Universidad de las Ciencias Informáticas

Facultad 6

Título: Sistema para la Gestión de la Información de los Laboratorios de la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología: Implementación del Módulo de "Liberación Analítica"

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autores: Carlos Lima Román
Javier Toledo Pineda

Tutores: Lic. Yoandry Pacheco Águila
Ing. Alfredo Rodríguez Ruiz

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

DECLARACIÓN DE AUTORÍA

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

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

_____	_____
Javier Toledo Pineda	Carlos Lima Román
Autor	Autor
_____	_____
Ing. Alfredo Rodríguez Ruiz	Lic. Yoandry Pacheco Águila
Tutor	Tutor

AGRADECIMIENTOS

A nuestros familiares que tanto amor y cariño nos han brindado y que tantas noches de desvelo han tenido junto a nosotros.

A nuestros tutores por su apoyo en todo momento, a pacheco por estar dispuesto en todo momento a aclarar cualquier duda.

A los profesores del LIMS por toda la ayuda brindada.

A nuestros amigos casi hermanos por el apoyo y cariño ofrecido. En especial a la gente del “LIMS” por estar juntos todo el tiempo, enfrentando cualquier reto, cada bateo de Symfony. A todos los que molestamos con cualquier duda; particularmente al Yuri, Saidel, Ángel.

DEDICATORIA

De Javier:

Dedico el presente trabajo:

A mis padres por el amor y apoyo brindado durante todos estos años y por ser lo más grande en el mundo.

A mi hermano por ser un ejemplo a seguir en todo momento.

A Billo, por ser mi segundo padre.

RESUMEN

En la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología (CIGB) se manipulan diariamente grandes volúmenes de información acerca de los procesos que allí se realizan para garantizar el control de la calidad de los productos que se elaboran en dicho centro. Toda esta información se gestiona manualmente, por lo que su manejo se torna un tanto engorroso. Dada la necesidad de administrar y controlar toda esta información de manera rápida y eficiente se está desarrollando un Sistema de Gestión de Información de los Laboratorios (LIMS, del inglés Laboratory Information Management Systems) para esta área, en conjunto con la Universidad de las Ciencias Informáticas (UCI). El presente trabajo describe la fase de implementación del Módulo de Liberación Analítica, uno de los grupos pertenecientes a la Dirección de Calidad del CIGB.

Palabras claves: Calidad, CIGB, Gestión de Información, LIMS

TABLA DE CONTENIDOS

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC) han experimentado un acelerado avance a nivel mundial, constituyendo uno de los factores de desarrollo más influyentes en los sectores político, económico y social de numerosos de países, además de constituir una importante diferencia entre una sociedad desarrollada y otra en vías de desarrollo. La expansión de las TIC en todos los ámbitos y estratos de la sociedad se han producido a gran velocidad, y es un proceso que continúa incrementándose a medida que surgen nuevos avances tecnológicos.

Debido al auge que ha tenido en el mundo la implantación y uso de las TIC, Cuba se ha planteado la necesidad de lograr su introducción de forma paulatina, con el objetivo de informatizar el país y comenzar el tránsito hacia una sociedad basada en el conocimiento. Esta labor se ha visto reflejada en importantes centros científicos del país, tal es el caso del Centro de Ingeniería Genética y Biotecnología (CIGB), entidad que cuenta con una trayectoria de más 20 años de investigaciones científicas, durante los cuales ha alcanzado numerosos e importantes logros que han tenido gran repercusión en los campos de la biomedicina y la bioindustria a nivel mundial. Todo esto ha sido posible gracias a la extraordinaria labor realizada por el personal científico que trabaja en dicho centro, permitiendo así, que Cuba goce de un alto prestigio internacional en el desarrollo y producción de productos biológicos.

Los numerosos fármacos producidos en el CIGB se encuentran en uso dentro del sistema de salud cubano y de diversas naciones de América Latina, distinguiéndolos su eficacia y seguridad en el tratamiento, para lograr esto en la estructura del CIGB se encuentra la Dirección de Calidad, compuesta por el Departamento de Aseguramiento de la Calidad y el Departamento de Control de la Calidad. (Ver anexo1)

“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 satisfagan los requisitos de calidad establecidos, es el encargado también de velar por el cumplimiento de las Buenas Prácticas de Producción (BPP), Buenas Prácticas de Laboratorio (BPL) y Buenas Prácticas Clínicas (BPC)”. [1] Este Departamento está compuesto por dos departamentos y cuatro grupos de trabajo:

- Departamento de Inspección y Auditoría.
- Departamento de Mejoramiento e Ingeniería de Calidad.

- Grupo de Administración y Costos.
- Grupo de Liberación.
- Grupo de Documentación.
- Grupo de Metrología.

El Departamento de Control de la Calidad realiza varias funciones como son las relacionadas con el muestreo, las especificaciones, los ensayos y la evaluación de la calidad de los productos que se generan en el centro. Para el desempeño de las mismas, cuenta con la ayuda de dos departamentos con sus grupos de trabajos específicos, y otros tres grupos de trabajo:

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

El **Grupo de Liberación Analítica** juega un papel muy importante en el control de la información que debe ser procesada para determinar la calidad de los productos analizados en los laboratorios. Es el primer componente de la cadena lógica de procesamiento y gestión de los datos referentes a los procesos de Control de la Calidad. En el mismo se manipulan y generan grandes volúmenes de información referentes a la recepción y distribución de las muestras a los laboratorios, así como la

recogida de los resultados analíticos emitidos por los mismos. Para el almacenamiento de dicha información se utilizan libros, planillas y expedientes en formato duro, y hojas de cálculo de Microsoft Excel, documentos Microsoft Word y archivos .pdf y .txt para conservar la información en formato digital.

Los documentos generados en este departamento deben ser revisados y supervisados por la dirección del grupo, por lo que la información consultada no se obtiene con la inmediatez requerida cuando se necesita de un tiempo de respuesta mínimo para la toma de decisiones, lo mismo sucede con la elaboración de reportes rutinarios u ocasionales que se ven afectados por los lentos métodos de búsqueda entre los numerosos documentos a consultar, además de que la cantidad de material de oficina empleado para el desempeño laboral de los trabajadores es considerable.

Por lo anteriormente visto se puede concluir que los procesos de gestión de la información en el Grupo de Liberación Analítica carecen de la rapidez y eficiencia necesarias, con el objetivo de eliminar dichas deficiencias es que se desea desarrollar un Sistema de Gestión de Información de los Laboratorios (LIMS), que no es más que un conjunto de herramientas basadas en sistemas informáticos que permiten la adquisición y gestión de toda la información generada en los laboratorios, trayendo consigo el aumento de la productividad y eficiencia de estos, razones que hacen que el empleo de los LIMS en la industria moderna resulte casi imprescindible.

Luego del estudio de los mecanismos de almacenamiento y procesamiento de la información empleados en el Grupo de Liberación Analítica, un equipo de analistas desarrolló el modelado del negocio, identificando los procesos que allí se llevan a cabo. Se analizó la arquitectura a utilizar, definiéndose las clases del diseño con sus atributos y relaciones, culminando así las fases de análisis y diseño, obteniéndose la base de datos relacional correspondiente al módulo.

Por lo expuesto anteriormente se identifica como **problema científico**: ¿Cómo obtener un producto funcional para el Módulo de Liberación Analítica del LIMS de Calidad del CIGB?

El problema planteado tiene como **objeto de estudio**: Los Sistemas de Gestión de la Información de los Laboratorios.

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

En la búsqueda de la solución al problema planteado se define como **objetivo general**: Implementar el Módulo de Liberación Analítica.

Para cumplir este objetivo se trazaron las siguientes **tareas**:

- Estudio de la documentación obtenida en las iteraciones realizadas anteriormente
- Estudio de la base de datos del Módulo de Liberación Analítica
- Familiarización con las herramientas y tecnologías definidas en la arquitectura del proyecto
- Implementación del Módulo de Liberación Analítica
- Realización de las pruebas unitarias
- Validación de la seguridad del módulo
- Realización de diagramas de componentes

El presente trabajo consta de Introducción, 2 Capítulos, Conclusiones, Bibliografía y Anexos.

En el **Capítulo 1: "Fundamentación Teórica"** se abordan distintos temas que sirven de soporte teórico a la solución del problema, se hace una breve descripción acerca de los Sistemas de Gestión de Información y como parte de ellos los LIMS existentes en el mundo, además se fundamenta la utilización de las herramientas empleadas en la solución del problema.

En el **Capítulo 2: "Implementación del Sistema"** se describe la implementación de los componentes, se muestran algunos diagramas representativos de los mismos y se analizan algunos fragmentos de código.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

Introducción.

En este capítulo se abordan distintos temas que sirven de soporte teórico a la solución del problema, se hace una breve descripción acerca de los Sistemas de Gestión de Información y como parte de estos, los LIMS existentes en el mundo, además se fundamenta la utilización de las herramientas empleadas en la solución del problema.

1.1 Sistemas de Gestión de la Información.

En la era de la información, de la explosión de sus tecnologías, se vive la etapa en que la humanidad ha alcanzado un desarrollo imprevisible y donde la información es un elemento fundamental para el desarrollo. Con el paso de los años, la gestión de la información ocupa un mayor espacio en la economía de los países, es por esta razón que surge la necesidad inmediata de implantar modelos para la gestión de la calidad total en las instituciones de información.

Los Sistemas de Gestión de la Información son un conjunto de elementos relacionados y ordenados según ciertas reglas que aportan al sistema objeto, es decir, a la organización a la que sirven, y que marcan sus directrices de funcionamiento así como la información necesaria para el cumplimiento de sus fines, para ello debe recoger, procesar y almacenar datos procedentes tanto de la organización como de fuentes externas, con el propósito de facilitar su recuperación, elaboración y presentación. Un sistema de gestión de información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de la información.

1.1.1 Tipos de Sistemas de Gestión de la Información.

Los Sistemas de Gestión de la Información cumplen tres objetivos básicos dentro de las organizaciones:

- Automatización de procesos operativos.
- Proporcionar información que sirva de apoyo al proceso de toma de decisiones.
- Lograr ventajas competitivas a través de su implantación y uso.

Los Sistemas de Información que logran la automatización de procesos operativos dentro de una organización, son llamados frecuentemente Sistemas Transaccionales, su función primordial consiste en procesar transacciones tales como pagos, cobros, pólizas, entradas, salidas, entre otras.

Los Sistemas de Información que apoyan el proceso de toma de decisiones son los Sistemas de Soporte a la Toma de Decisiones, Sistemas para la Toma de Decisión de Grupo, Sistemas Expertos de Soporte a la Toma de Decisiones y Sistema de Información para Ejecutivos. El tercer tipo de sistema, de acuerdo con su uso u objetivos que cumplen, es el de los Sistemas Estratégicos, los cuales se desarrollan en las organizaciones con el fin de lograr ventajas competitivas, a través del uso de las tecnologías de la información.

1.1.2 Sistemas de Gestión de la Información de los Laboratorios (LIMS).

Con el objetivo de incorporar al ambiente del laboratorio los beneficios y mejoras aportados por los Sistemas de Gestión de la Información, es que a finales del siglo 20 surgen los **Sistemas de Gestión de la Información de los Laboratorios**.

LIMS: Es un programa de gestión de laboratorios que permite recoger, almacenar, calcular y gestionar datos en una amplia variedad de formas. Los LIMS representan una importante herramienta para la gestión global de un laboratorio en un entorno de calidad, agilizando temas de registro de datos primarios, archivos, trazabilidad, entre otros, y minimizando los errores provocados por la transferencia de información. [2]

Los LIMS proporcionan un conjunto de herramientas basadas en Sistemas Informáticos que permiten la aplicación de técnicas de adquisición y gestión avanzada de la información producida en el laboratorio.
El uso de los LIMS ha facilitado grandemente la manipulación de los datos que se generan en los laboratorios por las ventajas que brinda su aplicación, algunas de estas son:

- Revisión y visualización de datos más completa, flexible y accesible.
- Generación más rápida y efectiva de informes.
- Centralización de la información en una única base de datos, lo que conlleva que la accesibilidad a los datos sea más segura, rápida, cómoda y sencilla.
- Mantenimiento de estados ligados a proyectos, muestras, pruebas y resultados, de manera automática por el sistema.
- Generación automática de recibos de recepción de muestras.
- Planificación del trabajo del laboratorio y generación automática de hojas de trabajo.
- Realización automática de cálculos y gráficos.

