deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

TQS: Relatório de Especificação de Produto

Carolina Araújo [93248], Hugo Almeida [93195], Mariana Santos [93257], Miguel Almeida [93372]
v2020-05-09 to 2020-06-22

Introdução **1**
Visão geral do projeto 1
Limitações 4
Conceito do Produto **5**
Visão 5
Personas 8
Cenários principais 8
2.4 Project epics e Prioridades 9
Domain model **11**
Architecture notebook **12**
Key requirements e Restrições 12
Visão arquitetural 13
Arquitetura de deployment 14
API para os Developers **14**
Conclusão **18**
Referências e Recursos **18**

1 Introdução

1.1 Visão geral do projeto

TQS Delivery foi a empresa criada para garantir serviços de entrega a pronto para clientes em qualquer ponto do país. Esta empresa, no entanto, fornece a possibilidade de permitir que quaisquer lojas que pretendam usufruir do serviço de entregas de produtos, se possam conectar à sua API e registar-se para ficarem armazenados na base de dados. A partir daqui, qualquer loja associada tem a capacidade de permitir que os seus clientes façam encomendas e possam usufruir de entrega ao domicílio, se assim o desejarem, em vez de terem de se deslocar até à loja física mais próxima.

Figuras 1-4: Logótipos e icons TQS Delivery


HumberPeças, uma loja dedicada à venda de ferramentas e peças para construção, jardinagem e reparo de imobiliários/eletrônicos. A possibilidade de conseguir ajudar os seus clientes a facilmente poder adquirir ferramentas sem terem de sair do conforto da sua casa, e receber com mais pontualidade. Assim sendo, tendo já um website pronto, juntou-se ao serviço de **TQS Delivery**.

Figuras 5-7: Logótipos e icons Humber Peças

TQS Delivery permite que os estafetas associados à empresa possam registar uma conta, realizar o *login*, *logout* e aceder a uma aplicação móvel onde podem aceitar novas encomendas ainda pendentes (sem um estafeta associado) para ir buscar e entregar.
Por outro lado, os mesmos estafetas também têm acesso a uma página pessoal, *i.e* perfil, na plataforma destinada a *web browsers*. Aqui, poderão ver as últimas entregas que realizaram, a loja onde a compra foi feita, o nome do cliente que fez o pedido, o status da encomenda e a *review* que o cliente deu a essa entrega. Para além disto, os riders poderão também ver a quantidade total de entregas que já realizaram, a média de reviews de todas essas entregas e o número de clientes que o avaliaram.

Para garantir que os clientes do serviço de entregas têm direito ao melhor serviço possível, criaram-se postos de gestores de entregas e estafetas. Estes gestores, da TQS Delivery, têm a possibilidade de verificar todos os estafetas associados à empresa, a informação dos mesmos, por exemplo: nome, *average review* e número total de entregas realizadas. Têm acesso a um gráfico que mostra as 5 cidades onde um maior número de pedidos já foi realizado, bem como esse mesmo número; deste modo podem sempre considerar contratar mais estafetas numa zona bastante popular. Finalmente, podem, ainda sobre os estafetas, consultar o número total de estafetas registados, o tempo médio de entrega, a *average das reviews* de todos os estafetas e o número de encomendas que ainda faltam entregar.
Por outro lado, para garantir que vale a pena manter o serviço de entregas ativo e manter uma boa qualidade, os gestores também têm acesso às informações relativas a cada loja: nome, descrição e

número de pedidos feitos na mesma. Além disso, poderão também consultar o número médio de pedidos realizados por semana, o número total de pedidos e o número total de lojas associadas.

