deti universidade de aveiro departamento de eletrónica, telecomunicações e informática

TQS: Relatório de Especificação do Produto

Francisco Fontinha [76490], João Vasconcelos [88808], Tiago Mendes [88886], Vasco Ramos [88931]

v2020-06-02

1. Introdução

1.1. Apresentação do Projeto

Este projeto tem como principal objetivo aplicar metodologias de integração e testes automáticos com a implementação de uma bateria completa de testes. Para além disso, visa também aplicar metodologias ágeis para facilitar o desenvolvimento do projeto em equipa.

No geral, o principal foco é implementar um produto de *e-commerce* (*marketplace* online), seguindo uma estratégia de *Software Quality Assurance* (SQA).

1.1.1. Oportunidade de Negócio

Uma empresa com atividade no segmento imobiliário pretende alargar a sua área de atuação e competir contra empresas com forte presença no mercado nacional de arrendamento. Assim, esta empresa encontrou uma área pouco explorada no mercado de arrendamento imobiliário e, seguindo a tendência atual de *e-commerce*, pretende investir numa plataforma online de arrendamento, que tem como público alvo alunos universitários que procuram um espaço para alugar a curto/médio prazo.

Deste modo, a nossa equipa, em conjunto com a empresa interessada em investir, prossegue com a criação de uma plataforma que agiliza e simplifica o processo de arrendamento a universitários.

O público-alvo desta plataforma caracteriza-se por ser bastante heterogéneo e diversificado, pelo que é emergente a necessidade de criar um produto inovador e que traga valor para os utilizadores. Nesta perspetiva, a equipa considera que um aspeto fulcral do produto relaciona-se com reduzir ao máximo a burocracia envolvida no aluguer de imóveis, ou seja, expor os utilizadores o menos possível a toda a burocracia necessária, tentando automatizar o processo. É aqui que surgem duas características diferenciadoras: comparador de imóveis e gerar automaticamente um contrato de aluguer entre um locador e um locatário, diminuindo o tempo dispendido no processo, bem como eventuais despesas associadas.

Assim, aliando a estas funcionalidades a possibilidade de avaliar os imóveis e pesquisa inteligente, acreditamos ter um produto capaz de se destacar no mercado e preencher lacunas atualmente existentes no que diz respeito ao aluguer de imóveis a estudantes universitários.

1.1.2. Ideia do Produto

Para o/a:	Estudante ou Locador.
Que apresenta:	Dificuldade em encontrar imóveis para arrendamento (estudante universitário/ locatário) e, ainda, dificuldade em publicitar os seus imóveis (locador, que acaba por utilizar redes sociais e marketplaces generalistas).
O produto:	DOMUS, que é uma plataforma de arrendamento de imóveis.
Que:	Agiliza e simplifica o aluguer de espaços de habitação, especificamente para estudantes.
Ao contrário de:	Imovirtual e Airbnb.
O nosso produto:	Facilita a procura e escolha de um espaço para habitar que vai de encontro às necessidades do público-alvo (estudantes universitários). Otimiza o processo de decisão, pois, inclui avaliações do imóvel submetidas através do *feedback* de arrendatários anteriores. Permite a comparação de até três imóveis. Permite gerar automaticamente um contrato de arrendamento.
Competição:	Uniplaces

Tabela 1: Breve descrição do produto

1.2. Limitações

De um modo geral, o nosso sistema satisfaz quase a totalidade dos requisitos propostos, pelo que obtivemos um produto completo, eficiente e robusto. No entanto, existem duas funcionalidades que não foram implementadas, nomeadamente:

- A geração de um documento, em formato PDF, referente a cada contrato celebrado entre um locador e um locatário, para um determinado imóvel;
- A possibilidade de um locador efetuar uma avaliação de um determinado locatário, que teria bastante utilidade para que um outro locador obtivesse feedback sobre o bom comportamento de um determinado locatário que pretenda alugar o seu imóvel;

Neste sentido, e tendo em conta que estas duas funcionalidades estejam em falta no nosso sistema, gostaríamos de as desenvolver no futuro.

