Universidad de las Ciencias Informáticas

Facultad 6

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

Trabajo de Diploma para optar por el Título de Ingeniero Informático

Autores:

Indira Bello García

Aldis Joan Abreu Medina

Tutores:

Ing. Aida Portelles Valdés

Ing. Pedro Manuel Puig Díaz

Co-Tutor:

Ing. Elennis Díaz Laurencio

junio, 2009

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 _____.

Indira Bello García

_____

FIRMA DE LA AUTORA

Ing. Aida Portelles Valdés

_____

FIRMA DE LA TUTORA

Aldis Joan Abreu Medina

_____

FIRMA DEL AUTOR

Ing. Pedro Manuel Puig Díaz

_____

FIRMA DEL TUTOR

DATOS DE CONTACTO

Ing. Aida Portelles Valdés
Profesora recién graduada en la universidad de las Ciencias Informáticas.
Email: aportelles@uci.cu

Ing. Pedro Manuel Puig Díaz
Profesor recién graduado en la universidad de las Ciencias Informáticas.
Universidad de las Ciencias Informáticas, Habana, Cuba.
Email: pmpuig@uci.cu

Ing. Elennis Díaz Laurencio
Profesora graduada en la universidad de las Ciencias Informáticas con dos años de experiencia.
Universidad de las Ciencias Informáticas, Habana, Cuba.
Email: edíaz@uci.cu

AGRADECIMIENTOS:

De Indira:

A mis padres por ser mi sostén y mi guía, por ser mi orgullo y mis ejemplos. Por haber estado a mi lado siempre y creer en mí.

A mis abuelos por la grandeza de sus almas, por ese amor y cariño que me ofrecen cada día.

A mi hermana por llenar mi vida de alegría y optimismo.

Pocas personas tienen la dicha de tener dos madres, yo como una de esas pocas personas me siento orgullosa y dichosa de tenerlas, a mi tía Janet por ser esa segunda madre.

A mis primos Roly y Roger por cuidarme tanto, por su cariño y apoyo en todo, por ser los hijos que mis padres no tuvieron y los hermanos que siempre quise.

A mi familia por regalarme que cada día me sienta orgullosa de tenerlos. Por esa unidad y ese amor que nos caracteriza, por el cariño y apoyo que siempre he encontrado en cada uno de ustedes.

A los amigos que he conocido, a los que están y los que no están, a los que se han ido y los que no se han ido, a los que fueron y a los que son, gracias de todo corazón. Les agradezco que me quieran más allá de mis defectos que son muchos, les agradezco su apoyo incondicional en cada paso del camino recorrido.

A mi "Piquete LIMS" por soportarme durante todo este tiempo, soportar ese pesimismo que me caracteriza, esos ataques de pánico cuando pensaba que no se podía, por su comprensión, su amistad y por regalarme gratos e inolvidables recuerdos de la UCI. En especial a Javier por todo lo que luchó conmigo y lo mucho que me ayudó siempre.

A mi compañero de tesis por su paciencia y su confianza en que todo nos saldría bien.

A mis tutores por su paciencia, por su apoyo, por el ánimo y la fuerza que nos inculcaban. Por confiar en nosotros.

A todas aquellas personas que contribuyeron en mi formación como persona y como profesional, que han logrado que le ponga todo mi empeño a lo que realmente amo y en lo que realmente creo.

A la revolución y a Fidel que me dieron la oportunidad de hacer realidad este sueño.

AGRADECIMIENTOS:

De Aldis:

A todos los del LIMS por el tiempo dedicado a nuestro trabajo, todos siempre compartiendo conocimientos con un mismo objetivo.

A Javier por no tener nunca un no como respuesta.

A nuestros tutores por el apoyo incondicional en todo momento y sus palabras de fuerza.

DEDICATORIA:

De Indira:

A mis padres por su apoyo constante, por su amor y confianza en mí, a quienes debo lo que soy y he logrado en la vida. Todos mis éxitos llevan sus nombres.

A Alberto por estar siempre a mi lado, por brindarme su apoyo incondicionalmente, por demostrarme que a veces es preciso caminar entre la sombra y que siempre podemos levantarnos y volver a comenzar.

De Aldis:

A mis padres, por dedicar toda su vida a la mía, por no abandonar ni un momento y estar conmigo en casa tarea que me ha presentado la vida , a su lado solo hay una cosa segura y es el triunfo ….

A mi abuelo, cada consejo en mi niñez fue una herramienta para la vida…….

A mi abuela, por aquel valioso tiempo a mi educación…….

A puchita por compartir todo momento como si fuera suyo, cada victoria, cada derrota, cada lágrima……..

A mi madrina, Cyndi, boqui la trucha Borges, el fleco Borges presentes en cada momento de estos 5 años dando su dosis de fuerza…….

A Tamara, a la rata, Mileydi, Hugo, Milagro por enseñarme las otras cosa de la vida que también se deben aprender para los momentos especiales y por terminar con mis momentos de hambre insaciable.

A los amigos que quedaron, al barrio, a los profesores que formaron parte de mi formación….

Al grupo antiguo grupo 6207, los mejores momentos en la UCI quedaron en él….

A la revolución por ser tan grande y a Fidel por hacer realidad el sueño de este joven…

RESUMEN

El Centro de Ingeniería Genética y Biotecnología (CIGB) es uno de los centros más importantes del país como sistema de investigación-producción en la esfera de la Biotecnología, aplicada a diversas ramas de la sociedad. La dirección de calidad del centro, teniendo en cuenta la importancia del control y aseguramiento de la calidad de sus producciones y ante el manejo existente de grandes volúmenes de información relacionados con sus actividades, se ha trazado como objetivo implantar un Sistema para la Gestión de la Información eficiente que controle el acceso a la información generada y su correcto almacenamiento. Es en la Universidad de las Ciencias Informáticas (UCI) donde se desarrolla actualmente un Sistema de Gestión de Información de los Laboratorios (LIMS, del inglés Laboratory Information Management System) como solución a esta problemática.

Las principales actividades del Grupo de Biología Molecular de este centro se ven directamente afectadas por el control manual de la documentación generada como resultado de sus acciones y procesos investigativos fundamentalmente. El presente trabajo de diploma describe la implementación de los procesos que realiza el grupo de Biología Molecular, dando continuidad a la investigación de un grupo de analistas que anteriormente identificaron los procesos a informatizar, así como los requisitos funcionales y el diseño para una primera versión del sistema.

PALABRAS CLAVE: CIGB, Control y aseguramiento de la calidad, Gestión de la Información, LIMS.

Tabla de Contenido

INTRODUCCIÓN **1**

CAPÍTULO **5**

FUNDAMENTACIÓN TEÓRICA **5**

1.1 Sistemas de Gestión de Información de Laboratorio. Aplicaciones 5

1.2 Aplicaciones informáticas desarrolladas para la gestión de información de Laboratorios. 6

1.3 Metodología de Desarrollo de Software. 9

1.4 Herramientas y Tecnologías actuales. 12

Herramienta CASE de modelado. 12

Lenguajes de programación. 12

Servidor Web: Apache 2.2.6 13

Entorno de Desarrollo: Eclipse 3.3.1.1 13

Framework de desarrollo. 14

Sistema gestor de base de datos 15

1.5 Patrones 15

Patrón de Arquitectura 15

Patrones de Diseño. 16

1.6 Conclusiones del capítulo 18

CAPÍTULO **19**

CARACTERÍSTICAS DEL SISTEMA **19**

2.1 Requerimientos Funcionales. 19

2.2 Requisitos No Funcionales 26

2.3 Arquitectura del sistema 29

Vista de despliegue 29

Patrón de Arquitectura. 31

Patrones de Diseño. 32

2.4 Diagramas de Componentes. 35
2.5 Estándares de codificación 38
2.6 Componentes reutilizables 39
2.7 Representación del código principal. 40
2.8 Análisis de la seguridad del sistema implementado. 45
2.9 Pruebas. 46
2.10 Conclusiones del capítulo 48
CONCLUSIONES 50
RECOMENDACIONES 51
REFERENCIAS BIBLIOGRÁFICAS. 52
BIBLIOGRAFÍA. 54
ANEXOS. 56
GLOSARIO DE TÉRMINOS 57

INTRODUCCIÓN

A partir de los años 60 el uso de las Tecnologías de la Información y las Comunicaciones (TIC) inicia una nueva etapa de desarrollo socioeconómico y la configuración de una sociedad superior. Toda actividad política, social, económica y cultural es mediada por procesos electrónicos de almacenamiento, procesamiento y transmisión de datos a través de computadoras. Se conforma todo un espacio virtual de relaciones del que surgen nuevas vías de comunicación, producción e intercambio financiero.

Cuba no ha quedado atrás en este sentido. Previendo el uso racional y eficiente trata de elevar al máximo el empleo de las TIC a todas las esferas de la sociedad. Se ha comenzado a trabajar en los lugares más importantes del país, pilares de su desarrollo económico y social, centros donde la principal tarea es garantizar la producción, el cuidado del medio ambiente, la alimentación, el desarrollo industrial y fundamentalmente la salud humana.

El Centro de Ingeniería Genética y Biotecnología (CIGB) es uno de los principales productores en el campo de la Biotecnología en Cuba. Inmerso en la tarea de informatización, el CIGB ha planificado la vinculación de sus actividades al empleo de las TIC. De esta forma se lograría minimizar la labor de los trabajadores del centro y una gestión eficiente en cuanto al control, acceso y manejo de la información de manera que aumente la producción y comercialización de productos biológicos.

La estructura del Centro, descrita en el Anexo 1, está conformada por la Dirección de Calidad compuesta por las direcciones de Control de la Calidad y Aseguramiento de la Calidad.

“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 nuestros productos y servicios satisfacen los requisitos de calidad establecidos. Vela por el cumplimiento de las Buenas Prácticas de Producción (BPP), Buenas Prácticas de Laboratorio (BPL) y Buenas Prácticas Clínicas (BPC)” (1). 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, entre las cuales están las relacionadas con el muestreo, las especificaciones, los ensayos y la evaluación de la calidad de los productos que se generan en el centro.

Para el desempeño de las mismas, cuenta con la ayuda de dos departamentos con sus grupos de trabajo específicos y cuenta además con 3 grupos de trabajo:

- Grupo de Estudios de Estabilidad y Materiales de Referencia.
- Grupo de Administración y Costo.
- Grupo de Liberación Analítica.
- 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.
    - Grupo de Inmunoquímica.
    - Grupo de Ensayos Biológicos.
    - Grupo de Biología Molecular.

El grupo de Biología Molecular es el encargado de realizar los ensayos de Determinación de ADN contaminante en muestras del ingrediente farmacéutico activo (IFA) y muestras de proceso de cada uno de los productos del Centro mediante modernas técnicas de biología molecular entre las que se encuentran la hibridación de ácidos nucléicos y la reacción en cadena de la polimerasa (PCR). Efectúa los análisis de estabilidad de los Bancos de Células Primario (BCP) y los Bancos de Células de Trabajo (BCT) de todas las producciones del Centro y también es responsable de la certificación y custodia del BCP y de la certificación del BCT. Al mismo tiempo realiza la preparación de los Bancos y chequeo de las cepas microbiológicas que intervienen en las producciones del CIGB, conjuntamente con los análisis de restricción, determinación de la concentración de ADN por su densidad óptica a 260 nm y la determinación de RNA contaminante y el porciento de supercoil por electroforesis de agarosa a productos de ADN. (2)

El grupo de Biología Molecular, así como los demás grupos de la dirección de calidad, presenta afectaciones y atrasos en el desarrollo de sus técnicas debido a que el almacenamiento de datos en

documentos carece de un soporte digital o plataforma organizativa que centralice este flujo de información. Por tal motivo a mediados del 2006 este Centro decidió en conjunto con la Universidad de las Ciencias Informáticas (UCI) iniciar un proyecto productivo en la Facultad 6 de esta última institución con el objetivo de desarrollar un sistema gestor de información para el CIGB: Sistema de Gestión de Información de los Laboratorios (LIMS, del inglés Laboratory Information Management System).Tomando como premisa el desarrollo de los flujos de trabajo de Modelamiento del Negocio y Análisis y Diseño, definidos por la metodología de desarrollo: Proceso Unificado de Desarrollo (RUP, de sus siglas en inglés Rational Unified Process), el presente trabajo de diploma pretende dar continuidad al proceso de desarrollo del módulo Biología Molecular, donde la gestión de la Información no fluye todavía de manera eficiente, asegurando la calidad de sus actividades.

Por todo lo antes expuesto se propone como **Problema Científico** de esta investigación: ¿Cómo obtener un producto funcional para el módulo Grupo de Biología Molecular del LIMS de Calidad del CIGB?

El problema planteado se enmarca en el **Objeto de Estudio**: Los Sistemas de Gestión de Información de los Laboratorios, que específicamente aborda como **Campo de Acción:** El proceso de desarrollo de los Sistemas de Gestión de Información de los Laboratorios de Calidad.

Para dar solución al problema se define como **Objetivo General:** Implementar el Módulo Biología Molecular del Sistema de Gestión de la Información de los Laboratorios de la Dirección de Calidad del CIGB.

Para lograr este objetivo, se realizarán las siguientes **Tareas de Investigación**:

1. Estudio de la documentación obtenida en las iteraciones anteriores realizadas al módulo Biología Molecular.
2. Familiarización con las herramientas y tecnologías definidas por la arquitectura del proyecto.
3. Implementación del módulo de Biología Molecular.
4. Realización de pruebas de unidad.
5. Realización de los diagramas de componentes.
6. Validación de la Seguridad del módulo.

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

Capítulo I: "Fundamentación Teórica". Se analizan las perspectivas actuales del desarrollo de Sistemas de Gestión de Información de Laboratorio como elemento distintivo en el proceso de informatización a nivel mundial, del que se nutre la base teórico conceptual de esta investigación. A continuación se definen las características fundamentales de la metodología de desarrollo y herramientas, definidas por la arquitectura, que son utilizadas en la implementación del módulo. Además se manifiestan las tendencias actuales de los roles que participan en el proceso de desarrollo del software en el flujo de trabajo de implementación.

Capítulo II: "Implementación de la solución". Se describe la implementación de los componentes a partir de los requerimientos identificados. Se describe el diagrama de componentes para uno de los casos de uso implementados. Además se muestran fragmentos relevantes del código, mecanismos de implementación y ejemplos de las validaciones realizadas; así como ejemplos de las pruebas de unidad aplicadas al sistema implementado que permite examinar la estructura interna del programa y garantizar la calidad del software.

CAPÍTULO 1
FUNDAMENTACIÓN TEÓRICA

En este capítulo se analizan las perspectivas actuales del desarrollo de los Sistemas de Gestión de Información de Laboratorio como elemento distintivo en el proceso de informatización a nivel mundial, del que se nutre la base teórico conceptual de esta investigación. A continuación se definen las características fundamentales de la metodología de desarrollo y herramientas, definidas por la arquitectura, que son utilizadas en la implementación del módulo. Además se manifiestan las tendencias actuales de los roles que participan en el proceso de desarrollo del software en el flujo de trabajo de implementación.

