

UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA

## Roomie

*Diogo Pinto Ribeiro, A84442*

*João Nuno Cardoso Gonçalves de Abreu, A84802*

*José Diogo Xavier Monteiro, A83638*

*Rui Filipe Moreira Mendes, A83712*

*Vasco António Lopes Ramos, PG42852*



Engenharia de Aplicações

4º Ano, 2º Semestre

Departamento de Informática

14 de junho de 2021

# Índice

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Motivação e Objetivos . . . . .	1
1.2	Contextualização . . . . .	1
1.3	Caracterização das Personas . . . . .	2
1.3.1	Senhorio . . . . .	2
1.3.2	Inquilino . . . . .	3
<b>2</b>	<b>Modelação</b>	<b>5</b>
2.1	Inquéritos . . . . .	5
2.1.1	Inquilinos . . . . .	5
2.1.2	Senhorios . . . . .	7
2.2	Requisitos . . . . .	8
2.2.1	Funcionais . . . . .	8
2.2.2	Não Funcionais . . . . .	10
2.3	Diagrama de Use Cases . . . . .	11
2.4	Descrição dos Use Cases . . . . .	12
2.4.1	Adicionar Imóvel . . . . .	12
2.4.2	Ver Detalhes de Imóvel . . . . .	14
2.4.3	Ver Perfil de Inquilino . . . . .	15
2.4.4	Editar Perfil . . . . .	16
2.4.5	Avaliar Inquilino . . . . .	17
2.4.6	Listar Imóveis . . . . .	17
2.4.7	Submeter Candidatura . . . . .	18
2.4.8	Consultar Candidaturas . . . . .	20
2.5	Modelos de Tarefas . . . . .	20
2.5.1	Adicionar uma casa . . . . .	20
2.5.2	Candidatar a uma casa . . . . .	21
2.5.3	Aceitar uma candidatura de um inquilino . . . . .	22
2.6	Diagrama de Classes . . . . .	22
2.7	Proposta de Interfaces . . . . .	23

<b>3</b>	<b>Implementação</b>	<b>36</b>
3.1	Frontend . . . . .	36
3.1.1	Componentes . . . . .	36
3.1.2	Views . . . . .	36
3.1.3	Routes . . . . .	37
3.1.4	API . . . . .	37
3.2	Backend . . . . .	38
3.2.1	<i>Controllers</i> . . . . .	38
3.2.2	<i>Services</i> . . . . .	39
3.2.3	Segurança . . . . .	39
3.2.4	Documentação . . . . .	39
3.3	Base de Dados . . . . .	41
<b>4</b>	<b>Deployment</b>	<b>43</b>
<b>5</b>	<b>Análise de Carga</b>	<b>45</b>
5.1	Considerações Iniciais . . . . .	45
5.2	Cenários de Teste . . . . .	45
5.2.1	Landlord . . . . .	46
5.2.2	Tenant . . . . .	48
<b>6</b>	<b>Análise de Usabilidade</b>	<b>51</b>
6.1	Apresentação da Interface . . . . .	51
6.2	Princípios de Usabilidade . . . . .	53
6.2.1	<i>Learnability</i> . . . . .	53
6.2.2	<i>Flexibility</i> . . . . .	54
6.2.3	<i>Robustness</i> . . . . .	54
6.3	Heurísticas de Nielsen . . . . .	55
6.3.1	<i>Visibility of System Status</i> . . . . .	55
6.3.2	<i>Match between system and the real world</i> . . . . .	56
6.3.3	<i>User control and freedom</i> . . . . .	57
6.3.4	<i>Consistency and standards</i> . . . . .	57
6.3.5	<i>Error prevention</i> . . . . .	57
6.3.6	<i>Recognition rather than recall</i> . . . . .	57

6.3.7	<i>Flexibility and efficiency of use</i>	58
6.3.8	<i>Aesthetic and minimalist design</i>	58
6.3.9	<i>Help users recognize and recover from errors</i>	59
6.3.10	<i>Help and documentation</i>	59
<b>7</b>	<b>Conclusões e Trabalho Futuro</b>	<b>60</b>
<b>A</b>	<b>Docker Swarm/Stack: Especificação</b>	<b>61</b>

