UniSENAI

Trilha 06

Integração com Back-End

6 - Integração com Back-End (Node.js e MySQL)

Nesta unidade, exploraremos a **integração do Vue.js com um back-end baseado em Node.js e MySQL**. Esse processo permitirá criar aplicações completas, onde o front-end Vue.js consome dados de uma API construída em Node.js, utilizando **Express.js** como framework e **Sequelize** como ORM para comunicação com o banco de dados MySQL.

6.1 Vue.js e Node.js

A combinação do **Vue.js no front-end** com **Node.js no back-end** é uma abordagem poderosa e moderna para o desenvolvimento full-stack. O Vue.js cuida da interface do usuário e da interação com a API, enquanto o Node.js, junto com o Express.js, gerencia as requisições e acessa o banco de dados.

Vantagens da Integração Vue.js + Node.js

✔ **Alta performance** – O Node.js é baseado em eventos assíncronos, tornando-o rápido para lidar com múltiplas requisições.
✔ **JavaScript full-stack** – Permite o uso da mesma linguagem (JavaScript) no front-end e no back-end.
✔ **Escalabilidade** – O modelo assíncrono do Node.js facilita a construção de aplicações escaláveis.
✔ **Facilidade de integração** – Axios, Fetch API e WebSockets permitem consumir APIs com facilidade no Vue.js.

6.2 Introdução ao Node.js e Express

O que é o Node.js?

O **Node.js** é um ambiente de execução JavaScript no lado do servidor, permitindo criar aplicações web escaláveis e assíncronas. Ele utiliza o **V8 JavaScript Engine** (o mesmo do Google Chrome) para interpretar e executar código JavaScript.

O que é o Express.js?

O **Express.js** é um framework minimalista para Node.js que facilita a criação de servidores e APIs RESTful, proporcionando um conjunto robusto de funcionalidades para lidar com requisições HTTP.

Instalação do Node.js e Express

Passo 1: Instalar o Node.js

Baixe e instale o Node.js a partir do site oficial:
🔗 https://nodejs.org/

Verifique a instalação rodando os comandos no terminal:

```
node -v
npm -v
```

Passo 2: Criar um projeto Node.js

```
mkdir backend
cd backend
npm init -y
```

Esse comando cria um arquivo `package.json` que gerencia as dependências do projeto.

Passo 3: Instalar o Express.js

```
npm install express
```

Criando um Servidor Express Básico

Crie um arquivo `server.js` e adicione o seguinte código:

```
const express = require('express');
const app = express();
const PORT = 3000;

app.get('/', (req, res) => {
  res.send('Servidor Express rodando com sucesso!');
});

app.listen(PORT, () => {
  console.log(`Servidor rodando em http://localhost:${PORT}`);
});
```

Agora, inicie o servidor com:

```
node server.js
```

Acesse `http://localhost:3000` no navegador e veja a mensagem.

6.3 Configuração do Banco de Dados MySQL com Sequelize

O que é o Sequelize?

O **Sequelize** é um **ORM (Object-Relational Mapper)** para Node.js que simplifica a interação com bancos de dados SQL, permitindo definir tabelas e manipular registros de forma programática.

Instalação do MySQL e Sequelize

Antes de continuar, instale o MySQL no seu ambiente. Após a instalação, crie um banco de dados chamado `meubanco`.

Agora, instale o Sequelize e os drivers do MySQL:

```
npm install sequelize mysql2
```

Crie um arquivo `config/database.js` para configurar a conexão com o banco de dados:

```
const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('meubanco', 'root', 'senha', {
  host: 'localhost',
  dialect: 'mysql'
});

sequelize.authenticate()
  .then(() => console.log('Conexão com o MySQL estabelecida!'))
  .catch(err => console.error('Erro ao conectar:', err));

module.exports = sequelize;
```

Para testar, execute:

```
node config/database.js
```

Se a conexão for bem-sucedida, a mensagem **"Conexão com o MySQL estabelecida!"** será exibida.

Definição de Modelos no Sequelize

Crie um modelo de usuário em `models/Usuario.js`:

```
const { Sequelize, DataTypes } = require('sequelize');
const sequelize = require('../config/database');

const Usuario = sequelize.define('Usuario', {
  nome: { type: DataTypes.STRING, allowNull: false },
  email: { type: DataTypes.STRING, allowNull: false, unique: true },
  senha: { type: DataTypes.STRING, allowNull: false }
});

sequelize.sync(); // Cria a tabela caso ela não exista

module.exports = Usuario;
```

6.4 Criação de APIs RESTful

Agora, vamos criar uma API RESTful que permite **CRUD (Create, Read, Update, Delete)** para usuários.

Criando Rotas para a API

Edite `server.js` e adicione:

```
const express = require('express');
const Usuario = require('./models/Usuario');

const app = express();
app.use(express.json());

// Criar usuário
app.post('/usuarios', async (req, res) => {
  const usuario = await Usuario.create(req.body);
  res.json(usuario);
});

// Listar usuários
app.get('/usuarios', async (req, res) => {
  const usuarios = await Usuario.findAll();
  res.json(usuarios);
});

// Atualizar usuário
app.put('/usuarios/:id', async (req, res) => {
  await Usuario.update(req.body, { where: { id: req.params.id } });
  res.send('Usuário atualizado!');
});

// Deletar usuário
app.delete('/usuarios/:id', async (req, res) => {
  await Usuario.destroy({ where: { id: req.params.id } });
  res.send('Usuário deletado!');
});

app.listen(3000, () => console.log('API rodando em http://localhost:3000'));
```

Agora, a API está pronta para manipular usuários.