Por estas y otras ventajas numerosas empresas se han dedicado a fabricar LIMS, tal es el caso de:

- **LabWare LIMS.**

LabWare LIMS es un sistema de información de laboratorios (LIMS) con acceso equivalente cliente/servidor y web, que se integra sin costuras en cualquier entorno informático. Presenta una completa funcionalidad para el seguimiento de muestras, certificación de usuarios, gestión de instrumentos, auditorías, y planificación de muestras e informes, así como muchas otras funciones. [3]

Características principales:

- La aplicación corre sobre Windows™ y sobre un explorador web, tal como Microsoft IE, Safari, Opera, etc.
- La plataforma puede instalarse en Windows™, y cualquier versión de Unix o Linux.
- Como base de datos, se pueden usar todos los productos comerciales, desde ORACLE™ o SQL Server™ hasta DB2™.
- LabWare LIMS cumple las Buenas Prácticas de Laboratorio y los requisitos de la FDA respecto al almacenamiento y firma electrónica, de acuerdo a la regla 21 CFR Part 11.
- Sencillez en su uso.

- **Bika LIMS.**

Bika LIMS se estructura 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 es basado 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 y orientado a objetos. Está liberado bajo la licencia pública general (GPL).

- **Open LIMS.**

El objetivo de Open LIMS es desarrollar una organización y estructura independientes del software. Los módulos son generalmente escritos en PHP, para el cálculo de resultados cuenta con sistema de trabajo escrito en Java, requiere Java VM. Brinda la posibilidad de 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 escritas en XML. Permite al propietario del proyecto establecer toda una serie de permisos, divididos en diferentes tipos. Utiliza la codificación de caracteres UTF8 . Publicado bajo la licencia GPL, todavía se encuentra en desarrollo.

¿Por qué desarrollar un LIMS?

En investigaciones anteriores se decidió que era necesaria la realización de un LIMS con características adaptables al CIGB, pues necesita ser flexible a los cambios que surgen con regularidad en el centro, además de que la mayoría de los proveedores de LIMS en el mundo son de Estados Unidos, constituyendo esto un problema a la hora de la compra de licencias y demás. Resulta menos costoso desarrollarlo, debido a que los procesos de instalación, mantenimiento y actualización cuestan miles de dólares y el sistema que necesita la Dirección de Calidad del CIGB requiere mantenimiento de por vida por ser tan grande y variable. Una vez desarrollado en Cuba puede ser extensible a otros centros del Polo Científico y en general a entidades que lo requieran para mejorar y agilizar los procesos relacionados con el control de la calidad de sus producciones.

1. 2 Herramientas y Metodologías utilizadas.

Para desarrollar una aplicación informática siempre se deben definir los tipos de tecnologías a utilizar así como las herramientas y metodologías que serán de mayor utilidad para su implementación. Luego del estudio de las características y necesidades requeridas para el desarrollo de un LIMS, quedaron definidas las herramientas, así como la metodología a utilizar.

Una metodología de desarrollo de software es un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software, la cual puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y finales.

Entre las metodologías de desarrollo más conocidas a nivel mundial se encuentran: Programación Extrema (Extreme Programming, XP) clasificada como una metodología ágil, Desarrollo Guiado por la Funcionalidad (Feature Driven Development, FDD) clasificada como un proceso de término medio, ya que puede ser ágil o robusta y Proceso Unificado de Desarrollo (Rational Unified Process, RUP) que se clasifica como una metodología robusta, siendo esta última la definida para el desarrollo del sistema. A continuación se menciona las herramientas y metodologías utilizadas en el desarrollo de la aplicación.

1.2.1 Metodología de desarrollo de software.

Proceso Unificado de Desarrollo (RUP).

La metodología RUP, plantea quién hace qué, cuándo y cómo, tiene como objetivo asegurar la producción de software de calidad dentro de plazos y presupuestos predecibles. Se actualiza constantemente para tener en cuenta las mejores prácticas, y utiliza UML como lenguaje de modelado. [4]

El ciclo de vida de RUP se caracteriza por ser:

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo.

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo, por lo que describe los elementos del modelo más importantes para su construcción. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto.

RUP agrupa sus actividades en 9 flujos de trabajo, 6 de ingeniería y 3 de soporte que se desarrollan durante 4 fases, donde define roles por cada miembro del equipo de desarrollo. Las fases que componen son las siguientes:

- **Inicio:** el objetivo es determinar la visión del proyecto.
- **Elaboración:** el objetivo es determinar la arquitectura óptima.
- **Construcción:** el objetivo es obtener la capacidad operacional inicial.
- **Transición:** el objetivo es obtener el producto del proyecto.

Cada fase concluye con un hito bien definido:

- **Inicio:** visión de los objetivos.
- **Elaboración:** prototipo de la arquitectura.
- **Construcción:** capacidad operacional inicial.
- **Transición:** liberación del producto.

RUP define los roles a jugar por cada miembro del equipo de desarrollo en cada una de las fases por las que transita el desarrollo de un sistema y facilita la comunicación entre los diferentes miembros del equipo de desarrollo.

Flujo de trabajo a realizar: Implementación.

En la implementación se comienza con el resultado del diseño, y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. [5]

El propósito principal de la implementación es desarrollar la arquitectura y el sistema como un todo, de forma más específica los propósitos de la implementación son:

- Planificar las integraciones del sistema necesarias en cada iteración. Siguiendo para ello un enfoque incremental, lo que da lugar a un sistema que se implementa en una sucesión de pasos pequeños y manejables.
- Distribuir el sistema, asignando componentes ejecutables a nodos en el diagrama de despliegue. Esto se basa fundamentalmente en las clases activas encontradas durante el diseño.
- Implementar las clases y subsistemas encontrados durante el diseño. En particular, las clases se implementan como componentes de fichero que contienen código fuente.
- Probar los componentes individualmente, y luego integrarlos, compilándolos y enlazándolos en uno o más ejecutables antes de ser enviados para ser integrados y llevar a cabo las comprobaciones de sistema.

Los trabajadores que participan en este flujo de trabajo son:

- Arquitecto de Software.
- Implementador.
- Integrador.
- Revisor Técnico.

Las principales actividades que se realizan en este flujo de trabajo son:

- Estructurar el Modelo de Implementación.
- Planificar la Integración.
- Implementar Componentes.

Roles y artefactos.

Rol: Un rol es una definición abstracta de un conjunto de actividades realizadas y de artefactos obtenidos. Los roles son realizados típicamente por un individuo, o un conjunto de individuos, trabajando juntos en equipo. Un miembro del equipo de proyecto cumple normalmente muchos roles, estos describen cómo los individuos se comportan en el negocio y qué responsabilidades tienen.

Artefacto: Producto tangible del proyecto que es producido, modificado y usado por las actividades. Los artefactos pueden ser modelos, elementos dentro del modelo, código fuente o ejecutables. Los trabajadores en el flujo de trabajo Implementación son:

- Arquitecto de software.
- Programador.
- Integrador.

Obteniéndose los siguientes artefactos:

- Modelo de Implementación.
- Subsistema de Implementación.
- Plan de Integración.
- Diagrama de Componentes.

El arquitecto de software es el encargado de estructurar el modelo de implementación, actualizar el Documento de la Arquitectura del Software, añadiendo la Vista de Implementación. Además resulta actualizado el Modelo de Implementación, ya sea porque se crea por primera vez o se añaden nuevos elementos de implementación a los ya existentes provenientes de una iteración anterior. El arquitecto es además responsable también de la arquitectura del modelo de implementación.
El integrador es quien tiene la labor de planificar la integración al sistema, la cual tiene como principal resultado el Plan de Integración del Sistema. Este artefacto provee un orden para efectuar la integración al sistema.

El programador es el responsable de realizar la implementación de los componentes. El propósito de esta actividad es completar una parte de la implementación que podría ser entregada para la integración. En esta actividad el programador escribe código fuente, reutiliza código, compila, realiza pruebas unitarias e implementa los elementos del modelo de diseño. Si encuentra defectos en el diseño, regresa a la fase de análisis y diseño y los corrige.

Modelo de Implementación.

Describe cómo los elementos del modelo de diseño, se implementan en términos de componentes, ficheros de código fuente, ejecutables etc. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje de programación utilizado, y cómo dependen los componentes unos de otros.

Subsistema de implementación.

Es una colección de componentes y otros subsistemas de implementación, usados para estructurar el modelo de implementación y dividirlos en pequeñas partes que pueden ser integradas y probadas por separado. Los subsistemas de implementación incluyen dependencias y otras informaciones. También podrían incluir modelos claves del subsistema (diagrama de componentes y modelo de despliegue).

Componente.

Parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces, proporciona la realización de los mismos. Un componente generalmente contiene clases, y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios, etc.)

Diagrama de Componentes.

Se representa como un grafo de componentes unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfaces que estos soportan. Es un diagrama que muestra un conjunto de elementos del modelo tales como componentes, subsistemas de implementación y sus relaciones.

Se utilizan para modelar la vista estática de un sistema, muestra la organización y las dependencias lógicas entre un conjunto de componentes, sean estos componentes de código fuente, librerías, binarios o ejecutables. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema.

Un paquete en un diagrama de componentes representa una división física del sistema. Los paquetes se organizan en una jerarquía de capas donde cada capa tiene una interfaz bien definida. Un ejemplo típico de una jerarquía en capas de este tipo es: interfaz de usuario, paquetes específicos de la aplicación, paquetes reusables, mecanismos claves y paquetes hardware y del sistema operativo.

Lenguaje de Modelado.

UML

El lenguaje de modelado empleado es el Lenguaje Unificado de Modelado (UML). El mismo se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de software. Dicho lenguaje es una notación unificada con la que se permite lograr un entendimiento que propicie el intercambio entre los usuarios y los desarrolladores. Se ha convertido en un estándar de la industria del software. El UML estándar está compuesto por tres partes: elementos de construcción (tales como clases, objetos y mensajes), relaciones entre los bloques tales como asociación, generalización y diagramas.
Los principales beneficios de UML son:

- Mejores tiempos totales de desarrollo.
- Permite la modelación de sistemas utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos.

1.2.2 Herramienta CASE.

Las herramientas CASE (Computer Aided Software Engineering, en español Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de los mismos en términos de tiempo y dinero. Dichas herramientas ayudan en diversos aspectos del ciclo de vida de desarrollo del software, tales como el proceso de diseño del proyecto, cálculo de costos, generación automática de una parte del código, compilación automática, documentación o detección de errores, entre otras.

Entre las herramientas CASE más empleadas a nivel mundial se encuentran Umbrello, MagicDraw, Rational Rose y Visual Paradigm. Siendo esta última la empleada en el desarrollo del proyecto.

Visual Paradigm for UML 6.1 Enterprise Edition.
Es una potente herramienta CASE para visualizar y diseñar elementos de software, para ello utiliza UML como lenguaje de modelado, proporciona a los desarrolladores una plataforma que les permite diseñar un producto con calidad de manera rápida. Visual Paradigm soporta el ciclo de vida completo del desarrollo del software, tiene la capacidad de ser multiplataforma, facilita la interoperabilidad con

otras herramientas CASE y se integra con los siguientes softwares: Eclipse/IBM, WebSphere, NetBeans IDE, Oracle Jdeveloper, BEA Weblogic, así como con varios lenguajes de programación, por ejemplo: PHP, Java y C#.

1.2.3 Lenguaje de Programación.

PHP 5.2.5

Es un lenguaje de programación del lado del servidor, gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación, además es usada normalmente para la creación de páginas web dinámicas.

