Offline First Pattern

Curso de Engenharia de Software
Laboratório de
Desenvolvimento de Aplicações Distribuídas e Móveis
Profs.: Hugo de Paula, Cleiton Tavares e Cristiano Neto

Implementando Padrão Offline-First no Flutter

PUC Minas

- O objetivo deste aula é apresentar a implementação do padrão de design Offline-First em um aplicativo Flutter, permitindo que o aplicativo funcione eficientemente mesmo quando não há conexão com a internet.
- Documentação de referência: https://docs.flutter.dev/app-architecture/design-patterns/offline-first

Conceito de Offline-First

- É uma abordagem de desenvolvimento que prioriza a experiência do usuário quando o dispositivo está sem conexão à internet.
- **Princípio fundamental**:
  - O aplicativo deve funcionar primeiro offline, com a conectividade sendo tratada como um aprimoramento, não como requisito.

Offline-First vs Offline-Friendly

- **Offline-Friendly**:
  - aplicativos que podem continuar funcionando quando perdem a conexão
- **Offline-First**:
  - aplicativos projetados desde o início para funcionar sem conexão, tendo isso como parte central da arquitetura

Benefícios

- Melhor experiência do usuário (menor latência, funcionamento ininterrupto)
- Redução de uso de dados
- Maior confiabilidade
- Menor dependência de qualidade de rede
- Capacidade de funcionar em emergências (quando redes podem estar sobrecarregadas)

Diferentes abordagens para implementação

- **Cache-First**: Prioriza dados armazenados localmente, atualiza quando possível
- **Server-First com Fallback**: Tenta buscar dados do servidor primeiro, usa cache como alternativa
- **Sincronização periódica**: Sincroniza dados em intervalos regulares quando conectado
- **Sincronização sob demanda**: Usuário inicia processo de sincronização manualmente
- **Fila de operações**: Armazena operações pendentes quando offline para executar quando conectado
- **Resolução de conflitos**: Estratégias para lidar com alterações que ocorrem tanto no dispositivo quanto no servidor

Arquitetura Offline-First no Flutter

- **Repositories como fonte única da verdade Padrão Repository:**
  - Uma camada de abstração entre fontes de dados e a lógica de negócios
  - **Princípio de responsabilidade única:** Repositories gerenciam acesso e manipulação de dados
  - **Fonte única da verdade:** Todas as operações de dados passam pelo Repository, garantindo consistência
  - **Vantagens:**
    - Isolamento da lógica de manipulação de dados
    - Facilidade de teste unitário
    - Desacoplamento entre UI e fontes de dados
    - Centralização da lógica de sincronização

Arquitetura Offline-First no Flutter

- **Combinação de fontes de dados locais e remotos**
  - **Fontes de dados típicas:**
    - **Remote Data Source:** API REST, GraphQL, WebSockets
    - **Local Data Source:** SQLite (sqflite), Hive, shared preferences
  - **Estratégias de combinação:**
    - Local-first: Dados locais como principal fonte, sincronização em segundo plano
    - Remote-first: Tenta remoto, cai para local em caso de falha
    - Dual-stream: Emite dados locais primeiro para UI rápida, depois atualiza com dados remotos
  - **Mapeamento de entidades:**
    - Conversão entre entidades da API e entidades locais
    - Estratégias para lidar com diferentes esquemas

