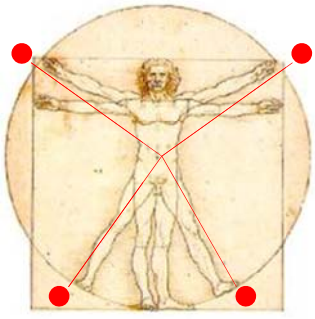
TECNOLOGÍ@ y *DESARROLLO*

Revista de Ciencia, Tecnología y Medio Ambiente

VOLUMEN XIV. AÑO 2016

SEPARATA



DATES: RED SOCIAL DE CALENDARIOS

Daniel Gómez Pascual, Antonio J. Reinoso

UNIVERSIDAD ALFONSO X EL SABIO
Escuela Politécnica Superior
Villanueva de la Cañada (Madrid)

Junio, 2016.
http://www.uax.es/publicacion/dates-red-social-de-calendarios.pdf

Escuela Politécnica Superior.
Universidad Alfonso X el Sabio.
28691, Villanueva de la Cañada (Madrid).
ISSN: 1696-8085
Editor: Javier Morales Pérez – tecnologia@uax.es

DATES: RED SOCIAL DE CALENDARIOS

Daniel Gómez Pascual[a], Antonio J. Reinoso[b]

[a)] Máster en Ingeniería de Desarrollo para Dispositivos Móviles, Graduado en Ingeniería de Sistemas de Información
Universidad Alfonso X el Sabio.
Avda. De la Universidad nº1, Villanueva de la Cañada, 28691, Madrid. España.
dgomepas@myuax.com

[b)]Doctor Ingeniero en Informática
Adjunto a la Jefatura de Estudios
Departamento de Ingenierías TIC, Escuela Politécnica Superior, Universidad Alfonso X el Sabio.
Avda. De la Universidad nº1, Villanueva de la Cañada, 28691, Madrid. España.
areinpei@myuax.com

Resumen

En la actualidad los dispositivos móviles están abarcando cada vez más áreas en el desarrollo de sistemas y en la gestión de la información, creando nuevas oportunidades en el desarrollo del software. Un desarrollo móvil conlleva una nueva forma de gestionar la información, donde la disponibilidad y eficiencia son los principales elementos para determinar el éxito o el fracaso de una aplicación.

En este documento se pretende realizar un estudio sobre una aplicación informática que funcione en dispositivos móviles bajo sistema operativo *Android*, esta aplicación consistirá en un sistema de gestión de calendarios, eventos y citas, donde los usuarios podrán compartir información entre ellos y estar actualizados de las diferentes modificaciones que se puedan producir.

La aplicación desarrollada, estudiará el comportamiento del usuario que trabaja con la aplicación y a partir de la información en la que este se encuentre interesado, sus búsquedas y sus intereses, el sistema presentará la información que más se aproxime a sus preferencias.

En este documento se va a realizar en un primer momento un análisis de la situación, para obtener una especificación de los requisitos que debe tener nuestro sistema, una vez obtenidos estos requisitos, se diseñará la arquitectura del sistema, para posteriormente, y tras los análisis pertinentes, realizar la generación del código fuente, seguido de sus pruebas y de su implantación en un entorno real, para así poder obtener unos resultados acerca del uso de la aplicación.

PALABRAS CLAVE: Android, Gestión calendarios, Programación dinámica, Java Reflection

1. Introducción

La aplicación que se va a desarrollar consiste en una gestión de eventos, calendarios y citas, donde los usuarios podrán crear información y compartirla con los usuarios que ellos deseen, funcionando de forma similar a una de red social.

El uso de esta aplicación pretender ser un sistema útil para agendar exámenes, eventos, notas o citas de interés, en definitiva, la creación de un lugar donde poder encontrar las citas y los eventos de un determinado lugar.

En este documento se va a presentar el desarrollo de una aplicación sobre dispositivos móviles, en concreto sobre el sistema Android y cuáles son las partes más interesantes que se han realizado, no pretende ser una explicación detallada del desarrollo, sino más bien, un pequeño resumen de las posibilidades que nos podemos encontrar con estos "nuevos" tipos de aplicaciones.

Se va a realizar un estudio de alternativas y su valoración, posteriormente, se comprobará si merece la pena realizar el proceso, y una vez analizado todo, se planteará el desarrollo, así como las partes más importantes que se deben realizar, para finalizar con unas conclusiones y con una valoración final del trabajo realizado.

La aplicación diseñada consiste en el desarrollo de una red social donde los usuarios tienen la posibilidad de compartir eventos y citas entre los diferentes usuarios que se encuentran dentro del sistema.

Este tipo de aplicación, sería muy útil para Universidades, Ayuntamientos o centros con una gran cantidad de eventos, donde las fechas son algo muy importantes, pudiendo disponer de una plataforma donde los usuarios puedan encontrar todas las citas que se pueden producir de una forma clara y muy sencilla, pero sobre todo siendo una plataforma accesible.

Para poder realizar esto la aplicación implementará mecanismos de sincronización, permitiendo a los usuarios trabajar tanto de forma online como offline, un sistema de compras para mejorar el posicionamiento de sus resultados a la hora de ser presentados a los diferentes usuarios del sistema o un sistema de notificaciones PUSH, para hacer llegar la información al usuario en el momento en que se produzca una modificación, aunque este no la solicite, evitando así muchos de los problemas que se pueden producir cuando se produce un cambio de fecha, o una modificación, y el usuario no se entera.

2. Estado del arte

Actualmente nos encontramos con un gran número de productos que ofrecen al usuario diferentes de soluciones para implementar su propio gestor de eventos, fechas y notas, el problema, es que cada uno de estos sistemas solamente trabaja con sus propios datos y cuentas, como se puede ver por ejemplo en los sistemas de Google o de Apple, ambas empresas presentan soluciones muy válidas, pero que se encuentran restringidas al uso de sus propios productos, con este sistema se pretende homogeneizar una plataforma común donde usuarios y empresas puedan compartir información sin

necesidad de tener una cuenta específica o algún tipo de restricción implementada por las empresas propietarias de los productos.

Se ha observado que no hay, o por lo menos no se ha encontrado una aplicación con una finalidad similar al desarrollo que estamos planteando, no obstante, vamos a estudiar algunas de las aplicaciones de gestión de tiempo encontradas en el repositorio de Google Play que pueden tener un fin similar a nuestra aplicación:

- aCalendar - Android Calendar: Aplicación desarrollada por Tapir Apps GmBH, los cuales son considerados como desarrollador destacado en la Google Play, esta aplicación, presenta un aspecto muy agradable al usuario, lo cual facilita mucho el uso de esta aplicación, sincroniza los eventos que dispone perfectamente con el calendario que se encuentra en el dispositivo, disponen de aplicación tanto gratuita como de pago, en la aplicación de pago, se pueden obtener características adicionales como diferentes modos de vista, eliminación de banners publicitarios o numerosos temas para personalizar los calendarios que se creen. Esta aplicación se puede obtener desde https://play.google.com/store/apps/details?id=org.withouthat.acalendar
- Today Calendar 2016: Desarrollada por Jack Underwood Productividad, es una aplicación muy completa, tras realizar un estudio detallado se ha observado que en el uso de la aplicación se presentan algunos problemas en cuanto a rendimiento. Trabaja con varios calendarios de carácter propio, pero no puede seguir otros calendarios propuestos por otros usuarios. La aplicación se puede obtener en Google Play en la siguiente dirección: https://play.google.com/store/apps/details?id=com.underwood.calendar_beta
- SolCalendar - Calendar / To do: Desarrollada por Kakao Corp. Productivida, posee un diseño muy atractivo con un diseño minimalista. Ofrece sincronización con otros tipos de calendarios como puede ser el calendario de Hotmail o de Google, también muestra en un apartado todas las tareas que tiene para realizar en el día. La aplicación se puede obtener desde la siguiente dirección: https://play.google.com/store/apps/details?id=net.daum.android.solcalendar
- Month Calendar Widget: Desarrollada por José González D'Amico, esta aplicación es un sencillo widget que muestra el calendario y los eventos que tiene declarados en cada uno de los días, gran aceptación en Google Play. Una aplicación muy sencilla pero con una gran aceptación, se puede obtener desde https://play.google.com/store/apps/details?id=com.josegd.monthcalwidget
- Business Calendar (calendario): Desarrollada por Appgenix, es una aplicación de gestión de eventos claramente enfocada al mundo empresarial, diseño de pantallas un tanto escaso, pero de una gran aceptación entre el sector empresarial. Incluye visualización de días laborales. Se puede obtener desde Google Play en : https://play.google.com/store/apps/details?id=netgenius.bizcal

