

Urband

Pràctica 2 – Anàlisis de Requisitos

3/03/2020

Oriol Vivó, Iván Méndez, Èlia Isart, Antoni Carol

Índice

Introducción	3
Recogida y análisis de los requisitos del sistema software	4
Escenarios	4
Encuesta	7
Especificación formal de los requisitos	10
Diagrama de casos de uso	16
Plan de validación y trazabilidad	17
Plan de gestión de la evolución de los requisitos	20

Introducción

En el desarrollo de este documento, se decidirá y justificará las técnicas o [métodos de recogida de requisitos](#), ya que existen muchos y de diferente tipo. Como los requisitos se documentan, también se escribirá el [documento de especificación de requisitos](#), adaptando éste a cada proyecto, utilizando diagramas / plantilla. Para garantizar que el desarrollo cumpla con los requisitos, en esta práctica también se definirá el [plan de trazabilidad y validación de los requisitos](#) que se realizará durante el desarrollo del proyecto. Y como los requisitos pueden variar, se especificará la [gestión de la evolución de los requisitos](#).

Recogida y análisis de los requisitos del sistema software

En esta etapa, entendemos y interactuamos con los *stakeholders*, todo usuario involucrado de alguna manera o otra en nuestro sistema, mediante diferentes técnicas como entrevistas, la observación, etc.

En nuestro caso para conseguir feedback del usuario y transformarlo en requisitos para nuestro sistema, hemos decidido usar dos técnicas diferentes; entrevistas y escenarios. Recopilamos esta información a partir de usuarios potenciales de la aplicación, en forma de requisitos más específicos, pero también con las encuestas abiertas a todo tipo de público, valoraremos si nuestra idea podría acabar funcionando o no.

Escenarios

Descripción de los escenarios

Usaremos la técnica de los escenarios para la recogida de requisitos, describiendo diferentes casos de uso.

Estos escenarios seguirán todos la misma plantilla donde se definirá; quien participa en el caso de uso, el **actor**, los **precondición**, pre-requisitos que necesita el usuario para empezar la tarea, el **escenario principal**, los pasos detallados que el usuario debe hacer para cumplir el objetivo propuesto y, si existe una manera secundaria de cumplirlo, definiremos un **escenario secundario**, **WCGW** (*what can go wrong*), las cosas que podrían salir mal en el proceso, para acabar y el resultado esperado, la **postcondición**.

Usando esta plantilla definida anteriormente, hemos definido 3 escenarios para profundizar en los recursos más significativos del sistema.

1. Buscar un músico en la zona

Actor:	Músico interesado en encontrar otro músico para colaborar, aprender, etc.
Precondición:	<ul style="list-style-type: none"> • Tiene que estar registrado en la aplicación • Perfil y su descripción completados • Disponer de conexión en Internet y gps
Escenario Principal:	<p>El usuario debe introducir los datos que quiere encontrar, aplicarlos en los filtros para que el sistema de match encuentre usuarios compatibles.</p> <p>Al realizarse la búsqueda, debe inspeccionar y encontrar a su usuario ideal de la lista presentada ordenada por mas compatible a menos.</p> <p>Al encontrarlo, debe entrar en su perfil y si es lo que está buscando, contactar con el.</p>
WCGW	<ul style="list-style-type: none"> • Error en los filtros; no se filtre correctamente o el match no es adecuado • Error humano en seleccionar mal el match, o los filtros • Error en los datos del perfil • La lista de matches no contiene ningún usuario, debido al rango en KM
Postcondición	El usuario puede contactar con otro músico compatible.

