

Urband

Sprint 4- Consolidación

20/05/2020

Oriol Vivó, Iván Méndez, Èlia Isart, Antoni Carol

Índice

Revisión del trabajo a realizar y elementos a implementar en el Sprint	4
TAREA 1 - Perfil de Usuario y sistema de roles	5
TAREA 2 - Arreglos visuales generales interfaces de usuario	8
TAREA 3 - Filtros y Matching entre usuarios	9
Definición y realización de los tests	11
User Stories	14
Aspectos de calidad del software	15
Actas de las reuniones (Scrums)	16
Principales resultados	17
Evolución de los requisitos	18
Revisión de todos los requisitos	21
Proceso seguido	22
Herramientas	22

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

En este sprint, hemos establecido diferentes tareas a realizar durante estas semanas en la que se trabajará. Nos encontramos en el Sprint 4, el sprint de consolidación, por eso vamos a dividir nuestras tareas en 2 grupos:

Tareas Consolidación - Nos centraremos en las decisiones que tomamos.

Tarea 1 - Perfil De Usuario y roles de usuario

- Esta tarea se concretó en el Sprint número 1, pero a causa de los resultados al final del sprint, decidimos trabajar en un sprint posterior de consolidación, debido a su baja complejidad
- **Roles de usuario:** esta tarea se definió anteriormente pero por su poca prioridad, ya que de momento en nuestro MVP no queremos implementar el sistema de roles completo, pero sí que nos gustaría que se pudiera escoger con qué rol de usuario entrar.

Tarea 2- Arreglos visuales generales interfaces de usuario

- Como en nuestro calendario ya definimos, nos presentaron, este sprint de consolidación, como grupo, pensamos en dejar la estética de la aplicación para este momento del desarrollo.

FORMATO: Por cada tarea, vamos a definir los requisitos afectados, un poco de su evolución, mostrando los cambios más significativos entre este sprint y anteriores, porque hacemos estos cambios y que aportan al sistema.

Tareas Desarrollo - Nos Centraremos en la validación de los requisitos

Tarea 4 - Filtros y match entre usuarios

- Necesitamos implementar todas las funcionalidades más básicas, para poder presentar nuestro MVP a los usuarios e inversores, por eso no podemos saltarnos esta fase del desarrollo, muy importante y bastante compleja.

FORMATO: Vamos a definir los requisitos afectados, su evolución y un poco de su trazabilidad, no nos centraremos tanto en dar nuestro punto de vista y razonamiento, sino en su tecnicidad.

TAREA 1 – Perfil de Usuario y sistema de roles

Como comentamos en la introducción, esta tarea estaba planificada para ser desarrollada de lo primero, pero se aplazó su implementación.

Se divide en 2 partes, referentes a los diferentes tipos de perfil, su información:

- Privado : Perfil que el usuario logueado verá toda su información añadida
- Público : Perfil de otro usuario, dónde se reducirán los campos, para solo mostrar lo más importante en una búsqueda
 - ❖ *Dividimos la tarea para dar importancia a estos 2 tipos de perfil, con el privado queremos que nuestro usuario pueda ver y editar su perfil mediante un acceso rápido. En el caso del público, queremos que solo sea visible y que no cargue al usuario con mucha información.*

Por parte de los roles de usuarios, en esta tarea sólo se da la opción de elegir el rol, sin funcionalidades añadidas específicamente para los roles.

Requisitos

En esta tarea, por la parte del perfil de usuario, nos centraremos en el requisito 2, y el requisito número 7 definidos en el documento de Ingeniería de Requisitos.

2. El usuario necesita un perfil para identificarse



Proporcionaremos al usuario una pantalla dónde inicialmente, la primera vez que se inicia sesión se rellenan, y en posteriores logins se editan

Por parte del sistema de roles atacaremos al requisito 12.

12. Nuestro sistema debe contener un sistema de roles para diferenciar los tipos de usuarios que utilizarán la aplicación



En la aplicación tendremos cuatro tipos de roles diferentes, cada uno de ellos aportará características diferentes a nuestra app y esto hará que pueda llegar a un público más general, estos roles son los siguientes:


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



Nuestra aplicación ofrecerá la opción de ver los perfiles de los músicos con los que te ha emparejado en forma de lista con un scroll, donde los usuarios están ordenados según hayan mayores coincidencias en lo que están buscando.

