Universidad de las Ciencias Informáticas

“Facultad 6”

Título: “Sistemas de Gestión de Información de Laboratorios de la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología: Implementación del módulo Inmunoquímica”

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor(es): Dennis Muñiz Pérez
Fidel Cesar Ortega Pino

Tutor(es): Ing. Niusvel Acosta Mendoza
Ing. Loismarx Peña González

Ciudad de La Habana, Junio del 2009
"Año del 50 Aniversario del Triunfo de la Revolución"

“La vida de un hombre es lo que sus pensamientos hacen de ella.”

Marco Aurelio

DECLARACIÓN DE AUTORÍA

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

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

Dennis Muñiz Pérez

_____

Firma Autor

Fidel Cesar Ortega Pino

_____

Firma Autor

Ing. Niusvel Acosta Mendoza

_____

Firma Tutor

Ing. Loismarx Peña González

_____

Firma Tutor

DATOS DE CONTACTO

Ing. Niusvel Acosta Mendoza (nacosta@uci.cu)
Graduado de Ingeniero en Ciencias Informáticas de la Universidad de las Ciencias Informáticas (UCI) de la Habana. Profesor Instructor de la Facultad 6.

Ing. Loismarx Peña González (lpgonzalez@uci.cu)
Graduado de Ingeniero en Ciencias Informáticas de la Universidad de las Ciencias Informáticas (UCI) de la Habana. Profesor Instructor de la Facultad 6.

AGRADECIMIENTOS

A mi familia que me hizo el hombre que soy y que sin ella no podría estar hoy donde estoy.

A nuestra Revolución y en especial a nuestro Comandante en Jefe, por ser el creador de esta Universidad del Futuro…

A mi novia Yanet por su apoyo incondicional en todo momento

A Niusvel y Loismarx, mis dos tutores, sin la sabiduría y consejo de ellos no se podría haber hecho este trabajo.

A Niusvel en especial por ser más que un tutor, ser un amigo, hermano y compañero.

A Fidel, mi compañero de tesis, sin sus conocimientos y esfuerzo no se hubiese terminado este trabajo.

A mis amigos (hermanos) que siempre estuvieron pendientes a cualquier problema que tuviese.

A todos los compañeros de mi aula que en el transcurso de estos 5 años compartimos muchas cosas

A todos los que un momento determinado nos brindaron su apoyo desinteresado…

A toda esa gente linda, muchas gracias, le estaremos eternamente agradecidos…

Dennis Muñiz Pérez

A toda mi familia por estar siempre presentes y por haberme ayudado a ser el hombre que soy.

A mis abuelos por estar siempre que los necesite.

A Omar mi padrastro que me ayudo siempre que necesite de su ayuda.

Al Comandante en Jefe Fidel y a la revolución, por dejarnos formar parte de la primera Universidad surgida al calor de la Batalla de Ideas.

A mi compañero Dennis que más que un amigo, fue como un hermano para mí y sin su ayuda no sería posible realizar este trabajo.

A mi dos tutores Niusvel y Loismarx por estar siempre pendientes de nosotros y habernos guiado por el camino de su sabiduría.

A Niusvel que más que un tutor fue un amigo y un hermano, y me apoyo en todo momento.

Al Yury, a Tony y al ansia de Michel por ser grandes amigos y ayudarme.

A mis amigos de Camagüey (Joan, Pavel, Ramón y Abel).

A Daryanis por ser buena amiga y apoyarme siempre.

A todos mis amigos que de una forma u otra me ayudaron a hacer posible mi formación y me brindaron su apoyo desinteresado.

A mis compañeros de aula que compartieron conmigo muchas experiencias en los 5 años de mi carrera.

Y a todo aquel me que ayudo en algún momento de de mi carrera, muchas gracias.

Fidel Cesar Ortega Pino

DEDICATORÍA

A mi madre Olga Lidia, una mujer brillante en todos los sentidos, que me ha apoyado y guiado en todo momento, una mujer que debo mi vida entera y gracias a ella estoy en lugar que estoy ahora.
A mi padre Aurelio, de quien admiro insaciablemente su inteligencia y su carácter el cual me ha hecho ser el hombre que soy.
A mi abuela Lidia que ha estado junto a mí en las buenas como en las malas y a mi abuelo Sergio (Nené) que en paz descanse le debo mucho en esta vida por cómo me ayudo en ella.
A mis tíos Pipo y Zoila que para mí son como mis segundos padres.
A mi prima Zolangel que es como una hermana para mí.
A mi novia Yanet que ha sido mi fiel confidente más que una novia, mi compañera, amiga…
A todos mis amigos.

Dennis Muñiz Pérez

A mis padres que fueron mi guía, mi ejemplo a seguir y TODO en este mundo.
A mis hermanos Ferna, Alfre y Yinet que siempre los tengo presentes y por apoyarme en todo momento.
A mis dos sobrinos.
A mis abuelos Mariano que en paz descanse, a mi abuela Olga por ser como una segunda madre para mi, a mi abuela Gertrudis y a mi abuelo Ñico, a todos ellos que siempre los tengo presentes.
A mis tíos que siempre me apoyaron y porque me quieren como si fuera un hijo para ellos.
A mis primos que los admiro a todos.

Fidel Cesar Ortega Pino

RESUMEN

En la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología (CIGB) se maneja a diario un gran volumen de información referente a cada ensayo realizado a las muestras que allí se estudian, con el objetivo de llevar el control de la calidad de las mismas. El proceso de gestionar tanta información se torna muy lento e impreciso por el personal de cada grupo, cuestión que dificulta un eficiente control de la información.

La presente investigación aborda la implementación de uno de los grupos de la Dirección de Calidad del CIGB con el objetivo de informatizar el manejo de esta información, es por ello que se está desarrollando un Sistema de Gestión de Información de Laboratorios (LIMS por sus siglas en inglés). El trabajo se basará en la utilización del framework Symfony para desarrollar el módulo Inmunoquímica (IQ), donde se identificarán las funcionalidades que deberá cumplir el sistema.

Palabras claves: CIGB**,** LIMS**,** Inmunoquímica.

ÍNDICE

ÍNDICE DE FIGURAS

Introducción

Desde los años 80 estamos inmersos en una época que se ha llamado “la era de la Ciencia”. En la actualidad el uso de las Tecnologías de la Información y de las Comunicaciones (TICs), están dando lugar a una nueva vía para fortalecer la economía. En la década del 90 el hombre empieza a vincular la informática con la salud y da grandes pasos de avances en la investigación de la genómica humana.

La implantación y utilización de las TICs en todo el mundo brindan una amplia gama de servicios, aplicaciones y tecnologías que utilizan diversos tipos de equipos y programas informáticos. En los últimos años en el país se está llevando a cabo un proceso de informatización de la sociedad, este proyecto se ha realizado por la dirección del mismo alcanzando resultados satisfactorios en áreas tan esenciales como la Educación, la Salud y la Investigación.

Los efectos de esta labor han llegado a centros importantes del país, que se informatizan exitosamente, logrando grandes avances en el procesamiento de la información que en estos se generan. Entre ellos está el Centro de Ingeniería Genética y Biotecnología (CIGB) con una trayectoria de más de veinte años de investigación científica y la obtención de productos reconocidos a nivel mundial. Es uno de los centros investigativos que se ha destacado por los resultados obtenidos no sólo en cuanto a la investigación, sino también en el desarrollo, producción y comercialización de productos biológicos obtenidos a través de los métodos de la biotecnología moderna.

El trabajo realizado por el CIGB ha tenido gran impacto en la biomedicina, salud animal y la bioindustria, ha 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.

La calidad es la imagen del Centro de Ingeniería Genética y Biotecnología como organización comprometida con sus clientes y con la sociedad. Los productos desarrollados y elaborados en el Centro se distinguen por su eficacia, seguridad y calidad, para ello en su estructura existe la Dirección de Calidad, compuesto por el Departamento de **Aseguramiento de la Calidad** y el Departamento de **Control de la Calidad.**

En el entorno actual, tan complejo y dinámico, la Dirección de Calidad se mantiene en un proceso de búsqueda e investigación con respecto a las normas y regulaciones aplicables a la industria

biofarmacéutica como necesidad imperiosa para la proyección del trabajo en función de lograr la calidad de los productos que oferta el CIGB así como la satisfacción y confianza de los clientes.

El Departamento 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 los 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).

Este Departamento está compuesto por dos Departamentos y los Grupos de trabajo (1):

- Departamento de Mejoramiento de la Calidad (DMC)
- Departamento de Inspección y Auditoria (DIA)
- Grupo de Documentación (GDOC)
- Grupo de Metrología (GM)
- Grupo de Liberación de Lotes (GLL)

El Departamento de **Control de la Calidad** tiene entre sus funciones fundamentales las relacionadas con el muestreo, las especificaciones, los ensayos y la evaluación de la calidad de los productos que se generan en la empresa. Además debe comprobar y poner en práctica todos los procedimientos de control, evaluar, mantener y almacenar los materiales de referencia, asegurar que se controle la estabilidad de los ingredientes farmacéuticos activos (IFA) y los productos terminados. Además es el responsable de autorizar o rechazar las materias primas y los productos intermedios. Para acometer estas funciones, la Dirección está compuesta por dos Departamentos y tres Grupos de trabajo (2):

- Grupo de Administración y Costo (GAC)
- Grupo de Estabilidad y Materiales de Referencia (GEMR)
- Grupo de Liberación Analítica (GLA)
- Departamento Físico – Químico compuesto por cuatro laboratorios:
  - Grupo Análisis Químico (GAQ)
  - Grupo Aseguramiento Analítico (GAA)
  - Grupo de Cromatografía y Electroforesis (GCE)
  - Grupo de Sistemas Críticos (GSC)
- Departamentos Biológicos compuesto por cinco laboratorios:

- Grupo de Microbiología (GM)
- Grupo de Biología Molecular (GBM)
- Grupo de Ensayos Biológicos I (GEB I)
- Grupo de Ensayos Biológicos II (GEB II)
- **Grupo de Inmunoquímica (GIQ) (Ver Anexo 1)**

En el Departamento de Biológicos se ejecutan las técnicas analíticas relacionadas con la determinación de la actividad biológica *in vivo e in vitro*, cuantificación e identificación de proteínas contaminantes y proteínas de interés por métodos inmunológicos tipo ELISA, Western-Blot, Inmunodot. Entre sus laboratorios se encuentra el Laboratorio de Inmunoquímica el cual genera grandes volúmenes de datos pero que no están integrados en una plataforma capaz de organizar y centralizar todo el flujo de información.

El almacenamiento de los datos es en formato duro, como: libros de reportes, registros y expedientes para lotes de productos; lo que significa un cúmulo de información considerable, y dificulta la búsqueda y actualización de la misma, por lo que se comenzó el proceso de informatización de la gestión de la información de dicho grupo, constituyendo éste un módulo del Sistema de Gestión de la Información del Laboratorio (LIMS por sus siglas en inglés) que se está desarrollando para la Dirección de Calidad del CIGB.

Toda esta documentación siempre es revisada, supervisada y aprobada por las personas con rangos superiores dentro del centro, lo cual puede provocar una pérdida de tiempo considerable, tiempo que se pierde cuando se necesita una respuesta rápida para la toma de decisiones.

Se puede concluir que el proceso de Gestión de la Información del Laboratorio de Inmunoquímica no fluye de manera eficiente, segura y rápida.

En el año 2008 estudiantes de la Universidad de la Ciencias Informáticas (UCI) presentaron un trabajo de diploma que tiene como objetivo realizar una aplicación para mantener de forma organizada y centralizada todo el flujo de información generada por el laboratorio de Inmunoquímica. El desarrollo de la misma para el módulo de Inmunoquímica sólo llegó hasta la fase de Análisis y Diseño. Durante el presente trabajo se realizará la implementación del mismo para que la primera versión del módulo este a disposición del cliente, por tanto surge como **problema científico:**

¿Cómo desarrollar un sistema para mejorar el proceso de almacenamiento de la información para el módulo Laboratorio de Inmunoquímica del LIMS de Calidad del CIGB?

El problema planteado se enmarca en el **objeto de estudio:** Los Sistemas de Gestión de Información de Laboratorios.

El objeto de estudio delimita el **campo de acción:** Proceso de desarrollo de los Sistemas de Gestión de Información de los Laboratorios.

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

Tareas para dar cumplimiento al objetivo general:

- Estudio de la documentación obtenida en las iteraciones anteriores realizadas al módulo Inmunoquímica.
- Familiarización con las herramientas y tecnologías definidas en la arquitectura del proyecto.
- Análisis de los procesos y la documentación generada en el Laboratorio Inmunoquímica.
- Estudio sobre el Framework Symfony.
- Implementación del módulo Inmunoquímica.
- Realizar los diagramas de componentes.
- Realizar la seguridad del módulo.

Estructura del trabajo:

El trabajo está compuesto por una Introducción, dos Capítulos, Conclusiones, Recomendaciones, Bibliografías, Referencias Bibliográficas, Anexos y un Glosario de términos.

Capítulo 1. Fundamentación Teórica: En este capítulo se definen los sistemas de gestión de la información, se hace mención a las funcionalidades de algunos LIMS como ejemplo de sistemas de gestión de la información de laboratorios. Además se realizará una caracterización de las tecnologías y las herramientas para la solución más óptima del trabajo.

Capítulo 2. Implementación del Sistema: Este capítulo consta con la descripción de los patrones de diseño y de arquitectura que se usaron para la realización de la aplicación, así como la descripción de la solución a implementar para la construcción del software.

Capítulo 1
Fundamentación Teórica

1.1 Introducción

En este capítulo se abordan aspectos generales y se hace alusión a las funcionalidades de algunas aplicaciones a nivel mundial dedicadas a la realización de LIMS, se hace referencia a tendencias y tecnologías actuales que se utilizan en el mundo para el desarrollo de aplicaciones Web y para un correcto desempeño del rol de implementador en el proyecto LIMS de Calidad.

1.2 Sistema de Gestión de Información

Los Sistemas de Gestión de Información (SGI) constituyen un conjunto de elementos relacionados y ordenados. Estos ayudan a tomar decisiones estratégicas, analizar problemas y controlar una organización.
Los SGI permiten:

- Comprender la marcha de las organizaciones desde un enfoque analítico (donde queremos estar), evaluador (donde estamos) y creativo (donde podríamos estar).
- Develar oportunidades que merezcan ser explotadas y contrarrestar amenazas.
- Establecer los factores que resulten críticos y las necesidades asociadas al SGI.
- Estudiar el impacto de los SGI en la posición del negocio y buscar nuevas oportunidades.

