Estrutura de Dados do Aplicativo Quizzer ODS

Banco de Dados: SQLite

O **SQLite** é um banco de dados relacional leve que será utilizado para armazenar e gerenciar as informações dos usuários. O banco de dados é responsável por manter os seguintes detalhes:

- **Tabela: Users**
  - **id**: Chave primária, identificador único do usuário (tipo INTEGER, autoincrement).
  - **username**: Nome do usuário (tipo TEXT, não nulo).
  - **email**: Endereço de e-mail do usuário (tipo TEXT, não nulo, único).
  - **passwordHash**: Hash seguro da senha do usuário (tipo TEXT, não nulo).
  - **passwordSalt**: Salt usado para fortalecer a segurança da senha (tipo TEXT, não nulo).
  - **createdAt**: Data e hora em que o usuário foi criado (tipo TIMESTAMP, não nulo).
  - **updatedAt**: Data e hora da última atualização do usuário (tipo TIMESTAMP).

Back-End: Node.js + Express

O back-end será construído com **Node.js** e o framework **Express**, que juntos gerenciam as rotas da API para funcionalidades como cadastro e login.

- **Rotas da API**:
  - **POST /register**: Rota para cadastro de um novo usuário.
    - Valida os dados do usuário (username, email, senha).
    - Gera o hash e salt da senha antes de armazenar no banco de dados.
  - **POST /login**: Rota para autenticação do usuário.
    - Verifica as credenciais do usuário (username ou email, senha).
    - Valida a senha usando hash e salt armazenados.
  - **DELETE /deleteUser**: Rota para exclusão de conta.
    - Exclui todos os dados relacionados ao usuário, mediante confirmação.
- **Hashing de Senha**:
  - O hashing das senhas é realizado utilizando uma biblioteca como `bcrypt`, que implementa técnicas de hash e salt para maior segurança.

Cliente (Android)

O aplicativo Android utiliza a biblioteca **Retrofit** para comunicação HTTP com o servidor. O cliente lida com a coleta de dados e o envio de requisições para o back-end.

- **Biblioteca Retrofit**: Utilizada para realizar requisições HTTP de forma eficiente, como:
    - **Cadastro**: Envia os dados do usuário (username, email, senha) para o servidor.
    - **Login**: Envia as credenciais para autenticação e recebe uma resposta do servidor.

Persistência Local

O aplicativo pode armazenar localmente informações como `username` utilizando uma solução de persistência simples, como **SharedPreferences**, permitindo o uso eficiente de dados sem a necessidade de fazer requisições repetidas ao servidor.

Fluxo de Dados

1. **Coleta de Dados no Cliente**:
    - O usuário entra com seus dados no aplicativo (username, email, senha).
    - O aplicativo envia esses dados ao servidor por meio de requisições HTTP usando o Retrofit.
2. **Validação e Interação com o Banco de Dados no Servidor**:
    - O servidor recebe a requisição e valida os dados.
    - Caso seja uma operação de cadastro, a senha é processada (hash + salt) e os detalhes do usuário são armazenados no banco de dados SQLite.
    - Caso seja uma operação de login, a senha fornecida é comparada com o hash armazenado no banco de dados.
3. **Segurança das Senhas**:
    - As senhas não são armazenadas em texto simples; em vez disso, são transformadas usando uma função de hash segura, com um salt único adicionado antes do armazenamento.
4. **Resposta para o Cliente**:
    - O servidor retorna uma resposta ao cliente (sucesso ou falha na operação).
    - O cliente apresenta essa resposta ao usuário.

Forms no Cliente

O aplicativo Android possui formulários que permitem ao usuário realizar operações específicas:

- **Formulário de Exclusão de Dados**:
    - Quando o usuário decide excluir sua conta, ele acessa um formulário específico.
    - O formulário coleta a confirmação do usuário e envia uma solicitação para a API `DELETE /deleteUser` no servidor.
- **Formulário de Alteração de Dados**:
    - Se o usuário deseja alterar ou excluir sua conta, ele é redirecionado para uma seção que permite executar essas ações.