

UNIVERSIDADE DO MINHO

PERFIL DE ENGENHARIA DE APLICAÇÕES

Trabalho Prático
RELATÓRIO DE DESENVOLVIMENTO

Mestrado Integrado em Engenharia Informática

Desenvolvido pelo grupo 3:
Filipa Alves dos Santos, a83631
Hugo André Coelho Cardoso, a85006
João da Cunha e Costa, a84775
Luís Miguel Arieira Ramos, a83930
Válter Ferreira Picas Carvalho, a84464

Conteúdo

1	Introdução	4
1.1	Motivação	4
1.2	Contextualização	4
1.3	Caracterização do utilizador	4
2	Modelação	5
2.1	Requisitos	5
2.1.1	Funcionais	5
2.1.2	Não Funcionais	5
2.2	Diagrama de Domínio	6
2.3	Diagrama de Tecnologias (Arquitetura)	6
2.4	Diagrama de Use Cases	7
2.5	Modelos de Tarefas	8
2.5.1	Modelo 1: Fazer publicação sobre recurso	8
2.5.2	Modelo 2: Carregar recurso	9
2.5.3	Modelo 3: Fazer download de um recurso	9
2.6	Protótipo da Interface	9
2.6.1	Página Home	10
2.6.2	Página de Perfil	10
2.6.3	Página dos Recursos	11
2.6.4	Página de um Recurso	13
2.6.5	Página dos Utilizadores	14
3	Implementação	15
3.1	Frontend	15
3.1.1	Views	15
3.1.2	Pedidos à API	15
3.2	Backend	15
3.2.1	Controllers (Servlets)	16
3.2.2	Enterprise Java Beans	16
3.2.3	Persistência	17
4	Deploy	18
4.1	Considerações Iniciais	18
4.2	Frontend	18
4.3	Backend	19
4.4	Persistência	19
4.5	Resultados e Problemas	19
5	Análise de Carga	20
5.1	Considerações Iniciais	20
5.2	Testes de Carga	20
6	Interface Final	21
6.1	Página Inicial	21
6.2	Página Inicial - Login	21
6.3	Página Inicial - Registo	22
6.4	Home	22
6.5	Utilizadores	23
6.6	Recursos - Search	23

6.7	Recursos	24
6.8	Adicionar Recurso	24
6.9	Recurso	25
6.10	Adicionar Classificação	26
6.11	Editar Recurso	26
6.12	Adicionar Publicação	27
6.13	Publicação	27
6.14	Editar Publicação	28
6.15	Perfil	28
6.16	Editar Perfil	29
7	Análise de Usabilidade	30
7.1	Princípios de Usabilidade	30
7.1.1	Learnability	30
7.1.2	Flexibility	30
7.1.3	Robustness	30
7.2	Heurísticas de Nielsen	30
7.2.1	Visibility of system status	30
7.2.2	Match between system and the real world	31
7.2.3	User control and freedom	31
7.2.4	Consistency and standards	31
7.2.5	Error prevention	31
7.2.6	Recognition rather than recall	31
7.2.7	Flexibility and efficiency of use	32
7.2.8	Aesthetic and minimalist design	32
7.2.9	Help users recognize and recover from errors	32
7.2.10	Help and documentation	32
8	Conclusão e Trabalho Futuro	33

Listagens

Lista de Figuras

1	Diagrama de Domínio	6
2	Diagrama de Tecnologias	7
3	Diagrama de Use Cases	8
4	Diagrama de Tarefa 1 - Fazer publicação sobre recurso	8
5	Diagrama de Tarefa 2 - Carregar recurso	9
6	Diagrama de Tarefa 3 - Fazer download de um recurso	9
7	Mockup da Página Home	10
8	Mockup da Página de Perfil	10
9	Mockup da Página de Editar o Perfil	11
10	Mockup da Página dos Recursos	11
11	Mockup da Página de Adicionar um Recurso	12
12	Mockup da Página de Criar uma Publicação	12
13	Mockup da Página de um Recurso Individual I	13
14	Mockup da Página de um Recurso Individual II	13
15	Mockup da Página dos Utilizadores	14
16	Arquitetura do Deployment	18
17	Testes de Carga	20
18	Página Inicial	21
19	Página Inicial de Login	21
20	Página Inicial de Registo	22
21	Página Inicial com login efetuado	22
22	Página de utilizadores	23
23	Página de escolha de recursos por tipo	23
24	Página dos recursos	24
25	Adicionar um recurso	24
26	Preview de um ficheiro que será adicionado	24
27	Página do recurso	25
28	Página do recurso	25
29	Preview dos ficheiros do recurso	25
30	Publicações sobre o recurso	26
31	Nova classificação do recurso	26
32	Editar o recurso	26
33	Nova publicação sobre o recurso	27
34	Página da publicação	27
35	Continuação da página da publicação	27
36	Editar a publicação	28
37	Página do perfil	28
38	Página do perfil	28
39	Editar o perfil	29
40	Load dos recursos	31
41	Confirmação ao apagar um recurso	31
42	Erro no login	32

1 Introdução

1.1 Motivação

Em diversas circunstâncias, a procura de certos recursos educativos digitais, tais como uma tese sobre determinado assunto ou até mesmo uma aplicação para um objetivo específico, provasse difícil na amplitude de informação disponível *online* atualmente.

Para além disso, mesmo quando plataformas desse género são implementadas, são muito limitadas em termos de tipo de conteúdo que pode ser partilhado e também pouco intuitivas na pesquisa de recursos de um certo tipo ou de uma certa área. Outro aspeto em falta é a parte social, onde *feedback* possa ser dado de uma maneira mais fácil e onde ideias de diferentes utilizadores possam ser facilmente trocadas, de modo informal.

Assim, a ideia deste projeto surgiu como uma maneira de resolver a necessidade de um repositório de partilha de recursos de diversos tipos, com uma componente social embutida e outras funcionalidades extra que ajudam a aprimorar a procura e partilha de recursos por parte do utilizador.

1.2 Contextualização

A **Redu** é uma plataforma de disponibilização de recursos educativos *online*, em que utilizadores podem submeter os seus próprios recursos e/ou fazer o *download* de outros disponíveis. Em suma, as principais características da aplicação são as seguintes:

1. Utilizadores podem registar-se e ter acesso a uma página de perfil pessoal, que pode ser editada pelo próprio
2. Os recursos disponíveis podem ser filtrados por algum dos seus meta dados
3. Utilizadores podem dar *upload* a novos recursos, com um ou mais ficheiros, sendo que os podem editar posteriormente
4. Adicionalmente, podem fazer o download dos recursos disponíveis, bem como classificá-los
5. Ficheiros dos recursos podem ser pré-visualizados antes do *download*
6. Têm também acesso a um lista dos utilizadores, com filtro disponível
7. Podem também fazer publicações sobre os recursos e comentar nestas mesmas

1.3 Caracterização do utilizador

Como esta plataforma se indica à partilha de recursos educativos em forma digital, o público alvo seria alguém com as seguintes características:

1. Alguma experiência a utilizar com tecnologia
2. Entre os 18 e os 70 anos
3. Alguma formação académica
4. Estudantes, professores ou outro cargo na área da investigação e/ou do ensino

2 Modelação

A modelação é essencial para a realização de uma boa aplicação pois permite-nos visualizar um sistema como queremos que seja e especificar a estrutura /comportamento dele para orientar a construção do mesmo.

Nestas secções seguintes, serão mostrados alguns dos diagramas e *mockups* criados no início do desenvolvimento deste projeto, sendo que alguns não coincidem totalmente com o resultado final devido a mudanças feitas posteriormente.

2.1 Requisitos

O levantamento de requisitos vem em função do tipo de utilizadores definidos para a aplicação, pelo que serão expostos de seguida

2.1.1 Funcionais

A aplicação deverá cumprir os seguintes requisitos funcionais quanto ao Utilizador:

- Permitir o registo na aplicação fornecendo informação pessoal (nome, email, password, descrição, filiação, cargo, entre outros);
- Permitir o login na aplicação, fornecendo o email e password;
- Visualizar a lista de outros utilizadores registados;
- Visualizar o perfil de um utilizador registado;
- Editar o seu perfil, modificando os seus dados pessoais e a fotografia;
- Inserir um recurso novo na plataforma indicando os ficheiros e metadados (título, descrição, data de registo, entre outros);
- Editar ou eliminar um recurso já inserido, assim como marcá-lo como público ou privado;
- Realizar uma publicação sobre um recurso disponível na plataforma; Visualizar a lista de recursos disponíveis;
- Visualizar um recurso em concreto;
- Pré visualizar os ficheiros de um recurso;
- Descarregar um ou mais recursos da plataforma.

2.1.2 Não Funcionais

A nível não funcional, é imperativo que a aplicação tenha as seguintes características:

1. A aplicação deverá estar escrita em português;
2. Os recursos não devem conter conteúdo sensível;
3. A aplicação deverá suportar os *browsers* clássicos (Chrome, Firefox, etc);
4. O sistema deve estar disponível o máximo de tempo possível;
5. A aplicação deve ter uma interface reativa;
6. A interface deverá ser facilmente navegável;

7. O sistema deverá ser tolerante a falhas;
8. O sistema deverá ter tempos de resposta inferiores a 2s, removendo os casos de atrasos rede;

2.2 Diagrama de Domínio

O Diagrama de Domínio ilustra as classes significativas no domínio do problema em questão. Neste caso em específico, temos as classes principais como o **Consumidor**, que por sua vez pode ser um Produtor e até um Administrador (embora este último não tenha sido implementado). O consumidor pode criar e fazer *download* de um **Recurso**, que tem associado um ou mais **Ficheiros**, ou ainda fazer uma **Publicação** sobre um determinado recurso ou até fazer um **Comentário** em publicações já existentes. Cada uma destas classes principais têm dados associados como o **Nome** no caso do produtor e **Título** no caso do recurso.

