Documentación Siremu

Proyecto final



Desarrollo de aplicaciones

Luis Ángel Olivo Martínez  
Luis Roberto Herrera Hernández

Contenido

[1. INTRODUCCIÓN 4](#_Toc45275383)

[Objetivo: 4](#_Toc45275384)

[REQUERIMIENTOS 4](#_Toc45275385)

[Contexto 4](#_Toc45275386)

[Recolección y especificación de requerimientos 4](#_Toc45275387)

[Clases de usuarios 5](#_Toc45275388)

[Consumidor de música: 5](#_Toc45275389)

[Creador de contenido: 6](#_Toc45275390)

[Requerimientos funcionales 6](#_Toc45275391)

[Requerimientos no funcionales 7](#_Toc45275392)

[RNF-01 Seguridad 7](#_Toc45275393)

[RNF-01 Rendimiento 8](#_Toc45275394)

[RNF-03 Mantenibilidad 8](#_Toc45275395)

[RNF-04 Portabilidad 8](#_Toc45275396)

[ANÁLISIS Y DISEÑO 8](#_Toc45275397)

[Diseño arquitectónico 8](#_Toc45275398)

[Vista de casos de uso 8](#_Toc45275399)

[Vista lógica 31](#_Toc45275400)

[Vista de implementación 35](#_Toc45275401)

[Vista de Despliegue 37](#_Toc45275402)

[Modelo de datos 38](#_Toc45275403)

[Prototipo UI Cliente 40](#_Toc45275404)

[CONSTRUCCIÓN 53](#_Toc45275405)

[Selección justificada de lenguajes de programación 53](#_Toc45275406)

[Estándares de codificación 53](#_Toc45275407)

[Selección justificada de tecnologías 74](#_Toc45275408)

[Selección justificada de herramientas para el despliegue y manejo de versiones 74](#_Toc45275409)

[Selección justificada de prácticas de construcción 78](#_Toc45275410)

[Prácticas de codificación 78](#_Toc45275411)

[Trabajo en equipo 81](#_Toc45275412)

[Aseguramiento de la calidad 82](#_Toc45275413)

[Herramientas 82](#_Toc45275414)

[PRUEBA 82](#_Toc45275415)

[Plan general de prueba 82](#_Toc45275416)

[Procedimiento de prueba 83](#_Toc45275417)

[Resultado 84](#_Toc45275418)

[Conclusiones 84](#_Toc45275419)

# 1. INTRODUCCIÓN

El presente documento, es una descripción del sistema SIREMU, y marcara las pautas para el desarrollo de este, así como sus especificaciones y diseño

## Objetivo:

El propósito de este documento es proporcionar la información necesaria para controlar el proyecto SIREMU y describir su funcionalidad, además de dotarlo de una base teórica susceptible de revisión, modificaciones y mejoras

Los Objetivos particulares del documento son:

-Definir Los requerimientos funcionales y no funcionales del sistema

-Definir las clases de usuario

-Describir la funcionalidad y estructura general y detallada del sistema

-Especificar como se llevará a cabo el desarrollo del sistema, considerando la manera de trabajar, las tecnologías seleccionadas y los estándares considerados

-Describir las pruebas que se realizaran en el sistema, asi como los resultados de estas

-Redactar las conclusiones, planes para el futuro y reflexiones finales

# REQUERIMIENTOS

## Contexto

El sistema para desarrollar (SIREMU) es un sistema de reproducción de musica, que se desarrolla con el objetivo de permitir la reproducción remota de la musica almacenada en el servidor, se desarrollará como un sistema en red y tendrá soporte de escritorio y móvil.

El sistema permitirá como ya se mencionó, la reproducción de musica remota, las creación de listas de reproducción, manejo de cuentas de usuario, reproducción del historial, subida de canciones públicas, subida de canciones privadas, descarga de canciones y manejo de canciones subidas.

## Recolección y especificación de requerimientos

Para la recolección de los requerimientos, se basó en la especificación proporcionada por el cliente de las funcionalidades requeridas.

Por lo que se analizó la lista de las funcionalidades requeridas y con el análisis y la consulta a los clientes que usaran el sistema (usuarios de reproductores de musica) se obtuvieron los requerimientos que se necesitan en el sistema y para la especificación de los requerimientos, se especificaron con el uso de EARS (easy approach to requirements syntax) por lo que se usó la siguiente plantilla de especificación de requerimientos

La plantilla está compuesta de ocho espacios y cada uno se estructuró pensando en una especificación sintáctica simple y a la vez robusta para abarcar la mayor cantidad de tipos de requisitos en varios tipos de sistemas. Los rectángulos en color amarillo representan condicionales; los rectángulos color gris se usan para representar la familia de sistemas, el sistema o una parte de éste; los rectángulos de color naranja representan el grado de obligación; los de color verde son las actividades que caracterizan el sistema; los rectángulos color azul representan los objetos (sustantivos), con sus respectivas cantidades y complementos; y el rectángulo de color purpura describe el criterio medible de verificación del requisito. Este último es de carácter opcional y por ese la unión con el resto de la plantilla se hace a través de una línea punteada.

Captura de pantalla de un celular

Descripción generada automáticamente

## Clases de usuarios

Los usuarios del sistema son usuarios con poco conocimiento de los sistemas informáticos, y el único rol es de consumidor de musica, aunque se puede pasar al segundo rol (creador de contenido) cuando el usuario entra en dicho modo para subir su musica.

### Consumidor de música:

El consumidor de musica es el usuario más común en el sistema, ya que en realidad todos los usuarios del sistema serán consumidor.

Los consumidores de musica pueden buscar canciones, crear listas de reproducción, buscar listas de reproducción, reproducir musica, descargarla, subir musica privada para escucharla sin conexión, registrarse, iniciar sesión y modificar su perfil.

### Creador de contenido:

El creador de contenido es el segundo usuario que en realidad cuenta con las mismas funciones que el creador de musica, con las adiciones de poder subir canciones al sistema como autor, para que los demás usuarios la escuchen, administrar sus canciones, sus álbumes y consultar su popularidad en las listas de reproducción

## Requerimientos funcionales

RF01: Si el consumidor quiere acceder al sistema entonces el sistema deberá permitir a los usuarios acceder si ingresan sus credenciales válidas.

RF02: Si el consumidor no está registrado, entonces el sistema deberá permitirle registrarse si ingresa su nombre, apellidos, usuario, contraseña, correo y fecha de nacimiento

RF03: Si el consumidor necesita modificar la informacion de su perfil, entonces el sistema deberá permitirle modificar su propia informacion

RF04: Si el consumidor quiere crear una lista de reproducción, entonces el sistema le deberá permitir registrar una nueva con un nombre y una ilustración

RF05: Si el consumidor quiere agregar una canción a una lista de reproducción, entonces el sistema deberá mostrar la lista de las listas del usuario para permitirle agregar la canción a la lista

RF06: Si el consumidor quiere ver sus listas de reproducción, entonces el sistema le deberá mostrar la lista de las listas del usuario

RF07: Si el consumidor quiere ver las listas de reproducción populares, entonces el sistema le deberá mostrar una lista con las listas de reproducción populares actualmente.

RF08: Si el consumidor quiere ver las canciones de una lista de reproducción, entonces el sistema le deberá mostrar una lista de las canciones que conforman la lista de reproducción

RF09: Si el consumidor quiere subir canciones locales para escuchar sin conexión, entonces el sistema le deberá permitir subir una canción al sistema desde su memoria local

RF10: Si el consumidor quiere ver sus canciones locales, entonces el sistema deberá listar las canciones subidas desde su memoria.

RF11: Si el consumidor quiere reproducir una canción, entonces el sistema deberá poder reproducir la canción seleccionada.

RF12: Si el consumidor quiere ver las canciones que ha reproducido, entonces el sistema deberá listar el historial de canciones del usuario

RF13: Si el consumidor quiere buscar una canción o una lista de reproducción, entonces el sistema le deberá permitir buscar por nombre y mostrarle las coincidencias

RF14: Si el consumidor quiere escuchar canciones similares a alguna canción en reproducción, el sistema deberá poder obtener canciones similares y generar una estación de radio

RF15: Si el creador de contenido quiere crear un álbum para sus canciones, entonces el sistema le deberá permitir registrar un álbum nuevo con nombre, ilustración, fecha de lanzamiento y compañía discográfica

RF16: Si el creador de contenido quiere subir una nueva canción, entonces el sistema deberá permitirle subirla con nombre, genero, duración, si es publica o no y si es de promoción o no. Además del archivo

RF17: Si el creador de contenido quiere modificar un álbum, entonces el sistema deberá permitirle hacer la modificación de su álbum

RF18: Si el creador de contenido quiere consultar las listas en donde aparecen sus canciones, entonces el sistema deberá listarle las listas de reproducción donde se encuentra su canción.

RF19: Si el consumidor quiere valorar una canción, el sistema deberá permitirle dar “like” a una canción.

RF20: Si un consumidor inicia sesión, el sistema deberá mostrarle canciones populares y en promoción.

## Requerimientos no funcionales

### RNF-01 Seguridad

Si un usuario entra al sistema, el sistema deberá mantener sus datos y funcionalidades determinadas por su cuenta, impidiendo accesos no deseados a los datos.

### RNF-01 Rendimiento

Si se recibe una solicitud del cliente, el sistema deberá tardar un tiempo mínimo en la respuesta. En un tiempo maximo de 20 segundos.

### RNF-03 Mantenibilidad

Si se va a dar mantenimiento al cliente, el sistema deberá contar con la documentación necesaria para mejorar el proceso del manteniendo.

### RNF-04 Portabilidad

El sistema podrá usarse en computadoras con Windows 10 y dispositivos móviles Android

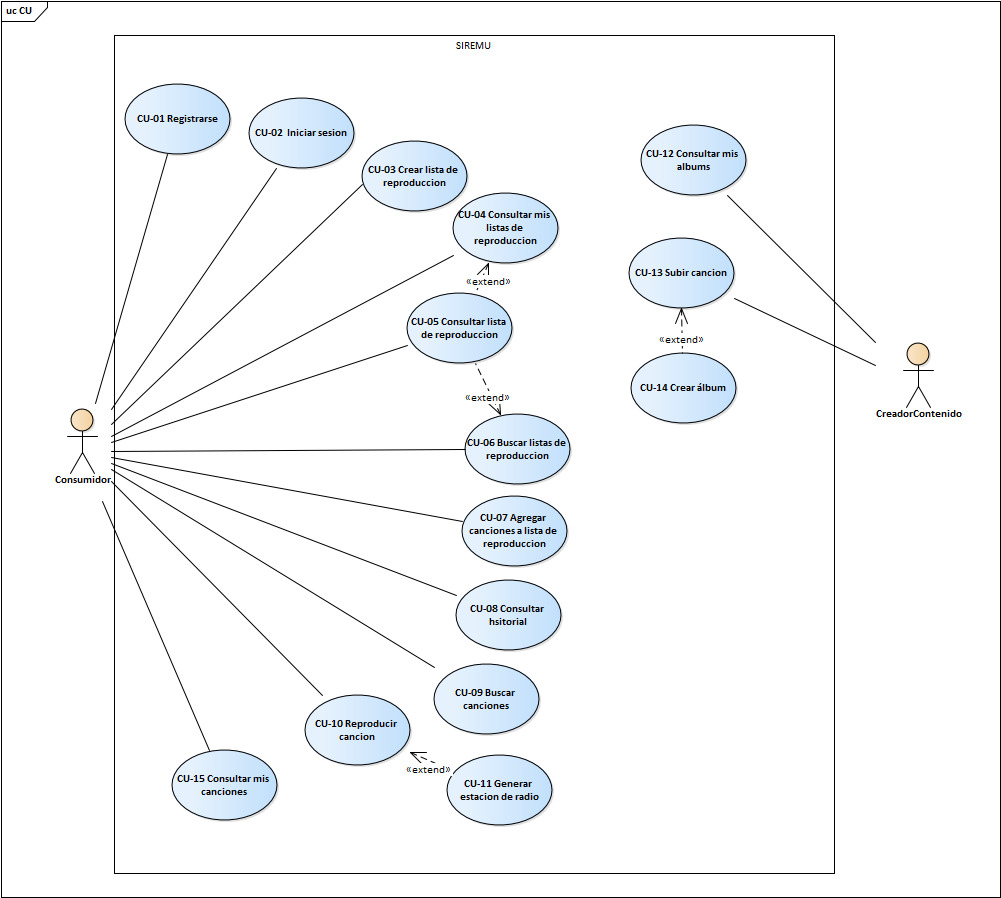
# ANÁLISIS Y DISEÑO

## Diseño arquitectónico

### Vista de casos de uso

#### Modelo de casos de uso

En el siguiente modelo de casos de uso, se representan todos los casos de uso a considerar para el desarrollo del sistema, por lo que son la base para el desarrollo de la funcionalidad.



#### Descripciones de casos de uso

A continuación, se presentan las descripciones de los casos de uso. Estas denotan la interacción de manera escrita del sistema.

