UCAB Universidad Católica ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA DE INFORMATICA

REINGENIERIA DEL PRODUCTO BANCA EN LINEA EMPRESAS (BOLE) DE MERCANTIL BANCO UNIVERSAL C.A. PARA OPTIMIZAR SU CALIDAD Y SERVICIOS

TRABAJO INSTRUMENTAL DE GRADO

Presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Como parte de los requisitos para optar al título de

INGENIERO EN INFORMATICA

REALIZADO POR TORRES GARCIA, Leonardo

PROFESOR GUÍA CARDOSO, Lucia

FECHA Caracas, Febrero de 2.013

UCAB Universidad Católica ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA DE INFORMATICA

REINGENIERIA DEL PRODUCTO BANCA EN LINEA EMPRESAS (BOLE) DE MERCANTIL BANCO UNIVERSAL C.A. PARA OPTIMIZAR SU CALIDAD Y SERVICIOS

REALIZADO POR TORRES GARCÍA, Leonardo Alberto

TUTOR ACADEMICO CARDOSO, Lucia

FECHA Febrero de 2013

UCAB Universidad Católica ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

REINGENIERIA DEL PRODUCTO BANCA EN LINEA EMPRESAS (BOLE) DE MERCANTIL BANCO UNIVERSAL C.A. PARA OPTIMIZAR SU CALIDAD Y SERVICIOS

Este Jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado: DIECINUEVE (19) PUNTOS

J U R A D O E X A M I N A D O R

Firma: Roquefelix Benal
Nombre: ROQUEFELIX BENAL

Firma: Lucia P. Cardoso M.
Nombre: LUCIA ISABEL CARDOSO M.

Firma:
Nombre: Susana García

REALIZADO POR TORRES GARCÍA, Leonardo Alberto

TUTOR EMPRESARIAL BARRIENTOS, Jusbelis

TUTOR ACADÉMICO CARDOSO, Lucia

FECHA Febrero de 2013

Dedicatoria

A Mis Abuelas Mildred y Sofía

Agradecimientos

Mis más sinceros agradecimientos a mi tutora Lucia Cardoso por toda su ayuda incondicional y constante prestado durante el desarrollo de este trabajo, por todos los conocimientos aportados a lo largo de mi formación profesional y por su excelencia académica.

A mi mamá por estar a mi lado día a día, por soportarme y por enseñarme a vivir.

También quiero agradecer a mi hermano, ya que de no ser por su ayuda no hubiese terminado nunca este trabajo

Y por último, a mi papá quién me apoyó en mi crecimiento profesional.

Índice General

Dedicatoria i

Agradecimientos ii

Índice General iii

Índice de Tablas v

Índice de Figuras vi

Resumen viii

Introducción 1

Capítulo I. Planteamiento del Proyecto 3

I.1 Necesidades de la Empresa. 3

I.2 Solución Propuesta 4

I.3 Objetivo General. 4

I.3.1 Objetivos Específicos. 5

I.4 Alcance y Limitaciones 5

I.5 Justificación 8

Capítulo II. Marco Referencial. 9

II.1 Aplicación Web. 9

II.2 Banca en Línea Empresas. 10

II.3 Tutorial. 10

II.4 Calidad de Software. 12

II.5 Diseño de Soluciones Informáticas de Calidad. 14

Capítulo III. Metodología de Trabajo 22

Capítulo IV. Desarrollo 25

IV.1 Desarrollo del tutorial. 25

IV.2 Herramientas de Calidad de Código. 31

IV.3 Implementación de la Herramienta de Calidad de Código Seleccionada. 36

IV.4 Migración de Ayudas Mercantil 42

IV.5 Segundo Factor de Autenticación BMELE. 49

IV.6 Diseño de la nueva arquitectura. 52

IV.7 Implementación de la Arquitectura. 60

Capitulo V. Resultados 69

Capítulo VI. Conclusiones y Recomendaciones 71

Conclusiones. 71

Recomendaciones. 72

Bibliografía 73

Apéndice A 76

Apéndice B 88

Apéndice C 104

Apéndice D 107

Apéndice E 109

Apéndice F 132

Apéndice G 138

Apéndice H 154

Apéndice I 161

Índice de Tablas

Capítulo III.

Tabla 0 - 1. Metodología 24

Capítulo IV.

Tabla 2 - 1. Comparación de herramientas 34

Índice de Figuras

Capítulo II.

Figura 0 - 1. Arquitectura web. 16

Figura 0 - 2. Estructura Fachada. 19

Figura 0 - 3. Estructura Comando. 20

Figura 0 - 4. Estructura DAO. 21

Capítulo III.

Figura 0 - 5. Ilustración de XP 23

Capítulo IV.

Figura 1 - 1. Diagrama de Casos de Uso 27

Figura 1 - 2. DF: Tutoriales 28

Figura 1 - 3. Ventana principal tutoriales 29

Figura 1 - 4. Tutorial 30

Figura 1 - 5. Cuestionario 30

Figura 1 - 6. Mensaje de culminación de tutorial 31

Figura 3 - 1. Proyectos analizados en Sonar 37

Figura 3 - 2. Errores totales de la clase evaluada 38

Figura 3 - 3. Ejemplo error javadoc 39

Figura 3 - 4. Solución error javadoc 39

Figura 3 - 5. Ejemplo errores if/else 40

Figura 3 - 6. Solución error if/else 40

Figura 3 - 7. Ejemplo de error equals 40

Figura 3 - 8. Solución de error equals 40

Figura 3 - 9. Reporte final de errores 41

Figura 4 - 1. Botón de ayuda 42

Figura 4 - 2. Página de ayudas 42

Figura 4 - 3. Vínculo al servidor externo 43

Figura 4 - 4. Estructura del proyecto de ayudas 45

Figura 4 - 5. DF: Ayudas mercantil 46

Figura 4 - 6. Directorios Docroot 47
Figura 4 - 7. Código de referencia al servidor externo 47
Figura 4 - 8. Código de referencia al docroot 47
Figura 4 - 9. Página de ayuda desde docroot 48
Figura 5 - 1. Ventana de búsqueda token 50
Figura 5 - 2. Resultado de búsqueda 51
Figura 5 - 3. Página de administración de token 51
Figura 5 - 4. Detalle token 52
Figura 6 - 1. Arquitectura actual bole_mercantil 53
Figura 6 - 2. Métricas Sonar 54
Figura 6 - 3. Arquitectura nueva 55
Figura 6 - 4. ProcessPaging AdminToken 57
Figura 6 - 5. ProcessPaging AdminTokenOficina 57
Figura 6 - 6. Comando processsPaging 58
Figura 7 - 1. Distribución anterior 60
Figura 7 - 2. Helpers 62
Figura 7 - 3. Fachada de acceso a datos 63
Figura 7 - 4. Nueva distribución 63
Figura 7 - 5. Código original cargar consignación 64
Figura 7 - 6. Código final cargar consignación 65
Figura 7 - 7. Aceptar consignación principal 65
Figura 7 - 8. Método para procesar formulario 66
Figura 7 - 9. Método final para procesar formulario 66
Figura 7 - 10. Ventana de verificación de consignación 67
Figura 7 - 11. Método inicial aceptar consignación 67
Figura 7 - 12. Código final aceptar consignación 68

Resumen

REINGENIERIA DEL PRODUCTO BANCA EN LINEA EMPRESAS (BOLE) DE MERCANTIL BANCO UNIVERSAL C.A. PARA OPTIMIZAR SU CALIDAD Y SERVICIOS

Torres García, Leonardo
torres.leo16@gmail.com

Banca en Línea Empresas es un producto que ofrece Mercantil Banco Universal C.A. a sus clientes para facilitar la gestión administrativa de sus empresas . Fue desarrollado hace mas de 10 años, y actualmente se encuentra desactualizado, presentando algunos problemas en su mantenimiento consecuencia de una serie de malas prácticas que se han utilizado a lo largo de los años en su desarrollo.

Este Trabajo Instrumental de Grado tiene como base 5 puntos clave: Desarrollar un tutorial para la instalación del producto, investigar, instalar y aplicar herramientas de calidad de código para eliminar errores del producto, desarrollar nuevos servicios para la plataforma y construir una nueva arquitectura para la aplicación que permita eliminar los grandes problemas que hoy en día enfrenta para poder seguir brindándole a sus clientes el servicio de alta calidad con el que siempre han contado.

Para la ejecución del proyecto se aplicó la metodología ágil Extreme Programing que permitió definir diferentes iteraciones de cada proyecto que iban generando versiones para la revisión de los clientes y la inclusión de modificaciones.

PALABRAS CLAVE: Desarrollo de Software, Banca Electrónica, Arquitectura de Software.

Introducción

Este Trabajo Instrumental de Grado está conformado por cinco (5) proyectos: el primero consiste en la elaboración de una aplicación web cuya función principal es la de proveer a los desarrolladores de la Banca en Línea Empresas de un tutorial que les permita instalar el producto en un ambiente local.

El segundo proyecto consiste en investigar, configurar e instalar herramientas que permitan mejorar la calidad del producto en al menos un 25%. Para demostrar la funcionalidad de la herramienta seleccionada se reducirán algunos errores que sean detectados.

El tercer proyecto consiste en desarrollar una solución que permita disociar a la Banca en Línea Empresas del servidor externo que contiene las páginas de ayuda que utiliza la aplicación para brindarle apoyo a sus usuarios.

Un cuarto proyecto consiste en desarrollar un nuevo servicio para el módulo de bancos de la Banca en Línea Empresas. El servicio consiste en generar un módulo administrativo que permita realizar diferentes operaciones sobre los dispositivos que se utilizarán para incluir un segundo factor de autenticación para ingresar al portal.

Finalmente se evaluará la arquitectura actual del producto para diseñar una nueva que implemente patrones de diseño para corregir los fuertes problemas que han surgido con el tiempo. Entres estos problemas destacan una gran cantidad de código duplicado y errores de código fuente que generan grandes conflictos en el mantenimiento de la aplicación.

En el capítulo I se encuentra una breve descripción del trabajo en forma introductoria, incluyendo las necesidades de la empresa, la solución propuesta, los objetivos, la justificación, los alcances y limitaciones.

El capítulo II está comprendido totalmente por el Marco Referencial, el cual incluye los conceptos e investigaciones que se llevaron a cabo en la fase inicial de este proyecto. A través de esta actividad se edificó una base para sustentar todo el desarrollo del trabajo instrumental.

Posteriormente en el capítulo III, se encuentra la Metodología del Trabajo en el cual se explica las estrategias utilizadas para llevar a cabo el desarrollo del proyecto.

En el capítulo IV se explica el proceso de construcción de cada uno de los proyectos de este Trabajo Instrumental.

Por otra parte, en el Capítulo V se exponen los resultados obtenidos al culminar este trabajo y por último en el capítulo VI se presentan las conclusiones del Trabajo Instrumental de Grado, de igual forma se proponen recomendaciones a fin de complementar el trabajo realizado.

Capítulo I. Planteamiento del Proyecto.

I.1 Necesidades de la Empresa.

Banca en Línea Empresas (BOLE) es un producto que ofrece Mercantil Banco Universal C.A. a todas las empresas que poseen por lo menos una cuenta en este Banco. BOLE Mercantil es un sistema que, mediante el uso de tecnologías web, permite a sus usuarios corporativos y jurídicos realizar gran cantidad de operaciones financieras y no financieras de manera fácil, segura y rápida; como por ejemplo, el pago de nómina de una empresa afiliada o consultar sus estados de cuenta.

Este sistema fue desarrollado hace muchos años y actualmente se encuentra en proceso de migración a una plataforma más actualizada. Es por esto, que surge la oportunidad de realizar un proceso de reingeniería del software que permita detectar los defectos actuales del sistema y replantearlo con el fin de mejorar la calidad, el rendimiento y los servicios que este ofrece.

La documentación del producto es escasa y la existente se encuentra desactualizada, presentando muchas inconsistencias que causan problemas al momento de instalar el producto para desarrollar nuevas funcionalidades. Este proceso, que podría tardar dos días en ejecutarse, suele tomar entre tres y cuatro semanas en realizarse debido al inconveniente explicado anteriormente.

Otra oportunidad que surge es detectar y corregir diferentes errores del código fuente que ocurren debido a la aplicación de malas prácticas que se han efectuado a lo largo de los años. Esta situación deterioraba la calidad del producto, por lo fue posible mejorarlo significativamente en este aspecto aplicando medidas correctivas.

Finalmente, se ofrece la oportunidad de desarrollar nuevos requerimientos que permitieron mejorar y ampliar los servicios que ofrece BOLE a sus usuarios con el

fin de seguir creciendo y brindando un servicio estable, seguro y confiable a quienes lo utilizan.

I.2 Solución Propuesta

Se propone realizar un proceso de reingeniería del software, evaluando las nuevas prácticas de desarrollo que existen en el mercado, como lo son los patrones de diseño. Estos pueden implementarse creando una nueva arquitectura que permita mejorar el rendimiento del producto y su calidad.

Se desarrollarán nuevos requerimientos para el producto, y uno de ellos implementará los patrones de diseño estudiados y seleccionados previamente para validar y corroborar su efectividad al momento de mejorar el rendimiento del sistema.

De igual forma, se establecerán distintas métricas utilizando diferentes herramientas de control de calidad de software sobre el código fuente de BOLE Mercantil, con la finalidad de establecer un nivel de calidad que sea aceptable por la institución y alcanzarlo.

Se plantea desarrollar un tutorial que guíe a los desarrolladores durante el proceso de instalación del producto sin la necesidad del seguimiento permanente de los técnicos especializados en la materia, mejorando el tiempo de instalación del producto.

Todas las actividades realizadas serán correctamente documentadas para el uso y respaldo de la institución.

I.3 Objetivo General.

Desarrollar la reingeniería del producto Banca En Línea Empresas (BOLE) de Mercantil Banco Universal C.A. para optimizar su calidad y servicios.

I.3.1 Objetivos Específicos.

- Diseñar, desarrollar e implementar un tutorial que facilite el proceso de instalación de BOLE Mercantil.
- Evaluar y seleccionar herramientas que permitan disminuir los errores de código fuente del producto.
- Configurar la herramienta seleccionada para disminuir los errores de código fuente del producto.
- Diseñar y desarrollar una aplicación web que permita integrar el sistema de ayudas de la Banca en Línea Empresas a BOLE Mercantil.
- Desarrollar un segundo factor de autenticación a BOLE Mercantil en la parte del Módulo de Bancos.
- Diseñar una arquitectura nueva para BOLE Mercantil aplicando patrones de diseño y mejores prácticas de la ingeniería del software.
- Desarrollar un requerimiento del sistema aplicando la nueva arquitectura.

I.4 Alcance y Limitaciones

El equipo de Calidad de Software de la institución es el departamento encargado de establecer las métricas de calidad y desarrollo de software; así como también, de establecer valores cuantitativos que se presentan por medio de reportes a los desarrolladores. Estos reportes permiten determinar si una pieza de código (clase, método, entre otros) cumple con el estándar aceptado por la compañía. Para generar estos reportes utilizan diferentes herramientas y aplicativos que existen en el mercado con el propósito de cumplir este objetivo.

El tutorial a desarrollar debe permitir la instalación del producto en un período máximo de tres días con un apoyo mínimo por parte de los técnicos especializados en la aplicación. Dicho tutorial será una aplicación web que presenta paso a paso el proceso de instalación de cada uno de los programas que componen el

sistema BOLE, y ayuda al desarrollador a aprender del sistema al mismo tiempo que lo instala y configura.

Las herramientas a evaluar, seleccionar y configurar permiten reducir los errores de código fuente del producto en al menos un 25%, ya que esto representa una mejora significativa a la calidad actual del producto. Estos errores se definen como todo código desarrollado aplicando malas prácticas de programación, o código que genera errores o "bugs" en el aplicativo, que pudieron optimizarse utilizando mejores prácticas de programación o aplicando una refactorización al código fuente. Esta medida también es determinada por el equipo de calidad de software de la empresa y se ve reflejada en el reporte que ellos generan, actualmente existen 2.813 errores de este tipo. Se corrigieron 200 errores para corroborar el funcionamiento de las herramientas utilizadas.

Se diseñará y desarrollará una aplicación web que permite integrar las páginas de ayuda de BOLE Mercantil al proyecto principal, dichas ayudas se encuentran actualmente en un servidor externo y se llaman por medio de hipervínculos desde BOLE Mercantil. Esta aplicación tiene como objetivo desarrollar nuevamente estas páginas de ayuda en un formato mucho más manejable que el actual, implementando nuevas técnicas de desarrollo que garanticen la escalabilidad del sistema y su fácil mantenimiento.

Se desarrollará un segundo factor de autenticación a BOLE Mercantil para el módulo de bancos del sistema. El módulo de bancos es una aplicación web que, a diferencia del módulo transaccional (MELE), es utilizado por los funcionarios del banco para desempeñar diferentes operaciones. Se requiere elaborar una aplicación web integrada al módulo de bancos (BMELE) que permita realizar las operaciones administrativas que son necesarias para incluir este factor.

Adicionalmente, es necesario desarrollar el módulo de auditorías para el módulo de banco que consiste en llevar un control de todas las operaciones realizadas por un funcionario dentro de BMELE para poder realizar las consultas necesarias al momento de que ocurra un proceso de auditoría.

El proyecto de incluir el segundo factor de autenticación debe integrar nuevas funcionalidades a los módulos de administración de grupos, usuarios y empresas que permitan manejar a los usuarios que posean este nuevo mecanismo de autenticación que sustituirá el de certificados digitales.

Finalmente, se realizarán ajustes al proyecto de certificados digitales para reconocer a los usuarios que posean el segundo factor de autenticación de manera que queden inhabilitados para usar estos servicios.

Se replanteará la arquitectura del producto BOLE Mercantil, aplicando patrones de diseño del software que permitan mejorar el rendimiento y la calidad del software. Esta arquitectura se apoyará en las virtudes y defectos que presenta la actualmente implementada para garantizar una mejora general en el código del producto. Se considera la mejora del código si es más legible, reusable, escalable y fácil de mantener.

Para demostrar el funcionamiento de la arquitectura planteada se desarrollará un requerimiento que aplique la nueva arquitectura. Este podrá ser comparado con el código actual para determinar si efectivamente se consiguieron las mejoras que se plantearon anteriormente.

I.5 Justificación

La realización de este trabajo requiere de la aplicación de todos los conocimientos sobre las áreas de ingeniería del software, base de datos, metodología del software y desarrollo del software, lo que permite obtener un crecimiento a nivel profesional significativo para un ingeniero informático.