1.1 Sistemas de Gestión de Información de Laboratorio. Aplicaciones

El desarrollo tecnológico, como resultado de la evolución del pensamiento humano, constituye un proceso incremental que aún no alcanza sus límites e influye directamente en el desarrollo de la sociedad. La era digital o revolución tecnológica actual sitúa a la sociedad en un proceso de informatización que fructifica del potencial de las nuevas tecnologías.

La Informatización de la Sociedad es el proceso de utilización ordenada y masiva de las Tecnologías de la Información y las Comunicaciones en la vida cotidiana, para satisfacer las necesidades de todas las esferas de la sociedad, en su esfuerzo por lograr cada vez más eficacia y eficiencia en todos los procesos y por consiguiente mayor generación de riqueza y aumento en la calidad de vida de los ciudadanos. (3)

En este sentido el proceso de informatización está dando un nuevo enfoque a la producción de software. La modernización de las empresas se ha inclinado al desarrollo de sistemas administrativos informatizados, definidos como sistemas de gestión de información, que agilizan la sistematización de los datos e información. Los Sistemas de Gestión de Información, por sus ventajas en cuanto a la automatización de procesos, constituyen una alternativa imprescindible y han llegado a convertirse en una herramienta integral de gerencia, necesaria para alcanzar con éxito los resultados propuestos de una organización y lograr ventajas competitivas a través de su implantación.

Un LIMS es una variante de un Sistema de Gestión de Información, para laboratorios. Su principal diferencia es que frecuentemente son utilizados en la industria y generalmente dirigidos a investigaciones o análisis comerciales.

Un 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, archivo, trazabilidad, entre otros, y minimizando los errores provocados por la transferencia de información. (4)

El uso de un LIMS como herramienta integral para garantizar la gestión moderna del control y aseguramiento de la calidad, permite la gestión, evaluación y almacenamiento de la información de laboratorios de investigación, de desarrollo o de prestación de servicios en diferentes sectores industriales como el químico, medioambiental, agroalimentario y el farmacéutico. Puede automatizar funciones y procesos de rutina. Reduce el tiempo empleado en cálculos o revisiones, los errores humanos o el tiempo de entrega de un producto determinado. Aporta soluciones eficaces permitiendo hacer rápidos análisis de datos en control de calidad o para identificar tendencias y admite el enlace o interacción con participantes externos, ya sean departamentos u otra empresa.

El término de LIMS surge a finales del siglo pasado y debido a las potencialidades que ofrece, algunas empresas a nivel mundial se encargan de desarrollarlos.

1.2 Aplicaciones informáticas desarrolladas para la gestión de información de Laboratorios.

Matrix LIMS

El Matrix LIMS fue desarrollado por Autoscribe Limited, empresa líder mundial en el desarrollo de LIMS por el uso de innovadoras herramientas en estos productos (5). Se caracteriza por su configuración auténtica al ser suministrado con un conjunto de herramientas de configuración de uso sencillo, que garantizan que puedan diseñarse y personalizarse un número ilimitado de flujos de trabajo en un laboratorio y especificar cómo se enlazan las diferentes interfaces del sistema, por lo que permite adaptar el Matrix LIMS a cualquier industria y laboratorio y solucionar las necesidades de grandes o pequeñas empresas.

Compatible con una variedad de bases de datos comerciales, incluyendo Oracle y Microsoft SQL Server, en dependencia de las preferencias del usuario y con posibilidades también de configuración con múltiples grupos de bases de datos. (5)

LabWare LIMS

LabWare LIMS es un Sistema de Gestión de Información de Laboratorios que opera en diferentes plataformas según la cantidad de usuarios que utilicen el sistema y se adapta a diferentes tipos de empresas. Es modular, por lo que el sistema puede expandirse de acuerdo a las necesidades de trabajo. Posee acceso equivalente cliente/servidor y Web, y se integra en cualquier entorno informático. La aplicación corre sobre Windows. (6)

Gracias a su completa funcionalidad para seguimiento de muestras, certificación de usuarios, gestión de instrumentos, auditoría, y planificación de muestras e informes, así como muchas otras funciones, LabWare LIMS cumple las Buenas Prácticas de Laboratorio. LabWare LIMS fue desarrollado por PerkinElmer, empresa de Estados Unidos que se ha ido consolidando como una de las empresas líderes en el mercado de instrumental analítico de la más alta calidad y en soporte técnico. (6)

Se caracteriza por la amigabilidad y sencillez mediante el uso de menús tradicionales y el bajo costo de operación debido a que el cliente puede configurar y mantener el LIMS por sí mismo. Como base de datos se pueden usar todos los productos comerciales, como ORACLE™, SQL Server™ o PostgreSQL que son de las más usadas actualmente.

BIKA LIMS

Bika LIMS se estructura en un servidor estándar de edición para que los clientes puedan agregar módulos opcionales y personalizaciones. Emplea la tecnología moderna, independiente de la plataforma y basados en la web.

Se construye en Plone, un marco de gestión de contenido que está estrechamente integrado con Zope, que es un servidor de aplicaciones Web ampliamente utilizado de fuente abierta orientado a objetos (7). Esta liberado bajo la licencia pública general (GPL).

Open LIMS

El objetivo de Open LIMS es desarrollar una organización y estructura independiente de software. Los módulos son generalmente implementados en PHP. Para el cálculo de resultados cuenta con el sistema de trabajo que está escrito en Java y requiere Java VM (Virtual Machine). Es posible trabajar en proyectos paralelos. Proporciona módulos experimentales para la posterior análisis. Es posible organizar el experimento con diferentes subproyectos. Cada proyecto puede tener un número ilimitado de subproyectos. Se basa en plantillas las cuales están escritos en XML (Extensible Markup Language). Permite al propietario del proyecto establecer toda una serie de permisos, dividido en

diferentes tipos de permisos. Utiliza la codificación de caracteres UTF8 .Publicado bajo la licencia GPL, este todavía se encuentra en desarrollo.

Los laboratorios de biotecnología utilizan estos tipos de herramientas para recopilar, validar, archivar, hacer informes, analizar y gestionar sus datos. El sistema permite a los usuarios recopilar los datos generados en cualquiera de los sistemas de la empresa, y organizar dentro de un sistema de gestión de documentos electrónicos.

Los laboratorios de biotecnología se enfrentan al crecimiento exponencial de los datos, regulaciones estrictas, las presiones para acelerar el descubrimiento de fármacos y el tiempo del mercado. La elección de hacer uso de un LIMS le hace frente a estos desafíos, donde al mismo tiempo, se hace cada vez mayor la comunicación, colaboración e integridad de los datos.

El sistema que se plantea desarrollar debe ser diseñado para incluir un alto nivel de configuración, lo cual es importante en la biotecnología, donde los nuevos métodos y equipos se incorporan en los laboratorios con mucha frecuencia. A través de la automatización de procesos y la integración con la instrumentación se logra mayor velocidad. Disminuye la entrada de datos.

Para Cuba y específicamente el CIGB, constituye un asunto engorroso el hecho de comprar un LIMS debido a términos de costos, actualizaciones, licencias y adaptabilidad del software al sistema de producción. La mayoría de los proveedores de LIMS en el mundo pertenecen a Estados Unidos y por tanto el bloqueo económico implantado imposibilitaría su adquisición.

Los procesos desarrollados en el CIGB son tan variados y complejos que resulta difícil adaptarlos a LIMS libres como Bika LIMS, teniendo en cuenta que estos procesos se comportan de manera diferente a como lo gestionan estos LIMS. Considerando las ventajas que ofrece un LIMS, se quiere desarrollar en el CIGB, polo científico de Biotecnología en Cuba, un LIMS que exponga el máximo esfuerzo que realiza Cuba por alcanzar un desarrollo vertiginoso en cuanto a la aplicación de nuevas tecnologías de la informatización y explotar al máximo las potencialidades que este desarrollo ofrece. El objetivo de desarrollar un primer LIMS en el país no es tan solo minimizar las necesidades propias del CIGB, sino extender su aplicación a las principales empresas y polos científicos del país; que constituya además una línea base para investigaciones posteriores y que garantice fundamentalmente la calidad de las producciones de cualquier entidad o institución que lo posea.

1.3 Metodología de Desarrollo de Software.

Todo desarrollo de software es riesgoso y difícil de controlar. Lo importante es saber determinar la metodología adecuada a utilizar, para ello es necesario definir cuáles son las características o condiciones en la que se desarrollará el software y cuáles ofrecen cada una de las metodologías.

Una de las metodologías más usadas para el desarrollo de aplicaciones robustas es la metodología RUP (por sus siglas en inglés Rational Unified Process).

La metodología RUP es el proceso unificado de desarrollo, que plantea quién hace qué, cuándo y cómo. Tiene como objetivo asegurar la producción de software de calidad dentro de plazos y presupuestos predecibles.

RUP es diseñada para desarrollar grandes y complejos proyectos, con la ventaja de unificar los mejores elementos de metodologías anteriores a ella. Es una metodología que se centra en la arquitectura y donde los procesos se caracterizan por ser iterativos e incrementales en cada una de sus fases. Es dirigida por casos de uso, orientada a objetos y utiliza el lenguaje de modelado unificado para la representación visual. 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.

El presente trabajo de diploma se enmarcará en el desarrollo del flujo de trabajo de Implementación, de mayor peso en la fase de construcción, y donde se define cómo se organizarán las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes, así como la estructura de capas de la aplicación.

Los trabajadores que participan en este flujo de trabajo son:

- Programador.
- Integrador del sistema.
- Arquitecto del software.

Las principales actividades que se realizan:

- Estructurar el modelo de Implementación.
- Planificar la Integración (Plan de Integración).
- Implementar componentes.

El rol definido por RUP para el flujo de trabajo de implementación es el rol de Desarrollador.

“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. Los roles describen

cómo los individuos se comportan en el negocio y qué responsabilidades tienen”.**¡Error! No se encuentra el origen de la referencia.** (8)

Trabajadores del Rol Desarrollador:

- Arquitecto de software: Conduce y coordina las actividades y los artefactos técnicos a través del proyecto. El Arquitecto de Software establece la estructura total para cada visión arquitectónica: la descomposición de la vista, la agrupación de elementos, y las interfaces entre agrupaciones mayores. Por lo tanto, en contraste con otros roles, la visión del Arquitecto de Software es más amplia en comparación con otras.
- Implementador: Responsable de desarrollar y de probar componentes de acuerdo con los estándares adoptados en el proyecto para la integración en subsistemas más grandes. El implementador es también responsable de desarrollar y de probar los componentes de prueba y los subsistemas correspondientes.
- Integrador: Combina los componentes probados para producir una estructura. Un integrador es también responsable de planear la integración, que ocurre en los niveles del subsistema y de sistema con cada uno teniendo un espacio de trabajo separado de integración.

Cada rol produce artefactos específicos como resultados de las actividades que ellos realizan. Un artefacto es un conjunto de información que es producida, modificada o usada como resultado tangible de actividades definidas por el proceso de desarrollo de software.

Artefactos del Arquitecto de Software:

- Modelo de implementación: El modelo de implementación describe como se implementan los elementos del modelo del diseño, dígase las clases del diseño, en términos de componentes (ficheros de código fuente, binario o ejecutables) y como se organizan a partir de los mecanismos de estructuración y modularización de acuerdo al entorno de desarrollo para la implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes uno de otro.
- Diagrama de despliegue: Muestra la configuración de tipos de nodos del sistema, en los cuales se hará el despliegue de los componentes. En el flujo de trabajo de implementación se actualiza asignando a los nodos los componentes ejecutables.
- Descripción de la arquitectura: El papel de esta descripción es guiar al equipo a través del ciclo de vida del sistema. Se reflejan las 4 más 1 vista de la arquitectura de los casos de uso arquitectónicamente significativo. En el flujo de trabajo de Implementación específicamente se

actualiza el documento con la vista de implementación y se refina la vista de la arquitectura del modelo de despliegue, donde los componentes ejecutables son asignados a nodos.

Artefactos del Implementador.

- Diagramas de componentes: Los diagramas de componentes modelan la vista estática de un sistema. Estructuran el modelo de implementación en términos de subsistemas de implementación y muestran las relaciones de dependencia (compilación, ejecución) entre los elementos de implementación, además de interfaces que estos pueden presentar.
- Implementación de Subsistemas: Un subsistema de implementación es una colección de componentes y otros subsistemas de implementación. Incluyen dependencias y pueden implementar interfaces que exporten su funcionalidad en forma de operaciones.

Los subsistemas de implementación permiten estructurar el modelo de implementación en piezas manejables que se integren y prueben de forma separada. Deben seguir la traza uno a uno de los subsistemas de diseño correspondiente.

Artefactos del rol Integrador.

- Plan de integración o construcciones: Describe la secuencia de pasos o acciones necesarias en cada iteración para construir el software incrementalmente. En cada paso se obtienen pequeños problemas de integración o prueba que constituyen una versión ejecutable del sistema o parte específica de él. El plan de integración se centra en cuales subsistemas de implementación podrían ser implementados y un orden para efectuar la integración al sistema en la iteración actual.

El plan de integración contiene:

- Enumerados él o los Casos de Uso que se están construyendo.
- Enumerados los Subsistemas de Implementación que se integrarán. Es importante que el orden en que se enumeren coincidan con el orden a seguir para su integración o construcción. Orden en que se sumarán los subsistemas de implementación, y qué componentes de ellos en cada construcción.

Lenguaje de modelado.

RUP define como lenguaje de modelado el Lenguaje Unificado de Modelado (UML del inglés Unified Model Language). El uso de este lenguaje permite, visualizar, especificar, construir y documentar artefactos de un sistema de software. Garantiza un entendimiento haciendo uso de una notación unificada que propicia el intercambio entre los usuarios y los desarrolladores.

El uso de UML reduce el tiempo de desarrollo, permite la modelación de sistemas orientados a objetos y garantiza una mejor planeación y control a partir de la reutilización y minimización de costos.

1.4 Herramientas y Tecnologías actuales.

A continuación se presentan las herramientas y tecnologías actuales definidas en el proyecto que posibilitan el desarrollo e implementación del LIMS de Calidad, específicamente el grupo de Biología Molecular citado en este trabajo de diploma.

Herramienta CASE de modelado.

Las Herramientas CASE (Computer-Aided Software Engineering, en español Ingeniería de Software Asistida por Computadoras) son aplicaciones informáticas que permiten la automatización del proceso de desarrollo de un sistema y que despunta a una nueva cultura de ingeniería.
Las CASE son reconocidas a nivel mundial por los resultados que los desarrolladores han obtenido al emplearla en sus aplicaciones, garantizando la exactitud y minuciosidad de los procesos y entre los procesos. A pesar de que algunas herramientas CASE no ofrecen soluciones potenciales en el análisis de los requerimientos de la aplicación, garantizan la aplicación práctica de metodologías de desarrollo de software agilizando el trabajo y la productividad en los sistemas de información. Contribuyen a elevar la calidad y la estandarización de la documentación a partir de la utilización de gráficos. Permite la reutilización de componentes, portabilidad de las aplicaciones. Facilita el trabajo con la gestión de proyecto, diseño de prototipos, generación de códigos y el chequeo de errores.

Visual Paradigm for UML 6.1 Enterprise Edition.

Es una de las herramienta CASE que ha sido diseñada para las actividades de los múltiples roles en un proceso de desarrollo de software. Al utilizar el UML como lenguaje de modelado, permite modelar diseños orientados a objetos y la generación de códigos para diferentes gestores de base de datos como MySQL, Oracle y PostgreSQL, y lenguajes de programación como C#, JAVA y PHP.

