RAW App

Pedro H. Pino, Samuel S. Rocha, Filipe C. de Jesus

Instituto de Ensino Superior de Brasília

Ciência da Computação

pedro.haluch@gmail.com, samuelsilv.rocha@gmail.com, filipe.cdj@gmail.com

Resumo. *Neste trabalho os conhecimentos adquiridos ao longo do semestre foram usados para conceber o* MVP *de um aplicativo Android com temática de review de produtos. Dentre as ferramentas empregadas estão construção de APIs, CRUD em banco de dados, framework MVVMi, e requisições HTTPs, além da elaboração de um roadmap geral de próximos passos.*

1. Introdução e Contextualização

A temática inicial do projeto foi definida como uma plataforma para avaliação de artigos sexuais, trazendo um ambiente mais livre para a discussão sobre esse tipo de produto que é considerado tabu. Durante o desenvolvimento, no entanto, chegou-se à conclusão que o tema era muito expansivo, e que focar na lógica de negócio e valor comercial afetaria o foco e a capacidade de utilizar o ferramental adquirido ao longo do curso. Decidimos então usar a lógica de *MVP*s (*minimum viable products*): fazer a implementação de cada feature e suas possíveis integrações, mesmo que de forma simples.

2. Referencial Teórico e Trabalhos Correlatos

Para oferecer um serviço como esse, alguns requisitos são necessários. Do ponto de vista do usuário temos, portanto:

- Necessidade de autenticação,
- cadastro e leitura de informações,
- acesso a localizações, e
- suporte via chatbot.

Já do ponto de vista técnico, precisamos:

- manter ambientes autenticado e não-autenticado distintos,
- delegar a parte de autenticação a um serviço dedicado,
- capacidade de fazer requisições web,
- lidas com essas últimas em camadas que possam ser responsáveis por partes distintas das nossas validações,
- conectar a um banco de dados e fazer as operações básicas,
- acessar serviços de localização geográfica.

A expectativa é, portanto, demonstrar como essas ferramentas trabalham em conjunto para oferecer seus respectivos serviços e facilitar o desenvolvimento desse tipo de produto.

3. Referencial Teórico e Trabalhos Correlatos

MVVMi, Fragmentos, Injeção de Dependências e RecyclerViews

A escolha de como estruturar o aplicativo de forma a facilitar escala e integração com outras ferramentas passa por decidir como os componentes e suas funções serão distribuídos e abstraídos. Um pilar claro desde o início é a extrema dependência de programação orientada a objetos para conseguir executar esse tipo de desenvolvimento. Sem as conveniências de abstração oferecidas por objetos dentro do código seria muito difícil implementar a maioria das ferramentas utilizadas.

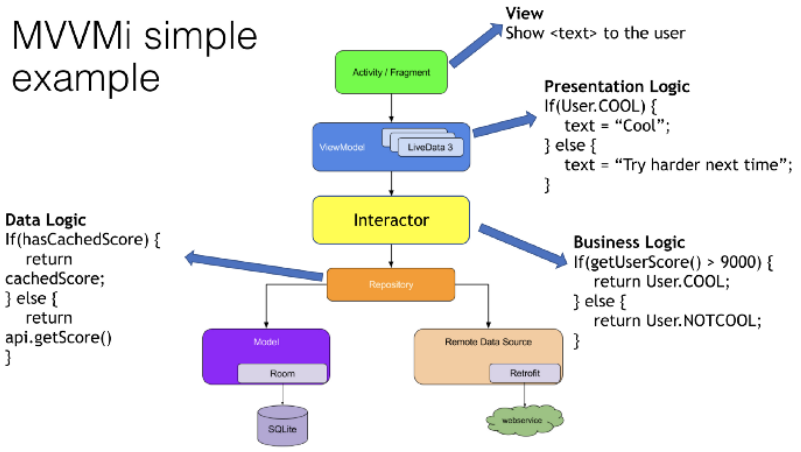

Fig 1. Arquitetura MVVMi

https://medium.com/@thereallukesimpson/clean-architecture-with-mvvmi-architecture-components-rxjava-8c5093337b43

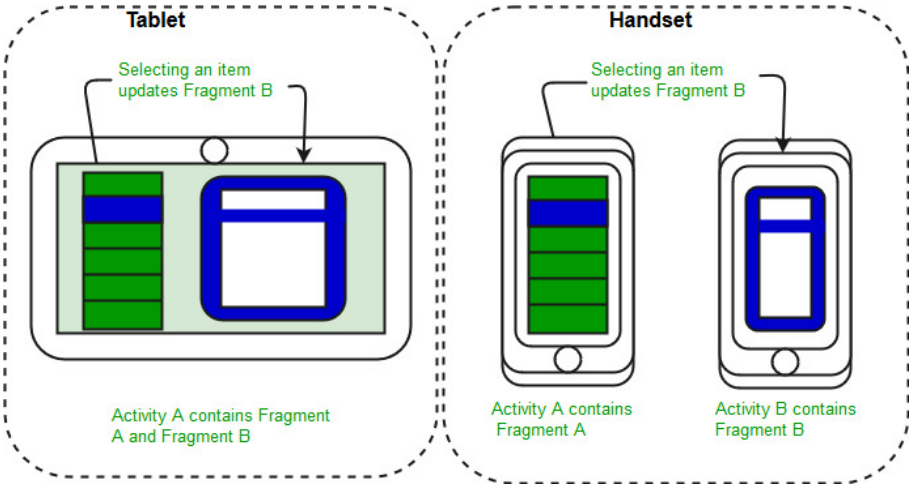

Fig 2. Benefícios do Uso de Fragmentos
https://www.geeksforgeeks.org/introduction-fragments-android/

O grande diferencial estrutural das escolhas feitas é a modularização e abstração das estruturas em código. O uso da dependência hilt-dagger e suas injeções de dependência permite criar cadeias de classes que não precisam de construtores sendo instanciados a todo momento. O benefício é imediatamente notado quando se usa a arquitetura MVVMi (Figura 1). Aqui o objetivo é criar camadas de responsabilidades em relação aos dados e objetos sendo transitados. Tanto o desenvolvimento quanto a manutenção ficam mais diretos definindo classes específicas para tratar por exemplo de validação de negócio e validação técnica. O código fica mais modularizado, e os métodos sofrem menos efeito tanto "do que vem antes" quanto "do que vem depois".

Ainda em busca de maior modularização e abstração fica motivado o uso de fragmentos (*Fragments*) em vez de Activities para criar tanto as classes internas quanto as próprias telas. Os fragmentos são versáteis: podem desde ocupar completamente a tela no lugar

de uma Activity como ser apenas um elemento dela. Além disso, são mais plásticos em sua habilidade de manter estados em alterações como, por exemplo, rotação de tela. A comunicação e navegação entre fragmentos também é facilitada com NavGraphs que, também, abstraem as “tecnicalidades” de integrar esses objetos e seus atributos.

Ainda sobre a estruturação uma escolha importante a ser feita é a do uso de RecyclerViews. Esses elementos são responsáveis por dispor na tela estruturas list-like, em diversos formatos distintos. O benefício do uso desse recurso específico está na habilidade de, trabalhando em conjunto com um Adapter (classe responsável por gerenciar os dados dos elementos da lista, entre outras coisas), gerenciar muito melhor a carga desse tipo de estrutura no aplicativo, instanciando e removendo itens de memória à medida que entram e saem da tela.