Universidade Federal de Pernambuco - UFPE
Bacharelado em Ciência da Computação
Centro de Informática - CIn

SyntaxMate - Aplicativo para Consulta de Sintaxes

Amanda Costa
Bruno Martins
Erick Riso

Programação 3

Recife, Março de 2025

1. Introdução

1.1 Resumo

O aplicativo de consulta de sintaxes tem como objetivo facilitar a pesquisa de estruturas básicas de código em diferentes linguagens de programação. A aplicação será especialmente útil para pessoas que já possuem conhecimento em lógica de programação e uma linguagem específica, mas precisam consultar rapidamente sintaxes equivalentes em outras linguagens.

1.2 Justificativa

A dificuldade em lembrar sintaxes específicas de linguagens novas atrasa o desenvolvimento de projetos e compromete a produtividade. A proposta é criar um aplicativo que centralize as informações de maneira organizada e rápida

2. Requisitos

2.1 Requisitos Funcionais

RF1: Lifecycle & Intents

Descrição: A aplicação deve gerenciar corretamente o ciclo de vida das atividades e fragmentos, garantindo que o estado da aplicação seja preservado adequadamente durante mudanças de configuração e navegação. Além disso, deve utilizar Intents para iniciar atividades, serviços e compartilhar dados entre componentes do sistema.

Tipo: Obrigatório

Cumprimento: Implementado através do uso adequado do ciclo de vida das Activities e Fragments. Intents também foram utilizados para navegação.

RF2: Data Management

Descrição: A aplicação deve ser capaz de armazenar, recuperar e manipular dados de forma eficiente, utilizando bancos de dados locais (ex: Room, SQLite), armazenamento em arquivos, ou soluções baseadas na nuvem (ex: Firebase, API REST).

Tipo: Obrigatório

Cumprimento: Foi utilizado o Room para salvar favoritos e estados de buscas de forma eficiente e persistente. Persistência local implementada para armazenar Linguagens, Estruturas de Sintaxe e Favoritos. Os arquivos de repositório fazem mediação entre ViewModels e os DAOS.

RF3: Background Processing

Descrição: Deve suportar a execução de tarefas em segundo plano sem comprometer a experiência do usuário, utilizando APIs adequadas, como corrotinas, AsyncTask (obsoleto), ExecutorService ou WorkManager, dependendo do caso de uso.

Tipo: Obrigatório

Cumprimento: Esse requisito foi atendido por meio de coroutines com uso de Dispatchers.IO para garantir o não-bloqueio da UI e execução eficiente de tarefas de I/O. Co-routines foram utilizados no escopo de viewModel para chamadas assíncronas do BD.

RF5: Services & WorkManager

Descrição: Deve implementar serviços para tarefas contínuas e de longa duração, como sincronização de dados e atualizações em tempo real. O WorkManager deve ser utilizado para gerenciar tarefas agendadas e garantir a execução mesmo em condições adversas, como restrições de bateria e conectividade.

Tipo: Obrigatório

Cumprimento: O WorkManager foi utilizado para sincronização de dados em segundo plano, como inicialização do banco de dados.

RF4: Notification

Descrição: A aplicação deve ser capaz de enviar e gerenciar notificações para alertar os usuários sobre eventos relevantes, utilizando o sistema de notificações do Android (Notification Manager) e notificações push via Firebase Cloud Messaging (FCM), se necessário.

Tipo: Opcional - Trabalhos futuros

2.2 Requisitos Não-Funcionais

RNF1: Performance

Descrição: A aplicação deve ser otimizada para oferecer tempos de resposta rápidos, minimizar o consumo de memória e CPU e evitar bloqueios na interface do usuário. Técnicas como lazy loading, caching e otimização de consultas ao banco de dados devem ser aplicadas.

Tipo: Obrigatório

Cumprimento: Uso de coroutines para evitar bloqueio da UI além de consultas otimizadas com DAO e uso de StateFlow para atualizações reativas.

RNF2: Usability

Descrição: A interface deve ser intuitiva e acessível, seguindo as diretrizes do Material Design e boas práticas de UX para garantir uma experiência fluida e amigável ao usuário.

Tipo: Opcional

Cumprimento: A interface foi projetada com base nas diretrizes do Material Design 3, além de garantir navegação intuitiva com ícones e layouts responsivos.

RNF3: Security

Descrição: Deve garantir a segurança dos dados armazenados e transmitidos, aplicando criptografia, autenticação segura (OAuth, JWT), proteção contra injeção de SQL e boas práticas de segurança da plataforma Android.

Tipo: Obrigatório

Cumprimento: Uso de viewModel e repositório para encapsular lógica sensível e Room com @Query parametrizado afim de evitar SQL injection.

RNF4: Compatibility

Descrição: A aplicação deve ser compatível com diferentes versões do Android (definidas no minSdkVersion e targetSdkVersion) e adaptável a uma variedade de dispositivos, tamanhos de tela e resoluções.

Tipo: Obrigatório

Cumprimento: minSdkVersion e targetSdkVersion definidos corretamente além de testes realizados em diferentes tamanhos de tela.

RNF5: Scalability

Descrição: Deve ser desenvolvida considerando futuras expansões, permitindo a adição de novas funcionalidades sem comprometer o desempenho ou a arquitetura da aplicação. Padrões como MVVM e injeção de dependência (Hilt/Dagger) devem ser considerados para facilitar a manutenção e escalabilidade.

Tipo: Arquitetura MVVM implementada (ViewModel - Repository - DAO + Room). Projeto também foi estruturado de forma modular para permitir fácil adição de novos recursos.

3. Apresentação Técnica

3.1 Principais Funcionalidades

- **Busca por sintaxe:** O usuário pode pesquisar diretamente uma estrutura (ex.: “for”, “if”, “lista”)
- **Filtros por Linguagem:** Escolher a linguagem de programação desejada
- **Visualização em Cards:** Cada card mostra a estrutura na linguagem selecionada, com um exemplo simples
- **Favoritos:** O usuário pode marcar sintaxes favoritas para acesso rápido

3.2 Telas

1. **Tela Inicial**
    - Barra de Pesquisa no topo
    - Lista de categorias ou cards de sintaxes
    - Acesso rápido aos favoritos
    - Requisitos:
        - Exibe resultados em cards, o que atende ao objetivo de fácil visualização (RNF2)
        - Requisições podem são feitas de forma assíncrona utilizando Coroutines (RF3).
2. **Tela de Resultados** (Pode ser acoplada à Tela Inicial)
    - Exibe os cards com as sintaxes filtradas
    - Cada card contém: Nome da estrutura, exemplo na linguagem escolhida e um botão de favoritar
    - Requisitos:
        - Uso de Intents para navegação (RF1)

3. **Tela de Favoritos**
    - Lista com as sintaxes favoritadas
    - Requisitos:
        - Programa permite salvar consultas favoritas no banco de dados (RF2).

4. Trabalhos futuros

Nosso objetivo é gerar a alimentação a partir de APIs já selecionadas ao invés de um json estático para que o app se mantenha atualizado com as atualizações das linguagens e possa, inclusive, notificar usuários.