# Documentação do Projeto Front-Hackthon

## Índice



## Introdução

Esta documentação descreve a arquitetura e as funcionalidades de uma aplicação que gerencia turmas e atividades para professores. A aplicação é dividida em duas partes: uma API construída em Node.js e um front-end interativo.

## Arquitetura do Sistema

- **Back-End**: API desenvolvida em Node.js com Express e MySQL como banco de dados.
- **Front-End**: Aplicação web desenvolvida em HTML, CSS e JavaScript, utilizando Axios para realizar requisições à API.

# Documentação da API

## Endpoints

### *1. Turmas*

- **GET** /api/turma/:email
    - Descrição: Retorna todas as turmas associadas ao professor com o email fornecido.
    - Parâmetros: email (string)
- **POST** /api/turma
    - Descrição: Cria uma nova turma.
    - Corpo da requisição:

```
{
  "nome": "string",
  "periodo_letivo": "string",
  "professor_id": "integer"
}
```

- **PUT** /api/turma/:id
    - Descrição: Atualiza uma turma existente.
    - Parâmetros: id (integer)
    - Corpo da requisição:

```
{
  "nome": "string",
  "periodo_letivo": "string"
}
```

- **DELETE** /api/turma/:id
    - Descrição: Deleta uma turma existente.
    - Parâmetros: id (integer)

### *2. Professores*

- **GET** /api/prof
    - Descrição: Retorna todos os professores.
- **POST** /api/prof
    - Descrição: Cria um novo professor.
    - Corpo da requisição:

```
{
  "nome": "string",
  "senha": "string",
  "email": "string"
}
```

- **POST** /api/prof/login
    - Descrição: Realiza login de um professor.
    - Corpo da requisição:

```
{
  "email": "string",
  "senha": "string"
}
```

- **PUT** /api/prof/:id
    - Descrição: Atualiza as informações de um professor.
    - Parâmetros: id (integer)
    - Corpo da requisição:

```
{
  "senha": "string"
}
```

- **DELETE** /api/prof/:id
    - Descrição: Deleta um professor.
    - Parâmetros: id (integer)

### *3. Atividades*

- **GET** /api/atividades/:id
    - Descrição: Retorna todas as atividades de uma turma.
    - Parâmetros: id (integer)
- **POST** /api/atividades
    - Descrição: Cria uma nova atividade.
    - Corpo da requisição:

```
{
  "nome": "string",
  "descricao": "string",
  "data_entrega": "string",
  "peso_nota": "integer",
```

```
  "turma_id": "integer"
}
```

- **PUT** /api/atividades/:id
    - Descrição: Atualiza uma atividade existente.
    - Parâmetros: id (integer)
    - Corpo da requisição:

```
{
  "nome": "string",
  "descricao": "string",
  "data_entrega": "string",
  "peso_nota": "integer",
  "turma_id": "integer"
}
```

- **DELETE** /api/atividades/:id
    - Descrição: Deleta uma atividade existente.
    - Parâmetros: id (integer)

## Modelos de Dados

7. **Turma**
    - id: integer
    - nome: string
    - periodo_letivo: string
    - professor_id: integer
8. **Professor**
    - id: integer
    - nome: string
    - senha: string
    - email: string
9. **Atividade**
    - id: integer
    - nome: string
    - descricao: string
    - data_entrega: date
    - peso_nota: integer
    - turma_id: integer

# Documentação do Front-End

## Estrutura do Projeto

O front-end é desenvolvido em JavaScript, utilizando a biblioteca Axios para realizar requisições HTTP. A aplicação possui uma interface para gerenciar turmas e atividades.

## URLs da API

A URL base da API é: http://localhost:3000/api

## Funcionalidades Principais

### *1. Login de Professor*

A função login permite que um professor faça login na aplicação.

```
async function login(event) {
    event.preventDefault();

    const email = document.getElementById('username').value;
    const password = document.getElementById('password').value;

    try {
        const response = await axios.post(`${url}/prof/login`, {
            email: email,
            senha: password
        });

        if (response.status === 200) {
            localStorage.setItem('professorEmail', email);
            window.location.href = 'principal.html';
        } else {
            alert("Login falhou. Verifique suas credenciais.");
        }
    } catch (error) {
        console.error("Erro no login:", error);
        alert("Erro ao tentar fazer login. Tente novamente.");
    }
```

```
}
```

### *2. Cadastro de Professor*

A função cadastrarProfessor permite que um novo professor seja cadastrado.

```
async function cadastrarProfessor(event) {
    event.preventDefault();

    const nome = document.getElementById('inputName').value;
    const email = document.getElementById('inputEmail').value;
    const senha = document.getElementById('inputPassword').value;

    try {
        const response = await axios.post(`${url}/prof`, {
            nome: nome,
            email: email,
            senha: senha
        });

        if (response.status === 201) {
            alert("Professor cadastrado com sucesso!");
            window.location.href = 'index.html';
        } else {
            alert("Erro ao cadastrar professor. Tente novamente.");
        }
    } catch (error) {
        console.error("Erro no cadastro:", error);
        alert("Erro ao tentar cadastrar. Verifique os dados e tente novamente.");
    }
}
```