El desarrollo de una aplicación web que sirve como tutorial para la instalación de la Banca en Línea Empresas permitió también realizar un aporte de gran valor para la organización, ya que se tomó esta oportunidad para brindarle a los coordinadores herramientas administrativas que les permitirán gestionar y monitorear el crecimiento de sus desarrolladores.

La evaluación de las herramientas de calidad de código realizó un aporte de gran impacto a la organización, ya que se introdujo una nueva forma de evaluar los proyectos de la Banca en Línea Empresas y se unificó el trabajo de los equipos de desarrollo, arquitectura y calidad de software.

La arquitectura desarrollada durante este trabajo aportó una solución confiable al mayor problema que afronta la plataforma actualmente, como es el proceso de mantenimiento del mismo que, debido a la gran cantidad de código duplicado y malas prácticas aplicadas, han afectado de forma negativa a la calidad del producto

Finalmente, se presenta la oportunidad de realizar nuevos servicios para la Banca en Línea Empresas donde se deben aplicar todos los conocimientos obtenidos en la carrera para crear productos de calidad que cumplan con las expectativas de los usuarios.

Capítulo II. Marco Referencial.

En este capítulo se sustentan teóricamente los tópicos del Trabajo Instrumental de Grado.

II.1 Aplicación Web[MICROSOFT01].

Para comprender el ambiente en el que se desenvuelve este Trabajo Instrumental de Grado es necesario conocer ante nada que es una aplicación web, este concepto se define de la siguiente forma:

"Es una aplicación que puede ser accedida por los usuario a través de un navegador web o un agente de usuario especializado. El núcleo de una aplicación web es la lógica del servidor. La aplicación puede contener varias capas distintas. El ejemplo típico es una arquitectura de tres capas compuesta de presentación, los negocios y las capas de datos."

Estas herramientas dan pié a una infinidad de posibilidades, gracias a ellas los usuarios pueden realizar diferentes operaciones desde sus hogares por medio de la internet, actualmente es posible comprar productos de cualquier parte del mundo utilizando alguna aplicación web como por ejemplo el servicio que ofrece Amazon.com

La banca mundial también ha dispuesto de estas herramientas para permitirle a sus usuarios realizar una gran cantidad de operaciones como consultar saldos, pagar tarjetas, transferir a otras cuentas, entre otras, desde la comodidad del hogar y de forma sencilla, segura y rápida.

II.2 Banca en Línea Empresas[MERCANTIL01]

La Banca en Línea Empresas (BOLE) se define como un servicio de banca electrónica seguro y confiable, que facilita la gestión administrativa de las empresas y maximiza su eficiencia, con sólo acceder a su página web.

BOLE está compuesto por dos grandes módulos: El transaccional o MELE, que es el portal al que acceden los usuarios de Mercantil Banco Universal para realizar sus transacciones y el módulo de bancos o BMELE, que es el área administrativa que utilizan los funcionarios del banco para mantener, monitorear y dar soporte a los usuarios de MELE en todo momento.

Algunas de las ventajas que ofrece BOLE a sus usuarios son las siguientes:

- ✓ Facilita y simplifica la gestión administrativa
- ✓ Ofrece altos niveles de seguridad
- ✓ Amplia la gama de servicios acorde con las necesidades del mundo empresarial, disponible las 24 horas del día, los 7 días de la semana.
- ✓ Histórico de las transacciones realizadas por cada uno de los usuarios de las empresas

Otra ventaja que introdujo la Banca en Línea es el sistema de notificaciones. Cada vez que un usuario realiza una transacción, o el Banco tiene la necesidad de hacer llegar una información a los clientes se utiliza la Banca en Línea como medio de información, a través de campañas personalizadas que se despliegan en la página principal de un usuario y correos electrónicos que reciben de forma oportuna.

II.3 Tutorial[DEFINICION01]

Un tutorial "se trata de un curso breve y de escasa profundidad que enseña los fundamentos principales para poder utilizar algún tipo de producto o sistema, o para

poder realizar una cierta tarea" en la informática son "pequeñas guías o compilados de de instrucciones, destinadas a los usuarios que no tienen la capacidad de manejo necesaria para utilizar un producto o servicio"

Los tutoriales son bastante prácticos ya que no sólo sirven para instalar cierto producto, también dejan un aprendizaje en las personas que los utilizan y los capacitan para realizar las actividades para las que fueron diseñados. Cuando una persona no tiene idea de cómo realizar un objetivo puede pasar semanas intentando cumplirlo sin obtener resultados, pero con la ayuda de un tutorial bien elaborado esta actividad puede reducirse a unos pocos día o incluso a unas pocas horas.

Son un método también de transmisión de la información, y no es necesaria la participación de un segundo para su ejecución ya que lo ideal es que el usuario que utilice la herramienta pueda ejecutar la actividad siguiendo cada uno de los pasos establecidos en un orden lógico.

Es interesante que un tutorial puede crearse en la forma de una aplicación web, de modo que, cualquier persona con acceso a la red podría ingresar a su contenido e interactuar con esta aplicación para aprender y capacitarse para ejecutar una actividad. Tomando como referencia la definición desarrollada anteriormente se pueden definir que un tutorial debe poseer las siguientes funcionalidades:

- ✓ Páginas simples que expliquen paso a paso como llevar a cabo una operación.
- ✓ Transiciones entre cada uno de los pasos presentados.
- ✓ Exámenes de refuerzo para reforzar la comprensión del contenido.

II.4 Calidad de Software[FERNANDEZ01][ISO01]

La calidad del software es un tema de gran peso sobre cualquier producto de software. Se define como un "conjunto de cualidades que caracterizan al software y determinan su utilidad y existencia."

La ISO/IEC 9126-1 establece que todo software de calidad debe poseer las siguientes características:

- ✓ **Funcionalidad:** Es un conjunto de atributos que influyen en la existencia de un grupo de funciones y sus propiedades especificadas.
- ✓ **Fiabilidad:** Es un conjunto de atributos que influyen en la capacidad del software para mantener su nivel de desempeño por un período determinado de tiempo en unas condiciones específicas.
- ✓ **Usabilidad:** Es un conjunto de atributos que influyen en el esfuerzo necesario para el uso del software, y en la valoración individual de tal uso por un grupo de usuarios.
- ✓ **Eficiencia:** Es un conjunto de atributos que influyen en la relación entre el nivel de rendimiento del software y la cantidad de recursos utilizados por el mismo.
- ✓ **Mantenibilidad:** Es un conjunto de atributos que influyen en el esfuerzo necesario para realizar modificaciones al software.
- ✓ **Portabilidad:** Es un conjunto de atributos que influyen en la capacidad del software para ser transferido de un ambiente a otro.

Es posible medir la calidad de software utilizando diferentes técnicas y herramientas que sirven para este propósito, aunque cada software necesita diferentes niveles de calidad. Un software que se utiliza para realizar experimentos científicos avanzados debe ser mucho más preciso y aprueba de fallos que otro cuya finalidad es simplemente entretener a una persona durante un corto período de tiempo.

Para obtener un software de calidad es necesario planificarlo desde su inicio y aplicar metodologías para el análisis, diseño, programación y prueba del mismo que permitan uniformar la forma de trabajo con la finalidad de lograr una mayor fiabilidad, mantenibilidad y facilidad para realizar pruebas. La planificación de buenas políticas contribuyen a lograr la calidad del software pero no pueden garantizarlo, para poder asegurarlo es necesario controlar y evaluar el software.

El control de calidad de software se efectúa definiendo indicadores, criterios y parámetros de medición.. Con el transcurrir de los años han surgido una gran cantidad de herramientas cuya función es automatizar el proceso de control de calidad de software. Dichas herramientas utilizan una serie de reglas de programación y métricas para determinar el nivel de calidad que posee un software en un momento dado. [GARZAS01]

Estas herramientas realizan un análisis del código fuente de un proyecto para detectar los errores que poseen, el resultado de este análisis es presentado a los usuarios a través de reportes que presentan los valores de las métricas implementadas en el análisis y las violaciones a las reglas de programación establecidas, con la finalidad de que sean corregidos para obtener un código de mayor calidad.

Existen 7 ejes de la calidad de código: [SONAR01]

Arquitectura y diseño[MICROSOFT02]**:** Es la definición de una solución estructurada que cumple con todos los requisitos técnicos y funcionales de un sistema, optimizando al mismo tiempo los atributos de calidad comunes, tales como el rendimiento, la seguridad y la mantenibilidad.

Duplicación del código[SPINELLIS01]**:** Son porciones de código que se repiten varias veces en un proyecto. La duplicación de código indica que hay una falla en la utilización de mecanismos de abstracción en el programa, como métodos y funciones, por lo que el código es más largo de lo que debería ser.

Pruebas unitarias[MICROSOFT03]**:** Consisten en tomar la pieza más pequeña de software que puede ser ejecutada al aislarla del resto del código para verificar su funcionalidad. Este proceso ayuda a corregir Bugs potenciales a tiempo y facilita la mantenibilidad del sistema, ya que si la prueba unitaria se realizo correctamente se puede filtrar el motivo que ocasionó un error con mayor facilidad.

Complejidad del código[CHARNEY01]**:** Tiene que ver directamente con la mantenibilidad de un sistema. Se refiere a la cantidad de esfuerzo necesario para comprender una pieza de software y modificarla correctamente.

Bugs potenciales[PCMAG01]**:** Son posibles errores que pueden ocasionar que el sistema funcione de manera incorrecta, pueden ocasionarse por los desarrolladores al cometer algún error en la escritura del código fuente, o al momento de diseñar la aplicación.

Reglas de programación[VALID01]**:**Se define como un conjunto de reglas y guías que sirven para dar un formato estandarizado al código fuente de una aplicación, ayudando a facilitar la comprensión del mismo.

Comentarios[UTAH01]**:** Consiste en colocar descripciones legibles para los seres humanos que explican lo que una pieza de código debe hacer. El objetivo de los comentarios es facilitar el mantenimiento de las aplicaciones y la búsqueda de Bugs dentro de la aplicación.

El análisis del código de una aplicación utilizando estas siete características permiten generan código de calidad.

II.5 Diseño de Soluciones Informáticas de Calidad

II.5.1 Arquitectura del Software[MICROSOFT02]

Como se indica en el apartado anterior, todo software de calidad debe ser planificado desde sus inicios, es por esto que la arquitectura seleccionada es de suma importancia para desarrollar una solución informática de calidad. La arquitectura del software se define de la siguiente forma:

"Es el proceso de definición de una solución estructurada que cumple con todos los requisitos técnicos y de funcionamiento, optimizando al mismo tiempo los atributos de calidad comunes, tales como el rendimiento, la seguridad y capacidad de administración. Implica una serie de decisiones basadas en una amplia gama de factores, y cada una de estas decisiones pueden tener un impacto considerable sobre la calidad, el rendimiento, mantenimiento, y el éxito general de la aplicación."

La importancia de la arquitectura del software "al igual que cualquier otra estructura compleja, el software debe ser construido sobre una base sólida. No considerar escenarios clave, los problemas comunes, o las consecuencias a largo plazo de las decisiones clave pueden poner en riesgo la aplicación. Una arquitectura mal diseñada produce riesgos como la creación software inestable, incapacidad de soportar requerimientos de negocio actuales o futuros, o dificultad para implementar o gestionar en un entorno de producción".

II.5.1.1 Arquitectura de tres capas[MICROSOFT04].

Es una arquitectura que consiste en segmentar o dividir los componentes de una aplicación en tres niveles, cada nivel representa un servicio. Los niveles no necesariamente deben corresponderse con ubicaciones físicas, sino mas bien corresponde a una división lógica de la aplicación.

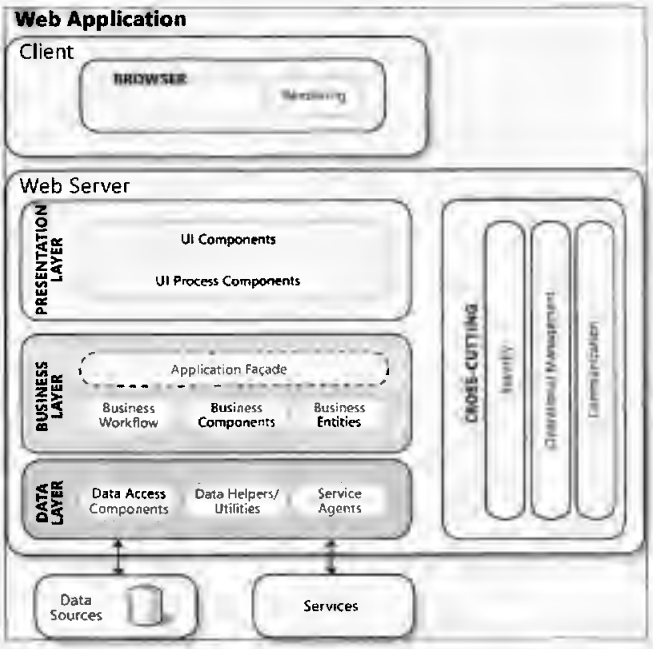


Figura 0 - 1: Arquitectura web - Fuente: [MICROSOFT01]

A cada nivel se le asocian los siguientes servicios:

- ✓ Presentación o capa de servicios de usuario: proporciona un acceso de usuario a la aplicación.
- ✓ Negocios: consiste en reglas de negocio y los datos.
- ✓ Datos: Interactúa con los datos persistentes usualmente almacenados en una base de datos.

La implementación de esta arquitectura tiene varias ventajas entre las que destacan las siguientes:

- ✓ Permite compartir y reutilizar los componentes desarrollados.
- ✓ Dividir proyectos grandes y complejos en otros más simples y asignarlos a diferentes desarrolladores.

- ✓ Permite implementar componentes y servicios en el servidor para mantenerlos al día con los cambios, y volver a implementarlos a medida que vaya creciendo el sistema.

II.5.2 Patrones de Diseño[SOURCE01]

"En ingeniería del software, un patrón de diseño es una solución general repetible a un problema que ocurre comúnmente en el área de diseño de software. El patrón de diseño no es un diseño finalizado que puede ser transformado inmediatamente a código. Es una descripción o plantilla de cómo se debe resolver un problema que puede ser utilizada en diferentes situaciones".

Los patrones de diseño tienen varios usos, son capaces de acelerar los tiempos de desarrollo de un software, ya que proveen paradigmas de desarrollo probados. Ayudan a prevenir errores en la etapa de diseño de la aplicación que puede que no sean evidentes en un principio y pueden causar problemas mayores en etapas futuras del desarrollo. También sirven para mejorar la legibilidad del código, en especial para desarrolladores que estén familiarizados con estas técnicas.

Ayuda también a los desarrolladores a comprender de una mejor manera el código, ya que muchas veces es complicado saber cómo aplicar una solución existente a otro problema similar, los patrones de diseño proveen soluciones generalizadas que están documentadas de tal forma que no están atados a una solución en específico sino que pueden atacar un gran conjunto de problemas.

También ayudan a la comunicación entre desarrolladores, muchos pueden facilitar el trabajo en equipo, y todos pueden mejorar con el tiempo para convertirse en soluciones más robustas. Los patrones de diseño se pueden clasificar de la siguiente forma:

- ✓ Creacionales: Son todos aquellos que tratan problemas comunes de instanciación de clases.
- ✓ Estructurales: Son los que tratan problemas de composición de los objetos y las clases.
- ✓ De comportamiento: Son los que tratan problemas de comunicación de clases y objetos.

Los patrones de diseño utilizados en este trabajo instrumental de grado son los siguientes:

II.5.2.1 Fachada:

Es un patrón de diseño estructural que se utiliza cuando se necesita acceder a un subsistema muy complejo. La fachada define una interfaz unificada de acceso a dicho subsistema para simplificar su complejidad, permitiéndole al usuario abstraerse de dicha complejidad al interactuar con la fachada.

La fachada encapsula el comportamiento de un subsistema más complejo en una única interfaz, reduciendo la curva de aprendizaje que se necesita para usar dicho subsistema, también promueve la disociación del subsistema y sus clientes.

En la figura 2 se puede observar un ejemplo que presenta la estructura de la fachada.

Aplicar este patrón de diseño tiene grandes ventajas, ya que facilita el trabajo en una arquitectura basada en capas, al definir una fachada como punto de acceso a cada una de las capas se logra abstraer a otras capas del funcionamiento de la capa con la que desean comunicarse. Como consecuencia de esto se facilita el trabajo en equipo, debido a que diferentes desarrolladores pueden estar trabajando en diferentes capas de la aplicación independientemente, y lo único que necesitarían conocer de las otras es la fachada que encapsula su funcionamiento.

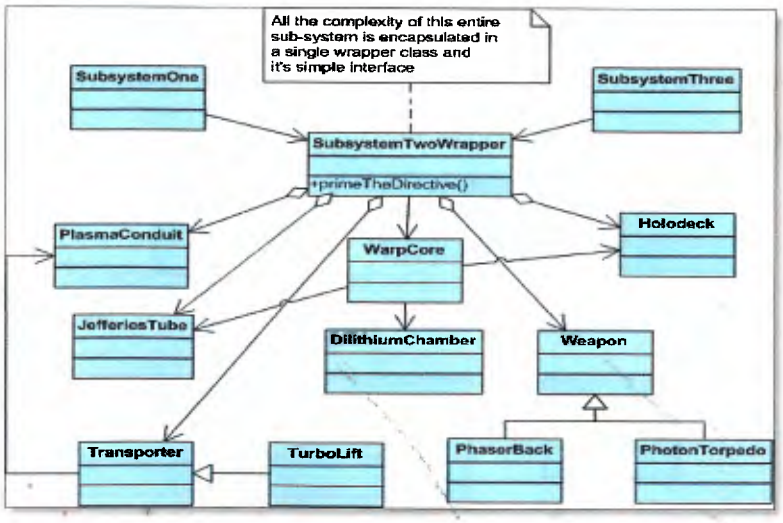


Figura 0 - 2: Estructura Fachada - Fuente: [SOURCE01]

II.5.2.2 Comando:

Es un patrón de diseño de comportamiento que se utiliza cuando es necesario realizar una operación haciendo transparente el proceso que se debe ejecutar para la entidad que la solicita. El comando encapsula métodos o peticiones en objetos permitiendo realizar operaciones como el hacer (do) y el deshacer (undo).

Este patrón disocia al objeto que invoca una operación del que la ejecuta, realizando una especie de proceso de caja negra, para lograr esto se crea una clase abstracta con un método "ejecutar()" que se encarga de realizar la operación solicitada.

En la figura 3 se aprecia la estructura de este patrón de diseño:

Aplicar este patrón de diseño tiene varias ventajas, una de ellas es la portabilidad y reusabilidad del código, ya que si, por ejemplo, se logra encapsular el proceso de inicio de sesión y se llegase a necesitar en otro proyecto, basta con

transportar el comando de un proyecto a otro y es posible que sin modificaciones, o modificaciones muy mínimas, continúe funcionando de la misma forma.