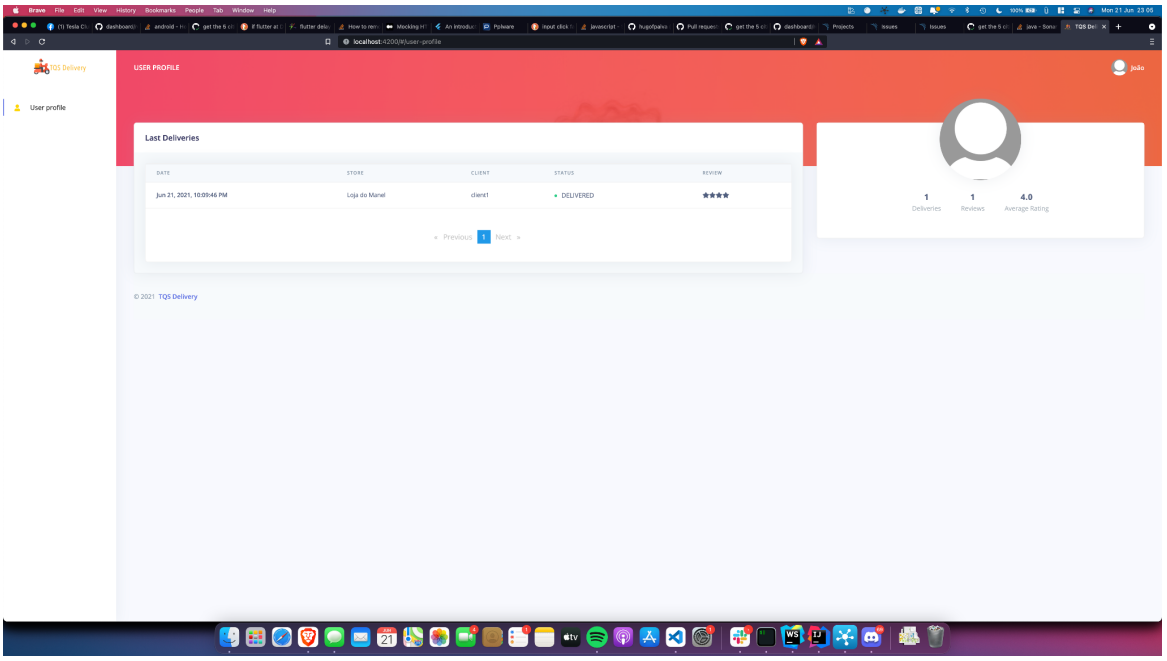

Figura 8: Página web para os riders

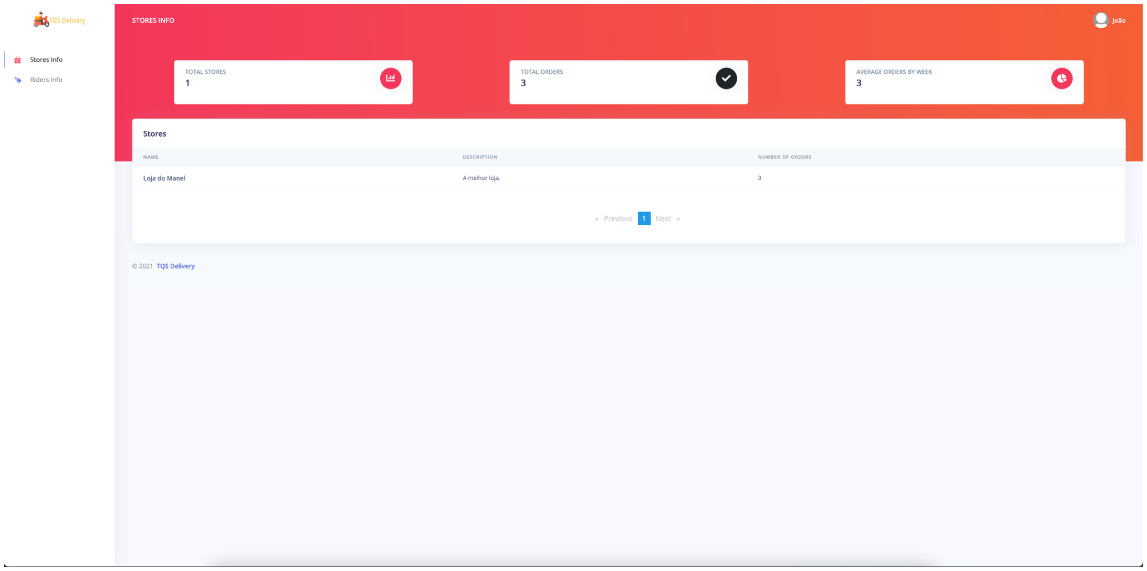

Figura 9: Página web para o manager: informação relativa às várias lojas

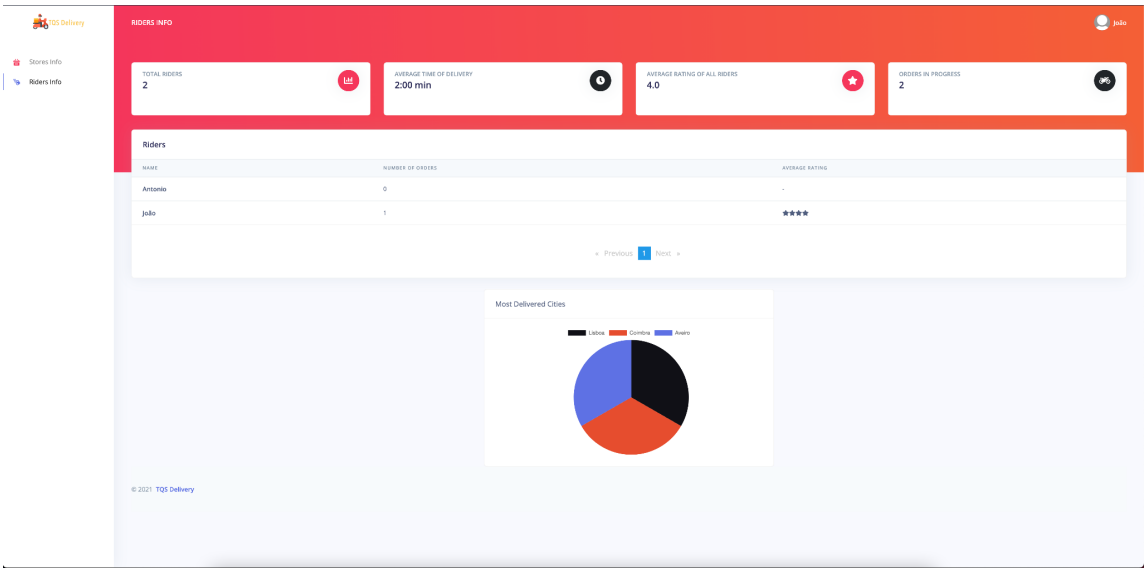

Figura 10: Página web para o manager: informação relativa aos vários riders

No que toca à ligação da criação destes serviços com a disciplina de Testes e Qualidade de Software, o objetivo traçado desde início foi a implementação do sistema, garantindo a qualidade do sistema geral com a criação de testes para os serviços disponibilizados:

- ***Template Integration Tests*** para todos os controllers e os respetivos endpoints mapeados pelos mesmos (testa todas a interligação de todos os componentes do sistema)
- **Testes unitários de todos os controllers e respetivos endpoints**, utilizando **Mockito** (testa o comportamento do controller) , MockMVC e Rest Assured MockMVC.
- **Testes dos serviços criados para suportar os vários controllers**, utilizando **Mockito** (testa o comportamento dos serviços em concreto)
- **Testes das queries criadas e utilizadas para ir buscar os dados à base de dados** (testes de repositório).

Para além disso, foi também considerada a avaliação estática de código (static code analysis) a partir da ferramenta *Sonar Qube* e foram criados testes funcionais da interface de utilizador com *Selenium*. Finalmente, foram criadas pipelines de *CI/CD usando a ferramenta de actions do github* que garantem que a solução passa por todos os testes com sucesso, com o grau de satisfação definido pelo grupo - quality gates.

1.2 Limitações

Quanto às limitações da solução, em termos de *features* que foram consideradas mas que acabaram por não ser realizadas, não houve nenhuma. Tudo aquilo a que o grupo se propôs fazer inicialmente, em termos de requisitos funcionais, foi alcançado com sucesso.
No entanto, considerando a implementação de todas as features, bem como a testagem de cada uma das mesmas, acabou por não haver tempo para criar testes de Flutter, o que seria interessante por motivos de aprendizagem e garantir ainda uma melhor qualidade.
Por outro lado, tendo em conta as dificuldades sentidas, o grupo encontrou impedimentos aquando a criação de testes da interface de utilizador: devido à utilização de **ng-bootstrap modals** nalgumas

páginas, os testes falham devido a pequenas incongruências cuja origem não foi possível apontar. Por vezes os testes passam, por vezes falham, embora não tenham havido quaisquer tipo de alterações. Várias soluções foram tentadas, como obrigar o código a esperar numa certa parte para garantir que o modal está já aberto, criar script que dê scroll para a posição dos modals para garantir que se carrega no sítio certo, colocar uma espera para verificar se o elemento já tinha surgido, *etc*. Como nada funcionou, os testes foram desativados, embora permaneçam no código entregue, de modo a que seja ainda possível analisar a lógica por detrás dos mesmos e o objetivo pretendido com os testes.

2 Conceito do Produto

2.1 Visão

O sistema será utilizado para permitir atribuição dinâmica de pedidos feitos pelos clientes em diferentes aplicações parceiras de TQS Delivery, a um condutor que irá responsabilizar-se pela recolha do produto escolhido e consequente entrega. Os produtos disponibilizados no serviço serão oferecidos por lojas que se registem, representando as aplicações parceiras, como é o exemplo de HumberPeças, ambos já mencionados previamente.

Foi desenvolvida uma plataforma que facilita a criação de um serviço de entregas próprio e personalizado a pequenas e grandes empresas que não tenham a capacidade técnica de o fazer.

Enquanto serviços como Uber Eats e Glovo agregam todas as lojas na sua plataforma, a proposta apresentada pretende providenciar uma plataforma própria para cada loja, permitindo que os proprietários do negócio, tenham uma maior liberdade na venda (marketing) e gestão dos seus produto. No entanto, do mesmo modo que estas duas plataformas mencionadas, oferecemos a possibilidade de beneficiar do sistema de entregas ao domicílio comum.