Gestión y Evolución de los Requisitos


Empezaremos con el requisito 2, más primordial que el 7 ya que necesitamos añadir la información al perfil para ver nuestro perfil privado.

2. El usuario necesita un perfil para identificarse 
<p>2.1 Acceso al perfil ...</p> <p>2.2 Instanciación del perfil: <i>dividimos la configuración del perfil en 3 etapas</i></p> <p>- 2.2.1.</p> <p>*****Requisito nuevo</p> <p>2.3 Modificación del Perfil : el usuario debe poder modificar su perfil.</p> <p>- 2.3.1 El sistema ofrecerá en la interfaz del perfil de Usuario privado, unas opciones que permitirán al usuario modificar sus datos</p>

Para la parte del **perfil privado**, para poder analizar y evaluar, el requisito nuevo, 2.3 hijo del requisito 2, usaremos las siguientes user-stories.

USER-STORIES
3.El usuario debe poder ver su perfil siempre que quiera, una vez creado

Por parte del requisito 7, referente al perfil público de un usuario que no es el logeado, también ha sufrido unos cambios para su mejor validación.

7. El usuario debe ser capaz de consultar perfiles de otros usuarios 
<p>*****Editado</p> <p>7.1 El usuario verá el perfil completo si se trata de un Usuario al cual esté suscrito.</p> <ul style="list-style-type: none"> ➤ 7.1.1 Si el usuario no está suscrito, sólo le mostrará la imagen de perfil, el nombre de usuario y su experiencia general como músico ➤ 7.1.2 Si está suscrito, se le presentará la información restante; habilidades(Instrumentos y géneros musicales), y su información más personal y de contacto.

Cómo comentamos en la introducción de esta tarea, por parte de los Roles De Usuario, no trataremos los requisitos, ya que solo se implementa una instanciación para futuras versiones.

★ Atacamos estos requisitos a raíz de las tareas definidas por eso queremos analizar y gestionar estos requisitos.

Trazabilidad

Una vez definidos los requisitos y su evolución vamos a comprobar mediante diferentes tipos de pruebas los resultados conseguidos en esta fase de diseño.

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
2.1	2	6	Check_profile, Check_parameters, Check_profile_parameters
2.3	2	6	Check_Modify_Profile
7.1.1	3	7	Check_profile, Check_parameters, POSTMAN : Suscribe_tests
7.1.2	3	7	Check_profile, Check_parameters, POSTMAN : Suscribe_tests

- ★ Vamos a comprobar sólo los requisitos hijo, ya que son los que forman al padre, una vez los hijos estén validados, el padre será automáticamente comprobado.
- ★ No hacemos trazabilidad del sistema de roles, todo que existan tests para este requisito ya que solo se crea una instanciación, será comprobado en futuros sprints

TAREA 2 - Arreglos visuales generales interfaces de usuario

Esta tarea de consolidación se basa en dejar la vista cómo queremos que el usuario la vea, esta tarea se ha decidido hacer en esta fase debido a querer enfocarnos en tener las funcionalidades logradas y con nitidez, y a que su complejidad de diseño visual en android no es la tarea más complicada que tenemos adelante

Esta tarea no tiene ningún requisito asociado, ya que se trata de una tarea de consolidación, comprobar un requisito que no hace falta que sea definido; las vistas deben tener en cuenta aspectos de UX (User-Experience), y se deben ver y adaptar a todo tipo de dispositivo android

Frente el diseño visual, comentar que no será un diseño muy complejo, a continuación vamos a ver los cambios más significativos en el diseño visual actividad por actividad:

