UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Título del Proyecto:

MusicaNovApp

Autor:

Casasempere Sanchis, Francisco Javier

Director:

Tomás Gironés, Jesús

TESINA PARA LA OBTENCIÓN DEL TÍTULO DE:

Máster en Desarrollo de Aplicaciones sobre Dispositivos Móviles

Septiembre del 2015

Contenido

Tabla de ilustraciones

Introducción

En este proyecto vamos a desarrollar una aplicación que permita compartir de forma clara y útil toda la información de las actuaciones desarrolladas por la Societat Musical Nova d'Alcoi. El principal objetivo es publicar la información de horarios y actuaciones entre los músicos de la banda y obtener respuestas en cuanto al control de asistencias.

Se usará la plataforma Android para desarrollar la aplicación MusicaNovApp por dos razones principales:

- Gran público objetivo. El 89,4% de smartphones en España en el segundo trimestre de 2015 son Android[1].
- Google API's. Las herramientas que Google provee a los desarrolladores (muchas de ellas gratuitas) facilitan el desarrollo de cualquier aplicación.

Este proyecto se desarrolla como un ejercicio dentro del marco del proyecto final del Máster en Desarrollo de Aplicaciones sobre Dispositivos Móviles de la UPV, pero es bastante posible que se lleve a producción y que posteriormente sufra mejoras y modificaciones para acomodarlo a los usuarios finales.

Descripción del problema

El punto de partida de este proyecto es la necesidad de difundir avisos y noticias sobre la actividad de la Sociedad Musical Nova d'Alcoi. Se trata de una sociedad cuya actividad central es la banda sinfónica compuesta por más de 80 músicos. Los avisos de las actuaciones, ensayos y demás actividades siempre se han realizado colgando carteles en el panel de anuncios de la propia sociedad. Normalmente con esto debería ser suficiente, pero siempre hay quien por razones diversas no ha podido consultar el panel y no se ha presentado a algún acto.

Hace bastantes años, también se contrataron diversas líneas de móvil, por lo que algunos avisos de mayor importancia o en caso de improviso se realizan mediante envíos de SMS. Esto solía ser un problema, ya que según el modelo de teléfono la posibilidad de enviar mensajes masivos o de crear grupos estaba limitada a cierto número de contactos.

Un poco más adelante, se empezó a usar el WhatsApp, esta aplicación estaba bastante bien y cumplía su cometido y más cuando desde Noviembre de 2014 se aumentó el tamaño máximo de los grupos de 50 a 100 contactos[2]. En Mayo de 2015 se creó un grupo con todos los integrantes de la banda.

[1] http://www.kantarworldpanel.com/global/smartphone-os-market-share/

[2] http://www.elgrupoinformatico.com/whatsapp-aumenta-100-numero-participantes-grupo-t21337.html

Con esto ya se contaba con una buena herramienta, se usaba el envío de imágenes para adjuntar una fotografía del calendario de actuaciones y el propio chat para los avisos/notificaciones, etc. Casi la totalidad de los miembros recibían las notificaciones en tiempo y en forma, pero a su vez conlleva otros problemas.

El primero de ellos es que en un grupo con 91 miembros, cualquier aviso o comunicación suele crear cierto debate. Se ha comentado muchas veces que hay que intentar evitarlo, pero aun así sigue pasando. En un momento puedes recibir más de 50 notificaciones de mensajes nuevos en el grupo. Esto causa molestias, consumo de batería y además provoca que muchos silencien las notificaciones del grupo, por lo que pierde la funcionalidad más importante de avisar de cambios o imprevistos con poco tiempo de antelación, por ejemplo, que se suspenda un acto por mal tiempo, o que se adelante una actuación etc.

Otro problema es que se ha detectado que algunos socios tienen problemas para encontrar un mensaje concreto entre tal cantidad de mensajes. Por ejemplo, el mensaje donde se comunicaba la hora de un acto, o encontrar la fotografía con el calendario de ensayos. Claro está, que es sencillo buscar una imagen si exploramos las imágenes desde el propio grupo, pero hay bastantes usuarios que no son capaces o simplemente no conocen esta opción.

Y por último, aunque de momento no es un problema real, el límite de 100 usuarios por grupo al que estamos acercándonos y que con WhatsApp no es posible superar. Otras aplicaciones como Telegrama tienen un límite en 200 miembros por grupo, pero sólo solventaría este último punto.

Otro de los problemas más comunes, es el control de asistencias. Además de comunicar a los socios las distintas actividades programadas, es importante controlar la asistencia a las mismas.

Hasta el momento, esto siempre se ha realizado de forma manual, habiendo un encargado de controlar la asistencia mediante listados en papel. Posteriormente son trasladados de forma manual a una BBDD (Excel) para su tratamiento. Sería interesante poder controlar la asistencia mediante una aplicación móvil, de una forma sencilla.

Objetivos

Ante la problemática descrita en el punto anterior, se plantea como trabajo de fin de master el desarrollo de una herramienta que plantee una solución a estos temas.

Para poder dirigir mejor los esfuerzos por conocer y comprender las características del proyecto, es necesario fijar unos objetivos que abarquen las actividades que se pretender realizar y, además, permitan al final de las mismas conocer el grado de desarrollo y cumplimiento alcanzado

Los objetivos que el proyecto debe conseguir son:

- Conocer las necesidades reales de la sociedad. Conocer bien como son los medios de comunicación que usan y qué es lo realmente necesitan.
- Buscar alternativas en el mercado. Tal vez exista alguna aplicación o utilidad que pueda solventar estas necesidades.
- Proveer de una herramienta de comunicación sencilla y visual que permita visualizar la programación de actos de la sociedad.
- Diferenciar perfiles de usuario y administrador, para que sólo los administradores puedan publicar avisos.
- Ofrecer una herramienta para avisar de forma rápida a los músicos de cualquier imprevisto.
- Proporcionar una herramienta para el control de asistencias a eventos.

Motivación

La motivación que me llevó a elegir este proyecto como trabajo de fin de master, ha sido sobre todo personal. Coincidió en el tiempo la puesta en marcha del grupo de WhatsApp con el momento de elegir proyecto. La cantidad de mensajes y notificaciones que se generaban en el grupo los primeros días, fue una de las razones para empezar a pensar que la mejor opción era tener otra herramienta para comunicar de forma efectiva todo lo que fuese necesario.

Estoy ligado a esta Sociedad desde hace más de 20 años y también he estado en la junta directiva durante algún tiempo. Conozco bastante bien la forma de trabajar de la Sociedad y estoy seguro que una herramienta que facilite el trabajo será bien recibida. Unos meses antes de empezar con el proyecto, estuve comentando con varios compañeros sobre esta herramienta y especificamos algunos de los puntos importantes con los que la aplicación debería de contar.

Por otro lado, con un público inicial de casi 100 usuarios es una buena prueba de fuego, ya que la idea sería que superado un proceso de pruebas, poder publicar la aplicación para que otras sociedades musicales puedan hacer uso de ella.

Estado del arte

En la actualidad la gran mayoría de los usuarios poseen un Smartphone y continuamente lo llevan consigo. He comprobado que casi la totalidad de los usuarios finales disponen de un terminal Android, por lo que la primera versión será en esta tecnología. Es posible que si la aplicación sigue creciendo se desarrolle para iOS.

Tecnologías utilizadas

Brevemente comentar la tecnología usada en la que se apoyará el proyecto. Por un lado, se usará la plataforma de Cloud Messaging de Google para todas las notificaciones vía push. Por otro lado se usará un servidor web como backend para almacenar toda la información de sincronización de los dispositivos móviles.

Google Cloud Messaging

Se trata de un servicio de mensajería que permite enviar información desde nuestro servidor a un dispositivo móvil. No voy a explicar en profundidad como funciona esta tecnología, puesto que ya ha sido tratada durante el transcurso de la formación del Master, pero sí destacaré algunas de sus ventajas, frente a otras alternativas de mensajería como mqtt (Mosquitto).

- Cubre casi la totalidad de dispositivos Android del mercado: Android > = 2.2 y Google Play Store instalado.
- No se establecen nuevas comunicaciones ya que usa la existente de los dispositivos Android con Google.
- No es necesario contar con un servicio en la aplicación constantemente a la espera. Por tanto el consumo de batería no se ve penalizado.

Backend: Apache + PHP/MySQL

Aunque no es un punto estudiado con mucho detenimiento durante el Máster, sí que es una tecnología que hemos usado en diversos proyectos como en Google Glass, por ejemplo. A la hora de realizar este proyecto, tengo una instancia de un servidor (siscocasasempere.es) con PHP + MySQL.