CU-01

|  |  |
| --- | --- |
| **­­ID:** | CU01 |
| **Nombre:** | Registrarse |
| **Autor(es):** | Luis Roberto Herrera Hernández |
| **Fecha:** | 07 de abril de 2020 |
| **Actor(es):** | Consumidor |
| **Descripción:** | El sistema permitirá al consumidor registrarse en el sistema. |
| **Precondiciones:** |  |
| **Flujo Normal:** | FN Registrarse   1. El sistema muestra la ventana “Registrarse” y muestra un formulario donde solicita los datos del USUARIO: nombre, apellido, nombre de usuario, fecha de nacimiento, genero, correo y contraseña. También muestra los botones “Registrar” y “Cancelar”. 2. El consumidor ingresa los datos y presiona “Registrar”. (EX2) 3. El sistema valida que se ingresaron todos los datos, crea el USUARIO y lo manda al servidor para que sea guardado en la base de datos con creador de contenido en false. (FA1) (FA2) (EX1) 4. Termina el caso de uso. |
| **Flujos Alternos:** | FA1 Datos incompletos   1. El sistema muestra el mensaje “Todos los datos son obligatorios”. 2. Regresa al paso 2 del flujo normal.   FA2 Correo o nombre de usuario ya repetidos   1. El sistema muestra el mensaje “Ya existe un usuario con ese correo/nombre de usuario. Cámbielo e intente de nuevo”. 2. Regresa al paso 2 del flujo normal. |
| **Excepciones:** | EX1 El Sistema no se pudo conectar con el servidor   1. El sistema muestra el mensaje “No hay conexión con él servidor. Inténtelo de nuevo más tarde”. 2. Termina el caso de uso.   EX2 Cancelar   1. El líder del evento selecciona “Cancelar”. 2. Termina el caso de uso. |
| **Postcondiciones:** | POST-1.- El sistema guardó el USUARIO en la base de datos. |
| **Incluye:** | Ninguno |
| **Extiende:** | Ninguno |

CU-02

|  |  |  |
| --- | --- | --- |
| **ID** | CU-02 | |
| **Nombre** | Iniciar sesión | |
| **Versión** | 1.0 | |
| **Autores** | Luis Ángel Olivo Martínez | |
| **Actor** | Consumidor | |
| **Disparador** | Inicio del sistema | |
| **Descripción** | El caso de uso permite ingresar al sistema con su usuario y contraseña | |
| **Precondición** |  | |
| **Secuencia**  **Normal** | **Paso** | **Acción** |
| 1 | El sistema abre la ventana “Login” donde muestra un formulario para ingresar Usuario y contraseña, además de los botones “Iniciar sesión” y “Registrarse” |
| 2 | El Consumidor llena el formulario y presiona “Iniciar sesión” (FA1) |
| 3 | El sistema solicita al servidor verificar si los datos son correctos (EX1) (FA2) |
| 4 | El servidor devuelve la confirmación y los datos del usuario |
| 5 | El sistema ingresa al menú principal del sistema y lo muestra al usuario |
| **Postcondición** | Los datos no se ven alterados | |
| **Flujos alternos**  **(FA1) se presiona “Registrarse”** | **Paso** | **Acción** |
| 1 | Extiende al caso de uso “Registrarse” |
| 2 | Regresa al paso 1 del flujo normal |
| **(FA2) Datos inválidos** | **Paso** | **Acción** |
| 1 | El sistema escribe un mensaje indicando que los datos son inválidos, que los revise y lo intente de nuevo |
| 2 | Regresa al paso 2 del flujo normal |
| **Excepciones**  (EX1) Error al conectar con el servidor | **Paso** | **Acción** |
| 1 | El sistema manda un mensaje indicando que ocurrió un error al conectar con el servidor |
| 2 | Termina el FN |
| 3 |  |
| **Importancia** |  | |
| **Comentarios** |  | |

CU-03

|  |  |  |
| --- | --- | --- |
| **ID** | CU-03 | |
| **Nombre** | Crear lista de reproducción | |
| **Versión** | 1.0 | |
| **Autores** | Luis Ángel Olivo Martínez | |
| **Actor** | Consumidor | |
| **Disparador** | Presionar el botón crear nueva lista de reproducción | |
| **Descripción** | El caso de uso le permite a un consumidor crear una nueva lista de reproducción | |
| **Precondición** |  | |
| **Secuencia**  **Normal** | **Paso** | **Acción** |
| 1 | El sistema abre una ventana emergente “CrearListaReproduccion” en donde muestra un formulario para ingresar el nombre de la lista de reproducción, subir una imagen, agregar una descripción y seleccionar si será pública o privada, con los botones “Agregar” y “Cancelar” |
| 2 | El Consumidor llena el campo y selecciona el tipo y presiona “Agregar” (EX2) |
| 3 | El sistema recoge los datos, crea un nuevo LISTAREPRODUCCION con los datos, lo asocia al usuario que la creó y lo manda a guardar en la base de datos del servidor (EX1) |
| **Postcondición** | Se guarda un nuevo LISTAREPRODUCCION en la base de datos del servidor | |
| **Excepciones**  (EX1) Error al conectar con el servidor | **Paso** | **Acción** |
| 1 | El sistema manda un mensaje indicando que ocurrió un error al conectar con el servidor |
| 2 | Termina el FN |
| 3 |  |
| **(EX2) Se presiona cancelar** | **Paso** | **Acción** |
| 1 | Termina el FN |
| **Importancia** | Media | |
| **Comentarios** |  | |

CU-04

|  |  |
| --- | --- |
| **­­ID:** | CU-04 |
| **Nombre:** | Consultar mis listas de reproducción |
| **Autor(es):** | Luis Roberto Herrera Hernández |
| **Fecha:** | 07 de abril de 2020 |
| **Actor(es):** | Consumidor |
| **Descripción:** | El sistema permitirá al consumidor consultar las LISTAREPRODUCCIÓN creadas por él mismo. |
| **Precondiciones:** | El consumidor ha creado alguna LISTAREPRODUCCIÓN |
| **Flujo Normal:** | FN Consultar mis listas de reproducción   1. El sistema muestra la ventana “Mis playlists”, recupera de la base de datos todas las LISTAREPRODUCCIÓN subidas por el consumidor y las muestra en una lista con su nombre, si es pública o privada, y en caso de ser pública también muestra su número de likes. (EX1) 2. El consumidor observa las LISTAREPRODUCCIÓN. 3. Termina el caso de uso. |
| **Flujos Alternos:** | FA1 Consultar lista de reproducción   1. El consumidor selecciona una LISTAREPRODUCCIÓN. 2. Extiende al caso de uso CU-05 Consultar lista de reproducción. |
| **Excepciones:** | EX1 El Sistema no se pudo conectar con el servidor   1. El sistema muestra el mensaje “No hay conexión con el servidor. Inténtelo de nuevo más tarde”. 2. Termina el caso de uso. |
| **Postcondiciones:** | POST-1.- El sistema recuperó y mostró todas las LISTAREPRODUCCIÓN creadas por el consumidor. |
| **Incluye:** | Ninguno |
| **Extiende:** | CU-05 Consultar lista de reproducción |

CU-05

|  |  |  |
| --- | --- | --- |
| **ID** | CU-05 | |
| **Nombre** | Consultar lista de reproducción | |
| **Versión** | 1.0 | |
| **Autores** | Luis Ángel Olivo Martínez | |
| **Actor** | Consumidor | |
| **Disparador** | Se presiona clic sobre una lista de reproducción | |
| **Descripción** | El caso de uso permite abrir una lista de reproducción, y permite ver las canciones que tiene. | |
| **Precondición** | La lista de reproducción elegida tiene canciones asociadas. | |
| **Secuencia**  **Normal** | **Paso** | **Acción** |
| 1 | El sistema abre la ventana “VistaPlaylist” y muestra el nombre de la lista de reproducción arriba y también recupera del servidor las canciones de la lista de reproducción elegida y muestra su nombre, nombre de su álbum y duración en minutos. (EX1) |
| 2 | El Consumidor observa las canciones. (FA1) (FA2) |
| 3 | Fin del caso de uso. |
| **Postcondición** | Los datos no se ven alterados | |
| **Flujos alternos**  **(FA1) Reproducir canción** | **Paso** | **Acción** |
| 1 | El consumidor selecciona una canción de la lista. |
| 2 | Extiende al caso de uso CU-10 Reproducir canción. |
| **(FA2) Agregar canciones a una lista de reproducción** | **Paso** | **Acción** |
| 1 | El consumidor da un clic largo sobre una canción de la lista. |
| 2 | Extiende al caso de uso CU-07 Agregar canciones a lista de reproducción. |
| **Excepciones**  (EX1) Error al conectar con el servidor | **Paso** | **Acción** |
| 1 | El sistema manda un mensaje indicando que ocurrió un error al conectar con el servidor |
| 2 | Termina el FN |
| **Importancia** |  | |
| **Comentarios** |  | |

CU-06

|  |  |
| --- | --- |
| **­­ID:** | CU-06 |
| **Nombre:** | Buscar listas de reproducción |
| **Autor(es):** | Luis Roberto Herrera Hernández |
| **Fecha:** | 07 de abril de 2020 |
| **Actor(es):** | Consumidor |
| **Descripción:** | El sistema permitirá al consumidor buscar LISTAREPRODUCCIÓN públicas por su nombre. |
| **Precondiciones:** | Hay LISTAREPRODUCCIÓN públicas registradas. |
| **Flujo Normal:** | 1. FN Buscar listas de reproducción 2. El sistema muestra la ventana “Buscar”, y muestra una barra de búsqueda y los botones “Canciones” y “Playlists”. 3. El consumidor ingresa de forma completa o parcial el nombre de alguna LISTAREPRODUCCIÓN y presiona el botón “Listas”. 4. El sistema recupera del servidor todas las LISTAREPRODUCCIÓN que coincidan con el nombre introducido y las muestra en una lista con su nombre, el nombre de usuario de su creador y su número de likes. (EX1) 5. Termina el caso de uso. |
| **Flujos Alternos:** | FA1 Consultar lista de reproducción   1. El consumidor selecciona una LISTAREPRODUCCIÓN. 2. Extiende al caso de uso CU-05 Consultar lista de reproducción. |
| **Excepciones:** | EX1 El Sistema no se pudo conectar con el servidor   1. El sistema muestra el mensaje “No hay conexión con el servidor. Inténtelo de nuevo más tarde”. 2. Termina el caso de uso. |
| **Postcondiciones:** | POST-1.- El sistema recuperó y mostró todas las LISTAREPRODUCCIÓN correspondientes con la búsqueda. |
| **Incluye:** | Ninguno |
| **Extiende:** | CU-05 Consultar lista de reproducción |

CU-07

|  |  |  |
| --- | --- | --- |
| **ID** | CU-07 | |
| **Nombre** | Agregar canción a lista de reproducción | |
| **Versión** | 1.0 | |
| **Autores** | Luis Ángel Olivo Martínez | |
| **Actor** | Consumidor | |
| **Disparador** | Se da un clic largo sobre una canción. | |
| **Descripción** | El caso de uso permite agregar una o más canciones a una lista de reproducción hecha por el usuario. | |
| **Precondición** | Hay canciones públicas existentes.  El usuario tiene listas de reproducción creadas. | |
| **Secuencia**  **Normal** | **Paso** | **Acción** |
| 1 | El sistema abre la ventana de selección de canciones donde muestra los botones de imagen, “Me gusta” y “Agregar a lista”, además de una lista de selección múltiple con las canciones de la ventana anterior. |
| 2 | El Consumidor selecciona todas las canciones deseadas de la lista y luego pulsa el botón de imagen de “Agregar a lista”. (FA1) |
| 3 | El sistema recupera del servidor las listas de reproducción del usuario. Luego abre la ventana emergente “Agregar a lista” donde abre un combo box para seleccionar la lista a la que se va a agregar, además del botón “Cancelar”. (EX1) |
| 4 | El Consumidor selecciona la lista de reproducción a la que desea agregar las canciones. (EX2) |
| 5 | El sistema agrega las canciones a la lista seleccionada y manda a guardar los cambios al servidor. (EX1) |
| **Postcondición** | Se guarda la relación de las canciones con la lista en el servidor. | |
| **Flujos alternos**  **(FA1) Presiona “Me gusta”** | **Paso** | **Acción** |
| 1 | El sistema agrega las canciones seleccionadas a la lista de “Me gusta” del usuario y manda a guardar los cambios al servidor. |
| 2 | Termina el caso de uso. |
| **Excepciones**  (EX1) Error al conectar con el servidor | **Paso** | **Acción** |
| 1 | El sistema manda un mensaje indicando que ocurrió un error al conectar con el servidor |
| 2 | Termina el FN |
| **(EX2) Se presiona cancelar** | **Paso** | **Acción** |
| 1 | Termina el FN |
| **Importancia** |  | |
| **Comentarios** |  | |

CU-08

|  |  |  |
| --- | --- | --- |
| **ID** | CU-08 | |
| **Nombre** | Consultar historial | |
| **Versión** | 1.0 | |
| **Autores** | Luis Angel Olivo Martínez | |
| **Actor** | Consumidor | |
| **Disparador** | Se presiona clic sobre el botón “Ver historial” | |
| **Descripción** | El caso de uso permite mostrar el historial de canciones reproducidas por el consumidor | |
| **Precondición** | El Consumidor debe haber reproducido alguna canción | |
| **Secuencia**  **Normal** | **Paso** | **Acción** |
| 1 | El sistema abre la ventana “Historial”, solicita al servidor la lista de CANCION que el Consumidor ha reproducido y las muestra en una lista de las canciones con la imagen del álbum, con la duración |
| 2 | El Consumidor presiona una canción (FA1) |
| 3 | El sistema solicita al servidor la canción seleccionada y extiende al caso de uso “Reproducir canción” (EX1) |
| **Postcondición** | Los datos no se ven alterados | |
| **Excepciones**  (EX1) Error al conectar con el servidor | **Paso** | **Acción** |
| 1 | El sistema manda un mensaje indicando que ocurrió un error al conectar con el servidor |
| 2 | Termina el FN |
| 3 |  |
| **Importancia** |  | |
| **Comentarios** |  | |

CU-09

|  |  |
| --- | --- |
| **­­ID:** | CU-09 |
| **Nombre:** | Buscar canciones |
| **Autor(es):** | Luis Roberto Herrera Hernández |
| **Fecha:** | 07 de abril de 2020 |
| **Actor(es):** | Consumidor |
| **Descripción:** | El sistema permitirá al consumidor buscar CANCION públicas por su nombre. |
| **Precondiciones:** | Hay CANCION públicas registradas. |
| **Flujo Normal:** | 1. FN Buscar listas de reproducción  2. El sistema muestra la ventana “Buscar”, y muestra una barra de búsqueda y los botones “Canciones” y “Playlists”.  3. El consumidor ingresa de forma completa o parcial el nombre de alguna CANCION y presiona el botón “Canciones”.  4. El sistema recupera del servidor todas las CANCION que coincidan con el nombre introducido y las muestra en una lista con su nombre, artista y duración. (EX1)  5. Termina el caso de uso. |
| **Flujos Alternos:** | FA1 Reproducir canción  1. El consumidor selecciona una CANCION.  2. Extiende al caso de uso “Reproducir canción”. |
| **Excepciones:** | EX1 El Sistema no se pudo conectar con el servidor   1. El sistema muestra el mensaje “No hay conexión con el servidor. Inténtelo de nuevo más tarde”. 2. Termina el caso de uso. |
| **Postcondiciones:** | POST-1.- El sistema recuperó y mostró todas las LISTAREPRODUCCIÓN correspondientes con la búsqueda. |
| **Incluye:** | Ninguno |
| **Extiende:** | CU-05 Consultar lista de reproducción |

