Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Ingeniería de Software

Aplicación multiplataforma para la gestión y visualización de eventos musicales

Autor:
Víctor Barreda Sáez

Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Ingeniería de Software

Aplicación multiplataforma para la gestión y visualización de eventos musicales

Autor:
Víctor Barreda Sáez

Tutores:
Mario Corrales Astorgano
Juan Alberto Muñoz Cristóbal

Me gustaría agradecer a mi familia y amigos, que me han acompañado durante todo el proceso.
Muchas gracias por todo.

Resumen

La música es uno de los principales pasatiempos de ocio para las personas, siendo un elemento fundamental en la vida social y personal. Es por eso que este proyecto nace con el objetivo de potenciar la experiencia musical en la provincia de Valladolid, facilitando tanto el acceso a eventos como su creación y gestión.

El proyecto contempla el desarrollo de dos aplicaciones. La primera aplicación estará enfocada en la visualización información sobre eventos musicales . Esta aplicación permitirá a los usuarios visualizar diferentes eventos musicales, aplicar filtros de búsqueda para una navegación más precisa, y compartir los conciertos de su interés. Además, ofrecerá la posibilidad de guardar eventos en una lista de favoritos, facilitando el seguimiento de los mismos.

Por otro lado, se ha desarrollado una aplicación orientada a la gestión de dichos eventos a través de navegadores web. Para utilizarla, es necesario contar con una cuenta de usuario. Una vez autenticados, los usuarios podrán administrar (agregar, editar y visualizar) los eventos musicales disponibles para el público general. Las funcionalidades disponibles dependerán del rol asignado. Los usuarios con rol de administrador tendrán acceso a opciones avanzadas, como aceptar o rechazar propuestas realizadas por otros usuarios, crear conciertos directamente, crear usuarios y modificar eventos existentes. En cambio, los usuarios con rol de externo podrán presentar propuestas de eventos, las cuales quedarán sujetas a revisión por parte de los administradores.

Abstract

Music is one of the main leisure activities for people, playing a fundamental role in both social and personal life. This project was born with the aim of enhancing the musical experience in the province of Valladolid, making it easier to access, create, and manage music events.

The project involves the development of two applications. The first application focuses on displaying information about musical events. It allows users to view various concerts, apply search filters for more precise navigation, and share events of interest. Additionally, it offers the option to save events to a favorites list, making it easier to keep track of them.

On the other hand, a second application has been developed for managing these events through web browsers. To use it, users must have an account. Once authenticated, they can manage (add, edit, and view) music events available to the general public. The available features depend on the assigned role. Users with an administrator role will have access to advanced options, such as accepting or rejecting proposals submitted by other users, creating concerts directly, creating user accounts, and editing existing events. In contrast, users with an external role will be able to submit event proposals, which will be subject to review by administrators.

Índice

Índice de figuras

Índice de cuadros

Capítulo 1

1. Introducción

1.1. Contexto

Watios y Decibelios [50] es una pagina web dedicada a la promoción y difusión de eventos musicales, especialmente conciertos, giras y festivales, que tienen lugar en Valladolid y sus alrededores. La página está orientada a facilitar a los usuarios el acceso a información sobre los eventos musicales de la región. Ofrece herramientas de búsqueda categorizadas, permitiendo filtrar eventos según el tipo de actividad musical. Además, incluye datos identificativos esenciales de cada concierto.

Actualmente, esta web funciona sobre la plataforma WordPress, un sistema de gestión de contenidos que permite crear, gestionar y mantener sitios web sin necesidad de conocimientos avanzados en programación.

Su gestión depende en gran medida del administrador, quien debe realizar manualmente la carga, edición y organización del contenido. Este enfoque, aunque funcional, resulta poco eficiente debido a la dependencia de procesos repetitivos y a la falta de herramientas avanzadas que faciliten la administración y optimización de la experiencia de usuario.

1.2. Motivación

El objetivo principal de este proyecto es la mejora y actualización de la página web actual Watios y Decibelios.

Esta actualización que se propone en este TFG se enfocará en tres áreas clave: la optimización del flujo de trabajo del administrador, la mejora en la organización y visualización de los contenidos, y la incorporación de funcionalidades avanzadas que aporten mayor valor tanto al administrador como a los usuarios de la página.

Para lograr esto se hará el uso de React-Native como tecnología para la interfaz de usuario, la incorporación de un backend personalizado para manejar la lógica y funcionalidad de la plataforma, y una gestión de la base de datos más robusta, empleando PostgresSQL para desarrollar una aplicación para dispositivos móviles.

Por otro lado se desarrollará una aplicación de gestión empleando Pyhton para la gestión de la parte lógica junto con la librería Streamlit para el desarrollo de la interfaz. Concretamente, el proyecto abordará la creación de una interfaz de administración más intuitiva y eficiente, que permita a los gestores de Watios y Decibelios organizar y subir el contenido con un mínimo de pasos, reduciendo los tiempos de trabajo.

Respecto al ámbito personal, la motivación principal de este proyecto radica en el deseo de desarrollar un trabajo que sea útil y, al mismo tiempo, permita aprender y emplear herramientas modernas del desarrollo de aplicaciones, tanto web como móviles. Este proyecto busca aprovechar la oportunidad para explorar cómo se utilizan actualmente las tecnologías y metodologías más recientes en la creación de aplicaciones.

Otro aliciente importante es que este proyecto dará lugar a una aplicación real y funcional, lo que añade una motivación extra para su desarrollo. La posibilidad de crear algo que llegue a ser

utilizado por un público real incrementa la satisfacción personal al saber que el trabajo realizado tendrá una utilidad práctica y tangible.

1.3. Aplicaciones similares

En primer lugar, es importante ofrecer una breve descripción de la aplicación web que buscamos complementar con el desarrollo de nuestra aplicación móvil. Watios y Decibelios se trata de una página web que permite consultar los conciertos programados en Valladolid y provincia. Su funcionalidad principal se centra en ofrecer tres opciones: la visualización de conciertos, la visualización de giras y la visualización de festivales.

Actualmente, la aplicación Watios y Decibelios sigue operativa y funcional, dado que el objetivo principal del desarrollo de la nueva aplicación no es reemplazar la versión existente, sino complementarla y expandir sus funcionalidades. La aplicación original permanece accesible a través del siguiente enlace: [50].

Sin embargo, la aplicación actual presenta una serie de limitaciones que justifican la necesidad de una nueva implementación. Al estar desarrollada sobre la plataforma WordPress, su estructura resulta básica y poco flexible para las necesidades actuales. La gestión del contenido depende completamente del administrador, quien debe realizar de manera manual la carga, edición y eliminación de cada elemento.

Además, la ausencia de herramientas colaborativas o de permisos diferenciados restringe la posibilidad de que otros usuarios o miembros del equipo contribuyan al mantenimiento y actualización de los contenidos, lo cual facilitaría en gran manera el trabajo del administrador e invitaría a más usuarios a la aplicación.

1.3.1. Watios y Decibelios

La página web de Watios y Decibelios actualmente sigue activa y funcional a través del siguiente enlace: [50]. Esta brinda acceso a información sobre conciertos giras y festivales musicales en la provincia de Valladolid. Esta es gestionada por un usuario el cual introduce y gestiona todos los datos de la misma.

Desventajas

- La plataforma actual (WordPress) presenta limitaciones en cuanto a flexibilidad y personalización de funciones.
- La dependencia del administrador para la carga y edición de contenido es poco eficiente.
- No existen herramientas colaborativas ni permisos diferenciados para permitir la contribución de más personas al mantenimiento de la página.

Ventajas

- Libertad total en estructuras de datos y diseño: el administrador controla completamente la información y su presentación en la web debido a que la tecnología empleada (WordPress) es muy permisiva.

Figura 1: Inicio de la página web

En este caso, la información relativa a los conciertos se presenta de forma sencilla, utilizando un menú desplegable que muestra los conciertos agrupados en funcion de los dias en los que se llevaran a cabo.

Figura 2: Expositor de los festivales

Figura 3: Vista de los datos de un festival

1.3.2. Wegow

Wegow es una aplicación móvil diseñada para ofrecer a los usuarios información sobre conciertos y eventos musicales. Permite descubrir, seguir y asistir a conciertos de artistas y grupos favoritos, comprar entradas, y recibir notificaciones sobre eventos cercanos. Además, Wegow brinda la opcion de un chat el cual comunica a usuarios con gustos musicales similares. Esta tiene un enfoque bastante diferente al nuestro ya que esta enfocada en la venta de entradas pero tiene características muy similares a las que buscamos en el proyecto. Además al igual que nuestro proyecto presenta un desarrollo multiplataforma siendo la pagina web la siguiente [51]

Figura 4: Vista del inicio

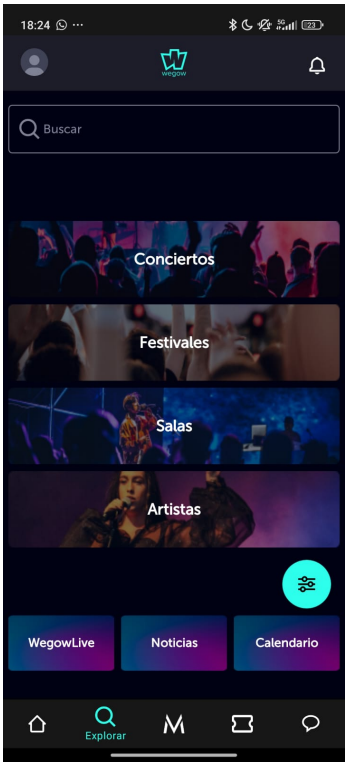


Figura 5: vista de las búsquedas

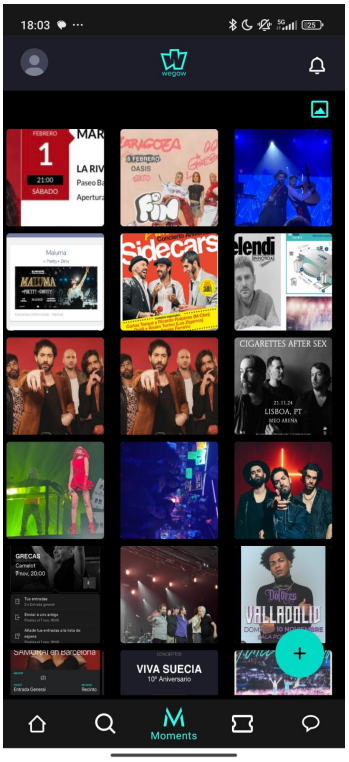


Figura 6: Expositor de los diversos conciertos Moments

Ventajas

- Diseño e implementación muy moderno e intuitivo.
- Muy buena distribución de contenidos.
- Notificaciones personalizadas: La aplicación envía alertas y recordatorios sobre eventos.

Desventajas

- Diseño sobrecargado: La pantalla principal puede sentirse saturada con demasiada información y opciones
- Necesidad de identificación de usuario: Es necesario identificarse para emplearla

1.3.3. Songkick

Songkick es una aplicación móvil que permite a los usuarios descubrir conciertos y festivales en función de sus gustos musicales y ubicación. La app ofrece información detallada sobre cada evento, incluyendo fechas, horarios, ubicación, artistas participantes y opciones para comprar entradas. Esta se caracteriza por su diseño limpio e intuitivo y por su conectividad con plataformas como por ejemplo Spotify. Podemos encontrar esta aplicación a continuación: [41]

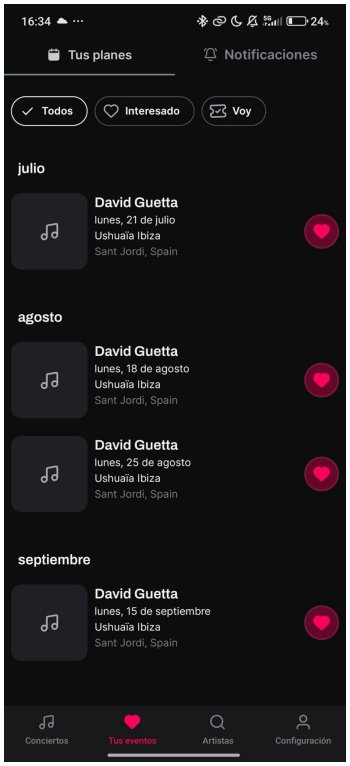


Figura 7: Sección de favoritos de songkick

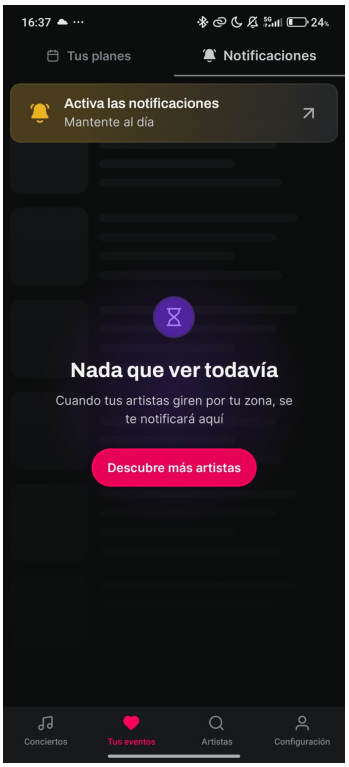


Figura 8: Sección de notificaciones

Ventajas

- Los usuarios reciben notificaciones personalizadas
- Interfaz muy limpia e intuitiva
- Compatible con plataformas musicales

Desventajas

- Algunos eventos o ubicaciones pueden quedar fuera del radar, especialmente en ciudades pequeñas.
- Menor cobertura para músicos emergentes o de nichos específicos.
- En ocasiones, los eventos pueden no estar completamente actualizados o los usuarios reciben avisos tardíos.

1.4. Planteamiento sobre tecnologías a emplear

Antes de empezar a abordar el proyecto debían plantearse el conjunto de herramientas que se iban a emplear para llevar a cabo el desarrollo. Es aquí cuando surge la duda principal de que framework de desarrollo emplear para desarrollar la aplicación móvil habiendo dos principales candidatos, React-Native y flutter, ambos frameworks populares en el desarrollo de aplicaciones móviles multiplataforma.

Por otro lado, la tecnología a emplear para llevar a cabo el desarrollo de la aplicación web estuvo clara desde un comienzo. Se emplearía Python junto con Streamlit, una librería de python

la cual brinda muchas facilidades a la hora de implementar interfaces gráficas.

1.4.1. Comparativa de frameworks de desarrollo frontend para aplicacion de dispositivos moviles

En esta subsección, se comparan dos de los frameworks más populares para el desarrollo de aplicaciones móviles multiplataforma: Flutter y React Native. Ambos poseen características que los hacen destacar, pero también presentan diferencias clave que influirán en la elección del framework según las necesidades del proyecto. Esta comparativa se fundamenta en la información recopilada del foro [26], donde se analizan las ventajas y desventajas de cada tecnología en distintos escenarios de desarrollo. Las características de estos son las siguientes:.

1. Lenguaje de programación:

- **Flutter:** Utiliza **Dart**, un lenguaje optimizado para el desarrollo de aplicaciones móviles de alto rendimiento. Aunque no es tan popular como JavaScript, su integración con Flutter mejora la velocidad y eficiencia de las aplicaciones.
- **React Native:** Emplea **JavaScript**, el lenguaje de programación más utilizado en desarrollo web. Su gran popularidad permite que los desarrolladores web se adapten rápidamente a este framework, además de aprovechar su vasto ecosistema de herramientas.

2. Rendimiento:

- **Flutter:** Ofrece un rendimiento superior al compilar directamente a código nativo, lo que resulta en aplicaciones más rápidas y fluidas. Su diseño permite manejar de manera eficiente animaciones complejas.
- **React Native:** Utiliza un puente entre JavaScript y el código nativo, lo que puede generar algo de sobrecarga en el rendimiento, especialmente con animaciones complejas. A pesar de esto, su rendimiento ha mejorado con las optimizaciones recientes.

3. Ecosistema y comunidad:

- **React Native:** Su comunidad es más grande y madura debido a su existencia desde 2015. Esto se traduce en una mayor cantidad de bibliotecas, tutoriales y soluciones de problemas.
- **Flutter:** Aunque más reciente (lanzado en 2017), Flutter ha crecido rápidamente y continúa expandiéndose, especialmente por el apoyo de Google y su integración con otras herramientas como Firebase.

4. Experiencia de desarrollo:

- **Flutter:** Ofrece una experiencia de desarrollo fluida con su característica de *hot reload*, lo que permite a los desarrolladores ver los cambios en tiempo real sin reiniciar la aplicación. Además, la personalización de la interfaz es altamente consistente en ambas plataformas.
- **React Native:** También cuenta con *hot reload*, pero las diferencias entre iOS y Android pueden generar pequeños desajustes en la UI, lo que podría requerir ajustes adicionales en el diseño.

5. Integración con plataformas nativas:

- **Flutter:** Ofrece una excelente integración con plataformas nativas, aunque puede requerir escribir código adicional cuando se necesitan características específicas de la plataforma.
- **React Native:** Tiene una buena integración con las plataformas nativas y la mayoría de las API de iOS y Android, pero en algunos casos, las funcionalidades avanzadas pueden requerir escribir código nativo en Swift o Java/Kotlin.

6. Noticias de actualidad:
Recientemente, la compañía detrás de Flutter ha llevado a cabo despidos masivos, lo que podría afectar el futuro del framework en términos de soporte y desarrollo. Aunque Flutter sigue siendo una opción robusta, esta situación genera incertidumbre sobre su evolución a largo plazo.

- [2] señala que Google ha despedido a un equipo completo de ingenieros, reemplazándolos por empleados con costos más bajos, lo que podría afectar el desarrollo de Flutter a largo plazo.
- Por otro lado, [16] informa que Google ha recortado equipos de desarrolladores de código abierto, lo que ha generado incertidumbre en la comunidad de Flutter y otras tecnologías mantenidas por la empresa.
- Finalmente, [42] y [40] analizan el impacto de estos despidos en el futuro de Flutter, destacando que esta situación podría beneficiar a otras soluciones como React Native.

7. Conclusión:
Ambos frameworks tienen ventajas y desventajas que dependen del tipo de proyecto y los requisitos específicos. Flutter se destaca en rendimiento y consistencia de la interfaz, mientras que React-Native ofrece una comunidad más grande y un ecosistema bien establecido.

Sin embargo, debido a los recientes despidos en el equipo de Flutter y la incertidumbre sobre su futuro [2, 16, 42, 40], se ha optado por un desarrollo en React-Native. Esta decisión se basa en la estabilidad de su comunidad, el respaldo continuo de Meta y la previsibilidad de su evolución a largo plazo.

1.4.2. Elección del Framework para desarrollo de pagina web Streamlit

Para el desarrollo del sistema de gestión de eventos, se ha seleccionado **Streamlit** como framework principal debido a sus características que se alinean perfectamente con los requisitos funcionales y de información del proyecto. A continuación, se detallan las razones que justifican esta elección:

1. Simplicidad y rapidez:
Streamlit permite desarrollar aplicaciones web completas utilizando únicamente Python, eliminando la necesidad de aprender tecnologías adicionales como HTML, CSS o JavaScript. Esto es clave para implementar funcionalidades como: - Añadir eventos. - Modificar eventos. - Agregar eventos desde un archivo CSV.

2. Interactividad y dinamismo:
El framework proporciona widgets interactivos como botones, formularios y tablas dinámicas, que son esenciales para múltiples tareas, como permitir a los administradores aceptar o rechazar eventos o mostrar listas de eventos con filtros dinámicos basados en atributos del evento.

3. Manejo de datos estructurados:
Streamlit facilita la integración con bibliotecas de Python como Pandas, lo que permite gestionar datos estructurados como fechas y horas o cargar listas de eventos desde archivos CSV.

4. Accesibilidad y despliegue sencillo:
Las aplicaciones desarrolladas con Streamlit son accesibles desde cualquier navegador web, lo que garantiza que tanto administradores como usuarios externos puedan interactuar con el sistema sin necesidad de instalar software adicional.

En resumen, Streamlit es una herramienta ideal para realizar la sección de la pagina web de gestión y revisión de contenidos.

1.4.3. Framework de desarrollo Backend

Para complementar el desarrollo en React Native, se ha optado por emplear **Node.js** como entorno para el backend. Esta elección se basa en los siguientes factores:

- **Compatibilidad con React Native**: Node.js utiliza JavaScript, lo que facilita la integración y la posibilidad de compartir lógica o tipos entre el frontend y el backend.
- **Amplio ecosistema**: Node.js cuenta con una gran cantidad de librerías y herramientas disponibles, así como una comunidad muy activa, lo que simplifica la implementación de funcionalidades avanzadas y el soporte de bases de datos como PostgreSQL,

Gracias a estas características, **Node.js** proporciona una base estable, eficiente y flexible para el backend de la aplicación.

Para la gestión de la base de datos, se ha optado por utilizar DBeaverCE, lo que permite una interacción directa y controlada con el sistema de almacenamiento. Las principales razones para esta elección son:

- **Control total sobre las consultas**: Utilizar el cliente oficial permite escribir consultas SQL personalizadas y optimizadas según las necesidades del proyecto.
- **Ligereza**: Al prescindir de un ORM, se reduce la complejidad y el peso de la aplicación, lo que puede traducirse en un mejor rendimiento en ciertos escenarios.
- **Flexibilidad**: Se puede adaptar la lógica de acceso a datos de manera precisa, sin las restricciones que a veces imponen los ORM.
- **Compatibilidad y soporte**: Los clientes oficiales suelen estar bien mantenidos y documentados, lo que facilita su integración y actualización.

De este modo, la combinación de **Node.js** con el cliente oficial de la base de datos asegura una gestión eficiente y personalizada de los datos, adaptándose a las necesidades concretas del proyecto.

1.4.4. Base de datos empleada

Para la gestión de la base de datos, se ha optado por utilizar **PostgreSQL**, una base de datos relacional de código abierto altamente robusta y escalable. Para interactuar con ella de manera eficiente. Las razones de esta elección son:

- **Alta fiabilidad y rendimiento**: PostgreSQL es conocida por su estabilidad y capacidad de manejo de grandes volúmenes de datos, siendo una de las bases de datos más confiables.
- **Soporte completo para SQL**: Permite el uso de complejas consultas SQL, funciones y transacciones, lo que la hace ideal para aplicaciones empresariales.
- **Escalabilidad y flexibilidad**: PostgreSQL es altamente escalable y flexible, lo que permite manejar una amplia variedad de proyectos, desde pequeños proyectos hasta proyectos con gran carga. 1
- **Comunidad activa y documentación amplia**: PostgreSQL cuenta con una comunidad activa y una amplia documentación online.

1.5. Tecnologías usadas

Para el desarrollo de este proyecto se han realizado dos aplicaciones empleando las tecnologías que se comentarán a continuación:

1.5.1. Python

Debido a que el desarrollo de nuestro proyecto ha estado dividido en dos principales secciones, desarrollo de aplicación web y desarrollo de aplicación móvil, se ha debido abordar que tecnologías emplear para realizar cada una de las tareas.

En este caso, y buscando la simplicidad, para trabajar con el backend de la aplicación web y tratar con la lógica de esta, se ha optado por emplea Python [37], un lenguaje ampliamente utilizado en el ámbito académico y con una documentación abundante.

Figura 9: Logo de Python

1.5.2. Streamlit

Por otro lado, para desarrollar la interfaz de usuario de la aplicación web se ha optado por utilizar *Streamlit* [45], una librería de Python que proporciona las herramientas necesarias para construir aplicaciones web de forma sencilla y estructurada.

Esta elección se debe a que Streamlit facilita considerablemente la creación de formularios simples, la visualización de datos en tablas y la integración de diversos elementos interactivos. Además, su diseño modular permite dividir el código de forma clara, lo que contribuye a una mejor organización y mantenimiento del proyecto.

Figura 10: Logo de streamlit

1.5.3. React Native

Para el desarrollo de la aplicación móvil se ha optado por emplear *React Native*, un framework de código abierto basado en JavaScript que permite crear aplicaciones móviles nativas para Android e iOS a partir de un único código base.

Esta tecnología resulta ideal para implementar las funcionalidades de visualización y navegación de eventos que se detallarán más adelante, permitiendo además una experiencia de usuario fluida y consistente en varias plataformas.

Respecto al por que de la selección de este framework se abordará posteriormente en la sección siguiente: 1.4.1

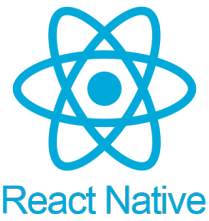


Figura 11: Logo de React-Native

1.5.4. Node.js

La sección del backend correspondiente a la aplicación móvil se ha optado por llevar a cabo el desarrollo utilizando *Node.js*, un entorno de ejecución para JavaScript que permite crear aplicaciones del lado del servidor de forma eficiente y escalable.

Gracias a su arquitectura basada en eventos y su modelo de operaciones no bloqueantes, Node.js facilita la gestión de múltiples peticiones concurrentes, lo cual resulta especialmente útil para aplicaciones móviles que requieren respuestas rápidas y en tiempo real.

Figura 12: Logo de Node.js

1.5.5. postgreSQL

Para la gestión de la base de datos se ha empleado la plataforma **postgreSQL**, un sistema de gestión de bases de datos relacional que ofrece una estructura sólida, robusta y altamente escalable para el almacenamiento de datos estructurados.

Esta tecnología fue seleccionada además de por sus capacidades técnicas, porque ya había sido utilizada previamente en el entorno académico de la universidad, lo cual facilitó su adopción e integración en el proyecto.

Figura 13: Logo de postgreSQL

1.5.6. Docker

Para facilitar la exportación, despliegue y puesta en marcha de la aplicación en distintos entornos, se ha utilizado **Docker** como herramienta de empaquetado y contenedorización.

Cada una de las tres secciones del proyecto —base de datos, aplicación web y aplicación móvil— ha sido configurada dentro de su propio contenedor Docker. Esta separación modular permite desplegar fácilmente el sistema en otros dispositivos, garantizando la portabilidad y simplicidad en la configuración del entorno de ejecución.

Figura 14: Logo de docker

1.6. Objetivos

El objetivo principal de este trabajo es desarrollar aplicaciones complementarias al sitio web *Watios y Decibelios* que faciliten a cualquier usuario la visualización de eventos próximos en Valladolid

desde dispositivos móviles, permitiendo a los usuarios compartir eventos y seguir sus conciertos favoritos.

Adicionalmente, se desarrollará una aplicación web para optimizar la gestión de eventos por parte del administrador y permitir a usuarios externos proponer conciertos de interés.

Los objetivos específicos son:

- Diseñar y desarrollar una aplicación móvil para visualización de información de eventos
- Diseñar y desarrollar una aplicación web para gestión integral de eventos
- Habilitar a usuarios externos (es decir, que no son administradores) para creación de dichos eventos
- Mejorar la experiencia del usuario mediante:
  - Funciones para compartir contenido
  - Herramientas de seguimiento de eventos
  - Funciones avanzadas de búsqueda
- Facilitar las tareas de gestión de los datos:
  - Permitir a los usuarios externos proponer eventos
  - Habilitar gestión básica de eventos (propios y propuestos por otros)
  - Controlar el número de usuarios con acceso a la sección de gestión

1.7. Metodología

Para el desarrollo de esta práctica, he decidido emplear la **metodología iterativa e incremental** [1] debido a su flexibilidad y capacidad de adaptación a las características específicas del proyecto. Esta metodología permite abordar el desarrollo de la aplicación en ciclos que combinan planificación, diseño, implementación y pruebas, generando incrementos funcionales en cada iteración. Esto resulta especialmente útil en un entorno como el de este proyecto, donde los requisitos pueden evolucionar con el tiempo y las funcionalidades se pueden ajustar para mejorar la experiencia del usuario.

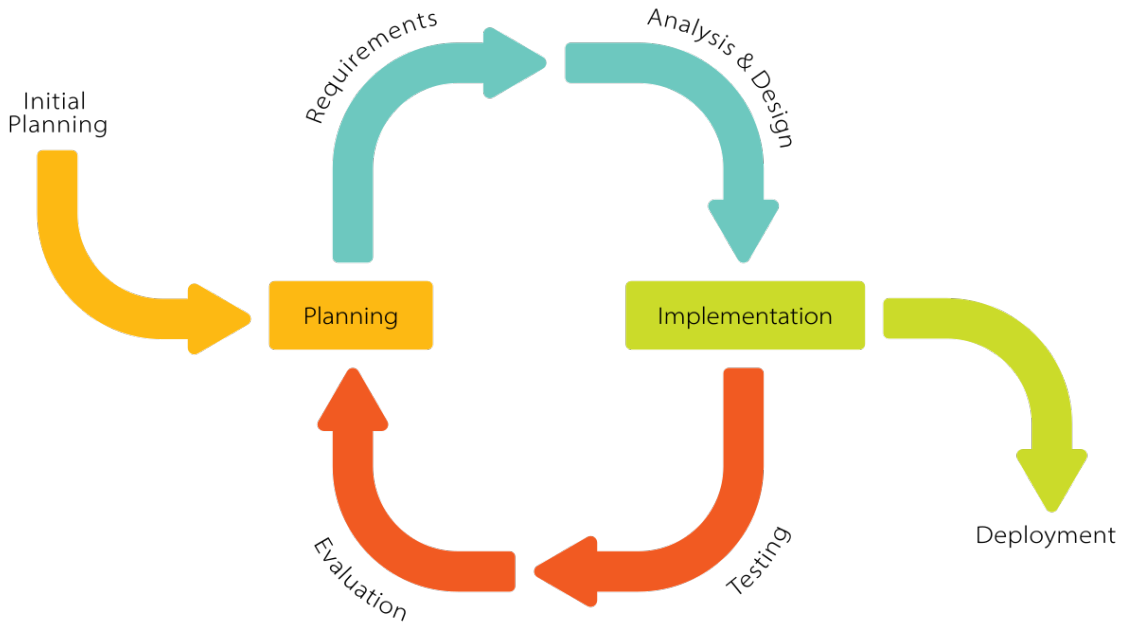


Figura 15: Representación de fases dentro de metodología iterativa e incremental [13]

1.8. Estructura de la memoria

- **Introducción**: Presentación del contexto y objetivos del proyecto, incluyendo la motivación que ha llevado a su desarrollo y la relevancia de la solución propuesta.
- **Planificación**: Detalle del cronograma y las estrategias a seguir durante el desarrollo.
- **Análisis**: Planteamiento previo al desarrollo de la aplicación en el cual se han estudiado las necesidad y posibles formas de abarcar el proyecto futuro.
- **Diseño**:Sección en la que se detallará en profundidad organización del proyecto.
- **Descripción de las fases**: Descripción de cada ciclo de desarrollo dentro de la metodología iterativa e incremental, explicando cómo se abordaron las funcionalidades y los avances realizados.
- **Pruebas**: Sección en la que se detalla el conjunto de pruebas llevadas a cabo sobre el proyecto.
- **Conclusiones**: Expone las conclusiones obtenidas tras realizar el desarrollo del proyecto.
- **Apéndices**: En este apartado se incluyen materiales complementarios que respaldan la información presentada en la sección principal del documento.
- **Bibliografía**: Listado de las fuentes y referencias utilizadas durante el desarrollo del proyecto, desde artículos académicos hasta documentación técnica de las tecnologías empleadas.

Capítulo 2

2. Planificación

2.1. Planificación Inicial

En esta sección se detallará la planificación inicial del proyecto, abarcando tanto los aspectos temporales como las principales fases de trabajo a seguir y los principales hitos de estas.

Las tareas se organizarán en iteraciones, siguiendo una **metodología iterativa e incremental**, para asegurar que el desarrollo sea progresivo, flexible y adaptado a posibles modificaciones y mejoras durante el proceso.

La planificación del proyecto se ha basado en las recomendaciones y buenas prácticas que se encuentran en las guías de Project Management Institute [36] y en el enfoque práctico para proyectos de sistemas de información propuesto por Cadle y Yeates [7]. Estas referencias han servido como apoyo para estructurar y organizar las fases y tareas del proyecto, adaptándolas a las necesidades específicas del desarrollo.

2.1.1. Organización temporal

En esta sección se presenta la distribución temporal del proyecto, dividido en cinco etapas principales. Cada etapa tiene objetivos específicos que se desarrollarán de manera progresiva, desde la definición inicial hasta las pruebas finales, garantizando el cumplimiento de los plazos y requisitos establecidos.

El desarrollo de este proyecto abarcará aproximadamente 340 horas, las cuales se distribuirán principalmente a lo largo del segundo cuatrimestre. Sin embargo, cabe destacar que la primera fase del proyecto ya ha sido parcialmente abordada durante el primer cuatrimestre, lo que permite una mejor organización de las fases posteriores.
La carga de trabajo semanal se organizará destinando 4 horas diarias durante 5 días a la semana, con un enfoque que permita alcanzar los objetivos establecidos en cada etapa del desarrollo.

- **Planificación**: En esta se establecerán las bases del proyecto definiendo el alcance con las partes interesadas, seleccionando las herramientas y el framework de desarrollo, y realizando un análisis de los principales requisitos. Además, se desarrollará el modelo de dominio y los casos de uso iniciales, asegurando una representación clara de los conceptos clave y sus relaciones. Estas bases se presentarán a las partes interesadas para asegurar su aprobación y entendimiento. El desarrollo la planificación se detallará en profundidad en el apartado 5.1. Se estima que esta tomará aproximadamente **80 horas de trabajo** divididas entre el primer y segundo cuatrimestre finalizando a mediados de Febrero aproximadamente **4 semanas de trabajo aproximadamente**.
- **Primera iteración**: Esta comenzara sobre el 18 de Marzo y tendrá una duración aproximada de **5 semanas** y estará enfocada en el desarrollo inicial de las funcionalidades principales de la aplicación. Durante este período, se implementarán las funcionalidades de registro de usuarios, identificación y cierre de sesión. Además, se desarrollarán las funciones esenciales de la aplicación, como añadir eventos y visualizar los eventos disponibles. El proceso de desarrollo llevado a cabo durante la segunda iteración se explica con detalle en el apartado 5.2.

- **Segunda iteración**: Esta iteración, con una duración aproximada de **3 semanas** comenzará el 22 de abril y consistirá en el desarrollo de las funcionalidades de prioridad media del sistema. Los avances y modificaciones realizados en esta segunda iteración se describen en profundidad en el apartado 5.3.
- **Tercera iteración**: Esta iteración, con una duración aproximada de **3 semanas** comenzará el 13 de mayo y estará enfocada en la implementación de las resto de las funcionalidades estipuladas para completar el desarrollo del sistema. Los avances y modificaciones realizados en la tercera iteración se describen en profundidad en el apartado 5.4.
- **Cuarta iteración**: La cuarta y última iteración tendrá una duración aproximada de **2 semanas**, se llevara a cabo entre el 2 y el 16 de Junio y estará dedicada a la realización de las pruebas finales de usuario. En esta etapa, se evaluará que el sistema cumpla con los objetivos principales establecidos, asegurando su funcionalidad, usabilidad y conformidad con los requisitos. Tras la validación de las pruebas, se realizarán los ajustes necesarios y se procederá a la finalización del proyecto, dejando la aplicación lista para su entrega. Para conocer el desarrollo y resultados obtenidos en la cuarta iteración, se puede consultar el apartado 5.5.

Respecto a la estimación el numero de horas dedicadas en cada una de las semanas ha sido distinto. En la primera iteración, debido a una asignatura pendiente el tiempo dedicado al desarrollo del TFG ha sido menor y mas irregular, por lo que con semanas se hace referencia al tiempo que se hubiese dedicado en caso de realizar un desarrollo constante de horas.
Se estima que semanalmente se dedicaran 20 horas al trabajo por lo que el desarrollo estimado serian $\frac{300}{20} = 15$ semanas. Debido a posibles variaciones e impedimentos en el desarrollo de estos se han sumado 40 horas adicionales, contando así con aproximadamente 17 semanas de desarrollo.
En la figura 16 se muestra la gráfica de barras que representa las semanas de desarrollo de este proyecto

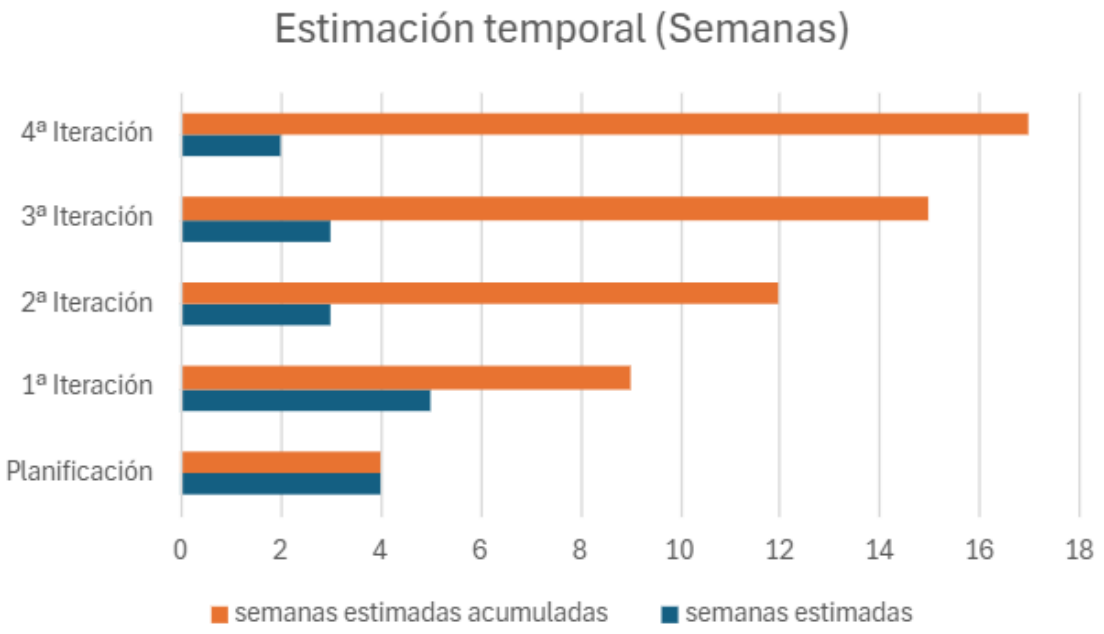


Figura 16: Gráfica de barras de estimación inicial

Respecto a la distribución en horas por cada iteración ha sido la que se muestra en la figura 17

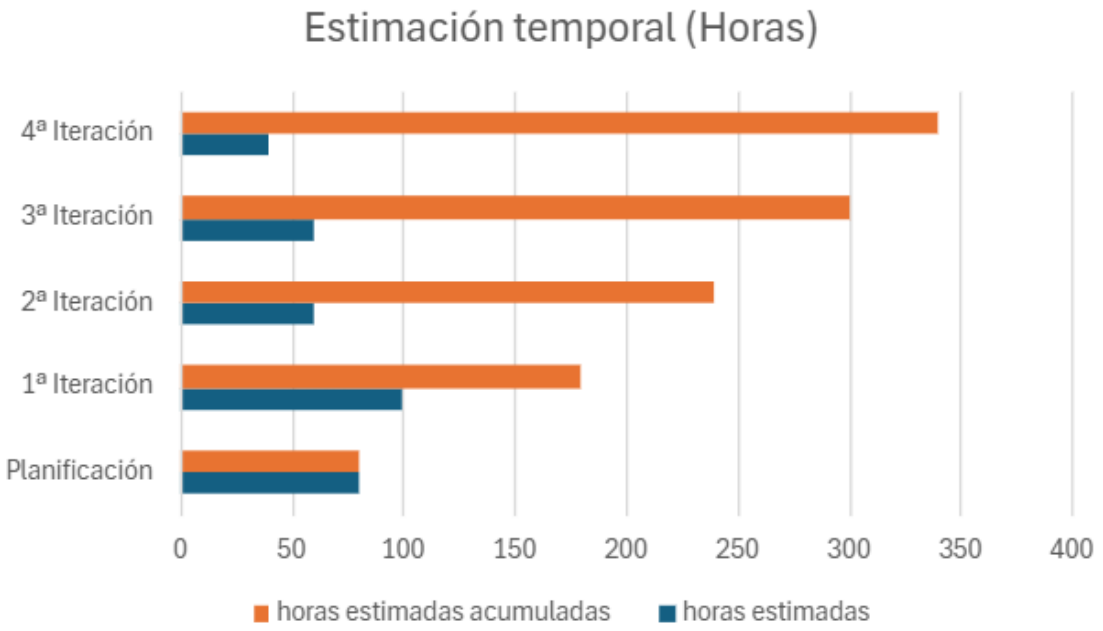


Figura 17: Gráfica de barras de estimación de horas

2.1.2. Diagrama de Gantt

A continuación, se presenta en la figura 18 la planificación inicial del proyecto, representada mediante un diagrama de Gantt [5]. Este se estructura en cinco fases diferenciadas, cada una identificada con un color distinto para facilitar su comprensión.

En la parte inferior del diagrama se indican también las revisiones periódicas realizadas con los tutores de la asignatura, las cuales han servido para hacer un seguimiento y control del desarrollo del proyecto.

Se ha procurado seguir las indicaciones y buenas prácticas expuestas en Kerzner [24], quien ofrece una guía detallada para el desarrollo de diagramas Gantt.

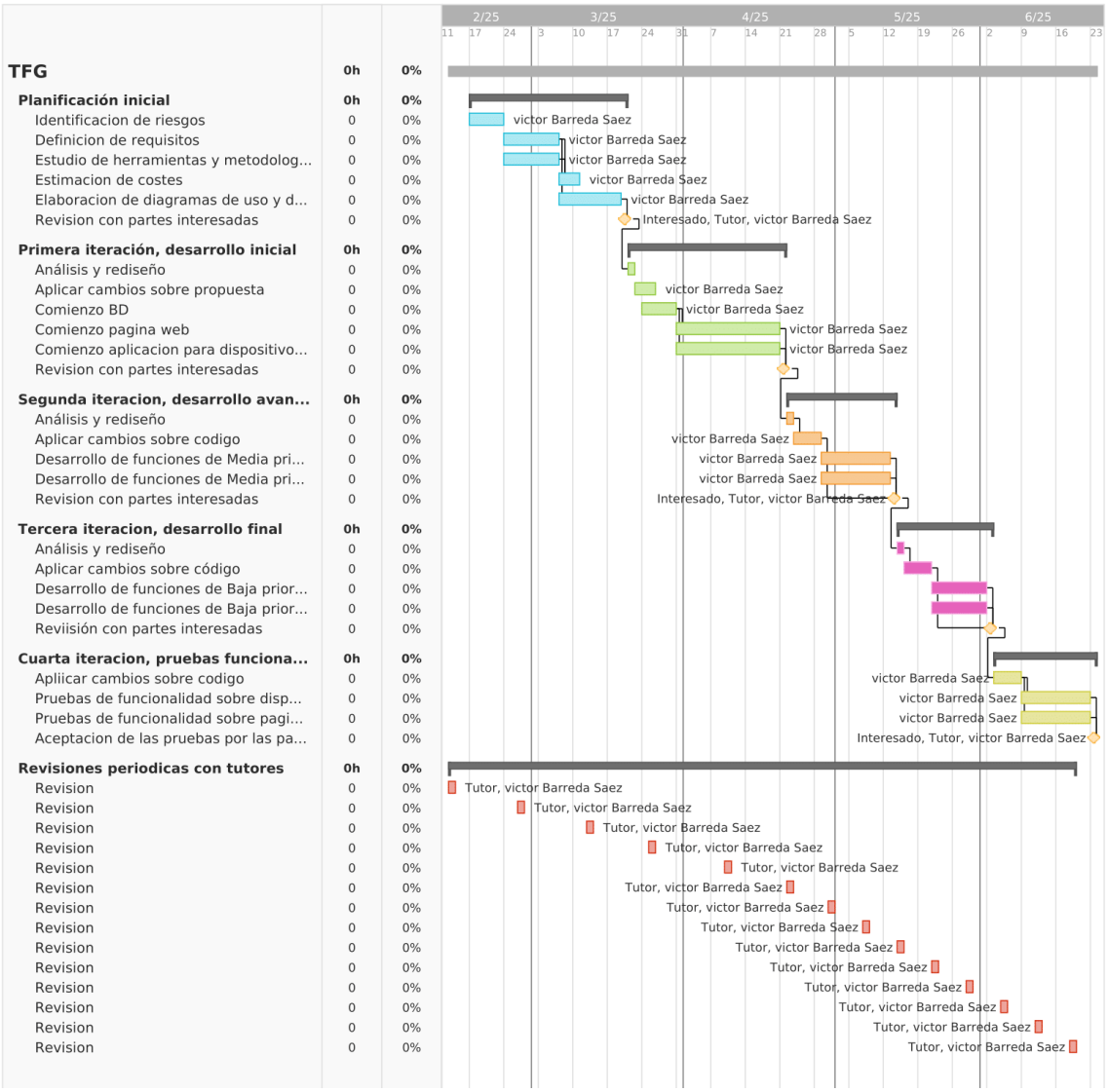


Figura 18: Diagrama en el que se muestran las fases de desarrollo del proyecto junto a los hitos.

2.1.3. Análisis de riesgos

Realizar un análisis de los principales riesgos en las primeras etapas de un proyecto permite identificar, evaluar y gestionar posibles problemas o imprevistos que podrían impactar negativamente en su desarrollo. Este análisis temprano ayuda a prevenir obstáculos que podrían retrasar el avance del proyecto o aumentar los costos.

Para realizar el siguiente análisis de riesgos nos hemos ceñido a las metodologías y recomendaciones propuestas por Hall [20], que ofrecen un enfoque estructurado para identificar, evaluar y mitigar riesgos durante las diferentes fases del proyecto, asegurando una gestión proactiva y efectiva frente a posibles contingencias.

Además, permite al equipo anticiparse a situaciones complejas, proporcionando estrategias de mitigación y planes de contingencia que faciliten una respuesta eficiente ante cualquier eventualidad.

- **ID**: Identificador único para cada riesgo.

- **Riesgo**: Nombre con el que nos referimos al riesgo.
- **Descripción**: Explicación breve del riesgo identificado.
- **Probabilidad**: Posibilidad de que el riesgo ocurra.
- **Impacto**: Efecto del riesgo en el proyecto si el riesgo se materializa.
- **Riesgo total**: Combinación de probabilidad e impacto del riesgo.
- **Acciones de mitigación**: Estrategias para reducir la probabilidad o el impacto del riesgo.
- **Acciones correctivas**: Medidas tomadas después de que el riesgo se materializa.

Para la elaboración de la tabla de riesgos, se han seguido las directrices y recomendaciones indicadas en el artículo disponible en el sitio web del INCIBE: [23]

ID	Nombre	Descrip.	Proba.	Impacto	Riesgo T.
1	Problema de red	Posibles fallos en la conectividad de red	MEDIA	ALTO	MEDIO
2	Alcance mal definido	No tener claridad en el alcance del TFG puede llevar a desviaciones, hacer más trabajo del necesario o no cubrir aspectos esenciales.	BAJA	ALTO	MEDIO
3	Comunicación no fluida con el tutor	El tutor asignado esta implicado en el desarrollo del TFG y no puede dedicar el tiempo necesario en exclusiva al nuestro	BAJA	ALTO	MEDIO
4	Dificultades con las fuentes	Puede haber dificultades para acceder a las fuentes de información o datos necesarios	MEDIA	MEDIO	MEDIO
5	Asignatura suspensa	Suspender una asignatura que impida la presentación del TFG	BAJA	ALTO	MEDIO
6	Bibliografía errónea	Los datos a los que ha accedido el usuario y en los que esta basando su trabajo son incorrecto	BAJA	MEDIO	BAJO
7	Imposibilidad de trabajo en periodo planificado para ello	Existe el riesgo de que, debido a causas de fuerza mayor, el alumno no pueda trabajar en el TFG durante el tiempo previsto, lo cual podría afectar a la calidad del proyecto o retrasar su entrega.	BAJA	ALTO	MEDIO
8	Problemas técnicos de software	Dificultades con el entorno de desarrollo, como errores inesperados en herramientas o librerías, que pueden retrasar el desarrollo o afectar la calidad del trabajo	MEDIA	ALTO	ALTO

Cuadro 1: Tabla de Análisis de Riesgos

ID	Nombre	Acciones de mitigación	Acciones correctivas
1	Problema de red	Configurar redes alternativas, como datos móviles o conexiones públicas seguras	Usar otro lugar con conexión estable para trabajar
2	Alcance mal definido	Revisiones constantes con los guías para definir bien el alcance y hacer las rectificaciones necesarias	Acortar el alcance
3	Comunicación no fluida con el tutor	Planificar correctamente las horas de tutoría y definir los canales de comunicación pronto	Desarrollar el TFG con ayuda de otro tutor
4	Dificultades con las fuentes	Cambiar de fuentes	Acceder a fuentes contrastadas
5	Asignatura suspensa	Solicitar ayuda al profesor responsable de la asignatura si se presenta alguna dificultad	Aplazar la presentación del TFG y presentarse a convocatoria extraordinaria el primer cuatrimestre del siguiente curso
6	Bibliografía errónea	Consultar con el tutor la veracidad de las fuentes	Buscar una nueva fuente de conocimientos
7	Imposibilidad de trabajo en periodo planificado para ello	Realizar un planning de trabajo del TFG que comience con suficiente antelación como para poder soportar un periodo de tiempo en el que no se pueda trabajar	Re-planificar el TFG contando con el tiempo de trabajo perdido
8	Problemas técnicos de software	Mantener backups regulares y usar herramientas alternativas disponibles	Cambiar a otro software que cumpla los requerimientos del proyecto

Cuadro 2: Plan de acción para los riesgos identificados

2.2. Entorno de trabajo

En esta sección se detalla el entorno de trabajo empleado durante el desarrollo de la aplicación. Se describe tanto la configuración utilizada como los dispositivos principales que han formado parte del proceso.

