Universidad de las Ciencias Informáticas
FACULTAD 6

Aplicación de escritorio del Sistema de Entrega Digital

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Nilmar Sánchez Muguercia

Tutora: Ing. Yordanys Piñeiro Gómez

Co-tutor: Lic. Omar Mar Cornelio

La Habana, Marzo de 2013

"Año 55 de la Revolución"

Teniendo la no limitación como límite.

Bruce Lee

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio.

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

Nilmar Sánchez Muguercia

_____

Firma del Autor

Ing. Yordanys Piñeiro Gómez

_____

Firma del Tutor

DATOS DE CONTACTO

Ing. Yordanys Piñeiro Gómez

Ing. en Ciencias Informáticas. Graduado en el año 2007, con 5 años de experiencia y categoría docente de Asistente.

Correo electrónico: ypineirog@uci.cu

Lic. Omar Mar Cornelio

Lic. en Educación en la especialidad de Informática. Graduado en el año 2006, con 6 años de experiencia y categoría docente de Asistente.

Correo electrónico: omarmar@uci.cu

AGRADECIMIENTOS

Quisiera agradecer de forma muy especial a mi familia. A mi mamá, mi gorda por aguantar 9 meses y luego 26 años conmigo a cuestas, por formarme y convertirme en la persona que hoy soy, aunque sé que en el fondo sigo siendo tu nene lindo. A mi papá, mi gordo, por pasarme los genes que formaron la persona adicta por veces a la perfección, por todos los consejos que me distes, pues dicen que soy una copia tuya, hasta caminando. A abuela Elena: Pepe, por escucharme y comprenderme cuando hacía falta, por aconsejarme de la manera más tierna posible, todavía siento que me mimas, que rico. A abuela Elodia, que aunque no está entre nosotros, siempre fue la que estuvo al tanto del niño y sus cosas, por ser mi caballito cuando pequeño. A Abuelo Ca, por formar en gran medida el hombre que soy, por sus consejos que a pesar de ser pequeño calaron hondo, por estar siempre presente en mi pensamiento, por empezar a esculpir ese diamante y no dejar que se convirtiera en carbón. Realmente es una lástima que no estés, eres la persona a la que más me hubiera gustado hacer sentir orgulloso. Quizás no te lo haya podido decir nunca, pero te extraño mucho y me haces mucha falta.

Agradezco a la mujer que me ha presentado los mayores retos de mi vida, solo para darme cuenta que siempre, si uno se lo propone puede salir adelante, por ayudarme a conocerme más allá del límite. A ella y a su pedacito, por robarme sonrisas cuando más lo necesitaba, a mi novia Arletys y la princesita Brenda.

A mi tutora Yordanys por la paciencia y el aguante con este carácter mío, aunque no lo creas, te entiendo profe, fuiste toda una amiga por sobre todo, además de tutora. A mi co-tutor Omar, por darme la posibilidad de ingresar a Informatización y presentarme el mayor reto si de aplicaciones se trata.

A mi grupo: los 35 iniciales, una familia que se ha dispersado, pero con los cuales pasé muy buenos momentos. A todos los técnicos junto a los cuales inicié en el Docente 4, los mejores momentos fueron allí. Al piquete de Informatización, a Jeny por ser la madre consejera, con mano firme pero tierna, el Vity por ser el socio de compartir calamidades de responsabilidades. A Eddy, Susana y el Vla, por hacerme pasar los momentos más graciosos. A Abelito por

AGRADECIMIENTOS

enseñarme a escuchar cuando se discute. A Juan Ka por demostrarme que se puede ser un desastre organizado. A Yasse por las conversaciones de fiestas y chicas. A Aquiles por compartir el aula además del trabajo. A Vanoty por el impulso en Linux. A Noe porque fue la primera mujer que me demostró que las féminas también programan y lo hacen muy bien, te admiro.

Agradezco a todas aquellas personas que confiaron en mí, y a los que no confiaron por hacerme esforzar cada vez más cuando quería lograr algo. A los muchachos de la graduación del 2010 de la facultad 9. Al centro DATEC, por ser mi trabajo en los últimos meses de la carrera y brindarme apoyo prestándome una computadora para la tesis, de forma especial a Niurka.

A mis profesores, que quizás en el momento no lo valoramos pero el conocimiento que nos pasaron es muy valioso. A este tribunal por escucharme y no aburrirse, que reto.

A la Oficina de Asuntos Históricos del Consejo de Estado, de la que me siento también hijo, pues las tías y personas que me quieren me sobran allí. Al piquete de los Hardboys del tecnológico, menos mal que aún seguimos en contacto, ahí nació el Bomby. A las tatas, que aunque ya no están aquí, son las amigas que nunca imaginé, mi pulguita Daylin y mi pekis Teresa, Yahima porque siempre confiaron en mí.

Agradezco al destino, por como ha sido conmigo, pues las situaciones que se me presentaron son las que me han llevado a este momento, pues después de mucho pensar, trato de sacar algo positivo de lo negativo que pudo suceder en el camino.

DEDICATORIA

A mi familia, por confiar en mí, por apoyarme con amor infinito en todos los retos que he afrontado en mi carrera.

A todas aquellas personas que me quieren y me desean bien.

RESUMEN

Con el creciente desarrollo de las tecnologías computacionales, se hace imprescindible poner las mismas a disposición del avance en el desarrollo económico y social. En Cuba, esta tarea se lleva a cabo en diversos sectores, ejemplo de ello es la Universidad de las Ciencias Informáticas (UCI), donde la puesta en marcha del Sistema de Entrega Digital (SED), brinda la posibilidad de traspasar la responsabilidad material de los laboratorios de un técnico a otro. Dicho sistema está desarrollado sobre plataforma Web, de esta forma no ofrece la posibilidad de brindar servicios ante la ausencia de conexión de red. La presente investigación tiene como objetivo desarrollar una versión de escritorio del SED para contribuir a la continuidad de los procesos de entrega y recibo de los medios de cómputo ante la ausencia de conexión de red. Para su desarrollo se emplearon desde el punto de vista metodológico métodos teóricos y científicos. El uso de una metodología ágil propició obtener la solución funcional en poco tiempo, mientras que la implementación basada en patrones arquitectónicos y de diseño presento las bases para futuras actualizaciones y mejoras de manera sencilla. Siguiendo la política de migración de la UCI al software libre, las herramientas utilizadas se caracterizaron además por ser de código abierto. Como resultado se obtuvo una aplicación *Rich Internet Applicaction (RIA)* que haciendo uso de las ventajas del entorno de escritorio y el entorno Web logra erradicar la deficiencia en los procesos de entrega y recibo de los laboratorios al presentarse problemas en la conexión de red.

PALABRAS CLAVE

Entrega, *Javascript,* Recibo, Tecnologías Web.

TABLA DE CONTENIDOS

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE TABLAS

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

INTRODUCCIÓN

En un entorno empresarial globalizado como el actual, donde los usuarios son más exigentes, resulta cada vez más difícil lograr un compromiso sólido con los clientes, por lo que se hace imprescindible construir aplicaciones que estén más cerca de las necesidades del usuario final. Es en este marco, donde la plataforma Web se ha convertido en los últimos años en una de las preferidas por los desarrolladores y usuarios [1], pues las aplicaciones implementadas sobre ella tienen considerables ventajas. Entre las más importantes se tiene, que sólo necesitan un navegador y conexión para su uso, la curva de aprendizaje es menos elevada y las actualizaciones llegan por igual a todos los clientes.

Tanto es así, que a nivel internacional, ha surgido una tendencia acelerada a portar aplicaciones de escritorio hacia la Web [2], lo que ha fomentado el establecimiento firme de las mismas, ofreciendo grandes beneficios tanto a los usuarios como a los desarrolladores [1]. Entre las aplicaciones que han sido migradas, se encuentran los sistemas de inventario, logrando centralizar la existencia de niveles de información sobre los productos, de forma simple y rápida.

Grandes empresas alrededor del mundo han apostado por este movimiento [3] y Cuba no ha quedado exento, teniendo como centro de vanguardia en el desarrollo de software, a la Universidad de las Ciencias Informáticas (UCI) [4]. Debido a su tamaño, en esta entidad se deben controlar grandes volúmenes de información sobre los medios y materiales que allí se almacenan, es por esta razón que muy específicamente en el área de laboratorios, se hace necesario un sistema de inventario que permita de forma organizada y rápida llevar a cabo el traspaso de la responsabilidad material sobre las computadoras que prestan servicios en los mismos. Es así que surge en el año 2011 el Sistema de Entrega Digital (SED) [5], aplicación Web que proporciona un servicio de gestión para la entrega y recibo de los medios de cómputo de los laboratorios de la Universidad, proceso que hace uso de la firma digital, método criptográfico mediante el cual se garantiza la validez jurídica de los documentos generados.

En la etapa de pruebas del SED se vieron afectados sus servicios por un período de aproximadamente 3 meses, debido a actualizaciones llevadas a cabo en el nodo central, que acarrearon algunos inconvenientes sobre el sistema de seguridad de la aplicación, llevando a estar fuera de servicio las principales funcionalidades que constituyen el núcleo operativo del sistema. Esto trajo consigo que la entrega y recibo de los laboratorios sufriera afectaciones, incurriendo en gastos de recursos materiales e innumerables esfuerzos como la impresión de modelos de entrega, traslado de los técnicos por laboratorio para recibir los medios de cómputo, así como reunión de los directivos con los técnicos para comprobar las deficiencias encontradas. De forma similar, ha persistido un conjunto de problemas

que han propiciado que un área presente dificultades con la conexión de red imposibilitando el uso del sistema. Ello ha ocasionado no sólo gastos en recursos materiales y esfuerzo, sino también en recursos humanos, tiempo y costo, al volver a utilizar el papel para la impresión de los modelos de entrega.

Asociado a la problemática anterior, se realiza una reestructuración en la Dirección de Laboratorios de la UCI. Dicho cambio implicó la disolución del grupo de desarrollo del Departamento de Soluciones Informáticas, encargado de garantizar la construcción y mantenimiento de los proyectos informáticos de la dirección mencionada anteriormente, entre las que se encontraba el SED.

Teniendo en cuenta esta situación, la inexistencia de documentación ingenieril sobre el proceso de construcción del sistema de entrega, y desaparición del objeto social del grupo de desarrollo de Informatización, queda automáticamente sin recibir mantenimiento la aplicación SED y no se planifican futuras versiones que puedan seguir generando documentos de validez jurídica. Si se tiene en cuenta que dicho sistema está soportado sobre una arquitectura cliente-servidor, no cubre la deficiencia de mantenerse brindando servicios ante la ausencia de conexión de red.

A raíz de la situación problemática anteriormente expuesta, se plantea como **problema a resolver:** ¿Cómo contribuir a la continuidad de entrega y recibo de los medios de cómputo de los laboratorios de la UCI ante la ausencia de conexión de red?

Para dar solución a dicho problema se plantea como **objetivo general:** Desarrollar una aplicación de escritorio del Sistema de Entrega Digital, que utilice tecnologías Web, para garantizar la entrega y recibo de los medios de cómputo de los laboratorios ante la ausencia de conexión de red.

Del objetivo general formulado se derivan los **objetivos específicos:**

1. Caracterizar el desarrollo de aplicaciones de escritorio con tecnologías Web.
2. Desarrollar una versión de escritorio del Sistema de Entrega Digital utilizando tecnologías Web.
3. Validar la solución propuesta.

En correspondencia con el problema planteado, el **objeto de estudio** lo constituye: Proceso de desarrollo de aplicaciones de escritorio con tecnologías Web, componiendo su **campo de acción:** Proceso de desarrollo de aplicaciones de escritorio con tecnologías Web para el Sistema de Entrega Digital.

Para responder a cada uno de los objetivos específicos antes expuestos se definen las siguientes **tareas**:

INTRODUCCIÓN

1. Comparación de las alternativas existentes sobre el desarrollo de aplicaciones de escritorio con tecnología Web a partir del análisis de la literatura científica sobre el desarrollo de las mismas.
2. Selección de las herramientas y tecnologías a utilizar, para establecer las bases del desarrollo.
3. Realización del análisis de la versión de escritorio del Sistema de Entrega Digital para obtener los requisitos funcionales de la aplicación.
4. Elaboración del diseño de la versión de escritorio del Sistema de Entrega Digital para facilitar la implementación de la aplicación.
5. Codificación de la versión de escritorio del Sistema de Entrega Digital para dar cumplimiento a los requisitos funcionales.
6. Validación de la versión de escritorio del Sistema de Entrega Digital para garantizar la entrega y recibo de los medios de cómputo de los laboratorios ante la ausencia de conexión de red.

Observando la situación actual de los laboratorios de la UCI, se define como **idea a defender:** Si se desarrolla una aplicación de escritorio del Sistema de Entrega Digital se garantizará la continuidad del proceso de gestión de entrega y recibo de los medios de cómputo de los laboratorios, ante la ausencia de conexión de red.

Desde el punto de vista metodológico se emplearon los siguientes **métodos científicos**:

Métodos Teóricos:

1. Analítico-sintético: permitió el estudio de diversas bibliografías, asociadas al objeto de estudio, de las cuales se tomaron los rasgos distintivos más importantes, afines con el desarrollo de aplicaciones de escritorio con tecnologías Web, con el objetivo de arribar a conclusiones que sustentaran la necesidad de la investigación.
2. Histórico-lógico: se utilizó para analizar la evolución histórica de soluciones similares, las tendencias más recientes del desarrollo de aplicaciones de escritorio con tecnologías Web y basándose en estos datos, complementar las características necesarias para la solución que se propone.
3. Modelado: se utilizó para realizar el modelado de un Sistema Informático que permitiera desarrollar aplicaciones de escritorio con tecnologías Web.

INTRODUCCIÓN

Métodos Empíricos:

1. Observación: se utilizó para llevar a cabo el estudio de las tendencias y expectaciones del objeto de estudio, para corroborar que ante la ausencia de conexión de red, se retornaba al método de imprimir reportes para llevar a cabo la entrega y recibo de los laboratorios, verificando así la necesidad de una aplicación de escritorio del Sistema de Entrega Digital.
2. Entrevista: se realizaron entrevistas a los usuarios del Sistema de Entrega Digital versión Web para constatar las necesidades de nuevas funcionalidades y mejoras de los servicios existentes.

Posibles resultados:

1. Versión de escritorio del Sistema de Entrega Digital desarrollado con tecnologías Web.
2. Documentación técnica del proceso ingenieril de la versión de escritorio del Sistema de Entrega Digital.

La investigación ha sido estructurada de la siguiente manera:

Resumen, Introducción, 4 Capítulos, Conclusiones, Recomendaciones, Glosario de términos, Bibliografía y Anexos. La presente investigación tiene su mayor peso en la estructura capitular, la cual está organizada de la siguiente manera:

Capítulo 1. Fundamentación teórica: Se describen los conceptos relacionados con las aplicaciones Web, sus diferencias respecto a las aplicaciones nativas y las tendencias, técnicas, tecnologías, metodologías y software usados para el desarrollo de aplicaciones de escritorio desarrolladas con tecnologías Web.

Capítulo 2. Análisis de la aplicación: Describe las características de la aplicación a desarrollar. Se realizan las especificaciones de las funcionalidades del sistema.

Capítulo 3. Diseño de la aplicación: Se describen los patrones arquitectónicos y de diseño empleados en el desarrollo de la aplicación. Se presentan los diagramas de clases y las tarjetas CRC, con el objetivo de mostrar cada una de las funcionalidades que forman parte de la solución.

Capítulo 4. Implementación y pruebas de la aplicación: Se presentan las principales tareas de implementación a realizar y los casos de prueba para las historias de usuario.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

INTRODUCCIÓN

Para comprender en qué consiste la presente investigación, se hace necesario conocer los aspectos más significativos relacionados con los sistemas de inventarios y la construcción de aplicaciones nativas desarrolladas con tecnología Web. Este capítulo constituye la base teórica de la solución propuesta, de ahí que sea en esta temática donde se aborden varios conceptos, tecnologías y herramientas, que son objeto de análisis y estudio con el fin de fundamentar su uso en la solución que se propone.

1.1 CONCEPTOS ASOCIADOS AL DOMINIO DEL PROBLEMA

Con el propósito de que el lector pueda alcanzar una mejor comprensión de los temas que serán tratados en este capítulo, directamente relacionados con el objeto de estudio de la investigación, se describen a continuación un grupo de conceptos coligados al dominio del problema.

Control de Inventario

El Control de Inventarios es la técnica que permite mantener la existencia de productos a niveles deseados. El inventario corresponde al almacenamiento de bienes y productos, los cuales se mantienen en un sitio dispuesto para tal efecto. Los inventarios tienen un papel fundamental en la economía de las empresas e instituciones. Por lo tanto, debe encontrarse en la administración de inventarios un área fructífera para reducir los costos. Desde el punto de vista de la empresa, los inventarios representan una inversión, ya que se requiere de capital para tener reservas de materiales en cualquier estado [6]. Mantener un sistema de inventarios proporciona ventajas como:

1. Reducir los costos de manejo de materiales.
2. Realizar compras masivas o al mayoreo con descuento.
3. Tener un margen para reducir la incertidumbre de los pedidos.
4. Lograr una recuperación favorable de la inversión.

Sistemas Informáticos para el Control de Inventario

Los Sistemas Informáticos para el Control de Inventario son programas, que utilizados por medio de una computadora, permiten llevar el control y la administración de los productos que se encuentran dentro de un almacén. Estos registran las entradas y salidas de productos, así como el control de los costos por cada movimiento de inventario. Administran los productos que se transfieren entre

almacenes y los más sofisticados pueden manejar almacenes virtuales y almacenes móviles para entregas justo a tiempo. De estos sistemas se pueden generar pólizas contables e información estadística del comportamiento de los productos que se almacenan, para lograr ver la rotación de inventarios, y evitar obsolescencias [7].

Sistema de Entrega Digital

Dentro de los Sistemas de Control de Inventarios clasifica el SED, el cual permite gestionar de forma organizada la cantidad de medios de cómputo de un laboratorio, así como las incidencias ocurridas en el mismo. Dicho sistema de control de inventarios es muy útil en el proceso de traspasar la responsabilidad material de los medios del laboratorio. Desarrollado sobre plataforma Web maneja de forma centralizada la información de las diferentes áreas que poseen laboratorios en la UCI.

Aplicaciones RIA

Devenidas en la fusión de las ventajas del software tradicional y las páginas Web, las aplicaciones *RIA* se han convertido en una opción a tener en cuenta en el momento de desarrollar Sistemas de Control de Inventarios, pues al manejar productos que pueden encontrarse en diferentes almacenes centralizan la información relacionada con los mismos. Pueden trabajar tanto localmente como conectadas a la red.

1.2 ANTECEDENTES DEL DESARROLLO DE APLICACIONES DE ESCRITORIO CON TECNOLOGÍAS WEB.

Con el objetivo de comprender las ideas concernientes al desarrollo de aplicaciones de escritorio con tecnologías Web, se hace necesario analizar los antecedentes que dan origen a dicho tema. De esta manera se puede establecer una línea temporal que lleve a comprender el porqué de las ideas utilizadas en la comparación de las aplicaciones Web con las aplicaciones nativas y la fusión de ambos entornos para lograr un máximo aprovechamiento en el proceso de desarrollo de este tipo de software.

1.2.1 Historia de las aplicaciones Web

En los últimos años ha habido una tendencia acelerada en mover aplicaciones de escritorio hacia la Web [2]. Esto ha venido incidido por ciertos factores como son:

1. Crecimiento de Internet y medios de comunicaciones.
2. Relativa facilidad en el desarrollo de aplicaciones Web.
3. Habilidad para funcionar en varios sistemas operativos gracias a los navegadores.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En sus inicios la Web era muy distinta a como se conoce actualmente. El desarrollo sobre esta plataforma estuvo basado en *HTML* y *Javascript*, luego la inclusión de las Hojas de Estilo en Cascada, permitió darle formato a la información que se obtenía a través del navegador. Así se mantuvo por varios años, hasta que recientemente con el uso de las funcionalidades *Ajax*, se le hizo posible a los desarrolladores Web, crear más interactividad en las aplicaciones y así ir más allá del esquemático flujo de trabajo cliente-servidor. Es así como en la plataforma Web se comienzan a ofrecer aplicaciones enriquecidas con un alto grado de complejidad e interactividad. De esta manera se acuña el término Aplicaciones Enriquecidas de Internet (*RIA,* en inglés *Rich Internet Application*), el cual describe un nuevo tipo de aplicaciones en la Web [10], las cuales ofrecen grandes beneficios tanto a los usuarios como a los desarrolladores.

1.2.2 Diferencia de las aplicaciones Web respecto a las aplicaciones de escritorio

Las aplicaciones Web hoy en día se han vuelto más complejas y han puesto al límite las capacidades de funcionamiento de los navegadores y de lo que en plataforma Web se puede hacer, debido a esta limitante el modelo de seguridad de las aplicaciones Web restringe la comunicación con la computadora del usuario, por lo que las mismas no ofrecen el tipo de interacciones que ofrecen las aplicaciones de escritorio. Antes de decidir entre desarrollar una aplicación Web o nativa, sería conveniente saber lo que se necesita y desea hacer, para así evaluar las ventajas y desventajas de cada entorno [11].

Ventajas de las aplicaciones desarrolladas sobre plataforma Web:

1. Pueden ser utilizadas en cualquier ordenador con un navegador.
2. Las actualizaciones se distribuyen instantáneamente a todos los clientes.
3. La movilidad, facilidad, soporte y mantenimiento no dependen del Sistema Operativo en que se ejecute.
4. No necesitan instalación en la computadora del usuario.
5. Las aplicaciones son administradas en un solo lugar, el servidor.
6. Al no depender de estar frente a una computadora específica, la productividad aumenta.
7. Consumen pocos recursos de hardware del lado del cliente.
8. Es la mejor forma de desarrollo multiplataforma.
9. Los marcos de desarrollo actuales, han disminuido la diferencia respecto a las aplicaciones de escritorio, al mismo tiempo que han facilitado su desarrollo.