Para efeitos de demonstração, irá-se usar como caso de uso uma loja de ferramentas que se irá associar ao nosso serviço.

Em termos daquilo que foi alterado, relativamente à ideia inicial, apenas se pode apontar o facto de que foram adicionadas features adicionais para complementar o sistema. Exemplos destas features complementares, algumas das quais foram sugeridas pelo professor, são: adicionar a possibilidade de o manager poder verificar a informação de forma concisa, *i.e* gráfico, bem como a implementação da geração de coordenadas de entrega e a sugestão aos riders apenas das compras realizadas numa proximidade geográfica.

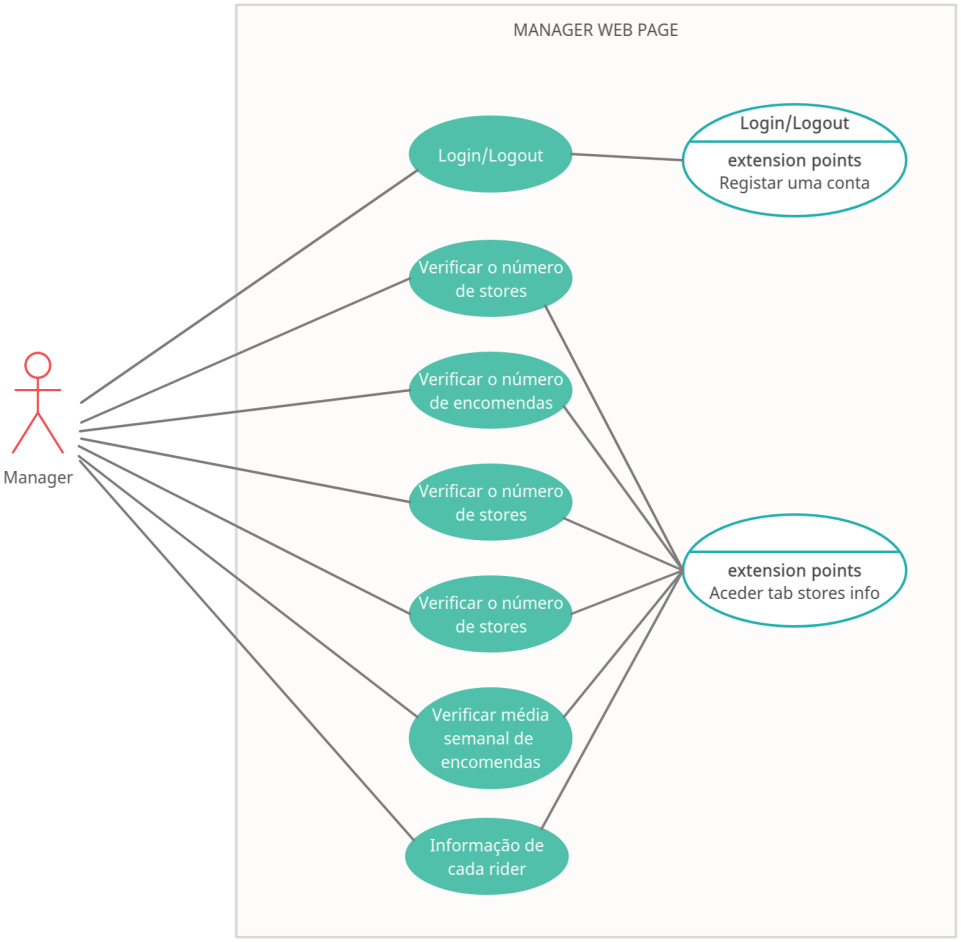

Figura 11: UML Case Diagram de manager relativamente à informação das lojas

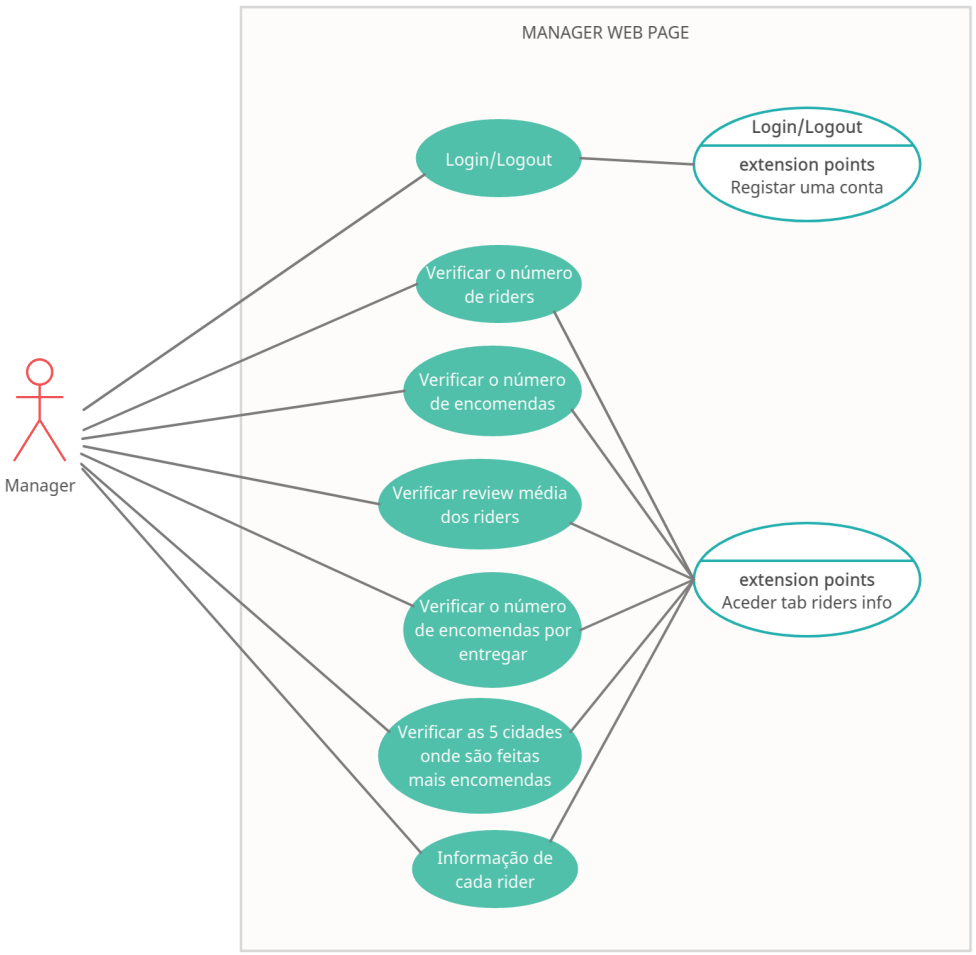

Figura 12: UML Case Diagram de manager relativamente à informação dos riders

O grupo reuniu-se em fases iniciais e, através de brainstorming, tentou entender e conceber a solução que deveria ser criada. Com o apoio do professor foi possível retificar ideias, algumas das quais teriam ficado mal cimentadas, sendo também proposto pelo docente algumas ideias que seriam interessantes implementar, quais poderiam ser descartadas (quer fosse por falta de tempo ou por não serem obrigatoriamente necessárias) e quais os pontos fulcrais que teriam de ser implementados para garantir uma boa solução.

