PROYECTO FIN DE GRADO

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

NOMBRE Y APELLIDOS: Kleivan Alexander Torrealba Blanco

TÍTULO DEL PROYECTO: Gestor de eventos musicales

CURSO ESCOLAR 2024-2025

Tutora de proyecto integrado

M.ª Carmen Buenestado Fernández

RESUMEN

El presente proyecto consiste en el desarrollo de una aplicación móvil multiplataforma llamada GestorEventosMusicales, orientada a la gestión eficiente de eventos musicales. La aplicación permite a los managers registrar y administrar eventos, asociando locaciones, artistas e instrumentos de manera dinámica. Está desarrollada con la tecnología .NET MAUI y utiliza C# como lenguaje de programación. La base de datos empleada es PostgreSQL, gestionada a través de pgAdmin, con acceso mediante ADO.NET y la librería Npgsql.

El sistema implementa funcionalidades de inicio de sesión y registro, manteniendo la sesión activa mediante la herramienta Preferences de .NET MAUI. El diseño está orientado a dispositivos Android, y la distribución final se realiza mediante archivo APK o publicación en Google Play Store. Durante el desarrollo y pruebas, se utilizó la herramienta ngrok para facilitar el acceso remoto a servicios locales.

La aplicación presenta una interfaz intuitiva que facilita al usuario la creación, edición y visualización de eventos. Se ha implementado un flujo de navegación claro y adaptado a la experiencia móvil, con persistencia de datos y validación de entradas. El proyecto cumple con los objetivos propuestos, ofreciendo una solución funcional y adaptable para la gestión musical profesional en entornos móviles.

ABSTRACT

This project involves the development of a multiplatform mobile application called GestorEventosMusicales, aimed at the efficient management of musical events. The application allows managers to register and manage events, dynamically associating venues, artists, and instruments. It is developed with .NET MAUI technology and uses C# as the programming language. The database used is PostgreSQL, managed through pgAdmin, with access via ADO.NET and the Npgsql library.

The system implements login and registration functionalities, maintaining the active session through the .NET MAUI Preferences tool. The design is targeted at Android devices, and final distribution is via APK file or publication on the Google Play Store. During development and testing, the ngrok tool was used to facilitate remote access to local services.

The application features an intuitive interface that makes it easy for the user to create, edit, and view events. A clear navigation flow has been implemented, adapted to the mobile experience, with data persistence and input validation. The project meets its objectives, offering a functional and adaptable solution for professional music management in mobile environments.

ÍNDICE

1. INTRODUCCIÓN

Es una aplicación de Gestión y Planificación de Espectáculos Musicales la cual diseñada para facilitar la organización y coordinación de eventos musicales de manera eficiente y accesible. Con esta app, los organizadores de espectáculos podrán gestionar todos los aspectos relacionados con los eventos, como la planificación de fechas, la asignación de artistas y la gestión de recursos.

1.1. PLANTEAMIENTO DEL PROBLEMA.

Este proyecto será un programa para Android enfocada en la gestión y planificación de espectáculos musicales. Cuyo objetivo es el de facilitar la organización de eventos, permitiendo a los usuarios coordinar horarios, locaciones y recursos de manera eficiente. La app busca resolver los desafíos logísticos que enfrentan los organizadores, optimizando la comunicación y la planificación en un solo lugar.

1.2. ANÁLISIS DEL MERCADO

En el mercado de hoy en día, hay diversas herramientas orientadas a la gestión de eventos. Y aunque muchas de ellas no están específicamente diseñadas para el ámbito de los espectáculos musicales. Existen Algunas aplicaciones populares como Eventbrite, Asana, Trello o Google Calendar ofrecen soluciones para organizar tareas, agendar eventos y coordinar equipos. Sin embargo,abordan de forma integral las necesidades específicas del sector musical como lo son: programación de shows, asignación de locaciones, gestión de riders técnicos o la logística del personal artístico, entre otros.

Por otro lado, plataformas como Bandsintown Manager y Gigwell están más orientadas a músicos y agentes para la gestión de giras y presentaciones, sin enfocarse tanto en la coordinación general de espectáculos desde el punto de vista de un productor o planificador de eventos.

1.3. JUSTIFICACIÓN DEL PROYECTO

La industria del espectáculo requiere una planificación precisa y una gestión eficiente de recursos humanos y materiales para garantizar montajes exitosos, pero la falta de herramientas digitales especializadas dificulta la coordinación entre los equipos de producción.

Este programa busca resolver estos desafíos proporcionando una plataforma centralizada para la gestión de espectáculos musicales.

2. DEFINICIÓN DEL PROYECTO

2.1. OBJETIVOS GENERALES Y ESPECÍFICOS.

Los objetivos generales los cuales busco cumplir con esta aplicación, son:

- facilitar la organización de espectáculos
- permitir gestionar los roles de usuario.
- mejorar la asignación de los recursos y de equipos
- Integrar un sistema de comunicación para facilitar la coordinación entre el equipo encargado del montaje y las bandas.
- Ofrecer un historial y seguimiento de eventos
- garantizar la accesibilidad desde dispositivos móviles.

Los objetivos específicos de la aplicación son:

- implementar un sistema de autenticación
- desarrollar una base de datos con información sobre los eventos, usuarios, equipos, recursos necesarios para el montaje e incluso lugares.
- Permitir la creación, edición y eliminación de eventos
- integrar un sistema de notificaciones para mantener informados a los usuarios sobre cambios en el montaje
- diseñar una interfaz intuitiva y responsive
- incorporar un sistema de registro de equipos y materiales
- optimizar la aplicación para un rendimiento eficiente
- garantizar la seguridad de los datos

2.2. CONTEXTO Y PÚBLICO OBJETIVO

El propósito de esta aplicación es facilitar la gestión y coordinación del montaje de espectáculos, proporcionando una plataforma centralizada donde los usuarios puedan planificar eventos y comunicarse de manera eficiente.

El alcance de esta aplicación estará dirigida a organizadores de eventos, técnicos de montaje y demás personal involucrado en la producción de espectáculos musicales en las que permitirá el acceso mediante un sistema de autenticación.

3. ANÁLISIS DE LA APLICACIÓN

3.1. MEDIOS HARDWARE Y SOFTWARE A UTILIZAR

Los medios hardware son:

• Un dispositivo Android o un emulador de Android para comprobar el funcionamiento del programa.

• Un ordenador con S.O Windows para el desarrollo del programa

• Un servidor encargado de almacenar los datos-

Los medios software planteados en un inicio son:

• visual estudio code: Será el editor de código del programa.

• .NET SDK: Será la base para desarrollar en C#

• Android SDK: para compilar y probar la aplicación en Android

• .NET MAUI Workload: será necesario para el desarrollo de aplicaciones móviles en C#.

• SpostgreSQL: es la base de datos.

3.2. METODOLOGÍA DE DESARROLLO

Escogeré la metodología Scrum porque ayuda a organizar tareas, priorizar y avanzar de forma estructurada en el desarrollo.

3.3. PRESUPUESTO Y ANÁLISIS DE COSTOS

En un principio usaré solamente herramientas gratuitas, al igual que me encargaré yo de todo el diseño. Lo único que tengo que costear sería la tarifa única de Google Play Developer (25$)

4. DISEÑO DE LA APLICACIÓN

4.1. ESPECIFICACIÓN DE REQUISITOS

La especificación de requisitos es una parte fundamental que ayuda a definir qué funcionalidades y características debe tener el software que se va a desarrollar. En este punto deja una introducción sobre lo que contiene este punto. Debes incluir los siguientes puntos:

4.1.1. REQUISITOS FUNCIONALES

Los requisitos funcionales describen las funciones y comportamientos específicos que el sistema debe ofrecer para cumplir con las necesidades de los usuarios y los objetivos del proyecto. Estos requisitos definen qué tareas puede realizar el software y cómo debe responder a las acciones del usuario.

Gestión de Eventos

- Crear, editar y eliminar espectáculos musicales.
- Asignar fechas, horarios y locaciones a cada evento.
- Gestionar listas de artistas y bandas participantes.

Gestión de Recursos

- Registrar y administrar equipos, instrumentos y personal.
- Asignar recursos a cada evento según disponibilidad.

Usuarios y Roles

- Registro e inicio de sesión para organizadores y artistas.
- Diferentes roles de usuario con permisos específicos (administrador, staff, artista).

