Universidad de las Ciencias Informáticas

Facultad 7

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

Título: Diseño dinámico de la Ficha de Investigación Epidemiológica del Sistema de Información Hospitalaria alas HIS

Autor:

Jorge Fidel Tillán Gómez

Tutores:

Ing. Yoandy González Martínez

Ing. Omar García Bonelly

La Habana, junio de 2011

"Año 53 de la Revolución"

DATOS DE CONTACTO

Ing. Yoandy González Martínez.

Profesor asistente graduado en el año 2007 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Profesor de la Facultad # 7. Ha impartido las asignaturas de Gráficos por Computadoras, Preparación para la Prueba de Nivel de Programación, y Programación Web. Se ha desempeñado como jefe del módulo Bloque Quirúrgico en el Departamento de Gestión Hospitalaria del CESIM. Culminó su período de adiestramiento en el curso 2008-2009.

Correo electrónico: ygonzalezm@uci.cu

Ing. Omar García Bonelly

Instructor recién graduado en el año 2009 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Profesor vinculado a la Facultad 7 y miembro del Departamento de Sistema de Gestión Hospitalaria.

Correo electrónico: obonelly@uci.cu

AGRADECIMIENTOS

A la Universidad de las Ciencias Informáticas, recinto de magnificas condiciones para la formación académica.

A mi carrera por ser un hito fundamental en mi superación profesional.

A mis profesores y tutores, fuentes de conocimientos invaluables y loables que supieron compartirlos conmigo.

A mis padres, hermana y abuelas por su total e incondicional apoyo.

Un especial agradecimiento a mi tío y cuñado, el primero por tener total y segura confianza en mí, el segundo por su inagotable y desbordante inteligencia natural, ambos una guía oportuna y aceptada para la realización de esta tesis.

Y a todas aquellas personas que colaboraron de manera intelectual, moral y espiritual para el desarrollo de esta tesis.

Índice

Resumen

En la actualidad, la Generación Dinámica se ha convertido en una importante herramienta para el desarrollo de aplicaciones. Esta es muy usada en diversos ámbitos ya que permite construir interfaces, imágenes, tablas, reportes, plantillas; todo esto en tiempo de ejecución, lo que beneficia en gran medida a los desarrolladores de sistemas informáticos. Debido a la escasa existencia en el mundo de aplicaciones asociadas a instituciones hospitalarias que construyan dinámicamente "Fichas de Investigación Epidemiológicas", se hace necesario el desarrollo de los procesos para la construcción de las mismas.

En el presente trabajo se continúa el desarrollo del módulo de Epidemiología a través de la implementación dinámica de Fichas de Investigación Epidemiológica. Se utiliza como metodología de desarrollo de software el Proceso Unificado Racional (por sus siglas en inglés, RUP). El patrón arquitectónico que se aplica es el Modelo-Vista-Controlador, que proporciona una arquitectura bien definida aplicable al diseño e implementación de los procesos de creación de Fichas de Investigación Epidemiológicas.

La utilización del sistema en centros hospitalarios, brindará mayor rapidez y eficiencia en cuanto a la creación y a la configuración de las Fichas de Investigación Epidemiológicas. Al mismo tiempo, permitirá agregarle secciones a las mismas, además de construir nuevas fichas a las enfermedades no registradas en el sistema. Con la implementación de esta nueva funcionalidad en el módulo de Epidemiología se logra una mayor eficiencia en los centros que utilicen esta herramienta, pues facilita la manipulación, configuración y gestión de la Ficha Epidemiológica.

Palabras claves: Ficha de Investigación Epidemiológica, desarrolladores, generación dinámica, módulo, secciones.

Introducción

El manejo de la información se ha convertido en uno de los principales problemas cuando se quiere controlar los servicios brindados por una organización. Debido al gran crecimiento de la información al pasar el tiempo y al ser mucho más amplia y variada, se hace necesario encontrar mejores métodos para su almacenamiento y manipulación. Actualmente en las instituciones hospitalarias, la utilización de equipos médicos de nueva generación, conjuntamente con los avances tecnológicos para consultar pacientes han contribuido a una generación masiva de información médica. Cada día los procesos de registro, seguimiento y tratamiento del paciente deben mejorarse, modificarse y sustentarse en tecnologías para hacer más eficiente y eficaz los movimientos y labores rutinarias de un hospital, centro de salud o clínica.

Las nuevas tecnologías son cada vez más aceptadas en la sociedad debido a la gran comodidad que estas proporcionan en su uso diario. Son diversas las técnicas que se aplican en el campo de la informática con el objetivo de automatizar procesos que se realizan de forma manual. Una de estas técnicas es la generación dinámica, la misma se ha convertido en una opción muy popular y usada en el mundo. Su gran auge y uso en diversos ámbitos se debe a que el resultado obtenido no es necesariamente estático. Permite generar interfaces, imágenes, tablas, reportes, plantillas entre otros en tiempo de ejecución, además de crear de manera más rápida las respuestas a las demandas para generar flujos de información que respondan las necesidades tanto de usuarios como de empresas.

La generación dinámica se han convertido en uno de los estandartes para el desarrollo de aplicaciones y al mismo tiempo beneficia a los implementadores de sistemas informáticos. Permite obtener herramientas en tiempo de ejecución sin necesidad de interrumpir el funcionamiento del sistema, elemento muy necesario para el uso de los mismos en diversas instituciones.

Numerosas son las aplicaciones en el mundo que emplean la generación dinámica, debido a que de esta forma es posible construir estructuras en tiempo de ejecución en función del criterio del usuario. En Cuba, existen instituciones que han utilizado esta técnica de desarrollo como es el caso de la Universidad de las Ciencias Informáticas (UCI). En este centro se desarrolló un sistema que permite crear reportes dinámicamente que facilita la manipulación y gestión de los mismos. Además, se implementó un sistema que genera a partir de un diseño de clases, la capa de acceso a datos y la base de datos de una aplicación específica. Por otra parte, el equipo técnico de la

Biblioteca Virtual de Salud desarrolló una herramienta que permite la creación dinámica de interfaces y la incorporación de contenidos publicados en Infomed.

Debido a todas las facilidades que provee la generación dinámica surge la necesidad de seguir el desarrollo del campo de la medicina, utilizando la informática como herramienta en pleno auge y aprovechando las comodidades que aporta este tipo de diseño. Por las características que posee el gran universo de las ciencias médicas, se va haciendo cada vez más imprescindible el uso de las ventajas de la informática en un entorno caracterizado por: [3]

- Incremento del número y complejidad de las especialidades clínicas.
- Aumento de la disponibilidad y prestaciones de los ordenadores.
- Mayor familiaridad médico – máquina.
- Creciente necesidad de guardar y transmitir gran cantidad de información.

La informática médica crea una nueva disciplina que pretende, con el empleo de sistemas de información, establecer una relación entre el contenido de la medicina y el de la tecnología informática. Para lograr un buen desarrollo en este sentido, se requiere un conocimiento básico de las ciencias médicas, estadística, epidemiología, ciencias de la decisión, economía de la salud, ética médica y conocimientos de informática. [4]

Estos sistemas de información, conocidos comúnmente como Sistemas de Información Hospitalaria (HIS, por sus siglas en inglés), han ido evolucionando, atendiendo a mejoras en el funcionamiento de procesos médicos y también a la estructura presentada por las instituciones hospitalarias. Están orientados a satisfacer las carencias existentes de generación de información, para almacenar, procesar y reinterpretar datos médico-administrativos de cualquier institución sanitaria, permitiendo: [5]

- Hacer más eficiente la administración de los recursos humanos y materiales.
- Minimizar los inconvenientes y retrasos que enfrentan los pacientes a fin de lograr que estos sean los más beneficiados.
- Los profesionales de la salud encuentren en estos sistemas un recurso idóneo, favorable y flexible que responda a las necesidades de información de la institución hospitalaria o de salud.

Los HIS cuentan además con varios módulos o subsistemas que responden a las necesidades de cada departamento o área especializada con funcionamiento y características particulares. Entre estos módulos se encuentra el de Epidemiología, que es definido según el Dr. Kenneth Maxcy, profesor de epidemiología en la Universidad Johns Hopkins de Higiene y Salud Pública como "el

campo de las ciencias médicas que se interesa por las relaciones de los diferentes factores y condiciones que determinan la frecuencia y distribución de un proceso infeccioso, una enfermedad o un estado fisiológico, en una comunidad humana". [6]

Internamente en este módulo se encuentra la gestión de Fichas de Investigación Epidemiológica, elaboradas principalmente con el propósito de registrar los datos de las enfermedades de notificación obligatoria (ENO) para confirmarlas o descartarlas y de esta manera conocer si el paciente posee o no una enfermedad contagiosa. Su elaboración es muy necesaria debido a que en ellas se ordenan de forma sistemática los datos que se desean recoger, consiguiendo de esta manera facilitar la investigación. Las Fichas de Investigación Epidemiológica son requeridas por los sistemas de vigilancia epidemiológica como instrumentos para la recolección de los datos.

El proceso de creación de formatos para cada Ficha de Investigación Epidemiológica, vincula diferentes secciones teniendo en cuenta cada enfermedad, este paso es complejo para los epidemiólogos que definen estos formularios para el trabajo posterior con las fichas. La especificación de los componentes necesarios para un nuevo formulario es una de las partes de mayor relevancia para la creación de los mismos con verdadera calidad.

Una herramienta de apoyo a la gestión Epidemiológica que facilite la creación de Fichas de Investigación Epidemiológica que ofrezca un conjunto estático de componentes y una colección de interfaces gráficas invariables, produciría un inconveniente para los desarrolladores. Esto se debe a que cada vez que se necesite investigar una enfermedad nueva que requiere por consiguiente un nuevo formato de ficha, estos, tendrían que implementarla trayendo como consecuencia tiempo de desarrollo y la interrupción del servicio de la aplicación en el hospital.

En estos momentos, el Sistema de Información Hospitalaria alas HIS cuenta con una ficha de investigación de caso que es aplicable a cualquier enfermedad sin un formato definido. La misma solo permite registrar texto lo que limita la extracción de datos estadísticos y no orienta sobre aspectos específicos que se deben crear para cada ENO. Por lo tanto el sistema presenta un limitado número de Fichas de Investigación Epidemiológicas para la investigación de las ENO. De esta forma, si en alguna de las entidades que utilice el sistema se investiga una de las enfermedades para la cual no se tiene ficha de investigación, será necesario implementar una nueva en tiempo de ejecución, acción que hasta el momento no es posible realizar.

Considerando lo analizado anteriormente, sobre la situación actual de las Fichas de Investigación Epidemiológica en los hospitales, se plantea como **problema a resolver**: ¿Cómo diseñar

dinámicamente la Ficha de Investigación Epidemiológica en el Módulo de Epidemiología del sistema alas HIS?

En correspondencia con el problema, el **objeto de estudio** está enfocado en el proceso de diseño dinámico de interfaces para el desarrollo de aplicaciones informáticas en el campo de la medicina.

El **campo de acción** se centra en el proceso de diseño dinámico de Fichas de Investigación Epidemiológicas en el Módulo de Epidemiología del sistema alas HIS.

Para solucionar el problema a resolver, se define como **objetivo general**: Desarrollar un componente que permita el diseño dinámico de las Fichas de Investigación Epidemiológicas que faciliten la investigación de las enfermedades de notificación obligatoria en el Módulo de Epidemiología del sistema alas HIS.

Para dar cumplimiento al objetivo anteriormente planteado se definen las siguientes **tareas** a desarrollar:

1. Realizar un análisis crítico y valorativo de las tendencias, técnicas y tecnologías que se utilizan en el mundo para la generación dinámica de interfaces.
2. Aplicar la arquitectura definida por el Departamento de Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones.
3. Generar los artefactos correspondientes a los Flujos de Trabajo: “Implementación” y “Pruebas”.
4. Aplicar las pautas de desarrollo de los entregables y diseño definidas en el Departamento de Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones.
5. Analizar las diferentes secciones que componen las Fichas de Investigación Epidemiológicas del módulo de Epidemiología del Sistema de Información Hospitalaria alas HIS.
6. Determinar las herramientas compatibles con las tecnologías utilizadas en el desarrollo del sistema alas HIS, que se utilizan para generar componentes dinámicamente.
7. Implementar las funcionalidades asociadas al diseño y configuración de las Fichas de Investigación Epidemiológica del módulo de Epidemiología del Sistema de Información Hospitalaria alas HIS.
8. Obtener el acta de liberación otorgada por Calidad.

De esta forma se puede destacar que el desarrollo del diseño dinámico de la Ficha de Investigación Epidemiológica del Sistema de Información Hospitalaria, brindará los siguientes beneficios a partir de su puesta en práctica:

1. Flexibilidad para la configuración de Fichas de Investigación Epidemiológica.
2. Mejor gestión y manejo de las Fichas de Investigación Epidemiológicas.
3. La aplicación no dejará de funcionar en la entidad que este brindando sus servicios.

El presente trabajo, se ha organizado en cuatro capítulos a fin de lograr una adecuada organización de su información. En el primero: "**FUNDAMENTACIÓN TEÓRICA"**, se realiza un estudio concerniente al estado del arte en la que se realiza un análisis enfocado en los sistemas que utilicen la generación dinámica conjuntamente con los sistemas para la salud que contienen un área de Epidemiología y lleven el control de la vigilancia epidemiológica, además de enunciar las tendencias, tecnologías, metodologías y herramientas que fueron utilizadas para el desarrollo de este trabajo.

Posteriormente en el segundo capítulo: **"DESCRIPCIÓN DE LA ARQUITECTURA",** se define un modelo de dominio que describe las distintas entidades implicadas en el sistema, contiene la fundamentación de la arquitectura, la estrategia de integración y los elementos que garantizan la seguridad en el sistema, conjuntamente con la especificación de los estándares y estilos de codificación a utilizar.

En el tercer capítulo "**DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA**" se realiza una descripción de las clases a implementar y operaciones necesarias en el proceso de desarrollo así como una valoración sobre del diseño propuesto asociado al trabajo.

El cuarto y último capítulo "**MODELO DE PRUEBA**" se especifica el método de prueba a utilizar y se describen los casos de prueba.

Capítulo 1: Fundamentación Teórica

En el presente capítulo se realiza una descripción con relación a conceptos básicos relacionados con el dominio del problema a resolver. Se aborda detalladamente el estudio de los sistemas que generan interfaces dinámicamente y el estudio de sistemas enfocados a la vigilancia epidemiológica. Posteriormente se describen las técnicas, tecnologías, metodologías y herramientas de software definidas por la dirección del Departamento de Gestión Hospitalaria del Centro Especializado en Soluciones de Informática Médica (CESIM).

1.1 Conceptos básicos relacionados con el problema planteado

A modo de facilitar la comprensión de dicha investigación se especifican algunos conceptos relacionados con el problema planteado.

Un Sistema de Información Hospitalario (HIS), es un sistema de información integrado, diseñado para manejar los aspectos administrativos, financieros y clínicos de un hospital. Esto abarca el proceso de información basado en papel así como también los datos procesados en máquinas. [7]

Estos sistemas están compuestos por varios módulos, dentro de los cuales se encuentra el de epidemiología. Uno de los principales aspectos de este es la gestión de Fichas de Investigación Epidemiológica (FIE), las cuales son documentos donde se registran todos los datos que tienen importancia epidemiológica y administrativa respecto de un caso.

Algunas de estas son únicas, pero existen otras que en dependencia de la patología permiten ser elaboradas en más de una ocasión. Una Ficha de Investigación Epidemiológica se crea a los pacientes que se les detecta una Enfermedad de Notificación Obligatoria (ENO). No hay modelos de fichas definidos para todas las ENO, debido a que estos son creados en dependencia de la necesidad epidemiológica en un área determinada.

Las Enfermedades de Notificación Obligatorias (ENO) son aquellas que se consideran de gran importancia para la salud pública. Las agencias locales, estatales y nacionales exigen que dichas enfermedades se notifiquen cuando sean diagnosticadas por parte de los médicos o laboratorios. La notificación permite recoger datos estadísticos que muestren la frecuencia con que ocurre la enfermedad, lo cual, a su vez, ayuda a los investigadores a identificar las tendencias de la enfermedad y a rastrear los brotes de la misma. Esto puede ayudar a controlar brotes futuros. [8]

Los eventos sujetos a notificación obligatoria son en general enfermedades de tipo transmisibles o algún otro evento que requiere investigación inmediata para evitar la aparición de nuevos casos (por ejemplo, algunas intoxicaciones). Su notificación permitiría generar el estado de alerta a las autoridades sanitarias y promover acciones de control y prevención. [9]

La vigilancia epidemiológica por su parte, es el conjunto de actividades que permite reunir la información indispensable para conocer en todo momento la conducta o historia natural de la enfermedad, detectar o prever cualquier cambio que pueda ocurrir por alteraciones en los factores condicionantes con el fin de recomendar oportunamente, sobre las bases firmes, las medidas indicadas, eficientes, que lleven a la prevención y control de la enfermedad. [10]

La función de los sistemas informáticos para la vigilancia epidemiológica es facilitar el seguimiento de casos y el análisis sistemático de la información existente en cuanto al comportamiento de una enfermedad en el tiempo, en las diferentes áreas geográficas y en diferentes grupos de la población, por ejemplo en los diferentes grupos de edad. [11]

1.2 Sistemas automatizados existentes

Los sistemas que emplean la generación dinámica de interfaces además de proporcionar información, se adaptan a las características de diseño del usuario de forma que el trabajo a realizar por el computador sea menos pesado y más rápido. Con el empleo de esta técnica en el campo de la medicina, específicamente en la rama de epidemiología, es posible construir estructuras dinámicas que tomen su forma definitiva en tiempo de ejecución en función del modelo del usuario.

A nivel mundial existen aplicaciones que emplean la generación dinámica de interfaces, entre las cuales se encuentra Pegasus, un sistema genérico creado en la Universidad Autónoma de Madrid, para sistemas hipermedia de enseñanza adaptativa altamente independiente de la representación del conocimiento del dominio y del mantenimiento del estado de la aplicación. Su principal tarea es realizar mínimas suposiciones sobre cómo se representa el conocimiento educativo, con el objetivo de poder ser utilizada con diferentes formas de representación, y por lo tanto con diferentes sistemas de construcción y gestión de cursos.

En tiempo de ejecución, el usuario interactúa con la aplicación desde un navegador web. Cada vez que el usuario se desplaza a un objeto, PEGASUS responde generando una página HTML. Para ello, el sistema:

1. Resuelve la petición del usuario para determinar el objeto al que se debe ir.
2. Localiza la instancia en el modelo del dominio.
3. Actualiza el modelo del dominio y del usuario.
4. Genera la presentación HTML, aplica las reglas pertinentes y la plantilla correspondiente a su clase.

La unidad de interacción con el usuario es pues la petición HTTP. La actualización del modelo del usuario se lleva a cabo teniendo en cuenta únicamente la información transmitida en estas peticiones. Esta implementado bajo código Java. Las características de la plataforma e interfaz de usuario se capturan en el cliente mediante código JavaScript que PEGASUS inserta en las páginas HTML generadas. Dicha información es enviada al servidor como parte de la petición HTTP cuando el usuario pulsa sobre enlaces y botones de las páginas. Esta suposición simplifica sensiblemente la arquitectura del sistema y la integración con módulos y herramientas externos. [12]

El Grupo de Ingeniería Web de la Universidad de Litoral en Argentina, desarrolló un Repositorio Institucional de Objetos de Aprendizaje. El mismo cuenta con una perspectiva dinámica para la generación de sus interfaces a partir de modelos basados en esquemas XML. Este repositorio utiliza un contexto educativo universitario, en reemplazo de los procedimientos desarticulados que se encuentran en uso en la actualidad.

La generación de las interfaces se realiza a partir de los datos aportados por un modelo de entrada. Este modelo debe contener la información necesaria sobre los campos, tipos de datos y multiplicidades, entre otros, a ser generados e incluidos en las interfaces resultantes. Para la generación de estas interfaces se propuso una aplicación Web basada en tecnología Java del lado del servidor. Esta aplicación, toma como datos de entrada la representación del modelo descrita en el punto anterior.

La arquitectura de la aplicación propuesta está compuesta por dos bloques principales: Modelo y Vista. El bloque Modelo utiliza Java Architecture for XML Binding (JAXB) para transformar y/o adaptar a un modelo de objetos el esquema utilizado como entrada. Una vez finalizada esta tarea, se expone el modelo resultante al proceso de análisis que es llevado a cabo por Servlets Java. El mismo proceso invoca los métodos de validación adecuados según los tipos de datos de los elementos que constituyen el modelo. Finalmente, los datos resultantes son enviados al bloque Vista.

El bloque Vista está compuesto por JavaServer Pages (JSP) y clases de utilidades Web escritas en JavaScript. Estas clases se asocian a los elementos del modelo de objetos; sus métodos aprovechan las ventajas de los comportamientos adaptativos del W3C Document Object Model (DOM) para realizar la verificación de la sintaxis y la validación de los datos.

La aplicación agrega dos características muy importantes a las JSP que las distinguen de las JSP estándar. La primera es su comportamiento adaptativo, basado en el W3C DOM, donde los elementos opcionales y/o compuestos del modelo (que se corresponden con los tipos de datos no requeridos y complejos, respectivamente) se visualizan en base a pedidos en lugar de por defecto. La segunda y quizás la más importante característica es la utilización de librerías de etiquetas (tag libraries) definidas especialmente para mejorar los tipos de datos complejos presentes en la definición de esquemas XML. [13]

En la Universidad Carlos III de Madrid, en el año 2009, desarrolló una herramienta para la gestión de los requisitos de un proyecto de software con generación dinámica de interfaces gráficas. Puesto que dentro de la Ingeniería de Requisitos, la información sobre la que trabaja el equipo de analistas se convierte en un flujo variable de información de difícil tratamiento. La especificación de requisitos en un proyecto de este tipo, es una de las partes de mayor relevancia para la obtención de software de calidad. Una deficiente especificación de requisitos conlleva, en la gran mayoría de los casos, a un fracaso del sistema.

