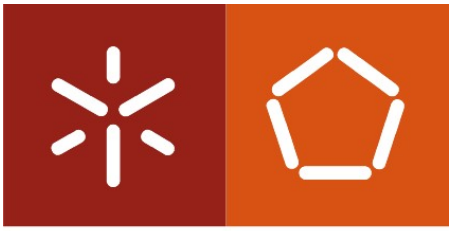

Universidade do Minho

Repositório de Recursos Didáticos (RRD)

Representação e Processamento de Conhecimento na Web

Carolina Vila Chã
pg47100@alunos.uminho.pt

Sofia Santos
a89615@alunos.uminho.pt

Universidade do Minho
Departamento de Informática
Mestrado em Engenharia Informática

19 de junho de 2022

Resumo

Este documento descreve o desenvolvimento de uma aplicação Web que implementa e dá suporte a um repositório de recursos didáticos, com as seguintes caraterísticas: autenticação de utilizadores, informação pública e privada, possibilidade de comentar os recursos disponíveis, notificações, entre outras.

Conteúdo

0.1 API-Server 5
0.1.1 Rotas 7
0.1.2 Autenticação 7
0.1.3 Logging 7
0.2 App-Server 8
0.2.1 Autenticação 8
0.2.2 Recursos 8
0.2.3 Notificações 9
0.2.4 Administração 9
0.3 Instalação 10
0.3.1 Instalação Manual 10
0.3.2 Instalação Automática 10
0.4 Exemplos 10

Lista de Figuras

1 Modelo OAIS 3
2 Caption 5
3 (Esquerda para a direita, cima para abaixo) Página Inicial, Criação de uma Conta, Login, Gestão de Conta 11
4 (Esquerda para a direita, cima para abaixo) Adicionar Recurso, Explorar Recursos, Página de Recurso, Visualização de Ficheiro 11
5 (Esquerda para a direita, cima para abaixo) Página de Boas Vindas, Gestão de Utilizadores, Painel de Admin, Criação de Notificação 12

Descrição do Problema

No dia a dia de um professor a necessidade de publicar conteúdos de forma organizada para os seus alunos é uma necessidade constante.

O objetivo deste projeto é criar um arquivo seguro e com controlo de acesso aos recursos didáticos, permitindo acesso direto a qualquer recurso em qualquer momento e lugar (desde que haja internet).

Para além deste serviço de acesso aos recursos, pretende-se que sejam implementados serviços adicionais: gestão de notícias, gestão dos utilizadores, gestão dos recursos que se vão adicionando ao repositório, possibilidade dos utilizadores comentarem e classificarem os recursos disponíveis.

É possível consultar o projeto na sua totalidade no repositório[1].

O projeto deve, por isso seguir o modelo OAIS[2]:

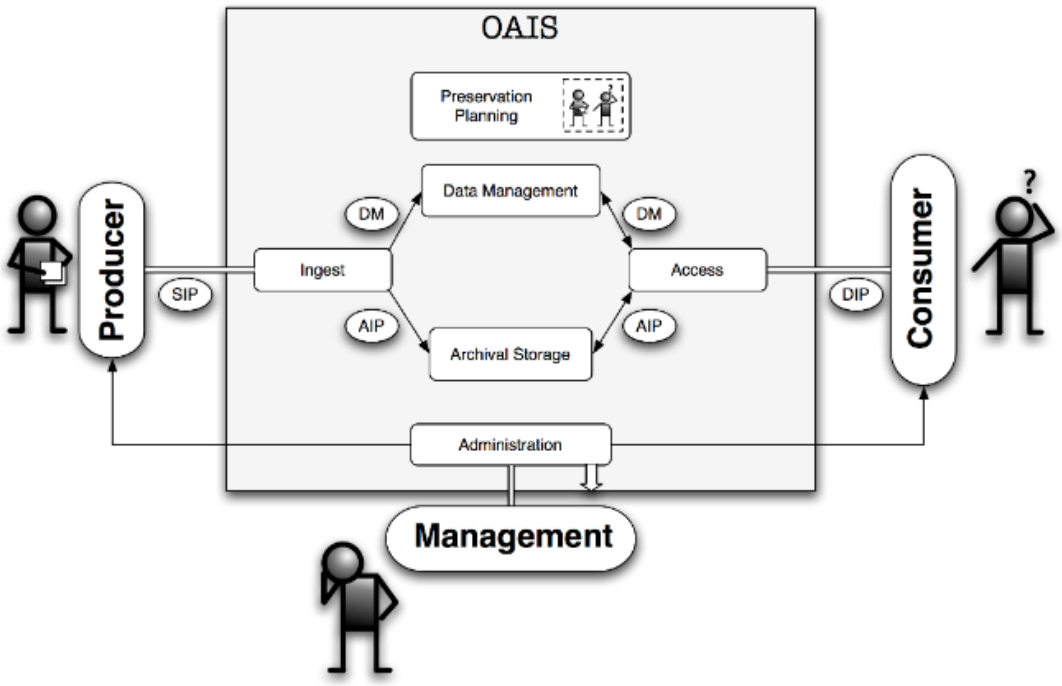

Figura 1: Modelo OAIS

Devem existir três tipos de atores:

1. Produtor - deposita recursos no repositório;
2. Administrador - acesso a todas as operações, papel de manutenção;
3. Consumidor - consulta, pesquisa, e descarrega informação.

Devem também existir três megaprocessos:

1. Ingestão - receção de materiais a arquivar;
2. Administração - gestão interna do sistema (utilizadores, arquivos, etc);
3. Disseminação - fornece a informação arquivada.

O sistema também utilizará três pacotes de informação diferentes:

1. SIP[3] - Submissão
2. AIP[4] - Arquivo
3. DIP[5] - Disseminação

Decisões e Implementação

0.1 API-Server

O servidor correspondente à API de dados foi desenvolvido em NodeJS, com recurso à *framework* Express. Os dados relativos aos recursos recebidos pela API, em pacotes SIP, são armazenados numa base de dados do MongoDB, enquanto que os recursos em si são distribuídos por um conjunto de diretorias, de acordo com o produtor dos mesmos e o seu tipo. Por outras palavras, os mesmos são convertidos em pacotes AIP, armazenados de forma mista.

Quando um consumidor pretende aceder aos dados armazenados na API, esta cria pacotes DIP, que podem conter apenas um ficheiro ou o conjunto de ficheiros que constituem um recurso, em formato ZIP.

Um pacote SIP possui a seguinte estrutura:

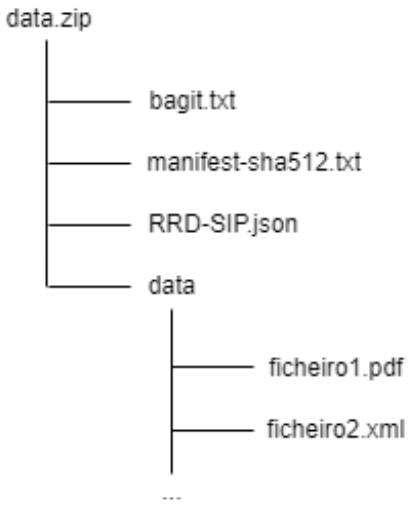

Figura 2: Caption

Os ficheiros `bagit.txt` e `manifest-sha512.txt` são ficheiros auxiliares, indicam ao sistema que os ficheiros seguem o sistema hierárquico *BagIt*, sendo que o segundo ficheiro contém o nome de todos os ficheiros de dados presentes no pacote e a sua *checksum*, neste caso no formato `SHA512`, que permite ao sistema validar a integridade dos ficheiros. A diretoria `data` contém todos os ficheiros que se pretende armazenar no sistema, neste caso um ficheiro PDF e um ficheiro XML. É possível armazenar qualquer tipo de ficheiro no sistema. Por último, o ficheiro `RRD-SIP.json` contém a informação sobre o pacote que deverá ser armazenada na base de dados, como o título do recurso, a sua descrição ou o ID do produtor do mesmo.

```
{
    "title": "Recurso",
    "description": "Recurso sobre um tema interessante.",
    "type": "recurso",
    "semester": 0,
    "academicYearStart": 2021,
    "authors": [
        "JCR"
    ],
    "submitter": "[ID do produtor]",
    "files": [
        {
            "name": "ficheiro1.pdf",
            "path": "data/ficheiro1.pdf",
            "size": "tamanho do ficheiro em bytes (inteiro)",
            "mimetype": "application/pdf",
            "hash": "[hash do ficheiro]"
        },
```

```
        {
            "name": "ficheiro2.xml",
            "path": "data/ficheiro2.xml",
            "size": "tamanho do ficheiro em bytes (inteiro)",
            "mimetype": "text/xml",
            "hash": "[hash do ficheiro]"
        }
    ]
}
```