10. No poseen amenazas recurrentes de virus.
11. Brindan más facilidad al momento de ofrecer soluciones genéricas.
12. Su curva de aprendizaje es más suavizada.

Desventajas de las aplicaciones desarrolladas sobre plataforma Web:

1. Involucran varios lenguajes de programación en su desarrollo.
2. Funcionan en base a aplicaciones de escritorio.
3. La ausencia de conexión de red, ancho de banda y escasas prestaciones del servidor son las limitaciones más serias.
4. Tienen pobre rendimiento e interacción en comparación con las aplicaciones de escritorio.
5. No cubren todo el espectro de posibilidades de desarrollo de las aplicaciones nativas.
6. No se integran con el escritorio del usuario.
7. Poseen mayor tipo de vulnerabilidades (*XSS, SQL injection, CSRF*, entre otras).
8. La mayoría de las aplicaciones se desarrollan sobre lenguajes interpretados por lo que se desaprovecha el rendimiento máximo del servidor.

Ventajas de las aplicaciones desarrolladas para el escritorio:

1. Poseen mayor eficiencia, robustez, velocidad y capacidades gráficas de más resolución.
2. Por lo general, su desarrollo está basado en un solo lenguaje.
3. Ante la inexistencia de conexión de red pueden seguir brindando servicios.
4. No están tan expuestas en cuestiones de seguridad, en comparación a las aplicaciones Web.
5. Poseen mejor experiencia de usuario.
6. Cuentan con mecanismos para acceder a datos remotamente.
7. Hacen uso de un mayor aprovechamiento del entorno de ejecución al estar desarrolladas para un sistema y un hardware específico.
8. Tienen un mayor espectro de actuación.

Desventajas de las aplicaciones desarrolladas para el escritorio:

1. Duplicidad de datos por falta de la unificación de los mismos.

2. No cuentan con portabilidad a diferentes sistemas operativos (resuelto por algunas plataformas como *Java*).
3. Se hace difícil a la hora de realizar actualizaciones o correcciones al programa ya que las instalaciones están diseminadas.
4. Poseen un alto grado de complejidad en el momento de configurar cada una de las instalaciones, dependiendo de las necesidades de cada usuario.
5. Se necesitan de ciertos prerrequisitos para poder instalarlas.

Conclusiones parciales sobre los entornos de desarrollo

Abordados los antecedentes de las aplicaciones Web y sus diferencias respecto a las aplicaciones de escritorio, lo ideal, sería una combinación de los dos entornos de ejecución. Para ello se tendría la velocidad, riqueza visual, integración con la computadora del cliente y lo amigable de las aplicaciones de escritorio, realizando el acceso a datos remotamente, teniendo la actualización en Internet y desarrollando sobre tecnologías Web, de forma tal que si falla la conexión de red, se pueda seguir trabajando y guardando la información localmente, para después actualizar en el servidor [11]. Este tipo de software es también conocido como aplicaciones *RIA* embebidas en el escritorio.

1.2.3 Aplicaciones Enriquecidas de Internet (*RIA*)

Las aplicaciones *RIA*, han surgido como apoyo a la portabilidad de las aplicaciones tradicionales hacia la Web. Las mismas, pueden utilizar un navegador Web estandarizado para ejecutarse, y por medio de complementos o mediante una máquina virtual se le agregan funcionalidades adicionales al software de este tipo. Buscan mejorar la experiencia del usuario, y en su desempeño, son solicitados únicamente los datos que se necesitan [12]. Esto ha traído consigo considerables ventajas, por lo que se han abierto paso en el mundo del desarrollo de software, estableciéndose como una alternativa muy viable.

Ventajas de las aplicaciones RIA:

1. Mejoran el aspecto visual, en comparación con las aplicaciones Web tradicionales.
2. Mejoran el uso de conectividad.
3. Conocen el estado de conexión.
4. Pueden ser utilizadas sin conexión.
5. Poseen medios de comunicación más avanzados.

6. Su instalación es mucho más rápida y fácil, en comparación con las aplicaciones de escritorio.
7. Cuentan con actualizaciones automáticas.
8. Logran parcialmente la interacción con la computadora del cliente.

1.2.4 Análisis de los antecedentes

Por lo analizado con anterioridad sobre los antecedentes del desarrollo sobre la plataforma Web se concluye que: actualmente el desarrollo del software ha inclinado su balanza de forma decisiva hacia la plataforma Web [13], siempre teniendo en cuenta que la misma posee algunas limitaciones que hacen que las aplicaciones nativas no desaparezcan por completo. Las aplicaciones *RIA* han llegado a ser una alternativa muy factible a las restricciones presentes en la Web, logrando insertarse en la plataforma de escritorio, dando la posibilidad de desarrollar sobre ella de forma sencilla, del mismo modo que se hace sobre la Web.

1.3 PROCESO DE DESARROLLO DE SOFTWARE

El proceso de desarrollo de software no es una tarea sencilla. Muchos proyectos de los que se construyen presentan una serie de problemas ya que tienen importantes retrasos, o simplemente no cumplen con las expectativas del cliente. Para no incurrir en estas dificultades, la presente investigación realiza un estudio sobre los distintos tipos de metodologías de desarrollo, seleccionando de ellas la que mejor se adapta a las exigencias y condiciones de la solución propuesta. Unido a esto se abordan el lenguaje de modelado y herramienta CASE seleccionada para modelar los diferentes diagramas que sirven como soporte y documentación ingenieril en el proceso de desarrollo.

1.3.1 Metodologías de desarrollo de software

La incorrecta planificación a la hora de construir un software, es uno de los problemas que con más frecuencia se presentan actualmente. Como solución a esta problemática surgen las metodologías de desarrollo de software, las cuales se pueden definir como un conjunto de actividades encaminadas a lograr un fin específico. Las metodologías imponen un trabajo disciplinado sobre el desarrollo de software con la finalidad de hacerlo más eficiente.

Las mismas pueden ser definidas como un conjunto de métodos, reglas o pautas a seguir, que sirven como un modelo flexible para desarrollar el software y que guían a la dirección del proyecto y a los componentes de desarrollo en la realización de ciertas comprobaciones sistemáticas. Es de este modo que el producto o resultado del trabajo final puede tener mayor posibilidad de éxito, a medida que se vayan cumpliendo los objetivos acordados al inicio, se entregue en tiempo, forma y con la calidad

exigida por el cliente [14]. Existen fundamentalmente dos tipos de metodologías para contribuir al desarrollo de software: Las metodologías tradicionales, también conocidas como metodologías pesadas y las metodologías ágiles o ligeras.

Metodologías pesadas. Proceso Unificado de Desarrollo

Las metodologías pesadas están basadas en normas provenientes de estándares seguidos por el entorno de desarrollo, ofrecen por lo general cierta resistencia a los cambios que puedan producirse durante el ciclo de desarrollo del producto, son procesos muy bien controlados con muchas políticas y normas a seguir. En este grupo se puede encontrar la metodología: Proceso Unificado de Desarrollo (*RUP*, en inglés *Rational Unified Process)*, actualmente la más utilizada de esta alternativa. La misma se caracteriza por ser dirigida por casos de uso, estar centrada en la arquitectura y ser iterativa e incremental. Entre sus principios se encuentran adaptar el proceso a la necesidad del cliente, equilibrar prioridades, demostrar valor iterativamente, fomentar la colaboración entre equipos, elevar el nivel de abstracción y enfocarse en la calidad. Sin embargo *RUP* presenta algunos inconvenientes para su aplicación en algunos proyectos como son: límite de tiempo demasiado ajustado, la documentación al ser tan exhaustiva, hace mucho énfasis en la planificación, no siempre se puede llevar a cabo la implementación del software de acuerdo a su planificación, pues aparecen cambios los cuales solo pueden ser percibidos en el momento de la codificación, entre otras.

Metodologías ágiles. Programación Extrema

Por otro lado las metodologías ágiles están basadas en heurísticas provenientes de prácticas de producción de código, y están especialmente preparadas para sufrir cambios durante el proyecto, son un proceso menos controlado y con pocos principios, por lo que desarrollan software en cortos períodos de tiempo y exigen poca documentación. Entre ellas se puede encontrar la metodología Programación Extrema (*XP*, en inglés *Extreme Programming)*.

La *XP* fue concebida por Kent Beck, como un proceso de creación de software diferente al convencional. En palabras de Beck: "*XP* es una metodología ligera, eficiente, con bajo riesgo, flexible, predecible y divertida para desarrollar software" [15]. *XP* es una de las metodologías ágiles para el desarrollo de software más exitosas de la actualidad. Se utiliza en proyectos con pequeños equipos de desarrollo y con corto plazo de entrega. Se basa en la retroalimentación entre el cliente y el equipo de desarrollo, buena comunicación entre los participantes y simplicidad en las soluciones implementadas. Consiste en una programación rápida, cuya particularidad es tener como miembro del equipo al usuario final. Es adecuada para proyectos con requisitos imprecisos, muy cambiantes, y donde existe un alto riesgo técnico [16].

En *XP* se sigue la idea de la programación en pares, dada las ventajas que ofrece ésta respecto a la creación del código, pues se pueden evitar errores y malos diseños al controlar cada línea de código y decisión de diseño instantáneamente. La interacción entre ambos desarrolladores puede generar discusiones que lleven a mejores estructuras y algoritmos, aumentando la calidad del software. La *XP* incluye buenas prácticas de desarrollo como son el Desarrollo Guiado por Test (TDD, en inglés *Test Development Driver)* [17] e Integración Continua [18].

1.3.2 Lenguaje de modelado

Un lenguaje de modelado visual se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Con ellos es posible diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Entre los más conocidos se encuentran: *UML*, *MOF*, *Ecore*, entre otros.

UML 1.0

Entre los lenguajes de modelado visual más conocidos se encuentra el Lenguaje de Modelado Unificado (*UML*, en inglés *Unified Model Language)*, el cual incluye conceptos semánticos, notación y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como, generadores de informes. La especificación de *UML* no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. Es una consolidación de muchas de las notaciones y conceptos más usados en la modelación orientada a objetos.

Es importante recalcar que *UML* no es una guía para realizar el análisis y diseño orientado a objetos, no es un proceso, sino un lenguaje que permite la modelación de sistemas, con tecnología orientada a objetos. Desde el año 1995, *UML* es un estándar aprobado por la *ISO* como *ISO/IEC* 19501:2005 *Information technology*; está respaldado por el *Object Management Group (OMG)* [19].

1.3.3 Herramienta CASE

Las herramientas de Ingeniería de Software Asistido por Computadora (CASE, en inglés *Computer Aided Software Engineering)* son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de un software reduciendo el costo del mismo. Las mismas pueden servir de ayuda durante el ciclo de vida de desarrollo del software en tareas como el diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente a partir del diseño dado, compilación

automática, documentación o detección de errores. Existen múltiples herramientas con estos fines, tales como *Enterprise Architect*, *Visual Paradigm*, *ArgoUML*, *Rational Rose*, entre otras.

Visual Paradigm (8.0)

Visual Paradigm es una herramienta *CASE* considerada muy completa, fácil de usar, con soporte multiplataforma y de probada utilidad para el analista, que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Posee un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, análisis y diseño, hasta la generación del código fuente y la documentación. Tiene licencia dual, gratuita y comercial. Soporta las últimas versiones de *UML* y la Notación y Modelado de Procesos de Negocios.

En adición al soporte de Modelado *UML*, *Visual Paradigm* provee un generador de mapeo de objetos-relacionales para los lenguajes de programación como *Java*, *.NET* y *PHP*. Una de sus principales ventajas es que incorpora el soporte para trabajo en equipo, lo que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros [20].

Logra integrarse con varias herramientas entre las que se encuentran:

1. *Eclipse/IBM WebSphere*
2. *JBuilder*
3. *NetBeans IDE*
4. *Microsoft Visual Studio*
5. *JDeveloper*

1.3.4 Análisis del proceso de desarrollo ingenieril de software

Por lo analizado con anterioridad sobre el proceso de desarrollo ingenieril de software se concluye que: *XP* fue seleccionada como metodología de desarrollo por ser éste un proyecto donde sólo interviene un programador, en un ambiente de trabajo donde el riesgo de desarrollo es elevado debido al corto tiempo de entrega y a los continuos cambios de requisitos. El cliente forma parte del equipo de desarrollo, de esta manera se logra una mejor retroalimentación, corrección de errores y las posibilidades de entrega de un producto final con la calidad requerida son mayores. Por otra parte se opto *UML* como lenguaje de modelado por su facilidad, conocimiento de antemano y ventajas que provee el mismo. Se eligió como herramienta *CASE Visual Paradigm* por ser multiplataforma dando así

la posibilidad de personalizaciones para una mejor integración con las demás herramientas, en el proceso de desarrollo.

1.4 CONCEPCIÓN GENERAL SOBRE LAS TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS

Antes de implementar la solución que se propone, una de las primeras tareas a ejecutar es llevar a cabo un estudio de las tendencias actuales, tecnologías existentes y selección de las herramientas que giran alrededor del desarrollo de aplicaciones de escritorio con tecnologías Web. De ahí que pueda salir a relucir la oportunidad de reutilización de código y el apoyo en software previamente desarrollados, para facilitar esta labor. Cabe destacar que las principales metas son poder contar con herramientas que faciliten en la mayor medida posible el trabajo, las cuales sean software libre, multiplataforma y de código abierto.

1.4.1 Tendencia actual

En los últimos años, las aplicaciones Web han comenzado a competir en funcionalidad con las aplicaciones nativas. *Gmail* y *Google Maps* fueron dos de los primeros sitios en demostrar que *HTML* y *Javascript* podían proveer una rica experiencia de usuario. Desde entonces se vive un ambiente en el que las aplicaciones Web empujan el desarrollo de tecnologías Web y viceversa. *HTML5* o *WebGL* son algunas de las tecnologías que forman parte indiscutible del presente del desarrollo Web, cuyos horizontes abarcan ya desde los juegos, ofimática, retoque fotográfico, entre otros [21].

Hoy en día existe una tendencia a llevar algunos servicios que se ofrecen vía Web, al escritorio. Las ventajas radican en que sólo se soliciten los datos ofrecidos por el servicio, que realmente sean necesarios y su uso se haga independiente de la plataforma. Ello gracias al empleo de protocolos estándar bien conocidos como lo son *HTTP*, *SOAP*, *XML*, entre otros [22].

Uno de los pioneros en esta práctica fue Google con aplicaciones como *Google Analytics*. Esto supone una ventaja en algunos casos pues no se tendría que gastar una pestaña del navegador y se almacenarían de forma segura las credenciales de acceso de distintos sitios para así poder ser usadas más adelante en otros servicios. En muchas ocasiones, las aplicaciones desarrolladas de esta forma brindan mejor experiencia, con una interfaz más cómoda que la aplicación original vía Web, destacando también que por lo general, que este tipo de software está desarrollado con *HTML5*, *CSS* y *Javascript*. Una vez más se puede apreciar, la preferencia del desarrollo de aplicaciones *RIA,* tomando lo mejor de los 2 entornos de desarrollo.

1.4.2 Plataformas para el desarrollo de aplicaciones de escritorio con tecnologías Web

En el avance del estudio para el desarrollo de la presente investigación, se tiene como uno de los puntos de partida el análisis de las plataformas que actualmente en el mercado permiten el desarrollo de aplicaciones de escritorio con tecnologías Web. A pesar de su reducido número y novedosa llegada, un análisis a profundidad permitirá obtener una valoración de las mismas. Entre dichas plataformas se encuentran:

Adobe Air

Adobe Air es un conjunto de herramientas destinadas al desarrollo de aplicaciones *RIA*, en diferentes lenguajes. Las características de *Air* van más allá de la mera construcción de aplicaciones *RIA,* pues puede interactuar con base de datos locales como *SQLite*. Las aplicaciones desarrolladas con esta plataforma, pueden ser escritas totalmente en *Javascript*, el cual es modificado ligeramente en relación al *script* que corre en los navegadores, ya que por cuestiones de seguridad el código debe ejecutarse en un entorno seguro para que no ataque el sistema del usuario. *Adobe* lo define como un entorno de ejecución que no necesita navegador, para poder portar aplicaciones *RIA* al escritorio [10].

Las aplicaciones implementadas con *Adobe Air* requieren el empaquetamiento, firma digital, e instalación en el sistema de archivos del usuario. Las mismas, proporcionan acceso al sistema de archivos y al almacenamiento local, pueden funcionar sin conexión a Internet, y permiten más funcionalidades una vez esté disponible el mismo [10].

Adobe Air provee 3 formas de desarrollar aplicaciones *RIA*, mediante *HTML5*/*Javascript*, *Flex* y *Flash*. Tiene la posibilidad de trabajar con los datos de 4 maneras, servidor de base de datos a través de Internet, archivos locales de *XML*, base de datos local en *SQLite* y almacenamiento local cifrado (incluido en *Air*).

Actualmente *Adobe* abandonó el soporte para plataformas Linux, al mismo tiempo que amplió el número de dispositivos sobre los cuales corren sus programas. En un paso arriesgado, hizo donación del marco *Flex* a *Apache Software Foundation.* El desarrollo de la interfaz visual sigue basándose en *CSS*.

JavaFX

JavaFX es un conjunto de tecnologías de *Oracle Corporation* para la creación de aplicaciones *RIA*, que pueden ser ejecutadas en una amplia variedad de dispositivos. Está construida sobre tecnología *Java*

para que los desarrolladores puedan crear aplicaciones de forma más sencilla. Expuesto por primera vez en la conferencia *JavaOne* 2007, presenta grandes ventajas para los desarrolladores *Java* y las compañías que forman parte del ecosistema del mismo entorno [23].

Para el desarrollo de aplicaciones *JavaFX* se utiliza un lenguaje declarativo, tipado, llamado *JavaFX Script*, en el cual se puede integrar código *Java*. El código de *JavaFX* es compilado a *Java*, por lo que las aplicaciones implementadas pueden ser ejecutadas en computadores con la máquina virtual de *Java* instalada, o celulares corriendo *JavaME*. La intención de *Oracle Corporation* al desarrollar *JavaFX* es competir en el espacio que ya ocupan Flash de Adobe y *Silverlight* de *Microsoft* [23].

TideSDK

TideSDK (antes conocido como Appcelerator Titanium Desktop) es una plataforma de código abierto para el desarrollo de aplicaciones de escritorio con tecnologías Web. Para el desarrollo sobre la misma pueden ser utilizados lenguajes como *HTML5*, *CSS*, *Javascript, PHP*, *Ruby* o *Python*. Trabaja sobre el marco del motor de renderizado *Webkit*, logrando interactuar con el sistema del usuario, poseer base de datos integrada, entre otras funcionalidades

Básicamente las aplicaciones desarrolladas con *TideSDK* son un conjunto de páginas Web pero con funcionalidades de aplicaciones nativas, teniendo como ventaja no tener que lidiar con las complejidades de los distintos sistemas operativos. En la actualidad *Appcelerator Titanium* centra más sus objetivos en el desarrollo de aplicaciones para plataformas móviles con tecnologías Web. El desarrollo de la interfaz visual sigue basándose en *CSS*. A pesar de ser un proyecto casi nuevo, la plataforma cuenta con una API bien documentada y una activa comunidad de desarrolladores [24].

AppJS

AppJs es una tecnología muy reciente que permite crear aplicaciones de escritorio con *HTML5*, *CSS* y *NodeJS,* orientado principalmente a los programadores Web, utiliza *Chromiun* como núcleo por lo cual se tiene a disposición las últimas *API* de *HTML5.* Al correr sobre *NodeJs*, tiene a su disposición todos los módulos del mismo. A pesar de ser un proyecto muy reciente, se puede vislumbrar el potencial de construir aplicaciones con *AppJs*, pues le brinda la posibilidad a los desarrolladores Web de poner sus conocimientos en nuevos entornos de desarrollo. La construcción de la interfaz visual se sigue llevando a cabo con Hojas de Estilo en Cascada [25].

Deficiencias de las plataformas existentes

Las plataformas abordadas anteriormente presentan algunas deficiencias que son claves en el desarrollo de aplicaciones *RIA* embebidas en el escritorio entre ellas se tienen:

1. No poseen *ORM* integrado.
2. No optimizan el código final generado.
3. No tienen soporte nativo de Internacionalización.
4. No poseen configuración de temas de Interfaz de Usuario.
5. No poseen una estructura de desarrollo organizada.
6. Su desarrollo no está basado en un único lenguaje web.
7. No llegan de forma masiva a los desarrolladores Web.
8. No implementan las mejores prácticas en la construcción de software de calidad.

Estas son las razones por las que se decide no utilizar ninguna de las plataformas anteriormente expuestas, para el desarrollo de la presente investigación.

1.4.3 Lenguajes a utilizar

El lenguaje de codificación es uno de los elementos pilares en el desarrollo de un software. Su elección está basada en el tipo de software que será construido y la plataforma sobre la cual funcionará.

HTML5

Lenguaje de Marcado de Hipertexto (*HTML*, en inglés *HyperText Markup Language*), es mantenido por la *World Wide Web Consortium (W3C),* es utilizado para la creación de páginas Web estáticas. Esta versión incorpora algunas etiquetas nuevas pensadas para hacer que la estructura de la página Web sea más lógica y funcional.

El enfoque general ha cambiado bastante respecto a versiones anteriores de *HTML*, añadiendo semántica y accesibilidad implícitas, especificando cada detalle y borrando cualquier ambigüedad. También se tiene en cuenta que muchas páginas Web actuales son dinámicas, pareciéndose más a aplicaciones que a documentos. Algo básico es que *HTML5* está definido en base al *DOM* (la representación interna de una Web con la que trabaja un navegador), dejando de lado la representación real, definiendo a la vez un estándar *HTML* y *XHTML* [26].