El sistema, mediante una serie de pasos, permite adaptar las herramientas a cada proyecto o incluso adecuarla a las características específicas de un equipo de trabajo además de realizar una gestión de los requisitos con un conjunto de interfaces adaptadas. Estos pasos son:

- Permite seleccionar los tipos de requisitos específicos para cada proyecto. La interfaz generada sólo ofrece la posibilidad de insertar requisitos de los tipos seleccionados.
- Define los campos necesarios para cada tipo de requisitos. El usuario sólo tendrá que completar los campos especificados. Se consigue: mayor facilidad para identificar campos esenciales de cada requisito, mayor velocidad en la inserción de información y menor posibilidad de incluir información superflua o innecesaria.
- Una misma estructura puede utilizarse como base para la gestión de requisitos de diferentes proyectos. Incluso podrá convertirse en el patrón específico de un equipo de trabajo.

Para la elaboración de este software, se aplicó el patrón Modelo-Vista-Controlador. Se utilizó UModel para la creación e interpretación del diseño del software. El proyecto utilizará XML como

herramienta para el almacenamiento de la información, por lo que todos los archivos de la aplicación serán documentos XML con estructuras específicas. Está implementado en el lenguaje de programación orientado a objetos Visual Studio .NET 2005. Funciona bajo el sistema operativo Microsoft Windows XP Professional. [14]

Entre las principales limitantes del sistema anteriormente citado se encuentra, en primer lugar que no está enfocado a la generación dinámica de componentes necesarios en el campo de la salud. Conjuntamente a esto, el modelo de almacenamiento de datos que utiliza es XML, siendo más lento a la hora de realizar consultas y obtener datos. Además de que es un método obsoleto que se utiliza para gestionar y almacenar pequeñas cantidades de información. Al no tener un gestor de base de datos, no consta de un mecanismo de gestión de usuarios, esto impide que exista un control de permisos y restricciones a estos que garanticen la seguridad de la información. Por último, ofrece una lista fija de tipos de requisitos sobre los que se seleccionan los específicos del proyecto, sin posibilidad de añadir en tiempo de ejecución nuevos tipos de requisitos no definidos.

Además del estudio de sistemas que utilizan la generación dinámica, se trataron algunos enfocados a la medicina especialmente en el campo de la epidemiología.

En agosto de 1999 se inicia la obtención de datos ampliados a través de la aplicación y envío de la Ficha de Investigación Epidemiológica de Muerte Materna (FIEMM) en Perú. Esta información amplía la posibilidad del análisis que se realiza en la Oficina General de Epidemiología así como en las oficinas de epidemiología de las Direcciones de Salud de ese país.

Durante la implementación, desarrollo y consolidación de la vigilancia epidemiológica de mortalidad materna, se han incrementado y ampliado las actividades inherentes a solucionar el problema de información en torno a las muertes maternas ocurridas. Refiriendo la necesidad de contar con bases de información cada vez más precisas y oportunas para la toma de decisiones dirigidas a la disminución de este daño, se llega a la siguiente determinación:

- La magnitud, distribución y tendencias de la mortalidad materna.
- Identificación de factores de riesgo.
- De la Vigilancia del daño a la vigilancia de riesgos (Vigilancia Integral).

Sin embargo la información contenida en la FIEMM, es procesada parcialmente en todos los niveles (local, regional y central), debido a que no se cuenta con un sistema de base de datos desarrollada en referencia a la FIEMM. Esto trae como consecuencia que no se cuente con información de

calidad y produce un atraso en la oportunidad de la información, además que también la información es discordante tanto al nivel regional como central.

Es por ello que en el año 2002 el grupo temático de muerte materna desarrolló una base de datos, donde se ingresan las FIEMM que son remitidas a la Oficina General de Epidemiología. Una de las características de esta base de datos es que recoge información de la FIEMM en las que se depositan alrededor de 150 variables donde se explora diferentes aspectos inherentes a la ocurrencia de la muerte materna. Es necesario mencionar que la información correspondiente a la mortalidad materna se consigna mediante dos bases de datos: [15]

- Base de Datos de la Vigilancia Epidemiológica Semanal Noti 99.
- Base de Datos de la Ficha de Investigación Epidemiológica de Muerte Materna.

La restricción más significativa de este sistema en relación con los objetivos que se han propuesto estriba en su limitada capacidad en cuanto al ingreso de las Fichas de Investigación Epidemiológicas, las cuales se realizan de forma manual a causa de no poseer una herramienta que las genere dinámicamente.

Epi Info, es un software diseñado por el Centro para el Control de Enfermedades de Atlanta, se caracteriza por ser un programa ligero, de buen funcionamiento en lugares donde la conectividad de red es limitada. Es utilizado en todo el mundo para la evaluación rápida de los brotes de enfermedades, para el desarrollo de sistemas de vigilancia de pequeñas y medianas empresas.

Posee herramientas sencillas que permiten la rápida creación de instrumentos de recolección y visualización de datos, análisis y presentación de informes. Utiliza métodos epidemiológicos, de esta manera los epidemiólogos y otros profesionales de la salud y la medicina consiguen; recopilar información, personalizar el proceso de entrada de datos, desarrollar cuestionarios de manera rápida y realizar análisis estadísticos avanzados.

Epi Info opera en sistemas operativos Windows 98, NT 4.0, 2000, XP o Vista. La memoria RAM necesaria varia en relación al sistema operativo donde trabaje, requiere una memoria de 32 MB para Windows 98 y 64 MB en caso de Windows NT o 2000. Para Windows XP o Vista necesita una capacidad de 128 MB de memoria. Necesita un procesador de 200 megahercios, aunque se recomienda que para el caso de Windows XP o vista sea de 300 megahercios.

Para su instalación debe existir en el disco duro una capacidad libre como mínimo de 260 megabytes. Un requisito importante para el trabajo en Windows es que tenga instalado el Service

Pack 3 de Windows XP; y para Vista debe estar instalado el control de edición para aplicaciones DHLM. [16]

Epi Info, a pesar de que evalúa de manera rápida brotes de enfermedades, para el desarrollo de sistemas de vigilancia no cuenta con una funcionalidad para crear fichas de investigación epidemiológicas, siendo esta limitante un factor imprescindible para el objetivo de esta investigación.

En el año 2001, con el objetivo de fortalecer el sistema de vigilancia epidemiológica en Argentina, se comienza el desarrollo de un sistema informático llamado Sistema Nacional de Vigilancia en Salud (SNVS). Durante el segundo semestre del año 2002, este sistema fue probado en algunas provincias del país. A partir de las dificultades evidenciadas en su uso, se introdujeron cambios que permitieron optimizar su diseño y funcionamiento.

Este sistema, de carácter modular, realiza el ingreso de la información vía online complementario a la notificación que diariamente se envía por e-mail al Ministerio de Salud del Gobierno de la Ciudad de Buenos Aires. Para su implementación se realizó sobre la base del conocimiento de la situación en la región con relación al equipamiento informático y cantidad y calidad del recurso humano disponible.

Antes del desarrollo del SNVS, la devolución de información era muy dificultosa, tanto por cuestiones operativas, como por limitaciones en la disponibilidad de recursos. Desde el punto de vista operativo, el no contar con un sistema informatizado para la recolección y análisis de los datos, imponía una carga muy intensa para los diferentes epidemiólogos de cada región, quienes debían realizar todos estos pasos en forma manual. A partir de la puesta en marcha del SNVS en el país, y más concretamente en la región, todos estos procesos han mejorado en gran medida. [17]

Entre las principales características con que cuenta este software se encuentra: [18]

- Se ha concebido en diferentes módulos que permiten el desarrollo independiente y compatible entre sí de los subsistemas de vigilancia.
- Se utiliza Internet como soporte comunicacional, con datos que se cargan desde los distintos departamentos directamente a servidores centrales.
- Las herramientas que permiten el análisis básico de la información están disponibles on-line, además se ha desarrollado un paquete de herramientas complementarias que funciona off-line.

- La plataforma de desarrollo es ASP y la base de datos MS SQL 2000 Advanced Server.

En el año 2008, la Universidad de las Ciencias Informáticas desarrolló el Sistema de Control Sanitario Internacional (CSI) por el grupo de Sistemas Especializados en salud. Este sistema cuenta con diversos módulos especializados, entre ellos se encuentra el de Higiene y Epidemiología, este módulo desarrolla fundamentalmente dos tópicos, el control de enfermedades tropicales donde en su primera versión solo se trabajó en el control de Dengue mediante la vigilancia a pacientes con síntomas febriles inespecíficos y el control y chequeo de los viajeros con el fin de evitar enfermedades provenientes de otras regiones.

Es un sistema totalmente centralizado, impide la duplicidad de la información y la pérdida parcial o total de esta. Su estructura en materia de información permite a los especialistas, la toma de decisiones precisas y la asignación de recursos necesarios para dar respuesta a una situación determinada, así como para prevenirla. El Módulo de Higiene y Epidemiología permite la gestión de pruebas de laboratorio y el seguimiento a pacientes con dengue o con sospechas de portar la enfermedad. [19]

Las herramientas, técnicas y tecnologías que se pusieron en práctica para llevar a cabo el software fueron seleccionadas teniendo en cuenta su condición de software libre para garantizar económicamente ahorro en cuanto a licencias, soporte y mantenimiento. Para ello se escogió la plataforma LAMP formada por la agrupación de Linux, Apache, MySQL y PHP, además de JavaScript que brinda el soporte tecnológico del software, el cual estuvo desarrollado a su vez con el uso de la herramienta Zend Studio y modelado con la herramienta CASE Visual Paradigm; siguiendo en todo momento la metodología RUP. [20]

Aunque este sistema fue realizado en Cuba y desarrollado con tecnología no propietaria; no se utiliza ya que no implementa una solución enfocada a los procesos de vigilancia epidemiológica de un hospital, principalmente a la gestión y creación de las fichas de investigación epidemiológicas. Esto se debe a que principalmente está enfocado a llevar el control de la gestión de la información relacionada con la vigilancia epidemiológica solamente de los viajeros.

Nombre	País	Empresa	Tipo licencia		Tipo Aplicación		Código	Enfoque	
			Libre	Privado	Web	Desktop		Generador Dinámico	Salud
Pegasus	España	Univ. Madrid	X		X		Java	X	
Repositorio	Argentina	Litoral	X		X		Java	X	
Herramienta CASE	España	Univ. Carlos III		X		X	Visual Studio .NET 2005	X	
FIEMM	Perú	RENACE		X		X			X
Epi Info	Estados Unidos	CDC Atlanta		X		X	Visual Basic Versión 6		X
SNVS	Argentina	Ministerio de Salud		X	X		ASP.NET		X
CSI	Cuba	UCI	X		X		PHP		X

Tabla 1.2.1. Análisis de los sistemas investigados

Luego de haber realizado un estudio acerca de los sistemas tratados en la investigación, se puede observar que estos sistemas no se ajustan totalmente con las características necesarias para resolver el problema de la generación dinámica de Fichas de Investigación Epidemiológicas. En el caso específico de los mencionados que utilizan la generación dinámica de interfaces, no se aplican al campo de la salud pública. Por otra parte algunos de ellos son propietarios, lo que impide su uso en Cuba. Conjuntamente a esto, los sistemas enfocados a la epidemiología no cuentan con una funcionalidad para crear Fichas de Investigación Epidemiológicas, siendo esta limitante un factor imprescindible para el objetivo de esta investigación.

1.3. Herramientas y tecnologías de desarrollo

1.3.1. Eclipse

Eclipse es una poderosa herramienta que permite montar e integrar diferentes aplicaciones para convertirse en un Entorno Integrado de Desarrollo (IDE por sus siglas en inglés integrated development environment). La arquitectura de plugins de Eclipse permite, además de integrar

diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, etc.

La plataforma Eclipse, cuando se combina con Java Development Tools (JDT), ofrece muchas de las características que se esperaría de un software de calidad IDE. Posee un editor de sintaxis, la compilación de código incremental, un depurador de nivel fuente tipo thread-aware, un navegador de clases, y las interfaces con los sistemas estándar de control de código fuente, como CVS y ClearCase. También incluye una serie de características únicas, como la refactorización de código y actualizaciones automáticas a través del Update Manager. [21]

1.3.2. JBoss Seam

JBoss Seam es un framework que integra y unifica los distintos estándares de la plataforma Java EE 5.0, pudiendo trabajar con todos ellos siguiendo el mismo modelo de programación. Ha sido diseñado para simplificar al máximo el desarrollo de aplicaciones, basando el diseño en Plain Old Java Objects (POJOs) con anotaciones. Estos componentes se usan desde la capa de persistencia hasta la de presentación, poniendo todas las capas en comunicación directa.

Seam integra tecnologías como JavaScript asíncrono y XML (AJAX), Java Persistence API, Enterprise Java Beans (EJB 3.0) y Business Process Management (BPM) en una unificada y completa solución de pila, con sofisticadas herramientas. Ha sido diseñado desde cero para eliminar la complejidad en la arquitectura y los niveles de la API. Permite a los desarrolladores ensamblar aplicaciones web complejas con simples clases Java anotadas y un amplio conjunto de componentes de interfaz de usuario. [22]

Integra además el concepto de workspaces (entornos de trabajo) permitiendo que el usuario tenga en varios tabs o ventanas del navegador actividades del negocio con contextos completamente aislados. Seam integra transparentemente la administración de procesos del negocio vía jBoss jBPM, haciendo muy fácil implementar y optimizar complejas colaboraciones e interacciones con el usuario. [23]

1.3.3. JBoss Server

JBoss Server es un servidor de aplicaciones J2EE (Java Platform, Enterprise Edition) de código abierto implementado en Java. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible Java, además de que implementa todo el paquete de servicios de J2EE. Soporta todas las especificaciones correspondientes, incluyendo servicios

adicionales como clusterizar, carga en memoria caché y persistencia, por ser una plataforma con certificación JEE 5. Es ideal como servidor de aplicaciones Java y aplicaciones Web. También soporta Enterprise Java Beans (EJB) 3.0. Cuenta con un conjunto de componentes claves como son: JBoss AS 4.2, Hibernate 3.2.4, Seam 2.0. [24]

Las características destacadas de JBoss incluyen: [25]

- Producto de licencia de código abierto sin coste adicional.
- Soporta los estándares:
  - Portlet Specification and API 1.0 (JSR-168)
  - Content Repository for Java Technology API (JSR-170)
  - Java Server Faces 1.2 (JSR-252)
  - Java Management Extensión (JMX) 1.2
  - Compatibilidad 100% con J2EE 1.4 al utilizar JBoss AS
- Confiable a nivel de empresa.
- Incrustable, orientado a arquitectura de servicios.
- Servicios del middleware para cualquier objeto de Java.

1.3.4. JBoss Tools

Jboss Tools es un conjunto de herramientas para Eclipse. Entre estas herramientas, Jboss Tools dispone de plugins que proporcionan soporte en Eclipse para Hibernate, JBoss AS, Drools, jBPM, JSF, XHTML, Seam, Smooks, JBoss ESB o JBoss Portal, entre otros.

JBoss Tools consta de varios módulos: [26]

- Editor visual JSF y Facelets.
- Herramientas de generación, refactorización y completado de código Seam.
- Herramientas para Hibernate como ficheros de mapeo, anotaciones, ingeniería reversa con JPA.
- Herramienta para inicializar y parar el servidor de aplicaciones JBoss.
- Editor de workflows jBPM.
- Herramientas basadas en SoapUI para el desarrollo de web services.

1.3.5. RichFaces

RichFaces es una librería de componentes visuales para JSF, escrita en su origen por Exadel y adquirida por JBoss. RichFaces posee un framework avanzado para la integración de funcionalidades Ajax en dichos componentes visuales, mediante el soporte de la librería Ajax4JSF.

Son características de RichFaces las siguientes: [27]

- Se integra perfectamente en el ciclo de vida de JSF.
- Incluye funcionalidades Ajax, de modo que no se utiliza JavaScript y tiene un contenedor Ajax propio.
- Contiene un set de componentes visuales, los más comunes para el desarrollo de una aplicación web rica (Rich Internet Application), con un número bastante amplio que cubren casi todas las necesidades,
- Soporta facelets.
- Soporta css themes o skins.
- Es un proyecto open source, activo y con una comunidad también activa.

1.3.6. PosgtreSQL

PostgreSQL es un sistema de gestión de base de datos relacional, publicado bajo la licencia Berkeley Software Distribution (BSD). El desarrollo de PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

Los desarrolladores de proyectos basados en software libre tienen muy en cuenta PostgreSQL cuando los requerimientos de un proyecto exigen prestaciones de alto nivel. Esto se debe a que PostgreSQL se destaca por su amplísima lista de prestaciones que lo hacen capaz de competir con cualquier Sistema de Gestión de Base de Datos (SGBD) comercial: [28]

- Está desarrollado en C, con herramientas como Yacc y Lex.
- La API de acceso al SGBD se encuentra disponible en C, C++, Java, Perl, PHP, Python y TCL, entre otros.
- Cuenta con un rico conjunto de tipos de datos, permitiendo además su extensión mediante tipos y operadores definidos y programados por el usuario.
- Su administración se basa en usuarios y privilegios.

- Sus opciones de conectividad abarcan TCP/IP, sockets Unix y sockets NT, además de soportar completamente ODBC.
- Control de concurrencia multi-versión, lo que mejora sensiblemente las operaciones de bloqueo y transacciones en sistemas multi-usuario.
- Soporte para vistas, claves foráneas, integridad referencial, disparadores, procedimientos almacenados, subconsultas y casi todos los tipos y operadores soportados en SQL92 y SQL99.
- Implementación de algunas extensiones de orientación a objetos. En PostgreSQL es posible definir un nuevo tipo de tabla a partir de otra previamente definida.

1.3.7. SQL Manager for PostgreSQL

SQL Manager for PostgreSQL es una herramienta de alto rendimiento para la administración de bases de datos PostgreSQL. Esta aplicación de orden avanzado, ofrece la oportunidad de administrar completamente los servidores basados en lenguaje PostgreSQL. Cuenta con un funcionamiento verdaderamente sólido y comprensible si el usuario en cuestión, conoce al pie de la letra el funcionamiento y dinámica de trabajo del lenguaje nombrado anteriormente.

Algunas de sus características principales son el apoyo de datos UTF8 (8-bit Unicode Transformation Format). Presenta excelentes herramientas visuales y de texto para las consultas, gestión de datos y la navegación rápida, herramientas avanzadas para la manipulación de datos, seguridad, capacidad de importar los datos y exportaciones, el acceso al servidor PostgreSQL a través del protocolo HTTP y gestión de la seguridad. [29]

1.3.8. Framework Hibernate

Hibernate es un poderoso framework de persistencia y mapeo de objetos en bases relacionales para la plataforma Java. Permite desarrollar un modelo de datos que utiliza todo el poder de un lenguaje orientado a objetos, incluyendo asociación, herencia, polimorfismo, composición y el framework de colecciones de Java. [30]

Hibernate, para poder funcionar, además de las clases y la base de datos, necesita información que le permita mapear las clases contra las tablas. Esta información se puede ingresar mediante código o bien como archivos de configuración en XML. Además, también se requiere información adicional para que el framework se pueda conectar a la base de datos. Proporciona un lenguaje de consulta

rica para acceder a objetos, proporcionar almacenamiento en caché y el apoyo JMX. El es destinado a proporcionar la persistencia de alto rendimiento con bajos recursos. [31]

1.3.9. JPA

JPA es un framework de persistencia, que se abstrae de las bases de datos y brinda un estandar para persistir los datos en java. JPA viene a solucionar el vacio que hay entre utilizar objetos y persistirlos en una DB relacional. Este framework mapea automáticamente las clases en la base de datos de manera transparente y utiliza un estandar, que entre otras cosas permite poder migrar de motor cuando se desee, y poder compartir código o trabajar en equipo sin ningún problema.

JPA trabaja fuertemente con anotaciones. Para mapear un bean (una clase java) con una tabla de la base de datos, se tiene que escribir lo que se llama un Entity. Esto es tan sencillo como escribir un Bean, con sus atributos y métodos get y set. Y después añadirle la anotación "@Entity" a la par de seleccionar uno de sus atributos como clave primaria, por ejemplo "@Id". [32]

1.3.10. Visual Paradigm

Visual Paradigm es un Lenguaje de Modelado Unificado (UML) que permite el diseño de sistemas con todo tipo de diagrama UML, invertir el código fuente al modelo de UML, generar código fuente a partir de diagramas y desarrollar la documentación. Visual Paradigm ofrece a los analistas del sistema todas las herramientas necesarias para capturar y organizar los requisitos.

Visual Paradigm produce documentación del sistema en formato PDF, HTML y formatos de MS Word. Proporciona además abundantes tutoriales de UML, así como demostraciones interactivas de dicho lenguaje.

Entre sus características se puede mencionar: [33]

- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Modelado colaborativo con CVS y Subversion.
- Ingeniería inversa - Código a modelo, código a diagrama
- Ingeniería inversa Java, C++, Esquemas XML, XML,.NET exe/dll, CORBA IDL
- Diagramas EJB - Visualización de sistemas EJB.
- Generación de código y despliegue de EJB´s - Generación de *beans* para el desarrollo y despliegue de aplicaciones.
- Diagramas de flujo de datos

- Soporte ORM - Generación de objetos Java desde la base de datos
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos
- Importación y exportación de ficheros XMI
- Integración con Visio - Dibujo de diagramas UML con plantillas (*stencils*) de MS Visio
- Editor de figuras
- Otras herramientas y *plugins* de modelado UML.

Esta herramienta case se integra a la Plataforma Java, que funciona en sistemas operativos Windows, Linux y Mac OS X. La integración es vista con ambientes inteligentes de desarrollo (SDE) para Eclipse, NetBeans, Oracle JDeveloper, JBuilder, entre otros.

1.4. Lenguaje de programación

1.4.1. Java

Java es un lenguaje de programación orientado a objetos, independiente de la plataforma, posee un entorno de programación multiproceso, el lenguaje hereda mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel. La programación en Java, permite el desarrollo de aplicaciones bajo el esquema de Cliente Servidor, como de aplicaciones distribuidas. Esta característica lo hace capaz de conectar dos o más computadoras u ordenadores, ejecutando tareas simultáneamente, y de esta forma logra distribuir el trabajo a realizar.

