Urband

Práctica 4

Sprint 1, 2 y 3

5/05/2020

Oriol Vivó, Iván Méndez, Èlia Isart, Antoni Carol

CONTENIDO

Sprint 1

Revisión del trabajo a realizar y elementos a implementar en el Sprint 1

En este sprint, el primero, hemos establecido diferentes tareas a realizar durante estas semanas en la que se trabajará.

- **Tarea 1 – Registro de Usuario:** implementar el sistema de registro.
- **Tarea 2 – Inicio de sesión:** implementar el sistema de inicio de sesión.
- **Tarea 3 – Bases del perfil de usuario:** implementar la base del perfil.+

Para la buena revisión del trabajo a implementar, y una buena gestión, evolución y trazabilidad de los requisitos definidos en el documento de la Ingeniería de Requisitos, vamos a definir una estructura a seguir para cada tarea y su documentación.

Formato de cada la documentación de cada tarea:

1. Introducción : introduciremos la tarea, especificaremos en sus diferentes partes, nuestro objetivo, etc.

2. Requisitos y user-stories: citaremos los requisitos y las user-stories creadas para que la aplicación cumpla estos requisitos, la prioridad de estos requisitos, la complejidad, etc.

3. Gestión y Evolución: se mostrarán los cambios en los requisitos ocasionados por diferentes decisiones tomadas en el desarrollo, cambios de estado, tests desarrollados y resultados

El requisito que atacaremos en este sprint para su validación o actualización, serán el requisito 1.

1.El sistema debe tener un método para que los usuarios se registren. Urband
Necesitamos diferentes pantallas, una para introducir de registro del usuario, otra para entrar (loguearse) en la aplicación, con los datos del registro, y una pantalla de validación de registro

Se trata de unos de los requisitos más importantes del sistema ya que sin un método de identificación de los usuarios, ninguna otra función del sistema podría llegar a ser desarrollada.

TAREA 1 – Implementar el sistema de registro de usuario

La primera tarea, es implementar el sistema de registro de usuario, que tiene que permitir al usuario crear su cuenta en la aplicación.

En esta tarea vamos a buscamos el requisito número 1.1 y sus descendientes. En este Sprint 1, no se finalizará debido a la complejidad de integrar otros sistemas, así encontrarán las funcionalidades más básicas.

Requisitos

En esta tarea, abordaremos el requisito **1.1**, asociado con el registro del usuario, incluyendo sus requisitos "hijos" de sistema.

1.1 El usuario necesita una pantalla dónde registrarse con sus datos Urband

1.1.1 Pantalla de Registro :

- **1.1.1.1** La contraseña tendrá como mínimo 8 caracteres, incluyendo una mayúscula, un dígito y un carácter especial
- **1.1.1.2** El correo y el teléfono tienen que ser existentes, sino saltará un error, con una notificación "Toast".
- **1.1.1.3** El botón "Sign up" te redirige a la pantalla de validación, si todo fue correcto, y sinó saltará una notificación de error.

Este requisito es uno de los más prioritarios de sistema y de usuario. Debemos verificar, o actualizar este requisito con tal de el buen funcionamiento del sistema.

Por eso, a raíz de esta prioridad, nace un nuevo requisito referido a la base de datos integrada en el sistema para el almacenamiento de datos dinámico. Este se define a continuación en la evolución y gestión de los requisitos,

En el apartado anterior, hemos citado los requisitos adheridos a nuestras tareas y los hemos clasificados según su prioridad.

En este apartado veremos los cambios que decidimos hacer durante el proceso de este Sprint, la evolución de los requisitos y su gestión usando la matriz de trazabilidad definida en el documento de Ingeniería de Requisitos.

Al requisito 1.1, se le añadirá un requisito de sistema, referido a la base de datos. Necesitamos guardar el registro del usuario para que se pueda logear a continuación.

Urband

- ...
- 1.1.1.3 El botón "Sign up" te redirige a la pantalla de validación, si todo fue correcto, y sinó saltará una notificación de error.

++ Requisito añadido

- ***1.1.1.4 El sistema debe guardar los datos de registro del usuario, para poder redirigirlo al logeo de usuario.***

Comentar, que en su momento pensamos en que tendríamos que usar una base de datos para guardar la información del usuario, registro y perfil, pero decidimos dejar pasar el tiempo hasta que se hiciera realidad, para añadir este requisito.

Urband

18. Nuestro sistema debe integrar una base de datos donde consultar la información sobre usuarios u otro tipo de información dinámica.

Requisito del sistema

18.1 Base de datos

- **18.1.1** La base de datos será de tipo relacional, MySQL.
- **18.1.2** Usaremos la API de retrofit de android, para hacer esta conexión entre dispositivo y base de datos.
- **18.1.3** El lenguaje de el código de

Este requisito es muy necesario, ya que toda la interacción y el almacenamiento de datos se produce en la base de datos integrada. Por eso, el usuario siempre necesitará tener conexión a internet, entrando en juego el requisito 3.

Urband

3.El usuario en todo momento debe disponer de conexión a Internet

Trazabilidad

En cada tarea vamos a realizar una trazabilidad de los requisitos anidados a estas tareas. Usaremos la matriz de trazabilidad definida en el documento de Ingeniería de Requisitos.

Para la trazabilidad más específica, de cada requisito hijo, vamos a mostrar una fracción de la matriz para ver qué tests que definimos más abajo, nos ayudarán a actualizar el estado de este requisito buscando dejarlo verificado, juntamente con su complejidad y su prioridad para ser solucionado.

Requisito	Prioridad (1-5)	Complejidad (10-1)	Test Usado
1.1.1.1	2	7	Test_1
1.1.1.2	2	4	Test_1 --- Test2
1.1.1.3	1	3	Test_3 -- Test2 -- Test_1
1.1.1.4	1	8	TESTING POSTMAN , differente test

Refiriéndonos al requisito 3 y el nuevo, 18, aún no se testean o se comprobarán, debido a que simulamos un servidor en local para hacer las implementaciones.

Después de definir los tests que van a comprobar estos requisitos, definido en definición y realización de pruebas unitarias, vamos a ver como nuestra matriz de trazabilidad de requisitos se va construyendo con el paso de los sprints. Esta matriz de trazabilidad, está definida y explicada detalladamente en el Documento de Aspectos Introductorios.

Urband	IDENTIFICACION		ESTADO			CARACTERISTICAS		OBJETIVO Y VALIDACION		
23/04/2020	Id	Tipo	Version	Estado	Fecha de estado	Prioridad	Complejidad	Objetivo	Test usado	Resultados test
	1.1.1	SISTMA	0.1	Developement	07/03/2020	1	2	Contraseña	Unit Test 1	Cool - Comprobado
	1.1.2	SISTEMA	0.1	Developement	08/03/2020	1	4	Telefono existe?	Unit Test 2	Cool - Comprobado
	1.1.3	SISTEMA	0.1	Developement	08/03/2020	1	8	Registro Completo	Unit Tests 1 , 2	Cool - Comprobado
	1.1.1.4	BASE DATOS	0.1	Developement	10/03/2020	1	9	Datos guatdados en DB?	Postman Tests, Unit Test 3	Cool - Comprobado
	1.1. 1	Usuario	1	Developement	11/03/2020	1	8	Pantalla Registro	Instrumented Tests	Achieved objectives.
	1.1	Usuario	1	DONE	12/03/2020	1	9	Registro de usuario	All sons tests	Cheched, Achieved.

Al finalizar este proceso, vemos que el requisito 1.1, hijo del requisito 1, pasaría a estar validado, mediante los tests definidos para comprobar su buena funcionalidad.

TAREA 2 - Login de usuario

En esta tarea, queremos a partir de los datos del registro del usuario, queremos autenticar a este usuario mediante su nombre de usuario y la contraseña.

No solo se trata de recoger estos datos y inciar la sesión del usuario, sino que incorpora un background complicado que debemos tratar para que no existan conflictos entre usuarios.

Requisitos

1.2 El usuario necesita una pantalla donde poder iniciar sesión con sus datos Urband

1.2.1 Pantalla de Logeo : Un formulario con dos campos, el identificador, telèfono o correo electrónico, y su contraseña. Contendrá un botón para avanzar a la siguiente pantalla.

- - **1.2.1.1** La cuenta tiene que estar creada sinó no se producirá la autenticación de la cuenta, si es el caso, la aplicación mostrará una notificación de error.
- - **1.2.1.2** El botón te redirigirá a la pantalla de inicio de la aplicación si los datos son válidos, mostrará mensaje de error en caso contrario

Como el anterior, este requisito es muy importante, debemos conseguir que el sistema registre y autentifique usuarios de una manera limpia y consistente. En la implementación, estos datos de usuario se consultarán en nuestra base de datos, y si el usuario es correcto, mostrará un mensaje y creará un token de autenticación, para futuros inicios de sesión.

Gestión y Evolución de los Requisitos

En el apartado anterior, hemos citado los requisitos adheridos a nuestras tareas y los hemos clasificados según su prioridad.

En este apartado veremos los cambios que decidimos hacer durante el proceso de este Sprint, la evolución de los requisitos y su gestión usando la matriz de trazabilidad definida en el documento de Ingeniería de Requisitos.

Al empezar esta tarea, vimos la verdadera complicación del sistema de logeo de usuarios, reconocer que usuario se está intentando lograr, y crear un token de autentificación para saber en todo momento del sistema que ese usuario en concreto está usando la aplicación.

Este paso en la aplicación es súper importante, ya que sin él los usuarios no podrán usar la aplicación como es debido.

A raíz de eso y de no haber pensado en ello al definir los requisitos, nacen estos 2 requisitos de sistema, referentes al logeo de usuario.

- **1.2.1.3** El sistema debe autentificar el usuario logeado para poder crear un token de validación de usuario
- **1.2.1.4** El sistema debe guardar este token creado en el dispositivo, permitiendo iniciar sesión directamente usando el token
- **1.2.1.5** El sistema debe borrar el toquen de autenticación cuando el usuario salga de su cuenta mediante el menú

Trazabilidad

En cada tarea vamos a realizar una trazabilidad de los requisitos anidados a estas tareas. Usaremos la matriz de trazabilidad definida en el documento de Ingeniería de Requisitos.

Para la trazabilidad más específica, de cada requisito hijo, vamos a mostrar una fracción de la matriz para ver qué tests que definimos más abajo, nos ayudarán a actualizar el estado de este requisito buscando dejarlo verificado, juntamente con su complejidad y su prioridad para ser solucionado.