CSS 3

Las Hojas de Estilos en Cascada (*CSS*, por sus siglas en inglés) es un lenguaje creado para controlar el aspecto o presentación de los documentos electrónicos definidos con *HTML* y *XHTML*. *CSS* es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas Web

complejas. Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos *HTML/XHTML* bien formateados y con significado completo (también llamados documentos semánticos). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes [27].

Al crear una página Web, se utiliza en primer lugar el lenguaje *HTML/XHTML* para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, entre otros. Una vez creados los contenidos, se utiliza el lenguaje *CSS* para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, además de otras funcionalidades [27].

Javascript

Javascript (también conocido por *ECMAScript*) es un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas del lado del cliente. Una página Web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, *Javascript* es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con *Javascript* se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

A principios de los años 90, la mayoría de usuarios que se conectaban a Internet lo hacían con *modems* a una velocidad máxima de 28.8 *kbps*. En esa época, empezaban a desarrollarse las primeras aplicaciones Web y por tanto, las mismas comenzaban a incluir formularios complejos. Con páginas cada vez más complejas y una velocidad de navegación tan lenta, surgió la necesidad de un lenguaje de programación que se ejecutara en el navegador del usuario. De esta forma, si el usuario no rellenaba correctamente un formulario, no se le hacía esperar mucho tiempo hasta que el servidor volviera a mostrar el formulario indicando los errores existentes. Brendan Eich, un programador que trabajaba en *Netscape*, pensó que podría solucionar este problema adaptando otras tecnologías existentes (como *ScriptEase*) al navegador *Netscape Navigator 2.0*, que iba a lanzarse en 1995. Inicialmente, Eich denominó a su lenguaje *LiveScript* [28].

Posteriormente, *Netscape* firmó una alianza con *Sun Microsystems* para el desarrollo del nuevo lenguaje de programación. Además, justo antes del lanzamiento *Netscape* decidió cambiar el nombre por el de *Javascript*. *ECMA* creó el comité TC39 con el objetivo de estandarizar un lenguaje de script

multiplataforma e independiente de cualquier empresa. El primer estándar que creó el comité TC39 se denominó ECMA-262, en el que se definió por primera vez el lenguaje *ECMAScript* [28].

Javascript es un lenguaje orientado a objetos basado en el estilo de programación con prototipos, donde las clases no están presentes y la reutilización de comportamiento (conocido como herencia en lenguajes basados en clases) se lleva a cabo a través de un proceso de decoración de objetos existentes que sirven de prototipos. Este modelo también se conoce como programación sin clases, orientada a prototipos o basada en prototipos, un estilo de programación orientada a objetos.

JSON

Javascript Object Notation, es un formato ligero para el intercambio de datos. *JSON* es un subconjunto de la notación literal de objetos de *Javascript*. Es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C++, C#, *Java*, *Javascript*, *Perl*, *Python*, y muchos otros. Estas propiedades hacen que *JSON* sea un lenguaje ideal para el intercambio de datos [28]. *JSON* es empleado habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia cuando la fuente de datos es explícitamente de fiar y donde no es importante el no disponer de procesamiento *XSLT* para manipular los datos en el cliente [29].

1.4.4 Tecnologías a utilizar

El desarrollo de un software muy pocas veces es iniciado desde cero, pero la mayor de las veces es producto de la reutilización de otros proyectos. En la presente investigación se tienen 3 proyectos base principalmente para el desarrollo de la solución informática, un proyecto de *Mozilla, Chromeless,* el marco de trabajo de *Javascript Qooxdoo* y el gestor de base de datos *SQLite.*

La unión de estos 3 proyectos provee funcionalidades no soportadas por las plataformas anteriormente mencionadas, al mismo tiempo que realizan las tareas principales en el desarrollo de aplicaciones de escritorio con tecnologías Web.

Chromeless 0.4

Chromeless es un proyecto de *Mozilla Foundation*, fue desarrollado para construir fácilmente aplicaciones de escritorio usando solamente tecnologías Web. Específicamente para que el usuario pudiera desarrollar su propio navegador mediante lenguajes como *HTML*, *CSS* y *Javascript*. Entre sus capacidades está la de embeber y manejar contenido Web de forma segura.

Incluye un conjunto de módulos, que proveen una *API Javascript* de alto nivel que puede ser usada para desarrollar las aplicaciones tradicionales. También posee un conjunto de herramientas que permiten testear y empaquetar la aplicación una vez terminada para ser distribuida.

Chromeless está diseñado para ser un contenedor seguro y es principalmente una aplicación Web, por lo que tiene a su disposición todas las facilidades que este entorno ofrece listas para la construcción de aplicaciones de este tipo, incluyendo las últimas actualizaciones del motor intérprete de *Firefox*, a lo cual se le suman las nuevas tendencias del *HTML5*. Además, *Chromeless* incluye librerías *Javascript* que dan acceso a la máquina, y permiten integrar la aplicación con el sistema operativo, reuniendo lo mejor de ambas tecnologías. *Chromeless también* permite generar un paquete aparte, que combinado con la máquina virtual *Xulrunner*, da paso a una aplicación de escritorio sólida [30].

Las aplicaciones desarrolladas con *Chromeless* corren sobre la capa de seguridad del sistema operativo, por lo cual solo trabaja con los permisos que se le den por el sistema. Por el lado de la navegación tiene las mismas restricciones de seguridad que las aplicaciones desarrolladas para navegar en la Internet.

Actualmente el código fuente de *Chromeless,* se encuentra compartido para todos en *Github*. El proyecto ha sido abandonado por *Mozilla Foundation*, pero aún sirve como punto de partida para desarrollar otros proyectos.

SQLite 3

SQLite es un proyecto de dominio público creado por D. Richard Hipp que implementa una pequeña librería de aproximadamente 350Kb programada en lenguaje C, que funciona como un sistema de gestión de base de datos relacionales [31]. A diferencia de los motores de base de datos convencionales con la arquitectura cliente-servidor, *SQLite* es independiente, ya que no se comunica con un motor de base de datos, sino que las librerías de *SQLite* pasan a integrar la aplicación. La misma utiliza las funcionalidades de *SQLite* a través de llamadas simples a subrutinas y funciones.

Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un solo fichero estándar, en la máquina local. Esto hace que cada usuario pueda crear tantas bases de datos como desee sin la necesidad de la intervención de un administrador de bases de datos que gestione los espacios de trabajo, usuarios y permisos de acceso [31].

En su versión 3, *SQLite* soporta bases de datos de hasta 2 *Terabytes (Tb)* de tamaño, y también permite la inclusión de campos tipo Blob. *SQLite* puede ser utilizado de 2 formas, como gestor de base de datos local en una computadora. De esta forma se pueden gestionar bases de datos con *SQLite* igual que si se estuviera trabajando con un sistema gestor de base de datos como *MySQL* sin necesidad de instalar nada, ya que *SQLite* se compone de un único archivo ejecutable. Si se utiliza mediante un lenguaje, se usan funcionalidades específicas del lenguaje, sin necesidad de tener instalado o conectar con un servidor de base de datos [31].

Qooxdoo 2.0.1

Qooxdoo es un framework *Javascript* universal que permite crear aplicaciones para una gran variedad de plataformas. Con un modelo de programación orientada a objetos, se pueden construir aplicaciones *RIA*, tanto Web tradicionales, como aplicaciones para móviles, incluso aplicaciones que corran fuera del navegador. Se pueden desarrollar aplicaciones a cualquier escala, tomando ventaja de un grupo de herramientas que posee. Es un proyecto de código abierto mantenido en gran parte por su comunidad [32].

Qooxdoo brinda ventajas como:

1. Está basado en *namespaces.*
2. Posee fácil integración con otras librerías.
3. Tiene una *API* completa de las clases del *framework.*
4. Brinda la posibilidad de generar la *API* de la aplicación desarrollada.
5. Permite la optimización y compresión del código final.
6. Permite la internacionalización de la aplicación.
7. Posee una herramienta de generación de pruebas unitarias.
8. El desarrollo de su interfaz gráfica es como la de *Qt* o *SWT.*
9. No requiere conocimiento de *HTML* o *CSS* para construir las interfaces.
10. Posee una fácil configuración de temas para la apariencia.
11. Posee paquetes de iconos y temas por defecto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Firma Digital

La autenticidad de algunos documentos legales y en general, cualquier tipo de documento se determina mediante el uso de la firma manuscrita. Para que los documentos enviados de forma digital tengan la misma validez que un documento manuscrito, se crea la firma digital, método criptográfico que asocia una identidad ya sea de una persona en particular o de un equipo, a un mensaje enviado a través de una transmisión por la red.

Su uso puede ser diferente dependiendo de lo que se desee hacer con la firma ya que existen las posibilidades de validar que el documento es emitido por una persona en específico, se puede expresar conformidad con algún documento de tipo legal y asegurar que no podrá modificarse el contenido del mensaje. La firma digital permite tener más seguridad a la hora de emitir un documento de manera íntegra a través de la Web, siendo el resultado de aplicar a un documento en línea, un procedimiento matemático que requiere datos que exclusivamente conoce la persona que firma, encontrándose ésta bajo su absoluto control. Esta firma debe ser susceptible a verificación por terceras partes, de manera tal que dicha comprobación permita, simultáneamente, identificar al firmante y detectar cualquier cambio al documento digital posterior a su firma [33].

JSingPdf 1.4.4

JSingPdf es una aplicación *Java* que agrega firmas digitales a un documento pdf. Puede ser utilizada como un distribuible solo o como un agregado de la herramienta *OpenOffice*. La aplicación usa la librería *jsingpdf-itext* para la manipulación de los archivos pdf. Es software libre y puede ser usado libremente tanto en sectores privados como de negocios. Posee 2 modalidades para su uso, desde el terminal, donde se le indican una serie de comandos que configuran la firma a utilizar y documento, o en entorno visual, donde se puede realizar un diseño del documento y firma a incrustar [34].

1.4.5 Herramientas a utilizar

En la presente investigación se lleva a cabo un estudio de las principales herramientas de desarrollo disponibles para la construcción de la solución que se propone. Las premisas de mayor importancia para su selección son: políticas de migración de la UCI hacia el software libre y la capacidad de las mismas para funcionar sobre varias plataformas.

IDE de desarrollo

Un Entorno de Desarrollo Integrado (IDE, en inglés *Integrated Development Environment*) consiste básicamente en un software que previamente ha sido instalado en la máquina y cuyo principal objetivo es el desarrollo de otro software. Es un entorno de programación que ha sido empaquetado como un

programa de aplicación. Puede ser exclusivo para un lenguaje de programación o bien, para varios. Suele consistir de un editor de código (con facilidades como resaltado de sintaxis, completamiento de código y navegación entre clases), un compilador y herramientas de automatización de la compilación, un depurador y en algunos casos un constructor de interfaz gráfica. Una vez seleccionado el lenguaje para implementar y el tipo de aplicación a desarrollar, se valoro el uso de varios IDE como fueron: *Netbeans*, *Aptana*, *Eclipse* y *Spket*, siendo este último el seleccionado por su sencillez, poco consumo de recursos, ser multiplataforma y software libre.

Spket 1.6.22

Spket es un *IDE*, liviano, sencillo para la implementación de aplicaciones *RIA*. Tiene soporte para la mayoría de las plataformas que desarrollan *RIA*, así como para las librerías de *Javascript* más modernas. Proporciona funciones como el completado de código, el coloreado de sintaxis y el contenido en línea que ayuda a los desarrolladores a crear de forma productiva un código eficiente. Ofrece muchas características para aumentar la productividad de los desarrolladores de *Mozilla Extentions*. El potente editor de *Javascript* proporciona características como la finalización de código y contenido en línea. Puede ser utilizado como una aplicación aparte o como un módulo del *IDE Eclipse*. [35].

Sistema Gestor de Base de Datos (SGBD)

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. El propósito general de los sistemas de gestión de bases de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización. Permiten describir los elementos de datos con su estructura, sus interrelaciones y sus validaciones. De los SGBD para el desarrollo con *SQLite* se evaluaron 2 alternativas principalmente: *SQLitemanager y SQLite Studio,* seleccionando este último al no depender de otra herramienta como es el caso de *SQLitemanager,* ya que el mismo se distribuye en forma de *Addon* del navegador *Firefox.*

SQLite Studio 2.0.28

SQLite Studio es un SGBD *SQLite* multiplataforma, libre, potente y liviano, que de forma gráfica permite realizar consultas *SQL* y exportar las mismas a diferentes formatos. Con una interfaz intuitiva *SQLite Studio* ha sido traducido a varios idiomas, permitiendo llevar un historial de las consultas ejecutadas. Entre sus características más destacadas se tienen: archivo ejecutable único, no hay necesidad de instalar o desinstalar, interfaz intuitiva, todas las funciones de *SQLite2* y *SQLite*3 se

encuentran en la *GUI* simple, exporta en varios formatos (*SVG*, *XML*, *HTML*, *SQL*), facilidades como formateado de código, historial de consultas ejecutadas, comprobación de sintaxis, entre otras [36].

1.4.6 Análisis de las tecnologías y herramientas a utilizar

Por lo analizado con anterioridad sobre las tecnologías y herramientas a utilizar se concluye que: portar algunos servicios de la Web al escritorio posee algunas ventajas, pues la integración de ambas aplicaciones desarrolladas en entornos diferentes facilita el trabajo de los usuarios finales, brindándoles la posibilidad de ahorrar conexión en ancho de banda y un acceso más directo a la información buscada. También cabe destacar que las plataformas existentes para el desarrollo de aplicaciones de escritorio con tecnologías Web, anteriormente abordadas, no cumplen las exigencias demandadas por los programadores Web, pues presentan algunas deficiencias que son claves para la construcción de este tipo de aplicaciones.

Los criterios fundamentales sobre los que se basó la selección del lenguaje, fueron la facilidad de uso, su flexibilidad para funcionar en varios dispositivos, por constituir el lenguaje de desarrollo de las principales tecnologías asociadas a la construcción de aplicaciones de escritorio con tecnologías Web. La selección de las tecnologías presentadas, su fusión para dar solución a las deficiencias expuestas en las plataformas estudiadas, está basada principalmente en el excelente acoplamiento del proyecto *Chromeless 0.4* y el marco de trabajo *Qooxdoo 2.0.1*, al tener como base para su desarrollo el lenguaje *Javascript* y ser ambos software libre y de código abierto. En caso del soporte de persistencia de datos, *SQLite*3 fue seleccionado por ser muy liviano, rápido y poder integrarse de manera sencilla y eficiente al producto desarrollado.

De las herramientas investigadas, *Spket 1.6.22* y *SQLite Studio 2.0.28* fueron las seleccionadas primeramente por ser software libre, multiplataforma y de código abierto, dando la posibilidad de ser extendidas para la personalización e interacción con la plataforma a desarrollar, además de tener en cuenta la experiencia de desarrollo con las mismas.

CONCLUSIONES PARCIALES

Después de abordados los elementos concernientes a la fundamentación teórica de la investigación, como fueron los antecedentes, proceso de desarrollo de software, tendencias actuales, herramientas y tecnologías de desarrollo, se concluye que la selección de herramientas, metodologías y tecnologías para el desarrollo de aplicaciones de escritorio con tecnología Web, está sustentada sobre la base de cumplimiento de las necesidades específicas de la aplicación a desarrollar. A pesar de que los análisis hechos arrojaron que las soluciones existentes para el desarrollo del producto no brindan una solución

completa, el estudio de su comportamiento y diseño sirvieron como punto de partida para la unión de los proyectos *Chromeless* y *Qooxdoo*. Por lo cual el perfil tecnológico queda estructurado de la siguiente manera:

1. Metodología de desarrollo: *XP*
2. Lenguaje de modelado: *UML* 1.0.
3. Herramienta *CASE*: *Visual Paradigm* 8.0.
4. Lenguaje de desarrollo: *Javascript.*
5. Tecnologías: *Chromeless 0.4*, *Qooxdoo 2.0.1, SQLite 3 y JSingPdf 1.4.4.*
6. Herramientas de desarrollo: *Spket 1.6.22*, *SQLite Studio 2.0.28.*

CAPÍTULO 2: ANÁLISIS DE LA APLICACIÓN

INTRODUCCIÓN

Actualmente cuando se lleva a cabo el desarrollo de un proyecto, no se tienen totalmente claros los aspectos y objetivos más importantes para el mismo. De igual modo se desconoce el alcance que tendrá, tiempo de demora o esfuerzo a emplear para su construcción. En gran medida la planificación ayuda a contrarrestar estos inconvenientes, sirviendo de guía para el trabajo del equipo de desarrollo ya que de esta manera se obtienen mayores posibilidades de éxito.

A partir de la metodología de desarrollo de software seleccionada en el capítulo anterior para la construcción de la aplicación propuesta, se estudian 3 fases para llevar a cabo las tareas de planificación: la fase de exploración, la fase de planificación y la fase de iteración. Este capítulo recoge los aspectos significativos relacionados con el análisis del sistema a codificar. Para ello se valora el flujo actual del proceso de entrega y recibo de los laboratorios, se tienen en cuenta los objetivos a automatizar encontrados en el proceso anterior y se establecen las características del sistema propuesto, donde se obtienen las historias de usuario, iteraciones, cronograma de liberación del producto y requisitos no funcionales que debe poseer la aplicación.

2.1 FLUJO ACTUAL DEL PROCESO

El Sistema de Entrega Digital se encuentra brindando servicios en la actualidad, en esta situación analizar el flujo de procesos para la presente investigación es innecesario. Todo lo contrario sucede cuando un área determinada, presenta deficiencias en la conexión de red, para lo cual se examinarán por separado los procesos de recibo y entrega de laboratorios.

2.1.1 Recibo del laboratorio

El proceso de recibo comienza cuando el técnico saliente entrega un reporte firmado con las cantidades de los medios de cómputo del laboratorio así como las anomalías encontradas en los diferentes puestos. Cuando llega el técnico que recibe, realiza un conteo de los diferentes medios de cómputo presentes en el laboratorio, dígase: torres, monitores, teclados, entre otros. Posteriormente, registra los datos obtenidos en el reporte de entrega, comparándolos con los datos plasmados por el técnico saliente. Seguidamente recorre cada computadora asegurándose que las características del *setup* escritas en el parte de entrega por el técnico anterior coinciden plenamente. En caso de encontrarse una anomalía en alguno de los puestos de trabajo, escribe la misma en el área de

observaciones. De la misma manera ocurre, si encuentra en el parte una observación que no se corresponde al laboratorio, la cual es eliminada.

Una vez llenados y verificados los datos del parte, el técnico que recibe pasa a firmar el documento, plasmando su nombre, laboratorio, área a la que pertenece, la hora y la firma. Finalmente los técnicos se reúnen con el jefe de área para debatir las deficiencias encontradas durante el recibimiento de los laboratorios asignados, y este último guarda el parte de entrega para tener un historial de las deficiencias encontradas en los laboratorios recibidos.

2.1.2 Entrega del laboratorio

El proceso de entrega comienza cuando el técnico que recibe inicia su turno de guardia, el mismo debe verificar cada 2 o 3 horas las cantidades de los medios de cómputo del laboratorio, así como las observaciones plasmadas en el reporte de recibo, de manera tal que coincidan con las presentes en el laboratorio. Es durante este tiempo que puede agregar, eliminar o modificar las cantidades y las observaciones del reporte nuevo. Esta información queda respaldada al rubricar con su nombre y firma el documento; de esta manera el reporte queda listo para cuando llegue el técnico que recibirá el laboratorio.

2.2 OBJETO DE AUTOMATIZACIÓN

Expuesto el flujo de proceso de la entrega y recibo de los laboratorios, puede ser evaluada la existencia de elementos que constituyen objetos de automatización, debido a que actualmente ante la inexistencia de conexión de red, el recibo y entrega de los medios de cómputo de los laboratorios a pesar de funcionar es ineficiente en su mayoría, pues existe cierto nivel de desorganización sobre los reportes firmados, se producen errores que llevan a tachaduras en los mismos, se extravían los reportes, entre otras deficiencias. Para un mejor desarrollo de las actividades antes expuestas se hace necesaria la automatización de varios procesos, lo cual resultará en una mejora práctica, revertida en mayor fiabilidad, rendimiento y facilidad de trabajo.

En este sentido se ha determinado automatizar 2 procesos considerados de vital importancia, la generación de los reportes de entrega con las cantidades de los medios de cómputo y las observaciones, y la firma que otorgue validez jurídica al documento obtenido. Se debe tener en cuenta que para llevar a cabo esta tarea se deben digitalizar otras labores de menor peso que son la base para la ejecución de dichos procesos, entre ellas se encuentran: la gestión de los reportes cuando se entrega y se recibe, la sincronización de los datos con el SED, la comprobación de funcionamiento del SED, entre otras.

CAPÍTULO 2: ANÁLISIS DE LA APLICACIÓN

2.3 CARACTERÍSTICAS DE LA PROPUESTA DEL SISTEMA

La aplicación a desarrollar tiene como objetivo facilitar el proceso de entrega y recibo de los medios de de cómputo de los laboratorios de la UCI en los momentos en que no exista conexión de red. Para ello cuenta con una serie de funcionalidades que garantizarán la automatización de estos procesos, que abarcarán desde la gestión de cantidades de medios, hasta la sincronización de datos con el SED, pasando por la generación de documentos de validez jurídica. Para lograr una mejor organización en el recibo y entrega de los laboratorios se hace necesario realizar una planificación de dicho proceso de desarrollo de software, basándose en las fases que propone la metodología seleccionada, resultando de las mismas las tareas que reemplazarán un gran documento de requisitos.