- WunderList -Lista de tareas: Aplicación desarrollada por Wunderkinder GmbH aplicación de gestión de eventos y tareas de un usuario, gran aceptación entre el público, ofrece versión de pago y gratuita, es considerada como una de las grandes aplicaciones de Android. Posibilidad de adjuntar ficheros a los diferentes eventos que se crean. Ofrece la posibilidad de compartir información entre los usuarios registrados en la aplicación. Se puede obtener en: https://play.google.com/store/apps/details?id=com.wunderkinder.wunderlistandroid

A continuación se presenta una tabla comparativa con las principales aplicaciones analizadas, donde se puede observar los puntos fuertes y débiles de cada una de ellas, en esta tabla, también se incluye la aplicación Dates, sobre la cual se basa este artículo.

	Cuenta de Gmail	**Calendarios Predefinidos**	**Días Festivos del País**	**Tipo de licencia**	**Incorpora publicidad**	**Seguir calendarios de usuarios**	**Trabajo offline**	**Problemas de rendimiento**	**Tareas diarias**	**Sincronización con calendarios externos**	**Presentar Días Laborables**	**Avisos en modificaciones usuarios externos**
Google Calendar	Si	No	Si	Libre	No	Si	Si	No	Si	Si	No	No
aCalendar	Si	No	No	Libre / Comercial	Si	No	Si	No	No	Si	No	No
Today Calendar 2016	No	Si	No	Libre / Comercial	Si	No	Si	Si	No	No	No	No
SolCalendar	No	No	No	Libre / Comercial	Si	No	Si	No	Si	Si	No	No
Month Calendar	No	No	No	Libre / Comercial	Si	No	Si	No	Si	No	No	No
Calendario Menstrual	No	No	No	Libre / Comercial	Si	No	Si	No	No	No	No	No
Business Calendar	No	No	No	Libre / Comercial	Si	No	Si	No	No	No	Si	No
WunderList	No	No	No	Libre / Comercial	Si	Si	Si	No	Si	No	No	Si
Dates	No	Si	Si	Libre	No	Si	Si	No	No	No	Si	Si

Tabla 2-1 Tabla comparativa de las aplicaciones que gestionan tiempos y eventos de Google Play.

En el momento de plantearse una aplicación que conlleva un nuevo desarrollo (coste), se debe realizar un estudio de las diferentes alternativas (aplicaciones) que existen en el mercado, y a partir de aquí tratar de elegir si alguna de las aplicaciones ya existentes se adaptan a las necesidades que queremos cubrir, o si, por el contrario, es más conveniente desarrollar una aplicación que trate de cubrir los aspectos que estamos buscando.

Para facilitar esta tarea, se han analizado las anteriores aplicaciones planteadas, tratando de encontrar cuales son los puntos fuertes y débiles de cada una de ellas, como se puede ver en la tabla superior, una vez estudiadas, a partir de los diferentes criterios que se plantean a continuación seremos capaces de decidir si merece la pena la realización de un nuevo desarrollo.

2.1.1. Criterios de Negocio

Estos criterios pretenden establecer los límites del alcance de lo que queremos realizar, es decir tratan de solventar aquellas preguntas para tratar de crear una necesidad en el propio cliente, por tanto, podríamos decir que estos criterios deberían responder a las preguntas de ¿Para que necesito esta aplicación? ¿Cuáles son los motivos que me pueden llevar a adquirir este producto? ¿Qué inversión debemos realizar? ¿Cuánto nos puede costar? Una vez que tenemos claro cual es el objetivo de negocio que queremos plantear, debemos examinar los criterios técnicos que vamos a emplear en el desarrollo, como respuesta ante estas preguntas podemos encontrarnos con el objetivo de esta aplicación, una aplicación para que los usuarios del sistema puedan gestionar de una manera eficiente y muy sencilla eventos y citas, o como pueden compartir información entre usuarios de una forma ágil.

2.1.2. Criterios Técnicos

En este apartado se van a estudiar los diferentes aspectos técnicos que hay que tener en cuenta para tratar de establecer las características técnicas que vamos a tener que cubrir: sistema operativo que se va a utilizar, servidores externos, forma de almacenar la información,…

- *Sistemas operativos:* Se deben estudiar cuales son los sistemas operativos que se encuentran actualmente en el mercado y partir del estudio hay que desarrollar la aplicación en aquel que consideremos que tiene una mejor aceptación y que tiene más posibilidades de tener éxito. Si nos centramos en el mercado español, podemos ver que el sistema dominante es Android.
- *Infraestructura necesaria* Para crear la infraestructura de la aplicación, se necesita un servidor web donde almacenar la información generada por los usuarios, aparte de esto, debido a que se va a realizar una aplicación sobre el sistema operativo Android, vamos a necesitar también varios terminales que utilicen este sistema.

- *Funcionalidades y extensibilidad* La aplicación se tiene que desarrollar siempre con el pensamiento de aumentar el número de funcionalidades en el futuro, ya que lo más seguro cuando se publique la aplicación, gracias al feedback de los usuarios aparecerán nuevos requerimientos, por tanto, la aplicación, debe de estar preparada para ser extensible de una forma muy sencilla.
- *Simplicidad de uso***:** Uno de los puntos más importante en el desarrollo y en la aceptación de las aplicaciones móviles. Se debe tomar como máxima, que la aplicación debe ser sencilla de utilizar, si la aplicación no es sencilla no valdrá para nada, independientemente de todas las características que este tenga.
- *Soporte profesional***:** Una vez realizada la aplicación se debe ofrecer un soporte para resolver las diferentes dudas e incidencias que puedan tener los diferentes usuarios de la aplicación. Del mismo modo, una de las acciones que más se están poniendo de auge actualmente es el desarrollo de video tutoriales. Una vez desarrollado el proyecto y la aplicación haya llegado a un punto estable del mismo se podrán realizar una serie de videos accesibles desde plataformas como YouTube.
- *Estabilidad del proyecto* Para mantener una buena estabilidad del proyecto se hace necesario la utilización de gestores de código fuente, un control de versiones, para mantener no solamente una copia, sino una autoría del trabajo que se ha ido realizando. Con el paso del tiempo, a media que vayamos desarrollando la aplicación podremos ir creando diferentes ramas (*branches)* de trabajo, como por ejemplo una rama principal de desarrollo de la aplicación y otra con las diferentes actualizaciones que podemos producir. En este punto se pueden utilizar gestores de código como SVN o Git.
- *Almacenamiento de información***:** Como se ha dicho anteriormente podemos encontrar diferentes tipos donde podemos almacenar la información tanto en el servidor externo, como en el dispositivo móvil, en este punto hay que considerar cuales son las tecnologías más apropiadas para cada uno de los escenarios, por ejemplo, para los dispositivos móviles, parece que la plataforma que más aceptación tiene es SQLite, ya que ofrece una interface de comunicación de alto nivel y los tiempos de respuesta son muy buenos, en cuanto a la posibilidad de almacenar la información en el servidor depende del servidor que se contrate y de las especificaciones que queramos implementar, los más comunes son los servidores de MySQL , Oracle o Postgres. El más económico de todos ellos son los servidores que ofrecen gestión sobre MySQL, el cual también ofrece unos rendimientos muy elevados trabajando junto Apache y PHP.
- *Soporte multilenguaje* Cuando se trata de aumentar mercados es importante que la aplicación pueda estar disponible en diferentes idiomas, lo cual nos permitirá llegar de una forma más sencilla a mercados externos como el anglosajón o el chino, o sin tener que irnos fueras de nuestras fronteras, poder ofrecer nuestra

aplicación en los diferentes idiomas del territorio español: castellano, catalán, gallego o euskera.

