UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO

POLO ZONA OESTE – UERJ ZO

COLEGIADO DE COMPUTAÇÃO E MATEMÁTICA APLICADA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

APLICATIVO DE QUIZ COM PERSISTÊNCIA LOCAL

"QUIZAPP"

MATHEUS DA SILVA SANTOS

DAVI DO NASCIMENTO BOMFIM

RIO DE JANEIRO

2025

Trabalho apresentado à disciplina de Dispositivos Móveis e Embarcados, como requisito parcial para a obtenção de nota da P2.

Professor: Denis Cople

Sumário

1. INTRODUÇÃO

Este documento detalha o desenvolvimento do "QuizApp", um aplicativo móvel interativo desenvolvido utilizando a plataforma Expo e o framework React Native. O objetivo principal do aplicativo é proporcionar uma experiência de quiz dinâmica, onde os usuários podem testar seus conhecimentos sobre “Dispositivos Móveis e Embarcados”.

Uma funcionalidade central do “QuizApp” é a implementação da persistência de dados local. Isso permite que tanto as perguntas do quiz quanto o histórico de pontuações dos jogadores sejam armazenados diretamente no dispositivo do usuário, garantindo o funcionamento completo do aplicativo mesmo sem conexão à internet. O projeto demonstra a integração de componentes de interface de usuários modernos, lógica de jogo interativa e um sistema robusto de armazenamento de dados offline.

2. ARQUITETURA DO APLICATIVO

O “QuizApp” é uma aplicação móvel construída sobre o framework React Native, utilizando a plataforma Expo para simplificar o processo de desenvolvimento e teste. Sua arquitetura é modular, organizada em componentes e módulos que se comunicam para proporcionar uma experiência de usuário fluida e funcional.

2.1 App.js (Ponto de Entrada Principal)

Este é o arquivo raiz da aplicação. Sua principal responsabilidade é inicializar o banco de dados SQLite (initDatabase) e, em seguida, garantir que as perguntas do quiz (populateQuestions) sejam carregadas no banco, caso ainda não existam. Após a inicialização bem-sucedida do banco de dados, o **App.js** renderiza o componente AppNavigator, que gerencia a navegação entre as diferentes telas do aplicativo.

2.2 Módulo navigation/AppNavigator.js (Gerenciamento de Navegação

Utiliza a biblioteca @react-navigation/native e @react-navigation/native-stack para definir a pilha de navegação do aplicativo. Ele estabelece as rotas para as três telas principais (**QuizScreen**, **ResultScreen**, **HistoryScreen**) e determina a tela inicial (**QuizScreen**).

2.3 Módulo screens/ (Telas da Aplicação)

2.3.1 QuizScreen.js

É a interface principal do quiz. É responsável por:

- Carregar perguntas do banco de dados.
- Exibir a pergunta atual e suas opções de resposta.
- Detectar cliques do usuário nas opções.
- Verificar a correção da resposta e atualizar a pontuação.
- Avançar para a próxima pergunta ou navegar para a **ResultScreen** ao final do quiz.

Figura 1: Tela Principal do Quiz com Perguntas.

2.3.2 ResultScreen.js

Exibe a pontuação final do jogador. Permite que o usuário insira um nome e salve sua pontuação no banco de dados. Oferece opções para jogar novamente ou ver o histórico.