2.3.1 Fase de exploración

Esta es la primera fase de *XP*, en la que los clientes plantean a grandes rasgos las funcionalidades que son de interés para la elaboración del producto, transformándose las mismas en historias de usuario. Partiendo de la información obtenida, el equipo de desarrollo evaluará de forma general el tiempo de codificación, se familiarizará con las herramientas, tecnologías y prácticas que serán utilizadas en el proyecto, así como se explorarán las posibilidades de la arquitectura del sistema construyendo un prototipo para ello. La fase de exploración toma de pocas semanas a pocos meses, dependiendo de la habilidad que tengan los programadores con las tecnologías seleccionadas. Es necesario aclarar que las estimaciones llevadas a cabo en esta etapa son primarias, ya que las mismas se basan en datos de alto nivel los cuales pueden variar a medida que se analicen con mayor cuidado y detalle en las siguientes fases [37].

2.3.1.1 Personal relacionado con el sistema

Una de las bases fundamentales que se deben tener en cuenta cuando se comienza el desarrollo de un sistema informático es la delimitación de la audiencia a la cual va dirigido el mismo, lo cual en algunas ocasiones, puede ser un personal relacionado con el sistema o alguien completamente ajeno a él. Ha de especificarse que esta audiencia a su vez puede ser dividida en grupos atendiendo a sus necesidades.

Se define como persona relacionada con el sistema, a toda aquella que de una manera u otra interactúe con este, obteniendo un resultado de uno o varios procesos que se ejecutan en el mismo. También son considerados como personas relacionadas con el sistema, aquellas que se encuentran involucradas en dichos procesos, que participan en ellos, pero no obtienen ningún resultado de valor.

Personal	Justificación
Técnico de laboratorio que recibe.	Es el encargado de recibir las cantidades de los medios de cómputo y las observaciones, información que aparecerá en el parte de recibo, firma digitalmente los reportes.
Técnico de laboratorio que entrega.	Es el encargado de actualizar las cantidades de los medios de cómputo y las observaciones. También puede llevar a cabo la sincronización del sistema con la versión Web.

Tabla 1. Personal relacionado con el sistema.

2.3.1.2 Historias de Usuario

Una historia de usuario es una simple descripción de una funcionalidad del software por la que el cliente va a pagar. Los detalles extra son agregados cuando la historia de usuario es implementada, siendo estos procesos de requerimientos simples y sencillos. En cualquier momento estas historias pueden romperse, remplazarse, unirse o dividirse. Entre las principales características que debe tener una buena historia de usuario se encuentran: la historia debe ser entendida por el cliente (representa un concepto y no una especificación detallada), cada historia debe devolver algún valor para el cliente, el tamaño de las historias de usuario debe ser tal que se puedan construir varias de ellas en una iteración (duración aproximada entre 1-3 semanas ideales), las historias deben ser independientes unas de otras, cada historia debe ser testeable [38].

Respecto a la información contenida en la historia de usuario, existen varias plantillas sugeridas pero no existe un consenso al respecto. En muchos casos sólo se propone utilizar un nombre y una descripción, quizás además una estimación de esfuerzo en días. Otras de las actividades a tener en cuenta es la granularidad de las historias, lo cual es muy variable en dependencia de la complejidad del sistema, debido a esto, cada característica importante debe establecerse como una historia de usuario. No hay que preocuparse si en un principio no se identifican todas las historias de usuario, puesto que el proceso de elaboración de las iteraciones conlleva una retroalimentación en sí mismo. Al comienzo de cada iteración estarán registrados los cambios en las historias de usuario y es a partir de ello que se planificará la siguiente. Las historias de usuario son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración.

En el análisis para el desarrollo de la versión de escritorio del SED inicialmente fueron obtenidas las siguientes historias de usuario:

1. Autenticar usuario.

2. Recibir laboratorio.
3. Actualizar entrega de laboratorio.
4. Generar documento de validez jurídica.
5. Comprobar si SED está en línea.
6. Sincronizar datos.

Como se planteó anteriormente una historia de usuario debe poder ser estimada, de lo contrario, si es muy grande, la misma es dividida en 2 o más historias de usuario, debido a lo inestable de la planificación de horas dedicadas al desarrollo de cada historia de usuario se decidió establecer el aproximado del tiempo de estimación en días ideales (semanas es la medida de tiempo más utilizada). El tiempo de estimación expuesto en las tablas es un aproximado, así como la prioridad de cada historia de usuario. En las historias de usuario más complejas se utiliza una programación exploratoria, la cual permite tener una idea del código a implementar para completar la misma. Es por ello que para la presente aplicación son divididas algunas de las historias de usuario, y siguiendo los principios que conforman las buenas historias de usuario, quedan conformadas de la siguiente manera:

Historia de Usuario		
ID: 1	**Nombre:** Autenticar usuario.	
Descripción: El técnico que va a entregar o recibir debe identificar de forma inequívoca el laboratorio donde realizará la tarea, identificándose por su nombre completo y su correo.		
Estimación aproximada: 5	**Prioridad:** Alta	**H.U original:** 1
Observaciones:		

Tabla 2. H.U: Autenticar usuario.

Historia de Usuario		
ID: 2	**Nombre:** Actualizar cantidad de medios del reporte de entrega.	
Descripción: Cuando el técnico está en el laboratorio debe llevar a cabo cada cierto tiempo un conteo de los medios presentes en el mismo y registrarlo en el reporte de entrega.		
Estimación aproximada: 11	**Prioridad:** Alta	**H.U original:** 3
Observaciones:		

Tabla 3. H.U: Actualizar cantidad de medios del reporte de entrega.

Historia de Usuario		
ID: 3	**Nombre:** Actualizar observaciones del reporte de entrega.	
Descripción: Cuando el técnico está en el laboratorio debe revisar cada cierto tiempo las deficiencias por cada puesto de trabajo, quitar aquella que no se corresponda y agregar alguna que aparezca nueva en el reporte de entrega.		
Estimación aproximada: 9	**Prioridad:** Alta	**H.U original:** 3
Observaciones:		

Tabla 4. H.U: Actualizar observaciones del reporte de entrega.

Historia de Usuario		
ID: 4	**Nombre:** Recibir cantidades de medios del reporte de recibo.	
Descripción: Cuando llega el técnico que recibe, realiza un conteo de las cantidades de los medios del laboratorio, las compara con las cantidades plasmadas por el técnico que entrega y procede a escribir los datos resultantes de esta operación. El jefe de área le entrega un nuevo reporte para que sea llenado durante su turno de trabajo.		
Estimación aproximada: 11	**Prioridad:** Alta	**H.U original:** 2
Observaciones:		

Tabla 5. H.U: Recibir cantidades de medios del reporte de recibo.

Historia de Usuario		
ID: 5	**Nombre:** Recibir observaciones del reporte de recibo.	
Descripción: Cuando llega el técnico que recibe realiza una revisión de las deficiencias por puesto, que deben coincidir con las plasmadas por el técnico que entrega, elimina aquella que no se corresponda y agrega alguna que aparezca nueva en el reporte de entrega. El jefe de área le entrega un nuevo reporte para que sea llenado durante su turno de trabajo.		
Estimación aproximada: 9	**Prioridad:** Alta	**H.U original:** 2
Observaciones:		

Tabla 6. H.U: Recibir observaciones del reporte de recibo.

Historia de Usuario		
ID: 6	**Nombre:** Generar reporte.	
Descripción: Se debe generar un documento electrónico cuando se recibe, con las cantidades y observaciones plasmadas por el técnico que entrega y el técnico que recibe.		
Estimación aproximada: 15	**Prioridad:** Alta	**H.U original:** 4
Observaciones: Programación de exploración		

Tabla 7. H.U: Generar reporte.

Historia de Usuario		
ID: 7	**Nombre:** Firmar reporte.	
Descripción: En el momento de actualizar el reporte, el técnico debe poner su firma electrónica que será guardada hasta el momento en que le reciban. En el momento de recibo el técnico plasma su firma digital. Una vez que se genere el documento electrónico las firmas de ambos técnicos son colocadas en el mismo, lo cual brinda validez jurídica al reporte, se elimina la firma del técnico que entregó y se mantiene la firma del técnico que recibió para futuras actualizaciones del reporte.		
Estimación aproximada: 10	**Prioridad:** Alta	**H.U original:** 4
Observaciones: Programación de exploración		

Tabla 8. H.U: Firmar reporte.

Historia de Usuario		
ID: 8	**Nombre:** Verificar SED.	
Descripción: Verificar que el Sistema de Entrega Digital esté en línea o no.		
Estimación aproximada: 2	**Prioridad:** Baja	**H.U original:** 5
Observaciones:		

Tabla 9. H.U: Verificar SED.

Historia de Usuario		
ID: 9	**Nombre:** Sincronizar áreas.	
Descripción: Se deben obtener del SED las áreas, para que la versión de escritorio esté sincronizada con la versión Web. Al cambiar las áreas, los reportes antiguos son borrados y el técnico debe de actualizar el reporte de entrega.		
Estimación aproximada: 3	**Prioridad:** Media	**H.U original:** 6
Observaciones:		

Tabla 10. H.U: Sincronizar áreas.

Historia de Usuario		
ID: 10	**Nombre:** Sincronizar problemas.	
Descripción: Se deben obtener del SED, diferentes tipos de problemas, para que la versión de escritorio este sincronizada con la versión Web. Al cambiar los problemas, los reportes antiguos son borrados y el técnico debe de actualizar el reporte de entrega.		
Estimación aproximada: 3	**Prioridad:** Media	**H.U original:** 6
Observaciones:		

Tabla 11. H.U: Sincronizar problemas.

Historia de Usuario		
ID: 11	**Nombre:** Sincronizar observaciones.	
Descripción: Se deben obtener del SED, las observaciones, para que la versión de escritorio esté sincronizada con la versión Web. Al cambiar las observaciones, los reportes antiguos son borrados y el técnico debe de actualizar el reporte de entrega.		
Estimación aproximada: 3	**Prioridad:** Media	**H.U original:** 6
Observaciones:		

Tabla 12. H.U: Sincronizar observaciones.

Historia de Usuario		
ID: 12	**Nombre:** Sincronizar reportes.	
Descripción: Los datos de los reportes realizados en la versión de escritorio se deben sincronizar con la versión Web.		
Estimación aproximada: 11	**Prioridad:** Alta	**H.U original:** 6
Observaciones: Programación exploratoria		

Tabla 13. H.U: Sincronizar reportes.

2.3.2 Fase de planificación

El propósito general de la fase de planificación es que los clientes y desarrolladores se pongan de acuerdo en qué historias de usuario deben estar listas para la primera liberación. De forma particular, es en esta fase donde los programadores obtienen la estimación del esfuerzo necesario para la elaboración de las historias de usuario. La medida para calcular dicho esfuerzo es el punto y por lo general, un punto equivale a una semana ideal, en el caso particular de la presente investigación se mantendrá durante todo el documento como valor de un punto, un día ideal debido a las características del equipo de desarrollo y el entorno de trabajo.

El tiempo ideal es aquel en el cual se escribe el código sin distracciones y con una dedicación a tiempo completo. Esta estimación incluye todo el esfuerzo asociado a la implementación de la historia de usuario, como son: las pruebas unitarias, integración, refactorización del código, la preparación y ejecución de las pruebas de aceptación, entre otras actividades.

De acuerdo a los intereses del cliente será asignada la prioridad a cada historia de usuario, con vista a cumplir con uno de los principales objetivos que persigue *XP*, aumentar el valor del producto final en el menor tiempo de desarrollo posible. Es en esta fase donde se obtiene un aproximado del cronograma de entrega de cada una de las liberaciones. Esta es una fase de muy corta duración, pero clave en el avance de la solución a desarrollar.

2.3.2.1 Prioridad de las Historias de Usuario

La asignación de prioridad a las historia de usuario por el cliente es lo que decide el orden en el cual se implementarán las mismas, siempre y cuando estén de acuerdo usuario y desarrollador. Las historias de usuario con menor número de prioridad son las más importantes, quedando conformadas de la siguiente manera:

Historia de Usuario	Prioridad
Autenticar usuario	1
Actualizar cantidad de medios del reporte de entrega	3
Actualizar observaciones del reporte de entrega	3
Recibir cantidades de medios del reporte de recibo	2
Recibir observaciones del reporte de recibo	2
Generar reporte	4
Firmar reporte	4
Verificar SED	7
Sincronizar reportes	5
Sincronizar áreas	6
Sincronizar problemas	6
Sincronizar observaciones	6

Tabla 14. Prioridad de Historias de Usuario.

2.3.2.2 Estimación de esfuerzo de las Historias de Usuario

Una vez asignadas las prioridades de las historias de usuario, los desarrolladores comienzan a estimar el esfuerzo necesario para la elaboración de cada una de ellas. Esta estimación es basada principalmente en la velocidad del equipo de desarrollo y en la semejanza con historias de usuario desarrolladas con anterioridad. Las historias de usuario de la presente investigación tienen un valor entre 5-15 puntos (días ideales), 1-3 puntos (medida utilizada normalmente, semanas ideales).

Los puntos estimados son expresados en días ideales, ha de tenerse en cuenta que muy pocas veces este cronograma se lleva a cabo exactamente como se planifica. El esfuerzo necesario para construir las historias de usuario está basado en la técnica de estimación para el desarrollo ágil, quedando conformada de la siguiente manera:

Historia de Usuario	Esfuerzo necesario (puntos estimados)
Autenticar usuario	9
Actualizar cantidad de medios del reporte de entrega	12
Actualizar observaciones del reporte de entrega	13
Recibir cantidades de medios del reporte de recibo	12
Recibir observaciones del reporte de recibo	13
Generar reporte	15
Firmar reporte	12
Verificar SED	5
Sincronizar reportes	15
Sincronizar áreas	7
Sincronizar problemas	7
Sincronizar observaciones	7

Tabla 15. Estimación del esfuerzo necesario por Historia de Usuario.

2.3.2.3 Cronograma de liberación

Cuando se planifica la liberación de un software se debe llevar un balance de la misma, puesto que si se realiza muy pronto no se tendrán suficientes funcionalidades que ameriten dicha liberación, por otro lado esperar mucho tiempo entre liberaciones, llevará a que el software desarrollado quede atrás respecto a la competencia. Contrario a la creencia que existe sobre *XP*, que se debe llevar a cabo una liberación cada vez que es terminada una iteración, las liberaciones son lanzadas con un producto funcionando en óptimas condiciones, ejemplo de un escenario donde no ocurre esto, es al momento de dividir historias de usuario lo cual llevaría a esperar varias iteraciones, de forma tal que al ser liberado éste, contenga todas las funcionalidades necesarias de la historia de usuario original. La planificación de la liberación es un esfuerzo unido entre el cliente y el desarrollador, el primero decide qué historias de usuario tienen la mayor prioridad y el segundo estima el tiempo que le llevará implementar las mismas.

Uno de los elementos a tener en cuenta en el cronograma de liberación es que rara vez un plan marcha respecto a lo acordado, es por esto que el plan de liberación está en constante cambio. Ejemplos de ello se tienen, cada vez que el usuario cambia la prioridad de una historia, cuando el desarrollador divide o une historias de usuario, cuando aparecen imprevistos en las tecnologías a utilizar, entre otras acciones. Es por esto que estos cambios deben ser aceptados. El cronograma de liberación de la presente investigación queda conformado de la siguiente manera:

Iteración	Fecha de liberación
Primera iteración	1ra semana de Febrero de 2013
Segunda iteración	
Tercera iteración	
Cuarta iteración	3ra semana de Marzo de 2013

Tabla 16 Cronograma de liberación.

2.3.3 Fase de Iteración

Luego de que las historias de usuarios fueron descritas e identificadas, así como estimado el esfuerzo que cada una de ellas conlleva, se procede a realizar la planificación de las etapas de implementación del sistema. Para ello se define un plan que contiene las iteraciones a realizar, así como cuáles historias de usuario serán implementadas y en qué orden para cada iteración, teniéndose en cuenta la duración de las mismas. La cantidad de iteraciones necesarias (IN) para realizar todas las historias de usuario identificadas es obtenida sumando todos los puntos de esfuerzo (PE) de las historias de usuario y dividiéndolos entre la velocidad de iteración del equipo (VIE).

IN = PE + VIE
PE = 127 días (25.4 semanas)
IN = 25.4 / 7.5
IN = 3.38

La velocidad de iteración del equipo (VIE) es obtenida dividiendo la cantidad de desarrolladores (CD) entre el factor de dedicación (FD) al proyecto (en el caso de la presente investigación es de 2 [50%]) y multiplicado por el tiempo de duración máximo de una iteración (DMI) (en el caso de la presente investigación es de 15 días máximo).

*VIE = (CD / FD) * DMI*
CD = 1, FD = 50% (2), DMI = 15
*VIE = (1 / 2) * 15*
VIE = 7.5

De ahí que se obtiene por valor de la velocidad de equipo aproximadamente 8, y las iteraciones necesarias para desarrollar las historias de usuario 4. De manera que el tiempo de desarrollo de las iteraciones este balanceado, se tomaron todas aquellas historias de usuario de la más alta prioridad (1,

2, 3, 4, 5, 6, 7, 9) y se promediaron sus puntos de esfuerzo (34), quedando las iteraciones conformadas de la siguiente manera:

Iteración		
Numero: 1	**H.U (por orden):** 1, 2, 4	**Duración total:** 33
Descripción: Se establece la arquitectura a utilizar en el desarrollo de la aplicación. Se desarrollan las primeras historias de usuario, las que poseen una mayor prioridad, aquellas que conformen un producto que aún sin terminar puedan mostrar las funcionalidades del sistema, dará una primera vista al cliente que suministrará su criterio para la incorporación de nuevos elementos y modificación de los existentes. Al finalizar dicho proceso se contará con las funcionalidades descritas en las historias de usuarios 1, 2 y 4.		

Tabla 17. Primera iteración.

Iteración		
Numero: 2	**H.U (por orden):** 3, 5	**Duración total:** 26
Descripción: Se continúan desarrollando historias de usuario de la más alta prioridad que por cuestión de tiempo no pudieron añadirse a la primera iteración. Se llevan a cabo los señalamientos hechos por el cliente sobre la primera iteración. Al finalizar dicho proceso se contará con las funcionalidades descritas en las historias de usuarios 3 y 5. De esta manera se completa el proceso de gestión de los datos del reporte a generar.		

Tabla 18. Segunda iteración.

Iteración		
Numero: 3	**H.U (por orden):** 6, 7	**Duración total:** 27
Descripción: Se continúan desarrollando historias de usuario de la más alta prioridad que por cuestión de tiempo no pudieron añadirse a la segunda iteración. Se llevan a cabo los señalamientos hechos por el cliente sobre la primera y segunda iteración. Al finalizar dicho proceso se contará con las funcionalidades descritas en las historias de usuarios 6 y 7, de esta manera se completa el proceso de firmado del documento generado, obteniéndose las funcionalidades principales del producto deseado.		

Tabla 19. Tercera iteración.

Iteración		
Numero: 4	**H.U (por orden):** 8, 9, 10, 11 y 12	**Duración total:** 41
Descripción: Se desarrolla una historia de usuario de prioridad alta, 3 de prioridad media y 1 de prioridad baja. Al finalizar dicho proceso se contará con las funcionalidades descritas en las historias de usuarios 8, 9, 10, 11 y 12 en las que se garantiza que la versión de escritorio de SED esté sincronizada con la versión Web, finalizándose de esta manera todas las funcionalidades descritas para la aplicación y obteniéndose el producto final, versión 1.0.		

Tabla 20. Cuarta iteración.

2.3.3 Características no funcionales

Al concebir las historias de usuario no se detallan ciertos aspectos que se deben de precisar en la elaboración de una aplicación. Estos aspectos influyen en gran medida en el funcionamiento del producto final. La metodología *XP* orienta que los detalles de implementación de la historia de usuario se tienen en cuenta en el mismo momento de la concepción, por estas razones se describen a continuación de forma general, algunas de las características no funcionales que debe tener el sistema.

La interfaz del producto a desarrollar debe ser simple, con una navegación por pestañas donde se puedan identificar de forma rápida las acciones a llevar a cabo por los botones presentes. De esta manera se logra una navegación efectiva en la aplicación, pudiéndose alcanzar rápidamente los objetivos por los cuales se está en interacción con el software. Los mensajes deben ser claros, sin ambigüedades y buena ortografía.

Este sistema está destinado principalmente a los técnicos de laboratorio que poseen usuario UCI, los mismos deberán tener conocimientos básicos de computación, facilitándoles un mejor uso de las funcionalidades que brinda la aplicación, la cual debe aprovechar los recursos disponibles en función de mejorar su rendimiento, eficiencia, reflejándose en un mejor tiempo de respuesta (dada en segundos) para una acción. Se emplearán componentes que indiquen al usuario el estado de los procesos que por su complejidad requieran de un tiempo de procesamiento apreciable. La base de datos se encontrará embebida en la aplicación, de esta manera los tiempos de intercambio de datos se reducen al mínimo posible. El tiempo de respuesta está dado por la cantidad de información a procesar, entre mayor cantidad de información mayor será el tiempo de procesamiento. Como tal el sistema no cuenta con muchos gráficos ni imágenes, lo cual contribuye a la reducción del tiempo respuesta.

CAPÍTULO 2: ANÁLISIS DE LA APLICACIÓN

La aplicación funcionará sobre los permisos de seguridad asignados por sistema operativo, internamente se manejan varios aspectos para la protección de la información, como son: encriptación de contraseñas, sincronización de datos e identificación de usuarios por encadenamiento de caracteres y nombre de usuario. Los datos almacenados en el sistema solamente podrán ser manejados por los usuarios autenticados en el mismo, previamente identificados La información manejada dentro del sistema debe ser objeto de una cuidadosa protección contra la corrupción y estados de inconsistencia.

Para su óptimo rendimiento la aplicación requiere una computadora con un mínimo de *512 Mb* de memoria RAM, un espacio mínimo en disco duro de *20 Gb*, la máquina debe tener un microprocesador Pentium 4 en adelante. Para trabajar con el presente software, debe estar instalado en la computadora un sistema operativo dígase *Windows XP* o superior, o *GNU/Linux*. Entre los software corriendo sobre el sistema, la máquina debe tener, la Máquina Virtual de *Java*.