Listing 1: Exemplo de um ficheiro `RRD-SIP.json`

Quando a API de dados recebe o SIP, esta começa por abrir o ficheiro ZIP e extrair o ficheiro `RRD-SIP.json`. Depois, para cada ficheiro descrito no mesmo, irá verificar se as *hashes* correspondem, isto é, se os ficheiros estão intactos. Se o ficheiro for do tipo XML, irá ainda verificar se o mesmo está de acordo com o *schema* disponibilizado pelo professor. Em caso afirmativo, os ficheiros são armazenados na diretoria `data`, localizada ao nível da API de dados. Dentro desta diretoria, existem subdiretorias correspondentes a cada produtor e dentro das mesmas subdiretorias correspondentes a cada tipo de recurso. Neste caso, os ficheiros seriam então armazenados em `./data/[ID do produtor]/[tipo do recurso]/`.

O resto da informação do ficheiro `RRD-SIP.json` é armazenada na base de dados, de acordo com o seguinte *schema*:

```
const ResourceSchema = new mongoose.Schema({
    title : String,
    description : String,
    semester : Number, // 1, 2 ou 0 se for anual
    academicYearStart : Number, // 2021 no caso do ano letivo 2021/22
    dateUploaded : Date,
    authors : [String],
    submittedBy : String, // id da pessoa que fez a submissao
    resourceType : String, // e.g.: teste/exame, slides, manual, relatorio, tese,
    etc.
    files : [{
        name : String,
        path : String,
        size : Number,
        mimetype : String,
        hash : String
    }],
    comments : [{
        author : String, // id da pessoa que comentou
        text : String,
        timestamp: Date
    }],
    reviews : [{
        _id: false,
        author : String,
        value : Number
    }],
    views : Number,
    downloads : Number
})
```

Aqui, `semester` e `academicYearStart` dizem respeito ao semestre e ano letivo no qual o recurso foi produzido. Consideramos que esta informação é mais relevante do que uma data específica, visto que muitas vezes os recursos como fichas ou slides não têm uma data específica de produção.

Para além da informação que consta do `RRD-SIP.json`, temos aqui também uma lista de comentários, *reviews* e contadores de visualizações e transferências. Estes campos são inicializados com valores nulos e serão posteriormente preenchidos através do uso do servidor da aplicação.

Por último, o campo `path` de cada ficheiro é atualizado para refletir o novo caminho do mesmo dentro do *filesystem* da API de dados.

Usamos o módulo `JSZip` para leitura e escrita de ficheiros ZIP, tanto na API de dados como no servidor da aplicação. Usamos também o módulo `libxmljs2` para a validação dos ficheiros

XML.

0.1.1 Rotas

A tabela abaixo contém todas as rotas presentes na API de dados e uma breve descrição das mesmas.