Conclusiones (Requisitos adquiridos):

- 1.El sistema debe tener un método para que los usuarios se registre.
- 2.El usuario necesita un perfil para identificarse
- 3.Sistema wifi y gps
- 4.Sistema de matching con filtraje
- 5.Consulta perfiles ajenos
- 6.Contacto entre usuarios al hacer match
- 7.Se tiene que poder “volver atrás” en todo el proceso

2. Buscar conciertos en la zona

Actor:	Persona interessada en música local
Precondición:	<ul style="list-style-type: none"> • El usuario tiene que estar registrado • Disponer de conexión a Internet y gps
Escenario principal	El usuario iniciará la aplicación y iniciará sesión, si aún no se ha <i>logeado</i> . Tendrá que dirigirse a la sección/pantalla y seleccionar la opción de filtrar

	<p>conciertos, donde tendrá por gustos como: género musical, distancia en km del evento, etc.</p> <p>Una vez aplicado el filtro, le aparecerá una lista con los conciertos más adecuados para el, y seleccionará el que más le interese.</p>
WCGW	<ul style="list-style-type: none"> • Error humano al seleccionar los filtros • No hay conciertos cerca del usuario
Postcondición	Al usuario se le presentará la información del evento escogido.

Conclusiones:

- Muestra y navegación de eventos cercanos
- Filtraje sobre eventos

3. Buscar promocionarse en la ciudad

Actor:	Patrocinador
Precondición:	<ul style="list-style-type: none"> • Registrarse con el rol de patrocinador • Disponer de conexión a internet y gps
Escenario Principal	<p>Si el usuario ya ha iniciado sesión, tendrá que acceder a la sección/pantalla de "compra" de patrocinio. En esta ventana se le ofrecerán distintos tipos de patrocinio, el usuario seleccionará el método que más le convenga.</p> <p>Al seleccionarlo, deberá rellenar una solicitud de patrocinio, que será respondida por el equipo mediante correo electrónico o la misma aplicación.</p>
WCGW	<ul style="list-style-type: none"> • Fallo al rellenar la solicitud • Error en el envío de la solicitud • Retraso en la respuesta por parte de los gestores de la aplicación, resultando una pérdida de interés del patrocinador
Postcondición	El patrocinador sabe si se puede patrocinar o no en la aplicación, y si es el caso realizar el pago debido.

Conclusiones:

- Sistema de Roles
- Integración de compra online
- Sistema de patrocinio

Encuesta

Preparamos una encuesta general enfocada a todo tipo de usuario. Diseñada y publicada en Google Docs. Creando diferentes tipos para diferentes tipos de músicos.

1. ¿Qué Sistema Operativo usas en tu dispositivo móvil?

Podemos ver que un 81,5 % de los participantes de la encuesta usan Android y el 18,5 % usan iOS. Con la obtención de estos resultados podemos ver que hacer inicialmente la aplicación sólo en Android no sería un gran problema ya que la mayoría de los usuarios podrían usar la aplicación sin problemas.

2. ¿Dedicarías tiempo usando una app para resolver tu necesidad ?

Un 92,6 % de usuarios usarían una aplicación de estas características, estos resultados nos hacen intuir que los músicos en general necesitan una aplicación que cumpla con los requisitos necesarios para poder comunicarse mejor entre ellos y así tener más oportunidades en el mundo de la música.

3. ¿Estarías dispuesto a pagar por el uso de la app?

En la encuesta podemos ver claramente que los usuarios en su mayoría, un 85,2 %, no quieren pagar por una app de estas características aunque les resulte positivo en el aspecto musical.

4. ¿Qué forma de pago preferirías?

En el caso que los usuarios tuvieran que pagar más de la mitad de ellos prefieren un solo pago a hacer pagos mensuales o anuales.

5. ¿Tocas algún instrumento (la voz también cuenta como instrumento)?

Un 63 % de los usuarios tocan más de un instrumento por un 29,6 % que sólo toca un instrumento.

6. ¿Estás dispuesto a aprender a tocar un nuevo instrumento musical?

Casi un 50 % de los usuarios tienen pensado aprender a tocar más instrumentos, por otra parte más del 40 % no tienen pensado aprender a tocar otro instrumento aunque no descartan hacerlo en un futuro y menos del 10 % no quieren tocar más instrumentos.

7. ¿Conoces gente de tu alrededor para tocar?

El 81,5 % de los músicos conocen otras personas para poder tocar.

8. ¿Te gustaría conocer nuevas personas con quien tocar?

Un 92,6 % de los usuarios quieren conocer a nuevos usuarios para poder tocar, como resultado quiere decir que aunque tengan contactos, creen que no es suficiente.

9. ¿Asistes a conciertos en locales cercanos?