2.2 Personas

O António tem 30 anos, é proprietário de uma mota e quer rentabilizar o seu tempo livre. Trabalhou num supermercado durante 15 anos e está cansado de ter sempre a mesma rotina. Como não continuou a estudar depois do que era obrigatório, tem dificuldade em encontrar emprego. Tem facilidade com novas tecnologias e gosta de experimentar coisas diferentes.

Adalberto, com apenas 23 anos, é um faz-tudo na sua aldeia. É conhecido por muita gente pela sua habilidade em reparar coisas e está sempre pronto a ajudar quem lhe pede. Normalmente costuma ser o próprio Adalberto a deslocar-se à cidade para comprar as peças. No entanto, percebe que esta solução não é a mais rentável, fazendo-o perder tempo com as deslocações e também ter gastos adicionais.

O Juvenal tem 56 anos e quase toda a sua vida foi dono de uma pequena loja de ferramentas. Apesar de gostar do seu trabalho, às vezes sente-se cansado e gostaria de conseguir expandir o seu negócio de uma forma simples. Para além disso, tem visto o lucro da sua loja a decrescer por causa da pandemia. Não está acostumado a usar a internet a não ser para ler as notícias.

A Regina tem 25 anos e acabou há pouco tempo a sua licenciatura. Sente-se agora capaz de poder gerir uma equipa e estar sempre à frente de toda a informação relativa à empresa que ajuda a controlar. Está habituada a usar tecnologias, embora não seja muito boa a inglês. É assertiva, o que faz dela uma manager muito independente e competente.

2.3 Cenários principais

- Cenário 1**: Distribuidor**

  Uma vez que o António está muito à vontade com as novas tecnologias, ele decidiu **realizar entregas de produtos** das diversas lojas associadas ao serviço genérico, através da plataforma disponibilizada.

- Cenário 2: **Cliente**

  Devido ao constante número de pedidos, ter de se deslocar à cidade todas as vezes que fosse necessário algo novo tornar-se-ia demasiado dispendioso. Assim sendo, começou a optar pelo nosso serviço, mais rápido, barato e cômodo!

- Cenário 3: **Empresário** (dono de loja)

  O Juvenal, devido à pandemia, tem tido pouco lucro e, por esse motivo, queria arranjar uma forma de poder potencializar o seu negócio. Ao aderir ao nosso serviço, conseguiu potencializar as suas vendas, permitindo aos clientes receberem os produtos sem saírem de casa.

- Cenário 4: **Manager** (de riders e stores)

  A Regina aproveitou o facto de não ter de se deslocar até à sua empresa para poder gastar mais tempo a investigar o progresso dos empregados de TQS Delivery, os riders, bem como das lojas que estão associadas ao serviço de entregas. Assim, é capaz de garantir que está a par do crescimento da empresa e pode gerir recursos conforme o necessário.

2.4 Project epics e Prioridades

Para implementação da nossa solução iremos usar uma metodologia *agile* com *sprints* semanais (1 semana). A totalidade do projeto foi realizada em 4 sprints, *i.e* semanas.

- **Semana 1 (25/05 a 31/05)**

  1 - Configure Specific Backend App DataModel
  2 - Configure Generic Backend App DataModel
  3 - Create Prototype of Web Apps
  4 - Create Mobile App Prototype
  5 - WebApplication Docker Setup
  6 - Setup HTTP Client in Specific Backend App
      6.1 - Start creating endpoints for main use cases

  * Nota: O grupo apenas ficou esclarecido relativamente à necessidade e propósito de criação de *issues* user story driven no fim desta iteração. Assim sendo, apenas a partir daqui as Epics começaram a ser consideradas como constituídas por *issues* no formado de: **As a <user>, I want to <a goal> so that <benefit>**.

- **Semana 2 (01/06 a 07/06)**

  1 - **(Epic)** The web application client can create an account and log in
  2 - **(Epic)** The rider has a functional mobile application
  3 - **(Epic)** Rider can access the web application and go to the 'user profile' tab so that they can access all of their information and the status+reviews of their deliveries.

- **Semana 3 (08/06 a 14/06)**

  1 - **(Epic)** Rider manager can access the web application and go to the 'riders info' tab so that they can check the statistics of all the deliveries that have already been made.
  2 - **(Epic)** Rider can access the web application and go to the 'user profile' tab so that they can access all of their information and the status+reviews of their deliveries.
  3 - **(Epic)** The web application client can create an account and log in
  4 - **(Epic)** All of the users can trust the application and the dev-ops are assured that everything is OK
  5 - **(Epic)** The web application client has access to a functional shop page

- **Semana 4 (15/06 a 21/06)**

  1 - **(Epic)** Rider manager can access the web application and go to the 'riders info' tab so that they can check the statistics of all the deliveries that have already been made.
  2 - **(Epic)** The rider has a functional mobile application

  3 - **(Epic)** Rider can access the web application and go to the 'user profile' tab so that they can access all of their information and the status+reviews of their deliveries.
  4 - **(Epic)** The web application client has access to a functional shop page
  5 - **(Epic)** The web application client has a functional profile page
  6 - **(Epic)** All of the users can trust the application and the dev-ops are assured that everything is OK
  7 - **(Epic)** Rider manager can access the web application and go to the 'stores info' tab so that they can check the information of all the stores associated with the delivery service.

  * Nota: Adicionalmente, nesta semana, apesar de não corresponderem a nenhum Epic, foram também concluídas as seguintes tarefas:

  1 - (Issue) As a Rider I want to get the order with the Store Nearest To Me
  2 - (Issue) As a user I want to be able to access the application even if there is a big amount of traffic, so that I can use it normally without problems or delays.

3 Domain model

Como é possível observar na **Figura 13**, TQS Delivery Service model domain, o sistema do serviço genérico de entregas que foi criado, *i.e* TQS Delivery, é suportado por uma base de dados com ***postgresql***. O modelo da base de dados utilizado pode ser repartido por 4 entidades diferentes: **address, store, person** e **purchase**. É importante relembrar o objetivo deste sistema, integrar outros serviços (lojas específicas); como consequência, o mais lógico foi criar estas entidades com colunas bastantes genéricas, sendo que não estarão relacionadas com nenhuma loja apenas, mas sim várias.

