Taller de Nuevas Tecnologías
Año 2025
Giuliano Ignacio Poeta

Documentación Completa del Proyecto TPI-TNT

Introducción y Visión General

El proyecto **Pixdex** representa una aplicación móvil completa desarrollada con **React Native** y **Expo**, que combina un juego interactivo de **Hangman** con un sistema de **scoreboard en tiempo real**. Esta aplicación demuestra la implementación de tecnologías modernas de desarrollo móvil, autenticación robusta, gestión de estado avanzada y comunicación en tiempo real.

La aplicación está diseñada para funcionar como una plataforma de entretenimiento donde los usuarios pueden registrarse, autenticarse mediante múltiples métodos, jugar al clásico juego del ahorcado con contenido audiovisual, y competir en un ranking global que se actualiza instantáneamente. El proyecto utiliza **Supabase** como backend-as-a-service para manejar la autenticación, base de datos y funcionalidades en tiempo real.

Arquitectura del Sistema

Estructura General

La aplicación sigue una arquitectura modular basada en **componentes** y **contextos de React**, implementando patrones de diseño modernos para aplicaciones móviles. La estructura del proyecto está organizada de manera que cada módulo tiene responsabilidades claras y bien definidas.

El sistema se divide en varias capas principales: la **capa de presentación** que maneja la interfaz de usuario, la **capa de lógica de negocio** que contiene la lógica del juego y la gestión de datos, la **capa de servicios** que se comunica con APIs externas, y la **capa de datos** que gestiona el almacenamiento y persistencia.

Tecnologías Principales

React Native con Expo forma la base del desarrollo móvil, proporcionando una experiencia de desarrollo rápida y eficiente. Expo facilita el desarrollo multiplataforma y ofrece herramientas integradas para el manejo de assets, navegación y configuración del proyecto.

TypeScript se utiliza en todo el proyecto para proporcionar tipado estático, mejorando la calidad del código, facilitando el mantenimiento y reduciendo errores en el tiempo de ejecución. La implementación de interfaces y tipos personalizados asegura la consistencia de datos en toda la aplicación.

Supabase actúa como el backend principal, proporcionando servicios de autenticación, base de datos PostgreSQL, y funcionalidades en tiempo real. La integración con Supabase permite un desarrollo rápido sin necesidad de configurar servidores tradicionales.

Patrones de Diseño Implementados

El proyecto implementa el **patrón de Context API** de React para la gestión global del estado, separando claramente las responsabilidades entre autenticación y datos. Este patrón permite que los componentes accedan a datos compartidos sin necesidad de prop drilling.

El **patrón de servicios** se utiliza para encapsular la lógica de comunicación con APIs externas, proporcionando una interfaz limpia y reutilizable para operaciones de datos. Cada servicio maneja un dominio específico de la aplicación.

La **arquitectura de navegación** utiliza Expo Router para implementar navegación basada en archivos, proporcionando una experiencia de usuario fluida y una estructura de código organizada.

Sistema de Autenticación

Implementación de Autenticación

El sistema de autenticación está construido utilizando **Supabase Auth**, que proporciona una solución robusta y escalable para la gestión de usuarios. La implementación incluye múltiples métodos de autenticación para maximizar la accesibilidad y comodidad del usuario.

Autenticación por Email y Contraseña permite a los usuarios registrarse e iniciar sesión utilizando credenciales tradicionales. El sistema incluye validación de formularios, verificación de email, y recuperación de contraseñas.

Autenticación OAuth integra proveedores externos como Google y GitHub, permitiendo a los usuarios autenticarse utilizando sus cuentas existentes. Esta funcionalidad mejora la experiencia de usuario y reduce la fricción en el proceso de registro.

Gestión de Estado de Autenticación

El estado de autenticación se gestiona a través del **AuthContext**, un contexto de React que proporciona acceso global a la información del usuario y métodos de autenticación. Este contexto maneja automáticamente la persistencia de sesiones y la sincronización del estado.

