UNIVERSIDADE DA CORUÑA

PROGRAMACIÓN DE SISTEMAS 20/21 Q1

"Fagamos barullo"

Autores: Iñaki y Santiago
Fecha: *A Coruña, Enero 2021*

Índice

Capítulos **Página**

1. Introducción **1**
1.1. Objetivos . . . 1
1.2. Motivación . . . 1
1.3. Trabajo relacionado . . . 1

2. Análisis de requisitos **1**
2.1. Funcionalidades . . . 1
2.2. Prioridades . . . 2

3. Planificación inicial **2**
3.1. Iteraciones . . . 2
3.1.1. Iteración 1: registro y persistencia . . . 2
3.1.2. Iteración 2: perfil músico . . . 2
3.1.3. Iteración 3: explorar músicos . . . 2
3.1.4. Iteración 4: crear grupos . . . 3
3.1.5. Iteración 5: promotores . . . 3
3.2. Responsabilidades . . . 3
3.3. Hitos . . . 3
3.3.1. Hito 1: explorar músicos . . . 3
3.3.2. Hito 2: explorar bandas . . . 3
3.3.3. Hito 3: promotores . . . 3
3.4. Incidencias . . . 4
3.4.1. Indicendia 1: *back-end* y capa de acceso a servicios . . . 4

4. Estado actual **4**

5. Diseño **4**
5.1. Arquitectura . . . 4
5.1.1. Vistas . . . 4
5.1.2. Controladores . . . 6
5.1.3. Modelo . . . 6
5.2. Interfaces de usuario . . . 6
5.3. Pruebas . . . 7
5.3.1. Pruebas de interfaz . . . 7
5.4. Librerías de terceros . . . 7
5.4.1. Picasso . . . 7
5.4.2. Butter Knife . . . 7
5.4.3. FlexboxLayout . . . 7

6. Trabajo futuro **7**

Anexo de figuras **8**

Cuadro 1: Tabla de versiones.

Versión	Fecha	Autor
1.0	13/10/2020	Iñaki y Santiago
2.0	29/10/2020	Iñaki y Santiago
3.0	28/11/2020	Iñaki y Santiago
4.0	23/12/2020	Iñaki y Santiago
5.0	26/01/2020	Iñaki y Santiago

1. Introducción

1.1. Objetivos

Desarrollar una aplicación para encontrar integrantes para bandas emergentes locales de una forma rápida y sencilla. Nos centraremos en crear una aplicación con una experiencia de usuario al nivel del 2020 y que cumpla con las expectativas de los usuarios —es decir, que les ayude realmente a encontrar músicos para sus bandas—.

También, la aplicación servirá para que promotores, u organizadores de eventos, puedan encontrar bandas locales y prometedoras con ganas de pisar un escenario por primera vez.

1.2. Motivación

Encontrar integrantes para un nuevo grupo es muy difícil. Si alguna vez te has visto en esta situación, te habrás dado cuenta de que no importa cuántos carteles cuelgues en las salas de ensayos locales, ni siquiera cómo de llamativos sean. Todos tus esfuerzos serán en vano y tu cartel será completamente ignorado. Además, si eres nuevo en la ciudad, o simplemente estás empezando en el mundo de la música, vas a tener pocos contactos en tu agenda de los que poder depender.

Otra opción son las páginas de anuncios en línea, como por ejemplo `https://www.milanuncios.com`. Sin embargo, estas páginas tampoco son una buena opción. Los anuncios de grupos que buscan integrantes rápidamente quedan tapados por otros más atractivos —guitarras que se venden por ejemplo— que generan más visitas y se acaban posicionando de primeros en los resultados de búsqueda.

Por último, promocionar una banda es igual o más difícil que montarla. Por eso, queremos desarrollar una aplicación sencilla y eficaz que ayude a bandas emergentes a empezar a despegar en el mundo de la música.

1.3. Trabajo relacionado

Sin duda existen ya infinidad de aplicaciones que ayudan a músicos a encontrar bandas. Además, ahora que las redes sociales son prácticamente el CV del siglo XXI, casi cualquier página sería competidora de nuestra aplicación. Sin embargo, creemos que tenemos potencial por encima de estas plataformas, ya que ofrecemos una funcionalidad muy específica que ayuda a empezar y a promocionar bandas locales de forma sencilla y directa.