Requisito	Prioridad (1-5)	Complejidad (10-1)	Test Usado
1.2.1.1	2	6	Test 4, postman Test 1
1.2.1.2	4	2	Test 4, Test mensajes err
1.2.1.3	2	7	Test 4, Postman Test 2,3
1.2.1.4	2	7	Test 4, Test Shared Pref
1.2.1.5	4	6	Test 5, Postman test 4

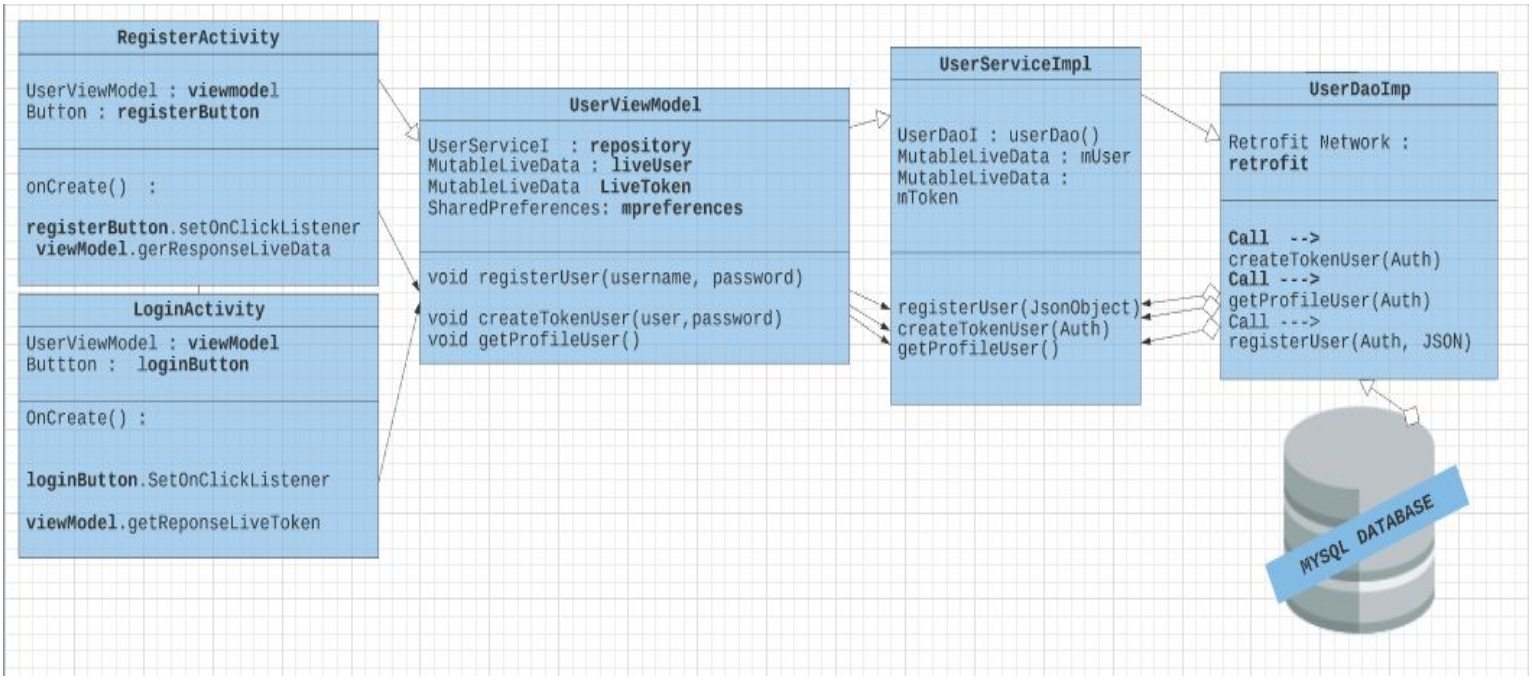
Urband	IDENTIFICACION		ESTADO		CARACTERISTICAS		OBJETIVO Y VALIDACIÓN		
	Id	Tipo	Versión	Estado	Fecha de estado	Prioridad	Complejidad	Objetivo	Test usado	Resultados test
	1.2.1.1	Sistema	0.1	Developement	20/03/2020	1	3	Comprobar usuario	Test 4	Cool- Comparbado
	1.2.1.2	Sistema	0.1	Developement	22/03/2020	2	4	Mensaje de error	Instrumented test 1	Cool- Comparbado
	1.2.1.3	Sistema	0.1	Developement	27/03/2020	1	7	Crear Token en DB	Postman 1, Test 4	Cool- Comparbado
	1.2.1.4	Sistema	0.1	Developement	27/03/2020	1	6	Guardar Token en disp	Shared pref Test	Cool- Comparbado
	1.2.1.5	Sistema	0.1	Developement	02/04/2020	3	4	Borrar Token en DB y disp	Postman 2, Test4	Cool- Comparbado
	1.2. 1	Usuario	0.1	DONE	07/03/2020	2	2	Pantalla Logeo	Instrumented test	Cool- Comparbado
	1.2	Usuario	1	DONE	12/03/2020	1	3	Logeo de Usuario	All sons tests	Achieved!

Diagrama de clases

El diagrama de classes tarta de indicar la comunicación entre clases en el sistema. En nuestro caso, al aplicar en Android el Modelo Vista Vista Modelo (MVVM), tenemos la comunicación entre componentes muy clara.

Para hacer más entendedor este diagrama de clases vamos a citar, describir los diferentes 4 bloques que se comunican en el sistema

- **Vista** : interfaç del usuario, lo que visualiza.
- **ViewModel** : se encarga de comunicar al repositorio las acciones que el usuario hace en las vistas.
- **Repositorio :** Capa dónde hacemos las llamada a la base de Datos (DAO), y la traducción de esta información a nuestros modelos.
- **Dao**: Encargado de hacer las llamadas a la API y retornar el valor a la capa Repositorio para el tratamiento de la Llamada a la api (Call<>)



Definición y realización de pruebas unitarias

Para realizar las pruebas unitarias que consideramos que son necesarias para comprobar la funcionalidad principal de nuestra aplicación, seguiremos siempre una misma estructura para definir y organizar los tests:

Nombre de la tarea:

- **1r Test:**
  -Título del test: Nombre que le proporcionamos al test.
  -Descripción y objetivos: Analizamos y explicamos el motivo del test.
  -Flujo/contenido: Este apartado lo encontraremos en los casos de tests instrumentados.
  -Resultados: Analizaremos los resultados para sacar las conclusiones adecuadas.

Tarea 1 – Registro de Usuario
Título: IsValidpassword
Descripción y objetivos: El objetivo de este test es comprobar que los requisitos de la contraseña son válidos: - Comprobamos que la contraseña tiene como mínimo 8 caracteres. - Comprobamos que la contraseña tiene como mínimo 1 número. - Comprobamos que la contraseña tiene como mínimo 1 mayúscula. Comprobamos que la contraseña cumple el patrón que especifica los requisitos que debe cumplir para que sea válida: "^(?=.*[0-9])(?=.*[A-Z])(?=\\S+$).{8,}$". Utilizamos AssertTrue para entrar una contraseña que cumple con los requisitos correctos: “Igualada88”.
Resultados: El test ha sido implementado y valida que se cumplen los requisitos de la contraseña.

Para aumentar la fiabilidad del test, creamos otro llamado Is_not_valid_password donde en este caso introducimos una contraseña que no cumple las características. El test está implementado y valida que la contraseña no es correcta.

Tarea 1 – Registro de Usuario
Título: IsValidPhone
Descripción y objetivos: El objetivo de este test es comprobar que el teléfono introducido es válido: - Comprobamos que el teléfono contiene exactamente 9 números. Comprobamos que el teléfono cumple el patrón que especifica los requisitos que debe cumplir para que sea válido: "^(?=.*[0-9]).{9,9}$" Utilizamos AssertTrue para entrar un teléfono que cumple con los requisitos correctos: "607689879".
Resultados: El test ha sido implementado y valida que se cumplen los requisitos de la contraseña.

Para aumentar la fiabilidad del test, creamos otro llamado Is_not_valid_phone donde en este caso introducimos un teléfono que no cumple las características. El test está implementado y valida que el teléfono no es correcto.

Tarea 1 – Registro de Usuario
Título: SendOK
Descripción y objetivos: El objetivo de este test es comprobar que se envían los datos del registro correctamente a la Api.
Resultados: El test se encuentra en proceso y valida los datos en la Api.

Tarea 1 – Registro de Usuario
Título: Check_correct_register
Descripción y objetivos: El objetivo de este test es comprobar que el registro del usuario es correcto: - Comprobamos si el usuario y la contraseña son correctos.
Resultados: El test se encuentra en proceso y valida que el usuario y la contraseña introducidos son correctos.

Tarea 2 – Login de Usuario
Título: Check_correct_login
Descripción y objetivos: El objetivo de este test es comprobar que el login del usuario se lleva a cabo correctamente: - Comprobamos que se crea el token de usuario y que nos devuelve el perfil de usuario. .
Resultados: El test se encuentra en proceso. Valida que se crea el token correctamente .y que se devuelve su perfil.

Tarea 3 – Bases del perfil: Esta tarea no dispone de ninguna prueba unitaria ya que con los tests de Tarea 1 y Tarea 2 ya validamos la base del perfil (usuarios correctos).

Pruebas Postman (API)

Para comprobar que la base de datos funciona correctamente y realiza siempre las operaciones esperadas adecuadamente, realizamos los siguientes tests:

Tarea 1

- Test 1: Este test se encarga de comprobar que se crea el usuario correctamente en la base de datos. Llamamos a la ruta donde nos registramos y enviamos el Json del usuario.
- Test 2: Este test se encarga de comprobar que se crea el token del usuario correctamente.
- Test 3: Este test se encarga de comprobar que se elimina el token del usuario correctamente.

La tarea 2 no dispone de ningún test ya que con los que realizamos en la primera tarea es suficiente.

