UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERÍA

ESCUELA DE INFORMATICA

IMPLEMENTACIÓN DE UN SISTEMA PARA DISPOSITIVOS MÓVILES. CASO PRÁCTICO: MANEJO DE AUTORIZACIONES REMOTAS EN UNA ENTIDAD FINANCIERA.

Trabajo de titulación previo a la obtención del Título de Ingeniero de Sistemas

Autores:
Hernán Santiago González Toral
Oscar Federico Malgiaritta Morales

Director:
Mabel Méndez Rojas

Cuenca, Ecuador
Junio de 2011

Resumen

Hoy en día las prioridades de los Oficiales de las entidades financieras han ido cambiando de modo que en la actualidad no pueden pasar en sus oficinas a lo largo del día, ya que tienen que salir en busca de clientes y ofrecer sus productos; siendo muy necesario que ellos posean a la mano una herramienta para poder procesar sus autorizaciones desde cualquier lugar de manera rápida, sencilla y segura. Para aquello se desarrollará una aplicación móvil que pueda ser adaptada a cualquier núcleo financiero para que mediante una sencilla funcionalidad de notificación, revisión de información y decisión puedan realizar estos procesos desde el lugar que se encuentren y desde cualquier dispositivo móvil inteligente con acceso a la red de Internet (o Intranet mediante VPN). La aplicación utiliza tecnologías HTML5 y CSS3, es multiplataforma y su funcionalidad puede ser extendida para ser ejecutada dentro de cualquier plataforma de forma nativa.

Palabras Clave:

Sistema, software, aplicación, entidad, financiera, banco, cooperativa, oficial, autorización, móvil, celular, inteligente, blackberry, android, ios, internet, https, seguro, jquery, phonegap, html5, ccss3, jboss, java, notificación, smartphone, email

Abstract

Nowadays, the financial entities Officers priorities have changed so that actually they don’t stay all day in their offices because they have to go out and find new customers that require theirs products. So it’s quite necessary that they need a tool to process authorizations in easy, simple and secure way wherever they are. For that we develop a movil application that can be adapted to a core banking so that by a simple "notification – information verification – decision" functionality, they can make an authorization process on any smartphone connected to Internet (or Intranet using VPN). The application is multiplatform; uses HTML5 & CSS3 technologies and its functionality could be extended for any specific platform for running natively.

Hoy en día las prioridades de los Oficiales de las entidades financieras han ido cambiando de modo que en la actualidad no pueden pasar en sus oficinas a lo largo del día, ya que tienen que salir en busca de clientes y ofrecer sus productos; siendo muy necesario que ellos posean a la mano una herramienta para poder procesar sus autorizaciones desde cualquier lugar de manera rápida, sencilla y segura. Para aquello se desarrollará una aplicación móvil que pueda ser adaptada a cualquier núcleo financiero para que mediante una sencilla funcionalidad de notificación, revisión de información y decisión puedan realizar estos procesos desde el lugar que se encuentren y desde cualquier dispositivo móvil inteligente con acceso a la red de Internet (o Intranet mediante VPN). La aplicación utiliza tecnologías HTML5 y CSS3, es multiplataforma y su funcionalidad puede ser extendida para ser ejecutada dentro de cualquier plataforma de forma nativa.

Tabla de contenido

Lista de Figuras

Lista de tablas

UNIVERSIDAD DE CUENCA

HERNÁN SANTIAGO GONZÁLEZ TORAL, reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de INGENIERO DE SISTEMAS. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

HERNÁN SANTIAGO GONZÁLEZ TORAL
0301861340

HERNÁN SANTIAGO GONZÁLEZ TORAL, certifica que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor/a.

HERNÁN SANTIAGO GONZÁLEZ TORAL
0301861340

UNIVERSIDAD DE CUENCA

OSCAR FEDERICO MALGIARITTA MORALES, reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de INGENIERO DE SISTEMAS. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

Oscar Malgiaritta

OSCAR FEDERICO MALGIARITTA MORALES
1712637758

OSCAR FEDERICO MALGIARITTA MORALES, certifica que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor/a.

Oscar Malgiaritta

OSCAR FEDERICO MALGIARITTA MORALES
1712637758

Dedicatoria

Gracias a los Geingsts y a todas y cada una de las personas que colaboraron en la investigación realizada.

A nuestra familia y amigos que sin esperar nada a cambio compartieron pláticas, conocimientos y diversión. A todos aquellos que durante los cinco años que duró este sueño lograron convertirlo en una realidad.

A mi mamá, papá y tios que me educaron con los mejores valores y que siempre me apoyan y creen en mi. A mis amigos Meme, Chulla, Oscar, Pio y mi primo J. Felipe que siempre estuvieron en las buenas y las malas.

Capítulo 1: Estado Del Arte

1.1 Antecedente

Actualmente en el mercado existen varias aplicaciones móviles relacionadas con las entidades financieras, mediante las cuales casi todas las entidades bancarias permiten realizar operaciones sencillas (tales como consulta de saldo, transferencia de dinero, etc.), sin embargo, estás se han centrado únicamente en brindar servicios a los clientes y no precisamente a los funcionarios de la entidad.

Hoy en día, las prioridades de los Oficiales de las entidades financieras han ido cambiando de modo que en la actualidad no pueden pasar en sus oficinas a lo largo del día, ya que tienen que salir en busca de clientes y ofrecer sus productos; por esto es importante que ellos puedan procesar sus autorizaciones desde cualquier lugar de ~~una~~ manera rápida, sencilla y segura, como lo hacen actualmente desde sus oficinas.

Gracias a la existencia y popularidad de los dispositivos móviles inteligentes en los que se pueden instalar distintas aplicaciones y desde los que se puede tener acceso a internet desde cualquier lugar en donde exista señal, es posible integrar una aplicación que permita el control de dichas autorizaciones.

El producto a desarrollar podrá ser utilizado por cualquier núcleo financiero únicamente adaptando el módulo de comunicación con éste. Para el proyecto en cuestión, la aplicación funcionará con el núcleo FIT que la empresa Bantec Inc. ofrece al mercado y que varias entidades financieras ya utilizan. La empresa brinda soluciones orientadas a controlar y mejorar la rentabilidad del negocio financiero, totalmente multicompañía, multimoneda, multidioma; estando lista para soportar el crecimiento de la institución.

El desarrollo de este tema se realizó mediante la investigación en documentos y publicaciones en idioma inglés. Es por ellos que los créditos a sus autores y sus

referencias en su gran mayoría podrán ser revisados en la sección de referencias bibliográficas.

1.2 Justificación

Al crear la aplicación, los oficiales de las sucursales de las entidades financieras podrán controlar las distintas autorizaciones que tienen a su cargo, en cualquier momento y desde cualquier lugar, mediante una conexión segura que permita integrar el aplicativo con el núcleo financiero que utilizan.

De esta forma, los usuarios que dispongan de la aplicación podrán responder a cualquier autorización en tiempo real y recibirán notificaciones cuando se requiera de su participación de modo que puedan procesarse de la manera más eficiente posible. Además, la aplicación será dinámica, pudiendo integrarse con cualquier núcleo financiero.

1.3 Objetivos

1.3.1 General

Implementar una aplicación para dispositivos móviles que permita a los oficiales de las distintas sucursales de una entidad financiera procesar distintas autorizaciones (solicitud de apertura de cuenta, solicitud de chequera, etc.) que les competa mediante una conexión segura a través de Internet.

1.3.2 Específicos

1. Desarrollar una aplicación que funcione en un dispositivo móvil y que permita controlar al menos 10 de las autorizaciones remotas más comunes.
2. Proveer de un generador Web que permita una visualización adaptable de la aplicación en Smartphones con BlackBerry OS 6, iOS 4 y Android 2.3.4.
3. Asegurar una comunicación segura entre los dispositivos móviles y el servidor de la entidad financiera mediante el registro de los dispositivos de los oficiales.

Capítulo 2: Introducción

2.1 Dispositivos móviles

Hoy en día las aplicaciones para dispositivos móviles se las puede realizar de dos formas: aplicaciones nativas, que son programas compilados que se ejecutan en el dispositivo nativamente, y aplicaciones web móviles, que se ejecutan dentro de los navegadores web del dispositivo móvil.

Las aplicaciones nativas son las más usadas estos días, especialmente por el éxito financiero que las empresas han tenido en la tienda de Apple iTunes Store y la tienda de Android. Además éstas brindan varias ventajas como son su rapidez y el acceso a todas las capacidades que la plataforma brinda. Pero también poseen ciertas limitaciones importantes, entre ellas, no son portables; así, si se quiere que una aplicación esté disponible para múltiples plataformas, se debe escribir en múltiples lenguajes, dando como resultado, el tener que mantener varios códigos de la misma aplicación.

Las aplicaciones web móviles en otro lado, son creadas utilizando HTML5, CSS3 y librerías JavaScript, que se ejecutan en los navegadores web del dispositivo móvil, esto significa que solo se debe mantener un código fuente, pero en este tipo de aplicaciones aún se necesita tener en cuenta las variaciones de los navegadores entre plataformas.

Los últimos navegadores móviles están más cercanos a sus similares de escritorio, pueden mostrar páginas web estándar a escala, permite al usuario realizar zoom sobre ella, cambia la forma de mostrar la página al momento de rotar ciertos dispositivos y tiene soporte para las últimas tecnologías web como CSS3, HTML5, AJAX, y su implementación es basada en la plataforma estándar Web Kit. Sin embargo, aún sigue siendo dificultoso el uso de ellos, sobretodo en temas de navegación. El navegador ocupa más recursos de batería, su ejecución es más lenta, y simplemente no puede aprovechar toda la tecnología del dispositivo como cuándo ejecutamos una aplicación nativa.

Una solución genérica a estos dos tipos de punto de vista que se puede tomar, es la utilización de una capa de abstracción de plataforma, mezclada con una librería para interfaces que funcione en el mayor rango de dispositivos existentes. Así se tendrá un solo código de aplicación que mantener, podrá ser ejecutada de forma nativa en distintos dispositivos, con acceso a ciertas capacidades que las plataformas brindan, y su visualización se adaptará a la resolución que el móvil posea. A este tipo de aplicaciones se las conoce como Widgets y son aplicaciones de único propósito; basadas en tecnología Web, realizan una tarea en particular pudiendo acceder a las características específicas del dispositivo como cámara o lista de contactos.

Entre las ventajas que brindan los widgets están: [1]

- La capacidad de ejecutarse como una aplicación nativa en el móvil sin necesidad del uso del navegador web.
- Es mucho más fácil para el usuario entender e interactuar con widgets.
- Se puede acceder a los recursos del dispositivo móvil.
- Un widget puede almacenar en cache cierta información para reducir la necesidad de llamadas Ajax para traer dicha información.
- Se puede construir interfaces de usuario atractivas e intuitivas y animaciones, que permiten una interacción más sencilla entre el usuario y la aplicación. Estos widgets se desarrollan mediante el uso de HTML5, CSS3 y librerías JavaScript, es decir, permiten el uso de la última tecnología existente para el desarrollo de aplicaciones web.

Aunque los navegadores ya se estén basando en estándares, aún se debe tener en cuenta una cierta adaptación de la aplicación móvil en el lado del servidor. Se pueden tener variaciones en tamaños de pantalla, métodos de entrada (interfaz) y velocidades de red, que pueden ser resueltas mediante métodos de adaptación en el servidor. Algunas formas de implementar esto es mediante el uso de WURFL, que es un estándar para la detección e información de dispositivos móviles.

2.2 Limitaciones

Es complicado lograr facilidad de uso en los dispositivos móviles, no sólo porque los usuarios se encuentran en otro contexto cuando acceden a aplicaciones web, si no porque estos dispositivos tienen sus propias limitaciones. Los dispositivos móviles no son computadoras, y además de esto cada uno de ellos es diferente. Esto requiere que, al momento de realizar una aplicación web, además de probar con varios

exploradores, emuladores y dispositivos; tengamos que estar dispuestos a escuchar a los usuarios para hacer ajustes cuándo se necesite.

Algunas de las limitaciones más importantes que podemos encontrar en los dispositivos móviles son: [1]

- *Tamaño de la pantalla*: Tienen un tamaño de entre 120x120 pixeles y 320x240 pixeles. Esto quiere decir que se pueden mostrar en promedio sólo 6 líneas de 25 caracteres.
- *Métodos de entrada*: Aunque hay dispositivos que poseen un teclado QWERTY, existe gran cantidad de estos en dónde sólo disponemos de un teclado numérico.
- *Procesamiento*: La mayoría de estos dispositivos tienen procesadores que no pueden realizar cálculos complicados. A veces incluso la RAM de estos no es suficiente para cargar una página web completa.
- *Soporte de formatos*: La cantidad de formatos de imágenes, y formatos multimedia que soportan los dispositivos es limitada. La mayoría de ellos, no soportan GIFs animados.
- *Renderizado*: Dependiendo del explorador que estemos utilizando, una página web puede renderizarse de distintas formas. Algunos, modificarán todo el formato y otros, encogerán la página para que se pueda ver.

2.3 Estándares [2]

Hoy en día, la tecnología de la Web se ha vuelto muy poderosa, es por esto que a diario, podemos hacer uso de aplicaciones mucho más complejas que hace poco sólo funcionaban en dispositivos de escritorio. A continuación, se describen algunas de las características que han hecho que esto sea posible.

Gráficos

Los gráficos son una parte muy importante en cualquier aplicación Web. En la actualidad, lo mejor es utilizar estructuras SVG (Gráficos Vectoriales Escalables), ya que al hacer uso de estas, podemos redimensionar nuestras ilustraciones (sin perder calidad) y animarlas (permitiendo la creación de interfaces mucho más avanzadas). Además, la integración de vectores en HTML5, permite crear filtros, los mismos que facilitan el trabajo al momento de generar contenido multimedia.

Tanto SVG, cómo HTML5 pueden ser estilizados. Para ello, lo único que tenemos que hacer, es usar hojas de estilo (CSS), que permiten implementar una gran

cantidad de efectos gráficos (esquinas redondeadas, imágenes de fondo complejas, sombras, contenidos rotados, animaciones y hasta efectos en 3D).

Además de lo descrito anteriormente, también es importante tener en cuenta la fuente que se va a utilizar en la aplicación, ya que en dispositivos móviles la cantidad de 'tipos de letra' no es muy amplia.

Multimedia

Gracias a HTML5, se ha podido lograr una gran mejora al momento de crear contenido multimedia en la Web. Esto se ha debido principalmente a la inclusión de 2 etiquetas (<audio> y <video>), las mismas que permiten acceder a este tipo de contenido sin necesidad de aplicaciones/plugins adicionales.

Además de las etiquetas nombradas anteriormente, gracias a HTML5, se ha desarrollado una API para la edición de imágenes (Canvas 2D Context), que se podría incluso utilizar para editar vídeos.

Adaptación de dispositivos

Los dispositivos móviles, no sólo se diferencian de los dispositivos de escritorio, éstos, también tienen muchas diferencias entre sí ya que muchas de sus características pueden variar (tamaño de la pantalla, resolución, tipo de teclado, capacidad de almacenamiento, etc.).

Gracias al uso de CSS Media Queries, hoy en día es posible adaptar la disposición de nuestra aplicación Web, a casi cualquier dispositivo. Haciendo uso de esta tecnología, podemos definir que estilo (CSS) utilizar, dependiendo de las características de nuestro dispositivo.

Formularios

La creación de formularios es muy importante cuándo necesitamos que el usuario interactué con algún servidor externo. Debido a que los teclados de los dispositivos móviles son muy limitados, es necesario hacer uso de ciertos controles que permitan optimizar la relación entre el usuario y el servidor. Es por esto, que siempre es bueno hacer uso de calendarios, formateadores, patrones, validaciones, pistas, etc.

Interacciones con el usuario

En la actualidad, cada día es más común el uso de dispositivos táctiles. Por esto, un criterio muy importante al implementar aplicaciones para dispositivos móviles, es tener en cuenta la creación de interfaces bien adaptadas, que requieren la definición de eventos (táctiles) en el DOM (Modelo de Objetos de Documentos).

Almacenamiento de información

Uno de los componentes más críticos de las aplicaciones Web es su capacidad de guardar estados, exportar contenidos e integrar información de otros archivos y servicios. Afortunadamente, en la actualidad, podemos hacer uso de dos mecanismos para almacenar información durante nuestra sesión: el localStorage que almacena la información de manera indefinida (permite su eliminación usando JavaScript o borrando la caché desde el navegador), y el sessionStorage que guarda la información sólo durante la sesión (la información se elimina cuándo se cierra el navegador).

Optimización y Rendimiento

Debido a que los dispositivos móviles tienen una batería limitada y a que su procesador no es muy potente, siempre es muy importante tener en cuenta el desempeño de nuestras aplicaciones. Para ello, disponemos de algunos tipos de optimizaciones, entre las que están el tiempo de navegación, el tiempo de recursos, el tiempo de rendimiento, el tiempo de usuario, la API para la visibilidad de páginas, la API para control de batería, entre otras.

2.4 Seguridad [3]

A continuación, vamos a tratar dos temas que nos van a ayudar a realizar aplicaciones móviles seguras. En primer lugar, explicaremos los mayores problemas que encontramos hoy en día en los dispositivos móviles. Y luego daremos algunos consejos para mitigar estos problemas y desarrollar aplicaciones seguras.

2.4.1 Mayores problemas de seguridad

- *Seguridad Física*: Los dispositivos móviles pueden ser robados, o simplemente podrían perderse. Al hablar de hardware la pérdida no es muy grande, sin embargo, información muy importante podría llegar a las manos equivocadas.

- *Almacenamiento de Datos Seguros*: Este problema está muy relacionado con el anterior. Lamentablemente en los dispositivos móviles toda la información se encuentra almacenada localmente. Sin embargo, este no sería un problema, si lográramos que nuestra información sea inaccesible a personas no autorizadas.

- *Autenticación fuerte con teclados pobres*: Para lograr una autenticación fuerte, nuestra contraseña debe contener una combinación de letras (al menos una mayúscula), números, símbolos especiales y un espacio. Y respetar los requisitos de una autenticación fuerte es imprescindible, en especial cuando necesitamos acceder a información sensible (como una cuenta bancaria). Sin embargo, lograr esto con los teclados de los dispositivos móviles es muy difícil, ya que es más complicado introducir información.
- *Soporte para varios usuarios*: Los sistemas operativos tradicionales permiten el acceso por parte de varios usuarios, mostrando siempre un entorno diferente. En los dispositivos móviles, esto no es así. Pero sería importante, poder separar nuestras aplicaciones, de modo que las personales, no se mezclen/confundan con las utilizadas en nuestro trabajo.
- *Entorno de navegación seguro*: Uno de los mayores riesgos cuando usamos dispositivos móviles, es el comportamiento de navegación del usuario, y esto, trae varios problemas. Un problema fundamental, es la falta de espacio de visualización en el dispositivo móvil, que permite que un estafador nos engañe de manera mucho más simple (Por ejemplo, la incapacidad de ver una dirección de Internet completa, podría hacer que accedamos a un sitio no deseado).
- *Sistemas operativos seguros*: Asegurar un sistema operativo no es simple, pero cualquier software para dispositivos móviles debe realizar esta tarea. La seguridad, en estos dispositivos puede relacionarse tanto con la pérdida de información, cómo con el tiempo de inactividad del sistema. Siempre hay que tener en cuenta cuándo aplicar dicha seguridad puesto que el uso excesivo de la misma puede debilitar la experiencia del usuario.
- *Aislamiento de aplicaciones*: Una de los principales usos de los dispositivos es realizar llamadas telefónicas. Sin embargo, hoy en día, la instalación de aplicaciones se ha vuelto fundamental. La capacidad de aislar las aplicaciones, y la información que estas requieran, es uno de los pasos más importantes para lograr que un juego, por ejemplo, no tenga acceso a información de una aplicación de nuestra empresa.

- *Divulgación de información*: El tema de la información ya se ha tocado anteriormente a lo largo de este documento, sin embargo es importante mencionarla cómo una categoría aparte. El hecho de que la información almacenada en un dispositivo, es más importante que el dispositivo mismo, no es algo nuevo en dispositivos de computación. En los dispositivos móviles, esto afecta mucho más, ya que son mucho más fáciles de perder, y no sólo podríamos exponer información sensible, ya que se podría incluso permitir el acceso a una red interna a través de una VPN.

- *Virus, gusanos, troyanos, spyware y malware*: Como cualquier dispositivo que tiene acceso a Internet, en los dispositivos móviles es importante controlar la filtración de software maligno. Afortunadamente, debido a la gran experiencia que se ha obtenido con los computadores de escritorio, sólo se requiere migrar las formas de control ya existentes.
- *Uso de SSL*: El uso de SSL es indispensable para obtener una aplicación segura. Sin embargo, con los dispositivos móviles, podemos llegar a tener problemas. El principal problema se da, porque algunos dispositivos no tienen la suficiente capacidad de procesamiento cómo para soportar SSL sin afectar la experiencia del usuario. Es decir, que si queremos que nuestra aplicación llegue a todos los usuarios, tendremos que permitir que algunos usuarios hagan uso de SSL y otros no.
- *Cross-Site Request Forgery (CSRF)*: El CSRF es un tipo de ataque que normalmente afecta a aplicaciones web, y que permite que el atacante modifique la información de la víctima. Para que el ataque funcione, la víctima sólo tiene que abrir un enlace, que además de abrir la página deseada, envía peticiones web a otra aplicación, permitiendo el acceso a la misma, sin que el usuario lo sepa.
- *Privacidad de la ubicación*: Todos los usuarios quieren privacidad, sin embargo mucha de esta desaparece cuándo por ejemplo accedemos a una aplicación cómo Google Latitude. Esto se debe a que con los dispositivos móviles y el uso de GPS, cualquiera podría llegar a saber nuestra ubicación actual si no hemos configurado adecuadamente nuestro dispositivo.
- *Drivers inseguros:* Si la estructura de nuestra aplicación fue diseñada adecuadamente, esta no debería poder acceder al sistema operativo del dispositivo, pero los drivers sí. Es por esto que hay que tener cuidado al momento de hacer uso de estos, y asegurarnos de que son seguros.

- *Autenticación multifactorial (MFA)*: La MFA es fuertemente requerida en dispositivos móviles. Esta consiste en asegurarnos de quién está ingresando a la aplicación de acuerdo al explorador, dirección de acceso y a las cabeceras HTTP. Para lograr esto, se hace uso de una firma, que contiene toda la información anterior, ésta firma, se recalcula cada vez que el usuario trata de ingresar a la aplicación, y se compara con la firma almacenada en la base de datos del servidor; si la información no cuadra, el usuario tendrá que seguir una secuencia de pasos (Enviar un e-mail, SMS o realizar una llamada) para confirmar que se trata de él. Sin embargo, debido a que la información que los dispositivos móviles proveen no es muy amplia, es muy difícil hacer uso de una autenticación multifactorial robusta. Esto es un problema, ya que si un atacante logra comprometer los datos de autenticación a una aplicación, no le va a ser muy difícil engañar a la MFA.

2.4.2 Consejos para el desarrollo de aplicaciones móviles seguras

Lamentablemente, la forma en que se desarrolla una aplicación web segura, depende de la plataforma que estemos usando (Android, iPhone, BlackBerry, Symbian, JME, WinMobile). De todos modos, existen ciertas guías básicas, las mismas que se explican a continuación.

- *Aprovechar TLS/SSL*: La solución más simple siempre es la mejor. Es importante hacer uso de TLS o SSL por defecto, y requerir su uso a lo largo de toda la aplicación.
- *Seguir prácticas de programación seguras*: La mayoría de aplicaciones móviles son escritas en C, C++, C# o Java. Si se hace uso de estos lenguajes, tenemos que aprovechar de la gran investigación que se ha hecho sobre seguridad hasta el momento, y hacer uso de prácticas de programación seguras.
- *Validar datos de entrada*: Sea cuál sea nuestra aplicación, la validación de los datos siempre es imprescindible.
- *Aprovechar el modelo de permisos utilizados por el sistema operativo*: Hoy en día el sistema operativo de varios celulares incluye ya un módulo de permisos, y es importante aprovecharlo al máximo.
- *Utilizar el modelo de privilegio mínimo para acceder a la aplicación*: Este consiste en hacer que la aplicación tenga acceso únicamente a lo que necesite. Es decir

que tenemos que enumerar los servicios, permisos, archivos y procesos que nuestra aplicación requiera, y limitar la aplicación para que sólo pueda usar estos.

- *Almacenar información confidencial correctamente*: Nunca se debe almacenar información sensible (usuario, contraseña) en archivos de texto planos. Algunos dispositivos (Android, iPhone) ofrecen métodos de encriptación, y deberíamos aprovechar de estos.
- *Firmar código de la aplicación:* Aunque firmar el código no hace que este sea más seguro, este permite que los usuarios estén seguros de que la aplicación ha seguido las prácticas requeridas por la tienda de aplicaciones.
- *Construir un proceso de actualización seguro y fuerte*: La idea es desarrollar un proceso mediante el cual sea posible actualizar nuestra aplicación de forma rápida, fácil y sin requerir de mucho ancho de banda.
- *Comprender los puntos fuertes y las limitaciones en seguridad del navegador móvil*: Al armar una aplicación que va a hacer uso de un explorador, es importante tener en cuenta que limitaciones existen (cookies, cache) y que aspectos de seguridad poseen (algunos implementan una característica que sólo permiten lectura).
- *Uso de URL seguras e intuitivas*: Al hacer uso de estas podemos evitar el phishing, cuando hacemos aplicaciones móviles siempre es bueno usar el prefijo "m.".

2.5 Mejores prácticas [4]

La w3c define un documento de recomendaciones para las mejores prácticas en el desarrollo de aplicaciones web móvil. El objetivo de este documento es ayudar al desarrollo de aplicaciones web móvil enriquecidas y dinámicas. Conlleva las prácticas más relevantes en la ingeniería, promoviendo las que permiten una mejor experiencia de usuario y a su vez, prevé contra aquellas que son consideradas perjudiciales. Aunque el enfoque del documento son las mejores prácticas para aplicaciones que se ejecutan en un navegador, en muchos casos estas recomendaciones son igualmente aplicables en otros tipos de ejecución web, como las aplicaciones denominadas Widgets. Las mejores prácticas asumen que los dispositivos tienen soporte para XHTML estándar, JavaScript y capacidades CSS.

2.5.1 Datos de la aplicación.

Uso escaso de cookies.

Las cookies son un método efectivo para el almacenamiento corto de información de estado en el cliente. Estas son comúnmente utilizadas para almacenar tokens o claves que representan una identidad de usuario para permitir un inicio de sesión automático. La información almacenada en las cookies son enviadas al servidor en cada petición realizada, y el uso excesivo de ellas podría causar un impacto negativo, particularmente en las redes móviles. A su vez, en el contexto móvil, el soporte de cookies puede estar deshabilitado ya sea en la configuración del dispositivo o en la red móvil. Por esta razón, las aplicaciones deben ser funcionales aun cuando las cookies no estén disponibles.

Uso apropiado de las tecnologías de almacenamiento de datos locales en el cliente.

Si el dispositivo tiene soporte, las APIs de almacenamiento de información en el lado del cliente proveen un mecanismo para guardar cantidades más extensas de datos. El uso de esta técnica brinda una paridad entre las aplicaciones Web y las aplicaciones nativas en términos tiempos de inicio de respuesta. Los datos de la aplicación almacenados localmente pueden ser mostrados de manera inmediata cuando la aplicación es iniciada, pudiendo continuar su operación normal mientras la señal de red no sea fiable y no se puedan realizar actualizaciones replicando cambios desde el servidor mediante un proceso en segundo plano.

Replicar datos locales al servidor si es necesario.

Si una API de almacenamiento en el lado del cliente utiliza datos que no son visibles en otros dispositivos del usuario, es apropiado para algunas formas de datos (por ejemplo preferencias, estados) enviar éstos datos de regreso al servidor para proveer una vista consistente entre dispositivos y hacer posible la recuperación de datos si el dispositivo se pierde o daña.

2.5.2 Seguridad y Privacidad.

No ejecutar datos JSON no confiables.

Una técnica común de transmisión de datos es el uso de JSON, donde el cliente utiliza la funcióneval() de JavaScript para parsearlo, esta es en una técnica muy poderosa, sin embargo, la ejecución directa sobre datos que contienen código

generado, representan un riesgo de seguridad que se puede tornar muy peligroso en dispositivos móviles, puesto que la información personal podría estar expuesta. Para evitar esto, es mejor utilizar una implementación de un parser, ya que no es posible asegurarse de que los datos dentro de un JSON estén correctamente formateados.

Control y consciencia al usuario.

Permite al usuario controlar el comportamiento de la aplicación que podría ser no tan aparente, como el acceso a la red o a los datos del dispositivo (por ejemplo imágenes, contactos, ubicación, conectividad)

Es preferible confiar en las funcionalidades nativas del navegador para notificar al usuario sobre estas actividades, sin embargo, las mejores prácticas promueven el aviso sobre estas actividades dentro del comportamiento de la aplicación, en situaciones donde la funcionalidad nativa no es suficiente.

Asegurarse que el usuario esté informado acerca del uso de información personal y del dispositivo.

Se recomienda asegurarse que el usuario esté informado si la aplicación necesita algún acceso a la información personal o del dispositivo. Estos avisos se pueden proveer cuando el usuario accede por primera vez a la aplicación, o en el primer acceso a la información, dando la suficiente información para que el usuario juzgue en dar o no permisos. En muchos casos es mejor el uso de las APIs que proveen acceso a la información dentro del dispositivo, mediante el uso de interfaces nativas para que sea claro para el usuario, y se lo realice una sola vez.

Permitir la autenticación automática.

Si una aplicación requiere la identificación del usuario es usual presentar al usuario una interfaz para el ingreso de las credenciales y la opción de activar la autenticación automática en siguientes usos de la aplicación. Esto es muy importante en dispositivos donde la entrada de datos es un poco dificultosa. Hay que tener en cuenta que si la autenticación automática es habilitada, se debe proveer un enlace para cerrar sesión.

Las credenciales pueden ser almacenadas en una cookie o en almacenamiento local. Sin embargo, hay que tener en cuenta no almacenar la información de contraseña sin encriptar. Comúnmente un token generado mediante un hash seguro,

que podría ser revocado en el servidor, es almacenado localmente para permitir así el inicio de sesión automático.

2.5.3 Uso Apropiado de Recursos

El uso mínimo de memoria, procesador del dispositivo y ancho de banda de la red asegura una ejecución liviana y de baja latencia de la aplicación.

Utilizar compresión en la transferencia

La compresión en HTTP 1.1, que utiliza los algoritmos gzip y DEFLATE, es muy soportada. Los servidores Web pueden ser configurados apropiadamente para enviar respuestas comprimidas. Sin embargo, el costo (en tiempo y uso de batería) en descomprimir los datos puede ser balanceados contra la ganancia en eficiencia en transporte. Mientras se configura la compresión en HTTP 1.1 hay que tomar en cuenta:

- Varios formatos de imagen (JPEGs) no se benefician de la compresión, pero SVG si lo hace.
- Varios otros formatos de medios (por ejemplo audio, vídeo) no se benefician de la compresión.
- Archivos muy pequeños (por ejemplo <1k) generalmente no se benefician de la compresión.

Minimizar el tamaño de datos y de la aplicación.

Las pequeñas aplicaciones se descargan y ejecutan más rápidamente y de forma más eficiente. Es una buena técnica el procesamiento de archivos HTML, JS y CSS para remover espacios en blanco y ser minificados antes de su envío.

Evitar redireccionamientos.

El redireccionamiento de peticiones es utilizado para el intercambio de información entre servidores (por ejemplo autenticación). El retraso dado por las redirecciones es mucho más alto sobre las redes móviles, por lo que el número de redireccionamientos se debe mantener al mínimo.

Optimizar las peticiones de red.

Establecer las conexiones necesarias para poder completar una petición HTTP puede tornarse muy larga en una red móvil. Ya que el ancho de banda es más

restringido en este tipo de redes, es preferible hacer pocas y más largas peticiones. Se deben considerar las siguientes posibilidades al momento de diseñar una aplicación:

- *Peticiones en lote*: una petición para más datos provee una mejor experiencia de usuario, que peticiones pequeñas. Cuando sea posible, enviar en lote múltiples peticiones a nivel de aplicación.
- *Acelerar peticiones de baja prioridad*: en algunas aplicaciones ciertas peticiones pueden ser menos críticas que otras (por ejemplo peticiones de inicio de sesión). Acelerar las peticiones de baja prioridad asegura que éstas no bloqueen la red y así prevenir que peticiones más críticas sean atendidas rápidamente.
- *No atender en periodos de inactividad*: si la aplicación solicita actualizaciones, se debe monitorizar la actividad de usuario y así no enviar actualizaciones en periodos de inactividad.
- *Contexto del dispositivo*: verificar la conectividad del dispositivo para seleccionar un apropiado nivel de interacción.

Minimizar el uso de recursos externos.

Una aplicación usualmente requiere un número de recursos (hojas de estilo, scripts, imágenes, etc.) las cuales a su vez requieren una petición HTTP. Las peticiones HTTP en una red móvil son particularmente costosas, por lo que pocas y largas peticiones son consideradas mejores en lugar de pequeñas peticiones. Para llevar a cabo esto, se puede utilizar alguna implementación de minificación que combine todos los recursos de su mismo tipo y las una como un solo antes de ser enviado.

Agregar imágenes estáticas en un recurso simple compuesto o Sprite.

Las aplicaciones usualmente dependen de un número de imágenes estáticas para proveer íconos, botones, etc., lo mejor es utilizar técnicas CSS y crear Sprites para tener un solo recurso compuesto a ser transferido. Es eficiente combinar imágenes de tamaños y colores similares.

Incluir imágenes de fondo dentro de hojas de estilo CSS.

Efectos visuales como imágenes de fondo o gradientes pueden ser incluidos en el CSS (utilizando el esquema URI: url(' [data])) como string en base64 para evitar una petición HTTP por aquellos. La codificación base64 añade cerca de un 10% más de espacio al tamaño de la imagen después de la compresión

gzip y este costo adicional es muy beneficioso contra los beneficios de una petición por los recursos no comprimidos.

No enviar información de cookies innecesarios.

Recursos estáticos no necesitan de cookies, por lo que el desempeño puede ser mejorado si se los trae desde una ruta o un subdominio en donde las cookies estén deshabilitadas.

Mantener un tamaño razonable del DOM.

El tamaño en memoria del DOM (Document Object Model) puede ser limitado en los dispositivos móviles. Se debe limitar la cantidad de información de éste utilizando paginación u otras técnicas apropiadas.

2.5.4 Experiencia al Usuario

La experiencia al usuario viene dado por un número de factores que incluye: latencia, métodos de interacción y consistencia de datos. Además esta viene fuertemente influenciada por la ejecución inicial de la aplicación. Aplicaciones Web fuera de línea como HTML5 AppCache brinda a las aplicaciones Web la posibilidad de ser utilizadas aun cuando la cobertura de red sea intermitente. Los siguientes pasos pueden ser considerados para minimizar el tiempo de inicio de las aplicaciones:

- *Usar tecnología Offline*: AppCache permite a los recursos de la aplicación ser especificadas y almacenadas localmente, para que esta pueda ser iniciada sin necesidad de una petición al servidor.
- *Considerar el particionamiento de scripts largos*: en aplicaciones complejas, el análisis del código JavaScript puede contribuir al costo en tiempo de inicio de la aplicación. Si ciertas funcionalidades son raramente utilizadas, estas pueden ser movidas en scripts separados para ser cargadas bajo demanda, bajando así la cantidad de código que necesita ser analizado al inicio de la aplicación.
- *Usar almacenamiento local*: si es apropiado, se puede almacenar una captura del último estado de la aplicación para que puede ser mostrada inmediatamente en el siguiente inicio sin necesidad de hacer una petición al servidor.
- *Minimizar el número de consultas a almacenamientos locales*: un número considerable de consultas llegaría a afectar el inicio de la aplicación en su latencia. Es recomendable minimizar este tipo de consultas al momento de mostrar por primera vez la aplicación.

Minimización de la latencia.

La reducción de la latencia en un factor importante en el mejoramiento de la usabilidad de la aplicación. Se pueden utilizar una serie de técnicas para minimizar la latencia:

- *Permitir una renderización incremental*: colocar el código JavaScript al final de la página y configurar la página para que la información útil que pueda estar disponible y sea visible mientras el contenido principal de la aplicación sigue su carga.
- *Mantener al usuario informado de la actividad*: utilizar barras de progreso para mantener al usuario informado durante la carga de red y APIs del dispositivo, para que así no piense que la aplicación se ha detenido.
- *Evitar recargas de páginas*: para reflejar cambios de estado o mostrar vistas distintas dentro de la aplicación, es mejor actualizar las páginas dinámicamente (manipulando el DOM) en lugar de haciendo una recarga.
- *Precargar siguientes probables vistas*: precargar datos de ubicaciones utilizadas frecuentemente en la aplicación, permite que esta pueda ser mostrada más rápidamente cuando el usuario realiza una petición a ellas.

Diseñar para múltiples métodos de interacción.

Los métodos de interacción varían entre dispositivos. Se pueden considerar 3 métodos principales de interacción al diseñar la interfaz de usuario:

- Basado en foco: el navegador se enfoca y salta de elemento en elemento.
- Basado en puntero: la navegación basada en teclas controla el puntero que puede cubrir cualquier parte de la pantalla.
- Basado en táctil: los eventos son relacionados directamente al toque y posición de un dedo en la pantalla.

La configuración óptima de los elementos de la interfaz depende del método de interacción utilizado por el dispositivo. Idealmente, la interfaz de usuario podría ser adaptada basándose en el conocimiento de los métodos de interacción usados por el dispositivo. Si no es posible, la interfaz debe ser diseñada para proveer una buena experiencia para cada una de los distintos métodos.

Capítulo 3: Programación de aplicaciones móviles

3.1 Enfoques de Desarrollo Nativo, Web, Widget [5]

Hoy en día la mayor parte de plataformas móviles, o por lo menos las más utilizadas (iOS, Android, Blackberry OS) nos proporcionan 3 enfoques o formas de desarrollo de aplicaciones para sus dispositivos dependiendo del alcance que los desarrolladores tengan para estas. Dichos enfoques son los siguientes: Desarrollo para navegadores web, desarrollo Widget y desarrollo nativo. Cada enfoque posee sus ventajas, desventajas y similitudes. La elección de uno de ellos para el desarrollo de una aplicación depende de ciertos factores tantos profesionales, como de requerimientos y alcance del aplicativo.

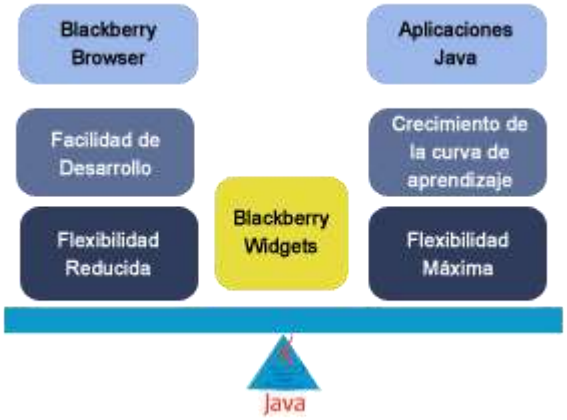


Figura 3-1: Ejemplo de plataforma BlackBerry

En definitiva, la manera de escoger el tipo de enfoque de desarrollo de una aplicación móvil se puede basar en las restricciones que se hallen en el momento del diseño y planificación, como las siguientes:

- Recursos físicos.
- Conocimiento técnico.
- Requerimientos.

- Factor financiero.

Una forma de visualizar esto, es mediante el análisis del Triángulo para el Manejo de Proyectos, una herramienta gráfica donde se toman en cuenta los factores de costo, alcance y planificación, Opuestas unas de otras, y en donde se puede realizar un análisis de la solución de las metas de un proyecto en base a las restricciones que esta posea.

Figura 3-2: Triángulo para el manejo de proyectos

3.2 Desarrollo para Blackbery

La plataforma BlackBerry nos proporciona 3 formas de desarrollo de aplicaciones para sus dispositivos dependiendo del alcance que los desarrolladores tengan para estas, pudiendo realizar acciones sobre ellas como la entrada y búsqueda de datos compatibles con un subprocesamiento múltiple, internacionalización, comunicación de red y el almacenamiento local de datos. Las aplicaciones pueden comunicarse con las redes mediante conexiones estándar TCP y HTTP, a pesar de la red inalámbrica subyacente. Además, los desarrolladores también pueden crear aplicaciones integradas por completo con los programas principales del dispositivo BlackBerry, como la lista de mensajes, las aplicaciones de organizador, el teléfono y el explorador, para una experiencia de usuario perfecta. En definitiva, Blackberry nos proporciona 3 enfoques para el desarrollo de aplicaciones en su plataforma: Browser, Widget y Java Nativo. [6]

Estos enfoques de desarrollo poseen ciertas similitudes, como el acceso a la Red, seguridad (Internet Services, Enterprise Server) y almacenamiento fuera de conexión (aunque la implementación en cada caso difiere). A continuación se expone las

fortalezas y limitaciones de cada una de ellas:

3.2.1 Enfoques de desarrollo Java, Web, Widget [5]

Desarrollo para browser Blackberry: entre sus características posee soporte estándar para Ajax, CSS y HTML, navegación por pestañas, enviar datos de forma proactiva a los usuarios y acceso a la ubicación del usuario mediante GPS.

