

Semana 12

Segundo Sprint



Programación de Aplicaciones Móviles Nativas
10 de diciembre de 2023

Autores:

Ana del Carmen Santana Ojeda (ana.santana152@alu.ulpgc.es)

Alejandro David Arzola Saavedra (alejandro.arzola101@alu.ulpgc.es)

Índice

1	Introducción	2
2	Enlace Github	2
3	Pila del Sprint	2
4	Historias de Usuario	3
5	Capturas de pantalla de las funcionalidades	5
6	Valoracion de las historias de usuario	7
7	Organizacion del equipo	8
8	Pila de Sprint de Github Projects	8



1. Introducción

Para este segundo sprint, nos enfocamos en **perfeccionar** las vistas ya desarrolladas e incorporar nuevas funcionalidades a nuestra aplicación.

En particular, hemos introducido **vistas detalladas** que permiten a los usuarios escuchar las canciones. Para lograr esto, nos conectamos a través de una **API** permitiendo a los usuarios buscar y reproducir cualquier canción disponible.

Además de las **vistas de perfil**, hemos implementado vistas de perfil que brindan a los usuarios la capacidad de visualizar y gestionar su información de perfil. Esto agrega una capa adicional de personalización y mejora la experiencia del usuario.

Otra adición clave es la **vista de elección de género**, donde los usuarios pueden organizar las canciones según su preferencia musical. Esta función proporciona una forma más personalizada de explorar y descubrir nuevas canciones.

Con estas mejoras, buscamos ofrecer a nuestros usuarios una experiencia más completa y personalizada al interactuar con la aplicación.

Adicionalmente se han mejorado el resto de vistas del anterior Sprint.

2. Enlace Github

El enlace al repositorio de GitHub es el siguiente:

[Clicka aquí para ver el segundo Sprint de Tunewave en Github](#)

3. Pila del Sprint

PILA DEL SPRINT	
PUNTOS	HISTORIA DE USUARIO
6	Vista detalle de la cancion
8	Conexion API de canciones
2	Conexion con firestore
2	Vista de genero musical
4	Vista de Perfil

Figura 1: Pila del Sprint



4. Historias de Usuario




WAVE  HISTORIA DE USUARIO 	
ID: 10	Usuario: Desarrollador 
Nombre historia: <i>Vista Detalle de las canciones</i>	
Prioridad en negocio: Muy Alta	Riesgo en desarrollo: Medio
Puntos estimados: 6	Iteración asignada: 2
Programador responsable: <i>Alejandro</i>	
Descripción: 1.1 El usuario podrá seleccionar una canción y poder reproducirla, adelantarla o volver a reproducir	
Validación: <ul style="list-style-type: none"> - El usuario debe poder de reproducir la canción - El usuario si se va para atras debe poder volver a la lista de canciones 	

Figura 2: Historia de Usuario: Vista detalle de las canciones




WAVE  HISTORIA DE USUARIO 	
ID: 11	Usuario: Desarrollador 
Nombre historia: <i>Conexion API de las canciones</i>	
Prioridad en negocio: Muy Alta	Riesgo en desarrollo: Muy Alta
Puntos estimados: 8	Iteración asignada: 2
Programador responsable: <i>Alejandro</i>	
Descripción: 1.1 Para poder obtener informacion de las distintas canciones y albumes se usara una Api.	
Validación: <ul style="list-style-type: none"> - La llamada a la Api debera ofrecer las canciones actuales - En caso de no encontrar una cancion debe informar al usuario 	

Figura 3: Historia de Usuario: Conexion API de las canciones






WAVE  HISTORIA DE USUARIO 	
ID: 13	Usuario: Desarrollador 
Nombre historia: <i>Vista género musical</i>	
Prioridad en negocio: <i>Media</i>	Riesgo en desarrollo: <i>Bajo</i>
Puntos estimados: 2	Iteración asignada: 2
Programador responsable: <i>Ana</i>	
Descripción: 1.1 El usuario podra filtrar musica por género musical .	
Validación: <ul style="list-style-type: none"> - Deberá haber una amplia gama de géneros para buscar canciones(Mas de 5) - Si el usuario selecciona un género debera aparecerle canciones de ese género 	