La gestión de la información se vincula con la generación y la aplicación de estrategias, el establecimiento de políticas, así como con el desarrollo de una cultura organizacional y social dirigida al uso racional, efectivo y eficiente de la información en función de los objetivos y metas trazadas en materia de desempeño y de calidad.

1.3 Sistema de Manejo de Información de Laboratorios (LIMS)

Un LIMS o "Laboratory Information Management System" es un programa de gestión de la información de los laboratorios que permite recoger, almacenar, calcular y gestionar datos en una amplia variedad

de formas. Estos representan una importante herramienta para la gestión global de un laboratorio en un entorno de calidad, agilizando temas de registro de datos primarios, archivo, trazabilidad, entre otros. Ayuda considerablemente a minimizar los errores producidos durante la transferencia de datos, ya que no se podrá duplicar con facilidad ningún tipo de información y el usuario va a estar obligado a insertar solo el tipo de información que tenga predefinido el sistema.

Algunos de los beneficios de los sistemas de gestión de la información de los laboratorios, son:

- Mejor organización de la información.
- Integridad de la información adquirida en los Ensayos.
- Rapidez en la gestión de informes.
- Obtención y validación de los resultados.

El hacer uso de un LIMS revoluciona el trabajo dentro del laboratorio y con éste a todos sus trabajadores. Además, es de gran importancia en el proceso de integración de la tecnología en el laboratorio.

1.3.1 Fabricantes de LIMS y características de sus productos

Después de realizar varias investigaciones se encuentra que en el mundo se viene trabajando desde hace varios años en los Sistemas de Manejo de Información de Laboratorios (LIMS). Con el objetivo de integrar el Polo científico a estos avances, se ha indagado entre los diferentes software que existen para el manejo de la información y entre los buscados se encuentran:

La Corporación StarLIMS, esta compañía se dedica a crear sistemas de manejo de información de laboratorios aportando soluciones completas, transforman los datos del laboratorio en información para facilitar a las empresas una toma de decisiones mucho más fácil. Respeta las organizaciones y sus reglas de negocio, utilizando la misma base de software. Esta corporación es una de las de mayor crecimiento en el mundo y en el desarrollo de soluciones de calidad para la gestión de información de laboratorios.

AssayNet es una compañía de desarrollo de software especializada en LIMS en el área ambiental y de minería. Su objetivo es promover la innovación en políticas ambientales y tecnología a través del intercambio profesional y la actividad comercial en este sector.

LIMS 2003 contiene todas las especificaciones que se espera en un LIMS moderno capaz de satisfacer las necesidades de la industria. Desde el recibo de las muestras, la preparación de las mismas hasta el análisis y el reporte, el programa ha sido diseñado para trabajar en el mismo ritmo que lo hace el laboratorio. Es intuitivo, con interface de menú, fácil de usar, con una serie de módulos tales como: QC, interface Web, facturación, códigos de barra, inventarios, etc. todo integrado en la misma pantalla. (3)

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).

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 (la licencia GNU General Public License (GPL) y 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.

Open LIMS: El objetivo de Open LIMS es desarrollar una organización y estructura independiente de software, que permite un acceso fácil a todos los miembros y los diferentes resultados que se gestionen en el LIMS. Los módulos son generalmente escritos 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 el posterior análisis. Es posible organizar el experimento con diferentes subproyectos. Cada proyecto puede tener un número ilimitado de subproyectos. 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, este todavía se encuentra en desarrollo.

Luego del estudio de las soluciones que proponen los diferentes software estudiados, se llega a la conclusión de que:
Utilizan varias características que facilitan el manejo de la información que se generan en los laboratorios, además de gestionar, recopilar y almacenar la misma y de esta forma llevar un mejor control. Además estas aplicaciones poseen altos costo para su adquisición, también la mayoría de los proveedores de LIMS son de Estados Unidos lo que hace que sea casi imposible de adquirirlos debido al Bloqueo Económico. Por otro lado existen algunos software que son libres, pero estos dificultan la utilización debido a que el CIGB presenta características que impiden que se pueda configurar y adaptarlos para su uso particular.
Por lo tanto, el objetivo del trabajo que se presenta es la implementación del primer LIMS en Cuba con las características para poder adaptarlo al Centro de Ingeniería Genética y Biotecnología.

1.4 Herramientas y Metodologías utilizadas en la aplicación

En este epígrafe se analizaran algunas de las metodologías y herramientas que se utilizarán en la construcción de un sistema para que este cumpla con todos los requisitos necesarios. Luego de estudiar estas herramientas y metodologías se llego a una propuesta final y en este epígrafe se analizará el ¿por qué? de su utilización para el desarrollo del sistema.

1.4.1 Metodología de desarrollo de software

Una metodología es un conjunto de actividades necesarias para llegar a un fin, con el propósito de transformar los requisitos de un usuario en un sistema de software. El objetivo de una metodología de desarrollo es garantizar la eficacia y la eficiencia en el proceso de generación de software. Existen diversas metodologías, citando algunos ejemplos se puede mencionar Microsoft Solution Framework (MSF), Extreme Programming (XP) y Rational Unified Process (RUP), respondiendo cada una de ellas a diferentes clasificaciones, agiles, flexibles y robustas.
En investigaciones anteriores realizadas se decidió utilizar como guía para el proceso de desarrollo de la solución, Rational Unified Process (RUP).

Proceso Unificado de Desarrollo de Software (RUP, Rational Unified Process)

El **Proceso Unificado de Rational** (*Rational Unified Process* en inglés, habitualmente resumido como **RUP**) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML por sus siglas en ingles), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso.

RUP está compuesto de 4 fases denominadas:

- **Inicio:** El objetivo de esta fase es establecer los requisitos de negocio que cubrirá el sistema identificando los principales casos de usos que interactúan con el sistema (personas, sistemas, etc.), identificar los riesgos fundamentales del mismo y hacer una valoración de la viabilidad del proyecto.
- **Elaboración:** El objetivo de esta fase es entender muy bien el problema desde el punto de vista del equipo de desarrollo. Lleva consigo la elaboración de la arquitectura marco del sistema y el diseño de la solución técnica, así como determinar el plan del proyecto y eliminar los riesgos fundamentales del mismo.
- **Construcción:** En esta fase se profundiza en el diseño de los componentes y de manera iterativa se van añadiendo las funcionalidades al software a medida que se construyen y prueban, permitiendo a la vez que se puedan ir incorporando cambios.
- **Transición:** La fase final se ocupa de la instalación del software a los usuarios finales que harán uso del sistema. Como consecuencia de esto pueden surgir nuevos requisitos.

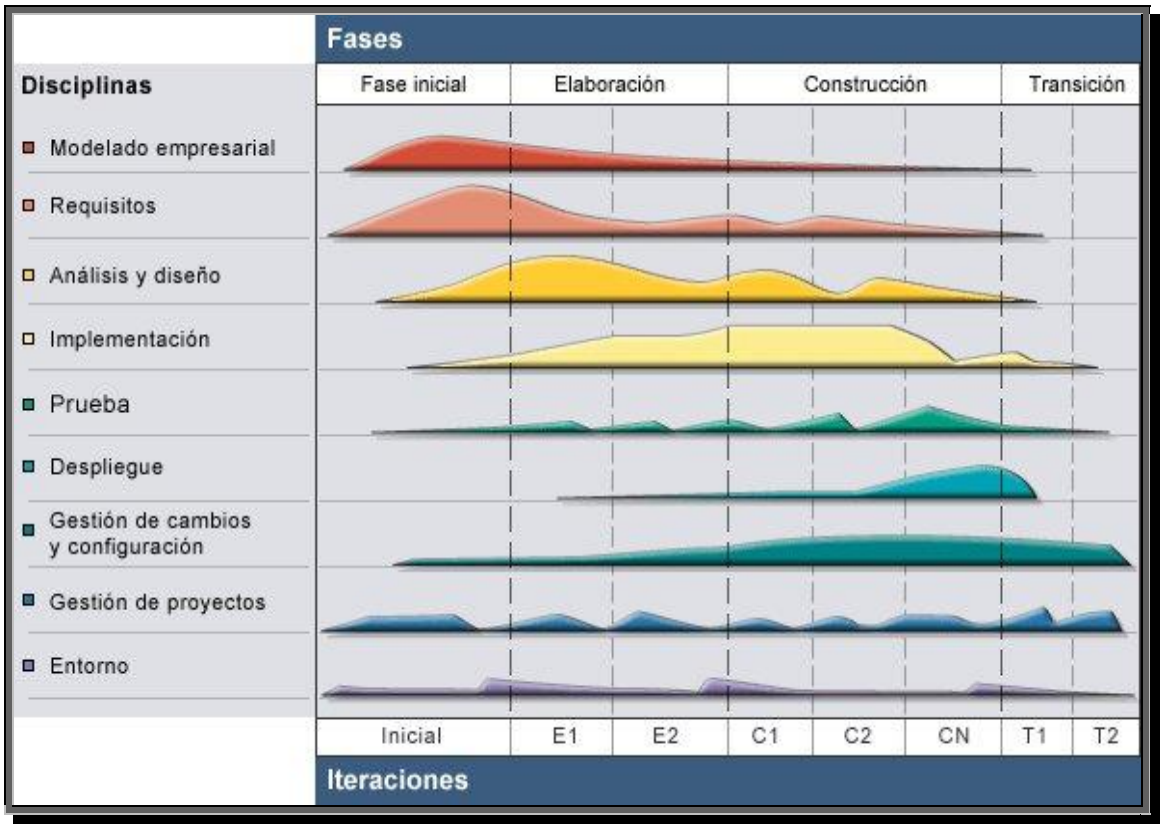


Figura 1: RUP en dos dimensiones

RUP también define nueve flujos de trabajo distintos:

1. **Modelado del negocio:** define todo el proceso de la organización para un mejor entendimiento y definir el ámbito del sistema.
2. **Requerimiento:** es uno de los flujos de trabajo más importantes, porque proporciona un conjunto de requisitos detallados sobre lo que debe hacer el sistema
3. **Análisis y diseño:** su objetivo es transformar los requisitos en un diseño del sistema en creación y adaptar el diseño para que se ajuste al entorno de implementación, con un diseño pensado para el rendimiento.
4. **Implementación:** definir la organización del código, en términos de los subsistemas e implementar los elementos de diseño en términos de los elementos de implementación.
5. **Pruebas:** proporciona orientación sobre cómo evaluar y valorar la calidad del producto.
6. **Despliegue:** describe las actividades asociadas al garantizar que el producto de software esté disponible para los usuarios.

7. **Gestión de configuración y cambios*:*** mantiene la integridad de todos los artefactos que se crean en el proceso, así como controlar y sincronizar la evolución que se ha seguido.
8. **Gestión del proyecto:** se centra en la planificación del proyecto, la gestión del riesgo, la supervisión del progreso y la métrica.
9. **Gestión del entorno:** proporcionan el entorno de desarrollo de software que da soporte al equipo de desarrollo, incluidos los procesos y las herramientas.

El desarrollo de este trabajo se centra en el flujo de trabajo de Implementación, que se explicará más adelante.

1.4.2 Lenguaje Unificado de Modelado (UML)

Cualquier rama de ingeniería o arquitectura ha encontrado útil desde hace mucho tiempo la representación de los diseños de forma gráfica. La falta de estandarización en la manera de representar gráficamente un modelo impedía que los diseños gráficos realizados se pudieran compartir fácilmente entre distintos diseñadores por lo que se necesitaba un lenguaje no sólo para comunicar las ideas a otros desarrolladores sino también sirve de guía, de apoyo en el proceso de análisis de un problema. Con este objetivo se creó el Lenguaje Unificado de Modelado (UML: Unified Modeling Language).

UML se ha convertido en ese estándar para visualizar, especificar, construir y documentar un sistema de software a través de los diferentes diagramas y modelos. El lenguaje UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático. **(Ver Anexo 2)**

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones (4):

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.

- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

1.4.3 Herramienta CASE para la modelación del Sistema

Las herramientas CASE (***C**omputer **A**ided **S**oftware **E**ngineering*, Ingeniería de Software Asistida por Ordenador) aumentan la productividad del desarrollo de software reduciendo el coste de las mismas en términos de tiempo y coste. La herramienta Case nos puede ayudar en los aspectos del ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue, ayuda a una más rápida construcción de aplicaciones de calidad.

Visual Paradigm 6.1

La herramienta CASE a utilizar para el desarrollo de la aplicación Visual Paradigm porque soporta el ciclo completo de vida del desarrollo del software, permite importación desde el Rational Rose. Este producto se pensó exclusivamente para el uso profesional. También posee una serie de características que la hacen más fácil de utilizar:

- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo.
- Capacidad de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de integrarse en los principales IDE de programación.
- Disponibilidad para múltiples plataformas

1.4.4 Lenguaje de Programación

El lenguaje de programación permite crear las funcionalidades de cada una de las interfaces graficas accedidas por el usuario. **PHP** es un lenguaje de programación interpretado, de propósito general para la creación de páginas web dinámicas y puede ser incrustado directamente dentro de cogido HTML. Es usado principalmente en la interpretación del lado del servidor, es gratuito e independiente de plataforma, rápido y con una amplia variedad de documentación para su mejor entendimiento.

PHP 5

Es uno de los mejores lenguajes de programación del lado del servidor para la creación de páginas web, es orientado a objeto, haciendo así uso de las ventajas que este paradigma nos ofrece. Permite trabajar con la lectura de ficheros XML de forma sencilla y la integración con diferentes Bases de Datos existentes en el mundo y es de fácil entendimiento e implementación.

PHP 5 incluye una serie de ventajas como:

- Soporte sólido y REAL para Programación Orientada a Objetos (OOP) con PHP Data Objects.
- Mejoras de rendimiento.
- Mejor soporte para MySQL con extensión completamente reescrita.
- Mejor soporte a XML (XPath, DOM...).
- Soporte nativo para SQLite.
- Soporte integrado para SOAP.
- Iteradores de datos.
- Excepciones de errores.

1.4.5 Servidor Web

Apache 2.2.3

Apache esta hecho por excelencia, su configurabilidad, robustez y estabilidad permite crear de forma rápida y sencilla un servidor http en el ordenador. Es el software más usado en el mundo. La licencia Apache es descendiente de las licencias BSD y no GPL, esta licencia permite hacer lo que desees con el código fuente.