Su principal ventaja y el motivo por el cual ha sido seleccionada por el proyecto como la herramienta CASE a utilizar es por ser una herramienta multiplataforma. Su uso garantiza la calidad en todo el ciclo de vida del software al permitir la comunicación entre los desarrolladores, ya que garantiza un leguaje común para todos los roles que intervienen en el proceso de desarrollo del software.

Lenguajes de programación.

PHP 5.2.5.

PHP es un lenguaje de “código abierto” interpretado de alto nivel, con una sintaxis muy sencilla y fácil de aprender. PHP es un lenguaje script usado normalmente para el desarrollo de páginas Web

dinámicas, del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación, cuyos fragmentos de código se intercalan fácilmente en páginas HTML. Ha sido el lenguaje de programación seleccionado por el proyecto por las características que posee que lo hacen superior a las demás versiones, y es que PHP5 presenta un entorno de programación orientado a objetos mucho más completo con respecto a versiones anteriores, que permite que el código PHP proporcione un alto rendimiento a las aplicaciones Web empresariales a nivel de las plataformas J2EE y .NET (9). Dispone de una alta conectividad con la mayoría de Sistemas de Gestión de Bases de Datos. Incluye gran cantidad de funciones y no requiere la definición de tipos de variables ni manejo detallado del bajo nivel.

Javascript

Javascript es un lenguaje utilizado para crear pequeños programas que luego son insertados en una página Web y en programas más grandes, orientados a objetos mucho más complejos. Es un lenguaje basado en acciones que posee menos restricciones. Gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, utilización de teclas, cargas de páginas entre otros. Este es un lenguaje interpretado, no requiere compilación y es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera y Mozilla Firefox.

Por todas las características mencionadas anteriormente ha sido utilizado para las validaciones del sistema del lado del cliente.

Servidor Web: 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 (10)**.**

Dentro de las principales características que posee y por las que se establece en el proyecto que debe ser utilizado se encuentra la eficiencia, flexibilidad y rapidez; permitiendo personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es multiplataforma e ideal para instalar máquinas GNU/Linux, que aseguran un sistema operativo con comunicaciones excelentes.

Entorno de Desarrollo: Eclipse 3.3.1.1

El eclipse es definido como el entorno de desarrollo a utilizar, ya que permite ejecutar programas sobre cualquier plataforma. Es una extensible plataforma de código abierto para desarrollar herramientas o

aplicaciones clientes de cualquier tipo. Compila en tiempo real y utiliza editores de sintaxis resaltada para HTML, CSS, XML.
Es ampliamente utilizada por sus peculiares características de haber sido diseñada para facilitar el desarrollo de una aplicación.

Pese a que Eclipse esté escrito en su mayor parte en Java (salvo el núcleo), se ejecute sobre máquina virtual de ésta y su uso más popular sea como un IDE para Java, Eclipse es neutral y adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, Cobol, C#, XML, PHP.
La característica clave de Eclipse es la extensibilidad. Eclipse es una gran estructura formada por un núcleo y muchos plug-ins que van conformando la funcionalidad final. La forma en que los plug-ins interactúan es mediante interfaces o puntos de extensión; así, las nuevas aportaciones se integran sin dificultad ni conflictos (11).

Framework de desarrollo.

Un framework simplifica el desarrollo de un sistema mediante la aplicación y automatización de patrones utilizados para resolver tareas comunes. Proporciona estructura al código fuente legible y más fácil de mantener. Encapsula operaciones complejas en instrucciones sencillas por lo que facilita la programación de aplicaciones.

Symfony 1.0.18.

Symfony es un framework diseñado para optimizar el desarrollo de aplicaciones Web al separar la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web, facilitar varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja y automatizar las tareas más comunes. Symfony tiene la ventaja de una fácil instalación y configuración en la mayoría de las plataformas, fácil de extender permitiendo su integración con librerías desarrolladas por terceros, y es independiente del Sistema Gestor de Base de Datos.

Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. Sigue la mayoría de las mejores prácticas y patrones de diseño para la Web. Es fácil de extender, lo que permite su integración con librerías de otros fabricantes.

Por todas las características mencionadas anteriormente y teniendo en cuenta que Symfony es uno de los framework de desarrollo PHP más populares, profesional y con mucha documentación se define

como el framework de desarrollo a utilizar para dar solución al problema científico de esta investigación.

Sistema gestor de base de datos.

Un Sistema de Gestión de Bases de Datos (SGBD) se define como un conjunto de programas, procedimientos y lenguajes que proporcionan las herramientas necesarias para trabajar con una base de datos. Reúne 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.

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. Dentro de las características que identifican al PostgreSQL 8.2 como el gestor de base de datos a utilizar se encuentran 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.

1.5 Patrones

Los patrones contribuyen con la experiencia colectiva de la ingeniería de Software al ocuparse de los problemas recurrentes y proporcionar un vocabulario o entendimiento común.

Son una abstracción de “problema-solución” e identifican y especifican abstracciones de niveles más altos que componentes o clases individuales.

Los patrones presentan varias clasificaciones de acuerdo a las características y soluciones que ofrecen. Los patrones de arquitectura y de diseño se clasifican como patrones de producto de software, definidos como un conjunto de literatura sobre la resolución de problemas habituales en el diseño de software orientado a objetos.

Patrón de Arquitectura

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. (12)

Los patrones de arquitectura ofrecen el esquema fundamental de organización para sistemas de software, donde cada actividad de desarrollo se rige por esta estructura. Proveen un conjunto de

subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos.

Modelo Vista Controlador

El patrón de arquitectura Modelo Vista Controlador pertenece a la familia de los estilos arquitectónicos de Llamada y Retorno.

El MVC propone la solución ante la problemática del cambio de las interfaces en tiempo de ejecución sin que se afecte el núcleo del código de la aplicación. Para ello divide el sistema en tres partes: modelo, vista y controlador. El modelo encapsula los datos y la funcionalidad de la aplicación. La vista o las vistas, teniendo en cuenta que pueden existir varias vistas, despliega la información contenida en el modelo. El controlador está asociado a cada vista, recibe entradas que traduce en invocaciones de métodos del Modelo o de Vista. El usuario interactúa con el sistema solamente vía controladores.

El MVC es un patrón ampliamente utilizado en múltiples plataformas y lenguajes. Las vistas proveen mayor flexibilidad y agilidad. Garantiza una mayor claridad en el diseño y facilita una mayor escalabilidad.

Por las características y soluciones que el patrón Modelo Vista Controlador (MVC) ofrece se ha identificado como el patrón de arquitectura a utilizar para desarrollar el sistema. Además es el patrón fundamental en el que se basa Symfony.

Patrones de Diseño.

Los patrones de diseño ofrecen esquemas para definir estructuras de diseño con las que construir sistemas software.

Con el uso de patrones de diseño se evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente. Permite formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño.

Patrones GoF.

Los patrones Gang of Four (GoF) se adaptan a problemas generales de diseño en un contexto particular al describir como se comunican las clases y objetos entre sí. Facilitan el aprendizaje y la comunicación entre programadores y diseñadores. Su clasificación varía en dependencia del factor solución: los creacionales resuelven problemas relativos a la creación de objetos, entre los que se encuentran Singleton, Abstract Factory y Factory Method. Los estructurales resuelven problemas relativos a la composición de objetos, entre los que se encuentran Proxy, Decorator y Composite.

Los patrones GoF de comportamiento resuelven problemas relativos a la interacción entre objetos, entre los que se encuentran Observer y Strategy.

Patrones GRASP.

GRASP es el acrónimo de General Responsability Assignment Software Patterns (Patrones de Software para la asignación General de Responsabilidad).

Estos patrones de diseño describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades, entendiéndose responsabilidades como obligaciones de un objeto respecto a su comportamiento generalmente reflejadas en dos tipos:

- Conocer:
  - ✓ Conocer los datos privados encapsulados.
  - ✓ Conocer los objetos relacionados.
  - ✓ Conocer las cosas que puede derivar o calcular.
- Hacer:
  - ✓ Hacer algo él mismo, como crear un objeto o hacer un cálculo.
  - ✓ Iniciar una acción en otros objetos.
  - ✓ Controlar y coordinar actividades en otros objetos.

  Entre los patrones GRASP más usados se encuentran los patrones Creador, Controlador, Alta Cohesión, Bajo Acoplamiento y Experto.

1.6 Conclusiones del capítulo

En este capítulo se analizó la situación a nivel mundial respecto a la producción de LIMS y su aporte al desarrollo tecnológico en el proceso de informatización, a partir de lo cual se evidenció la gran importancia de desarrollar para el CIBG este LIMS de Calidad. A partir del análisis de los conceptos fundamentales que definieron el estado del arte de nuestro objeto de estudio y teniendo en cuenta los requisitos para el desarrollo de la solución se plantean las características fundamentales de cada herramienta a utilizar en el proceso de implementación de la solución. Por tal motivo, haciendo uso de la metodología de desarrollo de software RUP, se considera implementar el módulo de Biología Molecular de acuerdo a las actividades que se definen en el flujo de trabajo de implementación. La herramienta CASE a utilizar será el Visual Paradigm, que al igual que RUP utilizan el UML como lenguaje de modelado. El sistema gestor de base de datos es el PostgreSQL. El módulo será implementado en PHP5 y Javascript para la validación, con Eclipse como entorno de desarrollo y Symfony como framework de desarrollo.

CAPÍTULO 2
CARACTERÍSTICAS DEL SISTEMA

En este capítulo se describe la implementación de los componentes a partir de los requerimientos identificados. Se describe el diagrama de componentes para uno de los casos de uso implementados. Además se muestran fragmentos del código fuente de importancia para la implementación, mecanismos de implementación, casos de pruebas de unidad e integración y ejemplos de las validaciones realizadas; así como ejemplos de las pruebas de unidad aplicadas al sistema implementado que permite examinar la estructura interna del programa y garantizar la calidad del software.

La implementación se comienza con los resultados del diseño e implementa el sistema en términos de componentes, teniendo como propósito principal el desarrollo de la arquitectura y el sistema como un todo.

2.1 Requerimientos Funcionales.

Un requerimiento funcional se define como la condición o capacidad que el sistema debe cumplir. Como resultado del flujo de trabajo de Requerimientos definido por la metodología de desarrollo RUP se identifican los siguientes requisitos funcionales del sistema a implementar:

Requisitos Funcionales.

CU Gestionar Registro de Temperatura (SIC-0125R).

RF1.1 Crear nuevo Registro de Temperatura.

RF1.2 Modificar datos del Registro de Temperatura.

RF1.3 Registrar traza del Registro de Temperatura.

RF1.4 Buscar el Registro de Temperatura .

RF1.5 Visualizar el Registro de Temperatura .

RF1.6 Imprimir el Registro de Temperatura.

CU Gestionar registro de Descargo para banco de células (SIC-0190).

RF2.1 Crear un nuevo registro de Descargo para banco de células.

RF2.2 Modificar datos del registro de Descargo para banco de células.

RF2.3 Registrar traza del registro de Descargo para banco de células.

RF2.4 Buscar el registro de Descargo para banco de células.

RF2.5 Visualizar el registro de Descargo para banco de células.

RF2.6 Imprimir el registro de Descargo para banco de células.

CU Gestionar registro de Determinación de la Estabilidad Plasmídica (SIC -0149).

RF3.1 Crear nuevo registro de Determinación de la Estabilidad Plasmídica.

RF3.2 Modificar el registro de Determinación de la Estabilidad Plasmídica.

RF3.3 Registrar traza de Determinación de la Estabilidad Plasmídica.

RF3.4 Buscar el registro de Determinación de la Estabilidad Plasmídica.

RF3.5 Visualizar el registro de Determinación de la Estabilidad Plasmídica.

RF3.6 Imprimir el registro de Determinación de la Estabilidad Plasmídica.

CU Gestionar Registro de Determinación del marcador de selección (SIC- 0152).

RF4.1 Crear nuevo Registro de Determinación del marcador de selección .

RF4.2 Modificar datos del Registro de Determinación del marcador de selección .

RF4.3 Registrar traza del Registro de Determinación del marcador de selección .

RF4.4 Buscar el Registro de Determinación del marcador de selección .

RF4.5 Visualizar el Registro de Determinación del marcador de selección .

RF4.6 Imprimir el Registro de Determinación del marcador de selección.

CU Gestionar Registro de Muestreo de los bancos de Células (SIC-0951).

RF5.1 Crear nuevo Registro de Muestreo de los bancos de células.

RF5.2 Modificar datos del Registro de Muestreo de los bancos de células.

RF5.3 Registrar traza.

RF5.4 Buscar el Registro de Muestreo de los bancos de células.

RF5.5 Visualizar el registro de Muestreo de los bancos de células.

RF5.6 Imprimir registro de Muestreo de los bancos de células.

CU Gestionar Cronograma de chequeo de bancos de células (SIC-0130).

RF6.1 Crear nuevo Cronograma de chequeo de bancos de células.

RF6.2 Modificar datos del Cronograma de chequeo de bancos de células.

RF6.3 Registrar traza.

RF6.4 Buscar el Cronograma de chequeo de bancos de células.

RF6.5 Visualizar el Cronograma de chequeo de bancos de células.

RF6.6 Imprimir el Cronograma de chequeo de bancos de células.

CU Gestionar Certificado de Liberación de los bancos de Células transformados (SIC-0229).
RF7.1 Crear nuevo Certificado de Liberación de los bancos de Células transformados.
RF7.2 Modificar datos del Certificado de Liberación de los bancos de Células transformados.
RF7.3 Registrar traza del Certificado de Liberación de los bancos de Células transformados.
RF7.4 Buscar el Certificado de Liberación de los bancos de Células transformados.
RF7.5 Visualizar el Certificado de Liberación de los bancos de Células transformados.
RF7.6 Imprimir el Certificado de Liberación de los bancos de Células transformados.

CU Gestionar Solicitud de destrucción de bancos de células (SIC-0840).
RF8.1 Crear nueva Solicitud de destrucción de bancos de células.
RF8.2 Modificar datos del registro de Solicitud de destrucción de bancos de células.
RF8.3 Registrar traza del registro de Solicitud de destrucción de bancos de células.
RF8.4 Buscar el registro de Solicitud de destrucción de bancos de células.
RF8.5 Visualizar el registro de Solicitud de destrucción de bancos de células.
RF8.6 Imprimir registro de Solicitud de destrucción de bancos de células.

CU Gestionar Registro de Destrucción de Banco de Células (SIC-0841).
RF9.1 Crear nuevo Registro de Destrucción de Banco de Células.
RF9.2 Modificar datos del Registro de Destrucción de Banco de Células.
RF9.3 Registrar traza del Registro de Destrucción de Banco de Células.
RF9.4 Buscar el Registro de Destrucción de Banco de Células.
RF9.5 Visualizar el Registro de Destrucción de Banco de Células.
RF9.5 visualizar el Registro de Destrucción de Banco de Células.
RF9.6 Imprimir Registro de Destrucción de Banco de Células.