Actividad/Vista	Modificación
ChoserActivity	<ul style="list-style-type: none"> ➤ Se añadirán elementos para dar más imagen de nuestra marca.
RegisterActivity	<ul style="list-style-type: none"> ➤ Se añadirá una interacción de “¿Ya tienes una cuenta, inicia sesión” ➤ e añadirá otro campo que será la repetición de la contraseña para verificar su registro de usuario
LoginActivity	<ul style="list-style-type: none"> ➤ Se añadirá interacción de “No tienes cuenta?, regístrate!” ➤ Se corrigen errores en los campos de texto.
DefaultActivity (MAP)	<ul style="list-style-type: none"> ➤ Se añadirán diferentes iconos para los diferentes roles que existen. ➤ Se remodelará el Menú Inferior para que esté presente en todas las otras actividades; Filters, Match, Profile. ➤ Se corregirán errores y se añadirán más módulos al menú desplegable situado en la parte superior derecha.
Profile Set Up Dialog Step 1-2-3	<p>Step 1 :</p> <ul style="list-style-type: none"> ➤ Se corregirá el formato de la fecha seleccionada. ➤ Se añadirán elementos para dar más imagen de nuestra marca. <p>Step 2 - Step 3 :</p> <ul style="list-style-type: none"> ➤ Se añadirán elementos para dar más imagen de nuestra marca. ➤ Se editará el ítem de la RecyclerView
Private Profile	<ul style="list-style-type: none"> ➤ Se añadirán elementos para dar más imagen de nuestra marca.
Public Profile Dialog	<ul style="list-style-type: none"> ➤ Se añadirán elementos para dar más imagen de nuestra marca.
FiltersActivity	<ul style="list-style-type: none"> ➤ Se añadirán elementos para dar más imagen de nuestra marca.
MatchDialog	<ul style="list-style-type: none"> ➤ Se añadirán elementos para dar más imagen de nuestra marca.
FirstTimeDialog	<ul style="list-style-type: none"> ➤ Reformato del estilo del diálogo completo

TAREA 3 – Filtros y Matching entre usuarios

Esta será la última tarea de desarrollo implementada en la aplicación para completar las funcionalidades mínimas para nuestro MVP (Minimum Value Product). Es la funcionalidad más compleja de sistema y requiere una implementación costosa y profunda. Por eso, implementamos un sistema de match bastante más básico de lo que teníamos pensado.

Estas dos tareas, filtros y matching entre usuarios, están relacionadas, su diferencia es que el match le ofrecerá directamente el usuario más compatible a él, y los filtros le darán la opción de buscar entre la lista de usuarios.

8. El usuario debe poder contactar con resultantes de un match



6. El sistema debe contener un sistema de matching efectivo.



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



USER-STORIES

1.El usuario debe tener un mecanismo para encontrar usuarios que coincidan con los gustos y experiencias que se han indicado en el perfil de usuario

2.El usuario debe poder buscar otros usuarios con características concretas.

Gestión y Evolución de los Requisitos

Cómo comentamos anteriormente, esta tarea no quedará finalizada en esta parte del diseño, pero si será funcional para ver su interacción. De este modo vamos a ver cómo adaptamos los requisitos para su correcta validación.

6. El sistema debe contener un sistema de matching efectivo.




6.1 El sistema va a buscar un usuario por los parámetros más importantes:

- **6.1.1 Distancia** : según la distancia entre usuarios
- **6.1.2 Géneros** : comprobará si tiene algún género en común
- **6.1.3 Instrumentos** : si no encuentra ningún match, buscará por instrumentos

6.2 El sistema notificará al usuario cuando haya surgido un match

- **6.2.1 Aparecerá un diálogo** donde se presentará el match, conteniendo el nombre de usuario, su foto de perfil y su experiencia general
- **6.2.2 El usuario tendrá 2 opciones** al visualizar un match: ver su perfil y ponerse en contacto con el, o guardar el match e inspeccionar más adelante

A raíz de la tarea de filtrar usuarios, nace un requisito nuevo, ya que la acción se realiza en una actividad diferente y necesitamos comprobar a parte sus resultados

20. El usuario debe poder filtrar entre todos los usuarios de la aplicación  Urband
<p>20.1 El sistema presentará un módulo que abrirá una actividad dónde el usuario podrá filtrar según sus intereses</p> <ul style="list-style-type: none"> - 20.1.2 De cada usuario en la lista de resultados, el usuario presentará su foto de perfil, nombre de usuario y experiencia general como músico. <p>20.2 El sistema ofrecerá la posibilidad de filtrar por Instrumentos y Géneros musicales que los usuarios hayan escogido para presentar en su perfil.</p>

Trazabilidad

Una vez definidos los requisitos y su evolución vamos a comprobar mediante diferentes tipos de pruebas los resultados conseguidos en esta fase de diseño.

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
6.1	1	10	Check_Match
6.2	1	8	Show_Match_User
20.1.2	3	7	Check_Users_Filtered
20.2	3	7	Check_Filters

- ★ No hacemos trazabilidad del requisito 8, ya que aún no se implementa esta información, este requisito se nombra por el contexto en el que nos encontramos, y lo mostramos para tener en cuenta todos los requisitos a los que podemos atacar usando una tarea.

