Universidad de las Ciencias Informáticas

Facultad 7

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Sistema para la gestión de nomencladores

AUTORES: Renán Vázquez Moreno

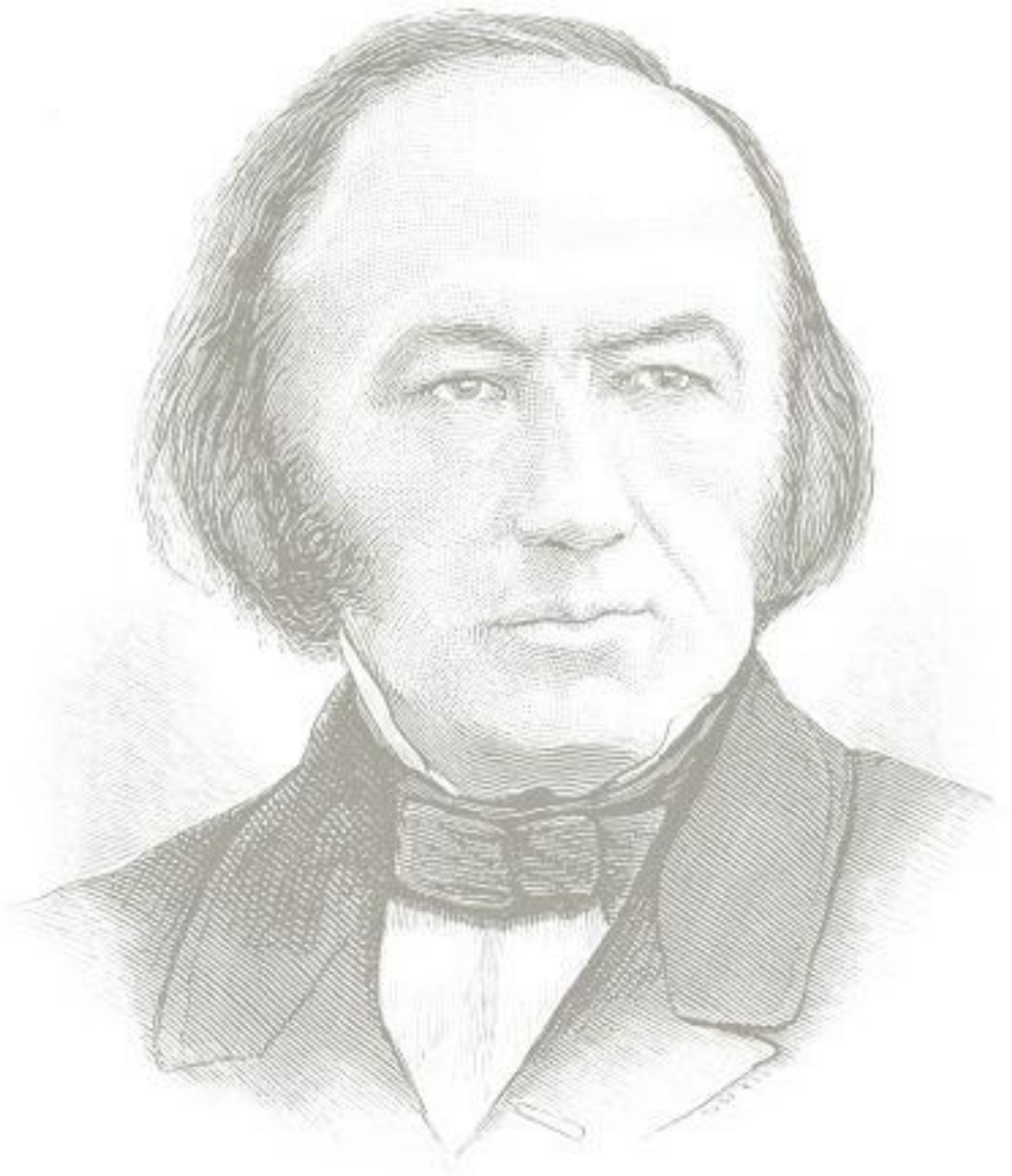
José Mojena Alpizar

TUTORA: Ing. Annia Arencibia Morales

COTUTOR: Ing. Rolando Pompa González

La Habana, junio 2011

"Año 53 de la Revolución"



"Cuando un hombre de ciencia busca conocimientos, aún no hallándolos en su totalidad, descubre fragmentos muy importantes, que son precisamente los que constituyen la ciencia."

Claude Bernard

DATOS DE CONTACTO

TUTOR:

- **Ing. Annia Arencibia Morales** (aarencibia@.uci.cu):Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2007. Profesor de la Universidad de las Ciencias Informáticas, pertenece al Centro de Informática Médica (CESIM), posee categoría docente de profesor Asistente. Se desempeña como líder del Proyecto Synta, Se encuentra cursando la Maestría Informática Aplicada.

COTUTOR:

- **Ing. Rolando Pompa González** (rpompa@uci.cu):Obtuvo el título de Ingeniero en Ciencias Informáticas en la tercera graduación, curso 2008-2009, de la Universidad de las Ciencias Informáticas (UCI). Es miembro del Centro de Informática Médica (CESIM) en el Departamento de Sistemas de Apoyo a la Salud (SAS) como desarrollador del proyecto Synta y Arquitecto Principal del departamento.

Agradecimientos de José

A mi familia por apoyarme en todo momento.

A mis queridos padres Raquel y Mojena por ser mi fuerza inspiradora para ser cada día mejor y lograr que las cosas me salgan bien.

A mi tía Leimis por ser una de las personas que tomé como ejemplo en la vida para seguir adelante y estar pendiente de mi vida como si fuera mi otra madre.

A mi querido hermano Javier que me ha obligado a ser cada día mejor ya que en todo momento me está tomando como ejemplo.

A mi abuela Victoria que siempre ha estado ahí para demostrarme la fortaleza que puede llegar a tener las personas.

A mi abuelo Orfilio que sin saber lo que es una escuela, ni si tenía importancia estudiar o no, cuando entré en la universidad me apoyó hasta el día de su partida.

A mis abuelos Inés y Orestes que aunque ella no esté físicamente hubiera estado contentísima conmigo por lo logrado y mi abuelo que aunque en estos momentos su mente no puede llegar a valorar cosas se, que en plenitud de conciencia estaría lleno de alegría.

A mí querida novia Idayana que me ha ayudado mucho en estos 5 años y que sin ella creo que hubiera sido todo un poco más difícil

A los tutores Annia y Pompa por la guía durante este tiempo en el que hemos trabajo juntos.

A mi compañero de tesis Renán por haber trabajado incansablemente para que la investigación terminara satisfactoriamente y en la cual hicimos un buen equipo.

A la UCI por darme la oportunidad de formarme como ingeniero, a mis amigos que son muchos y necesitaría mucho espacio para citarlos a todos pero que están presentes, a mis profes de la carrera, a todos muchas gracias.

Agradecimientos de Renán

Detrás de un trabajo como este, hay un grupo de personas que te ayudan y apoyan incondicionalmente. Es por eso que quisiera agradecer:

A mi familia, por el inmenso apoyo que en todo momento me ha dado.

A mis padres Vivian y Renán, por el amor y apoyo que de ellos siempre he recibido y que han hecho que en los momentos difíciles haya podido seguir adelante. Por ser los guías en mi vida, por sus consejos y por la educación que me han dado, que ha hecho de mí todo lo que soy.

A mi tía Solange, por estar siempre pendiente de mí y mis estudios, convirtiéndose en otra madre para mí.

A mi hermana Dayana, por su cariño y por estar siempre a mi lado.

A mi novia Dayanna, por su amor, dedicación y comprensión en todo momento. Por ver más allá de los problemas y por aceptarme con mis virtudes y defectos.

A mis abuelos Rosa María e Ignacio y Celia y Elpidio, por creer en su nieto siempre y por darme las mejores enseñanzas que la experiencia puede brindar.

A mi compañero de tesis, por la dedicación y esfuerzo que en este trabajo ha puesto y por ayudarme en mi formación como ingeniero.

A mis compañeros de estos cinco años, en especial a mi hermanita Yurlenis, a los que siempre han estado a mi lado y a los que ya no están, pero que siempre quedarán en mi recuerdo.

A los profesores, por sus enseñanzas y a los tutores por guiarnos durante el desarrollo del trabajo.

A la revolución y en especial a su máximo líder, nuestro invencible Comandante en Jefe Fidel Castro Ruz, por crear esta hermosa Universidad de las Ciencias Informáticas que ha me ha dado la oportunidad de formarme como ingeniero.

A los que de una forma u otra me han ayudado a alcanzar este resultado, muchas gracias a todos.

Dedicatoria

Dedicado a mi familia y en especial a mis padres, por ser el tesoro más valioso que poseo. Por enseñarme lo verdaderamente valiosos en la vida y por indicarme siempre el camino correcto, ese mismo camino que hoy me ha conducido a obtener este resultado, que espero los haga sentir orgullosos de mí.

Renán Vázquez Moreno

Dedico el resultado de esta investigación a las personas más importantes de mi vida que son mi familia y en especial a mis padres, dos personas por las que soy capaz de enfrentarme a cualquier cosa en la vida, también dedico esta tesis a mi novia que forma parte de mi vida desde hace bastante tiempo y que está muy orgullosa de mí.

José Mojena Alpizar

RESUMEN

A partir de la necesidad de gestionar información común y poco variable en el tiempo se crea el Módulo Nomenclador del sistema Balance y Planificación de Insumos Médicos (alas BAP), con el objetivo de centralizar la gestión de los nomencladores. Dicho módulo presentaba deficiencias en su funcionamiento: no existían servicios Web que permitieran acceso a la información de los nomencladores por otros sistemas y presentaba una dependencia total del sistema alas BAP, por lo que se dio la tarea de desarrollar un sistema que eliminara las deficiencias existentes en el mismo.

Para el desarrollo del sistema se utilizó Symfony como framework de desarrollo de aplicaciones Web que implementa el patrón de arquitectura de software Modelo Vista Controlador (MVC). Como lenguaje de programación se utilizó Hypertext Pre-processor (PHP), el cual incluye el Mapeador de Objetos Relacional (ORM por sus siglas en inglés) Doctrine para el acceso a datos. Como herramienta de modelado se empleó Enterprise Architect (EA); herramienta de análisis y diseño con Lenguaje Unificado de Modelado (UML). Como Sistema de Gestión de Base de Datos (SGBD) se utilizó PostgreSQL y para el manejo de datos la herramienta pgAdmin.

La solución desarrollada constituirá un sistema configurable y flexible ante posibles cambios que permitirá el almacenamiento y gestión de nomencladores y servirá como mecanismo de apoyo a otras aplicaciones para la gestión y utilización de nomencladores, brindando dicha información a través de servicios Web XML, agilizando de esta manera el tiempo de desarrollo de estos sistemas.

Palabras claves*: Symfony, Doctrine, Nomencladores.*

ÍNDICE

INTRODUCCIÓN

La informatización de la sociedad es el proceso de utilización ordenada y masiva de las Tecnologías de la Información y las Comunicaciones (TIC) en la vida cotidiana, para satisfacer las necesidades de todas las esferas de la sociedad, en su esfuerzo por lograr cada vez más eficacia y eficiencia en todos los procesos y por consiguiente mayor generación de riqueza y aumento en la calidad de vida de los ciudadanos. (1)

Una sociedad que aplique la informatización en todas sus esferas y procesos deberá ser más eficaz, eficiente y competitiva. Es evidente que para los países subdesarrollados resulta un reto el logro de este propósito, producto a su problemática fundamental que está en lograr la supervivencia de sus pueblos. (2)

Cuba ha identificado desde muy temprano la conveniencia y necesidad de introducir en la práctica social las TIC; y lograr una cultura digital como una de las características imprescindibles del hombre nuevo, lo que facilitaría a la sociedad acercarse más hacia el objetivo de un desarrollo sostenible. (3)

El Ministerio de Salud Pública (MINSAP) ha sido uno de los sectores beneficiado con la aplicación de las TIC. En este sentido se desarrolla la Red Telemática de Información de Salud (INFOMED), que interconecta: policlínicos, hospitales, centros de enseñanza, bibliotecas y otras instituciones en todas las provincias del país.

Existen empresas que participan en la informatización del sector de salud, tal es el caso de la Empresa Nacional de Software (DESOFT), la Empresa de Soluciones Informáticas para el Sistema de Salud (SOFTEL), y la Universidad de las Ciencias Informáticas (UCI); esta última es un centro estudiantil que inserta a sus estudiantes y profesores a proyectos productivos en los que se desarrollan software para las distintas esferas de la sociedad cubana.

La UCI está conformada por 7 facultades en la sede central y tres facultades regionales, dentro de las primeras se encuentra la Facultad 7 que cuenta con un centro productivo llamado Centro de Informática Médica (CESIM) estructurado en 6 departamentos, uno de estos es el Departamento de Sistemas de Apoyo a la Salud (SAS) encargado de implementar y dar soporte a sistemas orientados a la toma de

decisiones y gestión de equipos médicos. Uno de los sistemas que desarrolló fue el Sistema de Balance y Planificación de Insumos Médicos (alas BAP). Este está estructurado en 5 módulos estos son: Almacén, Economía, Planificación, Balance y Nomenclador, este último debía permitir: (4)

- Creación de campos de forma dinámica, brindando las opciones de nombrar, cambiar y modificar desde el tamaño hasta el tipo de componente.
- Asignar campos: a los elementos nomenclados se le deben realizar una asignación de campos en dependencia de las características del nomenclador que se esté creando, permitiendo que cada grupo tenga nada más la información que le corresponda.
- Información: corresponde a los valores de los nomencladores, esta rama del árbol se maneja como hojas y estas obtendrían los campos de sus padres, a través de la herencia.
- Códigos: a la hora de crear un nomenclador el usuario puede crear un código a cada rama siguiendo varios criterios, solo tendría que realizar el juego de datos y asignar a cada rama el código que desee.
- Servicios Web: se crean para brindar la información de los nomencladores, esta es una de las principales funcionalidades con las que se cuenta, permitiendo a través de la publicación que sus principales características sean consumidas por el usuario.

El Módulo Nomenclador fue un avance en el sistema alas BAP, pero presenta algunas deficiencias que se enuncian a continuación:

- No es posible gestionar el código de forma correcta, teniendo en cuenta que este es un atributo fundamental en la creación de un nomenclador, constituye esto una deficiencia importante.
- El módulo no brindaba servicios Web que permitieran a otras aplicaciones clientes consumir esta información ya estructurada.
- Para el desarrollo del módulo se utilizó el ORM Propel para el manejo de la Base de Datos siendo difícil el intercambio con esta, ya que el lenguaje de consultas es complejo, no siendo así con el ORM Doctrine el cual tiene incluido un Lenguaje de Consulta de Doctrine (DQL por sus siglas en

inglés) muy parecido al Lenguaje de Consulta Estructurado (SQL por sus siglas en inglés) por lo que las consultas a la Base de Datos se harían de manera más clara y entendible.

- La estructura y jerarquía de los campos que componían a los nomencladores dificultaba el trabajo con los valores multivaluados y con los componentes de selección y multiselección, algo que atentaba contra la funcionalidad del módulo.
- El Módulo Nomenclador, como su nombre lo indica, era dependiente del sistema alas BAP ya que representaba un módulo de dicho sistema, por lo que no podía ser utilizado como una aplicación independiente.

Después de analizar las deficiencias de la versión inicial las cuales simplificaron el sistema desarrollado a solo una parte de las funcionalidades inicialmente analizadas y diseñadas surge como **problema a resolver:** Gestión inadecuada de la información poco variable en el tiempo en el Módulo Nomenclador del sistema Balance y Planificación de Insumos Médicos (alas BAP). Centrándose en el **objeto de estudio** proceso de gestión de la información poco variable en el tiempo durante el desarrollo de sistemas informáticos. Definiéndose como **campo de acción** proceso de gestión de la jerarquía de la información poco variable en el tiempo durante el desarrollo de sistemas informáticos.

El **objetivo general** es desarrollar un sistema nomenclador que gestione la jerarquía de la información poco variable en el tiempo, facilitando el desarrollo de sistemas informáticos.

Para darle cumplimiento al objetivo antes planteado se han definido las siguientes tareas:

- Valorar los sistemas de gestión de nomencladores a nivel internacional y en Cuba, estableciendo similitudes con la investigación en curso.
- Analizar el Módulo Nomenclador obteniendo las deficiencias presentadas logrando un modelo de desarrollo único que sirva de guía para la implementación del sistema.
- Asimilar las herramientas y tecnologías propuestas por el Departamento de Sistemas de Apoyo a la Salud (SAS), para el desarrollo de la solución.

- Generar los artefactos correspondientes a los flujos de trabajo propuestos por la metodología de desarrollo RUP sirviendo de base para el desarrollo del sistema.
- Implementar un sistema nomenclador aplicando las pautas de diseño definidas por el Departamento SAS siguiendo lo establecido en la especificación de requisitos de software.

Entre los beneficios que se esperan con el desarrollo de la investigación se encuentran:

- Constituirá un sistema configurable y flexible, permitiendo al usuario personalizar la información que desee gestionar, a través de la creación y asociación de campos. La estructura del código en su creación podrá ser tan flexible como el usuario desee.
- Servirá como mecanismo de apoyo a otras aplicaciones, disminuyendo el tiempo de desarrollo y permitiendo que estas se centren en el consumo y no la implementación de los nomencladores.
- Los sistemas no tendrán que reprogramarse, solamente actualizarían su información en caso de que esta cambiara en un tiempo determinado.
- El sistema podrá ser utilizado por otras aplicaciones independientemente de la plataforma en que estén desarrolladas, haciendo uso de servicios Web XML.

El presente trabajo está divido en cuatro capítulos que se describen a continuación:

Capítulo 1: Fundamentación Teórica, se incluye el estado del arte del tema tratado tanto a nivel internacional como nacional. Se describen los conceptos fundamentales asociados al dominio del problema y se hace un análisis entre las soluciones existentes y la propuesta que se realiza. Además se analizan las herramientas, tecnologías y metodologías a usar en la solución del problema planteado.

Capítulo 2: Características del Sistema, se realiza una breve descripción del Modelo de Dominio así como de sus elementos más importantes. Además se especifican los requerimientos funcionales y no funcionales, a partir de los cuales se representan los casos de uso del sistema y la descripción de los mismos.

Capítulo 3: Diseño del Sistema, se justifican los patrones a usar en el diseño de la aplicación, se define la estructura y los elementos del diseño, se muestra el modelo de datos y los diagramas de clases del

diseño de los casos de uso del sistema. Se conforma el Modelo de Diseño, el cual constituye una base para la futura implementación.

Capítulo 4: Implementación, las clases y subsistemas encontrados durante el diseño se describen en términos de componentes. Se describen las funcionalidades más significativas del sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

El presente capítulo tiene como objetivo hacer un estudio del estado del arte de los sistemas existentes para la gestión de nomencladores a nivel nacional e internacional. Además se analizan las tecnologías, metodologías y herramientas a utilizar en el desarrollo del sistema.

1.1 Antecedentes

Después de una búsqueda de sistemas informáticos que gestionen información poco variable en el tiempo, tanto en el ámbito internacional como nacional se encontraron aplicaciones que aún cuando no se pueda generalizar su uso, sirven de referencia para comprender los procesos de gestión de la información manejada.

1.1.1 Características de algunos sistemas internacionales

- **Nomenclador Nacional**

El Nomenclador Nacional fue creado en el año 1977 es, básicamente, un listado codificado de servicios médicos quirúrgicos, de laboratorios y hospitalarios, valorizados en unidades convencionales ajustados periódicamente. (5)

Está orientado a un: (6)

- Listado que conforma un catálogo, con códigos indicadores (prestaciones-grupo de prestaciones-diagnósticos-equipos médicos) para orientar las actividades administrativas y de control.
- Cuerpo normativo y reglamentario que dará las guías de trabajo para la calidad y la instrumentación del sistema arancelado.
- Orden en tablas para la valorización de los códigos por medio de unidades referidas a honorarios profesionales y gastos de la práctica, que dan el arancel de la prestación.

