1. O que são templating engines e qual sua utilidade?

Templating engines são ferramentas que permitem gerar HTML dinâmico combinando arquivos de template com dados fornecidos pela aplicação. Elas facilitam a construção de páginas web, permitindo que a estrutura do HTML seja definida em templates reutilizáveis, nos quais dados variáveis são inseridos em placeholders. Isso simplifica a manutenção e evita a repetição de código, além de possibilitar a separação entre a lógica da aplicação e a apresentação visual.

2. Explique as diferenças entre EJS e Pug.

EJS (Embedded JavaScript) e Pug são duas populares templating engines para Node.js, mas com abordagens diferentes. O EJS usa sintaxe muito próxima do HTML tradicional, permitindo inserir código JavaScript diretamente em tags <% %>, o que facilita a curva de aprendizado para quem já conhece HTML. Já o Pug utiliza uma sintaxe indentada e minimalista, eliminando muitas tags HTML tradicionais, o que torna o código mais limpo e conciso, mas exige adaptação. Além disso, Pug possui recursos avançados para composição de templates, como mixins, mas pode ser menos intuitivo para iniciantes.

3. Como configurar o EJS em um projeto Express?

Para configurar o EJS em um projeto Express, basta instalar o pacote ejs via npm, definir a engine de visualização com app.set('view engine', 'ejs') e criar uma pasta chamada views onde os arquivos .ejs serão armazenados. Ao usar res.render('nomeDoTemplate', dados), o Express irá processar o arquivo .ejs correspondente dentro da pasta views, inserindo os dados no template e retornando o HTML resultante.

4. O que são partials e como eles auxiliam na reutilização de código?

Partials são pequenos arquivos de template que representam fragmentos reutilizáveis de código HTML, como cabeçalhos, rodapés, menus ou componentes comuns. Eles ajudam a evitar a duplicação de código, permitindo incluir esses trechos em vários templates diferentes por meio de comandos de inclusão (como <%- include('partial') %> no EJS). Isso facilita a manutenção e atualização, pois mudanças feitas em um partial são refletidas em todas as páginas que o utilizam.

5. Como os dados do MySQL podem ser renderizados em um template?

Para renderizar dados do MySQL em um template, a aplicação precisa primeiro consultar o banco de dados usando um cliente MySQL para Node.js, como mysql2 ou sequelize. Após

recuperar os dados em formato de objeto ou array, esses dados são passados para o método res.render do Express, que os insere no template na forma de variáveis. O template então exibe essas informações dinamicamente, por exemplo, listando registros em tabelas ou cards na página HTML.

6. Explique o papel do body-parser em aplicações Express.

O body-parser é um middleware para Express que processa o corpo (body) das requisições HTTP, convertendo os dados recebidos em formatos como JSON ou URL-encoded para objetos JavaScript acessíveis no req.body. Isso é essencial para manipular dados enviados via formulários ou requisições AJAX, permitindo que a aplicação leia e utilize esses dados para processamento, validação ou armazenamento.

7. Quais são as boas práticas ao organizar templates em um projeto?

Boas práticas para organizar templates incluem separar arquivos por tipo ou funcionalidade, como ter pastas para layouts, partials e páginas específicas. Utilizar partials para componentes reutilizáveis (cabeçalhos, rodapés, menus) evita duplicação. Manter uma estrutura clara e coerente, nomear arquivos e pastas de forma intuitiva, e documentar o uso dos templates ajuda na manutenção e escalabilidade. Além disso, organizar templates para facilitar a aplicação de temas ou internacionalização também é recomendado.

8. Como funciona a inclusão de placeholders em layouts do EJS?

No EJS, layouts são arquivos que definem a estrutura básica de páginas, incluindo placeholders onde o conteúdo específico de cada página será inserido. Uma forma comum de fazer isso é criar partials para o cabeçalho e rodapé, e em cada template incluir esses partials para montar a página completa. O EJS não tem suporte nativo para layouts como alguns outros motores, mas pode-se usar técnicas como incluir arquivos (<%- include('header') %>) para simular placeholders.

9. Liste três aplicações práticas para o uso de formulários no Express.

Formulários em Express são usados para: (1) autenticação de usuários, coletando dados como login e senha; (2) cadastro ou atualização de informações de usuários ou produtos, permitindo interação dinâmica com o banco de dados; (3) envio de feedback ou contato, onde o usuário submete mensagens que podem ser armazenadas ou enviadas por e-mail.