Alrededor de la mitad de los músicos asisten en ocasiones a conciertos cercanos, por otra parte alrededor del 25 % no van nunca o van frecuentemente.

10. ¿Te interesaría anunciar tu banda o tu música?

Los usuarios en un 50 % cada uno quieren o no quieren anunciar su música públicamente. Hay una parte de los usuarios que quieren mantenerse en el anonimato.

11. ¿Subes o tienes intención de subir contenido como artista en solitario/grupo ?

Más del 60 % de los usuarios tienen intención de publicar música

12. ¿Ves la música como un medio de vida probable?

La mitad de los usuarios creen que se puede vivir de la música pero también buscan otras segundas opciones por si no se llega a triunfar, por otra parte un 30 % creen que no se puede vivir de ello y el resto cree que se puede vivir bien de ello.

<se juntara los resultados de la encuesta en forma de excel>

Conclusiones:

- Nuestros usuarios usan mucho más android
- La mayoría de los usuarios usarían la aplicación
- Los usuarios no quieren pagar , En el caso que tuvieran que pagar prefieren que sea un único pago
- La app debe tener la opción de poder elegir que tocas más de un instrumento, voz incluida
- El perfil del usuario ser tiene que poder modificar en un futuro ya que los usuarios pueden aprender a tocar un instrumento en este espacio de tiempo y así poderlo añadir a su perfil
- Tiene que poder ver y seleccionar los eventos cercanos
- Los usuarios deben poder publicar su música, pero no como opción obligatoria
- Los grupos y músicos han de tener la opción de anunciarse
- La aplicación tiene que estar enfocada en un ámbito semi-profesional ya que tanto los usuarios que creen que pueden vivir de la música como los que no puedan usar la aplicación

Especificación formal de los requisitos

En este apartado vamos a especificar los requisitos adquiridos mediante los diferentes sistemas, encuestas y escenarios. Vamos a profundizar en ellos, describiendo los funcionales, requisitos de usuario, y clasificando los no funcionales en diferentes apartados.

1.El sistema debe tener un método para que los usuarios se registren.



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

1.1 El usuario necesita una pantalla dónde registrarse con sus datos

Requisitos de sistema

1.1.1 Pantalla de Registro : Un formulario, el usuario puede entrar en el campo de identificador de usuario, el teléfono y el correo. Tendrá que introducir una contraseña, dentro de unas normas de seguridad y repetirla. Contendrá un botón para avanzar a la siguiente pantalla

- **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 sino saltará una notificación de error.

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

Requisitos de sistema

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

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

1.3 Debemos asegurar la autenticidad del usuario, con una validación de su registro

Requisitos de sistema

1.3.1 Pantalla de Validación : un formulario con solo un campo, introducir el código de validación. I un botón para comprobarlo, esta pantalla debe aparecer cuando el usuario acabe de completar el registro.

- **1.3.1.2** El código será un código autogenerado de 6 dígitos y con caducidad de 5 minutos.
- **1.3.1.3** El código se enviará a través del medio seleccionado de registro.

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

2.1 El usuario tiene que poder instanciar y modificar su perfil en todo momento.

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.

2.1.2 Pantalla del perfil : una estructura de formulario, con campos obligatorios y campos opcionales para completar su descripción de usuario. Contendrá un botón de "Aplicar Cambios", un botón de vuelta atrás.

- **2.1.1.1 Campos obligatorios :** imagen de perfil, nombre completo, género musical, al menos uno, que instrumentos tocas, al menos uno (pudiendo seleccionar más de uno incluyendo el canto), experiencia como músico, una breve descripción de los usuarios que buscas en la aplicación, e información detallada de tu ubicación y disponibilidad.
- **2.1.1.2 Campos opcionales :** una breve descripción personal, fotografías complementarias, referencias a tu contenido.
- **2.1.1.3** El botón de aplicar cambios, guardará los cambios en el perfil, para ser vistos para otros usuarios

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.

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



La aplicación solo se podrá utilizar si el usuario está conectado a alguna red WIFI

3.1 La aplicación debe pedir permisos al iniciar la aplicación para comprobar que el usuario dispone de conexión.

- **3.1.1** Se presentarán pantallas "Toast notification" "pop-up" típico de android para que confirme los permisos, en caso de no confirmarlos, no se iniciaría la app.