- Fortalezas:
  - Menor tiempo de desarrollo.
  - Costo cero en el despliegue de la aplicación.
  - Toda la aplicación se encuentra en el servidor.
  - Integración con smartphones: teléfono, email, mapas.
  - Expone algunos puntos claves de Blackberry: conexión constante, seguridad

- Limitaciones:
  - Contenido almacenado en el servidor: consumo de tiempo en el pedido de información.
  - Uso de Requerimientos de interfaz de usuarios avanzados: menús personalizados.

Desarrollo híbrido o Widget: Blackberry Web Works es una API basada en la especificación w3c, la cual hace uso de tecnologías HTML5, CSS3 y JavaScript, y donde el desarrollador puede optar por construir aplicaciones con tecnología web, además de hacer uso de las capacidades nativas del dispositivo mediante el uso de extensiones. Una extensión JavaScript es código construido en una aplicación Blackberry Web Works en el que es posible el paso de parámetros y el manejo del retorno de valores, permitiendo así el acceso a las funcionalidades de la API de Blackberry como el acceso al sistema de archivos, disparando notificaciones, etc., ya que este tipo de aplicaciones se ejecutan dentro de un contenedor administrable de la API de Blackberry. El modelo de distribución/mantenimiento es similar al de las aplicaciones nativas, y además Web Works da apertura a la construcción de nuestras propias extensiones personalizadas.

- Fortalezas:
  - Todas las del desarrollo browser, y no posee ninguna debilidad.
  - SDK no enlazado con el sistema operativo del dispositivo.
  - Existencia de herramientas para el desarrollo.
- Consideraciones:
  - Las extensiones también necesitan del desarrollo de cierto código Java para su enlace, por lo que no es un enfoque 100% libre de código nativo.

- La aparición de problemas en las extensiones puede resultar poco agradable: debug de clase sobre clase.
- Blackberry Web Works se encuentra disponible en versiones de OS 5.0 y superiores.

Desarrollo Nativo Java ME: la API de Blackberry es basada en código Java, donde el dispositivo posee una versión de la JVM (Java Virtual Machine) y su núcleo también es escrito en el mismo lenguaje. Entre sus ventajas se encuentra el uso directo de servicios de localización, multimedia mejorada (captura de video, streaming), capacidad de personalización de ventanas de texto e imágenes y la integración de un navegador dentro de una misma aplicación.

- Fortalezas:
  - Alto grado de personalización.
  - Rápida ejecución de código: mejor rendimiento.
- Consideraciones:
  - Mayor tiempo en el desarrollo de aplicaciones.
  - Curva de aprendizaje elevada.
  - API atada al Sistema Operativo OS (Desarrollos Widget y Browser se ejecutan en un container).

Después de un análisis de los enfoques de desarrollo que nos presenta Blackberry, la elección adecuada para el desarrollo de una aplicación se puede basar en los tipos de restricciones que se nos presenta en la captura de requerimientos, teniendo como patrón las siguientes consideraciones:

Restricción	**Elección**
Costo	Widget, Browser
Cronograma y tiempos de culminación	Widget, Browser
Ámbito de la aplicación y alcance	Nativo

Tabla 3-1: Consideraciones para el desarrollo de una aplicación

3.2.2 Principios de diseño para dispositivos BlackBerry [6]

Las aplicaciones diseñadas para dispositivos BlackBerry® deben ofrecer un equilibrio entre la mejor experiencia de usuario posible y una duración larga de la batería. Cuando se diseña una aplicación para dispositivo BlackBerry (y móviles en

general), se deben considerar las diferencias entre dispositivos móviles y ordenadores. Los dispositivos móviles:

- Tienen un tamaño de pantalla más pequeño que puede mostrar un número limitado de caracteres
- Utilizan redes inalámbricas que tienen un período de latencia más largo que las LAN estándar
- Tienen menos memoria disponible
- Tiene una batería que dura menos

Los usuarios de dispositivos móviles no utilizan del mismo modo las aplicaciones en su dispositivo móvil y las aplicaciones en un ordenador. En un dispositivo móvil, los usuarios esperan encontrar la información rápidamente. Por ejemplo, un sistema de gestión de relaciones con clientes (CRM) puede ofrecer una cantidad masiva de información, pero los usuarios sólo requieren una pequeña cantidad de dicha información a la vez. La interfaz de usuario del dispositivo BlackBerry se ha diseñado para que los usuarios puedan realizar las tareas fácilmente y acceder a la información rápidamente. Al momento de diseñar aplicaciones para dispositivos BlackBerry, se debe intentar ser tan coherente como sea posible con otras aplicaciones del dispositivo BlackBerry. Se puede considerar las siguientes directrices:

- Utilizar componentes existentes de la interfaz de usuario de forma que la aplicación pueda heredar el comportamiento predeterminado del componente.
- Seguir el modelo estándar de navegación tanto como sea posible para que los usuarios puedan hacer un uso completo del teclado y la bola de desplazamiento.
- Realizar todas las acciones disponibles desde el menú. Comprobar que las acciones disponibles en el menú son relevantes para el contexto actual de los usuarios.
- Centrarse en las tareas inmediatas de los usuarios. Simplificar la selección y presentación de datos para mostrar sólo la información que los usuarios necesitan en cualquier otro momento.
- Mostrar la información de manera que haga un uso efectivo de la pequeña pantalla.

3.2.3 Soluciones BlackBerry [6]

Los usuarios de dispositivos BlackBerry pueden utilizar tanto BlackBerry Enterprise Server como BlackBerry® Internet Service, o pueden utilizar ambos en el mismo dispositivo. La comprensión de las diferencias entre BlackBerry Enterprise Server y BlackBerry Internet Service y saber para qué tipo de usuarios tiene previsto ofrecer

compatibilidad, es importante, puesto que puede repercutir en los modos de transporte que utiliza y en cómo puede administrar la sincronización de datos.

BlackBerry Enterprise Solution

BlackBerry Enterprise Server forma parte de BlackBerry Enterprise Solution. BlackBerry Enterprise Server existe tras el firewall de la empresa y proporciona un Gateway inalámbrica a los usuarios del dispositivo BlackBerry en una empresa para obtener acceso al correo electrónico de la empresa y los datos del organizador. BlackBerry Enterprise Server también ofrece las siguientes características clave:

- Cifrado de datos y compresión
- Administración de dispositivos BlackBerry y utilidades de control
- Abastecimiento simplificado de la aplicación
- Gateway autenticada para el acceso a la intranet desde BlackBerry Java Application

BlackBerry Internet Service

Los usuarios de dispositivos BlackBerry que no están asociados con BlackBerry Enterprise Server pueden utilizar BlackBerry Internet Service. BlackBerry Internet Service es un servicio de correo electrónico y de Internet para dispositivos BlackBerry diseñado para proporcionar a los usuarios con entrega automática de mensajes de correo electrónico, acceso inalámbrico a archivos adjuntos de mensajes de correo electrónico y acceso a contenidos de Internet. BlackBerry Internet Service incorpora compatibilidad con la conectividad directa HTTP y TCP/IP a Internet desde una aplicaciónde terceros.

BlackBerry MDS

Para permitir que una aplicación BlackBerry® tenga acceso a los recursos ubicados detrás del firewall de la empresa, BlackBerry Enterprise Server incluye BlackBerry Mobile Data System. BlackBerry MDS proporciona proxies HTTP y TCP/IP para una aplicación BlackBerry Java, que permiten al dispositivo BlackBerry comunicarse con la aplicación y servidores Web tras el firewall de la empresa sin software adicional VPN. Las aplicaciones que envían datos mediante BlackBerry Enterprise Server como Gateway pueden capitalizar la conectividad simplificada de la empresa, el cifrado de datos y la compresión. BlackBerry MDS también ofrece una interfaz abierta, permitiendo a las aplicaciones de servidor detrás del firewall de la empresa insertar contenido en aplicaciones del dispositivo BlackBerry.

3.2.4 Firma y Distribución de la aplicación [7]

Antes de poder distribuir una aplicación para dispositivos Blackberry, ya sea Widget o Nativa, es necesario firmar dicha aplicación. Para aquello es necesario obtener las claves de códigos de firma de RIM desde la página https://www.blackberry.com/SignedKeys completando la forma de petición para posteriormente registrar dichas claves. Es necesario recordar el código PIN que se digitó en la petición, ya que será necesaria para el proceso de registro de las llaves. Una vez aceptada, se recibirán 3 archivos de registro (client-RBB-999999999.csi, client-RRT-999999999.csi, client-RCR-999999999.csi) al correo separados con información de su propósito.

El proceso de registro se lo puede realizar utilizando Eclipse IDE (Aplicaciones nativas), o mediante la aplicación que viene dentro de la API Web Works de Blackberry denominada RIM Signing Authority. Para el proceso de demostración se utilizará el último nombrado:

- Guardar los 3 archivos CSI en la siguiente ruta:
- Windows XP: C:\Program Files\Research In Motion\BlackBerry Web Works SDK <versión>\bin
- Windows 7: C:\Program Files (x86)\Research In Motion\BlackBerry Web Works SDK <versión>\bin
- Mac OS: /Developer/SDKs/Research In Motion/BlackBerry Web Works SDK <version>/bin
- Por línea de comando, navegar al directorio bin y escribir la siguiente sentencia:
- java -jar SignatureTool.jar [CSI file name].csi

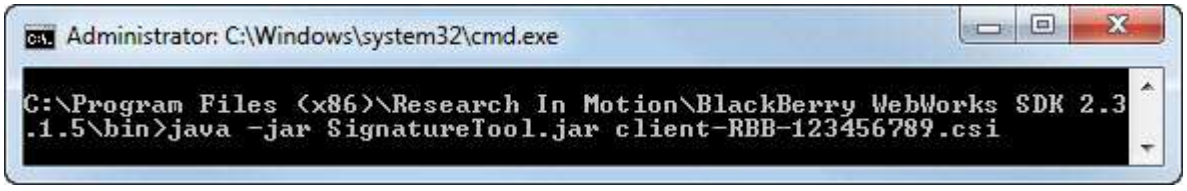


Figura 3-3: Proceso de registro en BlackBerry

- Si aparece un dialogo indicando que una llave privada no pudo ser encontrada, realizar las siguientes acciones:

Figura 3-4: Proceso de registro en BlackBerry

- Presionar Yes.
- En el campo Password, escribir una contraseña de al menos 8 caracteres. Esta será su contraseña que protegerá tu llave privada. Si esta se pierde se debe hace un nuevo registro en RIM, mientras que si es robada se debe contactar a RIM inmediatamente para revocarla.
- En el campo confirmar, se debe re-escribir la contraseña.
- Presionar OK.
- Realizar movimientos con el ratón para la generación de datos aleatorios para la nueva clave privada.
- Posteriormente en el campo de PIN de registro, ingresar el PIN que se envió en la petición de registro de las claves de firma.

Figura 3-5: Proceso de registro en BlackBerry

- Al presionar Register la aplicación intentará registrar su clave. Una vez completado, da presiona Exit. Se recibirá un correo de confirmación del registro

de la clave.

- Se debe repetir el proceso de registro haciendo uso de SignatureTool.jar para los dos archivos CSI restantes.

Después de un registro correcto, al construir las aplicaciones, estas ya estarán firmadas con la clave que no proporcionó RIM, y estarán listas para ser instaladas en un dispositivo o subidas a la App World de Blackberry

3.3 Otras plataformas móviles

3.3.1 iOS [8]

Para desarrollar aplicaciones para esta plataforma es necesario tener a mano el kit de desarrollo de iOS (SDK) y Xcode IDE. Estas herramientas solo se encuentran disponibles para computadores Apple con el Sistema Operativo MacOS. Xcode nos provee todo lo necesario para crear excelentes aplicaciones para iPhone, iPod touch e iPad. Además, para poder firmar y distribuir la aplicación, es necesario formar parte del Apple Developer Program. Este tiene un costo y nos brinda un acceso completo a todos los recursos disponibles en esta.

3.3.1.1 Características de la Plataforma

Los dispositivos basados en el Sistema Operativo iOS comparten varias características únicas demucha influencia en la experiencia de usuario de las aplicaciones que se ejecutan sobre estos.

La pantalla es el corazón de la experiencia de usuario sobre los dispositivos. Esta no solo permite ver al usuario una buena interfaz, sino que también permite la interacción física con su funcionalidad multi toque (Multi-Touch). Además las pantallas de distintas dimensiones y resoluciones pueden tener entre efectos comunes y distintos en la experiencia de usuario con una aplicación, como pueden ser:

- Un confortable tamaño mínimo de elementos de interfaz seleccionables de 44x44 px
- La calidad de arte de una aplicación es muy evidente.
- El enfoque del usuario esta en el contenido.
- Tamaños de pantalla:
- iPhone 4:
- 640 x 960 px

- 960 x 640 px
- iPad
- 768 x 1024 px
- 1024 x 768 px
- Otros dispositivos iPhone y iPod Touch
- 320 x 480 px
- 480 x 320 px
- Se puede cambiar la orientación de la pantalla al rotar el dispositivo, siendo muy útil para las distintas actividades del usuario, y manteniendo el enfoque en la funcionalidad principal de la aplicación:
- En iPhone e iPod touch, la pantalla de inicio es presentada en una sola orientación (vertical). Esto produce a que las aplicaciones en estos dispositivos sean ejecutados en esta orientación por defecto.
- En iPad, la pantalla de inicio es presentada en cualquier orientación, por lo que el usuario espera ejecutar una aplicación con la misma orientación en la que tiene el dispositivo en ese momento.
- Interacción mediante movimientos denominados gestos:
- Tap: presiona o selecciona un control o elemento (como un click del ratón)
- Drag: desplazamiento horizontal o vertical, o para mover un elemento.
- Flick: desplazamiento horizontal o vertical rápido.
- Swipe: realizado con un dedo, puede revelar por ejemplo un botón de Borrar. En iPad con cuatro dedos se puede intercambiar entre aplicaciones.
- Double tap: realiza un zoom hacia dentro o hacia afuera.
- Pinch: con un movimiento de apertura realiza un zoom hacia dentro, mientras que con un movimiento de cierre un zoom hacia afuera.
- Touch and hold: presenta una vista magnificada del texto editable que se encuentra actualmente en el cursor.
- Shake: deshace o rehace una acción.
- Una sola aplicación es visible a la vez. Al intercambiar de aplicación (en iOS 4 y superiores), la aplicación es ocultada de la interfaz y sigue ejecutándose como un proceso de segundo nivel.

3.3.1.2 Tipos de Aplicaciones que se ejecutan en iOS.

Existen dos tipos de software de aplicación que se puede desarrollar para dispositivos basados en iOS:

- Aplicaciones nativas iOS: desarrolladas haciendo uso del SDK de iOS, que residen en el dispositivo y que toman ventaja de las características del entorno de iOS.

- Contenido Web: alojado en un sitio Web que las personas pueden visitar a través de su dispositivo. Este tipo de software puede ser dividido en 3 categorías:
  - Aplicaciones Web: son páginas Web que proveen una solución enfocada en una tarea y que están conformadas por una serie de guías de interfaz que hacen que se comporten de manera similar a una aplicación iOS nativa.
    - Es posible esconder la interfaz de usuario del navegador Safari para que esta se vea más como una aplicación nativa.
    - Utilizando la característica web clip, es posible suministrar un ícono para que sea colocado en la pantalla de inicio para un acceso rápido y similar a la de una aplicación nativa.
  - Aplicaciones Web Optimizadas: son páginas web optimizadas para Safari de iOS y que funcionan como han sido designadas (a excepción de elementos de tecnologías no soportadas como Flash o Java).
  - Aplicaciones Web compatibles: son páginas Web que son compatibles para Safari de iOS y que funcionan como han sido designadas (a excepción de elementos de tecnologías no soportadas como Flash o Java).

Una aplicación iOS puede combinar elementos de interfaz de usuario nativas con acceso a contenido web dentro de un área de vista de contenido web o WebView. Así una aplicación puede verse y comportarse como una aplicación nativa sin que se de atención al hecho de que esta depende de fuentes Web.

En una Aplicación Web se pueden establecer las siguientes propiedades:

- Nombre de Aplicación: para establecer el nombre de la aplicación, se debe cambiar la etiqueta Title:

```
<title>Time</title>
```

- Ícono de aplicación: se debe utilizar un ícono de 57x57 px, mientras que en dispositivos con pantalla Retina debe ser de 114x114 px. Existen dos formas de establecer el ícono:

```
<link rel="apple-touch-icon-precomposed" href="img/icon.png"/>
<link rel="apple-touch-icon" href="img/icon.png"/>
```

- Pantalla de inicio: la imagen debe ser de 320x460 px:

<link rel="apple-touch-startup-image" href="img/splash.png" />

- Para mostrar la aplicación en pantalla completa ocultando la barra de herramientas y de menú que muestran la dirección web, la barra de búsqueda, entre otros:

<meta name="apple-mobile-web-app-capable" content="yes" />

- Se puede establecer el estilo de la barra de estado. Esta barra está ubicada en la parte superior de la pantalla y que muestra información como señal de red o batería. Se tiene las siguientes opciones (por defecto se tiene el gradiente normal gris):

<meta name="apple-mobile-web-app-status-bar-style" content="default" />
<meta name="apple-mobile-web-app-status-bar-style" content="black" />
<meta name="apple-mobile-web-app-status-bar-style" content="black-translucent" />

3.3.2 Android [9]

Esencialmente existen dos formas de desarrollar una aplicación para Android: como una aplicación cliente (desarrollada haciendo uso del SDK de Android e instalada en el dispositivo de usuario como una aplicación .apk) o como una aplicación web (utilizando estándares web y accedida mediante un navegador Web). El enfoque utilizado para una aplicación puede depender de ciertos factores, pero Android hace que dicha decisión.

3.3.2.1 Aplicaciones Web en Android

Para desarrollar una aplicación web Android provee las siguientes características:

- Soporte para las propiedades del viewport que permiten establecer el tamaño de la aplicación Web basado en el tamaño de la pantalla.
- Características CSS y JavaScript que permiten proveer diferentes estilos e imágenes basadas en la densidad en pixeles de la pantalla.

Una gran característica de Android es que no es necesario construir una aplicación pura para el cliente o pura para la web. Es posible mezclar ambos enfoques desarrollando una aplicación cliente que tenga empotrado algunas páginas web haciendo uso de un WebView, donde se puede definir además una interfaz para la ejecución de código JavaScript para hacer llamadas a las APIs de Android. Desde la

versión 2.0 de Android 2.0 se añadieron características del framework de Web Kit, por lo que el navegador Web de Android y un WebView soportan el mismo viewport y densidades de pantalla.

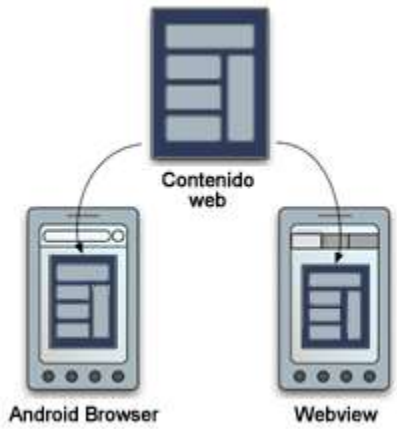


Figura 3-6: Enfoques de desarrollo en Android

3.3.2.2 Firmando aplicaciones en Android [10]

El sistema operativo de Android requiere que todas las aplicaciones que se instalen se encuentren firmadas digitalmente con un certificado que contenga la clave privada del desarrollador. Al firmar la aplicación, el sistema puede conocer el autor de cada aplicación, y de este modo permitir que las relaciones entre distintas aplicaciones se verifiquen.

A continuación se listan algunos puntos que se tienen que tener en cuenta sobre la firma de aplicaciones:

- Todas las aplicaciones deben firmarse: El sistema no permite la instalación de aplicaciones que no estén firmadas. Para probar y depurar una aplicación, las herramientas de Android firman la aplicación con una clave privada especial. Cuando la aplicación esta lista para ser utilizada por usuarios finales, el desarrollador debe firmarla con una clave adecuada. No se pueden publicar aplicaciones firmadas con la clave especial de Android.
- Se pueden utilizar certificados auto-firmados: No se necesita de una autoridad de certificados.
- *El sistema prueba que el certificado no esté expirado sólo al momento de instalar*

la aplicación: la aplicación se podrá seguir utilizando después de que el certificado expire, si esta ya se encuentra instalada.

- Para el firmado de la aplicación, se pueden generar las claves con herramientas comunes como keytool y jarsigner.

Proceso para firmar una aplicación de Android

- Las herramientas de Android firman la aplicación de una manera diferente dependiendo del modo en que se estén ejecutado (modo de depuración o modo de distribución).
- Cuando se genera la llave usando modo de depuración, el SDK de Android usa Keytool para generar una llave de depuración. Cada vez que se depura la aplicación, el SDK de Android se encarga además de hacer uso de Jarsigner para firmar la aplicación. Debido a que el SDK se encarga de todo el proceso, el usuario no tiene que intervenir cuando se realiza este proceso.
- Cuando se compila la aplicación en modo de distribución es necesario usar una llave privada para firmar la aplicación. La llave privada se puede generar utilizando Keytool, y la misma se firmará automáticamente cuando se compile la aplicación. Debido a que los credenciales de la llave las conoce solo el desarrollador, el SDK solicitará tanto el alias como la contraseña.

3.3.2.3 Estrategias para el firmado de aplicaciones

Es recomendable que un desarrollador firme todas sus aplicaciones con la misma llave privada. Esto se debe a:

- *Actualizaciones*: Si la aplicación va a tener nuevas versiones, es importante que todas se firmen con el mismo certificado para facilitar el proceso de actualización. Si una actualización no posee el mismo certificado que la aplicación original, esta se instalará como una aplicación completamente nueva.
- *Modularidad*: El sistema de Android permite que aplicaciones firmadas con el mismo certificado se ejecuten en un mismo proceso. De este modo se podría crear una aplicación modular, de modo que el usuario pueda actualizar/instalar solo los módulos de la aplicación que este requiera.
- *Compartición de información*: Android provee un sistema de permisos basado en la firma de las distintas aplicaciones. De este modo, aplicaciones que tengan una misma llave pueden compartir código y funcionalidad.

Otro aspecto importante que se debe tener en cuenta es el tiempo de validez del certificado:

- Si se planea que la aplicación tenga actualizaciones, habría que asegurarse de que el periodo de validez exceda la vida útil de la aplicación. Se recomienda usar un periodo de 25 años o más pues que si una llave expira el usuario ya no podrá actualizar la aplicación sin problemas.
- Si se planea firmar varias aplicaciones con la misma llave, hay que tener en cuenta que el periodo de validez supere la vida útil de todas las aplicaciones que se vayan a firmar.
- Si se va a publicar la aplicación en Google Play, el periodo de validez debe ser mayor al 22 de Octubre de 2033 pues este es un requerimiento para subir aplicaciones.

3.3.2.4 Pasos para firmar en modo de distribución

Cuándo una aplicación esta lista para ser distribuida se tienen que seguir los siguientes pasos:

Obtener una llave privada: para poder firmar una aplicación, lo primero que se tiene que hacer es obtener una llave privada adecuada. Una llave privada adecuada tiene que:

- Ser de posesión del desarrollador.
- Representar el personal, la corporación o la organización que desarrollo la aplicación.
- Tener un periodo de validez adecuado.
- No ser una llave de depuración generada por Android SDK.

Para generar la llave (si no se posee una), se puede hacer uso de *keytool* usando alguna o varias de las siguientes opciones:

- genkey: Genera una llave pública y una privada.
- v: Habilita el modo verboso.
- alias: Establece un alias para la llave. Tiene que tener máximo 8 caracteres.
- keyalg: Establece el algoritmo que se va a usar para encriptar la llave. Se puede usar DSA y RSA.
- keysize: Establece el tamaño de la llave en bits. Keytool usa un tamaño de 1024 bits pero se recomienda usar 2048 o más.
- dname: Establece un nombre descriptivo para saber quien creó la llave.
- keypass: Establece la contraseña de la llave.
- validity: Establece el periodo de validez de la clave en días.
- keystore: Establece el nombre del archivo que contendrá la llave.

- storepass: Establece una contraseña para el keystore.

Se podría generar un keystore usando el siguiente comando:

keytool -genkey -v -keystore fitnotification.keystore -alias fitnotification -keyalg RSA -keysize 2048 -validity 10000 -keypass password -storepass password

Compilar la aplicación en modo de distribución: para compilar la aplicación en modo distribución usando Eclipse tenemos que dar click derecho en el Explorador de Paquetes y seleccionar Herramientas/Exportar paquete sin firmar. Esto generará un archivo con extensión *.apk*.

Firmar la aplicación con una clave privada: Una vez que tenemos la aplicación sin firmar (archivo .apk), tenemos que firmarla usando Jarsigner. Esta aplicación provee las siguientes opciones:

- keystore: Nombre del keystore (generado con keytool).
- verbose: Habilita el modo verboso.
- sigalg: Estable el nombre del algoritmo que se quiere usar para firmar. Se debe usar “MD5 with RSA”.
- digestalg: Estable el algoritmo que se va a utilizar para procesar las entradas de la aplicación (*.apk*). Se debe usar SHA1.
- storepass: Contraseña del keystore.
- keypass: Contraseña de la llave privada

Para firmar una aplicación llamada fitnotification.apk, con el keystore generado anteriormente podríamos usar el comando:

jarsigner -verbose -sigalg MD5withRSA -digestalg SHA1 -keystore fitnotification.keystore -storepass password -keypass password fitnotification.apk alias

Además, si queremos verificar que el proceso se ejecutó correctamente podemos usar:

jarsigner -verify fitnotification.apk

Alinear el paquete final (.apk): Lo último que tenemos que hacer es usar zipaling en nuestro paquete firmado. Esto nos asegura que toda la información que se descomprima empiece con una alineación en particular en relación al inicio del archivo. El uso de esta herramienta no es indispensable, sin embargo ayuda a que la

aplicación se ejecute con un mayor rendimiento (utilice menos memoria RAM) puesto que Android puede usar nmap() para leer los distintos archivos, y no requiere cargar en memoria paquetes completos.

La aplicación *zipalign* está incluida en el SDK de Android y se puede ejecutar usando por ejemplo:

zipalign -v 4 fitnotification.apk fitnotification_final.apk

En donde -v activa el modo verboso y 4 indica el byte de alineación (En aplicaciones desarrolladas para Android este valor siempre debe ser 4).

Asegurando la llave privada

Mantener la llave privada en un lugar seguro es de suma importancia tanto para los desarrolladores como para los usuarios. Por ello siempre hay que tener en cuenta las siguientes recomendaciones:

- Escoger contraseñas fuertes.
- Procurar que el storepass y el keypass sean distintos.
- Al momento de generar el keystore con keytool es recomendable no indicar directamente las opciones -storepass y -keypass por que dicha información queda almacenada en el historial de la consola. Si las opciones no se indican, keytool las solicitará antes de crear la llave.
- Al momento de firmar la aplicación tener en cuenta el punto anterior.
- No prestar la llave privada a nadie. Tampoco facilitar las contraseñas.

3.4 HTML 5 y CSS3

3.4.1 HTML 5 [11],[12]

HTML ha pasado por una continua evolución desde que fue introducida al internet a inicios de los 90. Algunas características fueron introducidas en especificaciones, mientras que otras en liberaciones de versiones. La versión 4 de HTML se convirtió en una recomendación W3C en 1997. A pesar de que continúa sirviendo como una guía aproximada de muchas de las características básicas de HTML, esta no provee suficiente información para desarrollar implementaciones con una interoperabilidad entre ellas. El borrador de HTML5 refleja un esfuerzo iniciado en 2004, para estudiar

las implementaciones y contenido desplegado del HTML con temporario. Este cubre los siguientes tópicos:

- Define un lenguaje sencillo llamado HTML5 que puede ser escrito en sintaxis HTML y XML.
- Define los modelos de procesamiento detalladamente para promover la interoperabilidad entre implementaciones.
- Mejora en la definición de los documentos.
- Introduce más etiquetas y APIs.

HTML5 esta definido de manera que sea compatible hacia atrás en la forma en como los user agents manejaban los contenidos. De esta forma, los User Agents siempre tendrán que soportar aquellos elementos y atributos antiguos. A continuación se describen las nuevas características que trae HTML5.

3.4.1.1 Sintaxis

HTML5 define una sintaxis HTML compatible con documentos HTML4 y XHTML1, pero no con algunas características SGML de HTML. También define reglas detalladas de parseo, que incluyen “manejo de errores”, que son altamente compatibles con las más populares implementaciones. Los user agents deben utilizar estas reglas para recursos con tipo de media text/html. Un ejemplo de la sintaxis es la siguiente:

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Documento de ejemplo</title>
</head>
<body>
<p>Párrafo de ejemplo</p>
</body>
</html>
```

Otra sintaxis que puede ser utilizada por HTML5 es XML. Esta sintaxis es compatible con documentos e implementaciones XHTML1. Los documentos que usan esta sintaxis necesitan ser despachados con un tipo de media y sus elementos ser

colocados dentro del namespace http://www.w3.org/1999/xhtml siguiendo las reglas establecidas en las especificaciones de XML. A continuación se muestra un ejemplo de esta sintaxis:

```
<?xmlversión="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Documento de ejemplo</title>
</head>
<body>
<p>Párrafo de ejemplo</p>
</body>
</html>
```

3.4.1.2 Codificación de caracteres

En HTML5 se tienen 3 formas de configurar la codificación de caracteres:

A nivel de transporte utilizando por ejemplo la cabecera HTTP Content-Type.
Utilizando un marcador Unicode de Orden de Byte BOM (Unicode Byte Order Mark) al inicio del archivo.
Utilizado un elemento meta con el atributo charset que especifica la codificación dentro de los primeros 1024 bytes del documento. Por ejemplo <meta charset="UTF-8">, en lugar de la anterior sintaxis <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

Para sintaxis XML, para establecer la codificación de caracteres, se debe seguir las reglas establecidas en las especificaciones XML.

3.4.1.3 DOCTYPE

La sintaxis HTML en HTML5 requiere que se especifique un DOCTYPE para asegurar que los navegadores interpreten la página en modo estándar. EL DOCTYPE no posee otro propósito y es opcional en sintaxis XML. La declaración del DOCTYPE es <!DOCTYPE html> y es case-insensitive.

3.4.1.4 MathML y SVG

HTML5 permite elementos de MathML y SVG a ser utilizados dentro del documento. Por ejemplo:

```
<!doctypehtml>
<title>SVGintext/html</title>
<p>
Un círculo verde:
<svg><circler="50"cx="50"cy="50"fill="green"/></svg>
</p>
```

3.4.1.5 Nuevos elementos

Los siguientes elementos han sido introducidos para una mejor estructura:

- *section* representa un documento genérico o una sección de aplicación. Puede ser utilizado en conjunto con elementos h1, h2, h3, h4, h5, y h6.
- *article* representa un pedazo independiente de contenido de un documento.
- *aside* representa un pedazo de contenido que esta ligeramente relacionado con el resto de la página.
- *hgroup* representa el encabezado de una sección.
- *header* representa un grupo de introducción o ayudas de navegación.
- *footer* representa un pie de página de una sección y puede contener información del autor, información de copyright, etc.
- *nav* representa una sección del documento para propósitos de navegación
- *figure* representa un pedazo de flujo de contenido auto contenido. Por ejemplo:

```
<figure>
<video src="example.webm" controls></video>
<figcaption>Ejemplo</figcaption>
</figure>
```

A su vez hay muchos otros nuevos elementos:

- *video* y *audio* para contenido multimedia. Ambos proveen una API por lo que los autores de aplicaciones pueden desarrollar su propia interfaz, o disparar una interfaz que provee el user agent.
- *track* provee pistas de texto para un elemento de video.

- *embed* es utilizado para agregar contenido en forma de plugin.

- *mark*representa una serie de texto en un documento resaltado para fines de referencia, debido a su importancia en otro contexto.
- *progress* representa la finalización de una tarea, como una descarga o la realización de una serie de operaciones extensas.
- *meter* representa una medida, como el uso de disco.
- *time* representa una fecha y/o tiempo.
- *ruby*, *rt* y *rp* permite mark up anotaciones de Ruby.
- *bdi* representa un espacio de texto a ser aislado de su entorno para propósitos de formateo de texto bidireccional.
- *wbr* representa una oportunidad de salto de línea.
- *canvas* es utilizado para renderizar gráficosdinámicos bitmap en el vuelo de ejecución, como gráficos y juegos.
- *command* representa un comando que el usuario puede invocar.
- *details* representa información adicional o controles que el usuario pueda obtener bajo demanda. El elemento *summary* provee un resumen, leyenda o título.
- *datalist* en conjunto con el atributo *list* para un *input* puede ser utilizado para hacer comboboxes. Por ejemplo:

```
<input list="browsers">
<datalist id="browsers">
<option value="Safari">
<option value="Internet Explorer">
<option value="Opera">
<option value="Firefox">
</datalist>
```

- *keygen* representa un control para la generación de pares de claves.
- *output* representa algún tipo de salida, como un cálculo obtenido a través de un script.

El elemento *input* en su atributo *type* puede tener estos nuevos valores:

- tel
- search

- url
- email
- datetime
- date
- month
- week
- time
- datetime-local
- number
- range
- color

La idea de estos nuevos tipos de input es para que el user agent provea una interfaz de usuario, como un calendario, para poder enviar un formato ya definido al servidor. Esto nos da una mejor experiencia y menos tiempo de espera por retroalimentación.

Existen elementos que han sido ligeramente modificados para un mejor reflejo o mejor funcionalidad dentro de HTML5:

- El elemento *a* sin el atributo *href* ahora representa un marcador de posición para un enlace. También puede contener un flujo de contenido y no solo frases como contenido.
- El elemento *cite* ahora únicamente representa el título de un trabajo (por ejemplo un libro, paper, etc).
- El element *dl* ahora representa una lista de asociación de grupos nombre-valor.
- El elemento *head* ya no permite como hijo al elemento *object.*
- El elemento *hr* ahora representa una línea de quiebre a nivel de párrafo.
- El elemento *i* ahora representa un espacio de texto en un tono de voz distinto, o de otra forma puede ser un término técnico, un pensamiento, etc.
- El elemento *s* ahora representa contenidos que ya no son relevantes.
- El elemento *small* ahora representa comentarios secundarios.
- El elemento *strong* ahora representa importancia en lugar de un fuerte énfasis.

Existen elementos que ya no son usados por los desarrolladores. Los user agents seguiránsoportándolos y varias secciones de HTML5 definen como el parser debe tratar un elemento obsoleto. Los siguientes elementos ahora son manejados de mejor manera mediante CSS:

- basefont
- big
- center
- font
- strike
- tt

Los siguientes elementos ya no forman parte de HTML5 ya que su uso dañaría la usabilidad y accesibilidad de una aplicación:

- frame
- frameset
- noframes

Los siguientes elementos ya no se encuentran incluidos ya que no son utilizados debido a que crean confusión o su funcionalidad puede ser manejada por otros elementos:

- *acronym* crea mucha confusión. Los desarrolladores ahora utilizan *abbr* para abreviaciones.
- *applet* ha quedado obsoleto, dando lugar al elemento *object*.
- El uso de *isindex* puede ser remplazado por el uso controles de formulario.
- *dir* ha quedado obsoleto en favor del elemento *ul*.

3.4.1.6 Nuevos Atributos

HTML5 introduce una variedad de nuevos atributos para varios elementos que ya formaban parte de HTML4

- Los elementos *a*y*area*ahora poseen el atributo ***media*** por consistencia con el elemento *link*.
- El elemento *area* por consistencia con los elementos *a* y *link,* posee los atributos ***hreflang***, ***type*** y ***rel***.

- El elemento *base* ahora puede tener el atributo ***target***.

- El elemento *meta* ahora tiene el atributo ***charset*** y provee una buena forma de especificar la codificación de caracteres del documento.
- El nuevo atributo ***autofocus*** puede ser especificado en elementos *input* (que no posea el atributo ***hidden***) *select*, *textarea* y *button*. Provee una forma declarativa de colocarse en un elemento de un formulario al momento de cargarlo, mejorando la experiencia de usuario.
- El atributo ***placeholder*** puede ser especificado en elementos *input* y *textarea*. Representa una ayuda a la entrada de datos para el usuario.
- El atributo ***form*** para elementos *input*, *output*, *select*, *textarea*, *button*, *label*, *object* y *fieldset* permite la asociación de elementos a un formulario. Estos no necesariamente deben estar dentro de un elemento form y aun asíestarían asociados a este.
- El atributo ***required*** aplica sobre elementos *input* (no hidden), *image*, algunos tipos de *button* como el de tipo submit, *select* y *textarea.* Este indica al usuario que debe llenar el objeto antes de poder enviar el formulario. Para elementos *select* la primera opción debe tener un valor vacío. Por ejemplo:

<label>Color: <select name=color required>
<option value="">Escoja uno
<option>Rojo
<option>Verde
<option>Azul
</select></label>

- El elemento *fieldset* ahora permite los atributos ***disabled*** y ***name***.
- El elemento *input* posee una variedad de nuevos atributos para especificar restricciones: ***autocomplete***, ***min***, ***max***, ***multiple***, ***pattern*** y ***step***. También posee el atributo ***list*** que puede ser utilizado en conjunto con un elemento datalist; como también posee los atributos ***width*** y ***height*** para especificar dimensiones de imagen cuando se usa el atributo type=image.
- Los elementos *input* y *textarea* tienen un nuevo atributo llamad ***dirname*** que hace que la direccionalidad del elemento establecida por el usuario sea enviada correctamente.
- El elemento *textarea*también posee dos nuevos atributos, *maxlength* y *wrap* que controlan el máximo de entrada y la línea de ajuste respectivamente.
- El elemento *form* posee el atributo ***novalidate*** que puede ser utilizado para

deshabilitar la validación de envió.

- Los elementos *input* y *button* poseen los nuevos atributos ***formaction***, ***formenctype***, ***formmethod***, ***formnovalidate***, y ***formtarget***. Si se los usa, estos sobrescriben los atributos action, enctype, method, novalidate, and target del elemento *form*.
- El elemento *menu* tiene dos nuevos atributos: ***type*** y ***label***. Estos permiten al elemento transformarse en un menu típico de interfaz de usuario, como también en menu de contexto en conjunción con el atributo ***contextmenu***.
- El elemento *style* tiene el nuevo atributo ***scoped*** que puede ser utilizado para habilitar hojas de estilo de ámbito. Las reglas de estilo dentro de un elemento style solo aplican al árbollocal.
- El elemento *script* tiene el nuevo atributo ***async*** que influencia en la carga y ejecución del código del script.
- El elemento *html* posee el nuevo atributo ***manifest*** que apunta a la cache de la aplicación usada en conjunto con la API de Aplicaciones Web desconectadas u Offline Web applications API.
- El elemento *link* tiene el nuevo atributo ***sizes***. Que es utilizado en conjunto con la relación con ***icon*** para indicar su tamaño. Así permite íconos de distintas dimensiones.
- El elemento *ol* tiene el atributo ***reversed***. Cuando se utiliza, este indica que la lista de orden es descendente.
- El elemento *iframe* posee 3 nuevos atributos, ***sandbox***, ***seamless***, y **srcdoc** que permiten el uso de contenido en forma de “caja de arena” por ejemplo comentarios de un blog.

Gran variedad de atributos de HTML4 ahora aplican a todos los elementos. A estos se los llama atributos globales como ***accesskey***, ***class***, ***dir***, ***id***, ***lang***, ***style***, ***tabindex*** y ***title***. Adicionalmente en XHTML 1.0 solo se permitía el atributo xml:space en algunos elementos, el cual ahora es permitido en todos los elementos de un documento XHTML. A su vez también existen nuevos atributos globales:

- ***contenteditable*** indica que el elemento es un área editable. El usuario puede cambiar su contenido y manipular su código.
- ***contextmenu*** puede ser utilizado para apuntar a un menu de contexto.
- La colección de atributos ***data-**** definidas por el autor. Los desarrolladores pueden definir el atributo que deseen mediante el uso del prefijo data- para evitar conflictos con futuras versiones de HTML. El único requerimiento para estos

atributos es que no pueden ser utilizados en extensiones de agente de usuario.

- Los atributos ***draggable*** y ***dropzone*** pueden ser utilizados en conjunto con la nueva API drag & drop.
- ***hidden*** indica que un elemento no lo es relevante.
- Los atributos de colecciones ***role*** y ***aria-**** pueden ser utilizados para instruir a tecnologías de asistencia.
- ***spellcheck*** permite conocer si se debe comprobar la ortografía de su contenido.

HTML5 también usa todos los manejadores de eventos de HTML4, tomando el atributo global de formulario ***onevent-name***, y añade una variedad de nuevos manejadores para eventos que se definen dentro. Por ejemplo el evento *play* que es utilizado por la API de elemento de media como audio y video.

Existen ciertos cambios en atributos existentes, los que listamos a continuación:

- El atributo *value* para elemento *li* ya no es obsoleto como tampoco de presentación. De igual manera se aplica para el atributo *start* en el elemento *ol*.
- El atributo *target* para los elemento *a* y *area* ya no es obsoleto y es útil por ejemplo para aplicaciones web en conjunto con *iframe*.
- El atributo *type* en elementos *script* y *style* ya no es requerido si el lenguaje del script es ECMAScript y el lenguaje de estilo es CSS respectivamente.
- El atributo *border* en el elemento *table* solo permite los valores “1” y un string vacío.

Los siguientes atributos son permitidos pero los desarrolladores ya no los utilizan, por lo que optan por soluciones alternas:

- El atributo *border* en el elemento *img.* Es requerido que tenga el valor “0” cuando se utiliza, pero los desarrolladores pueden utilizar en su lugar CSS.
- El atributo *language* en elementos *script.* Es requerido que tenga el valor “JavaScript” y no tener conflictos con el atributo *type.* Los desarrolladores simplemente omiten esta función ya que no es muy útil.
- El atributo *name* en elementos *a.* Los desarrolladores optan por utilizar el atributo *id* en su lugar.
- Los atributos *width* y *height* en elementos *img* y otros que los soportan, ya no tienen permitido establecer valores como porcentajes.

Algunos atributos de HTML4 ya no son permitidos en HTML5. La especificación define como un user agent debe procesarlos, pero los desarrolladores ya no deben utilizarlos. La siguiente lista especifica aquellos atributos que no deben ser utilizados:

- *rev* y *charset* en elementos link y a.
- *shape* y *coords* en elementos a.
- *longdesc* en elementos img e iframe.
- *target* en elementos link.
- *nohref* en elementos área.
- *profile* en elementos head.
- *version* en el elemento HTML.
- *name* en elementos img, donde se debe utilizar *id* en su lugar.
- *scheme* en elementos meta.
- *archive*, *classid*, *codebase*, *codetype*, en elementos object.
- *valuetype* y *type* en elementos param.
- *axis* y *abbr* en td y th.
- *scope* en elementos td.
- *summary* en elementos table.

La siguiente lista de atributos ya no se encuentra en HTML5, ya que pueden ser manejados de mejor manera mediante CSS:

- *align* en elementos caption, iframe, img, input, object, legend, table, hr, div, h1, h2, h3, h4, h5, h6, p, col, colgroup, tbody, td, tfoot, th, thead and tr.
- *alink*, *link, text,vlink* y *background* en el elemento body.
- *bgcolor* en elementos table, tr, td, th and body.
- *border* en elementos object.
- *cellpadding* y *cellspacing* en elementos table.
- *char* y *charoff* en elementos col, colgroup, tbody, td, tfoot, th, thead y tr.
- *clear* en elementos br.
- *compact* en elementos dl, menu, ol y ul.

- *frame* y *rules* en elementos table.

- *frameborder*, *marginheight*, *scrolling* y *marginwidth*en elementos iframe.
- *height* en elementos td y th.
- *hspace* y *vspace* en elementos img y object.
- *noshade* y *size* en elementos hr.
- *nowrap* en elementos td y th.
- *type* en elementos li, ol y ul.
- *valign* en elementos col, colgroup, tbody, td, tfoot, th, thead y tr.
- *width* en elementos hr, table, td, th, col, colgroup and pre.

3.4.1.7 APIs

HTML5 introduce un número de APIs que brindan ayuda en la creación de aplicaciones Web. Estas pueden ser utilizadas en conjunto con los nuevos elementos introducidos en esta versión:

- API para la ejecución de video y audio, y que puede ser utilizado con los nuevos elementos *video* y *audio*.
- API que permite aplicaciones Web sin conexión u offline Web applications.
- API que permite a una aplicación web auto registrarse para ciertos protocolos o tipos de media.
- API de edición en combinación con el nuevo atributo global *contenteditable*.
- Drag & drop API en combinación con el atributo *draggable*.
- API que permite el acceso al historial y permite que se añadan paginas a esta para prevenir errores en el botón de regresar.
- Extensiones a HTMLDocument
    - HTML5 extiende la interfaz HTMLDocument del DOM nivel 2 en varias formas. La interfaz es ahora implementada en todos los objetos mediante la interfaz Document. Además posee algunos notables nuevos miembros:
    - getElementsByClassName() para seleccionar elementos en base a su nombre de clase. La forma en que este método esta definido permite trabajar con cualquier contenido con el atributo *class* y un objeto Document como SVG y MathML.

- innerHTML es una forma fácil de interpretar y serializar un documento HTML o XML. En un inicio este atributo solo estaba disponible en elementos HTML en navegadores web y no formaba parte de ningún estándar.

- activeElement y hasFocus determinan que elemento se encuentra actualmente seleccionado y si el Document posee selección respectivamente.
- Extensiones a HTMLElement
- La interfaz HTMLElement posee también algunas extensiones dentro de HTML5:
- getElementsByClassName() que es básicamente una versión dentro del ámbito del HTMLElement, y similar a la encontrada dentro de HTMLDocument.
- innerHTML también encontrada en los navegadores web actuales. También se encuentra definida para trabajar en el contexto XML.
- classList es una forma conveniente de acceder al className. El objeto que este retorna, expone métodos (contains(), add(), remove(), y toggle()) para manipular las clases de los elementos.

3.4.2 CSS 3 [13]

CSS no provee una versión en el sentido tradicional, en su lugar utiliza niveles. Cada nivel de CSS se construye en base a su anterior, refinando definiciones y añadiendo características, por lo que cada nivel alto es un conjunto superior de cualquier nivel inferior. Un agente de usuario conformado por el mayor nivel de CSS es entonces conformado también por todos los niveles inferiores.

3.4.2.1 Niveles CSS

CSS Nivel 1

La CSS Working Group considera la especificación CSS1 como obsoleta. CSS Nivel 1 viene definida por todas las características de la especificación CSS1 (propiedades, valores, at-rules, etc.) pero utilizando la sintaxis y definiciones de la especificación CSS2.1.

CSS Nivel 2

La especificación CSS2 es técnicamente una recomendación W3C, aunque pasado el tiempo de implementación y revisiones, ha traído a la luz muchos problemas en

ésta. Por tal razón en lugar de expandir una lista de errores difícil de manejar, la CSSWG optó por definir la CSS Nivel 2 Revisión 1 o más conocida como CSS2.1, la que contiene una definición definitiva en caso conflictos entre esta y la recomendación.

Una vez que la CSS2.1 pasó a ser una Recomendación Candidata y aunque no con el mismo nivel de estabilidad que CSS2, esta última pasó a ser obsoleta. CSS2.1 define la CSS Nivel 2 y la especificación CSS Style Attributes define su inclusión en elementos específicos.

CSS Nivel 3

CSS Nivel 3 se construye sobre CSS Nivel 2 módulo a módulo utilizando la especificación CSS2.1 como su núcleo. Cada módulo añade funcionalidad o reemplaza a parte de la especificación. La CSS Working Group trata de que los nuevos módulos CSS no contradigan la especificación CSS2.1. Una vez que cada módulo es completado, este será añadido al actual sistema CSS2.1.

Desde este nivel, los módulos son nivelados independientemente: por ejemplo Selectors Nivel 4 deben ser bien definidos antes del módulo CSS Line Nivel 3.

3.4.2.2 Especificaciones de CSS

Desde el año 2010, CSS o por sus siglas en inglés, Cascading Style Sheets, viene definida por las siguientes especificaciones:

- CSS Nivel 2 Revision 1 CSS2.1
- CSS Style Attributes
- Media Queries Level 3
- CSS Namespaces
- Selectors Level 3
- CSS Color Level 3

3.4.2.3 Perfiles CSS

No todas las implementaciones aplican todas las funcionalidades definidas en CSS. Por ejemplo una implementación puede únicamente implementar solo la funcionalidad requerida por un perfil CSS. Los perfiles definen un subconjunto de CSS considerados fundamentales para una clase específica de implementación CSS. La W3C CSS Working Group define los siguientes perfiles:

- CSS Mobile Profile 2.0
- CSS Print Profile 1.0
- CSS TV Profile 1.0

3.4.2.4 Perfil CSS 2.0 para móviles [14]

Selectores: estos se encuentran definidos dentro del respectivo capítulo de la especificación CSS 2.1. La tabla 3-1 especifica que selector debe ser soportado de entre los definidos por CSS 2.1

Tipo de Selector	Ejemplo	Definición	Aplicabilidad
Selector Universal	*	Cualquier elemento	REQUERIDO
Selectores de Tipo	E	Cualquier elemento E	REQUERIDO
Selectores Descendientes	E F	Cualquier elemento F que sea descendiente del elemento E	REQUERIDO
Selectores hijos	E > F	Cualquier elemento F que sea hijo del elemento E	REQUERIDO
Pseudoclase link	E:link	Elemento E que como hipervínculo aun no ha sido visitado por el usuario	REQUERIDO
Pseudoclase link	E:visited	Elemento E que como hipervínculo visitado por el usuario	REQUERIDO
Pseudoclases dinámicas	E:active	Elemento E que está siendo activado por el usuario	REQUERIDO
Pseudoclases dinámicas	E:focus	Elemento E en donde se encuentra el foco actualmente	REQUERIDO
Selector de clase	.warning	En XHTML, es el	REQUERIDO

		elemento en el que su atributo class tiene como valor el dicho nombre.	
Selector de Id	#myid	Cualquier elemento con el atributo id igual a “myid”	REQUERIDO
Agrupamiento	E1, E2, E3 { . . . }	Grupo de elementos que comparten las mismas declaraciones de estilo.	REQUERIDO

Tabla 3-2: Selectores soportados por CSS 2.1

Reglas: definidas en la sección “at-rules” de la especificación CSS 2.1. La tabla 3-2 identifica que reglas deben ser soportadas de entre las definidas.

Tipo de Selector	**Definición**	**Aplicabilidad**
@charset	Define la codificación de caracteres para la hoja de estilo	REQUERIDO
@import	Importa una hoja de estilos externa	REQUERIDO
@media	Agrupa reglas de estilo para ser aplicadas solo en una o más medias particulares.	REQUERIDO
@namespace	Para la declaración de namespaces.	OPCIONAL

Tabla 3-3: Reglas definidas por CSS 2.1

Propiedades: definidas en varios capítulos en las especificaciones CSS 2.1 y CSS box model. La tabla 3-3 identifica que propiedades (y sus limitaciones) deben ser soportadas de entre las definidas por las especificaciones.

Propiedad	**Sintaxis**	**Aplicabilidad**
background-color		REQUERIDO
background-image		REQUERIDO
background-repeat		REQUERIDO

background-attachment		REQUERIDO
background-position	top | center | bottom | left | right | inherit	REQUERIDO
background		REQUERIDO
border-top-width border-right-width border-bottom-width border-left-width		REQUERIDO
border-width		REQUERIDO
border-top-color border-right-color border-bottom-color border-left-color		REQUERIDO
border-color		REQUERIDO
border-top-style border-right-style border-bottom-style border-left-style	none | solid | dashed | dotted | inherit	REQUERIDO
border-style	none | solid | dashed | dotted | inherit	REQUERIDO
border-top border-right border-bottom border-left		REQUERIDO
border		REQUERIDO
bottom		OPCIONAL
clear		REQUERIDO
color		REQUERIDO
display	inline | block | list-item | none | inherit	REQUERIDO
float		REQUERIDO

font-family		REQUERIDO
font-style		REQUERIDO
font-variant		REQUERIDO
font-weight		REQUERIDO
font-size	<absolute-size> | <relative-size> | inherit	REQUERIDO
font		REQUERIDO
height		REQUERIDO
left		OPCIONAL
list-style-type	disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none | inherit	REQUERIDO
list-style-image		REQUERIDO
list-style		REQUERIDO
margin-top margin-right margin-bottom margin-left		REQUERIDO
margin		REQUERIDO
marquee-direction [CSS3BOX]		REQUERIDO
marquee-loop [CSS3BOX]		REQUERIDO
marquee-speed [CSS3BOX]		REQUERIDO
marquee-style [CSS3BOX]		REQUERIDO

max-height		REQUERIDO
max-width		REQUERIDO
min-height		REQUERIDO
min-width		REQUERIDO
outline-color		OPCIONAL
outline-style	none | solid | dashed | dotted | inherit	OPCIONAL
outline-width		OPCIONAL
outline		OPCIONAL
overflow [CSS3BOX]	auto	REQUERIDO
overflow-style [CSS3BOX]	marquee	REQUERIDO
padding-top padding-right padding-bottom padding-left		REQUERIDO
padding		REQUERIDO
position		OPCIONAL
right		OPCIONAL
text-indent		REQUERIDO
text-align		REQUERIDO
text-decoration	none | blink |underline | inherit	REQUERIDO
text-transform		REQUERIDO
top		OPCIONAL
vertical-align	top | middle | bottom | baseline | inherit	REQUERIDO
visibility		REQUERIDO

white-space		REQUERIDO
width		REQUERIDO
z-index		OPCIONAL

Tabla *3-4: Propiedades de CSS 2.1*

3.4.2.5 CSS Device Adaptation [15],[16]

CSS 2.1 especifica un bloque inicial que trata de las dimensiones de la ventana o viewport. Los dispositivos móviles poseen un viewport mucho más pequeño que el de un explorador de computadora de escritorio. Ciertos DOCTYPEs (como XHTML Mobile Profile) son utilizados para reconocer documentos que son diseñados para dispositivos móviles. Adicionalmente, una etiqueta META HTML puede ser introducida para permitir especificar el tamaño del bloque contenedor y el zoom inicial.

La especificación introduce una forma de sobrescribir el tamaño del viewport que provee el agente de usuario o user agent (UA). Para esto, es necesario introducir la diferencia entre *initial viewport* y *actual viewport*.

initial viewport: se refiere al viewport antes de que un UA o un autor lo sobrescriba. El tamaño del viewport inicial cambiará con el tamaño de la ventana o el área de vista.
actual viewport: es el viewport que se obtiene después de aplicar los descriptores y procedimientos de restricción asociados.

La regla @viewport: consiste de la clave @ seguida de un bloque de descriptores que definen el viewport. Los descriptores dentro de esta regla son definidos por documento, no hay herencia involucrada y sobrescribe los descriptores de reglas precedentes (como la regla @page). Ejemplo:

```
@viewport {
width: device-width;
}
```

Descriptores del viewport:

- min-width y max-width: especifica el mínimo y máximo ancho del viewport que es utilizado para establecer el tamaño inicial del bloque de contenido, donde:
  <viewport-length> = auto | device-width | device-height | <length> |

<percentage>

auto: el valor es calculado en base a los otros descriptores de acuerdo al procedimiento de restricción.
device-width: el ancho de la pantalla en pixeles CSS con un factor zoom de 1.0
device-height: el alto de la pantalla en pixeles CSS con un factor zoom de 1.0
<length>: un tamaño positivo absoluto o relativo.
<percentage>: un valor en porcentaje relativo al ancho o alto del viewport inicial con un factor zoom de 1.0.

- width: un descriptor taquigráfico para establecer los descriptores min-width y max-width.
- min-height y max-height: especifica el mínimo y máximo alto del viewport que es utilizado para establecer el tamaño inicial del bloque de contenido.
- height: un descriptor taquigráfico para establecer los descriptores min-height y max-height.
- zoom: especifica el zoom inicial para la ventana o área de vista.
    - *auto:* un factor positivo de 1.0 (o 100% para valores en porcentajes) significa que no se aplica zoom alguno, mientras que valores mayores o menores si lo hacen.
- min-zoom: especifica el menor factor de zoom permitido.
- max-zoom: especifica el mayor factor de zoom permitido.
- user-zoom: especifica el zoom que puede ser cambiado por la interacción del usuario o sin ella.
    - *zoom:* el usuario puede cambiar interactivamente el zoom.
    - *fixed:* el usuario no puede cambiar interactivamente el zoom.
- orientation: este descriptor es utilizado para pedir que un documento sea visualizado en modo portrait (horizontal) o landscape (vertical). Este puede ser utilizado para prohibir un cambio de orientación.
    - *auto:* el UA escoge automáticamente la orientación basado en la operación del dispositivo.
    - *portrait:* el documento es bloqueado solo para una presentación en forma vertical o portrait.
    - *landscape:* el documento es bloqueado solo para una presentación en forma horizontal o landscape.

Media queries: los efecto de las reglas de @viewport en media queries necesitan cierta atención extra, y es donde se hace el uso de media queries para la agrupación

y aplicación de descriptores. Un ejemplo del uso de estas características es la siguiente:

```
@viewport {
width: device-width;
}

@media screen and (min-width: 400px) {
div {color: red;}
}

@media screen and (max-width: 400px) {
div {color: green;}
}
```

Dado un tamaño de dispositivo de 320px y una hoja de estilos con un UA de 980px, la primera media query no es aceptada, pero la segunda si será utilizada.

3.5 Jquery Mobile [17]

Jquery Mobile es un Framework diseñado para ser utilizado en dispositivos inteligentes y tabletas, esta optimizado para permitir un buen manejo de dispositivos con pantalla táctil, aunque nos permite generar sistemas con interfaces de usuario amigables (basadas en HTML 5), para cualquier dispositivo móvil (de cualquier plataforma). Actualmente la versión más nueva de esta tecnología es la 1.1.0.

Uno de los objetivos de jQuery, es permitirnos hacer más, escribiendo menos líneas de código. Esto es posible ya que al desarrollar una aplicación usando esta tecnología, estamos creando una aplicación que se podrá ejecutar en los dispositivos inteligentes, tabletas y sistema de escritorio más populares del mercado. jQuery Mobile es soportado por iOS, Android, BlackBerry, bada, Windows Phone, palm webOS, sumbian y MeeGo.

jQuery pone mucho énfasis en el uso marcas. Es por esto que es muy similar a HTML y es muy fácil de utilizar. Además, incluye un sistema de navegación basado en Ajax que permite transiciones animadas, así como un grupo de widgets entre los que se encuentran: páginas, diálogos, barras de herramientas, listas, botones con íconos, elementos de formulario, acordeones y más.

3.5.1 Características:

- Está basado en el núcleo de jQuery por lo que es muy fácil de aprender si hemos utilizado jQuery anteriormente.
- Es liviano y hace uso mínimo de imágenes para permitir aplicaciones rápidas.
- Posee una arquitectura modular que permite incluir solo ciertas características de nuestra aplicación en otra aplicación en particular.
- Se puede configurar haciendo uso de HTML5 por lo que permite un desarrollo más rápido.
- La característica de crecimiento progresivo permite que su ejecución sea similar a la de aplicaciones nativas.
- Técnicas de diseño sensibles, hacen que el mismo código se ejecute de manera correcta tanto en pantallas de dispositivos móviles cómo en navegadores de escritorio.
- Soporta tanto eventos de ratón como eventos táctiles.

3.5.2 Componentes:

3.5.2.1 Páginas:

Las páginas en jQuery están optimizadas para soportar tanto páginas únicas como páginas internas múltiples en una sola página. Esto permite que los desarrolladores creen aplicaciones usando las mejores prácticas, que parecen aplicaciones nativas y que no se pueden obtener haciendo uso de pedidos HTTP.

Una aplicación de jQuery debe iniciarse con una etiqueta doctype. Además, si queremos indicar tanto las dimensiones como el nivel de acercamiento inicial de nuestra aplicación, tendremos que hacer uso de un metatag viewport en la cabecera. Una buena idea es usar el siguiente código:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

El mismo que se encargará de ajustar el ancho de nuestra aplicación dependiendo del dispositivo desde el que se ejecute.

Ya dentro del cuerpo (<body>), tendremos que colocar las páginas que tendrá nuestra aplicación. Para crear una página, tenemos que añadir un elemento de tipo div con el atributo data-role=”page”. Dentro de una página, podemos escribir cualquier código HTML, sin embargo siempre es recomendable utilizar otros divs con distintos data-roles (“header” para encabezados, “content” para contenidos y “footer” para los pie de página).

Si queremos que nuestra aplicación tenga varias páginas, simplemente tendremos que indicar tantos divs con data-role=”page” como queramos. Para poder navegar internamente entre las páginas, es importante que cada una tenga un identificador único, para ello, podemos hacer uso del atributo id.

jQuery Mobile permite además que se puedan cargar páginas de manera dinámica simplemente llamando al método $.mobile.changePage(). Esto es muy útil, en especial, cuándo generamos páginas en el lado del servidor. Es importante saber, que al momento de ejecutar $.mobile.changePage() se llama automáticamente a los eventos pagebeforechange (se ejecuta antes de que la página se cargue), pagechange (se ejecuta cuando la página se ha cargado por completo) y pagechangefailed (se ejecutar si ocurrió un error.)

Algo que también hay que tener en cuenta al momento de crear páginas, es hacer uso de un tema adecuado para nuestra aplicación. Esto se puede lograr haciendo uso de la etiqueta data-theme=”” he indicando el tema que más nos guste. Este atributo se puede utilizar en cualquiera de los divs de la aplicación, sin embargo lo óptimo es colocarlo a nivel de la página. Jquery provee inicialmente 5 temas distintos que se pueden obtener usando los nombres “a”, “b”, “c”, “d” o “e”.

3.5.2.2 Diálogos:

Cualquier página puede presentarse como un diálogo haciendo uso del atributo: data-rel=”dialog”. Cuándo hacemos uso de diálogos, jQuery Mobile nos muestra la página dentro de un cuadro con esquinas redondeadas, márgenes y un fondo negro.

Para poder cerrar un diálogo, es necesario crear un enlace con el atributo data-rel=”back”. Sin embargo, al presionar cualquier otro enlace dentro del diálogo, este se cerrará automáticamente y se mostrará la nueva página.

Para permitir una adecuada lectura del contenido, los diálogos tienen por defecto una anchura máxima de 500px, 15 px de relleno a los lados, y 10% de margen en la parte superior. Si se requiere modificar esto, será necesario editar la hoja de estilos estándar como se desee.

3.5.2.3 Barras de Tarea

En jQuery existen dos tipos de barras de tareas. Los encabezados (header) y los pies de página (footer).

Encabezados: Funcionan como el título de la página, por lo general es el primer elemento que aparece en la aplicación. Suele contener hasta dos botones. Normalmente utiliza la etiqueta H1 para definir el tamaño del texto, aunque se puede usar cualquiera (H1-H6).
Pies de página: Es el último elemento de la página y por lo general se encuentra en todas las páginas de la aplicación. Aquí se suele poner cualquier contenido (por lo general texto y botones).

Es muy común tener una barra de navegación dentro de la cabecera o del pie de página. Gracias a jQuery es posible lograr esto usando el elemento navbar.

Posicionamiento:

Los encabezados y pies de página se pueden posicionar dentro de la página de distintas maneras. Por defecto, estas se posicionan en modo “inline”. A continuación se describen los distintos modos de posicionamiento:

- Inline: Nos permite estar seguros de que las barras se puedan ver en todas las páginas de nuestra aplicación y en todos los dispositivos en los que se ejecute debido a que usa el comportamiento predeterminado de HTML (no requiere de CSS o JavaScript).
- Fixed: Se encarga de posicionar las barras ya sea en la parte superior o inferior de la pantalla, pero sólo en exploradores que soporten posicionamiento CSS (la mayoría de navegadores, iOS5+, Android 2.2+, BlackBerry 6, entre otros). Si el explorador no lo soporta, usa el modo “inline”

automáticamente. Una de las características de este modo, es que nos permite ocultar las barras si tenemos “tap-togglin” habilitado, de este modo el usuario puede expandir la pantalla sin necesidad de hacer un acercamiento.

- Fullscreen: Funciona igual que el modo “fixed” pero en este caso las barras sobreponen el contenido de la página. Se usa en aplicaciones de fotografía o vídeo, en donde se quiere que el contenido ocupe toda la pantalla. Es recomendable hacer uso de estas solo en situaciones específicas.

Características de la Cabecera

Por defecto, la cabecera usa el tema “a” (negro) sin embargo esto se puede modificar fácilmente usando la etiqueta data-theme. Además, las cabeceras vienen configuradas de tal modo que reservan un espacio para agregar botones a los lados del texto. Estos botones usan el estilo “inline” para ahorrar espacio, de modo que su altura no será mayor a la del contenido existente en la cabecera.

Cuando insertamos botones en una cabecera, jQuery por defecto coloca el primer botón a la izquierda y el segundo a la derecha del texto (si queremos modificar esto tendremos que usar las clases ui-btn-left o ui-btn-right). Además, al agregarlos, estos heredarán el tema “a”, es por esto que es importante especificar un tema diferente si queremos que nuestros botones resalten.

En todas las cabeceras, es posible además, agregar un botón para regresar. Esto es muy útil en aplicaciones nativas que se ejecutan en un webview. Para activar esta característica simplemente tendremos que agregar la etiqueta data-add-back-btn=”true” en el div de nuestra página. También podemos hacer que cualquier botón funcione como un botón de regreso si le agregamos a este el atributo data-rel=”back”.

Características del pie de página

El pie de página es muy similar a la cabecera respecto a opciones y configuración. La principal diferencia, es que está diseñado para ser menos estructurado, por lo que permite mayor flexibilidad.

Una de las características de los pies de páginas es que cualquier enlace que se agregue al mismo se convertirá automáticamente en un botón. Al igual que en la cabecera, todos los botones poseen estilo “inline”. Los botones agregados, pueden además agruparse. Para lograr esto tendremos que colocarlos dentro de un div que posea los atributos: data-role=”controlgroup” y data-type=”horizontal”.

Por defecto, los pies de página no incluyen “padding”, es por esto que es importante utilizar la clase “ui-bar” si queremos que el contenido luzca ordenado.

3.5.2.4 Barras de navegación (navbar)

jQuery incluye un widget que simula una barra de navegación, la misma que permite agregar hasta 5 botones en los que se puede poner íconos y/o texto.

Una barra de navegación se codifica como una lista desordenada dentro de un contenedor que posee el atributo data-role=”navbar”. Para hacer que uno de los elementos en la barra de tareas luzca como activo, tendremos que usar la clase “ui-btn-active”. A continuación se muestra el código de una barra de navegación simple:

```
<div data-role="navbar">
<ul>
<li><a href="a.html" class="ui-btn-active">A</a></li>
<li><a href="b.html">B</a></li>
</ul>
</div>
```

Las barras de navegación vienen configuradas para dividir el ancho de cada botón de acuerdo a la cantidad de botones que se especifiquen. Si usamos por ejemplo 3 botones, el tamaño de cada uno será la tercera parte de la pantalla. Si agregamos más de 5 elementos, jQuery se encargará de crear múltiples líneas.

Los “navbars” pueden posicionarse en cualquier parte de nuestra aplicación. Sin embargo, por lo general se agregan en la cabecera o en el pie de página. Si queremos, es posible además agregar íconos a cada uno de los elementos de la barra de navegación. Para lograr esto, sólo tenemos que agregar el atributo data-

icon e indicar en donde queremos que se posicione usando data-iconpos (top, bottom, left, right).

Al igual que los botones, las barras de navegación heredan el estilo de las barras de tareas (A) o el que se haya especificado en el cuerpo de la aplicación. Sin embargo, es posible modificar el estilo de cada uno de los elementos (<li>) especificando el atributo data-theme.

3.5.2.5 Botones e Iconos

Los botones se codifican haciendo uso de HTML, y luego jQuery se encarga de modificarlos para que se vean mejor. Es por esto, que podemos usar la etiqueta <a> si queremos agregar enlaces, o <input> y <button> si queremos agregar botones que envíen formularios. Para hacer que un enlace aparezca como un botón, será necesario hacer uso del atributo data-role=”button” (esto no se necesita hacer en el caso de los inputs y buttons).

Al igual que en las barras de navegación, es posible agregar íconos a los botones haciendo uso de los atributos data-icon y data-iconpos. Por defecto, jQuery permite hacer uso de los siguientes íconos:

arrow-l: Flecha hacia la izquierda
arrow-r: Flecha hacia la derecha
arroy-u: Flecha hacia arriba
arrow-d: Flecha hacia abajo
delete: Eliminar
plus: Signo Más (+)
minus: Signo Menos (-)
check: OK
gear: Configuración
refresh: Actualizar
forward: Adelante
back: Atrás
grid: Menú
star: Favoritos
alert; Advertencia

info: Información
home: Página de Inicio
search: Buscar

Si queremos agregar nuevos íconos tendremos que indicar un nuevo valor (por ejemplo mail) en el atributo data-icon, y especificar en nuestro archivo de estilos (CSS) las características del mismo bajo el nombre ui-icon-mail. Es recomendable que los nuevos íconos tengan un tamaño de 18x18 para mantener la consistencia con los íconos usados por defecto.

Por defecto todos los botones que se agreguen usaran el estilo “block” de modo ocuparán todo el ancho de la página. Si queremos que los botones sean compactos, habrá que especificar el atributo data-inline=”true”, cuándo usamos este atributo, además logramos que nuevos botones floten a la derecha del botón inicial.

3.5.3 Contenidos

El estilo del contenido de las páginas en jQuery puede modificarse tanto como se desee, sin embargo, jQuery, utiliza por defecto temas y tamaños estándar para ciertos elementos (cabeceras, párrafos, bloques, enlaces, listas y tablas). Además, jQuery provee algunas herramientas que hacen que sea mucho más fácil formatear nuestro contenido. Entre las herramientas que podemos utilizar, están los menús plegables y las rejillas de diseño.

3.5.3.1 Rejillas de diseño

Usar diseños de varias columnas no es siempre recomendable cuándo trabajamos con dispositivos móviles debido al limitado ancho de la pantalla. Sin embargo, a veces se requiere hacer uso de columnas para mostrar pequeños elementos que tienen que estar juntos.

Para esto, jQuery provee la clase ui-grid, la misma que se puede configurada de cuatro maneras:

-2 columnas usando la clase ui-grid-a
-3 columnas usando la clase ui-grid-b

-4 columnas usando la clase ui-grid-c
-5 columnas usando la clase ui-grid-d

Las rejillas utilizan el 100% de la pantalla, son invisibles y no tienen ni márgenes, ni relleno (padding). Cuándo se hace uso de estas, es importante colocar dentro bloques, los mismos que tienen que usar la clase ui-block-a/b/c/d

3.5.3.2 Menús plegables

Para utilizar menús plegables, tenemos que hacer uso de un nuevo atributo (data-role=”collapsible”). Cuando hacemos uso de estos elementos, tendremos que en primer lugar colocar el nombre del menú dentro de una etiqueta de título (h1-h6), y agregar cualquier contenido a continuación (mismo que se mostrará/ocultará cuando presionemos el menú).

Para personalizar los menús plegables, podemos hacer uso de los temas que se han indicado anteriormente (a-e). Para indicar el tema del menú, usaremos el atributo data-theme, y para el contenido, data-content-theme.

Los menús plegables, pueden además agruparse haciendo uso de otro atributo (data-role=”collapsible-set”). En estos elementos, siempre va a mostrarse el contenido de sólo uno de los menús que formen el grupo.

3.5.3.3 Formularios

Todos los elementos de formulario de HTML se pueden utilizar con jQuery, permitiendo que estos se vean mejor.

Para hacer uso de un formulario, tendremos que hacer uso de la etiqueta form, e indicar la acción y el método que se van a utilizar.

```
<form action="formulario.php" method="post"></form>
```

Algo importante que hay que tener en cuenta, cuando trabajamos con formularios y jQuery, es que todos los elementos de un formulario tienen que tener un id único, no sólo a nivel de la página, sino a nivel de toda la aplicación.

Los elementos que podemos utilizar en formularios, son los siguientes:

- Cuadro y Área de texto: Estos se codifican con HTML estándar, es decir que solo tendremos que usar la etiqueta “input” y los atributos type=”text” o type=”textarea” dependiendo de lo que se desee hacer. Por lo general, los inputs vienen acompañados por etiquetas (labels) las mismas que se posicionarán arriba de los inputs (Si queremos que las etiquetas se coloquen a la derecha, es recomendable usar el atributo data-role=”fieldcontain”)
- Cuadro de búsqueda: Estos son unos nuevos elementos de HTML que vienen por defecto con las esquinas redondeadas y que muestran un ícono para borrar el contenido una vez que empezamos a digitar sobre él. Para usar estos elementos, tendremos que usar la etiqueta “input” y el atributo type=”search”.
- Interruptor: Este es un elemento muy común en los dispositivos móviles. Funciona como un checkbox, puesto que nos permite escoger entre prendido y apagado. Para obtener este elemento, tendremos que hacer uso de la etiqueta “select” y asignarle el atributo data-role=”slider”. Cómo en la mayoría de elementos, es posible darle diferentes estilos usando el atributo data-theme.
- Control deslizante: Para agregar controles deslizantes o sliders a nuestra aplicación, tendremos que hacer uso de un input que tenga el atributo type=”range”. A través del atributo “value” se puede indicar el valor inicial del elemento, y gracias a “max” y “min” se puede indicar el rango de valores que se podrá seleccionar.
- RadioButtons: Son grupos de botones de los cuáles se puede escoger solo uno. Para usarlos, tendremos que agregar un input con el atributo type=”radio”. Para agrupar los distintos botones, es necesario que todos tengan el mismo nombre. También es recomendable colocar a todos los elementos dentro de un fieldset con data-role=”controlgroup”.
- CheckBoxes: A diferencia de los radiobuttons, cuándo usamos checkboxes esta permitido escoger más de una opción. Para crear checkboxes tendremos que agregar un “input” con type=”checkbox”. Para que se muestren bien este tipo de elementos en jQuery, es importante que los coloquemos dentro de un label.

- Menús de selección (Comboboxes): Se basan en los menús de selección estándar de HTML. Para usarlos, tenemos que usar la etiqueta select, y agregar adentro cada una de las opciones (option) indicando el valor y la etiqueta de cada una.
- Botones: Son inputs de tipo submit. Estos nos permiten enviar la información hacia el servidor.

Enviando información

En jQuery Mobile, por defecto se manejan todos los “Submits” haciendo uso de Ajax (si queremos, podemos eliminar el uso de Ajax. Para ello solo tendremos que usar el atributo data-ajax=”false”) de modo que la transición entre las pantallas sea suave. Para asegurar que la información se envíe, es importante indicar siempre tanto el método, como la acción que se va a tomar.

3.5.4 API

Eventos

jQuery Mobile ofrece varios eventos que permiten crear ganchos (hooks) muy útiles. Uno de los eventos más importantes es el evento ready() el cual se ejecuta cuándo el DOM de la primera página ha sido cargado. A continuación, se detallan algunos otros eventos, útiles tanto en aplicaciones de escritorio, como en aplicaciones móviles.

Eventos táctiles

- tap: Se ejecuta cuándo damos un toque rápido.
- taphold: Se ejecuta cuando tenemos presionada la pantalla más o menos un segundo.
- swipe: Se ejecuta cuando realizamos un deslizamiento de 30px o más en menos de un segundo.
- swipeleft, swiperight: Igual que swipe, pero varían de acuerdo a la dirección.

Además de los eventos táctiles, jQuery proporciona 6 eventos virtuales del ratón, que permiten abstraer los eventos táctiles y los no táctiles. Estos son: vmouseover, vmousedown, vmousemove, vmouseup, vmouseclick y vmouscancel.

Algunos otros eventos importantes se listan a continuación:

- orientationchange: Se ejecuta cuando giramos la pantalla del dispositivo.
- scrollstart: Se ejecuta cuando empezamos a desplazarnos.
- scrollstop: Se ejecuta cuando terminamos de desplazarnos.
- pagebeforeload: Se ejecuta antes de que se ejecute una petición de carga.
- pageload: Se ejecuta cuando la página se ha cargado por completo.
- pageloadfailed: Se ejecuta cuando la página no se pudo cargar.
- pagebeforechange: Se ejecuta antes de cualquier transición.
- pagechange: Se ejecuta cuando se ha cargado la nueva página y todas las transiciones se han ejecutado.
- pagechangefailed: Se ejecuta cuando no se pudo cargar la nueva página.
- pagecreate: Se ejecuta cuando se crea la página
- pageinit: Se ejecuta cuando la página ha sido inicializada.
- pageremove: Se ejecuta justo antes de que se trate de eliminar una página del DOM.
- updatelayout: Se ejecuta cuando se muestra/esconden elementos de forma dinámica.

Métodos y Utilidades

jQuery provee una serie de métodos que permiten ahorrarnos códigos al momento de desarrollar la aplicación. A continuación se indican algunos de ellos:

- $.mobile.changePage: Nos permite cambiar de página dentro de la aplicación.
- $.mobile..loadPage: Nos permite cargar una página externa y agregarla al DOM.
- $.fn.jqmData, $.fn.jqpRemoveData: Nos permiten trabajar con cierta información del core de jQuery.
- $.fn.jqmHijackable: Hace que siempre se ejecuten los submits sin hacer uso de Ajax.

- $.mobile.showPageLoadingMsg: Muestra un mensaje indicando que se está cargando una página.
- $.mobile.hidePageLoadingMsg: Esconde el mensaje que indica que se está cargando una página.
- $.mobile.path.parseUrl: Nos permite parsear una URL.
- $.mobile.path.makePathAbsolute: Hace que una ruta relativa se convierta en absoluta.
- $.mobile.path.isSameDomain: Nos permite saber si dos URL pertenecen al mismo dominio.
- $.mobile.silentScroll: Permite ejecutar un desplazamiento hasta cierta posición.
- $.mobile.activePage: Indica cual es la página que se esta mostrando en ese momento.

3.5.5 Plataformas [18]

jQuery está soportado en la mayoría de plataformas de escritorio, de dispositivos inteligentes y de tablets. A continuación se detalla el nivel de soporte para cada uno de ellos:

Tipo A: Compatibilidad completa. Permite transiciones animadas basadas en Ajax.

- Apple iOS 3.2-5.0
- Android 2.1-2.3
- Android 3.1 (Honeycomb)
- Android 4.0 (ICS)
- Windows Phone 7-7.5
- Blackberry 6.0
- Blackberry 7
- Blackberry Playbook (1.0-2.0)
- Palm WebOS (1.4-2.0)
- Palm WebOS 3.0
- Firefox Mobile (10 Beta)
- Chrome para Android (Beta)
- Skyfire 4.1
- Opera Mobile 11.5

- Meego 1.2
- Samsung bada 2.0
- UC Browser
- Kindle 3 and Fire
- Nook Color 1.4.1
- Chrome Desktop 11-17
- Safari Desktop 4-5
- Firefox Desktop 4-9
- Internet Explorer 7-9
- Opera Desktop 10-11

Clase B: Compatibilidad completa pero sin transiciones animadas basadas en Ajax.

- Blackberry 5.0
- Opera Mini (5.0-6.5)
- Nokia Symbian 3

Clase C: Básico. Funciona pero no soporta HTML mejorado.

- Blackberry 4.x
- Windows Mobile
- Todos los otros teléfonos y dispositivos inteligentes

Matriz de soporte de Exploradores [19]

Plataforma	Version	Nativo	Opera Mobile				Opera Mini		Fenec		Ozone	Netfront	Phonegap
iOS	v2.2.1	B											A
	v3.1.3 v3.2	A						A					A
	v4.0	A						A					A
Symbian S60	v3.1 v3.2	C	C	C		B	C	B			C	C	

	v5.0	A	C	C		A	C	A					A
Symbian UIQ	v3.0 v3.1			C							C		
	v3.2				C						C		
Symbian Platform	3	A											
BlackBerry OS	v4.5	C					C	C					
	v4.6 v4.7	C					C	B					C
	v5.0	B					C	A					A
	v6.0	A						A					A
Android	v1.5 v1.8	A											A
	v2.1	A											A
	v2.2	A				A		C		A			A
Windows Mobile	v6.1	C	C	C	C	B	C	B				C	
	v6.5.1	C	C	C	A	A	C	A					
	v7.0	A				A	C	A					
webOS	1.4	A											A
bada	1	A											
Maemo	5	B				B			C	B			
Meego	1.1	A				A				A			

Tabla 3-5: Matriz de soporte de Exploradores

A: Calidad Alta: Explorador que por lo menos soporta media queries. Se harán pruebas sobre estos exploradores de forma permanente.

B: Calidad Media: Exploradores que soportan media queries pero que no son tan conocidos en el mercado como para garantizar pruebas de forma permanente. De todas formas, se garantiza la corrección de errores.
C: Calidad Baja: Exploradores que no soportan media queries. Estos no funcionaran adecuadamente con jQuery Mobile y se limitarán a HTML plano y CSS.

3.6 Apache Córdoba

Apache Córdova o anteriormente llamado PhoneGap, es un marco de desarrollo para móviles open-source desarrollado por Nitobi Software. Este permite a los desarrolladores construir aplicaciones para dispositivos móviles haciendo uso de JavaScript, HTML5 y CSS3, en lugar de lenguajes menos conocidos como Objective-C. La aplicación resultante es híbrida, es decir ni tan nativa (la visualización es realizada a través de webview en lugar de Objective-C), ni tan basadas en web (muchas de las funciones pueden ser soportadas por HTML5). Una desventaja de las aplicaciones híbridas es que no tienen un acceso completo a la API del dispositivo. [20]

	iOS iPhone / iPhone 3G	iOS iPhone 3GS and newer	Android	OS 4.6-4.7	OS 5.x	OS 6.0+	WebOS	WP7	Symbian	Bada
ACCELEROMETER	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
CAMERA	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
COMPASS	✗	✓	✓	✗	✗	✗	✗	✓	✗	✓
CONTACTS	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓
FILE	✓	✓	✓	✗	✓	✓	✗	✓	✗	✗
GEOLOCATION	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MEDIA	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗
NETWORK	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (ALERT)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (SOUND)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (VIBRATION)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
STORAGE	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗

Figura 3-7: Soporte de Apache Córdoba en diferentes sistemas operativos

3.6.1 PhoneGap Build [21]

El servicio de PhoneGap denominado PhoneGap Build, nos permite hacer una abstracción en SDKs, compiladores y hardware, al simplemente escribir la aplicación utilizando HTML, CSS y JavaScript, subirla a este servicio desde la web http://build.phonegap.com y utilizar la aplicación resultante inmediatamente en las tiendas de Apple iOS, Google Android, Pal, Symbian, Blackberry y más. Al compilar sobre la nube con PhoneGap Build, se pueden obtener todos los beneficios de un desarrollo multiplataforma desarrollando aplicaciones a nuestra manera.

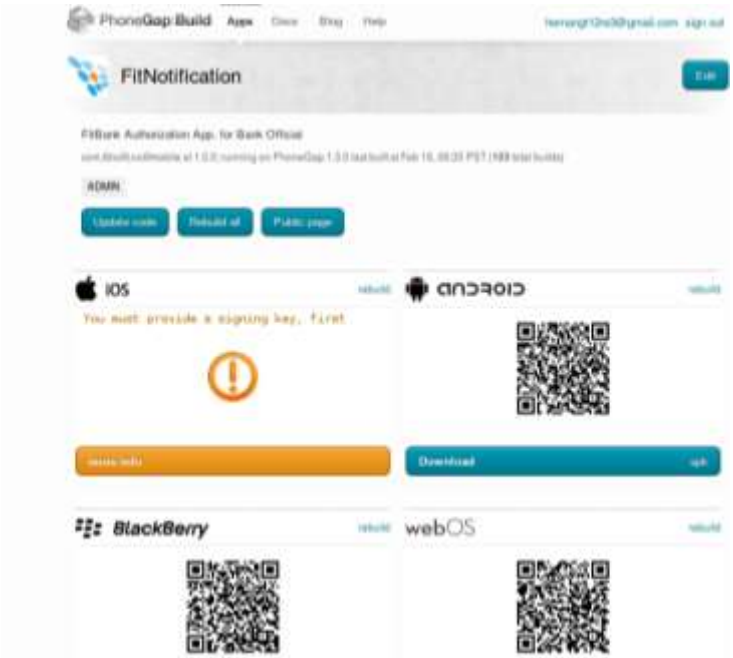


Figura 3-8: PhoneGap Build

3.6.2 Archivo de configuración config.xml [21]

Las aplicaciones construidas con PhoneGap Build pueden ser establecidas a través de la misma interfaz web del servicio, o mediante el uso del archivo de configuración config.xml. Este archivo, basado en la especificación W3C widget, permite a los desarrolladores establecer fácilmente metadatos para sus aplicaciones. Este archivo debe estar ubicado en el nivel superior de la aplicación (a nivel del archivo index.html), de otra forma no se cargará correctamente.

A continuación se describen las propiedades a poder usar en el archivo para la configuración de la aplicación:

Propiedades básicas:

- <widget>: el elemento widget debe estar en la raíz del documento XML, el cual nos da a conocer que se está utilizando la especificación W3C Widget. Al utilizar PhoneGap, hay que asegurarse de que se especifiquen los siguientes atributos:
- id: identificador único de la aplicación. Para soportar todas las plataformas disponibles, se debe utilizar el estilo de nombre reverse-domain, por ejemplocom.yourcompany.yourapp
- version: para mejores resultados se puede utilizar un estilo mayor/menos/parche

con tres números como 0.0.1

- versionCode: cuando se construye una aplicación para Android, se puede establecer el atributo versionCode dentro del config.xml.

- <name>: Nombre de la aplicación.
- <description>: Una breve descripción de la aplicación.

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8" ?>
<widget xmlns = "http://www.w3.org/ns/widgets"
xmlns:gap = "http://phonegap.com/ns/1.0"
id= "com.phonegap.example"
version   = "1.0.0">
<name>Ejemplo PhoneGap</name>
<description>
Ejemplo de PhoneGap Build.
</description>
<author href="https://build.phonegap.com" email="support@phonegap.com">
Tester
</author>
</widget>
```