Apache está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos, los cuales hacen que a menudo sean necesarias diferentes características o funcionalidades. Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular. Este diseño permite a los administradores de sitios web elegir qué características van a ser incluidas en el servidor seleccionando qué módulos se van a cargar, ya sea al compilar o al ejecutar el servidor (5).

Dadas las características que presenta este servidor es uno de los más utilizados y reconocidos a nivel mundial para desarrollar aplicaciones web:

- Es multiplataforma
- Es una tecnología gratuita de código abierto, lo que le da gran transparencia y si se quiere ver lo que se está instalando se puede hacer.
- Es altamente configurable y sencillo de configurar
- Permite personalizar las respuestas antes los errores que se produzcan en el servidor

Por todas estas características antes mencionadas es que se escoge Apache como servidor Web para desarrollar la aplicación.

1.4.6 Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado o, en inglés, *Integrated Development Environment* ('*IDE*'), es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse a un solo lenguaje de programación o bien, usarse para varios.

Eclipse PDT (PHP Development Tool)

El equipo de trabajo realizó un estudio de los diferentes entornos de desarrollo (Anjuta, Zend Studio, Eclipse, Nusphere, entre otros) y se llegó a la conclusión de utilizar el entorno de desarrollo Eclipse con un plugin para php.

Eclipse es un entorno de desarrollo integrado de código abierto para desarrollar diferentes aplicaciones. Esta plataforma es usada para múltiples aplicaciones basadas en framework así como para la construcción e otros IDEs que pueden ser usados para aplicaciones Web. Presenta un editor de texto, compilación en tiempo real. Provee asistencia y autocompletado de código, además ofrece un auto completamiento para el desarrollo del framework Symfony. Es un IDE de desarrollo multiplataforma y presenta multilicencia CC-By-Sa, GNU GPL y GNU GFDL.

1.4.7 Gestor de Base de Datos PostgreSQL

Los sistemas de gestión de base de datos (SGBD); (en inglés: DataBase Management System, abreviado DBMS) se utilizan para manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de los datos.

PostgreSQL 8.2

PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre. PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD.

Postgre es Full ACID compliant (Atomicity, Consistency, Isolation and Durability)

- Atomicidad (Indivisible) es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- Consistencia es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
- Aislamiento es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generará ningún tipo de error.
- Durabilidad es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema (6).

Debido a las características antes mencionadas y agregándole que PostgreSQL es una aplicación multiplataforma y se puede encontrar la documentación muy organizada, publica y libre, se llegó a la conclusión de utilizar esta herramienta para el desarrollo de la aplicación.

1.4.8 Framework para el desarrollo Web

Muchos de los desarrolladores de software conocen como mínimo o se han tropezados con el concepto de framework y a simple vista resulta que es muy fácil de definir como marco de trabajo. Sin embargo no resulta tan sencillo como parece esta definición.
Siendo muy simple, es **un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación**. Sí, es una definición muy genérica, pero también puede serlo un *framework*: sin ir más lejos, el paradigma MVC (Model-View-Controller) dice poco más que "separa en tu aplicación la gestión de los datos, las operaciones, y la presentación". En el otro extremo, otros *frameworks* pueden llegar al detalle de definir los nombres de ficheros, su estructura, las convenciones de programación, etc. (7)

Symfony 1.0.18

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux.) como en plataformas Windows.
Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. (8)

1.4.9 Patrones

Un patrón propone una solución o una vía para desarrollar un producto que surge en un contexto especifico. Existen diferentes patrones de llamadas entre objetos (similar a los patrones de diseño), decisiones y criterios arquitectónicos, de los cuales se pueden mencionar los patrones de arquitectura. Dentro de los patrones de diseño se encuentran los patrones GRASP y los patrones GOF.

Patrones de Diseño

Los patrones de diseño presentan una base para las soluciones de problemas en el momento de desarrollar un software, así como a otros ámbitos referentes al diseño de interacción o a la creación de interfaces.

Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles (9)

Entre los patrones de diseño se pueden mencionar los patrones GOF y los patrones GRASP.
Los patrones de diseño GOF se dividen en tres grupos principales:

- Patrones de Creación.
  - Patrón de Fábrica Abstracta
  - Patrón Constructor
  - Patrón del Método de Fabricación
  - Patrón Prototipo
  - Patrón de Instancia Única (Singleton)
- Patrones Estructurales.
  - Patrón Adaptador
  - Patrón Puente
  - Patrón Compuesto
  - Patrón Decorador
  - Patrón de Fachada
  - Patrón de Peso Mosca
  - Patrón Apoderado
- Patrones de Comportamiento.
  - Patrón de Cadena de Responsabilidad
  - Patrón de Comando
  - Patrón Intérprete
  - Patrón Iterador
  - Patrón Mediador

- Patrón Memento
- Patrón Observador
- Patrón de Estado
- Patrón de Estrategia
- Patrón del Método Plantilla
- Patrón Visitante

Los patrones de diseño GRASP son los siguientes:

- Creador
- Experto
- Alta Cohesión
- Controlador
- Bajo Acoplamiento

Patrones de Arquitectura

Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular recurrente que surge en un cierto contexto. Este esquema se especifica describiendo las componentes, con sus responsabilidades, relaciones, y las formas en que colaboran. Además expresan un esquema organizativo estructural fundamental para sistemas software.

Patrones para estructurar un sistema de software:

- Capas.
- Tuberías y Filtros.
- Tablero.

Patrones para distribuir un sistema de software:

- Intermediario

Patrones para sistemas de software que interactuar con seres humanos:

- MVC (Modelo Vista Controlador).
- PAC (Presentación-Abstracción-Control).

Patrones para sistemas de software cambiantes:

- Reflexión.
- Núcleo simple.

1.5 Flujo de trabajo de implementación

El objetivo principal de esta disciplina es convertir los elementos del diseño en elementos de implementación, dichos elementos son códigos fuentes, ejecutables, binarios, entre otros. Otra parte de esta disciplina son las pruebas de unidad, las cuales se limitan a los componentes de software implementados. De esta disciplina se obtiene un sistema ejecutable estable, constituido de los resultados producidos por los programadores individuales. (10)

En cada iteración de este flujo de trabajo habrá que hacer:

- Planear que subsistemas deben ser implementados y en qué orden deben ser integrados, formando el Plan de Integración.
- Cada implementador decide en qué orden implementa los elementos del subsistema. Si encuentra errores de diseño, los notifica.
- Se testean los subsistemas individualmente.
- Se integra el sistema siguiendo el plan.

Esta etapa de construcción del software está determinada por el lenguaje de programación que se decida utilizar para desarrollar el producto. Los diagramas de despliegue y los diagramas de componentes conforman lo que se conoce como un modelo de implementación al describir los componentes a construir, su organización y dependencia entre nodos físicos en los que funcionará la aplicación.

1.5.1 Roles, artefactos y actividades.

¿Qué es un rol?

Un rol define el comportamiento y responsabilidades de un individuo, o de un grupo de individuos trabajando juntos como un equipo. Una persona puede desempeñar diversos roles, así como un mismo rol puede ser representado por varias personas. (11)

Los roles propuestos por RUP para el flujo de trabajo de implementación son:

- **Rol de Implementador:** es responsable de los componentes de desarrollo y de prueba, de acuerdo con los estándares aprobados por el proyecto, para la integración en subsistemas más grandes. También es responsable del desarrollo y las pruebas de los componentes de prueba y los subsistemas correspondientes. **(Ver Anexo 3)**
- **Rol de Arquitecto de software:** dirige el desarrollo de la arquitectura de software del sistema como su nombre lo indica, que incluye la promoción y la creación de soporte para las decisiones técnicas clave que restringen el diseño global y la implementación para el proyecto. **(Ver Anexo 4)**
- **Rol de Integrador de sistemas:** es el responsable de la planificación y la ejecución de la integración de los subsistemas y los sistemas de implementación para producir compilaciones. **(Ver Anexo 5)**

Por la relevancia que tienen para el proyecto, durante esta investigación serán desempeñado todos los roles para darle solución al sistema en desarrollo.

¿Qué es un artefacto?

Un artefacto es un producto de trabajo en un proceso: los trabajadores utilizan artefactos para realizar actividades y producen artefactos como resultado de sus actividades. Los artefactos son resultados tangibles que se van generando durante todo el ciclo de vida del producto como consecuencia de las actividades que tengan que desempeñar los trabajadores.

Artefactos que son generados por el rol de implementador.

Subsistema de implementación

Los subsistemas de implementación proporcionan una forma de organizar los artefactos del modelo de implementación en partes manejables. Este artefacto consta de un conjunto de elementos de implementación. Estructura el modelo de implementación dividiéndolo en componentes más pequeños que se pueden integrar y probar separadamente.

El subsistema de implementación ayuda a organizar un modelo de implementación y, por lo tanto, a reducir su complejidad. Es el análogo físico del paquete de diseño. El modelo de implementación y los

subsistemas de implementación se definen inicialmente en la vista de implementación y, por lo tanto, son de vital importancia en el momento del desarrollo.

Es importante entender que un subsistema de implementación se manifiesta a través de un "mecanismo de empaquetamiento" concreto en un entorno de implementación determinado, tales como:

- Un paquete en Java y PHP.
- Un proyecto en Visual Basic.
- Un directorio de ficheros en un proyecto de C++.
- Un paquete en una herramienta de modelado visual como Rational Rose. (12)

La semántica de los subsistemas de implementación no es única, sino que va a variar con frecuencia y se refinará en dependencia del entorno de implementación en que se trabaje.

Los subsistemas de implementación están muy relacionados con los subsistemas de diseño en el modelo de diseño. De hecho, los subsistemas de implementación deberían seguir la traza uno a uno de sus subsistemas de diseño correspondientes.

Elementos de implementación

Los principales objetivos que se persiguen en el momento de realizar los elementos de implementación, son tener una estructura del modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones existentes entre los diferentes componentes.

Estos artefactos son los componentes físicos que forman una implementación, que incluyen archivos y directorios. Incluyen los archivos de código de software (origen, binario o ejecutable), los archivos de datos y los archivos de documentación, como los archivos de ayuda en línea.

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. Algunos estereotipos estándar de componentes son los siguientes:

- <<executable>> es un programa que puede ser ejecutado en un nodo.
- <<file>> es un fichero que contiene código fuente o datos.
- <<library>> es una librería estática o dinámica.

- <<table>> es una tabla de base de datos.
- <<document>> es un documento. (13)

Los componentes tienen relaciones de traza con los elementos que implementan. Es normal que un componente implemente varios elementos, por ejemplo varias clases.

Artefactos que son generados por el arquitecto de software

Modelo de Implementación.

El modelo de implementación describe cómo los elementos del modelo de diseño, las clases se van a implementar en términos de componentes. Este describe cómo se van a organizar los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y como los componentes dependen unos de los otros.

El modelo de implementación define una jerarquía, este se representa con un sistema de implementación que denota el subsistema de nivel superior. Por tanto esta es una forma de organizar el modelo en parte que sea más fácil de manejar para reducir la complejidad del mismo. Un modelo más detallado puede incluir un código fuente de bajo nivel y ficheros derivados, y sus relaciones con el modelo de diseño. Tales detalles son recomendados solo si se tiene una sincronización automatizada entre el modelo y los ficheros.

Un modelo de implementación no es obligado realizarlo. Si se elige crear un modelo de implementación, hay que tener dos aspectos esenciales: las decisiones de personalización claves son cómo relacionar el modelo de implementación y el modelo de diseño y qué elementos de implementación son lo suficientemente importantes para el modelo.

- Un modelo de implementación contiene:
- Introducción.
- Subsistemas de implementación.
- Elementos de implementación.
- Relaciones.
- Diagramas.

- Pruebas de Stub.
- Elementos de prueba.
- Vista de implementación.
- Interfaz

¿Qué es una actividad?

Una actividad consiste en una unidad de trabajo donde se mide el esfuerzo de una persona o individuo, durante su desempeño en uno de los roles definidos por RUP, la cual podrá ser realizada y siempre va a estar ligada a cierta necesidad. Una actividad es algo que realiza un trabajador para proveer un resultado de valor en el contexto de un proyecto.

Las actividades son descompuestas en pasos. Se puede distinguir tres categorías de pasos:

- Pasos de análisis: donde el trabajador comprende la naturaleza de la tarea, examina los artefactos de entrada, y formula las salidas.
- Pasos de acción: donde los trabajadores crean o actualizan algunos artefactos.
- Pasos de revisión: donde los trabajadores inspeccionan los resultados según determinados criterios.

Actividad realizada por el rol de implementador

Implementar Componentes (ingeniero de componentes).

La responsabilidad principal de implementar los componentes es de los ingenieros de componentes que son los encargados de definir y mantener el código fuente de uno o varios componentes, garantizando que cada componente implementa la funcionalidad correctamente. Y a veces también mantiene la integridad de uno o varios subsistemas de implementación. (**Ver Anexo 8)**
En esta actividad:

- Los implementadores escriben código fuente, adaptan el código fuente existente, compilan, enlazan y realizan pruebas de la unidad, a medida que van implementando los elementos del modelo de diseño.

- Los implementadores también arreglan los defectos de código y realizan pruebas de unidad para verificar los cambios. Finalmente, el código se revisa para evaluar la calidad y el cumplimiento de las directrices de programación.

Actividad que realiza el arquitecto de software

Implementación de la arquitectura.

El propósito de la implementación de la arquitectura es esbozar el modelo de implementación y su arquitectura mediante:

- La identificación de componentes significativos arquitectónicamente, tales como componentes ejecutables.
- La asignación de componentes a los nodos en las configuraciones de redes relevantes. (14)

El correcto desempeño del arquitecto de software parte del Modelo de Diseño y de Despliegue. Para ello se asignan componentes ejecutables a cada clase activa encontradas durante el diseño.

1.6 Conclusiones

El presente capítulo recogió el resultado de todo un análisis de los conceptos fundamentales relacionados con la producción de LIMS, además quedo evidenciada la necesidad de un Sistema de Manejo de la Información para los Laboratorios del CIGB.

Se llevaron a cabo estudios sobre posibles tecnologías y herramientas a utilizar para el desarrollo de la aplicación. La implementación del sistema estará basada en RUP como metodología para el proceso de desarrollo de Software, como lenguaje de modelado UML, como herramienta CASE Visual Paradigm, Eclipse como editor web orientado a la programación de páginas PHP, Symfony como framework de desarrollo, utilizando PHP5 y JavaScript como lenguajes de programación.