3.2 El sistema debe notificar al usuario en todo momento cuando hay problemas de conexión durante el uso de la aplicación.

- **3.2.1** <como el usuario lo verá>

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.

5. Nuestros usuarios usan mucho más android

5.1 El sistema será programada únicamente en android por el momento.

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



6.1 Nuestro usuario tiene que ser capaz de filtrar sus resultados según diferentes parámetros:

- **6.1.1 Género musical** : El usuario tiene que ser capaz de buscar por distintos tipos género musical.
 - **6.1.2.1** Implementado como una combo box, una lista desplegable que contendrá los diferentes géneros.
 - **6.1.2.2** Antes de que se le presente la lista de posibles géneros, se le hará una recomendación según sus datos.
- **6.1.2 Experiencia como músico** : El usuario tiene que tener la opción de filtrar por la experiencia musical del usuario.
 - **6.1.2.1** Implementado como una escala de valores, en forma de estrellas.
 - **6.1.2.2** Si el campo se deja en blanco, el filtraje se producirá igual, pero sin ordenar la lista de posibles matches por experiencia.
- **6.1.3 Instrumento con habilidades** : El usuario debe poder especificar en qué instrumentos tocan los usuarios que busca.
 - **6.1.3.1** Según la información en el perfil y el género musical seleccionado en el campo anterior , se te recomiendan los instrumentos más adecuados para colaborar.
 - **6.1.3.2** Primero tiene que seleccionar qué tipo de instrumento busca; de viento, percusión, etc.
 - **6.1.3.3** Luego se le presentará una combo box, donde aparecerán todos los instrumentos de esa familia.
 - **6.1.3.4 Cantantes?** : opción de mostrar también cantantes

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.

7.1 El usuario verá el perfil completo si se trata de un match.

- **7.1.2** El perfil mostrará la información, según su importancia, el contenido multimedia y las referencias se presenta como elemento principal (si el perfil contiene) para que directamente el usuario pueda ver en acción a su match. Acompañado de la foto de perfil, su nombre completo y su experiencia como artista musical.
- **7.1.3** Como información importante, presentará una breve descripción del usuario y la descripción de lo que él busca, sus habilidades; instrumentos, géneros musicales en los que toca o se identifica, su sexo,

7.2 La información que se le presentará si no se trata de un match, será la foto de perfil, nombre y apellidos, su sexo.

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



8.1 El usuario solo podrá acceder al perfil de un usuario que no sea su match, con información reducida, pero no podrá ponerse en contacto con él, tiene que ser un match

8.2 El usuario contará con diferentes opciones externas para ponerse en contacto, dependiendo de los datos del perfil

- **8.2.1 Métodos de contacto externos** : Whatsapp, Instagram, correo electrónico y facebook ofrecidos por el usuario en su perfil
- **8.2.2** Se podrá contactar por la misma aplicación, implementaremos un chat para que los usuarios no tengan que usar un método externo de contacto

9. El usuario tiene que poder retroceder, “volver atrás” en todo momento en el proceso.



9.1 En el sistema existirán tipos de interfaces, a parte de la pantalla principal:

- **9.1.1 intermedia** : Hace de medio de comunicación entre dos pantallas, por lo tanto nos mostrará en la parte superior siempre la opción de retroceder a la anterior
- **9.1.2 Finales** : Aquellas pantallas que no te llevan a ningún lado más, por lo tanto mostraremos una cruz para que esta interfaz deje de ser visible y pueda volver a la anterior.
- **9.1.3** Por último habrá pantallas que no requerirán una opción para retroceder, ya que la interacción se realizará en el mismo nivel y no habrá ninguna pantalla a la que retroceder

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** : 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.