Preferencias:

- Dentro de un archivo config.xml se puede agregar la etiqueta <preference>. Se puede tener cero o más de estos elementos presentes en el archivo config.xml. En el caso de que no se especifique alguna, se aplicarán las propiedades por defecto. La etiqueta <preference> posee los siguientes atributos:
    - name: valor requerido
    - value: valor requerido

- Version dePhoneGap: *phonegap-version*, con el número de versión de PhoneGap elegido. En el caso de que no se especifique la versión, la aplicación se establecerá por defecto con la versión actual en la que se la construya. Ejemplo:

  ```
  <preference name="phonegap-version" value="1.2.0" />
  ```

- Orientación del dispositivo: haciendo uso de *orientation* con los posibles valores: default, landscape, o portrait. El valor default significa que se permitirá tanto la

orientación landscape y portrait. Si se desea utilizar la propiedad por defecto de una plataforma, no se debe incluir esta propiedad en el archivo XML. Ejemplo:

<preference name="orientation" value="landscape" />

- Dispositivo específico: usando *target-device* con los valores handset, tablet, o universal. Esto es aplicable únicamente en iOS. Ejemplo:

  <preference name="target-device" value="universal" />

- Pantalla Completa: *fullscreen* con valores true o false. Esto es soportado en plataformas con iOS y Android. Ejemplo:

  <preference name="fullscreen" value="true" />

Específicos para iOS:

- WebView Bounce: *webviewbounce* con valores true o false. Controla cuando la pantalla "salta" cuando se desplaza cerca de la cabecera o fin de ella. Ejemplo:

  <preference name="webviewbounce" value="false" />

- Icono pre-renderizado: *prerendered-icon* con valores true o false. Ejemplo:

  <preference name="prerendered-icon" value="true" />

- Abrir todos los enlaces en un WebView: *stay-in-webview* con valores true o false. Ejemplo:

  <preference name="stay-in-webview" value="true" />

- Estilo de la barra de Estado: *ios-statusbarstyle* con los valores default, black-opaque o black-translucent. Ejemplo:

  <preference name="ios-statusbarstyle" value="black-opaque" />

Específicos para BlackBerry:

- Disable Cursor: *disable-cursor* con valores true o false. Esto previene el ícono del cursor sea mostrado en la aplicación. Ejemplo:

```
<preference name="disable-cursor" value="true" />
```

Específicos de Android:

- Mínima y máxima version de SDK: *android-minSdkVersion* y/o *android-maxSdkVersion*, con valores enteros. Estos valores son análogos a los que se establecen en el archivo AndroidManifest.xml. Los valores por defecto para minSdkVersion es 7 (Android 2.1), mientras que para maxSdkVersion es default. Ejemplos:

  ```
  <preference name="android-minSdkVersion" value="10" />
  <preference name="android-maxSdkVersion" value="15" />
  ```

- Ubicación de la instalación: *android-installLocation* con valores internalOnly, auto o preferExternal. Los dos últimos permiten que la aplicación sea instalada en una tarjera SD. Ejemplo:

  ```
  <preference name="android-installLocation" value="auto" />
  ```

Otros Elementos Útiles:

- Soporte para ícono: con el uso de la etiqueta <icon> se puede tener cero o más de estos elementos presentes en el archivo config.xml. Si no se especifica un ícono, PhoneGap utilizará su logo para el efecto. Esta etiqueta puede tener las siguientes propiedades:
- src: (requerido) especifica la ubicación de la imagen, relativa al directorio www.
- width: (opcional) recomendada su inclusión, con valor del ancho en pixeles.
- height: (opcional) recomendada su inclusión, con valor del alto en pixeles.

La utilización del ícono depende del tipo de plataforma en la que la aplicación va a ser instalada. Por defecto, para todas las plataformas, el ícono debe ser nombrado como *icon.png* y debe estar localizado en la raíz del directorio de la aplicación. Este tipo de configuración no asegura un funcionamiento adecuado en todas las plataformas. Ejemplo:

```
<icon src="icon.png" />
```

Para dispositivos con iOS, se tiene el soporte para pantallas clásicas, retina e iPad. Ejemplos:

```
<icon src="icons/ios/icon.png" width="57" height="57" />
<icon src="icons/ios/icon-72.png" gap:platform="ios" width="72" height="72" />
<icon src="icons/ios/icon_at_2x.png" width="114" height="114" />
```

Para Android se tiene el soporte para pantallas ldpi (240x320), mdpi (320x480), y hdpi (480x800). Ejemplos:.

```
<icon src="icons/android/ldpi.png" gap:platform="android" gap:density="ldpi" />
<icon src="icons/android/mdpi.png" gap:platform="android" gap:density="mdpi" />
<icon src="icons/android/hdpi.png" gap:platform="android" gap:density="hdpi" />
```

Para la plataforma BlackBerry los íconos deber ser de un tamaño menor a 16kb. A su vez se ofrece seleccionar otro tipo de imagen para un estado de “flote” o cuando se tiene seleccionado el ícono de la aplicación. Por defecto el ícono seleccionado será utilizado para este estado.

```
<icon src="icons/bb/icon.png" gap:platform="blackberry" />
<icon src="icons/bb/icon_hover.png" gap:platform="blackberry" gap:state="hover"/>
```

- Splash Screens o Pantallas de Inicio: se puede tener cero o más etiquetas <gap:splash>. Esta puede tener los atributos src, width y height, tal como el elemento <icon>. De igual forma que los íconos, estas imágenes deben ser de formato png. Si se quiere especificar esta propiedad para todas las plataformas, por defecto el archivo debe ser nombrado *splash.png* y ser colocado en la raíz del directorio de la aplicación. Por ejemplo:

```
<gap:splash src="splash.png" />
```

Ejemplos para dispositivos con iOS:

```
<gap:splash src="splash/ios/Default.png" width="320" height="480" />
<gap:splash src="splash/ios/Default_at_2x.png" width="640" height="960" />
<gap:splash src="splash/ios/Default-Landscape.png" width="1024" height="768" />
<gap:splash src="splash/ios/Default-Portrait.png" width="768" height="1024" />
```

Ejemplos para dispositivos Android:

```
<gap:splash src="splash/android/ldpi.png" gap:platform="android" gap:density="ldpi" />
<gap:splash src="splash/android/mdpi.png" gap:platform="android" gap:density="mdpi" />
<gap:splash src="splash/android/hdpi.png" gap:platform="android" gap:density="hdpi" />
```

Ejemplo para BlackBerry:

```
<gap:splash src="splash/bb/splash.png" gap:platform="blackberry" />
```

- Características de la API de PhoneGap: el elemento <feature> se puede utilizar para especificar que características del dispositivo utiliza la aplicación. Algunas de las características soportadas por PhoneGap son las siguientes:
  - http://api.phonegap.com/1.0/battery: permisos para android:BROADCAST_STICKY
  - http://api.phonegap.com/1.0/camera: permisos para android:CAMERA, winphone:ID_CAP_ISV_CAMERA, y winphone:ID_HW_FRONTCAMERA
  - http://api.phonegap.com/1.0/contacts: permisos para android:READ_CONTACTS, android:WRITE_CONTACTS, android:GET_ACCOUNTS, y winphone:ID_CAP_CONTACTS
  - http://api.phonegap.com/1.0/file: permisos para WRITE_EXTERNAL_STORAGE.
  - http://api.phonegap.com/1.0/geolocation: permisos para android:ACCESS_COARSE_LOCATION, android:ACCESS_FINE_LOCATION, android:ACCESS_LOCATION_EXTRA_COMMANDS, y winphone:ID_CAP_LOCATION
  - http://api.phonegap.com/1.0/media: permisos para android:RECORD_AUDIO, android:RECORD_VIDEO, android:MODIFY_AUDIO_SETTINGS, y winphone:ID_CAP_MICROPHONE
  - http://api.phonegap.com/1.0/network: permisos para android:ACCESS_NETWORK_STATE, and winphone:ID_CAP_NETWORKING
  - http://api.phonegap.com/1.0/notification: permisos para VIBRATE
  - http://api.phonegap.com/1.0/device: permisos para winphone:ID_CAP_IDENTITY_DEVICE

- La etiqueta <feature> es también utilizada para incluir plugins dentro del proyecto.

En el caso de que no se necesite algún tipo de permiso se debe añadir esta etiqueta de la siguiente manera (seguirá teniendo el permiso INTERNET en la aplicación, ya que PhoneGap la necesita):

```
<preference name="permissions" value="none"/>
```

Ejemplos:

```
<feature name="http://api.phonegap.com/1.0/battery"/>
<feature name="http://api.phonegap.com/1.0/camera"/>
<feature name="http://api.phonegap.com/1.0/contacts"/>
<feature name="http://api.phonegap.com/1.0/file"/>
<feature name="http://api.phonegap.com/1.0/geolocation"/>
<feature name="http://api.phonegap.com/1.0/media"/>
<feature name="http://api.phonegap.com/1.0/network"/>
<feature name="http://api.phonegap.com/1.0/notification"/>
```

- Elemento de Acceso: la etiqueta <access> permite proveer a la aplicación el acceso a recursos de otros dominios, en particular, permite a la aplicación cargar páginas de dominios externos sobre un webview. Una etiqueta vacía <access /> deniega el acceso a cualquier recurso externo, mientras que el comodín * de forma <access origin="*" /> permite el acceso a cualquier recurso externo. De otra manera, se puede especificar los dominios permitidos individualmente, incluidos subdominios, de la siguiente manera:

```
<access origin="https://build.phonegap.com" />
<access origin="http://phonegap.com" subdomains="true" />
```

Para asegurarse que los enlaces a los dominios tengan acceso únicamente desde el WebView, se debe utilizar el atributo browserOnly, que por defecto tiene un valor de false:

```
<access origin="http://phonegap.com" browserOnly="true" />
```

3.6.3 Firmado de la aplicación

Para que la API de PhoneGap Build pueda construir aplicaciones firmadas, es

necesario subir el archivo de firma obtenido en cada plataforma, en las preferencias del usuario:

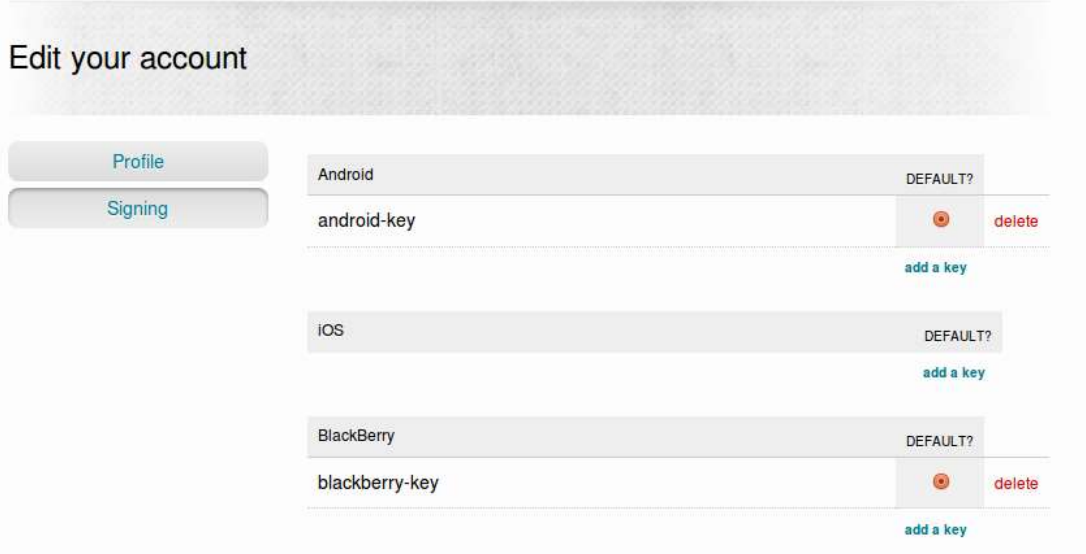


Figura 3-9: Firmado de aplicación en Android con PhoneGap

3.7 WURFL [22]

Una HTTP request involucra una petición de un cliente enviada a un servidor mediante su dirección IP. Esta petición posee una sección de encabezado y otra de cuerpo (este último es enviado generalmente cuando se realiza un petición de tipo POST).

El encabezado posee varios atributos definidos por el navegador y enviados al servidor (si no existe un proxy o gateway como intermediario). Algunos de estos atributos son:

- Header: Descripción
- User-Agent: El nombre del navegador o plataforma de donde se origina la petición.
- Accept:Una lista de valores separados por comas (CSV)de los tipos MIME que acepta el navegador.
- Accept-Charset: Una lista CSV con los tipos de codificación que acepta el navegador (por ejemplo ISO-8859-1, UTF-8)

- Accept-Language: Una lista CSV de los lenguajes preferidos por el navegador.
- Accept-Charset: Una lista CSV de los métodos de compresión disponibles.(por ejemplo gzip, deflate)

El atributo user-agent, actualmente puede llegar a contener hasta 6 diferentes definiciones de navegador al mismo tiempo y en el mismo string, debido a una larga historia de compatibilidades que los desarrolladores de estos han realizado, hasta el punto que estos difieren dependiendo del sistema operativo donde se ejecutan y el firmware utilizado. Esto hace que la detección de dispositivos utilizando el atributo user-agent se vuelva un poco complejo. Los siguientes ejemplos nos muestran el contenido de este atributo en navegadores de dispositivos como Nokia N95, Nokia 3510, Motorola v3, BlackBerry, iPhone 3.0 y Windows Mobile:

Mozilla/5.0 (SymbianOS/9.2; U; Series60/3.1 NokiaN95/20.0.015 Profile/ MIDP-2.0 Configuration/CLDC-1.1) AppleWebKit/413 (KHTML, like Gecko) Safari/413
Nokia3510i/1.0 (05.30) Profile/MIDP-1.0 Configuration/CLDC-1.0
MOT-V3i/08.B4.34R MIB/2.2.1 Profile/MIDP-2.0 Configuration/CLDC-1.1
BlackBerry8100/4.2.0 Profile/MIDP-2.0 Configuration/CLDC-1.1 VendorID/125
Mozilla/5.0 (iPhone; U; CPU like Mac OS X; en) AppleWebKit/420+ (KHTML, like Gecko) Version/3.0 Mobile/1A538a Safari/419.3
Mozilla/4.0 (compatible; MSIE 4.01; Windows CE; PPC; 240×320)

WURFL o por sus siglas en inglés Wireless Universal Resource File, es un DDR (Device Description Repository) o un Repositorio de Detección de Dispositivos, en el que se puede mapear el agente de usuario o user agent de una petición HTTP generada por un dispositivo móvil, a una descripción de las características soportadas por este. Comprende un conjunto de APIs y archivos de configuración XML que contienen información acerca de las capacidades y características de una variedad de dispositivos móviles. Desde su version 2.2, WURFL ha sido lanzado bajo licencia open source o de dominio público, ya que su información era contribuida por desarrolladores de todas partes del mundo. Posteriormente, con la conformación de ScientiaMobile, las APIs de WURFL fueron lanzadas tanto para licencias AGPL, como propietarias.

Gracias a esta interfaz, es posible desarrollar páginas de contenido utilizando cierta abstracción en sus elementos (por ejemplo botones, textos, etc.). En tiempo de ejecución, estos elementos pueden ser convertidos a una forma apropiada y especifica para cada dispositivo, simplemente mediante decisiones realizadas en base a las capacidades y características detectadas en el dispositivo; detección que es llevada a cabo por WURFL. Este contiene cerca de 500 capacidades para cada dispositivo, las que se encuentran divididas en 30 grupos. Además posee librerías de soporte para lenguajes como Java, PHP y .NET. Estas APIs hacen lectura del archivo wurfl.xml/wurfl.zip para poblar al DDR.

Capítulo 4: JBoss Server

4.1 Características

Jboss es un servidor de aplicaciones de código abierto que permite ejecutar aplicaciones hechas en Java que esta diseñado para ser flexible y rápido. A continuación se describen algunas de las características de este, en su versión actual (Jboss 7)

Velocidad sin precedentes: Jboss 7 cuenta con un proceso de inicialización optimizado, ya que los servicios se inician al mismo tiempo cuándo estos se inician por primera vez (de esta manera se evitan esperas innecesarias). Además, se aprovechan metadatos almacenados en caché, permitiéndonos así que incluso se pueda ahorrar tiempos en arranques posteriores.

Diseño Modular: Jboss 7 realiza la carga de clases de manera correcta. Para ello hace uso de módulos, que deciden que clases cargar y que clases no, evitando así el uso de clases innecesarias en nuestra aplicación.

Excepcionalmente ligero: Jboss 7 aprovecha muy bien el manejo de memoria para evitar pausas en el colector de basura de Java. Para ello, carga únicamente los JAR que se requieran y hace uso de metadatos indexados. Gracias a esto, el servidor puede correr en pequeños dispositivos permitiéndonos así mayor escalabilidad.

Administración elegante: La configuración de Jboss 7 es simple, está centralizada y orientada al usuario, ya que hace uso de un único archivo de configuración basado en un modelo de dominio directo que permite configurar varios servidores a la vez. (También se pueden configurar instancias individuales haciendo uso de un archivo similar). Jboss 7, además incluye una consola web, mediante la cual se puede realizar cualquier configuración de manera gráfica.

4.2 Uso y administración de Jboss [23]

Para la implementación de nuestra aplicación se hará uso del modo standalone (independiente), ya que cada módulo funciona por su cuenta. El servidor se configurará sobre Linux, y para lograrlo será necesario hacer lo siguiente:

1. Descargar y descomprimir el servidor de aplicaciones en cualquier carpeta del sistema.

2. Lo primero que vamos a hacer es probar que no tengamos ningún problema relacionado con Java. Para ello vamos a iniciar el servidor, entrando en la carpeta bin, y ejecutando el comando

   sh standalone.sh

Si todo salió bien, se nos indicará que el servidor se ha iniciado, y podremos acceder a él a través de la URL http://localhost:8080.

2. En el caso de que se quiera personalizar el servidor, lo único que tendremos que hacer es modificar el archivo standalone.conf.
3. Para deployar una aplicación lo único que tenemos que hacer es colocar el JAR/WAR bajo la ruta deployments (ya sea de la carpeta domain o standalone) e iniciar el servidor. Si queremos redeployarla, basta con sobre escribir el archivo.
4. En el caso de nuestra aplicación, se hace uso de hibernate para conectarnos con la base de datos. Y se requiere que se pueda acceder a la misma desde lugares remotos. Para lograr esto, tendremos que realizar varias modificaciones al archivo standalone.conf, nombrado anteriormente. Lo

primero que habrá que hacer, es indicar las características de nuestra base de datos, así como la información relacionada con el controlador de la BD de la siguiente manera:

```
<datasources>
    <datasource jndi-name="Tesis" pool-name="TESIS" enabled="true"
    jta="true" use-java-context="true" use-ccm="true">
            <connection-url>jdbc:mysql://localhost:3306/TESIS</connection-
            url>
            <driver-class>com.mysql.jdbc.Driver</driver-class>
            <driver>mysql-connector-java-5.1.17-bin.jar</driver>
            <security>
                    <user-name>root</user-name>
                    <password></password>
            </security>
    </datasource>
    <drivers>
            <driver                    name="mysql-connector-java-5.1.16.jar"
    module="com.mysql"/>
    </drivers>
</datasources>
```

Una vez hecho esto, tendremos que configurar al servidor para que acepte conexiones externas. Para ello tendremos que editar las interfaces indicando la IP en la que esta corriendo el servidor.

```
<interfacename="public">
        <inet-address value="127.0.0.1"/>
</interface>
```

5. Jboss 7 incluye varias librerías por defecto, sin embargo, muchas veces los controladores de las bases de datos no están incluidos. Es por esto, que en nuestro caso además de deployar nuestra aplicación, será necesario deployar el controlador de MYSQL.

4.3 Seguridades en Jboss [23]

En Jboss la seguridad se centra en mantener el sistema de administración lejos de intrusos. A continuación se detallan algunas características de este sistema y se explica como configurarlo. Más adelante, también se explica cómo hacer uso de conexiones seguras usando HTTPS.

Asegurando las interfaces de administración

Jboss 7 incluye dos interfaces de administración, mediante las cuales podemos modificar los diferentes archivos de configuración. Si queremos hacer uso de estas interfaces, para no tener que hacer todo a través de un editor de texto, tendremos que asegurarnos de que no cualquiera pueda acceder a las mismas.

Las interfaces que se incluyen son una interfaz HTTP, que es una consola de administración basada en GWT y al mismo tiempo permite ejecutar operaciones de mantenimiento usando JSON; y una interfaz nativa que permite ejecutar operaciones de mantenimiento usando un protocolo binario.

Las dos interfaces nombradas anteriormente se encuentran listadas en el archivo standalone.xml. Por defecto, la interfaz HTML usa el puerto 9990 y la interfaz nativa el 9999.

```
<management>
        <management-interfaces>
                <native-interface="management"port="9999"/>
                <http-interface="management"port="9990"/>
        </management-interfaces>
</management>

<interfaces>
        <interfacename="management">
                <inet-address value="127.0.0.1"/>
        </interface>
        <interfacename="public">
                <inet-address value="127.0.0.1"/>
        </interface>
```

```
        </interfaces>
```

Para asegurar estas dos interfaces, lo que hay que hacer es activar un método de seguridad (security realm). Para ello tendremos que agregar el código que se muestra a continuación dentro de la etiqueta “management”:

```
<security-realms>
    <security-realm name="PropertiesMgmtSecurityRealm">
        <authentication>
            <properties path="mgmt-users.properties"relative-
        to="jboss.server.config.dir"/>
        </authentication>
    </security-realm>
</security-realms>
```

Este código, se encarga de verificar los usuarios que se conecten contra el archivo mgmt-user.properties (localizado en la misma carpeta que el archivo de configuración). Por defecto, en el archivo no existen usuarios, por lo que será necesario agregarlos usando el formato: usuario=password.

Además, será necesario indicar qué método de seguridad utiliza cada interfaz. Para ello será necesario agregar la propiedad: security-realm=”PropertiesMgmtSegurityRealm” dentro de los dos atributos “interface”.

Finalmente, para iniciar el proceso que se encargará de la autentificación, tendremos que ejecutar:

```
./jboss-admin.sh --connect –file=scripts/secure-standalone-mgmt.cli
```

Para mantener a nuestro servidor seguro, también es recomendable deshabilitar el acceso remoto a este. Para ello lo único que se debe hacer es eliminar la siguiente línea del archivo de configuración.

```
<jmx-connector registry-binding="jmx-connector-registry" server-binding="jmx-
connector-server" />
```

Es importante que las aplicaciones que se vayan a ejecutar/deployar en el servidor, también se encuentren protegidas. Uno de los aspectos más delicados en el caso de nuestra aplicación, es el de la comunicación entre el cliente y el servidor, ya que al tratarse muchas veces de información delicada, esta no debería viajar en forma plana. Para lograr que la comunicación viaje encriptada, se puede hacer uso de HTTPS. A continuación se explica cómo lograr esto haciendo uso de Jboss.

Lo primero que tenemos que hacer es generar una llave privada. Para ello, podemos hacer uso de una herramienta llamada keytool de la siguiente manera:

keytool -genkey -alias fitnotification -keyalg RSA -keysize 1024 -keypass contraseña -keystore fitnotification.jks -storepass contraseña

Los parámetros que le pasamos a esta aplicación, indican:

- genkey: Que se va a generar una llave privada.
- alias: El nombre del certificado. Es recomendable que sea específico para la aplicación en la que se vaya a utilizar.
- keyalg: Indica el algoritmo que se utilizará para generar la llave.
- keysize: Indica el tamaño de la llave.
- keypass: Es la contraseña que va a tener el certificado.
- keystore: Es el archivo donde se va a guardar la llave.
- storepass: Es una contraseña para proteger el archivo. Se recomienda que sea diferente al keypass.

Una vez ejecutado este comando, se generará un archivo con el nombre fitnotification.jks. Lo siguiente que tendremos que hacer es generar un CSR (Solicitud de Certificado de Firma). Esto también se puede hacer utilizando keytool:

keytool -certreq -v -alias fitnotification -file csr.pem -keypass contraseña -storepass contraseña -keystore fitnotification.jks

Los parámetros utilizados son similares a los anteriores, pero en este caso, tenemos que indicar también el nombre del CSR usando -file.

Este último archivo, tendremos que certificarlo. Para ello, tendremos que mandarlo a alguna autoridad de certificación, la misma que nos devolverá el certificado firmado (csr-firmado.pem). Además, tendremos que conseguir el certificado raíz de la autoridad (csr-raiz.pem por ejemplo). Esto se puede hacer por ejemplo a través de Verisign (https://www.verisign.com/).

Finalmente, agregaremos al keystore, los 2 nuevos certificados que obtuvimos. Para ello hacemos:

keytool -import -v -trustcacerts -alias fitnotification -file csr-raiz.pem -keystore fitnotification.jks -storepass contraseña
keytool -import -v -alias fitnotification -file csr-firmado.pem -keystore fitnotification.jks -keypass contraseña -storepass contraseña

En el primer comando existe 1 parámetro nuevo (-trustcacerts) que indica que se está generando una llave con un certificado verdadero, en este caso, el certificado raíz de Verisign.

Una vez que tenemos nuestra llave privada lo único que nos queda por hacer es configurar Jboss para que haga uso del certificado. Para ello tendremos que editar el archivo standalone.xml y cambiar la línea:

<connector name="http" protocol="HTTP/1.1" socket-binding="http" scheme="http"/>

por:

<connector name="https" protocol="HTTP/1.1" socket-binding="https" scheme="https" secure="true">
<ssl name="https" key-alias =”fitnotification” password="contraseña" certificate-key-file="fitnotification.keystore"/>
</connector>

4.4 Servlets [24]

Un servlet es una clase del Java EE, y que conforma la Java Servlet API, un protocolo por el cual una clase Java puede responder a peticiones. Los servlets que están basados en el patrón de diseño MVC (Model-View-Controller) o Mediator, no están atados a un protocolo cliente-servidor especifico, pero son utilizados comúnmente en el protocolo HTTP. Estos son utilizados para extender las capacidades de los servidores de aplicaciones, por lo que un desarrollador podría utilizar un servlet para añadir contenido dinámico a un servidor web mediante el uso de la plataforma Java. El contenido generado es comúnmente HTML, pero también puede ser XML. La Figura 4-1 muestra el escenario de funcionamiento de un Servlet:

Figura 4-1: Funcionamiento de un servlet

Para desplegar y ejecutar un servlet, se debe utilizar un contenedor web, que esencialmente es un componente de un servidor Web que interactúa con este. El contenedor es responsable de la administración del ciclo de vida de un servlet, el mapeo de una URL a un servlet específico y el aseguramiento de que la petición correspondiente posea los permisos de acceso correctos.

El modelo M.V.C.

Este modelo, denominado en español, Modelo-Vista-Controlador, introducido por SmallTalk, fue utilizado para el desarrollo de interfaces de usuario. Dicho modelo, consta de tres componentes:

- *Modelo*: representa los datos u objetos de la aplicación. Es lo que se manipula y se presenta al usuario.
- *Vista*: es la representación en pantalla del estado actual del modelo.
- *Controlador*: define la forma en que la interfaz de usuario interactúa con la entrada del usuario. Este es el encargado de manipular el modelo.

La mayor ventaja del uso de este modelo es la separación de la vista y el modelo, pudiendo separar la presentación de la lógica de negocio, y así se podría lograr

cambiar las interfaces sin tener que cambiar el modelo, o la lógica del controlador. En definitiva, al hacer uso de M.V.C. los datos de la aplicación pueden ser representados por múltiples vistas.

Para implementar el modelo MVC en la parte del servidor, se podría combinar páginas JSP y servlets. En la Figura 4-2 se muestra una implementación, donde el Modelo es un JavaBean que representa los datos transmitidos/recibidos. El Controlador es un servlet que manipula/transmite los datos, y la Vista es una página JSP que presenta el resultado de la transacción. El proceso es el siguiente:

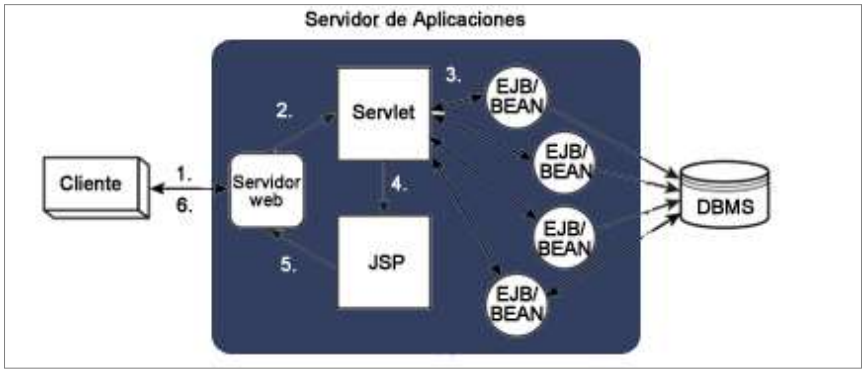


Figura 4-2: Implementación de un servidor

1. El cliente web hace una petición al servidor web
2. El servidor web pasa la petición al servlet (Controlador).
3. El Servlet realiza las operaciones necesarias a los EJB (Enterprise Java Beans) o el Modelo.
4. El controlador envía el resultado a la página JSP (Vista).
5. La vista da formato al modelo para ser mostrado y envía el resultado HTML al servidor web.
6. El servidor web transmite la información al cliente web.

Las fortalezas de los servlets como controladores son las siguientes:

- Los servlets poseen un procesamiento del lado del servidor muy robusto debido a que tienen acceso a todo el SDK de Java.

- La arquitectura de un servlet se presta muy bien a un estilo transaccional de la programación, lo que es muy análogo a los controladores MVC.
- Un servlet brinda eficiencia. La inicialización del código de un servlet es llevado a cabo una sola vez, cuando el servidor web lo carga. Una vez cargado solo es cuestión de llamar al método de servicio para atender nuevas peticiones.
- Los servlets ofrecen características de persistencia ya que pueden mantener un estado entre peticiones una vez este haya sido cargado.
- Ya que los servlets son desarrollados con lenguaje Java, estos son portables entre distintas plataformas, pudiendo ser movidos a otro ambiente sin necesidad de cambiar sus fuentes.
- Ya que Java es un lenguaje orientado a objetos, los servlets pueden ser extendidos en nuevos objetos para satisfacer las necesidades del desarrollador.

Los servlets brindan varias opciones de seguridad, ya que son ejecutados en el lado del servidor, es posible que puedan heredar la seguridad que provee el servidor web, o tomar ventajas de Java Security Manager. Las debilidades de los servlets como controladores son las siguientes:

- Los servlets requieren de un nivel avanzado de entendimiento del lenguaje Java, algo que los programadores de HTML usualmente no lo poseen.
- Los servlets generalmente requieren ser recompilados si existe algún cambio en su capa de presentación.

Arquitectura de un servlet.

La arquitectura define dos paquetes esenciales: javax.servlet y javax.servlet.http. El paquete javax.servlet contiene las clases e interfaces genéricas que son implementadas y extendidas por todo servlet. El paquete java.servlet.http contiene las clases que son extendidas al momento de crear servlets HTTP específicamente.

En el corazón de esta arquitectura se encuentra la interfaz javax.servlet.Servlet. Esta provee una estructura para todos los servlets, definiendo cinco métodos. Los 3 más importantes son init(),service() y destroy() cuya funcionalidad se explica más adelante. Todos los servlets deben implementar esta interfaz, ya sea directamente o mediante herencia. La Figura 4-3 muestra la estructura de la arquitectura:

Figura 4-3: Arquitectura de un Servlet

-

Al momento de desarrollar un servlet, se debe extender una de las dos clases principales, que son GenericServlet y HttpServlet. Cada vez que el servidor recibe una petición que apunta a un servlet, este llama al método service(), lee que tipo de método se utilizó para realizar la petición, y según este se llama al método doGet() o doPost(), que deben ser definidos en nuestro servlet, recibiendo los mismos parámetros que el método service(). El método service() viene definido genéricamente de la siguiente forma:

public abstract void service(ServletRequest req, ServletResponse res) throws ServletException, IOException;

Ciclo de vida de un servlet.

El ciclo de vida de un servlet de Java está basado en un diseño simple orientado a objetos. Un servlet es construido e inicializado. Posteriormente brinda un servicio a cero o más peticiones hasta que este servicio sea apagado. En este punto el servlet es destruido y el Garbage Collector de la JVM hace su trabajo. En definitiva, un

servlet es cargado una sola vez, con una instancia constante en memoria mientras esta atendiendo peticiones.

De este marco estructural de ciclo de vida se encarga la clase javax.servlet.Servlet, la cual define 3 métodos: init(), service(), destroy().

- init(): en este método es donde la vida del Servlet inicia. Es llamado inmediatamente y una sola vez en el momento en que el servidor hace una instancia al Servlet. El método crea e inicializa los recursos necesarios a utilizar en el manejo de las peticiones. Viene definido de la siguiente forma: *public void init(ServletConfig config) throws ServletException*.
- service(): este método maneja todas las peticiones enviadas por un cliente. No puede iniciar su servicio hasta que init() no haya sido ejecutado. Este método usualmente no es implementado directamente, solo en el caso que se extienda la clase abstracta GenericServlet. El uso más común es utilizar la clase HttpServlet, que implementa la interfaz Servlet extendiendo a GenericServlet, donde su método service() determina el tipo de petición enviado y llama al respectivo método. Viene definido de la siguiente forma:

public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException. El método implementa un paradigma de petición, respuesta. El objeto ServletRequest contiene información acerca de la petición encapsulando la información dada por el cliente. El objeto ServletResponse contiene la información que retorna al cliente en respuesta.

- destroy(): este método significa el fin del ciclo de vida del Servlet. Cuando el servicio es apagado, destroy() es llamado, limpiando todos los recursos que init() creó, por lo que es una buena práctica, limpiar todos los recursos utilizados en este método, como guardar toda información persistente y cerrar una conexión a una base de datos, por ejemplo. El método viene definido de la siguiente forma: *public void destroy();*

Manejo de la sesión en un servlet.

La mejor forma de manejar una sesión es mediante un seguimiento de esta. Definido en inglés como Session Traking, es la capacidad de mantener el estado actual de las secuencias de peticiones que realiza un cliente, sabiendo que el protocolo HTTP es stateless o no basado en estados, donde cada transacción es autónoma. Hay

distintas formas de hacer un seguimiento a la sesión de un cliente, las cuales describimos brevemente a continuación.

- *Uso de campos ocultos en un formulario: s*e basa en la creación de campos ocultos dentro de una etiqueta form de HTML, las cuales no son visibles al usuario pero son pasadas al servidor al realizar la petición.
- *Cookies:* una cookie es un par clave-valor, que es creado por el servidor y almacenado en el navegador del cliente. La API del Servet ofrece soporte para el manejo de cookies mediante la clase Cookie(String key,String value) basada en la RFC 2109, HttpServletResponse.setCookie(Cookie cookie) y HttpServletRequest.getCookie(String key).

Dentro del manejo de sesiones, las cookies deben ser establecidas antes de cualquier acción, ya que son almacenadas como cabeceras HTTP en la respuesta. También hay que tener en cuenta si el navegador soporta y/o tiene habilitado el uso de cookies.

- *Rescribir la URL:* si el navegador no posee el soporte para cookies, este método brinda otra buena alternativa y consiste en añadir a la dirección peticionada una identificación de sesión. El HttpServletResponse brinda los siguientes métodos para el uso de este método:

HttpServletResponse.sendRedirect(String
HttpServletResponse.encodeRedirectURL(String url))

- *Seguimiento de la sesión mediante la Servlet API:* Servlet API posee su propio soporte para realizar un seguimiento de la sesión mediante la clase HttpSession, la misma que posee 4 métodos para la realización de la misma. El método setAttribute(String name, Object value) ata un par nombre-valor y lo almacena en la sesión actual. Si el nombre ya existe, su valor es remplazado. El siguiente método getAttribute(String name) es utilizado para traer valores almacenados en la sesión. El tercer método getAttributeNames() retorna un arreglo de los nombres almacenados en la sesión. El último método removeAttribute(String name) elimina un valor de la sesión.