Capítulo 2
Implementación del Sistema

2.1 Introducción

En el presente capítulo se exponen los artefactos fundamentales que se obtienen como resultado del flujo de trabajo de implementación. Como base para la obtención de los diagramas de componentes documentados, así como el diagrama de despliegue del sistema y el plan de integración.
Se muestran fragmentos de código fuente relevante, ejemplos de validaciones y ejemplos visuales que reflejan los resultados concretos de dichas validaciones.

2.2 Requisitos funcionales

Para una correcta elaboración de la implementación, los desarrolladores centraron sus esfuerzos primeramente en el estudio de los modelos de negocio, de sistema y de diseño. El negocio permite la familiarización de los programadores con el entorno del módulo LIQ y con los procesos que se llevan a cabo en dicho laboratorio. El modelo del sistema proporciona los requisitos funcionales y las descripciones textuales de los CU que unido a los diagramas de clases del diseño representan el qué y el cómo será implementado el módulo.

En el trabajo de análisis y diseño precedente, fueron identificados ciento cuarenta requisitos funcionales (168 RF), los cuales fueron agrupados en veinte y ochos (28 CU) mediante la utilización del patrón CRUD (Create, Read, Update y Delete, por sus siglas en inglés). De éstos casos de uso se implementaron todos, incluyendo los siete arquitectónicamente significativos con que cuenta el sistema. A continuación se exponen los requerimientos funcionales correspondientes a dichos casos de usos:

R1. Gestionar libro de entrada de muestras (IQ).

R1.1 Registrar datos en el libro de entrada de muestras.

R1.2 Modificar.

R1.2.1 Modificar datos en el libro de entrada de muestras.

R1.2.2 Registrar traza.

R1.3 Generar reportes.

R1.4 Buscar y visualizar libro de entrada de muestras.

R1.5 Imprimir libro de entrada de muestras.

R2. Gestionar registro de control de inmunizaciones (SIC0745).

R2.1 Crear nuevo registro de control de inmunizaciones.

R2.2 Modificar.

R2.2.1 Modificar registro de control de inmunizaciones.

R2.2.2 Registrar traza.

R2.3 Buscar y visualizar registro de control de inmunizaciones.

R2.4 Imprimir registro de control de inmunizaciones.

R2.5 Generar reportes.

R3. Gestionar registro de control de extracciones (SIC0746).

R3.1 Crear nuevo registro de control de extracciones.

R3.2 Modificar.

R3.2.1 Modificar registro de control de extracciones.

R3.2.2 Registrar traza.

R3.3 Buscar y visualizar registro de control de extracciones.

R3.4 Imprimir registro de control de extracciones.

R3.5 Generar reportes.

R4. Gestionar registro de control de inmunizaciones y extracciones.

R4.1 Crear nuevo registro de control de inmunizaciones y extracciones.

R4.2 Modificar.

R4.2.1 Modificar registro de control de inmunizaciones y extracciones.

R4.2.2 Registrar traza.

R4.3 Buscar y visualizar registro de control de inmunizaciones y extracciones.

R4.4 Imprimir registro de control de inmunizaciones y extracciones.

R4.5 Generar reportes.

R5. Gestionar registro de determinación de la concentración de AcM anti-Hepatitis B (Método ELISA) (SIC-0193)

R5.1 Crear nuevo registro de determinación de concentración de AcM anti-Hepatitis B (Método ELISA) (SIC-0193)

R5.2 Modificar.

R5.2.1 Modificar registro de determinación de concentración de AcM anti-Hepatitis B (Método ELISA) (SIC-0193).

R5.2.2 Registrar traza.

R5.3 Buscar y visualizar registro de determinación de concentración de AcM anti-Hepatitis B (Método ELISA) (SIC-0193)

R5.4 Imprimir registro de determinación de concentración de AcM anti-Hepatitis B (Método ELISA) (SIC-0193)

R5.5 Generar reportes.

R6. Gestionar registro de determinación de la concentración de AcM anti-Hepatitis B por ELISA. (SIC-0135)

R6.1 Crear nuevo registro de determinación de concentración.

R6.2 Modificar.

R6.2.1 Modificar registro de determinación de concentración de AcM anti-Hepatitis B por ELISA. (SIC-0135).

R6.2.2 Registrar traza.

R6.3 Buscar y visualizar registro de determinación de concentración de AcM anti-Hepatitis B por ELISA. (SIC-0135)

R6.4 Imprimir registro de determinación de concentración de AcM anti-Hepatitis B por ELISA. (SIC-0135)

R6.5 Generar reportes.

R7. Gestionar registro de datos primarios de la técnica ELISA (RDPElisa).

R7.1 Crear nuevo registro de datos primarios de la técnica ELISA.

R7.2 Modificar.

R7.2.1 Modificar registro de datos primarios de la técnica ELISA.

R7.2.2 Registrar traza.

R7.3 Buscar y visualizar registro de datos primarios de la técnica ELISA.

R7.4 Imprimir registro de datos primarios de la técnica ELISA.

R7.5 Generar reportes.

R8. Gestionar Registro de Determinación de la Concentración de Inmunoglobulinas Tipo G (SIC-0115)

R8.1 Crear nuevo registro de Concentración de Inmunoglobulinas Tipo G.

R8.2 Modificar.

R8.2.1 Modificar registro de Concentración de Inmunoglobulinas Tipo G

R8.2.2 Registrar traza.

R8.3 Buscar y visualizar registro de Concentración de Inmunoglobulinas Tipo G.

R8.4 Imprimir registro de de determinación de Concentración de Inmunoglobulinas Tipo G.

R8.5 Generar reportes.

R9. Gestionar el Registro de Determinación de la concentración del Factor de Crecimiento Epidérmico. (SIC-0140)

R9.1 Crear nuevo registro de Determinación de la concentración del Factor de Crecimiento Epidérmico.

R9.2 Modificar.

R9.2.1 Modificar registro de Determinación de la concentración del Factor de Crecimiento Epidérmico.

R9.2.2 Registrar traza.

R9.3 Buscar y visualizar registro de Determinación de la concentración del Factor de Crecimiento Epidérmico.

R9.4 Imprimir registro de de Determinación de la concentración del Factor de Crecimiento Epidérmico.

R9.5 Generar reportes.

R10. Gestionar el Registro de Determinación de la concentración de impurezas proteicas de la cepa hospedera de proteínas recombinantes expresadas en E.coli. (SIC-0157)

R10.1 Crear nuevo registro de Determinación de la concentración de impurezas proteicas de la cepa hospedera de proteínas recombinantes expresadas en E.coli.

R10.2 Modificar.

R10.2.1 Modificar registro de Determinación de la concentración de impurezas proteicas de la cepa hospedera de proteínas recombinantes expresadas en E.coli.

R10.2.2 Registrar traza.

R10.3 Buscar y visualizar registro de Determinación de la concentración de impurezas proteicas de la cepa hospedera de proteínas recombinantes expresadas en E.coli.

R10.4 Imprimir registro de de Determinación de la concentración de impurezas proteicas de la cepa hospedera de proteínas recombinantes expresadas en E.coli.

R10.5 Generar reportes.

R11.Gestionar el Registro de Determinación de la concentración del alfa Interferón. (SIC-0194)

R11.1 Crear nuevo registro de Determinación de la concentración del alfa Interferón.

R11.2 Modificar.

R11.2.1 Modificar registro de Determinación de la concentración del alfa Interferón.

R11.2.2 Registrar traza.

R11.3 Buscar y visualizar registro de Determinación de la concentración del alfa Interferón.

R11.4 Imprimir registro de Determinación de la concentración del alfa Interferón.

R11.5 Generar reportes.

R12.Gestionar el Registro de Determinación de la concentración de proteínas contaminantes de levadura. (SIC-0753)

R12.1 Crear nuevo registro de Determinación de la concentración de proteínas contaminantes de levadura.

R12.2 Modificar.

R12.2.1 Modificar registro de Determinación de la concentración de proteínas contaminantes de levadura.

R12.2.2 Registrar traza.

R12.3 Buscar y visualizar registro de Determinación de la concentración de proteínas contaminantes de levadura.

R12.4 Imprimir registro de Determinación de la concentración de proteínas contaminantes de levadura.

R12.5 Generar reportes.

R13.Gestionar el Registro de Identificación de P64K. (SIC-0754P)

R13.1 Crear nuevo registro Identificación de P64K.

R13.2 Modificar.

R13.2.1 Modificar registro de Identificación de P64K.

R13.2.2 Registrar traza.

R13.3 Buscar y visualizar registro de Identificación de P64K.

R13.4 Imprimir registro de Identificación de P64K.

R13.5 Generar reportes.

R14. Gestionar el Registro de Determinación de la Concentración de Inmunoglobulinas Tipo G. (Método ELISA) (SIC-0137)

R14.1 Crear nuevo Registro de Determinación de la Concentración de Inmunoglobulinas Tipo G. (Método ELISA)

R14.2 Modificar.

R14.2.1 Modificar Registro de Determinación de la Concentración de Inmunoglobulinas Tipo G. (Método ELISA).

R14.2.2 Registrar traza.

R14.3 Buscar y visualizar Registro de Determinación de la Concentración de Inmunoglobulinas Tipo G. (Método ELISA)

R14.4 Imprimir Registro de Determinación de la Concentración de Inmunoglobulinas Tipo G. (Método ELISA)

R14.5 Generar reportes.

R15. Gestionar el Registro de Determinación de la Concentración de Interferón (Método ELISA) (SIC-0139)

R15.1 Crear nuevo Registro de Determinación de la Concentración de Interferón (Método ELISA).

R15.2 Modificar.

R15.2.1 Modificar Registro de Determinación de la Concentración de Interferón (Método ELISA).

R15.2.2 Registrar traza.

R15.3 Buscar y visualizar Registro de Determinación de la Concentración de Interferón (Método ELISA).

R15.4 Imprimir Registro de Determinación de la Concentración de Interferón (Método ELISA).

R15.5 Generar reportes.

R16. Gestionar el Registro de Determinación de la concentración del Factor de Crecimiento Epidérmico. (Método ELISA) (SIC-0140L)

R16.1 Crear nuevo Registro de Determinación de la concentración del Factor de Crecimiento Epidérmico. (Método ELISA).

R16.2 Modificar.

R16.2.1 Modificar Registro de Determinación de la concentración del Factor de Crecimiento Epidérmico. (Método ELISA).

R16.2.2 Registrar traza.

R16.3 Buscar y visualizar Registro de Determinación de la concentración del Factor de Crecimiento Epidérmico. (Método ELISA).

R16.4 Imprimir Registro de Registro de Determinación de la concentración del Factor de Crecimiento Epidérmico. (Método ELISA).

R16.5 Generar reportes.

R17. Gestionar el Registro de Determinación de la concentración de impurezas proteicas de la cepa hospedera de proteínas recombinantes expresadas en E.coli (Método de ELISA) (SIC-0157L)

R17.1 Crear nuevo Registro de Determinación de la concentración de impurezas proteicas de la cepa hospedera de proteínas recombinantes expresadas en E.coli (Método de ELISA).

R17.2 Modificar.

R17.2.1 Modificar Registro de Determinación de la concentración de impurezas proteicas de la cepa hospedera de proteínas recombinantes expresadas en E.coli (Método de ELISA)

R17.2.2 Registrar traza.

R17.3 Buscar y visualizar Registro de Determinación de la concentración de impurezas proteicas de la cepa hospedera de proteínas recombinantes expresadas en E.coli (Método de ELISA)

R17.4 Imprimir Registro de Registro de Determinación de la concentración de impurezas proteicas de la cepa hospedera de proteínas recombinantes expresadas en E.coli (Método de ELISA)

R17.5 Generar reportes.

R18. Gestionar el Registro de Determinación de la concentración de proteínas contaminantes de levadura (Método ELISA). (SIC-0753R)

R18.1 Crear nuevo Registro de Determinación de la concentración de proteínas contaminantes de levadura (Método ELISA).

R18.2 Modificar.

R18.2.1 Modificar Registro de Determinación de la concentración de proteínas contaminantes de levadura (Método ELISA).

R18.2.2 Registrar traza.

R18.3 Buscar y visualizar Registro de Determinación de la concentración de proteínas contaminantes de levadura (Método ELISA)

R18.4 Imprimir Registro de Determinación de la concentración de proteínas contaminantes de levadura (Método ELISA)

R18.5 Generar reportes.

R19. Gestionar el Registro de Identificación de P64K (Método ELISA). (SIC-0754)

R19.1 Crear nuevo Registro de Identificación de P64K (Método ELISA).

R19.2 Modificar.

R19.2.1 Modificar Registro de Identificación de P64K (Método ELISA).

R19.2.2 Registrar traza.

R19.3 Buscar y visualizar Registro de Identificación de P64K (Método ELISA)

R19.4 Imprimir Registro de Identificación de P64K (Método ELISA)

R19.5 Generar reportes.

R20. Gestionar registro de datos primarios de la técnica Western (RDPWestern).

R20.1 Crear nuevo registro de datos primarios de la técnica Western.

R20.2 Modificar.

R20.2.1 Modificar registro de datos primarios de la técnica Western.

R20.2.2 Registrar traza.

R20.3 Buscar y visualizar registro de datos primarios de la técnica Western.

R20.4 Imprimir registro de datos primarios de la técnica Western.

R20.5 Generar reportes.

R21. Gestionar registro de determinación de contaminantes de levadura (Western-Blot). (SIC-0117)

R21.1 Crear nuevo registro de determinación de contaminantes de levadura.

R21.2 Modificar.

R21.2.1 Modificar registro de determinación de contaminantes de levadura.

R21.2.2 Registrar traza.

R21.3 Buscar y visualizar registro de determinación de contaminantes de levadura.

R21.4 Imprimir registro de de determinación de contaminantes de levadura.

R21.5 Generar reportes.

R22. Gestionar el Registro para la Inmunoidentificación del Antígeno por la Técnica (Western-Blot). (SIC-0756)

R22.1 Crear nuevo registro para la Inmunoidentificación del Antígeno.

R22.2 Modificar.

R22.2.1 Modificar registro para la Inmunoidentificación del Antígeno.

R22.2.2 Registrar traza.

R22.3 Buscar y visualizar registro para la Inmunoidentificación del Antígeno.

R22.4 Imprimir registro para la Inmunoidentificación del Antígeno.

R22.5 Generar reportes.

R23. Gestionar el Registro de Determinación de Contaminantes por la Técnica (Western-Blot). (SIC-0117L)

R23.1 Crear nuevo Registro de Determinación de Contaminantes por la Técnica (Western-Blot).

R23.2 Modificar.

R23.2.1 Modificar Registro de Determinación de Contaminantes por la Técnica (Western-Blot).

R23.2.2 Registrar traza.

