Universidad de las Ciencias Informáticas

Facultad 6

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

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Alberto Sánchez García

Jorge Carlos Puig Pinto

Tutor(es): Ing. Lisdany De la Fuente Díaz

Ing. Oscar Reimar Cedeño Delgado

Ciudad de La Habana, Cuba

Junio de 2009

Grande es el arte de comenzar, pero mayor es el arte de concluir.

La clave del éxito depende sólo de lo que podamos hacer de la mejor manera posible.

Henry Wadsonrth Longfellow

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 _____.

_____	_____
Alberto Sánchez García	Jorge Carlos Puig Pinto
Autor	Autor
_____	_____
Ing. Lisdany De la Fuente Díaz	Ing. Oscar Reimar Cedeño Delgado
Tutor	Tutor

DATOS DE CONTACTO

Tutora:

Ing. Lisdany de la Fuente Díaz.

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

Email: ldelafuente@uci.cu.

Tutor:

Ing. Oscar Reimar Cedeño Delgado.

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

Email: ordeceno@uci.cu

A los inolvidables muchachos del grupo tres, por los momentos especiales vividos durante la carrera.

A nuestros profesores por dejar como herencia valiosa, el conocimiento.

A nuestros compañeros del proyecto LIMS por todo el esfuerzo compartido.

A nuestros tutores, por la paciencia y el ánimo. Por enseñarnos tanto y por confiar en nosotros.

A Fidel, por ser creador de esta idea.

A todos, les estamos eternamente agradecidos.

A Dios por hacer realidad este sueño.

A mamá por su amor infinito, por ser mi guía y ejemplo. Porque en los momentos difíciles no contempló lo imposible; jamás existirá forma de agradecer una vida de lucha, sacrifico y esfuerzo constate. Por los rezos y por confiar en mí.

A papá por quererme y enseñarme tanto, por ser mi fuerza. Porque nunca me ha faltado su mano.

A la niña, por alegrarme en los días tristes, por el cariño, por todo el tiempo que le robé pensando en mí, por ser tan especial.

A Ángel por la preocupación, por estar cuando lo necesito, por quererme y querer tanto a mamá.

A mami por acompañarme, por las noches de abuela.

A mis tías, por las llamadas y el desvelo. Por la dulzura.

A toda mi familia, gracias por lo que hemos logrado. Todo lo que soy se lo debo a ustedes.

A Massiel, por su inocencia y grandeza, por su amistad incondicional.

A Jorge Carlos por su paciencia. Por todo el apoyo y confianza.

A Yuni y Day por llegar en el momento preciso y quedarse para siempre.

A todos, gracias por existir.

Alberto

A mi mamá por brindarme su confianza, por siempre pedir más, por sentirse orgullosa de los niños que tiene y darme el amor más grande del mundo.

A mi papá ¨no por callado eres silencio¨ se que siempre ha estado a mi lado, gracias por ser mi luz al final del túnel.

A mi abuelo (mi segundo papá) por ser mi mejor maestro, por ser ejemplo inigualable.

A las dos viejitas que más quiero en el mundo (mis abuelas) por darme siempre lo mejor de ellas, a mi abuela Ena por su sabiduría y acertados consejos y a mi abuela Erada por su corazón tan blando.

A mi hermanita por ese cariño inmenso, por soportar al hermano más pesadito del mundo y por nuestra linda y dura competencia.

A mi novia Adriana por brindarme su apoyo todos estos tantos años, espero que su cariño y amor dure la eternidad.

A mi tío Magín por todo el apoyo brindado a mí y a la familia.

A mi prima Zulma gracias por el apoyo tecnológico.

A mi amigo casi hermano Jose Ernesto Acebal por todos los dolores de cabeza que le di y siempre acordarse de mí, gracias por quererme como tu hermano.

A Belkis Folgar por ser mi amiga y darme los mejores consejos del mundo.

A mis amigos Bladimir, Arsenio, Gustavo, Landy, Joan Seijo, Yoan Manuel, Carlos, Indira, Javier por su amistad brindada.

A mi compañero de Tesis que lo considero mi amigo gracias por buscar la perfección y trabajar tan duro, no me imaginé nunca tener tan buen compañero.

Jorge Carlos

A Mis Padres

Por su presencia y ser siempre el motivo más grande de aliento y estímulo para seguir adelante en la vida. Porque gracias a su cariño, apoyo y confianza en mí he llegado a realizar la más grade de mis metas.

Alberto

A Mi Familia

Por ser mi aliento para lograr este sueño.

Jorge Carlos

RESUMEN

La Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología (CIGB) cotidianamente consulta de forma manual grandes volúmenes de documentos, lo que representa una pérdida de tiempo considerable en la realización de sus actividades. Con el propósito de informatizar el acceso y control de esa información se está desarrollando un Sistema de Gestión de Información de Laboratorios (LIMS, del inglés Laboratory Information Management System). Un conjunto de analistas identificaron los principales procesos que realiza el Grupo de Análisis Químico (GAQ) perteneciente al CIGB y desarrollaron el Modelo de Negocio. Posteriormente se obtuvieron los requisitos del sistema y se realizaron todas las actividades del Análisis y Diseño.

Este trabajo de diploma se centra en la implementación del módulo Análisis Químico donde se realizan diferentes técnicas físico-químicas y bioquímicas para la determinación de impureza y pureza de proteínas y se lleva el control analítico de los reactivos y componentes críticos que son utilizados como materia prima en la producción de productos farmacéuticos en el CIGB.

PALABRAS CLAVE

CIGB, Gestión de la Información, LIMS

TABLA DE CONTENIDOS

INTRODUCCIÓN

El siglo XXI exige nuevos retos para alcanzar un desarrollo social. Las Tecnologías de la Información y las Comunicaciones (TICs) permiten disímiles modos de vida y de trabajo, logrando que cada ciudadano pueda beneficiarse de una economía basada en conocimientos. Siguen el ritmo de los continuos avances científicos y transforman las estructuras financieras, políticas y culturales de las sociedades modernas. Como máximo exponente de la cultura tecnológica potencian el trabajo en instituciones y centros educacionales obteniéndose resultados significativos en cualquier región o país del mundo.

En el contexto actual de desarrollo social, las TICs adquieren un impulso vertiginoso en las nuevas formas de educar y comunicarse. Constituyen la piedra angular de todo proceso, rediseñando aceleradamente las formas de producir, vender y competir. Se emplean para la solución de problemas en las esferas de las ciencias biomédicas y en la gestión de las instituciones de salud crecen a pasos agigantados, tanto en países altamente industrializados como los que están en proceso de desarrollo.

Cuba aboga por el uso masivo de las Tecnologías de la Información y las Comunicaciones. Revoluciona la educación para multiplicar los conocimientos de las nuevas generaciones, mediante la creación de los Joven Club de Computación y Electrónica y el uso intensivo en todos los centros de estas tecnologías. Prioriza la informatización del Sistema de Salud que en el momento actual es universal, gratuito y está al alcance de todos los cubanos. En otros sectores como el turismo y el comercio exterior se han obtenido resultados satisfactorios con la introducción y desarrollo de las TICs.

El Centro de Ingeniería Genética y Biotecnología (CIGB) es uno de los pilares fundamentales en el desarrollo de la biotecnología cubana. Enfocado a perfeccionar el sistema de salud, asume la responsabilidad de apoyar directamente el progreso económico y social del país. Actualmente se lleva a cabo el proceso de informatización de esta institución que espera aprovechar al máximo las múltiples ventajas que ofrecen las TICs.

Todos los productos desarrollados en este centro llevan el sello de la calidad que los distingue y los hace únicos entre los de su tipo en el mundo. La Dirección de Calidad, destinada a garantizar la eficiencia, eficacia y seguridad en sus producciones, está constituida por la Dirección de Aseguramiento de la Calidad y la Dirección de Control de la Calidad. (Ver anexo 1)

La Dirección de Aseguramiento de la Calidad garantiza que se lleven a cabo las acciones planificadas y sistemáticas que son necesarias para proporcionar la confianza de que sus productos y servicios satisfacen los requisitos de calidad establecidos. Verifica el cumplimiento de las Buenas Prácticas de

Producción (BPP), Buenas Prácticas de Laboratorio (BPL) y Buenas Prácticas Clínicas (BPC). Esta dirección está compuesta por dos departamentos y cuatro grupos de trabajo que se mencionan a continuación:

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

La Dirección de Control de la Calidad lleva a cabo una serie de funciones relacionadas con el muestreo, los ensayos, las especificaciones y la evaluación de la calidad de los productos que se desarrollan. Para obtener buenos resultados en la labor realizada cuenta con grupos de trabajo y departamentos, a continuación se hace referencia a su estructura organizativa:

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

- Grupo de Liberación Analítica

El **Grupo de Análisis Químico** realiza diferentes técnicas físico-químicas y bioquímicas para la determinación de impurezas y purezas de las diferentes proteínas que se producen o investigan en el centro. Lleva el control analítico de los reactivos y componentes críticos que son utilizados como materia prima en la producción de los productos farmacéuticos en el CIGB. Todas estas técnicas se realizan bajo un estricto cumplimiento de las Buenas Prácticas de Laboratorio garantizando así resultados confiables, lo cual ha sido demostrado en cada inspección realizada por inspectores internos, por la entidad nacional regulatoria (CECMED) y por organizaciones internacionales del nivel de la Organización Mundial de la Salud (OMS).En el grupo se realizan alrededor de 25 técnicas analíticas diferentes, las cuales se encuentran validadas. [1]

Todo el trabajo con las planillas en el Grupo Análisis Químico se realiza manualmente, la información que se recoge es muy amplia y resulta difícil la gestión de los datos. La gran cantidad de documentos a consultar genera una considerable pérdida de tiempo en los procesos que se realizan en el laboratorio. Se hace necesario llevar un control estricto de la información que resulta importante para un buen desempeño de la institución.

El desarrollo actual exige a las empresas que produzcan con la calidad requerida para poder entrar en la competencia del mercado mundial. Se hace necesario buscar mecanismos que permitan agilizar los procesos dentro de estas empresas, como una solución informática que maneje una cantidad significativa de datos surge a finales del pasado siglo los Sistemas de Gestión de Información. Bajo la premisa de garantizar un producto eficaz, un grupo de analistas propuso incorporar al entorno del CIGB los beneficios y mejoras de los Sistemas de Gestión de Información de Laboratorios (LIMS, del inglés Laboratory Information Management System). Estos Sistemas de Gestión de Información que están dedicados específicamente para mejorar el trabajo en los laboratorios, manejan variada y abundante información y resultan imprescindibles para la industria moderna. Permiten almacenar, calcular y gestionar datos de distintas formas, aumentando la productividad y la eficiencia en las actividades realizadas en los laboratorios.

Varios analistas realizaron un estudio a los mecanismos de almacenamiento y procesamiento de la información en el Grupo Análisis Químico y obtuvieron el modelado del negocio. Definieron los procesos a automatizar e identificaron las clases del diseño con sus atributos y relaciones. Por último, diseñaron la base de datos relacional correspondiente a este grupo. Actualmente se hace inmediata la

búsqueda de una solución funcional que posibilite la gestión de los datos que se manejan en el Grupo de Análisis Químico y consecuentemente la reducción del tiempo de realización de los procesos.

Por lo antes expuesto se identifica como **problema científico** de la investigación: ¿Cómo obtener un producto funcional para el módulo Grupo de Análisis Químico del LIMS de Calidad del CIGB?

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

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

En la búsqueda de la solución al problema planteado se define como **objetivo general**: Implementar el módulo Análisis Químico para el Sistema de Gestión de Información de los Laboratorios de la Dirección de Calidad del CIGB.

Para cumplir este objetivo se definieron las siguientes **tareas de la investigación**:

- Estudio de la documentación obtenida en las iteraciones anteriores realizadas al módulo Análisis Químico.
- Familiarización con las herramientas y tecnologías definidas en la arquitectura del proyecto.
- Elaboración de los diagramas de componentes.
- Implementación del módulo Análisis Químico.
- Realización de pruebas unitarias.
- Validación de la seguridad del módulo

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

Capítulo I: “Fundamentación Teórica”: En este capítulo se realiza un estudio de las tecnologías y tendencias que existen a nivel mundial en cuanto al desarrollo de software y que pudieran ser útiles en el avance de la propuesta de solución. Se caracteriza la metodología a utilizar y se describen los roles que se desempeñaran y los artefactos que se obtendrán para apoyar la solución de la problemática planteada.

Capítulo II: “Implementación del sistema”: En el presente capítulo se realiza un análisis preliminar de algunos artefactos generados en flujos de trabajo anteriores al de implementación y que se consideran útiles para el mismo. Se muestran los artefactos obtenidos durante este flujo de trabajo y pequeños fragmentos de código fuente relevante. Por último, se visualizan resultados de las pruebas de unidad realizadas por el framework Lime que posibilita la detección de defectos en el código.

Capítulo 1
Fundamentación Teórica

1.1 Introducción

En este capítulo se realiza un estudio de las tecnologías y tendencias que existen a nivel mundial en cuanto al desarrollo de software y que pudieran ser útiles en el avance de la propuesta de solución. Se caracteriza la metodología a utilizar y se describen los roles que se desempeñarán y los artefactos que se obtendrán para apoyar la solución de la problemática planteada.