Para poder obtener la sesión actual en una petición, se debe invocar al método HttpServletRequest.getSession(). Además se debe tener en cuenta que la sesión

expira, por lo que es necesario consultar la documentación del servidor para determinar el tiempo de vida de esta.

HTTP Tunneling.

Un Túnel HTTP es un método para leer y escribir objetos serializados utilizando una conexión HTTP. Así se crea un subprotocolo dentro del protocolo HTTP, por ello su nombre.

Para llevar a cabo esto es necesaria la serialización de los objetos a ser enviados. La serialización es una característica que permite crear objetos que son persistentes a través de varios medios. Para hacer un objeto serializable se debe implementar la interfaz Serializable que se encuentra en el paquete java.io.package. Viene definida de la siguiente manera:

```
public interface Serializable {
        static final long serialVersionUID = 1196656838076753133L;
}
```

Los pasos necesarios para el envío de un objeto serializado son los siguientes:

- Crear un objeto OutputStream a un objetivo, que puede ser, por ejemplo, un archivo o una conexión TCP/IP.
- Crear un ObjectOutputStream pasándole a su constructor el OutputStream creado.
- Llamar al método writreObject(Object obj) de ObjectOutputStream, donde el objeto que se pasa debe implementar la interfaz serializable.

Los pasos equivalentes involucrados en la lectura de un objeto serializado son los siguientes:

- Crear un objeto InputStream que apunte a la ubicación de donde se estaría recibiendo el objeto serializado.
- Crear un ObjectInputStream pasando a su constructor el InputStream creado.
- Llamar al método readObject()de ObjectInputStream, el mismo que retorna un objeto que debe ser transformado al tipo de objeto original en el que fue enviado.

4.5 Hibernate (Persistencias)

Para poder crear aplicaciones organizadas tenemos que hacer uso de dos modelos al momento de programar. El primero, el modelo de objetos, hace uso de los principios de la abstracción, encapsulación, modularidad, herencia, concurrencia, polimorfismo y persistencia, y permite obtener aplicaciones bien estructuradas. El segundo, en cambio, llamado modelo relacional, ayuda a crear modelos en los que toda la información esta organizada e integrada. Cuando hacemos uso de un framework para mapear estos dos modelos (ORM), logramos tener acceso a las ventajas de ambos.

Entre los beneficios que podemos obtener al usar un ORM se encuentran: [25]

- Productividad, porque hacemos uso de metadatos que permiten reducir el tiempo de desarrollo.
- Capacidad de crear prototipos de manera muy rápida.
- Capacidad de mantenimiento, ya que la mayoría de trabajo se hace al momento de configurar, y se requiere escribir pocas líneas de código.
- Independencia, porque podemos abstraer nuestra aplicación de la base de datos y su dialecto.

Uno de los ORM más utilizados en la actualidad es Hibernate, este está compuesto de la siguiente manera:

- *Núcleo (Hibernate Core)*: se encarga de generar el código SQL y evita que tengamos que manejar la conexión con la base de datos manualmente. Cuando usamos el núcleo, los metadatos se almacenan en archivos XML.
- *Anotaciones (Hibernate Annotations)*: permite definir los metadatos directamente en el código de Java.
- *Gestor de Entidades (Hibernate EntityManager)*: ayuda a programar interfaces y a crear reglas para el ciclo de vida de los objetos persistentes.

Configurando Hibernate con Archivos de Mapeo. [26]

Una de las formas de usar Hibernate, es crear archivos, que permitan mapear el estado de una entidad en Java con las columnas de su correspondiente tabla en la base de datos. Para ello tenemos que definir las clases que queremos mapear y los archivos hbm, que usan el formato XML y tienen que tener una extensión específica (.hbm.xml).

Código en Java:

```
public class Usuario {
	private String id;
	private String contraseña;
	// Getters y Setters
}
```

Archivo hbm:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="com.tesis.persistencia">
<class name="Usuario" table="USUARIO">
	<id name="id" type="string" column="ID" />
	<property name="contraseña" type="string" column="CONTRASEÑA" />
</class>
</hibernate-mapping>
```

Además, es necesario indicar la configuración que se va a utilizar, esto podemos hacerlo de tres maneras:

- *Configuración programática*: En la que hacemos uso de la API para cargar el archivo hbm, el driver de la base de datos y especificar los detalles de conexión.

```
Configuration configuration = new Configuration()
.addResource("com/tesis/persistencia/Usuario.hbm.xml")
.setProperty("hibernate.dialect", "org.hibernate.dialect.MySQLDialect")
.setProperty("hibernate.connection.driver_class", "com.mysql.jdbc.Driver")
.setProperty("hibernate.connection.url", "jdbc:mysql://localhost:3306/TESIS")
.setProperty("hibernate.connection.username", "root")
.setProperty("hibernate.connection.password", "");
SessionFactory factory = configuration.buildSessionFactory();
```

- *Configuración XML*: En donde hacemos uso de un archivo XML (hibernate.cfg.xml) para especificar la conexión y el archivo hbm.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
```

```
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
<property
name="connection.driver_class">org.hibernate.dialect.MySQLDialect</property>
<property name="connection.url">jdbc:mysql://localhost:3306/TESIS</property>
<property name="connection.username">root</property>
<property name="connection.password"></property>
<property name="dialect">com.mysql.jdbc.Driver</property>
<mapping resource="com/tesis/persistencia/Usuario.hbm.xml" />
</session-factory>
</hibernate-configuration>
```

- *Configuración con archivos de propiedades*: Es similar a la configuración XML, pero hacemos uso de un archivo llamado hibernate.properties.

```
hibernate.connection.driver_class = com.mysql.jdbc.Driver
hibernate.connection.url = jdbc:mysql://localhost:3306/TESIS
hibernate.connection.username = root
hibernate.connection.password =
hibernate.dialect = org.hibernate.dialect.MySQLDialect
```

Una vez que tenemos esto, lo único que tenemos que hacer es crear una sesión de Hibernate, la misma que representa una transacción en la base de datos.

```
Session session = factory.openSession();
try {
        //Hacemos uso de la sesión
}catch(Exception e){
        e.printStackTrace();
} finally {
        session.close();
}
```

Para obtener los datos de un objeto simplemente tenemos que llamarlo, esto se puede hacer de varias maneras:

```
Usuario usuario= (Usuario) session.get(Usuario.class, id);

Query query = session.createQuery("from Usuario where id=?");
```

```
            query.setString(1, id);
Usuario usuario= (Usuario) query.uniqueResult();
```

A veces, cuando cargamos cierta información desde la base de datos, necesitamos modificarla. Es por esto, que hibernate facilita el método SaveOrUpdate, con el cual, es posible modificar estos datos (haciendo uso de Setters y Getters). Este método, además de permitir modificar la información que se encuentra en la base de datos, permite crear nuevos registros. Para hacer uso de esta función, lo único que tenemos que hacer es escribir el siguiente código dentro de nuestra clase de Java:

```
session.saveOrUpdate(usuario);
```

Configurando Hibernate con Anotaciones. [26]

Las anotaciones, como se explicó anteriormente permiten realizar el mapeo sin la necesidad de archivos extras. Sin embargo, al momento de realizar la configuración, si es necesario definir las clases que hemos mapeado haciendo por ejemplo uso de la etiqueta <mapping class="com.tesis.tablas.Usuario"/> si estamos usando un archivo de configuración XML. Las anotaciones son simplemente palabras reservadas, a las que les precede una “@”, y que se tienen que anteponer los métodos y propiedades de la clase.

En nuestro caso, la clase Usuario, con anotaciones quedaría así:

```
package com.tesis.tablas;
import javax.persistence.Column;
import javax.persistence.*;
import javax.persistence.Entity;
import javax.persistence.Table;

@Entity
@Table (name="USUARIO")
public class Usuario {
        @Column (name="id")
        @Id
        String id;

        @Column (name="contraseña")
        String contraseña;

        //Getters y Setters
```

}

Llaves primarias compuestas

Por lo general, cuando estamos trabajando con aplicaciones que manejan mucha información, es necesario crear campos primarios en donde interviene más de un campo. En hibernate, esto se puede lograr usando la etiqueta <composite-id> (si usamos archivos hbm). Sin embargo, también lo podemos lograr usando anotaciones (annotations). En este caso, lo que tenemos que hacer es crear una clase extra, en la cual estén declaradas todas las variables que pertenecen a la llave compuesta. Finalmente, para lograr que la llave compuesta se mapee correctamente, sólo tenemos que usar la anotación @Id al momento de hacer uso de dicha clase.

Relaciones entre tablas

Muchas veces, necesitamos crear relaciones en nuestra base de datos, en donde un campo de una tabla, puede verse involucrado con muchos registros de otra. En este caso, podemos usar hibernate para controlar de una mejor manera estas relaciones. Si usamos archivos hbm, tenemos que hacer uso de la etiqueta <many-to-one>, la misma que tiene que estar definida dentro del archivo hbm que mapea al elemento único (al hacer uso de esta etiqueta es importante indicar el campo por el cual las dos tablas se relacionan). También se puede hacer uso de esta ventaja, utilizando anotaciones (para ello tenemos que utilizar @ManyToOne).

Las relaciones de uno a uno también son muy comunes cuando creamos aplicaciones que usan bases de datos, para hacer uso de estas podemos crear etiquetas <one-to-one> (en archivos hbm) o utilizar la anotación @OneToOne

HQL (Hibernate Query Language)

Hibernate ofrece una nueva forma de armar peticiones a las bases de datos llamada HQL). Para hacer uso de este lenguaje lo único que tenemos que hacer es usar las sentencia Session.createQuery(); o EntityManager.createQuery();

Una de las características más importantes de este lenguaje es que permite utilizar paginación de una manera muy sencilla ya que podemos contralar el número de registros que queremos obtener utilizando los métodos:

query.setFirstResult(n);

```
query.setMaxResults(n);}
```

Además, al usar HQL, nuestra aplicación se vuelve más segura, ya que hacemos uso de variables enlazadas (Parameter Binding), que permite evitar ataques de Inyección SQL.

Capítulo 5: Seguridad de datos sobre la Red

5.1 Arquitectura genérica de autenticación (GAA) [3]

Para poder implementar sistemas con una Arquitectura genérica de autenticación (GAA) podemos hacer uso de algunos mecanismos. A continuación se describen algunos de ellos:

Infraestructura de llave pública (PKI)

La criptografía basada en llaves públicas fue inventada en los años 70 cuándo se crearon 2 nuevos conceptos:

- Separación de las contraseñas para encriptar y desencriptar, haciendo que una de ellas sea pública.
- Uso de firma digital, que permite verificar las autenticidad de un documento.

Se creía que el uso de llaves públicas iba a permitir que grandes sistemas se puedan manejar de forma muy simple. Cada entidad del sistema tendría su propia

llave privada, y la llave pública se podría proveer a cualquier otra entidad poniéndola en una base de datos. Sin embargo, poco después se observó que en sistemas muy grandes el manejo de contraseñas públicas a través de una base de datos no era escalable (Se volvía difícil almacenar las llaves de manera segura y no se podía garantizar que las llaves públicas eran auténticas).
Afortunadamente el uso de la firma digital proveía solución a estos problemas, ya que permitía que las llaves públicas sean auténticas.

La PKI se usa con mucha frecuencia en sistemas de escritorio, y puede ser implementada en dispositivos móviles también. El único problema cuando trabajamos con dispositivos móviles es que estos sistemas suelen ser globales y requieren de un mayor esfuerzo al momento de implementarlos.

Uno de los casos de uso más comunes de PKI se da en la capa de transporte (TLS), que se usa a través del protocolo RFC2246.

Para el desarrollo de esta aplicación no se hace uso de esta tecnología, pero si de los métodos indicados a continuación.

Contraseñas

Una de las formas más comunes de lograr autenticación en aplicaciones que se encuentran en el Internet, es hacer uso de contraseñas. En estos casos, cada usuario posee un nombre de usuario y una contraseña que se supone solo la conoce él y que se puede cambiar en cualquier momento.

Cuándo navegamos por Internet, y usamos contraseñas para iniciar sesión, tenemos dos alternativas:

- La primera, es hacer uso de un formulario HTTP en donde la página contiene los campos usuario y contraseña, los mismos que viajan junto a la página, y en donde se puede proteger la información haciendo uso de HTTPS (SSL o TLS).
- La segunda, es usar HTTP Basic y Digest, en donde Basic envía la información sin encriptar, y Digest se encarga de encriptar la contraseña haciendo uso de algún HASH (por ejemplo MD5).

Para el desarrollo de esta aplicación hacemos uso de estos dos métodos. Para que el usuario pueda iniciar sesión, este ingresa su usuario y su contraseña en un formulario HTTP, esta información se envía a nuestro servidor usando HTTPS y finalmente se encripta la contraseña con MD5 para compararla con la contraseña almacenada en la base de datos.

Kerberos

Kerberos es un sistema de manejo de contraseñas de propósito general basado en contraseñas compartidas. Este sistema fue desarrollado en los años 80 y se basa en un Centro de Distribución de Contraseñas (KDC) que comparte una llave maestra con todos los usuarios. A su vez, cada usuario debe disponer de un dispositivo que genera tickets/tokens temporizados, los mismos que permiten el acceso a ciertos servicios.

En cierto modo, se ha hecho uso de esta tecnología al momento de implementar este sistema ya que el usuario requiere de un token vigente para poder interactuar con el servidor.

Requerimientos para GAA

Para poder implementar un sistema con arquitectura genérica de autenticación (GAA) es necesario tener en cuenta lo siguiente:

- Generalidad: Es importante que nuestra arquitectura pueda ser utilizada por diferentes aplicaciones y servicios.
- Separación de aplicaciones: Las garantías de seguridad que provee una aplicación con GAA no debe depender del correcto comportamiento de otra aplicación con GAA. Cada aplicación debe ser independiente, de modo que una aplicación de servidor mal configurada no implique que se vulneren otras.
- Independencia de acceso: El uso de GAA no debe depender de una tecnología de acceso en particular.
- Reutilización: Al momento de implementar aplicaciones con GAA hay que darle importancia a la reutilización de protocolos e infraestructura existentes.

- Protección de la infraestructura original: Al momento de diseñar, implementar y configurar un GAA hay que evitar poner en riesgo la seguridad y operatividad del Sistema universal de telecomunicaciones móviles (UTMS).

5.2 HTTPS [3]

El HTTPS, también conocido como HTTP Secure (HTTP seguro) es una combinación de HTTP con el protocolo SSL/TLS. Este, nos permite obtener una comunicación encriptada y una identificación segura con un servidor Web. Por lo general, se hace uso de esta tecnología cuando se realizan transacciones de pago o transacciones sensibles en ciertos sistemas de información.

HTTPS es un esquema URI (Identificador de Recursos Uniforme) que tiene la misma sintaxis que HTTP, pero que hace uso de una capa de encriptación (SSL/TLS) para proteger la información transportada. La principal idea de esta tecnología es crear un canal seguro de comunicación, que evite el ataque de intrusos, cuando navegamos en una red insegura.

Para lograr esto, se hacen uso de certificados de autorización (VeriSign, Microsoft, etc.), los mismos que deberán ser aceptados por el usuario solo cuando:

1. El usuario sabe que el explorador implementa correctamente estos certificados.
2. El usuario cree que el certificado proviene de un sitio legítimo.
3. El sitio provee un certificado válido, el mismo que ha sido firmado por una autoridad de confianza.
4. El certificado identifica correctamente al sitio.
5. El usuario confía en que el protocolo SSL/TLS es suficientemente seguro como para evitar espías.

Cuando usamos HTTPS podremos darnos cuenta de algunas diferencias al momento de navegar. En primer lugar veremos que la URL en vez de empezar con http:// empieza con https://. Además, al usar HTTPS, no se hace uso del puerto 80 si no del 8443. Por otro lado, HTTP funciona en la capa de aplicación del modelo OSI (Capa de aplicación), mientras que HTTPS lo hace en una capa inferior, permitiendo que el mensaje se encripte antes de que se transmita y se desencripte después de llegar al otro lado.

Los mensaje transmitidos con HTTPS son protegidos por completo, ya que se encripta tanto la cabecera, como el contenido de la petición/respuesta. Lo único que un atacante podría llegar a conocer es que se está dando una comunicación entre las dos partes, el nombre del dominio y sus direcciones IP.

Actualmente, se puede conseguir certificados en la red tanto gratuitos como de pago. Sin embargo algunos exploradores no aceptan con verdaderos los certificados gratuitos (como CACert), por lo que al hacer uso de estos se podrían enviar advertencias al usuario final.

Capítulo 6: Navegadores

6.1 Web Kit engine [27]

Hoy en día las aplicaciones móviles se han vuelto muy populares. Web Kit es una herramienta que ha permitido que el estado del arte en este tipo de aplicaciones pueda adoptar nuevos estándares (como HTML5 y CSS3), al mismo tiempo que ha mejorado el rendimiento de estándares existentes (como JavaScript). Web kit es muy poderoso cuando se lo usa en ambientes móviles y es revolucionario cuando es el núcleo de cualquier celular que sale al mercado.

Lo bueno de Web kit, es que cuando se desarrolla una aplicación haciendo uso de este, nos es posible ejecutarla prácticamente desde cualquier dispositivo. De este modo, no es necesario implementar una versión para IPhone, una para Android y una para Blackberry, ya que la mayoría de dispositivos lo soportan de forma nativa.

Web Kit, es el motor del explorador, es decir, que se encarga de realizar el trabajo de bajo nivel del explorador. Entre otras cosas, se preocupa de cargar páginas, buscando la mejor forma de mostrarlas y de ejecutar los scripts que estás requieran. Actualmente los dispositivos/sistemas operativos que soportan Web Kit son Android, iOS, HP/Palm webOS, BlackBerry, Amazon Kindle y Nokia.

Partes de Web Kit

Web Kit está compuesto por tres partes:

Web Kit: Se encarga de comunicarse con el sistema operativo. Esta librería está sobre los otros componentes.

WebCore: Es el motor que esta detrás del explorador. Se encarga de cargar páginas web, parsear documentos HTML y XML, resolver los estilos CSS, diseñar las páginas web y pintarlas en la pantalla del explorador.

JavaScriptCore: Es el parser de JavaScript y se encarga de ejecutar este tipo de código en el explorador.

Ventajas de Web Kit

Mayor rendimiento: Con Web Kit los exploradores pueden renderizar páginas Web y procesar código JavaScript de manera mucho más rápida, logrando una mejor experiencia para el usuario.

Mejor soporte con respecto a estándares: Web Kit soporta varios estándares (HTML5, CSS3, SVG, Xpath, XSLT, entre otros).

Mayor conformidad entre navegadores: Todos los navegadores que usan Web Kit soportan los mismos estándares.

Mayor cantidad de desarrolladores: Debido a que Web Kit es un proyecto libre, los desarrolladores pueden beneficiarse del esfuerzo de un gran equipo de personas.

6.2 BlackBerry Browser [27]

El explorador de BlackBerry es un navegador diseñado para renderizar y soportar la mayoría del contenido Web existente en la actualidad. En la versión 6 de este navegador, se han generado varias mejoras, las mismas que se han podido dar gracias al uso de Web Kit.

BlackBerry Browser 6 soporta complemente HTML 4.01 y menores, WML1.3, CSS 2.1 y menores, ECMAScript3, JavaScript 1.6 y menores, DOM Level 2, AJAX, XML 1.0/1.1, XSLT 1.0 y Xpath 1.0. Además soporta de forma parcial estándares cómo HTML5, CSS3, SVG y RSS.

Por otro lado, entre los tipos de contenido que soporta este navegador, están:

Imágenes: El navegador puede mostrar en pantalla imágenes en formato bmp, gif, jpg, png y svg.

Audio: Para reproducir audio, el navegador hace uso de una aplicación Multimedia que soporta los formatos 3g2, 3gp, aac, amr, flac, m4a, midi, mp3, mp4, ogg, wav y wma. Sin embargo, se recomienda hacer uso de mp4.

Video: Para reproducir video, el navegador hace uso de una aplicación Multimedia que soporta los formatos 3g2, 3gp, asf, avi, m4v, mov, mp4 y wmv. Sin embargo, se recomienda hacer uso de mp4.

Archivos: En el BlackBerry Browser se pueden ver archivos en formato bbaw, doc, jad, kml, kmz, pdf, ppt, vcf, xloc y xls.

Esquemas de enlaces: Este navegador permite abrir páginas que usen http, https, mailto, pin, rtsp, sms y tel.

6.3 Safari [28]

El explorador Safari provee la interfaz para navegar en la Web haciendo uso de dispositivos que están basados en iOS. Y aunque el explorador de escritorio es similar al de dispositivos móviles, estos no son iguales. Al igual que el explorador de BlackBerry, este hace uso de Web Kit.

Algunas características de este explorador en dispositivos móviles son:

Los usuarios pueden cambiar la cantidad de contenido que tiene que mostrarse haciendo un acercamiento o cambiando la orientación del dispositivo.

Safari incluye soporte para cookies, permitiendo así una interacción del usuario más fluida.

Safari en iOS no soporta Flash, Java, ni otros plugins de terceros. Sin embargo, puede reproducir audio y video haciendo uso de HTML5, así como animaciones y transiciones gracias a JavaScript y CSS3.

Debido a que los dispositivos basados en iOS son táctiles, no existen algunas acciones tales como el hover (cuando tenemos el puntero sobre un enlace por ejemplo).

Las aplicaciones Web que se abren desde Safari se ejecutan en pantalla completa. Esto hace que parezcan aplicaciones nativas.

6.4 Android Browser [9]

Actualmente la última versión de este explorador es la 4.0. Algunas características importantes de esta versión son:

Mejoras en el rendimiento, especialmente en JavaScript. Permitiendo que las páginas se renderizen mucho más rápido

Permite la sincronización de marcadores con Google Chrome.

Permite que el usuario pueda abrir páginas Web en su versión de escritorio.

El usuario puede guardar las páginas Web y visitarlas sin estar en línea.

Capítulo 7: Ingeniería de software

7.1 Captura de Requerimientos

7.1.1 Características de la aplicación

La aplicación a desarrollar va a permitir que los oficiales de las distintas sucursales de una entidad financiera tengan la posibilidad de procesar autorizaciones financieras sin necesidad de estar en el banco y desde un dispositivo móvil. Es por esto, que esta aplicación deberá cumplir con las siguientes características:

-Debido a que la aplicación va a conectarse a la entidad financiera a través de Internet, esta tendrá que ser segura. Para ello habrá que hacer que la comunicación sea cifrada (HTTPS) y asegurarse de que no cualquiera pueda acceder a la aplicación. Es importante hacer uso de sesiones temporizadas.
-La aplicación tendrá que notificar al oficial cuándo este reciba una nueva autorización. Para ello se enviará un correo electrónico en el que se proveerá información básica sobre la autorización y un enlace para acceder a la aplicación.
-Cuando el oficial inicie sesión por primera vez, se le solicitará que ingrese una nueva contraseña y una palabra clave para “revivir” autorizaciones. Si el usuario olvida su contraseña o su token tendrá que solicitar a un administrador que restablezca sus credenciales.
-Para mantener a la aplicación segura, las contraseñas de los usuarios expirarán después de cierto tiempo (parametrizable) junto a la palabra clave de los mismos. Cuando esto suceda, al usuario se le pedirá que ingrese una nueva contraseña y una nueva palabra clave.
-Cuando el usuario inicie sesión en la aplicación este podrá ver un listado con las autorizaciones que ha recibido organizadas por subsistema (Personas, Vista , Plazo, Préstamos, etc.) y podrá acceder a las mismas para obtener mayor información y procesarlas.
-El usuario podrá acceder a la aplicación desde un número limitado de dispositivos. El número de dispositivos tendrá que ser parametrizable. Además, los oficiales no podrán acceder a la aplicación desde un equipo de escritorio.
-Las autorizaciones tendrán un tiempo límite para ser procesadas. Si este tiempo expira, el usuario tendrá que “revivirlas”. Para ello poseerá una palabra clave.
-Al momento de procesar la autorización al oficial se le mostrarán tres opciones: “Aprobar”, mediante la que se aceptará la misma, “Negar”, mediante la que se rechazará la petición y “Postergar” que simplemente caducará la autorización.
-La aplicación funcionará dinámicamente, de este modo la información que se cargue para cada autorización variará dependiendo del tipo de autorización.

-El usuario tendrá acceso a una pantalla en la cual podrá actualizar su información personal. A través de esta, se podrá modificar el Nombre y Apellido, el correo electrónico a la que se envían las notificaciones y la contraseña.
-Para que la aplicación pueda integrarse con cualquier sistema bancario, es importante desarrollar un módulo que permita la comunicación entre dichas aplicaciones. Este módulo será el único que se tenga que implementar en el caso de requerir funcionalidad con otro tipo de software. En el caso de FitBank, la comunicación se realiza a través de archivos XML por lo que este módulo tendrá que ser capaz de leer y escribir dichos archivos. Además, debido a que el manejo de autorizaciones se realiza a través de BPM, la aplicación tendrá que ser capaz de enviar respuestas que sean reconocidas por dicha tecnología.
-Para que la aplicación sea independiente, se tendrá que crear una base de datos en la que se almacene toda la información necesaria. En esta base de datos se almacenarán usuarios, contraseñas, autorizaciones, sesiones, parámetros, etc.
-Para mantener la consistencia entre la aplicación móvil y el software bancario, habrá que manejar procesos para migrar usuarios. De tal manera que cada vez que se añada un usuario en la base de datos de la entidad bancaria, este se añada automáticamente en la base de datos de la aplicación.
-La aplicación se ejecutará desde dispositivos móviles, es por esto que la información que se despliegue tendrá que ser concisa. Además, la navegabilidad dentro la misma tendrá que ser sencilla para evitar que el usuario cometa errores (especialmente al momento de procesar la autorización).
-La aplicación tendrá que ser compatible con los dispositivos móviles inteligentes más comunes del mercado por lo que funcionará bajo los sistemas operativos BlackBerry OS, iOS y Android.
-Se tendrá que desarrollar una aplicación extra que permita que un Administrador modifique ciertos parámetros de la aplicación como tiempo de vida de usuarios, sesiones y tokens, cantidad máxima de dispositivos por usuario, contraseña por defecto, entre otros. Además, el Administrador podrá expirar sesiones activas, reiniciar contraseñas de usuarios y eliminar autorizaciones que no se hayan procesado correctamente.

7.1.2 Autorizaciones

A través de la aplicación se podrá procesar cualquier tipo de autorización. Cuando el oficial reciba una autorización este recibirá una página con información general de la

autorización (persona, cuenta, oficial, tipo de autorización, etc.) y una o más páginas con información adicional más específica. Inicialmente la aplicación manejará las siguientes autorizaciones:

-02-3011: AUTORIZACIÓN DE CAMBIO DE OFICIAL. Esta autorización se ejecuta cuando se quiere modificar el Oficial de un usuario. La autorización se envía al nuevo oficial para que acepte o rechace su nuevo usuario. El oficial recibirá como información los oficiales previos del usuario y todas las cuentas que posee (ahorros, corriente, préstamos, plazos, etc.).
-04-2014: AUTORIZACIÓN DE SOBREGIROS CONTRATADOS: Esta autorización se ejecuta cuando un usuario solicita un sobregiro contratado. La notificación es enviada al oficial de la cuenta, junto con información sobre los saldos del cliente.
-04-2015: AUTORIZACIÓN DE SOBREGIROS OCASIONALES: Esta autorización se ejecuta cuando un usuario solicita un sobregiro ocasional. La notificación es enviada al oficial de la cuenta, junto con información sobre los saldos del cliente.
-04-2016: AUTORIZACIÓN DE SOBREGIRO PARA PAGO SOBRE CHEQUES: Esta autorización se ejecuta cuando un usuario solicita un sobregiro para pagar únicamente cheques. La notificación es enviada al oficial de la cuenta, junto con información sobre los saldos del cliente.
-04-2044: AUTORIZACIÓN AUMENTO DE VALOR DE SOBREGIROS CONTRATADOS: Esta autorización se ejecuta cuando un usuario solicita que se aumente el monto de su sobregiro contratado. La notificación es enviada al oficial de la cuenta, junto con información sobre los saldos del cliente.
-04-3040: AUTORIZACIÓN DE ELIMINACIÓN DE DOCUMENTOS: Esta autorización se ejecuta cuando se trata de eliminar uno de los documentos de un cliente. La notificación es enviada al oficial.
-04-2090: AUTORIZACIÓN PARA OBTENCIÓN DE CHEQUERAS. Esta autorización se ejecuta cuando el usuario solicita una nueva chequera si la anterior tiene un porcentaje de uso menora 90. La autorización tiene que ser aprobada por el usuario que tenga el rol “Activación de Chequeras”. Para procesar la información el usuario recibirá como información los saldos del usuario.
-04-6028: AUTORIZACIÓN DE PROTESTO DE CHEQUE: Esta autorización se ejecuta cuando un usuario con cuenta abierta posee un cheque que no tiene fondos. Esta autorización se envía al oficial de la cuenta, junto con información relacionada al cheque (número del cheque, valor, beneficiario, etc.). Si el oficial aprueba la autorización se actualiza el estado del cheque a “Protestado”.

-04-6031: AUTORIZACIÓN DE PROTESTO DE CHEQUE: Esta autorización se ejecuta cuando un usuario con cuenta cerrada posee un cheque que no tiene fondos. Esta autorización se envía al oficial de la cuenta, junto con información relacionada al cheque (número del cheque, valor, beneficiario, etc.). Si el oficial aprueba la autorización se actualiza el estado del cheque a “Protestado”.
-09-6000: AUTORIZACIÓN DE LEVANTAMIENTO DE GARANTÍAS: Esta autorización se ejecuta cuándo el usuario quiere levantar una garantía. La misma se envía al oficial de la cuenta. El oficial recibe información sobre la garantía que se va a levantar (Tipo de garantía, tipo de bien, costo, características, etc.)

7.2 Patrones de diseño [29]

Los patrones de diseño permiten generar soluciones a problemas, que se pueden implementar de manera sencilla en el código de cualquier proyecto, y que permiten que el mismo crezca y pueda ser reutilizado sin ningún problema. Nos pareció importante hacer uso de ellos en este proyecto para nuestra aplicación se pueda expandir en el futuro. A continuación se describen algunos de los patrones de diseño más populares, los mismos que serán utilizados al momento de implementar este trabajo de tesis.

Strategy.

El patrón Strategy se utiliza cuando es necesario usar código que controla tareas específicas dentro de la aplicación. Este patrón es una alternativa a la herencia, que evita el tener que crear muchas clases con código sobrescrito, que realizan tareas simples.

El patrón, consiste en extraer las partes volátiles del código y encapsularlas en objetos. Estos objetos se pueden utilizar cuando sean necesarios, y se pueden personalizar combinándolos en tiempo de ejecución haciendo uso de polimorfismo. Es por eso, que este patrón es orientado a las tareas.

Decorator.

El Decorator es un patrón que permite extender la funcionalidad de una clase; es decir, que luego de crear una clase, se pueden agregar clases adicionales

(decoradores) que se extienden de la original y que permiten realizar acciones diferentes sin necesidad de modificar la clase original una y otra vez.

Este patrón de diseño, permite escribir código que no se va a modificar, pero que se podría extender en el futuro. El decorator permite agregar nuevas funcionalidades a un objeto de manera dinámica y da una alternativa flexible a la creación de subclases.

Para crear decorators, lo primero que tenemos que hacer es crear una clase abstracta, que se extienda de la clase principal/original. Luego, lo único que tenemos que hacer es extender de esta clase, pudiendo generar cualquier cantidad de decoradores. Es importante, que los decorators sepan que es lo que van a modificar, por esto, es necesario que reciban como parámetro el objeto original.

Factory.

El patrón Factory permite codificar objetos que pueden crear otros objetos; en otras palabras, permite trabajar con objetos de distintas familias evitando que estas se mezclen entre sí y haciendo transparente el tipo de familia que se esté usando en cierto momento.

Este patrón requiere de una clase “Factory” que devuelve diferentes objetos, de acuerdo a ciertos parámetros. Además, hace uso de una clase abstracta, desde la cual se extienden cada una de las clases que se podrían generar. De esta manera, para decidir qué objeto crear, será necesario únicamente llamar a la clase “Factory” con los parámetros adecuados.

El patrón Factory, puede utilizarse de una forma más flexible, permitiendo que subclases se encarguen de instanciar las nuevas clases. Para lograr esto, es necesario definir la clase “Factory” de manera abstracta, de modo que cada una de las subclases (las mismas que crearán los objetos) tenga que implementarse a partir de esta.

Observer.

El patrón Observer permite enviar notificaciones para actualizar o configurar un grupo de objetos (observadores), los mismos que serán notificados cuando ocurra un evento en especial. Algo interesante de este patrón, es que permite añadir o eliminar observadores en tiempo de ejecución.

Al momento de usar este patrón, es importante hacer uso de interfaces, las mismas que aseguran que todos los objetos que se vayan a utilizar, sean compatibles. Será necesario crear tanto una interfaz para el Sujeto (Objeto que registra, elimina y notifica), como para los observadores (Objetos que se dan de alta o de baja con el Sujeto). Para poder registrar a los observadores se deberá hacer uso de algún tipo de estructura de datos (vector, lista, conjunto, etc.).

En Java se puede hacer uso de este patrón de diseño utilizando las clases Observable y Observer, las mismas que permiten implementar Sujetos y Observadores respectivamente.

Singleton.

El patrón de diseño Singleton consiste en instanciar solo un objeto, en base a una clase en particular. Es decir, que una clase solo puede tener una instancia, permitiéndonos obtener un único acceso a la misma. Este patrón es útil cuando se quiere restringir los recursos, cuando hacemos uso de objetos sensibles a los que no cualquiera puede acceder o cuando se requiere del uso de hilos en donde no queremos que existan conflictos al momento de manipular información.

Al usar este patrón, lo primero que se debe tener en cuenta es que el constructor debe ser privado, de este modo se evita que cualquier otra clase pueda acceder a nuestro código. Parece una locura hacer que el constructor de una clase sea privado, sin embargo se podría acceder al mismo haciendo uso de un utilitario que se encuentre dentro de la clase y que se encargue de instanciar (haciendo uso de este, se podría lograr que se cree solo una instancia a la vez).

Cuando trabajamos con hilos, este patrón de diseño se usa muy frecuentemente. Es más, Java hace uso de él cuando declaramos clases o hilos serializables.

Flyweight.

El patrón Flyweight es muy similar al Singleton, sin embargo, cuando hacemos uso de este, parece que estamos haciendo uso de varios objetos. Este patrón, en vez de trabajar con un montón de objetos individuales, los combina, permitiéndonos trabajar con una cantidad más pequeña de objetos genéricos, que pueden ser configurados en tiempo de ejecución. En otras palabras, al usar Flyweight estamos haciendo uso de un objeto compartido, que llega a ser independiente dependiendo del contexto desde el que se lo llame.

Lo interesante de este patrón es que el número de objetos genéricos se puede reducir tanto como queramos, logrando incluso que se haga uso de uno solo. Cuando hacemos uso de este patrón, es necesario dejar afuera todo el código "especializado", logrando que la clase funcione como una plantilla (la misma que podría "especializarse" en tiempo de ejecución).

Adapter.

Algunas veces, los objetos no pueden acoplarse entre sí. De esto trata de ocuparse este patrón de diseño, que busca lograr que una clase u objeto pueda adaptarse a otro. Al hacer uso del Adapter, no es necesario modificar de forma directa ninguna de las clases que queremos relacionar.

Este patrón es muy útil cuando queremos hacer uso de aplicaciones existentes dentro de nuestra aplicación, ya que permite convertir interfaces externas en nuevas interfaces que nuestro programa pueda entender.

El Adapter, tiene dos variantes. La primera, cuando trabajemos con objetos, en donde hacemos uso de la composición (el adaptador contiene al objeto que estamos acoplando), de modo que se pueda transformar la información cuando se llamen a los distintos métodos; y la segunda, cuando usamos clases, en donde es necesario hacer uso de herencia.

Facade.

Este patrón es similar al Adaptador, sin embargo, este simplifica la vida, ya que en vez de acoplar el código a nuestra aplicación, logra que el código externo sea más fácil de usar. Es decir que el Facade, únicamente simplifica la interfaz entre una clase u objeto, y el código que hace uso del mismo. La utilización de este patrón es muy común cuando no es posible rescribir el código de una aplicación.

Una de las desventajas de este patrón, es que si el código de la aplicación externa es modificado, va a ser necesario recodificar la clase que creamos para facilitarnos el trabajo.

Iterator.

El Iterator es ideal cuando estamos trabajando con una colección de objetos, y hoy en día, donde es muy común hacer uso de árboles, árboles binarios, hashes, listas, vectores y muchos más, se ha vuelto indispensable. La manera en que se accede a cada una de estas estructuras de datos es muy diferente, es por esto, que este

patrón es muy importante, ya que permite explorar cada una de ellas de una manera estándar.

En otras palabras, este patrón de diseño, permite hacer uso de diferentes estructuras de datos, sin necesidad de saber como estas funcionan internamente. Actualmente, Java tiene implementada la clase Iterator, la misma que define tres métodos (next, hasNext y remove), con los cuáles se puede acceder a algunas de las estructuras nombradas anteriormente.

Composite.

Este patrón es un poco más complejo que el Iterator, ya que permite recorrer estructuras de datos más complejas, en donde cada elemento de la estructura es una estructura más, que se puede recorrer de la misma manera. De esta forma, al llamar al método que se encarga de procesar la estructura principal, se procesan todos los hijos de la misma.

Para implementar este patrón, todos los elementos del árbol tienen que hacer uso de una sola clase abstracta, de modo que los métodos sean los mismos en cada nivel.

Este patrón es muy utilizado cuando tratamos de leer un archivo XML, en donde cada uno de los elementos se extiende de la estructura “Node”. De esta manera, solo se requiere de un método recursivo para poder recorrer todo el archivo.

Proxy.

Cuando creamos un objeto proxy, lo que estamos haciendo es permitir que nuestro código pueda comunicarse con un servidor remoto, sin que nos demos cuenta de esto. Sin embargo, este patrón se puede utilizar tanto para lograr que objetos remotos parezcan locales, como para restringir el acceso a objetos remotos de alguna forma.

Poner a funcionar un proxy en Java es muy sencillo. Lo más común es hacer uso del Internet y de alguna técnica de conexión, como RMI (Invocación de Métodos Remotos), sin embargo, si lo único que se va a enviar son mensajes de texto, lo más simple es usar sockets y streams. Además, cuando estamos implementando un proxy, siempre tenemos que hacer uso de hilos, de modo que tanto el servidor, como el cliente se mantengan todo el tiempo en escucha.

Command.

El patrón Command, permite encapsular diferentes acciones en objetos que están configurados para objetivos (servidores) específicos. Es decir, que obtendremos un número de objetos, que pueden trabajar en base a un grupo de herramientas, que están listas para ser utilizadas. De esta forma, cuando se quiera ejecutar una tarea, no será necesario llamar a métodos uno a uno, sino al comando previamente construido y configurado.

Algo interesante de este patrón, es que además de permitir encapsular las operaciones, también permite deshacer ciertas acciones.

Este patrón requiere la creación de receptores (objetos sobre los que trabaja el comando), los mismos que pueden ejecutar métodos personalizados y que van a implementarse en base a una interfaz previamente definida.

Una vez que tenemos implementados los receptores, será necesario definir nuestros Comandos. Estos objetos tendrán que recibir como parámetro un receptor, de modo que cuando se llame a este, se puedan ejecutar las acciones específicas de cada uno de los receptores. Al igual, que en los receptores, se tiene que crear una interfaz desde la cuál se implementarán cada uno de los comandos requeridos.

Por último será necesario implementar un invocador, que es el encargado de poner a trabajar a los comandos. El invocador, no es esencial cuando hacemos uso de este patrón, sin embargo, este es el que permite hacer uso de un log o de una cola, para así poder deshacer ciertas acciones.

Mediator.

El patrón de diseño Mediator permite tener en nuestra aplicación un procesador central, de modo que no todos los objetos tengan que comunicarse entre sí, si no sólo a través del Mediator. En otras palabras, permite encapsular en un objeto, la forma en la que un grupo de objetos interactúan, de modo que el mantenimiento estas acciones sea mucho más sencillo.

Al hacer uso del mediator, hacemos uso de estados, de esta forma, sabremos cuando es posible llamar de un objeto a otro, y cuando no. Cada uno de los objetos que interactuarán entre si, tendrán que recibir como parámetro al Mediator, y tendrán la capacidad de modificar el estado actual.

7.3 Análisis y diseño de la base de Datos

La base de datos contendrá las siguientes tablas:

TAUTORIZACIONES

Tabla que contiene la información de cada una de las autorizaciones que han sido enviadas a los oficiales.