CU-10

|  |  |  |
| --- | --- | --- |
| **ID** | CU-10 | |
| **Nombre** | Reproducir canción | |
| **Versión** | 1.0 | |
| **Autores** | Luis Ángel Olivo Martínez | |
| **Actor** | Consumidor | |
| **Disparador** | Se presiona clic en alguna canción | |
| **Descripción** | El caso de uso permite reproducir una canción seleccionada | |
| **Precondición** |  | |
| **Secuencia**  **Normal** | **Paso** | **Acción** |
| 1 | El sistema muestra la ventana “Reproducción en curso” con la imagen y nombre del álbum de la canción, y el nombre, género y artista de la canción como título. También solicita el archivo de la CANCION al servidor para ser reproducido (EX1) |
| 2 | El Servidor devuelve el archivo |
| 3 | El sistema reproduce el archivo y cambia el botón de “Play” de SiReMu por el de “Pausa” y se muestran los botones de “Anterior Canción” y de “Siguiente Canción” |
| **Postcondición** | Los datos no se ven alterados | |
| **Flujos alternos**  **(FA1) se presiona “Anterior Canción”** | **Paso** | **Acción** |
| 1 | El sistema recupera los datos de la CANCION anterior de la lista y solicita el archivo al servidor (EX1) |
| **Flujos alternos**  **(FA2) se presiona “Siguiente Canción”** | **Paso** | **Acción** |
| 1 | El sistema recupera los datos de la CANCION siguiente de la lista y solicita el archivo al servidor  (EX1) |
| **(FA3) Se presiona el botón “Pausa”** | **Paso** | **Acción** |
| 1 | El sistema guarda el estado de la reproducción y detiene la reproducción |
| 2 | El Consumidor presiona clic en “play” |
| **Excepciones**  (EX1) Error al conectar con el servidor | **Paso** | **Acción** |
| 1 | El sistema manda un mensaje indicando que ocurrió un error al conectar con el servidor |
| 2 | Termina el FN |
| **Importancia** |  | |
| **Comentarios** |  | |

CU-11

|  |  |  |
| --- | --- | --- |
| **ID** | CU-11 | |
| **Nombre** | Generar Estación de radio | |
| **Versión** | 1.0 | |
| **Autores** | Luis Ángel Olivo Martínez | |
| **Actor** | Consumidor | |
| **Disparador** | Se presiona clic sobre el botón “Generar estación de radio” | |
| **Descripción** | El caso de uso permite crear una lista con canciones de géneros similares a la canción en reproducción | |
| **Precondición** | Debe haber una canción en reproducción | |
| **Secuencia**  **Normal** | **Paso** | **Acción** |
| 1 | El sistema abre la ventana “RadioCancion” y solicita al servidor una lista de CANCION con el mismo género que la actual (EX1) |
| 2 | El servidor devuelve la lista |
| 3 | El sistema muestra la lista de las canciones con su nombre, artista, imagen de su álbum, y duración. |
| 4 | El consumidor observa las canciones. (FA1) |
| **Postcondición** | Los datos no se ven alterados | |
| **Flujos alternos**  **(FA1) Reproducir canción** | **Paso** | **Acción** |
| 1 | El consumidor selecciona una canción de la lista. |
| 1 | Extiende al caso de uso CU-10 Reproducir canción. |
| **Excepciones**  (EX1) Error al conectar con el servidor | **Paso** | **Acción** |
| 1 | El sistema manda un mensaje indicando que ocurrió un error al conectar con el servidor |
| 2 | Termina el FN |
| **Importancia** |  | |
| **Comentarios** |  | |

CU-12

|  |  |
| --- | --- |
| **­­ID:** | CU-12 |
| **Nombre:** | Consultar mis álbumes |
| **Autor(es):** | Luis Roberto Herrera Hernández |
| **Fecha:** | 07 de abril de 2020 |
| **Actor(es):** | Creador de contenido |
| **Descripción:** | El sistema permitirá al creador de contenido consultar los ALBUM creados por él mismo. |
| **Precondiciones:** | El consumidor ha creado algún ALBUM |
| **Flujo Normal:** | FN Consultar mis álbumes   1. El sistema muestra la ventana “Mis álbumes”, recupera del servidor todos los ALBUM creados por el creador de contenido y muestra en una lista con su nombre, compañía discográfica y fecha de lanzamiento. (EX1) 2. El consumidor observa los ALBUM y selecciona uno. 3. El sistema abre la ventana “VistaAlbum” y recupera del servidor todas las CANCION del ALBUM seleccionado. Muestra el nombre, artista y duración de cada una. 4. Termina el caso de uso. |
| **Flujos Alternos:** |  |
| **Excepciones:** | EX1 El Sistema no se pudo conectar con el servidor   1. El sistema muestra el mensaje “No hay conexión con el servidor. Inténtelo de nuevo más tarde”. 2. Termina el caso de uso. |
| **Postcondiciones:** | POST-1.- El sistema recuperó y mostró todas los ALBUM creados por el creador de contenido. |
| **Incluye:** | Ninguno |
| **Extiende:** | CU-05 Consultar lista de reproducción |

CU-13

|  |  |
| --- | --- |
| **­­ID:** | UC13 |
| **Nombre:** | Subir canción |
| **Autor(es):** | Luis Roberto Herrera Hernández |
| **Fecha:** | 07 de abril de 2020 |
| **Actor(es):** | Creador de contenido |
| **Descripción:** | El sistema permitirá al creador de contenido subir una nueva CANCION al sistema. |
| **Precondiciones:** | Hay ÁLBUM registrados. |
| **Flujo Normal:** | FN Subir canción   1. El sistema muestra la ventana “Subir canción” y recupera del servidor todos los ÁLBUM creados por el creador de contenido y muestra sus nombres en una lista desplegable. También muestra un formulario donde solicita los datos de la CANCION: nombre, genero, el ÁLBUM y el archivo de la CANCION. Por último, muestra los botones “Agregar”, “Nuevo álbum” y “Subir archivo”. 2. El creador de contenido ingresa los datos, selecciona un ÁLBUM y presiona “Subir archivo”. (EX2) 3. El sistema abre el explorador de archivos del dispositivo. 4. El creador de contenido elige un archivo de música. 5. El sistema valida el archivo y muestra el nombre del mismo. 6. El creador de contenido presiona “Agregar”. 7. El sistema valida que se ingresaron todos los datos, crea y guarda la CANCION en el servidor con los atributos duración con la del archivo, pública en true y promoción en false. (FA1) (EX1) 8. Termina el caso de uso. |
| **Flujos Alternos:** | FA1 Datos incompletos   1. El sistema muestra el mensaje “Todos los datos son obligatorios”. 2. Regresa al paso 2 del flujo normal.   FA2 Nuevo álbum   1. El creador de contenido presiona “Nuevo álbum”. 2. Extiende al caso de uso CU-14 Crear álbum. |
| **Excepciones:** | EX1 El Sistema no se pudo conectar con el servidor   1. El sistema muestra el mensaje “No hay conexión con el servidor. Inténtelo de nuevo más tarde”. 2. Termina el caso de uso. |
| **Postcondiciones:** | POST-1.- El sistema guardó la CANCION en la base de datos. |
| **Incluye:** | Ninguno |
| **Extiende:** | Ninguno |

CU-14

|  |  |
| --- | --- |
| **­­ID:** | UC14 |
| **Nombre:** | Crear álbum |
| **Autor(es):** | Luis Roberto Herrera Hernández |
| **Fecha:** | 07 de abril de 2020 |
| **Actor(es):** | Creador de contenido |
| **Descripción:** | El sistema permitirá al creador de contenido registrar un nuevo ÁLBUM en el sistema. |
| **Precondiciones:** | El creador de contenido debe tener su USUARIO registrado. |
| **Flujo Normal:** | FN Crear álbum   1. El sistema muestra la ventana “Nuevo álbum” y muestra un formulario donde solicita los datos del ÁLBUM: nombre, fecha de lanzamiento, compañía discográfica y el archivo de la ilustración. También muestra los botones “Subir archivo” y “Agregar”. 2. El creador de contenido ingresa los datos y presiona “Subir archivo”. (EX2) 3. El sistema abre el explorador de archivos del dispositivo. 4. El creador de contenido elige una imagen. 5. El sistema valida la imagen. 6. El creador de contenido presiona “Agregar”. 7. El sistema verifica que se ingresaron todos los datos, crea el ÁLBUM y lo manda a guardar al servidor. (FA1) (EX1) 8. Termina el caso de uso. |
| **Flujos Alternos:** | FA1 Datos incompletos   1. El sistema muestra el mensaje “Todos los datos son obligatorios”. 2. Regresa al paso 2 del flujo normal. |
| **Excepciones:** | EX1 El Sistema no se pudo conectar con el servidor   1. El sistema muestra el mensaje “No hay conexión con el servidor. Inténtelo de nuevo más tarde”. 2. Termina el caso de uso. |
| **Postcondiciones:** | POST-1.- El servidor guardó el nuevo ALBUM. |
| **Incluye:** | Ninguno |
| **Extiende:** | Ninguno |

CU-15

|  |  |
| --- | --- |
| **­­ID:** | UC15 |
| **Nombre:** | Consultar mis canciones |
| **Autor(es):** | Luis Roberto Herrera Hernández |
| **Fecha:** | 07 de abril de 2020 |
| **Actor(es):** | Consumidor |
| **Descripción:** | El sistema permitirá al consumidor consultar las CANCION que tiene en todas sus LISTAREPRODUCCION. |
| **Precondiciones:** | El consumidor tiene LISTAREPRODUCCION con CANCION asignadas. |
| **Flujo Normal:** | FN Consultar mis canciones   1. El sistema muestra la ventana “Mis canciones”, recupera de la base de datos todas las CANCION que tiene en todas sus LISTAREPRODUCCION el consumidor y las muestra en una lista con su nombre, imagen del álbum, artista y duración. (EX1) 2. El consumidor observa las canciones. 3. Termina el caso de uso. |
| **Flujos Alternos:** | FA1 Reproducir canción   1. El consumidor selecciona una CANCION. 2. Extiende al caso de uso CU-10 Reproducir canción. |
| **Excepciones:** | EX1 El Sistema no se pudo conectar con el servidor   1. El sistema muestra el mensaje “No hay conexión con el servidor. Inténtelo de nuevo más tarde”. 2. Termina el caso de uso. |
| **Postcondiciones:** | POST-1.- El sistema recuperó y mostró todas las CANCION que se encuentran en alguna LISTAREPRODUCCION del consumidor. |
| **Incluye:** | Ninguno |
| **Extiende:** | Ninguno |

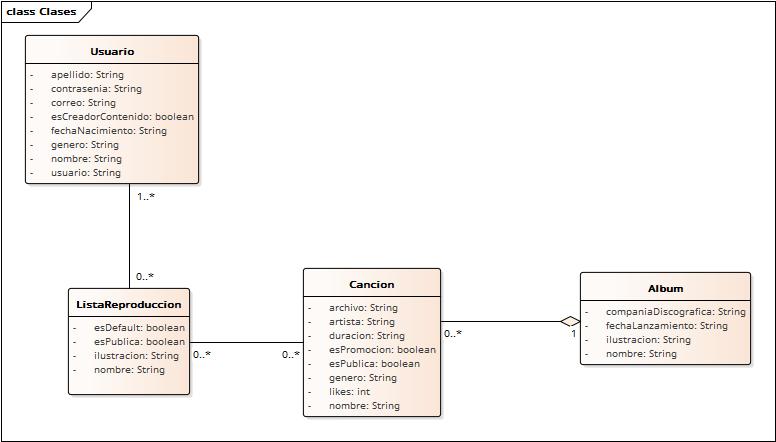
### Vista lógica

#### Diagrama UML de clases

Para mostrar las clases que usará el sistema, se desarrolló un diagrama UML de clases.

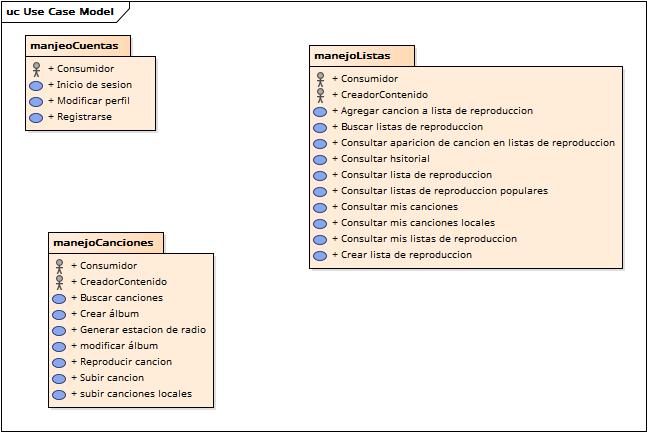
El diagrama UML de clases es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

En este caso, las clases funcionan como contenedores y no tienen métodos propios aparte de Get y Set

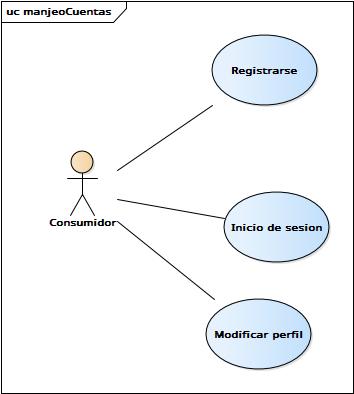


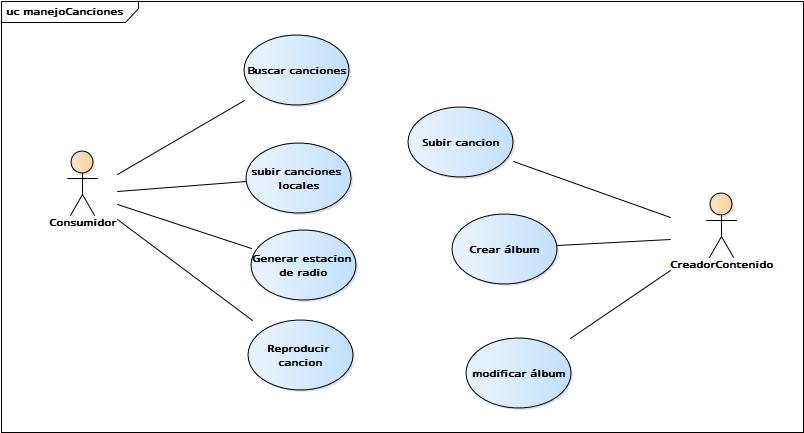
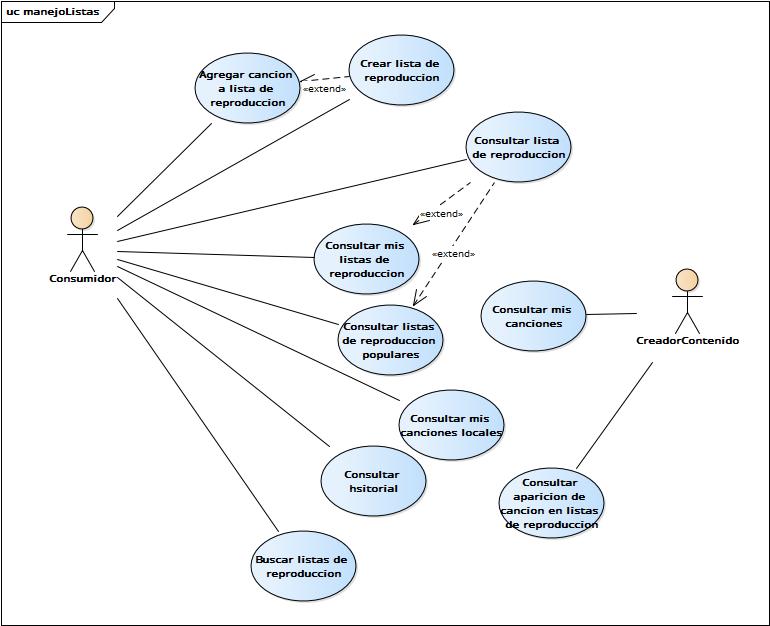
#### Diagrama UML de paquetes

Para la organización de los casos de uso, se desarrolló un diagrama de paquetes. Los paquetes de casos de uso del sistema, fueron divididos de acuerdo a la funcionalidad que proporcionan respecto al manejo de cuentas, manejo de listas y manejo de canciones.