Comunicación y Coordinación

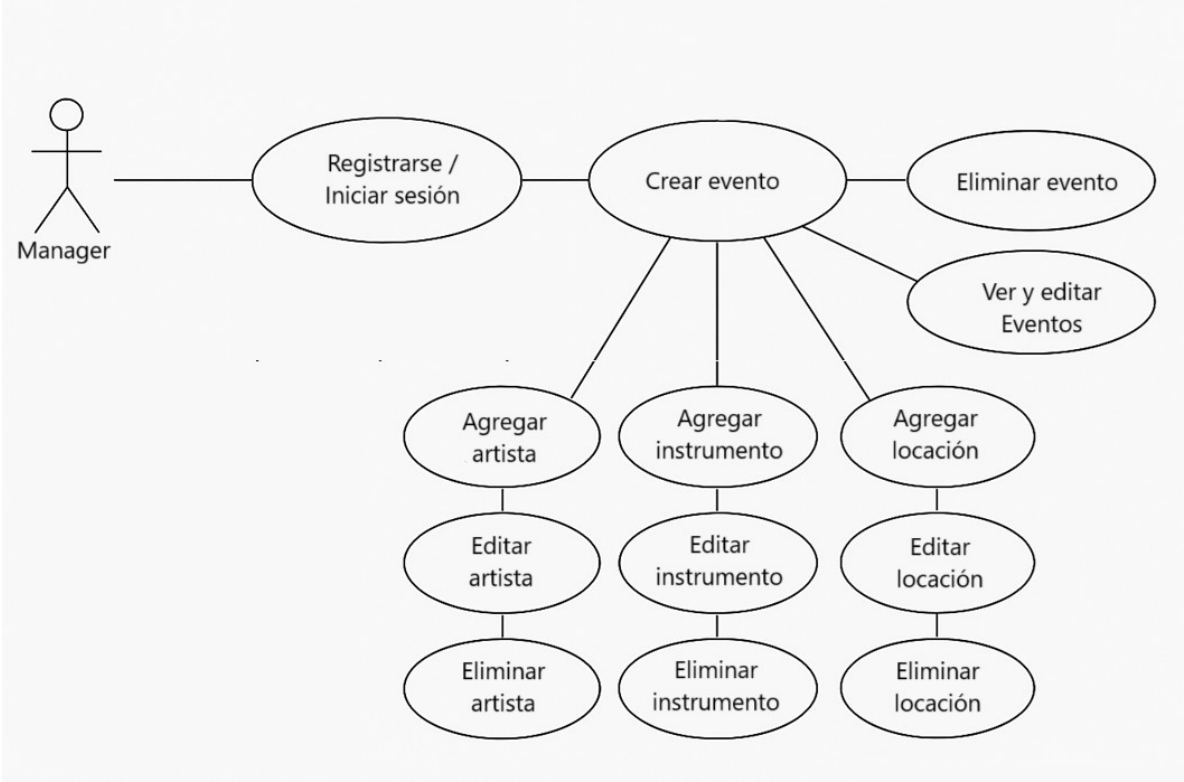
- Enviar notificaciones sobre cambios o recordatorios de eventos.
- Chat interno o foro de discusión entre organizadores y artistas.

Reportes y Seguimiento

- Visualización de eventos en un calendario interactivo.
- Generación de reportes sobre uso de recursos y asistencia esperada.

4.1.2. CASOS DE USO

- **Manager** (usuario autenticado del sistema, solo habrán managers)



Explicación de Casos de Uso

Autenticación

1. **Registrarse**: El manager crea una cuenta proporcionando sus datos.
2. **Iniciar sesión**: El manager se autentica para acceder a las funcionalidades.

Gestión de Eventos

3. **Crear evento**: Puede ingresar datos como nombre, fecha, locación, artistas, instrumentos y managers involucrados.
4. **Editar evento**: Modifica información del evento.
5. **Eliminar evento**: Elimina un evento creado.
6. **Ver eventos** : Muestra los eventos en los que está involucrado el manager actual.
7. **Ver detalles del evento**: Accede a toda la información del evento seleccionado.

Gestión de Artistas

8. **Agregar artista**: Permite añadir artistas al sistema.
9. **Editar artista**: Modifica la información del artista.
10. **Eliminar artista**: Elimina artistas del sistema.

Gestión de Instrumentos

11. **Agregar instrumento**
12. **Editar instrumento**
13. **Eliminar instrumento**

Gestión de Locaciones

14. **Crear locación**
15. **Editar locación**
16. **Eliminar locación**

4.1.3. REQUISITOS NO FUNCIONALES

Los requisitos no funcionales establecen las condiciones de calidad y restricciones que debe cumplir el sistema, tales como rendimiento, seguridad, usabilidad y escalabilidad. Estos requisitos garantizan que el software funcione correctamente bajo diferentes condiciones y proporcione una experiencia óptima al usuario.

Rendimiento

- La app debe cargar y mostrar eventos
- El procesamiento de datos no debe bloquear la interfaz de usuario.

Usabilidad

- Implementar una Interfaz intuitiva y fácil de usar para organizadores y artistas.
- Diseño responsive para adaptarse a diferentes tamaños de pantalla.

Seguridad

- Autenticación de usuarios.
- Cifrado de datos sensibles en la base de datos.

Escalabilidad

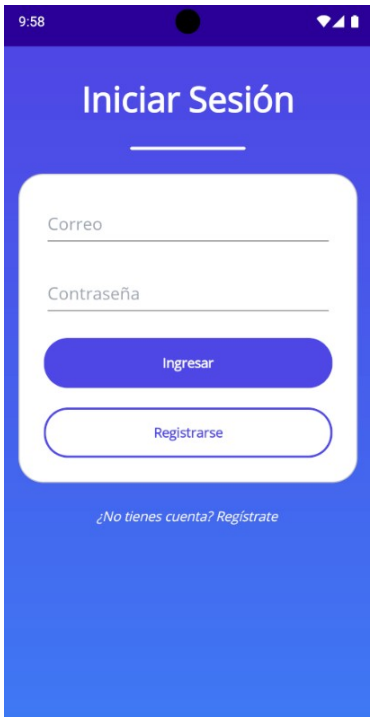
- La app debe permitir agregar más usuarios y eventos sin perder rendimiento.
- Posibilidad de migrar la base de datos a un servidor más potente si es necesario.

Mantenimiento

- Código modular para facilitar futuras actualizaciones.
- Registro de errores para depuración y mejoras continuas.

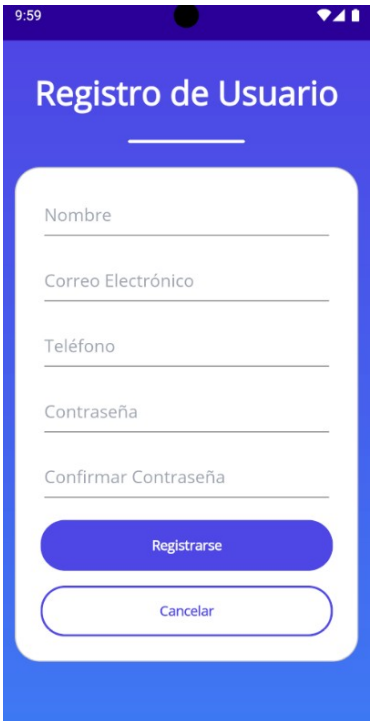
4.1.4. INTERFACES DE USUARIO

Pantalla de inicio de sesió:

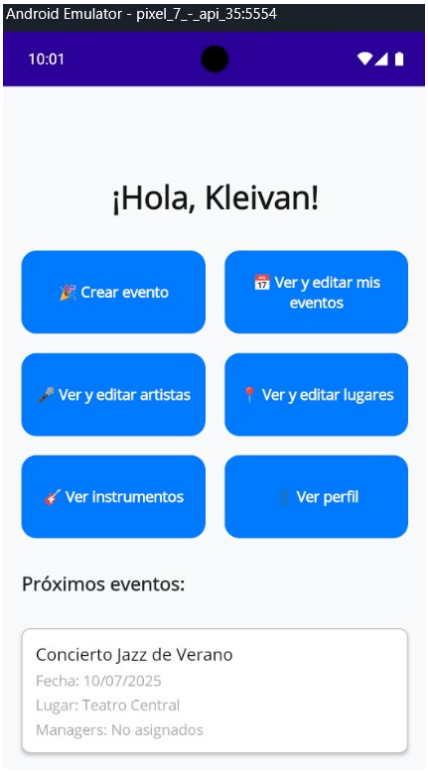


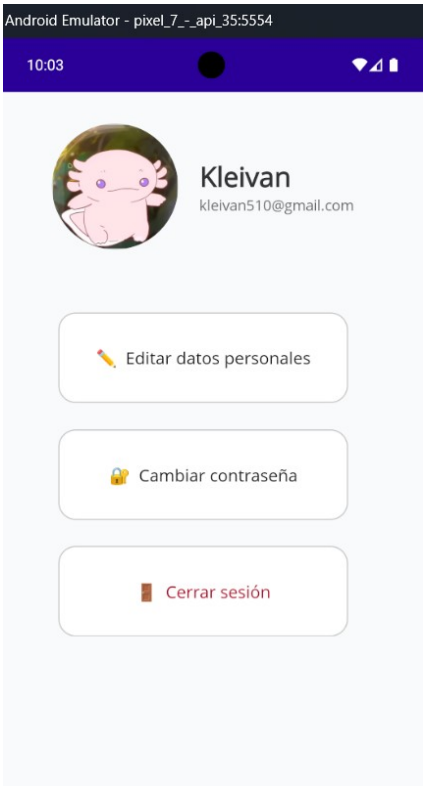
Pantalla de inicio.

