

UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA

PROCESSAMENTO E REPRESENTAÇÃO DE INFORMAÇÃO

---

# Resource Finder

---

Etienne Costa (a76089)  
Maurício Salgado (a71407 )

7 de fevereiro de 2021

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>4</b>
<b>2</b>	<b>Análise e Especificação</b>	<b>5</b>
2.1	Descrição do Projecto . . . . .	5
2.2	Especificação de Requisitos . . . . .	5
<b>3</b>	<b>Funcionamento e Arquitetura do Software</b>	<b>7</b>
3.1	Funcionamento . . . . .	7
3.2	Arquitetura do Software . . . . .	9
3.2.1	Modelo de dados . . . . .	9
3.2.2	Utilizadores . . . . .	10
3.2.3	Recursos . . . . .	10
3.2.4	Categorias . . . . .	12
3.2.5	Notícias . . . . .	12
3.3	Autenticação . . . . .	13
3.3.1	Rotas com autenticação . . . . .	13
<b>4</b>	<b>Resultados Obtidos</b>	<b>14</b>
4.1	Homepage . . . . .	14
4.2	Registar Utilizador . . . . .	15
4.3	Login . . . . .	16
4.4	Dashboard . . . . .	17
4.5	Registar recurso . . . . .	18
4.6	Estatísticas do Sistema . . . . .	19
4.7	Recursos . . . . .	20
4.8	Recurso . . . . .	21
4.9	Comentários . . . . .	22



## Lista de Figuras

1	<i>Open Archival Information System</i> . . . . .	7
2	<i>Arquitetura do Software</i> . . . . .	9
3	<i>Autenticação</i> . . . . .	13
4	Homepage . . . . .	14
5	Registar Utilizador . . . . .	15
6	Login de um utilizador . . . . .	16
7	Dashboard com o feed e reports . . . . .	17
8	Registar recurso . . . . .	18
9	Estatísticas do sistema . . . . .	19
10	Lista de recursos . . . . .	20
11	Recurso . . . . .	21
12	Comentários de um recurso . . . . .	22

# 1 Introdução

O presente relatório é referente ao trabalho prático desenvolvido na Unidade Curricular **Processamento e Representação de Informação**, do 4º ano do Mestrado Integrado de Engenharia Informática, da Universidade do Minho.

É pretendido com este trabalho prático desenvolver uma plataforma de gestão e disponibilização de recursos educativos tirando partido do *Express*, que é uma framework para aplicações web.

Numa primeira fase, será feita uma breve introdução dos requisitos à alcançar com a realização do projecto.

De seguida, é feita uma explicação mais detalhada do funcionamento e da arquitetura de todo o sistema, esmiuçando as respectivas componentes de ambos bem como os seus propósitos.

Por fim, será feita uma representação dos resultados obtidos, e um conclusão relatando o processo de desenvolvimento.

## 2 Análise e Especificação

### 2.1 Descrição do Projecto

O projecto em questão consistiu no desenvolvimento de uma aplicação web para realizar a gestão e disponibilização de recursos educativos, sendo que existem 4 entidades fundamentais que determinam o fluxo de informação que circula no sistema. Para tal implementação, procurou-se seguir à risca o *Open Archival Information System* (OAIS) que é um modelo de referência para a preservação de longo prazo de activos digitais.

### 2.2 Especificação de Requisitos

O levantamento de requisitos é considerada a etapa mais importante, pois é nela que se priorizam as necessidades dos futuros usuários do software, necessidades essas denominadas como *requisitos*. Os principais requisitos levantados nesta etapa foram os seguintes :

Requisitos de Descrição:

- Utilizadores: ID, Nome, Email, Nível de acesso, Data de registo, Data do último acesso, Password, Relatório de submissão
- Recursos :ID, Tipo, Título, Subtítulo, Data de criação, Data de registo, Visibilidade, Autor, Comentários, Downloads, Likes.
- Categorias: ID, Nome.

Requisitos de Exploração :

- Listar todos os utilizadores do sistema.
- Listar todos os recursos de um utilizador.
- Listar todos os recursos do sistema.
- Pesquisar recursos por categoria.
- Pesquisar recursos por título.
- Pesquisar recursos por autor.
- Realizar download de recursos.
- Realizar um relatório de erros.

- Realizar comentários e likes sobre recursos.

Requisitos de Controlo :

- Pesquisa de recursos pode ser efectuada por consumidos, produtores e administradores.
- O download de recursos só é permitido caso o mesmo seja público, ou então esteja a ser feito pelo seu produtor/administrador.
- A gestão de recursos no sistema é feita pelo seu produtor e administrador.

## 3 Funcionamento e Arquitetura do Software

### 3.1 Funcionamento

O funcionamento do sistema é baseado no *Open Archival Information System* que é um modelo de referência para a preservação de longo prazo de activos digitais. Embora não especifique a implementação, o *OAIS* é um guia indispensável para a criação de um arquivo com um conjunto de funções, responsabilidades e métodos que incentivam o arquivamento seguro e de longo prazo e o acesso a informações governamentais críticas.

