Universidad de las Ciencias Informáticas

Facultad 6

Título: "Sistema de Gestión de la Información de Laboratorios de Calidad del Centro de Ingeniería Genética y Biotecnología: Desarrollo del módulo Estudios de Estabilidad y Materiales de Referencia".

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

Autores: Yaima Grinión Rodríguez

Daryanis Cedeño Barrero

Tutores: Ing. Adisley Reyes Crespo

Ing. Rosayda Valiente Mesa

Ciudad de La Habana, Junio 2009

"Año 50 de la Revolución"

"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."

Albert Einstein.

DECLARACIÓN DE AUTORÍA

Declaramos ser autoras de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de Junio del 2008.

_____ _____

Yaima Grinión Rodríguez Daryanis Cedeño Barrero

_____ _____

Ing. Adisley Reyes Crespo Ing. Rosayda Valiente Mesa

Datos de Contacto

Ing. Adisley Reyes Crespo
Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.
Email: areyesc@uci.cu

Ing. Rosayda Valiente Mesa
Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.
Email: rvaliente@uci.cu

Agradecimientos

Agradecemos a nuestros padres y seres más queridos que hicieron posible que se cumplieran nuestros sueños y nos apoyaron cada día en el transcurso de nuestras vidas.

A la Revolución y en especial a nuestro Comandante en Jefe por ser el guía eterno de nuestro camino y creador de la Universidad de las Ciencias Informáticas.

A nuestros amigos que nos brindaron consuelo cuando el camino se tornó más difícil.

A Manuel Alejandro Cora González por su colaboración y ayuda incondicional.

A nuestros compañeros de estudios con los que pasamos 5 años maravillosos de universidad.

A nuestras tutoras, por su apoyo, preocupación y aporte importante en este trabajo.

A los profesores de la Universidad de las Ciencias Informáticas, por la formación que nos han dado a lo largo de estos cinco años de la carrera.

A los especialistas del CIGB especialmente a Gerardo García Illera e Indira Pla del Grupo de Estudios de Estabilidad y Materiales de Referencia del Centro de Ingeniería Genética y Biotecnología, por su valiosa contribución.

A todos los que hemos conocido y en algún momento nos brindaron su apoyo desinteresado…

Muchas gracias, los recordaremos siempre y le estaremos eternamente agradecidas…

Dedicatoria

Yaima Grinión Rodríguez

Dedico este trabajo en especial a mis padres Sucel Rodríguez Vega y James Valois Grinión Portillo por ser mi fuente de inspiración, preocuparse por mí cada día y por el esfuerzo realizado durante toda una vida.
A mi hermanita Yinet Grinión Rodríguez que la quiero mucho y por la cual me he esforzado tanto.
A mis abuelos Lourdes y Pablo por brindarme sus consejos y su cariño y por ser otros padres para mí.
A mis tías Maritza, Rosy, Beba, mis primas Dianelys y Daimerys, Diana y Betty y en especial a Cary por apoyarme a lo largo de estos cinco años y por todo el cariño y la ayuda que me han dado.
A mis tíos Kikito y Luisito por apoyarme como a una hija toda mi vida. A mis primos por su inmenso cariño.
A mis familiares en sentido general, que tanto han hecho por mí para que pudiera llegar a este día.
A mis amigas Mary, Milagros, Leinys, Yanara y especial a Yadira por estar siempre que las necesito, las que recordaré ¡Siempre!
A mi dúo (ex) de tesis por ser siempre paciente, por su cariño y apoyo.
A todas mis amistades en la UCI que siempre me han brindado su afecto y con los cuales he compartido momentos alegres y tristes.
A mis amigas de la infancia y la vocacional que se han preocupado siempre por mí y que me han brindado su cariño incondicional.
A todas aquellas personas que de una forma u otra contribuyeron en mi formación profesional y personal.
Gracias!!!

Daryanis Cedeño Barrero

Primeramente le agradezco y le dedico este trabajo a esa persona que ha dedicado la mayor parte de su vida a cuidarme y a protegerme. Gracias mi hermano querido, gracias José E Hernández por apoyarme tanto. Eres lo mejor que tengo en la vida.

A mi linda mamá María V. Barrero Tamayo por todo su apoyo y su confianza. Sin ella no hubiese sido posible llegar aquí.

A mi papá Guillermo Cedeño por todo el amor que me ha brindado, por toda su comprensión y por todo su cariño.

A Alejandro por siempre brindarme su ayuda incondicional, por ayudarme hacer mejor persona, por cuidarme, entenderme. Por su cariño y amor, por estar siempre presente para mí, por poner mis intereses por encima de los suyos.

A mi hermana Dayami Jorge Barrero por su cariño amor y cuidado y a mis lindos sobrinos por quererme y sentir que debo ser su ejemplo.

A toda mi familia por apoyarme y siempre estar atentos de mis estudios.

A mis otros padres Rafael Lao y Virgen Yudit Linares, por darme fuerza y apoyo, por estar siempre a mi lado cuando los necesito.

A mi Mérmere por estar siempre a mi lado, por brindarme su amistad y su cariño incondicional, por estar a mi lado en los mejores y peores momentos de mi vida. Más que una amiga se convirtió en mi hermana.

A mi querido dúo de tesis por soportarme y hacerme sentir fuerte. Por estar siempre dándome los mejores consejos.

A Francisquito por hacerme entender que la vida puede ser bella siempre y cuando la vivamos lo mejor que podemos.

A todos mis compañeros con los que compartí esta etapa de mi vida. Los extrañaré.

A todos los vecinos y amigos, a todos lo que de una forma u otra siempre estuvieron pendiente de mi en ese barrio que de por siempre será mi hogar, Tasajera.

¡A todos gracias!

Resumen

La Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología (CIGB) tiene como objetivo lograr un Sistema de Gestión de la Calidad eficiente. Este Centro manipula grandes volúmenes de información relacionada con los procesos que allí se realizan, lo cual es de gran importancia en el control y aseguramiento de la calidad. Toda esta información se almacena manualmente y en ocasiones se dificulta el acceso provocando la pérdida de tiempo de los trabajadores. Por la necesidad de contribuir a la gestión y control de la información que se genera, se está realizando un Sistema de Gestión de Información de los Laboratorios (LIMS, por sus siglas en inglés *Laboratory Information Management System),* en conjunto con la Universidad de las Ciencias Informáticas (UCI).

Luego de un estudio referente a los mecanismos de procesamiento y almacenamiento de la información y de que un grupo de analistas hiciese el Modelamiento de Negocio. El presente trabajo de diploma aborda el desarrollo del módulo Estudios de Estabilidad y Materiales de Referencia, donde se elaboran los materiales de referencia que se utilizan en las técnicas analíticas de todos los laboratorios, se realizan los estudios de estabilidad de todos los productos finales y principios activos que se producen, además de revisar y colaborar con el completamiento de toda la documentación necesaria ya sea para el registro de los productos en Cuba como fuera del país.

Se identifican las funcionalidades y características que deberá cumplir el módulo, se diseñan las clases del mismo y se implementan las funcionalidades. Se almacena la información referente a los procesos que se desarrollan, contribuyendo al manejo de los datos que se generan.

Palabras Claves:

Materiales de Referencia, Estudios de Estabilidad, CIGB, LIMS

Índice

Índice de Figuras

Índice de Tablas

Índices de Anexos

Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) forman parte de la cultura del mundo actual facilitando la interconexión entre las personas e instituciones. Las TIC proveen herramientas que ofrecen la posibilidad de encontrar soluciones novedosas ante los desafíos sociales de hoy. Se han convertido en un factor de desarrollo importante con profundas repercusiones en los sectores político, económico y social de numerosos países. Dichas tecnologías se presentan cada vez más como una necesidad en el contexto de sociedad donde los rápidos cambios, el aumento de los conocimientos y las demandas de una educación de alto nivel se convierten en una exigencia permanente.

Cuba ha defendido siempre el concepto de que el uso masivo de las TIC no es un fin, sino una herramienta poderosa para lograr el desarrollo. En los últimos años, ha emprendido el reto de la informatización de la sociedad, proyecto que se ha realizado de manera acelerada, auspiciado por la dirección del país y alcanzando resultados satisfactorios en áreas tan esenciales como la Educación, la Salud y la Investigación.

El Centro de Ingeniería Genética y Biotecnología (CIGB) forma parte del proyecto realizado para el desarrollo investigativo. “Su labor ha tenido gran impacto en la biomedicina, salud animal, mejoramiento vegetal y la bioindustria; y está encaminada a contribuir de manera significativa con el desarrollo económico y social del país. Con una labor de más de 20 años se han desarrollado nuevas vacunas y fármacos para la salud humana que se encuentran actualmente en uso dentro del Sistema de Salud Cubano, así como en diferentes países” [1].

Los productos desarrollados y elaborados en el CIGB se caracterizan por su eficacia y seguridad. Para ello la estructura del Centro cuenta con la Dirección de Calidad compuesta por las Direcciones de Control de la Calidad y Aseguramiento de la Calidad.

La Dirección de Aseguramiento de la Calidad “garantiza que se lleven a cabo las acciones planificadas y sistemáticas que son necesarias para proporcionar la confianza de que nuestros productos y servicios satisfacen los requisitos de calidad establecidos. Vela por el cumplimiento de las Buenas Prácticas de Producción (BPP), Buenas Prácticas de Laboratorio (BPL) y Buenas Prácticas Clínicas (BPC)” [1]. Esta Dirección está compuesta por dos departamentos y cuatro grupos de trabajo:

- ✓ Departamento de Inspección y Auditoría

- ✓ Departamento de Mejoramiento e Ingeniería de Calidad
- ✓ Grupo de Administración y Costos
- ✓ Grupo de Liberación
- ✓ Grupo de Documentación
- ✓ Grupo de Metrología

La Dirección de Control de la Calidad realiza varias funciones entre las cuales están las relacionadas con el muestreo, las especificaciones, los ensayos y la evaluación de la calidad de los productos que se generan. Además debe comprobar y poner en práctica todos los procedimientos de control, evaluación, mantenimiento y almacenamiento de los materiales de referencia, asegurar el control de la estabilidad de los ingredientes farmacéuticos activos y los productos terminados. Tiene la responsabilidad de autorizar o rechazar las materias primas y los productos intermedios. Para el desempeño de las mismas, cuenta con la ayuda de dos departamentos y tres grupos de trabajo (Ver Anexo 1):

- ✓ Departamento Biológico
    - Grupo de Microbiología
    - Grupo de Ensayos Biológicos I
    - Grupo de Inmunoquímica
    - Grupo de Ensayos Biológicos II
    - Grupo de Biología Molecular
- ✓ Departamento Físico Químico
    - Grupo de Aseguramiento Analítico
    - Grupo de Cromatografía y Electroforesis.
    - Grupo de Análisis Químico
    - Grupo de Sistemas Críticos
- ✓ **Grupo de Estudios de Estabilidad y Materiales de Referencia**
- ✓ Grupo de Administración y Costo
- ✓ Grupo de Liberación Analítica

El Grupo de Estudios de Estabilidad y Material de Referencia (EEMR) de la Dirección de Control de Calidad es responsable de los estudios de estabilidad de todos los productos finales y principios activos que se producen en el CIGB y que son registrados por el Centro Estatal para el Control de Calidad de los Medicamentos (CECMED). Es encargado además de revisar y colaborar con el completamiento de toda la documentación necesaria sobre el tema ya sea para el registro de los productos en Cuba como fuera del país.

En el Grupo también se elaboran los materiales de referencia que se utilizan en las técnicas analíticas de todos los laboratorios del Centro, esto implica los estudios de caracterización, homogeneidad y estabilidad que permiten el establecimiento de cada uno de ellos. Estos materiales de referencia se evalúan sistemáticamente cada 4 meses para comprobar que durante su tiempo de uso no exista alteración en las propiedades que fueron certificadas. También se lleva a cabo el control sistemático de todos estos patrones.

Los trabajadores de los laboratorios llenan una planilla de resultado, la envían al grupo de EEMR donde se hace los análisis estadísticos y se establecen todos los requisitos que deben tener los Materiales de Referencia para ser utilizados.

Los procesos generan grandes volúmenes de información impresa, por lo que se dificulta el almacenamiento organizado, confiable y duradero. Toda esta información es revisada, supervisada y aprobada por personas de cargos superiores, lo que en ocasiones provoca pérdida de tiempo, dificultando la toma inmediata de decisiones.

Por este motivo la Universidad de Ciencias Informáticas (UCI) conjuntamente con el CIGB comenzó el proyecto Sistema de Gestión de Información de Laboratorios de Calidad del CIGB. Tomando como antecedente el flujo de trabajo Modelamiento del Negocio realizado por analistas de la universidad, el presente trabajo de diploma pretende dar continuidad al proceso de desarrollo del módulo EEMR por lo que se plantea el siguiente ***problema científico***: ¿Cómo obtener un módulo funcional a partir de los procesos identificados en el módulo EEMR del Sistema de Gestión de la Información de Laboratorios de Calidad del Centro de Ingeniería Genética y Biotecnología?

El problema planteado se enmarca en el ***objeto de estudio***: El proceso de desarrollo de los Sistemas de Gestión de la Información de laboratorios.

El objeto delimita el ***campo de acción***: Proceso de desarrollo de los sistemas de gestión de la información de laboratorios basado en aplicaciones Web.

Para dar solución al problema se define como ***objetivo general***: Desarrollar el módulo EEMR para el Sistema de Gestión de la Información de Laboratorios de Calidad del CIGB.

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

- Realizar el levantamiento de requisitos del módulo EEMR.
- Realizar el diseño del módulo EEMR.
- Diseñar el Modelo Lógico de Datos para la Base de Datos.
- Implementar el módulo EEMR.

Para lograr los objetivos propuestos, se realizarán las siguientes ***tareas***:

- Estudio del Modelamiento del Negocio.
- Definición de los requisitos funcionales y no funcionales.
- Desarrollo del diagrama de casos de uso del sistema.
- Realización de los casos de uso del sistema.
- Realización de CU-diseño: diagramas de clases del diseño y diagramas de interacción.
- Elaboración del Mapa de Navegación.
- Identificación de las clases persistentes.
- Diseño del Diagrama Entidad – Relación.
- Realización de los diagramas de componentes.
- Implementación de los componentes.
- Realización de Pruebas de Unidad.

El presente trabajo está estructurado en cuatro capítulos. A continuación aparece de manera resumida el contenido de cada capítulo:

Capítulo 1 **"Fundamentación Teórica"**, se describe el estudio del estado del arte de las compañías más reconocidas a nivel mundial en la producción de LIMS. Se presenta la metodología de desarrollo a seguir en este trabajo. Se aborda sobre el rol, los artefactos y las actividades a realizar en los flujos de

trabajo Requerimientos, Diseño e Implementación. Se refleja el lenguaje de desarrollo web, la herramienta CASE y el framework utilizado en la implementación del módulo.

Capítulo 2 **"Características del Sistema"**, se definen los requisitos funcionales, los no funcionales, se describen los actores del sistema y se muestra el diagrama de casos de uso del sistema. Se realiza la descripción textual de los casos de uso definidos hasta el momento, se realiza la matriz de trazabilidad obtenida entre los requerimientos funcionales y casos de uso del sistema.

Capítulo 3 **"Diseño del Sistema"**, se muestran los prototipos no funcionales, se representan los diagramas de clases del diseño para aplicaciones Web, los diagramas de secuencia del diseño, el mapa de navegación y el modelo de datos. Se hace referencia a la arquitectura del sistema a través de la Vista Lógica y de Despliegue.

Capítulo 4 **"Implementación del Sistema"**, aborda la programación realizada partiendo de los requerimientos identificados y los diagramas de clases del diseño elaborados. Se presenta el diagrama de componentes correspondiente a cada uno de los casos de uso implementados. Se muestran fragmentos relevantes del código y ejemplos de las validaciones realizadas.

Capítulo 1
Fundamentación Teórica

Introducción

En este capítulo se describe el estudio del estado del arte de las compañías más reconocidas a nivel mundial en la producción de LIMS. Se aborda sobre los Sistemas de Gestión de Información de forma general. Se presenta el Proceso Unificado de Desarrollo (RUP) como metodología a seguir en este trabajo. Se incluyen los roles, artefactos y actividades a realizar en los flujos de trabajo Requerimientos, Análisis y Diseño e Implementación. Se refleja el lenguaje de desarrollo web, la herramienta CASE y el framework utilizado en la implementación del módulo.

1.1. Gestión de la Información

Actualmente la sociedad presenta organizaciones basadas en el aprendizaje, donde el capital más preciado es el ser humano, sustentándose en un desarrollo tecnológico constantemente en evolución, en el cual las grandes compañías planifican sus productos en función de la gestión de la información y de la viabilidad para su obtención.

“La gestión de la información es el proceso de analizar, utilizar, recuperar y almacenar la información que se ha obtenido y registrado, para permitir a los administradores tomar decisiones documentadas.” [1]

Su función es facilitar información precisa para la toma de decisiones, sin preocuparse por otros aspectos relacionados con el aprendizaje. Tiene una visión más mecanicista, el elemento humano tiene menos importancia. Implica determinar la información que se precisa, recoger y analizar la información, registrarla y recuperarla cuando sea necesaria, utilizarla y divulgarla.

Algunas de las funciones de la Gestión de Información [2]:

- Determinar necesidades internas de información, relativas a las funciones, actividades y procesos administrativos de la organización y a su satisfacción.

- Optimizar el flujo organizacional de la información y el nivel de la comunicación.
- Manejar eficientemente los recursos organizacionales de información.
- Entrenar a los miembros de la organización en el manejo o la utilización de los recursos informacionales.
- Contribuir a modernizar u optimizar las actividades organizativas y los procesos administrativos relacionados con los mismos.
- Garantizar la calidad de los productos de la organización y asegurar su diseminación efectiva.
- Determinar las necesidades de información externa de la organización y satisfacerlas.

1.1.1. Sistema de Gestión de la Información

"Un sistema de gestión de la información es una estructura probada para la gestión y mejora continua de las políticas, los procedimientos y procesos de la organización" [3]. El uso de un sistema de gestión probado le permite renovar constantemente su objetivo, sus estrategias, sus operaciones y niveles de servicio.

Son necesarios debido a que equilibran requisitos empresariales tales como:

- Rentabilidad
- Competitividad
- Globalización
- Velocidad de los cambios
- Capacidad de adaptación
- Crecimiento
- Tecnología

Estos requisitos pueden constituir un proceso difícil y desalentador, los sistemas de gestión permiten aprovechar y desarrollar el potencial existente en la organización.

La implementación de un sistema de gestión eficaz puede ayudar a:

- Gestionar los riesgos sociales, medioambientales y financieros
- Mejorar la efectividad operativa
- Reducir costos
- Aumentar la satisfacción de clientes y partes interesadas
- Proteger la marca y la reputación
- Lograr mejoras continuas

1.1.2. Sistema de Gestión de la Información de Laboratorios

A lo largo de la historia la ciencia ha sido capaz de alcanzar un alto grado de madurez, ocupando un lugar muy importante en la sociedad del conocimiento. Como consecuencia directa al desarrollo alcanzado en centros vinculados con la investigación y la producción al mismo tiempo, el flujo de la información manejada en los laboratorios de dichos centros ha aumentado con el transcurso de los años. Con el objetivo de flexibilizar el manejo de la información, se produce la invención de un software que gestiona toda la información producida en dichos laboratorios, y estos son los Sistemas de Gestión de la Información de Laboratorios (LIMS).

"Un LIMS proporciona un conjunto de herramientas basadas en Sistemas Informáticos que permiten la aplicación de técnicas de adquisición y gestión avanzada de la información producida en el laboratorio. La utilización de los LIMS ha favorecido en gran medida la manipulación de los datos que se obtienen en los laboratorios." [1]

Entre los beneficios más comunes que tiene su utilización se pudieran mencionar:

- Reproducibilidad en cualquier momento de toda la documentación generada con anterioridad.
- Control de cualquier cambio en todas aquellas zonas críticas de la información del laboratorio, para poder impedir modificaciones no autorizadas y además guardar una trazabilidad de los mismos.
- Aumento de la cantidad de información disponible y requisitos de manipulación de la misma.
- Aseguramiento de la calidad.
- Adquisición manual o automática de datos.
- Mejoras en el procesamiento de la información y en la productividad.
- Revisión y visualización de datos más completa, flexible y accesible.

- Centralización de la información en una única base de datos, lo que conlleva que la accesibilidad a los datos sea más segura, rápida, cómoda, sencilla.

1.1.3. Productos LIMS

Dado el creciente interés de las empresas por alcanzar una alta calidad, como un factor clave para favorecer la productividad, la eficacia y la imagen de los productos o servicios suministrados, las aplicaciones LIMS representan la solución imprescindible para una gestión del control y aseguramiento de la calidad.

A continuación aparecen reflejadas algunos productos de LIMS.

✓ **Labmatica** es un Sistema de Gestión de Información de Laboratorio (LIMS), este permite realizar:

- Gestión de muestras, para crear los tipos de muestras y los parámetros de los métodos.
- Generar fácilmente informes y certificados.
- Afinar los derechos de los usuarios.
- Se puede configurar fácilmente plantillas.
- El software es cliente / servidor con la solución de múltiples capacidades de usuario. Está escrito en JAVA y publicado bajo doble licencia a) la licencia GNU General Public License (GPL) y b) una licencia comercial. GNU / GPL versión se ejecuta en base de datos del servidor MySQL. La versión comercial se puede utilizar en otros sistemas de bases de datos relacionales.

✓ **BIKA LIMS:** Bika LIMS se estructuran a un servidor estándar de edición para que los clientes puedan agregar módulos opcionales y personalizaciones. Bika emplea la tecnología moderna. Es independiente de la plataforma y basados en la web. Se construye en Plone, un marco de gestión de contenido que está estrechamente integrado con Zope, que es un servidor de aplicaciones Web ampliamente utilizado de fuente abierta orientado a objetos. Esta liberado bajo la licencia pública general (GPL).

✓ **Open LIMS:** El objetivo de Open LIMS es desarrollar una organización y estructura independiente de software. Los módulos son generalmente escrito en PHP. Para el cálculo de resultados cuenta

con el sistema de trabajo que está escrito en Java y requiere Java VM (Virtual Machine). Es posible trabajar en proyectos paralelos. Proporciona módulos experimentales para la posterior análisis. Es posible organizar el experimento con diferentes sub-proyectos. Cada proyecto puede tener un número ilimitado de sub-proyectos. Se basa en plantillas las cuales están escritos en XML (Extensible Markup Language). Permite al propietario del proyecto establecer toda una serie de permisos, dividido en diferentes tipos de permisos. Utiliza la codificación de caracteres UTF8. Publicado bajo la licencia GPL. Todavía se encuentra en desarrollo. Es desarrollado por Roman Konertz la última actualización fue el 16 de enero 2009.

- ✓ **STARLIMS** es uno de los proveedores de LIMS con mayor crecimiento en el mundo, con un claro enfoque dirigido hacia la creación de fuertes relaciones con los clientes y el desarrollo de soluciones de calidad para la gestión de información de laboratorios. Aporta soluciones rentables y fáciles de utilizar para todos los mercados importantes. Todos utilizan la misma base de software STARLIMS, el cual es configurado y adecuado para cubrir las necesidades de cada organización y reglas de negocio.
- ✓ **Veolab** es un software de gestión para laboratorios de análisis bajo ISO 9001. Con una interfaz totalmente visual para facilitar el uso y aprendizaje. Fácil de usar, interfaz amigable, totalmente configurable, gestión y planificación de muestras y auditorías, informes de resultados personalizables, facturación integrada completa y sencilla.
- ✓ **La Suite Pharma LIMS** es una serie de aplicaciones LIMS destinadas a las industrias farmacéuticas. Esta Suite incluye entre sus aplicaciones LIMS a:
  - Newton LIMS para el lanzamiento de pruebas, control de calidad, formulación, estabilidad y desarrollo analítico.
  - Watson LIMS para bio-análisis pre-clínicos y clínicos.
  - Galileo LIMS para las pruebas In Vitro.

  El fabricante de la Suite Pharma LIMS es InnaPhase Corporation, empresa que más tarde fue adquirida por Thermo Electron Corporation de los Estados Unidos.

- ✓ **MSC LIMS** es un Sistema de Gestión de Información para Laboratorios pequeños. Permite a los encargados del laboratorio incluir hipervínculos a los documentos externos tales como correspondencia o facturas, e imprimir toda la información para los clientes. MSC LIMS fue desarrollado por Mountain States Consulting de los Estados Unidos.

- ✓ **Labworks LIMS** es un Sistema de Gestión de Información de Laboratorios que opera en diferentes plataformas según la cantidad de usuarios que utilicen el sistema y se adapta a diferentes tipos de empresas. Es modular, por lo que el sistema puede expandirse de acuerdo a las necesidades de trabajo. Las especificaciones de Labworks LIMS son: el ingreso de la muestra se realiza de forma manual y por códigos de barras, comunicación directa con los instrumentos y realización de informes automáticos y programación de mantenimiento y calibración del instrumento. Labworks LIMS fue desarrollado por PerkinElmer, empresa de Estados Unidos.

- ✓ **LabSoft LIMS** es un sistema para la administración de datos, monitoreo de los procesos y generación de certificados de análisis. Es fácil de usar, proporciona gran flexibilidad para la gestión de los datos del laboratorio y notifica sobre un acontecimiento específico en el sistema. Labsoft LIMS fue desarrollado por Computing Solutions Inc. (CSI), empresa de los Estados Unidos que ha estado proporcionando LIMS exclusivamente a la industria que soportan productos químicos, petroquímicos y productos alimenticios.

- ✓ **Matrix LIMS** es un Sistema de Gestión de Información de Laboratorios que se caracteriza por la flexibilidad de configuración, por ser adecuado para laboratorios pequeños y grandes organizaciones, incluye herramientas que son necesarias para cumplir los requerimientos de los laboratorios altamente regulados, es compatible con las bases de datos Oracle y Microsoft SQL Server, presenta una interfaz de usuario sencilla. Tiene tres opciones de producto: Matrix Express, Matrix Plus y Matrix Enterprise para brindar respuesta a las necesidades de cualquier laboratorio. Matrix LIMS considerado actualmente el estándar en la industria al que otros aspiran. El fabricante de Matrix LIMS es Autoscribe Limited del Reino Unido.

- ✓ **QUAASS-LAB Elite** tiene como objetivo satisfacer las necesidades de los laboratorios que necesitan tener un riguroso control de calidad y una completa gestión de toda la información de las muestras analizadas. Este Sistema de Gestión de Información de Laboratorios se desarrolló sobre una plataforma de hardware y software basada en estándares de mercado y una arquitectura cliente/servidor. QUAASS-LAB Elite fue creado por la empresa QUAASS, de Barcelona, España.

Luego de analizar el origen de los Sistemas de Gestión de la Información de Laboratorios se arriba a la conclusión de que es más factible desarrollar un LIMS que comprarlo puesto que se exige un precio muy alto, por ejemplo, el STARLIMS tiene un costo promedio entre 10 000 y 45 000 dólares

mensuales.
Además, la mayoría de sus proveedores son de Estados Unidos lo que implica que sea casi imposible adquirirlo debido al bloqueo que el gobierno de dicho país tiene contra Cuba. Por otro lado el CIGB tiene características específicas que impiden poder configurar un LIMS para su uso particular ya que se realizan ensayos diferentes a diferentes productos y eso provoca que se necesite un sistema adaptable a las necesidades específicas de la Dirección de Calidad.

1.2. Metodología y herramientas a utilizar

Desde el comienzo del proyecto Sistema de Gestión de Información de Laboratorios de Calidad del CIGB se definieron las herramientas y metodología a emplear. Debido al constante avance de las tecnologías se ha tenido que emplear nuevas versiones de dichas herramientas en el desarrollo del mismo, que serán abordadas en el presente epígrafe.

1.2.1. Procesos de Desarrollo de Software

El proceso de desarrollo de software "es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo" [4]. Tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente.

No existe un proceso de software universal que sea efectivo para todos los proyectos de desarrollo. A pesar de la variedad de propuestas de proceso de software, existe un conjunto de actividades fundamentales que se encuentran presentes en todos ellos**.**

1. **Especificación de software**: Se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.
2. **Diseño e Implementación**: Se diseña y construye el software de acuerdo a la especificación.
3. **Validación**: El software debe validarse, para asegurar que cumpla con lo que quiere el cliente.
4. **Evolución**: El software debe evolucionar, para adaptarse a las necesidades del cliente.

"Además de estas actividades fundamentales, Pressman menciona un conjunto de "actividades protectoras", que se aplican a lo largo de todo el proceso del software. Ellas se señalan a continuación" [5]:

- Seguimiento y control de proyecto de software.
- Revisiones técnicas formales.
- Garantía de calidad del software.
- Gestión de configuración del software.
- Preparación y producción de documentos.
- Gestión de reutilización.
- Mediciones.
- Gestión de riesgos.

Existen dos métodos muy ligados al desarrollo de los procesos de desarrollo de software, los llamados métodos pesados y métodos ligeros. Ambos métodos están encaminados a favorecer el trabajo de las personas que intervienen en dicho proceso. La diferencia fundamental entre ambos es que mientras los métodos pesados intentan conseguir el objetivo común por medio de orden y documentación, los métodos ligeros, también llamados métodos ágiles tratan de mejorar la calidad del software por medio de una comunicación directa e inmediata entre las personas que intervienen el proceso.

Entre los procesos de desarrollo más conocidos se tienen: Programación Extrema (XP, por sus siglas en inglés Extreme Programming) clasificado como método ligero, Desarrollo Guiado por la Funcionalidad (FDD, por sus siglas en inglés Feature Driven Development) clasificado como un proceso de término medio, debido a que puede ser un método ligero o pesado dependiendo del gusto del que lo va a usar y Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés Rational Unified Process) que se clasifica como método pesado. Este último ha sido definido por el proyecto como metodología de desarrollo a utilizar.

1.2.2. RUP

RUP (por sus siglas en inglés Rational Unified Process) "…es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del Lenguaje

Unificado de Modelado UML, y trabajo de muchas metodologías utilizadas por los clientes. La versión que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo" [6]. Se caracteriza por ser dirigido por casos de uso (CU), centrado en la arquitectura y por ser iterativo e Incremental.

Que este dirigido por casos de uso significa que los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos.

Que sea centrado en la arquitectura significa que la arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción.

Que sea un proceso iterativo e incremental posibilita, que se pueda dividir el trabajo en partes más pequeñas o en mini proyectos, permitiendo el equilibrio entre casos de uso y arquitectura durante cada mini proyecto. Cada mini proyecto se puede ver como una iteración de la cual se obtiene un incremento, provocando un aumento del producto.

RUP es uno de los procesos más generales del los existentes actualmente, ya que se adapta a cualquier proyecto. En este se puede ver la evolución del software en cuatro fases, al final de las cuales, y tras una serie de iteraciones, establece objetivos a alcanzar bien definidos:

- **Concepción o Inicio:** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.
- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, 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.
- **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene 1 o varios release del producto que han pasado las pruebas. Se ponen estos release a consideración de un subconjunto de usuarios.
- **Transición:** El release ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

Cada una de las fases antes mencionadas concluye con un hito bien definido, en el cual se deben tomar ciertas decisiones y alcanzar las metas clave antes de pasar a la siguiente fase. Los hitos para cada una de las fases son: Inicio -visión de los objetivos-, Elaboración -prototipo de la arquitectura-, Construcción - capacidad operacional inicial- y Transición -liberación del producto-. (Ver Anexo 2). Cada fase se completa con la realización de varias iteraciones en las que se desarrollan una serie de actividades, clasificadas en 9 disciplinas o flujos de trabajos. Los seis primeros son conocidos como flujos de ingeniería y los tres últimos como flujos de apoyo. Los flujos de trabajos son: Modelamiento del negocio, Requerimientos, Análisis y diseño, Implementación, Prueba (Testeo), Instalación, Administración del proyecto, Administración de configuración y cambios y Ambiente. (Ver Anexo 3).

- **Modelado del negocio:** en este flujo se realiza el entendimiento de las necesidades de negocio. Documentos de requisitos generales, reglas del negocio y glosarios de términos que ayudan a definir lo que el producto de software debe hacer.
- **Requisitos**: se traducen las necesidades del modelo de negocio a requisitos de sistemas automatizables. Se establecen las funcionalidades con las que debe cumplir el sistema.
- **Análisis y diseño**: se determinan a partir de los requisitos, la arquitectura del sistema más adecuada y el diseño detallado necesario para las actividades de implementación que se realizarán.
- **Implementación**: se implementan las clases y objetos en ficheros fuente, binarios, ejecutables. Con este flujo se ve si se cumplen con los requisitos del sistema de acuerdo al diseño realizado. El resultado final es un sistema ejecutable.
- **Pruebas**: comprobaciones hechas a todos los elementos que se producen (documentos, diseños o código) para ver que cumplen con los requisitos y con los estándares de calidad definidos para el proyecto.
- **Despliegue**: actividades que permiten tener el sistema instalado en los entornos en que finalmente va a ser explotado.
- **Gestión de configuración:** gestión de los cambios y todos los elementos que intervienen en el proceso de construcción.
- **Gestión del proyecto:** flujo encaminado a la gestión del desarrollo en cuanto a planes, recursos, seguimiento, control y gestión de riesgos.
- **Entorno:** actividades que van encaminadas a dotar al proyecto de recursos hardware y software para facilitar la puesta en marcha y mantenimiento de los distintos entornos de desarrollo y pruebas o la propia puesta en producción del sistema.

En el presente trabajo se desarrollan los flujos de trabajo: Requerimientos, Análisis y diseño (se abordará solo diseño) e Implementación.