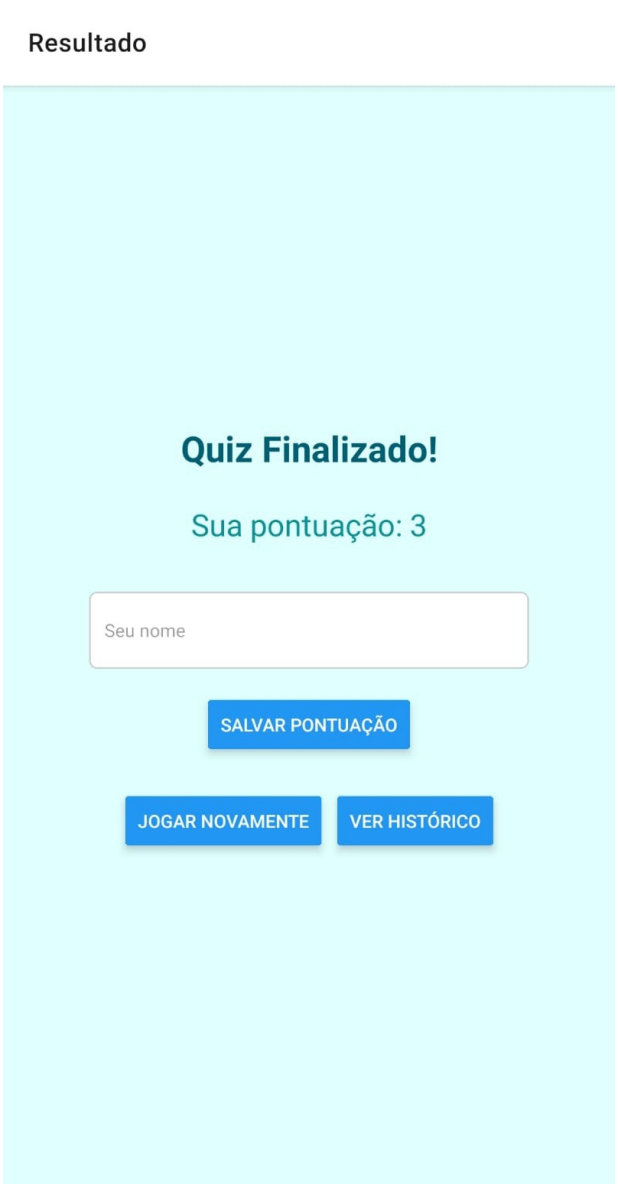

Figura 2: Tela de Resultado do Quiz.

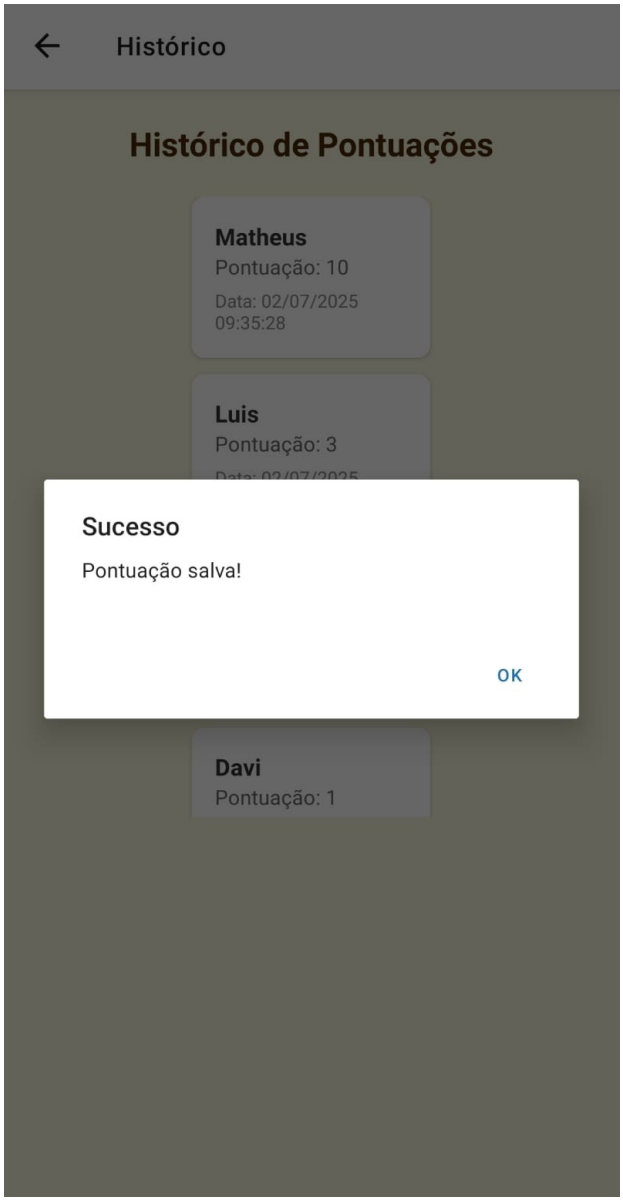

Figura 3: Alerta de Confirmação de Pontuação Salva

2.3.3 HistoryScreen.js

Exibe a lista de todas as pontuações salvas, ordenadas da maior para a menor.