Para el desarrollo de este proyecto se han empleado un **ordenador de sobremesa** con las especificaciones detalladas a continuación (ver tabla 3) y un dispositivo móvil **Xiaomi Redmi Note 13 5G** [52] (ver tabla 4).

Característica	Especificación
Sistema Operativo	Windows 10
Arquitectura del sistema	64 bits
Procesador	Intel Core i5-8400
Memoria	16 GiB RAM
Almacenamiento	1 TB NVMe 3.0

Cuadro 3: Especificaciones del equipo de sobremesa.

Característica	Especificación
Sistema Operativo	MIUI basado en Android
Pantalla	6,67”(2400 x 1080)
Procesador	MediaTek Dimensity 6080
Memoria RAM	8 GB
Almacenamiento	128 GB
Conectividad	5G, Wi-Fi, Bluetooth

Cuadro 4: Especificaciones del Xiaomi Redmi Note 13 5G.

2.3. Herramientas empleadas

Durante el desarrollo del proyecto, se han empleado diversas herramientas de software que han facilitado la implementación, documentación y comunicación. A continuación, se detallan las principales herramientas utilizadas:

- **Microsoft Teams**: Utilizado como plataforma principal de comunicación y coordinación. [28]
- **Visual Studio Code**: Editor de código fuente empleado para el desarrollo de la aplicación aplicación web con Python y Streamlit. [29]
- **SQL**: Lenguaje utilizado para la gestión y manipulación de bases de datos. [33]
- **Overleaf**: Plataforma de edición de documentos en LaTeX, utilizada para la redacción y estructuración del trabajo. [34]
- **Visual Paradigm**: Herramienta utilizada para la creación de diagramas UML y el modelado del sistema. [49]
- **Figma** : Herramienta empleada para el desarrollo de los mockups presentados a las partes interesadas. [17]
- **DBeaverCE** : Herramienta empleada para el desarrollo desarrollo, gestión y visualizacion de la base de datos. [11]
- **Android Studio** : Herramienta empleada para editar el código fuente de la aplicación móvil la cual brinda ciertas facilidades para correr pruebas sobre dispositivos simulados (Empleada para la parte relacionada con React-Native y Node.js) [25]
- **TeamGantt** Herramienta empleada para la realización de los diagramas Gantt empleados para planificar la duración de las iteraciones y los hitos de estas [46]
- **Microsoft Excel** Herramienta empleada para realizar los diagramas de barras con el fin de planificar las horas por iteración y monitorizar las horas reales dedicadas a cada una de estas. [14]
- **python**: lenguaje programacional empleado para tratar con la lógica de la aplicación de gestión del proyecto.[37]
- **Streamlit**: Herramienta de código abierto que permite crear aplicaciones web interactivas de forma sencilla en Python. [45]
- **Microsoft Forms**: Herramienta empleada para el desarrollo de los cuestionarios que se presentarían a los usuarios [30]
- **GitHub Copilot**: Asistente de programación basado en inteligencia artificial, integrado en Visual Studio Code, utilizado para acelerar el desarrollo del código y proporcionar sugerencias inteligentes durante la implementación. [19]

- **ChatGPT**: Modelo de lenguaje basado en inteligencia artificial desarrollado por OpenAI, utilizado como herramienta de apoyo para generación de contenido y revisión técnica durante el desarrollo del proyecto. [32]
- **GitLab UVa**: Plataforma de gestión y alojamiento de repositorios Git utilizada en la Universidad de Valladolid para facilitar la colaboración y control de versiones en proyectos de desarrollo de software [48].
- **Docker**: Tecnología de contenedores que permite empaquetar aplicaciones y sus dependencias en entornos aislados y portables, facilitando la implementación y escalabilidad [15].
- **PostgreSQL**: Sistema de gestión de bases de datos relacional de código abierto reconocido por su robustez, rendimiento y soporte para características avanzadas [35].
- **Astah Professional** - Herramienta profesional de modelado UML y diagramado de software desarrollada por Change Vision, utilizada principalmente para el desarrollo de los diagramas de paquetes [8].
- **React Native** - Framework de desarrollo móvil multiplataforma creado por Meta que permite construir aplicaciones nativas usando JavaScript y React [27].

2.4. Estimación de costes

En esta sección se detalla el cálculo aproximado del coste total asociado al desarrollo del proyecto. Este análisis incluye los recursos técnicos y humanos empleados, así como otros factores relevantes como licencias, herramientas de software, y dispositivos utilizados.

2.4.1. Costes de Recursos Humanos

El desarrollo de la aplicación ha sido llevado a cabo por un único desarrollador, el estudiante, quien ha dedicado un número significativo de horas al proyecto. Para estimar el coste asociado a la mano de obra, se considera una tarifa estándar de mercado para un desarrollador junior, multiplicada por el total de horas invertidas en el desarrollo.

El cálculo del coste de la mano de obra se obtiene mediante la siguiente fórmula:

$$C_{\text{mano de obra}} = H_{\text{total}} \times T_{\text{hora}} \tag{1}$$

donde:

- $C_{\text{mano de obra}}$ es el coste total de la mano de obra.
- H_{total} representa el número total de horas trabajadas.
- T_{hora} es la tarifa estimada por hora de trabajo.

Si el desarrollo se ha realizado en varios períodos de tiempo, el cálculo se puede expresar como:

$$C_{\text{mano de obra}} = \sum_{i=1}^{n} (H_i \times T) \tag{2}$$

donde H_i es el número de horas trabajadas en el período i, y n es el número total de períodos considerados.

Para este proyecto, si se ha estimado un total de 300 horas de trabajo con una tarifa de 15€/h, el coste de la mano de obra sería:

$$C_{\text{mano de obra}} = 300 \times 15 = 4{,}500€ \tag{3}$$

2.4.2. Costes de Software y licencias

Respecto a las tecnologías empleadas, algunas cuentan con versiones gratuitas y otras con planes de pago. En la mayoría de los casos, las funcionalidades necesarias para el desarrollo del proyecto han podido cubrirse utilizando versiones gratuitas. No obstante, existen herramientas cuyo uso profesional o completo conlleva un coste asociado. Las únicas aplicaciones que implican un gasto, ya sea directo o proyectado en un entorno profesional, son las siguientes:

- **Visual Paradigm**: Dispone de una versión gratuita con funcionalidades limitadas. Para uso profesional se requiere una suscripción. La edición estándar tiene un coste aproximado de 6€ al mes (cuando se factura anualmente). Esta herramienta esta implementada en el plan de estudios de la UVa, por lo tanto no ha incurrido ningún gasto adicional real. [49]
- **Microsoft 365 (incluyendo Excel, Teams y Forms)**: Estas herramientas están incluidas en las suscripciones de Microsoft 365. El plan Microsoft 365 Apps for Business tiene un coste aproximado de 8,80€ al mes por usuario.Al igual que Visual Paradigm, esta herramienta no ha supuesto ningún coste adicional, ya que la universidad proporcionó una cuenta de Microsoft 365. [14, 28]
- **GitHub Copilot**: Herramienta de asistencia de programación basada en IA. Requiere una suscripción de pago, con un precio actual de 10 $ al mes (aproximadamente 9,30€ mensuales) para usuarios individuales. [19]

El gasto total derivado del costo de Softwares y licencias asciende aproximadamente a **24,1 €/mes**. En este caso el proceso de desarrollo se estima que durara 3,5 meses por lo que el gasto ascenderá a **84,35€** Para calcular este se han tenido en cuenta las suscripciones mas baratas a las tecnologías anteriormente comentadas.

2.4.3. Costes de Hardware

El desarrollo del proyecto ha requerido el uso de diferentes dispositivos tanto para la programación como para las pruebas y validación de la aplicación. Estos equipos han sido fundamentales en el proceso, por lo que se ha considerado su impacto en los costes a través de la estimación de su depreciación proporcional al tiempo de uso.

Para este trabajo, se ha utilizado el ordenador de sobremesa del alumno, cuyo detalle de especificaciones se presenta en la tabla 3. Además, para realizar pruebas en un entorno real, se ha empleado un **Xiaomi Redmi Note 13 5G**, cuyas características se encuentran en la tabla 4. El enlace de compra de este dispositivo se puede consultar en [52].

La vida útil del **Xiaomi Redmi Note 13 5G** se estima en **48 meses**, con un coste aproximado de **180€**. En cuanto al ordenador de sobremesa, su vida útil se encuentra entre **4 y 6 años**, con un precio estimado de **1000€**, al que se suman **300€** correspondientes a los periféricos (monitor, teclado y ratón). Estos datos se utilizarán para calcular la depreciación anual de los dispositivos empleados en el proyecto.

Para determinar el impacto económico del hardware utilizado, se aplicará el método de depreciación lineal anual:

$$\text{Depreciación anual} = \frac{\text{Costo total del equipo}}{\text{Vida útil en años}} \tag{4}$$

El desarrollo del proyecto tiene una duración estimada de **4 meses**, por lo que la depreciación se ajustará proporcionalmente al tiempo de uso.

El **Xiaomi Redmi Note 13 5G** tiene un coste de **180€** y una vida útil de **4 años**. Aplicando la fórmula de depreciación, obtenemos:

$$D_{\text{Redmi Note 13 5G}} = \frac{180}{4} = 45€ \text{ anuales} \tag{5}$$

Por otro lado, el equipo de sobremesa, con un coste de **1000€**, más **300€** de periféricos, tiene un valor total de **1300€**. Considerando una vida útil media de **5 años**, la depreciación anual es:

$$D_{\text{ordenador}} = \frac{1300}{5} = 260€ \text{ anuales} \tag{6}$$

El coste total anual de la depreciación del hardware utilizado es entonces:

$$D_{\text{total}} = 45 + 260 = 305€ \text{ anuales} \tag{7}$$

Dado que el proyecto tiene una duración de **4 meses**, la depreciación proporcional para este período se calcula como:

$$D_{\text{proyecto}} = D_{\text{total}} \times \frac{4}{12} = 305 \times \frac{4}{12} = 101{,}67€ \tag{8}$$

Por lo tanto, el coste total de depreciación del hardware durante el desarrollo del proyecto es de **101.67€**.

2.4.4. Costes de Infraestructura

El costo mensual del servicio de hospedaje, que incluye internet y todos los servicios asociados, asciende aproximadamente a **250 euros mensuales**. Considerando que el desarrollo de la aplicación se extendió durante **17 semanas**, lo que equivale a aproximadamente **3.92 meses** (utilizando la conversión estándar de 4.33 semanas por mes), el costo total de infraestructura para el proyecto se calcula de la siguiente manera:

$$\text{Costo total} = \text{Costo mensual} \times \text{Duración en meses} \tag{9}$$

$$\text{Costo total} = 250 \text{ euros/mes} \times 3{,}92 \text{ meses} = 980{,}77 \text{ euros} \tag{10}$$

Por tanto, el **costo total de infraestructura** para el desarrollo completo de la aplicación fue de aproximadamente **981 euros** durante todo el ciclo de vida del proyecto.

2.4.5. Calculo final de la estimación de costes

Tras calcular las principales fuentes de gasto del proyecto podemos determinar que la estimación de costes de la aplicación sería aproximadamente 4.500€ del costo de mano de obra + 101,67€ de la depreciación del hardware + 980,77€ de gastos derivados del hospedaje e infraestructura además de 84.35€ referentes a los gastos software del proyecto, con lo cual el precio final del proyecto sería:

$$\text{Costo Total} = 4{,}500 + 101{,}67 + 980{,}77 + 84{,}35 = 5{,}666{,}75 \tag{11}$$

Por tanto, el costo total estimado para el desarrollo completo de la aplicación asciende a **5.582,44€**

2.5. Comparativa entre duración planificada y duración final

En esta sección se detalla la desviación que ha sufrido el proyecto tras haberse realizado en comparación con la planificación inicial.

Se comentarán las modificaciones realizadas a lo largo del desarrollo, las cuales surgieron en las distintas iteraciones del proyecto. Además, se analizarán las diferencias entre las horas y semanas estimadas inicialmente y las realmente empleadas, destacando las razones detrás de cualquier desviación y el impacto que estas modificaciones han tenido en el desarrollo del proyecto.

2.5.1. Fase de Planificación

Durante la fase de planificación inicial se estableció la estructura fundamental del proyecto y se definieron los objetivos principales a alcanzar. En esta etapa se realizó un diseño inicial de la arquitectura que tendría el proyecto, identificando los requisitos funcionales y no funcionales que debía cumplir la aplicación. Se acordaron los periodos aproximados para realizar cada una de las tareas posteriores y se estableció la forma de dividir estas en función de su prioridad. Se implementó una metodología iterativa incremental, lo que implicaba que cada iteración debía producir una "porción funcional"de la aplicación final, permitiendo la entrega progresiva de funcionalidades operativas y facilitando la identificación temprana de problemas durante el desarrollo.

Los tiempos establecidos en esta planificación se cumplieron satisfactoriamente y se alcanzaron todos los hitos propuestos sin desviaciones significativas. La única diferencia respecto a la planificación inicial fue que el cierre de cada iteración o Milestone no sería una entrevista con el interesado externo, ya que no hubo opción de plantear este mecanismo de retroalimentación. Esta modificación no afectó sustancialmente el desarrollo posterior del proyecto, aunque sí eliminó una fuente valiosa de feedback continuo.

2.5.2. Primera Iteración

Esta Iteración dio comienzo tras establecer las bases para el desarrollo futuro de la aplicación y esta se centró en el desarrollo de los componentes fundamentales del sistema, comenzando con el diseño de la base de datos. Se empleó PostgreSQL como sistema de gestión de base de datos, plataforma gratuita utilizada en el ámbito universitario, y se utilizó la herramienta DBeaver.CE como interfaz de gestión de la base de datos. Durante esta fase se desarrolló la funcionalidad básica de conexión entre ambas aplicaciones, tanto la aplicación web como la móvil, con la base de datos.

En esta iteración se dio especial importancia a una tabla denominada .events", orientada específicamente a la gestión de eventos, constituyendo esta el núcleo central del sistema de información. Paralelamente se desarrolló un endpoint en el backend implementado con Node.js, estableciendo así la comunicación básica entre el frontend y la base de datos. Todo el desarrollo de esta primera iteración se realizó dentro del periodo presupuestado inicialmente y no presentó complicaciones técnicas significativas que alteraran la planificación establecida.

2.5.3. Segunda Iteración

La segunda iteración comenzó con una reunión entre el estudiante y ambos profesores supervisores del proyecto. En este encuentro se identificaron cambios estructurales necesarios que originarían modificaciones sustanciales en la estructura base del proyecto, particularmente en el diseño de la base de datos. El principal cambio consistió en que la tabla .evento"pasaba a dividirse en varias tablas especializadas: çoncierto", "giraz "festival", lo que requirió una reestructuración completa del modelo de datos.

Esta reestructuración retrasó considerablemente el desarrollo, ya que fue necesario realizar cambios en múltiples componentes del sistema. Se tuvieron que crear varios endpoints adicionales para dar soporte a la nueva estructura de datos, además de modificar los existentes para adaptarlos al nuevo modelo. Sumado a esto, surgieron problemas específicos en la aplicación desarrollada con React Native, lo que contribuyó a generar un retraso estimado de una semana respecto a la planificación original. Adicionalmente, hubo que remodelar la estructura completa de la aplicación web para adaptarla a los nuevos requerimientos, lo que .ayudó"significativamente al retraso experimentado en esta iteración.

2.5.4. Tercera Iteración

La tercera iteración inició con el retraso acumulado de una semana proveniente de la iteración anterior, tras una reunión con los profesores comentando el desarrollo de la aplicación y posibles

variaciones. Durante este periodo surgieron complicaciones adicionales en el desarrollo de los modales destinados a la visualización de eventos. La principal dificultad se centraba en determinar la forma óptima de presentar los datos de conciertos y giras, planteándose dudas sobre la mejor aproximación para el display de esta información.

Estas dificultades estuvieron parcialmente relacionadas con la inexperiencia en algunas de las tecnologías empleadas, lo que ralentizó el proceso de toma de decisiones y implementación. Los problemas técnicos encontrados en esta fase resultaron más complejos de resolver de lo inicialmente previsto, derivando en una pérdida adicional de aproximadamente una semana respecto a la planificación original. Esta acumulación de retrasos comenzó a tener un impacto significativo en la planificación global del proyecto, requiriendo ajustes en las iteraciones posteriores.

2.5.5. Cuarta Iteración

Para cubrir el tiempo necesario y solventar los problemas acumulados, se destinó parte de la cuarta iteración a arreglar los errores y completar funcionalidades pendientes de las iteraciones anteriores. Esta iteración estaba destinada íntegramente a la realización de pruebas del sistema, pero debido a los retrasos acumulados fue necesario posponer dos semanas el inicio de la fase de testing para poder completar el desarrollo básico de la aplicación.

Para evitar una desviación aún mayor de la planificación original, se tomó la decisión estratégica de descartar la implementación de algunas funcionalidades secundarias. Entre las funcionalidades descartadas se encontraban la representación de eventos en forma de calendario, que había sido contemplada en los mockups iniciales, y la realización de una sección de compartir eventos muy detallada. Se optó por minimizar significativamente el alcance de las pruebas, concentrándose únicamente en la validación de las funcionalidades básicas del sistema.

Esta reducción en la fase de pruebas originó que el feedback de los usuarios de test llegase de forma tardía, imposibilitando su incorporación durante el ciclo de desarrollo actual. Como consecuencia, todas las propuestas provenientes de los usuarios de prueba fueron catalogadas como trabajo futuro, pendientes de implementación en fases posteriores del desarrollo del proyecto.

A continuación se muestra en las siguientes figuras la distribución final de semanas 19 y de horas 20, en las que podemos observar que ha habido desviación en las iteraciones 2 y 3 del proyecto.

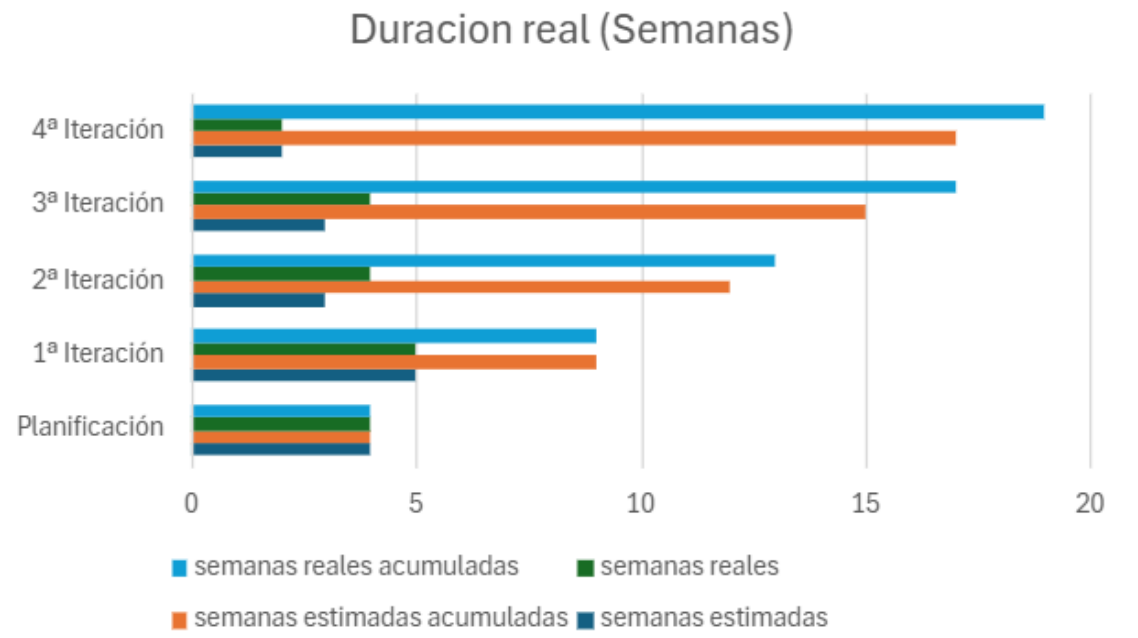


Figura 19: Gráfica de barras de estimación inicial

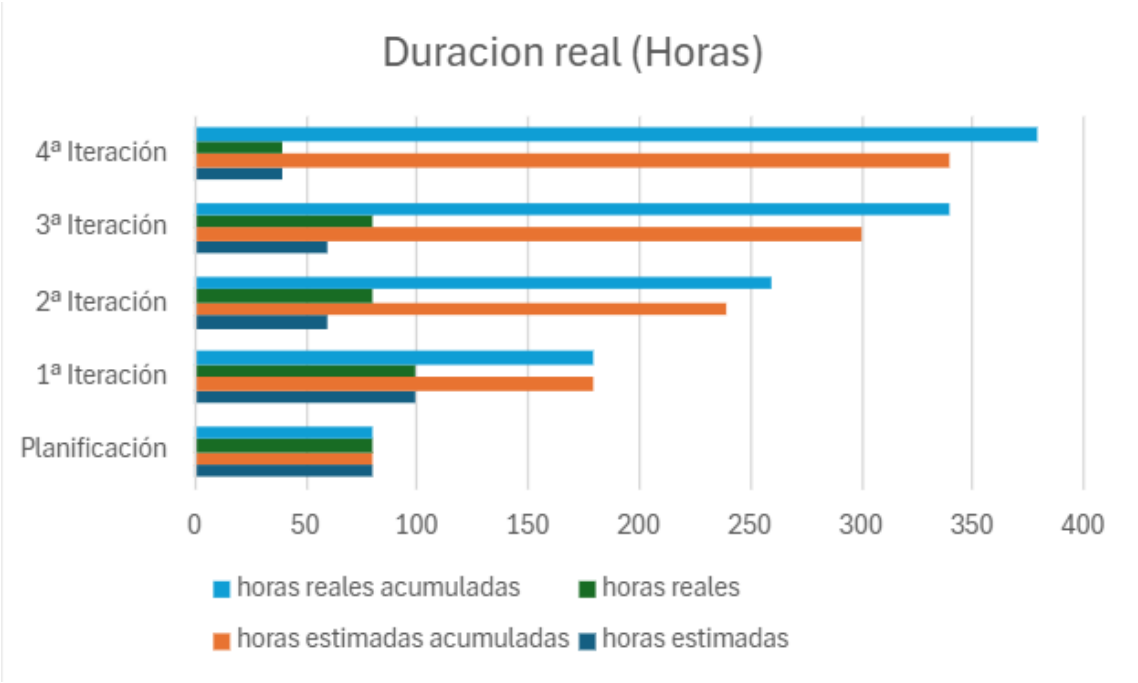


Figura 20: Gráfica de barras de duración real en horas de proyecto

En la siguiente tabla 5 podemos observar la comparación temporal entre las horas estimadas y las horas reales acumuladas a lo largo de las distintas iteraciones del proyecto.

Estimación temporal (Horas)	**Horas estimadas**	**Horas estimadas acumuladas**	**Horas reales**	**Horas reales acumuladas**
Planificación	80	80	80	80
1ª Iteración	100	180	100	180
2ª Iteración	60	240	80	260
3ª Iteración	60	300	80	340
4ª Iteración	40	340	40	380

Cuadro 5: Comparación entre horas estimadas y reales por iteración

En esta podemos observar que ha habido una pequeña desviación en las horas, originada en los tiempos de las iteraciones 2 y 3.

Capítulo 3

3. Análisis

El análisis de software es una de las fases más importantes en el desarrollo de un sistema, ya que establece las bases sobre las cuales se construirá todo el proyecto. A través de un análisis detallado y preciso, se identifican y documentan las necesidades del usuario, los requisitos del sistema y las restricciones que deben tenerse en cuenta.

Un análisis adecuado permite identificar y planificar correctamente los requisitos del sistema, lo que resulta en un desarrollo más eficiente, con menos correcciones y ajustes durante las fases posteriores.

3.1. Tipos de usuario

Antes de proceder con el análisis, es importante definir los términos utilizados para referirse a los distintos tipos de usuarios o roles implicados en el desarrollo de esta práctica.

A lo largo del documento, se distinguirán los siguientes tres perfiles:

- **Administrador**: Usuario con rol permisos superiores al resto de los usuarios, orientada para el desarrollador anterior de Watios y Decibelios. Este tiene permiso total sobre las principales funcionalidades sobre el proyecto de gestión
- **Externo**: Orientado a usuarios interesados en proponer sus conciertos en la página web de gestión, estos tienen menos permisos a la hora de gestionar eventos, y la mayoría de sus acciones requieren supervisión por parte de un administrador.
- **Visitante**: Usuario de aplicación de visualización móvil. No requiere loguin ni registro. Este puede disfrutar del contenido sobre conciertos próximos, agregar a favoritos ...

3.2. Requisitos

En los siguientes apartados se definirán los requisitos de la aplicación. Estos se dividen en **requisitos funcionales**, que describen las acciones y funciones específicas que el sistema debe ser capaz de realizar, los **requisitos no funcionales**, los cuales se refieren a las características del sistema que no están directamente relacionadas con las funcionalidades y por ultimo los **requisitos de información**, que establecen cómo debe manejarse la información dentro del sistema, incluyendo la gestión de datos, la privacidad y el cumplimiento de normativas legales.

3.3. Requisitos funcionales

A continuación se describen los requisitos funcionales del proyecto. Estos definen las funciones o acciones que el sistema debe realizar. Además son esenciales para que el software cumpla con las expectativas del usuario, especificando qué debe hacer el sistema, como permitir el registro de usuarios, procesar pagos o mostrar información en una interfaz específica.

Código	Nombre	Descripción	Prioridad
RF-01	Ver eventos disponibles	El sistema debe permitir a los usuarios ver los eventos disponibles y el contenido de estos	Alta
RF-02	Crear cuenta	El sistema debe permitir a los administradores crear cuentas/perfiles	Alta
RF-03	Identificación de usuarios	El sistema debe permitir al usuario identificarse en la aplicación. Al identificarse este quedara identificado como .Administrador.o como .Externo"	Alta
RF-04	Cerrar sesión	El sistema debe permitir a los usuarios identificados cerrar sesión	Alta
RF-05	Filtrar eventos	El sistema debe permitir al usuario filtrar los eventos en función de diversos filtros.	Media
RF-06	Añadir evento a favoritos	El sistema permitirá al usuario añadir eventos de interés a su lista de favoritos y recibir notificaciones de actualizaciones	Media
RF-07	Eliminar eventos de favoritos	El sistema permitirá al usuario eliminar eventos de su lista de favoritos	Media
RF-08	Compartir en redes	El sistema debe permitir al usuario compartir eventos en sus redes sociales.	Baja
RF-09	Ver calendario de favoritos	El sistema debe permitir al usuario acceder al calendario de eventos a los que sigue.	Media
RF-10	Proponer eventos	El sistema debe permitir a los usuarios .externos"proponer eventos.	Alta
RF-11	Modificar eventos propuestos	El sistema debe permitir a los usuarios .externos"modificar propuestas anteriores.	Media

Cuadro 6: Requisitos Funcionales del Sistema, Parte 1

Código	Nombre	Descripción	Prioridad
RF-12	Añadir eventos	El sistema debe permitir a los usuarios .ªdministradores.ªñadir eventos.	Alta
RF-13	Aceptar eventos.	El sistema debe permitir a los usuarios .ªdministradores.ªceptar eventos.	Alta
RF-14	Rechazar eventos	El sistema debe permitir a los usuarios .ªdministradoresrechazar eventos .	Alta
RF-15	Modificar eventos	El sistema debe permitir a los usuarios .ªdministradores"modificar eventos.	Media
RF-16	Agregar eventos desde archivo CSV	El sistema debe permitir a los usuarios .ªdministradoresz .externos.ªgregar una lista de eventos a través de un archivo .csv .	Baja
RF-17	Modificar Perfil	El sistema debe permitir a los usuarios .ªdministradoresz .externos"modificar las credenciales de acceso.	Alta.
RF-18	Envio de notificaciones	El sistema debe permitir el envió de notificaciones sobre los eventos en los que este interesado el usuario	BAJA

Cuadro 7: Requisitos Funcionales del Sistema , Parte 2

3.4. Requisitos no funcionales

Estos establecen las condiciones bajo las cuales el sistema debe operar, sin involucrar directamente su funcionalidad. Abarcan aspectos como el rendimiento, la seguridad, la confiabilidad y la usabilidad del sistema, asegurando que no solo se cumpla con las funciones, sino que el software sea eficiente, seguro y fácil de usar.

Código	Requisito No Funcional
RNF-01	El sistema debe ser compatible con las ultimas versiones Android, habiendo sido este probado en Android en sus versiones recientes (Android 15).
RNF-02	El sistema debe responder a las consultas del servidor con un tiempo de respuesta bajo en condiciones normales de carga.
RNF-03	El sistema debe garantizar un uso eficiente de los recursos del dispositivo, evitando un consumo excesivo de batería y memoria.
RNF-04	El sistema debe permitir el acceso a la sección de edición desde navegadores y a la sección de visualización del contenido desde dispositivos móviles
RNF-05	El sistema debe cumplir con las normativas de privacidad de datos, como el RGPD o la CCPA, según sea aplicable.
RNF-06	El sistema debe minimizar el almacenamiento de datos personales, recolectando únicamente los necesarios para las funcionalidades ofrecidas.
RNF-07	El sistema debe solicitar y registrar el consentimiento explícito de los usuarios para el tratamiento de sus datos personales.
RNF-08	El sistema debe incorporar mecanismos de autenticación y autorización seguros, garantizando el acceso solo a usuarios autorizados.

Cuadro 8: Requisitos No Funcionales del Sistema

3.5. Requisitos de información

Respecto a los requisitos de información, estos se refieren a la gestión y el manejo de los datos dentro del sistema. Estos requisitos incluyen consideraciones sobre la privacidad de los datos, la integridad y el almacenamiento.

Código	Requisito de Información
RI-0	El sistema debe guardar los siguientes campos de los usuarios registrados: correo electrónico único, nombre de usuario único, contraseña para autenticación y un indicador de permisos especiales (por ejemplo, administrador).
RI-02	El sistema debe almacenar los siguientes campos relacionados con los eventos: nombre del evento, lugar donde se realizará, fecha y hora del evento, imagen promocional opcional, precio opcional, enlace para compra de entradas opcional, enlace con información adicional sobre el grupo o artista opcional, información extra opcional, estado del evento (Disponible, Propuesto, Cancelado o Pospuesto) y tipo de evento (Concierto, Gira o Festival).
RI-03	El sistema debe manejar datos estructurados para representar fechas y horas, incluyendo día, mes y año para las fechas, así como hora, minutos y segundos para las horas.
RI-04	El sistema debe definir valores preestablecidos para ciertos atributos: el estado del evento (Disponible, Propuesto, Cancelado o Pospuesto) y el tipo de evento (Concierto, Gira o Festival).

Cuadro 9: Requisitos de Información del Sistema

3.6. Casos de uso

A continuación se definirán los casos de uso del sistema, los cuales detallan las interacciones entre los usuarios y la aplicación.
Los casos de uso constituyen una herramienta fundamental para modelar cómo interactúan los usuarios con un sistema, permitiendo identificar los actores involucrados, las acciones que realizan y los resultados esperados. Según Arlow y Neustadt, este enfoque proporciona una visión clara de

las funcionalidades del sistema desde la perspectiva del usuario [4].

Para mostrar los casos de uso se han diseñado dos diagramas diferenciados para reflejar de forma más clara las funcionalidades específicas de cada plataforma.

En la figura 21 se muestra el diagrama de uso de la página web, mientras que en la figura 22 representa los casos de uso correspondientes a la aplicación móvil . Esta separación permite visualizar de manera más precisa las interacciones particulares de los distintos usuarios con cada interfaz del sistema.

Figura 21: Diagrama de casos de uso: Página web

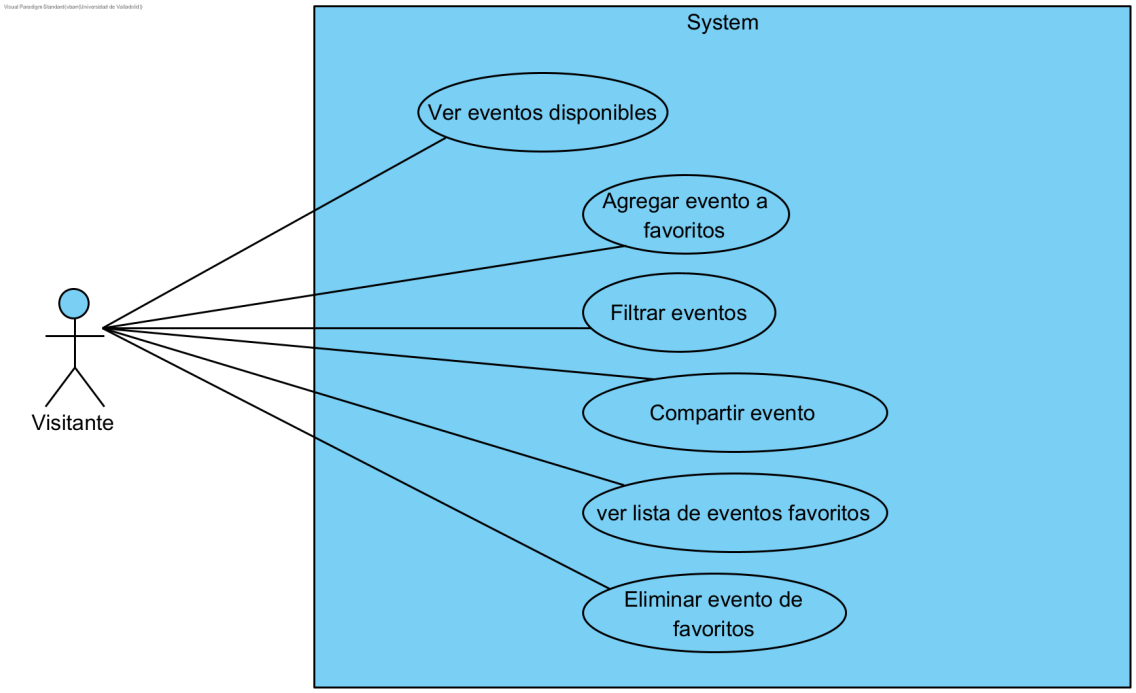


Figura 22: Diagrama de casos de uso: Aplicación móvil

3.7. Descripción de los casos de uso de aplicación Web

A continuación, se presentará una descripción detallada de los pasos involucrados en cada uno de los casos de uso correspondientes a la aplicación web. Estos pasos muestran cómo los distintos actores interactúan con el sistema en diversos escenarios, así como las respuestas que el sistema proporciona ante dichas interacciones.

3.7.1. Crear cuenta

CU-1	**Crear cuenta**
Precondiciones	El usuario debe estar identificado como Administrador.
Actores principales	Administrador.
Flujo principal	1. El Administrador solicita crear una cuenta. 2. El sistema presenta un formulario para introducir las credenciales del nuevo usuario. 3. El Administrador completa los campos requeridos. 4. El Administrador confirma la creación de la cuenta. 5. El sistema valida los datos, crea la cuenta y muestra un mensaje de confirmación.
Flujos alternativos	3.a. Si el Administrador no completa todos los campos obligatorios, el sistema muestra un mensaje indicando los campos faltantes. 5.a. Si los datos no son válidos el sistema muestra un mensaje de error correspondiente.
Postcondiciones	La cuenta del nuevo usuario queda creada y lista para ser utilizada.

Cuadro 10: Descripción del caso de uso - Creación de cuenta por el Administrador

3.7.2. Identificación

CU-2	**Identificarse**
Precondiciones	El usuario debe estar registrado en el sistema.
Actores principales	Administrador y Externo.
Flujo principal	1. El usuario solicita identificarse. 2. El sistema solicita las credenciales del usuario. 3. El usuario proporciona las credenciales. 4. El sistema verifica las credenciales en la base de datos. 5. El sistema autentica al usuario. 6. El sistema notifica al usuario el resultado de la autenticación y redirige al usuario a la página principal.
Flujos alternativos	1.a. Si las credenciales son incorrectas, el sistema notifica al usuario con un mensaje de error.
Postcondiciones	El usuario queda identificado como administrador o como externo.

Cuadro 11: Descripción del caso de uso - Identificarse

3.7.3. Cierre de sesión

CU-3	**Cierre de sesión**
Precondiciones	El usuario debe haber iniciado sesión previamente en el sistema.
Actores principales	Administrador y Externo
Flujo principal	1. El usuario solicita cerrar sesión en el sistema. 2. El sistema invalida la sesión activa del usuario. 3. El sistema confirma al usuario que la sesión se ha cerrado correctamente.
Flujos alternativos	1.a. Si el usuario no tiene una sesión activa, el sistema notifica que no es posible realizar la acción.
Postcondiciones	El usuario queda sin una sesión activa en el sistema.

Cuadro 12: Descripción del caso de uso - Cierre de sesión

3.7.4. Proponer evento

CU-4	Proponer un evento
Precondiciones	El usuario debe estar autenticado.
Actores principales	Externo.
Flujo principal	1. El usuario solicita realizar una propuesta de evento. 2. El sistema permite al usuario ingresar los detalles necesarios para la propuesta. 3. El usuario proporciona los datos requeridos sobre el evento y confirma el envío de la propuesta. 4. El sistema verifica que los datos ingresados sean correctos y completos. 5. Si la verificación es exitosa, el sistema registra la propuesta y la envía para revisión de un administrador.
Flujos alternativos	4.a. Si algún campo obligatorio no está completo o contiene errores, el sistema notifica al usuario indicando los detalles a corregir.
Postcondiciones	El evento queda registrado como una propuesta pendiente de aprobación por parte de un administrador.

Cuadro 13: Descripción del caso de uso - Proponer un evento

3.7.5. Modificar evento propuesto

CU-5	Modificar un evento propuesto
Precondiciones	El usuario debe estar autenticado como administrador y haber propuesto algun evento.
Actores principales	Externo.
Flujo principal	1. El usuario accede a la sección para revisar los eventos que había propuesto con anterioridad. 2. El sistema muestra los datos de los eventos que había propuesto anteriormente. 3. El usuario edita cualquiera de los campos del evento . 4. El sistema valida los datos proporcionados. 5. Si la validación es exitosa, el sistema envía una petición de cambio que posteriormente debera ser aceptada.
Flujos alternativos	5.a. Si algún dato es incorrecto o está incompleto, el sistema notifica al usuario los ajustes necesarios.
Postcondiciones	El evento queda propuesto para cambios, a la espera de que un administrador acepte los cambios realizados.

Cuadro 14: Descripción del caso de uso - Modificar evento propuesto

3.7.6. Añadir evento

CU-6	Añadir un evento
Precondiciones	El usuario debe estar autenticado como administrador.
Actores principales	Administrador.
Flujo principal	1. El usuario accede a la sección para añadir un nuevo evento. 2. El sistema solicita los datos necesarios para registrar el evento. 3. El usuario proporciona los datos del evento y confirma el registro . 4. El sistema valida los datos proporcionados. 5. Si la validación es exitosa, el sistema publica el evento, haciéndolo disponible para los usuarios.
Flujos alternativos	5.a. Si algún dato es incorrecto o está incompleto, el sistema notifica al usuario los ajustes necesarios.
Postcondiciones	El evento queda registrado y publicado en el sistema, disponible para los usuarios finales.

Cuadro 15: Descripción del caso de uso - Añadir un evento

3.7.7. Aceptar evento

CU-7	Aceptar evento
Precondiciones	El usuario debe estar autenticado y debe haber propuestas de eventos pendientes de revisión.
Actores principales	Administrador.
Flujo principal	1. El usuario accede a la sección para revisar las propuestas de eventos. 2. El sistema presenta una lista de propuestas realizadas por los usuarios externos. 3. El usuario selecciona un evento propuesto para aceptar. 4. El sistema ofrece dos opciones: 4.1. Aceptar directamente el evento desde la lista. 4.1.2 El sistema valida los datos del evento y lo publica, haciéndolo visible para los usuarios. 4.2. Desplegar los detalles del evento antes de aceptarlo. 4.2.1. El sistema muestra los detalles del evento. 4.2.2. El usuario confirma la aceptación del evento. 4.2.3. El sistema valida los datos del evento, lo publica y y notifica al Externo que la propuso sobre la aceptación.
Flujos alternativos	4.a. El usuario decide no aceptar el evento. 4.1.a. El sistema regresa a la lista de propuestas sin realizar cambios.
Postcondiciones	El evento aceptado queda registrado y publicado en el sistema, disponible para los usuarios finales.

Cuadro 16: Descripción del caso de uso - Aceptar evento

3.7.8. Rechazar evento

CU-8	Rechazar evento
Precondiciones	El usuario debe estar autenticado y debe haber propuestas de eventos pendientes de revisión.
Actores principales	Administrador.
Flujo principal	1. El usuario accede a la sección para revisar las propuestas de eventos. 2. El sistema presenta una lista de propuestas realizadas por los usuarios externos. 3. El usuario selecciona un evento propuesto para rechazar. 4. El sistema ofrece dos opciones: 4.1. Rechazar directamente el evento desde la lista. 4.1.2. El sistema elimina el evento de la lista de propuestas y notifica a los usuarios interesados sobre la cancelación. 4.2. Desplegar los detalles del evento antes de rechazarlo. 4.2.1. El sistema muestra los detalles del evento. 4.2.2. El usuario confirma el rechazo del evento. 4.2.3. El sistema elimina el evento de la lista de propuestas y notifica al Externo que la propuso sobre el rechazo.
Flujos alternativos	4.a. El usuario decide no rechazar el evento. 4.1.a. El sistema regresa a la lista de propuestas sin realizar cambios.
Postcondiciones	El evento queda registrado como Rechazadoz ya no está disponible en el sistema para los usuarios finales.

Cuadro 17: Descripción del caso de uso - Rechazar evento

3.7.9. Modificar evento

CU-9	Modificar evento
Precondiciones	El usuario debe estar autenticado y debe existir al menos un evento registrado en el sistema.
Actores principales	Administrador.
Flujo principal	1. El usuario accede a la sección para gestionar los eventos. 2. El sistema presenta una lista de eventos registrados. 3. El usuario selecciona el evento que desea modificar. 4. El sistema muestra los detalles actuales del evento. 5. El usuario actualiza la información del evento. 6. El usuario confirma los cambios. 7. El sistema valida la información proporcionada. 8. Si la validación es exitosa, el sistema actualiza el evento en el sistema y notifica a los usuarios interesados.
Flujos alternativos	7.a. Si algún dato es incorrecto, el sistema muestra un mensaje de error con los campos a corregir.
Postcondiciones	El evento queda actualizado en el sistema y los usuarios interesados son notificados.

Cuadro 18: Descripción del caso de uso - Modificar evento

3.7.10. Agregar eventos desde archivo CSV

CU-10	Agregar eventos desde archivo CSV
Precondiciones	El usuario debe estar autenticado como administrador o externo y debe estar trabajando desde la pagina web.
Actores principales	Administrador o externo.
Flujo principal	1. El usuario accede a la sección para añadir un nuevo evento. 2. El sistema solicita los datos necesarios para registrar el evento. 3. El usuario introduce el fichero CSV con los eventos que quiere introducir. 4. El sistema valida los datos proporcionados. 5. Si la validación es exitosa, el sistema publica la lista de eventos, haciéndolos disponibles para los demás usuarios.
Flujos alternativos	7.a. Si algún dato es incorrecto, el sistema muestra un mensaje de error.
Postcondiciones	La lista de eventos del CSV queda registrada en el sistema y visible para todos los usuarios.

Cuadro 19: Agregar eventos desde archivo CSV

3.7.11. Modificar cuenta

CU-11	Modificar cuenta
Precondiciones	El usuario debe estar loggueado.
Actores principales	Administrador.
Flujo principal	1. El usuario accede a la sección de modificacion de perfil. 2. El sistema solicita la nueva password y le pide que la repita 3. El usuario introduce ambos campos y envía el formulario. 4. El sistema valida los datos proporcionados. 5. Si la validación es exitosa, el sistema cambia las credenciales de la cuenta.
Flujos alternativos	7.a. Si la nueva contraseña es incorrecta o contiene caracteres indevidos, el sistema muestra un mensaje de error.
Postcondiciones	La cuenta pasa a tener otra contraseña, la que acabamos de establecer.

Cuadro 20: Modificar cuenta

3.8. Descripción de los casos de uso de aplicación Móvil

A continuación, se presentará una descripción detallada de los pasos involucrados en cada uno de los casos de uso correspondientes a la aplicación móvil. En esta sección se explicará cómo los distintos usuarios interactúan con la app en distintos contextos de uso, así como las respuestas del sistema ante dichas interacciones.

3.8.1. Ver eventos disponibles

CU-1	**Ver eventos disponibles**
Precondiciones	Ninguna
Actores principales	Usuario
Flujo principal	1. El usuario solicita visualizar los eventos disponibles. 2. El sistema muestra al usuario una lista de eventos disponibles con sus detalles.
Flujos alternativos	1.a. Si no hay eventos disponibles, el sistema notifica al usuario con un mensaje: ”No hay eventos disponibles en este momento”.
Postcondiciones	Ninguna.

Cuadro 21: Descripción del caso de uso - Ver eventos disponibles

3.8.2. Filtrar búsquedas

CU-2	**Filtrar eventos**
Precondiciones	El usuario debe estar en la vista de búsqueda.
Actores principales	Usuario
Flujo principal	1. El usuario accede a la vista de búsqueda. 2. El sistema muestra los filtros disponibles . 3. El usuario selecciona un filtro. 4. El usuario introduce el criterio de búsqueda deseado (como un nombre de artista o una fecha específica). 5. El sistema procesa los filtros y criterios de búsqueda. 6. El sistema muestra los resultados que coinciden con los filtros seleccionados.
Flujos alternativos	6.a Si no hay resultados que coincidan con los filtros seleccionados, el sistema muestra un mensaje indicando que no se encontraron eventos y sugiere modificar los filtros.
Postcondiciones	Ninguna.

Cuadro 22: Descripción del caso de uso - Filtrar eventos

3.8.3. Añadir evento a favoritos

CU-3	**Añadir evento a lista de favoritos**
Precondiciones	El evento seleccionado no esta en favoritos.
Actores principales	Usuario
Flujo principal	1. El usuario identifica el evento que desea agregar a favoritos. 2. El usuario elige agregar a favoritos el evento mediante alguna de las siguientes opciones: 2.1. Desde una lista general de eventos. 2.2. Desde la vista detallada de la información del evento. 3. El sistema registra la acción del usuario y actualiza su lista personal de eventos favoritos. 4. El sistema confirma que el evento ha sido añadido correctamente a la lista de eventos favoritos.
Flujos alternativos	Ninguno.
Postcondiciones	El evento se registra en la lista de eventos favoritos del usuario, accesible desde su cuenta personal.

Cuadro 23: Descripción del caso de uso - Seguir un evento

3.8.4. Eliminar evento de lista de favoritos

CU-4	**Eliminar evento de lista de favoritos**
Precondiciones	Ya existen eventos que figuran en la lista de favoritos.
Actores principales	Usuario
Flujo principal	1. El usuario identifica el evento que desea eliminar de la lista de favoritos. 2. El usuario elige sacar de la lista de favoritos el evento mediante alguna de las siguientes opciones: 2.1. Desde una lista general de eventos. 2.2. Desde la vista detallada de la información del evento. 3. El sistema registra la acción del usuario y actualiza su lista personal de eventos favoritos. 4. El sistema confirma que el evento ha sido eliminado correctamente de la lista de eventos favoritos.
Flujos alternativos	Ninguno.
Postcondiciones	El evento se registra en la lista de eventos favoritos del usuario, accesible desde su cuenta personal.

Cuadro 24: Descripción del caso de uso - Seguir un evento

3.8.5. Compartir evento

CU-5	Compartir evento
Precondiciones	El usuario debe estar en la vista de un evento específico.
Actores principales	Usuario
Flujo principal	1. El usuario selecciona la opción de compartir. 2. El sistema muestra las opciones de redes sociales disponibles. 3. El usuario selecciona una opción, como WhatsApp. 4. El sistema genera un enlace al evento y abre la aplicación seleccionada con el enlace cargado. 5. El usuario envía el enlace desde la aplicación externa.
Flujos alternativos	4.a. El usuario cancela antes de seleccionar una aplicación, . 4.1.a. El sistema cierra el menú sin realizar ninguna acción
Postcondiciones	Ninguna.

Cuadro 25: Descripción del caso de uso - Compartir evento por redes sociales

3.8.6. Acceder a eventos favoritos

CU-6	Revisar calendario de eventos
Precondiciones	Ninguna.
Actores principales	Usuario
Flujo principal	1. El usuario accede a la vista de calendario de eventos favoritos. 2. El sistema muestra un calendario con los eventos marcados como favoritos. 3. El usuario puede navegar por las fechas para visualizar los eventos correspondientes. 4. El sistema ofrece detalles básicos del evento.
Flujos alternativos	2.a. Si no hay eventos marcados como favoritos, el sistema muestra un mensaje indicando que no hay eventos para mostrar.
Postcondiciones	Ninguna.

Cuadro 26: Descripción del caso de uso - Revisar calendario de eventos

3.9. Modelo de dominio

A continuación, teniendo ya definidos los casos de uso y los requisitos funcionales, se presenta el modelo de dominio actualizado de la aplicación. Este modelo describe las entidades principales del sistema y sus relaciones, proporcionando una base conceptual sólida para el diseño e implementación de la solución.

El sistema está orientado a la gestión de eventos musicales como conciertos, giras y festivales, así como a la administración de usuarios con distintos niveles de acceso. Entre los principales elementos representados en el modelo de dominio se incluyen:

- **Usuario y Roles**: La clase *Usuario* contiene información de autenticación como correo, nombre de usuario y contraseña, y un atributo booleano de permisos. Los usuarios se especializan en dos roles definidos: *Admin*, *Externo* . Los administradores pueden gestionar completamente los eventos, mientras que los usuarios externos pueden proponer eventos. Por otro lado, Los visitantes no registrados tienen acceso limitado al sistema pudiendo solo emplear la aplicación movil.

- **Evento**: Es una clase abstracta que representa un evento genérico. A partir de ella se derivan tres subtipos: *Concierto*, *Gira* y *Festival*. Todos ellos incluyen atributos como cartel, nombre, información adicional y precios (en el caso de conciertos y festivales). Los conciertos incluyen además atributos específicos como lugar, fecha, hora y enlaces de interés.

- **Fecha y Hora**: Se definen como tipos de datos personalizados (*DataType* y *Primitive*) que encapsulan la información necesaria para representar la fecha y hora de los eventos.

- **Estado**: Es un tipo enumerado que define el estado en el que puede encontrarse un evento. Los valores posibles son: *Disponible*, *Propuesto*, *Cancelado* y *Pospuesto*.

- **Ubicación**: La información geográfica se modela mediante las clases *Provincia* y *Población*, vinculadas a los eventos para indicar el lugar donde se llevarán a cabo.

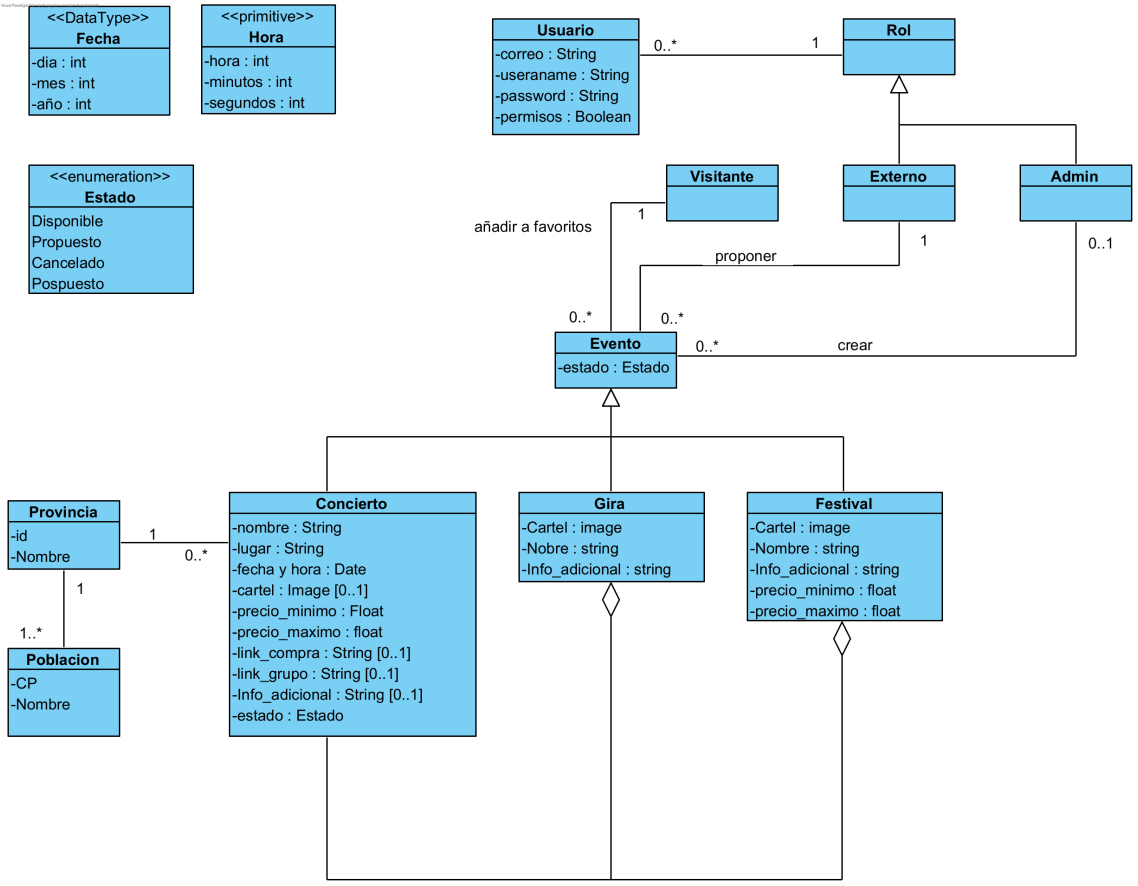


Figura 23: Modelo de dominio del supuesto

3.10. Diagramas de actividad - Aplicación Web

A continuación, se muestran los diagramas de actividad correspondientes a los casos de uso implementados en la aplicación web.

CU-1, Crear Cuenta

Cuando un usuario desea registrarse en el sistema, solicita la opción de registro. El sistema despliega un formulario que solicita información como nombre, correo y contraseña. Una vez que el usuario completa y confirma el formulario, el sistema valida los datos. Si son correctos, registra al usuario, lo notifica sobre el éxito del proceso y lo redirige a la página principal. En caso de errores, como campos incompletos o datos inválidos, el sistema informa al usuario con mensajes claros para que pueda corregirlos.

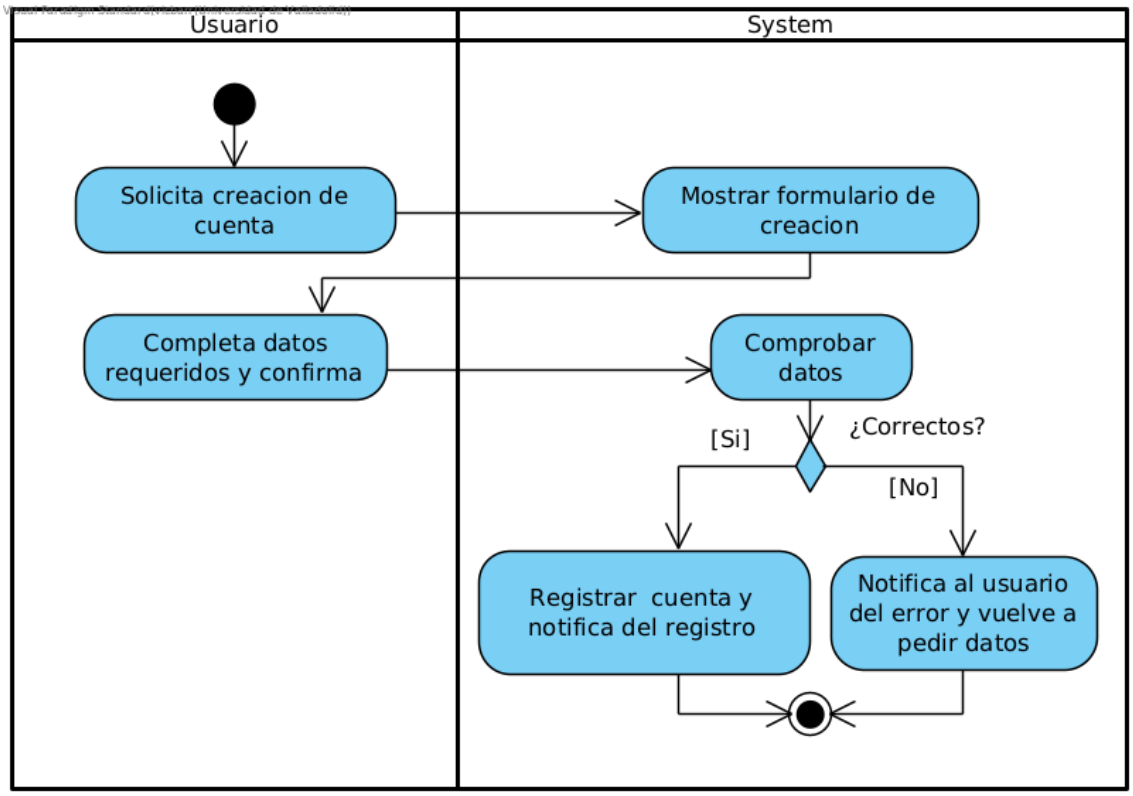


Figura 24: Diagrama de actividad del Caso de uso 1

CU-2, Identificación

Cuando el usuario desea identificarse envía una solicitud de identificación al sistema, este le solicita sus credenciales. Al introducirlas el sistema comprueba si estas son correctas y en caso de serlo queda identificado, en caso contrario sale un mensaje de error y vuelve a pedir los datos.

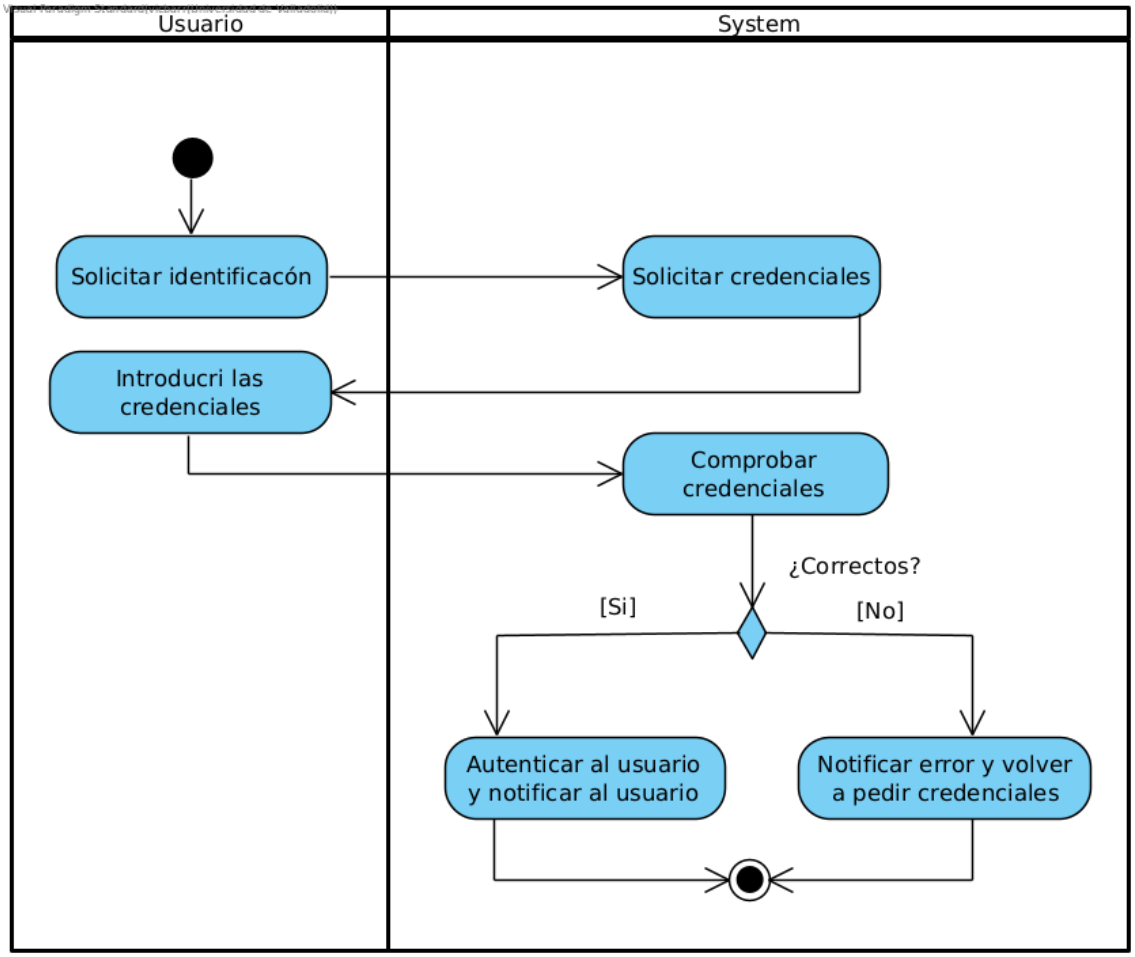


Figura 25: Diagrama de actividad del Caso de uso 2

CU-3, Cerrar sesión

Cuando un usuario desea cerrar su sesión activa, solicita esta opción dentro del sistema. El sistema invalida la sesión actual y confirma al usuario que el cierre de sesión se ha realizado con éxito. En caso de que el usuario no tenga una sesión activa, el sistema notifica que no es posible realizar la acción de cierre de sesión. Tras este proceso, el usuario queda sin una sesión activa y deberá identificarse nuevamente para acceder al sistema.

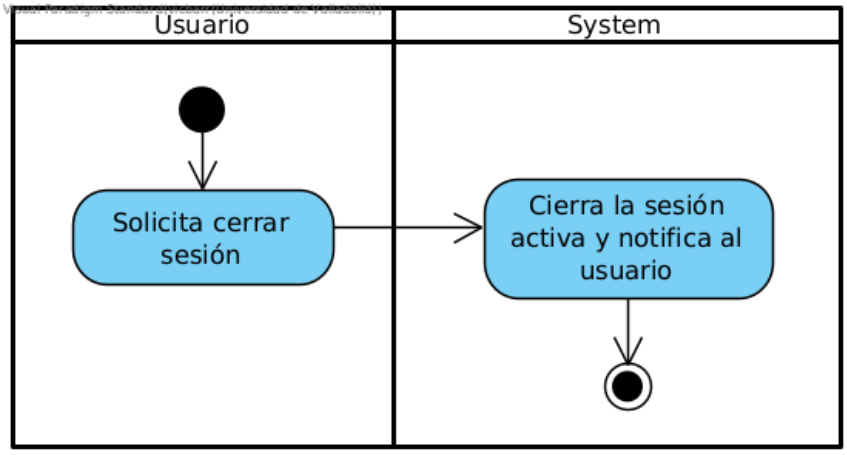


Figura 26: Diagrama de actividad del Caso de uso 3

CU-4, Proponer un evento.

Cuando un usuario externo desea proponer un evento, primero solicita realizar una propuesta y el sistema permite ingresar los detalles necesarios para la propuesta. El usuario proporciona la información requerida sobre el evento y confirma el envío. El sistema verifica que todos los datos sean correctos y completos. Si la verificación es exitosa, el sistema registra la propuesta y la envía para revisión de un administrador. Si algún campo está incompleto o contiene errores, el sistema notificará al usuario los detalles a corregir. La propuesta queda registrada y pendiente de aprobación por parte de un administrador.

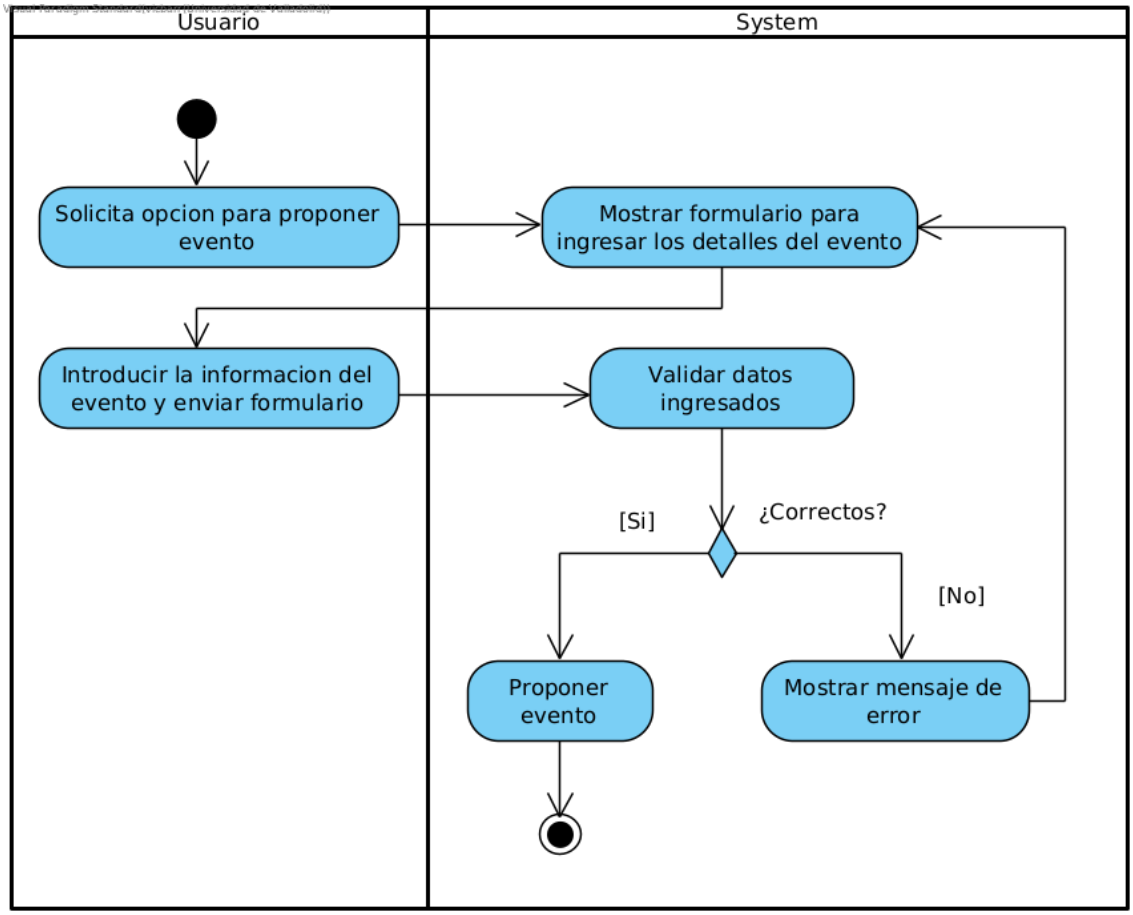


Figura 27: Diagrama de actividad del Caso de uso 4

CU-5, modificar un evento propuesto

Cuando un usuario externo desea modificar uno de los eventos que había propuesto con anterioridad primero accede a la sección correspondiente para modificar sus eventos propuestos. El sistema solicita los datos actuales con opción de cambio y el usuario proporciona la información de los cambios que quiere realizar, confirmando el registro. El sistema valida los datos proporcionados y, si la validación es exitosa, propone los cambios para el evento. Si algún dato es incorrecto o está incompleto, el sistema notificará al usuario los ajustes necesarios.

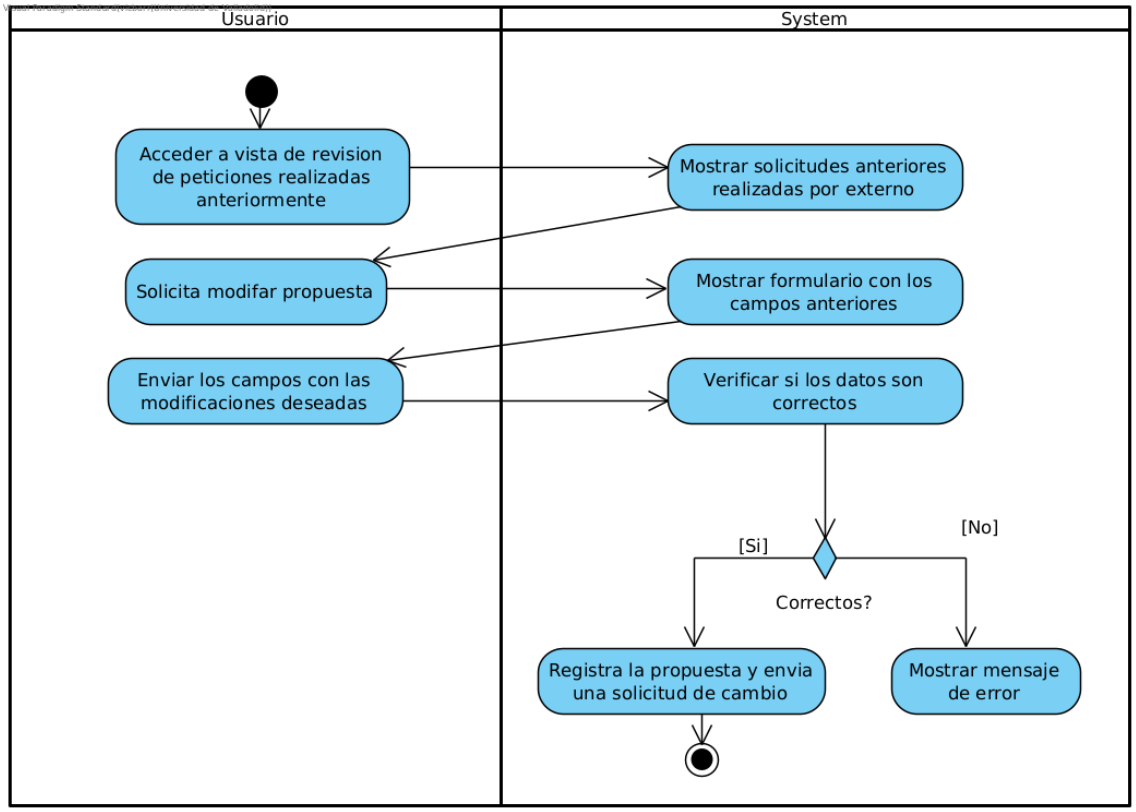


Figura 28: Diagrama de actividad del Caso de uso 5

CU-6, agregar evento

Cuando un usuario administrador desea añadir un evento, primero accede a la sección correspondiente para añadir un nuevo evento. El sistema solicita los datos necesarios para registrar el evento, y el usuario proporciona la información requerida, confirmando el registro. El sistema valida los datos proporcionados y, si la validación es exitosa, publica el evento, haciéndolo disponible para los usuarios. Si algún dato es incorrecto o está incompleto, el sistema notificará al usuario los ajustes necesarios. El evento queda registrado y publicado, disponible para los usuarios finales.

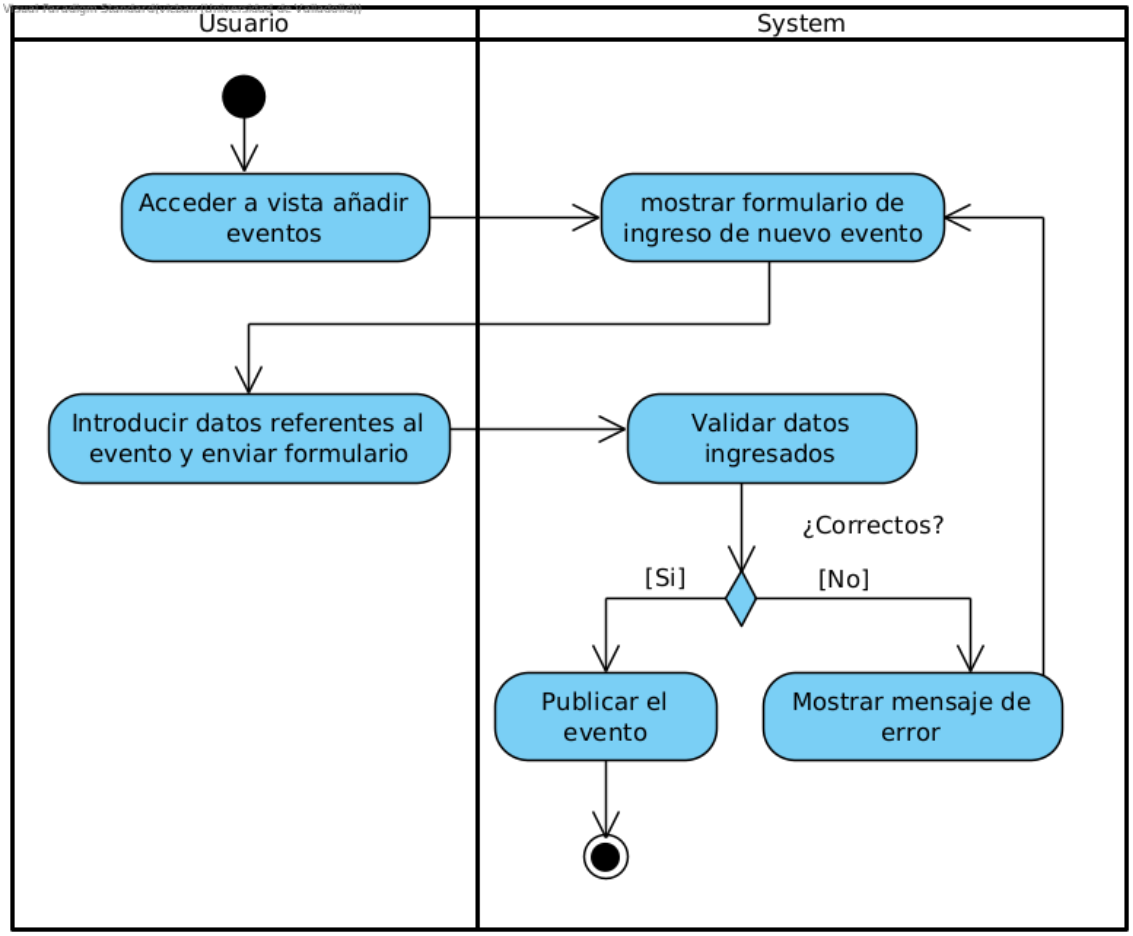


Figura 29: Diagrama de actividad del Caso de uso 6

CU-7, aceptar un evento

Cuando el usuario administrador quiere aceptar un evento este debe acceder a la sección para revisar las propuestas, entonces el sistema mostrara la lista de propuestas realizadas por Externos. El usuario tendrá la opción de aceptar la propuesta directamente desde la lisa o pulsando el desplegable para ver los datos de este evento. El sistema mostrará los datos del evento seleccionado, el usuario confirma que la aceptación del evento. El sistema validara los datos del evento y lo publicará, notificando al Externo que propuso este evento.

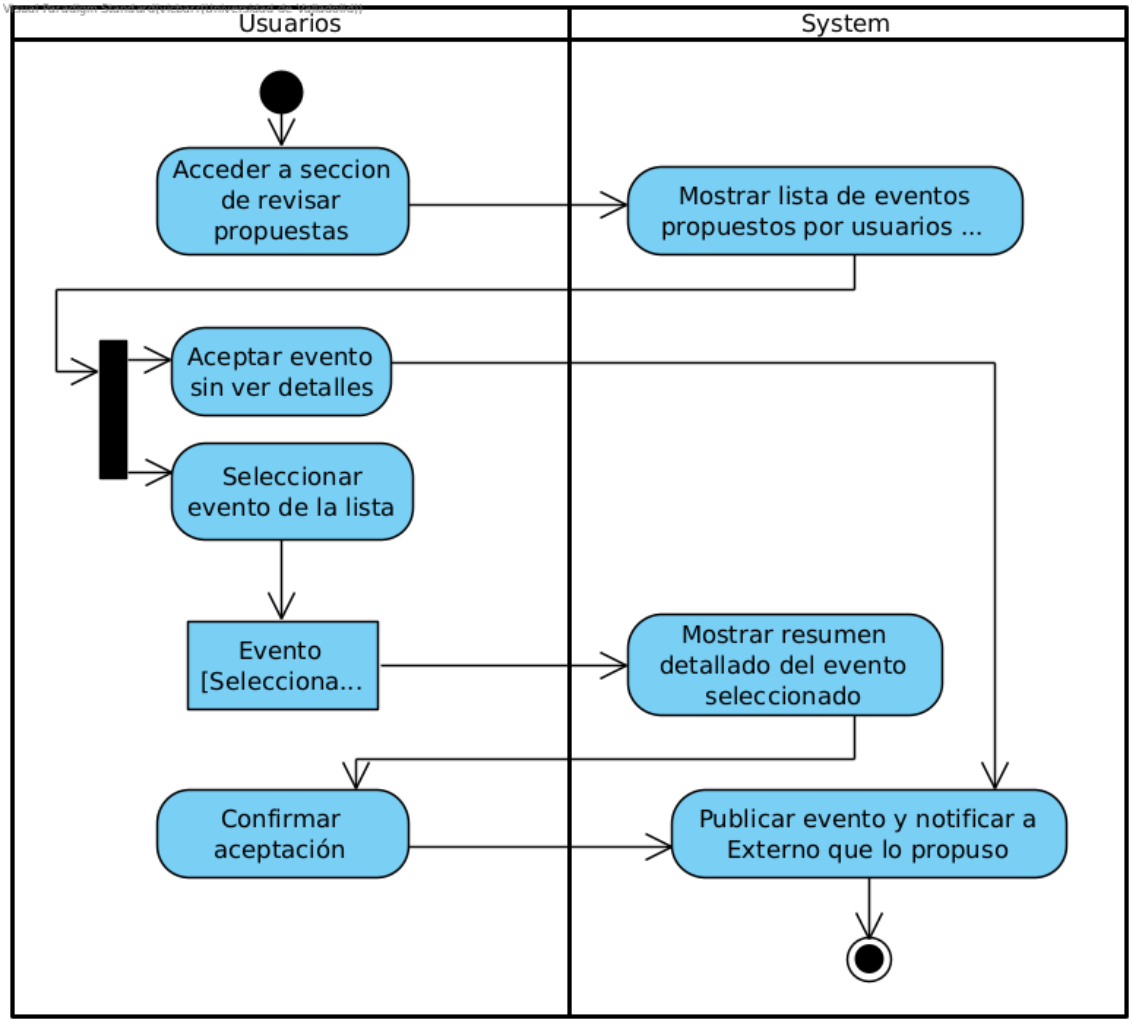


Figura 30: Diagrama de actividad del Caso de uso 7

CU-8, rechazar un evento

Cuando el usuario administrador desea rechazar un evento, debe acceder a la sección para revisar las propuestas. Entonces, el sistema mostrará una lista de las propuestas realizadas por los Externos. El usuario tendrá la opción de rechazar la propuesta directamente desde la lista o pulsando el desplegable para ver los detalles de este evento. El sistema mostrará los datos del evento seleccionado, y el usuario confirma el rechazo del evento. El sistema eliminará el evento de la lista de propuestas y notificará al Externo que propuso este evento sobre su rechazo.

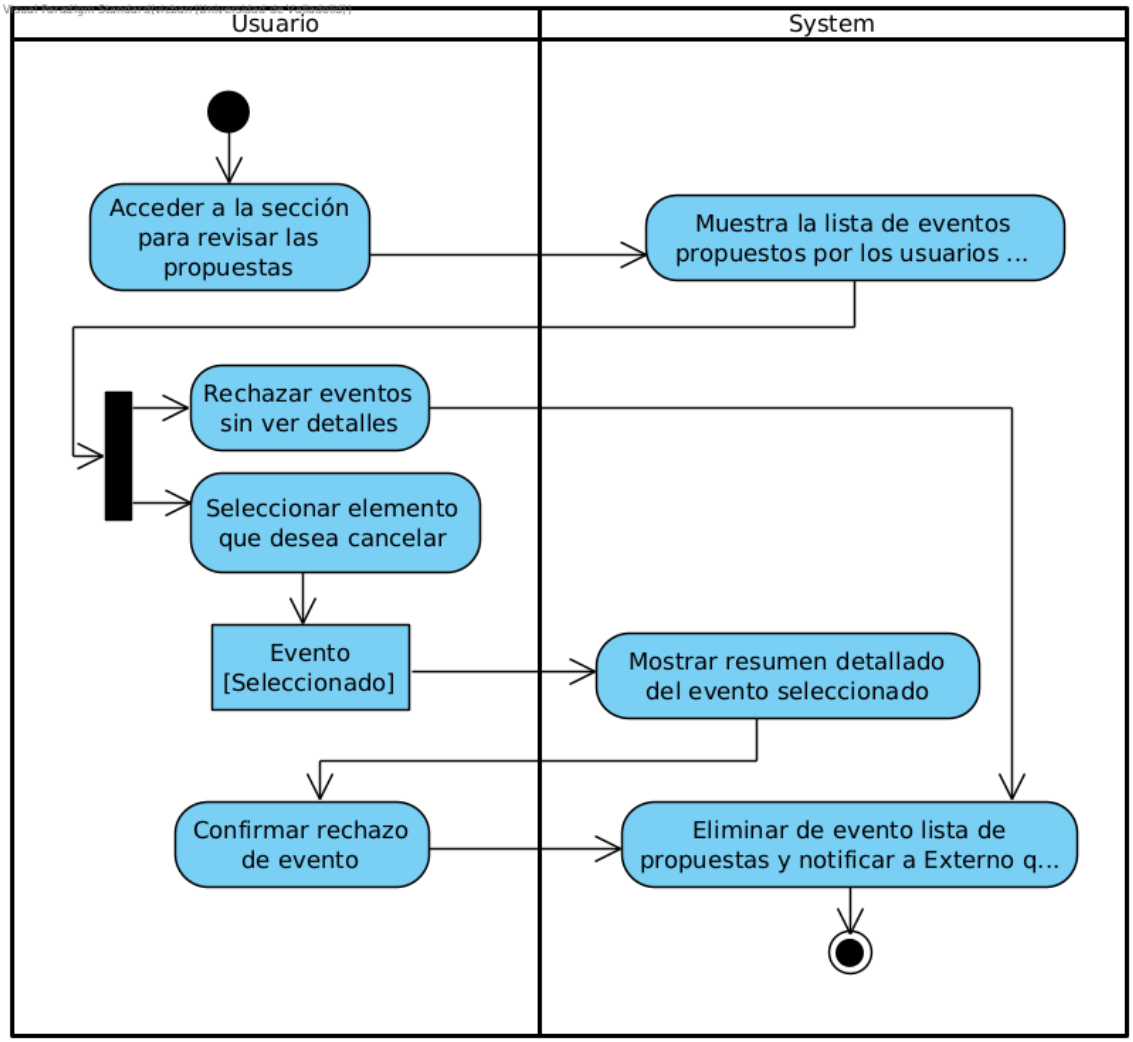


Figura 31: Diagrama de actividad del Caso de uso 8

CU-9, Modificar evento.

El usuario accede al panel de administración de eventos, el sistema muestra la lista de eventos registrados, el usuario selecciona el evento que desea modificar, el usuario muestra el formulario editable con los datos del evento, el usuario actualiza el contenido de los campos que quiera cambiar, el sistema valida los datos ingresados. En caso de que estos sean validos actualiza la información del evento y guarda los cambios, en caso contrario notifica del error.

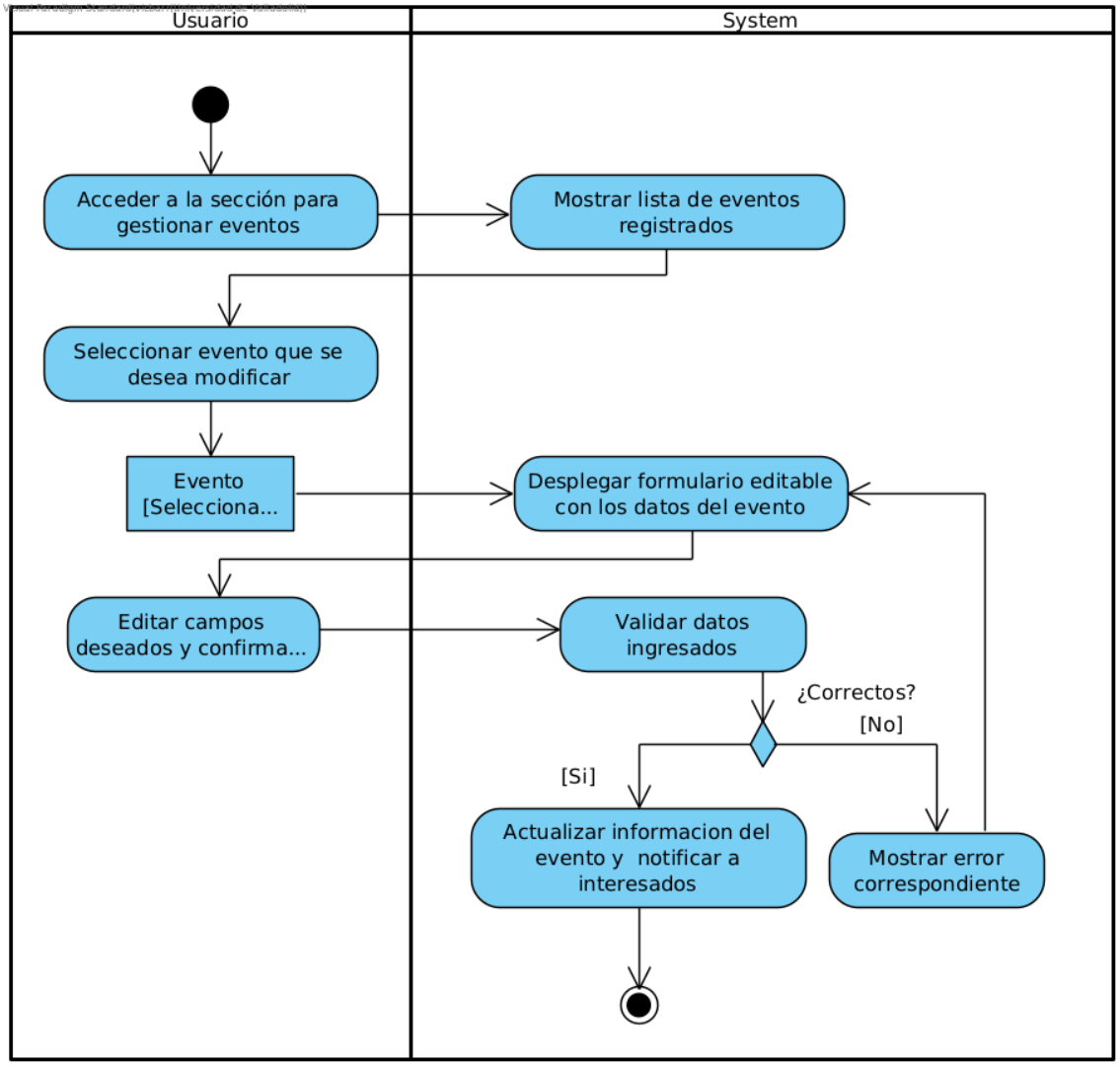


Figura 32: Diagrama de actividad del Caso de uso 9

CU-10, Agregar eventos a través de archivo CSV.

El usuario accede al panel de añadir eventos, el sistema muestra la lista de opciones para agregar un evento, el usuario selecciona la opción de añadir a través de un .CSV, el usuario inserta o selecciona el .CSV con los eventos que quiere cargar, el sistema valida los datos ingresados. En caso de que estos sean validos nos indica que el .CSV tiene una estructura incorrecta, en caso contrario notifica del error.

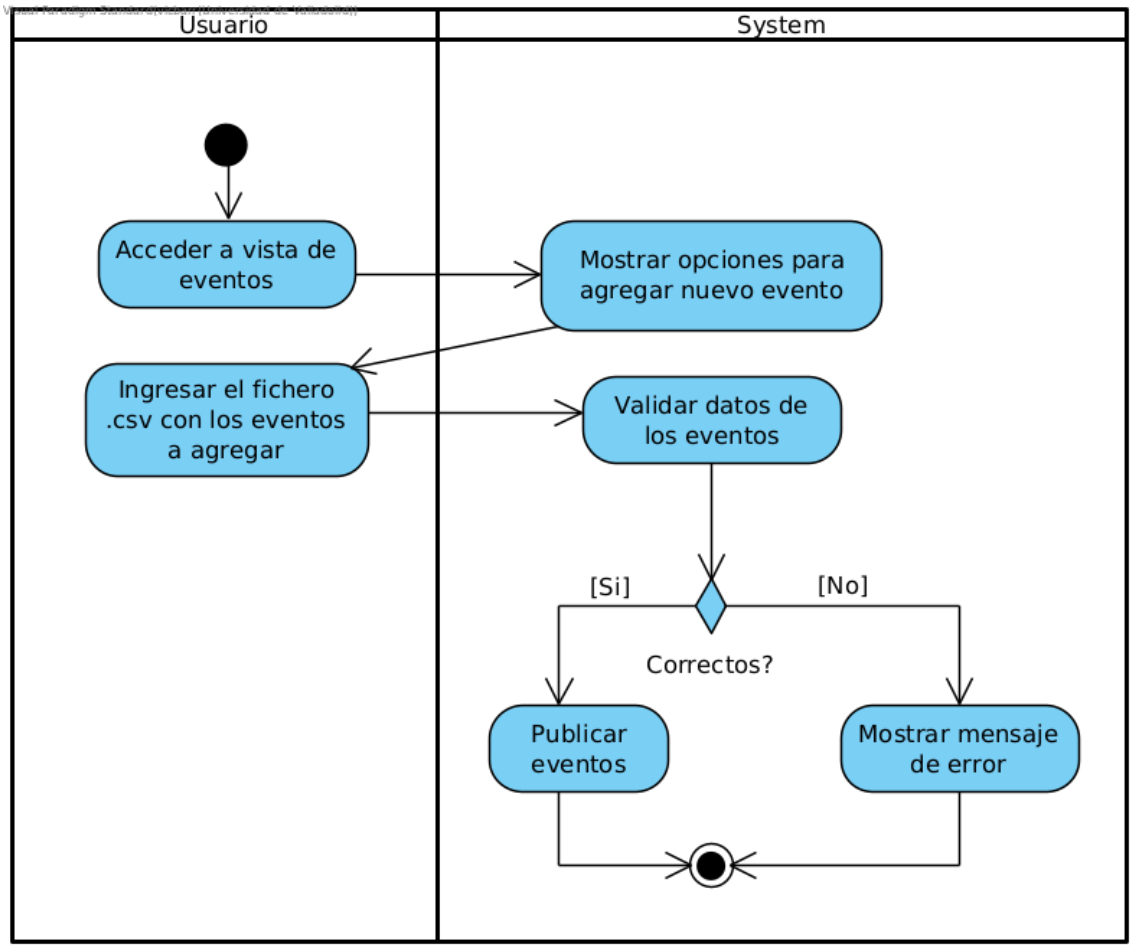


Figura 33: Diagrama de actividad del Caso de uso 10

CU-11, Modificar cuenta.

El usuario accede a la sección de modificación de perfil dentro del sistema, el sistema solicita que ingrese una nueva contraseña y que la repita en un segundo campo para confirmarla. El usuario introduce ambos campos con la nueva contraseña y envía el formulario. El sistema valida los datos proporcionados. Si la validación es exitosa, el sistema actualiza las credenciales de la cuenta con la nueva contraseña establecida. En caso de que la nueva contraseña sea incorrecta o contenga caracteres inválidos, el sistema muestra un mensaje de error indicando el problema y solicita al usuario corregir los datos ingresados.

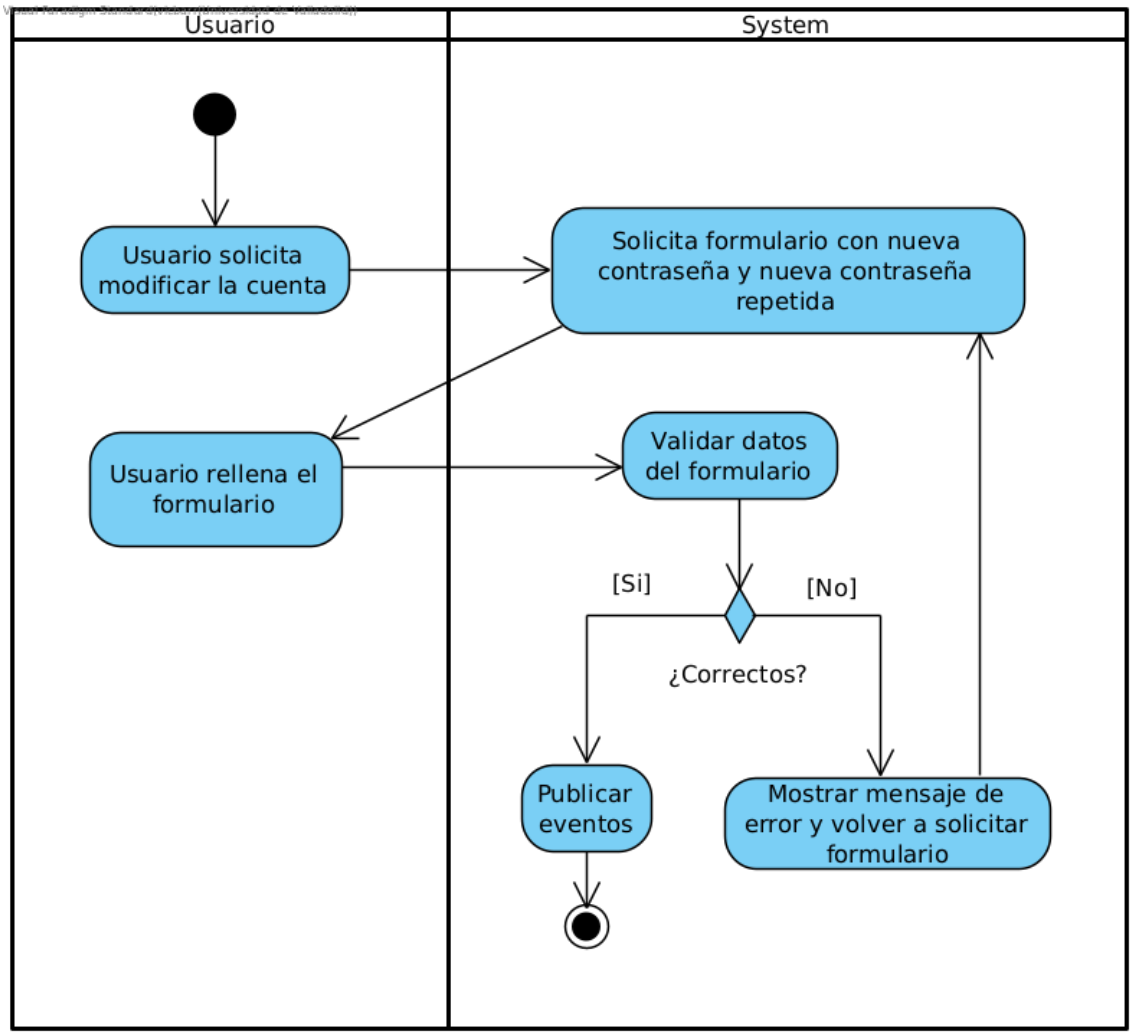


Figura 34: Diagrama de actividad del Caso de uso 11

3.11. Diagramas de actividad - Aplicación Móvil

A continuación, se muestran los diagramas de actividad correspondientes a los casos de uso implementados en la aplicación móvil.

CU-1, Ver eventos disponibles

Cuando el usuario quiere visualizar los eventos disponibles, solicita la lista desde el sistema. Este muestra una lista de eventos con sus respectivos detalles, permitiendo al usuario conocer las opciones disponibles. Si no hay eventos registrados, el sistema notifica al usuario con un mensaje claro indicando que no hay eventos disponibles en ese momento.