1.2.2.1. Artefactos a realizar

"Un artefacto es una pieza de información que es producida, modificada o usada por el proceso, define un área de responsabilidad para un rol y está sujeta a control de versiones. Un artefacto puede ser un modelo, un elemento de modelo o un documento" [4]. Los artefactos son los resultados tangibles del proyecto que se va creando y usando hasta obtener el producto final, este describe por sí mismo la esencia de un proceso, en dependencia de la etapa por la que transite en el proyecto que se está ejecutando. Los artefactos son generados en los distintos flujos de trabajo como consecuencia directa de las actividades que tengan que desempeñar los trabajadores en el mismo. A continuación se describen los artefactos que serán generados en los diferentes flujos de trabajo.

Flujo de Trabajo Requerimiento. **[7]**

- **Modelo de casos de uso:** Es un modelo del sistema que contiene actores, casos de uso y sus relaciones.
- **Caso de uso**: Fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.
- **Glosario de términos:** Términos comunes que se utilizan para describir el sistema.
- **Especificación de requisitos:** Captura los requerimientos de software para el sistema completo o una porción del sistema.
- **Paquetes de caso de uso:** Es una colección de casos de uso, actores, relaciones, diagramas, y otros paquetes, que son usados para estructurar el modelo de casos de uso dividiéndolo en pequeñas partes.

Flujo de Trabajo Diseño. [8].

- **Modelo de diseño:** Modelo de objetos que describe la realización física de los casos de uso en cómo los requisitos funcionales, no funcionales y otras restricciones del entorno de implementavión tienen impacto en el diseño. Está compúesto por subistemas de diseño, clases de diseño, realizaciones de casos de uso e Interfaz.

- **Clases de diseño:** Es una abstracción de una clase o construcciones similares de implementación.
- **Realización de casos de uso en el diseño:** Diagrama de clases + Diagrama de interacción + Descripción de los casos de uso + Requisitos de implementación.
- **Subsistemas de diseño:** Forma de organizar los artefactos de diseño de manera más manejable. Al ser mecanismos de agrupamiento tiene que cumplir las mismas características de los paquetes.
- **Interfaz:** Especifican las operaciones que proporcionan las clases y los subsistemas de diseño.
- **Diseño de la Base de Datos:** Estructura de la información que se desea persistente a partir del medio de almacenamiento que se elija.
- **Vista lógica:** Muestra los componentes principales de diseño y sus relaciones de forma independiente de los detalles técnicos y de cómo la funcionalidad será implementada en la plataforma de ejecución. Se describe la solución en términos de paquetes y clases de diseño.
- **Modelo de despliegue:** Modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo.
- **Vista física:** Muestra la distribución del procesamiento entre los distintos equipos que conforman la solución, incluyendo los servicios y procesos de base.

Flujo de Trabajo Implementacion.

- **Modelo de Implementación:** Está conformado por los diagramas de despliegue y componentes. Se describen los componentes a construir y su organización y dependencia entre nodos físicos en los que funcionará la aplicación.
- **Subsistemas de Implementación:** Es la colección de componentes y otros subsistemas de implementación usados para estructurar el modelo de implementación y dividirlos en pequeñas partes que pueden ser integradas y probadas de forma separada.

1.2.2.2. Actividades de los flujos de trabajo

Una actividad es la unidad de trabajo donde se mide el esfuerzo de una persona o individuo durante su desempeño en uno de los roles definidos por RUP. Es la agrupación de todo un conjunto de tareas a cumplir por una persona y que forman parte de un proceso en específico.

Para construir los artefactos generados en los diferentes flujos de trabajo antes mencionados se realizan las siguientes actividades:

Flujo de Trabajo Requerimiento [7]

- Identificar y clasificar requerimientos.
- Encontrar actores y casos de uso.
- Priorizar casos de uso.
- Detallar casos de uso.
- Estructurar el modelo de casos de uso.

Flujo de Trabajo Diseño. [8]

- **Diseño de casos de uso:** Identificar los objetos que son necesarios para realizar el caso de uso, distribuir el comportamiento entre los objetos y capturar los requisitos de implementación del caso de uso.
- **Diseñar las clases:** Identificar las clases de diseño a través de las cuales se realicen los casos de uso especificando sus atributos, operaciones, métodos y relaciones entre ellas.
- **Diseñar subsistemas:** Identificar subsistemas cohesivos y con bajo acoplamiento.

Flujo de trabajo de Implementación.

Estructurar el modelo de Implementación: actualiza el Documento de la Arquitectura del Software, añadiendo la Vista de Implementación. Se actualiza el modelo de implementación ya sea porque se crea por primera vez o se añaden nuevos elementos de Implementación a los ya existentes provenientes de una iteración anterior.

Implementar Componentes: Se completa una parte de la implementación que podría ser entregada para la integración. Aparecen además las actividades de revisión de código y el plan de integración de subsistemas.

1.2.2.3. Roles que se desempeñan en la investigación

El rol define las diferentes responsabilidades que deben cumplir uno o varios individuos en cierto flujo de trabajo, ya que cada persona tiene tareas diferentes que realizar según la etapa en que se encuentre el proyecto. Una persona puede desempeñar diversos roles y un rol puede ser representado por varias personas en un mismo momento. Los roles varían en dependencia de la fase o del flujo de trabajo por donde transite el proyecto.

Para el desarrollo de este trabajo se cumplieron con diversos roles que define RUP, a medida que se realizaban los diferentes flujos de trabajo.

- **Analista del sistema:** Define el alcance del sistema e identifica a los actores y casos de uso que permiten modelar el sistema y estructura el modelo de casos de uso. (Requerimientos)
- **Ingeniero de casos de uso:** Responsable de la integridad de las realizaciones de los casos de uso a él asociados y que estén en correspondencia con los requisitos a ellos asociados. (Diseño)
- **Diseñador:** Dirige el diseño de una parte del sistema, dentro de las restricciones de los requisitos, arquitectura y proceso de desarrollo. El diseñador identifica y define las responsabilidades, operaciones, atributos y relaciones de los elementos de diseño. El diseñador se asegura de que el diseño sea coherente con la arquitectura de software, y que esté detallado hasta un punto en que pueda proceder la implementación. (Diseño)
- **Diseñador de Interfaz de Usuario:** Conduce y coordina los prototipos y el diseño de la interfaz de usuario, no es responsable de la implementación de dicha interfaz, su papel solo se centra en el diseño y la "conformación visual".(Diseño)
- **Diseñador de Base de Datos:** Dirige el diseño de la estructura de almacenamiento de datos persistentes que se utilizará en el sistema. Define el diseño detallado de la base de datos, incluyendo tablas, índices, vistas, restricciones, desencadenantes, procedimientos almacenados y otras construcciones específicas de la base de datos necesarias para almacenar, recuperar y suprimir objetos persistentes.
- **Implementador:** Estructura la forma que va a adoptar el modelo de implementación y la correspondencia de este con el modelo de diseño.(Implementación)

1.2.3. Patrones

"Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Un sistema bien estructurado está lleno de patrones."[9]

Cada patrón describe un problema que ocurre consecutivamente en el ambiente y luego describe el núcleo de la solución a ese problema, de tal manera que se pueda usar esa solución varias veces más, sin tener que repetir la misma cosa dos veces.

Con el objetivo de que sea más fácil la utilización de los patrones se presentan elementos concretos de estos, los cuales se describen a continuación:

- **Nombre del patrón**: nombre estándar del patrón por el cual será reconocido en la comunidad (normalmente se expresan en inglés).
- **Clasificación del patrón**:
- **Intención**: ¿Qué problema resuelve el patrón?
- **También conocido como**: Otros nombres de uso común para el patrón.
- **Motivación**: Escenario de ejemplo para la aplicación del patrón.
- **Aplicabilidad**: Criterios de aplicabilidad del patrón.
- **Estructura**: Diagramas de clases oportunos para describir las clases que intervienen en el patrón.
- **Participantes**: Enumeración y descripción de las entidades abstractas (y sus roles) que participan en el patrón.
- **Colaboraciones**: Explicación de las interrelaciones que se dan entre los participantes.
- **Consecuencias**: Consecuencias positivas y negativas en el diseño derivadas de la aplicación del patrón.
- **Implementación**: Técnicas o comentarios oportunos de cara a la implementación del patrón.
- **Código de ejemplo**: Código fuente ejemplo de implementación del patrón.
- **Usos conocidos**: Ejemplos de sistemas reales que usan el patrón.
- **Patrones relacionados**: Referencias cruzadas con otros patrones.

Algunos patrones identificados en el desarrollo del trabajo son: Patrones de Casos de Uso, Patrones de Diseño, Patrones GRASP y Patrones GOF, los cuales se muestran en los epígrafes posteriores. La utilización específica de estos patrones se verá en el Capitulo 3.

1.2.4. Patrones de Casos de Uso

A las acciones básicas de cualquier sistema de información ya sea de entrada, salida de datos, procesamiento y almacenamiento de información se le denomina CRUD (por sus siglas en inglés *Creating, Reading, Updating and Deleting*, traducido al español: Crear, Leer, Actualizar y Eliminar).

CRUD propone identificar un CU, llamado "Información CRUD" o "Administrar Información", que modela todas las operaciones que se pueden realizar sobre una parte de información de cierto tipo. Debe ser usado cuando todos los flujos contribuyen al mismo valor de negocio, son cortos y sencillos de usar.

1.2.5. Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Se puede definir un patrón, además, como un proyecto o estructura de implementación que logra una finalidad determinada donde se pone de manifiesto la manera más práctica de describir ciertos aspectos de la organización de un programa. Este identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Posibilita flexibilidad en el código haciéndolo satisfacer ciertos criterios.

Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe formas de solucionar problemas que ocurren de forma frecuente en el desarrollo. Por tanto, están basados en la recopilación del conocimiento de los expertos en desarrollo de software, lo que lo convierte en su principal ventaja.

Estos patrones presentan características específicas de las que se pueden mencionar las siguientes:

- Son soluciones concretas, se proponen soluciones a problemas concretos, no son teorías genéricas.
- Son soluciones técnicas, indican resoluciones técnicas basadas en Programación Orientada a Objetos. En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje.
- Se utilizan en situaciones frecuentes ya que se basan en la experiencia acumulada la resolver problemas reiterativos.

• Favorecen la reutilización de código, ya que ayudan a construir software basado en la reutilización, ayudan a construir clases reutilizables. Los propios patrones se reutilizan cada vez que se vuelven a aplicar.

• El uso de un patrón no se refleja en el código. Al aplicar un patrón, el código resultante no tiene por qué delatar el patrón o patrones que lo inspiró.

1.2.6. Patrones GRASP

Los patrones GRASP (por sus siglas en inglés *General Responsability Assignment Software Patterns*), Patrones de Software para la asignación General de Responsabilidad describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

Se pueden destacar 5 patrones principales y 4 adicionales que se especifican a continuación:

Los 5 patrones principales son:

Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados, es decir, los atributos. Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada.

Creador: Este patrón como su nombre lo indica es el que crea, el guía la asignación de responsabilidades relacionadas con la creación de objetos.

Alta cohesión: Asigna una responsabilidad de manera que la cohesión permanezca alta. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos.

Bajo acoplamiento: Asigna una responsabilidad de manera que el acoplamiento permanezca bajo. Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades.

Controlador: Asigna el responsable de gestionar un evento de entrada al sistema. Es un evento generado por actores externos. Se asocian con operaciones del sistema, operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos.

Normalmente un controlador delega en otros objetos el trabajo que se necesita hacer; coordina o controla la actividad. No realiza mucho trabajo por sí mismo.

1.2.7. Patrones GOF

Según GOF (por sus siglas en inglés *Gang-Of-Four*), también conocido como el de la banda de los 4, existen 3 tipos de patrones:

- **De Creación:** Abstraen el proceso de creación de instancias, ocultan los detalles de cómo los objetos son creados o inicializados.
- **Estructurales:** Se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.
- **De Comportamiento:** Ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

Cada uno de estos patrones contiene patrones específicos. A continuación se muestras algunos de esos patrones.

Patrones Creacionales

- **Abstract Factory** o Fábrica abstracta: Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.
- **Singleton** o Instancia única: Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Patrones Estructurales

- **Composite** (Objeto compuesto): Permite tratar objetos compuestos como si fuese un objeto simple.
- **Decorator** (Envoltorio): Añade funcionalidad a una clase dinámicamente.

1.2.8. Lenguaje de modelado: UML.

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés *Unified Modeling Language*) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Permite la

modelación de sistemas con tecnología orientada a objetos. No es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso.

UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. Este ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. (Ver Anexo 4).

Los diagramas que se representan en el presente trabajo son los de casos de uso, clases, secuencia, componentes y despliegue.

1.2.9. Herramientas CASE

La Ingeniería de Software Asistida por Computación (CASE, por sus siglas en inglés *Computer Aided Software Engineering*), se considera como la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación. Dichos programas dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software.

Las herramientas CASE permiten organizar y manejar la información de un proyecto informático. Permitiéndoles a los participantes de un proyecto, que los sistemas se tornen más flexibles y comprensibles.

Algunos de los componentes de las herramientas CASE permiten: [10]

- Confeccionar la definición de requerimientos de los usuarios.
- Mejorar el diseño de los sistemas.
- Mejorar la eficiencia en la programación (por su generación automática de códigos).
- Otorgar a la administración un mejor soporte en la documentación.

Entre las herramientas CASE para el modelado de artefactos que existen a nivel mundial están las herramientas MetaCASE, Embarcadero ER/Studio, Erwin, Umbrello, MagicDraw, Rational Rose y Visual Paradigm. Esta última herramienta CASE, Visual Paradigm for UML en su versión 6.1 es la

seleccionada para el modelamiento de los diferentes diagramas, no solo por sus ventajas y características, sino que fue la herramienta definida por la dirección del proyecto para el desarrollo del mismo.

1.2.9.1. Características fundamentales que ofrece Visual Paradigm

Visual Paradigm es una herramienta CASE que utiliza UML, provee soporte para la generación de código, ingeniería inversa para Java, se integra con Eclipse, Borland® JBuilder®, Oracle JDeveloper y NetBeans, para soportar las fases de implementación en el desarrollo de software. Permite realizar ingeniería tanto directa como inversa a partir de un modelo relacional en SQL Server, o MySql, es capaz de desplegar todas las clases asociadas a las tablas.

Algunas de las características que pueden ser mencionadas de esta herramienta son las siguientes:

- Es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto.
- Genera la documentación del proyecto automáticamente en varios formatos como Web, .pdf, .doc, y permite control de versiones.
- Es una herramienta multiplataforma de modelado visual UML ya que aporta a los desarrolladores de software una plataforma de desarrollo para construir aplicaciones de calidad.
- Aporta una excelente interoperabilidad con otras herramientas CASE y muchos de los entornos IDE del mercado como NetBeans y Eclipse.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.

1.3. Herramienta para la Gestión de Requerimientos

Las herramientas para la gestión de requisitos dan soporte a las tareas que son deducidas en el proceso de gestión de requisitos, donde surge la necesidad de apoyarse en herramientas que den soporte a todas estas tareas. “Entre las que se pueden mencionar” [11]:

- Recogida de requisitos
- Formalización de requisitos
- Revisión
- Gestión

Dentro de las herramientas para la Gestión de Requerimientos se pueden mencionar: REM, RTH, OSRMT y Rational RequisitePro, esta última se caracterizará a continuación ya que es la que se utilizó en el proyecto para la gestión de requisitos.

1.3.1. Rational RequisitePro

Cuanto mejor sea la comunicación y administración de requerimientos, mayor será la probabilidad de que los proyectos se entreguen a tiempo, cuanto más efectiva sea la gestión de requerimientos, mayor será el resultado en calidad y satisfacción del cliente. La administración de requerimientos se logra con la utilización del software Rational RequisitePro para la gestión de requerimientos.

Rational RequisitePro es la herramienta de gestión de requisitos que ofrece IBM. Facilita que los miembros del equipo colaboren con los requerimientos del proyecto. La interfaz es sencilla e incluye un cliente Windows y uno Web. Permite al equipo crear y compartir sus requerimientos utilizando métodos familiares basados en documentos potenciados por la aplicación de las capacidades de una base de datos, tales como la trazabilidad y análisis de impacto. El resultado es una mejor comunicación y administración de requerimientos con una mayor probabilidad de completar los proyectos en tiempo, dentro del presupuesto y superando las expectativas.

La utilidad que se le da a dicha herramienta en el desarrollo del proyecto es en la creación de diferentes matrices, la generada por dicha herramienta es la matriz de trazabilidad entre Requisitos y Casos de Usos del Sistema.

Las matriz realizada se encuentra en el *Expediente de Proyecto.*

1.4. Base de Datos y sus elementos

La base de datos es un sistema formado por un conjunto de datos almacenados y un conjunto de programas que manipulen ese conjunto de datos. Estas facilitan el almacenamiento de grandes cantidades de información; permiten la recuperación rápida y flexible de los datos y se puede organizar y reorganizar la información, así como imprimirla.

Las Bases de Datos (BD) están definidas como: “Conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una BD puede considerarse una colección de datos variables en el tiempo” [12].

Una base de datos está formada por los diferentes elementos:

Datos: Se almacenan de acuerdo a la estructura externa y van a ser procesados para convertirse en información. Deben ser integrados posibilitando así que en la unión de los archivos que forman el sistema no exista redundancia de datos y compartidos donde varios usuarios pueden acceder a los mismos datos con fines diferentes.

Hardware: Es el soporte físico que permite almacenar la información de la base de datos, así como los dispositivos periféricos, por ejemplo, unidad de control y canales de comunicación.

Software: Es el que permite trabajar y gestionar la BD de la forma más eficiente. El Sistema Gestor de Bases de Datos (SGBD) es el encargado de gestionar la BD, por lo tanto todas las operaciones que se realicen sobre las mismas han de pasar por el SGBD.

Usuarios: Una base de datos típica conlleva a la existencia de tres tipos de usuario con relación a su diseño, desarrollo y uso:

- El administrador de bases de datos: Es el usuario más importante, pues se encarga de diseñar y modificar la estructura de la BD.
- El desarrollador de aplicaciones (programador): Se encarga de diseñar y programar las aplicaciones necesarias para la utilización de la BD, realizando las peticiones pertinentes al SGBD.

Los usuarios finales: Son las personas que se dedican a trabajar sobre los datos almacenados en la BD. Consultan y editan dichos datos mediante un lenguaje de consulta de alto nivel.

1.4.1. Modelos de Bases de Datos

Un modelo de datos es una descripción de algo conocido donde se guarda información, así como de los métodos para almacenar y recuperar información de donde es guardada. Los modelos de datos son abstracciones que permiten la implementación de un sistema eficiente de base de datos, por lo general se refieren a algoritmos y conceptos matemáticos. Algunos modelos con frecuencia utilizados en las bases de datos son los que a continuación se mencionan:

Modelo Relacional

Uno de los modelos matemáticos más importantes y actuales para la representación de las bases de datos, es el enfoque relacional. Se basa en la teoría matemática de las relaciones, las cuales podrían considerarse en forma lógica como conjuntos de datos llamados tuplas.

En el modelo relacional, tanto los objetos o entidades, como las relaciones que se establecen entre ellos, se representan a través de tablas, que no son más que las llamadas relaciones. Cada relación está compuesta de filas y es a lo que se denomina tuplas. También la relación está compuesta por columnas donde los atributos o campos toman valores en sus respectivos dominios.
Las bases de datos relacionales están estructuradas por una o más tablas relacionadas que contienen la información de una forma organizada. Generalmente contendrán muchas tablas y una tabla sólo contiene un número fijo de campos, en dependencia de la información almacenada. El orden de los campos de una tabla no está determinado y para cada campo existe un conjunto de valores posible.

Algunas de sus principales características son:

- Puede ser entendido y usado por cualquier usuario.
- Permite ampliar el esquema conceptual sin modificar las aplicaciones de gestión.
- Los usuarios no necesitan saber donde se encuentran los datos físicamente.

Modelo documental
Permiten la indexación a texto completo, y en líneas generales realizan búsquedas más potentes.

Modelo jerárquico

Este modelo utiliza árboles para la representación lógica de los datos. Este árbol está compuesto por elementos llamados nodos. Donde un nodo padre puede tener varios hijos y el nodo que no contenga un padre es llamado raíz, siendo este el nivel más alto del árbol. La representación gráfica de este modelo se realiza mediante la creación de un árbol invertido, los diferentes niveles quedan unidos mediante relaciones.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento. Puede representar dos tipos de relaciones entre los datos: relaciones de uno a uno y relaciones de uno a muchos. Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos, por lo que actualmente no es muy utilizado.

En este modelo se pueden mencionar algunos inconvenientes:

- Un segmento hijo no puede tener más de un padre.
- No se permiten más de una relación entre dos segmentos.
- Para acceder a cualquier segmento es necesario comenzar por el segmento raíz

- El árbol se debe de recorrer en el orden designado.

Modelo de red

En este modelo las entidades se representan como nodos y sus relaciones son las líneas que los unen. En esta estructura cualquier componente puede relacionarse con cualquier otro. A diferencia del modelo jerárquico, en este modelo, un hijo puede tener varios padres. Este modelo ofrece una solución eficiente al problema de redundancia de datos.

Modelo Orientado a Objetos

Este modelo trata de almacenar en la base de datos el estado y comportamiento de los objetos completos. En la base de datos se incorpora todos los conceptos importantes del paradigma de objetos:

- Encapsulación - Permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- Herencia - Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- Polimorfismo - Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación se especifica en dos partes: la interfaz de una operación que incluye el nombre de la operación y los tipos de datos de sus argumentos o parámetros y la implementación o método de la operación que se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado.

Modelo Objeto-Relacional

Este modelo permite que los desarrolladores ahorren tiempo pues no tienen que escribir el código necesario que comunique las tablas con los objetos pues este ya viene integrado. Surge como una respuesta a la incorporación de la tecnología de objetos en las bases de datos relacionales permitiendo el tratamiento de datos y relaciones complejas. Es compatible con la tecnología relacional y tiene un mejor soporte para aplicaciones voluminosas. Se combinan las nociones de extensibilidad,

orientación por objetos, y en algunos casos hasta tablas anidadas. Soporta datos y consultas complejas.

Modelo deductivo

Permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. También las bases de datos deductivas son llamadas base de datos lógica, a raíz de que se basan en lógica matemática.

El modelo de base de datos seleccionado es el Relacional, debido a que es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente, garantizando un alto grado de independencia de los datos. Además, brinda una serie de herramientas para evitar la duplicidad de registros, a través de campos claves o llaves; garantiza la integridad referencial, así al eliminar un registro elimina todos los registros relacionados dependientes y favorece la normalización de la BD. Los lenguajes matemáticos sobre los que se asienta este modelo, el álgebra y el cálculo relacionales, aportan un sistema de acceso y consultas orientado al conjunto.

1.4.2. Sistemas Gestores de Base de Datos

"El software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina **sistema de gestión de bases de datos** (SGBD)" [12].

Un SGBD permite crear y mantener una Base de datos, asegurando su integridad, confidencialidad y seguridad. Poseen lenguajes de definición de datos (DDL), lenguajes de manipulación de datos (DML) y lenguajes de consulta (SQL) que facilitan el proceso de diseño de aplicaciones y que los tratamientos sean más eficientes y rápidos, dando la mayor flexibilidad posible a los usuarios. Esto se le da cumplimiento a partir de los objetivos fundamentales de los sistemas de bases datos, de los que se puede mencionar: Independencia de los datos y los programas de aplicación, Minimización de la redundancia, Integración y sincronización de las bases de datos, Integridad de los datos, Seguridad y recuperación, Facilidad de manipulación de la información y Control centralizado.

Algunos de los productos que se encuentran disponibles en el mercado son:

SGBD libres

- PostgreSQL

- MySQL
- Firebird
- DB2 Express-C
- Apache Derby

SGBD no libres

- Advantage Database
- Fox Pro
- IBM DB2
- IBM Informix
- MAGIC
- Microsoft SQL Server
- Oracle

En este trabajo será caracterizado el SGBD libre PostgreSQL definido por el proyecto.

1.4.2.1. PostgreSQL

PostgreSQL es un servidor de base de datos relacional orientada a objetos de software libre. Está considerado como la base de datos de código abierto más avanzada del mundo. PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son las consultas SQL declarativas, el control de concurrencia multi-versión, el soporte de multi-usuario, transacciones, optimización de consultas, herencia, y arreglos.

PostgreSQL es capaz de competir con muchos gestores comerciales, aunque carezca de alguna característica casi imprescindible. La velocidad de respuesta que ofrece este gestor con bases de datos relativamente pequeñas puede parecer un poco deficiente, aunque esta misma velocidad la mantiene al gestionar bases de datos realmente grandes, cosa que resulta loable.

Algunas de las características que posee este gestor son:

- Gran escalabilidad. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de

peticiones simultáneas de manera correcta (ha llegado a soportar el triple de carga de lo que soporta MySQL).

- Implementa el uso de rollback's, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que MySQL no podría.
- Tiene la capacidad de comprobar la integridad referencial, así la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser Oracle.

Algunos de los inconvenientes que se pueden mencionar de este gestor son:

- El primer encuentro con este gestor puede parecer un poco engorroso ya que la sintaxis de algunos de sus comandos no es nada intuitiva.
- Resultan engorrosas las pequeñas variaciones que presenta este gestor en algunos de los tipos de datos que maneja.
- Consume gran cantidad de recursos.
- Es de 2 a 3 veces más lento que MySQL.

Para el desarrollo del trabajo se utilizará PostgreSQL no solo por sus ventajas y características positivas sino que fue la tecnología definida por el proyecto. PostgreSQL en su versión 8.3 (última versión lanzada por el Grupo Global de Desarrollo de PostgreSQL) tiene mejoras muy importantes de las que se pueden mencionar:

- Integración de la tecnología HOT (Heap Organized Tuples), mejora de forma crítica el rendimiento de las bases de datos que se actualizan con una frecuencia muy alta. “Reduce significativamente los problemas de mantención que tienen que ver con datos frecuentemente actualizados, disminuyendo la necesidad de hacer VACUUM y otorgando mejoras importantes de rendimiento para algunas aplicaciones.”[13].
- Commits asíncronos, permite que COMMIT retorne el control a la aplicación sin esperar la escritura física en disco, esto permite mejores tiempos de respuesta.
- “Soporte a MS Visual C++, permite que el código fuente de PostgreSQL sea compilado con el compilador C++ de Microsoft, en lugar de las herramientas MinGW. Esto mejora el rendimiento y la estabilidad en plataformas MS.”[13].
- Soporte SQL/XML. El nuevo tipo de dato XML soporta completamente la especificación SQL/XML de ANSI SQL: 2003, incluyendo verificaciones de sintaxis, chequeos de tipos de

datos, publicación SQL/XML y consultas XPath. La versión 8.3 también incluye funciones adicionales para exportar datos en formato XML.

1.5. Servidor Web

Con la realización del presente trabajo se obtendrá una aplicación Web que estará soportada por tecnología Cliente-Servidor por lo que será necesario un servidor para la Web.

Un servidor web es un programa que sirve para atender y responder a las diferentes peticiones de los navegadores, proporcionando los recursos que soliciten usando el protocolo HTTP o el protocolo HTTPS. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición. Algunos de estos servidores son Cherokee, IIS, Apache, Java Server.

El servidor web empleado debe ser compatible con el sistema operativo Windows, ser potente y a la misma vez debe ser capaz de soportar varias conexiones, ya que en la Dirección de Calidad del CIGB existen diferentes laboratorios con más de una PC cada uno. La dirección del proyecto definió un servidor que cumple con las características que se necesitan, este servidor es Apache.

1.5.1. Apache en su versión 2.2.6

El servidor HTTP Apache es un software libre de código abierto para plataformas Unix BSD, GNU/Linux, Windows y Macintosh, que implementa el protocolo HHTP. Este se desarrolla dentro del proyecto HTTP Server (httpd) de la compañía Apache Software Foundation, y fueron Brian Behlendorf y Cliff Skolnick quienes a través de una lista de correo coordinaron el trabajo y lograron establecer un espacio compartido de libre acceso para los desarrolladores.

Las sucesivas evoluciones y mejoras alcanzaron una gran implantación como software de servidor inicialmente solo para sistemas operativos UNIX y fruto de esa evolución es la versión para Windows.

Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, aunque no presenta una interfaz gráfica que ayude en su configuración es el servidor HTTP más usado.

La aceptación y popularidad de este servidor web se debe a sus características de las cuales se hacen mención a continuación [14]:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si queremos ver que es lo que estamos instalando como servidor, lo podemos saber, sin ningún secreto, sin ninguna puerta trasera ;).
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que los instalemos cuando los necesitemos. Otra cosa importante es que cualquiera que posea una experiencia decente en la programación de C o Perl puede escribir un modulo para realizar una función determinada.
- Apache trabaja con gran cantidad de Perl, PHP y otros lenguajes de script. Perl destaca en el mundo del script y Apache utiliza su parte del pastel de Perl tanto con soporte CGI como con soporte mod perl. También trabaja con Java y páginas jsp. Teniendo todo el soporte que se necesita para tener páginas dinámicas.
- Apache te permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Tiene una alta configurabilidad en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en tu servidor.

1.6. Editor web orientado a la programación de páginas PHP

Después de definido el servidor web a utilizar se necesita de un editor, de un entorno de desarrollo para la implementación de una aplicación web que facilite el completamiento de código, una herramienta que agilice el trabajo del programador en todo momento.

Un entorno de desarrollo integrado o IDE (Integrated Development Environment) es un programa compuesto por un conjunto de herramientas para un programador. Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación.

Un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación.

Algunos de estos IDEs son Eclipse, JDeveloper de Oracle_Coporation, JBuilder de Borland y NetBeans.

En el desarrollo del módulo Estudios de Estabilidad y Materiales de Referencia se va a utilizar como IDE de desarrollo el eclipse, con el plugin PDT (PHP Development Tool) que soporta el lenguaje PHP5.

1.6.1.1. Eclipse

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge, con el objetivo fundamental de crear una plataforma de desarrollo modular que cualquier herramienta de desarrollo pueda usar con el lenguaje de programación que se desee. Ahora es desarrollado por la Fundación Eclipse, una organización independiente que fomenta una comunidad de código abierto.

Eclipse es una plataforma universal para integrar herramientas de desarrollo pues emplea una estructura abierta de plug-ins que permite expandir las capacidades de la plataforma base hasta el infinito; pudiendo ser añadidos automáticamente al entorno de desarrollo, lo que lo convierte en uno muy adecuado para el desarrollo de software.

Es impulsor de la tecnología de servidores PHP, ofrece el control del editor de código, del compilador y del depurador desde una única interfaz de usuario. Evita tareas repetitivas, facilita la escritura de código correcto, disminuye el tiempo de depuración e incrementa la productividad del desarrollador. Estas tareas pueden realizarse de varias formas diferentes, en algunos casos mediante la inclusión de asistentes para las tareas más habituales y mecánicas de editores que completen automáticamente el código y señalen los errores sintácticos, de gestores de archivos fuente.

1.7. Lenguajes de Programación para la Web

Los lenguajes de programación son herramientas que permiten crear programas y software, facilitan la tarea de programación, ya que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar.

En la programación Web, el HTML (Hypertext Markup Language) es el lenguaje que permite codificar o preparar documentos de hipertexto, que viene a ser el lenguaje común para la construcción de una página Web.

Los lenguajes se clasifican en dos partes fundamentales que reconocen la propia arquitectura Cliente/Servidor de esta plataforma de desarrollo: los lenguajes del lado del Servidor y los lenguajes del lado del Cliente.

Esta distinción en los lenguajes ha sido necesaria debido a que la Web funciona en modo "Desconectado", o sea, un usuario a través de un navegador hace una petición de una página Web a un Servidor Web (Request), el Servidor recibe la petición, la procesa y le envía la respuesta al Cliente (Response), el que se desconecta al recibirla.

Del lado del cliente se encuentran principalmente el lenguaje JavaScript (JScript), que es el encargado de aportar dinamismo a la aplicación en los navegadores y además tienen la responsabilidad de ejercer funciones específicas como la validación y la impresión. JavaScript es un lenguaje de programación muy utilizado por todos los programadores para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Con JavaScript se puede lograr obtener efectos especiales en las páginas para todos los gustos y definir infinidad de interactividades con el usuario. El navegador que utilice el cliente es muy importante, pues será el encargado de interpretar todas las instrucciones JavaScript programadas y ejecutarlas.

HTML es un lenguaje básico para los programadores que comienzan a incursionar en el campo del desarrollo web, es por eso que después del HTML, JavaScript es considerado el nivel superior o el siguiente paso, que puede dar un programador de la web con vistas a decidirse por perfeccionar el estilo de sus páginas y la potencia de sus proyectos con el propósito de obtener un mayor alcance.

En el dominio de la red, los lenguajes del lado del servidor más ampliamente utilizados para el desarrollo de páginas dinámicas son el ASP, PHP, PERL y JAVA.

Para el desarrollo de esta aplicación, la dirección del proyecto decidió utilizar PHP5 por ser uno de los lenguajes de programación del lado del servidor que mayor auge ha alcanzado en la actualidad y por lo eficiente que resultan las aplicaciones desarrolladas con su metodología. Del lado del cliente se va a utilizar JavaScript para lograr obtener efectos especiales en las páginas y HTML para preparar documentos de hipertexto.

1.7.1. Qué es PHP

PHP (Hypertext Pre-processor) es un lenguaje de programación interpretado, es decir que no requiere compilación, creado en 1994 por Rasmus Lerdof, y publicado bajo la PHP License, considerada como software libre. Es utilizado habitualmente normalmente para la creación de páginas web dinámicas y aplicaciones para servidores. Permite conexión con todo tipo de bases de datos como MySQL, Postgre SQL, Oracle, DB2, Microsoft SQL Server, Firebird y SQLite.

Puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Es también el módulo Apache más popular entre las computadoras que utilizan Apache como servidor web. Corre sobre siete plataformas, funciona en 11 tipos de servidores, ofrece soporte para varios Sistemas Gestores de Bases de Datos (SGBD) y contiene unas 40 extensiones estables. La primera versión salida al mercado fue PHP3. Actualmente se usa PHP5 y específicamente en este trabajo se utiliza PHP 5 en su versión 5.2.5. La versión más reciente de PHP es la 5.2.9-1 para servidores con sistemas operativos Windows, lanzado en Marzo de 2009.

PHP presenta numerosas ventajas para la creación de páginas web dinámicas y aplicaciones para servidores, de las cuales a continuación se hace mención de algunas de ellas.

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos llamados extensiones.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluída.
- No requiere definición de tipos de variables.
- Existen muchos programadores entusiastas que escriben aplicaciones en PHP y las distribuyen libremente. Quizás alguna de éstas se acomode a lo que se desea realizar, y se dispone de más tiempo para programar otras cosas.

- PHP es suficientemente versátil y potente como para hacer tanto aplicaciones complejas que necesiten acceder a recursos de bajo nivel del sistema como pequeños scripts que envíen por correo electrónico un formulario llenado por un cliente.
- Tiene manejo de excepciones (desde PHP5).

Algunos de los IDEs para PHP más conocidos son:

- Zend Studio.
- PDT, con plugin de Eclipse.
- Komodo IDE, libre y gratuito, el IDE es licencia comercial.
- NuSphere PhpED, para Linux y windows.
- Quanta, gratuito, para GNU/Linux con QT.
- Bluefish, GPL y gratuito, para GNU/Linux con GTK.
- NetBeans, libre, para Linux y Windows.

Siendo este último IDE el que se usará para la implementación de la aplicación web del módulo EEMR como se explicó en epígrafes anteriores.

Algunos de los Frameworks que pueden ser mencionados en PHP son Zend Framework, PHP Prado, CakePHP, Qcodo, Kumbia y Symfony, siendo este último el definido por el proyecto para el desarrollo del mismo, más adelante se caracteriza el framework.

1.7.1.1. PHP 5

PHP 5, utiliza el motor Zend Engine II o Zend Engine 2, el cual provee mejor soporte para la Programación Orientada a Objetos, que en versiones anteriores era extremadamente rudimentario, con PHP Data Objects. Otras de las ventajas que se pueden mencionar de PHP 5 gracias a la utilización del motor Zend Engine 2 son:

- Mejoras de rendimiento, siendo posible un mejor aprovechamiento de la memoria.
- Presenta más seguridad a través de los filtros, que permiten a los programadores leer los campos de formularios de forma segura.
- Excepciones de errores.

- Simplicidad. Su sintaxis está inspirada en C, ligeramente modificada para adaptarla al entorno en el que trabaja, de modo que si se está familiarizado con esta sintaxis, le resultará muy fácil aprender PHP.
- El nuevo modelo orientado a objetos de PHP5 trae una sintaxis muy parecida a la del lenguaje Java. Si en algún momento se ha visto éste lenguaje, parecerá familiar.
- Existen un gran número de desarrolladores y colaboradores, que mantienen las actualizaciones del PHP, cualquier error que hubiese es rápidamente corregido. El código es constantemente revisado y los cambios son publicados en su página Web.
- Soporte a diferentes motores de bases de datos.

Específicamente se utiliza la versión 5.2.5 de PHP, la cual está centrada en mejorar la estabilidad con cerca de 60 errores solucionados, algunos de ellos relacionados con la seguridad.

1.8. Los Framework

Un framework puede ser traducido como marco de trabajo pero en realidad es más que eso. "Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. (…) un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. (…), un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas" [15].

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes ya que en un momento determinado puede que el programador necesite utilizar cierta funcionalidad para su aplicación y oportunamente la tendrá a su disposición con solo llamarla. Proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener, facilitan la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. Pueden incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Son capaces de representar una arquitectura de software que modele las relaciones generales de las entidades del dominio, y llevan consigo una estructura y una metodología de trabajo la cual extienden o utilizan las aplicaciones del dominio.

"Un framework permite separar en capas las aplicaciones, las más utilizadas son las aplicaciones en tres capas ya que:" [16]

- La lógica de presentación administra las interacciones entre el usuario y el software.
- La lógica de datos permite el acceso a un agente de almacenamiento persistente u otros.
- La lógica de dominio o de negocio, manipula los modelos de datos de acuerdo a los comandos recibidos desde la presentación.

La dirección del proyecto productivo en conjunto con los implementadores, determinó que para el desarrollo de los distintos módulos, la opción más apropiada sería utilizar Symfony, ya que se necesitan implementar un gran número de módulos y el proyecto en general es intenso, cada módulo tiene una numerosa cantidad de CU. Además se pueden desarrollar aplicaciones de forma muy rápida, con el uso de PHP, mediante las ventajas que ofrece Symfony. Acerca de él se puede encontrar abundante bibliografía en internet, muy actualizada, y cuenta con una creciente comunidad de desarrolladores que discuten a diario varias alternativas de solución a través de los fórum de discusiones en línea.

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

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

A continuación se muestran algunas de sus características citadas textualmente de la guía definitiva de Symfony de Fabien Potencier y Francois Zaninotto.

1.8.1. Características de Symfony

Symfony se diseñó para que se ajustara a los siguientes requisitos:

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

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

El equipo de trabajo encargado de implementar el módulo EEMR determinó que para el desarrollo del mismo, sería mejor utilizar la versión 1.2.2 de Symfony lo cual se exponen a lo largo del siguiente epígrafe.

1.8.2. Symfony en su versión 1.2.2

Esta nueva versión ha permitido numerosas mejoras del framework. Es una de las últimas versiones ya que posterior a ella lanzaron la versión 1.2.4. A continuación se describen algunas de las ventajas de Symfony 1.2.2.

Symfony 1.2 es compatible con PHP 5.2.4 o posterior, mientras que Symfony 1.1 funciona con PHP 5.1 y Symfony 1.0 con PHP 5.0. Symfony 1.2 explota todas las funcionalidades que le brinda PHP.

Symfony ha creado unos validadores que encapsulan todo el código PHP necesario para realizarlos. Un validador es una clase que proporciona un método llamado execute(). El método requiere de un parámetro que es el valor del campo de formulario y devuelve true si el valor es válido y false en otro caso. Un ejemplo de estos validadores es el nuevo sfValidatorDateRange para validar una serie de fechas lo que permite una mejor manipulación de la misma.

Ahora puede simular solicitudes PUT y DELETE desde un navegador utilizando el método POST y la adición de un parámetro especial sf_method. El form_tag () ha sido actualizado para generar automáticamente la etiqueta oculta de diferentes métodos de GET o POST.

Una de las principales limitaciones en las formas de impulsar symfony 1.1 fue la incapacidad para salvar automáticamente objetos de formas embebidas. Esto se ha aplicado en Symfony 1.2, lo que significa que al guardar un Propel form, Symfony guardará automáticamente el objeto principal y todos los objetos relacionados con las formas embebidas.

Symfony 1.2 viene con nuevos widgets que incluyen las clases que muestra el código HTML de los campos del formulario (<input>, <textarea>, <select>). Algunos de estos widgets son:

- sfWidgetFormDateRange
- sfWidgetFormFilterInput
- sfWidgetFormFilterDate
- sfWidgetFormI18nSelectCurrency
- sfWidgetFormSelectCheckbox

Antes de Symfony 1.2, todos los plugins instalados en el marco del directorio se cargaban automáticamente. A partir de Symfony 1.2, se necesita habilitar los plugins que desea utilizar en sus proyectos.

Symfony es ahora capaz de generar los filtros basados en el modelo gracias a propel:build-filters. Symfony 1.2 agrega el método sfToolkit::addIncludePath() para gestionar fácilmente la configuración de la directiva del archivo de configuración include_path de PHP, perteneciente este a las clases internas del Propel.

El Propel se utiliza para el mapeo de objetos a bases de datos conocido como ORM (por sus siglas en inglés *"object-relational mapping"*) a través de los archivos de configuración `schema.yml` y `propel.ini` los cuales se utilizan para que las librerías de Propel puedan interactuar con las clases de Symfony y con los datos de la aplicación. El Propel es una de las mejores capas de abstracción de objetos/relacional disponibles en PHP 5. Proporciona persistencia para los objetos y un servicio de consultas.

Conclusiones

En el presente capitulo se presentaron las compañías más reconocidas a nivel mundial en la producción de LIMS. El desarrollo del módulo EEMR estará basado en RUP como metodología para el proceso de desarrollo de Software. Se abordó sobre el rol, los artefactos y las actividades a realizar en los flujos de trabajo Requerimientos, Diseño e Implementación.

Al finalizar el capítulo se arriba a las siguientes conclusiones:

- Se utiliza UML como lenguaje de modelado.
- La herramienta CASE a utilizar es Visual Paradigm for UML en su versión 6.1, para modelar los diferentes diagramas generados en cada uno de los flujos de trabajo que define RUP.
- La herramienta para la gestión de requerimientos utilizada es Rational RequisitePro.
- Se utiliza el modelo de base de datos relacional.
- Utilizar PostgreSQL como gestor de base de datos, empleando para el desarrollo y administración de la Base de Datos: EMS SQL Manager for PostgreSQL.
- Como servidor web Apache en su versión 2.2.6
- El entorno de desarrollo en que se implementara el módulo es Eclipse y el lenguaje de programación para la web es PHP 5.2.5.
- Se utiliza el framework Symfony en su versión 1.2.2 para la implementación de la capa de acceso a datos.

Capítulo 2
Características del Sistema

Introducción

En el presente capítulo se definen las necesidades del cliente con respecto a lo que debe realizar el sistema mostrándose como requisitos funcionales y no funcionales, ayudando a los desarrolladores a un mejor entendimiento de los requerimientos del sistema. Se describen los actores y los casos de uso del sistema definidos. Se muestran las descripciones textuales resumidas de los casos de uso del sistema definidos hasta el momento y solo una descripción textual en su forma ampliada que será la que se tomará de ejemplo para ilustrar los artefactos del flujo de trabajo de Diseño. En el *Expediente de Proyecto* serán presentadas las descripciones textuales en su forma ampliada, así como la matriz de trazabilidad entre requisitos funcionales y casos de uso del sistema.

2.1. Propósito de Requerimientos

El propósito del flujo de trabajo de Requerimientos es guiar el desarrollo del sistema. Esto se consigue mediante una descripción detallada de los requisitos del sistema que permite llegar a un acuerdo entre el cliente y los desarrolladores sobre qué debe hacer el sistema.

2.1.1. Técnica de captura de requisitos

La captura de requisitos es la actividad mediante la cual el equipo de desarrollo de un sistema de software extrae de cualquier fuente la información disponible y las necesidades que debe cumplir dicho sistema.

Los requisitos deben ser elaborados conjuntamente entre los usuarios quienes aportan el conocimiento de negocio y los desarrolladores quienes conocen la viabilidad de un desarrollo y sus plazos. El liderazgo de la captura de requisitos debe ser asumido por el usuario, ya que las necesidades surgen del usuario en un momento dado.

La redacción de estos requisitos debe ser lo suficientemente clara para permitir su implementación.

Se deben recoger los requisitos funcionales que el sistema de información debe cubrir. Además, se deben identificar los requisitos no funcionales del sistema, es decir, las facilidades que ha de proporcionar el sistema, y las restricciones a que estará sometido.

La técnica que se utiliza en la captura de requisitos es la entrevista, con la técnica complementaria estudio de documentación.

Las entrevistas no deben improvisarse, por lo que conviene realizar las tareas previas:
Estudiar el dominio del problema: Para conocer el dominio del problema se puede recurrir a técnicas de estudio de documentación, a bibliografía sobre el tema y documentación de proyectos similares realizados anteriormente.
Seleccionar a las personas a las que se va a entrevistar: se debe minimizar el número de entrevistas a realizar.
Determinar el objetivo y contenido de las entrevistas: para minimizar el tiempo de la entrevista es fundamental fijar el objetivo que se pretende alcanzar y determinar previamente su contenido.
Planificar las entrevistas: la fecha, hora, lugar y duración de las entrevista deben fijarse teniendo en cuenta siempre la agenda del entrevistado.

Una adecuada captura de requisitos es necesaria pues incide en una mayor calidad de los productos entregados, ya que la mayoría de los proyectos que fracasan tienen su origen en una mala captura de requisitos. Además la participación del usuario en la captura de requisitos garantiza que el proyecto se adecua a sus necesidades.

2.1.2. Técnica de validación

La validación de requerimientos comprende un bosquejo completo del documento en lugar de requerimientos incompletos. Esta permite demostrar que los requerimientos definidos en el sistema son los que realmente quiere el cliente, además de revisar que no se haya omitido ninguno, que no sean ambiguos, inconsistentes o redundantes. Es muy importante la validación pues los errores en los requerimientos pueden conducir a costos excesivos si se descubren durante el desarrollo o después de la implantación, además, es difícil demostrar que un conjunto de requerimientos cumple con las necesidades del usuario.

Existen diferentes técnicas de validación entre los que se encuentran, las revisiones de requerimientos, la construcción de prototipos, la generación de casos de prueba y el análisis de consistencia automático. La técnica a realizar es la de construcción de prototipos no funcionales.

Para la validación de los requisitos a través de los Prototipos no Funcionales, se permite crear un modelo del software que debe ser construido, en la medida que esto se vaya realizando le serán mostrados al cliente con el objetivo de verificar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Lo que permitirá asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos.

2.1.3. Técnica de verificación de requisitos

La verificación de requerimientos es el proceso de evaluar un determinado segmento de un producto con respecto a las especificaciones proporcionadas a la entrada del mismo" [17], todo esto para asegurar su exactitud y consistencia.

Una vez que comience la implementación del sistema el analista se reunirá con el desarrollador cada 15 días para comprobar que la implementación que se esté realizando cumpla con todos los requisitos aprobados por el cliente.

Para realizar las verificaciones de la aplicación de un software se deben tomar en cuenta los siguientes aspectos:

- Declarar las especificaciones de los requerimientos en cada segmento de prueba del software.
- Que el software opere de acuerdo a lo descrito en los requerimientos.
- Verificar los resultados de operación en cada etapa.

La validación es terminada cuando los requerimientos del software cumplen con cada etapa de la verificación. Cualquier etapa que sea verificada y no cumpla con los requerimientos, en el sistema no puede ser validado. Cada etapa de verificación implica un proceso de validación.

2.1.4. Requerimientos Funcionales y no Funcionales

Los requerimientos ayudan a revisar que el proceso de recogida sea completo, su nivel más alto de clasificación es la que se realiza en: requisitos funcionales y requisitos no funcionales.

2.1.5. Requisitos Funcionales

Los requerimientos funcionales son condiciones o capacidades que debe alcanzar un sistema para resolver un problema o lograr un objetivo. Los requerimientos funcionales que se identificaron en el módulo EEMR son:

R1. Gestionar Resultado de EE(SIC-0894)
- R1.1. Crear un nuevo Resultado de EE.
- R1.2. Modificar Resultado de EE.
  - R1.2.1. Modificar datos de un Resultado de EE.
  - R1.2.2. Registrar Trazas.
- R1.3. Buscar el Resultado de EE.
- R1.4. Visualizar el Resultado de EE.
- R1.5. Imprimir el Resultado de EE.

R2. Gestionar Programa de EE (SIC-0892).
- R2.1. Crear un nuevo Programa de EE.
- R2.2. Buscar el Programa de EE.
- R2.3. Modificar Programa de EE.
  - R2.3.1. Modificar datos del Programa de EE.
  - R2.3.2. Registrar traza.
  - R2.3.3. Modificar Cronogramas de EE asociados al Programa de EE.
- R2.4. Visualizar el Programa de EE.
- R2.5. Imprimir el Programa de EE.

R3. Gestionar Recordatorio de solicitud de material.
- R3.1. Crear un Recordatorio de solicitud de material.
- R3.2. Buscar un Recordatorio de solicitud de material.
- R3.3. Modificar Recordatorio de solicitud.
  - R3.3.1. Modificar un Recordatorio de solicitud de material.
  - R3.3.2. Registrar traza.
- R3.4. Visualizar un Recordatorio de solicitud de material.
- R3.5. Imprimir un Recordatorio de solicitud de material.

R4. Gestionar Solicitud de producto (SDF-01S).

R4.1. Crear una nueva Solicitud de producto.

R4.2. Buscar Solicitud de producto.

R4.3. Modificar Solicitud de producto.

R4.3.1. Modificar datos de la Solicitud de producto.

R4.3.2. Registrar traza.

R4.4. Visualizar Solicitud de producto.

R4.5. Imprimir la Solicitud de producto.

R5. Gestionar Cronograma de EE (SIC-0893).

R5.1. Crear un Cronograma de EE.

R5.2. Buscar el Cronograma de EE.

R5.3. Modificar Cronograma de EE.

R5.3.1. Modificar datos del Cronograma de EE.

R5.3.2. Registrar traza.

R5.4. Visualizar el Cronograma de EE.

R5.5. Imprimir el Cronograma de EE.

R6. Gestionar Registro de Descargo de MR (SIC-0827).

R6.1. Crear un nuevo Registro de Descargo de MR.

R6.2. Registrar una entrega en el Registro de Descargo de MR.

R6.3. Buscar el Registro de Descargo de MR.

R6.4. Modificar Registro de Descargo de MR.

R6.4.1. Modificar datos del Registro de Descargo de MR.

R6.4.2. Registrar traza.

R6.5. Visualizar el Registro de Descargo de MR.

R6.6. Imprimir el Registro de Descargo de MR.

R6.7. Realizar reporte del Registro Descargo de MR.

R6.7.1. Generar reporte.

R6.7.2. Imprimir reporte.

R7. Gestionar Registro de Salida de documentos de EE.

R7.1. Registrar datos en el Registro de Salida de documentos de EE.

R7.2. Buscar Registro de Salida de documentos de EE.

R7.3. Modificar Registro de Salida de documentos de EE.

R7.3.1. Modificar datos del Registro de Salida de documentos de EE.

R7.3.2. Registrar traza.

R7.4. Visualizar Registro de Salida de documentos de EE.

R7.5. Imprimir Registro de Salida de documentos de EE.

R7.6. Realizar reporte del Registro de Salida de documentos de EE.

R7.6.1. Generar reporte.

R7.6.2. Imprimir reporte.

R8. Gestionar Solicitud de EE (SIC-0882).

R8.1. Crear una nueva solicitud de EE.

R8.2. Buscar solicitud indicada

R8.3. Modificar Solicitud de EE.

R8.3.1. Modificar una solicitud de EE.

R8.3.2. Registrar traza.

R8.4. Visualizar solicitud indicada.

R8.5. Imprimir solicitud de EE.

R9. Gestionar Diseño de MR.

R9.1. Crear un nuevo Diseño de MR.

R9.2. Buscar el Diseño de MR.

R9.3. Modificar Diseño de MR.

R9.3.1. Modificar datos del Diseño de MR.

R9.3.2. Registrar traza.

R9.4. Visualizar el Diseño de MR.

R9.5. Imprimir el Diseño de MR.

R10. Gestionar Diseño Experimental de Estudio de Homogeneidad del MR.

R10.1. Crear un nuevo Diseño Experimental de Homogeneidad.

R10.2. Buscar el Diseño Experimental de Homogeneidad.

R10.3. Modificar Diseño Experimental de Homogeneidad.

R10.3.1. Modificar datos del Diseño Experimental de Homogeneidad.

R10.3.2. Registrar traza.

R10.4. Visualizar el Diseño Experimental de Homogeneidad.

R10.5. Imprimir el Diseño Experimental de Homogeneidad.

R11. Gestionar Diseño Experimental de Estudio de Caracterización del MR.

R11.1. Crear un nuevo Diseño Experimental de Estudio de Caracterización.

R11.2. Buscar el Diseño Experimental de Estudio de Caracterización.

R11.3. Modificar Diseño Experimental de Estudio de Caracterización.

R11.3.1. Modificar datos del Diseño Experimental de Estudio de Caracterización.

R11.3.2. Registrar traza.

R11.4. Visualizar el Diseño Experimental de Estudio de Caracterización.

R11.5. Imprimir el Diseño Experimental de Estudio de Caracterización.

R12. Gestionar Certificado del MR (SIC-0232).

R12.1. Crear un nuevo Certificado del MR.

R12.2. Buscar el Certificado del MR.

R12.3. Modificar Certificado del MR.

R12.3.1. Modificar datos del Certificado del MR.

R12.3.2. Registrar traza.

R12.4. Visualizar el Certificado del MR.

R12.5. Imprimir el Certificado del MR.

R13. Gestionar Solicitud de Destrucción de Productos (SIC-0830).

R13.1. Crear una nueva Solicitud de Destrucción.

R13.2. Buscar la solicitud de Destrucción.

R13.3. Modificar Solicitud de Destrucción.

R13.3.1. Modificar datos de la Solicitud de Destrucción.

R13.3.2. Registrar traza.

R13.4. Visualizar la Solicitud de Destrucción.

R13.5. Imprimir la Solicitud de Destrucción

R13.6. Realizar reporte de la Solicitud de Destrucción.

R13.6.1. Generar reporte.

R13.6.2. Imprimir el reporte.

R14. Gestionar Registro de Control de Copias de los Certificados de MR (SIC-0861).

R14.1. Crear un nuevo Registro de Control.

R14.2. Registrar datos en el Registro de Control.

R14.3. Buscar el Registro de Control.

R14.4. Modificar datos del Registro de Control.

R14.4.1. Modificar datos del Registro de Control.

R14.4.2. Registrar traza.

R14.5. Visualizar el Registro de Control.

R14.6. Imprimir el Registro de Control.

R14.7. Realizar reporte del Registro de Control.

R14.7.1. Generar reporte.

R14.7.2. Imprimir reporte.

R15. Gestionar Expediente de MR.

R15.1. Crear un nuevo Expediente de MR.

R15.2. Buscar el Expediente de MR.

R15.3. Modificar Expediente de MR.

R15.3.1. Modificar el Expediente de MR.

R15.3.2. Registrar traza.

R15.4. Visualizar el Expediente de MR.

R15.5. Imprimir Expediente de MR.

R16. Gestionar Libro de Salida de Muestras.

R16.1. Crear un nuevo libro de salida de muestras.

R16.2. Registrar datos en el libro de salida de muestras.

R16.3. Buscar el libro de salida de muestras.

R16.4. Modificar libro de salida de muestras.

R16.4.1. Modificar datos en el libro de salida de muestras.

R16.4.2. Registrar traza.

R16.5. Visualizar el libro de salida de muestras.

R16.6. Imprimir el libro de salida de muestras.

R16.7. Realizar reporte del libro de salida de muestras.

R16.7.1. Generar reporte.

R16.7.2. Imprimir reporte.

R17. Gestionar Pedido de MR.

R17.1. Crear un nuevo Pedido de MR.

R17.2. Buscar Pedido de MR.

R17.3. Modificar Pedido de MR.

R17.3.1. Modificar un Pedido de MR.

R17.3.2. Registrar traza.

R17.4. Visualizar Pedido de MR.

R17.5. Imprimir Pedido de MR.

R17.6. Realizar reporte con un listado ordenado por fecha de creación de los pedidos de MR.

R17.6.1. Generar reporte.

R17.6.2. Imprimir Reporte.

R18. Gestionar Planilla de Análisis Atrasados.

R18.1. Crear una Planilla de Análisis Atrasados.

R18.2. Buscar una Planilla de Análisis Atrasados.

R18.3. Modificar una Planilla de Análisis Atrasados.

R18.3.1. Modificar datos de una Planilla de Análisis Atrasados.

R18.3.2. Registrar traza.

R18.4. Visualizar una Planilla de Análisis Atrasados.

R18.5. Imprimir una Planilla de Análisis Atrasados.

R19. Gestionar Carta de Solicitud del material candidato a MR.

R19.1. Crear una nueva Carta de Solicitud del material.

R19.2. Buscar una Carta de Solicitud del material.

R19.3. Modificar Carta de Solicitud.

R19.3.1. Modificar datos de una Carta de Solicitud del material.

R19.3.2. Registrar traza.

R19.4. Visualizar una Carta de Solicitud del material.

R19.5. Imprimir una Carta de Solicitud del material.

R19.6. Insertar un recordatorio a la solicitud indicada.

R19.7. Buscar recordatorios de una solicitud.

R19.8. Imprimir recordatorio.

R20. Gestionar Diseño Experimental para EE del MR.

R20.1. Crear un nuevo Diseño Experimental para EE del MR.

R20.2. Buscar el Diseño Experimental para EE del MR.

R20.3. Modificar Diseño Experimental para EE del MR.

R20.3.1. Modificar datos del Diseño Experimental para EE del MR.

R20.3.2. Registrar traza.

R20.4. Visualizar el Diseño Experimental para EE del MR.

R20.5. Imprimir el Diseño Experimental para EE del MR.

2.1.6. Requerimientos no Funcionales

Los requerimientos no funcionales especifican propiedades o cualidades que el producto debe tener. Estas características hacen al producto atractivo, usable, rápido y confiable. Las categorías para clasificar los requerimientos no funcionales que se emplean son: apariencia o interfaz externa, usabilidad, rendimiento, soporte, portabilidad, seguridad, políticos culturales, legales, confiabilidad, ayuda y documentación en línea, software, hardware, restricciones en el diseño y la implementación.

- ✓ **Apariencia o interfaz externa.**

El sistema tendrá los colores correspondientes al logo del CIGB. Las páginas de la aplicación no se cargarán con mucha información y contendrán sólo las imágenes necesarias que respondan a las pautas de diseño definidas por la Dirección de Diseño de la Universidad para la línea INFORMÁTICA MÉDICA.

- ✓ **Usabilidad.**

La aplicación podrá ser utilizada por personal vinculado a la Biotecnología, que tengan conocimientos básicos de computación y de aplicaciones Web.

- ✓ **Rendimiento.**

El sistema dará las respuestas realizadas al servidor central en cuestión de segundos (no debe excederse los 3 segundos en las respuestas).

- ✓ **Soporte.**

Cuando se instale el LIMS de Calidad en el CIGB, se les dará un curso de familiarización con la nueva herramienta, ya que la misma la usarán para la mayoría de las tareas que desarrollan en estos momentos. El equipo de desarrollo de la aplicación visitará a los Laboratorios/Grupos 1 vez semanalmente, en el 1er mes de instalada la aplicación, y luego visitará 1 vez al mes en los restantes 4 meses después de instalada, con el objetivo de dar mantenimiento a la misma. Después de instalada la aplicación, se cogerá la primera semana para realizar todas las pruebas necesarias, según el diseño de las pruebas para probar la funcionalidad completa del sistema

- ✓ **Portabilidad.**

El sistema podrá ser usado sobre los sistemas operativos Windows y Linux.

- ✓ **Seguridad.**

El sistema tendrá un registro de trazas de documentos y planillas modificadas por los usuarios, para garantizar el control de las operaciones de este tipo en el sistema. Se debe garantizar que la información sensible sólo pueda ser vista por los usuarios con el nivel de acceso adecuado y que las funcionalidades del sistema se muestren de acuerdo al usuario que esté activo. El sistema debe contar con protección contra acciones no autorizadas o que puedan afectar la integridad de los datos. Se podrá acceder a algunas funcionalidades del sistema desde cualquier computadora personal que esté fuera del CIGB

- ✓ **Políticos Culturales.**

El idioma que se empleará en la aplicación será el español. El sistema tendrá logotipos e imágenes en correspondencia con el carácter científico y profesional del CIGB. Algún cambio que se desee realizar en la aplicación, será tramitado por la dirección del proyecto por parte del CIGB y canalizado por los directivos de Producción de la UCI.

- ✓ **Legales.**

El sistema será registrado en el Centro Nacional de Derecho de Autor a través de la Dirección de Servicios Legales de la UCI previo acuerdo con el CIGB. El sistema se dará a conocer a todos los trabajadores de la Dirección de Calidad como la herramienta para gestionar la información de cada uno de los grupos y laboratorios. Se estará usando para el desarrollo de la aplicación herramientas de software libre con licencia GNU/GPL con la excepción de la herramienta para la gestión de requerimientos que es un software propietario pero es la más indicada para ese proceso.

- ✓ **Confiabilidad.**

El sistema será usado y administrado solamente por trabajadores de la Dirección de Calidad del CIGB, por lo tanto la información que fluirá en el mismo, será la emitida por cada uno de los grupos y laboratorios. Podrán acceder a visualizar ciertas informaciones, directivos de otras áreas, con previa consulta a la dirección del proyecto y a los desarrolladores de la aplicación.

- ✓ **Ayuda y documentación en línea.**

El sistema tendrá una ayuda para el uso de la herramienta, así como posibilitará la comunicación entre los desarrolladores de la aplicación y los usuarios, para cualquier dificultad que pueda ser tramitada de inmediato.

- ✓ **Software.**

Para el servidor de la aplicación el sistema operativo recomendado es Windows XP o cualquier distribución de Linux. Se debe instalar un servidor Web Apache 2.2.6 o superior. Las computadoras clientes de los usuarios accederán al sistema usando cualquier navegador que soporte JavaScript; entre ellos, los siguientes navegadores: Internet Explorer 5.5 o superior, Mozilla 1.7 o superior. El servidor de Base de datos debe tener instalado Postgree SQL 8.3, el Sistema Operativo debe ser Linux o Windows XP.

- ✓ **Hardware.**

Se deberá contar con impresora en las computadoras clientes que interactúen con la aplicación. El servidor de Base de datos debe tener las siguientes características: capacidad de disco duro superior a 160 GB, microprocesador Pentium IV superior a 2.0 GHz y como mínimo 1.0 GB de RAM. El servidor de aplicaciones debe tener las siguientes características: capacidad de disco duro superior a 80 GB, microprocesador Pentium IV superior a 2.0 GHz y como mínimo 1.0 GB de RAM.

- ✓ **Restricciones en el diseño y la implementación.**

La lógica de presentación constituirá una capa independiente de la lógica de negocio, centrando su función en la interfaz de usuario y validaciones de los datos de entrada. Se usará el lenguaje de programación PHP 5 y el gestor de bases de datos PostgreSQL 8.3, así como Symfony como framework de desarrollo.

2.2. Actores del Sistema a automatizar: módulo EEMR

Luego del análisis de los actores y trabajadores del negocio en el módulo EEMR, definido en tesis anteriores se especifica el siguiente actor del sistema:

Nombre del actor	Descripción
Analista	Manipula todas las funcionalidades del módulo EEMR del LIMS de Calidad de la Dirección de Calidad.

Tabla 2.1 Actor del Sistema

2.3. Casos de Uso del Sistema

La identificación de los casos de uso es la guía del Ingeniero de Software que lleva adelante el desarrollo de un sistema. Se muestran a continuación los Casos de Uso del Sistema que se identificaron para el módulo EEMR, basados en los requerimientos expuestos con anterioridad:

1. Gestionar Resultado de EE (SIC-0894).
2. Gestionar Programa de EE (SIC-0892).
3. Gestionar Recordatorio de solicitud de material.
4. Gestionar Solicitud de producto (SDF-01S).
5. Gestionar Cronograma de EE (SIC-0893).
6. Gestionar Registro de Descargo de MR (SIC-0827).
7. Gestionar Registro de Salida de documentos de EE.
8. Gestionar Solicitud de EE (SIC-0882).
9. Gestionar Diseño de MR.
10. Gestionar Diseño Experimental de Estudio de Homogeneidad del MR.
11. Gestionar Diseño Experimental de Estudio de Caracterización del MR.
12. Gestionar Certificado de MR (SIC-0232).
13. Gestionar Solicitud de Destrucción de Productos (SIC-0830).
14. Gestionar Registro de Control de Copias de los Certificados de MR (SIC-0861).
15. Gestionar Expediente MR.
16. Gestionar Libro de Salida de Muestras.
17. Gestionar Pedido de MR.
18. Gestionar Planilla de Análisis Atrasados.
19. Gestionar Carta de Solicitud del material candidato a MR.
20. Gestionar Diseño Experimental para EE del MR.

2.4. Matriz de Trazabilidad

La existencia de distintos tipos de requisitos y la derivación de unos a partir de otros, hace necesario que se establezcan mecanismos de trazabilidad. Los requisitos no son independientes entre sí, para poder determinar el impacto de un cambio en un requisito, las relaciones entre requisitos deben ser entendidas, documentadas y mantenidas. Para mantener la trazabilidad se realizan las matrices de seguimiento. En ellas se muestra la relación entre requisitos y entre requisitos y casos de uso.

Para ver la matriz de trazabilidad del módulo EEMR, remitirse al *Expediente de Proyecto.*

2.5. Diagrama de Casos de Uso del Sistema: Módulo EEMR

Se estructuró el diagrama de casos de uso del sistema en paquetes que en su interior contienen los actores del sistema y los Casos de Uso, éstos últimos son todos significativos, por lo que el diagrama de Casos de Uso del sistema coincide con la Vista de casos de uso.