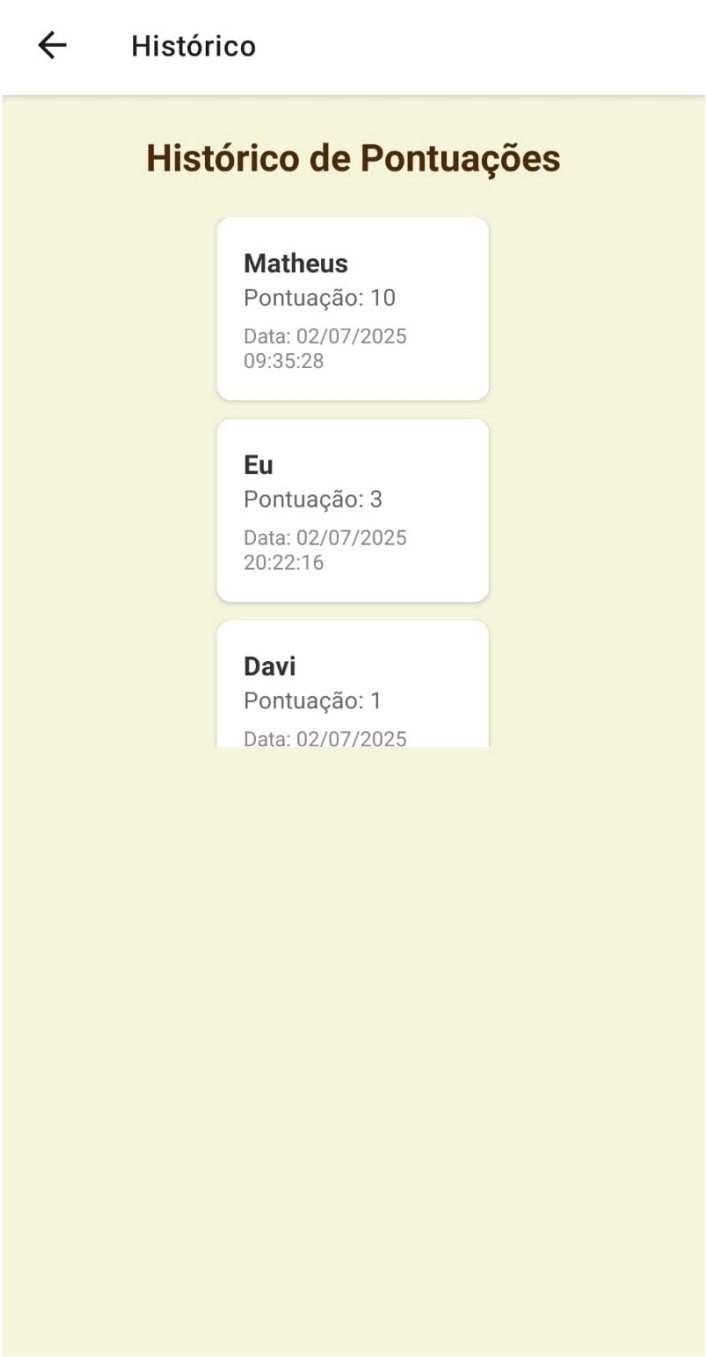

Figura 4: Tela de Histórico de Scores.

2.4 Módulo database/database.js (Persistência de Dados SQLite)

Este módulo encapsula toda a lógica de interação com o banco de dados SQLite. Ele é responsável por:

- Abrir e inicializar o banco de dados (initDatabase).
- Criar as tabelas **questions** e **scores** se elas não existirem.

- Popular as perguntas iniciais na tabela **questions** (populateQuestions).
- Fornecer funções para adicionar (addQuestion, addScore) e consultar (getQuestions, getScores) dados do banco.