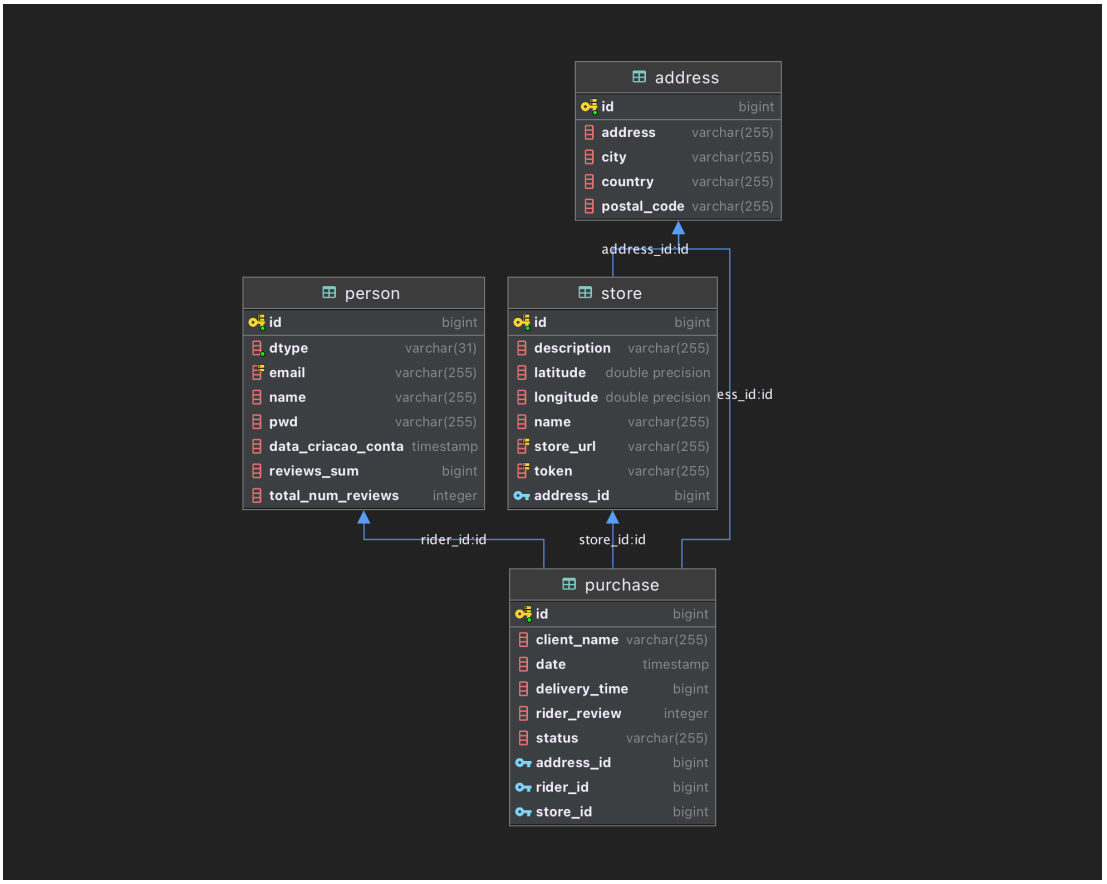

Figura 13: TQS Delivery Service model domain

Para além deste, para o sistema que suporta a HumberPeças, uma loja (serviço específico) de venda de ferramentas e outros bens necessários para arranjos e construção, foi criado também um modelo. Este é constituído por 5 entidades diferentes: **person, address, purchase, product** e **purchase_products**. O modo como estas entidades se relacionam entre si pode ser observada na **Figura 14**, HumberPeças model domain.

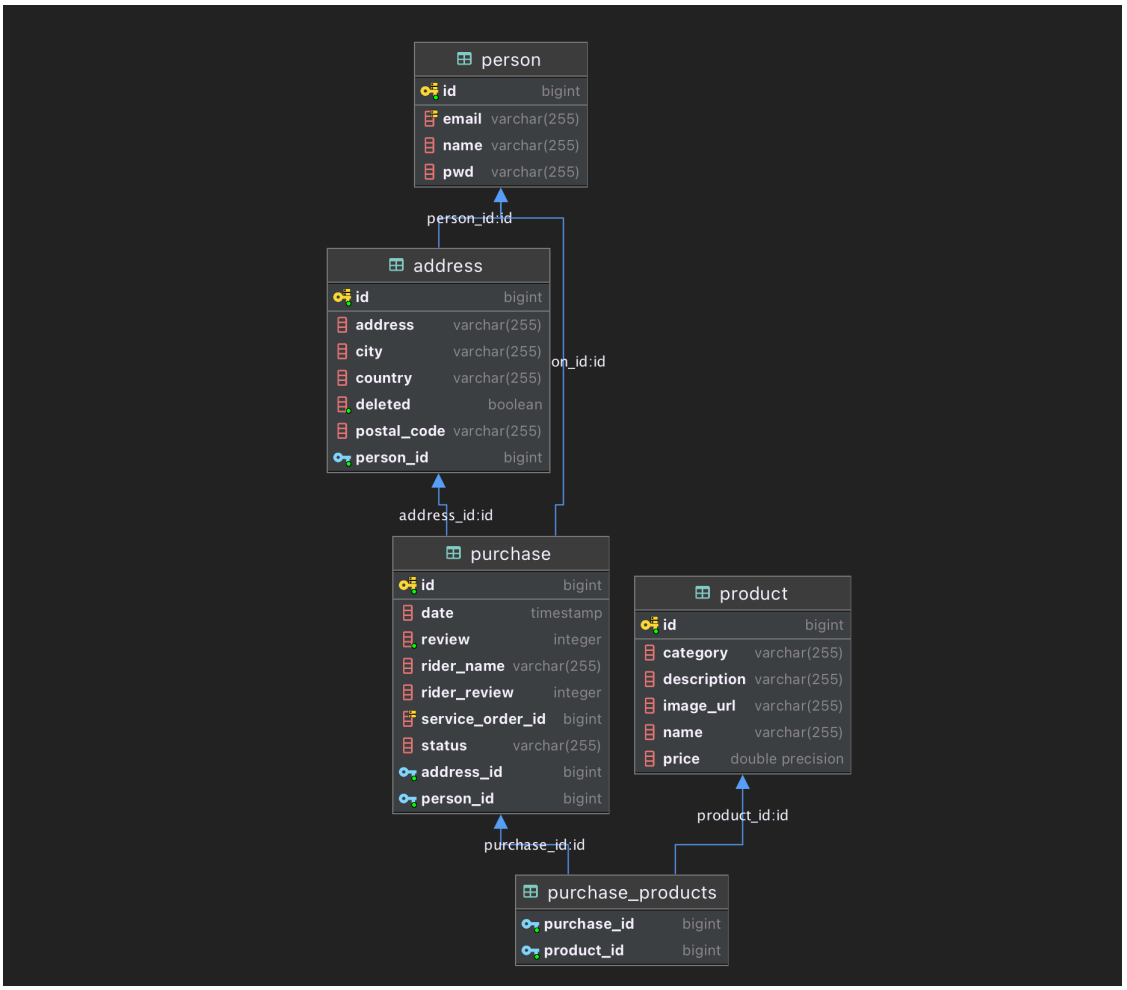

Figura 14: HumberPeças model domain

4 Architecture notebook

4.1 Key requirements e Restrições

A escolha da arquitetura foi feita tendo em conta as seguintes questões:

- O sistema de entregas deve ser capaz de hospedar várias lojas, tendo a capacidade de receber vários pedidos em simultâneo.
- O sistema não pode permitir acesso por terceiros a dados confidenciais, necessitando de um sistema de autenticação.
- O sistema deve ser alojado numa máquina virtual a correr num servidor externo.
- O sistema deve ter uma plataforma específica para que os entregadores possam ser notificados e aceder de forma fácil às suas encomendas.
- O sistema deve ter uma plataforma para o administrador, distinta das do entregadores, para que estes possam ver estatísticas sobre o uso do sistema.

