

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

**Desarrollo de una solución nativa y multiplataforma para
la interacción desde una plataforma móvil con una
aplicación bancaria**

Autor: Carlos Sánchez Marín

Director: Fernando Pérez Costoya

MADRID, ENERO DE 2019

% La numeración debe ser romana hasta la primera sección

ÍNDICE

1. Introducción	1
2. Estado del arte	1
2.1. Tecnologías	1
2.1.1. iOS	2
2.1.2. Android	2
2.1.3. Cordova	3
2.2. MOTIF	3
2.3. Modos de conexión con MOTIF	4
2.3.1. Peticiones POST	4
2.3.2. Motif Connector	5
3. Otra sección	7
3.1. Una subsección	7
3.2. Otra subsección	7
4. Resultados y conclusiones	7
4.1. Pruebas	7
4.2. Conclusiones	7
5. Referencias	7

ÍNDICE DE FIGURAS

1. Diagrama de módulos predefinidos de MOTIF 4

ÍNDICE DE TABLAS

Resumen

Aquí el texto del abstract.

Palabras clave: palabra 1, palabra 2, palabra 3...

Abstract

Here goes the abstract text.

Keywords: keyword 1, keyword 2, keyword 3...

1. INTRODUCCIÓN

En los últimos años el sector bancario ha visto como los pagos con tarjetas ha crecido exponencialmente. Según los datos ofrecidos por el Banco de España, desde 2002 hasta 2017 el número de operaciones creció un 245 % y el importe de las operaciones un 188 %, llegando a un gasto total de 135.246,47 millones de € en el 2017[1].

Este rápido incremento de las operaciones con tarjeta también se ha trasladado al entorno corporativo, generando nuevas necesidades para el control por parte de las empresas de las tarjetas que dan a sus empleados. Con el objetivo de cubrir esta necesidad, tanto Mastercard ¹ como Visa ² han implementado nuevas funcionalidades de control de tarjetas que permiten el control directo por parte de las empresas de sus tarjetas, de manera que se puedan recibir alertas de gasto o bloquear determinados tipos de pago.

Este servicio es ofrecido a través de diversas API REST y en Vipera hemos implementado una aplicación que abstrae a los clientes finales (bancos y empresas) de las API proporcionadas por Visa y Mastercard, ofreciendo una API REST centralizada que simplifica las operaciones. Esta API interactúa con un servicio propietario de Vipera que integra toda la funcionalidad requerida por los distintos clientes finales.

Los SDK desarrollados para la integración de nuestro servicio en las aplicaciones móviles de los clientes pretenden abstraer totalmente de las llamadas al servicio de manera que los desarrolladores tan solo vean funciones totalmente descriptivas sin tener que preocuparse de la configuración necesaria para la interacción. Así mismo, ya que la demo técnica realizada por Vipera está implementada con Ionic³, se ha implementado un plugin de Cordova⁴ que habilita la llamada a los SDK de iOS y Android.

2. ESTADO DEL ARTE

2.1. Tecnologías

En el mercado móvil actual iOS y Android lideran con mano de hierro dejando una cuota residual a otros entornos como Windows Phone y otros sistemas menores. En conjunto, iOS y Android acapararon el 99,7 % de las ventas en el primer trimestre de 2018 en España, dejando a Windows fuera de las estadísticas al no lograr el 0,1 % de las ventas totales⁵. Teniendo presentes estos datos, resulta obvio que cualquier tipo de desarrollo fuera de Android e iOS tendría un impacto muy reducido y difícilmente sería rentable para cualquier aplicación que no tenga como objetivo a todos los usuarios del mercado móvil.

¹<https://developer.mastercard.com/product/spend-controls>

²<https://developer.visa.com/capabilities/vctc>

³<https://ionicframework.com/>

⁴<https://cordova.apache.org/>

⁵<https://bit.ly/2Cu8NVy>

2.1.1. iOS

El sistema operativo iOS es un sistema propietario desarrollado por Apple⁶ para sus dispositivos móviles iPhone, iPad y iPod Touch. Fue presentado junto con el primer iPhone en junio de 2007. En un principio Apple no iba a proporcionar SDK para desarrollo de aplicaciones nativas por parte de terceros, pero ante la reacción negativa de todos los desarrolladores, Apple reconsideró su decisión y terminó lanzando la primera versión del SDK en Marzo de 2008 junto a la segunda versión de iOS.