User stories

1. **El usuario debe poder abrir la aplicación y recibir feedback de su correcto funcionamiento**

    a. Se presentará al usuario una primera pantalla, con el logo de la app y con la posibilidad de avanzar a un registro o a un login

2. **El usuario debe poder registrarse en nuestra app**

    a. Se Presentará un formulario, donde se pide solo número de teléfono y contraseña, con un botón para completar el registro. Se presentará un Toast que de feedback del estado de la operación

3. **El usuario debe de poder hacer login en nuestra aplicación**

    a. Se Presentará un formulario con nombre de usuario(telf,pass) y un botón para continuar a la Activity por defecto. Se presentará un Toast para comunicar feedback sobre el correcto funcionamiento de su login

Aspectos de calidad software

Planned Software Test

En esta fase del desarrollo, no hay un plan de desarrollo, ni ningún otro tipo de gestión de testing, más allá del debug manual y, de forma ordenada, el testeo manual del correcto funcionamiento de Activity a Activity, función de API a función de API, al integrar un nuevo commit, por cualquiera de los 4 integrantes del grupo. Cada integrante se ocupa de revisar el ciclo de vida usual de la app al añadir cualquier "feature" a la rama principal de desarrollo.

Revisión e inspección

Después de cada commit, el jefe de programación, en este caso, Iván, revisa el código minuciosamente de forma manual.Se deben seguir una serie de buenas practicas a lo largo de todo el desarrollo que se definen aquí, y no cambian en el transcurso de los diferentes sprints, estas son:

- Activity: Toda pantalla conocidos como activity siempre será precedida de un nombre que inicie en mayuscula tambien, ejemplo: MusicoActivity, LoginActivity, etc..
- Clases: deberán seguir la convención camelCase en el nombramiento de las clases, y serán descriptivas con el carácter "I" en tanto a si son interfaces, o "Impl" si implementa la interfaz y, en el caso del modelo de datos, acabaran

con la "source" de la que proviene el modelo. Ejemplos: UserServiceI, UserDAO, UserDaoImpl, etc…
- Métodos: también siguen la convención camelCase, precedido de get o set si setean o presentan un atributo.
- ViewModel: Cada método que retorne cualquier tipo de dato, tendrá este tipo de dato como prefijo, Ejemplo: LiveDataGetUsuarios…
- Control de errores: Cada Try/Catch, imprime, aunque sea minimamente por pantalla, el mensaje de la excepción capturada, su tipo de excepción, y su causa.
- Routing: En las rutas que se declaran en la api, para las llamadas por Retrofit, no se usan camel cae, sino underscore, por ejemplo, /show_all_users.
- Inicialización de variables: Se inician a null en la medida de lo posible, y solo se le da valor cuando se vayan a usar.

Review de documentación

El encargado de la documentación, en nuestro caso Oriol, a medida que se avanza la documentación, de forma individual por cada integrante, va corrigiendo los errores de formato. Acto seguido, Toni, revisa el contenido, asegurándose de que compla los objetivos para ese documento, y que no falte ningún detalle. Finalmente, se hace una última inspección cuando todos los integrantes asegiran haver cumnplido su división del trabao, conjuntamente por Toni e Iván.

Plantillas

En este sprint, se produce la plantilla que se usará durante todo el desarrollo del proyecto, por una parte, encabezados, pies de página, anotaciones, formato de fechas, colores,etc… Por otra parte, y siendo más específicos, se ha desarrollado una plantilla específica para la documentación de los Sprints, con las tareas individuales marcadas, por secciones. Llevado a cabo por parte de Oriol, quien tiene rol de documentador, y siendo la plantilla que forma ahora mismo este documento.

Actas de las reuniones (Scrums)

Durante el periodo del primer Sprint no ha habido reuniones entre los integrantes del grupo que hayan sido documentadas.

Principales resultados

Se ha completado la funcionalidad básica del sistema de registro, aunque su resolución no ha estado perfecta.

- Pantalla de registro -> **hecho**
- 1.1.1.1 La contraseña tendrá como mínimo 8 caracteres, incluyendo una mayúscula, un dígito y un carácter especial -> **hecho**
- 1.1.1.2 El correo y el teléfono tienen que ser existentes, sino saltará un error, con una notificación "Toast". -> **hecho**
- 1.1.1.3 El botón "Sign up" te redirige a la pantalla de validación, si todo fue correcto, y sinó saltará una notificación de error. -> **hecho**

Problemas/comentarios encontrados al finalizar el sprint:

Login:

- Se puede acceder a la App sin poner información en los campos a rellenar
- No hay notificaciones para que el usuario sepa que pasa en la App en cada momento

Perfil:

- No hay notificaciones para que el usuario sepa que pasa en la App en cada momento
- Se tiene que eliminar el token de la base de datos
- El botón back vuelve a pedir que se haga login, pero tendría que cerrar la App si el usuario ya está logueado.

Registro:

- No hay notificaciones para que el usuario sepa que pasa en la App en cada momento
- Se tiene que volver a la pantalla de login después de hacer un registro correcto.

Calidad:

- El repositorio tendría que ser el que gestiona las SharedPreferences, no las actividades y acceder con el VM.
- Unificar la clase que hace los call.enqueue
- Todos los strings tienen que estar en el fichero strings.xml
- Hay warnings y partes de código que no son usadas.
- Definir las constantes para ser leídas y así no estar hardcodeadas en el código

Otros:

- Error en la comparación de strings
- La pantalla tiene que estar bloqueada al girar, no horizontal.

Sprint 2

Revisión del trabajo a realizar y elementos a implementar en el Sprint 2

En este sprint, hemos establecido diferentes tareas a realizar durante estas semanas en la que se trabajará.

- **Tarea 1- Instanciación del perfil de usuario**
- **Tarea 2-Geolocalización**

Para la buena revisión del trabajo a implementar, y una buena gestión, evolución y trazabilidad de los requisitos definidos en el documento de la Ingeniería de Requisitos, vamos a definir una estructura a seguir para cada tarea y su documentación.

Formato de cada la documentación de cada tarea:

1. Introducción : introduciremos la tarea, especificaremos en sus diferentes partes, nuestro objetivo, etc.

2. Requisitos y user-stories: citaremos los requisitos y las user-stories creadas para que la aplicación cumpla estos requisitos, la prioridad de estos requisitos, la complejidad, etc.

3. Gestión y Evolución: se mostrarán los cambios en los requisitos ocasionados por diferentes decisiones tomadas en el desarrollo, cambios de estado, tests desarrollados y resultados,...

Al definir, documentar, y estas tareas y sus requisitos, veremos la matriz de Trazabilidad de Requisitos, qué requisitos ya están satisfechos, cuales en proceso, durante todo el tiempo de trabajo del Sprint.

Ya que el Sprint 3 es una consolidación del Sprint 2, la matriz de trazabilidad se verá actualizada en su documentación, referenciando todos los requisitos del Sprint 2.

TAREA 1 – Instanciación del perfil de usuario

En esta tarea, vamos a buscar terminar el perfil de usuario. Esto significa que desde un inicio, el usuario al loguearse por primera vez, deba configurar su perfil de usuario.

La información del perfil se divide en 4 grandes grupos

- *Información Personal* – Contiene los campos como el nombre y apellidos del usuario, su ubicación, foto de perfil… Información esencial para identificarte.
- *Tabla Usuario-Instrumentos* – Contiene cada instrumento que el usuario ha decidido añadir, con su nombre, un icono y la experiencia en cada fila de la tabla.

Requisitos

En esta tarea buscamos satisfacer uno de los requisitos de usuario más importantes y prioritarios, también uno de los más complejos a nivel de su división en muchos requisitos de sistema.

2. El usuario necesita un perfil para identificarse	Urband

A raíz de este requisito, con una prioridad máxima (1) , definimos diferentes "user-stories" para satisfacer y actualizar estos requisitos, para su futura validación.

USER-STORIES
1-2-3-4-5-7-8

Gestión y Evolución de los Requisitos

En el apartado anterior, hemos citado los requisitos adheridos a nuestras tareas y los hemos clasificados según su prioridad.
En este apartado veremos los cambios que decidimos hacer durante el proceso de este Sprint, la evolución de los requisitos y su gestión usando la matriz de trazabilidad definida en el documento de Ingeniería de Requisitos.

Al definir el requisito 2, no pensamos en que en esta etapa, lo re definiríamos de completo. Adaptamos este requisito a nuestras necesidades en android, así pudiendo comprobarlos mucho más fácilmente.

++Evolución Urband
2.1.2 Modificación del perfil: *dividimos la configuración del perfil en 3 etapas*
- *2.1.2.1 Dialogo con los datos más personales del usuario*
  - *Nombre completo, fecha de nacimiento, género, experiènica general en la música, foto de perfil y una pequeña descripción.*

- *2.1.2.2 Dialogo con las habilidades músicales, instrumentos, etc.*
  - *El sistema debe mostrar una tabla dinámica que se actualice a medida que el usuario añade o borra instrumentos. Incorporando elementos visuales para añadir o eliminar elementos*

- *2.1.2.3 Dialogo con géneros musicales favoritos o identificativos.*
  - *El sistema debe mostrar una tabla dinámica que se actualice a medida que el usuario añade o borra géneros musicales. Incorporando elementos visuales para añadir o eliminar géneros*

El requisito 2.1.3, y el referente en el 2.1.2.1, editar la foto de perfil, es un requisito que se incorporará en sprints futuros, debido a su complejidad y a su prioridad no muy elevada.

2.1.3 El usuario debe aceptar unos permisos para poder subir imágenes desde su teléfono. Urband

Trazabilidad

En cada tarea vamos a realizar una trazabilidad de los requisitos anidados a estas tareas. Usaremos la matriz de trazabilidad definida en el documento de Ingeniería de Requisitos.

Nos centraremos en la primera parte del seteo de perfil, y los instrumentos. Dejando los géneros musicales para un posterior sprint 3 de consolidación.

Requisito	Prioridad (1-5)	Complejidad (10-1)	Test Usado
2.1.2.1	2	4	Tests First step, Chech_Fille, Chech_Data
2.1.2.2	1	7	Tests Instrumentos, Add, Remove, Get

TAREA 2- Geolocalización

En esta tarea, vamos a implementar el sistema de el mapa, conectado con el gps del usuario. Este elemento se mostrará en la Default Activity, la actividad por defecto, visible en todo momento por el usuario.

Presenta características como la posición aproximada, nunca exacta por temas de privacidad, de los usuarios en línea o usuarios registrados activos en la aplicación.

Requisitos

En esta tarea, atacamos mayoritariamente al requisito 4, buscaremos completarlo. También vamos por una parte, el requisito número 10, ya que hace referència al elemento visual que queremos implementar en este sprint.

