Universidad de las Ciencias Informáticas

Facultad 6

Título: Aplicación Web PanaConsul para el Análisis de Informes de Recaudos Consulares.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Alejandro Pérez de la Cruz

Tutor(es): Ing. Frank Alberto Rodriguez Solana

La Habana, mayo de 2013

“Año 55 de la Revolución”

Declaración de Autoría

Declaramos ser autores de la presente tesis que tiene por título: y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Alejandro Pérez de la Cruz

_____

Firma del Autor

Ing. Frank Alberto Rodriguez Solana

_____

Firma del Tutor

Datos de Contacto

Autor

Alejandro Pérez de la Cruz
Consulado de Panamá en Cuba, La Habana, Cuba
e-mail: neowxp@gmail.com

Agradecimientos

Agradezco a mi futura esposa por la paciencia, mi familia y amigos y mi tutor que sin él no hubiera podido encontrar el rumbo.

Dedicatoria

A mi padre por haberme iniciado en este camino, a mi madre por haberme dado el último espaldarazo.

Resumen

El Departamento Control Financiero Consular, ha mantenido un atraso en promedio de hasta tres meses, en cuanto a la presentación de los Informes de Recaudos Consulares debidamente analizados. Esta situación lleva a la Autoridad Marítima de Panamá a tener que usar estimados a la hora de tomar decisiones, debido a la carencia de información actualizada en sus Balances Económicos. Al no manejarse cifras reales al rendir cuenta de la gestión, la Contraloría General de la República en reiteradas ocasiones ha señalado esto como una deficiencia a la cual se le ha de dar solución. En el presente trabajo tiene como objetivo dar solución a tal deficiencia. Para ello se desarrolló una aplicación Web que agiliza el proceso de análisis de los Informes de Recaudos Consulares, lo cual permite que el Departamento de Control Financiero Consular cuente con información actualizada para evitar el uso de estimados y poder así realizar una toma de decisiones más fiable.

Palabras claves: Balances Económicos, Control Financiero Consular, Recaudos Consulares.

Índice de Contenido

Índice de Figuras

Índice de Tablas

Introducción

Introducción

En todo el mundo, la representación de un país ante personas naturales y jurídicas en otro país, le corresponde a los consulados. El consulado es la entidad del servicio exterior de un país, de carácter político, de información y propaganda administrada, aduanas, notarial y registro político; encargada de: proteger en el estado receptor los intereses del estado emisor y de sus nacionales; fomentar el desarrollo de las relaciones comerciales, económicas, culturales y científicas; extender pasaportes y documentos de viaje a los nacionales del estado que envía y visados o documentos de viaje a los nacionales del país receptor que deseen viajar al país emisor; comunicar decisiones judiciales y extrajudiciales y diligenciar comisiones rogatorias de conformidad con los acuerdos internacionales en vigor y, a falta de los mismos, de manera que sea compatible con las leyes y reglamentos del Estado receptor; y ejercerlos derechos de control o inspección de los buques que tengan la bandera de dicho estado, de las aeronaves matriculadas en el mismo y, también, de sus tripulaciones (1).

El consulado de Panamá utiliza su propio sistema económico en el cual se establecen los llamados recaudos consulares, los cuales se conforman a partir del dinero recaudado por la prestación de servicios consulares en los países receptores. Estos recaudos consulares se generan a partir de los recibos oficiales emitidos por los distintos consulados de Panamá y de estos recaudos se deducen los gastos consulares para que al cierre del Balance Económico realizado a final de mes, se pueda realizar la Remesa Consular la cual consiste en la diferencia entre los gastos y los ingresos del consulado en el período contable.

El Departamento Control Financiero Consular, dentro de la Autoridad Marítima de Panamá (AMP), es el encargado de recibir los Recaudos Consulares y analizarlos partiendo del Informe de Recaudos Consulares emitidos por los consulados en los países receptores. Estos informes se envían impresos en papel hacia Panamá un mes después de realizado el Balance Económico en los consulados y tardan otro mes más en ser procesados, teniendo que ser capturada la información nuevamente para introducirlos en hojas de cálculo Excel. Debido a este proceso tan lento, el Departamento de Control Financiero no es capaz de presentar información actualizada en sus Balances Económicos lo cual conlleva a que el departamento tenga que usar cifras estimadas y que no se manejen cifras reales en el momento de tomar decisiones.

La Contraloría General de la República de Panamá es la encargada de velar por los fondos públicos y ha delegado en la Autoridad Marítima de Panamá (AMP) la revisión de los Recaudos Consulares. La AMP rinde cuentas a la Contraloría General y esta ha hecho énfasis en la actualización de las cifras manejadas en todo el proceso de Control Financiero lo cual representa una deficiencia de la AMP. Las

cifras utilizadas en todo este proceso de control son arrojadas por el Proceso de Análisis de Informes Consulares.

De lo expuesto con anterioridad se identifica el **problema a resolver**: ¿Cómo agilizar el Proceso de Análisis de Informes Consulares?

Determinando como **objeto de estudio** el proceso de control a las finanzas consulares de Panamá, que involucra como **campo de acción** el proceso de revisión de Informes Consulares.

El **objetivo general** del presente trabajo para dar solución al problema planteado es: Desarrollar una aplicación Web para gestionar el proceso de análisis de los Informes de Recaudos Consulares.

Para dar cumplimiento al objetivo planteado se identificaron las siguientes **tareas investigación**:

- Estudiar las principales tecnologías, metodologías y herramientas de desarrollo Web a utilizar en la implementación del sistema.
- Definir los requisitos a tener en cuenta para implementar la aplicación.
- Realizar análisis y diseño de una aplicación que de solución al problema planteado.
- Implementación de los requisitos funcionales definidos.
- Elaborar el diseño de los casos de prueba.
- Ejecutar los casos de prueba para validar el correcto análisis de los datos por parte del sistema.

Como posible resultado de la investigación se propone la siguiente **idea a defender**: Producto del uso de la aplicación desarrollada para gestionar el proceso de análisis de los Informes de Recaudos Consulares, se puede acelerar el proceso de análisis en la sede central de la Autoridad Marítima de Panamá.

Métodos de investigación

Para darle cumplimiento a las tareas propuestas se utilizaron los siguientes métodos de investigación:

Teóricos:

- **Histórico-Lógico:** este método se utiliza con el fin de recopilar la información que se posee hasta el momento sobre las herramientas que utilizan en el proceso de Control Financiero Consular y resumir los aspectos fundamentales de su evolución para el desarrollo de la investigación en curso.
- **Analítico-Sintético:** este método se utiliza para analizar y estudiar las teorías recopiladas en la bibliografía y extraer los elementos más importantes para el desarrollo del procedimiento de

Control Financiero Consular.

- **Modelación:** este método se usó para descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio. La modelación es el proceso mediante el cual se crearon los modelos de clases que componen el sistema de gestión procedimiento de análisis y Control Financiero Consular, por lo que es el más importante en el proceso de construcción del software.

Estructura capitular

Capítulo 1: "Fundamentos Teóricos".
En este capítulo se abordan los principales conceptos que se manejan en el proceso de Análisis y Control Financiero Consular. Se hace un estudio de las principales características de la metodología y herramientas a utilizar para cumplir con el objetivo de este trabajo.

Capítulo 2:"Características del Sistema".
En este capítulo se describen las principales características del sistema, se muestran los requisitos del mismo, se definen las historias de usuarios y se realizará el diagrama de clases. En este capítulo también se planifican las iteraciones en las que se implementará cada una de las historias de usuario y se estimará el tiempo de duración de cada tarea de la ingeniería.

Capítulo 3: "Validación y prueba".
En este capítulo se presenta la estrategia de prueba a seguir por este trabajo, se abordan las técnicas a utilizar, se exponen los casos de prueba por cada historia de usuario, y por último se brinda un resumen de los resultados arrojados por los casos de pruebas basados en historias de usuarios.

Capitulo1. Fundamentos Teóricos

En los últimos tiempos el desarrollo de las Tecnologías de la Información y las Comunicaciones (TICs), ha provocado una tendencia, por parte de las empresas, a informatizar muchos de sus procesos internos, sobre todo los procesos contables, con el propósito de evitar errores humanos en los cálculos, agilizar el procesamiento de datos, obtener resultados más fiables y disminuir el tiempo de procesado de la información contable. En el presente capítulo se hace un estudio de las principales herramientas y tecnologías en el campo del desarrollo de aplicaciones Web. Además se analizan las tecnologías a emplear para el desarrollo de la aplicación Web teniendo en cuenta que la misma debe contar con una arquitectura cliente servidor para que la información a procesar se concentre en un mismo lugar.

1.1 Sistema de Análisis y Control Financiero Consular

El **Manual De Procedimientos Para Las Operaciones De Recaudos Consulares (Segunda Versión),** es el documento rector por el cual se rigen los Consulados de Panamá para realizar el proceso de Análisis y Control Financiero Consular, y el mismo se encuentra vigente desde enero de 2007. Este documento describe detalladamente cómo se establece el procedimiento jurídico para llevar a cabo las Operaciones de Recaudos Consulares.

Este proceso de análisis se establece partiendo de la revisión de los ingresos y gastos generados en todos los consulados de Panamá. Los ingresos son registrados a través del Recibo Oficial General y del Recibo de Impuestos y Tasas; y los gastos se controlan mediante la emisión de cheques o transferencias. Esta revisión es realizada por un auditor el cual es el encargado de fiscalizar las cuentas de uno o más consulados.

El auditor parte del saldo inicial del mes anterior, le adiciona el monto de los recibos oficiales verificando los cobros y resta los gastos emitidos comprobando que se correspondan con el presupuesto autorizado por la AMP. Este balance se concilia contra el dinero real presente en la cuenta de banco de cada consulado, y el balance del último día del mes es el monto que debe conformar la Remesa Consular la cual se compara con el dinero real que se recibió del consulado. El resultado final ideal sería un saldo igual a cero, este puede ser negativo si por un mal manejo le faltó dinero a las Remesas Consulares o positivo si se envió dinero de más.

Los consulados están obligados a facturar las actuaciones consulares; que son las operaciones: notariales, de registro público, servicios a buques y naves, migratorias, etc.; mediante los Recibos Oficiales Generales y los Recibos de Impuesto y Tasas. Para el funcionamiento del consulado, el

mismo tiene autorizado un presupuesto mensual que se ejecuta mediante cheques o transferencias que se controlan por la resolución de presupuesto vigente y por el libro de banco que sirve al auditor para fiscalizar la cuenta del consulado.

En este trabajo se analizaron las herramientas Peachtree y Asset las cuales presentan varios puntos en común con el proceso de gestión de los informes consulares dado que permiten manejar la conciliación de las cuentas bancarias, hacer pagos y emitir facturas. Pero ninguna de las dos permite crear los roles que intervienen en el proceso de análisis financiero consular, los cuales tienen una participación específica en el mismo ya que ningún sistema implementa la interacción que existe entre los roles de un auditor y los administradores de consulados. El proceso de liquidación, tampoco está contemplado por estos softwares. Por ende el proceso de Análisis y Control Financiero Consular se hace único para los consulados de Panamá, por lo que existe la necesidad de crear un sistema completamente nuevo, que incorpore estos instrumentos económicos para poder informatizar dicho proceso y que controle el flujo de procesos que integran todo el proceso de análisis.

1.2 Instrumentos utilizados en el proceso de Análisis y Control Financiero

Los instrumentos que intervienen en el proceso de Análisis y Control financiero son:

- Recibos Oficiales Generales y Recibos de Impuestos y Tasas: Son las facturas por la cual se ingresa fondos a la contabilidad y sirve como constancia de pago al cliente.
- Cheques y transferencias bancarias: Son las vías de pago que utilizan los consulados para costear sus gastos.
- Libro de Banco: Es el documento donde se registra el estado inicial a principio del período contable, y los ingresos y egresos diarios de la contabilidad consular.
- Libro Diario: Este documento registra en detalle todas las operaciones hechas en el día.
- Conciliación bancaria: Es el documento donde se modifica el balance real existente en la cuenta bancaria para hacerla coincidir con el balance del Libro de Banco.
- Cheques en tránsito: Son los cheques emitidos por el consulado, los cuales no han sido cobrados en el banco.

A través del uso de la herramienta informática Excel, se creó un libro digital el cual se utiliza actualmente para agilizar el manejo de los datos contenidos en las herramientas descritas anteriormente y facilitar la información de los instrumentos, imprimiendo algunas de las páginas del libro digital para conformar el Informe de Recaudos Consulares.

Este informe es impreso en el consulado y se envía en una valija diplomática hacia Panamá. Luego en

la AMP se digitaliza nuevamente este informe para realizar los análisis pertinentes. Todo este proceso demora de dos a tres meses hasta la liquidación que es el cierre del análisis, lo que conlleva a trabajar con cifras atrasadas durante este periodo.

1.3 Lenguajes de programación Web

Un lenguaje de programación es como un idioma que puede ser interpretado por las computadoras para llevar a cabo las órdenes del programador. Existen diferentes tipos de lenguajes, que van desde los lenguajes de bajo nivel hasta los de alto nivel, algunos son compilados, otros son interpretados, unos sirven para desarrollar aplicaciones desktop y otros son más propicios para el desarrollo Web.
Entre los lenguajes de programación Web, se encuentran ASP, Ruby, JSP, Python y PHP.