Definición y realización de los tests

Pruebas unitarias

En relación con el requisito 2 (el usuario necesita un perfil para identificarse) surgen las siguientes pruebas unitarias con el fin de validar todos los aspectos relacionados con el perfil:

Tarea 1
Título: Check_profile
Descripción y objetivos: El objetivo de este test es comprobar que al pulsar sobre el botón de perfil, se muestra correctamente el perfil del usuario.
Resultados: El test valida que al pulsar el botón de perfil se muestra el layout correspondiente.

Tarea 1
Título: Check_profile_parameters
Descripción y objetivos: el objetivo de este test es comprobar que cada parámetro del perfil de usuario es correcto y se muestra por pantalla: -Foto de perfil. -Nombre de usuario
Resultados: El test valida que los parámetros del perfil son correctos.

Tarea 1
Título: Check_profile_database
Descripción y objetivos: el objetivo de este test es comprobar que toda la información del perfil se guarda correctamente en la base de datos si se modifica.
Resultados: El test valida que los cambios realizados en el perfil se guardan correctamente en la base de datos.

En relación con el requisito 12 (Nuestro sistema debe contener un sistema de roles para diferenciar los tipos de usuarios que utilizarán la aplicación) realizamos esta prueba para validar el requisito:

Tarea 1
Título: Check_Rol
Descripción y objetivos: el objetivo de este test es comprobar que se guarda correctamente el rol escogido por el usuario.
Resultados: El test valida que el usuario está logueado con un rol.

Tarea 2: En esta tarea no se realizan pruebas unitarias. Al ser una tarea dedicada exclusivamente a arreglos visuales consideramos que no es necesario realizar ningún test.

En relación con el requisito 6 (El sistema debe contener un sistema de matching efectivo) realizamos las siguientes pruebas unitarias para validar el sistema de matching.

Tarea 3
Título: Show_Match_User
Descripción y objetivos: el objetivo de este test es comprobar que se muestran los usuarios que se deben mostrar una vez realizado el matching.
Resultados: El test valida que se muestran los usuarios que han hecho match.

Tarea 3
Título: Check_Match
Descripción y objetivos: el objetivo de este test es comprobar que el matching se realiza correctamente y muestra los usuarios que coinciden.
Resultados: El test valida que el matching es correcto.

En relación con el requisito 19 (el usuario debe poder filtrar entre todos los usuarios de la aplicación) realizamos la siguiente prueba unitaria para validar el proceso de filtraje.

Tarea 3
Título: Check_Filters.
Descripción y objetivos: el objetivo de este test es comprobar que el usuario puede filtrar correctamente los usuarios que busca. Para ello realizamos diferentes pruebas de filtraje con diferentes valores.
Resultados: El test valida que se filtra de forma correcta.

Tarea 3
Título: Check_Modify_Profile
Descripción y objetivos: el objetivo de este test es comprobar que cuando el usuario cambia algún valor de su perfil, este se guarda y se muestra correctamente.
Resultados: El test valida que los cambios en el perfil se realizan correctamente.

Tarea 3
Título: Check_Users_Filtered
Descripción y objetivos: el objetivo de este test es comprobar que el usuario obtiene el nombre de otro usuario cuando introduce filtros que coinciden.
Resultados: El test valida que el filtrado nos muestra al usuario correcto.

Pruebas POSTMAN (API)

Tarea 1

- Test 1: Realizamos una prueba para comprobar que se guardan correctamente los roles.
- Test 2: Realizamos una prueba para comprobar que se guardan correctamente los datos del perfil.

Tarea 2 :

- Esta no dispone de pruebas ya que no hay ninguna interacción con la API en esta tarea.

Tarea 3

- Test 1: Realizamos una prueba para comprobar que se obtiene el match correcto.
- Test 2: Realizamos una prueba para comprobar que se filtra correctamente.

User Stories

1. El usuario debe tener un mecanismo para encontrar usuarios que coincidan con los gustos y experiencias que se han indicado en el perfil de usuario

- Se presenta, inicialmente, un botón/icono en el que el usuario pueda clicar desde la actividad inicial , para obtener un “match” de un usuario similar a el.