El SDK originalmente proporcionaba un acceso muy limitado a las funciones del sistema operativo, además de tan solo proporcionar soporte para el lenguaje Objective-C⁷. Año tras año Apple ha actualizado su sistema operativo móvil junto con el SDK aprovechando el lanzamiento de los nuevos iPhone.

En 2014, Apple lanzó Swift⁸, un nuevo lenguaje más moderno que sustituyó a Objective-C como lenguaje de referencia para el desarrollo de aplicaciones iOS. No obstante, ambos lenguajes son actualmente soportados por el SDK de Apple, aunque la documentación tan solo es actualizada para Swift.

2.1.2. Android

Android⁹ fue creado en 2003 como un fork del kernel de Linux para dispositivos móviles. Inicialmente fue concebido como un sistema operativo móvil inteligente que estuviese pendiente de la localización del usuario y sus preferencias. En 2005 la empresa propietaria del sistema operativo fue comprada por Google, que se hizo cargo del desarrollo. Dos años más tarde, Google hizo público el primer SDK en fase beta para Android, que fue distribuido a desarrolladores y fabricantes de dispositivos móviles. No fue hasta septiembre de 2008 que se lanzó al mercado el primer dispositivo con Android.

Las primeras versiones del SDK proporcionado por Google contaban tan solo con soporte para el lenguaje Java¹⁰. Al contrario de su principal competidor, el código fuente de Android siempre ha sido open source¹¹ lo que ha atraído a una gran cantidad de desarrolladores y fabricantes. De todos modos, la práctica totalidad de los fabricantes integran en sus terminales la versión de Android propietaria de Google, que incluye integración con las apps de Google bajo la capa Google Play Services¹².

⁶<https://apple.com/es/>

⁷<https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>

⁸<https://swift.org/>

⁹<https://www.android.com>

¹⁰<https://www.java.com>

¹¹<https://source.android.com/>

¹²<https://developers.google.com/android/guides/overview>

En el evento Google I/O de 2017, Google anunció de manera oficial que se añadía un nuevo lenguaje soportado al SDK de Android¹³, Kotlin¹⁴. Este lenguaje desarrollado por la empresa JetBrains¹⁵ funciona sobre la Java Virtual Machine, por lo que es totalmente compatible con el código Java y puede coexistir con este en una misma aplicación.

2.1.3. Cordova

En el año 2011, Adobe Systems¹⁶ compró la empresa Nitobi y renombró su producto como Phonegap¹⁷. Así mismo, lanzó una versión open source del producto nombrada Cordova que es mantenida por la Apache Software Foundation¹⁸. Este framework permite el desarrollo de aplicaciones híbridas basadas en la web, embebiendo una aplicación web dentro de una aplicación nativa para el sistema operativo deseado. Esta solución se acerca a la idea que tenía Apple de aplicaciones móviles en un principio, pero va más allá otorgando un sistema de plugins nativos que permiten la interacción con el sistema operativo.

Los plugins proporcionan una implementación en código nativo (Objective-C para iOS y Java para Android) que será invocada por un wrapper escrito en Javascript¹⁹. En tiempo de ejecución Cordova es capaz de saber sobre qué plataforma está ejecutándose e invocar al código nativo de esta, con lo que abstrae a los desarrolladores de aplicaciones de esta gestión y facilita la escritura de código.

2.2. MOTIF

El despliegue del lado del servidor de la aplicación bancaria con la que se ha operado para la realización de este trabajo está realizado sobre una plataforma propietaria propiedad de Vipera²⁰ cuyo nombre comercial es MOTIF²¹.