Pantalla de registro de usuario:

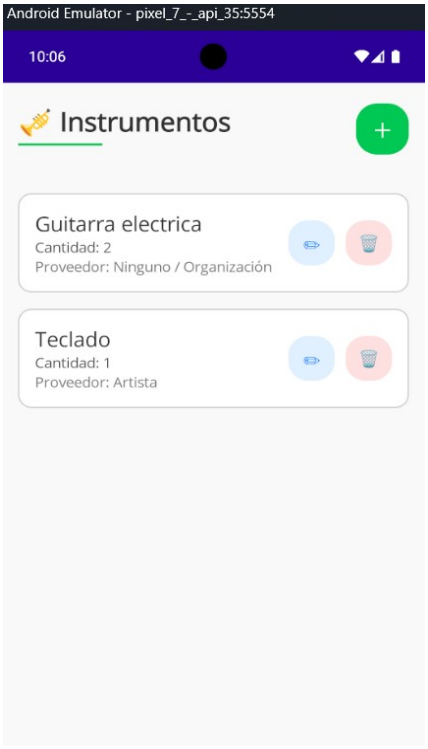


Pantalla de perfil

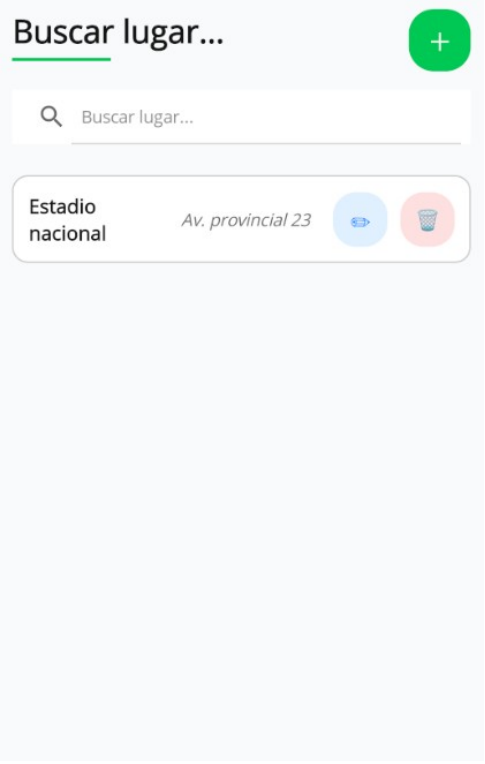




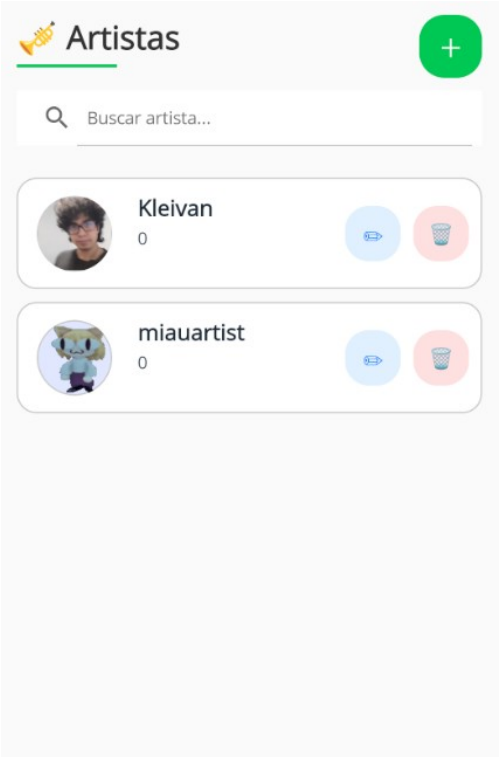
Pantalla para visualizar los instrumentos



Pantalla para ver los lugares

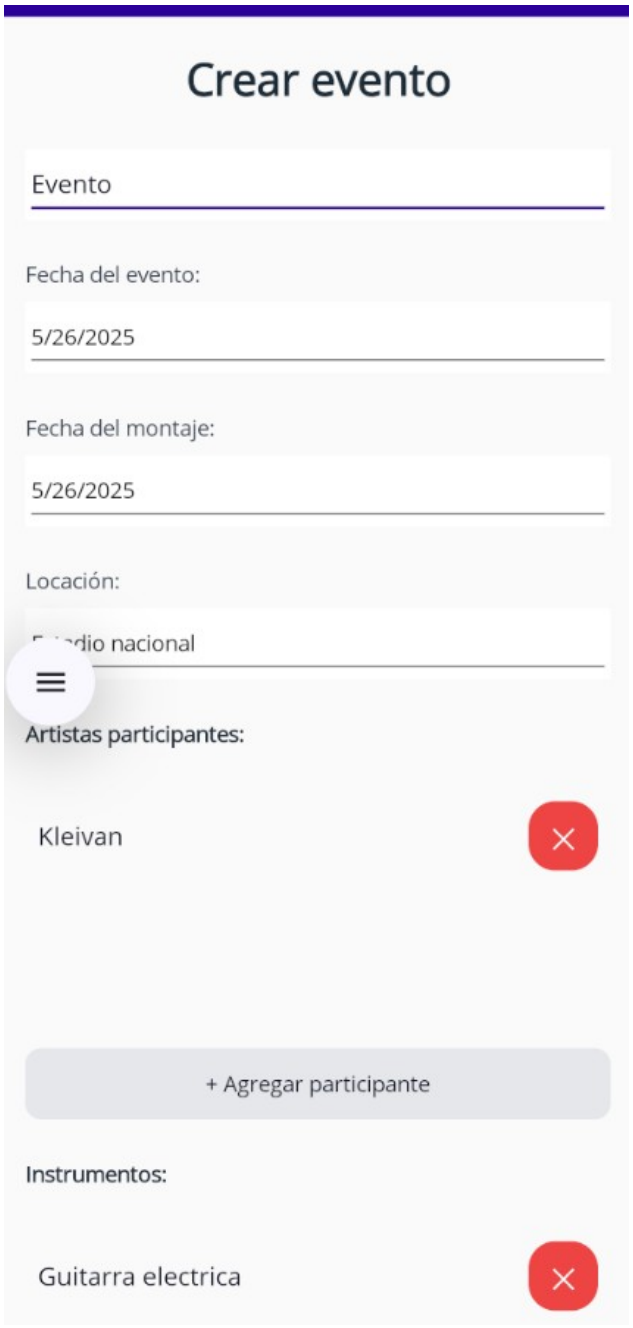


Pantalla para visualizar los artistas



Pantalla para visualizar los eventos

Pantalla para crear los eventos:



4.2. RESTRICCIONES DE DISEÑO

El diseño y desarrollo del sistema **Gestor de Eventos Musicales** se encuentra sujeto a las siguientes restricciones:

Metodología de Desarrollo

- Se utiliza una metodología ágil, con enfoque iterativo e incremental para permitir la mejora continua del sistema durante su desarrollo.

Lenguaje y Entorno de Desarrollo

- El proyecto está desarrollado en **C#** utilizando la plataforma **.NET MAUI** para asegurar compatibilidad multiplataforma (Android, Windows).
- Se emplea el IDE **Visual Studio 2022** , debido a su compatibilidad con .NET MAUI y su conjunto de herramientas integradas.
- La base de datos está implementada en PostgreSQL, con acceso mediante la biblioteca Npgsql.

Herramientas y Librerías

- Se utiliza **PostgreSQL** como base de datos principal.
- Se usa **Preferences** para almacenamiento local de sesión (ManagerId actual).
- Se permite el uso de **MVVM Toolkit**, librerías de conversión (IValueConverter) y bibliotecas estándar .NET para networking, UI y acceso a datos.

Restricciones de Componentes y Librerías

- No se permite el uso de librerías comerciales de pago.
- Se prioriza el uso de herramientas open-source y componentes internos desarrollados específicamente para el proyecto.
- El código debe mantenerse desacoplado y modular, evitando dependencias innecesarias.

Restricciones de Hardware y Compatibilidad

- El sistema debe funcionar correctamente en dispositivos Android con Android 8.0 (API 26) o superior.

- Se prioriza la compatibilidad con resoluciones comunes de smartphone, sin necesidad de ajustes para tablets o pantallas grandes.
- El rendimiento debe mantenerse aceptable en dispositivos con 2 GB de RAM o más.

5. DISEÑO DE LA BASE DE DATOS

5.1. DESCRIPCIÓN GENERAL

La base de datos elegida ha sido **PostgreSQL**, una bases de datos relacional

Características y Ventajas

- **Fiabilidad y rendimiento**: Maneja grandes volúmenes de datos sin comprometer la velocidad.
- **Soporte ACID**: Garantiza integridad y seguridad en las transacciones.
- **Escalabilidad y concurrencia**: Permite múltiples accesos simultáneos sin bloqueos.
- **Compatibilidad con .NET MAUI**
- **Código abierto y sin costos de licencia**.

Forma de Uso en el Proyecto

La base de datos estará alojada en un servidor, conectándose con la aplicación a través de una API en C# para manejar eventos, artistas y locaciones.

Motivo de Elección

Se eligió PostgreSQL por su estructura relacional, ideal para gestionar datos interconectados de espectáculos. A diferencia de Firebase, que no maneja bien relaciones complejas, y MySQL, PostgreSQL ofrece mejor concurrencia y optimización para consultas avanzadas.

5.2. MODELO DE DATOS

Se utiliza un modelo relacional usando PostgreSQL. Este sistema de gestión de bases de datos relacional organiza la información en tablas las cuales se componen de columnas y filas. Las entidades que forman parte de esta base de datos son: **usuarios**, **eventos**, **artistas**, **locaciones** y **recursos**.