1.2 Objeto de estudio

1.2.1 Sistema para la Gestión de la Información de Laboratorios

En la actualidad, la mayoría de la información y los documentos de un laboratorio se guardan electrónicamente en los ordenadores. Esta forma de almacenamiento en formato electrónico permite una manipulación de la información mucho más fácil que en cualquier otro tipo de formato. Sin embargo, la transferencia de dichos datos se hace difícil si no está automatizada, debido a la cantidad de información que debe ser almacenada y procesada en los laboratorios. La generación de la documentación necesaria para cumplir los requisitos de las normas de calidad (ISO 17025, 9000…) necesita a veces más tiempo que los propios análisis en sí mismos, consumiendo muchas veces la mayor parte del tiempo del trabajo diario del laboratorio, por tal motivo surgieron los LIMS (Laboratory Information Management Systems), sistemas de manejo de información diseñados para todo tipo de laboratorios.

Un LIMS (Laboratory Information Management System, o Sistema de Gestión de la Información del Laboratorio) es un conjunto de herramientas basadas en sistemas informáticos. Estas herramientas son esenciales para la reducción de costes y aumento de la eficiencia y productividad en las actividades realizadas en los laboratorios. [2]

Actualmente los LIMS están al alcance de todas las economías porque normalmente se ajustan a los equipos informáticos que posee un laboratorio, o bien las inversiones a realizar son de pequeña envergadura. Además aun siendo sistemas informáticos sofisticados, suelen ser sistemas fáciles de usar y de poner en marcha.

Es importante resaltar que un LIMS no acelera el proceso analítico, pero sí reduce el tiempo dedicado a la administración de la información con lo cual los analistas pueden dedicar más tiempo a labores puramente químicas y mejorar sus resultados y conocimientos.

1.2.2 Sistema para la Gestión de la Información de Laboratorios en el mundo

Las aplicaciones LIMS son herramientas esenciales para la reducción de costos y aumento de la eficiencia en las actividades realizadas en los laboratorios. Dado el creciente interés de las empresas hacia la calidad como factor clave para favorecer la productividad, la eficacia y la imagen de los productos o servicios suministrados, las aplicaciones LIMS representan la solución imprescindible para una gestión moderna del control y aseguramiento de la calidad. Su uso se extiende a todo tipo de laboratorios (control de calidad, investigación y desarrollo, prestación de servicios…) en sectores industriales tan diversos como el farmacéutico, químico y petroquímico, agroalimentario, medioambiental, así como en los laboratorios farmacéuticos de los hospitales.

Algunos de los LIMS existentes en el mundo son los siguientes:

LabWare LIMS es un Sistema de Gestión de Información del Laboratorio (LIMS) Cliente/Servidor real integrado en el entorno Windows. Esta arquitectura combina el poder y la seguridad de un servidor típico con la facilidad de uso provista por el GUI de Microsoft. LabWare LIMS es uno de los primeros productos LIMS del mundo configurable por el cliente, suministrando un grado de participación al cliente para adaptar el software a sus necesidades específicas. [3]

StarLIMS es un Sistema de Administración de Información en Laboratorios con tecnología innovadora, flexible y fácil de usar. La arquitectura Multi-Capa única a StarLIMS ha sido diseñada con el objetivo de poner fin a grandes re-inversiones periódicas y a la pérdida de contenidos resultante de las grandes actualizaciones de versión (upgrades). La independencia y la separación entre la tecnología, las reglas de negocio y los componentes de la base de datos, permite una completa partición entre el constante desarrollo y las tareas de mantenimiento. Diseñado para una amplia variedad de laboratorios operando en muchas disciplinas científicas industriales. [4]

Matrix LIMS se adapta a cualquier industria y laboratorio, tanto en los sectores altamente regulados como en los no regulados. Las aplicaciones Matrix LIMS solucionan las necesidades de grandes empresas y también de organizaciones de tamaño pequeño o mediano, desde incluso un único usuario o un departamento. Matrix LIMS se suministra con el conjunto más completo de herramientas para lograr una configuración absoluta sin necesidad de programar códigos a medida. Los distintos flujos de

trabajo de un laboratorio pueden diseñarse y personalizarse a través de una interfaz familiar al usuario. Las herramientas de configuración no debilitan la seguridad del sistema, sino que al contrario, la incrementan. [5]

De forma general los sistemas mencionados anteriormente se centran en funcionalidades que no se asemejan al proceso de desarrollo del CIGB. Son sistemas propietarios y su costo es elevado. Existen otros LIMS que aunque son de código abierto brindan soluciones que se encuentran muy distantes de las exigencias de este centro. A continuación se mencionan tres de ellos:

BIKA LIMS se estructuran a un servidor estándar de edición para que los clientes puedan agregar módulos opcionales y personalizaciones. Emplea la tecnología moderna. Es independiente de la plataforma y está basado en la web. Se construye en Plone, un marco de gestión de contenido que está estrechamente integrado con Zope, que es un servidor de aplicaciones Web ampliamente utilizado de fuente abierta orientado a objetos. Esta liberado bajo la licencia pública general (GPL). [6]

LABMATICA permite gestionar muestras, generar fácilmente informes y certificados que pudieran ser utilizados como constancia del trabajo realizado. Se puede configurar fácilmente plantillas.

El software es cliente / servidor con la solución de múltiples capacidades de usuario. Está escrito en JAVA y publicado bajo doble licencia, la primera de ellas: la licencia GNU General Public License (GPL) y la otra: una licencia comercial. GNU / GPL versión se ejecuta en base de datos del servidor MySQL. La versión comercial se puede utilizar en otros sistemas de bases de datos relacionales. [7]

OPEN LIMS "se basa en un modelo cliente/servidor, sólo se necesita un navegador para acceder a los datos del proyecto o introducir nuevos datos en el lado del cliente. 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 un 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, será publicado bajo la licencia GPL"[8]. Este LIMS se encuentra actualmente en fase de construcción.

Luego de un estudio realizado en investigaciones anteriores, a los Sistemas de Gestión de la Información de los Laboratorios mencionados, valorando sus características, ventajas y desventajas,

se determinó que el Centro de Ingeniería Genética y Biotecnología no cuenta con toda la tecnología necesaria para poder utilizarlos, esto conlleva al desarrollo de un Sistema para la Gestión de la Información de Laboratorios en la Dirección de Calidad del CIGB que responda a sus intereses particulares.

Por tal motivo, el CIGB y la Universidad de las Ciencias Informáticas (UCI) se han dado a la tarea de realizar este LIMS para la gestión del trabajo que se realiza en el área de calidad del CIGB, con el cual se pretende:

- Completo seguimiento y manejo de las muestras permitiendo su agrupación por experimentos o departamentos. Llevar un control estricto de la información.
- Generación de informes que pueden ser impresos. Permitir almacenar, calcular y gestionar datos de distintas formas.
- Reducción del tiempo de procesamiento de cada muestra y de la transferencia de los resultados. Minimizar al máximo los errores humanos.

1.3 Herramientas y metodologías que apoyan la solución

En cada proyecto de software es necesario realizar un análisis minucioso de las características de las metodologías y las herramientas, que influyen en el buen desempeño del mismo. Aunque definido por el proyecto, se hace necesario conocer las características generales de ambas y especificar las versiones utilizadas en las herramientas.

1.3.1 Metodología de desarrollo de software

Muchas veces el proceso de desarrollo de software resulta riesgoso y se convierte en una tarea difícil hallar el modo de controlar su curso de principio a fin. Su problema principal radica en cómo coordinar todas las actividades que comprende, sobre todo si se trata de un proyecto de gran envergadura. De modo que se torna imprescindible contar con una forma organizada y adecuadamente estructurada para trabajar.

Se necesita un proceso que integre las múltiples fases del desarrollo, un método común que:

- Proporcione una guía para ordenar las actividades de un equipo.
- Dirija las tareas de cada desarrollador por separado y del equipo como un todo.
- Especifique los artefactos que deben desarrollarse.

- Ofrezca criterios para el control y la medición de los productos y actividades del proyecto.

La presencia de un proceso bien definido y bien gestionado, es una diferencia esencial entre proyectos hiperproductivos y otros que fracasan. [9]

En ocasiones evadido, el uso de una metodología de desarrollo de software acertada, permite obtener desarrolladores satisfechos con el software de calidad que ha sido fruto de su trabajo y clientes conformes con el producto realizado. En este sentido se decidió aplicar el Proceso Racional Unificado para guiar el desarrollo del software que se propone.

1.3.2 Rational Unified Process (RUP)

El Proceso Racional Unificado (RUP, del inglés Rational Unified Process) es un proceso de desarrollo de software utilizado para realizar el análisis, diseño, implementación, documentación y pruebas a sistemas orientados a objetos. Se trata de un proceso iterativo e incremental debido a que se basa en la evolución de prototipos ejecutables que se muestran a los usuarios y clientes. Se caracteriza por ser centrado en la arquitectura porque establece refinamientos sucesivos de una arquitectura ejecutable, construida como un modelo evolutivo de manera que no se afecte de forma significativa ante posibles modificaciones, para lograr finalmente una arquitectura comprensible, adaptable y robusta. Por último, RUP está dirigido por los casos de uso, pues guía el desarrollo del proyecto manteniendo como un aspecto de vital importancia la satisfacción del usuario y no sólo teniendo en cuenta las funcionalidades del sistema.

Para guiar el proceso de desarrollo de software, RUP propone cuatro fases: inicio, elaboración, construcción y transición y se han agrupado las actividades en grupos lógicos definiéndose nueve flujos de trabajos: modelado del negocio, requerimientos, análisis y diseño, implementación, prueba, instalación, administración de proyectos, administración de configuración y cambios y ambiente.

Flujo de Trabajo: Implementación

Actualmente se lleva a cabo el flujo de trabajo implementación para darle respuesta a la problemática planteada. Este flujo posibilita describir cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. De forma más específica, la implementación facilita:

1. Planificar las integraciones de sistema necesarias en cada iteración. Se realiza un enfoque incremental, lo que da paso a una sucesión de acciones pequeñas y manejables.
2. Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue. Permite tener bien claro en qué nodo se ejecutará cada componente.
3. Implementar las clases y subsistemas encontrados durante el diseño. Encierra en gran medida el esfuerzo de trabajo en este flujo.
4. Probar los componentes individualmente e integrarlos para luego realizar las comprobaciones de sistema. [10]

Roles y artefactos del flujo de trabajo: Implementación

Un rol define las responsabilidades de un individuo, o de un grupo de individuos de un equipo de trabajo. Una persona puede desempeñar diversos roles y un rol puede ser representado por varias personas. [11]

Una actividad es una unidad de trabajo que una persona que desempeñe un rol puede realizar. Las actividades tienen un objetivo concreto, crear o actualizar algún producto.

Un artefacto es un fragmento de información que es producido, modificado o usado durante el proceso de desarrollo de software. Los artefactos son los resultados tangibles del proyecto que se van creando y usando hasta obtener el producto final.

Los roles que se desempeñarán en este trabajo de diploma son, el arquitecto, el integrador del sistema e implementador y los artefactos que se obtendrán o actualizarán son el modelo de implementación, el modelo de despliegue, la descripción de la arquitectura, el plan de integración del sistema, la implementación de subsistemas y los diagramas de componentes.

El arquitecto es el responsable de la integridad, corrección, completitud y legibilidad del modelo de implementación de acuerdo a lo descrito en el modelo de diseño y desempeña un papel primordial en la elaboración de la arquitectura del modelo de implementación. [10]

El integrador del sistema planifica la secuencia de construcciones necesarias en cada iteración y la integración de cada construcción cuando sus partes han sido implementadas.

El implementador tiene como actividad fundamental implementar el sistema, los elementos del diseño, los elementos de prueba de diseño y analizar el comportamiento del sistema en tiempo de ejecución.

A continuación se explican los artefactos principales que se consideran oportunos resaltar por su peso dentro de este flujo de trabajo:

Artefacto: Modelo de Implementación

El modelo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes. El modelo de Implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje de programación utilizados, y cómo dependen los componentes uno de otro. [12]

Artefacto: Diagramas de componentes

Son usados para estructurar el modelo antes mencionado en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. Representa una división física del sistema.

Se representa como un grafo de componentes de software unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfaces que éstos soporten. [10]

Se utilizan para modelar la vista estática de un sistema. Muestra la organización y las dependencias lógicas entre un conjunto de componentes de software. Se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

Artefacto: Modelo de despliegue

El modelo de despliegue provee un modelo detallado de la forma en la que los componentes se desplegarán a lo largo de la infraestructura del sistema. Especifica las capacidades de red, los requisitos de hardware y otra información relacionada con el despliegue del sistema propuesto. [12]

Artefacto: Subsistema de implementación

Los subsistemas de implementación proporcionan una forma de organizar los artefactos del modelo de implementación en pedazos más manejables. Un subsistema puede estar formado por componentes, interfaces y otros subsistemas (recursivamente). Además, un subsistema puede implementar y así proporcionar las interfaces que representan la funcionalidad que exportan en forma de operaciones.

Los subsistemas de implementación deberían seguir la traza uno a uno de sus subsistemas de diseño correspondiente.

1.3.3 Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML, del inglés Unified Modeling Language), es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. [13]

Se utiliza UML porque posee algunas características que favorecen la implementación:

- Permite modelar sistemas utilizando técnicas orientadas a objetos (OO).
- Permite especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos.
- Permite documentar todos los artefactos de un proceso de desarrollo.
- Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.

UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

1.3.4 Herramientas CASE: Visual Paradigm for UML Enterprise Edition (Versión 6.1)

Visual Paradigm 6.1 utiliza UML como Lenguaje de Modelado, es una Herramienta CASE poderosa y fácil de usar. Permite representar todo tipo de diagramas UML para las distintas fases como la captura de requisitos, análisis, diseño e implementación, permite además generar códigos, aplicar ingeniería inversa en los lenguajes Java, C++, CORBA IDL, PHP, XML, Ada y Python. Además la generación de código apoya C#, Visual Basic .NET, Object Definition Language (ODL), Flash, Delphi y Perl.