R23.3 Buscar y visualizar Registro de Determinación de Contaminantes por la Técnica (Western-Blot).

R23.4 Imprimir Registro de Determinación de Contaminantes por la Técnica (Western-Blot).

R23.5 Generar reportes.

R24. Gestionar el Registro para la Inmunoidentificación del Antígeno por la Técnica (Western-Blot). (SIC-0756L)

R24.1 Crear nuevo Registro para la Inmunoidentificación del Antígeno por la Técnica (Western-Blot).

R24.2 Modificar.

R24.2.1 Modificar Registro para la Inmunoidentificación del Antígeno por la Técnica (Western-Blot)

R24.2.2 Registrar traza.

R24.3 Buscar y visualizar Registro para la Inmunoidentificación del Antígeno por la Técnica (Western-Blot).

R24.4 Imprimir Registro para la Inmunoidentificación del Antígeno por la Técnica (Western-Blot).

R24.5 Generar reportes.

R25. Gestionar Registro de Control de la Técnica Inmuno-Dot (RDPInmuno).

R25.1 Crear nuevo Registro de Control de la Técnica Inmuno-Dot.

R25.2 Modificar.

R25.2.1 Modificar Registro de Control de la Técnica Inmuno-Dot.

R25.2.2 Registrar traza.

R25.3 Buscar y visualizar Registro de Control de la Técnica Inmuno-Dot.

R25.4 Imprimir Registro de Control de la Técnica Inmuno-Dot.

R25.5 Generar reportes.

R26. Gestionar Registro de Semicuantificación de proteínas contaminantes por Inmuno-Dot. (SIC-0141)

R26.1 Crear nuevo Registro de Semicuantificación de proteínas contaminantes por Inmuno-Dot.

R26.2 Modificar.

R26.2.1 Modificar Registro de Semicuantificación de proteínas contaminantes por Inmuno-Dot.

R26.2.2 Registrar traza.

R26.3 Buscar y visualizar Registro de Semicuantificación de proteínas contaminantes por Inmuno-Dot.

R26.4 Imprimir Registro de Semicuantificación de proteínas contaminantes por Inmuno-Dot.

R26.5 Generar reportes.

R27. Gestionar Registro de Semicuantificación de Proteínas contaminantes por Inmuno-Dot (SIC-1020)

R27.1 Crear nuevo Registro de Semicuantificación de proteínas contaminantes por Inmuno-Dot.

R27.2 Modificar.

R27.2.1 Modificar Registro de Semicuantificación de proteínas contaminantes por Inmuno-Dot.

R27.2.2 Registrar traza.

R27.3 Buscar y visualizar Registro de Semicuantificación de proteínas contaminantes por Inmuno-Dot.

R27.4 Imprimir Registro de Semicuantificación de proteínas contaminantes por Inmuno-Dot.

R27.5 Generar reportes.

R28. Gestionar Registro de Semicuantificación de Proteínas recombinantes por Inmuno-Dot (SIC-1021)

R28.1 Crear nuevo Registro de Semicuantificación de proteínas recombinantes.

R28.2 Modificar.

R28.2.1 Modificar Registro de Semicuantificación de proteínas recombinantes.

R28.2.2 Registrar traza.

R28.3 Buscar y visualizar Registro de Semicuantificación de proteínas.

R28.4 Imprimir Registro de Semicuantificación de proteínas recombinantes.

R28.5 Generar reportes.

2.3 Requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Estas propiedades hacen al producto atractivo, rápido y confiable.

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. Algún cambio que se quiera realizar en la aplicación, será tramitado por la dirección del proyecto por parte del CIGB y canalizado por los directivos de Producción de la Universidad de las Ciencias Informáticas.

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.

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.

Software: Se deberá disponer, para instalar la aplicación, del Sistema Operativo Windows XP o cualquier distribución de Linux. Las computadoras clientes de los usuarios accederán al sistema usando uno de los siguientes navegadores: Internet Explorer 5.5 o superior, Mozilla 1.7 o superior. Para el servidor de la aplicación el sistema operativo recomendado es Windows Server 2003 o superior o Linux. Se debe instalar un servidor Web Apache 1.3 o superior.

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 8.2.3, así como Symfony como framework de desarrollo.

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.

2.4 Patrón de arquitectura y diseño

2.4.1 Modelo Vista Controlador (MVC)

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Symfony está basado en dicho patrón clásico de diseño web. **(Ver Anexo 7)**

MVC divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- El **modelo** es el objeto de la aplicación, o sea es la representación específica de la información con la cual el sistema opera.
- La **vista** es la presentación en pantalla. Este presenta el modelo en un formato adecuado para interactuar con los usuarios.
- El **controlador** responde a eventos o acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

MVC permite reflejar un diseño que separa vistas de modelos. Pero el diseño de MVC es aplicable a un problema más general: separar objetos, por lo que el cambio de uno de ellos puede afectar a cualquier número de los otros, sin que se requiera del objeto cambiado para conocer los detalles de otros.

2.4.2 Patrones GRASP

En este epígrafe se explica cómo se utilizaron los patrones de diseño GRASP durante el desarrollo de la aplicación y en donde se manifiestan estos patrones en la misma.

Creador: En la clase Actions se encuentran todas las acciones definidas y además es donde se ejecutan cada una de ellas. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase Actions es "creador" de dichas entidades. Este patrón se puede ver claramente dentro de la clase Action del Libro de Entrada de Muestras, donde la clase tiene la responsabilidad de crear las diferentes acciones (Crear, Modificar, Visualizar, entre otras) que se ejecutan.

Experto: Este es uno de los más utilizados, puesto que Propel es la librería externa que utiliza Synfony 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. Con la utilización de este patrón se pudieron definir cuáles eran las clases responsable para la creación de un objeto, la cual debe ser la que conoce toda la información necesaria para crearlo. Esto permite que se mantenga el encapsulamiento de la información.

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. Este patrón indica que la información que se almacena en la clase debe ser coherente y además debe de estar relacionada con la misma, esto se puede ver claramente en la aplicación donde las acciones que se definen para las diferentes Actions están relacionadas con la misma.

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. El patrón controlador de tal forma es el que recibe los datos del usuario y el que los envía a las distintas clases según el método llamado.

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

2.4.3 Patrones GOF que implementa Symfony

El framework Symfony utiliza los patrones de diseño GOF, de los cuales en el trabajo se utilizaron los siguientes:

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. Por ejemplo en la aplicación cuando se desea crear un nuevo objeto para una petición, se

busca nombre de la clase que se debe utilizar para esta tarea y se crea un objeto de esta clase para poder trabajar con el objeto de otra clase.

En la categoría Estructurales:

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

Composite (Objeto compuesto): Permite tratar objetos compuestos como si se tratase de uno 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.

2.5 Descripción del sistema

Los casos de usos implementados fueron un total de veinte y ochos (28 CU). Se obtuvo un sistema que está conectado a una Base de Datos (DB en inglés) que utiliza como gestor de base de datos a PostgreSQL.

A través del sistema en esta DB se van a almacenar todos los datos recogidos por los analistas del laboratorio y posteriormente mediante consultas SQL se extrae la información guardada, con el objetivo de realizar comparaciones entre los resultados de las técnicas que se realizan en el LIQ. La DB está formada por tablas que se relacionan por medio de sus llaves primarias, que le permiten al sistema mostrar en tiempo de ejecución varios datos como (el folio, el producto, el número de parte, entre otros).

2.6 Diagramas de Componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, librerías compartidas, módulos, ejecutables, o paquetes. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema. En él se situarán librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema.

También se muestra como están distribuidos los componentes según el patrón arquitectónico mencionado en epígrafes anteriores, que utiliza Symfony como paradigma en su organización interna. En la vista de la aplicación aparece el layout.php relacionado con todos los archivos Success.php, así como la parte del código JavaScript y las hojas de estilo en cascada (*Cascading Style Sheets*, CSS por sus siglas en inglés) que se programan del lado del cliente**.**

La action.class.php se encuentra ubicada en la capa perteneciente al controlador, será controlada por la sfFrontWebController.class.php o controlador frontal, pues a través de este se van a ejecutar todas las peticiones los usuarios. La action.class.php está relacionada con todos los archivos Success.php de la vista y contiene todos los métodos y operaciones a realizar que serán mostrados en la vista. La action.class.php se vincula también con los archivos Seguridad.yml y Validación.yml, pues es la encargada de la seguridad y las validaciones de la aplicación. De igual forma trabaja sobre el modelo que es la capa que contiene la persistencia de los datos.

El modelo contiene todas las clases Peer.php o .php sola, así como las BasePeer.php y las Base.php. Estas últimas por medio del subsistema Propel que utiliza Symfony acceden a la BD Inmunoquímica
Symfony utiliza Propel como el mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping) y Propel utiliza Creole como la capa de abstracción de bases de datos. La principal ventaja que aporta el ORM es la reutilización, permitiendo llamar a los métodos de un objeto de datos desde varias partes de la aplicación e incluso desde diferentes aplicaciones. Estos dos componentes están completamente integrados en Symfony, por lo que se pueden considerar como una parte más del framework. Su sintaxis y sus convenciones se han adaptado de tal forma que difieran lo menos posible de las de Symfony. A continuación se

muestra uno de los diagramas de componentes correspondiente a uno de los casos de uso críticos del sistema.

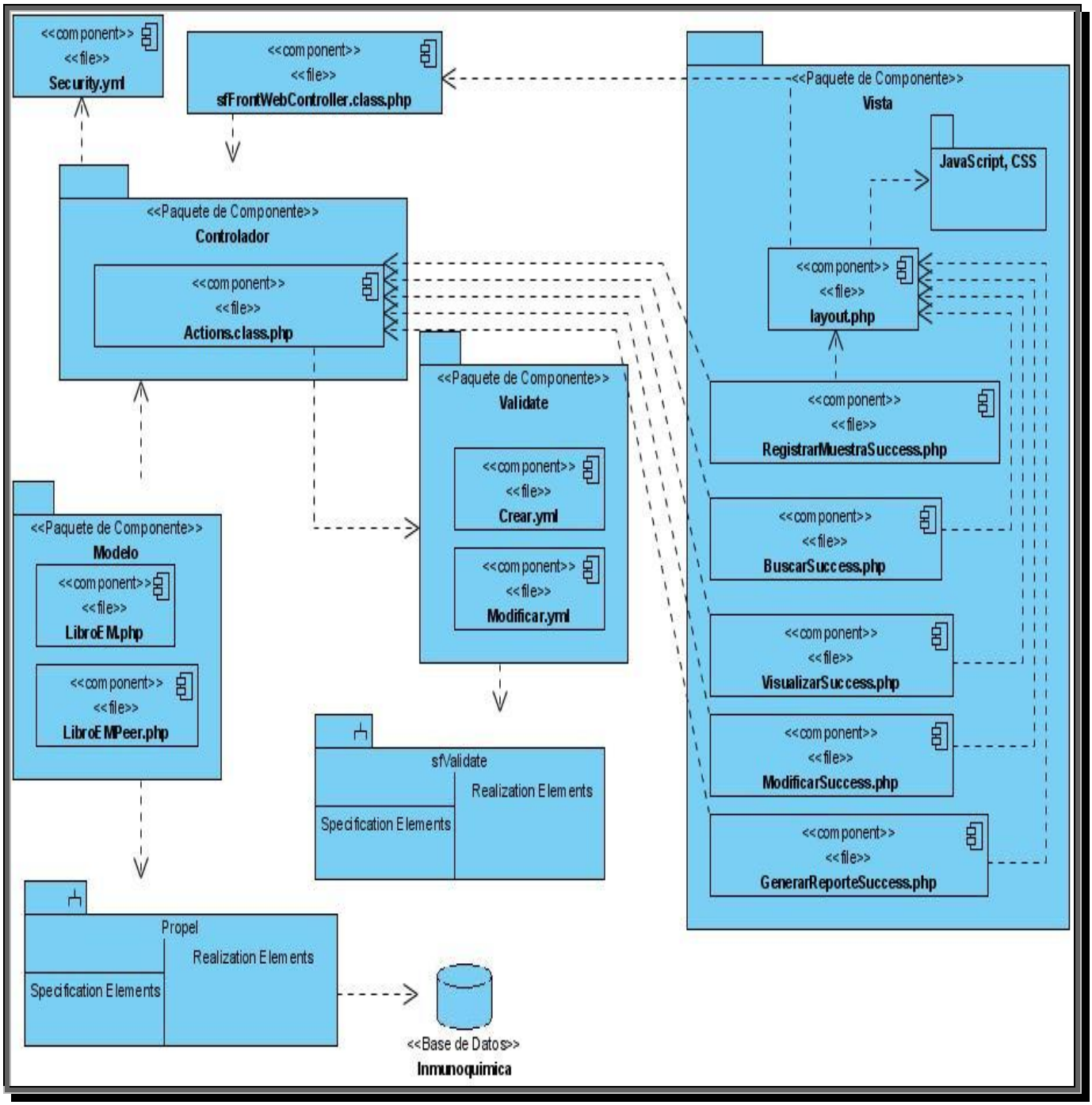


Figura 2: DC Gestionar Libro Entrada de Muestra

2.7 Diagrama de despliegue

El modelo de despliegue representa un modelo de objetos que muestra cómo se distribuye físicamente el sistema, teniendo en cuenta la funcionalidad entre cada nodo de cómputo. Este modelo es utilizado como base para la realización de las actividades de diseño e implementación.

Se puede observar lo siguiente sobre el modelo de despliegue:

- Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo hardware similar.
- Los nodos poseen relaciones que representan medios de comunicación entre ellos, tales como Internet, Intranet, bus y similares.
- El modelo de despliegue puede describir diferentes configuraciones de red, incluidas las configuraciones para pruebas y para simulación.
- La funcionalidad (los procesos) de un nodo se define por los componentes que se distribuyen sobre ese nodo.
- El modelo de despliegue en sí mismo representa una correspondencia entre la arquitectura de software y la arquitectura del sistema (hardware). (15)

En la siguiente figura se muestra el diagrama de despliegue con los componentes de implementación. De forma general se representan las plantillas SUCCESS que tienen una relación de dependencia con las ACTIONS y a su vez estas se relacionan con las PEER que se generan en el modelo. En el servidor de Base de Datos se almacenan los datos, todo el flujo de las acciones comienza cuando en la computadora cliente desde un navegador Web se solicita manejar alguna información contenida o no en la BD, así como crearla o visualizarla en algún momento.