PHP 5 mejorara los mecanismos de POO para solucionar las carencias de las anteriores versiones. Posee numerosas ventajas tales como:

- Soporte sólido y real para Programación Orientada a Objetos con PHP Data Objects.
- Mejoras de rendimiento.
- Mejor soporte a XML (XPath, DOM).
- Soporte integrado para SOAP.
- Excepciones de errores.

1.2.4 Framework de desarrollo.

Un framework en el desarrollo de software, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente puede incluir soporte de programas, bibliotecas y un lenguaje interpretado para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio. Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Proporciona estructura al código fuente, forzando al desarrollador a crear código legible y fácil de mantener. Facilita la programación de aplicaciones, pues encapsula operaciones complejas en instrucciones sencillas.

Symfony 1.0.18

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de

desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.**¡Error! No se encuentra el origen de la referencia.**

Symfony posee las siguientes características:

- Fácil de instalar y configurar en la mayoría de las plataformas.
- Independiente del sistema gestor de base de datos.
- Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Sigue las mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

1.2.5 Entorno de Desarrollo Eclipse 3.3.1.1

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma, creado inicialmente para JAVA. Aunque Eclipse es escrito en Java y su principal uso es como IDE para Java, este es un lenguaje neutral. El soporte para desarrollo en Java es proveído por un componente enchufado o plug-in, pero además están disponibles plugins para otros lenguajes, como C/C++, Cobol, C#, PHP. En principio permite ejecutar un programa sobre cualquier plataforma. Es una extensible plataforma de código abierto para desarrollar herramientas. Compila en tiempo real y tiene editores de sintaxis resaltada para HTML, CSS, XML entre otros.

1.2.6 Sistema Gestor de Base de Datos.

Los sistemas de gestión de base de datos (SGBD) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan, reúnen una serie de funciones que permiten definir registros, campos, relaciones, insertar, suprimir, modificar y consultar los datos en una base de datos.

PostgreSQL 8.2.

El sistema gestor de base de datos PostgreSQL 8.2 es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) basado en el proyecto POSTGRES, que presenta actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de

código abierto. Algunas de las características que identifican a PostgreSQL son: que a pesar de que no es puramente orientado a objetos, incluye la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Presenta una mejor escalabilidad en sistemas multiprocesador y mejor optimización de consultas sobre datos particionados.

1.2.7 Servidor Web.

Un servidor web es un programa que implementa el protocolo HTTP (HyperText Transfer Protocol), diseñado para transferir hipertextos, páginas web o páginas HTML (HyperText Markup Language), textos complejos con enlaces, figuras, formularios, botones, entre otros elementos. Es un programa que se ejecuta continuamente en un ordenador, manteniéndose a la espera de peticiones por parte de un cliente (navegador web) y que responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error.

Apache 2.2.6

Apache es un servidor http de código abierto, permite la creación de sitios web dinámicos mediante el uso de Server Side Includes (SSI), de lenguajes de scripting como PHP, JavaScript, Python, Java y páginas jsp. Es multiplataforma e ideal para instalar máquinas GNU/Linux, que aseguran un sistema operativo con comunicaciones excelentes.

1.3 Patrones.

Un patrón es un esquema de solución que se aplica a un tipo de problema, su aplicación no es mecánica, sino que requiere de adaptación y matices. Es un par problema/solución, con nombre, y que se puede aplicar en nuevos contextos, posee consejos sobre cómo aplicarlo en nuevas situaciones, o sea, un patrón es una descripción de un problema bien conocido que suele incluir: descripción, escenario de uso, solución concreta, consecuencias de utilizar el patrón, ejemplos de implementación y lista de patrones relacionados.

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 los componentes, con sus responsabilidades, relaciones, y las formas en que colaboran.

Expresan un paradigma fundamental para estructurar u organizar un sistema software. Proporcionan un conjunto de subsistemas o módulos predefinidos, con reglas y guías para organizar las relaciones entre ellos.

Modelo-Vista-Controlador (MVC).

El patrón conocido como Modelo-Vista-Controlador (MVC) separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:

Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Maneja la visualización de la información.

Controlador: Controla el flujo entre la vista y el modelo (los datos).

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo, independientemente de la representación visual.

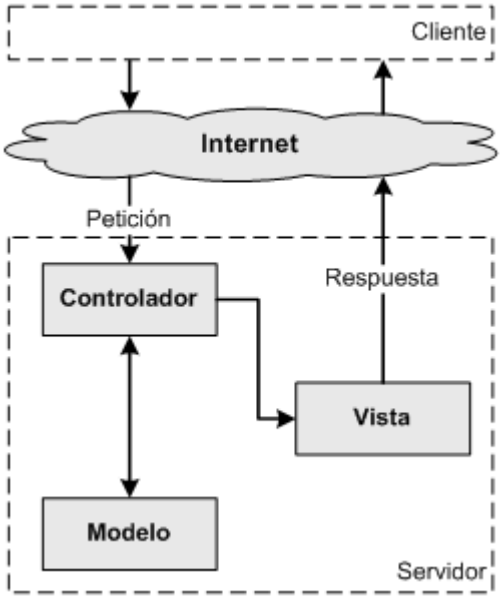


Fig. 1 Patrón de arquitectura Modelo-Vista-Controlador.

Entre las ventajas del estilo Modelo-Vista-Controlador están las siguientes:

Soporte de múltiples vistas*:* Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación Web pueden utilizar el mismo modelo de objetos mostrado de maneras diferentes.

Adaptación al cambio*:* Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o

requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

Patrones de Diseño.

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 [5]

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de un software. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares, identifican clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Los patrones de diseño ofrecen esquemas para definir estructuras de diseño con las que construir sistemas software y permiten formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño.

Los patrones de diseño se dividen en tres grupos principales:

- Patrones de Creación: Inicialización y configuración de objetos.
    - Patrón de Fábrica Abstracta.
    - Patrón Constructor.
    - Patrón del Método de Fabricación.
    - Patrón Prototipo.
    - Patrón de Instancia Única.

- Patrones Estructurales: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
    - Patrón Adaptador.
    - Patrón Puente.
    - Patrón Compuesto.
    - Patrón Decorador.
    - Patrón de Fachada.
    - Patrón de Peso Mosca.
    - Patrón Apoderado.
- Patrones de Comportamiento: Más que describir objetos o clases, describen la comunicación entre ellos.

- 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.

Conclusiones del capítulo

Luego de haber analizado la situación actual de los LIMS a nivel mundial y lo novedoso de su aplicación en las diferentes industrias, así como las ventajas de su uso, se evidencia la importancia de desarrollar un LIMS para el CIGB. Teniendo en cuenta las restricciones que su elaboración requiere se detallan las diferentes herramientas usadas en su desarrollo, se explican las características de RUP como metodología de desarrollo empleada, Visual Paradigm como herramienta case, UML como lenguaje de modelado, PHP como lenguaje de programación y Symfony como framework de desarrollo.

CAPÍTULO 2: IMPLEMENTACIÓN DEL SISTEMA

Introducción

En este capítulo veremos un listado de Requerimientos Funcionales y No Funcionales, la Arquitectura del Sistema (Vista Lógica), la Vista de Despliegue (Diagrama de Componentes) y los Estándares de Codificación utilizados. Además se ve el empleo de los componentes reutilizables con un ejemplo de código fuente. También se realiza un análisis de la seguridad del sistema implementado.

2.1 Requerimientos Funcionales: Liberación Analítica.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Como resultado de la realización de flujo de trabajo de Requerimientos definidos por RUP se identificaron los siguientes requisitos funcionales del sistema a implementar:

CU Gestionar muestras

RF1.1 Registrar recepción de nueva muestra.
RF1.2 Modificar datos de muestra.
RF1.3 Registrar traza.
RF1.4 Buscar muestra.
RF1.5 Visualizar datos de muestra.
RF1.6 Registrar expediente analítico.
RF1.7 Crear nuevo libro de centro.
RF1.8 Crear nuevo libro producto.

CU Gestionar libro de registros de cada producto

RF2.1 Modificar datos del libro.
RF2.2 Registrar traza.
RF2.3 Visualizar libro.
RF2.4 Imprimir libro.
RF2.5 Generar reportes.
RF2.6 Imprimir reporte.

CU Gestionar libro de registros por centros.

RF3.1 Modificar datos del libro.
RF3.2 Registrar traza.
RF3.3 Buscar libro.
RF3.4 Visualizar libro.

RF3.5 Imprimir libro.

RF3.6 Generar reportes.

RF3.7 Imprimir reporte.

CU Gestionar datos de muestras testigo.

RF4.1 Registrar nueva muestra testigo.

RF4.2 Modificar datos de muestra testigo.

RF4.3 Registrar traza.

RF4.4 Buscar muestra testigo.

RF4.5 Visualizar muestra testigo.

RF4.6 Imprimir

CU Gestionar libro de registros de muestras testigos por producto.

RF5.1 Buscar libro de muestras testigos por producto.

RF5.2 Visualizar libro de muestras testigos por producto.

RF5.3 Modificar los datos del libro de muestras testigos por producto.

RF5.4 Registrar traza.

RF5.5 Imprimir libro de muestras testigos por producto.

RF5.6 Generar reportes.

RF5.7 Imprimir reporte.

CU Gestionar solicitud de muestras testigo.

RF6.1 Registrar solicitud en planilla SIC-0935.

RF6.2 Buscar planilla SIC-0935.

RF6.3 Visualizar planilla SIC-0935.

RF6.4 Modificar los datos de la planilla SIC-0935.

RF6.5 Registrar traza.

RF6.6 Imprimir modelo en blanco de planilla SIC-0935.

RF6.7 Imprimir planilla SIC-0825 existente.

RF6.8 Generar reportes.

RF6.9 Imprimir reporte.

CU Gestionar control de salida de muestras testigo.

RF7.1 Registrar control de salida de muestras testigo en planilla SIC-0825.

RF7.2 Buscar planilla SIC-0825.

RF7.3 Visualizar planilla SIC-0825.

RF7.4 Modificar los datos de la planilla SIC-0825.

RF7.5 Registrar traza.

RF7.6 Imprimir modelo en blanco de planilla SIC-0825.

RF7.7 Imprimir planilla SIC-0825 existente.

RF7.8 Generar reportes.

RF7.9 Imprimir reporte.

CU Gestionar revisión del estado de muestras testigos.

RF8.1 Registrar revisión de estado de muestras testigos en planilla SIC-0824.

RF8.2 Buscar planilla SIC-0824.

RF8.3 Visualizar planilla SIC-0824.

RF8.4 Modificar los datos de la planilla SIC-0824.

RF8.5 Registrar traza.

RF8.6 Imprimir modelo en blanco de planilla SIC-0824.

RF8.7 Imprimir planilla SIC-0824 existente.

RF8.8 Generar reportes.

RF8.9 Imprimir reporte

CU Gestionar solicitud de destrucción de productos.

RF9.1 Registrar solicitud de destrucción de productos en planilla SIC-0830.

RF9.2 Buscar planilla SIC-0830.

RF9.3 Visualizar planilla SIC-0830.

RF9.4 Modificar los datos de la planilla SIC-0830.

RF9.5 Registrar traza.

RF9.6 Imprimir planilla SIC-0830.

RF9.7 Generar reportes.

RF9.8 Imprimir reporte.

CU Gestionar Análisis de características organolépticas.

RF10.1 Registrar análisis en planilla SIC-0123.

RF10.2 Buscar planilla SIC-0123.

RF10.3 Visualizar planilla SIC-0123.

RF10.4 Modificar datos de planilla SIC-0123.

RF10.5 Registrar traza.

RF10.6 Imprimir planilla SIC-0123 en blanco.

RF10.7 Imprimir planilla SIC-0123 existente.

CU Gestionar resultados de las pruebas.

RF11.1 Buscar resultados de pruebas.

RF11.2 Visualizar resultados de pruebas.

RF11.3 Imprimir planilla.