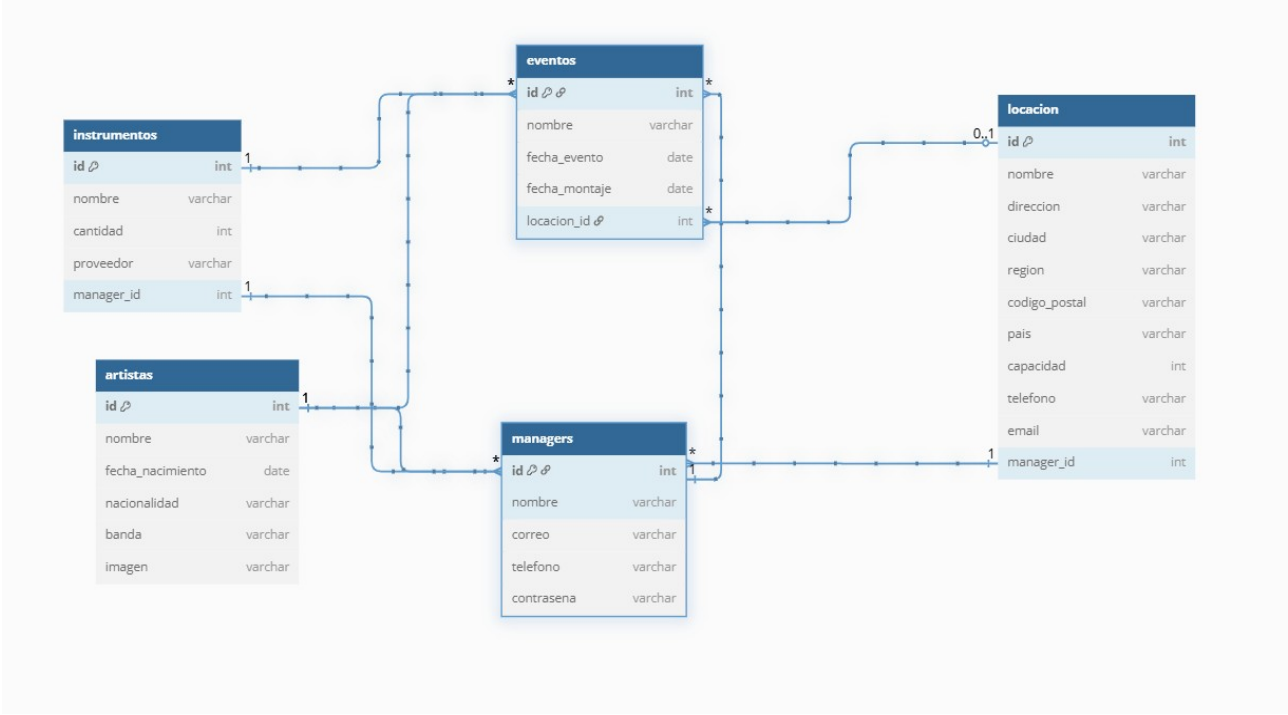
Las tablas clave incluyen:

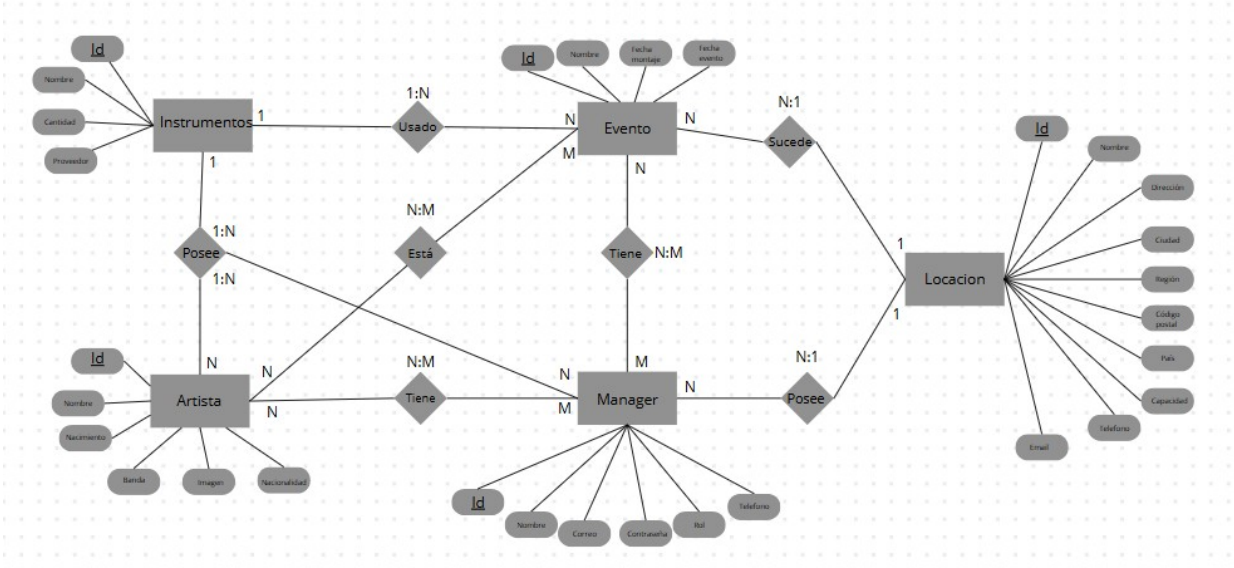
1. **Usuarios**: Almacena la información de los usuarios como organizadores, artistas y staff.

2. **Eventos**: Representa los espectáculos musicales, con atributos como nombre, fecha y locación.
3. **Artistas**: Contiene detalles de los artistas que participan en los eventos.
4. **Locaciones**: Información sobre las ubicaciones de los eventos.
5. **Instrumentos**: Almacena los instrumentos de los eventos.

Las tablas están **relacionadas** entre sí a través de **claves primarias**, lo que garantiza la integridad y permite la realización de consultas complejas sin perder rendimiento.

Diagrama Entidad Relación sin tablas intermedias


eventos
id int
nombre varchar
fecha_evento date
fecha_montaje date
locacion_id int
instrumentos
id int
nombre varchar
cantidad int
proveedor varchar
manager_id int
locacion
id int
nombre varchar
direccion varchar
ciudad varchar
region varchar
codigo_postal varchar
pais varchar
capacidad int
telefono varchar
email varchar
manager_id int
artistas
id int
nombre varchar
fecha_nacimiento date
nacionalidad varchar
banda varchar
imagen varchar
managers
id int
nombre varchar
correo varchar
telefono varchar
contrasena varchar


Instrumentos
Usado
Evento
Sucede
Locacion
Posee
Está
Tiene
Artista
Tiene
Manager
Posee
Id
Nombre
Cantidad
Proveedor
Fecha montaje
Fecha evento
Dirección
Ciudad
Región
Código postal
País
Capacidad
Teléfono
Email
Nacimiento
Banda
Imagen
Nacionalidad
Correo
Contraseña
Rol
1:N
N:1
N:M
1
N
M

5.2.1. MODELO ENTIDAD-RELACIÓN (Si tu base de datos es del tipo entidad-relación)

1. Manager

- **Atributos**:
  - id (**PK**)
  - nombre, correo, telefono, contrasena, rol
- **Relaciones**:
  - Un **manager** puede administrar **muchos artistas** (relación muchos a muchos con Artista a través de una tabla intermedia).
  - Un **manager** puede tener muchas **locaciones** (**1:N**).
  - Un **manager** puede poseer muchos **instrumentos** (**1:N**).
  - Un **manager** puede participar en muchos **eventos** (**N:M**).

2. Artista

- **Atributos**:
  - id (**PK**)
  - nombre, fecha_nacimiento, nacionalidad, banda, imagen
- **Relaciones**:
  - Un **artista** puede estar gestionado por varios **managers** (relación N:M con Manager)
  - Un **artista** puede participar en varios **eventos** (relación N:M con Evento)

3. Evento

- **Atributos**:
  - id (**PK**)
  - nombre, fecha_evento, fecha_montaje, locacion_id (**FK** → Locacion.id)
- **Relaciones**:
  - Cada **evento** está asociado a una única **locación** (**1:N** desde Locacion).
  - Un **evento** puede tener muchos **managers** (**N:M**)
  - Un **evento** puede tener muchos **instrumentos** (**N:M**)
  - Un **evento** puede tener muchos **artistas** (**N:M**)

4. Locación

- **Atributos**:
    - id (**PK**)
    - nombre, direccion, ciudad, region, codigo_postal, pais, capacidad, telefono, email, manager_id (**FK** → Manager.id)
- **Relaciones**:
    - Una **locación** puede tener varios **eventos** (**1:N**)
    - Cada **locación** pertenece a un único **manager** (**1:N** desde Manager)

5. Instrumento

- **Atributos**:
    - id (**PK**)
    - nombre, cantidad, proveedor, manager_id (**FK** → Manager.id)
- **Relaciones**:
    - Un **instrumento** pertenece a un **manager** (**1:N**)
    - Un **instrumento** puede estar presente en varios **eventos** (**N:M**)

5.3. PASO A ESTRUCTURAS DE ALMACENAMIENTO

5.3.1. Tablas principales:

- **Managers**(**id**, nombre, correo, telefono, contrasena, rol)
  Tabla que almacena los managers del sistema, incluyendo su rol para definir permisos y accesos.
  No tiene claves foráneas.
- **Artistas**(**id**, nombre, fecha_nacimiento, nacionalidad, banda, imagen)
  Almacena los datos de los artistas.
  No tiene claves foráneas.
- **Instrumentos**(**id**, nombre, cantidad, proveedor, *manager_id*)
  Instrumentos disponibles, cada uno asociado a un único manager.
  manager_id: clave foránea que referencia a **Managers**(**id**).
- **Locacion**(**id**, nombre, direccion, ciudad, region, codigo_postal, pais, capacidad, telefono, email, *manager_id*)
  Almacena las locaciones para eventos, cada una gestionada por un manager.
  manager_id: clave foránea que referencia a **Managers**(**id**).

- **Eventos**(**id**, nombre, fecha_evento, fecha_montaje, *locacion_id*)
  Tabla principal de eventos.
  locacion_id: clave foránea que referencia a **Locacion**(**id**).

