Configurando o Banco de Dados e Conectando ao Servidor

Objetivos da Aula:

- Compreender o que é um banco de dados e como ele é utilizado
- Configurar o banco de dados MySQL
- Criar tabelas para usuários e transações
- Configurar a conexão do banco de dados no servidor

1. O que é um Banco de Dados?

Conceito:

Um banco de dados é um sistema de armazenamento e gerenciamento de dados que permite que dados sejam armazenados de forma estruturada e consultados de maneira eficiente.

Ele é essencial para a maioria das aplicações, pois permite que informações sejam organizadas e recuperadas rapidamente.

2. Tabelas, Colunas e Linhas

Conceito:

- **Tabela**: Estrutura dentro do banco de dados que armazena dados relacionados.
- **Coluna**: Define o tipo de dados que a tabela armazena.
- **Linha**: Cada entrada de dados na tabela.

3. Configurando o Banco de Dados MySQL

Ação:

1. Baixe e instale o MySQL.

Conceito:

O MySQL é um sistema de gerenciamento de banco de dados relacional de código aberto. Ele é amplamente utilizado para armazenar e gerenciar dados em aplicações web.

Ação:

2. Crie um banco de dados e tabelas usando o MySQL Workbench ou outra ferramenta similar.

```
CREATE DATABASE finance_db;


USE finance_db;


CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255),
  email VARCHAR(255) UNIQUE,
  password VARCHAR(255),
  birth_date DATE
);
```

Para que já tinha criado sem a birth_date apenas altere a tabela

```
ALTER TABLE users ADD COLUMN birth_date DATE;


CREATE TABLE transactions (
  id INT AUTO_INCREMENT PRIMARY KEY,
  date DATE,
  amount DECIMAL(10, 2),
  description VARCHAR(255),
  category VARCHAR(255),
  account VARCHAR(255),
  user_id INT,
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Conceito:

Explicação das Queries:

- **CREATE DATABASE finance_db;:** Cria um novo banco de dados chamado **finance_db**.
- **USE finance_db;:** Seleciona o banco de dados **finance_db** para uso.
- **CREATE TABLE users (...);:** Cria uma nova tabela chamada **users** com colunas para **id, name, email, password** e **birth_date.**
- **CREATE TABLE transactions (...);:** Cria uma nova tabela chamada **transactions** com colunas para **id, date, amount, description, category, account e user_id**.

4. **Instalando Biblioteca MySQL**

Ação:

1. Instale a biblioteca **mysql2** para conectar ao banco de dados MySQL:

npm install mysql2

5. **Configurando a Conexão com o Banco de Dados**

Ação:

1. Crie uma pasta chamada **config**.

Conceito:

A pasta **config** armazenará arquivos de configuração, como a configuração da conexão ao banco de dados.

Ação:

2. Dentro da pasta **config**, crie um arquivo **db.js**.

Conceito:

O arquivo **db.js** conterá a lógica para conectar ao banco de dados MySQL.

Ação:

3. Adicione o seguinte código para importar a biblioteca mysql2 e criar a conexão:

Código javascript

```
//importar a biblioteca mysql2 e criar a conexão com o Banco de Dados

const mysql = require('mysql2'); // Importa o pacote mysql2 para conectar ao banco de dados

// Exibe as variáveis de ambiente carregadas
console.log('DB_HOST:', process.env.DB_HOST);
console.log('DB_USER:', process.env.DB_USER);
console.log('DB_PASS:', process.env.DB_PASS);
console.log('DB_NAME:', process.env.DB_NAME);
//depois pode apagar

const db = mysql.createConnection({
  host: process.env.DB_HOST, // Endereço do servidor de banco de dados
  user: process.env.DB_USER, // Nome de usuário para acessar o banco de dados
  password: process.env.DB_PASS, // Senha do usuário para acessar o banco de dados
  database: process.env.DB_NAME // Nome do banco de dados a ser acessado
});
```

Conceito:

Aqui estamos importando a biblioteca **mysql2** e criando a conexão com o banco de dados.

- **mysql.createConnection({ ... });:** Esta função cria uma nova conexão com o banco de dados utilizando as configurações fornecidas.
- **host:** O endereço do servidor de banco de dados (neste caso, localhost).
- **user:** O nome de usuário utilizado para acessar o banco de dados.
- **password:** A senha associada ao nome de usuário.
- **database:** O nome do banco de dados que queremos acessar.

Ação:

4. Adicione o seguinte código para conectar ao banco de dados e exportar a conexão:

Código javascript

```
db.connect((err) => {
 if (err) {
  console.error('Erro ao conectar ao banco de dados:', err); // Exibe mensagem de erro
  return;
 }
 console.log('Conectado ao banco de dados MySQL'); // Exibe mensagem de sucesso
});