4.Al usuario se le pedirán permisos para acceder a su ubicación, Gps. Urband
4.1 La aplicación pedirá los permisos al usuario por primera vez cuando rellene el campo obligatorio de dónde se localiza. En caso de no aceptar los permisos, el perfil queda incompleto pero el usuario podrá utilizar la aplicación pero con funciones restringidas.

10.El usuario debe ser capaz de inspeccionar y encontrar eventos cercanos Urband **a él.**
10.1 La aplicación tendrá la opción de poder buscar eventos, éstos aparecerán en un mapa en que los diferentes eventos programados estarán marcados con un marker que estará resaltado para que los usuarios lo puedan pulsar fácilmente. - **10.1.1 Mapa** : se podrá elegir el tipo de mapa que queremos visualizar (satélite, mapa, relieve), éste se podrá ampliar o alejar, dependiendo la zona en que quiera buscar el usuario. - **10.1.2 Marker** : éste elemento permitirá que los usuarios vean situado en el mapa a los otros usuarios. Al pulsar el marker aparecerá el perfil del usuario, en él habrá un resumen de ese usuario, si queremos más información del usuario buscado - **10.1.3 Filtros** : en la parte superior derecha del mapa habrá el botón de filtros. - **10.1.4 Músicos, grupos o conciertos :** en la parte inferior del mapa habrá la opción de poder elegir que lo que vemos marcado en el mapa pueden ser o conciertos, músicos o grupos.

Para la validación de este requisito intervienen los siguientes user-stories:

USER-STORIES
10-11-12-13

Gestión y evolución de los requisitos

En el apartado anterior, hemos citado los requisitos adheridos a nuestras tareas y los hemos clasificados según su prioridad.
En este apartado veremos los cambios que decidimos hacer durante el proceso de este Sprint, la evolución de los requisitos y su gestión usando la matriz de trazabilidad definida en el documento de Ingeniería de Requisitos.

En el caso del requisito 10, dirigido a poder ver eventos en un mapa, se ha reformado, generalizando más es un su objetivo; uniendo eventos y usuarios dentro de un mapa.

10.El usuario debe poder visualizar un mapa en la aplicación que le ofrezca interacciones. Urband

10.1 Mapa : se podrá elegir el tipo de mapa que queremos visualizar (satélite, mapa, relieve), éste se podrá ampliar o alejar, dependiendo la zona en que quiera buscar el usuario.
- **10.1.1** La aplicación tendrá la opción de poder buscar usuarios cercanos en el mapa, estos aparecerán con un marker acompañado de un icono para su clara interpretación.
  - **10.1.1.1 Marker** : éste elemento permitirá que los usuarios vean situado en el mapa a los otros usuarios. Al pulsar el marker aparecerá el perfil del usuario, en él habrá un resumen de ese usuario, si queremos más información del usuario buscado.
- **10.1.2** La aplicación tendrá la opción de poder buscar eventos, éstos aparecerán en un mapa en que los diferentes eventos programados estarán marcados con un marker que estará resaltado para que los usuarios lo puedan pulsar fácilmente.
  - **10.1.2.1 Músicos, grupos o conciertos :** en la parte inferior del mapa habrá la opción de poder elegir que lo que vemos marcado en el mapa pueden ser o conciertos, músicos o grupos.
- **10.1.3 Filtros** : en la parte superior derecha del mapa habrá el botón de filtros.

Trazabilidad

En cada tarea vamos a realizar una trazabilidad de los requisitos anidados a estas tareas. Usaremos la matriz de trazabilidad definida en el documento de Ingeniería de Requisitos.

En esta tarea, nos vamos a centrar en el requisito 10.1 y sus hijos.

Vamos a buscar completar estos para dejar la parte de usuarios en el mapa avanzada, creyendo que es una de las funcionalidades más importantes de la aplicación.

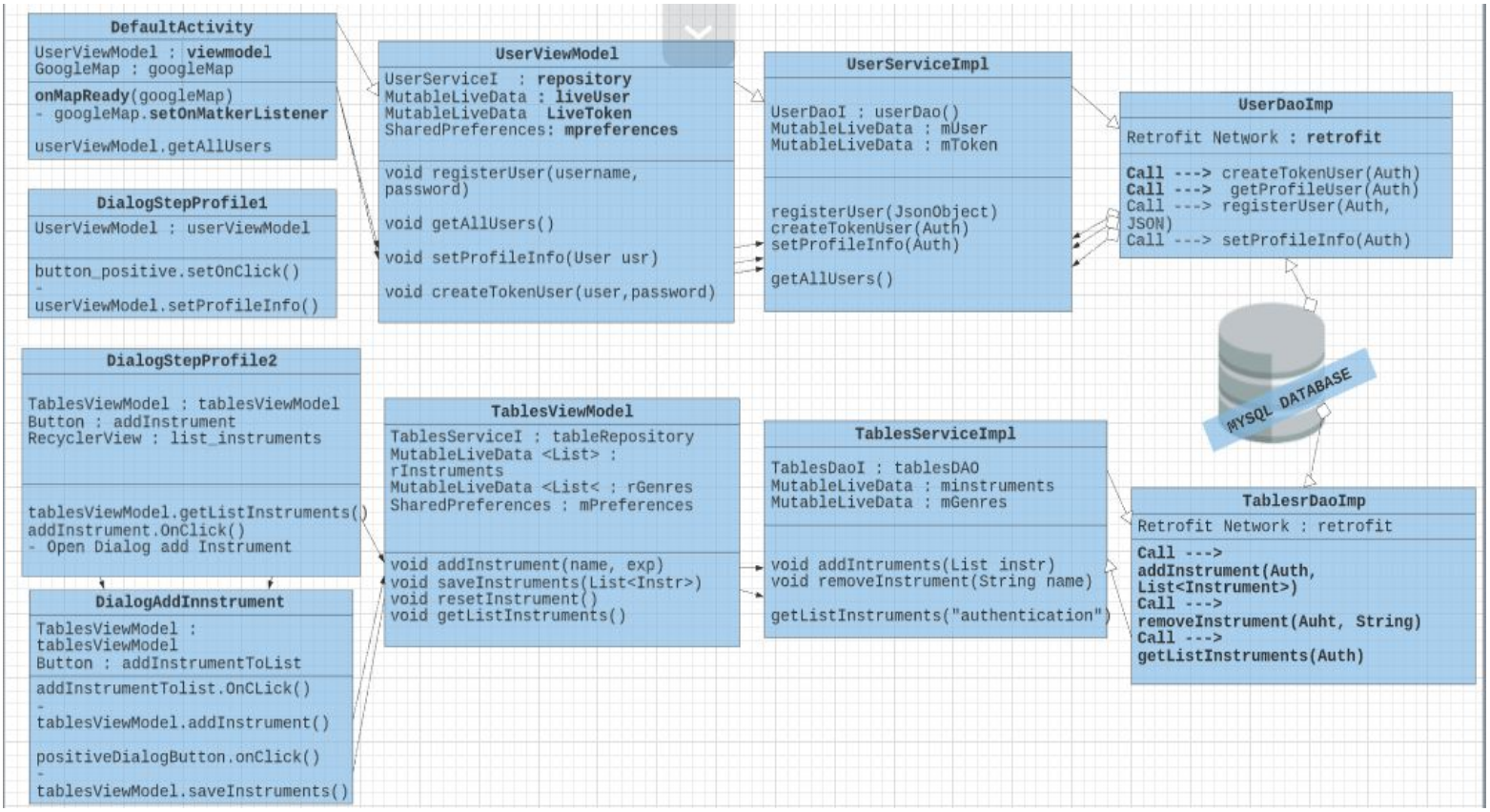
Requisito	Prioridad (1-5)	Complejidad (10-1)	Test Usado
10.1.1	1	8	Test 1 Geolocalización , test 2 Geolocalización
10.1.1.1	2	• 9	Test 3 Geolocalización, Test 1 Geolocalización , test 2 Geolocalización

Diagrama de clases

El diagrama de classes tarta de indicar la comunicación entre clases en el sistema. En nuestro caso, al aplicar en Android el Modelo Vista Vista Modelo (MVVM), tenemos la comunicación entre componentes muy clara.

La estructura del diagrama de clases està definida en el Sprint 1, en este sprint como introducción a su diagrama de clases, veremos principales diferencias:

- Login Activity, Register Activity, no han sido modificadas, por eso no aparecen.
- Nuevo ViewModel para gestionar las tablas de la aplicación, acompañado de su servicio y DAO para hacer y tratar llamadas a la Base de datos



Definición y realización de pruebas unitarias

Tarea 1 – Perfil de usuario
Título: Check_Fields_filled
Descripción y objetivos: El objetivo de este test es comprobar que se han rellenado todos los campos obligatorios del perfil: - Nombre: Comprobamos que el usuario escribe un nombre. - Apellido: Comprobamos que el usuario introduce 1 o 2 apellidos. - Fecha: Comprobamos que el usuario ha introducido una fecha correcta - Género: Comprobamos que el usuario ha seleccionado un género (Radio Button). - Experiencia general: Comprobamos que el usuario ha seleccionado su experiencia en la música (Rating Bar)
Resultados: El test se encuentra en proceso y valida que todos los campos han sido rellenados.

Tarea 1 – Perfil de usuario
Título: CheckData
Descripción y objetivos: El objetivo de este test es comprobar que el formato de la fecha es válido.
Resultados: El test se encuentra en proceso y valida la fecha introducida.

Para comprobar que la tabla usuario-instrumento funciona correctamente, realizamos tres pruebas unitarias que comprueban las tres funciones principales:

Tarea 1 – Perfil de usuario
Título: Check_AddInstruments
Descripción y objetivos: El objetivo de este test es comprobar que la función AddInstruments añade los instrumentos que selecciona el usuario correctamente en la tabla.
Resultados: El test se encuentra en proceso y valida que se han añadido los instrumentos correctamente .

Tarea 1 – Perfil de usuario
Título: Check_DeleteInstruments
Descripción y objetivos: El objetivo de este test es comprobar que la función Delete Instruments elimina los instrumentos que selecciona el usuario de la tabla.
Resultados: El test se encuentra en proceso y valida que se han eliminado los instrumentos correctamente .

Tarea 1 – Perfil de usuario
Título: Check_GetList
Descripción y objetivos: El objetivo de este test es comprobar que la función getlist coge los instrumentos que se encuentran en ese momento en la tabla.
Resultados: El test se encuentra en proceso y valida que se cogen instrumentos correctamente .