CU Gestionar análisis de resultados fuera de especificación.

RF12.1 Registrar análisis en planilla SIC-0837.

RF12.2 Buscar planilla SIC-0837.

RF12.3 Visualizar planilla SIC-0837.

RF12.4 Modificar los datos de la planilla SIC-0837.

RF12.5 Registrar traza.

RF12.6 Imprimir planilla SIC-0837 en blanco.

RF12.7 Imprimir planilla SIC-0837 existente.

RF12.8 Generar reportes.

RF12.9 Imprimir reporte.

CU Gestionar cálculo de la actividad específica.

RF13.1 Crear nueva planilla SIC-0709.

RF13.2 Buscar planilla SIC-0709.

RF13.3 Visualizar planilla SIC-0709.

RF13.4 Modificar datos de planilla SIC-0709.

RF13.5 Registrar traza.

RF13.6 Imprimir planilla SIC-0709 en blanco.

RF13.7 Imprimir planilla SIC-0709 existente.

CU Gestionar cálculo relación PRP: TT.

RF14.1 Crear nueva planilla SIC-0795.

RF14.2 Buscar planilla SIC-0795.

RF14.3 Visualizar planilla SIC-0795.

RF14.4 Modificar datos de planilla SIC-0795.

RF14.5 Registrar traza.

RF14.6 Imprimir planilla SIC-0795 en blanco.

RF14.7 Imprimir planilla SIC-0795 existente.

CU Gestionar cálculo de grupos funcionales residuales.

RF15.1 Crear nueva planilla SIC-0794.

RF15.2 Buscar planilla SIC-0794.

RF15.3 Visualizar planilla SIC-0794.

RF15.4 Modificar datos de planilla SIC-0794.

RF15.5 Registrar traza.

RF15.6 Imprimir planilla SIC-0794 en blanco.

RF15.7 Imprimir planilla SIC-0794 existente.

CU Gestionar cálculo de la actividad biológica por bulbo.

RF16.1 Crear nueva planilla SIC-0938.

RF16.2 Buscar planilla SIC-0938.

RF16.3 Visualizar planilla SIC-0938.

RF16.4 Modificar datos de planilla SIC-0938.

RF16.5 Registrar traza.

RF16.6 Imprimir planilla SIC-0938 en blanco.

RF16.7 Imprimir planilla SIC-0938 existente.

CU Gestionar Notificación de resultados no satisfactorios.

RF17.1 Registrar Notificación de Resultados no satisfactorios en planilla SIC-0931.

RF17.2 Buscar planilla SIC-0931.

RF17.3 Visualizar planilla SIC-0931.

RF17.4 Modificar los datos de la planilla SIC-0931.

RF17.5 Registrar traza.

RF17.6 Imprimir planilla SIC-0931 en blanco.

RF17.7 Imprimir planilla SIC-0931 existente.

RF17.8 Generar reportes.

RF17.9 Imprimir reporte.

CU Gestionar informe de análisis.

RF18.1 Registrar informe de análisis en planilla SIC-0092.

RF18.2 Buscar planilla SIC-0092.

RF18.3 Visualizar planilla SIC-0092.

RF18.4 Modificar los datos de la planilla SIC-0092.

RF18.5 Registrar traza.

RF18.6 Imprimir planilla SIC-0092 en blanco.

RF18.7 Imprimir planilla SIC-0092 existente.

RF18.8 Generar reportes.

RF18.9 Imprimir reporte.

CU Gestionar certificado de análisis del ingrediente farmacéutico.

RF19.1 Registrar certificado en planilla SIC-0300.

RF19.2 Buscar planilla SIC-0300.

RF19.3 Visualizar planilla SIC-0300.

RF19.4 Modificar los datos de la planilla SIC-0300.

RF19.5 Registrar traza.

RF19.6 Imprimir planilla SIC-0300 en blanco.

RF19.7 Imprimir planilla SIC-0300 existente.

RF19.8 Generar reportes.

RF19.9 Imprimir reporte.

CU Gestionar certificado de análisis del producto final.

RF20.1 Registrar certificado en planilla SIC-0301.

RF20.2 Buscar planilla SIC-0301.

RF20.3 Visualizar planilla SIC-0301.

RF20.4 Modificar los datos de la planilla SIC-0301.

RF20.5 Registrar traza.

RF20.6 Imprimir planilla SIC-0301 en blanco.

RF20.7 Imprimir planilla SIC-0301 existente.

RF20.8 Generar reportes.

RF20.9 Imprimir reporte.

CU Gestionar Informe analítico semanal de resultados.

RF21.1 Generar informe.

RF21.2 Imprimir informe.

CU Generar reportes generales (a inicio del sistema).

RF22.1 Visualizar nuevos resultados registrados.

RF22.2 Visualizar resultados de ensayos pendientes.

RF22.3 Visualizar informe de análisis que ya pueden ser generados.

RF22.4 Visualizar resultados no conformes.

RF22.5 Avisar la finalización del almacenamiento de muestras testigo.

CU Gestionar informe de costos mensuales.

RF23.1 Generar informe.

RF23.2 Imprimir informe.

CU Gestionar resumen semestral de productos.

RF24.1 Generar resumen semestral por producto.

RF24.2 Imprimir resumen.

CU Gestionar informe de relación entre lotes por producto.

RF25.1 Generar informe de relación de lotes por productos.

RF25.2 Imprimir informes.

CU Gestionar lote.

RF26.1 Visualizar lote.

RF26.2 Buscar lote.

RF26.3 Modificar lote.

RF26.4 Registrar traza.

CU Gestionar expediente analítico.

RF27.1 Buscar expediente analítico.

RF27.2 Visualizar expediente analítico.

RF27.3 Modificar expediente analítico.

RF27.4 Registrar traza.

CU Gestionar documento que avala la liberación para la comercialización.

RF28.1 Registrar documento en planilla SIC-0001.

RF28.2 Buscar planilla SIC-0001.

RF28.3 Visualizar planilla SIC-0001.

RF28.4 Modificar los datos de la planilla SIC-0001.

RF28.5 Registrar traza.

RF28.6 Imprimir planilla SIC-0001 en blanco.

RF28.7 Imprimir planilla SIC-0001 existente.

RF28.8 Generar reportes.

RF28.9 Imprimir reporte.

2.2 Requerimientos no Funcionales: Liberación Analítica

Los requerimientos no funcionales son propiedades o cualidades que el sistema necesita. Estas propiedades se ven como las características que hacen al producto atractivo, usable, rápido o confiable, los requerimientos no funcionales que se identificaron son:

- **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.

- **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 los Laboratorios/Grupos una vez semanalmente en el primer mes de instalada la aplicación, y luego visitará el centro una vez al mes en los restantes cuatro meses después de instalada con el objetivo de dar mantenimiento a la misma. Después de instalada la aplicación, durante la primera semana se realizarán 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.

- **Políticos-culturales**

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

- **Legales**

  El sistema será registrado en el Centro Nacional de Derecho de Autor a través de la Dirección de Servicios Legales de la UCI previo acuerdo con el CIGB. El sistema se dará a conocer a todos los trabajadores de la Dirección de Calidad como la herramienta para gestionar la

información de cada uno de los grupos y laboratorios. Se estará usando para el desarrollo de la aplicación herramientas de software libre con licencia GNU/GPL.

- **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**

  Para el servidor de la aplicación el sistema operativo recomendado es Windows XP o Linux. Se debe instalar un servidor Web Apache 1.3 o superior. 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 y el Sistema Operativo debe ser Windows XP o cualquier distribución de Linux. El servido de Base de datos debe tener instalado PostgreSQL 8.2, el Sistema Operativo debe ser Linux o Windows XP.

- **Hardware**.

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

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

  La lógica de presentación constituirá una capa independiente de la lógica de negocio, centrando su función en la interfaz de usuario y validaciones de los datos de entrada. Se usará el lenguaje de programación PHP 5 y el gestor de bases de datos PostgreSQL 8.2, así como Symfony como framework de desarrollo.

2.3 Arquitectura del sistema.

La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios para comprenderlo, desarrollarlo y producirlo. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas UML.

2.3.1 Patrón de Arquitectura.

Symfony está basado en el patrón de arquitectura Modelo-Vista-Controlador (MVC), por lo tanto al usar Symfony como framework de desarrollo del sistema el mismo estará implementado utilizando dicho patrón arquitectónico.

Implementación del MVC que realiza Symfony (ver Anexo 2)

- La capa del Modelo.
  - Abstracción de la base de datos.
  - Acceso a los datos.
- La capa de la Vista.
  - Vista.
  - Plantilla.
  - Layout.
- La capa del Controlador.
  - Controlador frontal.
  - Acción.

Como puede observarse la capa del modelo se subdivide a su vez en dos capas más, la de acceso a datos llamada ORM (object-relational mapping) o mapeo de objetos a bases de datos y la de abstracción de la base de datos. Esta división se realiza debido a que las bases de datos son relacionales, mientras que PHP 5 y Symfony son orientados a objetos, por lo tanto para acceder de forma efectiva a la base de datos desde un contexto orientado a objetos es necesaria una interfaz que traduzca la lógica de los objetos a la relacional, de esta manera las funciones que acceden a los datos no utilizan sentencias ni consultas que dependan de una base de datos en específico, sino que utilizan otras funciones para realizar las consultas, el encargado de realizar esta tarea es el ORM. Por otro lado la capa de abstracción de la base de datos obliga a utilizar una sintaxis específica para realizar las consultas a la base de datos, a cambio esta capa se encarga de optimizar y adaptar el lenguaje SQL a la base de datos que se está utilizando, haciendo posible cambiar la aplicación a otro sistema gestor de bases de datos, con solo cambiar una linea en un archivo de configuración. En el desarrollo de la aplicación se utilizó Propel como ORM y Creole como capa de abstracción de la base de datos, dos componentes completamente integrados en Symfony y que se pueden considerar como una parte más del framework.

La capa de la Vista es la encargada de producir las páginas que se muestran al usuario como resultado de ejecutar determinada acción. En la misma se encuentran las plantillas que presentan los datos de la acción que se está ejecutando en ese momento y el layout que contiene el código HTML común a todas las páginas.
La capa del Controlador es quien contiene el código que liga la lógica de negocio con la presentación, entre los componentes más importantes que posee se encuentran el controlador frontal, único punto de entrada a la aplicación y encargado de cargar la configuración y determinar la acción a ejecutarse, y las acciones que son las que contienen toda la lógica de la aplicación, verifican la integridad de la petición y preparan los datos requeridos por la presentación.

2.3.2 Patrones de Diseño

En Symfony la programación se puede simplificar si se utilizan patrones de diseño. De esta forma, las capas del modelo, la vista y el controlador se pueden subdividir en más capas. El patrón MVC puede contener conjunto de patrones de diseño asociados implícitamente:
En la implementación del sistema se utilizaron algunos patrones de diseño ejemplos de ello son:

Patrón GRASP.

Creador

Las clases Action son las la que contienen las funciones definidas para la aplicación y son las encargadas de ejecutarlas. En dichas acciones se crean objetos de las clases que representan las entidades del modelo, un ejemplo de ello se muestra en el siguiente fragmento de código perteneciente a la clase crearAction del módulo GLC (Gestionar Libro de Centro), donde se crean los objetos de la clase LibroCentros(), evidenciando así que la clase crearAction de GLC es creador de la clase LibroCentros().

```
class crearAction extends sfAction {

public function execute()
{
$centro = new LibroCentros();
$centro>setNombreCentro($this>getRequestParameter('nombre_centro'));
$centro->setFechaConfeccion('anno');
$centro->save();
}
```

Fig. 2 Ejemplo de código fuente donde se evidencia el uso del patrón Creador.

Alta Cohesión:

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

Controlador:

Todas las peticiones web son manejadas por un solo controlador frontal, 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 dirección URL entrada por el usuario.

Bajo Acoplamiento:

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

Patrones GoF.