2. Análisis de requisitos

2.1. Funcionalidades

Queremos que nuestra aplicación registre dos perfiles de usuario: músico y promotor. Con el perfil de músico creado, podemos proceder a: buscar bandas locales

que necesitan integrantes o crear nuestra propia banda y crear un anuncio para encontrar integrantes.

En cuanto al proceso de búsqueda de bandas, cada músico verá solo las bandas que buscan músicos con su perfil. Es decir, bandas que buscan los instrumentos que sepa tocar el usuario. Además, también será posible explorar bandas locales.

Para que los músicos destaquen, podrán personalizar su perfil con fotos y varias maquetas con las que demuestren su habilidad. También será posible valorar positivamente el perfil de un músico para que aparezca por encima del resto en las búsquedas locales. De la misma forma, las bandas podrán editar su perfil para venderse de forma más atractiva a los promotores y podrán recibir valoraciones positivas.

2.2. Prioridades

La prioridad número uno es desarrollar una versión dónde se puedan registrar los músicos y poder ver a los músicos de tu zona atendiendo a una serie de filtros (instrumento, edad, etc.). Tendrán un perfil básico con fotos, maquetas e información personal y de contacto.

En una fase posterior, la aplicación podrá registrar bandas, que son agrupaciones de uno o más músicos. El perfil de banda contiene fotos, alguna maqueta e información básica. Además, como con los músicos, también será posible explorar bandas. Podremos filtrar la búsqueda con filtros como el género, faltan integrantes, etc.

Por último, aparecería el perfil de promotor. Este perfil puede explorar bandas y músicos con el fin de contactar con ellos para ofrecer contratos de grabación, conciertos, etc.

3. Planificación inicial

3.1. Iteraciones

3.1.1. Iteración 1: registro y persistencia

Desarrollar una aplicación sencilla dónde podamos registrar músicos. También podremos identificarnos, por lo que tiene que existir persistencia.

3.1.2. Iteración 2: perfil músico

Al final de esta iteración podremos acceder a nuestro perfil y editarlo. Podremos modificar información de contacto y personal, fotos y maquetas.

3.1.3. Iteración 3: explorar músicos

En esta iteración implementaremos la funcionalidad de poder buscar otros músicos —pudiendo hacer uso de filtros— y acceder a sus perfiles. Además, podremos valorar sus perfiles positivamente.

3.1.4. Iteración 4: crear grupos

Se podrán crear bandas, editar su información y buscarlas. Existirá un filtro de búsqueda especial con el que encontrar bandas que busquen integrantes.

3.1.5. Iteración 5: promotores

Se introduce el perfil de promotor que podrá explorar bandas locales emergentes.

3.2. Responsabilidades

Por preferencia, Santiago prefiere el *front-end*, con un perfil especializado en la usabilidad y el desarrollo orientado al *testing* e Iñaki el *back-end*, ya que le parece un reto y le gusta ponerse a prueba diseñando arquitecturas sencillas y eficientes. Sin embargo, seremos flexibles y podremos ayudar el uno al otro. Idealmente descompondremos cada iteración en pequeñas tareas y trabajaremos de acuerdo a la filosofía *Git Flow*.

3.3. Hitos

3.3.1. Hito 1: explorar músicos

Como ya comentamos en el análisis de requisitos, lo más importante de la aplicación es poder buscar a otros músicos. Por eso, fijamos un hito que será una versión de la aplicación con la que podremos registrarnos y buscar perfiles locales.

Programaremos el registro, el *login*, la búsqueda y la edición y visualización de perfiles. Haremos tests de unidad cuando sea necesario y testeo de interfaz para comprobar que todos los casos de uso de este hito funcionan correctamente.

3.3.2. Hito 2: explorar bandas

El segundo hito de nuestro proyecto, será tener una aplicación en la que podamos crear bandas y anuncios para encontrar integrantes. De esta forma, tendremos una experiencia más rica a la hora de interactuar con la aplicación y no nos limitaremos únicamente a ver perfiles de otros músicos.

Programaremos la creación de grupos, la edición de bandas y la búsqueda local. De nuevo, haremos tests unitarios y probaremos los casos de uso comprobando que todo fluye correctamente.

3.3.3. Hito 3: promotores

Por último, aparecerá el rol de promotor que podrá explorar bandas y obtener su información de contacto para contratarlas, etc.