- **Nomenclador de procedimientos y servicios**

El nomenclador de procedimientos y servicios es un documento técnico que consiste en el catálogo estandarizado de los procedimientos y servicios de salud disponibles en el mercado privado de salud, constituyéndose, por lo tanto, en un valioso instrumento dentro del Sistema Integrado de Información en Salud. (7)

El nomenclador ha sido elaborado de manera conjunta por la Asociación de Clínicas Particulares (ACP), la Asociación Peruana de Empresas de Seguros (APESEG) y la Asociación Peruana de Entidades Prestadoras de Salud (APEPS). La Superintendencia de Entidades Prestadoras de Salud (SEPS) participó en su elaboración como facilitador y coordinador, en su calidad de principal usuario del Nomenclador como estándar de procedimientos y servicios médicos. (8)

El instrumento desarrollado se enmarca dentro de la propuesta de Transmisión Electrónica de Datos Estandarizados del Expediente de Facturación (TEDEF) que la Superintendencia de Entidades Prestadoras de Salud (SEPS) viene impulsando y tiene el propósito de contar con una base de datos estandarizada de información administrativa y económica, permanentemente actualizada, de manera que se constituye en un insumo importante para la toma de decisiones en todos los niveles de las organizaciones de salud. (9)

El nomenclador detalla todos los procedimientos y servicios disponibles en el mercado privado de salud. No todos ellos son cubiertos por los financiadores. Los procedimientos y servicios que no tienen unidades están sujetos a condiciones pactadas entre financiadores y proveedores. (10)

- **Nomenclador cartográfico para personas con deficiencia visual.**

Dos estudiantes de la Facultad de Matemática, Astronomía y Física de Córdova en Argentina, desarrollaron un software de distribución libre y gratuita, para facilitar el desplazamiento de los discapacitados visuales por la ciudad de Córdoba. El nomenclador brinda información para trasladarse de un punto a otro, para personas con discapacidad visual.

El programa se distribuirá bajo Licencia Pública General (GPL) para que pueda ser modificado, mejorado y ampliado por otros desarrolladores. Fue realizado en Java, un lenguaje que funciona en diversos sistemas operativos como Windows y Linux, entre otros. Tiene una arquitectura cliente-servidor. El servidor aloja la base cartográfica y procesa las solicitudes que recibe de los clientes. Así, se otorga

mayor seguridad en la protección de los datos. Los autores consideran que en el futuro se deberán incorporar datos sobre el recorrido del transporte público y sumar tecnología de geoposicionamiento satelital. (11)

El nomenclador cartográfico desarrollado durante los años 2007 y 2008 contiene las siguientes funcionalidades:

- Consulta sobre existencia de una dirección particular en la ciudad de Córdoba.
- Consulta del camino más corto a pie entre dos puntos de la ciudad.

El resultado de una consulta se compone de:

- Mapa del camino a realizar, contiene colores en alto contraste para facilitar la creación de un mapa mental por parte del usuario con deficiencia visual parcial.
- Instrucciones en texto que guían al usuario sobre cómo llegar a destino.
- Lectura audible de las instrucciones, permite guardar un archivo de sonido que podrá ser alojado en un celular o mp4 para su reproducción durante el trayecto.

1.1.2 Características de algunos sistemas nacionales

- **Registro de la Clasificación Internacional de Enfermedades (RCIE)**

El módulo tiene como objetivo principal la adopción de un lenguaje unificado y estandarizado, propuesto por la Organización Mundial de la Salud (OMS) en la Décima Revisión de la Clasificación Estadística Internacional de Enfermedades y Problemas Relacionados con la Salud (CIE-10), todos los módulos que trabajen con diagnósticos tienen relación con este módulo dado que es el que gestiona la codificación de todas las enfermedades y problemas relacionados con la salud que se puedan presentar en la atención médica diaria y causas de lesiones y muerte para su posterior cómputo y generación de estadísticas de morbilidad, mortalidad y servicios, entre otros. (12)

El Editor a nivel nacional de este módulo es el único que puede agregar, modificar o eliminar los capítulos, grupos, categorías, subcategorías y problemas de salud que constan en la CIE así como sus respectivas descripciones. Por otro lado este módulo tiene como valor agregado que sirve de una especie de software

educativo para el conocimiento de la CIE en el cual el usuario a modo visualizador puede hacer búsquedas o leer todo lo relacionado con las diferentes enfermedades. (13)

- **Registro de Problemas de Salud de la Atención Primaria (RPSAP)**

El módulo tiene como objetivo principal gestionar una clasificación especial para los problemas de salud que se presentan en la Atención Primaria de Salud (APS) tratando que resulte una clasificación más breve y de manejo más simple que la CIE-10 y además compatible con esta. Permite realizar búsquedas dinámicas de acuerdo a criterios seleccionados por el usuario. Ofrece un lenguaje unificado y estándar para la información de la morbilidad en la APS. (14)

El Editor a nivel nacional de este módulo es el único que puede gestionar (Agregar, Modificar y/o Eliminar) la información (Nombre, Grupo y Descripción) de los Riesgos, Enfermedades y Discapacidades registradas así como gestionar la información (Nombre y Tipo) de los Grupos y Sub-Grupos de discapacidades. En el caso del usuario visualizador puede obtener listado general de Grupos de Riesgos y Riesgos incluidos en cada grupo, Grupos de Enfermedades y Enfermedades incluidas en cada grupo, Grupos de Discapacidades y Discapacidades incluidas en cada grupo y realizar búsquedas y conseguir en base a los criterios de la búsqueda, Listado de Riesgos específicos por Grupos, Listado de Enfermedades específicas por Grupos y detalles de códigos de CIE-10 que agrupan, y Listado de Discapacidades específicas por Grupos y detalles de códigos de CIE-10 que agrupan. (15)

- **Registro de Enfermedades de Declaración Obligatoria (REDO)**

Este módulo tiene como objetivo principal apoyar el proceso de vigilancia en salud que se realiza en las diferentes instancias del sistema sanitario, a través de una rápida detección y notificación de enfermedades transmisibles y no transmisibles así como otras de interés para el Sistema Nacional de Salud (SNS). (16)

- **Registro de Indicadores y Conductas para la Atención Primaria (RICAPS)**

Este módulo tiene como objetivo gestionar la información de las conductas asistenciales y la información de los indicadores de la salud de la atención primaria y será el único sistema automatizado para la clasificación o codificación de los Indicadores y las Conductas Asistenciales definidos por el SNS para el nivel de APS. (17)

1.2 Tecnologías, técnicas, metodologías y software en las que sea poya la solución del problema.

Para el desarrollo del sistema se proponen tecnologías y herramientas definidas por el Departamento de Sistemas de Apoyo a la Salud (SAS) del CESIM de la Facultad 7, que pertenece a la Universidad de las Ciencias Informáticas.

1.2.1 Internet

Es una interconexión de redes informáticas que permite a las computadoras conectadas comunicarse directamente. El término suele referirse a una interconexión en particular, de carácter planetario y abierto al público, que conecta redes informáticas de organismos oficiales, educativos y empresariales. También existen sistemas de redes más pequeños llamados Intranet, generalmente para el uso de una única organización. (18)

La tecnología de Internet es precursora de la llamada superautopista de la información, un objetivo teórico de las comunicaciones informáticas que permitiría proporcionar a colegios, bibliotecas, empresas y hogares acceso universal a una información de calidad que eduque, informe y entretenga. (19)

1.2.2 Aplicación Web

Aplicaciones que los usuarios pueden utilizar accediendo a un servidor Web a través de Internet o de una Intranet mediante un navegador. Es decir, es una aplicación de software que se codifica en un lenguaje soportado por los navegadores Web en la que se confía la ejecución al navegador. Las aplicaciones Web son populares debido a la facilidad para actualizar y mantenerlas sin tener que distribuir e instalar software a miles de usuarios. Algunos ejemplos son los webmails, weblog o tiendas en línea. Una página Web puede contener elementos que permiten la comunicación activa entre el usuario y la información, accediendo a los datos de modo interactivo, como completar y enviar formularios, participar en juegos, entre otras. (20)

1.2.3 Servidor Web

Un servidor Web es un programa que sirve datos en forma de páginas Web, hipertextos o páginas HTML (Hypertext Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos. La comunicación de estos datos entre cliente y servidor se hace por medio de un protocolo, concretamente del protocolo HTTP (Hypertext Transfer Protocol). Con esto, un servidor Web se mantiene a la espera de peticiones HTTP, que son ejecutadas por un cliente HTTP; lo que solemos conocer como un navegador Web. El cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página. El servidor se encarga de transferir el código de la página sin llevar a cabo ninguna interpretación de la misma. (21)

- **Servidor Web Apache**

Apache es un servidor de código abierto para plataformas Unix, Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPD 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que originalmente Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, a patchy server (un servidor "parcheado"). (22)

Apache es usado primariamente para enviar páginas Web estáticas y dinámicas en la World Wide Web (WWW). Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o características propias de este servidor web. Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. (23)

Microsoft Internet Information Services (IIS) es el principal competidor de Apache, así como Sun Java System Web Server de Sun Microsystems y un anfitrión de otras aplicaciones como Zeus Web Server. Algunos de los más grandes sitios web del mundo están ejecutándose sobre Apache. La capa frontal (frontend) del motor de búsqueda Google está basada en una versión modificada de Apache, denominada Google Web Server (GWS). Muchos proyectos de Wiki media también se ejecutan sobre servidores web Apache. (24)

1.2.4 Arquitectura de Software

Es un grupo de abstracciones y patrones que brindan un esquema de referencia útil y sirven de guía en el desarrollo de software dentro de un sistema informático. Estas indican la estructura, funcionamiento e interacción entre las partes del software. Toda arquitectura de software debe describir diversos aspectos del software. Generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. Es importante destacar que cada uno de ellos constituye una descripción parcial de una misma arquitectura y es deseable que exista cierto solapamiento entre ellos. Esto es así porque todas las vistas deben ser coherentes entre sí, dado que describen la misma cosa. (25)

- **Arquitectura Cliente-Servidor**

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras. La interacción Cliente-Servidor es el soporte de la mayor parte de la comunicación por redes. Ayuda a comprender las bases sobre las que están construidos los algoritmos distribuidos. Algunas de las características de la arquitectura Cliente-Servidor son: (26)

- Combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos a compartir. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras, Módem.

- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidades del disco e input-output devices.

- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.

- Existe una clara distinción de funciones basadas en el concepto de “servicio”, que se establece entre clientes y servidores.

- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a los recursos compartidos.

- Los clientes corresponden a procesos activos en cuanto a que son estos los que hacen peticiones de servicios. Estos últimos tienen un carácter pasivo, ya que esperan peticiones de los clientes.

- **Modelo Vista Controlador**

Es un estilo 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 estilo de llamada y retorno 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. La descripción de cada componente es la siguiente: (27)

- El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

- La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

- El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

1.2.5 HTML

HTML es el acrónimo de Hypertext Markup Language (Lenguaje de Marcado de Hipertexto), creado en 1989 por Tim Berners-Lee. HTML es un lenguaje de composición de documentos y especificación de ligas de hipertexto que define la sintaxis y coloca instrucciones especiales que no muestra el navegador, aunque sí le indica cómo desplegar el contenido del documento, incluyendo texto, imágenes y otros medios soportados. HTML también le indica cómo hacer un documento interactivo a través de ligas especiales de hipertexto, las cuales conectan diferentes documentos ya sea en su computadora o en otras así como otros recursos de Internet, como FTP y Gopher. (28)

La facilidad de su uso y la particularidad de no ser propiedad de nadie, hizo de HTML el sistema idóneo para compartir información a través de Internet. Inicialmente su intención era que las etiquetas fueran capaces de marcar la información de acuerdo a su significado, pero por diversos motivos los creadores de los navegadores web fueron añadiendo más etiquetas HTML, dirigidas a controlar la representación de la información contenida en el documento.

1.2.6 XML

Extensible Markup Language (XML) es un lenguaje muy simple, que representa una manera distinta de hacer las cosas, más avanzada, cuya principal novedad consiste en permitir compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. Así pues, el XML juega un papel importantísimo en este mundo actual, que tiende a la globalización y la compatibilidad entre los sistemas, ya que es la tecnología que permitirá compartir la información de una manera segura, fiable, fácil. Además, XML permite al programador y los soportes dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos, ya que algunas tareas tediosas como la validación de estos o el recorrido de las estructuras corre a cargo del lenguaje y está especificado por el estándar, de modo que el programador no tiene que preocuparse por ello. (29)

1.2.7 SOAP

En el núcleo de los servicios Web se encuentra el Protocolo Simple de Acceso a Datos (SOAP por sus siglas en inglés), que proporciona un mecanismo estándar de empaquetar mensajes. SOAP ha recibido

gran atención debido a que se pueden realizar Llamadas a Procedimiento Remoto (RPC por sus siglas en inglés) es decir, se pueden bien en cliente o servidor realizar peticiones mediante HTTP a un servidor web. Los mensajes tienen un formato determinado empleando XML para encapsular los parámetros de la petición. Algunas de sus ventajas son: (30)

- No está asociado con ningún lenguaje.
- No se encuentra fuertemente asociado a ningún protocolo de transporte.
- Aprovecha los estándares existentes en la industria.
- Permite la interoperabilidad entre múltiples entornos.

1.2.8 WSDL

El Lenguaje de Descripción de Servicios Web (WSDL por sus siglas en inglés) permite que un servicio y un cliente establezcan un acuerdo en lo que se refiere a los detalles de transporte de mensajes y su contenido, a través de un documento procesable por dispositivos. WSDL representa una especie de contrato entre el proveedor y el que solicita. WSDL especifica la sintaxis y los mecanismos de intercambio de mensajes. (31)

1.2.9 Servicios Web XML

Un servicio Web es un componente de software que se comunica con otras aplicaciones codificando los mensaje en XML y enviando estos mensaje a través de protocolos estándares de Internet tales como el Hypertext Transfer Protocol (HTTP). Intuitivamente un servicio Web es similar a un sitio web que no cuenta con un interfaz de usuario y que da servicio a las aplicaciones en vez de a las personas. Un servicio Web, en vez de obtener solicitudes desde el navegador y retornar páginas Web como respuesta, lo que hace es recibir solicitudes a través de un mensaje formateado en XML desde una aplicación, realiza una tarea y devuelve un mensaje de respuesta también formateado en XML. (32)

1.2.10 CSS 2

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. (33)

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados *"documentos semánticos"*). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para *marcar* los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos. Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página. (34)

1.2.11 JavaScript 1.1

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (35)

JavaScript puede funcionar como un procedimiento y un lenguaje orientado a objetos. Los objetos se crean mediante programación en JavaScript, fijando los métodos y propiedades que de otro modo los objetos vacíos en tiempo de ejecución, a diferencia de las definiciones de clases sintácticas comunes en lenguajes compilados como C++ y Java. Una vez que un objeto se ha construido puede ser utilizado como un modelo (o prototipo) para la creación de objetos similares. (36)

1.2.12 AJAX

El término AJAX es un acrónimo de *Asynchronous JavaScript + XML*, que se puede traducir como "JavaScript asíncrono + XML". Es una técnica de desarrollo web para crear aplicaciones interactivas mediante la combinación de tres tecnologías ya existentes. AJAX no es una tecnología en sí mismo. (37)

Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano. Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor. (38)

1.2.13 Ext JS 3.1

Ext JS es una librería JavaScript que permite construir aplicaciones complejas en Internet. Esta librería incluye: (39)

- Componentes Interfaz de Usuario (UI) del alto performance y personalizables.
- Modelo de componentes extensibles.
- Un API fácil de usar.
- Licencias Open Source y comerciales.

Ext JS encaja dentro de este esquema como un motor que permite crear Ricas Interfaces de Aplicaciones (RIA) mediante JavaScript. Si enmarcamos a Ext JS dentro del desarrollo RIA, éste sería el render de la aplicación que controla el cliente y que se encarga de enviar y obtener información del servicio. (40)

Ext JS permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java Swing, gracias a esto provee una experiencia consistente sobre

cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, entre otros). (41)

Usar un motor de render como Ext JS permite tener además estos beneficios:

- Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.
- Eficiencia de la red: El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

1.2.14 PHP 5.3

Lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+. (42)

PHP es un acrónimo recursivo que significa Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo la implementación principal es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre. Entre sus principales características están: (43)

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una Base de Datos.

- El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de Programación orientada a objetos.

1.2.15 Symfony 1.4

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. (44)

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. (45)

Características y Ventajas: (46)

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).

- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de *"convenir en vez de configurar"*, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo. Código fácil de leer que incluye comentarios y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

1.2.16 Netbeans IDE 6.9

El IDE NetBeans es un IDE - una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto de Software Libre y gratuito sin restricciones de uso. El NetBeans IDE es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Apache Ant, control de versiones y refactoring. (47)

1.2.17 PostgreSQL 8.3

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. PostgreSQL utiliza un modelo cliente/servidor y usa *multiprocesos* en vez de *multihilos* para

garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (48)

Características más generales de PostgreSQL: (49)

- Integridad referencial
- Replicación asíncrona / Streaming replication - Hot Standby
- Unicode
- Multi-Version Concurrency Control (MVCC)
- Múltiples métodos de autentificación
- Acceso encriptado vía SSL
- Completa documentación
- Licencia BSD
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.

1.2.18 pgAdmin 1.8

pgAdmin es una herramienta para la administración de bases de datos para PostgreSQL. La aplicación se puede utilizar en Linux, FreeBSD, Solaris, Mac OSX y Windows para administrar PostgreSQL en cualquier plataforma, así como las versiones comerciales y derivados de PostgreSQL como Postgres Plus Advanced Server y base de datos Greenplum. (50)

pgAdmin está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica es compatible con todas las características de PostgreSQL y facilita la administración. La aplicación también incluye un editor de resaltado de sintaxis SQL, un editor de código del lado del servidor, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor se puede hacer a través de TCP/IP o Unix Domain Sockets (en plataformas * nix), y puede ser encriptado SSL para la seguridad. No se requieren drivers adicionales para comunicarse con el servidor de base de datos. (51)

pgAdmin es desarrollado por una comunidad de expertos de PostgreSQL en todo el mundo y está disponible en más de una docena de idiomas. Es un software libre publicado bajo la licencia BSD.

1.2.19 RUP

El Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés) es una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable. (52)

RUP es una metodología de desarrollo de software que está basado en componentes e interfaces bien definidas, y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. (53)

RUP divide el proceso en cuatro fases: Concepción o Inicio, Elaboración, Construcción y Transición, cada una de ellas representa un ciclo de desarrollo en la vida de un producto de software. Además agrupa las actividades en grupos lógicos, definiéndose 9 flujos de trabajo principales, los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como flujos de apoyo, ellos son: Modelado del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, Instalación o Despliegue, Administración del proyecto, Administración de configuración y cambios y Ambiente. (54)

El ciclo de vida de RUP se caracteriza por: (55)

- Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

- Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

- Iterativo e Incremental: Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

1.2.20 UML 2.1

El Lenguaje de Modelado Unificado (UML por sus siglas en inglés), es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. (56)

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos.

Es importante recalcar que UML no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

En las versiones previas del UML, se hacía un fuerte hincapié en que UML no era un lenguaje de programación. Un modelo creado mediante UML no podía ejecutarse. En el UML 2.0, esta asunción cambió de manera drástica y se modificó el lenguaje, de manera tal que permitiera capturar mucho más comportamiento (behavior). De esta forma, se permitió la creación de herramientas que soporten la automatización y generación de código ejecutable, a partir de modelos UML. (57)

1.2.21 Enterprise Architect 7.1

Es una herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. EA es una herramienta multi-usuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad. El UML provee beneficios significativos para ayudar a construir modelos de sistemas de software rigurosos y donde es posible mantener la trazabilidad de manera consistente; Enterprise Architect soporta este proceso en un ambiente fácil de usar, rápido y flexible. Entre sus características está: (58)

- Crear elementos del modelo UML para un amplio alcance de objetivos.
- Ubicar esos elementos en diagramas y paquetes.

- Crear conectores entre elementos.
- Documentar los elementos que ha creado.
- Generar código para el software que está construyendo.
- Realizar ingeniería reversa del código existente en varios lenguajes.
- Importación/Exportación XMI 2.1.

En este capítulo se realizó un estudio de aplicaciones que gestionan nomencladores a nivel internacional y nacional sin encontrar soluciones capaces de brindar las funcionalidades necesarias. También se realiza un estudio e investigación de las tecnologías, arquitecturas, lenguajes, metodologías de desarrollo y herramientas a utilizar en todo el proceso de desarrollo del sistema.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se hace una descripción de las características del sistema, definiéndose además la propuesta del Sistema para la Gestión de Nomencladores. Se desarrolla el Modelo de Dominio donde se recogen, de forma más explícita, los procesos del negocio. Se especifican y analizan los requisitos funcionales y no funcionales permitiendo tener un mejor dominio del sistema. Se realiza el modelado del sistema y se describen brevemente algunos casos de uso, identificados a partir de los requerimientos funcionales y se especifican los actores que intervienen como parte del sistema en la realización de cada una de las actividades.

2.1 Modelo de Dominio

Un Modelo de Dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, en la tarea construcción del modelo de dominio, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física. Los modelos de dominio pueden utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo. Similares a los mapas mentales utilizados en el aprendizaje, el modelo de dominio es utilizado por el analista como un medio para comprender el sector industrial o de negocios al cual el sistema va a servir. (59)

2.1.1 Conceptos fundamentales

Para obtener una mejor visión del diagrama Modelo de Dominio, a continuación se proporciona un marco conceptual con las definiciones identificadas en la elaboración del desarrollo del software.

Nomenclador: Representa el contenedor de la información que se desea gestionar, está estructurado jerárquicamente por grupos y hojas.