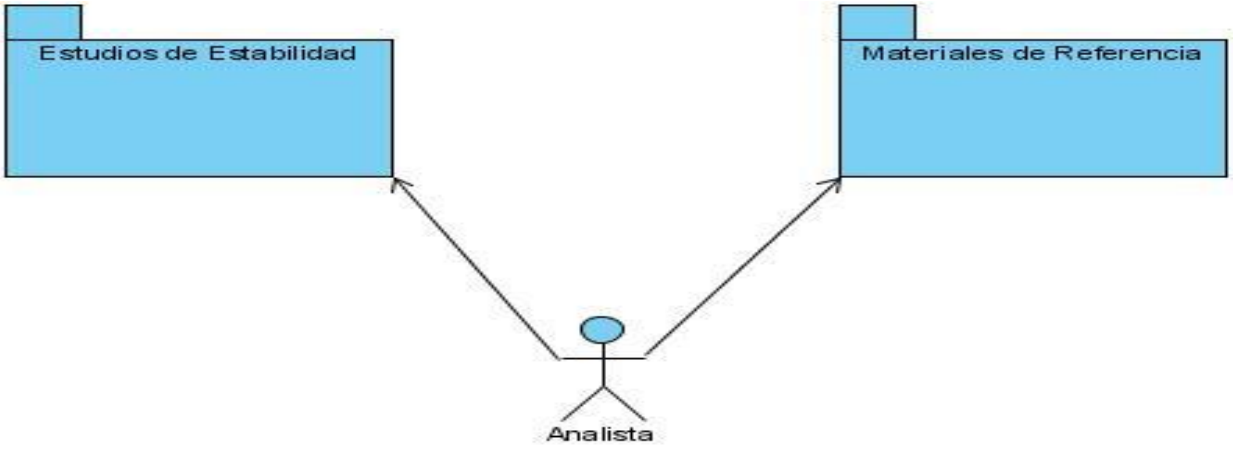


Figura 2.1 Diagrama de CU del Sistema 1

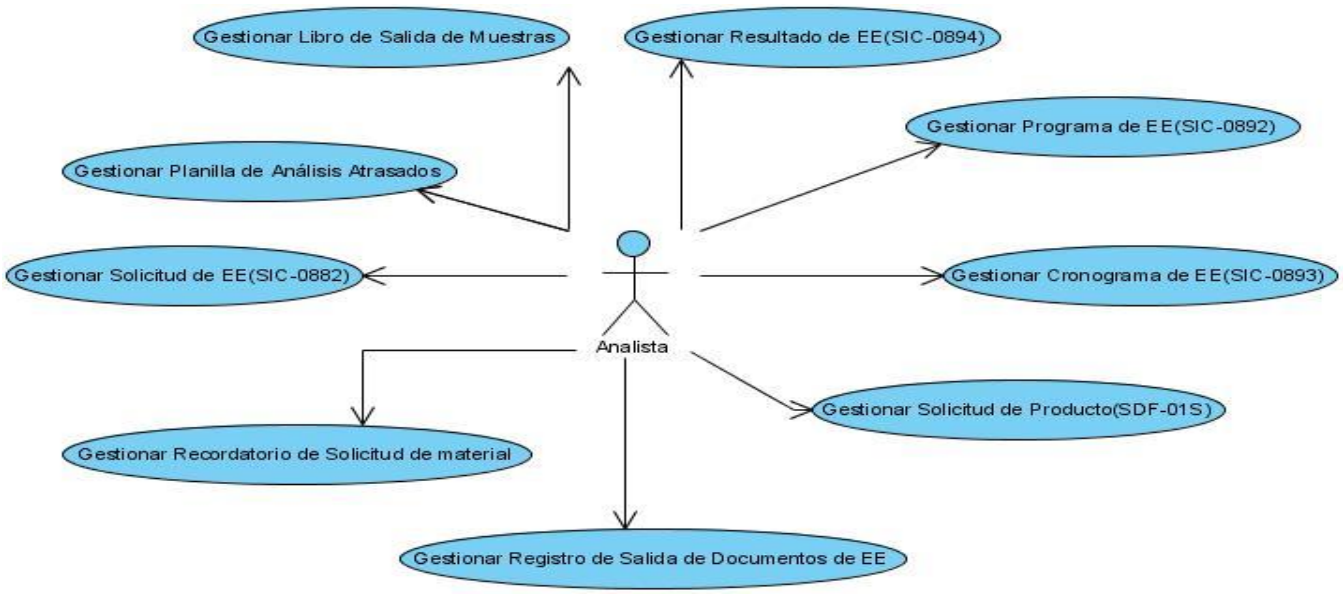


Figura 2.2 Diagrama de CU Paquete EE 1

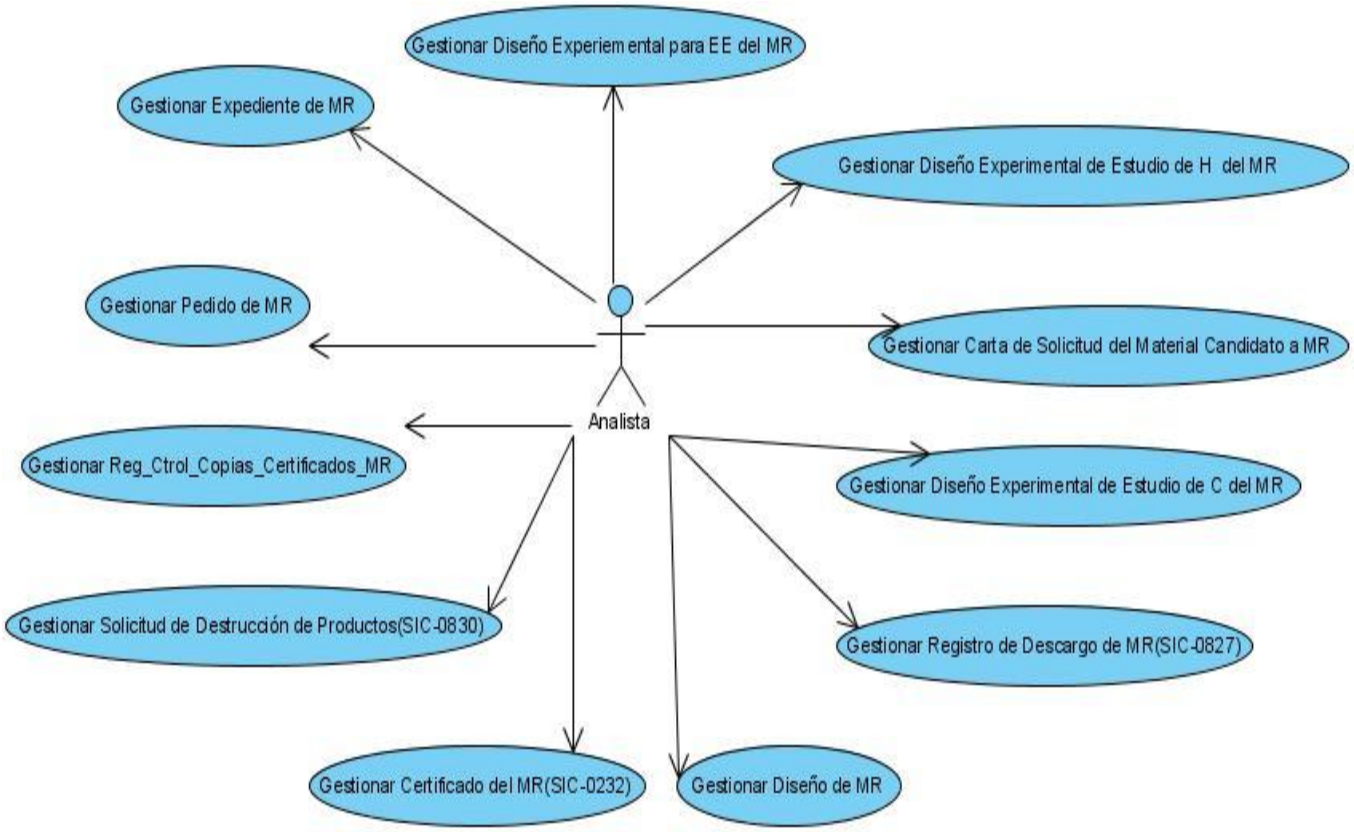


Figura 2.3 Diagrama de CU Paquete MR 1

2.6. Descripciones textuales

Las descripciones textuales se utilizan para detallar cada caso de uso, donde se describe el flujo de sucesos en una especificación precisa de la secuencia de acciones. Se incluye el comienzo, la terminación y la interacción entre los actores y lo que el sistema debe hacer. A continuación se muestra la descripción textual del CU "Gestionar Cronograma de EE (SIC-0893)" y las descripciones resumidas de los restantes casos de uso. Para ver las descripciones ampliadas remitirse al *Expediente de Proyecto*.

Nombre del Caso de Uso	Gestionar Cronograma de EE (SIC-0893).
Actores	Analista (inicia).
Propósito	Crear un cronograma de EE (SIC-0893) para un Estudio de Estabilidad,

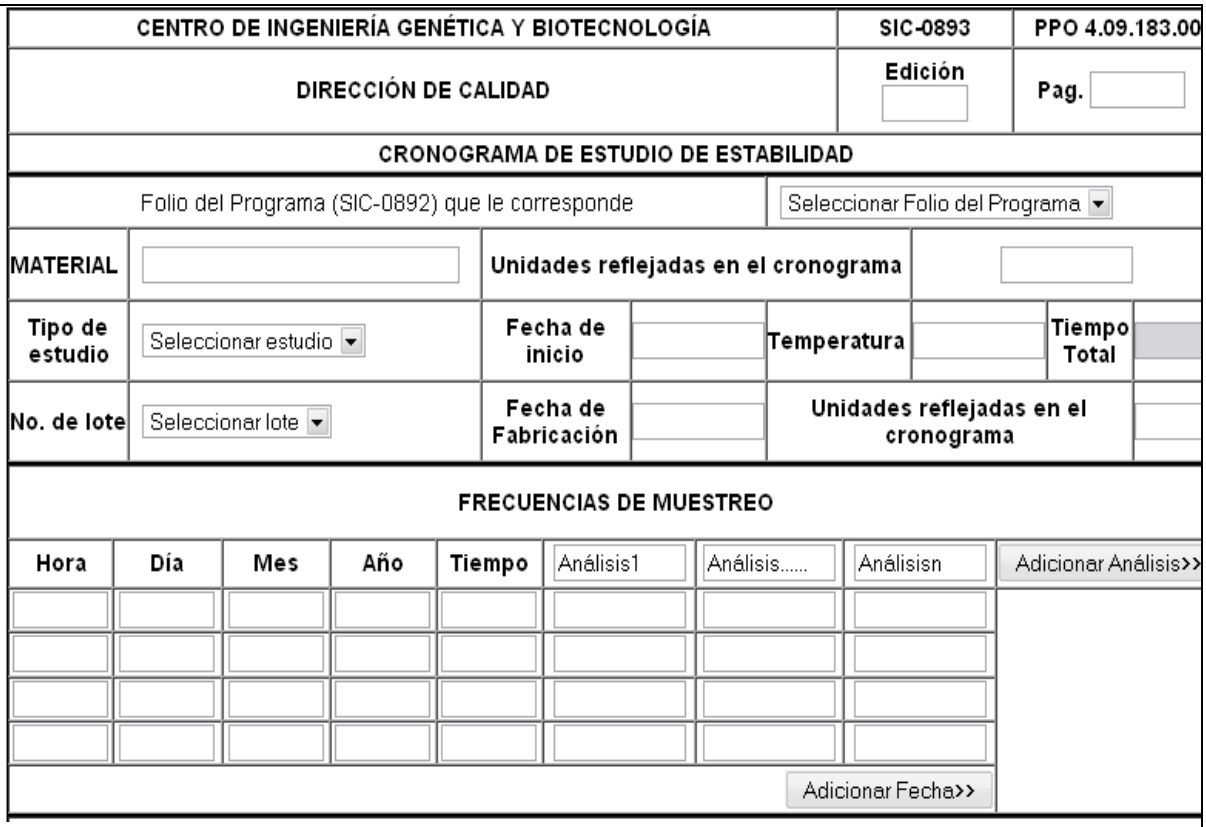
	buscar, visualizar, imprimir y modificar los datos de un Cronograma de EE existente.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con el cronograma de EE: • Crear nuevo cronograma de EE. • Buscar el cronograma de EE. • Modificar el cronograma de EE. • Visualizar el cronograma de EE. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.
Referencias	R5.1, R5.2, R5.2.1, R5.3, R5.4, R5.5
Precondiciones	Que el Analista se haya autenticado en la aplicación.
Pos-condiciones	El sistema crea un nuevo cronograma de EE, modifica datos, busca y visualiza datos de algún cronograma de EE existente.

Flujo normal de los eventos	
Acción del actor	**Respuesta del Sistema**
1. El Analista indica realizar una de las siguientes operaciones: • Crear nuevo cronograma de EE. • Buscar un cronograma de EE. • Modificar un cronograma de EE. • Visualizar un cronograma de EE.	2. El sistema, en dependencia de la operación que solicita realizar, hace lo siguiente: • Si indica crear un nuevo cronograma de EE, ir a Sección **"Crear nuevo cronograma de EE"**. • Si indica buscar un cronograma de EE, ir a la Sección **"Buscar un cronograma de EE"**. • Si indica modificar un cronograma de EE, ir a la Sección **"Modificar un cronograma de EE"**. • Si indica Visualizar un cronograma de EE ir a la Sección **"Visualizar cronograma de EE"**.
Sección "Crear nuevo cronograma de EE"	

Acción del actor	Respuesta del Sistema
	1. Muestra la interfaz correspondiente para la creación de un nuevo cronograma de EE. - El campo "Folio del programa que le corresponde" contiene los valores del campo "Folio" perteneciente a todos los programas de EE. - El campo "N_0 de lote" contiene el valor consecutivo del campo "N_0 de lote" del último Cronograma de EE.
2. El analista selecciona un valor para el campo "Folio del programa que le corresponde".	3. En tiempo de ejecución muestra los datos correspondientes al programa seleccionado.
4. El analista provee los datos para crear el cronograma. - Unidades reflejadas en el cronograma. - Tipo de estudio. - Fecha de inicio. (la fecha proporcionada al campo "Fecha de inicio" contiene Día/Mes/Año.) - Fecha de Fabricación. - Temperatura. - Tiempo Total. - Observaciones. - Realizado por: (NOMBRE Y APELLIDOS, FECHA). - Revisado por: (NOMBRE Y APELLIDOS). - Aprobado por: (NOMBRE Y	5. En tiempo de ejecución genera, busca, muestra el valor de varios campos y da la posibilidad de proporcionar otro valor para estos campos. - Al campo "MATERIAL" le asigna el valor del campo "MATERIAL" correspondiente al programa seleccionado. - El campo "Tipo de estudio" muestra el valor "Real" y/o "Acelerado" en dependencia de varios criterios: ▪ Si está seleccionado el campo "ESTUDIO REAL" correspondiente al programa seleccionado entonces el valor del campo "Tipo de estudio" muestra el valor "Real" y los demás datos tomados del programa se toman del estudio real. ▪ Si está seleccionado el campo "ESTUDIO ACELERADO" correspondiente al programa seleccionado entonces el valor del campo "Tipo de estudio" muestra el valor

APELLIDOS). Tabla Frecuencia de Muestreo - Fecha y Hora. - Tiempo. - Unidades/Análisis. - Análisis. - Terminado.	"Acelerado" y los demás datos tomados del programa se toman del estudio acelerado. ▪ Si está seleccionado el campo "ESTUDIO REAL" y el campo "ESTUDIO ACELERADO" correspondiente al programa seleccionado entonces el campo "Tipo de estudio" muestra dos valores, "Real" y "Acelerado". Da la posibilidad de que el analista seleccione uno de los dos valores mostrados. - Se llenan los campos de la tabla que contiene una fecha dividida en (Hora, Día, Mes, Año), Tiempo, Unidades/Análisis. ▪ La "Fecha" y "Hora" toman el valor indicado por el analista. ▪ El campo "Tiempo" toma el valor del campo "Frecuencia" del programa asociado. ▪ El campo "Unidades/Análisis", toma el valor del campo "No. Análisis" correspondiente al análisis del programa asociado. ▪ Los análisis a realizar de la tabla son los valores del campo "ANALISIS" correspondiente al programa seleccionado.
6. El analista indica crear el cronograma.	7. Verifica que el campo obligatorio "Folio del programa que le corresponde" contenga un valor.
	8. Verifica que el parámetro "Terminado" está seleccionado.
	9. Verifica que todos los campos tengan valor excepto los campos "Hora", "Día" y "Observaciones" que en ciertos momentos pueden quedarse vacíos.

	10. Crea el cronograma.
	11. Muestra un mensaje de verificación "Los datos han sido guardados satisfactoriamente" se ejecuta la acción 1 de la sección "**Crear nuevo cronograma de EE**".
Flujos alternos Sección "Crear nuevo cronograma de EE"	
2.1) El analista no provee un valor para el campo "Folio del programa que le corresponde", indica otra opción del sistema finalizando el caso de uso.	
4.1) El analista no provee los datos para la creación del cronograma, indica otra opción del sistema y finaliza el caso de uso.	
6.1) El analista no crea el cronograma, indica otra opción del sistema y finaliza el caso de uso.	
	7.1) Si no está seleccionado el campo "Folio del programa que le corresponde" para la creación del nuevo cronograma, lo señala.
	8.1) Si el campo "Terminado" no está seleccionado lo guarda como "No Terminado".
	9.1) Si el/los dato(s) necesario(s) para terminar el cronograma están vacíos los señala.
Prototipo	➢ Prototipos de Grupo de Desarrollo\Cronograma de EE (SIC-0893)\ Crear cronograma de EE.htm.



CENTRO DE INGENIERÍA GENÉTICA Y BIOTECNOLOGÍA		SIC-0893	PPO 4.09.183.00
DIRECCIÓN DE CALIDAD		Edición	Pag.

CRONOGRAMA DE ESTUDIO DE ESTABILIDAD

Folio del Programa (SIC-0892) que le corresponde	Seleccionar Folio del Programa				
MATERIAL		Unidades reflejadas en el cronograma			
Tipo de estudio	Seleccionar estudio	Fecha de inicio		Temperatura	Tiempo Total
No. de lote	Seleccionar lote	Fecha de Fabricación		Unidades reflejadas en el cronograma	

FRECUENCIAS DE MUESTREO

Hora	Día	Mes	Año	Tiempo	Análisis1	Análisis.....	Análisisn	Adicionar Análisis>>
								Adicionar Fecha>>

Sección "Buscar un cronograma de EE"

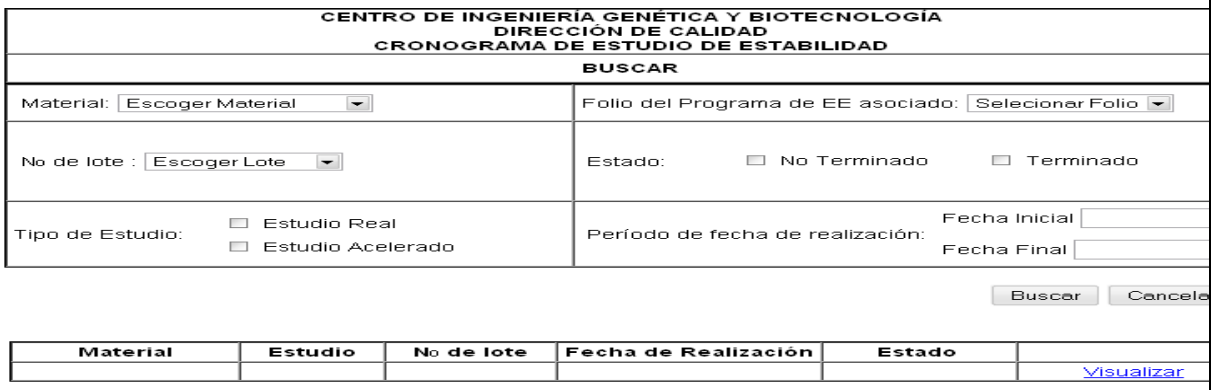
Acción del actor	Respuesta del Sistema
	1. Muestra la interfaz correspondiente a la búsqueda de un cronograma. - El campo "Material" contiene los valores del campo "MATERIAL" de todos los cronogramas de EE existentes en la BD. - El campo "Folio del programa asociado" contiene un listado con todos los valores del campo "Folio del programa que le corresponde" de los cronogramas existentes en la BD. - El campo "N_0 de lote" contiene los valores del campo "N_0 de lote" de todos los cronogramas de EE

	existentes en la BD. - Da la posibilidad de seleccionar todos los materiales.
2. El analista provee el o los parámetros para la búsqueda del cronograma. - Material. - Folio del programa asociado. - Tipo de Estudio (Acelerado o Real). - Período de Fecha de realización (Fecha inicial y Fecha final). - N_0 de lote. - Estado (Terminado, No terminado). - Buscar Todos los cronogramas existentes. El analista indica buscar el cronograma que cumpla con el o los parámetros indicados.	3. Verifica que al menos un parámetro tenga valor.
	4. Busca los cronogramas de EE que cumplan con los parámetros introducidos.
	5. Muestra el resultado de la búsqueda ordenado por el campo “Id” en una nueva interfaz donde se muestra una tabla que muestra los parámetros “MATERIAL”, “Estudio”, “Folio del Programa de EE”, “N_0 lote”, “Estado” y “Realizado por: (FECHA)”. Da la posibilidad de visualizar o modificar los cronogramas de EE encontrados.
6. El analista selecciona una de las siguientes opciones: - Visualizar cronograma.	7. En dependencia de la opción que solicita realizar, hace lo siguiente: - Si indica visualizar el cronograma, ir a la Sección

- Modificar cronograma.	**"Visualizar un cronograma".** - Si indica modificar el cronograma, ir a la Sección **"Modificar datos de un cronograma".**
Flujos alternos Sección "Buscar un cronograma de EE"	
2.1) El analista no realiza la búsqueda del cronograma e indica otra opción del sistema finalizando el caso de uso.	3.1) Si al menos un parámetro no tiene valor entonces emite un mensaje de error, "Especifique al menos un parámetro para la búsqueda".
	4.1) Si no encuentra un cronograma que cumpla con los parámetros indicados emite un mensaje de error, "No existe el Cronograma de EE indicado"
6.1) El analista no realiza ninguna de las opciones mostradas e indica otra opción del sistema finalizando el caso de uso.	

Prototipo

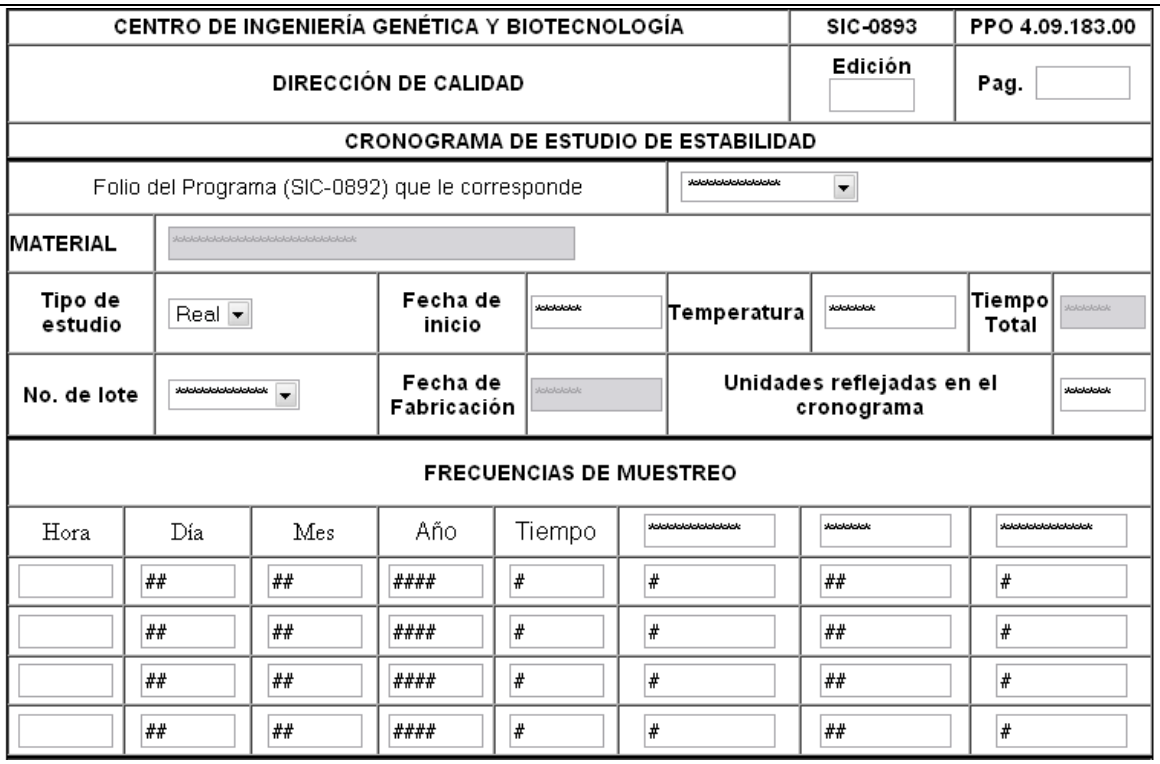
➢ Prototipos de Grupo de Desarrollo\Cronograma de EE (SIC-0893)\ Buscar Cronograma de EE.htm.



Sección "Modificar datos de un cronograma de EE"	
Acción del actor	**Respuesta del Sistema**
	1. Muestra el cronograma listo para modificar. - El campo "Folio del programa que le corresponde"

	contiene los valores del campo “Folio” perteneciente a todos los programas de EE existentes y da la posibilidad de seleccionar el valor de “No procede”.
2. El analista provee los datos a modificar: - Material. - Unidades reflejadas en el cronograma. - Tipo de estudio. - N_0 de lote. - Fecha de inicio. (la fecha proporcionada al campo “Fecha de inicio” contiene Día/Mes/Año.) - Fecha de Fabricación. - Temperatura. - Tiempo Total. - Observaciones. - Realizado por: (NOMBRE Y APELLIDOS, FECHA). - Revisado por: (NOMBRE Y APELLIDOS). - Aprobado por: (NOMBRE Y APELLIDOS). Tabla Frecuencia de Muestreo - Fecha y Hora. - Tiempo. - Unidades/Análisis. - Análisis. - Terminado. El analista indica modificar el	

cronograma.	
	3. Verifica que el campo obligatorio “Folio del programa que le corresponde” contenga un valor.
	4. Muestra los datos del cronograma seleccionado y da la posibilidad de modificarlo.
5. El analista solicita modificar los datos del cronograma.	
	6. Actualiza en la base de datos los cambios realizados.
	7. Actualiza registro de trazas.
	8. Muestra un mensaje de verificación “Los datos han sido guardados satisfactoriamente” y se ejecuta la acción 1 de la sección “**Modificar datos de un cronograma de EE**”.
Flujos alternos Sección “Modificar datos de un cronograma de EE”	
2.1) El analista no modifica el cronograma e indica otra opción del sistema finalizando el caso de uso.	3.1) Si el campo obligatorio no contiene un valor lo señala.
5.1) El analista no modifica los datos del cronograma e indica otra opción del sistema finalizando el caso de uso.	
Prototipo	➢ Prototipos de Grupo de Desarrollo\Cronograma de EE (SIC-0893)\ Modificar Cronograma de EE.htm.



CENTRO DE INGENIERÍA GENÉTICA Y BIOTECNOLOGÍA		SIC-0893	PPO 4.09.183.00
DIRECCIÓN DE CALIDAD		Edición	Pag.

CRONOGRAMA DE ESTUDIO DE ESTABILIDAD

Folio del Programa (SIC-0892) que le corresponde							
MATERIAL							
Tipo de estudio	Real	**Fecha de inicio**		**Temperatura**		**Tiempo Total**	
No. de lote		**Fecha de Fabricación**		**Unidades reflejadas en el cronograma**			

FRECUENCIAS DE MUESTREO

Hora	Día	Mes	Año	Tiempo			
	##	##	####	#	#	##	#
	##	##	####	#	#	##	#
	##	##	####	#	#	##	#
	##	##	####	#	#	##	#

Sección "Visualizar un cronograma de EE"	
Acción del actor	**Respuesta del Sistema**
	1. Muestra una interfaz con el Cronograma listo para imprimir.
2. El analista selecciona imprimir el cronograma.	3. El sistema imprime el cronograma.
Flujos alternos Sección "Visualizar un cronograma de EE"	
Acción del actor	**Respuesta del Sistema**
2.1 El analista no indica imprimir el cronograma e indica otra opción del sistema finalizando el caso de uso.	

Prototipo

➢ Prototipos de Grupo de Desarrollo\Cronograma de EE (SIC-0893)\ Visualizar Cronograma de EE.htm.

CENTRO DE INGENIERÍA GENÉTICA Y BIOTECNOLOGÍA							SIC-0893	PPO 4.09.183
DIRECCIÓN DE CALIDAD							Edición ***	Pag. ***
CRONOGRAMA DE ESTUDIO DE ESTABILIDAD								
Folio del Programa (SIC-0892) que le corresponde				**********				
MATERIAL	**********							
Tipo de estudio	********	Fecha de inicio	********	Temperatura	********	Tiempo Total	*****	
No. de lote	********	Fecha de Fabricación	********	Unidades reflejadas en el cronograma			*****	
FRECUENCIAS DE MUESTREO								
Hora	Día	Mes	Año	Tiempo	**********	**********	**********	
##:##	##	##	####	#	#	#	##	
##:##	##	##	####	##	#	#	##	
##:##	##	##	####	##	#	#	##	
##:##	##	##	####	#	#	#	##	

Tabla 2.2 Descripción del CUS "Gestionar Cronograma de EE (SIC-0893)"

Nombre del Caso de Uso	Gestionar Solicitud de producto (SDF-01S).
Actores	Analista (inicia).
Propósito	Crear una solicitud de producto, buscar, visualizar, imprimir, modificar los datos de una solicitud de producto existente.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con la Solicitud de producto: • Crear nueva Solicitud de producto. • Buscar una Solicitud de producto. • Modificar una Solicitud de producto. • Visualizar una Solicitud de producto. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta

	las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.3 Descripción del CUS "Gestionar Solicitud de producto (SDF-01S)."

Descripción del CUS "Gestionar el Recordatorio de solicitud de material".

Nombre del Caso de Uso	Gestionar el Recordatorio de solicitud de material.
Actores	Analista (inicia).
Propósito	Crear un recordatorio de solicitud de material, buscar, visualizar, imprimir, enviar y modificar los datos de un recordatorio de solicitud de material existente.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con el recordatorio de solicitud de material: • Crear nuevo Recordatorio de solicitud de material. • Buscar el Recordatorio de solicitud de material. • Modificar el Recordatorio de solicitud de material. • Visualizar el Recordatorio de solicitud de material. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.4 Descripción del CUS "Gestionar el Recordatorio de solicitud de material".

Nombre del Caso de Uso	Gestionar el Programa de EE (SIC-0892).
Actores	Analista (inicia).
Propósito	Crear un Programa de EE (SIC-0892) para un Estudio de Estabilidad, buscar, visualizar, imprimir y modificar los datos de un programa de EE existente.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con el programa de EE: • Crear nuevo programa de EE. • Buscar el programa de EE. • Modificar el programa de EE. • Visualizar el programa de EE. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta

	las acciones necesarias finalizando así el caso de uso.

Tabla 2.5 Descripción del CUS "Gestionar Programa de EE" (SIC-0892).

Nombre del Caso de Uso	Gestionar Registro de Descargo de MR (SIC-0827)
Actores	Analista (inicia).
Propósito	Crear un registro de descargo (SIC-0827) para controlar la entrega de cada Material de Referencia (MR), registrar una entrega, buscar, visualizar, imprimir, modificar los datos de un registro de descargo existente y generar reportes.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con el registro de descargo: • Crear nuevo registro de descargo. • Registrar entrega en el registro de descargo. • Buscar el registro de descargo. • Modificar el registro de descargo. • Visualizar el registro de descargo. • Generar reportes. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.6 Descripción del CUS "Gestionar Registro de Descargo de MR (SIC-0827).

Nombre del Caso de Uso	Gestionar Diseño de MR.
Actores	Analista (inicia).
Propósito	Crear un diseño de MR, buscar, visualizar, imprimir, modificar los datos de un diseño de MR existente.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con el Diseño de MR: • Crear nueva Diseño de MR. • Buscar el Diseño de MR. • Modificar el Diseño de MR. • Visualizar el Diseño de MR.

	El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.7 Descripción del CUS "Gestionar Diseño de MR".

Nombre del Caso de Uso	Gestionar Solicitud de EE (SIC-0882).
Actores	Analista (inicia).
Propósito	Crear una Solicitud de Estudios Formales Estabilidad (SIC-0882) para un Estudio de Estabilidad, crear, buscar, visualizar, imprimir y modificar los datos de una solicitud de EE existente.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con la solicitud de EE: • Crear una nueva solicitud de EE. • Buscar solicitud de EE. • Modificar solicitud de EE. • Visualizar solicitud de EE. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.8 Descripción del CUS "Gestionar Solicitud de EE (SIC-0882)".

Nombre del Caso de Uso	Gestionar Registro de Salida de documentos de EE.
Actores	Analista (inicia).
Propósito	Crear un Registro de Salida de documentos de EE para un Estudio de Estabilidad, registrar, buscar, visualizar, imprimir y modificar los datos de un Registro de Salida de documentos de EE existente.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguientes operaciones relacionadas con el Registro de Salida de documentos de EE. • Registrar Registro de Salida de documentos de EE. • Buscar el Registro de Salida de documentos de EE. • Modificar el Registro de Salida de documentos de EE. • Visualizar el Registro de Salida de documentos de EE.

	• Generar reportes. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.9 Descripción del CUS "Gestionar Registro de Salida de documentos de EE".

Nombre del Caso de Uso	Gestionar Certificado de MR (SIC-0232).
Actores	Analista (inicia).
Propósito	Crear un certificado para un material de referencia determinado que se va a buscar, visualizar, imprimir, modificar los datos de un certificado existente.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con el Certificado de MR: • Crear nuevo Certificado de MR. • Buscar el Certificado de MR. • Modificar el Certificado de MR. • Visualizar el Certificado de MR. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.10 Descripción del CUS "Gestionar Certificado de MR" (SIC-0232).

Nombre del Caso de Uso	Gestionar Registro de Control de Copias de los Certificados de MR (SIC-0861).
Actores	Analista (inicia).
Propósito	Crear un registro de control de copias para un certificado de material de referencia determinado que se va a buscar, visualizar, imprimir, modificar, registrar y generar los datos de un certificado existente.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con el Registro de Control de Copias de los Certificado de MR: • Crear nuevo Registro de Control de Copias de los Certificado de MR. • Buscar Registro de Control de Copias de los Certificado de MR. • Modificar Registro de Control de Copias de los Certificado de MR.

