Dominando a Persistência de Dados com Dart e MySQL

Nesta aula vamos explorar a integração de aplicações Dart com bancos de dados relacionais, focando no MySQL.
Você aprenderá a conectar, manipular dados e aplicar boas práticas para uma persistência robusta e segura.

Produzido por MARCIO SCHOENFELDER

Agenda

O Que Veremos Hoje

01

Conectando ao MySQL com Dart

Configurando o ambiente e estabelecendo a primeira conexão.

02

CRUD: Operações Essenciais

Explorando INSERT, SELECT, UPDATE e DELETE na prática.

03

Manipulação e Tratamento de Dados

Como trabalhar com os resultados das suas consultas.

04

Boas Práticas e Segurança

Dicas para persistência de dados eficiente e segura.

Objetivo da Aula

Nosso principal objetivo é capacitar você a integrar de forma eficaz aplicações Dart com um banco de dados relacional como o MySQL. Abordaremos desde a configuração inicial até a manipulação avançada de dados, garantindo que você compreenda as ferramentas e as melhores práticas.

1

Conexão Dart MySQL

Estabelecer e gerenciar conexões persistentes.

2

Uso do Pacote mysql_client

Dominar as funcionalidades da biblioteca para interação.

3

SQL na Prática

Executar comandos DML (INSERT, SELECT, UPDATE, DELETE).

4

Segurança e Boas Práticas

Desenvolver código robusto e seguro para persistência.

O Pacote mysql_client

O **mysql_client** é o pacote essencial para conectar e interagir com bancos de dados MySQL diretamente de suas aplicações Dart. Ele oferece uma API intuitiva para gerenciar conexões, executar consultas SQL e processar os resultados.

- **Conectividade:** Permite criar conexões síncronas e assíncronas.
- **Consultas:** Suporta execução de comandos SQL simples e preparados.
- **Resultados:** Facilita a leitura e manipulação dos dados retornados.
- **Transações:** Gerenciamento de transações para garantir a integridade dos dados.

Configurando e Conectando ao MySQL

Para iniciar, você precisa adicionar o pacote **mysql1** ao seu projeto e configurar os parâmetros de conexão. É fundamental que as credenciais do banco de dados sejam gerenciadas com segurança, evitando expô-las diretamente no código-fonte.

Dependência

Adicione `mysql_client` ao seu `pubspec.yaml`.

Configuração

Defina `host`, `port`, `user`, `password`, `db`.

Conexão

Use `Connection.connect()` para estabelecer a conexão.

Executando Comandos SQL: CRUD

As operações básicas de banco de dados — Criar, Ler, Atualizar e Deletar (CRUD) — são fundamentais. O **mysql1** simplifica a execução dessas operações, permitindo interagir com suas tabelas de forma programática.

INSERT (Criar)

Adicionar novos registros à tabela. Exemplo: `conn.query('INSERT INTO users (...) VALUES (...)')`.

SELECT (Ler)

Recuperar dados do banco de dados. Exemplo: `var results = await conn.query('SELECT * FROM users')`.

UPDATE (Atualizar)

Modificar registros existentes. Exemplo: `conn.query('UPDATE users SET name = ? WHERE id = ?', [newName, userId])`.

DELETE (Deletar)

Remover registros. Exemplo: `conn.query('DELETE FROM users WHERE id = ?', [userId])`.

Manipulação de Resultados

Após executar um comando SELECT, o pacote **mysql1** retorna um objeto Results que pode ser iterado para acessar os dados. É crucial entender como extrair e trabalhar com esses dados de forma eficiente no Dart.

Exemplo de Iteração

```
for (var row in results) {
  print(row[0]); // Acessa por índice
  print(row['name']); // Acessa por nome da coluna
}
```

Tratamento de Nulos e Tipos

Sempre verifique por valores nulos e faça o cast adequado para os tipos de dados esperados. Isso evita erros em tempo de execução e garante a robustez da sua aplicação.

Considere também o uso de modelos de dados (classes Dart) para mapear os resultados do banco, tornando o código mais legível e seguro.

Boas Práticas de Persistência

Segurança & Integridade

Criptografia, controle de acesso.

Boas Práticas

Validação, normalização, versionamento.

Backup & Recuperação

Snapshots, rotinas e testes regulares.

Monitoramento & Auditoria

Logs, métricas e rastreabilidade.

Produzido por MARCIO SCHOENFELDER

Dicas de Segurança e Manutenção

SQL Injection

Sempre use **comandos preparados** (Prepared Statements) com parâmetros, em vez de concatenar strings diretamente. Isso previne ataques de SQL Injection, uma das maiores vulnerabilidades em aplicações com banco de dados.

```
await conn.query(  'INSERT INTO users (name) VALUES (?)', ['Alice']);
```

Fechamento de Conexões

Sempre feche as conexões com o banco de dados após o uso para liberar recursos. Isso evita esgotamento de conexões e melhora a performance geral da aplicação.

```
try {
    // Operações com o banco
} finally {
    await conn.close();
}
```