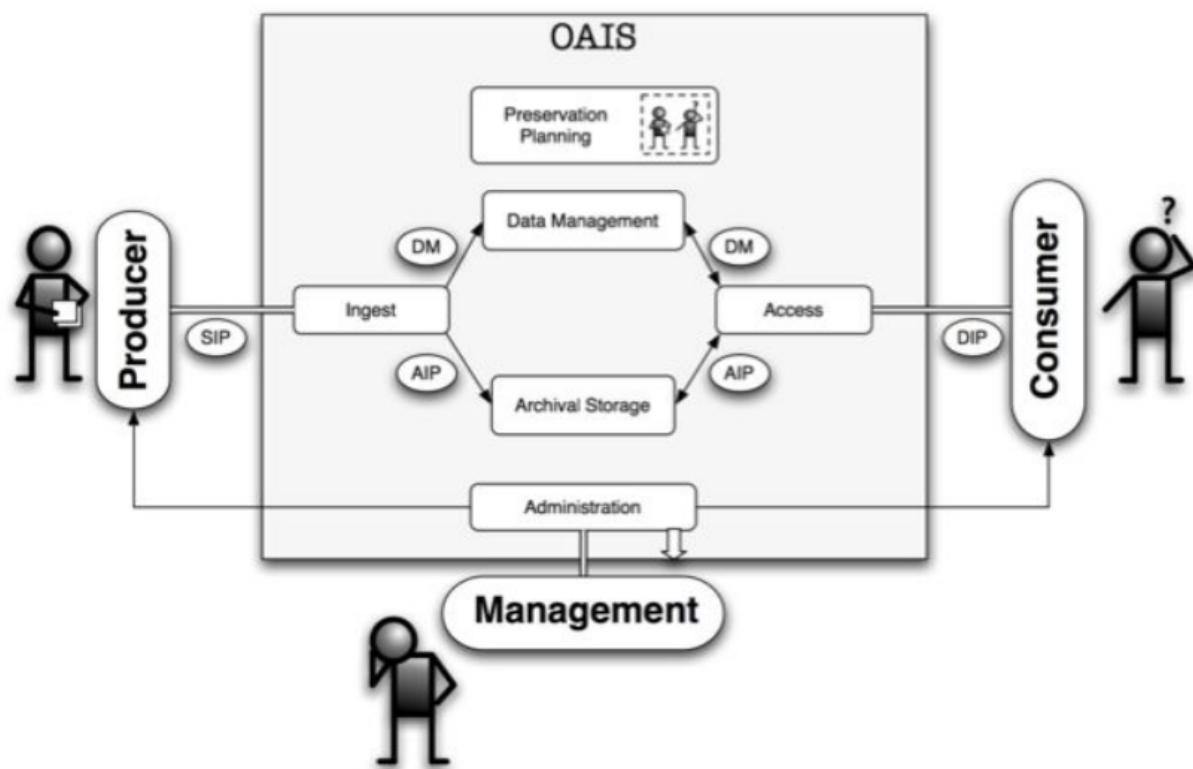


Figura 1: *Open Archival Information System*



Existem 6 unidades funcionais no OAIS :

- **Ingest Function:** Recebe informações dos produtores e as embala para armazenamento , Ela aceita um **SIP**, verificando o mesmo contra o *manifesto* e criando de seguida um **AIP** a partir do **SIP** e transfere o **AIP** recém criado para o armazenamento de arquivos.
- **Archival Storage Function:** Armazena, preserva e recupera **AIPs**. Ela aceita **AIPs** enviados da função **Ingest** atribui-os ao armazenamento de longo prazo, migra **AIPs** conforme necessário, verifica se há erros e fornece **AIPs** solicitados a função **Acess**.
- **Data Managment Function:** Coordena a informação descritiva dos **AIPs** e a informação do sistema que suporta o arquivo. Ela preserva a base de dados que contém as informações do arquivo, executando solicitações de consulta e gerando resultados.
- **Administration function:** Gere as operações diárias do arquivo. Esta função obtém acordos de submissão de produtores de informação realiza engenharia de sistema, audita **SIPs** para garantir a conformidade com acordos de submissão, desenvolve políticas e padrões.
- **Preservation Planning Function:** Oferece suporte a todas as tarefas para manter o material de arquivo acessível e compreensível a longo prazo, mesmo se o sistema de computação original se tornar obsoleto.
- **Acess Function:** Esta função inclui a interface do usuário e permite aos usuários recuperar informações do arquivo. Ela gera um **DIP** a partir do **AIP** relevante e o entrega ao consumidor que solicitou as informações.
- **Submission Information Package:** Um pacote de informações que é entregue pelo produtor ao **OAIS** para uso na construção ou actualização de um ou mais **AIPs** e as informações descritivas associadas, sendo que este **SIP** se encontra no formato **zip**.
- **Archival Information Package:** Um pacote de informações, que consiste nas informações de conteúdo e nas informações de descrição de preservação associadas, que são preservadas em um **OAIS**.
- **Dissemination Information Package:** Um pacote de informação, derivado de um ou mais **AIPs**, e enviado pelos arquivos ao consumidor em resposta a uma solicitação ao **OAIS**, sendo que este **DIP** se encontra no formato **zip**.

## 3.2 Arquitetura do Software

De modo a dividir a carga computacional , foi decidido que seriam implementados três servidores, concretamente *Authentication Server*, *API Server* e *APP Server* como se pode observar pela figura abaixo. Desta forma , os utilizadores têm acesso unicamente ao servidor aplicacional, que trata em background de fazer todos os pedidos aos restantes servidores, sendo que o *Authentication Server* como o próprio nome indica é responsável pela autenticação do utilizador no sistema e o *API Server* responsável pelos pedidos ao servidor do Mongo devolvendo dados no formato *json*.

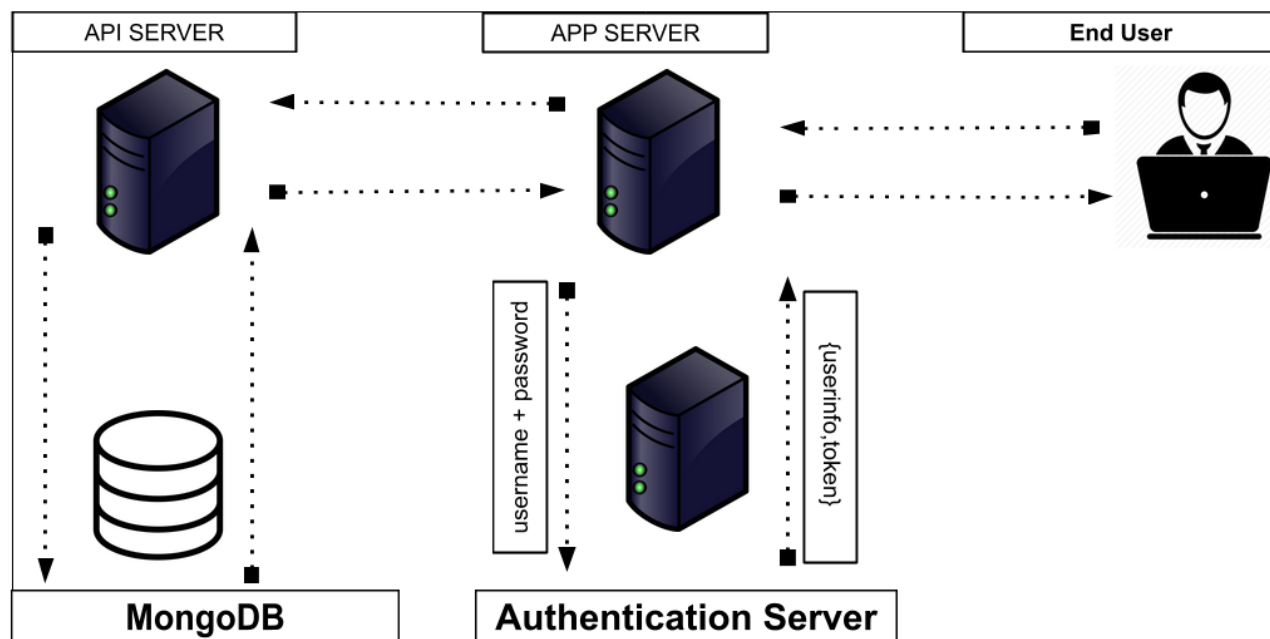


Figura 2: *Arquitetura do Software*

No que concerne à persistência dos dados , estes são armazenados em MongoDB, sendo que o *API Server* é o responsável pela comunicação com o *Mongo Server*.