10. Por que é importante validar dados recebidos de formulários?

Validar dados de formulários é fundamental para garantir a integridade, segurança e qualidade das informações que a aplicação recebe. Sem validação, dados incorretos, mal formatados ou maliciosos podem causar falhas, vulnerabilidades como injeção de código, ou gerar inconsistências no banco de dados. A validação assegura que apenas dados adequados e esperados sejam processados, melhorando a confiabilidade do sistema e a experiência do usuário.

Questões Práticas

1. Configure o EJS em um projeto Express e crie um template básico.

Para configurar o motor de templates EJS em um projeto Node.js utilizando o framework Express, primeiramente é necessário criar um ambiente de desenvolvimento adequado, o qual inclui a inicialização do gerenciador de pacotes npm na raiz do projeto. Isso pode ser realizado por meio do comando "npm init", que gera o arquivo "package.json", essencial para o gerenciamento das dependências da aplicação.

Em seguida, procede-se à instalação das bibliotecas necessárias, especificamente o Express, que proverá o servidor web, e o EJS, responsável pelo processamento dos templates, por meio do comando "npm install express ejs".

Com as dependências instaladas, deve-se criar o arquivo principal do servidor, convencionalmente denominado "server.js". Nesse arquivo, importa-se o módulo Express e inicializa-se uma instância da aplicação. Para habilitar o uso do EJS, configura-se a engine de visualização por meio do método "app.set("view engine", "ejs")". Essa configuração instrui o Express a utilizar o EJS para renderizar os arquivos de template.

Posteriormente, define-se uma rota básica, por exemplo, a raiz do servidor ("/"), na qual se invoca o método "res.render()" para renderizar um arquivo de template. Este método recebe como parâmetros o nome do template (sem a extensão ".ejs") e um objeto contendo dados que serão injetados no template, permitindo a geração de conteúdo dinâmico.

Os arquivos de template devem ser organizados dentro de uma pasta denominada "views", que é o diretório padrão onde o Express procura os arquivos para renderização. Dentro dessa pasta, cria-se o arquivo de template, por exemplo, "index.ejs", que contém a estrutura HTML com placeholders identificados pela sintaxe "<%= %>", onde serão inseridos os valores dinâmicos passados pelo servidor.

Para finalizar, inicia-se o servidor escutando em uma porta definida, geralmente a 3000, utilizando o método "app.listen()". Com o servidor em execução, a aplicação está apta a processar requisições, renderizar os templates EJS e retornar páginas HTML dinâmicas ao cliente.

Em suma, a configuração do EJS em um projeto Express consiste na instalação das dependências, configuração do motor de template, criação de rotas que renderizam templates dinâmicos e organização adequada dos arquivos de visualização, resultando em uma arquitetura que separa claramente a lógica do servidor da camada de apresentação.

Arquivos:

server.js

```
const express = require('express');
const app = express();
const PORT = 3000;
app.set('view engine', 'ejs');
app.get('/', (req, res) => {
  res.render('index', {
    title: 'Página Inicial',
    message: 'Olá, esse é um template básico com EJS!'
  });
});
app.listen(PORT, () => {
  console.log(`Servidor rodando em http://localhost:${PORT}`);
});
```

index.ejs

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8" />
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title><%= title %></title>
</head>
<body>
  <h1><%= message %></h1>
</body>
</html>
```

2. Implemente um template que liste usuários cadastrados em um banco MySQL.

```
const express = require('express');
const mysql = require('mysql2/promise');
const path = require('path');
const app = express();
const PORT = 3000;
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
const dbConfig = {
  host: 'localhost',
  user: 'seu_usuario',
  password: 'sua_senha',
  database: 'seu_banco'
};
app.get('/users', async (req, res) => {
  try {
    const connection = await mysql.createConnection(dbConfig);
   const [rows] = await connection.execute('SELECT id, name, email FROM users');
    await connection.end();
    res.render('users', { users: rows });
  } catch (error) {
    console.error('Erro ao buscar usuários:', error);
```

