Universidade do Minho

Braga, Portugal

TRABALHO PRÁTICO - RELATÓRIO

I am ... (in bits and bytes)

Engenharia Web

Departamento de Informática

Engenharia Informática 2024/25

Equipa de Trabalho:

A104537 - Afonso Pedreira

A104365 - Fábio Magalhães

Junho 2025

Índice

1. Introdução

O presente relatório descreve a implementação de um sistema de "diário digital" desenvolvido no âmbito da unidade curricular de Engenharia Web. A aplicação segue o modelo OAIS (Open Archival Information System) para gestão de conteúdos digitais, implementando um repositório pessoal com funcionalidades de ingestão, armazenamento e disseminação de artefactos.

O sistema permite aos utilizadores registarem-se e criar, editar e gerir diários pessoais com suporte para ficheiros multimédia, oferecendo funcionalidades de visualização pública e privada dos conteúdos.

2. Backend

2.1. Arquitetura do Sistema

O backend da aplicação foi desenvolvido em Node.js utilizando a framework `Express.js`. A aplicação principal comunica com um serviço de dados através de chamadas HTTP. É também utilizado o módulo `morgan` para implementação e criação de um ficheiro de `logs`.

2.2. Persistência de dados

A aplicação comunica com uma base de dados `MongoDB` através do módulo `mongoose` o que permite guardar informação sobre os modelos definidos. Isto é, permite criar tipos de dados e guarda-los para utilização pela API.

2.3. Modelos

2.3.1. File (ficheiro)

```
const fileSchema = new mongoose.Schema({
    name: String,
    originalname: String,
    mimetype: String,
    date: Date,
    size: Number,
    path: String,
    journal: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'Journal',
        required: true
    },
    isPublic: {
        type: Boolean,
        default: false
    }
}, { versionKey: false });
```

O modelo `File` representa os ficheiros associados aos diários dos utilizadores. Cada instância deste modelo contém metadados sobre o ficheiro carregado, tais como o nome original, o tipo MIME, o tamanho, a data e o caminho relativo onde está armazenado no servidor. Cada ficheiro está ligado a um diário (`journal`), garantindo rastreabilidade e organização. O campo `isPublic` define se o ficheiro é acessível publicamente ou não.

Campos principais:

- name: Nome do ficheiro gerado no servidor.
- originalname: Nome original do ficheiro enviado pelo utilizador.
- mimetype: Tipo de conteúdo (ex: image/png, application/pdf).
- date: Data do envio do ficheiro (a preencher no momento do upload).
- size: Tamanho do ficheiro em bytes.
- path: Caminho relativo para o ficheiro no sistema de ficheiros.
- journal: Referência ao diário a que o ficheiro pertence.
- isPublic: Indica se o ficheiro é público ou privado.

2.3.2. User (utilizador)

```
const userSchema = new mongoose.Schema({
    username: String,
    email: String,
    password: String,
}, { versionKey: false });
```

O modelo `User` representa os utilizadores da aplicação. Cada utilizador tem um username, um email (único para autenticação) e uma password encriptada. Este modelo serve como base para o sistema de autenticação, gestão de sessão e associação com os diários criados.

Campos principais:

- username: Nome visível do utilizador.
- email: Endereço de e-mail único utilizado para login.
- password: Palavra-passe armazenada de forma segura (encriptada com o módulo `crypto` e uma hash `SHA-256`).

2.3.3. Journal (diário)

```
const journalSchema = new mongoose.Schema({
    title: {
        type: String,
        required: true,
        trim: true
    },
    content: String,
    files: [{
        type: mongoose.Schema.Types.ObjectId,
        ref: 'File'
    }],
    author: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'User',
        required: true
    },
    authorName: {
        type: String,
        required: true
    },
    views: {
        type: Number,
        default: 0
    },
    downloads: {
        type: Number,
        default: 0
    },
    createdAt: {
        type: Date,
        default: Date.now
    },
    updatedAt: {
```

```
        type: Date,
        default: Date.now
    },
}, { versionKey: false });
```

O modelo `Journal` representa uma entrada de diário criada por um utilizador. Contém o título e o conteúdo em formato texto (`Markdown`), além de uma lista de ficheiros associados. Está diretamente relacionado com o autor e inclui campos para estatísticas de visualização e download. Os campos `createdAt` e `updatedAt` registam as datas de criação e modificação, respetivamente.