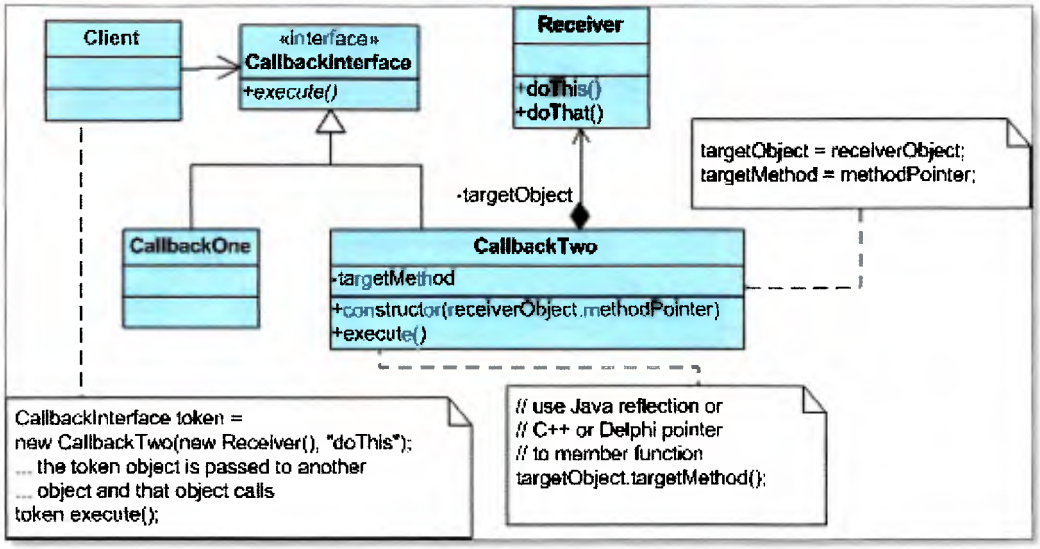


Figura 0 - 3: Estructura Comando - Fuente: [SOURCE01]

Otra ventaja es que reduce la duplicación de código al encapsular métodos que pueden ser utilizados por diferentes clases, en lugar de repetir el código que ejecuta una operación común ambas clases simplemente ejecutarán el mismo método.

Otra de las ventajas que ofrece es que facilita el proceso de mantenimiento del sistema. De ser necesario modificar una operación que se encuentra encapsulada un comando, basta con modificar dicho comando para que todas las clases que lo utilizan posean el cambio.

II.5.2.3 Objeto de Acceso a Datos (DAO)[ORACLE01]:

Es un patrón de diseño de comportamiento que se utiliza cuando es necesario acceder a la información que se encuentra en algún mecanismo de persistencia de datos, haciendo transparente el proceso que se debe ejecutar para la entidad que la solicita. Los DAO encapsulan el proceso de acceso a datos.

Este patrón disocia al objeto accede a los mecanismos de persistencia del que solicita la información, realizando un proceso de caja negra. El DAO implementa los métodos necesarios para acceder a una fuente de datos, mapea la información y devuelve los objetos solicitados por otras clases.

La estructura de este patrón de diseño es la siguiente:

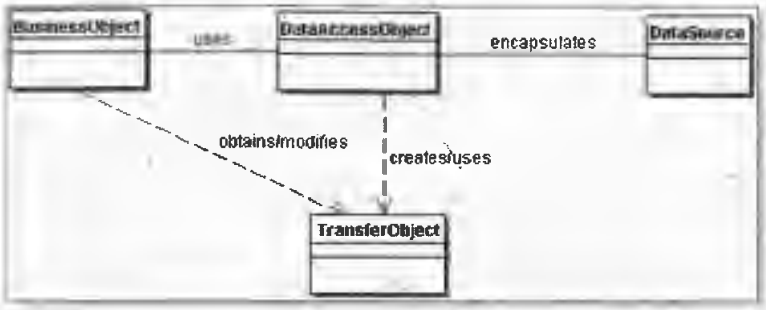


Figura 0 - 4: Estructura DAO - Fuente: [ORACLE01]

Este patrón de diseño permite que el proceso de acceso a datos sea transparente para los otros objetos de lógica de negocios, ya que les permite acceder a la fuente de datos sin saber exactamente su implementación. Hace fácil la migración de una base de datos a otra debido a que la única capa que debería cambiarse es la del DAO.

También ayuda a reducir la complejidad del código de las clases de lógica de negocios, fomenta el trabajo en capas y centraliza toda la lógica de acceso a datos en una de ellas.

Capítulo III. Metodología de Trabajo[WELLS01]

La metodología empleada para el desarrollo de este proyecto está basada en una de las técnicas ágiles conocidas como Extreme Programming (XP). Para el empleo de esta estrategia se plantean ciertos objetivos o "checkpoints" definidos de la siguiente manera:

- **Desarrollo del tutorial.**
- **Corrección de errores de código fuente.**
- **Desarrollo del proyecto de ayudas.**
- **Desarrollo del segundo factor de autenticación Módulo de Bancos.**
- **Desarrollo e implementación de la arquitectura.**

Para cada uno de los objetivos es necesario aplicar las cuatro (4) actividades o fases que se describen a continuación:

- **Escuchar:** Es una actividad que lleva a cabo el equipo de diseño del software de la empresa en la que se elaboran las historias de usuario y se levantan los requerimientos del sistema a realizar siguiendo la regla de planificación que establece la metodología.
- **Diseñar:** El equipo de diseño elabora la solución a los requerimientos obtenidos en la actividad anterior y realiza reuniones con el equipo de desarrollo para hacer entrega de los documentos necesarios para la comprensión del proyecto que se va a realizar.
- **Codificar:** Es la actividad que implica desarrollar el código de la funcionalidad que se desea implementar.
- **Probar:** Consiste en comprobar o verificar que la solución elaborada efectivamente satisface las condiciones necesarias para

afirmar que se ha cumplido el objetivo. Durante esta actividad el equipo de Calidad de Software de la institución realiza pruebas extensivas a la solución desarrollada.

En la figura 0 - 5 se ilustra esta metodología:

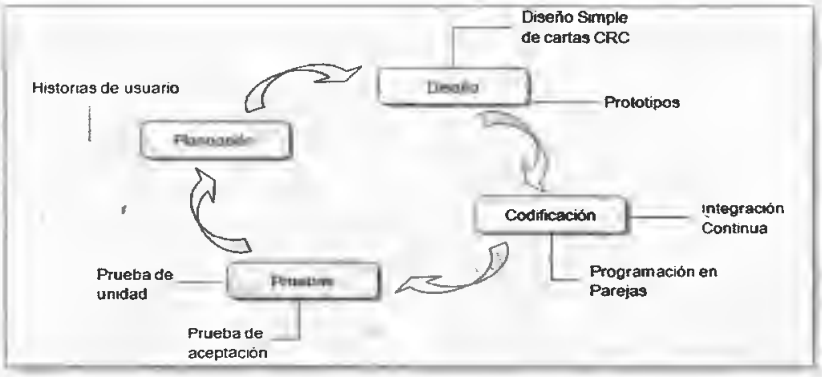


Figura 0 - 5: Ilustración de XP - Fuente: scielo.br

Justificación

La aplicación de esta metodología se realiza para poder desarrollar nuevas funcionalidades sobre un sistema que se encuentra en producción sin afectarlo directamente. Cuando un proyecto ha ejecutado cada una de las actividades definidas anteriormente se corrobora que, efectivamente, está preparado para entrar en el ambiente de producción.

Adicionalmente tiene la ventaja de permitir entregar el software que el cliente necesita cuando lo necesita, y está diseñado para responder y adaptarse fácilmente a los cambios de alcance aún cuando el proyecto se encuentre en las etapas finales de su ciclo de vida, lo que permite que los productos desarrollados bajo esta metodología sean escalables. También está pensada para ser implementada por grupos de trabajo pequeños, idealmente en parejas.

Implementación

Es importante mencionar que no todos los entregables son parte de este proyecto, las historias de usuario, algunos de los documentos de diseño y los reportes de QA son confidenciales para la empresa. En el apéndice H se encuentra el cronograma de actividades,

En el cuadro M - 1 se puede observar cómo se implementó cada actividad durante la ejecución de cada checkpoint. Sólo están contempladas las actividades realizadas durante el desarrollo de este Trabajo Instrumental de Grado.

Checkpoint	Fase	Actividad	Entregable
Tutorial	Escuchar	Reunión con el personal del banco y la coordinación de BOLE	Casos de uso, requerimientos
	Diseñar	Elaboración del documento de diseño y plan del proyecto	Doc. De diseño
	Codificar	Elaboración de la aplicación web	Aplicación web del tutorial
Corrección de errores	Escuchar	Reunión con los equipos de calidad y arquitectura de la empresa	Casos de uso, requerimientos
	Diseñar	Investigación, comparación y selección de herramientas	La herramienta de calidad de código
	Codificar	Corrección de los errores de código fuente	Los errores corregidos
Ayudas	Escuchar	Reunión con el equipo de mercadeo y personal del banco	Casos de uso, requerimientos
	Diseñar	Elaboración del documento de diseño y plan del proyecto	Doc. De diseño
	Codificar	Elaboración de la aplicación	Aplicación web de las ayudas
Segundo Factor BMELE	Codificar	Elaboración de los módulos administrativos y de auditoría	Módulo administrativo de token y auditorías
Arquitectura	Escuchar	Reunión con la coordinación de BOLE	Requerimientos
	Diseñar	Diseño y desarrollo de la arquitectura	La nueva arquitectura de BOLE
	Codificar	Implementación de un requerimiento sobre la arquitectura	Arquitectura implementada por un requerimiento

Tabla 0 - 1: Metodología - Fuente: Elaboración propia

Capítulo IV. Desarrollo

En esta sección se describen, en forma detallada, todos los procedimientos llevados a cabo durante la elaboración de este trabajo de investigación, para cumplir con los objetivos propuestos.

IV.1 Desarrollo del tutorial

La Banca en Línea Empresas es un producto que lleva varios años en el mercado, con el transcurso del tiempo la aplicación ha ido evolucionando, incorporando nuevas funcionalidades que permiten extender los servicios que se ofrecen por medio de este portal. Este crecimiento también implica que la complejidad del proyecto ha ido creciendo progresivamente, lo que implica para los desarrolladores la necesidad de un nivel de instrucción para poder desarrollar o modificar las funcionalidades que actualmente ofrece el producto.

Es por esto que surge la necesidad de desarrollar una aplicación web que guíe y ayude a los desarrolladores a comprender el producto con el que van a trabajar, esta aplicación presentará diferentes tutoriales que tienen como propósito permitir el desempeño de diferentes tareas a través de cursos breves que expliquen. paso a paso, que se debe hacer para lograr un objetivo. Inicialmente la aplicación está destinada a enseñar a los desarrolladores nuevos a instalar el producto BOLE Mercantil en sus ambientes de trabajo local.

IV.1.1 Requerimientos

La aplicación web a desarrollar debía cumplir las siguientes funcionalidades:

a) Clasificar usuarios según su cargo, b) permitir a los usuarios registrados acceder a los tutoriales de la plataforma con la que trabajan, c) llevar un registro de

los tutoriales culminados y d) permitir consultar el avance de los usuarios a través de los tutoriales

Los requerimientos no funcionales especificados fueron los siguientes:

- Autenticar usuarios para brindarles acceso a los servicios del aplicativo.

- El sistema debe ser escalable, a futuro debe ser posible incluir nuevas plataformas y tutoriales

- El sistema debe adaptarse y conectarse con diferentes bases de datos

IV.1.2 Caso de Uso

En base a los requerimientos explicados anteriormente se diseñó la aplicación para satisfacer esos objetivos y mucho más. Para comprender el diagrama de casos de uso que se puede analizar en la figura 1 - 1 es necesario definir los dos tipos de usuarios que pueden ingresar al sistema:

- Desarrollador: Es un usuario autenticado que solo tiene acceso a su información personal y a los tutoriales

- Coordinador: Es un usuario autenticado que, además de tener acceso a su información personal y a los tutoriales, posee una serie de herramientas administrativas para manejar los desarrolladores de su departamento.

Podría considerarse que existe un tercer tipo de usuario que consistiría en una persona que ingresa al sistema y no inicia ninguna sesión, en este caso la persona únicamente podría ingresar a las opciones de Inicio de sesión y a la pantalla de inicio del portal.

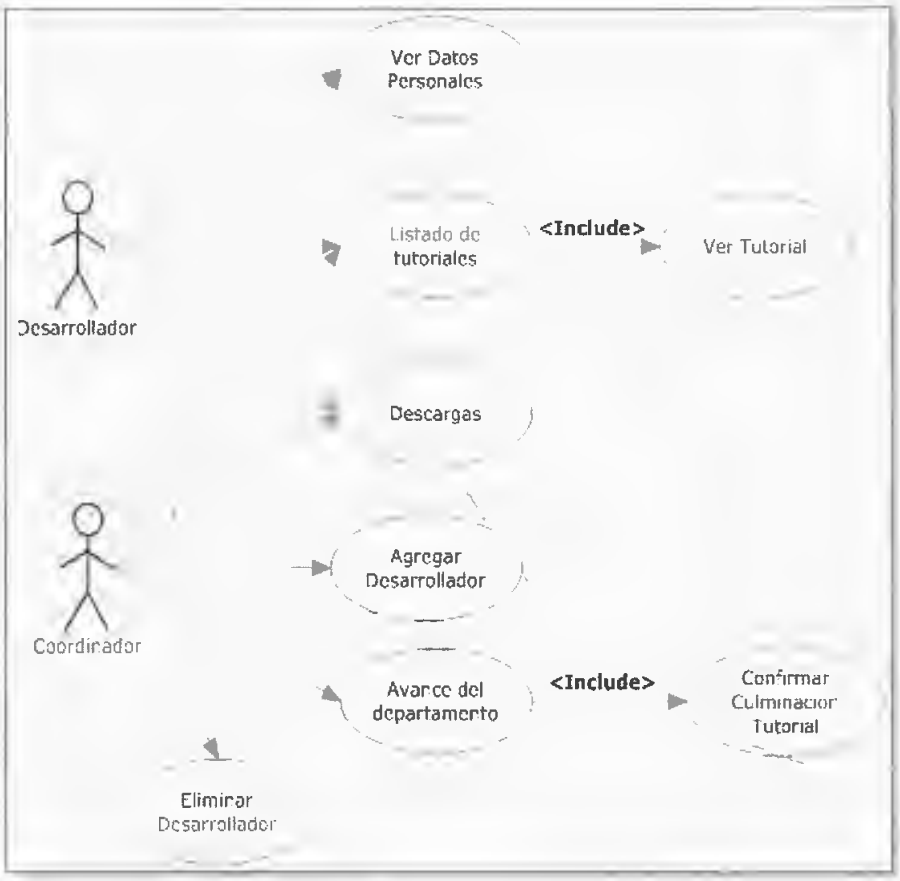


Figura 1 - 1: Diagrama de Casos de Uso - Fuente: Elaboración propia

A continuación se detallan cada uno de los casos de uso. El diseño de la aplicación se encuentra en el apéndice A: Diseño del tutorial.

En base a las consideraciones explicadas anteriormente se procedió a desarrollar la aplicación. Este portal tiene la particularidad de ser desarrollado utilizando ASP.Net a diferencia del resto de los aplicativos de la Banca en Línea Empresas que se desarrollaron utilizando Java. Esta decisión se tomó para que el equipo de BOLE tenga oportunidad de familiarizarse con otro tipo de plataformas de desarrollo que se utilizan en Mercantil Banco Universal.

IV.1.3 Flujo básico del tutorial

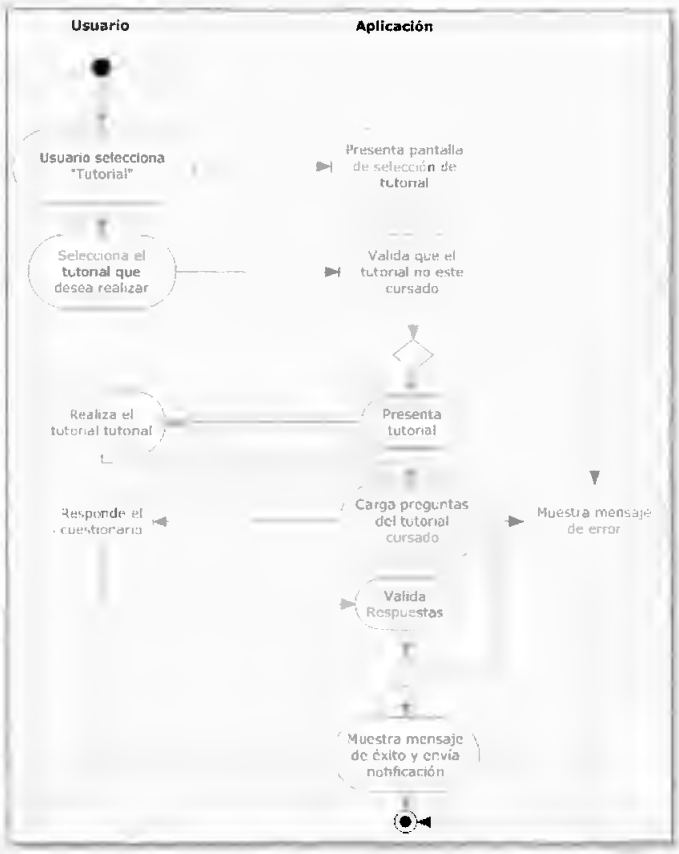


Figura 1 - 2: DF: Tutoriales - Fuente: Elaboración propia

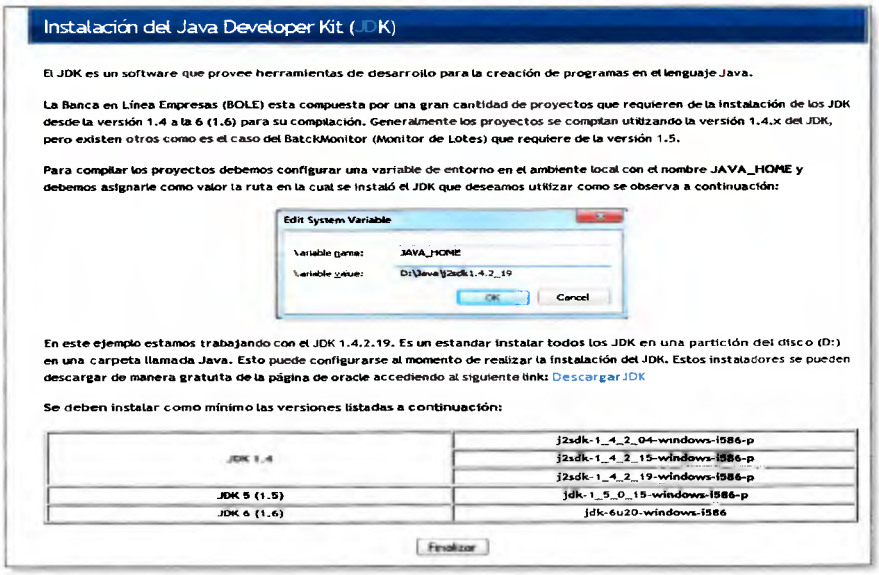
Este es el flujo principal de la aplicación, cuando un desarrollador o coordinador seleccionan la opción "Tutoriales" del menú se les presenta una pantalla de consulta donde pueden observar los tutoriales disponibles para su plataforma de desarrollo. En esta pantalla se presenta el porcentaje de avance que llevan en la realización de los tutoriales. Dicha consulta se puede observar en la figura 1 - 3.

