Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

"Interbank Banca Móvil iPad: Cliente iPad para la plataforma bancaria de la empresa Interbank, proyecto para optar por el título de Ingeniero en Computación con el grado académico de Bachiller."

Luis Alonso Vega Brenes

Lima, Octubre, 2013

Resumen Ejecutivo

El presente documento es el informe del proyecto Interbank Banca Móvil. Se abarcan distintas áreas del proceso de realización de la aplicación final, desde su descripción y contexto hasta una explicación desarrollada de conceptos más técnicos utilizados para la resolución del problema.

En la primera sección se verán antecedentes del proyecto, incluyendo la empresa, el entorno, el problema a solucionar, los posibles riesgos, criterios de aceptación del cliente, entre otros.

Más adelante se muestran detalles conceptuales, estructurales e internos del aplicativo. Incluyendo la arquitectura conceptual, con un vistazo a los diversos módulos de usuario así como módulos internos de manejo de servicios y operaciones. Un diagrama de clases con los principales bloques de construcción del sistema, desde modelos de datos hasta controladores. Se incluye una visión a la jerarquía y dependencias de clases para secciones compartidas como las de recuperación de claves dinámicas, así como módulos encargados únicamente del manejo de vistas de confirmación o resultado de las operaciones.

Se incluye un apartado para la interfaz de usuario y su historial de diseño, desde la concepción de los mock-ups hasta su actual implementación, incluyendo los distintos cambios que sufrió en varias etapas del proyecto según cada revisión hecha. Se dará una visión de la aplicación como una serie de componentes y servicios interactuando entre sí, y una explicación sobre cómo todos estos se correlacionan para formar el producto final.

Al final del documento se presenta una breve sección de conclusiones, sugerencias y comentarios relacionados con toda la etapa de desarrollo del aplicativo.

Palabras claves: iOS; iPad; Interbank; Aplicación móvil; Desarrollo móvil.

Tabla de contenido

1 Descripción del proyecto

1.1 Contexto del proyecto

El proyecto desarrollado consiste en una aplicación de banca para tabletas iPad, específicamente para la corporación de servicios financieros Interbank. Dicha entidad constituye una de las mayores instituciones financieras de Perú, con más de 4700 empleados para el año 2008, 230 tiendas y unos 3000 cajeros y agentes en todo Perú.

Como tal, se planea realizar un cliente para la plataforma iOS en los dispositivos iPad. El mismo debe tener la misma funcionalidad que su contraparte iPhone, la cual ofrece operaciones de transferencias, pagos de tarjetas y de servicios, sistema de ubicación de sucursales del banco, suscripción a inversiones y rescates, entre otros. Estas funcionalidades, presentes en los distintos clientes, los cuales incluyen Android, iPhone y Black Berry, son una subdivisión de un servicio web ofrecido por el banco.

El objetivo de crear dos aplicaciones distintas para el mismo sistema operativo (pero en distintos dispositivos) es aprovechar las características del iPad, específicamente su mayor espacio en pantalla, para agilizar la navegación y la realización de operaciones. Este replanteamiento, como se plantea en la propuesta de proyecto existente, requiere un cambio en muchos aspectos de la aplicación de iPhone, desde cuestiones de diseño gráfico hasta una reestructuración en la navegación que permita mayor rapidez.

1.2 Descripción del problema

Interbank cuenta con una plataforma web que permite a sus clientes realizar una serie de operaciones bancarias. El desarrollo tecnológico del banco Interbank ha permitido ofrecer clientes móviles para teléfonos inteligentes que permiten hacer uso de dichas operaciones. Estos clientes incluyen aplicaciones para los sistemas operativos Android, iOS y Black Berry.

El banco ha decidido comenzar a ofrecer aplicaciones móviles para tabletas electrónicas también, iniciando con el dispositivo de Apple: iPad. Con este nuevo cliente será posible hacer uso de los servicios de Interbank con mayor agilidad que utilizando sus otros clientes para teléfonos.

1.3 Análisis de riesgos

Se identificaron los siguientes riesgos de proyecto:

- No contar con la disponibilidad de los servicios en el backend del banco.
- Poca disposición del personal técnico en el lado del cliente puede dar lugar a retrasos en la instalación, configuración y despliegue del producto en ambientes de prueba o de producción.
- Rotación de personal asignado al proyecto puede dar lugar a retrasos en el proyecto.
- Tareas inesperadas, que no se consideraron como parte de este alcance, pueden aparecer durante el desarrollo incrementando el tiempo requerido para completar la solución.
- No haber realizado la refactorización de Banca Móvil imposibilitará la ejecución de este proyecto.

1.4 Objetivos y alcances del sistema

1.4.1 Objetivos

1.4.1.1 Objetivo general

- El objetivo principal de este proyecto es desarrollar una nueva versión de Banca Móvil pero diseñada para dispositivos iPad usando como base el código actualizado de Banca Móvil (luego del refactoring).

1.4.1.2 Objetivos específicos

- Mantener las funcionalidades existentes de Banca Móvil.
- Utilizar el código actualizado de Banca Móvil luego del refactoring.
- Rediseñar la aplicación en términos de interfaz gráfica y de navegación para aprovechar el mayor espacio disponible del dispositivo meta.

1.4.2 Documentos relacionados

El documento de propuesta está basado en información oral y escrita proporcionada por el cliente, del cual Avantica asume que es precisa y válida.

1.4.3 Requerimientos del cliente

1.4.3.1 Requerimientos funcionales

La solución propuesta debe implementar los siguientes requerimientos funcionales:

- Las funcionalidades existentes deben mantenerse:
  - Vista de cuentas propias, incluyendo detalles y movimientos.
  - Pago de tarjetas propias, tarjetas del Interbank y de otros bancos.
  - Bloqueo de tarjetas propias.
  - Transferencias entre cuentas propias, hacia cuentas del Interbank y hacia otros bancos.
  - Suscripciones a contratos propios, contratos de terceros y rescates.
  - Pago de servicios.
  - Módulo de ubicación donde se muestren las sucursales del banco más cercanas, incluyendo cajeros, oficinas y tiendas.
- El flujo actual debe adecuarse para aprovechar todo el espacio disponible en los dispositivos iPad.