NUMEROMENSAJE (VARCHAR): Id de la autorización. Es el mismo número que genera FIT.
FK2FHASTA (DATETIME): Fecha de caducidad del registro.
FKCESTATUSAUTORIZACIÓN (VARCHAR): Estado actual de la autorización. Esta definido en la tabla TESTATUSAUTORIZACION.
FKCTIPOAUTORIZACION (VARCHAR): Subsistema de FIT al que pertenece la autorización. Esta definido en la tabla TTIPOAUTORIZACIONES.
CTRANSACCION (VARCHAR): Transacción de FIT desde la que se envío la autorización.
NOMBRE (VARCHAR): Nombre de la autorización. Sirve para que el usuario conozca de inmediato de que se trata la autorización.
FKCUSUARIO (VARCHAR): Oficial al que se le envío la autorización.
CUSUARIO_ORIGEN (VARCHAR): Nombre de usuario de quién ejecuto la transacción.
FDESDE (DATETIME): Fecha de creación del registro.
FPROCESO (DATETIME): Fecha en la que se aprobó, negó o postergó la autorización.
FKTOKEN (VARCHAR): Token actual de la autorización.
MENSAJE (VARCHAR): Mensaje de error devuelto por FIT. Se almacena sólo cuándo FIT no pude procesar la autorización.

TESTATUSAUTORIZACIONES

Tabla en la que se almacena el estado de una autorización. Una autorización puede estar INGRESADA, APROBADA, NEGADA, ELIMINADA o puede ser un DETAIL de FIT.

CESTATUSAUTORIZACION (VARCHAR): Código del estado. Puede ser ING,APR,NEG,ELI,DET.
DESCRIPCION (VARCHAR): Descripción de cada uno de los estados.
FDESDE (DATETIME): Fecha de creación del registro.

TLOGAUTORIZACION

Tabla en la que se almacenan los archivos mediante los que se comunica la aplicación. (Tanto internamente cómo con FIT). En esta tabla se almacenan las autorizaciones en cada uno de sus estados.

NUMEROMENSAJE (VARCHAR): Id de la autorización. Es la misma que esta almacenada en la TAUTORIZACIONES.
CESTATUSAUTORIZACION (VARCHAR): Estado de la autorización.
MENSAJE_XML (LONGBLOB): Almacena el archivo en el que se esta transportando la información. Por lo general es un archivo XML.
ORIGEN_MENSAJE (BIT): Determina si el mensaje fue creado por la aplicación (true) o por FIT (false).

TPARAMETROSSISTEMA

Tabla en la que se almacenan propiedades de la aplicación.

FHASTA (DATETIME): Fecha de caducidad del registro.
CPARAMETRO (VARCHAR): Nombre del parámetro/propiedad.
FDESDE (DATETIME): Fecha de creación del registro.
VALOR CADENA (VARCHAR): Valor del parámetro en caso de que se almacene un VARCHAR.
VALOR ENTERO (INT): Valor del parámetro en caso de que se almacene un ENTERO.

TROLES

Tabla en la que se almacenan los roles de los usuarios. Permite determinar si el usuario que inicia sesión es un administrador de la aplicación o un oficial.

CROL (LONG): Código del rol.
DESCRIPCION (VARCHAR): Descripción de cada uno de los roles.
FDESDE (DATETIME): Fecha de creación del registro.
TSESIONES

Tabla en la que se almacenan las sesiones de los usuarios. Permite saber si el usuario se encuentra o sesión o no. Registra un historial de todas las veces que el usuario ha ingresado a la aplicación.

SESSIONID (VARCHAR): Número de sesión creado por la aplicación Web.
CUSUARIO (VARCHAR): Usuario que inicio sesión.
FINICIO (DATETIME): Fecha en la que el usuario inicio sesión.
FFIN (DATETIME): Fecha en la que el usuario cerro sesión. Si esta vacío quiere decir que el usuario se encuentra en sesión.

TTIPOAUTORIZACIONES

Tabla en la que se almacena el tipo de una autorización. Una autorización puede ser de PERSONAS, VISTA, PLAZO, PRÉSTAMOS, GARANTIAS, entre otras.

CTIPOAUTORIZACION (VARCHAR): Código del tipo de autorización.
DESCRIPCION (VARCHAR): Descripción de cada uno de los tipos.
FDESDE (DATETIME): Fecha de creación del registro.

TTIPOUSUARIOS

Tabla en la que se almacenan los tipos de los usuarios. Permite determinar si el usuario que inicia sesión es un administrador de la aplicación o un oficial.

CTIPOUSUARIO (VARCHAR): Código del tipo de usuario.
DESCRIPCION (VARCHAR): Descripción de cada uno de los tipos.
FDESDE (DATETIME): Fecha de creación del registro.

TTOKENS

Tabla en la que se almacenan los tokens de cada una de las autorizaciones.

TOKEN (VARCHAR): Número de identificación del token.
FDESDE (DATETIME): Fecha de creación del registro. Permite saber si el token esta caducado o no. El tiempo de caducidad del token se puede parametrizar.

TUSUARIODISPOSITIVOS

Tabla en la que se almacenan los dispositivos desde los cuales un usuario ha iniciado sesión.

MODELO (VARCHAR): Modelo del dispositivo.
ID (VARCHAR): Dirección IP desde la que se inició sesión.
FKCUSUARIO (VARCHAR): Usuario que inició sesión.
FHASTA (DATETIME): Fecha de caducidad del registro.
ESTADO (VARCHAR): Estado del dispositivo. Puede ser Activo (ACT) o Inactivo (DEL). Un usuario sólo puede tener un número determinado de dispositivos activos a la vez.
FDESDE (VARCHAR): Fecha de creación del registro.
FCONEXIÓN (DATETIME): Fecha de la última vez que se inició sesión desde el dispositivo.

TUSUARIOINFORMACIONADICIONAL

Tabla en la que se guarda toda la información del usuario.

FHASTA (DATETIME): Fecha de caducidad del registro.
FKCUSUARIO (VARCHAR): Usuario al que pertenece la información.
FDESDE (DATETIME): Fecha de creación del registro.
MAIL (VARCHAR): Correo electrónico del usuario. A esta dirección le llegarán las notificaciones al oficial.
NOMBRE (VARCHAR): Nombre completo del usuario.
KEYWORD (VARCHAR): Palabra clave del oficial para habilitar tokens caducados.

TUSUARIOPASSWORD

Tabla en la que se almacenan las contraseñas de los usuarios.

FHASTA (DATETIME): Fecha de caducidad del registro.
FKCUSUARIO (VARCHAR): Usuario al que pertenece la contraseña.
FCADUCIDAD (DATETIME): Fecha de caducidad de la contraseña. Este campo se genera de acuerdo a un parámetro en el que se indica la vigencia de las contraseñas. Si la contraseña caduca el usuario tendrá que actualizarla.
FDESDE (DATETIME): Fecha de creación del registro.
PASSWORD (VARCHAR): Contraseña del usuario. Se encuentra encriptada usando MD5.

TUSUARIOS

Tabla en la que se almacenan los usuarios que pueden acceder a la aplicación. Esta tabla se llena automáticamente cuando se crea un nuevo usuario en FIT.

CUSUARIO (VARCHAR): Nombre del usuario.
FKHASTA (DATETIME): Fecha de caducidad del registro.
ACTIVO (BIT): Determina si el usuario se encuentra activo o no.
FKCTIPOUSUARIO (VARCHAR): Tipo de usuario. Esta definido en la tabla TTIPOUSUARIOS.
FKCROL (INT): Rol del usuario. Esta definido en la tabla TROLES.
FDESDE (DATETIME): Fecha de creación del registro.
FVIGENCIADESDE (DATETIME): Fecha desde la que esta vigente el usuario.
FVIGENCIAHASTA (DATETIME): Fecha en la que el usuario dejará de funcionar. Este campo se genera de acuerdo a un parámetro en el que se indica la vigencia de los usuarios.

Persistencias

Para hacer uso de la información almacenada en la base de datos se hará uso de persistencias. En el caso de esta aplicación se usará hibernate. Las clases mediante las cuáles se podrá obtener y guardar información se listan a continuación:

com.fitnotif.persistence.tablas.TAutorizaciones
com.fitnotif.persistence.tablas.TEstatusAutorizaciones
com.fitnotif.persistence.tablas.TLogAutorizacion

com.fitnotif.persistence.tablas.TParametrosSistema
com.fitnotif.persistence.tablas.TRoles
com.fitnotif.persistence.tablas.TTipoAutorizaciones
com.fitnotif.persistence.tablas.TTipoUsuarios
com.fitnotif.persistence.tablas.TTokens
com.fitnotif.persistence.tablas.TUsuarioDispositivos
com.fitnotif.persistence.tablas.TUsuarioInformacionAdicional
com.fitnotif.persistence.tablas.TUsuarioPassword
com.fitnotif.persistence.tablas.TUsuarios
com.fitnotif.persistence.tablas.TSesiones

Diagrama de la base de datos

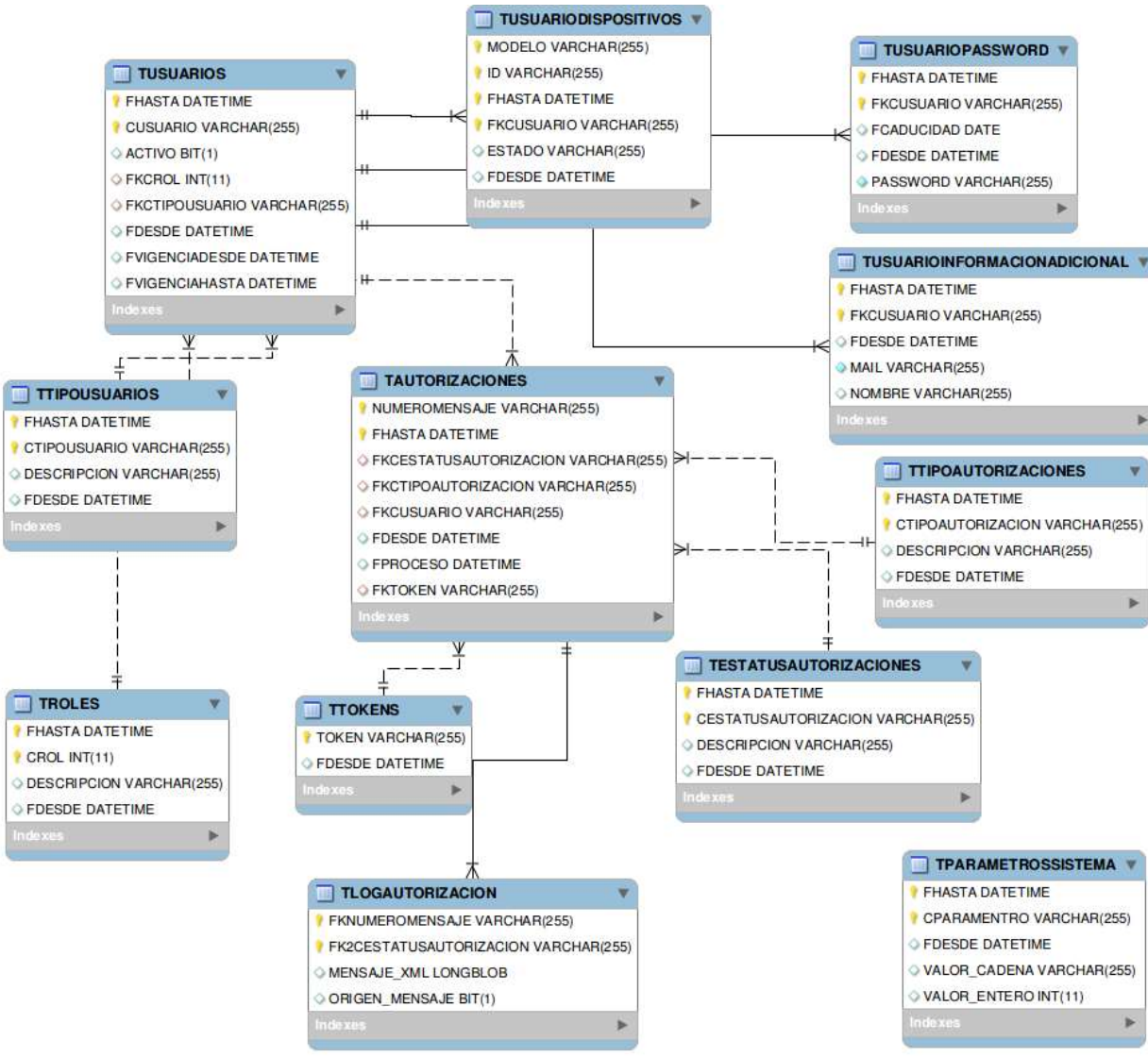


Figura 7-1: Diagrama de la Base de Datos

7.4 Análisis y diseño del módulo móvil e integración con el núcleo

7.4.1 BANTEC

Bantec Inc. es una empresa que ofrece sistemas financieros y que en la actualidad posee varios clientes tanto nacionales como internacionales. La empresa brinda soluciones orientadas a controlar y mejorar la rentabilidad del negocio financiero, las mismas que son multicompañía, multimoneda, multidioma; estando lista para soportar el crecimiento de una institución.

7.4.2 FITBANK

La aplicación a desarrollar tendrá que comunicarse con el CORE y el UCI de FIT. A continuación se detallan algunas características de estos dos sistemas:

FitBank UCI [30]

El UCI (Universal Channel Interface) de FitBank es una middleware diseñado para que las instituciones financieras puedan incorporar nuevos canales de comercialización de sus productos y servicios, permitiendo así la implantación de interfaces en línea con empresas externas de servicios. Además, permite la convivencia de varios sistemas de modo que la migración no sea un problema.

Algunos de los servicios que se pueden integrar haciendo uso del UCI son: Kioskos, Banca por Teléfono, Banca por Celular, Banca por Internet, ATMs, Centros de Atención Compartidos, Servicios de pagos y transferencias online y batch, transacciones vía e-mail, etc.

El UCI soporta distintos esquemas de comunicación, entre ellos TCP/IP (Sockets), SNA/SDLC (LU 0, LU 6.2), Serial Async (RS-232) y es capaz de procesar documentos XML y archivos planos en cualquier formato.

Para poder lograr procesar los distintos mensajes, el UCI sigue el siguiente proceso:

Normalización: Lo primero que se hace al recibir un mensaje, es normalizarlo a un formato XML.
Log y Seguridad: El mensaje normalizado se guarda en un log de entrada y se valida si el mensaje es correcto (es decir que tenga todos los parámetros necesarios). Si es incorrecto se devuelve un error y se almacena el archivo.
Conversión y Ruteo: Si el mensaje es correcto, se verifica si se requiere una conversión de formato (En caso de ser necesario se realiza este proceso).

UCI maneja todos los mensajes haciendo uso de multiples hilos, permitiendo indicar el número de conexiones simultáneas que se pueden dar. Además, hace uso de persistencia para poder tener un control total de los mensajes que llegan a cualquiera de los canales. Además, UCI tiene la capacidad de enviar un mismo mensaje a distintos canales de manera simultánea (sin importar que estos tengan el mismo formato), aunque es necesario indicar cuál es el mensaje oficial.

Core Bancario [31]

El Core Bancario de FitBank, es un sistema orientado a la rentabilidad, ya que nos permite saber cómo se comportan los clientes de la institución y cuáles de ellos son rentables para la misma.

Algunas de las características del sistema son:

Multicompañía, multimoneda, multidioma
Incluye plataforma de mercadeo y ventas integrada que soporta “credit scoring” y control de riesgo en línea
Permite la creación parametrizada de productos y servicios
Tiene un generador de transacciones incorporado (no se requiere programación para crear nuevas transacciones)
Soporta Banca por Internet, Banca telefónica, Kioscos, Celulares y PDAs
Es totalmente “non stop” puesto que esta disponible 7x24, no existen procesos ni de inicio ni de fin de día
Maneja imágenes de documentos
MIS incorporando nuevas normas del acuerdo de Basilea
Análisis multidimensional integrado tanto financiero como para mercadeo
Última tecnología, totalmente “web enable” y completamente basado en XML

Módulos

El Núcleo de FIT esta compuesto principalmente por el módulo de Contabilidad y de Clientes, y estos dos son alimentados de forma automática por los demás subsistemas. A continuación se detallan los módulos existentes:

- Contabilidad: Además de cubrir los requerimientos de las autoridades y de los organismos de control, proporciona a la gerencia del Banco la información necesaria para hacer un análisis de Activos y Pasivos.

- Clientes y CRM: Administra toda la información de los clientes permitiéndonos conocer las relaciones de un cliente con la institución y con los demás clientes, así como sus costumbres y preferencias de modo que se le pueda dar un trato adecuado.
- Cuentas a la Vista: Maneja las operaciones relacionadas con cuentas corrientes y de ahorros. Permite la creación de tantos productos y subproductos como requiera la institución. Además soporta el uso de distintos sobregiros, así como planes flexibles para el pago de intereses, también maneja cuentas solidarias.
- Cuentas a Plazo: Maneja la captación de recursos mediante Depósitos a Plazo y Certificados de depósito. Permite la creación de tantos productos y subproductos como requiera la institución.
- Colocaciones: Se encarga de todos los aspectos relacionados con créditos. Hace uso de flujos que hacen que la institución conozca el riesgo individual de cada operación. Se pueden crear tantos productos y subproductos como requiera la institución. Además, permite definir y controlar líneas de crédito y garantías.
- Inversiones: Permite automatizar la compra y venta de instrumentos, tanto de renta fija como variable.
- Tarjeta de débito: Es un producto complementario para las cuentas a la vista.
- Cajas: Permite automatizar todo lo relacionado con la atención a los clientes en ventanilla. Permite el uso de periféricos para que la transacción se realice de forma óptima y sin errores de digitación. Además soporta control de firmas.
- Remesas y Cámara: El módulo de remesas se encarga de todo lo relacionado con valores enviados al cobro a través de Bancos corresponsales. El módulo de Cámara permite enviar y recibir cheques. Estos módulos también capturan y almacenan las imágenes de los documentos.
- Plataforma de mercadeo y ventas: Permite que vendedores con un entrenamiento básico puedan comercializar con éxito todos los productos del Banco.

- Canales de Acceso: De forma paramétrica, el Banco puede decidir que transacciones estarán disponibles a través de otros canales (Internet, Kioskos, etc.).

Interfaz gráfico: Esta diseñado para ser usado a través de Internet de modo que el usuario pueda acceder desde cualquier navegador de manera clara y natural.

7.4.3 Flujo de funcionamiento la aplicación

A continuación se muestran dos gráficos que indican cómo se comportará la aplicación.

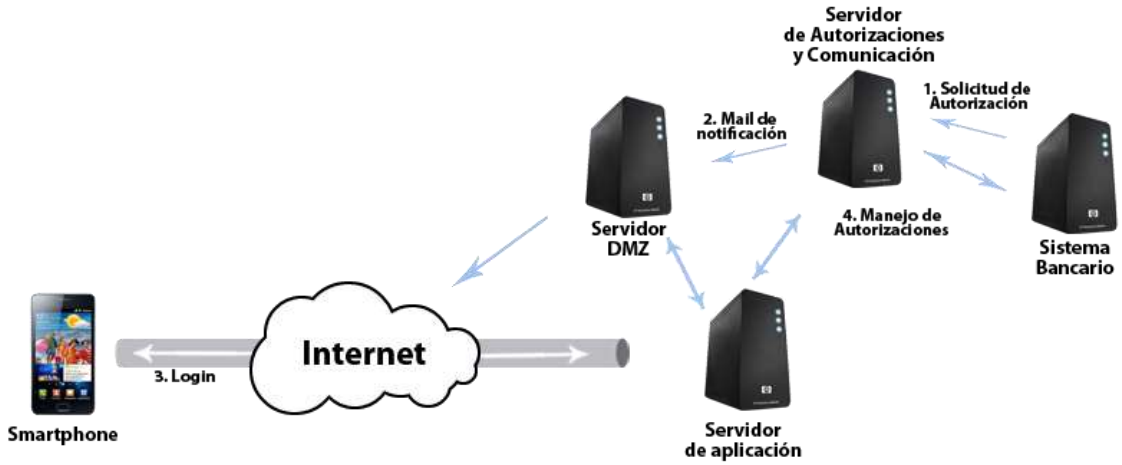


Figura 7-2: Funcionamiento externo de la aplicación

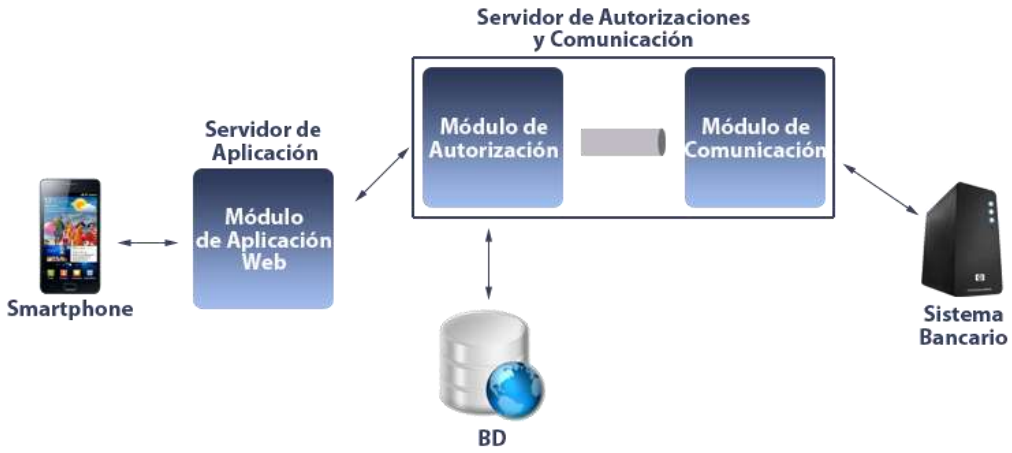


Figura 7-3: Funcionamiento interno de la aplicación

En los gráficos mostrados anteriormente se puede apreciar cómo se comunica la aplicación con el sistema bancario. Cómo se puede ver, existen 3 módulos:

El módulo de comunicación que permite comunicar el sistema bancario con el módulo de autorización/notificación.
El módulo de Aplicación Web que permite comunicar el módulo de autorización/notificación con el mundo exterior, es decir, con los dispositivos móviles. Esto se hace a través de una conexión segura (HTTPS).
El módulo de Autorización que comunica los dos módulos nombrados anteriormente con la base de datos. Este módulo también se encarga de enviar las notificaciones al usuario cuándo una autorización necesita procesarse.

7.4.4 Comunicación entre módulos

Para permitir una correcta comunicación entre los tres módulos de los que consta la aplicación se diseño un archivo XML. La estructura del mismo se muestra a continuación:

```
<AUTORIZATION>
        <HEADER>
                <ID></ID>
                <SID></SID>
                <IP></IP>
                <DEVICE></DEVICE>
```

```
                    <USER></USER>
            <ORIGIN_USER></ORIGIN_USER>
            <TYPE></TYPE>
            <TRANS></TRANS>
            <NAME></NAME>
            <DATE></DATE>
            <OP></OP>
            <STATUS></STATUS>
            <STACK></STACK>
        </HEADER>
        <DETAIL>
            <PAGE id="PAGE1">
                <REG id="0">
                    <COL name="COL1">VALOR1</COL>
                    <COL name="COL2">VALOR2</COL>
                    <COL name="COL3">VALOR3</COL>
                </REG>
            </PAGE>
            <PAGE id="PAGE2">
                <REG id="0">
                    <COL name="COL1">VALOR1A</COL>
                    <COL name="COL2">VALOR2A</COL>
                </REG>
                <REG id="1">
                    <COL name="COL1">VALOR1B</COL>
                    <COL name="COL2">VALOR2B</COL>
                </REG>
            </PAGE>
            <PAGE id="CONTROL">
                <REG id="0">
                    <COL name="CONTROL1">VALOR1</COL>
                    <COL name="CONTROL2">VALOR2</COL>
                </REG>
            </PAGE>
        </DETAIL>
</AUTORIZATION>
```

7.5 UML

La aplicación desarrollada consta de 3 actores principales:

- Sistema Bancario: Es el sistema de la entidad bancaria en la que se va a implementar esta nueva funcionalidad. En este caso son el CORE y el UCI de Fit-Bank.
- Usuario: Son las personas que van a hacer uso de la aplicación. Básicamente se van a dividir en Oficiales y Administrador.

A continuación se muestran los diagramas de Casos de Uso y de Secuencia, mediante los cuáles se puede entender de mejor manera cómo se relacionan los actores de la aplicación.

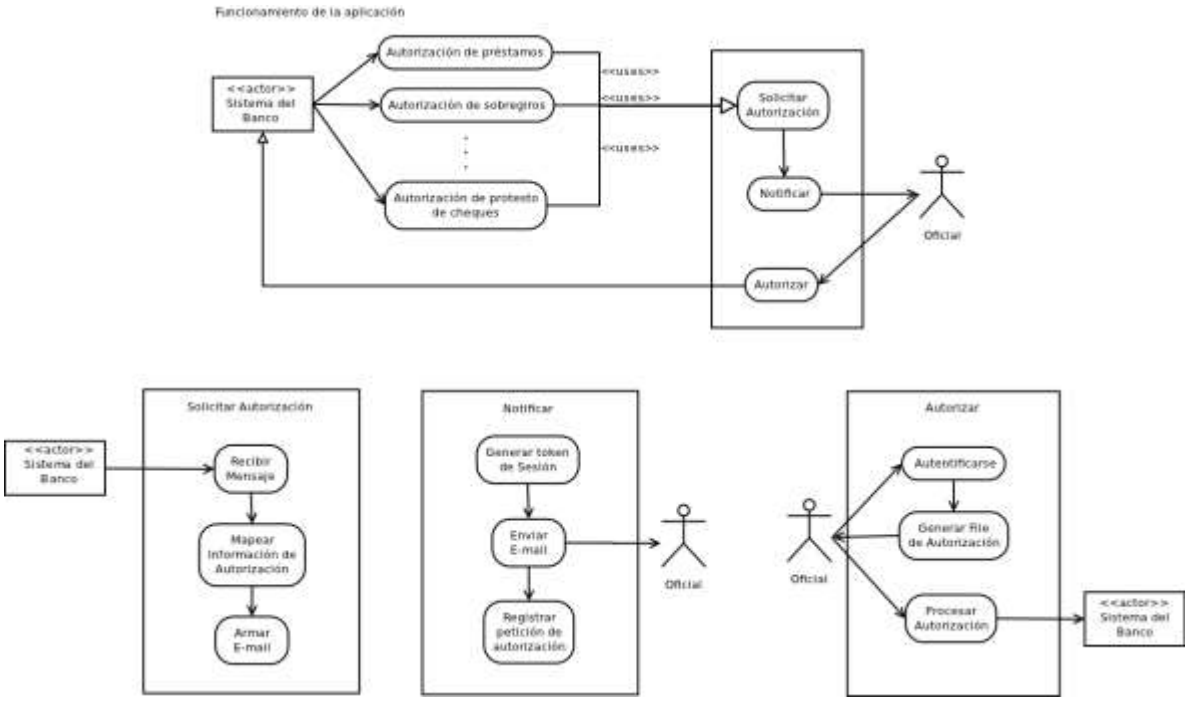


Figura 7-4: Diagrama de Casos de Uso

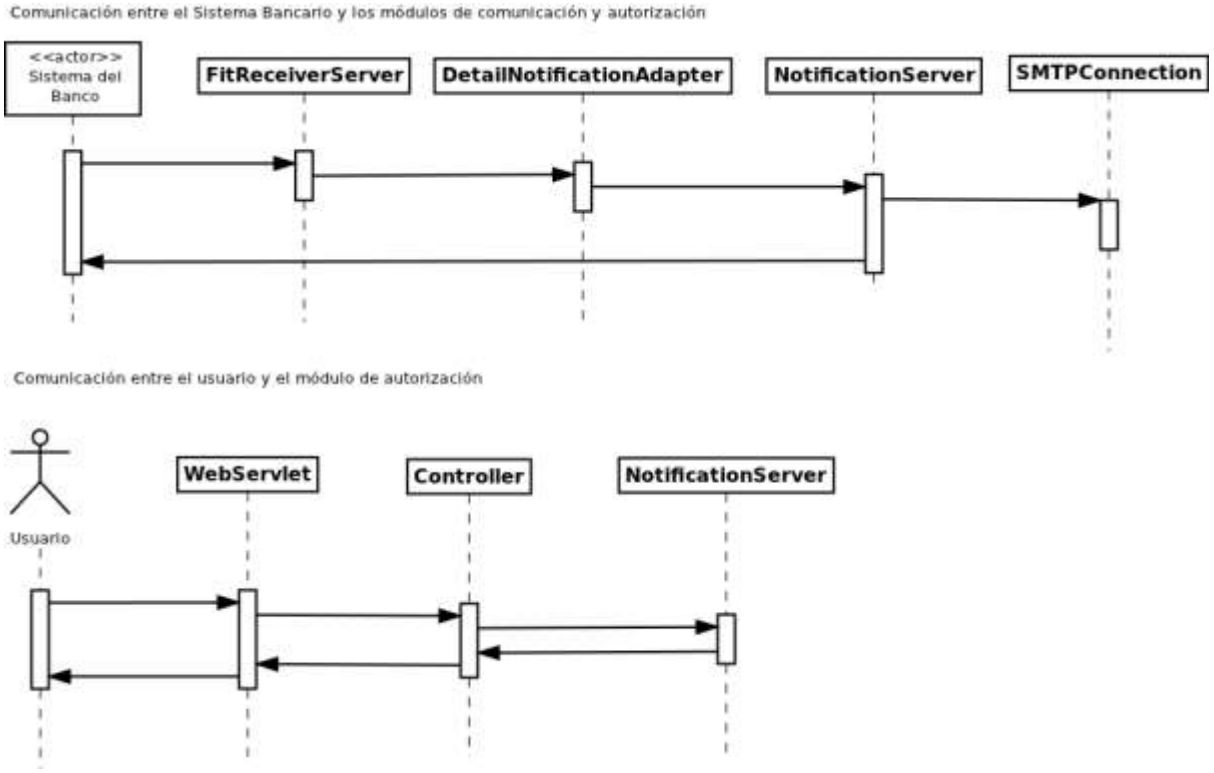


Figura 7-5: Diagrama de Secuencia

En los anexos, se podrá encontrar los diagramas de clases de la aplicación.

7.6 Herramientas a utilizar

7.6.1 Subversion [32]

Subversion (SVN) es un sistema de control de versiones (CVS) libre que nos permite manejar archivos a lo largo del tiempo. Para hacer uso de SVN, es necesario que creemos un repositorio central, en donde se almacenarán todos los archivos y todo su historial. Subversion permite que accedamos al repositorio a través de la red, por lo que este puede ser modificado por varios usuarios, permitiendo que varias personas puedan trabajar en un mismo proyecto al mismo tiempo.

Características

- Control de versiones en directorios: Subversion a diferencia de muchos CVS permite no solo el control de archivos, si no también de directorios completos.
- Historial de versiones real: Cuándo usamos Subversion, cada archivo se crea con un historial limpio y nuevo. Esto permite que podamos añadir, borrar, copiar y renombrar archivos y directorios sin afectar el historial de otros archivos.
- Commits atómicos: Subversion nos permite escoger que archivos queremos enviar al repositorio y que archivos.
- Control de versiones de Metadatos: Las propiedades de cada archivo o directorio también son almacenadas a lo largo del tiempo.
- Elección de capas de red: Subversion permite que el usuario escoja la forma en la que se accede al repositorio. Podemos hacer uso desde un servidor de Apache hasta de un túnel con SSH.
- Manejo de la información consistente: Subversion hace uso de un algoritmo de diferencias binario que funciona tanto con archivos binarios como con archivos de texto plano.
- Ramificación y etiquetado eficientes: Subversion permite crear sucursales y etiquetas de manera rápida ya que hace uso de un mecanismo similar al de un enlace.
- Usabilidad: Subversion esta implementado en base a librerías de C y APIs bien definidas. Esto permite que sea fácil de mantener y que se pueda utilizar desde otras aplicaciones o lenguajes.

Instalación y Configuración

Para instalar subversion, lo más simple es descargar el binario que se ofrece en el sitio web (http://subversion.tigris.org). Sin embargo, en el caso de Linux, también se puede instalar haciendo uso del gestor de aplicaciones. En el caso de Ubuntu/Debian, basta con ejecutar

sudo apt-get install subversion libapache2-svn

El comando anterior, nos instalará los siguientes componentes;

- svn: El cliente de subversion basado en línea de comandos.
- svnversion: Una aplicación para reportar el estado de una copia de trabajo.

- svnlook: Una herramienta para inspeccionar el repositorio.
- svndumpfilter: Un programa para filtrar ciertos flujos.
- mod_dav_svn: Un módulo de Apache que permite que el repositorio se pueda acceder a través de la red.
- svnserve: El servidor de subversion, que funciona como un demonio.

Una vez instalada la aplicación, tendremos que crear nuestro repositorio e importar nuestra información. Para ello tenemos que ejecutar:

svnadmin create /repositorio
svn import /proyecto/a/importar /repositorio -m “Importación Inicial”

Cuando creamos nuestro repositorio, no podremos observar el contenido de este directamente desde el repositorio. Si queremos obtener una copia de trabajo tendremos que hacer lo siguiente:

svn checkout /repositorio /copia/de/trabajo

Problemas del control de versiones

Todos los sistemas de control de versiones tienen que preocuparse por un problema fundamental: ¿Cómo hacer que los usuarios puedan compartir la información y evitar que por accidente un usuario cambie algo que otro usuario ya ha cambiado al mismo tiempo?

Subversion, se encarga de esto haciendo uso del modelo Copiar-Modificar-Mezclar (Copy-Modify-Merge). Este modelo se basa en que cada usuario tiene un repositorio propio (local) en donde puede hacer los cambios que requiera. El usuario podrá enviar sus cambios al repositorio cuando desee, sin embargo, estos cambios se mezclaran automáticamente con los que hayan sido realizados por otras personas evitando que se pierda código.

Habrá veces, en las que este proceso no se pueda completar satisfactoriamente debido a la existencia de conflictos (cuando 2 usuarios han modificado el mismo fragmento de código), en cuyos casos tendremos que desechar el código de una de las 2 partes.

Revisiones

Cada vez que queramos enviar nuevo contenido al repositorio (cambios que hemos realizado), tendremos que hacer un commit (svn ci -m “Mensaje”), y cada vez, que ejecutemos esta operación en nuestro repositorio se creará una revisión (número natural que indica la versión del repositorio).

Hay que tener en cuenta, que en subversion, las revisiones se aplican sobre todo el repositorio, y no sobre archivos individuales como en varios sistemas de control de versiones. Es por eso que a lo largo del tiempo puede haber varios archivos iguales con distintos números de revisión.

Operaciones comunes

Subversion tiene una gran cantidad de características y opciones, pero en el día a día sólo se necesitan hacer uso de algunas de ellas. A continuación se listan las operaciones más comunes que se realizan al trabajar con SVN:

- svn update: Actualiza la copia de trabajo.
- svn add:Agrega archivos al repositorio
- svn delete: Elimina archivos del repositorio
- svn copy: Copia un archivo dentro del repositorio
- svn move: Mueve un archivo dentro del repositorio
- svn status: Muestra el estado de la copia de trabajo
- svn diff: Muestra las diferencias entre la copia de trabajo y el repositorio
- svn revert: Revierte los cambios que se hayan realizado.
- svn commit: Envía cambios al repositorio. Siempre se tienen que acompañar por un mensaje usando el atributo --message

Cuando ejecutamos los comandos svn update, nuestra copia de trabajo se actualizará y se listarán los archivos que se han modificado acompañados de una letra. Esta letra indica lo que se ha hecho con el archivo:

U: El archivo se ha actualizado
A: El archivo se ha agregado

D: El archivo se ha borrado
R: El archivo se ha remplazado
G: El archivo se ha mezclado
C: El archivo tiene conflictos.

Sucursales (Branches) y Mezclas (Merging)

Las sucursales y las mezclas son conceptos muy comunes en casi cualquier sistema de control de versiones. A continuación se explica de qué tratan estos conceptos, y cómo se puede hacer uso de ellos en Subversion.

Las sucursales son líneas de desarrollo de un proyecto independientes entre sí y que varían en cosas muy específicas. Una sucursal siempre nace como una copia de trabajo, y a partir de ahí empieza a generar su propio historial. Subversion incluye algunos comandos que nos ayudan a tener un mejor control de las sucursales, y gracias a ellos podemos crear nuevas sucursales o duplicar ciertos cambios en distintas sucursales.

Para crear una sucursal lo único que tenemos que hacer es una copia del repositorio. Esto se puede hacer de dos maneras:

svn copy /directorio/origen /directorio/destino
svn commit -m “Nuevo repositorio.”
svn copy /url/origen \ /url/destino -m “Nuevo repositorio.”

Es importante saber, que cuando usamos cualquiera de estos 2 métodos, todos los archivos/directorios se copian recursivamente.

Una vez creada la sucursal, se puede trabajar sobre ella sin que se altere la información original en las otras sucursales. Además, es posible cambiarse entre las distintas sucursales mientras desarrollamos gracias al comandos svn switch.

Si en algún punto, se requiere copiar ciertos cambios hacia la copia de trabajo original será necesario hacer uso de las mezclas (svn merge). Cuando usamos este comando, tenemos que indicar las revisiones de las que se quiere hacer el merge, y los directorios de origen y destino.

svn merge -r 1:2 /directorio/origen /directorio/destino

Para el desarrollo de esta aplicación, se usarán dos sucursales. La primera “desarrollo” en la cual se irá creando todo el código y sobre la cual se harán todos los cambios, y la segunda “control”, en la cual se irá copiando el proyecto en desarrollo cuando se lleguen a ciertos puntos de control.
Además, existirá una sucursal en la que se copiará el código final de la aplicación (release) a la que llamaremos trunk. En la Figura 7-6 se muestra el diseño del repositorio.

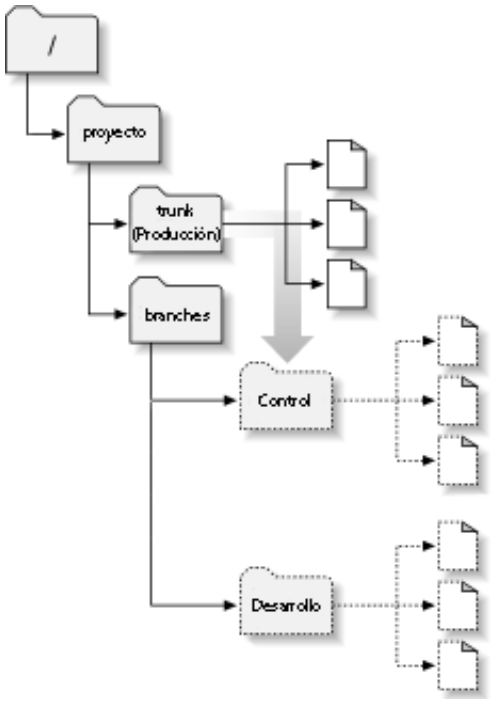


Figura 7-6: Diseño del repositorio

Configuración del servidor

Al montar un servidor de Subversion, es importante que puedan acceder al mismo tiempo cualquier cantidad de clientes, y desde cualquier lugar del mundo. A

continuación se explica como configurar el repositorio de modo que pueda manejarse usando clientes remotos.

Para iniciar nuestro servidor de SVN lo único que tenemos que hacer es ejecutar el comando svnserve -d. (Se usa -d para ejecutar el servidor como un demonio, si se desea se deben especificar también los atributos –listen-port y --listen-host).
Credenciales

Uno de los aspectos en los que hay que tener mucho cuidado al momento de configurar nuestro servidor es la seguridad del mismo. Es por esto, que subversion permite el uso de diferentes tipos de autenticación.

-Sin autenticación: Se puede configurar Subversion para que acepte acceso anónimo. Sin embargo, esto no es recomendable. Para permitir el acceso a cualquier persona, se tiene que modificar el archivo svnserve.conf estableciendo la variable anon-access en read o write.
-Usuarios: Consiste en el uso de usuarios, los mismos que requerirán de credenciales para interactuar con el repositorio y que tendrán que estar definidos en un archivo externo con el formato usuario=contraseña. La ruta de este archivo a su vez tendrá que indicarse en el archivo svnserve.conf en la variable password-db.
-SSH: El uso de SSH es la forma más segura en la que pueden interactuar los usuarios con el repositorio. Para hacer uso de este método, tenemos que conectarnos al repositorio usando la ruta svn+ssh://url/del/repositorio.

7.6.2 PMD [33]

PMD escanea código desarrollado en Java y busca ciertos problemas potenciales como:

- Posibles bugs: Como ciertos bloques vacíos (try/catch/finally/switch)
- Código muerto: Variables locales, parámetros o métodos privados que no se utilizan
- Código no óptimo: Mal uso de String/StringBuffer
- Expresiones muy complicadas: Condicionales innecesarios o bucles de tipo “for” que podrían ser de tipo “while”.
- Código duplicado: Si copiamos código podemos copiar también errores.