A continuación, se muestra el detalle de cada paquete.





### Vista de implementación

#### Diagrama UML de componentes

Para representar los componentes que conformarán el sistema, se desarrolló un diagrama UML de componentes.

Mientras que otros diagramas UML describen la funcionalidad de un sistema, los diagramas de componentes se utilizan para modelar los componentes que ayudan a hacer esas funcionalidades, representando la forma en la que estos se organizan y sus dependencias.

Como se puede observar en el siguiente diagrama, el sistema contará con diferentes componentes que en conjunto conformarán el sistema SIREMU.

A continuación se describen brevemente los componentes:

SiremuMovil:

Es la aplicación móvil del sistema. Esta aplicación se ejecuta en un teléfono y requiere comunicacion con los componentes ServerCanciones, ServerCuentas y ServerListas para obtener los datos que usará

SiremuDesktop:

Es la aplicación de escritorio del sistema, junto con SiremuMovil conforman el cliente del sistema y al igual que SiremuMovil requiere comunicación con los componentes ServerCanciones, ServerCuentas y ServerListas para obtener los datos que usará

ServerCanciones:

Este componente del sistema es el servidor que se encarga del manejo de las canciones por lo que se recibe las solicitudes de los clientes referentes a el manejo de canciones, consulta los datos en la base de datos y los devuelve al cliente, o recibe los datos de los clientes y los guarda en la base de datos

ServerCuentas:

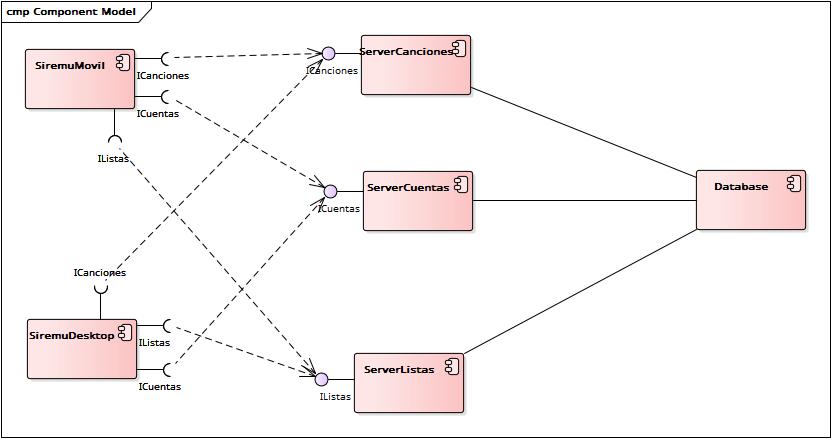
Este componente del sistema es el servidor que se encarga del manejo de las cuentas de los usuarios, por lo que recibe las solicitudes de inicio de sesión, registro y modificación del perfil de los clientes y con los datos almacenados en la base de datos, responde a las solicitudes de los clientes.

ServerListas:

Este componente del sistema es el servidor que se encarga del manejo de las listas de los clientes, por lo que recibirá las solicitudes de gestión de las listas almacenadas por los clientes y con los datos de la base de datos, responderá adecuadamente a las diferentes solicitudes que realicen los clientes.

Database:

Este componente es la base de datos del sistema y almacenará todos los datos del sistema para que puedan ser consultados y modificados por los componentes de servidor del sistema.



##### Descripción de las interfaces

A continuación se describen las interfaces provistas y requeridas de los componentes del diagrama de componentes.



Estas interfaces se encuentran implementadas con el Protocol Buffers Version 3, por lo que se definen mensajes para la comunicación con grpc y con ellos se comunican los diferentes componentes del sistema.

### Vista de Despliegue

#### Diagrama UML de despliegue

Para explicar el despliegue que se hará del sistema, se desarrollo el modelo UML de despliegue.

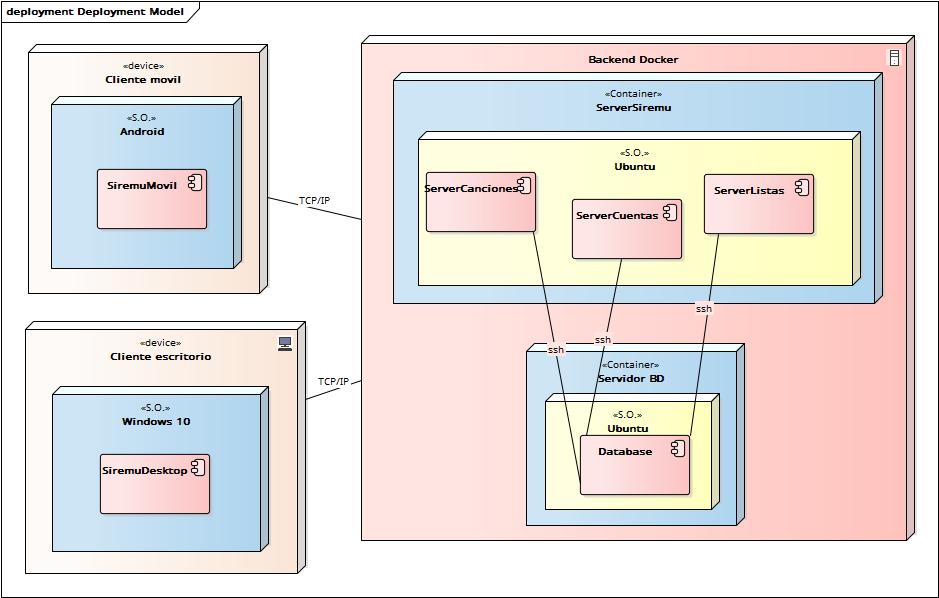
El Diagrama de Despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar la disposición física de los artefactos software en nodos.​Muestra la arquitectura del sistema como el despliegue de los artefactos de software a los objetivos de despliegue

El despliegue de la aplicación se hará de la siguiente manera

Se desplegará el cliente de la aplicación en dispositivos móviles y de escritorio con sistema operativo Android o Windows 10 respectivamente, el despliegue del cliente Android será continuo gracias a la posibilidad que ofrece Github, por lo que solo se deberá descargar la aplicación generada.

En el caso del servidor que está dividido en sus componentes, se desplegara en 2 contenedores Docker orquestados por Docker compose. Por lo que se creará una red de área virtual con Docker en donde se conectarán los contenedores del servidor y de la base de datos. De esta manera el componente del servidor podrá conectarse a la base de datos por medio de la red Docker. El cliente se conectará a la red de Docker para contactar con el servidor y así resolver sus solicitudes.

A continuación se muestra el diagrama de despliegue del sistema.



## Modelo de datos

Para representar los datos que manejará el sistema, se desarrolló el modelo de datos correspondiente

El modelado de datos es una manera de estructurar y organizar los datos para que se puedan utilizar fácilmente por las bases de datos.

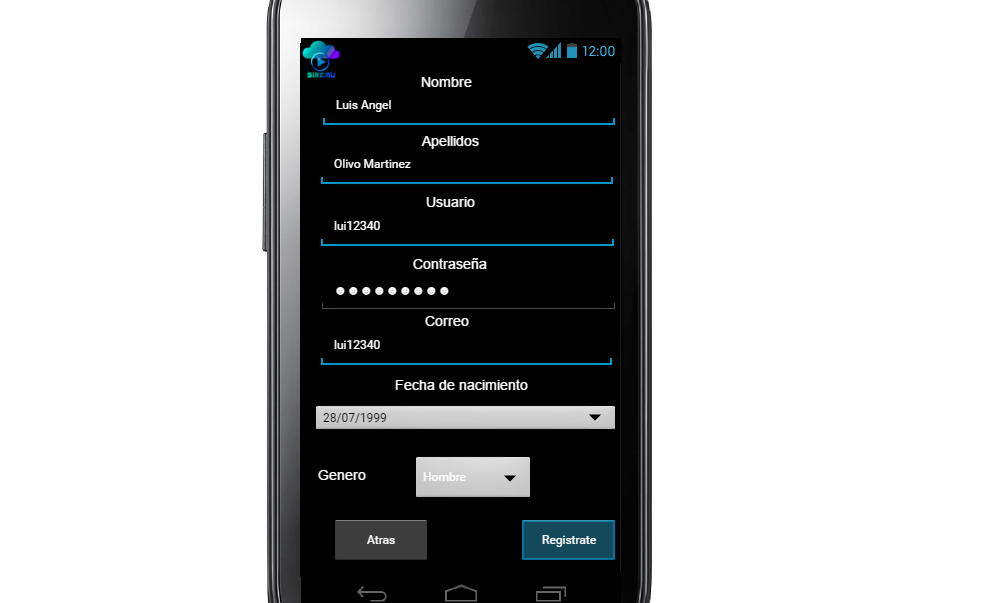
El modelo fue generado directamente de la base de datos y es una representación visual de los datos que almacenará la base de datos del sistema.



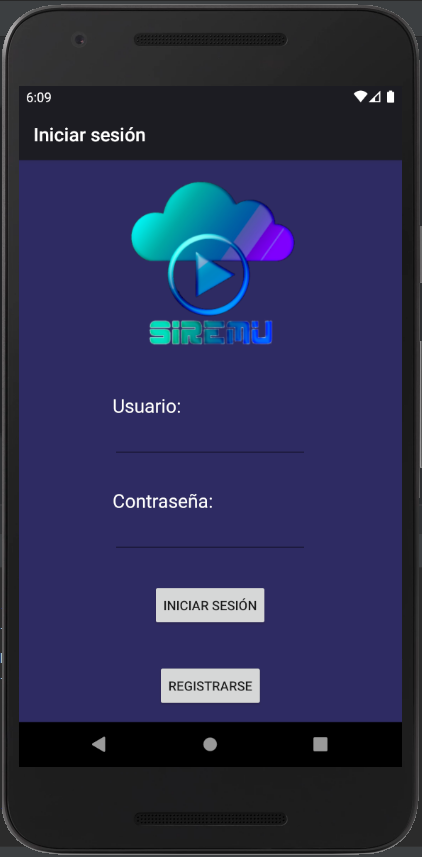
## Prototipo UI Cliente

A continuación, se muestran los prototipos para el desarrollo de los casos de uso, estos prototipos serán la base de diseño para las interfaces del cliente móvil.