En esta pantalla el usuario puede seleccionar cualquier tutorial que no haya cursado previamente para realizarlo. En caso de seleccionar un tutorial que ya realizó el sistema presentará un mensaje de error, ya que no podrá cursar un tutorial previamente aprobado.

Figura 1 - 3: Ventana principal tutoriales - Fuente: Elaboración propia

De igual manera, podría suceder que para acceder a ciertos tutoriales requiera culminar otros previamente. En este caso también se presentará un mensaje de error. Al seleccionar un tutorial que no esté cursado el sistema presentará la página del mismo como se puede apreciar en la figura 1 - 4.

Cada tutorial es en sí mismo una página personalizada que presenta una serie de pasos o instrucciones que el usuario deberá seguir para instalar o configurar cierto producto que compone el sistema de la Banca en Línea Empresas. Una vez finalizado cada uno de los tutoriales que existen en la plataforma de BOLE el usuario registrará un porcentaje de avance de 100% lo que indica que ha instalado el producto.



Instalación del Java Developer Kit (JDK)

El JDK es un software que provee herramientas de desarrollo para la creación de programas en el lenguaje Java.

La Banca en Línea Empresas (BOLE) esta compuesta por una gran cantidad de proyectos que requieren de la instalación de los JDK desde la versión 1.4 a la 6 (1.6) para su compilación. Generalmente los proyectos se compilan utilizando la versión 1.4.x del JDK, pero existen otros como es el caso del BatckMonitor (Monitor de Lotes) que requiere de la versión 1.5.

Para compilar los proyectos debemos configurar una variable de entorno en el ambiente local con el nombre JAVA_HOME y debemos asignarle como valor la ruta en la cual se instaló el JDK que deseamos utilizar como se observa a continuación:

Edit System Variable

Variable name: JAVA_HOME

Variable value: D:\Java\j2sdk1.4.2_19

OK Cancel

En este ejemplo estamos trabajando con el JDK 1.4.2.19. Es un estandar instalar todos los JDK en una partición del disco (D:) en una carpeta llamada Java. Esto puede configurarse al momento de realizar la instalación del JDK. Estos instaladores se pueden descargar de manera gratuita de la página de oracle accediendo al siguiente link: Descargar JDK

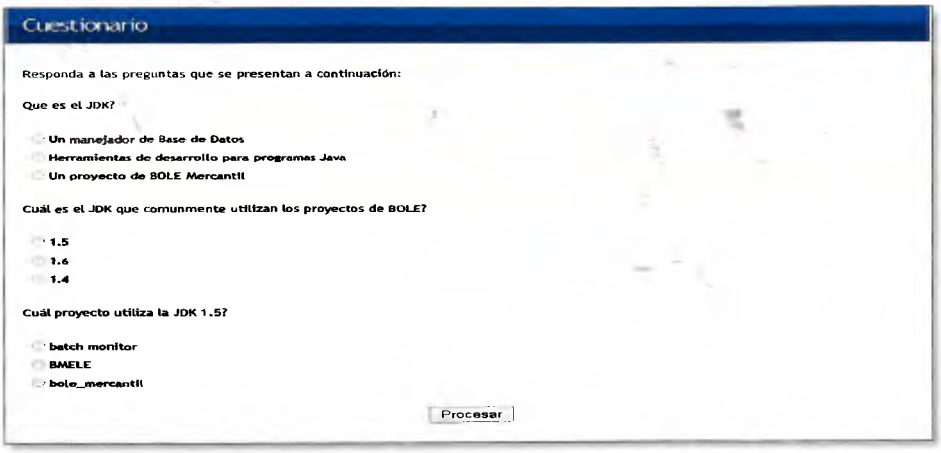
Se deben instalar como mínimo las versiones listadas a continuación:

JDK 1.4	j2sdk-1_4_2_04-windows-i586-p
	j2sdk-1_4_2_15-windows-i586-p
	j2sdk-1_4_2_19-windows-i586-p
JDK 5 (1.5)	jdk-1_5_0_15-windows-i586-p
JDK 6 (1.6)	jdk-6u20-windows-i586

Finalizar

Figura 1 - 4: Tutorial - Fuente: Elaboración propia

Los tutoriales se presentan en forma de "wizard", algunos tienen más páginas que otros. Al llegar a la página final de un tutorial se presenta el botón de finalización, al seleccionarlo el usuario será remitido a la parte final del tutorial, donde el sistema presentará un cuestionario de tres preguntas como se observa en la figura 1 - 5.



Cuestionario

Responda a las preguntas que se presentan a continuación:

Que es el JDK?

- Un manejador de Base de Datos
- Herramientas de desarrollo para programas Java
- Un proyecto de BOLE Mercantil

Cuál es el JDK que comunmente utilizan los proyectos de BOLE?

- 1.5
- 1.6
- 1.4

Cuál proyecto utiliza la JDK 1.5?

- batch monitor
- BMELE
- bole_mercantil

Procesar

Figura 1 - 5: Cuestionario - Fuente: Elaboración propia

El usuario que se encuentra cursando el tutorial debe contestar al menos dos(2) preguntas de forma correcta para aprobar el tutorial. Una vez culminado se presenta un mensaje de éxito o error, dependiendo del resultado del cuestionario y el sistema envía una notificación al coordinador de la plataforma para informarle que el usuario ha culminado un tutorial. De este modo los coordinadores están informados en todo momento del progreso de los desarrolladores y pueden asistirlos o confirmar la información de forma personal en cualquier momento. En la figura 1 - 6 se puede apreciar el mensaje de éxito.

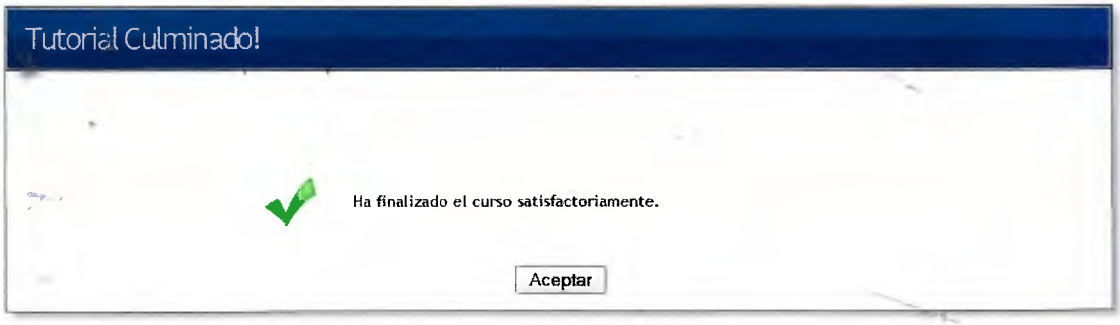


Figura 1 - 6: Mensaje de culminación de tutorial - Fuente: Elaboración propia

La aplicación web elaborada no sólo se diseñó para permitirles a los desarrolladores instalar fácilmente la Banca en Línea Empresas en un ambiente local, también les permitirá en un futuro acceder a diferentes contenidos que pueden contribuir con su formación dentro de la Empresa. También fue realizada pensando en los coordinadores, ya que les suministra un ambiente de control y supervisión de los desarrolladores que conforman su departamento, ya que anteriormente no existía ninguna herramienta que cumpliera este propósito. Las funcionalidades adicionales están contenidas en el apéndice B: Funcionalidad extras del tutorial.

IV.2 Herramientas de Calidad de Código

En el mercado existe una gran cantidad de herramientas de calidad de código, de todas ellas destacan tres: JTest es una herramienta desarrollada por la compañía

Parasoft, Google también ofrece una herramienta llamada CodePro AnalytiX y la herramienta de código abierto Sonar.

Estas herramientas realizan un análisis del código fuente de un proyecto para detectar los errores que poseen, el resultado de este análisis es entregado a los usuarios a través de reportes que presentan los valores de las métricas que se implementaron y las violaciones a las reglas de programación establecidas, con la finalidad de que sean corregidos para obtener un código de mayor calidad. Para Mercantil Banco Universal existen varios criterios que deben tomarse en cuenta al momento de elegir alguna de las opciones, estos son: Simplicidad de instalación, configuración y utilización, compatibilidad con Maven y Jenkins, seguridad, capacidad de configuración y costo.

JTest [PARASOFT01]

JTest es una solución de desarrollo integrado para la automatización de pruebas de una amplia gama de prácticas probadas para mejorar la calidad del software. Se centra en prácticas de validación de código Java y se integra perfectamente con otro producto de la compañía llamado Parasoft SOATest para permitir funcionalidades de extremo a extremo y cargar pruebas de alta complejidad.

JTest permite realizar un análisis estático del código fuente, realizar pruebas unitarias y detectar errores en tiempo de ejecución.

CodePro AnalytiX [GOOGLE01]

CodePro AnalytiX es una herramienta para pruebas de código Java para desarrolladores que trabajan con Eclipse y se preocupan por mejorar la calidad del software, reducir los costos de desarrollo y sus horarios. Ayuda a los desarrolladores a reducir los errores a medida que el código es desarrollado y mantiene las prácticas de programación acorde con los lineamientos de la organización.

Al igual que JTest permite realizar un análisis estático del código y generar pruebas unitarias. Adicionalmente permite establecer reglas de auditoría, habilitar métricas estándar de calidad de código y analizar las dependencias de un proyecto.

Sonar [SONAR01]

Sonar es una herramienta de código abierto que sirve para gestionar la calidad del código y por esto abarca sus 7 ejes: Arquitectura y diseño, duplicación del código, pruebas unitarias, complejidad del código, bugs potenciales, reglas de programación y comentarios. Es una aplicación web que permite configurar en línea reglas, alertas, exclusiones, ajustes, entre otros. Aprovechando su base de datos no sólo permite combinar indicadores en conjunto, sino que también permite mezclarlo con medidas históricas.

Entre las bondades de Sonar destacan las siguientes: Establecer reglas de programación personalizadas, realizar pruebas unitarias, utilizar métricas estándar, observar la evolución de las métricas aplicadas a un proyecto en el tiempo, compatibilidad con Maven y Jenkins, tutoriales de instalación, facilidad de visualización de los proyectos y permite establecer reglas de autenticación para brindar seguridad a las instancias de los proyectos.

Análisis comparativo

Las tres herramientas estudiadas cumplen con el mismo propósito y cada una de ellas presenta varias características en común, como lo son el desarrollo de pruebas unitarias, el análisis del código, el análisis de las métricas de calidad, la aplicación de reglas de programación seleccionadas para cada proyecto, entre otras. Pero son los criterios establecidos por Mercantil Banco Universal los que se deben analizar para poder llegar a una conclusión sobre cuál se adapta mejor a las necesidades de la compañía.

En el siguiente cuadro se puede observar, bajo estos criterios, lo que ofrecen cada una de las herramientas:

Criterios	JTest	Sonar	CodePro AnalytiX
Simplicidad	Requiere de cursos para instalarse que pueden tomar una semana	Entre media hora y 45 minutos.	Entre Media hora y 45 minutos.
Compatibilidad	Es compatible con otros productos Parasoft	Es compatible con Jenkins y Maven	Es compatible con Eclipse
Seguridad	Posee protocolos de autenticación y autorización	Posee protocolos de autenticación y autorización	Posee protocolos de autenticación y autorización
Configuración	Permite configurar reglas de programación	Permite configurar y crear reglas de programación	Permite configurar reglas de programación
Costo	Requiere la adquisición de licencias de más de 2.000 dólares	Es gratuita	Es gratuita

Tabla 2 - 1: Comparación de herramientas - Fuente: Elaboración propia

Al examinar esta comparación de las tres herramientas se puede tomar una decisión sobre cuál de las tres satisface las necesidades de la compañía y permite reducir los errores de código fuente del producto.

Decisión

El objetivo de esta pasantía es disminuir los errores de código fuente del producto BOLE Mercantil, para poder aumentar la calidad del mismo. A diferencia de JTest que es una herramienta que sirve para medir la calidad del software en general, Sonar y CodePro AnalytiX están dedicados única y exclusivamente a la calidad del código fuente y ofrecen una amplia variedad de reglas preinstaladas que ayudan a cumplir este propósito.

Desde el punto de vista de la instalación y configuración de las aplicaciones, para utilizar el JTest es necesario asistir a diferentes cursos de capacitación que ofrece

Parasoft, mientras que Sonar viene con un tutorial sencillo de dos minutos que explica como instalar el producto. Por su parte, CodePro AnalytiX se instala desde Eclipse utilizando la opción para instalar un software nuevo desde un sitio en la web.

Para utilizar JTest es necesario adquirir una serie de licencias, no solo para este producto, sino también para otros productos que ofrece la compañía para mejorar su funcionalidad. Sonar es una herramienta de código abierto que puede ser descargada de forma gratuita sin ninguna garantía o condición. CodePro AnalytiX es una herramienta gratuita que ofrece Google como parte de sus servicios.

Hasta este punto, Sonar y CodePro AnalytiX cumplen con los criterios establecidos por la compañía y son preferibles sobre JTest, lo que marca la diferencia entre estas dos herramientas es la compatibilidad que pueda tener cada una de ellas con el ambiente de desarrollo de la institución.

Todos los proyectos de la Banca en Línea Empresas fueron desarrollados como proyectos de Maven, en este sentido Sonar está listo para ser utilizado por proyectos Maven, basta con configurar el objetivo de compilación de el típico "clean install" a "Sonar" para poder utilizar las bondades que este producto ofrece, por su parte CodePro AnalytiX solo puede ser utilizado por medio de Eclipse y, aunque el banco cuenta con esta herramienta para programar, algunos desarrolladores prefieren utilizar otras opciones como NetBeans para desempeñar sus actividades. JTest, por otro lado, ofrece compatibilidad con otras herramientas que Parasoft ofrece, pero no es igual de conveniente, en este caso Sonar es mucho más sencillo de aplicar que las otras dos opciones.

Finalmente, otro factor decisivo es que Sonar se conecta fácilmente con herramientas de Integración continua como Jenkins, Hudson y Atlassian Bamboo. De estas opciones, en Mercantil Banco Universal, se utiliza Jenkins como herramienta de integración continua.

Con base a este análisis se puede concluir que la herramienta que satisface la mayor cantidad de criterios definidos por el banco y se integra mejor con el ambiente de trabajo bajo el cual está desarrollada la Banca en Línea Empresas es Sonar. La opción de Google es bastante buena, pero su dependencia de Eclipse, que es lo que hace que su instalación sea mucho más rápida que las otras dos herramientas, la descarta frente a Sonar. Por último, la opción de Parasoft requiere de una importante inversión de dinero por parte de la compañía que no garantiza una mejora significativa con respecto a las otras herramientas que son gratuitas.

Se utilizará Sonar como la herramienta elegida para reducir los errores de código fuente de la Banca en Línea Empresas.

IV.3 Implementación de la Herramienta de Calidad de Código Seleccionada.

Para la configuración del Sonar se requirió de la participación de los equipos de Arquitectura y Calidad de Software del banco. El equipo de arquitectura es el encargado de manejar la plataforma de integración continua, por su parte, el equipo de calidad de software está encargado de evaluar cada proyecto realizado por el equipo de desarrollo de software del banco para detectar errores que puedan presentar y finalmente determinar si dichos proyectos pasan a producción o deben ser modificados.

Una vez seleccionado Sonar como la herramienta de calidad de código a utilizar, ambos equipos se dieron la tarea de preparar los ambientes globales para instalar y probar las funcionalidades del producto.

El proceso de instalación del Sonar se encuentra detallado en el apéndice C: Instalación Sonar.

Una vez instalado el producto se procedió a determinar las reglas de programación con las cuales se mediría la calidad de código de la Banca en Línea Empresas. Este proceso fue ejecutado por el equipo de arquitectura de Mercantil Banco Universal en conjunto con el equipo de calidad de software, el producto que se generó fue un archivo XML que contiene todas las reglas a implementar, una muestra de este archivo se encuentra en el apéndice D: Reglas de Sonar.

Una vez instalado y configurado el ambiente con las reglas establecidas se procede a compilar el proyecto "bole_mercantil". Este proyecto fue creado utilizando Maven, debido a esto basta con cambiar el objetivo de la compilación a "Sonar" en lugar de "Clean Install" que es el objetivo que se utiliza regularmente.

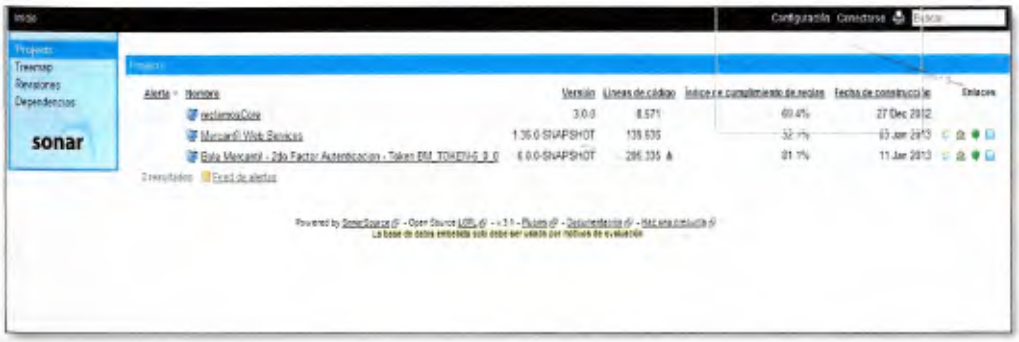


Figura 3 - 1: Proyectos analizados en Sonar - Fuente: Elaboración propia

El proceso de compilación tarda varios minutos cuando es la primera vez que se ejecuta, en este caso llevó media hora. Una vez finalizó este proceso se pudo apreciar en la página principal de Sonar la aparición del proyecto como se puede apreciar en la figura 3 - 1, al cual se le generaron los reportes resultantes de la aplicación y las métricas de calidad de software que posee la misma. Los errores de código fuente están clasificados en 5 tipos: Bloqueadores, críticos, mayores, menores e informativos, a continuación se describe en qué consiste cada uno:

Bloqueadores: Son los errores que impiden que un proyecto pueda pasar a producción, y se consideran "Bloqueadores" a todos aquellos errores que pueden causar que el sistema deje de funcionar.