2.5 Diagrama de Arquitetura

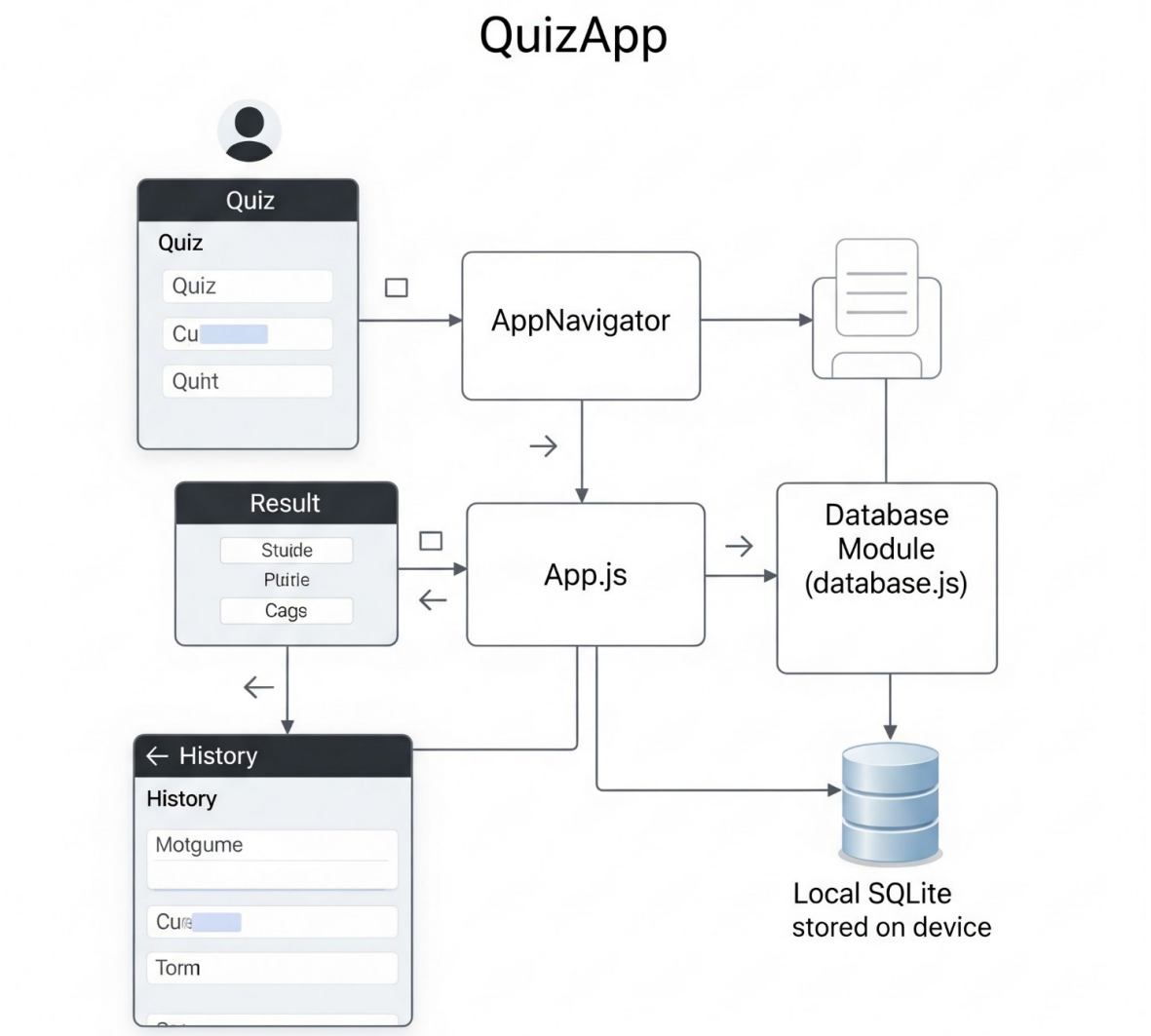

Figura 5: Diagrama de Arquitetura do QuizApp.

3. PERSISTÊNCIA DE DADOS LOCAL (SQLite)

A persistência de dados no "QuizApp" é implementada utilizando o **SQLite**, um sistema de gerenciamento de banco de dados relacional leve e embarcado. O SQLite foi escolhido por ser ideal para aplicações móveis que requerem armazenamento de dados local e

offline, sem a necessidade de um servidor de banco de dados externo ou conexão constante com a internet.

O módulo **database/database.js** é o responsável por toda a interação com o SQLite, encapsulando as operações de criação, leitura, atualização e exclusão (CRUD) de dados.