1.4.3.2 Requerimientos no funcionales

La solución del proyecto debe implementar los siguientes requerimientos no funcionales:

- El desarrollo se realizará basándose en la versión actualizada de Banca Móvil iOS (luego de la refactorización).
- Se reutilizarán los servicios que actualmente se están utilizando para Banca Móvil iOS, no se crearán nuevos servicios ni se modificarán los existentes.
- Se considera un trabajo de interfaces propias para dispositivos iPad, pero sin llegar a rediseñar la aplicación en su totalidad.
- El diseño gráfico forma parte de la propuesta.

1.4.4 Supuestos

Los supuestos de este proyecto son los siguientes:

- El desarrollo se realizará basándose en los estándares manejados hasta el momento en Banca Móvil.
- La refactorización de Banca Móvil se debe haber realizado previamente para poder ejecutar este proyecto.
- El personal de Interbank estará disponible para facilitar una rápida respuesta a las consultas técnicas y funcionales que puedan surgir durante el desarrollo del proyecto. Existirá un contacto técnico de alto nivel que facilite estas tareas.
- Interbank proveerá la data de prueba necesaria para la ejecución de pruebas durante las etapas de desarrollo y pruebas del proyecto.
- Avantica asegurará que la aplicación funcionará correctamente en los siguientes dispositivos:
    - iPad con iOS 6
    - iPad 4 con iOS 6

1.4.5 Exclusiones

- Las funcionalidades ajenas a las detalladas en la sección "Requerimientos Funcionales".
- Configuración de la infraestructura, soporte y mantenimiento. Incluidos los servidores físicos, sistemas operativos, servidores y sistemas relacionados con el software, las aplicaciones de terceros, etc.
- Diseño, desarrollo y pruebas de los servicios que consumirá el Backend.
- Software o configuración de hardware, mantenimiento, soporte y acceso a la concesión de los sistemas donde el servicio web, reside y trabaja.
- Capacitación técnica y capacitación a usuarios.
- Despliegue de la aplicación en el ambiente de UAT ni en el ambiente de Producción.
- Despliegue de la aplicación en el Apple Store.

1.4.6 Criterios de Aceptación

La aceptación del producto será lograda cuando:

- Se haya conseguido la aceptación por parte del departamento de certificación de software de Interbank.
- La estimación ha sido realizada tomando en consideración un máximo de 21 días hábiles para que Interbank inicie el reporte de las incidencias detectadas en su proceso de Certificación, contados a partir del siguiente día hábil de la fecha de entrega del producto, pasado este período, Avantica no garantiza la continuidad de los recursos que desarrollaron la aplicación; asimismo, la asignación de otros recursos no impactará la calidad ni el tiempo de corrección de las incidencias.

1.4.7 Consideraciones Especiales

El cliente debe ser consciente de las siguientes consideraciones:

- Cualquier solicitud adicional no considerada dentro de este documento tiene que ser evaluada y negociada con el Director de Proyecto asignado en su momento por Avantica para administrar el desarrollo del mismo.
- Si el cliente pide algún hardware adicional, aparte de las estaciones de trabajo de desarrollo y servidores que Avantica utiliza actualmente, este debe ser proporcionado por el cliente.
- El cliente acepta no contratar directamente ningún recurso de Avantica asignado al proyecto dentro de los siguientes 12 meses después de finalizar el proyecto; si esta situación ocurre, el cliente estaría compensando a Avantica con el pago de un total de 12 meses de servicios profesionales de un recurso.
- Avantica está comprometido a no contratar ningún empleado del cliente en los siguientes 12 meses después de terminado el proyecto.
- Los empleados de Avantica tendrán derecho a todos los días feriados reconocidos por la empresa en el país donde el empleado está trabajando. Además, cada empleado tiene derecho a un día de vacaciones por mes, que se pueden tomar o acumular, en virtud de acuerdo previo de todas las partes (el cliente, Avantica y el empleado).
- Los cambios en el diseño no son considerados dentro de esta propuesta.

2 Solución implementada

2.1 Arquitectura conceptual de la solución

La aplicación sigue un modelo conceptual de uso sencillo, basándose en un sistema Web de mayor complejidad ubicado en un servidor de la empresa, en este caso el Banco. Esta aplicación tiene una gran cantidad de módulos y funciones, de los cuales se seleccionaron las más importantes para implementar en una aplicación móvil. Los módulos indicados son los siguientes:

- Ingreso
- Listado y detalles de mis cuentas
- Pago de tarjetas de crédito
- Bloqueo de tarjeta
- Inversiones y fondos mutuos
- Pago de recibos
- Sistema de ubicación de sucursales

Dichos módulos realizan operaciones en las que se comunican con el servidor web enviando un mensaje con varios parámetros que definen el tipo de operación y los datos necesarios para realizarlas, y obtienen un mensaje del mismo con el resultado de la operación, ya sea de transacción exitosa junto con datos como el número de operación, o de error, indicando el código y descripción del mismo.

El modelo de uso se basa en dos movimientos principales: uno de ingreso con credenciales a la aplicación y luego una vista principal con sus distintos módulos en un menú accesible en todo momento.

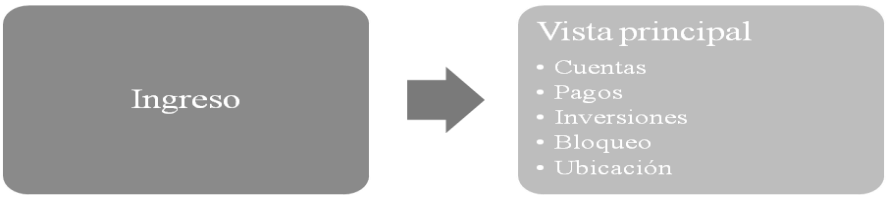

Figura 1 Modelo general de navegación

Sin embargo, esto no es más que el modelo transparente que lograría comprender un usuario de la aplicación. En un nivel de jerarquía interna, aparece una capa de funcionalidad menos visible encargada de llevar a cabo servicios como el de conexión, seguridad, manejo de errores y demás utilidades.