Críticos: Son piezas de código que podrían causar errores al momento de ejecución del proyecto y suelen exceder los niveles máximos de complejidad de código establecidos por los administradores del proyecto.

Mayores: Son piezas de código que pueden causar, en situaciones extremas, errores al momento de la ejecución del proyecto y suelen exceder los niveles mínimos de complejidad de código establecidos por los administradores. Entran en esta clasificación también las piezas de código que hacen que el proyecto sea difícil de mantener, esto puede ser por falta de documentación, la forma en que se elaboran los condicionales, entre otros.

Menores: Son piezas de código que no causan errores pero pueden optimizarse aplicando mejores prácticas de programación.

Informativos: Los errores informativos no impactan el funcionamiento del proyecto, pero ayudan a obtener un código mucho más limpio. El ejemplo típico es eliminar las librerías importadas de una clase en la que no se utilizan.

Figura 3 - 2: Errores totales de la clase evaluada - Fuente: Elaboración propia

Con este reporte generado se procedió a seleccionar una clase del proyecto que contuviera más de 200 errores, este fue el caso de la clase

"PagoInscritoBean.java". Esta clase poseía un total de 231 errores considerados "Mayores" como se aprecia en la figura 3 - 2. A continuación se detallarán cada uno de los errores que fueron corregidos durante el desarrollo de este Trabajo Instrumental de Grado:

Errores de "Javadoc Method

```
89
90    protected PagoInscritoBean(DatosInfoServicio [] info, NumberFormat nFormat) {
```

Javadoc Method

Missing a Javadoc comment.

Figura 3 - 3: Ejemplo error javadoc - Fuente: Elaboración propia

En la figura 3 - 3 se puede apreciar cómo se identifica en el reporte de sonar un error del tipo Javadoc Method. Para eliminar estos errores fue necesario realizar la documentación pertinente de cada uno de los métodos de la clase, Sonar verifica a detalle que esté debidamente documentado, no es suficiente colocar la instrucción @param info, donde info es un parámetro del método, sino que debe explicarse el parámetro brevemente. En la figura 3 - 4 se puede apreciar el error planteado anteriormente corregido. Esto se aplicó para los 84 errores de este tipo que existían.

```
/**
 * @param info informacion de datos del servicio
 * @param nFormat numero de formato
 */
protected PagoInscritoBean(DatosInfoServicio [] info, NumberFormat nFormat) {
```

Figura 3 - 4: Solución error javadoc - Fuente: Elaboración propia

Errores "If Stmts Must Use Braces", " If Else Stmts Must Use Braces "

```
210         this.valorAPagar.setValor(info[i].getValor());
211         if (!info[i].getValor().equals("")) this.valorAPagar.setValorCNconFormato(nFormat.format(Double.parseDouble(info[i].getValor())));
```

If Stmts Must Use Braces

Avoid using if statements without curly braces

Figura 3 - 5: Ejemplo errores if/else - Fuente: Elaboración propia

Como se puede observar en la figura 3-5, el condicional que presenta el error no posee llaves ({}) que indiquen donde inicia y culmina la condición, esto puede ocurrir en las sentencias que solo llevan una condición (únicamente "if") o incluso en condiciones que tienen un flujo alterno o "si no" (condiciones "if/else"). Ambos errores se catalogan de diferente forma pero la solución es la misma, es una buena práctica colocar las llaves al inicio y al final de cualquier condicional para obtener un código más comprensible, a pesar de que podría funcionar sin ellas, esta solución se aprecia en la figura 3 - 6. En total existen 48 errores de este tipo.

```
if (!info[i] getValor().equals("")){
        this.valorAPagar.setValorCNconFormato(nFormat.format(Double.parseDouble(info[i].getValor())));
```

Figura 3 -6: Solución error if/else - Fuente: Elaboración propia

Error "Equals Avoid Null"

```
118            if (info[i].getNombreVariable().equalsIgnoreCase("PAGOTERCEROS") ) {
   Equals Avoid Null
   String literal expressions should be on the left side of an equalsIgnoreCase comparison.
```

Figura 3 - 7: Ejemplo de error equals - Fuente: Elaboración propia

Como se distingue en la figura 3 - 7, estos errores ocurren cuando en una comparación es posible que el valor que se presenta a la izquierda sea nulo, lo que provocaría el envío de una excepción por parte del sistema. La solución a este problema es invertir la posición de la variable por la de la constante, el resultado se observa en la figura 3- 8. En la clase seleccionada existen un total de 69 errores de este tipo que fueron eliminados.

```
if ("PAGOTERCEROS".equalsIgnoreCase(info[i].getNombreVariable()) ) {
```

Figura 3 - 8: Solución de error equals - Fuente: Elaboración propia

Una vez corregidos todos los errores explicados anteriormente se volvió a generar el reporte de Sonar para verificar que efectivamente se logró mejorar la calidad del código fuente reduciendo sus errores, el resultado obtenido se puede apreciar en la figura 3 - 9

Figura 3 - 9: Reporte final de errores - Fuente: Elaboración propia

Sonar es una excelente herramienta de control de calidad de código, aunque es una aplicación de código abierto es confiable y estable, y puede ser utilizada para proyectos sencillos y complejos. La meta de este trabajo era reducir en un 20% los errores de código fuente de la aplicación BOLE Mercantil. Si se pueden determinar todos los errores del código fuente de un producto, entonces es posible reducirlos en un 100%, aunque este es un trabajo que requeriría una buena cantidad de tiempo, en especial si el proyecto es complejo.

Este es un proceso nuevo y BOLE Mercantil es un producto que tiene varios años en el mercado, el equipo de arquitectura de software de la organización junto con el de calidad del software decidieron que deben generarse los reportes de Sonar cada vez que exista un nuevo proyecto o requerimiento para la Banca en Línea Empresas, con la finalidad de que estas nuevas versiones no aumenten los valores actuales de las métricas, ya que lo ideal es que con cada nueva versión del producto se logre disminuir los errores del código fuente para que, poco a poco, el código de la aplicación obtenga un mayor nivel de calidad.

En la sección de apéndices se encuentran cada uno de los reportes generados utilizando la aplicación seleccionada, aquí se pueden detallar los 201 errores que fueron corregidos durante la ejecución de este trabajo. Los apéndices son el E: Reporte Sonar - Antes y el F: Reporte Sonar - Después.

IV.4 Migración de Ayudas Mercantil

El proyecto de ayudas consiste en crear una aplicación que almacene todas las página de ayuda de la Banca en Línea Empresas.

Figura 4 - 1: Botón de ayuda - Fuente: [MERCANTIL01]

Como se puede apreciar en la figura 4 - 1, en las páginas de la Banca en Línea Empresas existen botones de ayuda, estos se encuentra ubicados en la esquina superior derecha de la página, y cuando un usuario los selecciona despliegan una página de guía relacionada con la operación que están realizando (Figura 4 - 2).

Figura 4 - 2: Página de ayudas - Fuente: [MERCANTIL01]

Actualmente estas páginas se encuentran ubicadas en un servidor externo, y se accede a ellas a través de una referencia a una dirección URL que pertenece a ese servidor, como se observa en la figura 4 - 3. Mercantil Banco Universal desea eliminar la dependencia sobre este servidor y requiere del desarrollo de un proyecto que sirva para migrar las ayudas a una aplicación que esté contenida dentro de los proyectos de la Banca en Línea Empresas.

Figura 4 - 3: Vínculo al servidor externo - Fuente:[MERCANTIL01]

IV.4.1 Requerimientos

La aplicación a desarrollar debe cumplir con los siguientes requerimientos: a) transformar páginas HTML a JSP y b) vincular el acceso a estas página en BOLE Mercantil.

Los requerimientos no funcionales son: a) debe ser fácil de manipular para el departamento de contenidos, b) debe ser fácil acceder a las imágenes y modificarlas, c) debe ser posible que la aplicación registre los cambios sin apagar el servidor de aplicaciones y d) debe estar integrado a los proyectos actuales de la Banca en Línea Empresas.

IV.4.2 Solución[ORACLE01]

Para satisfacer los requerimientos listados anteriormente se investigó la funcionalidad de los proyectos de la Banca en Línea Empresas. Uno de estos proyectos es conocido como "Docroot" y consiste en un repositorio de imágenes que son referenciadas por los otros proyectos. Para cambiar alguna de estas imágenes no hace falta apagar el servidor en el que se encuentran alojados los otros proyectos, ya que se encuentra en un servidor diferente.

La documentación de Oracle define un "docroot" como una raíz de documentos alternativa que sirve para que una aplicación web tenga la capacidad de atender ciertas peticiones utilizando recursos que se encuentran en un directorio que no es el propio. Tomando esto en cuenta se decidió alojar el nuevo proyecto en el directorio alternativo junto a las imágenes que utilizan las aplicaciones, de este modo se podrá acceder a las páginas de ayudas como si se tratara de una imagen más.

El proyecto está estructurado en tres grandes paquetes:

- **Páginas:** Es el paquete en el que se alojan todos los JSP que contienen el cuerpo de las ayudas, este paquete se subdivide en un conjunto de directorios alusivos a cada aplicación de la Banca en Línea Empresas, y dentro de estos directorios particulares se alojan las ayudas que utiliza la aplicación representada. Por ejemplo, dentro del directorio "MELE" están todas las ayudas que posee el módulo transaccional de la Banca en Línea Empresas.

- **Enlace:** En este paquete se alojaría el archivo de propiedades que cumplirá con las funciones de una clase que aplica el patrón de diseño Fachada. Este es un archivo de propiedades cuya única función es la de mapear cada una de las imágenes con un identificador único, el cual será utilizado por las páginas para referenciar estas imágenes.

- **Imágenes:** Es un paquete que a su vez presenta varios subdirectorios en los que se alojan las imágenes que corresponden a un JSP, por este motivo el nombre de la carpeta debe ser idéntico al del JSP que utiliza las imágenes. La figura 4-4 esquematiza la estructura del proyecto.

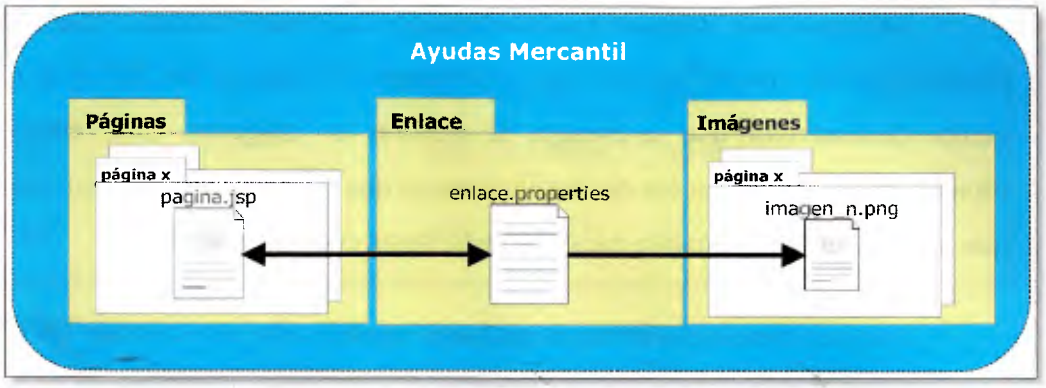


Figura 4 - 4: Estructura del proyecto de ayudas - Fuente: Elaboración propia

La comunicación de los tres paquetes está basada en un patrón de diseño llamado fachada. La base del patrón es crear un único punto de enlace entre dos clases con la finalidad de hacer la comunicación entre ambas más sencilla. En el caso de este proyecto, el archivo de propiedades cumple las funciones de un objeto que aplica el patrón, en el se mapean las rutas en las que se encuentran cada una de las imágenes y se identifican con una clave que se utilizará en los JSP para referirse a ellas.

La finalidad de la fachada es hacer más sencilla la relación del equipo de contenido con el código, ya que para sustituir una imagen no necesitan tener conocimientos de programación, simplemente pueden buscar la imagen que desean en el archivo de propiedades (este archivo puede modificarse utilizando cualquier editor de texto) y cambiar la dirección por la de la nueva imagen.

Otra posibilidad que se habilita es la de sobre escribir una imagen. Debido a que las páginas de ayudas y sus imágenes se encuentran en un servidor externo que no pertenece a Mercantil Banco Universal, es imposible copiar y pegar una imagen nueva sobre otra vieja para sustituirla, con la implementación de este proyecto es posible realizar esta operación.

Es importante destacar que dos equipos diferentes manejan este sistema, el primero son los desarrolladores, que se encargan de elaborar los JSP, y el otro es el equipo de contenido que se encarga de elaborar las imágenes. Una ventaja que les ofrece este mecanismo a los desarrolladores es que no necesitan conocer la ruta en la que se encuentra una imagen para utilizarla, basta con saber su clave.

Una vez creado el JSP no es necesario modificarlo para cambiar las imágenes, que es el único contenido que suele cambiar con el tiempo, simplemente se reemplaza la imagen o se modifica el archivo de propiedades, esto sirve para que no sea necesario reiniciar el servidor al momento de modificar una ayuda.

Para iniciar este desarrollo no fue necesaria la elaboración de un diagrama de casos de uso, ya que la única acción que se realiza es la de ingresar en la página de ayuda, el diagrama de flujo de la figura 4 - 5 explica este proceso.

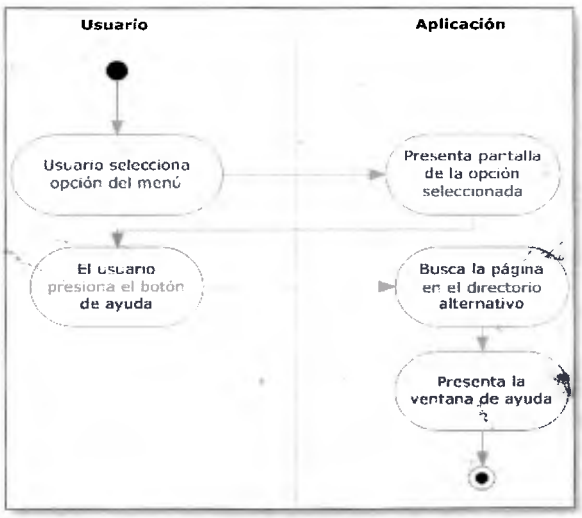


Figura 4 - 5: DF: Ayudas Mercantil - Fuente: Elaboración propia

Para aplicar este proyecto el primer paso que se debe seguir es descargar el proyecto "Docroot" de los repositorios del banco para descomprimirlo en el "docroot" de la instancia del proyecto en el servidor de aplicaciones, el resultado es un sistema

de directorios como el que se observa en la figura 4 - 6.

Name	Date modified	Type	Size
ayudas	27/08/2012 10:08 a...	File folder	
flash	20/07/2012 10:33 a...	File folder	
images	07/08/2012 10:27 a...	File folder	

Figura 4 - 6: Directorios Docroot - Fuente: Elaboración propia

En el directorio de ayudas se encuentra el proyecto desarrollado que contiene todas las ayudas con sus respectivas imágenes. Debido a que el proyecto está alojado en el "docroot" no es necesario apagar el servidor para que la aplicación tome los cambios realizados.

Para que los proyectos no sigan llamando a la URL del servidor externo es necesario realizar modificaciones al archivo de propiedades "UI_HELPS.properties" del proyecto MELE (Módulo Transaccional de BOLE) de la Banca en Línea Empresas. En la figura 4 - 7 se observa que todas las rutas de ayudas están mapeadas para buscar una URL del servidor externo.

```
145 #autogestion fideicomiso
146 jsp.fideicommissumIn.help      =  https://www.todo1.com/cs/Satellite?c=Page&cid=1259763548475&pagename=Ayudas%2FPage%2FAyudasBasicasMercantil
147 jsp.fideicommissumOu.help      =  https://www.todo1.com/cs/Satellite?c=Page&cid=1259763548379&pagename=Ayudas%2FPage%2FAyudasBasicasMercantil
148 jsp.fideicommissumApprove.help =  https://www.todo1.com/cs/Satellite?c=Page&cid=1259763548600&pagename=Ayudas%2FPage%2FAyudasBasicasMercantil
149 #Perfil - Administracion
150 jsp.perfil.administracion     = https://www.todo1.com/cs/Satellite?c=Page&cid=1266347778765&pagename=Ayudas%2FPage%2FAyudasBasicasMercantil
151 jsp.perfil.edicion            = https://www.todo1.com/cs/Satellite?c=Page&cid=1266347778758&pagename=Ayudas%2FPage%2FAyudasBasicasMercantil
152 #Transaction manager - Mensajes del historico de transaciones
153 jsp.transactionManager.history.help = https //www.todo1.com/cs/Satellite?c=Page&cid=1259763525637&pagename=Ayudas%2FPage%2FAyudasBasicasMercantil
154
155 ## ESTADO DE CUENTA DIGITA
```

Figura 4 - 7: Código de referencia al servidor externo - Fuente: MELE

Estas rutas deben ser sustituidas por la ruta del directorio alternativo, el formato puede visualizarse en la figura 4 - 8.

```
145 #autogestion fideicomiso
146 jsp.fideicommissumIn.help      =  https //www.todo1.com/cs/Satellite?c=Page&cid=1259763548475&pagename=Ayudas%2FPage%2FAyudasBasicasMercantil
147 jsp.fideicommissumOu.help      =  https //www.todo1.com/cs/Satellite?c=Page&cid=1259763548379&pagename=Ayudas%2FPage%2FAyudasBasicasMercantil
148 jsp.fideicommissumApprove.help =  https //www.todo1.com/cs/Satellite?c=Page&cid=1259763548600&pagename=Ayudas%2FPage%2FAyudasBasicasMercantil
149 #Perfil - Administracion
150 jsp.perfil.administracion     = https://www.todo1.com/cs/Satellite?c=Page&cid=1266347778765&pagename=Ayudas%2FPage%2FAyudasBasicasMercantil
151 jsp.perfil.edicion            = /mercantil/ayudas/src/main/webapp/Crear_perfil.jsp
152 #Transaction manager - Mensajes del historico de transaciones
```

Figura 4 - 8: Código de referencia al docroot - Fuente: MELE

Al ingresar nuevamente a la aplicación y seleccionar la opción de ayuda el sistema no buscará la ayuda en un servidor externo, ahora buscará dentro del "docroot" el JSP de ayuda establecido en el archivo de propiedades de MELE y desplegará la misma página que se encontraba antes (Figura 4 - 9).