Uno de los patrones de diseño GoF es el Singleton (Instancia única), patrón creacional diseñado para restringir la creación de objetos de una clase, o sea garantizar una sola instancia de una determinada clase y proporcionar un punto de acceso global a ella.

Symfony implementa este patrón mediante la clase sfContext. El singleton de contexto (que se obtiene mediante sfContext::getInstance()) almacena una referencia a todos los objetos que forman el núcleo de Symfony y puede ser accedido desde cualquier punto de la aplicación.

Para la implementación del sistema se aplicó además el patrón GoF Decorator (Envoltorio), patrón estructural que responde ante la necesidad de añadir dinámicamente funcionalidad a un objeto, evitando crear múltiples clases que contengan la misma funcionalidad, o que hereden de una misma clase que responda a una misma funcionalidad.

Este patrón se refleja en la implementación del archivo o componente layout.php, que como se ha explicado anteriormente es la plantilla global en todas las interfaces de la aplicación que almacena el código HTML que es común en ellas, evitando tener que repetirlo en cada página.

2.4 Vista de Despliegue

La Vista de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos).

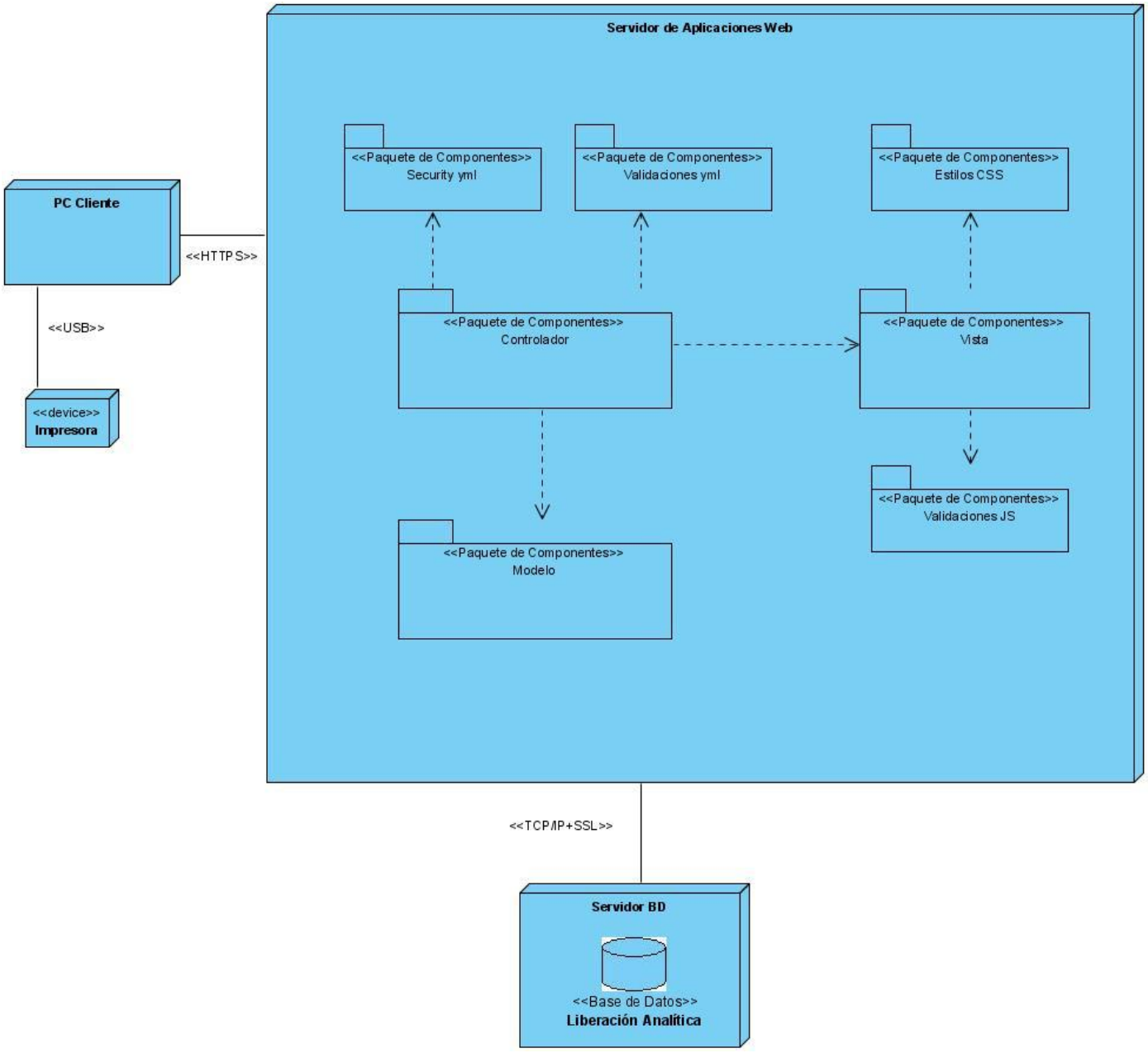


Fig. 4 Vista de Despliegue del módulo Liberación Analítica.

En la vista de despliegue mostrada en la figura anterior se observa la PC Cliente, que estará conectada al nodo dispositivo Impresora mediante el protocolo de comunicación Universal Serial Bus

USB, que a su vez estará conectado mediante el Protocolo seguro de Transferencia de Hipertexto HTTPS al nodo procesador que representa al Servidor de Aplicaciones Web.

En el servidor de aplicaciones se reflejan los componentes principales que se implementan y que se agrupan en siete paquetes de componentes. El paquete de componente Vista, agrupa todos los componentes relacionados con la interfaz del sistema, con el mismo se relacionan los paquetes de componentes Validaciones JS, que agrupa los componentes para las validaciones JavaScript y Estilos CSS.
El paquete de componentes Controlador agrupa los componentes relacionados con la lógica del negocio, es decir los componentes que responden a las funcionalidades del sistema, con dicho paquete se relaciona el paquete de componentes Security.yml, que agrupa los componentes que validan la seguridad del sistema y el paquete de componentes Validaciones.yml, que agrupa los componentes que garantizan las validaciones del sistema en el lado del servidor. Además está relacionado con el paquete de componente Vista para reflejar los resultados de las distintas acciones que se realizan y con el paquete de componentes Modelo para acceder a la información persistente en el sistema. El Modelo agrupa aquellos componentes que garantizan el acceso a la base de datos.

El servidor de aplicaciones estará conectado también al nodo procesador Servidor de base de datos donde se representa el componente de la base de datos a través del protocolo de TCP/IP+SSL, garantizando la seguridad de los datos a través de la encriptación de los datos.
El protocolo SSL (Secured Socket Layer) ha sido diseñado para dar seguridad al intercambio de datos entre dos aplicaciones, principalmente entre un servidor Web y un navegador. Este protocolo es ampliamente utilizado y es compatible con la mayoría de los navegadores web. Al nivel de la arquitectura de red, el protocolo SSL se inserta entre la capa TCP/IP (nivel bajo) y el protocolo de alto nivel HTTP.

2.5 Vista de Implementación: Diagrama de componente

Esta vista describe la descomposición del software en capas y subsistemas de implementación. También provee una vista de la trazabilidad de los elementos de diseño de la vista lógica para la implementación.

Esta contiene:

- Una enumeración de los subsistemas.
- Diagramas de componentes que ilustran la organización en capas y jerarquías de los subsistemas.
- Dependencia entre subsistemas.

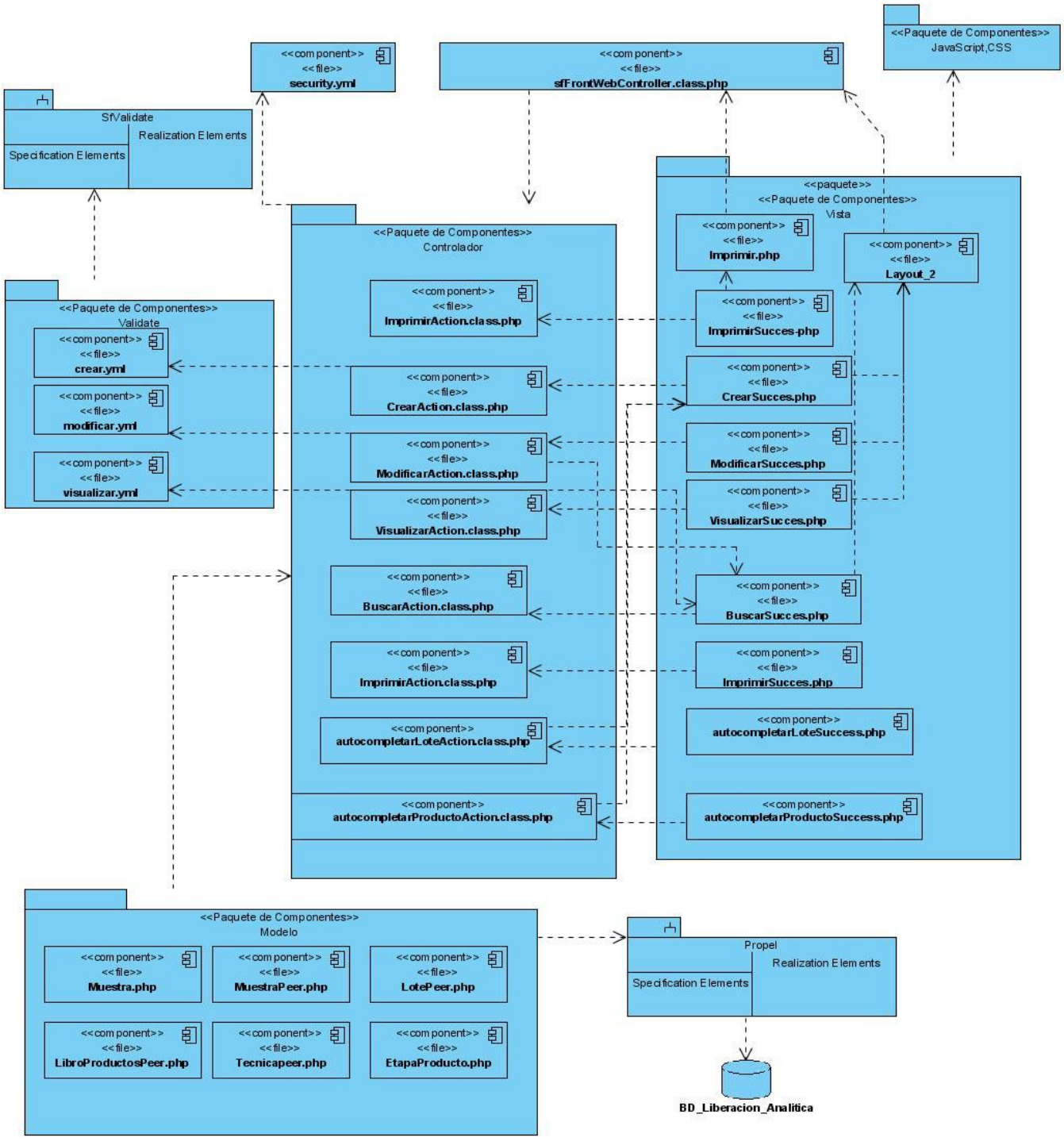


Fig. 3 Diagrama de Componentes del módulo Liberación Analítica.

En el diagrama antes visto se representan los componentes del caso de uso Gestiona Muestras, el mismo está realizado de acuerdo a la implementación que le da Symfony al patrón de arquitectura

MVC. En el diagrama se puede observar el componente sfController que representa la clase del controlador, y es el encargado de decodificar la petición y transferirla a la acción correspondiente, constituye el único punto de entrada al sistema.

El paquete de componentes Controlador contiene a todos los componentes .class.php encargados de realizar la lógica del negocio correspondiente al caso de uso en cuestión. Para restringir el acceso a cada una de las acciones se utiliza el componente security.yml encargado como su nombre lo indica de parte de la seguridad de la aplicación, pues en el mismo se asignan los privilegios de acceso a los usuarios, llamados credenciales, para así permitir que solo los usuarios con los permisos necesarios ejecuten determinadas acciones. Cada uno de los componentes del Controlador ejecuta un componente del paquete Vista, encargado este último de la presentación y que se relaciona además con el paquete JavaScript,CSS que contienen los archivos .js y .css.