En la figura 2 se muestran los componentes según su categoría dentro de la aplicación.

El diagrama es solo una abstracción y cada componente puede estar conformado por una o varias clases a nivel de código fuente.

La interacción entre componentes puede ser un tanto extensa de explicar. Para resumir, se puede mencionar que los controladores son los encargados de llamar a todos los demás componentes, mientras que estos últimos son más independientes entre sí y pocas o ninguna vez requieren invocarse entre sí.

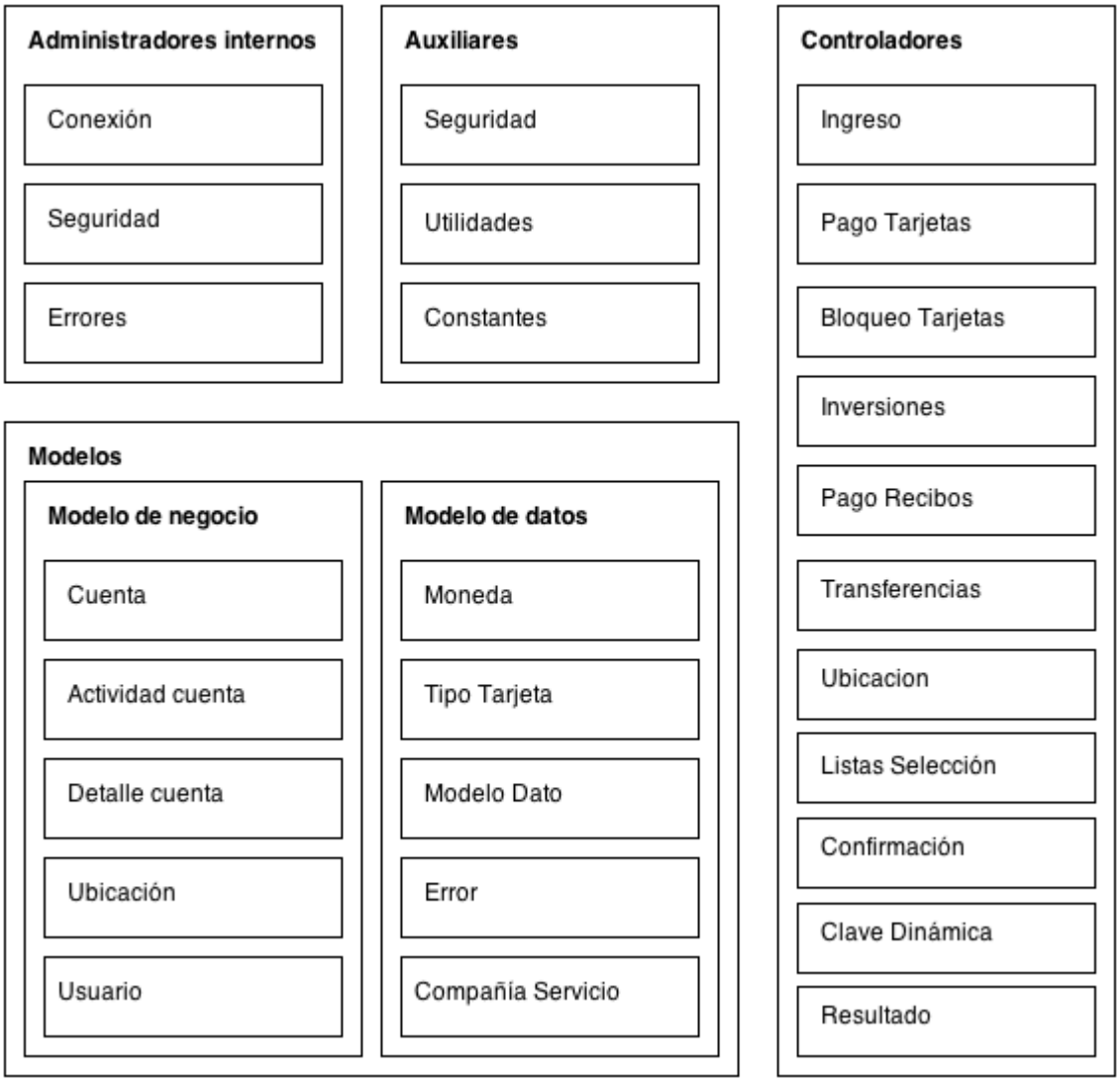

Figura 2 Diagrama de componentes

Los controladores tienen la función de ser los que procesan todas las entradas del usuario, y de ejecutar cualquier operación o transacción según se indique. Para ello, se hace uso de los distintos modelos de datos.

Un ejemplo simple: el usuario elige realizar una transferencia, para lo cual debe seleccionar dos cuentas, una a la que se le hará el cargo (una cuenta propia) y una cuenta destino donde se depositará el dinero (la cual podrá ser propia o de alguien más). En estos casos las listas de cuentas se manejan bajo el modelo de datos. Este ejemplo se visualiza en la figura 3, en la que

se muestra un diagrama de la vista de transferencias. Los campos de cuenta cargo y cuenta a transferir poseen un menú común de la lista de cuentas, la cual está sujeta a su sub controlador asignado.