```
    res.status(500).send('Erro ao acessar o banco de dados');
  }
});
app.listen(PORT, () => {
  console.log(`Servidor rodando em http://localhost:${PORT}`);
});
```

Banco MySQL

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  email VARCHAR(100)
);
```

server.js

```
const express = require('express');
const mysql = require('mysql2/promise');
const path = require('path');
const app = express();
const PORT = 3000;
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
const dbConfig = {
  host: 'localhost',
  user: 'seu_usuario',
  password: 'sua_senha',
  database: 'seu_banco'
};

app.get('/users', async (req, res) => {
  try {
```

```
    const connection = await mysql.createConnection(dbConfig);
    const [rows] = await connection.execute('SELECT id, name, email FROM users');
    await connection.end();

    res.render('users', { users: rows });
  } catch (error) {
    console.error('Erro ao buscar usuários:', error);
    res.status(500).send('Erro ao acessar o banco de dados');
  }
});

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

users.ejs

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Lista de Usuários</title>
</head>
<body>
  <h1>Usuários Cadastrados</h1>

  <% if (users.length === 0) { %>
    <p>Nenhum usuário cadastrado.</p>
  <% } else { %>
    <ul>
```

```
    <% users.forEach(user => { %>
      <li><strong><%= user.name %></strong> - <%= user.email %></li>
    <% }) %>
   </ul>
 <% } %>
</body>
</html>
```

3. Crie um layout com cabeçalho e rodapé utilizando partials.

Arquivos:

header.ejs

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title><%= title %></title>
</head>
<body>
  <header>
    <h1>Meu Site</h1>
    <nav>
      <a href="/">Início</a> |
      <a href="/users">Usuários</a>
    </nav>
    <hr>
  </header>
```

footer.ejs

```
  <hr>
  <footer>
```

```
  <p>&copy; <%= new Date().getFullYear() %> - Todos os direitos reservados</p>
 </footer>
</body>
</html>
```

home.ejs

```
<%- include('../partials/header', { title: 'Página Inicial' }) %>
<main>
 <h2>Bem-vindo ao meu site!</h2>
 <p>Essa é a página inicial, com conteúdo dinâmico.</p>
</main>
<%- include('../partials/footer') %>
```

server.js

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

app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

app.get('/', (req, res) => {
 res.render('pages/home', { title: 'Página Inicial' });
});

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

4. Desenvolva um formulário para cadastrar novos usuários no banco de dados.

SQL

```
CREATE TABLE users (
```

```
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL
);
```

server.js

```
const express = require('express');
const mysql = require('mysql2/promise');
const bodyParser = require('body-parser');
const path = require('path');

const app = express();
const PORT = 3000;
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
app.use(bodyParser.urlencoded({ extended: false }));
app.use(express.static('public'));
const dbConfig = {
  host: 'localhost',
  user: 'seu_usuario',
  password: 'sua_senha',
  database: 'seu_banco'
};
app.get('/cadastro', (req, res) => {
  res.render('form');
});
app.post('/cadastro', async (req, res) => {
  const { name, email } = req.body;

  try {
```

```
    const conn = await mysql.createConnection(dbConfig);
    await conn.execute('INSERT INTO users (name, email) VALUES (?, ?)', [name, email]);
    await conn.end();
    res.render('success', { name });
  } catch (error) {
    console.error('Erro ao inserir usuário:', error);
    res.status(500).send('Erro ao cadastrar usuário.');
  }
});
app.listen(PORT, () => {
  console.log(`Servidor rodando em http://localhost:${PORT}`);
});
```

forms.ejs

```
<%- include('partials/header', { title: 'Cadastro de Usuário' }) %>
<h2>Cadastro de Novo Usuário</h2>
<form action="/cadastro" method="POST">
  <label for="name">Nome:</label><br>
  <input type="text" id="name" name="name" required><br><br>

  <label for="email">E-mail:</label><br>
  <input type="email" id="email" name="email" required><br><br>

  <button type="submit">Cadastrar</button>
</form>

<%- include('partials/footer') %>
```

success.ejs

```
<%- include('partials/header', { title: 'Cadastro Realizado' }) %>
<h2>Usuário cadastrado com sucesso!</h2>
<p>Bem-vindo, <strong><%= name %></strong>!</p>

<a href="/cadastro">Cadastrar outro usuário</a>

<%- include('partials/footer') %>
```

Parcial header.ejs

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <title><%= title %></title>
</head>
<body>
  <header>
    <h1>Meu Site</h1>
    <hr>
  </header>
```