Esta plataforma, guardará toda la información necesaria para poder sincronizarla posteriormente con los dispositivos móviles. La comunicación se realizará a través de Web Services y PHP será el lenguaje del servidor encargado de realizar la comunicación entre la BBDD y los dispositivos móviles.

Arquitectura Cliente-Servidor

Dentro de la computación distribuida, la arquitectura cliente-servidor es un modelo de aplicación en el que las tareas se reparten entre dos actores. Por un lado están los clientes y por otro el servidor.

El servidor es la parte de la aplicación que va a prestar una serie de servicios al cliente. Por tanto se encargará de gestionar y almacenar la información de la aplicación y dará acceso a dicha información de forma parcial o total a los clientes.

El cliente es la parte de la aplicación que interacciona con el usuario. A diferencia del servidor, el cliente no comparte sus recursos con otros clientes.

El cliente es q leu toma la iniciativa de comenzar el diálogo con el servidor enviando peticiones. De cada petición que sea enviada al servidor, el cliente debe recibir una respuesta.

LA elección de este tipo de aplicaciones viene dada por la facilidad y escalabilidad que ofrece a la hora del diseño y desarrollo, ya que tenemos diferenciados el lado servidor y clientes. De esta forma conseguimos tener mayor seguridad en la aplicación manteniendo los datos e información en el servidor.

Arquitectura de la aplicación

Durante esta sección intentaré describir los puntos que componen la aplicación y cómo se relacionan entre ellos. También se justificarán las decisiones tomadas durante la etapa de desarrollo de la aplicación.

La plataforma Android

Durante este curso, hemos visto las posibilidades que Android nos ofrece a la hora de desarrollar una aplicación y todas las bondades que el sistema de Google ofrece a los desarrolladores. También hemos visto algunas de las desventajas que acucian Android, siendo posiblemente una de las más importantes el problema de la fragmentación.

El problema de la fragmentación es debido al gran número de dispositivos y versiones distintas de los dispositivos Android. En otras plataformas como iOS, la fragmentación es mucho menor. Se podría dividir en función del hardware y del software.

En cuanto al hardware la fragmentación más importante, en nuestro caso, se trataría de los distintos tamaños y resoluciones de pantalla. Android permite diseñar la aplicación para visualizar en función del tamaño de pantalla donde se está ejecutando. Otras aplicaciones pueden tener problemas al ser más exigentes en necesidades como sensores, memoria, procesador. El problema radica en que no estamos 100% seguros que una aplicación se comportará de la forma esperada en todos los dispositivos.

En cuanto al software, el problema viene dado por las distintas versiones de Android. Muchas veces porque los propios fabricantes dan por obsoletos ciertos modelos de dispositivos y quedan anclados con versiones antiguas de Android y con posibilidades muy limitadas para ser actualizados. Es por ello que a la hora de desarrollar una aplicación Android hay que tener en cuenta las versiones mínimas que a las que daremos servicio. En esta ocasión se opta como versión mínima el SDK 16 (Android 4.1.X Jelly Bean). Según la última estadística publicada en Android Developers actualmente más del 90% de los dispositivos Android soportan el SDK 16.

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	4.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	3.7%
4.1.x	Jelly Bean	16	12.1%
4.2.x		17	15.2%
4.3		18	4.5%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	15.9%
5.1		22	5.1%

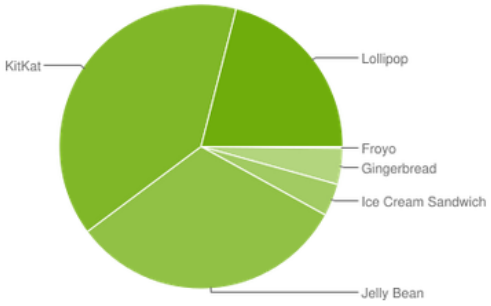


Ilustración 1. Fragmentación versiones Android (Septiembre 2015)

Usaremos el paradigma de diseño “Always-On” (siempre conectado). Hace unos años ciertas aplicaciones se desarrollaban teniendo en cuenta que en ocasiones la conexión a la red podía ser limitada o nula. Esas aplicaciones se diseñaban para contar con una base de datos local que periódicamente se sincronizaba con la nube en las ocasiones que había conectividad. La aplicación trabajaba en modo off-line y cuando necesitaba obtener datos se conectaba a la red para obtenerlos. Esto implicaba ciertos problemas a la hora de compartir información en tiempo real o del acceso a información por varios dispositivos al mismo tiempo.

Actualmente esto ya no tiene sentido, para la mayoría de aplicaciones, ya que se entiende que la gran mayoría de dispositivos cuenta con acceso total a la red en cualquier momento. Por esto, la base de datos que se manejará en local será mínima y la información será obtenida en tiempo real desde los servicios del croad.

Esquema del diseño

En el siguiente esquema veremos las partes más importantes que componen el proyecto, posteriormente explicaremos cada apartado de formas más detallada.

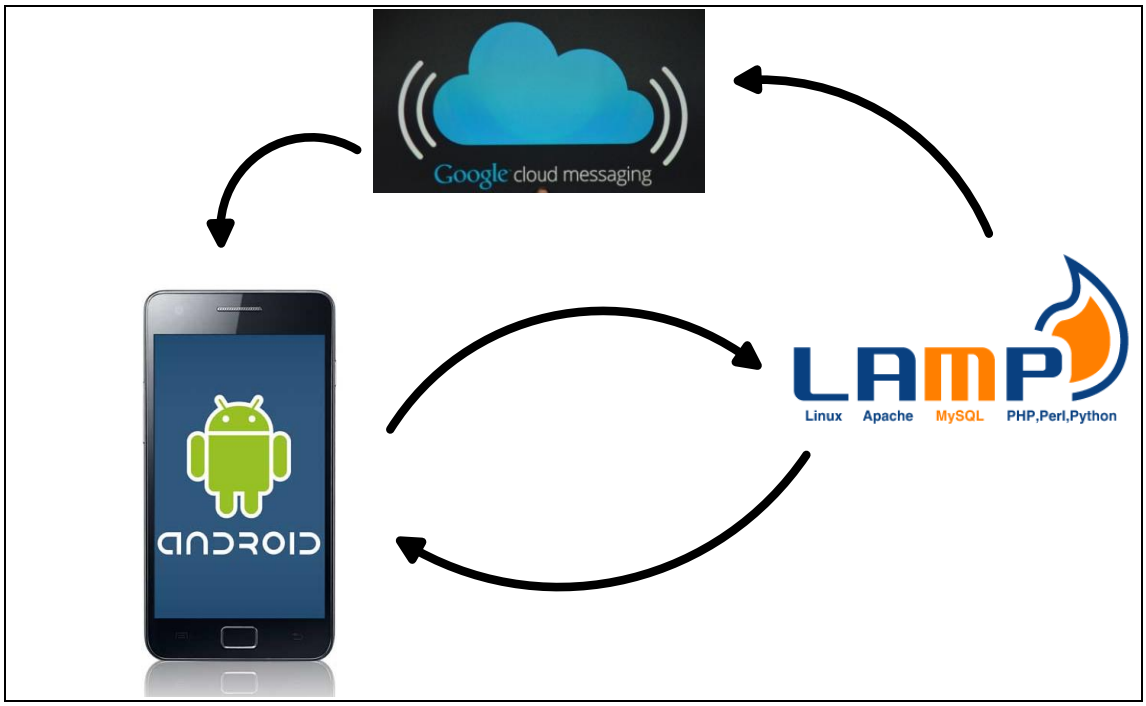


Ilustración 2. Esquema funcional del proyecto.

Google Cloud Messaging (GCM)

Como hemos comentando antes GCM, es una plataforma para la entrega de notificaciones de forma síncrona desde Internet hacia dispositivos Android principalmente (aunque otros también lo soportan), conocidas como notificaciones push.

Nuestra aplicación, necesitará contar con la capacidad de notificar a los usuarios, algunos eventos ocurridos. Para conseguir esto existen diversas estrategias como el polling, donde cada cierto tiempo el cliente busca en el servidor nuevas notificaciones. El polling implica un uso excesivo de recursos y la pérdida de la interacción en tiempo real

A partir de la versión de Android 2.2, Google incorporó la posibilidad de usar notificaciones push. Se trata de un mecanismo que comunica nuestro dispositivo con los servidores de Google y que ofrece un servicio para que nuestro propio servidor pueda explotar esta plataforma. Para ello, será necesario dar de alta el proyecto en Google Developers Console y activar el API de GCM para obtener las claves con las que tanto la aplicación como nuestro servidor se identificarán con Google.