La aplicación implementa **protección de rutas** basada en el estado de autenticación, redirigiendo automáticamente a los usuarios no autenticados a las pantallas de login. Esta funcionalidad asegura que solo los usuarios autenticados puedan acceder a las funcionalidades principales.

Verificación de Email

El sistema incluye **verificación obligatoria de email** para usuarios registrados con email y contraseña. Esta funcionalidad mejora la seguridad y permite a la aplicación enviar notificaciones importantes a los usuarios.

Los usuarios de OAuth se consideran automáticamente verificados, ya que estos proveedores ya han validado la identidad del usuario. Esta distinción permite una experiencia de usuario fluida mientras mantiene la seguridad.

Base de Datos y Gestión de Datos

Esquema de Base de Datos

La base de datos PostgreSQL en Supabase está diseñada para soportar las funcionalidades principales de la aplicación. El esquema incluye tablas para usuarios, jugadores, contenidos audiovisuales, géneros y tipos de contenido.

La **tabla de jugadores** almacena información sobre los usuarios del juego, incluyendo nombres, puntajes y referencias a usuarios autenticados. Esta tabla implementa políticas de seguridad Row Level Security para proteger los datos de los usuarios.

Las **tablas de contenido** almacenan información sobre series, películas y anime que se utilizan en el juego. Esta estructura permite una gestión eficiente del contenido y facilita la expansión futura del catálogo.

Políticas de Seguridad

Row Level Security (RLS) se implementa en todas las tablas para asegurar que los usuarios solo puedan acceder a datos apropiados. Las políticas están configuradas para permitir lectura pública de ciertos datos mientras protegen información sensible.

Las políticas de seguridad se configuran para permitir que usuarios autenticados lean datos del scoreboard, mientras que solo los propietarios pueden modificar o eliminar sus propios registros. Esta configuración equilibra la funcionalidad con la seguridad.

Sincronización en Tiempo Real

La aplicación utiliza las **funcionalidades de tiempo real de Supabase** para mantener el scoreboard actualizado instantáneamente. Los cambios en la tabla de jugadores se propagan automáticamente a todos los clientes conectados.

La implementación de suscripciones en tiempo real permite que múltiples usuarios vean actualizaciones del scoreboard sin necesidad de refrescar la aplicación. Esta funcionalidad mejora significativamente la experiencia de usuario y crea un sentido de comunidad.

Juego del Ahorcado

Lógica del Juego

El juego del ahorcado implementa la mecánica clásica del juego de adivinanza de palabras, adaptada para funcionar con títulos de contenido audiovisual. Los jugadores tienen un número limitado de intentos para adivinar las letras o el título completo.

La **selección de palabras** se realiza de manera aleatoria desde una base de datos de contenidos audiovisuales, incluyendo series, películas y anime. Esta variedad de contenido mantiene el juego interesante y educativo.

El sistema de **puntuación** recompensa a los jugadores por adivinar correctamente, con bonificaciones adicionales por adivinar el título completo de una vez. Esta mecánica incentiva tanto la estrategia como la toma de riesgos.

Interfaz de Usuario del Juego

La interfaz del juego está diseñada para ser intuitiva y atractiva visualmente. Los componentes incluyen un **teclado virtual** para seleccionar letras, un **modal para adivinar títulos completos**, y indicadores visuales del progreso del juego.

El **sistema de vidas** se representa visualmente con iconos de corazones, proporcionando retroalimentación inmediata sobre el estado del jugador. Esta representación visual hace que el juego sea más accesible y emocionante.

La **visualización de la palabra** muestra las letras adivinadas y oculta las no adivinadas con guiones bajos, permitiendo a los jugadores ver su progreso de manera clara. Esta funcionalidad es esencial para la mecánica del juego.

Gestión de Estado del Juego

El estado del juego se gestiona localmente en cada pantalla, utilizando hooks de React para manejar variables como letras adivinadas, vidas restantes, puntuación actual y palabra actual. Esta gestión local asegura un rendimiento óptimo.