Una de las características más importantes es que los programas “ejecutables”, creados por el compilador de Java, son independientes de la arquitectura. Se ejecutan indistintamente en una gran variedad de equipos con diferentes microprocesadores y sistemas operativos.

1. De momento, es público. Puede conseguirse un Java Developer's Kit (JDK) o Kit de desarrollo de aplicaciones Java gratis. No se sabe si en un futuro seguirá siéndolo.
2. Permite escribir Applets (pequeños programas que se insertan en una página HTML) y se ejecutan en el ordenador local.
3. Se pueden escribir aplicaciones para intrarredes, aplicaciones cliente/servidor, aplicaciones distribuidas en redes locales y en Internet.
4. Es fácil de aprender y está bien estructurado.

5. Las aplicaciones son fiables. Puede controlarse su seguridad frente al acceso a recursos del sistema y es capaz de gestionar permisos y criptografía. También, según Sun Microsystems, la seguridad frente a virus a través de redes locales e Internet está garantizada. Aunque al igual que ha ocurrido con otras tecnologías y aplicaciones, se han descubierto, y posteriormente subsanado, “agujeros” en la seguridad de Java.

Además de incorporar la ejecución como Applet, Java permite fácilmente el desarrollo tanto de arquitecturas cliente-servidor como de aplicaciones distribuidas, consistentes en crear aplicaciones capaces de conectarse a otros ordenadores y ejecutar tareas en varios ordenadores simultáneamente, repartiendo por lo tanto el trabajo. Aunque también otros lenguajes de programación permiten crear aplicaciones de este tipo, Java incorpora en su propio API estas funcionalidades. [34]

1.5 Metodologías de desarrollo

Se trata de una metodología objetiva a definir claramente "que" hace "qué", "cuándo y cómo", para todos aquellos involucrados directa o indirectamente con el desarrollo de software. También debe definir el papel de los técnicos, los usuarios y la gestión de la empresa en el proceso de desarrollo. Proporcionar un plan de trabajo, un proceso dinámico e interactivo para el desarrollo estructurado de proyectos, sistemas o software, buscando la calidad y la productividad de los proyectos.

1.5.1 RUP

Rational Unified Process (RUP), es un proceso de desarrollo de software evolutivo. Se utiliza para desarrollar sistemas basados en objetos y/o tecnologías basadas en componentes. Se basa en sólidos principios de ingeniería de software como la toma de un proceso iterativo, impulsado por la arquitectura y enfoque centrado en las necesidades de desarrollo de software.

RUP establece cuatro fases de desarrollo, cada uno de los cuales se organiza en una serie de iteraciones separadas que deben cumplir los criterios definidos antes de que la próxima fase se lleve a cabo.

- Fase inicial: Se define el alcance del proyecto y su modelo de negocio además de identificar los casos de uso del sistema.
- Fase de elaboración: Los desarrolladores analizan las necesidades del proyecto con mayor detalle y definen su creación arquitectónica, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.

- Fase de construcción: Se elabora el diseño de la aplicación y el código fuente. Se obtiene uno o varios *release* o versiones del producto que han pasado las pruebas.
- Fase de transición: Los desarrolladores entregar el sistema a los usuarios. El producto ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

UML define nueve tipos de diagramas: [35]

1. Diagramas de Clase: Los diagramas de Clases son la columna vertebral de casi todos los métodos orientados a objetos, incluyendo UML. Ellos describen la estructura estática de un sistema.
    - Dentro de los diagramas de clases se encuentran los diagramas de Paquetes, los cuales son un subconjunto de diagramas de clases, pero los desarrolladores a veces los tratan como una técnica aparte. Se encargan de organizar los elementos de un sistema en grupos relacionados para minimizar las dependencias entre paquetes.
2. Diagramas de Objetos: Describen la estructura estática de un sistema en un momento determinado. Pueden ser utilizados para probar los diagramas de clases de precisión.
3. Diagramas de Casos de Uso: Muestran la funcionalidad del sistema con los actores y casos de uso.
4. Diagramas de Secuencia: Describen las interacciones entre las clases en términos de un intercambio de mensajes a través del tiempo.
5. Diagramas de colaboración: Representan las interacciones entre objetos como una serie de mensajes secuenciados. Estos describen tanto la estructura estática y el comportamiento dinámico de un sistema.
6. Diagramas de Estado: Describen el comportamiento dinámico de un sistema en respuesta a estímulos externos. Son especialmente útiles en el modelado de objetos reactivos cuyos estados son desencadenados por eventos específicos.
7. Diagramas de Actividad: Ilustran el carácter dinámico de un sistema mediante el modelado del flujo de control de las actividades. Una actividad representa una operación de una clase en el sistema que resulta un cambio en el estado del sistema.
8. Diagramas de Componentes: Describen la organización de los componentes de software, incluyendo el código fuente, en tiempo de ejecución de código (binario), y ejecutables.
9. Diagramas de Implementaciones: Muestran los recursos físicos en un sistema, incluyendo los nodos, componentes y conexiones.

Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

Para el desarrollo del sistema, debido al manejo eficiente y racional de los recursos de memoria, por estar bajo licencias de código abierto y facilitar el trabajo de los desarrolladores, se utilizó RichFaces como biblioteca de componentes web, Hibernate como mapeador objeto-relacional, JBoss 4.2 GA como servidor de aplicaciones web, Eclipse como entorno de desarrollo integrado y Seam como framework de integración. Por otro lado se estudió la usabilidad de las aplicaciones investigadas, con el propósito de especificar patrones y atributos de usabilidad que puedan ser tenidos en cuenta sistemáticamente como factor adicional en la generación dinámica de las interfaces del sistema a desarrollar.

Capítulo 2: Descripción de la Arquitectura

En el presente capítulo se esboza de forma precisa los requerimientos no funcionales utilizados en el sistema a desarrollar, además de una breve explicación de cada uno de ellos. Conjuntamente se hace una descripción de la arquitectura del sistema. Debido a que no existen en el trabajo procesos de negocio en los que tiene lugar la configuración del sistema, se decide desarrollar un Modelo de Dominio. Este abarca las definiciones asociadas a los conceptos encontrados en el entorno donde está enmarcado el sistema, como las relaciones existentes entre ellos.

2.1. Requerimientos no funcionales

Los requerimientos no funcionales hacen relación a las características del sistema que aplican de manera general como un todo, más que a rasgos particulares del mismo. Estos requerimientos son adicionales a los requerimientos funcionales que debe cumplir el sistema, y corresponden a aspectos tales como la disponibilidad, mantenibilidad, flexibilidad, seguridad, facilidad de uso, etc. A diferencia de los requisitos funcionales, estos no modifican o alteran la funcionalidad del producto. Muestran las características que hacen al producto atractivo, usable, rápido o confiable.

Los requerimientos no funcionales se dividen en varias categorías, en este acápite se definen los más usados por la aplicación en desarrollo.

2.1.1. Fiabilidad

- En los servidores que estarán implantados en los centros hospitalarios se garantizará una arquitectura de máxima disponibilidad, tanto de servidores de aplicación como de base de datos. Adicionalmente se garantizarán políticas de respaldo a toda la información, para que no ocurran pérdidas en caso de desastres ajenos al sistema.
- La información médica relacionada con los pacientes y que vayan a ser intercambiadas con otros hospitales por la red pública, viajarán cifradas para evitar accesos o modificaciones no autorizadas.
- Los mecanismos serán capaces de informar al personal autorizado sobre posibles irregularidades que den indicios sobre la introducción de información falseada.
- Se mantendrá seguridad y control a nivel de usuario para garantizar el acceso de los mismos sólo a los niveles establecidos de acuerdo a la función que realizan. Las contraseñas podrán cambiarse solo por el propio usuario o por el administrador del sistema.

- Se llevará un registro de todas las acciones que se realicen y de esta manera llevar el control de las actividades de cada usuario en todo momento.
- Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la BD, independientemente de que para el sistema, este elemento ya no exista. Por lo que el sistema permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

2.1.2. Eficiencia

- El Centro de Datos permitirá agregar recursos para aumentar el poder de procesamiento y almacenamiento sin afectar los sistemas, garantizando expansiones motivadas por futuros requerimientos.
- El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria. Para ello se potenciará como regla guardar en la memoria caché datos y recursos de alta demanda.

2.1.3. Soporte

- Se permitirá la creación de usuarios, otorgamiento de privilegios y roles, asignación de perfiles y activación de permisos por direcciones IP además de la administración remota, monitoreo del funcionamiento del sistema en los centros hospitalarios y detección de fallas de comunicación.
- Se permitirá el chequeo de las operaciones y acceso de los usuarios al sistema. Además de permitir el establecimiento de parámetros de configuración del sistema y actualización de nomencladores.

2.1.4. Requerimientos de hardware

- **Estaciones de trabajo.**
  - En la solución se incluyen estaciones de trabajo para las consultas del Sistema de Información Hospitalaria alas HIS, las que necesitan capacidad de hardware que soporte un sistema operativo que cuente con un navegador actualizado y que siga los estándares web, se recomienda Internet Explorer 7, Firefox 2 o versiones superiores. Por lo que se escogieron estaciones de trabajo de 256 Mb de memoria RAM y un microprocesador de 2.0 Hz con sistema operativo Linux.

- **Servidores.**
  - La solución estará conformada, fundamentalmente, por servidores de alta capacidad de procesamiento y redundancia, que permitan garantizar movilidad y residencia de la información y las aplicaciones bajo esquemas seguros y confiables. Servidores de Base de datos: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 4GB de memoria y 2x72GB de disco y sistema operativo Linux. Servidores de Aplicaciones: 2 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 4GB de memoria y 2x72GB de disco y sistema operativo Linux. Servidores de Intercambio: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 2 GB de memoria y 2x72GB de disco y sistema operativo Linux.

2.1.5. Software

El sistema debe correr en sistemas operativos Windows, Unix y Linux y utilizar la plataforma JAVA (Java Virtual Machine, JBoss AS y Postgre SQL). El sistema deberá disponer de un navegador web, estos pueden ser Internet Explorer 7, Opera 9, Google chrome 1 y Firefox 2 o versiones superiores de estos.

2.1.6. Portabilidad

- El producto podrá ser utilizado bajo los sistemas operativos Linux o Windows.

2.1.7. Interfaz

- **Interfaces de usuario**
  - Las ventanas del sistema contendrán claro y bien estructurados los datos, además de permitir la interpretación correcta de la información.
  - La interfaz contará con accesos directos y menús desplegables que faciliten y aceleren su utilización.
  - La entrada de datos incorrecta será detectada claramente e informada al usuario.
  - Todos los textos y mensajes en pantalla aparecerán en idioma español.

- **Interfaces de comunicación**
  - Para el intercambio electrónico de datos entre aplicaciones se usará el estándar HL7 (Health Level Seven).
  - El sistema usará el formato estándar WSDL para la descripción de los servicios web.

- El sistema implementará mecanismos de encriptación de datos para el intercambio de información con sistemas externos.
- El sistema utilizará mecanismos de compactación de los datos que se intercambiarán con sistemas externos con el objetivo de minimizar el tráfico en la red y economizar el ancho de banda.

2.2. Descripción de la arquitectura

A continuación se describirá el diseño del sistema. Una primera impresión permite definir dos partes claramente diferenciadas: por un lado se dispone de un conjunto de clases que definen la interfaz de comunicación con el usuario; y por otro una colección de clases de gestión. Los patrones arquitectónicos permiten definir la estructura de un sistema de software, por lo que es importante la búsqueda del que mejor se adapte a los requerimientos del sistema.

La arquitectura de software es el conjunto de decisiones de diseño que, si son tomadas de forma equivocada, pueden provocar que el proyecto sea cancelado. Representa básicamente la estructura de un sistema y comprende los principales componentes del mismo, sus propiedades externamente visibles y las relaciones entre ellos. Proporciona conceptos y un lenguaje común que permite la comunicación entre los equipos que participen en un proyecto.

Para el desarrollo del sistema, teniendo en cuenta las herramientas, tecnologías y metodologías propuestas, se define como parte de la línea base de la arquitectura el patrón de diseño Modelo Vista Controlador (MVC), muy usado en aplicaciones web. Este patrón separa los datos de una aplicación, la interfaz de usuario y la lógica de control, en tres componentes distintos: el modelo, que abarca los datos y las reglas del negocio; la vista, que muestra la información del modelo al usuario; y el controlador, que gestiona las entradas del usuario. [36]

Desde el punto de vista de la arquitectura de la aplicación y con el objetivo de lograr una mejor organización de los componentes que conforman el diseño, se agruparon dichos componentes en tres capas fundamentales: presentación, negocio y acceso a datos. Esto brinda una mejor organización según la función que realizan, permitiendo que en un momento determinado un elemento de una capa pueda ser modificado o sustituido completamente causando el mínimo de alteraciones en otro elemento que lo utilice.

La capa de presentación está formada por páginas XHTML compuestas por controles JSF y la librería de componentes Richfaces, que se complementa fácilmente con el framework de

integración Seam. Mediante estos componentes se muestran los datos haciendo uso del componente Ajax4jsf y se validan los datos que el usuario provee en cada una de las operaciones que efectúa. Además al realizar el envío y carga de datos mediante los componentes ajax4sf se logra un efecto más agradable y natural al interactuar con el sistema.

La capa de negocio está compuesta por clases controladoras, que encierran la lógica del negocio del módulo, a las cuales, mediante anotaciones que provee el framework Seam, se les especifica el contexto en que se encuentran (conversacional, evento, página, etc.), los cuales definen el estado de los datos y las entidades que manipulan. En esta capa se tratan también las reglas del negocio, haciendo uso de Drools para proporcionarle mayor dinamismo y funcionalidad al sistema.

La capa de acceso a datos funciona como puente entre la capa lógica de negocio y el proveedor de datos, realiza el manejo de los mismos mediante Hibernate, lo cual permite seleccionar, modificar, eliminar y persistir la información en la base de datos a través de un manejador de entidades; además de que incorporan validación de campos. Esta estructura permite llevar las consultas a un lenguaje orientado a objeto.

2.3. Modelo de Dominio

El Modelo de Dominio o Modelo Conceptual, es un artefacto construido con las reglas de UML durante la fase de concepción. Presenta uno o más diagramas de clases y 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. Son utilizados por el analista como un medio para comprender el sector industrial o de negocios donde el sistema va a servir. [37]

En el presente modelo de dominio se trata de obtener un esquema conceptual de la base de datos a partir de la lista descriptiva de objetos y asociaciones identificadas en la organización durante el análisis.

2.3.1. Conceptos fundamentales del dominio

Con el objetivo de asegurar una mejor comprensión del Diagrama del Modelo de Dominio a continuación se realiza una breve descripción de cada uno de los conceptos encontrados en el ámbito del problema. Estos son:

Config_Ficha: Es toda la información referente a la configuración de la Ficha de Investigación Epidemiológica.

Config_Sección: Es la configuración de la sección a generar dinámicamente. Estas secciones pertenecen a la configuración de la Ficha de Investigación Epidemiológica.

Config_Campo: Es la configuración de los campos de cada sección a configurar. Incluye el nombre del campo y el tipo de dato al que pertenece.

Notificación: Notificación de una Enfermedad de Notificación Obligatoria (ENO) padecida por un paciente.

Diagnóstico: Representa el diagnóstico por el cual se va a crear un tipo determinado de ficha.

Tipo de Dato: Es una agrupación de tipos de datos para los campos que se crean en cada una de las secciones.

Datos_Ficha: Son los datos que se corresponden con una configuración de Ficha ya creada anteriormente.

Datos_Sección: Almacena todos los valores de los campos configurados.

Datos_Campos: Datos correspondientes a los campos configurados.

Sección_Síntoma: Sección predeterminada Síntoma de la Ficha.

Sección_Antecedentes Vacunales: Sección predeterminada Antecedentes Vacunales de la Ficha.

Sección_Contacto: Sección predeterminada Contacto de la Ficha.

Sección_Diagnóstico: Sección predeterminada Diagnóstico de la Ficha.

Sección_Factores de Riesgo: Sección predeterminada Factores de Riesgo de la Ficha.

Sección_Datos de Laboratorio: Sección predeterminada Datos de Laboratorio de la Ficha.

Sección_Datos Epidemiológicos: Sección predeterminada Datos Epidemiológicos de la Ficha.

2.3.2. Diagrama del Modelo de Dominio

Diagrama del Modelo de Dominio

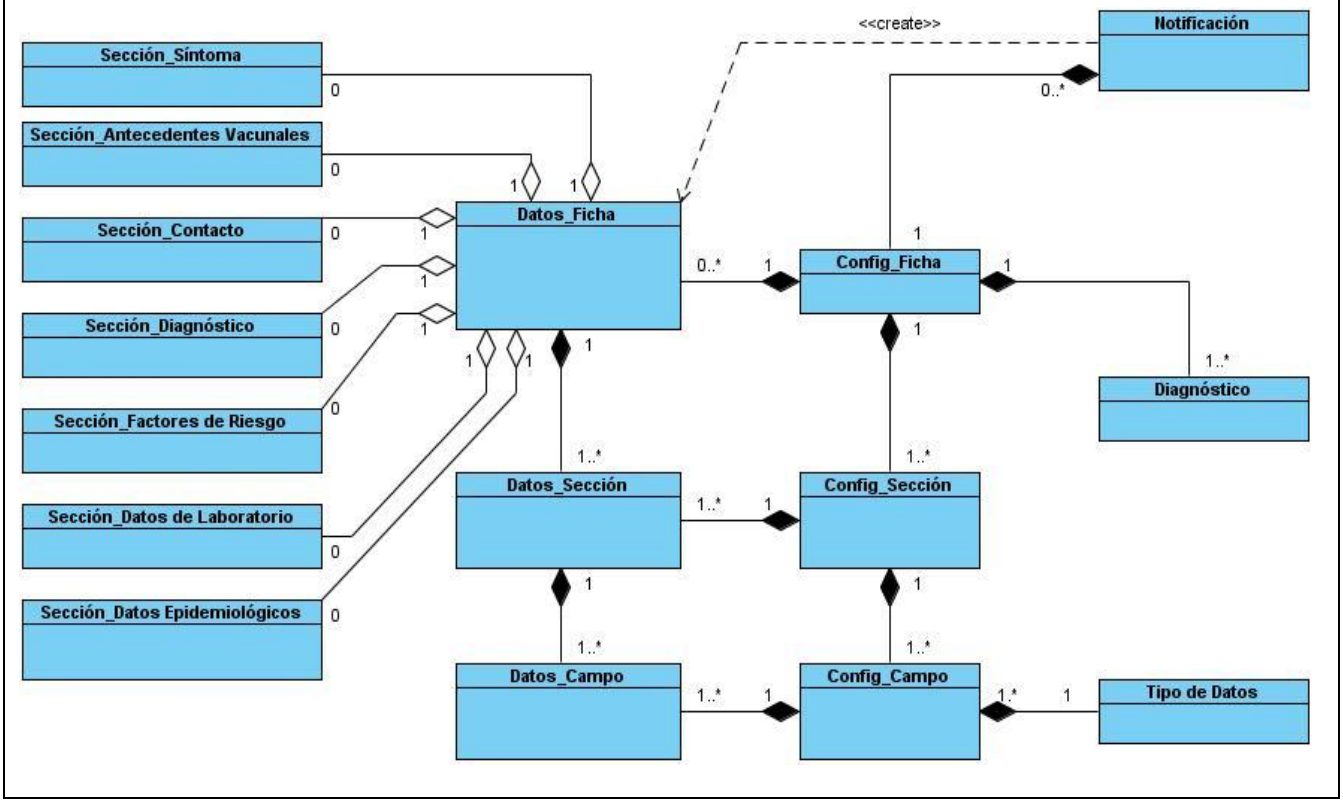

Figura 2.3.1. Diagrama del Modelo de Dominio

2.4. Seguridad

El sistema debe ser extremadamente seguro debido a que en el mismo se maneja información delicada de los pacientes, donde solo puede acceder el personal autorizado. En el mismo se define un control de acceso a nivel de usuarios, lo que permite que cada usuario pueda acceder y hacer cambios según el nivel de acceso que tenga, además de que solo el propio usuario y el administrador del sistema pueden cambiar su contraseña.

La autorización de la aplicación está basada en reglas lo que restringe el acceso a directorios, opciones del menú, controles, estas reglas no se encuentran en el código de la aplicación lo que le provee un mayor dinamismo al sistema puesto que en el caso de que cambie alguna no es necesario una nueva compilación. Esto es posible debido a la integración existente entre el motor de reglas JBoss Rules y el Framework de Seguridad de JBoss Seam. Además se cuenta con una

bitácora en la que se guardan todas las operaciones realizadas por el usuario, por lo que queda registrada la fecha, la hora y la actividad que llevó a cabo el usuario.

2.5. Vista de Despliegue

La vista de despliegue define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos hardware sobre los cuales pueden ejecutarse los elementos software. Un nodo es un recurso de ejecución tales como servidores, computadoras, además de dispositivos tales como impresoras y scanners. Siendo el diagrama de despliegue una parte fundamental de esta vista. Es un tipo de diagrama del Lenguaje Unificado de Modelado, que se emplea para modelar el hardware en nodos físicos donde el sistema debe ejecutarse. [38]

La mayoría de las veces el modelado de la vista de despliegue implica modelar la topología del hardware sobre el que se ejecuta el sistema. Aunque UML no es un lenguaje de especificación de hardware de propósito general, se ha diseñado para modelar muchos de los aspectos del hardware de un sistema, a un nivel suficiente para que un ingeniero de software, pueda especificar la plataforma sobre la que se ejecutará el sistema.