2.1.3. Barreras y Limitaciones

La principal barrera que nos podemos encontrar en el desarrollo de esta aplicación es que se va a desarrollar bajo el sistema operativo Android, y no vamos a poder copar todo el ámbito de dispositivos móviles, es decir esta aplicación no estará disponible en otras plataformas como puede ser iPhone o Windows Phone, no obstante, desarrollando la aplicación sobre Android, abarcaremos una gran parte del mercado en España, donde el lenguaje operativo predominante en dispositivo es Android, concretamente un 81%.

Otra de las barreras con las que inicialmente nos podríamos encontrar es que esta aplicación inicialmente necesita una conexión a internet, esto que en España es algo habitual y muy normal, en países en vías de desarrollo no es tan habitual, y puede ser un motivo por el cual nuestra aplicación no pueda llegar a todos los usuarios que quisiéramos.

Después de estudiar las diferentes alternativas que aparecen en el mercado, estudiar las barreras y limitaciones, y ver cuales son las características que las cada aplicación tiene, he considerado que la idea de aplicación planteada no se encuentra dentro de ninguna, por lo que tendríamos un “hueco” donde intentar llegar a nuestros potenciales clientes, por lo que considero viable el desarrollo de la app.

3. Metodología

En cuanto a las tecnologías y a los desarrollos planteados, he considerado que el mejor escenario para el desarrollo es realizar una aplicación sobre el sistema operativo Android, ya que el coste de desarrollo es prácticamente nulo, salvando el coste inicial para establecerse como desarrollador Android.

Otra de las características necesarias para el desarrollo planteado es la utilización de un servidor externo para almacenar la información, donde los usuarios “subirán” la información que posteriormente será compartida por los usuarios, en este punto se tendrá que contratar un servidor externo que disponga de procesamiento de scripts PHP y MySQL, ambas tecnologías no requieren licenciamiento por lo que el coste para conseguir la infraestructura de la aplicación vendría únicamente determinado por el precio que tenga el servidor web.

3.1. Definición de los Requisitos del sistema

El desarrollo de esta aplicación, implementa los siguientes requisitos a cumplimentar:

- Aplicación desarrollada en el dispositivo móvil estará escrita en lenguaje Android.
- La aplicación ha de ser adaptable a diferentes dispositivos con diferentes versiones del sistema operativo y con diferentes tamaños de pantallas.

- Almacenamiento de datos en base de datos SQLite en el dispostivo y MySQL en el servidor.
- Conocimiento completo de cómo se desarrolla el sistema, a nivel de estructuración, arquitectura y manipulación de datos.
- Se establecerá un modelado del sistema basado en una arquitectura en capas, implementando una arquitectura MVC para el desarrollo de la aplicación del dispositivo, como se recomienda desde la plataforma de desarrollo de Google.
- Debido a que se van a almacenar información en un servidor de internet, este debe ser un sistema seguro, tanto en las comunicaciones como en el almacenamiento de la información para ello, el sistema debe cumplir con las directrices OWASP (Internet OWASP, 2015), para de esta manera tratar de evitar los problemas relacionados con la seguridad más comunes que podemos encontrar
- La parte de la gestión del servidor debe estar desarrollada en PHP5, ya que es el tipo de servidor que más se usa en internet.

3.2. Definición de la Arquitectura del Sistema

En el momento de definir la arquitectura sobre la que se va a plantear la aplicación, hay que tener en cuenta que nos vamos a encontrar con dos escenarios, en uno de ellos se trabaja sobre el escenario de los dispositivos móviles, y el otros es el escenario de los servidores de internet donde se producirá el almacenamiento de la información, como es normal, ambos escenarios son diferentes, por lo que la arquitectura que se debe diseñar también lo será.

3.2.1. Arquitectura en el dispositivo móvil

En el dispositivo móvil, el software desarrollado va a estar construido en un sistema en capas similar a la propuesta que se realiza desde Google para el desarrollo de las aplicaciones basadas en Android. Cada una de estos niveles (capas) será independiente del resto, permitiéndonos una sustitución de componentes y/o optimización de cada uno de ellos sin tener que modificar el resto del diseño de la aplicación.

- **Capa de acceso a datos – Nivel Acceso Datos**: Se encuentra en relación directa con la base de datos y con los proveedores de contenido independientemente del lugar de procedencia de la información, ya sea desde la base de datos del dispositivo, como desde la fuente de datos del servidor de internet, la principal función de esta capa es centralizar todas las sentencias de interrogación sobre la base de datos y tratarlas para que se puedan entender con el motor de la base de datos.
- **Cada de reglas de negocio – Nivel Negocio**: Se encuentran los procesos de manejo de información y de manipulación de datos. En este nivel, se encuentran definidas las diferentes clases que se utilizan para la gestión del sistema, así

como clases de carácter general, como, por ejemplo, clases para el tratamiento de imágenes o gestión de conexiones con internet.

- **Capa de presentación de datos – Nivel Usuario**: Interfaces de usuarios (layout), que sirven para presentar la información al usuario. En esta capa también se incluyen los diferentes adapter utilizados para servir de nexo de unión entre los datos y las interfaces desarrolladas.

3.2.2. Arquitectura en el servidor

Arquitectura que se encuentra en el servidor web donde se almacena la información, en el servidor solamente nos encontramos con la capa de procesamiento de datos mediante la ejecución de scripts PHP

- **Capa de acceso a datos**: Proporciona las diferentes APIs para poder realizar llamadas web que devolverán la información obtenida en la base de datos almacenada en el servidor, esta capa pretende simular un servidor REST, donde por medio de llamadas a los diferentes scripts que se encuentran almacenados el sistema proporciona información de un tipo u otro.
  Esta capa da soporte a dos tipos de llamadas:
  - POST: Envíos de información para ser almacenadas en la base de datos.
  - GET: Obtener información proveniente de la base de datos.

El intercambio de información entre la capa de comunicaciones del dispositivo móvil con el servidor y viceversa se realiza utilizando datos en formato **JSON**, JSON es el acrónimo de JavaScript Object Notation, el cual es un formato ligero para el intercambio de datos entre los diferentes componentes del sistema.

3.3. Diseño de clases

Para el diseño de las clases que componen el sistema, se ha empleado una metodología orientada a objetos, debido a que es la forma de trabajar habitual, y más recomendada, que propone el lenguaje utilizado, tanto Android como PHP.

Las clases que componen la aplicación se han dividido en los siguientes paquetes dependiendo de su funcionalidad principal

- **Core**: Clases principales de la aplicación, definición de clases de objetos sobre los que se va a trabajar en la aplicación, estos objetos representan las diferentes estructuras de datos que se van a manejar.
- **Comunicaciones**: Clases encargadas de establecer la comunicación con el servidor de información, estas clases serán las encargadas de realizar las llamadas al servidor web y gestionar el envío y recepción de información.
- **UI**: Clases encargadas de presentar los datos al usuario, estas clases son las que cargan los diferentes *layuots* definidos e implementan las funcionalidades y las acciones para reaccionar ante los eventos generados el usuario al hacer uso de la aplicación.

No obstante, este paquete, se encuentra subdivido en otros paquetes para diferenciar el propósito y la funcionalidad de cada uno de ellos, como pueden ser el paquete registro donde se encuentra toda la interface relacionada con el registro de un usuario en el sistema, o el paquete adapter donde se encuentran las clases que implementan los *adapters* necesarios para rellenar las diferentes listas de la aplicación.

- **BD**: Clases encargadas de controlar la gestión con la base de datos, para almacenar la información en el propio dispositivo. Esta capa tiene su homóloga en el servidor web, lugar donde se almacenará la información.
- **API**: Clases localizadas en el servidor, estas clases recepcionan la información procedente del dispositivo y la almacenan y/o procesan para generar un resultado a la solicitud generada desde el dispositivo móvil.