Tipo	Rota	Descrição
GET	/recursos	Devolve um array em JSON com a informação dos recursos ordenados por ordem de carregamento dos mesmos, dos mais recentes para os mais antigos. Parâmetros opcionais (podem ser combinados): ?tipo=X ->devolve apenas os recursos do dado tipo. ?q=pal ->devolve apenas os recursos com "pal"no seu título. ?account=ID ->devolve apenas os recursos carregados pela conta com id ID. Parâmetro opcional (não pode ser combinado): ?recent=N ->devolve apenas os N recursos mais recentes.
POST	/recursos	Regista um recurso no repositório.
GET	/recursos/top?n=N	Devolve um array em JSON com os N recursos mais consultados, por ordem decrescente. Uma consulta equivale a uma visualização ou a uma transferência.
GET	/recursos/:rid	Devolve a informação em JSON relativa ao recurso com id rid.
PUT	/recursos/:rid	Altera a informação do recurso com id rid, de acordo com o pacote SIP fornecido.
DELETE	/recursos/:rid	Elimina o recurso com id rid e os ficheiros associados ao mesmo.
GET	/recursos/:rid/zip	Devolve um ficheiro ZIP com todos os ficheiros relativos ao recurso com id rid.
POST	/recursos/:rid/score	Adiciona uma classificação ao recurso com id rid.
POST	/recursos/:rid/comment	Adiciona um comentário ao recurso com id rid.
DELETE	/recursos/:rid/comment/:cid	Elimina o comentário com id cid do recurso com id rid.
GET	/recursos/:rid/:fid	Devolve a informação em JSON relativa ao ficheiro com id fid do recurso com id rid.
GET	/recursos/:rid/:fid/file	Devolve o ficheiro com id fid do recurso com id rid.

0.1.2 Autenticação

Todos os pedidos para a API de dados requerem um token, que deve ser fornecido na *query string* (e.g.: `GET /api/recursos?token=[TOKEN]`). Este token contém informação sobre o utilizador que fez o pedido e o seu nível de autorização. Apenas os autores de um recurso e os administradores do sistema têm permissão para realizar operações de edição ou remoção na API. Iremos falar em mais detalhe sobre os níveis de autorização na secção relativa ao servidor da aplicação.

0.1.3 Logging

Para além de ser apresentada na linha de comandos, a informação relativa aos pedidos realizados à API de dados podem ser consultados num ficheiro criado pelo servidor. Este ficheiro está disponível para consulta pelos administradores do sistema através da rota `GET /api/log`. A rota `GET /api/log?n=N` devolve apenas as N linhas mais recentes do ficheiro, em formato JSON.

0.2 App-Server

O servidor da aplicação foi também desenvolvido em NodeJS, com recurso à framework `Express` e ao `Tailwind` para o CSS.

0.2.1 Autenticação

Esta aplicação utiliza o `passport-local-mongoose` para a autenticação dos utilizadores. Todos os utilizadores da aplicação devem ter uma conta. Esta decisão foi feita pois o enunciado do projeto não especificava este detalhe. Contudo, como qualquer pessoa pode criar uma conta, a informação dos recursos pode ser considerada pública. Para iniciar sessão, os utilizadores devem fornecer o seu endereço de email e palavra-passe.

Quando um utilizador cria uma conta, esta tem o nível de autorização mais baixa (Consumidor). Um administrador pode, contudo, elevar as permissões de cada conta para Produtor ou para Administrador. A tabela seguinte contém um resumo de cada um destes níveis.

Consumidor	É capaz de consultar os recursos existentes e de transferir, comentar e classificar recursos. Consegue também visualizar notificações.
Produtor	É capaz de criar novos recursos, editar e remover os mesmos, para além de todas as permissões do Consumidor.
Administrador	Possui todas as permissões de Produtor, consegue editar e remover qualquer recurso e tem ainda acesso a um menu de administração no qual consegue gerir as contas, consultar o log e gerir notificações.

Para além disso, cada utilizador possui um token único, criado no momento do registo da conta. Com este token, é capaz de realizar pedidos à API, tal como foi mencionado acima. O utilizador pode a qualquer momento consultar o seu token nas definições da sua conta e gerar um novo token, no caso do seu token atual ter sido comprometido.

Nas definições, os utilizadores conseguem ainda editar os seus dados pessoais.

0.2.2 Recursos

Qualquer utilizador é capaz de consultar a lista de recursos existentes no repositório. É ainda possível filtrar os recursos por tipo ou pesquisar por um certo termo. Cada recurso apresenta a informação presente na base de dados acerca do mesmo, para além dos ficheiros existentes. Ao clicar num ficheiro, é possível pré-visualizar o mesmo, se for do tipo PDF, XML, imagem, áudio ou vídeo, ou transferi-lo. Os utilizadores podem ainda comentar num recurso e classificá-lo num sistema de cinco estrelas, tendo acesso à lista de comentários, à classificação média e ao total de visualizações e transferências de cada recurso. Existe ainda a opção de transferir o conjunto de ficheiros que constituem o recurso dentro de um ficheiro ZIP.