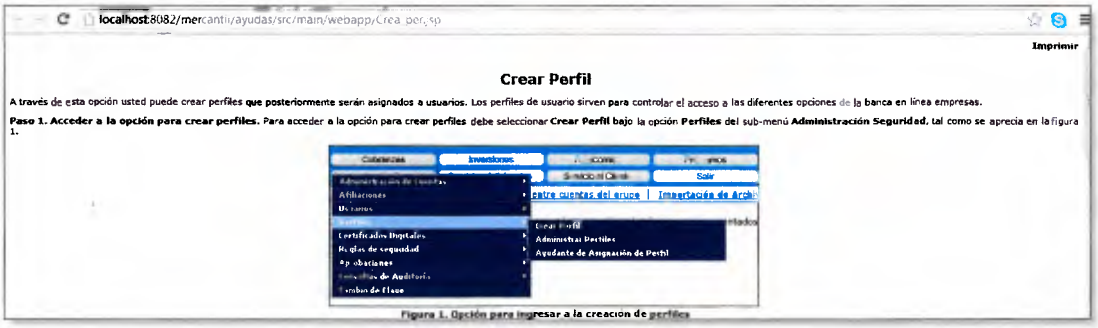


Figura 4 - 9: Página de ayuda desde docroot - Fuente: [MERCANTIL01]

La diferencia está en que el URL referencia el archivo del "docroot" y no la dirección del servidor externo (Figura 4 - 9). Esto genera varias ventajas:

Primero que todo, es más rápido acceder y manipular el "docroot" que un servidor externo. Para agregar una ayuda en el servidor externo actual un miembro del equipo de contenidos debe ingresar a una aplicación web y subir cada una de las imágenes que utiliza la página para generar el URL con el cual se va a referenciar. Luego es necesario colocar estos URL en el código HTML de las ayudas y finalmente debe subir la ayuda culminada al servidor. Utilizando este proyecto lo único que tendría que hacer es colocar las imágenes en una carpeta dentro del docroot y asignarles una clave única, en ese momento el equipo de desarrollo elabora el JSP y utiliza las claves creadas para referenciar las imágenes.

Otra ventaja es la eliminación de la probabilidad de un mal funcionamiento de las ayudas en caso que el servidor externo utilizado falle o se encuentre fuera de servicio. Debido a que las ayudas se encuentran alojadas en el directorio alternativo

de la aplicación sólo estarán fuera de servicio si la aplicación a la que pertenecen lo está.

Finalmente el desarrollo de este proyecto le da la oportunidad a la empresa de prescindir de los servicios del proveedor del servidor externo y utilizar el dinero que se invierte allí en otros proyectos.

IV.5 Segundo Factor de Autenticación BMELE

Durante la ejecución de este Trabajo Instrumental de Grado se elaboró el Módulo administrativo de Tokén y los requerimientos fueron los siguientes:

- Debe ser posible la búsqueda de usuarios o grupos con token por varios criterios (Código de grupo, RIF, serial de Token, usuario).

- Debe ser posible liberar un Token que tenga un usuario asignado.

- Debe ser posible suspender un Token que tenga o no usuario asignado.

- Debe ser posible sincronizar el número de un Token con el valor actual del servicio ENTRUST.

- Debe ser posible generar accesos de emergencia para un Token con usuario asignado.

- Debe ser posible reiniciar el número de accesos de emergencia para un Token con usuario asignado.

- Debe ser posible observar los detalles de un Token con usuario asignado.

- Debe ser posible auditar las acciones ejecutadas por usuarios del Módulo de Bancos.

- Debe modificarse la funcionalidad administrativa de grupos para incluir los Token.

Luego de especificados los requerimientos el equipo de diseño de la compañía entregó el caso de uso del proyecto al equipo de desarrollo e inició el proceso de codificación del proyecto.

IV.5.1 Administrar Token - Detalle Token

Para iniciar el proceso se ingresa a BMELE con un usuario del perfil "BANKPERFILPRODUCTOSEDICION" ó "BANKPERFILFUNCIONARIOS. En el menú se selecciona la opción "Token" seguido de la pestaña "Administrar Token" y el sistema presenta la pantalla de búsqueda de la figura 5 - 1:

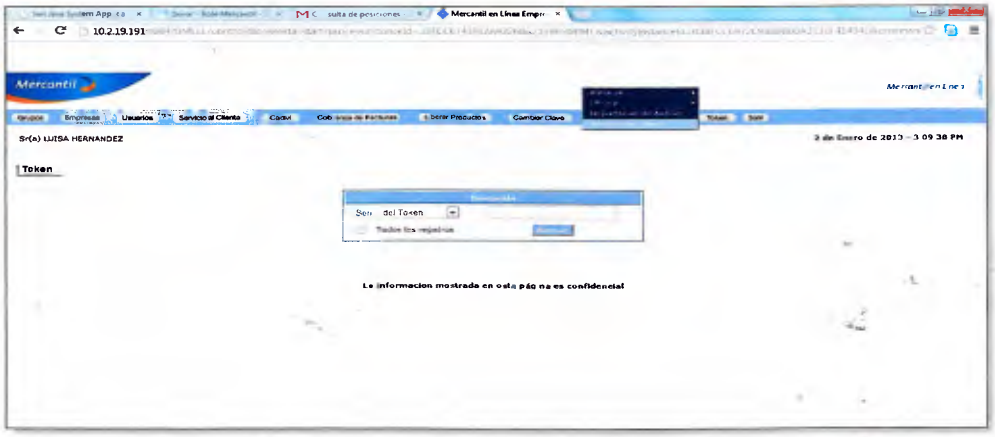


Figura 5 - 1: Ventana de búsqueda token - Fuente: Elaboración propia

En esta pantalla se toman en cuenta las siguientes validaciones:

- El campo no puede estar vacío, a menos que se seleccionen todos los registros
- El código introducido debe pertenecer a un grupo, token o usuario existentes

Si el usuario introduce criterios de búsqueda válidos para los parámetros presentados (Código de grupo, RIF, Serial del Token o Usuario) y presiona el botón "Buscar" el sistema procesa el criterio de búsqueda y le presenta al usuario el resultado de la búsqueda como se observa en la figura 5 - 2.

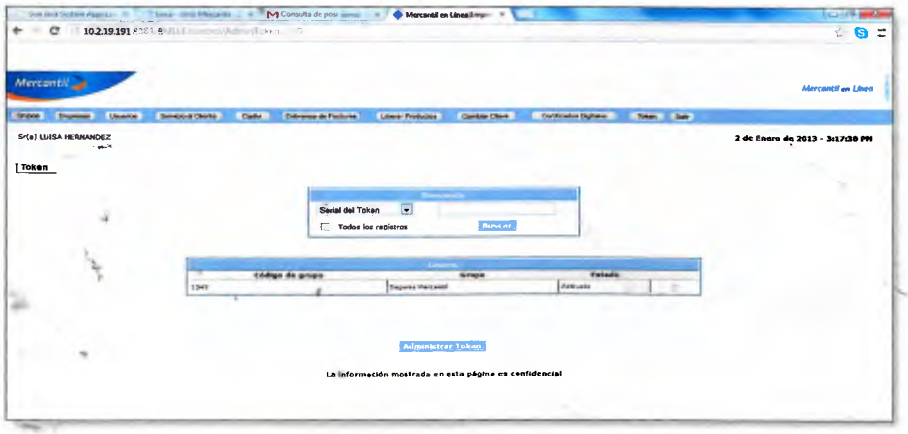


Figura 5 - 2: Resultado de búsqueda - Fuente: Elaboración propia

En la pantalla anterior el usuario debe seleccionar el grupo, usuario o token que desea consultar, al presionar el botón "Adminstrar Token" el sistema lo remitirá a la pantalla que se ilustra en la figura 5 - 3:

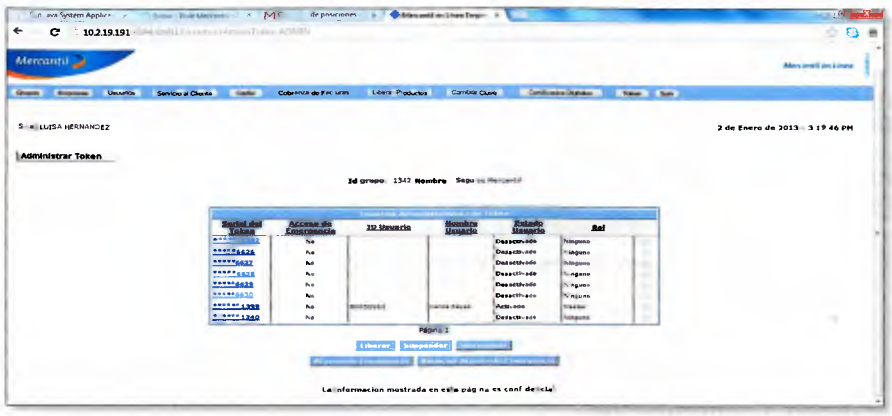


Figura 5 - 3: Página de administración de token - Fuente: Elaboración propia

Si el usuario no selecciona ninguna opción el sistema lo alertará y no realizará ninguna búsqueda.

Para consultar el detalle de un token el usuario debe hacer clic en el hipervínculo que presenta el serial de la pieza a examinar, este número se encuentra enmascarado y para realizar la operación el token debe poseer un usuario asignado. Esta proceso conducirá al usuario a la pantalla del detalle del token que se aprecia en la figura 5 - 4:

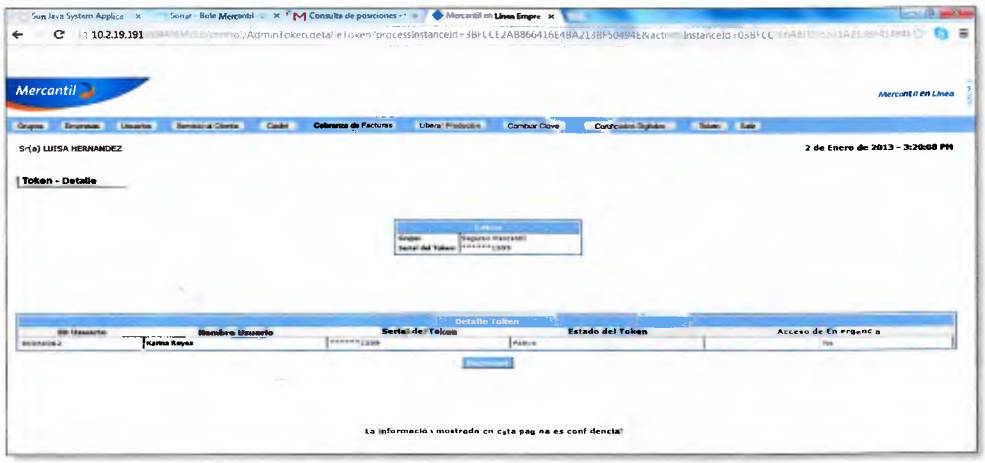


Figura 5 - 4: Detalle token - Fuente: Elaboración propia

Las opciones de liberar, suspender, reiniciar, acceso de emergencia y reiniciar accèso de emergencia junto con las modificaciones y el modulo de auditorías están contenidas en el apéndice G: Desarrollo de opciones administrativas Token.

IV.6 Diseño de la Nueva Arquitectura

La Banca en Línea Empresas está formada básicamente por tres proyectos que se describen a continuación:

Bole Mercantil: Es el núcleo de la aplicación, este proyecto contiene toda la lógica de negocios del sistema, está dividido en transacciones, cada una de ella tiene

una clase Enterprise Java Bean (EJB) en la cual se desarrolla el código de la transacción y otras clases que varían según las necesidades de cada transacción.

Módulo Transaccional (MELE): Es una aplicación web que utilizan los usuarios de Mercantil Banco Universal para realizar diferentes transacciones sobre sus cuentas empresariales.

Módulo de Bancos (BMELE): Es una aplicación web utilizada por funcionarios de Mercantil Banco Universal para realizar diferentes operaciones administrativas sobre la Banca en Línea Empresas.

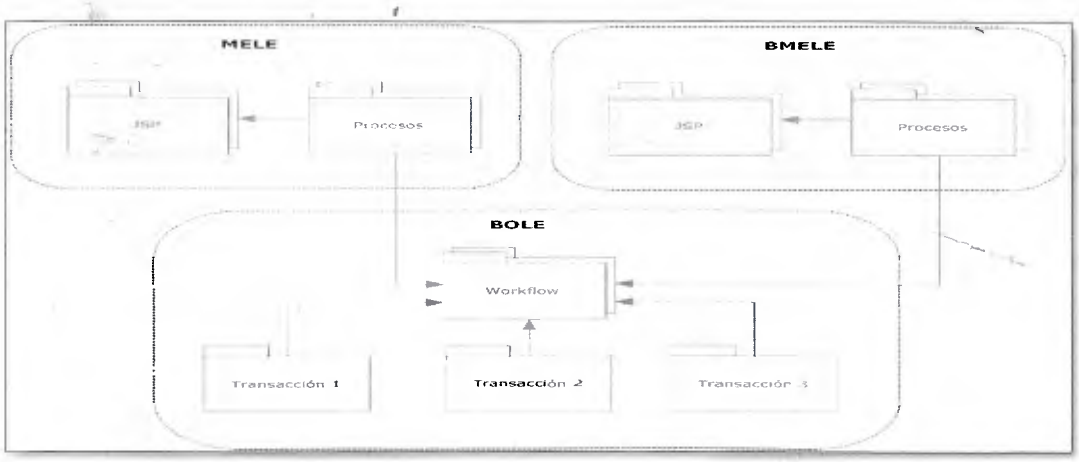


Figura 6 - 1: Arquitectura actual bole_mercantil - Fuente: Elaboración propia

En la figura 6 - 1 se puede apreciar la arquitectura que está actualmente implementada para el sistema. El sistema fue construido utilizando un framework al que le llaman "workflow", su funcionamiento es bastante sencillo, en las aplicaciones MELE y BMELE, cada uno de los JSP es mapeado en un archivo XML, dicho archivo es conocido como el "proceso", ya que contiene paso a paso cada una de las acciones que se realizan en la ejecución de una transacción (de aquí viene el nombre de "workflow").

En el proceso también se describe cuáles métodos se ejecutan en la transacción, y por este motivo en BOLE, cada una de las transacciones tiene una clase EJB donde se encuentran todos los procesos que serán ejecutados en la transacción, esto genera una arquitectura de dos capas, la presentación que serían las aplicaciones web MELE y BMELE y toda la lógica que está contenida en BOLE.

La capa lógica presenta un grave problema, no posee ninguna regla o patrón para desarrollar código sobre la plataforma, lo que ha generado casos como procesos cuyo código está desarrollado únicamente en el EJB, cada desarrollador puede hacer lo que considere mejor en cada una de las transacciones sin seguir ningún lineamiento. Es por este motivo que al evaluar el código de Bole Mercantil utilizando Sonar, la herramienta de calidad de software instalada anteriormente, se observa que existe una gran cantidad de errores y código repetido como se demuestra en la figura 6 - 2.

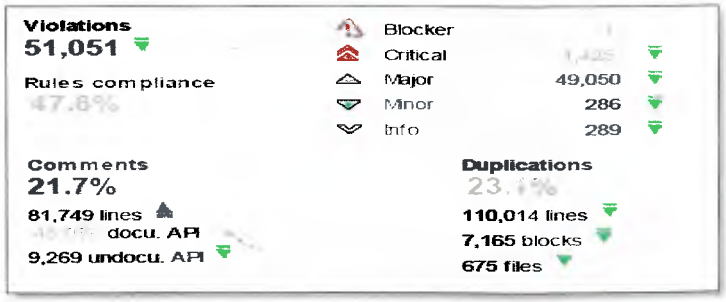


Figura 6 - 2: Métricas Sonar - Fuente: Elaboración propia

En una reunión con la coordinación de la Banca en Línea Empresas se elaboró una lista de problemas considerados graves de la arquitectura actual:

1) No existe reusabilidad ni portabilidad ya que cada pieza de código se desarrolla específicamente para satisfacer una transacción, 2) es difícil de mantener ya que cada desarrollador realiza el código de una forma diferente y 3) existe una gran cantidad de código repetido.

De igual forma se determinaron algunas virtudes de la arquitectura actual y se listaron: 1) utilización del workflow, 2) división de responsabilidades entre la interfaz y la lógica y 3) implementación de EJB.

Es por esto que se propone desarrollar una nueva arquitectura para la Banca en Línea Empresas que, basándose en las virtudes de la arquitectura anterior permita solucionar los problemas expuestos utilizando patrones de diseño de software.

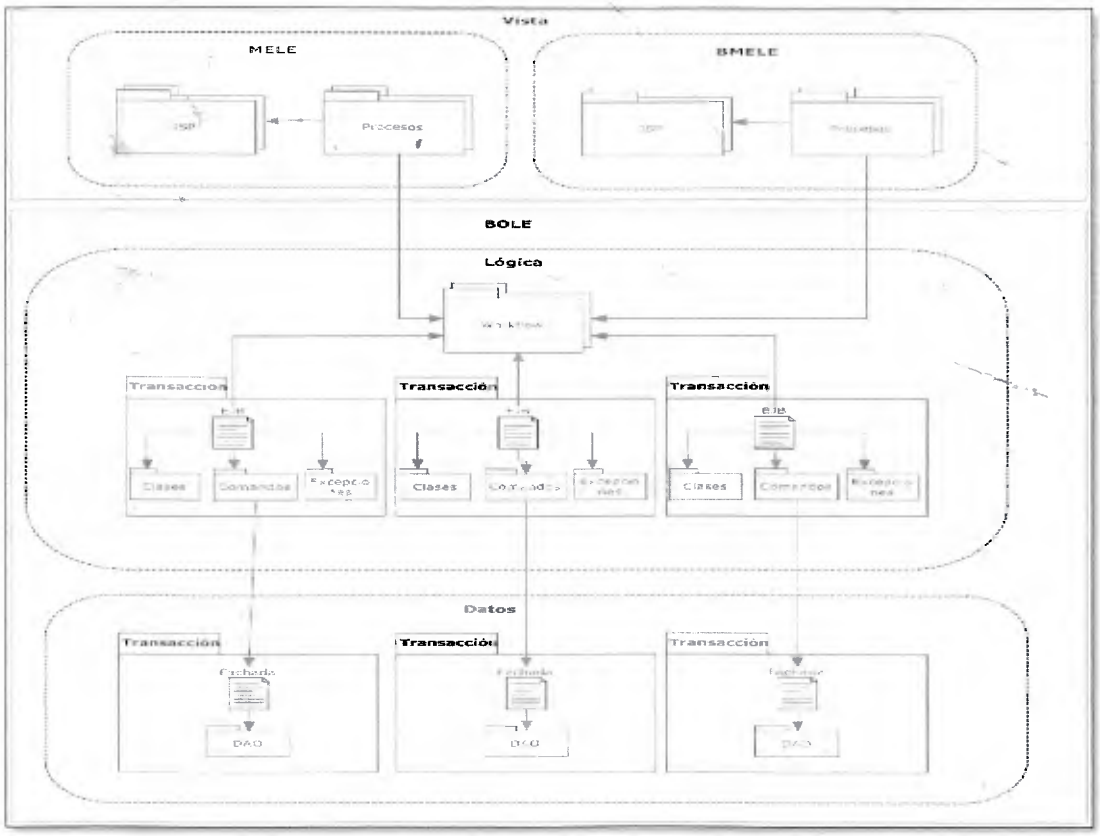


Figura 6 - 3: Arquitectura nueva - Fuente: Elaboración propia

Con las necesidades anteriores en mente se decidió cambiar la arquitectura de dos capas a una de tres capas como se observa en la figura 6 - 3. A continuación se detallará cada una de las capas:

Vista: Es la capa que representa la interfaz de usuario, esta está compuesta por los proyectos MELE y BMELE, en esta capa se aplica una variación del patrón conocido como Modelo-Vista-Controlador (MVC)

La vista está compuesta por todos los JSP que contienen los proyectos, el controlador viene dado por el archivo XML del proceso que especifica cuáles son los métodos y páginas que deben presentarse según las acciones ejecutadas en la vista. Finalmente el modelo está representado por las otras dos capas.