2. Conceito do Produto

2.1. Visão Geral dos CaU

A nossa aplicação, Domus , apresenta uma solução para o problema de procura de quartos por parte de estudantes universitários para arrendar. Esta procura normalmente é demorada e muitas vezes os estudantes têm que se deslocar várias vezes à cidade universitária até conseguirem arrendar um quarto. É este o problema que o nosso produto pretende resolver oferecendo uma solução que à distância de um clique permite ao utilizador verificar as características do quarto e se tiver interesse arrendar o mesmo.

Com este objetivo em mente, existem vários casos de uso essenciais no nosso produto como a pesquisa de quartos e avaliação dos mesmos por parte do locatário e a adição de publicações de quartos por parte do locador. Estes casos de uso e os restantes presentes na nossa solução estão representados através de um diagrama de casos de uso na figura 1.

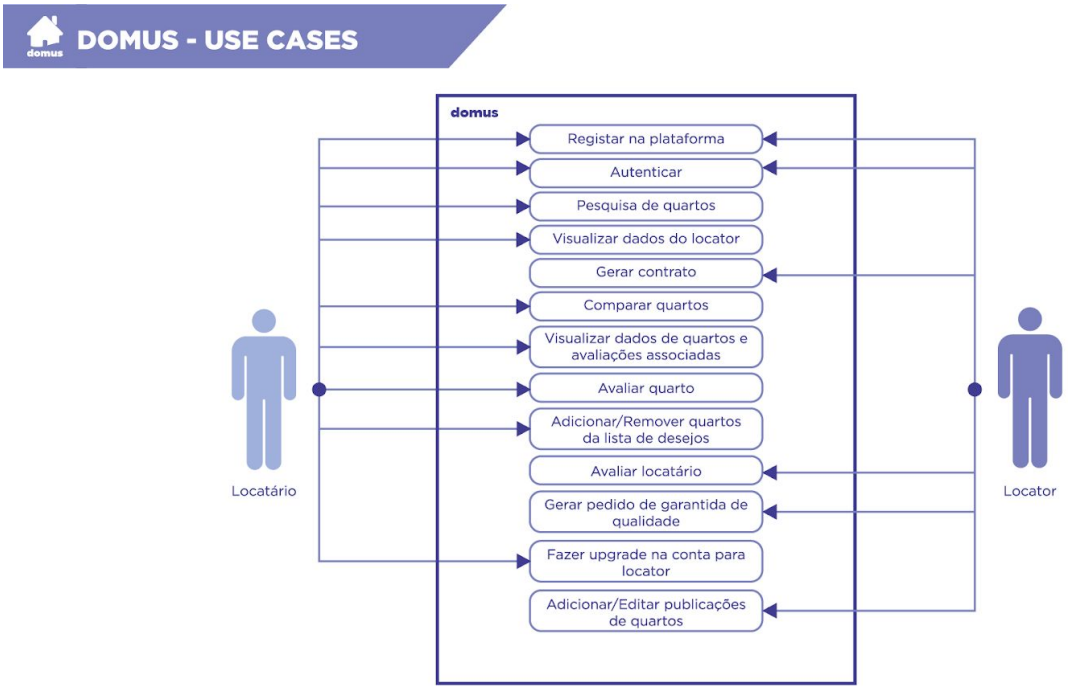

Figura 1: Diagrama de Casos de Uso

2.2. Personas

O desenvolvimento da aplicação domus foi realizado tendo em conta 2 Personas principais: o Carlos Santos, um locador e o Rafael Gomes, um locatário.

- **Carlos Santos**

O Carlos Santos, tem 40 anos e é médico no centro de saúde de Aveiro há 10 anos. É uma pessoa energética que gosta de jogar tênis, praticar atletismo e ouvir música nos seus tempos livres. Há cerca de 5 anos, o Carlos decidiu investir no mercado imobiliário em Aveiro e comprou 2 casas na vila jovem com o objetivo de arrendar a estudantes universitários.