La **transición entre palabras** se maneja automáticamente cuando un jugador adivina correctamente, proporcionando una experiencia fluida y continua. El sistema también maneja la finalización del juego cuando se agotan las vidas o se completan todas las palabras.

Sistema de Scoreboard

Implementación del Ranking

El scoreboard implementa un sistema de ranking global que muestra los mejores jugadores ordenados por puntuación. La lista se actualiza automáticamente en tiempo real cuando los jugadores completan partidas.

La **persistencia de puntuaciones** asegura que los logros de los jugadores se mantengan entre sesiones. El sistema utiliza el ID del usuario autenticado para asociar puntuaciones con cuentas específicas.

La **validación de nombres de jugador** previene duplicados y asegura que cada jugador tenga una identidad única en el scoreboard. Esta funcionalidad mantiene la integridad del sistema de ranking.

Actualizaciones en Tiempo Real

Las actualizaciones del scoreboard se manejan a través de **suscripciones de Supabase**, que escuchan cambios en la tabla de jugadores. Cuando un jugador actualiza su puntuación, todos los clientes conectados reciben la actualización automáticamente.

La implementación incluye **optimizaciones de rendimiento** para manejar múltiples actualizaciones simultáneas sin afectar la experiencia del usuario. El sistema ordena automáticamente la lista por puntuación después de cada actualización.

Interfaz del Scoreboard

La interfaz del scoreboard presenta la información de manera clara y atractiva, mostrando el nombre del jugador y su puntuación. El diseño utiliza colores y tipografía para destacar información importante.

Los **estados de carga y error** se manejan apropiadamente, proporcionando retroalimentación al usuario cuando los datos están cargando o cuando ocurren errores de conexión. Esta funcionalidad mejora la experiencia de usuario.

Componentes Compartidos

Biblioteca de Componentes

La aplicación incluye una biblioteca extensa de componentes reutilizables que mantienen consistencia visual y funcional en toda la aplicación. Estos componentes están diseñados para ser flexibles y configurables.

El **componente Button** proporciona una interfaz consistente para todas las acciones de la aplicación, con soporte para iconos, texto personalizable y estados de deshabilitado. Este componente asegura una experiencia de usuario coherente.

Los **componentes de texto** incluyen variantes especializadas como TextPressStart2P para títulos, proporcionando una identidad visual distintiva. Estos componentes encapsulan la lógica de tipografía y estilos.

Componentes de Estado

Los **componentes de estado** como LoadingState y ErrorState proporcionan retroalimentación visual consistente para diferentes estados de la aplicación. Estos componentes mejoran la experiencia de usuario al comunicar claramente el estado actual.

El **componente EmailVerificationBanner** alerta a los usuarios sobre el estado de verificación de su email, guiándolos hacia la verificación cuando es necesaria. Este componente es esencial para el flujo de autenticación.

Componentes de Navegación

Los **componentes de navegación** incluyen headers personalizados y botones de navegación que mantienen la consistencia visual. Estos componentes facilitan la navegación entre pantallas y proporcionan contexto al usuario.

Servicios y APIs

Arquitectura de Servicios

Los servicios de la aplicación están organizados en módulos específicos que manejan diferentes aspectos de la funcionalidad. Cada servicio encapsula la lógica de comunicación con APIs externas y proporciona una interfaz limpia para los componentes.

El **AuthService** maneja todas las operaciones relacionadas con autenticación, incluyendo registro, login, logout y reset de contraseñas. Este servicio abstrae la complejidad de la API de Supabase Auth.

Los **servicios de datos** como ContenidosService, GenerosService y PlayersService manejan las operaciones CRUD para sus respectivas entidades. Estos servicios implementan patrones consistentes para el manejo de errores y respuestas.

Manejo de Errores

El sistema de manejo de errores está diseñado para proporcionar información útil al usuario mientras mantiene la estabilidad de la aplicación. Los errores se capturan y procesan de manera consistente en todos los servicios.