5.3.2. Tablas puente (para relaciones muchos a muchos):

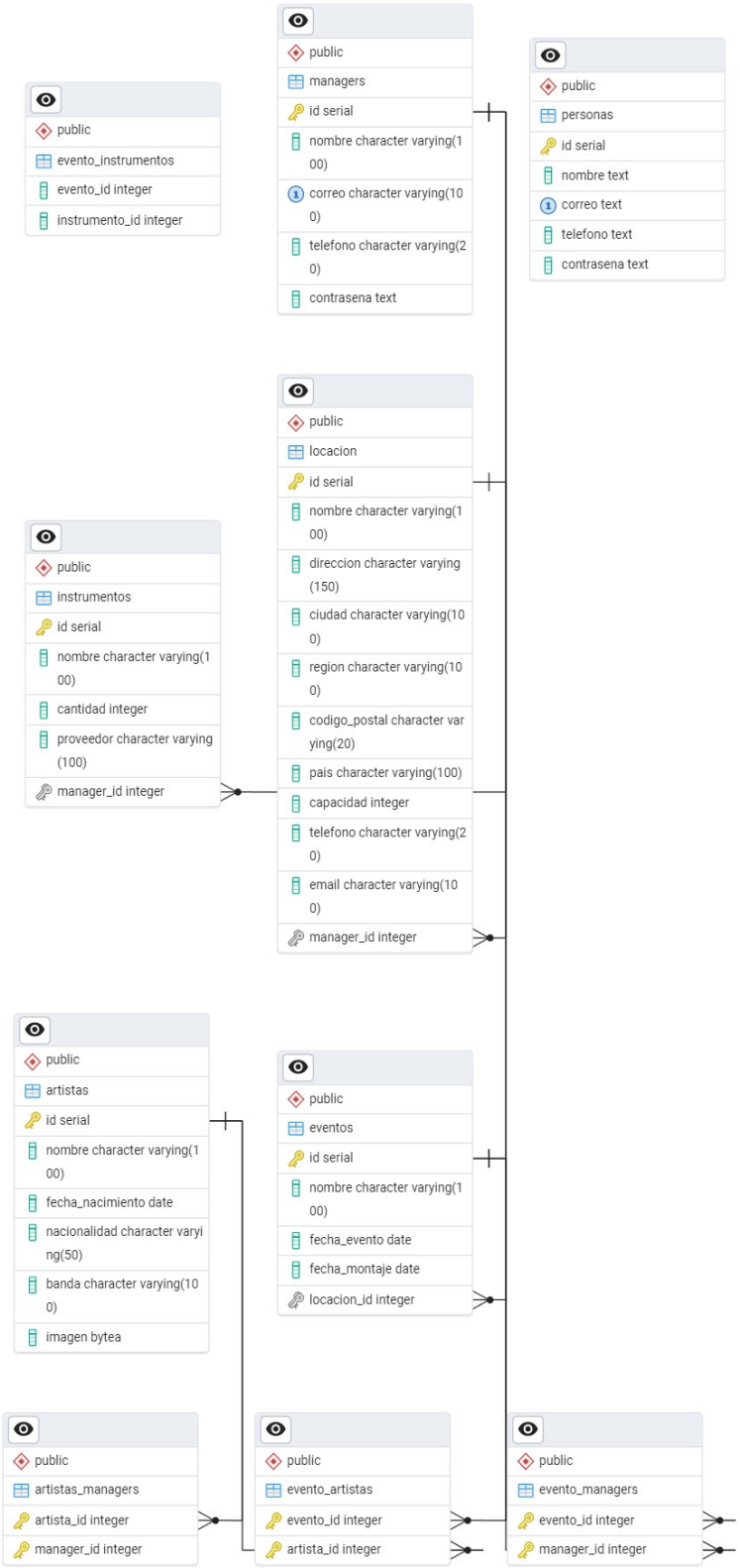
- **artistas_managers**(**artista_id**, **manager_id**)
  Relaciona artistas y managers.
  artista_id: FK a **Artistas**(**id**).
  manager_id: FK a **Managers**(**id**).

- **evento_artistas**(**evento_id**, **artista_id**)
  Relaciona eventos y artistas.
  evento_id: FK a **Eventos**(**id**).
  artista_id: FK a **Artistas**(**id**).

- **evento_instrumentos**(**evento_id**, **instrumento_id**)
  Relaciona eventos e instrumentos.
  evento_id: FK a **Eventos**(**id**).
  instrumento_id: FK a **Instrumentos**(**id**).

- **evento_managers**(**evento_id**, **manager_id**)
  Relaciona eventos y managers.
  evento_id: FK a **Eventos**(**id**).
  manager_id: FK a **Managers**(**id**).

5.4. DIAGRAMA DE TABLAS DE LA BASE DE DATOS



5.5. DICCIONARIO DE DATOS

Tabla: managers

COLUMNA	KEY	TIPO	NULO	TIPO DATO	LONG	DESCRIPCIÓN
id	SÍ	AUTOINCREMENT	NO	INT	4	Identificador único del manager.
nombre	NO	-	NO	VARCHAR	100	Nombre del manager.
correo	NO	-	NO	VARCHAR	100	Correo del manager.
telefono	NO	-	SÍ	VARCHAR	15	Teléfono de contacto del manager.
contrasena	NO	-	NO	VARCHAR	100	Contraseña del manager
rol	NO		NO	VARCHAR	50	Rol del manager (definición de permisos y acceso).

Tabla: locacion

COLUMNA	KEY	TIPO	NULO	TIPO DATO	LONG	DESCRIPCIÓN
id	SÍ	AUTOINCREMENT	NO	INT	4	Identificador único de la locación.
nombre	NO	-	NO	VARCHAR	100	Nombre de la locación.
direccion	NO	-	NO	VARCHAR	200	Dirección física de la locación.
ciudad	NO	-	NO	VARCHAR	50	Ciudad donde se encuentra la locación.
region	NO	-	SÍ	VARCHAR	50	Región/estado de la locación.
codigo_postal	NO	-	SÍ	VARCHAR	10	Código postal de la locación.
pais	NO	-	NO	VARCHAR	50	País de la locación.
capacidad	NO	-	SÍ	INT	4	Capacidad máxima del recinto.
telefono	NO	-	SÍ	VARCHAR	15	Teléfono de contacto de la locación.
email	NO	-	SÍ	VARCHAR	100	Email de contacto de la locación.
manager_id	NO	FK	NO	INT	4	Referencia al manager que administra la locación.

Tabla: eventos

COLUMNA	KEY	TIPO	NULO	TIPO DATO	LONG	DESCRIPCIÓN
id	SÍ	AUTOINCREMENT	NO	INT	4	Identificador único del evento.
nombre	NO	-	NO	VARCHAR	100	Nombre del evento.
fecha_evento	NO	-	NO	DATE	-	Fecha en que se realizará el evento.
fecha_montaje	NO	-	SÍ	DATE	-	Fecha de montaje o preparación del evento.
locacion_id	NO	FK	NO	INT	4	Referencia a la locación donde se realizará.

Tabla: artistas

COLUMNA	KEY	TIPO	NULO	TIPO DATO	LONG	DESCRIPCIÓN
id	SÍ	AUTOINCREMENT	NO	INT	4	Identificador único del artista.
nombre	NO	-	NO	VARCHAR	100	Nombre artístico o real.
fecha_nacimiento	NO	-	SÍ	DATE	-	Fecha de nacimiento del artista.
nacionalidad	NO	-	SÍ	VARCHAR	50	País de origen.
banda	NO	-	SÍ	VARCHAR	100	Nombre de la banda a la que pertenece.
imagen	NO	-	SÍ	BYTEA	-	Imagen del artista

Tabla: instrumentos

COLUMNA	KEY	TIPO	NULO	TIPO DATO	LONG	DESCRIPCIÓN
id	SÍ	AUTOINCREMENT	NO	INT	4	Identificador único del instrumento.
nombre	NO	-	NO	VARCHAR	100	Nombre del instrumento.
cantidad	NO	-	NO	INT	4	Cantidad disponible del instrumento.
proveedor	NO	-	SÍ	VARCHAR	100	Nombre del proveedor o fuente del instrumento.
manager_id	NO	FK	NO	INT	4	Referencia al manager dueño del instrumento.

Tabla: artistas_managers

COLUMNA	KEY	TIPO	NULO	TIPO DATO	LONG	DESCRIPCIÓN
artista_id	PK	FK	NO	INT	4	Identificador del artista.
manager_id	PK	FK	NO	INT	4	Identificador del manager

Tabla: evento_artistas

COLUMNA	KEY	TIPO	NULO	TIPO DATO	LONG	DESCRIPCIÓN
evento_id	PK	FK	NO	INT	4	Identificador del evento.
artista_id	PK	FK	NO	INT	4	Identificador del artista.

Tabla: evento_instrumentos

COLUMNA	KEY	TIPO	NULO	TIPO DATO	LONG	DESCRIPCIÓN
evento_id	PK	FK	NO	INT	4	Identificador del evento.
instrumento_id	PK	FK	NO	INT	4	Identificador del instrumento.

Tabla: evento_managers

COLUMNA	KEY	TIPO	NULO	TIPO DATO	LONG	DESCRIPCIÓN
evento_id	PK	FK	NO	INT	4	Identificador del evento.
manager_id	PK	FK	NO	INT	4	Identificador del manager asociado.