11. El usuario tiene que poder filtrar según el evento que le interese asistir Urband

Como hemos dicho en el apartado anterior, habrá un botón de filtros en la parte superior derecha del mapa. En esta pantalla podremos encontrar los requisitos que busca el usuario a la hora de buscar músicos, bandas o eventos. En este caso los filtros sobre los eventos serían:

- 11.1 Género musical: tipo de música o tipos que se tocarán en el evento/concierto.
- 11.2 Si en el evento programado hay más de un grupo, estilo que tocarán, también existirá la opción de seleccionar eventos con más de una actuación.
- 11.3 Tipo de evento: el usuario debe poder escoger la clase del evento; concierto, concierto benéfico, festival, etc.
- 11.4 Tipo de entrada : si la entrada es gratuita, si se trata de un bar dónde debes consumir, si hay precio de entrada, etc.
- 11.5 Ubicación : bar, local, sala de eventos, en el exterior, etc.

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

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:

- 12.1 Rol músico en solitario : los músicos compartirán con los otros usuarios sus perfiles, para así poder encontrar lo que necesiten o publicar su contenido, así como anunciarse. Todo esto depende de las funciones que incluya cada usuario ya que no todos lo configurarán de la misma manera.
 - 12.1.1 El rol de músico en solitario debe contener todas las funciones de contacto; sistema de matching, mapa gps, consulta y contacto con perfiles...
- 12.2 Rol Bandas : mismas características que los músicos pero la búsqueda será mucho más específica para encontrar artistas en solitario que quieran unirse a una banda.
- 12.3 Patrocinadores y propietarios de eventos : los patrocinadores pagarán una suma de dinero para anunciarse en el sitio con la condición de que puedan contactar con músicos y bandas para que toquen en sus locales. Así generar un ciclo donde los músicos puedan hacerse ver por un público y el público generar ingresos en los locales con las consumiciones o las entradas.
 - 12.3.1 El rol de patrocinador no contendrá funciones específicas de músico, pero a parte de ellos, dispondrán de una función de patrocinio.
 - 12.3.1.1 El sistema debe preguntar al usuario interesado en adquirir un patrocinio, qué tipo de patrocinador es; dueño de un local, publicista...
- 12.4 Usuarios sin registrarse/iniciar sesión : éstos usuarios podrán ver en qué lugares se hacen los eventos y así poder asistir a ellos.

13. Nuestro sistema debe integrar diferentes sistemas de pago online. Urband

En la aplicación, se tendrá que poder pagar por varios métodos cuando queramos comprar, esto conlleva una integración de diversos métodos de pagos online.

- 13.1 Integración API de Paypal: Una opción de pago muy usada, usaremos la API para desarrolladores que la misma empresa ofrece de manera gratuita, en este caso la versión para Java.
- 13.2 Integración API de RedSys: Menos conocido, pero permite pagar con tarjeta, una opción que no podemos perder de cara a los clientes. Igual que en la opción anterior, la propia empresa proporciona una API Rest para facilitar la integración.

14. La aplicación ofrecerá un sistema de patrocinio a los usuarios.



14.1 Proporcionaremos al usuario una pantalla, donde inicialmente, después de haberse registrado y hecho login como usuario “patrocinador” se le presentarán las opciones de su patrocinio.

Requisitos de sistema

- **14.1.1 Pantalla de introducción de datos para patrocinar en la app:** en esta primera pantalla aparecerán los datos del patrocinio que tenga activo, si tiene alguno, (fecha, precio, número de veces anunciado..), o un form para comprar su patrocinio, con dos opciones Paypal o Tarjeta.
 - **14.1.1.1** En la opción de **PayPal**, se abrirá una nueva pantalla pidiendo el usuario y la contraseña de paypal del usuario, y un botón para hacer login, aparece un pop up donde se verá la información de autenticación del usuario en paypal y el resultado de la compra. Se retornara a la página de patrocinios si se completa con éxito, o se mostrará un mensaje que indique el error si ha fallado la transacción
 - **14.1.1.2** En la opción de **RedSys**, se abrirá una nueva pantalla pidiendo los datos de la tarjeta como un form, CVV, número de tarjeta y fecha de caducidad, y un botón para hacer procesar la compra, aparece un pop up donde se verá la información y el resultado de la compra. Se retornara a la página de patrocinios si se completa con éxito, o se mostrará un mensaje que indique el error si ha fallado la transacción

15. Los grupos y músicos han de tener la opción de anunciarse



15.1 El sistema debe implementar una sección dónde los usuarios puedan consultar anuncios de otros usuarios.

- **15.1.1** El sistema ofrecerá una pantalla con un “feed”, información en tiempo real de lo que usuarios con los que has hecho match o sigues, subirán a la aplicación.