La arquitectura y el procesamiento de la información en un esquema muy resumido podrían similar a:

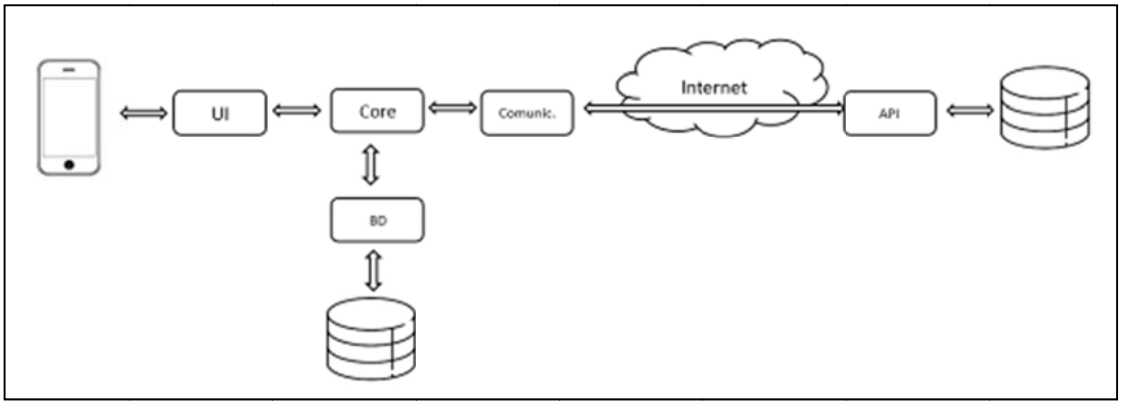


Fig. 3.3-1 Esquema de la arquitectura de la aplicación

Podríamos indicar que el funcionamiento de la aplicación sería similar a: el dispositivo presenta la interface de usuario (UI) al usuario de la aplicación, este al interactuar con la información (CORE) se establece una comunicación bien con el servidor externo por medio de una conexión vía internet (COMUNICACIONES) o bien con la base de datos del dispositivo si no se tiene conexión con internet (BD), una vez que se establece la comunicación los datos son almacenados por la capa correspondiente en la base de datos con la que se haya conectado el sistema.

3.4. Modelo de datos

Antes de mostrar como es el modelo físico de los datos que se utilizan en el sistema, hay que tener en cuenta que el modelo de datos se encuentra duplicado, tanto en el dispositivo móvil como en el servidor web, esto es debido a que la aplicación presenta la funcionalidad de funcionamiento offline, por lo que la aplicación debe de ser capaz

de manejar la información independientemente de donde se encuentre almacenada, es responsabilidad de las clases proveedoras de contenida de obtener la información de la base de datos correspondiente y tratarla para que las clases superiores del sistema puedan manejar sin importar su lugar de procedencia.

En el servidor web la base de datos estará soportada por un motor MySQL, mientras que en el propio dispositivo se utilizará SQLite.

Para mantener la integridad de los datos se realizan una serie de procesos de sincronización para que los datos del servidor sean también como los datos que se encuentran la base de datos del dispositivo, estos procesos se explicaran en los apartados siguientes del documento.

Como se puede comprobar en la propia estructuración de las tablas que se presenta a continuación, la definición es muy sencilla, todas las tablas están compuestas por una clave principal y el resto de atributos que dependen funcionalmente de ellas, las claves ajenas entre las tablas no se encuentran implementadas, ya que así los procesos de copia y restauración de la información se verán simplificados, la implementación de las claves de ajenas en el tratamiento de la información se verá realizado por medio de código programado.

El diseño de cada una de estas tablas se encuentran en 3FN: una tabla está normalizada en esta forma si todas las columnas que no son llave son funcionalmente dependientes por completo de la llave primaria y no hay dependencias transitivas. Comentamos anteriormente que una dependencia transitiva es aquella en la cual existen columnas que no son llave que dependen de otras columnas que tampoco son llave. Como bien se dice en (Hispano, 2015) y en (Universidad de Valencia, 2014).

A continuación, se presentan un par de ejemplos de estructuración de las tablas que conforman la arquitectura de los datos de la aplicación:

Calendarios

Columna	Tipo	Nulo	Predeterminado	Comentarios
id_calendario	int	No		Identificador de calendario. Clave principal
id_calendario_padre	int	Sí	*NULL*	Identificador del calendario desde el que se ha creado el calendario
titulo	varchar	Sí	*NULL*	Título del calendario
imagen	varchar	Sí	*NULL*	Imagen representativa del calendario
descripcion	varchar	No		Descripción
id_usuario	int	No		Usuario que crea el calendario
localizacion	varchar	Sí	*NULL*	Localización. Puede ser creada por el usuario u obtenida a partir de geoposición

Columna	Tipo	Nulo	Predeterminado	Comentarios
latitud	float	Sí	*NULL*	Latitud donde se crea el calendario
longitud	float	Sí	*NULL*	Longitud donde se crea el calendario
caracter	varchar	Sí	*NULL*	Carácter del calendario: P: Publico, S: Solo para el usuario, U: Usuarios seleccionados
variacion	varchar	Sí	*NULL*	Indica si se ha producido alguna variación en el calendario
abreviatura	varchar	Sí	*NULL*	3 Caracteres de abreviatura del calendario
sincronizar	varchar	Sí	N	Indica si se sincroniza el calendario o no
fec_alta	date	Sí	*NULL*	Fecha de creación del registro
fec_modif	date	Sí	*NULL*	Fecha de la última modificación que ha sufrido el registro
usr_crea	varchar	No		Usuario que crea el registro
usr_modif	varchar)	Sí	*NULL*	Último usuario que modifica estos datos
bloqueo	varchar	No	n	Registro bloqueado
marca_baja	varchar	No	n	Marca de registro marcado como eliminado.

Tabla 3.4-1 Estructura de la tabla de calendarios que se usa en la aplicación.

Usuarios

Columna	Tipo	Nulo	Predeterminado	Comentarios
id_usuario	Int	No		Identificador de usuario
nombre	varchar	No		Nombre
apellidos	varchar	Sí	*NULL*	Apellidos
sexo	varchar	Sí	*NULL*	Sexo
numero_telefono	varchar	Sí	*NULL*	Número de teléfono
fecha_nacimiento	date	Sí	*NULL*	Fecha de nacimiento
email	Varchar	No		Email. Utilizado para hacer login en la aplicación
clave	varchar	No		Clave de acceso en la aplicación
tokens	int	No		Tokens de usuario. Pueden comprarse por micro transacciones
imagen	Varchar	Sí	*NULL*	Imagen de usuario
notas	Text	Sí	*NULL*	Notas / Observaciones
fec_alta	Date	Sí	*NULL*	Fecha de creación del registro
fec_modif	Date	Sí	*NULL*	Fecha de la última modificación que ha sufrido el registro

Columna	Tipo	Nulo	Predeterminado	Comentarios
usr_crea	Varchar	No		Usuario que crea el registro
usr_modif	Varchar	Sí	*NULL*	Último usuario que modifica estos datos
bloqueo	Varchar	No	n	Registro bloqueado
marca_baja	varchar	No	n	Marca de registro marcado como eliminado.

Tabla 3.4-2 Estructura de la tabla de usuarios que se usa en la aplicación.

3.5. Definición de Interfaces de Usuario

Uno de los aspectos más importantes a la hora de desarrollar una aplicación, ya sea móvil, de escritorio, web o de cualquier ámbito, es el diseño de la interface de usuario, este aspecto, que en ciertos sectores puede parecer algo secundario, es sin duda uno de los pilares para que una aplicación informática tenga éxito, una aplicación por muy buena que sea, si no tiene un aspecto agradable y sea sencilla de manejar, no conseguirá nunca una buena aceptación y por tanto su aceptación por parte de los usuarios será más difícil, lo cual podrá llevar al fracaso.

Para la definición de la interface de usuario, se han seguido algunas reglas de diseño, que podríamos resumir en las siguientes líneas.