La **propagación de errores** desde los servicios hasta los componentes permite que la interfaz de usuario muestre mensajes apropiados. Este flujo asegura que los usuarios reciban retroalimentación clara sobre problemas.

Configuración de APIs

La configuración de APIs se centraliza en archivos de configuración que definen URLs base, endpoints y headers por defecto. Esta centralización facilita el mantenimiento y la configuración de diferentes entornos.

Gestión de Estado Global

Contextos de React

La aplicación utiliza **contextos de React** para gestionar el estado global de manera eficiente. El AuthContext maneja el estado de autenticación, mientras que el DataContext gestiona los datos de la aplicación.

La **separación de responsabilidades** entre contextos permite que cada uno maneje un dominio específico de la aplicación. Esta separación mejora la mantenibilidad y facilita las pruebas.

Hooks Personalizados

Los **hooks personalizados** como useEmailVerification encapsulan lógica compleja y proporcionan una interfaz simple para los componentes. Estos hooks promueven la reutilización de código y mejoran la legibilidad.

La implementación de hooks personalizados sigue las mejores prácticas de React, incluyendo el manejo apropiado de efectos secundarios y la limpieza de recursos. Esta implementación asegura un rendimiento óptimo.

Optimización de Rendimiento

La gestión de estado incluye **optimizaciones de rendimiento** como la memorización de valores calculados y la prevención de re-renderizados innecesarios. Estas optimizaciones son especialmente importantes en aplicaciones móviles.

Navegación y Enrutamiento

Configuración de Expo Router

La aplicación utiliza **Expo Router** para implementar navegación basada en archivos, proporcionando una experiencia de desarrollo intuitiva y una estructura de código organizada. Esta configuración facilita la gestión de rutas y parámetros.

La **navegación protegida** asegura que solo los usuarios autenticados puedan acceder a las funcionalidades principales. Esta protección se implementa a nivel de layout, proporcionando una capa de seguridad consistente.

Estructura de Pantallas

Las pantallas de la aplicación están organizadas en módulos lógicos que corresponden a las funcionalidades principales. Esta organización facilita la navegación y el mantenimiento del código.

La **navegación entre pantallas** se maneja de manera fluida, con transiciones apropiadas y gestión de parámetros. El sistema de navegación proporciona una experiencia de usuario consistente en toda la aplicación.

Configuración y Despliegue

Variables de Entorno

La aplicación utiliza **variables de entorno** para configurar diferentes aspectos del sistema, incluyendo URLs de Supabase y claves de API. Esta configuración permite el despliegue en diferentes entornos sin modificar el código.

La **gestión de secretos** asegura que las claves de API y configuraciones sensibles no se expongan en el código fuente. Esta práctica es esencial para la seguridad de la aplicación.

Configuración de Supabase

La configuración de Supabase incluye la creación de tablas, políticas de seguridad y habilitación de funcionalidades en tiempo real. Esta configuración se documenta en archivos de migración que facilitan el despliegue.

La **configuración de autenticación** incluye la habilitación de proveedores OAuth y la configuración de URLs de redirección. Esta configuración es necesaria para que la autenticación funcione correctamente.

Scripts de Despliegue

La aplicación incluye **scripts de automatización** para facilitar el proceso de despliegue y configuración. Estos scripts reducen la posibilidad de errores humanos y aceleran el proceso de configuración.

Pruebas y Calidad de Código

Configuración de Linting

La aplicación utiliza **ESLint** para mantener la calidad del código y detectar problemas potenciales. La configuración de ESLint incluye reglas específicas para React Native y TypeScript.

La **configuración de Prettier** asegura que el código tenga un formato consistente en todo el proyecto. Esta configuración mejora la legibilidad y facilita la colaboración entre desarrolladores.

Estructura de Archivos

La estructura de archivos del proyecto está organizada de manera lógica, separando claramente los diferentes aspectos de la aplicación. Esta organización facilita la navegación del código y el mantenimiento.