Tarea 2 – Geolocalización
Título: Correct_Coordinates
Descripción y objetivos: El objetivo de este test es comprobar que las coordenada se reciben correctamente desde la API.
Resultados: El test se encuentra en proceso y valida la obtención de las coordenadas.

Tarea 2 – Geolocalización
Título: Divide_Coordinates
Descripción y objetivos: El objetivo de este test es comprobar que las coordenadas se separan correctamente en latitud y longitud, separadas por una coma.
Resultados: El test se encuentra en proceso y valida que las coordenadas se han dividido correctamente.

Tarea 2 – Geolocalización
Título: **Real_coordinates**
Descripción y objetivos: El objetivo de este test es comprobar que las coordenadas son correctas una vez separadas en latitud y longitud.
Resultados: El test se encuentra en proceso y valida que las coordenadas introducidas existen.

Pruebas Postman (API)

Tarea 1

- Test 1: Este test se encarga de comprobar que la tabla de instrumentos se actualiza correctamente en la base de datos después de cada operación.
- Test 2: Este test se encarga de comprobar que la información de la tabla de instrumentos se coge correctamente.
- Test 3: Este test se encarga de comprobar que todos los cambios que se realizan en el set Profile Step1 se guardan correctamente en la base de datos.

User stories

1. **El usuario debe ser capaz de elegir foto de perfil**

    a. Implica obtener el permiso de cámara, y presentar al usuario una imagen por defecto, que al clicar abre el selector de imágenes del sistema android, más un botón para guardar la imagen escogida

2. **El usuario debe poder elegir experiencia general**

    a. Se añadirá un dropdown en el perfil de usuario que permite escoger la experiencia que tiene el usuario en el mundo musical.

3. **El usuario debe poder elegir instrumentos**

    a. Se añadirá unos Dropdowns en el perfil de usuario, que irán emparejados dos a dos, uno para el instrumento y otro para la experiencia con ese instrumento

4. **El usuario debe poder elegir su experiencia con cada instrumento en concreto**

    a. Idem "3"

5. **El usuario debe poder elegir los géneros que le interesan**

    a. Mostrarlos gráficamente e implementar handlers en los click. Desplegar en un combobox.

6. **El usuario debe poder logearse como banda.**

    a. Implica Añadir un campo usario/banda/patrocinador a la hora de hacer login, que permite al usuario elegir el rol con el que entra en la aplicación (radio button group)

7. **El usuario debe poder poner una descripción de sí mismo**

    a. Se presentará un textArea en el perfil del usuario que sea capaz de guardar una descripción.

8. **El usuario debe poder guardar sus cambios en el perfil**

a. Implica añadir un botón guardar que modifique y actualice todo el perfil del usuario al apretarlo.

9. **El usuario debe puede loguearse como patrocinador**

a. Idem “6”

10. **El usuario debe poder guardar/editar su geolocalización**

a. Implica pedir el permiso al usuario al iniciar la app, y guardarla una vez se completa el registro.

11. **El usuario debe poder verse geolocalizado en la pantalla por defecto**

a. Implica añadir librería de google maps y generar el mapa gráficamente, situando un marker en la posición del usuario

12. **El usuario ha de poder ver que usuario tiene cerca de sí**

a. Implica añadir markers por cada usuario en la pantalla de google maps.Expansión 11

13. **El usuario ha de poder identificar a los usuarios cercanos**

a. Implica cambiar el marker por defecto en el mapa, por un marker que incluya la información de usuario. Expansión 12

14. **El usuario debe poder tener un menú de logout y de acceso a perfil desde la parte gráfica**

a. Implica crear un menú “deslizante” que aparezca y desaparezca al marcar un icono de “hamburger” como se suele implementar.

Aspectos de calidad software

Planned Software Test

En esta fase del desarrollo se empiezan a diseñar unit tests, no cubren todos los casos, ni empleamos estrategias para asegurar la "coverability" al 100%, sino que seleccionamos funciones críticas, las que pueden dar los fallos más importantes para cada sección y Oriol junto con Toni diseñan una batería de unit tests para cada sección del desarrollo, desde el modelo de datos, hasta la geolocalización.

Revisión e inspección

Después de cada commit, el jefe de programación, en este caso, Iván, revisa el código minuciosamente de forma manual, siguiendo las reglas descritas en el primer sprint.

Backlog

Con el sprint Anterior ya evaluado, el propio profesor, Jordi en nuestro caso, señala unos ciertos errores en el código, la primera tarea del jefe de programación, no es continuar con las user stories que pueda tener asignadas, sino producir "fixes" para este backlog de errores hasta que queda vació.

Actas de las reuniones (Scrums)

Data: 31 de marzo de 2020 a las 17:15h
Asistentes: Toni Carol, Iván Méndez, Èlia Isart, Oriol Vivó
Orden del dia:

- General: Se han presentado las user stories y seguidamente dividido las diferentes tareas entre todos los integrantes del grupo. También se ha acordado una hora concreta para ponernos todos en contacto una vez al dia.

Data: 3 de abril de 2020 a las 17:00h
Asistentes: Toni Carol, Iván Méndez, Èlia Isart, Oriol Vivó
Orden del dia:

- General: Seguimiento de las user stories, exponer dudas a otros integrantes del grupo para resolver, discusión prototipo. División tareas sprint II Innovació.
- Toni: Documentación sobre los permisos de cámara, galería. Arreglar, cambiando nombres, las activities. Proponer nuevas maneras de hacer el código.
- Ivan: Arreglar problemas de programación android. Comentar que la ruta de fichero es diferente para cada uno y no tener problemas al subirlo.
- Oriol: Desplegar Docker y mirar API.
- Èlia: Acabar prototipo. Pasar en un PDF y subir al campus el Lean Canvas.

Data: 6 de abril de 2020 a las 17:00h
Asistentes: Toni Carol, Iván Méndez, Èlia Isart, Oriol Vivó
Orden del dia:

- Toni / Oriol (4-5 abril): Investigación tablas Falcon API. Empezar a setear la configuración del perfil (tabla user-instrument). Creación Viewmodel para las tablas.
- Toni / Oriol: Configurar comunicación MVVM. Mejora de los Layouts en la configuración de perfil.
- Toni: Finalización permisos de la cámara completo. Creación classes para la tabla clients-instrument y guardar estos datos a las shared preferences. Definición de las funciones que llamaran a la API.
- Oriol: Crear y definir campos para la experiencia general.
- Ivan: Investigación de la documentación de la API de Google Maps.
- Èlia: Finalización Lean Canvas y adelantos en el prototipo.

Data: 7 de abril de 2020 a las 17:00h
Asistentes: Toni Carol, Èlia Isart, Oriol Vivó
Orden del dia:

- Toni: Implementar RecyclerView para el Layout.
- Oriol: Desplegar Combobox con estilos de música.

Data: 8 de abril de 2020 a las 17:00h
Asistentes: Toni Carol, Iván Méndez, Èlia Isart, Oriol Vivó
Orden del dia:

- Toni / Oriol: Dejar los Layouts de seteo de perfil y modelo tabla género-user casi acabados.
- Oriol:
- Ivan: Poner el mapa en la activity.
- Èlia: Finalización pantallas de Usuario sin registrar del prototipo y empezar pantallas de Locales/eventos.

Data: 9 de abril de 2020 a las 17:00h
Asistentes: Toni Carol, Iván Méndez, Èlia Isart, Oriol Vivó
Orden del dia:

- Toni: Hacer un push para Oriol. Acabar RecyclerView para después hacer las conexiones con la API.
- Ivan: Añadir en el modelo de la API el campo de geolocalización.
- Èlia: Diseño de las pantallas de Locales/Patrocinadores y empezar a implementarlas.

Data: 20 de abril de 2020 a las 17:00h
Asistentes: Toni Carol, Iván Méndez, Èlia Isart, Oriol Vivó
Orden del dia:

- General: Se han distribuido las diferentes tareas de Innovació entre los integrantes del grupo y hecho un seguimiento de cómo va el Sprint 2 de Android.

Data: 22 de abril de 2020 a las 17:00h
Asistentes: Toni Carol, Èlia Isart, Oriol Vivó
Orden del dia:

- General: Llamada de ayuda y seguimiento de las user stories a entregar ese mismo dia.

Principales resultados

Tarea 1 – Perfil de usuario:

- 2. El usuario necesita un perfil para identificarse -> hecho
- 2.1 El usuario tiene que poder instanciar y modificar su perfil en todo momento -> hecho
    - 2.1.1 Acceso al perfil -> hecho
    - 2.1.2 Pantalla del perfil -> **no hecho**
    - 2.1.3 El usuario debe aceptar unos permisos para poder subir imágenes desde su teléfono -> **no hecho**

Tarea 2 – Geolocalización:

- 4. Al usuario se le pedirán permisos para acceder a su ubicación, Gps -> **hecho**
- **4.1** La aplicación pedirá los permisos al usuario por primera vez cuando rellene el campo obligatorio de dónde se localiza. En caso de no aceptar los permisos, el perfil queda incompleto pero el usuario podrá utilizar la aplicación pero con funciones restringidas -> **hecho**
- 10. El usuario debe ser capaz de inspeccionar y encontrar eventos cercanos a él -> **hecho**
- 10.1 La aplicación tendrá la opción de poder buscar eventos, éstos aparecerán en un mapa en que los diferentes eventos programados estarán marcados con un marker que estará resaltado para que los usuarios lo puedan pulsar fácilmente -> **hecho**
    - 10.1.1 Mapa -> **hecho**
    - 10.1.2 Marker -> **hecho**
    - 10.1.3 Filtros -> **hecho**
    - 10.1.4 Músicos, grupos o conciertos -> **hecho**

<u>Problemas/comentarios encontrados al finalizar el sprint:</u>

Login:

- No funciona al 100% perfecto

Pantalla de perfil:

- Inacabada
- Se tiene que hacer cambios para adaptar a Android

Permisos subir imagenes:

- Está en progreso

Sprint 3

Revisión del trabajo a realizar y elementos a implementar en el Sprint 3

En este sprint, hemos establecido diferentes tareas a realizar durante estas semanas en la que se trabajará.

Tarea 1 – Arreglos y modificación, inicialización del perfil

Tarea 2 – Implementación géneros musicales del usuario

Tarea 3– Inicio lógica suscripciones entre usuarios.