Lógica: En esta capa se encuentra la lógica de negocios de la aplicación, está dividida en transacciones y cada una de ellas posee los siguientes directorios: comandos, excepciones y clases. Además cada transacción debe tener un Enterprise Java Bean (EJB) en el cuál se aplicará el patrón de diseño fachada, ya que funcionará como una interfaz simplificada por medio de la cual la vista accederá a los métodos necesarios para ejecutar una operación.

Para eliminar el problema de reiteración de código, reusabilidad y portabilidad se aplicó el patrón de diseño comando, este consiste en encapsular métodos en objetos que representan una acción específica. Actualmente en la Banca en Línea Empresas si un desarrollador necesita un método para realizar la paginación de una lista debe reescribir el código ya existente en otras clases. En la figura 6 - 4 se puede observar el método processPaging en la clase AdminTokenEJB.java.

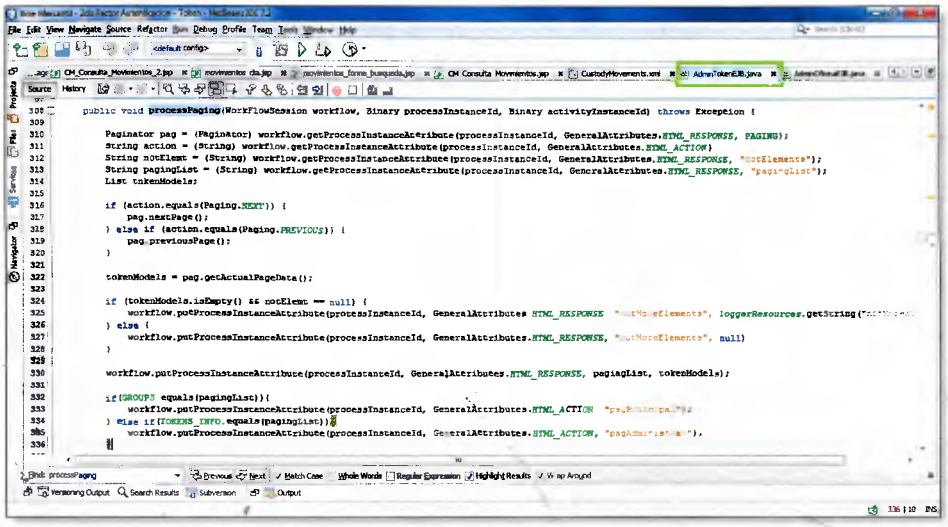


Figura 6 - 4: ProcessPaging AdminToken - Fuente: Código de bole_mercantil

Se puede observar en la figura 6 - 5 que el mismo método se repite también en la clase AdminTokenOfficeEJB.java.

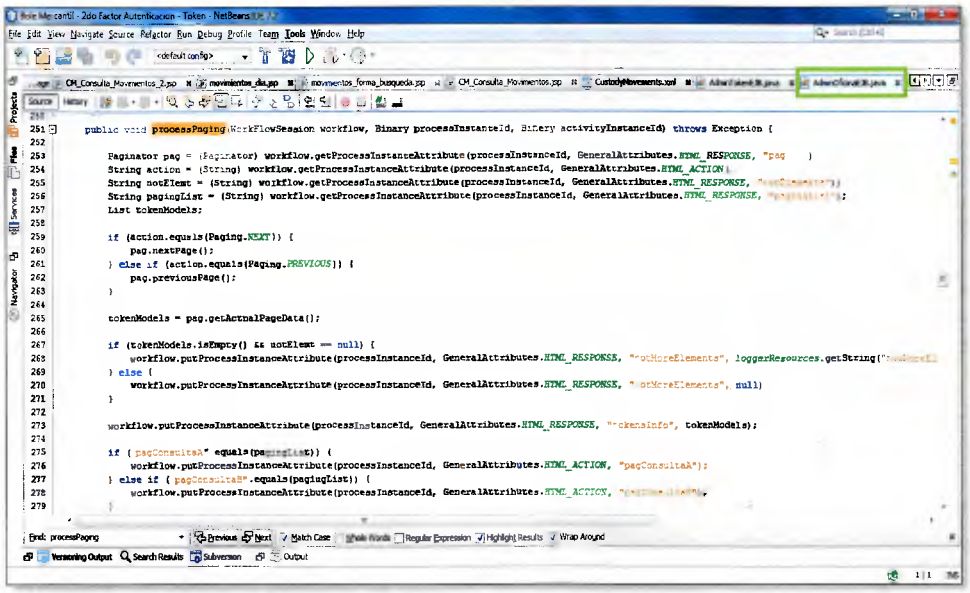


Figura 6 - 5: ProcessPaging AdminTokenOficina - Fuente: Código de bole_mercantil

Esto genera duplicación de código y hace complicado el mantenimiento del sistema, ya que si es necesario realizar alguna modificación a este método se debe realizar el cambio en cada una de las clases que lo aplican. En este ejemplo, si el método processPaging cambia, el desarrollador deberá modificar las clases AdminTokenEJB y AdminTokenOficinaEJB.

Al encapsular el método en una clase cuya única función es devolver la información de un usuario, todas las clases de la Banca en Línea Empresas podrán acceder a este método, en caso de una modificación, basta con sustituir el comando en cuestión y el cambio aplicará para todas las clases que lo utilicen. En la figura 6 - 6 se puede apreciar el método ejecutar del comando que resolvería el problema del ejemplo anterior.

```
public void Ejecutar() throws Exception{

    if (action.equals(Paging.NEXT)) {
        pag.nextPage();
    } else if (action.equals(Paging.PREVIOUS)) {
        pag.previousPage();
    }

    lista = pag.getActualPageData();

    if (lista.isEmpty() && notElemt == null) {
        workflow.putProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_RESPONSE, "notMoreElements", loggerResources.getString("notMoreEle
    } else {
        workflow.putProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_RESPONSE, "notMoreElements", null)
    }

    workflow.putProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_RESPONSE, "listaInfo", lista)

    if ( pagConsultaA".equals(pagingList)) {
        workflow.putProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_ACTION, "pagConsultaA");
    } else if ( pagConsultaB".equals(pagingList)) {
        workflow.putProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_ACTION, "pagConsultaB")
    }
}
```

Figura 6 - 6: Comando processPaging - Fuente: Elaboración propia

Basta con eliminar el código repetido de las clases anteriores e invocar el comando processPaging, que seguirá cumpliendo la misma funcionalidad que los dos métodos presentados anteriormente.

En caso que un desarrollador repita el mismo comando, la herramienta de calidad de código Sonar, instalada previamente, al realizar el análisis del código fuente identificará la duplicación y el desarrollador podrá sustituir el comando repetido por el ya existente.

Estos comandos serán los encargados de acceder a la última capa, la de datos, para solicitar información a la base de datos en caso de ser necesario.

Datos: La capa de datos está dividida en transacciones, cada transacción tiene una clase que, al igual que en la capa Lógica, implementa el patrón fachada, con la finalidad de simplificar el acceso a la información que contiene esta capa. Esta clase será utilizada por los diferentes comandos para obtener información de las bases de datos de la Banca en Línea Empresas.

La información de las bases de datos es accedida mediante a objetos de acceso a datos (DAO), este es un patrón que establece que para acceder a la información que se contiene en la base de datos es necesario crear un objeto el cuál se encargará únicamente de este proceso, haciendo transparente el acceso a los datos a todas las demás clases del sistema.

Para poner a prueba la arquitectura definida anteriormente se implementó un requerimiento de la Banca en Línea Empresas, este requerimiento consistió en lo siguiente:

La compañía requiere que en la aplicación de Módulo de Bancos (BMELE) se implemente una funcionalidad que permita a una persona que ingresa a la aplicación con el perfil de edición de productos (BANKPERFILPRODUCTOSEDICION) ó bajo el perfil de funcionario (BANKPERFILFUNCIONARIO) recibir una consignación de Token para la oficina a la cual pertenece.

Una consignación consiste en la entrega de un grupo de lotes de cajas de Token que realiza el almacén que los contiene a las diferentes oficinas de Mercantil Banco Universal, estos Token luego de ser recibidos podrán ser entregados a los cliente de la Banca en Línea Empresas.

Este requerimiento pertenece al proyecto de Token, y está dedicado al manejo de inventarios del mismo. Es crítica su implementación ya que si las oficinas no pueden aceptar las consignaciones que realizan los almacenes, no se podrá llevar un control del inventario de token que maneja la empresa. Es categorizado como un requerimiento principal del proyecto al que pertenece.

El diseño del requerimiento puede encontrarse en el apéndice I: Documento Técnico Token.

IV.7 Implementación de la Arquitectura

Este requerimiento había sido elaborado utilizando la arquitectura anterior, la transacción adminOficina del proyecto "bole_mercantil" contenía las clases que se utilizaban para aceptar una consignación. En la figura 7 - 1 se aprecia como estaba distribuido el proyecto.

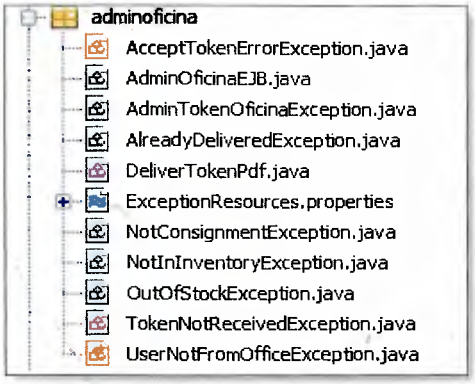


Figura 7 - 1: Distribución anterior - Fuente: Código de bole_mercantil

Como se puede observar, bajo esta distribución se desarrollaron varias excepciones que son utilizadas por la clase EJB. En esta clase se encuentra toda la lógica o el código que realiza las operaciones de los requerimientos: Aceptar consignación, consultar consignaciones y entregar token.

Para implementar la arquitectura planteada por este trabajo fue necesario realizar una redistribución y refactorización del código que se encontraba en la transacción de administrar oficina. Se crearon paquetes destinados a cumplir ciertos propósitos:

Excepciones: Paquete que contiene todas las excepciones que puede arrojar la transacción.

Comando: Paquete que contiene todos los comandos que aplican para la transacción. Este paquete es público, por lo que puede ser accedido por cualquier otra transacción. Se determinó durante la implementación que se pueden clasificar los comandos en dos tipos:

Generales: Son aquellos que pueden aplicar a otras transacciones sin ser modificados. Se identifican debido a que su método ejecutar() son funciones.

Específicos: Son aquellos particulares para una transacción, sirven para extraer información que será presentada al usuario final. Se caracterizan por ser procedimientos.

Clases: Contiene todas las clases que se requieran para ejecutar la transacción.

Datos: es la capa de datos y está compuesta por una clase que implementa el patrón Fachada y un paquete que contiene todos los Objetos de Acceso a Datos que utiliza la transacción, al igual que los comandos, estos objetos pueden ser accedidos por las fachadas de cualquier otra transacción.

La parte de acceso a datos anteriormente era manejada de dos formas:

Sin Objetos de Acceso a Datos: No poseían ningún método para abstraer la lógica del proceso de acceso a datos

Objetos de Acceso a Datos y ayudantes: Poseían objetos de acceso a datos y clases estáticas que se encargaban de llamar las funciones residentes en los DAO. Como se observa en la figura 7 - 2.

Figura 7 - 2: Helpers - Fuente: Código de bole_mercantil

La nueva arquitectura define un estándar diferente a los otros dos:

Se utilizan Objetos de Acceso a Datos que sólo pueden ser accedidos por las fachadas generadas para cada transacción. Los ayudantes hacían bien el trabajo de abstracción de una capa con otra (aunque originalmente no se define una capa de datos en la arquitectura original, esta implementación fue realizada por un desarrollador de la aplicación debido a que no existía ningún estándar de manejo de datos). La desventaja es que una transacción que utiliza cierto ayudante tiene acceso a todos los métodos del DAO que representa.

Al implementar una fachada particular para cada transacción se abstrae totalmente del proceso de acceso a datos, ya que cada una conocerá únicamente los métodos que puede utilizar. Esto facilita la comprensión del código, ya que es menos extenso y más preciso, y adicionalmente genera cierta seguridad dentro del mismo, ya que se hace imposible acceder a métodos que no le corresponden a una transacción y podrían llegar a generar errores. Se puede observar la fachada de datos en la figura 7 - 3.

```
public class TokenOfficeData {

    public static String getUserOffice(String id) throws Exception {...}

    /**...*/
     public static void acceptConsignment(TokenConsignmentBean consignment, LoginInf loginInf) throws Exception {...}

    /**...*/
     public static TokenConsignmentBean getConsignByOffice(String office, String status) throws Exception {...}

    /**...*/
    public static void updateBoxStatus(TokenBoxBean bean) throws TokenDaoException {...}

    /**...*/
     public static void updateConsignmentStatus(TokenStockBean tokenBean, String status, LoginInf loginInf) throws Exception {...}
}
```

Figura 7 - 3: Fachada de acceso a datos - Fuente: Elaboración propia

Se puede apreciar la distribución final en la figura 7 - 4.

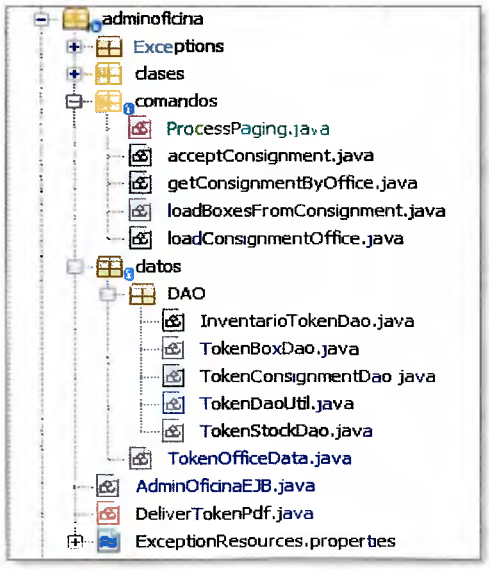


Figura 7 - 4: Nueva distribución - Fuente: Elaboración propia

IV.7.1 Evidencia del cambio

El cambio más profundo que realiza esta arquitectura es la eliminación de los EJB como clases que realizan todas las acciones y la introducción de los comandos, además de la clara definición de una tercera capa encargada únicamente del acceso a datos.

El requerimiento se desarrolló inicialmente utilizando tres métodos que procesaban la información que solicitaba la página, de necesitar acceder a la base de datos, utilizaban los ayudantes para lograrlo.

El primer método tiene como función cargar última consignación asignada a una oficina. El código original se encontraba en el EJB y se puede observar en la figura 7 - 5.

```
public void loadLastConsignment(WorkFlowSession workflow, Binary processInstanceId, Binary activityInstanceId) throws Exception {

        ArrayList listaCajas = new ArrayList();

        //Buscamos el id de la oficina del usuario que ejecuta el proceso.
        LoginInf login = (LoginInf) workflow.getAttributeValue(Constants.LOGIN);
        String userOfficeId = InventarioTokenHelper.getUserOffice(login.getUserId());
        if (userOfficeId.trim().length() == 0){
            throw new NotConsignmentException(exceptionResources.getString(NOT_INVENTORY_EXCEPTION));
        }
        //Cargamos las consignaciones enviadas a la oficina del usuario logeado.
        TokenConsignmentBean consignment = TokenConsignmentHelper.getConsignByOffice(userOfficeId, TokenConsignmentBean.CONSIGNMENT_STATUS_SEND);
        //Cargamos la lista de cajas del lote de la consignación enviada
        if (consignment != null) {
            InkenLoteBean lote;
            Iterator lotes = consignment.getLotes().iterator();

            while(lotes.hasNext()) {
                lote = (TokenLoteBean) lotes.next();
                Iterator iterador = lote.getTokenBoxes().iterator();
                while (iterador.hasNext()) {
                    TokenBoxBean caja = (TokenBoxBean) iterador.next();
                    listaCajas.add(caja);
                }
            }

            workflow.putProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_ACTION, "consignmentLoaded");
            workflow.putProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_RESPONSE, "consignment", consignment);
            workflow.putProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_RESPONSE, "listaCajas", listaCajas);
        } else {
            throw new NotConsignmentException(exceptionResources.getString(NOT_INVENTORY_EXCEPTION));
        }
```

Figura 7 - 5: Código original cargar consignación - Fuente: Código de bole_mercantil

Al implementar la nueva arquitectura este método se dividió en dos comandos generales: uno que obtiene las consignaciones de una oficina, y el otro que carga las cajas de token recibidas en una lista. Esto reduce la responsabilidad del EJB de realizar la transacción a combinar ambos métodos para enviar a la vista la información solicitada. El resultado se aprecia en la figura 7 - 6.

```
public void loadLastConsignment(WorkFlowSession workflow, Binary processInstanceId, Binary activityInstanceId) throws Exception {
    //Cargamos las consignaciones enviadas a la oficina del usuario logeado.
    TokenConsignmentBean consignment = new getConsignmentByOffice(workflow, processInstanceId, activityInstanceId).Ejecutar();
    //Cargamos la lista de cajas del lote de la consignación enviada.
    if (consignment != null) {
        ArrayList listaCajas = new loadBoxesFromConsignment(consignment).Ejecutar();
        workflow.putProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_ACTION, "consignmentLoaded");
        workflow.putProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_RESPONSE, "consignment", consignment);
        workflow.putProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_RESPONSE, "listaCajas", listaCajas);
    } else {
        throw new NotConsignmentException(exceptionResources.getString(NOT_INVENTORY_EXCEPTION));
    }
}
```

Figura 7 - 6: Código final cargar consignación - Fuente: Elaboración propia

Una vez cargada la consignación el sistema le presenta al usuario la pantalla de aceptación, el proyecto de la vista BMELE no sufrió ninguna modificación al implementar estos cambios, lo que evidencia la eficiencia de la arquitectura anterior al momento de manejar la interfaz con un patrón personalizado del Modelo-Vista-Controlador. La ventana es la que se presenta en la figura 7 - 7.