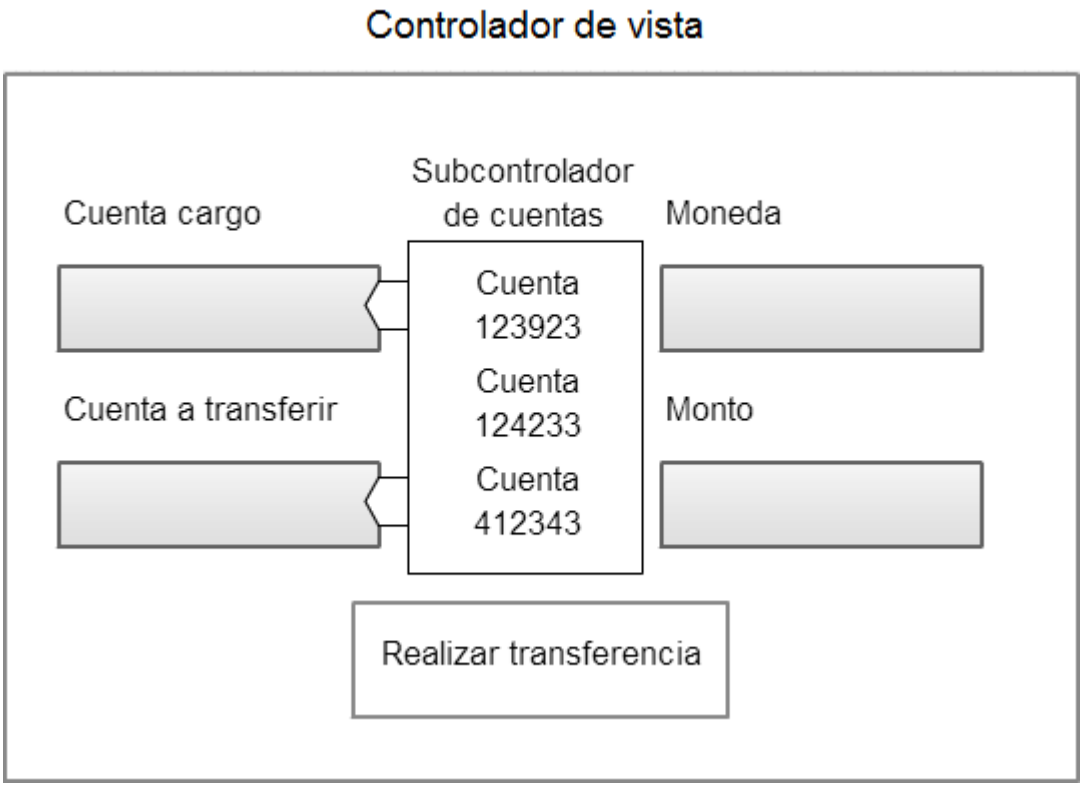

Figura 3 Modelo de formulario de vista

Aparte de los modelos, los controladores utilizarán los módulos de administradores y auxiliares al realizar operaciones. Para continuar con el ejemplo, al haber seleccionado cuentas de cargo y destino, y un monto a transferir, el usuario procede a ejecutar la transacción. Para esto el controlador recurrirá al módulo de conexión, tomando los datos de las cuentas modelo y empaquetando en un mensaje lo necesario para ser enviado al servidor. En algunos módulos es necesario realizar primero una confirmación de operación, en la cual el usuario simplemente debe permitir la ejecución, así como en algunos módulos además se requiere de ingresar una clave dinámica para realizar transferencias a cuentas externas. Estas confirmaciones y

solicitudes de claves dinámicas son los módulos que aparecen al final en el diagrama anterior, así como el de resultado que aparece si la operación resultó con éxito.

En el caso de que una operación falle por cualquier razón, el módulo de conexión obtendrá del servidor un código numérico de error. El módulo de errores es el encargado de conocer estos códigos y su significado, así que al fallar se enviará esta información directamente al controlador para que lo procese y muestre al usuario lo ocurrido.

2.1.1 *Comunicación entre módulos*

Este es un apartado extra para mencionar cómo se comunican los módulos entre sí. En este lenguaje y tecnología, la forma más recomendada de compartir datos o eventos entre clases es utilizando delegados[1]. Estos consisten en interfaces de clases para la implementación de ciertos métodos en una clase receptora que luego pueden ser invocados desde una clase emisora que quiera enviar un mensaje.

Los delegados permiten realizar operaciones asíncronas, pues los métodos (o mensajes, como se llaman correctamente en el lenguaje Objective C) simplemente tienen una declaración que podrá ser invocada por cualquier otra clase que así lo requiera.

Un ejemplo claro para este modelo de diseño ocurre con el módulo de conexión. Al realizar una operación, un controlador empaqueta un mensaje y se lo envía al módulo de conexión para que este a su vez lo envíe al servidor. Este mensaje se envía mediante la red pero la respuesta no es exactamente inmediata y la aplicación no puede detenerse 2a esperar que el servidor responda.

---

[1] Concepts in Objective C – C Programming, Delegates and Data Sources. Consultado el 31 de octubre de 2013, en https://developer.apple.com/library/ios/documentation/general/conceptual/CocoaEncyclopedia/DelegatesandDataSources/DelegatesandDataSources.html

El módulo de conexión se encarga de escuchar cuando la respuesta llega, y debe comunicárselo al controlador que envió originalmente el mensaje. Para esto, cada controlador de los módulos de la pantalla principal implementa un delegado de conexión, en el cual pueden recibir la respuesta del módulo conexión, ya sea de éxito o de error.

2.2 Diagramas de Clases

Este es un diagrama de las principales clases de la aplicación. Se podrá observar que la jerarquía es similar a la planteada en el modelo conceptual, con la variante de que algunos componentes requieren a su vez de más de una clase.

En el caso de los modelos de datos y varias clases auxiliares se omitieron las relaciones con otras clases por ser muchas y ofuscarían el entendimiento del diagrama.

El grupo más grande está compuesto por los controladores de las distintas vistas. Este grupo a su vez se divide en tres subgrupos:

- Módulos: Estas clases manejan el funcionamiento de cada una de las vistas principales de la aplicación, incluyendo las vistas de Ingreso de sesión, contraseña, transferencias, pagos y demás.
- Diálogos: Incluyen los submódulos que muestran confirmación y resultados de operación. También pertenecen a este grupo las solicitudes de clave dinámica y los delegados (clases implementadas para seguir el patrón de diseño estructural).
- Módulos auxiliares: Representan las clases controladoras de vistas que se muestran como menús emergentes. Ejemplo: Al seleccionar una cuenta bancaria, al elegir un tipo de moneda, al seleccionar una empresa de servicio, etc.