4.2 Visão arquitetural

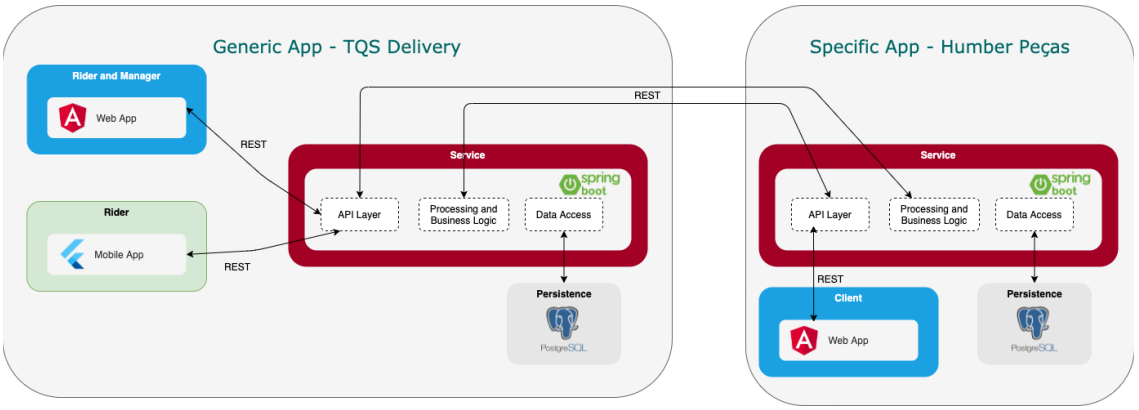

Figura 15: Arquitetura do Sistema

As bases de dados que garantem a persistência de dados, tanto do serviço genérico de entregas como para as lojas específicas, neste caso Humber Peças, serão suportadas por PostgreSQL. A partir do Spring Boot é possível aceder às bases de dados, processar os dados e por sua vez servir os dados através de uma Rest API para o frontend.
Já em relação à aplicação móvel, foi utilizado Flutter, enquanto que para as aplicações web foi utilizado Angular.

Utilizou-se Spring Boot por ser um requisito do trabalho, enquanto que o Angular foi por ser uma tecnologia à qual todos os membros do grupo já se encontravam acostumados.

O grupo decidiu usar uma base de dados SQL, pelo que se pensou inicialmente utilizar MySQL, mas como PostgreSQL é open source, pareceu mais indicado utilizar este sistema de gestão de gestão de bases de dados relacional. Entre bases de dados relacionais ou não relacionais, compreendendo quais as ligações que os dados iriam necessitar de ter e, tendo em conta todos os benefícios de usar SQL, e.g

maior velocidade, fácil portabilidade, etc., fez mais sentido suportar o sistema com bases de dados relacionais.

O Flutter foi, neste caso, utilizado porque permite desenvolvimento de mobile apps de forma simples.

O modo como todos estes componentes irão interagir encontra-se detalhado na Figura 15, acima, sendo que o maior destaque vai para a comunicação entre ambos os serviços que acontece quando, por exemplo, um cliente faz uma nova comprar, um cliente atribui uma review a uma entrega, um estafeta altera o estado da encomenda, entre outros, permitindo que tanto as entidades de um serviço como do outro tenham as informações atualizadas.

4.3 Arquitetura de *deployment*

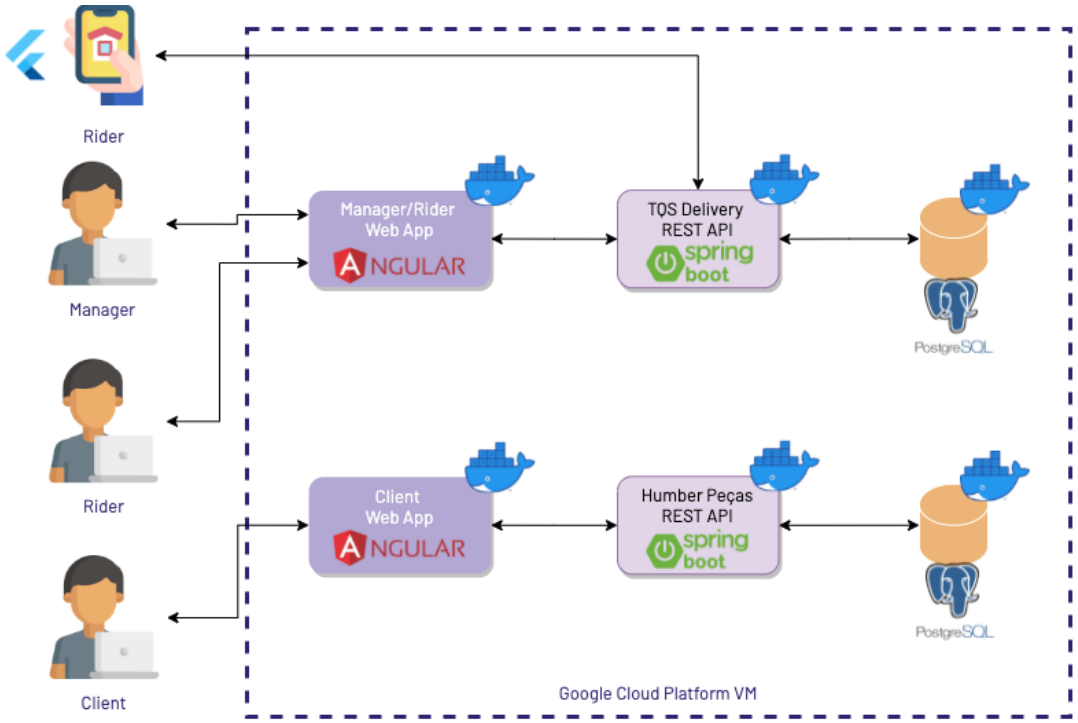

Figura 16: Arquitetura de deployment

Como é possível visualizar na Figura 16, o deploy foi feito com recurso a uma máquina virtual da Google Cloud Platform sendo que foram executados os diversos serviços, bases de dados e aplicações web com recurso a containers docker. Desta forma, tanto o deploy inicial, como o continuous deployment foi facilitado uma vez basta reconstruir os serviços alterados e iniciá-los novamente num container aquando novas alterações.

5 API para os Developers

Informação relativa à organização das duas REST APIs criadas: **HumberPeças** para o serviço específico (loja) e **TQS Delivery** para o serviço genérico de disponibilização da possibilidade de entrega ao domicílio, com diferentes riders espalhados pelo país.

