TFG-esports-calendar.

Aplicación de calendario de eventos competitivos de videojuegos.

1. Análisis del entorno y requisitos

El sector de los esports y los videojuegos competitivos ha crecido de forma exponencial en los últimos años. Existen múltiples plataformas que muestran eventos, pero muchas de ellas no centralizan bien la información, o quizá no ofrecen notificaciones personalizadas o tienen un seguimiento sencillo desde móvil

Esta aplicación nace con el objetivo de centralizar eventos competitivos de videojuegos y permitir al usuario recibir notificaciones relevantes antes de que ocurran..

Tecnologías utilizadas

Backend: Django + Django REST Framework

Frontend Web: React Native

Aplicación móvil: Android (Material 3)

API externa: PandaScore API

Diseño del prototipo: Figma

2. Público objetivo (Persona)

Edad: 18–30 años

Perfil: Jóvenes aficionados a los videojuegos competitivos que siguen torneos de esports (LoL, CS2, Valorant, etc.), consume contenido desde el móvil y le interesa recibir recordatorios de eventos importantes.

Dificultades y necesidades:

- Poder filtrar eventos por tipo o videojuego
- Recibir notificaciones antes de que empiece un evento

3. Diseño de la aplicación

Paleta de colores y estilo

Colores principales: tonos azules degradados, junto a grises y blancos. Cada tipo de evento se identifica con un color diferente en el calendario. Diseño orientado a reducir la carga visual y que sea fácil de observar.

Figma del diseño:
https://www.figma.com/design/ITsfrafGV7VT7wmIegFu5Y/Sin-t%C3%ADtulo?node-id=0-1&t=3bEpfzfTARzgS5hB-1

Tipografía: Inter / sans-serif moderna

Paradigma de diseño: Material Design 3

En Android uso de:

- Cards para eventos, Floating Buttons para acciones principales, Bottom Navigation para navegación principal, futura implementación de feedback…

Componentes y diseño general:

Pantalla de Login / Registro

Vista principal con calendario

Listado de eventos del día

Detalle del evento

Pantalla de ajustes y notificaciones, y quizá de perfil

4. Historias de usuario

Registro de usuario: como usuario, quiero registrarme en la aplicación para guardar mis preferencias y recibir notificaciones.

API: POST /api/auth/register

Inicio de sesión: para acceder a mi calendario personalizado.

API: POST /api/auth/login

Ver eventos en el calendario: para ver los eventos competitivos en un calendario para saber cuándo se celebran.

API: GET /api/events

(Datos obtenidos y sincronizados desde API de PandaScore)

Filtrar eventos por tipo o videojuego: filtrar los eventos por videojuego o categoría para encontrar los que me interesan.

API: GET /api/events?game=lol&type=tournament

Recibir notificaciones: recibir notificaciones antes de que empiece un evento para no olvidarlo.

API: POST /api/notifications/subscribe

5. Flujo de la aplicación

1. Pantalla de bienvenida
2. Registro / Login
3. Vista principal (Calendario)
4. Selección de evento
5. Detalle del evento
6. Activación de notificaciones

Notificación que notifique 1 día antes y/o 1 hora antes

6. Arquitectura de la aplicación

Arquitectura general

Arquitectura cliente-servidor

API REST como intermediario entre frontend y backend

Backend (Django)

Consumo de la API de PandaScore

Persistencia de eventos relevantes

Gestión de usuarios y notificaciones

Exposición de endpoints REST

Frontend

React native: consumo de la API REST

Android:

Activities principales

Fragments para navegación

RecyclerView para listas de eventos

CalendarView / componente personalizado de calendario

7. Modelo Entidad–Relación (Backend)

Entidades principales

Usuario

id

username

email

password

created_at

Evento

id

external_id (ID de PandaScore)

nombre

videojuego

tipo_evento

fecha_inicio

fecha_fin

color

created_at

Notificación (para permitir gestionar las notificaciones programadas por usuario y evento.)

id

usuario_id (FK)

evento_id (FK)

tipo_notificacion (1 día antes / 1 hora antes)

enviada (boolean)

Preferencia (para guardar filtros y gustos del usuario.)

id

usuario_id (FK)

videojuego

color_asociado

Relaciones

Usuario 1 — N Notificación

Evento 1 — N Notificación

Usuario 1 — N Preferencia

Usuario < Notificación > Evento

└— < Preferencia

Evento puede estar asociado a muchos usuarios mediante notificaciones.

Este modelo entidad-relación es escalable y encaja con Django ORM para poder usar correctamente la API externa junto a la lógica propia.