En cuanto a los colores de la aplicación, se ha desarrollado toda la declaración sobre el fichero *style.xml*, de esta forma si en un futuro se quieren cambiar los colores que conforman la aplicación tan solo se deberá modificar este fichero para que los colores de la aplicación cambien.

La aplicación está basada en 3 colores principales: dos tonos azules y un tono naranja para destacar la información sobre la que queremos que el cliente tenga un especial interés, por ejemplo, en una pantalla, el botón de la acción principal estaría sobre un color naranja, como se puede ver en las imágenes que aparecen en este documento

En la aplicación los textos que aparecen y que componen la interface de usuario, tienen la posibilidad de ser traducidos a varios idiomas, para ello, los diferentes strings que se encuentran en la aplicación se encuentran en el fichero *string.xml*, el cual es traducido a los diferentes idiomas en los que queremos que la aplicación este disponible, estos textos se mostrarán en la aplicación dependiendo del idioma que tenga el dispositivo, en las siguientes imágenes se pude ver la misma pantalla en inglés, español y chino.

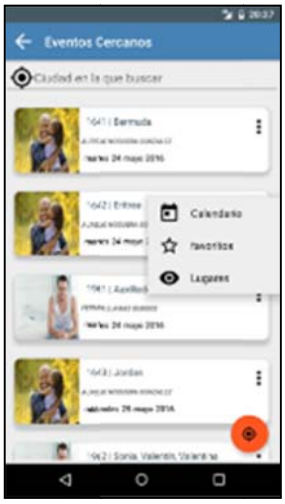


Fig. 3.5 -1 Aplicación en idioma castellano.

Fig. 3.5 -2 Aplicación en idioma chino.

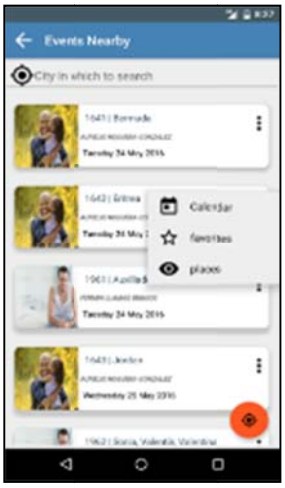


Fig. 3.5 -3 Aplicación en idioma inglés.

Uno de los aspectos más importantes que nos podemos encontrar en el diseño de la aplicación es la forma de presentar la información al usuario, para ello se han utilizado diferentes componentes que ofrece el sistema Android, como puede ser el ListView o el más reciente RecyclerView para la presentación de los datos en forma de lista, de una forma clara.

En los elementos que se presentan sobre recyclerView, se puede ver que sobre los datos se ha incorporado una barra de herramientas a cada uno de ellos, esta barra de herramientas mostrará las diferentes acciones que puede realizar el usuario de una forma muy sencilla y práctica:

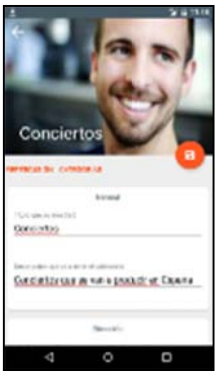


Fig. 3.5-4 Ejemplo 2 de pantalla. Edición de un elemento

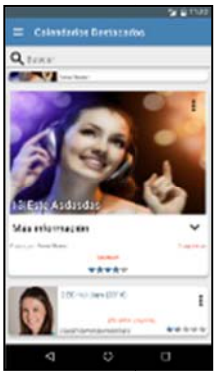


Fig 3.5-5 Pantalla de presentación de información Detalle de un elemento destacado

A la hora de presentar los datos se utilizan el componente Coordinator Layout, este componente presenta inicialmente la pantalla desplegada donde se muestra una imagen predominante en la parte superior de la pantalla, quedando esta última reducida al tamaño necesario para mostrar únicamente el título de la *activity*, si lo que queremos es mostrar la información de la activity.

El diseño de todas las pantallas que se encuentran en la aplicación se encuentra realizado sobre los ficheros *layout* (xml), independizando de esta manera, la capa de programación del sistema de la capa de presentación de la interface.

Gracias al desarrollo en capas, se puede cambiar la interface de la aplicación sin tener que modificar el código fuente del programa, un ejemplo real de lo comentado anteriormente es que en la aplicación cuando se presentan los calendarios en la listas de la aplicación, un calendario puede ser representado utilizando diferentes *layouts*, pero sin tener que modificar el código fuente que se encarga de gestionarlo, el código fuente solamente se encuentra desarrollado una vez.

4. Desarrollo de la solución

En el siguiente apartado del documento se van a comentar aquellas partes de la aplicación que pueden tener un mayor interés para su estudio bien sea por los métodos utilizados o bien por la labor de investigación que estos conllevan, no obstante, si se quiere profundizar en el desarrollo de las soluciones empleadas en esta solución se puede consultar en el Trabajo Fin de Máster “Dates: Red social de Calendarios”, en el cual se ha basado esta publicación.

4.1. Procesos de envío y recepción de Información

Uno de los aspectos a destacar del desarrollo realizado es el proceso de envío y recepción de información, este proceso se ha estandarizado a lo largo de toda la aplicación para ofrecer un mismo tipo de solución a todos los objetos con los que nos podemos encontrar en la app.

Para el estudio de esta tarea, se han utilizado técnicas de ***Reflection*** que ofrece Java y por consecuencia Android, para realizar este trabajo nos hemos basado en información obtenida a partir de (Invarato, 2015) (Internet Arume., 2015),(Developeando.com, 2015), (Codifica.me, 2015) y (Benito, 2004)

La reflection en Java significa programación dinámica, la cual permite llamar a métodos y atributos específicos de una determinada clase en tiempo de ejecución, en vez de en tiempo de desarrollo, permitiendo trabajar con todos los objetos del sistema de una forma general, gracias a esto, a partir de un objeto JSON, podemos crear los diferentes objetos que luego se usan dentro del sistema que serán procesados y tratados en las clases de gestión del sistema

4.1.1. Envío de la Información

Cuando enviamos un objeto, cada uno de sus atributos es parseado en un objeto JSON, este objeto se envía al servidor y el servidor lo reconoce, lo traduce para posteriormente procesarlo y tratar la información recibida.

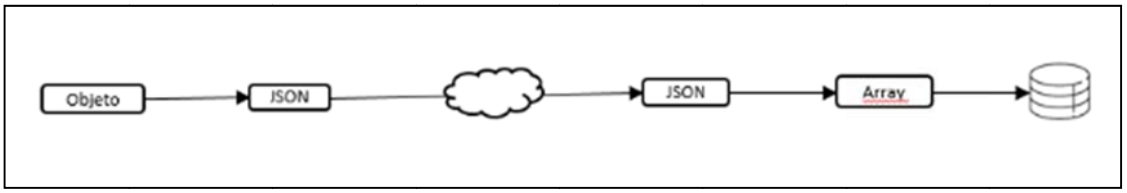


Fig. 4.1.1 -1 Proceso de envío de información desde el dispositivo móvil hacia el servidor web.

El objeto, convertido a string, es enviado al servidor, el cual procesa la información y será el encargado de almacenarlo en la base de datos correspondiente.

4.1.2. Recepción de Información

En el caso de recibir los datos se realiza de una manera prácticamente análoga, con la diferencia de que cuando se solicita una consulta, el servidor devuelve todos los datos contenidos en un array de objetos JSON, cada objeto JSON es parseado construyendo el objeto al cual representa:

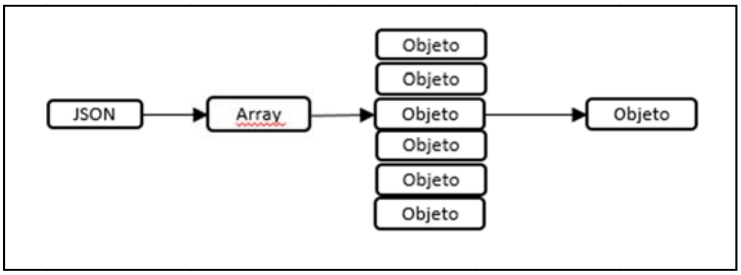


Fig. 4.1.2 -1 Proceso de recepción de información desde el dispositivo móvil.