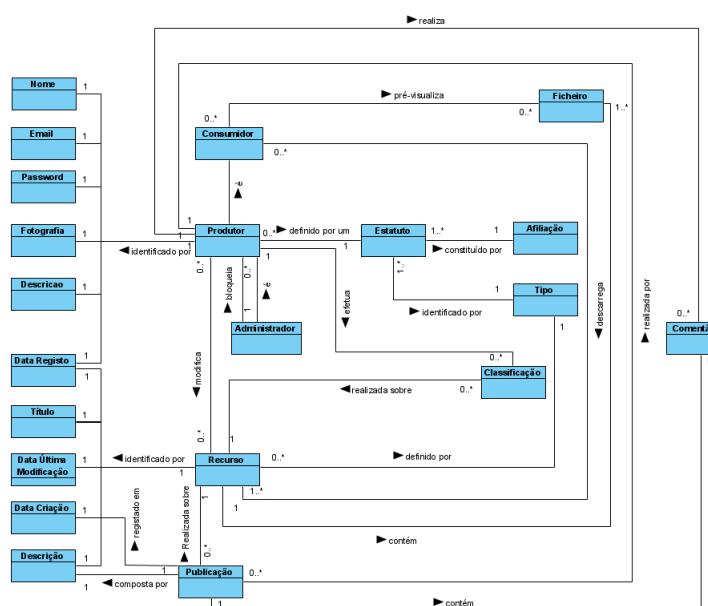


Figura 1: Diagrama de Domínio

2.3 Diagrama de Tecnologias (Arquitetura)

Também no início do projeto, foi feito um esquema que resume as tecnologias pretendidas para avaliar possíveis combinações. De seguida, são explicadas as escolhas das tecnologias para cada camada.

Começando pela **camada de dados**, optamos por usar **Hibernate** para utilizar para bases de dados relacionais (Hibernate ORM). O Hibernate mapeia classes Java para tabelas na base de dados e tipos de dados Java para tipos de dados SQL aliviando o desenvolvedor das tarefas de programação relacionadas à persistência de dados mais comuns. Alguns das suas vantagens incluem estratégias de "smart fetching", que minimizam acessos à base de dados, a criação automática de tabelas, que exclui a necessidade de as criar manualmente e queries independentes da base de dados pois, quando ocorre uma mudança de base de dados, não é necessário criar novas queries.

No que toca à **camada de lógica de negócio**, a framework de eleição foi **Spring**, a mais popular para desenvolvimento de aplicações no Java Enterprise, sendo uma das principais razões a grande compatibilidade com Hibernate. Outra das suas vantagens são utilizar MVC, encoraja a produtividade generalizando comportamento genérico reaproveitado por várias aplicações, tais

como o acesso à camada de dados e não ser necessário gerar código visto que o utilizador só necessita de editar ficheiros XML para alterar configurações

Finalmente, para a **camada de apresentação** optamos por **Vue**. Esta framework permite definir handlers encapsulados de eventos, nomeadamente cliques de ratos ou de teclas, através de diretivas associadas a certos elementos da vista, garantindo o correto mapeamento dos eventos ao código respetivo. É uma framework reativa, pelo que consegue recarregar elementos individuais da front-end em resposta a eventos, evitando a necessidade de recarregar a página inteira, permitindo assim uma gestão mais eficiente e adequada a este paradigma. É também fácil estender a aplicação a novos tipos de eventos que surjam, sendo facilmente escalável.

Concluindo, a combinação de frameworks escolhida proporciona um processamento bastante eficiente e encapsulado dos eventos ao longo das camadas, recorrendo a um estado centralizado na camada de visualização para evitar acessos desnecessários à base de dados, bem como uma grande escalabilidade do sistema.

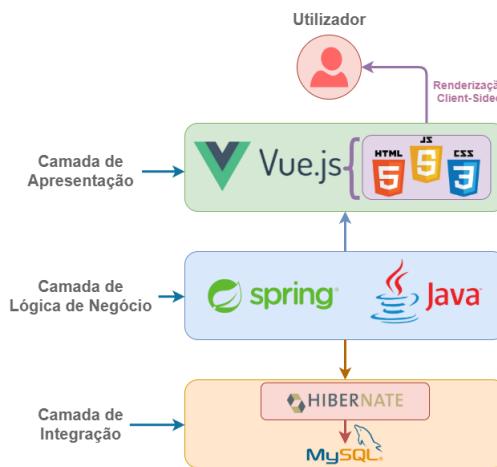


Figura 2: Diagrama de Tecnologias

2.4 Diagrama de Use Cases

O Diagrama de Use Cases mostra as funcionalidades do sistema do ponto de vista do utilizador, especificando todos os requisitos funcionais do sistema. Na imagem seguinte, temos todos os use cases considerados para este projeto, desde os mais específicos como "Classificar recurso" até aos principais como "Fazer download de um recurso".

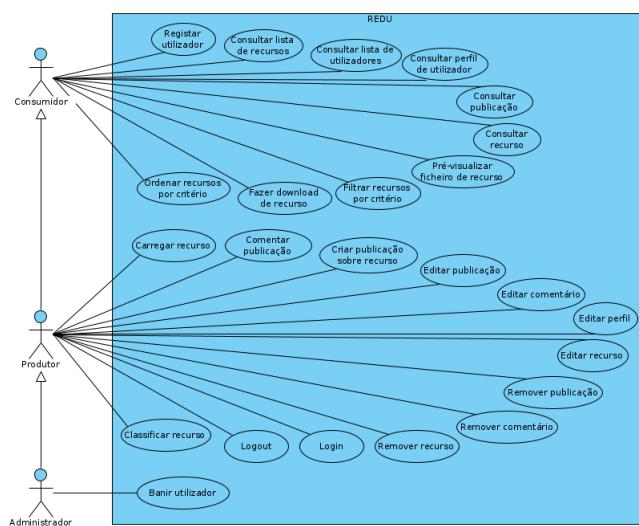


Figura 3: Diagrama de Use Cases

2.5 Modelos de Tarefas

Em termos de Modelos de Tarefas, foram feitos apenas os 3 principais: "Fazer publicação sobre recurso", "Carregar recurso" e "Fazer download de um recurso". Estes modelos servem para demonstrar a sequência de que têm ações de ser realizadas pelo sistema e pelo utilizador para executar determinada tarefa. Também contém detalhes sobre a ordem e diferentes rotas que podem ser tomadas.

2.5.1 Modelo 1: Fazer publicação sobre recurso

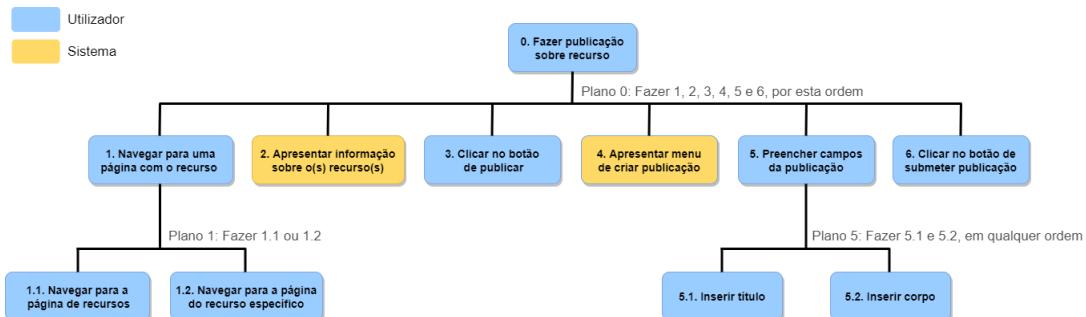


Figura 4: Diagrama de Tarefa 1 - Fazer publicação sobre recurso

2.5.2 Modelo 2: Carregar recurso

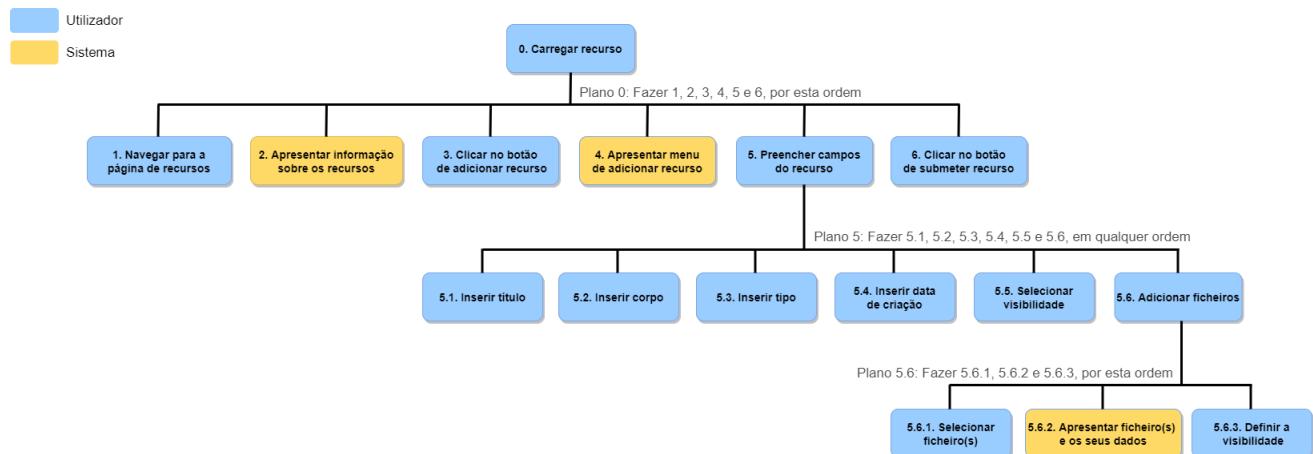


Figura 5: Diagrama de Tarefa 2 - Carregar recurso

2.5.3 Modelo 3: Fazer download de um recurso

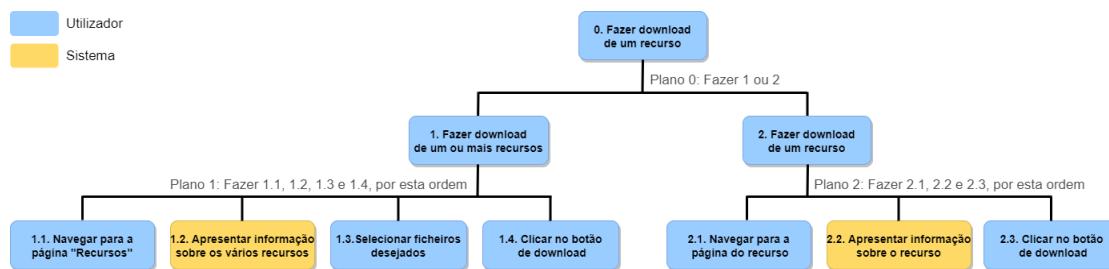


Figura 6: Diagrama de Tarefa 3 - Fazer download de um recurso

2.6 Protótipo da Interface

Para protótipos da interface, foram feitas mockups de todas as páginas web que eram esperadas na implementação para verificar sua usabilidade e a existência de potenciais problemas. Prototipagem como esta é sempre importante fazer para economizar tempo e recursos e colocar a equipa responsável pelo seu desenvolvimento na perspectiva do utilizador.

2.6.1 Página Home

Publicações

- Título da Publicação**
Recurso: Nome Recurso
Nome utilizador há 5 horas
- Título da Publicação 2**
Recurso: Nome Recurso 2
Nome utilizador 2 há 5 horas
- Título da Publicação 3**
Recurso: Nome Recurso 3
Nome utilizador 3 há 5 horas
- Título da Publicação 4**
Recurso: Nome Recurso 4
Nome utilizador 4 há 5 horas
- Título da Publicação 5**
Recurso: Nome Recurso 5
Nome utilizador 5 há 5 horas
- Título da Publicação 6**
Recurso: Nome Recurso 6
Nome utilizador 6 há 5 horas

Ver mais

Novos Recursos

- Nome do Recurso**
Estado: Novo
Tipo: Novo Tipo
Nome utilizador há 5 horas
- Nome do Recurso 2**
Estado: Atualizado
Tipo: Tipo 4
Nome utilizador há 2 meses
- Nome do Recurso 3**
Estado: Indisponível
Tipo: Novo Tipo
Nome utilizador há 5 horas
- Nome do Recurso 4**
Estado: Atualizado
Tipo: Tipo 3
Nome utilizador há 2 meses

© RR 2021

Figura 7: Mockup da Página Home

2.6.2 Página de Perfil

Nome do Utilizador

Filiação: Escola X (Estudante)
E-mail: escolax@gmail.com
Descrição: Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Editar Perfil **Mudar Foto**

Eu

2021
11 de fevereiro

16:10: Carreguei um novo recurso: *Nome Recurso*
16:05: Criei uma nova publicação: *Nome Publicação*

2021
11 de fevereiro

Registo na Plataforma

10:10: Registado com sucesso.

© RR 2021

Figura 8: Mockup da Página de Perfil

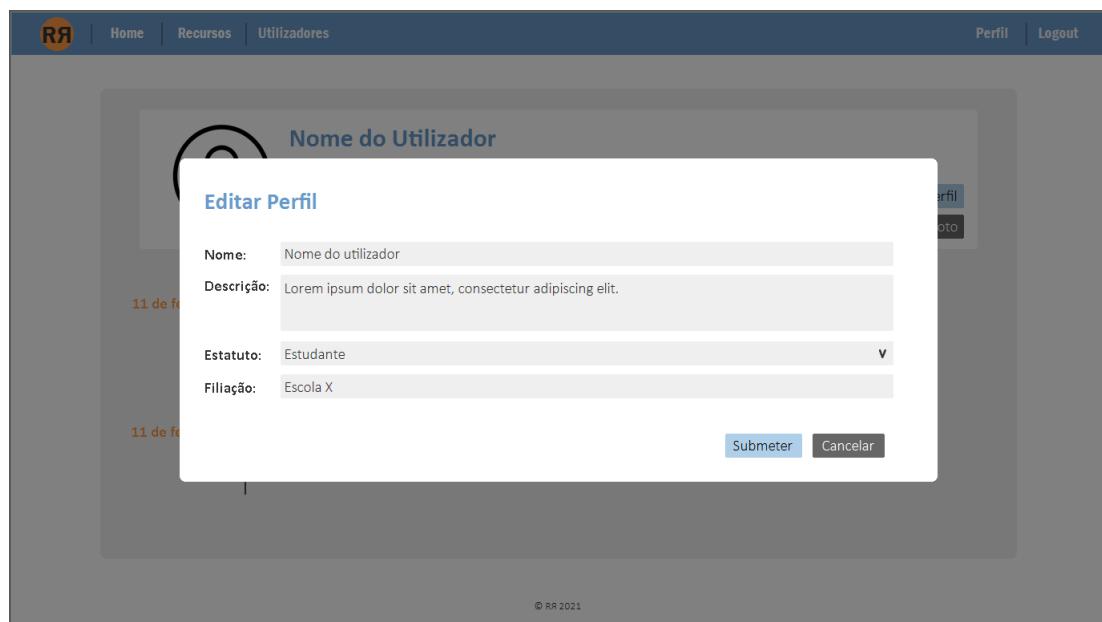


Figura 9: Mockup da Página de Editar o Perfil

2.6.3 Página dos Recursos

A screenshot of a web application interface showing a list of resources. The top navigation bar is identical to Figure 9, with the 'Recursos' link highlighted. Below the navigation, there's a search/filter bar with 'Novo Recurso' and 'Filtrar por...' buttons, and an orange 'Download' button on the right. The main content area displays a table of resources with the following columns: Título (Title), Tipo (Type), Autor (Author), Classificação (Classification), Nº de Downloads (Number of Downloads), and Última Modificação (Last Modification). Each row shows a small checkbox icon, the resource title, its type, author, a five-star rating icon, the download count, the last modification date, and a '+ Publicação' button. At the bottom of the table, there's a 'Ver mais' (View more) button. The footer of the page includes the copyright notice '© RR 2021'.

Título	Tipo	Autor	Classificação	Nº de Downloads	Última Modificação	
<input type="checkbox"/> Título do Recurso	Novo Tipo	Nome do Autor	★★★★★	0	11-02-2021	+ Publicação
<input type="checkbox"/> Título do Recurso	Tipo 4	Nome do Autor	★★★	1	12-02-2021	+ Publicação
<input type="checkbox"/> Título do Recurso	Novo Tipo	Nome do Autor	★★★★★	0	01-03-2021	+ Publicação
<input type="checkbox"/> Título do Recurso	Novo Tipo	Nome do Autor	★★★★★	2	20-01-2021	+ Publicação
<input type="checkbox"/> Título do Recurso	Tipo 7	Nome do Autor	★★★★★	4	11-12-2020	+ Publicação
<input type="checkbox"/> Título do Recurso	Tipo 9	Nome do Autor	★★★	0	10-04-2021	+ Publicação
<input type="checkbox"/> Título do Recurso	Novo Tipo	Nome do Autor	★★★★★	1	11-02-2021	+ Publicação
<input type="checkbox"/> Título do Recurso	Tipo 1	Nome do Autor	★★★★★	1	16-02-2021	+ Publicação

Figura 10: Mockup da Página dos Recursos

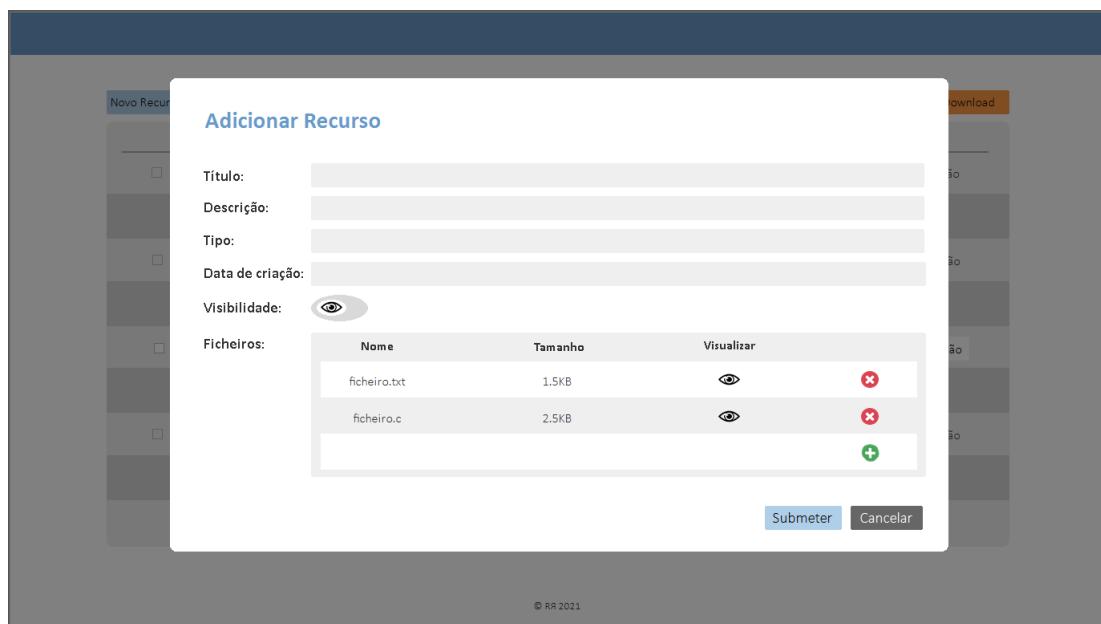


Figura 11: Mockup da Página de Adicionar um Recurso

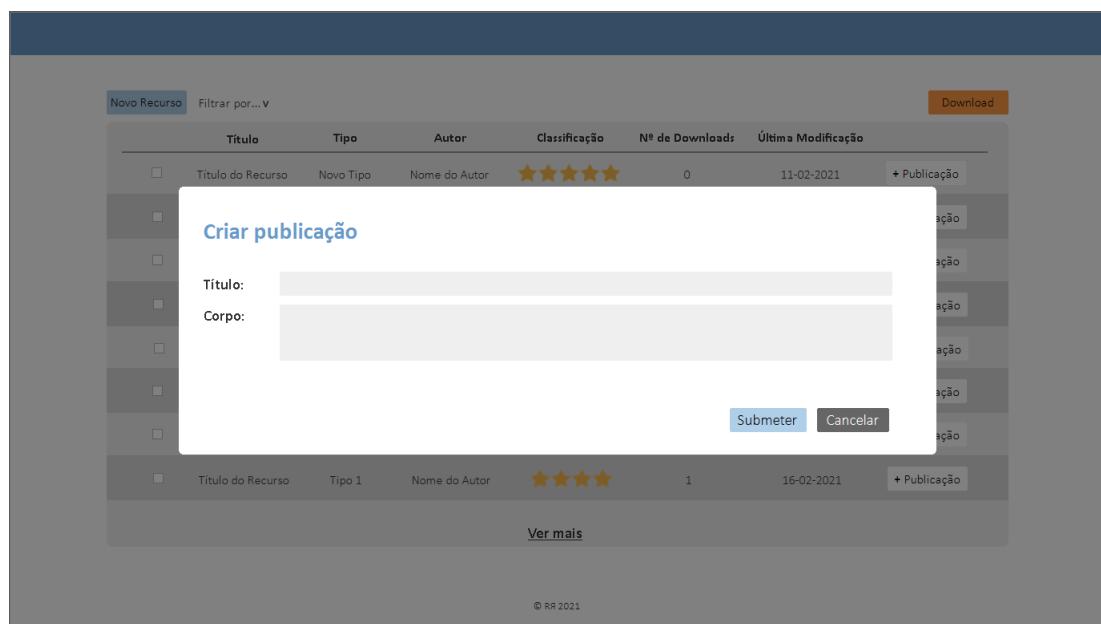


Figura 12: Mockup da Página de Criar uma Publicação

2.6.4 Página de um Recurso

The mockup shows a web page titled "Nome do Recurso". At the top, there are tabs for "Informação" and "Visualizar". Below these, detailed resource information is displayed in two columns:

Tipo: Tipo 4	Autor: Pedro Degaldo
Descrição: Lorem ipsum dolor sit amet, consectetur adipiscing elit.	Última Modificação: 15:41:12 18-01-2021
Data de criação: 20-01-2021	Número de publicações: 0
Data de registo: 15:41:12 18-01-2021	Número de downloads: 1
Classificação: ★★★★☆	Tamanho total: 5KB

At the bottom right of the page, it says "© RR 2021".

Figura 13: Mockup da Página de um Recurso Individual I

The mockup shows a web page titled "Nome do Recurso". At the top, there are tabs for "Informação" and "Visualizar". Below these, a table lists four files with their sizes and preview icons:

Ficheiro	Tamanho	Visualizar
ficheiro.txt	1.5KB	👁
ficheiro.c	2.5KB	👁
ficheiro.h	1.6KB	👁
ficheiro.y	1.5KB	👁

At the bottom left are "Download" and "Publicar" buttons. At the bottom right, it says "© RR 2021".

Figura 14: Mockup da Página de um Recurso Individual II

2.6.5 Página dos Utilizadores

The mockup displays a user interface for managing users. At the top, there is a navigation bar with links for 'Home', 'Recursos', 'Utilizadores', 'Perfil', and 'Logout'. A search bar is located at the top right. Below the navigation, a grid of user profiles is shown in three rows of three columns each. Each profile card contains a user icon, the user's name, their status ('Nome do Utilizador'), their category ('Estatuto'), their affiliation ('Filiação'), and their registration date ('Registado desde'). A 'Ver mais' link is at the bottom of the grid.

Nome do Utilizador	Estatuto	Filiação	Registado desde
Nome do Utilizador	Estudante	Escola X	02-02-2021
Nome do Utilizador	Docente	Escola Y	04-04-2021
Nome do Utilizador	Estudante	Escola X	31-01-2021
Nome do Utilizador	Trabalhador	Empresa X	10-03-2021
Nome do Utilizador	Estudante	Universidade X	02-02-2021
Nome do Utilizador	Estudante	Universidade Y	02-12-2020
Nome do Utilizador	Docente	Escola X	12-02-2021
Nome do Utilizador	Estudante	Escola X	02-03-2021
Nome do Utilizador	Estudante	Escola Z	02-02-2021

Figura 15: Mockup da Página dos Utilizadores

3 Implementação

3.1 Frontend

Para o desenvolvimento da **camada de apresentação**, como foi mencionado antes, optamos por **Vue**. É independente das outras camadas logo, vai utilizar pedidos à API para obter os dados que necessita apresentar ao utilizador.

De modo a tornar interface mais apelativa, foi utilizado **Vuetify**, uma das bibliotecas de vue mais utilizadas atualmente, que fornece uma grande quantidade de componentes para a construção de uma aplicação cativante para o utilizador.

Ao longo do projeto, também foram necessárias outros *packages* extra para certas funcionalidades. Por exemplo, no caso do download dos recursos foi necessário o *package* **js-file-download**.

3.1.1 Views

Relativamente às *views* foram criadas 13 diferentes que englobam todo o conteúdo disponível ao utilizador. Para além disso, foi criado um componente para a barra de navegação (*Navbar*) que está incluído em todas as views acima mencionadas.

Se o utilizador não estiver loggado na plataforma, apenas se depara com uma página inicial e com a opção de dar login na Navbar. Assim, quem efetua o login, já tem acesso ao resto das páginas existentes.

3.1.2 Pedidos à API

Para efetuar os pedidos à API utilizamos o *package* **axios**, que é baseado em promessas do tipo HTTP, cujo método pode ser do tipo POST, GET, etc. Como exemplo, temos de seguida o código correspondente a um pedido à base de dados para fazer o download de um recurso:

```
axios({
  method: "post",
  url: "http://localhost:8081/api/resource/download/",
  data: json,
  responseType: 'blob',
})
.then(response => {
  let fileName = Date.now() + ".zip"
  FileDownload(response.data, fileName)
})
.catch(err => {
  console.log(err)
  alert('Não foi possível realizar o download')
})
```

3.2 Backend

A back-end é a entidade que governa o fluxo entre os pedidos vindos da front-end que exigem acessos à camada de persistência, atuando, assim, como um servidor de API REST.

Esta, tal como mencionado anteriormente, tira partido da *framework* Spring para Java, que permite a criação de servidores aplicacionais de forma eficaz e robusta, que é uma das características mais relevantes para uma aplicação com os requisitos mencionados. Para o mapeamento de objetos Java em tabelas relacionais, é utilizada a *framework* Hibernate, tal como foi mencionado anteriormente.



Esta aplicação foi desenvolvida de modo a ser totalmente *stateless*, de modo que o processo de autenticação de um utilizador é realizado através de JSON Web Tokens (**JWTs**), isto é, sempre que um utilizador faz *login* no front-end, é feito um pedido á *back-end* pelo seu token correspondente, que será guardado como *cookie* no seu *browser* e é **único e assimétrico**, garantindo assim a integridade e o não repúdio de pedidos.

Consequentemente, todos os pedidos que envolvam informação crítica de um dado utilizador, terão de ir assinados com o *token* do mesmo e validados no *back-end* para garantir que é, de facto, um pedido não malicioso.

3.2.1 Controllers (Servlets)

Os *controllers* são *web servlets* que atuam sobre as rotas **REST** definidas para cada um deles.

As vantagens dos *Servlets* já foram exploradas durante o decorrer do semestre, mas essencialmente contribuem para a escalabilidade, dado que criam uma *thread* por pedido e não por cliente, e são responsáveis apenas por esse pedido em concreto, levando ao isolamento de *garbage collection* e recursos utilizados.

Cada uma destas rotas, no entanto, realiza operações que necessitam acessos à base de dados. Este processo rapidamente esgotaria as sessões permitidas em Hibernate (16-32, tipicamente) devido à criação de inúmeras *threads* cada uma a tentar obter uma das sessões disponíveis. Assim, é necessário a utilização dos Enterprise Java Beans, explorados na secção seguinte.

Por fim, assumindo que um pedido é bem sucedido, é retornado ao *front-end* dados sob a forma de **JSON**, pelo que é facilmente utilizado pelo Vue pela sua forte integração com JavaScript.

3.2.2 Enterprise Java Beans

Os Enterprise Java Beans são, também, uma *framework* com orientação á escalabilidade na medida em que permitem a utilização de um conjunto finito de objetos (Beans) evitando a instanciação desmedida de objetos e as consequentes *memory leaks* que surgem, assim como controlo de concorrência, entre outros.

Para os efeitos da aplicação serão utilizados Beans locais (não existem serviços remotos utilizados pela aplicação) e *stateless*, pelas razões vistas anteriormente, isto é, os pedidos são sempre desacoplados da noção de sessão.

Atualmente, existem os seguintes Beans na aplicação:

- **UserBean**: gera todos os acessos a operações que envolvam a edição de utilizadores, exceto a inserção ou edição da imagem de perfil.
- **UpdateBean**: gera todas as notícias dentro da aplicação.
- **PostBean**: gera todas as operações sobre publicações.
- **ResourceBean**: gera todas as operações sobre a inserção de recursos, exceto a criação de ficheiros no sistema operativo.
- **FileSystemBean**: gera todas as operações que envolvam *system calls* tais como guardar ou ler ficheiros.

Como referido anteriormente, os Beans são um número limitado de objetos geridos pela própria *framework*. Estes são acedidos pelos *Servlets*, pelo que o problema das sessões Hibernate é mitigado porque deixam de ser as possíveis milhares de *threads* a tentarem obter uma sessão, são agora esses Beans a tentarem a mesma operação, pelo que a performance geral da aplicação melhora de forma exponencial.

3.2.3 Persistência

A *framework* Hibernate permite o mapeamento simplificado de objetos Java com bases de dados relacionais, transformando tabelas persistentes em objetos que podem ser manipulados em *runtime*.

Consequentemente, foi utilizada uma base de dados relacional, mais concretamente MySQL, devido à sua adequação à aplicação em concreto visto que disponibilizava todas as ferramentas necessárias à sua elaboração.

O mapeamento disponibilizado pelo Hibernate, gerado utilizando Visual Paradigm, permite a utilização da **JPA** de forma mais simples e direta, para além do facto de que permite a manipulação direta de objetos em memória e essas alterações são imediatamente visíveis na base de dados, o que não é possível utilizando outras estratégias mais primitivas.

4 Deploy

4.1 Considerações Iniciais

A arquitetura simplificada inicial da aplicação é inviável devido à existência de múltiplos pontos de falha, apesar da modularidade da aplicação. Por exemplo, a falha do servidor singular de *front-end* levaria ao colapso da restante aplicação visto que o utilizador não interage diretamente com o *back-end*, muito menos com a base de dados.

Assim, após várias iterações durante a fase de desenvolvimento, foi obtida a seguinte arquitetura:

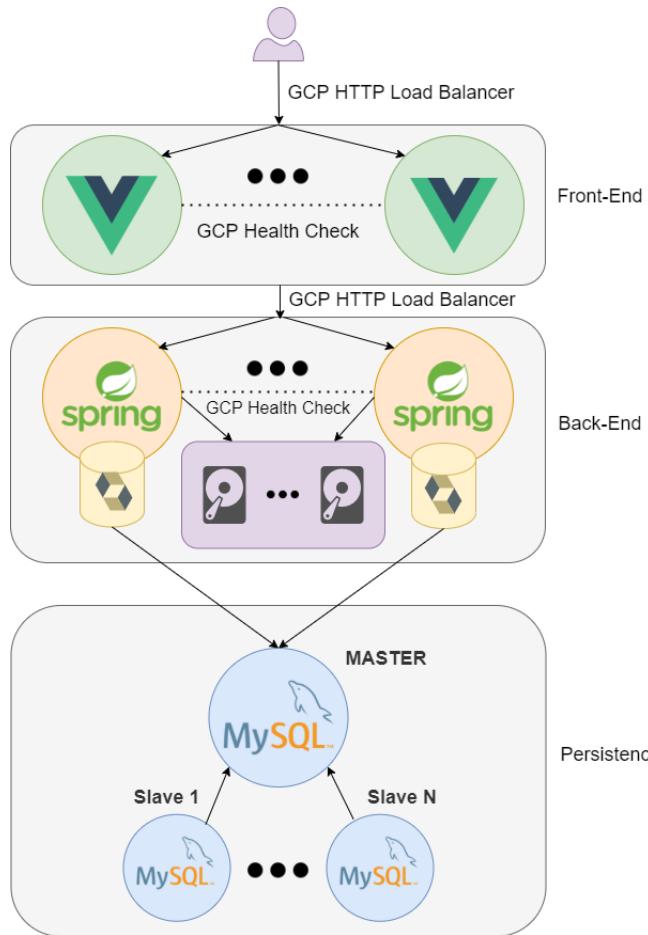


Figura 16: Arquitetura do Deployment

Essencialmente esta arquitetura garante, para cada camada, um número arbitrário de elementos, que será vista com mais detalhe nas secções seguintes.

4.2 Frontend

O front-end, tal como já foi visto, é um servidor em Vue.js, totalmente *cliente-sided* e *stateless*.

Estas características são ideais para a colocação de várias instâncias dos mesmos sob um balanceador de carga, que distribui o tráfego por várias instâncias e, como não há estado a ser gravado, é totalmente independente do servidor que acede.

O cliente, assim, acederia ao *front-end* por um ponto de acesso único, que é o IP fornecido pelo balanceador de carga, e a existência de servidores replicados seria totalmente transparente.

4.3 Backend

O *back-end* utiliza tokens JWT para realizar autenticação, assim, não é necessário guardar estados de sessão.

O *back-end* seria semelhante dado que também é *stateless*, isto é, existiria um balanceador de carga que garantia que o tráfego não era afunilado num só servidor e o servidor *front-end* apenas acederia a um deles, em cada pedido, escolhido pelo algoritmo do balanceador da Google.

No entanto, há o caso de ter de guardar recursos que o utilizador dá upload. Para este efeito, é utilizado um conjunto de discos partilhados entre as instâncias dos servidores com protocolos de replicação, para garantir que os dados não são perdidos no caso de falha de um disco e há consistência de informação.

4.4 Persistência

Uma desvantagem das bases de dados relacionais é não possibilitarem a escalabilidade horizontal, isto é, não permitem mais que uma instância da mesma base de dados a correr em simultâneo em máquinas diferentes, devido às transações sobre tabelas.

A camada de persistência tira partido da base de dados relacional MySQL, bastante popular e com bastantes funcionalidades, sendo que uma delas a configuração *Master-Slave*, que é utilizada nesta arquitetura, visto que tal como dito anteriormente, não é possível ter mais que uma instância da mesma base de dados a correr.

Esta primitiva permite a criação de um servidor principal, *Master*, com várias réplicas, *Slaves*, que funcionam apenas em modo leitura. As réplicas, nesta configuração, só servem para aumentar a velocidade de leitura devido a mecanismos de *caching* e evitar sobrecarregar o servidor *Master*.

4.5 Resultados e Problemas

A equipa docente forneceu créditos para utilização na Google Cloud Platform, que permite a instanciação de máquinas virtuais (que atuarão como as várias entidades do sistema) e, entre outras funcionalidades,平衡adores de carga.

A Google Cloud, devido aos créditos ilimitados, não permite um grande número de máquinas simultaneamente ligadas, pelo que o grupo teve de ter algumas considerações extra quanto ao limite de máquinas ativas.

No caso do *front-end*, foram criadas apenas 2 instâncias, pelo que para efeitos de produção não é viável, seria necessário um número mais elevado de instâncias.

No caso da camada de persistência, foi implementada esta configuração *Master-Slave* com um mestre e uma réplica, pelo que o aumento crescente de instâncias não implica necessariamente uma melhor performance (só são caches para leitura), no entanto seriam necessárias mais que uma réplica para ambientes de produção.

No *back-end*, a implementação final é uma instância individual do mesmo. Durante o desenvolvimento foram surgindo problemas com a replicação de discos e discos partilhados entre instâncias de máquinas virtuais, pelo que não foi possível obter a solução desejada. O *back-end* é, portanto, o ponto de falha mais evidente da aplicação visto que é uma só instância com um só disco, sem replicação.

Esta arquitetura foi a utilizada para efeitos de teste, que serão analisados na secção seguinte.

5 Análise de Carga

5.1 Considerações Iniciais

Um dos principais requisitos desta aplicação é o upload de recursos na plataforma por parte dos utilizadores.

Tendo isto em conta, o teste mais relevante que pode ser feito à aplicação é um teste de carga para descobrir o limite de utilizadores que, em simultâneo, podem realizar o upload dos recursos.

Com estes dados, sabendo que esta operação é também das mais custosas para o servidor de back-end devido à quantidade de *system calls* que tem de executar, obtemos uma boa aproximação do limite de utilizadores que a aplicação suporta em situações de carga muito elevada.

5.2 Testes de Carga

Para realizar os testes, foi utilizado o software Apache JMeter, que permite a execução automática de testes de carga sobre uma aplicação arbitrária.

A tabela seguinte mostra o número selecionado de utilizadores (100, 500, 1000, 2000) a realizarem o upload de um recurso com um único ficheiro relativamente pequeno (tamanho 10KB).

Clientes	Average	Min	Max	Std. Dev.	Error %	Throughput
100	1939	492	3138	897.77	0.00%	28.02691
500	7662	4645	15661	2471.77	2.40%	31.44456
1000	5518	260	12824	3945.75	14.40%	74.30525
2000	18433	263	53717	18157.02	20.80%	36.05488

Figura 17: Testes de Carga

Reparamos que, na configuração com um único *back-end*, a aplicação exibe alta disponibilidade (Error 1%) para um número de clientes a rondar os 100.

Para 500 clientes, a disponibilidade continua a ser elevada mas já causa uma taxa de erros superior, embora aceitável.

Com números a rondar os 1000 clientes, uma taxa de erro superior a 10 % leva a uma aplicação demasiado instável, pelo que não é viável para este número total de clientes.

Assim, conclui-se, que em situações de carga bastante elevada, a aplicação consegue lidar com cerca de 100 a 500 utilizadores.

6 Interface Final

6.1 Página Inicial



Figura 18: Página Inicial

6.2 Página Inicial - Login

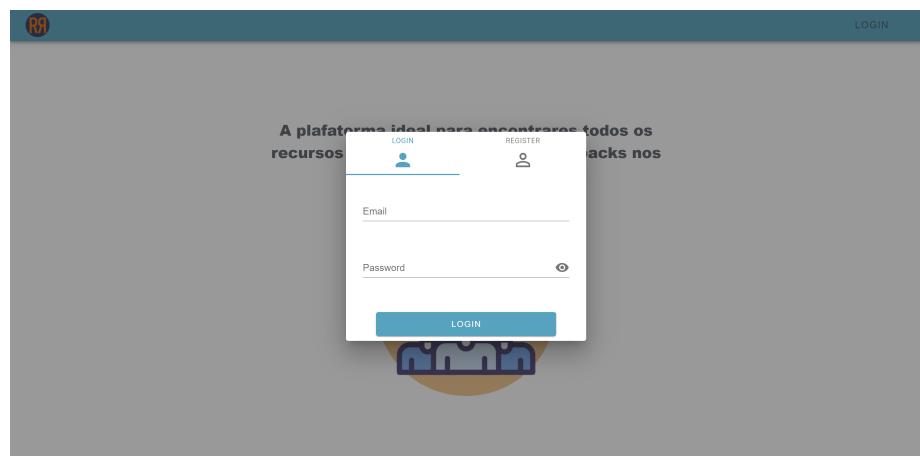


Figura 19: Página Inicial de Login

6.3 Página Inicial - Registo

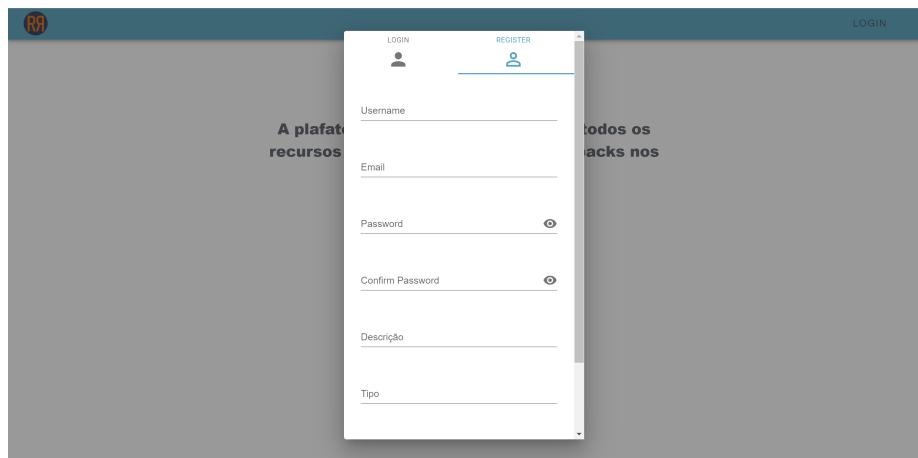


Figura 20: Página Inicial de Registo

6.4 Home

Publicações	Novos Recursos
<p>Bom relatório! Recurso: Relatório SDLE 2021 Alfredo Matos há 13 minutos</p>	<p>Classe Tipo Java Spring Estado: Disponível Tipo: Aplicações Adelina Lena há 12 minutos</p>
<p>Poucas bandeiras Recurso: Bandeiras de Países Alfredo Matos há 19 minutos</p>	<p>Aplicações em Python sobre Grafos Estado: Disponível Tipo: Aplicações Gloria Amália há 13 minutos</p>
<p>Material de apoio para o exame de SI Recurso: Testes Sistemas Interativos João Costa há 19 minutos</p>	<p>Políticos Portugueses Estado: Disponível Tipo: Imagens Gloria Amália há 15 minutos</p>
<p>Bons Testes Exemplares Recurso: Testes</p>	

Figura 21: Página Inicial com login efetuado

6.5 Utilizadores

The screenshot shows a list of users with their profile pictures, names, titles, and registration details:

- Admin**
Estatuto: admin
Filiação: admin
Registado desde 2021-01-10
- Váller Carvalho**
Estatuto: Aluno(a) de Mestrado
Filiação: Universidade do Minho
Registado desde 2021-02-01
- João Cunha**
Estatuto: Aluno(a) de Mestrado
Filiação: Universidade do Minho
Registado desde 2021-02-10
- Filipa Santos**
Estatuto: Aluno(a) de Mestrado
Filiação: Universidade do Minho
Registado desde 2021-01-02
- Hugo Cardoso**
Estatuto: Aluno(a) de Mestrado
Filiação: Universidade do Minho
Registado desde 2021-02-01
- Carlinhos Adelaide**
Estatuto: Aluno(a) de Licenciatura
Filiação: Universidade do Minho
Registado desde 2021-02-10
- Fernanda Urbano**
Estatuto: Aluno(a) de Licenciatura
Filiação: Universidade do Minho
Registado desde 2021-02-01
- Glória Amália**
Estatuto: Aluno(a) de Licenciatura
Filiação: Universidade do Minho
Registado desde 2021-02-13
- Ademar Marta**
Estatuto: Empregado(a)
Filiação: Bosch
Registado desde 2021-02-15

VER MAIS

Figura 22: Página de utilizadores

6.6 Recursos - Search

The screenshot shows a list of resource types under the heading "Tipo de Recursos".

- Acetatos
- Aplicações
- Artigos
- Atas
- Imagens
- Livros
- Outros
- Problemas Resolvidos

[Ver Todos](#)

Figura 23: Página de escolha de recursos por tipo

6.7 Recursos

	Título	Tipo	Autor	Classificação	Nº de downloads	Data de modificação	DOWNLOAD
<input type="checkbox"/>	Classe Tipo Java Spring	Aplicações	Adelina Lena	☆ ☆ ☆ ☆ ☆	0	2021-06-15	+ PUBLICAÇÃO
<input type="checkbox"/>	Aplicações em Python sobre Grafos	Aplicações	Gloria Amália	☆ ☆ ☆ ☆ ☆	0	2021-06-15	+ PUBLICAÇÃO
<input type="checkbox"/>	Políticos Portugueses	Imagens	Gloria Amália	☆ ☆ ☆ ☆ ☆	0	2021-06-15	+ PUBLICAÇÃO
<input type="checkbox"/>	Resolução de Teste de SDLE	Problemas Resolvidos	Gloria Amália	☆ ☆ ☆ ☆ ☆	0	2021-06-15	+ PUBLICAÇÃO
<input type="checkbox"/>	Acetatos SDLE	Acetatos	Carlinhos Adelaide	☆ ☆ ☆ ☆ ☆	0	2021-06-15	+ PUBLICAÇÃO
<input type="checkbox"/>	Acetatos de SI	Acetatos	Carlinhos Adelaide	★★★★★	0	2021-06-15	+ PUBLICAÇÃO
<input type="checkbox"/>	CSVs de Teste	Outros	Váter Carvalho	★★★★★	2	2021-06-15	+ PUBLICAÇÃO
<input type="checkbox"/>	Repositório GIT Readme	Repositórios	Filipa Santos	★★★★★	1	2021-06-15	+ PUBLICAÇÃO
<input type="checkbox"/>	Servlet Simples	Aplicações	João Cunha	☆ ☆ ☆ ☆ ☆	1	2021-06-15	+ PUBLICAÇÃO
<input type="checkbox"/>	Testes Sistemas Interativos	Testes e Exames	Váter Carvalho	★★★★★	2	2021-06-15	+ PUBLICAÇÃO

Figura 24: Página dos recursos

6.8 Adicionar Recurso

Figura 25: Adicionar um recurso

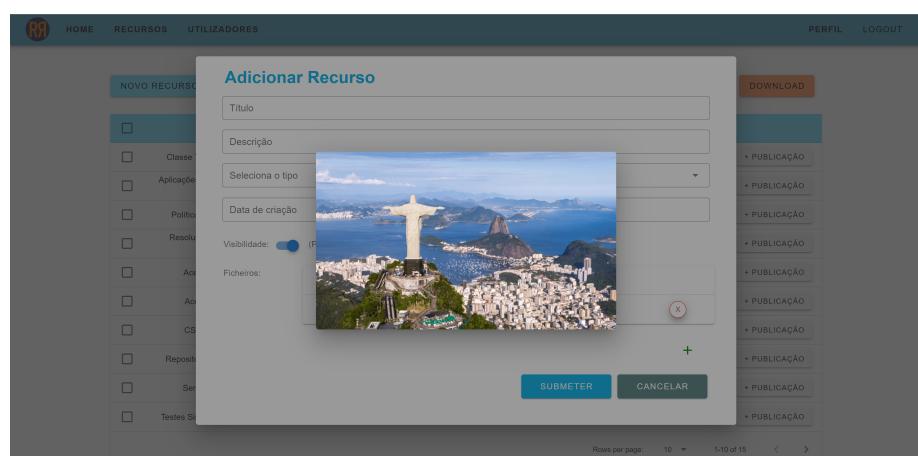


Figura 26: Preview de um ficheiro que será adicionado

6.9 Recurso

Testes Sistemas Interativos

Ver Publicações

Informação Visualizar

Tipo: Testes e Exames **Autor:** Valter Carvalho

Descrição: Testes de Sistemas Interativos
16/17 **Última Modificação:** 2021-06-15

Data de Criação: 2021-06-15 **Número de publicações:** 2

Data de Registo: 2017-05-10 **Número de downloads:** 2

Classificação:

Tamanho total: 871.7 KB

CLASSIFICAR **+ PUBLICAÇÃO**

Figura 27: Página do recurso

Testes Sistemas Interativos

Ver Publicações

Informação Visualizar

Nome	Tamanho	Visualizar
Teste 201617.pdf	808.2KB	
Teste Exemplo 2016.pdf	63.4KB	

DOWNLOAD

CLASSIFICAR **+ PUBLICAÇÃO**

Figura 28: Página do recurso

Teste de Sistemas Interativos

Módulo: Introdução à Engenharia Informática
2016/17

Duração prevista: 10:00

Lata teste com descrição.

Parte 1

Considerar que se pretende desenvolver um sistema de informação para gestão da Conferência. Considerar que existem pessoas que podem ser palestrantes ou moderadores. As sessões são controladas por apresentações. Um tutorial também pode estar disponível. Veja o resultado da figura abaixo.

```

graph TD
    Conference[Conference] -- "realiza em" --> Sessao[Sessão]
    Conference -- "realiza em" --> Palestra[Palestra]
    Conference -- "realiza em" --> Tutorial[Tutorial]
    Conference -- "realiza em" --> Evento[Evento]
    Téma[Téma] -- "é realizada por" --> Palestra
    Onador[Onador] -- "é realizada por" --> Palestra
    Data[Data] -- "é realizada por" --> Palestra
    Sessao -- "realiza em" --> Palestra
    Palestra -- "tem" --> Tutorial
    Palestra -- "tem" --> Evento
    Palestra -- "tem" --> Sessao
    Palestra -- "tem" --> Téma
    
```

Visualizar

CLASSIFICAR **+ PUBLICAÇÃO**

Figura 29: Preview dos ficheiros do recurso



Material de apoio para o exame de SI
Autor: João Costa
Publicado em 2021-06-15

Bons Testes Exemplares
Autor: João Cunha
Publicado em 2021-06-15

Figura 30: Publicações sobre o recurso

6.10 Adicionar Classificação

Informações

Tipo: Testes e Exames
Descrição: Testes de Sistemas Interativos
Data de Criação: 2021-06-15
Data de Registo: 2017-05-10
Classificação:

★ ★ ★ ★ ★

Figura 31: Nova classificação do recurso

6.11 Editar Recurso

Editar Recurso

Titulo: Servlet Simples

Descrição: Servlet Simples de Java

Selecionar o tipo: Aplicações

Visibilidade: (Público)

Ficheiros:

Nome	Tamanho	Visualizar
HelloWorld2.java	0.7KB	

+

GUARDAR CANCELAR

Figura 32: Editar o recurso

6.12 Adicionar Publicação



Figura 33: Nova publicação sobre o recurso

6.13 Publicação

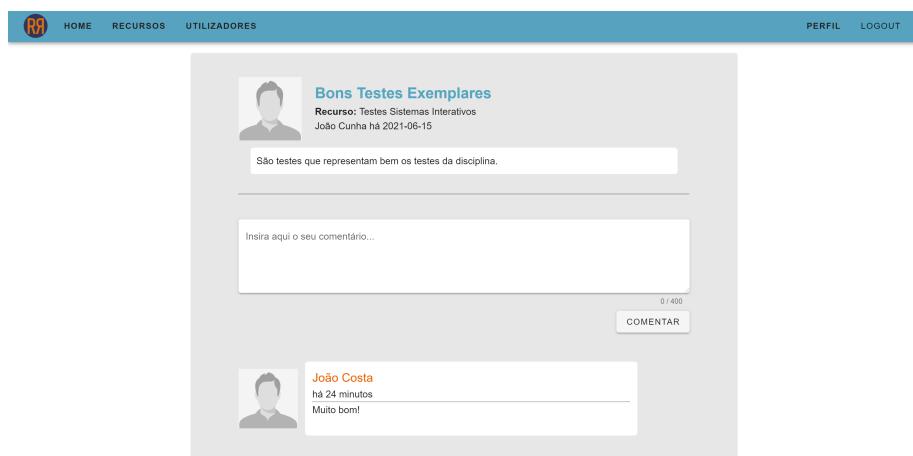


Figura 34: Página da publicação

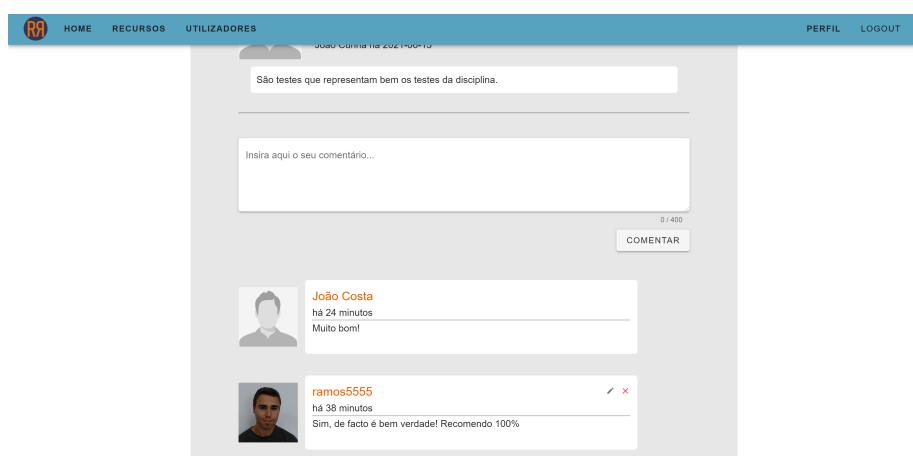


Figura 35: Continuação da página da publicação

6.14 Editar Publicação

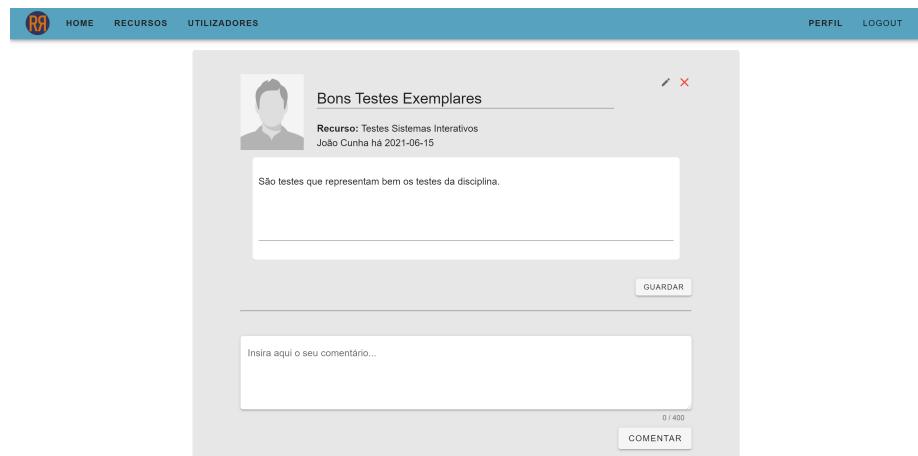


Figura 36: Editar a publicação

6.15 Perfil

João Cunha
Estatuto: Aluno(a) de Mestrado
Filiação: Universidade do Minho
Tipo: Aluno(a) de Mestrado
Descrição: Sou Aluno de MiEI
Registado desde 2021-02-10

- 15 Quinta-feira JULHO 2021

22:52 Adicionou um novo recurso

22:43 Adicionou um novo recurso

22:35 Fez uma nova publicação

Figura 37: Página do perfil

ramos555
Estatuto: aluno
Filiação: Universidade do Minho
Tipo: aluno
Descrição: Aluno do Mestrado de Engenharia Informática
Registado desde 2021-06-15

- 15 Quinta-feira JULHO 2021

23:13 Fez uma nova publicação

Figura 38: Página do perfil

6.16 Editar Perfil

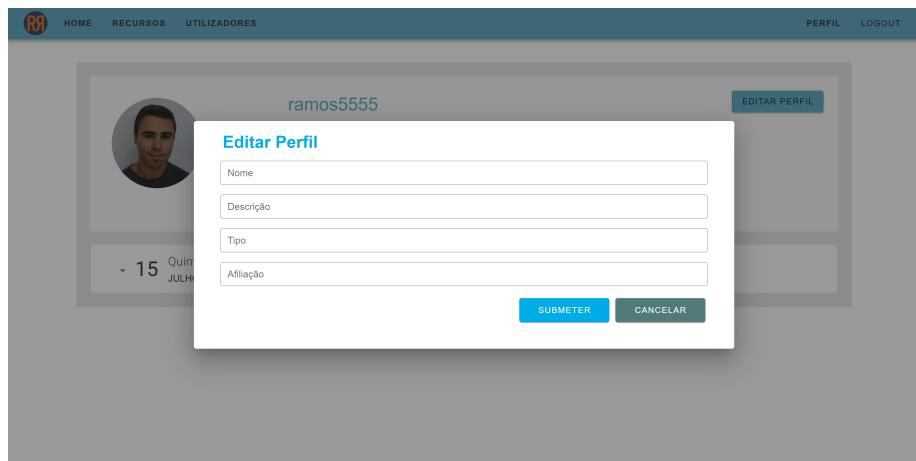


Figura 39: Editar o perfil

7 Análise de Usabilidade

7.1 Princípios de Usabilidade

7.1.1 Learnability

Este princípio baseia-se na facilidade com que um utilizador conseguem obter uma interação eficaz e alcançar a máxima performance nela, tendo as seguintes propriedades: **Predictability**, **Synthesizability**, **Familiarity**, **Generalizability** e **Consistency**.

Quanto à **Predictability**, é possível verificar no caso de inserção de um recurso (Figura 32) que o botão de "+" irá colocar um novo ficheiro. No caso da **Synthesizability**, quando um recurso é adicionado o utilizador é imediatamente redirecionado para a página, notando de imediato a alteração. Relativamente à **Familiarity**, no caso de registo (Figura 20), é um simples formulário que é presente em diversas aplicações. Por fim, **Consistency**, é sempre tido em conta na aplicação, seja em design como preenchimento de formulários, tais como o registo ou adicionar recursos.

7.1.2 Flexibility

A **Flexibility** diz respeito à multiplicidade de formas com que um utilizador e o sistema trocam informação, tendo as seguintes propriedades: **Dialogue Initiative**, **Multithreading**, **Task Migrability**, **Substitutivity** e **Customizability**.

Relativamente à **Dialogue Initiative**, o utilizador é quem controla o fluxo do programa, tal como é visível ao adicionar um recurso (Figura 25), onde o modal que bloqueia o ecrã é apenas mostrado em consequência das ações do utilizador. A **Multithreading** é respeitada na listagem de recursos (Figura 24), o utilizador consegue visualizar os recursos existente e filtrar ou ordenar pela ordem que lhe apetecer. Quanto a **Task Migrability**, não é aplicável a esta aplicação nem **Customizability** ou **Substitutivity** visto que as páginas são conteúdos estáticos fornecidos pelos servidores.

7.1.3 Robustness

Por fim, a **Robustness** refere-se ao nível de suporte fornecido ao utilizador de modo a atingir os seus objetivos, contendo as seguintes propriedades: **Observability**, **Recoverability**, **Responsiveness** e **Task Conformance**.

Quanto à **Observability**, é evidente pela lista de recursos que atua como estado do sistema, se esta está sempre atualizada então o utilizador consegue sempre saber o estado interno da aplicação. Relativamente à **Recoverability** devido à natureza da aplicação, não é aplicável. A **Responsiveness** é garantida pelo próprio Vue em todos os seus componentes, pelo que é uma característica sempre presente em todas as páginas. Por fim, **Task Conformance**, também não é aplicável ao sistema.

7.2 Heurísticas de Nielsen

7.2.1 Visibility of system status

A primeira heurística refere-se à **visibilidade do estado do sistema** no momento atual, de modo a manter o utilizador informado com o que está a acontecer.

Na Redu isso acontece quando se está a dar *load* da tabela dos recursos. Para comunicar ao utilizador que o sistema está naquele momento a recolher os dados requisitados, fica uma barrinha a correr, como podemos observar na seguinte figura.

Figura 40: Load dos recursos

7.2.2 Match between system and the real world

Também é necessário que a **aplicação web corresponda ao mundo real**, ou seja, que utilize linguagem familiar ao utilizador alvo. Isto é facilmente observado quando se atribui uma classificação a um recurso (Figura 31) dado que representar um *rating* através de estrelas é feito no mundo real (por exemplo, nos hotéis) e será facilmente compreendido por qualquer pessoa.

7.2.3 User control and freedom

Esta heurística defende que o **utilizador deve-se sentir em controlo e livre** ao navegar pela página web. Logo, deve-se fornecer saídas, como *undo* e *redo*, que o auxiliem.

Um bom exemplo isto é o menu de confirmação, ilustrado abaixo, que aparece quando o utilizador escolhe eliminar um recurso. A opção de cancelar dá-lhe a oportunidade de dar "undo" caso já não queira proceder com a eliminação.

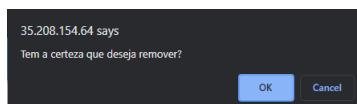


Figura 41: Confirmação ao apagar um recurso

7.2.4 Consistency and standards

Na construção de uma aplicação web, é importante que se **mantenha consistência e se respeitem os standards** existentes dentro da aplicação e relativamente a outros sites e aplicações já existente.

Em todos os menus onde existe o botão "Cancelar" (Figura 31, 32 e 33), este encontra-se sempre à direita e numa cor mais esbatida, o que é consistente em todas estas páginas bem como na maior parte das aplicações implementadas atualmente.

7.2.5 Error prevention

No que respeita **prevenção de erros**, a aplicação web deve ser capaz de prevenir uma grande quantidade de erros, que potencialmente podem causar conflitos no sistema.

Infelizmente, este é um dos pontos onde a aplicação está menos evoluída pois não implementa mecanismos deste género.

7.2.6 Recognition rather than recall

Já esta heurística refere que o utilizador **não tem que se esforçar para se lembrar de informação**, mas sim lembrar-se através de reconhecimento logo, não tem que se lembrar de informação de uma parte da interface para outra.

Um exemplo nisto na Redu é no menu de editar o recurso (Figura 32) onde a informação anterior permanece visível ao utilizador e assim, este pode editar os atributos sem ter que decorar os valores que tinha anteriormente.

7.2.7 Flexibility and efficiency of use

Para os utilizadores mais experientes, deve ser possível **acelerar a interacção** através de atalhos, por exemplo, garantido uma maior **flexibilidade e eficiência**.

Como se pode observar na Figura 23, oferece-se ao utilizador uma vista mais simples dos recursos, organizados por tipo, poupando o tempo de alguém que já saiba que tipo de recurso está à procura. Porém, devido à pequena dimensão da aplicação, não foram necessários atalhos.

7.2.8 Aesthetic and minimalist design

Em termos da interface gráfica, esta tem de ter um **design minimalista e estético**, onde a informação é apresentada de forma simples, sem conteúdo irrelevante apresentado.

Neste aspeto, a Redu cumpre os requisitos pois todas as páginas têm uma estética semelhante, simples e transmitem o que pretendem do utilizador de maneira evidente.

7.2.9 Help users recognize and recover from errors

Outro aspeto importante é ajudar os **utilizadores a perceberem e recuperarem de erros**, isto é, conseguir transmitir o erro ao utilizador de modo claro e direto e, se necessário, sugerir como avançar a partir daí.

Um exemplo disso nesta aplicação é no caso do login, quando o utilizador insere dados incorretos, como se pode verificar na figura seguinte.

The screenshot shows a login form with two tabs: 'LOGIN' and 'REGISTER'. Below the tabs are two user icons. The 'LOGIN' tab is active. Below the tabs is a red error message: 'Email ou Password incorretos!'. Below the message are two input fields: 'Email' containing '1' and 'Password' containing a redacted password. To the right of the 'Password' field is a visibility icon. At the bottom is a blue 'LOGIN' button.

Figura 42: Erro no login

7.2.10 Help and documentation

Finalmente, a última heurística de nielsen especifica que uma boa aplicação web tem de oferecer **ajuda e documentação** ao utilizador caso este tenha qualquer dúvida sobre o funcionamento da aplicação.

Devido à simplicidade da Redu e ao seu número limitado de funcionalidades, não houve a necessidade de implementar este aspeto na aplicação.

8 Conclusão e Trabalho Futuro

O desenvolvimento deste projeto permitiu a consolidação dos objetivos de estudo abordados nas aulas das unidades curriculares, nomeadamente web servers, load balancers, design patterns, prototipagem e métricas de usabilidade.

Foram utilizadas ferramentas novas como **Hibernate** e **Spring**, que se provarem ser muito eficazes, por exemplo, no mapeamento de tabelas para objetos, removendo processos *boilerplate*. Em suma, o grupo considera que foi um trabalho bem sucedido sendo que a plataforma faz o que foi estabelecido no início do desenvolvimento do projeto.

Como **Trabalho Futuro**, seria importante implementar **notificações** para o utilizador estar a par das publicações feitas sobre os seus recursos, bem como as classificações dadas, e também dos comentários feitos nas suas próprias publicações. Outra ideia seria a funcionalidade de **subscrever um recurso**, para que um utilizador conseguisse receber updates sobre atualizações feitas em recursos que está subscrito, bem como publicações feitas sobre este.

Por fim, seria também importante implementar perfis de **administrador**, que teria permissões adicionais à de um utilizador normal, de modo a poder regular todo o conteúdo da aplicação. Poderia apagar qualquer comentário, publicação, recurso ou utilizador que não se enquadrasse no contexto do projeto.