Usuario: Representa una persona o individuo que interactúa con el sistema Nomenclador y el cual posee ciertos grados de accesibilidad. Encargado de la gestión de la información del sistema.

Administrador: Representa el usuario del sistema encargado de la configuración y la gestión de la información del sistema, así como de la construcción de su estructura jerárquica, es decir grupos y hojas.

Grupos: Representa la raíz de la estructura jerárquica de un nomenclador, conformado a su vez por grupos y hojas.

Campo: Representa los distintos atributos o características con los que puede contar un nomenclador.

Código: Constituye un indicador que representa únicamente a un nomenclador específico. Este se construye a partir de la unión de los respectivos códigos de los grupos y hojas del nomenclador.

Asociar: Operación que permite establecer si un campo será Agregado o Asignado a un nomenclador.

Agregado: Clasificación que se le da a un campo cuando este solo va a pertenecer a un elemento padre.

Asignado: Clasificación que se le da a un campo cuando este no va a pertenecer a un elemento padre y será heredado por sus hijos, formando así parte de estos últimos.

Información: Representa el nivel final de la estructura jerárquica de un nomenclador.

Estado: Este concepto posibilita conocer el estado actual de la información en el sistema.

Activo: Representa el estado activo de la información. La información está siendo usada en el sistema.

Pasivo: Representa el estado pasivo de la información. La información, aunque ha sido eliminada en el sistema, permanece físicamente en la base de datos.

2.1.2 Diagrama del Modelo de Dominio

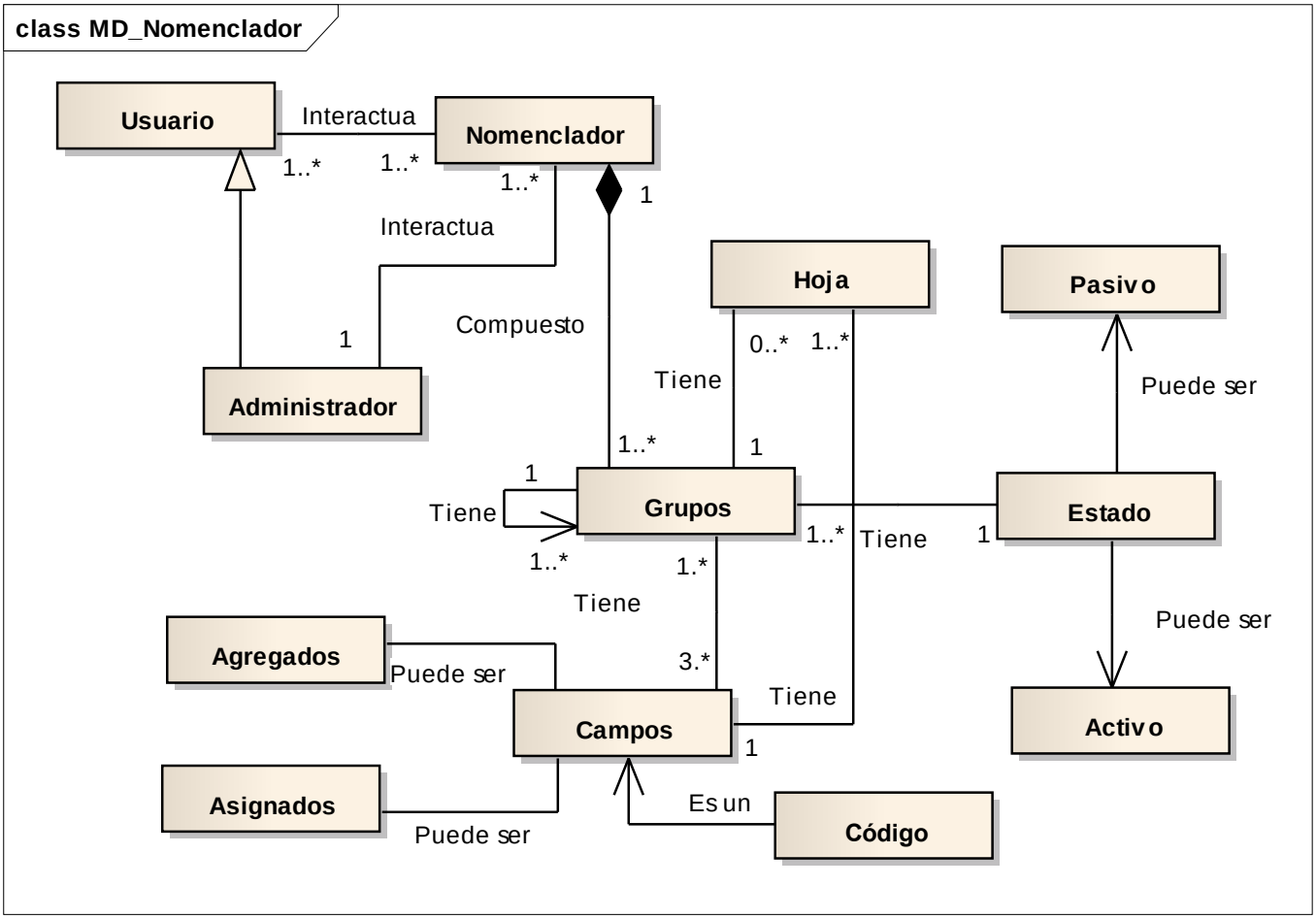


Figura1. Modelo de Dominio del Sistema

2.2 Propuesta del sistema

Sobre la base de análisis realizado al Módulo Nomenclador se propone el desarrollo de un Sistema para la Gestión de Nomencladores que permita nomenclar información común y poco variable en el tiempo y que posea características como la configuración y flexibilidad ante posibles cambios, por tanto este debe permitir:

- Gestionar los campos de los elementos nomenclados.

- Gestionar el código de cada nomenclador.
- Gestionar los grupos de los elementos a nomenclar.
- Asociar los campos a los nomencladores.
- Brindar servicios web que permitan el consumo de estos nomencladores por otras aplicaciones.
- Gestionar la información relacionada con cada nomenclador.
- Generar reportes de la información de un nomenclador específico o de varios de ellos.
- Gestionar los usuarios que interactúan con el sistema.

2.3 Requerimientos del software

Un requerimiento define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. Además son condiciones o capacidades que necesita un usuario para resolver un problema o lograr un objetivo. (60)

2.3.1 Requerimientos Funcionales (RF)

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir para darle solución al problema identificado. (61)

A continuación se muestra un listado con los requerimientos funcionales definidos para el sistema:

REQUERIMIENTOS FUNCIONALES	
RF1 Insertar Grupo	**RF12 Insertar Estructura de Código**
RF2 Modificar Información Grupo	**RF13 Modificar Estructura de Código**
RF3 Eliminar Grupo	**RF14 Buscar Estructura de Código**
RF4 Buscar Grupo	**RF15 Autenticación**
RF5 Listar Grupo	**RF16 Crear Usuario**
RF6 Insertar Campo	**RF17 Modificar Usuario**
RF7 Modificar Campo	**RF18 Eliminar Usuario**

RF8 Eliminar Campo	**RF19 Buscar Usuario**
RF9 Listar Campo	**RF20 Listar Usuario**
RF10 Buscar Campo	**RF21 Crear Reporte**
RF11 Asociar Campo	**RF22 Insertar Información Grupo**

Tabla 1. Requerimientos funcionales.

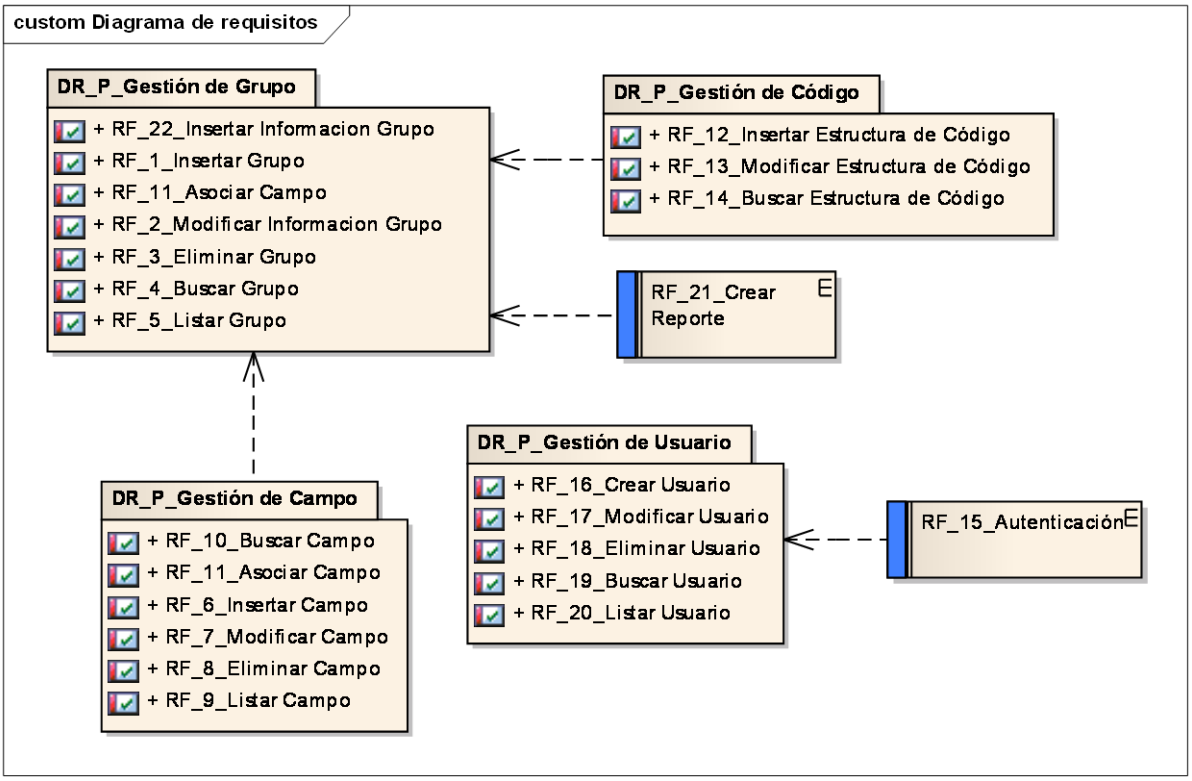


Figura 2. Diagrama de requisitos funcionales del sistema

2.3.2 Requerimientos No Funcionales (RNF)

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. (62)

2.3.2.1 Usabilidad

Estos requerimientos describen los niveles apropiados de usabilidad, dados los usuarios finales del producto, para ello debe revisarse la especificación de los perfiles de usuarios.

RNF1 El sistema sólo podrá ser utilizado por los usuarios registrados en él, ya sea ejerciendo el rol administrador o usuario.

RNF2 El usuario definido como administrador debe tener conocimientos básicos de informática.

RNF3 El sistema debe presentar un acceso fácil y rápido, para facilitar el uso del mismo por usuarios con pocos conocimientos en el campo de la informática.

RNF4 El usuario final deberá recibir una capacitación mínima de 7 días.

2.3.2.2 Seguridad

- **Confidencialidad**

RNF5 La información estará protegida contra accesos no autorizados utilizando mecanismos de autenticación propios del sistema.

RNF6 La autenticación será la primera acción del usuario en el sistema y consistirá en suministrar un nombre de usuario único y una contraseña que debe ser de conocimiento exclusivo de la persona que se autentica. Si el usuario autenticado no se encuentra registrado se debe reportar un error de acceso.

- **Integridad**

RNF7 La información podrá ser modificada solo por personal autorizado.

RNF8 La información manejada por el sistema será objeto de protección contra la corrupción y estados inconsistentes.

RNF9 Se harán validaciones de la información en el servidor contra ataques de inyección HTML o SQL.

- **Disponibilidad**

RNF10 La seguridad no implicará lentitud o retraso en la respuesta dada por el sistema, por lo que se debe minimizar y reducir el tiempo de respuesta, así como optimizar el código.

2.3.2.3 Soporte

RNF11 Una vez terminado el sistema se realizarán procesos de despliegue, capacitación y mantenimiento de software. El personal que trabaja con el sistema debe contar con el nivel técnico requerido mediante adiestramiento de servicio.

2.3.2.4 Requisitos para la documentación de usuarios en línea y ayuda del sistema

Describen los requisitos que provee a los usuarios y clientes de una documentación para la manipulación del sistema, así como para entender el mismo.

RNF12 Se dispondrá de un Manual de Usuario que indicará como interactuar con las funcionalidades del sistema.

RNF13 Se dispondrá de la documentación del sistema realizada con la metodología de desarrollo RUP.

2.3.2.5 Interfaz

- **Interfaces de usuario**

RNF14 El sistema debe tener una interfaz fácil de usar y amigable para que pueda ser utilizada sin mucho entrenamiento por el usuario.

RNF15 Las interfaces de usuario deben estar definidas y creadas siguiendo las pautas establecidas por el Departamento SAS. (Ejemplo: íconos, mensajes del sistema, etiquetado).

- **Interfaces Hardware común**

RNF16 Computadora personal Pentium IV.

RNF17 Se requiere tarjeta de red.

- **Interfaces Hardware para el cliente**

RNF18 Se requiere que tenga al menos 512 MB de memoria RAM, 1GB de disco duro, 512 MHz como mínimo en el cliente.

- **Interfaces Hardware para el servidor**

RNF19 Se requiere que tenga al menos 1GB de memoria RAM y 100 GB de disco duro, 2 GHz como mínimo.

- **Restricciones de diseño**

RNF20 Se utilizará PostgreSQL versión 8.3.

RNF21 Se utilizará Apache versión 2.2 o superior para el servidor Web.

RNF22 Para el desarrollo con PHP 5.3 y el Framework Symfony 1.4 se utilizará el NetBeans 6.9.

RNF23 Se utilizará un servidor con el sistema operativo instalado Windows XP o superior, o con un sistema operativo GNU/Linux. Debian 5preferentemente.

RNF24 En las computadoras de los clientes se requiere de un navegador Web (Internet Explorer versión 8.0 o superior, Mozilla Firefox versión 3.0 o superior).

RNF25 La comunicación de las computadoras clientes con el servidor será a través de conexiones de una red WAN o LAN.

RNF26 Todos los componentes del sistema deben desarrollarse siguiendo el principio de bajo acoplamiento.

RFN27 Para diseño de las páginas se utilizará CCS, HTML, JavaScript 1.1 y Ext JS 3.1.

- **Interfaces de Comunicación**

RNF28 La comunicación con el sistema se hará a través de Servicios Web, si la aplicación que la consume está desplegada en otro servidor, si la aplicación se encuentra en el mismo servidor la comunicación se realizará a través de agentes de implementación.

2.3.2.6 Estándares Aplicables

RNF29 Para la implementación del sistema se deberá seguir los estándares de codificación y diseño definidos por el Departamento SAS.

2.4 Actores del sistema

Representan papeles que la gente (o dispositivos) juegan como impulsores del sistema. Definido más formalmente, un actor es algo que comunica con el sistema o producto y que es externo al sistema en sí mismo. (63)

A continuación se definen los actores del sistema:

ACTOR	DESCRIPCION
Administrador	Usuario que posee privilegios administrativos para el uso y configuración del sistema.
Usuario	Usuario del sistema encargado de llenar la información de los nomencladores.

Tabla 2. Descripción de los actores del sistema.

2.5 Casos de uso

Los casos de uso modelan el sistema desde el punto de vista del usuario y son creados durante la obtención de requisitos. El caso de uso describe la manera en que los actores interactúan con el sistema, además describen escenarios que serán percibidos de distinta forma por distintos actores. (64)

CASOS DE USO DEL SISTEMA	
CUS1 Insertar Grupo	**CUS10 Modificar Estructura de Código**
CUS2 Modificar Información Grupo	**CUS11 Autenticar**
CUS3 Eliminar Grupo	**CUS12 Mostrar Detalles**
CUS4 Insertar Campo	**CUS13 Reportar**
CUS5 Modificar Campo	**CUS14 Crear Usuario**
CUS6 Buscar Campo	**CUS15 Modificar Usuario**
CUS7 Eliminar Campo	**CUS16 Buscar Usuario**
CUS8 Asociar Campo	**CUS17 Eliminar Usuario**
CUS9 Crear Estructura de Código	**CUS18 Insertar Información Grupo**

Tabla 3.Casos de uso del sistema.

2.5.1 Diagrama de casos de uso del sistema

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa. (65)

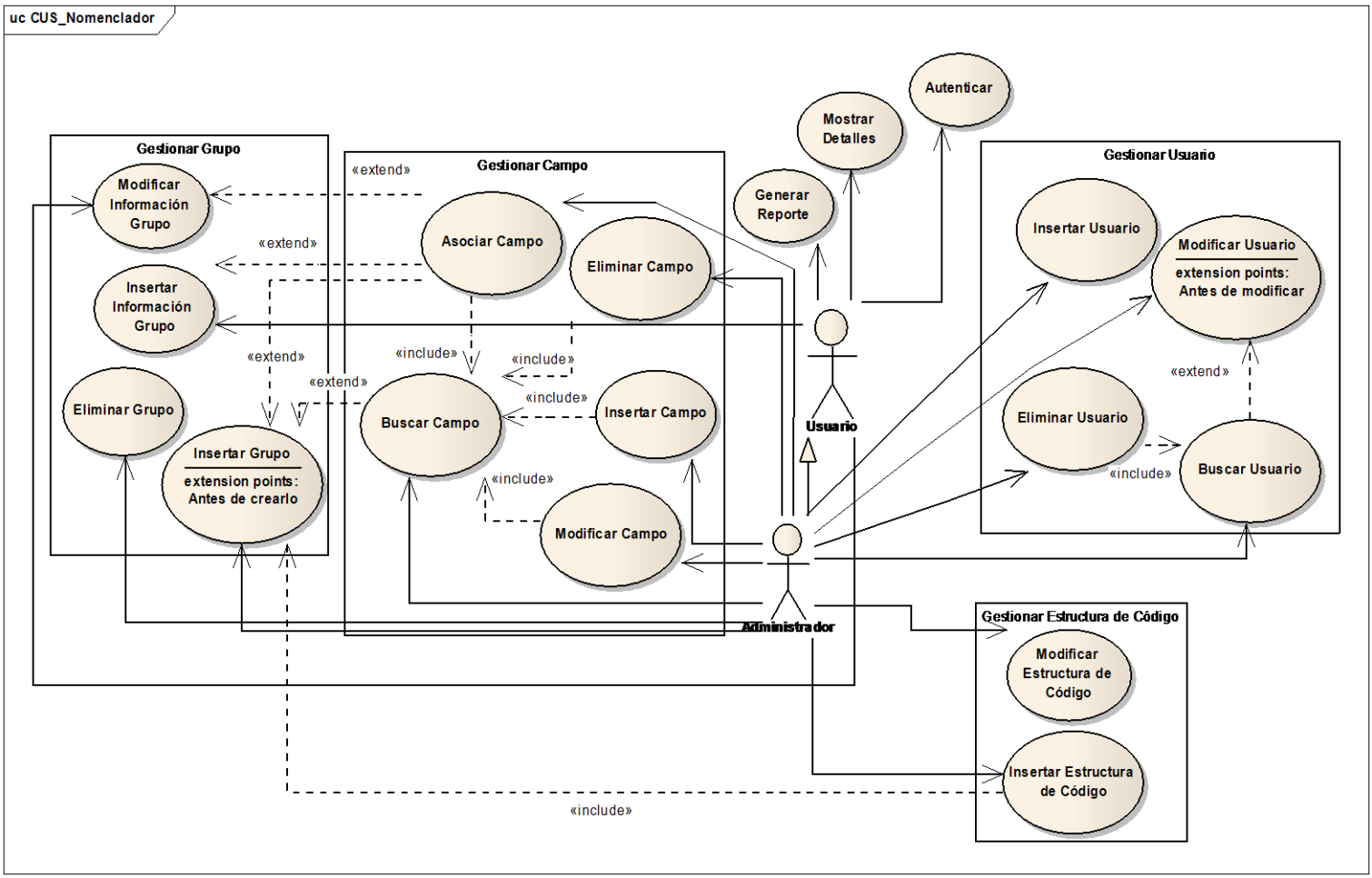


Figura 3. Diagrama de casos de uso del sistema

2.5.2 Descripción textual de casos de uso del sistema

A continuación se realiza la descripción de los casos de usos arquitectónicamente significativos. Las descripciones restantes serán mostradas en el Expediente de Proyecto**.**

Caso de Uso:	Insertar Campo
Actores:	**Administrador**
Resumen:	El caso de uso inicia cuando el actor accede a la opción Crear Campo, el sistema brinda la posibilidad de introducir y/o

	seleccionar los datos para crear el campo, el actor introduce los datos del campo, el sistema crea el campo, el caso de uso termina.
Precondiciones:	Debe buscar primero si existe el campo.
Referencias	RF6, RF9, RF10.
Prioridad	Útil

Tabla 4. Descripción textual CUS Insertar Campo.