Una vez que el objeto es recibo este se convierte en un objeto del sistema y pasa a ser gestionado por las funciones y procedimientos de la aplicación, como si fuese un objeto generado internamente en la aplicación.

4.2. Procesos de Sincronización

Otra de las características principales de la aplicación es la capacidad que el sistema dispone para sincronizar la información entre el dispositivo donde se está ejecutando y el servidor web donde se almacena la información de forma habitual.

El proceso de sincronización traspasa los datos del usuario almacenados en el servidor a la base de datos del dispositivo, asegurando de esta manera la posibilidad de seguir trabajando con la aplicación sin necesidad de tener una conexión de internet.

Este proceso está basado en una pareja de marcas de tiempo almacenadas en las diferentes bases de datos (servidor y dispositivo), los eventos o calendarios que hayan sido modificador en este intervalo de tiempo, serán los que se enviaran hacia cualquiera de los lados, consiguiendo tener la información actualizada en cualquiera de las dos bases de datos.

Cuando se comienza a ejecutar el proceso, el sistema obtiene la hora del sistema y la almacena como la hora actual de sincronización, y la marca de tiempo que tenía anteriormente la almacena como la fecha anterior en la que se había conectado:

Conexión		Hora Anterior	Hora Actual
1			13:47
2		13:47	13:52

Tabla. 4.2 -1 Tabla de tiempos de conexión utilizada para los procesos de sincronización en el sistema

Cada uno de los elementos que se van a sincronizar, mantienen una marca de tiempo, que informa el momento en que se ha modificado, por lo que si se modifica un calendario o un evento se modifica la fecha de sincronización y en la siguiente vez que se realice el proceso, los datos serán enviados a la base de datos correspondientes.

La parte más compleja del proceso de sincronización es debida a la posibilidad de crear nuevos elementos desde el dispositivo local, y posteriormente actualizarlos en el servidor web, para ello lo que se realiza es la creación de los nuevos elementos con identificadores con valor negativo, diferenciando así los nuevos elementos creados, con los que no, el motivo de realizar esto es para forzar al servidor a generar el nuevo identificador para el elemento que se está creando y almacenarlo de forma similar a como se están actualizando el resto de elementos.

Tanto en la base de datos del servidor web, como en la del dispositivo, los ids de los elementos serán los mismos en ambos casos, simplificando enormemente el proceso de sincronización entre ambas plataformas.

A continuación, se muestran los procesos esquemáticos de los mecanismos de sincronización:

4.2.1. Modificación de un elemento

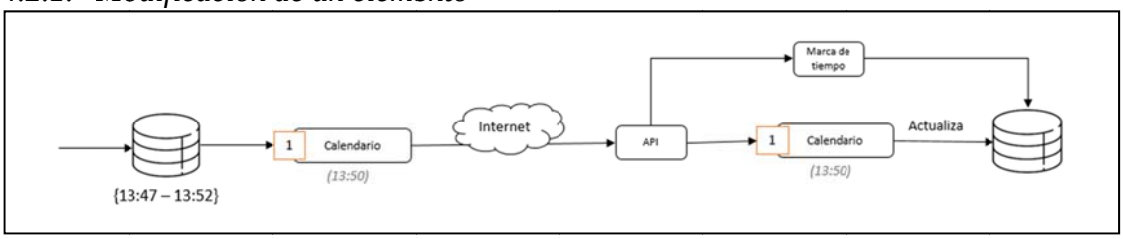


Fig. 4.2.1 -1 Proceso de envío de información para sincronizar un elemento modificado procedente del servidor web en la base de datos del dispositivo.

Secuencia de acción:

1. Se obtienen los elementos que se encuentran en el rango de tiempo generado desde el último proceso de sincronización del usuario
2. Se envía el calendario vía internet al servidor en formato JSON
3. El API del servidor se encarga de desempaquetar el objeto
4. Obtiene el objeto recibido y lo guarda en la base de datos siguiendo los procesos habituales
5. Establece la nueva marca de tiempo de sincronización de elementos

4.2.2. Inserción de un elemento nuevo en el dispositivo, creación en el servidor web

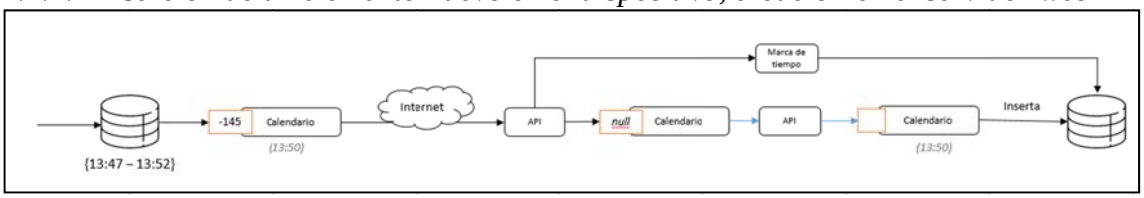


Fig. 4.2.2 -1 Proceso de envío de información para sincronizar un elemento nuevo procedente del servidor web en la base de datos del dispositivo.

Secuencia de acción

1. Se obtienen los elementos que se encuentran en el rango de tiempo generado desde el último proceso de sincronización del usuario
2. Se obtiene el calendario creado nuevo, este tendrá identificador negativo, los datos son enviados en formato JSON
3. El API del servidor se encarga de desempaquetar el objeto
4. El API detecta que el id del elemento es negativo, por lo que sabe que se debe a que es una creación de un elemento nuevo. Establece el id negativo a valor null, para que el API lo detecte como una inserción de un elemento nuevo.
5. Obtiene el objeto recibido y lo guarda en la base de datos siguiendo los procesos habituales
6. Establece la nueva marca de tiempo de sincronización de elementos

4.2.3. Inserción de elementos en el dispositivo local

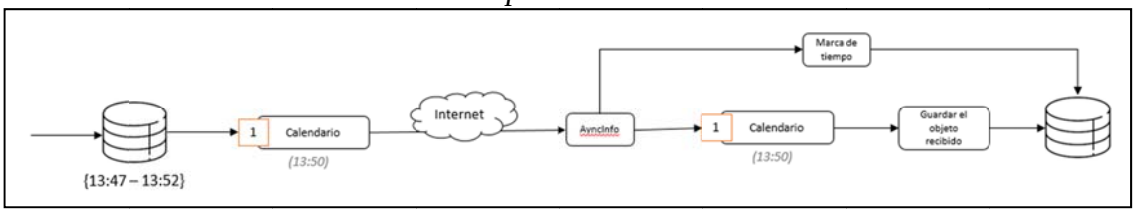


Fig. 4.2.3 -1 Proceso de envío de información para sincronizar un elemento nuevo en el dispositivo local.

Secuencia de acción:

1. Se obtienen los elementos que se encuentran en el rango de tiempo generado desde el último proceso de sincronización del usuario
2. Se envía el calendario vía internet desde el servidor al dispositivo en formato JSON
3. El dispositivo obtiene los valores.
4. Para cada valor que recibe verifica si los datos se encuentran almacenados en la base de datos local, si están almacenados realiza un update, en caso contrario un insert. Esto se puede realizar debido a que los ids de los elementos son los mismos tanto en el servidor web, como en el servidor de la aplicación.
5. Obtiene el objeto recibido y lo guarda en la base de datos siguiendo los procesos habituales
6. Establece la nueva marca de tiempo de sincronización de elementos

4.3. Proceso de Compras

Este es posiblemente uno de los procesos más complejos que se han implementado en el desarrollo de la aplicación, ya que para poder realizar pruebas de este proceso se debe subir la aplicación a los servidores de Google, e indicar los diferentes artículos que se pueden comprar.

Para poder utilizar este proceso y facilitar su desarrollo, Google, ofrece una serie de métodos para poder probar estos procesos por medio de una API de prueba, evitando de esta manera la utilización de datos reales para ejecutar los diferentes procesos de facturación.

En el desarrollo se ha realizado la implementación de compras dentro del dispositivo utilizando las librearías de Android In- App Billing v3 (API Android, 2014).

