

TQS: Product specification report

Rui Coelho[86182], Luís Costa[85044], João Laranjo[91153], Pedro Iglésias[89318]

v2020-06-05

1	Introduction	Erro! Marcador não definido.
1.1	Overview of the project	Erro! Marcador não definido.
1.2	Limitations	Erro! Marcador não definido.
2	Product concept	Erro! Marcador não definido.
2.1	Vision statement	Erro! Marcador não definido.
2.2	Personas	3
2.3	Main scenarios	Erro! Marcador não definido.
2.4	Project epics and priorities.....	3
3	Domain model	4
4	Architecture notebook	Erro! Marcador não definido.
4.1	Key requirements and constrains	Erro! Marcador não definido.
4.2	Architetural view	Erro! Marcador não definido.
4.3	Arquitetura de instalação	5
4.4	System architecture	Erro! Marcador não definido.
4.5	API for developers	6
5	API de integração	Erro! Marcador não definido.
6	References and resources	Erro! Marcador não definido.

1 Introdução

1.1 Visão do projeto

Projeto realizado no âmbito da disciplina de Testes e Qualidade de Software e tem como principais objetivos:

- Desenvolvimento de uma especificação de produto, desde os casos de uso até ao design técnico
- Propor, justificar e implementar uma arquitetura de software, com base nas estruturas empresariais;
- Aplicar práticas de trabalho colaborativo em equipas de desenvolvimento ágil assim como desenvolvimento contínuo e integrado. Apresenta-se 4Wheels, uma aplicação web e mobile que tem como objetivo possibilitar e facilitar a compra e venda de automóveis.

Posteriormente foram criados, quer o repositório git, quer o PivotalTracker (ferramenta utilizada para gestão de backlog), de modo a que estivessem reunidas todas as condições para se dar início ao trabalho.

1.2 Limitações

Todas as features em termos de software foram implementadas a única que não foi implementada, contudo, era almejada pela equipa, era a utilização de sensores para a monitorização do estado da build o mesmo não foi possível uma vez que não dispúnhamos do material necessário devido ao covid que nos impediu de deslocar a Aveiro.

2 Conceito do produto

2.1 Visão

O produto consiste numa apresentação da informação detalhada relativa a cada veículo. Desta forma deve ser possível a cada pessoa que utiliza a aplicação procurar e obter informação sem muita pesquisa e perda de tempo. Para além disso a venda de um automóvel pessoa deve ser também fácil para que quer empresas do setor que pessoas singular possam fazer e promover o seu negócio.

A aplicação será, então, usada para compra e venda de um produto, automóveis. O utilizador poderá:

- Pesquisar por carros
- Ver os favoritos de um utilizador
- Ver todos os carros disponíveis
- Ver informações de um carro

Este produto vai, assim, facilitar a vida dos seus utilizadores que passarão a ter uma interface, única, que junta um número enorme de automóveis possibilitando uma escolha mais fácil e informada.

A 4Wheels surgiu da necessidade de facilidade de controlo de dados e divulgação de informação.

2.2 Personas

António:

- O António tem 43 anos, mora em Braga, trabalha como gestor e adora novas tecnologias. É uma pessoa ponderada, atenta à atualidade, amigo do ambiente e poupado. O António viaja muito e tem um carro com alguns problemas, anda a pensar trocar de viatura mas precisa de se informar melhor.

João:

- O João tem 26 anos e vive em Aveiro. Trabalha numa empresa de software perto de casa e pretende usar um aglomerado de dados para desenvolver a sua aplicação. Como tal procura uma API com informações de carros.

Anabela:

- A Anabela é gestora de uma empresa automóvel. Gostava de expandir o seu negócio para o digital, como tal procura uma plataforma que a ajude com essa tarefa.

2.3 Cenários principais

Cenário 1:

- O António viu um modelo de um carro que gostou, tem um preço fantástico e tecnologia de ponta. Tendo ligação à internet, e autenticação feita, ao entrar na página principal pode procurar pelo carro que viu.

Cenário 2:

- O João pretende fazer uma aplicação para dispositivos móveis que apresenta vários modelos de carros assim como estatísticas e preços dos mesmos. Tendo ligação à internet, e autenticação feita, ao fazer um GET da API, recebe em formato JSON os dados que pretendia.