Caso de Uso:	Modificar Campo.
Actores:	**Administrador**
Resumen:	El caso de uso inicia cuando el actor selecciona un campo y accede a la opción Editar Campo, el sistema muestra los datos del campo y brinda la posibilidad de cambiar sus valores ya sea introduciendo nuevos o seleccionando diferentes, el actor modifica los datos que necesita, el sistema actualiza los datos del campo, el caso de uso termina.
Precondiciones:	Debe seleccionar primero el campo a editar.
Referencias	RF7, RF9, RF10.
Prioridad	Útil

Tabla 5. Descripción textual CUS Modificar Campo.

Caso de Uso:	Crear Estructura de Código
Actores:	**Administrador**
Resumen:	El caso de uso inicia cuando el actor accede a la opción Adicionar Nomenclador, el sistema brinda la posibilidad de introducir y/o seleccionar los datos para crear la Estructura de Código, el actor introduce los datos de la Estructura de Código, el sistema crea la

	Estructura de Código, el caso de uso termina.
Precondiciones:	El Administrador debe estar previamente autenticado.
Referencias	RF12
Prioridad	Crítico

Tabla 6. Descripción textual CUS Crear Estructura de Código.

Caso de Uso:	Modificar Estructura de Código.
Actores:	**Administrador**
Resumen:	El caso de uso inicia cuando el actor selecciona una Estructura de Código, el sistema muestra los datos de la Estructura de Código y brinda la posibilidad de cambiar sus valores ya sea introduciendo nuevos o seleccionando diferentes, el actor modifica los datos que necesita, el sistema actualiza los datos de la Estructura de Código, el caso de uso termina.
Precondiciones:	El Administrador debe estar previamente autenticado.
Referencias	RF13, RF14.
Prioridad	Útil

Tabla 7. Descripción textual CUS Modificar Estructura de Código.

Con el desarrollo de este capítulo se obtuvo una descripción de la propuesta realizada, se describieron las características del sistema en términos de requerimientos funcionales y no funcionales. Se logró identificar los actores y los casos de uso del sistema, los cuales fueron modelados gráficamente y se hizo una descripción de los mismos. Debe mencionarse que este capítulo constituye la plataforma para el desarrollo exitoso de los sucesivos flujos de trabajo definidos por RUP, siendo estos: Diseño e Implementación.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

Uno de los objetivos de este capítulo es convertir en especificaciones del sistema los requisitos funcionales y no funcionales, contribuyendo a obtener una arquitectura sólida y estable para la futura implementación del software. Se descomponen los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo. Para lograr lo antes expuesto, se definen los elementos del diseño, se hace una descripción de las clases, realizando para ello artefactos como los Diagramas de Clases del Diseño y de Interacción, y la Descripción de la Arquitectura del Sistema.

3.1 Descripción de la arquitectura

Para el desarrollo de la aplicación se utilizó el Framework Symfony, que está basado en un patrón clásico del diseño web conocido como arquitectura MVC, que está formado por tres niveles: (66)

- El Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La Vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Esta arquitectura separa la lógica de negocio (el modelo) y la presentación (la vista), por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo: el tipo de gestor de bases de datos utilizado por la aplicación. (67) (Ver Anexo 1)

Atendiendo a las características propias del Framework Symfony, la aplicación cuenta en el Controlador, la Vista y el Modelo, con sus características propias. En las aplicaciones Web, el controlador, generalmente se encuentra muy cargado, pues es el encargado de tareas: como el manejo de las

peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación, entre otras actividades. (68)

Por tal motivo, el controlador en Symfony normalmente se divide en un Controlador Frontal, que es único para cada aplicación, y las Acciones, que incluyen el código específico del controlador de cada página. Una de las principales ventajas de utilizar un controlador frontal es que ofrece un punto de entrada único para toda la aplicación. Así, en caso de que sea necesario impedir el acceso a la aplicación, solamente es necesario editar el script correspondiente al controlador frontal. Si la aplicación no dispone de un controlador frontal, se debería modificar cada uno de los controladores. (69)

En la Vista también se puede aprovechar la separación del código, por lo que Symfony la separa en layouts (capa externa), templates (plantilla) y lógica de la vista; agrupando en las primeras la parte de esta, que permanece invariable para todas o parte de las páginas de la aplicación, mientras que las segundas, solo se encargan de visualizar las variables definidas en el controlador. En esta capa se utiliza además, la librería Ext JS que incluye un conjunto de componentes que aportan mejoras visuales y facilitan un mejor intercambio con el sistema.

La capa del modelo se puede dividir en 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 funciones escritas en el lenguaje propio del ORM Doctrine. Así, si se cambia de sistema gestor de base de datos, que en el caso particular del sistema es PostgreSQL 8.3, solamente es necesario actualizar la capa de abstracción de la base de datos, ya que el acceso a datos se realiza mediante el uso del motor de persistencia Doctrine, que abstrae al sistema del uso de cualquier sistema gestor de base de datos, mediante el uso de la programación orientada a objeto, haciendo posible tratar las tablas como objetos del sistema. (70)(Ver Anexo 2)

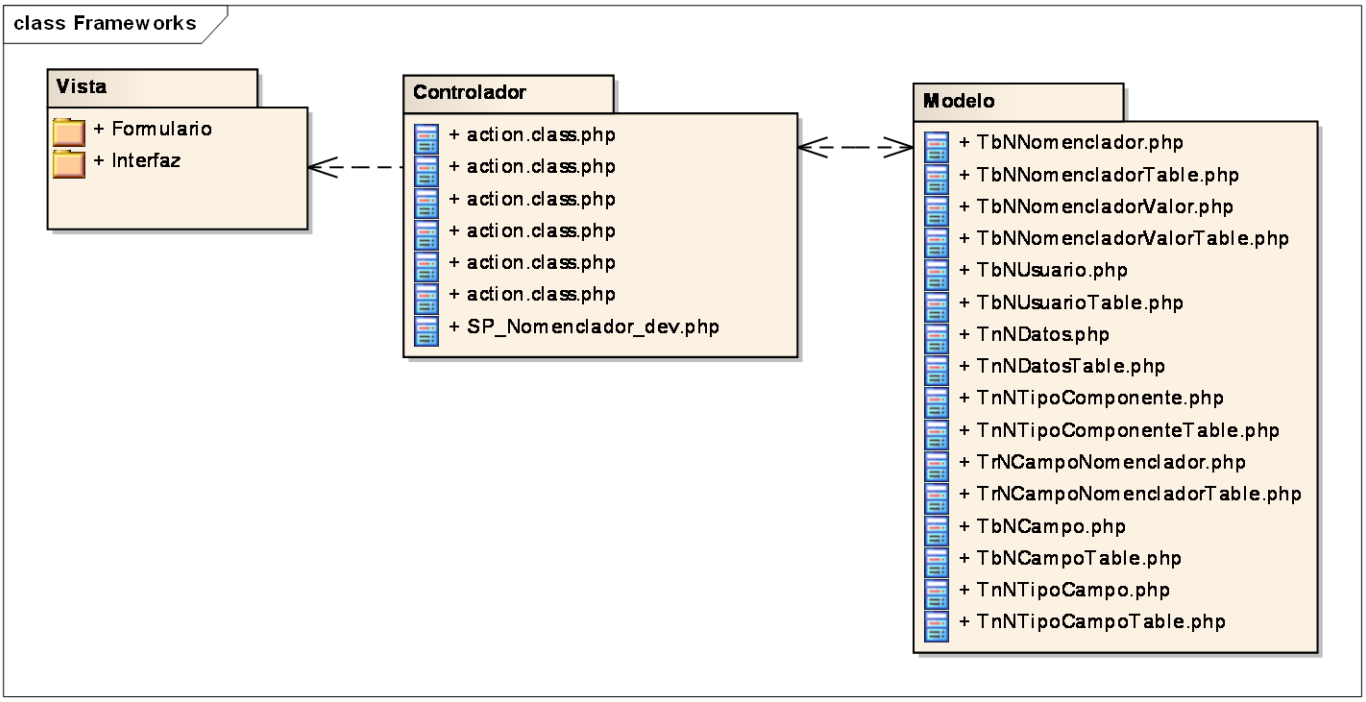


Figura 4. Representación de la arquitectura del sistema.

3.2 Modelo de análisis

El modelo de análisis nos ofrece una especificación más precisa de los requisitos, de modo que facilita su comprensión y en general su mantenimiento. Un modelo de análisis puede considerarse como una primera aproximación al modelo de diseño y es por tanto una entrada fundamental cuando se da forma al sistema en el diseño y la implementación. (71)

3.2.1 Diagramas de Clases del Análisis

Las clases de análisis siempre encajan en uno de tres estereotipos básicos: de interfaz, de control o de entidad. Cada estereotipo implica una semántica específica que constituye un método potente y consistente de identificar y describir las clases de análisis. (72)

A continuación se muestran algunos de los Diagramas de Clases del Análisis de los casos de uso arquitectónicamente significativos; los restantes pueden encontrarse en el Expediente de Proyecto.

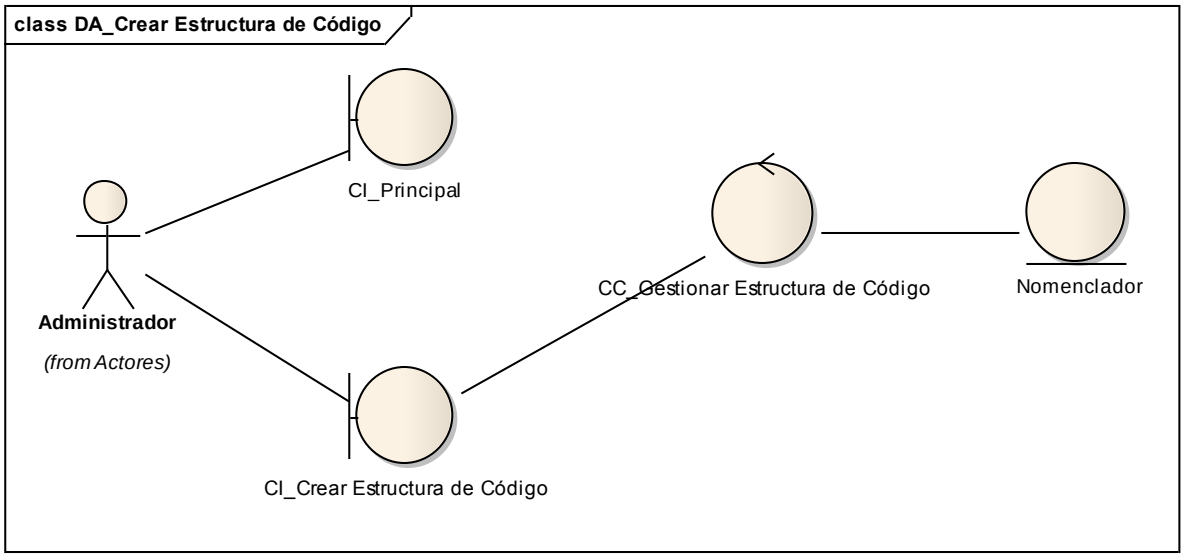


Figura 5. Diagrama de Clases de Análisis CUS_Crear Estructura de Código

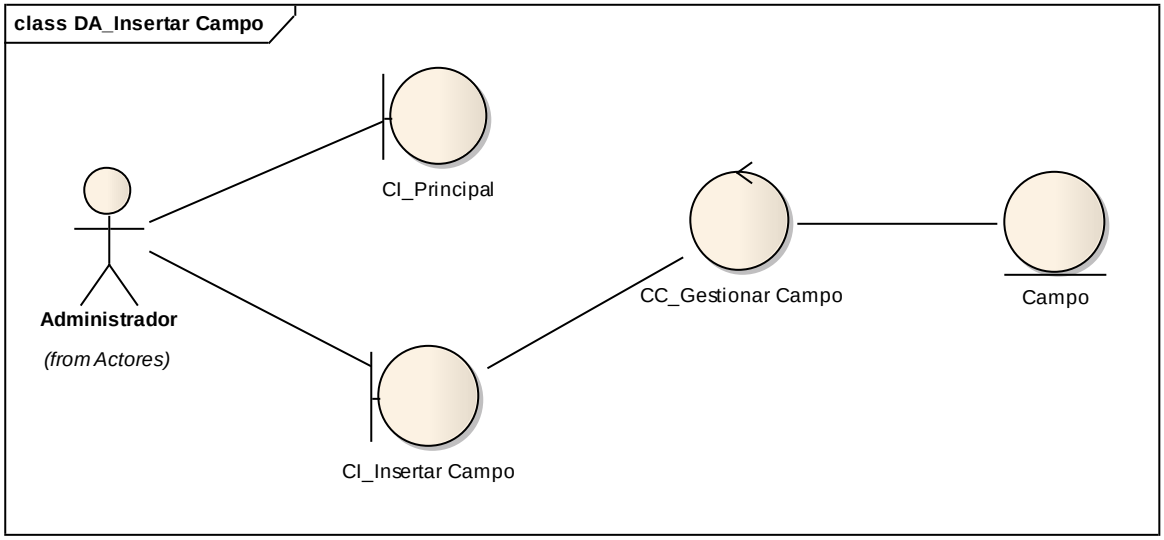


Figura 6. Diagrama de Clases de Análisis CUS_Insertar Campo

3.2.2 Diagramas de Comunicación

En los diagramas de colaboración mostramos las interacciones entre objetos creando enlaces entre ellos y añadiendo mensajes a esos enlaces. El nombre de un mensaje debería denotar el propósito del objeto invocante en la interacción con el objeto invocado. (73)

A continuación se muestran algunos de los diagramas de colaboración de casos de uso arquitectónicamente significativos; los restantes pueden encontrarse en el expediente de proyecto.

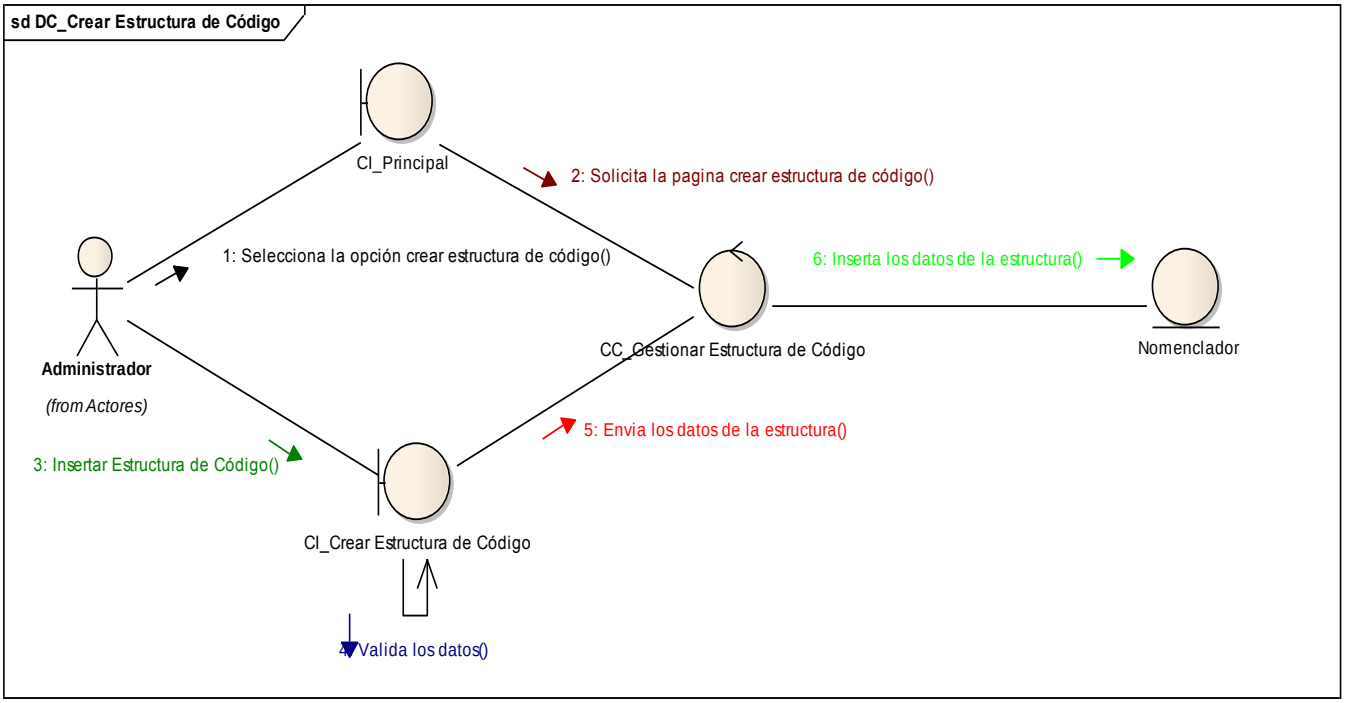


Figura9. Diagrama de Colaboración CUS_Crear Estructura de Código

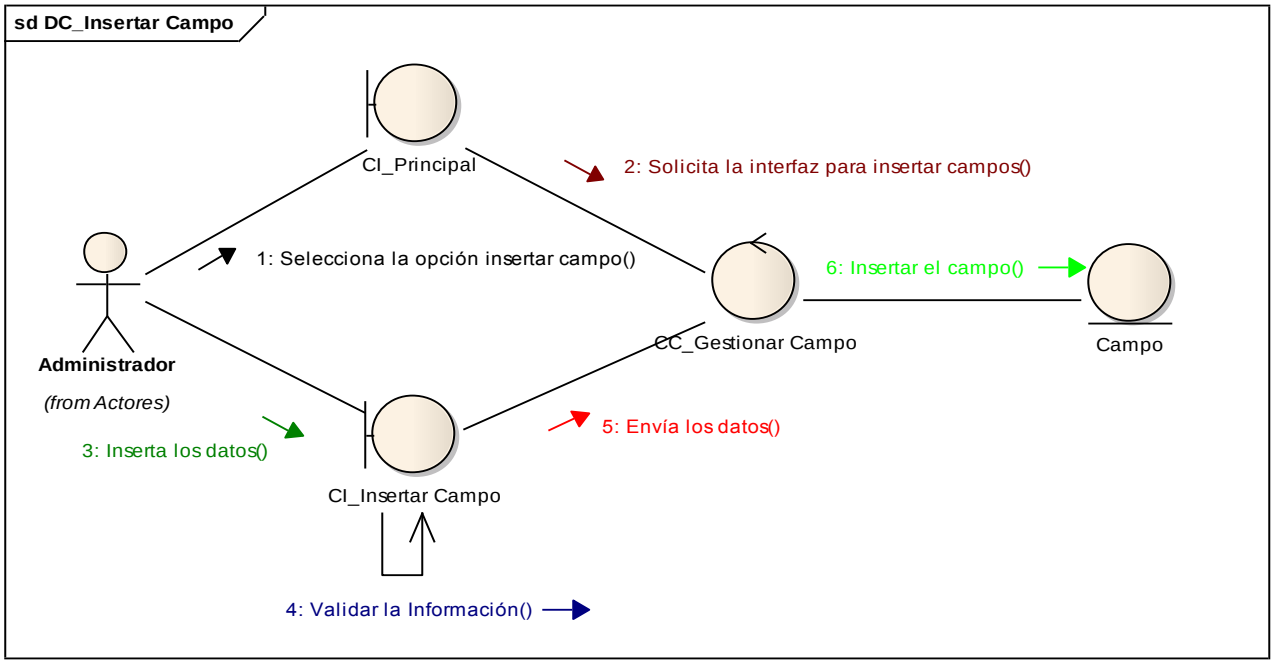


Figura10. Diagrama de Colaboración CUS_Insertar Campo

3.3 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales junto con otras restricciones relacionadas con el entorno de implementación tienen impacto en el sistema a considerar. Además, el modelo de diseño sirve de abstracción a la implementación del sistema y es, de ese modo, utilizado como entrada fundamental de las actividades de implementación. (74)

3.3.1 Definición de elementos del diseño

El sistema es una aplicación web que se modeló utilizando las clases UML estereotipadas “Server Page”, “Client Page” y “Form” empleadas para el código servidor, código cliente y formularios respectivamente, permitiendo además representar ficheros contenedores de sentencias script como por ejemplo PHP y JavaScript.

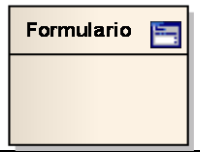
ESTEREOTIPOS WEB PARA LAS CLASES DEL DISEÑO	
ESTEREOTIPO	**DESCRIPCIÓN**
	**Server Page**: Representa una página web que tiene scripts que son ejecutados por el servidor. Estos scripts interactúan con recursos del servidor como bases de datos, lógica de negocio, sistemas externos y se encarga de construir (build) o generar el resultado HTML.
	**Client Page**: Es una página web con formato HTML y una mezcla de datos, presentación e incluso lógica. Las páginas clientes son representadas por los navegadores clientes, y pueden contener scripts que son interpretados por el navegador.
	**Form**: Es una colección de campos de entrada que forman parte de una página cliente. Los formularios envían sus datos al código servidor para ser procesados los pedidos (submit).