1.3.1 Python

El lenguaje Python fue diseñado por Guido van Rossum a principio de los años 90, posee una sintaxis simple y presenta una amplia variedad de características de lenguajes de programación como C++, Java, Modula-3 y Scheme. Debido a esto, una de las características notables de Python es su atractivo para los desarrolladores profesionales de programación, científicos investigadores, artistas y educadores. (2)

Python es un lenguaje de programación orientado a objetos, e interpretado lo cual lo hace muy útil para una amplia gama de tareas, desde un pequeño script hasta una aplicación completa. Es libremente accesible como binario o su código fuente y puede ser usado libre de derechos de autor en la mayoría de plataformas incluyendo Windows, Macintosh, Linux, FreeBSD y Solaris. (3)

Es bien conocido por su utilidad como una herramienta de desarrollo rápido de aplicaciones, pues a menudo se escucha de proyectos en Python que terminan en horas o días en vez de semanas o meses que se hubiera requerido con un lenguaje de programación tradicional. Este presume de un abundante set de bibliotecas estándar así como de la habilidad para interactuar con bibliotecas de otros lenguajes como C++. (3)

1.3.2 Ruby

Como lenguaje de programación, Ruby está diseñado para permitirle a los desarrolladores expresarse libremente. Este trata de operar al nivel del programador, llevando el enfoque lejos de la maquina y hacia el problema a mano. No obstante, Ruby es altamente flexible y no es más que una masilla puesta en las manos del desarrollador. Con una mente rígida se tiende a complicar las cosas, pudiendo producir código de Ruby complejo. Con un ligero y desahogado punto de vista, podrán

producirse programas con un código simple y bonito. (4)

Un lenguaje dinámico, expresivo y abierto no encaja bien en estrictos patrones de uso adecuado o inadecuado. Sin embargo, esto no significa que los desarrolladores experimentados en Ruby no utilicen estrategias generales para atacar problemas. De hecho, existe un alto grado de similitud en la manera en que los desarrolladores profesionales de Ruby enfocan un extenso rango de desafíos. (4)

Ruby tiene un conjunto de otras funcionalidades entre las que se encuentran las siguientes: (5)

- Manejo de excepciones, como Java y Python, para facilitar el manejo de errores.
- Un verdadero mark-and-sweep garbage collector para todos los objetos de Ruby. No es necesario mantener contadores de referencias en bibliotecas externas. Como dice Matz, “Esto es mejor para tu salud”.
- Escribir extensiones en C para Ruby es más fácil que hacer lo mismo para Perl o Python, con una API muy elegante para utilizar Ruby desde C. Esto incluye llamadas para embeber Ruby en otros programas, y así usarlo como lenguaje de scripting. También está disponible una interfaz SWIG.
- Puede cargar bibliotecas de extensión dinámicamente si lo permite el sistema operativo.
- Tiene manejo de hilos (threading) independiente del sistema operativo. De esta forma, tienes soporte multi-hilo en todas las plataformas en las que corre Ruby, sin importar si el sistema operativo lo soporta o no, ¡incluso en MS-DOS!
- Ruby es fácilmente portable: se desarrolla mayoritariamente en GNU/Linux, pero corre en varios tipos de UNIX, Mac OS X, Windows 95/98/Me/NT/2000/XP, DOS, BeOS, OS/2, etc.

1.3.3 PHP

PHP fue concebido en otoño de 1994 por Rasmus Lerdorf. Las primeras versiones no distribuidas al público fueron usadas en un sus páginas Web para mantener un control sobre quien consultaba su currículum. La primera versión disponible para el público a principios de 1995 fue conocida como "Herramientas para páginas Web personales"(Personal Home Page Tools). Consistían en un analizador sintáctico muy simple que solo entendía unas cuantas macros y una serie de utilidades comunes en las páginas Web de entonces, un libro de visitas, un contador y otras pequeñas cosas. El analizador sintáctico fue reescrito a mediados de 1995 y fue nombrado PHP/FI versión 2. FI viene de otro programa que Rasmus había escrito y que procesaba los datos de formularios. Así que combinó las "Herramientas para páginas Web personales", el "intérprete de formularios", añadió soporte para

mySQL y PHP/FI vio la luz. PHP/FI creció a gran velocidad y la gente empezó a contribuir en el código. (6)

Combinado con la base de datos MySQL, es el lenguaje estándar a la hora de crear sitios de comercio electrónico o páginas Web dinámicas. Entre sus características fundamentales están: (7)

- Gratuito. Al tratarse de software libre, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre.
- Gran popularidad. Existe una gran comunidad de desarrolladores y programadores que continuamente implementan mejoras en su código, y que en muchos casos estarán encantados de echarnos una mano cuando nos enfrentemos a algún problema. Baste decir que en el momento de escribir este libro son casi diez los millones de páginas Web desarrolladas con PHP.
- Enorme eficiencia. Con escaso mantenimiento y un servidor gratuito (en nuestro caso, Apache), puede soportar sin problema millones de visitas diarias.
- Sencilla integración con múltiples bases de datos. Esencial para una página Web verdaderamente dinámica, es una correcta integración con base de datos. Aunque MySQL es la base de datos que mejor trabaja con PHP (y la que, por tanto, estudiaremos en nuestra guía), puede conectarse también a PostgreSQL, Oracle, dbm, filePro, interbasem o cualquier otra base de datos compatible con ODBC (open Database Connectivity).
- Versatilidad. PHP puede usarse con la mayoría de sistemas operativos, ya sea basados en UNIX (Linux, Solares, FreeBSD), como con Windows, el sistema operativo de Microsoft.
- Gran número de funciones predefinidas. A diferencia de otros lenguajes de programación, PHP fue diseñado especialmente para el desarrollo de páginas Web dinámicas. Por ello, está dotado de un gran número de funciones que nos simplificarán enormemente tareas habituales como descargar documentos, enviar correos, trabajar con cookies y sesiones, etc.

Como se puede observar, cada lenguaje brinda tiene sus propias ventajas para los desarrolladores, pero como contratiempo el equipo de desarrollo no tiene experiencia en todos ellos, dedicar tiempo a la capacitación implicaría un atraso en el desarrollo de la solución. Por este motivo se estima conveniente hacer uso del lenguaje PHP ya que a pesar de tener sus propias ventajas, el equipo de desarrollo está mucho más familiarizado con él y tiene experiencia de desarrollos anteriores.

1.3.4 JavaScript

JavaScript es un lenguaje de programación muy útil en la creación de páginas Web dinámicas. Este es

un lenguaje interpretado por los navegadores Web por lo que se dice que es un lenguaje de programación Web del lado del cliente. Este lenguaje brinda muchas facilidades para mejorar la apariencia de la interfaz visual de las páginas Web, aunque este fue creado con el propósito de validar los datos de las aplicaciones Web antes de ser enviados al servidor y de esta forma aprovechar los recursos del servidor en otras tareas.

Entre las ventajas de este lenguaje se encuentra: (8)

- Es un lenguaje fácil de aprender debido a que es sencillo, lo que permite que se pueda comenzar a trabajar con él desde un principio, y que una vez que se conocen las bases del lenguaje, no hay que esforzarse mucho para crear grandes aplicaciones.
- Es rápido, dado que no necesita ser compilado y el navegador no tendrá que cargar una máquina virtual para ejecutar el código lo cual lo hace ideal para agregar ciertas funciones a una página Web.
- Es muy potente, ya que aunque no puede hacer nada directamente a nivel de máquina, es capaz de trabajar con muchas propiedades de los navegadores Web, páginas Web y a veces con el propio sistema donde se ejecuta el navegador.
- Utiliza una arquitectura orientada a objetos parecida a la de los otros lenguajes como Java o C++, y trabaja con las funciones del constructor o la estratificación de jerarquías lo que lo provee de más opciones para utilizar en el código.
- Es el lenguaje que más se utiliza en la Web, ya que millones de sitios lo incluyen dentro de sus páginas para aprovechar su ejecución del lado del cliente.
- Reduce la carga del servidor, debido a su ejecución del lado del cliente, los programadores Web lo incluyen en sus sitios para realizar gran parte de las funciones del cliente de las cuales antes se encargaba el servidor.

Con JavaScript es posible validar los elementos antes de que el usuario se los envíe al servidor. De esta forma se reduce la cantidad de transacciones que se efectúan a través del protocolo HTTP y las posibilidades de que se genere un error durante la inserción de datos. JavaScript también puede leer y escribir cookies, una operación que antes únicamente podía desarrollar el servidor Web. (8)

Algo interesante sobre JavaScript es que siempre se ejecuta dentro de un entorno, y el más popular es el entorno del navegador pero no es el único. JavaScript puede ejecutarse en el servidor, en un entorno de escritorio, y en un entorno enriquecido. Hoy en día se puede usar el JavaScript para hacer cosas como (9):

- Crear enriquecidas y potentes aplicaciones Web (el tipo de aplicación que se ejecuta dentro de un navegador Web, como Gmail).
- Escribir código del lado del servidor como scripts ASP[1] o por ejemplo código que será ejecutado usando Rhino (un motor JavaScript programado en Java).
- Creando enriquecidas aplicaciones multimedias (Flash, Flex) usando ActionScript.
- Escribir scripts que automaticen tareas administrativas en el escritorio de Windows, usando Windows Scripting Host.
- Escribir extensiones/plugins para un excesivo número de aplicaciones de escritorio como Firefox, Dreamweaver y Fiddler.
- Crear aplicaciones Web que almacenen información en base de datos locales en el escritorio del usuario, usando Google Gears.
- Crear Yahoo! widgets, Paneles de widgets de Mac, o aplicaciones Adobe Air que se ejecutan en tu escritorio.

Esto no es en absoluto una lista extensa, JavaScript comenzó dentro de las páginas Web, pero hoy en día se puede decir con confianza que está prácticamente todas partes. (9)

1.4 Herramientas a utilizar.

En este trabajo se hará uso de herramientas necesarias para la confección de la aplicación. Entre estas herramientas se encuentran las bibliotecas que nos facilitarán el desarrollo, el entorno de desarrollo integrado que utilizaremos para programar la solución y el sistema gestor de bases de datos que facilitará la gestión y el almacenamiento de la información generada por el proceso.

1.4.1 Biblioteca de JavaScript, jQuery

La biblioteca jQery, de JavaScript, es de código abierto y su trabajo es simplificar la interacción entre un documento HTML, o más precisamente el Modelo de Objeto del Documento o DOM (Document Object Model por sus siglas en inglés), y JavaScript. En otras palabras, jQery facilita la programación de una página Web dinámica. Específicamente facilita, la manipulación del documento HTML, el manejo de eventos en el navegador, animaciones en el DOM, la interacción con Ajax, y el desarrollo con JavaScript a través del navegador Web. (10)

jQuery brinda a los desarrolladores y diseñadores Web una vía fácil de crear sofisticados efectos con poco código. Debido a que es tan fácil para implementar, su popularidad ha ido creciendo. Se pueden

[1] Active Server Pages: Es una tecnología web de Microsoft del lado del servidor.

encontrar ejemplos de jQuery por todo internet, como Facebook y Twitter que usan un número de efectos de esta biblioteca. (11)

Esta biblioteca ofrece al desarrollador una serie de funcionalidades implementadas, que facilitará el desarrollo de aplicaciones complejas del lado del cliente. Por ejemplo, jQuery ofrece ayuda para la creación de interfaces de usuarios, efectos dinámicos, construcciones de gráficas, aplicaciones que hacen uso de Ajax, etc. Esta biblioteca posee la Licencia Pública General o GPL (General Public License por sus siglas en inglés).

1.4.2 Entorno de Desarrollo Integrado NetBeans.

NetBeans es un Entorno de Desarrollo Integrado IDE (Integrated Development Environment por sus siglas en inglés), es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Es un IDE potente, robusto y posee una interfaz sencilla lo que la hace amigable y atractiva para el desarrollador. Está desarrollado en lenguaje Java, pero existe un gran número importante de módulos que pueden incluírsele para hacerlo extensible y usarlo con otros lenguajes como PHP 5. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso, desarrollado por un proyecto de código abierto de gran éxito con una gran base de usuarios, y cuenta con una comunidad en constante crecimiento. (12)

Entre las principales ventajas del entorno de desarrollo integrado NetBeans se encuentra que (12):

- Posibilita una rápida e inteligente edición de código, ya que el mismo se encarga de indentar las líneas de código; hace corresponder palabras y llaves, paréntesis y corchetes; resalta los errores sintácticos y semánticos en las sintaxis de código.
- Provee al programador de plantillas de código, consejos de codificación, y herramientas de reconstrucción del código.
- Permite una fácil y eficiente administración de los proyectos, pues provee al desarrollador de varias vistas que organizan los archivos y carpetas que conforman el proyecto.
- Es multiplataforma, pues se puede instalar en varios sistemas operativos como debido a que está escrito en el lenguaje Java.
- Presenta soporte para múltiples lenguajes como Java que es el lenguaje que viene por defecto con el IDE, entre ellos C, C++ y PHP.

1.5 Sistemas gestores de bases de datos

"*Una base de datos es un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna empresa dada*" (13) Un sistema de bases de datos, es simplemente un sistema

para gestionar registros con la información de los datos persistentes. Es como un armario electrónico para depositar y almacenar información computarizada dentro de archivos. Los usuarios del sistema deben poder realizar operaciones como: (13)

- Agregar archivos sin registros.
- Insertar registros dentro de los archivos.
- Recuperar datos de los archivos existentes.
- Modificar los registros de los archivos existentes.
- Eliminar registros dentro de los archivos.
- Eliminar los archivos existentes de la base de datos.

Por su parte un DBMS (Database Management System por sus siglas en inglés) es el software o conjunto de software que manejan todo el acceso a la base de datos, controlando los permisos que tengan los usuarios para realizar determinadas operaciones sobre los mismos (13). Los RDBMS (Relational Database Management System por sus siglas en inglés) son los sistemas de bases de datos que implementan el modelo relacional, el cuál es el modelo más común de todos los existentes.