Cabe destacar que las compras que se realizan dentro de nuestra aplicación pueden realizarse tantas veces como el usuario del sistema quiera, por lo que cada una de estas compras debe de ser consumida en el momento de ser realizada, sino, Google no nos permitirá realizar este proceso varias veces, medida de seguridad implementada desde la consola de desarrollo de Google Inc.

Para poder realizar una implementación de compras mediante In-App Billing en nuestra aplicación debemos cumplir los siguientes requisitos (Internet Androidcode.com. , 2012):

- Las aplicaciones deben de ser publicadas obligatoriamente en Google Play.
- Se necesita una cuenta de Google Wallet de comerciante para usar IAB (clases que se han utilizado para la implementación de las compras en la aplicación)
- IAB requiere la versión 2.3.4 (o superior) de la aplicación de Google Play
- Una aplicación puede usar IAB solo si el dispositivo usa Android 1.6 (API level 4) o superior.
- IAB solo se puede utilizar para vender contenidos digitales.

En el proceso de realizar la compra, es importante indicar que en el momento en que compramos un producto debemos consumirlo, para indicarle a Google que el usuario puede comprar otro producto similar al que acaba de adquirir, si no realizamos este proceso, Google no nos dejara comprar más productos como el que acabamos de adquirir, ya que Google nos indicara que ese proceso ya le tenemos adquirido.

Para poder crear productos "vendibles" en nuestra aplicación se debe subir la aplicación a Google Play (Internet Androidcode.com, 2012), por lo que la aplicación debe en encontrarse firmada (keystore) y no debe estar en modo debug.

4.4. Proceso de Comunicación servidor Google Cloud Messaging (GCM)

Otro de los aspectos que se han incorporado al desarrollo son las notificaciones PUSH, estas notificaciones son envíos de información desde el servidor hacia los diferentes dispositivos, independientemente de que estos soliciten la información, comunicación asíncrona entre el servidor de la aplicación y los clientes.

Una de las posibles soluciones para implementar este tipo de solución es Google Cloud Messaging, según (Arume, 2014) este sistema está constituido por una serie de actores, los cuales intercambian información entre ellos, los actores que componen este sistema son:

- Google Cloud Messaging (GCM): Servicio de Google que se encarga de controlar y gestionar las notificaciones entre el servidor y el propio dispositivo. Este servicio es el que se encarga de enviar las diferentes notificaciones PUSH a los dispositivos que se encuentran conectados al sistema.
- Servidor: Se encuentra entre el servicio GCM y los propios dispositivos, este servidor será el encargado de gestionar la identificación del registro de los dispositivos y las solicitudes creadas, para posteriormente realizar el envío de las notificaicones
- Dispositivo: Donde se recibirán las notificaciones

Si nos centramos en el funcionamiento de esta plataforma podemos indicar que en el momento en el que el servidor de la aplicación quiera lanzar una notificación, consultará

a GCM cuáles son los dispositivos a los que se les quiere enviar la notificación y será el GCM quien se encargue de realizar el envío.

Una vez que tenemos registrado el dispositivo y el servidor ha sido informado de que se quiere obtener información de una determinada aplicación, será decisión del servidor el envío de la información hacia los diferentes clientes disponibles, el servidor se lo comunicara al servicio GCM y este enviará la notificación PUSH a cada uno de los dispositivos que se encuentren registrados sobre esa aplicación.

Estos procesos tienen como objetivo, establecer la comunicación entre el dispositivo móvil y el servidor web, pero en lugar de comenzar la comunicación desde el lado del dispositivo móvil como es habitual, la comunicación es iniciada por medio del servidor y recibida de forma asíncrona y sin ningún tipo de solicitud por parte del dispositivo.

La comunicación entre ambos elementos se realiza mediante comunicaciones PUSH, lo cual aportará una gran cantidad de ventajas adicionales a nuestra aplicación, desde la posibilidad de avisar a los usuarios de las novedades que se presentan en la información en la que esta interesado el usuario, así como la posibilidad en un futuro de hacer llegar información a los usuarios de anuncios de publicidad en función de sus intereses…, en definitiva, un nuevo canal de comunicación con unas posibilidades monetarias muy grandes y atractivas para las grandes empresas.

4.4.1. Implementación en el dispositivo móvil

En el momento en que un usuario se registra en el sistema, se genera un token en función del propio dispositivo desde el que nos estamos conectando, este token, es un identificador del dispositivo móvil único, por lo que el servidor de notificaciones GCM sabrá exactamente a que dispositivo debe de enviar la información que se genere desde la aplicación.

Este token es almacenado en el registro del propio usuario, con lo que queda enlazado directamente el usuario junto con su token correspondiente.

Para poder hacer uso de esta tecnología, debemos solicitar unos permisos especiales para realizar su ejecución, obtenido desde (Rodríguez, 2014):

- com.google.android.c2dm.permission.RECEIVE
- android.permission.INTERNET
- android.permission.GET_ACCOUNTS: Es necesario ya que GCM requiere una cuenta de Google.
- android.permission.WAKE_LOCK:
- <paquete de la aplicación>.permission.C2D_MESSAGE

4.4.2. Implementación en el servidor

Para implementar este mecanismo en el servidor se ha desarrollado un script, donde a partir de los token de registro de los dispositivos se envía la información necesaria a cada uno de ellos.

Por tanto, en el proceso de realizar un envío de información tan solo debemos cruzar la información que tenemos con los usuarios a los que queremos enviarles los mensajes y obtener sus correspondientes tokens.

El proceso de ejecución, se puede resumir esquemáticamente en los siguientes diagramas:

Fig. 4.4.2 - 1 Esquema de comunicación entre el dispositivo móvil y el servidor GCM. Fuente: http://developers.google.com

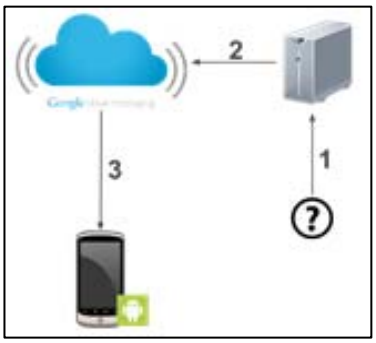


Fig. 4.4.2 - 2 Esquema de comunicación entre el servidor GCM hacia el dispositivo móvil. Fuente: http://developers.google.com

El dispositivo se registra sobre el servidor para poder recibir información, para el proceso de recepción de información, será el servidor el encargado de enviar el mensaje a los dispositivos que se encuentran registrados en el sistema.

5. Implantación, verificación y validación

Para implantar la aplicación y que esta comience a estar operativo, debemos crear un entorno real en el servidor para almacenar la información que vayamos generando, para ello necesitaremos un hosting con PHP y MySQL, donde almacenaremos la información, este servidor, debe tener una capacidad mínima de unos 500 MB, ya que se almacenará tanto la información de la base de datos, los scripts de ejecución en el servidor y las diferentes imágenes que suban los usuarios de la aplicación, como puede ser las imágenes de los eventos o de los calendarios.

Cuando ya esté el hosting preparado, bastará con subir todos los archivos por FTP a la carpeta pública a la que apunte el dominio, e importar la estructura y datos a la base de datos del servidor, para que la aplicación comience a trabajar sobre la dirección real de la aplicación.

En cuanto al dispositivo, para realizar la implantación, bastará con descargar la aplicación de Google Play, o bien utilizar el apk de la propia aplicación para que esta pueda ser ejecutada desde el dispositivo, cuando se ejecute, el sistema automáticamente, solicitará los permisos necesarios, así como la creación de la base de datos SQLite necesaria para su ejecución.