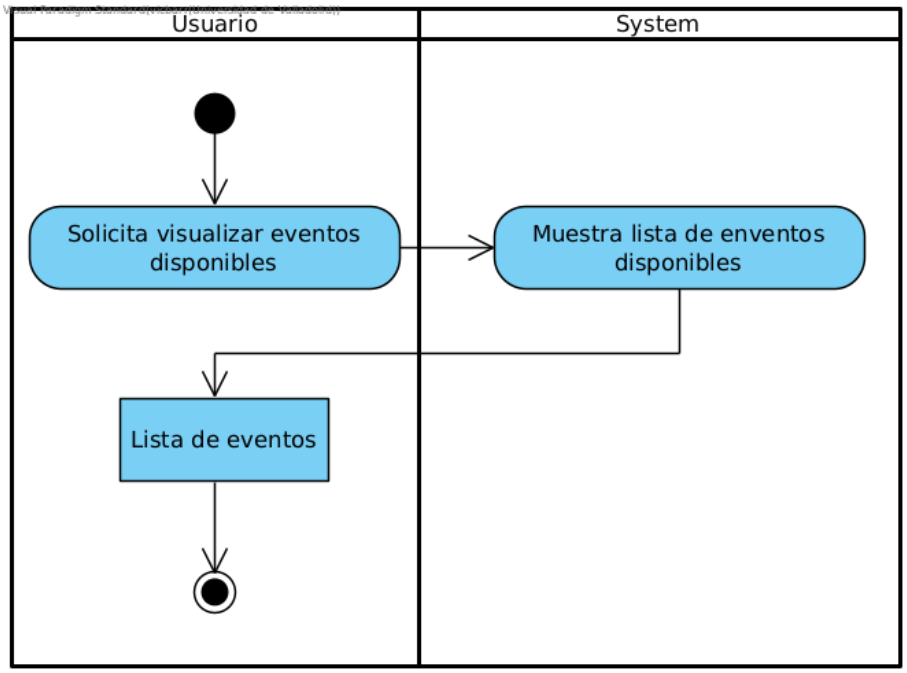


Figura 35: Diagrama de actividad del Caso de uso 1

CU-2, Filtrar búsquedas

Cuando el usuario desea filtrar búsquedas este debe acceder a la sección de búsqueda, el sistema le mostrara los filtros de búsqueda. Posteriormente el usuario seleccionará el filtro necesario e introducirá los datos de búsqueda. El sistema procesará estos filtros y criterios y mostrará los resultados.

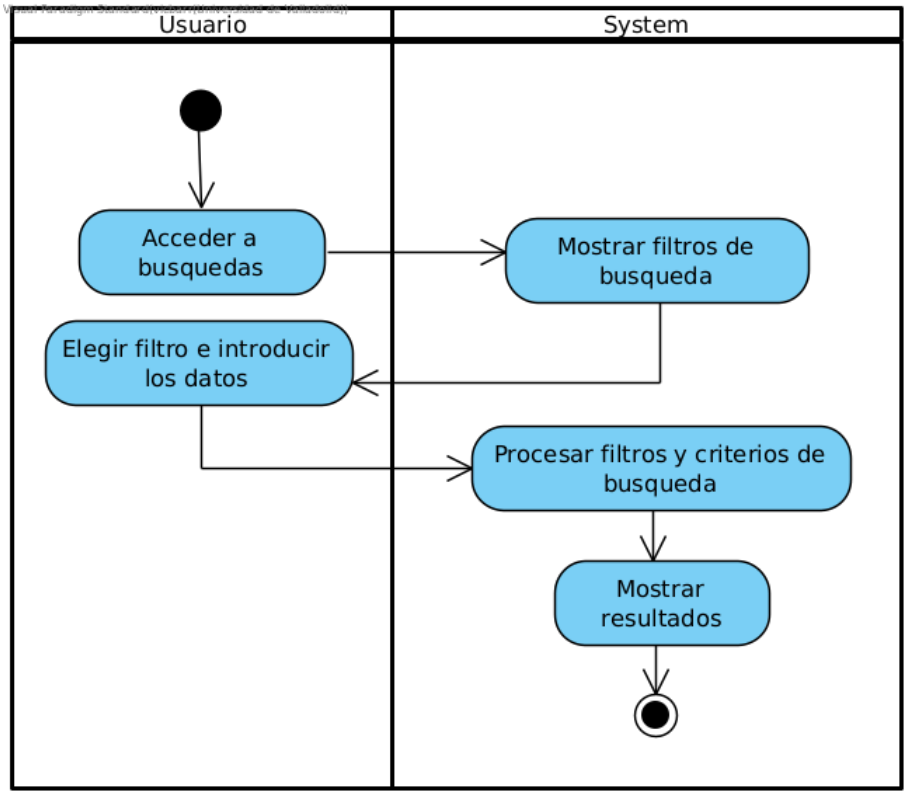


Figura 36: Diagrama de actividad del Caso de uso 2

CU-3, Agregar evento a favoritos

Cuando un usuario desea seguir un evento, primero identifica el evento que le interesa. El usuario puede elegir agregar a favoritos el evento desde la lista general de eventos o desde la vista detallada del evento. El sistema registra la acción del usuario y actualiza su lista personal de eventos seguidos, confirmando que el evento se ha añadido correctamente a su lista.

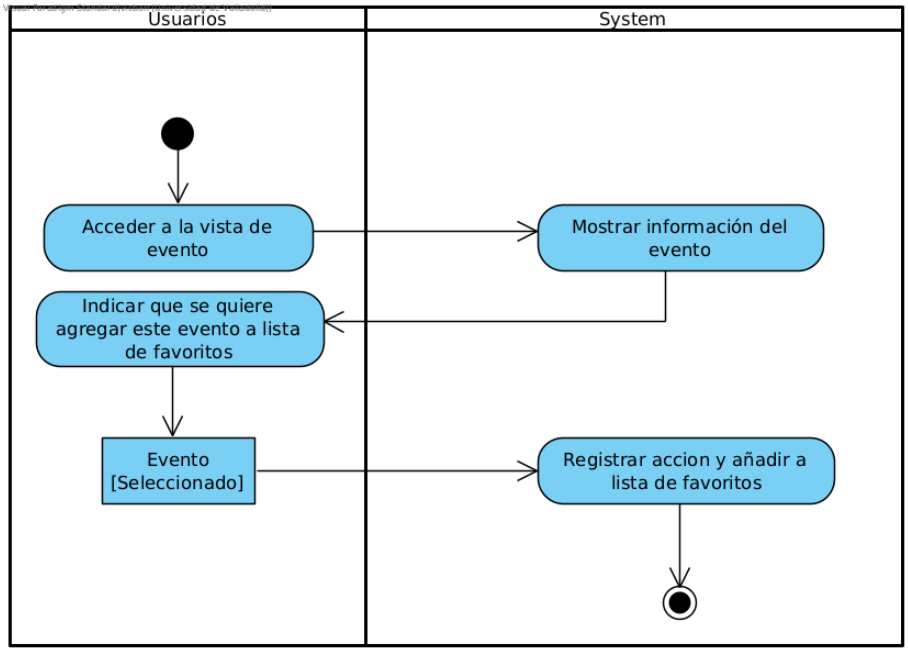


Figura 37: Diagrama de actividad del Caso de uso 3

CU-4, Eliminar evento a favoritos

Cuando un usuario desea eliminar un evento de la lista de favoritos primero identifica el evento que le interesa. El usuario puede elegir eliminar e evennto de la lista de favoritos desde la lista general de eventos o desde la vista detallada del evento. El sistema registra la acción del usuario y actualiza su lista personal de favoritos , confirmando que el evento se ha eliminado correctamente a su lista.

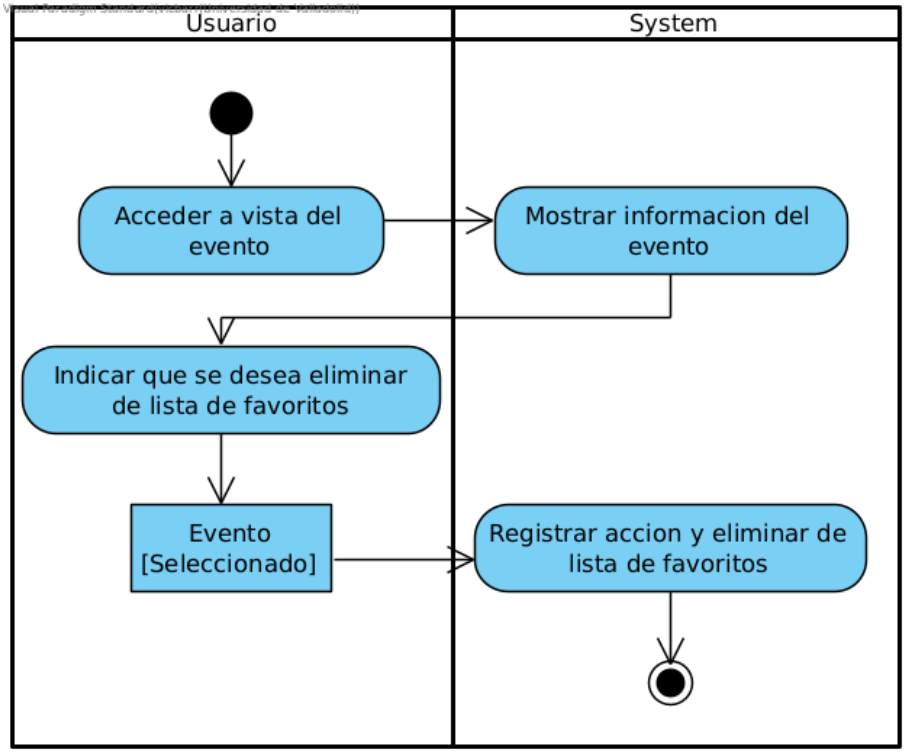


Figura 38: Diagrama de actividad del Caso de uso 4

CU-5, Compartir evento por redes sociales

Cuando el usuario pulse en la opción de compartir evento el sistema desplegará las opciones de compartir en redes sociales. El usuario seleccionará la opción y el sistema generará un enlace al evento y abrirá la aplicación seleccionada. El usuario enviará el enlace en la aplicación externa de mensajería o red social oportuna.

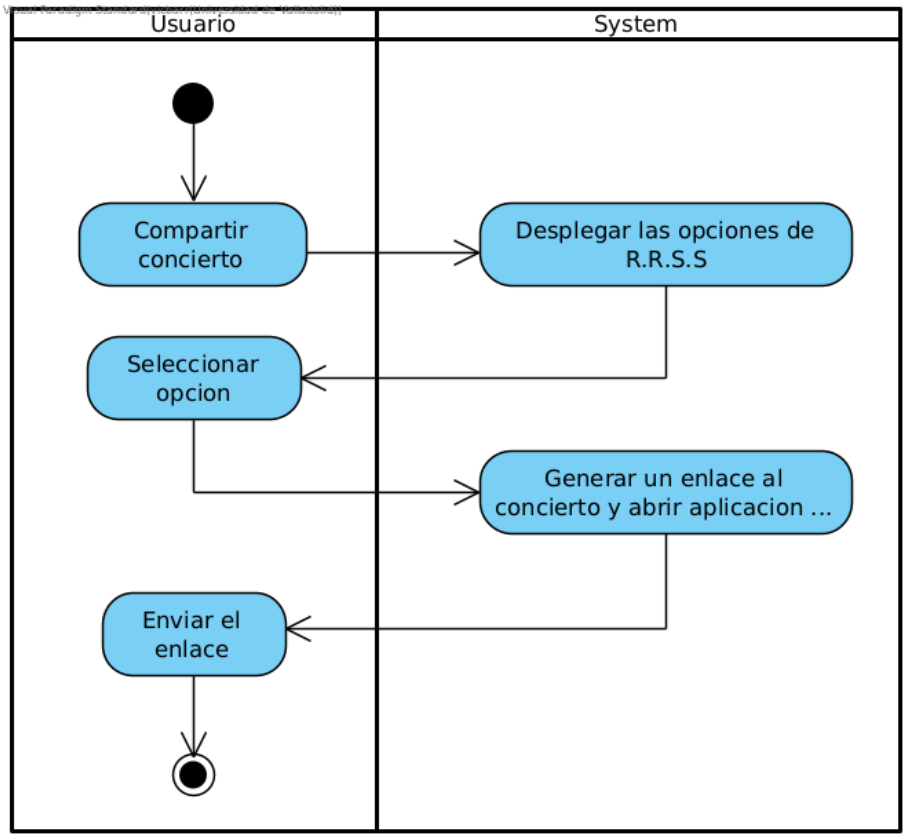


Figura 39: Diagrama de actividad del Caso de uso 5

CU-6, Ver lista de eventos favoritos

Cuando el usuario desea revisar el calendario de eventos que están en la lista de favoritos, accede a la vista de calendario de favoritos, donde el sistema presenta un calendario con los eventos que el usuario ha marcado como favoritos. El usuario puede navegar por las fechas para ver los eventos correspondientes y el sistema ofrece detalles básicos de cada evento. Si no hay eventos marcados como favoritos, el sistema mostrará un mensaje indicando que no hay eventos para mostrar.

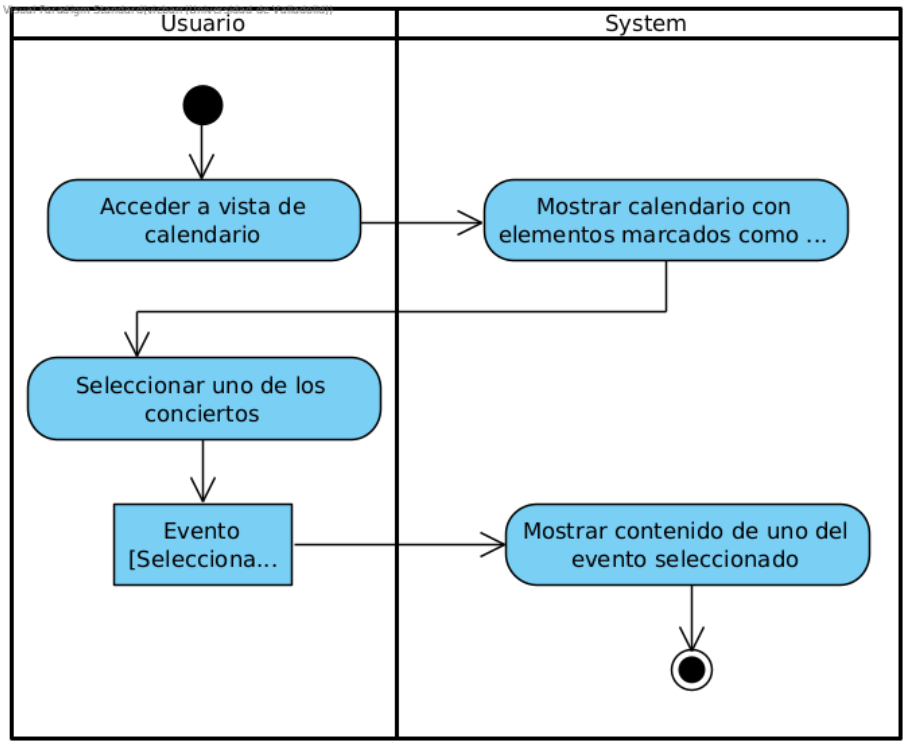


Figura 40: Diagrama de actividad del Caso de uso 6

3.12. Mock-Up de la Aplicación

Conociendo cuales son los requisitos de la aplicación, podemos plantear como sera la implementación de la UI.

A continuación se presentan los mock-ups diseñados para la interfaz de la aplicación [43]. Las figuras han debido ser divididas debido a que, de otro modo, la visualización de las funcionalidades era realmente complicada.

En la figuras 41 y 42 se muestra el diseño de la aplicación para dispositivos móviles, enfocado principalmente en la visualización de contenidos de manera intuitiva y accesible.

Por otro lado, en la Figura 43 y 44 se presenta el diseño de la versión web de la aplicación, orientada principalmente a la gestión de contenidos. Esta interfaz está pensada para facilitar a los administradores y usuarios autorizados la publicación y organización de eventos.

A continuación podemos ver la primera sección de vistas de la aplicación móvil. En concreto se explicarán la funcionalidad de las vistas definidas en la figura 41.

Se explicará el planteamiento de estas de izquierda a derecha, desde la parte superior hasta la inferior:

- **Visualización de eventos disponibles en vista de calendario**: Esta vista nos ofrece un display de los conciertos en forma de calendario haciendo accesible el contenido de los próximos eventos de una manera visualmente atractiva.
- **Visualización de eventos favoritos en vista de calendario**: Similar a la vista anterior, nos permite ver los conciertos favoritos del usuario.

- **formulario para agregar evento**: Esta vista muestra un pequeño formulario conformado por todos los campos necesarios para agregar un evento.
- **Vista principal de display de eventos disponibles**: consiste en un scroll vertical el cual nos permite visualizar los datos de los eventos más próximos. En este se muestran pequeñas tarjetas con el contenido principal de los eventos.
- **Visualización de desplegable de evento**: En esta vista se muestra el desplegable que sale al pulsar sobre el evento de interés. Este nos muestra todos los datos del evento y la opción de compartirlo.

Respecto a las interfaces mostradas en la segunda sección de la aplicación móvil, de nuevo siguiendo el orden de descripción (Arriba-Abajo, Izquierda-Derecha) se explicará brevemente la funcionalidad planteada para cada una de las visualizaciones descritas en la figura 42

- **Visualización de Login**: Esta vista consiste en el furmulario de login para los usuarios.
- **Visualización de Registro**: Consiste en el registro o Sign-Up de usuario. Sección a la que deberá acceder el usuario para crear su cuenta
- **Visualización de ajustes de usuario**: Muestra datos sobre el usuario registrado
- **Vista eventos propuestos por el propio usuario**: Muestra la lista de eventos que ha propuesto el usuario.
- **Visualización de edición de eventos propuestos**: Muestra formulario de edición el cual permite modificar los datos de un concierto ya propuesto.

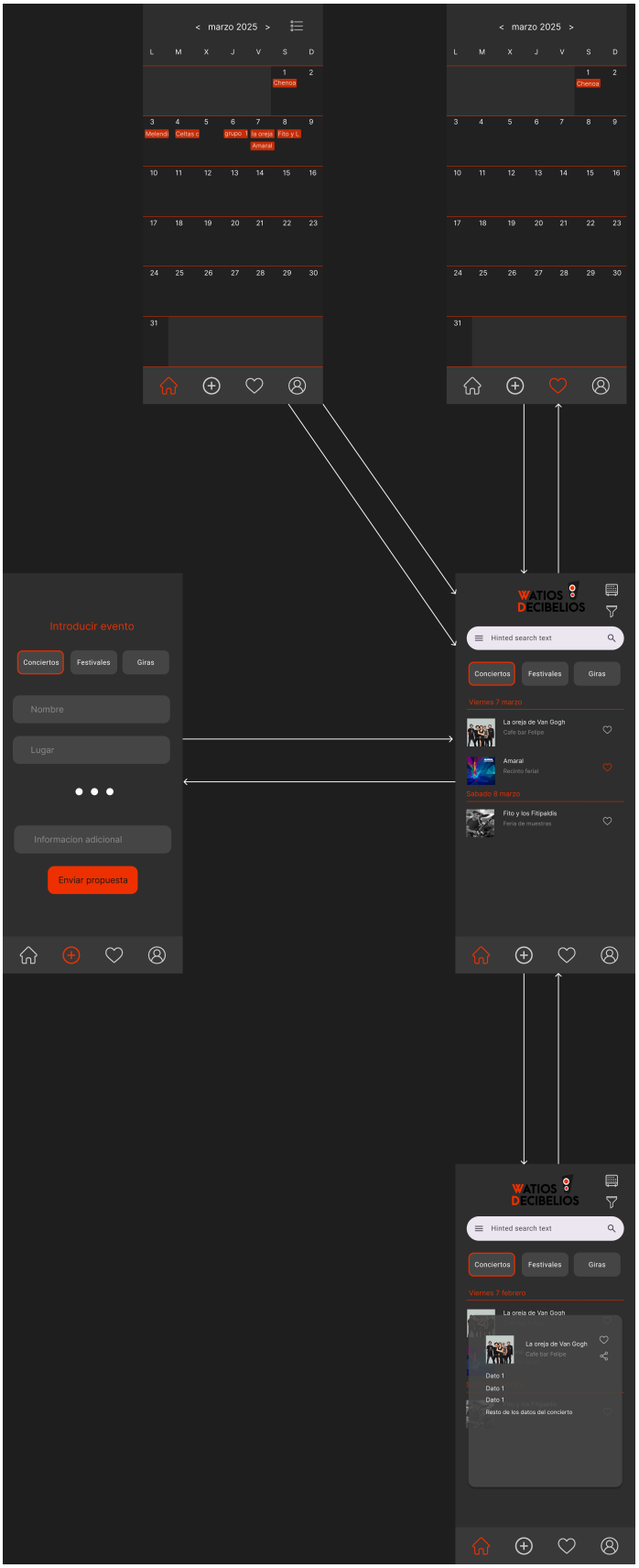


Figura 41: Diseño de la aplicación para dispositivos móviles, Parte I.

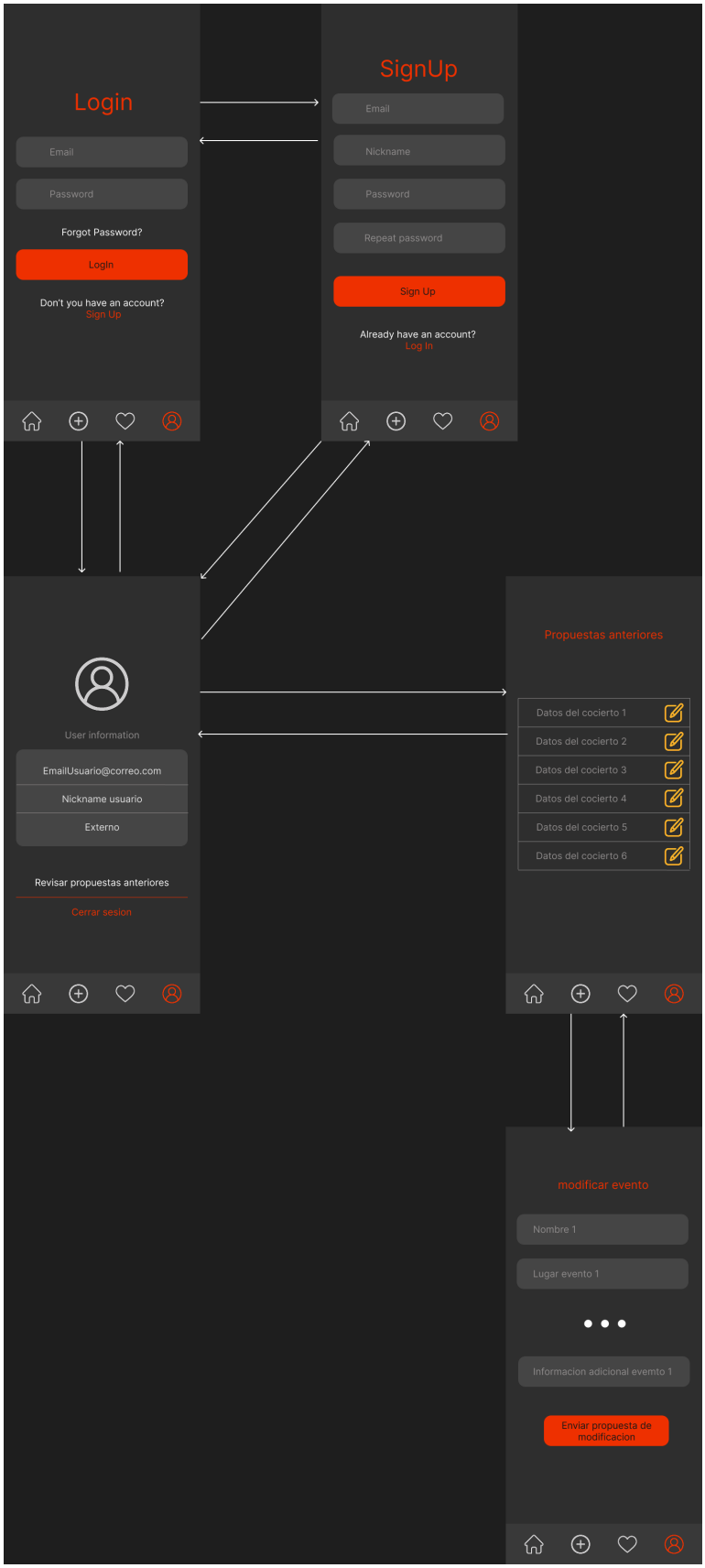


Figura 42: Diseño de la aplicación para dispositivos móviles, Parte II.

Continuación se muestran la interfaz de parte de las funcionalidades planteadas para la aplicación web. En La figura 43 se muestran las siguientes vistas:

- **Vista de Login**: permite a los usuarios loguearse con sus credenciales
- **Vista de display de eventos disponibles**: esta vista nos muestra una lista con los eventos disponibles mas próximos. Nos permite editar el contenido de estos.
- **Visualización de Registro**: Consiste en el registro o Sign-Up de usuario. Sección a la que deberá acceder el usuario para crear su cuenta
- **Vista Home**: Consiste en la vista a la cual accedemos tras un login o registro correcto

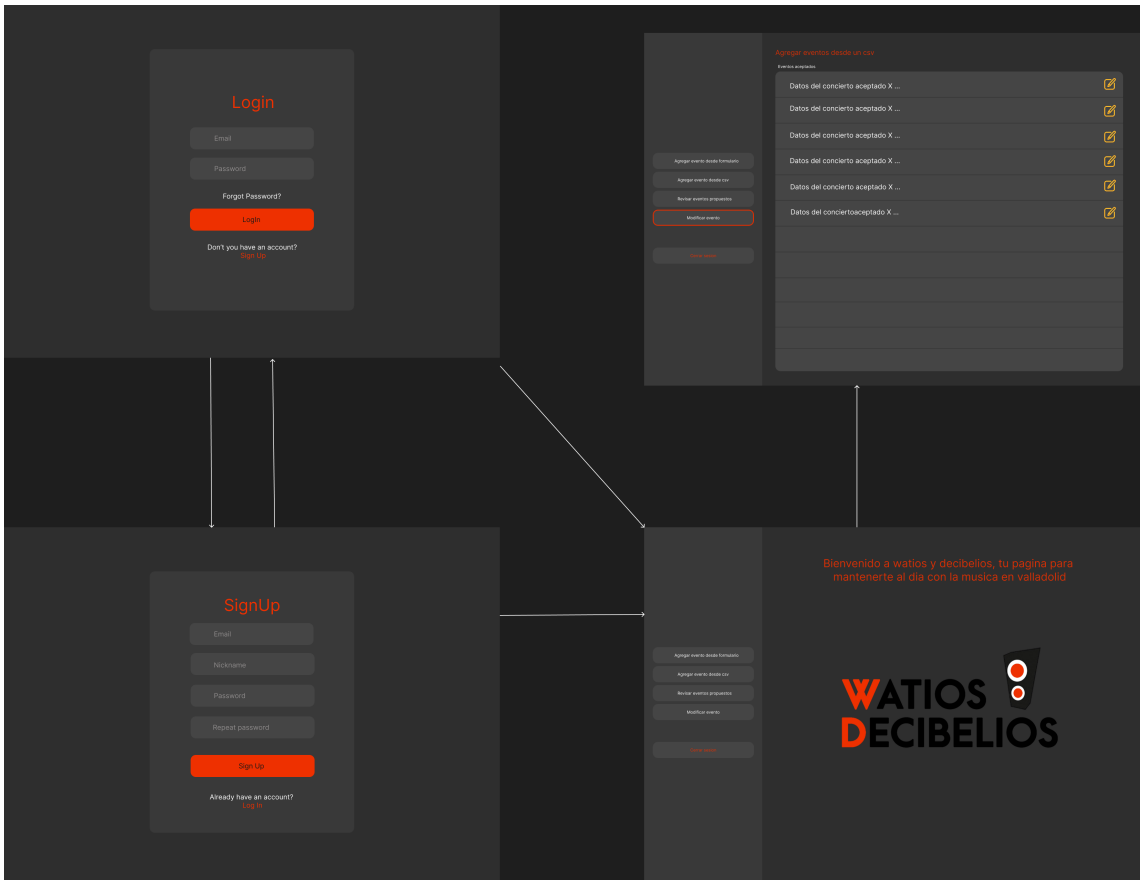


Figura 43: Diseño de la aplicación para plataformas web, Parte I

Para finalizar con esta sección se muestran la interfaz de las demas funciones disponibles en la aplicativo web. En La figura 44 se muestran las siguientes vistas:

- **Vista Home**: Vista inicial
- **Vista de display de eventos pendientes**: Nos muestra los datos de los conciertos propuesto y nos brinda la opción de aceptar, rechazar o editar estos
- **Visualización de formulario de adición de evento**: Nos muestra un formulario a traves del cual podremos introducir los datos del evento, tanto conciertos como giras y festivales.
- **Vista de agregar .CSV**: vista que nos permite subir un archivo que contenga múltiples conciertos agregando estos.

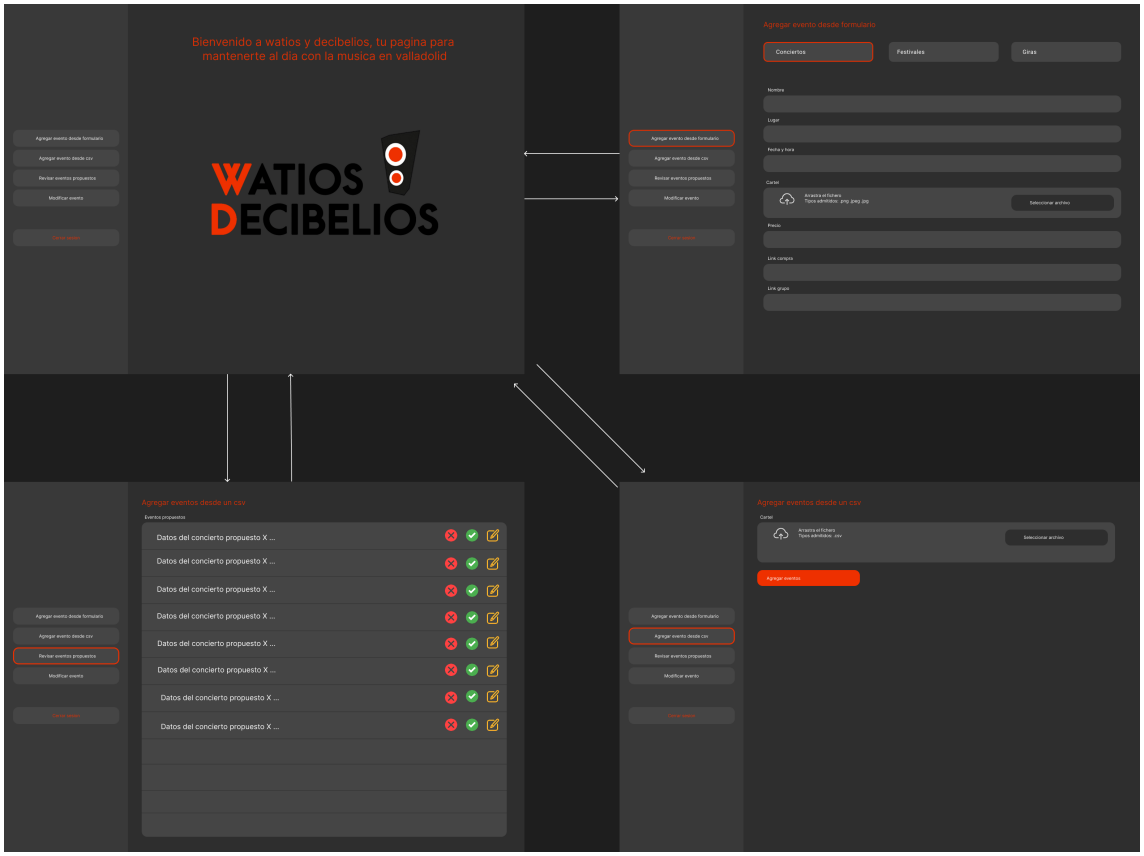


Figura 44: Diseño de la aplicación para plataformas web, Parte II

Capítulo 4

4. Diseño

4.1. Estructura de los proyectos

En la siguiente sección se va a detallar en profundidad cómo ha quedado organizada la estructura del proyecto, abarcando tanto la base de datos como las distintas partes que conforman el front-end y el back-end, tanto en la aplicación web como en la versión móvil.

4.1.1. Diseño de la base de datos

A continuación se muestran las tablas que conforman la base de datos del proyecto. Estas junto con el diagrama relacional representado en la figura 45 permiten entender de manera clara cómo se relacionan las distintas entidades y facilita la comprensión de la estructura interna del sistema. Para el diseño y desarrollo de estas se ha procurado seguir las buenas prácticas definidas en los siguientes libros: [38, 3]

Cabe destacar que la base de datos ha sufrido cambios sustanciales en el curso de la practica, como se muestra en el capítulo 5.

Las tablas son las siguientes:

- **estado**: recoge los distintos estados que puede tener un evento, como por ejemplo disponible, propuesto, cancelado o pospuesto.
- **usuarios**: almacena los datos de las personas registradas, incluyendo su correo electrónico, nombre de usuario, contraseña cifrada, permisos y otros detalles útiles para la gestión de cuentas.
- **provincias y poblaciones**: estas dos tablas contienen la información geográfica, es decir, las provincias y los municipios donde pueden celebrarse los eventos. Cada población está asociada a una provincia concreta.
- **conciertos**: aquí se guarda la información principal de cada concierto, como el nombre, el lugar, el cartel, el rango de precios, datos adicionales, el usuario que lo organiza, el estado en el que se encuentra, la fecha y la hora, además de la provincia y la población donde se realiza.
- **giras y festivales**: estas tablas sirven para gestionar las giras y festivales musicales, incluyendo su nombre, cartel, información adicional, estado, organizador y, en el caso de los festivales, el precio mínimo y máximo.
- **concierto_pertenece_festival y concierto_pertenece_gira**: son tablas auxiliares que permiten relacionar cada concierto con el festival o la gira a la que pertenece, facilitando así la organización de eventos agrupados.
- **user_sessions:** registra las sesiones activas de los usuarios, guardando información como la fecha de inicio y fin de sesión, la dirección IP y el agente de usuario.

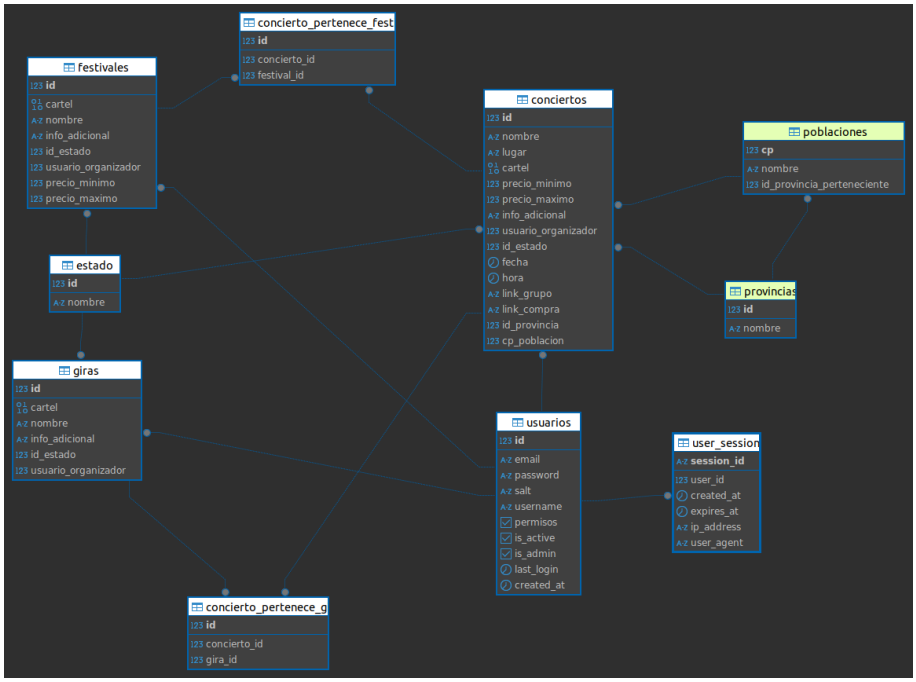


Figura 45: Diagrama relacional de la base de datos final

4.1.2. Estructura de los proyectos de la app web

La estructura del proyecto desarrollado con Python y Streamlit presenta una organización basada en una arquitectura cliente-servidor. En esta, el backend implementado en Python actúa como servidor, mientras que el frontend, accesible mediante un navegador web, funciona como cliente. Además, se sigue el patrón de diseño MVC (Modelo-Vista-Controlador), lo cual permite separar la lógica de negocio, la presentación de datos y la lógica de control en componentes distintos y modulares. Este patrón es explicado en profundidad en 4.2.1

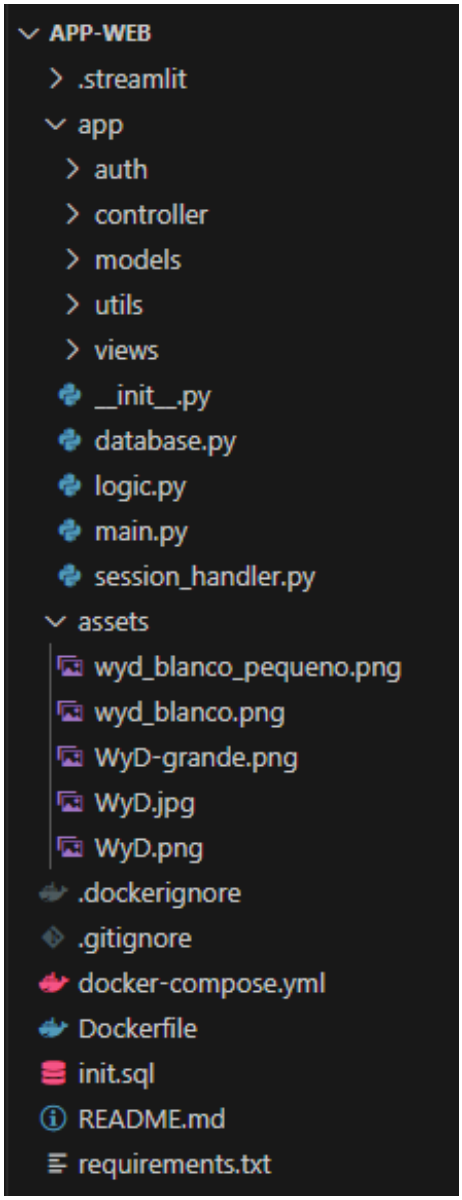


Figura 46: Sistema de carpetas del proyecto web

El directorio principal del proyecto, denominado `APP-WEB`, contiene toda la estructura organizada en diversos subdirectorios. Entre ellos destacan los siguientes:

- **`streamlit`**: Incluye configuraciones específicas del framework Streamlit, que facilita el desarrollo del frontend con una interfaz gráfica simple y ordenada.
- **`app`**: Núcleo funcional de la aplicación, donde se implementa el patrón MVC. Este directorio contiene varios subdirectorios:
  - `controller`: Contiene los controladores responsables de la lógica de negocio. Procesan las solicitudes del usuario y se comunican con los modelos para manejar los datos. Algunos archivos que se incluyen son `available_events_controller.py`, `concerts_controller.py` y `create_account_controller.py`.

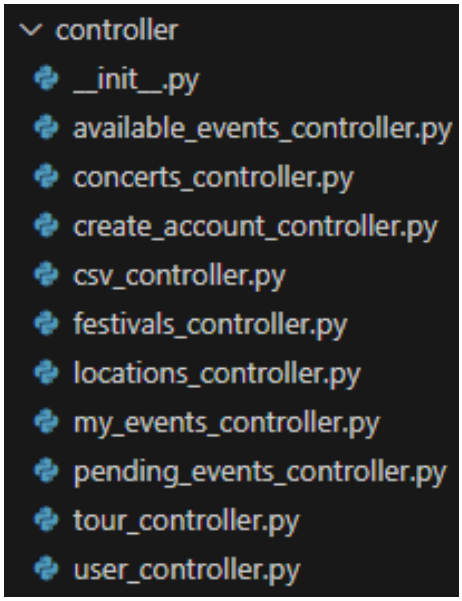


Figura 47: Archivos del directorio controller

- `models`: Incluye las clases que definen la estructura de datos y gestionan la interacción con la base de datos PostgreSQL. Cada modelo representa una entidad del sistema, como conciertos, festivales, giras, usuarios, etc. Algunos archivos destacados son `base_model.py`, `concert_model.py`, `festival_model.py` y `User_model.py`.

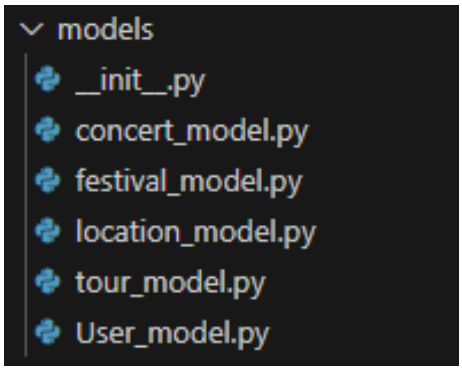


Figura 48: Archivos del directorio models

- `views`: Agrupa los componentes de la interfaz de usuario desarrollados con Streamlit. Define cómo se presentan los datos al usuario y cómo se gestionan sus interacciones. Algunos ejemplos son `account_settings_view.py` y `available_events_view.py`.

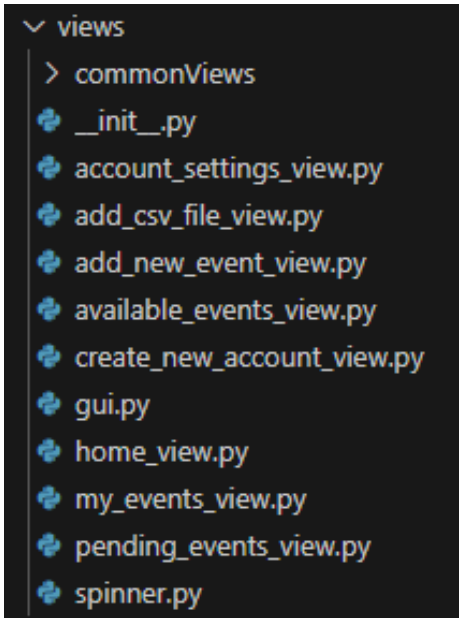


Figura 49: Archivos del directorio views

Ademas de estos directorios existen varios archivos que se encuentran en el directorio app, y su funcionalidad es la siguiente:

- **auth**: Encargado de la autenticación y gestión de usuarios, incluyendo el registro e inicio de sesión. Contiene archivos como `create_account.py` y `login.py`.
- **utils**: Reúne funciones auxiliares y componentes compartidos utilizados en distintas partes del proyecto. Entre los archivos más relevantes se encuentran `logic.py`, `database.py` y `session_handler.py`.
- **main.py** Es el punto de entrada de la aplicación. Importa y ejecuta la función principal de la interfaz gráfica (main) desde el módulo de vistas.
- **database.py**: Gestiona la conexión a la base de datos PostgreSQL usando un pool de conexiones. Implementa un patrón singleton para que solo exista una instancia de la clase Database y expone una instancia global (db_instance) para ser usada en otros módulos.
- **session_handler.py**: Maneja la gestión de sesiones de usuario en Streamlit. Crea, guarda, verifica y elimina sesiones persistentes usando un archivo JSON en disco, permitiendo mantener el estado de autenticación y otros datos del usuario entre recargas.
- **logic.py** Contiene la lógica para mostrar diferentes secciones de la aplicación según la selección del usuario. Llama a las funciones de las vistas correspondientes y maneja la carga de archivos (CSV/Excel) mostrando su contenido en Streamlit.

- `assets`: Almacena recursos estáticos como imágenes y logotipos utilizados en la interfaz de usuario.
- `venv`: Corresponde al entorno virtual de Python, que aísla las dependencias del proyecto para evitar conflictos con otros entornos.

Además de los directorios mencionados, en el nivel raíz del proyecto se encuentran archivos importantes como:

- `.env`: Contiene variables de entorno, incluyendo credenciales de bases de datos y parámetros sensibles.
- `docker-compose.yml`: Define la configuración para el despliegue de la aplicación usando Docker, incluyendo la orquestación de servicios.

4.1.3. Estructura de los proyectos de la app movil

A continuación se detallara la estructura que ha seguido el proyecto para dispositivos moviles basado principalmente en Node.js y en React-Native.

Estructura Back-End del proyecto Movil (Node.js)

El backend del proyecto móvil está desarrollado en Node.js utilizando Express.js como framework principal y PostgreSQL como sistema de gestión de base de datos. La arquitectura sigue un patrón de capas bien definido que separa las responsabilidades y facilita el mantenimiento del código.

El servidor se estructura en el directorio **server/** siguiendo una arquitectura modular donde cada componente tiene una responsabilidad específica. A continuación en la figura 51 se puede observar el sistema de carpetas de la sección correspondiente:

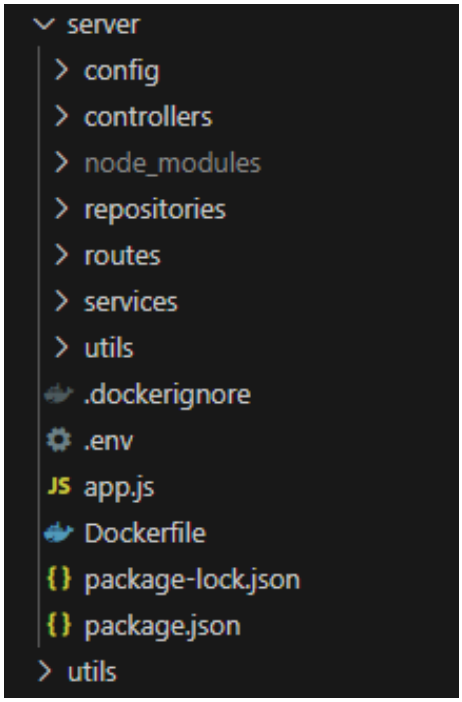


Figura 50: estructura de directorios del backend de la app-movil

El punto de entrada de la aplicación se encuentra en **app.js**, donde se configura el servidor Express con todos los middlewares necesarios. Este archivo establece la configuración CORS para permitir peticiones desde el cliente móvil, define el middleware de manejo de errores centralizado y configura el puerto de escucha del servidor. Además, integra el enrutador principal de la API para gestionar todas las peticiones entrantes.

El directorio **routes/** define el enrutamiento de la API REST. En este directorio, el archivo api.js es el encargado de realizar la centralización de todos los endpoints disponibles, definiendo los getters de cada uno de los endpoints del sistema. Este enrutador principal actúa como un distribuidor que redirige las peticiones HTTP hacia los controladores correspondientes según la entidad solicitada (conciertos, giras, festivales o ubicaciones).

Los **controllers/** implementan la capa de presentación del backend, donde cada controlador se especializa en una entidad específica del dominio. El controlador de conciertos gestiona todas las operaciones relacionadas con eventos musicales individuales, mientras que el controlador de giras maneja las operaciones de las giras, que agrupan múltiples conciertos. De manera similar a este,

el controlador de festivales se encarga de la gestión de festivales, y el controlador de ubicaciones administra la información geográfica de provincias y poblaciones donde se realizan los eventos.

La capa de **services/** encapsula la lógica de negocio específica de cada entidad del sistema. Estos servicios actúan como intermediarios entre los controladores y los repositorios, implementando las reglas de negocio, validaciones y transformaciones de datos necesarias. Cada servicio se especializa en las operaciones complejas de su dominio, como el cálculo de fechas de eventos o filtrado de conciertos por criterios específicos, entre otros.

El directorio **repositories/** constite en la capa de acceso a datos, proporcionando una abstracción sobre las operaciones con PostgreSQL. Esta capa se encarga de ejecutar las consultas SQL, mapear los resultados a objetos del dominio, y manejar las transacciones de base de datos de manera eficiente y segura.

La carpeta **config/** contiene las configuraciones del servidor, siendo db.js el archivo responsable de establecer y gestionar la conexión con la base de datos PostgreSQL. Este archivo define los parámetros de conexión, el pool de conexiones, y las configuraciones de timeout y reconexión automática.

