Technical Report - **Project Specification**

TPW - Segundo Projeto

Unidade Curricular:	TPW - Tecnologias e Programação Web
Data:	Aveiro, 22/12/2024
Students:	103070: Eduardo Lopes 104415: Inês Ferreira 113402: Hugo Ribeiro

Resumo: Este projeto foi inspirado no Olx, e trata-se de um website onde os utilizadores podem comprar e vender artigos temáticos de Star Wars. Este tem o objetivo de fornecer aos fãs do franchise, uma experiência boa, permitindo um maior número de funcionalidades possíveis ao utilizador.

O nome do projeto é **Outer Rim**.

Tabela de conteúdos:

Deployment

Tal como no projeto anterior, o backend encontra-se deploy no pythonanywhere. Este foi relativamente fácil de fazer, uma vez que já o tínhamos feito no projeto anterior e num dos guias práticos. O link do backend é:

https://odraude.pythonanywhere.com

O frontend encontra-se deploy no vercel. Este processo já não foi tão simples como o anterior, uma vez que tentamos fazer o deployment em quatro websites diferentes. O primeiro que tentamos foi o heruko. Este tinha um processo de criar conta bem extenso e pedia para associar um cartão antes de podermos criar uma app para dar deployment.

O seguinte foi o netlify. Este oferecia a possibilidade de fazermos deployment diretamente do código no nosso repositório github mas, infelizmente, estava a ter dificuldades a conectar-se ao github.

A terceira tentativa foi no github pages, onde conseguimos parcialmente ter sucesso. O único problema é que os links entre as diferentes páginas não estavam a dar. A navegação pela app era feita exclusivamente pelas urls no browser.

Por fim, tentamos o vercel. Este foi de longe o mais fácil de todos. Depois de criarmos conta, fizemos upload do código e só tivemos de seguir os passos. O próprio website já fornecia as diferentes maneiras de darmos build à nossa app. Assim, o frontend é acessível através do link:

https://outer-rim.vercel.app/

Arquitetura do Frontend

Desenvolvemos um frontend em Angular com foco na reutilização de componentes, garantindo uma interface modular e fácil de manter. A integração com o backend, desenvolvido em Django REST Framework (DRF), é feita através de serviços Angular que permitem uma comunicação eficiente e organizada entre as diferentes partes da aplicação. Esta abordagem facilita a gestão de dados, melhora a performance e promove a escalabilidade do projeto, assegurando uma experiência de utilizador fluida e consistente.

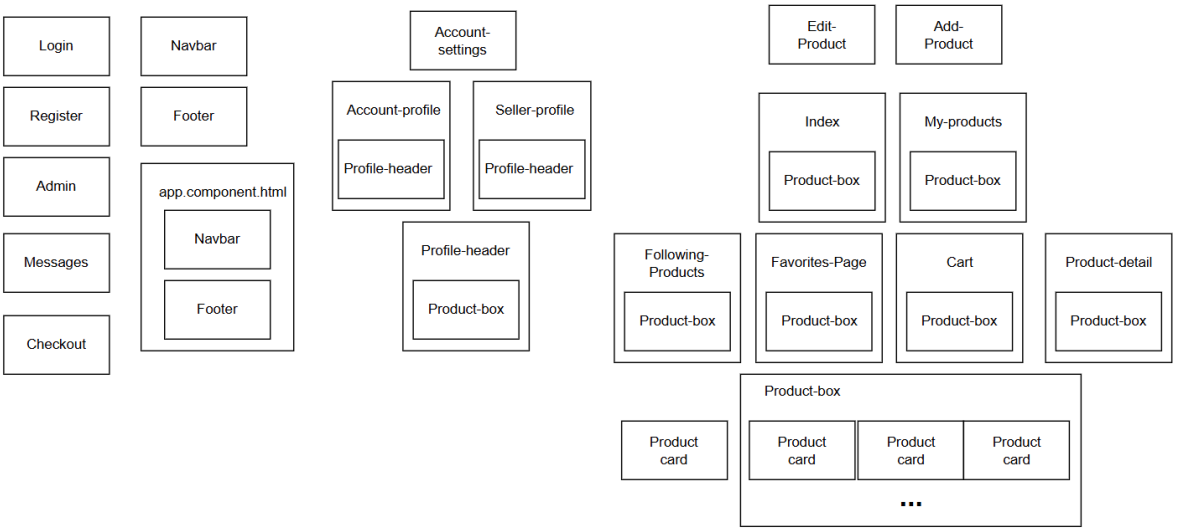

Figura 1. Componentes Angular e a sua interação

A Figura 1. mostra a estrutura do frontend da aplicação Angular, destacando a composição de componentes. As caixas no diagrama correspondem a diferentes componentes ou páginas da interface. Quando uma caixa está dentro de outra, isso indica que o componente externo contém e reutiliza o componente interno, refletindo a hierarquia e organização modular do projeto.