module.exports = db; // Exporta a conexão para ser usada em outros arquivos
```

Conceito:

- **db.connect((err) => {...});:** Esta função tenta conectar ao banco de dados utilizando as configurações fornecidas. Se a conexão for bem-sucedida, uma mensagem de sucesso será exibida no console. Caso contrário, uma mensagem de erro será exibida.
- **(err) => {...}:** Esta é uma **função arrow,** são uma forma concisa de escrever funções em JavaScript. Elas são especialmente úteis para funções de **call-back** (garante que uma função não seja executada antes que uma tarefa seja concluída, mas logo depois dessa tarefa ser concluída)**.** A função aqui é executada quando o **db.connect** tenta conectar ao banco de dados. Se houver um erro, ele será passado para a função e a mensagem de erro será exibida.
- **module.exports = db;:** Exportamos a conexão para que possa ser utilizada em outros arquivos do projeto.

6. Configurando Variáveis de Ambiente

Ação:

1. Crie um arquivo **.env** na raiz do projeto.

Conceito:

O arquivo **.env** é utilizado para armazenar variáveis de ambiente, como credenciais de banco de dados e outras configurações sensíveis.

Ação:

2. Adicione as variáveis de ambiente ao arquivo **.env**:

DB_HOST=localhost

DB_USER=root

DB_PASS=sua_senha

DB_NAME=finance_db

Conceito:

Essas variáveis de ambiente serão carregadas pelo **dotenv** e utilizadas na configuração da conexão com o banco de dados.

O arquivo **.env** permite que essas variáveis sejam carregadas no **process.env** para uso no código.

7. Testando a Conexão com o Banco de Dados

Importância do dotenv:

Por que iniciar com o dotenv?

O **dotenv** carrega variáveis de ambiente definidas em um **arquivo .env** e as adiciona ao **process.env**.

Isso deve ser feito no início do código, antes de qualquer outra operação que dependa dessas variáveis. Caso contrário, as variáveis não estarão disponíveis quando forem necessárias, resultando em erros como **"Access denied for user ''@'localhost' (using password: NO)"**.

Ação:

1. Atualize o arquivo **server.js** para importar e utilizar a conexão com o banco de dados:

//Inicie com o dotenv

const dotenv = require('dotenv'); // Importa o pacote dotenv para gerenciar variáveis de **ambiente dotenv.config();** // Carrega as variáveis definidas no arquivo .env para process.env

//importar as bibliotecas

const db = require('./config/db'); // Importa a conexão com o banco de dados

Conceito:

Aqui estamos importando a conexão com o banco de dados no arquivo **server.js** para garantir que a conexão seja estabelecida quando o servidor for iniciado.

Ação:

2. Para testar a conexão com o banco de dados, abra o terminal e inicie o servidor:

npm start

Ação:

3. Verifique o console para a mensagem de sucesso:

Conectado ao banco de dados MySQL

Servidor rodando na porta 3000

Conceito:

Aqui estamos importando a conexão com o banco de dados no arquivo **server.js** para garantir que a conexão seja estabelecida quando o servidor for iniciado. É importante garantir que o **dotenv** seja carregado no início do código para que as variáveis de ambiente estejam disponíveis para uso.

Se a mensagem **"Conectado ao banco de dados MySQL"** aparecer no console, a conexão foi estabelecida com sucesso. Caso contrário, verifique as configurações no arquivo **.env** e o código **no db.js.**

Observação:

Caso veja o erro **"Client does not support authentication protocol requested by server; consider upgrading MySQL client"** ou **"Access denied for user ''@'localhost' (using password:**

NO)", indica que o método de autenticação utilizado pelo servidor MySQL não é compatível com a versão do cliente MySQL que você está usando no Node.js.

Para resolver isso, você pode seguir um destes passos:

Método 1: Alterar o Método de Autenticação do Usuário MySQL

1. **Acessar o MySQL como root**:

Abra o terminal ou prompt de comando e acesse o MySQL como usuário root:

mysql -u root -p

2. **Atualizar o método de autenticação**:

No prompt do MySQL, execute os seguintes comandos para alterar o método de autenticação do usuário MySQL:

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '142536';

FLUSH PRIVILEGES;

Certifique-se de substituir **'root'** e **'localhost'** pelo nome de usuário e host corretos, se necessário.

3. **Sair do MySQL**:

EXIT;

4. **Testar a Conexão**:

Reinicie seu servidor Node.js e veja se a conexão com o banco de dados é estabelecida corretamente.

Método 2: Atualizar o Cliente MySQL no Node.js

1. **Instalar o mysql2**:

Em vez de usar o pacote **mysql**, você pode usar o pacote **mysql2**, que oferece suporte melhorado para métodos de autenticação modernos. Para instalar o **mysql2**, use o seguinte comando:

npm install mysql2

2. **Modificar a Conexão no Código**:

No seu arquivo **config/db.js**, substitua o **mysql** pelo **mysql2** e ajuste a conexão:

Código JavaScript

```
const mysql = require('mysql2'); // Alterar para mysql2

// Cria a conexão com o banco de dados utilizando as variáveis de ambiente
const db = mysql.createConnection({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASS,
  database: process.env.DB_NAME
});

// Conecta ao banco de dados e exibe uma mensagem de sucesso ou erro
db.connect((err) => {
  if (err) {
    console.error('Erro ao conectar ao banco de dados:', err);
    return;
  }
  console.log('Conectado ao banco de dados MySQL');
});

module.exports = db;
```

3. **Testar a Conexão**:

Reinicie seu servidor Node.js e veja se a conexão com o banco de dados é estabelecida corretamente.

Conclusão

Seguindo um desses métodos, você deve ser capaz de resolver o problema de autenticação e conectar-se ao banco de dados MySQL corretamente.

O Método 1 é uma solução rápida que altera o método de autenticação no servidor MySQL, enquanto o Método 2 envolve a atualização do cliente MySQL no Node.js para suportar métodos de autenticação modernos.

Parte Prática (1 hora):

Ação:

1. Configurar o banco de dados MySQL.
2. Criar as tabelas **users** e **transactions**.
3. Instalar a biblioteca **mysql2**.
4. Configurar a conexão com o banco de dados no arquivo **db.js**.
5. Atualizar o arquivo **server.js** para utilizar a conexão com o banco de dados.
6. Testar a conexão ao banco de dados e garantir que a mensagem de sucesso seja exibida no console.