Como producto, la presente aplicación se distribuye amparada bajo las normativas legales establecidas en el registro comercial emitido por las entidades jurídicas de la Universidad de las Ciencias Informáticas.

CONCLUSIONES PARCIALES

Después de investigados los elementos referentes al análisis de la aplicación a desarrollar se concluye que el flujo actual del proceso de entrega y recibo de los laboratorios ante la inexistencia de conexión de red es inseguro y caótico pues los reportes realizados muchas veces son tachados o se pierden, lo cual se traduce en poca fiabilidad al momento de validar dicho documento jurídicamente por la pérdida de algún medio de cómputo. Este escenario propició una rápida identificación de los procesos a automatizar, lográndose identificar de esta manera 6 historias de usuario al inicio, las cuales al ser estimadas fueron divididas, resultando finalmente 12. Una vez estimado el tiempo de duración de cada historia de usuario, se obtuvieron mediante los cálculos de fórmulas propuestas en la metodología *XP,* la cantidad de iteraciones necesarias para llevar a cabo la codificación de las mismas, así como el promedio de duración de las iteraciones, teniendo en cuenta la prioridad de las funcionalidades expuestas por el cliente. De esta manera se llegaron a obtener 4 iteraciones, las cuales fueron organizadas en 2 momentos de liberación del producto, cada uno con funcionalidades completas y listas para trabajar en entornos reales. De esta manera el escenario queda listo para comenzar a realizar el diseño de la aplicación.

CAPÍTULO 3: DISEÑO DE LA APLICACIÓN

INTRODUCCIÓN

Para algunas personas que sólo han tenido un contacto breve con la Programación Extrema, pareciera que la *XP* convoca a la muerte del diseño de software. No solamente se ridiculiza a la actividad de diseño, sino que técnicas como *UML*, marcos flexibles e incluso patrones son menospreciados o simplemente ignorados. Nada más lejos de la realidad pues de hecho la *XP* involucra mucho diseño, pero lo hace de una manera diferente a la de los procesos de software establecidos. La *XP* ha rejuvenecido la noción de diseño evolutivo con prácticas que le permiten una estrategia viable. También brinda nuevos retos y habilidades pues los diseñadores necesitan aprender cómo hacer diseño simple, cómo usar refactorización para mantener el diseño limpio y cómo usar patrones en un estilo evolutivo [39].

El presente capítulo aborda los temas de diseño, pues una vez realizado el análisis para llevar a cabo la implementación, las tareas de esta fase dan vida a lo que más tarde constituirá la arquitectura base del software así como el diseño de sus componentes. Es por esta razón que se proponen temas como los patrones de arquitectura, patrones de diseño y se hace una propuesta de la arquitectura base que brindará soporte a la aplicación. De igual manera se estructuran los diagramas *UML* de diseño de clases y las tarjetas CRC para de una manera global, tener idea de la responsabilidad a cumplir por cada clase, así como sus relaciones con el resto del sistema. Por último se presenta el diagrama entidad – relación de la base de datos que sustentará la información persistente sobre los reportes almacenados.

3.1 PATRONES

Los patrones son simples soluciones a problemas recurrentes que ocurren en el entorno. Luego de llegar a la solución se encapsulan todas las variables y factores de la misma, lo cual conforma una guía para resolver una y otra vez el mismo problema. Entre sus características se encuentran: describir el problema de forma sencilla, describir el contexto en el que ocurre, puntualizar los pasos a seguir, hacer énfasis en los puntos fuertes y débiles de la solución, referir otros patrones asociados, entre otras. En la presente investigación se abordan los patrones arquitectónicos y los patrones de diseño que son utilizados para conformar el diseño de la aplicación propuesta.

3.1.1 Patrones arquitectónicos

Los patrones arquitectónicos son patrones del software, que se encargan de definir la estructura de un sistema. Estos a su vez se componen de subsistemas con sus responsabilidades, también poseen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño. Un patrón arquitectónico se enfoca a dar solución a un problema en específico y abarca solo parte de la arquitectura. Aún cuando un patrón de este tipo brinda una imagen general de un sistema, él no es una arquitectura como tal. Es por esto que un patrón arquitectónico es un concepto que captura elementos esenciales de una arquitectura de software [40].

Patrones arquitectónicos por capas

En su libro *Patrones de Arquitectura para Aplicaciones Empresariales*, Martin Fowler, describe una serie de patrones sobre las diferentes capas que componen la arquitectura en capas, los cuales fueron utilizados como columna vertebral para componer el núcleo de la arquitectura de la aplicación a desarrollar. Dichos patrones se encuentran organizados en dependencia de su funcionalidad, entre las que se utilizaron en la presente investigación destacan: la lógica de negocio, perteneciente a la capa de dominio haciendo uso del patrón Modelo de Dominio, y la persistencia de datos, perteneciente a la capa de acceso a datos haciendo uso de los patrones Asignador de Datos, Identidad de Mapa, Identidad de Campo, Carga Perezosa, Consulta de Objeto.

3.1.2 Patrones de diseño

Los patrones de diseño son descripciones de clases y objetos relacionados que están particularizados para resolver un problema de diseño general en un determinado contexto [41]. Entre los más conocidos se encuentran los patrones de Asignación de Responsabilidades y los patrones de la Banda de los Cuatro.

Patrones GRASP

Los Patrones Generales de Asignación de Responsabilidades de Software (*GRASP*, por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, de forma tal que se pueda diseñar software orientado a objetos [42]. Los mismos no introducen ideas novedosas, son la mera codificación de los principios básicos más usados. Los patrones *GRASP* están compuestos por: Experto, Creador, Bajo Acoplamiento, Alta Cohesión, Controlador, Polimorfismo, Fábrica Pura, Indirección, y “No Hables con Extraños”, también conocido como “Variaciones Protegidas”. Estos principios básicos se tuvieron muy vigentes en el momento de planificar el diseño de los componentes de la aplicación propuesta.

CAPÍTULO 3: DISEÑO DE LA APLICACIÓN

Patrones GoF

Dentro de los patrones de la Banda de los Cuatro [*Gang of Four, (GoF* por sus siglas en inglés)], utilizados en el diseño de la presente investigación para el desarrollo de subsistemas más pequeños se encuentran: Cadenas de Responsabilidades*,* Decorador*,* Fachada *y* Solitario. Los patrones *GoF*, se encuentran agrupados en 3 divisiones en dependencia de su responsabilidad, estas son: comportamiento, creacionales y estructurales.

3.2 PROPUESTA DE ARQUITECTURA

Como se explicó anteriormente en la metodología de desarrollo de software *XP*, se establece en la primera iteración, la propuesta de arquitectura a utilizar durante el transcurso de construcción de la aplicación. Al hacer un análisis a profundidad de las historias de usuario a desarrollar, se detectaron funcionalidades muy similares, lo que dio paso a la idea de reutilizar la mayor cantidad de código posible, para lo cual se desacopló el sistema en componentes más pequeños. Para ello se estableció una arquitectura por capas, basándose en las ventajas que ello provee entre las que se encuentran el poder sustituir capas con otras alternativas sin afectar el resto de la aplicación, minimizar la dependencia de unos componentes con otros, entender una capa sin tener conocimiento de las demás, entre otras. De esta manera queda compuesta la arquitectura de la aplicación a desarrollar por capa de presentación, capa de dominio y capa de acceso a datos, de manera tal que se estructura y organiza la forma de desarrollar el software.

3.2.1 Presentación

En la capa de presentación se utilizó el patrón Modelo Vista Controlador Jerárquico (*HMVC* por sus siglas en inglés), basado en el Modelo Vista Controlador (*MVC*, por sus siglas en inglés), debido a que este último no permite manejar datos, eventos y flujo de aplicación de la forma más óptima cuando se trabaja con elementos de interfaz gráfica de usuario (*GUI*, por sus siglas en inglés), en cambio *HMVC* trabaja de forma eficiente en la capa de presentación, pues descompone toda la jerarquía en capas *MVC* padre-hijo, donde la repetición de este patrón permite estructurar dicha capa. También fueron utilizados otros patrones a más bajo nivel para conformar dicha capa entre los que se encuentran: Capa Supertipo, Cadena de Responsabilidades y Solitario. A continuación se presenta el diagrama de clases general para la capa de presentación:

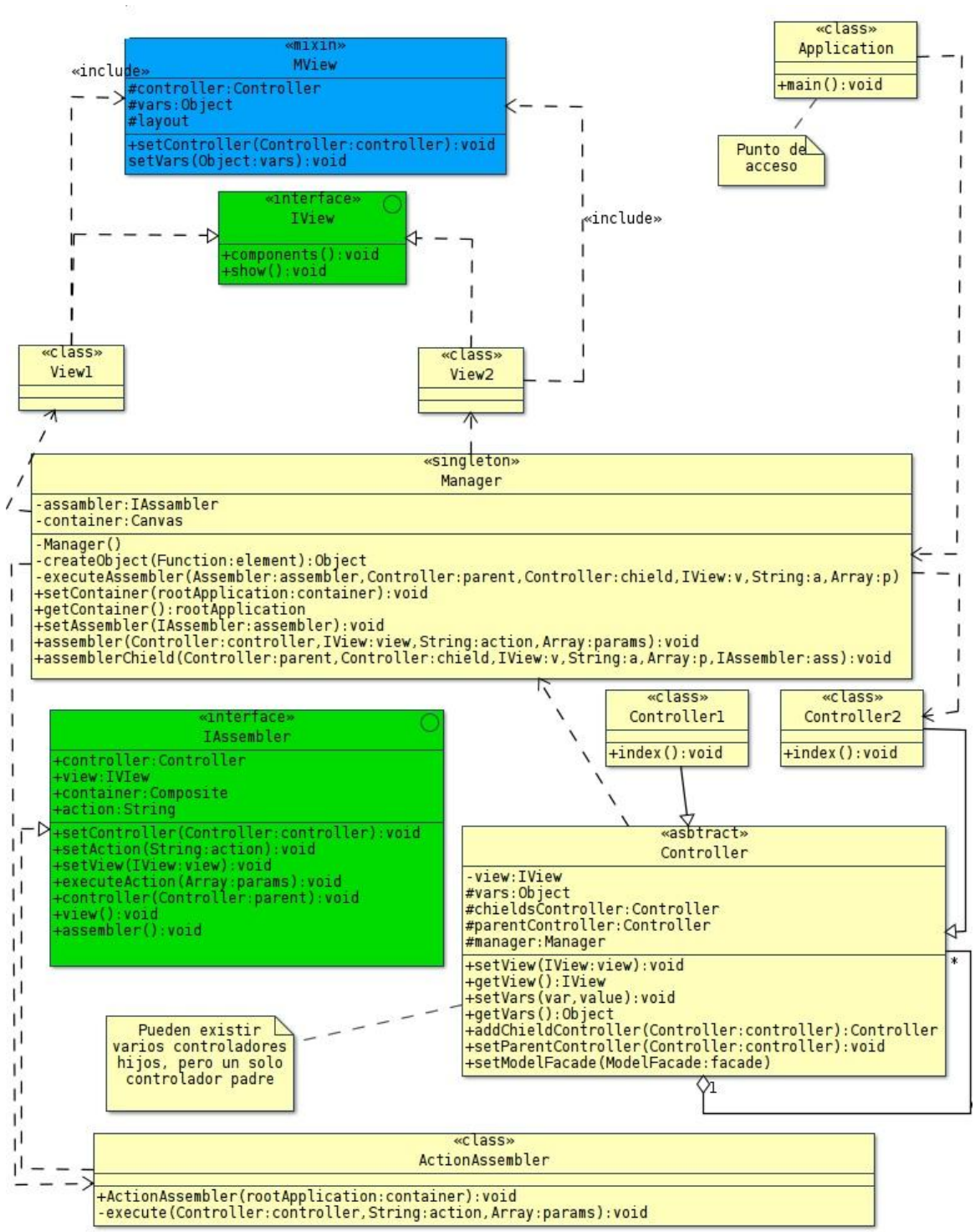

Figura 1. Diagrama de Clases: Capa de presentación.

3.2.2 Dominio

En la capa de dominio fue utilizado el patrón Fachada cuyo propósito es proporcionar una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. Este patrón satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas. De esta manera se promueve un acoplamiento débil entre el subsistema y sus clientes, se reduce el número de objetos con los que el cliente trata, entre otras ventajas. Para la comunicación entre los componentes de dicha capa se utilizaron los patrones: Modelo de Dominio, Capa Supertipo y Solitario. A continuación se presenta el diagrama de clases general para la capa de dominio:

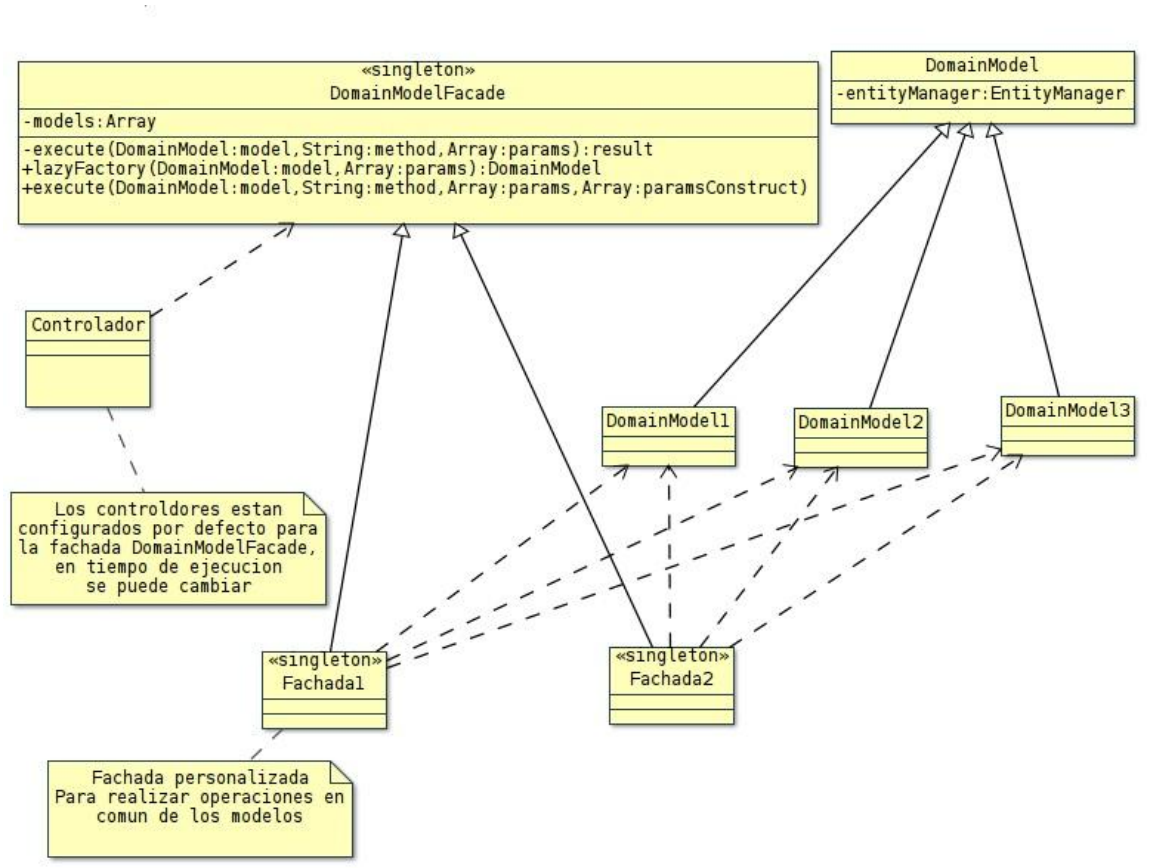

Figura 2. Diagrama de Clases: Capa de dominio.

3.2.3 Acceso a datos

En la capa de acceso a datos se utilizó el patrón Asignador de Datos, el cual permite intercambiar información de un objeto a una base de datos y viceversa, de forma transparente, sin hacer uso del lenguaje *SQL* y todas las actividades que ello conlleva. Para ello se compone de otros patrones a más

bajo nivel, que permiten ejecutar las tareas llevadas a cabo para la conexión e interacción con diferentes gestores de bases de datos, entre ellos se encuentran: Identidad de Mapa, Identidad de Campo, Carga Perezosa, Consulta de Objeto y Capa Supertipo. A continuación se presenta el diagrama de clases general para la capa de acceso a datos:

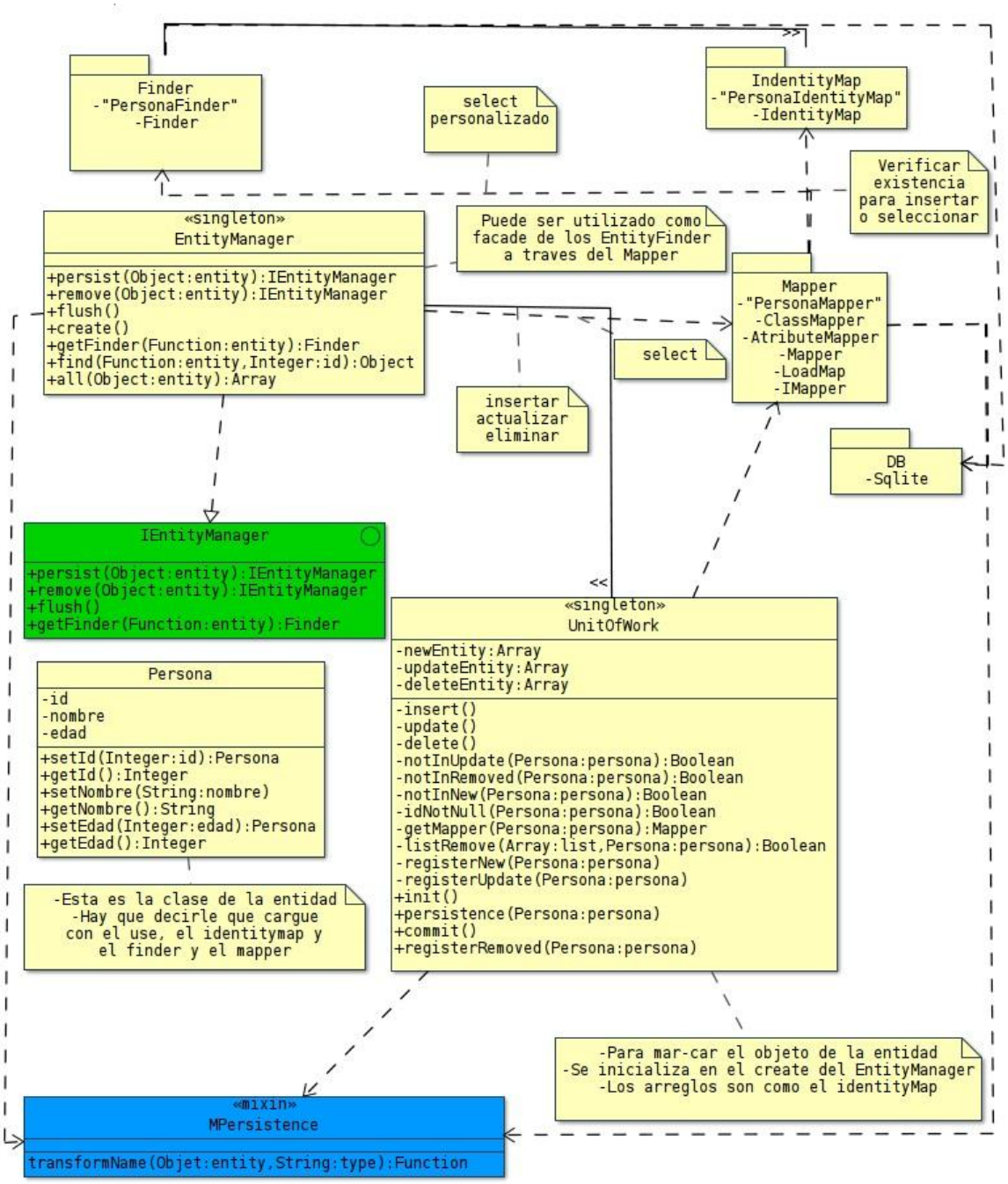

Figura 3. Diagrama de Clases: Capa de acceso a datos.

CAPÍTULO 3: DISEÑO DE LA APLICACIÓN

3.3 TARJETAS CRC

En el momento de diseñar, *XP* propone la idea de llevar a cabo las tareas de la manera más simple posible, aplicando para ello prácticas especializadas que la metodología propone. Dichas prácticas inciden directamente en la realización y elaboración del diseño de un software ya que, cuando se tiene un diseño simple se presta mayor atención a codificar, pues es esta la tarea sobre la cual más énfasis hace esta metodología.

Ciertamente la metodología *XP* no presta especial atención a los diagramas *UML,* pues existe la creencia de que los verdaderos seguidores de *XP* no diagraman. Esta creencia viene dada por 2 causas principales: primero está el hecho de que algunas personas encuentran los diagramas útiles y otras no, y la segunda causa es que los diagramas tienden a asociarse con procesos pesados. Tales procesos invierten mucho tiempo en dibujar diagramas que no ayudan y pueden dañar más que auxiliar. Para hacer un buen uso de los diagramas *UML,* lo primero que se debe tener en cuenta es saber para qué se dibujan, debido a que la comunicación es el principal objetivo de dichos diagramas. Un problema común con el uso de los diagramas es que por lo general se intenta hacerlos extensos, convirtiéndose la exhaustividad en la enemiga de su comprensión [37].

Un uso común de los diagramas *UML* es explorar un diseño antes de empezar a programarlo. Frecuentemente se tiene la impresión de que tal actividad es ilegal en *XP*, pero eso no es cierto. Cambiar el diseño no necesariamente implica cambiar los diagramas. Es perfectamente razonable bosquejar diagramas que te ayuden a entender el diseño y entonces hacerlos a un lado. El dibujarlos ayudó, y eso es suficiente para que valga la pena hacerlos, pero no tienen que ser artefactos permanentes [37].