Os produtores e administradores conseguem adicionar recursos, através de um formulário destinado a esse fim. No mesmo, podem definir a informação acerca do mesmo apresentada acima, como o título e a descrição, e carregar um ou mais ficheiros para fazerem parte do mesmo. Depois de preenchido o formulário, o servidor fica encarregue de gerar o pacote SIP, tal como mencionado acima.

Os recursos podem ainda ser editados ou removidos pelo produtor dos mesmos ou por um qualquer administrador. A edição permite, para além de modificar a informação do recurso, remover ou adicionar novos ficheiros sem ser necessário criar um novo recurso. É também possível aos administradores apagar comentários nos recursos, tal como ao autor dos mesmos.

Ao clicar no nome de utilizador do produtor, os utilizadores são levados para uma página na qual conseguem ver todos os recursos carregados pelo dado produtor.

0.2.3 Notificações

Quando um utilizador inicia sessão, é levado para uma página inicial, na qual consegue ver as notificações mais recentes, tendo também um botão que permite consultar a lista total de notificações. Apenas os administradores conseguem adicionar e/ou remover notificações. Estas não podem ser editadas, não por não termos considerado essa funcionalidade, mas porque consideramos que não faria sentido, num sistema real, ser possível editar notificações, tal como não é possível editar emails. Faz mais sentido enviar uma notificação nova com nova informação.

0.2.4 Administração

Os administradores têm acesso a um painel de administração, no qual têm acesso à lista total de utilizadores do sistema, podendo eliminar contas ou editar o nível de autorização das mesmas. Existe também uma página na qual é possível ver quais os recursos mais consultados e ainda um *log* de todas as operações na API de dados.

Utilização e Testes

0.3 Instalação

Existem duas formas de instalar os servidores.

0.3.1 Instalação Manual

Primeiro, é necessário transferir todos os ficheiros relativos aos servidores. Depois, dentro da pasta de cada um, deve correr-se o comando `npm install` para instalar todas as dependências. Para além disso, é necessário ter o MongoDB instalado e a correr na porta 27017 (já corre nesta porta por defeito).

Após este passo, devem-se abrir dois terminais.

No primeiro, na pasta do servidor API, corre-se o comando `npm start`. Isto irá iniciar a API de dados na porta 8000.

No segundo terminal, corre-se primeiro o comando `npm run seed` para carregar para a base de dados um utilizador com email e password iguais a `admin` e com permissões de administrador. Caso contrário, não haveria possibilidade de criar contas de administrador no sistema, visto que todas as contas criadas no servidor são por defeito de consumidores. Depois, deve-se correr o comando `npm run build:css` para compilar os ficheiros CSS. Como o servidor da aplicação da API usa uma framework para o CSS, este passo é fundamental para o CSS funcionar corretamente. Por último, usamos `npm start` para correr o servidor da aplicação na porta 3000.

0.3.2 Instalação Automática

Para a instalação automática, basta ter o Docker instalado, transferir o ficheiro `docker-compose.yml` e correr o comando `docker-compose up`. Desta forma, a API de dados irá correr na porta 8000 e o servidor da aplicação na porta 3000.

0.4 Exemplos

Seguem-se algumas imagens que ilustram a utilização do repositório.