Finalmente, el directorio **utils/** alberga funciones auxiliares y utilidades que pueden ser reutilizadas por diferentes componentes del servidor.

Estructura Front-End del proyecto Móvil (React-Native)

Respecto a la estructura del proyecto front-end este está desarrollado con React Native utilizando Expo como framework de desarrollo, permitiendo la compilación para múltiples plataformas desde una única base de código.

Los principales directorios en los que se estructura el código del cliente son los siguientes:

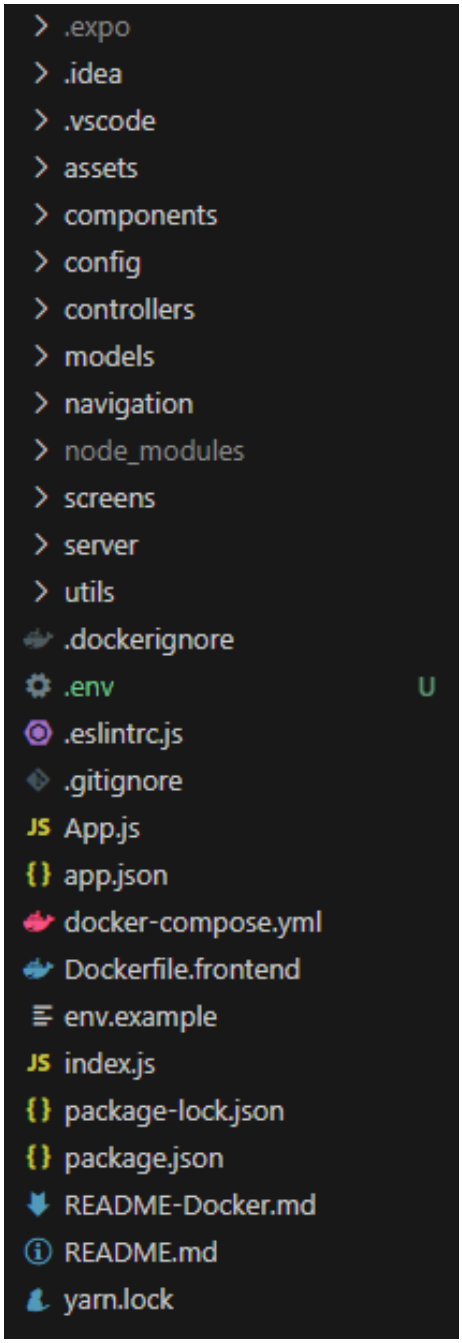


Figura 51: estructura de directorios del frontend de la app-movil

El componente raíz **App.js** actúa como el punto de entrada principal de la aplicación móvil. Este componente gestiona el estado de carga inicial de la aplicación, controla la visualización de la pantalla de bienvenida durante los primeros segundos, y configura el contenedor de navegación principal que envuelve toda la aplicación. Su responsabilidad principal es orquestar el flujo inicial de la aplicación y establecer el contexto de navegación.

El directorio **screens/** contiene las pantallas principales que conforman la interfaz de usuario de la aplicación. La **SplashScreen** proporciona una experiencia de bienvenida profesional mientras se cargan los recursos iniciales de la aplicación. La **HomeScreen** constituye el núcleo de la aplicación, presentando la lista completa de eventos disponibles con opciones de filtrado y búsqueda. La **FavoritesScreen** permite a los usuarios ver su lista de eventos favoritos, proporcionando

acceso rápido a estos.

El sistema de **navigation/** implementa la estructura de navegación de la aplicación utilizando React Navigation. Define la navegación principal y las transiciones entre pantallas, gestionando el stack de navegación y el paso de parámetros entre componentes. El **BottomTabNavigator** implementa la navegación por pestañas inferiores, proporcionando acceso rápido a las secciones principales de la aplicación con iconos intuitivos y estados activos visualmente diferenciados.

Los **components/** reutilizables encapsulan funcionalidades específicas de la interfaz de usuario que pueden ser utilizadas en múltiples pantallas. El **EventDetailModal** es un componente especializado que presenta información detallada de los eventos en un formato modal, incluyendo una serie de atributos determinados.

El directorio **models/** define la estructura de datos y la lógica de negocio del cliente. El **EventModel** encapsula toda la lógica relacionada con la gestión de eventos, incluyendo métodos para formatear fechas, categorizar eventos, gestionar estados de favoritos, y proporcionar interfaces consistentes para la manipulación de datos de conciertos, giras y festivales.

La configuración del cliente se encuentra en config/, donde el archivo api.js define la configuración de conectividad con el backend. Este archivo establece la URL base del servidor, configura los headers por defecto, define todos los endpoints disponibles, y proporciona funciones auxiliares para realizar peticiones HTTP de manera consistente y con manejo de errores centralizado.

Finalmente, el directorio assets/ almacena todos los recursos estáticos necesarios para la aplicación, incluyendo imágenes optimizadas para diferentes densidades de pantalla, iconos vectoriales para una presentación nítida en cualquier resolución, y fuentes personalizadas que definen la identidad visual de la aplicación.

4.2. Patrones empleados

A continuación se definirán los patrones empleados para llevar a cabo el desarrollo de este proyecto. En esta sección se comentarán los patrones empleados por ambos proyectos, tanto móvil como web

4.2.1. Patrón MVC

El patrón Modelo-Vista-Controlador (MVC) es una arquitectura de software que organiza aplicaciones en tres componentes interconectados pero independientes, facilitando la escalabilidad y el mantenimiento de proyectos [22]. Este patrón ha sido ampliamente descrito por Martin Fowler en su obra sobre arquitecturas empresariales [18]. En la siguiente figura 52 podemos observar el esquema del patrón MVC.

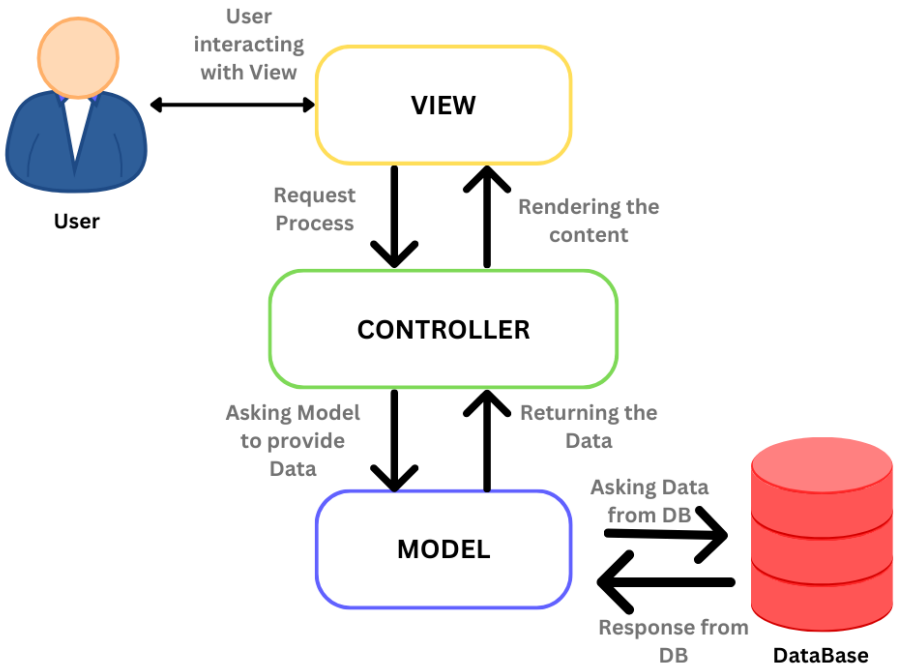


Figura 52: Esquema del patrón MVC [9]

Aquí una descripción detallada:

- **Modelo**: Gestiona los datos y la lógica de negocio (acceso a bases de datos, validaciones, reglas de negocio).
  Es independiente de la interfaz y se actualiza automáticamente cuando cambian los datos.
- **Vista**: Representa la interfaz de usuario (UI). No contiene lógica de negocio, solo presenta datos recibidos del controlador.
- **Controlador** Actúa como intermediario: recibe peticiones del usuario, solicita datos al modelo y los envía a la vista.
  Maneja la navegación y la interacción del usuario sin acceder directamente a los datos

4.2.2. Patron REST API

El patrón arquitectónico REST (Representational State Transfer) define un conjunto de principios para la comunicación entre sistemas distribuidos, especialmente entre cliente y servidor, mediante el protocolo HTTP [39]. Sus principales características son las siguientes:

- **Interfaz uniforme**: Las operaciones se realizan a través de los verbos HTTP estándar (GET, POST, PUT, DELETE), y cada recurso se identifica mediante una URL única.
- **Sin estado (stateless)**: Cada solicitud del cliente debe contener toda la información necesaria para que el servidor la procese. El servidor no almacena el estado de la sesión entre peticiones.
- **Cacheabilidad**: Las respuestas pueden ser almacenadas en cachés intermedios o en el cliente, lo que mejora el rendimiento del sistema y reduce la carga sobre el servidor.
- **Sistema en capas**: La arquitectura puede estar compuesta por múltiples capas (como servidores proxy, balanceadores de carga o firewalls), que funcionan de manera transparente para el cliente.

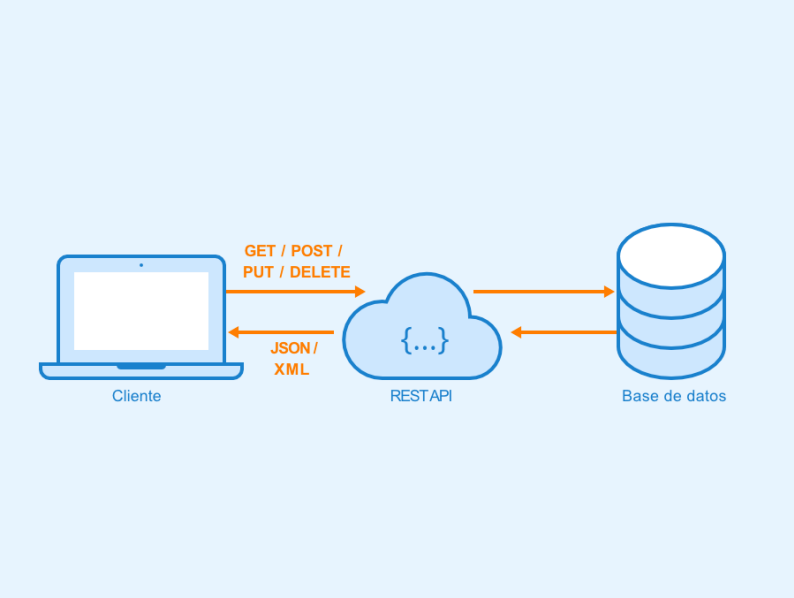


Figura 53: Esquema del patrón API REST [12]

4.2.3. Patrón Controller-Service-Repository

Empleado en el backend de nuestra aplicación móvil, esta arquitectura se caracteriza por la división de responsabilidades en tres capas. Este patrón de capas facilita la mantenibilidad y la escalabilidad del código, separando la gestión de peticiones HTTP, la lógica de negocio y el acceso a datos.

El patrón se implementa mediante una jerarquía de dependencias unidireccional: los controladores dependen únicamente de los servicios, los servicios solo de los repositorios, y los repositorios se comunican directamente con la base de datos PostgreSQL. Esta estructura evita el acoplamiento y asegura que cada capa tenga una responsabilidad específica dentro del sistema.

- **Capa de Controladores — Interfaz de Presentación**: Los controladores forman la capa de presentación del backend, actuando como adaptadores del protocolo HTTP que traducen las peticiones entrantes en operaciones de dominio específicas. Por ejemplo, el controlador `conciertos.js` implementa funciones como `getConciertos` y `getFilteredConciertos`, donde cada función encapsula la lógica necesaria para manejar una petición HTTP particular.
- **Capa de Servicios — Lógica de Negocio**: La capa de servicios encapsula la lógica de negocio del dominio de eventos musicales, implementando operaciones complejas que van más allá del acceso básico a datos. Por ejemplo, el `ConciertoService` contiene métodos como `getAllConciertos` y `getFilteredConciertos`, que implementan reglas de negocio específicas y transformaciones de datos necesarias.
- **Capa de Repositorios — Acceso a Datos** : La capa de repositorios es responsable del acceso directo a la base de datos PostgreSQL, encapsulando las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y las consultas específicas para el dominio de eventos musicales. Esta capa actúa como una abstracción entre la base de datos y la lógica de negocio, proporcionando una interfaz limpia y desacoplada para manipular los datos.

  Por ejemplo, los repositorios implementan métodos que consultan los conciertos almacenados, realizan filtrados en base a parámetros recibidos y gestionan la persistencia de nuevos eventos o modificaciones. Al centralizar el acceso a los datos en esta capa, se facilita el mantenimiento y la posibilidad de cambiar el motor de base de datos en el futuro sin impactar las capas superiores.

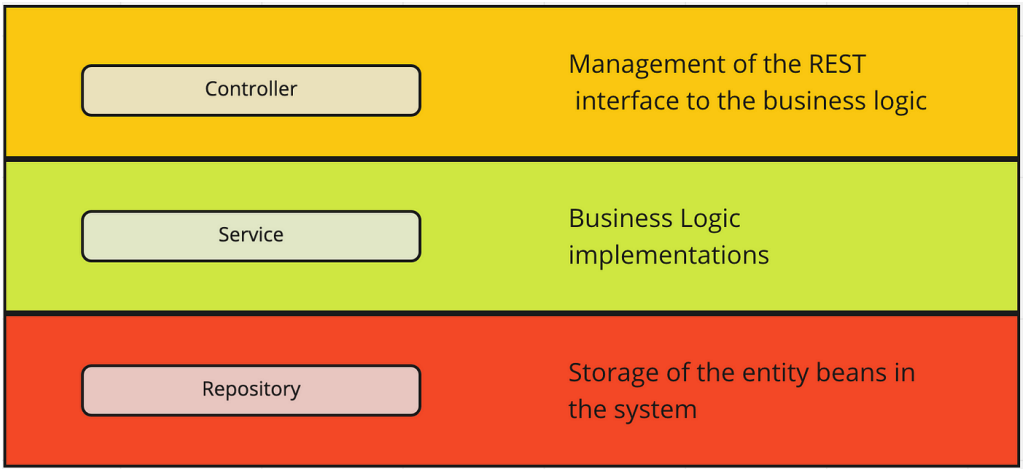


Figura 54: Esquema del patrón controller-service-repository [10]

4.3. Diagrama de paquetes

A continuación se van a presentar los diagramas de paquetes relativos a este proyecto.

Un diagrama de paquetes consiste en una representación visual que muestra cómo se organizan y agrupan los distintos elementos o clases de un sistema en paquetes o módulos. Estos paquetes son contenedores que permiten estructurar el proyecto de manera lógica y manejable.

Es útil para comprender la arquitectura general del sistema, facilitando la identificación de dependencias entre módulos, promoviendo la reutilización de componentes y ayudando a gestionar la complejidad del software al dividirlo en partes más pequeñas y manejables.

4.3.1. Diagrama de paquetes de Back-end de aplicación móvil

En la siguiente figura se muestra el diagrama de paquetes del backend respectivo al backend de la aplicación web.

En este podemos observar que se ha diferenciado la estructura principal de este en tres secciones bien diferenciadas. Como se comentó anteriormente, estas son controllers, services y repositories. La funcionalidad de estas es:

- **controllers**: Actúan como adaptadores de protocolo HTTP que reciben las peticiones web, extraen y validan los parámetros de entrada, coordinan con la capa de servicios para ejecutar la lógica de negocio, y transforman las respuestas en formato JSON con los códigos de estado HTTP apropiados. Su responsabilidad principal es gestionar la comunicación entre el protocolo web y el dominio de la aplicación.
- **services**: Encapsulan toda la lógica de negocio específica del dominio musical, implementando reglas de negocio complejas, transformaciones de datos, procesamiento de imágenes, operaciones de filtrado especializado y orquestación de múltiples operaciones de repositorio. Constituyen el núcleo donde reside la inteligencia del sistema y las reglas específicas del dominio de eventos musicales.
- **repositories**: Proporcionan una abstracción orientada a objetos sobre las operaciones de base de datos PostgreSQL, encapsulando consultas SQL optimizadas, construcción dinámica de filtros, mapeo de resultados hacia objetos del dominio y gestión de operaciones CRUD. Su función principal es aislar completamente la lógica de persistencia del resto del sistema.

Por otro lado encontramos la carpeta **routes/**, la cual contiene api.js la cual es la clase encargada de enrutador principal

para finalizar podemos observar la carpeta **config/** la cual contiene la funcionalidad necesaria para establecer la conexión con la base de datos.

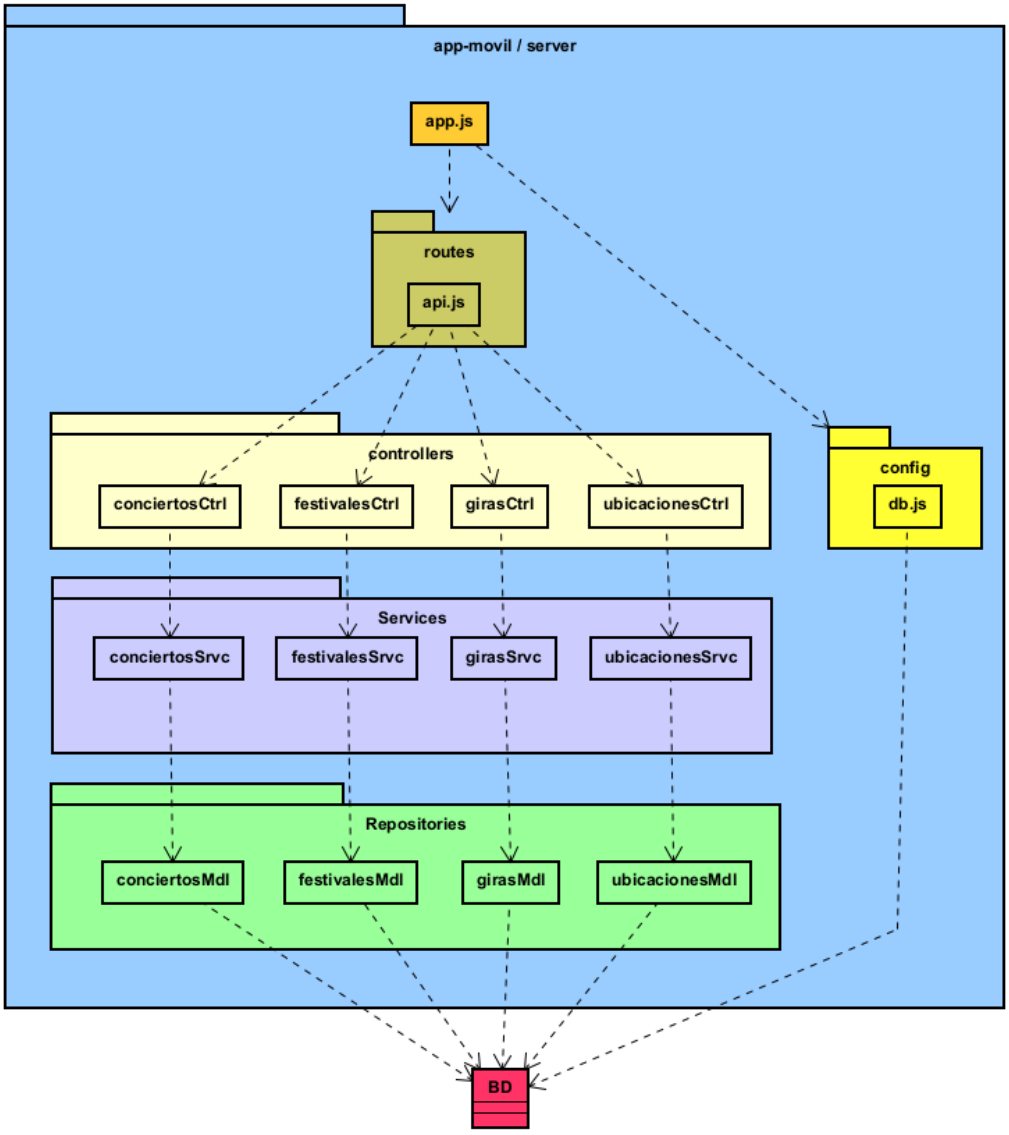


Figura 55: Diagrama de paquetes de backend de aplicación móvil

4.3.2. Diagrama de paquetes de Front-end de aplicación móvil

En la siguiente figura se muestra el diagrama de paquetes del frontend respectivo a la aplicación móvil 56. En este podemos observar que se ha diferenciado la estructura principal de este en cuatro secciones bien diferenciadas. Estas son Views, Controllers, Models y core. La funcionalidad de estas es:

- **Views**: Constituyen la capa de presentación visual de la aplicación móvil, organizándose en subcarpetas especializadas. La subcarpeta Screens contiene las pantallas principales como HomeScreen, FavoritesScreen y SplashScreen que implementan las interfaces de usuario específicas. La subcarpeta navigation gestiona el sistema de navegación entre pantallas utilizando React Navigation, mientras que components alberga componentes re-utilizables de interfaz que pueden ser utilizados por múltiples pantallas.

- **Controllers**: Actúan como intermediarios entre la interfaz de usuario y los modelos de datos, implementando la lógica de control específica del frontend móvil. El EventController gestiona las operaciones de eventos, coordina llamadas a la API, maneja el estado de la aplicación y procesa las interacciones del usuario antes de delegarlas a los modelos correspondientes.
- **Models**: Encapsulan la lógica de negocio del cliente y la gestión de datos locales, proporcionando una abstracción sobre las operaciones de datos específicas del frontend. El EventModel gestiona el estado de eventos, formateo de datos, operaciones con favoritos, persistencia local con AsyncStorage y comunicación con el backend a través de la API REST.

Por otro lado encontramos la carpeta core/, la cual contiene elementos fundamentales del sistema organizados en subcarpetas especializadas. utils alberga utilidades y funciones auxiliares, config contiene las configuraciones de conectividad con el backend incluyendo api.js que centraliza los endpoints REST, y assets almacena los recursos estáticos como imágenes, iconos y fuentes necesarios para la interfaz visual. Para finalizar podemos observar que esta arquitectura implementa un patrón MVC adaptado al desarrollo móvil donde las Views gestionan la presentación, los Controllers coordinan la lógica de interfaz, y los Models manejan tanto los datos como la comunicación con el backend, mientras que core proporciona la infraestructura base del sistema.

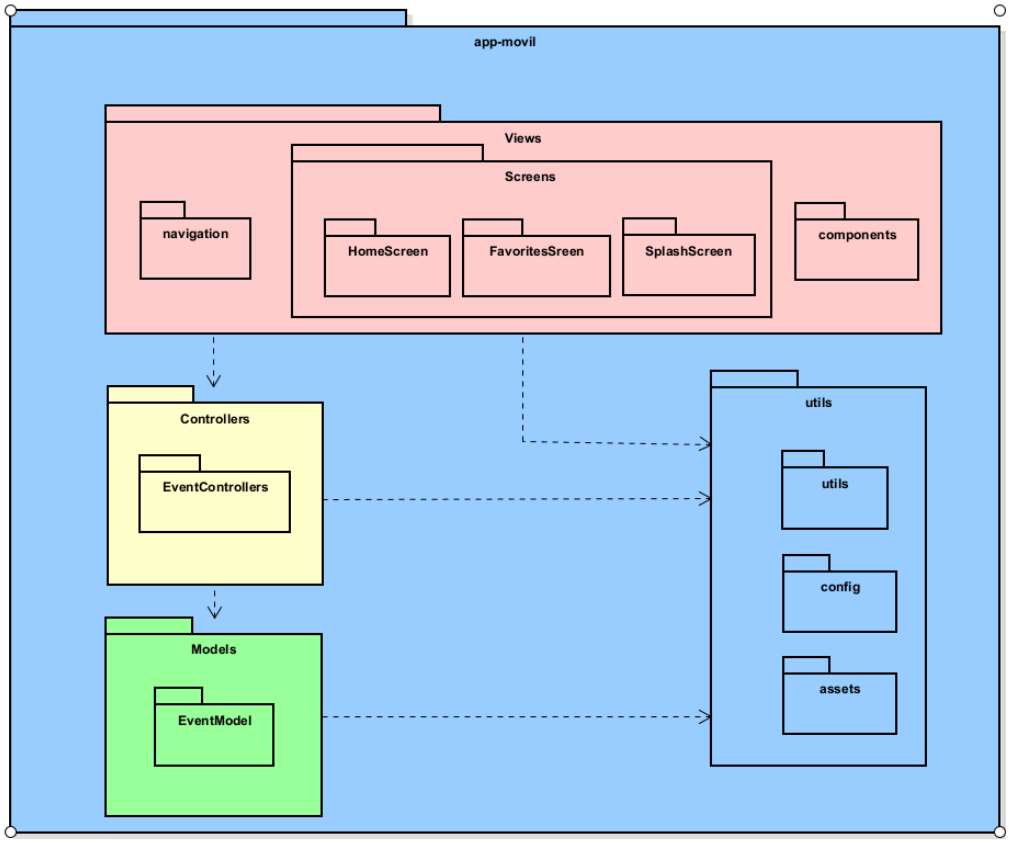


Figura 56: Diagrama de paquetes de frontend de aplicación móvil

4.3.3. Diagrama de paquetes de aplicación web

Respecto al diagrama de paquetes de la aplicación web podemos observar que cuenta con los siguientes directorios:

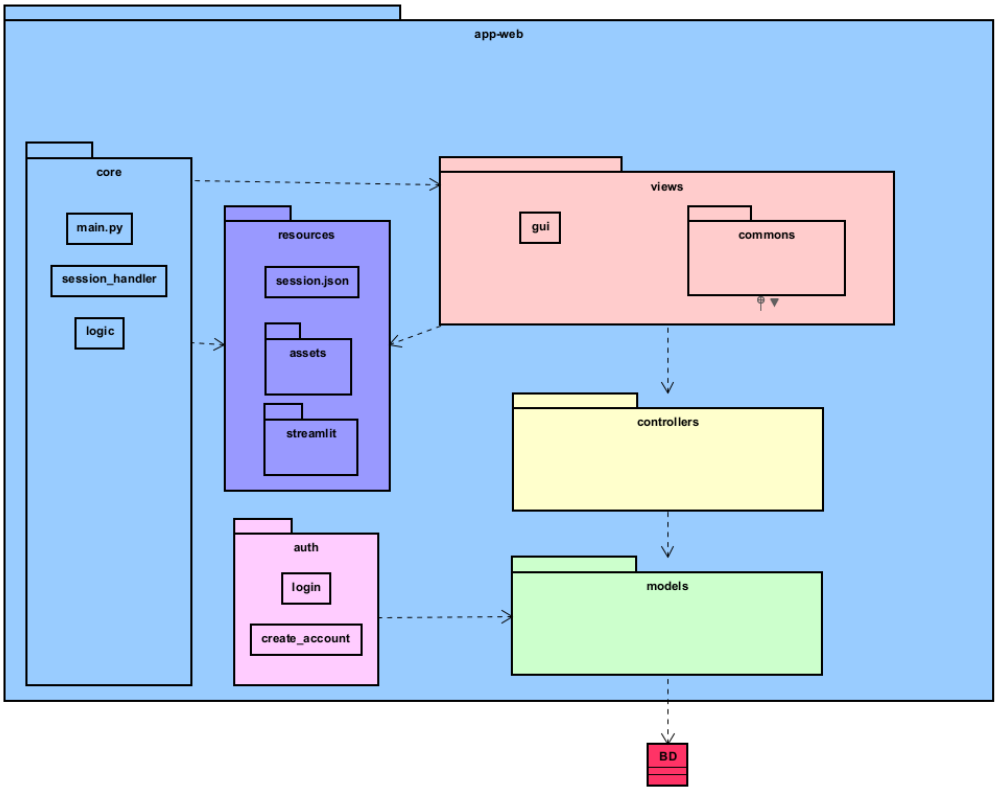


Figura 57: Diagrama de paquetes de aplicación web

El paquete **resources**: contiene las dependencias generadas de la librería **streamlit/**. Ademas de esta podemos observar el directorio **assets/** encargado de la gestión relativa a eventos visuales, en este caso principalmente los logos empleados en la interfaz.

Por ultimo, en este directorio podemos observar el **session.json**, encargado de la gestión de la sesión del usuario.

La carpeta **auth/** constituye el módulo especializado en la gestión de autenticación y autorización de usuarios dentro de la aplicación web administrativa. Este directorio encapsula toda la funcionalidad relacionada con el control de acceso al sistema, implementando mecanismos robustos de seguridad y gestión de sesiones. Dentro de esta tenemos las siguientes funciones:

- El archivo **login** implementa el sistema de autenticación de usuarios existentes
- El archivo **create_account** gestiona el registro de nuevos usuarios implementando un sistema completo de validación y creación de cuentas

El paquete **views** contiene todos los componentes relacionados con la interfaz de usuario y la presentación de datos al usuario final. Esta capa es responsable de definir cómo se muestran los datos de la aplicación.

En el diagrama podemos observar que el paquete views se subdivide en dos secciones principales: *gui*, que contiene la vista principal de la aplicación como home_view, account_settings_view, create_new_account_view, available_events_view, my_events_view, pending_events_view, add_csv_view y add_new_event_view; y *commons*, que agrupa vistas reutilizables como concert_edition, display_concert, tour_edition y festival_edition. Esta organización facilita la reutilización de componentes visuales y mantiene una estructura coherente en la presentación y que podemos observar en detalle a continuación 58.

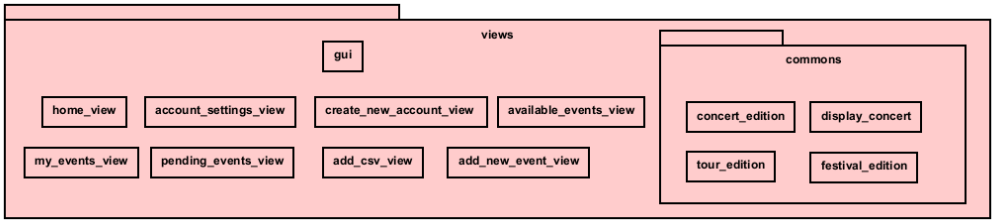


Figura 58: Diagrama de paquetes de carpeta views

El paquete **models** representa la capa de datos y lógica de negocio de la aplicación, siendo el componente que define las estructuras de datos y implementa los métodos necesarios para acceder y manipular la información.

En el diagrama 59 se puede apreciar los diferentes modelos *base_model* , *user_model*, *festival_model*, *concert_model*, *location_model* y *tour_model*. Encargados de tomar los datos de la base de datos.

El modelo es independiente de la interfaz de usuario y se encarga exclusivamente de la lógica del negocio, lo que permite mayor flexibilidad para modificar la presentación sin afectar la lógica.

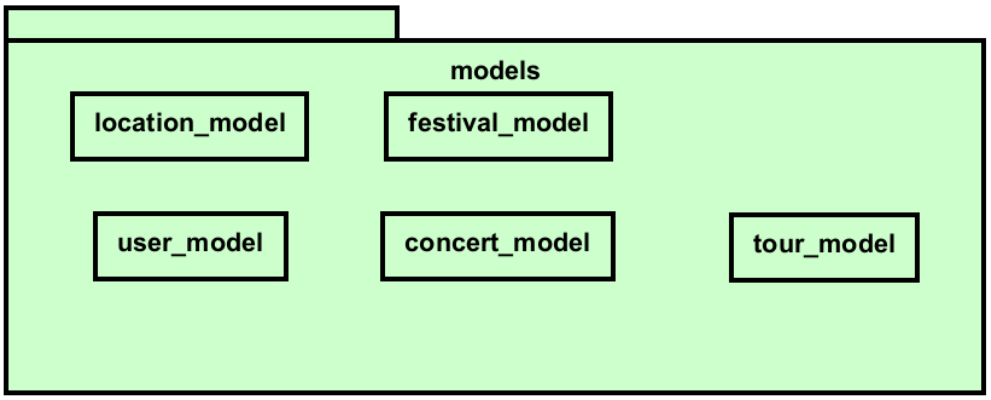


Figura 59: Diagrama del paquete

El paquete **controllers** contiene la lógica que coordina la comunicación entre los modelos y las vistas.
Los controladores son responsables de recibir las solicitudes del usuario, interactuar con los modelos correspondientes para obtener o modificar datos, y determinar qué vista debe presentarse al usuario.

En el dia se observa una amplia gama de controladores específicos: user_controller, create_account_controller, available_events_controller, festival_controllers, location_controller, my_events_controller, pending_events_controller, csv_controller, tour_controller y concert_controller.

Esto se puede observar en la figura 60

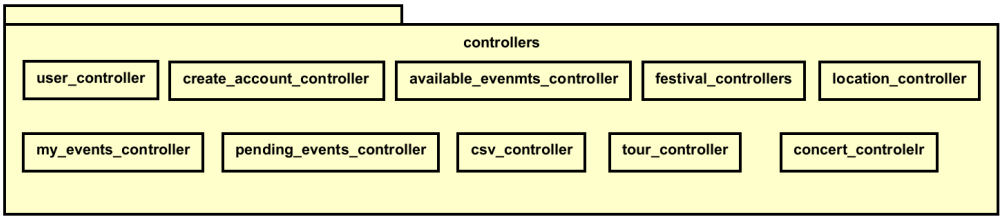


Figura 60: Estructura del paquete controllers mostrando los diferentes controladores organizados por funcionalidad

4.3.4. Diagrama de despliegue de la aplicación completa

Un diagrama de despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado (UML) que se utiliza para representar la disposición física de los artefactos de software en los nodos de hardware de un sistema. Es decir, muestra cómo los distintos componentes de software se distribuyen y ejecutan en dispositivos físicos [31].

El sistema se estructura en cuatro contenedores Docker especializados que operan de manera coordinada a través de una red compartida. El contenedor app-bd ejecuta una instancia de PostgreSQL que actúa como repositorio centralizado de datos, encapsulando toda la información del sistema en un entorno de ejecución aislado y optimizado para operaciones de base de datos.

El contenedor app-web aloja la aplicación administrativa desarrollada en Python con Streamlit, proporcionando una interfaz web completa para la gestión del sistema. Este contenedor incluye un entorno de ejecución Python configurado con todas las dependencias necesarias y expone la aplicación web a través del puerto correspondiente.

La aplicación móvil se despliega mediante dos contenedores especializados que implementan una arquitectura frontend-backend separada. El contenedor app-movil-backend ejecuta un servidor Node.js con Express que proporciona una API REST completa para las operaciones móviles, gestionando la lógica de negocio específica y la comunicación con la base de datos PostgreSQL. El contenedor app-movil-frontend aloja la aplicación React Native configurada con Expo, que puede ser accedida tanto desde navegadores web como dispositivos móviles nativos.

El backend Node.js gestiona la conexión con la base de datos PostgreSQL de forma automática a través de la red Docker compartida y expone una API REST completa que puede ser accedida por los clientes móviles y web a través de peticiones HTTP. Esta API proporciona todos los endpoints necesarios para las operaciones CRUD sobre eventos musicales, gestión de usuarios y funcionalidades de filtrado avanzado.

Los clientes del sistema pueden acceder a los servicios desde múltiples plataformas y dispositivos. Los usuarios de dispositivos Windows pueden acceder a la aplicación web administrativa a través de cualquier navegador moderno, estableciendo conexiones HTTP directas con el contenedor Streamlit. Los usuarios de dispositivos Android pueden ejecutar la aplicación móvil utilizando Expo Go, que se conecta al backend Node.js a través de peticiones HTTP para realizar operaciones de almacenamiento, obtención, actualización y borrado de datos.

La comunicación entre servicios se establece mediante una red Docker privada que permite la resolución de nombres entre contenedores, mientras que la comunicación con clientes externos se realiza a través de puertos expuestos que mapean los servicios internos hacia interfaces de red accesibles. Esta arquitectura garantiza el aislamiento de la lógica interna del sistema mientras proporciona puntos de acceso controlados para los diferentes tipos de clientes.

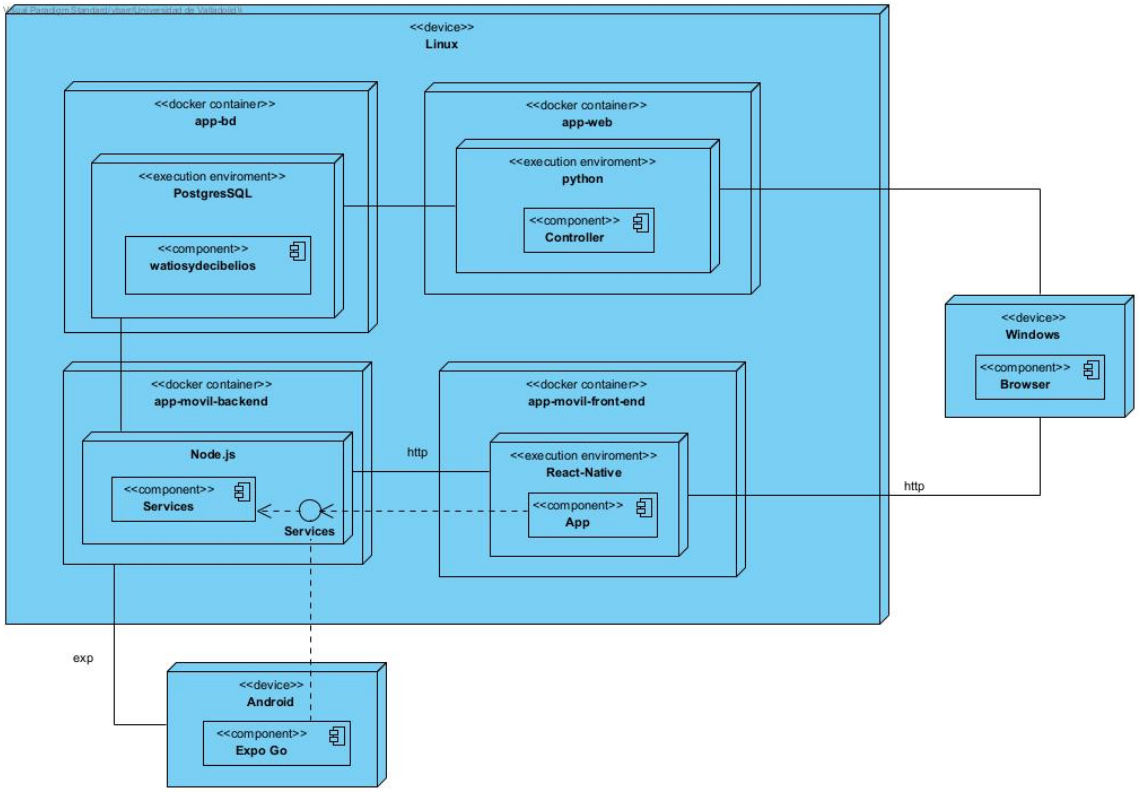


Figura 61: Diagrama de despliegue de todo el proyecto

Capítulo 5

5. Descripción de las fases

5.1. Planificación

En la primera fase, la planificación, el principal objetivo es seleccionar las herramientas y tecnologías que se utilizarán en el proyecto, así como realizar un análisis de los requisitos del sistema. También se comenzará el desarrollo, enfocándose en las primeras funcionalidades y estableciendo las bases para el resto del proyecto.

5.2. Primera iteración

En esta sección se abordará principalmente la creación del proyecto, incluyendo la inicialización de la base de datos. Además, se detallará el desarrollo tanto del back-end como del front-end para ambas partes del proyecto: la aplicación web y la aplicación móvil.

5.2.1. Desarrollo de la base de datos

En primer lugar, se llevó a cabo la conceptualización y creación de la base de datos. Esta etapa ha estado estrechamente relacionada con el análisis de los requisitos de la aplicación, a partir del cual se definió la estructura actual de la base de datos. Para el desarrollo y planificación de esta base de datos, se ha contado con el apoyo de referencias especializadas en el ámbito del diseño y gestión de bases de datos, tales como los trabajos de Ramakrishnan y Gehrke [38] y el curso práctico avanzado de PostgreSQL de Borja Orbegozo Arana [3]. La base de datos está compuesta por las siguientes tablas y su diagrama relacional 62, elaborado con la herramienta DBeaverCE, se muestra a continuación.

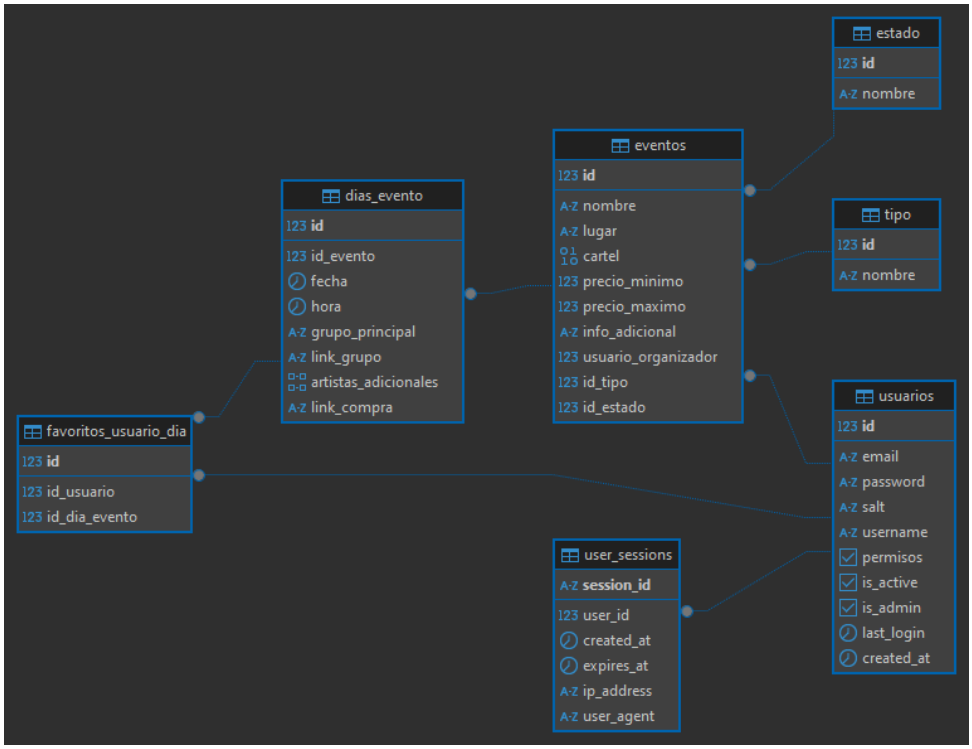


Figura 62: Diagrama relacional de las tablas de la base de datos.

Las tablas de la base de datos son las siguientes:

- **eventos**: Contiene la información principal de cada evento, como nombre, lugar, cartel, precios, información adicional, el usuario organizador y las claves foráneas al tipo y estado del evento.
- **dias_evento**: Registra los distintos días asociados a un evento, incluyendo la fecha, hora, grupo principal, enlaces relacionados y artistas adicionales.
- **favoritos_usuario_dia**: Tabla intermedia que almacena los días de eventos marcados como favoritos por cada usuario.
- **estado**: Define los posibles estados de un evento (por ejemplo, Disponible, Propuesto, etc.).
- **tipo**: Contiene los diferentes tipos o categorías de eventos disponibles.
- **usuarios**: Guarda los datos de los usuarios registrados, como email, contraseña, nombre de usuario, permisos, y otra información relevante para la autenticación y gestión de usuarios.
- **user_sessions**: Almacena las sesiones activas de los usuarios, incluyendo información como fecha de creación, expiración, dirección IP y agente de usuario.

5.2.2. Desarrollo de la aplicación Web (Python + Streamlit)

Una vez definida la base de datos que gestionará la capa de persistencia, se procede al desarrollo de la aplicación web. Esta decisión se debe a que la aplicación web será la encargada principal de gestionar la creación y eliminación de usuarios y eventos, por lo que resulta prioritario enfocarse primero en su desarrollo.

Entre las funcionalidades esenciales que deben implementarse desde el inicio se encuentran el inicio de sesión para usuarios registrados, así como la creación de nuevas cuentas y eventos.

En este caso y como se definió con anterioridad en los requisitos funcionales para el usuario administrador es mas conveniente que sea el el que cree las cuentas desde su interfaz de usuario, ya que así limita los usuarios que pueden interactuar con el sistema de gestión, facilitando así la misma. A continuación se mostrara la implementación de la vista de Log-in de nuestra interfaz Web 63

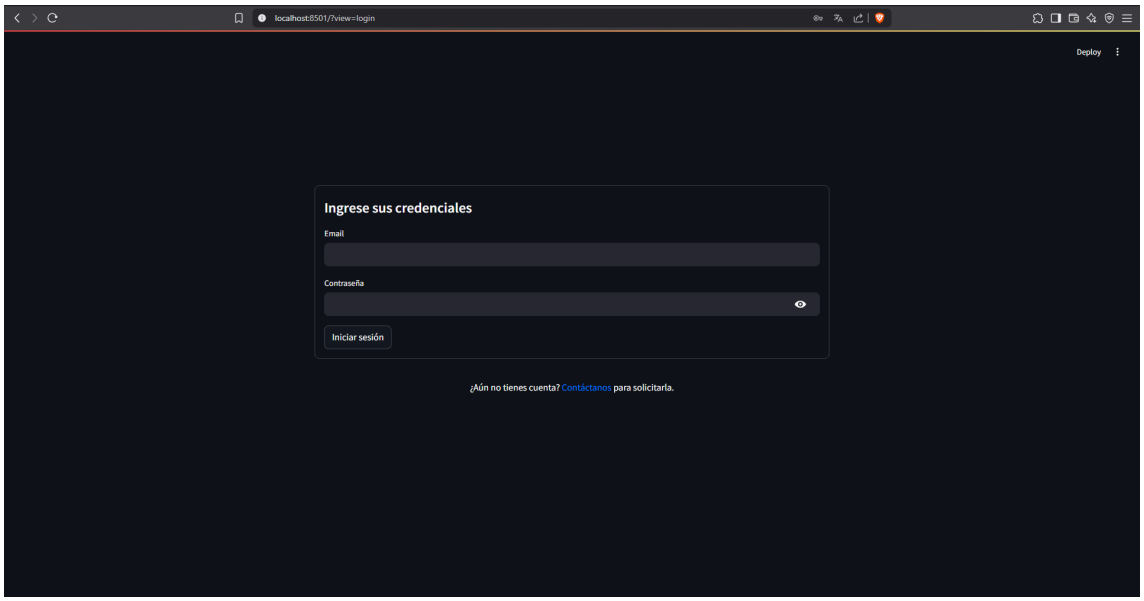


Figura 63: Vista de log-In desarrollada con Streamlit.

Posteriormente, se abordará el desarrollo del panel de usuario, que presentará distintas opciones según los permisos del usuario, diferenciando entre administradores y usuarios externos. En ambos casos, se incluirá una barra lateral (Side-Bar) que facilitará la navegación entre las diferentes funcionalidades disponibles para cada tipo de usuario.

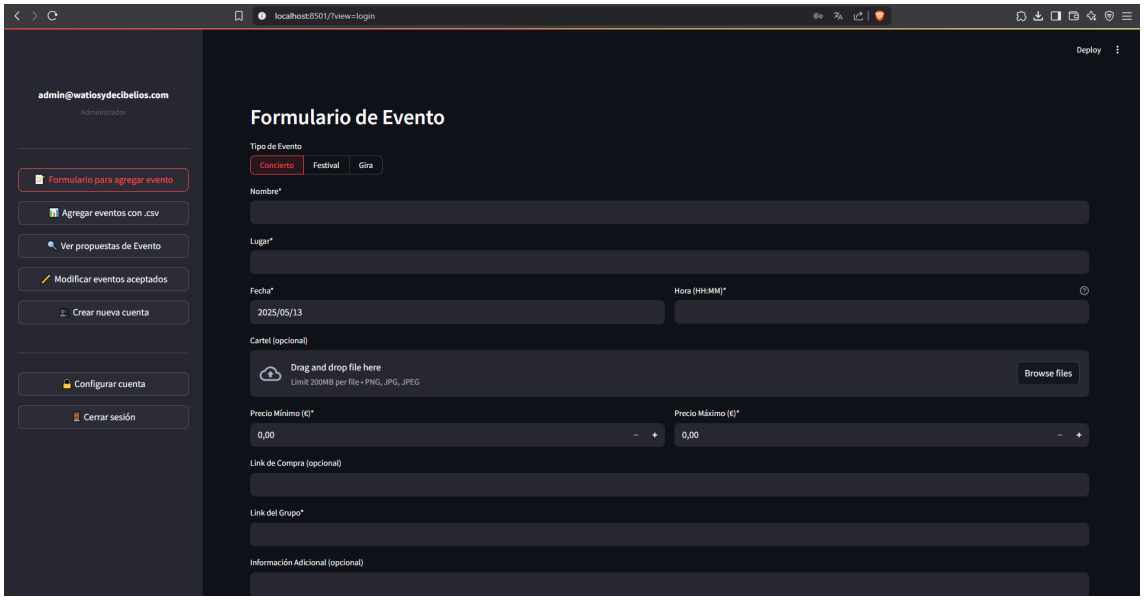


Figura 64: Vista de Administrador desarrollada con Streamlit.