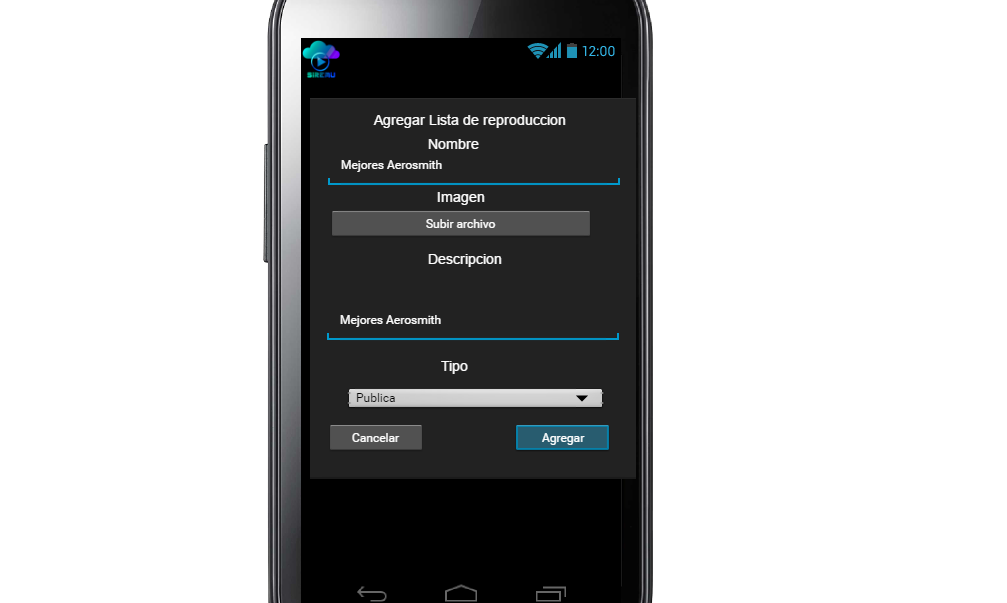
CU-01



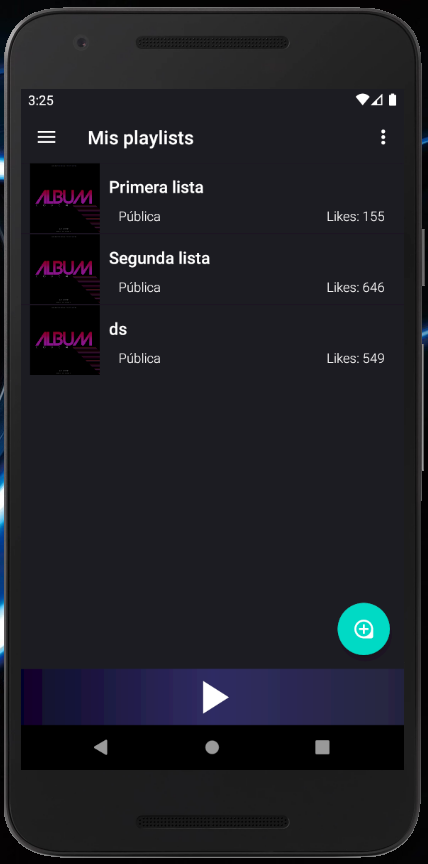
CU-02



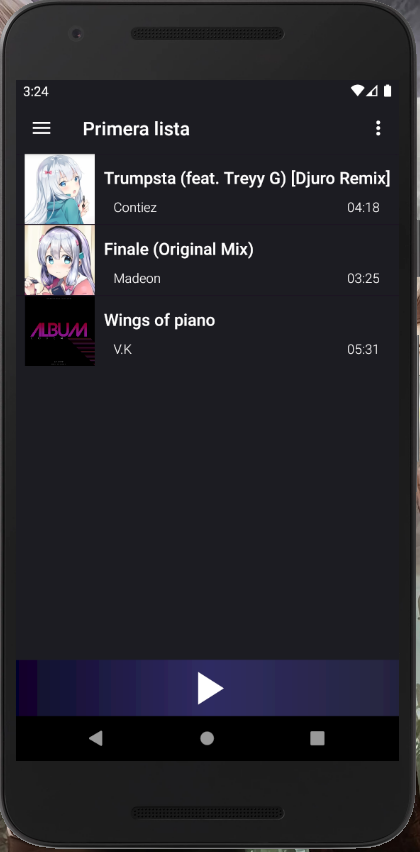
CU-03



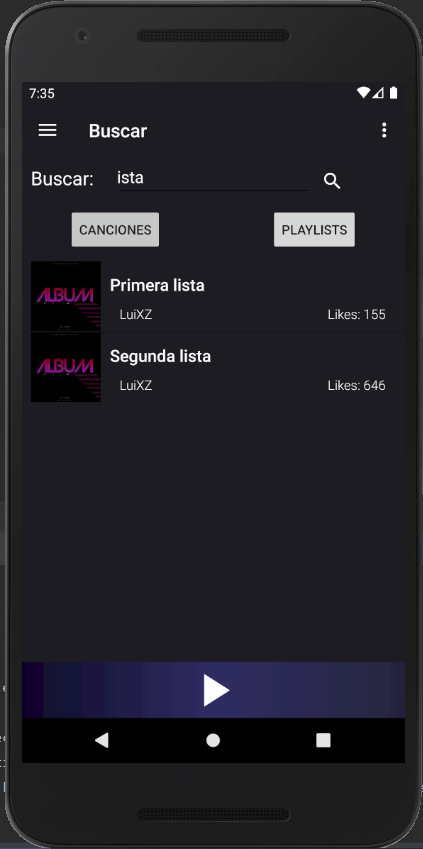
CU-04



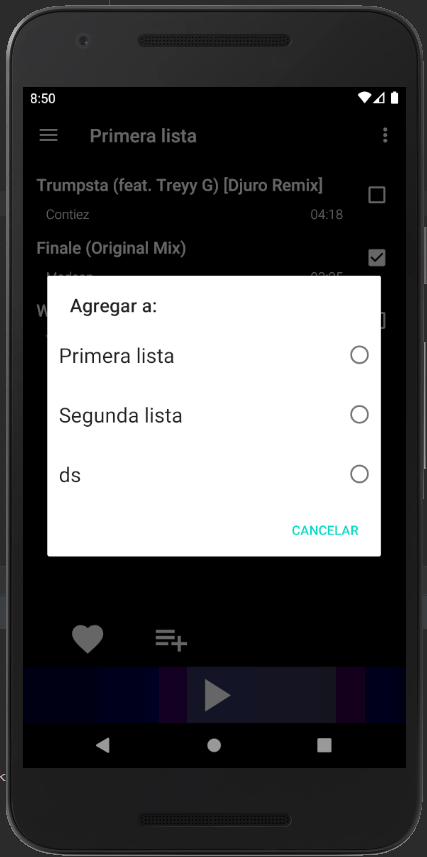
CU-05



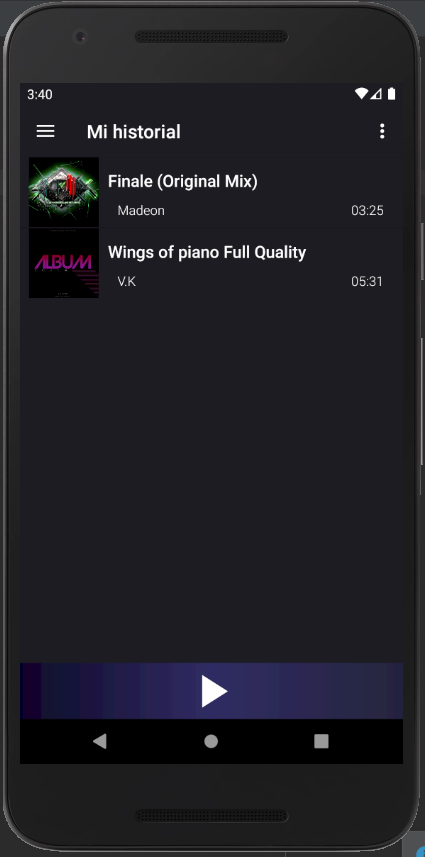
CU-06



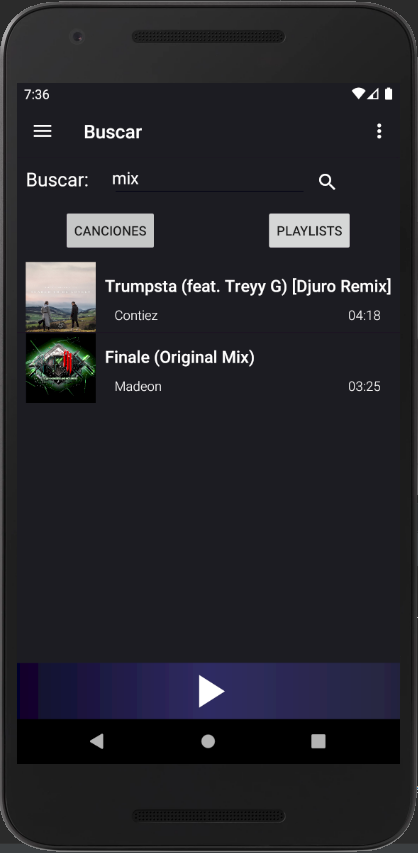
CU-07



CU-08

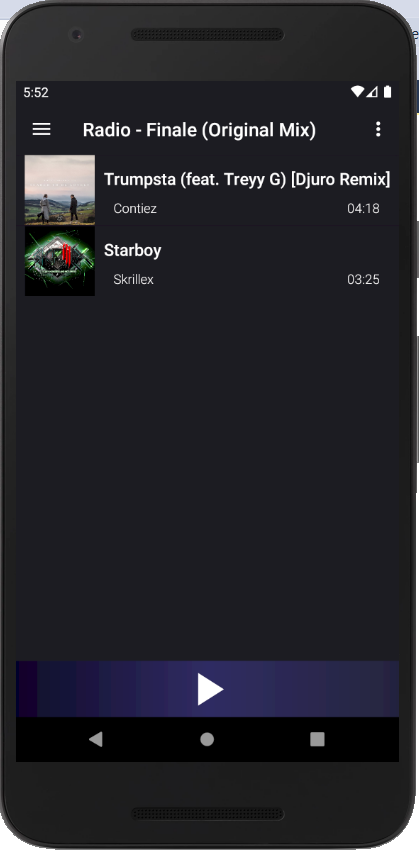


CU-09

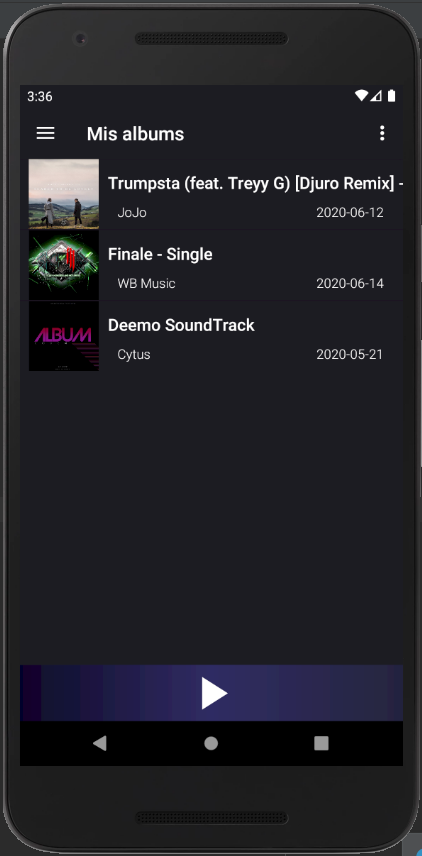


CU-10

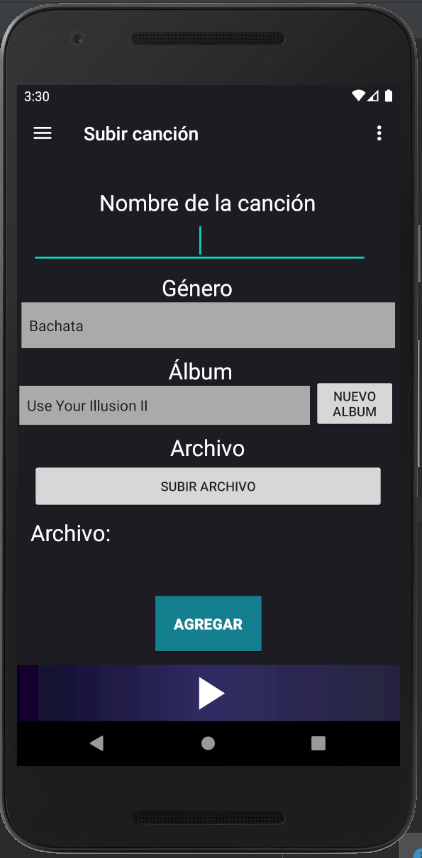
CU-11



CU-12

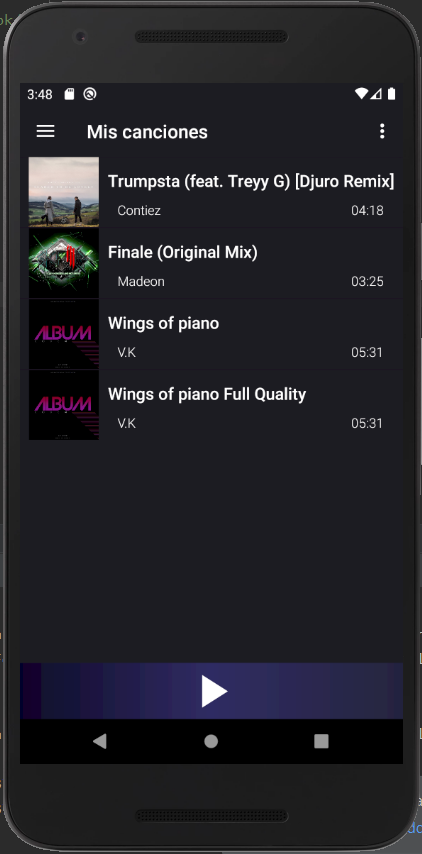
CU-13



CU-14



CU-15



# CONSTRUCCIÓN

## Selección justificada de lenguajes de programación

El lenguaje que se usará para el desarrollo del cliente móvil será Java. La selección de este lenguaje se debe a que es un lenguaje de programacion que está integrado con el IDE de Android Studio, además de ser un lenguaje muy popular para el mencionado IDE, por lo que cuenta con una amplia comunidad en intente, lo que facilita la consulta de dudas y la resolución de problemas, además de esto, Java es un lenguaje de programacion conocido por el equipo de desarrollo y con el que ya se tiene experiencia, por lo que con la selección de este lenguaje, pretendemos ahorrar tiempo de trabajo y de investigación para poder dedicar más tiempo al desarrollo.

## Estándares de codificación

Para la codificación de java en Android Studio, se basará el estándar en el Estilo de código Java del AOSP para colaboradores

Fuente: <https://source.android.com/setup/contribute/code-style>

Sé coherente

Una de las reglas más simples es SER COHERENTE. Si estás editando código, tómate algunos minutos para revisar el código que lo rodea y determinar su estilo. Si ese código usa espacios alrededor de las cláusulas if, tú también deberías hacerlo. Si los comentarios del código tienen pequeños cuadros de estrellas a su alrededor, haz lo mismo con tus comentarios.

El objetivo de tener pautas de estilo es contar con vocabulario de codificación en común para que los lectores puedan concentrarse en lo que estás diciendo y no en cómo lo dices. Aquí incluimos reglas de estilo globales para que conozcas el vocabulario. Sin embargo, el estilo local también es importante. Si el código que agregas en un archivo se ve muy distinto al código existente que lo rodea, puede confundir a los lectores cuando lo lean. Intenta evitar que esto suceda.

Reglas del lenguaje Java

Android cumple con las convenciones de codificación de Java estándar, además de las reglas adicionales que se describen a continuación.

No ignores las excepciones

Puede que te sientas tentado a escribir código que ignore una excepción, por ejemplo:

void setServerPort(String value) {

try {

serverPort = Integer.parseInt(value);

} catch (NumberFormatException e) { }

}

Evita lo anterior. Aunque creas que tu código nunca se encontrará con este error o que no es importante ocuparse de él, ignorar este tipo de excepciones crea defectos en tu código que otra persona podría aprovechar en el futuro. Debes manejar cada excepción en tu código de acuerdo con principios. El manejo específico depende de cada caso.