5.6. SCRIPT DE CREACIÓN E INSERCIÓN DE LA BASE DE DATOS

SCRIPT DE CREACIÓN

```
CREATE TABLE managers (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    correo VARCHAR(100) NOT NULL,
    telefono VARCHAR(15),
    contrasena VARCHAR(100) NOT NULL,
    rol VARCHAR(50) NOT NULL DEFAULT 'manager'
);


CREATE TABLE locacion (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    direccion VARCHAR(200) NOT NULL,
    ciudad VARCHAR(50) NOT NULL,
    region VARCHAR(50),
    codigo_postal VARCHAR(10),
    pais VARCHAR(50) NOT NULL,
    capacidad INT,
    telefono VARCHAR(15),
```

```
    email VARCHAR(100),
    manager_id INT NOT NULL,
    FOREIGN KEY (manager_id) REFERENCES managers(id)
);

CREATE TABLE eventos (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    fecha_evento DATE NOT NULL,
    fecha_montaje DATE,
    locacion_id INT NOT NULL,
    FOREIGN KEY (locacion_id) REFERENCES locacion(id)
);

CREATE TABLE artistas (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    fecha_nacimiento DATE,
    nacionalidad VARCHAR(50),
    banda VARCHAR(100),
    imagen BYTEA
);

CREATE TABLE instrumentos (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
```

```
    cantidad INT NOT NULL,
    proveedor VARCHAR(100),
    manager_id INT NOT NULL,
    FOREIGN KEY (manager_id) REFERENCES managers(id)
);

CREATE TABLE artistas_managers (
    artista_id INT,
    manager_id INT,
    PRIMARY KEY (artista_id, manager_id),
    FOREIGN KEY (artista_id) REFERENCES artistas(id),
    FOREIGN KEY (manager_id) REFERENCES managers(id)
);

CREATE TABLE evento_artistas (
    evento_id INT,
    artista_id INT,
    PRIMARY KEY (evento_id, artista_id),
    FOREIGN KEY (evento_id) REFERENCES eventos(id),
    FOREIGN KEY (artista_id) REFERENCES artistas(id)
);

CREATE TABLE evento_instrumentos (
    evento_id INT,
    instrumento_id INT,
    PRIMARY KEY (evento_id, instrumento_id),
```

```
    FOREIGN KEY (evento_id) REFERENCES eventos(id),
    FOREIGN KEY (instrumento_id) REFERENCES instrumentos(id)
);

CREATE TABLE evento_managers (
    evento_id INT,
    manager_id INT,
    PRIMARY KEY (evento_id, manager_id),
    FOREIGN KEY (evento_id) REFERENCES eventos(id),
    FOREIGN KEY (manager_id) REFERENCES managers(id)
);
```

SCRIPT DE INSERCIÓN

```
-- Managers
INSERT INTO managers (nombre, correo, telefono, contrasena) VALUES
('Ana Torres', 'ana@example.com', '123456789', 'pass123'),
('Carlos Pérez', 'carlos@example.com', '987654321', 'clave456'),
('Lucía Gómez', 'lucia@example.com', '456789123', 'secure789'),
('Marta Díaz', 'marta@example.com', '741852963', 'qwerty123'),
('Jorge Silva', 'jorge@example.com', '369258147', 'admin456');

-- Locaciones
INSERT INTO locacion (nombre, direccion, ciudad, region, codigo_postal, pais, capacidad, telefono, email, manager_id) VALUES
('Teatro Central', 'Av. Principal 123', 'Madrid', 'Madrid', '28001', 'España', 500, '911223344', 'central@teatro.com', 1),
```

('Auditorio Sur', 'Calle Sur 456', 'Sevilla', 'Andalucía', '41001', 'España', 300, '955667788', 'auditorio@sur.com', 2),

('Sala Norte', 'Av. Norte 789', 'Bilbao', 'País Vasco', '48001', 'España', 200, '944112233', 'norte@sala.com', 3),

('Centro Musical', 'Calle Música 321', 'Valencia', 'Comunidad Valenciana', '46001', 'España', 400, '963852741', 'musical@centro.com', 4),

('Espacio Sonoro', 'Paseo Acústico 654', 'Barcelona', 'Cataluña', '08001', 'España', 600, '933112233', 'sonoro@espacio.com', 5);

-- Artistas

INSERT INTO artistas (nombre, fecha_nacimiento, nacionalidad, banda, imagen) VALUES

('Lola Indigo', '1992-04-01', 'España', 'Lola Indigo Band', NULL),

('Pablo Alborán', '1989-06-12', 'España', NULL, NULL),

('Rosalía', '1993-09-25', 'España', NULL, NULL),

('David Bisbal', '1979-06-05', 'España', NULL, NULL),

('Vetusta Morla', NULL, 'España', 'Vetusta Morla', NULL);

-- Instrumentos

INSERT INTO instrumentos (nombre, cantidad, proveedor, manager_id) VALUES

('Guitarra eléctrica', 5, 'Yamaha', 1),

('Batería', 2, 'Pearl', 2),

('Teclado', 3, 'Roland', 3),

('Micrófono', 10, 'Shure', 4),

('Bajo', 4, 'Fender', 5);

-- Eventos

```
INSERT INTO eventos (nombre, fecha_evento, fecha_montaje, locacion_id) VALUES
('Festival Primavera', '2025-06-01', '2025-05-31', 1),
('Concierto de Verano', '2025-07-15', '2025-07-14', 2),
('Noche Acústica', '2025-08-20', '2025-08-19', 3),
('Rock Fest', '2025-09-10', '2025-09-09', 4),
('Pop Night', '2025-10-05', '2025-10-04', 5);

-- Relación Artistas - Managers
INSERT INTO artistas_managers (artista_id, manager_id) VALUES
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5);

-- Relación Evento - Artistas
INSERT INTO evento_artistas (evento_id, artista_id) VALUES
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5);

-- Relación Evento - Instrumentos
INSERT INTO evento_instrumentos (evento_id, instrumento_id) VALUES
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5);

-- Relación Evento - Managers
INSERT INTO evento_managers (evento_id, manager_id) VALUES
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5);
```

6. VISIÓN GENERAL DEL CÓDIGO

La aplicación **Gestor de Eventos Musicales** está desarrollada con .NET MAUI y utiliza una arquitectura sencilla, estructurada en carpetas que separan los distintos componentes del proyecto. A continuación, se detalla la estructura principal:

- **Converters/**
  Contiene convertidores utilizados en XAML para lógica de presentación.
- **Data/**
  Contiene la lógica de acceso a datos.
- **Modelos/**
  Define las clases que representan las entidades de la aplicación: Evento, Artista, Instrumento, Locacion, y Manager.
- **Paginas/**
  Contiene las vistas (páginas) de la app. Cada página XAML tiene su correspondiente archivo de código detrás (.xaml.cs).
- **Resources/**
  Contiene recursos gráficos, fuentes, íconos y estilos definidos para la UI.
- **App.xaml y AppShell.xaml**
  Definen la navegación principal de la aplicación y los estilos globales.
- **MauiProgram.cs**
  Configura los servicios y la inicialización de la aplicación.

7. PRUEBAS Y VALIDACIONES

7.1. Pruebas Unitarias

- **Objetivo**:el objetivo de las pruebas unitarias es el de verificar el correcto funcionamiento de las unidades más pequeñas de la aplicación. Por ejemplo, funciones, métodos y clases individuales.
- **Descripción**: Se realiza una prueba de cada función del sistema para asegurarse de que funcione como se espera en condiciones normales.
- **Herramientas a usar**: xUnit, NUnit, JUnit.
- **Ejemplo**: Se prueba la función EliminarArtistaAsync para garantizar que se eliminará correctamente un artista de la base de datos y se maneja de forma adecuada las relaciones con los managers.

7.2. Pruebas de Integración

- **Objetivo**: Asegurarse de que diferentes módulos del sistema interactúan correctamente entre sí.
- **Descripción**: Se prueban las interacciones entre diferentes partes del sistema, como las consultas a la base de datos, las APIs, y los servicios.

- **Ejemplo**: un ejemplo de donde podría hacer uso de estas pruebas es en la comprobación de que el proceso de creación de los eventos desde la interfaz de usuario se realicen correctamente, pasando los datos a la base de datos y manteniendo las relaciones entre eventos, artistas y managers.

7.3. Pruebas de Sistema

- **Objetivo**: Validar que el sistema completo funcione de acuerdo a lo esperado en un entorno controlado.
- **Descripción**: El sistema se prueba como un todo, incluyendo la interfaz de usuario, la funcionalidad del backend, la interacción con la base de datos y la integración con otros sistemas externos.
- **Ejemplo**: es al verificar que al registrar un nuevo manager y un artista, este último se asocie correctamente al manager y se muestre correctamente en la interfaz.

7.4. Pruebas de Aceptación

- **Objetivo**: Asegurarse de que el sistema cumpla con los requisitos del cliente y funcione como el cliente lo espera.
- **Descripción**: Estas pruebas son realizadas con usuarios finales para validar que el sistema es útil y cumple con las necesidades del usuario.
- **Herramientas a usar**: No se requieren herramientas específicas, se realizan de forma manual.
- **Ejemplo**: Los managers prueban la creación y eliminación de artistas y eventos para verificar que la aplicación funcione sin problemas desde la perspectiva del usuario final.

7.5. Pruebas de Rendimiento

- **Objetivo**: Asegurarse de que el sistema funcione de manera eficiente y responda adecuadamente bajo diferentes condiciones de carga.
- **Descripción**: Estas pruebas permiten simular múltiples usuarios interactuando con el sistema simultáneamente para observar cómo responde el sistema a la carga y la escalabilidad.
- **Herramientas a usar**: Apache JMeter, LoadRunner.
- **Ejemplo**: Simular que 100 usuarios intenten crear eventos simultáneamente para verificar que el sistema mantenga una respuesta adecuada sin caídas.

7.6. Pruebas de Seguridad

- **Objetivo**: Asegurarse de que el sistema esté protegido contra accesos no autorizados y posibles vulnerabilidades.

- **Descripción**: Se realizan pruebas para detectar posibles vulnerabilidades de seguridad como inyecciones de SQL.
- **Ejemplo**: Comprobar que un manager no pueda acceder a los datos de otros managers o artistas a los que no tenga permiso.