	• Visualizar Registro de Control de Copias de los Certificado de MR. • Generar Reporte El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.11 Descripción del CUS "Gestionar Registro de Control de Copias de los Certificados de MR"

Nombre del Caso de Uso	Gestionar Solicitud de Destrucción de Productos (SIC-0830).
Actores	Analista (inicia).
Propósito	Crear una solicitud de destrucción de un producto que se va a buscar, visualizar, imprimir, modificar los datos de una solicitud existente y generar reportes e imprimir el reporte.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con la Solicitud de Destrucción de Productos: • Crear nueva Solicitud de Destrucción de Productos. • Buscar la Solicitud de Destrucción de Productos. • Modificar la Solicitud de Destrucción de Productos • Visualizar la Solicitud de Destrucción de Productos. • Generar reportes. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.12 Descripción del CUS "Gestionar Solicitud de Destrucción de Productos (SIC-0830)"

Nombre del Caso de Uso	Gestionar Pedido de MR.
Actores	Analista (inicia).
Propósito	Crear un pedido de materiales de referencia que se va a entregar, buscar, visualizar, imprimir, modificar el estado de una solicitud mostrada, generar reportes e imprimir el reporte.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con el Pedido de Materiales de Referencia: • Crear un nuevo Pedido de Materiales de Referencia

	• Buscar un Pedido de Materiales de Referencia. • Modificar un Pedido de Materiales de Referencia. • Visualizar un Pedido de Materiales de Referencia. • Generar reportes. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.13 Descripción del CUS "Gestionar Pedido de MR".

Nombre del Caso de Uso	Gestionar Carta de Solicitud del Material Candidato a MR.
Actores	Analista (inicia).
Propósito	Crear una carta de solicitud del material candidato a material de Referencia que se va a buscar, visualizar, imprimir y modificar los datos de una solicitud existente
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguientes operaciones relacionadas con la Carta de Solicitud del Material Candidato a Material de Referencia: • Crear nueva Carta de Solicitud del Material Candidato a Material de Referencia. • Buscar la Carta de Solicitud del Material Candidato a Material de Referencia. • Modificar la Carta de Solicitud del Material Candidato a Material de Referencia. • Visualizar la Carta de Solicitud del Material Candidato a Material de Referencia. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.14 Descripción del CUS "Gestionar Carta de Solicitud del Material Candidato a MR".

Nombre del Caso de Uso	Gestionar Planilla de Análisis Atrasado.
Actores	Analista (inicia).

Propósito	Crear una planilla de análisis atrasado que se va a buscar, visualizar, imprimir y modificar los datos de una solicitud existente
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguientes operaciones relacionadas con la Planilla de Análisis Atrasado: • Crear nueva Planilla de Análisis Atrasado. • Buscar la Planilla de Análisis Atrasado. • Modificar la Planilla de Análisis Atrasado. • Visualizar la Planilla de Análisis Atrasado. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.15 Descripción del CUS "Gestionar Planilla de Análisis Atrasado".

Nombre del Caso de Uso	Gestionar Expediente de MR.
Actores	Analista (inicia).
Propósito	Crear un Expediente de Materiales de Referencia que se va a buscar, visualizar, imprimir y modificar los datos de un expediente existente.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguientes operaciones relacionadas con el Expediente de Materiales de Referencia: • Crear nuevo Expediente de Materiales de Referencia. • Buscar Expediente de Materiales de Referencia. • Modificar Expediente de Materiales de Referencia. • Visualizar Expediente de Materiales de Referencia. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.16 Descripción del CUS "Gestionar Expediente de MR".

Nombre del Caso de Uso	Gestionar el Diseño Experimental de Estudio de Caracterización del MR.
Actores	Analista (inicia).

Propósito	Crear un Diseño Experimental de Estudio de Caracterización del MR, buscar, visualizar, imprimir, modificar los datos de un Diseño Experimental de Estudio de Caracterización de un MR existente.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con el Diseño Experimental de Estudio de Caracterización del MR: • Crear nuevo Diseño Experimental de Estudio de Caracterización de MR. • Buscar el Diseño Experimental de Estudio de Caracterización de MR. • Modificar el Diseño Experimental de Estudio de Caracterización de MR. • Visualizar el Diseño Experimental de Estudio de Caracterización de MR. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.17 Descripción del CUS "Gestionar el Diseño Experimental de Estudio de Caracterización del MR".

Nombre del Caso de Uso	Gestionar el Diseño Experimental de Estudio de Homogeneidad del MR.
Actores	Analista (inicia).
Propósito	Crear un Diseño Experimental de Estudio de Homogeneidad del MR, buscar, visualizar, imprimir, modificar los datos de un Diseño Experimental de Estudio de Homogeneidad de un MR existente.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con el Diseño Experimental de Estudio de Homogeneidad del MR: • Crear nuevo Diseño Experimental de Estudio de Homogeneidad de MR. • Buscar el Diseño Experimental de Estudio de Homogeneidad de MR. • Modificar el Diseño Experimental de Estudio de Homogeneidad de MR. • Visualizar el Diseño Experimental de Estudio de Homogeneidad de MR.

	El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.18 Descripción del CUS "Gestionar el Diseño Experimental de Estudio de Homogeneidad del MR".

Nombre del Caso de Uso	Gestionar el Libro de Salida de Muestras.
Actores	Analista (inicia).
Propósito	Crear un Libro de Salida de Muestras que se va a buscar, visualizar, imprimir y modificar los datos de un expediente existente
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguientes operaciones relacionadas con el Libro de Salida de Muestras: • Crear nuevo Libro de Salida de Muestras. • Buscar Libro de Salida de Muestras. • Modificar Libro de Salida de Muestras. • Visualizar Libro de Salida de Muestras. • Generar Reporte. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.19 Descripción del CUS "Gestionar el Libro de Salida de Muestras".

Nombre del Caso de Uso	Gestionar Resultado de EE (SIC-0894).
Actores	Analista (inicia).
Propósito	Crear un Resultado de EE que se va a buscar, visualizar, imprimir y modificar.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con el Resultado de EE: • Crear un nuevo Resultado de EE • Buscar un Resultado de EE. • Modificar un Resultado de EE. • Visualizar un Resultado de EE. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta

	las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.20 Descripción del CUS "Gestionar Resultado de EE (SIC - 0894)".

Nombre del Caso de Uso	Gestionar Diseño Experimental para EE del MR.
Actores	Analista (inicia).
Propósito	Crear un Diseño Experimental para Estudios de Estabilidad a los Materiales de Referencia, buscar, visualizar, imprimir, modificar los datos de un Diseño Experimental para Estudios de Estabilidad de materiales de referencia existentes.
Resumen	El caso de uso se inicia cuando el analista va a realizar alguna de las siguiente operaciones relacionadas con el Diseño Experimental para Estudios de Estabilidad de materiales de referencia: • Crear nuevo Diseño Experimental para EE del MR. • Buscar el Diseño Experimental para EE del MR. • Modificar el Diseño Experimental para EE del MR. • Visualizar el Diseño Experimental para EE del MR. El sistema le muestra la interfaz correspondiente según su solicitud y ejecuta las acciones necesarias finalizando así el Caso de Uso.

Tabla 2.21 Descripción del CUS "Gestionar Diseño Experimental para EE del MR".

Conclusiones

En este capítulo se muestran las técnicas de captura, validación y verificación de requerimientos, se obtuvieron los requerimientos funcionales y no funcionales, se identificaron los actores y casos de uso del sistema y se realizó el diagrama de casos de uso donde se muestra la relación entre actores y casos de uso. Se describieron los casos de uso y se realizó la matriz de trazabilidad entre requisitos funcionales y casos de uso.

Capítulo 3
Diseño del Sistema

Introducción

En el Diseño se modela el sistema para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Se crea un punto de partida para las actividades de implementación capturando los requisitos o subsistemas individuales, interfaces y clases. Es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. En el presente capítulo se hace referencia a la arquitectura del sistema a través de la Vista Lógica y de Despliegue, se presentan los prototipos no funcionales correspondientes al caso de uso "Gestionar Cronograma de EE (SIC-0893)", además el diagrama de secuencia correspondiente a este caso de uso. Se muestran los diagramas de clases del diseño y el mapa de navegación, así como las clases persistentes y el modelo de datos.

Para ver el resto de los artefactos remitirse al *Expediente de Proyecto*.

3.1. *Propósitos del Diseño*

El diseño comienza con el modelo de los requisitos. Se trabaja por transformar este modelo y obtener cuatro niveles de detalles de diseño: la estructura de datos, la arquitectura del sistema, la representación de la interfaz y los detalles a nivel de componentes. Una vez que se analizan y especifican los requisitos del software, el diseño es la primera de las tres actividades técnicas (diseño, generación de código y pruebas) que se requieren para construir y verificar el software. Cada actividad transforma la información de manera que se obtenga un software con la calidad requerida.

Concretamente se puede definir como propósitos del diseño [2]:

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.

- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.
- Capturar las interfaces entre los subsistemas antes en el ciclo de vida del software, lo cual es muy útil cuando se utilizan interfaces como elementos de sincronización entre diferentes equipos de desarrollo.
- Para modelar el diseño se utilizan las clases del diseño que son una construcción similar en la implementación del sistema.
- El lenguaje utilizado para especificar una clase del diseño es lo mismo que el lenguaje de programación. Las operaciones, atributos, tipos, visibilidad, se pueden especifican con la sintaxis del lenguaje elegido.

Los métodos de una clase del diseño tienen correspondencia directa con el correspondiente método en la implementación de las clases.

3.2. *Principios de diseño*

"UML posee una extensión para el modelado de aplicaciones Web, propuesta por Conallen, dicha extensión es usada para el diseño de las clases. Los estereotipos que usa esta extensión son" [18]:

<<*Server Page*>> Representa la página Web que tiene código que se ejecuta en el servidor. Este código interactúa con recursos en el servidor. Las operaciones representan las funciones del código y los atributos las variables visibles dentro del alcance de la página.

Esta clase sólo puede tener relaciones con objetos en el servidor, una relación 1:1 con un fichero en el servidor. En las aplicaciones en PHP se corresponde con un fichero .php.

<<*Client Page*>> Una instancia de Página Cliente es una página Web, con formato HTML; mezcla de datos, presentación y lógica. Son interpretadas por el browser. Cada página cliente solo puede ser construida por una página servidor.

<<*Form*>> “Grupo de elementos de entrada que son parte de una página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML. Sus atributos son los elementos de entrada del formulario (Text Field, Text Area, Button, Label, Radio Button, Radio Group, Select, Check Box y Hidden Fields).” [18]

3.3. *Referencia a la arquitectura del Sistema*

"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución."[2]

Una Arquitectura de Software, también denominada *Arquitectura lógica*, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

La Arquitectura de Software establece los fundamentos para que analistas, diseñadores y programadores, trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

La arquitectura de un sistema consta de múltiples vistas, asociadas a diferentes dimensiones o perspectivas del sistema.

A continuación se especifica una de estas vistas, la Vista Lógica. Más adelante se caracteriza otra de las vistas, Vista de Despliegue.

3.3.1. Vista Lógica

La Vista Lógica describe la funcionalidad interna del sistema, representa los elementos de diseño más importantes para la arquitectura del sistema. Describe las clases más importantes, su organización en paquetes y subsistemas, éstos en capas y las realizaciones de casos de uso más importantes.

Para estructurar la vista del sistema, no se identificaron en el módulo EEMR subsistemas funcionales, se organizó por paquetes a partir de los casos de uso arquitectónicamente significativos, que

constituyen la totalidad de los casos de uso identificados. Existen paquetes formados por otros paquetes y éstos tendrán como todos, su diagrama de clases. Los paquetes generales que se identificaron son: Estudios de Estabilidad y Materiales de Referencia. A continuación se muestra la Vista Lógica.

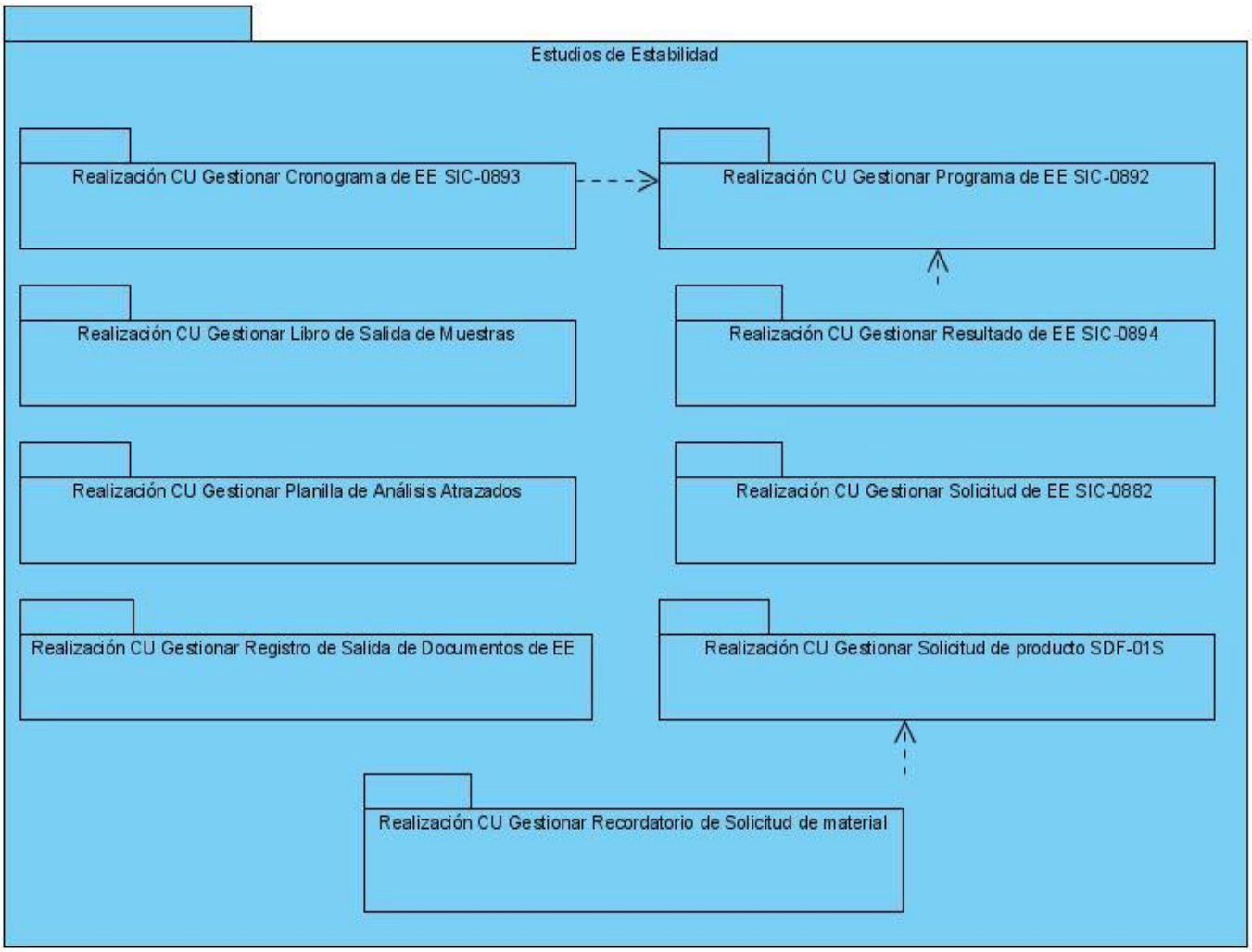


Figura 3.1 Vista Lógica: Paquete EE 1

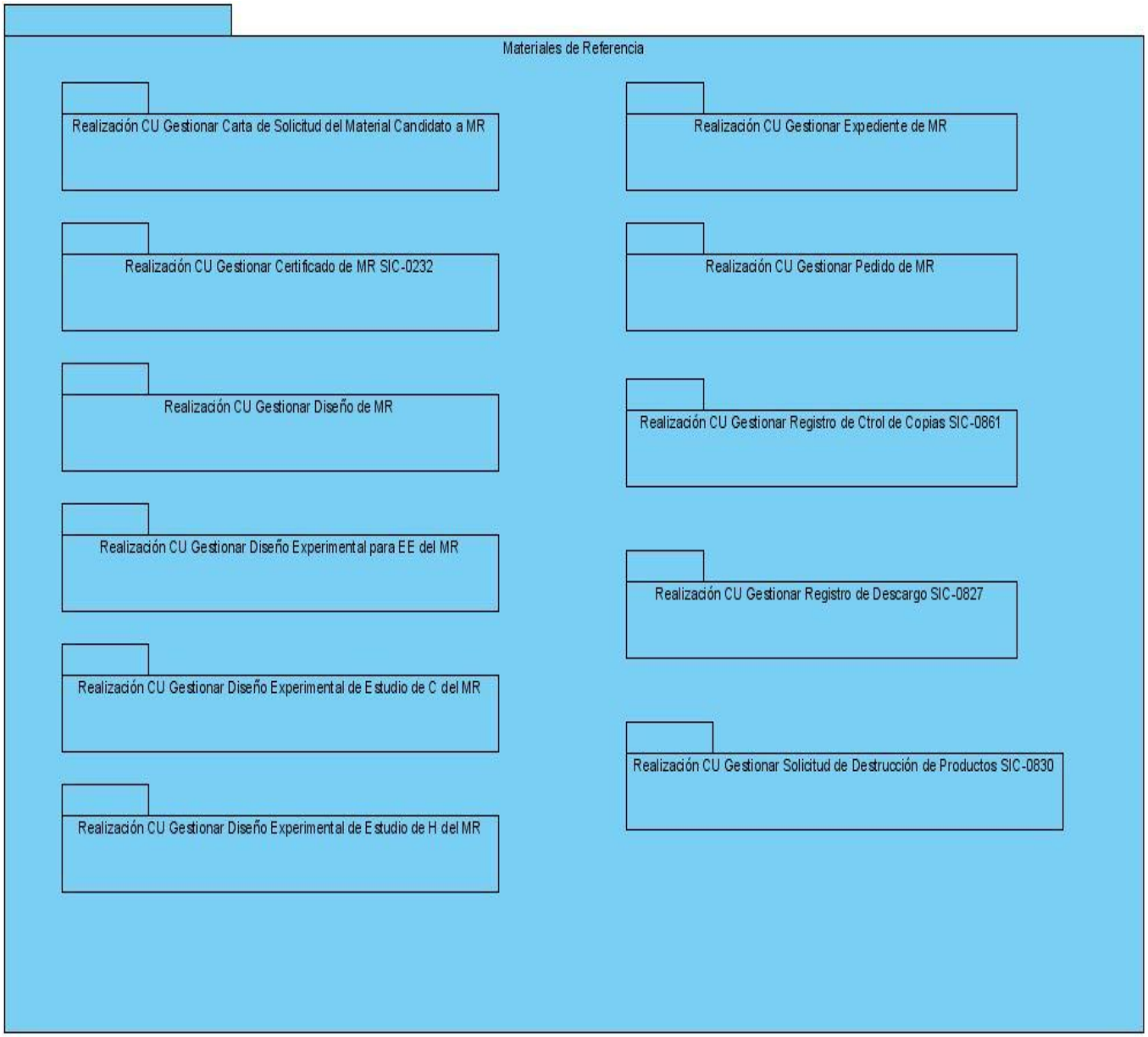


Figura 3.2 Vista Lógica: Paquete MR 1

Figura 3.3 Realización CU Gestionar Cronograma de EE (SIC-0893) 1

3.3.2. Patrones de arquitectura y diseño

Dentro de los patrones de arquitectura existentes Symfony utiliza el Modelo-Vista-Controlador. Como patrones de diseño los patrones GRASP (Creador, Experto, Alta Cohesión, Controlador, Bajo Acoplamiento) y los patrones GOF (Creacionales, Estructurales).

El MVC está formado por tres niveles:

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

"La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones." [15]. Este patrón puede transformar una aplicación simple realizada con PHP en una aplicación que sigue la arquitectura MVC.

El MVC da la posibilidad que se ejecuten tanto navegadores se deseen, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones. El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación (Ver Anexo 5).

Para la implementación del MVC que realiza Symfony se necesita de los siguientes componentes:

La capa del Modelo
• Abstracción de la base de datos
• Acceso a los datos

La capa de la Vista
• Vista
• Plantilla

• Layout

La capa del Controlador
• Controlador frontal
• Acción

En total son siete scripts, lo que parecen muchos archivos para abrir y modificar cada vez que se crea una página. Symfony toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. Esto se logra debido a que el controlador frontal y el layout son comunes para todas las acciones de la aplicación. Se pueden tener varios controladores y varios layouts, pero solamente es obligatorio tener uno de cada uno. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que Symfony lo genera de forma automática (Ver Anexo 6).

- **Patrones GRASP que implementa Symfony:**

Creador

En la clase Actions se encuentran las acciones definidas para la aplicación. En esta clase se crean los objetos de las clases que representan las entidades, por lo que de este modo la clase Actions es "creador" de dichas entidades.

Experto

Propel es la librería externa que utiliza Symfony para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

Alta Cohesión

Symfony permite asignar responsabilidades con una alta cohesión, por ejemplo la clase Actions tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, es decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el software sea flexible frente a grandes cambios.

Controlador

Todas las peticiones Web son manejadas por un solo controlador frontal (sfActions), que es el punto de

entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

Bajo Acoplamiento

La clase Action hereda solamente de sfActions para lograr un bajo acoplamiento de clases.

- **Patrones GOF que implementa Symfony:**

El framework Symfony utiliza una serie de patrones GOF como son:

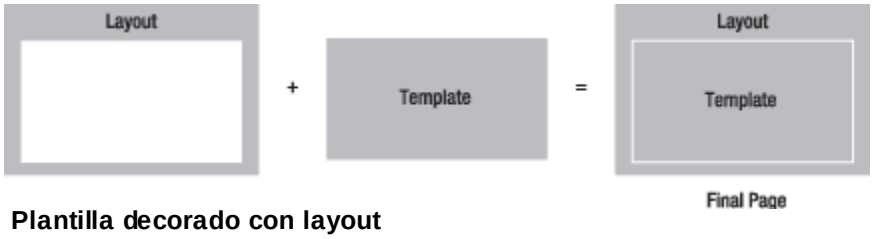
En la categoría Creacionales:

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En el controlador frontal hay una llamada a sfContext::getInstance(). En una acción, el método getContext(), un objeto muy útil que guarda una referencia a todos los objetos del núcleo de Symfony.

Abstract Factory (Fábrica abstracta): Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Cuando el framework necesita por ejemplo crear un nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

En la categoría Estructurales:

Decorator (Envoltorio): Añade funcionalidad a una clase, dinámicamente. El archivo layot.php, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, y este decora la plantilla. En la siguiente figura se ilustra lo antes mencionado:



Plantilla decorado con layout

El Decorator se debe usar para:

- Adicionar responsabilidades a objetos individuales dinámicamente sin afectar otros objetos.
- Para agregar responsabilidades que pueden ser retiradas.
- Cuando no es práctico adicionar responsabilidades por medio de la herencia.

Algunas ventajas de este patrón son las siguientes:

- Las responsabilidades pueden añadirse y eliminarse en tiempo de ejecución, es decir, es un poco más flexible que la herencia.
- Diferentes decoradores pueden ser conectados a un mismo objeto.
- Reduce el número de propiedades en las clases de la parte alta de la jerarquía.
- Es simple añadir nuevos decoradores de forma independiente a las clases que extienden.

Composite (Objeto compuesto): Permite tratar objetos compuestos como si se tratase de un objeto simple. Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol .Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.

3.3.3. Descripciones de las Clases del Diseño.

Las clases del diseño son abstracciones de clases o construcciones similares de implementación. Tienen, además, operaciones, parámetros, atributos, tipo; necesarios para su implementación en el lenguaje de programación elegido. Estas clases especifican los métodos los cuales tienen

correspondencia directa con los métodos en la implementación. Por tal motivo es de suma importancia describir dichas clases, ya que especifica lo que deben hacer estos métodos.

A continuación se muestra los métodos correspondientes a la clase actions del caso de uso "**Gestionar Cronograma de EE (SIC-0893)**", los demás métodos de las actions correspondientes a los restantes casos de uso no son descritos ya que los métodos se comportan de forma similar

Clases del Diseño del caso de uso "Gestionar Cronograma de EE (SIC-0893)"

Nombre de la clase:	**class SIC0893Actions**
Nombre del método:	**executeNew** (sfWebRequest $request)
Descripción:	Este método crea los diferentes formularios a utilizar.
Nombre del método:	**executeCreate** (sfWebRequest $request)
Descripción:	Este método crea la planilla "Cronograma de EE" de acuerdo a los parámetros entrado por el usuario.
Nombre del método:	**executeEdit**(sfWebRequest $request)
Descripción:	Este método construye el formulario para luego ser modificado.
Nombre del método:	**executeSearch** (sfWebRequest $request)
Descripción:	Este método construye el formulario de búsqueda SearchSIC0893.
Nombre del método:	**executeVisualizar** (sfWebRequest $request)
Descripción:	Visualiza la planilla del Cronograma de EE.

Tabla 3.1 Clases del Diseño del CU "Gestionar Cronograma de EE (SIC-0893)"

3.3.4. Diagramas de clases del diseño

Los diagramas de clases del diseño representan las clases y sus relaciones. Representan la parte estática del sistema ya que muestran las clases con sus métodos y atributos, así como las relaciones estáticas entre ellas.

Symfony está basado en la arquitectura Modelo-Vista-Controlador. Esta separa la lógica de negocio, es decir, el modelo, y la presentación, la vista, por lo que se consigue un sencillo mantenimiento de las aplicaciones.

El controlador contiene el código que liga la lógica de negocio con la presentación, tiene todas las funcionalidades que se realiza en el sistema. Es de donde sale la información que se va a realizar, es decir las acciones crear, buscar, etc.,

El modelo en Symfony es una capa tipo ORM realizada mediante el propel, el acceso y la modificación de los datos almacenados en la base de datos se realiza mediante objetos, de esta manera nunca se accede directamente a la bases de dato, el modelo controla la información que se va a guardar en el sistema. Este contiene las clases base las cuales dan el acceso a los atributos de los objetos persistentes en la base de datos y, las no bases las cuales heredan de las clases bases. En las clases peer se realizan las consultas definas por el usuario.

La vista se encarga de producir las páginas que se muestran como resultado de las acciones, es la interfaz que se le muestra al usuario. En la vista se encuentra el layout.php, plantilla global que almacena código HTML común a todas las páginas de la aplicación. En dependencia de la petición que le llega al controlador se selecciona la acción a realizar, la plantilla succes.php construye la client page correspondiente que trabaja con un form que es el encargado de hacerle el submit al index que es donde se realizan las acciones.

.

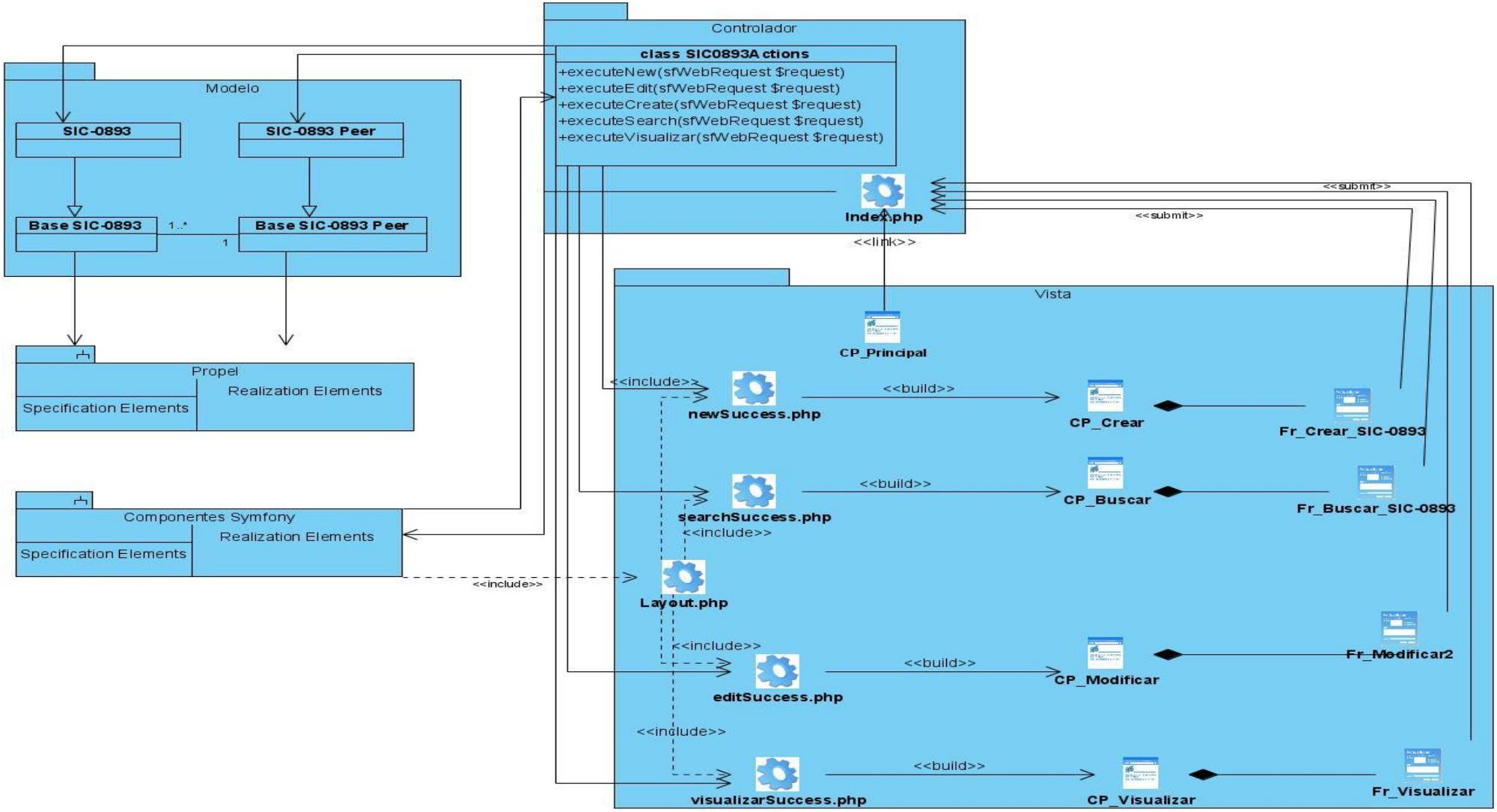


Figura 3.4 Diagrama de Clases del Diseño CU: Gestionar Cronograma de EE 1

3.3.5. Modelo de Despliegue

"Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación." [19].

La vista es la distribución de los componentes a través de los nodos. Dicha vista se especifica en el capítulo 4, donde se actualiza el diagrama de despliegue con dicha distribución. A continuación se muestra el diagrama de despliegue, el cual está estructurado de la siguiente manera:

Los nodos representan un recurso de cómputo, normalmente un procesador o un dispositivo hardware similar, se cuenta con un Servidor de Aplicaciones conectado a la PC Cliente, a través del protocolo HTTPS. Además se tendrá mediante una conexión USB una impresora para la impresión de documentos

Para el almacenamiento de los datos de la aplicación se utiliza un servidor de Base de Datos representado como un nodo y para lograr la conexión del sistema con la base de datos se utiliza TCP / IP + SSL como protocolo de comunicación.

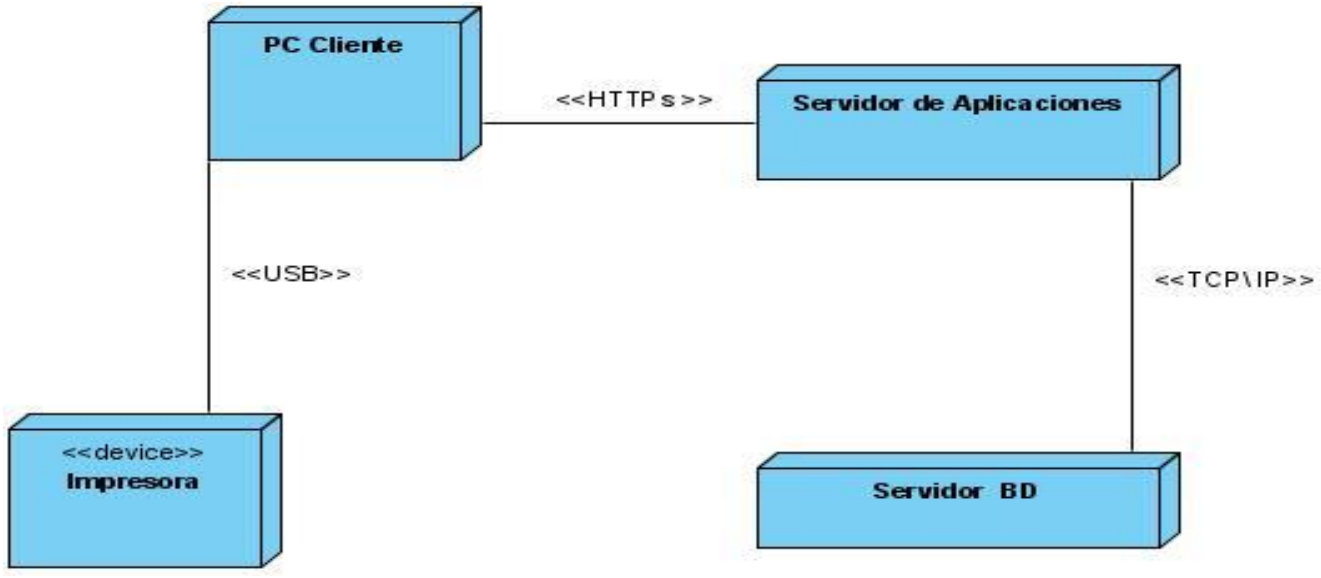


Figura 3.5 Diagrama de Despliegue 1

3.4. *Prototipos de interfaz de usuario*

Los prototipos de interfaz de usuario ayudan a comprender especificar las interacciones entre el sistema y los usuarios, son una representación de la funcionalidad contenida en los casos de uso donde permiten que el usuario verifique que el sistema va a satisfacer sus necesidades. De esta manera se válida que se esté cumpliendo con los requerimientos.