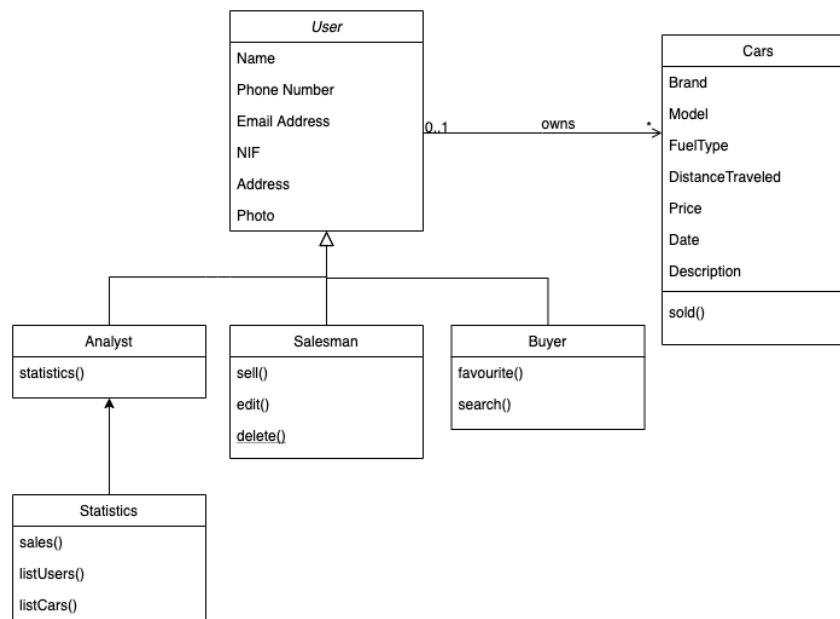
Cenário 3:

- A Anabela recebeu um novo modelo de um carro e os negócios têm corrido bem no entanto queria expandir o seu negócio para o digital para que possa ter uma maior margem de lucro. Tendo ligação à internet, e autenticação feita, a Anabela poderá colocar à venda os automóveis que acabaram de chegar.

2.4 Project epics and priorities

[Apresentar um plano indicativo para a implementação incremental da solução ao longo de várias iterações/releases, explicando as funcionalidades a atingir por [epics](#)]

3 Domain model



4 Arquitetura

4.1 Requisitos chaves e restrições

Principais requisitos:

- Os dados deverão ser atualizados e editáveis;
- Deve haver a possibilidade de promover produtos;
- A pesquisa deve ser o mais detalhada possível;

Restrições:

- Tempo para a realização do projeto é curto, devemos focar no mais importante.

Principais problemas:

- Como desenvolver o frontend para multiplataforma?
- Qual a melhor forma de armazenar e estruturar os nossos dados?
- Como apresentar os dados da API
- Como lidar com falhas?
- Devemos utilizar novas tecnologias?

4.2 Vista da arquitetura

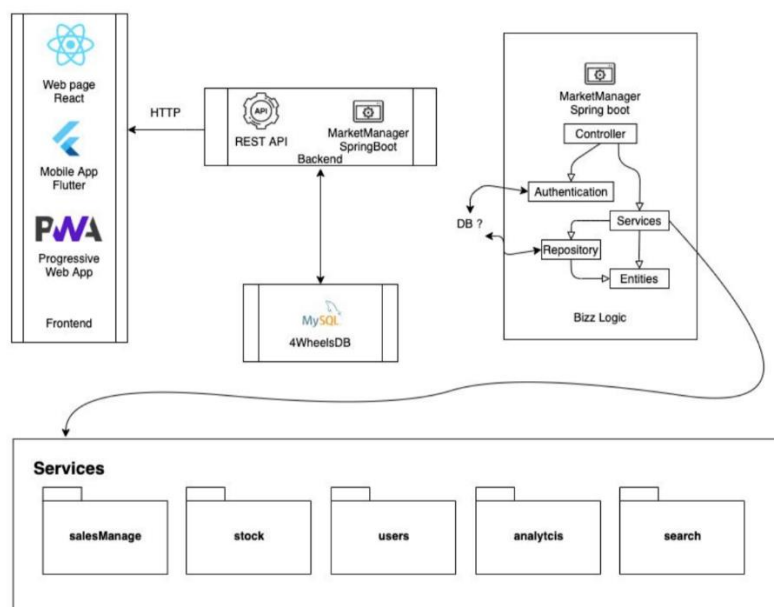
- **CLIENTE** - Dentro do cliente temos a nossa webApp, que essencialmente mostra ao utilizador os dados, estatísticas. Esta será programada em React e Flutter.
- **BACKEND** - Temos a business logic que categoriza e processa os diferentes dados (ex: user management, registos, etc). Temos também a exposição da API para outros utilizarem.
- **PERSISTÊNCIA** - Nesta componente temos uma base de dados de longo termo que irá ajudar a persistir os dados. Esta camada comunica diretamente com o backend daí estar incluído dentro da mesma, MySQL.
- **CONSUMER** – Aplicação móvel que consome os dados da API.

4.2.1 Interações modulares

De forma resumida deve ser possível aos utilizadores submeterem produtos para venda. Isto deve ser feito através do frontend e da mesma forma para o indivíduo ou para a empresa.

