Estruturas de Dados Utilizadas no Projeto Uber SafeStart

Introdução

Este documento apresenta as principais estruturas de dados utilizadas no desenvolvimento do aplicativo Android **Uber SafeStart**.

---

1. Classes e Objetos (Modelos de Dados)

São a base da Programação Orientada a Objetos, encapsulando atributos e comportamentos.

Exemplos:

- **User.java**: representa o usuário, com atributos como username, email, gender, etc.

Útil para registro, login e controle de perfil.

- **Achievement.java**: representa conquistas, com title, description, progress, etc.

Suporta o sistema de gamificação e exibição via adapter.

- **SimulatedUser.java**: simula motoristas e passageiros para testes de pareamento.

Abstrai a geolocalização, focando no protótipo.

- **Modelos de Requisição/Resposta** (ex: LoginRequest, ProfileResponse):

Facilita a comunicação com a API via Retrofit.

Justificativa Geral: Permitem organização, tipagem forte e reuso, alinhando-se aos princípios da POO.

---

2. Coleções Dinâmicas

2.1. List (ArrayList)

Representa sequências ordenadas de elementos.

Exemplos:

- **StaticUserManager**: armazena usuários simulados.

- **AchievementsAdapter** e **AchievementsActivity**: mantêm e exibem conquistas.
- **Respostas da API**: mapeadas automaticamente de JSON para List.

Justificativa: Ideal para ordenação, iteração e acesso indexado; compatível com componentes como RecyclerView.

2.2. Map (HashMap)

Armazena pares chave-valor com busca rápida.

Exemplos:

- **AchievementTracker**: armazena metadados de conquistas indexados por ID ou tipo.
- **Corpos de requisição dinâmicos**: criados como Map<String, Object>.

Justificativa: Excelente para buscas eficientes e estruturação flexível de dados.

---

3. Primitivos e Arrays

3.1. Strings

Usadas amplamente para nomes, mensagens, tipos, chaves de preferências, URLs, etc.

3.2. Arrays Primitivos

- **byte[]**: usado em criptografia (CryptoHelper.java).
- **int[]**: IDs de ícones e textos em barras de navegação.

Justificativa: Baixo custo e ideal para dados binários ou listas fixas.

3.3. Tipos Primitivos / Wrappers

- int, boolean, float e equivalentes Integer, Boolean, Float.

Usados para IDs, flags, pontuações e avaliações.

---

4. Estruturas Específicas do Android e Bibliotecas

4.1. SharedPreferences

Armazenamento persistente de pares chave-valor (ex: token, gênero, preferências).

4.2. Intent & Bundle

Navegação entre telas e transporte de dados (ex: destino da viagem, tipo de usuário).

4.3. RecyclerView.Adapter

Exibe listas otimizadas com List<T> e ViewHolders (ex: conquistas).

4.4. JSONObject

Montagem e leitura manual de JSON criptografado (ex: EncryptionInterceptor).

4.5. okio.Buffer

Lê e escreve dados binários em requisições HTTP com OkHttp.

4.6. AtomicBoolean

Controla estado mutável em callbacks assíncronos (ex: conquista completada).

Conclusão

As estruturas de dados escolhidas no **Uber SafeStart** refletem boas práticas de engenharia de software. O projeto faz uso eficiente dos recursos da linguagem Java, do Android SDK e de bibliotecas auxiliares, garantindo clareza e desempenho.