CU Gestionar Informe de análisis de los bancos de células (SIC-0950).
RF10.1 Crear nuevo Informe de análisis de los bancos de células.
RF10.2 Modificar datos de Informe de análisis de los bancos de células.
RF10.3 Registrar traza de Informe de análisis de los bancos de células.
RF10.4 Buscar el Informe de análisis de los bancos de células.
RF10.5 Visualizar el Informe de análisis de los bancos de células.
RF10.6 Imprimir el Informe de análisis de los bancos de células.

CU Gestionar Caracterización de productos de ADN (SIC-0798).

RF11.1 Crear nueva Caracterización de productos de ADN.

RF11.2 Modificar datos de la Caracterización de productos de ADN.

RF11.3 Registrar traza de la Caracterización de productos de ADN.

RF11.4 Buscar la Caracterización de productos de ADN.

RF11.5 Visualizar la Caracterización de productos de ADN .

RF11.6 Imprimir Caracterización de productos de ADN.

CU Gestionar Chequeo de restricción de plásmidos (SIC-0186).

RF12.1 Crear nuevo registro de Chequeo de restricción de plásmidos.

RF12.2 Modificar datos del Chequeo de restricción de plásmidos.

RF12.3 Registrar traza del Chequeo de restricción de plásmidos.

RF12.4 Buscar Chequeo de restricción de plásmidos.

RF12.5 Visualizar Chequeo de restricción de plásmidos.

RF12.6 Imprimir Chequeo de restricción de plásmidos.

CU Gestionar Registro de Extracción de ADN de células de organismos superiores (SIC-0187).

RF13.1 Crear nuevo Registro de Extracción de ADN de células de organismos superiores.

RF13.2 Modificar datos de Registro de Extracción de ADN de células de organismos superiores.

RF13.3 Registrar traza de Registro de Extracción de ADN de células de organismos superiores.

RF13.4 Buscar Registro de Extracción de ADN de células de organismos superiores.

RF13.5 Visualizar Registro de Extracción de ADN de células de organismos superiores.

RF13.6 Imprimir Registro de Extracción de ADN de células de organismos superiores.

CU Gestionar Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino Masivo (SIC-0178).

RF14.1 Crear nuevo Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino Masivo.

RF14.2 Modificar datos del Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino Masivo.

RF14.3 Registrar traza del Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino Masivo.

RF14.4 Buscar Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino Masivo.

RF14.5 Visualizar Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino Masivo.

RF14.6 Imprimir Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino Masivo.

CU Gestionar Registro de Purificación de ADN plasmídico de *E.Coli* por Minialcalino (SIC-0733).

RF15.1 Crear nuevo Registro de Purificación de ADN plasmídico de *E.Coli* por Minialcalino.

RF15.2 Modificar datos de Registro de Purificación de ADN plasmídico de *E.Coli* por Minialcalino.

RF15.3 Registrar traza de Registro de Purificación de ADN plasmídico de *E.Coli* por Minialcalino.
RF15.4 Buscar Registro de Purificación de ADN plasmídico de *E.Coli* por Minialcalino.
RF15.5 Visualizar Registro de Purificación de ADN plasmídico de *E.Coli* por Minialcalino.
RF15.6 Imprimir Registro de Purificación de ADN plasmídico de *E.Coli* por Minialcalino.

CU Gestionar Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino de 50 mL(SIC-0734).
RF16.1 Crear nuevo Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino de 50 mL.
RF16.2 Modificar datos del Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino de 50 mL.
RF16.3 Registrar traza del Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino de 50 mL.
RF16.4 Buscar Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino de 50 mL.
RF16.5 Visualizar Registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino de 50 mL.
RF16.6 Imprimir registro de Purificación de ADN plasmídico de *E.Coli* por Alcalino de 50 mL.

CU Gestionar Registro de Extracción de ADN de alto peso de bacteria (SIC-0150).
RF17.1 Crear nuevo Registro de Extracción de ADN de alto peso de bacteria.
RF17.2 Modificar datos del Registro de Extracción de ADN de alto peso de bacteria.
RF17.3 Registrar traza del Registro de Extracción de ADN de alto peso de bacteria.
RF17.4 Buscar Registro de Extracción de ADN de alto peso de bacteria.
RF17.5 Visualizar Registro de Extracción de ADN de alto peso de bacteria.
RF17.6 Imprimir Registro de Extracción de ADN de alto peso de bacteria.

CU Gestionar Registro de Extracción de ADN de alto peso de levadura (SIC-0721).
RF18.1 Crear nuevo Registro de Extracción de ADN de alto peso de levadura.
RF18.2 Modificar datos del Registro Extracción de ADN de alto peso de levadura.
RF18.3 Registrar traza del Registro Extracción de ADN de alto peso de levadura.
RF18.4 Buscar Registro de Extracción de ADN de alto peso de levadura.
RF18.5 Visualizar Registro de Extracción de ADN de alto peso de levadura.
RF18.6 Imprimir Registro de Extracción de ADN de alto peso de levadura.

CU Gestionar Registro de Extracción de ADN de hojas de tabaco (SIC-0787).
RF19.1 Crear nuevo Registro de Extracción de ADN de hojas de tabaco.
RF19.2 Modificar datos de Registro de Extracción de ADN de hojas de tabaco.
RF19.3 Registrar traza de Registro de Extracción de ADN de hojas de tabaco.

RF19.4 Buscar Registro de Extracción de ADN de hojas de tabaco.

RF19.5 Visualizar Registro de Extracción de ADN de hojas de tabaco.

RF19.6 Imprimir registro de Extracción de ADN de hojas de tabaco.

CU Gestionar Registro de Determinación del Patrón de Integración por Southern Blot (SIC-0151).

RF20.1 Crear nuevo Registro de Determinación del Patrón de Integración por Southern Blot.

RF20.2 Modificar datos del Registro Determinación del Patrón de Integración por Southern Blot.

RF20.3 Registrar traza del Registro Determinación del Patrón de Integración por Southern Blot.

RF20.4 Buscar Registro de Determinación del Patrón de Integración por Southern Blot.

RF20.5 Visualizar Registro de Determinación del Patrón de Integración por Southern Blot.

RF20.6 Imprimir registro de Determinación del Patrón de Integración por Southern Blot.

CU Gestionar Registro de Preparación de Sondas (SIC-0179).

RF21.1 Crear nuevo Registro de Preparación de Sondas.

RF21.2 Modificar datos del Registro de Preparación de Sondas.

RF21.3 Registrar traza del Registro de Preparación de Sondas.

RF21.4 Buscar Registro de Preparación de Sondas.

RF21.5 Visualizar Registro de Preparación de Sondas.

RF21.6 Imprimir Registro de Preparación de Sondas.

CU Gestionar Registro de Preparación de Sondas de ADN Cromosomal (SIC-0785).

RF22.1 Crear nuevo Registro de Preparación de Sondas de ADN Cromosomal.

RF22.2 Modificar datos del Registro de Preparación de Sondas de ADN Cromosomal.

RF22.3 Registrar traza del Registro de Preparación de Sondas de ADN Cromosomal.

RF22.4 Buscar Registro de Preparación de Sondas de ADN Cromosomal.

RF22.5 Visualizar Registro de Preparación de Sondas de ADN Cromosomal.

RF22.6 Imprimir Registro de Preparación de Sondas de ADN Cromosomal.

CU Gestionar Registro de Cuantificación de ADN de levadura por DOT-BLOT (SIC-0114).

RF23.1 Crear nuevo Registro de Cuantificación de ADN de levadura por DOT-BLOT.

RF23.2 Modificar datos del Registro de Cuantificación de ADN de levadura por DOT-BLOT.

RF23.3 Registrar traza del Registro de Cuantificación de ADN de levadura por DOT-BLOT.

RF23.4 Buscar Registro de Cuantificación de ADN de levadura por DOT-BLOT.

RF23.5 Visualizar Registro de Cuantificación de ADN de levadura por DOT-BLOT.

RF23.6 Imprimir Registro de Cuantificación de ADN de levadura por DOT-BLOT.

CU Gestionar Registro de Cuantificación de ADN de ratón por DOT-BLOT (SIC-0153).
RF24.1 Crear nuevo Registro de Cuantificación de ADN de ratón por DOT-BLOT.
RF24.2 Modificar datos del Registro de Cuantificación de ADN de ratón por DOT-BLOT.
RF24.3 Registrar traza del Registro de Cuantificación de ADN de ratón por DOT-BLOT.
RF24.4 Buscar Registro de Cuantificación de ADN de ratón por DOT-BLOT.
RF24.5 Visualizar Registro de Cuantificación de ADN de ratón por DOT-BLOT.
RF24.6 Imprimir Registro de Cuantificación de ADN de ratón por DOT-BLOT.

CU Gestionar Registro de Cuantificación de ADN de *E.coli* por DOT-BLOT (SIC-0154).
RF25.1 Crear nuevo Registro de Cuantificación de ADN de *E.coli* por DOT-BLOT.
RF25.2 Modificar datos del Registro de Cuantificación de ADN de *E.coli* por DOT-BLOT.
RF25.3 Registrar traza del Registro de Cuantificación de ADN de *E.coli* por DOT-BLOT.
RF25.4 Buscar Registro de Cuantificación de ADN de *E.coli* por DOT-BLOT.
RF25.5 Visualizar Registro de Cuantificación de ADN de *E.coli* por DOT-BLOT.
RF25.6 Imprimir Registro de Cuantificación de ADN de *E.coli* por DOT-BLOT.

CU Gestionar Registro de Datos Primarios para DOT-BLOT (SIC-0184).
RF26.1 Crear nuevo Registro de Datos Primarios para DOT-BLOT**.**
RF26.2 Modificar datos del Registro de Datos Primarios para DOT-BLOT.
RF26.3 Registrar traza del Registro de Datos Primarios para DOT-BLOT.
RF26.4 Buscar Registro de Datos Primarios para DOT-BLOT.
RF26.5 Visualizar Registro de Datos Primarios para DOT-BLOT.
RF26.6 Imprimir Registro de Datos Primarios para DOT-BLOT.

CU Gestionar Registro de Cuantificación de ADN de *CHO* por DOT-BLOT (SIC-0732).
RF27.1 Crear nuevo Registro de Cuantificación de ADN de *CHO* por DOT-BLOT.
RF27.2 Modificar datos del Registro de Cuantificación de ADN de *CHO* por DOT-BLOT.
RF27.3 Registrar traza del Registro de Cuantificación de ADN de *CHO* por DOT-BLOT.
RF27.4 Buscar Registro de Cuantificación de ADN de *CHO* por DOT-BLOT.
RF27.5 Visualizar Registro de Cuantificación de ADN de *CHO* por DOT-BLOT.
RF27.6 Imprimir Registro de Cuantificación de ADN de *CHO* por DOT-BLOT.

CU Gestionar Registro de Cuantificación de ADN de *LE392* por DOT-BLOT (SIC-0786).
RF28.1 Crear nuevo Registro de Cuantificación de ADN de *LE392* por DOT-BLOT.
RF28.2 Modificar datos del Registro de Cuantificación de ADN de *LE392* por DOT-BLOT.

RF28.3 Registrar traza del Registro de Cuantificación de ADN de *LE392* por DOT-BLOT.
RF28.4 Buscar Registro de Cuantificación de ADN de *LE392* por DOT-BLOT.
RF28.5 Visualizar Registro de Cuantificación de ADN de *LE392* por DOT-BLOT.
RF28.6 Imprimir Registro de Cuantificación de ADN de *LE392* por DOT-BLOT.

CU Gestionar Registro de Cuantificación de ADN de Tabaco por DOT-BLOT (SIC-0788).
RF29.1 Crear nuevo Registro de Cuantificación de ADN de Tabaco por DOT-BLOT.
RF29.2 Modificar datos del Registro de Cuantificación de ADN de Tabaco por DOT-BLOT.
RF29.3 Registrar traza del Registro de Cuantificación de ADN de Tabaco por DOT-BLOT.
RF29.4 Buscar Registro de Cuantificación de ADN de Tabaco por DOT-BLOT.
RF29.5 Visualizar Registro de Cuantificación de ADN de Tabaco por DOT-BLOT.
RF29.6 Imprimir Registro de Cuantificación de ADN de Tabaco por DOT-BLOT.

CU Gestionar Registro de Cuantificación de ADN de E. Coli de GC366 por DOT-BLOT (SIC-0789).
RF30.1 Crear nuevo Registro de Cuantificación de ADN de E. Coli de GC366 por DOT-BLOT.
RF30.2 Modificar datos del Registro de Cuantificación de ADN de E. Coli de GC366 por DOT-BLOT.
RF30.3 Registrar traza del Registro de Cuantificación de ADN de E. Coli de GC366 por DOT-BLOT.
RF30.4 Buscar Registro de Cuantificación de ADN de E. Coli de GC366 por DOT-BLOT.
RF30.5 Visualizar Registro de Cuantificación de ADN de E. Coli de GC366 por DOT-BLOT.
RF30.6 Imprimir Registro de Cuantificación de ADN de E. Coli de GC366 por DOT-BLOT.

CU Gestionar Registro de Cuantificación de ADN de *S. cereviseae* por DOT-BLOT (SIC-0789A).
RF31.1 Crear nuevo Registro de Cuantificación de ADN de *S. cereviseae* por DOT-BLOT.
RF31.2 Modificar datos del Registro de Cuantificación de ADN de *S. cereviseae* por DOT-BLOT.
RF31.3 Registrar traza del Registro de Cuantificación de ADN de *S. cereviseae* por DOT-BLOT.
RF31.4 Buscar Registro de Cuantificación de ADN de *S. cereviseae* por DOT-BLOT.
RF31.5 Visualizar Registro de Cuantificación de ADN de *S. cereviseae* por DOT-BLOT.
RF31.4 Imprimir Registro de Cuantificación de ADN de *S. cereviseae* por DOT-BLOT.

2.2 Requisitos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Estas propiedades reflejan las características que hacen al producto atractivo, usable, rápido y 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 a los Laboratorios/Grupos una vez semanalmente, en el primer mes de instalada la aplicación, y luego visitará 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, se dispondrá la primera semana para realizar todas las pruebas necesarias, según el diseño de las pruebas para probar la funcionalidad completa del sistema.

- Portabilidad

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

- Seguridad

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

- Políticos-culturales

El idioma que se empleará en la aplicación será el español. El sistema tendrá logotipos e imágenes en correspondencia con el carácter científico y profesional del CIGB. Algún cambio que se desee realizar

en la aplicación, será tramitado por la dirección del proyecto por parte del CIGB y canalizado por los directivos de producción de la UCI.

- Legales

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

- 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 es Windows XP o cualquier distribución de Linux. El servidor de Base de datos debe tener instalado Postgree SQL 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 8.2.0, así como Symfony como framework de desarrollo. El sistema, tendrá los colores correspondientes al logo del CIGB. Las páginas de la aplicación no se cargarán con mucha información y contendrán solo las imágenes necesarias que respondan a las pautas de diseño definidas por la Dirección de Diseño de la Universidad para la línea INFORMÁTICA MÉDICA.

2.3 Arquitectura del sistema

Uno de los artefactos que propone realizar RUP es el Documento de arquitectura de Software, donde se proporciona una descripción entendible de la arquitectura del sistema de software que sirve como medio de comunicación entre el arquitecto de software y otros miembros de equipo del proyecto, con respecto a las decisiones arquitectónicamente significativas que se han tomado en el proyecto. Contiene varias vistas, que muestran aspectos distintos del sistema: Vista de Casos de Uso, Vista lógica, Vista de Implementación, Vista de Procesos y Vista de Despliegue.