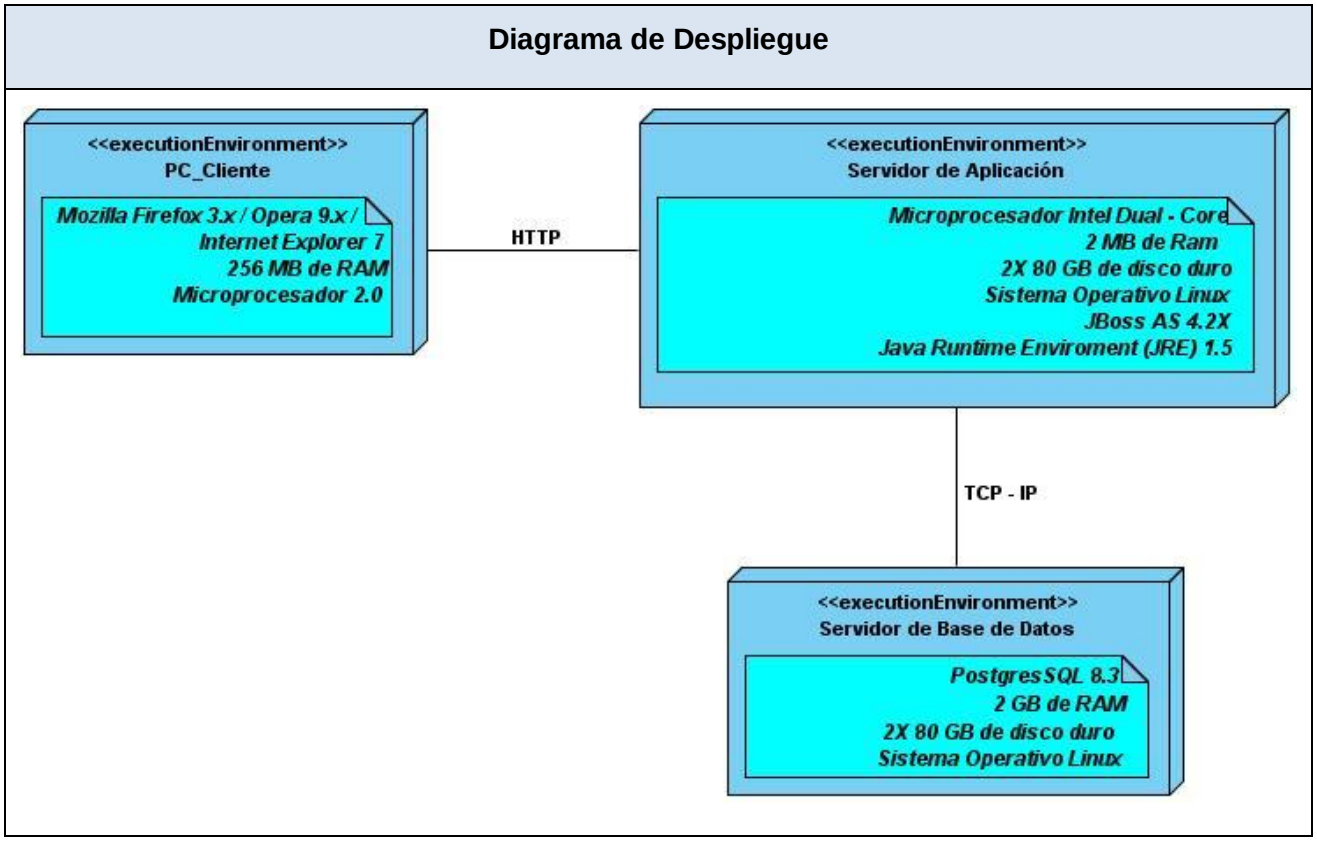

Figura 2.5.1. Diagrama de Despliegue

2.6. Estrategias de codificación. Estándares y estilos a utilizar.

Los estándares de codificación son reglas que se deben seguir con el objetivo de lograr un mayor entendimiento del código fuente. Estos facilitan que a los sistemas se les pueda dar mantenimiento en un futuro ya que ayuda a la comprensión del código, además de que posibilita una mayor organización y limpieza del código. Entre los estándares utilizados en el sistema se encuentran:

Se recomienda dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función. Se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código.

Ejemplo: campo = nomCampo

Se debe evitar comentar cada línea de código. Cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se escribe a continuación de la instrucción. El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, la cual identificará el tipo de datos al que se refiere en caso de que sea un nombre compuesto se empleará notación CamellCasing*.

Ejemplo: sNombreSeccion

El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma. Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing*.

Ejemplo: MiCampo ()

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

Ejemplo: sTipoCampo.

Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que hace la función. Se empleará notación CamellCasing*.

Ejemplo: function buscarFicha ().

Si son funciones que obtienen un dato se emplea el prefijo get y si fijan algún valor se emplea el prefijo set. El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.

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

(*) Notación PascalCasing: Las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto cada palabra debe comenzar con mayúscula. Ejemplo: NotacionPascalCasing.

2.7. Propuesta del Sistema

Con el propósito de lograr una interfaz de fácil aprendizaje y que agilice el tratamiento de las Fichas de Investigación Epidemiológica se propone una herramienta de apoyo a la creación y gestión de las mismas que ofrezca un conjunto dinámico de componentes y una colección de interfaces variables. De forma tal que las fichas se puedan crear dinámicamente partiendo del criterio de diseño de un usuario. Esta propuesta reportará un conjunto de beneficios significativos. Entre estos beneficios se citan:

- No se perderá tiempo a la hora de crear una nueva ficha.
- Las fichas las podrán crear los propios usuarios.
- El servidor no tendrá que reiniciarse debido a la creación de nuevas fichas.
- La aplicación no dejará de funcionar en la entidad que este brindando sus servicios.

Para ello, surge la idea de amoldar un diseño dinámico en la que se pueda crear Fichas de Investigación Epidemiológicas en la que sus interfaces se adapten en la medida de lo posible a las características de cada epidemiólogo. El cual definirá una estructura en la que se determinan:

- Los tipos de secciones de la nueva ficha. Se reduce de esta forma el conjunto de componentes, a los imprescindibles en la ficha.
- Los campos que el epidemiólogo deberá seleccionar para completar la definición de una sección. Este conjunto de campos, serán específicos de cada tipo de ficha.
- Las características de cada uno de estos campos, en relación al tipo de dato (por ejemplo cadena de texto, número…), si es requerido o no. Esta especificación permitirá posteriormente generar una interfaz mejor adaptada.

De esta forma, la visualización de la aplicación variará en función de las especificaciones realizadas en la definición de la estructura. La interfaz gráfica de comunicación con el usuario se adaptará a cada ficha creada e incluso a las costumbres o técnicas de trabajo de cada especialista. (Ver Anexo 1)

En el presente capítulo, se profundizó en la seguridad del sistema, la vista de despliegue a utilizar, el modelo de dominio, se enfatizó en los conceptos básicos así como en su diagrama. Además de las estrategias de codificación, estándares y estilos a utilizar. Con el objetivo de que el sistema que se obtenga como resultado de la presente investigación, sea una aplicación robusta y confiable.

Capítulo 3: Descripción y análisis de la solución propuesta

En este capítulo se describe el sistema así como sus características, explicando los procedimientos y funcionalidades con que constará la solución propuesta. Se aborda el análisis y diseño de la aplicación haciendo énfasis en la descripción del modelo de datos y la vista de implementación.

3.1. Valoración crítica del diseño propuesto por el analista

Cuando se desea diseñar un sistema de computación se debe tener en cuenta que el proceso de un diseño incluye, pensar y concebir algo así como hacer un dibujo, modelo o croquis. El diseño en el desarrollo de un software interpreta los requisitos del sistema de forma tal que se pueda lograr una descripción de cómo se implementará, o sea, el objetivo del diseño es indicar como el sistema será generado en la fase de implementación. Además, se crea un modelo de diseño y opcionalmente un modelo de análisis. El modelo de diseño contiene clases estructuradas dentro de un paquete de diseño y un subsistema de diseño con un conjunto de interfaces bien definidas; representando lo que serán los componentes de la aplicación; también contiene descriptores de cómo los objetos del diseño de clases, colaboran para interpretar los casos de uso. [39]

El modelo de diseño ofrece la posibilidad de descomponer los trabajos de implementación en componentes más manejables, visualizándolos mediante una notación común. Contiene los diagramas, las clases, paquetes, subsistemas, cápsulas, protocolos, interfaces, relaciones, colaboraciones, atributos y las realizaciones de los casos de uso. El resultado del modelo de diseño son especificaciones muy detalladas de todos los objetos, incluyendo sus operaciones y atributos.

Para construir el diseño se utiliza un grupo de patrones o modelos para lograr los objetivos esperados. En el caso de la presente investigación se tuvieron en cuenta los patrones GRASP. GRASP es el acrónimo de General Responsability Asignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos.

Los patrones GRASP constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objetos esencial, y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Este enfoque para la comprensión y utilización de los principios de diseño se basa en los patrones de asignación de responsabilidades, ya que se encargan de asignar responsabilidades

a las diferentes clases que se definen en el Diseño. Dentro de este grupo de patrones se pueden mencionar: experto, creador, alta cohesión, bajo acoplamiento y controlador. [40]

Entre los elementos que componen al modelo de diseño se encuentran los diagramas de clases de diseño y los diagramas de interacción. Los diagramas de clases presentan las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. Los diagramas de secuencia y los diagramas de colaboración son llamados diagramas de interacción que se usan para modelar aspectos dinámicos de un sistema. Un diagrama de interacción muestra una interacción, que consta de un conjunto de objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos. Los diagramas de interacción cubren la vista dinámica de un sistema. [41]

Un diagrama de Secuencia muestra una interacción de manera organizada en dependencia de la secuencia de los eventos. En particular, define los objetos que participan en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. Un Diagrama de Colaboración muestra una interacción ordenadamente de los objetos que toman parte en la interacción y los vínculos entre los mismos. A diferencia de los Diagramas de Secuencia, los Diagramas de Colaboración muestran las relaciones entre los roles de los objetos. En la etapa de Diseño, el diagrama más utilizado es el de Secuencia, pues el Diagrama de Colaboración es utilizado mayormente en la fase de Análisis.

El Diagrama de Paquetes, también utilizado en este trabajo, permite dividir el sistema en partes manejables para su futura implementación. Estos tipos de diagramas muestran como el sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Los Diagramas de Paquetes suministran una descomposición de la jerarquía lógica de un sistema además de estar normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes. [42]

A continuación se muestra el Diagrama de Paquetes, el Diagrama de Clases del Diseño y el Diagrama de Secuencia respectivamente del sistema propuesto:

Diagrama de Paquetes

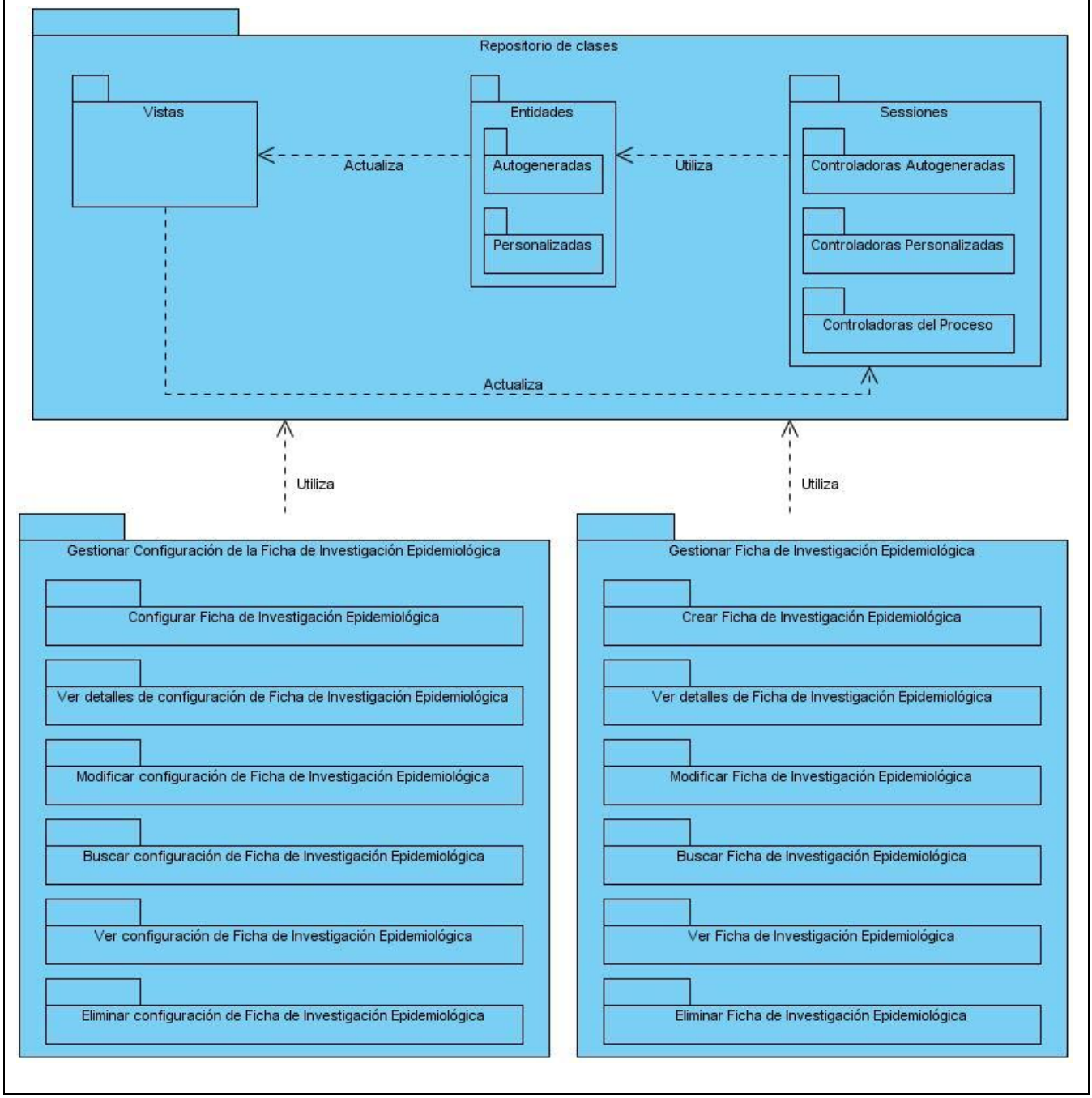

Figura 3.1.1 Diagrama de Paquetes

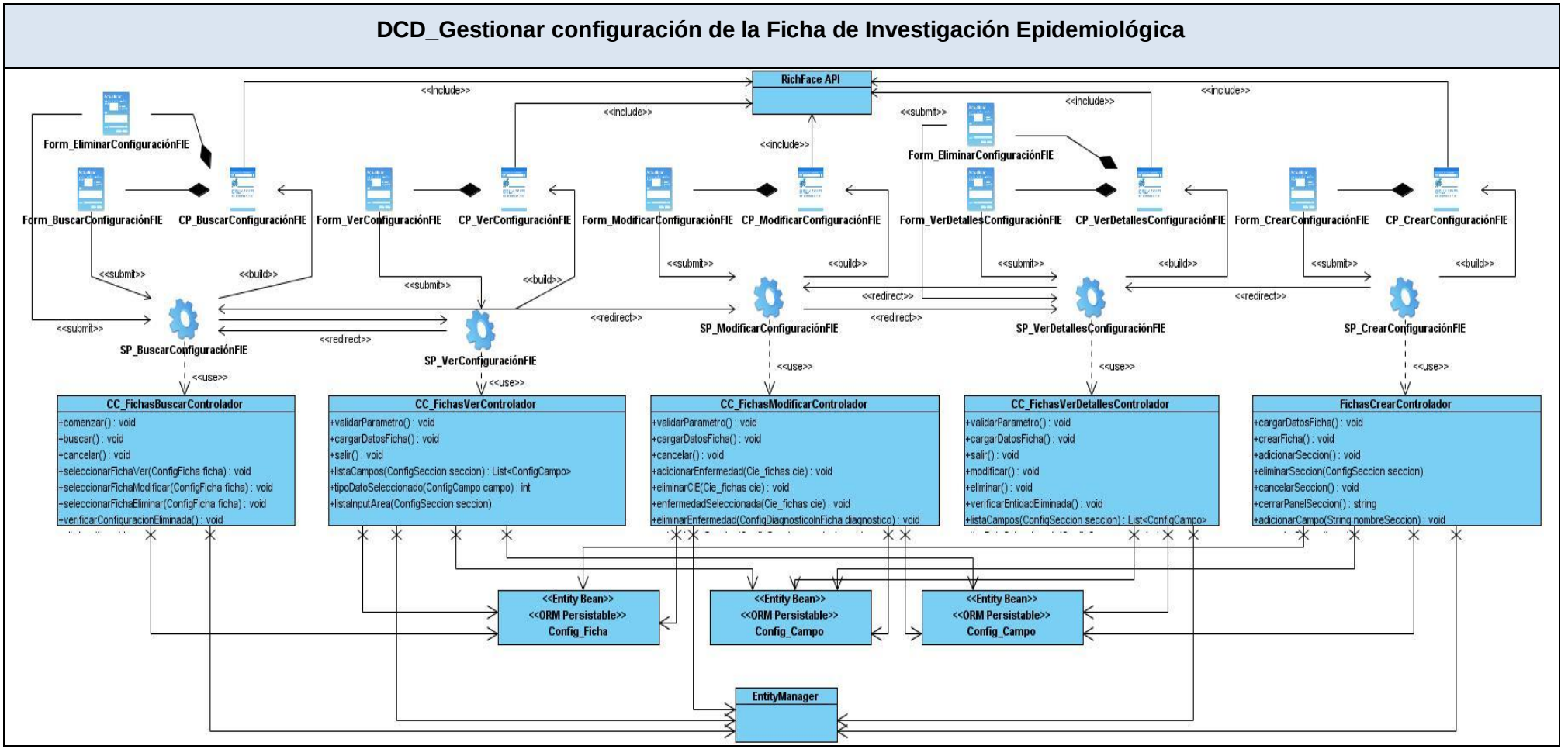

Figura 3.1.2. Diagrama de Clases del Diseño – Gestionar configuración de Ficha de Investigación Epidemiológica.

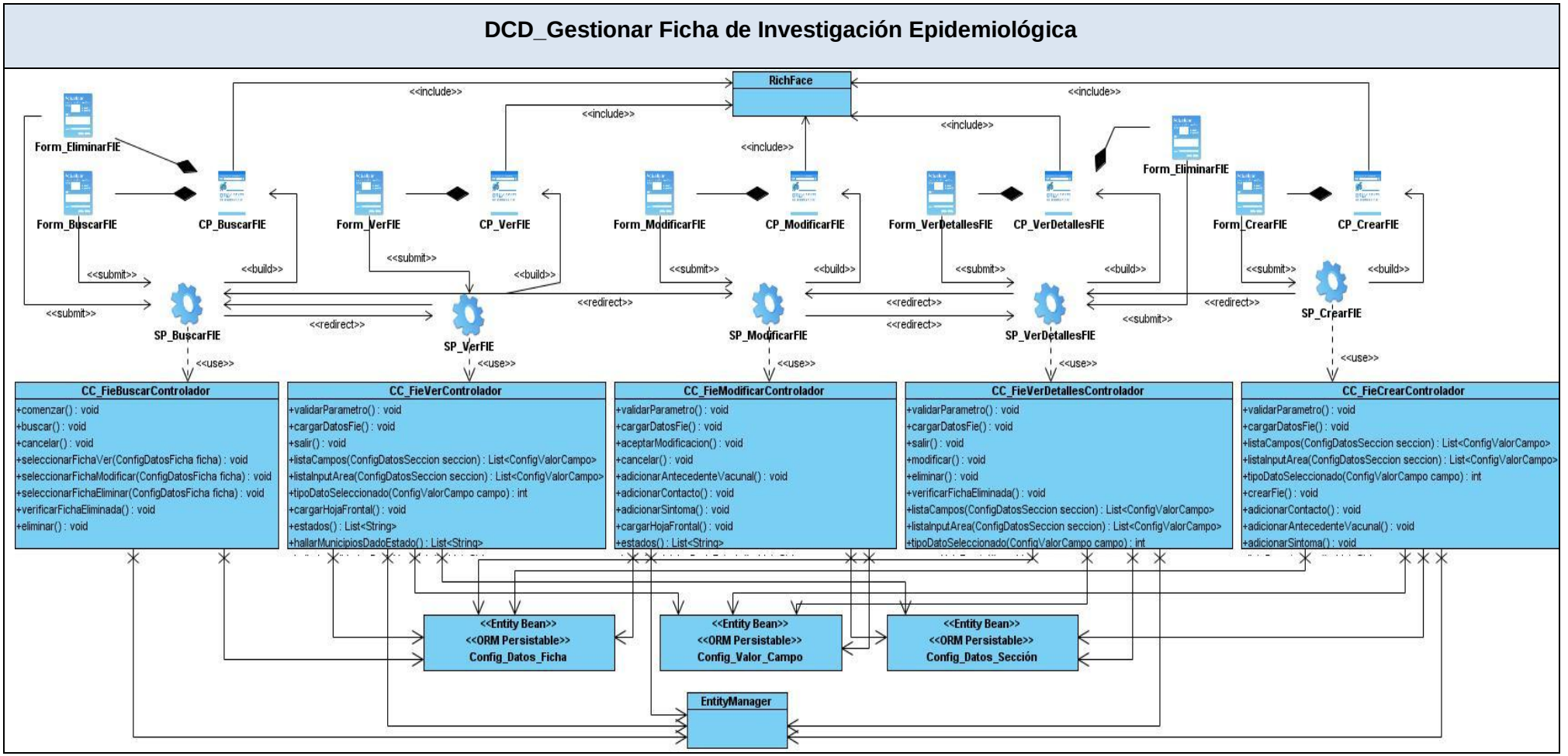

Figura 3.1.3. Diagrama de Clases del Diseño – Gestionar Ficha de Investigación Epidemiológica.