### 3.2.1 Modelo de dados

De modo garantir a persistência dos dados, foi necessário identificar previamente as entidades do sistema bem com as suas principais características. Feito isto , é apresentado de seguida os modelos criados de modo a atingir os requisitos anteriormente mencionados.

### 3.2.2 Utilizadores

Por razões óbvias a primeira entidade a ser tratada foram os utilizadores. Sendo que existe a necessidade de os mesmos registarem-se e autenticarem-se no sistema foi necessário definir campos como o *username*, *email* e *password*, sendo o último campo encriptado após o registo. Relativamente ao nível de permissões, todos os utilizadores com excepção ao administrador, são registados como *consumers* podendo esse nível ser modificado caso haja alguma produção de recurso. Por razões estatísticas são armazenadas a data de registo bem com o último acesso, tendo ainda um relatório de erros devido a possibilidade de estar a tentar registar recursos que não obedecem as regras do sistema.

```
1  var mongoose = require('mongoose');
2
3
4  var reportSchema = mongoose.Schema({
5    username:{ type: String, required: true },
6    content:{ type: String, required: true },
7    date:{ type: String, required: true }
8  });
9
10 var userSchema = new mongoose.Schema({
11   username: { type: String, required: true },
12   password: { type: String, required: true },
13   email:{ type: String, required: true },
14   register:{ type: String, required: false },
15   lastaccess: { type: String, required: false },
16   level: { type: String, required: true },
17   reports: { type: [reportSchema], required: true }
18 })
19
20 module.exports = mongoose.model('users', userSchema);
```

### 3.2.3 Recursos

Relativamente aos recursos, foi guardada toda metainformação pertinente, tais como:

- Tipo
- Título
- Data de Criação
- Data de Registo
- Nome do produtor/autor do recurso

- Breve descrição
- Caminho aonde se encontra o ficheiro.
- Likes
- Downloads
- Comentários

Foi decidido acrescentar o número de *likes* e *downloads* dos respectivos recursos, sendo estes bons indicadores da qualidade dos mesmos. Quanto aos dados do recurso adicionado, decidiu-se apenas conservar o caminho aonde o mesmo se encontra e não os respectivos bytes, pois o Mongo tem limitações quanto aos bytes que consegue armazenar.

```

1
2 let mongoose = require('mongoose');
3
4 var commentSchema = mongoose.Schema({
5   postId:{ type: String, required: true },
6   comment:{ type: String, required: true },
7   author:{ type: String, required: true },
8   date: { type: String, required: true }
9 });
10
11 var resourceSchema = mongoose.Schema({
12   type:{ type: String, required: true },
13   title:{ type: String, required: true },
14   subtitle:{ type: String, required: true },
15   creationdate:{ type: String, required: true },
16   registerdate:{ type: String, required: true },
17   visibility:{ type: String, required: true },
18   author:{ type: String, required: true },
19   authorId:{ type: String, required: true },
20   description:{ type: String, required: true },
21   comments:{ type: [ commentSchema ], required: false },
22   path:{ type: String, required: true },
23   likes:{ type: [ String ], required: false },
24   downloads:{ type: Number, required: true }
25
26 });
27
28 module.exports = mongoose.model('resources', resourceSchema);

```

### 3.2.4 Categorias

A razão de termos um modelo para registar categorias , consiste na necessidade de prever o aumento de novos tipos de recursos no sistema, sendo que essas são registadas após o sucesso de submissão de um recurso.

```
1  const mongoose = require('mongoose');
2
3  const categorySchema = mongoose.Schema({
4    name:{ type: String, required: true }});
5
6  module.exports = mongoose.model('categories', categorySchema);
```

### 3.2.5 Notícias

A submissão de novos recursos no sistema , origina uma inserção da informação pertinente sobre o mesmo numa timeline , que é por sua vez partilhada por todos os utilizadores do sistema, sendo que se o mesmo recurso for público é possível fazer o *download* do mesmo.

```
1  const mongoose = require('mongoose');
2
3  const newsSchema = mongoose.Schema({
4    date:{ type: String, required: true },
5    author:{ type: String, required: true },
6    news:{ type: String, required: true },
7    description:{ type: String, required: true },
8    resource:{ type: String, required: true }
9  });
10
11 module.exports = mongoose.model('news', newsSchema);
```

### 3.3 Autenticação

Quanto à autenticação no sistema, o utilizador efectua o *login* através do username e password, sendo que é feito um pedido ao *Authentication Server* e o mesmo fornece os dados deste utilizador descriptando a password de modo a fazer a sua validação. Quando o mesmo efectua o *login*, é gerada uma sessão para o mesmo, sendo que esta permite que o utilizador permaneça autenticado mesmo que o servidor sofre uma falha. Permite, também, que o utilizador possa abrir uma nova janela na aplicação, permanecendo autenticado. Para a implementação deste sistema de sessões foi utilizada uma estratégia local tirando partido do módulo *passport* para gestão destas.

Relativamente ao *logout* do sistema, é feita a destruição da sessão em uso, permitindo assim que o utilizador se desautentique completamente do sistema, redireccionando para a *página inicial*.