Mediante esta herramienta se puede desarrollar un producto de calidad teniendo el diseño centrado en casos de usos y enfocado al negocio. Facilita la comunicación ya que utiliza un lenguaje estándar común a todo el equipo de desarrollo. También posee disponibilidad de múltiples versiones, en múltiples plataformas y es fácil de instalar y actualizar. Presenta la posibilidad de la interoperabilidad con otras aplicaciones como es el Rational Rose.

Tiene disponible distintas versiones: Enterprise, Professional, Standard, Modeler, Personal y Community.

1.3.5 Sistema de Gestión de Base de Datos: PostgreSQL (Versión 8.2.0)

Con vista a satisfacer los requisitos impuestos por las nuevas aplicaciones, la investigación y desarrollo de las bases de datos sigue diferentes tendencias, las cuales generalmente involucran la integración de la tecnología de las bases de datos con los lenguajes de programación orientado a objetos. Las tendencias principales pueden caracterizarse de la siguiente forma:

1. Extensión de los sistemas relacionales: Esta tendencia está cercana a los Sistemas de gestión de bases de datos (SGBD) relacionales. En general hay una tendencia a extender los SGBD relacionales con varias funciones, por ejemplo, representación de objetos complejos o para definir disparadores (triggers): acciones que sean ejecutadas automáticamente cuando el sistema alcance condiciones específicas concernientes a los datos.

2. Sistema de gestión de bases de datos orientado a objetos: Integran la tecnología de bases de datos con el paradigma orientado a objetos que se ha desarrollado en el área de los lenguajes de programación y la ingeniería de software.[14]

Aunque en general, las diversas tendencias se basan en aproximaciones diferentes, se puede prever con bastante claridad que la mayoría de las próximas generaciones de SGBD tendrán un conjunto de características comunes que incluirán la posibilidad de definir y manipular objetos complejos, ciertas formas de jerarquías de tipos, mecanismos para manipular reglas deductivas y restricciones de integridad.

PostgreSQL, fue desarrollado por el equipo de Oracle. Se recomienda la utilización de PostgreSQL para la elaboración de un sistema robusto y para lograr mayor escalabilidad. El proyecto PostgreSQL tiene actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto. [15]

Es un Gestor de Bases de Datos objeto-relacional altamente extensible, soporta operadores y tipos de datos definidos por el usuario y es capaz de manejar complejas rutinas y reglas de modo que su avanzada funcionalidad se pone de manifiesto con las consultas SQL declarativas, el control de concurrencia multiversión, soporte multiusuario, transacciones, optimización de consultas, herencia y valores no atómicos (atributos basados en vectores y conjuntos). Además cuenta con un mejor soporte para subselects, triggers, vistas y procedimientos almacenados en el servidor, además tiene ciertas características orientadas a objetos, puede manejar un ilimitado número de base de datos y permite

una fácil gestión de los usuarios Este sistema cuenta con una API (Application Program Interface) flexible lo cual ha permitido dar soporte para el desarrollo con PostgreSQL en diversos lenguajes de programación entre los que se incluyen: Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

"MVCC, o control de versiones múltiples concurrentes, es la tecnología que PostGreSQL usa para evitar los seguros (locking) innecesarios". [16] Esto significa que mientras se consulta una base de datos, cada transacción ve una imagen de los datos (una versión de la base de datos) como si fuera tiempo atrás, sin tener en cuenta el estado actual de los datos que hay por debajo.

PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo.

1.3.6 Servidor Web: Apache (Versión 2.2.6)

Apache es un servidor HTTP, de código abierto y licenciamiento libre. Ha desempeñado un papel muy importante en el crecimiento de la red mundial, y continua siendo el servidor HTTP más utilizado, siendo además el servidor de facto contra el cual se realizan las pruebas comparativas y de desempeño para otros productos competidores. Apache es desarrollado y mantenido por una comunidad de desarrolladores auspiciada por Apache Software Foundation [17]:

- Está disponible para una gran multitud de plataformas como GNU/Linux, Mac OS, Mac OS X Server, Netware, Open Step/Match, UNIX, Solaris, SunOS, UnixWare, Windows entre otras.
- Permite la autenticación de usuarios en varias formas con el objetivo de restringir el acceso a determinadas páginas de un sitio Web de una forma sencilla y de fácil mantenimiento.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor dando la posibilidad de ejecutar un determinado script cuando ocurra un error en concreto.
- Permite la creación de sitios Web dinámicos mediante el uso de CGI's, de Server Side Includes (SSI), de lenguajes de Scripting como PHP, JavaScript, Python, Java y páginas jsp.

- Apache está diseñado para ser un servidor Web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos.[18]

1.3.7 Entorno de Desarrollo: Eclipse (Versión 3.3.1.1)

Como herramientas de desarrollo de apoyo a la programación por su facilidad de uso son muy utilizados los entornos de desarrollo integrado. Un entorno de desarrollo integrado o en inglés Integrated Development Environment ('IDE') es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un lenguaje de programación o bien, poder utilizarse para varios.

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

1.3.8 Framework: Symfony (Versión 1.0.18)

En el desarrollo de software un framework es muy utilizado; es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente puede incluir soporte de programas, librerías y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Agrega funcionalidad extendida a un lenguaje de programación y proporciona una estructura al código. Facilita a los desarrolladores que escriban un código más entendible y fácil de mantener. Además hace la programación más fácil, convirtiendo complejas funciones en sencillas instrucciones.

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

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

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares)
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador sólo debe configurar aquello que no es convencional.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.[19]

1.3.9 Lenguaje de Programación: PHP 5 (Versión 5.2.5)

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje "Open Source" interpretado de alto nivel, especialmente pensado para desarrollos web y el cual puede ser embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP. [20]

PHP es gratuito, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre y al ser de código abierto, puede ser utilizado, modificado y redistribuido sin coste alguno. PHP ha alcanzado gran popularidad y existe una amplia comunidad de desarrolladores y programadores que continuamente implementan mejoras en su código.

Este lenguaje posee ventajas que lo convierten en una poderosa herramienta para el diseño de páginas web dinámicas, entre las cuales se encuentran:

- Lenguaje rápido, de fácil aprendizaje que permite la conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad Posee gran librería de funciones y mucha documentación disponible.
- Generalmente es usado junto con Apache, lo que lo hace extremadamente veloz.

- PHP es Open Source, el usuario no depende de una compañía específica para arreglar cosas que no funcionan y no tiene que pagar actualizaciones anuales para tener una versión que funcione.
- Trabaja del lado del servidor, con independencia de plataformas.
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.

1.4 Patrones

Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos sobre cómo aplicarlo en nuevas situaciones, o sea, "un patrón es una descripción de un problema bien conocido que suele incluir: descripción, escenario de uso, solución concreta, consecuencias de utilizar el patrón, ejemplos de implementación y lista de patrones relacionados". [21]

1.4.1 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos, comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades. [22]

Un patrón de diseño es una solución a un problema de diseño, para que una solución sea considerada un patrón debe poseer ciertas características una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores, otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

1.4.2 Patrones Arquitectónicos

Son los patrones que definen la estructura de un sistema software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema.

1.5 Conclusiones

En este capítulo se ha realizado un estudio de los elementos más importantes que rodean las tendencias actuales tanto nacionales como internacionales, escogiendo además las herramientas y metodologías que apoyarán esta investigación. Se puede pasar a la implementación del sistema con la plena confianza de haber escogido el lenguaje de programación más versátil en cuanto a conexiones a

Base de Datos. Visual Paradigm se elige gracias a sus potencialidades en el modelado visual y su soporte a UML. Consumiendo la política de software libre y dando paso a garantizar futuras migraciones en una segunda versión del producto se elige por todas sus funcionalidades el framework Symfony y como sistema gestor de base de datos PostgreSQL.

Capítulo 2
Implementación del Sistema

2.1 Introducción

En el presente capítulo se realiza un análisis preliminar de algunos artefactos generados en flujos de trabajo anteriores al de implementación y que se consideran útiles para el mismo. Se muestran los artefactos obtenidos durante este flujo de trabajo y pequeños fragmentos de código fuente relevante. Por último, se visualizan resultados de las pruebas de unidad realizadas, las cuales posibilitan la detección de defectos en el código.

2.2 Requerimientos

Cada flujo de trabajo en el ciclo de desarrollo de un software según RUP, utiliza las salidas generadas por los flujos anteriores como punto de partida para el proceso de obtención de sus propios artefactos. Tal es el caso del flujo de trabajo de Implementación que comienza con el resultado del diseño e implementa el sistema en términos de componentes. Aún así, es necesario conocer los requisitos funcionales y no funcionales obtenidos durante la fase de requerimiento porque resultan necesarios para comprender y guiar todo el proceso restante.

2.2.1 Requerimientos funcionales

Las capacidades o condiciones que el sistema debe tener constituyen requerimientos funcionales. En investigaciones anteriores realizadas en el mismo módulo los analistas lograron identificar 210 que se agruparon en 42 casos de uso utilizando el patrón CRUD. De estos requisitos se implementaron un total de 167 que responden a 28 casos de uso, los cuales se listan seguidamente:

CU Gestionar Registro de preparación de soluciones (SIC-0020).

RF 1.1 Crear nuevo SIC-0020.

RF 1.2 Buscar SIC-0020.

RF 1.3 Visualizar SIC-0020.

RF 1.4 Modificar SIC-0020 y registrar traza.

RF 1.5 Imprimir SIC-0020.

CU Gestionar Resultado de la Determinación de proteínas por el método de Lowry (SIC-0107).

RF 2.1 Crear nuevo SIC-0107.

RF 2.2 Buscar SIC-0107.

RF 2.3 Visualizar SIC 0107.

RF 2.4 Modificar SIC-0107 y registrar traza.

RF 2.5 Imprimir SIC-0107.

CU Gestionar Curva de calibración de Lowry (SIC-0107C).

RF 3.1 Crear nuevo SIC-0107C.

RF 3.2 Buscar SIC-0107C.

RF 3.3 Visualizar SIC-0107C.

RF 3.4 Modificar SIC-0107C y registrar traza.

RF 3.5 Imprimir SIC-0107C.

CU Gestionar Curva de calibración de carbohidratos (SIC-0112C).

RF 4.1 Crear nuevo SIC-0112C.

RF 4.2 Buscar SIC-0112C.

RF 4.3 Visualizar SIC-0112C.

RF 4.4 Modificar SIC-0112C y registrar traza.

RF 4.5 Imprimir SIC-0112C.

CU Gestionar Curva de calibración de lípidos (SIC-0113C).

RF 5.1 Crear nuevo SIC-0113C.

RF 5.2 Buscar SIC-0113C.

RF 5.3 Visualizar SIC-0113C.

RF 5.4 Modificar SIC-0113C y registrar traza.

RF 5.5 Imprimir SIC-0113C.

CU Gestionar Curva de calibración de Tiocianato (SIC-0116C).

RF 6.1 Crear nuevo SIC-0116C.

RF 6.2 Buscar SIC-0116C.

RF 6.3 Visualizar SIC-0116C.

RF 6.4 Modificar SIC-0116C y registrar traza.

RF 6.5 Imprimir SIC-0116C.

CU Gestionar Curva de calibración de Orcinol (SIC-0762C).

RF 7.1 Crear nuevo SIC-0762C.

RF 7.2 Buscar SIC-0762C.

RF 7.3 Visualizar SIC-0762C.

RF 7.4 Modificar SIC-0762C y registrar traza.

RF 7.5 Imprimir SIC-0762C.

CU Gestionar Curva de calibración de Grupos sulfihidrilos (SIC-0759C).

RF 8.1 Crear nuevo SIC-0759C.

RF 8.2 Buscar SIC-0759C.

RF 8.3 Visualizar SIC-0759C.

RF 8.4 Modificar SIC-0759Cy registrar traza.

RF 8.5 Imprimir SIC-0759C.

CU Gestionar Curva de calibración de fósforo (SIC-1001C).

RF 9.1 Crear nuevo SIC-1001C.

RF 9.2 Buscar SIC-1001C.

RF 9.3 Visualizar SIC-1001C.

RF 9.4 Modificar SIC-1001C y registrar traza.

RF 9.5 Imprimir SIC-1001C.

CU Gestionar Curva de calibración de Ácido Siálico (SIC-0760C).

RF 10.1 Crear nuevo SIC-0760C.

RF 10.2 Buscar SIC-0760C.

RF 10.3 Visualizar SIC-0760C.

RF 10.4 Modificar SIC-0760C y registrar traza.

RF 10.5 Imprimir SIC-0760C.

CU Gestionar Curva de calibración de Biuret (SIC-0710C).

RF 11.1. Crear nuevo SIC-0710C.

RF 11.2. Buscar SIC-0710C.

RF 11.3. Visualizar SIC-0710C.

RF 11.4. Modificar SIC-0710C y registrar traza.

RF 11.5 Imprimir SIC-0710C.

CU Gestionar Curva de calibración de Tween (SIC-0761C).

RF 12.1. Crear nuevo SIC-0761C.

RF 12.2. Buscar SIC-0761C.

RF 12.3. Visualizar SIC-0761C.

RF 12.4. Modificar SIC-0761C y registrar traza.

RF 12.5 Imprimir SIC-0761C.

CU Gestionar Curva de calibración del ácido Bicinconínico (SIC-0763C).