La **separación de responsabilidades** se refleja en la estructura de directorios, con carpetas específicas para componentes, servicios, contextos y utilidades. Esta separación mejora la escalabilidad del proyecto.

Funcionalidades Avanzadas

Gestión de Contenido Audiovisual

El sistema de gestión de contenido incluye una base de datos extensa de series, películas y anime, con metadatos como géneros, descripciones e imágenes. Esta funcionalidad proporciona la base para el juego del ahorcado.

La **categorización de contenido** por géneros y tipos permite una organización eficiente y facilita la expansión futura del catálogo. Esta estructura es escalable y mantenible.

Sistema de Puntuación

El sistema de puntuación implementa lógica compleja para calcular y almacenar las puntuaciones de los jugadores. La puntuación se basa en múltiples factores, incluyendo la precisión de las adivinanzas y la eficiencia del juego.

La **persistencia de puntuaciones** asegura que los logros de los jugadores se mantengan entre sesiones y dispositivos. Esta funcionalidad es esencial para la competitividad del juego.

Optimización de Rendimiento

La aplicación incluye múltiples optimizaciones de rendimiento, incluyendo la carga diferida de componentes, la memorización de valores calculados y la optimización de imágenes. Estas optimizaciones son especialmente importantes en dispositivos móviles.

La **gestión eficiente de memoria** asegura que la aplicación funcione de manera fluida incluso con grandes cantidades de datos. Esta gestión incluye la limpieza apropiada de recursos y la prevención de fugas de memoria.

Seguridad y Privacidad

Protección de Datos

La aplicación implementa múltiples capas de seguridad para proteger los datos de los usuarios. Las políticas de Row Level Security en Supabase aseguran que los usuarios solo puedan acceder a datos apropiados.

La **validación de entrada** previene ataques comunes como inyección de SQL y cross-site scripting. Esta validación se implementa tanto en el cliente como en el servidor.

Gestión de Sesiones

El sistema de gestión de sesiones utiliza tokens seguros proporcionados por Supabase Auth. Estos tokens se renuevan automáticamente y se invalidan apropiadamente cuando el usuario cierra sesión.

La **persistencia segura de sesiones** permite que los usuarios permanezcan autenticados entre sesiones de la aplicación, mejorando la experiencia de usuario sin comprometer la seguridad.

Escalabilidad y Mantenimiento

Arquitectura Escalable

La arquitectura del proyecto está diseñada para ser escalable, permitiendo la adición de nuevas funcionalidades sin afectar el código existente. La separación de responsabilidades y la modularidad facilitan la expansión.

La **configuración de base de datos** incluye índices apropiados para optimizar las consultas frecuentes. Esta configuración mejora el rendimiento a medida que la aplicación crece.

Documentación del Código

El código está completamente documentado utilizando **JSDoc**, proporcionando información detallada sobre funciones, componentes y tipos. Esta documentación facilita el mantenimiento y la colaboración entre desarrolladores.

La **documentación de APIs** incluye ejemplos de uso y descripciones de parámetros. Esta documentación es esencial para el desarrollo futuro y la integración con otros sistemas.

Conclusión

El proyecto **Pixdex** representa una implementación completa y robusta de una aplicación móvil moderna, demostrando las mejores prácticas en desarrollo con React Native, gestión de estado, autenticación y comunicación en tiempo real. La arquitectura modular, la documentación exhaustiva y la atención al detalle hacen de este proyecto un excelente ejemplo de desarrollo de aplicaciones móviles de calidad profesional.

La aplicación combina funcionalidades de entretenimiento con tecnologías avanzadas, creando una experiencia de usuario atractiva y técnicamente sólida. El uso de TypeScript, Supabase y patrones de diseño modernos asegura que el código sea mantenible, escalable y de alta calidad.

El proyecto sirve como una base sólida para futuras expansiones y demuestra la capacidad de crear aplicaciones móviles complejas utilizando tecnologías modernas y mejores prácticas de desarrollo. La documentación completa y la estructura organizada facilitan la comprensión, mantenimiento y evolución del proyecto.