MOTIF es un servicio desarrollado en Java haciendo uso del framework OSGi²², creado en 1999 como una primera aproximación al mundo de los microservicios en Java. En concreto, el desarrollo de MOTIF ha sido llevado a cabo empleando Apache Felix²³ como implementación del framework OSGi. El uso de este framework facilita la integración con los clientes finales, por lo general banco y otros servicios financieros, ya que estos suelen emplear Java en sus propios servicios y la arquitectura OSGi permite una integración sencilla de diferentes módulos a los que se llama *plugins* con el resto del sistema. En la figura 1 se representan los servicios básicos implementados en el core de MOTIF.

¹³<https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/>

¹⁴<https://kotlinlang.org/>

¹⁵<https://www.jetbrains.com/>

¹⁶<https://www.adobe.com/>

¹⁷<https://phonegap.com/>

¹⁸<https://www.apache.org/>

¹⁹<https://developer.mozilla.org/bm/docs/Web/JavaScript>

²⁰<http://www.vipera.com/>

²¹<http://www.vipera.com/motif>

²²<https://www.osgi.org/>

²³<https://felix.apache.org/>

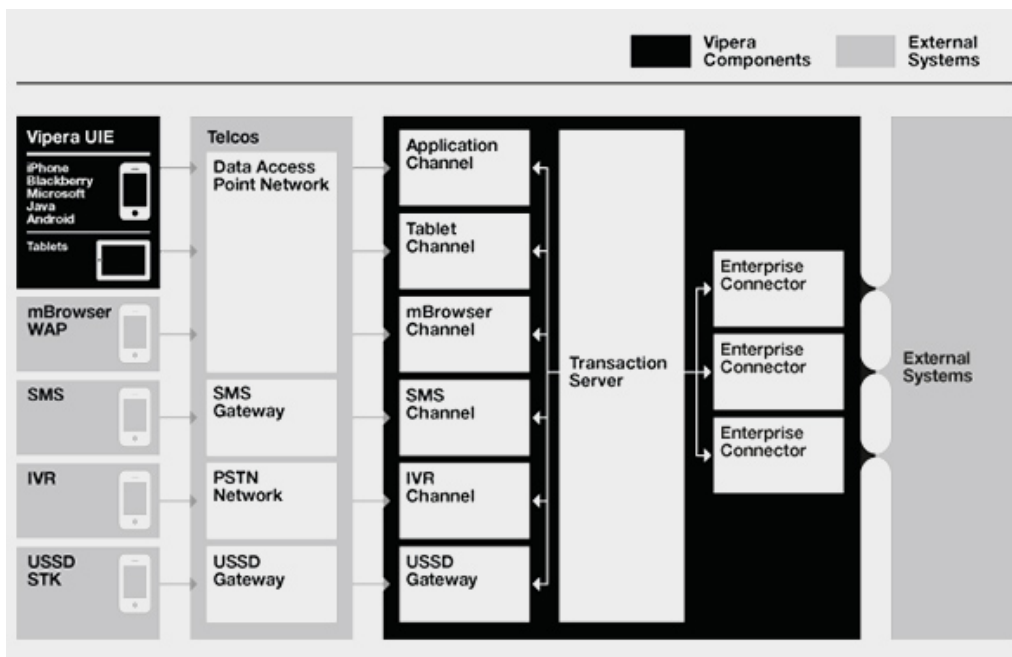


Figura 1: Diagrama de módulos predefinidos de MOTIF

En el caso concreto de la aplicación para la que se ha implementado la solución desarrollada como trabajo fin de grado, se ha hecho uso de los plugins que conforman el core de MOTIF, además de distintos plugins que permiten la interacción con los servicios de Mastercard y VISA, así como plugins de persistencia de datos empleando la librería Hibernate²⁴ y otros que implementan toda la lógica de negocio, que será explicada más adelante.

Para realizar la conexión con MOTIF, por defecto existen 2 métodos distintos:

- **API REST:** Está pensada para dar servicio a implementaciones de administración del sistema como podría ser un Help Desk.
- **JSON Protocol:** Este método de conexión es el que se emplea en las soluciones móviles. Se basa en peticiones POST a un endpoint estático, el cual procesa la información enviada en la petición y determina el comportamiento que debe seguir el servidor.