Por otra parte se encuentran las clases administradoras, las cuales manejan las siguientes áreas:

- Seguridad: Comprobar si las operaciones (pago de tarjetas, transferencias) se encuentran disponibles para el usuario actual.
- Conexión: Realizar la conexión al servidor, enviar y recibir datos.
- Intérprete: Leer secuencias de caracteres recibidas y transcribirlas a un formato entendible por la aplicación.

- Errores: Se encarga de obtener la información de un error a partir de su código y de informar a sus delegados.

Otras clases auxiliares incluyen:

- Contenedor de llavero: Permite guardar y obtener datos de forma segura. Abstrae un funcionamiento nativo de la plataforma.
- Utilidades: Varias funciones de conversión y procesamiento básico de datos.

Las clases del modelo de datos son bastante auto-explicativas, pues únicamente poseen campos y métodos de acceso según el tipo de objeto.

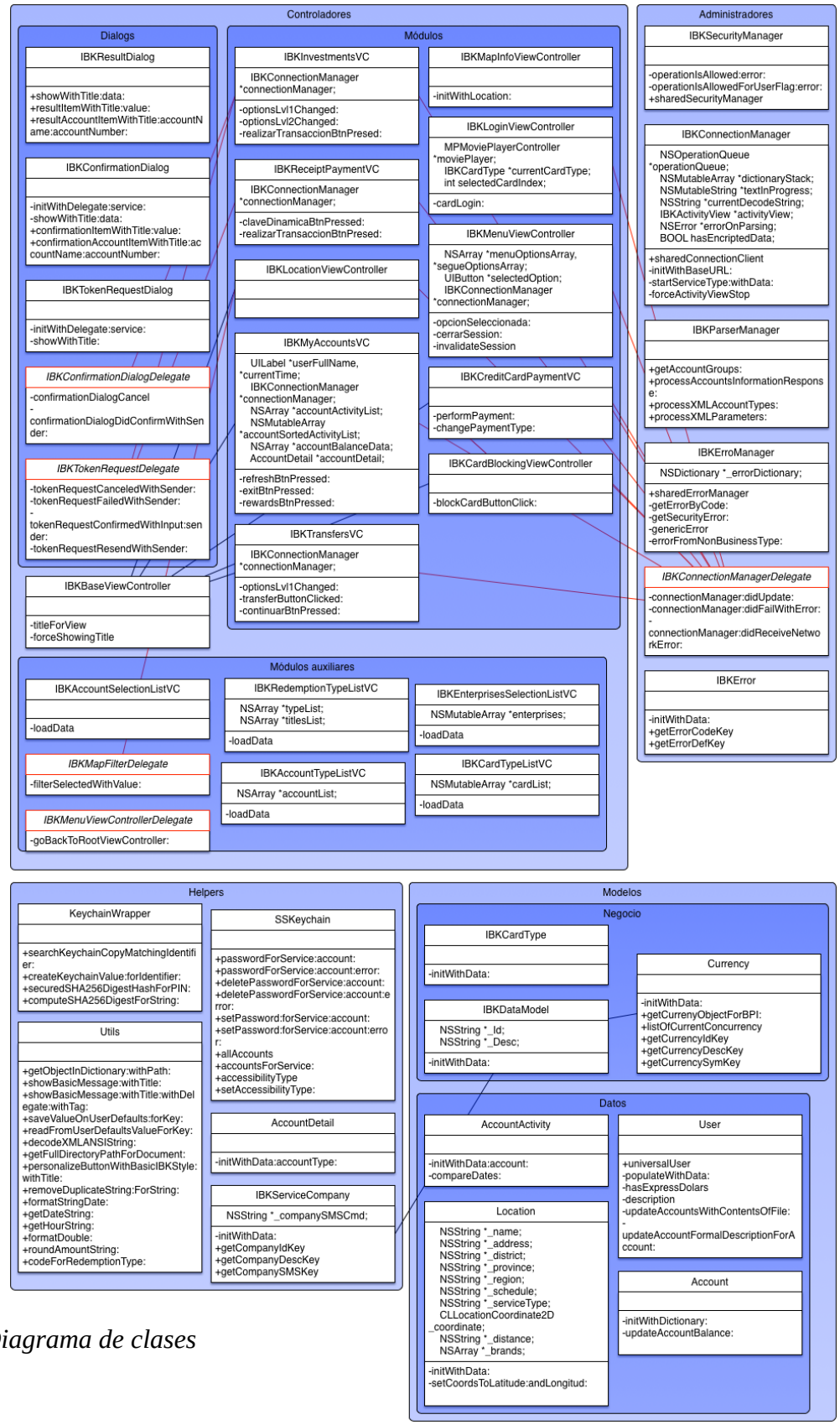

Figura 4 Diagrama de clases

2.3 Interfaces de Usuario

Las interfaces gráficas de usuario siguen un patrón de diseño similar, pues la mayoría son formularios para realizar operaciones bancarias.