1.5.1 Sistema gestor de bases de datos PostgreSQL

PostgreSQl es el sistema gestor de bases de datos de código abierto más avanzado del mundo (14). Implementa el modelo de datos objeto-relacional y está basado en Postgres 4.2, el cual fue desarrollado por el departamento de Ciencias de la Computación de la Universidad de Berkeley en California. Este gestor fue pionero de muchos conceptos que se agregaron años después en algunos sistemas de bases de datos comerciales.

Este es descendiente del código original de Berkeley, soporta la mayor parte del estándar SQL y presenta características como las consultas complejas, los triggers, las llaves foráneas, integridad transaccional, control de concurrencia multiversión y las vistas actualizables. Por otro lado estas características pueden ser extendidas por el usuario añadiendo funciones, tipos de datos, funciones de agregado, métodos de indexado, operadores e incluso lenguajes de procedimiento. Y dado su licencia, PostgreSQl puede ser usado modificado y distribuido por cualquiera libre de costo para cualquier propósito privado, comercial o académico (15).

1.5.2 Sistema gestor de bases de datos MySQL

MySQL es un Sistema Gestor de Bases de Datos Relacionales, de código abierto, capaz de almacenar enormes cantidades de datos de gran variedad y distribuirlos para poder cubrir las necesidades de

cualquier tipo de organización, desde pequeños negocios hasta grandes empresas y organismos. Este sistema de base de datos compite con otros sistemas como Oracle, SQL Server y DB2 de la compañía IBM. (16)

Entre las principales ventajas que lleva a los desarrolladores a usar MySQL como gestor de bases de datos, se encuentran (7):

- **Alto rendimiento:** MySQL es muy rápido, esto se puede observar a través de las pruebas realizadas al gestor en la siguiente Web: www.mysql.com/information/benchmarks.html.
- **Bajo costo:** MySQL está disponible de forma gratuita, bajo una licencia de código abierto o con una licencia comercial por precios reducidos.
- **Facilidad de uso:** Utiliza el lenguaje SQL estándar, por lo que tener conocimientos previo de cualquier gestor relacional facilitará su uso, a demás de requerir poco esfuerzo para su uso.
- **Portabilidad:** Se puede utilizar en los sistemas operativos basados en Unix y en las versiones de Windows.

En el mundo informático existen otros gestores de bases de datos como Oracle, SQL Server, DB Desktop, pero en la relación costo-beneficio PostgreSQl y MySQl son los más aventajados, dado que presentan buenas características que cubren con la necesidad existente de la aplicación y por otro lado son libres de costo. Dentro de la Autoridad Marítima de Panamá existen otras aplicaciones que utilizan MySQL y para ahorrar costo y tiempo en el adiestramiento de los administradores de bases de datos en nuevas tecnologías, se determinó utilizar MySQL para implementar la aplicación.

1.6 Metodologías de desarrollo de software

La ingeniería de software es la disciplina que recoge todos los aspectos de la producción de software desde que se comienzan a especificarse los requisitos que tendrá el sistema a implementar, hasta el mantenimiento que se le brindará al mismo una vez concluido su desarrollo. En la ingeniería se adopta un enfoque sistemático y organizado para producir software con calidad, seleccionando la metodología más apropiada dadas las circunstancias y que permita elaborar un producto determinado de forma eficiente (17).

1.6.1 Metodología de desarrollo RUP

El Proceso Unificado de Rational o RUP (Rational Unified Process, por sus siglas en inglés) es formalmente una metodología de desarrollo de software. Esta metodología hace énfasis en comenzar con las actividades de obtención de requisitos, el análisis, el diseño y el modelado de datos de todo el proyecto antes de comenzar con la implementación del software. Con este enfoque RUP busca

mantener el proyecto alineado con las expectativas del usuario, y mitiga los riesgos alentando al equipo a desarrollar primero la parte riesgosa del proyecto permitiendo más tiempo para reconocer y responder al tema o problema en cuestión. Esto también reduce los cambios cuando el diseño requiere modificación (18).

RUP define dos espacios básicos. En uno se agrupan las actividades lógicamente acordes a la disciplina de la que esta es responsable de su ejecución. Las disciplinas centrales de esta metodología según (19) son:

- **Modelado del negocio:** Como su nombre lo indica, el propósito de esta disciplina es el modelado del negocio. La idea es mientras mejor se conozca el negocio el software desarrollado puede encajar más propiamente. Esta disciplina produce los modelos de caso de uso del negocio y el modelo de objetos del negocio.

- **Requisitos:** Esta disciplina se enfoca a desarrollar una sólida comprensión de los requerimientos. Es un intento por lograr un acuerdo con el cliente tanto como proveer una guía para los desarrolladores. Esta disciplina produce el modelo de caso de uso y puede producir un prototipo de interfaz de usuario.

- **Análisis y diseño:** Los requisitos capturados en la disciplina de requisitos son analizados y transformados dentro del diseño en la disciplina de análisis y diseño. Se desarrolla una arquitectura para guiar el esfuerzo de desarrollo restante. Los modelos de análisis y diseños son generados como parte de esta disciplina.

- **Implementación:** En esta disciplina el diseño se transforma en código de la implementación. Se desarrolla una estrategia para dividir el sistema en subsistemas. El resultado final es una serie de componentes implementados y probados que forman el producto.

- **Prueba:** Esta disciplina se encarga de verificar el sistema. Entre otras cosas, esto típicamente significa verificar todos los requerimientos que han sido definidos, confirmar que los componentes trabajan acoplados como se espera e identificar cualquier defecto restante en el producto. La principal salida de esta disciplina son los modelos de prueba y los defectos generados como resultado de las pruebas.

- **Despliegue:** Esta disciplina efectúa que el producto esté disponible para los usuarios

finales. Tales como cubrir los detalles de empaquetamiento del software, instalación, capacitación a los usuarios y la distribución del producto.

También RUP soporta las disciplinas de administración de cambios y configuración, administración de proyecto y entorno.

En el otro espacio, RUP maneja las estructuras de las iteraciones en un proyecto de software las cuales se agrupan en 4 fases. Estas fases son una vista en forma de cascada para los propósitos de planificación del proyecto, dado que el desarrollo de software puede ser un proceso iterativo. Cada fase puede definir una o más iteraciones, y termina con un hito el cual representa un punto de decisión al nivel de administración. Las fases definidas por RUP están organizadas acorde a su ejecución en el tiempo como aparecen a continuación (19):

- Inicio: Esta fase trata acerca de la extensión del proyecto en términos del producto, entendiendo los requisitos, costo y principales riesgos. Aquí se genera el documento visión del producto identificando un grupo inicial de caos de usos y actores, desarrollando el caso del negocio para el proyecto y creando un plan de proyecto que contiene las fases y las iteraciones planificadas.

- Elaboración: Esta es quizás la fase más significante, dado que en ella los requisitos son analizados en detalles y sobre todo porque la arquitectura es desarrollada para llevar el proyecto a su finalización. Lograr estabilidad en los requerimientos y sobre todo en la arquitectura son las expectativas para el final de esta fase. En esta fase destaca el desarrollo del modelo de caso de uso, el modelo del análisis, el modelo del diseño, el prototipo de la arquitectura y el plan de desarrollo.

- Construcción: Esta fase se enfoca en el diseño y la implementación del software. El logro de esta fase es la evolución del prototipo inicial en el producto final. El aporte al final de esta fase es el producto en sí.

- Transición: En esta fase el producto es alistado para los usuarios. Esta puede involucrar el arreglo de defectos identificados durante las pruebas betas, la adición de cualquier capacidad que se haya perdido, capacitando a los usuarios, entre otras cosas. El producto final es entregado al cliente al final de esta fase.

Esta metodología de desarrollo envuelve muchos más artefactos de los que se mencionan en este

epígrafe. Por otra parte RUP necesita de muchos modelos y diagramas que son de mucha ayuda para proyectos grande y abarcadores, pero para el desarrollo de aplicaciones relativamente pequeñas significan mucho más trabajo y esfuerzo por parte del equipo de desarrollo.

1.6.2 Metodología de desarrollo XP

La metodología de desarrollo Programación Extrema o XP *eXtreme Programming* por sus siglas en inglés, es un enfoque creado por Kent Beck. Esta es la más destacada de las metodologías ágiles en el desarrollo de software, y su diferencia con las metodologías tradicionales consiste principalmente en que enfatiza más la adaptabilidad, tomando en consideración los cambios en los requisitos como algo normal e inevitable en el desarrollo de los proyectos de software (20). Según Beck, XP es una metodología ligera, eficiente, de bajo riesgo, flexible, predecible, científica y divertida para el desarrollo de software (21).

Esta metodología se distingue del resto por tener una retroalimentación temprana, continua y concreta de sus ciclos cortos. Posee un enfoque incremental, presenta la habilidad de planificar de forma flexible la implementación de una funcionalidad, asegura las pruebas automáticas escritas por el programador y el cliente para monitorear el progreso de la implementación; brinda confianza en: la comunicación oral, pruebas y código fuente, en el revolucionario proceso de diseño, en la colaboración de programadores con habilidades ordinarias y por último en la práctica de su funcionamiento con los instintos a corto plazo del programador y los intereses a largo plazo del proyecto (21).

Los equipos de desarrollos que utilizan XP como metodología llevan a cabo simultáneamente cada tarea del desarrollo de software. El análisis, diseño, implementación, pruebas e incluso el despliegue, ocurren muy frecuentemente, y todo este trabajo en XP se realiza a través de iteraciones haciendo en cada semana una pequeña parte del análisis, otra parte del diseño, de la implementación y de las pruebas y así sucesivamente, enfocándose desarrollar las historias de usuarios que están conformadas por pequeñas funcionalidades o partes de estas, pero que son de valor para el cliente. En cada semana trabajan en todas las fases de desarrollo de cada funcionalidad y al final de la semana se despliega el software para una revisión interna (22).

En XP se realizan en cada iteración las siguientes actividades, las cuales están basadas en las fases tradicionales del desarrollo de software:

- **Planificación:**

  Cada equipo incluye un experto en el negocio, el cuál es responsable de las decisiones del negocio por la parte del cliente. Este debe guiar el proyecto en la dirección correcta, aclarando

la visión del mismo, creando funcionalidades, construyendo el plan de liberación y manejando los riesgos, auxiliándose de estimaciones y sugerencias provistas por los programadores.

El esfuerzo en la planificación es más fuerte en las primeras semanas del proyecto. Durante el resto del tiempo el cliente continúa revisando y mejorando la visión y el plan de proyecto para proveer de nuevas oportunidades y eventos inesperados. En adición al plan general, el equipo crea un plan detallado al inicio de cada semana al inicio de cada iteración. El equipo revisa las bases todos los días en un espacio de trabajo para mantener informado a todos del progreso del proyecto.

- **Análisis:**

  En vez de definir los requisitos del sistema, con el equipo de desarrollo todo el tiempo debe mantenerse una persona calificada que sea capaz de determinar lo que el software debe o no hacer. Son los encargados de descubrir los requerimientos generales del software, usando su conocimiento como cliente combinado con algunas técnicas, y detallarlos muy bien antes de que sean implementados por el programador.

  Algunos requerimientos son difíciles de entender y definir, por lo que deben de formalizarse por el cliente con la asistencia de los probadores creando las pruebas del cliente mediante que no son más que ejemplos detallados y chequeados. Estas pruebas son creadas a medida que el programador implementa las historias de usuario, y para ayudar con la comunicación se utiliza un lenguaje común en el diseño y código.

- **Diseño y codificación:**

  XP utiliza un diseño y arquitectura incrementales para continuamente crear y mejorar el diseño con pequeños avances. El trabajo se basa en un desarrollo guiado por pruebas que une inseparablemente las pruebas, el código, el diseño y la arquitectura. Para apoyar esta forma de desarrollo los programadores trabajan en pares, son los responsables de manejar su entorno de desarrollo, mantienen un estándar de codificación, comparten el dominio del código y lo integran cada un período de tiempo de pocas horas asegurándose de que cada integración se pueda desplegar.

- **Pruebas:**

  Cada miembro del equipo hace su propia contribución a la calidad del software, produciendo solamente un puñado de fallos por mes en un trabajo completo. Los programadores en el desarrollo guiado por pruebas crean las pruebas unitarias y de integración para asegurar que el software realiza las funciones que el programador pretende que haga. De igual forma que las pruebas del cliente aseguran que el programador cumpla con las expectativas del cliente. El

cliente revisa el trabajo en progreso, para asegurarse que la interfaz de usuario funciona de la forma esperada, y produce ejemplos, para el programador, trucan las reglas del negocio.

Los probadores ayudan a entender al equipo como sus esfuerzos producen código de calidad, usando pruebas exploratorios para encontrar fallos y brechas en el software y realizan pruebas a las características no funcionales del sistema como desempeño y estabilidad. Cuando estos encuentran una falla, los programadores automáticamente crean pruebas que demuestren que la falla ha sido solucionada.

- **Despliegue:**

  El equipo de desarrollo mantiene el software listo para desplegar al final de cada iteración, y lo despliega cada semana para la preparación del demo semanal. El despliegue a los clientes reales se organiza en dependencia de las necesidades reales del negocio.