"Duda cada vez que tengas una cláusula de captura vacía. Aunque, en algunos casos, es lo correcto, deberías reflexionar al respecto. En Java, resulta difícil no pensar que algo está mal". James Gosling

Las alternativas aceptables (en orden de preferencia) son las siguientes:

Muestra la excepción al emisor de tu método.

void setServerPort(String value) throws NumberFormatException {

serverPort = Integer.parseInt(value);

}

Muestra una excepción nueva que sea apropiada para tu nivel de abstracción.

void setServerPort(String value) throws ConfigurationException {

try {

serverPort = Integer.parseInt(value);

} catch (NumberFormatException e) {

throw new ConfigurationException("Port " + value + " is not valid.");

}

}

Maneja el error correctamente y reemplaza un valor apropiado en el bloque catch {}.

/\*\* Set port. If value is not a valid number, 80 is substituted. \*/

void setServerPort(String value) {

try {

serverPort = Integer.parseInt(value);

} catch (NumberFormatException e) {

serverPort = 80; // default port for server

}

}

Captura la excepción y muestra una instancia nueva de RuntimeException. Como esto puede resultar peligroso, hazlo solo cuando estés seguro de que, si el error ocurre, lo correcto es generar una falla.

/\*\* Set port. If value is not a valid number, die. \*/

void setServerPort(String value) {

try {

serverPort = Integer.parseInt(value);

} catch (NumberFormatException e) {

throw new RuntimeException("port " + value " is invalid, ", e);

}

}

Nota: La excepción original se pasa al constructor de RuntimeException. Si tu código debe compilar en Java 1.3, debes omitir la excepción que sea la causa.

Como último recurso, si estás seguro de que ignorar la excepción es lo correcto, puedes hacerlo, aunque debes comentar por qué y dar un buen motivo.

/\*\* If value is not a valid number, original port number is used. \*/

void setServerPort(String value) {

try {

serverPort = Integer.parseInt(value);

} catch (NumberFormatException e) {

// Method is documented to just ignore invalid user input.

// serverPort will just be unchanged.

}

}

No captures excepciones genéricas

También puede resultar tentador ser perezoso en el momento de capturar excepciones y hacer lo siguiente:

try {

someComplicatedIOFunction(); // may throw IOException

someComplicatedParsingFunction(); // may throw ParsingException

someComplicatedSecurityFunction(); // may throw SecurityException

// phew, made it all the way

} catch (Exception e) { // I'll just catch all exceptions

handleError(); // with one generic handler!

}

Evita lo anterior. En casi todos los casos, es inapropiado capturar Exception o Throwable genéricos (es preferible que no sea Throwable, ya que incluye excepciones de Error). Es peligroso porque significa que las excepciones que no esperabas (incluidas las excepciones de tiempo de ejecución, como ClassCastException) se capturan en el manejo de errores a nivel de la app. Como se ocultan las propiedades de manejo de fallas de tu código, si alguien agrega un tipo nuevo de excepción en el código al que estás llamando, el compilador no te indicará que debes manejar el error de otro modo. En la mayoría de los casos, no deberías manejar del mismo modo los distintos tipos de excepciones.

La excepción inusual a esta regla es el código de prueba y el código de nivel superior en el que quieres capturar todos los tipos de errores (para evitar que se muestren en una IU o a fin de mantener un trabajo por lotes en ejecución). En estos casos, puedes capturar una Exception genérica (o un Throwable) y manejar el error de manera adecuada. Sin embargo, analízalo con cuidado antes de hacerlo y explica en los comentarios por qué es seguro en este contexto.

Alternativas para capturar excepciones genéricas:

Captura cada excepción por separado como parte de un bloque de captura múltiple, por ejemplo:

try {

...

} catch (ClassNotFoundException | NoSuchMethodException e) {

...

}

Refactoriza tu código para tener un manejo de errores más detallado con varios bloques de prueba. Divide el IO del análisis y maneja los errores por separado en cada caso.

Vuelve a mostrar la excepción. En muchos casos, no necesitas capturar la excepción en este nivel. Solo deja que el método la muestre.

Recuerda que las excepciones son tus aliadas. Cuando el compilador se queje de que no estás capturando una excepción, no te lo tomes a mal. ¡Sonríe! El compilador acaba de facilitarte la captura de problemas de tiempo de ejecución en tu código.

No uses finalizadores

Los finalizadores son una forma de ejecutar un fragmento de código cuando un objeto es una recolección de elementos no utilizados. Si bien pueden ser útiles para la limpieza (especialmente de recursos externos), no hay garantías de cuándo se llamará a un finalizador (ni de que se lo llamará).

Android no usa finalizadores. En la mayoría de los casos, puedes usar un buen manejo de excepciones. Si el finalizador es imprescindible, define un método close() (o similar) y documenta exactamente cuándo es necesario llamar a ese método (consulta InputStream para ver un ejemplo). En este caso es apropiado, pero no obligatorio, imprimir un mensaje de registro breve del finalizador, siempre que no se espere que sobrecargue los registros.

Importaciones completamente calificadas

Si quieres usar la clase Bar del paquete foo, puedes importarla de dos maneras posibles:

import foo.\*;

Reduce potencialmente la cantidad de instrucciones de importación.

import foo.Bar;

Deja en claro qué clases se usaron y los encargados de mantener el código pueden leerlo con más facilidad.

Usa import foo.Bar; para importar todo el código de Android. Se crea una excepción explícita para las bibliotecas estándar de Java (java.util.\*, java.io.\*, etc.) y el código de prueba de unidades (junit.framework.\*).

Reglas de la biblioteca de Java

Hay convenciones sobre el uso de herramientas y bibliotecas Java de Android. En algunos casos, la convención cambió de maneras significativas y es posible que el código más antiguo use un patrón o una biblioteca obsoletos. Si trabajas con este tipo de código, está permitido seguir usando el estilo existente. Sin embargo, cuando crees componentes nuevos, nunca uses bibliotecas obsoletas.

Reglas de estilo de Java

Usa comentarios estándar de Javadoc

Cada archivo debe tener una declaración de derechos de autor en la parte superior, seguida de instrucciones de paquete e importación (cada bloque separado por una línea en blanco) y, por último, la declaración de clase o interfaz. En los comentarios de Javadoc, describe la función de la clase o la interfaz.

/\*

\* Copyright 2020 The Android Open Source Project

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package com.android.internal.foo;

import android.os.Blah;

import android.view.Yada;

import java.sql.ResultSet;

import java.sql.SQLException;

/\*\*

\* Does X and Y and provides an abstraction for Z.

\*/

public class Foo {

...

}

Cada clase y cada método público no trivial que escribas deberán contener un comentario de Javadoc con al menos una oración que describa la función de la clase o el método. Esta oración deberá comenzar con un verbo descriptivo en tercera persona.

Ejemplos

/\*\* Returns the correctly rounded positive square root of a double value. \*/

static double sqrt(double a) {

...

}

o

/\*\*

\* Constructs a new String by converting the specified array of

\* bytes using the platform's default character encoding.

\*/

public String(byte[] bytes) {

...

}

No es necesario que escribas Javadoc para métodos triviales de obtención y establecimiento como setFoo() si todo lo que tu Javadoc va a decir es "sets Foo". Si el método realiza una acción más compleja (por ejemplo, si aplica una restricción o tiene un efecto secundario importante), entonces debes documentarlo. Si no es evidente qué significa la propiedad·"Foo", se debe documentar.

Todos los métodos que escribas, publiques y demás se beneficiarán con Javadoc. Los métodos públicos forman parte de una API y, por lo tanto, requieren Javadoc. Android no aplica un estilo específico para escribir comentarios de Javadoc, sino que debes seguir las instrucciones de Cómo escribir comentarios doc para la herramienta Javadoc.

Escribe métodos breves

Siempre que sea posible, asegúrate de que los métodos sean breves y estén enfocados. Sabemos que a veces los métodos largos son más apropiados, de manera que no hay un límite estricto en cuanto a la longitud del método. Si un método tiene más de unas 40 líneas, considera dividirlo sin dañar la estructura del programa.

Define campos en lugares estándar

Define campos en la parte superior del archivo o inmediatamente antes de los métodos que los usan.

Limita el alcance de una variable

Mantén el alcance de las variables locales al mínimo. De este modo, aumenta la legibilidad y el mantenimiento de tu código y se reducen las probabilidades de error. Declara cada variable en el bloque más interno que abarca todos los usos de la variable.

Declara las variables locales en el momento en que se usan por primera vez. Casi todas las declaraciones de variables locales deben incluir un inicializador. Si todavía no tienes información suficiente para inicializar una variable como corresponde, pospón la declaración hasta que dispongas de los datos necesarios.

La excepción son las declaraciones de captura y prueba. Si una variable se inicializa con un valor de resultado de un método que muestra una excepción verificada, se debe inicializar dentro de un bloque de prueba. Si el valor se debe usar fuera del bloque de prueba, debes declararlo antes del bloque de prueba, donde todavía no se puede inicializar como corresponde:

// Instantiate class cl, which represents some sort of Set

Set s = null;

try {

s = (Set) cl.newInstance();

} catch(IllegalAccessException e) {

throw new IllegalArgumentException(cl + " not accessible");

} catch(InstantiationException e) {

throw new IllegalArgumentException(cl + " not instantiable");

}

// Exercise the set

s.addAll(Arrays.asList(args));

Sin embargo, puedes evitar incluso este caso si encapsulas un bloque de captura y prueba en un método:

Set createSet(Class cl) {

// Instantiate class cl, which represents some sort of Set

try {

return (Set) cl.newInstance();

} catch(IllegalAccessException e) {

throw new IllegalArgumentException(cl + " not accessible");

} catch(InstantiationException e) {

throw new IllegalArgumentException(cl + " not instantiable");

}

}

...

// Exercise the set

Set s = createSet(cl);

s.addAll(Arrays.asList(args));

Declara las variables de bucle en la instrucción "for", a menos que haya un motivo válido para hacerlo de otro modo:

for (int i = 0; i < n; i++) {

doSomething(i);

}

y

for (Iterator i = c.iterator(); i.hasNext(); ) {

doSomethingElse(i.next());

}

Ordena las declaraciones de importación

El orden de las instrucciones de importación es el siguiente:

Importaciones de Android

Importaciones de terceros (com, junit, net, org)

java y javax

Para que concuerden de manera exacta con la configuración de IDE, las importaciones deben:

Deben seguir un orden alfabético dentro de cada grupo, en el que las letras en mayúscula van antes que las letras en minúscula (por ejemplo, Z va antes que a).

Deben estar separadas por una línea en blanco entre cada grupo significativo (android, com, junit, net, org, java, javax).

Originalmente, no había ningún requisito de estilo con respecto al orden, de manera que los IDE siempre cambiaban de orden, o bien los desarrolladores de IDE tenían que inhabilitar las funciones de administración de importación automática y mantener las importaciones de manera manual. Esto se consideraba poco efectivo. Cuando se preguntó sobre el estilo Java, los estilos preferidos variaban considerablemente y se determinó que Android solo necesitaba "elegir un orden y ser coherente". Por lo tanto, elegimos un estilo, actualizamos la guía de estilo e hicimos que los IDE lo cumplan. Esperamos que, a medida que los usuarios de IDE trabajen en el código, las importaciones en todos los paquetes coincidan con este patrón sin necesidad de trabajo de ingeniería adicional.

Elegimos este estilo de modo que ocurra lo siguiente:

Las importaciones que las personas quieran ver primero aparezcan en la parte superior (android).

Las importaciones que las personas quieran ver menos aparezcan en la parte inferior (java).

Las personas puedan cumplir con el estilo fácilmente.

Los IDE puedan cumplir con el estilo.

Coloca las importaciones estáticas por encima del resto de las importaciones en el mismo orden que las importaciones regulares.

Usa espacios para la sangría

Utilizamos cuatro (4) sangrías de espacio para los bloques y nunca usamos tabulaciones. Si tienes dudas, mantén la coherencia con el código que lo rodea.

Utilizamos ocho (8) sangrías de espacio para las uniones de líneas, incluidas las asignaciones y las llamadas de función.

Recomendado

Instrument i =

someLongExpression(that, wouldNotFit, on, one, line);

No recomendado

Instrument i =

someLongExpression(that, wouldNotFit, on, one, line);

Cumple con convenciones de nomenclatura de campo

Los nombres de campos no públicos y no estáticos comienzan con m.

Los nombres de campos estáticos comienzan con s.

Otros campos comienzan con una letra minúscula.

Los campos públicos estáticos finales (constantes) son ALL\_CAPS\_WITH\_UNDERSCORES.

Por ejemplo:

public class MyClass {

public static final int SOME\_CONSTANT = 42;

public int publicField;

private static MyClass sSingleton;

int mPackagePrivate;

private int mPrivate;

protected int mProtected;

}

Usa el estilo de llave estándar

Coloca las llaves en la misma línea que el código que las antecede, no en su propia línea:

class MyClass {

int func() {

if (something) {

// ...

} else if (somethingElse) {

// ...

} else {

// ...

}

}

}

Es necesario colocar llaves alrededor de las afirmaciones de una condición. Excepción: Si toda la condición (la condición y el cuerpo) cabe en una línea, puedes colocarla en una sola línea, aunque no es obligatorio. Se acepta, por ejemplo, el siguiente caso:

if (condition) {

body();

}

Y el siguiente:

if (condition) body();

No se acepta, sin embargo, el caso que se muestra a continuación:

if (condition)

body(); // bad!

Limita la longitud de la línea

Cada línea de texto en tu código debe tener, como máximo, 100 caracteres. Si bien hubo mucho debate con respecto a esta regla, el máximo permitido sigue siendo de 100 caracteres, con las siguientes excepciones:

Si una línea de comentario contiene un comando de ejemplo o una URL literal de más de 100 caracteres, esa línea puede exceder los 100 caracteres a fin de brindar mayor facilidad para cortar y pegar.

Las líneas de importaciones pueden superar el límite, ya que las personas raramente las ven (esto también simplifica la escritura de herramientas).

Usa anotaciones Java estándar

Las anotaciones deben preceder a otros modificadores del mismo elemento del lenguaje. Las anotaciones de marcadores simples (por ejemplo, @Override) pueden mostrarse en la misma línea con el elemento del lenguaje. Si hay varias anotaciones o anotaciones parametrizadas, muéstralas una por línea en orden alfabético.