Ingreso

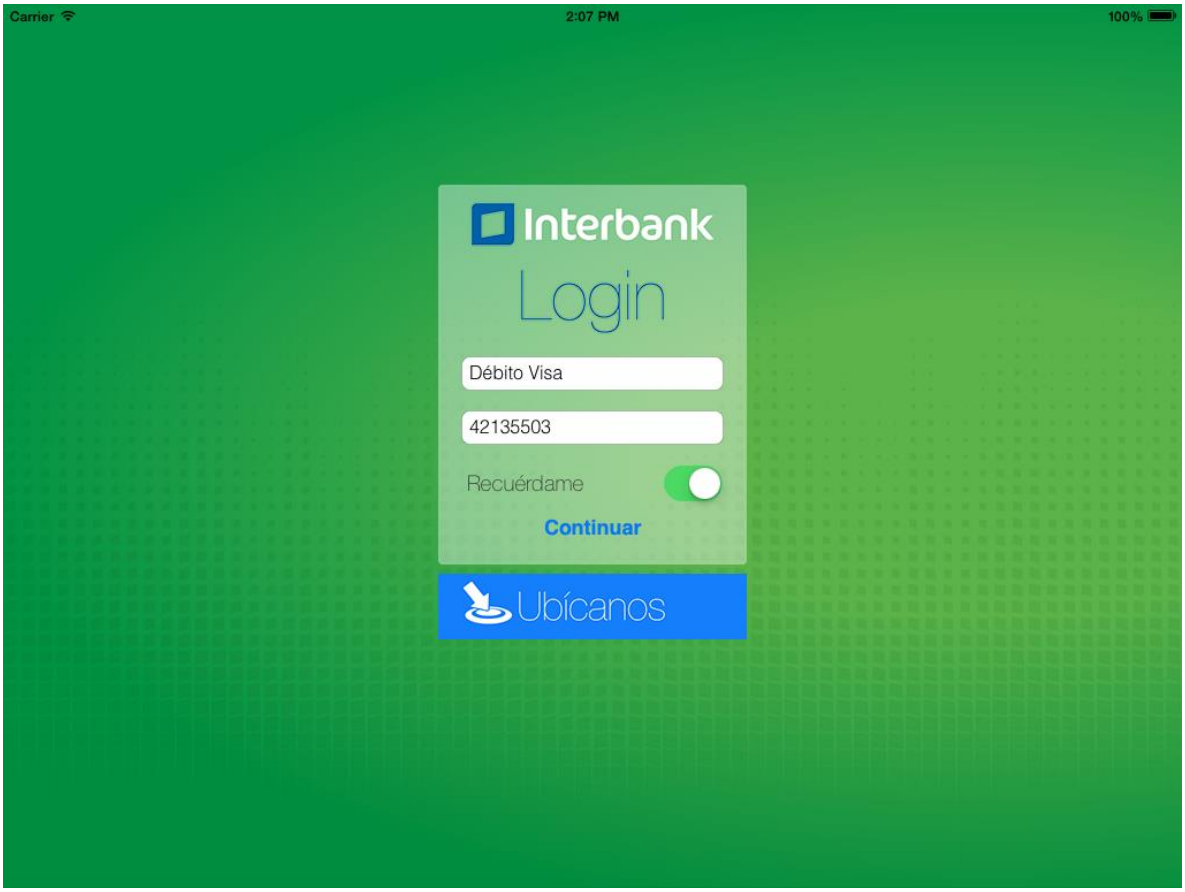

Figura 5 Pantalla de ingreso

Gráficamente siguen un estilo consistente con la marca de la empresa para la cual está diseñada la aplicación. Las pantallas iniciales, es decir, las de ingreso, consisten en espacios casi vacíos con cuadros de diálogo en el centro para ingresar los datos requeridos.

2.3.1 Mis cuentas

Figura 6 Pantalla de listado de cuentas

Luego de ingresar exitosamente a la aplicación se muestra el primer módulo de cuentas. En este se listan las cuentas del usuario y se puede ver el detalle de cualquier con tan solo tocar en ella. Adicionalmente se podrá notar que en el lado izquierdo se ubica el menú de acceso a todos los módulos de la vista principal.

2.3.2 Transferencias

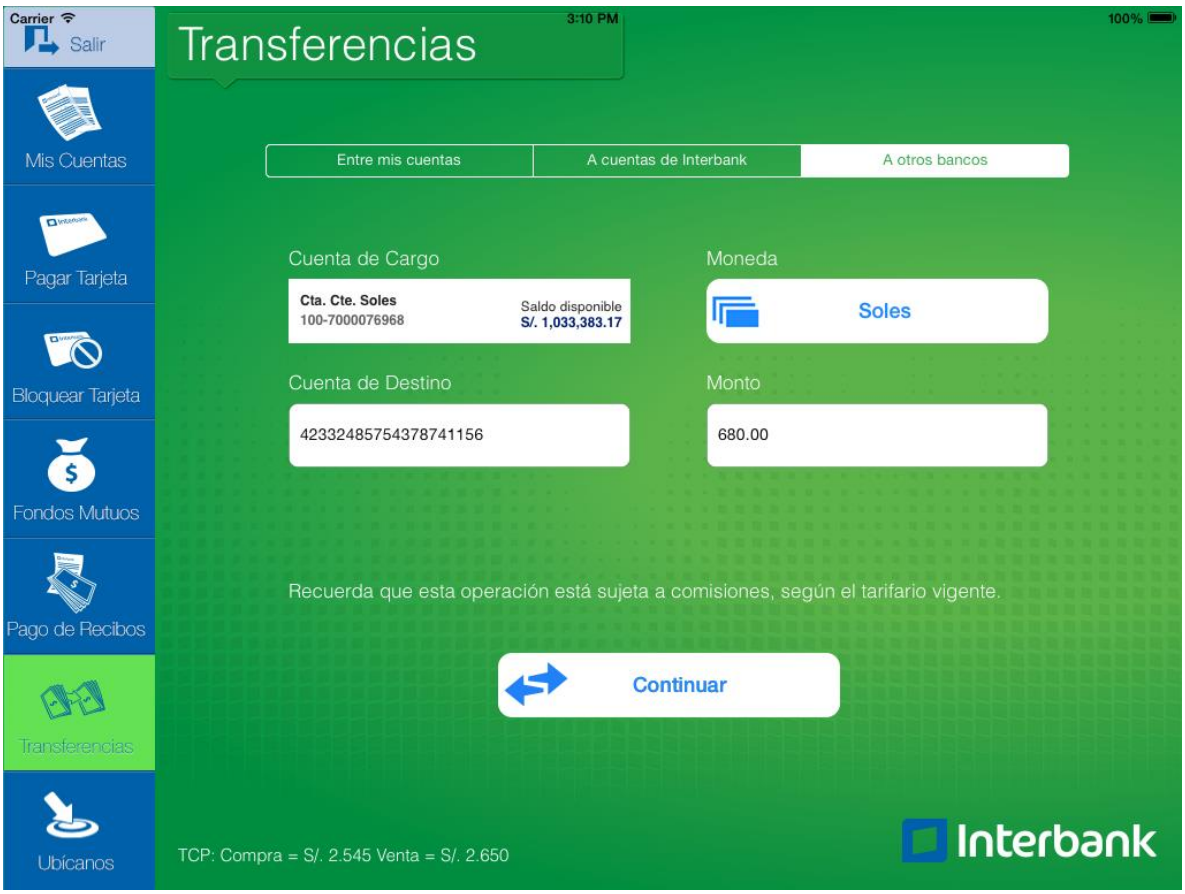

Figura 7 Pantalla de transferencias

La mayoría de los módulos sigue un estilo de formulario tabulado para ingresar los datos requeridos para cada operación. En estos siempre aparece un botón de acción en la parte inferior de la pantalla, el cual generalmente muestra un diálogo de confirmación.