En esta vista podemos observar las opciones de **crear un evento**, agregar un evento desde un .csv, ver las propuestas de evento de otros usuarios externos (que recordemos que necesitaban la aceptacion de un usario administrador para pasar a estar disponibles), modificar eventos aceptados, **crear nuevas cuentas** y las opciones de configuracion y **salida de la cuenta**, pero en esta iteración nos centraremos en las funcionalidades resaltadas.

En el panel principal podemos observar un segmented_control [44], el cual funciona a modo de radio_button y permite al usuario elegir que tipo de evento quiere agregar eligiendo entre los tres tipos de evento definidos: concierto, festival y gira.

posteriormente podemos observar un formulario que contiene todos los datos necesarios requeridos para organizar un evento. En función de la selección del segmented_control el formulario nos ofrecerá un formulario u otro. Los formularios entre concierto y Gira son iguales, y la diferencia entre estos y los festivales es que los festivales pueden tener más de un día asociados.

Respecto a la sección de creación de una nueva cuenta podemos observar un formulario con cuatro campos, nombre de usuario, correo electrónico, contraseña y contraseña repetida.
En este caso el administrador introduce los datos de los nuevos usuarios que quiere y comprueba que las contraseñas coinciden. En caso de que así sea la cuenta queda generada.

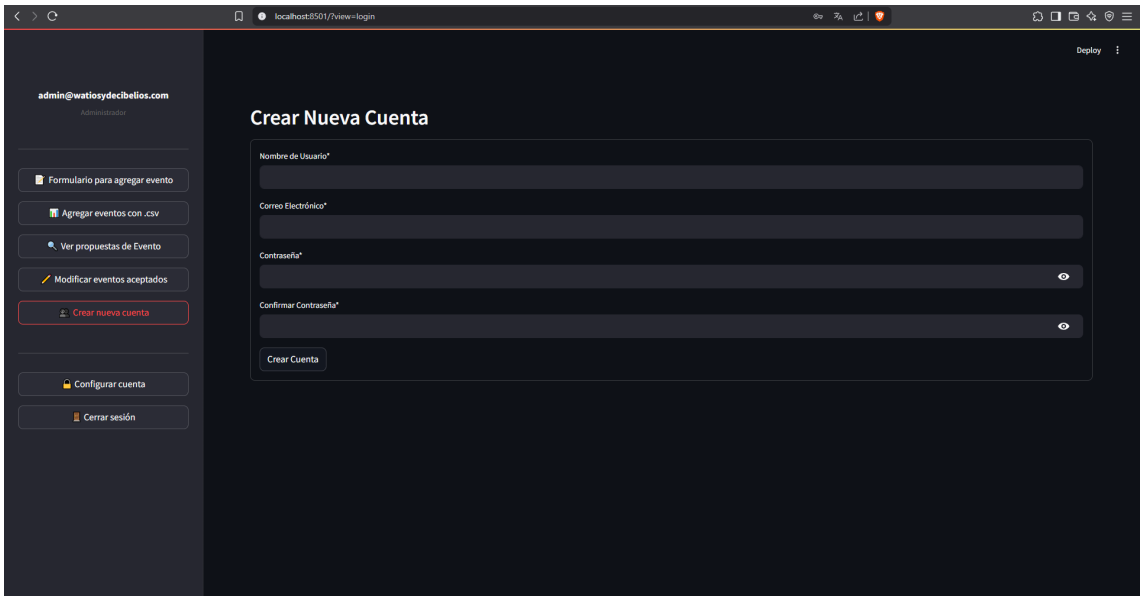


Figura 65: Vista de creacion de usuario desarrollada con Streamlit.

Cabe destacar que a la hora de generar la cuenta se revisan que los datos sean correctos, en caso de serlo se procede a crear el usuario en la base de datos. Para la seguridad de las contraseñas, utilizo la librería *bcrypt* [6], que permite generar hashes seguros mediante el añadido de un "salt.ªleatorio, protegiendo así las credenciales de los usuarios incluso en caso de brecha de seguridad.

Componentes principales de Streamlit utilizados

La interfaz de usuario se desarrolló usando principalmente los siguientes componenetes de Streamlit:

Elementos de captura de datos:

- **st.text_input**: Implementado para la recogida de datos textuales breves.

- **st.text_area**: Empleado para la recogida de descripciones extensas.
- **st.form**: Utilizado para estructurar conjuntos de campos relacionados, permitiendo la validación y envío simultáneo de múltiples entradas, ideal para los formularios que hemos implementado.
- **st.button**: Representwa un boton, encargado de desencadenar acciones específicas.
- **st.file_uploader**: Implementado para facilitar la incorporación de archivos al sistema, como imágenes o documentos adjuntos.

Componentes de visualización:

- **st.dataframe**: Aplicado para representar conjuntos de datos tabulares de manera interactiva, facilitando la exploración de registros.
- **st.metric**: Empleado para destacar indicadores clave con sus correspondientes valores y variaciones.
- **st.columns**: Utilizado para estructurar la disposición espacial de elementos relacionados, mejorando la organización visual.
- **st.markdown**: Implementado para presentar contenido textual con formato personalizado mediante Markdown.

Gestión del estado de aplicación:

- **st.session_state**: Fundamental para mantener la persistencia de datos entre interacciones, permitiendo la conservación del contexto de usuario y estados de edición.

Elementos de navegación:

- **st.sidebar**: Implementado como elemento estructural para albergar los controles de navegación y opciones principales del sistema.
- **st.segmented_control**: Aplicado para presentar opciones mutuamente excluyentes de forma intuitiva.

Notificaciones al usuario:

- **st.success**: Utilizado para comunicar finalizaciones exitosas de operaciones.
- **st.warning**: Implementado para alertar sobre situaciones potencialmente problemáticas.
- **st.error**: Aplicado para notificar errores o fallos en procesos críticos.

Personalización visual:

- **st.markdown** con el parámetro `unsafe_allow_html=True`: Empleado estratégicamente para implementar estilos personalizados mediante código HTML y CSS, extendiéndose más allá de las limitaciones estilísticas predefinidas.

5.2.3. Implementación de Interfaces para visualizar Eventos

A continuación, se expone la sección correspondiente a la vista de la aplicación destinada a dispositivos móviles. En esta iteración se ha implementado la funcionalidad principal, que consiste en la visualización de los eventos disponibles.

Asimismo, se ha establecido la conexión con la base de datos, a través de la cual se obtienen los datos relativos a los eventos. Esta operación se realiza mediante el endpoint “Eventos”, donde se lleva a cabo una consulta de los atributos de interés relacionados con los conciertos.

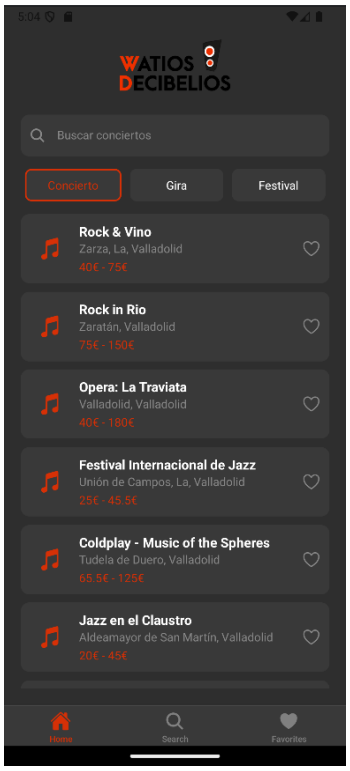


Figura 66: Sección de visualización de los conciertos

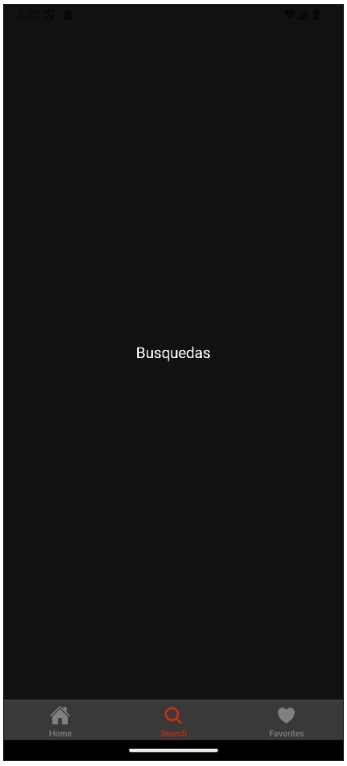


Figura 67: Sección de visualización de búsquedas

Figura 68: Sección de visualización de favoritos

En cada una de las tarjetas de visualización de los eventos [Los pequeños paneles en los que se muestra la info de los eventos] se muestra el **nombre del evento**, la **localidad y provincia** en la que se celebran los conciertos, así como el **rango de precios** de dichos eventos.

Cabe destacar que estos paneles de visualización son idénticos para los tres tipos de eventos: **Conciertos, Giras y Festivales**.

En la zona superior de la vista Home podemos observar que se ha implementado un bloque de búsqueda para realizar búsquedas sobre los campos de nuestros eventos. En este caso aun no se ha implementado todavía la funcionalidad de esta.

En la siguiente figura 27 podemos observar las tareas planteadas para un principio de esta iteración y si estas se han llevado a cabo. podemos observar que en esta iteración se han desarrollado todas las tareas planteadas en un principio.

Inicializar infraestructura BD	Sí
Inicializar infraestructura para AppWeb	Sí
Inicializar infraestructura para AppMovil	Sí
Identificación de usuario	Sí
ver eventos disponibles	Sí
crear cuenta	Sí
cerrar sesion	Sí
proponer eventos	Sí
añadir eventos	Sí
aceptar eventos	Sí
rechazar eventos	Sí

Cuadro 27: Comparación entre trabajo esperado y realizado para la 1ª Iteración

5.3. segunda iteración

5.3.1. Modificaciones respecto al planteamiento anterior

En esta segunda iteración del proyecto, hemos llevado a cabo una reestructuración fundamental de la base de datos que refleja un cambio de paradigma significativo en la concepción de nuestra aplicación. Los cambios implementados no solo mejoran la organización de los datos, sino que también proporcionan una base más sólida y escalable para el futuro desarrollo del sistema.

5.3.2. Evolución del Modelo Conceptual de Eventos

El cambio más importante en esta versión tiene que ver con cómo hemos repensado la estructura básica de los eventos musicales. En la versión anterior, nuestro enfoque era bastante más simple: teníamos únicamente una tabla llamada .^evento"que funcionaba como el lugar donde guardábamos toda la información de los conciertos. Esta tabla central almacenaba los datos importantes, mientras que para manejar la complejidad de las giras y festivales que duran varios días, usábamos una tabla adicional que se encargaba de los diferentes días y los artistas que tocaban cada jornada.

Sin embargo, después de reflexionar más sobre cómo funcionan realmente estos eventos musicales, hemos adoptado una perspectiva mucho más rica y realista. Ahora entendemos las giras y los festivales como conjuntos de conciertos individuales, lo que representa un cambio fundamental en nuestro modelo de datos. Esta nueva forma de verlo reconoce que tanto las giras como los festivales son, básicamente, grupos de eventos musicales independientes que comparten ciertas características organizativas y temáticas, pero que mantienen su propia identidad.

Este enfoque nos permite manejar de forma más natural la complejidad de eventos que se desarrollan durante varios días, en diferentes ubicaciones o que involucran distintas configuraciones de artistas. El nivel de detalle que ahora conseguimos es mucho más preciso y se ajusta mejor a la realidad brindándonos una ligera libertad similar a la del WordPress.

5.3.3. Refinamiento en la Gestión de Ubicaciones

Otro aspecto que hemos mejorado considerablemente es la forma en que manejamos la información de dónde se realizan los eventos. En la versión anterior, dábamos demasiada libertad a los usuarios a la hora de definir dónde se llevarían a cabo los eventos, lo que causaba inconsistencias, información duplicada y una falta de uniformidad que complicaba las búsquedas y el análisis de datos.

Para solucionar este problema, hemos implementado una estructura más organizada y controlada que divide la información de ubicación en tres niveles bien definidos. Primero, definimos el "lugarçomo el espacio físico específico donde se realizará el evento, que puede ser un recinto, bar, plaza, auditorio o cualquier otro tipo de instalación. Segundo, establecemos la "poblaciónçomo el municipio donde se encuentra ubicado ese lugar. Por último, incluimos la "provincia.^en la que se llevara a cabo.

Actualmente, nuestro sistema está configurado para funcionar específicamente en la provincia de Valladolid, pero la estructura ha sido diseñada pensando en crecer en el futuro ya que en caso de querer implementar la funcionalidad para otras provincias valdría con insertar las nuevas provincias de interés y poblaciones pertenecientes a esta.

A continuación se puede observar el diagrama relacional en el que se pueden observar todos los atributos de las nuevas tablas en profundidad.

5.3.4. Implementación de Interfaces para la Gestión de Eventos

En esta sección se detallarán únicamente aquellas interfaces que han sido modificadas de manera notable o que representan desarrollos completamente nuevos en esta segunda iteración del

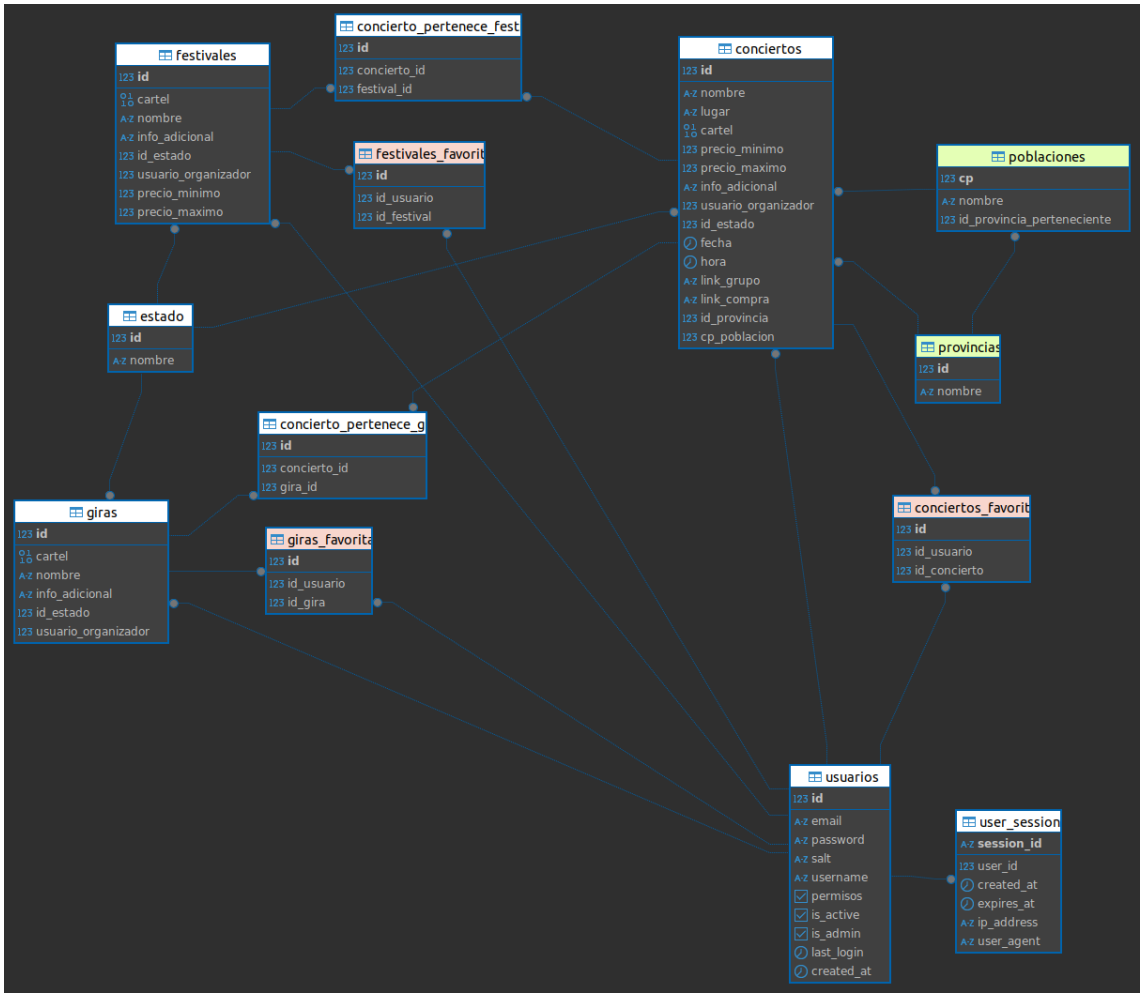


Figura 69: Diagrama entidad-relación de la segunda iteración de la base de datos

sistema. El objetivo es documentar los cambios más significativos en la experiencia de usuario y las decisiones de diseño que han guiado la implementación de estas nuevas funcionalidades.

Gestión Mejorada de Localización: El cambio más importante en la creación de conciertos ha sido la implementación de un sistema de localización estructurado. Ahora utilizamos dos cuadros combinados para representar la información geográfica: el primero para seleccionar la provincia (actualmente Valladolid) y el segundo para elegir la población específica. Complementamos esta información con un campo de texto libre para especificar el lugar exacto del evento.

Sistema de Gestión de Precios: Hemos incorporado un slider de rango que permite establecer precios mínimo y máximo de manera intuitiva. Además, incluimos una checkbox para eventos gratuitos que simplifica la marcación de conciertos sin coste de entrada, eliminando la necesidad de establecer precios en cero.

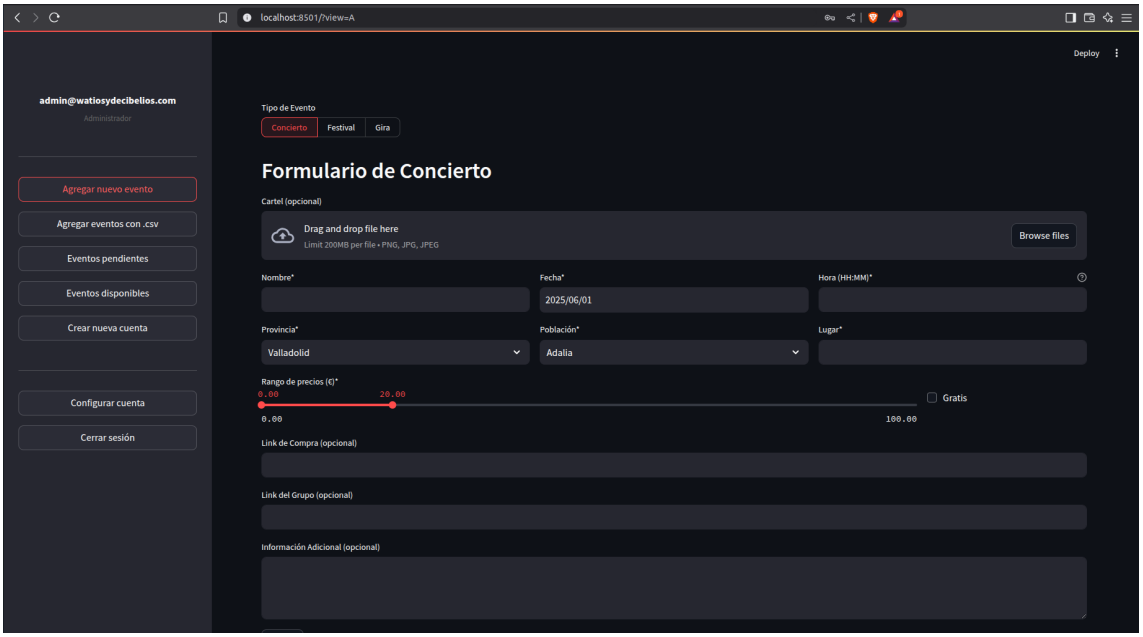


Figura 70: Vista de la interfaz de creación de conciertos

Diseño de creación de giras y festivales

La implementación de ambos formularios utiliza componentes previamente desarrollados, manteniendo coherencia con el sistema. Destaca el uso de **st.multiselect** para seleccionar conciertos asociados, mostrando un listado filtrable que facilita agregar múltiples eventos mediante una interfaz intuitiva.

La diferencia principal radica en que los festivales incluyen **gestión de precios** con rango mínimo/máximo, mientras las giras heredan los precios de cada concierto individual. Esto refleja necesidades comerciales distintas entre ambos tipos de eventos.

A continuación se muestra la interfaz de creación de giras 71 y la interfaz de creación de festivales respectivamente 72

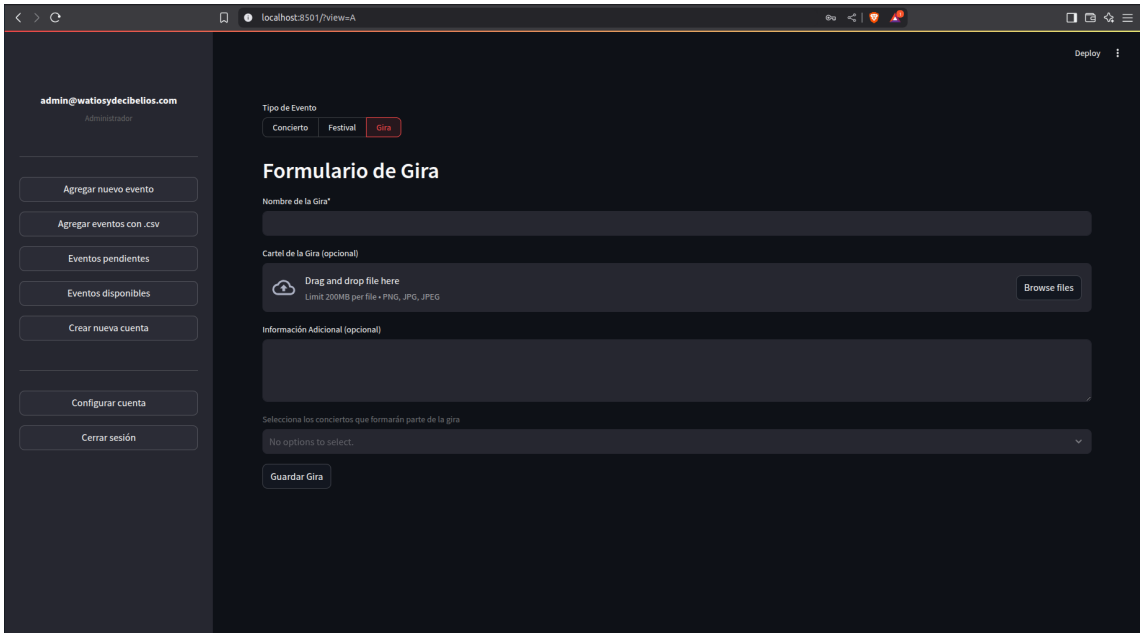


Figura 71: Vista de la interfaz de creación de giras

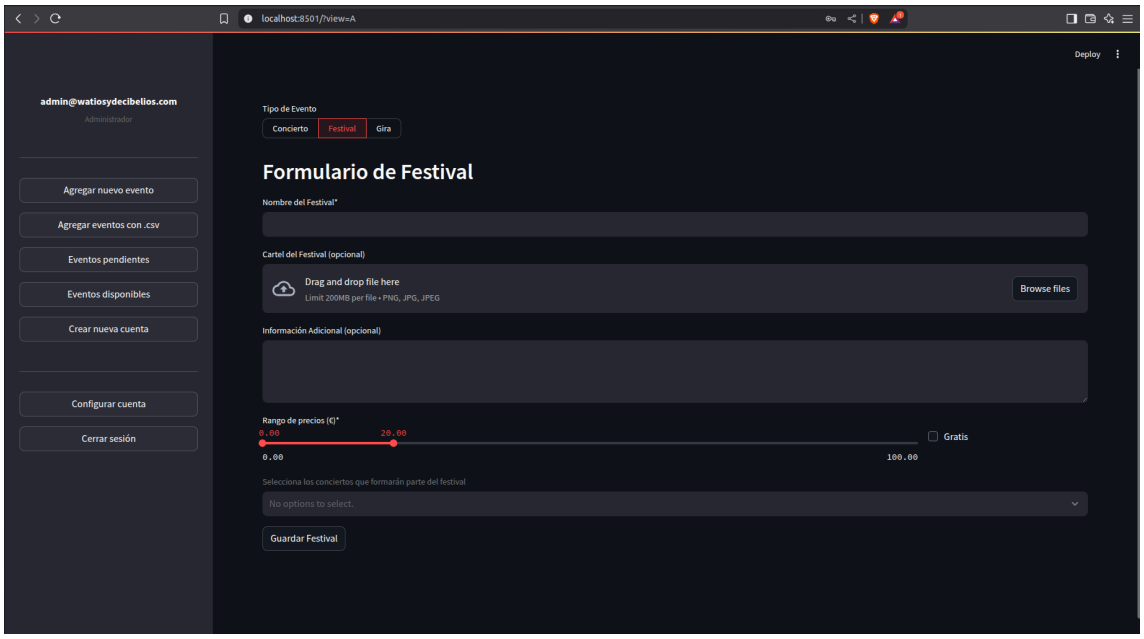


Figura 72: Vista de la interfaz de creación de festivales

Otra de las funcionalidades principales incorporadas en esta iteración es la posibilidad de subir conciertos de forma multiple mediante un archivo CSV o XLSX. Para facilitar este proceso, la interfaz proporciona una plantilla que muestra la estructura exacta que debe seguir el archivo, ayudando así a los usuarios a preparar correctamente sus datos antes de la importación. Esta mejora agiliza considerablemente la gestión de grandes volúmenes de conciertos y reduce la probabilidad de errores durante la carga. También existe una sección en el código la cual nos muestra

las provincia y poblaciones disponibles en nuestro sistema de gestión.

En la siguiente figura se muestra la interfaz 73

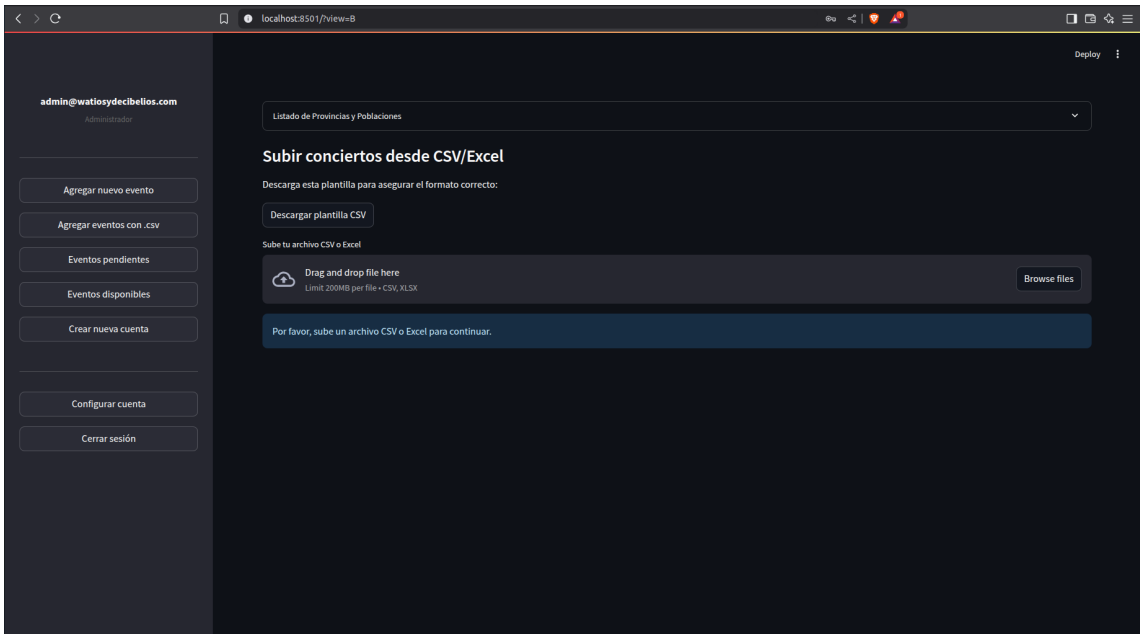


Figura 73: Vista de la interfaz de adición de conciertos de forma múltiple

5.3.5. Implementación de Interfaces para visualizar Eventos

En esta iteración se han finalizado las funcionalidades propuestas inicialmente. En comparación con la iteración anterior, los cambios a nivel de diseño han sido mínimos.

El trabajo se ha centrado principalmente en desarrollar la funcionalidad que permite almacenar eventos como favoritos de forma local y eficiente, así como en implementar la vista correspondiente para su gestión. Además, se ha añadido la opción para que los usuarios puedan modificar las credenciales de su cuenta en la sección de configuración.

Por otro lado, la principal novedad funcional consiste en que, al crear nuevos usuarios, el administrador tiene ahora la posibilidad de asignarles permisos de administrador. Esta mejora facilita las tareas de administración y gestión de los conciertos.

Respecto a la comparación entre tareas planificadas y realizadas

filtrar eventos	Filtro simple
Añadir eventos a favoritos	Sí
Eliminar eventos de favoritos	Sí
ver eventos favoritos	Parcialmente
modificar eventos propuestos	Sí
modificar eventos	Sí
importar conciertos de forma masiva	Sí

Cuadro 28: Comparación entre trabajo esperado y realizado para la 2ª Iteración

5.4. tercera iteración

Modificaciones respecto al planteamiento anterior

En esta iteración, los cambios introducidos han sido menores. A diferencia de la iteración anterior, las modificaciones realizadas en esta fase han sido mínimas, ya que no se han producido variaciones conceptuales significativas. El principal ajuste ha consistido en la eliminación de las tablas destinadas a almacenar los eventos favoritos de los usuarios, dado que carecen de sentido en el contexto actual.

Estas tablas estaban diseñadas para guardar los eventos marcados como favoritos por los usuarios. Sin embargo, una de las premisas fundamentales del proyecto es que los usuarios de la aplicación móvil no necesiten identificarse para utilizarla. Como consecuencia, al no existir un modelo de usuario en la base de datos, la presencia de dichas tablas resulta innecesaria. En su lugar, la gestión de los eventos favoritos se realizará de forma local en el dispositivo, almacenando los identificadores de los diferentes eventos seleccionados por el usuario

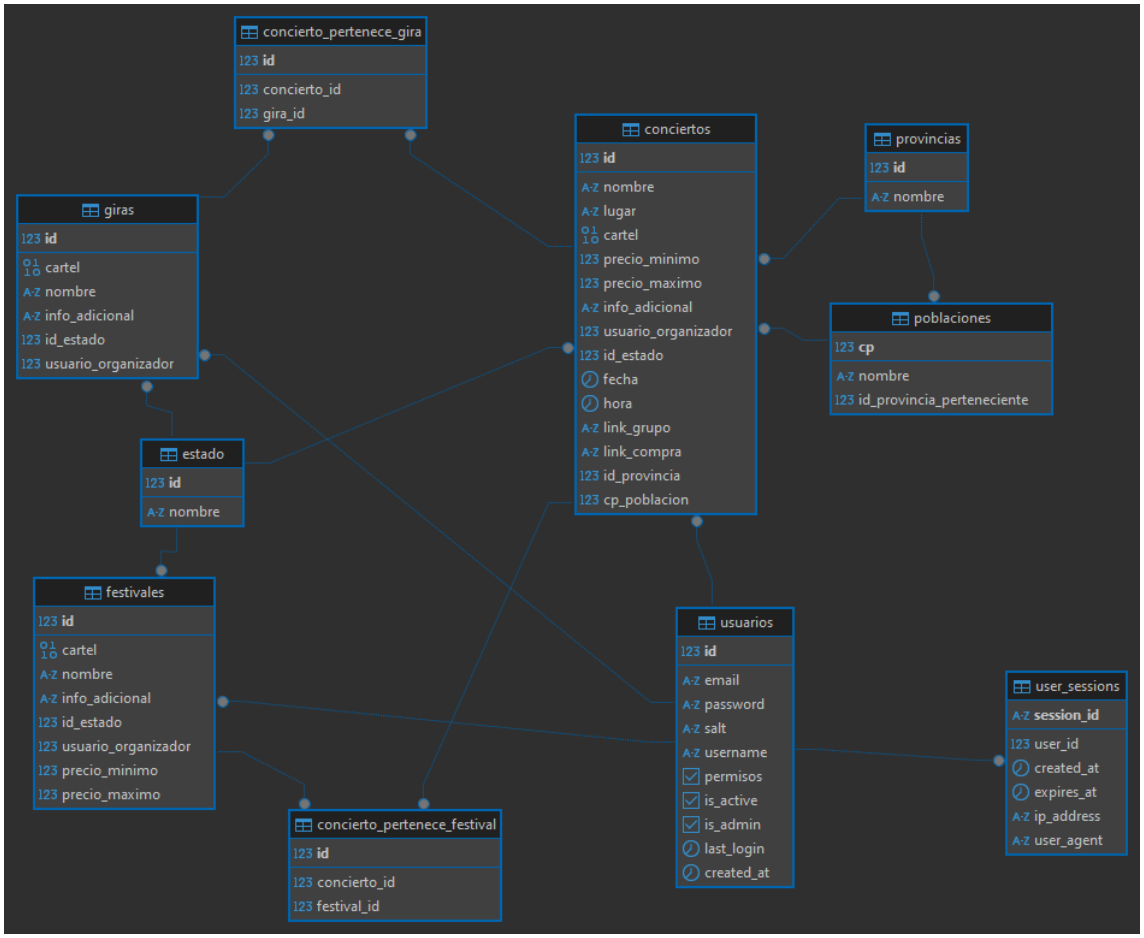


Figura 74: Diagrama relacional de la base de datos en la Iteración 3

En la figura 74 se puede observar que el diagrama relacional ha sufrido una simplificación en comparación con el de la iteración anterior.

Implementación de Interfaces para la Gestión de Eventos

Las principales implementaciones realizadas en esta iteración de la aplicación de gestión han sido las siguientes:

En todos los campos de entrada destinados a valores monetarios, se ha reemplazado el control deslizante (slider) por dos campos numéricos independientes. Esta modificación tiene como objetivo proporcionar un rango de precios considerablemente más amplio y flexible, permitiendo a los administradores establecer valores económicos con mayor precisión y sin las limitaciones inherentes de los controles deslizantes.

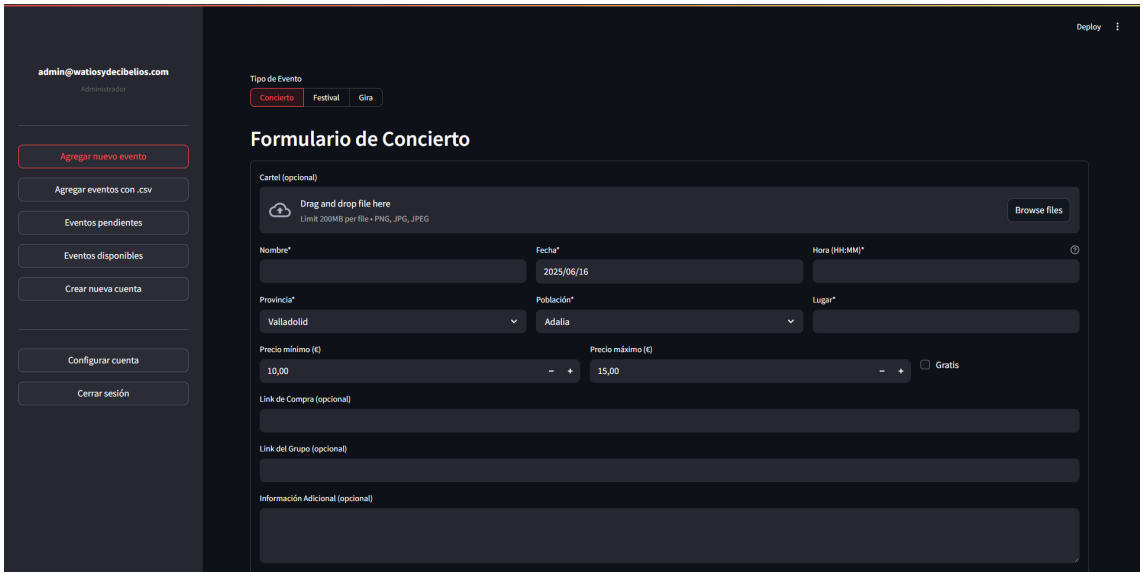


Figura 75: Vista de la interfaz de formulario de concierto actualizado

Por otra parte, en la interfaz destinada a la creación de nuevos usuarios, se ha incorporado una casilla de verificación (checkbox) que permite a los administradores asignar permisos de administrador a los nuevos usuarios durante el proceso de registro. Esta funcionalidad facilita la gestión de roles y permisos desde el momento de la creación de las cuentas, optimizando el flujo de trabajo administrativo.

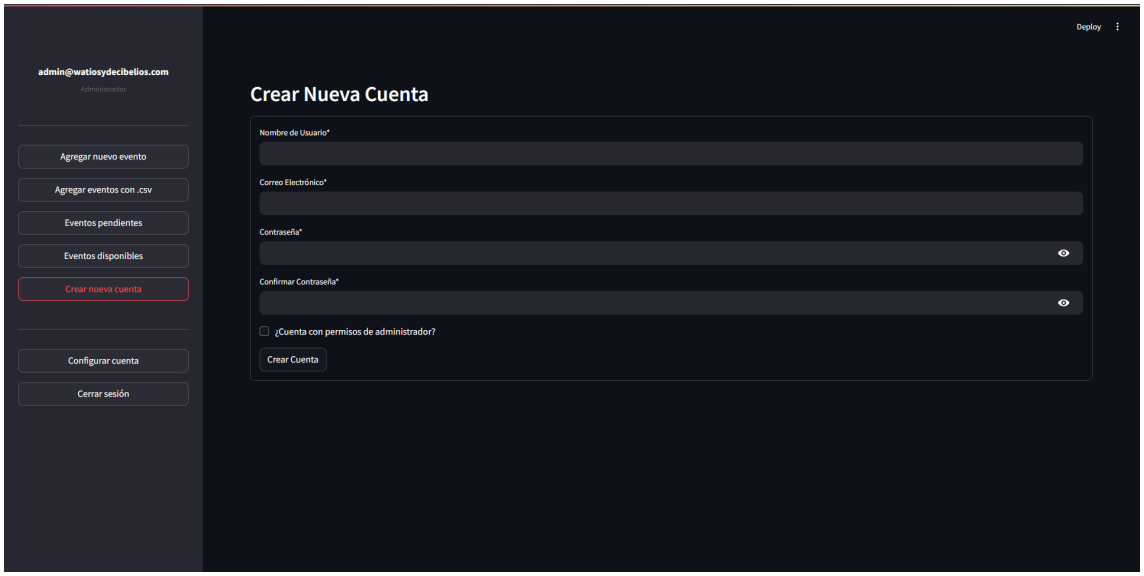


Figura 76: Vista de la interfaz de creación de cuenta actualizado

Además, se ha implementado una funcionalidad destinada a permitir que los usuarios modifiquen sus credenciales de acceso. La función de modificación de credenciales permite a los usuarios actualizar tanto su nombre de usuario como su contraseña de forma segura, garantizando que mantengan el control sobre el acceso a sus cuentas

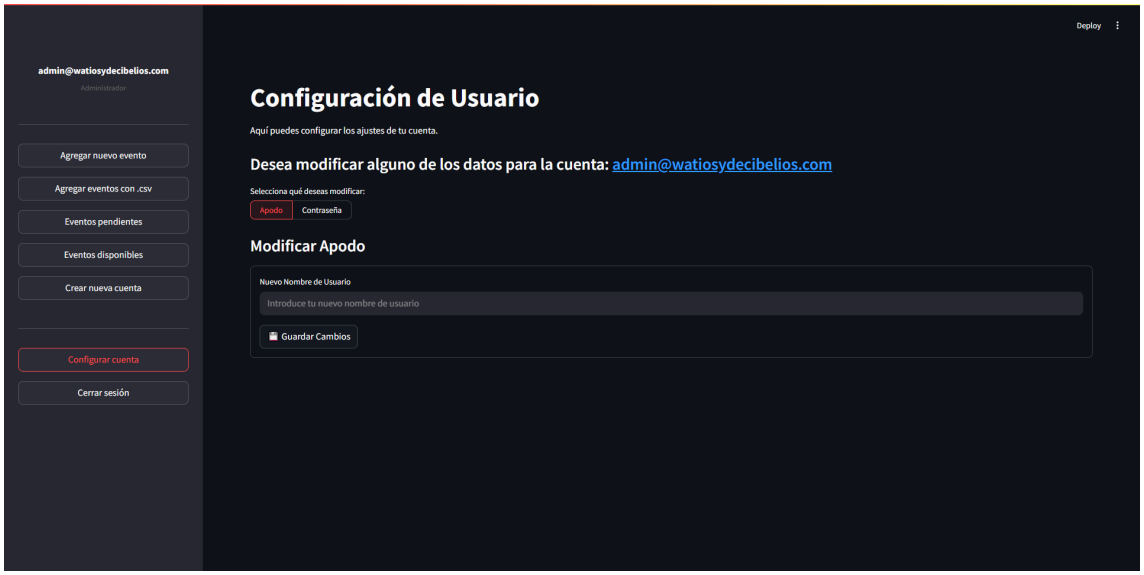


Figura 77: Vista de la interfaz de configuración de usuario

5.4.1. Implementación de Interfaces de aplicación Movil

Respecto a la aplicación para dispositivos móviles, se han implementado diversas modificaciones significativas que mejoran tanto la funcionalidad como la experiencia de usuario.

El primer cambio fue quitar la sección de búsqueda de la barra de navegación inferior (Bottom Navigation Bar). Esto debido a que la búsqueda ya se puede hacer directamente desde la pantalla principal, lo que hace que la aplicación sea más fácil de usar.

La pantalla inicial ahora muestra eventos en tarjetas con un fondo más contrastado, lo que facilita la lectura y la diferenciación de cada elemento.

En esta iteración también se llevo a cabo la opción de buscar de manera correcta, buscar empleando filtros específicos y poder observar los eventos que pertenecen a un concierto, aunque de manera bastante rudimentaria. Ademas se creo la opción de compartir evento y se mejoro los modales en los que se mostraban los eventos en un principio.

En la figura 30 podemos observar la diferencia entre los funcionalidades planificadas para esta iteración y las que se han acabado realizando.

Además de los cambios aquí mostrados se han realizado múltiples cambios en ambas interfaces y en los proyectos.
A continuación se muestra la tabla comparativa entre las tareas planificadas para esta iteración y las que se acabaron realizando.
Ademas de realizar las tareas que en teoría estaban destinadas a esta iteración se acabaron de implementar las tareas de iteraciones anteriores.

vista de eventos de favoritos	Sí
Filtro compuesto	Sí
Compartir en redes	Sí
Enviar notificaciones	Si
Mejora en ambas interfaces	Sí

Cuadro 29: Comparación entre trabajo esperado y realizado para la 3ª Iteración

5.5. Cuarta iteración

En un principio, la cuarta iteración estaba planificada exclusivamente para la ejecución de las pruebas finales de usuario. Sin embargo, durante su desarrollo surgieron ciertos problemas que obligaron a reorientar parte del tiempo y los recursos disponibles. Principalmente, la gestión de los errores detectados en etapas previas no fue tan eficaz como se esperaba, lo que derivó en que tuviéramos que dedicar buena parte de la iteración a resolver incidencias y a realizar ajustes técnicos para garantizar la estabilidad del sistema.

Tras realizar los cambios y ajustes necesarios sobre el proyecto,solucionamos estos contratiempos. Finalmente logramos llevar a cabo las pruebas con usuarios reales en el tiempo restante. Los resultados y el análisis detallado de estas pruebas se presentarán en apartados posteriores, donde se evaluará el alcance de los objetivos propuestos y el grado de cumplimiento de los requisitos funcionales y de usabilidad.

Los cambios principales que ha habido que llevar a cabo en esta iteración han sido solucionar la manera de mostrar los eventos en los modales ya que algunos estaban generando algunos errores a la hora de mostrar datos, mostrando datos como nulos o no especificados lo cual era un error.

Se han mejorado algunos aspectos técnicos como la distribución del código y se han modificado algunas secciones de código para brindar una mejor experiencia de uso al usuario final.

Respecto a las pruebas realizadas sobre nuestra aplicación, debido a que han surgido varios inconvenientes se ha debido optar por acortar el numero de pruebas a realizar, teniendo así que descartar la opción de realizar pruebas unitarias.
En la Sección 6 se aborda este tema con mayor detalle y se presentan las pruebas de usabilidad realizadas a los usuarios.

Pruebas unitarias	No
Pruebas de usabilidad	Sí
Modificaciones de código	Sí
Modificaciones en interfaces	Si
Dockerización	Si

Cuadro 30: Comparación entre trabajo esperado y realizado para la 3ª Iteración

Capítulo 6

6. Pruebas

En este capítulo se aborda la metodología y estrategias para realizar las pruebas de software sobre el código desarrollado en el proyecto . El testing constituye una etapa fundamental del ciclo de desarrollo de software, representando entre el 15 y 25 % del presupuesto de este.

El proceso de testing abarca desde pruebas unitarias automatizadas hasta validaciones con usuarios finales, asegurando que la aplicación cumple tanto con los requisitos funcionales como con las expectativas de usabilidad y rendimiento.

Dada la naturaleza del proyecto y las limitaciones temporales, se ha adoptado por priorizar la validación con usuarios finales por encima de la cobertura qeue brindan las pruebas unitarias . Esta decisión se fundamenta en el principio de que las pruebas más valiosas son aquellas que revelan problemas que los usuarios reales experimentarían.
Para la implementación de pruebas unitarias, el proyecto emplea unittest, la biblioteca estándar de Python para realizar pruebas unitarias . Esta herramienta está basada en JUnit de Java, framework con el cual se han realizado prácticas académicas previas.

Las pruebas unitarias tienen como objetivo fundamental simular diferentes entradas y verificar que las salidas generadas por la aplicación coincidan con los resultados esperados . Este proceso permite validar el comportamiento de funciones individuales de manera aislada, facilitando la detección temprana de errores y la refactorización segura del código.

6.1. Problemas en la implementación de Test Unitarios

Pese a la importancia reconocida de las pruebas unitarias en el desarrollo de software y la preparación inicial para su implementación mediante unittest, durante el desarrollo del proyecto no se pudieron llevar a cabo debido a limitaciones temporales.

Esta situación se debe principalmente a un error en el cálculo inicial del alcance del proyecto, donde se subestimó el tiempo necesario para el desarrollo de las funcionalidades principales de la aplicación. La planificación inicial no contempló adecuadamente la complejidad de ciertas implementaciones, lo que resultó en una redistribución de recursos temporales hacia el desarrollo de características esenciales del sistema.

En consecuencia, se priorizó la finalización de las funcionalidades core de la aplicación y la realización de pruebas con usuarios finales, considerando que estas últimas proporcionarían feedback más directo sobre la usabilidad y efectividad de la solución desarrollada en el contexto real de uso.

Esta decisión, aunque no ideal desde una perspectiva de ingeniería de software, refleja las limitaciones prácticas que pueden surgir en proyectos con restricciones temporales definidas, donde es necesario priorizar entre diferentes aspectos del desarrollo para asegurar la entrega de un producto funcional.
Esta cuestión se verá reflejada en la sección **??**

6.2. Test de usabilidad

6.2.1. Importancia de los tests de usabilidad

Como se expone en [21], los tests de usabilidad permiten identificar de forma objetiva posibles fricciones que experimenta el usuario al interactuar con un sistema. Gracias a estos test, se pueden optimizar aspectos como la navegación, la disposición de los elementos o la comprensión de las funcionalidades, con el objetivo de mejorar la experiencia global del usuario. Realizar un test de usabilidad no solo contribuye a detectar errores o puntos débiles, sino que también aporta

información valiosa para tomar decisiones centradas en el usuario, maximizando así la eficacia del producto final.

6.3. Desarrollo de los Tests de usablilidad

Con el objetivo de verificar que ambas aplicaciones cumplen con los requisitos establecidos y que su uso resulta sencillo e intuitivo para personas sin experiencia previa, se ha diseñado una serie de pruebas de usuario. Para ello, se solicitará a los participantes que realicen diversas tareas en ambas plataformas, evaluando su grado de satisfacción y comodidad mediante métricas objetivas y subjetivas.