Durante o verão, antes do início do novo ano letivo, muitos dos locatários do Carlos saem das suas casas e este fica com a tarefa de encontrar novos inquilinos que queiram arrendar um quarto. O Carlos acaba por despender muito tempo a mostrar casas a clientes interessados e perde muitos clientes porque não se pode ausentar do trabalho para visitar as moradias com eles.

Motivação: O Carlos gostava de possuir uma forma mais fácil de apresentar os seus imóveis a clientes interessados.

Figura 2: Carlos Santos

- **Rafael Gomes**

O Rafael tem 18 anos e vive em Coimbra, onde concluiu o ensino secundário. É uma pessoa motivada, praticando basquetebol e natação nos seus tempos livres. Além disso o Rafael gosta de jogar jogos de computador com os seus amigos e pretende estudar Engenharia Informática em Aveiro.

Com o seu primeiro ano de faculdade a chegar, os pais do Rafael pretendem alugar um quarto para o filho em Aveiro, onde ele possa viver durante o período escolar. Como os seus pais trabalham durante a semana, apenas podem ajudá-lo na procura de quartos ao fim de semana, por isso durante a semana o Rafael desloca-se várias vezes a Aveiro sozinho para visitar novos quartos. Ele leva sempre um bloco de notas onde aponta as características dos quartos e das casas para poder mais tarde comparar os vários quartos que visitou.

Este processo acaba por ser muito dispendioso e cansativo, ocupando um grande intervalo de tempo dos dias do Rafael e dos seus pais.

Motivação: O Rafael gostava de possuir uma forma rápida e acessível para procurar quartos e poder compará-los.

Figura 3: Rafael

2.3. Principais Cenários de Utilização

Estes cenários de utilização principais descrevem como o nosso produto se integra com as atividades dos utilizadores finais e como eles as executam de forma a alcançar os seus objetivos.

- **O Carlos pública um novo quarto:**

Um dos locatários do Carlos acabou de sair de uma das suas casas por isso o locador tem um quarto livre para arrendar. Por esse motivo, o Carlos abre a web app e depara-se com uma página onde são apresentadas as funcionalidades principais do **domus**. Após efetuar login, ele seleciona a opção **new room** e depara-se com um formulário onde são pedidas diversas informações sobre o quarto que o locador quer publicar.

Após preencher todas as informações pedidas, o Carlos submete o formulário e recebe uma mensagem de confirmação, informando-o que o quarto foi publicado com sucesso

- **O Rafael procura por um quarto na cidade de Aveiro:**

O Rafael precisa de encontrar um quarto na cidade de Aveiro. Com esse objetivo em mente, após efetuar login no **domus**, o Rafael observa a listagem de quartos disponíveis na aplicação e insere na barra de pesquisa a query “Aveiro” para filtrar os resultados pela cidade de Aveiro. Após clicar no botão de pesquisa, a página é atualizada e são apresentados os resultados de quartos disponíveis na cidade de Aveiro.

- **O Rafael adiciona um quarto à sua wishlist**

O Rafael pretende adicionar todos os quartos que lhe interessam à sua wishlist para poder mostrar aos seus pais e decidir qual seria o melhor para ele. Por isso, após efetuar o login, o Rafael observa a listagem de quartos disponíveis e clica no ícone com o formato de um coração para adicionar os quartos à sua wishlist.

2.4. Epics e Prioridades

Tendo em conta a especificação do produto e as funcionalidades pensadas, iremos focar-nos inicialmente em conseguir disponibilizar os imóveis, pelo que temos primeiro de concluir as funcionalidades de procura e publicação de casas. Após concluir estes dois epics, a equipa focar-se-á no desenvolvimento da ferramenta de comparação e na possibilidade de feedback. Deixando para o fim o desenvolvimento da wishlist e a geração de contratos automática.

Sprint	Epics
1	Procura de uma casa, Publicação de uma casa
2	Comparação de Imóveis, Colocação de uma review, Adicionar casa à wishlist
3	Gerar pedido de garantia de qualidade, Formalização de um contrato

Tabela 2: Implementação incremental por epics

2.5. Metodologia de Levantamento de Requisitos

Com o objetivo de tornar o Domus uma plataforma mais credível, funcional e segura, uma das etapas de maior importância foi o levantamento de requisitos. No decorrer desta etapa, recorremos a questionários, à documentação fornecida pelo professor e a diversas fontes online.