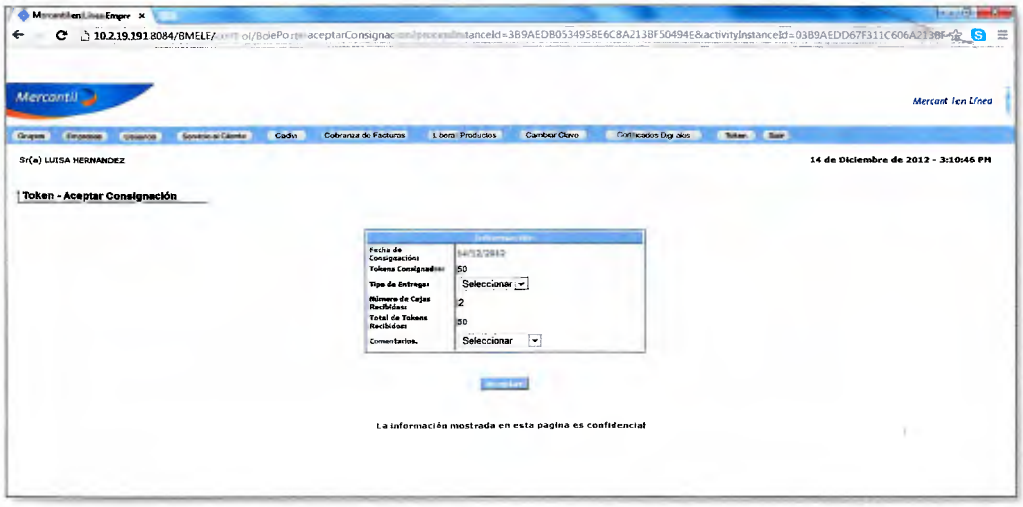


Figura 7 - 7: Aceptar consignación principal - Fuente: Código de BMELE

Cuando el usuario introduce los datos solicitados en el formulario el sistema procede a validarlos y prepararlos para su confirmación, este proceso era realizado por el EJB en un método cuya responsabilidad era ejecutar todo el proceso y enviar la información de verificación. El código de este método es muy extenso por lo que no

se puede apreciar del todo bien, aunque la figura 7 - 8 permite comprender su longitud.

Figura 7 - 8: Método para procesar formulario - Fuente: Código de bole_mercantil

En la nueva arquitectura se utilizó un único comando específico, cuya labor era igualmente la de procesar el formulario. La responsabilidad se extrajo totalmente del EJB resultando en un método cuya única labor es llamar al comando generado como se aprecia en la figura 7 - 9.

```
public void loadConsginmentOffice(WorkFlowSession workflow, Binary processInstanceId, Binary activityInstanceId) {
    try {
        loadConsignmentOffice comando = new loadConsignmentOffice(workflow, processInstanceId, activityInstanceId);
        comando.Ejecutar();
    } catch (ProcessInstanceNotFoundException e) {
        logger.error(e.getMessage(), e);
    } catch (InvalidAttributeException e) {
            logger.error(e.getMessage(), e);
    } catch (RemoteException e) {
            logger.error(e.getMessage(), e);
    }
}
```

Figura 7 - 9: Método final para procesar formulario - Fuente: Elaboración propia

Una vez validado y procesado el formulario el sistema presenta al usuario la ventana de verificación, en la que deberá analizar que los datos a procesar estén

correctos, en caso que lo estén procede a aceptar y el sistema almacenará la información de la consignación. Se puede apreciar la ventana en la figura 7 - 10.

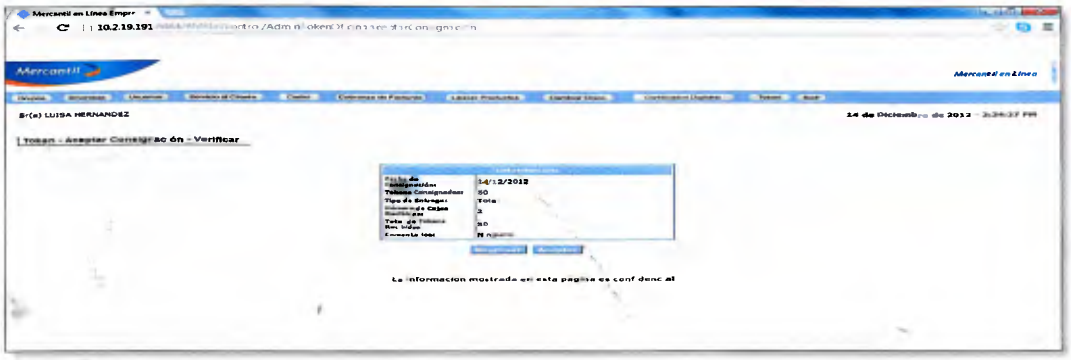


Figura 7 - 10: Ventana de verificación de consignación - Fuente: Código de BMELE

El método final tiene como responsabilidad almacenar la data en la base de datos del sistema, esta operación al igual que las demás era realizada en el EJB de la transacción, como se observa en la figura 7 - 11.

```
public void acceptConsignment(WorkflowSession workflow, Binary processInstanceId, Binary activityInstanceId) throws ProcessInstanceNotFoundException, InvalidAttributeException

    Map response;
    response = (Map) workflow.getProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_RESPONSE);
    LoginInf login = (LoginInf) workflow.getAtributeValue(Constants.LOGIN);
    TokenConsignmentBean consignment = (TokenConsignmentBean) response.get("consignmentVer");
    Collection cajasNoAceptadas = (Collection) response.get(                    );
    TokenBoxBean box;
    TokenStockBean token;
    try {
        TokenConsignmentHelper.acceptConsignment(consignment, login);
        Iterator iterador = cajasNoAceptadas.iterator();
        Iterator tokensIt;

        while (iterador.hasNext()) {
            box = (TokenBoxBean) iterador.next();
            box.setStatus(TokenBoxBean.BOX_STATUS_NOT_RECEIVED);
            TokenBoxHelper.updateStatus(box);
            tokensIt = box.getTokens().iterator();
            while (tokensIt.hasNext()) {
                token = (TokenStockBean) tokensIt.next();
                TokenStockHelper.updateConsignmentStatus(token, TokenStockBean.CONSIGNMENT_STATUS_NOT_ACCEPTED, login);
            }
        }

        workflow.putProcessInstanceAttribute(processInstanceId, GeneralAttributes.HTML_ACTION, "AcceptSuccess");
        logger.info(USUARIO + " " + login.getUserId() + " Consignación aceptada el " + consignment.getConsignmentDateFormatted() + " a la oficina " + consignment.getOfficeName(
    } catch (Exception e) {
        logger.error(USUARIO + " " + login.getUserId() + " Problema aceptando la consignación realizada el " + consignment.getConsignmentDateFormatted() + " a la oficina " + c
    }
```

Figura 7 - 11: Método inicial aceptar consignación - Fuente: Código de bole_mercantil

En la nueva arquitectura se implementó de igual forma un único comando cuya función era comunicarse con la capa de datos a través de la fachada de la

transacción y almacenar la información generada. El EJB únicamente debía llamar al comando correspondiente como se observa en la figura 7 - 12.

```
public void acceptConsignment(WorkFlowSession workflow, Binary processInstanceId, Binary activityInstanceId)
    acceptConsignment comando = new acceptConsignment(workflow, processInstanceId, activityInstanceId);
    comando.Ejecutar();
}
```

Figura 7 - 12: Código final aceptar consignación - Fuente: Elaboración propia

Si el proceso se lleva a cabo exitosamente el sistema muestra al usuario un mensaje de éxito y de lo contrario le informa que no se logró procesar la transacción.

Como se puede evidenciar anteriormente, la nueva arquitectura provee al sistema de mecanismos que son capaces de eliminar los problemas de los que sufría la arquitectura anterior. La inclusión del patrón de diseño comando permite construir piezas de código más limpias y sencillas, contribuyendo con una mejora en el proceso de mantenimiento del sistema. Los comandos generales ayudan a eliminar el problema del código repetido, fomentando la reutilización del código ya escrito.

La implementación de una tercera capa encargada de manejar todo lo que tenga que ver con obtención de datos y la implementación de las fachadas de cada transacción ayudan a abstraer cada capa del funcionamiento de la otra, lo que ayuda a la portabilidad junto con la implementación de los comandos generales.

Finalmente la formalización de este método de trabajo contribuye a generar un código estandarizado, por lo que su compresión es mucho más sencilla que en la arquitectura anterior donde los desarrolladores podían hacer cualquier cosa en el proyecto bole_mercantil. Al complementar esta arquitectura con el uso de la herramienta de calidad de código Sonar, se contribuye a mejorar la calidad general de la aplicación, con la ventaja de que la interfaz que observa el usuario final no es afectada por los cambios realizados en la nueva arquitectura.

Capítulo V. Resultados.

En el capítulo iv.1 se presentó el proceso de desarrollo de la aplicación web que sirve como tutorial de instalación de la Banca en Línea Empresas. Esta aplicación aportó también una nueva herramienta administrativa para los coordinadores, quienes pueden utilizarla para gestionar y monitorear a los desarrolladores de su departamento, este desarrollo se evidencia en los apéndices A y B. Adicionalmente se documentó el proceso de instalación del producto como contribución al problema de escasez de información que existe en la plataforma.

En el capítulo iv.2 y iv.3se presentó el proceso de instalación de una herramienta de calidad de código que se adecúa perfectamente a las necesidades de la institución y permite mejorar la calidad del producto BOLE Mercantil. Para demostrar la funcionalidad de la aplicación se eliminaron 200 errores del código fuente de la Banca en Línea Empresas, estos se pueden observar en los apéndices E y F, y se demostró que todos los errores producidos por las malas prácticas de programación y la falta de estándares pueden ser eliminados en su totalidad utilizando este mecanismo, aunque este es un proceso que requiere de una inversión importante de tiempo debido a la complejidad del proyecto. Esto aportó una nueva manera de evaluar el código de los proyectos de la institución unificando el trabajo de los equipos de desarrollo, arquitectura y calidad del software.

Durante el capítulo iv.4 se presentó el proceso de desarrollo de un proyecto que permite migrar las ayudas de la Banca en Línea Empresas del servidor externo utilizado actualmente a el código interno de la aplicación sin impactar el funcionamiento de la misma, aportando una solución a la dependencia del servidor mencionado que generaba gastos innecesarios a la compañía.

En el capítulo iv.5 y los apéndices G, e I se presentó el proceso de desarrollo del módulo administrativo del proyecto para incluir un segundo factor de

autenticación a la Banca en Línea Empresas en el Módulo de Bancos, este proyecto permite realizar funciones administrativas sobre los dispositivos que implementará la compañía para acceder al portal en un futuro cercano. Además, se implementó un módulo de auditorías que permite a los funcionarios del banco monitorear las actividades realizadas en cada una de las plataformas y se modificaron los módulos que utilizaban certificado digital para incorporar las funcionalidades de este nuevo proyecto.

Finalmente durante el capítulo iv.6 se diseñó una nueva arquitectura basada en las virtudes y defectos de la implementada actualmente. Esta arquitectura utiliza patrones de diseño para corregir los problemas de estándares y duplicación de código presentes en el sistema actualmente. Se implementó un requerimiento de la Banca en Línea Empresas utilizando esta nueva arquitectura para demostrar sus virtudes, entre ellas se puede mencionar una mejora considerable en la mantenibilidad del sistema y la posibilidad de reutilizar el código ya desarrollado sin impactar la capa de presentación de la aplicación.

Capítulo VI. Conclusiones y Recomendaciones.

Conclusiones.

Luego de haber realizado todas las investigaciones y pruebas pertinentes a este trabajo de instrumental, se concluyó que:

- La implementación del tutorial demostró lo versátiles que pueden llegar a ser las aplicaciones web, ya que no sólo se desarrolló un tutorial para instalar BOLE, también se creó una aplicación que sirve a los coordinadores para gestionar sus departamentos.
- Se demostró que existen diversas herramientas de calidad de código que se adecúan a las diferentes necesidades de las compañías y facilitan en gran medida el proceso de evaluación de calidad de un producto.
- Se obtuvieron nuevos mecanismos para manejar los contenidos de una página web. Estos contenidos no necesitan estar anclados a un servidor externo ya que actualmente existen soluciones más económicas y prácticas que pueden aplicarse para manejar estos procesos.
- Se observó claramente como la aplicación de patrones de diseño en una arquitectura puede generar una mejora significativa en el proceso de mantenimiento del sistema. También se demostró que su buena aplicación y la división de responsabilidades en una arquitectura en capas reduce el impacto que los cambios puedan tener en un sistema.
- Finalmente, se concluye que a través de la ingeniería informática se pueden realizar innovaciones a diferentes proyectos de tecnología para mejorar su funcionamiento y adaptarlos a las necesidades cambiantes del mundo actual.

Recomendaciones.

Para futuros proyectos es recomendable expandir las funcionalidades del tutorial para que preste nuevos servicios tanto a los coordinadores como a los desarrolladores. Es posible mejorar las áreas de notificaciones y descargas, incluyendo un servicio de envío de correos y la descarga de videos. Adicionalmente se puede mejorar el proceso de preguntas para que sean aleatorias y garanticen un aprendizaje significativo, ya que actualmente son un simple formalismo

Otra área que se puede mejorar son las ayudas, ya que presentan un formato muy antiguo y podrían actualizarse incluyendo contenido multimedia como videos y presentaciones dinámicas.

Es recomendable apoyarse en la nueva arquitectura para definir nuevos estándares de programación que sean debidamente documentados y se les faciliten a los desarrolladores por medio de la aplicación que se desarrolló para el tutorial.

Es posible también rediseñar la capa de presentación del Módulo de Bancos de la Banca en Línea Empresas para brindar un mejor servicio a sus usuarios.

Es posible desarrollar programas que puedan automatizar el proceso de creación de reglas para SONAR.

Finalmente sería altamente recomendable migrar los proyectos de la Banca en Línea empresas a un nuevo servidor de aplicaciones como Tomcat o JBoss de manera que se abandone el uso del Sun AppServer 7 que actualmente se encuentra en desuso. Igualmente es necesario realizar una actualización de la versión Java que utilizan estas aplicaciones, ya que la versión 1.4 que se utiliza actualmente no permite utilizar una gran cantidad de funcionalidades que se aplicaron desde la versión 1.5 en adelanto.

Bibliografía

Libros, Artículos, Informes Técnicos

-[CHARNEY01] - Charney, R. (2012). "Programming Tools: Code Complexity Metrics". <http://www.linuxjournal.com/article/8035>

-[FERNANDEZ01] - Fernández, Oscar. y otros. (1995). "Un enfoque actual sobre la calidad del software". <http://eprints.rclis.org/5424/1/aci05395.htm>

-[GARZAS01] - Garzás, Javier (2012). "Una lista de herramientas de calidad del software imprescindibles". <http://www.javiergarzas.com/2012/03/herramientas-de-calidad-software.html>

-[GOOGLE01] - (2012). " CodePro Analytix User Guide". Google Incorporated.< https://developers.google.com/java-dev-tools/codepro/doc/?hl=es-ES>

-[MICROSOFT01] - (2009). "Microsoft Application Architecture Guide, 2nd Edition: Chapter 21: Designing Web Applications". Microsoft Corporation. <http://msdn.microsoft.com/en-us/library/ee658099.aspx>

-[MICROSOFT02] - (2009). "Microsoft Application Architecture Guide, 2nd Edition: Chapter 1: What is Software Architecture?". Microsoft Corporation. <http://msdn.microsoft.com/en-us/library/ee658098.aspx>

-[MICROSOFT03] - (2003). "Unit Testing". Microsoft Corporation. <http://msdn.microsoft.com/en-us/library/aa292197%28v=vs.71%29.aspx>

-[MICROSOFT04] - (2012). "Using a Three-Tier Architecture Model (COM+)". Microsoft Corporation. <http://msdn.microsoft.com/en-us/library/windows/desktop/ms685068%28v=vs.85%29.aspx>

-[ORACLE01] - (2013). "Core J2EE Patterns - Data Access Object". Oracle Corporation. <http://www.oracle.com/technetwork/java/dataaccessobject-138824.html>

-[ORACLE01] - (2010). "Alternate Document Roots". Oracle Corporation. <http://docs.oracle.com/cd/E19776-01/820-4496/geqpl/index.html>

-[PCMAG01] - (2012). "software bug". PC Magazine. Publicado por Ziff Davis Publishing Holdings Inc. <http://www.pcmag.com/encyclopedia_term/0,2542,t=software+bug&i=51664,00.asp>

-[SPINELLIS01] - Spinellis, Diomidis (2006). "Repetition". <http://www.informit.com/articles/article.aspx?p=457502&seqNum=5>

-[UTAH01] - (2013). "Commenting". Publicado por la Universidad de UTAH. <http://www.cs.utah.edu/~germain/PPS/Topics/commenting.html>

-[WELLS01] - Wells, Don (2009). "Extreme Programming: A gentle introduction". <http://www.extremeprogramming.org/>

Sitios Web

-[DEFINICION01] Definición. <http://definicion.de/tutorial/>

-[ISO01] Software Quality Assurance Definition. <http://www.sqa.net/iso9126.html>

-[MERCANTIL01] Mercantil Banco Universal C.A. <http://www.bancomercantil.com/mercprod/site/empresas/02_mel.html>

-[PARASOFT01] Parasoft Corporation. <http://www.parasoft.com/jsp/home.jsp>

-[SONAR01] SONAR. <http://www.sonarsource.org/>

-[SOURCE01] Source Making. <http://sourcemaking.com/design_patterns>

-[VALID01] Valid-Computing. <http://www.valid-computing.com/benefits-of-coding-standards.html>