Tabla 8. Clases del diseño estereotipadas.

ESTEREOTIPOS PARA LAS RELACIONES ENTRE LAS CLASES	
link	Representa un apuntador desde una “client page” hacia una “client page” o “server page”.
submit	Esta relación siempre ocurre entre una “form” y una “server page”, la “server page” procesa los datos que la “form” le envía.
build	Se establece cuando la “server page” crea una “client page”. Una “server page” puede crear varias “client page”, pero una “client page” sólo puede ser creada por una sola “server page”.
redirect	Esta es también una relación unidireccional que indica que una página web redirecciona el procesamiento a otra página.

Tabla 9. Estereotipos para las relaciones entre las clases.

3.3.2 Diagramas de Clases del Diseño

Son clases conectadas a la realización de un caso de uso, mostrando sus clases principales, subsistemas y sus relaciones. De esta forma se puede guardar la pista de los elementos participantes en la realización de un caso de uso. (75)

A continuación se muestran algunos de los Diagramas de Clases del Diseño de los casos de uso arquitectónicamente significativos; los restantes pueden encontrarse en el Expediente de Proyecto.

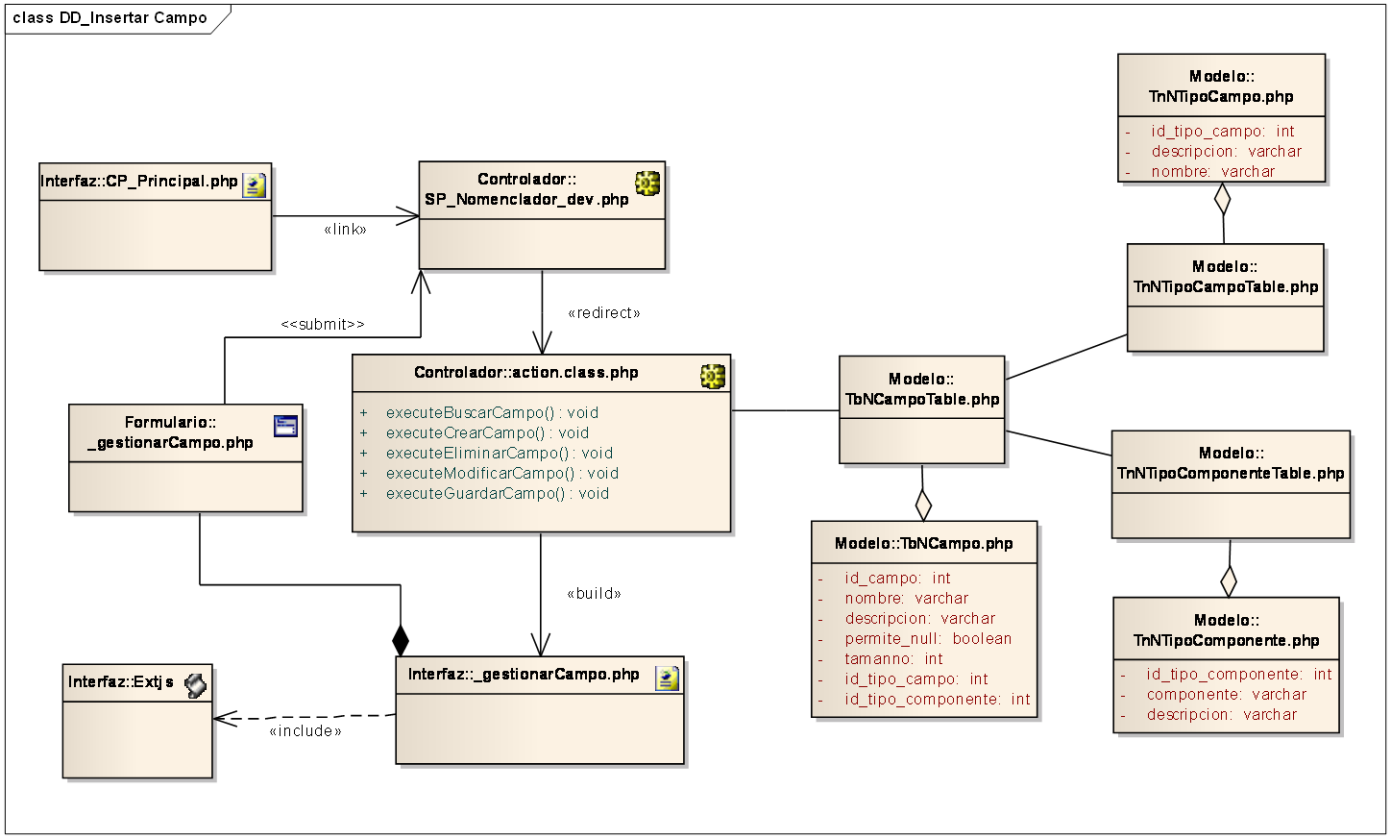


Figura 11. Diagrama de Clases del Diseño: DD_Insertar Campo.

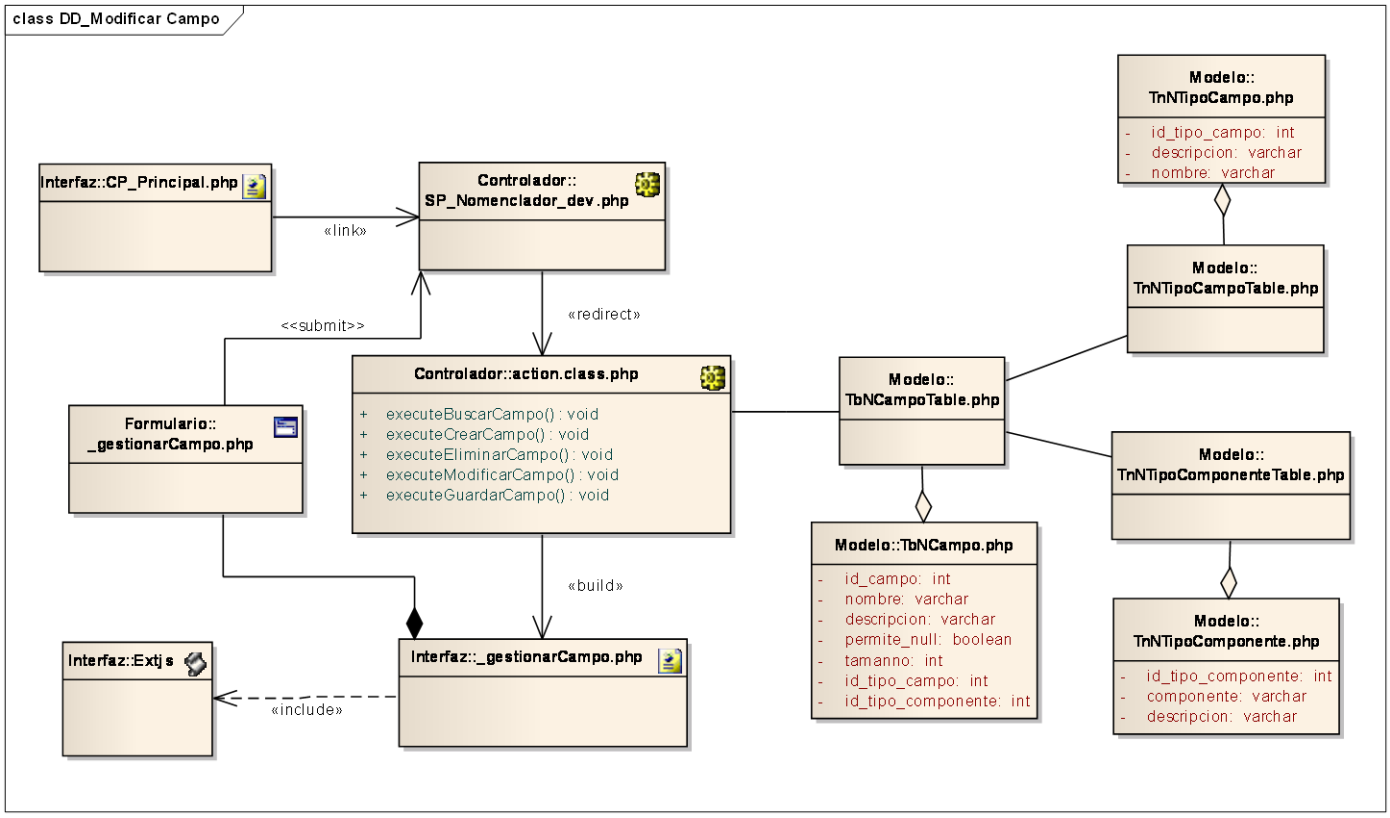


Figura 12. Diagrama de Clases del Diseño: DD_Modificar Campo.

3.3.3 Descripción de las clases

A continuación se describen algunas clases que han sido identificadas para la posterior implementación del sistema. De esta manera se poseerá un mejor entendimiento sobre el funcionamiento del mismo.

Nombre	**SP_Nomenclador_dev.php**
Tipo	Página servidora
Descripción	El controlador frontal es el único punto de entrada a la aplicación. Carga la configuración y determina la acción a ejecutarse. Se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Tabla 10. Descripción de la clase SP_Nomencladores_dev.php.

Nombre	**Extjs.php**
Tipo	Librería
Descripción	Es utilizada para la creación de las interfaces de usuario. Permite crear interfaces de usuario dinámicas y organizadas capaces de bridar todas las funcionalidades necesarias para la gestión de la información.

Tabla 11. Descripción de la clase Extjs.php.

Nombre	**Tb*NombreTabla*.php**
Tipo	Acceso a datos
Descripción	Esta clase hereda todos los métodos de las clases bases y no les afectan las modificaciones en el esquema permite empezar a programar desde el primer momento. La estructura de archivos creada permite personalizar y evolucionar el modelo. Es una clase objeto que representan un registro de la base de datos. Permiten acceder a las columnas de un registro y a los registros relacionados.

Tabla 12. Descripción de la clase TbTabla.php.

Nombre	**Tb*NombreTabla*Table.php**
Tipo	Acceso a datos
Descripción	Esta clase hereda todos los métodos de las clases bases y no les afectan las modificaciones en el esquema permite empezar a programar desde el primer momento. La estructura de archivos creada permite personalizar y evolucionar el modelo. Tienen métodos estáticos para trabajar con las tablas de la base de datos. Proporcionan los medios necesarios para obtener los registros de las tablas. Sus métodos devuelven normalmente un objeto o una colección de objetos de la clase objeto relacionada.

Tabla 13. Descripción de la clase Tb*NombreTabla*Table.php.

Nombre	**action.class.php**
Tipo	Página servidora
Descripción	Estas clases contienen toda la lógica del módulo al que pertenece. Las acciones son

	métodos de una clase que utilizan el modelo y definen variables para la vista. Cuando se realiza una petición web en una aplicación Symfony, la URL define una acción y los parámetros de la petición.

Tabla 14. Descripción de la clase action.class.php.

3.4 Modelo de datos

3.4.1 Descripción de algunas de las tablas de la base de datos

3.4.1.1 Descripción de la tb_n_campo

Nombre: tb_n_campo			
Descripción: Almacena información que permite la representación de los campos como componentes de la Interfaz Gráfica de Usuario (GUI por sus siglas en inglés), ejemplo: el campo Nombre se le asocia para representarlo (textbox), el tipo de dato del campo (string).			
Atributo	**Tipo**	**Nulo**	**Descripción**
id_campo(PK)	integer	No	Identificador único para el campo.
nombre	character(variable)	No	Nombre del campo.
descripción	character(variable)	No	Descripción del campo.
permite_null	boolean	No	Indica si el campo que se crea permite valores nulos.
tamaño	integer	Si	Indica el tamaño máximo del campo.
id_tipo_componente(FK)	integer	No	Identificador asociado a un componente para representar el campo en la interfaz de usuario, el cual está nomenclado en el esquema Nomencladores.tb_n_tipo_componente.
id_tipo_campo(FK)	integer	No	Identificador para representar el tipo de datos del campo, está nomenclado en el esquema Nomencladores.tb_n_tipo_campo.

Tabla 15. Descripción de tb_n_campo.

3.4.1.2 Descripción de la tb_n_nomenclador

Nombre: tb_n_nomenclador			
Descripción: Almacena información referente a los grupos de elementos, subgrupo de elementos y los elementos, el objetivo de esta tabla es nomenclar estructuras dinámicas del negocio.			
Atributo	**Tipo**	**Nulo**	**Descripción**
id_grupo(PK)	integer	No	Identificador único para las tuplas.
nombre	character(variable)	No	Nombre del elemento.
descripción	character(variable)	No	Descripción del elemento.
código	character(variable)	No	Código del elemento.
es_hoja	boolean	No	True es una hoja, false es un grupo.
expresión_regular	character(variable)	No	Estructura que va a tener el código

Tabla 16. Descripción de tb_n_nomenclador.

3.4.1.3 Descripción de la tr_n_campo_nomenclador

Nombre: tr_n_campo_nomenclador			
Descripción: Almacena información referente a los campos que se encuentran en tb_n_campo que pueden describir a un grupo, subgrupo o elemento, definido en la tabla tb_n_nomenclador.			
Atributo	**Tipo**	**Nulo**	**Descripción**
id_campo_nomenclador(PK)	integer	No	Identificador único para la asociación (campo, grupo).
id_grupo(FK)	integer	No	Identificador asociado al grupo.
id_campo(FK)	integer	No	Identificador del campo.
asignado	boolean	No	Especifica si el campo es agregado o asignado

Tabla 17. Descripción de tr_n_campo_nomenclador.

3.5 Diagrama de despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Muestra la configuración de los elementos de hardware (nodos) y cómo los elementos y artefactos del software se trazan en esos nodos.

La especificación detallada de cada nodo se encuentra en la planilla Modelo de Despliegue del proyecto Synta del Departamento SAS.

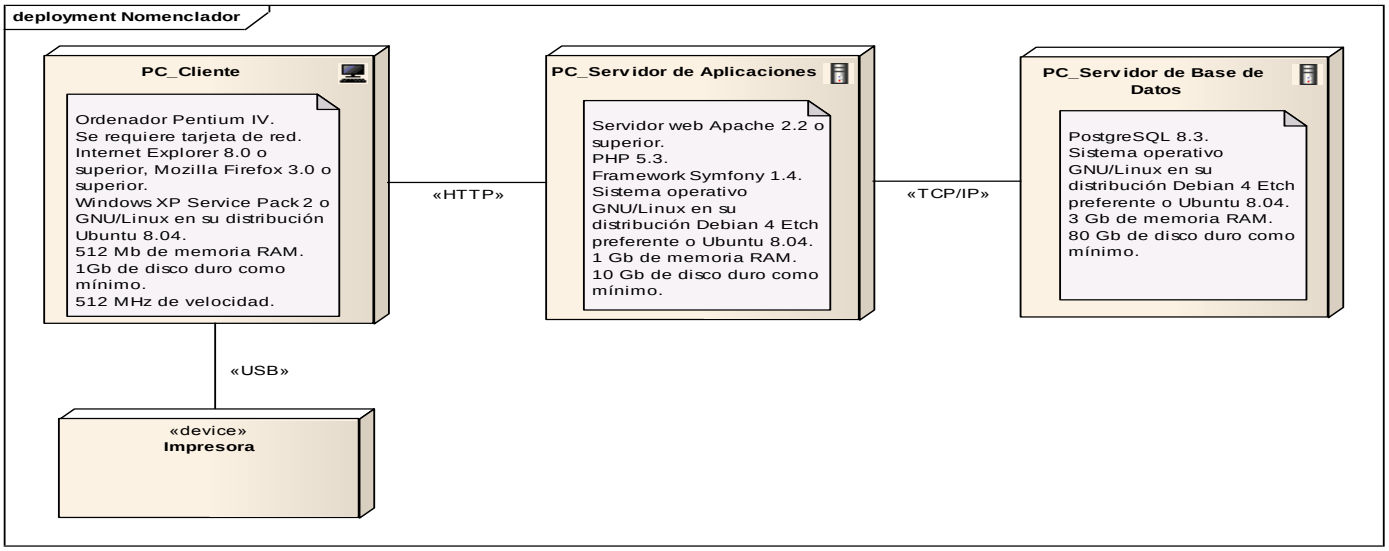


Figura 13. Diagrama de despliegue.

Con el desarrollo de este capítulo se logró modelar el sistema de forma que soporte los requisitos funcionales. Se realizó la estructuración de las clases de análisis, así como de los elementos del diseño de acuerdo a las características y estructura propuesta por el Framework Symfony. Se obtuvo el modelo de datos en el que se reflejan las tablas de la base de datos a utilizar para almacenar la información que será manejada en el sistema. Finalmente se con creó el diagrama de despliegue del sistema que muestra como este estará distribuido una vez que se despliegue.

CAPÍTULO 4: IMPLEMENTACIÓN

El presente capítulo tiene como objetivo la implementación de las clases y subsistemas obtenidos durante la etapa de diseño. El sistema es modelado en términos de componentes (ficheros de código fuente, scripts, ficheros de código binario, ejecutable y similar). Se incluye en el desarrollo del capítulo el diagrama de componentes y las estrategias de codificación utilizadas en la implementación del sistema.

4.1 Modelo de implementación

El Modelo de implementación describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código de fuentes y ejecutables. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros. (76)

4.2 Diagrama de componentes

Un Diagrama de componentes representa como un sistema es dividido en componentes y muestra las dependencias entre estos. Modelan la vista estática de un sistema y tienen un nivel de abstracción más elevado que un diagrama de clases, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Uno de los usos principales es que puede servir para ver qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema. (77)

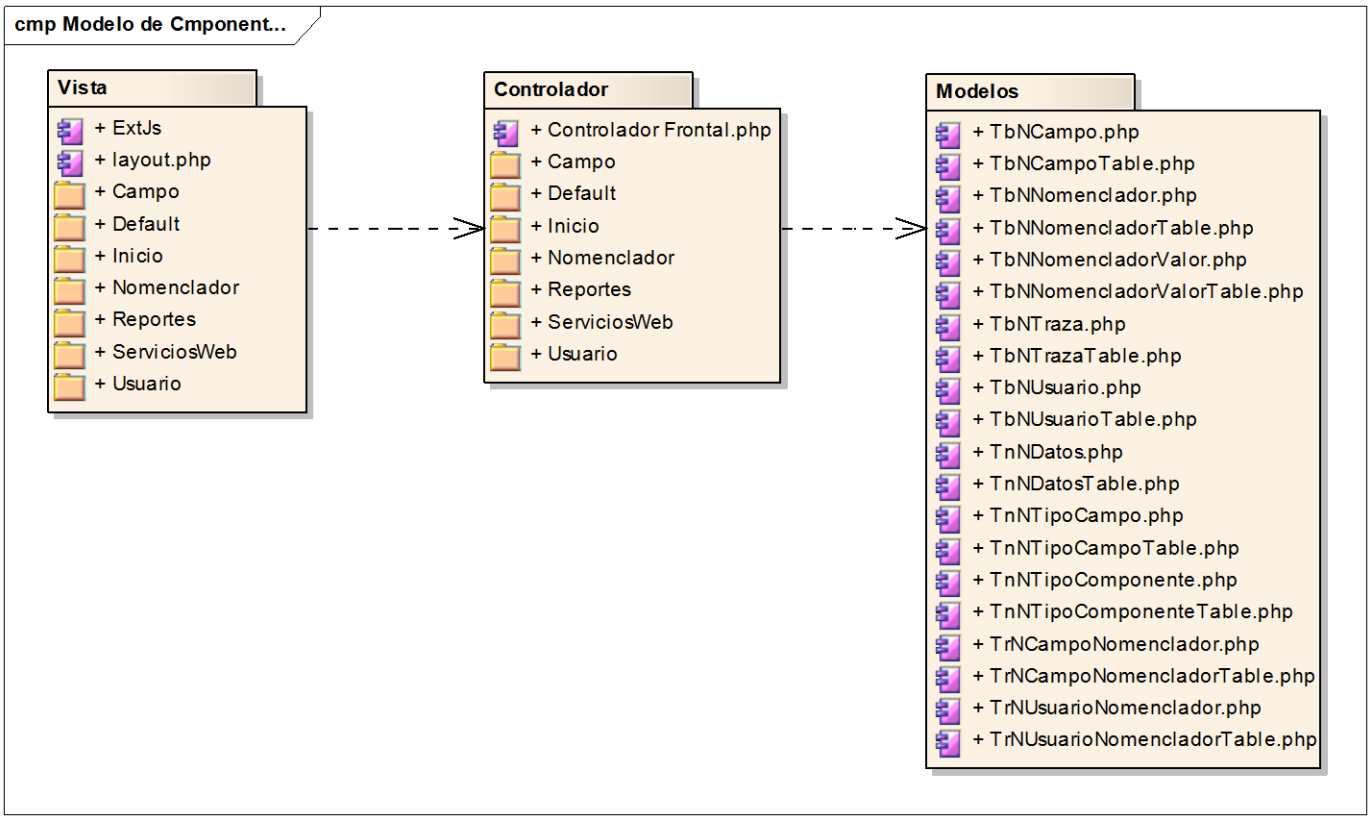


Figura 14. Diagrama general de Componentes.