Encontra-se disponível no ***swagger***, uma Interface Description Language usada para descrever RESTful APIs, expressando-as utilizando JSON. [1]

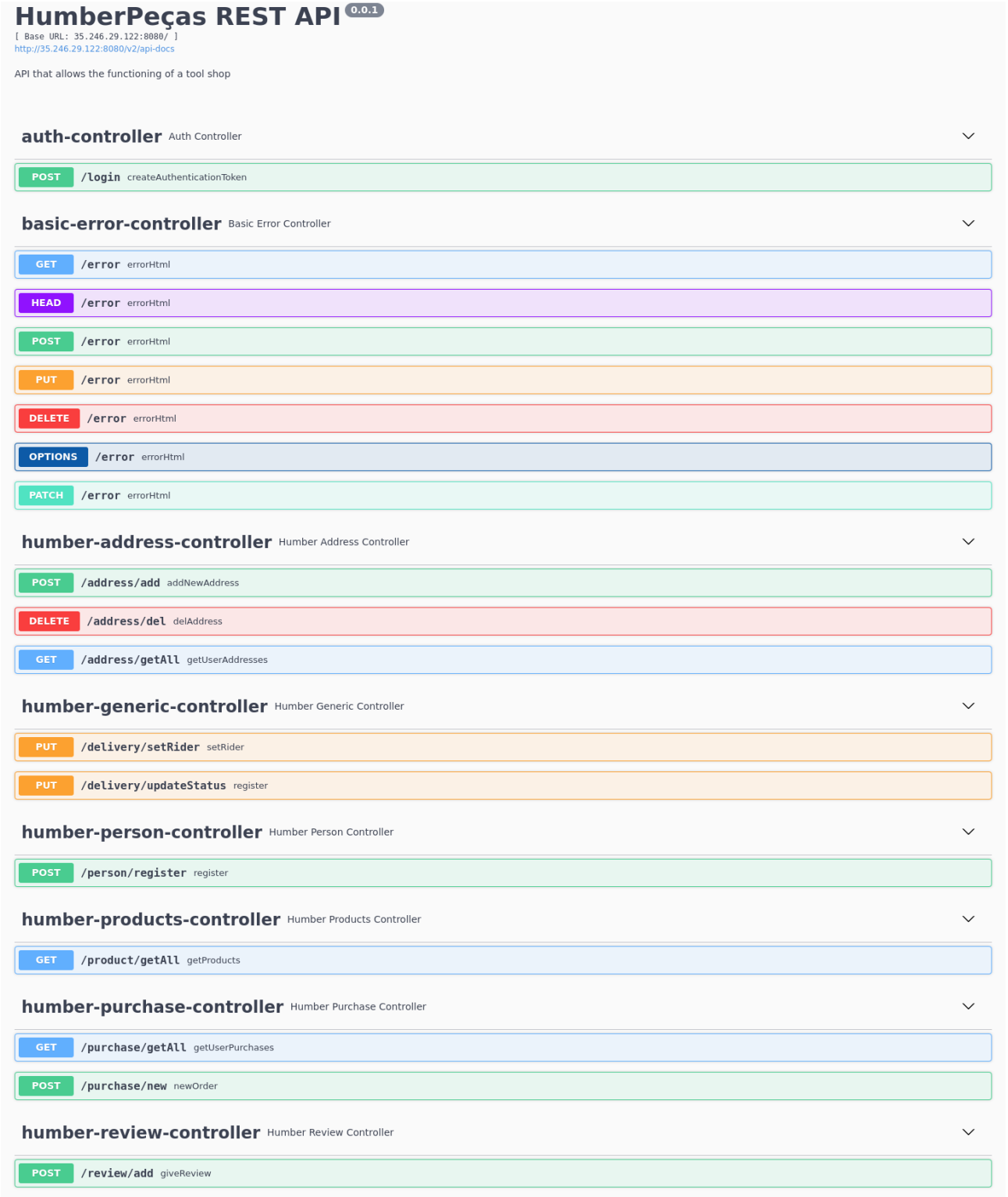

Figura 17: HumberPeças REST API in detail

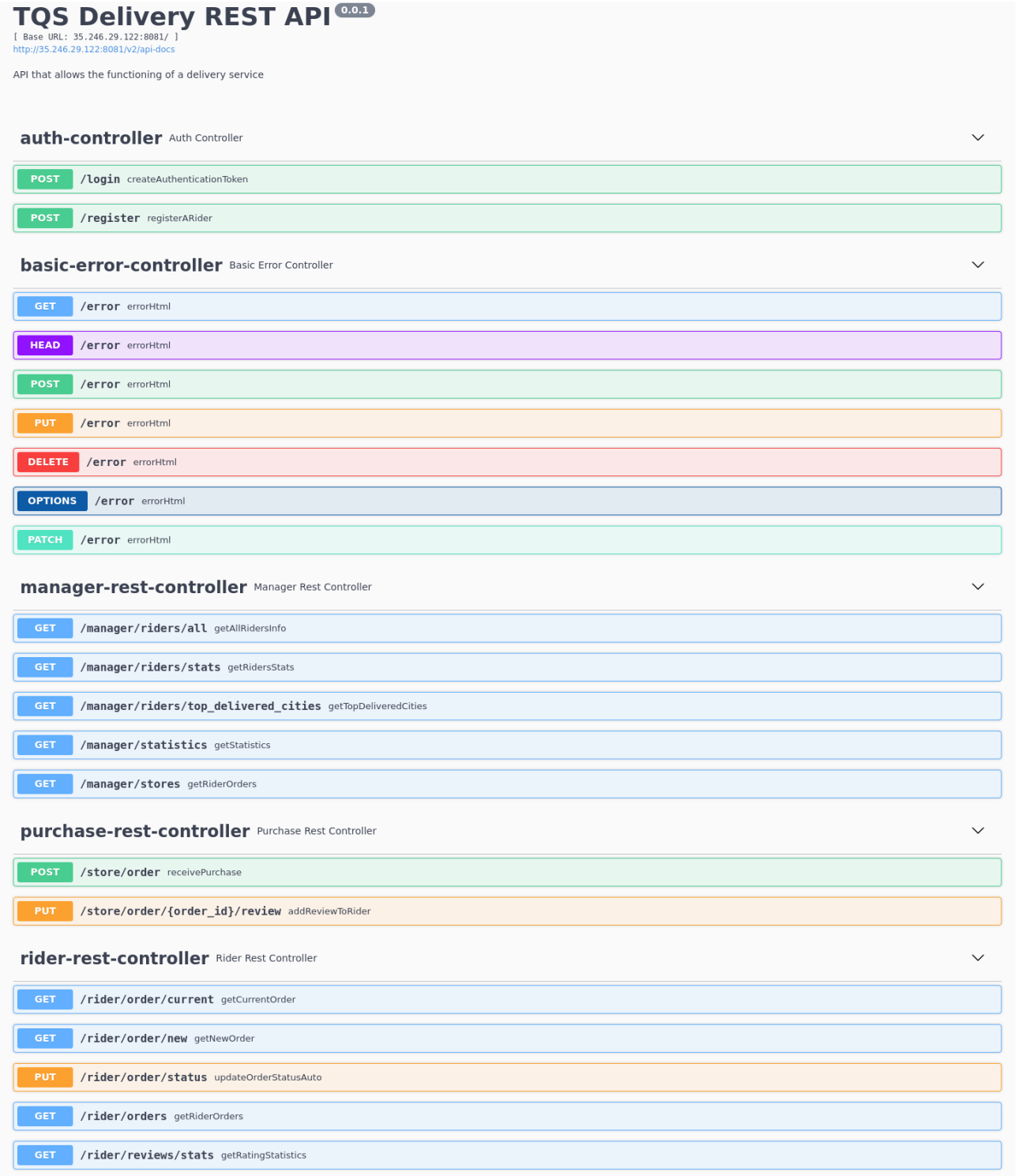

Figura 18: TQS Delivery REST API in detail

Qualquer desenvolvedor que pretenda criar uma solução, seja ela qual for, tirando partido das *REST APIs* disponibilizadas, como visto nas **Figuras 17** e **18**, poderá fazê-lo e ter acesso a toda a informação e recursos especificados nessas mesmas figuras. A título de exemplo: utilizando a *REST API* de **TQS Delivery**, para, *e.g* ligar uma plataforma web de loja ao serviço de entregas, um developer poderá consultar a documentação disponibilizada pelo Swagger, estudar como os *endpoints* se comportam com o diferente tipo de pedidos realizados, as variáveis que podem ser necessárias para chamar certas funcionalidades ou verificar se e é necessário um request body, se irão receber um response body, *etc*.

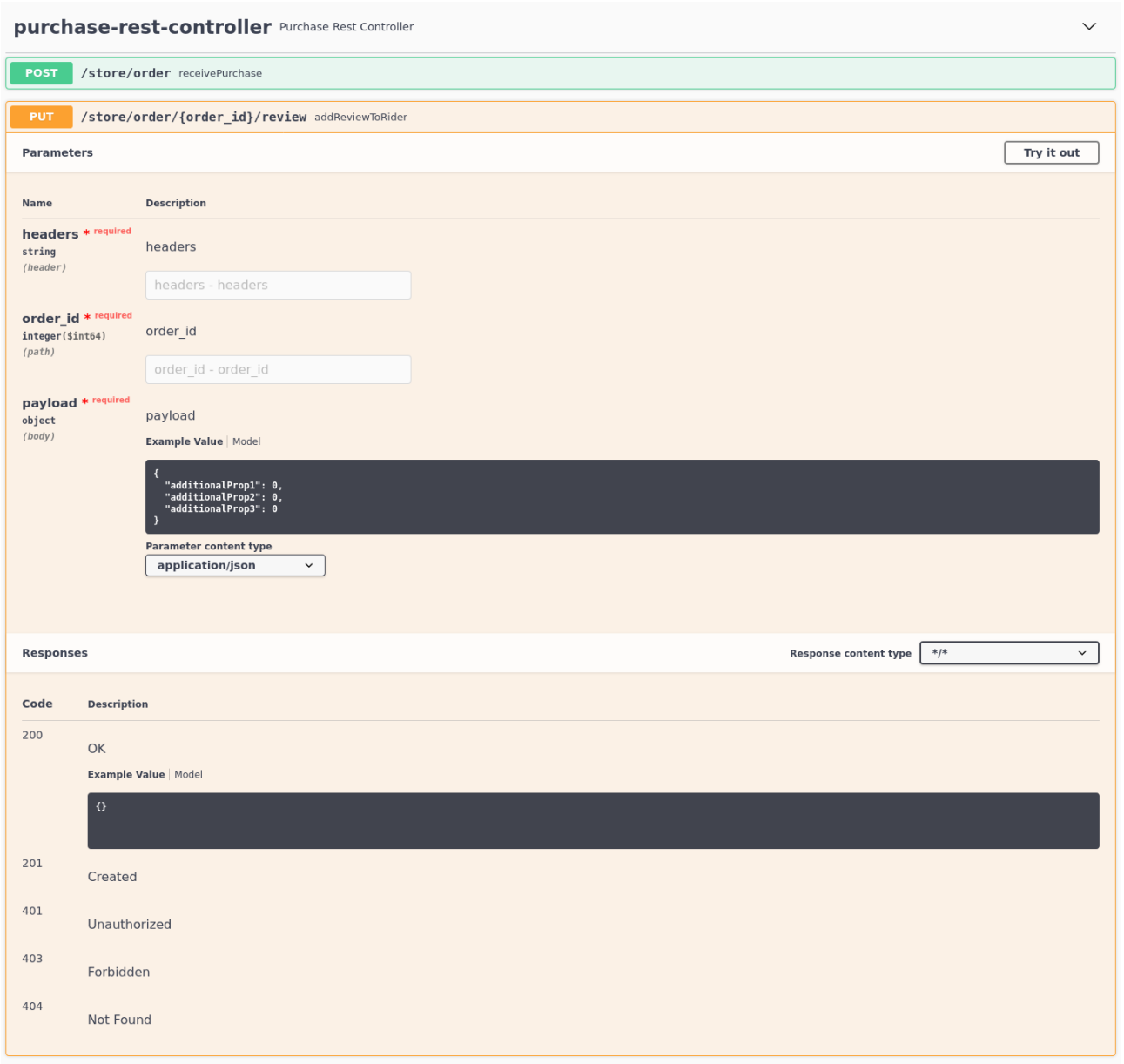

Figura 19: Exemplo da informação que é possível retirar sobre cada endpoint a partir do Swagger

6 Conclusão

Em tom de remate, é de notar a motivação e o empenho que a equipa colocou neste projeto. Não tendo perfeitamente percebido os objetivos do trabalho ao início, o tempo de ajuste à ideia foi ainda um pouco atribulado. Tendo de se alterar a arquitetura da solução, depois da primeira semana e meia, isto implicou ter de dar refactor completo do modelo de