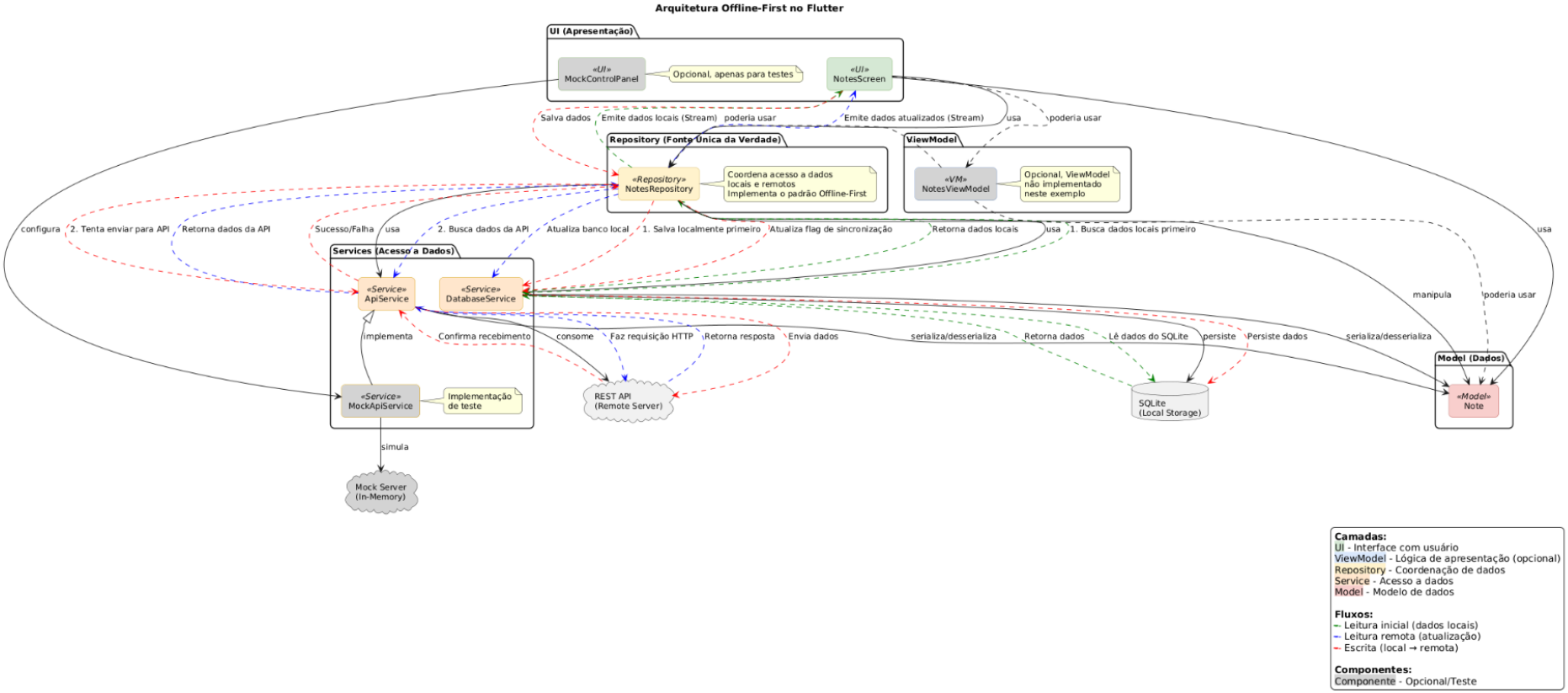
Arquitetura Offline-First no Flutter

- **Visão geral da arquitetura que será implementada**
  - **Camadas da arquitetura:**
    - **UI:** Widgets e Screens
    - **ViewModel:** Gerencia estado e conecta UI com Repository
    - **Repository:** Coordena acesso a dados de múltiplas fontes
    - **Data Services:**
      - ApiClientService: Comunicação com API REST
      - DatabaseService: Persistência local com SQLite
    - **Models:** Classes de dados que representam entidades do aplicativo

Arquitetura Offline-First no Flutter

- **Visão geral da arquitetura que será implementada**
  - **Fluxo de dados:**
    - **Leitura:**
      - ViewModel solicita dados ao Repository
      - Repository obtém dados do banco local e/ou API remota
      - Repository combina/filtra dados e retorna para ViewModel
      - ViewModel atualiza UI
    - **Escrita:**
      - ViewModel recebe ação do usuário
      - ViewModel envia dados para Repository
      - Repository salva no banco local
      - Repository tenta sincronizar com API remota
      - Repository marca status de sincronização
  - **Sincronização:**
    - Timer periódico para verificar e enviar alterações pendentes
    - Flag de sincronização para controlar estado
    - Tratamento de erros e reconexão

Arquitetura Offline-First no Flutter



Dependências necessárias

PUC Minas

```
dependencies
  flutter
    sdk  flutter
  http  ^1.1.0            # Para chamadas de API
  sqflite  ^2.3.0         # Para banco de dados local
  path_provider  ^2.1.0  # Para localização de arquivos
  connectivity_plus  ^5.0.1  # Para verificar conectividade
  freezed_annotation  ^2.4.1  # Para classes imutáveis
  json_annotation  ^4.8.1

   dev_dependencies
  build_runner  ^2.4.6
  freezed  ^2.4.5
     json_serializable  ^6.7.1
```

Testando a Aplicação

- Teste com Conectividade
  - Executar o aplicativo com conexão à internet
  - Verificar se os dados são carregados da API
  - Verificar se as alterações são sincronizadas com o servidor
- Teste sem Conectividade
  - Desativar a conexão à internet do dispositivo
  - Verificar se os dados são carregados do banco de dados local
  - Fazer alterações e verificar se o status de sincronização é atualizado
  - Reativar a conexão e verificar se as alterações são sincronizadas

Documentações

- Flutter: https://docs.flutter.dev/app-architecture
- sqflite: https://pub.dev/packages/sqflite
- freezed: https://pub.dev/packages/freezed
- connectivity_plus: https://pub.dev/packages/connectivity_plus

Conclusão

- **Offline-First**: aplicativo capaz de oferecer funcionalidade mesmo sem conexão com a internet
- **Repository**: atua como fonte única da verdade, combinando fontes de dados locais e remotas
- **Estratégias de Leitura de Dados**:
  - Usar dados locais como fallback
  - Usar Stream para combinar dados locais e remotos
  - Usar apenas dados locais com sincronização periódica
- **Estratégias de Escrita de Dados**:
  - Apenas online (para garantir consistência)
  - Offline-first (para melhor experiência do usuário)
- **Sincronização de Estado**:
  - Sincronização periódica com Timer
  - Flag de sincronização nos dados
  - Push de dados do servidor