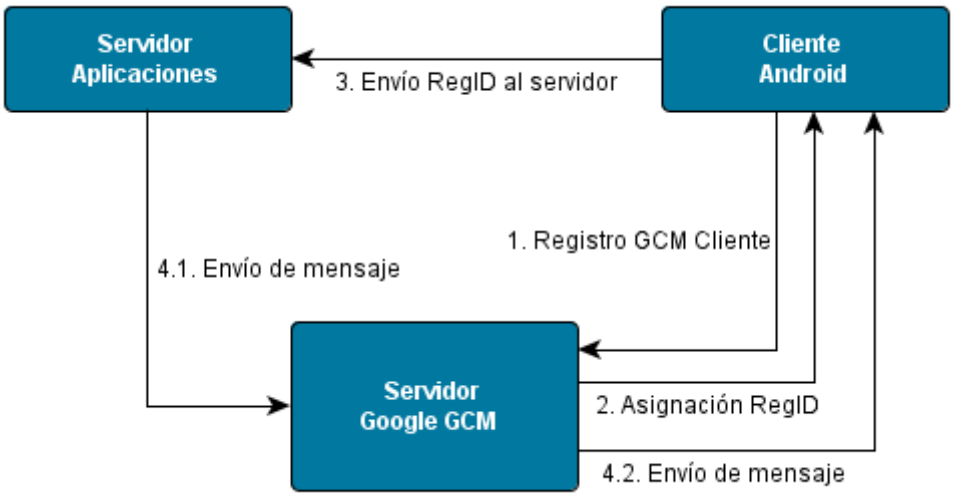


Ilustración 3. Diagrama de registro y envío de mensajes GCM: Fuente: http://www.sgoliver.net

EL protocolo de registro y envío de mensajes de la Ilustración 3 muestra el proceso completo que usa nuestra aplicación. En un primer momento, se detecta si el dispositivo móvil puede usar GCM, en ese momento se registra su RegID en Google Services y a partir de ese momento nuestro servidor podrá enviar mensajes al dispositivo usando el RegID y la plataforma de GCM.

Servidor LAMP: Linux + Apache + MySQL+ PHP

PHP es un lenguaje de programación interpretado, es decir, se ejecuta mediante un intérprete, por lo que resulta más flexible que los lenguajes compilados aunque puede resultar más lento. La principal ventaja es que se re trata de un lenguaje multiplataforma y que es ampliamente utilizado dentro de las tecnologías web.

Apache es un servidor web de código abierto multiplataforma y nos servirá para ofrecer el servicio web y pasar las peticiones para que PHP las interprete. Al igual que PHP es una de las plataformas más usadas en las tecnologías web.

Por último como motor de bases de datos relacional usaremos MySQL. MySQL es un software bastante ligero que es capaz de gestionar gran cantidad de información de forma rápida.

Estas tres tecnologías son de las más usadas a la hora de programar cualquier aplicación web. Lo más común a la hora de contratar un alojamiento web es encontrarse estas tecnologías juntas, frente a otras tecnologías como Tomcat, IIS+ASP o Django que está teniendo un fuerte crecimiento en las últimas fechas.

Dentro de nuestro proyecto, el servidor LAMP se encargará de ofrecer y almacenar vía web services toda la información necesaria para los clientes Android y a su vez comunicarse con los servidores de GCM para enviar los mensajes push cuando sea necesario.

Aplicación Android

La aplicación Android como ya hemos comentado será la encargada de la interacción con el usuario. Presentará toda la información y proveerá de las acciones oportunas para que los usuarios puedan operar de forma fácil e intuitiva. Será capaz de enviar y recibir información al servidor LAMP a través de los web services, de forma que la información almacenada en el dispositivo sea únicamente la necesaria para poder realizar las acciones oportunas y se devuelva al servidor para que posteriormente pueda ser accedida por otro dispositivo móvil. Además debe ser capaz de recibir las notificaciones push enviadas por el servidor LAMP a través del servicio de GCM.

A la vista de las necesidades la aplicación contará con dos vistas principales: eventos y avisos. A nivel de diseño la aplicación se desarrollará mediante pestañas y un menú lateral deslizante, al estilo de Material Design aunque en esta versión no se incorporarán todas las recomendaciones de diseño en cuanto a botones, estilos, etc. Se usará una actividad principal que controlará las pestañas que serán fragments, otras ventanas auxiliares que se usarán con posterioridad serán creadas mediante DialogFragments.

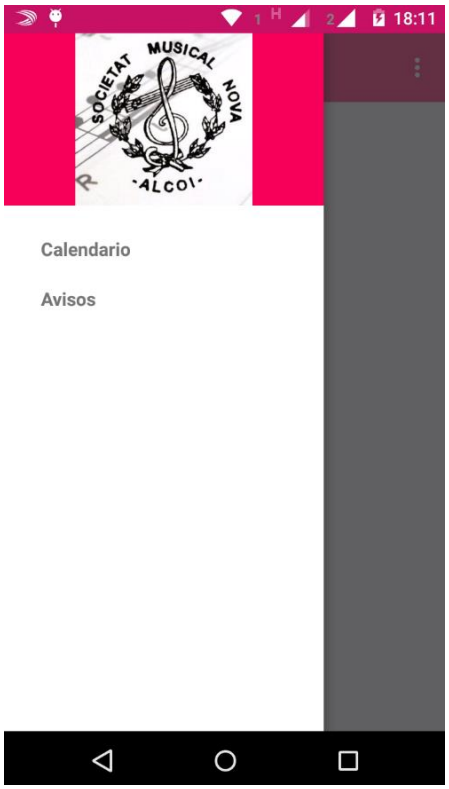


Ilustración 4. Menú lateral de opciones.

Por último, comentar que la aplicación es multi-idioma, está preparada para trabajar en inglés y en castellano. En este caso, la aplicación está muy focalizada y será muy raro que algún usuario trabaje en inglés, pero por estilo de desarrollo la aplicación está preparada y enteramente traducida a los dos idiomas. Esto también facilitará si en algún momento es necesario añadir un idioma como el catalán (Valenciano).

Casos de uso

En este apartado se describirá los usos de la aplicación y los componentes que intervienen en cada caso. Entiendo que es la mejor forma de explicar cómo funciona una aplicación y explicar todas las decisiones que se han tomado durante el desarrollo.

Registro del usuario.

En esta pantalla se recogerán los datos de los usuarios para acceder al sistema. Además en este activity se comprobará que el dispositivo cumple los requisitos y se registrará en GCM, todos los demás campos serán para uso de nuestra aplicación. Los datos serán:

- e-mail
- Instrumento
- Nombre
- Clave de acceso a la aplicación.

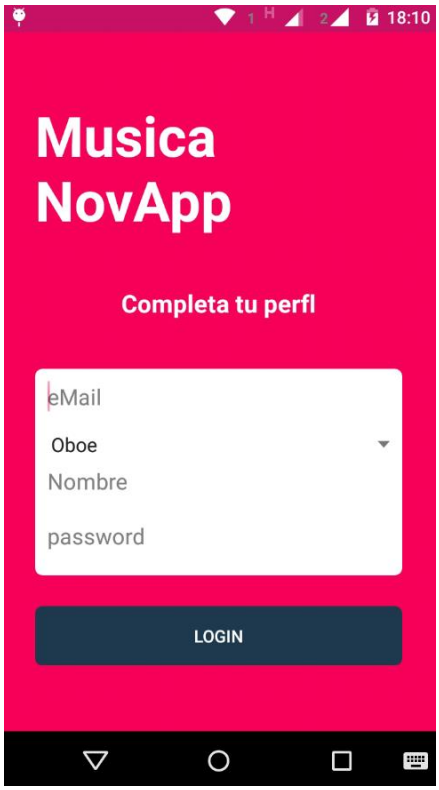


Ilustración 5. LaunchActivity. Recogida de datos.

La clave de acceso es un campo un poco particular. Esta clave lo único que hará es asegurarnos que nadie externo a la sociedad se registra en la plataforma. Realmente no hay información privada o sensible que no se pueda compartir con el público en general, pero es como una medida de seguridad adicional. Para la versión “beta” se ha usado la contraseña

genérica "clave", en versiones posteriores esta clave debería ser generada aleatoriamente para cada usuario.

En esta activity se obtendrá el regID del dispositivo para actuar con GCM y ese dato más los otros datos obtenidos serán guardados en el servidor LAMP para su posterior uso. Además se guardarán en las preferencias de la aplicación para de esa manera indicar a la aplicación que el usuario ya está registrado y por tanto no le pida más esta información en posteriores arranques de la aplicación.

Perfil de usuario

Seguidamente se describirán los casos de uso para el perfil de usuario. Estos casos de uso serán comunes para el perfil de administrador.

Vista del calendario de eventos.