Parcial footer.ejs

```
  <hr>
  <footer>
    <p>&copy; <%= new Date().getFullYear() %> - Todos os direitos reservados</p>
  </footer>
</body>
```

```
</html>
```

5. Configure uma rota POST que receba os dados do formulário e insira no MySQL.

SQL

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL
);
```

server.js

```
const express = require('express');
const mysql = require('mysql2/promise');
const bodyParser = require('body-parser');
const path = require('path');

const app = express();
const PORT = 3000;
const dbConfig = {
  host: 'localhost',
  user: 'seu_usuario',
  password: 'sua_senha',
  database: 'seu_banco'
};
app.use(bodyParser.urlencoded({ extended: false }));
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
app.get('/cadastro', (req, res) => {
  res.render('form');
});
app.post('/cadastro', async (req, res) => {
  const { name, email } = req.body;
```

```
  try {
    const connection = await mysql.createConnection(dbConfig);
    await connection.execute(
      'INSERT INTO users (name, email) VALUES (?, ?)',
      [name, email]
    );
    await connection.end();

    res.send(`<p>Usuário <strong>${name}</strong> cadastrado com sucesso!</p><a
href="/cadastro">Voltar</a>`);
  } catch (error) {
    console.error('Erro ao cadastrar usuário:', error);
    res.status(500).send('Erro ao inserir no banco de dados.');
  }
});
app.listen(PORT, () => {
  console.log(`Servidor rodando em http://localhost:${PORT}`);
});
```

form.ejs

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <title>Cadastro</title>
</head>
<body>
  <h2>Formulário de Cadastro</h2>
  <form action="/cadastro" method="POST">
    <label for="name">Nome:</label><br>
    <input type="text" name="name" id="name" required><br><br>
```

```
    <label for="email">E-mail:</label><br>
    <input type="email" name="email" id="email" required><br><br>

    <button type="submit">Cadastrar</button>
  </form>
</body>
</html>
```

6. Crie um template que exiba mensagens de erro ao processar dados.

Arquivos

error.ejs

```
<%- include('partials/header', { title: 'Erro' }) %>

<h2>Ocorreu um erro</h2>

<p style="color: red;"><%= errorMessage %></p>

<a href="/cadastro">Voltar ao formulário</a>

<%- include('partials/footer') %>
```

server.js

```
app.post('/cadastro', async (req, res) => {
  const { name, email } = req.body;

  try {
    const connection = await mysql.createConnection(dbConfig);
    await connection.execute(
```

```
      'INSERT INTO users (name, email) VALUES (?, ?)',
      [name, email]
    );
    await connection.end();

    res.send(`<p>Usuário <strong>${name}</strong> cadastrado com sucesso!</p><a href="/cadastro">Voltar</a>`);
  } catch (error) {
    console.error('Erro ao cadastrar usuário:', error);
    res.status(500).render('error', {
      errorMessage: 'Não foi possível salvar os dados. Tente novamente mais tarde.'
    });
  }
});
```

header.ejs

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <title><%= title %></title>
</head>
<body>
  <header>
    <h1>Meu Site</h1>
    <hr>
  </header>

  <hr>
  <footer>
    <p>&copy; <%= new Date().getFullYear() %> - Todos os direitos reservados</p>
```

```
  </footer>
</body>
</html>
```

7. Implemente um sistema de navegação entre páginas utilizando layouts reutilizáveis.

SQL

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL
);
```

