DESENVOLVIMENTO DE CASE TÉCNICO MOBILE

ALURA

GERENCIADOR DE ENDEREÇOS

FERRAMENTAS UTILIZADAS:

- ANDROID STUDIO

- LINGUAGEM (KOTLIN)

- BANCO DE DADOS (ROOM)

- GIT E GITHUB

2022

1. Conteúdo

2. Capítulo – Introdução

Neste documento estarei descrevendo como foi minha experiência em planejar e desenvolver um aplicativo que armazena endereços para o case técnico solicitado pela Alura, contarei sobre motivos que me fizeram escolher determinado caminho, dificuldades enfrentadas, possiveis melhorias que poderiam ser realizadas em caso de mais tempo e experiência e uma breve conclusão sobre a conclusão do aplicativo feito em Android nativo.

3. Capítulo – Planejamento

Este capítulo será utilizado para contar sobre como foi realizada a tomada de decisões para utilizar determinada arquitetura e que padrões utilizei no desenvolvimento.

3.1 – Arquitetura Utilizada

A arquitetura de software escolhida para reger o aplicativo foi a MVVM (Model View ViewModel) que por sua vez é a recomendação do Google (Grande parte das práticas utilizadas foram escolhidas por conta do próprio Google recomendar e ser um padrão usual entre os desenvolvedores).

Essa arquitetura em suma é utilizada para separar a Interface que é mostrada pelo usuário (View) que se conecta com a camada que coordena o funcionamento entre os

dados da sua aplicação e a interface (ViewModel) que por sua vez fica atrelada a lógica de negócio da sua aplicação (Model).

3.2 – Banco de dados

O banco de dados utilizado foi o Room que por sua vez também é recomendado pelo Google e que substitui o SQLite anteriormente utilizado como padrão pelos aplicativos desenvolvidos de maneira nativa em Android.

Escolhido não somente por conta da recomendação mas também pela facilidade de implementa-lo e realizar manutenção ou adicionar novas funcionalidades.

3.3 – Boas Práticas

As práticas utilizadas a escrita do código foram superficialmente tiradas da ideia de código limpo, por ser uma prática que não domíno 100% haverá algumas falhas de utilização, porém, acredito que o código produzido possuí sua devida facilidade de manutenção e com um pouco de consulta no guia do programador no site de desenvolvedor androids sustentado pelo Google é possível encontrar as referências para a produção do mesmo.

4. Cápitulo – Desenvolvimento

Aqui contarei como foi ordenado os processos de desenvolvimento do aplicativo.

4.1 – Model

Primeiramente utilizei do PDF enviado para escolher os dados que seriam utilizados dentro da nossa aplicação que se resumia em:

- Nome
- Telefone

- CEP
- Logradouro
- Número
- Bairro
- Complemento
- Cidade
- Estado

Determinei que todos seriam utilizados em String pois não havia operações a serem realizadas entre os dados.

Então com essa informação em mãos criei o objeto "*Endereco.kt*" que ficaria encarregado de ser nossa Linha no banco de dados, também estabelecendo as colunas que seriam utilizadas. Optei por não utilizar um *'id'* como identificador por conta da complexidade do aplicativo ser bem simples, então deixei o nome como identificar primário e passível de ser duplicado sem interferências. Os nomes das colunas por mais que o próprio Room soubesse que poderiam condizer com os nomes atribuidos nos devidos atributos optei por específicar para possiveis alterações. Todos os atributos do objeto *Endereco* foram designados com um operador *"?"* para que sejam tratados caso suas entradas fossem nulas podendo acarretar em um erro.

4.2 – Data Acess Object (DAO)

Como o nome sugere o *DAO* é uma interface utilizada para acessar os dados do objeto em questão, então é nele que possuímos as chamadas ao banco de dados sejam elas para organizar, inserir ou deletar as informações contidas.