Figura 4: Historia de Usuario: Vista género musical




WAVE  HISTORIA DE USUARIO 	
ID: 15	Usuario: Desarrollador 
Nombre historia: <i>Vista de Perfil</i>	
Prioridad en negocio: <i>Media</i>	Riesgo en desarrollo: <i>Media</i>
Puntos estimados: 4	Iteración asignada: 2
Programador responsable: <i>Ana</i>	
Descripción: 1.1 El usuario podra ver la información de su perfil .	
Validación: <ul style="list-style-type: none"> - Deberá poderse ver la informacion persona del usuario - Si se permite cambiar algun campo debe atualizarse en la bd. 	

Figura 5: Historia de Usuario: Vista de Perfil



WAVE HISTORIA DE USUARIO	
ID: 12	Usuario: Desarrollador
Nombre historia: <i>Conexion con firestore</i>	
Prioridad en negocio: <i>Media</i>	Riesgo en desarrollo: <i>Media</i>
Puntos estimados: 2	Iteración asignada: 2
Programador responsable: <i>Ana y Alejandro</i>	
Descripción: 1.1 Se debera guardar la informacion personal del usuario en Firestore	
Validación: <ul style="list-style-type: none"> - Debera poderse almacenar en firestore el nombre, correo, imagen y contraseña - En caso de no poder almacenarlo por cualquier problema debe informarlo. 	

Figura 6: Historia de Usuario: Conexion con firestore

5. Capturas de pantalla de las funcionalidades

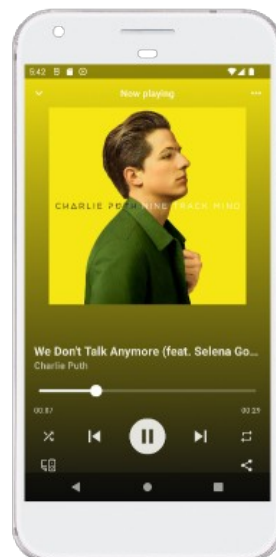


Figura 7: Captura de la vista detalle de la cancion

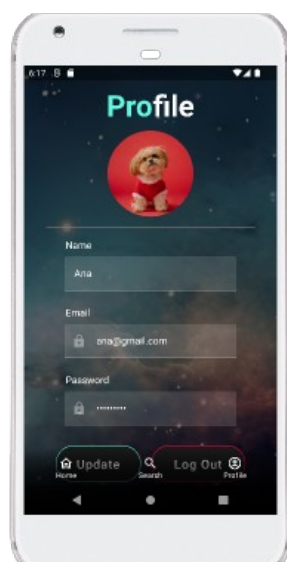


Figura 8: Captura del vista de Perfil

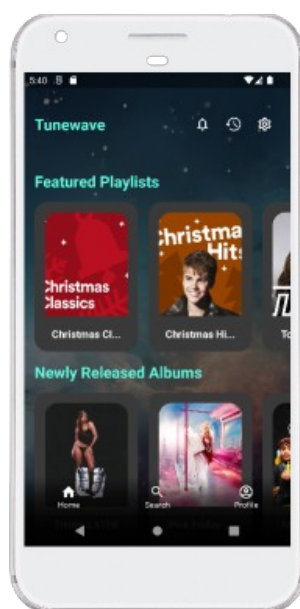


Figura 9: Captura mejorando la interfaz de canciones

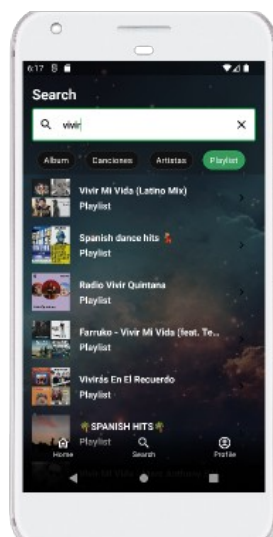


Figura 10: Captura mejorando el home

6. Valoración de las historias de usuario

A continuación, detallamos cómo evaluamos las historias de usuario en nuestro primer sprint:

Vista Detalle de las Canciones

Complejidad: 6 puntos

La implementación de la vista detalle de las canciones implica la incorporación de funcionalidades adicionales, como la reproducción de audio y la gestión de metadatos.

Vista de Género Musical

Complejidad: 2 puntos

La vista de género musical implica organizar y presentar las canciones según categorías específicas. Aunque menos compleja que la vista de detalle, aún requiere una lógica de presentación y filtrado, justificando su nivel de complejidad.

Conexión con Firestore

Complejidad: 2 puntos

La conexión con Firestore implica la integración con una base de datos en la nube, lo cual, aunque esencial, agrega una capa adicional de complejidad en términos de gestión de datos en tiempo real y manejo de posibles errores de conexión.

Conexión API de Canciones

Complejidad: 8 puntos

La conexión con la API de canciones es una tarea más compleja, ya que implica la obtención y gestión de datos externos. Además, la variedad de datos y la necesidad de manejar diversas situaciones requieren una mayor complejidad en la implementación.

Vista de Perfil

Complejidad: 4 punto

La vista de perfil, al centrarse en la presentación estática de información del usuario, se considera relativamente sencilla en comparación con otras tareas más interactivas. Aunque el manejo de la interacción del usuario que puede hacer que requiera mayor complejidad.



7. Organizacion del equipo

En esta etapa, optamos por explorar nuevas herramientas destinadas a la organización de las diversas tareas del sprint. En consecuencia, elegimos hacer uso de una de las funcionalidades de GitHub para estructurar y coordinar nuestras labores: GitHub Projects.

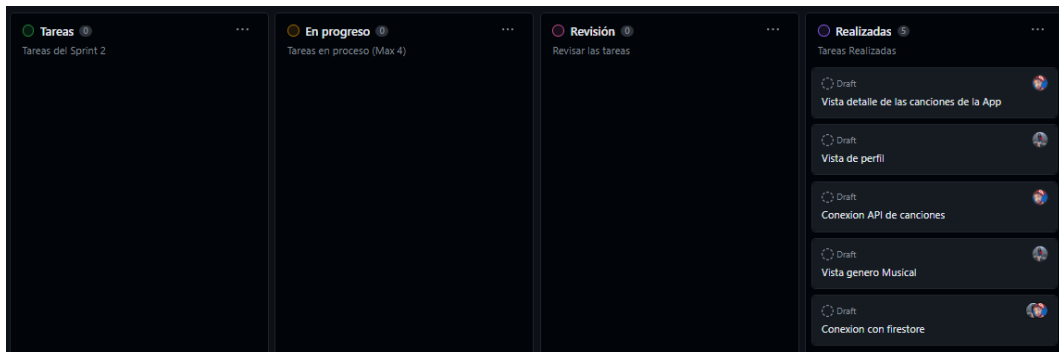


Figura 11: Captura de Github Projects

8. Pila de Sprint de Github Projects

	Title	Assignees	Status
1	Vista detalle de las canciones de la App	AlejandroDavidArzo...	Realizadas
2	Vista Lista de favoritos	AnaSantana016	Realizadas
3	Vista Lista de "Me gusta"	AnaSantana016	Realizadas
4	Vista genero Musical	AlejandroDavidArzo...	Realizadas
5	Vista de perfil	AnaSantana016	Realizadas
6	Conexion con firestore	AlejandroDavidArzo...	Realizadas
7	Conexion API de canciones	AlejandroDavidArzo...	Realizadas

Figura 12: Captura Pila de Sprint de Github Projects