3.1 Estrutura do Banco de Dados

O banco de dados **quiz.db** do aplicativo é composto por duas tabelas principais:

3.1.1 Tabela questions

Finalidade: Armazena todas as perguntas do quiz, suas opções de resposta e a resposta correta.

Campos:

- **id** (INTEGER PRIMARY KEY AUTOINCREMENT): Identificador único para cada pergunta. É autoincrementado automaticamente a cada nova pergunta.
- **question** (TEXT): O texto completo da pergunta.
- **optionA** (TEXT): O texto da primeira opção de resposta.
- **optionB** (TEXT): O texto da segunda opção de resposta.
- **optionC** (TEXT): O texto da terceira opção de resposta.
- **optionD** (TEXT): O texto da quarta opção de resposta.
- **correctAnswer** (TEXT): O texto da opção de resposta que é considerada correta.

3.1.2 Tabela scores

Finalidade: Registra o histórico de pontuações dos jogadores que completam o quiz.

Campos:

- **id** (INTEGER PRIMARY KEY AUTOINCREMENT): Identificador único para cada registro de pontuação.
- **playerName** (TEXT): O nome do jogador que registrou a pontuação.
- **score** (INTEGER): A pontuação final alcançada pelo jogador.
- **date** (TEXT): A data e hora em que a pontuação foi registrada.

3.2 Operações de Persistência Realizadas

As principais operações que o aplicativo executa no banco de dados incluem:

- initDatabase(): Inicializa a conexão com o banco de dados e cria as tabelas questions e scores se elas ainda não existirem.

- populateQuestions(): Insere um conjunto inicial de perguntas na tabela questions na primeira execução do aplicativo, garantindo que o quiz tenha conteúdo. Verifica se as perguntas já existem para evitar duplicatas.
- getQuestions(): Recupera todas as perguntas da tabela questions para serem exibidas durante o quiz.
- addScore(playerName, score): Registra a pontuação final de um jogador na tabela scores, juntamente com seu nome e a data/hora.
- getScores(): Recupera o histórico de todas as pontuações salvas da tabela scores, geralmente ordenadas da maior para a menor.

4. NATUREZA CLIENTE-SIDE DO APLICATIVO

O "QuizApp" é concebido e opera como uma **aplicação puramente cliente-side**. Isso significa que toda a sua lógica de negócios, interface de usuário e persistência de dados residem integralmente no dispositivo do usuário (o "cliente"). Não há um componente de servidor remoto (backend) ou uma API REST sendo utilizada para o funcionamento das funcionalidades principais do aplicativo.

O banco de dados **SQLite** é embarcado e gerenciado localmente no dispositivo móvel. Portanto, o "QuizApp" oferece uma **experiência de usuário completamente autônoma**, funcionando plenamente mesmo na ausência de conexão com a internet. O dispositivo cliente, seja um smartphone ou tablet, é o único ambiente de execução necessário para todas as operações do quiz e o gerenciamento de pontuações.

5. TECNOLOGIAS E FERRAMENTAS UTILIZADAS

O desenvolvimento do "QuizApp" foi realizado utilizando um conjunto de tecnologias e ferramentas modernas, amplamente empregadas no ecossistema de desenvolvimento de aplicações móveis e JavaScript:

- JavaScript: Linguagem de programação principal utilizada para toda a lógica da aplicação, tanto na interface quanto na manipulação de dados.
- React Native: Framework de código aberto, baseado em JavaScript, para a construção de interfaces de usuário nativas para dispositivos iOS e Android a partir de uma única base de código.
- Expo CLI: Conjunto de ferramentas e serviços que simplifica o desenvolvimento, teste e implantação de aplicativos React Native, abstraindo a complexidade do ambiente nativo.