2.3. Modos de conexión con MOTIF desde una aplicación móvil

2.3.1. Peticiones POST

La forma más básica de hacer llamadas al endpoint JSON de MOTIF es la construcción de peticiones POST haciendo uso de aplicaciones como Postman²⁵ o empleando

²⁴<https://hibernate.org/>

²⁵<https://www.getpostman.com/>

directamente las librerías HTTP de los distintos lenguajes. El siguiente JSON es un ejemplo del cuerpo de una llamada a MOTIF.

```
1 {
2   "req" : {
3     "dom" : "{{domain}}",
4     "app" : "{{application}}",
5     "srv" : "{{service}}",
6     "op"  : "{{operation}}",
7     "header" :
8     {
9       "user": "user1@company1.com",
10      "otp": "123456",
11      "private:pwd": "1111",
12      "private:vpwd": "1111",
13      "email": "user1@company1.com"
14    }
15  }
16 }
```

Fragmento de código 1: Cuerpo de una petición POST a MOTIF

Como se puede apreciar, todos los campos de la llamada están encapsulados dentro del parámetro *req*, el cual es interceptado por el endpoint JSON de MOTIF. Una vez se ha interceptado la llamada, MOTIF comprueba los campos *dom* (dominio al que se está llamando), *app* (aplicación del dominio), *srv* (servicio de la aplicación) y *op* (operación del servicio).

Por último, en el campo *header* se encontrarán contenidos los parametros requeridos por la operación. En este caso se trata de una llamada de ejemplo para la activación de usuarios, por lo que se pasan como parámetros el usuario, OTP (One Time Password), contraseña y su verificación y el email del usuario. Cabe destacar que los campos ligados a la contraseña tienen el prefijo *private*, el cual al ser detectado por MOTIF le indica que no debe quedar constancia en los logs de esos campos.

2.3.2. Motif Connector

Uno de los equipos de desarrollo de Italia implementó una librería para abstraer de ciertos detalles a los desarrolladores de las aplicaciones móviles que emplean como backend MOTIF. Esta librería fue implementada tanto en Objective-C para iOS como en Java para Android. También se implementó un plugin de Cordova para facilitar su uso en aplicaciones híbridas.

En el siguiente código podemos observar como se compone la misma llamada a

MOTIF expuesta en el apartado anterior empleando el plugin para Cordova. Podemos observar el campo *header* sigue existiendo y siendo definido como un JSON, pero el usuario queda abstraído del formato del JSON que configura el dominio, aplicación, servicio y operación.

Por último, se puede observar en la composición de la llamada al servidor que hay un campo booleano marcado como *false*. Este campo determina si la llamada es segura, lo que implicaría la necesidad de que en la llamada incluya un ID de sesión para el usuario que la hace. El plugin implementa una función que guarda en la instancia este ID una vez el usuario se autentica con el servido. En este caso la llamada es de activación de un usuario, por lo que no se puede tener un ID de sesión.

```
1  let myHeader = {
2      "user": "user1@company1.com",
3      "otp": "123456",
4      "private:pwd": "1111",
5      "private:vpwd": "1111",
6      "email": "user1@company1.com"
7  };
8
9  let request: MotifRequest =
    ↪  this.motifClient.buildServerRequest("register",
    ↪  "activateUser", myHeader, false);
10
11  this.motifClient.sendRequest(request).then((res:
    ↪  MotifResponse) => {
12      /*Codigo a ejecutar en caso de éxito*/
13  }, (err) => {
14      /*Codigo a ejecutar en caso de error*/
15  });
```

Fragmento de código 2: Ejemplo de una llamada empleando el plugin Motif Connector

Esta librería es la que se emplea como base para todas las soluciones personalizadas ofrecidas por Vipera, incluyendo la solución expuesta en este documento.

3. OTRA SECCIÓN

3.1. Una subsección

3.2. Otra subsección

4. RESULTADOS Y CONCLUSIONES

4.1. Pruebas

4.2. Conclusiones

5. REFERENCIAS

- [1] B. de España. (2018). Estadísticas Tarjetas para la Web 2T-2018, dirección: <https://www.bde.es/f/webbde/SPA/sispago/ficheros/es/estadisticas.pdf> (visitado 05-11-2018).