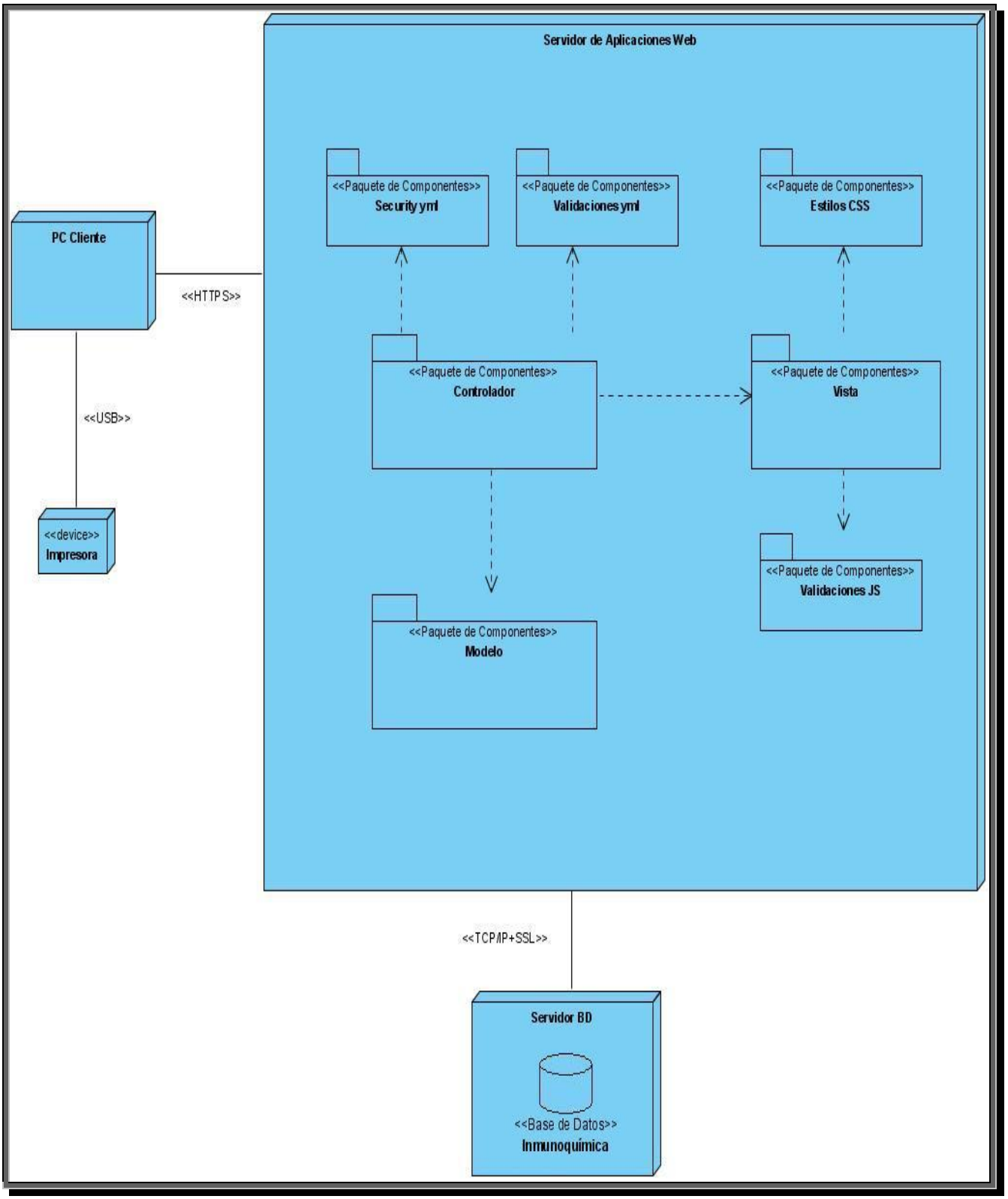


Figura 3: Diagrama de despliegue.

2.8 Ejemplos de Código

A continuación se presenta fragmentos del código del caso de uso Gestionar Registro de Datos Primarios Western-Blot.

Modificar Registro de Datos Primarios Action

La funcionalidad de este método es recibir una variable de estado por URL, que dicha variable de estado lo que contiene es el identificador de la planilla que se desea modificar, luego se busca mediante esta variable en la base de datos con una funcion que trae definida Symfony (retrieveByPK) el objeto que corresponde con el mismo identificador. Luego mediante el método Criteria() se busca en las tablas de la base de datos todos los valores que pertenecen a esa planilla y luego se le muestra el template ModificarSuccess correspondiente que se desea modificar con los datos que estaban insertados. Luego el usuario selecciona los datos que desea modificar y estos datos que se modificaron se actualizan en la base de datos y se redirecciona al usuario hacia la pagina donde se encuentran las acciones para que el usuario seleccione la acción que desea realizar.

```
public function executeModificar()
  {
    if($this->getRequest()->getMethod()!= sfRequest::POST)
    {
      $this->rdpw=RdpWesternPeer::retrieveByPK($this-
>getRequestParameter('estado'));
      $this->getUser()->setAttribute('rdpw', $this->rdpw);

      $r= new Criteria();
      $r->add(RdpWesternPeer::ID_RDP_WESTERN,$this->getRequestParameter('estado'));
      $r->addJoin(SolucionesPeer::ID_RDP_WESTERN,RdpWesternPeer::ID_RDP_WESTERN);
            $r->addAscendingOrderByColumn(SolucionesPeer::ID_SOLUCIONES);
            $this->solucion=SolucionesPeer::doSelect($r);
            $this->getUser()->setAttribute('soluciones', $this->solucion);

      $d=new Criteria();
      $d->add(RdpWesternPeer::ID_RDP_WESTERN,$this->getRequestParameter('estado'));
```

```
      $d->addJoin(ReactivosIqPeer::ID_RDP_WESTERN,RdpWesternPeer::ID_RDP_WESTERN);
      $d->addAscendingOrderByColumn(ReactivosIqPeer::ID_REACTIVOS);
      $this->reactivosiq=ReactivosIqPeer::doSelect($d);
      $this->getUser()->setAttribute('reactivos', $this->reactivosiq);

      $e =  new Criteria();
      $e->add(RdpWesternPeer::ID_RDP_WESTERN, $this->getRequestParameter('estado'));
      $e-
>addJoin(ReactivosBiologicosIqPeer::ID_RDP_WESTERN,RdpWesternPeer::ID_RDP_WESTERN);
      $e-
>addAscendingOrderByColumn(ReactivosBiologicosIqPeer::ID_REACTIVOS_BIOLOGICOS);
      $this->reactivosbiq = ReactivosBiologicosIqPeer::doSelect($e);
      $this->getUser()->setAttribute('reactivosbiq', $this->reactivosbiq);

      $mat=new Criteria();
      $mat->add(RdpWesternPeer::ID_RDP_WESTERN,$this-
>getRequestParameter('estado'));
      $mat-
>addJoin(MaterialesCriticosIqPeer::ID_RDP_WESTERN,RdpWesternPeer::ID_RDP_WESTERN);
      $mat->addAscendingOrderByColumn(MaterialesCriticosIqPeer::ID_MATERIAL_IQ);
      $this->materiales=MaterialesCriticosIqPeer::doSelect($mat);
      $this->getUser()->setAttribute('materiales', $this->materiales);

      return sfView::SUCCESS;
    }
    else
    {
      $rdp_obj = $this->getUser()->getAttribute('rdpw');
      $materiales = $this->getUser()->getAttribute('materiales');
      $soluciones = $this->getUser()->getAttribute('soluciones');
      $reactivos = $this->getUser()->getAttribute('reactivos');
      $reactivosbiq = $this->getUser()->getAttribute('reactivosbiq');

      $rdp_obj->setFolio($this->getRequestParameter('folio'));
      $rdp_obj->setFecha($this->getRequestParameter('fecha'));
      $rdp_obj->setAcrilamidagel($this->getRequestParameter('acrilamida'));
```

```
	$rdp_obj->setNombreTecnica($this->getRequestParameter('objetivo'));
	$rdp_obj->setNumeroTecnica($this->getRequestParameter('numero'));
	$rdp_obj->setObservaciones($this->getRequestParameter('observaciones'));
	$rdp_obj->setRealizadoPor($this->getRequestParameter('realizadoP'));
	$rdp_obj->setRecibidoPor($this->getRequestParameter('recibidoP'));
	$rdp_obj->setRevisadoPor($this->getRequestParameter('revisadoP'));
	$rdp_obj->setFechaRealizado($this->getRequestParameter('fechaRealizado'));
	$rdp_obj->setFechaRecibido($this->getRequestParameter('fechaRecibido'));
	$rdp_obj->setFechaRevisado($this->getRequestParameter('fechaRevisado'));
	if ($this->getRequestParameter('registroTerminado') == 1)
		$rdp_obj->setTerminado(1);
	else
		$rdp_obj->setTerminado(0);
	$rdp_obj->save();

	$arraySoluciones=array(array($this->getRequestParameter('numeroLote1'), $this-
>getRequestParameter('fechaVencimiento1')),
		array($this->getRequestParameter('numeroLote2'), $this-
>getRequestParameter('fechaVencimiento2')),
		array($this->getRequestParameter('numeroLote3'), $this-
>getRequestParameter('fechaVencimiento3')),
		array($this->getRequestParameter('numeroLote4'), $this-
>getRequestParameter('fechaVencimiento4')),
		array($this->getRequestParameter('numeroLote5'), $this-
>getRequestParameter('fechaVencimiento5')),
		array($this->getRequestParameter('numeroLote6'), $this-
>getRequestParameter('fechaVencimiento6')),
		array($this->getRequestParameter('numeroLote7'), $this-
>getRequestParameter('fechaVencimiento7')),
		array($this->getRequestParameter('numeroLote8'), $this-
>getRequestParameter('fechaVencimiento8')),
		array($this->getRequestParameter('numeroLote9'), $this-
>getRequestParameter('fechaVencimiento9')),
		array($this->getRequestParameter('numeroLote10'), $this-
>getRequestParameter('fechaVencimiento10')),
```

```
            array($this->getRequestParameter('numeroLote11'), $this-
>getRequestParameter('fechaVencimiento11')),
            array($this->getRequestParameter('numeroLote12'), $this-
>getRequestParameter('fechaVencimiento12')),
            array($this->getRequestParameter('numeroLote13'), $this-
>getRequestParameter('fechaVencimiento13')),
            array($this->getRequestParameter('numeroLote14'), $this-
>getRequestParameter('fechaVencimiento14')),
            array($this->getRequestParameter('numeroLote15'), $this-
>getRequestParameter('fechaVencimiento15')),
            array($this->getRequestParameter('numeroLote16'), $this-
>getRequestParameter('fechaVencimiento16')),
            array($this->getRequestParameter('numeroLote17'), $this-
>getRequestParameter('fechaVencimiento17')));

      SolucionesPeer::ModificarRDPWestern($arraySoluciones, $soluciones);

      $arrayReactivos=array(array($this->getRequestParameter('cantidadVolumen1'),
$this->getRequestParameter('lote1')),
            array($this->getRequestParameter('cantidadVolumen2'), $this-
>getRequestParameter('lote2')),
            array($this->getRequestParameter('cantidadVolumen3'), $this-
>getRequestParameter('lote3')),
            array($this->getRequestParameter('cantidadVolumen4'), $this-
>getRequestParameter('lote4')),
            array($this->getRequestParameter('cantidadVolumen5'), $this-
>getRequestParameter('lote5')));

      ReactivosIqPeer::ModificarReactivosRdpWestern($arrayReactivos, $reactivos);

      $arrayReactivosBio=array(array($this-
>getRequestParameter('tipoAnticuerpo1'),$this-
>getRequestParameter('tipoAnticuerpo2'),$this-
>getRequestParameter('tipoAnticuerpo3'),$this-
>getRequestParameter('tipoAnticuerpo4'),$this-
```

```
>getRequestParameter('tipoAnticuerpo5'),$this-
>getRequestParameter('fechaAnticuerpo')),
            array($this->getRequestParameter('tipoConjugado1'),$this-
>getRequestParameter('tipoConjugado2'),$this-
>getRequestParameter('tipoConjugado3'),$this-
>getRequestParameter('tipoConjugado4'),$this-
>getRequestParameter('tipoConjugado5'),$this-
>getRequestParameter('fechaConjugado')));


      ReactivosBiologicosIqPeer::ModificarReactivosBiologicosRdpWestern($arrayReacti
vosBio, $reactivosbiq);

      $materiales[0]->setNumeroParte($this->getRequestParameter('nitrocelulosa1'));
      $materiales[0]->setNumeroLote($this->getRequestParameter('nitrocelulosa2'));
      $materiales[0]->save();

      $this->rdpw = $rdp_obj;
      $this->redirect('RDP_Western/Modificar?estado='.$rdp_obj->getIdRdpWestern());
    }
  }
```

Soluciones Peer

En este fragmento de código se ve como se interactuó con las clases del modelo para hacer más fácil la implementación y reducir el código de la acción para su mejor entendimiento, así como optimizar la rapidez de la aplicación asiendo de esta más eficaz.

```
class SolucionesPeer extends BaseSolucionesPeer
{
      public static function InsertarRDPWestern($array)
      {
            $i = 0;
            foreach ($array as $key => $value)
            {
```

```
                $soluciones = new Soluciones();
                  $arr = $array[$i];

                  $soluciones->setNombreSolucion($arr[0]);
                  $soluciones->setNumeroParte($arr[1]);
                  $soluciones->setNumeroLote($arr[2]);
                  $soluciones->setFechaVencimiento($arr[3]);
                  $soluciones->setIdRdpWestern($arr[4]);

                  $soluciones->save();

                  $i++;
            }
      }
      public static function ModificarRDPWestern($array, $soluciones)
      {
            $i = 0;
            foreach ($array as $key => $value)
            {
                  $arr = $array[$i];
                  $soluciones[$i]->setNumeroLote($arr[0]);
                  $soluciones[$i]->setFechaVencimiento($arr[1]);
                  $soluciones[$i]->save();

                  $i++;
            }
      }
}
```

2.9 Métodos de validación y manejo de errores con Symfony

La validación de los datos de la acción es una tarea repetitiva y tediosa. Symfony provee distintos métodos para validar acciones (16):

- Por acción: Esta validación se realiza cuando un usuario hace una petición a miAccion, este busca siempre el primer método llamado validateMiAccion(). Si lo encuentra, ejecuta ese método. El valor de retorno de esta validación determina el siguiente método que se ejecuta: si devuelve verdadero, entonces se ejecuta el método executeMiAccion(); en otro caso, se ejecuta el método handleErrorMiAccion() de las acciones del módulo, allí se define que hacer (volver al formulario mostrando errores, intentar corregirlos).**(Ver Anexo 8)**
- Por configuración: La validación por configuración se ejecuta por defecto si el request proviene de un método POST, pero también se pueden validar acciones por GET.