dados, adicionar serviços e todos os subsequentes testes, visto que estes também necessitaram de sofrer alterações. Conclui-se desta forma que a definição do conceito do produto, deve ser bem definida numa inicial do desenvolvimento do projeto, de maneira a não ter que fazer alterações drásticas a meio do desenvolvimento, uma vez que isto tem um impacto brutal nos custos.

Embora na pipeline de *CI*, se tenha testes unitários, funcionais e de integração, tanto para a *branch* principal como para a de desenvolvimento, foi possível concluir que esta abordagem para projetos de maior dimensão não faria muito sentido, uma vez que seria um processo que gastaria bastante tempo, toda a vez que se adicionava uma nova funcionalidade. Para esses casos talvez fará mais sentido, apenas correr os testes funcionais e de integração sempre que se faz, por exemplo, uma nova *release*.

Apesar de todas as dificuldades sentidas no ínicio, por exemplo até apenas a tentar integrar uma metodologia *TDD*, o grupo trabalhou para ter pronto o serviço, deployed, com uma grande quantidade de testes a comprovar a sua qualidade, métricas excelentes no SonarQube, testes de interface, bem como apresentar ainda os resultados da utilização de JMeter para testar functional behavior.

Assim sendo, considera-se que todas as metas foram cumpridas, o grupo sente que aprendeu bastante com este trabalho e que foi uma mais valia.

7 Referências e Recursos

[1] Swagger (software) https://en.wikipedia.org/wiki/Swagger_(software), disponível Online, acedido a 22 de junho, 2021.