UniSENAI

Módulo 3: Integração com MySQL em Node.js

Módulo 3: Integração com MySQL em Node.js

Objetivo:

Ensinar como conectar um banco de dados MySQL a um projeto Node.js, executando operações CRUD (Create, Read, Update, Delete) e utilizando boas práticas para manipulação de dados de forma segura.

Conteúdo Detalhado:

Aula 3.1: Configuração do ambiente para MySQL e Node.js

Requisitos para Integração

- Node.js instalado.
- MySQL configurado e rodando no ambiente local.
- Editor de texto recomendado: Visual Studio Code.

Passos de Configuração

1. **Instalação do MySQL**
    - Baixe e instale o MySQL Community Server em mysql.com.
    - Configure um usuário padrão e lembre-se das credenciais.
2. **Configuração do MySQL Workbench**
    - Crie um schema chamado `projeto_node` para organização dos dados.
3. **Instalação do pacote `mysql2`**

Execute o comando:

```
npm install mysql2
```

4. **Configuração do projeto**

Crie uma pasta chamada `projeto-mysql` e inicialize com:

```
npm init -y
```

Aula 3.2: Conectando o Node.js ao MySQL

Criação de Conexão

- Exemplo de conexão básica:

```
const mysql = require('mysql2');

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'sua_senha',
  database: 'projeto_node'
});

connection.connect((err) => {
  if (err) {
    console.error('Erro ao conectar ao MySQL:', err);
    return;
  }
  console.log('Conectado ao MySQL com sucesso!');
});
```

Configuração de Variáveis de Ambiente

Use o pacote `dotenv`:

```
npm install dotenv
```

Arquivo `.env`:

```
DB_HOST=localhost
DB_USER=root
DB_PASS=sua_senha
DB_NAME=projeto_node
```

Integração no código:

```
require('dotenv').config();

const connection = mysql.createConnection({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASS,
  database: process.env.DB_NAME
});
```

Aula 3.3: Executando Consultas no MySQL via Node.js

Estrutura Básica de Consultas

- Exemplo de SELECT:

```
connection.query('SELECT * FROM usuarios', (err, results) => {
  if (err) {
    console.error('Erro ao executar consulta:', err);
    return;
  }
  console.log('Resultados:', results);
});
```

Consultas Assíncronas

- Use `mysql2/promise`:

```
const mysql = require('mysql2/promise');

async function executarConsulta() {
  const connection = await mysql.createConnection({
    host: process.env.DB_HOST,
    user: process.env.DB_USER,
    password: process.env.DB_PASS,
    database: process.env.DB_NAME
  });

  const [rows] = await connection.execute('SELECT * FROM usuarios');
  console.log(rows);
}

executarConsulta();
```

Manipulação de Erros

- Adicione tratamento de erros para conexões e consultas.

Aula 3.4: Implementando Operações CRUD no Node.js com MySQL

Inserindo Dados (Create)

```
const usuario = { nome: 'João', email: 'joao@email.com' };
connection.query('INSERT INTO usuarios (nome, email) VALUES (?, ?)', [usuario.nome,
usuario.email], (err) => {
  if (err) {
    console.error('Erro ao inserir dados:', err);
    return;
  }
  console.log('Usuário inserido com sucesso!');
});
```

Consultando Dados (Read)

- Consulta simples:

```
connection.query('SELECT * FROM usuarios', (err, results) => {
  if (err) throw err;
  console.log(results);
});
```

Atualizando Dados (Update)

```
connection.query('UPDATE usuarios SET nome = ? WHERE id = ?', ['Carlos', 1], (err) =>
{
  if (err) throw err;
  console.log('Usuário atualizado com sucesso!');
});
```

Excluindo Dados (Delete)

```
connection.query('DELETE FROM usuarios WHERE id = ?', [1], (err) => {
  if (err) throw err;
  console.log('Usuário excluído com sucesso!');
});
```

Uso de Prepared Statements

- Proteção contra SQL Injection:

```
const nome = 'Carlos';
connection.query('SELECT * FROM usuarios WHERE nome = ?', [nome], (err, results)
=> {
  if (err) throw err;
  console.log(results);
});
```

Aula 3.5: Conexão com Pool e Boas Práticas

Conexão com Pool

```
const pool = mysql.createPool({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASS,
  database: process.env.DB_NAME,
  waitForConnections: true,
  connectionLimit: 10
});

pool.query('SELECT * FROM usuarios', (err, results) => {
  if (err) throw err;
  console.log(results);
});
```

Estruturação do Código em Módulos

- Organize conexões e consultas em arquivos separados:

`db.js`:

```
const mysql = require('mysql2/promise');

const pool = mysql.createPool({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASS,
  database: process.env.DB_NAME,
  waitForConnections: true,
  connectionLimit: 10
});

module.exports = pool;
```

`usuarios.js`:

```
const pool = require('./db');

async function listarUsuarios() {
  const [rows] = await pool.query('SELECT * FROM usuarios');
  return rows;
}

module.exports = { listarUsuarios };
```

Logs e Monitoramento

- Use pacotes como `winston` ou `pino` para registrar logs.
- Configure alertas para erros críticos.

Lista de Exercícios

Questões Teóricas

1. Explique a diferença entre `mysql2` e `mysql2/promise`.
2. Quais as vantagens de usar variáveis de ambiente na conexão com o banco de dados?
3. O que é um prepared statement e como ele ajuda na segurança da aplicação?
4. Por que é recomendável usar conexões em pool em aplicações Node.js?
5. Descreva o papel do módulo `dotenv` em projetos Node.js.

6. Qual é a estrutura básica para realizar uma consulta SELECT com o pacote `mysql2`?
7. Liste três boas práticas ao conectar o Node.js com o MySQL.
8. Explique o conceito de transações no MySQL e como ele pode ser implementado em Node.js.
9. Quais erros podem ocorrer ao conectar ao MySQL, e como tratá-los no Node.js?
10. Por que é importante organizar o código em módulos ao trabalhar com bancos de dados?

Instruções para a Entrega das Atividades

1. **Elaboração e Envio do Arquivo**
   - Responda todas as questões de forma clara e objetiva.
   - Gere um arquivo no formato **.PDF** contendo as respostas de cada questão.
   - Envie o arquivo para os e-mails dos professores responsáveis.
2. **Validação da Atividade**
   - Após o envio do arquivo, procure o(s) professor(es) para realizar a validação da atividade.
   - **Não inicie a próxima atividade sem antes validar a anterior com o professor.**
3. **Forma de Validação**
   - **Explicação Verbal**: Explique cada resposta verbalmente ao(s) professor(es).
   - **Perguntas e Respostas**: Esteja preparado para responder aos questionamentos do(s) professor(es) sobre o conteúdo das respostas.
   - **Orientação**: Receba orientações sobre a apresentação do(s) tema(s).