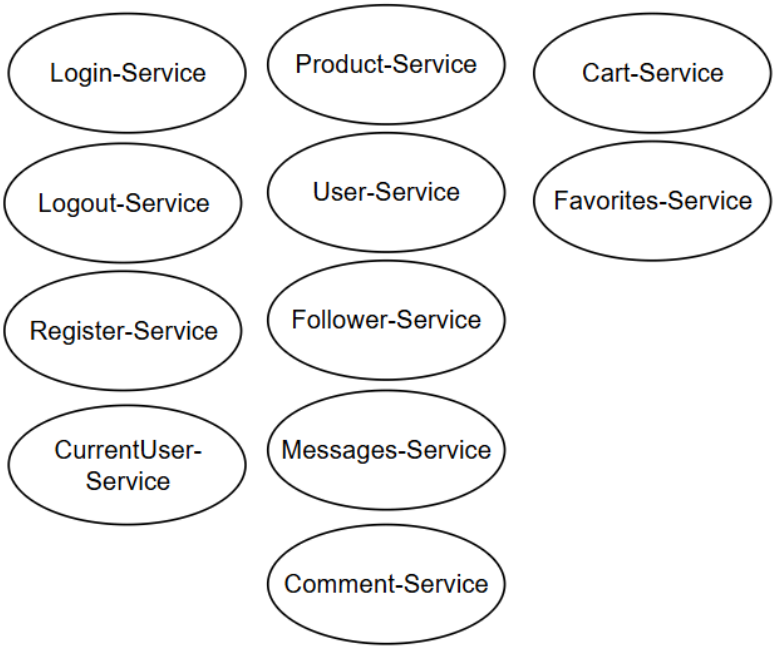

Figura 2. Servicos Angular

A Figura 2. representa os diferentes serviços Angular que fazem parte da aplicação. Cada elipse corresponde a um serviço específico.. Através deste diagrama, será explicado como cada um destes serviços interage com as diferentes páginas e componentes do frontend, assegurando a separação de responsabilidades e promovendo a reutilização de lógica em toda a aplicação.

- **Login-Service:** Este serviço interage com a página Login e serve para um utilizador autenticar-se.

- **Logout-Service:** Este serviço interage com Navbar, e serve para um user fazer logout.

- **Register-Service:** Este serviço interage com a página Register e serve para um utilizador criar uma conta.

- **CurrentUser-Service:** Este serviço interage com quase todas as páginas e serve para identificar o usuário atual da app, para processamento de dados e mesmo para passar a outros serviços para queries user-specific.
    - Usado para update das informações/imagen do current user.

- **Product-Service:**
    - Usado em Add-Product para adicionar um produto.
    - Usado em Admin para fetch de todos os produtos.
    - Usado em Edit-Product para ir buscar os detalhes de um produto e para aplicar as alterações quando este é editado.
    - Usado em Following-Products para ir buscar os produtos de um user seguido pelo current user.
    - Usado em Index para ir buscar todos os produtos que não estão vendidos.
    - Usado em My-Products para ir buscar todos os produtos de o current user.
    - Usado em Product-Card para implementar o botão “delete product”.
    - Usado em Product-Detail para ir buscar os detalhes de um produto e para recomendar produtos da mesma categoria ao current user.
    - Usado em Profile-Header para ir buscar os produtos do perfil em questão.

- **User-Service:**
    - Usado em Admin para is buscar todos os utilizadores.
    - Usado em Seller profile para ir buscar os dados de um determinado Seller.

- **Follower-Service:**
    - Usado em Account-Profile para ir buscar os followers do current user.
    - Usado em Profile-Header para ir buscar o numero de seguidores do current user e de pessoas que o current user segue, e para alterar o estado do botão follow/Unfollow

- **Messages-Service:**
    - Usado em Messages para ir buscar todas as mensagens que envolvem o current user e para o current user poder enviar mensagens.
    - Usado em Product-Detail para o current user poder enviar uma mensagem ao dono do artigo que está a ver.

