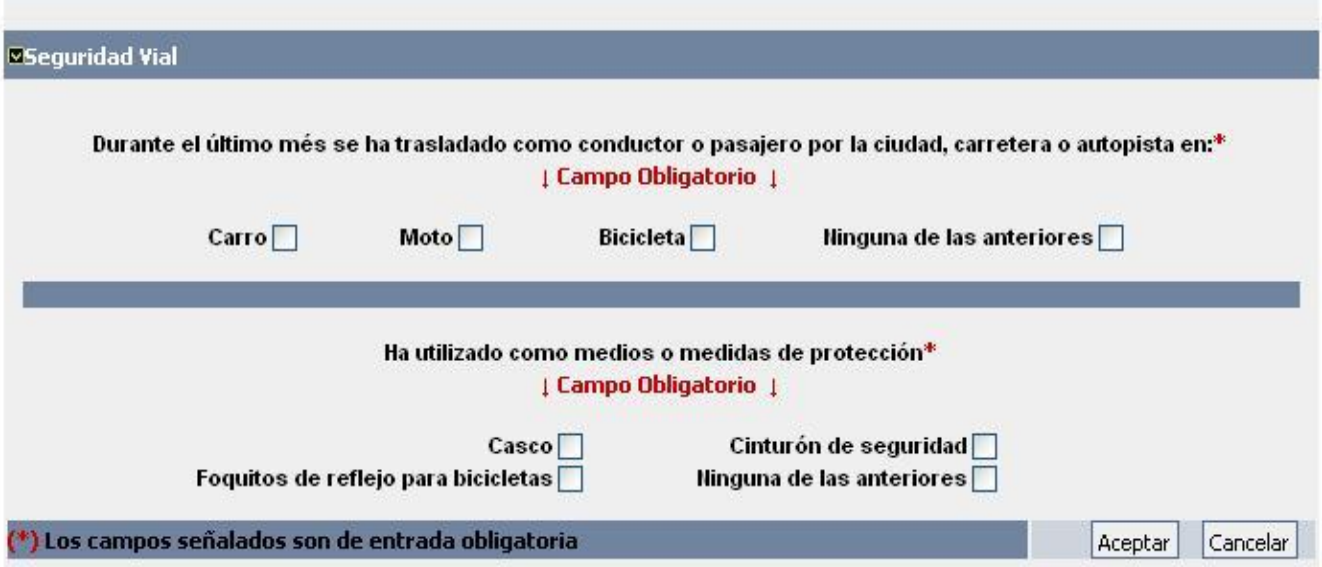
Mensaje que se produce cuando se selecciona para la inserción de los datos complementarios, un paciente ya insertado en el registro.

Mensaje que se produce al insertar con éxito los datos complementarios de un paciente en el registro.

Mensaje que se produce al modificar con éxito los datos complementarios de un paciente.

Interfaces de error que se producen al quedarse campos de entrada obligatoria sin ser llenados.


Hábitos tóxicos
Tabaquismo
↓ Campo Obligatorio ↓
Ha fumado alguna vez*
Si
No
No Sabe
Diga los problemas de salud causados por fumar*
↓ Campo Obligatorio ↓
Infección Urinaria
Cancer
Neumonía
Enfermedades cerebro vasculares
Enfermedades corazón
Bronquitis y eficema pulmonar
Enfermedades de las encias
Alguna vez ha consumido bebidas alcohólicas*
Si
No
No sabe
↓ Campo Obligatorio ↓
Tiempo sin consumir bebidas alcohólicas*
Años
Meses
No sabe
↓ Campo Obligatorio ↓
Acostumbra ha consumir bebidas alcohólicas los fines de semana*
Si
No


Seguridad Vial
Durante el último més se ha trasladado como conductor o pasajero por la ciudad, carretera o autopista en:*
↓ Campo Obligatorio ↓
Carro
Moto
Bicicleta
Ninguna de las anteriores
Ha utilizado como medios o medidas de protección*
↓ Campo Obligatorio ↓
Casco
Cinturón de seguridad
Foquitos de reflejo para bicicletas
Ninguna de las anteriores
(*) Los campos señalados son de entrada obligatoria
Aceptar
Cancelar

Glosarios de Términos.

En el glosario aparecen un grupo de conceptos básicos relacionados con la elaboración del sistema informático.

Siglas:

1 - RECUEC: Registro Genético Preventivo de Familias con Enfermedades Comunes.

2 - CNGM: Centro Nacional de Genética Médica.

3 - INFOMED: Red Nacional de Salud de Cuba.

4 - MINSAP: Ministerio de Salud Pública.

5 - SIGM: Sistema Informático de Genética Médica. Producto del proyecto. Primera Versión.

6 - AlasMEDIGEN: Sistema Informático de Genética Médica. Producto del proyecto. Versión Actual.

7 - SOFTEL: Empresa que ofrece soluciones informáticas para el Sistema de Salud.

8 - SISalud: Sistema de Información para la salud.

9 - SNS: Sistema Nacional de Salud.

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

11- RUP: Proceso Unificado de Rational (Rational Unified Process). Es un proceso unificado de software.

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

13 - UML: Lenguaje Unificado de Modelado (Unified Modeling Languaje).

14 - ORM: mapeo de objetos a bases de datos (object-relational mapping).

15 - SOA: Arquitectura Orientada a Servicios.

Específicos:

1- Hipertensión arterial: condición médica caracterizada por un incremento continúo de las cifras de presión arterial.

2- Cardiopatía isquémica: conjunto de enfermedades del corazón o cardiopatías cuyo origen radica en la incapacidad de las arterias coronarias (coronariopatía) para suministrar el oxígeno necesario a un determinado territorio del músculo cardiaco.

3- Diabetes mellitus: síndrome orgánico, multisistémico y crónico que se caracteriza por un aumento de los niveles de glucosa en la sangre.

4- Asma bronquial: enfermedad inflamatoria de la mucosa bronquial que se acompaña de síntomas nasales oculares o de otras mucosas.

5- Depresión: trastorno emocional que en términos coloquiales se presenta como un estado de abatimiento e infelicidad que puede ser transitorio o permanente.

6- Cáncer de mama: proliferación acelerada, desordenada y no controlada de células con genes mutados, los cuales actúan normalmente suprimiendo o estimulando la continuidad del ciclo celular pertenecientes a distintos tejidos de una glándula mamaria.

7- Cáncer de colon: enfermedad en la cual se encuentran células cancerosas en los tejidos del colon.

8- Cáncer de próstata: enfermedad en la que las células cancerosas se desarrollan en la próstata. Esta es una glándula en forma de nuez que rodea la uretra.

9- Esquizofrenia: diagnóstico psiquiátrico de tipo crónico y severo que describe un grupo de trastornos mentales en personas con alteraciones en la percepción o la expresión de la realidad.

10- Trastorno bipolar: trastorno del estado del ánimo que cuenta con períodos de depresión repetitivos (fases depresivas) que se alternan con temporadas de gran euforia (fases maníacas).

11- Enfermedad de Parkinson: proceso neurodegenerativo del sistema extrapiramidal que afecta aproximadamente al 1% de las personas mayores de 50 años.

12- Enfermedad de Alzheimer: es una forma de demencia, una afección cerebral progresiva y degenerativa que afecta la memoria, el pensamiento y la conducta.

13- Adicción al alcohol: también llamada alcoholismo, puede que sea una de las adicciones más antiguas y extendidas que se conozcan. En este caso, la sustancia activa que afecta al cerebro del bebedor es el etanol, compuesto con propiedades depresivas que con el tiempo propicia la aparición de una fuerte dependencia.