Vista de despliegue

La vista de Despliegue muestra la arquitectura física del sistema. Representa la distribución e instalación de las partes de un sistema, es decir los nodos que forman parte de la arquitectura sobre la que se ejecuta el sistema a través de sus componentes. Se construye a partir del diagrama de despliegue que se obtiene en el flujo de trabajo de diseño y la representación en cada uno de los nodos procesadores o nodos dispositivos los componentes del sistema. Los nodos procesadores son aquellos nodos con capacidad de procesamiento que pueden ejecutar un componente y los nodos dispositivos son aquellos nodos sin capacidad de procesamiento.

Los componentes son los elementos que participan en la ejecución de un sistema. Los nodos son los elementos donde se ejecutan los componentes. Los componentes representan el empaquetamiento físico de los elementos lógicos. Los nodos representan el despliegue físico de los componentes. (13)

A continuación se representa la Vista de Despliegue para el grupo de Biología Molecular. En esta vista se reflejan 4 nodos principales. El nodo procesador *PC Cliente*, el nodo procesador *Servidor de Aplicaciones Web*, el nodo procesador *Servidor de BD* y el nodo dispositivo *Impresora*.

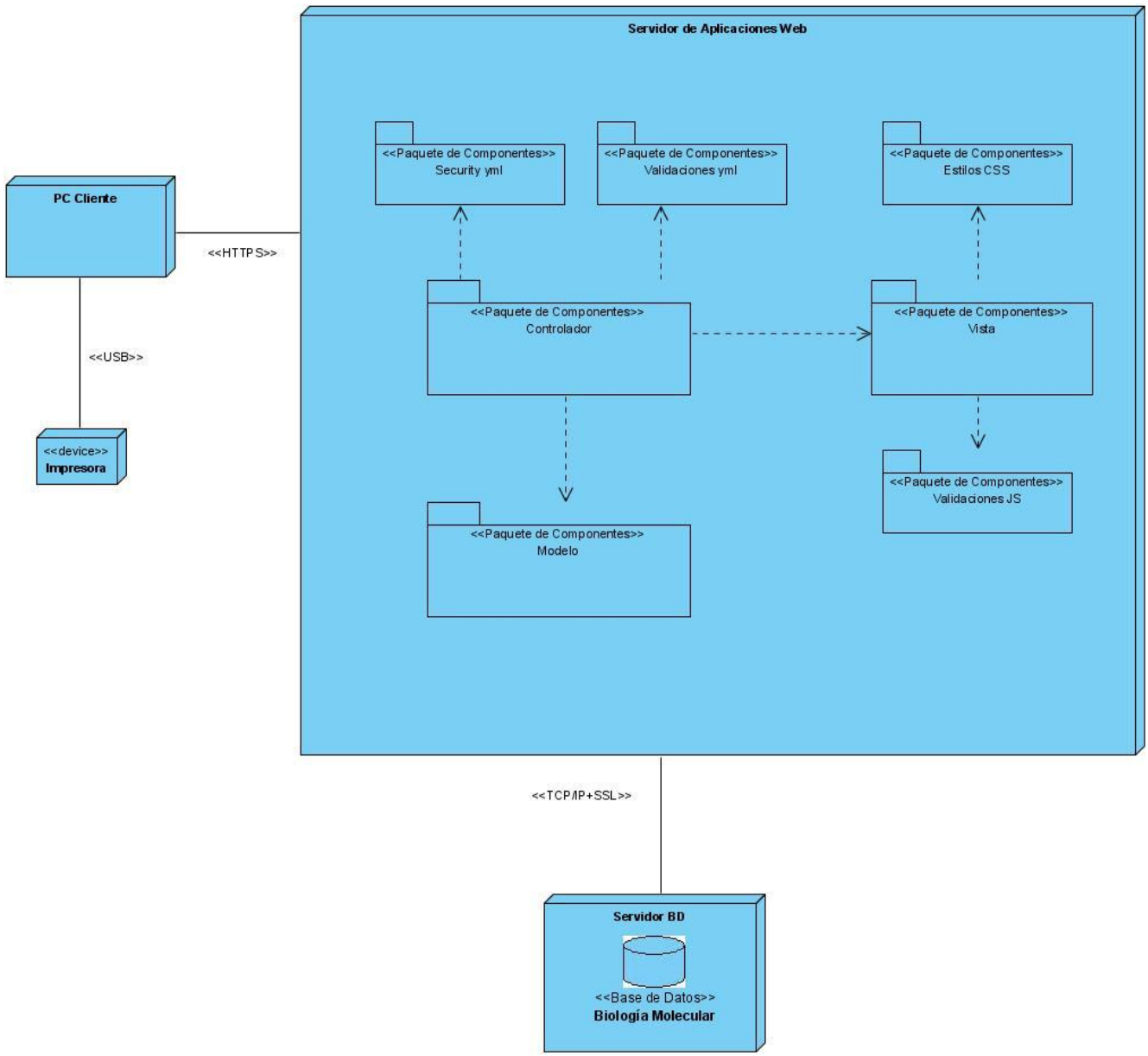


Figura 1: Vista de Despliegue Grupo de Biología

La *PC Cliente* estará conectada al nodo dispositivo *Impresora* mediante el protocolo de comunicación Universal Serial Bus *USB* y 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 paquete vista se relacionan los paquetes de componentes *Validaciones JS,* que agrupa los componentes para las validaciones javaScript y *Estilos CSS que agrupa los componentes* que definen el diseño del entorno visual de la aplicación.

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 este 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 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 (14).

Patrón de Arquitectura.

Cuando se aplica el patrón de arquitectura Modelo Vista Controlador, las vistas y los controladores conforman la interfaz de usuario y se asegura la consistencia entre la interfaz y el modelo. Al estar separados el modelo de los componentes Vista y del Controlador se pueden tener múltiples vistas del mismo modelo; por lo que si se cambia el modelo de una vista a través del controlador, todas las otras vistas dependientes de ese modelo deberán reflejar los cambios y las vistas recuperan los nuevos datos del modelo, actualizando la información que muestran al usuario. Además garantiza un menor acoplamiento al desacoplar las vistas de los modelos y los modelos de la forma en que se muestran e

ingresan los datos, y garantiza una mayor cohesión al estar altamente especializado en su tarea: la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio.
El uso de este patrón garantiza fundamentalmente:

- Poder crear múltiples vistas de un modelo.
- Poder crear, añadir, modificar y eliminar nuevas vistas dinámicamente.
- Cambiar el modo en que una vista responde al usuario sin cambiar su representación visual.
- Sincronizar las vistas.
- Más claridad de diseño.
- Facilitar el mantenimiento.
- Mayor escalabilidad.

Cuando se separa la vista, el controlador y el modelo el diseño resulta mucho más fácil, la única tarea del controlador sería obtener los datos del modelo y pasarlo a la vista. El modelo a su vez solo se encargaría del acceso a los datos, por lo que las funciones del modelo pueden reutilizarse desde diferentes controladores.

Symfony implementa el MVC de la siguiente forma:
En la Capa del modelo tiene en cuenta la abstracción a la base de datos y el acceso a los datos, garantizando la indepencia de la base de datos utilizada. Las clases del modelo son generadas automáticamente por Symfony y quien se encarga de generarlas es la librería *Propel.*
La abstracción de la base de datos es completamente invisible al programador, ya que la realiza otro componente específico llamado Creole. Así, si se cambia el sistema gestor de bases de datos en cualquier momento, no se debe reescribir ni una línea de código, ya que tan sólo es necesario modificar un parámetro en un archivo de configuración. (15)
En la capa vista se separa el código del elemento común de todas las interfaces en un *layout* y una plantilla. El layout es global en toda la aplicación y la plantilla es lo que cambia entre una interfaz y otra al visualizar las distintas variables definidas en el controlador.
En la capa del controlador Symfony define un controlador frontal, único para cada aplicación, encargado de cargar la configuración y determinar la acción a ejecutarse; y se definen además acciones que incluyen el código específico de cada página.

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.(16)

El patrón MVC puede contener conjunto de patrones de diseño asociados implícitamente:

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.(17)
En la figura se muestra un ejemplo de código que refleja el uso de este patrón, donde mediante la llamada a sfContext´::getInstance()->*getModuleName()* se obtiene el nombre del módulo que se ejecuta en ese momento.

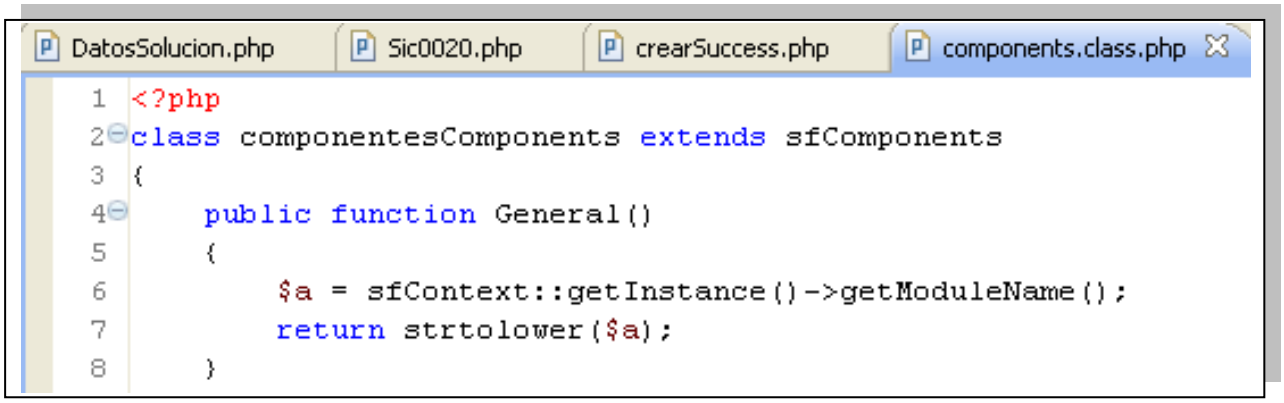


Figura 2 Ejemplo del patrón GoF Singleton

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.

La siguiente figura muestra el layout de la aplicación contiene un menú desplegable donde se reflejan las principales técnicas que se realizan al grupo de Biología Molecular.

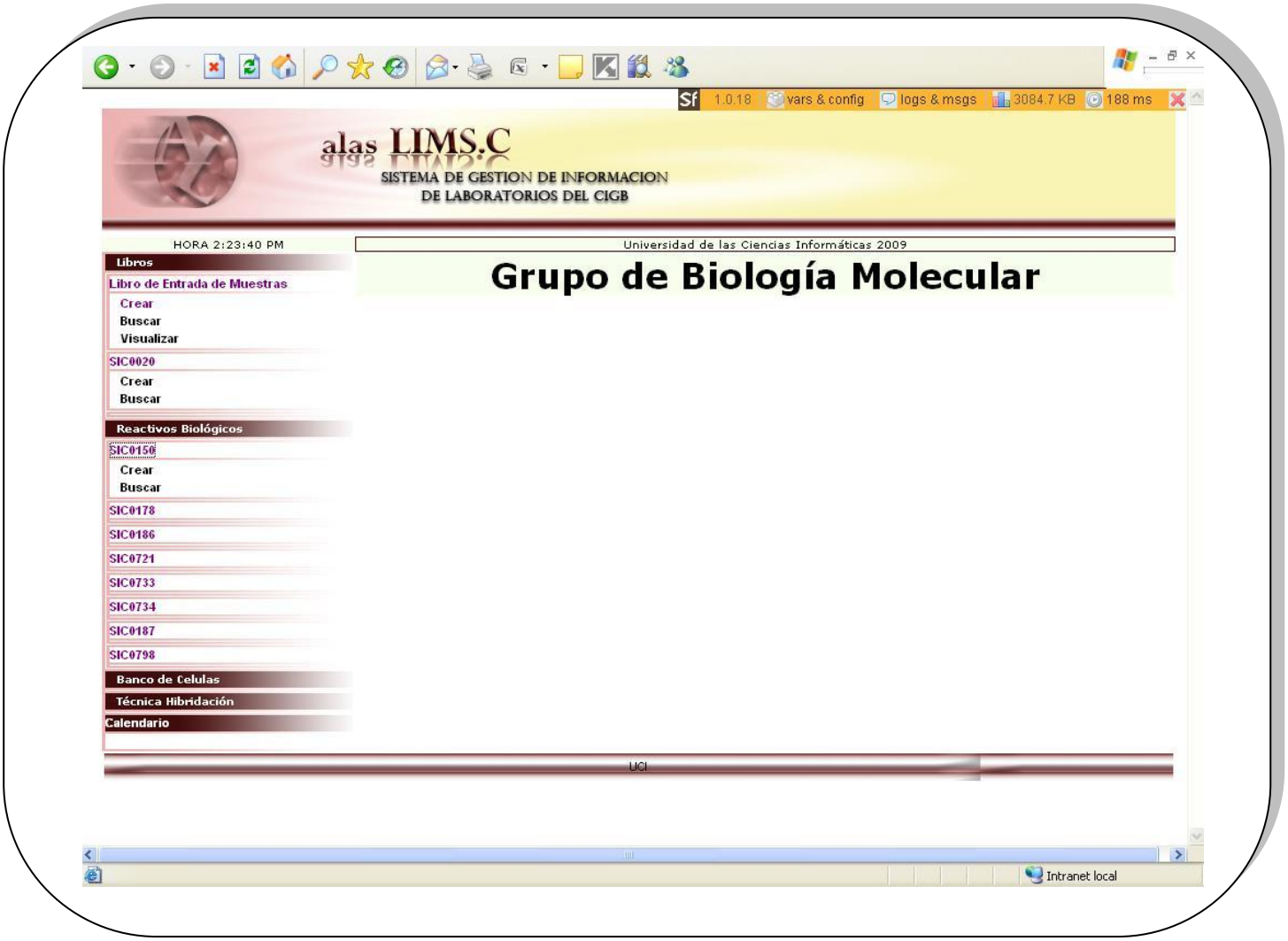


Figura 3 Layout de la aplicación

Patrones GRASP

- Creador: El patrón creador identifica el responsable de la instanciación de objetos o clases.

Symfony implementa este patrón al establecer en la clase Acción (Actions), las acciones definidas para cada módulo correspondiente y el módulo a ejecutar en cada una de ellas, evidenciando de este modo que la clase Actions es "creador" de dichos módulos. La clase Actions tiene la información necesaria para realizar la creación de un objeto.

- 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.

2.4 Diagramas de Componentes.

Un componente se define como una parte modular de un sistema que encapsula implementación y es desplegable y reemplazable.
Los diagramas de componentes se utilizan para mostrar la estructura de alto nivel del modelo de implementación, detallando los subsistemas de implementación existentes y sus dependencias en el momento de importar código y como organizarlos en capas.
En la elaboración de un diagrama de componentes se tiene en cuenta la reutilización y las restricciones impuestas por las herramientas utilizadas en el desarrollo de la aplicación.
La siguiente figura muestra el diagrama de componentes para el caso de uso Gestionar Extracción de ADN de alto peso de *Bacteria* (sic-0150), el cual se representa teniendo en cuenta el patrón de arquitectura que implementa Symfony Modelo Vista Controlador (MVC).