2.El usuario debe poder buscar otros usuarios con características concretas

- Se presentará al usuario una activity, a la que se accede desde el menu hamburger, que ofrezca un dropdown para género, y otro para instrumento, Con una lista “RecyclerView” inmediatamente debajo de los mismos, que presente en tiempo real los usuarios encontrados según los filtros.

3.El usuario debe poder ver su perfil siempre que quiera, una vez creado

- Se creará una pantalla, donde se mostrará toda la información del usuario que esté usando la app, y se dejará modificar desde ella, con los inputs necesarios , todas las características del perfil.

4.El usuario ha de ver una interfaz pulida y uniforme

- Se cambiarán y pulirá todo texto, imagen marco, que no siga los colores, fuentes, tamaño de fuentes, por los oficiales ya definidos en un documento anterior. Se bloqueara la opción de ver pantalla en horizontal, para evitar bugs visuales, y se pasarán todas las medidas de los elementos a porcentajes del viewport, en vez de pixeles, para que la experiencia sea la misma en diferentes dispositivos.

Como se puede apreciar, en este sprint se atacan user stories de desarrollo por una parte (1,2) y de “consolidación” (3,4),

No queríamos acabar la fase de desarrollo sin las “features” más básicas que se comentaban desde el principio, pero hay una falta de pulidez en la app que nos preocupaba, por eso mismo, no nos hemos dirigido a ninguna delas dos direcciones en extremo, si no que hemos intentado llegar a un compromiso. Aun así, faltan otras “features” menos importantes, y siguen habiendo aspectos de programación aún mejorables, pero nos sentimos satisfechos con lo obtenido.

Aspectos de calidad del software

Planned Software Test

Más que nunca, es necesaria la filosofía TDD (Test Driven Development) , lo que quiere decir, que a medida que se desarrollan las últimas piezas de software, se intenta crear el test, o como mínimo testear manualmente, cada función o evento. Esto lo conseguimos con tests unitarios, y aunque se ha planteado la inclusión de tests de integración , finalmente no se han añadido por escasez de tiempo y planificación inicial del sprint.

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, da feedback de ciertos errores y mejoras, tanto en la API, como en Android., 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 vacío.

Actas de las reuniones (Scrums)

Data: 6 de mayo de 2020 a las 20:45h

Asistentes: Toni Carol, Iván Méndez, Èlia Isart, Oriol Vivó

Orden del día:

- General: Acuerdo en la creación de un Sprint que refleje las tareas que se han creado pero razonando el motivo por el que las hacemos y también exponer qué no hacemos y porqué.

Data: 13 de mayo de 2020 a las 15:30h

Asistentes: Toni Carol, Iván Méndez, Èlia Isart, Oriol Vivó

Orden del día:

- General: Creación y distribución de las tareas a hacer para la presentación de Especificació.

Principales resultados

Tarea 1 – Perfil de usuario

- 2. El usuario necesita un perfil para identificarse → **hecho**
 - 2.1 Acceso al perfil... → **hecho**
 - 2.2 Instanciación del perfil → **hecho**
 - 2.3. Modificación del perfil → **hecho**
- 3. El usuario debe poder ver su perfil siempre que quiera, una vez creado → **hecho**

Tarea 2 – Arreglos visuales generales interfaces de usuario

- Choose Option → **en proceso**
- Registro → **en proceso**
- Login → **en proceso**
- Default → **en proceso**
- Profile Set up Dialogs → **en proceso**
- Formato fecha seleccionada → **en proceso**
- Filters → **en proceso**
- Private Profile → **en proceso**
- Public Profile Dialog → **en proceso**
- Match Dialog → **en proceso**

Tarea 3 – Filtros y matching entre usuarios

- 6. El sistema debe contener un sistema de matching efectivo → **hecho**
 - 6.1 El sistema va a buscar un usuario por los parámetros más importantes → **hecho**
 - 6.1.1 Distancia → **hecho**
 - 6.1.2 Géneros → **hecho**
 - 6.1.3 Instrumentos → **hecho**
 - 6.2 El sistema notificará al usuario cuando haya surgido un match → **hecho**
 - 6.2.1 Aparecerá un diálogo donde se presentará el match → **hecho**
 - 6.2.2 El usuario tendrá 2 opciones al visualizar un match → **hecho**
- 8. El usuario debe poder contactar con resultantes de un match → **hecho**
- 20. El usuario debe poder filtrar entre todos los usuarios de la aplicación → **hecho**
 - 20.1 El sistema presentará un módulo que abrirá una actividad dónde el usuario podrá filtrar según sus intereses → **hecho**
 - 20.1.2. El usuario presentará su foto de perfil, nombre de usuario y experiencia general como músico → **hecho**