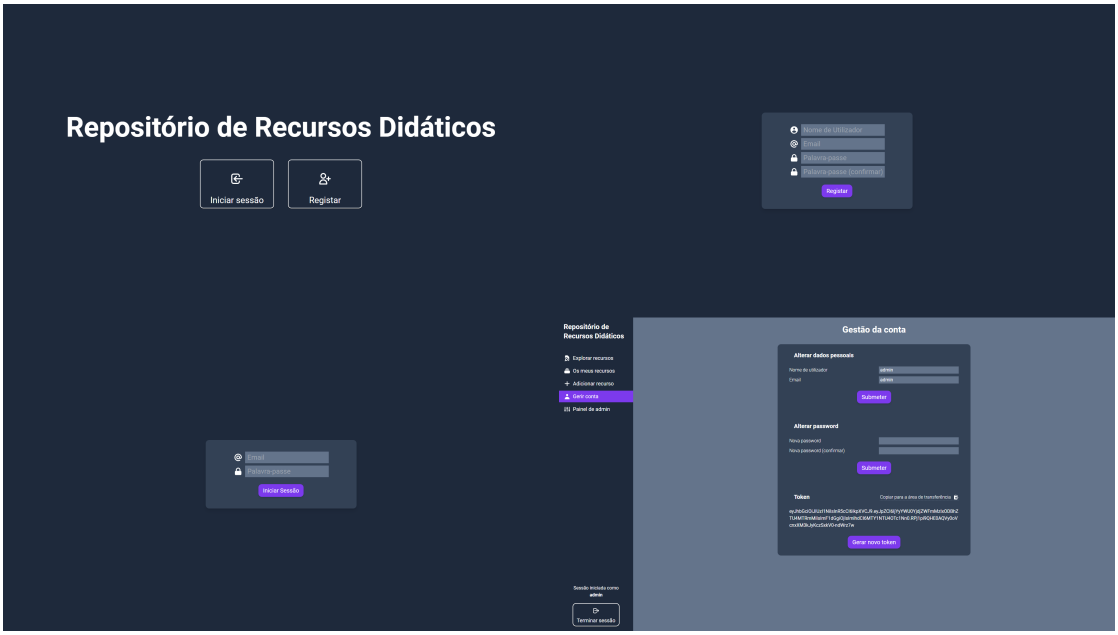

Figura 3: (Esquerda para a direita, cima para abaixo) Página Inicial, Criação de uma Conta, Login, Gestão de Conta

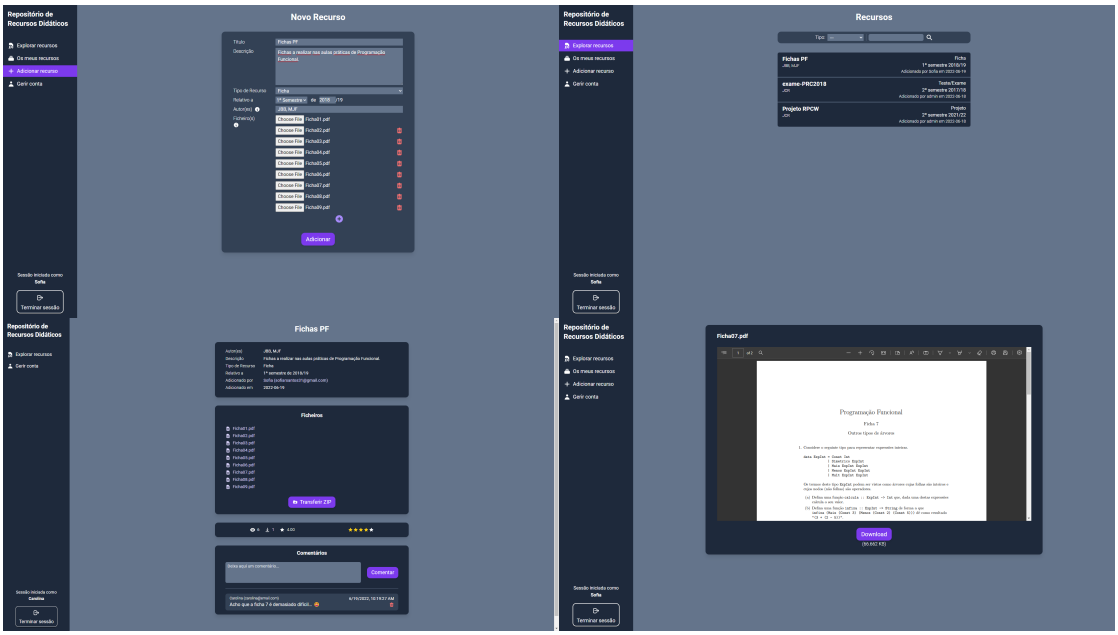

Figura 4: (Esquerda para a direita, cima para abaixo) Adicionar Recurso, Explorar Recursos, Página de Recurso, Visualização de Ficheiro