Com este projeto, pretendemos criar uma plataforma online atrativa e disponível a todos os sistemas operativos, para assim podermos atrair cada vez mais clientes. Posteriormente, pretendemos de forma algo ambiciosa, criar uma app mobile para tornar o Domus mais acessível e prático, potenciando ainda mais a proposta que nos foi apresentada.

Nome	Responsabilidades	Valor para a Plataforma
Estudantes Universitários	Utilizadores da plataforma que observam as ofertas presentes nesta.	São os estudantes os principais responsáveis para que a plataforma se mantenha em bom funcionamento. Fazem parte do core do negócio desenvolvendo o papel de principais clientes no serviço.
Locador	Utilizadores da plataforma que publicam ofertas de imóveis direcionadas aos clientes no serviço, mais especificamente, os estudantes universitários.	Tal como os estudantes, os locadores fazem parte do core do negócio desenvolvendo o papel de principais vendedores no serviço.
Real Estate Holdings	Empresa que inicialmente encomendou a plataforma que se está a desenvolver.	Principal investidor.

Tabela 3: Perfis dos Stakeholders

3. Modelo de Domínio

No diagrama do modelo de domínio, descrevemos as várias relações entre as entidades existentes no sistema. Esta modelação foi essencial para conseguirmos programar mais facilmente as relações entre as diferentes entidades do sistema. Como podemos inferir através da figura 4, existem 2 *users* principais no nosso sistema: o **Locador** e o **Locatário**. O **Locador** publica imóveis no sistema, representados pela entidade **Imóvel** e poderá estar associado a um locatário através de um contrato de arrendamento, representado pela entidade **Contrato.** O **Locatário** pode pesquisar imóveis, adicionar os seus favoritos à wishlist, representada pela entidade **ListaDesejos** e pode submeter avaliações a imóveis através da entidade **Avaliação.**