4.3 – Application

Essa classe é utilizada para fazermos acesso aos repositórios e ao banco de dados somente pelo escopo fora da *Thread* principal pois qualquer tipo de modificação ou acesso ao banco de dados deve ser feito fora dela para que não haja travamento ou lentidão na execução do aplicativo, ela também será usada para que instanciemos apenas uma vez o banco de dados após o aplicativo ser instalado.

4.4 – Repository

Repository é um padrão de projeto utilizado resumidamente para ocultar detalhes de como os dados são armazenados ou acessados, usando também para separar a lógica que recupera os dados da fonte de dados.

4.5 – Views

Essa camada possuí nossas interfaces e interações com o usuário e está localizada no pacote *"ui"* no código da nossa aplicação, dentro do nosso aplicativo existe a tela principal onde é exibido uma lista dos nossos endereços salvos, Por padrão o aplicativo insere automaticamente uma endereço exemplo para mostrar como é o design. Possuímos também a tela de novo endereço que por sua vez tem toda a lógica utilizada para salvar os dados corretamente com suas devidas restrições de entrada.

4.6 – ViewModel

Como citado anteriormente a ViewModel é utilizada para separar a camada de interface do usuário da camada de dados e nela é possível encontrar as funções que foram previamente criadas no *Repository* facilitando assim a manutenção e colocando os métodos acessores ou modificadores do banco de dados nos seus devidos escopos.

4.7 – RecyclerView

Para estar exibindo a lista de endereços salvos optei por utilizar o *RecyclerView* que é uma forma de mostrar uma lista de maneira performática por conta de carregar somente o que será mostrado pelo usuário, em meu GitHub existe um breve repositório explicando como fazer a implementação do *RecyclerView* passo a passo (link citado : https://github.com/MatheusMelo8/RecyclerView-Android-Kotlin).

4.8 – Melhorando Layout

Por ultimo mas não menos importante, resolvi polir o layout deixando ele com cores mais agradáveis, utilizei algumas cores do próprio site da Alura que na minha opinião contrastavam bem com a proposta do aplicativo e do case fornecido.

Também inseri um botão de deletar todos os registros para que após um cansativo dia de trabalho do Mobilino ele possa apagar todos seus endereços para um possível próximo dia.

5. Cápitulo - Dificuldades

Ao decorrer do desenvolvimento do aplicativo, notei algumas dificuldades em implementar a arquitetura em MVVM então tirei um dia específico para estar estudando ela pois em alguns aspectos eu não sabia para que estar utilizando então foi um tempo bem corrido para estar solidificando a base.

Em relação as *Coroutines* eu tenho pouca familiaridade com sua utilização, então foi bem difícil o começo do código quando eu estava pensando a maneira correta de usar elas.

6. Cápitulo - Melhorias

Em relação ao que poderia ser melhor, eu acredito que em questão de animações de interface e talvez uma tela introdutória mais amigável ao aplicativo, por conta do prazo e das barreiras que encontrei no começo do desenvolvimento acabei optando por simplificar a interface em troca de entregar funcionalidade desejada dentro do esperado.

O botão de deletar todos os registros foi previamente planejado porém meu objetivo principal era deletar todos ou somente um registro, porém por conta do tempo que o planejamento da arquitetura me tomou acabei optando por deixar somente o botão de deletar todos.

Em utlimo caso acredito que melhorar a distribuição da arquitetura e a legibilidade do código seria uma ótima forma de estar entregando algo a mais para os próximos desenvolvedores que fossem realizar a manutenção do código ou até mesmo adicionando alguma funcionalidade.

7. Cápitulo – Conclusão

Todo esse processo foi bem interessante, o case técnico me agradou bastante, me fazendo sair da zona de conforte, de colocar em prática uma arquitetura que eu tenho dificuldade de estar produzindo mas isso me fez estudar e correr atrás da entrega do aplicativo de forma que ele esteja utilizável e com uma interface agradável, acredito que consigo melhorar com estudos frequentes e bastante prática.