Una vez más, probaremos las unidades que sean necesarias y probaremos los casos de uso relacionados.

3.4. Incidencias

No seguiremos un plan de gestión de riesgos. Sin embargo, seremos cuidadosos y si encontramos un problema que ponga en riesgo el transcurso del proyecto, contactaremos con los profesores para obtener alternativas y soluciones.

3.4.1. Indicendia 1: *back-end* y capa de acceso a servicios

El *back-end* es un hueso importante de la aplicación. Es imprescindible poder comunicarnos con un servidor para poder acabar la iteración 1.

En un primer momento pensamos en desarrollar un *back-end* pequeño usando entornos de desarrollo sencillos y con una curva de aprendizaje pequeña (i.e. Python Flask). Sin embargo, nos dimos cuenta de que también necesitamos crear la capa de acceso a servicios en el código *Java*. Entonces, decidimos resolver el problema usando Firebase. Así, no tendremos que implementar el servidor remoto ni la capa de acceso a servicios y nos podremos centrar en lo que importa: que la aplicación funcione.

4. Estado actual

Hito 1 cumplido (Iteraciones 1, 2 y 3). La aplicación es completamente funcional y resuelve el objetivo principal: poder encontrar integrantes para bandas emergentes locales. Además, tiene una interfaz de usuario moderna y sigue los mejores principios de usabilidad (ver fotos del anexo).

Concretamente, la apliación permite registrarnos para poder acceder al servicio. Desde ahí, podemos visualizar nuestro perfil y editarlo (descripción, instrumentos, fotos y maquetas). Por último, podemos buscar y ver el perfil de otros usuarios filtrandolos por nombre, instrumentos y "favoritos".

5. Diseño

5.1. Arquitectura

Seguiremos el modelo MVVP visto en clase para desarrollar la aplicación por capas. Las actividades representarán las vistas y el *presenter* las engranará con la lógica de negocio (i.e. el modelo). Así conseguiremos responsabilidades únicas y podremos trabajar por interfaces. Lo que hace que el reparto de tareas sea más sencillo y se pueda ajustar más al perfil de cada desarrollador.

5.1.1. Vistas

Las vistas son las encargadas de mostrar por pantalla la información al usuario. En el caso de Android, las vistas se implementan a través de Actividades. Hasta el

Cuadro 2: Actividades de la aplicación.

Actividad	Descripción
`SplashScreen`*	Punto de entrada
`Register`	Implementa la lógica de registro
`Login`	Implementa la lógica de login
`MainActivity`	Landing page (una vez identificado)
`ProfileSettings`	Lógica para modificar nuestro perfil
`AccountSettings`	Lógica para modificar la cuenta
`Licenses`	Licencias de librerías de terceros
`AddPhoto`	Lógica para añadir una foto
`AddMp3`	Lógica para añadir una maqueta

(*) Es la actividad principal de la aplicación y punto de entrada que vemos en el launcher de Android. Esta actividad nos redirige a `MainActivity` si ya estamos identificados o a `Register` en el caso contrario.

Cuadro 3: Fragmentos de la aplicación.

Fragmento	Descripción
`Profile`	Muestra información de un usuario
`Search`	Implementa la lógica de búsqueda
`Settings`	Punto de entrada al resto de ajustes

momento, la aplicación se compone de nueve actividades (ver cuadro 5.1.1) y tres fragmentos (ver cuadro 5.1.1).

La actividad `SplashScreen` es el punto de entrada a la aplicación. Se encarga de determinar si ya estamos identificados o no. En el caso de estarlo nos redirige a la `MainActivity`, sino a la actividades `Register` o `Login`.

Las actividades `Register` y `Login` son las encargadas de, respectivamente, ofrecer la funcionalidad de registro e identificación. Son triviales, ya que no presentan ningún elemento complejo.

Por otro lado, la activiadad `MainActivity` es algo más compleja. Esta actividad es la página de entrada a la aplicación cuando nos hemos identificado. Tiene una *NavigationBar* que nos permite movernos por tres pestañas distintas: mi perfil, búsqueda de usuarios y ajustes. Estas pantallas están implementadas como *Fragments*[1] (`Profile`, `Search` y `Settings`) y con un controlador por cada uno de ellos[2].

En cuanto al resto de actividades, de nuevo, son triviales. Implementan la funcionalidad que indica el cuadro 5.1.1 de forma sencilla e intuitiva para el usuario.