Campos principais:

- title: Título do diário (obrigatório e formatado).
- content: Corpo do diário (texto em `Markdown`).
- files: Lista de ficheiros (`File`) associados ao diário.
- author: Referência ao utilizador (`User`) que criou o diário.
- authorName: Nome do autor no momento da criação.
- views: Número de vezes que o diário foi visualizado.
- downloads: Total de downloads de ficheiros associados.
- createdAt: Data de criação do diário.
- updatedAt: Data da última modificação.

2.4. Rotas da API

Para o backend foram implementadas 3 rotas principais:

2.4.1. `/files`

Expoem as funcionalidades relacionadas com ficheiros, como upload, download e listagem dentro do repositório pessoal do utilizador.

- `GET /files`: Lista todos os ficheiros do sistema.
- `GET /files/:id`: Lista um ficheiro específico.
- `GET /files/journal/:id`: Lista os ficheiros associados a um diário específico.
- `POST /files/create`: Permite o upload de um novo ficheiro.
- `DELETE /files/:id`: Elimina um ficheiro específico.
- `PUT /files/:id`: Atualiza as informações de um ficheiro específico.
- `POST /files/:id/toggle-visibility`: Altera a visibilidade de um ficheiro específico.

2.4.2. `/journals`

Gerencia os diários pessoais, permitindo a criação, edição e listagem dos mesmos.

- `GET /journals`: Lista todos os diários do sistema.
- `GET /journals?author=autor`: Lista todos os diários de um dado utilizador.
- `GET /journals/:id`: Lista um diário específico.
- `POST /journals/create`: Cria um novo diário.
- `DELETE /journals/:id`: Elimina um diário específico.
- `PUT /journals/:id`: Atualiza as informações de um diário específico.
- `POST /journals/:id/incViews`: Incrementa o contador de visualizações de um diário específico.

- `POST /journals/:id/incDownloads`: Incrementa o contador de downloads de um diário específico.

2.4.3. `/users`

Gerencia os utilizadores, permitindo o registo, autenticação e gestão de perfis.

- `GET /users`: Lista todos os utilizadores do sistema.
- `GET /users/:id`: Lista um utilizador específico.
- `POST /users/verify`: Verifica as credenciais de um utilizador.
- `POST /users/create`: Regista um novo utilizador.
- `POST /users/change-password`: Altera a palavra-passe de um utilizador.
- `POST /users/change-username`: Altera a palavra-passe de um utilizador.
- `POST /users/delete`: Elimina um utilizador específico.

3. Frontend

3.1. Arquitetura do Sistema

O frontend da aplicação foi desenvolvido em Node.js utilizando a framework `Express.js`. A aplicação utiliza o módulo `axios` para comunicação HTTP com o servidor backend.

3.1.1. Tecnologias Utilizadas

O frontend utiliza uma stack moderna baseada em:

- Engine: EJS (Embedded JavaScript)
- CSS: Bootstrap
- JavaScript: Vanilla JS com bibliotecas especializadas

3.2. Páginas implementadas

3.2.1. Página de Login

Aqui, os utilizadores registados podem autenticar-se com o seu e-mail e palavra-passe. Para novos utilizadores, está disponível uma ligação para a página de registo (signup), onde podem criar uma nova conta. O processo fornece mensagens de erro ou sucesso para ajudar o utilizador em caso de falhas na autenticação.