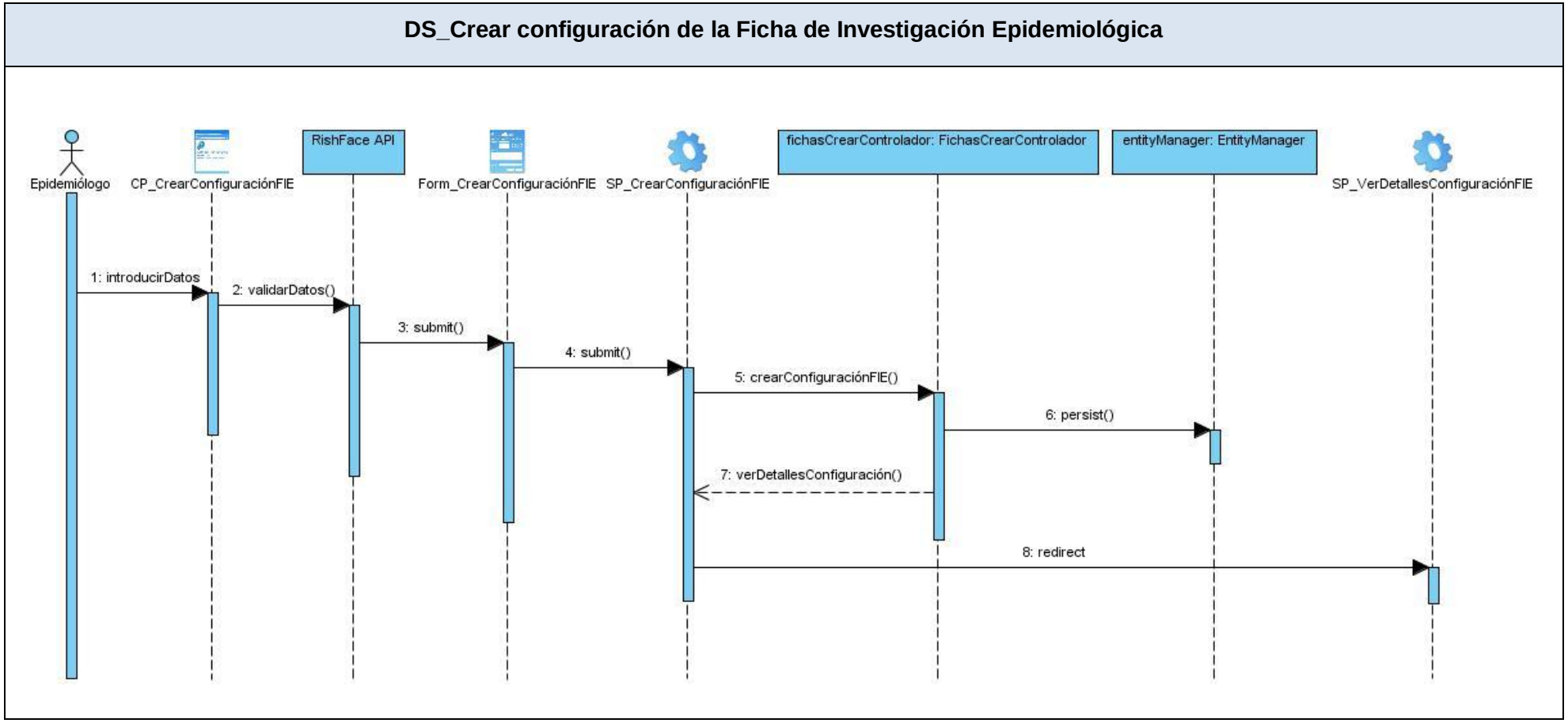

Figura 3.1.4. Diagrama de Secuencia – Crear configuración de la Ficha de Investigación Epidemiológica.

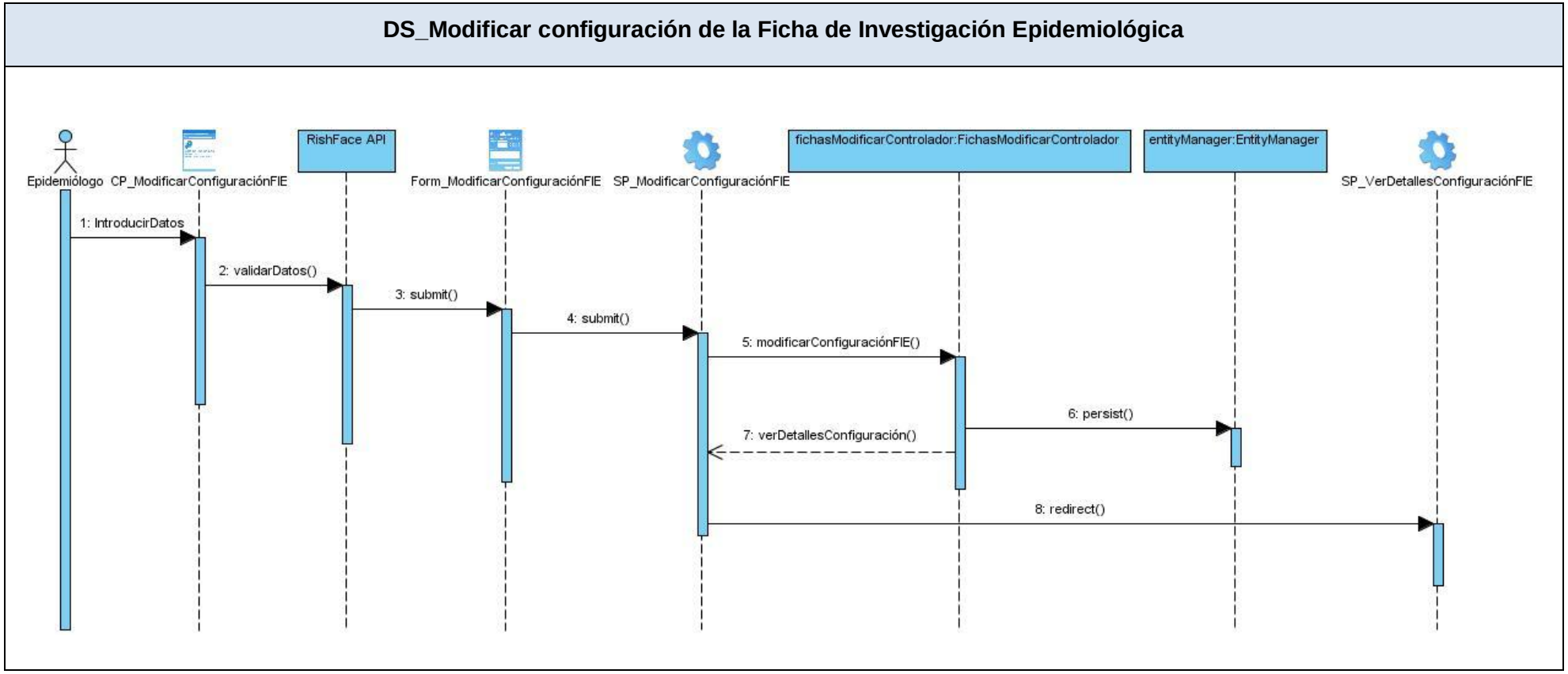

Figura 3.1.5. Diagrama de Secuencia – Modificar configuración de la Ficha de Investigación Epidemiológica.

DS_Ver configuración de la Ficha de Investigación Epidemiológica

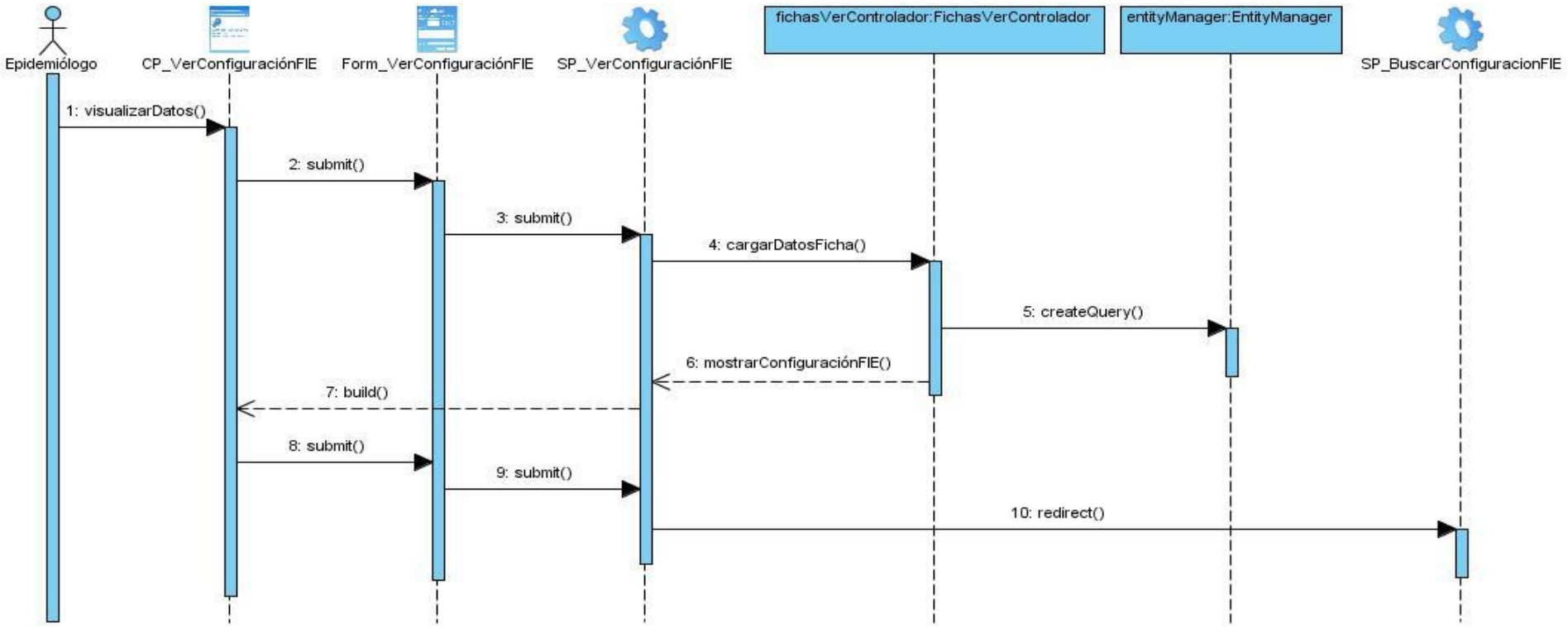

Figura 3.1.6. Diagrama de Secuencia – Ver configuración de la Ficha de Investigación Epidemiológica.

DS_Ver Detalles configuración de la Ficha de Investigación Epidemiológica

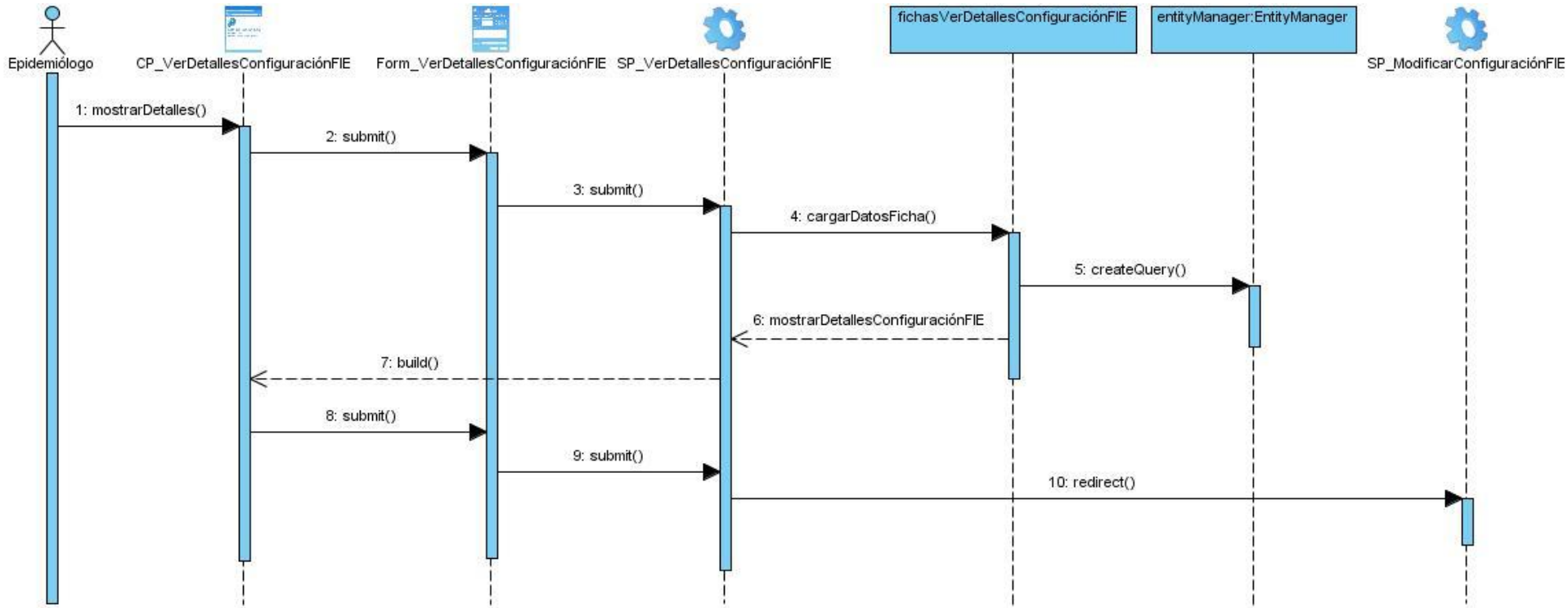

Figura 3.1.7. Diagrama de Secuencia – Ver Detalles configuración de la Ficha de Investigación Epidemiológica.

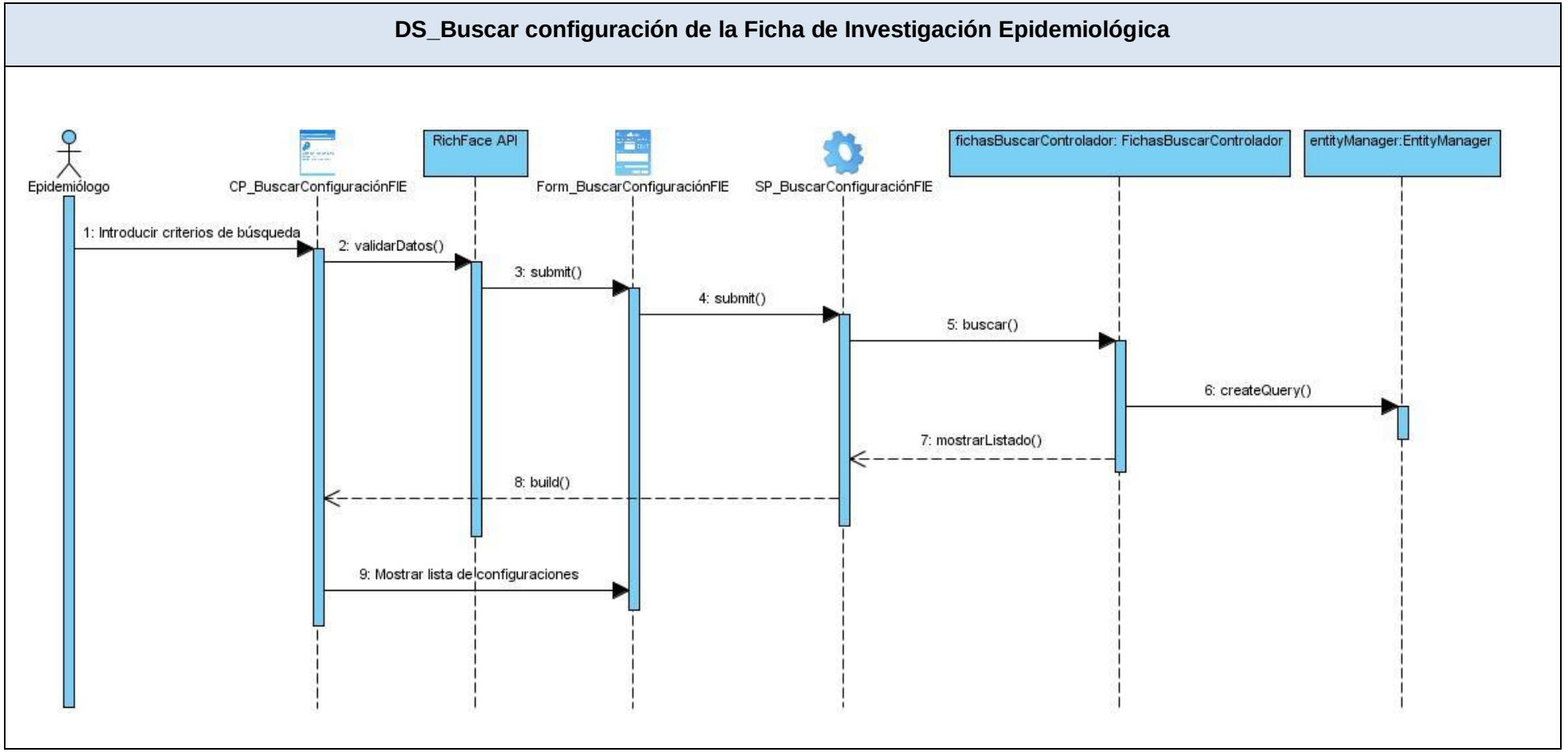

Figura 3.1.8. Diagrama de Secuencia – Buscar configuración de la Ficha de Investigación Epidemiológica.

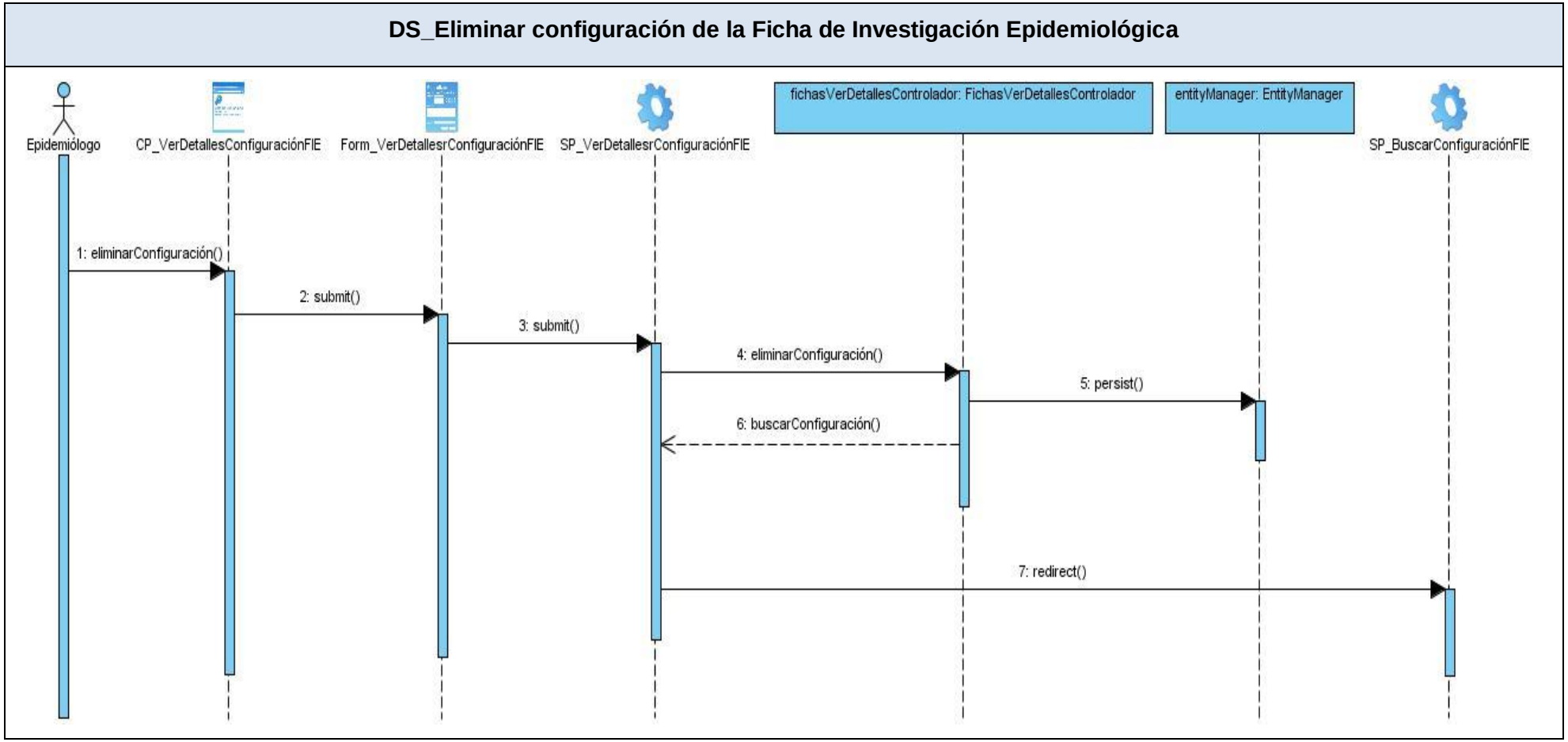

Figura 3.1.9. Diagrama de Secuencia – Eliminar configuración de la Ficha de Investigación Epidemiológica.