Las características de esta metodología la hacen propicia para llevar a cabo el desarrollo de la solución propuesta. Lo primero que se tiene en cuenta es que se trata de una metodología ágil y que no genera demasiados artefactos como otras metodologías más tradicionales por lo que se consagra más el tiempo al desarrollo de la aplicación. Por otra parte esta metodología incluye dentro del equipo de desarrollo un cliente o persona capacitada en el negocio para asumir los cambios y guiar mantener en el equipo la visión del proyecto y el equipo de desarrollo de esta solución, está plenamente capacitado en el negocio. Por último una misma persona capacitada podrá llevar a cabo los roles que se plantean en esta metodología, teniendo en cuenta la envergadura del proyecto.

1.7 Conclusiones del capítulo

En este capítulo se presentan los aspectos más relevantes del proceso de Análisis y Control Financiero Consular que servirán de base para cumplir con el objetivo de este trabajo. Se abordaron algunos de los lenguajes de programación Web, donde PHP se seleccionó para implementar la solución en la parte del servidor y el JavaScript como lenguaje del lado del cliente. Se utilizará jQuery como biblioteca de JavaScript para facilitar el trabajo en la creación de las páginas Web dinámicas. Como entorno de desarrollo integrado se utilizará NetBeans debido a la experiencia que posee el equipo con la misma. Como gestor de base de datos se definió el uso de MySQL, por su facilidad de integración con el lenguaje de programación, entre otras características, y como metodología de desarrollo se hará uso de XP debido a sus características.

Capítulo 2. Descripción de la solución

En este capítulo se describen las principales características que presentará el sistema PanaConsul. Se define el modelo de dominio para complementar el entendimiento del sistema. Se muestran los requisitos del mismo, se definen las historias de usuarios y para lograr un mejor entendimiento de la aplicación se realizará el diagrama de clases como complemento de la metodología definida. También se planifican las iteraciones en las que se implementará cada una de las historias de usuario y se estimará el tiempo de duración de cada tarea de la ingeniería. Todo esto contribuye a lograr una implementación correcta y eficaz.

2.1 Identificación del problema

El análisis de las operaciones consulares por parte de la Autoridad Marítima de Panamá es necesario para que la misma lleve un control sistemático sobre estas operaciones. La Contraloría General de la República de Panamá en las auditorías realizada a la AMP ha señalado como deficiencia que este análisis demora demasiado tiempo debido a que la información tarda hasta tres meses en llegar a Panamá y ser procesada por el Departamento de Control Financiero Consular de la AMP que es el encargado de realizar dicho análisis. Esta deficiencia trae consigo que la AMP tenga que tomar decisiones con cifras aproximadas dado el atraso existente con este proceso por lo que se hace necesaria una herramienta que sea capaz de registrar las operaciones consulares, realizar el análisis de la información actualizada y que los analistas sean capaces de realizar el control de dicha información para que la AMP pueda tomar decisiones basándose en información real. El presente trabajo de diploma desarrollará una herramienta que sea capaz de realizar este proceso con la información actualizada y disponible.

2.2 Modelo de Dominio

El modelo de dominio es una representación de los conceptos del dominio del problema en el mundo real, para lo cual se necesita definir las clases conceptuales más significativas que brindan un mejor entendimiento del problema. Este diagrama se conforma con abstracciones de los objetos reales asociados al proyecto a realizar y las relaciones entre ellos. El modelo de dominio ayuda a comprender los conceptos que manejan los usuarios y con los que deberá trabajar la aplicación. A pesar de que este modelo no forma parte de la metodología XP se decide realizar el mismo para un mejor entendimiento del problema.

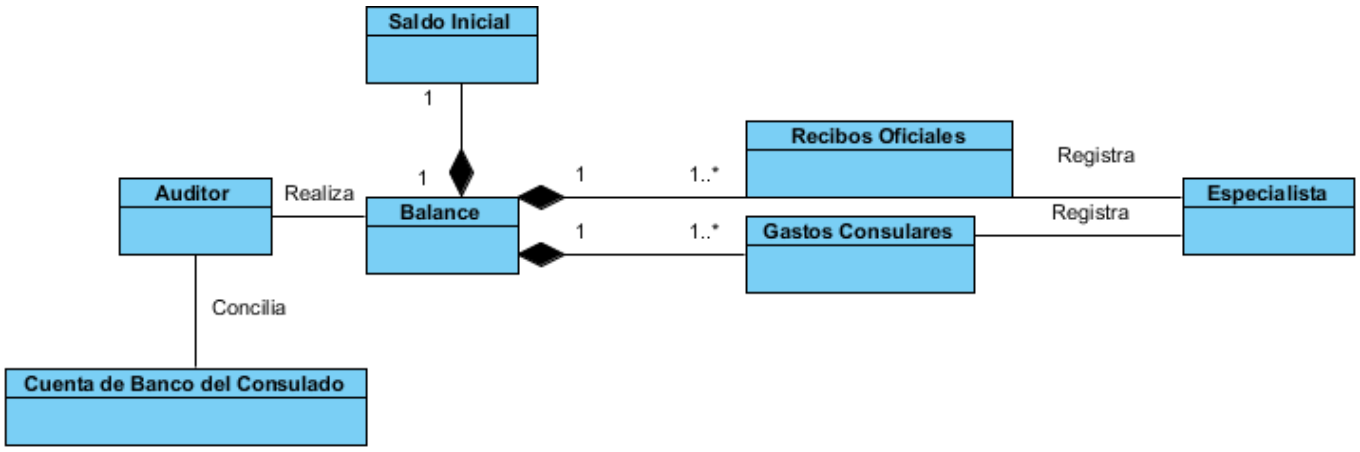

Figura 1: Modelo de Dominio del Sistema

Saldo Inicial: Saldo inicial que presenta la cuenta bancaria a inicios del mes.

Auditor: Persona encargada de fiscalizar las cuentas de uno o más consulados.

Balance: Es el saldo resultante de la suma de los montos de los Recibos Oficiales y resta de los Gastos Consulares al saldo inicial del mes. Este saldo conforma la Remesa Consular del mes entrante.

Recibos Oficiales: Documentos legales que hacen efectivo el cobro de las actuaciones consulares.

Gastos Consulares: Cheques o transferencias que hacen efectivo los gastos consulares.

Especialista: Persona encargada de gestionar los ingresos y gastos del consulado.

Cuenta de Banco del Consulado: Cuenta bancaria que contiene los fondos capitales del consulado.

2.3 Propuesta de la aplicación a desarrollar.

PanaConsul

La herramienta PanaConsul para el análisis y control financiero de los consulados de Panamá es una aplicación Web la cual puede ser accedida a través de Internet desde cualquier consulado panameño en cualquier país del mundo, de tal forma que la información recogida y procesada se mantenga todo el tiempo disponible para la Autoridad Marítima de Panamá. Esta aplicación brinda la posibilidad de registrar las actuaciones consulares que se contabilizan y forman parte de las finanzas del consulado panameño. Estas actuaciones pueden ser los pagos realizados por el Consulado, los Recibos Oficiales y los Recibos de Ingresos, dichas actuaciones son registrados en la aplicación para luego procesar y analizar la información para así contribuir en la toma de decisiones económicas en el Consulado de Panamá. La aplicación se encarga de generar en forma de reporte la información necesaria, producida como resultado de los análisis financieros, y estos reportes pueden ser exportados en formato PDF o en Microsoft Excel.

2.4 Historias de usuarios

La metodología XP para especificar los requisitos que debe cumplir la aplicación en desarrollo utiliza las

historias de usuarios. Estas historias no son tarjetas en papel que se encargan de describir brevemente las características que el sistema debe poseer, ya sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuarios es flexible y muy dinámico dado que en cualquier momento estas pueden ser reemplazadas por otras historias de usuarios más específicas o más generales, agregar nuevas o pueden ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (23).

Para el desarrollo de la aplicación PanaConsul se definieron 13 historias de usuarios, 4 con prioridad muy alta para el negocio, 5 con prioridad alta y las 4 restantes con prioridad media. De estas historias de usuarios se muestran a continuación las de mayor prioridad.

A continuación se muestran las principales historias de usuarios del sistema y una breve descripción de cada una.

En la Tabla 1. ***Historia de Usuario: Registrar Recibo Oficial.*** se muestra la historia de usuario número 2 que describe la funcionalidad de la aplicación que se encarga de registrar los recibos oficiales. Esta Historia tiene una prioridad muy alta por el valor que representa para la aplicación ya que es uno de los puntos de partida para proveer parte de la información que manejará la aplicación por lo que el riesgo que representa su no implementación es alto. Se estima que su correcta implementación dure 2 semanas. Esta historia de usuario incluye un prototipo de la interfaz de usuario que presentará esta funcionalidad.

Historia de Usuario	
Número: 2	**Nombre de la Historia de Usuario:** Registrar Recibo Oficial
Cantidad de modificaciones a la Historia de Usuario: ninguna	
Usuario: Alejandro	**Iteración asignada:** 1ra
Prioridad en negocio: Muy Alta	**Puntos estimados:** 2 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 2 semanas
Descripción: Permitirá el registro y eliminación de los recibos oficiales en el sistema.	
Observaciones:	
Prototipo de interfaces:	

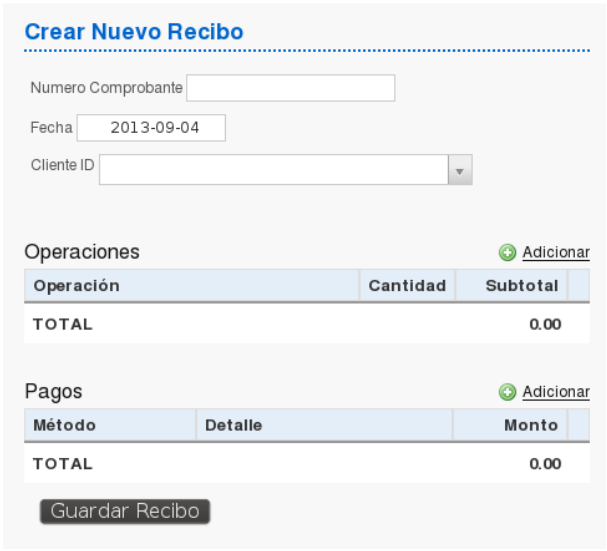

Tabla 1. Historia de Usuario: Registrar Recibo Oficial.

La Tabla 2. ***Historia de Usuario: Registrar Recibo de Impuesto.Tabla 1. Historia de Usuario: Registrar Recibo Oficial.*** muestra la historia de usuario número 3 que describe la funcionalidad de la aplicación que se encarga de registrar los recibos de impuesto. La prioridad de esta historia de usuario es Muy Alta al igual que la anterior, producto de ser otro de los puntos de partida para proveer de información que manejará la aplicación y el riesgo que representa su no implementación es alto. Se estima que su correcta implementación dure 2 semanas e incluye un prototipo de la interfaz de usuario que presentará esta funcionalidad.

Historia de Usuario	
Número: 3	**Nombre de la Historia de Usuario:** Registrar Recibo de Impuesto
Cantidad de modificaciones a la Historia de Usuario: ninguna	
Usuario: Alejandro	**Iteración asignada:** 1ra
Prioridad en negocio: Muy Alta	**Puntos estimados:** 2 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 2 semanas
Descripción: Permitirá el registro y eliminación de los recibos de impuesto en el sistema.	
Observaciones:	
Prototipo de interfaces:	

Tabla 2. Historia de Usuario: Registrar Recibo de Impuesto.

La otra historia de usuario que representa una de las funcionalidades que registra información a manejar por la aplicación es la número 4 y se muestra en la Tabla 3. ***Historia de Usuario: Registrar Pago.***. Esta funcionalidad de la aplicación que se encarga de registrar gastos por concepto de pago entre otros. Esta historia tiene una prioridad Muy Alta por el valor que representa para la aplicación y el riesgo que constituye su no implementación es alto. Se estima que la correcta implementación de esta historia de usuario dure 2 semanas. También se incluye un prototipo de la interfaz de usuario que presentará esta funcionalidad.

Historia de Usuario	
Número: 4	**Nombre de la Historia de Usuario:** Registrar Pago
Cantidad de modificaciones a la Historia de Usuario: ninguna	
Usuario: Alejandro	**Iteración asignada:** 1ra
Prioridad en negocio: Muy Alta	**Puntos estimados:** 2 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 2 semanas
Descripción: Permitirá el registro y eliminación de los pagos en el sistema.	
Observaciones:	
Prototipo de interfaces:	

Tabla 3. Historia de Usuario: Registrar Pago.

Por último en la Tabla 4. ***Historia de Usuario: Conciliación Diaria.*** se muestra la historia de usuario número 5 que describe la funcionalidad de la aplicación que se encarga realizar la conciliación diaria de los ingresos y gastos. Esta Historia tiene una prioridad Muy Alta puesto que los análisis de la información parten de la conciliación de los cheques y comprobantes con el saldo de la cuenta bancaria. El riesgo que representa su no implementación es alto y se estima que su correcta implementación dure 3 semanas. Se incluye un prototipo de la interfaz de usuario que presentará esta funcionalidad.

Historia de Usuario	
Número: 5	**Nombre de la Historia de Usuario:** Conciliación diaria
Cantidad de modificaciones a la Historia de Usuario: ninguna	
Usuario: Alejandro	**Iteración asignada:** 1ra
Prioridad en negocio: Muy Alta	**Puntos estimados:** 3 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 3 semanas
Descripción: Permitirá la conciliación diaria de los pagos y los recibos.	
Observaciones: La aplicación debe permitir retroceder el último día conciliado.	
Prototipo de interfaces:	

Tabla 4. Historia de Usuario: Conciliación Diaria.