- **Comment-Service:**
    - Usado em Seller-Profile para ir buscar todos os comentários de um determinado user e para o current user poder deixar um comentário.
    - Usado em Account-Profile para ir buscar todos os comentários do perfil do current user.

    - Usado em Admin para ir buscar todos os comentários.

- **Cart-Service:**
    - Usado em Product-Details para o current user adicionar um produto ao carrinho.
    - Usado em Product-Card para o current user poder remover produtos do carrinho.
    - Usado em Checkout para ir buscar todos os produtos do carrinho do current user e para este conseguir efetuar o checkout.
    - Usado em Cart para ir buscar todos os produtos do carrinho do current user.
    - Usado em Admin para ir buscar todos os Orders ja efetuados.
    - Usado em Account-Profile para ir buscar todos os Orders do current user.

- **Favorites-Service:**
    - Usado em Product-Card para definir o estado do botão Adicionar/Remover dos favoritos e também para implementar este botão.
    - Usado em Product-Detail para definir o estado do botão Adicionar/Remover dos favoritos e também para implementar este botão.

Arquitetura do Backend

O nosso backend é a app desenvolvida em django para o projeto anterior mas com o acréscimo do django Rest framework (as funcionalidades antigas e a interface continuam acessíveis através do link disponibilizado na secção do deployment). Sendo assim, para este projeto, acrescentamos novas views e urls para funcionar segundo o conceito de Rest API (estas encontram-se bem diferenciadas das anteriores no código através de comentários). A API é autenticada através do Token disponível no módulo authentication do django Rest framework. Assim, sempre que fazemos chamadas à API, basta incluirmos o Token (gerado no momento em que o utilizador cria conta) no header do pedido. No lado da API, as views possuem decorators associados que tratam da validação do token. Os decorators utilizados são os seguintes: **@authentication_classes([SessionAuthentication, TokenAuthentication])** e **@permission_classes([IsAuthenticated])**. O primeiro define os métodos que vão ser utilizados para autenticar a chamada à view associada. O segundo, garante que só utilizadores autenticados conseguem aceder à view associada aos decorators.

Este método foi nos aconselhado por alunos mais velhos que realizaram a cadeira em anos anteriores. Depois de uma pesquisa na documentação e na internet, achamos que era o melhor método para implementar a autenticação.

Relativamente aos endpoints da API, a tabela seguinte apresenta todos aqueles que implementamos juntos com uma breve descrição da sua função.

URl	Método	Descrição
/api/products	GET	Fetch da lista de produtos.
/api/login	POST	Fazer login.
/api/register	POST	Criar uma nova conta.
/api/user	GET	Fetch do utilizador atualmente logado.
/api/users	GET	Fetch de todos os utilizadores.
/api/user/comments/<int:id>	GET	Fetch de todos os comentários feitos no perfil de um utilizador.
/api/user/products/<int:id>	GET	Fetch de todos os produtos de um utilizador.

/api/user/following-products/<int:id>	GET	Fetch de todos os produtos de um utilizador seguido.
/api/products/<int:id>	GET	Fetch dos detalhes de um produto.
/api/<int:id>/followers	GET	Fetch dos seguidores de um utilizador.
/api/<int:id>/following	GET	Fetch dos utilizador que um utilizador está seguindo.
/api/user/<str:username>	GET	Fetch de um utilizador pelo nome de utilizador.
/api/addComment	POST	Adicionar um comentário ao perfil de um utilizador.
/api/<int:id>/followUser	POST	Seguir um utilizador.
/api/<int:id>/unfollowUser	POST	Deixar de seguir um utilizador.
/api/favorites/toggle	POST	Adicionar um produto aos favoritos
/api/favorites/is-favorite/<int:product_id>/<int:user_id>	GET	Verificar se um produto está nos favoritos
/api/products/<int:product_id>	DELETE	Eliminar um produto.
/api/favorites/<int:user_id>	GET	Fetch dos produtos favoritos de um utilizador.
/api/products/<int:product_id/update	PUT	Atualizar os detalhes de um produto.
/api/products/<int:product_id/updateImage	PUT	Atualizar a imagem de um produto.
/api/product/add	POST	Adicionar um novo produto.
/api/deleteAccount/<int:id>	DELETE	Excluir uma conta de utilizador.
/api/updateAccount/<int:id>	PUT	Atualizar uma conta de utilizador.
/api/updateProfile/<int:id>	PUT	Atualizar os detalhes do