El usuario dispondrá de una vista calendario en la que se mostrarán las fechas de los eventos programados. Al entrar a esta actividad la aplicación sincronizará mediante el Web Service los eventos programados. Estos eventos serán almacenados en la base de datos local de la aplicación.

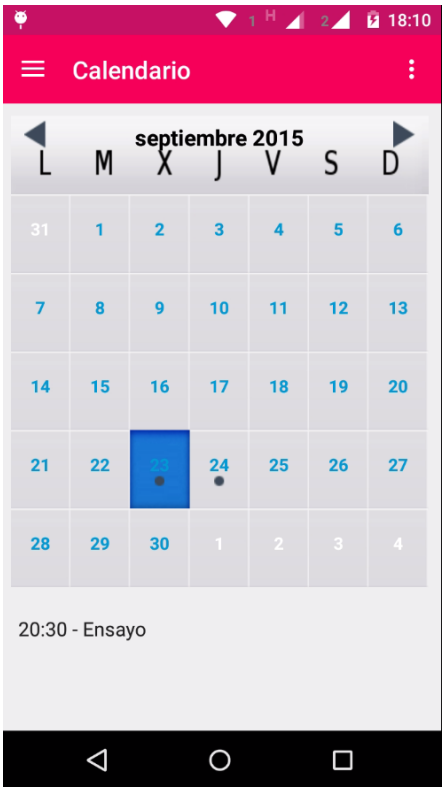


Ilustración 6. CalendarFragment. Vista de eventos.

El usuario dispondrá de una vista calendario mensual donde se marcaran los días con eventos programados y al pulsar sobre estos días se mostrará un listview con las actividades de ese día. En un primer momento, se intentó realizar esto con el widget CalendarView propio de Android, pero no cumplió todas las necesidades. CalendarView es un widget pensado para

seleccionar fechas, pero no está preparado para resaltar ciertos días. Para conseguir esto deberíamos de personalizar el widget.

Evaluando posibles alternativas se decide usar MFCalendarView[3]. Se trata de un proyecto de código abierto que muestra un componente calendario preparado para añadir eventos a días y altamente personalizable. Aun así, ha sido necesario retocar este componente para cambiar su comportamiento ya que no estaba preparado para configurar el primer día de la semana como lunes.

Detalle de evento y confirmación de asistencia.

En esta DialogFragment podremos visualizar todos los datos del evento. También podremos confirmar nuestra asistencia o no al evento, si no lo hemos hecho antes al recibir la notificación push. Esta confirmación se enviará al servidor LAMP para que los administradores puedan aprovechar esta información.

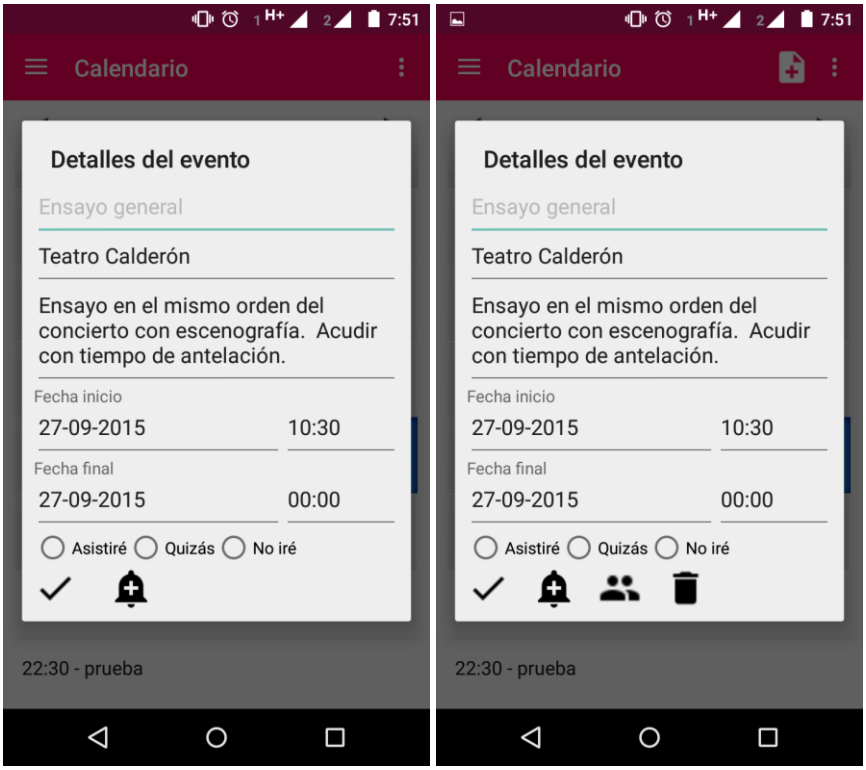


Ilustración 7. Detalles del evento. Usuario /Administrador

Cómo se aprecia en la ilustración anterior, en esta vista el administrador contará con dos opciones más, borrado de evento y control de asistencias, más tarde veremos estos casos.

Desde esta aplicación además, es posible exportar el evento a Google Calendar. De esta forma es posible aprovechar las propias notificaciones y recordatorios de este servicio sin ser intrusivos. Una opción que se barajó era que el propio sistema contara con un sistema de alertas, incluso sin confirmación por el propio usuario. Aunque se podría hacer, no creo que sea una opción de desarrollo correcta, ya que existen otras herramientas para ello y que el

[3] https://github.com/MustafaFerhan/MFCalendarView

usuario debería ser capaz de aceptar o rechazar estos eventos sin ser forzado a ello por la propia aplicación.

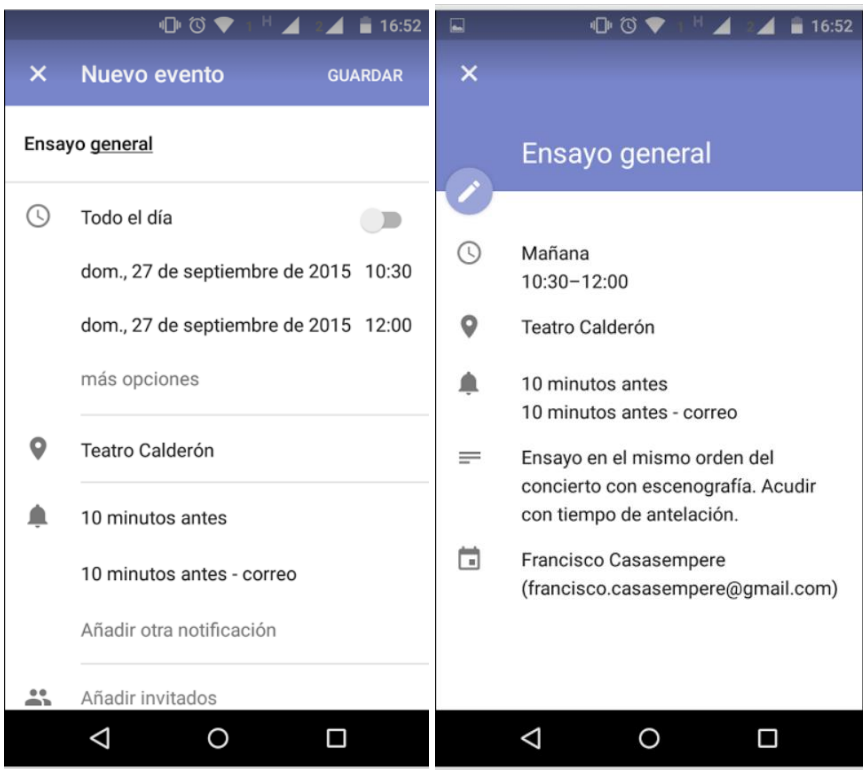


Ilustración 8. Exportación y evento creado en Google Calendar.

Vista de avisos.

En esta vista, el usuario podrá revisar en un listview todos los avisos recibidos, junta con la fecha de recepción. Estos avisos se encuentran almacenados en la base de datos local de la aplicación, desde el momento que la notificación push es recibida. Simplemente se trata de un listview ordenado de menos reciente (en la parte superior) a más reciente (en la parte inferior) y con un desplazamiento a la parte inferior de la pantalla, de modo que si existen más avisos de los que caben en la pantalla se verán siempre los más recientes.