16. La aplicación tiene que estar enfocada en un ámbito semi-profesional



16.1 La aplicación actuará como “trampolín” conectando músicos o bandas con patrocinadores que puedan ofrecer marketing, locales o contactos para ampliar su carrera musical. Añadirá la posibilidad de conectar para aprender a tocar un instrumento para ampliar su repertorio y agrandar el abanico de posibilidades del músico.

17. Los usuarios deben poder publicar su música, pero no como opción obligatoria

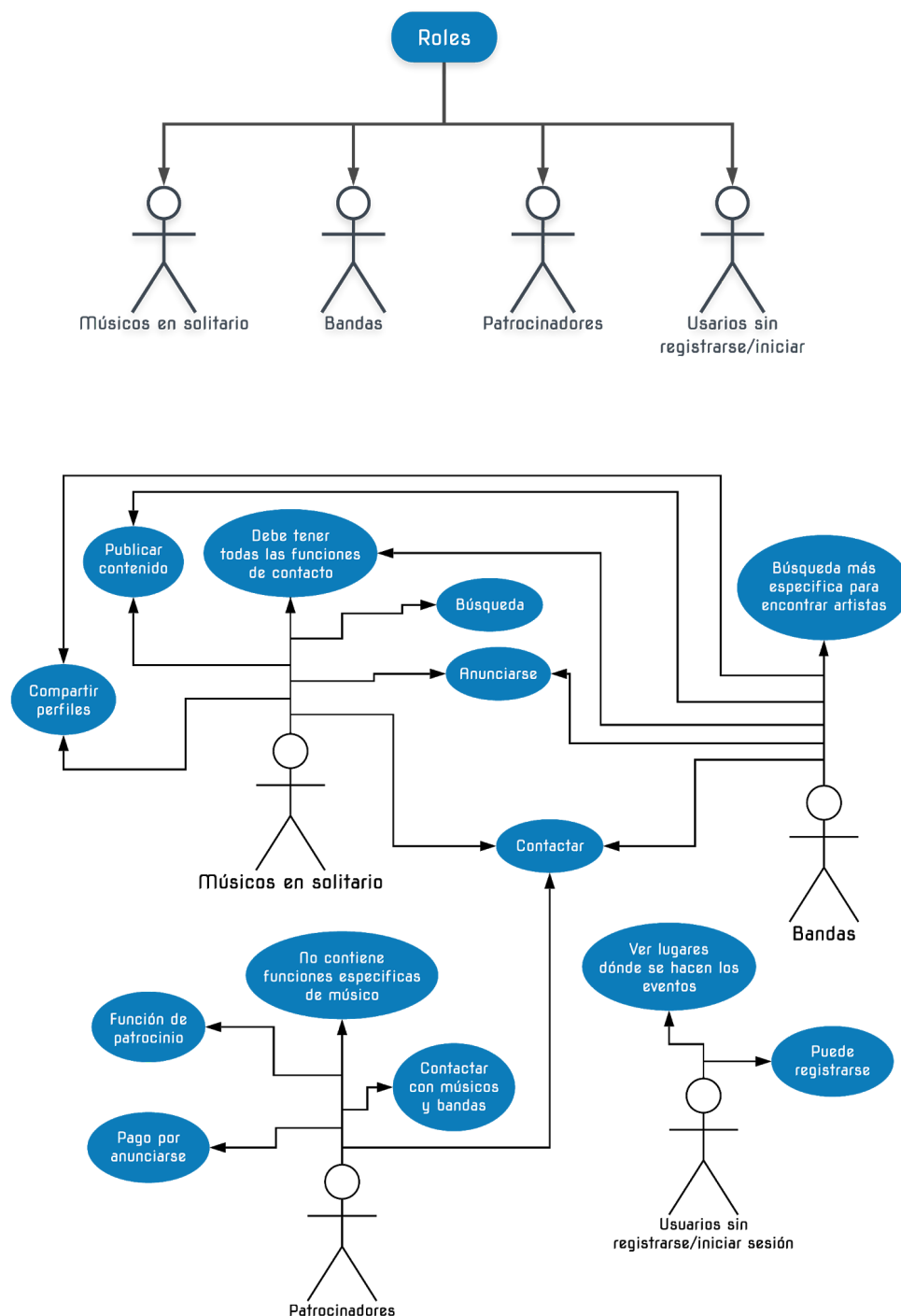


17.1 En la pantalla “social”, se dará la posibilidad de publicar contenido por parte del usuario, apuntando especialmente a música propia para darla a conocer.

- **17.1.1** En esta pantalla, veremos publicaciones de usuarios a los que hayamos dado match o hayamos decidido seguir, las publicaciones se verán ordenadas cronológicamente por la fecha de publicación, será visible el autor de las mismas, y se podrá inspeccionar el perfil del autor de dicha publicación.

Diagrama de casos de uso

En el primer diagrama nos enseña los diferentes roles que tenemos dentro de la aplicación. El segundo diagrama hemos dividido los diferentes roles, los mismos que los del primer diagrama, con sus respectivos casos de uso y enlazando cada uno de ellos con sus funciones



Plan de validación y trazabilidad

Para validar nuestros requisitos, hemos usado diferentes técnicas:


- **Pruebas de validación:** Nos hemos dirigido a nuestro cliente objetivo para recoger y especificar las necesidades que se reflejan en ellos. Hemos realizado una encuesta enfocada al futuro usuario de nuestra aplicación. Con las preguntas que proponemos, dejamos que el usuario se exprese con libertad, para después poner todos los casos en común y lograr extraer qué necesidades debe cubrir nuestra aplicación para que los usuarios se sientan satisfechos de usar nuestra aplicación.