PMD se puede integrar con varios IDEs tales como Jdeveloper, Eclipse, Jedit, Jbuilder, Netbeans, Maven, Emacs, entre otros y actualmente se encuentra en la version 5.0.

PMD incluye por defecto una serie de reglas aunque cada usuario puede agregar/crear las reglas que crea conveniente. A continuación se detallan algunas de las reglas más comunes y útiles para cualquier desarrollador.

- ForLoopShouldBeWhileLoop: Algunos bucles for pueden simplificarse a bucles while. Esto hace que sean más concisos y que se ejecuten más rápido.
- ReturnFromFinallyBlock: Hay que evitar que se retornen valores desde un bloque “finally” porque se pueden perder excepciones.
- UnconditionalIfStatement: No es necesario usar condicionales cuaándo el resultado es siempre verdadero o falso.
- CollapsibleIfStatements: A veces es posible consolidar 2 condicionales en 1 solo.
- AvoidDecimalLiteralsInBigDecimalConstructor: Al momento de instanciar objetos de tipo BigDecimal es importante hacerlo usando una cadena (new BigDecimal(“0.1”) en vez de new BigDecimal(0.1)) porque por ejemplo 0.1 no puede representarse como un objeto de tipo doble.
- BigIntegerInstantiation: No es necesario instanciar objetos de tipo BigInteger que ya existen. Usar BigInteger.ZERO en vez de new BigInteger(0).
- SimplifyBooleanExpressions: Muchas veces es posible simplificar expresiones en las que están involucrados booleanos.
- SwitchStmtsShouldHaveDefault: Todo bloque de código switch debe tener la opción default.
- AvoidDeeplyNestedIfStmts: Nunca es bueno crear muchos condicionales anidados.
- EqualsNull: Cuando comparamos alguna variable con null, no deberíamos usar el método equals sino el operador ==. AbstractClassWithoutAbstractMethod: Una clase abstracta siempre debería tener métodos abstractos.
- CompareObjectsWithEquals: Dos objetos no se deberían comparar haciendo uso de ==.

- PositionLiteralsFirstInComparisons: Es recomendable que los literales se coloquen primero en comparaciones. Esto permite que se eviten excepciones por objetos nulos.
- PreserveStackTrace: Si lanzamos una nueva excepción sin pasarle a esta el Stacktrace, mucha información se va a perder. No es recomendable hacer esto.
- EmptyCatchBlock: Un bloque catch no debería estar vacío.
- EmptyIfStmt: Si un condicional esta vacío no debería ser necesario utilizarlo.
- EmptyWhileStmt: Un bucle while nunca debería estar vacío. Si se desea crear un loop para pausar la ejecución, se debería utilizar Thread.sleep().
- EmptyWhileStmt: No es necesario utilizar bloques try sin código.
- EmptyFinallyBlock: Los bloques finally sin código no tienen sentido.
- EmptySwitchStatements: Todos los bloques switch que se encuentren vacíos deberían eliminarse.
- AvoidDuplicateLiterals: No deberían repetirse más de 4 cadenas en una clase. Es mejor hacer uso de constantes.
- StringInstantiation: Los objetos de tipo String no debería instanciarse.
- StringToString: Es innecesario llamar al método toString() si estamos trabajando con cadenas.
- UselessStringValueOf: A veces no es necesario utilizar el método valueOf()
- UseEqualsToCompareStrings: Usar equal() en vez de == para comparar cadenas.
- NpathComplexity: La complejidad del método es muy grande. Esto se mide de acuerdo a la cantidad de caminos a cíclicos que puede haber en el método. Este error se muestra cuándo este valor supera 200.
- ExcessiveParameterList: Es complicado dar mantenimiento a métodos que reciben demasiados parámetros. En estos casos es mejor usar nuevos objetos.
- CyclomaticComplexity: Este tipo de complejidad se determina de acuerdo a la cantidad de puntos de decisión (if, for, while y case). Este valor no debería ser mayor a 8.
- UnusedPrivateField: Indica que una variable privada no se está utilizando.
- UnusedLocalVariable: Indica que una variable local no se está utilizando.
- ShortVariable: Los nombres de las variables no deberían ser muy cortos para poder saber para qué sirven.
- LongVariable: Las variables muy largas hacen que sea difícil seguir el código de una aplicación.

- VariableNamingConventions: Una variable final debería estar escrita en mayúsculas. Las variables no finales no debería usar guiones bajos en su nombre.
- MethodNamingConventions: Los métodos siempre deben empezar con una minúscula y no deben tener guiones bajos.
- ClassNamingConventions: Los nombres de las clases siempre deberían empezar con una mayúscula.
- AvoidDollarSigns: Hay que evitar hacer uso de $ en los nombres de interfaces, clases, métodos o variables.

Capítulo 8: Características de la aplicación

8.1 Módulo de notificaciones

El módulo de notificaciones es la interfaz que permite que se pueda realizar la comunicación entre la aplicación Web y la base de Datos. Es decir, que es un Objeto de acceso a Datos, también conocido como DAO.

Este módulo también se comunica con el módulo de Comunicación permitiendo así la comunicación con el sistema bancario. Para lograr esto, dicho módulo hace uso de varios paquetes, los mismos que se describen a continuación.

8.1.1 Persistences

En este paquete se encuentran las clases que representan a cada una de las tablas de la base de datos. En cada una de estas clases están declaradas variables para cada uno de los campos de la BD, en las que se indica no sólo el tipo de dato, sino también si es o no parte de la llave principal (para ello se hace uso de Anotaciones). En estas clases también se encuentran métodos que permiten obtener o establecer los valores de cada uno de los campos.

Helper:

En este paquete se encuentran clases que permiten obtener directamente la información de la base de datos. Los métodos de cada una de estas clases por lo general devuelven objetos que representan registros completos de la BD. La mayoría de las clases de este paquete son estáticas y hacen uso de lenguaje HQL para comunicarse con la base de datos. A continuación se listan cada una de las clases y se describe brevemente que hace cada una de ellas.

- AuthorizationHelper: Esta clase permite obtener un listado con las autorizaciones pendientes de procesar que tiene un usuario. También posee métodos para actualizar el estado de una autorización, para caducarla e incluso para guardar u obtener el archivo XML de la autorización.
- DeviceHelper: Con esta clase podemos obtener información sobre los dispositivos que tiene un usuario. Entre otras cosas podemos saber el número de dispositivos desde los que se ha conectado, un listado con los dispositivos activos o el último dispositivo desde el que se conectó. También existen métodos que permiten almacenar o caducar un dispositivo en específico.
- LoginHelper: Se usa para obtener un registro de la tabla TUSUARIOSPASSWORD o para caducar la contraseña de un usuario.
- ParameterHelper: Permite obtener y establecer los parámetros de la aplicación que se encuentran parametrizados en la BD.
- SessionHelper: Clase encargada del manejo de sesiones. Permite guardar y caducar sesiones e incluso verificar si un usuario esta en sesión o no.
- TokenHelper: Permite obtener un registro completo del token. Se puede buscar por el id del token o por el id de la autorización a la que esté asignada.
- UserHelper: Esta clase se puede usar para obtener usuarios pero también permite verificar si la contraseña de un usuario es incorrecta u obtener la información adicional de un usuario en especial.

8.1.2 Herramientas

En este paquete se encuentran clases extras que permiten algunas de las funcionalidades de la aplicación. Algunas de las cosas que se pueden hacer con este paquete son:

- Obtener la fecha actual de sistema y la fecha de expiración de un registro.
- Encriptar contraseñas.
- Crear excepciones con mensajes claros para que el usuario entienda a que se debe un error.
- Manejar los flujos de información que se dan entre los distintos módulos de la aplicación.
- Crear tokens únicos para las autorizaciones.
- Leer y escribir archivos XML y convertirlos en objetos para manejarlos de una manera más sencilla.
- Manejar de una forma sencilla los archivos de propiedades de la aplicación.

En el paquete de Herramientas también tenemos clases que permiten representar el XML mediante el cual se puede transmitir información entre los distintos módulos. Las clases usadas para esto son:

- Request: Es una clase genérica. Contiene la cabecera del XML.
- Notification: Se extiende de un Request. Representa un archivo XML completo y contiene cualquier cantidad de Pages.
- Page: Representa una tabla. Cada Page puede tener varios Registers.
- Register: Es un registro de una tabla. Contiene Columns.
- Column: Es el elemento más pequeño. Permite almacenar el nombre de una columna y el valor de la misma.

8.1.3 MailSender

Se encarga de armar el mensaje HTML que se enviará al oficial para notificarle que ha recibido una autorización. Este paquete también se preocupa por enviar el mensaje al correo electrónico que tenga registrado el oficial en la base de datos.

8.1.4 Processors

Aquí se encuentran los procesos que se van a ejecutar dependiendo del mensaje que reciba este servidor desde el servidor Web. A continuación se encuentra un listado de cada uno de los procesos y una explicación de lo que hace cada uno de ellos:

- AuthorizeProcessor: Se encarga de procesar una autorización. Este proceso actualiza el estado de la autorización en la base de datos de la aplicación y actualiza el Detail que se va a devolver a FIT. Antes de realizar el proceso, se verifica que el token de la autorización esté vigente
- CloseSessionProcessor: Cierra la sesión actual de un usuario. Se ejecuta cuando el usuario cierra la aplicación.
- CloseUserSessionProcessor: Este proceso es usado por el Administrador para cerrar la sesión de un usuario en particular.
- DeleteNotificationProcessor: Este proceso es usado por el Administrador para eliminar una autorización que no se ha logrado procesar correctamente.
- ExpireDeviceProcessor: Permite eliminar un dispositivo de la base de datos cuando el usuario lo solicita. Este proceso se tiene que ejecutar siempre cuando el usuario ha excedido el número de dispositivos activos permitidos.
- FillUsersTableProcessor: Es utilizado para migrar los usuarios de FIT en la base de datos de la aplicación. Este proceso llena las tablas TUSUARIOS, TUSUARIOPASSWORD (guarda una contraseña por defecto) y TUSUARIO INFORMACIONADICIONAL (se llena sólo si el usuario tiene dicha información en la base de datos de FIT).
- ListDevicesProcessor. Devuelve un listado con los dispositivos activos de un usuario.
- ListNotificationsProcessor: Devuelve las autorizaciones que tiene pendiente un usuario en particular. Si el usuario conectado es el Administrador, devuelve todas las autorizaciones que no se pudieron procesar adecuadamente.

- LoginProcessor: Es el proceso que se ejecuta cuando un usuario inicia sesión. Aquí se verifica que la contraseña sea correcta y que esté vigente. También se valida que el usuario esté vigente y si es la primera vez que se inicia sesión le solicita al usuario que actualice su contraseña.
- NotifyProcessor: Es el proceso que se encarga de enviar la notificación (correo electrónico) al oficial.
- RegisterDeviceProcessor: Permite registrar un dispositivo en la base de datos. Se ejecuta después de que el usuario inicia sesión.
- ResetPasswordProcessor: Se usa para reiniciar la contraseña del usuario.
- ReturnParameterProcessor: Devuelve el valor de un parámetro almacenado en la base de datos.
- ReturnParametersProcessor: Devuelve todos los parámetros del sistema.
- ReturnRoleProcessor: Devuelve el rol que tiene un usuario en particular.
- UpdateParametersProcessor: Es utilizado por el administrador cuando este quiere actualizar el valor de algún parámetro del sistema.
- UpdatePasswordProcessor: Permite actualizar la información del usuario. Se puede modificar la contraseña, el nombre y el correo electrónico.
- UpdateTokenProcessor: “Revive” un token cuando este ha caducado.

8.1.5 NotificationServer

Es el paquete en el que se encuentra el servidor en sí. El servidor funciona de la siguiente manera:

- Apenas se deploya la aplicación el servidor elimina cualquier sesión que haya quedado activa.
- Luego, este se pone en modo de escucha. El servidor puede recibir mensajes tanto del Módulo de Comunicación como de la aplicación Web.
- Cuándo recibe un mensaje, se inicia una transacción de hibernate.
- Dependiendo del tipo de mensaje, el servidor elige qué procesador ejecutar. En el caso de que el mensaje se utilice para aprobar o negar una autorización se envía un mensaje al Módulo de Comunicación.
- Después de procesar el mensaje se devuelve el resultado (esto ocurre sólo cuándo el mensaje fue enviado por la aplicación Web).
- Luego se cierra la conexión con el cliente.

- Finalmente, si el proceso se ejecutó sin errores se hace commit de toda la información. Caso contrario se hace un rollback.

8.2 Módulo de comunicación Dispositivo - Aplicación

Este módulo permitirá para la comunicación entre los usuarios y el núcleo bancario que utilizan, teniendo como puente al servidor de Notificaciones. Su diseño está basado en la estrategia MVC o Modelo – Vista – Controlador, donde el módulo en cuestión actuará como el controlador de procesos para las acciones de usuario, y a su vez de contenedor de la interfaz gráfica.

8.2.1 Definición de usuarios del sistema

El sistema contará básicamente con dos roles de usuario para el acceso:

- Oficiales: podrán realizar acciones de autorización/negación sobre las operaciones asignados a estos desde el núcleo bancario, como también realizar el mantenimiento de sus datos en la aplicación.
  - Privilegios: acceso a los datos propios de la aplicación como también a l información de autorizaciones asignadas.
- Administrador: podrá realizar el mantenimiento de datos de configuración y comunicación de la aplicación con el núcleo bancario, datos y estado de los usuarios, y la realización de acciones sobre las autorizaciones pendientes de los usuarios.
  - Privilegios: acceso a toda la información alojada en la aplicación.

8.2.2 Requerimientos funcionales

- Inicio de Sesión de Usuario
  - Id: sign
  - Descripción: realiza el proceso de petición de inicio de sesión de un usuario, el manejo de dispositivos y la obtención de información de Usuario en caso de obtener un proceso exitoso desde el servidor de notificaciones.
  - Prioridad: Alta
- Cambio de Información de Usuario
  - Id: chpwd
  - Descripción: realiza la actualización de la información de Usuario (Nombres, Correo Electrónico de envío de notificaciones y Contraseña).
  - Prioridad: Media
- Listado de Dispositivos de Usuario

    - Id: ldev
  - Descripción: obtiene la lista de dispositivos de usuario registrados al momento de iniciar sesión.
  - Prioridad: Media
- Actualización de Dispositivos Registrados de Usuario
  - Id: upd
  - Descripción: Actualiza el estado de los dispositivos de Usuario
  - Prioridad: Media
- Listado de Autorizaciones Pendientes
  - Id: notif
  - Descripción: Obtiene el listado de información de las autorizaciones de un Oficial pendientes por autorizar/negar.
  - Prioridad: Alta
- Cierre de Sesión de Usuario
  - Id: lout
  - Descripción: Realiza la petición de cierre de sesión de usuario al módulo de notificaciones. Al obtener un proceso exitoso eliminará la sesión HTTP del usuario en cuestión.
  - Prioridad: Alta
- Petición de Autorización
  - Id: authr
  - Descripción: Realiza la petición de un token de vigencia para una autorización en el caso de que la asignada a la misma ya se encuentre caducada.
  - Prioridad: Alta
- Petición de Detalle de Autorización
  - Id: authd
  - Descripción: Obtiene la información de una autorización específica y finaliza con la obtención de la interfaz designada para enviarla al usuario.
  - Prioridad: Alta
- Autorización/Negación de una Autorización
  - Id: auth
  - Descripción: Envía al servidor de Notificaciones la decisión del oficial sobre una autorización en cuestión.
  - Prioridad: Alta
- Obtención de la Información de Usuario
  - Id: usinf
  - Descripción: Envía a la interfaz la información del usuario conectado.
  - Prioridad: Media
- Obtención de los Parámetros del Servidor de Notificaciones
  - Id: servd
  - Descripción: Obtiene los parámetros configurables del módulo de

Notificaciones
  - Prioridad: Media
- Actualización de los Parámetros del Servidor de Notificaciones
  - Id: upsrd
  - Descripción: Actualiza los parámetros del servidor de Notificaciones enviados desde el usuario Administrador.
  - Prioridad: Media
- Reinicialización de Contraseña del Oficial
  - Id: rupas
  - Descripción: Realiza la petición de reinicio de la contraseña de un Oficial desde el usuario Administrador al módulo de Notificaciones.
  - Prioridad: Media
- Finalización de una Sesión de un Oficial
  - Id: fussn
  - Descripción: Realiza la petición de finalización de la sesión de un Oficial desde el usuario Administrador al módulo de Notificaciones.
  - Prioridad: Media
- Eliminación de Autorización “Muerta”
  - Id: dauth
  - Descripción: Realiza una petición de eliminación de una autorización con código de Error.
  - Prioridad: Alta

8.2.3 Requerimientos no funcionales

- Manejo de Dispositivos
  - Id: ninguno
  - Descripción: se debe tener un administrador de los Dispositivos que se conectan a la aplicación, pudiendo filtrar aquellos que sean permitidos y/o compatibles en base a una parametrización.
  - Prioridad: Alta
- Manejo de Imágenes de Interfaz de Usuario
  - Id: ninguno
  - Descripción: los logos de la aplicación deberán ser estandarizados para poder ser intercambiados fácilmente con los de la entidad bancaria que

adquiera el sistema.
    - Prioridad: Baja
- Manejo de Errores
    - Id: error
    - Descripción: realiza el manejo y envio al usuario de errores obtenidos en el módulo en cuestión como también de los enviados por el módulo de notificaciones.
    - Prioridad: Alta.
- Número de Dispositivos Registrados
    - Id: aedev
    - Descripción: Verifica el número de dispositivos ya registrados en el servidor para fines de redirección a nivel de la interfaz de usuario.
    - Prioridad: Alta
- Verificación de Rol de Usuario
    - Id: uspri
    - Descripción: verifica el rol que tiene el usuario para fines de redirección a una correcta interfaz de usuario.
    - Prioridad: Media
- Obtención de un Parámetro del Servidor
    - Id: gsysp
    - Descripción: Realiza la petición de un valor de un parámetro del módulo de Notificaciones, para la toma de decisiones en la funcionalidad del controlador.
    - Prioridad: Alta

8.3 Módulo de comunicación aplicación-núcleo

El módulo de comunicación es el que permite que se pueda comunicar el UCI de FIT con el módulo de notificaciones de la aplicación. Este es el único módulo que se tendría que modificar en el caso de que se requiera instalar la aplicación en otro sistema bancario. Este módulo esta compuesto únicamente por tres paquetes:

DTO

Es un paquete que contiene clases de FIT para manejar un Detail.

BankClient

Es el paquete en el que se indica cómo se tiene que conectar la aplicación al UCI de FIT. Aquí tenemos que indicar en un archivo de propiedades tanto la IP como el puerto al que nos queremos conectar. Este paquete se encarga de realizar la conexión y de enviarle mensajes al UCI.

FitComunication

Este paquete posee únicamente dos clases:

- DetailNotificationAdapter: Es una clase que usa el patrón de diseño Adapter. Esta recibe cómo parámetro un archivo XML y lo transforma. Si se recibe un Detail (archivo de FIT) este adaptador se encarga de transformarlo en un Notification. Por el otro lado, cuando se recibe un Notification, la clase permite devolver un Detail.
- FitReceiverServer: Este es el servidor se encuentra en escucha todo el tiempo al igual que el módulo de Notificaciones pues se deploya al mismo tiempo. El comportamiento del servidor varía dependiendo si el mensaje que recibe es un Detail o un Notification.

Si se recibe un Detail, el servidor funciona así:

- Primero convierte el Detail en Notification.
- Si es un Detail de autorización (Es usado en FIT para aprobar/negar una autorización), le envía al servidor de Notificaciones un mensaje indicándole que tiene que procesar la autorización en la base de datos.
- Si es un Detail de migración de usuario, le envía al servidor de notificaciones un mensaje indicándole que tiene que almacenar dichos usuarios en la base de datos.
- Caso contrario se tratará de una autorización que se acaba de generar. En este caso le indica al servidor de notificaciones que tiene que almacenar la autorización en la base de datos y notificar al oficial indicado.

En el caso de ser un Notification, el servidor sabe que se trata de un proceso de autorización que se tiene que enviar al UCI, por eso hace lo siguiente:

- Saca del Notification un Detail. El mismo contiene la respuesta que el oficial ha dado a la autorización.
- Agrega en el Detail información importante para que el UCI pueda procesar la transacción. Se guarda el número de identificación de la autorización, la transacción que se tiene que ejecutar, la respuesta del oficial y la observación que el mismo allá indicado.
- Luego se envía el Detail al UCI y se espera una respuesta.
- Finalmente se envía la respuesta al módulo de notificaciones para que este sepa si se pudo o no procesar la autorización.

8.4 Interfaz Gráfica

La interfaz gráfica deberá ser diseñada para dispositivos móviles, y en específico para terminales Blackberry. Las pantallas deberán presentar una información resumida y legible para el Oficial bancario, y la toma de decisiones deberá ser sencilla sin dar cabida a posibles errores por parte del usuario.

Las interfaces necesarias a ser construidas son las siguientes:

- Interfaz de Autenticación de Usuario e Información de la Entidad Bancaria
  - Descripción: se deberá tener una interfaz para la autenticación de los usuarios donde se tendrán componentes comunes para el resto de interfaces y fácilmente intercambiables con los logos o imágenes de la entidad bancaria.
  - Requerimientos asociados
    - Manejo de Imágenes de Interfaz de Usuario
    - Manejo de Errores
    - Número de Dispositivos Registrados
    - Verificación de Rol de Usuario
    - Obtención de un Parámetro del Servidor

- Interfaz de Oficial Bancario
  - Descripción: la interfaz podrá mostrar el listado de autorizaciones pendientes del usuario, el detalle de una autorización escogida (que puede consistir en varias páginas dinámicamente armadas), como también una interfaz sencilla para el envío de la decisión tomada. Además se debe proporcionar interfaces para el mantenimiento de los dispositivos registrados y la información de usuario (Nombre, Contraseña y Correo electrónico para el envío de notificaciones).

- Requerimientos asociados
  - Inicio de Sesión de Usuario
  - Cambio de Información de Usuario
  - Listado de Dispositivos de Usuario
  - Actualización de Dispositivos Registrados de Usuario
  - Listado de Autorizaciones Pendientes
  - Cierre de Sesión de Usuario
  - Petición de Autorización
  - Petición de Detalle de Autorización
  - Autorización/Negación de una Autorización
  - Obtención de la Información de Usuario
  - Manejo de Dispositivos

- Interfaz de Administrador
  - Descripción: la interfaz debe brindar el fácil mantenimiento de la información del sistema de Notificaciones, pudiendo ser accedido desde una terminal de escritorio o un dispositivo móvil.
  - Requerimientos asociados
    - Obtención de los Parámetros del Servidor de Notificaciones
    - Actualización de los Parámetros del Servidor de Notificaciones
    - Reinicialización de Contraseña del Oficial
    - Finalización de una Sesión de un Oficial
    - Eliminación de Autorización “Muerta”

8.4.1 Elección del enfoque de desarrollo

Según un análisis elaborado en octubre del 2011 por comScore [34], sobre el impacto que tienen los distintos sistemas operativos de dispositivos móviles en el mercado latinoamericano, y basándose en el tráfico de Internet, se obtuvo la Figura 8-1 sobre la cantidad de los dispositivos más utilizados:

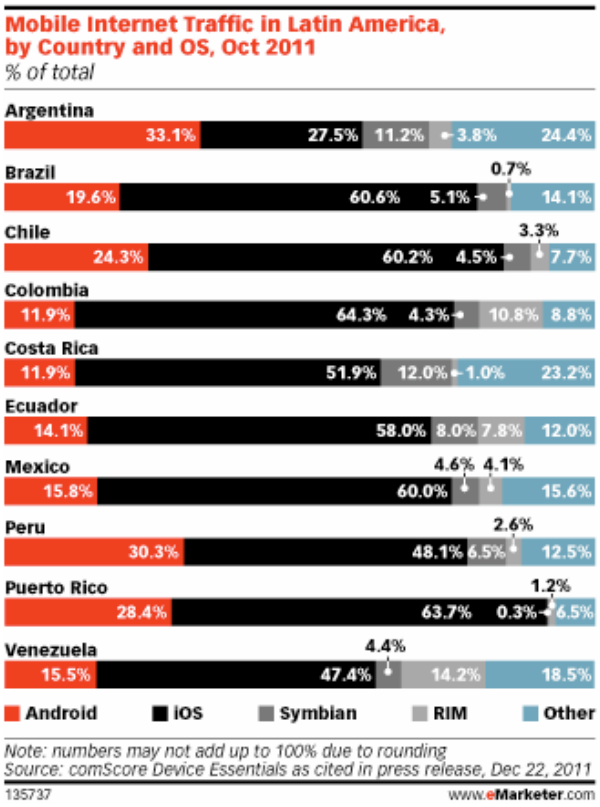


Figura 8-1: Dispositivos más utilizados en Latinoamérica

Se puede visualizar que los dispositivos con el sistema iOS dominan el sector, con Android como rival, pero algo lejano. Por lo que además de realizar un análisis de las distintastecnologías y enfoques de desarrollo para móviles, se ha escogido desarrollar una aplicación híbrida o Widget, donde se utilizará estándares HTML5, CSS3 y JavaScript (librerías jQuery). Esta podrá ser adaptada y utilizada en las distintas plataformas mediante la utilización de Apache Córdoba y su servicio sobre la nube PhoneGap Build. En definitiva la aplicación móvil se desarrollará en un inicio para dispositivos Blackberry de manera nativa, e iOS y Android en disponibilidad mediante el navegador Web, pero que podrán ser fácilmente migradas a un entorno nativo para un futuro.

Capítulo 9: Diseño de la interfaz gráfica para móviles

La interfaz gráfica de la aplicación será desarrollada para dispositivos móviles

Blackberry, Android y basados en iOS, cuyos tamaños de pantalla varían entre 480x320 pixeles (Blackberry), hasta los 960x640 pixeles (iOS). Debido que para el desarrollo de la interfaz gráfica se utilizará la librería jQueryMobile v1.0.1, donde se utilizan estándares HTML5 y CSS3, y que en conjunto con reglas CSS propias, entre ellas el uso de Media Queries para la manipulación del viewport, esta será acoplable a cualquier tamaño de pantalla de un móvil, e incluso a la de una terminal de escritorio. La navegación de la interfaz será mediante un modelo multipágina y de forma vertical, por lo que el desplazamiento será únicamente en esa dirección.

9.1 Diseño estructural de las páginas

Cada página de la interfaz contendrá los siguientes elementos en común:

- Encabezado: contendrá el nombre de la aplicación y un logo que podrá ser intercambiado con el correspondiente a la entidad bancaria. Dicho logo deberá tener un tamaño de 40x40 pixeles. Además contendrá botones de navegabilidad hacia atrás, actualización y de cierre de sesión, según sea el caso.
- Menú: las páginas que pueden ser accedidas después de un inicio de sesión exitoso contendrán un menú inferior para el acceso a las páginas de autorización u opciones. Además, las páginas que contienen subpáginas tendrás un menú de navegación en su parte superior para poder realizar transiciones entre ellas.

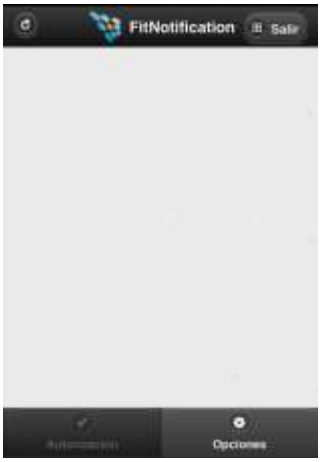


Figura 9-1: Menú de navegación

- Zona de Contenido: el texto de contenido se adapta al tamaño de la pantalla del dispositivo mediante el uso de media queries dentro de la hoja de estilo.

Figura 9-2: Pantalla de contenido

- Componentes: se hará uso de una ventana Pop-up para peticiones de token de autorización, y a su vez se utilizará una ventana de diálogo para notificar de la ejecución exitosa o no de ciertos procesos.

Figura 9-3: Pop-up para renovación de token

- Zona de mensajes

    - Panel de mensajes: Ciertos procesos lanzarán una ventana de diálogo si el servidor devuelve un error ocurrido en alguno de los módulos de la aplicación.

Figura 9-4: Mensaje de error

- Estilo de campos: Los campos ingresables que no pasen una validación predeterminada, serán rodeados por un borde rojo.
- Barra de estado: Las páginas que contengan campos ingresables contendrán una barra de estado inferior, que informará de errores de ingreso por parte del Usuario, o en ciertos casos de errores ocurridos en el servidor.

Figura 9-5: Mensaje de error al iniciar sesión

- Validaciones
  - Validación de Campos de Verificación de Información
    - Tipo: Texto
    - Longitud: mayor a cero
    - Obligatoriedad: Si
    - Caracteres especiales: No

- Valores máximos/mínimos: el campo en cuestión y el de verificación deben tener el mismo valor.

Figura 9-6: Mensaje de error por ingresar un correo incorrecto

- Validación de Campos Obligatorios
  - Tipo: Texto
  - Longitud: mayor a cero
  - Obligatoriedad: Si
  - Caracteres especiales: No
  - Valores máximos/mínimos: el campo no debe ir vacío

Figura 9-7: Mensaje de error de un campo requerido

9.2 Páginas de la interfaz gráfica

- Inicio Aplicación: esta página contiene el logotipo que identifica a la entidad bancaria, y donde se podrá colocar opcionalmente alguna información adicional sobre de ella. Incluye un botón para el acceso a la página de Inicio de Sesión. La imagen del logotipo deberá tener un tamaño de 480x320 pixeles.

Figura 9-8: Splash de la aplicación

Figura 9-9: Opción que permite agregar la aplicación al dispositivo en iPhone

Figura 9-10: Pantalla inicial

- Inicio de Sesión: está conformado por un contenedor con los campos de Nombre de Usuario y Contraseña que el actor debe ingresar.

Figura 9-11: Pantalla de inicio de sesión

- Entorno: contendrá la información de las autorizaciones pendientes o de las páginas del detalle de una en particular.

Figura 9-12: Pantalla donde se muestran las autorizaciones

- Opciones
  - Dispositivos: contiene la información de los dispositivos registrados del Usuario incluida la Fecha de último acceso. Aquí también se podrá eliminar los dispositivos que se encuentran habilitados.

Figura 9-13: Pantalla de dispositivos activos

  - Información de Usuario: contiene los campos con la información de Usuario actual y donde se permitirá a su vez la actualización de los mismos.

- Páginas de Detalle de Autorización: es un contenedor con todas las páginas del detalle de la autorización que han sido obtenidas del núcleo bancario y generadas visualmente desde el servidor.

Figura 9-14: Detalle de una autorización

- Autorización: contiene los campos necesarios para la toma de la decisión sobre la autorización, más un campo de comentarios.

Figura 9-15: Error por no ingresar un comentario al procesar una autorización

- Página de Errores: esta página contendrá algún error lanzado por el servidor y

será mostrada únicamente en casos cruciales de ingreso de Información por parte del Usuario: Inicio de Sesión, Actualización de Información de Usuario y la Decisión sobre una Autorización.

Figura 9-16: Mensaje de error con stacktrace

- Entorno de Administrador: es un contenedor que será mostrado únicamente a usuarios administradores y que contendrá las siguientes páginas:
  - Configuración: contiene los parámetros del sistema de Notificaciones, y es donde pueden ser modificados.

Figura 9-17: Pantalla de configuración del Administrador

- Sesiones: lista los usuarios conectados actualmente, pudiendo realizar una búsqueda y proceder a realizar la finalización de la sesión de uno de ellos.

Figura 9-18: Pantalla para que el administrador elimine sesiones activas

- Contraseña: permite el ingreso de un Nombre de Usuario para realizar la

petición de reinicio de Sesión.

Figura 9-19: Pantalla para que el Administrador reinicie información de un usuario

- Autorizaciones: lista las autorizaciones de los Oficiales que obtuvieron algún código de error al momento de que un Oficial envió una respuesta al núcleo bancario, para que el usuario Administrador tenga la oportunidad de poder eliminar ciertas autorizaciones consideradas “muertas”.

Figura 9-20: Pantalla para eliminar autorizaciones que no se pudieron procesar

La tabla 9-1 muestra la relación entre las páginas de la interfaz gráfica y las funcionalidades requeridas para el sistema:

Requerimiento	**Pant. 1**	**Pant. 2**	**Pant. 3**	**Pant, 4**	**Pant. 5**	**Pant. 6**	**Pant. 7**	**Pant. 8**
Manejo de Imágenes de Interfaz de Usuario	X	X	X	X	X	X	X	X
Manejo de Errores							X	
Verificación de Rol de Usuario								X
Inicio de Sesión de Usuario		X						
Cambio de Información de Usuario				X				
Listado de Dispositivos de Usuario				X				

Actualización de Dispositivos Registrados de Usuario				X				
Listado de Autorizaciones Pendientes			X					
Cierre de Sesión de Usuario			X	X				X
Petición de Autorización			X					
Petición de Detalle de Autorización			X		X			
Autorización/Negación de una Autorización						X		
Obtención de la Información de Usuario				X				
Manejo de Dispositivos				X				
Obtención de los Parámetros del Servidor de Notificaciones								X
Actualización de los Parámetros del Servidor de Notificaciones								X
Reinicialización de Contraseña del Oficial								X
Finalización de una Sesión de un Oficial								X
Eliminación de Autorización“Muerta”								X

Tabla 9-1: Relación entre las pantallas de la aplicación

9.3 Armado de las Páginas de detalle de una autorización

Una autorización que proviene del núcleo bancario debe enviar un flujo XML con

páginas del detalle de una autorización con el siguiente formato:

```
<PAGE id="E-GENERAL">
	<REG id="0">
		<COL name="ASUNTO">AUTORIZACION DE PROTESTO DE CHEQUE</COL>
		<COL name="USUARIO">OSCAR</COL>
		<COL name="NOMBRE_USUARIO">LOPEZ SALCEDO JUAN PEDRO</COL>
		<COL name="SUCURSAL">MATRIZ CUENCA</COL>
		<COL name="OFICINA">CASA MATRIZ</COL>
		<COL name="AREA">AUDITORIA</COL>
		<COL name="TRANSACCION ">04-6028 PROTESTO DE CHEQUES</COL>
		<COL name="DATA">CODIGO CLIENTE|1176791-!-TIPO IDENTIFICACION|CED-!-IDENTIFICACION|0602760811-!-NOMBRE|FIALLOS CALLE GERMAN  DARIO-!-CUENTA|40000000669-!-MONTO|100000.00-!-HISTORIAL|DAVID ALMACENES JUAN ELJURI-!-</COL>
	</REG>
</PAGE>
```

Es obligatorio enviar como mínimo una página con el nombre “E-GENERAL” con la información consolidada de la autorización, aunque el núcleo bancario puede enviar cualquier número de páginas con el detalle de una autorización. Debido a que la aplicación se ejecuta sobre un dispositivo móvil, se recomienda enviar un máximo de 4 páginas. Cada página con el detalle de una autorización tendrá uno de los siguientes formatos:

- Columna: Contendrá un solo registro por cada campo de descripción y estos serán presentados de forma vertical.
- Tabla: Contendrá más de un registro por cada campo de descripción y será presentada de forma tabular.

9.4 Diagrama de navegación

La figura 9-21 muestra el flujo de navegación de las páginas para el usuario Oficial:

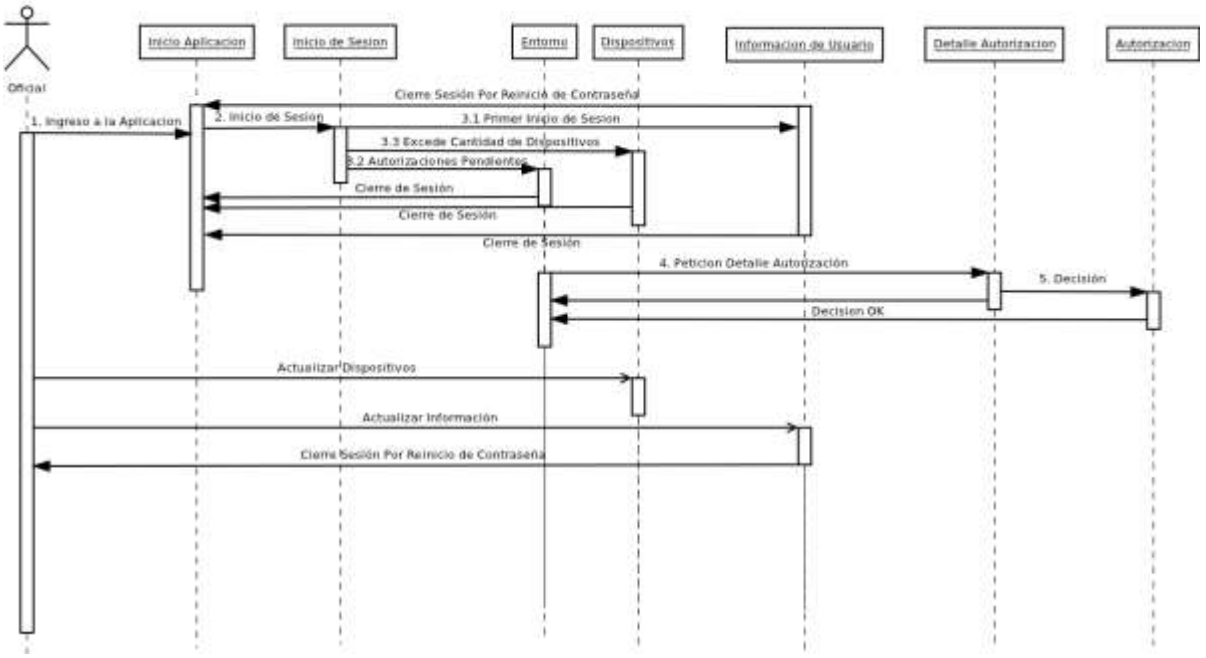


Figura 9-21: Flujo de navegación de un Oficial

La figura 9-22 muestra el flujo de navegación de las páginas para el usuario Administrador:

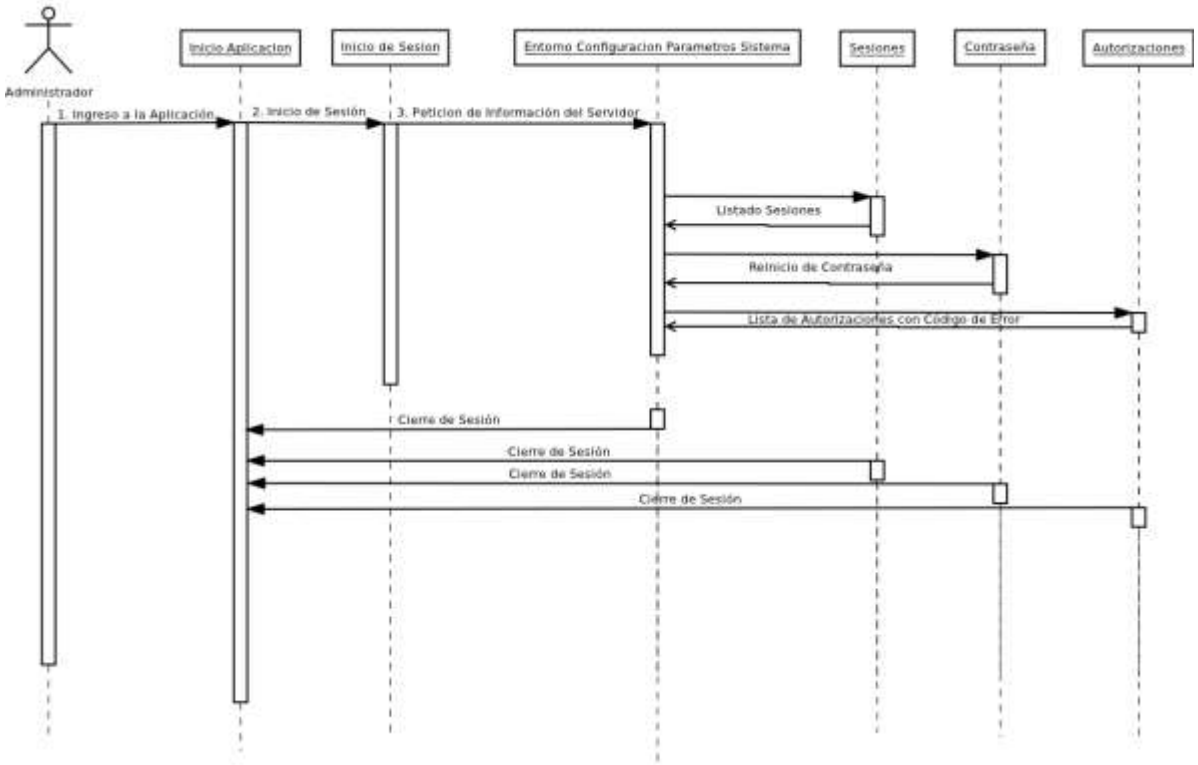


Figura 9-22: Flujo de navegación del Administrador

Capítulo 10: Implementación y Pruebas del Software.