RF 13.1 Crear nuevo SIC-0763C.

RF 13.2 Buscar SIC-0763C.

RF 13.3 Visualizar SIC-0763C.

RF 13.4 Modificar SIC-0763C y registrar traza.

RF 13.5 Imprimir SIC-0763C.

CU Gestionar Resultado del análisis por cromatografía de capa fina (SIC-1000).

RF 14.1. Crear nuevo SIC-1000.

RF 14.2. Buscar SIC-1000.

RF 14.3. Visualizar SIC-1000.

RF 14.4. Modificar SIC-1000 y registrar traza.

RF 14.5 Imprimir SIC-1000.

CU Gestionar Resultado de la Determinación de PH (SIC-0133).

RF 15.1. Crear nuevo SIC-0133.

RF 15.2. Buscar SIC-0133.

RF 15.3. Visualizar SIC-0133.

RF 15.4. Modificar SIC-0133 y registrar traza.

RF 15.5 Imprimir SIC-0133.

CU Gestionar Resultado de la Determinación de la concentración de aluminio (SIC-0105).

RF 16.1 Crear nuevo SIC-0105.

RF 16.2 Buscar SIC-0105.

RF 16.3 Visualizar SIC-0105.

RF 16.4 Modificar SIC-0105 y registrar traza.

RF 16.5 Imprimir SIC-0105.

CU Gestionar Resultado de la Determinación de concentración de carbohidratos por el método de Antrona (SIC-0112).

RF 17.1 Crear nuevo SIC-0112.

RF 17.2 Buscar SIC-0112.

RF 17.3 Visualizar SIC-0112.

RF 17.4 Modificar SIC-0112 y registrar traza.

RF 17.5 Imprimir SIC-0112.

CU Gestionar Resultado de la Determinación del contenido de Lípidos (SIC-0113).

RF 18.1 Crear nuevo SIC-0113.

RF 18.2 Buscar SIC-0113.

RF 18.3 Visualizar SIC-0113.

RF 18.4 Modificar SIC-0113 y registrar traza.

RF 18.5 Imprimir SIC-0113.

CU Gestionar Resultado de la Determinación de la concentración de Tiocianato (SIC-0116).

RF 19.1 Crear nuevo SIC-0116.

RF 19.2 Buscar SIC-0116.

RF 19.3 Visualizar SIC-0116.

RF 19.4. Modificar SIC-0116 y registrar traza.

RF 19.5 Imprimir SIC-0116.

CU Gestionar Libro de entrada de muestras de producción.

RF 20.1 Crear libro de entrada de muestras de producción.

RF 20.2 Buscar libro de entrada de muestras de producción.

RF 20.3 Visualizar libro de entrada de muestras de producción.

RF 20.4 Modificar datos en el libro de entrada de muestras de producción y registrar traza.

CU Gestionar Resultado de la Determinación de Carbohidratos por el método de Orcinol (SIC-0762).

RF 21.1 Crear nuevo SIC-0762.

RF 21.2 Buscar SIC-0762.

RF 21.3 Visualizar SIC-0762.

RF 21.4 Modificar SIC-0762 y registrar traza.

RF 21.5 Imprimir SIC-0762.

CU Gestionar Resultado de la Determinación de proteínas por el método de Acido Bicinconínico (SIC-0763).

RF 22.1 Crear nuevo SIC-0763.

RF 22.2 Buscar SIC-0763.

RF 22.3 Visualizar SIC-0763.

RF 22.4 Modificar SIC-0763 y registrar traza.

RF 22.5 Imprimir SIC-0763.

CU Gestionar Resultado de la Determinación colorimétrica de fósforo por el método de Chen (SIC-1001).

RF 23.1 Crear nuevo SIC-1001.

RF 23.2 Buscar SIC-1001.

RF 23.3 Visualizar SIC-1001.

RF 23.4 Modificar SIC-1001 y registrar traza.

RF 23.5 Imprimir SIC-1001.

CU Gestionar Resultado de la Determinación de la concentración de Tween (SIC-0761).

RF 24.1 Crear nuevo SIC-0761.

RF 24.2 Buscar SIC-0761.

RF 24.3 Visualizar SIC-0761.

RF 24.4 Modificar SIC-0761 y registrar traza.

RF 24.5 Imprimir SIC-0761.

CU Gestionar Resultado de la Determinación de ácido Siálico (SIC-0760).

RF 25.1 Crear nuevo SIC-0760.

RF 25.2 Buscar SIC-0760.

RF 25.3 Visualizar SIC-0760.

RF 25.4 Modificar SIC-0760 y registrar traza.

RF 25.5 Imprimir SIC-0760.

CU Gestionar Resultado de la Determinación de grupo sulfihidrilos (SIC-0759).

RF 26.1 Crear nuevo SIC-0759.

RF 26.2 Buscar SIC-0759.

RF 26.3 Visualizar SIC-0759.

RF 26.4 Modificar SIC-0759 y registrar traza.

RF 26.5 Imprimir SIC-0759.

CU Gestionar Resultado de la Concentración de proteínas en geles de hidróxido de aluminio (SIC-0185).

RF 27.1 Crear nuevo SIC-0185.

RF 27.2 Buscar SIC-0185.

RF 27.3 Visualizar SIC-0185.

RF 27.4 Modificar SIC-0185 y registrar traza.

RF 27.5 Imprimir SIC-0185.

CU Gestionar Resultado de la Determinación de la concentración de proteínas por el método de Biuret (SIC-0710).

RF 28.1 Crear nuevo SIC-0710.

RF 28.2 Buscar SIC-0710.

RF 28.3 Visualizar SIC-0710.

RF 28.4 Modificar SIC-0710 y registrar traza.

RF 28.5 Imprimir SIC-0710.

La documentación del módulo Análisis Químico y el análisis de los requerimientos evidencia que la aplicación debe permitir realizar búsquedas de acuerdo a parámetros estipulados por el usuario, visualizar el resultado de las búsquedas y modificar los campos que el cliente definió en el proceso de negocio como posibles a ser modificados una vez insertados en la base de datos.

2.2.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Los definidos en el proyecto 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 1.0.18 como framework de desarrollo.

2.3 Arquitectura del sistema

La arquitectura de un software permite tener una vista general del sistema, de cierta forma encierra los componentes principales del mismo y evidencia la manera en que estos interactúan. Según la IEE la arquitectura de software es la organización fundamental de un sistema, encarnada en sus

componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. [23]

Resulta difícil en un solo modelo o diagrama reflejar toda la arquitectura de un software. Para manejar esta complejidad se representan diferentes aspectos y características de la arquitectura en múltiples vistas. “El modelo más aceptado a la hora de establecer las vistas necesarias para describir una arquitectura software es el modelo 4+1”. [24]. Por su importancia para la implementación se abordarán en este trabajo de diploma dos de las vistas que recoge este modelo, ellas son: la vista de despliegue y la vista lógica.

2.3.1 Vista Lógica

Cuando se realiza una vista lógica se descompone el sistema en subsistemas y paquetes. Una vez presentadas estas unidades lógicas principales, se profundiza en ellas hasta el nivel que se considere adecuado.

Para estructurar la vista del sistema, no se identificaron en el módulo Análisis Químico subsistemas funcionales, se organizó por paquetes a partir de los casos de uso arquitectónicamente significativos, que constituyen la totalidad de los casos de uso identificados. Los paquetes más generales que se identificaron son los siguientes: Determinar_Grado_Purezas_Impurezas_Proteinas, Realizar_Ensayo_Materias_Primas, Realizar_Ensayo, y Determinar_Estado_de_Materias_Primas, en los que se agrupan los casos de uso arquitectónicamente significativos relacionados con la finalidad del paquete. [25]

La vista lógica se realizó en investigaciones anteriores, para un mejor entendimiento de la imagen mostrada en el Anexo 2 debe remitirse al expediente del proyecto Alas LIMS.

2.3.2 Vista de despliegue

La vista de despliegue representa la arquitectura en tiempo de ejecución del sistema. Muestra cómo se distribuye físicamente el sistema y dónde se ejecutan cada uno de los componentes teniendo en cuenta sus funcionalidades.

La vista de despliegue está compuesta por:

- nodos que representan objetos físicos existentes en tiempo de ejecución y generalmente modelan recursos que tienen memoria y capacidad de proceso.

- instancias de componentes de software localizadas dentro de los nodos. Pueden estar unidas por relaciones de dependencia.
- relaciones entre nodos, éstas representan el tipo de comunicación entre ellos.

En la figura 1 se puede apreciar la vista de despliegue del módulo Análisis Químico que contiene cuatro nodos; computadora del cliente, impresora, servidor de aplicaciones y servidor de base de datos. Además se han representado siete paquetes de componentes; “Security yml”, “Validaciones yml”, “Estilos CSS”, “Validaciones JS”, “Controlador”, “Modelo” y “Vista”.

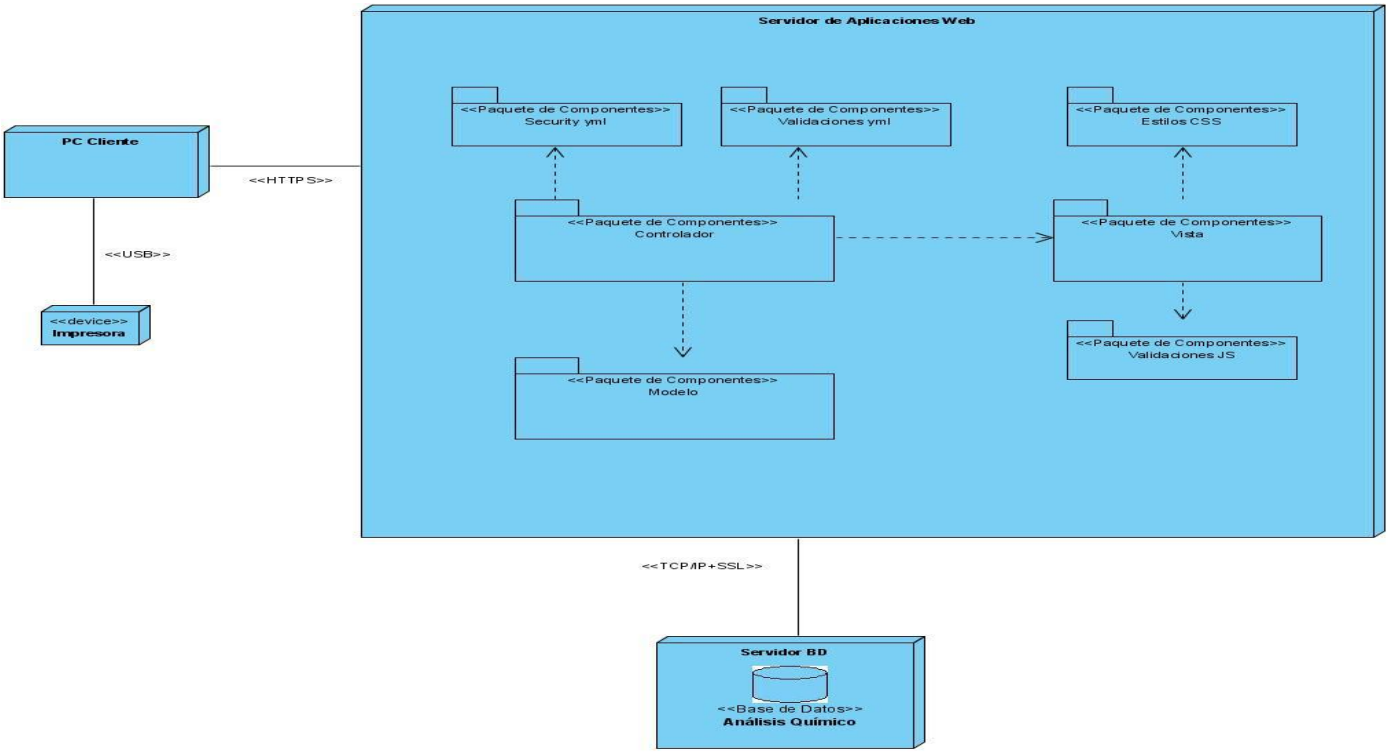


Figura 1: Diagrama de despliegue del módulo Análisis Químico

La aplicación será desplegada de la siguiente manera: la PC cliente estará instalada en el departamento de Análisis Químico del CIGB y será conectada por Bus Universal en Serie (USB, del inglés Universal Serial Bus) a una impresora que le permita llevar al papel todos los documentos que se gestionen en el módulo. Se conectará a un servidor de aplicaciones mediante el protocolo seguro

de transferencia de hipertexto (HTTPS, del inglés Hypertext Transfer Protocol Secure) que es un protocolo de red basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto.

En el despliegue se debe contar además con la base de datos Análisis Químico que estará hospedada en el servidor de base de datos representado en uno de los nodos de la vista de despliegue. Esta debe cumplir con los requerimientos no funcionales especificados anteriormente. Se conectará al servidor de aplicaciones mediante el protocolo TCP/IP y el protocolo de capa de conexión segura (SSL, del inglés Secure Sockets Layer) que proporciona autenticación del servidor, integridad del mensaje y encriptado de datos.

La estructura del servidor de aplicaciones está basada en la arquitectura que utiliza Symfony; Modelo Vista Controlador (MVC). Se ha representado la lógica relacionada con los datos en un paquete de componente denominado "Modelo", la presentación de la interfaz con la que interactúa el usuario en el paquete de componentes "Vista" y el último de estos tres es el paquete "Controlador" que representa la lógica de la aplicación. En el epígrafe siguiente se explica detalladamente cómo Symfony implementa el patrón MVC.