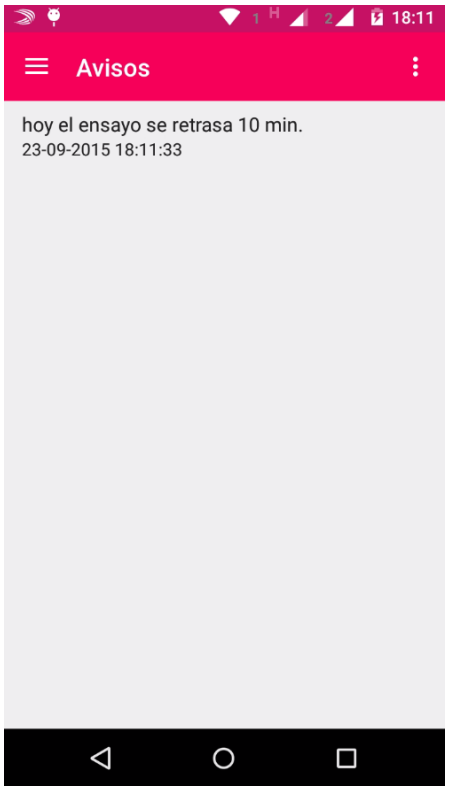


Ilustración 9. Vista de avisos.

Recepción de notificaciones

Esta es la auténtica ventaja de la aplicación. Gracias a los mensajes push cualquier información es recibida por el resto de usuarios en el mismo momento. La aplicación está preparada para recibir dos tipos de notificaciones: avisos y eventos.

En el primer caso, al recibir una notificación de aviso, se lanzará una notificación adjuntando el mensaje recibido. Este mensaje será almacenado en base de datos y mostrado en la vista de avisos. Si el usuario pulsa sobre la notificación será enviado a la vista de avisos.

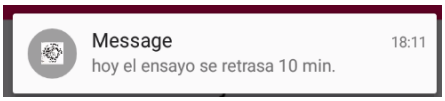


Ilustración 10. Notificación tipo Aviso.

En el caso de un evento la cosa cambia. En la barra de notificaciones se mostrará la notificación con el título del evento, fecha y hora y además tres botones de acción para confirmar asistencia:

- Asistiré
- Quizás
- No asistiré

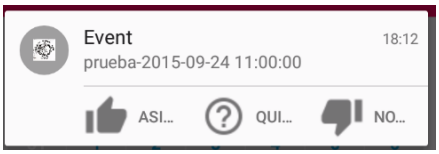


Ilustración 11. Notificación de nuevo Evento

Al pulsar sobre el evento, la aplicación mostrará el detalle del evento, pero al pulsar sobre alguna de las acciones además se guardará el estado seleccionado y esa información se enviará al web service para que los administradores puedan consultar si los usuarios han confirmado o no su asistencia al evento.

Perfil de administrador

El perfil de administrador tiene las mismas opciones que los usuarios, pero además tendrá disponible la creación de contenidos y avisos para el resto de usuarios.

Menú ajustes

En primer lugar para habilitar el modo administrador hay que entrar en ajustes. Se nos mostrará un dialog donde tendremos que introducir la contraseña “admin1234”. Esta clave podría ser sustituida en una versión posterior por un sistema que permitiera habilitar permisos de administrador a un usuario en concreto a través de la propia aplicación. Se estudiará en posteriores versiones.

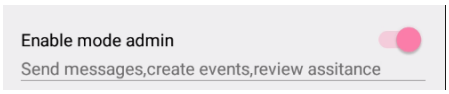


Ilustración 12. Modo administrador (inglés)

Creación de eventos

En la vista de calendario nos aparecerá un botón en la barra de herramientas para crear un nuevo evento. La aplicación mostrará un Fragment que recogerá varios datos. Los datos son los mismos que pide un evento de Google Calendar al ser creado, de esta forma, estos eventos podrán ser después exportados automáticamente a Google Calendar.

Se han incorporado dos vistas auxiliares para facilitar la elección de fecha y hora, se trata dos Dialogs para elegir fecha y hora.

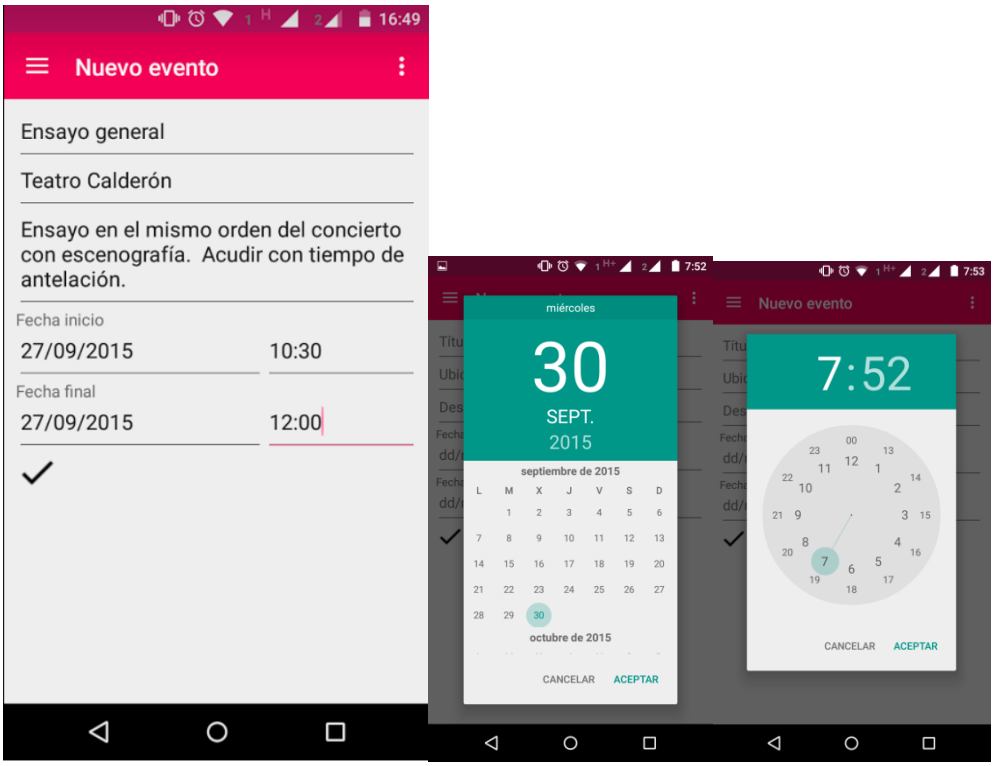


Ilustración 13. Nuevo evento. DatePickerDialog y TimePickerDialog

Una vez terminado de definir el evento esta información se enviará al servidor LAMP, y se enviará un mensaje push a todos los usuarios.

Envío de mensajes

Desde la pestaña de avisos, aparecerá al final de la pantalla un campo de texto y un botón. El administrador podrá escribir el mensaje que desee y enviarlo pulsando sobre el botón. Este mensaje será transmitido al servidor LAMP y este buscará los regid almacenados en su BBDD y reenviará el mensaje usando GCM a cada uno de los dispositivos registrados en la aplicación.

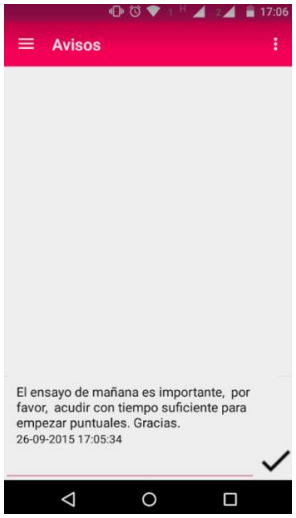


Ilustración 14. Envío de mensajes.

Control de asistencias

En la pantalla de detalle de eventos, los administradores disponen de un botón para acceder al control de asistencias. En esta pantalla se mostrará un listado en que todos los usuarios notificados habrán marcado su estado. Independientemente a este estado, el administrador dispondrá de un checkbox para confirmar la asistencia a un determinado acto ("pasar lista").

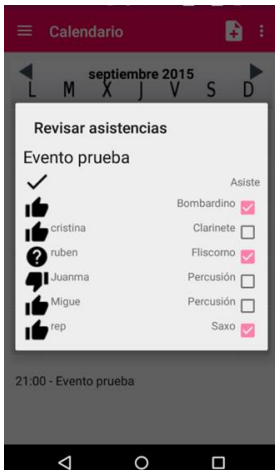


Ilustración 15. Control de asistencias.

Una vez confirmadas todas las asistencias, esta información se subirá al servidor LAMP. Además en caso de volver a entrar a esta opción por este dispositivo o cualquier otro dispositivo con perfil de administrador, se bajará la información anteriormente guardada para poder corregir cualquier dato.

Modelo de datos

Base de datos en servidor

En la base de datos del servidor se almacenará toda la información necesaria para los usuarios. Como ya hemos comentado en este proyecto hemos usado la base de datos de MySQL. En esta sección explicaremos las tablas de datos usadas y sus relaciones.