2.5 Lista de reserva del producto

Para el desarrollo de una aplicación, se hace necesario tener conocimiento de los requerimientos que debe presentar el sistema ya sean funcionales o no funcionales. En la metodología XP estos se representan en un artefacto conocido como lista de reserva del producto en la cual se ordenan los requerimientos según la prioridad y con la estimación de cada uno para su implementación por semanas y el rol que realiza la estimación del requerimiento.

Item *	Descripción	Estimación	Estimado por
Prioridad: Muy alta			
1	Seguridad del sistema	1	Analista
2	Registrar Recibo Oficial	2	Analista
3	Registrar Recibo de Impuesto	2	Analista
4	Registrar Pago	2	Analista
5	Conciliación diaria	2	Analista
Prioridad: Alta			
6	Recaudos Consulares (Cuadro A)	1	Analista
7	Detalle Mensual de Gatos Autorizados (Anexo 4)	1	Analista
8	Recaudos y Gastos Consulares	1	Analista
9	Pre-análisis Consular	1	Analista
10	Libro de Banco	1	Analista
Prioridad: Media			
11	Detalle Recibos Oficiales	1	Analista

12	Detalle Recibos de Impuesto	1	Analista
13	Cheques en Circulación	1	Analista
14	Conciliación Bancaria	1	Analista
Prioridad: Baja			

Tabla 5. Lista de Reserva del Producto.

2.6 Tareas de la Ingeniería

Las historias de usuarios son descompuestas en tareas específicas descritas desde la perspectiva del desarrollador, estas son las llamadas tareas de la ingeniería. Estas tareas describen objetivos que se deben cumplir, a cada una se le planifica un estimado del tiempo en que se debe realizar y se le asigna a un programador la responsabilidad de llevar a cabo dicha tarea. En las siguientes tablas se muestran algunas de las tareas de la ingeniería correspondiente a las principales historias de usuarios.

La Tabla 6. ***Tarea 4 de la ingeniería.*** contiene la descripción de la tarea número 4 correspondiente a la historia de usuario Registrar Recibo Oficial. Esta tarea consiste en el desarrollo de la funcionalidad de registrar los recibos oficiales al sistema.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario** 2
Nombre Tarea: Registrar Recibo Oficial	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Alejandro Pérez de la Cruz	
Descripción: Llevar a cabo el registro de los recibos oficiales.	

Tabla 6. Tarea 4 de la ingeniería.

La tarea número 7 pertenece a la historia de usuario Registrar Recibo de Impuesto. Esta terea consiste en el desarrollo de la funcionalidad de eliminar un recibo de impuesto del sistema, lo cual consiste en un objetivo que debe cumplir la historia de usuario a la que pertenece. Esta tarea se puede observara continuación en la Tabla 7. **Tarea 7 de la ingeniería.**

Tarea de Ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** 3
Nombre Tarea: Anular Recibo de Impuesto	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Alejandro Pérez de la Cruz	
Descripción: Llevar a cabo el registro de los recibos oficiales.	

Tabla 7. Tarea 7 de la ingeniería.

En la Tabla 8. **Tarea 10 de la ingeniería** y la Tabla 9. **Tarea 11 de la ingeniería** que se encuentran a continuación, se muestran las tareas de la ingeniería pertenecientes a la historia de usuario conciliación diaria. Estas tareas representan la implementación las funcionalidades específicas de la historia de usuario a la que pertenecen.

Tarea de Ingeniería	
Número Tarea: 10	**Número Historia de Usuario:** 5
Nombre Tarea: Realizar la conciliación diaria	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Alejandro Pérez de la Cruz	
Descripción: Llevar a cabo el registro de los recibos oficiales.	

Tabla 8. Tarea 10 de la ingeniería

Tarea de Ingeniería	
Número Tarea: 11	**Número Historia de Usuario:** 5
Nombre Tarea: Retroceder la última conciliación	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Alejandro Pérez de la Cruz	
Descripción: Llevar a cabo el registro de los recibos oficiales.	

Tabla 9. Tarea 11 de la ingeniería

2.7 Plan de iteraciones

El plan de iteraciones es el que mantiene el ritmo del proyecto. Este plan contiene un estimado del tiempo de duración de cada liberación, la planificación de las historias de usuarios a implementar para cada una de las liberaciones del producto y determina el orden de implementación de las mismas. Este plan de iteraciones es el que rige el trabajo de los programadores que son los encargados de implementar cada historia de usuario.

Release	Descripción de la iteración	Orden de las HU a Implementar	Duración
1	Implementar las historias de usuario con prioridad muy alta. Estas historias de usuario se encargarán de introducir la información con la cual el sistema deberá interactuar.	1-2-3-4-5	9 semanas
2	Implementar las historias de usuario de prioridad media. Estas historias de usuarios se encargarán de producir los modelos y reportes más importantes, generados a partir de la información registrada en el sistema.	5-6-7-8-9	5 semanas
3	Implementar las historias de usuario de prioridad media. Estas historias de usuarios se encargarán de producir el resto de los modelos y reportes que debe generar el sistema.	11-12-13	3 semanas

Tabla 10. Plan de Iteraciones.

2.8 Diseño del sistema

El buen diseño de las clases de un sistema incrementa la calidad del mismo de forma proporcional y provee un buen desempeño del mismo. En otras metodologías este diseño se realiza a través de diagramas, en el caso de XP se realiza mediante las tarjetas Clase Responsabilidad Colaboración (CRC) estructurando el conjunto de todas las tarjetas para ayudar al refinamiento de las clases.

2.8.1 Tarjetas CRC

Las tarjetas CRC no son más que simples fichas que representan a una entidad en el sistema y sus

principales características de son su simpleza y adaptabilidad. En estas tarjetas aparece el nombre de la clase, las responsabilidades que cumple y las colaboraciones. Las responsabilidades son las acciones que puede realizar un objeto de esa clase o la información que provee, pero esta no siempre contiene la información necesaria por lo que necesita de las colaboraciones de otras clases para poder llevar a cabo su cometido (24).

Tarjeta CRC	
Clase: *ComprobantesController*	
Responsabilidades	**Colaboraciones**
Registrar Recibo Oficial Anular Recibo Oficial	*Comprobantes* *ComprobantesPag Details* *ComprobantesOperaciones* *ComprobantesOperaciónConceptos*

Tabla 11. Tarjeta CRC de la clase ComprobantesController.

Tarjeta CRC	
Clase: *ComprobantesImpuestoController*	
Responsabilidades	**Colaboraciones**
Registrar Recibo de Impuesto Anular Recibo de Impuesto	*ComprobantesImpuesto* *ComprobantesImpuestoConcepto* *ComprobantesImpuestoPagos*

Tabla 12. Tarjeta CRC de la clase ComprobantesImpuestoController.

Tarjeta CRC	
Clase: *ChequesController*	
Responsabilidades	**Colaboraciones**
Registrar Pago Anular Pago	*Cheques*

Tabla 13. Tarjeta CRC de la clase ChequesController.

Tarjeta CRC	
Clase: *ConciliacionController*	
Responsabilidades	**Colaboraciones**
Conciliación Diaria Retroceder Conciliación	*DeudasConciliacion* *LibroBanco*

	Comprobantes *ComprobantesImpuesto* *Cheques*

Tabla 14. Tarjeta CRC de la clase ConciliacionController.

2.8.2 Diagrama de clases

El diagrama de clases es un modelo definido en las metodologías tradicionales que permite al programador tener una visión más amplia del sistema y en este diagrama se pueden apreciar a grandes rasgos, las principales clases del mismo y las principales interacciones entre ellas. En el caso del diagrama presentado en la Figura 2***: Diagrama de Clases*** se muestran las clases organizadas por las capas según el patrón de arquitectura que sigue la aplicación, el cual se explica en el siguiente epígrafe.

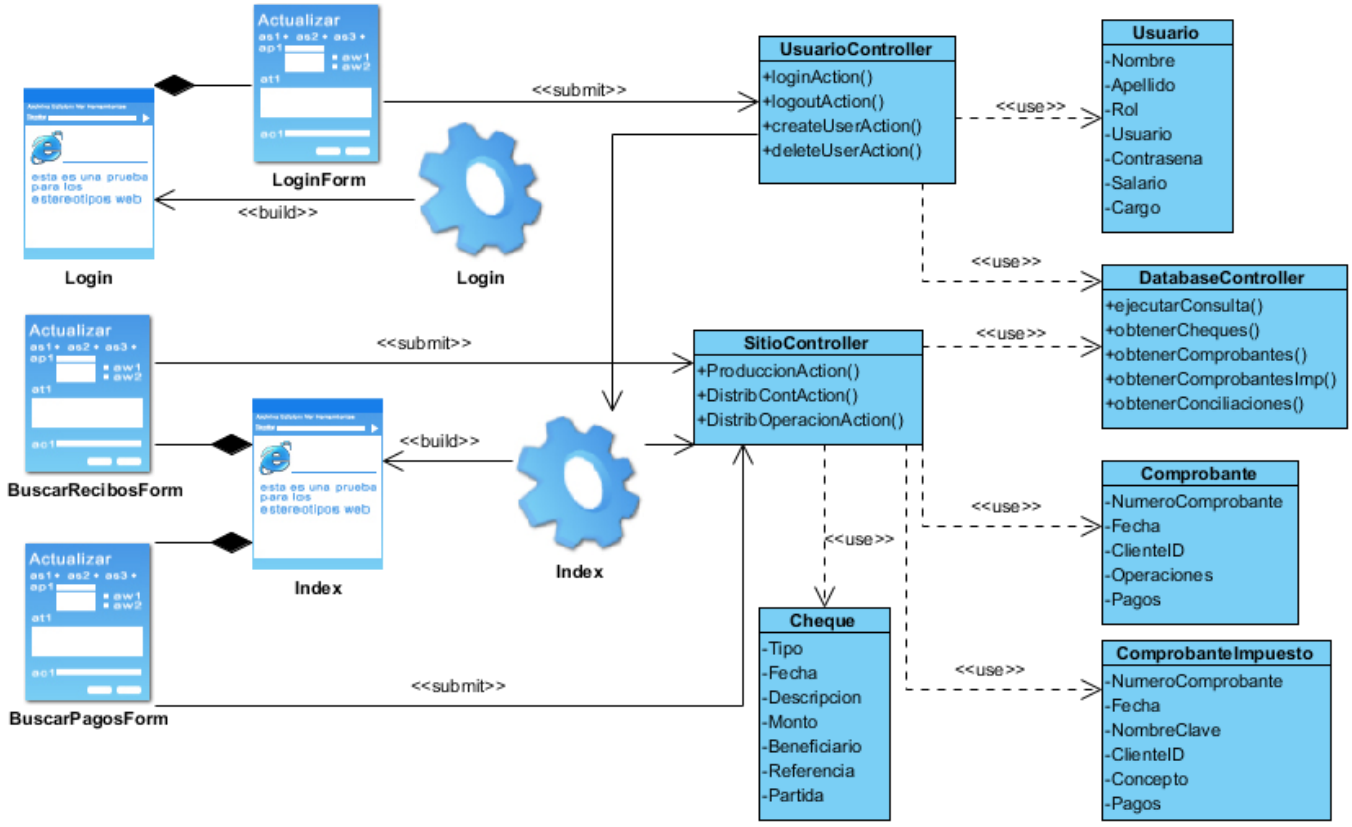

Figura 2: Diagrama de Clases

2.9 Patrón de Arquitectura

Los patrones de arquitectura son patrones de alto nivel en un sistema y expresan fundamentalmente el esquema organizativo del software. Estos proveen de una serie de subsistemas predefinidos, especifican sus responsabilidades e incluyen guías y reglas para organizar la relación entre ellos; lo que ayuda a especificar la estructura fundamental de la aplicación (25).

En este trabajo se propone el desarrollo de una aplicación Web siguiendo el patrón Modelo-Vista-Controlador (MVC), el cual organiza y separa las clases del software en dependencia de tres roles que cumplen las clases dentro del sistema. El rol del modelo lo siguen las clases que contienen las funcionalidades básicas del negocio y los datos. La vista está compuesta por las clases que extraen los datos del modelo y los acomoda para presentarlos al usuario. Por último el rol de controlador lo realizan las clases que dirigen el flujo de la aplicación recibiendo las solicitudes del usuario y delegando las acciones correspondientes a las clases del modelo y de la vista.