Las prácticas estándar de Android para las tres anotaciones predefinidas en Java son las siguientes:

Usa la anotación @Deprecated cada vez que se desaconseje el uso del elemento con anotación. Si usas la anotación @Deprecated, también debes tener una etiqueta @deprecated de Javadoc, la cual debe nombrar una implementación alternativa. Además, recuerda que un método @Deprecated debe seguir funcionando. Si ves código antiguo con una etiqueta @deprecated de Javadoc, agrega la anotación @Deprecated.

Usa la anotación @Override cada vez que un método anule la declaración o implementación de una superclase. Por ejemplo, si usas la etiqueta @inheritdocs de Javadoc y derivas de una clase (no de una interfaz), también debes anotar que el método anula el método de la clase superior.

Usa la anotación @SuppressWarnings solo en circunstancias en las que sea imposible eliminar una advertencia. Si una advertencia pasa esta prueba de "imposible de eliminar", debes usar la anotación @SuppressWarnings para garantizar que todas las advertencias reflejen los problemas reales en el código.

Cuando una anotación @SuppressWarnings sea necesaria, deberá tener un prefijo con un comentario TODO que explique la condición "imposible de eliminar". Por lo general, esto identifica una clase infractora que tiene una interfaz inusual. Por ejemplo:

// TODO: The third-party class com.third.useful.Utility.rotate() needs generics

@SuppressWarnings("generic-cast")

List<String> blix = Utility.rotate(blax);

Cuando se requiere una anotación @SuppressWarnings, refactoriza el código para aislar los elementos de software en los que se aplica la anotación.

Trata los acrónimos como palabras

Trata los acrónimos y las abreviaturas como palabras en las variables, los métodos y las clases de nomenclatura a fin de que los nombres sean más fáciles de leer:

Good Mal

XmlHttpRequest XMLHTTPRequest

getCustomerId getCustomerID

Html de clase HTML de clase

url de string URL de string

id largo ID largo

Como las bases de código JDK y Android son muy poco coherentes en cuanto a los acrónimos, es prácticamente imposible que haya coherencia con el código que las rodea. Por lo tanto, siempre trata los acrónimos como palabras.

Usa los comentarios "TODO"

Usa comentarios TODO para el código que sea temporal; una solución a corto plazo; o código que sea lo suficientemente bueno, pero no perfecto. Estos comentarios deben incluir la string TODO en mayúscula, seguida de dos puntos:

// TODO: Remove this code after the UrlTable2 has been checked in.

y

// TODO: Change this to use a flag instead of a constant.

Si tu comentario TODO tiene el formato "En una fecha futura realizar una acción", asegúrate de incluir una fecha específica ("Corregir antes del noviembre de 2005") o un evento específico ("Quitar este código una vez que todos los mezcladores de producción comprendan el protocolo V7").

Registra con moderación

Si bien el registro es necesario, tiene un impacto negativo en el rendimiento y pierde su utilidad con rapidez si no se mantiene relativamente conciso. Las utilidades de registro proporcionan cinco niveles de registro diferentes:

ERROR: Se usa cuando se produce un error irrecuperable, es decir, algo tendrá consecuencias visibles para el usuario y no podrá recuperarse sin borrar datos, desinstalar apps, limpiar las particiones de datos o sobrescribir todo el dispositivo (o algo peor). Este nivel se registra siempre. Los problemas que ameritan que se creen registros en el nivel ERROR son buenos candidatos para informarlos a un servidor de recopilación de estadísticas.

WARNING: Se usa cuando ocurre un problema serio e inesperado; p. ej., una situación que tendrá consecuencias visibles para el usuario, pero que probablemente se pueda recuperar sin perder datos si realizas una acción explícita, como esperar o reiniciar una app, volver a descargar una versión nueva de una aplicación o reiniciar el dispositivo. Este nivel se registra siempre. Los problemas que ameritan que se creen registros en el nivel WARNING también pueden ser candidatos para informarlos a un servidor de recopilación de estadísticas.

INFORMATIVE: Se usa para indicar que sucedió algo interesante, es decir, cuando se detecta una situación que tal vez tenga un impacto generalizado, aunque no necesariamente sea un error. Dicha condición solo debe ser registrada por un módulo que crea que es el más autorizado en ese dominio (para evitar registros duplicados por parte de componentes no autorizados). Este nivel se registra siempre.

DEBUG: Se usa a fin de indicar lo que sucede en el dispositivo que podría ser relevante para investigar y depurar comportamientos inesperados. Solo debes registrar lo que sea necesario para recopilar información suficiente sobre lo que sucede con tu componente. Si tus registros de depuración dominan el registro, deberías usar el registro detallado.

Este nivel se registra incluso en compilaciones de versiones y debe estar rodeado por un bloque if (LOCAL\_LOG) o if LOCAL\_LOGD), en el que LOCAL\_LOG[D] se define en tu clase o subcomponente, de manera que exista la posibilidad de inhabilitar todo el registro. Por lo tanto, no debe haber una lógica activa en un bloque if (LOCAL\_LOG). También se debe colocar dentro del bloque if (LOCAL\_LOG) toda la compilación de string para el registro. No refactorices la llamada de registro en una llamada de método si se va a ocasionar que la compilación de string se realice fuera del bloque if (LOCAL\_LOG).

Todavía hay partes del código que dicen if (localLOGV). Esto también se considera aceptable, aunque el nombre no sea estándar.

VERBOSE: Se usa para todo lo demás. Este nivel solo se registrará en las compilaciones de depuración y debe estar rodeado por un bloque if (LOCAL\_LOGV) (o su equivalente) para que se pueda compilar de manera predeterminada. Todas las compilaciones de string se quitarán de las compilaciones de versión y deben mostrarse dentro del bloque if (LOCAL\_LOGV).

Notas

Dentro de un módulo determinado, que no sea en el nivel VERBOSE, los errores solo se deben informar una vez de ser posible. Dentro de una sola cadena de llamadas de función de un módulo, solo la función más interna debe mostrar el error, y los emisores del mismo módulo solo deben agregar registros si eso ayuda considerablemente a aislar el problema.

En una cadena de módulos, siempre que no sea en el nivel VERBOSE, cuando un módulo de nivel inferior detecta datos no válidos provenientes de un módulo de nivel superior, el módulo de nivel inferior solo debe registrar la situación en el registro DEBUG, y solo si el registro brinda información que, de otro modo, no estaría disponible para el emisor. Específicamente, no es necesario registrar situaciones en las que se muestra una excepción (la excepción debería incluir toda la información relevante) ni situaciones en las que la única información que se registra está incluida en un código de error. Esto es importante sobre todo en la interacción entre el marco de trabajo y las apps. Además, las condiciones que generen las apps de terceros que estén manejadas correctamente por el marco de trabajo no deberían activar registros en niveles superiores a DEBUG. Las únicas situaciones que deberían activar registros en el nivel INFORMATIVE o niveles superiores son aquellas en las que un módulo o una app detecta un error que ocurre en su propio nivel o que proviene de un nivel inferior.

Cuando es probable que una condición que normalmente justificaría el registro ocurra muchas veces, puede convenir implementar algún mecanismo de límite de frecuencia a fin de evitar el exceso de registros con muchas copias duplicadas de la misma información (o de información muy similar).

Las pérdidas de conexión de red se consideran comunes y esperadas, y no deberían registrarse de forma indiscriminada. Una pérdida de conexión de red que tenga consecuencias dentro de una app debería registrarse en el nivel DEBUG o VERBOSE (según si las consecuencias son lo suficientemente serias e inesperadas para registrarse en una compilación de actualización).

Tener un sistema de archivos completo en un sistema de archivos al que se pueda acceder en apps de terceros o en su nombre no debería registrarse en un nivel superior a "INFORMATIVE".

Los datos no válidos provenientes de cualquier fuente que no sea de confianza (incluidos los archivos en el almacenamiento compartido o los datos que provengan de una conexión de red) se consideran esperados y no deberían activar ningún registro en un nivel superior a DEBUG cuando se detecte que no son válidos (incluso entonces, el registro debería ser lo más limitado posible).

Cuando se usa en objetos String, el operador + crea de manera implícita una instancia de StringBuilder con el tamaño de búfer predeterminado (16 caracteres) y potencialmente otros objetos String temporales. Por lo tanto, crear objetos StringBuilder de manera explícita no es más costoso que recurrir al operador + predeterminado (de hecho, puede ser mucho más eficiente). Ten en cuenta que un código que llama a Log.v() se compila y se ejecuta en compilaciones de versión, incluida la compilación de strings, incluso aunque no se lean los registros.

Todos los registros que otras personas deban poder leer o que estén disponibles en compilaciones de versión deben ser comprensibles y lo suficientemente concisos sin ser crípticos. Esto incluye todos los registros hasta el nivel DEBUG.

Cuando sea posible, mantén el registro en una sola línea. Se aceptan longitudes de línea de hasta 80 o 100 caracteres. Si es posible, evita longitudes superiores a 130 o 160 caracteres (incluida la longitud de la etiqueta).

Si el registro informa sobre operaciones realizadas correctamente, no lo uses en niveles superiores a VERBOSE.

Si usas el registro temporal para diagnosticar un problema que es difícil de reproducir, mantenlo en el nivel DEBUG o VERBOSE y delimítalo con bloques "if" que permitan inhabilitarlo en el momento de la compilación.

Ten cuidado con los filtros de seguridad a través del registro. Evita registrar información privada. En particular, evita registrar información sobre el contenido protegido. Esto es importante sobre todo cuando escribes el código del marco de trabajo, ya que no resulta fácil saber con anticipación qué será información privada o contenido protegido y qué no.

Nunca uses System.out.println() (o printf() para el código nativo). System.out y System.err se redireccionan a /dev/null, por lo que tus instrucciones de impresión no tienen efectos visibles. Sin embargo, la compilación de strings para estas llamadas se ejecuta de todos modos.

La regla de oro del registro es que tus registros no deben expulsar innecesariamente otros registros fuera del búfer, del mismo modo que otros registros no pueden hacerlo con los tuyos.

Reglas de estilo de Javatests

Sigue las convenciones de nomenclatura del método de prueba y usa un guión bajo para separar el objeto de la prueba del caso específico que estás probando. Con este estilo, es más sencillo ver qué casos se están probando. Por ejemplo:

testMethod\_specificCase1 testMethod\_specificCase2

void testIsDistinguishable\_protanopia() {

ColorMatcher colorMatcher = new ColorMatcher(PROTANOPIA)

assertFalse(colorMatcher.isDistinguishable(Color.RED, Color.BLACK))

assertTrue(colorMatcher.isDistinguishable(Color.X, Color.Y))

}

## Selección justificada de tecnologías

Para el desarrollo del cliente Android, se usará el IDE Android Studio

La decisión del uso de Android Studio se basa en que en Android Studio, todo proyecto tiene módulos contentivos de archivos de código fuente y archivos de recursos, lo que facilita la organización de la estructura del proyecto. Además de esto, Android Studio cuenta con la facilidad de usar emuladores para la prueba de una aplicación, editor inteligente de código, plantillas de código, es muy rápido y fácil de usar. Por lo que el uso de Android Studio se consideró como la mejor opción para un desempeño eficiente del cliente móvil.

## Selección justificada de herramientas para el despliegue y manejo de versiones

Manejo de versiones

Para realizar el manejo de versiones del sistema, se escogió el uso de GitHub.

GitHub es el sitio web número uno para almacenar repositorios git. El sistema está diseñado para permitir a los usuarios crear fácilmente sistemas de control de versiones basados en Git

GitHub nos ofrece diferentes ventajas, entre ellas:

Seguimiento de errores. Esta característica pertenece a las funciones de colaboración y permite mejorar la calidad del código al mantener registro de los errores de software detectados en el proyecto.

Búsqueda rápida: El repositorio proporciona una estructuración conveniente de proyectos que permite una búsqueda y clasificación eficiente. Además una indexación adecuada permite a los usuarios encontrar cualquier cadena de código en los repositorios públicos.

Comunidad: GitHub reporta tener más de 20 millones de usuarios a abril del 2017. Esta enorme comunidad de desarrolladores en todo el mundo es una gran fuente de experiencia y habilidades compartidas.

Compartir: El código fuente del proyecto no sólo se puede copiar con Git, también se puede descargar como archivo.

Trabajo conjunto: GitHub brinda funciones eficientes para la administración de equipos.

Compatibilidad: Los proyectos con el código en GitHub se pueden personalizar fácilmente a cualquier servicio host en la nube.

Admite importación con Git, SVN, TFS

Además de todas estas ventajas que nos ofrece GitHub, fue de prioridad su elección, ya que es la herramienta más conocida por el equipo, lo que simplifica su uso.

GitHub nos permite crear un repositorio e ir manejando un control de cambios y de versiones, además de que nos permite hacer un despliegue continuo automático de la aplicación.

Github sobre otras herramientas

Ademas de github, existen otras opciones a github, como gitlab o bitbucket

A continuación se aborda una comparación entre github y gitlab

En realidad ambas herramientas tienen el mismo objetivo y son muy similares algunas diferencias podrían ser

Colaboradores por repositorio

Esta diferencia reside en el número máximo de colaboradores por repositorio. GitHub permite crear repositorios privados de manera ilimitada, pero con un máximo de 3 colaboradores. Por su parte GitLab permite también crear repositorios privados ilimitados y gratuitos, pero en esta ocasión no hay límite de colaboradores.

Tamaño máximo

Otra característica importante y que puede hacer que un usuario elija una u otra opción es el tamaño máximo permitido. En el caso de GitHub ofrece hasta 1 GB por repositorio de manera gratuita

Mientras que en gitlab, se ofrecen 10 gb

Basándonos en estas diferenacias, podemos acordar que no son muchas y son diferencias que en diferenetes situaciones no van a afectar realmente en su uso.

Gitlab parece tener algunas ventajas sobre github, sin embargo nuestra elección de github se basó en que el tipo de proyecto a realizar no requería de mucho mas de lo que nos ofrece github. Además de que ya habíamos trabajado con github varias veces antes. Y no iba a haber tiempo a dedicar a aprender a usar una opción nueva.

Uso de github en el proyecto.

Debido a que no se desarrollará el sistema de cliente por grandes módulos y el equipo de trabajo es muy pequeño, no se harán ramas en el control de versiones, todo se desarrollará sobre la rama máster.

Debido a esto, solo se harán commits una vez a la semana, una vez que se junte todo lo desarrollado durante la semana. Y los avances se dividirán para que se suban los commits por los dos miembros del equipo.