7.7. Validación de Integridad de Datos

- **Objetivo**: Asegurar que los datos sean consistentes en la base de datos y que las relaciones entre las tablas no se rompan.
- **Descripción**: esta prueba se asegura que en los registros de las tablas estén correctamente relacionados, y que no haya valores nulos o inconsistentes que puedan romper la integridad referencial.
- **Ejemplo**: Validar que todos los artistas tengan un manager_id válido que apunte a un registro existente en la tabla de managers.

8. DIFICULTADES DETECTADAS Y POSIBLES MEJORAS

Durante el desarrollo de la aplicación, me encontré con diversos problemas que afectaron al diseño y a la implementación del sistema. Algunos de los problemas encontrados son:

8.1. Dificultades Detectadas

- **Complejidad en la gestión de relaciones entre tablas**: la base de datos incluye varias relaciones que eran de tipo muchos a muchos, esto me obligó la creación de tablas intermedias y una logica adicional para el código para poder insertar, eliminar o modificar datos.
- **Errores en la conversión de tipos de datos**: Un error que tenía era el intento de convertir valores de long e int de forma incorrecta, lo cual me generaba excepciones que impedían la correcta ejecución de funciones de edición.
- **Restricciones de claves foráneas**: Hubieron varios momentos que al tratar de insertar datos en tablas como “locacion” se producían errores debido a que el manager_id u otros casos más concretos, aún no existía.
- **Falta de tiempo para implementar ciertos roles**: en un principio, pensaba en implementar un diseño en el que el sistema era mucho más amplio, siendo que los artistas debían registrarse como usuarios para poder ser asociados a un manager. Sin embargo, debido al tiempo limitado disponible, se optó por eliminar la gestión de usuarios tipo artista, ya que requería un flujo más complejo donde el artista debía crear su cuenta antes de poder ser asignado a un manager y participar en eventos.

8.2. Posibles Mejoras:

- **Añadir nuevamente el sistema de usuarios para artistas:** Como primera mejora, se podría seguir el plan inicial de introducir un sistema de registro también para los artistas: permitiendo así que estos puedan crear una cuenta, administrar sus datos y ver sus participaciones en los eventos
- **Sistema de autenticación y roles más completo:** con el programa actual, solo se gestiona el inicio de sesión para los managers. También se puede añadir un sistema de roles donde también haya usuarios de tipo staff, técnicos, o productores con distintos niveles de acceso.
- **Gestión avanzada de disponibilidad de recursos:** Se podría implementar un sistema que gestione la disponibilidad de los instrumentos para los eventos, evitando asignaciones en fechas duplicadas.
- **Mejoras en la interfaz de usuario (UI/UX):** Se pueden implementar interfaces más intuitivas y modernas, también implementar notificaciones o mensajes de confirmación más detallados.
- **Exportar pdfs**: Se podría implementar la generación de reportes en PDF en el que se muestre la información de los eventos, artistas, y distintos recursos usados.
- **Validaciones automáticas de conflictos**: El sistema podría mejorar al incorporar validaciones que detecten conflictos entre fechas de eventos, asignación duplicada de artistas o instrumentos, entre otros.
- **Implementación de sistema de mensajería interna**: también es posible desarrollar un sistema de chat o mensajería interna que facilite la comunicación directa entre managers, trabajadores y artistas (o entre distintos tipos de usuarios en el futuro), mejorando la coordinación para la organización de eventos.

9. TEMPORALIZACIÓN DEL PROYECTO

Fase / Tarea	Fecha de Inicio	Fecha de Finalización	Estado
Análisis de requisitos	05/04/2025	06/04/2025	Completado
Diseño de base de datos	07/04/2025	08/04/2025	Completado
Creación de modelos y estructura inicial	09/04/2025	11/04/2025	Completado
Desarrollar la funcionalidad pantalla de login y registro de managers	12/04/2025	13/04/2025	Completado
Diseño e implementación inicial de las pantallas principales (creación y edición de eventos, artistas, locaciones e	14/04/2025	14/04/2025	Completado

Fase / Tarea	Fecha de Inicio	Fecha de Finalización	Estado
instrumentos, sin lógica interna)			
Desarrollo de la funcionalidad de las paginas de artistas, instrumentos y locación	15/04/2025	20/04/2025	Completado
Desarrollo de las funcionalidades de las pantallas de creación y edición de eventos	21/04/2025	24/04/2025	Completado
Sistema de eliminación y edición de las pantallas previamente creadas	25/04/2025	27/04/2025	Completado
Ajustes en la base de datos	28/04/2025	29/04/2025	Completado
Documentación técnica	07/04/2025	30/05/2025	Completado
Implementación pantalla de perfil	07/05/2025	11/05/2025	Completado
Mejoras visuales y decoración (UI/UX)	12/05/2025	21/05/2025	Completado
Pruebas finales y validación	27/05/2025	27/05/2025	Completado
Entrega final	27/05/2025	27/05/2025	Completado

10. DESPLIEGUE DE LA APLICACIÓN

La aplicación “Gestor de Eventos Musicales” ha sido desarrollada con el framework de .NET MAUI. es un framework orientado a aplicaciones multiplataforma. El objetivo principal es distribuir la aplicación en dispositivos Android a través de Google Play Store.

10.1. Modalidad de despliegue

- Actualmente, la aplicación se ejecuta **de forma local** en el dispositivo Android, conectándose directamente a una base de datos **PostgreSQL** que debe estar alojada de forma accesible (por ejemplo, mediante una dirección IP o a través de un túnel como **Ngrok**).
- En esta versión no se utiliza un backend hospedado en la nube ni un servidor intermedio con API REST.

10.2. Tecnologías utilizadas

- **Plataforma de desarrollo**: .NET MAUI (Multi-platform App UI)

- **Lenguaje de programación**: C#
- **Base de datos**: PostgreSQL
- **Cliente de administración de base de datos**: pgAdmin
- **Acceso a datos**: ADO.NET mediante la librería **Npgsql**
- **Gestión de sesión local**: Preferences de .NET MAUI
- **IDE**: Visual Studio 2022
- **Sistema operativo objetivo**: Android
- **Distribución final**: Archivo APK para distribución en **Google Play Store**
- **• Túnel de desarrollo remoto:** ngrok (para exponer servicios locales a dispositivos móviles durante pruebas)

10.3. Acceso y ejecución en entorno de desarrollo

Para ejecutar la aplicación en desarrollo:

1. Se debe contar con un servidor PostgreSQL activo y accesible (por ejemplo, vía **Ngrok**, como se configuró durante el desarrollo).
2. La cadena de conexión se encuentra configurada en la clase DatabaseService.
3. Se ejecuta el proyecto desde Visual Studio 2022, conectando un emulador o dispositivo físico Android.
4. La app funciona de manera totalmente local desde el móvil, conectándose a la base de datos remota configurada.

10.4. Proyección de despliegue en producción

- En producción, la aplicación será exportada como **APK/AAB** para su **publicación en Google Play Store**.
- El servidor de base de datos deberá estar accesible mediante una IP fija o hosting en la nube si se requiere conectividad global.
- Se prevé para versiones futuras la incorporación de una **API REST intermedia** que separe la lógica de negocio del cliente móvil.

10.5. Conexión con base de datos remota mediante túnel seguro (Ngrok)

Durante el desarrollo de la aplicación, fue necesario realizar una conexión remota entre el dispositivo Android y una base de datos PostgreSQL alojada en un servidor local. Sin embargo, debido a restricciones impuestas por el proveedor de servicios de Internet (ISP), no fue posible abrir los puertos necesarios para habilitar conexiones externas directamente.

Como solución, se utilizó **NGROK**, una herramienta que permite exponer servicios locales a Internet a través de túneles seguros. Gracias a **NGROK**, fue posible habilitar el acceso remoto a la base de datos sin necesidad de modificar la configuración del router ni pagar servicios adicionales al ISP.

10.5.1. Funcionamiento general:

- NGROK se ejecuta en el mismo equipo que aloja la base de datos PostgreSQL.
- A través de un comando en la terminal (por ejemplo: ngrok tcp 5432), se genera un túnel hacia el puerto 5432 (por defecto usado por PostgreSQL).
- La herramienta devuelve una URL pública con un nuevo puerto de acceso (por ejemplo: tcp://4.tcp.ngrok.io:12345), que puede ser utilizada temporalmente en la cadena de conexión de la aplicación.

10.5.2. Consideraciones importantes:

- El enlace y puerto generados por NGROK son **temporales**. Cada vez que se reinicia NGROK, se obtiene una nueva dirección y puerto, por lo que es necesario **actualizar manualmente la cadena de conexión** en el código.
- Ejemplo de cadena de conexión actualizada:

private readonly string _connectionString =
"Host=4.tcp.ngrok.io;Port=12345;Username=postgres;Password=0603;Database=GestorEventosDB";

Este cambio se tiene que hacer **SOLO** en el archivo **DataService.cs** alojado en la carpeta Data y solo se tiene que cambiar **UNA** linea de código, la cuál es la siguiente:

- Este método resulta útil para **pruebas de desarrollo**, pero no es viable a largo plazo para un entorno de producción.

En versiones futuras, se recomienda utilizar un servidor de base de datos con IP fija o alojado en la nube, o bien implementar una API intermedia que permita desacoplar la lógica de acceso a datos de la aplicación móvil.

11. MANUAL DE USUARIO

11.1. Descripción del producto

El producto de **“Gestor de Eventos Musicales”** es una aplicación orientada a los managers dentro del ámbito musical en el que se permite organizar y gestionar eventos musicales de forma sencilla e intuitiva. El propósito es facilitar la administración de eventos, permitiendo agregar y modificar información relevante como “locaciones”, “artistas”, “instrumentos” y los responsables (managers) implicados.