6. Valoración de la solución

En este apartado se presenta una tabla con los resultados de satisfacción que se han obtenido tras seleccionar un grupo de 100 usuarios para probar la aplicación, estos datos han sido obtenidos a partir de una encuesta de satisfacción realizada a los usuarios tras estar utilizando la aplicación desarrollada:

	Pregunta	**Si**	**No**
1	¿Le ha resultado sencillo el manejo de la aplicación?	93	7
2	¿Considera más eficiente este sistema de citas que lo suministrado por las aplicaciones tradicionales?	84	16
3	¿Han encontrado las citas más rápidamente que con los métodos tradicionales?	77	23
4	¿Cuando se ha producido alguna modificación de una cita se ha enterado mejor que con las aplicaciones tradicionales?	98	2
5	¿Ha encontrado cita de interés cercanas a la posición donde se encuentra?	80	20
6	¿Le ha resultado útil la clasificación de calendarios y eventos como favoritos?	54	46
7	¿Las recomendaciones suministradas por la aplicación se adecuan a sus intereses?	70	30
8	¿Ha encontrado fácilmente los calendarios y citas que estaba buscando?	80	20
9	¿Ha creado algún calendario o evento en el sistema?	91	9
10	¿Ha creado algún calendario compartido entre usuarios?	69	31
11	¿Considera útil la posibilidad de compartir calendarios entre diferentes grupos de usuarios?	100	0
12	¿Ha realizado la compra de productos en la aplicación?	2	98
13	¿Utilizaría esta aplicación si tuviese un coste para ser adquirida (o de alguna similar)?	20	80
14	¿Le gustaría poder acceder a los datos desde internet?	100	0
15	¿Le ha resultado útil la aplicación?	91	9
16	¿Aún tiene instalado la aplicación en su sistema?	100	0
17	¿Le recomendaría la utilización de este sistema a sus amigos y conocidos?	84	16

Tabla. 6 -1 Respuestas obtenidas en la encuesta realizada a los usuarios tras la utilización de la aplicación Dates.

Como se puede ver con los datos presentados en la tabla superior, la aplicación ha tenido una buena aceptación entre los usuarios que la han utilizado, considerándola que les facilita las laboras cotidianas para las que está pensada, un 84% de estos usuarios encuestados considera que con esta aplicación es más eficiente la hora de obtener la información de las citas que dispone, del mismo modo indicar, que un 94% de los encuestados consideran que con esta aplicación se enteran de las modificaciones de las citas y/o eventos de una manera mucho más eficiente y segura.

Como contrapunto, comentar el bajo número de compras que se han realizado, así como los pocos usuarios que estarían dispuestos a pagar por la aplicación o por alguna similar, por lo que la monetización de la aplicación se debe centrar en la venta de productos dentro de la propia aplicación (productos *in-app purchase*) e incluso habría que sopesar la posibilidad de incluir publicidad cuando los usuarios consulten la información.

7. Conclusiones y trabajo futuro

Una vez realizado todo el trabajo y tener desarrollada la aplicación, podemos indicar que para la realización del trabajo que aquí se ha descrito se ha realizado una importante labor de investigación, incorporando elementos relativamente nuevos al desarrollo del sistema y utilizando técnicas bastante novedosas para cubrir las necesidades planteadas, como puede ser la utilización de elementos basados en las nuevas guías de Material Design de Google o en la implementación de notificaciones PUSH haciendo uso de Google Cloud Messaging o incluso la implementación de un sistema de compras de productos dentro de la propia aplicación.

Junto el desarrollo realizado, a tratarse de una aplicación cuyo fin es compartir información, se ha debido implementar una estructura de procesamiento de información en el servidor, este sistema implementado en el servidor se encuentra basado en un *framework* escrito en PHP que es capaz realizar un procesamiento de información, este sistema, consiste en un conjunto patrones capaces de gestionar las sentencias de interrogación sobre la bases de datos de forma prácticamente automática, lo cual ha simplificado considerablemente el desarrollo de la aplicación en el lado del servidor web, permitiendo centrar todos los esfuerzos en el desarrollo de la aplicación en el lado del dispositivo y en el sistema Android.

Si miramos al trabajo futuro de la aplicación desarrollada, debemos fijarnos en el desarrollo de la misma sobre el sistema iOS, ya que actualmente la aplicación se encuentra limitada a usuarios que dispongan de dispositivos Android, y aunque el porcentaje de usuarios en España se encuentre cerca del 80%, si todos los usuarios no pueden utilizar esta aplicación, su funcionamiento no llegará a ser pleno ya que no se conseguirá una implementación total y su aceptación se verá reducida considerablemente.

Si nos centramos en las nuevas funcionalidades que se pueden implementar se puede pensar en incluir un servicio de compra de entradas vía PayPal o incluso incluir tecnología NFC para gestionar la validación de entradas a eventos o controles de asistencia, potenciación de los resultados obtenidos en función de la zona geográfica en la que se encuentra el usuario del sistema o en la incorporación de las recomendaciones de las personas que se encuentren más próximas a nosotros en función de las búsquedas e intereses de las personas con las que podemos estar relacionadas.

8. Bibliografía

INTERNET, OWASP (2015*): "OWASP Top 10 Vulnerabilities"*, en OWASP *[*en línea], accesible en https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project

VARIOS (2016)*: "Wikipedia Android"*. en Wikipedia [en línea], accesible en https://es.wikipedia.org/wiki/Android

tutorialspoint.com (2015)*: "SQLite Tutorial. Simply Easy Learning by tutorialspoint.com"* en tutorialspoint.com [en línea], accesible en http://www.tutorialspoint.com/sqlite/sqlite_tutorial.pdf

KABIR, MOHAMMED J. (2015*): "La biblia del Servidor Apache*" en Anaya.

HISPANO, MYSQL. (2014)*: "Normalización de bases de datos"*, en MySQL Hispano *[*en línea], accesible en http://www.eet2mdp.edu.ar/alumnos/MATERIAL/MATERIAL/info/infonorma.pdf

UNIVERSISDAD DE VALENCIA, VARIOS (2014): "Adquisición y tratamiento de datos - Diseño de bases de datos relacional."

INVARATO, Ramón. (2015)*: "Relection en Java"* en jarroba . [en línea], accesible en http://jarroba.com/reflection-en-java/

ARUME - Varios. (2015)*: "Java Relection"*. en Arume *[*en línea], accesible en http://www.arumeinformatica.es/blog/java-reflection-parte-1/

DEVELOPEANDO.COM, Internet – Varios (2015)*: "Reflection Java"*. en Developeando [en línea], accesible en http://www.developeando.com/

CODIFICA.ME. (2015)*: "Reflection en Java"* en OWASP *[*en línea], accesible en http://www.codifica.me/codigo/java/reflexion-en-java/

BENITO, Eneko González. (2004)*: "Introducción al API Reflection (Reflexión) de Java"* en Eneko Gonzáles Benito *[*en línea], accesible en http://static1.1.sqspcdn.com/static/f/923743/14427617/1317485580050/reflection.pdf?token=uHHM5IUpDcX9GiAyzi1puUt3mmw%3D

API, ANDROID (2014)*: "Implementing In-app Billing". 2014*. en Android [en línea], accesible en https://developer.android.com/google/play/billing/billing_integrate.html

INTERNET, Androidcode.es (2012)*: "Monetizando – Introduccion a In-App Billing"*. en Androcode [en línea], accesible en http://androcode.es/2012/05/monetizando-introduccion-a-in-app-billing/

VARIOS, Internet (2014)*: "Configurando Notificaciones Push en Android"* en Adictos al Trabajo [en línea], accesible en https://www.adictosaltrabajo.com/tutoriales/configurando-notificaciones-push-android/

VARIOS, Internet (2014)*: "Android Push Notifications using Google Cloud Messaging GCM - Android Example"* en Android Developers [en línea], accesible en https://developers.google.com/cloud-messaging/

RODRÍGUEZ, Miguel Arlandy. (2014)*: "Configurando Notificaciones Push para desarrollos Android con Google Cloud Messaging"* en Adictos al trabajo [en línea], accesible en https://www.adictosaltrabajo.com/tutoriales/configurando-notificaciones-push-android/

VARIOS Internet (2012) *: "GCM with PHP (Google Cloud Messaging)"* en Android Hive [en línea], accesible en http://www.androidhive.info/2016/02/android-push-notifications-using-gcm-php-mysql-realtime-chat-app-part-1/