Para la buena revisión del trabajo a implementar, y una buena gestión, evolución y trazabilidad de los requisitos definidos en el documento de la Ingeniería de Requisitos, vamos a definir una estructura a seguir para cada tarea y su documentación.

Formato de cada la documentación de cada tarea:

1. Introducción : introduciremos la tarea, especificaremos en sus diferentes partes, nuestro objetivo, etc.

2. Requisitos y user-stories: citaremos los requisitos y las user-stories creadas para que la aplicación cumpla estos requisitos, la prioridad de estos requisitos, la complejidad, etc.

3. Gestión y Evolución: se mostrarán los cambios en los requisitos ocasionados por diferentes decisiones tomadas en el desarrollo, cambios de estado, tests desarrollados y resultados,...

Al definir, documentar, y estas tareas y sus requisitos, veremos la matriz de Trazabilidad de Requisitos, qué requisitos ya están satisfechos, cuales en proceso, durante todo el tiempo de trabajo del Sprint.

Como se comentó en la documentación del Sprint 2, actualizaremos la matriz de trazabilidad juntando los requisitos de estos 2 sprints.

TAREA 1 – Arreglos y modificación, inicialización del perfil

Debemos solucionar problemas con tal de poder avançar en el diseño e implementación de una manera segura y robusta. Hemos citado algunos de los problemas con los que nos encontramos anteriormente y en esta tarea buscaremos solucionar los errores referido a la API, la base de datos.

Para empezar con esta tarea, citaremos algunos de los errores o problemas más primordiales a solucionar:

- En aspectos de la base de datos
  - Relaciones entre modelo usuario e instrumento
  - Filtraje de usuario para reconocer autentificación.
  - Errores en login; crear token de usuario
  - Estilo sucio, sin seguir una cálida
- En aspectos de Android
  - Errores en la actividad de Login
  - Errores en layouts, vista landscape horizontal
  - Errores en actualizar lista instrumentos.

Requisitos

2. El usuario necesita un perfil para identificarse Urband

Seguiremos con el requisito 2, tratado en la Tarea 2 del sprint 2, para buscar su validación. En este sprint, añadiremos el menú de opciones que contiene los diferentes módulos que el usuario puede usar; con un contenido de momento de solo el perfil.

2.1 El usuario tiene que poder instanciar y modificar su perfil en todo momento. Urband **Requisitos de sistema** **2.1.1 Acceso al perfil :** Si es la primera vez que entras, se abrirá inmediatamente al hacer click en “Sign in”.También se accede desde un botón del menú principal.

En relación al requisito 2.1.1, se ha implementado la lógica que comprueba si es la primera vez que el usuario se logea, si es el caso, el sistema le avisará y le obligará a instanciar su perfil.

Para la validación de este requisito intervienen los siguientes user-stories:

USER-STORIES
1-2-4-5-8-9

Gestión y Evolución de los Requisitos

Como vimos en el anterior sprint, el requisito 2 fue retocado. En este paso, el requisito no evoluciona, pero si se adapta para su mejor entendimiento.

2. El usuario necesita un perfil para identificarse Urband

2.1 El usuario tiene que poder instanciar y modificar su perfil en todo momento.
Requisitos de sistema
2.1.1 Acceso al perfil : El usuario debe poder acceder a su perfil, y editarlo
- 2.1.1.1 Si es la primera vez que se logea, el sistema le dirigirá a inicializar su perfil
- 2.1.1.2 Si el usuario ya tiene el perfil completo, podrá acceder desde el menú de opciones

2.1.2 Modificación del perfil: *dividimos la configuración del perfil en 3 etapas*
- *2.1.2.1 Dialogo con los datos más personales del usuario*
  - *Nombre completo, fecha de nacimiento, género, experiènica general en la música, foto de perfil y una pequeña descripción.*

++Requisito nuevo
- *2.1.2.2 Dialogo con las habilidades músicales, instrumentos, etc.*
  - *El sistema debe mostrar una tabla dinámica que se actualice a medida que el usuario añade o borra instrumentos. Incorporando elementos visuales para añadir o eliminar elementos*

2.1.3 El usuario debe aceptar unos permisos para poder subir imágenes desde su teléfono.
- **2.1.3.1** se le aparecerá una "Toast" o un "popup", donde el usuario deberá aceptar el permiso de que el sistema pueda recoger imágenes del usuario, si no se acepta, el usuario no podrá subir multimedia.

Como en el sprint anterior, el requisito 2.1.3, se va a dejar para posteriores sprints debido a su complejidad y prioridad no tan elevada.

Trazabilidad

En esta etapa, la trazabilidad, comprobaremos de diferentes formas que estos errores se hayan solucionado con el trabajo hecho, se usarán tests muy específicos para comprobar que el sistema notifica correctamente de los errores, y las funcionalidades que no se implementaron del todo correctamente, ahora funcionen sin ningún problema.

Requisito	Prioridad (1-5)	Complejidad (10-1)	Test Usado
2.1.2.1	2	6	Test 1, postman tests
2.1.2.1	2	8	Test 1, postman tests

TAREA 2 - Implementación géneros musicales del usuario

Esta es una tarea que estaba destinada a ser desarrollada y completada en el Sprint 2, pero debido a la falta de tiempo y a diferentes problemas surgidos durante la implementación, se pospuso para este sprint.

Esta tarea es muy parecida a la parte de instrumentos de usuario, siguiendo el mismo patrón en android y uno de muy parecido en la estructura de base de datos.

Requisitos

Seguiremos con el requisito 2, uno de los requisitos más importante y prioritario.

2. El usuario necesita un perfil para identificarse Urband

Para la validación de este requisito intervienen los siguientes user-stories:

USER-STORIES
4

Gestión y Evolución de los Requisitos

El requisito 2, debido a esta nueva implementación, y su anterior evolución, se la añade otro requisito hijo a la parte de modificación de perfil

Urband

- 2.1.2.2 Dialogo con las habilidades músicales, instruementos

- *El sistema debe mostrar una tabla dinámica que se actualice a medida que el usuario añade o borra instrumentos. Incorporando elementos visuales para añadir o eliminar elementos*

##########################Requisito Nuevo

-2.1.2.3 Dialogo con preferencias musicales, géneros musicales

- *El sistema debe mostrar una tabla dinámica que se actualice a medida que el usuario añade o borra Géneros musicales. Incorporando elementos visuales para añadir o eliminar elementos*

Trazabilidad

Requisito	Prioridad (1-5)	Complejidad (10-1)	Test Usado
2.1.2.3	2	8	Test 2, 3 --- Related to Genre -- postman tests

TAREA 3 -Inicio lógica suscripciones entre usuarios

Se trata de un pequeño avançe en esta funcionalidad, dejando preparado todo para su buena implementación.

Requisitos

En esta tarea atacaremos al requisito 7, y a raíz de él nacerá otro requisito nuevo, definido en el siguiente apartado evolución de los requisitos

7. El usuario debe ser capaz de consultar perfiles de otros usuarios Urband

De momento no podemos completar este requisito de ninguna manera, pero lo debemos tener en cuenta, ya que será uno que atacaremos en el siguiente sprint.

Para la validación de este requisito intervienen los siguientes user-stories:

USER-STORIES
6-7

Gestión y evolución de los requisitos

Como hemos citado en el apartado anterior, nace un nuevo requisito a raíz de esta tarea

19. El usuario debe poder suscribirse a otros usuarios para seguir su movimiento en la aplicación Urband
19.1 Se le ofrecerá una opción de seguir al usuario en primera opción al aceptar un match **19.2** Si consulta un perfil ajeno, no match, también se le ofrecerá la opción.

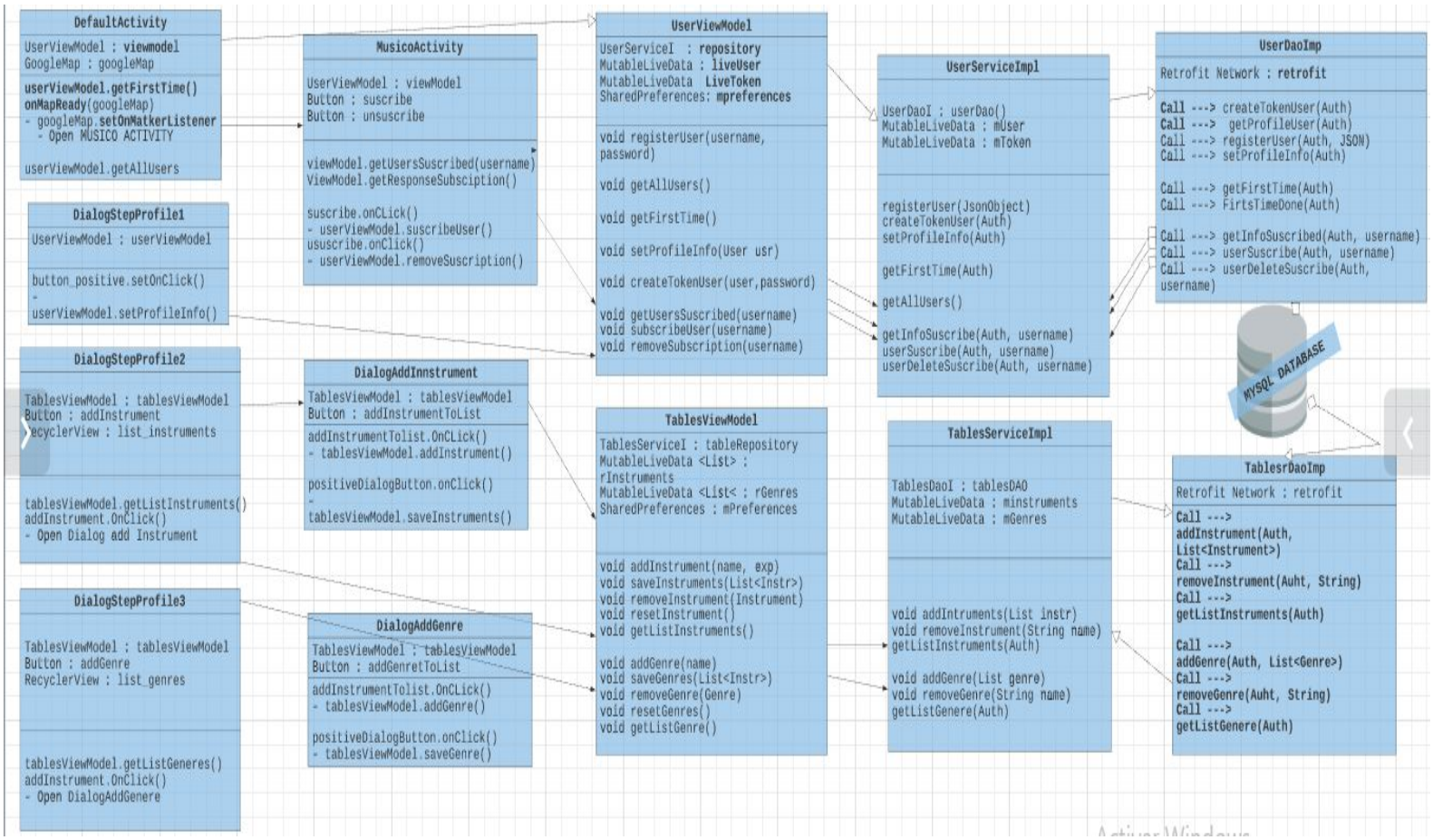
Trazabilidad

Ya que solo definimos la lógica más básica del sistema de suscripciones, la comprobación de esta función y sus requisitos serán comprobados en futuros Sprints, cuando la tarea esté más desarrollada.