3.2. Descripción de las nuevas clases u operaciones necesarias

Nombre: FichasCrearControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
cid	Integer
idFicha	Integer
problemasPersistencia	Boolean
ficha	ConfigFicha_configuracionep
listaDiagnosticos	List<ConfigDiagnosticoInFicha_configuracionep>
diagnostico	ConfigDiagnosticoInFicha_configuracionep
tiposSecciones	List<String>
seccionesSeleccionadas	List<String>
adicionSeccion	Boolean
seccionEliminada	Boolean
seccionRepetida	Boolean
seccion	ConfigSeccion_configuracionep
listaSecciones	List<ConfigSeccion_configuracionep>
campo	ConfigCampo_configuracionep
tipoDato	String
tipoDatos	List<ConfigTipoDato_configuracionep>
adicionCampo	Boolean
nombreSeccionConfigurar	String
Para cada responsabilidad:	
Nombre:	cargarDatosFicha
Descripción:	Carga los datos de la ficha
Nombre:	adicionarSeccion
Descripción:	Adiciona una sección a la ficha
Nombre:	eliminarSeccion(ConfigSeccion_configuracionep seccion)
Descripción:	Elimina una sección de la ficha
Nombre:	cancelarSeccion
Descripción:	Cancela la adición de una sección de la ficha
Nombre:	cerrarPanelSeccion
Descripción:	Indica si se puede cerrar el modal panel de la adición de la sección

Nombre:	adicionarCampo(String nombreSeccion)
Descripción:	Adiciona un campo a la sección creada
Nombre:	cancelarCampo
Descripción:	Cancela la adición de un campo
Nombre:	cerrarPanelCampo
Descripción:	Indica si se puede cerrar el modal panel de la adición del campo
Nombre:	eliminarCampo(ConfigCampo_configuracionep campo,ConfigSeccion_configuracionep seccion)
Descripción:	Dada una sección, elimina un campo
Nombre:	DevolverNombre(String nombreSecc)
Descripción:	Devolver el nombre de la sección a configurar
Nombre:	crearFicha
Descripción:	Salva la configuración en la Base de Datos
Nombre:	listaTipoDatos
Descripción:	Devuelve la lista de tipos de datos
Nombre:	adicionarEnfermedad(Cie_fichas cie)
Descripción:	Adiciona una enfermedad desde la tabla cie
Nombre:	eliminarCie(Cie_fichas cie)
Descripción:	Elimina una enfermedad desde la tabla cie
Nombre:	enfermedadSeleccionada(Cie_fichas cie)
Descripción:	Indica si una enfermedad esta seleccionada o no
Nombre:	eliminarEnfermedad(ConfigDiagnosticoInFicha_configuracionep diagnostico)
Descripción:	Elimina una enfermedad desde la tabla de las enfermedades

Tabla 3.2.1 FichasCrearControlador

Nombre: FichasModificarControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
cid	Integer
idFicha	Integer
problemasPersistencia	Boolean
parametroValido	Boolean
parametro	String
entrada	String
listaDiagnosticos	List<ConfigDiagnosticoInFicha_configuracionep>

diagnosticosEliminados	List<ConfigDiagnosticoInFicha_configuracionep>
diagnostico	ConfigDiagnosticoInFicha_configuracionep
tiposSecciones	List<String>
seccionesSeleccionadas	List<String>
seccionesSeleccionadasEliminadas	List<String>
adicionSeccion	Boolean
seccionEliminada	Boolean
seccionRepetida	Boolean
seccion	ConfigSeccion_configuracionep
listaSecciones	List<ConfigSeccion_configuracionep>
seccionesEliminadas	List<ConfigSeccion_configuracionep>
camposEliminados	List<ConfigCampo_configuracionep>
campo	ConfigCampo_configuracionep
tipoDato	String
tipoDatos	List<ConfigTipoDato_configuracionep>
adicionCampo	Boolean
nombreSeccionConfigurar	String
Para cada responsabilidad:	
Nombre:	validarParametro
Descripción:	Valida que el parámetro sea el correcto
Nombre:	cargarDatos
Descripción:	Carga los datos de la ficha
Nombre:	aceptarModificacion
Descripción:	Modifica los objetos en la Base de Datos
Nombre:	cancelar
Descripción:	Cancela la acción de Modificar
Nombre:	adicionarEnfermedad(Cie_fichas cie)
Descripción:	Adiciona una enfermedad desde la tabla cie
Nombre:	eliminarCie(Cie_fichas cie)
Descripción:	Elimina una enfermedad desde la tabla cie
Nombre:	enfermedadSeleccionada(Cie_fichas cie)
Descripción:	Indica si una enfermedad esta seleccionada o no
Nombre:	eliminarEnfermedad(ConfigDiagnosticoInFicha_configuracionep diagnostico)
Descripción:	Elimina una enfermedad desde la tabla de las enfermedades

Nombre:	adicionarSeccion
Descripción:	Adiciona una sección a la ficha
Nombre:	eliminarSeccion(ConfigSeccion_configuracionep seccion)
Descripción:	Elimina una sección a la ficha
Nombre:	cancelarSeccion
Descripción:	Cancela la adición de una sección
Nombre:	cerrarPanelSeccion
Descripción:	Indica si se puede cerrar el modal panel de la adición de la sección
Nombre:	adicionarCampo
Descripción:	Adiciona un campo a la sección indicada
Nombre:	cancelarCampo
Descripción:	Cancela la adición de un campo
Nombre:	cerrarPanelCampo
Descripción:	Indica si se puede cerrar el modal panel de la adición del campo
Nombre:	eliminarCampo(ConfigCampo_configuracionep campo,ConfigSeccion_configuracionep seccion)
Descripción:	Dado una sección, elimina un campo determinado
Nombre:	listaTipoDatos
Descripción:	Devuelve la lista de tipos de datos

Tabla 3.2.2 FichasModificarControlador

Nombre: FichasVerControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
cid	Integer
idFicha	Integer
problemasPersistencia	Boolean
parametroValido	Boolean
entidadEliminada	Boolean
parametro	String
entrada	String
listaSecciones	List<ConfigSeccion_configuracionep>
listaCampos	List<ConfigCampo_configuracionep>
Para cada responsabilidad:	

Nombre:	validarParametro
Descripción:	Valida que el parámetro sea el correcto
Nombre:	cargarDatosFicha
Descripción:	Carga los datos de la ficha
Nombre:	salir
Descripción:	Sale de la página Ver
Nombre:	listaCampos(ConfigSeccion_configuracionep seccion)
Descripción:	Dada una sección devolver una lista de campos, ordenando los CheckBox
Nombre:	tipoDatoSeleccionado(ConfigCampo_configuracionep campo)
Descripción:	Devuelve un número que indica el tipo de dato seleccionado
Nombre:	listaInputArea(ConfigSeccion_configuracionep seccion)
Descripción:	Dada una sección devolver una lista de campos de tipo InputTextArea

Tabla 3.2.3 FichasVerControlador

Nombre: FichasVerDetallesControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
cid	Integer
idFicha	Integer
problemasPersistencia	Boolean
parametroValido	Boolean
entidadEliminada	Boolean
parametro	String
entrada	String
listaSecciones	List<ConfigSeccion_configuracionep>
listaCampos	List<ConfigCampo_configuracionep>
Para cada responsabilidad:	
Nombre:	validarParametro
Descripción:	Valida que el parámetro sea el correcto
Nombre:	cargarDatosFicha
Descripción:	Carga los datos de la ficha
Nombre:	salir
Descripción:	Sale de la página Ver Detalles
Nombre:	modificar

Descripción:	Va a la página Modificar
Nombre:	eliminar
Descripción:	Elimina la entidad
Nombre:	verificarEntidadEliminada
Descripción:	Verifica si la entidad está eliminada
Nombre:	listaCampos(ConfigSeccion_configuracionep seccion)
Descripción:	Dada una sección devolver una lista de campos, ordenando los CheckBox
Nombre:	tipoDatoSeleccionado(ConfigCampo_configuracionep campo)
Descripción:	Devuelve un número que indica el tipo de dato seleccionado
Nombre:	listaInputArea(ConfigSeccion_configuracionep seccion)
Descripción:	Dada una sección devolver una lista de campos de tipo InputTextArea

Tabla 3.2.4 FichasVerDetallesControlador

Nombre: FichasBuscarControlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
cid	Integer
busquedaIniciada	Boolean
configuracionEliminada	Boolean
problemasPersistencia	Boolean
ficha	ConfigFicha_configuracionep
Para cada responsabilidad:	
Nombre:	comenzar
Descripción:	Carga todos los datos de la página
Nombre:	buscar
Descripción:	Busca según el criterio de búsqueda
Nombre:	cancelar
Descripción:	Sale de la página Buscar
Nombre:	seleccionarFichaVer(ConfigFicha_configuracionep ficha)
Descripción:	Selecciona una ficha para ver
Nombre:	seleccionarFichaModificar(ConfigFicha_configuracionep ficha)
Descripción:	Selecciona una ficha para modificar
Nombre:	seleccionarFichaEliminar(ConfigFicha_configuracionep ficha)
Descripción:	Selecciona una ficha para eliminar

Nombre:	verificarConfiguracionEliminada
Descripción:	Verifica si la configuración está eliminada
Nombre:	eliminar
Descripción:	Elimina la entidad

Tabla 3.2.5 FichasCrearControlador

3.3. Modelo de Datos

El diseño de bases de datos es el proceso por el que se determina la organización de una base de datos, incluidos su estructura, contenido y las aplicaciones que se han de desarrollar. Durante mucho tiempo, el diseño de bases de datos fue considerado una tarea para expertos: más un arte que una ciencia. Sin embargo, se ha progresado mucho en el diseño de bases de datos y éste se considera ahora una disciplina estable, con métodos y técnicas propios. [43]

El diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico. El diseño conceptual parte de las especificaciones de requisitos de usuario y su resultado es el esquema conceptual de la base de datos. Su objetivo principal es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información. El diseño lógico parte del esquema conceptual y da como resultado un esquema lógico, es un lenguaje usado para especificar esquemas lógicos (modelo relacional, modelo de red, etc.). El diseño físico parte del esquema lógico y da como resultado un esquema físico, es una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. [43]

Con respecto al diseño de bases de datos, el modelado de datos puede considerarse como los requerimientos de información y proceso de una aplicación de uso intensivo de datos (por ejemplo, un sistema de información), construir una representación de la aplicación que capture las propiedades estáticas y dinámicas requeridas para dar soporte a los procesos. Además de capturar las necesidades dadas en el momento de la etapa de diseño, la representación debe ser capaz de dar cabida a eventuales futuros requerimientos. Un modelo de datos es por tanto una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas (entidades, propiedades y relaciones) y dinámicas (operaciones con las entidades, propiedades o relaciones) de una aplicación con un uso de datos intensivo. [44]

El modelo entidad-relación es el modelo conceptual más utilizado para el diseño conceptual de bases de datos. Originalmente, el modelo entidad-relación sólo incluía los conceptos de entidad,

relación y atributo. Más tarde, se añadieron otros conceptos, como los atributos compuestos y las jerarquías de generalización, en lo que se ha denominado *modelo entidad-relación extendido*. [43]

DER: Configurar Ficha de Investigación Epidemiológica

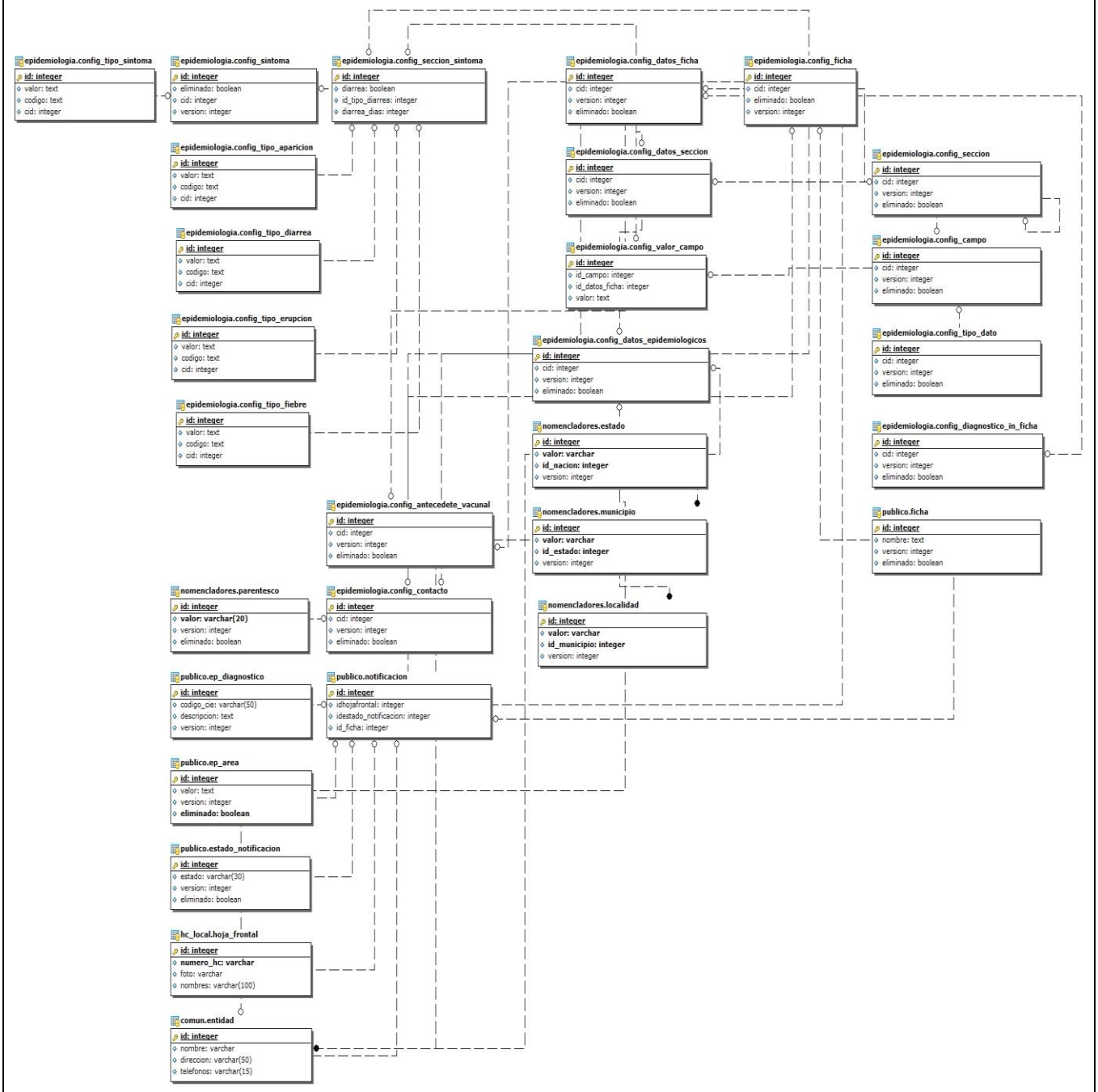

Figura 3.3.1 Diagrama Entidad – Relación – Configurar Ficha de Investigación Epidemiológica

3.4. Descripción de las tablas

Nombre: config_ficha		
Descripción: Tabla para registrar la configuración de la ficha		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador
cid	integer	Identificador de registro de la bitácora de sucesos
eliminado	boolean	Si fue eliminado del registro
versión	integer	Indica con que versión de la entidad se está trabajando
nombre	text	Nombre de la ficha
descripcion	text	Descripción de la ficha
signos_sintomas	boolean	Indica si esta sección está incluida en la ficha
datos_epidemiologicos	boolean	Indica si esta sección está incluida en la ficha
factores_riesgo	boolean	Indica si esta sección está incluida en la ficha
antecedentes_vacunales	boolean	Indica si esta sección está incluida en la ficha
diagnostico	boolean	Indica si esta sección está incluida en la ficha
datos_laboratorio	boolean	Indica si esta sección está incluida en la ficha
contactos	boolean	Indica si esta sección está incluida en la ficha
Id_notificacion	integer	Notificación asociada a la configuración

Tabla 3.4.1 Descripción de la tabla config_ficha.

Nombre: config_datos_ficha		
Descripción: Tabla para registrar los datos pertenecientes a la configuración de la ficha		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador
cid	integer	Identificador de registro de la bitácora de sucesos
eliminado	boolean	Si fue eliminado del registro
version	integer	Indica con que versión de la entidad se está trabajando
id_config_ficha	integer	Configuración asociada a los datos de la ficha
id_notificacion	integer	Notificación asociada a los datos
fecha	date	Fecha de creación de la Ficha

numero_ficha	integer	Número de la Ficha
semana_epidemiologica	integer	Semana epidemiológica de la Ficha
distrito_sanitario	text	Distrito Sanitario de la Ficha
terminacion	integer	Porciento de terminación
laboratorio_externo	boolean	Indica se de le hicieron pruebas de laboratorio a la Ficha

Tabla 3.4.2 Descripción de la tabla config_datos_ficha.

Nombre: config_seccion		
Descripción: Tabla para registrar la configuración de las secciones de la ficha		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador
cid	integer	Identificador de registro de la bitácora de sucesos
eliminado	boolean	Si fue eliminado del registro
versión	integer	Indica con que versión de la entidad se está trabajando
nombre	text	Nombre de la Sección
id_ficha	integer	Ficha asociada a esa Sección

Tabla 3.4.3 Descripción de la tabla config_seccion.

Nombre: config_campo		
Descripción: Tabla para registrar la configuración de los campos de las secciones		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador
cid	integer	Identificador de registro de la bitácora de sucesos
eliminado	boolean	Si fue eliminado del registro
versión	integer	Indica con que versión de la entidad se está trabajando
nombre	text	Nombre del campo
id_tipo_dato	integer	Tipo de dado asociado a ese campo
requerido	boolean	Si el campo es requerido o no
campo_padre	boolean	Si es campo padre o no
lista_hijos	text	Lista de los id de los campos hijos

id_generado	text	Id autogenerado del campo
id_seccion	integer	Sección asociada a ese campo

Tabla 3.4.4 Descripción de la tabla config_campo.

Nombre: config_tipo_dato		
Descripción: Agrupación de tipos de datos para los campos que se crean en cada una de las secciones.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador
cid	integer	Identificador de registro de la bitácora de sucesos
eliminado	boolean	Si fue eliminado del registro
versión	integer	Indica con que versión de la entidad se está trabajando
valor	text	Valor del tipo de dato

Tabla 3.4.5 Descripción de la tabla config_tipo_dato.

3.5. Breve valoración de las técnicas de validación (Integridad y Normalización de la Base de datos)

Validar los datos es el proceso de comprobar los valores que se escriben en los datos de la aplicación. Comprobar estos valores antes de enviar actualizaciones al almacén de datos subyacente es una buena práctica que reduce la posible cantidad de acciones de ida y vuelta entre una aplicación y el almacén de datos. [45]

Para confirmar que son válidos los datos que se escriben, se puede construir comprobaciones de validación en el propio conjunto de datos. El conjunto de datos puede comprobar los datos independientemente de cómo se esté realizando la actualización, ya sea directamente mediante los controles de un formulario, desde dentro de un componente o de alguna otra manera. Dado que el conjunto de datos forma parte de la aplicación, es lógico construir una validación específica de la aplicación (a diferencia de integrar las mismas comprobaciones en el servidor de bases de datos). [46]

Otras de las técnicas de validación muy utilizadas en la programacion es el tratamiento de errores, ya sean causados por fallas o limitaciones del hardware (errores de lectura de archivos por ejemplo) y/o por el software (casos en los cuales no se cumple una determinada condición). En los lenguajes convencionales la única forma de tratar estas situaciones era a través del uso de

funciones especiales que retornaban valores de control que indicaban si ocurrió un error y de qué tipo, el problema con esta forma de tratar los errores era que el código a escribir era cada vez más grande mientras más posibilidades de error existían.

Para facilitar el tratamiento de errores en Java se utilizó el concepto de Excepciones, que se refiere a una situación de error en la ejecución de un programa. Cada vez que ocurre una excepción (un error) el programa debe de tratarla, normalmente se muestra un mensaje de error y se ejecuta alguna rutina de tratamiento de errores. Para facilitar este control se han definido las palabras reservadas ***catch*** y ***try***, además también se ha definido una jerarquía de clases que se usan para crear objetos que mantienen información sobre las excepciones que han ocurrido. [47]

3.6. Vista de Implementación

La vista de implementación muestra el empaquetado físico de las partes reutilizables del sistema en unidades sustituibles, llamadas componentes. Una vista de implementación muestra los elementos físicos del sistema mediante componentes, así como sus interfaces y dependencias entre componentes. Los componentes son piezas reutilizables de alto nivel a partir de las cuales se pueden construir los sistemas.

El diagrama de componentes describe la descomposición física del sistema en componentes, a efectos de construcción y funcionamiento. La descomposición del diagrama de componentes se realiza en términos de componentes y de relaciones entre los mismos. Los componentes identifican objetos físicos que hay en tiempos de ejecución, de compilación o de desarrollo y tienen identidad propia con una interfaz bien definida. Los componentes incluyen código en cualquiera de sus formatos, DLL, Active X, bases de datos,... Cada componente incorpora la implementación de ciertas clases del diseño del sistema.

En un diagrama de componentes se muestran las diferentes relaciones de dependencia que se pueden establecer entre componentes. Los componentes bien diseñados no dependen de otros componentes sino de las interfaces que ofrecen los componentes. En ese caso, un componente en un sistema puede ser sustituido por otro componente que ofrezca las interfaces apropiadas. [49]

Seguidamente se muestra el Diagrama de Componente perteneciente al diseño dinámico de la Ficha de Investigación Epidemiológica.

Diagrama de componentes

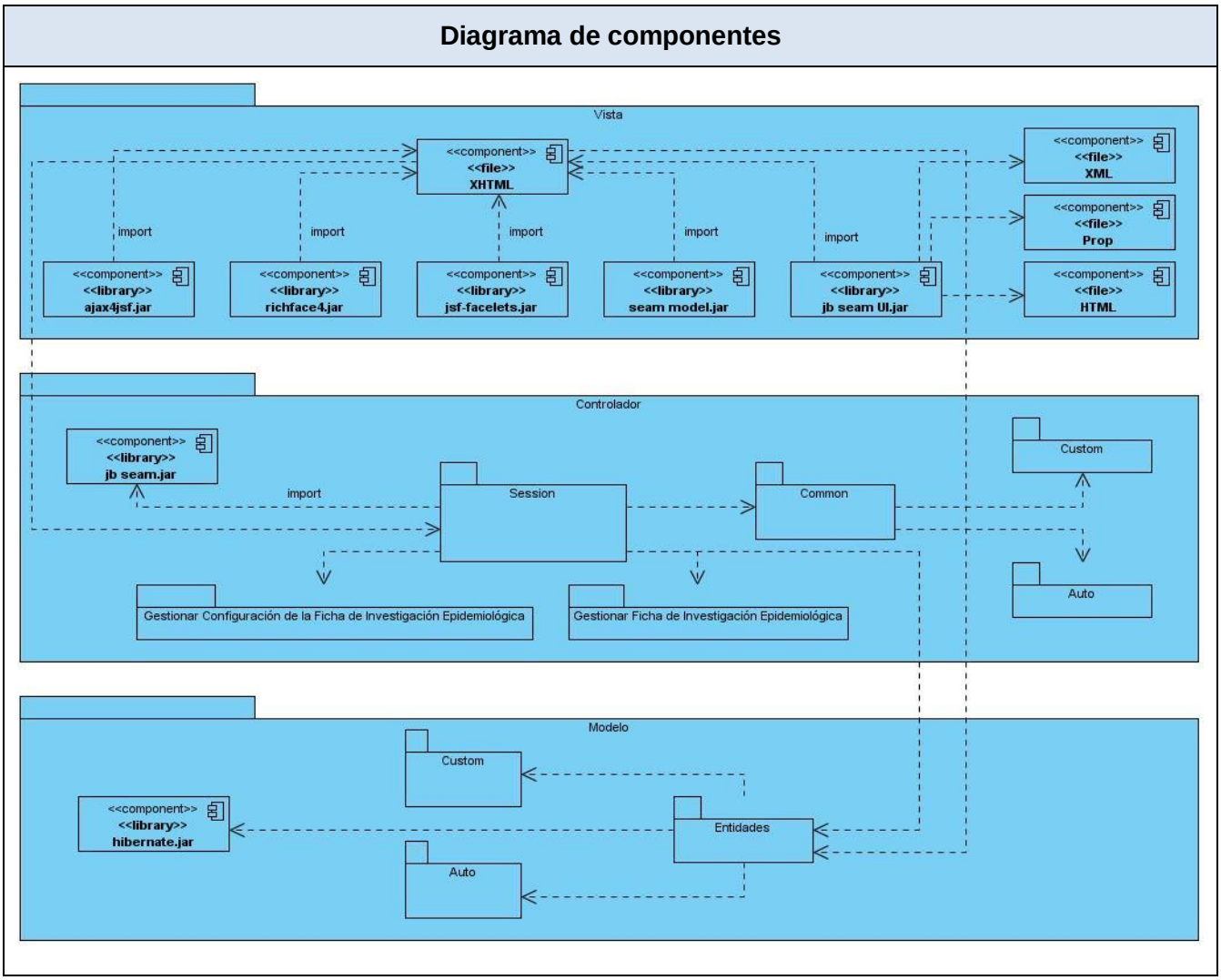

Figura 3.6.1 Diagrama de componentes

En el presente capítulo, se describió el modelo de datos del sistema con vista a obtener una representación de la aplicación a desarrollar y basado en el mismo comenzar con el flujo de trabajo de implementación. Se abordó el diseño propuesto con un análisis de los procedimientos y funcionalidades concebidas para la solución del sistema. Además de la elaboración del diagrama de componentes ayudando de esta forma a entender mejor las relaciones de dependencia entre ellos.