[1]De esta forma se pueden reusar en la aplicación. En concreto, el fragmento de `Profile` se usa indistintamente para mostrar información sobre nuestro perfil y el de otros usuarios.

[2]También podíamos haber usado un *mega-presenter*, con el que probablemente se hubiese podido reusar código entre las vistas. Sin embargo, el problema de esta aproximación es que hay que crear más interfaces entre los fragmentos y propagar los resultados entre más vistas, lo que añade complejidad de forma innecesaria.

5.1.2. Controladores

Uno por cada vista descrita en la subsección anterior. Unen la vista con la lógica de negocio (i.e. modelo). De esta forma, conseguimos que las actividades simplemente gestionen lo relacionado con la vista y que el modelo solo implemente la lógica de negocio. Estas clases simplemente transforman los flujos de datos para que, tanto las vistas como los modelos, reciban únicamente los datos que les interesan.

5.1.3. Modelo

En este caso, la lógica de negocio se encarga de: la autenticación y el almacenamiento de información sobre usuarios en la red. A su vez, se divide en dos capas distintas: los servicios —que implementan la lógica propiamente dicho— y los *data-sources*, que abstraen el acceso a los datos (tanto si son por red, al almacenamiento local, etc.).

Su implementación no tiene mayor relevancia —llamadas entre capas y se usan las librerías de Firebase para acceder a la red—. Por lo que nos limitaremos a comentar los datos que usamos en nuestra lógica de negocio para implementar la funcionalidad de la aplicación. En cuanto al servicio de autenticación, guardamos los siguientes datos para un usuario

Email*	Password	UID

Sobre cada usuario se guarda la siguiente información

UID*	Full name	Phone number	Profile pic	Instruments

Además, también se guarda la relación de perfiles favoritos

Following*	Follower*

Fotos

FID*	UID	PhotoBlob	PhotoTitle

Y maquetas

DID*	UID	DemoBlob	DemoPhotoBlob	DemoTitle

5.2. Interfaces de usuario

Las interfaces de usuario siguen las mejores prácticas de las guías de diseño de *Material Design*. De esta forma, la aplicación ofrece una interfaz agradable y sencilla con una buena experiencia de usuario (ver anexo de figuras).

5.3. Pruebas

5.3.1. Pruebas de interfaz

Se han programado *UITests* con el *framework* Espresso para llevar al límite los casos de uso `Register` y `Login` y comprobar si funciona todo correctamente —también permiten comprobar que la aplicación es robusta, ya que los datos se validan antes de enviarlos al servidor—. Además, también se ha probado la aplicación manualmente para comprobar que la navegación es consistente (i.e. cuando te identificas no puedes volver para atrás).

Nota: Sería interesante poder automatizar la ejecución de las pruebas de unidad e interfaz con las *pipelines* de GitLab. De este modo, cuando se empujase un *commit* al repositorio, se notificaría al desarrollador de si los cambios han pasado los *tests* de forma satisfactoria o no. De este modo, podríamos realizar un desarrollo orientado a pruebas en el que primero programásemos los *tests* y después los casos de uso.

5.4. Librerías de terceros

Todas las licencias están correctamente embebidas en la actividad `Licenses` para cumplir con las condiciones de uso del *software*.

5.4.1. Picasso

Permite gestionar la descarga de imágenes de forma sencilla y se encarga de *cachear* resultados automáticamente, liberándonos de gran parte de la gestión con el sistema de ficheros de Android.

5.4.2. Butter Knife

Gestiona la asociación de *widgets* a variables automáticamente. Permite eliminar gran parte del *boilerplate* necesario para poder trabajar con las actividades.

5.4.3. FlexboxLayout

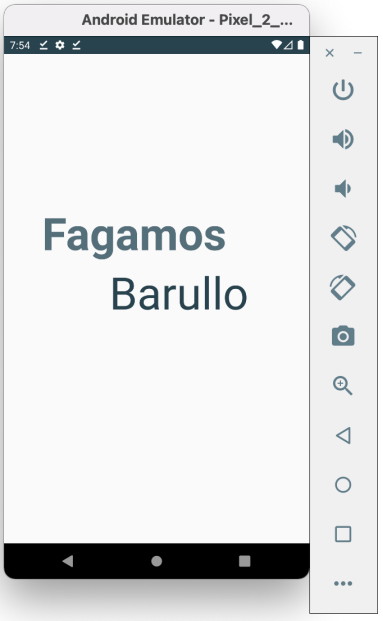
Es una librería que permite usar un *layout* similar al *CSS Flexible Box Layout* en Android. De esta forma, no tenemos que preocuparnos de gestionar vistas complejas en la que los elementos se mueven y tienen que estar perfectamente centrados.