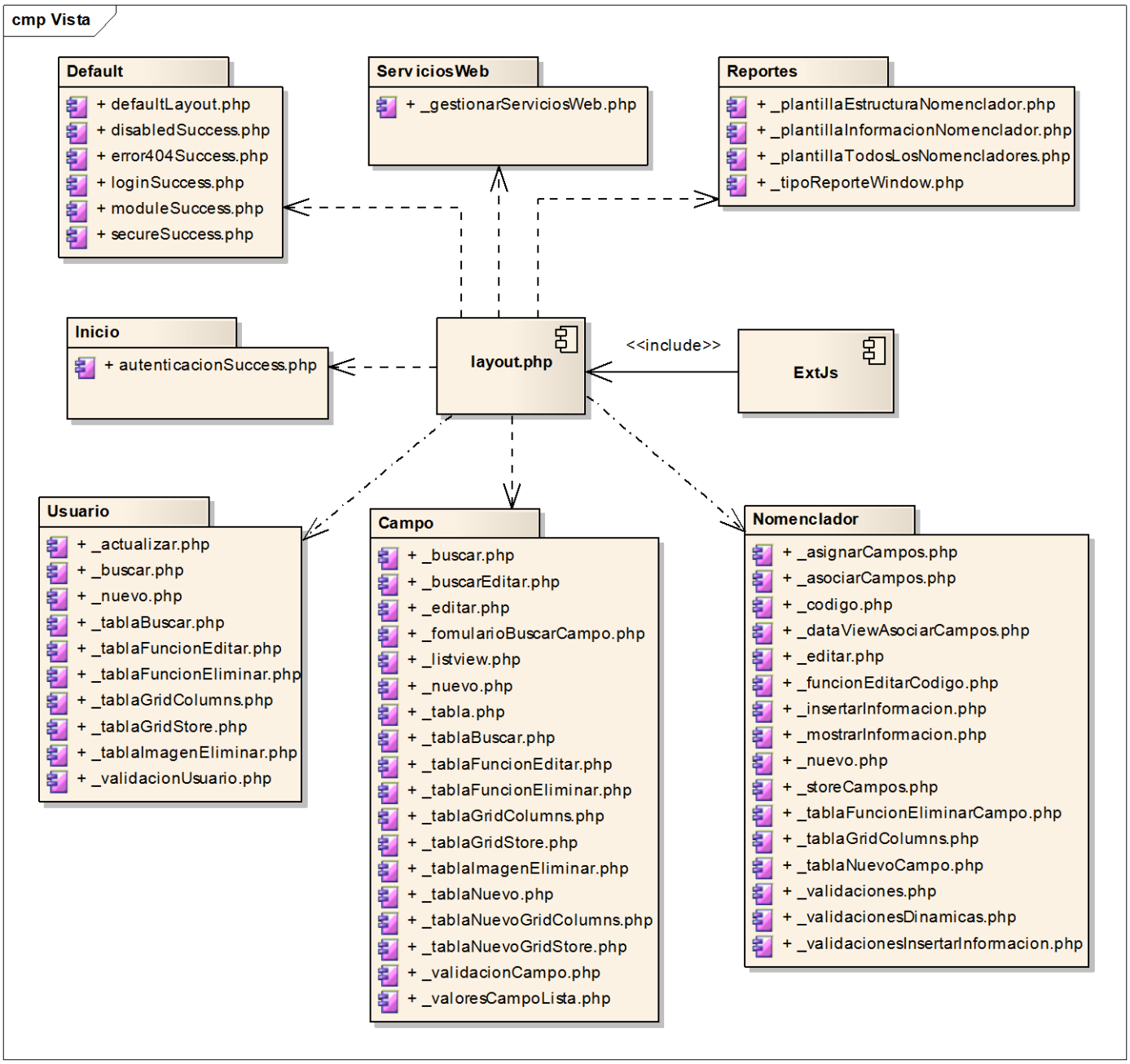


Figura 15. Diagrama de Componentes Vista.

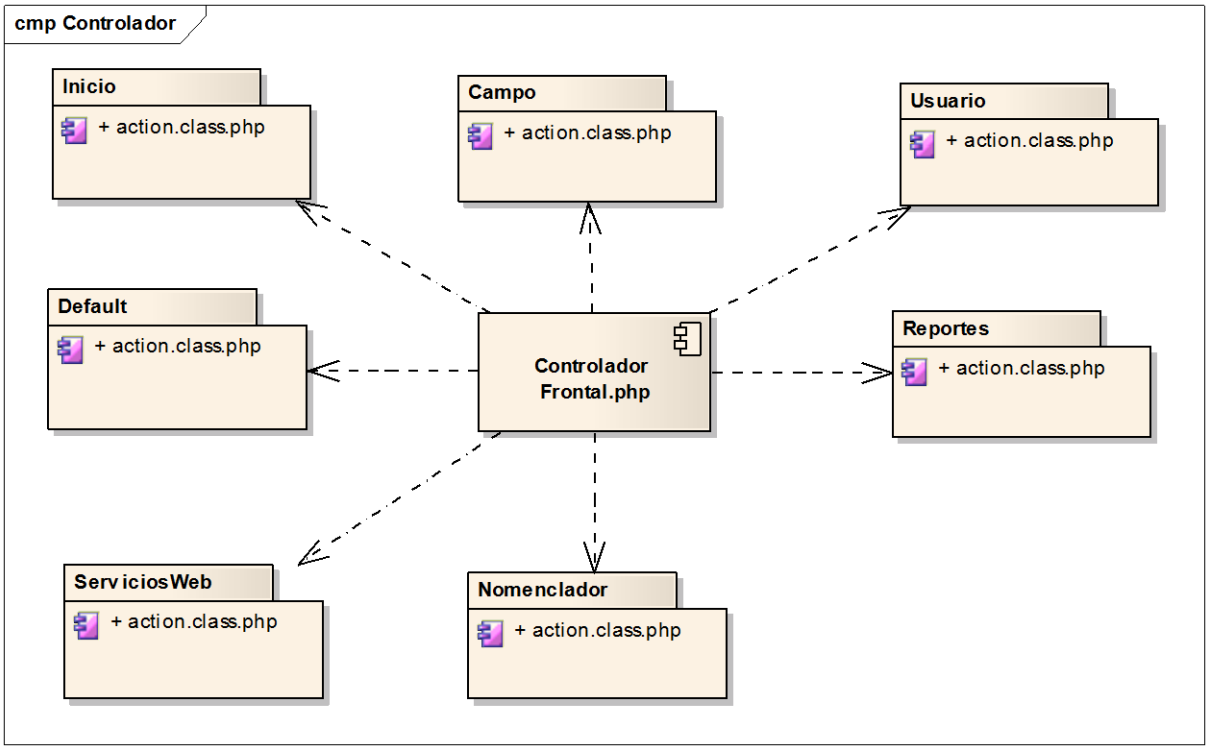


Figura 16. Diagrama de Componentes Controlador.

4.3 Descripción de las clases

A continuación serán expuestas algunas de las clases más importantes del sistema para lograr un mejor entendimiento sobre el funcionamiento del mismo.

Nombre: sp_Nomencladores.php
Descripción: Esta clase es el único punto de entrada a la aplicación, carga la configuración y determina la acción a ejecutarse. Además realiza las siguientes funciones: • Carga la clase de configuración del proyecto y las librerías de Symfony. • Crea la configuración de la aplicación y el contexto de Symfony. • Carga e inicializa las clases del núcleo del framework. • Carga la configuración. • Decodifica el Localizador Uniforme de Recurso (URL por sus siglas en inglés) de la petición para

determinar la acción a ejecutar y los parámetros de la petición. • Si la acción no existe, re direccionará a la acción del Error 404. • Activa los filtros (por ejemplo, si la petición necesita autenticación). • Ejecuta los filtros, primera pasada. • Ejecuta la acción y produce la vista. • Ejecuta los filtros, segunda pasada. • Muestra la respuesta.

Tabla 18. Descripción de la clase sp_Nomencladores.php.

Nombre: sp_myUser.php
Descripción: Esta clase es la encargada de gestionar todo lo relacionado con las sesiones de usuario así como los permisos y roles de cada usuario del sistema. Cuenta con las funcionalidades necesarias para gestionar la información relacionada con la seguridad y el acceso de los usuarios de determinadas páginas y funcionalidades.

Tabla 19. Descripción de la clase sp_myUser.php.

Clase controladora perteneciente al módulo Campo

Nombre: sp_actions.class.php
Descripción: Esta clase es la encargada de gestionar toda la información relacionada con los campos. Entre sus funciones se encuentra (Adicionar Campo, Modificar Campo, Buscar Campo, Eliminar Campo, AsignarValoresCamposMultivaluados). Esta clase tiene una gran importancia ya que a través de la creación de los campos se logra la flexibilidad del sistema en cuanto a la información de cada nomenclador.

Tabla 20. Descripción de la clase sp_actions.class.php del módulo Campo.

Clase controladora perteneciente al módulo Nomenclador

Nombre: sp_actions.class.php
Descripción: Esta clase es la encargada de gestionar toda la información de los nomencladores, así

como los campos asociados y asignados, su código y estructura jerárquica. Cuenta con las funcionalidades para la gestión de los nomencladores y funciones auxiliares necesarias para el funcionamiento correcto de los mismos.

Tabla 21. Descripción de la clase sp_actions.class.php del módulo Nomenclador.

Clase controladora perteneciente al módulo Servicios web

Nombre: sp_actions.class.php	
Descripción: Esta clase es la encargada de gestionar toda la información acerca de los servicios web que brinda el nomenclador a los distintos sistemas clientes. Cuenta con funcionalidades que devuelven un nomenclador dado su identificador, un nomenclador dado su nombre y todos los nomencladores hijos dado el identificador de un nomenclador padre. A continuación se muestran dos de las funcionalidades de los servicios web así como sus parámetros de entrada y salida.	
Nombre de la función: getNomencladorLikeNombre()	
Parámetros de entrada	
Descripción	**Tipo de dato**
sNombre	string
Parámetros de salida	
Descripción	**Tipo de dato**
Id	int
Nombre	string
Código	string
Descripción	string
ArrayCampos	array
Descripción de la funcionalidad	
La función devuelve los nomencladores con su información asociada, dado un nombre pasado como parámetro, en caso de no encontrar ningún nomenclador que coincida con la búsqueda devuelve un objeto vacío.	

```
public function executeGetNomencladorLikeNombre($request)
{
 if($this->isSoapRequest())
   {
      $parametros = $request->getParameter('ck_web_service_plugin.param');
      $nombre=$parametros[0];
      $oObjetos=Doctrine::getTable('TbNNomenclador')->getNomencladoresLikeNombre($nombre);
      $stdobject=new stdClassNomencladoresArray();

      $stdcampovalor=new stdCampoValor();
      $arraynomencladores=array();
      foreach($oObjetos as $objeto)
      {
        $stdnomencladores=new stdClassNomencladores();
        $stdnomencladores->nombre=$objeto->getNombre();
        $stdnomencladores->codigo=$objeto->getCodigo();
        $stdnomencladores->descripcion=$objeto->getDescripcion();
        $stdnomencladores->id=$objeto->getId();
        $campos=Doctrine::getTable('TbNNomenclador')->getCamposDadoNomenclador($objeto->getId());
        $stdnomencladores->ArrayCampos=$this->CampoValor($campos);
        $arraynomencladores[]=$stdnomencladores;
      }
      $stdobject->ArrayNomencladores=$arraynomencladores;
      $this->result=$stdobject;
      return sfView::SUCCESS;
   }
  else
  {
     return sfView::ERROR;
  }
}
```

Nombre de la función: getNomencladorDadoId()	
Parámetros de entrada	
Descripción	**Tipo de dato**
sId	int
Parámetros de salida	
Descripción	**Tipo de dato**
Id	int
Nombre	string
Código	string

Descripción	string
ArrayCampos	array

Descripción de la funcionalidad

La función devuelve el nomenclador que posee el identificador igual al entrado como parámetro devolviendo su información así como los campos asociados y asignados, en caso de no encontrar ningún nomenclador que coincida con la búsqueda devuelve un objeto vacío.

```
public function executeGetNomencladorDadoId($request){
    if($this->isSoapRequest()){
        $parametros = $request->getParameter('ck_web_service_plugin.param');
        $identificador = $parametros[0];
        $oStdclassResultado= new stdClassResultado();
        $oStdNomencladores = new stdClassNomencladores();
        $oStdCamposValor=new stdCampoValor();
        if(is_int($identificador))
         {
            $oObjeto=Doctrine::getTable('TbNNomenclador')->getNomencladorById($identificador);
            if($oObjeto!=null)
            {
                $campos=Doctrine::getTable('TbNNomenclador')->getCamposDadoNomenclador($oObjeto->getId());
                $oStdNomencladores->nombre=$oObjeto->getNombre();
                $oStdNomencladores->descripcion=$oObjeto->getDescripcion();
                $oStdNomencladores->codigo=$oObjeto->getCodigo();
                $oStdNomencladores->id=$oObjeto->getId();
                $oStdNomencladores->ArrayCampos= $this->CampoValor($campos);
                $oStdclassResultado->NomencladorRetorno = $oStdNomencladores;
                $this->result=$oStdclassResultado;
                return sfView::SUCCESS;
            }
            else
            {
              $this->result=$oStdclassResultado;
              return sfView::SUCCESS;
            }
        }
        else
        {
          return sfView::ERROR;
        }
    }
    else
    {
      return sfView::ERROR;
    }
}
```

A continuación se muestra el WSDL asociado al Servicio Web SWNomencladores.