		perfil de um utilizador.
/api/updateProfileImage/<int:id>	PUT	Atualizar a imagem do perfil de um utilizador.
/api/comments	GET	Fetch de todos os comentários.
/api/deleteComment/<int:id>	DELETE	Excluir um comentário.
/api/cart/<int:id>	GET	Fetch do carrinho de um utilizador.
/api/addToCart	POST	Adicionar um produto ao carrinho.
/api/removeFromCart	POST	Remover um produto do carrinho.
/api/messages/add	POST	Enviar uma mensagem para um utilizador.
/api/messages/<int:user_id>	GET	Fetch das mensagens de um utilizador.
/api/products/recommended	GET	Fetch dos produtos recomendados.
/api/checkout	POST	Comprar os produtos no carrinho.
/api/orders	GET	Fetch de todos os pedidos de um utilizador.
/api/orders/<int:id>	GET	Fetch dos pedidos de um utilizador específico.

Users e Funcionalidades

Nesta seção vamos apresentar os utilizadores criados para interagir com a nossa app e as funcionalidades que cada tipo de utilizador tem.

Começando então pelos os utilizadores, a nossa app tem conta criada para os seguintes

User	Password	Admin
joão	password1	False
maria	password2	False
ricardo	password3	False
ana	password4	False
tiago	password5	False
mateus	password123	True

As funcionalidades são diferenciadas para três tipos de utilizadores: utilizador sem login, com login e admin. As funcionalidades novas, em relação à primeira entrega, estão a negrito.

O utilizador sem login pode executar as seguintes funcionalidades:

- Login/Criar conta
- Ver todos os produtos disponíveis na app (visão mais geral dos produtos)
- Procurar/Filtrar por produtos

O utilizador com login pode fazer todas as funcionalidades anteriores mais as seguintes:

- Ver a página dos detalhes do produto
- Adicionar os seus próprios artigos para venda
- Ver os detalhes dos seus próprios artigos (consegue ver as vezes que aquele artigo foi visto por outras pessoas)
- Remover os artigos que está a vender
- Editar os detalhes dos artigos que está a vender (mudar preço, descrição ou foto do produto)
- Adicionar produtos de outros utilizadores ao carrinho e realizar a compra dos mesmos
- Seguir outros utilizadores (pode ver os produtos das pessoas que segue na página Following)
- Consegue ver na sua página de perfil as pessoas que o seguem

- Adicionar artigos aos favoritos (página dedicada para ver esses artigos)
- Pode enviar e receber mensagens de outros utilizadores (para começar uma conversa tem de ir à página dos detalhes de um artigo e mandar uma mensagem ao vendedor. A partir daí a conversa vai estar disponível na página dedicada para isso)
- Receber mensagens quando outros utilizadores compram um dos seus produtos
- Ver os perfis dos outros utilizadores
- Mudar o seu próprio perfil (pode alterar a foto de perfil, descrição da conta, nome, username, email e password)
- Adicionar comentários nos perfis dos outros utilizadores
- Ver os produtos que tem no carrinho
- Fazer logout
- **Ver o seu histórico de compras (na página do perfil)**
- **Ver recomendações para outros produtos da mesma categoria (na página de detalhes de um produto)**

O admin tem todas as funcionalidades descritas anteriormente mais as funcionalidades de gestão da app, que são as seguintes:

- Remover qualquer utilizador da app (isto remove automaticamente os produtos vendidos por esse utilizador e todos os comentários realizados nos perfis dos outros utilizadores)
- Remover qualquer artigo
- Remover qualquer comentário que não seja adequado
- Pesquisar por utilizadores, comentários e produtos (isto tudo na página do admin)
- Ver todos os artigos que já foram comprados na app

Conclusão

Concluímos que os objetivos do projeto foram alcançados, tirando o máximo partido das funcionalidades do Django Rest Framework e explorando extensivamente a reutilização de componentes no Angular. A integração entre frontend e backend foi implementada de forma eficiente. Consideramos que este projeto reflete um elevado nível de dedicação e compreensão das tecnologias envolvidas, como também a pesquisa e a aprendizagem autônoma que nos permitiram melhorar o mesmo.