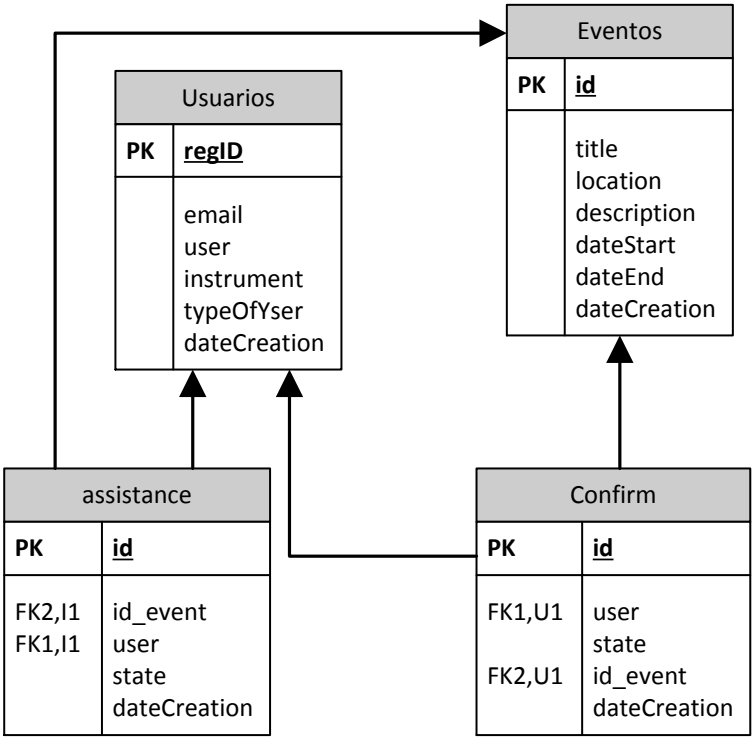


Ilustración 16. Diagrama base de datos servidor.

- Usuarios: En esta tabla se almacenarán los datos solicitados en la pantalla de registro más la fecha de alta. La clave primaria será regID. Al diseñar la aplicación la idea era que user fuera la clave primaria, pero más tarde se observó que un mismo usuario podría usar varios dispositivos y con regID se cumple esta premisa a la perfección.
- Eventos: En esta tabla se almacenan todos los datos necesarios y generados por la aplicación Android. La clave primaria será un id (INTEGER, AUTO INCREMENT).
- Assistance: En esta tabla se recogerán los datos de asistencia recogido desde las aplicaciones con perfil usuario, es decir, se almacenan los datos de asistencia que los usuarios han pulsado desde las notificaciones o desde el detalle del evento. La clave primaria será un id auto incrementado, pero tendrá una clave única con los campos user y id_event, de forma que no podrá haber dos estados distintos del mismo usuario para un determinado evento.
- Confirm: Esta tabla recogerá los datos de asistencia confirmada para los administradores, desde el DialogFragment de control de asistencia. Las claves de esta tabla serán exactamente las mismas explicadas en la tabla anterior.

Base de datos en cliente.

En el dispositivo Android también tendremos una pequeña base de datos (SQLite) para almacenar aquellos datos que se necesitan para la operativa normal, o aquellos datos que una vez descargados desde el web service no van a ser variados.

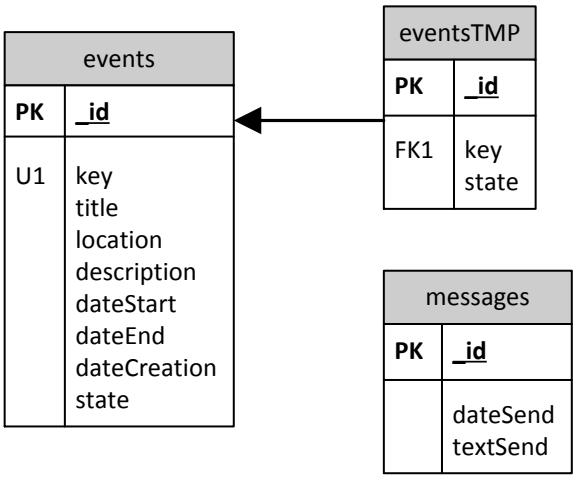


Ilustración 17.Diagrama de base de datos Android.

- Events: El campo _id será un entero con auto-increment generado por la propia base de datos, pero el campo key será con el que trabajaremos. Este campo viene desde el web service y será el "id" del evento en el servidor LAMP.
- Messages: En esta tabla se almacenarán todos los datos referentes a los mensajes recibidos.
- eventsTMP: >Esta tabla es para solventar una situación un poco peculiar. Los eventos son sincronizados al entrar en la vista de calendario. Esta tabla solo se usa cuando un usuario confirma su asistencia desde la barra de notificaciones. En ese caso, como el evento aún no existe, lo que hacemos es almacenar en una tabla temporal. Cuando se sincronice ese evento, se recogerá el estado marcado y se eliminará el registro de la tabla temporal. El estado normal de esta tabla será estar vacía.

Web Services.

En este apartado se describirán los web services usados desde la aplicación Android. En esta aplicación hemos incorporado dos tipos de mensajes, por un lado http con peticiones POST y por otro lado envío de mensajes JSON en las cabeceras.

Envío de mensaje push (push)

- Parámetros de entrada: Mensaje (POST)
- Enviará el mensaje a todos los dispositivos registrados a través de la plataforma GCM.

Registro (shareRegId)

- Parámetros de entrada: regID, email, name, instrument. (POST)
- Almacenará en BBDD todos estos datos para su posterior uso.

Nuevo evento (event)

- Parámetros de entrada: title, location, description, dateStart, dateEnd.

- Almacenará en BBDD estos datos y enviaría un mensaje PUSH a todos los dispositivos registrados pasando como parámetro el título y la fecha/hora de inicio.

Recogida de eventos (getEvents)

- Devolverá un objeto JSON con un array de todos los eventos programados con fecha de inicio, la fecha actual menos 30 días.
- Parámetros de salida:

```
[
    {
        id:1,
        Title:"Titulo",
        Location:"Ubicación del acto",
        Description:"Descripción del acto",
        dateStart: "Fecha/hora inicio",
        dateEnd:"Fecha/hora fin",
        dateCreation:"Fecha/hora creación"
    },
    …

]
```

Confirma asistencia (setAssistance)

- Parámetros de entrada: key,user,state
- Almacenará en BBDD el estado seleccionado por un determinado usuario para un evento con clave key.

Recogida de asistencias (getAssistance)

- Parámetros de entrada: key (clave de evento)
- Devolverá un JSON array con los datos de confirmación de asistencias seleccionados tanto por los usuarios como por los administradores. Se añade el campo instrumento para que el listview que mostrará estos datos se ordene por él ya que es mucho más sencillo pasar lista por cuerdas que por nombre de los músicos,
- Parámetro de salida:

```
[
    {
        user:"Usuario",
        state:"Confirmación de asistencia del usuario",
        instrument:"instrumento",
        confirm:"Confirmación de asistencia por un
        administrador"
    },
    …
]
```

Confirmación de asistencias administrador (confirm)

- Parámetros de entrada: Se envía un objeto JSON con un array en su interior con las asistencias confirmadas.

```
{
    Key: "Clave del evento",
    [
        {
            user:"Usuario",
            state:"Confirmación de asistencia por administrador",
            instrument:"instrumento",
        },
        …
    ]
}
```

- Almacenará en la base de datos esta información para posterior uso por parte de los administradores.

Borrado de evento (deleteEvent)

- Parámetros de entrada: key
- Eliminará de la base de datos el evento indicado por key, de estao forma en la próxima actualización de la lista de eventos por parte de los clientes, este evento será borrado de la base de datos local de cada dispositivo.

Vistas

En el siguiente esquema se muestra el esquema de navegación de la aplicación.

- LaunchActivity. Petición de datos y registro del usuario.
- MainActivity. Pantalla principal con las vistas calendario y avisos.
- Calendario. Desde esta pantalla se puede ver el detalle de los eventos.
    - En el caso de los administradores tendrán disponible un botón para crear nuevos eventos.
    - Detalle de eventos:
        - Desde aquí podrán confirmar la asistencia al evento.
        - Los administradores podrán además revisar las asistencias.
        - Se podrá exportar la cita a Google Calendar.

- Avisos: Desde esta pantalla se podrá revisar los últimos avisos recibidos y en el caso de usuarios del perfil administrador, enviar nuevos mensajes.

