Módulo 1: Itens Iniciais, Serialização e AsyncStorage

Este módulo define conceitos fundamentais de armazenamento e apresenta a solução mais simples para persistência local: o AsyncStorage.

|  |  |  |
| --- | --- | --- |
| Ponto Importante | Detalhe | Citação |
| **Propósito e Pré-requisitos** | O objetivo é construir aplicativos com persistência de dados. O ambiente de desenvolvimento exige instalação de **JDK, Android Studio, VS Code, MongoDB, Node.js**, além de react-native-cli e expo-cli. |  |
| **Serialização** | É o processo de expressar dados de forma contígua (sequência de bytes ou texto) para permitir sua transmissão em rede ou armazenagem em disco. No React Native, é comum usar **JSON** (JavaScript Object Notation). A transformação de objeto para JSON usa **JSON.stringify**, e a recuperação usa **JSON.parse** (de-serialização). |  |
| **Persistência Assíncrona** | Em dispositivos móveis, as operações de entrada/saída (I/O) devem ser **assíncronas** para evitar o travamento da interface gráfica. As funções assíncronas são definidas com **async** e esperam a execução com **await** ou **then**. |  |
| **AsyncStorage** | É uma forma simples de persistência assíncrona que permite guardar e recuperar **pares do tipo chave-valor**. Para usar objetos como valores, eles devem ser serializados no formato JSON. |  |
| **Métodos Principais** | Os métodos de AsyncStorage incluem: **setItem** (armazena), **getItem** (obtém), **removeItem** (remove), e **getAllKeys** (obtém todas as chaves). |  |

--------------------------------------------------------------------------------

Módulo 2: Persistência no Modelo Relacional (SQLite)

Este módulo aborda o modelo relacional e o uso do SQLite para persistência local estruturada.

|  |  |  |
| --- | --- | --- |
| Ponto Importante | Detalhe | Citação |
| **Modelo Relacional** | Baseado na **álgebra e cálculo relacional**. Repositório de dados que mantém relacionamentos consistentes entre elementos armazenados (tabelas/tuplas). |  |
| **Linguagem SQL** | É a linguagem de consulta padronizada que facilita a interoperabilidade. Os comandos principais incluem DDL (**CREATE, ALTER, DROP**) para definição de estruturas, e DML (**INSERT, UPDATE, DELETE**) para manipulação de dados. **SELECT** é o comando de consulta. |  |
| **Mapeamento Objeto-Relacional (OR/M)** | É a técnica de conversão de tabelas e registros em **coleções de entidades (objetos)** e vice-versa, para que os dados do banco relacional possam ser trabalhados em um ambiente de programação orientado a objetos. |  |
| **SQLite** | É um **banco de dados relacional minimalista que não necessita de servidor**, permitindo armazenamento local. Faz parte da arquitetura básica do Android e possui propriedades **ACID** (atomicidade, consistência, isolamento e durabilidade). |  |
| **Acesso ao SQLite** | O acesso ao banco é feito por meio de openDatabase. Comandos SQL são executados dentro de uma **transação** (transaction) utilizando o método **executeSql**. |  |

--------------------------------------------------------------------------------

Módulo 3: Persistência no Modelo Orientado a Objetos (Realm)

Este módulo apresenta as características do modelo orientado a objetos e o uso do Realm, que elimina a necessidade de mapeamento objeto-relacional.

|  |  |  |
| --- | --- | --- |
| Ponto Importante | Detalhe | Citação |
| **Bancos Orientados a Objetos (OODBMS)** | Permitem representar estruturas complexas de forma mais eficaz do que bancos relacionais, usando conceitos como herança e polimorfismo. Utilizam **ponteiros** em vez de junções, permitindo uma execução mais rápida. |  |
| **Realm** | É uma alternativa de banco de dados **open source** ao SQLite, com abordagem mais amigável ao desenvolvedor, trabalhando diretamente no **padrão orientado a objetos**. É leve, de baixo consumo de memória e adequado para trabalhar **offline**. |  |
| **Estrutura Realm** | A configuração exige a definição de um **esquema estrutural** (schema) que mapeia as classes, propriedades (atributos e tipos) e a chave primária (primaryKey). |  |
| **Manipulação de Dados** | As operações de inclusão, alteração e exclusão devem ocorrer dentro de um bloco de **escrita transacional** (db.write). Para inclusão, usa-se db.create. A alteração de um objeto é feita diretamente, sem método de update explícito, bastando alterar o atributo dentro do bloco write. |  |
| **Controle de Transações** | Transações podem ser controladas explicitamente com db.beginTransaction(), db.commitTransaction(), e db.cancelTransaction(). |  |
| **Consultas** | A consulta retorna objetos de uma coleção (ex: db.objects('Produto')), podendo ser filtrada pelo método filtered. |  |

--------------------------------------------------------------------------------

Módulo 4: Persistência no Modelo NoSQL (MongoDB) e Acesso Remoto

Este módulo foca na abordagem NoSQL para armazenamento remoto, utilizando MongoDB, Express (servidor) e Axios (cliente).

|  |  |  |
| --- | --- | --- |
| Ponto Importante | Detalhe | Citação |
| **Modelo NoSQL** | Significa **Not Only SQL**. Prioriza **disponibilidade, velocidade e escalabilidade**. Permite que as estruturas de dados sejam alteradas **dinamicamente** (esquemas flexíveis), ao contrário do modelo relacional. |  |
| **MongoDB** | Banco de dados NoSQL do tipo **documental**, baseado no formato **JSON**. Os documentos exigem um campo identificador obrigatório chamado **\_id** (chave primária). Oferece operadores específicos de consulta (ex: $gt, $in, $exists). |  |
| **Arquitetura Servidora** | O acesso ao MongoDB, por ser uma tecnologia servidora, não pode ser feito diretamente a partir do dispositivo móvel. É necessário um **servidor Web intermediário** (back-end) para gerenciar as operações. |  |
| **Servidor REST** | Um servidor minimalista pode ser criado com **Node.js** e a biblioteca **Express**. Ele segue o padrão **REST** (Representational State Transfer), utilizando os métodos HTTP para manipular os dados: **GET** (consulta), **POST** (inserção), **DELETE** (remoção). |  |
| **Tunelamento (ngrok)** | O aplicativo Android não pode usar localhost. O **ngrok** é recomendado para criar um **tunelamento** do servidor Express local, fornecendo um endereço **HTTPS** (necessário para a segurança em versões recentes do Android) para acesso remoto. |  |
| **Cliente React Native (Axios)** | O aplicativo cliente no React Native utiliza a biblioteca **Axios** para realizar chamadas REST (GET, POST, DELETE) de forma assíncrona ao endereço do servidor/ngrok. |  |

Exercícios:

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.