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xm
  <wsdl:types xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://localhost/SyntaRepo/web/">
  </wsdl:types>
  <wsdl:portType name="SWNomencladoresPortType">
    <wsdl:operation name="GetNomencladorDadoId" parameterOrder="id">
      <wsdl:input message="tns:GetNomencladorDadoIdRequest"/>
      <wsdl:output message="tns:GetNomencladorDadoIdResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetNomencladorLikeNombre" parameterOrder="nombre">
      <wsdl:input message="tns:GetNomencladorLikeNombreRequest"/>
      <wsdl:output message="tns:GetNomencladorLikeNombreResponse"/>
    </wsdl:operation>
    <wsdl:operation name="GetHijosDadoIdPadre" parameterOrder="id">
  </wsdl:portType>
  <wsdl:binding xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="SWNomencladoresBinding" type="tns:SWNomencladoresPortType">
  <wsdl:message name="GetNomencladorDadoIdRequest">
    <wsdl:part name="id" type="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="GetNomencladorDadoIdResponse">
    <wsdl:part name="result" type="tns:stdClassResultado"/>
  </wsdl:message>
  <wsdl:message name="GetNomencladorLikeNombreRequest">
    <wsdl:part name="nombre" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="GetNomencladorLikeNombreResponse">
  <wsdl:message name="GetHijosDadoIdPadreRequest">
  <wsdl:message name="GetHijosDadoIdPadreResponse">
  <wsdl:service xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="SWNomencladoresService">
</wsdl:definitions>
```

Tabla 22. Descripción de la clase sp_actions.class.php del módulo Servicios Web.

4.4 Estrategias de codificación. Estándares y estilos a utilizar

En el desarrollo del sistema se hizo uso de íconos, mensajes del sistema y pautas de etiquetado, estándares de implementación y codificación, algunos de estos se enuncian a continuación:

Íconos, mensajes del sistema y pautas de etiquetado

El **Splash** a utilizar en el producto **Nomenclador** será el siguiente: NOMENCLADOR, y el nombre del producto debajo SISTEMA NOMENCLADOR DE INFORMACIÓN.

Figura 17. Splash del sistema.

Página de Autenticación: Al abrir la aplicación esta es la primera página con la que interactúa el usuario, los campos para el usuario y la contraseña son editables, al igual que el botón Aceptar.

Figura 18. Página de autenticación del sistema.

Encabezado: Contendrá en la parte izquierda el logo Nomenclador, nombre de la aplicación y del producto; mientras que en la parte superior derecha mostrará información del usuario como: el nombre. En la parte inferior derecha se mostrará un menú con los links principales definidos.

- Inicio: Link que permite regresar a la Pantalla de Bienvenida.
- Estilo: Para personalizar los colores del sistema.
- Idioma: Para personalizar los idiomas del sistema.

- Ayuda: Acceso a la ayuda del sistema.
- Salir: Acceso que permite cerrar la sesión.

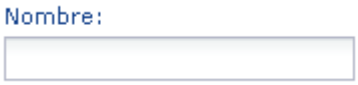
Figura 19. Encabezado del sistema.

Menú: Zona de la interfaz en la que se detallan las secciones o categorías en las que está dividida la información, propiamente son funcionalidades de la aplicación. A continuación se muestra una imagen de ejemplo:

Figura 20. Menú del sistema.

Diseño de Etiquetas:

- Todas las etiquetas estarán en el mismo idioma, serán palabras similares a las operaciones que los usuarios deseen llevar a cabo.
- Todas las etiquetas irán en azul.
- Todas las etiquetas estáticas terminarán con dos puntos (:).



- Las etiquetas mostrarán al lado derecho un ícono rojo, señalando que el campo es de llenado obligatorio.

- Las etiquetas de selección deberán llevar por defecto el texto <<Seleccione>>.

Algunas pautas de etiquetado:

Nombre	Acción	Íconos
Inicio	Permite al usuario ir a la pantalla de bienvenida.	
Estilo	Permite al usuario cambiar el color de la aplicación.	
Idioma	Permite al usuario cambiar el idioma del sistema de Español a Inglés.	
Ayuda	Permite al usuario visualizar la ayuda.	
Salir	Salir del sistema.	
Aceptar	Acepta una operación.	
Cancelar	Cancela una operación.	
Eliminar	Elimina un elemento.	
Reporte	Permite generar un reporte.	

Tabla 23. Pautas de etiquetado.

Mensajería, algunos ejemplos usados en el sistema:

- En las ventanas de Advertencia se utilizarán dos botones centrados en la parte inferior y contendrán los textos: “Si” y “No”.
- En los mensajes de creación de una nueva entidad, el sistema debe mostrar el siguiente mensaje: “Se adicionó (el ó la) <Entidad> correctamente.”
- En los mensajes de eliminar una entidad, el sistema debe mostrar el siguiente mensaje:

"(La o él) <Entidad> ha sido eliminada".

- En los mensajes de modificar una entidad, el sistema debe mostrar el siguiente mensaje: "Se modificó (el ó la) <Entidad> correctamente."

Identación o sangría ***Objetivo:*** Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento.		
Inicio y fin de bloque	Se recomienda dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones if, else, for, while, do while, switch, foreach.	
Aspectos Generales	El identado debe ser de dos espacios por bloque de código. No se debe usar el tabulador; ya que este puede variar según la PC o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deber estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción. Nunca colocar {en la línea de un código cualquiera, esto requiere una línea propia.	
Comentarios, separadores, líneas, espacios en blanco y márgenes. ***Objetivo:*** Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo leerlo una vez.		
Ubicación de comentarios	Al inicio de cada clase o función y al final de cada bloque de código.	Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los
Variables y constantes		
Apariencia de variables	Las variables tendrán un prefijo para el tipo de datos en minúscula.	El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, la cual identificara el tipo de datos al que se refiere (ver tabla 25), en caso de que sea un nombre compuesto se empleará notación CamellCasing**. Ejemplo: sNombrePaciente
Apariencia de constantes	Todas sus letras en mayúscula	Se deben declarar las constantes con todas sus letras en mayúscula.

Aspectos generales	Nombres de las variables y constantes	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.
Clases y Objeto ***Objetivo:*** Nombrar las clases e instancias de forma estándar para todas las aplicaciones.		
Apariencia de clases y objetos	Primera letra en minúscula	Los nombres de las clases deben comenzar con la primera letra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamellCasing*. Ejemplo: miClase(). Para el caso de las instancias se comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.
Apariencia de atributos	Primera letra en minúscula	El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, la cual estará en correspondencia al tipo de dato al que se refiere, en caso de que sea un nombre compuesto se empleará notación CamellCasing**.
Apariencia de las funciones	Primera letra en minúscula	Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que hace la función. Se empleará notación CamellCasing*. Ejemplo: function buscarUnidad(). Si son funciones que obtienen un dato se emplea el prefijo get y si fijan algún valor se emplea el prefijo set
Declaración de parámetro en funciones	Agrupados por tipos Poner los string 1 numéricos 2, además, agrupar según valores por defecto.	Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que estén agrupados por el tipo de dato que contienen, especificando el tipo de datos (Tabla 25).
Aspectos	Sobre las clases, los	El nombre empleado para las clases, objetos,

generales	objetos, los atributos y las funciones.	atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.
Bases de Datos, Tablas, esquemas y Campos		
Apariencia de la Base de Datos.	Las 2 primeras letras representan el tipo.	Los nombres de las Bases de Datos deben comenzar con el prefijo bd a continuación underscoard y luego el nombre completamente en minúscula, en caso de que sea un nombre compuesto se empleará notación. Los nombres serán cortos y descriptivos. Ejemplo: “bd_balancematerial”
Apariencia de las tablas	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para las tablas debe comenzar con el prefijo tb seguido de underscoard y luego debe escribirse todas las letras en minúscula, en caso de que sea un nombre compuesto se utilizara underscoard para separarlo. Ejemplo: “tb_producto”
Tablas que representen Relaciones	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tr seguido de underscoard y el nombre de será la concatenación del nombre de las dos tablas que la generaron separados por uderscoard todo en minúscula. Ejemplo: “tr_paciente_enfermedad”
Apariencia de los campos	Todas las letras en minúscula.	El nombre a emplear para los campos debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor. Ejemplo: “id_producto”
Nombre de los campos	En caso de identificadores.	Todos los campos identificadores van a comenzar con el identificador id seguido de underscoard y posteriormente el nombre del campo

		Ejemplo: “id_municipio”
Aspectos generales	Sobre las BD, vistas, tablas atributos y procedimientos.	El nombre empleado para las Bases de Datos, las vistas, las tablas, los campos y los procedimientos almacenados, deben permitir que con sólo leerlos se conozca el propósito de los mismos.

Tabla 24. Descripción de las estrategias de codificación.

- **Notación CamellCasing: Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula excepto la primera palabra que debe iniciar con minúscula. Ejemplo: notacionCamelCasing.

Tipo Datos	Prefijo	Ejemplo
Int	i	iCantPacientes
Float	f	fPesoPaciente
double	d	dPesoCarro
bool	b	bPacienteActivo
string	s	sNombrePaciente
char	c	cLetra
byte	b	bCantDiasPaciente
sbyte	sb	sbEdadPaciente
short	sh	shVariableShort
uint	ui	uiVariableUint
decimal	dc	dcVariableDecimal
Objetos	o	oPacienteHistorico
Objetos de tipo Struct	st	stUnaStruct

Tabla 25. Ejemplo de parámetros y variables.

Con el desarrollo de este capítulo se obtuvo el diagrama de componentes. También se realizó una breve descripción de las principales clases para un mejor entendimiento de cómo funciona el sistema. Finalmente se explicaron los estándares de codificación que se siguieron para la implementación del sistema.

CONCLUSIONES

Con la realización del presente trabajo de diploma se ha cumplido con el objetivo general propuesto, así como con las tareas de la investigación obteniéndose las siguientes conclusiones:

- El estudio de los principales sistemas nomencladores encontrados a nivel internacional y nacional evidenció que estos no poseen las características de ser configurables y flexibles ante la posible gestión de otro tipo de información que no esté contemplada en su negocio, por lo que no pueden ser utilizados para solucionar el problema planteado.

- Se realizó una valoración de las funcionalidades del Módulo Nomenclador del sistema alas BAP, para identificar las principales deficiencias que el mismo posee, siendo este el punto de partida para la especificación de los requerimientos que debía cumplir el nuevo sistema a desarrollar.

- Se construyó un sistema que cumple con la arquitectura y tecnologías definidas por el Departamento SAS, generando además todos los artefactos relacionados con los flujos definidos por la metodología de desarrollo RUP.

- Se obtuvo un sistema para la gestión de nomencladores que supera funcionalmente al Módulo Nomenclador del sistema alas BAP y que corrige las deficiencias que este posee, incorporando además un mecanismo flexible y configurable para la definición de la jerarquía de la información poco variable en el tiempo.

- La descripción de diferentes servicios Web XML que provee el sistema permite su consumo por otros, los cuales pudieran obviar el desarrollo de estos requerimientos agilizando su correspondiente proceso de construcción.

RECOMENDACIONES

Una vez concluido el trabajo el equipo de desarrollo recomienda:

- Incorporar nuevas funcionalidades al sistema:

    1. Permitir la comunicación entre nomencladores creados. Con esta funcionalidad se garantizará la reutilización de la información común para varios nomencladores, al permitir que un nomenclador pueda agregar a su estructura información de otro nomenclador, evitando que esta se repetida en la base de datos del sistema.

    2. Permitir la restricción de valores en posiciones específicas del código de un grupo. Esta funcionalidad dará la posibilidad de personalizar aún más una estructura de código al poderle definir valores fijos o rango de valores en posiciones específicas. Esto facilitará la posterior validación de la estructura de código, reduciendo la ocurrencia de errores y aumentando la fiabilidad de la información.

    3. Gestionar las trazas del sistema. Esta funcionalidad permitirá tener un registro de las operaciones que se realizan en el sistema y quien las ejecuta, manteniendo así un mejor control de la información perteneciente a los nomencladores del sistema.

REFERENCIAS BIBLIOGRÁFICAS

1. **Ministerio de la Informática y las Comunicaciones.** *Ministerio de la Informática y las Comunicaciones.* [En línea] [Citado el: 17 de Noviembre de 2010.] http://www.mic.gov.cu/sitiomic/servlet/hinfosoc.

2. Ídem a la 1. [En línea]

3. Ídem a la 1. [En línea]

4. **Morales, Ing.Annia Arencibia.** Informática 2011. [En línea] [Citado el: 25 de Enero de 2011.] http://www.informaticahabana.cu/sites/default/files/documentos/2011/60/p1778_06-12-18:16:32.doc.

5. *New Lici Salud.* [En línea] [Citado el: 18 de febrero de 2011.] http://www.obrasociales.com.ar/Prod_productos.asp?producto=Nomencladores.

6. **Deangelillo, Cristina.** *Federación Odontóloga de la Ciudad de Buenos Aires.* [En línea] [Citado el: 20 de enero de 2011.] http://www.fociba.org.ar/docs/Discurso%20Dra.%20Deangelillo.pdf.

7. Superintendencia Nacional de Aseguramiento en Salud. [En línea] [Citado el: 8 de noviembre de 2010.] http://www.seps.gob.pe/servicios/nomenclador/nomenclador_presentacion.aspx?opcion=12&seccion=178.

8. Ídem a la 7. [En línea]

9. Ídem a la 7. [En línea]

10. Ídem a la 7. [En línea]

11. **Francisco Gabriel Malbrán, Germán Eduardo Trouillet.** Nomenclador Cartográfico Para Personas con Discapacidad Visual. 2008.