Dentro del repositorio se encontrará la documentación del proyecto. La documentación del proyecto (Este documento) se presentará en el repositorio como un archivo para que cualquier persona pueda tener acceso a ella. La documentación también deberá ser actualizada una vez a la semana para reflejar los cambios de la semana y así mantener la documentación actualizada

En el repositorio también se hará un despliegue automático de la aplicación móvil. Esto se realizará gracias a Github actions que nos permite realizar el despliegue continuo de la aplicación móvil. Este despliegue se hará cada vez que alguien haga commit en la rama máster.

Aparte de esto, se agregará un proyecto para identificar por casos de uso lo que falta por desarrollar del proyecto móvil

Una captura de pantalla de una computadora

Descripción generada automáticamente

Como ultima anotación, debido a los errores que se pueden generar, si ocurre un error en el repositorio, se creará un nuevo repositorio para evitar los problemas generados pero estos repositorios que vallan quedando en desuso, deben ser incluidos en el wiki del proyecto para expresar la constancia del trabajo.

A continuación se presenta el diagrama que explica el flujo de trabajo que se llevó con el manejo de versiones de GitHub:

30 junio

18 junio

junio

15 abril

Tercer repositorio final y corrección de actions

Segundo repositorio Github actions

Primer repositorio Kotlin

Descripción de los repositorios:

Primer repositorio Kotlin:

En este primer repositorio fue donde comenzó a trabajar. Para el comienzo del desarrollo del proyecto, seleccionamos kotlin como nuestro lenguaje de programacion original, en ese reopsitorio se comenzó el proyecto y se desarrollaron varias ventanas para el proyecto. Sin embargo se encontraron varias complicaciones a la hora de trabajar con grpc y kotlin por lo que se decidió cambiar el proyecto a java para el segundo repositorio.

Segundo repositorio Github actions

Este fue el primer repositorio con el proyecto en java. Se trabajó en cambiar el proyecto que teníamos de kotlin a java. Y además se aplicaron githubb actions. Sin embargo existía un problema en el repositorio que complicó el uso de github actions. Y al no poder ser arreglado, se cambió a un tercer repositorio

Tercer repositorio final y corrección de actions.

En este tercer repositorio, se agregó el nuevo proyecto y es donde se trabajó mejor. Se corrigieron las actions de github para tener el despliegue continuo que en este repositorio si funcionó. Además, en este repositorio se trabajó con Docker para el servidor y se agregaron los archivos de aprovisionamiento para los contenedores de Docker.

Despliegue

Se utilizará Docker para el despliegue del servidor, ya que utilizar Docker realmente facilita la creación, implementación y ejecución de aplicaciones mediante el uso de contenedores. Y los contenedores nos permiten empaquetar una aplicación con todas las partes que necesita, como bibliotecas y otras dependencias, y enviarla en un solo paquete. Al hacerlo, se puede estar seguro de que la aplicación se ejecutará en cualquier otra máquina Linux, independientemente de las configuraciones personalizadas que la máquina pueda tener que puedan diferir de la máquina utilizada para escribir y probar el código.

Aparte de Docker se utilizará Docker-Compose para poder manejar los diferentes contenedores Docker que utilizaremos para el despliegue y coordinarlos para que ofrezcan en conjunto todo lo que se necesita.

## Selección justificada de prácticas de construcción

### Prácticas de codificación

Las prácticas de codificación que tomaremos en cuenta para el desarrollo del sistema fueron tomadas de aquí: <http://www.spri.eus/euskadinnova/documentos/1216.aspx>

Esto porque es muy interesante y es una perfecta descripción de las practicas a seguir las cuales están descritas a continuación:

Nunca programar para ti mismo

Siempre que el programador piense que él es el único que va a utilizar el código que genera, aunque realmente sea el único programador del proyecto, introducirá muchos de los errores que hemos comentado. Cada vez quedan menos proyectos de un sólo programador, o en los que participe un solo programador durante todo el ciclo del proyecto.

El código que haya generado un programador puede lastrar todo el proyecto. El resto de los programadores que participen en el proyecto en etapas posteriores tendrán que lidiar con el código generado por los programadores que han participado con anterioridad en el mismo proyecto.

Como programadores debemos convencernos de dicha situación y programar siempre para otros.

Pero ¿cómo se forma un programador para no cometer los errores anteriores?

Leer código

Leer un fragmento de código al día nos permite crecer como programadores.

Además de leer el programador debe revisar el código que lee, en busca de errores o técnicas que le permitan mejorar.

El código que leemos o que generamos, podemos clasificarlo dentro de estas tres

categorías:Captura de pantalla de un celular

Descripción generada automáticamente

Partiendo de estas categorías podemos generar el esquema mental de las preguntas que

como programadores debemos hacernos cuando lee o escribe código.

Una captura de pantalla de una red social

Descripción generada automáticamente

Si nos hacemos estas preguntas veremos dónde nuestro código se sale de los

estándares y las buenas prácticas sin motivo justificado.

Pair Programming

Al emparejar a programadores, entre ambos se aportan experiencia y convergen sus ideas en puntos comunes que permiten crear buen código.

Esto puede significar una gran mejora para ambos programadores.

Organizar el código

Organizar nuestro código es importante para mantener durante el proceso de desarrollo un proyecto saludable y con unos costes de mantenimiento controlados.

Después de la observación de patrones de programación, se ha puesto nombre a cómo organizan (o no) los programadores el código.La Mejor Opción

El mejor código que podemos llegar a generar es aquel organizado en capas y particionado en componentes que interactúan entre sí,

Poner comentarios en el código

Tan importante como tener una buena organización de código, es comentarlo.

Un código documentado es un código que va a poder ser mantenido en el tiempo, por diferentes personas y en diferentes proyectos.

Cuando un programador tiene que realizar mantenimientos o escalar código que no entiende y que no está comentado, en lugar de reutilizarlo, reescribe la funcionalidad, solo con tal de evitarse aprender cómo funciona.

(Esta última está limitada por lo dictado en el estándar de codificación)

KISS: El principio de la simplicidad

Este principio recomienda desarrollar usando código simple y comprensible.

También se conoce como "Keep it Short and Simple", para evitar lo tosco del acrónimo original.

¿Cómo lo aplicamos?

Leer lo de otros - siempre nos ayudará a escribir mejor

Escribir lo mejor que sepamos

Releer nuestro código

Mejorarlo

Dos técnicas nos permiten hacer el código más legible y más

sencillo:

PARTICIONADO

Se basa en el principio de dividir secciones grandes de código en partes más pequeñas, lo que hace cada una de las secciones más comprensible.

SIMPLIFICACIÓN

Se basa en el principio de hacer el código más comprensible eliminando todo aquello que no sea útil o modificándolo para hacerlo más sencillo.

Iterar sobre estos dos puntos las veces que sea necesario y que el tiempo nos lo permita, hará que nuestro código quede limpio, sencillo y legible

### Trabajo en equipo

En el caso del trabajo en equipo, el desarrollo del sistema será continuo entre los miembros del equipo de desarrollo, se manejará un desarrollo iterado por semanas, por lo que cada semana se hará una breve reunión para ver avances y planear los próximos avances en el equipo, ambos miembros del equipo desarrollarán el sistema por separado de manera que se mantendrán comunicados en línea para saber en qué se debe avanzar y en que ya se avanzó.

Plan de trabajo Semana 06/07/2020

Actividades para realizar

-Realizar contenedores Docker

-Actualizar servidores con Docker

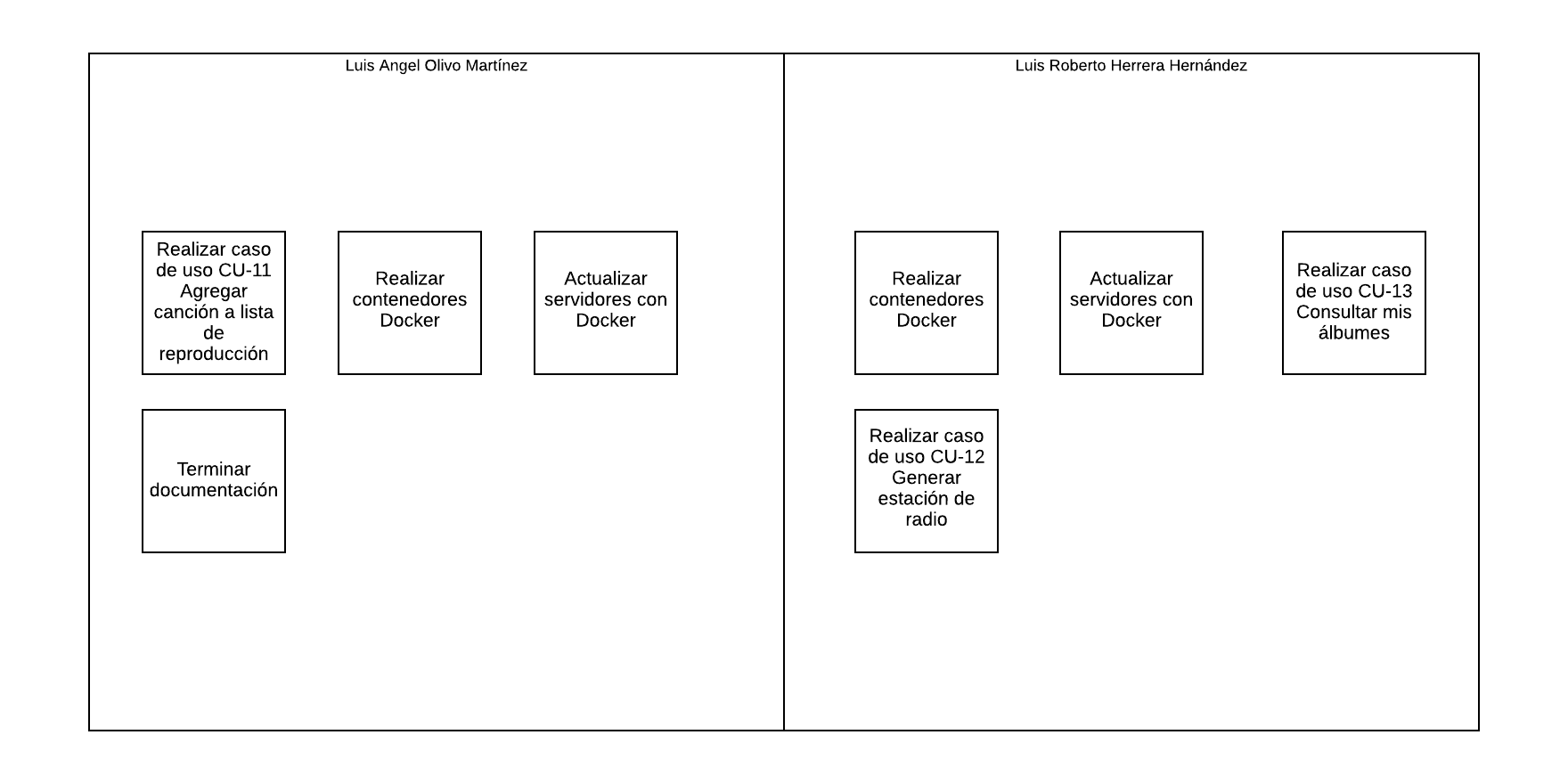
-Realizar caso de uso CU-11 Agregar canción a lista de reproducción

-Realizar caso de uso CU-12 Generar estación de radio

-Realizar caso de uso CU-13 Consultar mis álbumes

-Terminar documentación

Asignación de actividades de la semana



### Aseguramiento de la calidad

Para asegurar la calidad del software, se harán revisiones al software por lo que se ejecutará un analisis estático del código generado y el resultado se mostrará aquí, además de eso se realizaran pruebas descritas en la sección 5 para verificar que el software haga correctamente lo que se planteó por cada caso de uso.

### Herramientas

Para realizar las pruebas, se harán manualmente y para el analisis de código estático, se usará la herramienta que nos proporciona el IDE Android Studio

# PRUEBA

Las pruebas descritas a continuación, se asegurarán de que el software desarrollado cumpla con la calidad requerida, se harán diferentes pruebas. No se busca hacer un conjunto de pruebas demasiado complejo y completo, pero si abarcar lo necesario.

## Plan general de prueba

El desarrollo de las pruebas en el sistema desarrollado, como se comentó anteriormente no busca hacer pruebas totales y completas al 100% del sistema. Solo contar con las suficientes para demostrar el correcto funcionamiento del sistema.

Las estrategias de pruebas que usaremos para las pruebas del sistema serán las siguientes:

Pruebas unitarias

Pruebas de integración

Pruebas funcionales

Estas pruebas, nos permitirán probar el sistema en los aspectos mas importantes: las pruebas unitarias nos permitirán verificar que cada componente desarrollado funciona correctamente por separado, las pruebas de integración permitirán revisar que los componentes funcionan correctamente en conjunto una vez ensamblado el sistema. Por ultimo las pruebas funcionales nos permitirán verificar que las funciones requeridas para el sistema fueron cumplidas. Para esto se basarán las pruebas en los casos de uso.

## Procedimiento de prueba

El procedimiento de las pruebas se debe ejecutar de la siguiente manera:

Pruebas de unidad.

Se desarrollarán pruebas de unidad para los métodos de cada caso de uso. Por lo que solo se desarrollarán dichas pruebas para probar los métodos por separado que implementan cada funcionalidad del sistema. Una vez desarrollada la prueba, se llenará la siguiente tabla para cada método probado.

|  |  |
| --- | --- |
| Responsable de la prueba |  |
| Método probado |  |
| Resultado de la prueba |  |

Adicionalmente se agregará una captura de pantalla de la ejecución de la prueba con su salida.

Pruebas de integración:

Para las pruebas de integración, se desarrollarán pruebas que comprueben que el sistema funciona correctamente por medio de las funcionalidades descritas por los casos de uso y haciendo un conjunto con los métodos que implementan esta funcionalidad los cuales fueron probados con las pruebas de unidad descritas anteriormente.

Una vez ejecutadas las pruebas, se anexará en los resultados una captura de pantalla donde se muestre el resultado de las pruebas.

Pruebas funcionales.

Estas pruebas se desarrollarán manualmente y se basarán en los casos de uso por lo que para cada caso de uso, se desarrollará una prueba para demostrar que la funcionalidad fue implementada correctamente.

|  |  |
| --- | --- |
| Responsable de la prueba |  |
| Caso de uso probado |  |
| Resultado de la prueba |  |

Para cada caso de uso, se llenará la siguiente tabla:

## Resultado

# Conclusiones