En el paquete Validate contiene los componentes.yml donde a cada uno de estos se relaciona con un componente del Controlador, ya que estos son los encargados de validar la entrada de datos, por lo que este paquete se relaciona con el paquete sfValidate, el cual contiene todos los validadores que implementa Symfony.

El paquete Modelo contiene las clases generadas por el ORM que utiliza Symfony, en este caso Propel, que es representado mediante el subsistema Propel, las clases contenidas en el Modelo permiten el acceso a la base de datos sin importar el gestor de base de datos empleado.

2.6 Estándares de codificación

Para un mejor entendimiento del código en la implementación del sistema es necesario establecer un estándar de codificación. En el desarrollo del módulo Liberación Analítica se ha tenido en cuenta el estilo de código propuesto para la implementación en lenguaje PHP5. Se pueden señalar los siguientes aspectos:

- Las variables usadas para el control de un ciclo son nombradas con un solo carácter como i, j o k. Los signos lógicos y de operación se separan por un espacio antes y después de los mismos.
- Todas las variables y nombres de funciones a utilizar se definieron en idioma español.
- En el caso de los objetos que se utilizan como por ejemplo los campos de texto en su nombre incluyen el nombre asociado al valor que va contenido.

- Los nombres de las variables utilizadas comienzan en minúscula y son nemotécnicos, cortos, claros y describen su propósito.

En la parte superior de cada Action dentro de los módulos se describen datos importantes del segmento del programa, la palabra paquete identifica el nombre del Proyecto, el sub-paquete es el nombre del módulo, por ejemplo:

```
/**
 * SIC0020 actions.
 *
 * @package    Proyecto-Lims
 * @subpackage SIC0001
 */
```

2.7 Componentes reutilizables

En ocasiones es necesario incluir cierto código HTML o PHP en varias páginas para no tener que repetirlo. Symfony define 3 alternativas al uso de la instrucción include() y que permiten manejar de forma inteligente los fragmentos de código:

- Si el fragmento contiene poca lógica, se puede utilizar un archivo de plantilla al que se le pasan algunas variables. En este caso, se utilizan los elementos parciales (*partial*).
- Si la lógica es compleja (por ejemplo se debe acceder a los datos del modelo o se debe variar los contenidos en función de la sesión) es preferible separar la presentación de la lógica, en este caso, se utilizan componentes (*component*).
- Si el fragmento va a reemplazar una zona específica del layout, para la que puede que exista un contenido por defecto, se utiliza un *slot*.

Elementos Parciales

Un elemento parcial es un trozo de código de plantilla que se puede reutilizar. Al igual que las plantillas, los elementos parciales son archivos que se encuentran en el directorio templates/, y que contienen código HTML y código PHP. El nombre del archivo de un elemento parcial siempre comienza con un guión bajo (_), lo que permite distinguir a los elementos parciales de las plantillas, ya que todos se encuentran en el mismo directorio templates/.

Componentes

Un componente es como una acción, solo que mucho más rápido. La lógica del componente se guarda en una clase que hereda de sfComponents y que se debe guardar en el archivo

actions/components.class.php. Su presentación se guarda en un elemento parcial. Los elementos parciales que se utilizan como presentación de un componente, se deben llamar igual que los componentes, sustituyendo la palabra execute por un guión bajo. [10]

Ejemplo de código fuente

Debido a que en varios casos de uso del módulo de Liberación Analítica se deben registrar datos pertenecientes a un producto determinado previamente registrado en la base de datos, se decidió usar componentes de Symfony aprovechando las ventajas de su aplicación, en cuanto a reutilización de código.

El siguiente ejemplo de código corresponde a la lógica del componente Producto, ubicado en *modules/componentes/actions/components.class.php.*

```
class componentesComponents extends sfComponents
{

public function executeProducto()
{
        if($this->getRequestParameter('producto') != NULL)
        {
          $producto_id = $this->getRequestParameter('producto');
          $etapa_id = $this->getRequestParameter('etapa');
          $nombre_modulo = $this->getRequestParameter('modulo');
          $producto = LibroProductosPeer::retrieveByPK($producto_id);
          $etapa = EtapaProductoPeer::retrieveByPK($etapa_id);
        }
        else
        {
         $modulo = sfContext::getInstance()->getModuleName();
         $nombre_modulo = strtolower($modulo);
         $producto = NULL;
         $etapa = NULL;
        }
        $mt = false;
        $fecha_inicio = $this->fecha_inicio;
        $fecha_fin = $this->fecha_fin;
        switch ($nombre_modulo)
        {
                case 'gm':
                        $origenes = LibroProductosPeer::OrigenesExistentes($nombre_modulo);
                        $mt = false;
                        break;
                case 'sic0935':
                        $origenes = LibroMuestrasTestigosPeer::OrigenesMT($nombre_modulo);
```

```
                $mt = true;
                break;
            case 'sic0825':
                $origenes = Sic0935solicitudMtPeer::OrigenesSic0935SinDespachar();
                $mt = true;
                break;
            case 'sic0301':
                $origenes = Sic0301CertificadoPtPeer::OrigenesSic0301();
                $mt = false;
                break;
            case 'gmt':
                $origenes = LibroProductosPeer::OrigenesExistentes($nombre_modulo);
                $mt = false;
                break;
            case 'sic0931':
                $lotes = Sic0931NotifResNosPeer::lotesPosiblesSic0931();
                $origenes = Sic0931NotifResNosPeer::OrigenesSic0931($lotes);
                $mt = false;
                break;
        }
        $this->origenes = $origenes;
        $this->mt = $mt;
        $this->mod = $nombre_modulo;
        $this->producto = $producto;
        $this->etapa = $etapa;
    }
}
```

El siguiente ejemplo de código corresponde a la presentación del componente Producto, ubicado en *modules/componentes/templates/_producto.php*

```
<?php use_helper('Object') ?>
<?php use_helper('Javascript')?>

<tr>
 <td>
  <div>Origen:<?php echo select_tag('proviene', $origenes,'onchange = mostrar_Autorizar()')?></div>
 </td>
 <td>
  <div id="producto">Producto:<?php echo select_tag('productos', Array('-1'=>'---Seleccione---'),-1)?>
  <?php echo input_hidden_tag('mt',$mt)?>
  <?php echo input_hidden_tag('mod',$mod)?></div>
 </td>
 <td>
```

```
  <div id="etapa">Etapa:<?php echo select_tag('etapas', Array('-1'=>'---Seleccione---'),-1)?></div>
   </td>
</tr>
<?php echo observe_field('proviene', array(
                        'update' => 'producto',
                        'url' => 'componentes/actualizarProductos',
                        'script' => true,
                        'with' => '"proviene=' + value + '&mt=' + $('mt').value + '&mod=' + $('mod').value ",))
?>
<?php else : ?>

<?php echo observe_field('proviene', array(
                        'update' => 'producto',
                        'url' => 'componentes/actualizarProductos',
                        'script' => true,
                        'with' => '"proviene=' + value + '&mt=' + $('mt').value + '&mod=' + $('mod').value",))
?>
<?php endif; ?>
<?php echo observe_field('proviene', array(
                        'update' => 'etapa',
                        'url' => 'componentes/limpiarEtapas',
                        'script' => true,))
?>
```

Fig.5 Vista de la presentación del elemento parcial _producto correspondiente al componente producto.

2.8 JavaScript

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. JavaScript es un lenguaje orientado a objetos propiamente dicho. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM. Siendo el lenguaje que permite interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web dinamismo y vida. En la implementación del sistema se empleó JavaScript fundamentalmente para

validar los datos de entrada proveídos por el usuario, o sea que cada dato de entrada cumpliera con los estándares definidos por el cliente.

Ajax

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

JavaScript es el lenguaje interpretado en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Actualmente las aplicaciones web incluyen numerosas interacciones en el lado del cliente, ya sean efectos visuales complejos o comunicaciones asíncronas con el servidor, todos ellos se pueden realizar con JavaScript, pero programarlos resulta un tanto tedioso y requiere mucho tiempo para corregir posibles errores. Es por esto que Symfony incluye una serie de helpers que automatizan muchos de los usos comunes de JavaScript en las plantillas, programando la mayoría de los comportamientos del lado del cliente sin usar una línea de código JavaScript.

Observe_field.

Los formularios más modernos no solo se encargan de enviar los datos cuando el usuario pulsa el botón de envío, sino que también reaccionan a los cambios producidos por el usuario sobre alguno de sus campos, Symfony proporciona el helper observe_field () para realizar dicha tarea, el siguiente fragmento de código muestra un ejemplo de uso de dicho helper en el que se observa el campo de ***'cant_unidades',*** y donde cada carácter escrito en dicho campo lanza una petición Ajax que actualiza el valor del elemento **'cant_disponibles'** , a través de la ejecución de la acción actualizarCantDisponibles.

```
<?php echo observe_field(
'cant_unidades', array(
'update' => 'cant_disponibles',
'url' =>'SIC0935/actualizarCantDisponibles',
'script' => true,
'with' => "'cant_unidades=' + value + '&lote=' + $('lote').value",))
?>
```

Fig.6 Ejemplo de código fuente de un observe_field.

2.9 Seguridad

Seguridad en la acción

La posibilidad de ejecutar una acción puede ser restringida a usuarios con ciertos privilegios. Las herramientas proporcionadas por Symfony para este propósito permiten la creación de aplicaciones seguras, en las que los usuarios necesitan estar autenticados antes de acceder a alguna característica o a partes de la aplicación. Añadir esta seguridad a una aplicación requiere dos pasos: declarar los requerimientos de seguridad para cada acción y autenticar a los usuarios con privilegios para que puedan acceder estas acciones seguras. [6]

Restricción de acceso

Antes de ser ejecutada, cada acción pasa por un filtro especial que verifica si el usuario actual tiene privilegios de acceder a la acción requerida. En Symfony, los privilegios están compuestos por dos partes:

- Las acciones seguras requieren que los usuarios estén autenticados.
- Las credenciales son privilegios de seguridad agrupados bajo un nombre y que permiten organizar la seguridad en grupos.

Para restringir el acceso a una acción se crea y se edita un archivo de configuración YAML llamado ‘security.yml en el directorio config/ del módulo. En este archivo, se pueden especificar los requerimientos de seguridad que los usuarios deberán satisfacer para cada acción o para todas las acciones. [7]

Mecanismo de escape

Cuando se insertan datos generados dinámicamente en una plantilla, se debe asegurar la integridad de los datos. Por ejemplo, si se utilizan datos obtenidos mediante formularios que pueden rellenar usuarios anónimos, existe un gran riesgo de que los contenidos puedan incluir scripts y otros elementos maliciosos que se encargan de realizar ataques de tipo XSS (*cross-site scripting*). Por tanto,

se debe aplicar un mecanismo de escape a todos los datos mostrados, de forma que ninguna etiqueta HTML pueda ser peligrosa. [8]

2.10 Pruebas

La automatización de pruebas es uno de los mayores avances en la programación desde la invención de la orientación a objetos. Concretamente en el desarrollo de las aplicaciones web, las pruebas aseguran la calidad de la aplicación incluso cuando el desarrollo de nuevas versiones es muy activo. [9]

Automatización de pruebas

Probar de forma correcta una aplicación supone un gran esfuerzo, ya que crear casos de prueba, ejecutarlos y analizar sus resultados constituye una tarea tediosa. Además, es habitual que los requisitos de una aplicación varíen constantemente, con el consiguiente aumento del número de versiones de la aplicación y la refactorización continua del código. En este contexto, es muy probable que aparezcan nuevos errores.