Capítulo 4: Modelo de Pruebas

En el presente capítulo, para concluir con el desarrollo del trabajo, se realizará una breve descripción del método escogido para modelar los casos de pruebas. Se utilizará el método de caja negra con el objetivo de probar las funcionalidades del sistema propuesto. Se expondrán los diferentes casos de pruebas realizados a los casos de uso descritos anteriormente en el flujo de Análisis y Diseño.

Las pruebas de software se pueden dividir en dos tipos, las pruebas de caja blanca o de caja negra. Las pruebas de caja blanca verifican que las operaciones internas se ajustan a lo especificado y que los componentes internos andan bien, por su parte, las pruebas de caja negra, conociendo la función del programa, intentan demostrar que las funciones están correctas.

4.1 Pruebas de caja negra

Un buen diseño de prueba, sistemáticamente saca a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. Para garantizar la efectividad de las pruebas, deben ser realizadas por un equipo independiente al que realizó el sistema. [50]

Las pruebas de **caja negra** se llevan a cabo sobre la interfaz del software y son completamente indiferentes al comportamiento interno y la estructura del programa. Con estas se pretende demostrar que las funcionalidades del sistema son operativas, que la entrada se acepta de forma adecuada, que se produce un resultado correcto y que se mantiene la integridad de la información externa.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están: [51]

- ✓ Técnica de la partición de equivalencia.
- ✓ Técnica del análisis de valores límites.
- ✓ Técnica de grafos de causa-efecto.

4.2 Descripción de los casos de pruebas

4.2.1 Caso de uso: Crear configuración de la Ficha de Investigación Epidemiológica

Escenarios del Crear configuración	Descripción de la funcionalidad	Flujo Central
EC 1: Crear configuración de la ficha de investigación epidemiológica	Crear configuración de la ficha de investigación epidemiológica satisfactoriamente.	Se selecciona la opción Crear configuración de la ficha de investigación epidemiológica. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Aceptar. Se crea la configuración de la ficha de investigación epidemiológica. Muestra la interfaz Ver detalles configuración de la ficha de investigación epidemiológica. Ver DCP: **Ver detalles de la configuración de la Ficha de Investigación Epidemiológica.**
EC 2: Cancelar operación	Cancelar la opción de Crear configuración de la ficha de investigación epidemiológica.	Se selecciona la opción Crear configuración de la ficha de investigación epidemiológica. Se introducen o seleccionan los datos correspondientes. Se selecciona la opción Cancelar. Se regresa a la vista anterior.
EC 3: Existen datos incompletos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber incompletos muestra un indicador sobre los campos incompletos.	Se selecciona la opción Crear configuración de la ficha de investigación epidemiológica. Se introducen los datos incompletos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incompletos.
EC 4: Existen datos incorrectos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber incorrectos muestra un indicador sobre los campos incorrectos.	Se selecciona la opción Crear configuración de la ficha de investigación epidemiológica. Se introducen los datos incorrectos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incorrectos.

SC 1: Crear configuración de la Ficha de Investigación Epidemiológica

Id del escenario	EC 1	EC 2	EC 3	EC 4
Escenario	Crear configuración de la ficha de investigación epidemiológica	Cancelar operación	Existen datos incompletos	Existen datos incorrectos
Variable 1 (Nombre)	V		I	I
Variable 2 (Descripción)	V		I	I
Botón 1 (Aceptar)	NA		NA	NA
Botón 2 (Cancelar)		NA		
Respuesta del Sistema	Crea la configuración de la ficha de investigación epidemiológica y se	Regresa a la vista anterior.	Muestra un indicador sobre los campos incompletos.	Muestra un indicador sobre los campos incorrectos.

	visualiza la interfaz Ver detalles.			
Resultado de la Prueba				

4.2.2 Caso de uso: Ver detalles de configuración de la Ficha de Investigación Epidemiológica.

Escenarios del Ver detalles de la configuración.	Descripción de la funcionalidad	Flujo Central
EC 1: Ver detalles de la Configuración de la Ficha de Investigación Epidemiológica	Ver detalles de la Configuración de la Ficha de Investigación Epidemiológica satisfactoriamente.	Se muestra la interfaz Ver detalles de Configuración de la Ficha de Investigación Epidemiológica. Se muestran los datos de la Configuración de la Ficha de Investigación Epidemiológica. Selecciona la opción Salir. Regresa a la vista anterior.
EC 2: Modificar Configuración de la Ficha de Investigación Epidemiológica	Permite acceder al caso de uso modificar Configuración de la Ficha de Investigación Epidemiológica.	Se muestra la interfaz Ver detalles de Configuración de la Ficha de Investigación Epidemiológica. Se muestran los datos de la Configuración de la Ficha de Investigación Epidemiológica. Se selecciona la opción Modificar. Ver DCP **Modificar Configuración de la Ficha de Investigación Epidemiológica.**
EC 3: Eliminar Configuración de la Ficha de Investigación Epidemiológica	Permite acceder al caso de uso eliminar Configuración de la Ficha de Investigación Epidemiológica.	Se muestra la interfaz Ver detalles de Configuración de la Ficha de Investigación Epidemiológica. Se muestran los datos de la Configuración de la Ficha de Investigación Epidemiológica. Se selecciona la opción Eliminar. Ver DCP **Eliminar Configuración de la Ficha de Investigación Epidemiológica.**

SC 1: Ver detalles de la configuración de la Ficha de Investigación Epidemiológica

Id del escenario	EC 1	EC 2	EC 3
Escenario	Ver detalles de Configuración de la Ficha de Investigación Epidemiológicas satisfactoriamente.	Modificar Configuración de la Ficha de Investigación Epidemiológica.	Eliminar Configuración de la Ficha de Investigación Epidemiológica.
Botón 1 (Salir)	NA		
Botón 2 (Modificar)		NA	

Botón 3 (Eliminar)			NA
Respuesta del Sistema	Regresa a la vista anterior.	Muestra la interfaz para modificar.	Muestra la interfaz para eliminar.
Resultado de la Prueba			

4.2.3 Caso de uso: Modificar configuración de la Ficha de Investigación Epidemiológica

Escenarios del modificar configuración	Descripción de la funcionalidad	Flujo Central
EC 1: Modificar configuración de la ficha de investigación epidemiológica	Modificar una configuración de ficha de investigación epidemiológica satisfactoriamente.	Muestra la interfaz para modificar configuración de la ficha de investigación epidemiológica. Se modifican los datos correspondientes. Se selecciona la opción Aceptar. Se modifica la configuración de la ficha de investigación epidemiológica. Muestra la interfaz Ver detalles de configuración de la ficha de investigación epidemiológica. Ver DCP **Ver detalles de configuración de la ficha de investigación epidemiológica.**
EC 2: Cancelar operación	Cancelar la opción de Modificar configuración de la ficha de investigación epidemiológica.	Muestra la interfaz para modificar configuración de la ficha de investigación epidemiológica. Se introducen los datos correspondientes. Se selecciona la opción Cancelar. Se regresa a la vista anterior.
EC 3: Existen datos incompletos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber incompletos muestra un indicador sobre los campos incompletos.	Muestra la interfaz para modificar configuración de la ficha de investigación epidemiológica. Se introducen los datos incompletos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incompletos.
EC 4: Existen datos incorrectos	Luego de haber introducido los datos, el sistema los verifica y valida, de haber incorrectos muestra un indicador sobre los campos incorrectos.	Muestra la interfaz para modificar configuración de la ficha de investigación epidemiológica. Se introducen los datos incorrectos. Se selecciona la opción Aceptar. Muestra un indicador sobre los campos incorrectos.

SC 1: Modificar configuración de la ficha de investigación epidemiológica

Id del escenario	EC 1	EC 2	EC 3	EC 4
Escenario	Modificar configuración de la ficha de investigación epidemiológica satisfactoriamente.	Cancelar operación	Existen datos incompletos	Existen datos incorrectos

Variable 1 (Nombre)	V		I	I
Variable 2 (Descripción)	V		I	I
(Aceptar)	NA		NA	NA
(Cancelar)		NA		
Respuesta del Sistema	Modifica la configuración de la ficha de investigación epidemiológica y se visualiza la interfaz ver detalles.	Regresa a la vista anterior.	Muestra un indicador sobre los campos incompletos.	Muestra un indicador sobre los campos incorrectos.
Resultado de la Prueba				

4.2.4 Caso de uso: Eliminar configuración de la Ficha de Investigación Epidemiológica

Escenarios del eliminar configuración	Descripción de la funcionalidad	Flujo Central
EC 1: Eliminar configuración de la Ficha de Investigación Epidemiológica	Eliminar una configuración de una Ficha de Investigación Epidemiológica satisfactoriamente.	Se muestra el mensaje: "Se eliminará la configuración seleccionada. Al seleccionar Sí se perderán todos los datos. ¿Desea continuar?". Selecciona la opción Si. Elimina la configuración de la Ficha de Investigación Epidemiológica
EC 2: Cancelar operación	Cancelar la opción de eliminar configuración de la Ficha de Investigación Epidemiológica.	Se muestra el mensaje: "Se eliminará la configuración seleccionada. Al seleccionar Sí se perderán todos los datos. ¿Desea continuar?". Se selecciona la opción de No. Se regresa a la vista anterior.

SC 1: Eliminar configuración de la Ficha de Investigación Epidemiológica

Id del escenario	EC 1	EC 2
Escenario	Eliminar configuración de la Ficha de Investigación Epidemiológica satisfactoriamente.	Cancelar operación .
Botón 4 (Si)	NA	
Botón 5 (No)		NA

Respuesta del Sistema	Elimina la configuración de la Ficha de Investigación Epidemiológica.	Regresa a la vista anterior.
Resultado de la Prueba		

4.2.5 Caso de uso: Buscar configuración de la Ficha de Investigación Epidemiológica

Escenarios del buscar configuración	Descripción de la funcionalidad	Flujo Central
EC 1: Buscar Configuración de la Ficha de Investigación Epidemiológica.	Buscar una Configuración de una Ficha de Investigación Epidemiológica dado criterios.	Muestra la interfaz para buscar la configuración de la Ficha de Investigación Epidemiológica. Se introducen los datos correspondientes a la búsqueda simple. Se selecciona la opción Buscar. Muestra un listado de las configuraciones de las Fichas de Investigación Epidemiológicas que cumplen con los criterios de búsqueda.
EC 2: Búsqueda avanzada.	Permite realizar la búsqueda por otros criterios.	Muestra la interfaz para buscar la Configuración de la Ficha de Investigación Epidemiológica. Se selecciona la opción búsqueda avanzada. Se introducen los datos correspondientes a la búsqueda avanzada. Se selecciona la opción Buscar. Muestra un listado de configuraciones de las Fichas de Investigación Epidemiológicas que cumplen con los criterios de búsqueda.
EC 3: No se encuentra información.	No se encuentra información que cumpla con los criterios de búsqueda.	Muestra la interfaz para buscar la Configuración de la Ficha de Investigación Epidemiológica. Se introducen los datos correspondientes. Se selecciona la opción Buscar. Muestra el mensaje de información “No se encontró información que cumpla con los criterios de búsqueda.”.
EC 4: Cancelar operación	Cancelar la opción de Buscar Configuración de la Ficha de Investigación Epidemiológica.	Muestra la interfaz para buscar la Configuración de la Ficha de Investigación Epidemiológica. Se introducen los datos correspondientes. Se selecciona la opción Cancelar. Se regresa a la vista anterior.

EC 5: Ver datos de la Configuración de la Ficha de Investigación Epidemiológica	Ver los datos de una Configuración de la Ficha de Investigación Epidemiológica.	Muestra la interfaz para buscar la Configuración de la Ficha de Investigación Epidemiológica. Se introducen los datos correspondientes. Se selecciona la opción Buscar. Muestra un listado de las configuraciones de las Fichas de Investigación Epidemiológicas que cumplen con los criterios de búsqueda. Se selecciona la opción Ver. Ver DCP **Ver datos de Configuración de la Ficha de Investigación Epidemiológica.**
EC 6: Modificar Configuración de la Ficha de Investigación Epidemiológica.	Permite acceder al caso de uso modificar Configuración de la Ficha de Investigación Epidemiológica.	Muestra la interfaz para buscar la Configuración de la Ficha de Investigación Epidemiológica. Se introducen los datos correspondientes. Se selecciona la opción Buscar. Muestra un listado de las configuraciones de las Fichas de Investigación Epidemiológicas que cumplen con los criterios de búsqueda. Se selecciona la opción Modificar. Ver DCP **Modificar Configuración de la Ficha de Investigación Epidemiológica.**
EC 7: Eliminar Configuración de la Ficha de Investigación Epidemiológica.	Permite acceder al caso de uso eliminar Configuración de la Ficha de Investigación Epidemiológica.	Muestra la interfaz para buscar la Configuración de la Ficha de Investigación Epidemiológica. Se introducen los datos correspondientes. Se selecciona la opción Buscar. Muestra un listado de las configuraciones de las Fichas de Investigación Epidemiológicas que cumplen con los criterios de búsqueda. Se selecciona la opción Eliminar. Ver DCP **Eliminar Configuración de la Ficha de Investigación Epidemiológica.**
EC 8: Ordenar.	Permite ordenar el resultado ascendente o descendentemente por un atributo.	Muestra la interfaz para buscar la Configuración de la Ficha de Investigación Epidemiológica. Se introducen los datos correspondientes. Se selecciona la opción Buscar. Muestra un listado de las configuraciones de las Fichas de Investigación Epidemiológicas que cumplen con los criterios de búsqueda. Se ordenan los campos del listado por atributos.

SC 1: Buscar configuración de la Ficha de Investigación Epidemiológica

Id del escenario	EC 1	EC 2	EC 3	EC 4	EC 5	EC 6	EC 7
Escenario	Buscar Configuración de la Ficha de Investigación Epidemiológica.	No se encuentra información	Cancelar operación	Ver datos de la Configuración de la Ficha de Investigación Epidemiológica.	Modificar Configuración de la Ficha de Investigación Epidemiológica.	Eliminar Configuración de la Ficha de Investigación Epidemiológica.	Ordenar
Variable 1 (Nombre)	V	I					
Botón 1 (Orden ascendente)							NA
Botón 2 (Orden descendente)							NA
Botón 3 (Buscar)	NA	NA					
Botón 4 (Cancelar)			NA				
Botón 5 (Ver)				NA			
Botón 6 (Modificar)					NA		
Botón 7 (Eliminar)						NA	
Respuesta del Sistema	Muestra: • Nombre de la configuración	Muestra el mensaje de información: "No se encuentra información que cumpla con los criterios de búsqueda.".	Regresa a la vista anterior.	Muestra la interfaz para ver datos.	Muestra la interfaz para modificar.	Muestra la interfaz para eliminar.	Muestran los atributos del listado ordenados.
Resultado de la Prueba							

4.2.6 Caso de Uso: Ver datos de la configuración de la Ficha de Investigación Epidemiológica

Escenarios del ver datos de la configuración	Descripción de la funcionalidad	Flujo Central
EC 1: Ver datos de configuración de la ficha de investigación epidemiológica.	Ver datos de una configuración de una ficha de investigación epidemiológica satisfactoriamente.	Se muestra la interfaz Ver datos de la configuración de la ficha de investigación epidemiológica. Se muestran los datos de la configuración de la ficha de investigación epidemiológica. Selecciona la opción Salir. Regresa a la vista anterior.

SC 1: Ver datos de la configuración de la ficha de investigación epidemiológica

Id del escenario	EC 1
Escenario	Ver datos de la configuración de la ficha de investigación epidemiológica.
Botón 6 *(Salir)*	NA
Respuesta del Sistema	Regresa a la vista anterior.
Resultado de la Prueba	

Conclusiones

Con el desarrollo de la investigación para dar solución al problema de la generación dinámica de Fichas de Investigación Epidemiológicas se arribó a las siguientes conclusiones:

1. Se obtuvo un Sistema que permite la generación dinámica de Fichas de Investigación Epidemiológicas atendiendo a las características particulares de cada ficha a crear.
2. El estudio del estado del arte realizado evidenció el predominio de aplicaciones privadas y carentes de funcionalidades para la creación de fichas de investigación epidemiológicas, lo que limita su adquisición y usabilidad para la investigación.
3. El proceso de diseño de una Ficha de Investigación Epidemiológica necesita de calidad y de la selección de los componentes adecuados para su correcto uso.
4. El diseño dinámico de una Ficha de Investigación Epidemiológica permitió que el epidemiólogo tenga un mejor control del estudio e investigación de la enfermedad.
5. El patrón de arquitectura Modelo Vista Controlador, facilita una mejor organización de las funcionalidades, disminuyendo así la complejidad en la implementación de los casos de uso.
6. La utilización de las pautas definidas garantizan la homogeneidad de todas las interfaces del sistema.

Recomendaciones

Se recomienda:

- Incluir una funcionalidad que permita duplicar una configuración de Ficha de Investigación Epidemiológica. De forma tal que se puedan modificar elementos de la misma a fin de utilizarla en otra investigación.
- Modificar la sección de Datos de laboratorio de modo tal que permita visualmente escoger si desea tener Prueba, Examen o Muestra en dependencia de la ficha que se esté mostrando.
- De forma general, especificar con un mayor grado de detalle los formatos de las secciones predeterminadas de manera que queden correctamente descritas para su tratamiento.
- Continuar con el desarrollo de las funcionalidades correspondientes a la generación dinámica de componentes. Incluir el componente Hora y lograr el funcionamiento de campos padres.