#### 3.3.1 Rotas com autenticação

Relativamente a rotas protegidas, usou-se duas abordagens diferentes, sendo que uma delas consiste em proteger a rota com um token, tendo sido usada essa abordagem para proteger a informação de todo o sistema, podendo ser consultada exclusivamente pelo administrador do sistema. E a segunda abordagem consistiu em usar a informação contida no token para fazer a verificação dos recursos a apresentar e as acções sobre o mesmo. A geração do token foi feita com recurso ao módulo *jsonwebtoken*.

```
1 jwt.sign({ userId, username, level }, privateKey, { algorithm: 'RS256' })
```

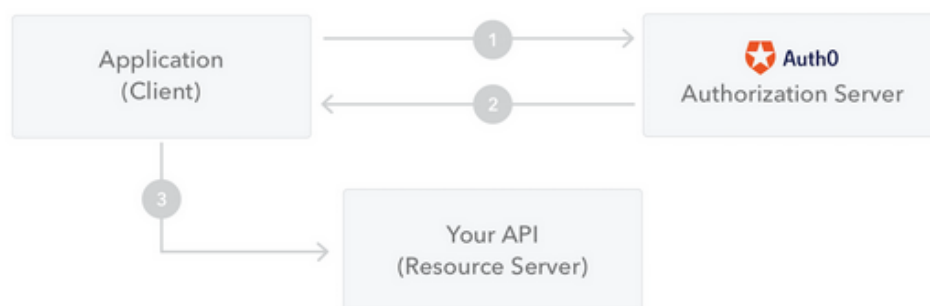


Figura 3: *Autenticação*

## 4 Resultados Obtidos

De seguida é feita a apresentação dos resultados finais.

### 4.1 Homepage

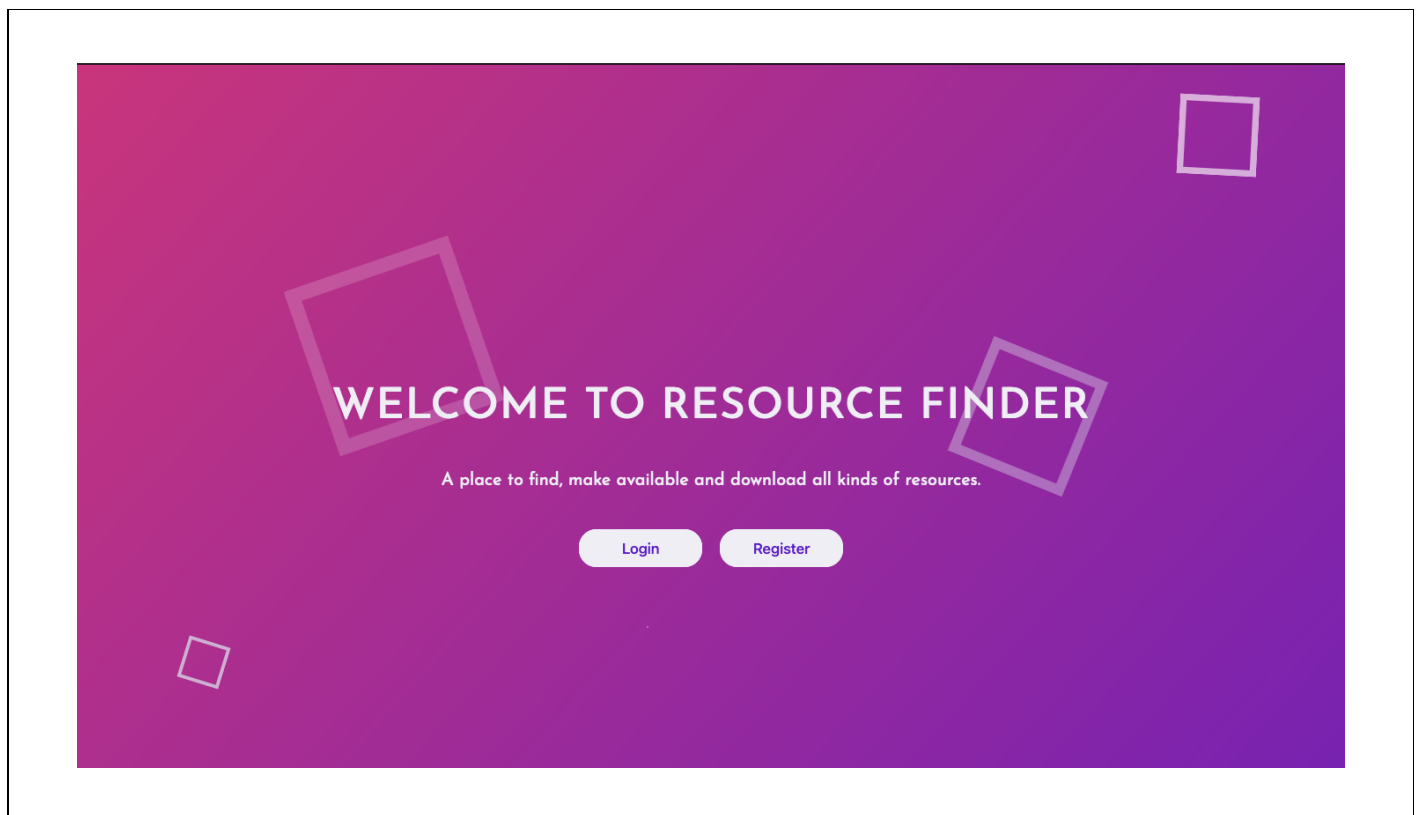
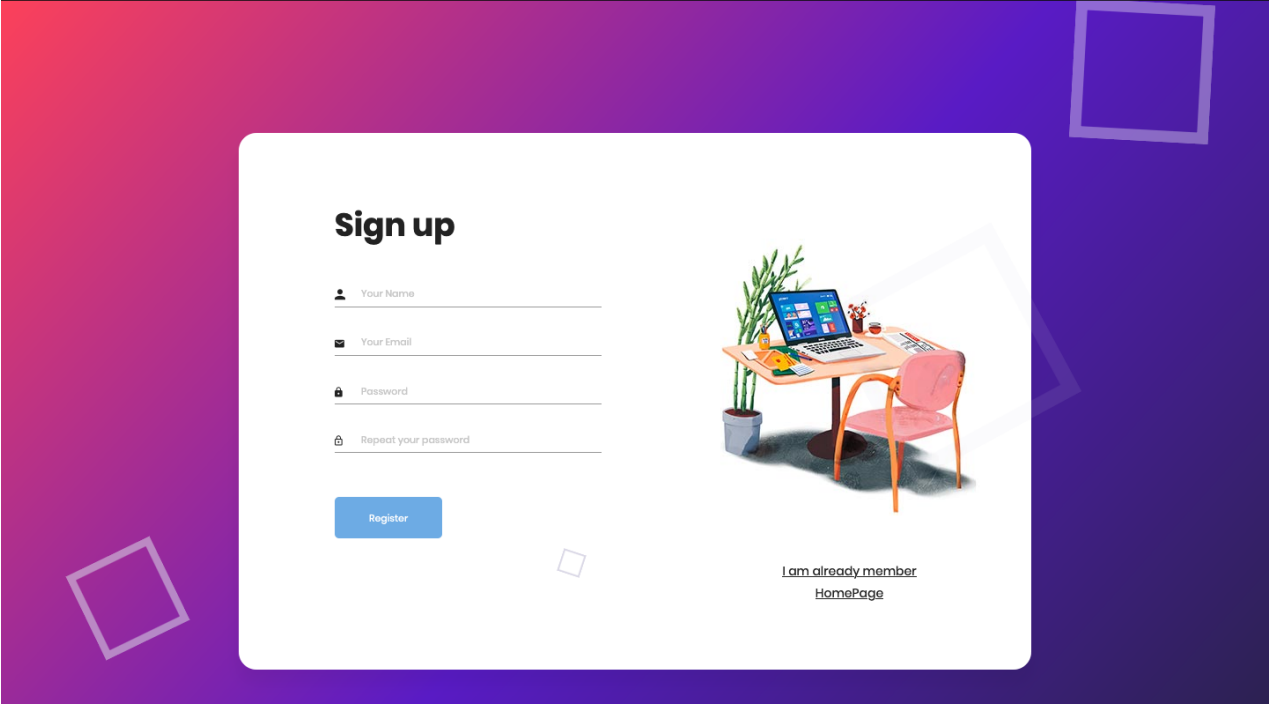