- **Pruebas de consistencia:** Hemos realizado un prototipo virtual de nuestra aplicación para poder comprobar y mejorar aspectos importantes en la usabilidad y la estética. Gracias al prototipo podemos visualizar la aplicación y comprobar que no hay ningún conflicto entre requisitos y que todos son compatibles entre ellos. Hemos buscado a diferentes usuarios genéricos y les hemos ofrecido el prototipo para que se muevan por el y nos den su punto de vista. Los feedbacks de estos usuarios genéricos nos enfocan hacia la manera de diseñar nuestra app de la forma más clara, fácil y usable.
- **Pruebas de completitud:** Teniendo en cuenta las respuestas de los usuarios a las encuestas y las decisiones tomadas a lo largo del tiempo, reflejadas en los documentos, comprobaremos si las funcionalidades y restricciones que decidimos implementar realmente están implementadas y de la forma óptima.
- **Pruebas de realismo:** Para comprobar que nuestros requisitos pueden ser programados y utilizados en nuestra aplicación, los programadores se encargan de analizar estos requisitos para decidir si es posible

implementarlos teniendo en cuenta las limitaciones a las que nos sometemos.

- Pruebas de verificación: Para verificar nuestros requisitos, realizamos:
 - Inspecciones del software (validación estática): Analizaremos las representaciones estáticas para describir los problemas que encontremos,
 - Pruebas del software (verificación dinámica): Observaremos y ejercitaremos el comportamiento de nuestra aplicación. Ejecutaremos la aplicación con datos de prueba y observaremos su comportamiento para verificar el cumplimiento de los requisitos.

Trazabilidad

Para tener un seguimiento de los requisitos, hemos decidido realizar una matriz de trazabilidad con la siguiente composición:

<div>  <div> <div>Identificación</div> <div>Estado</div> <div>Características</div> <div>Objetivo y validación</div> </div> </div>										
Id	Tipo	Versión	Estado	Fecha de estado	Prioridad	Complejidad	Objetivo	Test usado	Resultados tes	
1	Sistema	1.0	Instanciado	26/02/2020	5	3	Sistema de login	...	Fallo total	
1.1	Software	1.0	
1.2	...	1.0	
...	
...	
...	
1	Sistema	1.1	Testeado, mejorable	03/03/2020	5	3	Argelos base de datos	