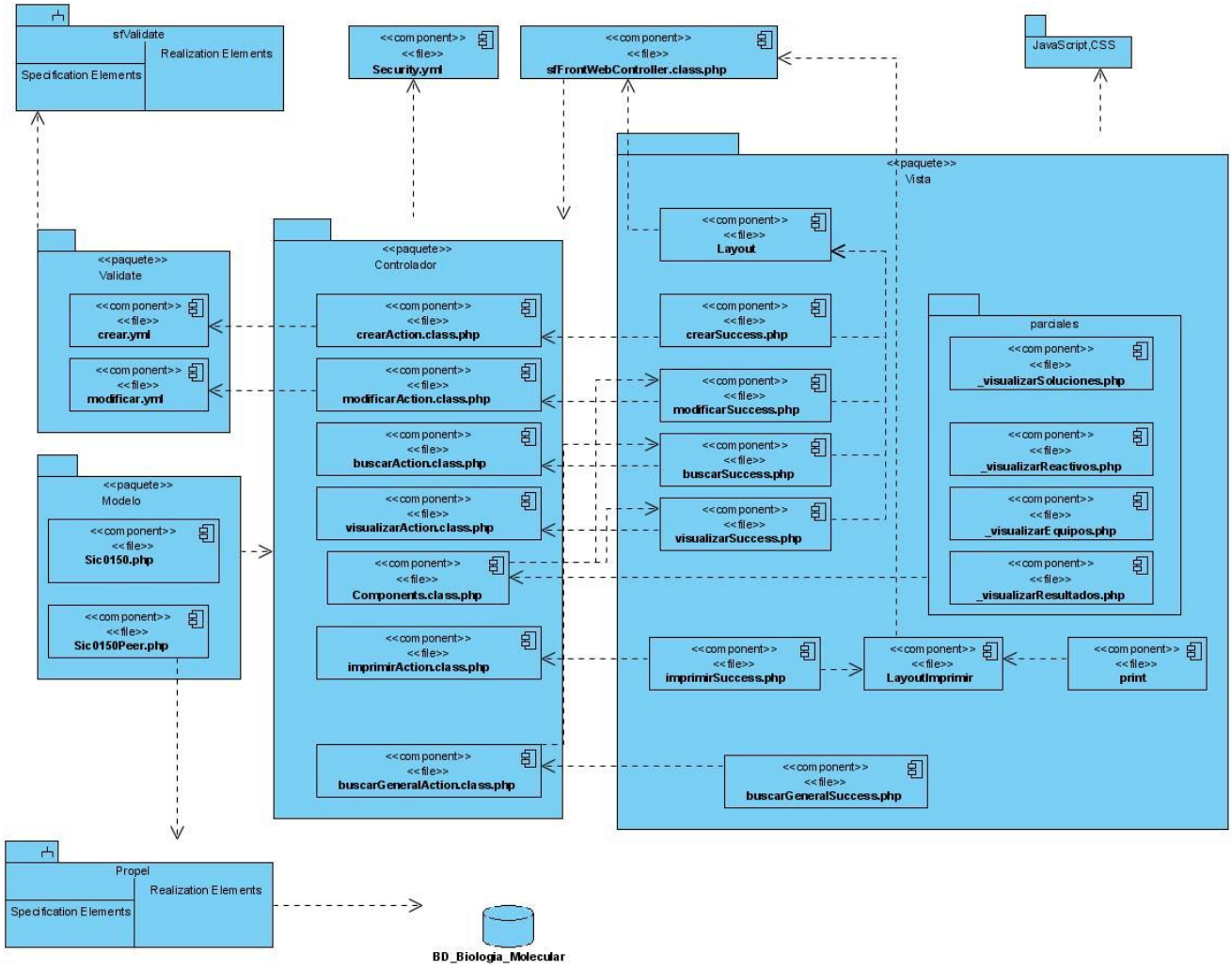


Figura 4 Diagrama de componentes Gestionar Extracción de ADN de alto peso de *Bacteria* (SIC0150)

Al aplicar este patrón al diseño del diagrama, los componentes quedan agrupados en tres paquetes fundamentales:

1- El paquete de componentes "*Modelo"*, agrupa los componentes que contienen los datos de las clases persistentes.

2- El paquete de componentes "*Vista",* agrupa los componentes que permiten la interacción del usuario con el sistema.
3- El paquete de componentes "*Controlador",* agrupa los componentes que comprenden la lógica del negocio, donde se agrupan los servicios que son compartidos en la aplicación.

El componente que permite que el usuario inicie su interacción con la aplicación, conocido como el punto de entrada a la aplicación es el *"sfWebController.class.php"* componente implementado por Symfony que se encarga de decodificar la petición y transferirla a la acción correspondiente, es decir qué módulo-acción se ejecutará.

Los componentes del paquete controlador, ficheros "*Action.class.php"* o "*components.class.php"* garantizan el cumplimiento de las funcionalidades del sistema definidas en los requisitos funcionales: crear, buscar, visualizar, modificar, imprimir. Para restringir el acceso a estas acciones se utiliza un componente "*security.yml",* garantizando así que el usuario se autentique obligatoriamente antes de acceder a la aplicación. Una vez autenticado se comprueba que el usuario tenga los privilegios necesarios para ejecutar determinada acción.

Cada uno de estos componentes ejecuta los componentes del paquete vista (ficheros *Success.php*) que muestran los resultados de sus acciones y que están contenidas en una plantilla global para toda la aplicación (*layout*).

En el paquete vista se muestra un componente *layout_imprimir* que tiene como objetivo sustituir al layout principal por otro que junto al componente *print* elimine la decoración y los botones para la impresión.

Este paquete contiene además un paquete "*parciales"* donde se encuentran agrupados los elementos parciales que responden a las acciones definidas en el componente "*components"* del paquete controlador y que tiene como objetivo reutilizar el código común en la implementación del sistema.

El paquete vista está relacionado con un paquete de componentes "*javaScript/CSS"* que utilizan cada uno de los componentes definidos en la vista y que contiene archivos *.css* para las hojas de estilo en cascada y archivos *.js* las funciones javaScript ubicados en la carpeta web del proyecto y que son aplicados a los elementos que serán mostrados al usuario, con el objetivo de validar la entrada de datos y hacer más agradable la interacción entre el usuario y la aplicación. Symfony ofrece un mecanismo para validar las acciones y la entrada de datos la base de datos mediante los archivos *.yml*

agrupados en el paquete de componentes "*validate"* que utiliza el subsistema "*sfValidate"* que contiene las funciones que implementa Symfony para la validación.
En el paquete de componentes "*Modelo"* se agrupan los componentes que contienen las clases construidas por el subsistema *propel* de Symfony (*sic0150.php* y *sic0150Peer.php*) para el acceso a datos que permiten a la capa del "mapeo de objetos a bases de datos" (ORM, de sus siglas en inglés "object-relational mapping") acceder a la base de datos Biología Molecular.

Además en el modelo se encuentran los componentes *bases* y *bases peer* por cada módulo creado en la aplicación que genera Symfony, que constituyen las clases bases del modelo de datos.
Symfony genera un conjunto de ficheros a los cuales se le hicieron modificaciones para el desarrollo de la aplicación:

- ✓ *security.yml:* Se define si está activada la seguridad de la aplicación y se pueden especificar los requerimientos de seguridad que los usuarios deberán satisfacer para cada acción o para todas las acciones.
- ✓ *view.yml*: Establece la estructura inicial de la vista por defecto: el nombre del layout, el título de la página, las hojas de estilos y los archivos JavaScript que se incluyen.
- ✓ *database.yml*: indica el nombre de la bases de datos, los datos de acceso. Se definen los parámetros para la conexión a la bases de datos.
- ✓ *schema.yml y propel.ini*: son los archivos de configuración que utiliza Propel para el acceso a los datos. Se utilizan para que las librerías de Propel puedan interactuar con las clases de Symfony y con los datos de la aplicación.
- ✓ *schema.yml*: contiene la representación del modelo de datos relacional del proyecto, se crea mediante el comando *Propel: build-schema.*
- ✓ *propel.ini*: Se genera de forma automática. Se define la conexión a la base de datos.

2.5 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 Biología Molecular 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 ó 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.

- 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 y el autor la persona, por ejemplo:

```
/**

* SIC0229 actions.
*
* @package    LIMS
* @subpackage SIC0229
* @author     INDIS
*/
```

Figura 5 Ejemplo de estándar de Codificación: SIC0229

2.6 Componentes reutilizables.

Muchas veces se hace necesario reutilizar código en el desarrollado de una aplicación, optimizando el tiempo y evitando la redundancia en la implementación. Symfony define alternativas para incluir un mismo código en varias páginas y manejar de forma inteligente estos fragmentos de códigos, independientes de la plantilla que lo utilice.
Si el fragmento contiene poca lógica, se puede utilizar un archivo de plantilla al que se le pueden pasar variables. Este archivo de planilla se define y utiliza como un elemento parcial (*partial*). En cambio si se debe acceder a los datos del modelo o variar los contenidos en función de la sesión, es decir que la lógica sea compleja, es aconsejable separar la presentación de la lógica utilizando componentes (*component*).

Parciales.
Un elemento parcial es un fragmento de código de plantilla que se reutiliza y que al igual que las plantillas son archivos encontrados en el directorio *templates/,* y que contienen código HTML y PHP. Estos parciales siempre comienzan con un guión bajo (_), para que puedan ser identificados entre las

plantillas que se encuentran en el mismo directorio. Para incluirlos en una plantilla se utiliza el helper *include_partial()* al que se le especifica en los parámetros el nombre del parcial a utilizar y el módulo donde está definido.

Componentes.

Un componente se comporta 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 *action/components.class.php.* Su presentación se guarda en un elemento parcial. Los métodos de la clase *sfComponents* empiezan con la palabra execute, como sucede con las acciones, y pueden pasar variables de igual forma en la que se pasan variables en las acciones. 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 (*_miComponente.php*).

Al igual que sucede con los elementos parciales, se pueden pasar parámetros adicionales a los componentes mediante un array asociativo. Dentro del elemento parcial se puede acceder directamente a los parámetros mediante su nombre y en el componente se puede acceder a ellos mediante el uso de *$this.*

2.7 Representación del código principal.

Una de las actividades del grupo de Biología Molecular es aplicar varias técnicas a diferentes reactivos biológicos. La figura a continuación muestra los campos que tienen en común estas técnicas que se le realizan a los distintos reactivos biológicos, donde se les introducen las diferentes concentraciones y se calculan a partir de ellas los contaminantes.

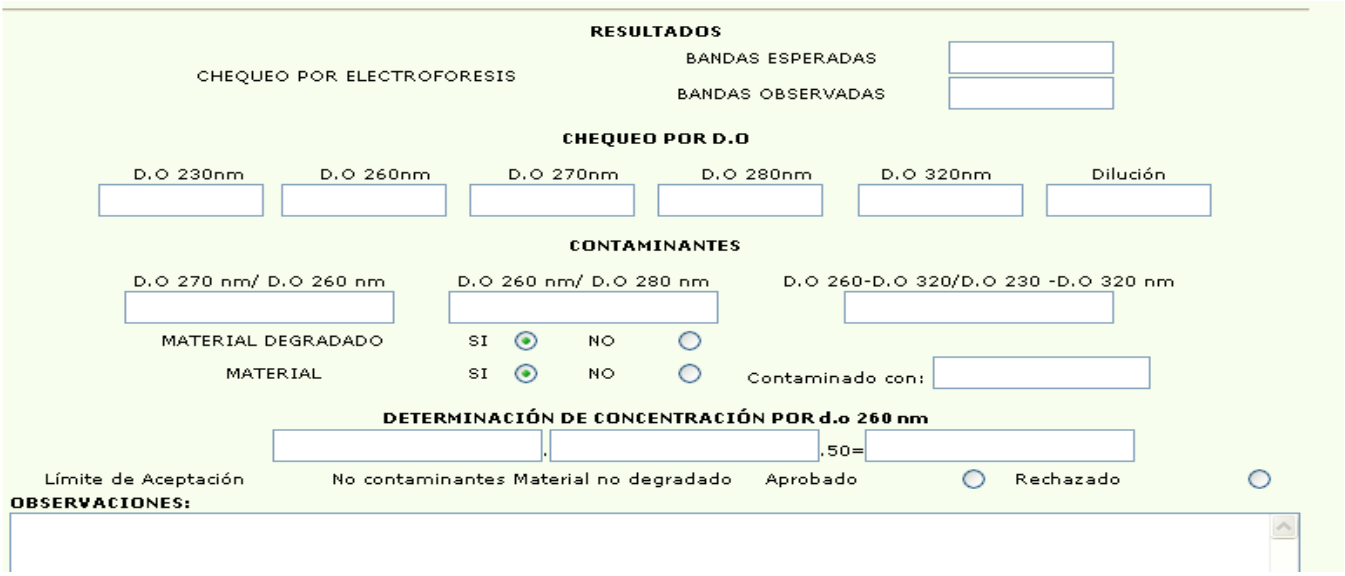


Figura 6 Parcial Resultados. Reactivos Biológicos

Visualmente todas estas técnicas muestran los resultados del cálculo a medida que se introducen los valores y actualizan los campos que dependen de ellos. Para dar solución a estas restricciones se aprovecharon alguna de las facilidades que Symfony ofrece; se creó un parcial *_resultados.php* que permite reutilizar el código del elemento visual y que es simplemente incluido en las plantillas, y se implementa en el parcial un helper de Symfony llamado *observe_field()* que lanza una petición Ajax que actualiza un valor de un campo en la medida que otro es observado, es decir que el usuario realiza alguna acción sobre él.

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones son ejecutadas en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano, por lo que resulta posible realizar cambios sobre las páginas sin necesidad de recargarlas, aumentando 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. 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 como el observe_field, los formularios más modernos que 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.
A continuación se muestran algunos fragmentos de código fuente del parcial creado donde se utiliza el *observe_field()* para calcular el campo de la concentración por 260nm.