# **Lista de Figuras**

1	Cidades de residência de inquilinos . . . . .	6
2	Percentagem de inquilinos que utilizaram plataformas online . . . . .	6
3	Utilidade da existência de um histórico associado a um inquilino . .	6
4	Cidades nas quais senhorios possuem imóveis . . . . .	7
5	Percentagem de senhorios que utilizaram plataformas online . . . . .	7
6	Utilidade da existência de um histórico associado a um inquilino . .	8
7	Diagrama de Use Cases . . . . .	12
8	Modelo de Tarefas - Adicionar casa . . . . .	21
9	Modelo de Tarefas - Candidatar a casa . . . . .	21
10	Modelo de Tarefas - Aceitar candidatura de inquilino . . . . .	22
11	Diagrama de Classes . . . . .	23
12	Proposta de interface para a página principal . . . . .	24
13	Proposta de interface para o <i>login</i> . . . . .	24
14	Proposta de interface para o primeiro momento do processo de registo para senhorios e inquilinos . . . . .	25
15	Proposta de interface para o registo do senhorio . . . . .	25
16	Proposta de interface para o registo do inquilino . . . . .	26
17	Proposta de interface para a confirmação do registo . . . . .	26
18	Proposta de interface para a página principal do inquilino . . . . .	27
19	Proposta de interface para a página principal do senhorio . . . . .	27
20	Proposta de interface para o perfil do próprio inquilino . . . . .	28
21	Proposta de interface para o perfil do próprio senhorio . . . . .	28
22	Proposta de interface para o perfil de outros inquilinos . . . . .	29
23	Proposta de interface para o perfil de outros senhorios . . . . .	29
24	Proposta de interface para um inquilino avaliar inquilinos . . . . .	30
26	Proposta de interface para procurar imóveis por parte de um senhorio	30
27	Proposta de interface para um senhorio poder ver as suas casas . .	31
28	Proposta de interface para um inquilino poder ver o seu historial de arrendamentos . . . . .	31
29	Proposta de interface para um inquilino poder ver um determinado imóvel . . . . .	32

30	Proposta de interface para um senhorio poder ver um determinado imóvel . . . . .	32
31	Proposta de interface para um senhorio adicionar um imóvel . . . . .	33
32	Proposta de interface para um senhorio editar um imóvel . . . . .	33
33	Proposta de interface para um senhorio confirmar a adição de uma casa . . . . .	34
35	Proposta de interface para um senhorio verificar as candidaturas aos seus imóveis . . . . .	34
36	Proposta de interface para um inquilino verificar as suas candidaturas . . . . .	35
37	Página Inicial Documentação . . . . .	40
38	Documentação para Login . . . . .	40
39	Diagrama EER . . . . .	42
40	Arquitetura de <i>Deployment</i> . . . . .	44
41	Evolução do Número de Utilizadores . . . . .	47
42	Total de Pedidos por Segundo . . . . .	48
43	Evolução dos Tempos de Resposta . . . . .	48
44	Evolução do Número de Utilizadores . . . . .	49
45	Total de Pedidos por Segundo . . . . .	49
46	Evolução dos Tempos de Resposta . . . . .	50
47	Exemplo de um <i>modal</i> . . . . .	56
48	Exemplo de um <i>calendar picker</i> . . . . .	56
49	Exemplo de <i>input validation</i> . . . . .	57
50	Exemplo de uma <i>navbar</i> . . . . .	58
51	Página de pesquisa de imóveis . . . . .	58

## **Lista de Tabelas**

1	Especificação do Use Case Adicionar Imóvel . . . . .	13
2	Especificação do Use Case Ver Detalhes de Imóvel . . . . .	14
3	Especificação do Use Case Ver Perfil de Inquilino . . . . .	15
4	Especificação do Use Case Editar Perfil . . . . .	16
5	Especificação do Use Case Avaliar Inquilino . . . . .	17
6	Especificação do Use Case Listar Imóveis . . . . .	18
7	Especificação do Use Case Submeter Candidatura . . . . .	19
8	Especificação do Use Case Consultar Candidaturas . . . . .	20
9	Resumo dos resultados do cenário de teste para o Landlord . . . . .	47
10	Resumo dos resultados do cenário de teste para o Tenant . . . . .	49

# 1 Introdução

## 1.1 Motivação e Objetivos

Durante toda a história da humanidade, uma das questões mais importantes foi a procura de habitação. Desde motivos básicos como sobrevivência até à procura de uma habitação maior ou com características específicas. Muitas das vezes, por motivos pessoais ou profissionais, existe a necessidade de procurar uma habitação num local específico ou com determinadas características, sendo que para a maioria das pessoas não lhes é possível adquirir um imóvel nessas situações.