- Identificación del requisito:
 - Id: Números asociados a cada uno de los requisitos.
 - Tipo: Definimos el tipo de requisito a analizar, de sistema, de software, de usuario...
- Estado del requisito:
 - Versión: en que momento de su desarrollo, análisis se encuentra el requisito

- Estado: Definición corta del comportamiento del requisito, si está finalizado con el visto bueno, si se debe arreglar, corregir o eliminar, etc.
- Fecha de estado: fecha en la que este requisito, con su versión es procesado.
- Características del sistema:
 - Prioridad: qué importancia frente a otros tiene el requisito, se expresa en una escala numérica; 1 como más prioritario y 5 como menos prioritario.
 - Complejidad: qué dificultad presenta este requisito, su versión, para ser solucionado, procesado. Se expresa como una escala numérica: 1 como más difícil y 5 poco complicado
- Objetivo y validación:
 - Objetivo: qué metas queremos lograr en esta fase del requisito
 - Test usado: el método usado para validar o evaluar esta fase.
 - Resultado del test: Resultados de las evaluaciones.

Esta matriz de trazabilidad se irá completando y usando posteriormente a medida de que queramos comprobar y validar nuestros requisitos en futuras implementaciones.

Plan de gestión de la evolución de los requisitos

Una vez tenemos hemos analizado y especificado nuestros requisitos, vamos a planear cómo gestionar-los, es decir, en qué orden, quién hace qué, qué prioridad tienen y cómo gestionamos las crisis.

Planificación

Usaremos una herramienta de gestión llamada Trello, que nos permite determinar “tareas” y asignarlas en tiempo y por responsables. En vez de tareas, nosotros tendremos requisitos. Con esta herramienta damos prioridad a el desarrollo de los requisitos, cuanto más arriba en el “backlog”, una sección donde agrupar los requisitos, más prioridad tendrá el desarrollo del requisito.

La documentación adjunta a cada requisito, irá dentro de esta “tarea” en el trello, ya que se pueden adjuntar ficheros a placer. Además, iremos moviendo cada requisito entre diferentes “estados”, definidos por nosotros mismos, siendo cada estado una sección similar a la del “backlog”, estos estados serán: backlog, pending, development, testing, client pending, done.

Backlog: Estado inicial del requisito, aquí tenemos todos los requisitos inicialmente, y cuando el responsable o los responsables del requisito deban empezar a trabajar en el, pasará de estado.

Pending: Este estado denota que hay que trabajar en el requisito, pero o bien, todos los responsables están ocupados, o se depende de un recurso al que aún no se tiene acceso.

Development: tenemos en este estado, requisitos en los que se está trabajando activamente, y de los que el desarrollo aún no ha finalizado.

Testing: El desarrollo del requisito ha acabado, pero debe de pasar aún por pruebas o tests para asegurar su correcto cumplimiento .

Client Pending: Falta recibir feedback del cliente para saber si se ha completado correctamente y a gusto del cliente el requisito en cuestión.

Done: Último estado de un requisito, una vez llegado aquí, se da por finalizado el trabajo sobre este requisito.

Cabe destacar, que no solo podemos pasar de estados en una “dirección” un requisito puede pasar de testing a development por un cambio o fallo descubierto en una fecha posterior.

Distribución

Como ya se comentaba, podemos asignar mediante el trello “responsables” para cada requisito, pero esto es demasiado genérico. Así, que tendremos un excel con los responsables de cada requisito, con orden en los responsables, por si falla alguien, quien tiene que sustituirlo.

Una vez establecida la jerarquía, podemos usar métodos agile para estimar el esfuerzo que costará completar satisfactoriamente cada requisito, es decir, puntuamos en “horas” cuánto nos costará cada requisito, y con esta información, podremos crear un diagrama de Gantt, donde se verá este esquema de tiempo mucho más claro.

Seguimiento

Aunque el trello ya nos permite un seguimiento “general” del estado de los requisitos, también usaremos herramientas como slack para comentar la evolución de los mismos y tendremos mínimo dos reuniones semanales (horas de clase) para gestionar novedades o ponernos al día en cuanto al cumplimiento de los requisitos.