Para ver los prototipos no funcionales de los Casos de Uso del Sistema remitirse al *Expediente de Proyecto.* En la descripción textual ampliada del caso de uso "Gestionar Cronograma de EE (SIC-0893)" se muestran los prototipos correspondientes.

3.5. *Mapa de Navegación*

Un mapa de navegación es una forma ordenada de distribuir tanto la información como las labores que esto implica, adecuándose a las necesidades especificas de las personas que lo utilizaran. Es una representación del sitio a través de una secuencia de interfaces. Para la construcción del mapa de navegación se utilizó UML, la navegabilidad se estructuró de la siguiente manera: a partir de la página principal (Index) se podrá acceder a los diferentes departamentos y grupos de trabajo, entre ellos el grupo EEMR, una vez en él, estarán disponibles las páginas que contienen información detallada de los procesos que se allí realizan, y a través de ellas tener acceso a la gestión de los diferentes documentos. Como parte de la navegabilidad del sitio se podrá ir de las diferentes páginas que representan los SIC u otras planillas al grupo de trabajo EEMR, y de este último se podrá volver a la página principal (Index) y viceversa.

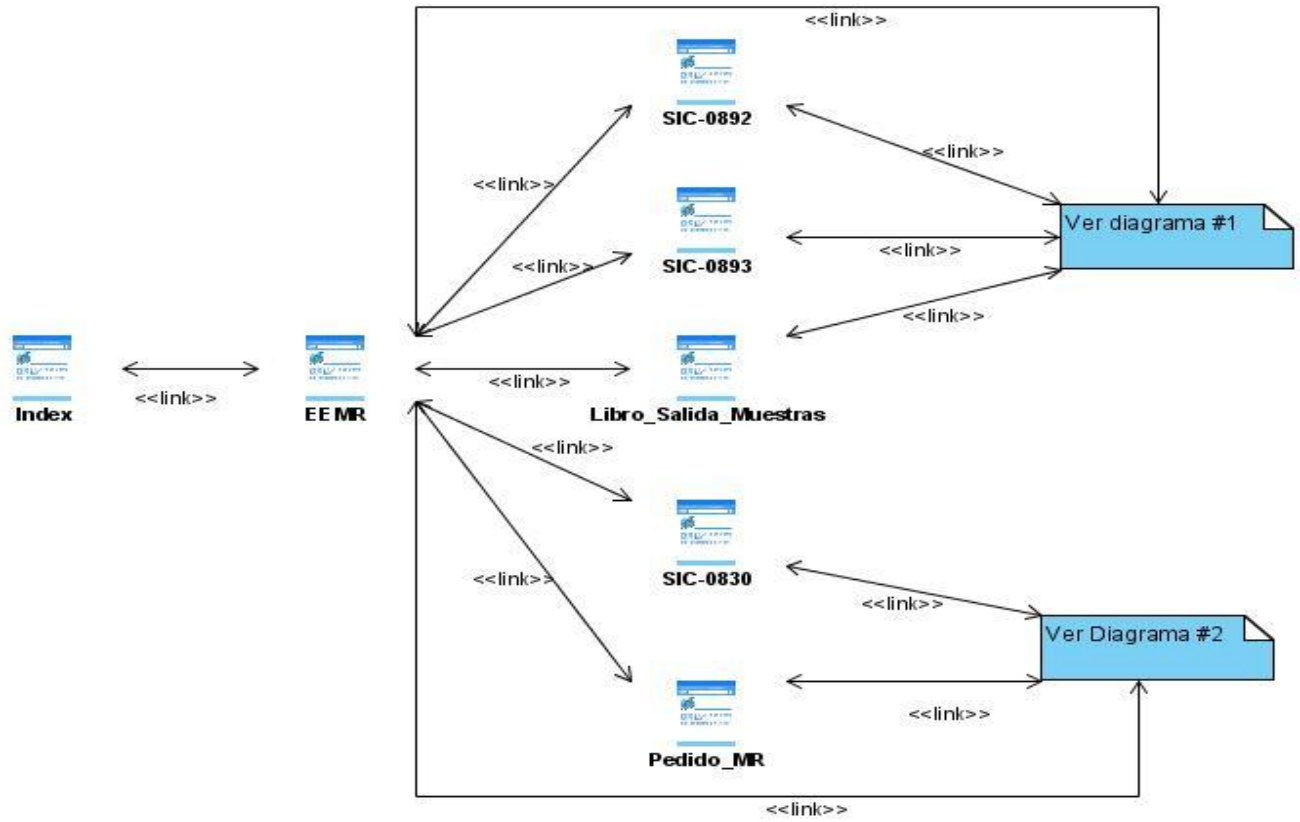
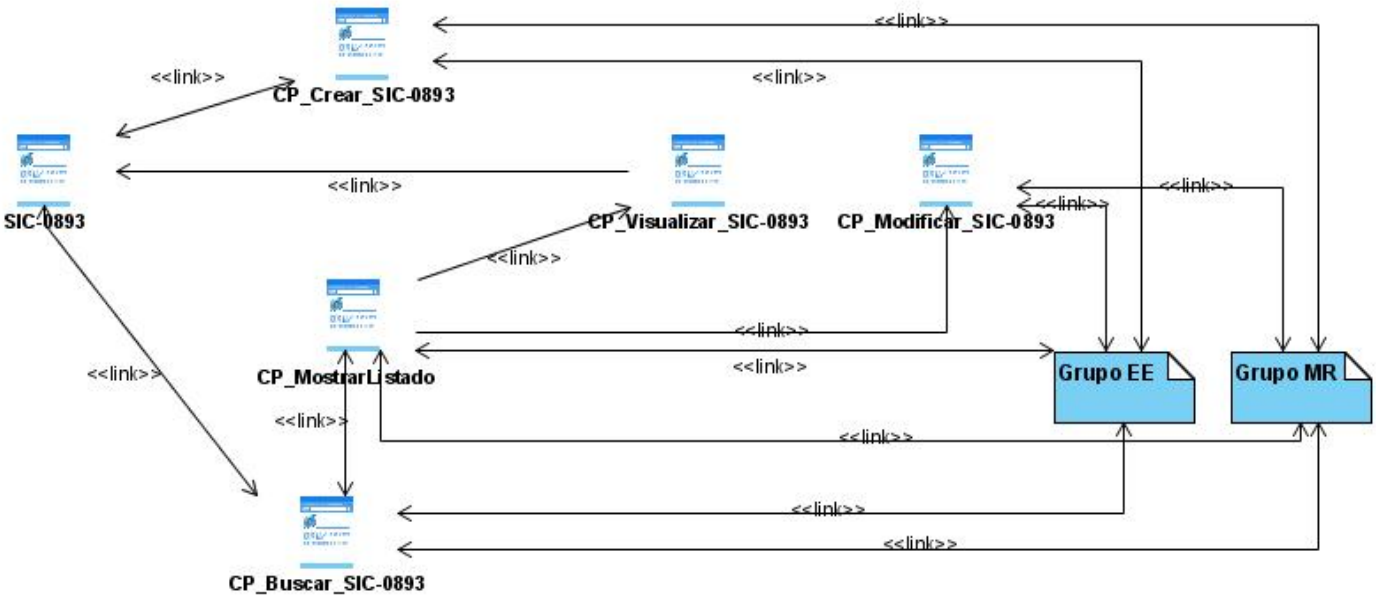


Figura 3.6 Mapa de Navegación 1

La siguiente figura muestra un ejemplo donde se observan las relaciones que existen entre las acciones que se podrán realizar cuando se selecciona un SIC u otra planilla, se podrá seleccionar una de las opciones de: Crear o Buscar, luego de ser buscado el SIC u otra planilla se muestra un Listado donde se podrán modificar o visualizar para posteriormente imprimir. Las paginas Crear, Buscar y Modificar pueden navegar hacia todas las páginas de la aplicación, lo cual se define por las notas "Grupo EE" (en este grupo están las planillas correspondiente a SIC-0892, SIC-0893 y Libro de Salida de Muestras) y "Grupo MR" (en este grupo están las planillas correspondientes a SIC-0830 y Pedido de MR) Sólo se muestra la navegabilidad del diagrama SIC-0893, para ver los demás diagramas remitirse al *Expediente de Proyecto.*



3.6. *Diagrama de Secuencia*

Un diagrama de secuencia tiene como objetivo fundamental encontrar una secuencia de interacciones entre objetos, detalladas y ordenadas en el tiempo. El diagrama de secuencia muestra en los diferentes escenarios de un caso de uso, los eventos generados por actores externos, y los eventos internos del sistema. A continuación se presentan los escenarios: *Escenario Buscar, Escenario Crear SIC-0893, Escenario Modificar SIC-0893 y el Escenario Visualizar SIC-0893,* que corresponden al CU *"Gestionar Cronograma de EE"*.

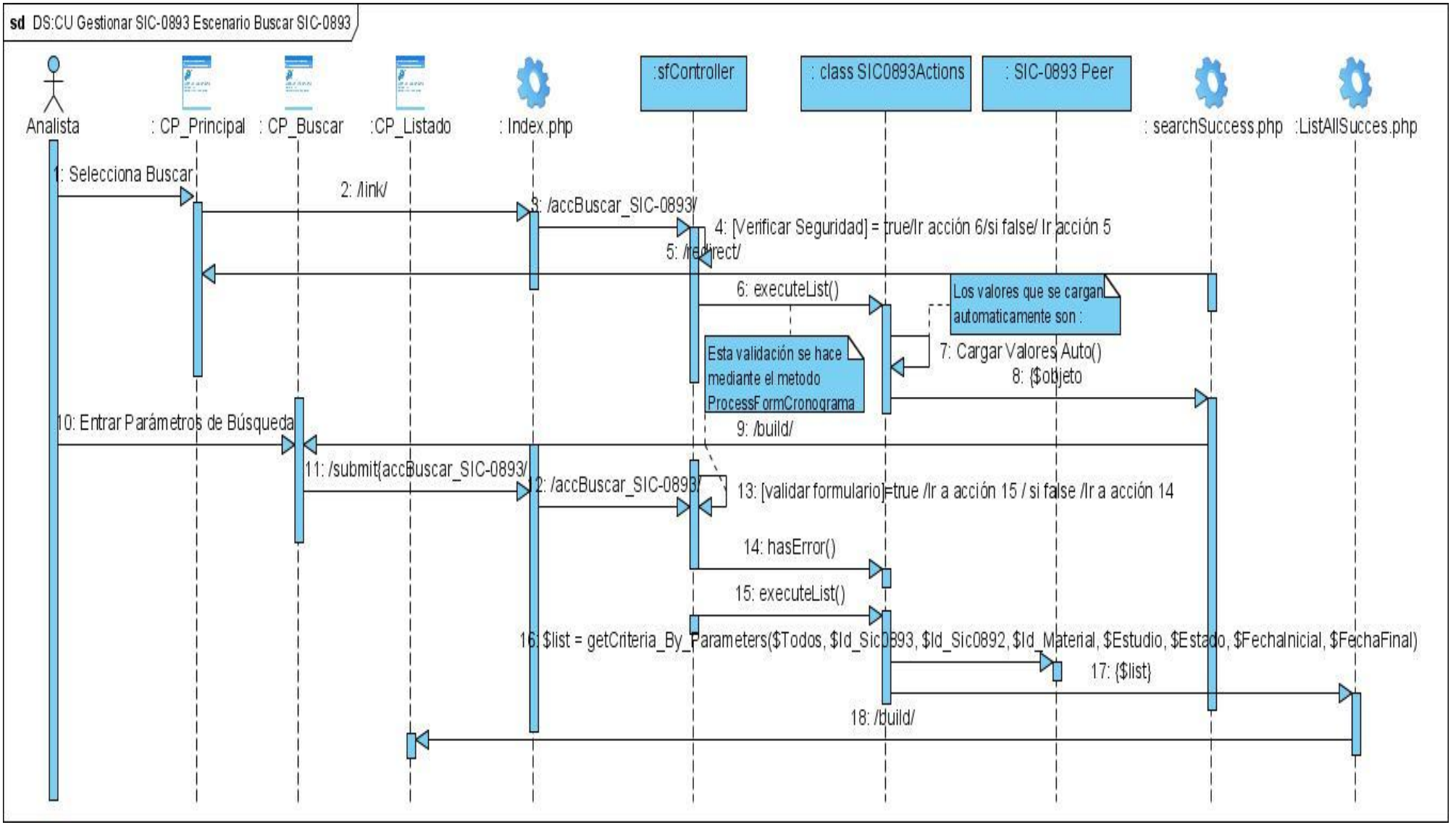

sd DS:CU Gestionar SIC-0893 Escenario Buscar SIC-0893
Analista
: CP_Principal
: CP_Buscar
:CP_Listado
: Index.php
:sfController
: class SIC0893Actions
: SIC-0893 Peer
: searchSuccess.php
:ListAllSucces.php
1: Selecciona Buscar
2: /link/
3: /accBuscar_SIC-0893/
4: [Verificar Seguridad] = true/Ir acción 6/si false/ Ir acción 5
5: /redirect/
6: executeList()
Los valores que se cargan automaticamente son :
7: Cargar Valores Auto()
8: {$objeto
Esta validación se hace mediante el metodo ProcessFormCronograma
9: /build/
10: Entrar Parámetros de Búsqueda
11: /submit(accBuscar_SIC-0893/
12: /accBuscar_SIC-0893/
13: [validar formulario]=true /Ir a acción 15 / si false /Ir a acción 14
14: hasError()
15: executeList()
16: $list = getCriteria_By_Parameters($Todos, $Id_Sic0893, $Id_Sic0892, $Id_Material, $Estudio, $Estado, $FechaInicial, $FechaFinal)
17: {$list}
18: /build/

Figura 3.7 Diagrama de Secuencia: Escenario Buscar 1

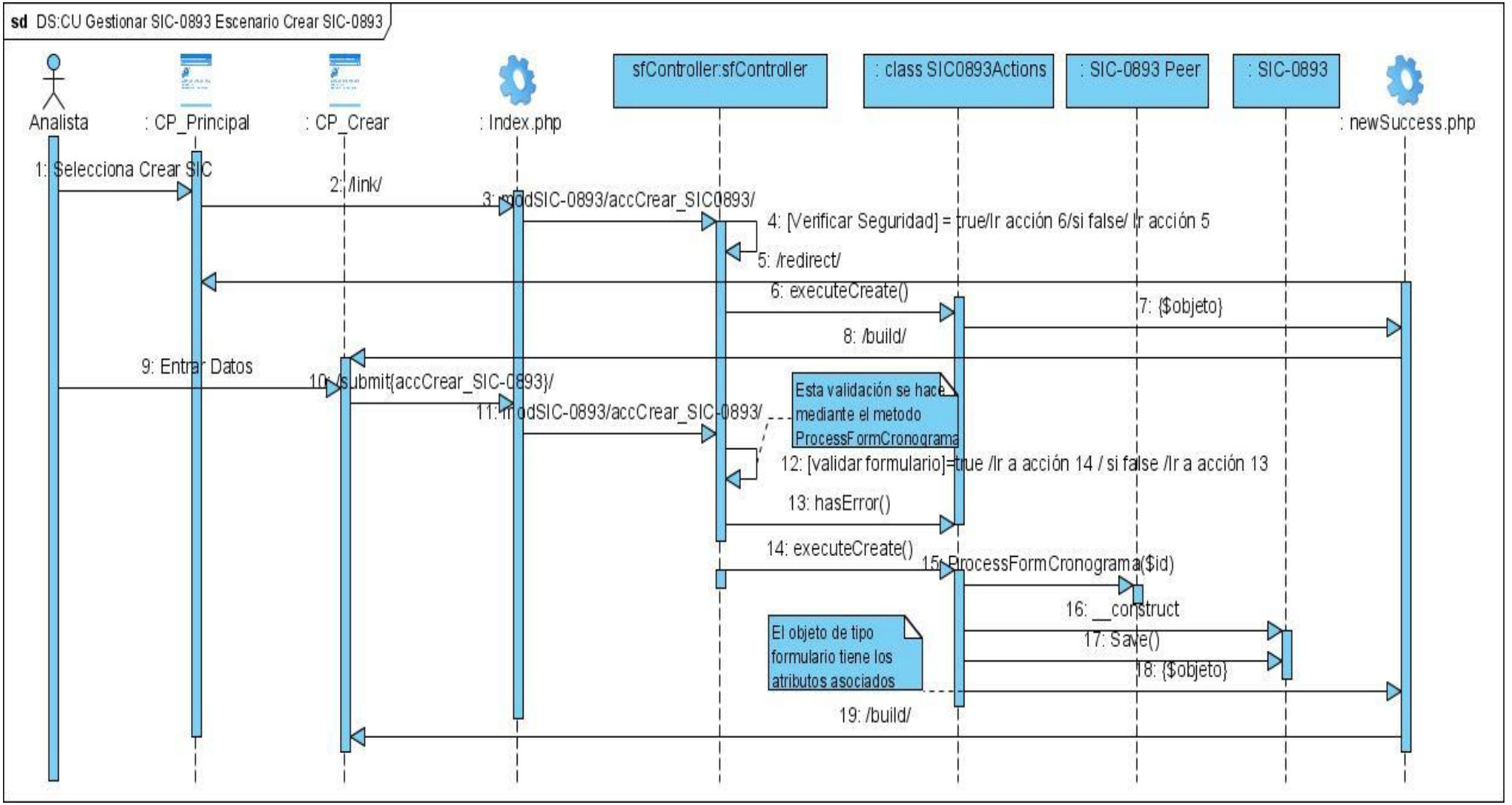


Figura 3.8 Diagrama de Secuencia: Escenario Crear 2

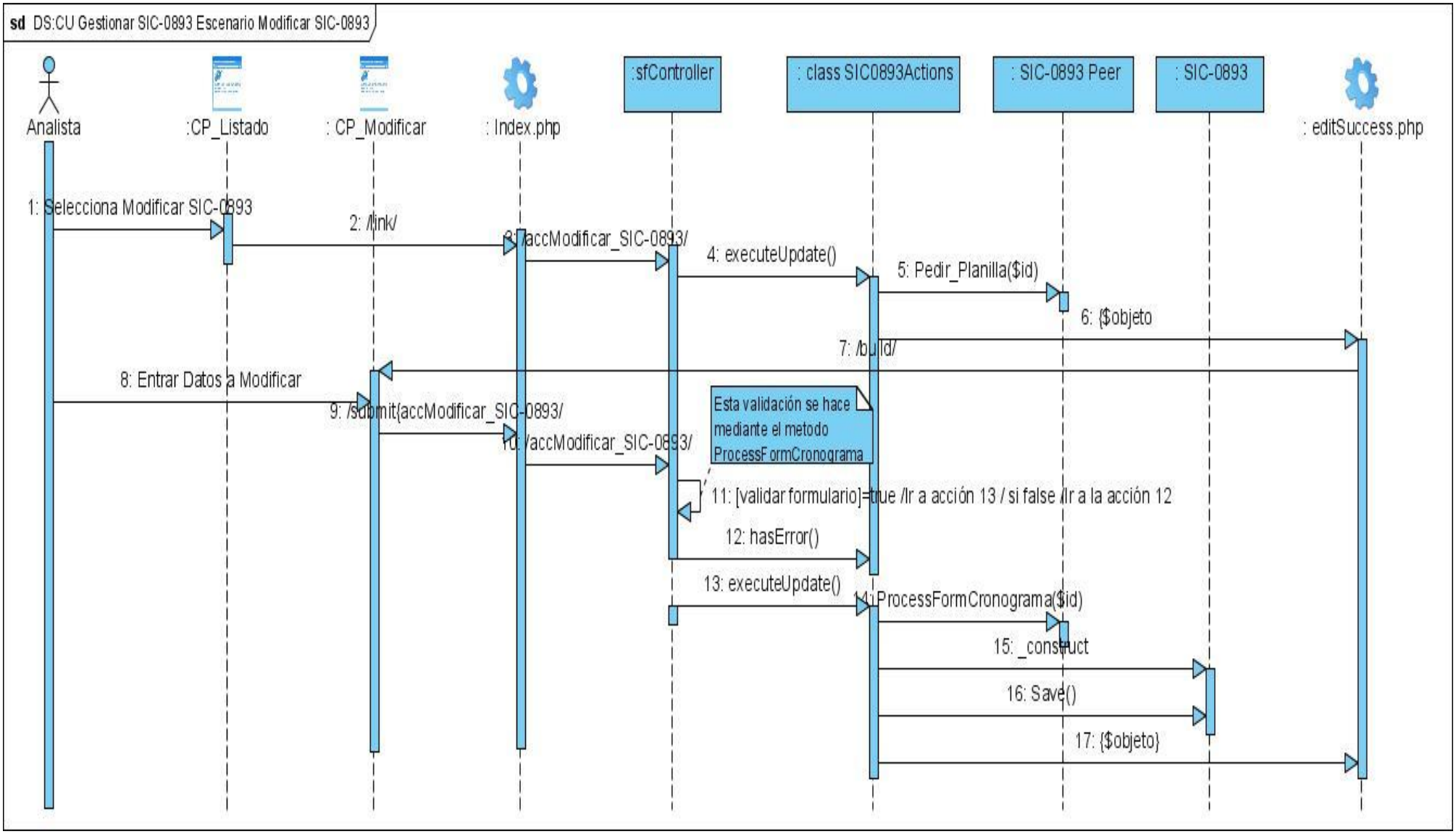

sd DS:CU Gestionar SIC-0893 Escenario Modificar SIC-0893
Analista
:CP_Listado
: CP_Modificar
: Index.php
:sfController
: class SIC0893Actions
: SIC-0893 Peer
: SIC-0893
: editSuccess.php
1: Selecciona Modificar SIC-0893
2: /link/
3: /accModificar_SIC-0893/
4: executeUpdate()
5: Pedir_Planilla($id)
6: {$objeto
7: /build/
8: Entrar Datos a Modificar
9: /submit{accModificar_SIC-0893/
10: /accModificar_SIC-0893/
Esta validación se hace mediante el metodo ProcessFormCronograma
11: [validar formulario]=true /Ir a acción 13 / si false /Ir a la acción 12
12: hasError()
13: executeUpdate()
14: ProcessFormCronograma($id)
15: _construct
16: Save()
17: {$objeto}

Figura 3.9 Diagrama de Secuencia: Escenario Modificar 1

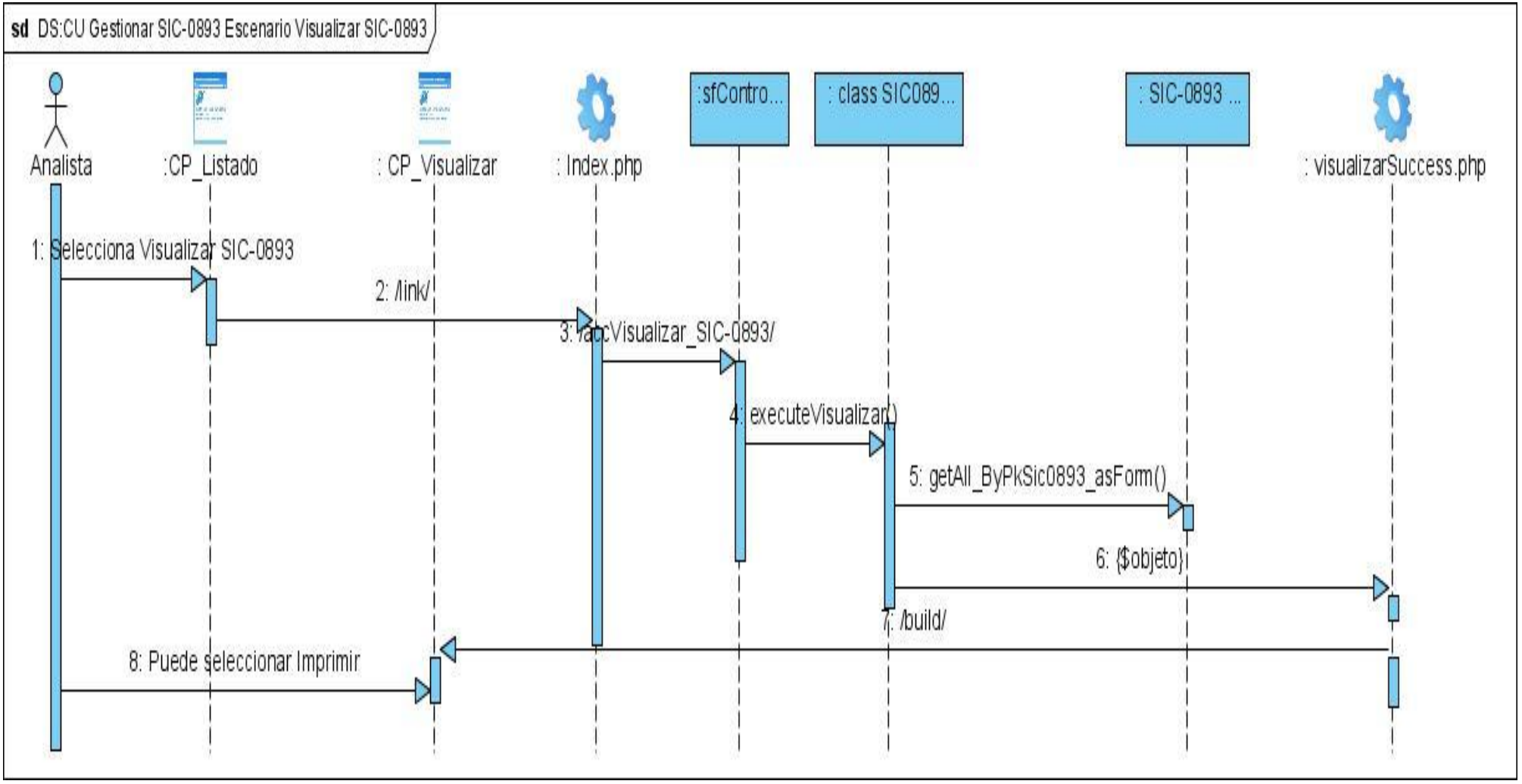


Figura 3.10 Diagrama de Secuencia: Escenario Visualizar 1

3.7. *Clases Persistentes. Diagrama de Clases Persistentes*

"La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Generalmente las clases entidades definidas en el negocio dan origen a las clases persistentes, modelan la información del sistema y el comportamiento asociado de algún fenómeno o concepto." [20]. Las clases persistentes se definen para conocer la información real representada en las tablas de la BD. No todas las clases identificadas en el dominio del análisis son persistentes, porque pueden contener información que no sea necesario almacenar en la BD.

El diagrama de clases se utiliza para modelar la estructura lógica de la BD, con clases representando tablas, y atributos de clases representando columnas. El diagrama de clases persistentes con los atributos correspondiente se muestra en los anexos. (Ver Anexo 7).

3.7.1. Descripción de las Clases Persistentes.

La descripción de las clases persistentes muestra los atributos y el tipo de datos que presenta cada uno. A continuación se muestran algunas tablas. El resto de las tablas se encuentran en el *Expediente de Proyecto* generadas con Visual Paradigm en formato de aplicación Web.

Nombre: SIC-0893	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_SIC-0893	Integer
folio_programa	Integer
No_lote	Integer
fecha_inicio	float
fecha_fabricacion	float
tiempo_total	float
observaciones	string
realizado_por	string
fecha_realizado	float
revisado_por	string
fecha_revisado	float
aprobado_por	string

fecha_aprobado	float
terminado	boolean
Responsabilidad:	
Nombre:	SIC-0893()
Descripción:	Constructor

Tabla 3.2 Descripción de la clase SIC-0893

Nombre: Frecuencia_Muestreo	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_frecuencia	Integer
hora	float
dia	Integer
mes	Integer
ano	Integer
tiempo	float
analisis	String
Responsabilidad:	
Nombre:	Frecuencia_Muestreo()
Descripción:	Constructor

Tabla 3.3 Descripción de la clase Frecuencia_Muestreo

3.8. *Diseño de la BD*

La traducción de un modelo de datos en una base de datos es el punto clave para alcanzar los objetivos de negocio del sistema. El conjunto de información almacenada en las diferentes bases de datos facilita la minería de datos o el descubrimiento de conocimiento.

3.8.1. Modelo de Datos

El modelo de datos se utiliza para describir la representación lógica y física de la información persistente manejada por el sistema. La base de datos fue normalizada hasta la Primera Forma Normal (PFN). El proceso de normalización de bases de datos consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional. La PFN prohíbe:

- Los atributos multivaluados, los atributos compuestos y sus combinaciones.
- Tener un conjunto de valores, una tupla de valores o una combinación de ambas como valor de un atributo para una tupla individual.
- Las “relaciones dentro de relaciones” o “relaciones como atributos de tuplas”.
- Las relaciones anidadas.
- Los valores no atómicos.

El modelo de datos con los atributos de las clases, los tipos de datos y las llaves que se exportan debido a las relaciones existentes se muestra en el Anexo 8.

Conclusiones

En este Capítulo se desarrolló la totalidad de los diagramas de clases del diseño para los veinte casos de uso identificados y solo se le realizaron diagramas de secuencia a cinco casos de uso, plasmando solamente uno en el documento. Además se realizaron los prototipos funcionales para los veinte casos de uso. Se confeccionó el mapa de navegación para garantizar una adecuada navegabilidad por parte del usuario y se obtuvo como una referencia a la arquitectura del sistema la Vista Lógica y de Despliegue.

Se hizo una descripción y fundamentación de la arquitectura de la BD, teniendo en cuenta las necesidades del centro. Se mostró el diagrama de clases persistentes y el modelo de datos; definiendo un total de cuarenta y cinco clases y cuarenta y cinco entidades. Se realizó una breve descripción de cada una de ellas, así como de sus atributos y tipos de datos, facilitando así el entendimiento de los diagramas y de cada una de sus partes.

Capítulo 4
Implementación del Sistema

Introducción

En este Capítulo se realiza el flujo de trabajo Implementación donde se comienza a trabajar con el diseño realizado y se implementa el sistema en términos de componentes, es decir ficheros de código fuente, librerías, ejecutables, scripts, fichero de código binario y otros similares. Se aborda la programación que se realiza partiendo de los requerimientos diseño elaborados. Se presenta el diagrama de componentes correspondiente a cada uno de los casos de uso implementados. Se muestran fragmentos relevantes del código y ejemplos de las validaciones realizadas a esta implementación.

4.1. *Propósitos de la Implementación*

La implementación es el centro durante las iteraciones de construcción, aunque también se lleva a cabo en la fase de elaboración para crear la línea base ejecutable de la arquitectura, y durante la transición para tratar los defectos tardíos.

El Propósito de la implementación es:

Planificar las integraciones de sistema necesarias en cada iteración. Siguiendo un enfoque incremental, lo que da lugar a un sistema que se implementa en una sucesión de pasos pequeños y manejables.

Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue. Esto se basa en las clases activas encontradas durante el diseño.

Implementar las clases y subsistemas encontrados durante el diseño, fundamentalmente las que se implementan como componentes de ficheros que contienen código fuente.

Probar los componentes individualmente e integrarlos posteriormente enlazándolos en uno o más componentes ejecutables, antes de ser enviados para ser integrados y llevar a cabo las comprobaciones del sistema.

4.2. *Modelo de Implementación: diagramas de componentes*

El modelo de Implementación describe como los elementos de diseño, se implementan en términos de componentes, describe como se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y como dependen los componentes unos de otros.

4.2.1. Diagrama de Componentes

Los diagramas de componentes muestran las organizaciones y dependencias lógicas entre los diferentes componentes de software. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

Un Componente no es más que la “Parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces y proporciona la realización de los mismos. Un componente típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios, etc.)”.[21]

En los diagramas de componentes realizados, se muestran como están distribuidos según el patrón arquitectónico Modelo-Vista-Controlador que utiliza Symfony como paradigma en su organización interna.

El componente *sfFrontWebController o Controlador Frontal* maneja todas las peticiones web, siendo el punto de entrada de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL escrita o pinchada por el usuario. El controlador frontal se encarga de despachar las peticiones que le llegan a través de los diferentes formularios correspondientes a las success, lo que implica algo más que detectar la acción que se ejecuta. De hecho, ejecuta el código común a todas las acciones. En pocas palabras el controlador frontal es el encargado de determinar qué combinación de módulo-acción se ejecutará.

El paquete de componentes Controlador, representa todas las acciones. La actions.class.php contiene todos los métodos y operaciones a realizar y, está relacionada con todos los archivos Success.php de la vista. En el paquete de componentes Vista se muestran las posibles acciones a solicitar, las cuales se integran con el layout encargado de ¨decorar¨ las plantillas, siendo este archivo común para todas las páginas de la aplicación y, este a su vez necesita de código javascript para habilitar la navegabilidad a las diferentes páginas de la aplicación, esto se logra, específicamente con el script.js. Las success tienen relación de dependencia con la actions ya que cuando se ejecuta una actions posteriormente se ejecuta la succes asociada a ella, es decir, la succes no existe sino existe la clase actions. La vista utiliza CCS para mostrar los estilos de las diferentes planillas.

Cuando se accede a las diferentes acciones lo primero que se hace es verificar si está activada la seguridad y se verifica si el usuario está autenticado y tiene los permisos correspondientes a la acción que desea realizar. La seguridad está dada por los módulos sfGuard: sfGuardAuth encargado de gestionar la autenticación, sfGuardUser que gestiona los diferentes usuarios creados, sfGuardPermission que gestiona los permisos a cada usuario y sfGuardGroup gestiona todos los grupos de usuarios existentes. Dichos módulos se agrupan en el paquete sfGuard.