- SQLite: Sistema de gerenciamento de banco de dados relacional leve e embarcado, utilizado para a persistência local de perguntas e scores.
- expo-sqlite: Biblioteca que permite a integração e manipulação de bancos de dados SQLite em aplicativos Expo/React Native.
- React Navigation: Biblioteca de roteamento e navegação modular e personalizável para aplicativos React Native, utilizada para gerenciar as transições entre as telas (Quiz, Resultado, Histórico).
- Node.js: Ambiente de execução JavaScript runtime, essencial para o funcionamento do npm e de ferramentas como o Expo CLI.
- npm (Node Package Manager): Gerenciador de pacotes padrão para o Node.js, utilizado para instalar e gerenciar as dependências do projeto.
- Visual Studio Code (VS Code): Editor de código-fonte leve e poderoso, utilizado para escrever e depurar o código da aplicação.
- Git: Sistema de controle de versão distribuído, utilizado para gerenciar as alterações do código-fonte e para integrar o projeto ao GitHub.
- GitHub: Plataforma de hospedagem de código-fonte baseada em Git, utilizada para armazenar e versionar o projeto na nuvem.
- Expo Go: Aplicativo móvel para dispositivos Android e iOS que permite testar rapidamente aplicativos Expo React Native em desenvolvimento, escaneando um QR code.

6. COMO EXECUTAR O APLICATIVO

Para executar o "QuizApp" em um ambiente de desenvolvimento ou em um dispositivo móvel, siga os passos abaixo. Certifique-se de que seu sistema operacional é Windows, pois as instruções de linha de comando são baseadas nele.

6.1 Pré-requisitos

a. Node.js: Baixe e instale a versão LTS (Long Term Support) do Node.js no site oficial: https://nodejs.org/. O npm (Node Package Manager) será instalado junto.

b. Git: Baixe e instale o Git para Windows no site oficial: https://git-scm.com/download/win. Durante a instalação, selecione a opção "Git from the command line and also from 3rd-party software" para que o Git seja acessível pelo Prompt de Comando.

c. Expo CLI: Abra o Prompt de Comando (CMD) como Administrador e instale o Expo CLI globalmente: npm install -g expo-cli.

d. Visual Studio Code (VS Code): Embora não seja estritamente necessário para executar, é o editor recomendado para visualizar e editar o código. Baixe e instale em: https://code.visualstudio.com/.

e. Expo Go (no Celular): No seu dispositivo móvel (Android é o recomendado para este projeto), baixe e instale o aplicativo "Expo Go" na Google Play Store.

6.2 Obter o Projeto

Você pode obter o código-fonte do projeto de duas maneiras:

a. Via GitHub (Recomendado): * Abra o Prompt de Comando (CMD) como Administrador. * Navegue até o diretório onde deseja armazenar o projeto (ex: cd C:\Users\SeuUsuario\Documents\Projetos). * Clone o repositório do GitHub (substitua mathsants pelo seu nome de usuário se o repositório for outro): git clone https://github.com/mathsants/QuizApp.git. * Entre na pasta do projeto clonado: cd QuizApp.

b. Via Arquivo Compactado (.zip): * Faça o download do arquivo .zip do projeto e descompacte-o em uma pasta de sua escolha (ex: C:\Users\SeuUsuario\Documents\QuizApp). * Abra o Prompt de Comando (CMD) como Administrador. * Navegue até a pasta raiz do projeto descompactado: cd C:\Users\SeuUsuario\Documents\QuizApp.

6.3 Instalação das Dependências

Com o Prompt de Comando (CMD) aberto na pasta raiz do projeto (QuizApp), execute o seguinte comando. Este comando instalará todas as bibliotecas e dependências necessárias para o projeto funcionar, forçando a compatibilidade de versões:

npm install react@18.2.0 react-dom@18.2.0 react-native@0.73.6 react-native-web@~0.19.6 expo@~50.0.0 expo-status-bar@~1.11.1 expo-sqlite@~13.1.0 @react-navigation/native@^6.0.0 @react-navigation/native-stack@^6.0.0 react-native-screens@~3.29.0 react-native-safe-area-context@4.8.2 --force

Pode ser necessário digitar 'y' e Enter se o npm perguntar sobre conflitos de dependência. Após a instalação das dependências, execute o comando de correção do Expo: npx expo install –fix.

6.4 Limpeza do Ambiente e Reinstalação do Expo Go (Crucial para o Primeiro Uso)

Para garantir que o aplicativo carregue corretamente e o banco de dados seja inicializado:

a. No seu dispositivo móvel: Desinstale completamente o aplicativo "Expo Go" e, em seguida, reinstale-o a partir da loja de aplicativos (Google Play Store para Android). b. No Prompt de Comando (CMD): Limpe o cache do Expo: npm cache clean --force e em seguida: del /s /q %TEMP%\metro-* e del /s /q %TEMP%\expo-*.