Arquivos

server.js

```
const express = require('express');
const mysql = require('mysql2/promise');
const bodyParser = require('body-parser');
const path = require('path');

const app = express();
const PORT = 3000;

const dbConfig = {
  host: 'localhost',
  user: 'seu_usuario',
  password: 'sua_senha',
  database: 'seu_banco'
};

app.use(bodyParser.urlencoded({ extended: false }));
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
```

```
app.get('/', (req, res) => {
  res.render('pages/home', { title: 'Início' });
});

app.get('/cadastro', (req, res) => {
  res.render('pages/form', { title: 'Cadastro' });
});

app.get('/usuarios', async (req, res) => {
  try {
    const connection = await mysql.createConnection(dbConfig);
    const [users] = await connection.execute('SELECT * FROM users');
    await connection.end();
    res.render('pages/users', { title: 'Usuários Cadastrados', users });
  } catch (error) {
    res.status(500).render('pages/error', {
      title: 'Erro',
      errorMessage: 'Erro ao buscar usuários.'
    });
  }
});

app.post('/cadastro', async (req, res) => {
  const { name, email } = req.body;

  try {
    const connection = await mysql.createConnection(dbConfig);
    await connection.execute(
      'INSERT INTO users (name, email) VALUES (?, ?)',
      [name, email]
```

```
    );
    await connection.end();
    res.redirect('/usuarios');
  } catch (error) {
    res.status(500).render('pages/error', {
      title: 'Erro',
      errorMessage: 'Não foi possível salvar os dados. Tente novamente mais tarde.'
    });
  }
});

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

header.ejs

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <title><%= title %></title>
</head>
<body>
  <header>
    <h1>Meu Site</h1>
    <nav>
      <a href="/">Início</a> |
      <a href="/cadastro">Cadastro</a> |
      <a href="/usuarios">Usuários</a>
    </nav>
    <hr>
  </header>
```

footer.ejs

```
  <hr>
  <footer>
    <p>&copy; <%= new Date().getFullYear() %> - Todos os direitos reservados</p>
  </footer>
</body>
</html>
```

home.ejs

```
<%- include('../partials/header', { title }) %>

<h2>Bem-vindo ao sistema</h2>
<p>Use o menu para navegar entre as páginas.</p>

<%- include('../partials/footer') %>
```

form.ejs

```
<%- include('../partials/header', { title }) %>
<h2>Formulário de Cadastro</h2>
<form action="/cadastro" method="POST">
  <label for="name">Nome:</label><br>
  <input type="text" name="name" id="name" required><br><br>

  <label for="email">E-mail:</label><br>
  <input type="email" name="email" id="email" required><br><br>

  <button type="submit">Cadastrar</button>
</form>

<%- include('../partials/footer') %>
```

users.ejs

```
<%- include('../partials/header', { title }) %>

<h2>Usuários Cadastrados</h2>

<% if (users.length === 0) { %>
  <p>Nenhum usuário encontrado.</p>
<% } else { %>
  <ul>
    <% users.forEach(user => { %>
      <li><strong><%= user.name %></strong> - <%= user.email %></li>
    <% }) %>
  </ul>
<% } %>

<%- include('../partials/footer') %>
```

error.ejs

```
<%- include('../partials/header', { title }) %>

<h2>Ocorreu um erro</h2>
<p style="color: red;"><%= errorMessage %></p>

<a href="/">Voltar ao início</a>

<%- include('../partials/footer') %>
```

8. Crie um placeholder dinâmico para o título da página em um layout.

Arquivos

layout.ejs

```
<!DOCTYPE html>
```

```
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <title><%= title %></title>
</head>
<body>
  <header>
    <h1>Meu Site</h1>
    <nav>
      <a href="/">Início</a> |
      <a href="/cadastro">Cadastro</a> |
      <a href="/usuarios">Usuários</a>
    </nav>
    <hr>
  </header>

  <main>
    <%- body %>
  </main>

  <hr>
  <footer>
    <p>&copy; <%= new Date().getFullYear() %> - Todos os direitos reservados</p>
  </footer>
</body>
</html>
```

server.js

```
const express = require('express');
const ejs = require('ejs');
const path = require('path');
```

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

app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

app.get('/', (req, res) => {
  ejs.renderFile(
    path.join(__dirname, 'views/pages/home.ejs'),
    {},
    (err, str) => {
      res.render('layout', {
        title: 'Página Inicial',
        body: str
      });
    }
  );
});

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

9. Integre dados do MySQL em um menu dinâmico renderizado via EJS.

SQL

```
CREATE TABLE categorias (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(100) NOT NULL
);

INSERT INTO categorias (nome) VALUES
('Tecnologia'),
```

('Casa'),

('Moda'),

('Livros'),

('Esportes');