En las actions se lanza el mecanismo de validación mediante el método bind(). Los formularios en Symfony están compuestos por Widgets (campos) y Validators (validadores), cada campo debe tener asignado un validador para comprobar los datos que vienen de las vistas y proteger de ataques al sistema. En el paquete Validators están recogidos las clases que permiten las validaciones de los formularios para cada tipo de datos.

El modelo es la capa que contiene la persistencia de los datos, donde se representan los ficheros SIC-0893, SIC0893Peer, y los ficheros SIC-0892 y SIC0892Peer ya que SIC-0893 depende totalmente del SIC-0892. Estos componentes permiten personalizar comportamiento, agregar nuevos métodos

Existen otros componentes del modelo que son generados por Symfony y que no están representados en el diagrama de componentes ya que nunca se modifican, porque cada vez que se genera el modelo, se borran todas las clases. Estos componentes son las clases bases y base Peer. Las clases con nombre Base son las que se generan directamente a partir del esquema. Las clases de objetos propias, SIC 0893 y SIC0893Peer heredan de las clases con nombre Base. Estas clases no se modifican cuando se ejecuta el comando propel-build-model, por lo que son las clases en las que se añaden los métodos. Las clases peer tienen métodos estáticos para trabajar con las tablas de la base

de datos. Proporcionan los medios necesarios para obtener los registros de las tablas. Sus métodos devuelven normalmente un objeto o una colección de objetos de la clase objeto.

Las clases del modelo son construidas por el subsistema Propel de Symfony, mediante el comando propel-build-model, permitiendo el acceso a la base de datos EEMR mediante el mapeo de objetos a bases de datos u ORM. Después que se hace el mapeo de objeto relacional, el modelo le hace las diferentes consultas a la base de datos y devuelve un arreglo con los resultados correspondiente a la actions, los cuales serán mostrado mediante las success.

Estos no son los únicos componentes que utiliza el framework. Symfony genera un conjunto de ficheros a los cuales se le hicieron modificaciones para el desarrollo de la aplicación:

- ✓ *routing.yml:* Se define la ruta del navegador. Almacena las reglas de enrutamiento, que permiten transformar las URL habituales de las aplicaciones web en URL sencillas de recordar.
- ✓ *filters.yml:* La seguridad la ve el módulo SfGuard. los filtros son trozos de código que se ejecutan con cada petición.
- ✓ *security.yml:* Se define si esta activada la seguridad de la aplicación, (is_secure se pone en on).
- ✓ *settings.yml*: contiene las principales opciones de configuración de una aplicación Symfony. Permite especificar si se activa o no la cache. Se habilitan los módulos SfGuard para la seguridad y la autenticación, se definen los módulos que hacen el logueo y la autenticación, el módulo es el sfGuardAuth.
- ✓ *view.yml*: Establece la estructura inicial de la vista por defecto: el nombre del layout, el título de la página, las hojas de estilos y los archivos JavaScript que se incluyen.
- ✓ *ProjectConfiguration.class.php*: Define las rutas de symfony para que el proyecto sea ejecutado sin que esté instalado symfony en la máquina.
- ✓ *database.yml*: indica el nombre de la bases de datos, los datos de acceso. Se definen los parámetros para la conexión a la bases de datos.
- ✓ *schema.yml y propel.ini*: son los archivos de configuración que utiliza Propel para el acceso a los datos. Se utilizan para que las librerías de Propel puedan interactuar con las clases de Symfony y con los datos de la aplicación.
- ✓ *schema.yml*: contiene la representación del modelo de datos relacional del proyecto, se crea mediante el comando Propel:build-schema.
- ✓ *propel.ini*: Se genera de forma automática. Se define la conexión a la base de datos.

En la figura que aparece a continuación se muestra el diagrama de componentes correspondientes al caso de uso “Gestionar Cronograma de EE (SIC-0893)” implementado en el módulo EEMR. Los demás diagramas se encuentran en el *Expediente de Proyecto*.

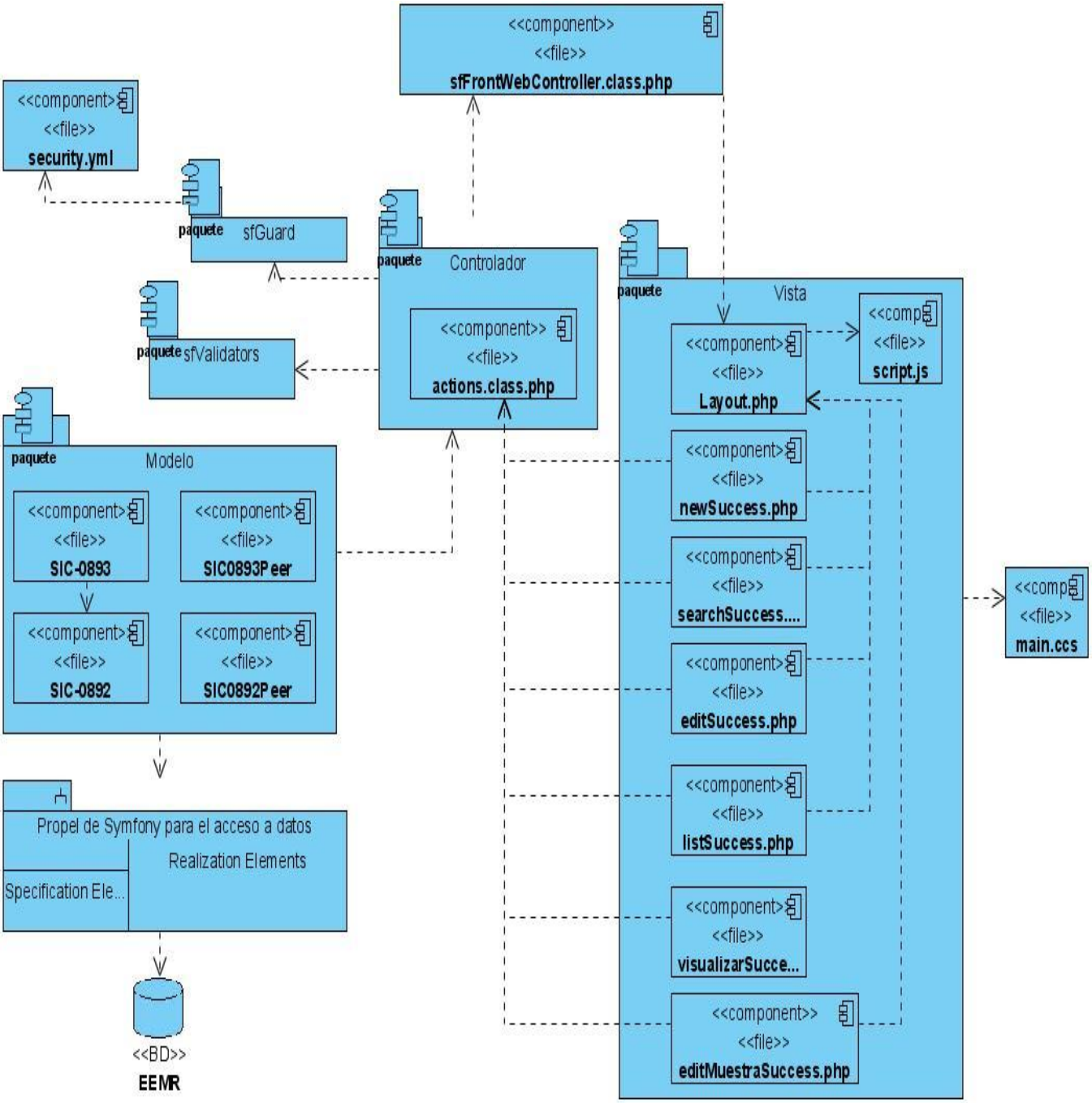


Figura 4.1 Diagrama de Componentes SIC0893 1

4.3. *Vista de Despliegue*

A continuación se actualiza el diagrama de despliegue con la distribución física de los componentes en cada nodo, lo cual representa a la vista de despliegue.

La vista de despliegue ilustra la distribución del procesamiento entre los distintos equipos que conforman la solución, incluyendo los servicios. Existe una traza directa del modelo de implementación, puesto que cada componente físico debe estar almacenado en un nodo. En el nodo que representa el Servidor de Aplicaciones se encuentran todos los componentes que son implementados en la aplicación. Así como los paquetes de sfGuard y sfValidators, los cuales representan la seguridad y la validación respectivamente.

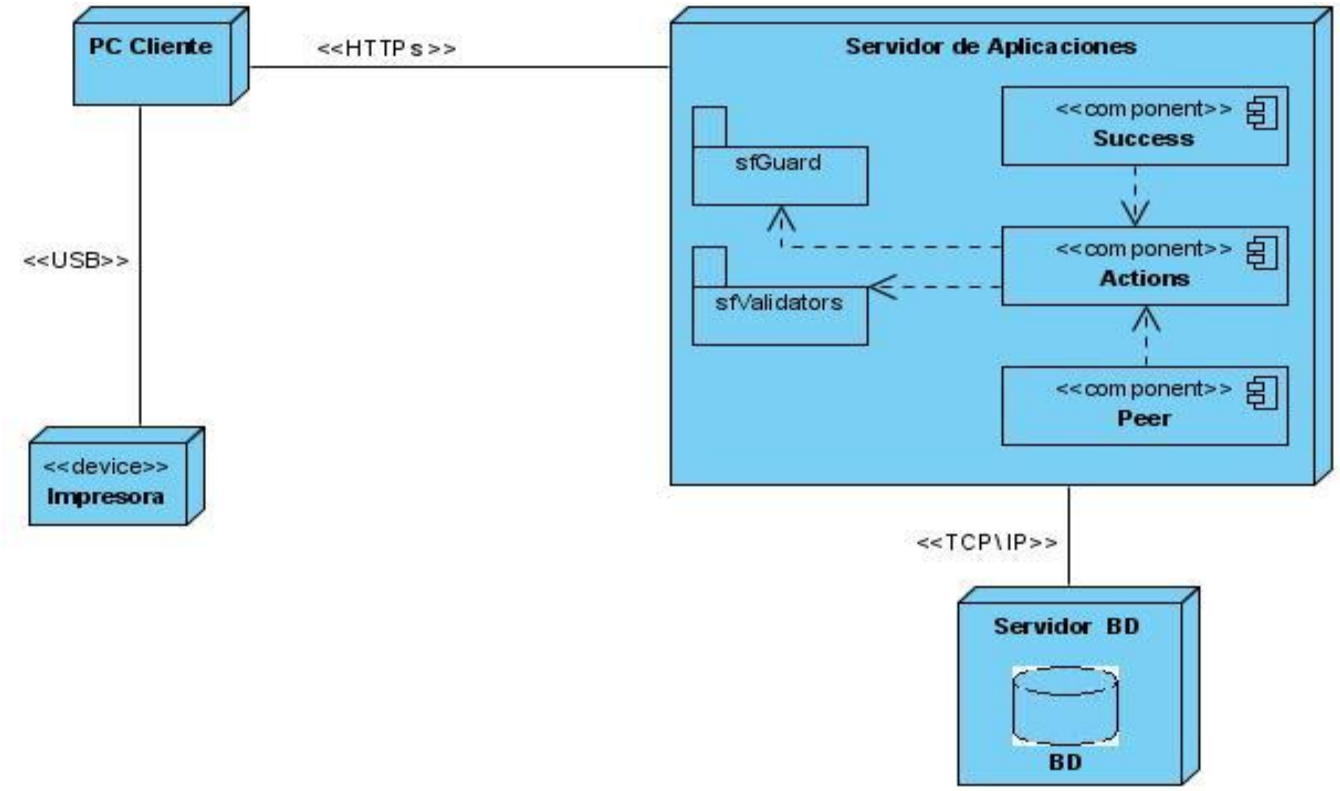


Figura 4.2 Vista de Despliegue 1

4.4. *Ejemplo de código fuente*

A continuación se muestran fragmentos de código que dan cumplimiento al requisito funcional CrearSIC-0893 de la clase SIC0893Actions. Donde la explicación del código se dividió para un mejor entendimiento.

```
public function executeCreate(sfWebRequest $request) {
    $this->forward404Unless($request->isMethod('post'));

    if($this->getUser()->getAttribute('E_Cronograma') == 0) {
        $this->formCronograma = new Sic0893Form();
        $Temp = $this->processFormCronograma($request, $this->formCronograma);

        if($Temp != null)
            $this->formCronograma = new Sic0893Form($Temp);
    }
    else {
        $Temp = $this->processFormCronograma($request, $this->formCronograma =
                                            new Sic0893Form($Cronograma = Sic0893Peer::retrieveByPK(
                                            $this->getUser()->getAttribute('PK_Cronograma'))));
        $this->formCronograma = new Sic0893Form($Temp);
    }
```

En el código anterior, primeramente se redireccionará a una página de error en caso de que no se llame a la acción crear mediante el método submit del formulario de la planilla SIC-0893. Luego, en caso de que el Cronograma no exista se procesará el formulario que viene por el request y si este proceso devuelve null es debido a que existen errores en la validación y se procederá a informar al usuario de ello, en caso contrario se le muestra el formulario con los datos ya guardados y que puedan modificarlos si es preciso. En el caso que el cronograma exista se procesa el formulario para actualizar sus datos.

```
if($this->getUser()->getAttribute('E_Cronograma') == 1) {
    $Lista_Fechas = array();
    $Lista_Tiempos = $request->getParameter('frecuencia_muestreo_tiempo');
    $Lista_Analisis = $request->getParameter('frecuencia_muestreo_analisis');
    $Lista_Unidades = $request->getParameter('frecuencia_muestreo_unidades_analisis');

    $Cont = 0;
    foreach ($Lista_Tiempos as $xT) {
        $Fecha = $request->getParameter('MyFecha'.$Cont.'[year]').'/'.
                 $request->getParameter('MyFecha'.$Cont.'[month]').'/'.
                 $request->getParameter('MyFecha'.$Cont.'[day]').' '.
                 $request->getParameter('MyFecha'.$Cont.'[hour]').':'.
                 $request->getParameter('MyFecha'.$Cont.'[minute]').':'.
                 '00'
        ;

        if($request->getParameter('MyFecha'.$Cont.'[year]') != null &&
           $request->getParameter('MyFecha'.$Cont.'[month]') != null &&
           $request->getParameter('MyFecha'.$Cont.'[day]') != null &&
           $request->getParameter('MyFecha'.$Cont.'[hour]') != null  &&
           $request->getParameter('MyFecha'.$Cont.'[minute]') != null)

           $Lista_Fechas[] = $Fecha;
        else
            $Lista_Fechas[] = '';

        $Cont++;
    }
```

Si el Cronograma fue procesado correctamente se procede a tomar los datos de los análisis correspondientes al estudio que referencia dicho cronograma.

```
        for ($i = 0 ; $i < count($Lista_Fechas) ; $i++) {
            $Objeto = new FrecuenciaMuestreo();
            $Objeto->setTiempo($Lista_Tiempos[$i]);
            $Objeto->setFechaYHora($Lista_Fechas[$i]);
            $Objeto->setAnalisis($Lista_Analisis[$i]);
            $Objeto->setUnidadesAnalisis($Lista_Unidades[$i]);
            $Objeto->setSic0893idSic0893($this->getUser()->getAttribute('PK_Cronograma'));
            $Objeto->save();
        }
    }

    $this->listaMuestras = FrecuenciaMuestreoPeer::getAll_ByPkSic0893_asForm(
                                                    $this->getUser()->getAttribute('PK_Cronograma'));
    $this->formFrecuenciaMuestreo = new FrecuenciaMuestreoForm();
    $this->setTemplate('new');
}
```

Cada análisis asociado al cronograma es guardado con sus respectivos atributos. Finalmente es mandado a la Success las variables necesarias para la correcta visualización de la planilla.

4.5. *Análisis de la seguridad del sistema implementado*

La seguridad del sistema se logra llevando un control del usuario el cual pertenece a diferentes grupos, además de las credenciales que son privilegios de seguridad agrupados bajo un nombre y que permiten organizar la seguridad en grupos. De acuerdo a esto será el nivel de acceso a la aplicación. A continuación se explica lo antes mencionado.

Cuando un usuario trata de acceder a una acción restringida según sus credenciales:
▪ Si el usuario está autenticado y tiene las credenciales apropiadas, entonces la acción se ejecuta.
▪ Si el usuario no está autenticado, es redireccionado a la acción de login.
▪ Si el usuario está autenticado, pero no posee las credenciales apropiadas, será redirigido a la acción segura por defecto.

La autenticación, sinónimo de seguridad, permite limitar el acceso a partes de la aplicación. Significa que los usuarios tendrán que iniciar una sesión, es decir autenticarse para acceder a ciertas áreas. Diferentes usuarios pueden tener diferentes privilegios, diferente autorización. De esta manera se asegura la aplicación para diferentes tipos de usuarios y se administran los privilegios.

La seguridad se ejecuta por los módulos sfGuard, los cuales se encuentran en la carpeta sfGuardPlugin del proyecto. El sfGuardPlugin brinda el modelo (usuario, grupo y objetos de permisos) y los módulos (backend y frontend) para asegurar la aplicación.

Para el frontend el cual se refiere a una aplicación para el usuario solo necesita el modulo sfGuardAuth y para el backend que es para los administradores del sitio, se necesitan los módulos: sfGuardUser, sfGuardGroup, sfGuardPermission.

sfGuardPlugin ha añadido 7 tablas a la Base definidas en '/plugins/sfGuardPlugin/config/schema.yml':

1. **sf_guard_user** - La tabla con todos los usuarios, tiene username , password, etc.
2. **sf_guard_permission** - La tabla con todos los permisos, incluye permissionname y description

3. **sf_guard_user_permission** - La tabla que relaciona n:m ‘user’ y ‘permission’, es decir un usuario puede tener múltiples permisos y un permiso puede ser compartido con múltiples usuarios.
4. **sf_guard_group** - La tabla que define groups
5. **sf_guard_user_group** - La tabla que lista los users de un group, un user puede ser parte de multiples groups
6. **sf_guard_group_permission** - La tabla que relaciona n:m ‘group’ y ‘permission’
7. **sf_guard_remember** - almacena la ‘ip-address’ y ‘remember-key’ de los usuarios

Para asegurar la aplicación se realizaron las siguientes acciones:

- Se habilitó el módulo sfGuardAuth en settings.yml del frontend. El cual se logra removiendo el '#' al inicio de la línea del 'settings.yml'

- Se modificó los módulos *login* y *secure* en settings.yml.

```
all:
    .actions:
      login_module:           sfGuardAuth
      login_action:           signin
      secure_module:          sfGuardAuth
      secure_action:          secure
```

- Se cambió la clase padre en myUser.class.php de la aplicación.

```
    class myUser extends sfGuardSecurityUser
    {
    }
```

- Se aseguran los módulos o toda la aplicación en security.yml

```
    default:
      is_secure: on
```

4.6. *Métodos de validación y manejo de errores de Symfony*

Los formularios de Symfony están formados por campos los cuales se identifican por un nombre único. Para mostrar el formulario al usuario, se asocia un tipo de widget a cada campo, cada campo debe tener asignado un validador para comprobar los datos que vienen de las vistas como se menciona en epígrafes anteriores.

La validación de los campos se realiza mediante objetos que heredan de la clase sfValidatorBase. Para validar el formulario de contacto, se define un objeto validador para cada uno de los campos: material_referencia, folio_programa, no_lote, estado, estudio, rando_de_fechas. A continuación se muestra cómo crear estos validadores en la clase del formulario utilizando el método setValidators().

```
class SearchSIC0893 extends sfForm
{
    protected static $estado = array(0 => 'No Terminado', 1 =>'Terminado');
    protected static $estudio = array('Real' => 'Real' , 'Acelerado' => 'Acelerado');

    public function configure() {
        $this->setWidgets(array('material_referencia' => new sfWidgetFormPropelChoice(array('model'
                                                        => 'MaterialReferencia', 'add_empty' => true)),
                                'folio_programa' => new sfWidgetFormPropelChoice(array('model'
                                                        => 'Sic0892', 'add_empty' => true)),
                                'no_lote' => new sfWidgetFormPropelChoice(array('model'
                                                        => 'Sic0893', 'add_empty' => true)),
                                'estado' => new sfWidgetFormChoice(array('choices' => self::$estado,
                                                        'expanded' => true)),
                                'estudio' => new sfWidgetFormChoice(array('choices' => self::$estudio,
                                                        'expanded' => true)),
                                'rango_de_fechas' => new sfWidgetFormDateRange(array('from_date'
                                                => new sfWidgetFormDate(), 'to_date'
                                                => new sfWidgetFormDate(), 'template'
                                                => '%from_date% <br/>   %to_date%') ),
        ));
```

```
        $this->setValidators(array('material_referencia' => new sfValidatorPropelChoice(array('required'
                                        => false, 'model' => 'MaterialReferencia', 'column' => 'id_material')),
                                'folio_programa' => new sfValidatorPropelChoice(array('required'
                                                => false, 'model' => 'SicO893', 'column' => 'id_sicO892')),
                                'no_lote' => new sfValidatorPropelChoice(array('required'
                                                => false, 'model' => 'SicO893', 'column' => 'id_sic_O893')),
                                'estudio' => new sfValidatorChoice(array('required' => false,
                                                        'choices' => array_keys(self::$estudio))),
                                'rango_de_fechas' => new sfValidatorDateRange(array('from_date'
                                                => new sfValidatorDate(array('required' => false)),
                                        'to_date' => new sfValidatorDate(array('required' => false))))
        ));

        $this->widgetSchema->setNameFormat('searchSICO893[%s]');
        $this->validatorSchema->setOption('allow_extra_fields', true);
        $this->validatorSchema->setOption('filter_extra_fields', false);
    }
```

El código anterior utiliza tres validadores diferentes:

- sfValidatorPropelChoice: valida que sea seleccionado alguno de los valores que se encuentra en los select.
- sfValidatorDateRange: valida un rango de fechas.
- sfValidatorChoice: valida que el valor se encuentra entre una lista de valores predefinidos

Los validadores aceptan como primer argumento una lista de opciones. Al igual que los Widgets, a pesar de su nombre algunas opciones son opcionales y otras obligatorias. El validador sfValidatorChoice por ejemplo dispone de una opción obligatoria llamada choices. Todos los validadores pueden establecer las opciones required y trim, que definen sus valores por defecto en la clase sfValidatorBase: donde si la opción required tiene como valor por defecto true indica que es obligatorio rellenar este campo. Si la opción trim tiene como defecto false Indica si se deben eliminar los espacios en blanco del principio y del final antes de validar el valor del campo

Para que se realice el mecanismo de validación es necesario realizar el método bind, encargado de hacer coincidir los campos que llegan a través del request con los campos del formulario.

```
protected function processFormCronograma(sfWebRequest $request, sfForm $form) {
    $sic0893 = null;
    $form->bind($request->getParameter($form->getName()), $request->getFiles($form->getName()));

    if ($form->isValid()) {
        $sic0893 = $form->save();
        $this->getUser()->setAttribute('E_Cronograma', 1);
        $this->getUser()->setAttribute('PK_Cronograma', $sic0893->getIdSic0893());
    }

    return $sic0893;
}
```

Una vez asociado, el formulario se puede validar mediante el método isValid(). Si el valor devuelto por el método es true, el formulario es válido. El proceso de validación añade los mensajes de error al formulario para que puedan ser mostrados al usuario de forma global mediante el método renderGlobalErrors() o según campos específicos mediante el método renderError() de cada widget en particular.

```
<tr>
    <th height="33"><?php echo $formCronograma['tipo_estudio']->renderLabel() ?></th>
    <td align="center" id="ss">
        <?php echo $formCronograma['tipo_estudio']->renderError() ?>
        <?php echo $formCronograma['tipo_estudio'] ?>
    </td>

    <th width="147"><?php echo $formCronograma['fecha_inicio']->renderLabel() ?></th>
    <td width="163" align="center">
      <?php echo $formCronograma['fecha_inicio']->renderError() ?>
      <?php echo $formCronograma['fecha_inicio'] ?>
    </td>

    <th> <?php echo $formCronograma['temperatura']->renderLabel() ?> </th>
    <td align="center" id="ssss">
        <?php echo $formCronograma['temperatura']->renderError() ?>
        <?php echo $formCronograma['temperatura'] ?>
    </td>
</tr>
```

4.7. *Pruebas Unitarias a la Aplicación*

La automatización de pruebas es uno de los mayores avances en la programación desde la invención de la orientación a objetos. El objetivo de las mismas es la detección de defectos en el software asegurando así la calidad de la aplicación.

Los conjuntos de casos de prueba garantizan que la aplicación hace lo que debe hacer. Las pruebas automatizadas permiten garantizar que los cambios no introducen incompatibilidades en el funcionamiento de la aplicación. Un buen conjunto de pruebas muestra la salida que produce la aplicación para una serie de entradas de prueba, por lo que es suficiente para entender el propósito de cada método.

Symfony aplica este principio a su propio código. El código interno del framework se valida mediante la automatización de pruebas. Las pruebas unitarias aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular, se encargan de un único caso cada vez, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto. El objetivo de las pruebas de unidad o unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas.

Symfony incluye el framework LIME para automatizar las pruebas.

El framework de pruebas Lime

"Lime proporciona el soporte para las pruebas unitarias, es más eficiente que otros frameworks de pruebas de PHP y tiene las siguientes ventajas:" [15]

- Ejecuta los archivos de prueba en un entorno independiente para evitar interferencias entre las diferentes pruebas. No todos los frameworks de pruebas garantizan un entorno de ejecución "limpio" para cada prueba.
- Las pruebas de Lime son fáciles de leer y sus resultados también lo son. En los sistemas operativos que lo soportan, los resultados de Lime utilizan diferentes colores para mostrar de forma clara la información más importante.

• Symfony utiliza Lime para sus propias pruebas y su “regression testing”, por lo que el código fuente de Symfony incluye muchos ejemplos reales de pruebas unitarias y funcionales.

• El núcleo de Lime se valida mediante pruebas unitarias.

• Está escrito con PHP, es muy rápido y está bien diseñado internamente. Consta únicamente de un archivo, llamado lime.php, y no tiene ninguna dependencia.

Cada prueba unitaria consiste en una llamada a un método de la instancia de *lime_test*. Para ejecutar el conjunto de pruebas, se utiliza la tarea *test-unit* desde la línea de comandos. El resultado mostrado en consola es fácil de entender lo que permite localizar las pruebas que han fallado y las que se han ejecutado correctamente.

La tabla que se presenta a continuación representa las pruebas realizadas a algunas funcionalidades implementadas al módulo SIC0893. En las mismas se recogen los resultados obtenidos después de aplicar aquellos métodos que ofrece **lime** para comprobar el correcto funcionamiento de las funcionalidades a comprobar. En el primer campo (Método) corresponde al nombre de la funcionalidad que será comprobada, es decir el nombre del método; el segundo (Método de Prueba) es el método de la clase **lime_test** utilizado, el tercer campo (Recibe) contiene los parámetros que recibe la funcionalidad al ser comprobada, el cuarto campo (Esperado) es el resultado que se espera que devuelva el método de prueba y el quinto campo (Obtenido) es el resultado real obtenido. El resultado es satisfactorio si coinciden los resultados de las pruebas con los esperados, o sea, los campos Esperado y Obtenido.

Método	**Método de Prueba**	**Recibe**	**Esperado**	**Obtenido**
get_NextNoLote()	is	2	Ok	OK
get_All_SIC0893_ByIDPrograma($IdPrograma)	is_deeply	array()	Ok	Ok
contieneFrecuenciaMuestreo($idAnalisis)	is	true	Ok	Ok
contieneFrecuenciaMuestreo($idAnalisis)	cmp_ok	‘==’, true	OK	OK
get_IdX()	is	52	Ok	Ok
__toString()	is	1	Ok	Ok
__toString()	cmp_ok	1	Ok	Ok

En el directorio test/unit de la aplicación se encuentran los archivos que responden a las pruebas realizadas a los módulos de la aplicación. Cada archivo comienza con el nombre del módulo al que se le ha realizado la prueba y termina con Test.php.

4.8. *Prototipos Funcionales.*

A continuación se muestran el prototipo funcional "Crear SIC-0893" obtenido como resultado del CU Cronograma de EE (SIC-0893) implementado en el módulo EEMR.

Figura 4.3 Prototipo Funcional "Crear SIC-0893" 1

Conclusiones

En este Capítulo se realizaron los diagramas de componentes de los cinco casos de uso a los que se les viene dando seguimiento desde el diseño, desarrollándose su implementación. Se muestran fragmentos de código del caso de uso *"Gestionar Cronograma de EE (SIC-0893)",* se hace una breve descripción del manejo de la seguridad en el framework de desarrollo web Symfony, se especifican algunas validaciones efectuadas a esta implementación y se muestran los prototipos funcionales como resultado de la implementación realizada.

Conclusiones Generales

Como resultado de la investigación realizada se puede concluir con los siguientes aspectos:

- La utilización de RUP como proceso de desarrollo de software permitió estructurar el trabajo de forma organizada en sus diferentes etapas. El uso de UML resultó de apoyo para una mejor comprensión de los diagramas realizados.
- La arquitectura Modelo-Vista-Controlador utilizada por el framework Symfony facilitó estructurar el diseño y la implementación de la aplicación Web.
- Se realizó un estudio de los procesos que se desarrollan en el Grupo EEMR lo que permitió identificar los requerimientos del sistema. Se diseñaron todas las clases para este módulo del LIMS, utilizando como herramienta de modelado el Visual Paradigm for UML.
- Se diseñó el Modelo de Datos para la bases de datos EEMR, el cual consta de cuarenta y cinco tablas, normalizadas en la Primera Forma Normal. Para realizar este diseño se seleccionó el modelo relacional, y como SGBD PostgreSQL. Además, se realizaron pruebas para la entrada de datos a la BD a través de la utilización de la herramienta SQL Manager 2007 for PostgreSQL, así como la implementación de consultas, para verificar el funcionamiento de dichas acciones, obteniendo como resultado el cumplimiento de todos los requisitos funcionales de los casos de usos a implementar.
- Fueron implementados treinta y ocho requisitos funcionales, agrupados en cinco casos de uso. Se obtuvo un diagrama de componentes por cada caso de uso.

Por todo lo antes expuesto se concluye que todos los objetivos enunciados en la investigación se cumplieron satisfactoriamente.

Recomendaciones

- Se use RUP como guía de desarrollo de software y como lenguaje de modelado UML, para modelar los artefactos correspondientes.
- Se use para modelar los artefactos de los demás flujos de trabajo la herramienta CASE Visual Paradigm pues este es un software libre.
- Se realice una activa vinculación de los directivos de Calidad con los desarrolladores, para completar el levantamiento de los requerimientos funcionales y la realización del sistema.
- Dar continuidad al presente trabajo y culminar con la implementación de los restantes casos de uso.
- Realizar las pruebas de unidad una vez que se concluya con la implementación de todos los CU.
- Lograr la integración con el resto de los módulos del LIMS de Calidad del CIGB.
- Una vez desarrollado en Cuba un producto de este tipo, pudiera ser extensible a otros centros del Polo Científico y en general a entidades que lo requieran para mejorar y agilizar la calidad de sus productos.

Referencias Bibliográficas

1. **Laurencio, Elennis Díaz y Cervantes, Liusmila Nieto.** *LIMS de Calidad del Centro de Ingeniería Genética yBiotecnología: Análisis de la Sección de Mejoramiento de la Calidad y del Grupo de Desarrollo.* Cuidad de la Habana : s.n., 2007.

2. **Quintana, Yadira Sulmainiz Tamarit y Garcia, Yalina Lamas.** *Sistema para la Gestión de la Información de los Laboratorios de la Dirección de Calidad del Centro de Ingeniería Génetica y Biotecnología: Análisis y Diseño del módulo Análisis Químico.* Ciudad de la Habana : s.n., 2008.

3. **México, Grupo BSI.** BSI. [En línea] 2009. [Citado el: 2 de Marzo de 2009.] http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/.

4. Entorno Virtual de Aprendizaje. Conferencia 1 Introducción al proceso de desarrollo de software. Curso 2007-2008. *http://telefomacion.uci.cu.* [En línea] [Citado el: 10 de Febrero de 2009.] http://teleformacion.uci.cu/mod/resource/view.php?id=11402.

5- [En línea] [Citado el: 10 de Febrero de 2009.] http://www.dsic.upv.es/asignaturas/facultad/lsi/doc/IntroduccionProcesoSW.doc

6. Entorno Virtual de Aprendizaje. Conferencia I de Ingenieria de Software I. Curso 2007-2008. *http://teleformación.uci.cu.* [En línea] [Citado el: 3 de Marzo de 2009.] http://teleformacion.uci.cu/mod/resource/view.php?id=11402.

7. Entorno Virtual de Aprendizaje. Conferencia 4 de Ingenieria de Software I. Curso 2007-2008. *http://teleformación.uci.cu.* [En línea] [Citado el: 4 de Marzo de 2009.] http://teleformacion.uci.cu/mod/resource/view.php?id=12103.

8. Entorno Virtual de Aprendizaje. Conferencia 7 de Ingenieria de Software I. Curso 2007-2008. *http://teleformación.uci.cu.* [Online] [Cited: Marzo 4, 2009.] http://teleformacion.uci.cu/course/view.php?id=102.

9. Entorno Virtual de Aprendizaje. Conferencia 2: Arquitectura de Patrones. Ingenieria de Software II. Curso 2007-2008. *http://teleformación.* [Online] [Cited: Febrero 25, 2009.] http://teleformacion.uci.cu/mod/resource/view.php?id=14077.