3.2.2. Página do Utilizador

A página do utilizador permite ao utilizador autenticado gerir as suas informações pessoais. Nesta secção, é possível alterar o nome de utilizador e redefinir a palavra-passe. Adicionalmente, o utilizador pode optar por apagar permanentemente a sua conta, uma ação irreversível que remove todos os seus dados de login.

3.2.3. Página Inicial

A página inicial apresenta uma listagem pública de todos os diários disponíveis. Estes diários são visíveis a todos os utilizadores, autenticados, permitindo a exploração e leitura de conteúdos partilhados (Consumidores). A listagem inclui título, autor e data de criação, mantendo uma linhagem temporal do conteúdo.

3.2.4. Página dos "Meu Diários"

Esta página está disponível apenas para o próprio utilizadore autenticado (Produtor e Manager) e mostra todos os diários que este criou. Para cada diário, são apresentadas informações como o número de visualizações, o número de downloads e opções para editar o seu conteúdo e metainformação. Esta secção oferece um painel de controlo pessoal completo para os autores gerirem o seu conteúdo.

3.2.5. Pagina de Edição/Publicação

A página de edição/publicação permite ao utilizador editar ou criar um diário. Aqui, o utilizador pode alterar o título, substituir ou adicionar novos ficheiros anexos e modificar o conteúdo em `Markdown`. Após as alterações, o utilizador pode publicar o diário, tornando-o visível na página inicial. Este também é responsável por gerir os ficheiros anexados bem como a sua visualização e possivel download por outros utilizadores.

4. Modelo OAIS

No modelo `OAIS` os módulos `jszip` e `xml2js` permitiram a transcrição da informação da base de dados retornada pela API/backend em formato ZIP e a sua posterior ingestão.

4.1. Atores do Sistema

O sistema implementa os principais atores do modelo OAIS:

- Produtor (Producer)

  Função: Utilizadores que criam conteúdo

  Responsabilidades: Criação de diários e upload de ficheiros

  Implementação: Interface de criação e edição de diários com suporte multimédia

- Gestor (Manager)

  Função: Próprio utilizador numa segunda função

  Responsabilidades: Controlo de visibilidade, organização do repositório pessoal

  Implementação: Painéis de gestão, controlo de privacidade dos ficheiros

- Consumidor (Consumer)

  Função: Utilizadores que acedem ao conteúdo público

  Responsabilidades: Visualização, pesquisa e download de conteúdos

  Implementação: Páginas públicas de visualização e pesquisa

5. Conclusão

O sistema de "diário digital" desenvolvido demonstra a aplicação prática dos princípios de engenharia web na criação de uma plataforma funcional para a gestão e partilha de conteúdos pessoais. Ao adotar a estrutura do modelo OAIS, a aplicação cumpre

eficazmente as fases de ingestão, armazenamento e disseminação, permitindo aos utilizadores não só criar e gerir os seus diários, mas também partilhar os seus conteúdos com controlo sobre a visibilidade e o acesso.

Através da integração de autenticação de utilizadores, manipulação de ficheiros, estatísticas de visualização e edição de conteúdos em Markdown, a aplicação proporciona uma experiência rica e coerente. A implementação modular e baseada em padrões da indústria (com a utilização de uma REST API separada) assegura a escalabilidade e manutenção futura do sistema.

Este projeto serviu ainda como uma oportunidade de consolidar competências em desenvolvimento web com Node.js, Express e MongoDB, aliando aspetos técnicos a preocupações de usabilidade, segurança e organização e persistência de dados.

6. Bibliografia

[1] Ramalho, José Carlos. «Diapositivos EWEB2025». Disponível em: https://epl.di.uminho.pt/~jcr/AULAS/EngWeb2025

[2] «ExpressJS Guides». Disponível em: https://expressjs.com/en/guide/routing.html

[3] «Docker Docs». Disponível em: https://docs.docker.com

[4] «Bootsrap Docs». Disponível em: https://getbootstrap.com/docs