2.9.1 Validadores estándar de Symfony

Symfony contiene varios validadores ya definidos y que se pueden utilizar directamente en los formularios: (17)

- sfStringValidator
- sfNumberValidator
- sfEmailValidator
- sfUrlValidator
- sfRegexValidator
- sfCompareValidator
- sfPropelUniqueValidator
- sfFileValidator
- sfCallbackValidator

2.10 Seguridad

Todo sistema de software debe brindar la posibilidad de administrar a los diferentes usuarios que interactúen con el mismo, así como restringir las acciones de los usuarios que no tengan los diferentes privilegios para poder usar el sistema.

En el módulo Inmunoquímica, para que un usuario pueda realizar cualquier acción sobre el sistema primeramente debe autenticarse y luego el sistema verifica las credenciales del mismo y de acuerdo a

ellas se le habilitan los privilegios necesarios para que este pueda realizar las acciones asociadas con sus permisos. Los privilegios en el módulo están compuestos por dos partes:

- Los usuarios deben estar autenticados en el sistema.
- Los usuarios deben tener las credenciales necesarias para realizar cualquier acción.

Para restringir el acceso a una acción se crea un archivo de configuración YAML llamado security.yml en el directorio config/ del módulo. En este archivo, se especifican los requerimientos de seguridad que los usuarios deben satisfacer para cada acción o para todas las acciones. Las acciones no incluyen restricciones de seguridad por defecto, por lo que todas las acciones son accesibles por todos los usuarios. Si existe un archivo security.yml, Symfony busca por el nombre de la acción y si existe, verifica que se satisfagan los requerimientos de seguridad (18).

En el sistema implementado cuando un usuario intenta acceder a una zona restringida el mismo verifica si el usuario se encuentra autenticado en el sistema y si tiene los permisos necesarios para realizar cualquier acción, en caso de que no se encuentre autenticado se redirecciona al usuario para la pagina de login para que se autentique.

Symfony incluye este sistema que se aplica a todos los datos mostrados mediante las variables de las plantillas. El mecanismo se activa mediante un único parámetro en el archivo settings.yml de la aplicación. Se configura de forma global para toda la aplicación en el archivo settings.yml.

2.11 Pruebas

Las pruebas son uno de los mayores avances en la programación, estas tienen como objetivo la detección de defectos en el software asegurando así la calidad de la aplicación. Las mismas se usan para determinar si los sucesos ocurren en un momento determinado durante la ejecución del software y se comporta de forma correcta.

Las pruebas unitarias aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. El método de prueba utilizado para detectar defectos al software desarrollado fue el de de Caja Blanca. Permitió asegurar que las operaciones internas se ajustan a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

Framework de pruebas Lime

En el ámbito de PHP existen muchos frameworks para crear pruebas unitarias, siendo los más conocidos PhpUnit y SimpleTest. Symfony incluye su propio framework llamado Lime. Se basa en la librería Test::More de Perl y es compatible con TAP(*"Test Anything Protocol"*).

Las pruebas unitarias de Symfony son archivos PHP que se encuentran en el directorio test/unit/ del proyecto. Su sintaxis es sencilla y fácil de leer. Se le realizaron pruebas al método que calcula el volumen de la proteína a inmunizar que se encuentra en la planilla Registro de control de inmunizaciones. Desde la clase sic0745Test.php se llama al objeto lime_test de Lime y se utilizó el método "is" que compara 2 valores y la prueba pasa si los 2 son iguales. El código de la clase queda de la siguiente forma:

```
<?php
include(dirname(__FILE__).'/../bootstrap/unit.php');

$t = new lime_test(3, new lime_output_color());

$t->is(Sic0745Peer::Calculo(3,44,34),660,'Valor de actividad calculado
correctamente');
$t->is(Sic0745Peer::Calculo(0,34,4),70,'Valor de actividad calculado
correctamente');
$t->is(Sic0745Peer::Calculo(2,3,4),6,'Valor de actividad calculado correctamente');
?>
```

En la tabla que se presenta a continúan se representan los valores de entrada y de salida de las pruebas que se le hicieron al método que calcula el volumen con el framework Lime.

class Sic0745Peer				
Método	**Método de Prueba**	**Recibe**	**Esperado**	**Salida**
Calculo(a, b, c)	is	3,44,45	660	660
Calculo(a, b, c)	is	0,34,4	70	Not ok

Calculo(a, b, c)	is	2,3,4	6	6

Tabla 1: Prueba de unidad Lime: Valores esperados y de entrada.

Luego de haber realizado la prueba se observo que en el segundo cálculo del denominador se encontraba indefinido y en el tercero se pudo realizar la corrección del mismo y de esta forma darle solución al error encontrado en el código.

2.12 Conclusiones

Como resultado de este capítulo se obtuvo un diagrama de componentes por cada CU implementado. También se desarrollo el diagrama de despliegue donde quedaron modelados los nodos en los que se distribuye la aplicación especificando las conexiones y los protocolos que los unen, se hizo referencia y se realizó una pequeña descripción de la utilidad de los patrones que se utilizaron para la construcción de la aplicación. Se muestran también algunos segmentos del código fuente significativos para la implementación, así como ejemplos de validaciones que usa Symfony y de los cuales se utilizaron en la aplicación.

CONCLUSIONES GENERALES

Como resultado del estudio de los sistemas de gestión de información de laboratorios y los resultados obtenidos en investigaciones precedentes se puede concluir que la presente investigación logró darle cumplimiento al objetivo propuesto en el trabajo; para alcanzar este resultado se desarrollo:

- Un estudio de la documentación obtenida en iteraciones anteriores y de los diferentes software que existen para el manejo de información de los laboratorios.

- Una exhaustiva investigación para la utilización acertada de tecnologías y herramientas adecuadas para el desarrollo de la aplicación, estas fueron, PHP como lenguaje de programación, PostgreSQL como gestor de Bases de Datos y Symfony como marco de trabajo (framework); entre otras.

- La implementación de la aplicación, obteniendo un producto funcional para el módulo Inmunoquímica del LIMS de la Dirección de Calidad del CIGB que cumple con los requisitos identificados y satisfaciendo las necesidades del cliente, se mostró en el documento ejemplo del código que se obtuvo durante el desarrollo de la aplicación.

Gracias a las tareas que se plantearon al inicio del presente trabajo se logró todo el proceso de implementación de la aplicación Web, llegando a lograr el objetivo final que no era más que desarrollar la implementación del módulo Inmunoquímica, para el Sistema de Gestión de la Información de los Laboratorios de la Dirección de Calidad del CIGB**.**

RECOMENDACIONES

A partir del producto obtenido se recomienda:

- Realizar la integración del módulo Inmunoquímica con los restantes módulos que formarán parte del LIMS de Calidad del CIGB.
- Extender la aplicación para otros polos científicos del país y en general a entidades que lo requieran para mejorar y agilizar la calidad de sus productos.
- Realizar la ayuda para el módulo Inmunoquímica.

REFERENCIAS BIBLIOGRÁFICAS

1. **CIGB.** Departamento de Aseguramiento de la Calidad. [En línea] [Citado el: 20 de noviembre de 2008.] http://www.cigb.edu.cu/index.php?option=com_content&task=view&id=55&Itemid=121.
2. **CIGB.** Departamento de Control de Calidad. [En línea] [Citado el: 20 de noviembre de 2008.] http://www.cigb.edu.cu/index.php?option=com_content&task=view&id=54&Itemid=120.
3. **Medio Ambiente** Online. [En línea] [Citado el: 3 de marzo de 2009.] http://www.medioambienteonline.com/site/root/info_and_tools/software/3423.html.
4. **Hernández, Enrique.** Departamento de Informática y Sistemas de Computación. [En línea] [Citado el: 4 de marzo de 2009.] http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.
5. **LinuxParaTodos.** [En línea] [Citado el: 4 de marzo de 2009.] http://www.linuxparatodos.net/portal/staticpages/index.php?page=servidor-web#.
6. **Quiñones, E.** Introduccion al PostgreSQL. [En línea] [Citado el: 6 de marzo de 2009.] http://www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf.
7. **Jordisan.net.** [En línea] [Citado el: 6 de marzo de 2009.] http://jordisan.net/blog/2006/que-es-un-framework/.
8. **LibrosWeb.es.** [En línea] [Citado el: 7 de marzo de 2009.] http://www.librosweb.es/symfony_1_0/capitulo1/symfony_en_pocas_palabras.html.
9. **Patrones de diseño de software.** [En línea] [Citado el: 10 de mayo de 2009.] http://www.proactiva-calidad.com/java/patrones/index.html
10. **Ministerio del Poder Popular para las Comunicaciones y la Informatica.** MeRinde. [En línea] [Citado el: 7 de marzo de 2009.] http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=138&Itemid=193.
11. **Universidad Politécnica de Valencia.** Departamento de Sistemas Informáticos y Computación. [En línea] [Citado el: 1 de abril de 2009.] https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducción%20a%20RUP.doc.
12. **Jacobson, I, Booch, G y Rumbaugh, J.** *EL PROCESO DE DESARROLLO DE SOFTWARE.* Madrid : PEARSON EDUCACION, S.A, 2000. pág. 260. 84-7829-036-2.
13. **Jacobson, I, Booch, G y Rumbaugh, J**. *EL PROCESO DE DESARROLLO DE SOFTWARE.* Madrid : PEARSON EDUCACION, S.A, 2000. págs. 257-258. 84-7829-036-2.
14. **Jacobson, I, Booch, G y Rumbaugh, J**. *EL PROCESO DE DESARROLLO DE SOFTWARE.* Madrid : PEARSON EDUCACION, S.A, 2000. pág. 268. 84-7829-036-2.

15. **Jacobson.** *El proceso Unificado de desarrollo de software.* La Habana : Félix Varela, 2004.

16. **LibrosWeb.es.** [En línea] [Citado el: 6 de junio de 2009.] http://www.librosweb.es/symfony_1_0/capitulo6/metodos_de_validacion_y_manejo_de_errores.html

17. **LibrosWeb.es.** [En línea] [Citado el: 10 de abril de 2009.] http://www.librosweb.es/symfony_1_0/capitulo10/validacion_de_formularios.html.

18. **LibrosWeb.es.** [En línea] [Citado el: 6 de junio de 2009.] http://www.librosweb.es/symfony_1_0/capitulo6/seguridad_de_la_accion.html.

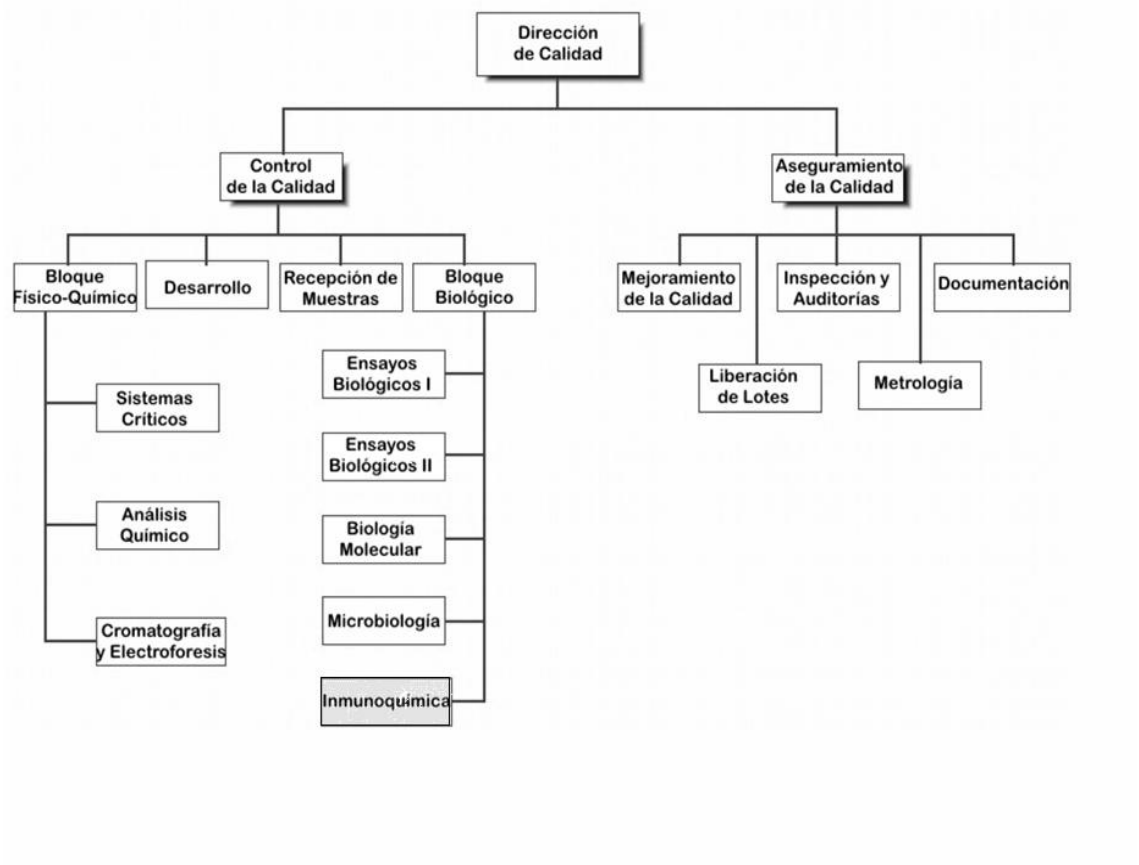
BIBLIOGRAFÍA

1. **CiberSociedad.** 2008. [En Línea] http://www.cibersociedad.net/archivo/articulo.php?art=218
2. **Ciberaula.** 2008. [En Línea] http://linux.ciberaula.com/articulo/linux_apache_intro/
3. **STARLIMS, C**. *¿Quiénes Somos?*, 2008. [En Línea] http://www.starlims.com/es/company/index.htm
4. **STARLIMS CORPORATION**. *STARLIMS Nota Técnica*, 2008. [En Línea] http://www.starlims.com/brochureinSpanish/EmpowerSp.pdf
5. **AUTOSCRIBE.** *LIMS & Successful Scientific & Business Software Solutions from Autoscribe*, 2008. [En Línea] http://www.autoscribe.co.uk
6. **MEDIO AMBIENTE ONLINE.** *Portada: Información y herramientas para el sector ambiental: Software para el medio ambiente*, 2009. [En Línea] http://www.medioambienteonline.com/site/root/info_and_tools/software/index.html
7. **LabSoftLIMS.** LabSoftLIMS by Computing Solution, Inc. 2008. [En línea] http://www.labsoftlims.com/US/computing_solutions.htm
8. **LinuxParaTodos.** 2009. [En Línea] http://www.linuxparatodos.net/portal/staticpages/index.php?page=servidor-web
9. **Jacobson, I., Booch, G., & Rumbaugh, J.** EL PROCESO DE DESARROLLO DE SOFTWARE **PEARSON EDUCACION,** S.A Madrid. 2000 ISBN: 84-7829-036-2
10. **Viera Achang, Jose Rafael y Peña Gonzalez, Loismarx.** LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Implementación del Módulo de Análisis Químico. Ciudad de La Habana : Ciudad Habana : s.n., 2008
11. **Eclipse.** 2009. [En línea] http://www.eclipse.org/home/categories/index.php?category=frameworks
12. **Visual Paradigm.** Visual Paradigm. 2009. [En línea] http://www.visual-paradigm.com/product/vpuml
13. **Hernádez , Enrique**. DISCA Lenguaje Unificado de Modelado (UML). DISCA. 2008. [En línea] www.disca.upv.es.
14. **Quiñones, E**. *Introduccion a PostgreSQL*. 2009. [En línea] http://www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf
15. **LibrosWeb.es.** Symfony. 2008. [En Línea] http://www.librosWeb.es/symfony/capitulo1/symfony_en_pocas_palabras.html