12. **Díaz, Dr. Miguel Eusebio Marín.** *Informática en Salud 2009.* [En línea] 11 de 12 de 2008. [Citado el: 6 de Octubre de 2010.] http://informatica2009.sld.cu/Members/marin/nomencladores-medicos-nacionales-para-la-informatizacion-de-la-atencion-medica-en-el-sistema-nacional-de-salud/at_download/trabajo.

13. Ídem a la 12. [En línea]

14. Ídem a la 12. [En línea]

15. Ídem a la 12. [En línea]

16. Ídem a la 12. [En línea]

17. Ídem a la 12. [En línea]

18. **EcuRed.** EcuRed. [En línea] 9 de Marzo de 2011. [Citado el: 15 de Marzo de 2011.] http://www.ecured.cu/index.php/Internet.

19. Ídem a la 18. [En línea]

20. **HOOPING PUBLICIDAD S.L.** Hooping.net. [En línea] [Citado el: 3 de Noviembre de 2010.] http://www.hooping.net/glossary/aplicaciones-web-146.aspx.

21. **EcuRed.** EcuRed. [En línea] 4 de Marzo de 2011. [Citado el: 10 de Marzo de 2011.] http://www.ecured.cu/index.php/Servidores_Web.

22. **Nolf, Marcelo.** SAD. [En línea] 11 de Mayo de 2011. [Citado el: 16 de Mayo de 2011.] http://mistock.lcompras.biz/tallersoftware/1259-apache-una-alternativa-viable-para-el-servidor-web.

23. Ídem a la 22. [En línea]

24. Ídem a la 22. [En línea]

25. **EcuRed.** EcuRed. [En línea] 28 de Diciembre de 2010. [Citado el: 3 de Enero de 2011.] http://www.ecured.cu/index.php/Arquitectura_de_software.

26. —. EcuRed. [En línea] 14 de Septiembre de 2010. [Citado el: 3 de Noviembre de 2010.] http://www.ecured.cu/index.php/Arquitectura_Cliente_Servidor.

27. —. EcuRed. [En línea] 20 de Octubre de 2010. [Citado el: 3 de Noviembre de 2010.] http://www.ecured.cu/index.php/Patr%C3%B3n_Modelo_Vista_Controlador.

28. Chuck Musciano, Kennedy Bill. [En línea] [Citado el: 1 de noviembre de 2010.] http:// bibliodoc.uci.cu/pdf/reg01313.pdf.

29. **Alvarez, Miguel Angel.** desarrolloweb.com. [En línea] 13 de Junio de 2001. [Citado el: 25 de Noviembre de 2010.] http://www.desarrolloweb.com/articulos/449.php.

30. **González, Benjamín.** desarrolloweb.com. [En línea] 07 de julio de 2004. [Citado el: 27 de febrero de 2011.] http://www.desarrolloweb.com/articulos/1557.php.

31. Guía Breve de Servicios Web. [En línea] [Citado el: 26 de mayo de 2011.] http://www.w3c.es/divulgacion/guiasbreves/ServiciosWeb.

32. **González, Benjamín.** desarrolloweb.com. [En línea] 24 de Junio de 2004. [Citado el: 26 de Mayo de 2011.] http://www.desarrolloweb.com/articulos/1545.php.

33. librosweb.es. *librosweb.es.* [En línea] [Citado el: 6 de noviembre de 2010.] http://www.librosweb.es/css/capitulo1.html..

34. Ídem a la 33. [En línea]

35. Introducción a JavaScript. [En línea] [Citado el: 3 de noviembre de 2010.] http://www.librosweb.es/javascript/pdf/introduccion_javascript.pdf.

36. **Mozilla Developer Network.** MDN Doc Center. *MDN Doc Center.* [En línea] [Citado el: 3 de Noviembre de 2010.] https://developer.mozilla.org/en/About_JavaScript.

37. librosweb.es. *librosweb.es.* [En línea] [Citado el: 1 de noviembre de 2010.] http://www.librosweb.es/ajax/capitulo1.html.

38. Ídem a la 37. [En línea]

39. Desarrollo en Web . [En línea] [Citado el: 12 de noviembre de 2010.] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

40. Ídem a la 39. [En línea]

41. Ídem a la 39. [En línea]

42. **ProgramacionWeb.net.** ProgramacionWeb.net. [En línea] 06 de Septiembre de 2010. [Citado el: 18 de Enero de 2011.] http://www.programacionweb.net/articulos/articulo/?num=686.

43. Ídem a la 42. [En línea]

44. **Potencier, Fabien y Zaninotto, Francois.** librosweb.es. [En línea] 30 de diciembre de 2008. [Citado el: 14 de Marzo de 2011.] http://www.librosweb.es/symfony_1_2/.

45. Ídem a la 44. [En línea]

46. Ídem a la 44. [En línea]

47. **DosIdeas.** DosIdeas. [En línea] 04 de Octubre de 2010. [Citado el: 16 de Febrero de 2011.] http://www.dosideas.com/wiki/NetBeans.

48. **Development Group.** PostgreSQL. *PostgreSQL.* [En línea] [Citado el: 10 de octubre de 2010.] http://www.postgresql.org/about/.

49. Ídem a la 48. [En línea]

50. PgAdmin ArPug -PostgreSQL Argentina-Grupo de Usuarios. [En línea] [Citado el: 7 de diciembre de 2010.] http://www.arpug.com.ar/trac/wiki/PgAdmin.

51. PgAdmin III - Guía Ubuntu. [En línea] [Citado el: 7 de diciembre de 2010.]

52. GSInnova. [En línea] [Citado el: 4 de febrero de 2011.] http://www.rational.com.ar/herramientas/rup.html.

53. Proceso Unificado de Rational. [En línea] [Citado el: 4 de febrero de 2011.] http://www.buenastareas.com/ensayos/Proceso-Unificado-De-Rational/1333981.html.

54. Proceso Unificado de Desarrollo - Ecured. [En línea] [Citado el: 4 de febrero de 2011.] http://www.ecured.cu/index.php/RUP.

55. Ídem a la 54. [En línea]

56. Diccionario de Términos técnicos de Internet. *Glosario.net.* [En línea] 27 de octubre de 2006. [Citado el: 4 de noviembre de 2010.] http://tecnologia.glosario.net/terminos-tecnicos-internet/uml-1655.html.

57. Epidata consulting. *Introducción a UML 2.0 .* [En línea] [Citado el: 15 de noviembre de 2010.] http://www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=15.

58. **Sparx Systems.** Sparx Systems. [En línea] [Citado el: 21 de Noviembre de 2010.] http://www.sparxsystems.com.ar/products/ea.html.

59. **Garcerant, Iván.** Tecnología y Synergix. [En línea] 10 de Julio de 2008. [Citado el: 6 de Diciembre de 2010.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

60. **eumed.net.** Eumed.net. [En línea] [Citado el: 12 de Diciembre de 2010.] http://www.ecured.cu/index.php/Flujo_de_Trabajo_Requerimiento.

61. **Sommerville, Ian.** *Ingeniería del software.* Madrid, España : Pearson Educación, 2005. ISBN.

62. *Ídem a la 61.*

63. **Pressman, Roger S.** *Ingenieria del Software: Un Enfoque Prcictico.* 2001.

64. *Ídem a la 63.*

65. **Clikear.com.** Clikear.com. [En línea] [Citado el: 6 de Diciembre de 2010.] http://www.clikear.com/manuales/uml/diagramascasouso.aspx.

66. **Fabien Potencier, François Zaninotto.** librosweb.es. [En línea] 30 de Diciembre de 2008. [Citado el: 14 de Marzo de 2011.] http://www.librosweb.es/symfony_1_2/pdf/.

67. Ídem a la 66. [En línea]

68. Ídem a la 66. [En línea]

69. Ídem a la 66. [En línea]

70. Ídem a la 66. [En línea]

71. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación, S.A, 2000. 84-7829-036-2.

72. *Ídem a la 71.*

73. *Ídem a la 71.*

74. *Ídem a la 71.*

75. *Ídem a la 71.*

76. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software. .* España : Addison Wesley, 2004. ISBN.

77. Ing. en Informática. Instituto Tecnológico de Colina. [En línea] [Citado el: 25 de marzo de 2011.] http://inginformatica.tech.officelive.com/Programacion.aspx.

BIBLIOGRAFÍA

1. Alvarez, Miguel Angel. desarrolloweb.com. [En línea] 13 de Junio de 2001. [Citado el: 25 de Noviembre de 2010.] http://www.desarrolloweb.com/articulos/449.php.

2. Clikear.com. Clikear.com. [En línea] [Citado el: 6 de Diciembre de 2010.] http://www.clikear.com/manuales/uml/diagramascasouso.aspx.

3. Curso de Ext Js Framework [En Línea][Citado el: 10 de octubre de 2010]HTTP://www.quizzpot.com/2009/01/ext-js-framework/

4. Chuck Musciano, Kennedy Bill. [En línea] [Citado el: 1 de noviembre de 2010.] http:// bibliodoc.uci.cu/pdf/reg01313.pdf.

5. Deangelillo, Cristina. Federación Odontóloga de la Ciudad de Buenos Aires. [En línea] [Citado el: 20 de enero de 2011.] http://www.fociba.org.ar/docs/Discurso%20Dra.%20Deangelillo.pdf.

6. Desarrollo en Web . [En línea] [Citado el: 12 de noviembre de 2010.] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

7. Development Group. PostgreSQL. *PostgreSQL.* [En línea] [Citado el: 10 de octubre de 2010.] http://www.postgresql.org/about/.

8. Díaz, Dr. Miguel Eusebio Marín. *Informática en Salud 2009.* [En línea] 11 de 12 de 2008. [Citado el: 6 de Octubre de 2010.] http://informatica2009.sld.cu/Members/marin/nomencladores-medicos-nacionales-para-la-informatizacion-de-la-atencion-medica-en-el-sistema-nacional-de-salud/at_download/trabajo.

9. Diccionario de Términos técnicos de Internet. *Glosario.net.* [En línea] 27 de octubre de 2006. [Citado el: 4 de noviembre de 2010.] http://tecnologia.glosario.net/terminos-tecnicos-internet/uml-1655.html.

10. DosIdeas. DosIdeas. [En línea] 04 de Octubre de 2010. [Citado el: 16 de Febrero de 2011.] http://www.dosideas.com/wiki/NetBeans.

11. EcuRed. [En línea] 14 de Septiembre de 2010. [Citado el: 3 de Noviembre de 2010.] http://www.ecured.cu/index.php/Arquitectura_Cliente_Servidor.

12. EcuRed. [En línea] 20 de Octubre de 2010. [Citado el: 3 de Noviembre de 2010.] http://www.ecured.cu/index.php/Patr%C3%B3n_Modelo_Vista_Controlador.

13. EcuRed. EcuRed. [En línea] 28 de Diciembre de 2010. [Citado el: 3 de Enero de 2011.] http://www.ecured.cu/index.php/Arquitectura_de_software.

14. EcuRed. EcuRed. [En línea] 28 de Junio de 2010. [Citado el: 12 de Diciembre de 2010.]

15. EcuRed. EcuRed. [En línea] 4 de Marzo de 2011. [Citado el: 10 de Marzo de 2011.] http://www.ecured.cu/index.php/Servidores_Web.

16. EcuRed. EcuRed. [En línea] 9 de Marzo de 2011. [Citado el: 15 de Marzo de 2011.] http://www.ecured.cu/index.php/Internet.

17. Epidata consulting. *Introducción a UML 2.0 .* [En línea] [Citado el: 15 de noviembre de 2010.] http://www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=15.

18. eumed.net. Eumed.net. [En línea] [Citado el: 12 de Diciembre de 2010.] http://www.ecured.cu/index.php/Flujo_de_Trabajo_Requerimiento.

19. Fabien Potencier, François Zaninotto. librosweb.es. [En línea] 30 de Diciembre de 2008. [Citado el: 14 de Marzo de 2011.] http://www.librosweb.es/symfony_1_2/pdf/.

20. *Federacion Odontologa de la Ciudad de Buenos Aires.* [En línea] Cristina A. Deangelillo. [Citado el: 20 de enero de 2011.] http://www.fociba.org.ar/docs/Discurso%20Dra.%20Deangelillo.pdf.

21. Francisco Gabriel Malbrán, Germán Eduardo Trouillet. Nomenclador Cartográfico Para Personas con Discapacidad Visual. 2008.

22. Garcerant, Iván. Tecnología y Synergix. [En línea] 10 de Julio de 2008. [Citado el: 6 de Diciembre de 2010.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

23. González, Benjamín. desarrolloweb.com. [En línea] 07 de julio de 2004. [Citado el: 27 de febrero de 2011.] http://www.desarrolloweb.com/articulos/1557.php.

24. González, Benjamín. desarrolloweb.com. [En línea] 24 de Junio de 2004. [Citado el: 26 de Mayo de 2011.] http://www.desarrolloweb.com/articulos/1545.php.

25. GSInnova. [En línea] [Citado el: 4 de febrero de 2011.] http://www.rational.com.ar/herramientas/rup.html.

26. Guía Breve de Servicios Web. [En línea] [Citado el: 26 de mayo de 2011.] http://www.w3c.es/divulgacion/guiasbreves/ServiciosWeb.

27. HOOPING PUBLICIDAD S.L. Hooping.net. [En línea] [Citado el: 3 de Noviembre de 2010.] http://www.hooping.net/glossary/aplicaciones-web-146.aspx.

28. Ing. en Informática. Instituto Tecnológico de Colina. [En línea] [Citado el: 25 de marzo de 2011.] http://inginformatica.tech.officelive.com/Programacion.aspx.

29. Introducción a JavaScript. [En línea] [Citado el: 3 de noviembre de 2010.] http://www.librosweb.es/javascript/pdf/introduccion_javascript.pdf.

30. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación, S.A, 2000. 84-7829-036-2.

31. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso unificado de desarrollo de software. .* España : Addison Wesley, 2004. ISBN.

32. librosweb.es. *librosweb.es.* [En línea] [Citado el: 1 de noviembre de 2010.] http://www.librosweb.es/ajax/capitulo1.html.

33. librosweb.es. *librosweb.es.* [En línea] [Citado el: 6 de noviembre de 2010.] http://www.librosweb.es/css/capitulo1.html..

34. Ministerio de la Informática y las Comunicaciones. *Ministerio de la Informática y las Comunicaciones.* [En línea] [Citado el: 17 de Noviembre de 2010.] http://www.mic.gov.cu/sitiomic/servlet/hinfosoc.

35. Morales, Ing.Annia Arencibia. Informática 2011. [En línea] [Citado el: 25 de Enero de 2011.] http://www.informaticahabana.cu/sites/default/files/documentos/2011/60/p1778_06-12-18:16:32.doc.

36. Mozilla Developer Network. MDN Doc Center. *MDN Doc Center.* [En línea] [Citado el: 3 de Noviembre de 2010.] https://developer.mozilla.org/en/About_JavaScript.

37. *New Lici Salud.* [En línea] [Citado el: 18 de febrero de 2011.] http://www.obrasociales.com.ar/Prod_productos.asp?producto=Nomencladores.

38. Nolf, Marcelo. SAD. [En línea] 11 de Mayo de 2011. [Citado el: 16 de Mayo de 2011.] http://mistock.lcompras.biz/tallersoftware/1259-apache-una-alternativa-viable-para-el-servidor-web.

39. PgAdmin ArPug -PostgreSQL Argentina-Grupo de Usuarios. [En línea] [Citado el: 7 de diciembre de 2010.] http://www.arpug.com.ar/trac/wiki/PgAdmin.

40. PgAdmin III - Guía Ubuntu. [En línea] [Citado el: 7 de diciembre de 2010.]

41. Potencier, Fabien y Zaninotto, Francois. librosweb.es. [En línea] 30 de diciembre de 2008. [Citado el: 14 de Marzo de 2011.] http://www.librosweb.es/symfony_1_2/.

42. Pressman, Roger S. *Ingenieria del Software: Un Enfoque Prcictico.* 2001.

43. Proceso Unificado de Desarrollo - Ecured. [En línea] [Citado el: 4 de febrero de 2011.] http://www.ecured.cu/index.php/RUP.

44. Proceso Unificado de Rational. [En línea] [Citado el: 4 de febrero de 2011.] http://www.buenastareas.com/ensayos/Proceso-Unificado-De-Rational/1333981.html.

45. ProgramacionWeb.net. ProgramacionWeb.net. [En línea] 06 de Septiembre de 2010. [Citado el: 18 de Enero de 2011.] http://www.programacionweb.net/articulos/articulo/?num=686.

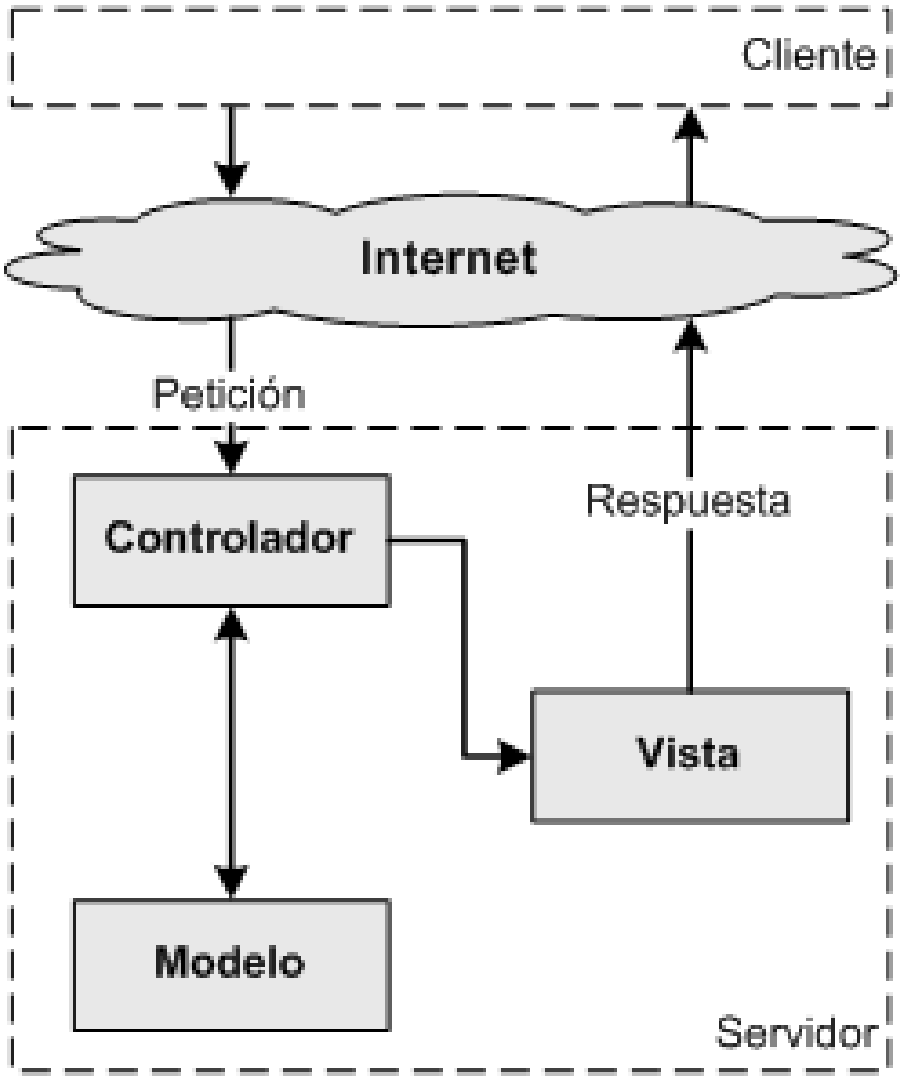
46. Sommerville, Ian. *Ingeniería del software.* Madrid, España : Pearson Educación, 2005. ISBN.

47. Sparx Systems. Sparx Systems. [En línea] [Citado el: 21 de Noviembre de 2010.] http://www.sparxsystems.com.ar/products/ea.html.

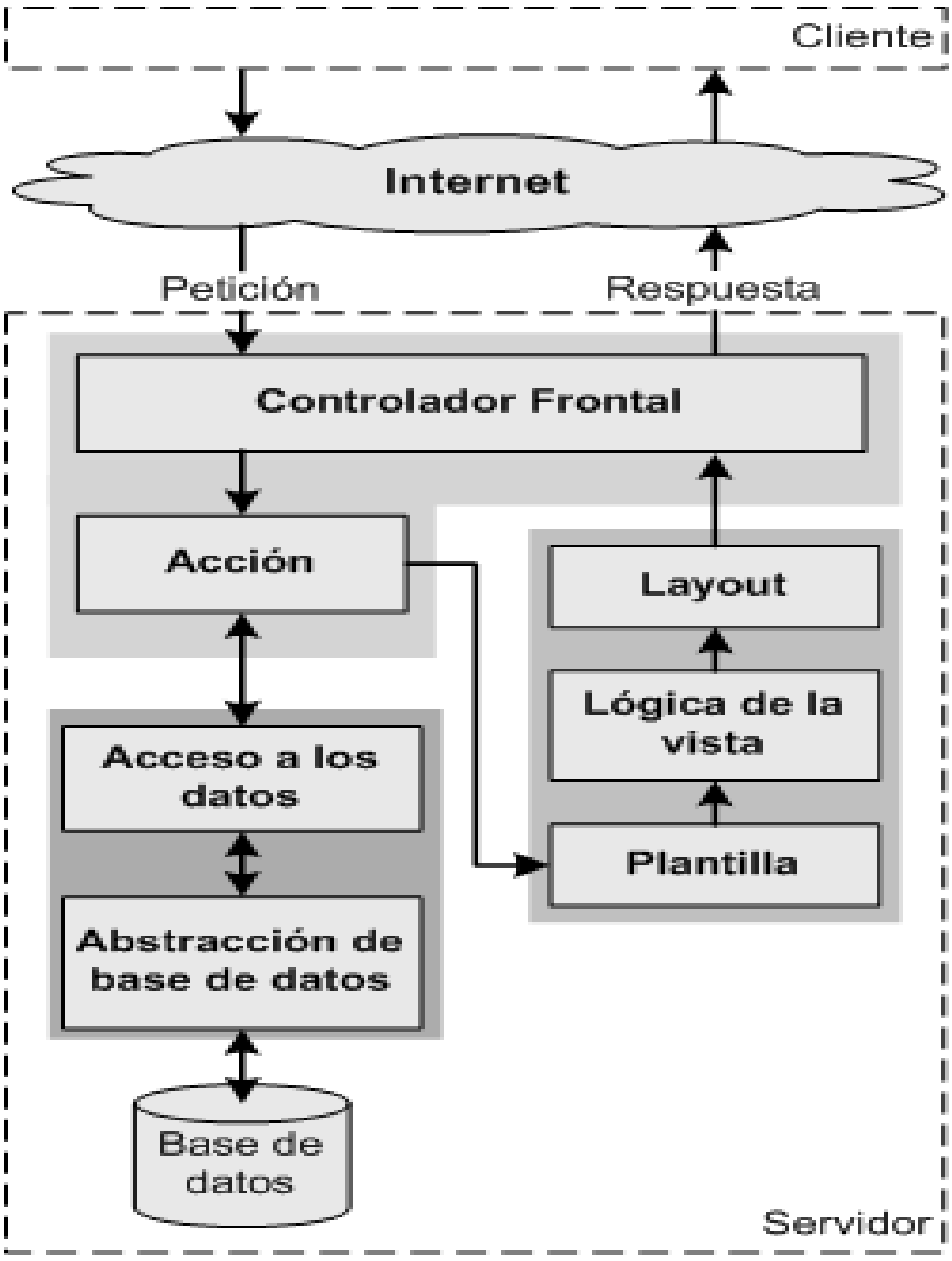
48. Superintendencia Nacional de Aseguramiento en Salud. [En línea] [Citado el: 8 de noviembre de 2010.] http://www.seps.gob.pe/servicios/nomenclador/nomenclador_presentacion.aspx?opcion=12&seccion=178.

ANEXOS

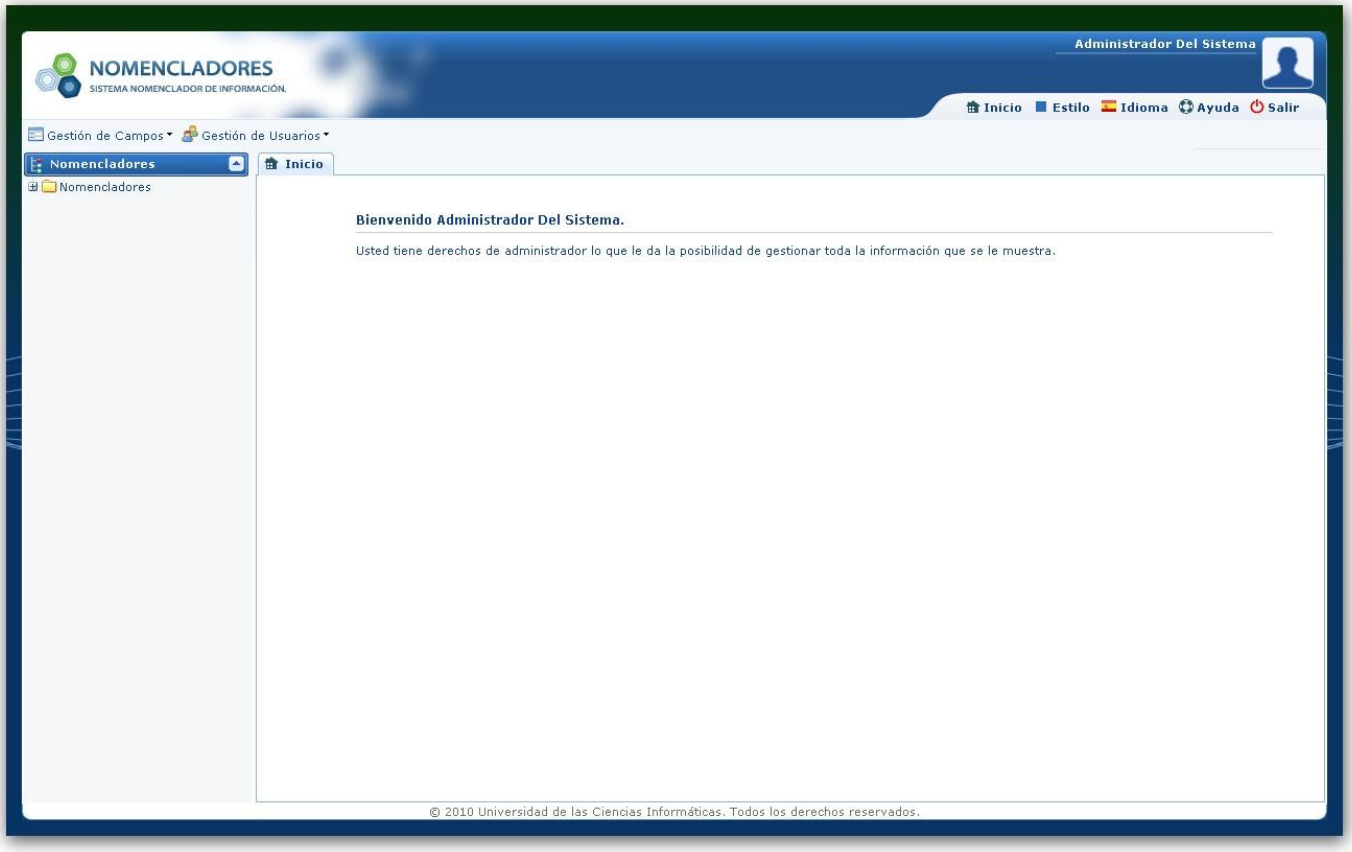
Anexo 1: Diagrama de representación del patrón MVC.



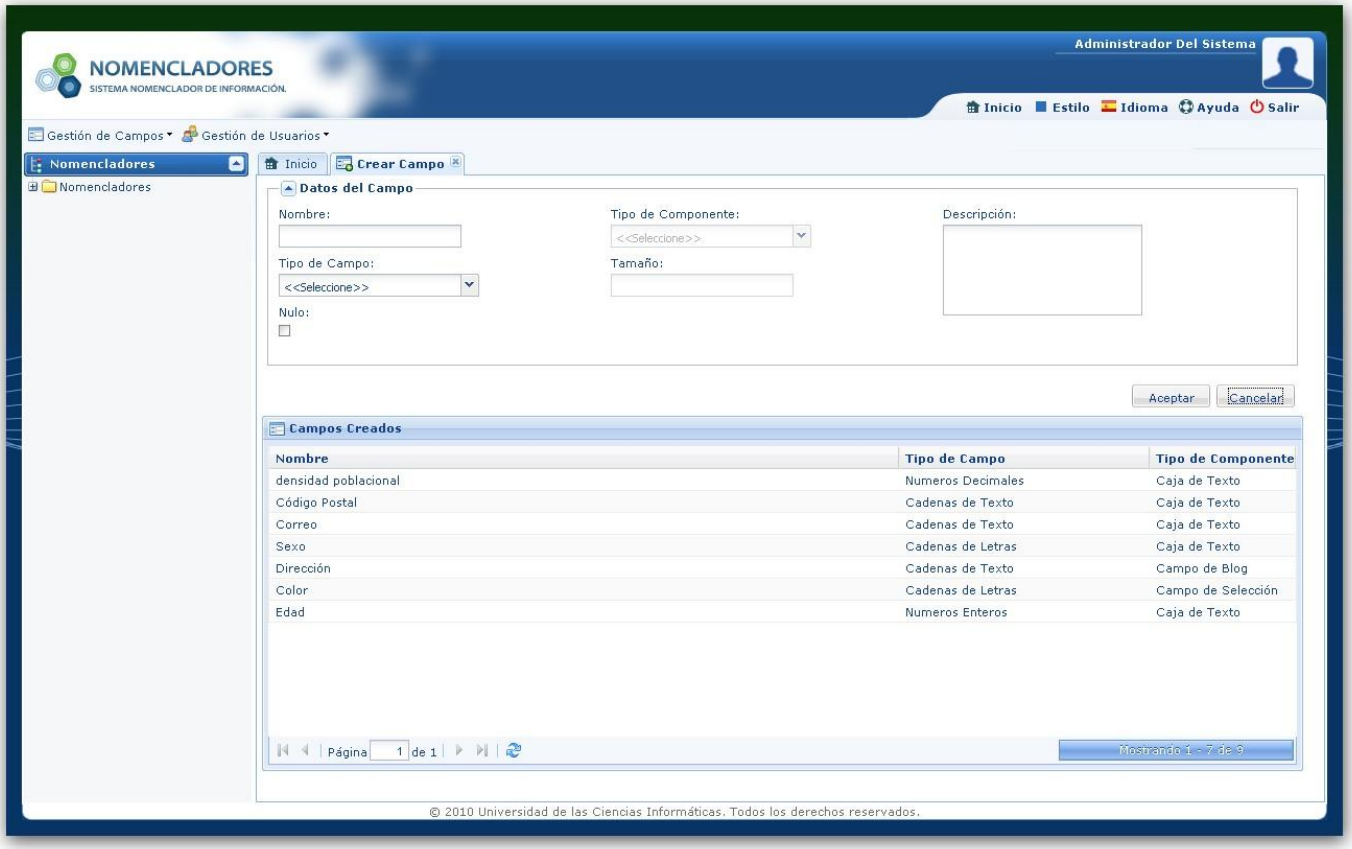
Anexo 2: El flujo de trabajo de Symfony.



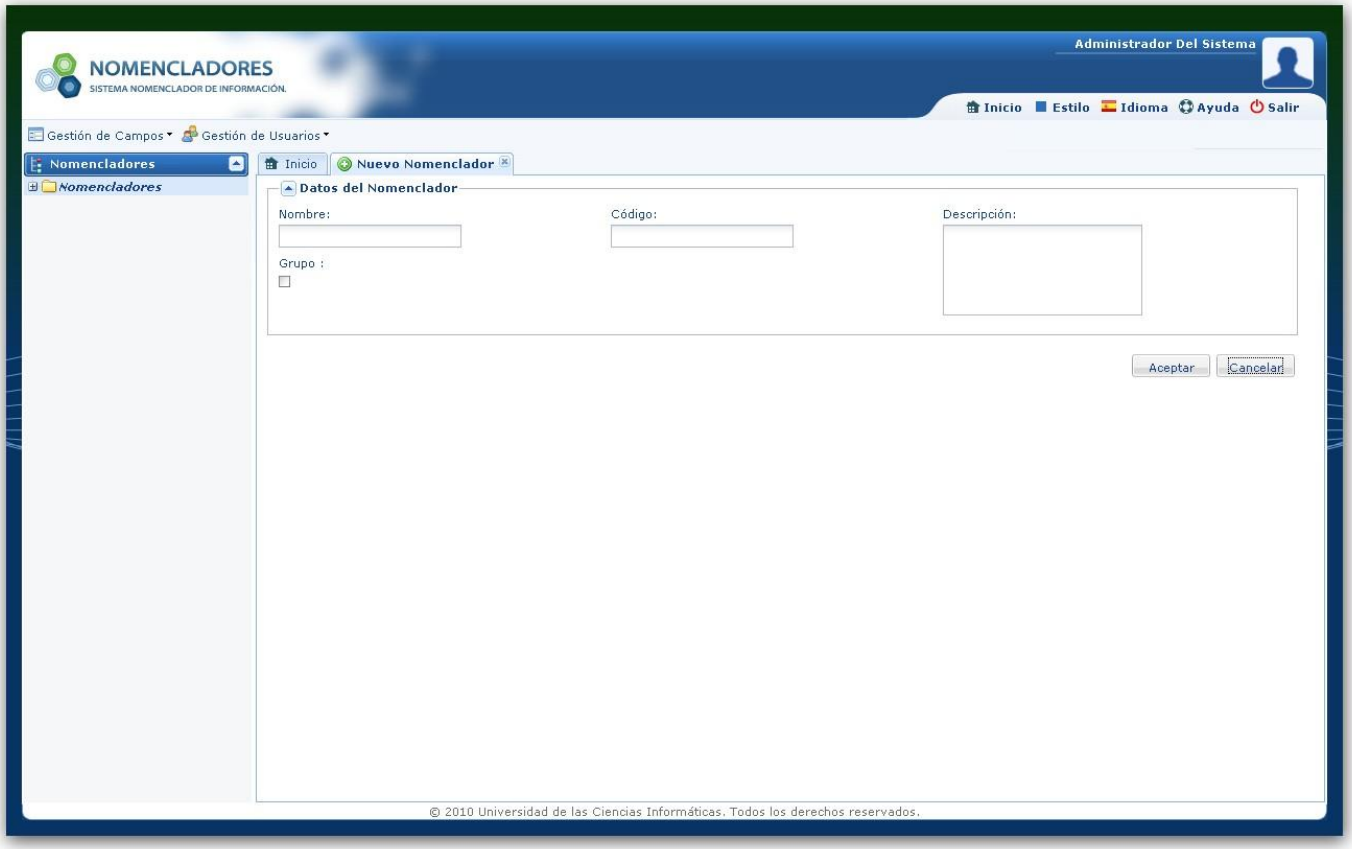
Anexo 2: Interfaz principal del sistema.



Anexo 3: Interfaz del sistema para crear un campo.



Anexo 4: Interfaz del sistema para crear un grupo.



Anexo 5: Interfaz del sistema para crear un usuario.