10. cyta.com.ar. *Herramientas CASE.* [En línea] [Citado el: 20 de Febrero de 2009.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.

11. ines.org.es. *Proyecto Vulcano. Forja de proyectos software de calidad. OSMRT.* [En línea] http://www.ines.org.es/vulcano/wp-content/uploads/2007/09/d6-estudio-de-herramientas-de-certificacion-tid.pdf.

12. **García, Lic. Rosa María Mato.** *Diseño de Bases de Datos. http://teleformacion.uci.cu/mod/resource/view.php?id=3362*. 1999.

13-. PostgreSql. [En línea] [Citado el: 25 de Marzo de 2009.] http://www.postgresql.org/about/press/features83.

14. ciberaulalinux . *Una Introduccióna a Apache.* [En línea] [Citado el: 24 de Marzo de 2009.] http://linux.ciberaula.com/articulo/linux_apache_intro/.

15. **Potencier, François Zaninotto y Fabien.** librosweb.es. *Symfony, la guia definitiva.* [En línea] [Citado el: 24 de Enero de 2009.] http://www.librosweb.es/symfony_1_2/. ISBN-13: 978-1590597866.

16. **González, Loismarx Peña y Achang, José Rafael Viera.** *LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Implementación del Módulo de Análisis Químico.* Ciudad de la Habana : s.n., 2008.

17. MIDE, Metrología Integral y Desarrollo, S.A. [Online] Abril 25, 2008. [Cited: Abril 24, 2008.] http://www.midelab.com.mx/notas_tec/validacion_y_verificacion.pdf.

18. **Almaenares, Alieski Sarmiento y Díaz, Elian Cutiño.** *LIMS de Calidad del Centro de Ingeniería Genética yBiotecnología: Análisis del Grupo de Recepción de Muestras y Manipulación de Expedientes.* Cuidad de la Habana : s.n., 2006.

19. **Marca, Hugo Michael Huallpara y Quisbert, Nancy Susana Limachi** Trabajo de Investigación y Exposisción. *"Diagrama de Despliegue"* [En línea] [Citado el: 16 de Abril de 2009.] http://virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc

20. **González, Dra. Anaisa Hernández.** Computación y Sistema. Um metodo para el Diseño de la Base de Datos a partir del Modelo Orientado a Objetos. *redalyc.* [Online] 2004. [Cited: Abril 20, 2009.] http://redalyc.uaemex.mx/redalyc/pdf/615/61570402.pdf. ISSN(Versión Impresa): 1405-5546.

21. Entorno Virtual de Aprendizaje. Conferencia 4 de Ingenieria de Software I. Curso 2007-2008. *http://teleformación.uci.cu.* [En línea] [Citado el: 4 de Marzo de 2009.] http://teleformacion.uci.cu/mod/resource/view.php?id=12103.

Bibliografía

1. [En línea] [Citado el: 24 de Marzo de 2009.] http://sophia.javeriana.edu.co/~lcdiaz/ADOO2006-3/grasp_cpaternostro-lvargas-jviafara.pdf.

2. cibernetia. Conceptos básicos del servidor web. [Online] http://www.cibernetia.com/manuales/instalacion_servidor_web/1_conceptos_basicos.php.

3. cyta.com.ar. *Herramientas CASE.* [En línea] [Citado el: 20 de Febrero de 2009.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.

4. Entorno Virtual de Aprendizaje. Conferencia 2: Arquitectura de Patrones. Ingenieria de Software II. Curso 2007-2008. *http://teleformación.* [Online] [Cited: Febrero 25, 2009.] http://teleformacion.uci.cu/mod/resource/view.php?id=14077.

5. Entorno Virtual de Aprendizaje. Conferencia 4 de Ingenieria de Software I. Curso 2007-2008. *http://teleformación.uci.cu.* [En línea] [Citado el: 4 de Marzo de 2009.] http://teleformacion.uci.cu/mod/resource/view.php?id=12103.

6. Entorno Virtual de Aprendizaje. Conferencia 7 de Ingenieria de Software I. Curso 2007-2008. *http://teleformación.uci.cu.* [Online] [Cited: Marzo 4, 2009.] http://teleformacion.uci.cu/course/view.php?id=102.

7. Entorno Virtual de Aprendizaje. Conferencia de la semana 7 de Ingenieria de Software II. Curso 2007-2008. *http://teleformación.uci.cu.* [En línea] [Citado el: 5 de Marzo de 2009.] http://teleformacion.uci.cu/mod/resource/view.php?id=14094.

8. Entorno Virtual de Aprendizaje. Conferencia I de Ingenieria de Software I. Curso 2007-2008. *http://teleformación.uci.cu.* [En línea] [Citado el: 3 de Marzo de 2009.] http://teleformacion.uci.cu/mod/resource/view.php?id=11402.\

9. ibercom.com. [En línea] https://www.ibercom.com/soporte/index.php?_m=knowledgebase&_a=viewarticle&kbarticleid=30.

10. ines.org.es. *Proyecto Vulcano. Forja de proyectos software de calidad. OSMRT.* [En línea] http://www.ines.org.es/vulcano/wp-content/uploads/2007/09/d6-estudio-de-herramientas-de-certificacion-tid.pdf.

11. Kynetia. SOFTWARE FOR BUSINESS SOLUTIONS. [En línea] [Citado el: 10 de Marzo de 2009.] http://www.kynetia.es/calidad/rup-rational-unified-process.html.

13. Patrones GRASP. [En línea] [Citado el: 16 de Febrero de 2009.] http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/exposiciones2005B/polimorfismoFabricacionPura_G01/.

14. **Alvarez, Sara.** desarrolloweb.com. *Modelo de Bases de Datos.* [En línea] 14 de Agosto de 2007. [Citado el: 26 de Marzo de 2009.] http://www.desarrolloweb.com/articulos/modelos-base-datos.html.

15. **Astudillo, Marcello Visconti y Hernán.** inf.utfsm.cl. *Fundamentos de Ingenieria de Software.* [Online] [Cited: Marzo 24, 2009.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/16-PatronesGRASP.pdf\.

16. **Business Quality & Technology, S.L.** www.quaass.com. Business Quality & Technology. [En línea] 2008. [Citado el: 2 de Marzo de 2009.] http://www.quaass.com/LIMS.

17. **Capmany, Maria Aurèlia.** Addlink. Software Científico. [En línea] [Citado el: 6 de Marzo de 2009.] http://www.addlink.es/productos.asp?pid=473.

18. **García, Lic. Rosa María Mato.** *Diseño de Bases de Datos. http://teleformacion.uci.cu/mod/resource/view.php?id=3362*. 1999.

19. **Gutierrez, Jorge A. Saavedra.** El Mundo Informático<<Patrones GRASP>>. [En línea] [Citado el: 24 de Marzo de 2009.] http://jorgesaavedra.wordpress.com/category/patrones-grasp/.

20. **México, Grupo BSI.** BSI. [En línea] 2009. [Citado el: 2 de Marzo de 2009.] http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/.

21. **Pressman., Roger S.** *Ingeniería de Software. Un enfoque Práctico.*

22. **Rabaix, Thomas.** groups.google.com. *symfony users.* [Online] Mayo 18, 2008. [Cited: Abril 17, 2009.] http://groups.google.com/group/symfony-users/msg/cd94d2ddb2057355?pli=1.

23. **Viklund, Andreas.** stackframe.net. *Patrones de Diseño.* [Online] Octubre 31, 2008. [Cited: Abril 17, 2009.] http://www.stackframe.net/es/content/10-2008/patrones-de-diseno-abstract-factory-pattern.

24. masadelante.com. [En línea] [Citado el: 24 de Marzo de 2009.] http://www.masadelante.com/faq-servidor-web.htm.

25. ciberaulalinux . *Una Introduccióna a Apache.* [En línea] [Citado el: 24 de Marzo de 2009.] http://linux.ciberaula.com/articulo/linux_apache_intro/.

26. **Pastor, Javier.** The INQUIRER.es. *PostgreSQL 8.3 disponible.* [En línea] 5 de Febrero de 2008. [Citado el: 24 de Marzo de 2009.] http://www.theinquirer.es/2008/02/05/postgresql_83_disponible.html.

27. PostgreSql. [En línea] [Citado el: 25 de Marzo de 2009.] http://www.postgresql.org/about/press/features83.

29. **Potencier, Fabien.** symfony. *Symfony tutorial What's new in Symfony 1.2?* [Online] [Cited: Abril 6, 2009.] http://www.symfony-project.org/tutorial/1_2/whats-new.

30. **Potencier, François Zaninotto y Fabien.** librosweb.es. *Symfony, la guia definitiva.* [En línea] [Citado el: 24 de Enero de 2009.] http://www.librosweb.es/symfony_1_2/. ISBN-13: 978-1590597866.

31. **JACOBSON, IVAR, BOOCH, GRADY y RUMBAUGH, JAMES.** *El Proceso Unificado de Desarrollo de Software.* Madrid : PEARSON EDUCACION, S.A, 2000. ISBN:84-7829-036-2.

32. CIGB. *Centro de Ingeniería Genética y Biotecnología de Cuba.* [En línea] [Citado el: 10 de Febrero de 2009.] http://www.cigb.edu.cu/index.php?option=com_content&task=view&id=25&Itemid=81.

33. **Mendoza, Gonzalo Mena.** *Procesos de la Ingeniería de Requerimientos.* [Multimedia] Santiago de Querétaro : s.n., 2005.

34. siona.udea.edu.co. *¿Arquitectura de Software?* [En línea] [Citado el: 20 de Abril de 2009.] http://siona.udea.edu.co/~aoviedo/Arquitectura%20de%20Software/Arquitectura%20de%20Software.htm.

35. espanol.finance.yahoo.com. [En línea] [Citado el: 23 de Abril de 2009.] http://espanol.finance.yahoo.com/q/hp?s=LIMS&a=04&b=24&c=2007&d=03&e=24&f=2009&g=m.

36. **Laurencio, Elennis Díaz y Cervantes, Liusmila Nieto.** *LIMS de Calidad del Centro de Ingeniería Genética yBiotecnología: Análisis de la Sección de Mejoramiento de la Calidad y del Grupo de Desarrollo.* Cuidad de la Habana : s.n., 2007.

37. **Mesa, Rosayda Valiente y Pérez, Idelsis Castillo.** *LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Desarrollo de la Base de Datos del módulo Sección de Mejoramiento de la Calidad.* Ciudad de la Habna : s.n., 2008.

38. **González, Loismarx Peña y Achang, José Rafael Viera.** *LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Implementación del Módulo de Análisis Químico.* Ciudad de la Habana : s.n., 2008.

39. *Ayuda Extendida Rational Unified Process(RUP).* 2003.

40. **Quintana, Yadira Sulmainiz Tamarit y Garcia, Yalina Lamas.** *Sistema para la Gestión de la Información de los Laboratorios de la Dirección de Calidad del Centro de Ingeniería Génetica y Biotecnología: Análisis y Diseño del módulo Análisis.* Ciudad de la Habana : s.n., 2008.

41. Entorno Virtual de Aprendizaje. Conferencia 1 Introducción al proceso de desarrollo de software. Curso 2007-2008. *http://telefomacion.uci.cu.* [Online] [Cited: Febrero 10, 2009.] http://teleformacion.uci.cu/mod/resource/view.php?id=11402.

42. MIDE,Metrología Integral y Desarrollo, S.A. [Online] Abril 25, 2008. [Cited: Abril 24, 2008.] http://www.midelab.com.mx/notas_tec/validacion_y_verificacion.pdf.

43. BIKA Jabs System. [En línea 2005-2009] [Citado el: 22 de Mayo de 2009.] http://www.bikalabs.com/.

44. **Roman Konertz, PD Dr. Ludwig Eichinger and Dr. Budi Tunggal.** *Open LIMS.* [En línea] [Citado el: 22 de Mayo de 2009.] http://www.open-lims.org/.

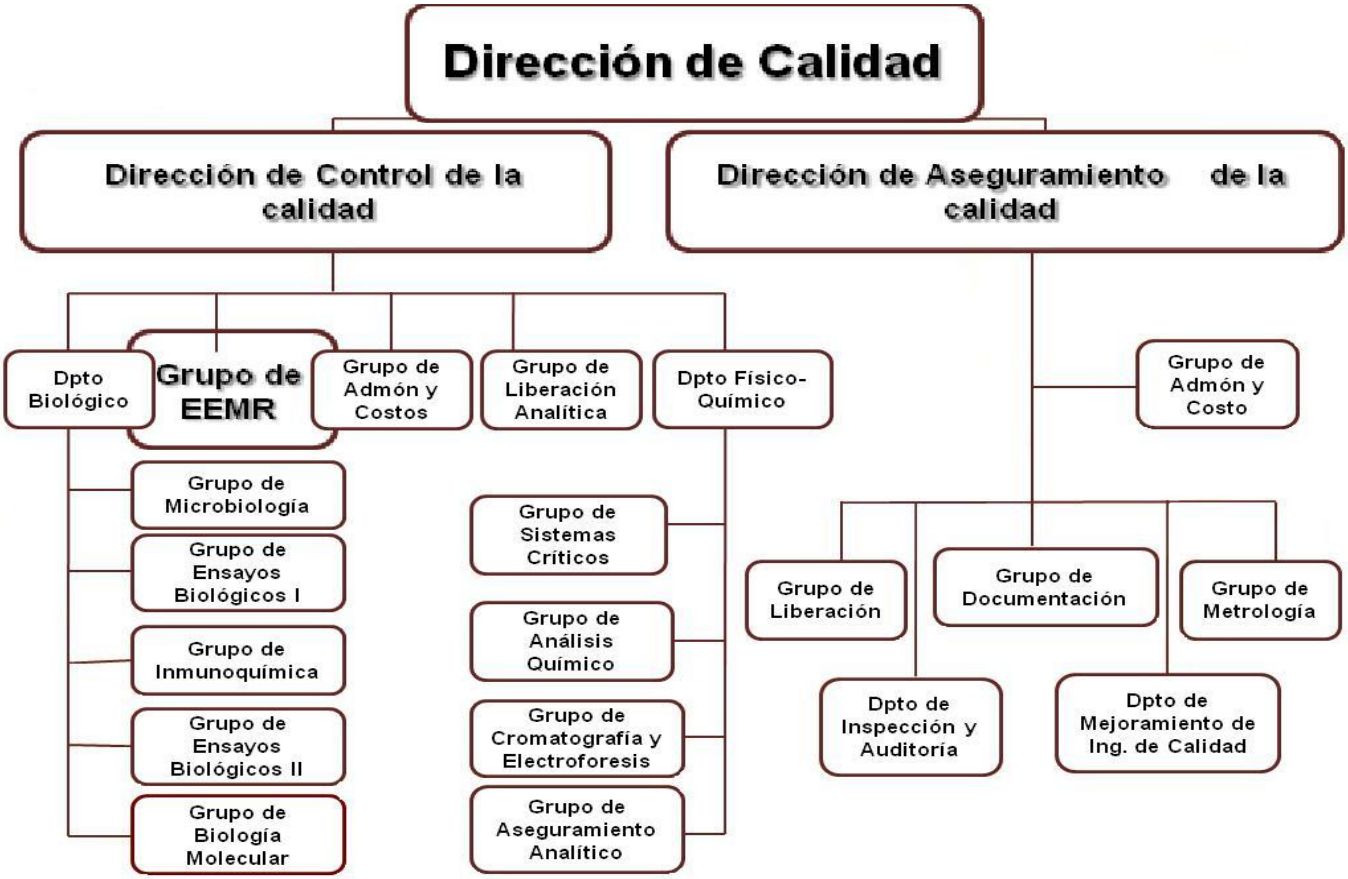
45. *Open Source Laboratory Automation &informatics .* [En línea] [Citado el: 22 de Mayo de 2009.] http://www.labmatica.com/.

46. Grupo de Soluciones, INNOVA, Rational RequisitePro, [En línea] [Citado el: 26 de Mayo de 2009.] http://www.rational.com.ar/herramientas/requisitepro.html.

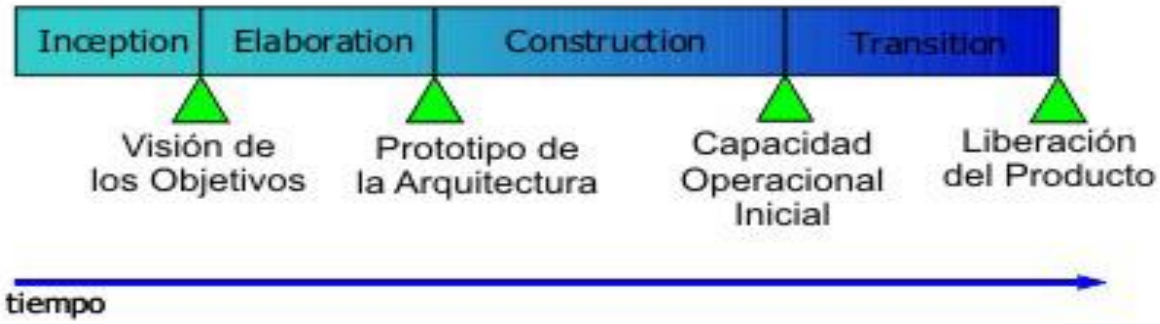
47. CorporaciónSybven, Productos-IBM [En línea] [Citado el: 26 de Mayo de 2009.] http://www.corporacionsybven.com/portal/index.php?option=com_content&task=view&id=216.

Anexos

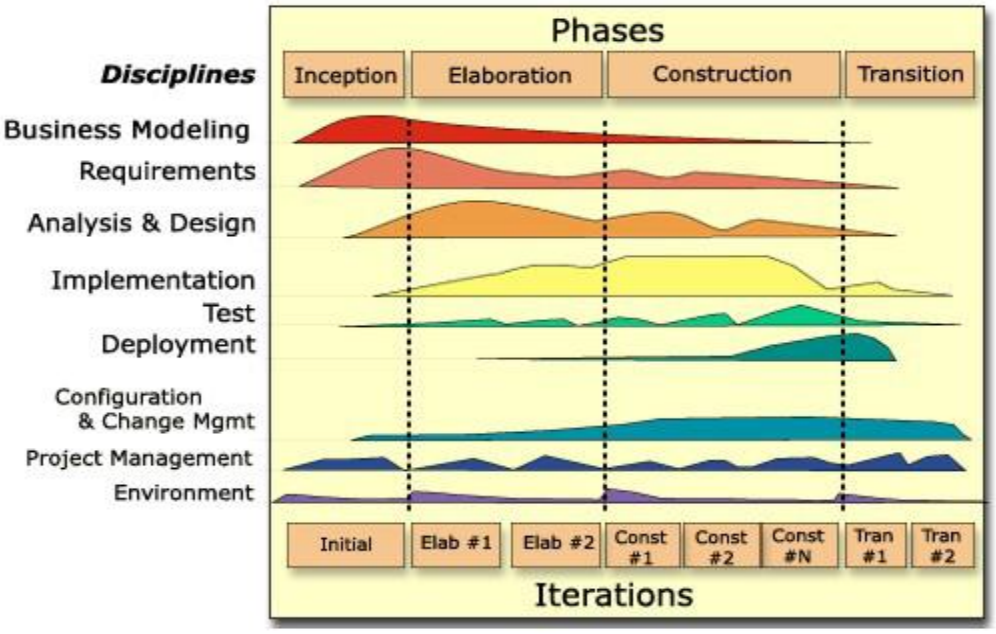
1. **Anexo 1** *Estructura jerárquica del Área de Calidad del Centro de Ingeniería Genética y Biotecnología*



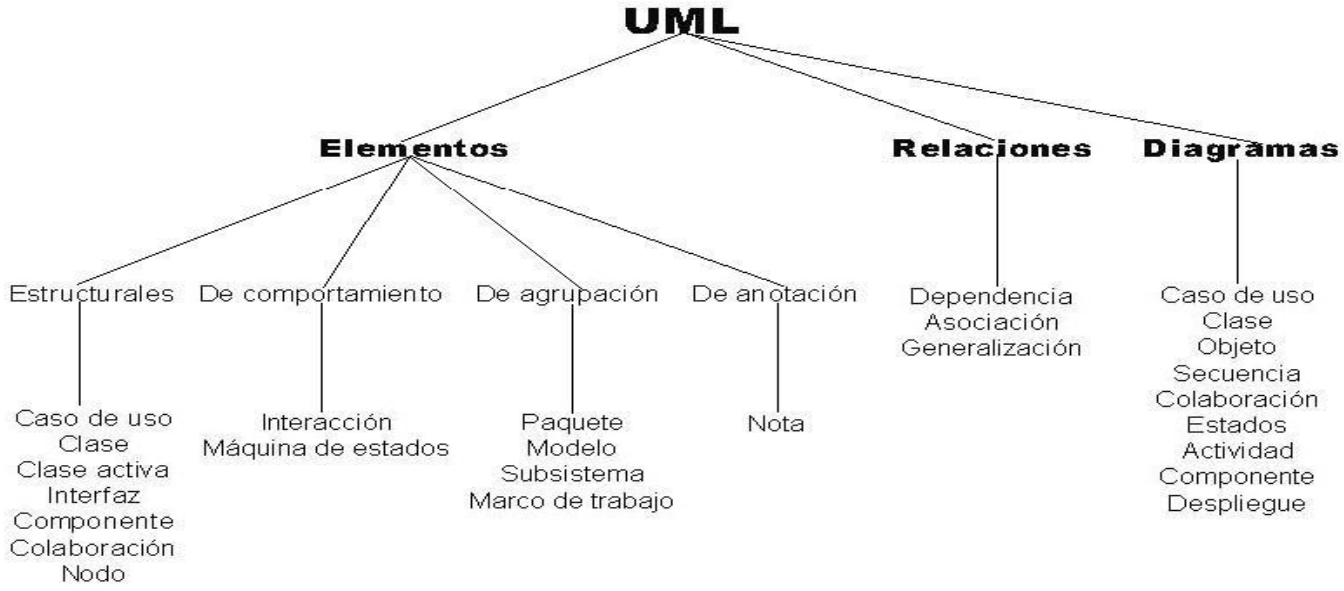
2. Anexo 2 *Hitos de RUP*



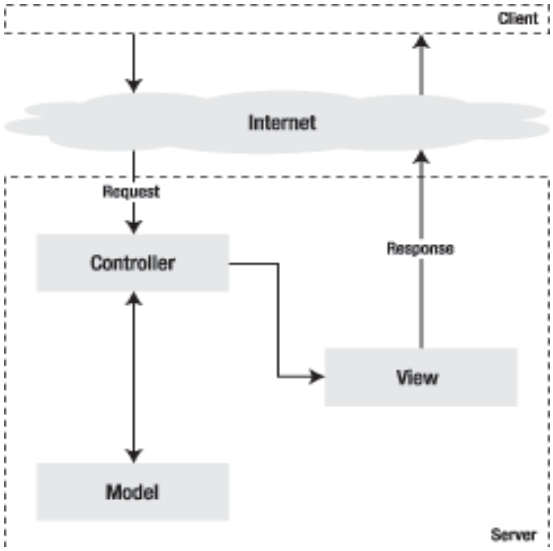
3. Anexo 3 *Fases y Flujos de Trabajos defino por RUP*



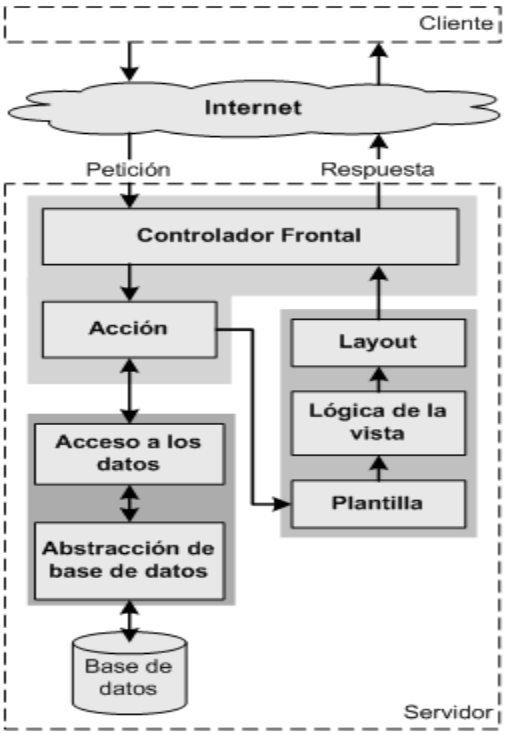
4. Anexo 4 *El vocabulario de UML*



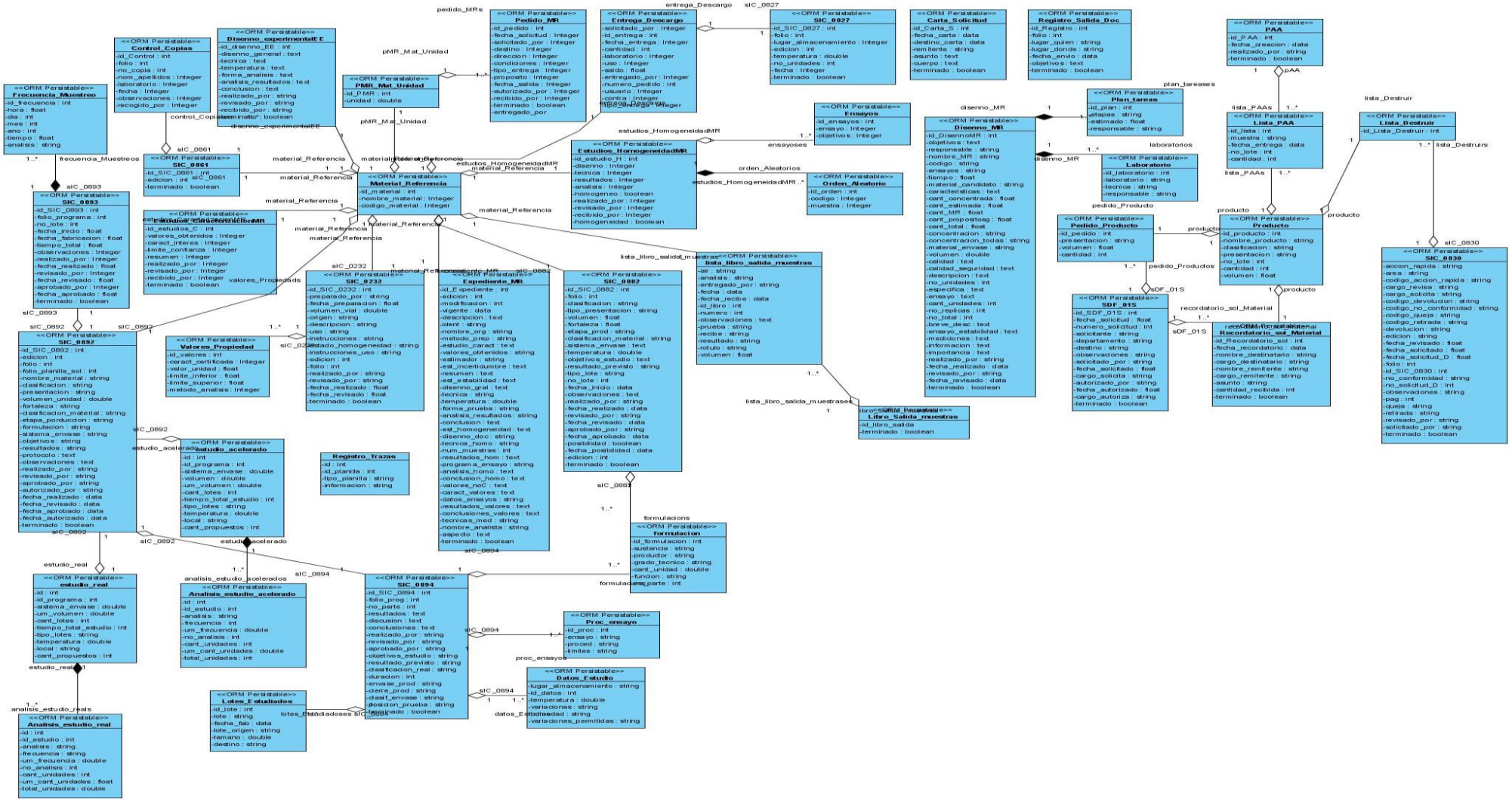
5. Anexo 5 *Patrón MVC*



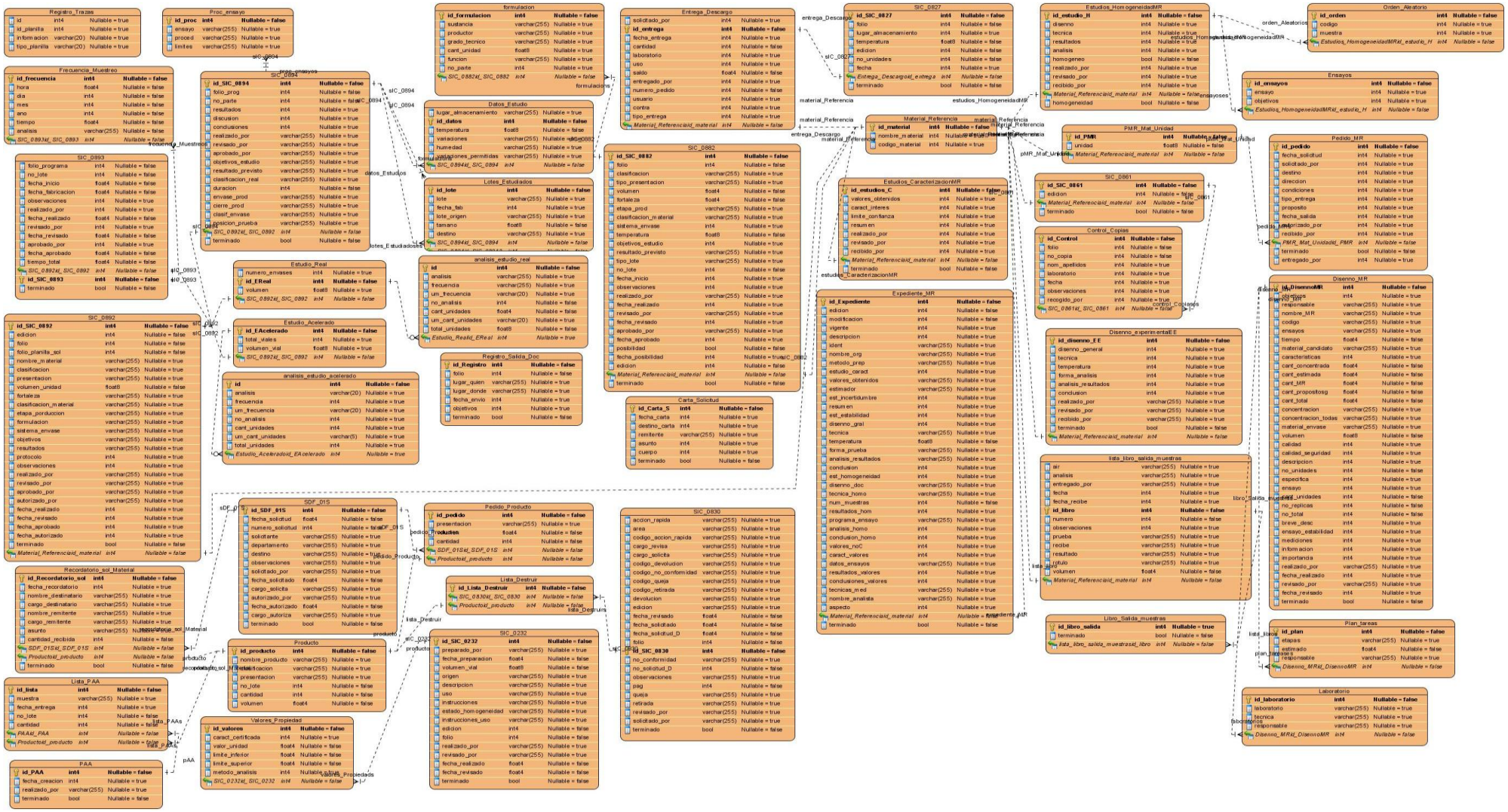
6. Anexo 6 *El flujo de trabajo de Symfony*



7. **Anexo 7** *Clases Persistentes con todos sus atributos*



8. **Anexo 8** *Modelo de Datos con todos sus atributos*



Glosario de Términos

Caracterización: Proceso por el cual se atribuyen valores cualitativos y cuantitativos a las diferentes características de un producto o material de referencia.

Estabilidad: Propiedad de cualquier forma farmacéutica contenida en un determinado material de envase de mantener, dentro de ciertos límites y durante el tiempo de almacenamiento y uso, las características físicas, químicas, microbiológicas, toxicológicas y terapéuticas que tenía en el momento de su fabricación.

Estudio de Estabilidad: Estudio de las características físicas, químicas, microbiológicas y biofarmacéuticas de un "producto" en un sistema de envase y cierre determinado, bajo condiciones de almacenamiento específicas y durante un período de tiempo que avale la fecha de vencimiento propuesta. Serie de ensayos que permiten obtener información para establecer el período de validez de un medicamento en su envase original y en las condiciones de almacenamiento especificadas.

Homogeneidad: Característica que dice que todas las porciones de un todo son iguales.

Material de Referencia: Sustancia farmacéutica activa o producto intermedio de calidad y pureza establecidas en comparación con estándar primario, que se utiliza como sustancia de referencia para los ensayos rutinarios de laboratorio.

Muestra: Pequeña parte que es representativa de un lote en un tiempo y condiciones específicas o determinadas.

Validación: Acción documentada que demuestra, de acuerdo con los principios de las Buenas Prácticas de Fabricación, que cualquier procedimiento, proceso, equipo, material, actividad o sistema realmente brinda los resultados esperados.

Módulos: Cada aplicación está formada por uno o más módulos. Un módulo representa a una página web o a un grupo de páginas con un propósito relacionado.