- 20.2 El sistema ofrecerá la posibilidad de filtrar por Instrumentos y Géneros musicales que los usuarios hayan escogido para presentar en su perfil → **hecho**

Evolución de los requisitos

Al inicio del proyecto se especificaron 17 requisitos diferentes con sus correspondientes sub requisitos para que la aplicación tuviese la funcionalidad básica para su uso. Durante este proceso se han ido realizando en función de las necesidades que se nos haya requerido y para tener la operatividad solicitada para la entrega final.

Gestión de los requisitos

A continuación podemos ver los requisitos finales que se han implementado en la app. En ellos ha habido modificaciones durante el proceso, es por ello que algunos requisitos han estado modificados o bien han estado creados.

Requisitos creados → Gris

Requisitos modificados → Rojo

- 1.El sistema debe tener un método para que los usuarios se registren
 - 1.1 El usuario necesita una pantalla dónde registrarse con sus datos
 - 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.
 - 1.1.1.4 El sistema debe guardar los datos de registro del usuario, para poder redirigirlo al logeo de usuario
 - 1.2 El usuario necesita una pantalla donde poder iniciar sesión con sus datos
 - 1.2.1 Pantalla de Logeo: Un formulario con dos campos, el identificador, teléfono 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
 - 1.2.1.3 El sistema debe autenticar 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 token de autenticación cuando el usuario salga de su cuenta mediante el menú
- 2. El usuario necesita un perfil para identificarse
 - 2.1 El usuario tiene que poder instanciar y modificar su perfil en todo momento
 - 2.1.1 Acceso al perfil
 - 2.1.2 Instanciación y Modificación del perfil
 - 2.1.2.1 Dialogo con los datos más personales del usuario
 - Nombre completo, fecha de nacimiento, género, experiencia general en la música, foto de perfil y una pequeña descripción.
 - 2.1.2.2 Dialogo con las habilidades musicales, 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 elementos
 - 2.1.3 El usuario debe aceptar unos permisos para poder subir imágenes desde su teléfono
 - 2.1.3.1. Aparición "Toast" o "popup"
 - 2.1.4 El sistema ofrecerá en la interfaz del perfil de Usuario privado, unas opciones que permitirán al usuario modificar sus datos
- 3. El usuario en todo momento debe disponer de conexión a Internet
- 4. Al usuario se le pedirán permisos para acceder a su ubicación, Gps
 - 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
- 6. El sistema debe contener un sistema de matching efectivo
 - 6.1 El sistema va a buscar un usuario por los parámetros más importantes
 - 6.1.1 Distancia
 - 6.1.2 Géneros
 - 6.1.3 Instrumentos
 - 6.2 El sistema notificará al usuario cuando haya surgido un match

- 6.2.1 Aparecerá un diálogo donde se presentará el match
- 6.2.2 El usuario tendrá 2 opciones al visualizar un match
- 7. El usuario debe ser capaz de consultar perfiles de otros usuarios
 - 7.1 El usuario verá el perfil completo si se trata de un Usuario al cual esté suscrito
 - 7.1.1 Si el usuario no está suscrito, sólo le mostrará la imagen de perfil, el nombre de usuario y su experiencia general como músico
 - 7.1.2 Si está suscrito, se le presentará la información restante; habilidades (Instrumentos y géneros musicales), y su información más personal y de contacto
- 8. El usuario debe poder contactar con resultantes de un match
- 10. El usuario debe ser capaz de inspeccionar y encontrar eventos cercanos 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
 - 10.1.2 Marker
 - 10.1.3 Filtros
 - 10.1.4 Músicos, grupos o conciertos
- 12. Nuestro sistema debe contener un sistema de roles para diferenciar los tipos de usuarios que utilizarán la aplicación
- 18. Nuestro sistema debe integrar una base de datos donde consultar la información sobre usuarios u otro tipo de información dinámica.
 - 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
- 19. El usuario debe poder suscribirse a otros usuarios para seguir su movimiento en la aplicación
 - 19.1 Se le ofrecerá una opción de seguir al usuario en la primera opción al aceptar un match
 - 19.2 Si consulta un perfil ajeno, no match, también se le ofrecerá la opción
- 20. El usuario debe poder filtrar entre todos los usuarios de la aplicación
 - 20.1 El sistema presentará un módulo que abrirá una actividad donde el usuario podrá filtrar según sus intereses
 - 20.1.2. El usuario presentará su foto de perfil, nombre de usuario y experiencia general como músico

- 20.2 El sistema ofrecerá la posibilidad de filtrar por Instrumentos y Géneros musicales que los usuarios hayan escogido para presentar en su perfil

Revisión de todos los requisitos

La siguiente tabla representa esta gestión y trazabilidad de los requisitos, se va a estructurar de la siguiente forma: en color **verde** los requisitos completados y probados, en color **naranja** los que están en proceso, u en **rojo** los pendiente.

Requisito	Tipo	Prioridad	Complejidad	Estado
1	Usuario	1	8	DONE
3	Usuario	1	4	DONE
5	Sistema	1	1	DONE
6	Sistema	1	10	DONE
7	Usuario	2	9	DONE
8	Usuario	2	7	DONE
10	Usuario	3	7	DONE
18	Sistema	2	5	DONE
19	Usuario	2	8	DONE
20	Usuario	2	9	DONE
2	Usuario	2	8	PENDING 2.1.3
9	Usuario	2	3	IN PROGRESS
12	Sistema	2	7	Just Initialized
4	Usuario	3	6	TO DO
11	Usuario	4	8	TODO
13	Sistema	4	8	TODO
14	Sistema	5	9	TODO
15	Usuario	5	7	TODO
16	Sistema	5	4	TODO
17	Usuario	5	5	TODO

Hasta este momento se han realizado un 50% de los requisitos programados, y un 15% están en desarrollo actualmente. Estos datos quieren decir que solamente falta un 35% de desarrollo de la aplicación para que sea totalmente funcional.

Cómo podemos ver, la estructura general de los requisitos iniciales se ha mantenido durante el proceso, habiendo pequeños cambios en ellos, los cambios más considerables han sido los casos que se han añadido nuevos requisitos. Aunque se han añadido nuevos requisitos ya sabíamos que seguramente durante el transcurso de la implementación tendríamos que necesitar otros requisitos que en el momento de la planificación no nombramos, esto es a causa de que a medida del proceso hemos ido viendo que nos surgían nuevas necesidades que tendrían que ser cumplidas para poder hacer otros requisitos previamente descritos.

Proceso seguido

Durante el transcurso de la asignatura el grupo ha tenido buena relación, es por ello no ha habido problemas entre integrantes del mismo y por lo general las tareas requeridas durante este proceso se han cumplido en las fechas programadas. Para poder llegar a cumplir el objetivo de tener una aplicación con las funcionalidades más básicas para la entrega final hemos tenido en cuenta una serie de medidas para la mejor organización y optimización del tiempo del grupo.

Acto seguido a la presentación de cada sprint, el grupo se ha reunido para hacer la consiguiente división de las tareas para el sprint organizado. En estas reuniones se ha marcado una fecha límite en el grupo para tener la tarea realizada y así poder tener un margen de unos días para hacer las correspondientes mejoras o cambios necesarios para tener unas entregas bien realizadas. Durante cada semana de sprints se han hecho reuniones entre el grupo entero o integrantes del grupo para ver cómo está yendo el desarrollo de la práctica. Además de estas reuniones también se ha trabajado conjuntamente, es decir, algunos integrantes del grupo han hecho las tareas juntos para así poder ayudarse.

Herramientas

Seguidamente nombraremos las diferentes herramientas que hemos usado para completar este proceso.

Desarrollo de la aplicación:

- PyCharm: para el desarrollo de la aplicación
- Android Studio: para el desarrollo de la aplicación
- Postman: testeo de la aplicación
- Docker: automatizar despliegue aplicaciones

Desarrollo documentación:

- Drive: escritura de la documentación
- LucidChart: creación de los diagramas de clases

Reuniones de grupo:

- Skype: reuniones de grupo o con profesores
- Discord: reuniones de grupo o llamadas individuales

Organización del grupo:

- Trello: ver el desarrollo de las tareas programadas