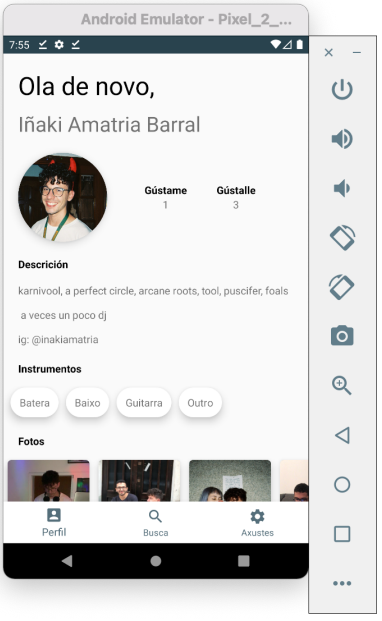
6. Trabajo futuro

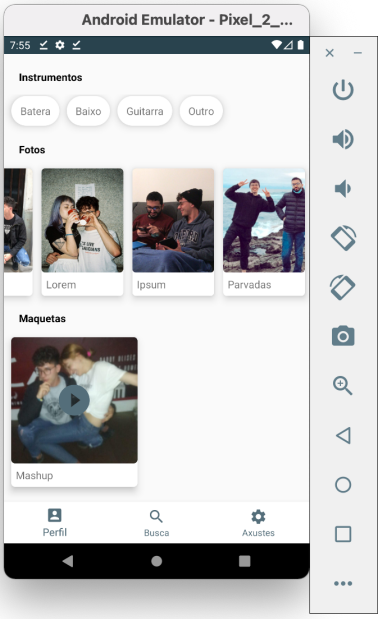
Con el estado actual de la aplicación, sería sencillo acabar con los hitos 2 y 3. De esta forma, la aplicación podría pasar a usarse en una ciudad —i.e. Pontevedra—

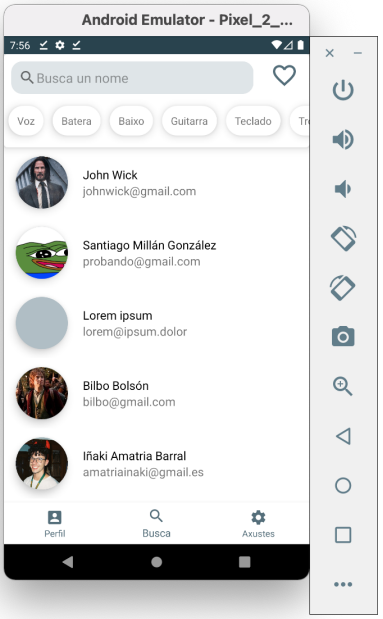
para que músicos jóvenes puedan formar una banda, empezar a ensañar en el local de música del *concello* y fácilmente conseguir "bolos" al ser descubiertos por promotores, sin necesidad de anunciarse por la ciudad o yendo de puerta en puerta.