Se han colocado otros paquetes de componentes que complementan las funciones de los tres paquetes mencionados anteriormente. El paquete de componente "Validaciones yml" es usado para el tratamiento de todo tipo de caracteres o entradas no validadas del lado del servidor y el de "Validaciones JS" se utiliza con el mismo objetivo pero permite la validación del lado del cliente. En el paquete de componentes "Estilos CSS" representa a las hojas de estilo en cascada (CSS, del inglés Cascading Style Sheets) utilizado para tener un control centralizado de los estilos empleados a la aplicación, entiéndase efectos visuales. Se ha colocado el paquete de componente "Security yml" que permite restringir el acceso de los usuarios a todas las acciones de los módulos.

2.3.3 Patrón de arquitectura

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura Modelo Vista Controlador (MVC), que está formado por 3 niveles [19]:

- Modelo: es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- Vista: presenta el modelo en un formato adecuado que permite al usuario interactuar con él.

- Controlador: responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y en la vista.

En el anexo 3 se puede observar cómo Symfony toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo.

Este framework hace otra división en el modelo, lo divide en una capa de acceso a datos y otra de abstracción de datos. La abstracción indica qué quiere de la base de datos, y la capa de acceso hace las consultas necesarias para obtener esa información. Si se cambia de sistema gestor de bases de datos, solamente es necesario actualizar la capa de abstracción de la base de datos.

Generalmente dentro de una aplicación las páginas Web contienen elementos que son comunes y sólo cambia su interior. Por tal motivo, dentro de la vista se incluye un layout (plantilla) que es común para todas las páginas, estás solamente tienen que visualizar las variables definidas por el controlador. De esta forma en la capa Vista también se aprovecha la separación del código.

Un parte importante del trabajo de los controladores es prácticamente igual en toda la aplicación. "Entre las tareas comunes se encuentran el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares" [19]. En este sentido, el patrón que utiliza el framework hace una división en esta capa y coloca un controlador frontal que es único para toda la aplicación y las acciones que incluyen código específico del controlador en cada página.

A continuación se realiza un esbozo general de lo que sucede cuando un usuario interactúa con una aplicación que utiliza este tipo de arquitectura:

- El usuario interactúa con la interfaz de usuario, por ejemplo, necesita buscar una de las planillas que se gestionan en el módulo Análisis Químico.
- El controlador recibe la notificación de la acción "buscar" solicitada por el usuario y accede al modelo para obtener los datos y devolverlos a la vista.
- La vista obtiene sus datos a través de del controlador y genera la interfaz apropiada para el usuario, en este caso, el resultado de la búsqueda.
- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Con la utilización de esta arquitectura las aplicaciones resultan más sencillas, la división en las capas modelo, vista y controlador facilita a los desarrolladores la comprensión del código y por ende, el mantenimiento de las aplicaciones.

2.3.4 Patrones de Diseño

En la implementación se han utilizado diferentes patrones de diseño, aunque el framework implícitamente aplica muchos de ellos sería saludable poner a consideración algunos ejemplos donde se han empleado estos patrones.

Ejemplos de patrones GRASP utilizados en la implementación

El patrón Creador fue muy utilizado en la implementación. Las clases Action donde se crean los objetos de las clases que representan entidades son creadoras de dichas entidades. Un ejemplo de esto es la clase "registrarAction.class.php" que pertenece al módulo SIC0112, en esta clase se crea el objeto `$sic0112` de la entidad `Sic0112DeterminacionCarbohidratos`, de esta forma la clase se convierte en creadora de la entidad.

El patrón Experto se utilizó en la implementación de diferentes métodos dentro de las clases generadas por Symfony. Muchas de las planillas que se gestionaron en la aplicación necesitaban un número de folio incremental que era calculado a partir del anterior que estaba insertado en la Base de Datos. Esto se logró con un método en cada clase donde estuviese esta información, tal es el caso de el método `CargarFolio_0107C()` que pertenece a la clase `Sic0107cCcLowry`, esta clase contiene el número de folio que necesita la planilla de `Lowry` cada vez que se va a registrar. De esta forma se logra asignarle la responsabilidad de cargar el folio al experto en la información.

Para lograr que las clases Actions tuviesen un comportamiento lo más cohesivo posible, se distribuyeron sus responsabilidades. En cada módulo se crearon diferentes clases que heredan de la clase "Actions.class.php" que genera el framework, de esta forma en cada Actions se realizan acciones lo más específicas posible. La figura 2 muestra la estructura del módulo SIC0116 donde las acciones registrar, modificar, visualizar, imprimir y buscar se realizan separadas. Esto garantiza que en lugar de que haya una súper clase que haga todo, haya muchas clases que hagan unas pocas tareas cada una en forma coordinada.

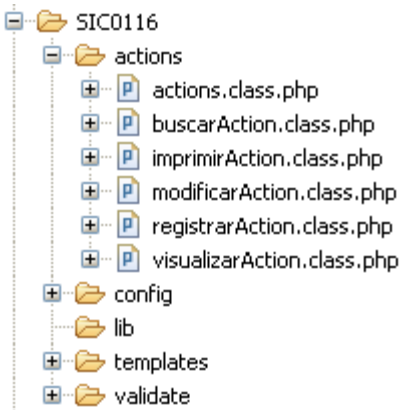


Figura 2: Estructura de los módulos: Módulo SIC0116.

Como el patrón Controlador indica asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas y recomienda dividir los eventos del sistema en el mayor número de controladores, la figura 2 evidencia la utilización no sólo del patrón Alta Cohesión sino también del Controlador y el Bajo acoplamiento. Dividiendo las acciones en varias Actions se ha logrado aumentar la cohesión y disminuir el acoplamiento y en este mismo sentido los eventos que el usuario solicite serán respondido por una acción específica, ya sea registrar, buscar, modificar, imprimir o visualizar.

Patrón GOF utilizado

Dentro de la categoría estructurales en los patrones GOF se encuentra el Decorator. Se ha utilizado este patrón con la implementación de una plantilla global denominada layout. Este archivo es común para toda la aplicación. En la implementación del módulo Análisis Químico se han empleado 3 layout, uno que se le muestra al usuario antes de autenticarse en el sistema, otro que se utiliza para quitar todo el decorado cuando el usuario desea imprimir y por último el layout que permite la navegabilidad del usuario por la aplicación, este se puede observar en el figura 3.

Figura 3: Empleo del layout en Análisis Químico: Layout con menú.

2.4 Diagrama de componentes

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. En él se refleja la relación entre los paquetes de componentes de los diferentes módulos que conforman el LIMS. El módulo tratado: Análisis Químico, dentro del modelo de implementación está compuesto por varios diagramas de componentes. A continuación se explica cómo se estructuran los diagramas de componentes que pertenecen a Análisis Químico.

Componente: Parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces y proporciona la realización de los mismos. Un

componente típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios, etc.).[10]

Los *diagramas de componentes* son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. [10]

Estos diagramas muestran la estructura de alto nivel del modelo de implementación, exponen las dependencias de compilación de los ficheros de código, relaciones de derivación entre ficheros de código fuente y ficheros que son resultados de la compilación, dependencias entre elementos de implementación y sus respectivos elementos de diseños que son implementados.

La Figura 4 responde al diagrama de componentes del CU Gestionar Curva de Calibración de Acido Bicinconícico (SIC-0763C). Su estructura está basada en la arquitectura Modelo Vista Controlador que utiliza Symfony. Cuando el usuario interactúa con la aplicación, el componente *sfFrontWebController,* que ofrece un punto de entrada único a toda la aplicación, determina qué combinación de módulo-acción se ejecutará.

El paquete de componentes “Controlador” contiene la lógica del negocio donde se ubican las cinco acciones básicas que responden a los requisitos definidos para este caso de uso: buscar, imprimir, visualizar, modificar y registrar. Se incorpora además, en el controlador, el componente “Components” que hereda de sfComponents y su función es mejorar la reutilización del código, ha sido muy importante en la implementación la utilización de este componente porque evita tener fragmentos de código repetido en acciones separadas.

Symfony ofrece una alternativa para no repetir código PHP y poder validar las acciones, en este caso registrar y modificar, mediantes archivos YAML que se han ubicado en el diagrama dentro del paquete “YAML”, estos archivos son clases que heredan de sfValidator, es por esto que tienen una relación de dependencia con este subsistema. Para restringir el acceso a las acciones se crea un componente *security* garantizando así que el usuario obligatoriamente se autentique antes de acceder a algunos de los privilegios que ofrece la aplicación.

Symfony proporciona el helper observe_field() para estar a tono con los formularios más modernos que no sólo se encargan de enviar sus datos cuando el usuario pulsa sobre el botón de envío, sino que también pueden reaccionar a los cambios producidos por el usuario sobre alguno de sus campos. Es por ello que se incluye dentro del propio paquete de componentes “Controlador”, otro llamado

"Calculo/Actions" que guarda la lógica del negocio de los helper observe_field(), para dar solución a los cálculos que el usuario necesita en tiempo de ejecución.

Cada vez que el usuario teclee un carácter en el campo "características del material de referencia", se realiza una llamada a la acción remota Utilizar/características. La interacción denominada "características" consiste en un cuadro de texto que muestra una lista de valores relacionados con los caracteres que teclea el usuario. Este efecto se puede conseguir con una única llamada al helper input_auto_complete_tag(). En el diagrama mostrado se ha colocado un componente dentro del paquete "Controlador" que brinda esta funcionalidad.

Terminando con el "Controlador" se pasa ahora a detallar en los elementos ubicados dentro del paquete de componentes denominado "Vista", que se encarga de producir las páginas que se muestran como resultado de las acciones. Se ha colocado el componente "layout_imprimir" con el objetivo de eliminar cualquier tipo de decoración fuera del componente imprimir que utiliza a "print" como un estilo para eliminar los botones en la impresión. El otro "layout" contiene código común para las restantes plantillas, dígase menú principal, banner y otros elementos. Las acciones completas están conformadas por componentes modificar, registrar, buscar o visualizar y su plantilla mostradas en los componentes con los mismos nombres y el sufijo Success. El componente "VisualizarAction.class.php" del paquete de componentes "Controlador" depende totalmente de "BuscarSuccess.php", "RegistrarSuccess.php" y "ModificarSuccess.php" ubicados en la "Vista", esto se debe a que sólo se visualiza una vez registrado, modificado o buscado en este caso de uso. De igual manera el componente "ImprimirAction.class.php" sólo se ejecuta una vez visualizada alguna planilla, es por eso su dependencia de "VisualizarSuccess.php".

Se ha colocado un paquete de componentes dentro del paquete Vista, denominado "components/templates" donde se encuentran los elementos parciales que responden a las acciones definidas en el componente "Components" del controlador. A su vez, el paquete de componentes "Calculo/templates" incluye los componentes que visualizan las acciones de las cuales dependen en el controlador. De igual manera el componente "caracteristicasSuccess.php" visualiza la acción "características.class.php" que se encuentra en el "Controlador".

Necesario para la aplicación, se crea un componente "JS" que garantiza las validaciones en la capa de presentación. Por otra parte, el componente "CSS" incluye los estilos que se le aplican al layout y a todas las plantillas, garantizando así el decorado de las mismas.

El paquete de componentes “Modelo” recoge cuatro componentes que representan clases construidas por el subsistema *Propel de Symfony 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 *Análisis Químico*. Los componentes representados en este diagrama se han utilizado para implementar diversos métodos, cada una de ellas tiene una relación de dependencia con las acciones en los cuáles son llamadas.

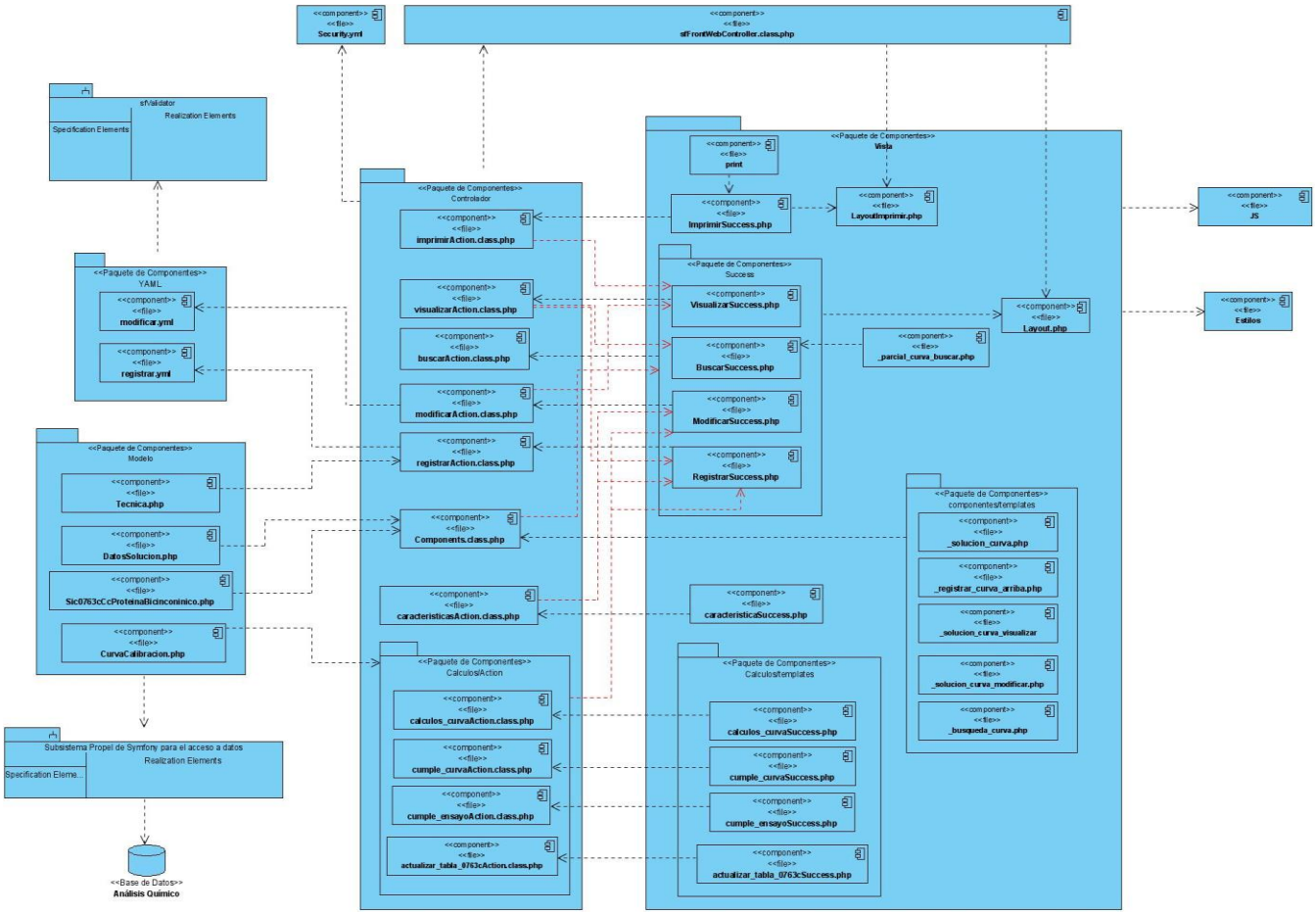


Figura 4: Diagrama de componentes del CU Gestionar Curva de Calibración de Acido Bicinconícico (SIC-0763C).