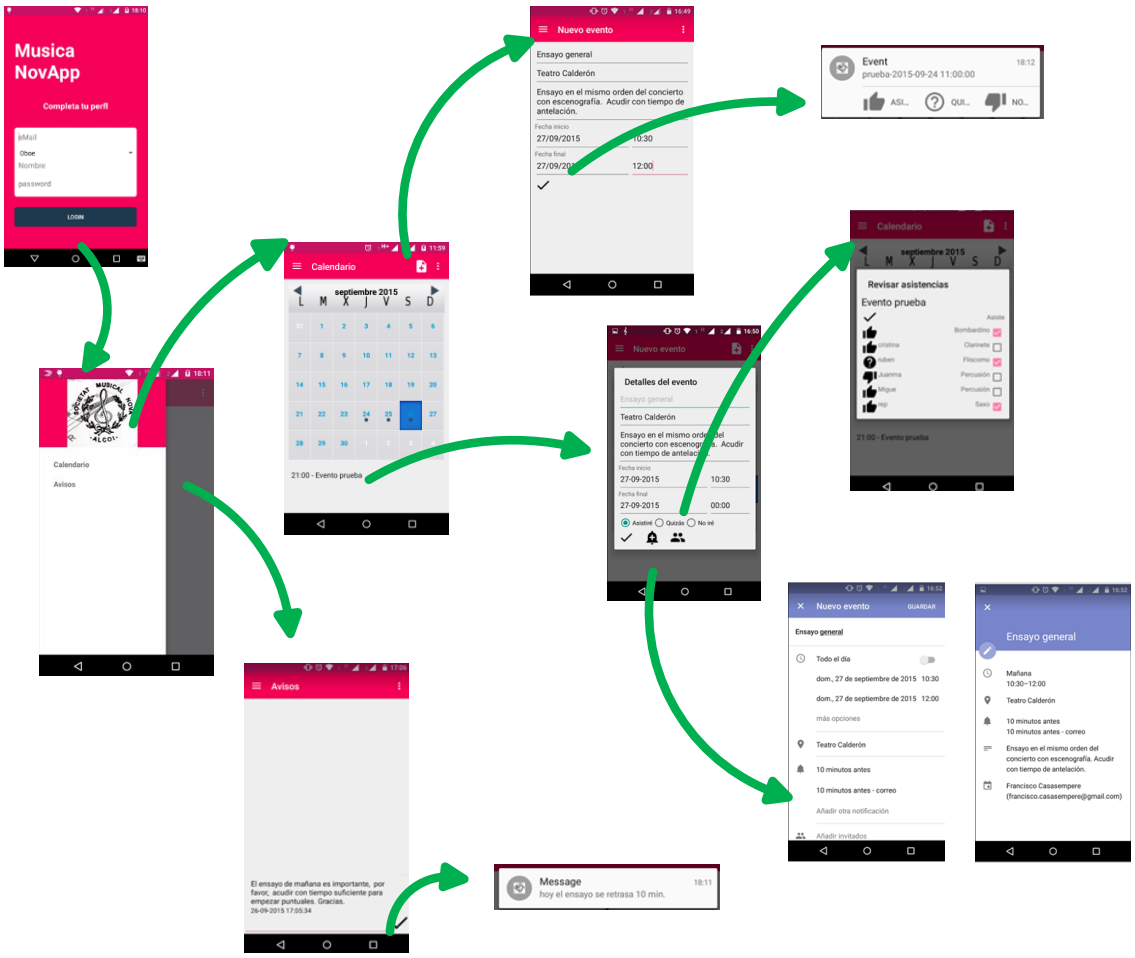


Ilustración 18. Esquema de navegación

Conclusiones

En este capítulo se hace un repaso global al proyecto, presentando las conclusiones finales en comparación con los objetivos marcados inicialmente y los posibles trabajos futuros.

Grado de cumplimiento de los objetivos planteados.

Una vez concluidas las principales tareas que forman este proyecto es el momento de hacer balance y crítica de los resultados obtenidos. Repasando los diferentes objetivos pueden sacarse las siguientes conclusiones:

- **Conocer las necesidades reales de la sociedad.** Después de varias reuniones y conversaciones con miembros de la junta directiva de la sociedad musical y también con el conocimiento previo del funcionamiento que ya teníamos. Hemos podido simplificar al máximo las necesidades de comunicación de la sociedad y determinar los mecanismos que el proyecto debería de contemplar, únicamente calendario y avisos.
- **Buscar alternativas en el mercado.** Se han hecho varias búsquedas de aplicaciones y comparativas de herramientas como slack, telegram, WhatsApp y si bien todas podrían ser válidas, se creyó oportuno desarrollar una aplicación a medida que permitiera mayor control sobre el manejo y generación de la información.
- **Generación de una aplicación propia.** La aplicación desarrollada cumple las necesidades que se solicitaron.
    - Publicación de eventos.
    - Envío de avisos.
    - Control de asistencias
    - Perfil de usuario/administrador.

Estos puntos o "historias de usuario"[4] han sido claves para desarrollar los puntos que la aplicación debía soportar. Se han desarrollado completamente y en algunos casos mejorados y ampliados.

En conclusión, puede decirse que la aplicación abarca la totalidad de los objetivos iniciales.

Líneas abiertas.

A la vista del resultado final se plantean varios puntos que habría que mejorar en futuras versiones.

[4] [4] http://www.javiergarzas.com/2011/12/historia-de-usuario-diferente-de-requisito.html

1. Cambiar el sistema de login. Sería necesario una opción para invitar y dar de alta a los usuarios nuevos. Se plantea la posibilidad de crear un formulario para invitar a usuarios. En este formulario se recogería: nombre de usuario, instrumento y clave y se subiría al servidor. La pantalla de login cambiaría a sólo usuario y clave, ya que todos los demás datos están almacenados en el servidor. Además existiría una vista para visualizar todos los usuarios diferenciando entre los usuarios ya registrados y los no activados, en el que aparecerían los datos de login para poder ser enviados vía: SMS, correo y otro programa de mensajería.
2. Retocar el diseño general de la aplicación para adaptarla a las especificaciones de Material Design.
3. Permitir envío de imágenes desde la pantalla de avisos. Podría ser útil en determinadas ocasiones.
4. Por último, preparar la aplicación para que pudiera ser utilizada por otras sociedades.

El punto 4, será evaluado después de la puesta en producción de la aplicación en la Societat Musical Nova d'Alcoi como experiencia piloto. Después de un tiempo de uso, seguro que surgen sugerencias y mejoras que una vez consolidadas podrían llevar a la publicación de la aplicación para uso por otras sociedades e incluso la valoración de un modelo freemium (usuarios gratis, administradores pago por suscripción).

Consideraciones personales.

En cuanto a las valoraciones personales, cabe decir que programar en Android es un trabajo agradecido y que me gusta. Normalmente por mi trabajo, no suelo programar en esta plataforma y la inmediatez de resultados que proporciona es muy gratificante. Además realizar un proyecto para una sociedad a la cual pertenezco desde hace más de 20 años y en la que tengo tantos amigos, siempre es un placer y un reto. Sobre todo un reto, por las expectativas que se crean y que posteriormente hay que cumplir.

El proyecto en concreto me ha gustado, ya que al plantearme un proyecto para el final de master siempre busqué algo que pudiera probar con personas y que fuese práctico. Además creo que la herramienta realmente será útil y se usará completamente.

En cuanto a la parte de aprendizaje, ha sido el primer proyecto que he desarrollado íntegramente con Android Studio a parte de algunos ejemplos realizados durante el transcurso del máster. Me ha encantado la forma de trabajar y las ayudas que ofrece. Al principio hay ciertos puntos que confunden un poco, ya que cambian bastante desde Eclipse. El archivo gradle seguramente era uno de los puntos que más me ha costado entender, pero una vez hecho la verdad es que facilita enormemente la tarea de desarrollo

Anexos

Código fuente en BitBucket (Android)

El código fuente está en BitBucket en el siguiente repositorio Git.

https://SiscoCS@bitbucket.org/SiscoCS/musicanovapp.git

La razón de usar este repositorio frente a otros repositorios on-line gratuitos como gitHub es por la posibilidad de tener repositorios privados. Una vez terminada la evaluación del proyecto el repositorio será otra vez privado.