Esta aplicación está dirigida a profesionales del sector musical, en especial a managers que necesitan una herramienta rápida y funcional para gestionar múltiples eventos desde un solo lugar.

11.2. Instrucciones de instalación o ejecución

11.2.1. Si se entrega a través de Google Play Store:

1. Acceda a la Play Store desde su dispositivo Android.
2. Busque el nombre de la aplicación: **Gestor de Eventos Musicales**.
3. Pulse en "Instalar" y espere a que finalice la descarga.
4. Una vez instalada, podrá acceder a la aplicación desde el menú principal del dispositivo.

11.2.2. Si se entrega como APK:

1. Transfiera el archivo .apk al dispositivo Android.
2. Active la opción **"Permitir instalar aplicaciones de fuentes desconocidas"** desde Ajustes > Seguridad.
3. Ejecute el archivo .apk y siga los pasos del instalador.
4. Una vez instalada, podrá acceder a la aplicación desde el menú principal del dispositivo.

11.2.3. Si se entrega como proyecto MAUI:

1. Abra el proyecto con **Visual Studio 2022 o superior** (con soporte para .NET MAUI).
2. Compile el proyecto en el dispositivo Android o emulador usando la opción **Deploy > Android Emulator / Android Device**.
3. Una vez instalado, la aplicación estará lista para su uso.

11.3. Requerimientos mínimos

11.3.1. Software:

- Android 8.0 (Oreo) o superior.
- .NET MAUI compatible en caso de compilar desde código.
- Visual Studio 2022 con workload de MAUI para ejecutarlo desde el código fuente.

11.3.2. Hardware:

- Dispositivo Android con al menos 2 GB de RAM.
- 100 MB de espacio libre en el dispositivo.
- Conexión a internet para sincronización remota (opcional si se usa con base de datos local).

11.4. Acceso a la aplicación

Al iniciar la app, el usuario podrá:

1. **Registrarse** como manager ingresando su nombre, correo electrónico y teléfono.
2. **Iniciar sesión** con su correo previamente registrado.

Una vez logueado, el usuario permanecerá en su sección incluso si cierra la app, hasta que seleccione manualmente la opción de cerrar sesión.

11.5. Funcionalidades principales

El usuario puede:

- Crear, editar y eliminar eventos musicales.
- Asignar locaciones, artistas, instrumentos y managers a cada evento.
- Visualizar la lista de eventos actuales.
- Editar su propio perfil de manager.

11.6. Interfaz de usuario (Resumen)

La interfaz del programa **Gestor de Eventos Musicales** es intuitiva y minimalista. Al ingresar, el usuario encuentra una pantalla principal con pestañas o botones que permiten navegar entre:

- **Editar y ver eventos**: donde puede ver la lista de eventos y agregar nuevos.
- **Perfil**: con los datos del manager actual.
- **Crear evento**: pantallas específicas con campos editables y listas personalizadas de selección de artistas, instrumentos y locaciones.
- **Editar y ver Artistas**:permite gestionar el catálogo de artistas disponibles, incluyendo la creación de nuevos artistas, edición de los existentes y eliminación si es necesario.
- **Editar y ver locaciones:** ofrece una vista completa de las locaciones registradas, permitiendo modificarlas, eliminarlas o añadir nuevas con todos sus datos (dirección, capacidad, ciudad, etc.).
- **Editar y ver instrumentos:** permite administrar los instrumentos disponibles, asociarlos a eventos y realizar cambios en sus propiedades.

Por ejemplo, para crear un nuevo evento:

1. Presione el botón “Nuevo Evento”.
2. Ingrese el nombre, seleccione fecha de evento y fecha de montaje.
3. Elija una locación y añada artistas e instrumentos disponibles.
4. Guarde el evento para almacenarlo en la base de datos.

Los formularios guían paso a paso al usuario con campos validados para evitar errores.

12. MANTENIMIENTO

El mantenimiento de la aplicación “Gestor de Eventos Musicales” se divide en dos áreas principales: la aplicación móvil y la base de datos PostgreSQL. Mantener el sistema en buen estado requiere monitorización periódica, actualización de versiones, y ajustes técnicos tanto en el código como en el entorno de ejecución. Además, con la incorporación del nuevo rol administrador (admin), se deben contemplar tareas específicas relacionadas con la gestión de permisos y seguridad.

12.1. Mantenimiento de la aplicación móvil

- Las actualizaciones se realizarán mediante nuevas versiones publicadas en **Google Play Store**.
- Cualquier cambio en funcionalidades, corrección de errores o mejoras visuales se desarrollarán en el entorno de desarrollo (**Visual Studio 2022**) y se exportarán como archivos **.APK** para su despliegue.
- Se recomienda probar cada nueva versión en un dispositivo real antes de su publicación.
- El responsable del mantenimiento debe revisar regularmente los reportes de fallos y opiniones de los usuarios para priorizar mejoras.
- Se deberá asegurar que la lógica de control de acceso basada en roles funcione correctamente, garantizando que los usuarios con rol admin tengan privilegios exclusivos para gestionar usuarios, modificar configuraciones críticas y acceder a funcionalidades restringidas.

12.2. Mantenimiento de la base de datos

- La base de datos PostgreSQL debe mantenerse operativa y segura, revisando periódicamente la disponibilidad del servidor, realizando copias de seguridad y monitorizando el acceso.

- Debido al uso de **Ngrok** que es un túnel temporal durante la etapa de desarrollo, se plantea la posibilidad de migrar a una solución más estable (como servicio en la nube) para el entorno de producción.
- Los cambios en la estructura de la base de datos (como agregar nuevas tablas o campos) deben planificarse cuidadosamente para evitar afectar la integridad del sistema. En caso de querer hacer pruebas, se recomienda crear una nueva base de datos y hacer las pruebas allí.

12.3. Rol de la persona encargada del mantenimiento

- El **desarrollador principal** o el **equipo técnico asignado** será responsable de:
    - Realizar mejoras y aplicar actualizaciones periódicas.
    - Corregir errores reportados por usuarios.
    - Realizar migraciones y respaldos de la base de datos.
    - Mantener actualizada la cadena de conexión si cambia la ubicación del servidor PostgreSQL.
    - Monitorizar el correcto funcionamiento del sistema en dispositivos móviles.

13. CONCLUSIONES

El desarrollo del proyecto **Gestor de Eventos Musicales** ha sido una experiencia enriquecedora, tanto a nivel técnico como personal. Desde el inicio hasta su implementación final, se han abordado múltiples aspectos clave del desarrollo de aplicaciones móviles, incluyendo diseño de interfaces, modelado de bases de datos, gestión de usuarios y relaciones entre entidades complejas.

13.1. Evaluación de resultados y cumplimiento de objetivos

Los objetivos iniciales del proyecto se han cumplido en su totalidad:

- Se ha desarrollado una aplicación funcional y estable orientada a managers dentro del ámbito musical.
- Se han implementado las funcionalidades principales previstas: creación, edición y visualización de eventos, así como la gestión de entidades relacionadas.
- Se ha garantizado una experiencia de usuario intuitiva y fluida, gracias a un diseño de interfaz limpio.

Además, se han añadido mejoras no previstas inicialmente, como la persistencia de sesión tras el login y una lógica personalizada de visibilidad según el usuario activo.

13.2. Dificultades encontradas y soluciones

Durante el desarrollo se presentaron diversas dificultades técnicas, entre ellas:

- **Relaciones muchos-a-muchos:** entre eventos y otras entidades como artistas o instrumentos. Estas se resolvieron creando tablas intermedias y adaptando la lógica de inserción y consulta.
- **Gestión de la sesión del usuario:** que implicó el uso de Preferences para mantener el estado activo entre cierres de la aplicación.
- **Interfaz dinámica sin Pickers:** lo cual requirió una aproximación personalizada para seleccionar y mostrar datos sin depender de controles predefinidos.

Estas dificultades fueron abordadas con investigación adicional, pruebas constantes y adaptando el diseño inicial cuando fue necesario, lo que permitió adquirir nuevos conocimientos y mejorar la calidad final del proyecto.

13.3. Valoración final

El proyecto no solo ha cumplido con los objetivos funcionales planteados, sino que también me ha permitido profundizar en el uso de tecnologías modernas como .NET MAUI y consolidar buenas prácticas de desarrollo. El resultado es una aplicación práctica, sólida y con potencial de mejora o expansión futura.

14. GLOSARIO DE SIGLAS Y ABREVIATURAS

Acrónimo	Significado
MAUI	Multi-platform App UI (Interfaz de Aplicaciones Multiplataforma)
.NET	Network Enabled Technology (Tecnología habilitada para red)
C#	C Sharp (Lenguaje de programación orientado a objetos desarrollado por Microsoft)
IDE	Integrated Development Environment (Entorno de desarrollo integrado)
APK	Android Package (Paquete de instalación para dispositivos Android)
AAB	Android App Bundle (Formato de publicación recomendado por Google para apps Android)
DB	Database (Base de datos)
ORM	Object-Relational Mapping (Mapeo Objeto-Relacional)
UI	User Interface (Interfaz de Usuario)
UX	User Experience (Experiencia de Usuario)
SQL	Structured Query Language (Lenguaje de Consulta Estructurado)
pgAdmin	PostgreSQL Administration Tool (Herramienta para administración de PostgreSQL)
Npgsql	.NET Data Provider for PostgreSQL

	(Proveedor de datos .NET para PostgreSQL)
REST	Representational State Transfer (Estilo de arquitectura para servicios web)
API	Application Programming Interface (Interfaz de Programación de Aplicaciones)
VPS	Virtual Private Server (Servidor Privado Virtual)

Tutora: María del Carmen Buenestado Fernández