10.1 Funcionamiento actual de las Autorizaciones en Fit-Bank

El CORE y jBPM

El CORE transaccional de Fit-Bank posee actualmente un módulo de autorizaciones que hace uso de la API jBPM para el direccionamiento de autorizaciones a los usuarios adecuados que mediante flujos BPM. Cada tipo de autorización tiene asociado un flujo y unas reglas definidas por la entidad bancaria.

jBPM es un motor de flujos de código abierto escrito en Java y que puede ejecutar procesos de negocio definidos en leguaje jPDL (o en BPMN 2.0 en su versión 5.0). En esencia jBPM toma las descripciones gráficas de los procesos como entrada, donde estos están compuestos de tareas interconectadas con flujos de secuencia.

Los procesos representan la ejecución de un flujo., mientras que el diagrama gráfico de un proceso es utilizado como la base de comunicación entre los usuarios no técnicos y los desarrolladores.

Un proceso de escucha jBPM es levantado al momento de inicializar el servicio del CORE transaccional en un hilo independiente, el cual se encarga del manejo de los flujos o instancias que se encuentran en proceso o que van a ser procesados.

Transacciones de Fit-Bank

Los formularios de Fit-Bank están identificados mediante un número de transacción. En cada transacción se pueden agregar comandos adicionales, que se ejecutarán dependiendo de la operación que se realice (Consulta o Mantenimiento). Aquellos son parametrizados a nivel de la base de datos por lo que la funcionalidad de cualquier transacción puede extenderse tanto como se quiera.

Transacción de creación de flujos BPM

Esta transacción es invocada en cualquier formulario que realice una llamada a un flujo BPM. Posee un comando que crea el flujo y almacena en la base de datos del CORE transaccional la información de este, como su estado y el número de instancia.Este último será útil para el envío de una señal de respuesta al flujo para la continuación de la ejecución de su proceso.

Autorizaciones mediante el canal Web

El canal Web habilitado en Fit-Bank posee un conjunto de transacciones para el listado, monitoreo y la aprobación/negación de los flujos que requieran ser procesados por un usuario de la entidad bancaria. Cuándo un usuario procese una autorización desde la aplicación Web, esta automáticamente se procesará en la base de datos de la aplicación móvil, logrando de esta manera que no existan inconsistencias entre las dos aplicaciones.

Consulta de autorizaciones pendientes

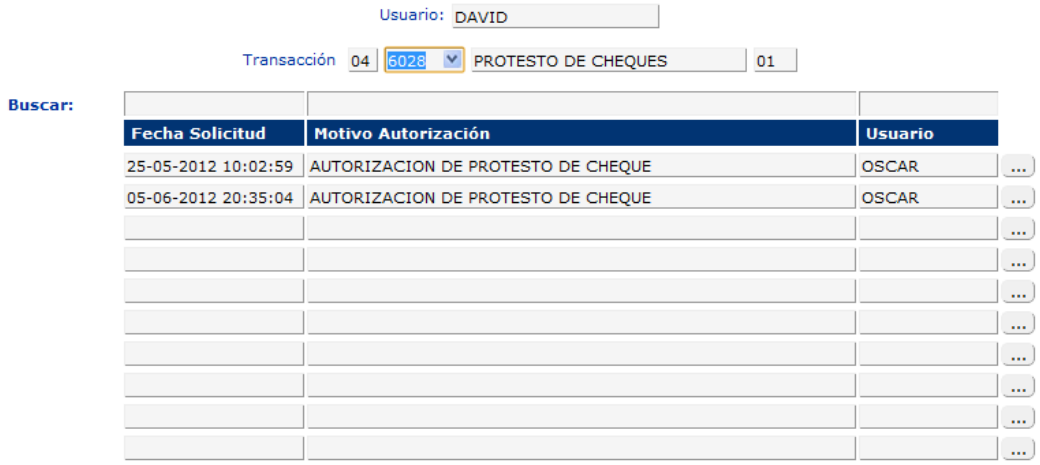


Figura 10-1: Consulta de autorizaciones pendientes en Fit-Bank

Autorización de transacciones

Figura 10-2: Autorización de transacciones en Fit-Bank

Monitoreo de procesos de autorización


MONITOREO DE PROCESOS DE AUTORIZACIÓN

Transacción 04 5021 PROTESTO DE CHEQUES 01

Fecha	Motivo	Usuario	Terminal	Actividad Actual
21-05-2012 00:00:00	PROCESO FINALIZADO		SANTIAGO	Autorizacion Protesto
21-05-2012 00:00:00	PROCESO FINALIZADO		SANTIAGO	Autorizacion Protesto
21-05-2012 00:00:00	PROCESO FINALIZADO		SANTIAGO	Autorizacion Protesto
21-05-2012 00:00:00	PROCESO FINALIZADO		SANTIAGO	Autorizacion Protesto
21-05-2012 00:00:00	PROCESO FINALIZADO		SANTIAGO	Autorizacion Protesto
25-05-2012 00:00:00	AUTORIZACION DE PROTESTO DE CHEQUE	OSCAR	OSCAR	Autorizacion Protesto
05-06-2012 00:00:00	AUTORIZACION DE PROTESTO DE CHEQUE	OSCAR	OSCAR	Autorizacion Protesto
05-06-2012 00:00:00	AUTORIZACION DE PROTESTO DE CHEQUE	OSCAR	OSCAR	Autorizacion Protesto
05-06-2012 00:00:00	AUTORIZACION DE PROTESTO DE CHEQUE	OSCAR	OSCAR	Autorizacion Protesto

Figura 10-3: Monitoreo de procesos de autorización en Fit-Bank

10.2 Parametrización al sistema de Fit-Bank y adaptación del Sistema de Notificaciones

Comunicación con el UCI

Para poder comunicarse con el núcleo bancario de Fit-Bank, fue necesaria la creación de un nuevo canal en el administrador de canales o UCI, en escucha de mensajes XML y en una comunicación de tipo Petición/Respuesta. Mediante la parametrización de este canal, el servicio será creado y automáticamente redireccionará los mensajes XML del sistema de Notificaciones hacia el CORE transaccional en su formato definido para su proceso, y donde finalmente retornará la respuesta del proceso ejecutado.

Ajustes a nivel del CORE

Mediante una parametrización, el CORE transaccional tendrá la capacidad de ejecutar comandos de tipo “N” o Notificación, que estarán asociados a la transacción que realiza una llamada a la creación de un flujo BPM, y que serán capaces de la creación de detalles de una autorización mediante la ADJUNTACION de ellos en el mensaje XML a enviar al servidor de Notificacionesen el formato definido.

Envío de información de Autorización al Servidor de Notificaciones

La transacción encargada de la creación de flujos instancias BPM, direccionará, una vez creado el flujo, la información y detalle de la autorización XML creada hacia el servidor de Notificaciones mediante el uso de un canal de comunicación para que

sea éste el encargado de enviar la información al dispositivo del usuario autorizador. Esto se realizará mediante la parametrización de un comando para el efecto.

Transacción de migración de Oficiales Bancarios

Fit-Bank posee una transacción para la creación de Usuarios en la base de datos. Para mantener homogeneidad entre Fit-Bank y la aplicación de autorizaciones se desarrolló un comando especial que se encarga de enviar dicha información cuando se ejecute un mantenimiento sobre dicha transacción. Dicho comando también se parametrizó en una nueva transacción diseñada para realizar una migración inicial de todos los oficiales. El comando que se ejecuta, se encarga de adjuntar en el mensaje XML los nombres de usuario y su información adicional.

Ingreso y modificación de usuarios


INGRESO Y MODIFICACIÓN DE USUARIOS

Tipo Usuario EMPLEADO COMPAÑIA GRUPO

Buscar:

Idioma(*)	Usuario(*)	Persona(*)	Nombre Legal	Estatus(*)	Alias(*)	Normal	Oficial	Cajero	Bóveda
ES	AAA	1176904		ACT	ANITA	✓			
ES	AA2	1174947	ALMACENES JUAN ELJURI	ACT	AAA	✓			
ES	ADMIN	1	MIGRACION	INA	ADMIN		✓		
ES	ALEX	1	MIGRACION	INA	ALEX			✓	✓
ES	ALEXANDRA	1	MIGRACION	INA	ALEXANDRA		✓		
ES	ANDRES	1	MIGRACION	INA	ANDRES		✓		
ES	BA01000603	1	MIGRACION	ACT	BA01000603		✓		
ES	BA01000633	1	MIGRACION	ACT	BA01000633		✓		
ES	BA01001111	1	MIGRACION	ACT	BA01001111		✓		
ES	BA01002701	1	MIGRACION	ACT	BA01002701		✓		

Figura 10-4: Ingreso y modificación de usuarios en Fit-Bank

Migración inicial de Usuarios

Figura 10-5: Migración inicial de usuario en Fit-Bank

10.3 Manejo de Sesiones [35]

El manejo de sesiones de Usuario se llevará a cabo utilizando las propiedades del objeto HttpSession, el que es obtenido mediante el HttpServletRequest en la petición del Servlet. En la primera llamada al objeto HttpSession, el administrador de sesiones utiliza una de las tres vías para establecer el seguimiento de una sesión: uso de cookies, rescritura de URL o información de SSL. Para nuestro caso se hará el uso de cookies. En este caso, el administrador de sesiones crea una Id de sesión única, la cuál es enviada al navegador como una cookie. A continuación cualquier petición subsecuente del usuario en cuestión sobre el mismo navegador pasará la cookie que contiene la Id de la sesión y el administrador de Sesiones la utilizará para encontrar el objeto HttpSession existente. Además será posible almacenar cierta información de Usuario dentro de la sesión creada.

Como un paso más en el manejo de sesiones adjuntará la Identificación de Sesión de Usuario en el proceso de Inicio de Sesión enviado al canal del servidor de Notificaciones habilitado para su registro en la base de datos, y será este último el que maneje el tiempo de vida de una sesión abierta. Si el usuario envía un proceso de cierre de Sesión y el servidor de Notificaciones devuelve una respuesta exitosa, o si envía una respuesta de sesión caducada al invocar cualquier otro tipo de proceso a petición del Usuario, se hará una llamada al proceso *invalidate()* para eliminar la sesión del Servlet.

10.4 Envío de notificaciones/autorizaciones

Para el envío de notificaciones a los oficiales se hará uso de una API de Java diseñada justamente para esto llamada JavaMail. El uso de esta herramienta es muy simple ya que sólo basta indicar la información del servidor de correo a través del cual se enviarán los correos electrónicos.

Debido a que la instalación de un servidor de correo no era parte esencial del desarrollo de la aplicación, se optó por usar el servidor de correo de Gmail. Para ellos se creóo un correo electrónico a través del cual se enviarán cada una de las notificaciones. Al usar este servidor, además estamos asegurándonos de que la información permanezca segura pues Gmail hace uso de SSL para enviar estos mensajes.

10.5 Ejemplo de funcionamiento con cada autorización

Los ejemplos de funcionamiento del sistema con cada autorización se encuentran en el anexo de la aplicación.

Capítulo 11: Conclusiones

- Mediante el uso de Infraestructuras Digitales móviles, tales como Apache Cordova, es posible realizar aplicaciones híbridas, que sin necesidad de conocer el lenguaje de programación propio de los dispositivos, permiten hacer uso de las mismas funcionalidades y/o capacidades que estos últimos proveen a una

aplicación nativa; Además gracias a nuevas tecnologías, tales como HTML5, CSS3 y WebKit, es posible realizar una aplicación móvil compatible con la gran mayoría de dispositivos existentes en el mercado actual, aislando la interfaz gráfica de la lógica de la aplicación.

- En los últimos años las aplicaciones móviles han tenido gran demanda en distintos campos, tanto de entretenimiento como profesionales, brindando nuevas soluciones no sólo para los clientes sino también para los empleados de una empresa o corporación. Con el desarrollo de esta aplicación, se facilitarán algunos de los procesos de negocio de las entidades financieras, pertiéndo así una mejor y más rápida atención a sus clientes.
- La abstracción en la interfaz gráfica utilizada al momento de desarrollar la aplicación permitirá que esta pueda ser ampliada, dándole nuevas y mejores funcionalidades en un modo más nativo para una plataforma que una entidad financiera requiera en específico.
- El uso de canales y mensajes XML para la comunicación entre servidores ha resultado como un enfoque muy útil al momento de desarrollar aplicaciones como estas, puesto que brindan una interacción eficiente, facililtan el crecimiento de sus funcionalidades y permiten la creación de nuevos canales tecnológicos para cumplir tareas extas que se puedan requerir a futuro.

Capítulo 12: Recomendaciones

- Al momento de desarrollar aplicaciones móviles simpre es importante tener en cuenta los requerimientos de compatibilidad de estas. Si se requiere que la aplicación funcione únicamente en una sola plataforma es recomendable hacer uso de su lenguaje nativo para implementarla puesto que aunque la curva de aprendizaje se eleve un poco se pueden aprovechar de mejor manera las características del dispositivo.
- Cuándo se implementan aplicaciones móviles se debería hacer de manera previa un análisis estadístico para determinar la plataforma móvil que predomine entre los usuarios en ese momento. De ese modo se puede saber con que dispositivos tendrá que ser compatible la aplicación, y se logrará satisfacer a la mayor cantidad de usuarios posible.

Capítulo 13: Bibliografía

[1] N. Mehta. *Mobile Web Development.* Birmingham: Packt Publishing, 2008.

[2] D. Hazael-Massieux. *Standards for Web Applications on Mobile: February 2011 current state and roadmap.* Febrero de 2011. www.w3c.org/2011/02/mobile-web-app-state.html (último acceso: Noviembre de 2011).

[3] H. Dwivedi, C. Clark and D. Thiel. *Mobile Application Security.* New York: McGraw-Hill, 2010.

[4] A. Connors and B. Sullivan. *Mobile Web Application Best Practices.* 14 de Diciembre de 2010. (último acceso: Noviembre de 2011).

[5] Blackberry Support Community. *Introduction to BlackBerry Development Options - Blackberry Support Community Forums.* 13 de Mayo de 2010. http://supportforums.blackberry.com/t5/Java-Development/Introduction-to-BlackBerry-Development-Options/ta-p/503185?name=IntroBlackBerryDev (último acceso: 20 de Noviembre de 2011).

[6] Research In Motion. *Blackberry Java Development Enviroment v4.6.0: Guia de Conceptos Básicos.* Waterloo: Research In Motion, 2008.

[7] —. *Set up for signing smartphone apps - Blackberry HTML5/Webworks.* Enero de 2012. https://developer.blackberry.com/html5/documentation/ww_publishing/signing_setup_smartphone_apps_1920010_11.html (último acceso: 10 de Enero de 2012).

[8] Apple Inc. *iOS Human Interface Guidelines.* 12 de Octubre de 2011. https://developer.apple.com/library/safari/#documentation/UserExperience/Conceptual/MobileHIG/Characteristics/Characteristics.html#//apple_ref/doc/uid/TP40006556-CH7-SW1 (último acceso: Marzo de 2012).

[9] Android. *Web Apps Overview - Android Developers.* Enero de 2012. http://developer.android.com/guide/webapps/overview.html (último acceso: Marzo de 2012).

[10] Android. *Signing Your Applications | Android Developers.* Enero de 2012. http://developer.android.com/tools/publishing/app-signing.html (último acceso: Abril de 2012).

[11] —. *HTML5.* 25 de Mayo de 2011. http://www.w3.org/TR/html5/spec.html (último acceso: Diciembre de 2011).

[12] —. *HTML5 differences from HTML4.* 25 de Mayo de 2011. http://www.w3.org/TR/html5-diff/ (último acceso: Diciembre de 2011).
[13] MIT, ERCIM and Keio. *Cascading Style Sheets (CSS) Snapshot 2010.* 12 de Mayo de 2011. http://www.w3.org/TR/CSS/ (último acceso: Marzo de 2012).
[14] —. *CSS Mobile Profile 2.0.* 10 de Diciembre de 2008. http://www.w3.org/TR/css-mobile/ (último acceso: Marzo de 2012).
[15] —. *CSS Device Adaptacion.* 15 de Septiembre de 2011. http://dev.w3.org/csswg/css-device-adapt/ (último acceso: Marzo de 2012).
[16] —. *Media Queries.* Abril de 2012. http://www.w3.org/TR/css3-mediaqueries/ (último acceso: Abril de 2012).
[17] jQuery, Foundation. *jQuery Mobile: Demos and Documentation.* 2012. http://jquerymobile.com/demos/1.1.0/ (último acceso: Febrero 2012).
[18] jQuery, Foundation. *Mobile Graded Browser Support | jQuery Mobile.* 2012. http://jquerymobile.com/demos/1.1.0/ (último acceso: Mayo 2012).
[19] jQuery, Foundation. *Original Graded Browser Matrix | jQuery Mobile.* 2012.
[20] Adobe Systems. *PhoneGap.* Noviembre de 2011. https://phonegap.com/ (último acceso: Noviembre de 2011).
[21] —. *PhoneGap Build.* Noviembre de 2011. https://build.phonegap.com/docs (último acceso: Noviembre de 2011).
[22] ScientiaMobile Inc. *ScientiaMobile - Mobile Device Detection Products.* Noviembre de 2011. http://www.scientiamobile.com/what (último acceso: Noviembre de 2011).
[23] H. Braun. *Admin Guide - JBoss AS 7.0 - Project Documentation Editor.* Julio de 2011. https://docs.jboss.org/author/display/AS7/Admin+Guide (último acceso: Noviembre de 2011).
[24] J. Goodwill. *Developing Java Servlets 2nd Edition.* Sams Publishing, 2001.
http://jquerymobile.com/original-graded-browser-matrix/ (último acceso: Mayo 2012).
[25] S. Guruzu and G. Mak. *Hibernate Recipes: A Problem-Solution Approach.* New York: Apress, 2010.
[26] G. King, C. Bauer, A. Rydahl Andersen, E. Bernarda and S. Ebersole. *Hibernate: Relational Persistence for Idiomatic Java.* 24 de Junio de 2009. http://docs.jboss.org/hibernate/core/3.3/reference/en/html/index.html (último acceso: Noviembre de 2011).
[27] C. Minnick. *WebKit for Dummies.* Hoboken: John Wiley & Sons, Inc., 2012.
[28] —. *Safari HTML Reference.* 12 de Julio de 2011.
[29] S. Holzner. *Design Patterns for Dummies.* Hoboken: Wiley Publishing, Inc., 2006.
https://developer.apple.com/library/safari/#documentation/appleapplications/reference/SafariHTMLRef/Articles/Attributes.html#//apple_ref/doc/uid/TP40008058-SW1 (último acceso: Marzo de 2012).
[30] FitBank. *UCI.* http://bantecinc.com/files/UCI.PDF (último acceso: Diciembre de 2011)

[31] FitBank. *Core Bancario.* http://bantecinc.com/files/Fitbank.pdf (último acceso: Diciembre de 2011)
[32] B. Collins-Sussman, B. Fitzpatrick and M. Pilato. *Version Control with Subversion.* 2006.
[33] InfoEther. *Welcome to PMD.* 2011. http://pmd.sourceforge.net/pmd-5.0.0/ (último acceso: Noviembre de 2011).
[34] comScore. *Tráfico móvil en Latinoamérica, el rey es iOS.* Marzo de 2012. http://www.puntogeek.com/2012/03/07/trafico-movil-en-latinoamerica-el-rey-es-ios-grafico/ (último acceso: Marzo de 2012).
[35] IBM. *Desarrollo de la gestión de sesiones en servlets.* 18 de Febrero de 2009. http://pic.dhe.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=%2Fcom.ibm.websphere.express.doc%2Finfo%2Fexp%2Fae%2Ftprs_sesi.html (último acceso: Marzo de 2012).
[36] —. *Blackberry WebWorks SDK v1.5 Development Guide.* Waterloo: Research In Motion, 2010.
[37] —. *Networking Transports II - Blackberry Support Community Forums.* 18 de Febrero de 2010. http://supportforums.blackberry.com/t5/Java-Development/Networking-Transports-II/ta-p/446742?name=NetworkingTransportsII (último acceso: 20 de Noviembre de 2011).

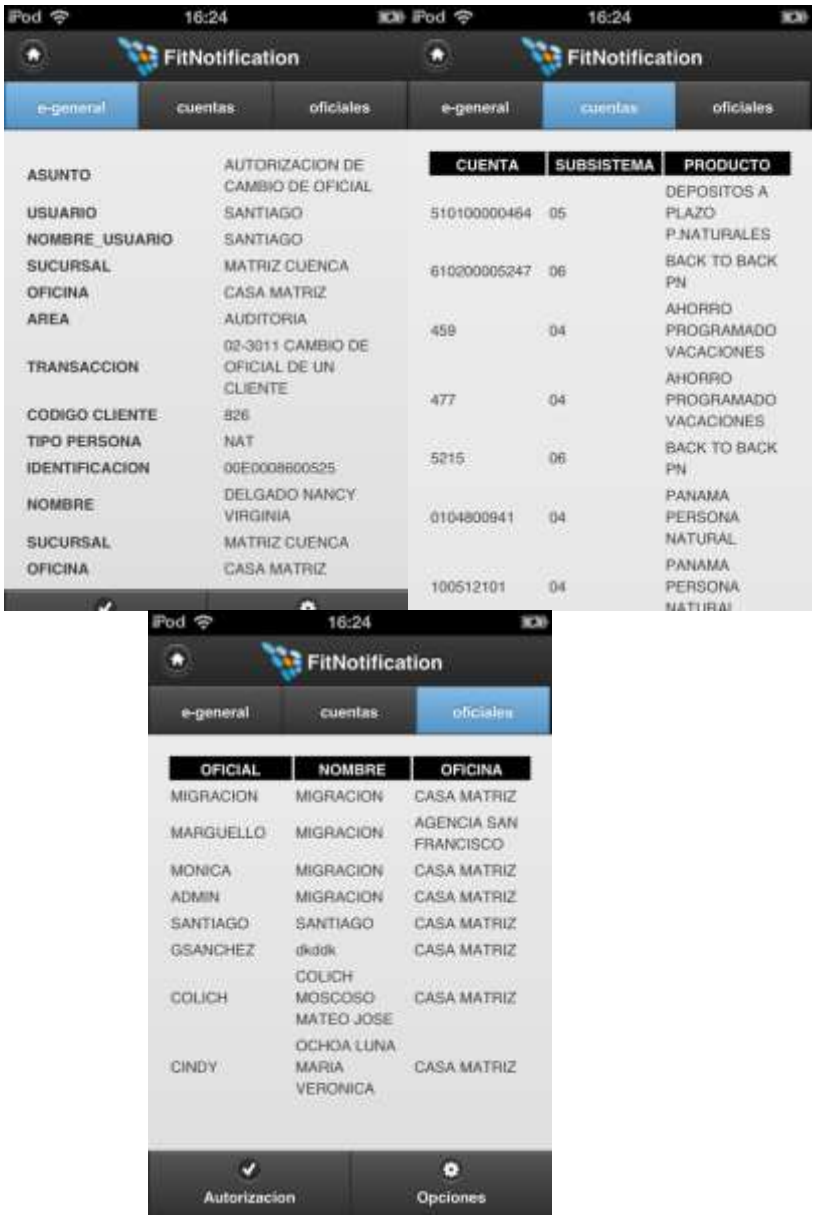
Capítulo 14: Anexos

14.1. Pruebas de ejecución de autorizaciones

Cambio de Oficial de Cliente


CAMBIO DE OFICIAL DE CLIENTE - FitBank - Google Chrome
CAMBIO DE OFICIAL DE CLIENTE
DESARROLLO
NIVEL UNO
CAMBIO DE OFICIAL DE CLIENTE
Nombre Sucursal
Nombre Oficina
Nombre Oficial
MATRIZ CUENCA
CASA MATRIZ
MONICA
BANTEC
SE HA INICIADO EL PROCESO DE AUTORIZACION


iPod
16:23
FitNotification
Salir
HOLA MIGRACION
PERSONAS
1
AUTORIZACION DE CAMBIO DE OFICIAL
Usuario Origen: SANTIAGO



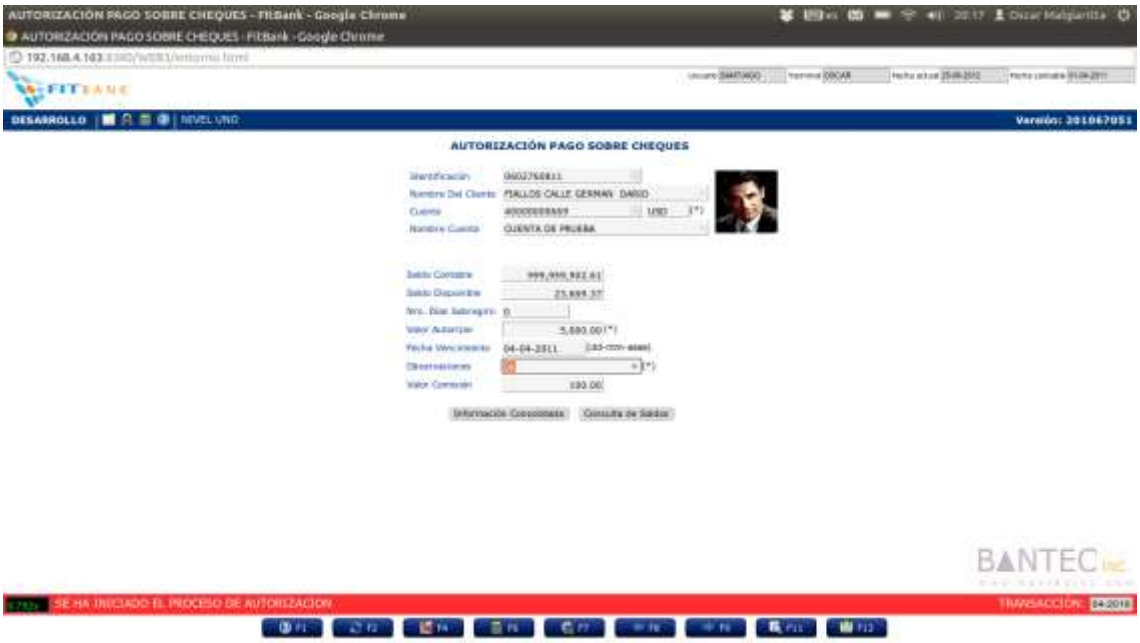
Autorización de Sobregiros Contratados


AUTORIZACION DE SOBREGIROS CONTRATADOS - FitBank - Google Chrome
AUTORIZACIÓN DE SOBREGIROS CONTRATADOS - FitBank - Google Chrome
DESARROLLO
NIVEL UNO
AUTORIZACIÓN DE SOBREGIROS CONTRATADOS
DETALLE AUTORIZACIÓN
BANTEC
SE HA INICIADO EL PROCESO DE AUTORIZACION.


iPod
16:40
AUTORIZACION PARA SOBREGIRO
Usuario Origen: SANTIAGO
AUTORIZACION PARA SOBREGIRO
Usuario Origen: SANTIAGO
AUTORIZACION PARA SOBREGIRO
Usuario Origen: SANTIAGO
GARANTIAS
Autorizacion
Opciones

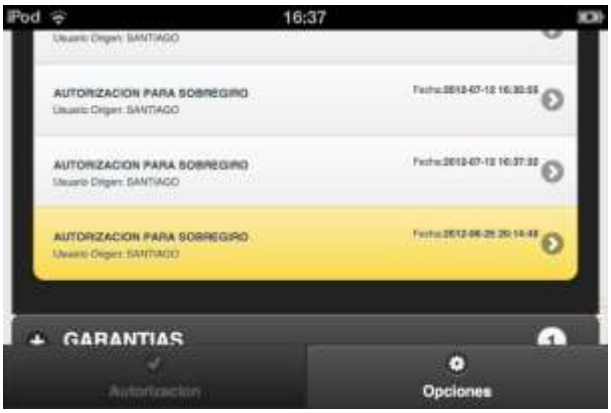
Autorizaciones de Sobregiros Ocasionales

Autorización pago sobre cheques


AUTORIZACION PAGO SOBRE CHEQUES - FitBank - Google Chrome
AUTORIZACIÓN PAGO SOBRE CHEQUES - FitBank - Google Chrome
DESARROLLO
NIVEL UNO
AUTORIZACIÓN PAGO SOBRE CHEQUES
BANTEC
SE HA INICIADO EL PROCESO DE AUTORIZACION.


iPod
16:34
AUTORIZACION PARA SOBREGIRO
AUTORIZACION PARA SOBREGIRO
AUTORIZACION PARA SOBREGIRO
AUTORIZACION PARA SOBREGIRO
GARANTIAS

Autorización de Aumento del Valor del Sobregiro Contratado

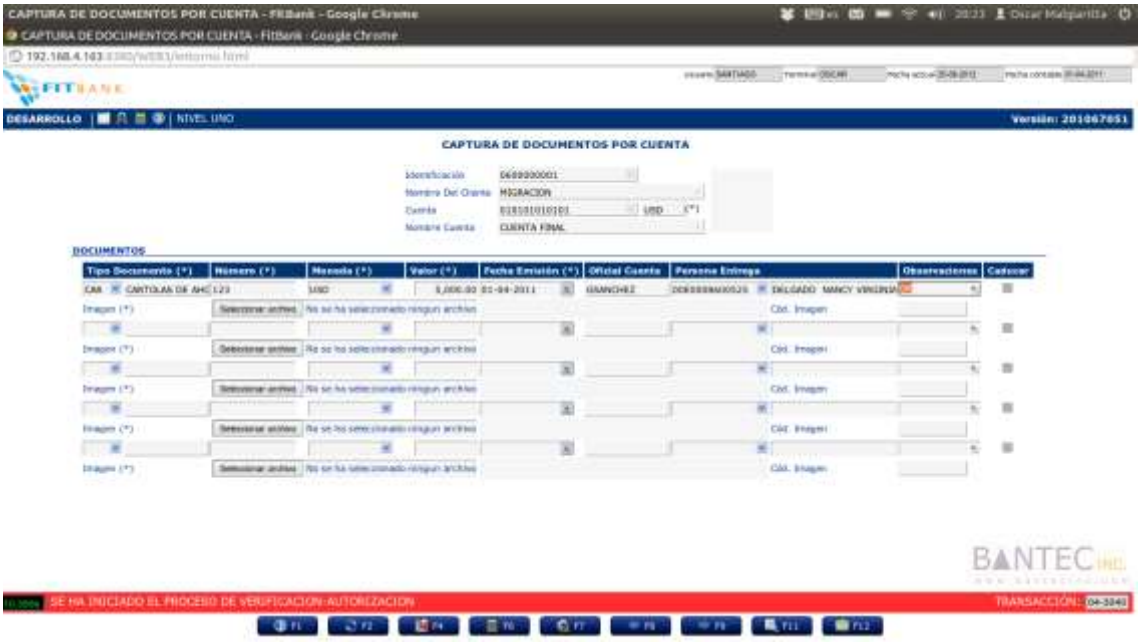


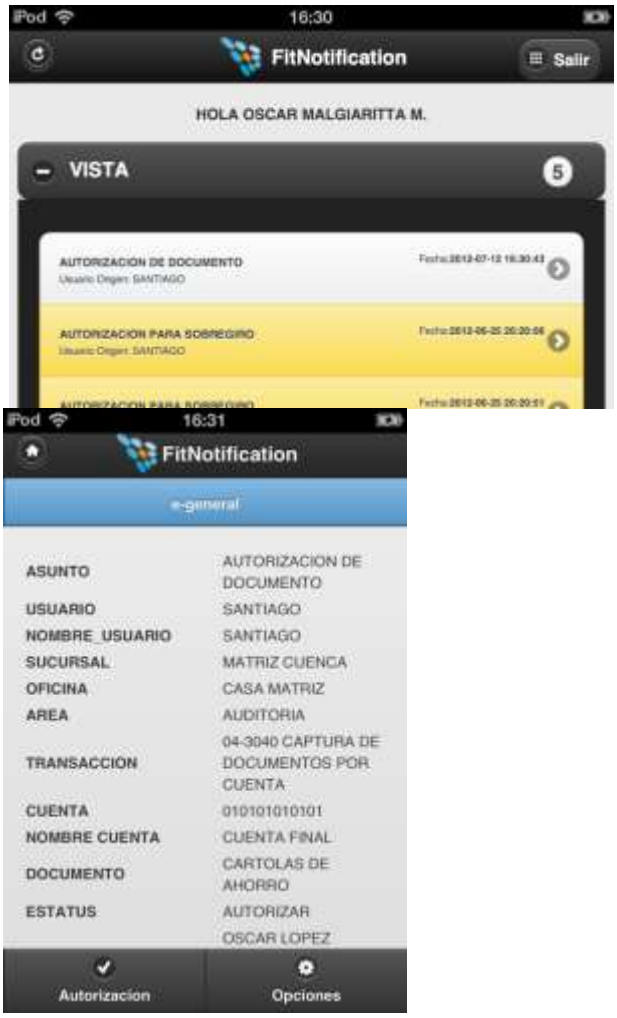
Solicitud de Chequeras Especiales


SOLICITUD DE CHEQUERAS ESPECIALES - FitBank - Google Chrome
DESARROLLO
NIVEL UNO
SOLICITUD DE CHEQUERAS ESPECIALES
CUENTA DE PRUEBA
SE HA INICIADO EL PROCESO DE AUTORIZACION

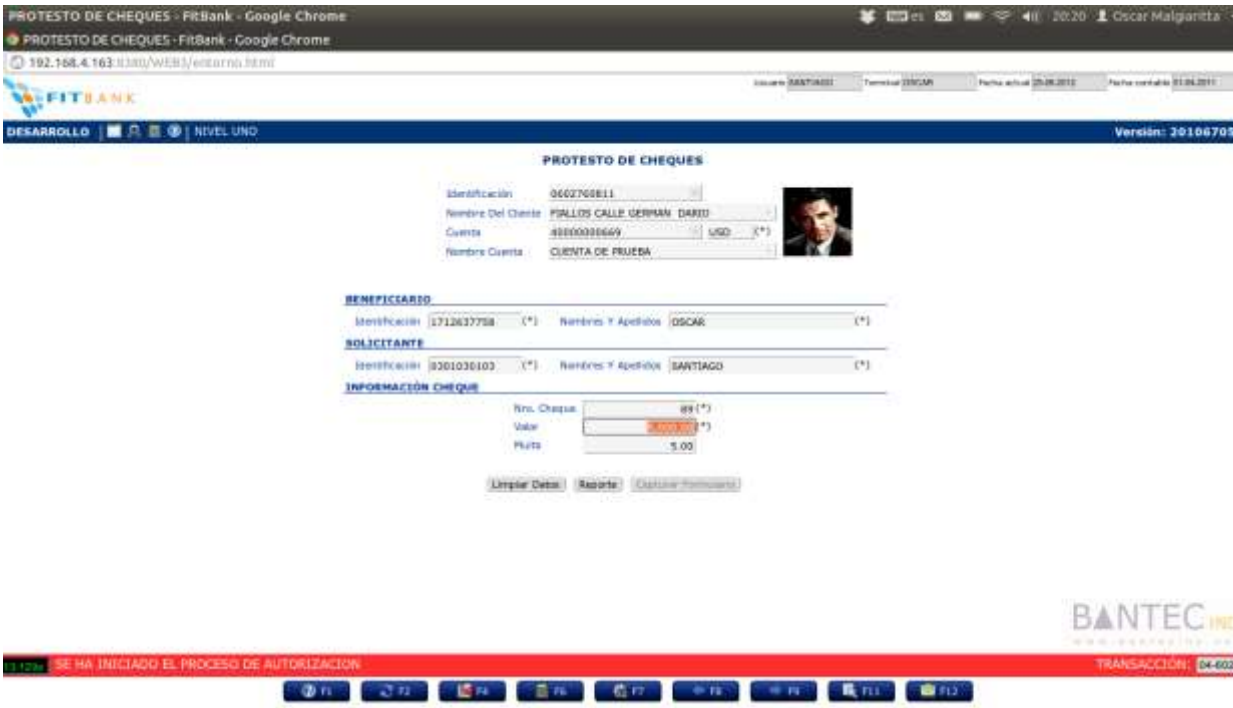

iPod
16:46
FitNotification
Salir
HOLA SANTIAGO GONZALEZ T.
VISTA
1
AUTORIZACION CREAR NUEVA CHEQUERA

Captura de Documentos por Cuenta





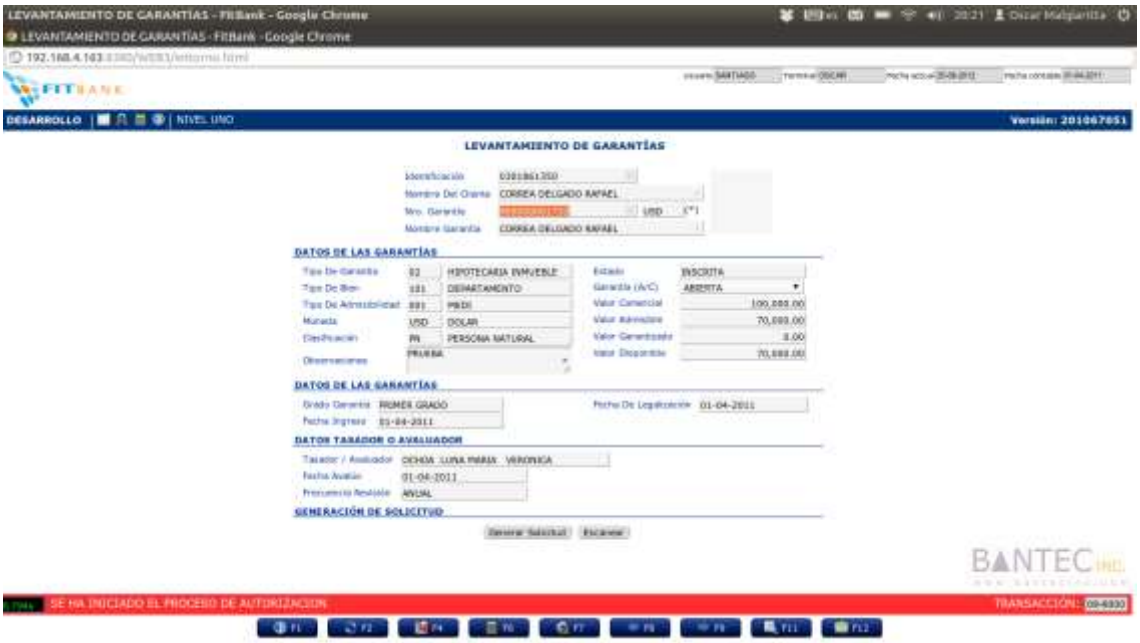
Protesto de Cheques


PROTESTO DE CHEQUES - FitBank - Google Chrome
PROTESTO DE CHEQUES
DESARROLLO
NIVEL UNO
PROTESTO DE CHEQUES
BENEFICIARIO
SOLICITANTE
INFORMACIÓN CHEQUE
Limpiar Datos
Reporte
SE HA INICIADO EL PROCESO DE AUTORIZACION
TRANSACCIÓN:


iPod
16:27
HOLA DAVID ANDRES
VISTA
2
AUTORIZACION DE PROTESTO DE CHEQUE
AUTORIZACION DE PROTESTO DE CHEQUE

Protestos de Cheques a una Cuenta Cerrada

Levantamiento de Garantías


LEVANTAMIENTO DE GARANTIAS - FitBank - Google Chrome
DESARROLLO
NIVEL UNO
LEVANTAMIENTO DE GARANTÍAS
DATOS DE LAS GARANTÍAS
DATOS DE LAS GARANTÍAS
DATOS TASADOR O AVALUADOR
GENERACIÓN DE SOLICITUD
BANTEC
SE HA INICIADO EL PROCESO DE AUTORIZACION


iPod
16:32
Salir
HOLA OSCAR MALGIARITTA M.
VISTA
5
GARANTIAS
1
AUTORIZACION DE LEVANTAMIENTO DE LA G...
Opciones


iPod 16:32
FitNotification
garantia
e-general
ASUNTO
AUTORIZACION DE LEVANTAMIENTO DE LA GARANTIA
USUARIO
SANTIAGO
NOMBRE_USUARIO
SANTIAGO
SUCURSAL
MATRIZ CUENCA
OFICINA
CASA MATRIZ
AREA
AUDITORIA
TRANSACCION
09-6000 LEVANTAMIENTO DE GARANTIAS
NUMERO GARANTIA
900000001733
CODIGO CLIENTE
1176792
IDENTIFICACION
0301861350
NOMBRE
CORREA DELGADO RAFAEL OSCAR LOPEZ
iPod 16:32
FitNotification
garantia
e-general
GARANTIA
900000001733
MONEDA
USD
FRECUENCIA INSPECCION
8 - ANUAL
AVALUADOR
0104039144 - OCHOA LUNA MARIA VERONICA
TIPO GARANTIA
02 - HIPOTECARIA INMUEBLE
TIPO BIEN
101 - DEPARTAMENTO
DETALLE
PRUEBA
VALOR
100000
Autorizacion
Opciones