Como se puede apreciar la metodología *XP* no recomienda la utilización de diagramas de clases, debido a la gran cantidad de documentación e inconvenientes que ello requiere, aunque tampoco excluye su elaboración, en su lugar orienta la realización de Tarjetas de Contenido, Responsabilidad, Colaboración, (*CRC*, por sus siglas en inglés), las cuales en cada una de las iteraciones van siendo definidas como herramientas de reflexión en el diseño de software orientado a objetos. Estas tarjetas brindan un acercamiento a la identificación de clases y la información referente a los objetos desarrollados en el proyecto. Cada tarjeta identifica una clase, sus responsabilidades y relaciones con otras clases.

En la presente investigación se usó una mezcla de tarjetas *CRC* y diagramas *UML,* usando la técnica más útil al momento de transmitir la idea del funcionamiento del sistema. A continuación se presentan los principales diagramas de clases para la lógica de negocio de la aplicación a desarrollar:

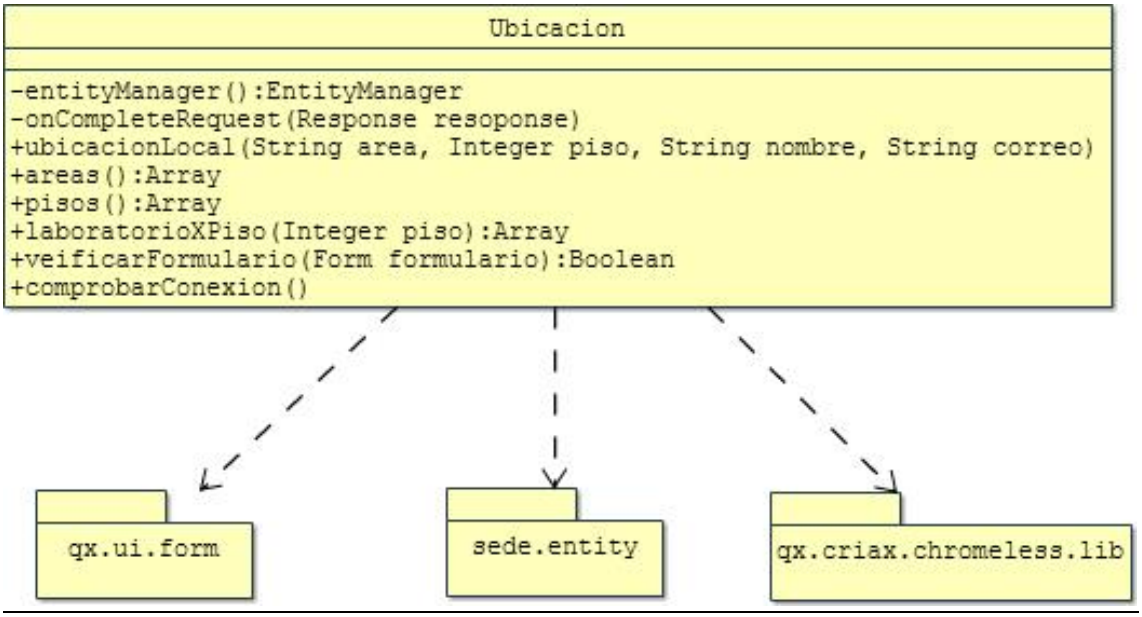

Figura 4. Diagrama de Clases: Autenticar usuario.

Figura 5. Diagrama de Clases: Recibir cantidades.

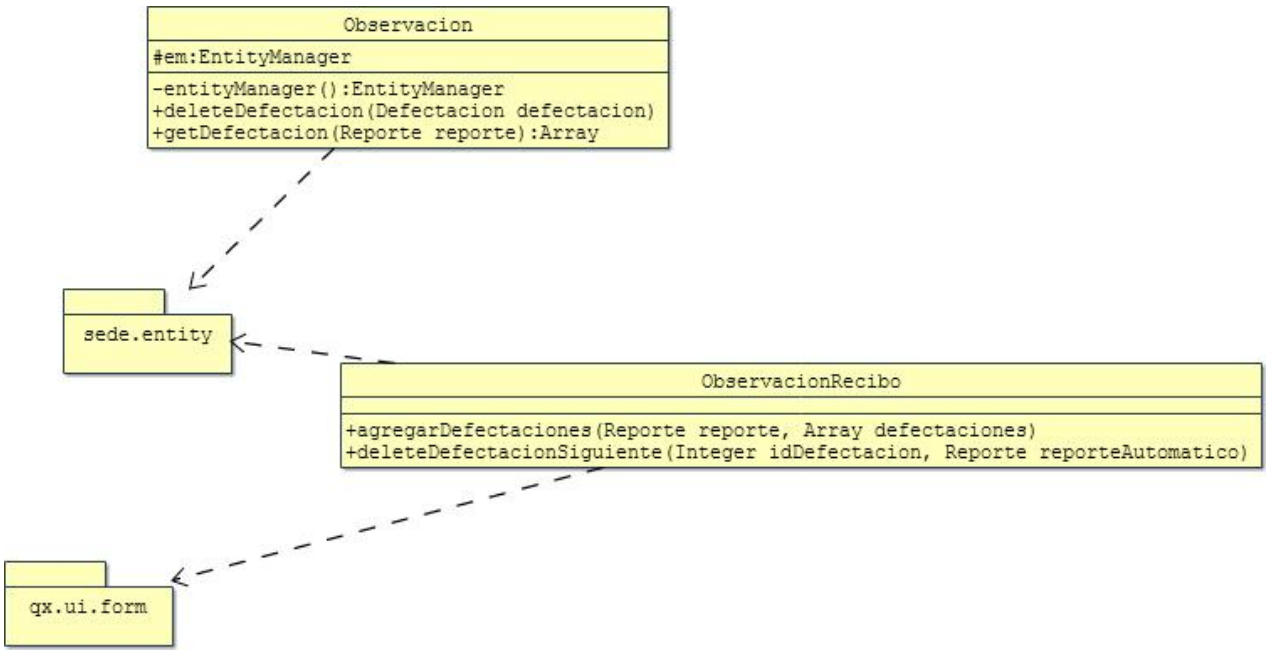

Figura 6. Diagrama de Clases: Recibir observaciones de medios.

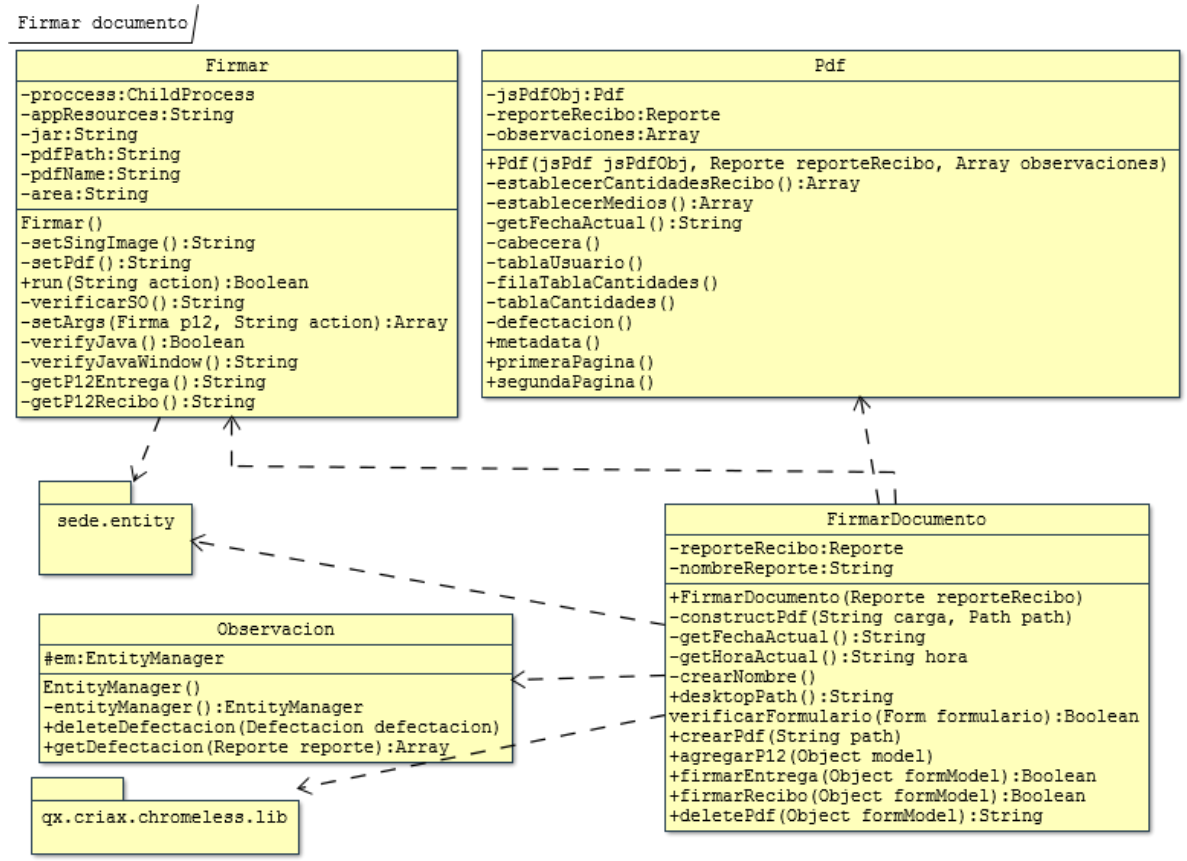

Figura 7. Diagrama de Clases: Firmar reporte.

CAPÍTULO 3: DISEÑO DE LA APLICACIÓN

Con la finalidad de obtener un diseño simple y entendible para aquellos que se dispongan a trabajar, con el fin de no caer en la implementación de características que ya lo están o no son necesarias, se elaboró una tarjeta CRC para cada clase presente en la lógica de negocio, las cuales son listadas a continuación:

Clase: Reporte	
Responsabilidades:	**Colaboraciones:**
1. Obtener características de un laboratorio dado el nombre. 2. Obtener características de un área dado el nombre. 3. Clase base para Recibir, Actualizar, ActualizarCantidades.	

Tabla 21. Tarjeta CRC: Clase Reporte.

Clase: Recibir	
Responsabilidades:	**Colaboraciones:**
1. Crear un recibo a partir del reporte de entrega. 2. Crear entrega automática del reporte de recibo.	1. Reporte

Tabla 22. Tarjeta CRC: Clase Recibir.

Clase: Actualizar	
Responsabilidades:	**Colaboraciones:**
1. Actualizar una entrega del reporte.	1. Reporte

Tabla 23. Tarjeta CRC: Clase Actualizar.

Clase: ActualizarCantidades	
Responsabilidades:	**Colaboraciones:**
1. Actualizar cada uno de los medios de cómputo del reporte.	1. Reporte

Tabla 24. Tarjeta CRC: Clase ActualizarCantidades.

Clase: Observacion	
Responsabilidades:	**Colaboraciones:**
1. Eliminar observación del reporte. 2. Obtener observaciones a partir de un reporte. 3. Clase base para ObservacionActualizar, ObservacionRecibo.	

Tabla 25. Tarjeta CRC: Clase Observacion.

Clase: ObservacionActualizar	
Responsabilidades:	**Colaboraciones:**
1. Obtener los tipos de problemas existentes. 2. Obtener los problemas a partir de un tipo de problema. 3. Adicionar una observación a un reporte de entrega.	1. Observacion

Tabla 26. Tarjeta CRC: Clase ObservacionActualizar.

Clase: ObservacionRecibo	
Responsabilidades:	**Colaboraciones:**
1. Adicionar varias observaciones a un reporte de recibo. 2. Eliminar una observación del reporte de entrega automático.	1. Observacion

Tabla 27. Tarjeta CRC: Clase ObservacionRecibo.

Clase: Pdf	
Responsabilidades:	**Colaboraciones:**
1. Crear pdf de reporte con la información del recibo.	

Tabla 28. Tarjeta CRC: Clase Pdf.

Clase: FirmarDocumento	
Responsabilidades:	**Colaboraciones:**
1. Firmar el reporte de pdf generado.	1. Pdf 2. Observacion

Tabla 29. Tarjeta CRC: Clase FirmarDocumento.

Clase: Ubicacion	
Responsabilidades:	**Colaboraciones:**
1. Almacenar localmente la información del técnico autenticado. 2. Obtener las áreas existentes. 3. Obtener los laboratorios dado un piso. 4. Comprobar funcionamiento de versión Web de SED.	1. TecnicoUbicacion

Tabla 30. Tarjeta CRC: Clase Ubicacion.

Clase: TecnicoUbicacion	
Responsabilidades:	**Colaboraciones:**
1. Establecer la ubicación del técnico.	

Tabla 31. Tarjeta CRC: Clase TecnicoUbicacion.

3.4 DISEÑO DE BASE DE DATOS

En cualquier aplicación en la cual se gestione información, la base de datos desempeña un papel fundamental. El diseño de la misma es algo esencial para el desarrollo de sistemas de este tipo, ya que esta permite el almacenamiento de la información de forma coherente y organizada, para evitar pérdidas e inconsistencias, de esta manera la recuperación de la información es de forma rápida y

flexible. Por tal motivo enfocarse en su desarrollo es una tarea vital. A continuación se muestra el diagrama entidad - relación de la base de datos que se utilizó:

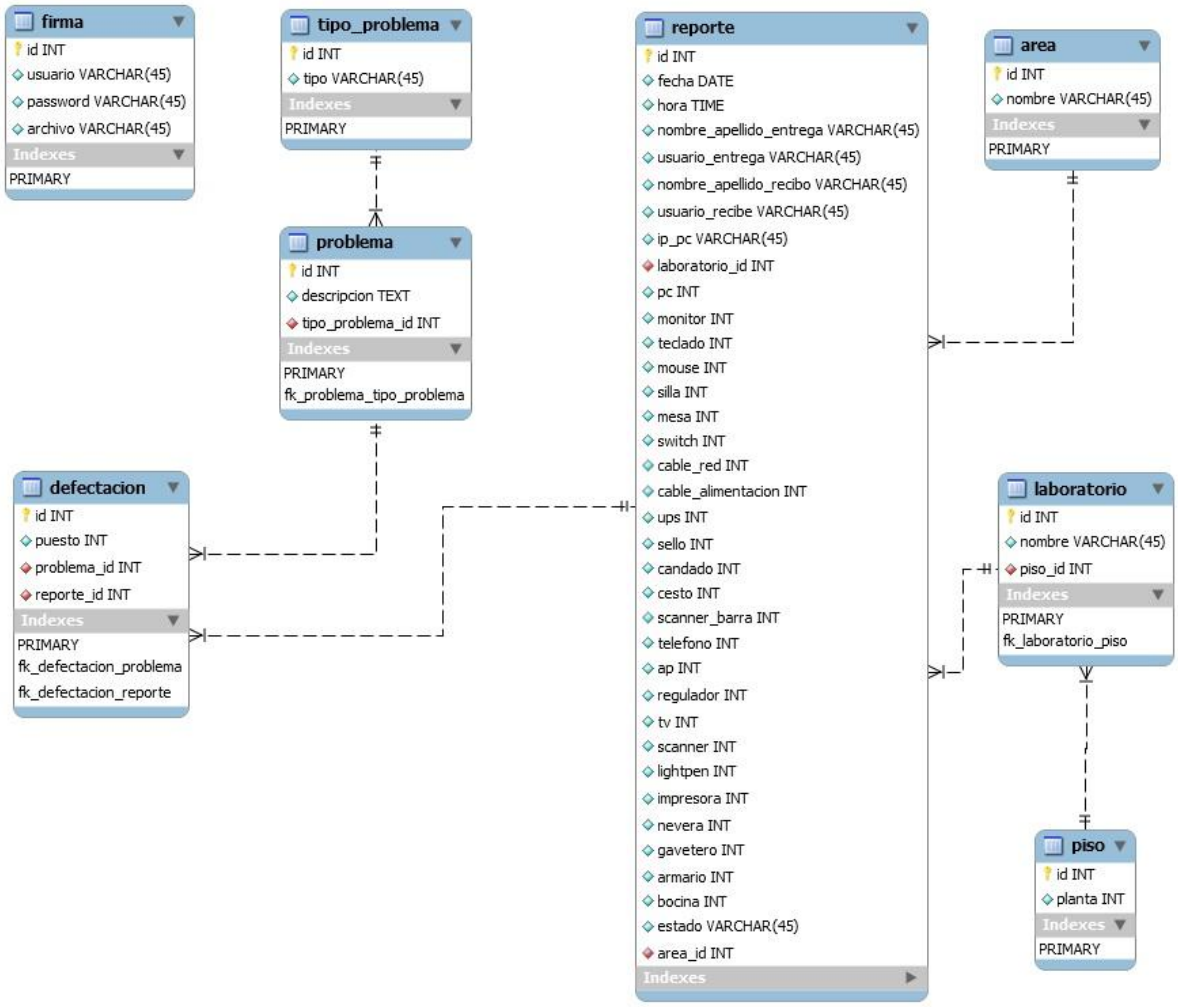

Figura 8. Diagrama Entidad Relación de la Base de Datos.

CONCLUSIONES PARCIALES

Después de investigados los elementos referentes al diseño de la aplicación a desarrollar se pudo concluir que la construcción de un sistema informático debe estar sustentado sobre un buen diseño arquitectónico. De lo contrario, realizar actualizaciones y mejoras al mismo supondría una pérdida cuantiosa de tiempo, pues la refactorización a realizar abarcaría desde el núcleo, hasta las funcionalidades más simples. Es por eso que la presente investigación sustenta su diseño sobre patrones arquitectónicos y de diseño muy bien establecidos, los cuales fueron seleccionados en

dependencia de su función, tenida en cuenta al momento de estructurar las diferentes capas que constituyen un software de este tipo. Aunque la metodología propuesta presenta sus propias técnicas para la elaboración de las tareas de diseño, fue utilizada una combinación de diagramas de clases y tarjetas CRC. Los diagramas *UML*, se utilizaron para brindar una idea global de las piezas más importantes de la aplicación. Para detallar las responsabilidades de cada clase presente en la lógica de negocio fueron utilizadas tarjetas CRC, pues por su simplicidad permiten una rápida actualización en caso de cambios en la concepción del diseño. A manera de tener una idea de cómo fue elaborada la persistencia de la información manejada por la aplicación, se presentó el diagrama entidad relación de la base de datos utilizada. Es así como concluyen las actividades relacionadas con el diseño, dando paso a la implementación de toda la planificación de los diferentes artefactos generados.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS DE LA APLICACIÓN

INTRODUCCIÓN

Después de investigados los elementos referentes al diseño de la aplicación a desarrollar, la siguiente fase es la implementación, guiada por las tareas en las que se disecciona una historia de usuario. El presente capítulo se compone de 2 acápites, Implementación y Pruebas. En la sección de implementación se elaboran las tareas a realizar una vez definidos los artefactos de análisis y diseño, así como las historias de usuario y las iteraciones. Por otra parte la fase de pruebas le permite a los desarrolladores, realizar ensayos constantemente el producto buscando la mayor fiabilidad del mismo, intentando lograr la ausencia de errores en su desarrollo los cuales serán cada vez menores a medida que se avance en la codificación del mismo.

4.1 IMPLEMENTACIÓN

Las tareas llevadas a cabo en la fase de implementación, pueden ser escritas utilizando un lenguaje técnico y no necesariamente deben ser entendibles por el cliente, finalmente son asignadas a un equipo o programador responsable, aunque siempre la codificación se lleva a cabo por una pareja de programadores. Esta labor se efectúa para detallar mejor las historias de usuario, facilitando con ello el entendimiento en el proceso de implementación. Cada historia de usuario puede contener una o más tareas de ingeniería, explicando de forma general las acciones que se realizan en la misma.

4.1.1 Tareas de las historias de usuario

A continuación se presentan las tareas definidas para el desarrollo de la presente investigación:

Tareas de Historia de Usuario		
ID: 1	**H.U:** Autenticar usuario.	**Iteración:**1
Nombre: Establecer arquitectura base.		
Tipo: Desarrollo	**Puntos estimados:** 1	
Fecha inicio: 22/10/12	**Fecha Fin:** 23/10/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Establecer la arquitectura y diseño de clases a utilizar como base en el desarrollo de la aplicación.		

Tabla 32. Tarea: Establecer arquitectura base.

Tareas de Historia de Usuario		
ID: 2	**H.U:** Autenticar usuario.	**Iteración:**1
Nombre: Diseñar Base de Datos.		
Tipo: Desarrollo	**Puntos estimados:** 1	
Fecha inicio: 23/10/12	**Fecha Fin:** 24/10/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Establecer el diseño de la base de datos a utilizar por la aplicación.		

Tabla 33. Tarea: Diseñar Base de Datos.

Tareas de Historia de Usuario		
ID: 3	**H.U:** Autenticar usuario.	**Iteración:**1
Nombre: Establecer prototipo de diseño.		
Tipo: Desarrollo	**Puntos estimados:** 0.5	
Fecha inicio: 24/10/12	**Fecha Fin:** 24/10/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Establecer el prototipo de interfaz de usuario a utilizar en la aplicación.		

Tabla 34. Tarea: Establecer prototipo de diseño.

Tareas de Historia de Usuario		
ID: 4	**H.U:** Autenticar usuario.	**Iteración:**1
Nombre: Diseñar vista de autenticar.		
Tipo: Desarrollo	**Puntos estimados:** 3	
Fecha inicio: 24/10/12	**Fecha Fin:** 27/10/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Desarrollar la interfaz de usuario de la primera vista de la aplicación, centrándose en el formulario de autenticación.		

Tabla 35. Tarea: Diseñar vista de autenticar.