2.3.3 Ubícanos

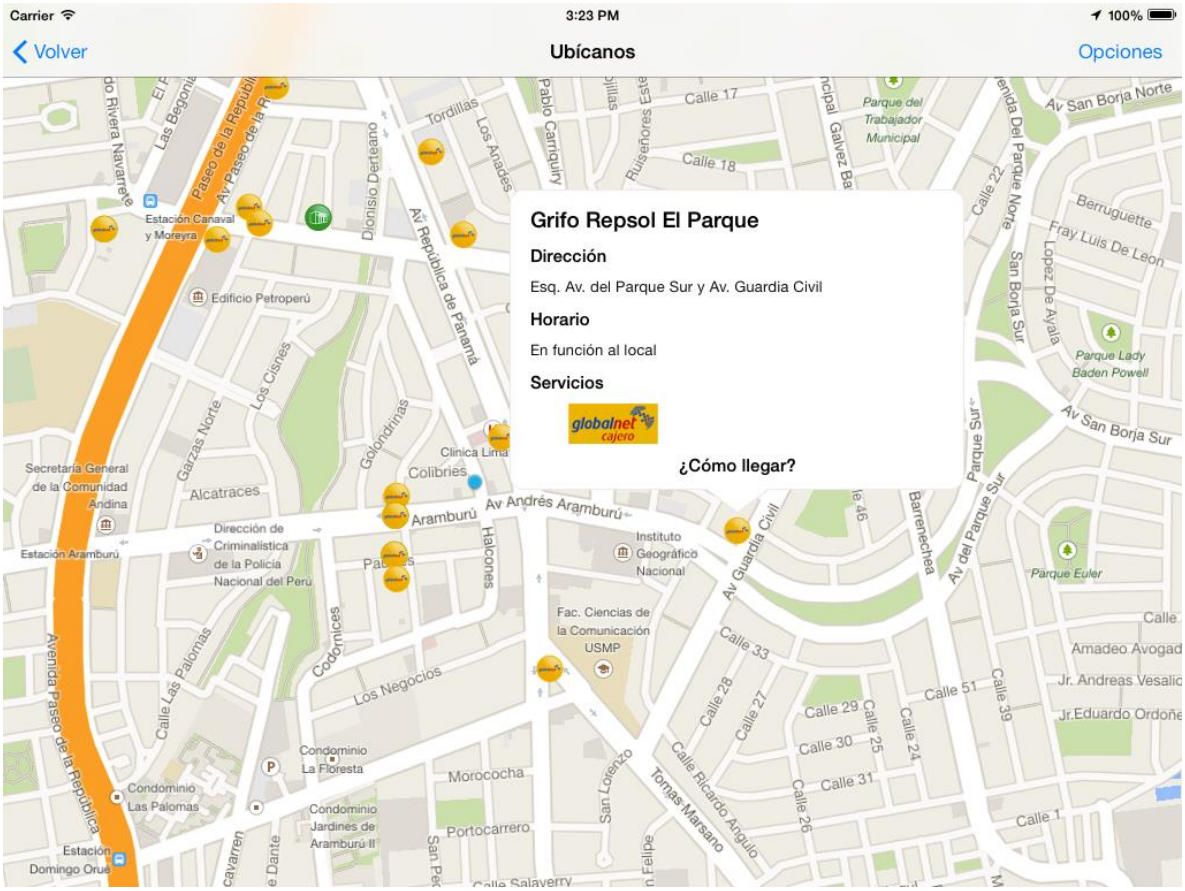

Figura 8 Vista de ubicación geográfica

El módulo de ubícanos (en la imagen accedido desde el botón en la pantalla de ingreso), muestra un mapa ocupando la mayor parte de la pantalla. En este se aprecia la ubicación actual (punto azul) y los marcadores de locales cercanos del banco. Al hacer clic en cualquier punto se mostrará una burbuja de información relacionada con la dirección textual del lugar, el horario de atención y los servicios ofrecidos en ese punto.

2.4 Componentes y Servicios

2.4.1 *Módulos*

Los distintos módulos son los ya mencionados en secciones anteriores de este documento. Para recapitular, los principales son los siguientes:

- **Ingreso:** Módulo encargado de verificar la identidad de un usuario mediante un número de tarjeta, un sello y una contraseña.
- **Mis cuentas:** Listado de cuentas propias de distintos tipos, así como de los detalles y movimientos de cada una de estas, y otros detalles varios.
- **Pagar tarjeta de crédito:** Ofrece un formulario para realizar el pago de un monto a una tarjeta de crédito (la cual puede ser propia, del mismo banco o de otro banco).
- **Bloquear tarjeta:** Permite realizar un bloqueo a una, varias o todas las tarjetas de crédito o débito propias.
- **Fondos mutuos:** En este apartado se pueden realizar inversiones o suscripciones a distintos contratos. También se pueden realizar rescates de cualquier contrato.
- **Pago de recibos:** Desde esta sección es posible realizar la cancelación de distintas deudas que se pudiesen tener en varias empresas afiliadas al banco.
- **Transferencias:** Como su nombre lo indica, desde aquí se puede transferir un monto de una cuenta a otra.
- **Ubícanos:** En este módulo se despliega un mapa centrado en la posición actual del usuario y una serie de marcadores que representan cajeros, oficinas o sucursales del banco cercanas.