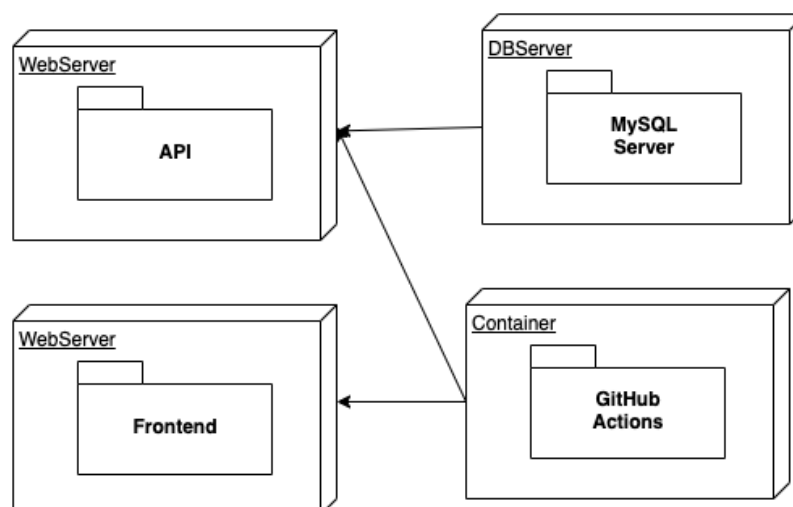
Uma API deve ser exposta/chamada através de endpoints específicos. A informação deve ser em formato JSON e deverá ser o mais detalhada possível.

Deve existir uma zona de persistência para guardar não só as interações, mas também as informações relativas a cada produto anunciado.



4.3 Arquitetura de instalação

Todo o processo de deployment tem o seu começo no servidor da Microsoft, no caso, o do Github Actions no qual todo o código é previamente testado, após isso, é feito o seu deployment para o heroku o qual se relaciona, no caso da API Spring, com a base de dados e o webserver do frontend e, no caso do frontend, o mesmo só se relaciona com a API Spring.



5 API for developers

A API encontra-se documentada com a utilização do swagger. A abordagem para a utilização destes endpoints consiste na tipologia de utilizador, no caso, neste sistema, podem ser de três tipos:

- Vendedor
- Comprador
- Analista

Consoante o tipo de utilizador a que a conta está associada a reação aos pedidos varia. As validações deste tipo são com recorrência a JWT [1]. À cabeça todos os paths estão protegidos com exceção dos que contêm a informação publica dos carros.

analytics-controller : Analytics Controller

Show/Hide | List Operations | Expand Operations

GET	/analytics/	List general analytics from the platform.
GET	/analytics/vendors/cars/registered	List the amount of cars vendors have registered on the platform.
GET	/analytics/vendors/cars/selling	List the amount of cars vendors have for sale on the platform.
GET	/analytics/vendors/cars/sold	List the amount of cars vendors have sold on the platform.

authentication-controller : Authentication Controller

Show/Hide | List Operations | Expand Operations

POST	/authenticate	createAuthenticationToken
POST	/register	saveUser

car-controller : Car Controller

Show/Hide | List Operations | Expand Operations

GET	/car/	Get all the cars on the database.
POST	/car/	Insert a car on the database.
GET	/car/brand/{content}	Search a car by brand.
GET	/car/fuel/{content}	Search a car by fuel.
GET	/car/model/{content}	Search a car by model.
PUT	/car/sold/{id}	Mark car as sold.
GET	/car/vendor	List all the cars of a certain vendor.
GET	/car/vendor/selling	List all the cars of a certain vendor.
GET	/car/vendor/sold	List all the cars of a certain vendor.
GET	/car/year/{content}	Search a car by year.
DELETE	/car/{id}	Remove a car from the database.
GET	/car/{id}	Get car details for a specific car.
PUT	/car/{id}	Edit car details for a specific car.

favourite-controller : Favourite Controller

Show/Hide | List Operations | Expand Operations

GET	/favourite/	Get favourite cars by user.
DELETE	/favourite/{id}	Delete favourite car by user.
POST	/favourite/{id}	Save a car in the user favourites list.

profile-controller : Profile Controller

Show/Hide | List Operations | Expand Operations

DELETE	/profile/	Delete a profile from the database.
GET	/profile/	Get profile info for a specific user.
POST	/profile/	Insert a profile on the database.
PUT	/profile/	Edit profile details for a specific user.

users-controller : Users Controller

Show/Hide | List Operations | Expand Operations

GET	/users/	List all the users registered on the platform.
GET	/users/{type}	List all the buyers/vendors registered on the platform given the type on the url.

6 Referências

- [1] “Spring Boot Security + JWT Hello World Example,” 05 Maio 2020. [Online]. Available: <https://www.javainuse.com/spring/boot-jwt>.