Diagrama de clases

El diagrama de classes tarta de indicar la comunicación entre clases en el sistema. En nuestro caso, al aplicar en Android el Modelo Vista Vista Modelo (MVVM), tenemos la comunicación entre componentes muy clara.

Este diagrama de clases varía en muy poco aspectos. Hay cambios como el añadido de nuevas clases, Dialog Step 3, configuración y elección de géneros musicales, y la actividad Músico dónde se encuentra la lógica de las suscripciones.



por problemas con su creación y su exportación, le proporcionamos un enlaçe para que pueda ver el diagrama de clases en mejores condiciones.

https://drive.google.com/drive/folders/13mV1i_vOJEluqCCd2eKNV4s-zA3pOScL

Definición y realización de pruebas unitarias

Tarea
Título: Check_Message_Show
Descripción y objetivos: El objetivo de este test es comprobar que los mensajes de error se muestran correctamente.
Resultados: El test se encuentra en proceso y valida que se muestra el mensaje.

Tarea
Título: Check_AddGenere
Descripción y objetivos: El objetivo de este test es comprobar que la función AddGenere añade los generos que selecciona el usuario correctamente en la tabla.
Resultados: El test se encuentra en proceso y valida que se han añadido los géneros correctamente .

Tarea
Título: Check_DeleteGenere
Descripción y objetivos: El objetivo de este test es comprobar que la función DeleteGenere elimina los generos que selecciona el usuario de la tabla.
Resultados: El test se encuentra en proceso y valida que se han eliminado los géneros correctamente .

Tarea
Título: Check_GetList
Descripción y objetivos: El objetivo de este test es comprobar que la función getlist coge los géneros que se encuentran en ese momento en la tabla.
Resultados: El test se encuentra en proceso y valida que se cogen los géneros correctamente.

Tarea 3
Título: Check_AddSubscription
Descripción y objetivos: El objetivo de este test es comprobar que la función AddSubscription añade la subscripción que se selecciona correctamente.
Resultados: El test se encuentra en proceso y valida que se ha añadido la suscripción correctamente .

Tarea 3
Título: Check_DeleteSubscription
Descripción y objetivos: El objetivo de este test es comprobar que la función DeleteSubscription elimina la subscripción seleccionada correctamente.
Resultados: El test se encuentra en proceso y valida que se ha eliminado la suscripción correctamente .

Tarea 3
Título: Check_GetSubscription
Descripción y objetivos: El objetivo de este test es comprobar que la función getSubscription coge la subscripción que se encuentra disponible en ese momento..
Resultados: El test se encuentra en proceso y valida que se coge la suscripción correctamente .

Pruebas Postman (API)

Tarea 1

- Test 1: Este test se encarga de comprobar que la tabla de géneros se actualiza correctamente en la base de datos después de cada operación.

- Test 2: Este test se encarga de comprobar que la información de la tabla de instrumentos se coge correctamente.

User stories

1.El usuario debe de poder hacer input de su experiencia con instrumentos fácilmente

a) Se cambiarán los dropdowns anteriores por un spinner para cada instrumento, pudiendo seleccionar más fácilmente la experiencia con ese instrumento.

2.El usuario debe ser capaz de recibir feedback detallado sobre el correcto funcionamiento de la app

b) Se añadirán mensajes de error mucho más específicos que detallen exactamente qué ha fallado, ya sea en la API externa o en el sistema Android.

3.El usuario debe poder escoger con qué rol quiere hacer Login

c) Aseguraremos el correcto funcionamiento del radio button group, que anteriormente funcionaba como una lista de checkboxes individuales, a la hora de escoger el rol en la pantalla de login

4.El usuario debe poder escoger los generos de musica favorita

d) Se implementará un RecyclerView en vez de los dropdowns originales para escoger los géneros musicales favoritos del usuario al editar o crear su perfil.

5.El usuario debe de tener un menú de acceso para la gestión de sus perfil y la app

e) Se implementará un “harmburger menu”, deslizante que de acceso a la configuración interna de la app, a la posibilidad de hacer logout y a editar el perfil.

6.El usuario debe poder enviar invitaciones y aceptarlas

f) Se añadirá la posibilidad de ver el perfil de usuarios ajenos como un diálogo

7.El usuario debe de poder diferenciar usuarios amigos de los que no

g) Al clicar en un marker posicionado en nuestro mapa Google Maps, pudiendo invitar a estos usuarios a ser nuestros amigos, cambiando así la información disponible para nosotros sobre ellos.

8.El usuario debe visualizar correctamente y de forma ajustada a su pantalla, las diferentes activities.

h) Se bloquea el giro del dispositivo en horizontal, asegurándose que se respeten las medidas con las que se concibe la app para su correcta presentación en todo tipo de pantallas.

9.El usuario debe de poder visualizar fácilmente qué instrumentos puede escoger

i) Se añadirán iconos identificativos de cada instrumento, para ayudar a la selección que el usuario pretenda llevar a cabo en su perfil y en los filtros.

Aspectos de calidad software

Planned Software Test

El desarrollo de testing se ve impulsado por "cuando" se implementa en esta fase del desarrollo, si en anteriores es sprints se hacía de forma posterior, ahora se sigue una filosofía más cercana al TDD (test Driven Development), lo que quiere decir, que a medida que se va avanzando en el código, inmediatamente se crean los tests unitarios indicados, y se planea la implementación de "Instrumentation Tests" o Tests de integración para el siguiente sprint. Aún se encarga Oriol, pero a medida que se desarrollan las user stories, se comunica más lo que se ha hecho para que pueda revisarse y testearse.

Revisión e inspección

Después de cada commit, el jefe de programación, en este caso, Iván, revisa el código minuciosamente de forma manual, siguiendo las reglas descritas en el primer sprint.

Backlog

Con el sprint Anterior ya evaluado, el propio profesor, Jordi en nuestro caso, señala unos ciertos errores en el código, la primera tarea del jefe de programación, no es continuar con las user stories que pueda tener asignadas, sino producir "fixes" para este backlog de errores hasta que queda vació.

Actas de las reuniones (Scrums)

Data: 25 de abril de 2020 a las 16:00h
Asistentes: Toni Carol, Iván Méndez, Èlia Isart, Oriol Vivó
Orden del dia:

- General: Creación de nuevas User Stories y la distribución de ellas para la realización del Sprint 3 de Android.

Data: 27 de abril de 2020 a las 17:00h
Asistentes: Toni Carol, Iván Méndez, Èlia Isart, Oriol Vivó
Orden del dia:

- General: Cada uno trabaja en su apartado del informe de Especificació y Android.
- Toni / Oriol: Se ha trabajado en la RecyclerView de los géneros.

Principales resultados

Tarea 1 – Arreglos y modificación, inicialización del perfil:

- 2. El usuario necesita un perfil para identificarse -> **hecho**
- 2.1 El usuario tiene que poder instanciar y modificar su perfil en todo momento. -> **hecho**
    - 2.1.1 Acceso al perfil : Si es la primera vez que entras, se abrirá inmediatamente al hacer click en "Sign in".También se accede desde un botón del menú principal. -> **hecho**

Tarea 2 – Implementación géneros musicales del usuario:

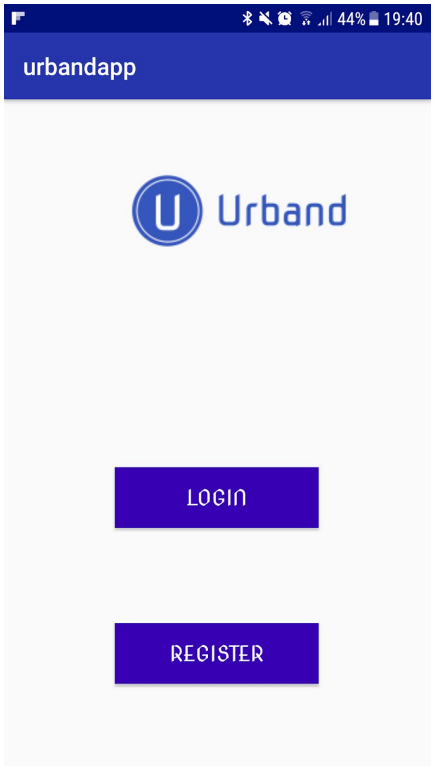
- 2. El usuario necesita un perfil para identificarse -> **hecho**
- 2.1 El usuario tiene que poder instanciar y modificar su perfil en todo momento -> **hecho**
    - 2.1.1 Acceso al perfil -> **hecho**
    - 2.1.2 Modificación del perfil -> **hecho**
        - 2.1.2.1. Diálogo con los datos más personales del usuario -> **hecho**
        - 2.1.2.2. Diálogo con las habilidades musicales, instrumentos, etc -> **hecho**
    - 2.1.3 El usuario debe aceptar unos permisos para poder subir imágenes desde su teléfono -> **no hecho**
        - 2.1.3.1. Aparición "Toast" o "popup" -> **no hecho**