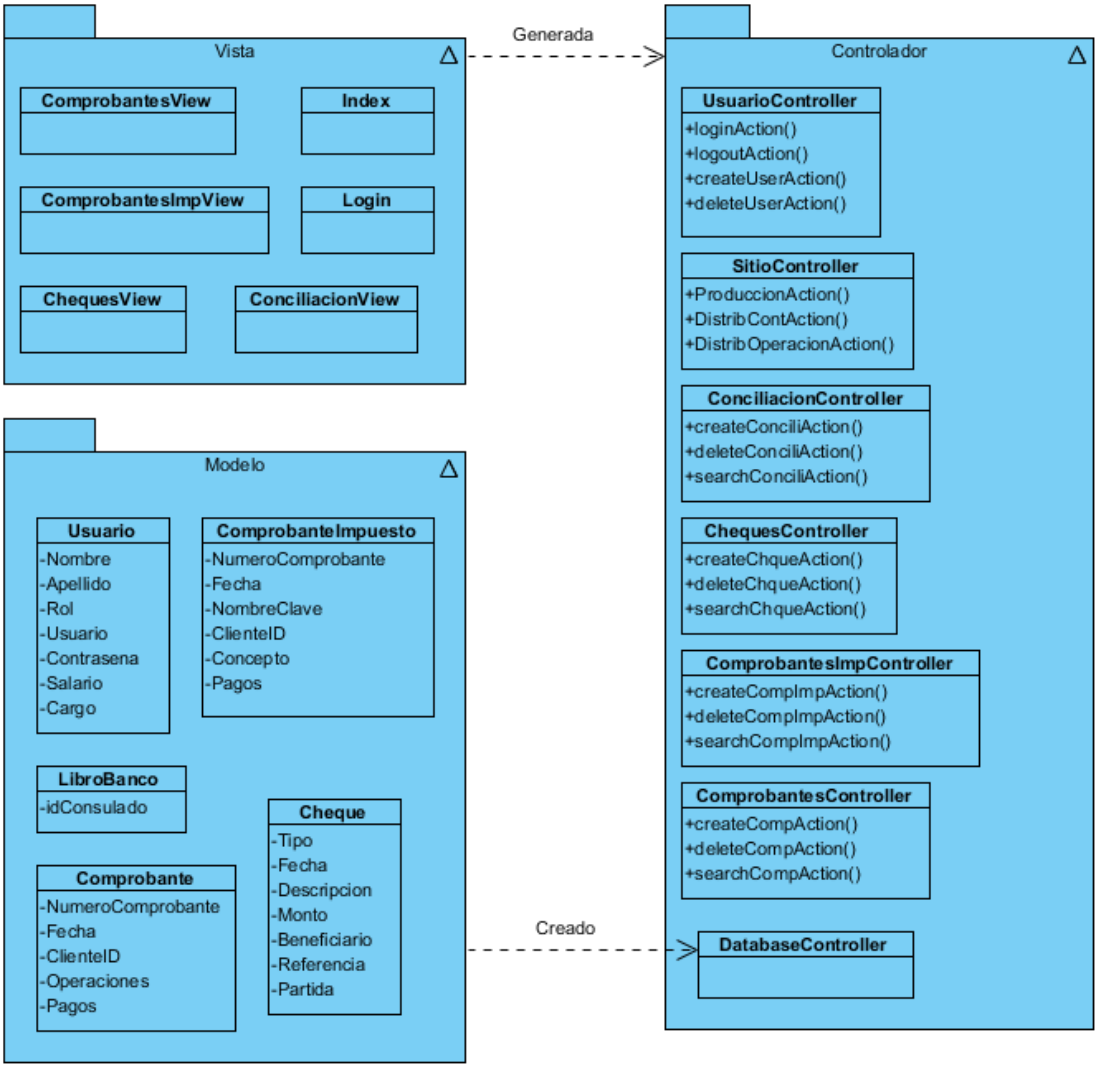

Figura 3: Diagrama del patrón Modelo-Vista-Controlador

En la Figura 3***: Diagrama del patrón Modelo-Vista-Controlador*** observa cómo están distribuidas las principales clases de la aplicación según el rol que ejecutan dentro del patrón arquitectónico. Las clases del modelo son nombradas por el objeto del negocio al cual representan dado que contienen la misma información y realizan las mismas funciones del negocio, dentro del sistema. Las clases que pertenecen

a la vista terminan seguidas de la palabra *View* para diferenciarlas, estas clases se encargan de mostrar al usuario la información que se encuentra contenida en las clases del modelo. Por último, las clases controladoras que son las que su nombre culmina con la palabra *Controller*, son las encargadas de procesar las peticiones del usuario haciendo uso de las clases del modelo para obtener la información y delegando a las vistas la organización del resultado de la petición.

2.10 Patrones de Diseño

Los subsistemas que componen la arquitectura del software, normalmente consisten en pequeñas unidades de arquitectura las cuales son descritas usando patrones de diseño. Estos patrones son más pequeños que los patrones de arquitectura y tienen a ser dependientes de algún lenguaje o paradigma de programación (25). En la programación orientada a objeto existen dos grupos de patrones de diseños, los patrones de asignación de responsabilidad o GRASP (*General Responsibility Assignment Software Patterns*) y el otro grupo de patrones aparecen en el libro "*Design Patterns*" de los autores Gamma, Helm, Johnson y Vlissides, conocido también como "Banda de los Cuatro" o "*Gang of Four*" de ahí que a estos se le conocen como patrones GoF.

Los patrones GRASP son importantes para el diseño de programas orientados a objetos, dado que la decisión acerca de asignar responsabilidad tiene suma importancia cuando se quiere hacer software robusto y fiable. Para esto debe quedar bien definida la función que cumple cada objeto dentro del sistema. Dentro de estos patrones que se siguen para asignar estar responsabilidades se encuentran:

Experto en Información: Este patrón nos indica que la clase tiene la información necesaria para realizar cierta responsabilidad debe ser quien la lleve a cabo. Esto se evidencia en las clases controladoras ejemplo *ChequesController*, esta clase es la encargada de atender las peticiones del usuario correspondiente a las acciones que realiza el sistema sobre los cheques por lo que esta misma debe contener la información necesaria para realizar la búsqueda de cheques de entre todos.

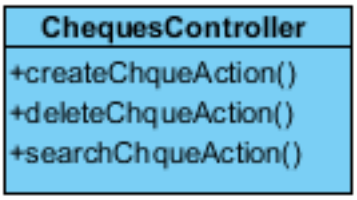

Figura 4: Clase ChequesController

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. Un ejemplo de este patrón en este trabajo se evidencia en la clase *Comprobante* del modelo, la cual es la clase encargada de crear las instancias de los comprobantes en el sistema.

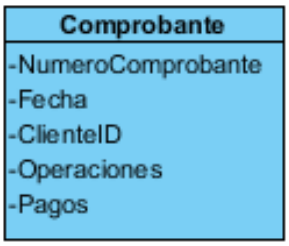

Figura 5: Clase Comprobante.

Alta Cohesión y Bajo Acoplamiento: Estos patrones consisten en hacer que una clase contenga responsabilidades altamente relacionadas y que la dependencia con otras clases sea mínima. Estos patrones se evidencian en la clase *DatabaseController*, dado que esta clase es la encargada de realizar las operaciones con la base de datos, por lo que todas las tareas que realiza están relacionadas con la interacción del sistema con la base de dato y a su vez no depende de otras clases para realizar su trabajo.

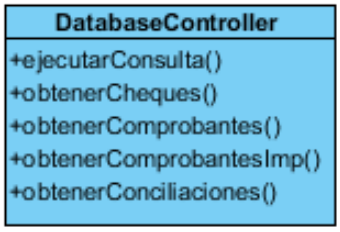

Figura 6: Clase DatabaseController.

Controlador: Este patrón consiste en asignar a una clase la responsabilidad de recibir o manejar un mensaje de evento a un subsistema. En el caso del patrón arquitectónico que se utiliza en este trabajo, todas las clases controladoras tienen la responsabilidad de atender las peticiones del usuario como en la clase *SitioController*, la cual se encarga de pasar las peticiones relacionadas con la interfaz principal del sistema. Otro ejemplo también es la clase *DatabaseController* que se encarga de manejar las peticiones del resto de las clases hacia la base de datos.

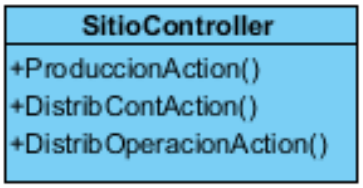

Figura 7: Clase SitioController.

2.11 Estándares de codificación

Los estándares de codificación son modelos de programación enfocados en la estructura y apariencia física del código para facilitar la lectura y la comprensión para poder realizar mantenimientos en el futuro. La metodología XP promueve la codificación basándose en estándares para que el código sea entendible por todo el equipo de desarrollo para que se facilite la recodificación.

Identación

El identado es de 4 espacios, y no es recomendable el uso de la tabulación debido a que no existe un estándar que determine con precisión el ancho que va a producir la tabulación, aunque según el editor de código esta puede configurarse.

Longitud de la Línea

Evitar líneas con más de 80 caracteres, en caso de ocurrir que una sentencia sobrepase una línea simple del editor, se debe crear otra nueva línea. La separación se puede realizar después de un operador, preferiblemente luego de una coma. Al realizar la ruptura la siguiente línea debe ser identada con 5 espacios.

Comentarios

Para comentar una sentencia deben usarse comentarios de una sola línea. Para la documentación formal o para comentar porciones de código se usarán los comentarios de bloques. Los comentarios en bloque cumplirán con la misma identación y longitud de línea que el código.

Declaración de variables y funciones

Para declarar los nombres de las variables y las funciones se utilizará la notación Camel Case. Cada variable debe de ser declarada en una línea y comentada. En el caso de las funciones no debe haber espacio entre el nombre y el paréntesis izquierdo, ni entre el paréntesis derecho y la llave de comienzo del cuerpo de la función. Las sentencias del cuerpo deben estar en la línea siguiente. La llave que cierra debe estar alineada con la línea de declaración de la función.

Identificadores

Los identificadores pueden estar formados por cualesquiera de las letras desde la A hasta la Z ya sea en minúsculas o mayúsculas, los dígitos del 0 al 9 y el guión bajo. No deben usarse caracteres acentuados, ni caracteres extraños como la letra ñ o las diéresis, ni símbolos de ningún tipo. El uso del guión bajo solo se usará al inicio de un identificado y será reservado únicamente para los nombres de las clases abstractas y las funciones recursivas.

2.12 Interfaz principal de la aplicación

En la Figura *8:* ***Interfaz principal de PanaConsul.*** se muestra la interfaz principal de la aplicación PanaConsul. La misma está conformada por un menú el cual permite acceder a las distintas funcionalidades de la aplicación. Debajo del menú se encuentran el Panel de Inicio el cual presenta a la izquierda, un listado los gastos por concepto de pago y debajo otro listado con los ingresos por concepto de Recibos Oficiales y de Impuestos. En la parte derecha del panel se observa un resumen de los totales de Ingresos, Gastos y Beneficios en cada mes del año en curso y un gráfico interactivo del comportamiento de dichos gastos. En el gráfico se puede mostrar el comportamiento de la producción en general del año en curso y también se puede mostrar dividida conceptos o por operaciones.

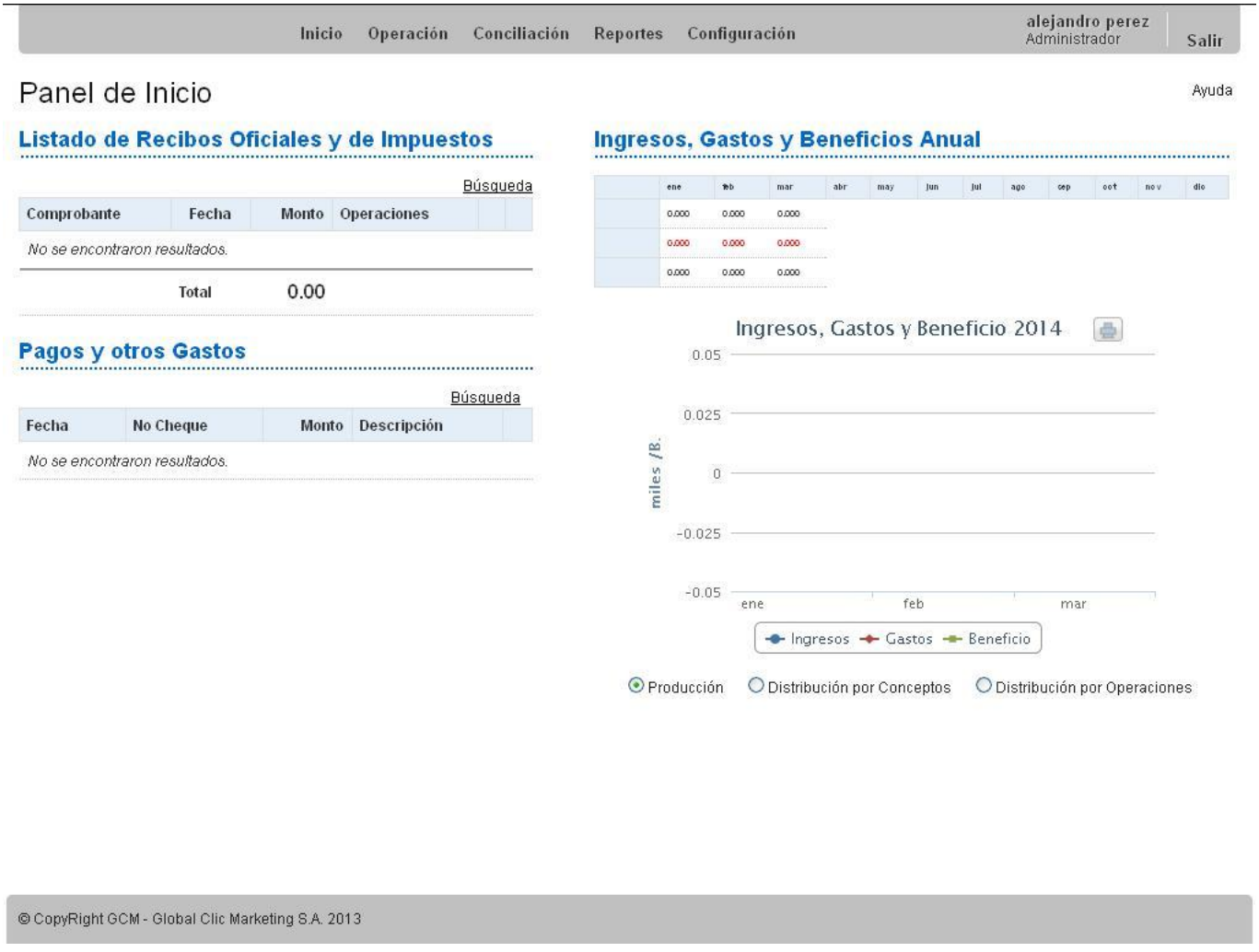

Figura 8: Interfaz principal de PanaConsul.