6.5 Consumo de Dados no Vue.js com Axios

No Vue.js, utilizamos o **Axios** para consumir APIs.

Instale o Axios no projeto Vue:

```
npm install axios
```

Agora, crie um serviço em `services/api.js`:

```
import axios from 'axios';

const api = axios.create({
  baseURL: 'http://localhost:3000'
});

export default api;
```

Exemplo de Consumo no Vue.js

Crie um componente `Usuarios.vue`:

```
<template>
  <div>
    <h2>Lista de Usuários</h2>
    <ul>
      <li v-for="usuario in usuarios" :key="usuario.id">
        {{ usuario.nome }} - {{ usuario.email }}
      </li>
    </ul>
  </div>
</template>

<script>
import api from '../services/api';

export default {
  data() {
    return {
      usuarios: []
    };
  },
  async created() {
    const response = await api.get('/usuarios');
    this.usuarios = response.data;
  }
};
</script>
```

6.6 Autenticação e Gerenciamento de Sessões

Para autenticação, utilizamos **JSON Web Token (JWT)**.

Instale a biblioteca:

```
npm install jsonwebtoken bcryptjs
```

Agora, crie um middleware `middlewares/auth.js`:

```
const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {
  const token = req.headers.authorization;
  if (!token) return res.status(401).json({ erro: 'Acesso negado!' });

  jwt.verify(token, 'segredo', (err, decoded) => {
    if (err) return res.status(401).json({ erro: 'Token inválido!' });
    req.userId = decoded.id;
    next();
  });
};
```

Adicione autenticação à API e proteja rotas, garantindo que apenas usuários logados possam acessá-las.

Conclusão

✔ Criamos um **back-end em Node.js** com **Express.js**.
✔ Configuramos o **Sequelize** para interagir com **MySQL**.
✔ Construímos uma **API RESTful** e consumimos no **Vue.js com Axios**.
✔ Implementamos **autenticação JWT** para segurança.

Com esse conhecimento, você pode desenvolver **aplicações full-stack modernas e seguras!** 🚀

Atividades - Integração com Back-End (Node.js e MySQL)

Questões Teóricas (10 questões)

1. **O que é o Node.js e por que ele é uma escolha popular para o desenvolvimento de back-end?**
2. **Explique o papel do Express.js no desenvolvimento de APIs em Node.js. Quais são suas principais vantagens?**
3. **O que é um ORM e qual a vantagem de utilizar o Sequelize para manipular bancos de dados no Node.js?**
4. **Como funciona a comunicação entre Vue.js e Node.js? Qual a importância do uso de Axios nesse processo?**
5. **Explique o conceito de APIs RESTful e a importância dos métodos HTTP (`GET`, `POST`, `PUT`, `DELETE`) na construção dessas APIs.**
6. **O que é um middleware no Express.js e como ele pode ser utilizado para manipular requisições e respostas em uma API?**
7. **Quais são os benefícios de utilizar JWT (JSON Web Token) na autenticação de usuários em uma aplicação Node.js?**
8. **Explique a diferença entre uma requisição síncrona e uma requisição assíncrona no Node.js. Como o uso de `async/await` facilita a programação assíncrona?**
9. **Quais são os principais desafios ao integrar um banco de dados MySQL a uma aplicação Node.js e como o Sequelize ajuda a resolver esses desafios?**
10. **Quais são as boas práticas para estruturar um projeto de back-end em Node.js com Express e Sequelize?**

Questões Práticas (10 questões)

1. **Crie um servidor básico utilizando Express.js que responda com a mensagem "Servidor rodando corretamente!" ao acessar a rota `/`.**
2. **Configure uma conexão com um banco de dados MySQL utilizando Sequelize. Certifique-se de incluir o tratamento de erros caso a conexão falhe.**
3. **Crie um modelo Sequelize para representar usuários em um banco de dados. O modelo deve conter campos para `id`, `nome`, `email` e `senha`.**
4. **Implemente um endpoint `/usuarios` na API que permita criar um novo usuário no banco de dados utilizando o método `POST`.**
5. **Desenvolva uma rota `/usuarios` no Express.js que retorne uma lista de usuários cadastrados no banco de dados.**
6. **Utilizando Vue.js e Axios, crie um componente que consuma a API e exiba a lista de usuários cadastrados.**

7. **Crie um endpoint no Express.js que permita a atualização dos dados de um usuário específico utilizando o método PUT.**
8. **Implemente um sistema de autenticação no back-end utilizando JWT. O sistema deve validar o login do usuário e gerar um token de acesso.**
9. **Proteja uma rota da API utilizando middleware de autenticação JWT. Somente usuários autenticados devem acessar essa rota.**
10. **Crie um formulário de login no Vue.js que envie as credenciais do usuário para a API. Caso o login seja bem-sucedido, armazene o token JWT e redirecione o usuário para uma página protegida.**

- *Responda todas as questões teóricas e práticas.*
- *Organize as respostas em um arquivo PDF, contendo nome e data.*
- *Envie o PDF aos professores responsáveis, seguindo o padrão de nomenclatura.*
- *Valide o material com um professor antes de prosseguir.*
- *Certifique-se de cumprir o prazo de entrega.*

Padrão de nomeclatura
NomeCompleto_TrilhaX_DataDeEntrega.pdf
Exemplo: JoãoSilva_Trilha1_2025-01-30.pdf