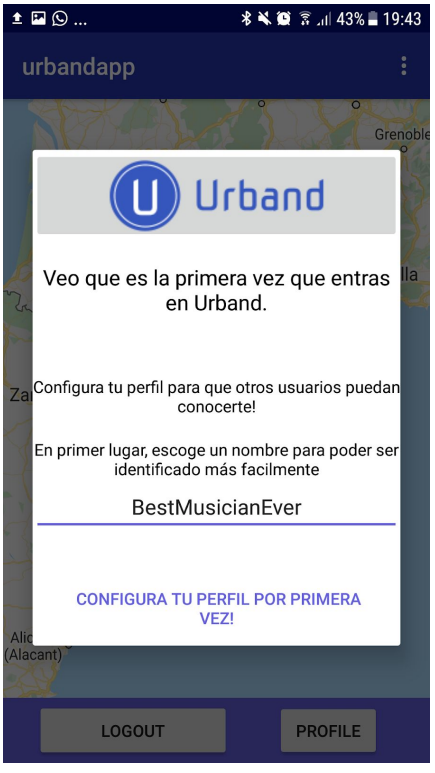
Tarea 3 – Inicio lógica suscripciones entre usuarios:

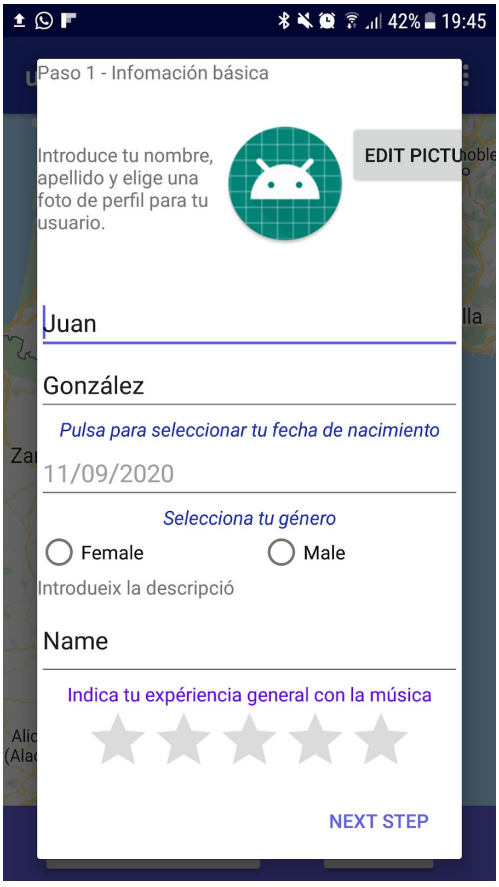
- 7. El usuario debe ser capaz de consultar perfiles de otros usuarios -> **hecho**
- 19. El usuario debe poder suscribirse a otros usuarios para seguir su movimiento en la aplicación -> **hecho**
    - 19.1 Se le ofrecerá una opción de seguir al usuario en la primera opción al aceptar un match -> **hecho**
    - 19.2 Si consulta un perfil ajeno, no match, también se le ofrecerá la opción -> **hecho**

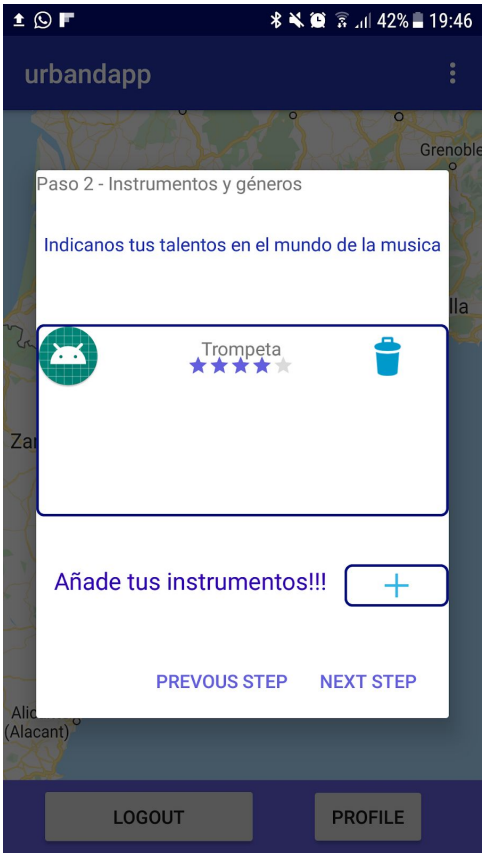
Conclusiones – Ciclo de vida

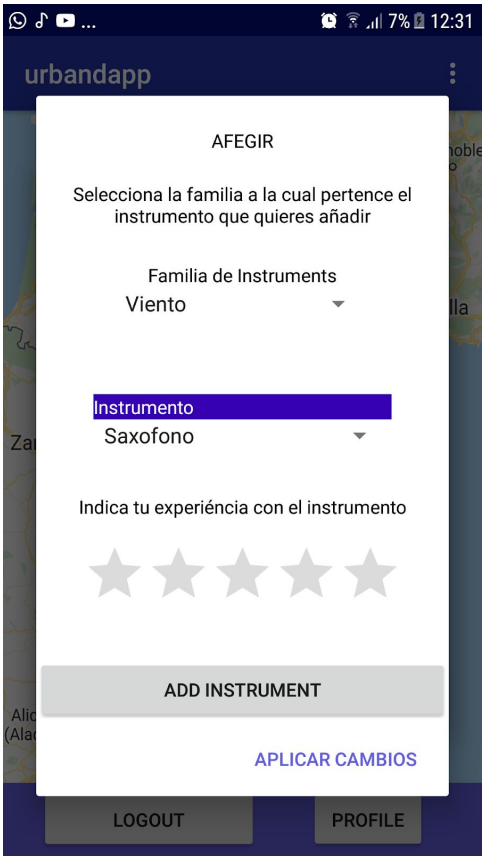
En este último apartado vamos a mostrar el ciclo de vida que sigue nuestra aplicación desde su primer uso. El camino que sigue el usuario al entrar en la aplicación es:
El usuario se registra en la aplicación y seguidamente se loguea. El usuario se encuentra con un diálogo que se muestra al ser la primera vez que entra. A medida que va avanzando los pasos del diálogo, el usuario va completando los datos de su perfil. Una vez hecho esto el usuario ya puede utilizar la aplicación.

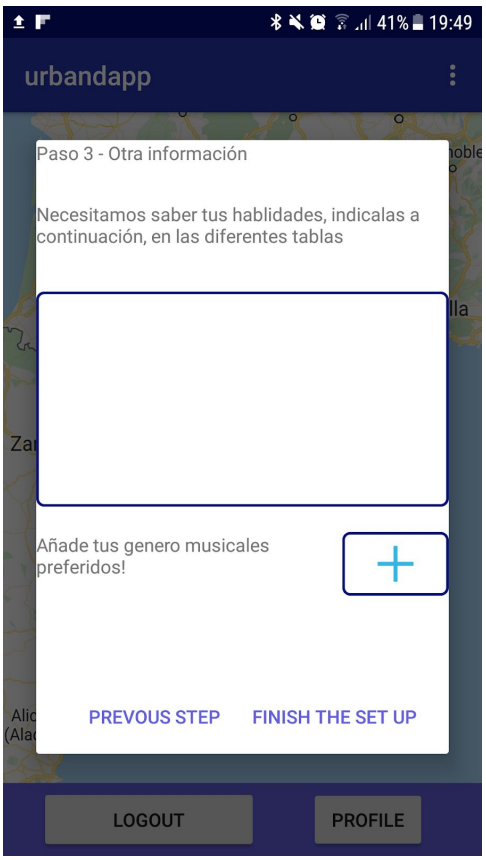


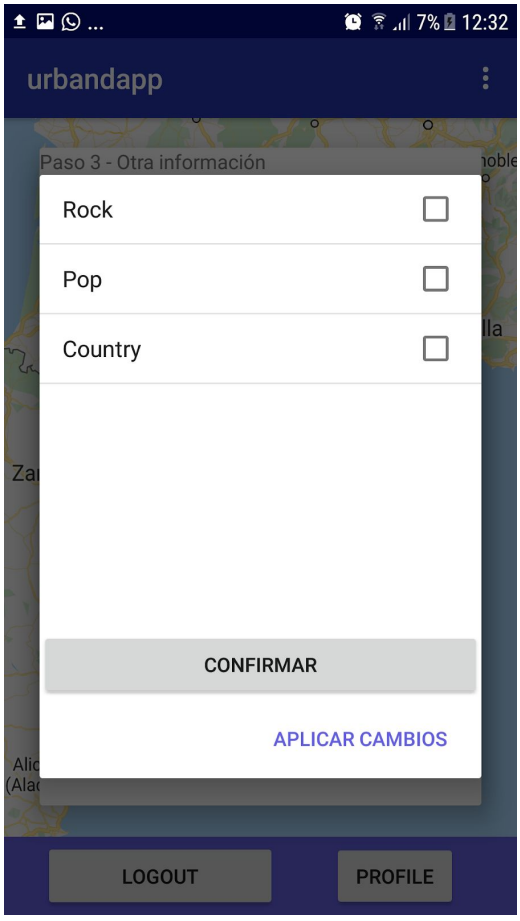



42% 19:45
Paso 1 - Infomación básica
Introduce tu nombre, apellido y elige una foto de perfil para tu usuario.
EDIT PICTU
Juan
González
Pulsa para seleccionar tu fecha de nacimiento
11/09/2020
Selecciona tu género
Female
Male
Introdueix la descripció
Name
Indica tu expériencia general con la música
NEXT STEP


42% 19:46
urbandapp
Paso 2 - Instrumentos y géneros
Indicanos tus talentos en el mundo de la musica
Trompeta
Añade tus instrumentos!!!
PREVOUS STEP
NEXT STEP
LOGOUT
PROFILE


7% 12:31
urbandapp
AFEGIR
Selecciona la familia a la cual pertence el instrumento que quieres añadir
Familia de Instruments
Viento
Instrumento
Saxofono
Indica tu experiéncia con el instrumento
ADD INSTRUMENT
APLICAR CAMBIOS
LOGOUT
PROFILE


41% 19:49
urbandapp
Paso 3 - Otra información
Necesitamos saber tus hablidades, indicalas a continuación, en las diferentes tablas
Añade tus genero musicales preferidos!
PREVOUS STEP
FINISH THE SET UP
LOGOUT
PROFILE


7% 12:32
urbandapp
Paso 3 - Otra información
Rock
Pop
Country
CONFIRMAR
APLICAR CAMBIOS
LOGOUT
PROFILE


43% 19:42
urbandapp
Francia
Lista de usuarios
Barcelona
Madrid
Valencia
España
Granada
Málaga
Argel
LOGOUT
PROFILE