Código fuente Servidor LAMP

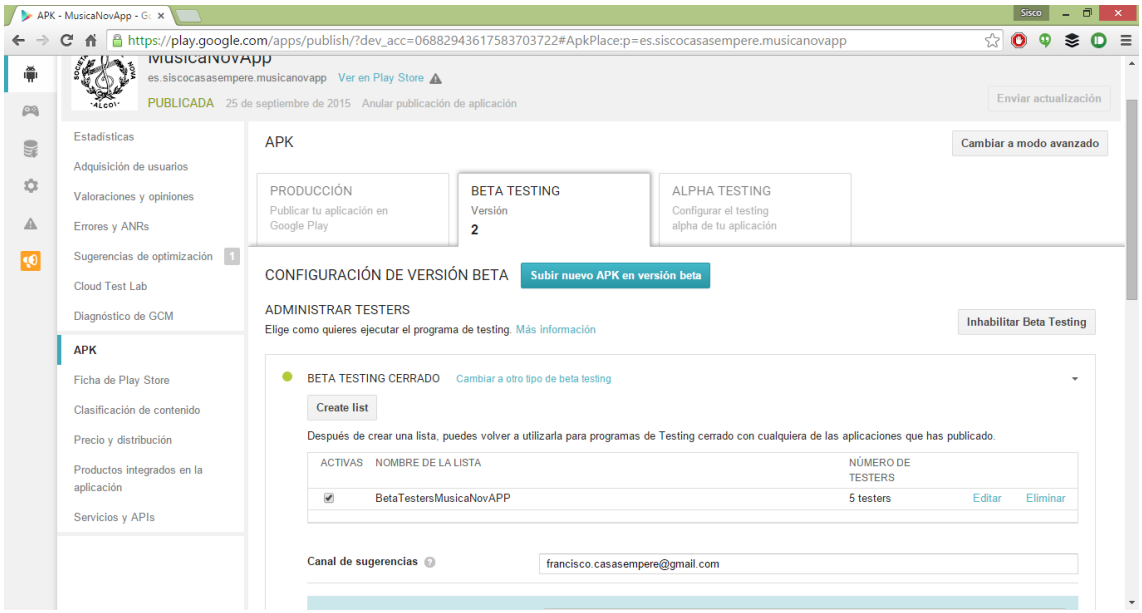
El código fuente de los web services, está incluído en la entrega del proeycto y sólo cuenta con dos archivos:

- Gcm.php (Web Service general, parámetros GET y POST)
- Wsgcm.php (Web Service para control de asistencias, parámetros de entrada tipo JSON)

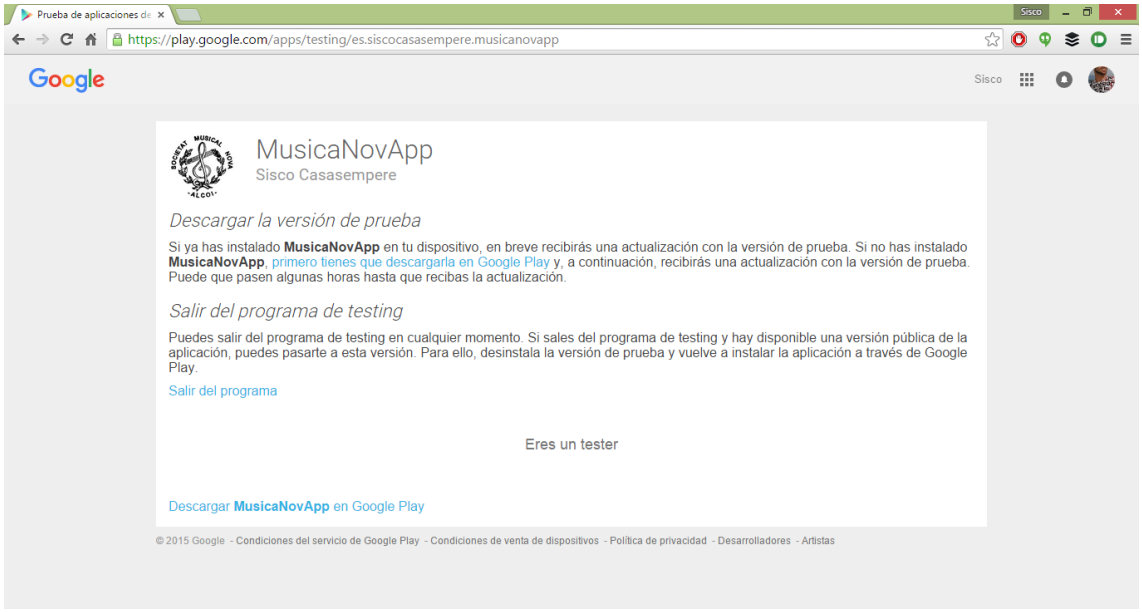
Publicación de versiones beta en Google Play

Google Play Developer Console provee de una herramientas para publicar versiones Betas. La aplicación no aparece en Google Play salvo para las cuentas de Google que hayan sido especificadas en la lista de betatesters. La gran ventaja es que ya no hay que distribuir el apk entre los beta testers de forma manual. Esa forma de distribuir implica que haya que activar la opción de ubicaciones desconocidas en los dispositivos de prueba. Además cualquier actualización que se vaya publicando es necesario volver a distribuir el apk.

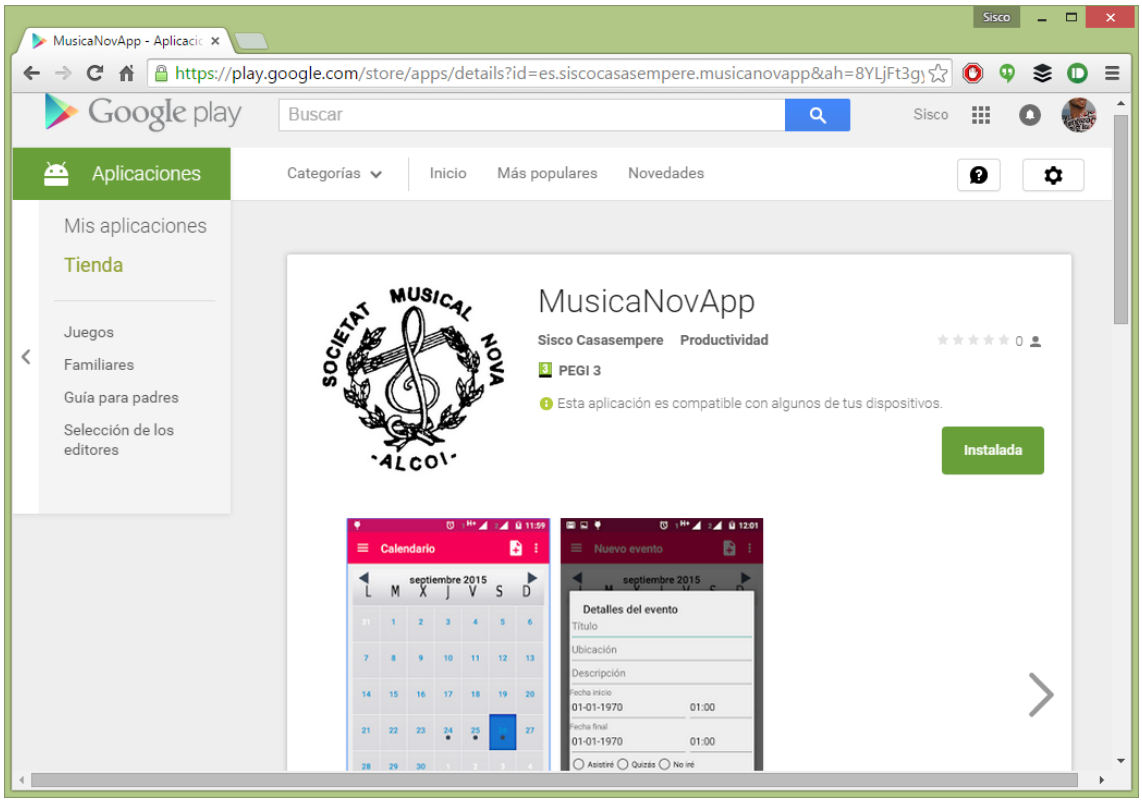
El mecanismo es el mismo que para la publicación de cualquier aplicación. Una vez dada de alta la aplicación y completados todos los campos obligatorios para la clasificación de la aplicación. Al subir el APK se especificará que se trabaja en modo BETA TESTING.



Se creará un listado con las direcciones de corre de los betatesters. Y una vez que el alta de la aplicación se haya completado aparecerá un enlace que tendremos que distribuir manualmente a nuestros betatesters.



Una vez completado este paso, ya aparecerá la pantalla de Google Play para proceder a la descarga e instalación de la versión beta de nuestra app.



La gran ventaja de este sistema son varias:

- Las actualizaciones de la aplicación serán notificadas automáticamente a los betatesters.
- Se obtendrán estadísticas de uso e instalación de la app.
- Se proveerá de un sistema para capturar errores y bugs y de una dirección de correo para notificar posibles problemas.

La única desventaja reseñable es el tiempo en que tarda en notificarse desde que se lanza una actualización hasta que los usuarios son notificados. Pero cuando estamos hablando de versiones BETA, en las que se supone que la aplicación está práticamente lista para lanzarse como versión estable no debería ser demasiado problema. Agradecer a los compañeros de la banda y del trabajo que se han brindado a probarla aplicación y sugerirme ciertos cambios.