O conceito de arrendar um imóvel tem já uma determinada maturidade, dado que cada vez mais é uma opção na vida das pessoas. Em muitas destas situações existe a necessidade de um imóvel ser arrendado por mais do que uma pessoa, quer por motivos financeiros, quer pelas características do imóvel. Assim, é bastante recorrente existir uma certa dificuldade em encontrar mais pessoas com quem dividir o imóvel, dado que geralmente a procura é feita num ambiente ou localidade que nos é novo ou estranho. Este processo acaba por se tornar complexo e levanta diversas dúvidas, quer por parte dos Senhorios, que não conhecem as pessoas a quem arrendam os imóveis, quer por parte dos Inquilinos, que não conhecem as pessoas com quem irão dividir o imóvel.

Surge assim a necessidade de uma plataforma que permita agilizar o processo de encontrar pessoas com quem dividir o arrendamento de um imóvel.

## 1.2 Contextualização

No contexto do perfil de Engenharia de Aplicações foi proposta a realização de um projeto que permitisse aplicar as diferentes tecnologias e conhecimentos exploradas no mesmo. Depois de alguma deliberação, o grupo chegou à ideia de desenvolver uma aplicação *web* designada **Roomie**.

A Roomie é uma **Plataforma de Procura e Disponibilização de Imobiliário para Arrendamento** que procura não só facilitar o processo de encontrar uma habitação como também o processo da escolha do ambiente em que se pre-

tende viver. Este processo só será possível através da intervenção de ambas as partes: o **Senhorio** que terá de vetar a lista de candidatos para que sejam aceites só aqueles que ele acredite que encaixem nos seus requisitos, e os **Inquilinos** que se irão candidatar às habitações que acreditam serem adequadas às suas necessidades.

Para este efeito, algumas das características que esta plataforma terá de apresentar são as seguintes:

- Permitir o registo por parte dos Inquilinos e dos Senhorios;
- Permitir aos Senhorios adicionar e remover os seus imóveis, em conjunção com todas as suas características;
- Permitir aos Inquilinos consultar as listagens dos imóveis, tendo acesso aos perfis e avaliações dos seus possíveis futuros colegas de quarto;
- Permitir aos Inquilinos realizar candidaturas;
- Permitir aos Senhorios aceitar e rejeitar candidaturas, tendo acesso às avaliações dos candidatos;
- Por fim, permitir aos Senhorios e Inquilinos avaliar os Inquilinos da sua habitação.

### 1.3 Caracterização das Personas

Com o intuito de se fazer uma melhor caracterização dos utilizadores da plataforma, decidiu-se fazer uma pequena investigação capaz de nos auxiliar a identificar o público alvo e os tipos de utilização que poderão surgir.

De seguida, apresentam-se duas *personas* que visam representar e ilustrar as necessidades e objetivos que a plataforma supre:

#### 1.3.1 Senhorio

**Nome:** Pedro Alves

**Idade:** 52

**Morada:** Póvoa de Lanhoso

**NIF:** 273955241

O Sr. Pedro tem 52 anos e é um ex operário fabril que, tal como muitos outros, entrou no fundo de desemprego por consequência da pandemia “*Covid-19*” que tem vindo a afectar a economia global. Após o falecimento da sua mãe, tornou-se no recém herdeiro de uma moradia localizada nas redondezas da Universidade do Minho. Sendo que a casa está inocupada e apta para habitação, acredita ter nas suas mãos uma boa oportunidade para criar algum rendimento extra.