### *3. Carregar Turmas*

A função carregarTurmas busca as turmas associadas ao professor logado e as adiciona a um elemento <select>.

```
async function carregarTurmas() {
    try {
        const response = await axios.get(`${url}/turma/${email}`);
```

```
        const turmas = response.data.result;

        const selectCourse = document.getElementById('select-course');

        turmas.forEach(turma => {
            const option = document.createElement('option');
            option.value = turma.id;
            option.textContent = turma.nome;
            selectCourse.appendChild(option);
        });
    } catch (error) {
        console.error("Erro ao carregar turmas:", error);
        alert('Erro ao carregar turmas. Tente novamente.');
    }
}
```

### *4. Selecionar Turma*

A função selectTurma é chamada quando uma turma é selecionada. Ela habilita os botões de edição e deleção.

```
async function selectTurma(id) {
    selectedTurmaId = id;
    document.getElementById('edit-turma-button').disabled = false;
    document.getElementById('delete-turma-button').disabled = false;
}
```

### *5. Editar Turma*

A função editTurma permite que o professor edite uma turma existente.async function editTurma(id) {

```
    const nome = prompt("Novo nome da turma:");
    const periodo_letivo = prompt("Novo período letivo:");

    if (nome && periodo_letivo) {
        axios.put(`${url}/turma/${id}`, { nome, periodo_letivo })
            .then(response => {
                alert("Turma alterada com sucesso!");
                location.reload();
```

```
            })
            .catch(error => {
                console.error("Erro ao alterar turma:", error);
                alert("Erro ao alterar turma. Tente novamente.");
            });
    } else {
        alert("Por favor, preencha todos os campos.");
    }
}
```

### *6. Deletar Turma*

A função deleteTurma permite que o professor delete uma turma após confirmação.

```
async function deleteTurma(id) {
    if (confirm("Você tem certeza que deseja deletar esta turma?")) {
        axios.delete(`${url}/turma/${id}`)
            .then(response => {
                alert(response.data.result);
                location.reload();
            })
            .catch(error => {
                console.error("Erro ao deletar turma:", error);
                alert("Erro ao deletar turma. Tente novamente.");
            });
    }
}
```

### *7. Carregar Atividades*

A função carregarAtividades busca as atividades de uma turma específica e as exibe em uma tabela.

```
async function carregarAtividades() {
    if (!selectedTurmaId) {
        return;
    }

    try {
        const response = await axios.get(`${url}/atividades/${selectedTurmaId}`);
```

```
        const atividades = response.data.result;
        const tableBody = document.getElementById('table-body');
        tableBody.innerHTML = '';

        atividades.forEach(atividade => {
            const dataEntrega = new
Date(atividade.data_entrega).toLocaleDateString('pt-BR');
            const row = document.createElement('tr');
            row.innerHTML = `
                <td>${atividade.turma_id}</td>
                <td>${atividade.nome}</td>
                <td>${atividade.descricao}</td>
                <td>${dataEntrega}</td>
                <td>${atividade.peso_nota}</td>
                <td>
                    <button class="btn btn-warning"
onclick="editAtividade(${atividade.id})">Alterar</button>
                    <button class="btn btn-danger"
onclick="deleteAtividade(${atividade.id})">Deletar</button>
                </td>
            `;
            tableBody.appendChild(row);
        });
    } catch (error) {
        console.error("Erro ao carregar atividades:", error);
    }
}
```

### *8. Editar Atividade*

A função editAtividade permite que o professor edite uma atividade existente.

```
async function editAtividade(id) {
    const nome = prompt("Novo nome da atividade:");
    const descricao = prompt("Nova descrição da atividade:");
    const data_entrega = prompt("Nova data de entrega: AAAA-MM-DD");
    const peso_nota = prompt("Novo peso da nota:");

    if (nome && descricao && data_entrega && peso_nota) {
        axios.put(`${url}/atividades/${id}`, { nome, descricao, data_entrega, peso_nota })
```

```
            .then(response => {
                alert("Atividade alterada com sucesso!");
                location.reload();
            })
            .catch(error => {
                console.error("Erro ao alterar atividade:", error);
            });
    } else {
        alert("Por favor, preencha todos os campos.");
    }
}
```

### *9. Deletar Atividade*

A função deleteAtividade permite que o professor delete uma atividade após confirmação.

```
async function deleteAtividade(id) {
    if (confirm("Você tem certeza que deseja deletar esta atividade?")) {
        axios.delete(`${url}/atividades/${id}`)
            .then(response => {
                alert(response.data.result);
                location.reload();
            })
            .catch(error => {
                console.error("Erro ao deletar atividade:", error);
            });
    }
}
```

### *10. Pesquisa de Atividades*

O evento search-button é associado à função carregarAtividades, que carrega as atividades da turma selecionada.

```
document.getElementById('search-button').addEventListener('click', function () {
    selectedTurmaId = document.getElementById('select-course').value;
    carregarAtividades();
});
```

## Dependências

- **Axios**: Para realizar requisições HTTP. Certifique-se de instalar a biblioteca em seu projeto:

```
npm install axios
```

## Conclusão

Esta documentação fornece uma visão abrangente das funcionalidades e estrutura da aplicação. Para mais informações, consulte o código-fonte ou a documentação da API. Se houver dúvidas ou necessidades adicionais, sinta-se à vontade para perguntar!