2.13 Conclusiones del capítulo

En el capítulo han sido analizados los elementos que describen las características y diseño del sistema PanaConsul. Se diseñó una propuesta con las perspectivas de solucionar el problema identificado, para eso fueron desarrolladas las fases de diseño y codificación propuestas por la metodología utilizada en las que se definieron un total de 18 tareas de la ingeniería agrupadas en 5 historias de usuarios. Se generó además el plan de iteraciones para de esta forma especificar en la iteración que se desarrollará cada historia de usuario. Se describen un total de 7 tarjetas CRC como parte del diseño en la metodológica XP. Se muestran los patrones de arquitecturas y de diseño que se emplearon.

Capítulo 3. Validación y pruebas

En la metodología XP existen un grupo de normas para asegurar la implementación correcta de los productos, de forma tal que su funcionamiento sea correcto. En este capítulo se analizará la estrategia de pruebas que propone la metodología, se diseñarán los casos de prueba que se encargarán de validar el correcto funcionamiento de la aplicación, además se mostrarán los resultados arrojados por las pruebas.

3.1 Pruebas

Las pruebas en una aplicación constituyen una parte fundamental del desarrollo de la misma, y el objetivo de estas pruebas es detectar errores que puedan encontrarse en la aplicación. Siguiendo una estrategia y basándose en técnicas a emplear en cada una de las pruebas, se organizan las pruebas en dependencia del objetivo que persiguen. Para evaluar el correcto funcionamiento de un módulo de la aplicación y la integración entre varios de ellos se realizan las llamadas Pruebas Unitarias y las Pruebas de Integración respectivamente. También se utilizan las Pruebas Exploratorias para cubrir los todos los aspectos del sistema así como encontrar errores que no se puedan prever en el funcionamiento de todo un sistema. De igual forma que los ejemplos mencionados anteriormente, existen otras técnicas de pruebas persiguen evaluar un objetivo específico en el funcionamiento del sistema.

3.1.1 Estrategias de pruebas

La metodología XP, la cual guía el desarrollo de la aplicación, define el desarrollo guiado por pruebas lo que hace que el funcionamiento del código que se esté implementando se compruebe constantemente para asegurarse que se esté produciendo código más fiable. Esta metodología propone las pruebas unitarias enfocadas solo a validar un módulo, una clase o un método. Con las pruebas de integración se podrá evaluar la comunicación entre la aplicación y el exterior de la misma, ejemplo con la base de datos. Luego se realizarán las pruebas de funcionalidad con el objetivo de evaluar el correcto funcionamiento de cada funcionalidad del software. Por último se realizarán las pruebas de aceptación por parte del cliente para las cuales se definirán varios casos de pruebas para evaluar todo el funcionamiento del sistema en su conjunto.

Pruebas Unitarias

Estas pruebas se enfocan en validar una parte determinada del código, dado que estas se ejecutan enteramente en memoria, en dependencia de la plataforma, la herramienta de prueba debe ser capaz de ejecutar al menos 100 pruebas unitarias por segundo. Con estas pruebas cada clase, o colección de clases relacionadas (Módulos), debe ser probada por separado por lo que se requiere de un buen diseño de clases, de lo contrario estas pruebas serán muy difíciles de programar. (22)

Pruebas de integración

Estas pruebas están dirigidas a probar el comportamiento del software al utilizar la base de datos, hacer uso del sistema de archivos, comunicarse a través de la red o incluso a tratar de salirse de los límites de la aplicación. Requieren de la preparación de las dependencias externas, y deben comportarse exactamente de la misma forma en cada ejecución, independientemente del orden en que se ejecuten o del estado de la máquina antes de ejecutarse el test. Se realiza un test por cada clase que interactúe con el exterior, y cada uno deberá enfocarse en comprobar una sola integración a la vez. En caso de necesitar más de un test para comprobar una misma integración significa que existe poca cohesión entre las clases y esto se deberá a un mal diseño de las mismas. (22)

Pruebas de funcionalidad

Las pruebas de funcionalidad serán realizadas por el rol de probador y su objetivo es examinar las funcionalidades del software desde la perspectiva del desarrollador. El trabajo del probador en este caso está inspirado en los clientes como punto de partida para introducir variaciones en los datos que probablemente provoquen una ruptura de la aplicación. En estas pruebas, el probador deberá tratar de hacer fallar el software que supuestamente está listo para liberarse (21).

Dentro de este tipo de pruebas se encuentran: las pruebas de estrés destinadas a evaluar un sistema o componente más allá de límites especificados o requerimientos del sistema, las pruebas de rendimiento dedicadas a evaluar el comportamiento del sistema bajo determinados requisitos de ejecución, y las pruebas de usabilidad destinadas a evaluar la magnitud con la que un usuario puede aprender a operar el sistema o componente además de introducir datos e interpretar las salidas (26).

Pruebas de aceptación

Estas pruebas son realizadas por el cliente a todo el sistema integrado y se crean para probar cada historia de usuario. Cada escenario se convertirá en una prueba para que el cliente quede conforme con cada una de las funcionalidades del software (21). A diferencia de las pruebas de funcionalidad, estas pruebas de aceptación están enfocadas a revisar las funcionalidades desde la perspectiva del cliente con el objetivo de encontrar errores o un funcionamiento erróneo en el comportamiento del software frente a casos muy específicos del negocio.

3.1.2 Técnicas de prueba seleccionada

Para llevar a cabo la estrategia de prueba se debe contar con métodos y técnicas a utilizar en la aplicación de las pruebas. Las técnicas más comunes, y de las cuales se hace uso en la realización de las pruebas en este trabajo son las llamadas Pruebas de Caja Blanca y Pruebas de Caja Negra.

Pruebas Estructurales o de Caja Blanca

Estas pruebas constituyen una técnica para que los ingenieros de software puedan verificar si su código funciona de la forma esperada. Estas tienen en cuenta el mecanismo interno de un sistema o componente. Con el uso de la técnica de Caja Blanca se pueden diseñar casos de pruebas que ejerciten caminos independientes en un módulo o unidad del código; repasen la estructura interna de los datos para asegurarse de su validez; revisar las decisiones lógicas en ambos casos; y que ejecuten ciclos cerca de sus límites y dentro de su límite operacional (27).

Esta técnica se utiliza principalmente en las pruebas unitarias para examinar si la unidad se encuentra correctamente codificada, de esta forma se asegura que el código sea sólido antes de integrarlo al resto ya que una vez que el código se encuentra integrado se vuelve más difícil encontrar un fallo en la ejecución del programa. También se utiliza la técnica de Caja Blanca en las pruebas de integración para evaluar la interacción entre la aplicación con las dependencias exteriores, aunque en este caso se puede utilizar también la técnica de Caja Negra en caso de existir múltiples unidades para interactuar con el exterior.

Pruebas de Funcionalidad o Caja Negra

Las pruebas de Caja Negra, también llamadas pruebas funcionales, son pruebas en las que se ignora el mecanismo interno del sistema o componente y se enfoca solamente en las salidas generadas en respuesta a las entradas seleccionadas y las condiciones de ejecución. En este tipo de pruebas, el probador no debería tener acceso al código fuente, considerando el código como una gran caja negra en la cual el probador conoce que información poner dentro de ella, y la misma deberá retornar algo en respuesta. Basándose en los requerimientos, el probador conoce que debe esperar del retorno de la caja negra y realiza las pruebas para asegurarse de que la caja negra devuelve lo que se supone que debe devolver. (26)

Esta técnica, al igual que la de Caja Blanca, también se utiliza en las llamadas pruebas de integración con el objetivo de verificar que todos los componentes trabajen correctamente cuando estén integrados, pero a diferencia de la otra, esta técnica es utilizada para saber si hay pérdida de datos, si los mensajes no se envían correctamente entre los componentes o existe alguna transformación inadecuada en la información a través de todo el sistema (26). Esta técnica también se utiliza en las pruebas funcionales para examinar las especificaciones de requisitos del cliente y planificar los casos de prueba para asegurar el funcionamiento deseado. Luego en las pruebas de aceptación, el cliente ejecuta los test basándose en las expectativas de la funcionalidad del software.

3.1.3 Casos de pruebas basados en historias de usuarios

Las Historias de Usuarios representan las funcionalidades del sistema y utilizando la técnica de caja negra se diseñaron los casos de prueba basados en las historias de usuario para validar el correcto funcionamiento del software. En el caso de las pruebas funcionales que se le realizaron a la aplicación en búsqueda de evaluar la eficacia de la misma para cumplir su objetivo, se utilizó como material de apoyo varios libros de cálculo de la herramienta Excel en los que, por cada libro, se almacena toda la información contable de un mes y que el sistema debe ser capaz de manejar.

Primero se tomó un libro de un mes contable y se volcó toda la información del libro en la aplicación para comprobar que esta llevara a cabo todas las operaciones correctamente y que los cálculos realizados fueran correctos. Luego de introducir los datos y efectuar las operaciones en el sistema, se recogieron las no conformidades resultantes para darles solución y volver a probar el sistema. Estas pruebas se realizaron en 5 iteraciones y en cada iteración se tomó un libro distinto para la obtención de los datos. En las 2 primeras iteraciones se encontraron no conformidades las cuales fueron corregidas inmediatamente. Ya para la 3era iteración las pruebas no arrojaron no conformidades por lo que se iteró 2 veces más en las que se obtuvo el mismo resultado de la 3era iteración.

3.1.4 Presentación de los resultados de las pruebas funcionales

Del primer libro seleccionado para las pruebas en la primera iteración se extrajeron los datos que aparecen en la Tabla 15***: Información de la primera iteración de las prueba*** para introducir en el software. La columna Insertados representa los casos de prueba con los que se ensayó la historia de usuario que aparece en la primera columna. Para aceptar la historia de usuario Conciliación Diaria se realizó la conciliación de cada día laborable del mes recogido en el libro y por último para comprobar que las demás historias de usuarios correspondientes a los reportes que debe generar la aplicación, se probó que cada uno de ellos coincidiera exactamente con los reportes que aparecen en libro del mes correspondiente.

Libro 1	Insertados	No conformidades
Registrar Recibos oficiales	78	6
Registrar Recibo de Impuesto	26	3
Registrar Pago	28	4
Conciliación diaria	23	5
Reportes	9	2

Tabla 15: Información de la primera iteración de las pruebas.

Las cantidades de los Recibos Oficiales y Recibos de Impuesto, así como los Pagos varían en dependencia del

trabajo mensual que se haya hecho por en la embajada y dado que se escogió un libro distinto para cada caso de prueba existe variación en alguno de los datos entre un libro y otro. En el caso de la conciliación diaria varía entre 20 y 23 en dependencia de la cantidad de días hábiles que tenga el mes procesado y por último los reportes son constantes dado que representan los tipos de reportes emitidos por el sistema. En la Tabla 16**: Información de la segunda y tercera iteración de las pruebas.** se muestran los datos de segunda y la tercera iteración. Como se puede apreciar se fueron resolviendo las no conformidades a medida que se realizaron las iteraciones comenzando por la primera.

	Libro 2		Libro 2	
	Insertados	No conformidades	Insertados	No conformidades
Registrar Recibos oficiales	98	2	66	0
Registrar Recibo de Impuesto	24	1	19	0
Registrar Pago	21	2	16	0
Conciliación diaria	22	3	20	0
Reportes	9	1	9	0

Tabla 16: Información de la segunda y tercera iteración de las pruebas.

En la Figura 9**: Gráfico del resultado general de las pruebas.** se muestra un gráfico con los resultados de las 3 primeras iteraciones de pruebas que se realizaron al software. Posteriormente se le realizaron 2 iteraciones más sin que estas arrojaran no conformidades, por lo que se considera que la aplicación tiene un correcto funcionamiento y será evaluada por el cliente por un período de 6 meses debido a la importancia de la información que maneja la misma.

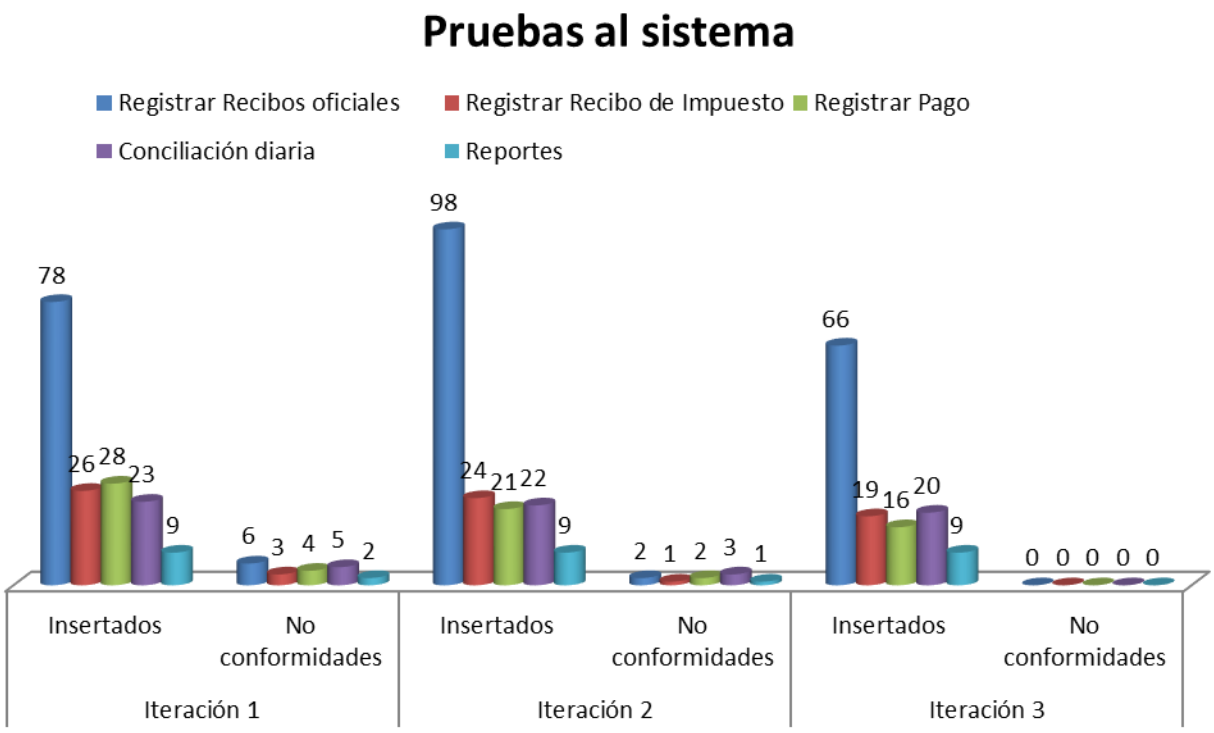

Figura 9: Gráfico del resultado general de las pruebas.

