Documentação do Aplicativo GRAPE

1. Objetivo do Projeto

O aplicativo foi desenvolvido para otimizar a experiência de compra de produtos (uvas) diretamente de uma fazenda urbana. Ele permite que os usuários:

- Cadastrem seus dados.
- Edite seus dados caso seja necessário na tela “Meus Dados”.
- Visualizem produtos disponíveis para compra.
- Selecionem itens e adicionem ao carrinho.
- Finalizem pedidos com opções de pagamento.
- Acompanhem o status de suas compras na tela “Meus Pedidos”

O projeto busca oferecer:

- Interface amigável e funcional, projetada para dispositivos Android.
- Integração com backend para armazenamento e gestão de pedidos.
- Sincronização em tempo real entre as aplicações Web, Desktop e Mobile.

2. Arquitetura e Stack

2.1. Arquitetura

O aplicativo segue o padrão MVVM (Model-View-ViewModel), adaptado para o Android, com as camadas organizadas da seguinte forma:

- **Frontend (Interface de Usuário):**
  - Desenvolvido em Java com layouts em XML.
  - Componentes:
    - RecyclerView e Adapters para exibição dinâmica de listas.
    - Dialogs para interações personalizadas (ex: confirmação de PIX).
- **Backend:**
  - Comunicação via API REST, configurada com o Retrofit.
  - Segurança gerenciada por meio do arquivo network_security_config.xml.
- **Persistência Temporária:**
  - Dados do carrinho e pedidos armazenados localmente em objetos manipulados com ArrayLists.

- **Banco de Dados:**
  - SQL Server é utilizado para armazenar informações de clientes, funcionários, produtos e pedidos, com integração entre as três aplicações propostas no PIM 4:
    - Web.
    - Desktop.
    - Mobile.

2.2. Stack Tecnológica

- **Linguagem de Programação:** Java.
- **Ferramentas e Bibliotecas:**
  - Retrofit: Comunicação HTTP com a API.
  - Glide: Carregamento e exibição de imagens.
  - Android SDK: Componentes nativos para desenvolvimento Android.
- **Backend:** Servidor Node.js com integração ao banco de dados SQL Server.
- **Banco de Dados:** SQL Server.

3. Funcionalidades

1. **Cadastro e Login:**
   - Cadastro de novos usuários com validações:
     - Campos obrigatórios: Nome, CPF, E-mail, Telefone, Senha.
     - Verificação de correspondência entre Senha e Confirmação de Senha.
   - Login por e-mail e senha.
2. **Visualização de Produtos:**
   - Lista de produtos com nome, descrição, imagem e preço.
   - Suporte para adicionar produtos ao carrinho.
3. **Checkout Antes de Finalizar o Pedido:**
   - Visualização dos produtos selecionados.
   - Cálculo dinâmico do valor total.
   - Preenchimento do endereço de entrega.

4. **Finalização de Pedidos:**
    - Escolha do método de pagamento: PIX ou Cartão.
    - Confirmação do pedido e envio para o backend.
5. **Consulta de Pedidos:**
    - Histórico de pedidos com status atualizado em tempo real:
        - "Preparando Pedido", "Em Trânsito", “Concluído” ou “Cancelado”.
6. **Atualização de Dados Pessoais:**
    - Tela “Meus Dados” para edição de informações do usuário.

4. Desafios e Soluções

4.1. Transição de Dados Entre Telas

- **Desafio:**
    - Levar os produtos selecionados e o valor total da tela de produtos para a tela de checkout.
    - Transferir os dados do pedido concluído para a tela de pedidos, permitindo acompanhar o status em tempo real.
- **Solução:**
    - Uso de Intents no Android para transportar os dados entre as telas:
        - Na transição entre telas, os dados (como produtos selecionados e valores) são serializados como objetos Java e enviados por meio de extras no Intent.
    - Implementação de um Objeto Compartilhado (PedidoStorage):
        - Centraliza os dados do carrinho e mantém informações persistentes enquanto o pedido está em andamento.
    - Integração com a API:
        - O pedido concluído é enviado ao backend, que retorna um ID e um status inicial. Esses dados são exibidos na tela de pedidos.

4.2. Integração com Banco de Dados

- **Desafio:**
    - Sincronizar os pedidos entre as aplicações Desktop e Mobile.

- Garantir que os status atualizados na aplicação Desktop reflitam em tempo real na Mobile.

- **Solução:**
  - **Estrutura do Banco de Dados:**
    - Tabelas centralizadas no SQL Server, compartilhadas entre as aplicações.
    - Relações configuradas para gerenciar clientes, produtos, pedidos e itens dos pedidos.
  - **Integração:**
    - Aplicação Mobile realiza chamadas à API para buscar dados atualizados do banco.
    - Aplicação Desktop altera os status diretamente no banco, e a aplicação Mobile consulta periodicamente para obter os novos status.
  - **Endpoints na API:**
    - **GET /pedidos:** Retorna os pedidos atualizados com seus status.
    - **POST /pedidos:** Recebe novos pedidos enviados pela aplicação Mobile.

5. Testes e Validações

5.1. Testes

- **Funcionalidades testadas:**
  - Cadastro e login de usuários.
  - Exibição e manipulação de produtos no carrinho.
  - Finalização e envio de pedidos.
  - Consulta de status dos pedidos.
- **Tipos de Testes:**
  - Testes manuais realizados no emulador do Android Studio.
  - Simulação de falhas, como:
    - Dados inválidos no formulário.
    - Falha de conexão com o backend.

5.2. Validações

- **Frontend:**
  - Validação de campos no cadastro e login.
  - Restrições para impedir que pedidos sejam finalizados sem dados válidos.

- **Integração:**
  - Testes de integração para garantir:
    - Comunicação correta entre o aplicativo e a API.
    - Sincronização de status com a aplicação Desktop.
- **Simulações:**
  - Simulação de múltiplos pedidos para validar o desempenho e sincronização em tempo real.

6. Atualizações e Monitoramento

6.1. Atualizações

- Planejamento para futuras funcionalidades:
  - Notificações push para informar mudanças no status dos pedidos.
  - Suporte a filtros de produtos por categorias.
  - Melhoria na interface gráfica.

6.2. Monitoramento

- Monitoramento em tempo real da API para detectar falhas.
- Logs de erros registrados durante a execução do aplicativo.

7. Manutenção e Utilização

7.1. Manutenção

- **Frontend:**
  - Revisão contínua do código Java para otimização e remoção de redundâncias.
  - Atualizações de layout para melhorar a experiência do usuário.
- **Backend:**
  - Ajustes em endpoints da API conforme novas funcionalidades sejam adicionadas.
  - Monitoramento de desempenho do banco de dados.

7.2. Utilização

- Utilizado para gerenciar pedidos e simplificar a experiência de compra.
- Sincronização com as aplicações Web e Desktop para uma experiência integrada.

Link do projeto no **GitHub**: https://github.com/Leonardodiaspereira/GrapeAplicativoMobile

APK gerado através do Android Studio: