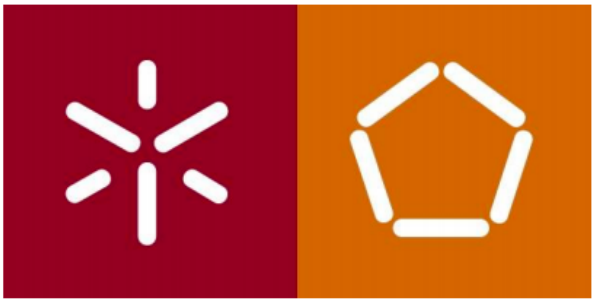

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Desenvolvimento de Aplicações WEB

myGoogleDrive

Eduardo Lourenço da Conceição (A83870)
Rui Nuno Borges Cruz Oliveira (A83610)

07/02/2021
Braga

Conteúdo

1 Introdução

Para o trabalho prático de DAW, no ano letivo de 2020/2021, foi-nos proposta a implementação de um sistema de consumo e produção de recursos. Com este intuito, iremos expor neste relatório o sistema que criamos, baseado no funcionamento do produto da *Google*, o *Google Drive*.

2 Arquitetura

Nesta secção iremos descrever a arquitetura do sistema.

2.1 Sistema

De um modo geral, podemos descrever a arquitetura do sistema da seguinte forma:

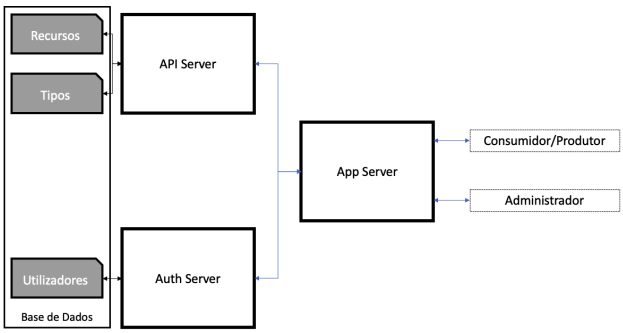

Figura 1: Arquitetura do sistema

Como podemos averiguar pela figura, o sistema comporta três servidores diferentes: um servidor de acesso aos dados dos recursos (**api-server**), um servidor de autorização dos utilizadores (**auth-server**) e um servidor de *front end* que trata dos pedidos dos clientes à aplicação (**app-server**). O *auth-server* e o *api-server* têm ambos acesso à base de dados (implementada em *MongoDB*), de forma a poderem aceder aos documentos que permitem a compleção das suas tarefas.

2.2 Utilizadores

Como podemos ver pela descrição da figura 1, e um pouco contra aquilo que se encontra no enunciado, existem dois tipos de utilizadores do sistema: **consumidores/produtores** e **administradores**.

- **Consumidores/Produtores**: Esta categoria de utilizadores pode fazer *upload* e *download* de conteúdo, bem como aceder ao seu conteúdo e ao conteúdo público, bem como ser notificado caso conteúdo público seja inserido na aplicação. A

decisão de não separar consumidores de produtores adveio do facto que, na nossa visão, o consumidor iria ter acesso a poucas *features*, tornando-o em pouco mais do que um produtor sem a capacidade de produzir, pelo que decidimos que seria melhor se as *features* dos dois tipos de utilizadores fossem combinadas num só.

- **Administradores**: Administradores têm as mesmas capacidades que um utilizador normal, isto é, um consumidor/produtor, com o acréscimo de ter direito a aceder a todos os recursos, não só os públicos, apagar recursos e aceder à lista de utilizadores e suas informações. Este tipo de utilizador **não se pode registar**, tem de ser registado manualmente na base de dados.

  Ambos tipos de utilizador são descritos pelo mesmo conjunto de informações.

3 Base de Dados

A base de dados do sistema, como foi dito anteriormente, foi implementada em *MongoDB*, e a sua comunicação com o restante sistema é feita utilizando *mongoose*. Existem três tipos de documentoes na base de dados: **recursos**, **utilizadores** e **tipos**.

- **Recursos**: descrevem um documento que foi inserido no sistema, tendo o seguinte esquema:
  - *email_produtor*: *Email* do utilizador que fez *upload* deste recurso;
  - *tipo*: Documento JSON referente ao tipo (a ver mais à frente);
  - *titulo*: Título do recurso;
  - *subtitulo*: Subtítulo do recurso (opcional);
  - *nome*: *path* para o recurso;
  - *dataReg*: data de registo do recurso;
  - *privacidade*: se o recurso é privado ou não;
  - *estrelas*: Avaliação agregada de um recurso, de 1 a 5, contendo a informação de quem fez uma *rating* do recurso;
  - *avaliacao*: Média da avaliação de todos os utilizadores;
  - *comentarios*: Lista de comentários feitos a um recurso, com o respetivo utilizador que fez cada um.
- **Tipo**: relativo ao tipo que um documento pode ter.
  - *descricao*: nome do tipo.
- **Utilizadores**: documento que contém toda a informação de um utilizador, e os seus atributos são os seguintes:
  - *email*: *E-mail* do utilizador, que é utilizado como chave primária;
  - *nome*: nome do utilizador;

- *password*: *password* do utilizador, que é cifrada aquando da sua inserção na base de dados;
- *filiacao*: filiação do utilizador que é algo que ele escolhe aquando do registo;
- *nivel*: "0"se for administrador, "1"caso contrário;
- *dataReg*: data de registo do utilizador;
- *dataUlt*: data da última vez que o utilizador entrou no sistema, sendo que é atualizada sempre que o mesmo ocorre;
- *notificacoes*: lista das notificações do utilizador.

4 Servidores

Nesta secção iremos explicar as várias interações que os diferentes tipos de servidores podem fazer.

4.1 API *Server*

Este servidor vai tratar dos pedidos relacionados com o armazenamento dos dados de um ficheiro na base de dados. De notar que não é este o servidor que vai armazenar o ficheiro em si, sendo que isso é feito por parte do *api-server*.

Outro aspeto que é importante mencionar é que não implementamos os requisitos relacionados com o *Submission* ou *Dissemination Information Package*, porque não conseguimos implementar as funcionalidades relacionadas com o mesmo a tempo.

Para o funcionamento correto deste servidor, e para ser um pouco diferente dos outros de modo a mostrar outra forma de implementar a troca dos *tokens*, o *token* da mensagem é recebido na *query string*. Sem este, os recursos não serão aceites e será devolvido um erro.

De modo a que possamos responder aos pedidos, foi desenvolvido um conjunto de rotas a que o servidor pode responder, que são as seguintes:

- GET "/recurso/:id": Obter o recurso com o identificador *id*;
- GET "/recursos/produzidos": Obter uma lista dos recursos de um utilizador;
- GET "/recursos/publicos": Obter a lista dos recursos públicos ou, no caso do pedido vir de um administrador, todos os recursos;
- GET "/recursos/anos": Listar os anos em que recursos foram produzidos;
- GET "/recursos/tipos": Listar os tipos de recurso disponíveis;
- GET "/recursos/ano/:ano": Lista os recursos de um dado ano;
- GET "/recursos/tipo/:tipo": Listar os recursos de um determinado tipo;
- GET "/recursos/sortAutor": Obter a lista de recursos ordenados pelo nome do autor;

- GET "/recursos/sortClass": Obter a lista de recursos ordenados por classificação;
- GET "/recursos/sortData": Obter a lista de recursos ordenados pela data de inserção;
- GET "/recursos/sortTipo": Obter a lista de recursos ordenados pelo nome do tipo;
- GET "/recursos/sortTitulo": Obter a lista de recursos ordenados pelo seu nome;
- GET "/recursos/com/:titulo": Obter a lista dos recursos com uma determinada *string* no seu nome;
- GET "/recurso/:id/comentarios": Obter os comentários de um recurso com o identificador *id*;
- POST "/recurso/addTipo": Adicionar um novo tipo de recursos;
- POST "/recurso/upload/:filename": Adiciona um recurso à base de dados;
- POST "/recurso/:id/comentario": Adicionar um comentário a um recurso;
- POST "/recurso/:id/avaliacao": Adicionar uma avaliação a um recurso;
- PUT "/recurso/alteraPrivacidade/:id/:priv": Alterar a visibilidade de um recursos com o identificador *id* para *priv*;
- PUT "/recurso/:id/avaliacao": Refazer uma avaliação a um recurso;
- DELETE "/recurso/apagaRecurso/:id": Apagar um recursos com o identificador *id*.

4.2 *Authorization Server*

O *auth-server* tem o papel de gerir a informação dos vários utilizadores, juntamente com a criação de um *token* para uma sessão para o utlizador. Este *token* tem duração máxima de três horas e encontra-se guardado nas *cookies* do cliente.

O servidor, por base, encontra-se na porta 8002, e todas as rotas começam por "/utilizadores/", uma vez que todos os *requests* feitos a este servidor são relativos a utilizadores.

Os pedidos que podem ser feitos têm de conter o *token* (na *query string*, no corpo da mensagem ou nas *cookies*) ou, caso seja para o *login* ou registo, este é gerado.

As rotas disponíveis são as seguintes:

- GET "/": para obter a lista dos utilizadores;
- GET "/notificacoes": para obter as notificações de um dado utilizador;
- POST "/login": para fazer o *login* na aplicação;
- POST "/registar": para adicionar um novo utilizador à BD e iniciar a sua sessão;

- POST "/notificacao/nova": para registar uma nova notificação, em todos os utilizadores;
- POST "/notificacao/apagar": para apagar uma notificação num utilizador quando este a marca como vista;

4.3 *Application Server*

Por fim, o servidor mais robusto é o servidor aplicacionar de *front end*, pois este tratará das *views* e dos pedidos diretos vindos dos utilizadores.

As rotas deste servidor estão dividas em dois: as dos utilizadores, com o prefixo "/users/", e as rotas normais, que não têm nenhum prefixo em particular.

- Rotas *Index*:
  - GET "/": Leva para a página de registo/*login*.
- Rotas *User*:
  - GET "/home": Leva para a página principal do utilizador;
  - GET "/recursos/com": Obter os recursos com uma dada string no título, e redirecionar para a página que os mostra;
  - GET "/meusRecursos": Obter a lista com os meus recursos e mostrar a mesma;
  - GET "/outrosRecursos": Obter a lista com os recursos públicos de outros utilizadores;
  - GET "/recursosAno/:ano": Obter os recursos publicados num dado ano;
  - GET "/recursosTipo/:tipo": Obter os recursos de um dado tipo;
  - GET "/recurso/:id": Mostrar a página de um recurso;
  - GET "/sortAutor": Obter a lista dos recursos ordenadas pelo nome do autor;
  - GET "/sortClass": Obter a lista dos recursos ordenadas pela classificação;
  - GET "/sortData": Obter a lista dos recursos ordenadas pela data de inserção;
  - GET "/sortTipo": Obter a lista dos recursos ordenadas pelo nome do tipo;
  - GET "/sortTitulo": Obter a lista dos recursos ordenadas pelo nome;
  - GET "/apagarNotificacao": Marcando uma notificação como vista, esta é apagada;
  - GET "/": Mostrar uma lista com os utilizadores (que só é acedida pelo administrador).
  - POST "/registar": Registo de um utilizador;
  - POST "/login": *Login* de um utilizador;

- POST "/logout": *logout* de um utilizador, que cria uma nova *cookie* com validade de um milissegundo, tornando-a quase imediatamente inútil, de forma a não permitir que o utilizador volte logo a entrar na aplicação sem fazer *login*;
- POST "/upload": Para carregar um novo ficheiro na aplicação;
- POST "/download/:filename": Para descarregar um ficheiro;
- POST "/recurso/:id/comentario": Para carregar um novo comentário num determinado recurso;
- POST "/recurso/:id/avaliacao": Para classificar um determinado recurso;
- POST "/recurso/addTipo": Registar um novo tipo;
- POST "/alteraPrivacidade/:id/:priv": Alterar a privacidade de um dado recurso;
- POST "/apagarRecurso/:id": Apagar um recurso, que só o administrador pode fazer.

Tendo em conta todas as rotas que o *app-server* suporta, existem algumas *features* que são notórias e de implementação mais interessante. Dos serviços que a aplicação pode fazer, alguns são de destaque, nos nossos olhos:

- **Notificações**: para fazer as notificações, quando um utilizador carrega um novo recurso, é feita uma verificação da visibilidade do recurso. Caso este seja público, o *api-server* envia um pedido ao *auth-server* para adicionar uma nova notificação. Quando o *auth-server* recebe este pedido, vai inserir uma nova notificação em todos os utilizadores exceto o que fez a inserção do recurso. Desta forma, quando um utilizador que tem a notificação entrar na aplicação, terá a notificação na barra lateral da inserção do recurso.

- ***Rating***: para avaliar os recursos, o utilizador tem à sua disponibilidade 1 a 5 estrelas como avaliação. Quando um utilizador carrega em avaliar, é informado o *api-server* com o email do utilizador, bem como o número de estrelas que escolheu, sendo que este atualiza o recurso avaliado com a adição ou atualização da lista de avaliações feitas ao mesmo. Posteriormente, faz a média destes mesmos valores e define esta como a avaliação geral do recurso. Portanto, é assim possível ao utilizador aceder à classficação pessoal e geral de um recurso.

- ***Preview* de recurso**: na página pessoal de cada recurso é possível visualizar o documento em si, de modo a que o utilizador tenha a possibilidade de aceder ao conteúdo do ficheiro, sem necessitar de efetuar *download* do mesmo para a sua máquina previamente. Isto é possível recorrendo à *tag*. *embed*.

- **Pesquisa de recurso com base no título**: o utilizador a partir de uma barra de pesquisa pode escrever determinada *string*, que é, posteriormente, enviada ao *api-server*, que verifica e devolve os recursos que contém a *string* inserida no título, para serem disponibilizados através de uma tabela pelo *app-server*.

5 Conclusão e Trabalho Futuro

No que toca a trabalho futuro, a implementação do SIP e DIP seria imperativo, bem como a verificação dos pacotes inseridos no sistema. Esta é a única *feature* que foi pedida que o sistema não suporta, e temos pena não termos conseguido implementar a mesma. Mais ainda, poderíamos alterar alguns dos aspetos da UI de modo a serem mais agradáveis.

Tendo isto em conta, acreditamos que conseguimos fazer tudo o resto que nos foi proposto, adaptando alguns dos requisitos de modo a seguir um modelo que nós achássemos mais interessante. Como tal, acreditamos ter tido sucesso na tarefa proposta.