Figura 4: Homepage

## 4.2 Registrar Utilizador



**Sign up**

Your Name

Your Email

Password

Repeat your password

[I am already member](#)  
[HomePage](#)

The registration form is centered on a white background with rounded corners. It features a title 'Sign up' in bold black text. Below the title are four input fields: 'Your Name', 'Your Email', 'Password', and 'Repeat your password'. Each field has a small icon (person, envelope, and padlock respectively) to its left. A blue 'Register' button is positioned below the input fields. To the right of the input fields is a 3D illustration of a desk with a laptop, a potted plant, and a pink chair. At the bottom right of the form, there are two links: 'I am already member' and 'HomePage'. The entire form is set against a background with a purple-to-pink gradient and faint, tilted square outlines.

Figura 5: Registrar Utilizador



## 4.3 Login

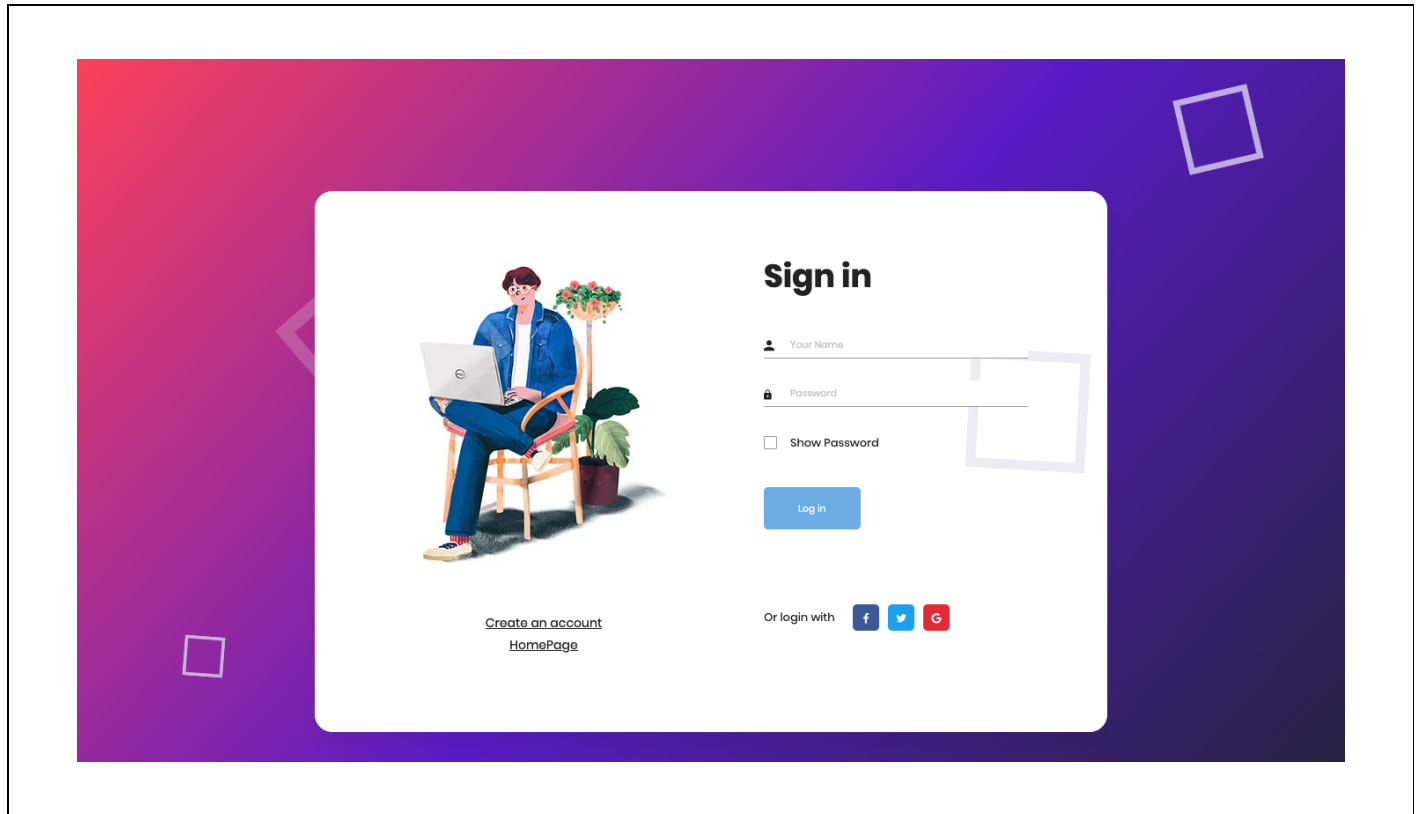


Figura 6: Login de um utilizador

## 4.4 Dashboard

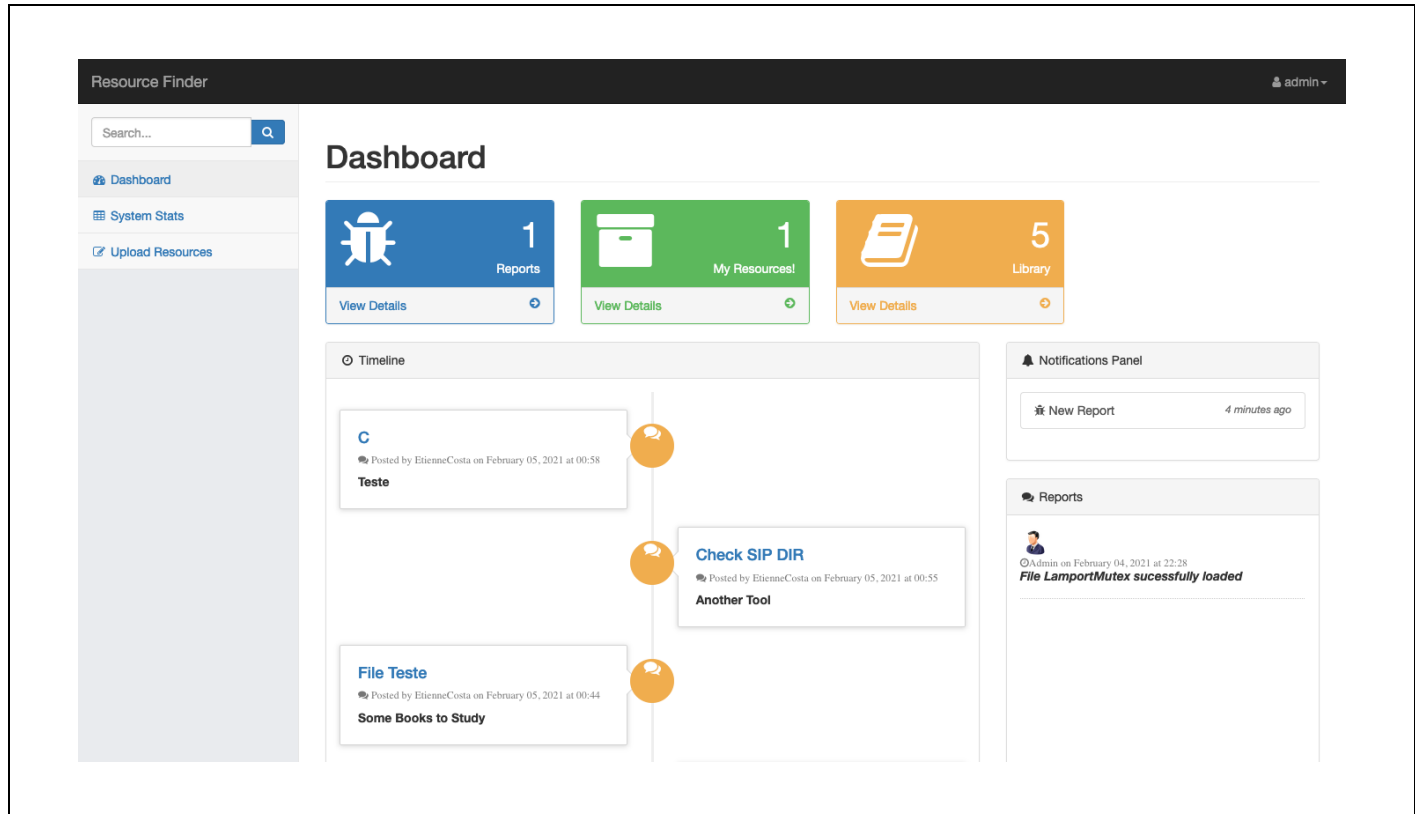


Figura 7: Dashboard com o feed e reports

## 4.5 Registrar recurso

The screenshot shows a web application titled "Resource Finder" with a user profile "EtienneCosta" in the top right. A left sidebar contains a search bar and navigation links: "Dashboard", "System Stats", and "Upload Resources" (which is highlighted). The main content area is titled "Register Resources" and contains a "Register Form". The form includes the following fields and controls:

- Title:** A text input field with placeholder text "Insert your title ...".
- Subtitle:** A text input field with placeholder text "Insert your subtitle ...".
- Type:** A dropdown menu with "Choose..." as the selected option and a small downward arrow.
- Add New Type:** A blue button.
- Creation Date:** A date input field with the placeholder "dd / mm / aaaa".
- Visibility:** Radio buttons for "On" (selected) and "Off".
- File Input:** A button labeled "Explorar..." followed by the text "Nenhum ficheiro selecionado."
- Description:** A large text area for writing the description.
- Submit and Reset:** Two buttons at the bottom, "Submit" (green) and "Reset" (red).