Referencias Bibliográficas

[1] Castells, Pablo y Macías José A. Un sistema de presentación dinámica en entornos web para representaciones personalizadas del conocimiento. Madrid, España, 2002. 10p. [Citado el: 4 de Marzo de 2010] *Disponible en:* http://arantxa.ii.uam.es/~castells/publications/aepia02.pdf

[2] Corpobox. CORPOBOX. 2007. [Citado el: 4 de Marzo de 2010]
Disponible en: http://corpobox.com/SalesBox.aspx

[3] Informed. Informática Médica. [Citado el: 4 de Marzo de 2010]
Disponible en: http://www.terra.es/personal/cseron/tesis/informed.htm

[4] Morejón Giraldoni, F. Alain; Hernández Barrio, Esteban; Rodríguez Izaguirre, Teresita del Carmen; Moreno Torres, Jirlén; Seife Echevarría, Aimee. MediSur. 2007 [Citado el: 4 Octubre 2010]
Disponible en: http://www.medisur.sld.cu/index.php/medisur/rt/printerFriendly/356/368

[5] Fernández Puerto, Francisco. Sistema de Información Hospitalaria. México, Distrito Federal. 2003. 24p. [Citado: 12 Enero de 2011].
Disponible en: http://www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf

[6] Ortiz, Zulma; Esandi, María Eugenia; Bortman, Marcelo. Epidemiología Básica y Vigilancia de la Salud. 2004. [Citado: 23 de Enero de 2011].
Disponible en: http://es.scribd.com/doc/7036726/Epidemiologia-1

[7] García Nogueira, Keila. Procesos Básicos Epidemiológicos para un Sistema de Gestión Hospitalario. Universidad de las Ciencias Informáticas. Ciudad de la Habana. 2009. 10p.
[Citado el: 1 Octubre 2010]

[8] MedlinePlus. ADAM. 26 octubre 2009 [Citado el: 7 Octubre 2010]
Disponible en: http://www.nlm.nih.gov/medlineplus/spanish/ency/article/001929.htm

[9] Sigma. Vigilancia de eventos de notificación obligatoria. 2005. [Citado el: 15 Octubre 2010]
Disponible en: http://www.sigma.org.ar/eno/index.php

[10] Traverso Arguedas, Ciro. Epidemiología básica, 29 enero 2007. [Citado el: 15 Octubre 2010]
Disponible en: http://www.mailxmail.com/curso-epidemiologia-basica/sistema-vigilancia-epidemiologica

[11] Curso de Gerencia para el Manejo Efectivo del Programa Ampliado de Inmunización (PAI).2005 [Citado el: 23 Octubre 2010]
Disponible en: http://www.paho.org/english/ad/fch/im/isis/epi_mod/spanish/4/vigilancia_si.asp

[12] Castells, Pablo y Macías José A. Un sistema de presentación dinámica en entornos web para representaciones personalizadas del conocimiento. Madrid, España, 2002. 10p.
[Citado: 24 Octubre 2010] *Disponible en:* http://arantxa.ii.uam.es/~castells/publications/aepia02.pdf

[13] Zarza, Gonzalo y Minni, Hugo. Generación dinámica de interfaces de usuario a partir de modelos representados mediante esquemas XML. Universidad Nacional de Litoral. Argentina.4p [Citado: 23 Octubre 2010] *Disponible en:* http://www.ing.unp.edu.ar/wicc2007/trabajos/TIAE/143.pdf

[14] *Barandiarán Gordo, Laura; Martín de Andrés, Diego. Generación automática de interfaces gráficas de usuario para una herramienta CASE de toma de requisitos. Universidad Carlos III. Madrid.2009.122p.* [Citado: 01 Noviembre 2010]

Disponible en: http://e-archivo.uc3m.es/handle/10016/6133

[15] Napanga Saldaña, O. Edwin y Fabián Manzano, Judith. Ministerio de Salud, Oficina General de Epidemiología. Lima Perú. 2003. [Citado el: 05 Noviembre 2010]

Disponible en: http://saludarequipa.gob.pe/epidemiologia/ASIS/docs/Enlaces/020.pdf

[16] Centers for Disease Control and Prevention (CDC). ¿What Is Epi Info™? Agosto 2008. [Citado el: 04 Noviembre 2010] *Disponible en:* http://www.cdc.gov/epiinfo/

[17] *Valerga, Dr. Héctor. Boletín Epidemiológico Nº 6. Junio 2005.* [Citado el: 13 Noviembre 2010]

Disponible en: http://www.sigma.org.ar/BoletinEpidemiologico6.pdf

[18] Rico Cordeiro, Dr. Osvaldo. VI Congreso Internacional de Informática en Salud. 2007. [Citado el: 13 Noviembre 2010]

Disponible en: http://www.sld.cu/galerias/pdf/sitios/vigilancia/vigilanciaargentina.pdf

[19] Rangel Marrero, Rudny y Rojas González, Arletis. Diseño e implementación de los procesos de Apoyo a la vigilancia epidemiológica del Sistema de Información Hospitalaria alas HIS. Universidad de las Ciencias Informáticas. Ciudad de la Habana. 2010. 79p. [Citado el: 18 Noviembre 2010]

[20] Ricardo Collada de la Rosa, Yunaysy Ortíz Batista, Runer Céspedes Aldana, Malena Guzmán Pérez, Daybert Hernández Hernández. Módulo de Higiene y Epidemiología del Sistema Informático de Control Sanitario Internacional. Universidad de las Ciencias Informáticas. Ciudad de la Habana. 2009. 6p. [Citado el: 18 Noviembre 2010]

[21] Scott, Storkel. O'Reilly on Java.com. Noviembre 2002. [Citado el: 18 Noviembre 2010]

Disponible en: http://onjava.com/pub/a/onjava/2002/12/11/eclipse.html

[22] SeamFramework.org. Next Generation Enterprise Java Development. 2009. [Citado el: 25 Noviembre 2010] Disponible en: http://seamframework.org/Seam2

[23] ALBET, SA. Documento de arquitectura del Sistema de Información Hospitalaria alas HIS. ALBET, Ingeniería y Sistemas, 2008. [Citado el: 25 Noviembre 2010]

[24] Jamae David y Johnson, Peter**.** *JBoss in Action.* 2009. [Citado el: 24 Febrero 2011]

[25] JBoss Community. JBoss.org. [Citado: 23 de Febrero de 2011]

Disponible en: http://docs.jboss.com/jbportal/v2.7.1/userGuide/html/features.html

[26] JBoos Tool, Gobierno de Cantabria, 10 Octubre de 2008. [Citado: 27 Noviembre de 2010]

Disponible en: http://amap.cantabria.es/confluence/display/DEV/JBoss+Tools

[27] Sánchez Suárez, José Manuel. Introducción a RichFaces. 2 de enero de 2010.
[Citado: 03 de Diciembre de 2010]
Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro
[28] Ginestà, Marc Gibert y Pérez Mora, Oscar. Base de Datos en PostgreSQL. UOC. 78p. [Citado el: 24 de Febrero de 2011]
Disponible en: http://www.sewebs.com.ar/cosas/biblioteca/06_bases_de_datos_en_postgresql.pdf
[29] Galvão de Oliveira Aline; Barbosa Chagas, Boni; Kaminski Aires, Simone. Alternativas para el desarrollo de Software. ADMpg. v. 2, p.119-123, 2009. [Citado el: 24 de Febrero de 2011]
Disponible en: http://www.admpg.com.br/revista2009/v2/artigos/a14.pdf
[30] Red Hat Middleware Relational Persistence for Java and .NET.2009 [Citado: 23 de Febrero de 2011] *Disponible en:* https://hibernate.bluemars.net/
[31] Bauer, Cristian y King, Gavin. Java Persistence with Hibernate. 2005. [Citado el: 23 de Febrero de 2011]
[32] Aprendiendo Java. Que es JPA. 25 Junio de 2010. [Citado el: 5 de Mayo de 2011]
Disponible en: http://www.aprendiendojava.com.ar/index.php?topic=54.0
[33] Visual Paradigm for UML. 5 de marzo del 2007. [Citado el: 07 de Diciembre de 2010]
Disponible en:
http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_14720_p/
[34] Arnold, Ken; Gosling, James; Holmes, David. El lenguaje de Programación Java. Addison Wesley. 2009. [Citado el: 14 de Enero de 2011]
Disponible en: http://www.lenguajes-de-programacion.com/programacion-java.shtml
[35] Visware. Desarrollo de Software. 2005. [Citado el: 28 de Febrero de 2011]
Disponible en: http://www.visware.com.ar/vsi/site/secciones/glosario/
[36] Sánchez, Edgar. MVC - Modelo Vista Controlador. Enero 2009. [Citado el: 05 Enero de 2011]
Disponible en: http://ecuador.latindevelopers.net/blogs/edgarsanchez/archive/2009/01/28/el-patr-243-n-mvc-para-asp-net-ya-est-225-en-release-candidate.aspx
[37] Garcerant, Iván. Tecnología y Synergix. Julio 2008. [Citado: 16 de Enero de 2011]
Disponible en: http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/
[38] Marrero López, Yadira; Rosales García, Adonis R; Mejias Cesar, Yuleidys. Propuesta del diseño arquitectónico de una plataforma para la predicción de actividad biológica de compuestos orgánicos. Universidad de las Ciencias Informáticas. Ciudad de la Habana. 2009. 5p.
[Citado: 17 de Enero de 2011] *Disponible en*: http://www.informatica2009.sld.cu/Members/ymlopez
[39] Calahorrano Narváez, Amy Indira.Herramienta para evaluar la calidad del código fuente generado en C ANSI. Escuela Politécnica Nacional. Quito. 2007. 164p. [Citado:6 de Marzo de 2011]
Disponible en: http://bibdigital.epn.edu.ec/bitstream/15000/559/1/CD-1068.pdf

[40] Sandoval Carlos, Oscar Andrés y Castillo Rojas, Wilson Andrés. Integración de patrones de diseño y aplicaciones móviles. 2002. [Citado: 6 de Marzo de 2011]
Disponible en: http://arxiv.org/ftp/arxiv/papers/1008/1008.2542.pdf
[41] Álvarez Sarmiento, Pablo Ricardo. Sistema basado en componentes para automatizar la herramienta Balanced Scoreca. Sangolquí. 2007. 296p. [Citado: 6 de Marzo de 2011]
Disponible en: http://repositorio.espe.edu.ec/bitstream/21000/582/1/T-ESPE-021814.pdf
[42] Torres Montano, Juan Carlos. Diagramas. 2009. *[Citado: 22 de Marzo de 2011]*
Disponible en: http://t1diagramas.blogspot.com/2009/08/diagrama-de-paqutes.html
[43] García Chávez, Carlos Alberto. Diseño de bases de datos relacionales. 2005. *[Citado: 17 de Marzo de 2011]* Disponible en: http://www.mailxmail.com/curso-diseno-base-datos-relacionales/diseno-conceptual-bases-datos-modelo-entidad-relacion
[44] Pérez, Chantal. Bases de datos y bases de conocimiento. 2002. *[Citado: 17 de Marzo de 2011]*
Disponible en: http://ddd.uab.cat/pub/elies/elies_a2002v18/522.html
[45] MSDN. Validar datos mientras se modifica la fila. 2005. [*Citado: 22 de Marzo de 2011]*
Disponible en: http://msdn.microsoft.com/es-es/library/1120xds5%28v=vs.80%29.aspx
[46] MSDN. Información general sobre validación de datos. 2007. [*Citado: 22 de Marzo de 2011]*
Disponible en: http://msdn.microsoft.com/es-es/library/kx9x2fsb%28v=vs.90%29.aspx
[47] Instituto Nacional de Estadística e Informática INEI. Tratamiento de Errores: Excepciones.1997. [*Citado: 22 de Marzo 2011].*
Disponible en: http://www.inei.gob.pe/biblioineipub/bancopub/inf/lib5069/6-1.HTM
[48] Pérez García, Alejandro. Hibernate Validator, y como definir las validaciones sobre los objetos de negocio. 2008. [*Citado: 22 de Marzo de 2011]*
Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hibernateValidator
[49] Platero Dueñas, Carlos. UML el modelo dinámico y de Implementación. 2009. [Citado: 22 de Marzo de 2011]. Disponible en:
http://www.elai.upm.es:8009/spain/Asignaturas/InfoInd/apuntesAOOD/cap5UMLDinamicoImpl.pdf
[50] Juristo Natalia, Moreno Ana M., Vegas Sira. TÉCNICAS DE EVALUACIÓN DE SOFTWARE. [Citado: 4 de Mayo de 2011]
[51] Pressman, R. S. Ingeniería del software. Un enfoque práctico. 1997. [Citado: 4 de Mayo de 2011]

Bibliografía

1. Castells, Pablo y Macías José A. Un sistema de presentación dinámica en entornos web para representaciones personalizadas del conocimiento. Madrid, España, 2002. 10p.
Disponible en: http://arantxa.ii.uam.es/~castells/publications/aepia02.pdf
2. Corpobox. CORPOBOX. 2007. Disponible en: http://corpobox.com/SalesBox.aspx
3. Informed. Informática Médica.
Disponible en: http://www.terra.es/personal/cseron/tesis/informed.htm
4. Morejón Giraldoni, F. Alain; Hernández Barrio, Esteban; Rodríguez Izaguirre, Teresita del Carmen; Moreno Torres, Jirlén; Seife Echevarría, Aimee. MediSur. 2007
Disponible en: http://www.medisur.sld.cu/index.php/medisur/rt/printerFriendly/356/368
5. Fernández Puerto, Francisco. Sistema de Información Hospitalaria. México, Distrito Federal. 2003. 24p. Disponible en: http://www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf
6. Ortiz, Zulma; Esandi, María Eugenia; Bortman, Marcelo. Epidemiología Básica y Vigilancia de la Salud. 2004. Disponible en: http://es.scribd.com/doc/7036726/Epidemiologia-1
7. García Nogueira, Keila. Procesos Básicos Epidemiológicos para un Sistema de Gestión Hospitalario. Universidad de las Ciencias Informáticas. Ciudad de la Habana. 2009. 10p.
8. MedlinePlus. ADAM. 26 octubre 2009
Disponible en: http://www.nlm.nih.gov/medlineplus/spanish/ency/article/001929.htm
9. Sigma. Vigilancia de eventos de notificación obligatoria. 2005.
Disponible en: http://www.sigma.org.ar/eno/index.php
10. Traverso Arguedas, Ciro. Epidemiología básica, 29 enero 2007.
Disponible en: http://www.mailxmail.com/curso-epidemiologia-basica/sistema-vigilancia-epidemiologica
11. Curso de Gerencia para el Manejo Efectivo del Programa Ampliado de Inmunización (PAI).2005
Disponible en: http://www.paho.org/english/ad/fch/im/isis/epi_mod/spanish/4/vigilancia_si.asp
12. Castells, Pablo y Macías José A. Un sistema de presentación dinámica en entornos web para representaciones personalizadas del conocimiento. Madrid, España, 2002. 10p.
Disponible en: http://arantxa.ii.uam.es/~castells/publications/aepia02.pdf
13. Zarza, Gonzalo y Minni, Hugo. Generación dinámica de interfaces de usuario a partir de modelos representados mediante esquemas XML. Universidad Nacional de Litoral. Argentina.4p
Disponible en: http://www.ing.unp.edu.ar/wicc2007/trabajos/TIAE/143.pdf
14. *Barandiarán Gordo, Laura; Martín de Andrés, Diego. Generación automática de interfaces gráficas de usuario para una herramienta CASE de toma de requisitos. Universidad Carlos III. Madrid.2009.122p.*
Disponible en: http://e-archivo.uc3m.es/handle/10016/6133
15. Napanga Saldaña, O. Edwin y Fabián Manzano, Judith. Ministerio de Salud, Oficina General de Epidemiología. Lima Perú. 2003.
Disponible en: http://saludarequipa.gob.pe/epidemiologia/ASIS/docs/Enlaces/020.pdf
16. Centers for Disease Control and Prevention (CDC). ¿What Is Epi Info™? Agosto 2008.
Disponible en: http://www.cdc.gov/epiinfo/
17. *Valerga, Dr. Héctor. Boletín Epidemiológico Nº 6. Junio 2005.*
Disponible en: http://www.sigma.org.ar/BoletinEpidemiologico6.pdf
18. Rico Cordeiro, Dr. Osvaldo. VI Congreso Internacional de Informática en Salud. 2007.
Disponible en: http://www.sld.cu/galerias/pdf/sitios/vigilancia/vigilanciaargentina.pdf
19. Rangel Marrero, Rudny y Rojas González, Arletis. Diseño e implementación de los procesos de Apoyo a la vigilancia epidemiológica del Sistema de Información Hospitalaria alas HIS. Universidad de las Ciencias Informáticas. Ciudad de la Habana. 2010. 79p.
20. Ricardo Collada de la Rosa, Yunaysy Ortíz Batista, Runer Céspedes Aldana, Malena Guzmán Pérez, Daybert Hernández Hernández. Módulo de Higiene y Epidemiología del Sistema Informático

de Control Sanitario Internacional. Universidad de las Ciencias Informáticas. Ciudad de la Habana. 2009. 6p.
21. Scott, Storkel. O'Reilly on Java.com. Noviembre 2002.
Disponible en: http://onjava.com/pub/a/onjava/2002/12/11/eclipse.html
22. SeamFramework.org. Next Generation Enterprise Java Development. 2009.
Disponible en: http://seamframework.org/Seam2
23. ALBET, SA. Documento de arquitectura del Sistema de Información Hospitalaria alas HIS. ALBET, Ingeniería y Sistemas, 2008.
24. Jamae David y Johnson, Peter**.** *JBoss in Action.* 2009.
25. JBoss Community. JBoss.org.
Disponible en: http://docs.jboss.com/jbportal/v2.7.1/userGuide/html/features.html
26. JBoos Tool, Gobierno de Cantabria, 10 Octubre de 2008.
Disponible en: http://amap.cantabria.es/confluence/display/DEV/JBoss+Tools
27. Sánchez Suárez, José Manuel. Introducción a RichFaces. 2 de enero de 2010.
Disponible en: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro
28. Ginestà, Marc Gibert y Pérez Mora, Oscar. Base de Datos en PostgreSQL. UOC. 78p.
Disponible en: http://www.sewebs.com.ar/cosas/biblioteca/06_bases_de_datos_en_postgresql.pdf
29. Galvão de Oliveira Aline; Barbosa Chagas, Boni; Kaminski Aires, Simone. Alternativas para el desarrollo de Software. ADMpg. v. 2, p.119-123, 2009.
Disponible en: http://www.admpg.com.br/revista2009/v2/artigos/a14.pdf
30. Red Hat Middleware Relational Persistence for Java and .NET.2009
Disponible en: https://hibernate.bluemars.net/
31. Bauer, Cristian y King, Gavin. Java Persistence with Hibernate. 2005.
32. Aprendiendo Java. Que es JPA. 25 Junio de 2010.
Disponible en: http://www.aprendiendojava.com.ar/index.php?topic=54.0
33. Visual Paradigm for UML. 5 de marzo del 2007.
Disponible en:
http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_14720_p/
34. Arnold, Ken; Gosling, James; Holmes, David. El lenguaje de Programación Java. Addison Wesley. 2009. Disponible en: http://www.lenguajes-de-programacion.com/programacion-java.shtml
35. Visware. Desarrollo de Software. 2005.
Disponible en: http://www.visware.com.ar/vsi/site/secciones/glosario/
36. Sánchez, Edgar. MVC - Modelo Vista Controlador. Enero 2009.
Disponible en: http://ecuador.latindevelopers.net/blogs/edgarsanchez/archive/2009/01/28/el-patr-243-n-mvc-para-asp-net-ya-est-225-en-release-candidate.aspx
37. Garcerant, Iván. Tecnología y Synergix. Julio 2008.
Disponible en: http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/
38. Marrero López, Yadira; Rosales García, Adonis R; Mejias Cesar, Yuleidys. Propuesta del diseño arquitectónico de una plataforma para la predicción de actividad biológica de compuestos orgánicos. Universidad de las Ciencias Informáticas. Ciudad de la Habana. 2009. 5p.
Disponible en: http://www.informatica2009.sld.cu/Members/ymlopez
39. Calahorrano Narváez, Amy Indira.Herramienta para evaluar la calidad del código fuente generado en C ANSI. Escuela Politécnica Nacional. Quito. 2007. 164p.
Disponible en: http://bibdigital.epn.edu.ec/bitstream/15000/559/1/CD-1068.pdf
40. Sandoval Carlos, Oscar Andrés y Castillo Rojas, Wilson Andrés. Integración de patrones de diseño y aplicaciones móviles. 2002.
Disponible en: http://arxiv.org/ftp/arxiv/papers/1008/1008.2542.pdf
41. Álvarez Sarmiento, Pablo Ricardo. Sistema basado en componentes para automatizar la herramienta Balanced Scoreca. Sangolquí. 2007. 296p.
Disponible en: http://repositorio.espe.edu.ec/bitstream/21000/582/1/T-ESPE-021814.pdf
42. Torres Montano, Juan Carlos. Diagramas. 2009.
Disponible en: http://t1diagramas.blogspot.com/2009/08/diagrama-de-paqutes.html

43. García Chávez, Carlos Alberto. Diseño de bases de datos relacionales. 2005. Disponible en: http://www.mailxmail.com/curso-diseno-base-datos-relacionales/diseno-conceptual-bases-datos-modelo-entidad-relacion
44. Pérez, Chantal. Bases de datos y bases de conocimiento. 2002. *Disponible en: http*://ddd.uab.cat/pub/elies/elies_a2002v18/522.html
45. MSDN. Validar datos mientras se modifica la fila. 2005. *Disponible en:* http://msdn.microsoft.com/es-es/library/1120xds5%28v=vs.80%29.aspx
46. MSDN. Información general sobre validación de datos. 2007. *Disponible en:* http://msdn.microsoft.com/es-es/library/kx9x2fsb%28v=vs.90%29.aspx
47. Instituto Nacional de Estadística e Informática INEI. Tratamiento de Errores: Excepciones.1997. *Disponible en:* http://www.inei.gob.pe/biblioineipub/bancopub/inf/lib5069/6-1.HTM
48. Pérez García, Alejandro. Hibernate Validator, y como definir las validaciones sobre los objetos de negocio. 2008. *Disponible en:* http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hibernateValidator
49. Platero Dueñas, Carlos. UML el modelo dinámico y de Implementación. 2009. Disponible en: http://www.elai.upm.es:8009/spain/Asignaturas/InfoInd/apuntesAOOD/cap5UMLDinamicoImpl.pdf
50. Juristo Natalia, Moreno Ana M., Vegas Sira. TÉCNICAS DE EVALUACIÓN DE SOFTWARE.
51. Pressman, R. S. Ingeniería del software. Un enfoque práctico. 1997.
52. Arnold, Ken; Gosling, James; Holmes, David. El lenguaje de Programación Java. Addison Wesley. 2009. Disponible en: http://www.lenguajes-de-programacion.com/programacion-java.shtml
53. Allen, Dan**.** Seam in Action. MEAP. 2008. 556p
54. Morejón Giraldoni, F. Alain; Barrio, Esteban; Rodríguez Izaguirre, Teresita del Carmen. MediSur. 2007. Disponible en: http://www.medisur.sld.cu/index.php/medisur/rt/printerFriendly/356/368
55. Jamae, David y Johnson, Peter**.** JBoss in Action. 2009.
56. Hat, Red. RichFaces developer Guide. 2007. 156p
57. *UML y patrones Tomo I Prentice Hall Iberoamericana.* Larma, Craig**.** 1999.
58. *Marrero López, Yadira; Rosales García, Adonis R; Mejias Cesar, Yuleidys. Propuesta del diseño arquitectónico de una plataforma para la predicción de actividad biológica de compuestos orgánicos. Universidad de las Ciencias Informáticas. Ciudad de la Habana. 2009. 5p Disponible en:* http://www.informatica2009.sld.cu/Members/ymlop
59. Bauer, Cristian y King, Gavin. Java Persistence with Hibernate. 2005. 243
60. The PostgreSQL Global Development Group. PostgreSQL 8.1.0 Documentation. California: University of California, 2005.
61. Grau, Xavier Ferré. Tutorial UML, Desarrollo Orientado a Objetos con UML. 2004. Disponible en: http://www.clikear.com/manuales/uml/introduccion.aspx.
62. García Linares, Reche Martínez, D. Richarte Reina. *Un nuevo concepto en Sistemas de Información Hospitalarios.* Madrid : Comunicaciones S.A. 2003.
63. Patrones de Diseño en aplicaciones Web con Java J2EE. 2007. Disponible en:http://java.ciberaula.com/articulo/diseno_patrones_j2ee.
64. García de Jalón Javier, Rodriguez Iñigo Mingo José Ignacio, Alfonso Brazález Aitor Imaz, Larzabal Alberto, Calleja Jesús, García Jon. Aprenda Java como si estuviera en primero. Escuela Superior de Ingenieros Industriales. Universidad de Navarra. España. Enero 2000