Tareas de Historia de Usuario		
ID: 5	**H.U:** Autenticar usuario.	**Iteración:**1
Nombre: Mapear Base de Datos.		
Tipo: Desarrollo	**Puntos estimados:** 2	
Fecha inicio: 27/10/12	**Fecha Fin:** 29/10/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Crear las clases de persistencia que servirán para el intercambio de información con la Base de Datos.		

Tabla 36. Tarea: Mapear Base de Datos.

Tareas de Historia de Usuario		
ID: 6	**H.U:** Autenticar usuario.	**Iteración:**1
Nombre: Conectar autenticación con Base de Datos.		
Tipo: Desarrollo	**Puntos estimados:** 4	
Fecha inicio: 29/10/12	**Fecha Fin:** 2/11/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Conectar la vista de autenticación con la Base de Datos y validar los datos del formulario.		

Tabla 37. Tarea: Conectar autenticación con Base de Datos.

Tareas de Historia de Usuario		
ID: 7	**H.U:** Actualizar cantidades de medios de un reporte.	**Iteración:**1
Nombre: Diseñar vista de cantidades.		
Tipo: Desarrollo	**Puntos estimados:** 4	
Fecha inicio: 3/11/12	**Fecha Fin:** 7/11/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Desarrollar la interfaz de usuario de la primera vista de las cantidades, centrándose en el formulario de las cantidades.		

Tabla 38. Tarea: Diseñar vista de cantidades.

Tareas de Historia de Usuario		
ID: 8	**H.U:** Actualizar cantidades de medios de un reporte.	**Iteración:**1
Nombre: Actualizar cantidades del reporte de entrega.		
Tipo: Desarrollo	**Puntos estimados:** 8	
Fecha inicio: 7/11/12	**Fecha Fin:** 15/11/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Conectar la vista de cantidades con la Base de Datos.		

Tabla 39. Tarea: Actualizar cantidades del reporte de entrega.

Tareas de Historia de Usuario		
ID: 8	**H.U:** Recibir cantidades de medios de un reporte.	**Iteración:**1
Nombre: Reutilizar diseño de la vista de cantidades.		
Tipo: Desarrollo	**Puntos estimados:** 2	
Fecha inicio: 16/11/12	**Fecha Fin:** 18/11/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Realizarle pequeños cambios a la vista de cantidades para que pueda ser reutilizada en al momento de recibir un laboratorio.		

Tabla 40. Tarea: Reutilizar diseño de la vista de cantidades.

Tareas de Historia de Usuario		
ID: 9	**H.U:** Recibir cantidades de medios de un reporte.	**Iteración:**1
Nombre: Actualizar cantidades del reporte de recibo.		
Tipo: Desarrollo	**Puntos estimados:** 6	
Fecha inicio: 18/11/12	**Fecha Fin:** 24/11/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Conectar la vista de cantidades con la Base de Datos.		

Tabla 41. Tarea: Actualizar cantidades del reporte de recibo.

Tareas de Historia de Usuario		
ID: 10	**H.U:** Recibir cantidades de medios de un reporte.	**Iteración:**1
Nombre: Crear cantidades de la entrega automática.		
Tipo: Desarrollo	**Puntos estimados:** 4	
Fecha inicio: 24/11/12	**Fecha Fin:** 28/11/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Crear el reporte de entrega automática para el técnico que recibió.		

Tabla 42. Tarea: Crear cantidades de la entrega automática.

Tareas de Historia de Usuario		
ID: 11	**H.U:** Actualizar observaciones del reporte.	**Iteración:**2
Nombre: Diseñar vista de observaciones.		
Tipo: Desarrollo	**Puntos estimados:** 5	
Fecha inicio: 29/11/12	**Fecha Fin:** 4/12/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Desarrollar la interfaz de usuario de la primera vista de las observaciones, centrándose en el formulario de las observaciones.		

Tabla 43. Tarea: Diseñar vista de observaciones.

Tareas de Historia de Usuario		
ID: 12	**H.U:** Actualizar observaciones del reporte.	**Iteración:**2
Nombre: Agregar observaciones del reporte de entrega.		
Tipo: Desarrollo	**Puntos estimados:** 5	
Fecha inicio: 4/12/12	**Fecha Fin:** 9/12/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Conectar el formulario de la vista de las observaciones en el reporte de entrega, con la Base de Datos.		

Tabla 44. Tarea: Agregar observaciones del reporte de entrega.

Tareas de Historia de Usuario		
ID: 13	**H.U:** Actualizar observaciones del reporte.	**Iteración:**2
Nombre: Eliminar observaciones del reporte de entrega.		
Tipo: Desarrollo	**Puntos estimados:** 3	
Fecha inicio: 9/12/12	**Fecha Fin:** 12/12/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Listar cada una de las observaciones presentes en el reporte de entrega y dar la posibilidad de eliminarlas una a una.		

Tabla 45. Tarea: Eliminar observaciones del reporte de entrega.

Tareas de Historia de Usuario		
ID: 14	**H.U:** Recibir observaciones del reporte.	**Iteración:**2
Nombre: Reutilizar diseño de la vista de observaciones.		
Tipo: Desarrollo	**Puntos estimados:** 2	
Fecha inicio: 13/12/12	**Fecha Fin:** 15/12/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Realizarle pequeños cambios a la vista de observaciones para que pueda ser reutilizada en al momento de recibir un laboratorio.		

Tabla 46. Tarea: Reutilizar diseño de la vista de observaciones.

Tareas de Historia de Usuario		
ID: 15	**H.U:** Recibir observaciones del reporte.	**Iteración:**2
Nombre: Agregar observaciones al reporte de recibo.		
Tipo: Desarrollo	**Puntos estimados:** 3	
Fecha inicio: 15/12/12	**Fecha Fin:** 18/12/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Conectar el formulario de la vista de las observaciones en el reporte de recibo con la Base de Datos.		

Tabla 47. Tarea: Agregar observaciones al reporte de recibo.

Tareas de Historia de Usuario		
ID: 16	**H.U:** Recibir observaciones del reporte.	**Iteración:**2
Nombre: Eliminar observaciones del reporte de recibo.		
Tipo: Desarrollo	**Puntos estimados:** 1	
Fecha inicio: 18/12/12	**Fecha Fin:** 19/12/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Listar cada una de las observaciones presentes en el reporte de recibo y dar la posibilidad de eliminarlas una a una.		

Tabla 48. Tarea: Eliminar observaciones del reporte de recibo.

Tareas de Historia de Usuario		
ID: 17	**H.U:** Recibir observaciones del reporte.	**Iteración:**2
Nombre: Agregar observaciones a la entrega automática.		
Tipo: Desarrollo	**Puntos estimados:** 4	
Fecha inicio: 19/12/12	**Fecha Fin:** 23/12/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Obtener la observación que se agregó al reporte de recibo y adicionarla a la entrega automática.		

Tabla 49. Tarea: Agregar observaciones a la entrega automática.

Tareas de Historia de Usuario		
ID: 17	**H.U:** Recibir observaciones del reporte.	**Iteración:**2
Nombre: Eliminar observaciones de la entrega automática.		
Tipo: Desarrollo	**Puntos estimados:** 3	
Fecha inicio: 23/12/12	**Fecha Fin:** 26/12/12	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Obtener la observación que se eliminó del reporte de recibo y eliminarla en la entrega automática.		

Tabla 50. Tarea: Eliminar observaciones de la entrega automática.

Tareas de Historia de Usuario		
ID: 18	**H.U:** Generar Reporte.	**Iteración:**3
Nombre: Crear archivo pdf.		
Tipo: Desarrollo	**Puntos estimados:** 7	
Fecha inicio: 27/12/12	**Fecha Fin:** 7/1/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Crear un documento pdf de 2 páginas, con un formato específico de nombre y salvarlo en el escritorio.		

Tabla 51. Tarea: Generar archivo pdf.

Tareas de Historia de Usuario		
ID: 19	**H.U:** Generar Reporte.	**Iteración:**3
Nombre: Establecer las cantidades en el documento pdf.		
Tipo: Desarrollo	**Puntos estimados:** 4	
Fecha inicio: 7/1/13	**Fecha Fin:** 11/1/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Poner en la primera página del documento las cantidades tanto de entrega como de recibo.		

Tabla 52. Tarea: Establecer las cantidades en el documento pdf.

Tareas de Historia de Usuario		
ID: 20	**H.U:** Generar Reporte.	**Iteración:**3
Nombre: Establecer las observaciones en el documento pdf.		
Tipo: Desarrollo	**Puntos estimados:** 4	
Fecha inicio: 11/1/13	**Fecha Fin:** 15/1/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Poner en la segunda página del documento las observaciones.		

Tabla 53. Tarea: Establecer las observaciones en el documento pdf.

Tareas de Historia de Usuario		
ID: 21	**H.U:** Firmar Reporte.	**Iteración:**3
Nombre: Almacenar datos de la firma del técnico que entrega.		
Tipo: Desarrollo	**Puntos estimados:** 4	
Fecha inicio: 16/1/13	**Fecha Fin:** 20/1/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Almacenar en la base de datos de forma encriptada los datos de la firma del técnico que entrega.		

Tabla 54. Tarea: Almacenar datos de la firma del técnico que entrega.

Tareas de Historia de Usuario		
ID: 22	**H.U:** Firmar Reporte.	**Iteración:**3
Nombre: Colocar firma del técnico que entrega en el pdf.		
Tipo: Desarrollo	**Puntos estimados:** 4	
Fecha inicio: 20/1/13	**Fecha Fin:** 24/1/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Plasmar en el pdf la firma del técnico que entrega		

Tabla 55. Tarea: Colocar firma del técnico que entrega en el pdf.

Tareas de Historia de Usuario		
ID: 23	**H.U:** Firmar Reporte.	**Iteración:**3
Nombre: Colocar firma del técnico que recibe en el pdf.		
Tipo: Desarrollo	**Puntos estimados:** 4	
Fecha inicio: 24/1/13	**Fecha Fin:** 28/1/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Plasmar en el pdf la firma del técnico que recibe y borrar los datos de la firma del técnico que entrega.		

Tabla 56. Tarea: Colocar firma del técnico que recibe en el pdf.

Tareas de Historia de Usuario		
ID: 24	**H.U:** Verificar SED.	**Iteración:**4
Nombre: Verificar que el SED se encuentre en línea.		
Tipo: Desarrollo	**Puntos estimados:** 5	
Fecha inicio: 29/1/13	**Fecha Fin:** 3/2/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Establecer un botón que muestre un mensaje para saber si el SED está en línea o no.		

Tabla 57. Tarea: Verificar que el SED se encuentre en línea.

Tareas de Historia de Usuario		
ID: 25	**H.U:** Sincronizar reportes.	**Iteración:**4
Nombre: Seleccionar la información de los reportes a sincronizar.		
Tipo: Desarrollo	**Puntos estimados:** 8	
Fecha inicio: 4/2/13	**Fecha Fin:** 12/2/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Se selecciona la información que se enviará por petición POST al servidor de SED.		

Tabla 58. Tarea: Seleccionar la información de los reportes a sincronizar.

Tareas de Historia de Usuario		
ID: 26	**H.U:** Sincronizar reportes.	**Iteración:**4
Nombre: Enviar la información encriptada de los reportes al servidor de SED.		
Tipo: Desarrollo	**Puntos estimados:** 7	
Fecha inicio: 12/2/13	**Fecha Fin:** 19/2/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Enviar la información al servidor del SED encriptada de manera reversible.		

Tabla 59. Tarea: Enviar la información encriptada de los reportes al servidor de SED.

Tareas de Historia de Usuario		
ID: 27	**H.U:** Sincronizar áreas.	**Iteración:**4
Nombre: Obtener la información exportada por el SED sobre las áreas.		
Tipo: Desarrollo	**Puntos estimados:** 3	
Fecha inicio: 20/2/13	**Fecha Fin:** 23/2/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Se obtendrá la información de las áreas existentes en el SED en formato json.		

Tabla 60. Tarea: Obtener la información exportada por el SED sobre las áreas.

Tareas de Historia de Usuario		
ID: 28	**H.U:** Sincronizar áreas.	**Iteración:**4
Nombre: Almacenar las áreas obtenidas.		
Tipo: Desarrollo	**Puntos estimados:** 2	
Fecha inicio: 23/2/13	**Fecha Fin:** 25/2/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Agregar las áreas a la aplicación		

Tabla 61. Tarea: Almacenar las áreas obtenidas.

Tareas de Historia de Usuario		
ID: 29	**H.U:** Sincronizar áreas.	**Iteración:**4
Nombre: Eliminar reportes antiguos.		
Tipo: Desarrollo	**Puntos estimados:** 2	
Fecha inicio: 25/2/13	**Fecha Fin:** 27/2/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Eliminar los reportes antiguos y dejar solamente la última entrega realizada, sincronizada esta con su área.		

Tabla 62. Tarea: Eliminar reportes antiguos.

Tareas de Historia de Usuario		
ID: 30	**H.U:** Sincronizar problemas.	**Iteración:**4
Nombre: Obtener la información exportada por el SED sobre los problemas.		
Tipo: Desarrollo	**Puntos estimados:** 3	
Fecha inicio: 28/2/13	**Fecha Fin:** 3/3/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Se obtendrá la información de los problemas existentes en el SED en formato *JSON*.		

Tabla 63. Tarea: Obtener la información exportada por el SED sobre los problemas.

Tareas de Historia de Usuario		
ID: 31	**H.U:** Sincronizar problemas.	**Iteración:**4
Nombre: Almacenar los problemas obtenidos.		
Tipo: Desarrollo	**Puntos estimados:** 2	
Fecha inicio: 3/3/13	**Fecha Fin:** 5/3/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Agregar los problemas a la aplicación		

Tabla 64. Tarea: Almacenar los problemas obtenidos.

Tareas de Historia de Usuario		
ID: 32	**H.U:** Sincronizar observaciones.	**Iteración:**4
Nombre: Obtener la información exportada por el SED sobre los tipos de problemas.		
Tipo: Desarrollo	**Puntos estimados:** 3	
Fecha inicio: 6/3/13	**Fecha Fin:** 9/3/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Se obtendrá la información de los tipos de problemas existentes en el SED en formato JSON.		

Tabla 65. Tarea: Obtener la información exportada por el SED sobre los tipos de problemas.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS DE LA APLICACIÓN

Tareas de Historia de Usuario		
ID: 33	**H.U:** Sincronizar observaciones.	**Iteración:**4
Nombre: Almacenar los tipos de problemas obtenidos.		
Tipo: Desarrollo	**Puntos estimados:** 3	
Fecha inicio: 9/3/13	**Fecha Fin:** 12/3/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Se obtendrá la información de los tipos de problemas existentes en el SED en formato JSON.		

Tabla 66. Tarea: Almacenar los tipos de problemas obtenidos.

Tareas de Historia de Usuario		
ID: 34	**H.U:** Sincronizar observaciones.	**Iteración:**4
Nombre: Eliminar reportes antiguos.		
Tipo: Desarrollo	**Puntos estimados:** 2	
Fecha inicio: 12/3/13	**Fecha Fin:** 14/3/13	
Responsable: Nilmar Sánchez Muguercia		
Descripción: Eliminar los reportes antiguos y dejar solamente la última entrega realizada, sincronizada las observaciones de la misma con los tipos de problemas.		

Tabla 67. Tarea: Eliminar reportes antiguos.

4.1.2 Conclusiones de la implementación

Las tareas de implementación llevadas a cabo sirvieron como soporte organizativo en desarrollo de la codificación del software propuesto. Las mismas permitieron distribuir de forma organizada la programación entre los miembros del equipo de desarrollo.

4.2 PRUEBAS

Entre las prácticas de *XP*, se encuentra la de llevar a cabo un Desarrollo Guiado por Pruebas (*TDD*, por sus siglas en inglés), pues esta reduce el número de errores no detectados, así como el tiempo entre la introducción de estos en el sistema y su descubrimiento. Es por esta razón que el producto en desarrollo está siendo testeado constantemente, ya que la funcionalidad que no pueda ser demostrada con un test automatizado, simplemente no existe. En *XP,* se pueden clasificar las pruebas en 2 grupos principalmente, las pruebas unitarias y las pruebas de aceptación.

4.2.1 Pruebas unitarias

Los pruebas unitarias o test unitarios son los más importantes para el practicante *TDD*. Cada test unitario o test unidad es un paso andado en el camino de la implementación del software. Todo test unitario debe ser: atómico, independiente, inocuo y rápido. Si no cumple estas premisas entonces no es un test unitario, aunque se ejecute con una herramienta tipo *xUnit*. Para conseguir cumplir estos requisitos, un test unitario aísla la parte del programa que necesita ejecutar, de tal manera que el resto está inactivo durante la ejecución [43]. Cabe destacar que una de las actividades que conforma la fase de pruebas en *XP* es la refactorización, constante actividad de reestructuración del código. Dicha actividad tiene como objetivo remover la duplicidad, mejorar la legibilidad, simplificar y hacer más flexible la codificación, para facilitar los posteriores cambios.

4.2.2 Pruebas de aceptación

Las pruebas de aceptación o test de aceptación permiten comprobar que el software cumple con un requisito de negocio. Una funcionalidad escrita con el lenguaje del cliente pero que puede ser ejecutada por la máquina. Estas pruebas son creadas a partir de las historias de usuario. Las mismas son el punto de partida del desarrollo en cada iteración [43]. Permiten además comprobar que la funcionalidad testeada sea la esperada por el cliente. Este tipo de pruebas funcionan como una caja negra pues cada una de ellas representa una salida esperada del sistema, donde es responsabilidad del cliente verificar la corrección de las pruebas y tomar decisiones acerca de las mismas. A continuación se detallan algunas de las pruebas llevadas a cabo a la aplicación desarrollada:

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS DE LA APLICACIÓN

Prueba de aceptación		
Código: H.U1_P.A1	**H.U:** Autenticar usuario.	
Nombre: Identificar usuario autenticado.		
Descripción: Según los datos entrados por el usuario obtener el nombre de usuario procesado.		
Condiciones de Ejecución: El sistema almacena durante el proceso de entrega o recibo, los datos del usuario autenticado.		
Entrada / Pasos de ejecución: El usuario llena los datos para la entrega o recibo, los mismos son procesados y almacenados durante todo el proceso.		
Resultados esperados:	**Escenario positivo:** Datos correctos	**Escenario negativo:** Datos incorrectos
	Es procesado y almacenado correctamente el usuario del técnico.	No puede ser almacenado el usuario del técnico pues los datos fueron llenados incorrectamente, señalando el sistema los campos con borde rojo.
Evaluación: Satisfactoria		

Tabla 68. Prueba de Aceptación: H.U1_P.A1.

Prueba de aceptación		
Código: H.U2_P.A1	**H.U:** Actualizar cantidad de medios del reporte de entrega.	
Nombre: Actualizar entrega.		
Descripción: El técnico que está de guardia, actualiza las cantidades de algunos de los medios, así como las observaciones, eliminando algunas y adicionando otras, sube su p12 al sistema.		
Condiciones de Ejecución: A medida que van siendo actualizadas las cantidades se van modificando en la base de datos, para no tener que esperar al final del proceso, se almacena en uno de los directorios de la aplicación el p12 del técnico en turno, así como se ofusca su información en la base de datos.		
Entrada / Pasos de ejecución: El técnico va actualizando las cantidades de los medios, el técnico elimina o adiciona una observación, el técnico sube su p12 al sistema.		
Resultados esperados:	**Escenario positivo:** Datos correctos	**Escenario negativo:** Datos incorrectos
	Si el archivo p12 que se sube es correcto, el técnico puede terminar la actualización.	Si el archivo que se sube no es correcto se le muestra un mensaje de error al técnico.
Evaluación: Satisfactoria		

Tabla 69. Prueba de Aceptación: H.U2_P.A1.

Prueba de aceptación		
Código: H.U4_P.A1	**H.U:** Recibir cantidades de medios del reporte de recibo.	
Nombre: Agregar una nueva entrega.		
Descripción: Cuando el técnico recibe se crea un reporte de entrega nuevo automático.		
Condiciones de Ejecución: A partir del reporte de recibo el sistema debe generar automáticamente un reporte de entrega nuevo.		
Entrada / Pasos de ejecución: El técnico que recibe actualiza las cantidades, así como las observaciones, adicionando o eliminando aquellas que lo requieran, sube su p12 y genera el reporte de recibo, automáticamente se crea un reporte de entrega que le servirá durante su turno de guardia.		
Resultados esperados:	**Escenario positivo:** Datos correctos	**Escenario negativo:** Datos incorrectos
	Si el archivo p12 que se sube es correcto, el técnico puede terminar de recibir.	Si el archivo que se sube no es correcto se le muestra un mensaje de error al técnico que recibe.
Evaluación: Satisfactoria		

Tabla 70. Prueba de Aceptación: H.U4_P.A1.

Prueba de aceptación		
Código: H.U6_P.A1	**H.U:** Generar reporte.	
Nombre: Crear un pdf.		
Descripción: Cuando se recibe se genera un reporte en pdf.		
Condiciones de Ejecución: Los datos del técnico que entrega y del técnico que recibe deben estar llenos.		
Entrada / Pasos de ejecución: Una vez que el técnico que recibe lleno sus datos, pasa a generar el reporte, se obtiene ambos juegos de datos, se crea un pdf en la ruta seleccionada por el técnico y se escriben los datos en él.		
Resultados esperados:	**Escenario positivo:** Datos correctos	**Escenario negativo:** Datos incorrectos
	Se genera un pdf con las cantidades y las observaciones del técnico que recibe y así como del que entrega.	Si el técnico no recibe no se genera el pdf, en caso de Linux el directorio seleccionado para guardar el pdf tiene que tener permiso de escritura, en caso contrario se le muestra un mensaje al usuario.
Evaluación: Satisfactoria		

Tabla 71. Prueba de Aceptación: H.U6_P.A1.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS DE LA APLICACIÓN

Prueba de aceptación		
Código: H.U7_P.A1	**H.U:** Firmar reporte.	
Nombre: Firmar un pdf.		
Descripción: Firmar digitalmente un documento pdf.		
Condiciones de Ejecución: Debe generarse el pdf a firmar.		
Entrada / Pasos de ejecución: Tomar los datos de la firma del técnico que entrega y firmar el pdf. Tomar los datos provistos por el técnico que recibe y firmar el reporte. Una vez que se firmo bien el pdf, se almacenan los datos del técnico que recibe en la base de datos. Se elimina los datos del técnico que entrega.		
Resultados esperados:	**Escenario positivo:** Datos correctos	**Escenario negativo:** Datos incorrectos
	Datos del técnico que entrega y técnico que recibe, son correctos se firma pdf.	Datos del técnico que entrega y técnico que recibe, son incorrectos se muestra mensaje de error en cada caso.
Evaluación: Satisfactoria		

Tabla 72. Prueba de Aceptación: H.U7_P.A1.