Arquivos

server.js

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

const app = express();
const PORT = 3000;

const dbConfig = {
  host: 'localhost',
  user: 'seu_usuario',
  password: 'sua_senha',
  database: 'seu_banco'
};

app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

app.get('/', async (req, res) => {
  try {
    const conn = await mysql.createConnection(dbConfig);
    const [categorias] = await conn.execute('SELECT id, nome FROM categorias');
    await conn.end();

    res.render('home', {
      title: 'Página Inicial',
```

```
    categorias
  });
 } catch (error) {
  res.status(500).send('Erro ao carregar categorias');
 }
});

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

home.ejs

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
 <meta charset="UTF-8">
 <title><%= title %></title>
</head>
<body>
 <header>
  <h1>Meu Site</h1>
  <nav>
   <ul>
    <% categorias.forEach(cat => { %>
     <li><a href="/categoria/<%= cat.id %>"><%= cat.nome %></a></li>
    <% }) %>
   </ul>
  </nav>
  <hr>
 </header>

 <main>
```

```
  <h2>Bem-vindo</h2>
  <p>Selecione uma categoria no menu.</p>
 </main>

 <footer>
  <hr>
  <p>&copy; <%= new Date().getFullYear() %> - Todos os direitos reservados</p>
 </footer>
</body>
</html>
```

home.ejs

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
 <meta charset="UTF-8">
 <title><%= title %></title>
</head>
<body>
 <header>
  <h1>Meu Site</h1>
  <nav>
   <ul>
    <% categorias.forEach(cat => { %>
     <li><a href="/categoria/<%= cat.id %>"><%= cat.nome %></a></li>
    <% }) %>
   </ul>
  </nav>
  <hr>
 </header>

 <main>
```

```
  <h2>Bem-vindo</h2>
  <p>Selecione uma categoria no menu.</p>
 </main>

 <footer>
  <hr>
  <p>&copy; <%= new Date().getFullYear() %> - Todos os direitos reservados</p>
 </footer>
</body>
</html>
```

10. Valide os campos de entrada do formulário e exiba mensagens de erro caso os dados sejam inválidos.

Arquivos:

server.js

```
const express = require('express');
const mysql = require('mysql2/promise');
const bodyParser = require('body-parser');
const path = require('path');

const app = express();
const PORT = 3000;

const dbConfig = {
 host: 'localhost',
 user: 'seu_usuario',
 password: 'sua_senha',
 database: 'seu_banco'
};

app.use(bodyParser.urlencoded({ extended: false }));
app.set('view engine', 'ejs');
```

```
app.set('views', path.join(__dirname, 'views'));
app.get('/cadastro', (req, res) => {
  res.render('form', { title: 'Cadastro', errors: [], formData: {} });
});
app.post('/cadastro', async (req, res) => {
  const { name, email } = req.body;
  const errors = [];
  if (!name || name.trim().length < 3) {
    errors.push('O nome deve ter pelo menos 3 caracteres.');
  }
  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  if (!email || !emailRegex.test(email)) {
    errors.push('Informe um e-mail válido.');
  }

  if (errors.length > 0) {
    return res.status(400).render('form', {
      title: 'Cadastro',
      errors,
      formData: { name, email }
    });
  }

  try {
    const connection = await mysql.createConnection(dbConfig);
    await connection.execute('INSERT INTO users (name, email) VALUES (?, ?)', [name, email]);
    await connection.end();
    res.redirect('/cadastro');
  } catch (error) {
    res.status(500).render('error', {
      title: 'Erro',
```

```
      errorMessage: 'Erro ao salvar no banco de dados.'
    });
  }
});

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

form.ejs

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <title><%= title %></title>
</head>
<body>
  <h2>Formulário de Cadastro</h2>

  <% if (errors.length > 0) { %>
    <ul style="color: red;">
      <% errors.forEach(erro => { %>
        <li><%= erro %></li>
      <% }) %>
    </ul>
  <% } %>

  <form action="/cadastro" method="POST">
    <label for="name">Nome:</label><br>
    <input type="text" id="name" name="name" value="<%= formData.name || '' %>"
required><br><br>
```

```
    <label for="email">E-mail:</label><br>
    <input type="email" id="email" name="email" value="<%= formData.email || '' %>"
required><br><br>

    <button type="submit">Cadastrar</button>
  </form>
</body>
</html>
```

error.ejs

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <title><%= title %></title>
</head>
<body>
  <h2>Erro</h2>
  <p style="color: red;"><%= errorMessage %></p>
  <a href="/cadastro">Voltar</a>
</body>
</html>
```