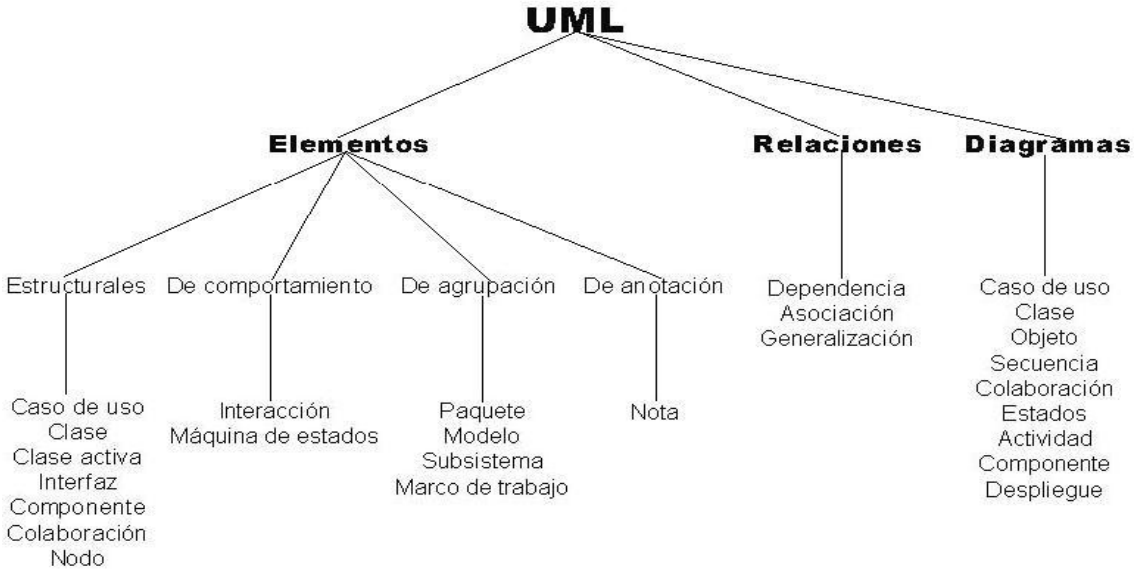
16. **¿Qué es un framework?** ¿Qué es un framework? 2009. [En Línea] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf
17. Jacobson. *El proceso Unificado de desarrollo de software.* La Habana : Félix Varela, 2004.
18. BIKA Jabs System. [En línea] 2005-2009. http://www.bikalabs.com/
19. **Roman Konertz, PD Dr. Ludwig Eichinger and Dr. Budi Tunggal.** OpenLIMS. [En línea] http://www.open-lims.org/
20. open source laboratory automation &informatics. [En línea] http://www.labmatica.com/

ANEXOS

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



Anexo 2: Estructura de UML

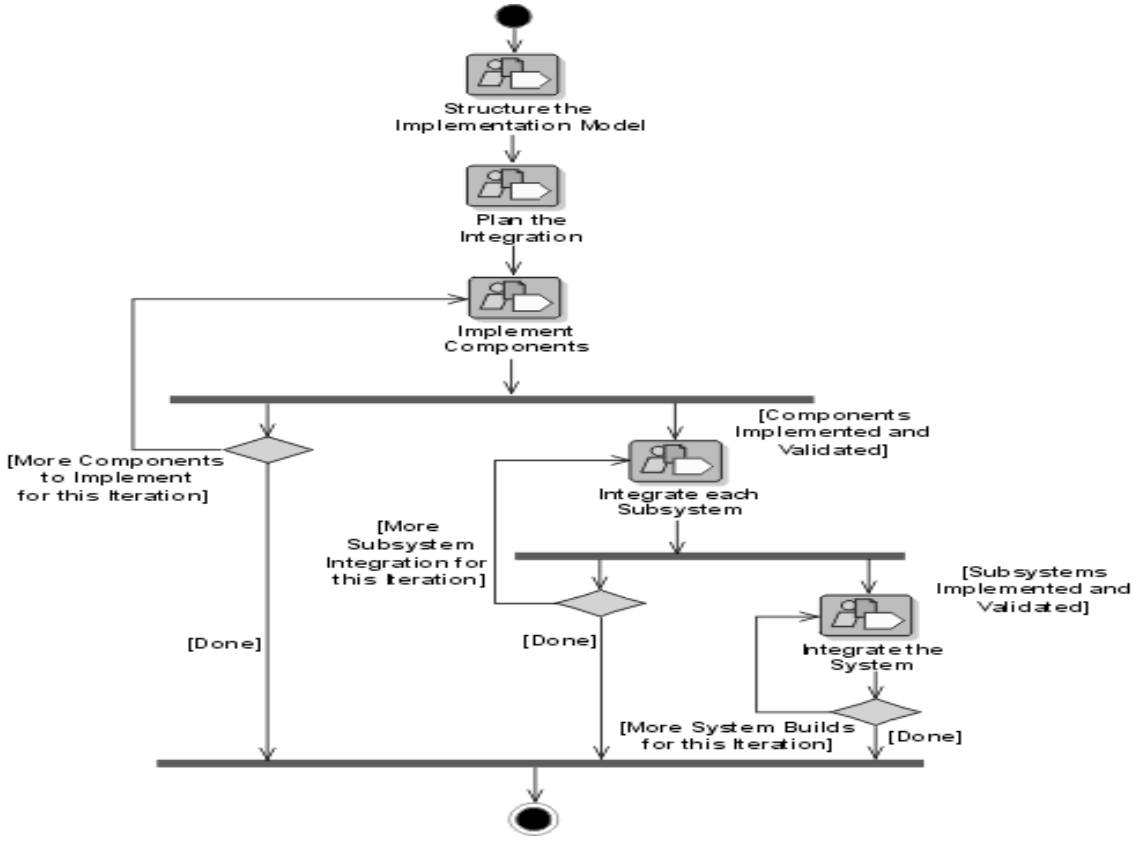


Anexo 3: Rol de Implementador

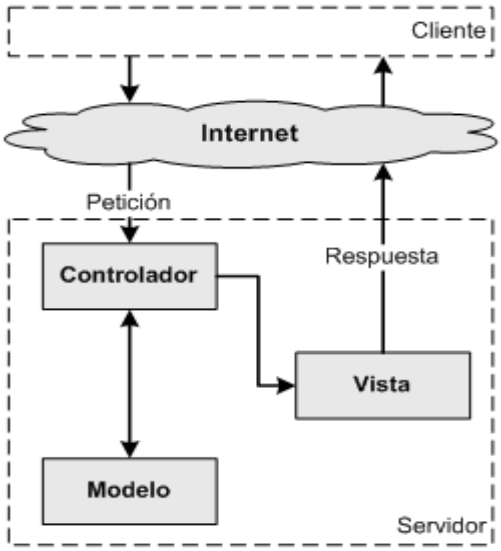
Anexo 4: Rol de Arquitecto de software

Anexo 5: Rol de Integrador del sistema

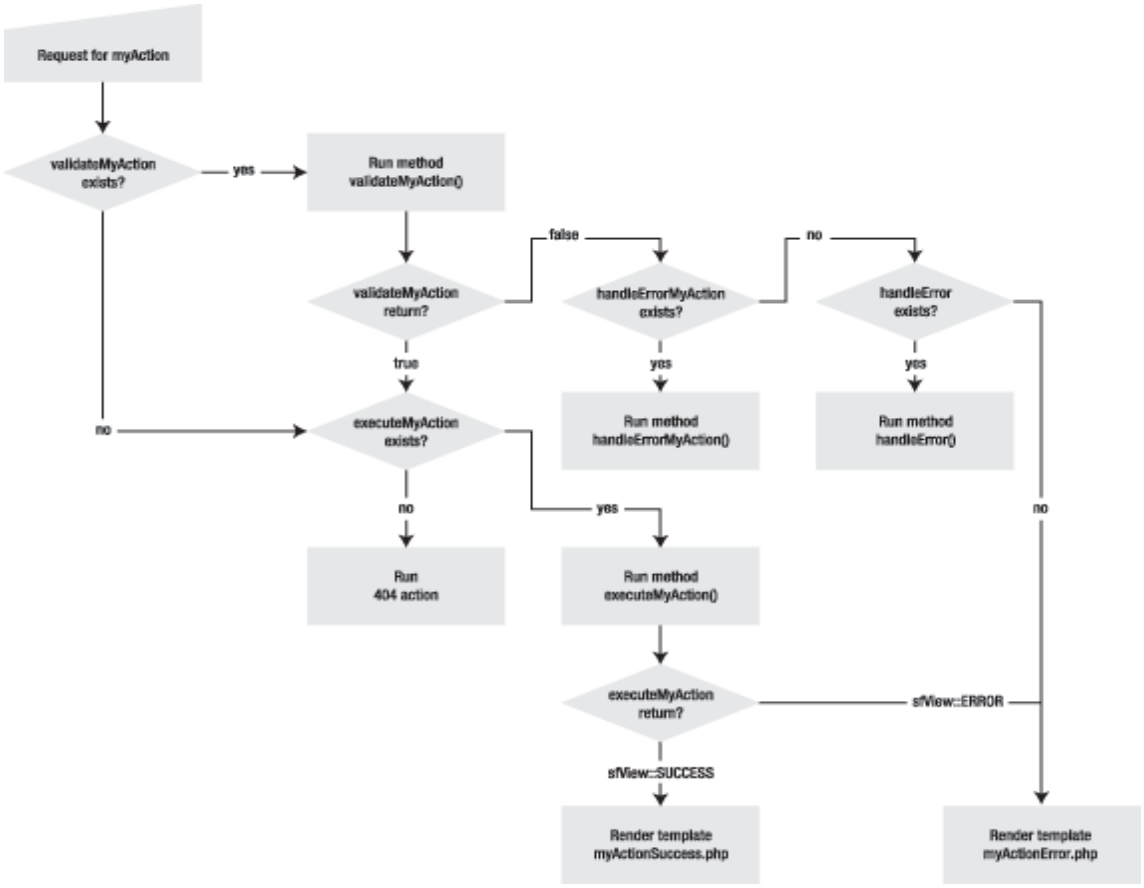
Anexo 6: Actividades del FT de Implementación



Anexo 7: Estructura del Patrón Modelo-Vista-Controlador (Básico)



Anexo 8: Proceso de validación en Symfony



GLOSARIO DE TÉRMINOS

Artefacto: Unidad tangible utilizada o producida por un proceso de desarrollo de software, como un documento externo o el producto de un trabajo. Un artefacto puede ser un modelo, una descripción o un software.

Aseguramiento de la Calidad: Conjunto de acciones planificadas y sistemáticas que son necesarias para proporcionar la confianza de que un producto o servicio satisface los requisitos de calidad preestablecidos.

Calidad: Grado en que un conjunto de características inherentes cumplen con los requisitos.

CU (Casos de Usos): Especificación de las secuencias de acciones, incluyendo secuencias variantes y una descripción de un conjunto de secuencias de acciones, incluyendo variaciones, que un sistema lleva a cabo y que conduce a un resultado observable de interés para un actor determinado.

Control de la calidad: Parte de la gestión de la calidad orientada al cumplimiento de los requisitos de la calidad.

CIGB: Centro de Ingeniería Genética y Biotecnología

DB: Data Base (Base de datos).

ELISA (Enzyme-Linked ImmunoSorbent Assay) Inmunoanálisis ligado a enzimas): Prueba rápida donde un anticuerpo o antígeno se une a una enzima como medio para detectar una compatibilidad entre el anticuerpo y el antígeno. Esta técnica se aplica en los ensayos realizados a los laboratorios para obtener un resultado factible de la calidad del producto.

Inmunoquímica: Laboratorio que realiza el control de la calidad a los productos obtenidos por vía recombinante que se producen en el CIGB, empleando las siguientes técnicas Inmunoquímicas: ELISA, Inmuno-Dot y Western-Blot.

Inmuno-Dot: Es una de las variantes del ELISA, también conocido como inmunopunto o dot-blot, el cual emplea una membrana de nitrocelulosa como soporte para el acoplamiento del antígeno, en lugar del soporte de polietileno, empleado en el ELISA convencional. Esta técnica se aplica en los ensayos realizados a los laboratorios para obtener un resultado factible de la calidad del producto.

Inmunización: Prueba para medir la presencia y cantidad de anticuerpos en la sangre.

Laboratorio: Local donde se realizan los ensayos a las muestras, posee todo el equipamiento necesario para desarrollar las pruebas a todas las muestras que son recepcionadas.

LIQ: Laboratorio de Inmunoquímica

MVC: (Model-View-Controller) Modelo-Vista-Controlador.

ORM: (object-relational-mapping) o “mapeo de objetos a bases de datos”.

RUP: Proceso Unificado de Desarrollo de Software.

SIC (Sistema de Información de la Calidad): Planilla usada para el registro de resultados, controles y análisis de determinada operación en el laboratorio.

URL: Uniform Resource Locator o localizador uniforme de recurso.

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.