Prueba de aceptación		
Código: H.U9_P.A1	**H.U:** Sincronizar áreas.	
Nombre: Obtener datos de *URL* en formato *JSON*.		
Descripción: Obtener y procesar datos obtenidos de un archivo *JSON*.		
Condiciones de Ejecución: Debe generarse el pdf a firmar.		
Entrada / Pasos de ejecución: A partir de una URL JSON procesar los datos obtenidos, actualizar la base de datos con los datos nuevos y eliminar los antiguos.		
Resultados esperados:	**Escenario positivo:** Datos correctos	**Escenario negativo:** Datos incorrectos
	El *JSON* está bien formateado y sus datos son guardados en la base de datos.	El JSON es obtenido directamente de la base de datos central, por lo cual no hay manera de que los datos sean incorrectos.
Evaluación: Satisfactoria		

Tabla 73. Prueba de Aceptación: H.U9_P.A1.

4.2.3 Registro de no conformidades

Uno de los detalles a no pasar por alto al momento de realizar las pruebas son las no conformidades detectadas, las cuales se traducen en los errores encontrados y funcionalidades no deseados por el cliente. Al final de cada iteración se le muestra al cliente una versión funcional del software de forma que pueda detectar aquellas no conformidades que serán corregidas al inicio de la subsiguiente iteración. La presente investigación está dividida en 4 iteraciones, a continuación se listan las no conformidades encontradas en cada una de ellas:

No conformidades	
Iteración:	
Primera	1. Errores de falta de ortografía en los mensajes de la pantalla de autenticación. 2. Se detiene el indicador de carga. 3. Retardo de 4-7 segundos al cancelar el recibo. 4. No se especifica en el mensaje mostrado al usuario los errores que tiene el formulario al autenticarse. 5. El nombre del usuario no se pasa a mayúsculas.
Segunda	1. Retardo en la carga de las pestañas. 2. Pérdida de datos. 3. Mezcla de datos. 4. Inconsistencia en ventanas emergentes.
Tercera	1. Error en la sincronización de las áreas. 2. Después de sincronizados los problemas, estos no cargan en la pestaña de observaciones. 3. Después de sincronizados los tipos de problemas estos no cargan en la pestaña de observaciones. 4. Al ocurrir un error en la generación del documento por parte del técnico que entrega no muestra mensaje. 5. Retardo en sistema operativo Windows.
Cuarta	1. Errores de concordancia en los mensajes. 2. Mensajes de error poco claros.

Tabla 74. Registro de no conformidades.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS DE LA APLICACIÓN

Como parte de la metodología *XP,* las no conformidades encontradas en cada iteración son las primeras tareas a resolver de la iteración siguiente, siendo el cliente el encargado de ordenarlas por prioridad. Algunas de ellas al no ser críticas, son arrastradas a la siguiente iteración. Llevando a cabo este proceso, se logran minimizar los niveles de aceptación de errores. De esta manera quedaron resueltas las no conformidades detectadas en la aplicación desarrollada.

4.2.4 Conclusiones de las pruebas

En la presente investigación las pruebas permitieron determinar en tiempo los errores introducidos en la aplicación. Una de las ventajas que provee *XP* al realizar las pruebas, es la posibilidad de una vez mostrado el avance de la aplicación al cliente, este puede identificar las inconformidades existentes las cuales el equipo de desarrollo corrige de inmediato, ya que las mismas poseen una batería de test en su base.

CONCLUSIONES PARCIALES

Concluido el estudio de las fases de implementación y pruebas se puede afirmar que el desarrollo de la aplicación de la presente investigación está basado sobre una codificación simple y fiable, argumento que se apoya en la seguridad ofrecida por las pruebas realizadas al software a medida que se fue confeccionando. Las mismas arrojaron diversas no conformidades que ayudaron a mejorar el producto elaborado. Las tareas realizadas en esta fase están muy bien delimitadas y organizadas, propósito que se logra gracias a la granularidad que se lleva a cabo en las historias de usuario, al desarrollar poco a poco los pasos necesarios para darle un acabado con grandes posibilidades de éxito.

CONCLUSIONES

CONCLUSIONES

Se culmina la presente investigación logrando cumplir los objetivos trazados con técnicas y metodologías para cada uno de ellos. En vista de lograr la mejor aproximación a estos, se puede arribar a los siguientes resultados:

1. Se investigó la existencia de soluciones como punto de partida para definir la adopción de alguna de ellas o la implementación de una propia.
2. Se pudo evidenciar que el desarrollo de aplicaciones *RIA* embebidas en el escritorio no está difundido del todo, pues si es cierto que la tecnología y su enfoque son relativamente novedosos, por lo general los desarrolladores suelen ser bastante esquemáticos en las herramientas y tecnologías a utilizar. Su mayor ventaja radica en utilizar lo mejor de ambos entornos de desarrollo, obteniendo la velocidad, riqueza visual e integración de las aplicaciones de escritorio y desarrollando sobre tecnologías Web, de esta manera se obtiene un producto de alta calidad.
3. La integración de las tecnologías de *Chromeless* y *Qooxdoo* es el principio de una base sólida para el desarrollo de aplicaciones *RIA* embebidas en el escritorio.
4. Para el cumplimiento de los objetivos y en concordancia con las necesidades expuestas por el cliente, se requirió hacer un estudio de la situación actual del proceso de gestión de entrega y recibo de los medios de cómputo de los laboratorios de la Universidad de las Ciencias Informáticas, para con ello elevar la eficiencia del proceso en cuestión.
5. Son palpables las ventajas obtenidas al utilizar la metodología *XP* que permitió una codificación y proceso ingenieril rápido y centrado en la codificación de la solución propuesta. El ahorro de tiempo fue clave debido al corto tiempo de desarrollo disponible.
6. El uso de patrones de diseño y arquitectónicos brindó como resultado un diseño solido y flexible para la codificación de la aplicación.
7. Las pruebas realizadas al software se convirtieron en el punto principal al momento de la codificación, pues *TDD* permite comprobar la aplicación al mismo tiempo que está siendo desarrollada.
8. Resultado de la presente investigación se obtuvo una herramienta informática que permite gestionar los procesos de entrega y recibo de los laboratorios de la UCI en los momentos que falla la conexión de red.

RECOMENDACIONES

Como resultado del proceso de investigación y realización de la presente investigación se sugieren aspectos que serían recomendables a tener en cuenta para el futuro perfeccionamiento del sistema:

1. Implementar nuevas funcionalidades dirigidas a las necesidades de gestión local de áreas y problemas.
2. Mantener una constante interacción con los usuarios de la aplicación que propicie una retroalimentación, dando así margen a la corrección de posibles errores que no hayan sido detectados.
3. Desplegar el sistema en los laboratorios de la Universidad, para garantizar la continuidad del servicio una vez que la conexión de red falle.
4. Integrar la solución con el SED, permitiendo la sincronización total de la información del mismo con la aplicación de escritorio.

BIBLIOGRAFÍA

1. Mazalán Comunicaciones: Desarrolladores Web mexicanos se reúnen en el primer DEVELOPER DAY para conocer nuevas herramientas para mejorar la web [Consultado el: 2 de Diciembre de 2012]. Disponible en: http://www.mazalan.com/nota/1122/Desarrolladores-Web-mexicanos-se-reunen-en-el-primer-DEVELOPER-DAY-para-conocer-nuevas-herramientas-para-mejorar-la-Web.html.
2. Ortiz, Antonio. Error 500: El futuro de la Web y el escritorio 2007 Última actualización: 29 de mayo de 2007. [Consultado el: 5 de Noviembre de 2012]. Disponible en: http://www.error500.net/futuro-web-escritorio/.
3. Pimentel, Victor. Genbeta: Google Gears, la apuesta de Google por acceder a sus aplicaciones desconectado Última actualización: 31 de Mayo de 2007. [Consultado el: 6 de Diciembre de 2012]. Disponible en: http://www.genbeta.com/web/google-gears-la-apuesta-de-google-para-acceder-a-sus-aplicaciones-desconectado.
4. Ecured: Historia de la Informática en Cuba. [Consultado el: 25 de Noviembre de 2012]. Disponible en: http://www.ecured.cu/index.php/Historia_de_la_Inform%C3%A1tica_en_Cuba#La_UCI.
5. Cornelio, Omar Mar. Sistema de Entrega de Guardia para los Laboratorios de la UCI publicado el: 25 de Noviembre de 2011 de 2011, última actualización: 25 de Noviembre de 2011. Disponible en: http://semanatecnologica.fordes.co.cu/index.php/xist/xi/paper/view/465/261.
6. Baquero, Carlos Dorian Añazco. Emagister: Administración de una bodega, Capítulo 12 Control de Inventarios 2010. Última actualización: 6 de Agosto de 2010. [Consultado el: 6 de Diciembre de 2012]. Disponible en: http://www.emagister.com/curso-administracion-bodega/control-inventarios.
7. Martha Lozada, Diana Elizabeth Salguero Garzón, Aracely Margoth Tobar Molina. Diseño de un sistema de control de inventarios de activos fijos en el Departamento Financiero de la Fuerza Aérea Ecuatoriana ALA No. 12. LATACUNGA, 2007, Disponible en: http://repositorio.espe.edu.ec/handle/21000/4302.
8. Definición: aplicación. [Consultado el: 9 de Enero de 2013]. Disponible en: http://definicion.de/aplicacion/.
9. Ecured: Aplicación Web. [Consultado el: 9 de Enero de 2013]. Disponible en: http://www.ecured.cu/index.php/Aplicaci%C3%B3n_web.

BIBLIOGRAFÍA

10. Incorporated, Adobe Systems. Building Adobe AIR Applications. publicado el: 5 de Septiembre de 2012 de última actualización: 5 de Septiembre de 2012. 256 p. Traducido de: Construyendo aplicaciones Adobe AIR. Disponible en: http://www.adobe.com/devnet/air/air_buildingapps.pdf.
11. Porque porqué: Aplicaciones web vs. Aplicaciones de escritorio. 2011. Última actualización: 26 de Mayo de 2012. [Consultado el: 3 de Septiembre de 2012]. Disponible en: http://pqpq.es/porqueses/informatica/aplicaciones-web-vs-aplicaciones-de-escritorio/.
12. Bot, Jackie. Wikipedia: Rich Internet Applications 2011. Última actualización: 7 de enero de 2013. [Consultado el: 4 de Septiembre de 2011]. Disponible en: http://es.wikipedia.org/wiki/Rich_Internet_Applications.
13. NExTenia: Aplicaciones ¿de escritorio o web? 2012 [Consultado el: 8 de Octubre de 2012]. Disponible en: http://www.nextecnia.com/2012/03/aplicaciones-de-escritorio-o-web.html.
14. Ecured: Patrones Gof. [Consultado el: 1 de Diciembre de 2012]. Disponible en: http://www.ecured.cu/index.php/Patrones_Gof.
15. Kent Beck, Y Otros. Planning Extreme Programming. Addison-Wesley, 2000. 160 p. ISBN 0-201-7109-9
16. Aguilar, Catherine. Aplicación de Conceptos de Gestión de Proyectos y Gestión de Riesgos en el Desarrollo de Productos Nuevos en el Campo de la Tecnología de Información. Universidad de Puerto Rico, 2005.
17. Ron Jeffries, Chet Hendrickson, Ann Anderson. Extreme Programming Installed. Addison-Wesley Professional, 2000. ISBN 0-201-70842-6.
18. Michele Marches, Y Otros. Extreme Programming Perspectivas. Addison Wesley, 2002. ISBN 0-201-77005-9.
19. Ivar Jacobson, Grady Booch, James Rumbaugh El lenguaje unificado de modelado - Manual de Referencia. Addison Weslley, 2000.
20. Visual Paradigm: 10 Reasons to Choose Visual Paradigm. 2006. [Consultado el: 10 de Enero de 2012]. Disponible en: http://www.visual-paradigm.com/aboutus/10reasons.jsp.
21. Ruiz, Víctor R. Linotipo: Ubuntu WebApps: Fusión de web y escritorio 2012 Última actualización: 20 de julio de 2012. Disponible en: http://rvr.linotipo.es/2012/07/ubuntu-webapps-fusi%C3%B3n-de-web-y-escritorio.html.
22. Cruz, Maximiliano Santa. VB-Mundo: Consumir servicios web desde una aplicación Win32 2008. Última actualización: 10 de enero de 2008. [Consultado el: 3 de mayo de 2012]. Disponible en: http://www.mediacenter.vb-mundo.com/index.php?option=com_content&task=view&id=226&Itemid=36.

BIBLIOGRAFÍA

23. Network, Oracle Technology. JavaFX Documentation 2011. Disponible en: http://docs.oracle.com/javafx/.
24. Chimote, Jatin. SANIsoft: Introduction to creating desktop applications with PHP and Titanium 2011. Última actualización: 3 de Enero de 2011. [Consultado el: 5 de Noviembre de 2012]. Disponible en: http://www.sanisoft.com/blog/2011/01/03/introduction-to-creating-desktop-applications-with-php-and-titanium/.
25. Morteza Milani, Brandon Benvie. AppJs: Build Desktop Applications 2012. [Consultado el: 1 de Noviembre de 2012]. Disponible en: http://appjs.org/.
26. Nieto, Tito Fernando Ale. Introducción al HTML5 2010. [Consultado el: 11 de Enero de 2013]. Disponible en: http://www.monografias.com/trabajos89/introduccion-al-html-5/introduccion-al-html-5.shtml.
27. Pérez, Javier Eguíluz. LibrosWeb: Introducción a CSS 2007. de 2011]. Disponible en: http://www.librosweb.es/css/capitulo_1/breve_historia_de_css.html.
28. Pérez, Javier Eguíluz. LibrosWeb: Introducción a JavaScript 2007. de 2011]. Disponible en: http://librosweb.es/javascript/capitulo_1/breve_historia.html.
29. JSON. [Consultado el: 11 de Enero de 2013]. Disponible en: http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=1090.
30. Chromeless Browser. 2011. [Consultado el: 6 de Enero de 2012]. Disponible en: https://github.com/mozilla/chromeless/.
31. Maldonado, Daniel Martin. SQLite, el motor de base de datos ágil y robusto 2008. [Consultado el: 11 de Enero de 2013]. Disponible en: http://www.aplicacionesempresariales.com/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html.
32. Qooxdoo. 2011. de 2012]. Disponible en: http://qooxdoo.org/docs.
33. CertSuperior.com: Firmas Digitales: ¿Qué son las Firmas Digitales? 2012. Disponible en: http://www.certsuperior.com/FirmasDigitales.aspx.
34. About JSingPdf 2013. Última actualización: 2013]. Disponible en: http://jsignpdf.sourceforge.net/.
35. Spket IDE. Última actualización: 31 de Diciembre de 2011. [Consultado el: 5 de Noviembre de 2011]. Disponible en: http://www.spket.com/.
36. Sqlite Studio. 2012. Última actualización: 31 de Diciembre de 2012. [Consultado el: 6 de Octubre de 2012]. Disponible en: http://sqlitestudio.one.pl/.
37. Patricio Letelier, María Carmen Penadés. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). 17 p. Disponible en: http://www.cyta.com.ar/ta0502/v5n2a1.htm.

BIBLIOGRAFÍA

38. Kent Beck, Martin Fowler. Planificando la programación extrema. Traducido de: Planning Extreme Programming.
39. Fowler, Martin. Ha muerto el diseño?. 2000, 12 p. Traducido de: Is Design Dead? Disponible en: http://www.martinfowler.com/articles/designDead.html.
40. Velásquez, Key. Patrones Arquitectónicos [Consultado el: 16 de Enero de 2013]. Disponible en: http://www.buenastareas.com/ensayos/Patrones-Arquitectonicos/1069013.html.
41. Erick Gamma, Richard Helm, Ralph Jonhson, Jonh Vlissides. Patrones de Diseño: elementos de software orientados a objetos reutilizables. Traducido por: Acebal, C. F. Addison Wesley, 313 p. Traducido de: Design Patterns: Elements of Reusable Object-Oriented Software.
42. Grosso, Andrés. Prácticas de Software: Patrones Grasp 2011. Última actualización: 21 de Marzo de 2012]. Disponible en: http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.
43. Jurado, Carlos Blé. Diseño ágil con TDD. 2010. 298 p. ISBN 978-1445264714.

ANEXOS

Anexo 1: Plantilla para especificación de Historias de Usuario

Historia de Usuario		
ID:	**Nombre:**	
Descripción:		
Estimación aproximada:	**Prioridad:**	**H.U original:**
Observaciones:		

ID: número de identificación de la Historia de Usuario.

Nombre: Nombre de la Historia de Usuario.

Descripción: Breve descripción del proceso que define la historia.

Estimación aproximada: Duración aproximada (en días), de duración de la confesión de las tareas relacionadas con la Historia de Usuario.

Prioridad: Prioridad estimada de manera general de la Historia de Usuario.

H.U original: Si la presente Historia de Usuario es producto de la división de otra, aquí se pone el nombre de la Historia de Usuario original.

Observaciones: Alguna acotación importante a señalar acerca de la Historia de Usuario.

Anexo 2: Plantilla para la especificación de Iteraciones

Iteración		
Numero:	**H.U (por orden):**	**Duración total:**
Descripción:		

Número: Número de la Iteración.

H.U: Historias de Usuario ordenadas, a implementar en la iteración.

Duración total: Duración total en días de la iteración.

Descripción: Breve descripción del proceso que define la iteración.

ANEXOS

Anexo 3: Plantilla para la especificación de las Tarjetas CRC.

Clase:	
Responsabilidades:	**Colaboraciones:**

Clase: Nombre de la clase.

Responsabilidades: Funcionalidades que realiza la clase.

Colaboraciones: Clases que tienen relación con la clase de la tarjeta.

Anexo 4: Plantilla para la especificación de las Tareas de Historias de Usuario.

Tareas de Historia de Usuario		
ID:	**H.U:**	**Iteración:**
Nombre:		
Tipo:	**Puntos estimados:**	
Fecha inicio:	**Fecha Fin:**	
Responsable:		
Descripción:		

ID: Número de identificación de la tarea

H.U: Historia de Usuario a la que pertenece la tarea.

Iteración: Iteración a la que pertenece la tarea.

Nombre: Nombre de la tarea.

Tipo: Tipo de tarea.

Puntos estimados: Duración aproximada (en días), de la tarea.

Fecha de inicio: Fecha en la que comienza la tarea.

Fecha de fin: Fecha en la que termina la tarea.

Responsable: Nombre del miembro del equipo que realizará la tarea.

Descripción: Breve descripción del proceso que define la tarea.

ANEXOS

Anexo 5: Plantilla para la especificación de las Pruebas de aceptación.

<table>
<tr><th colspan="4">Prueba de aceptación</th></tr>
<tr><td colspan="2">Código:</td><td colspan="2">H.U:</td></tr>
<tr><td colspan="4">Nombre:</td></tr>
<tr><td colspan="4">Descripción:</td></tr>
<tr><td colspan="4">Condiciones de Ejecución:</td></tr>
<tr><td colspan="4">Entrada / Pasos de ejecución:</td></tr>
<tr><td rowspan="2">Resultados esperados:</td><td>Escenario positivo:</td><td colspan="2">Escenario negativo:</td></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td colspan="4">Evaluación:</td></tr>
</table>

Código: Código de identificación de la prueba de aceptación

H.U: Historia de Usuario a la que pertenece la prueba de aceptación.

Nombre: Nombre de la prueba de aceptación.

Descripción: Breve descripción del proceso que define la prueba de aceptación.

Condiciones de ejecución: Condiciones en las que se debe encontrar la aplicación para el proceso de la prueba de aceptación.

Entrada / Pasos de ejecución: Pasos que realiza la prueba para su ejecución.

Resultados esperados: Respuesta de la aplicación en dependencia de la respuesta de la prueba de aceptación.

Evaluación: Resultado de la ejecución de la tarea.

GLOSARIO DE TÉRMINOS

Aplicación informática

Es un tipo de software que permite al usuario realizar uno o más tipos de trabajos utilizando una computadora [8].

Aplicación web

Software que el usuario puede utilizar accediendo a un servidor Web mediante un navegador [9].

Aplicación de escritorio

Es un programa instalado en la computadora, que inicia sesión una vez arrancado el sistema operativo, asumiendo que se va a abrir la aplicación.

Dirección de Laboratorios

Antigua dirección de la UCI, encargada de manejar el área de los laboratorios.

Java

Plataforma para el desarrollo de aplicaciones de escritorio. Se desarrolla sobre un lenguaje del mismo nombre.

Integración continua

Una práctica del desarrollo de software donde los miembros del equipo integran su trabajo con frecuencia: normalmente, cada persona integra de forma diaria, conduciendo a múltiples integraciones por día. Cada integración es comprobada por una construcción automática (incluyendo las pruebas) para detectar errores de integración tan rápido como sea posible.

Mozilla Foundation

Fundación de *Mozilla*. Empresa encargada de desarrollar el navegador *Firefox*.

Github

Repositorio de código, principalmente de proyectos de software libre.