Código Fuente del parcial "_resultado-php".

```
<table width="640" border="0" align="center" cellpadding="0" cellspacing="0"
bordercolor="#000000">
   <tr><td colspan="2" align="center"> </td></tr>
   <tr><td colspan="2" align="center"><strong>DETERMINACI&Oacute;N DE
 CONCENTRACI&Oacute;N POR d.o 260 nm</strong></td></tr>
   <tr><td colspan="2" align="center"><div></div></td></tr>
   <tr><td><div align="right"> <?php echo input_tag('cal1','')?>.<?php echo
 input_tag('cal2','')?>.50=</div></td>
       <td><div id="r4"><?php echo input_tag('result4','') ?></div></td>
  </tr>
</table>
<table width="640" border="0" align="center" cellpadding="0" cellspacing="0"
bordercolor="#000000">
  <tr><td width="155" align="center"><div>L&iacute;mite de Aceptaci&oacute;n
 </div></td>
      <td width="267" align="center"><div>No contaminantes Material no
 degradado</div></td>
      <td width="81">Aprobado</td>
      <td width="18"><strong><?php echo
 radiobutton_tag('aceptado',true,false)?></strong></td>
      <td width="100"><div>Rechazado</div></td>
      <td width="19"><strong><?php echo
 radiobutton_tag('aceptado',false,false)?></strong></td>
  </tr>
  <tr>
      <td colspan="6"><div><strong> OBSERVACIONES:</strong></div>
      <?php  echo form_error('observaciones') ?>
      <?php echo textarea_tag('observaciones', '', 'size=90x10') ?></strong></td>
 </tr>
 </table>
 <?php echo observe_field('DO_260', array(

             'update' =>'r4',

             'url' =>'sic0187/calculoN',

             'script' =>true,

             'with' => "'DO_260=' + value +'&dilucion=' + $('dilucion').value",))

 ?>
 <?php echo observe_field('dilucion', array(

             'update' =>'r4',

             'url' =>'sic0187/calculoN',

             'script' =>true,

             'with' => "'dilucion=' + value +'&DO_260=' + $('DO_260').value",))

 ?>
```

Una vez que se registren estas técnicas la aplicación brinda la posibilidad de buscarlas y visualizarlas, por lo que igual se necesita un parcial que visualice estos resultados; con la diferencia que los resultados a mostrar varían en dependencia de la técnica. El uso de un componente resuelve esta problemática; al separar la presentación de la lógica. Su presentación se guarda en un elemento parcial y la lógica es implementada en un archivo *action/components.class.php* como se explica en el epígrafe anterior.

Código Fuente de "sic0734/visualizarAction.class.php" .

```
public function execute()
  {

    if($this->getRequest()->getMethod()!= sfRequest::POST)
      {

         $this->sic=Sic0734Peer::retrieveByPK($this->getRequestParameter('estado'));

      $c=new Criteria();
      $c->add(Sic0734Peer::ID_0734,$this->getRequestParameter('estado'));
      $v=Sic0734Peer::doSelectOne($c);
      $this->sic0734=$v;

           $c=new Criteria();
           $c->add(Sic0734Peer::ID_0734,$this->getRequestParameter('estado'));
           $c->addJoin(Sic0734Peer::ID_0734,EquiposMedicionBmPeer::ID_0734);
           $v=EquiposMedicionBmPeer::doSelect($c);
           $this->equipo=$v;

          $c=new Criteria();
          $c->add(Sic0734Peer::ID_0734,$this->getRequestParameter('estado'));
          $c-
>addJoin(Sic0734Peer::ID_0734,MaterialesReactivosBiologicoPeer::ID_0734);
          $v=MaterialesReactivosBiologicoPeer::doSelect($c);
          $this->reactivos=$v;

          $c=new Criteria();
          $c->add(Sic0734Peer::ID_0734,$this->getRequestParameter('estado'));
          $c->addJoin(Sic0734Peer::ID_0734,Sic0020Peer::ID_0734);
          $v=Sic0020Peer::doSelect($c);
          $this->sol=$v;

       return sfView::SUCCESS;
     }

Else { }

}
```

Fragmentos de código fuente de "sic0734/visualizarSuccess.php"

```
<?php echo form_tag('sic0734/crear')?>

<table width="750" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
     <td width="854" height="750" bordercolor="#EFEFEF">

<table width="750" border="0" bordercolor="#000000" cellpadding="0" cellspacing="0">
  <tr>
    <td width="525" rowspan="3" align="center"><div align="center"><strong>CENTRO DE
INGENIER&Iacute;A GEN&Eacute;TICA Y BIOTECNOLOG&Iacute;A</strong></div>      <div
align="center"><strong>DIRECCI&Oacute;N DE CALIDAD PURIFICACI&Oacute;N DE
ADN</strong> </div>
<div align="center"><strong>PLASM&Iacute;DICO DE E.coli POR MINIALCALINO DE 50 mL
</strong></div></td>

   <td width="105" align="center"><div><strong>SIC-0734</strong></div></td>
   <td width="112"><div><p><strong>PPO 4.09.173.01</strong></p></div></td>
  </tr>

  <tr>
   <td align="center"><div><strong>Edici&oacute;n 02</strong><div></div></td>
  </tr>
  <tr>
   <td><div align="center"><strong>FOLIO</strong></div></td>
   <td><?php echo input_tag('folio',$sic0734->getFolio(),'size=10') ?></td>
  </tr>
</table><hr>

<?php
include_component('componentes','visualizarReactivos',array('Reactivos'=>$reactivos)
)?>
<?php include_component('componentes','visualizarEquipos',array('EIM'=>$equipo))?>
<?php include_component('componentes','visualizarSoluciones',array('sol'=>$sol)) ?>
<?php
include_component('componentes','visualizarResultados',array('resultado'=>$sic0734))
?>
<?php include_component('componentes','visualizarDatos',array('id'=>$sic0734)) ?>

<div align="center"><strong><?php echo 'Estado:';echo
checkbox_tag('estado',0,false)?></strong></div>

<div><?php echo button_to('Modificar','Sic0734/modificar'.'?estado='.$sic0734-
>getId0734())?><?php echo button_to('Cancelar','sic0734/buscar') ?></div>

</td>
</tr>
</table>
```

2.8 Análisis de la seguridad del sistema implementado.

Symfony avala la autenticación y la gestión de credenciales simplificando la gestión de la seguridad de usuario y la creación de secciones restringidas.

Seguridad de la Acción

La posibilidad de ejecutar una Acción (Actions) puede ser restringida a usuarios con ciertos privilegios. Symfony proporciona mecanismos para este propósito que 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. El estado autenticado se estable con el método setAuthenticated() y se puede comprobar con el método isAuthenticated().Se pueden extender los privilegios del usuario mediante llamadas a métodos del objeto sfUser.

Cada Acción antes de ser ejecutada 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. Symfony busca en el archivo security.yml por el nombre de la acción y si existe, verifica que se satisfagan los requerimientos de seguridad. Lo que sucede cuando un usuario trata de acceder a una acción restringida depende de sus credenciales:

- Si el usuario está autenticado y tiene las credenciales apropiadas, entonces la acción se ejecuta.
- Si el usuario no está autenticado, es redireccionado a la acción para autenticarse.
- Si el usuario está autenticado, pero no posee las credenciales apropiadas, será redirigido a la acción segura por defecto.

Sesiones de usuario

El manejo de sesiones de Symfony está basado en las sesiones de PHP; logrando desarrollar un mecanismo más configurable y más fácil de usar. Maneja automáticamente las sesiones del usuario y es capaz de almacenar datos de forma persistente entre peticiones.

Symfony dispone de una clase sfUser que trae asociado un conjunto de parámetros que permiten guardar los atributos de un usuario en él. El objeto sesión de cada usuario se accede en la acción con el método getUser(), que es una instancia de esta clase. Toda la información referente a cada usuario estará disponible en otras peticiones hasta terminar la sesión del usuario.

El manejo de sesiones de Symfony se encarga de gestionar automáticamente el almacenamiento de los identificadores de sesión tanto en el cliente como en el servidor. En el lado del servidor, Symfony guarda por defecto las sesiones de usuario en archivos, e igual permite almacenarlas en la base de datos. Las clases de almacenamiento de sesiones disponibles son sfMySQLSessionStorage, sfPDOSessionStorage y sfPostgreSQLSessionStorage; siendo la última la usada en la implementación del sistema.

La opción de la base de datos define la conexión a utilizar; luego utiliza databases.yml para determinar los parámetros de la conexión (host, nombre de la base de datos, usuario y contraseña) para realizar la conexión.

La expiración de la sesión se produce automáticamente después de sf_timeout segundos. El valor de esta constante es 30 minutos por defecto y puede ser modificado para cada entorno en el archivo de configuración settings.yml.

2.9 Pruebas.

Las pruebas del software constituyen un elemento crítico para la garantía de la calidad del software. Con el diseño de pruebas se descubren diferentes clases de errores y se asegura que los defectos encontrados sean corregidos. El proceso de prueba se enfoca en la lógica interna del software y las funciones externas. Es el proceso de ejecución de un programa con la intención de descubrir un error y se define como un buen caso de prueba si tiene una alta probabilidad de mostrar un error no descubierto hasta entonces. La prueba no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software.

Cuando se realizan pruebas para demostrar que cada función es completamente operativa, se hace referencia a las pruebas de Caja Negra. Sin embargo, cuando las pruebas aseguran que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada se hace referencia a las pruebas de Caja Blanca.

Las pruebas de Caja Blanca es un método de diseño de casos de prueba que garantiza que se ejerciten todos los caminos independientes de cada módulo, se ejerciten todas las decisiones lógicas, se ejecutan todos los bucles y se ejecutan las estructuras de datos internas.

La prueba de unidad es un tipo de prueba que utiliza el método de Caja Blanca. El objetivo de las pruebas de unidad o unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura, puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.

Framework Lime.

Symfony incluye el framework LIME para automatizar las pruebas; las pruebas automatizadas constituyen uno de los mayores avances en la programación desde la orientación a objetos.

Lime proporciona el soporte para las pruebas unitarias. Ejecuta los archivos de prueba en un entorno independiente para evitar conflictos entre las diferentes pruebas. Las pruebas de Lime y sus resultados son fáciles de leer. Consta únicamente de un archivo, llamado lime.php, y no tiene ninguna dependencia. Cada prueba unitaria consiste en una llamada a un método de la instancia de *lime_test.*

Las pruebas unitarias validan la forma en la que las funciones y métodos trabajan en cada caso particular. Son archivos PHP cuyo nombre termina en *test.php* y se encuentran en el directorio *test/unit* de la aplicación. Para ejecutar el conjunto de pruebas, se utiliza la tarea *test-unit* desde la línea de comandos. El resultado de esta tarea en la línea de comandos es muy explícito, lo que permite localizar fácilmente las pruebas que han fallado y las que se han ejecutado correctamente. Se crea un nuevo objeto lime_test y el número de pruebas a lanzar se pasa como argumento.

Todas las pruebas unitarias escritas con lime comienzan con el código:

require_once dirname(__FILE__).'/../bootstrap/unit.php';

Para la *Caracterización de Productos de ADN (sic0798)* se le realizan varias pruebas al método Calcular Promedio.

calcularPromedioTest.php

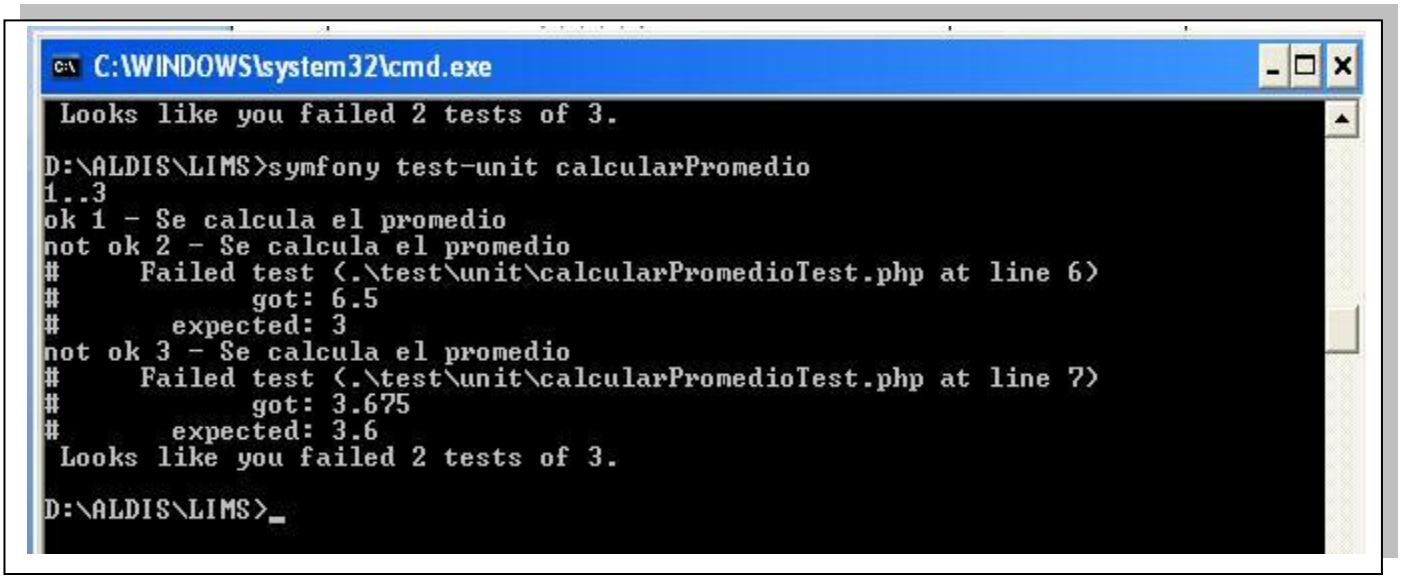
```
1 <?php
2 include(dirname(__FILE__).'/../bootstrap/unit.php');
3
4 $t = new lime_test ( 3, new lime_output_color ( ) );
5 $t->is ( ChequeoConcentracion::CalcularPromedio(array(1,2,3,4,5)), 3, 'Se calcula el promedio' );
6 $t->is ( ChequeoConcentracion::CalcularPromedio(array(6,7)), 3, 'Se calcula el promedio');
7 $t->is ( ChequeoConcentracion::CalcularPromedio(array(3.5,3.6,3.7,3.9)), 3.60, 'Se calcula el promedio');
8
```

Figura 7 Prueba unitaria al sic0798 CalcularPromedioTest

El método de prueba utilizado fue “is” que compara 2 valores y la prueba pasa si los 2 son iguales.

Datos de Entrada	Método de Prueba	Valor esperado	Valor Obtenido
CalcularPromedio(1,2,3,4,5)	IS	3	3
CalcularPromedio(6,7)	IS	3.5	3
CalcularPromedio(3.5,3.6,3.7,3.9)	IS	3.60	3.675

Figura 8 Resultados Prueba CalcularPromedio



Si se definen las suficientes pruebas unitarias como para probar la mayor parte de una aplicación es mucho más seguro refactorizar el código de la aplicación y añadir nuevas características. Las pruebas reducen el tiempo requerido para la documentación técnica del proyecto.

2.10 Conclusiones del capítulo

En este capítulo se analiza cómo los requisitos funcionales identificados para el grupo de Biología Molecular fueron implementados respondiendo eficientemente a la funcionalidad del sistema y a las restricciones de los requisitos no funcionales.

Se muestra bajo que estándares de codificación se realiza la implementación de acuerdo a lo planteado por la metodología RUP en el flujo de trabajo de implementación. En correspondencia con los patrones de diseño y arquitectura que define el framework de desarrollo Symfony fue guiada la

implementación; donde se evidencia además como se enriquece la misma a partir de las ventajas y elementos que ofrece Symfony, así cómo también se valida la seguridad del sistema y se realizan las pruebas unitarias para eliminar la mayor cantidad de errores posibles. De forma general en este capítulo se reflejan los resultados obtenidos en la implementación del sistema.

Conclusiones

El estudio de los resultados obtenidos en investigaciones anteriores permitió una mejor comprensión de los procesos y actividades que se realizan en el grupo de Biología Molecular y ayudaron en gran medida con la implementación del sistema. El estudio realizado sobre los diferentes LIMS desarrollados en el mundo permitió una mayor comprensión del funcionamiento y comportamiento de un LIMS lográndose identificar las funcionalidades para el desarrollo del sistema. Por lo que se concluye de la siguiente forma:

- Se lograron implementar los 31 casos de uso del grupo Biología Molecular.