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 Análisis Químico se ha tenido en cuenta el estilo de código propuesto para la implementación en lenguaje PHP5. Se pueden señalar los siguientes aspectos:

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

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

```
/**
 * SIC0020 actions.
 *
 * @package    Quimico
 * @subpackage SIC0020
 * @author     JORGE C. PUIG
 */
```

2.6 Componentes reutilizables

Muchas de las planillas que gestiona el laboratorio Análisis Químico, de alguna manera, contaban con elementos muy similares. Para implementar casos como estos se utilizaron elementos parciales o componentes, en dependencia de la complejidad de la lógica.

Si el fragmento de código contenía poca lógica, se utilizó un archivo de plantilla al que se le pasaron algunas variables, este archivo se denomina parcial. Generalmente las planillas en las que se pudieron manejar de forma adecuada estos fragmentos de código, contenían en la parte inferior, campos como el nombre de las personas que llenaron esa planilla y la fecha en que se hizo.

La figura 5 refleja el parcial utilizado en los 11 módulos referidos a las curvas de calibración (Conjunto de concentraciones que describen el intervalo en el cuál se deberá cuantificar un compuesto químico)

Figura 5: Parcial inferior de las curvas de calibración: _parcial_curva

En los casos donde la lógica resultaba compleja; se tenía que acceder a los datos del modelo, fue necesario utilizan componentes, separando la presentación de la lógica. Casi la totalidad de las planillas utilizaban materiales críticos, equipos e instrumentos de medición y soluciones, estas últimas variando en dependencia de la planilla donde se encontraba el usuario.

La figura 6 refleja la presentación del componente antes mencionado que se guarda en un elemento parcial que permitió reducir considerablemente las líneas de código. Una de las funciones de esta interfaz dinámica es insertar nuevos campos o eliminarlos (entiéndase: equipos, materiales, soluciones) esto se logró con la utilización del helper update_element_function(). De no haber utilizado este componente se hubiese repetido el código innecesariamente en el desarrollo de 25 requisitos.

Figura 6: Presentación del componente: _gestionar_curva

En función del diseño y la implementación realizada en la capa de la vista, se emplearon más de 15 componentes en la implementación del módulo. De esta forma, queda evidenciada la reutilización del código.

2.7 Integración con AJAX

AJAX parece ser la palabra de moda en el "mundo" del desarrollo de aplicaciones Web. En realidad, no es una tecnología, sino la unión de varias tecnologías que juntas pueden lograr cosas realmente

impresionantes. “AJAX es el acrónimo para Asynchronous JavaScript + XML y el concepto es: Cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en background, los datos que son usados para actualizar la página solo re-renderizando la página y mostrando u ocultando porciones de la misma”. [26]

Con el objetivo de ganar en usabilidad, interactividad y rapidez en las aplicaciones web, Symfony incluye varios helpers de Ajax, algunos de ellos se utilizaron en la implementación del módulo Análisis Químico.

Con el empleo de un botón de envío tradicional, si un usuario realiza una búsqueda, la página se actualiza completamente y se pierden los parámetros de búsquedas especificados. Con la utilización del helper submit_to_remote() se logra actualizar sólo el fragmento de la página que se especifique. Este botón de envío Ajax fue útil en todas las búsquedas que se implementaron en el módulo Análisis Químico.

En varias planillas de la aplicación se necesitaba realizar cálculos que determinaban resultados de interés para los analistas del CIGB. La ejecución de una función remota cada vez que el usuario cambiaba el valor de un campo mediante el helper observe_field() posibilitó al usuario ir valorando los resultados de los cálculos a medida que cambiaba los valores de entrada, sin necesidad de enviar los datos con un botón de envío.

El helper input_auto_complete_tag() posibilitó autocompletar valores almacenados en la base de datos, relacionados con los caracteres que teclee el usuario. Este helper realiza una llamada remota una acción específica que devuelve una lista de valores de acuerdo a lo sugerido por el usuario. El resultado se muestra en forma de una lista de elementos HTML y si el usuario selecciona uno de estos el cuadro de texto del helper adquiere este valor.

2.8 Representación del código principal

A continuación se muestran fragmentos de código de un componente que permitió optimizar la implementación del módulo Análisis Químico. La lógica del componente se guarda en una clase que hereda de sfComponents, en esta se formula la acción que se va a ejecutar. Los valores que se envían al parcial dependen del módulo donde se encuentra el usuario, esto se ha garantizado con la utilización del un switch dentro de la acción Busqueda_curva().

```
class ComponentesComponents extends sfComponents
```

```
{
public function executeBusqueda_curva()
  {

$modulo = sfContext::getInstance()->getModuleName();
$nombre_modulo = strtolower($modulo);
switch ($nombre_modulo)
{
case 'sic0112c':
$material = CaractMaterialReferencia::DevolverNombreMaterialReferencia0112C();
$nombre_curva = "CURVA DE CALIBRACI&Oacute;N DE  CARBOHIDRATOS";
break;
case 'sic0113c':
$material = CaractMaterialReferencia::DevolverNombreMaterialReferencia0113C();
$nombre_curva = "CURVA DE CALIBRACI&Oacute;N DE L&Iacute;PIDOS";
break;
case 'sic0116c':
$material = CaractMaterialReferencia::DevolverNombreMaterialReferencia0116C();
$nombre_curva = "CURVA DE CALIBRACI&Oacute;N DE TIOCINATO";
break;
case 'sic0759c':
$material = CaractMaterialReferencia::DevolverNombreMaterialReferencia0759C();
$nombre_curva = "CURVA DE CALIBRACI&Oacute;N DE GRUPOS SULFIH&Iacute;DRILOS";
break;
case 'sic0760c':
$material = CaractMaterialReferencia::DevolverNombreMaterialReferencia0760C();
$nombre_curva = "CURVA DE CALIBRACI&Oacute;N DE &Aacute;CIDO SI&Aacute;LICO";
break;
case 'sic0761c':
$material = CaractMaterialReferencia::DevolverNombreMaterialReferencia0761C();
$nombre_curva = "CURVA DE CALIBRACI&Oacute;N DE TWEEN";
break;
case 'sic0762c':
$material = CaractMaterialReferencia::DevolverNombreMaterialReferencia0762C();
$nombre_curva = "CURVA DE CALIBRACI&Oacute;N DE ORCINOL";
break;
case 'sic0763c':
$material = CaractMaterialReferencia::DevolverNombreMaterialReferencia0763C();
$nombre_curva = "CURVA DE CALIBRACI&Oacute;N DE &Aacute;CIDO BICINCON&Iacute;NICO";
break;
case 'sic1001c':
$material = CaractMaterialReferencia::DevolverNombreMaterialReferencia1001C();
$nombre_curva = "CURVA DE CALIBRACI&Oacute;N DE F&Oacute;SFORO";
break;
case 'sic0710c':
$material = CaractMaterialReferencia::DevolverNombreMaterialReferencia0710C();
$nombre_curva = "CURVA DE CALIBRACI&Oacute;N DE  BIURET";
break;
case 'sic0107c':
$material = CaractMaterialReferencia::DevolverNombreMaterialReferencia0107C();
$nombre_curva = "CURVA DE CALIBRACI&Oacute;N DE LOWRY";
break;
}
   $this->material= $material;
   $this->nombre_curva=$nombre_curva;
```

```
}
```

Las variables **$material** y **$nombre_curva** pasan a la capa de presentación del componente que se guarda en un elemento parcial. El siguiente fragmento de código es el parcial que responde a la ejecución del fragmento de código anterior.

```
<table class="tabla">
   <tr>
     <td>
      CENTRO DE INGENIER&Iacute;A GEN&Eacute;TICA Y BIOTECNOLOG&Iacute;A
      DIRECCI&Oacute;N DE CALIDAD
    <td>
  </tr>
 <tr>
      <td><?php echo $nombre_curva ?></td>
</tr>
</table>
<hr />
<table class="tabla">
 <tr>
   <td PAR&Aacute;METROS DE LA B&Uacute;SQUEDA</td>
  </tr>
</table>

<table class="tabla">
       <tr>
         <td >Nombre del material de referencia empleado:
<?php echo
select_tag('Nombre_solucion_referencia',options_for_select($material))?>
        </td>
      </tr>
</table>

 <table class="tabla">
      <tr >
         <td>Fechas del Proceso: </td>
      </tr>
     <tr>
        <td> Desde<?php echo input_date_tag('Inicio', 'now', 'rich=true')?></td>
        <td Hasta<?php echo input_date_tag('Final', 'now', 'rich=true')?> </td>
    </tr>
</table>

<table class="tabla">
<tr>
   <td >
       <?php echo submit_tag('Buscar') ?>
       <?php echo button_to('Cancelar','Principal/Index')?>
   </td>
</tr>
</table>
<hr />
```

Para llamar al componente anterior, en las plantillas de los módulos donde se utilizó, se escribe la siguiente línea de código:

```
<?php include_component('Componentes', 'busqueda_curva') ?>
```

2.9 Análisis de la Seguridad del módulo

En el módulo Análisis Químico, para que una acción pueda ejecutarse tiene que pasar primero por un filtro especial que controla si el usuario que va a navegar en la aplicación tiene los privilegios necesarios para ejecutar la acción deseada.

Se ha garantizado la seguridad de forma tal que sólo los analistas registrados en este módulo sean los autorizados a acceder a las planillas que gestiona el mismo, en este sentido, todas las acciones requieren que los usuarios estén autenticados.

Para restringir el acceso a una acción se crea un archivo de configuración YAML llamado security.yml en el directorio config/ del módulo. En este archivo, se especifican los requerimientos de seguridad que los usuarios deben satisfacer para cada acción o para todas las acciones. Esta constituye otra de las bondades de Symfony, trabaja de forma tal que cuando un usuario intenta acceder a una acción restringida se verifiquen si es analista del módulo Análisis Químico:

- Si el usuario está autenticado entonces la acción se ejecuta.
- Si el usuario no está autenticado, es re-direccionado a la acción de login.

La seguridad es un módulo que debe trabajarse más. Lo que se ha realizado hasta el momento no explota todas las funcionalidades que brinda el framework para este tema pero lo realizado garantiza los requerimientos necesarios para que sólo las personas autorizadas puedan acceder a la aplicación.

2.10 El Framework de pruebas Lime

Las pruebas no pueden asegurar la ausencia de defectos; sólo pueden demostrar que existen defectos en el software. Involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados. Las condiciones controladas pueden ser normales o anormales. La prueba puede intencionalmente esforzar al programa y producir errores en las respuestas para determinar si los sucesos ocurren cuando no tendrían que ocurrir o cuando los hechos no suceden cuando deberían suceder.

El método de prueba utilizado para detectar defectos al software desarrollado fue el de de Caja Blanca. Permitió asegurar que las operaciones internas se ajustan a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

Haciendo uso intensivo de las técnicas de pruebas de Caja Blanca, las pruebas de unidad verificaron los caminos de control importantes y permitieron descubrir errores dentro del ámbito del módulo. Se utilizó la descripción del diseño como guía.

Symfony incluye un framework llamado Lime que permitió crear las pruebas unitarias. Las pruebas de Lime son fáciles de leer y sus resultados también lo son. Está escrito con PHP, es muy rápido y está bien diseñado internamente. Consta solamente de un archivo, denominado lime.php, y no tiene ninguna dependencia.

En el directorio test/unit de la aplicación se encuentran los archivos que responden a las pruebas realizadas al software. Se ha estructurado de forma tal que cada archivo comienza con el nombre del método al que se le ha realizado la prueba y termina con Test.php. Su sintaxis es sencilla y fácil de leer.