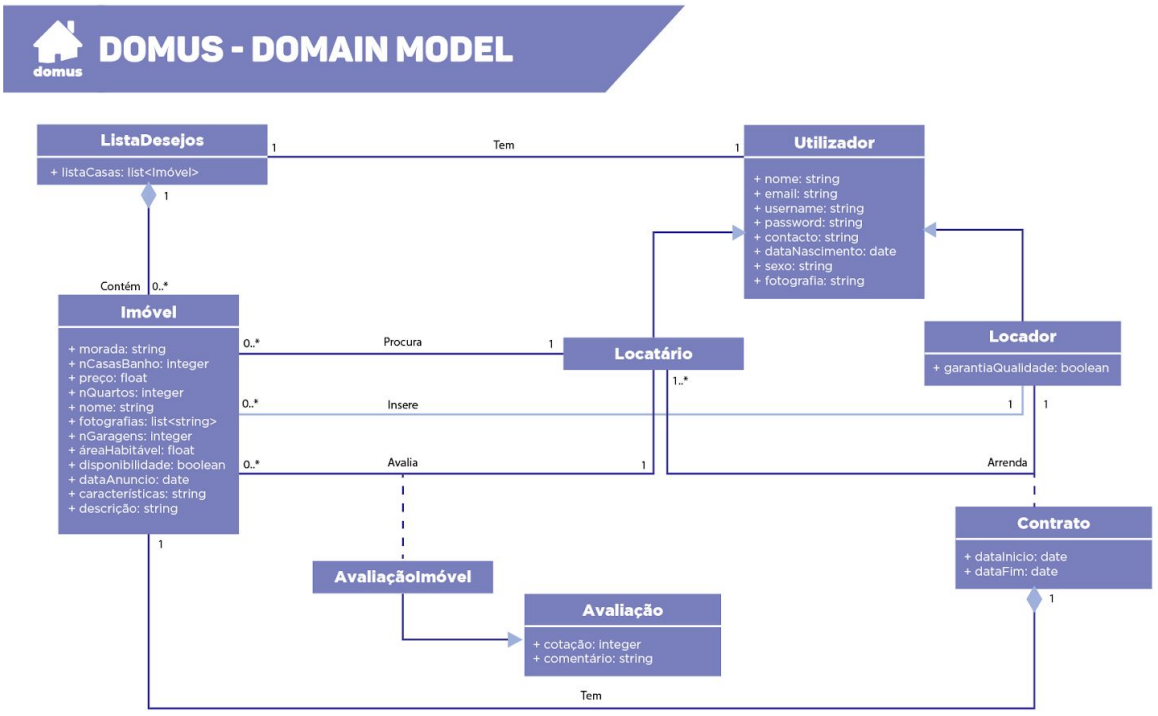

Figura 4: Modelo de Domínio

4. Arquitetura do Sistema

4.1. Principais Requisitos e Limitações

Relativamente à arquitetura do nosso sistema, existem diversos constrangimentos que devem ser tidos em conta.

Primeiramente, devemos referir que o nosso sistema irá ser desenvolvido de raiz, sem qualquer adaptação a um sistema *legacy.* Neste sentido, é estritamente necessário analisar, nesta fase inicial, não só as implicações associadas a cada componente da arquitetura, mas também a forma como todos os componentes irão comunicar e interagir entre si. Assim, esperamos desenvolver um sistema minimamente complexo mas que seja suficientemente robusto para futuras manutenções e/ou integrações com entidades externas.

No geral, o nosso sistema irá ter 4 grandes componentes: uma aplicação web, uma aplicação móvel, um módulo de processamento e business logic e, por fim, uma base de dados. Estes componentes irão interagir entre si, essencialmente, através de pedidos e respostas HTTP. Cada componente deverá ser executado num ambiente próprio, como uma máquina virtual ou num ambiente de conteinerização, como o Docker.

Como referido anteriormente, queremos que o nosso sistema seja suficientemente robusto para que, no futuro, possa ser integrado com um qualquer serviço externo que lhe acrescente mais funcionalidades e, consequentemente, mais valor.

Relativamente à interação do nosso sistema com os utilizadores, iremos ter tanto uma aplicação web, bem como uma aplicação móvel. Esta última deverá ser capaz de funcionar em dispositivos Android e iOS.

Em termos de escalabilidade, gostaríamos que o nosso sistema fosse suficientemente robusto para suportar milhares de pedidos efetuados por diversos dispositivos.

Numa fase inicial, e devido a limitações de recursos, teremos de considerar o seguinte:

- A aplicação web deverá ser executada num container Docker, a correr numa máquina virtual;
- O módulo de backend deverá ser executado num container Docker, a correr numa máquina virtual;
- A base de dados deverá ser executada num container Docker, a correr numa máquina virtual;
- Todas as comunicações entre dispositivos de clientes e o nosso sistema deverão ser feitas dentro da rede da Universidade de Aveiro (fisicamente ou através de uma VPN)

4.2. Vista da Arquitetura

Como já foi referido anteriormente, o nosso sistema irá ter 4 grandes componentes:

- **Frontend:** Aplicação web
- **Frontend:** Aplicação móvel
- **Backend:** REST API, processamento, lógica de negócio e acesso à informação
- **Persistência:** Base de dados relacional

Para cada um destes componentes, iremos utilizar uma tecnologia específica. Assim, tem-se que:

- Para a aplicação web, decidimos utilizar **ReactJS**, uma biblioteca JavaScript para construir interfaces de utilizador. Os principais benefícios do uso do ReactJS são a sua vasta biblioteca de modelos e elementos pré-fabricados, a sua facilidade de uso e capacidade de criar elementos gráficos e reativos de uma forma simples e eficaz. A sua popularidade também foi tida em conta.

- Para a aplicação mobile, decidimos utilizar o **React Native**. A principal vantagem desta tecnologia é a sua elevada portabilidade, que nos irá permitir desenvolver uma aplicação que funcione tanto para dispositivos Android bem como dispositivos iOS. De um modo geral, as restantes vantagens do React Native são as mesmas que as do ReactJS.