- Se realizaron pruebas unitarias para eliminar posibles errores en el código y una mejor comprensión del mismo durante el desarrollo de la aplicación.

Con el desarrollo de esta investigación se alcanzó satisfactoriamente el objetivo propuesto, pues se logró la implementación del módulo Biología Molecular para el Sistema de Gestión de Información de los laboratorios de la dirección de calidad del CIGB, lo cual permitirá la mejora de los procesos que se realizan por los especialistas de dicho centro.

Recomendaciones

Al concluir este trabajo se recomienda:

- Integrar el grupo de Biología Molecular a los demás grupos del CIGB en futuras versiones.
- Seguir profundizando en el estudio de los LIMS a nivel mundial para lograr incorporar al sistema la información que muestran los diferentes equipos que se utilizan en el centro.

REFERENCIAS BIBLIOGRÁFICAS.

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

2. **Centro de Ingeniería Genética y Biotecnología**. *Centro de Ingeniería Genética y Biotecnología.* [Online] CIGB, noviembre 13, 2008. [Cited: noviembre 13, 2008.] http://www.cigb.edu.cu/index.php?option=com_content&task=view&id=73&Itemid=142

3. **Ministerio de la Informática y las Comunicaciones de Cuba**. *Ministerio de la Informática y las Comunicaciones de Cuba.* [En línea] MIC, 24 de febrero de 2009. [Citado el: 24 de febrero de 2009.] http://www.mic.gov.cu/hinfosoc.aspx. 3

4. **Quass**, *¿POR QUÉ NECESITO UN LIMS*?, [En Línea, Citado: 24de febreo de 2009], Disponible en: http://www.quaass.com/web/unlims_1.htm.

5. **Addlink Software Científico, S.L.** Matrix LIMS La herramienta de productividad para cualquier laboratorio. *Addlink Software Científico.* [En línea] [Citado el: 18 de diciembre de 2008.] http://www.addlink.es/productos.asp?pid=472.

6. **Eslavy Pedraza Gonález, Pedro Manuel Puig Díaz.** Sistema para la Gestión de la Información de los Laboratorios de la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología: Análisis y Diseño del módulo Biología Molecular. La Habana : s.n., 2008.

7. **F., Luis Farley Ortiz.** Campus Virtual con Software Libre. [Online] diciembre de 2006. [Citado em: 20 de mayo de 2009.] http://ecampuslibre.blogspot.com/.

8. **Hanz Cocchi Guerrero**, *Rol* [En Línea, Citado: 27/1/2008] disponible en: http://hanzcocchi.net/el-rol-del-analista-en-rup/].

9. **Ciberaula.** *Ciberaula.* [Online] Ciberaula, 2006. [Citado em: 9 de junio de 2009.] http://www.ciberaula.com/curso/php5/que_es/.

10- **Alcides Romero, Carlos Batista, Nacarit España de Romero .** [Online] noviembre de 2007. [Citado em: 9 de junio de 2009.] http://es.geocities.com/bati144/fase3/t4.html.

11. —. 2005. RedKsr.com. [Online] 2 de agosto de 2005. [Citado em: 9 de junio de 2009.] http://www.redksr.com/~talos/JAVA/JAVA3D/eclipse.pdf

12. **Oktaba, Hanna.** Unam Posgrado Ciencia e Ingeniería de la Computación. Unam Posgrado Ciencia e Ingeniería de la Computación. [Online][Cited: abril 25, 2009] Disponible en: http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.

13. Modelo de Implementación:Diagramas de Componentes y Despliegue. [Online] [Citado em: 8 de junio de 2009.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

14. **4D SAS.** Utilizar el protocolo SSL. 4D. [Online] 2003. [Citado em: 9 de junio de 2009.] http://www.4d.com/docs/CMS/CMS02064.HTM.

15. Potencier, Fabien y Zaninotto, François. Symfony, la guía definitiva. s.l. : Apress, 2008 pág33.

16. Potencier, Fabien y Zaninotto, François. Symfony, la guía definitiva. s.l. : Apress, 2008 pág29.

17. Potencier, Fabien y Zaninotto, François. Symfony, la guía definitiva. s.l. : Apress, 2008 pág35.

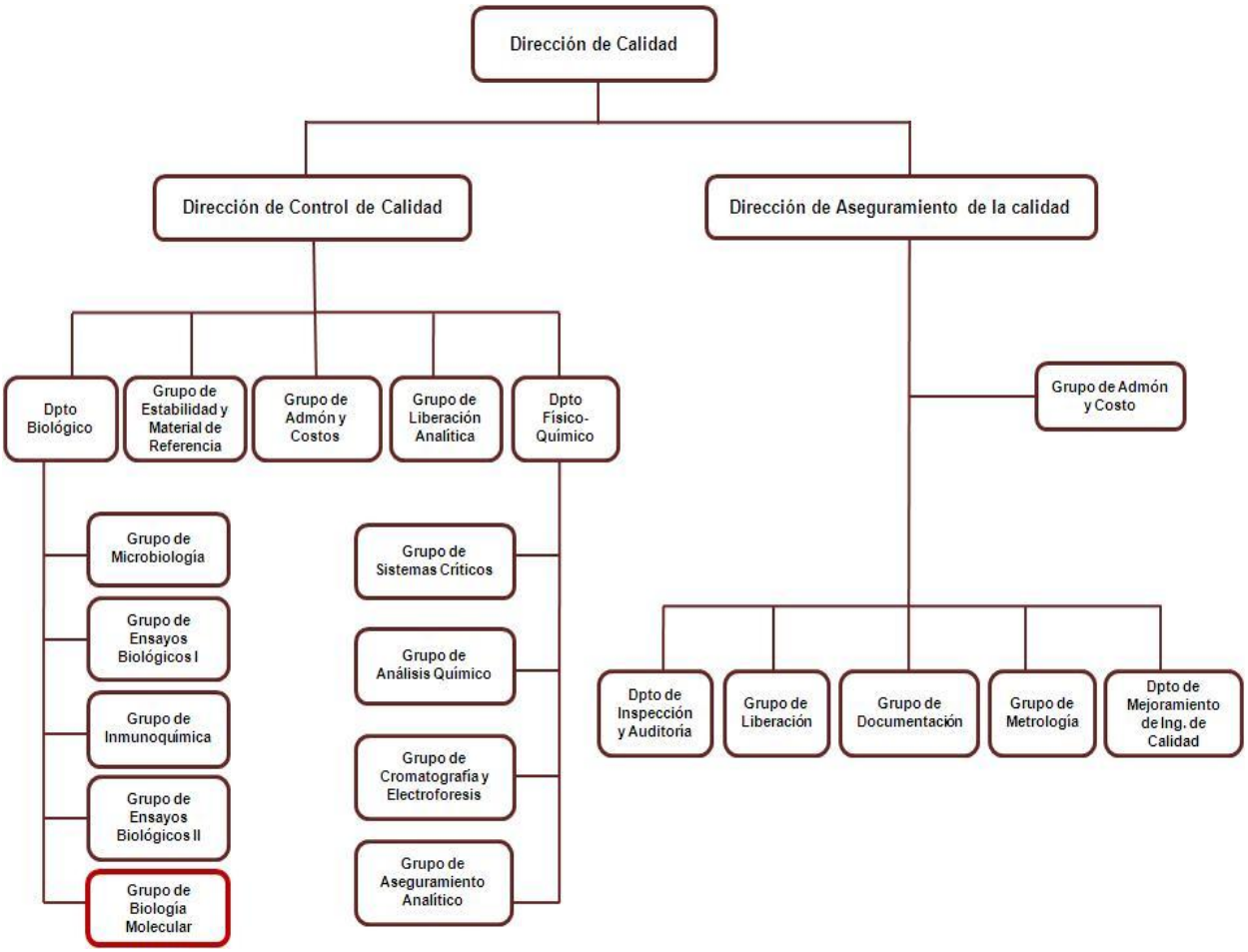
BIBLIOGRAFÍA.

- **Equipo Editor de Atina Chile**. *Atina Chile*. Atina Chile. [En línea] [Citado el: 24 de febrero de 2009.] http://www.atinachile.cl/node/4125.
- **Pressman, Roger S**. *Ingeniería de Software. Un enfoque práctico.* s.l: McGraw-Hill/Interamericana, 2002.
- **Larman, Graig.** *Uml y patrones*, introducción al análisis y diseño orientado a objetos. México : Prentice Hall, 1999. 970-17-0261-1.
- **LabWare Ltd.** Niederlassung Deutschland . Chemie.de.[En línea] LabWare Ltd. Niederlassung Deutschland . [Citado el: 24 de febrero de 2009.] http://www.chemie.de/products/es/52345/.
- **Oktaba, H.** (n.d.). *Unam Posgrado Ciencia e Ingeniería de la Computación*. Retrieved abril 25, 2009, from Unam Posgrado Ciencia e Ingeniería de la Computación:

  http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html
- **Oktaba, Hanna.** *Introducción a Patrones*. Notas de cursos. Facultad de Ciencias, UNAM. http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html, noviembre 2008.
- **Molpeceres, Alberto.** *JavaHispano*. JavaHispano. [Online] mayo 15, 2001. [Cited: abril 25, 2009.] http://www.javahispano.org/contenidos/es/patrones_de_software_parte_1/.
- **Welicki, León.** *Patrones y Antipatrones: una Introducción* - Parte II. Patrones y Antipatrones: una Introducción -Parte II. [Online] 2009. [Cited: abril 25, 2009.] http://msdn.microsoft.com/es-es/library/bb972251.aspx#M16.
- **Prieto, Félix.** Patrones de diseño. [Online] 2008. [Cited: abril 25, 2009.] http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf.
- **CIGB.** Centro de Ingeniería Genética y Biotecnología. Centro de Ingeniería Genética y Biotecnología. [Online] CIGB, noviembre 12, 2008. [Cited: noviembre 12, 2008.] http://www.cigb.edu.cu/index.php?option=com_content&task=view&id=55&Itemid=121.
- **Centro de Ingeniería Genética y Biotecnología.** Centro de Ingeniería Genética y Biotecnología. [Online] CIGB, noviembre 13, 2008. [Cited: noviembre 13, 2008.] http://www.cigb.edu.cu/index.php?option=com_content&task=view&id=73&Itemid=142.
- **Ministerio de la Informática y las Comunicaciones de Cuba**. Ministerio de la Informática y las Comunicaciones de Cuba. [En línea] MIC, 24 de febrero de 2009. [Citado el: 24 de febrero de 2009.] http://www.mic.gov.cu/hinfosoc.aspx. 3
- **Quass,** ¿POR QUÉ NECESITO UN LIMS?, [En Línea, Citado: 24de febrero de 2009], Disponible en: http://www.quaass.com/web/unlims_1.htm.

- **Addlink Software Científico, S.L.** Matrix LIMS La herramienta de productividad para cualquier laboratorio. *Addlink Software Científico.* [En línea] [Citado el: 18 de diciembre de 2008.] http://www.addlink.es/productos.asp?pid=472.
- **Hanz Cocchi Guerrero**, Rol [En Línea, Citado: 27/1/2008] disponible en: http://hanzcocchi.net/el-rol-del-analista-en-rup/].
- **Oktaba, Hanna.** Unam Posgrado Ciencia e Ingeniería de la Computación. Unam Posgrado Ciencia e Ingeniería de la Computación. [Online][Cited: abril 25, 2009] Disponible en: http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.
- **Potencier, Fabien y Zaninotto, François.** Symfony, la guía definitiva. s.l. : Apress, 2008.
- **BIKA Jabs System.** [En línea] 2005-2009. Disponible en:http://www.bikalabs.com/.
- **Roman Konertz,** PD Dr. Ludwig Eichinger and Dr. Budi Tunggal. OpenLIMS. [En línea] Disponible en: http://www.open-lims.org/.
- **Open source laboratory automation &informatics.** [En línea] Disponible en:http://www.labmatica.com/.
- **Eslavy Pedraza Gonález, Pedro Manuel Puig Díaz.** Sistema para la Gestión de la Información de los Laboratorios de la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología: Análisis y Diseño del módulo Biología Molecular. La Habana : s.n., 2008.
- **F., Luis Farley Ortiz.** Campus Virtual con Software Libre. [Online] diciembre de 2006. [Citado em: 20 de mayo de 2009.] http://ecampuslibre.blogspot.com/.
- **Ciberaula.** *Ciberaula.* [Online] Ciberaula, 2006. [Citado em: 9 de junio de 2009.] http://www.ciberaula.com/curso/php5/que_es/.
- **Alcides Romero, Carlos Batista, Nacarit España de Romero .** [Online] noviembre de 2007. [Citado em: 9 de junio de 2009.] http://es.geocities.com/bati144/fase3/t4.html.
- **Puebla, Iván García.** 2005. [Online] agosto de 2005.
- —. 2005. RedKsr.com. [Online] 2 de agosto de 2005. [Citado em: 9 de junio de 2009.] http://www.redksr.com/~talos/JAVA/JAVA3D/eclipse.pdf.
- Modelo de Implementación:Diagramas de Componentes y Despliegue. [Online] [Citado em: 8 de junio de 2009.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf. (referencia 16 en la parte de disp)
- **4D SAS**. Utilizar el protocolo SSL. 4D. [Online] 2003. [Citado em: 9 de junio de 2009.] http://www.4d.com/docs/CMS/CMS02064.HTM.

ANEXOS.



Anexo 1: Organigrama del CIGB

GLOSARIO DE TÉRMINOS.

- **Aseguramiento de la Calidad:** Parte de la gestión de la calidad orientada a proporcionar confianza en que se cumplirán los requisitos de calidad.
- **BCP:** Bancos de Células Primario
- **BCT:** Bancos de Células de Trabajo
- **BPC:** Buenas Prácticas Clínicas.
- **BPL:** Buenas Prácticas de Laboratorio.
- **BPP:** Buenas Prácticas de Producción.
- **Buenas Prácticas:** El concepto de buenas prácticas se refiere a las políticas se caracterizan por lograr cumplir eficazmente las metas planteadas.
- **Calidad:** Grado en que un conjunto de características inherentes cumplen con los requisitos.
- **CIGB:** Centro de Ingeniería Genética y Biotecnología. Centro de investigación-producción en la esfera de la Biotecnología ubicado en la Ciudad de la Habana en Cuba.
- **Control de la Calidad:** Parte de la gestión orientada al cumplimiento de los requisitos.
- **CU:** Caso de Uso
- **IFA:** Ingrediente Farmacéutico Activo.
- **UCI:** Universidad de la Ciencias Informáticas. Universidad que surge como fruto de la Batalla de Ideas y como estandarte en la revolución informática y tecnológica vigente en el país.
- **LIMS:** Laboratory Information Management System. Sistema de Gestión de Información de los Laboratorios. Es uno de los sistemas de gestión de la información implementados en el mundo que ofrece múltiples ventajas a los procesos que se realizan en los laboratorios.
- **PCR:** Reacción en cadena de la polimerasa.
- **RUP:** Rational Unified Process. Proceso Unificado de Desarrollo. Es una metodología de desarrollo que indica cómo construir técnicamente un software.

- **TIC:** Tecnologías de la Información y las Comunicaciones.