TIC em trilhas

Módulo 04

Upload de arquivos

TIC em trilhas

Aula 1 - Dominando uploads

Conceito, estrutura, funcionamento e exemplo prático

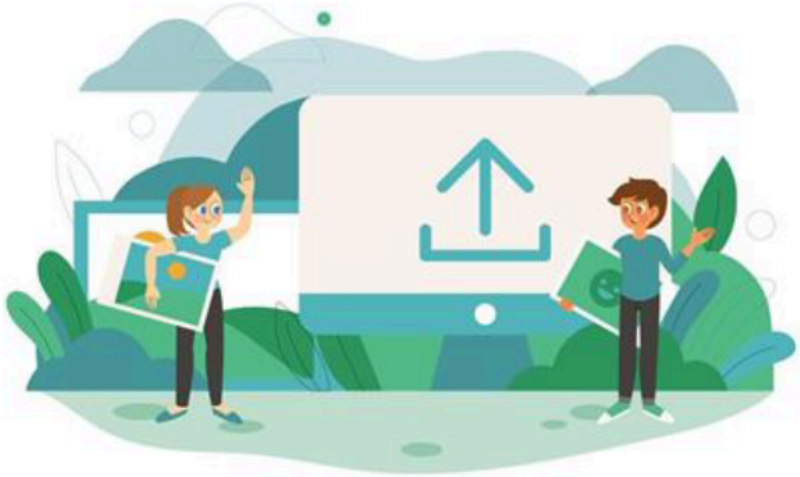

Imagem extraída através do acervo gratuito do Freepik

1.1 O que é um upload?

Um **upload** é o processo de transferir dados, como arquivos, imagens, vídeos ou qualquer tipo de informação digital, de um dispositivo local (como um computador, smartphone ou tablet) para um servidor remoto ou outro dispositivo.

Essencialmente, é o oposto de um **download**, que é quando os dados são transferidos do servidor remoto para o dispositivo local.

Imagem extraída através do acervo gratuito do Freepik

1.2 Funcionamento

Em Node.js, você pode fazer o upload de arquivos para o servidor utilizando algumas bibliotecas populares, como **express-fileupload**, **multer** ou **formidable**.

Os arquivos podem ser salvos em um banco de dados ou em repositórios locais (pastas).

Imagem extraída através do acervo gratuito do Freepik

1.3 Pré-requisitos

Para realizarmos o upload de arquivos, utilizamos as seguintes dependências no Node.js:

- Express
- Express-fileupload

Para ter acesso a documentação completa do express-file-upload, acesse o site oficial do NPM: https://www.npmjs.com/package/express-fileupload

Imagem extraída através do acervo gratuito do Freepik

1.4 Exemplo prático

- Criar rota para realizar o upload de arquivos.
- Realizar o envio de arquivos.
- Tratativa de exceções.
- Validando o arquivo enviado.

No terminal, crie uma nova pasta para o projeto e inicie o Node.js:

```
mkdir upload-example
cd upload-example
npm init -y
```

Instale o Express e o multer para lidar com uploads de arquivos:

```
npm install express multer
```

E agora vamos criar um arquivo index.js para efetuar o envio de fotos:

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

// Configurar o armazenamento com multer
const storage = multer.diskStorage({
    destination: (req, file, cb) => {
        cb(null, 'uploads/');
    },
    filename: (req, file, cb) => {
        cb(null, Date.now() + path.extname(file.originalname));
    }
});
```

```
// Filtro de validação de arquivos (somente imagens)
const fileFilter = (req, file, cb) => {
    const allowedTypes = ['image/jpeg', 'image/png', 'image/gif'];
    if (allowedTypes.includes(file.mimetype)) {
        cb(null, true);
    } else {
        cb(new Error('Tipo de arquivo inválido. Apenas imagens são permitidas.'),
    }
};
```

Do autor

TIC em trilhas

Configurar o multer e criar a nossa rota para p envio de arquivos:

```
// Configurar o multer com armazenamento e filtro
const upload = multer({
    storage: storage,
    fileFilter: fileFilter,
    limits: { fileSize: 1024 * 1024 * 5 } // Limite de 5MB
});

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

// Rota para upload de arquivos
app.post('/upload', upload.single('arquivo'), (req, res) => {
    try {
        res.send('Arquivo enviado com sucesso!');
    } catch (err) {
        res.status(500).send('Erro ao processar o upload.');
    }
});
```

Do autor

E por fim fazer a tratativa de erros e iniciar o nosso servidor:

```
// Tratativa de erros de upload
app.use((err, req, res, next) => {
    if (err instanceof multer.MulterError) {
        return res.status(400).send('Erro no upload: ' + err.message);
    } else if (err) {
        return res.status(400).send(err.message);
    }
    next();
});

// Criar pasta de uploads se não existir
const fs = require('fs');
const dir = './uploads';
if (!fs.existsSync(dir)){
    fs.mkdirSync(dir);
}

// Iniciar o servidor
app.listen(PORT, () => {
    console.log(`Servidor rodando na porta ${PORT}`);
});
```

Do autor

E para efetuar os tests criaremos um novo arquivo index.html com uma estrutura básica de html e dentro do seu body:

```
<h1>Upload de Arquivo</h1>
<form id="form-upload" enctype="multipart/form-data">
    <input type="file" name="arquivo" id="arquivo" required>
    <button type="submit">Enviar Arquivo</button>
</form>
<p id="resultado"></p>
```

E logo depois uma tag <script> contendo:

```
<script>
    document.getElementById('form-upload').addEventListener('submit', function(event) {
        event.preventDefault();
        const formData = new FormData();
        const arquivo = document.getElementById('arquivo').files[0];
        formData.append('arquivo', arquivo);
        fetch('http://localhost:3000/upload', {
            method: 'POST',
            body: formData,
        })
        .then(response => response.text())
        .then(data => {
            document.getElementById('resultado').textContent = data;
        })
        .catch(error => {
            console.error('Erro:', error);
            document.getElementById('resultado').textContent = 'Ocorreu um erro ao enviar o arquivo.';
        });
    });
</script>
```

Agora para executar o teste vamos iniciar o servidor no terminal utilizando:

```
node index.js
```

E abrir http://localhost:3000
Agora selecione um arquivo (por exemplo, uma imagem) no formulário e clique em "Enviar Arquivo". O servidor processará o arquivo e, se tudo ocorrer bem, exibirá a mensagem de sucesso.

Agora, caso ocorra algum erro como:

- **Nenhum arquivo enviado:** O multer automaticamente responde com um erro se o campo de arquivo estiver vazio.
- **Tipo de arquivo inválido:** Se o arquivo enviado não for uma imagem (JPEG, PNG, GIF), o servidor retornará um erro com a mensagem apropriada.
- **Limite de tamanho:** Se o arquivo exceder o limite de tamanho definido (5MB), o multer gerará um erro que será capturado pelo middleware de erro.

1.5 Explore mais

Quer aprender mais sobre o upload de arquivos? Existem materiais bem interessantes utilizando o express-fileupload, além de outras dependências, confira abaixo:

- NPM - Express-fileupload
- GeekForGeeks - How to upload with Formidable
- LogRocket - Multer: Easily upload files with Node.js and Express

Bons estudos

Imagem extraída através do acervo gratuito do Freepik

TIC em trilhas

Referências bibliográficas

Snyk: How to use express-fileupload. Disponível em: https://snyk.io/advisor/npm-package/express-fileupload/functions/express-fileupload. Acesso em fevereiro de 2024.

Npm: Express-fileupload. Disponível em: https://www.npmjs.com/package/express-fileupload. Acesso em fevereiro de 2024.

Scaller: Express.js File Upload. Disponível em: https://www.scaler.com/topics/expressjs-tutorial/express-file-upload. Acesso em fevereiro de 2024.

SalesforceDrillers: Express FileUpload. Disponível em: https://salesforcedrillers.com/learn-mean/express-fileupload. Acesso em fevereiro de 2024.