Se realizaron varias pruebas a los métodos donde se calcula la pendiente, intercepto y coeficiente de correlación en las curvas de calibración, antes mencionadas. Desde la clase SIC0112CTest.php se llama al objeto lime_test de Lime para realizar las pruebas, la clase es la siguiente:

```php
<?php
require_once dirname( __FILE__ ) . '/../bootstrap/unit.php';
$curva = new CurvaCalibracion();

$t=new lime_test(3,new lime_output_color( ));
$t->is($curva->CalcularPendiente( 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 ),'2', 'Resultado de la Pendiente_1' );
$t->is($curva->CalcularPendiente( 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1),'2','Resultado de la Pendiente_2');
$t->is($curva->CalcularPendiente( 5,5,5,5,5,0,0,0,0,0,1,1,1,1,1),'0','Resultado de la Pendiente_3' );
?>

```

Figura 7: Fragmento de código de la clase SIC0112Test.php.

En la clase se hace referencia al método “CalcularPendiente ()” que utiliza valores de concentraciones y absorbancias para determinar la pendiente de una curva de calibración. La tabla 1 muestra los datos de entrada, los valores esperados y el método de prueba utilizada, en este caso 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
CalcularPendiente(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)	IS	2	2
CalcularPendiente(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)	IS	2	0(Warning)
CalcularPendiente(5,5,5,5,5,0,0,0,0,0,1,1,1,1,1)	IS	0	0

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

Una vez terminada la prueba se pudieron detectar errores en la implementación del método, no se había validado la posibilidad de la indefinición del denominador en el cálculo, la segunda prueba posibilitó la corrección de este error, el resultado de Lime en el Test 1 fue el siguiente.

```
D:\USERS\Alberto\Quimico>symfony test-unit SIC0112C
1..3
ok 1 - Resultado de la Pendiente_1

Warning: Division by zero in D:\USERS\Alberto\Quimico\lib\model\CurvaCalibracion
.php on line 50
not ok 2 - Resultado de la Pendiente_2
#     Failed test (.\test\unit\SIC0112CTest.php at line 11)
#            got: 0
#       expected: '2'

Warning: Division by zero in D:\USERS\Alberto\Quimico\lib\model\CurvaCalibracion
.php on line 50
ok 3 - Resultado de la Pendiente_3
 Looks like you failed 1 tests of 3.
```

Figura 8: Prueba de unidad Lime: Test 1.

Después de terminada la prueba 1, se realiza la corrección del defecto en el código. Con valores esperados para las tres pruebas (2, 0, 0) respectivamente, por segunda vez se verifica con Lime, la figura 9 muestra los resultados.

```
D:\USERS\Alberto\Quimico>symfony test-unit SIC0112C
1..3
ok 1 - Resultado de la Pendiente_1
ok 2 - Resultado de la Pendiente_2
ok 3 - Resultado de la Pendiente_3
 Looks like everything went fine.
```

Figura 9: Prueba de unidad Lime: Test 2.

2.11 Conclusiones

Partiendo de los requisitos funcionales y no funcionales y luego de analizar la arquitectura Modelo Vista Controlador definida por Symfony, se realizan los artefactos generados por el flujo de trabajo

implementación y se culmina con la implementación. Desde un principio se utilizó un estándar de codificación que garantizó mayor entendimiento del código. Con la reutilización de componentes se aseguró un alto nivel de eficiencia en el trabajo, consumiendo así algunas de las ventajas que ofrece este Framework.

El diagrama de despliegue permitió una visión más abarcadora de la distribución física del sistema. Se valida la seguridad del sistema utilizando las facilidades que brinda el Framework. Por último se puede asegurar que las pruebas de unidad de Lime, posibilitó la corrección de defectos y permitieron comprobar diferentes métodos utilizados.

CONCLUSIONES

Culminada la implementación de 28 Casos de Usos del Módulo Análisis Químico del Sistema para la Gestión de la Información de Laboratorios de la Dirección de Calidad del Centro de Ingeniería Genética y Biotecnología; se concluye lo siguiente:

- El estudio de la documentación obtenida en las iteraciones anteriores realizadas al módulo Análisis Químico, permitió un mejor entendimiento de los procesos realizados en el CIGB que facilitaron, en gran medida, la implementación.
- La elaboración de los diagramas de componentes como parte de la ingeniería de software garantizó documentación para próximas versiones del producto y mejor estructura del trabajo realizado.
- La utilización del Framework Lime para la realización de las pruebas de unidad posibilitó detectar y corregir errores en algunos fragmentos de código.
- Se obtuvo un producto funcional para el módulo Grupo de Análisis Químico del LIMS de Calidad del CIGB.
- Se da cumplimiento a la implementación de 28 casos de uso del módulo Análisis Químico para el Sistema de Gestión de Información de los Laboratorios de la Dirección de Calidad del CIGB.

RECOMENDACIONES

- Se continúe con la implementación de los restantes requisitos funcionales que exige el cliente en el módulo Análisis Químico.
- Integrar el módulo Análisis Químico con los restantes módulos del LIMS para lograr un producto más completo.

REFERENCIAS BIBLIOGRÁFICAS

1. CIGB. (14 de 11 de 2008). *Centro de Ingeniería Genética y Biotecnología*. Recuperado el 14 de 11 de 2008, de Centro de Ingeniería Genética y Biotecnología: http://www.cigb.edu.cu/index.php?option=com_content&task=view&id=77&Itemid=146

2. GURETEX. GURETEX. *GURETEX.* [En línea] [Citado el: 2 de marzo de 2009.] www.guretek.net/lims_docu_1.htm..

3. LABWARE SOLUTIONS. *LABWARE SOLUTIONS.* [En línea] [Citado el: 2 de marzo de 2009.] http://www.labware.com/LWWebES.nsf/lp/es04.

4. Innovatek Ltda. *Innovatek Ltda.* [En línea] http://www.innovatek.com.co/ootb.htm.

5. MODELICA,el nuevo estándar de lenguaje. *MODELICA,el nuevo estándar de lenguaje.* [En línea] [Citado el: 8 de junio de 2009.] http://74.125.47.132/search?q=cache:K5XLRQ1e2GQJ:www.forumtecnologico.net/descargas/REVDescarga/ft_08.pdf+Las+aplicaciones+Matrix+LIMS+solucionan+las+necesidades+de+grandes+empresas+y+tambi%C3%A9n+de+organizaciones+de+tama%C3%B1o+peque%C3%B1o+o+mediano&cd.

6. BIKA Jabs System. [En línea] 2005-2009. http://www.bikalabs.com/.

7. Open Source laboratory automation &informatics. [En línea] http://www.labmatica.com/.

8. Roman Konertz, PD Dr. Ludwig Eichinger and Dr. Budi Tunggal. OpenLIMS. [En línea] http://www.open-lims.org/.

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

10. Departamento de Ingeniería de Software. Teleformación. Teleformación. [En línea] 6 de junio de 2009. [Citado el: 6 de junio de 2009.] http://eva.uci.cu/mod/resource/view.php?id=14094

11. JACOBSON, IVAR, BOOCH, GRADY and RUMBAUGH, JAMES. 25. Departamento de Sistemas Informáticos. *Departamento de Sistemas Informáticos.* [Online] abril 2009, 25. [Cited: abril 2009, 25.] http://www.dsi.uclm.es/asignaturas/42541/pdf/M1tema4.pdf.

12. Spark System. Spark System. [En línea] [Citado el: 6 de junio de 2009.] http://www.sparxsystems.com.ar/resources/tutorial/physical_models.html.

13. NEXOS SOFTWARE. *NEXOS SOFTWARE.* [En línea] [Citado el: 2 de marzo de 2009.] http://www.nexos-software.com.co/Articulo_22.htm.

14. AUTOR BERTINO, ELISA y AUTOR MARTINO, LORENZO. *Sistemas de bases de datos orientado a objetos.* s.l. : Ediciones Díaz de Santos, 1995.

15. APRENDE. *APRENDE.* [En línea] [Citado el: 2 de marzo de 2009.] http://www.eaprende.com/gestor-de-basededatos-mysql-postresql-sqlite.html

16. Dominicano, Fundación Código Libre. *Introducción a Base de datos con PostgreSQL.* República Dominicana : s.n., 2009.

17. Fundación Código Libre Dominicana. Fundación Código Libre Dominicana. [En línea] [Citado el: 6 de junio de 2009.] http://www.codigolibre.org/libro/Howto/apache_v2.pdf.

18. TODOS, LINUX PARA. LINUX PARA TODOS. *LINUX PARA TODOS.* [En línea] [Citado el: 2 de marzo de 2009.] http://www.linuxparatodos.net/portal/staticpages/index.php?page=como-apache

19. Potencier, Fabien and Zaninotto, François. *Symfony, la guía definitiva.* s.l. : Apress, 2008.

20. PHP, SITIO. MANUAL PHP. *MANUAL PHP.* [En línea] [Citado el: 2 de marzo de 2009.]

    http://www.php.net/manual/es/preface.php

21. Mora, Roberto Canales. Adictos al trabajo. Adictos al trabajo. [En línea] 22 de 12 de 2003. [Citado el: 10 de mayo de 2009.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=grasp.

22. Gracia, Joaquin. 25. Ingenieros de Software. *Ingenieros de Software.* [Online] mayo 2005, 25. [Cited: abril 2009, 16.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

23. Garzas, Javier. La Arquitectura Software. La Arquitectura Software. [En línea] [Citado el: 6 de junio de 2009.] http://jgarzas.googlepages.com/4mas1.

24. Departamento de Ingeniería de Software, UCI. Teleformación. Teleformación. [En línea] [Citado el: 6 de junio de 2009.] http://eva.uci.cu/mod/resource/view.php?id=14111.

25. Lamas, Yalina and Tamarit, Yadira. *Sistema para la Gestión de la Información de los Laboratorios de la Dirección de Calidad del Centro de Ingeniería.* Ciudad de la Habana : s.n., 2008.

26. Amartino, Mariano. Denkenyber. Denkenyber. [En línea] 28 de marzo de 2005. [Citado el: 5 de mayo de 2009.] http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php.