2.4.2 *Datos*

Muchos datos de la aplicación, tales como textos y diccionarios de información son guardados aparte en archivos de cadenas de caracteres y listas de datos (extensiones `.strings` y `.plist` respectivamente).

Tener cadenas de textos y valores almacenados de tal forma, representa una mayor facilidad en cualquier modificación de títulos y bloques de texto que se deban mostrar en pantalla. Cualquier cambio se realiza a nivel de estos archivos de datos y se ve su actualización al instante. De esta forma se permite una mayor flexibilidad al realizar modificaciones pero también se organiza con mayor orden la aplicación.

2.4.3 Recursos

Aparte de las clases de código fuente, el sistema está conformado por un paquete que incluye una serie de recursos multimedia como imágenes. Las imágenes utilizadas en el proyecto fueron proporcionadas por un diseñador gráfico específicamente asignado que se encargó de seguir la guía de mock-ups y al mismo tiempo el estilo definido por la empresa cliente (Interbank).

2.4.4 *Frameworks*

La aplicación utiliza varios frameworks, incluyendo a parte de los nativos de iOS los de Google Maps y Testflight, para la distribución y control de binarios y versiones a dispositivos físicos.

Para la sección de Ubícanos, se despliega un mapa. Otra opción para no tener que usar el API de mapas de Google, era utilizar el propio que ofrece Apple. Sin embargo, debido a cualidades en los mapas en el área de Lima y Perú en general, se optó por aprovechar la interfaz de programación de Google y sus mapas. El framework es ofrecido de forma gratuita y libre, y únicamente requiere de tener una llave del API para su uso.

Por otro lado, el otro framework utilizado se integró para facilitar la fase de aseguramiento de la calidad de la aplicación. Testflight permite generar builds de la aplicación para ser entregados directa y fácilmente a varios dispositivos (previamente agregados y con los permisos respectivos). De esta forma se puede generar un nuevo build y entregarlo a un ingeniero de aseguramiento de calidad en cuestión de minutos. El framework también permite obtener un registro de cualquier mensaje o excepción que haya ocurrido durante el uso del aplicativo y dárselo a conocer a los desarrolladores, lo cual facilita el entendimiento de lo que pueda haber salido mal, permitiendo reaccionar más rápidamente y actuar sobre el problema inmediatamente.

2.5 Diseño de Base de Datos

Para esta aplicación, al estar fundamentada en un servicio web, y siendo el caso que se deben de seguir muchas cuestiones de seguridad, no se hace uso de bases de datos locales. Sin embargo, sí se utiliza un método de almacenamiento seguro, conocido como el keychain[2] o llavero de Apple. Este es un componente que ofrece iOS como forma de almacenar datos de forma cifrada dentro del dispositivo, y siempre requiere de una llave privada para ser usado. El keychain ofrece una interfaz de programación un tanto compleja, pero se pueden encontrar algunas librerías compactas que proporcionan una visión más simplificada de dicho componente tal como lo es sskeychain[3]. Esta librería es de código abierto y muestra una interfaz de únicamente cinco métodos para el acceso a contraseñas almacenadas en el sistema.

Por cuestiones de privacidad y confidencialidad del proyecto, no es posible indicar qué valores se almacenan.

---

[2] iCloud Keychain. Consultado el 31 de octubre de 2013 en http://www.apple.com/support/icloud/keychain/

[3] sskeychain, librería para uso del Apple keychain. Consultado el 31 de octubre de 2013 en https://github.com/soffes/sskeychain

3 Conclusiones y comentarios

Los objetivos propuestos se realizaron de forma correcta. Al finalizar el proyecto, la aplicación quedó lista para iniciar la fase de pruebas del cliente, para luego salir a producción y ser puesta en el App Store para que los clientes del Interbank puedan hacer uso de su funcionalidad.

Los productos entregados son los siguientes:

- Binario de la aplicación: Archivo ejecutable de iOS, con los requerimientos planteados y revisados por un proceso de aseguramiento de calidad.
- Manual de usuario.

El aplicativo posee lo que el cliente propuso, sin embargo quedan espacios para mejoras y extensiones de funcionalidad. Algunas de estas ya se estaban considerando al finalizar este proyecto, para ser realizadas en un proyecto próximo únicamente de mejoras. Algunas de las proposiciones de mejoras son las siguientes:

- Recordar localmente el número de tarjeta al ingresar.
- Mostrar el monto a cargar al realizar pago de servicios.

Con respecto a las experiencias adquiridas en el tiempo de práctica se encuentra el fortalecimiento de varias técnicas de programación y diseño estructural bastante prácticas en el lenguaje Objective C, incluyendo el uso de delegados y de protocolos para la comunicación entre objetos.

4 Glosario

Backend: Generalmente se refiere a este concepto para referirse al sistema de software y conjunto de servicios web en un servidor.

Mock-up: Planteamiento de diseño gráfico o de interfaz gráfica.

Keychain: Sistema de administración de contraseñas de Apple.

Build: Aplicación binaria generada por el compilador.

5 Referencias

1. iCloud Keychain. Consultado el 31 de octubre de 2013 en http://www.apple.com/support/icloud/keychain/
2. Concepts in Objective C – C Programming, Delegates and Data Sources. Consultado el 31 de octubre de 2013, en https://developer.apple.com/library/ios/documentation/general/conceptual/CocoaEncyclopedia/DelegatesandDataSources/DelegatesandDataSources.html
3. TestFlight app. Herramienta para la realización de pruebas beta. Consultado el 31 de octubre de 2013, en https://testflightapp.com/
4. Google Maps SDK for iOS. Framework de mapas de Google para la plataforma iOS. Consultado el 31 de octubre de 2013, en https://developers.google.com/maps/documentation/ios/
5. API sskeychain, librería para uso del Apple keychain. Consultado el 31 de octubre de 2013 en https://github.com/soffes/sskeychain