Se define brevemente el perfil de usuario participante (user persona): usuarios con conocimientos básicos de informática, sin experiencia previa en las aplicaciones, pero familiarizados con el uso habitual de dispositivos móviles y navegadores web.

Las métricas principales que se emplearán para valorar la experiencia son la eficacia, la eficiencia y la satisfacción. Para cuantificar estos aspectos, se registrarán las siguientes variables durante la realización de cada tarea:

- Tiempo empleado en completar la tarea.
- Número de clics o acciones necesarias.
- Errores cometidos o dificultades encontradas.
- Si la tarea se completó correctamente o no.

A continuación, se detallan las tareas que los usuarios deberán realizar en cada plataforma:

Desde la aplicación móvil:

- Añadir a favoritos un concierto específico llamado Celta Medieval: Se le proporcionará al usuario el nombre de un concierto concreto y deberá localizarlo en la aplicación. Una vez encontrado, tendrá que utilizar la funcionalidad de "añadir a favoritos" para guardarlo en su lista personal de eventos preferidos.
- Compartir por mensaje una gira que incluya un concierto el día 26-09-2025: El usuario deberá buscar una gira que contenga al menos un concierto programado para el 25-09-2025. Tras identificarla, tendrá que compartir la información de la gira usando alguna de las opciones de mensajería o redes sociales integradas en la app.
- Buscar todos los conciertos de Medina de Rioseco cuyo precio sea inferior a 50 euros: El usuario deberá emplear las herramientas de búsqueda y filtrado de la aplicación para localizar todos los conciertos que se celebren en Medina de Rioseco y cuyo precio de entrada sea menor a 50 euros. Se evaluará su capacidad para aplicar correctamente los filtros y obtener la lista solicitada.
- Eliminar concierto de favoritos: el usuario debe eliminar el concierto con nombre Celta Medieval de la lista de favoritos.

Desde la aplicación web:

- Loguearse con las credenciales de usuario administrador en la plataforma.
- Crear un usuario con rol de administrador: El usuario deberá acceder al apartado de gestión de usuarios y completar el proceso de creación de una nueva cuenta, asignándole el rol de administrador.
- Crear un nuevo concierto desde el formulario de creación de conciertos.

- Subir un fichero previamente creado con varios conciertos: Se entregará al usuario un archivo en formato .csv con varios conciertos ya preparados. El usuario deberá utilizar la funcionalidad de importación de la plataforma web para cargar estos conciertos en el sistema.
- Crear una gira que incluya tres conciertos en fechas distintas: El usuario deberá crear una nueva gira, introduciendo los datos solicitados y asociando a la gira tres conciertos diferentes, cada uno con una fecha distinta.
- Crear un festival que contenga dos conciertos específicos en la misma fecha: Se ha solicitado al usuario que cree un festival que contenga dos conciertos que figuraban en el csv introducido anteriormente
- Editar la imagen de un concierto existente: Se solicitará al usuario que localice un concierto ya creado y utilice la opción de edición para cambiar la imagen asociada al evento, comprobando que la nueva imagen se guarda y muestra correctamente.
- Aceptar un concierto en estado pendiente: El usuario tendrá que acceder a la sección de conciertos pendientes, seleccionar uno para marcarlo como cancelado.
- Cancelar evento disponible: el usuario deberá eliminar un concierto especifico de la sección de disponibles.
- Cerrar sesión: Para finalizar el usuario debera cerrar la sesión actual, dando así por concluidas las pruebas.

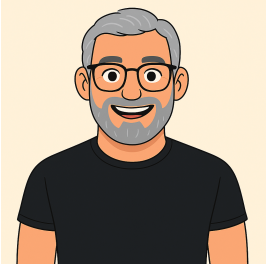
La recopilación y el análisis de los datos obtenidos permitirán valorar de manera objetiva el grado de usabilidad de ambas aplicaciones, así como identificar posibles áreas de mejora en la experiencia de usuario. Se podrían realizar tantas pruebas como requisitos funcionales tiene la aplicación, ahora bien, consideramos que con estas tares se prueba la usabilidad de ambas aplicaciones.

6.4. Descripción de usuarios

A continuación se mostrará una pequeña descripción de los usuarios diferenciando estos entre usuarios los cuales han ejecutado pruebas sobre la aplicación web y usuarios que han realizado pruebas sobre la aplicación móvil

6.5. Usuarios de la prueba en aplicación web

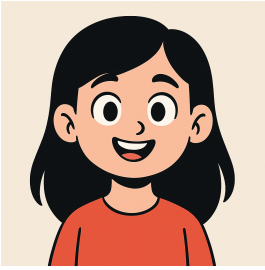
Ficha del Usuario administrador



Nombre completo:Alberto
Edad:52
Genero:Masculino
Frecuencia de asistencia a eventos musicales:1 vez al mes
Uso habitual de aplicaciones musicales: Sí
Uso habitual de redes sociales: Sí
Forma de descubrir nuevos eventos musicales: A través de las redes sociales

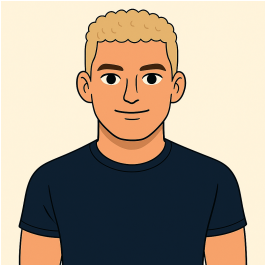
6.6. Usuarios de la prueba en aplicación móvil

Ficha del Usuario



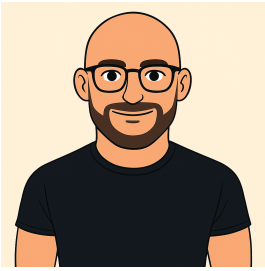
Nombre completo: Lucia
Edad: 21 Años
Genero: Femenino
Frecuencia de asistencia a eventos musicales: 1 vez al mes
Uso habitual de aplicaciones musicales: Si
Uso habitual de redes sociales: Si
Forma de descubrir nuevos eventos musicales: A través de las redes sociales

Ficha del Usuario



Nombre completo: Rodrigo
Edad: 23 Años
Genero: Masculino
Frecuencia de asistencia a eventos musicales:1 vez al mes
Uso habitual de aplicaciones musicales:Si
Uso habitual de redes sociales:Si
Forma de descubrir nuevos eventos musicales: A través del boca a boca

Ficha del Usuario



Nombre completo: Javier
Edad: 58 Años
Genero: Masculino
Frecuencia de asistencia a eventos musicales: 1 vez al mes
Uso habitual de aplicaciones musicales: No
Uso habitual de redes sociales: No
Forma de descubrir nuevos eventos musicales: A través de redes sociales

6.7. Resultados del test de usabilidad

En la siguiente sección se mostrarán los resultados obtenidos al solicitar a los nuevos usuarios que realicen las tarea planteadas anteriormente.

Hay que tener en cuenta que los usuarios no han tenido contacto previo con la app por lo que puede que necesiten mas pulsaciones de las que tomarían las acciones normalmente.

Además, el estudiante ha realizado estas mismas pruebas para tener un barómetro con el que medir los tiempos de los usuarios.

6.7.1. Resultados del test de usabilidad en aplicación móvil

En este caso el único sujeto de pruebas ha sido el propietario de la aplicación web inicial WatiosYDecibelios la cual comentamos en un principio y podemos encontrar en el siguiente link: [50]

Respecto al primer supuesto el usuario no ha tenido ningún problema para loguearse en la aplicación web.

Login de administrador	Usuario Administrador	Estudiante
Tarea completada	Si	Si
Tiempo empleado	15 secs	6 secs
Errores o sugerencias	No	No
Número de clics	3	3

Cuadro 31: Métricas de test de usabilidad para login

Realizando el caso de uso de crear un nuevo usuario en principio el usuario ha navegado un poco por la interfaz de la aplicación pero rápidamente ha accedido a la sección indicada.

Registro de nuevo administrador	Usuario Administrador	Estudiante
Tarea completada	Si	Si
Tiempo empleado	40 secs	20 secs
Errores o sugerencias	No	No
Número de clics	8	6

Cuadro 32: Métricas para crear cuenta con rol administrador

Respecto a la creación de usuario no ha habido mayor contratiempo, a excepción de que se solicitaba una entrada especifica y el usuario ha tenido que preguntar los datos para introducirlos correctamente. De nuevo el usuario ha navegado brevemente antes de realizar la prueba viendo las funcionalidades implementadas.

Crear un concierto nuevo	Usuario Administrador	Estudiante
Tarea completada	Si	Si
Tiempo empleado	1:15 min	35 secs
Errores o sugerencias	No	No
Número de clics	13	12

Cuadro 33: Métricas para crear un concierto individual

En este caso el usuario intuitivamente ha entendido que la opción para la subida masiva de conciertos se trataba de la opción de subir un csv. Respecto a este no ha habido ningún inconveniente y se ha realizado correctamente.

Subir varios conciertos desde fichero	Usuario Administrador	Estudiante
Tarea completada	Si	Si
Tiempo empleado	30 secs	15 secs
Errores o sugerencias	No	No
Número de clics	5	4

Cuadro 34: Métricas de test de usabilidad para importar eventos desde archivo

El fichero subido en el caso anterior contenía varios conciertos que se utilizarán a continuación. Al agregar una gira que incluía múltiples conciertos, surgieron dudas sobre cómo incorporar correctamente los conciertos contenidos en ella. En lugar de utilizar la opción de indexar por nombre

que ofrece la lista, el usuario intentó localizarlos manualmente. Tras un periodo de búsqueda, finalmente empleó la función de indexación por nombre, y el resto del proceso transcurrió sin mayores inconvenientes.

Crear una gira musical	Usuario Administrador	Estudiante
Tarea completada	Si	Si
Tiempo empleado	1:15 min	26 secs
Errores o sugerencias	No	No
Número de clics	7	8

Cuadro 35: Métricas de usabilidad para la creación de giras

En este caso el usuario de nuevo ha vuelto a buscar los conciertos manualmente desde la tabla en la que se visualizan los eventos.

Crear un festival con varios artistas	Usuario Administrador	Estudiante
Tarea completada	Si	Si
Tiempo empleado	1 min	20 secs
Errores o sugerencias	No	No
Número de clics	9	7

Cuadro 36: Métricas para creación de festivales

En el caso de la modificación de un concierto ha surgido un pequeño inconveniente. Debido a la paginación el usuario no encontraba el concierto, ya que este estaba en la segunda pagina, lo cual se ha traducido en algo más de tiempo de ejecución.

Respecto a esto el usuario ha sugerido que el botón de compaginación no este tan alejado ya que «El peso visual de la app cae sobre el lado derecho y es difícil percatarse de la opción de Siguiente en la paginación»

Modificar imagen de un evento	Usuario Administrador	Estudiante
Tarea completada	Si	Si
Tiempo empleado	1:50 min	17 secs
Errores o sugerencias	Sugerencia	No
Número de clics	7	6

Cuadro 37: Métricas de edición de eventos existentes

A la hora de aceptar un evento no ha habido ningún problema y el caso se ha desarrollado correctamente. Intuitivamente ha clicado en los conciertos desplegando así el panel de gestión correctamente.

Aceptar evento pendiente	Usuario Administrador	Estudiante
Tarea completada	Si	Si
Tiempo empleado	33 secs	15 secs
Errores o sugerencias	No	No
Número de clics	3	4

Cuadro 38: Métricas de gestión de eventos pendientes

Respecto a la hora de cancelar un evento disponible, de nuevo, el usuario no ha tenido ningún problema

Cancelar evento disponible	Usuario Administrador	Estudiante
Tarea completada	Si	Si
Tiempo empleado	26 secs	14 secs
Errores o sugerencias	No	No
Número de clicks	4	4

Cuadro 39: Métricas de cancelación de eventos

Para finalizar con las pruebas de usabilidad de la sección web el usuario ha realizado la tarea de cerrar sesión perfectamente.

Cerrar sesión del sistema	Usuario Administrador	Estudiante
Tarea completada	Si	Si
Tiempo empleado	2 secs	3 secs
Errores o sugerencias	No	No
Número de clicks	1	1

Cuadro 40: Métricas para el cierre de sesión del sistema

Tras realizar estas pruebas sobre la interfaz de usuario podemos entender que su aprendizaje resulta simple y que para usuarios nuevos resulta intuitiva.

6.7.2. Resultados del test de usabilidad en aplicación móvil

Respecto a las pruebas realizadas sobre dispositivos móviles las pruebas de estas han sido realizadas por tres usuarios, de los cuales hemos podido obtener los siguientes resultados:

La **primera prueba** realizada consistía en agregar a favoritos un evento con un nombre determinado. En este caso los resultados han sido variados.

En uno de los casos el usuario no se dio cuenta de la existencia de un filtro hasta después de la prueba. En otro caso, uno de los usuarios no encontraba la opción de agregar a favoritos, y ha comenzado a navegar a través de las diferentes opciones de la aplicación: Tipos de eventos, vista inicial, vista de favoritos ... Posteriormente se dio cuenta de la barra y realizó correctamente la prueba.

En este caso, algo algo que resulta curioso es que el usuario accedió antes al filtro de búsqueda compuesta que al de búsqueda simple

Añadir a favoritos un concierto específico	Usuario 1	Usuario 2	Usuario 3	Estudiante
Tarea completada	Si	Si	Si	Si
Tiempo empleado	13 secs	5 secs	1:08 min	6 secs
Errores o sugerencias	No	No	No	No
Número de clicks	9	3	13	2

Cuadro 41: Métricas de test de usabilidad de la tarea de añadir evento a favoritos

Respecto a la **segunda prueba**, esta trataba sobre compartir por mensajes una gira con unas condiciones especificas (<50€ y en la población de Medina de Rioseco).

En este caso algunos de los usuarios no se dieron cuenta de la existencia del filtro compuesto hasta algo entrada la prueba.

Por otro lado, otro de los usuarios no recayó en que se encontraba dentro de una sección de eventos diferente a la solicitada.

Respecto a la opción de enviar mensaje todos los usuarios realizaron el envió intuitivamente sin complicaciones.

Compartir por mensaje una gira específica	Usuario 1	Usuario 2	Usuario 3	Estudiante
Tarea completada	Si	Si	Si	Si
Tiempo empleado	20 secs	45 secs	40 secs	22 secs
Errores o sugerencias	No	No	No	No
Número de clics	6	16	10	4

Cuadro 42: Métricas de test de usabilidad para la tarea de compartir un evento especifico

Respecto a la realización de las pruebas del tercer test estas han transcurrido sin mayor error.

A la hora de realizar estas, los usuarios empleaban la búsqueda compuesta intuitivamente.

Buscar conciertos con filtro	Usuario 1	Usuario 2	Usuario 3	Estudiante
Tarea completada	Si	Si	Si	Si
Tiempo empleado	20 secs	7 secs	25 secs	15 secs
Errores o sugerencias	No	No	No	No
Número de clics	6	4	7	4

Cuadro 43: Métricas de test de usabilidad para la tarea de buscar un concierto con filtro de búsqueda compuesta

Por ultimo, respecto a la cuarta y ultima prueba ejecutada sobre dispositivos móviles, eliminar el concierto agregado anteriormente a la lista de favoritos.

En este caso los usuarios no han tenido ningún problema a la hora de realizar la tarea. Con esto se puede apreciar que la curva de aprendizaje es positiva o progresiva, donde al principio hubo errores menores, pero al realizar unas pequeñas pruebas, se alcanzó un desempeño correcto.

Eliminar concierto de favoritos	Usuario 1	Usuario 2	Usuario 3	Estudiante
Tarea completada	Si	Si	Si	Si
Tiempo empleado	5 secs	4 secs	13 secs	5 secs
Errores o sugerencias	No	No	No	No
Número de clics	3	2	7	3

Cuadro 44: Métricas de test de usabilidad de eliminar concierto de favoritos

6.8. Test de usabilidad

Para evaluar la usabilidad de la aplicación, se diseñó un cuestionario basado en el estándar System Usability Scale (SUS). Este cuestionario fue administrado a los usuarios tras la realización de las pruebas. Aunque se adaptaron algunas preguntas, el formato general siguió las recomendaciones establecidas para obtener métricas fiables y estandarizadas.

En la figura B se muestra el cuestionario aplicado, compuesto por 12 ítems. Algunas preguntas fueron formuladas con fines informativos —por ejemplo, ”La app tiene un diseño visual atractivo.º ”No he tenido problemas técnicos al usar la app”—, mientras que las restantes se utilizaron como base para el cálculo del índice SUS.

Para permitir su análisis cuantitativo, las respuestas cualitativas se convirtieron a una escala de Likert de 1 a 5, como se muestra en la tabla 45. Además, como requiere la metodología SUS, se ajustaron las puntuaciones de las preguntas con redacción negativa (pares) invirtiendo su valor antes del cómputo final. En la tabla 46 se ejemplifica este proceso de transformación de los datos.

Respuesta	Valor
Totalmente en desacuerdo	1
En desacuerdo	2
Neutral	3
De acuerdo	4
Totalmente de acuerdo	5

Cuadro 45: Escala de valores utilizada en el cuestionario

Pregunta original	Valor	Pregunta ajustada (SUS)	Valor parseado
La app es fácil de aprender a usar.	5	La app es difícil de aprender a usar.	1
Me siento cómodo/a utilizando la app sin ayuda externa.	4	Necesito ayuda externa para usar la app.	2
Puedo completar mis tareas rápidamente usando la app.	5	Tardo demasiado en completar mis tareas con la app.	1
Es fácil encontrar la información que necesito.	4	Es difícil encontrar la información que necesito.	2
Las funciones de la app están bien organizadas.	5	Las funciones de la app están mal organizadas.	1
La app responde de forma rápida a mis acciones.	3	La app responde lentamente a mis acciones.	3
El lenguaje utilizado en la app es claro y comprensible.	2	El lenguaje utilizado en la app es confuso.	4
Me siento satisfecho/a con la experiencia general de uso.	5	Estoy insatisfecho con la experiencia de uso.	1
Las opciones y menús están ubicados de forma lógica.	4	Las opciones están mal ubicadas.	1
Confío en que la app funcionará como espero.	5	No confío en que la app funcionará correctamente.	1

Cuadro 46: Ejemplo de adaptación y parseo de preguntas SUS

Con base en lo anterior, es posible plantear las conclusiones derivadas de la ejecución de estas pruebas.

6.8.1. Conclusión de Test de usabilidad de aplicación web

A lo largo de las pruebas realizadas en dispositivos móviles, se observó una curva de aprendizaje progresiva.

Aunque en las primeras tareas algunos usuarios presentaron dificultades puntuales, como no identificar ciertos filtros o no localizar de inmediato determinadas funciones, estos problemas fueron superados en las siguientes pruebas. En general, todas las funcionalidades fueron completadas correctamente desde el inicio, aunque en algunos casos determinadas algunas acciones tomaron algo más de tiempo del esperado.

En las pruebas posteriores, las tareas se realizaron de forma más fluida e intuitiva, lo que evidencia una mejora en la comprensión y uso de la interfaz con un mínimo de experiencia previa.

Cabe destacar que se detectó una diferencia significativa en el comportamiento entre usuarios habituales de redes sociales y aquellos que no lo son. Los primeros realizaron un mayor número de interacciones en menos tiempo, mientras que el usuario sin experiencia previa en redes sociales tardó más tiempo en completar las tareas y le tomaba más tiempo realizar cada click, lo que sugiere

una influencia directa del hábito digital en la eficiencia de uso de la aplicación.

.

A continuación se mostraran los resultados recogidos y parseados tras completar le cuestionario de usuario

Respuestas a formulario de usuario de aplicación web

A continuación, se presentan los resultados obtenidos tras la cumplimentación del cuestionario por parte de los usuarios. Estos ya están parseados siguiendo el formato mostrado en las tablas 45 y 46.

Pregunta Ajustada	**Usuario 1**	**Usuario 2**	**Usuario 3**
1. La app es fácil de aprender a usar.	5	5	5
2. Me siento **incómodo/a** utilizando la app sin ayuda externa.	2	1	1
3. Puedo completar mis tareas rápidamente usando la app.	5	5	5
4. Es **difícil** encontrar la información que necesito.	1	1	2
5. Las funciones de la app están bien organizadas.	5	5	4
6. La app responde de forma **lenta** a mis acciones.	1	2	1
7. El lenguaje utilizado en la app es claro y comprensible.	5	5	5
8. Me siento **insatisfecho/a** con la experiencia general de uso.	1	1	1
9. Las opciones y menús están ubicados de forma lógica.	4	5	4
10. **Desconfío** en que la app funcionará como espero.	1	2	1

Cuadro 47: Resultados parseados para cálculo SUS (posiciones pares invertidas). Valores: 1 a 5.

Puntaje SUS

Calculamos la media de cada pregunta (posiciones pares invertidas) para los 3 usuarios:

Pregunta	**Media (X)**	**Ajuste SUS**
1	$\frac{5+5+5}{3} = 5{,}00$	5.00
2	$\frac{2+1+1}{3} = 1{,}33$	1.33
3	$\frac{5+5+5}{3} = 5{,}00$	5.00
4	$\frac{1+1+2}{3} = 1{,}33$	1.33
5	$\frac{5+5+4}{3} = 4{,}67$	4.67
6	$\frac{1+2+1}{3} = 1{,}33$	1.33
7	$\frac{5+5+5}{3} = 5{,}00$	5.00
8	$\frac{1+1+1}{3} = 1{,}00$	1.00
9	$\frac{4+5+4}{3} = 4{,}33$	4.33
10	$\frac{1+2+1}{3} = 1{,}33$	1.33

Cuadro 48: Medias por pregunta (escala 1–5).

Sumamos todas las medias de las columnas ajustadas siguientdo la guia siguiente [47]:

$$\text{Suma positivos} = 5{,}00 + 5{,}00 + 4{,}67 + 5{,}00 + 4{,}33 = \mathbf{24} -> 24 - 5 = \mathbf{19}$$

$$\text{Suma negativos} = 1{,}33 + 1{,}33 + 1{,}33 + 1{,}00 + 1{,}33 = \mathbf{6{,}33} -> 25 - 6{,}33 = \mathbf{18{,}67}$$

$$\text{Puntaje SUS} = (19 + 18{,}67) \times 2{,}5 = \mathbf{94{,}175}$$

Según diversos estudios, una puntuación SUS entre 70 y 100 se considera indicativa de una buena usabilidad, lo que sugiere que la aplicación es válida y bien aceptada por los usuarios .

Respuestas obtenidas de usuario de aplicación web

- **¿Consideras la aplicación útil?**
  - Usuario 1: Sí.
  - Usuario 2: Sí.
  - Usuario 3: Sí.
- **¿Cuál es la mayor ventaja de la aplicación?**
  - Usuario 1: Puedes tener toda la información junta en la misma aplicación.
  - Usuario 2: El filtro.
  - Usuario 3: Informacion sobre los conciertos.
- **¿Y el mayor inconveniente?**
  - Usuario 1: Su aspecto.
  - Usuario 2: Me costó ver la opción de conciertos, giras y festivales.
  - Usuario 3: Falta información adicional de los festivales.
- **¿Agregarías alguna funcionalidad a la aplicación? En caso afirmativo, ¿cuál sería?**
  - Usuario 1: la opción de poder adquirir la entrada online a través de la aplicación.
  - Usuario 2: Un mapa donde aparezcan gráficamente.
  - Usuario 3: Si, informacion de: accesos a los conciertos, parking cercanos, transporte público, barras, forma de pago, duracion de concierto estimada, presencia de baños, si se puede salir y volver a entrar. Además el filtro podría estar escrito en letras y no con un icono. Por último que se de información sobre los campings para los festivales

6.8.2. Conclusión de Test de usabilidad de aplicación web

Respecto a los datos obtenidos en el cuestionario realizado, estos han sido los mnostrados en la tabla 49.

De nuevo y al igual que con las pruebas realizadas sobre usuarios de dispositivos movilesllos datos obtenidos ya están parseados siguiendo el formato mostrado en las tablas 45 y 46.

Pregunta Ajustada	**Usuario Admin**
1. La app es fácil de aprender a usar.	5
2. Me siento **incómodo/a** utilizando la app sin ayuda externa.	1
3. Puedo completar mis tareas rápidamente usando la app.	5
4. Es **difícil** encontrar la información que necesito.	1
5. Las funciones de la app están bien organizadas.	5
6. La app responde de forma **lenta** a mis acciones.	1
7. El lenguaje utilizado en la app es claro y comprensible.	5
8. Me siento **insatisfecho/a** con la experiencia general de uso.	1
9. Las opciones y menús están ubicados de forma lógica.	5
10. **Desconfío** en que la app funcionará como espero.	1

Cuadro 49: Resultados parseados para cálculo SUS (posiciones pares invertidas). Valores: 1 a 5.

Sumamos los valores de la columna "Valor ajustado":

$$5+5+5+5+5 = \mathbf{25} -> 25-5 = \mathbf{20}$$

$$1+1+1+1+1 = \mathbf{5} -> 25-5 = \mathbf{20}$$

$$\text{Puntaje SUS} = ((20+20) \times 2{,}5 = \mathbf{100}$$

Según la escala SUS estándar la puntuación ha sido perfecta por lo que la aplicación funciona conforme a lo estipulado.

Respuestas obtenidas de usuario de aplicación web

- **¿Cuánto tiempo dedicabas semanalmente a la gestión de Watios y Decibelios**: En subirlos 30 minutos al día, tardo más en recopilar información.
- **¿Cuánto tiempo te llevaba subir un evento a la web?**: 5 minutos
- **Al subir eventos, ¿sueles publicar uno solo o varios a la vez? Cuáles eran las tareas más repetitivas o tediosas en el proceso de subir eventos?**: Suelo publicarlos cuando tengo varios, pero no siempre.
- **¿Había algún tipo de plantilla o automatización que utilizabas para agilizar el proceso?**: Utilizo una plantilla de word
- **¿Con qué frecuencia actualizabas la información de los eventos ya publicados?**: Se actualiza cada día, incluso varias veces al día.
- **¿Tuviste alguna vez problemas con errores o inconsistencias al publicar? ¿Cómo los resolvías?**: Alguna que otra vez
- **La gestión de la plataforma la llevabas tú solo o colaboraba más gente?**: la llevo yo solo

- **¿Cuál es la mayor ventaja de la aplicación?**: información completa ordenada y gratuita.
- **¿Y el mayor inconveniente?**: Que es muy artesanal, tiene muchas cosas a mejorar.
- **¿Consideras la aplicación útil?**: filtros por estilos

Tras el cuestionario, podemos observar que con la práctica se cubren algunas de las necesidades que el usuario planteaba como mejoras para su aplicación. Además, al emplear funcionalidades como el uso de plantillas, puede aprender fácilmente a introducir conciertos a través de ficheros externos en formato CSV.

Capítulo 7

7. Conclusiones

En esta sección se comentarán las principales conclusiones y aprendizajes obtenidos después del desarrollo de la aplicación, así como las funcionalidades que se plantean como trabajo futuro para este tipo de aplicaciones.

El desarrollo de este trabajo se realizó tomando como referencia la aplicación Watios y Decibelios, mostrada en diversas ocasiones a lo largo de este proyecto. En todo momento se ha intentado seguir como ejemplo su estructuración respecto a los datos de conciertos, giras y festivales, lo que ha proporcionado una base sólida para el diseño de nuestra propuesta. Al estar programada en WordPress, la aplicación original presenta una gran libertad a la hora de agregar eventos, lo que considero una excelente forma de crear una página web dinámica y flexible. Esta característica representa el punto más positivo de la aplicación web previamente existente.
Sin embargo, esta misma flexibilidad conlleva cierta complejidad en la estructuración de los datos. Al no seguir modelos ni estructuras completamente definidas, resulta complicado estandarizar la información, lo que puede constituir la principal desventaja de nuestra aplicación en comparación con la original.

La experiencia ha resultado sumamente enriquecedora e interesante. Generar un producto real y plantear su posible funcionalidad me ha permitido no solo trabajar con muchos de los ámbitos estudiados durante la carrera, sino también aprender nuevas tecnologías y continuar profundizando en aquellas que ya conocía. He podido presenciar de primera mano cómo es desarrollar una aplicación prácticamente desde cero, partiendo de una idea inicial y llevándola a cabo paso a paso. Esta experiencia, además de interesante, me ha mostrado que pueden surgir diversos problemas durante el desarrollo de aplicaciones, no solo a nivel funcional sino también a nivel de diseño.

El desarrollo de ambas plataformas ha sido tan diferente entre sí que podría catalogarse como dos proyectos completamente separados. Estos han tenido estructuras completamente diferentes, desde el backend hasta el frontend, a excepción de la capa de persistencia. Esta situación, si se hubiese abordado mejor desde el principio, podría haber tenido una solución más eficiente desarrollando la aplicación íntegramente en React-Native, que ya es multiplataforma por naturaleza.

Otro de los errores significativos fue la gestión del tiempo y el manejo de los fallos. La sensación que tengo tras realizar la práctica es que perdí tiempo debido a planteamientos poco eficientes. En parte del proyecto me basé únicamente en la funcionalidad, dejando de lado la estructuración del código, lo que posteriormente originó la necesidad de una reestructuración completa. Esto, sumado a la implementación de tecnologías desconocidas, resultó en desviaciones considerables del tiempo planificado originalmente.

Todos los retrasos y problemas generados a raíz de malas prácticas han desembocado en que no se puedan realizar pruebas unitarias sobre ambos proyectos. Debido a la mala estimación de tiempo he tenido que ir posponiendo esta tarea hasta finalmente, y como se comenta en la sección 6.1 se ha omitido la sección de pruebas unitarias, siendo este un error a tener en cuenta en próximos proyectos.
Me gustaría destacar que, si tuviese la oportunidad de realizar el proyecto nuevamente, haría muchas cosas de manera diferente. Sin embargo, considero que de los errores se aprende, y esta práctica me ha proporcionado aprendizajes muy valiosos tanto a nivel técnico como metodológico. Esta experiencia ha reforzado mi comprensión sobre la importancia de una planificación detallada, la correcta selección de tecnologías, y la necesidad de mantener una estructura de código sólida desde el inicio del desarrollo. Estos aprendizajes serán fundamentales para futuros proyectos profesionales.

7.1. Trabajos futuros

En la etapa inicial de planificación se definieron una serie de requisitos que la aplicación debía cumplir. Estos fueron clasificados por niveles de prioridad: alta, media y baja. El desarrollo comenzó abordando los requisitos de prioridad alta, seguido por los de prioridad media y, finalmente, los de menor prioridad.

Durante el transcurso del desarrollo, surgieron nuevas ideas relacionadas con funcionalidades adicionales y diferentes enfoques de implementación. Las principales funcionalidades que se plantearon como trabajos futuros para seguir ampliando y mejorando el sistema. Entre ellas destacan las siguientes:

- **Sección de gestión de los usuarios**: Ahora mismo tenemos una sección de creación de los usuarios, pero no tenemos opciones de eliminarlos o de editarlos, por ejemplo brindando opciones para deshabilitarlos temporalmente o simplemente para realizar un cambio de permisos, de administrador a externo o viceversa.
- **Sección de visualización de conciertos mejorada**: Uno de los planteamientos principales y mas vistosos dentro del diseño principal de los Mock-Ups era la opción de poder observar los conciertos en una vista similar a un calendario. En el desarrollo de la aplicación se opto por priorizar otras funcionalidades antes que esta por lo que la implementación de esta finalmente no se pudo llevar a cabo, pero seria uno de los objetivos si se retomase el desarrollo de la aplicación.
- **Pruebas**: Me gustaría poder realizar más pruebas para poder probar mas a fondo la funcionalidad completa de la aplicación.
- **Pruebas en dispositivos IOS**: Debido a que no cuento con ningún dispositivo IOS no se han podido realizar pruebas sobre este SO, pero en un futuro seria bastante conveniente poder realizar pruebas ya que cubren una gran parte del mercado mundial
- **Adición de nuevos atributos**: Una funcionalidad que podría resultar útil en un futuro podría ser la introducción de nuevos atributos como por ejemplo los géneros musicales.

  Me gustaría aprovechar esta sección para agradecer al propietario de la aplicación Watios y Decibelios por permitirme desarrollar su idea y por involucrarme en un proyecto tan interesante.

  También me gustaría agradecer a los profesores que han tutelado este trabajo ya que han estado involucrados en todo momento y me han facilitado en gran manera el desarrollo del TFG.

8. Bibliografía

Referencias

[1] Proyectos Ágiles. *Desarrollo iterativo e incremental*. 2025. URL: https://proyectosagiles.org/desarrollo-iterativo-incremental/.

[2] Andro4all. *Google despide a un equipo entero de ingenieros: los sustituirá por otros más baratos*. 2024. URL: https://www.lavanguardia.com/andro4all/google/google-despide-a-un-equipo-entero-de-ingenieros-los-sustituira-por-otros-mas-baratos.

[3] Borja Orbegozo Arana. *Curso práctico avanzado de PostgreSQL*. Autoeditado, 2020.

[4] Jim Arlow e Ila Neustadt. *UML 2 y el proceso unificado*. Anaya Multimedia, 2006.

[5] Atlassian. *Gantt Charts Explained [+ How to Create One]*. 2025. URL: https://www.atlassian.com/es/agile/project-management/gantt-chart.

[6] bycrypt. *bcrypt: Hashing Passwords in Python with BCrypt*. 2025. URL: https://pypi.org/project/bcrypt/.

[7] James Cadle y Donald Yeates. *Project Management for Information Systems*. 5.ª ed. Pearson Education, 2008.

[8] Change Vision, Inc. *Astah Professional Official Website*. 2024. URL: https://astah.net/products/astah-professional/.

[9] Tom Collings. *Controller-Service-Repository*. 2021. URL: https://tom-collings.medium.com/controller-service-repository-16e29a4684e5.

[10] Tom Collings. *Controller-Service-Repository*. 2021. URL: https://tom-collings.medium.com/controller-service-repository-16e29a4684e5.

[11] DBeaver Community. *DBeaver Community Edition (CE)*. 2025. URL: https://dbeaver.io/.

[12] Rock Content. *API REST: qué es, funcionamiento y ejemplos*. 2025. URL: https://rockcontent.com/es/blog/api-rest/.

[13] Wikipedia contributors. *Desarrollo iterativo y creciente*. 2025. URL: https://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente.

[14] Microsoft Corporation. *Microsoft Excel*. 2025. URL: https://www.microsoft.com/es-es/microsoft-365/excel.

[15] Docker Inc. *Docker Documentation*. 2024. URL: https://docs.docker.com/.

[16] Business Insider España. *Google recorta equipos de desarrolladores de código abierto*. 2024. URL: https://www.businessinsider.es/google-recorta-equipos-desarrolladores-codigo-abierto-1383134.

[17] Figma. *Figma Community*. 2025. URL: https://www.figma.com/community.

[18] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003. ISBN: 978-0321127426.

[19] GitHub, Inc. and OpenAI. *GitHub Copilot: Your AI pair programmer*. 2025. URL: https://github.com/features/copilot.

[20] Elaine M. Hall. *Managing Risk*. Addison-Wesley, 1998.

[21] Hiberus. *¿Qué es un test de usabilidad y por qué deberías hacerlo?* 2022. URL: https://www.hiberus.com/crecemos-contigo/que-es-un-test-de-usabilidad-y-por-que-deberia-hacerlo/.

[22] Pablo Huet. *Arquitectura de software: Qué es y qué tipos existen*. 2022. URL: https://openwebinars.net/blog/arquitectura-de-software-que-es-y-que-tipos-existen/.

[23] INCIBE - Instituto Nacional de Ciberseguridad. *Análisis de riesgos*. Último acceso: 21 de enero de 2025. 2025. URL: https://www.incibe.es/empresas/blog/analisis-riesgos-pasos-sencillo.

[24] Harold Kerzner. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. 12.ª ed. Wiley, 2017.

[25] Google LLC. *Android Studio User Guide*. 2025. URL: https://developer.android.com/studio.

[26] Alan López. *React Native vs Flutter: Una comparativa detallada para 2023*. Último acceso: 21 de enero de 2025. 2023. URL: %5Curl%7Bhttps://www.linkedin.com/pulse/react-native-vs-flutter-una-comparativa-detallada-para-alan-l%C3%B3pez-xlmvf/%7D.

[27] Meta Platforms, Inc. *React Native Documentation*. 2024. URL: https://reactnative.dev/docs/getting-started.

[28] Microsoft. *Microsoft Teams - Plataforma de comunicación y colaboración*. 2025. URL: https://www.microsoft.com/es-es/microsoft-teams/.

[29] Microsoft. *Visual Studio Code - Editor de código fuente*. 2025. URL: https://code.visualstudio.com/.

[30] Microsoft Corporation. *Microsoft Forms: Create Surveys, Quizzes, and Polls*. 2025. URL: https://www.microsoft.com/forms.

[31] Miro. *¿Qué es un diagrama de despliegue UML?* 2025. URL: https://miro.com/es/diagrama/que-es-diagrama-despliegue-uml/.

[32] OpenAI. *ChatGPT: AI Language Model by OpenAI*. 2025. URL: https://chat.openai.com.

[33] Oracle. *SQL - Lenguaje de consulta estructurado*. 2025. URL: https://www.w3schools.com/sql/.

[34] Overleaf. *Overleaf - Plataforma de edición en LaTeX*. 2025. URL: https://www.overleaf.com/.

[35] PostgreSQL Global Development Group. *PostgreSQL Documentation*. 2024. URL: https://www.postgresql.org/docs/.

[36] Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*. Newtown Square, PA: Project Management Institute, 2004.

[37] Python Software Foundation. *Python Official Website*. 2025. URL: https://www.python.org/.

[38] Raghu Ramakrishnan y Johannes Gehrke. *Sistemas de gestión de bases de datos*. 3.ª ed. McGraw-Hill, 2007.

[39] Equipo de Reactive Programming. *Representational State Transfer (REST)*. 2025. URL: https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/rest.

[40] Christopher Saez. *Flutter Layoff: A Good News for React Native in 2024?* 2024. URL: https://medium.com/@SaezChristopher/flutter-layoff-a-good-news-for-reactnative-in-2024-415ba55c5751.

[41] *Songkick*. Último acceso: 21 de enero de 2025. 2025. URL: https://www.songkick.com/es.

[42] The New Stack. *What's Next for Flutter After Layoffs Hit Google Team*. 2024. URL: https://thenewstack.io/whats-next-for-flutter-after-layoffs-hit-google-team/.

[43] Coursera Staff. *What Is a Mockup?* 2025. URL: https://www.coursera.org/articles/what-is-mockup.

[44] Streamlit. *Streamlit st.segmented$_c$ontrol*. 2025. URL: https://docs.streamlit.io/develop/api-reference/widgets/st.segmented_control.

[45] Streamlit Inc. *Streamlit: The fastest way to build and share data apps*. 2025. URL: https://streamlit.io/.

[46] TeamGantt. *TeamGantt - Project Planning & Management*. 2025. URL: https://www.teamgantt.com/.

[47] uifrommars. *¿Cómo medir la usabilidad? Qué es el SUS (System Usability Scale)*. Abr. de 2022. URL: https://www.uifrommars.com/como-medir-usabilidad-que-es-sus/.

[48] Universidad de Valladolid. *GitLab UVa*. 2025. URL: https://gitlab.inf.uva.es/.

[49] Visual Paradigm. *Visual Paradigm - Herramienta de modelado UML*. 2025. URL: https://www.visual-paradigm.com/.

[50] *Watios y Decibelios*. Último acceso: 16 de enero de 2025. 2025. URL: https://watiosydecibelios.com/.

[51] *Wegow*. Último acceso: 21 de enero de 2025. 2025. URL: https://www.wegow.com/.

[52] Xiaomi. *Redmi Note 13 5G*. Ültimo acceso: 25 de enero de 2025. 2025. URL: https://www.mi.com/es/product/redmi-note-13-5g/.

Apéndice

Apéndice A

A. Acrónimos

A continuación, se presenta una lista de acrónimos utilizados a lo largo del proyecto:

TFG: Trabajo Fin de Grado

UVa: Universidad de Valladolid

R.R.S.S.: Redes Sociales

CSV: Comma-Separated Values (Valores Separados por Comas)

API: Application Programming Interface (Interfaz de Programación de Aplicaciones)

UI: User Interface (Interfaz de Usuario)

UX: User Experience (Experiencia de Usuario)

SQL: Structured Query Language (Lenguaje de Consulta Estructurado)

RGPD: Reglamento General de Protección de Datos

CCPA: California Consumer Privacy Act (Ley de Privacidad del Consumidor de California)

IDE: (Integrated Development Environment, en inglés). Es una aplicación de software que proporciona un conjunto completo de herramientas para el desarrollo de software

Apéndice B

B. Cuestionarios

En esta sección del apéndices se muestran los cuestionarios solicitados a los sujetos de prueba

Cuestionario Inicial user-persona

1. Nombre
2. Edad
3. Género
4. ¿Con que frecuencia sueles ir a eventos musicales?
    a) Nunca
    b) 1 vez al mes
    c) 2 veces al mes
    d) 3 o más
5. ¿Estas habituado a emplear aplicaciones musicales?
    a) Si
    b) No
6. ¿Estas habituado a emplear redes sociales?
    a) Si
    b) No
7. ¿Cuál es tu forma habitual de enterarte sobre eventos musicales?
    a) Por el boca a boca
    b) Por redes sociales
    c) Por aplicaciones especificas de conciertos
    d) Por carteles de propaganda
    e) Otra

Cuestionario para ambas aplicaciones

	Totalmente en desacuerdo	En desacuerdo	Neutral	De acuerdo	Totalmente de acuerdo
La app es fácil de aprender a usar.	◯	◯	◯	◯	◯
Me síento cómodo/a utilizando la app sin ayuda externa.	◯	◯	◯	◯	◯
Puedo completar mis tareas rapidamente usando la app.	◯	◯	◯	◯	◯
La app tiene un díseño visual atractivo.	◯	◯	◯	◯	◯
Es fácil encontrar la información que necesito.	◯	◯	◯	◯	◯
Las funciones de la app están bien organizadas.	◯	◯	◯	◯	◯
La app responde de forma rápida a mis acciónes.	◯	◯	◯	◯	◯
No he tenido problemas técnicos al usar la app.	◯	◯	◯	◯	◯
El lenguaje utilizado en la app es claro y comprensible.	◯	◯	◯	◯	◯
Me siento satisfecho/a con la experíencia general de uso.	◯	◯	◯	◯	◯
Las opciones y menús están ubicados de forma logica.	◯	◯	◯	◯	◯
Confio en que la app funcionará como espero.	◯	◯	◯	◯	◯

Figura 78: Cuestionario para usuarios

B.1. Cuestiones generales

1. ¿Consideras la aplicación útil?
2. ¿Cuál es la mayor ventaja de la aplicación?
3. ¿Y el mayor inconveniente?
4. ¿Agregarías alguna funcionalidad a la aplicación? En caso afirmativo, ¿cuál sería?

B.2. Cuestiones para usuario administrador

1. ¿Cuánto tiempo dedicabas semanalmente a la gestión de Watios y Decibelios?
2. ¿Cuánto tiempo te llevaba subir un evento a la web?
3. Al subir eventos, ¿solías publicar uno solo o varios a la vez?
4. ¿Cuáles eran las tareas más repetitivas o tediosas en el proceso de subir eventos?
5. ¿Había algún tipo de plantilla o automatización que utilizabas para agilizar el proceso?

6. ¿Con qué frecuencia actualizabas la información de los eventos ya publicados?
7. ¿Tuviste alguna vez problemas con errores o inconsistencias al publicar? ¿Cómo los resolvías?
8. ¿La gestión de la plataforma la llevabas tú solo o colaboraba más gente?

Apéndice C

C. Manual de despliegue

El sistema *WatiosyDecibelios* está compuesto por tres aplicaciones principales que se ejecutan mediante contenedores Docker:

- **Base de Datos (app-bd):** PostgreSQL 15 con datos inicializados
- **Aplicación Web (app-web):** Interfaz web desarrollada en Streamlit
- **Aplicación Móvil (app-movil):** Backend Node.js + Frontend React Native/Expo

El despliegue se ha automatizado completamente mediante scripts de PowerShell que garantizan la configuración correcta de redes, variables de entorno y dependencias entre servicios.

C.1. Prerrequisitos del Sistema

Antes de proceder con el despliegue, asegúrese de que el sistema cumple con los siguientes requisitos:

- **Docker Desktop:** Versión 4.0 o superior
- **Docker Compose:** Incluido en Docker Desktop
- **PowerShell:** Versión 5.1 o superior (Windows)
- **Git:** Para clonación de repositorios
- **Conexión a Internet:** Para descarga de imágenes Docker
- **Expo Go** Para correr las pruebas en dispositivo móvil (versión empleada SDK 52)
- **ngrok:** Para acceso desde dispositivos móviles físicos (`npm install -g ngrok`)

C.1.1. Puertos Requeridos

El sistema utiliza los siguientes puertos que deben estar disponibles:

- **5432** - PostgreSQL: Base de datos
- **8501** - Streamlit: Aplicación web
- **3000** - Node.js: API Backend móvil
- **19006** - Expo: Frontend móvil (web)
- **19000-19002** - Expo: Herramientas de desarrollo

C.2. Proceso de Despliegue Automatizado

El despliegue se realiza mediante dos scripts de PowerShell que deben ejecutarse en secuencia:

C.2.1. Script 0: Clonación de Repositorios

.\00-clone-repositorios.ps1

Este script realiza las siguientes operaciones:

- Clona el repositorio de la base de datos (`app-bd`)
- Clona el repositorio de la aplicación web (`app-web`)
- Clona el repositorio de la aplicación móvil (`app-movil`)

C.2.2. Script 1: Creación de archivos .env

`.\01-create-envs.ps1`

C.2.3. Script 2: Configuración y Despliegue

`.\02-ejecutar-docker.ps1`
Este script automatiza el proceso completo de configuración y despliegue:

- Configuración de variables de entorno para cada aplicación
- Verificación/creación de la red Docker `app-network`
- Configuración de comunicación entre contenedores
- Despliegue del contenedor PostgreSQL
- Inicialización automática con datos de prueba
- Conexión del contenedor a la red compartida
- Construcción de la imagen de la aplicación Streamlit
- Despliegue del contenedor web
- Configuración de conexión a base de datos
- Construcción de imágenes backend (Node.js) y frontend (Expo)
- Despliegue de contenedores móviles
- Configuración de APIs y variables de entorno
- Comprobación del estado de todos los contenedores
- Visualización de puertos y servicios disponibles

C.3. Configuración de Variables de Entorno

C.3.1. Aplicación Móvil (app-movil/.env)

- `DB_HOST=app_db`
- `DB_PORT=5432`
- `DB_NAME=watiosydecibelios`
- `DB_USER=root`
- `DB_PASSWORD=root`
- `EXPO_PUBLIC_API_URL=http://192.168.1.85:3000/api` (para red local)

Nota: Para dispositivos móviles físicos, modifique `EXPO_PUBLIC_API_URL` con la URL proporcionada por ngrok.

C.3.2. Aplicación Web (app-web/.env)

- `DB_HOST=app`$_d b$`DB_PORT=5432`
- `DB_NAME=watiosydecibelios`
- `DB_USER=root`
- `DB_PASSWORD=root`

C.3.3. Aplicación Móvil (app-movil/.env)

- `DB_HOST=app_db`
- `DB_PORT=5432`
- `DB_NAME=watiosydecibelios`
- `DB_USER=root`
- `DB_PASSWORD=root`

C.4. Verificación del Despliegue

C.4.1. Verificación de Contenedores

- Ejecutar `docker ps`
- Verificar que aparecen:
  - **app_db** - Base de datos PostgreSQL
  - **app-web-app-web-1** - Aplicación web Streamlit
  - **app_movil_backend** - API Node.js
  - **app_movil_frontend** - Frontend Expo

C.5. Configuración para Dispositivos Móviles Físicos

Para ejecutar la aplicación móvil en dispositivos físicos conectados por cable o en diferentes redes, se requiere configuración adicional:

C.5.1. Opción 1: Tunnel Mode (Recomendado)

La aplicación móvil está configurada para usar tunnel mode automáticamente, lo que permite conexión directa sin configuración adicional.

C.5.2. Opción 2: Usando ngrok

1. Instalar ngrok: `npm install -g ngrok`
2. Exponer el backend móvil: `ngrok http 3000`
3. Copiar la URL proporcionada (ej: `https://abc123.ngrok-free.app`)
4. Actualizar `app-movil/.env`:
   - `EXPO_PUBLIC_API_URL=https://abc123.ngrok-free.app/api`
5. Reiniciar contenedores móviles:
   - `cd app-movil`
   - `docker-compose restart`

Importante: La URL de ngrok cambia en cada sesión, por lo que debe actualizarse el archivo `.env` cada vez que se reinicie ngrok.