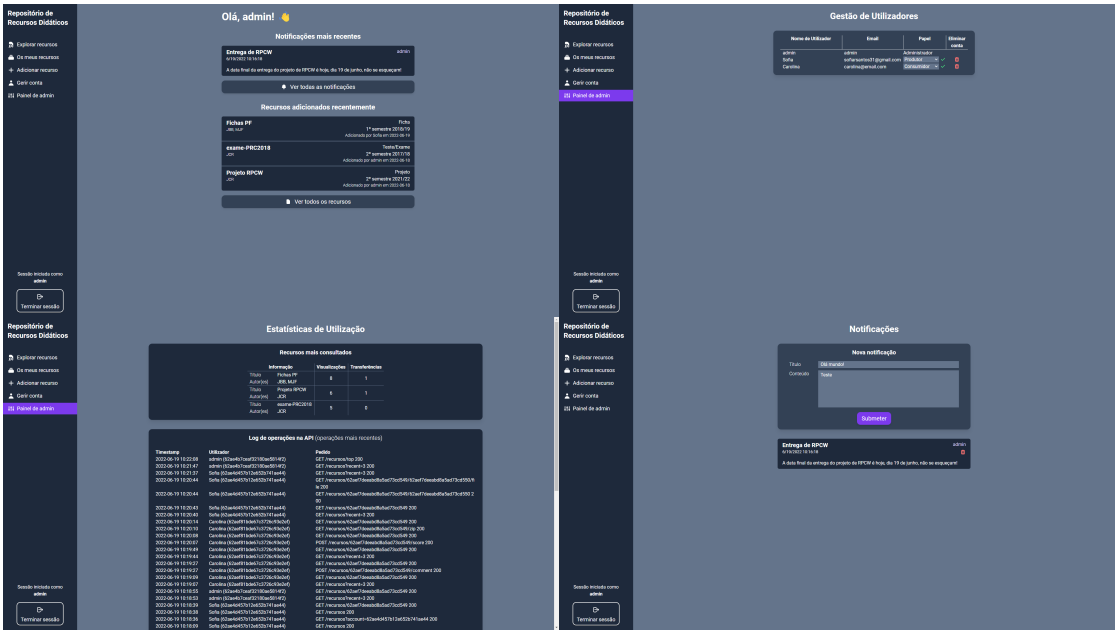

Figura 5: (Esquerda para a direita, cima para abaixo) Página de Boas Vindas, Gestão de Utilizadores, Painel de Admin, Criação de Notificação

Conclusão

Em suma, este documento descreve a nossa resolução face ao problema proposto, assim como os mecanismos por detrás dos dois servidores e da aplicação em geral, não só o código *backend*, como a sua parte *frontend*.

Consideramos que o projeto realizado está num estado satisfatório, e que nos permitiu consolidar os conhecimentos adquiridos ao longo do semestre, não só sobre o desenvolvimento de uma aplicação, mas também na sua configuração inicial, e também no seu *deployment*.

Como trabalho futuro, consideramos que poderíamos adicionar mais funcionalidades ao projeto; pequenos detalhes para "qualidade de vida", como a possibilidade de ter uma foto de perfil, ou de poder mudar o tema da aplicação para outras cores (por exemplo, para um tema mais claro, ou para um tema mais escuro); poderíamos também adicionar funcionalidades maiores, tais como enviar um email aos utilizadores quando surgem novas notificações, ou permitir resposta a comentários.

Referências

[1] Repositório do projeto - https://github.com/carolinavc99/Projeto.

[2] Documentação sobre o modelo de referência OAIS - http://www.oais.info/, último acesso em 17/06/2022.

[3] Submission Information Package (SIP) - https://www.iasa-web.org/tc04/submission-information-package-sip, último acesso em 17/06/2022.

[4] Archival Information Package (AIP) - https://www.iasa-web.org/tc04/archival-information-package-aip, último acesso em 17/06/2022.

[5] Dissemination Information Package (DIP) - https://www.iasa-web.org/tc04/formats-and-dissemination-information-packages-dip, último acesso em 17/06/2022.