3.2 Conclusiones del capítulo

En este capítulo se definió la estrategia de pruebas a realizar con la aplicación para asegurar un producto funcionalmente correcto. Se abordaron las técnicas de pruebas a utilizar. Las pruebas realizadas a la aplicación desarrollada por este trabajo posibilitaron validar el correcto funcionamiento de la misma. Se mostraron los resultados de los 3 casos de prueba más relevantes. Estas pruebas permitieron identificar fallos de la aplicación relacionados con la implementación del software y corregirlos de manera inmediata antes de pasar a la siguiente iteración de pruebas.

Conclusiones Generales

- Para agilizar el Proceso de Análisis de Informes Consulares, es necesario el desarrollo de una aplicación que cumpla con los requisitos del proceso de Control Financiero.
- Se concluye que con el uso de las tecnologías estudiadas y seleccionadas en este trabajo se desarrolló la aplicación Web PanaConsul, la cual cumple con las necesidades del cliente.
- Mediante análisis y diseño de la aplicación a desarrollar, resultaron 14 historias de usuarios y 20 tareas de ingeniería, las cuales guiaron el proceso de desarrollo para que el mismo se mantuviera cumpliendo con las expectativas del cliente.
- Se realizaron pruebas a la aplicación Web PanaConsul con las que se validó el correcto funcionamiento de la misma para realizar las tareas que se necesita que realice para contribuir al proceso de análisis de los Recaudos Consulares.

Recomendaciones

- Implementar el control de inventario de documentos de valor.

Referencias Bibliográficas

1. **Organization of American States.** Organization of American States. *Organization of American States.* [En línea] 24 de abril de 1963. [Citado el: 23 de septiembre de 2013.] http://www.oas.org/legal/spanish/documentos/convvienaconsulares.htm..

2. **Downey, Allen, Elkner, Jeffrey y Meyers, Chris.** *Aprenda a pensar como un programador de Python.* Wellesley, MA : Green Tea Press, 2002. 0-9716775-0-6.

3. **Brueck, Dave y Tanner, Stephen.** *Hungry Minds.* New York : Hungry Minds, Inc., 2001. 0-7645-4807-7.

4. **Brown, Gregory.** *Ruby Best Practices.* Sebastopol, CA : O'Reilly Media, Inc., 2009. 978-0-596-52300-8.

5. **Comunidad de Ruby.** Ruby. *Ruby. El mejor amigo de un desarrollador.* [En línea] Comunidad de Ruby, 01 de 01 de 2006. [Citado el: 8 de Enero de 2014.] https://www.ruby-lang.org/es/about/.

6. **Bakken, Stig Sæther, y otros.** *Manual de PHP.* [Grupo de documentación de PHP] s.l. : Grupo de documentación de PHP, 2001.

7. **Gallego Vázquez, José Antonio.** *Desarrollo Web con PHP y MySQL.* Madrid : Anaya Multimedia, 2003. 84-415-1525-5.

8. **Brandendaugh, Jerry.** *Programación de aplicaciones Web con JavaScript.* Madrid : EDICIONES ANAYA MULTIMEDIA, 2000. 84-41 5-1 070-9.

9. **Stefanov, Stoyan.** *Object-Oriented JavaScript.* Birmingham, Mumbai : Packt Publishing Ltd., 2008. 978-1-847194-14-5.

10. **jQuery Community Experts.** *jQuery Cookbook.* Sebastopol, CA : O'Reilly Media, Inc., 2010. 978-0-596-15977-1.

11. **Beighley, Lynn.** *jQuery for Dummies.* Indianapolis : Wiley Publishing, Inc., 2010. 978-0-470-58445-3.

12. **ORACLE.** NetBeans IDE. *NetBeans IDE.* [En línea] ORACLE, 01 de 01 de 2010. [Citado el: 4 de 12 de 2013.] https://netbeans.org/index.html.

13. **Date, C. J.** *Introducción a los sistemas de bases de datos.* Naucalpan de Juárez, MX : Pearson Educación de México, 2001. 968-444-419-2.

14. **Momjian, Bruce.** *PostgreSQL Introduction and concepts.* Madrid : Adison-Wesley, 2000. 0-201-70331-9.

15. **The PostgreSQL Global Development Group.** PostgreSQL 9.3.0 Documentation. *PostgreSQL 9.3.0 Documentation.* [En línea] 1 de 01 de 2013. [Citado el: 16 de 11 de 2013.] http://www.postgresql.org/docs/9.3/static/index.html.

16. **Gilfillan, Ian.** *La biblia de MySQL.* s.l. : Anaya Multimedia, 2003. 84-41515581.

17. **Sommerville, Ian.** *Ingeniería de Software.* Madrid : Pearson Educación, S.A., 2005. 84-7829-074-5.

18. **Ashmore, Derek C.** *The J2EE Architect's Handbook.* Lombard, IL : Derek C. Ashmore, 2004. 0972954899.

19. **Zaman Ahmed, Khawar y Umrysh, Cary E.** *Developing Enterprise Java Applications with J2EE and UML.* s.l. : Addison-Wesley Pub Co. 0201738295.

20. **Villafuerte R., Victor.** Extreme Programming. *Extreme Programming.* [En línea] 01 de 01 de 2009. [Citado el: 06 de 12 de 2013.] http://extremeprogramming.host56.com/.

21. **Beck, Kent.** *Extreme Programming Explained: Embrace Change.* New York : Adison-Wesley, 2004. 0201616416.

22. **Shore, James y Warden, Shane.** *The art of agile development.* Sebastopol, CA : O'Reilly Media, Inc., 2007. 0-596-52767-5.

23. *Métodologías ágiles para el desarrollo de software: eXtreme Programming XP.* **Letelier, Patricio y Penadés, María Carmen.** 26, Buenos Aires : Técnica Administrativa, 2006, Vol. 05. 1666-1680.

24. *Identificación y modelado de aspectos tempranos dirigido por tarjetas de responsabilidades y colaboraciones.* **Casas, Sandra y Reinaga, Héctor.** Buenos Aires : Universidad Nacional de Chilecito, 2008. Vol. 14. 978-987-24611-0-2.

25. **Buschmann, Frank, y otros.** *Pattern Oriented Software Architecture. A system of patterns.* Chichester, ENG : John Wiley & Sons, 1996. 0-471-95889-7.

26. **Williams, Laurie.** *Testing Overview and Black-Box Testing Techniques.* 2006.

27. —. *White-Box Testing.* 2006.

28. **Garland, Jeff y Anthony, Richard.** *Large-Scale Software Architecture.* West Sussex, England : John Wiley & Sons Ltd, 2003. 0-470-84849-9.

29. **Martin, Robert C.** *Clean Code. A Handbook of Agile Software Craftsmanship.* Boston, MA : Pearson Education, Inc., 2009. 978-0-13-235088-4.

30. **Pilgrim, Mark.** *Inmersión en Python 3.* San Francisco, California : José Miguel González Aguilera, 2009.

Bibliografía

1. **Organization of American States.** Organization of American States. *Organization of American States.* [En línea] 24 de abril de 1963. [Citado el: 23 de septiembre de 2013.] http://www.oas.org/legal/spanish/documentos/convvienaconsulares.htm..

2. **Downey, Allen, Elkner, Jeffrey y Meyers, Chris.** *Aprenda a pensar como un programador de Python.* Wellesley, MA : Green Tea Press, 2002. 0-9716775-0-6.

3. **Brueck, Dave y Tanner, Stephen.** *Hungry Minds.* New York : Hungry Minds, Inc., 2001. 0-7645-4807-7.

4. **Brown, Gregory.** *Ruby Best Practices.* Sebastopol, CA : O'Reilly Media, Inc., 2009. 978-0-596-52300-8.

5. **Comunidad de Ruby.** Ruby. *Ruby. El mejor amigo de un desarrollador.* [En línea] Comunidad de Ruby, 01 de 01 de 2006. [Citado el: 8 de Enero de 2014.] https://www.ruby-lang.org/es/about/.

6. **Bakken, Stig Sæther, y otros.** *Manual de PHP.* [Grupo de documentación de PHP] s.l. : Grupo de documentación de PHP, 2001.

7. **Gallego Vázquez, José Antonio.** *Desarrollo Web con PHP y MySQL.* Madrid : Anaya Multimedia, 2003. 84-415-1525-5.

8. **Brandendaugh, Jerry.** *Programación de aplicaciones Web con JavaScript.* Madrid : EDlClONES ANAYA MULTIMEDIA, 2000. 84-41 5-1 070-9.

9. **Stefanov, Stoyan.** *Object-Oriented JavaScript.* Birmingham, Mumbai : Packt Publishing Ltd., 2008. 978-1-847194-14-5.

10. **jQuery Community Experts.** *jQuery Cookbook.* Sebastopol, CA : O'Reilly Media, Inc., 2010. 978-0-596-15977-1.

11. **Beighley, Lynn.** *jQuery for Dummies.* Indianapolis : Wiley Publishing, Inc., 2010. 978-0-470-58445-3.

12. **ORACLE.** NetBeans IDE. *NetBeans IDE.* [En línea] ORACLE, 01 de 01 de 2010. [Citado el: 4 de 12 de 2013.] https://netbeans.org/index.html.

13. **Date, C. J.** *Introducción a los sistemas de bases de datos.* Naucalpan de Juárez, MX : Pearson Educación de México, 2001. 968-444-419-2.

14. **Momjian, Bruce.** *PostgreSQL Introduction and concepts.* Madrid : Adison-Wesley, 2000. 0-201-70331-9.

15. **The PostgreSQL Global Development Group.** PostgreSQL 9.3.0 Documentation. *PostgreSQL 9.3.0 Documentation.* [En línea] 1 de 01 de 2013. [Citado el: 16 de 11 de 2013.] http://www.postgresql.org/docs/9.3/static/index.html.

16. **Gilfillan, Ian.** *La biblia de MySQL.* s.l. : Anaya Multimedia, 2003. 84-41515581.

17. **Sommerville, Ian.** *Ingeniería de Software.* Madrid : Pearson Educación, S.A., 2005. 84-7829-074-5.

18. **Ashmore, Derek C.** *The J2EE Architect's Handbook.* Lombard, IL : Derek C. Ashmore, 2004. 0972954899.

19. **Zaman Ahmed, Khawar y Umrysh, Cary E.** *Developing Enterprise Java Applications with J2EE and UML.* s.l. : Addison-Wesley Pub Co. 0201738295.

20. **Villafuerte R., Victor.** Extreme Programming. *Extreme Programming.* [En línea] 01 de 01 de 2009. [Citado el: 06 de 12 de 2013.] http://extremeprogramming.host56.com/.

21. **Beck, Kent.** *Extreme Programming Explained: Embrace Change.* New York : Adison-Wesley, 2004. 0201616416.

22. **Shore, James y Warden, Shane.** *The art of agile development.* Sebastopol, CA : O'Reilly Media, Inc., 2007. 0-596-52767-5.

23. *Métodologías ágiles para el desarrollo de software: eXtreme Programming XP.* **Letelier, Patricio y Penadés, María Carmen.** 26, Buenos Aires : Técnica Administrativa, 2006, Vol. 05. 1666-1680.

24. *Identificación y modelado de aspectos tempranos dirigido por tarjetas de responsabilidades y colaboraciones.* **Casas, Sandra y Reinaga, Héctor.** Buenos Aires : Universidad Nacional de Chilecito, 2008. Vol. 14. 978-987-24611-0-2.

25. **Buschmann, Frank, y otros.** *Pattern Oriented Software Architecture. A system of patterns.* Chichester, ENG : John Wiley & Sons, 1996. 0-471-95889-7.

26. **Williams, Laurie.** *Testing Overview and Black-Box Testing Techniques.* 2006.

27. —. *White-Box Testing.* 2006.

28. **Garland, Jeff y Anthony, Richard.** *Large-Scale Software Architecture.* West Sussex, England : John Wiley & Sons Ltd, 2003. 0-470-84849-9.

29. **Martin, Robert C.** *Clean Code. A Handbook of Agile Software Craftsmanship.* Boston, MA : Pearson Education, Inc., 2009. 978-0-13-235088-4.

30. **Pilgrim, Mark.** *Inmersión en Python 3.* San Francisco, California : José Miguel González Aguilera, 2009.