Este es el motivo por el que la automatización de pruebas es una recomendación, aunque no una obligación, útil para crear un entorno de desarrollo satisfactorio. Los conjuntos de casos de prueba garantizan que la aplicación hace lo que se supone que debe hacer. Incluso cuando el código interno de la aplicación cambia constantemente las pruebas automatizadas permiten garantizar que los cambios no introducen incompatibilidades en el funcionamiento de la aplicación. Un buen conjunto de pruebas muestra la salida que produce la aplicación para una serie de entradas de prueba, por lo que es suficiente para entender el propósito de cada método.

Pruebas unitarias

Las pruebas unitarias aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular. Las pruebas unitarias se encargan de un único caso cada vez, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto.

Las pruebas unitarias no están pensadas para lanzarlas en un servidor de producción. Se trata de herramientas para el programador, por lo que solamente deberían ejecutarse en la máquina de desarrollo del programador y no en un servidor de producción.

El 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 de pruebas, llamado Lime.

Lime proporciona el soporte para las pruebas unitarias y es más eficiente que otros frameworks de pruebas de PHP y posee las siguientes ventajas:

- Ejecuta los archivos de prueba en un entorno independiente para evitar interferencias entre las diferentes pruebas.
- Las pruebas de Lime son fáciles de leer y sus resultados también lo son. En los sistemas operativos que lo soportan, los resultados de Lime utilizan diferentes colores para mostrar de forma clara la información más importante.
- Symfony utiliza Lime para sus propias pruebas y su “regression testing”, por lo que el código fuente de Symfony incluye muchos ejemplos reales de pruebas unitarias y funcionales.
- El núcleo de Lime se valida mediante pruebas unitarias.
- Está escrito con PHP, es muy rápido y está bien diseñado internamente. Consta únicamente de un archivo, llamado lime.php, y no tiene ninguna dependencia.

Pruebas Unitarias

Las pruebas unitarias de Symfony son archivos PHP normales cuyo nombre termina en Test.php y que se encuentran en el directorio test/unit/ de la aplicación. Su sintaxis es sencilla y fácil de leer.

Las siguientes figuras muestran las pruebas unitarias realizadas a los casos de uso

- Gestionar cálculo de la actividad específica.
- Gestionar cálculo de grupos funcionales residuales.
- Gestionar cálculo relación PRP: TT .
- Gestionar cálculo de la actividad biológica por bulbo.

sic0709Test.php | sic0794Test.php | sic0795Test.php | sic0938Test.php

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

function CalculoActEspecifica($a,$b)
{
    return Sic0709ActividadEspecificaPeer::CalcularActividadEspecifica($a,$b);
}

$t = new lime_test(1, new lime_output_color());
$t->is(CalculoActEspecifica(25,5),5,'Valor de actividad especifica calculada correctamente');
?>
```

Fig. 7 Prueba unitaria realizada al caso de uso gestionar cálculo de la actividad específica

sic0709Test.php | sic0794Test.php | sic0795Test.php | sic0938Test.php

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

 function CalculoGFR()
 {
    return Sic0794FuncionalesPeer::CalcularGruposFuncionalesRes(9);
 }

 $t = new lime_test(1, new lime_output_color());
 $t->is(CalculoGFR(),450,'Valor de los grupos funcionales residuales calculado correctamente');
?>
```

Fig. 8 Prueba unitaria realizada al caso de uso gestionar cálculo de grupos funcionales residuales.

sic0709Test.php | sic0794Test.php | sic0795Test.php | sic0938Test.php

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

function CalculoRel_PRP_TT($prp_total, $prp_libre, $tt)
{
    return Sic0795prpTtPeer::CalcularRelacion_PRP_TT($prp_total, $prp_libre, $tt);
}

$t = new lime_test(1, new lime_output_color());
$t->is(CalculoRel_PRP_TT(25,10,5),3,'Valor de relacion PRP:TT calculado correctamente');
?>
```

Fig. 9 Prueba unitaria realizada al caso de uso gestionar cálculo relación PRP: TT.

sic0709Test.php | sic0794Test.php | sic0795Test.php | sic0938Test.php

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

function CalculoAct_Biol_bulbo($act_biol, $vol_bulbo)
{
    return Sic0938ActividadBulboPeer::CalcularActividadBiologicaBulbo($act_biol, $vol_bulbo);
}

$t = new lime_test(1, new lime_output_color());
$t->is(CalculoAct_Biol_bulbo(25,10),250,'Valor de la actividad biologica por bulbo calculada correctamente');
?>
```

Fig. 10 Prueba unitaria realizada al caso de uso gestionar cálculo de la actividad biológica por bulbo.

La siguiente figura muestra los resultados obtenidos luego de realizadas las pruebas, las cuales cumplieron con el valor esperado.

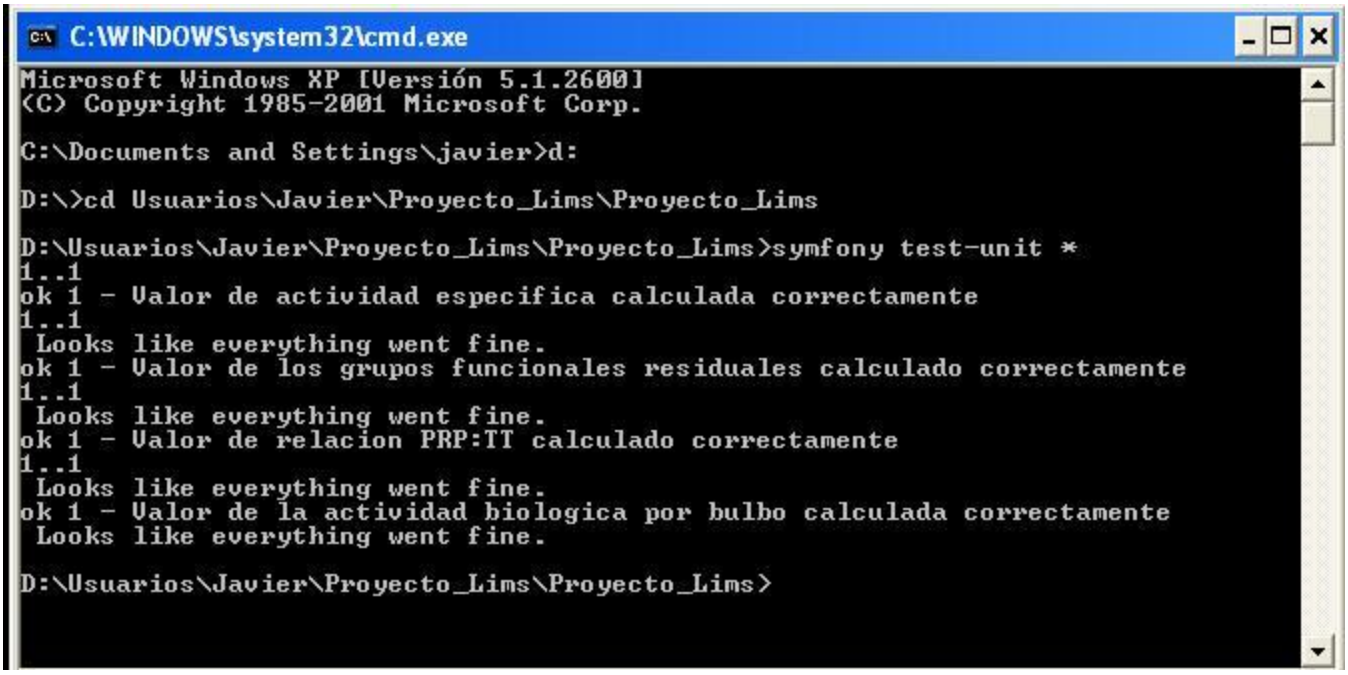


Fig. 11 Resultado de las pruebas unitarias realizadas a los casos de uso.

Conclusiones del Capítulo

Después de haber realizado la implementación del sistema se le dio solución a 28 requisitos funcionales, cumpliendo estrictamente con los requisitos no funcionales, y analizando además la arquitectura del sistema, así como el patrón de arquitectura MVC que utiliza Symfony. Se realizaron los diagramas de componentes para cada uno de los casos de uso con el propósito de lograr una mejor comprensión de la lógica de programación mediante componentes. Los estándares de codificación garantizaron un mejor entendimiento del código realizado y el manejo de los componentes reutilizables proporcionó una mayor eficiencia a la implementación. En todos los casos de usos la seguridad fue validada correctamente. Por otra parte se le efectuaron pruebas unitarias al código para comprobar su buen funcionamiento.

CONCLUSIONES GENERALES

Concluida la implementación del sistema, del módulo de Liberación Analítica de la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología, se concluye:

- El análisis de los resultados obtenidos en investigaciones anteriores proporcionó una mejor comprensión de los procesos que se realizan en el grupo de Liberación Analítica y favorecieron en la implementación del sistema.
- El estudio de los diferentes LIMS desarrollados en el mundo aportó elementos importantes para comprender el proceso de desarrollo de los LIMS y hacer más eficiente la funcionalidad del sistema
- La familiarización con la metodología de desarrollo de software, herramientas y tecnologías permitió un mejor desarrollo del sistema, aprovechando las potencialidades de las mismas.
- Se lograron implementar los 28 casos de uso del grupo Liberación Analítica identificados como componentes de implementación a los cuales se les realizó pruebas unitarias para eliminar posibles errores en el código.
- Con la implementación del sistema se logra dar solución al problema científico planteado y se cumple el objetivo general de esta investigación.

RECOMENDACIONES

- Implementar servicios que permitan la integración con equipos médicos.

- Implementar Servicios que contribuyan a la toma de decisiones. Como pudiera ser un Cuadro de Mando Integral.

- La integración con los módulos Análisis Químico, Biología Molecular, Inmunoquímica y Estudios de Estabilidad y Materiales de Referencia.

REFERENCIAS BIBLIOGRÁFICAS

1. CIGB Centro de Ingeniería Genética y Biotecnología. [En línea] [Citado el: 12 de noviembre de 2008.] http://www.cigb.edu.cu/index.php?option=com_content&task=view&id=55&Itemid=121..

2. Quass. [En línea] [Citado el: 2 de diciembre de 2008.] http://www.quaass.com/.

3. Química.es. [En línea] http://www.quimica.es/productos/es/52345/.

4. **Gutiérrez., Javier J.** ¿Que es un framework web? [En línea] [Citado el: 16 de enero de 2009.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

5. **Lago, Ramiro.** Patrones de diseño de software. [En línea] Abril de 2007. [Citado el: 2 de febrero de 2009.] http://www.proactiva-calidad.com/java/patrones/index.html.

6. **Fabien Potencier, François Zaninotto.** *Symfony la guía definitiva.* s.l. : Librosweb, 2008. pág. 107.

7. *Symfony la guía definitiva.* s.l. : Librosweb, 2008. pág. 108.

8. *Symfony la guía definitiva.* s.l. : Librosweb, 2008. pág. 144.

9. *Symfony la guía definitiva.* s.l. : Librosweb, 2008. pág. 318.

10. *Symfony la guía definitiva.* s.l. : Librosweb, 2008. pág. 129.