6.5 Executar o Aplicativo

Com o Prompt de Comando (CMD) ainda na pasta raiz do projeto (QuizApp), inicie o servidor de desenvolvimento do Expo:

npx expo start –clear

O parâmetro --clear garante que o bundler do Expo comece do zero e aplique todas as mudanças.

6.6 Abrir o Aplicativo no Celular

O comando npx expo start --clear exibirá um QR Code no terminal do CMD. No seu dispositivo móvel, abra o aplicativo "Expo Go" recém-instalado. No Expo Go, selecione a opção para escanear o QR Code e aponte a câmera para o QR Code exibido no CMD. O aplicativo "QuizApp" deverá carregar no seu dispositivo móvel, e você poderá interagir com o quiz, salvar pontuações e visualizar o histórico.

7. MELHORIAS E EXPANSÕES FUTURAS

O "QuizApp" foi desenvolvido com foco em sua funcionalidade principal e persistência de dados local. No entanto, o projeto possui diversas oportunidades para futuras melhorias e expansões, que poderiam enriquecer ainda mais a experiência do usuário e a robustez da aplicação:

- Categorias de Perguntas: Implementar um sistema de categorias (ex: "Hardware", "Software", "Redes") permitindo que os usuários escolham o tema do quiz.
- Modos de Jogo: Adicionar diferentes modos de jogo, como:
- Contra o tempo: Onde o usuário tem um limite de tempo para responder a cada pergunta.
- Modo de estudo: Onde o aplicativo fornece feedback imediato (correto/incorreto) após cada resposta.
- Feedback Visual Aprimorado: Fornecer um feedback visual mais claro e divertido após cada resposta (animações, cores diferentes para certo/errado).
- Personalização do Perfil: Permitir que os usuários criem perfis e acompanhem seu progresso ao longo do tempo.

- Notificações: Adicionar notificações para lembrar o usuário de jogar ou para informar sobre novos quizzes (se houver conexão externa).
- Integração com API Externa: Para um futuro mais complexo, integrar o aplicativo a uma API remota para buscar perguntas online, permitindo um banco de perguntas dinâmico e atualizável sem a necessidade de reinstalar o app.
- Sons e Música: Adicionar efeitos sonoros para respostas corretas/incorretas e música de fundo para aprimorar a imersão.
- Compartilhamento de Pontuação: Permitir que os usuários compartilhem seus scores em redes sociais.

Essas adições futuras poderiam transformar o "QuizApp" em uma plataforma de aprendizado e entretenimento ainda mais completa.

8. CONCLUSÃO

O desenvolvimento do "QuizApp" foi uma experiência de aprendizado enriquecedora, permitindo a aplicação prática de conceitos fundamentais de desenvolvimento móvel com React Native. O projeto demonstrou a capacidade de construir uma aplicação interativa com gerenciamento de estado complexo, navegação entre telas e, crucialmente, a implementação de persistência de dados local utilizando SQLite.

Os desafios enfrentados durante o processo, como a resolução de conflitos de dependências, a adaptação a diferentes versões de SDK do Expo e a depuração de erros de ambiente, foram oportunidades valiosas para aprimorar as habilidades de resolução de problemas e a compreensão do ecossistema de desenvolvimento. O resultado é um aplicativo funcional e robusto, que atende aos requisitos propostos e valida o conhecimento adquirido em “Dispositivos Móveis e Embarcados”.

9. REFERÊNCIAS

As informações e a base para o desenvolvimento do "QuizApp" foram consultadas nas documentações oficiais das seguintes tecnologias:

Node.js: https://nodejs.org/docs/

React Native: https://reactnative.dev/docs/

Expo: https://docs.expo.dev/

React Navigation: https://reactnavigation.org/docs/

SQLite (via Expo SQLite): https://docs.expo.dev/versions/latest/sdk/sqlite/

Git: https://git-scm.com/doc

GitHub: https://docs.github.com/

Visual Studio Code: https://code.visualstudio.com/docs