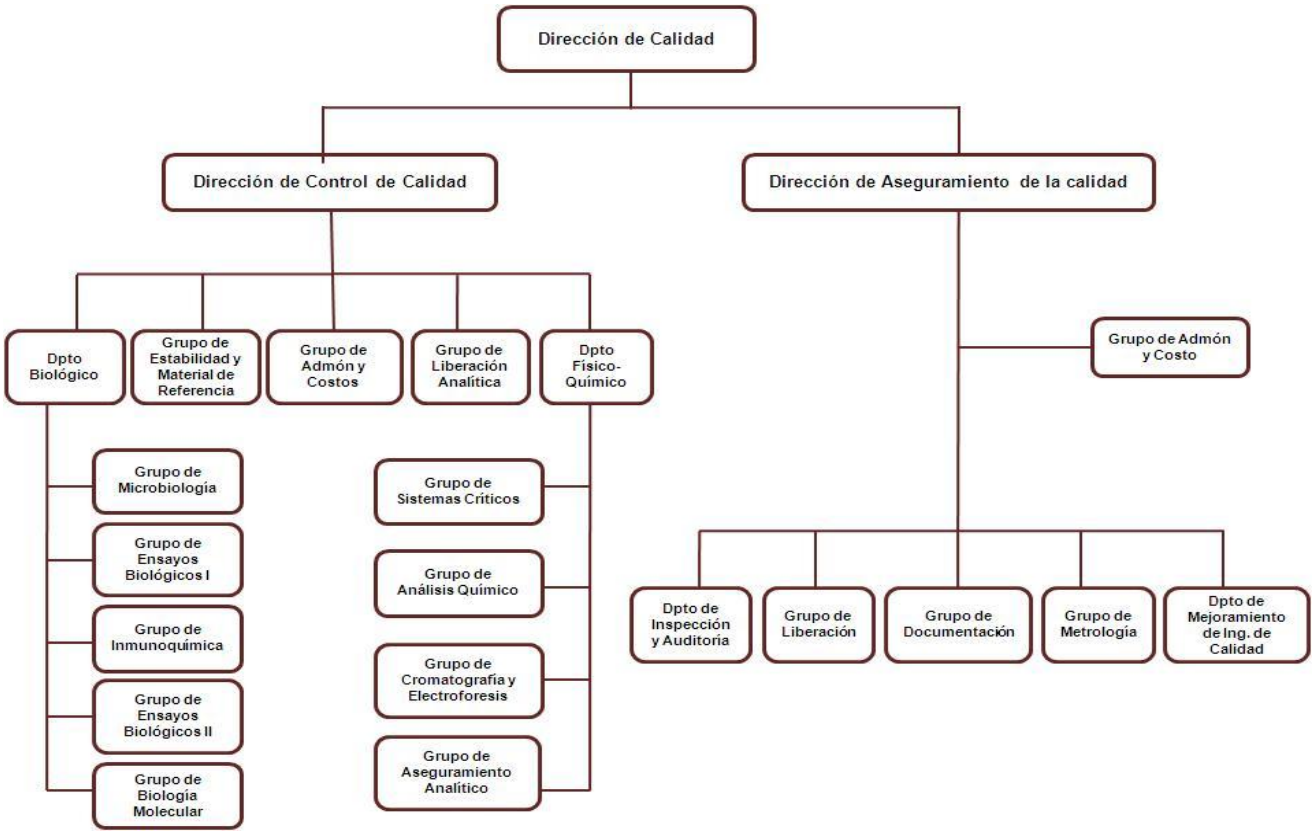
BIBLIOGRAFÍA

- Potencier, Fabien y Zaninotto, François. *Symfony, la guía definitiva.* s.l. : Apress, 2008.
- Harris, Daniel C. Análisis Químico Cuantitativo. *Análisis Químico Cuantitativo.* s.l. : Editorial Reverté.
- 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.
- Jacobson*. El proceso Unificado de desarrollo de software*. La Habana : Félix Varela, 2004.
- Scientiae, Elfos. 1997. *Cuantificación de Proteía absorbida en gel de hidróxido de Aluminio*. 1997. págs. 111-113. Vol. 2.
- Naranjo, Mariela y Machado, Maylen. 2009. *Caracterización preliminar de antígenos reconocidos por sueros de individuos cacunados con vax-SPIRAL en preparaciones de membrana externa de Leptospira pomona mozdok.* Ciudad de la Habana : Centro de Investigación y producción de vacunas, 2009.
- Pizarro, Pablo. 2006. Arquitectrua de Software. Arquitectrua de Software. [En línea] 23 de mayo de 2006. [Citado el: 10 de abril de 2009.] http://arquitectura-de-software.blogspot.com/2006/05/orm-object-relational-mapping-ii-parte.html.
- CIGB. (14 de 11 de 2008). Centro de Ingeniería Genética y Biotecnología. Recuperado el 14 de 11 de 2008, de Centro de Ingeniería Genética y Biotecnología: http://www.cigb.edu.cu/index.php?option=com_content&task=view&id=77&Itemid=146
- GURETEX. GURETEX. GURETEX. [En línea] [Citado el: 2 de marzo de 2009.] www.guretek.net/lims_docu_1.htm..
- LABWARE SOLUTIONS. LABWARE SOLUTIONS. [En línea] [Citado el: 2 de marzo de 2009.] http://www.labware.com/LWWebES.nsf/lp/es04.
- AUTOR BERTINO, ELISA y AUTOR MARTINO, LORENZO. Sistemas de bases de datos orientado a objetos. s.l. : Ediciones Díaz de Santos, 1995.
- NEXOS SOFTWARE. NEXOS SOFTWARE. [En línea] [Citado el: 2 de marzo de 2009.] http://www.nexos-software.com.co/Articulo_22.htm.

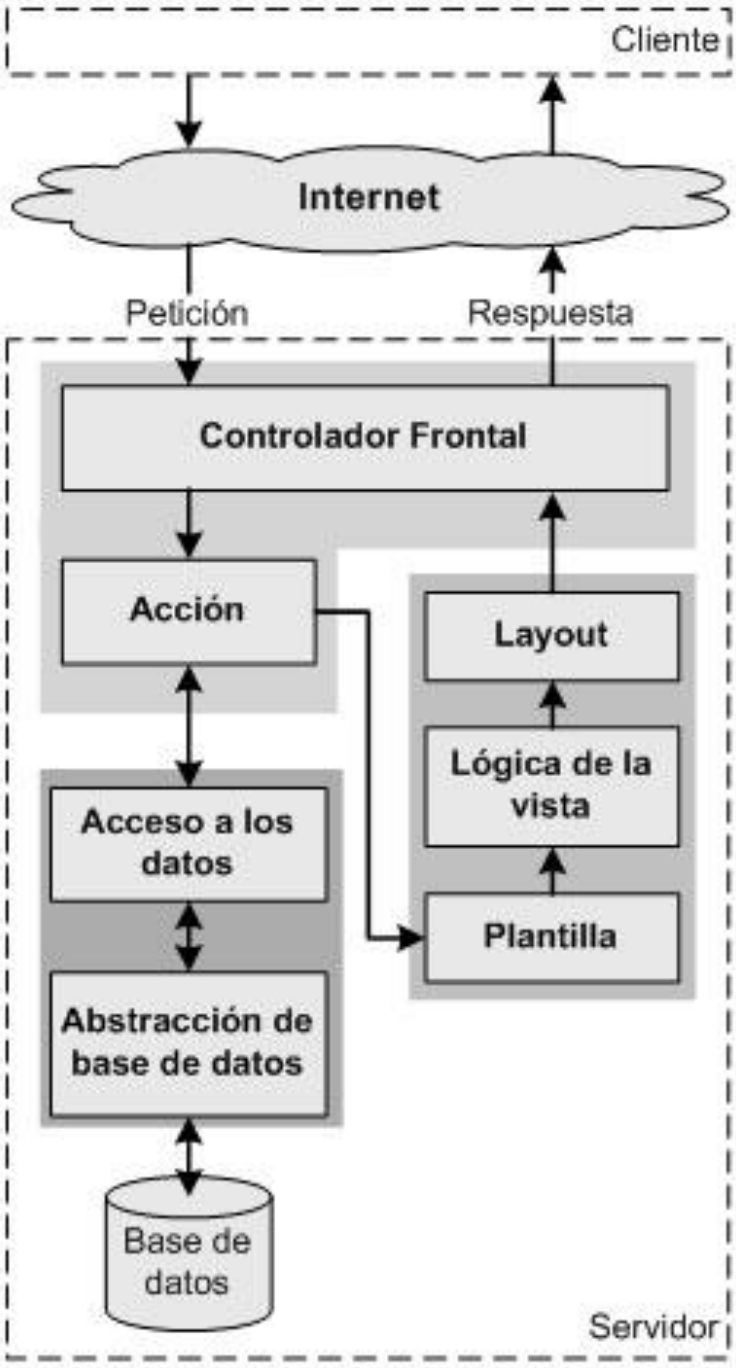
- APRENDE. APRENDE. [En línea] [Citado el: 2 de marzo de 2009.] http://www.eaprende.com/gestor-de-basededatos-mysql-postresql-sqlite.html .
- Dominicano, Fundación Código Libre. Introducción a Base de datos con PostgreSQL. República Dominicana : s.n., 2009.
- BIKA Jabs System. [En línea] 2005-2009. http://www.bikalabs.com/.
- Open Source laboratory automation &informatics. [En línea] http://www.labmatica.com/.
- Roman Konertz, PD Dr. Ludwig Eichinger and Dr. Budi Tunggal. OpenLIMS. [En línea] http://www.open-lims.org/.
- PHP, SITIO. MANUAL PHP. MANUAL PHP. [En línea] [Citado el: 2 de marzo de 2009.] http://www.php.net/manual/es/preface.php.
- TODOS, LINUX PARA. LINUX PARA TODOS. LINUX PARA TODOS. [En línea] [Citado el: 2 de marzo de 2009.] http://www.linuxparatodos.net/portal/staticpages/index.php?page=como-apache
- Gracia, Joaquin. 25. Ingenieros de Software. Ingenieros de Software. [Online] mayo 2005, 25. [Cited: abril 2009, 16.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.
- JACOBSON, IVAR, BOOCH, GRADY and RUMBAUGH, JAMES. 25. Departamento de Sistemas Informáticos. Departamento de Sistemas Informáticos. [Online] abril 2009, 25. [Cited: abril 2009, 25.] http://www.dsi.uclm.es/asignaturas/42541/pdf/M1tema4.pdf.
- Robles, Ing. V. Lenguajes de scripting ¿Una nueva forma de programar? Ecuador : s.n., 2007.

ANEXOS

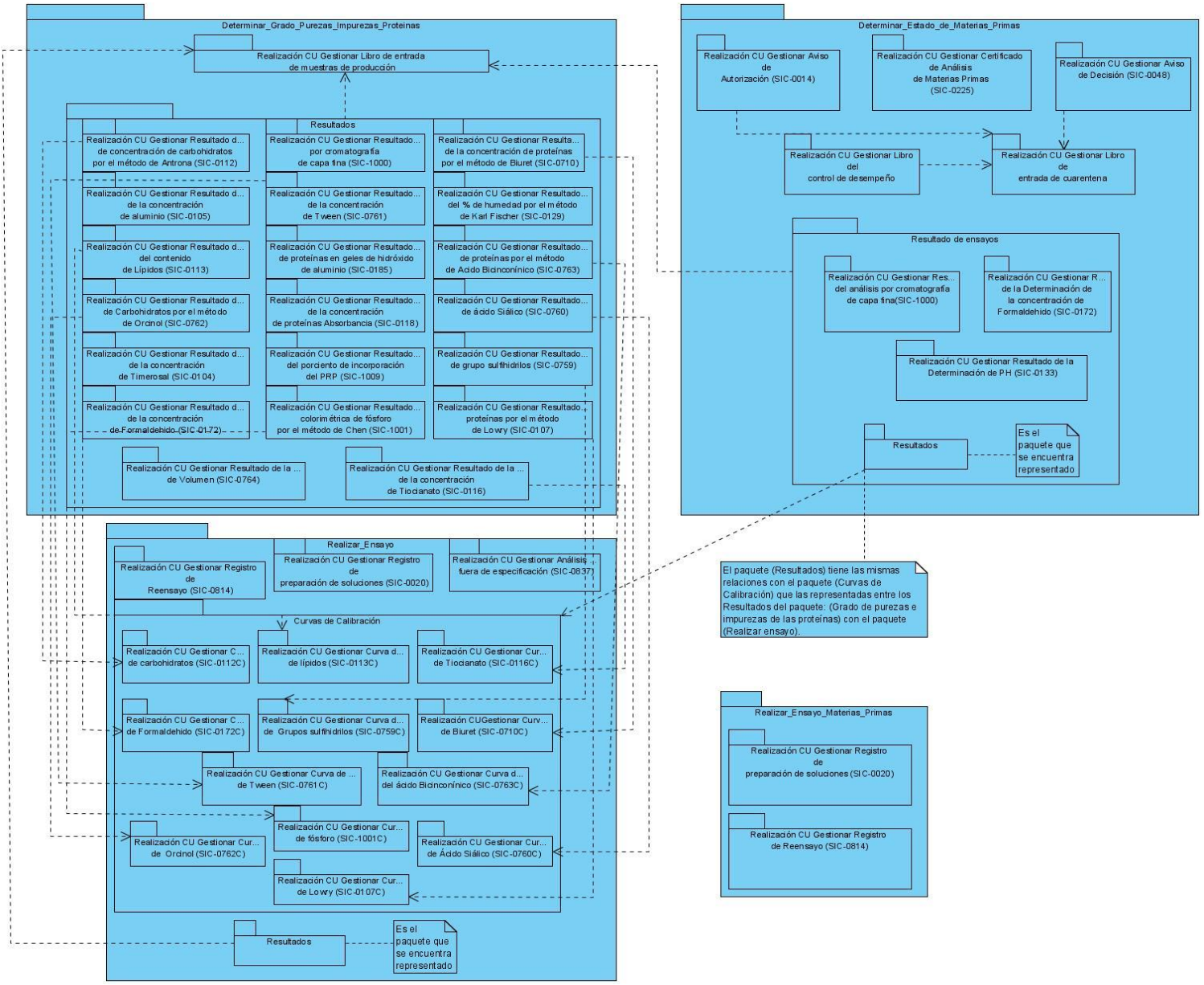
Anexo 1: Estructura jerárquica del Área de Calidad del Centro de Ingeniería Genética y Biotecnología



Anexo 2: Patrón: Modelo-Vista-Controlador.



Anexo 3: Vista lógica



Nota: *La explicación detallada de la vista lógica se encuentra en la siguiente dirección:* http://10.34.19.22:3389/svn/alasLIMSC/docs/Expediente%20de%20Proyecto/Ingenieria/Modulos/Analisis%20Quimico/7%20Tesis/

GLOSARIO DE TÉRMINOS

Arquitectura de Software: conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

Aseguramiento de la Calidad: parte de la gestión de la calidad orientada a proporcionar confianza en que se cumplirán los requisitos de la calidad.

AQ: Análisis Químico.

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

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

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

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

CSS: lenguaje artificial usado para definir la presentación de un documento estructurado escrito en HTML o XML

Curva de Calibración: función que describe las respuestas del detector a lo largo de un rango de concentraciones y es utilizada para predecir la concentración de una muestra desconocida, basándose en la respuesta del detector.

Ensayo: es la aplicación de un análisis a una o varias muestras.

GPL: (del inglés General Public License) licencia pública general. Es una licencia que está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

IDE: entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Lenguaje de Scripting: es un lenguaje de programación que se utiliza para manipular, adaptar y automatizar una página web. En los sistemas en los que se aplica un script ya existe una interfaz de usuario que funciona perfectamente sin necesidad de dichos lenguajes. Lo que hacen los scripts es servir como mecanismo para controlar con mayor precisión sus funcionalidades.

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

Plone: sistema de gestión de contenidos o CMS por sus siglas en inglés (Content Management System), basado en Zope. Es un desarrollo basado en código abierto.

Producto: es el resultado de una investigación al que finalmente se le establecen dos nombres: uno genérico y otro comercial. Está destinado al uso que su investigador haya propuesto y definido.

SIC: sistema de Información y control.

SSL: protocolo criptográfico que proporcionan comunicación segura por una red.

TCP/IP: protocolos más importantes de la familia de protocolos de internet. Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP).

Trigger: es un disparador en una Base de datos, es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

USB: puerto que se utiliza para conectar periféricos a una computadora.

Validación: acción documentada que demuestra, de acuerdo con los principios de las buenas prácticas de fabricación, que cualquier procedimiento, proceso, equipo, material, actividad o sistema realmente brinda los resultados esperados.

Vista: es una presentación de un modelo, la cual es una descripción completa de un sistema desde una particular perspectiva (Kruchten, 1995).

XML: lenguaje de marcas extensible, del inglés Extensible Markup Language. Es considerado como un metalenguaje de definición de documentos estructurados mediante marcas o etiquetas.

Zope: servidor de aplicaciones web de código abierto escrito en el lenguaje de programación Python.