Figura 8: Registrar recurso

## 4.6 Estatísticas do Sistema

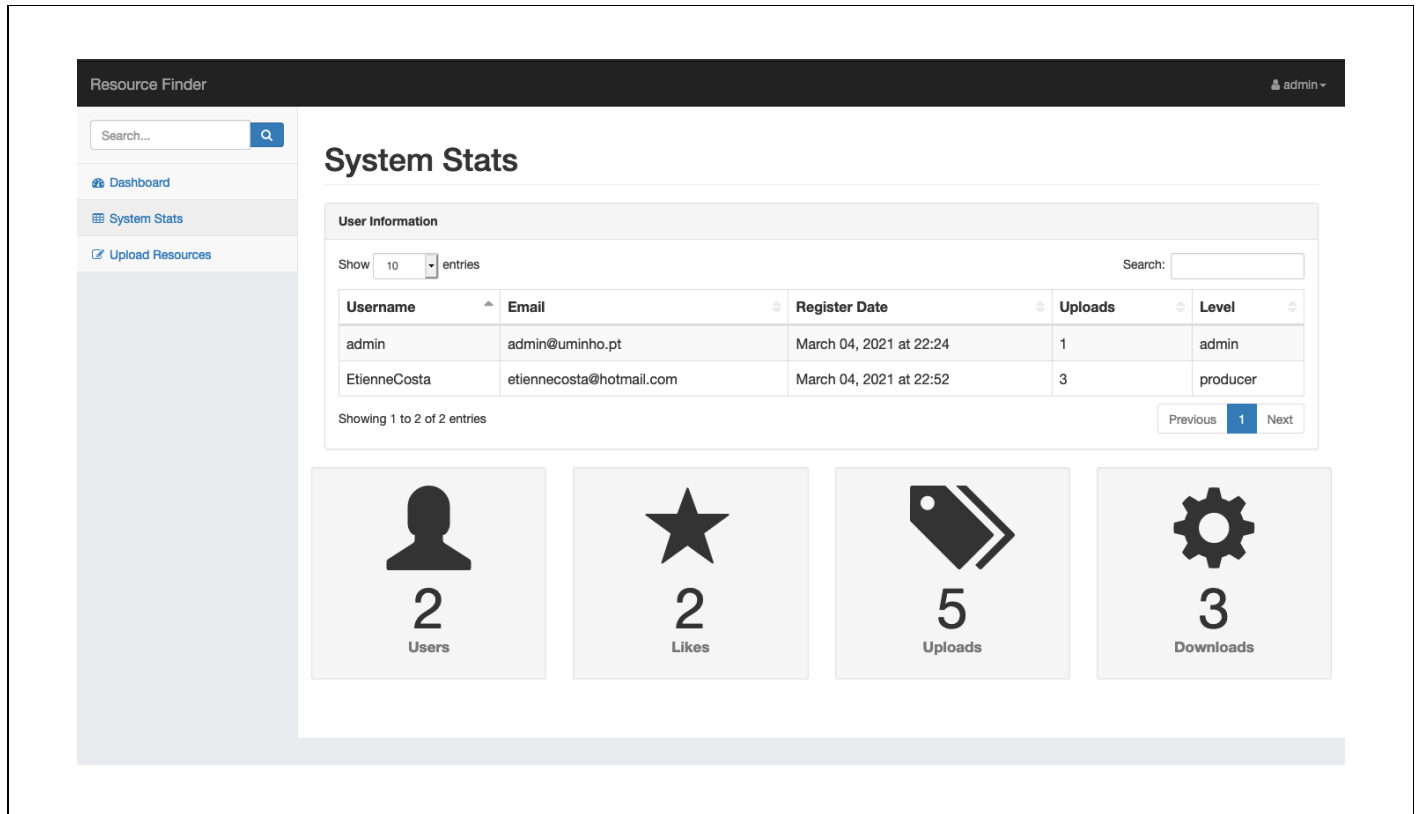


Figura 9: Estatísticas do sistema

## 4.7 Recursos

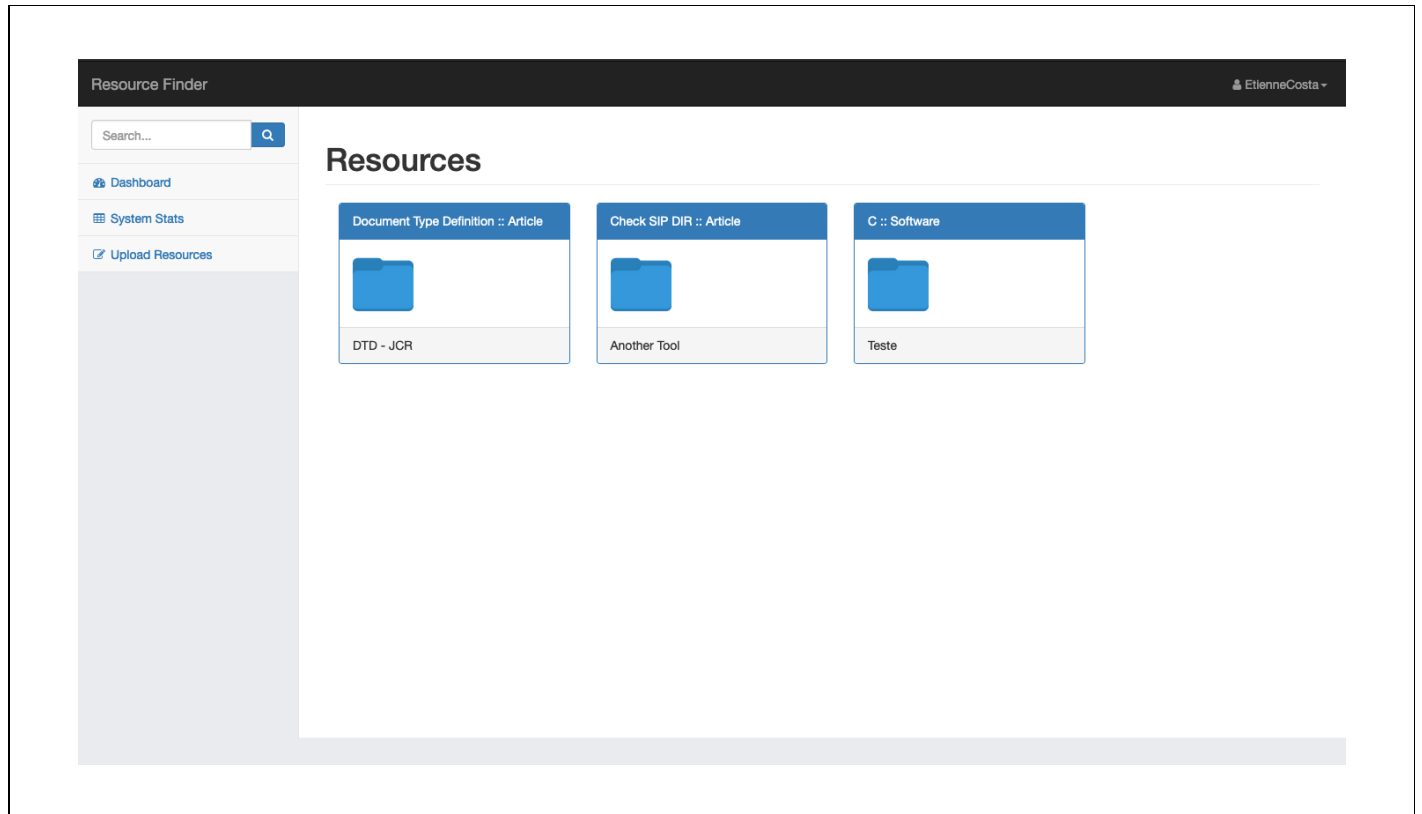


Figura 10: Lista de recursos

## 4.8 Recurso

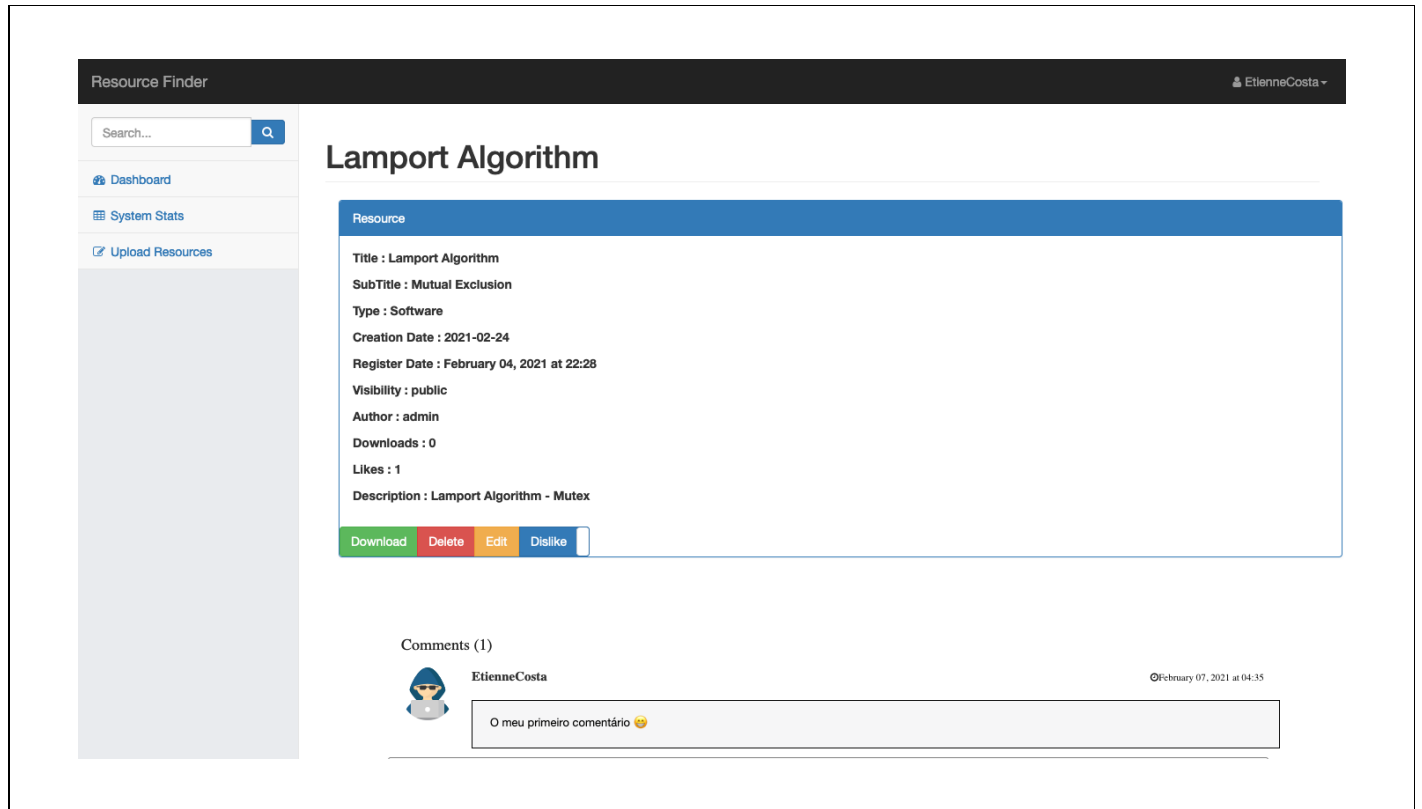


Figura 11: Recurso

## 4.9 Comentários

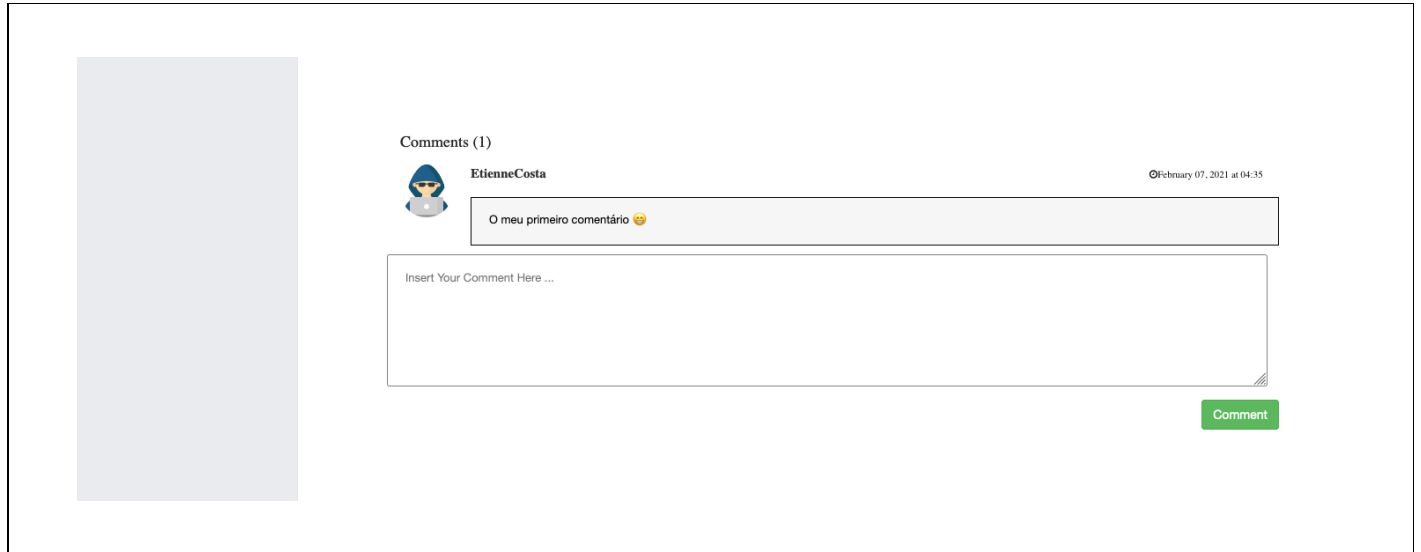


Figura 12: Comentários de um recurso

## 5 Conclusão

Durante a realização do relatório foram abordados os passos referentes ao desenvolvimento e implementação de uma aplicação web que permitisse a partilha de recursos entre utilizadores. Foi, desta forma, necessário pensar nos componentes cruciais a uma aplicação destas, bem como funcionalidades que seriam produtivas de implementar. Tendo em conta que a utilização de um servidor API de dados e um servidor Aplicacional permitem dividir o custo computacional, bem como reduzir o bottleneck da aplicação, esta foi a arquitetura escolhida. Assim, foi necessário manter uma ligação constante entre os servidores, permitindo a troca de dados. No que toca aos objetivos propostos, todos estes foram implementados com sucesso. Visto que todo software é passível de actualizações, o nosso não foge a regra, sendo que num futuro próximo surgirá a necessidade de fazer-se uma revisão futura de modo a reduzir code smells presentes no programa. De modo geral, pode-se dizer que a realização deste trabalho exigiu a aplicação de diversos conceitos lecionados, permitindo assim alcançar os objetivos propostos.