- Para o módulo de backend, iremos utilizar o **Spring Boot**. Em primeiro, por ser a ferramenta recomendada pelo docente da unidade curricular. No entanto, esta é uma tecnologia extremamente completa, no sentido em que permite:
  - Desenvolver uma REST API, para que as aplicações web e móvel possam comunicar com o nosso sistema
  - Processar toda a informação segundo a nossa lógica de negócio
  - Acessar a informação armazenada em persistência
  - Realizar testes unitários e de integração (e outros)

- Para a camada de persistência, iremos utilizar uma base de dados relacional em **MySQL**. A principal razão por termos escolhido este sistema de gestão de bases de dados deve-se, em grande parte, à familiaridade de todos os elementos do grupo na sua utilização. Outro motivo deve-se ao

facto de existirem diversas relações entre as entidades do nosso sistema, pelo que somos obrigados a utilizar uma base de dados relacional.

De seguida, apresentamos o diagrama tecnológico da nossa arquitetura:

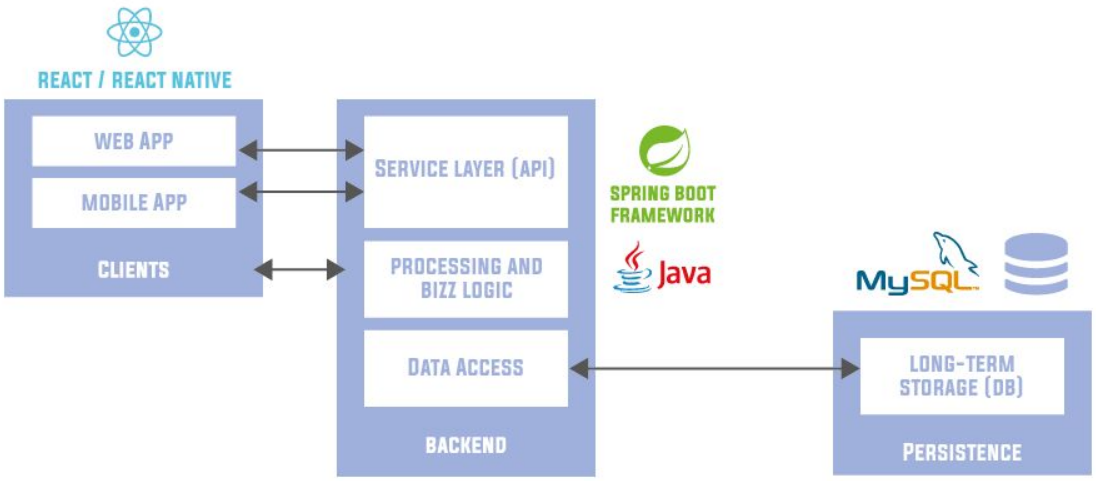

Figura 5: Diagrama de Arquitetura - Tecnologias

4.3. Arquitetura de instalação

Como já referimos anteriormente, iremos ter disponível uma máquina virtual para suportar os requisitos de deployment do nosso sistema. Como tal, decidimos que iremos utilizar 3 containers Docker, um para cada componente do nosso sistema, que irão ser executados na nossa máquina virtual. O procedimento a seguir está ilustrado no seguinte diagrama de instalação:

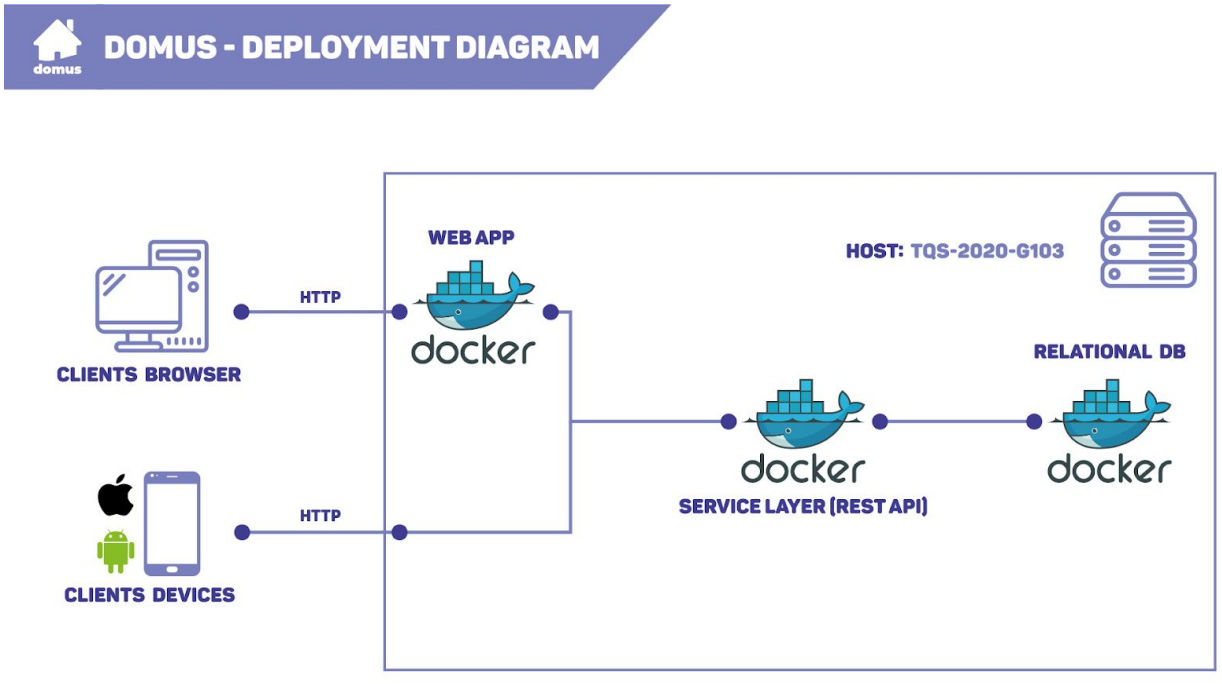

Figura 6: Diagrama de Arquitetura - Instalação

5. API para Developers

A REST API está feita de forma a permitir realizar pedidos, sendo utilizador ou sendo uma aplicação externa. As queries permitem um acesso controlado à BD fazendo ao mesmo tempo o tratamento de dados, caso necessário.

A descrição detalhada pode ser encontrada em: http://192.168.160.60:8080/api/swagger-ui.html.

URL	Método	Descrição
http://192.168.160.60:8080/api/houses	GET	Procura por todas as casas disponíveis (possibilidade de filtrar por parâmetros).
http://192.168.160.60:8080/api/houses/{id}	GET	Procura por uma casa por id.
http://192.168.160.60:8080/api/houses	POST	Adiciona uma nova casa.
http://192.168.160.60:8080/api/houses/{id}	PUT	Edita uma casa já existente.
http://192.168.160.60:8080/api/houses/{id}	DELETE	Elimina uma casa.
http://192.168.160.60:8080/api/locadores/{id}	GET	Procura por um locador pelo id.
http://192.168.160.60:8080/api/locatarios/{id}	GET	Procura por um locatário pelo id.
http://192.168.160.60:8080/api/locadores	POST	Adiciona um novo locador.
http://192.168.160.60:8080/api/locatarios	POST	Adiciona um novo locatário.
http://192.168.160.60:8080/api/locatarios/wishlist/{id}	GET	Procura pela wishlist pelo email do locatário.
http://192.168.160.60:8080/api/locatarios/wishlist	POST	Adiciona uma cada à wishlist de um locatário.
http://192.168.160.60:8080/api/locatarios/wishlist	DELETE	Elimina a casa da wishlist do locatário (parâmetros definidos no *payload* do pedido).

Tabela 4: Descrição de alguns endpoint da REST API

6. Referências e Recursos

- Gerar Documentação Swagger: https://www.baeldung.com/swagger-2-documentation-for-spring-rest-api
- Conceitos de Agile: https://www.atlassian.com/agile/project-management/epics-stories-themes