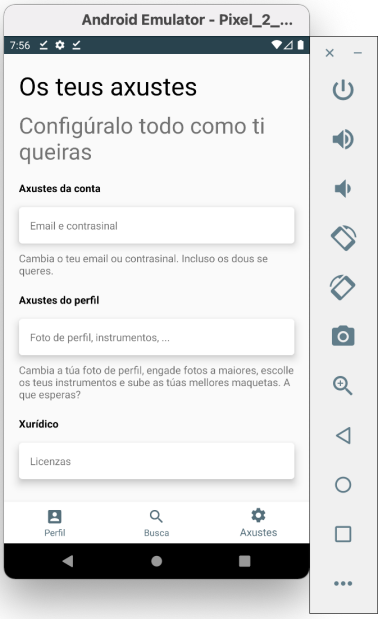
Anexo de figuras

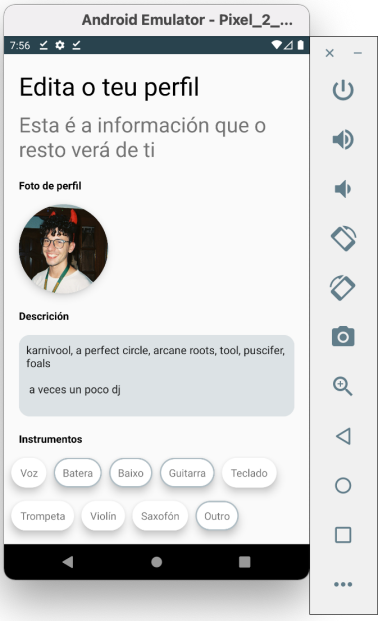


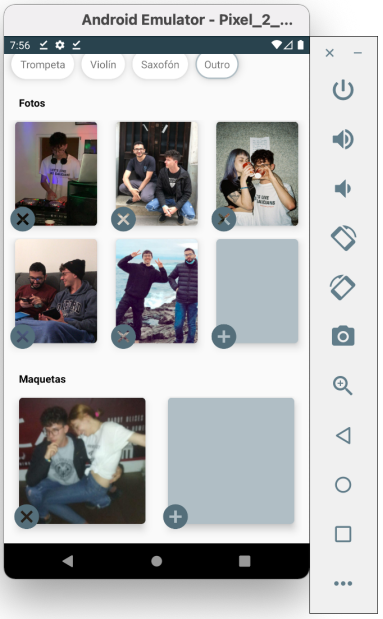



Android Emulator - Pixel_2_...
7:55
Instrumentos
Batera
Baixo
Guitarra
Outro
Fotos
Lorem
Ipsum
Parvadas
Maquetas
Mashup
Perfil
Busca
Axustes

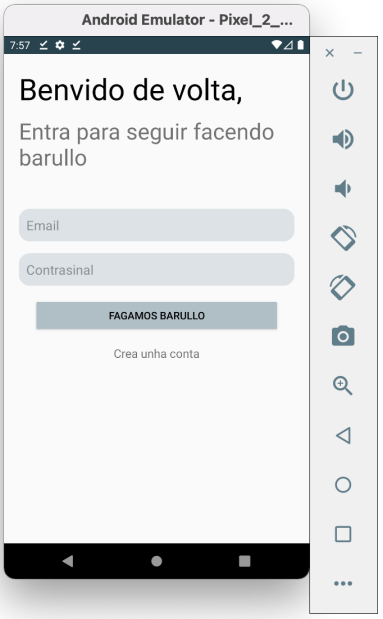

Android Emulator - Pixel_2_...
7:56
Busca un nome
Voz
Batera
Baixo
Guitarra
Teclado
John Wick
johnwick@gmail.com
Santiago Millán González
probando@gmail.com
Lorem ipsum
lorem@ipsum.dolor
Bilbo Bolsón
bilbo@gmail.com
Iñaki Amatria Barral
amatriainaki@gmail.es
Perfil
Busca
Axustes


Android Emulator - Pixel_2_...
7:56
Os teus axustes
Configúralo todo como ti queiras
Axustes da conta
Email e contrasinal
Cambia o teu email ou contrasinal. Incluso os dous se queres.
Axustes do perfil
Foto de perfil, instrumentos, ...
Cambia a túa foto de perfil, engade fotos a maiores, escolle os teus instrumentos e sube as túas mellores maquetas. A que esperas?
Xurídico
Licenzas
Perfil
Busca
Axustes


Android Emulator - Pixel_2_...
7:56
Edita o teu perfil
Esta é a información que o resto verá de ti
Foto de perfil
Descrición
karnivool, a perfect circle, arcane roots, tool, puscifer, foals
a veces un poco dj
Instrumentos
Voz
Batera
Baixo
Guitarra
Teclado
Trompeta
Violín
Saxofón
Outro


Android Emulator - Pixel_2_...
7:56
Trompeta
Violín
Saxofón
Outro
Fotos
Maquetas


Android Emulator - Pixel_2_...
7:57
Comeza a facer o teu barullo,
Rexístrate agora!
Nome completo
Email
Contrasinal
Número móbil
REXÍSTRAME
Xa estou rexistrado!


Android Emulator - Pixel_2_...
7:57
Benvido de volta,
Entra para seguir facendo barullo
Email
Contrasinal
FAGAMOS BARULLO
Crea unha conta