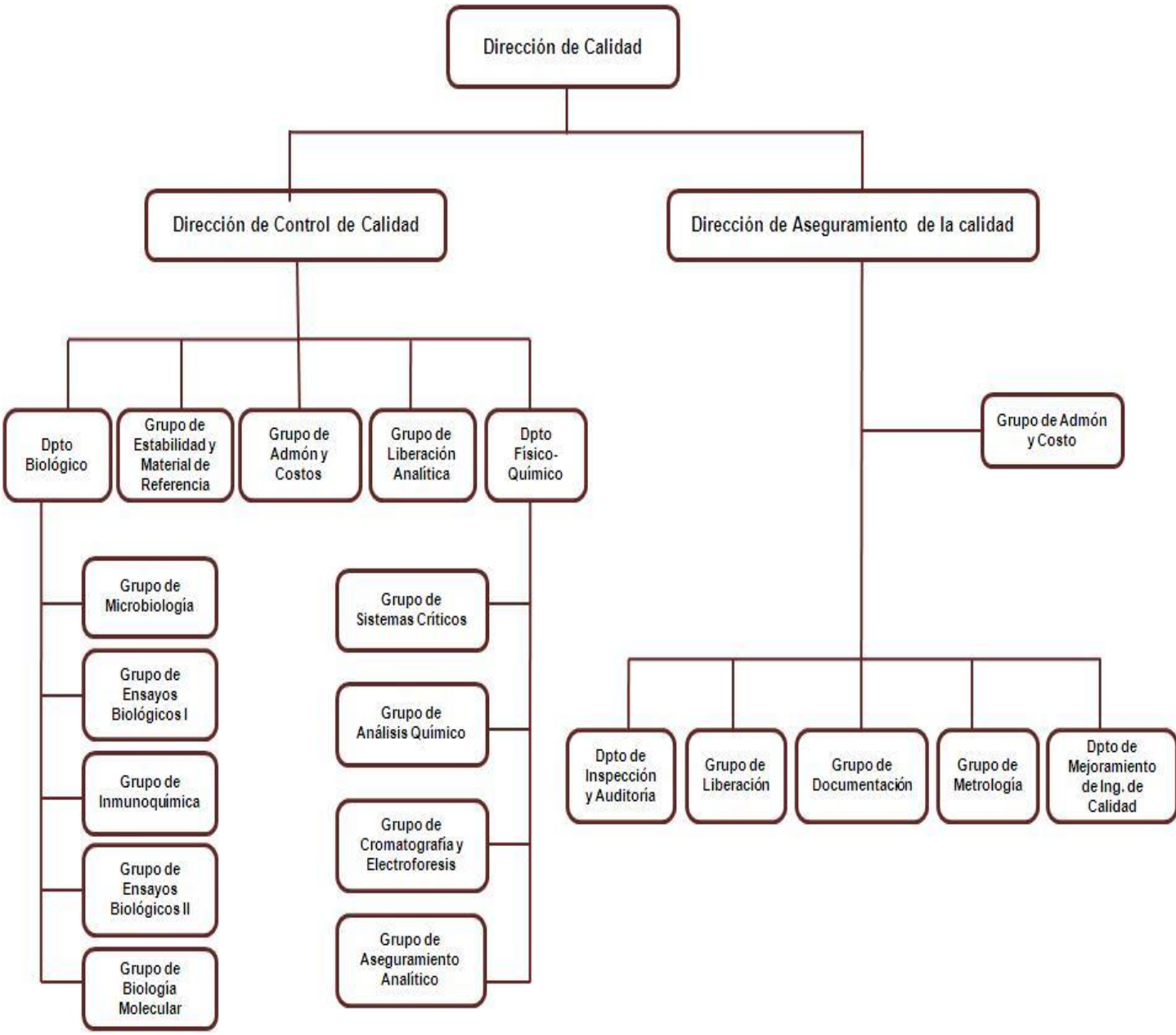
BIBLIOGRAFÍA

- **LibrosWeb.es.** Symfony. 2008. [En Línea] http://www.librosWeb.es/symfony/capitulo1/symfony_en_pocas_palabras.html
- **Casares, Claudio. 2003.** Maestros del Web. [En línea] 10 de agosto de 2003. www.maestrosdelweb.com/editorial/tutsql17/.
- **Fabien Potencier, François Zaninotto. 2008.** *Symfony la guí definitiva .* 2008.
- *Gestión de información, gestión del conocimiento y gestión de la calidad en las organizaciones.* **Quiroga, Lic. Lourdes Aja. 2002.** Ciudad de La Habana : s.n., 2002.
- **Kioskea.** [En línea].2007 http://es.kioskea.net/contents/genie-logiciel/design-patterns.php3.
- **Microsof**. [En línea] http://msdn.microsoft.com/es-es/library/bb972240.aspx.
- **proactiva-calidad**. [En línea] http://www.proactiva-calidad.com/java/patrones/index.html.
- **2004-2008.** WebTutoriales. [En Línea] 2004-2008. http://www.webtutoriales.com/tutoriales/programacion/php/comparar-fechas.73.html.
- **2007-2008.** Paraiso geek. [En Línea] 2007-2008. http://www.paraisogeek.com/la-funcion-date-en-php/.
- **2009.** Mi código beta. [En Línea] Walter, enero 14, 2009. http://micodigobeta.com.ar/?p=189.
- **CIGB.** Centro de Ingeniería Genética y Biotecnología. Centro de Ingeniería Genética y Biotecnología. [En Línea] CIGB, noviembre 12, 2008. [Citado: noviembre 12, 2008.] http://www.cigb.edu.cu/index.php?option=com_content&task=view&id=55&Itemid=121.
- **2003-2009.** Tutores. [En Línea] 2003-2009. http://www.tutores.org/?codigos=2&pg=2.
- **2009.** snippets.symfony-project. [En Línea] 2009. http://snippets.symfony-project.org/snippets/tagged/criteria/order_by/date.
- **Quass,** ¿POR QUÉ NECESITO UN LIMS?, [En Línea, Citado: 1/12/2007], Disponible en: http://www.quaass.com/web/unlims_1.htm.
- **2008-2009**. BIKA lab system. [En línea] 2008-2009. http://www.bikalabs.com/
- **2008.** open source laboratory automation &informatics. [En línea] 2008. http://www.labmatica.com/

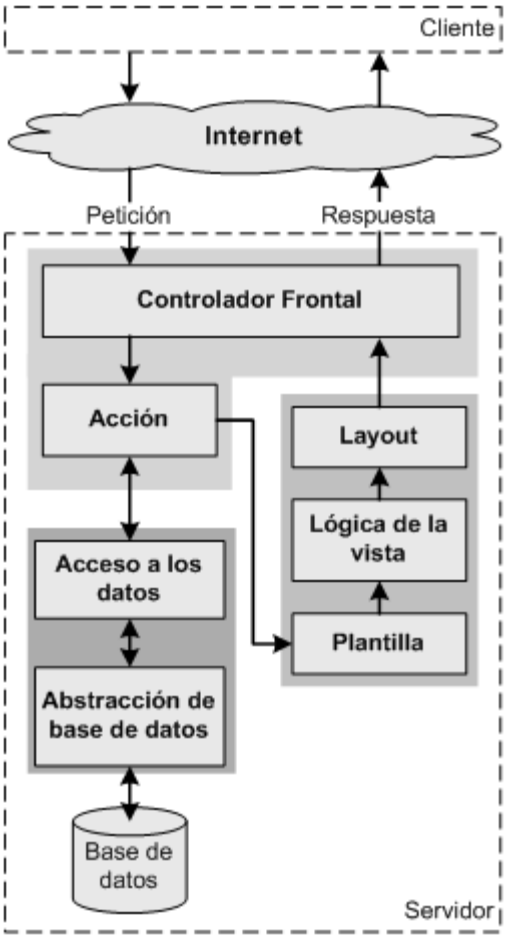
- **Roman Konertz, PD Dr. Ludwig Eichinger and Dr. Budi Tunggal. 2008.** Open LIMS**. [En línea] 2008. http://www.open-lims.org/**

ANEXOS

Anexo 1: Organigrama de la Dirección de Calidad del CIGB.



Anexo 2: El flujo de trabajo de Symfony



GLOSARIO DE TÉRMINOS

C

CIGB: Centro de ingeniería Genética y Biotecnología.

Cliente/Servidor: Los distintos aspectos que caracterizan a una aplicación (proceso, almacenamiento, control y operaciones de entrada y salida de datos) en el sentido más amplio, están situados en más de un computador, los cuales se encuentran interconectados mediante una red de comunicaciones.
Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información.

CSS: Las hojas de estilo en cascada (*Cascading Style Sheets*) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML.

21 CFR Parte 11: la regla 21 CFR Parte 11 “Electronic Records; Electronic Signatures” (ER / ES) establece los requisitos para la grabación de datos electrónicos y el uso de firmas electrónicas. La intención de esta regla es facilitar la introducción de tecnología electrónica a la manufactura y producción. La Parte 11 busca dar guías de sentido común en cuanto a cómo hacer de forma electrónica lo que antes se hacía manualmente.

F

FDA: Acrónimo de Food and Drug Administration (Administración de Alimentos y Fármacos, por sus siglas en inglés) es la agencia del gobierno de los Estados Unidos responsable de la regulación de alimentos (tanto para seres humanos como para animales), suplementos alimenticios, medicamentos (humanos y veterinarios), cosméticos, aparatos médicos (humanos y animales), productos biológicos y productos hemáticos.

G

GNU: La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente su acrónimo del inglés GNU GPL, es una licencia creada por la Free

Software Foundation a mediados de los ´80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

H

HTML: Siglas de HyperText Markup Language (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

I

Iteración: Se refiere a la acción de repetir una serie de pasos un cierto número de veces.

J

Java: es un lenguaje de programación orientado a objetos. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Java VM: Es una Máquina virtual Java (en inglés Java Virtual Machine, JVM), un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.

L

LIMS: Acrónimo de Laboratory Information Management Systems (Sistema de Gestion de Informacion de los Laboratorios por sus siglas en inglés).

M

MySQL: es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

Multiplataforma: Término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

P

PDA: Del inglés Personal Digital Assistant (Asistente Digital Personal), es un computador de mano originalmente diseñado como agenda electrónica (calendario, lista de contactos, bloc de notas y recordatorios) con un sistema de reconocimiento de escritura. Hoy día (2009) estos dispositivos, pueden realizar muchas de las funciones de una computadora de escritorio (ver películas, crear documentos, juegos, correo electrónico, navegar por Internet, reproducir archivos de audio, etc.)pero con la ventaja de ser portátil.

Plone: Plone es un Sistema de Gestión de Contenidos o CMS por sus siglas en inglés (Content Management System), basado en Zope y programado en Python. En un desarrollo basado en código abierto. Puede utilizarse como servidor intranet o extranet, un Sistema de Publicación de documentos y una herramienta de trabajo en grupo para colaborar entre entidades distantes.

R

Regression Testing: Consiste en realizar un juego de pruebas predefinidas cada vez que se hace alguna implementación nueva o se soluciona un bug en el cual debimos modificar gran parte del código o "tocar" la aplicación en varios lugares. A lo que apunta es, para ponerlo de manera sencilla, a controlar que no rompamos una cosa mientras arreglamos otra.

S

Software: La palabra software se refiere al equipamiento lógico o soporte lógico de un computador digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema.

SOAP:(siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Es uno de los protocolos utilizados en los servicios Web.

T

TIC: Tecnologías de la Información y las Comunicaciones.

U

UTF-8: Es un tipo de codificación de caracteres para Unicode que nos permite escribir nuestras páginas web y no preocuparnos por si se va a ver correctamente o van a aparecer caracteres extraños.

X

XSS: Su nombre original es "Cross Site Scripting", y es abreviado como XSS para no ser confundido con las siglas CSS, (hojas de estilo en cascada). Las vulnerabilidades de XSS originalmente abarcaban cualquier ataque que permitiera ejecutar código de "scripting", como VBScript o Java Script, en el contexto de otro sitio web (y recientemente esto se podría clasificar más correctamente como "distintos orígenes").

Y

YAML: Acrónimo de "YAML Ain't Another Markup Language (en castellano, "YAML no es otro lenguaje de marcado"). A comienzos de su desarrollo, YAML significaba "Yet Another Markup Language" ("Otro lenguaje de marcado más") para distinguir su propósito centrado en los datos en lugar del marcado de documentos. Sin embargo, dado que se usa frecuentemente XML para serializar datos y XML es un auténtico lenguaje de marcado de documentos, es razonable considerar YAML como un lenguaje de marcado ligero.

Z

Zope: es un servidor de aplicaciones web escrito en el lenguaje de programación Python. Puede ser manejado casi totalmente usando una interfaz de usuario basada en páginas Web. Un sitio web de Zope está compuesto de objetos en lugar de archivos, como es usual con la mayoría de los otros sistemas de servidores web. Las ventajas de usar objetos en lugar de archivos son.