C.5.3. Acceso a las Aplicaciones

- Aplicación Web: http://localhost:8501
- API Backend: http://localhost:3000
- Frontend Móvil: http://localhost:19006
- Base de Datos: `localhost:5432`
- Desde expo start (emuladores): URL: exp://[IP de equipo]:19000
- **Dispositivos móviles físicos:** Usar tunnel mode automático o ngrok

C.6. Comandos de Gestión del Sistema

C.6.1. Parar el Sistema

- `cd app-movil`
- `docker-compose down`
- `docker network connect app-network app`$_d b2 > \$null$`cd ..`
- `cd app-web`
- `docker-compose down`
- `cd ..`
- `cd app-bd`
- `docker-compose down`
- `cd ..`

C.6.2. Limpieza Completa del Sistema

- Ejecutar: `.\03-limpiar-docker.ps1`
- Elimina:
  - Todos los contenedores del sistema
  - Imágenes Docker construidas
  - Volúmenes de datos
  - Redes personalizadas
  - Caché de construcción

C.7. Consideraciones futuras

- Las credenciales por defecto (`root/root`) son solo para desarrollo
- En producción se debe usar autenticación robusta
- Realizando pruebas en ocasiones el docker de la bd no iniciaba correctamente debido a que el puerto 5432 ya estaba ocupado por postgres. Esto podría requerir o bien cambiar los puertos de nuestra base de datos o ejecutar el comando que pare el otro servicio. En mi caso sudo systemctl stop postgresql.
- **Para dispositivos móviles físicos:** El sistema utiliza tunnel mode por defecto. Como alternativa, se puede usar ngrok para exponer el backend, modificando la variable `EXPO_PUBLIC_API_URL` en el archivo `app-movil/.env`

- **Conectividad ngrok:** Las URLs de ngrok son temporales y deben actualizarse en cada sesión. Para entornos de desarrollo permanentes, considere configurar un dominio personalizado en ngrok o usar servicios de túnel alternativos.

Cambiar URL de API para ngrok:

- Editar: `app-movil/.env`
- Modificar: `EXPO_PUBLIC_API_URL=https://nueva-url.ngrok-free.app/api`
- Reiniciar: `cd app-movil && docker-compose restart`

Verificar configuración de API:

- `docker exec app_movil_frontend printenv | findstr EXPO_PUBLIC_API_URL`

Apéndice D

D. Manuel de uso

Esta sección se dimirá en dos secciones principales, la sección en la que se explicara el manual de uso de la aplicación web y en la aplicación para dispositivos móviles.

D.1. Manuel de uso de aplicación movil

La primera vista que podemos observar al acceder a nuestra aplicación movil es la splash screen que se muestra en la figura 79, la cual funciona a modo de pantalla de bienvenida.

Figura 79: Splash screen inicial de la aplicación móvil

Figura 80: Modal en el que se muestran los filtros compuestos

Después de un par de segundos, esta vista se desvanece y se muestra la vista principal Home, que en este caso es la vista encargada de mostrar el contenido de los eventos próximos.

En la parte superior de la interfaz se presenta un pequeño logo de watios a decibelios. Debajo de este se muestra la barra de búsqueda que contiene la opción de aplicar filtros, representada con el logo de las tres lineas horizontales. En esta, podemos elegir entre buscar a trabes de la barra de búsqueda simple o aplicar los filtros de búsqueda compuesta, permitiendo indexar entre diferentes campos simultáneamente (población, precio y fecha del concierto). Este modal esta representado en la figura 80.

Debajo de los filtros se encuentran tres botones que permiten seleccionar el tipo de evento que se desea visualizar en pantalla. Las opciones disponibles son: conciertos, giras y festivales.

A continuación, en la sección principal de la vista, se muestra un listado vertical de eventos próximos, organizados de forma cronológica, desde los más cercanos a los más lejanos en el tiempo.

Cada tarjeta de evento presenta información básica según su tipo:

- **Conciertos**: Se muestra el nombre del evento, fecha y hora, población y lugar de celebración, así como el rango de precios.
- **Giras**: Se presenta el nombre de la gira y el rango de fechas en las que se desarrollará.
- **Festivales**: Se indica el nombre, el rango de fechas del festival y el rango de precios.

en las figuras siguientes se muestra el contenido que se muestra en los conciertos en la figura 81, giras en la figura 82 y festivales
en la figura 83

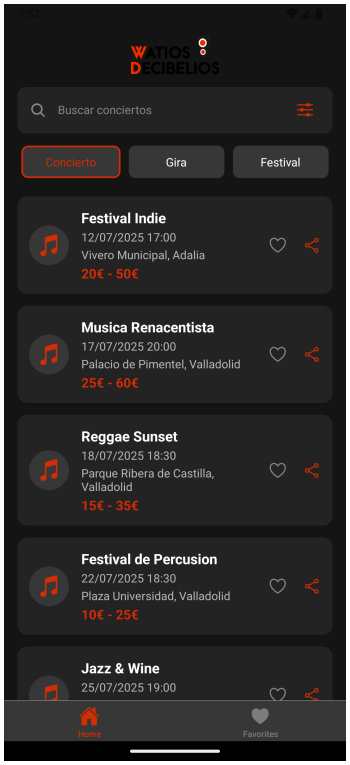


Figura 81: Vista de display de los conciertos

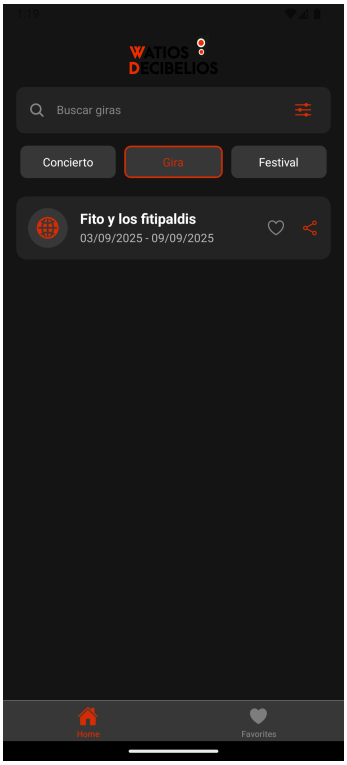


Figura 82: Vista de display de las giras

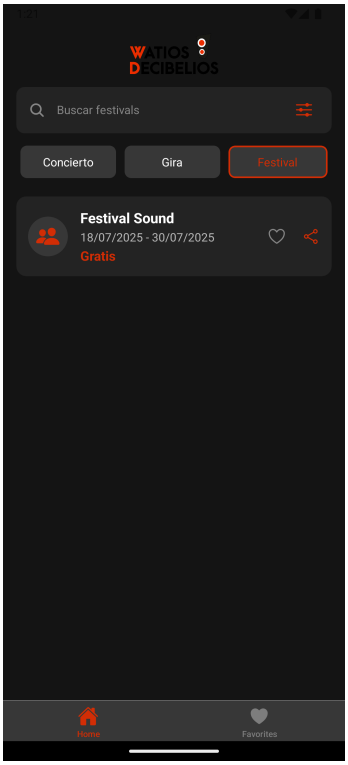


Figura 83: Vista de display de los festivales

Figura 84: Vista de Home

Cada una de las tarjetas mostradas en la sección anterior despliega contenido adicional al ser pulsada, proporcionando información detallada sobre los eventos.

- **Conciertos**:Se muestra una imagen representativa (si está disponible), junto con opciones para compartir el evento y añadirlo o quitarlo de la lista de favoritos. También se incluyen la fecha y hora, población y lugar de celebración, rango de precios, enlaces tanto para la compra de entradas como para visualizar contenido del grupo, el usuario organizador del evento y, finalmente, información adicional relevante. Podemos observar como se muestra en la figura 85.

- **Giras**: Se presenta el cartel promocional, opciones para compartir y gestionar favoritos, el rango de fechas de la gira, nombre del organizador, información adicional y una lista de los conciertos que forman parte de la gira. Esta vista se muestra en la figura 86.
- **Festivales**: La información mostrada es la misma a la de las giras, añadiendo además el rango de precios del festival. Podemos observar estos datos en 87.

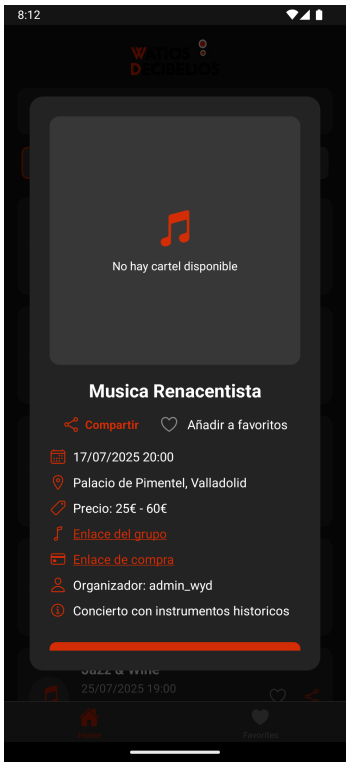


Figura 85: Vista de características de concierto específico

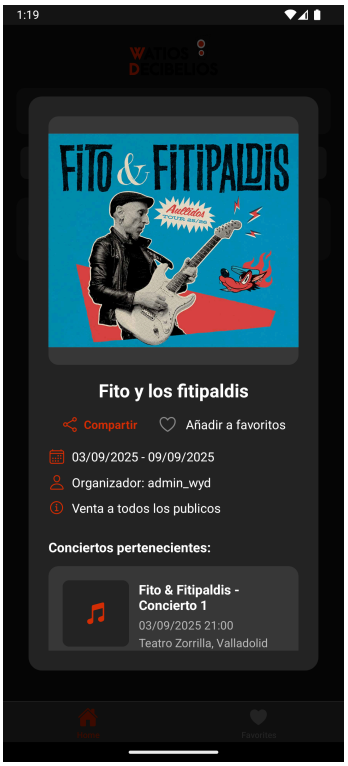


Figura 86: Vista de características de gira específica

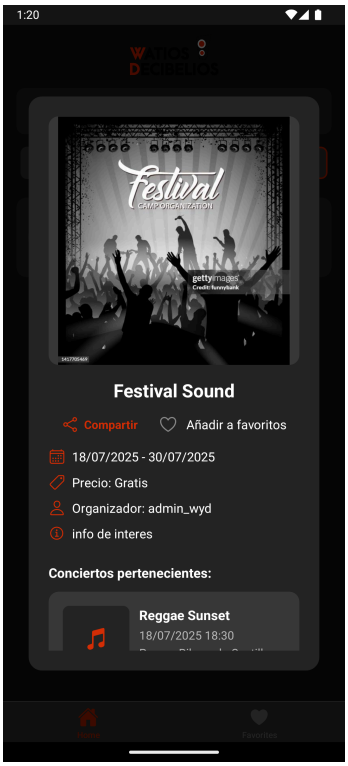


Figura 87: Vista de características de festival específico

Figura 88: Vista de características adicionales

En la parte inferior de la vista principal home podemos observar que hay una botom navigation bar, la cual nos muestra la opción Home (Icono de una casa) y por otro lado la opción favorites (Icono de una casa).

En caso de pulsar el icono de la casa navegaremos a la sección favoritas en la cual se muestra de la misma forma que en home el conjunto de eventos que hayamos marcado como favoritos.

En este caso, al no tener eventos favoritos, como se muestra en la figura que representa las giras favoritas 90

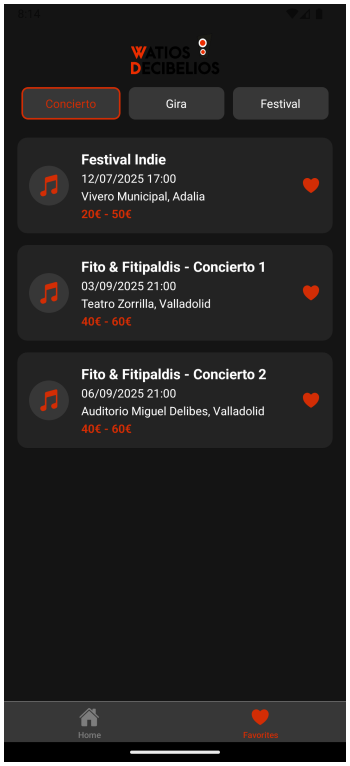


Figura 89: Vista de display de los conciertos favoritos

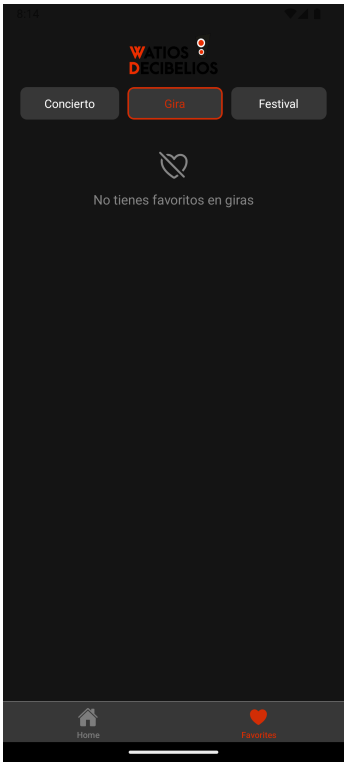


Figura 90: Vista de display de las giras favoritas

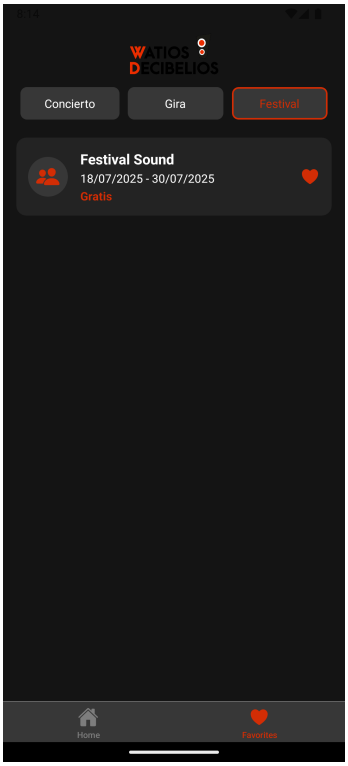


Figura 91: Vista de display de los festivales favoritos

Figura 92: Vistas de favoritos

D.2. Manuel de uso de aplicación web

Al acceder a la aplicación web lo primero que podemos observar es la pantalla de login. Esta permite a los usuarios que ya tengan una cuenta pueda acceder con sus credenciales y en caso de no tenerla permite a estos solicitarla pulsando la opción Contactanos abriendo así el gestor de correos y permitiendo enviar al propietario de la app un mensaje solicitando credenciales de uso. Esta vista se muestra en la figura 93

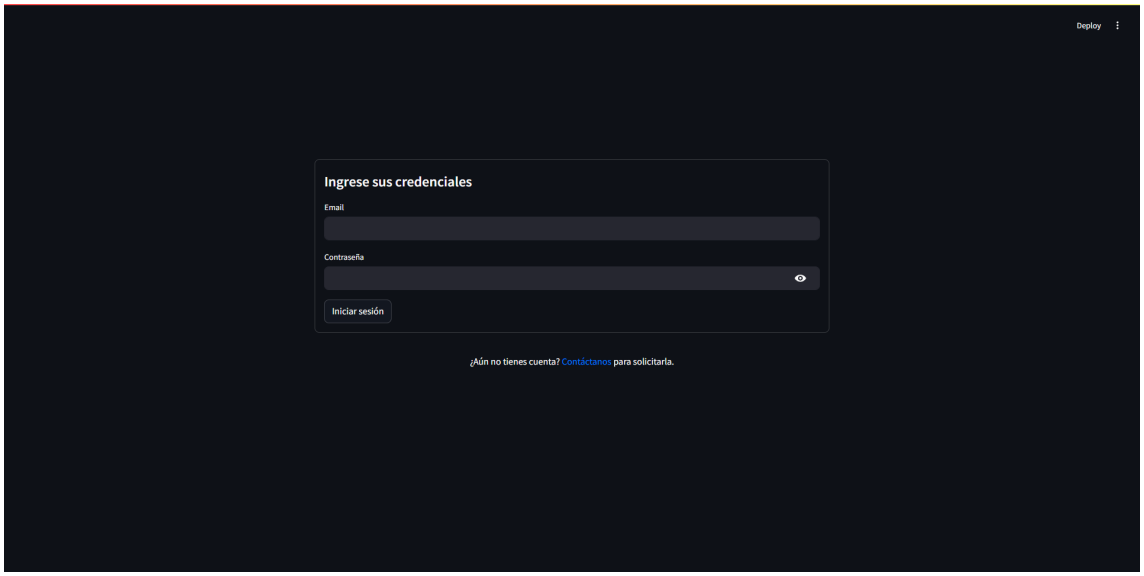


Figura 93: Login de la página web

Tras introducir nuestras credenciales existen dos opciones. En caso de que el usuario tenga permisos de administrador este tendrá las opciones de administrador en la sección izquierda en el side bar menu como se muestra en la figura 94. Estas son agregar eventos a través de formularios, agregar conciertos a de forma masiva a través de un csv, revisar las propuestas por usuarios externos, revisar los conciertos disponibles, crear nuevos usuarios, modificar cuenta y cerrar sesión.

En ambos casos, el usuario puede cerrar sesión pulsando el botón, de este modo el usuario se deslogueará inmediatamente y navegara a la vista de login anteriormente comentada.

Figura 94: Vista home de administrador

En caso de que este no tenga los permisos entonces las opciones que tendrá este serán algo mas reducidas permitiéndonos proponer evento a través de un formulario, proponer conciertos a través de un csv, ver propuestas propias, modificar cuenta y cerrar sesion. Estas se muestran en la

siguiente figura95.

Figura 95: Vista home de externo

A partir de ahora se definirán las funciones de administrador ya que ha nivel de usabilidad las diferencias que existen entre unas y otras son muy leves.

D.2.1. Creación de eventos a través de formulario

Dentro de las opciones que tenemos, la primera es la creación de nuevos eventos a través de formularios. Dentro de este tenemos en la parte un botón que permite seleccionar el tipo de evento que queremos agregar, siendo las opciones concierto, festival y gira. Estos formularios permiten introducir los campos principales de los eventos los cuales son:

- **Conciertos**: Imagen representativa (si está disponible), fecha y hora, población, lugar de celebración, rango de precios, enlace de compra, enlace a contenido del grupo e información adicional.
- **Festivales**: Cartel promocional, información adicional, lista de conciertos asociados y rango de precios.
- **Giras**: Cartel promocional, rango de fechas, información adicional y lista de conciertos asociados.

Los formularios de conciertos, festivales y giras se muestran respectivamente en las figuras 96, 97 y 98

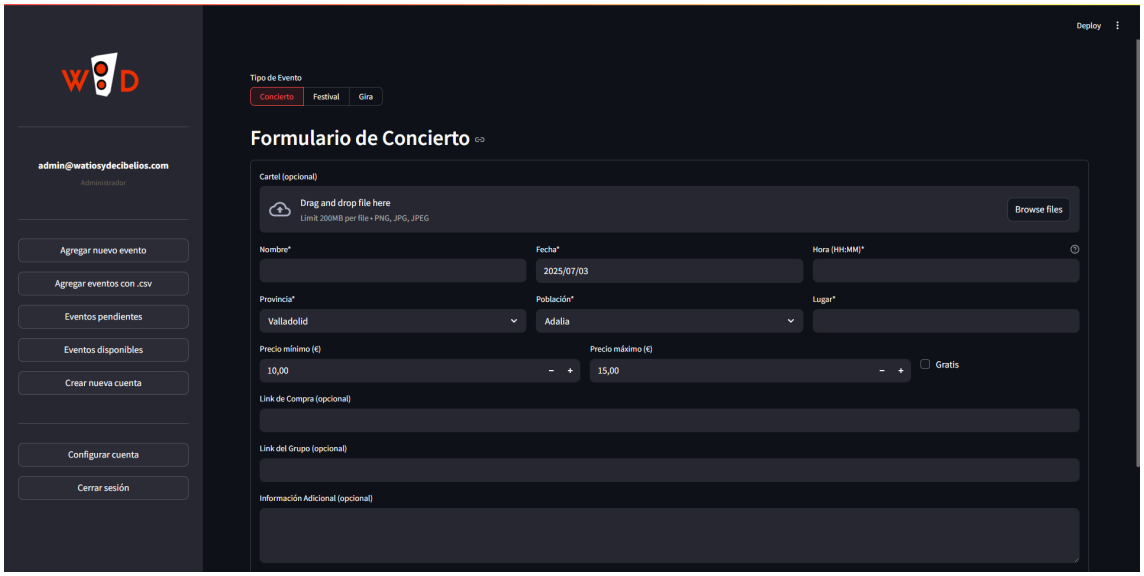


Figura 96: Formulario para agregar conciertos

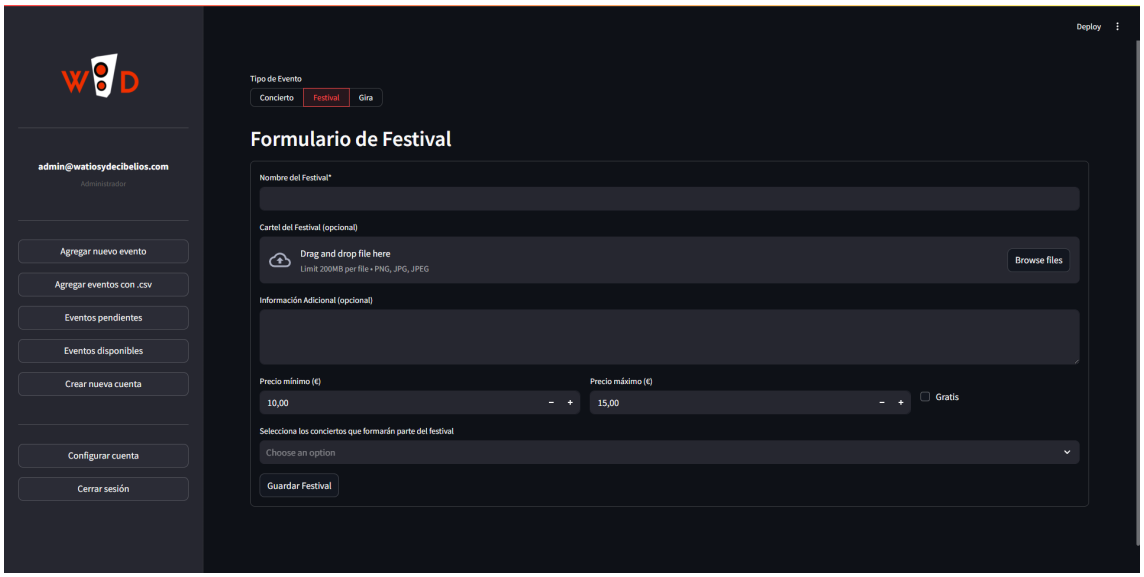


Figura 97: Formulario para agregar festivales

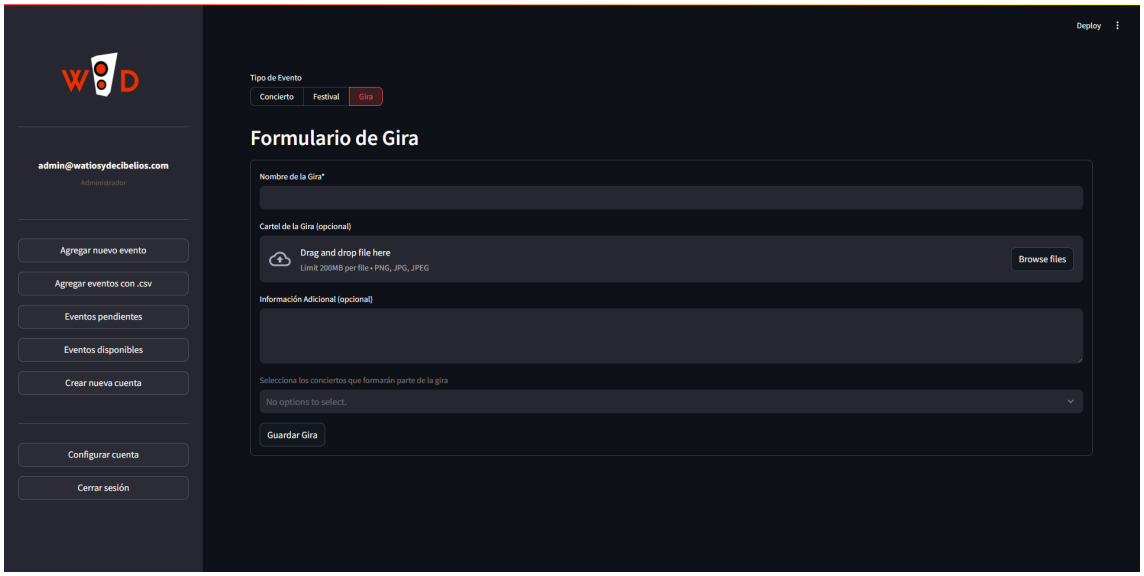


Figura 98: Formulario para agregar giras

D.2.2. Agregar varios conciertos de forma simultáneamente

La siguiente opción que se presenta en el side bar menu consiste en la opción de subir varios conciertos de manera simultáneamente a través de un archivo csv. Podemos observar esta vista en la figura 99

En esta vista podemos observar que se nos muestra un desplegable que al ser clicado indica de manera breve los datos que contiene cada concierto y podemos descargar un fichero el cual nos sirve como template o plantilla.

Posteriormente se muestra un campo para subir archivos. Al pulsar este, se nos avre una ventana del sistema de carpetas y nos permite elegir el fichero que se pretende cargar.

Tras esto, se comprueba si los datos son validos o no, en caso de no serlo nos indica linea y error. Finalmente, nos muestra la opción de aceptar o cancelar la subida de los datos. En caso afirmativo se nos muestra un mensaje de confirmación y se restablece la vista.

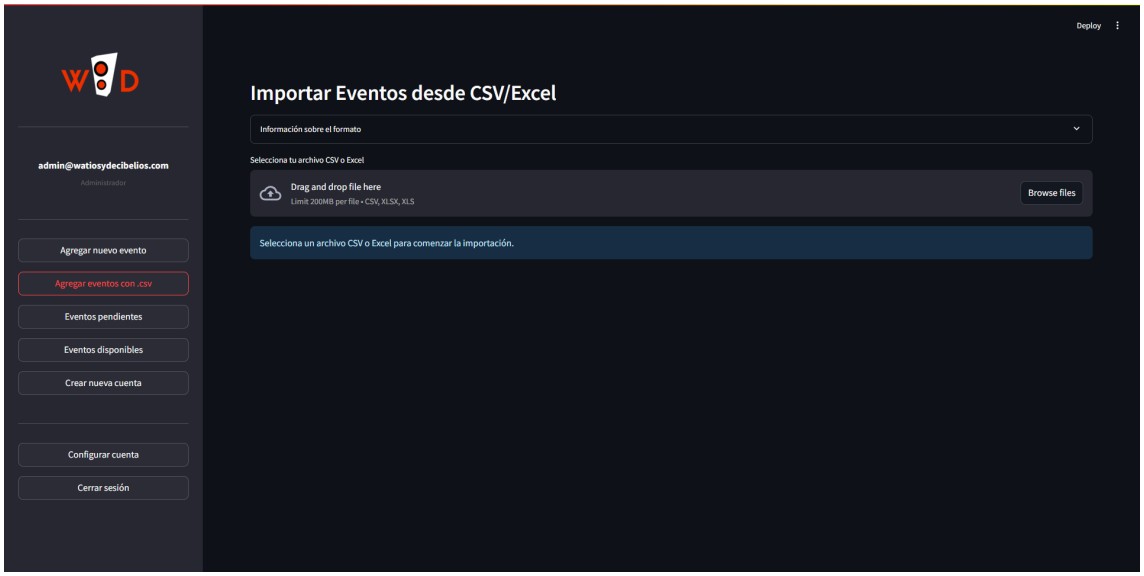


Figura 99: Vista de agregado de conciertos de forma masiva

D.2.3. Gestionar eventos pendientes

La siguiente opción es unicamente para los usuarios administradores.

En esta vista se presenta un listado con los eventos propuestos por los externos. Para ver los datos de estos y poder aceptarlos el usuario debe desplegar las tarjetas que representan cada uno de los eventos.

Posteriormente después de desplegar la vista completa del evento se presentan al usuario las opciones de aceptar, rechazar y modificar estas propuestas.

A continuación en la figura se mostrará la vista de conciertos pendientes de aprobación, en la figura 100, y también se mostrara la vista desplegada de un festival en la figura 101.

En este caso las vistas entre conciertos giras y festivales son muy similares a nivel funcional, por lo tanto se ha considerado redundante mostrar un ejemplo de cada listado y edición por tipo de evento

Figura 100: Vista en la que se muestra el listado de conciertos pendientes

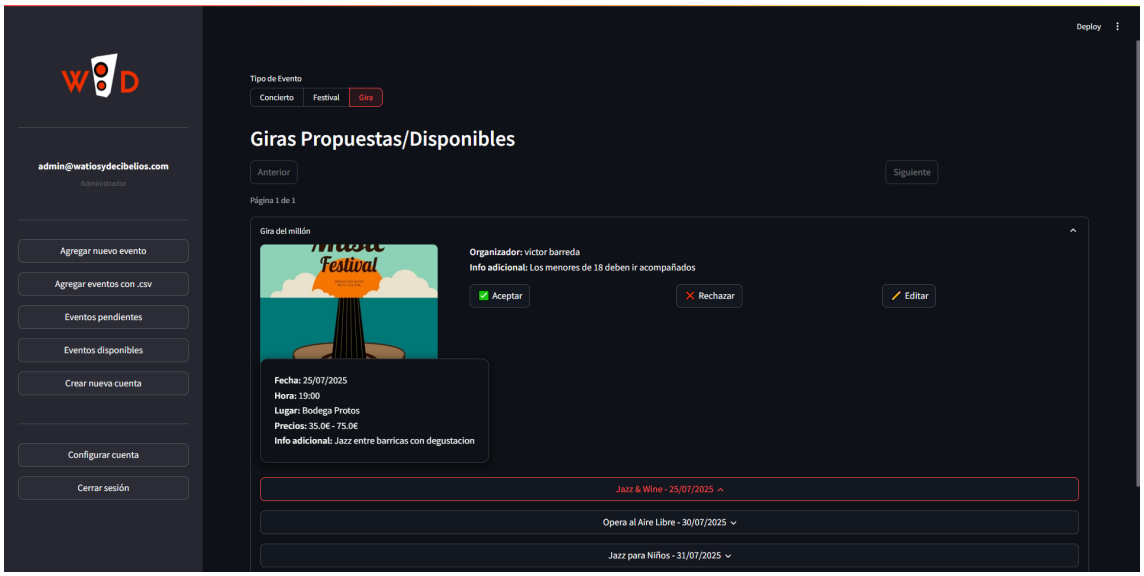


Figura 101: Vista en la que se muestra el desplegable de un festival pendiente

D.2.4. Gestionar eventos disponibles

La siguiente opción es unicamente para los usuarios administradores.

Esta permite a los usuarios revisar los eventos disponibles (Es decir, que o bien han sido agregados por otros administradores, o que han sido propuestos por externos y aceptados por administradores). Dentro de esta vista se nos muestra un listado con los eventos agrupados de 10 en 10 y donde se ha empleado paginación.

Aquí se muestra pequeñas tarjetas desplegables las cuales al clicar nos muestra los diferentes datos en función de el tipo de evento que estemos revisando. Además de esto se brinda al usuario la opción de editar o cancelar el concierto, en cuyo caso el concierto dejaría de figurar en la lista

de disponibles, ya que sufriría un cambio de estado.

A continuación se mostrara la lista de conciertos en la figura 102 y el desplegable de las giras en la figura 103. En este caso las vistas entre festivales, giras y conciertos son muy similares con la diferencia que al desplegar las giras y festivales también muestran los datos principales de los conciertos que contienen.

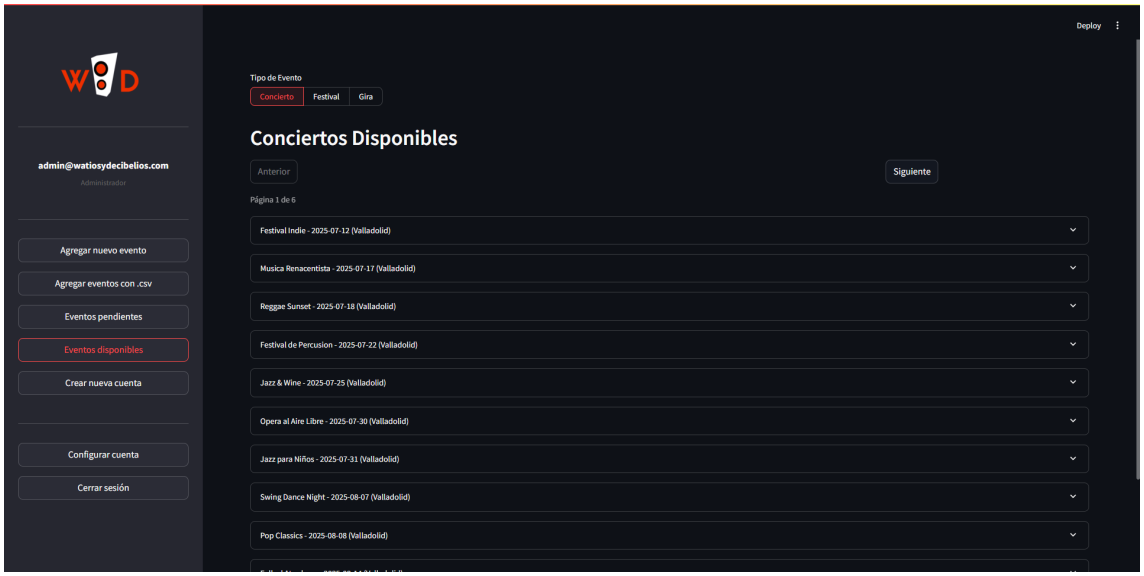


Figura 102: Vista en la que se muestra el listado de conciertos disponibles

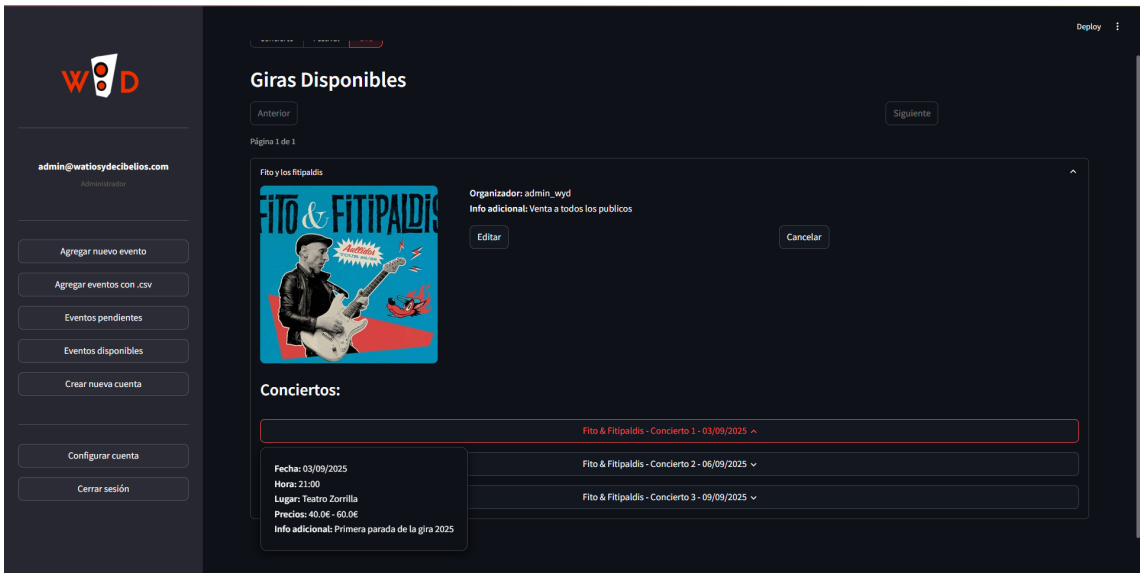


Figura 103: Vista en la que se muestra el desplegable de una gira disponible

D.2.5. Modificación de eventos

En las dos secciones anteriores se ha comentado que se pueden realizar modificaciones sobre eventos de diferentes maneras. En ambos casos lo que se realizar es mostrar un formulario muy

similar al de adición visto en la sección D.2.1 del apéndice.

Los formularios de edición se muestran a continuación representando la figura 104 el formulario de conciertos, la figura 105 el formulario de giras y por ultimo la figura 106 el formulario de festivales.

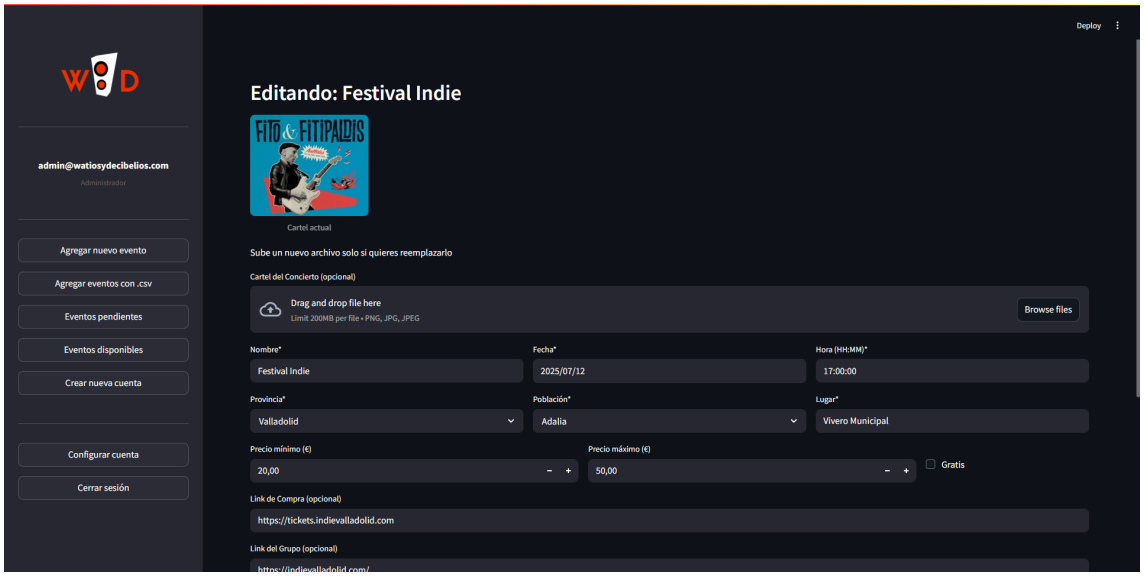


Figura 104: Vista de formulario de edición de conciertos

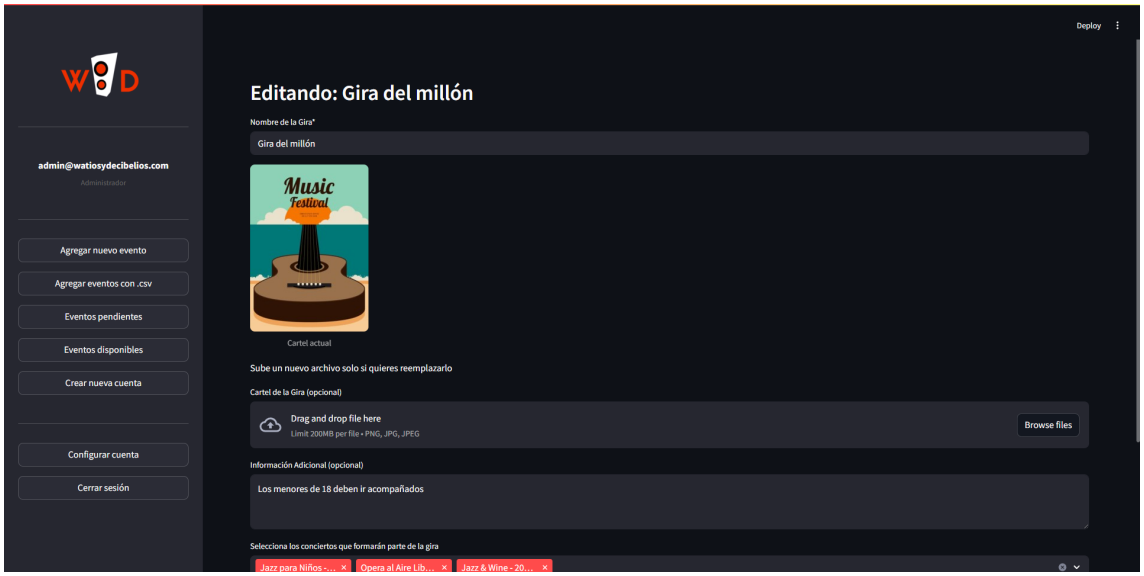


Figura 105: Vista de formulario de edición de giras

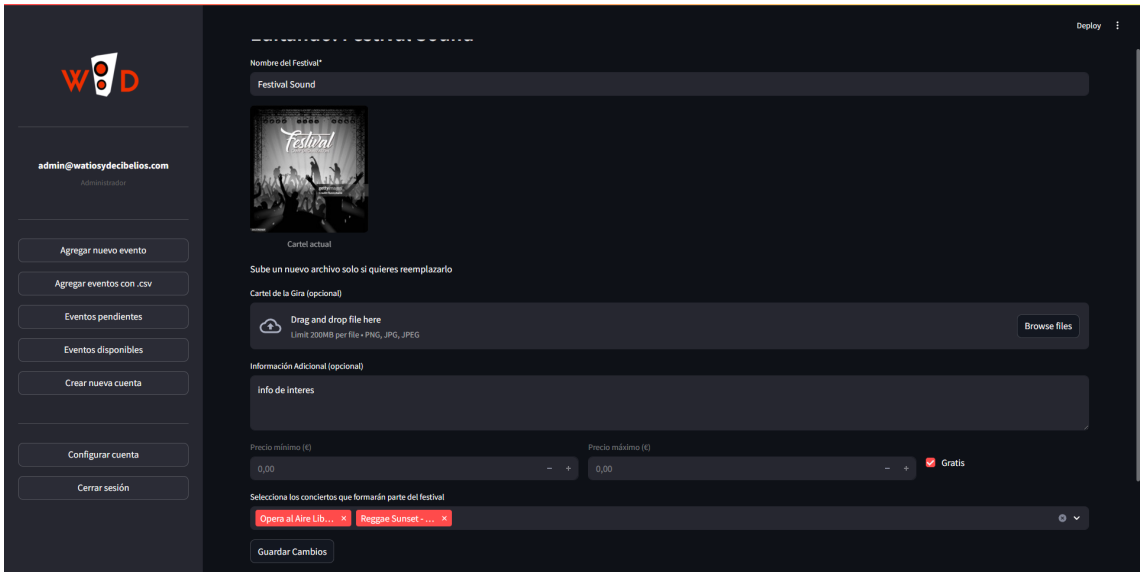


Figura 106: Vista de formulario de edición de festivales

D.2.6. Creación de cuenta

La siguiente opción es unicamente para los usuarios administradores.

En esta vista se permite a los usuarios administradores crear nuevos usuarios eligiendo si brindarles permisos de administrador, generado así un usuario administrador, o no, creando así usuario externo. La interfaz implementada para realizar esta labor ha sido la que se muestra en la figura 107

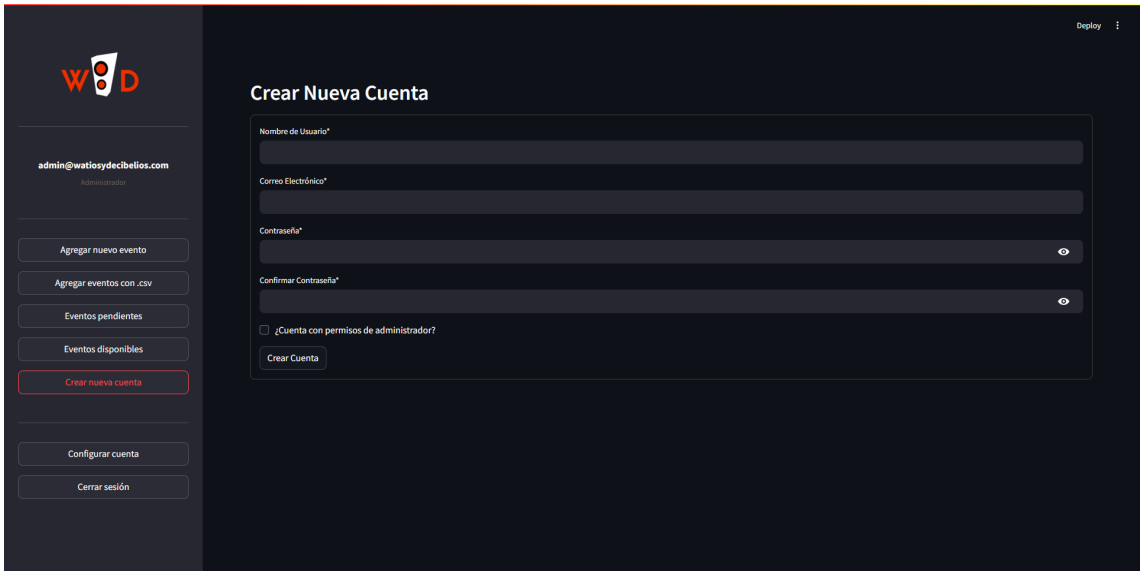


Figura 107: Vista de formulario de creación de una cuenta de usuario

D.2.7. Vista de modificacion de cuenta

Esta opcion permite a usuarios, tanto administradores como externos realizar cambios sobre los datos de su cuenta permitiendo a estos cambiar tanto el apodo como se muestra en la figura

108 como la contraseña de la cuenta como se muestra en la figura 109.

Figura 108: Vista de sección de configuración de nombre de usuario

Figura 109: Vista de sección de configuración de contraseña de usuario

D.2.8. vista de revision de los eventos propuestos por el externo

La siguiente opción es unicamente para los usuarios externos.

En este caso los usuarios externos pueden acceder a una revision de sus eventos independientemente de en que estado este. Esta vista consiste en un display muy similar al de eventos disponibles o eventos propuestos con la ligera diferencia de que en cada concierto se nos indica el estado de este como se muestra en la figura 110

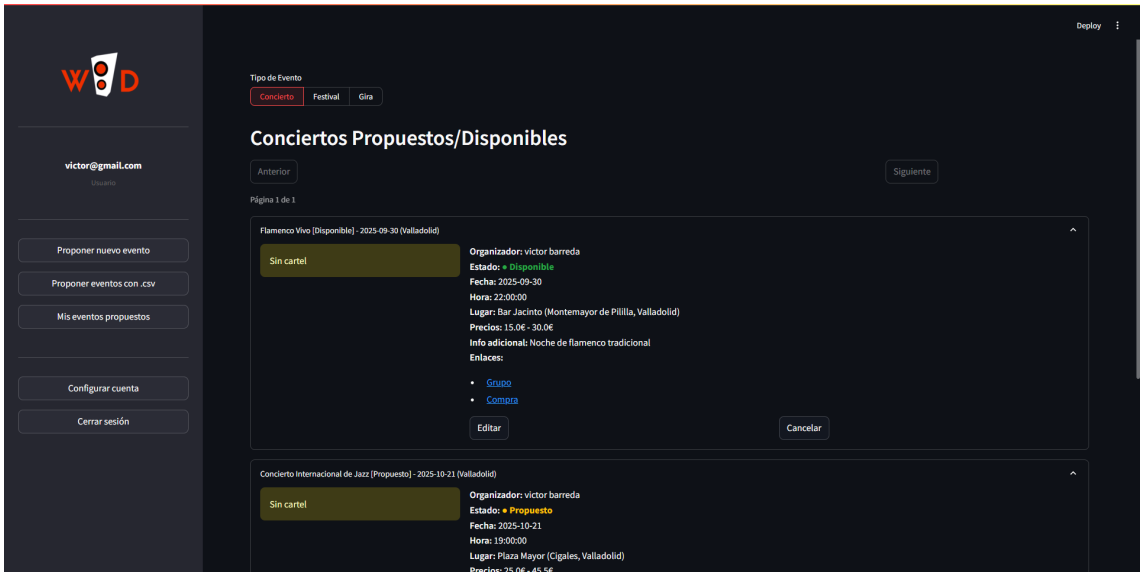


Figura 110: Vista de sección de revisión de eventos propuestos por usuario administrador