Após alguma deliberação e tendo em conta a localização e o elevado número de divisões do seu novo imóvel, o Sr. Pedro chegou à conclusão de que uma faixa etária mais jovem, mais especificamente, estudantes universitários, fariam um fantástico público alvo. No entanto, devido à sua falta de experiência, não faz ideia de quais as vias de negociação que tem ao seu dispor. Por fim, para piorar a situação, tem ouvido nas notícias os problemas que têm ocorrido com as praxes académicas e tem medo do nível de maturidade e respeito dos seus possíveis futuros inquilinos.

Tendo exposto este problema com a sua família, o seu filho deu-lhe a conhecer uma aplicação chamada Roomie. Interessado, decide experimentar o serviço e disponibiliza o seu imóvel, com todas as características que considera essenciais, no mesmo. Após algum tempo de espera, apareceram candidatos cuja avaliação e interesses aparentam ser compatíveis com os seus e passa à realização do contrato, sendo assim satisfeitas as suas necessidades.

### 1.3.2 Inquilino

**Nome:** Rui Fernandes

**Idade:** 22

**Morada:** Lisboa

**NIF:** 263282996

O Rui tem 22 anos, é natural de Lisboa e é estudante do Mestrado Integrado em Engenharia Informática na Universidade do Minho. Por estar tão longe de

casa, e não tendo acesso a qualquer bolsa social, precisa naturalmente de uma habitação em Braga.

Para o ajudar na procura de casas que satisfaçam as suas necessidades, tendo em conta as suas possibilidades, já há algum tempo que o Rui utiliza a aplicação Roomie que lhe permite procurar casas de acordo com os seus requisitos, bem como perceber se as pessoas com quem irá partilhar a casa serão bons colegas, podendo assim tomar uma decisão mais informada e ponderada.

Tendo já estado em 3 casas diferentes desde que começou a utilizar a aplicação, o Rui conta já com um número considerável de avaliações, todas elas bastante positivas, tanto por parte dos senhorios como dos seus colegas de casa, pelo que cada vez se torna mais fácil para o Rui ser aceite numa nova casa por um senhorio que não o conhece, o que facilita bastante a sua procura e aceitação em novas casas.

O Rui, estando quase a terminar o seu curso, como tem a vontade de continuar em Braga, pretende continuar a utilizar a aplicação, pois, dada a sua excelente classificação, já percebeu que consegue casas com melhores condições em termos que lhe são mais favoráveis (tanto a nível dos colegas com quem partilha a casa, bem como a nível de preços, visto que consegue quase sempre negociar com os senhorios preços um pouco mais baixos que os referidos nos anúncios).

## 2 Modelação

Na etapa de Modelação concentrámos os nossos esforços na definição de todas as necessidades e objetivos inerentes à nossa plataforma. Esta etapa engloba a análise de inquéritos efetuados a potenciais utilizadores, o levantamento de requisitos funcionais e não-funcionais, bem como a elaboração de um Diagrama de *Use Cases* e a especificação dos mesmos.

### 2.1 Inquéritos

Uma das tarefas por nós realizadas foi a elaboração de inquéritos para cada um dos atores do nosso sistema: Inquilino e Senhorio. Com estes inquéritos procurámos entender um pouco melhor o que é mais relevante para a nossa plataforma do ponto de vista de potenciais utilizadores, bem como, perceber um pouco “quem são” os nossos potenciais utilizadores.

As respostas destes inquéritos irão de certo modo influenciar algumas das decisões tomadas nesta etapa de modelação, sendo esse o foco dos mesmos.

#### 2.1.1 Inquilinos

Relativamente aos inquéritos efetuados a inquilinos, obtivemos um **total de 43 respostas**, sendo que a **média de idades** dos entrevistados é de **22 anos**, num **intervalo de idades** entre **17 e 29**.

Alguns dos dados recolhidos passaram pela cidade de residência do inquilino, se já utilizou plataformas *online* para arrendamento de imóveis e a utilidade de possuir um histórico de avaliação de potenciais colegas de casa.

Alguns dos problemas identificados por inquilinos que já utilizaram plataformas online passam pela **falta de informação nos anúncios** e pelo facto de algumas serem **pouco intuitivas** de utilizar. Outra questão relevante que colocámos no inquérito consistiu em obter as características que seriam mais interessantes de ser avaliadas num colega de casa, tendo a maioria das respostas consistido em: **limpeza, arrumação, ser amigável, e, privacidade**.

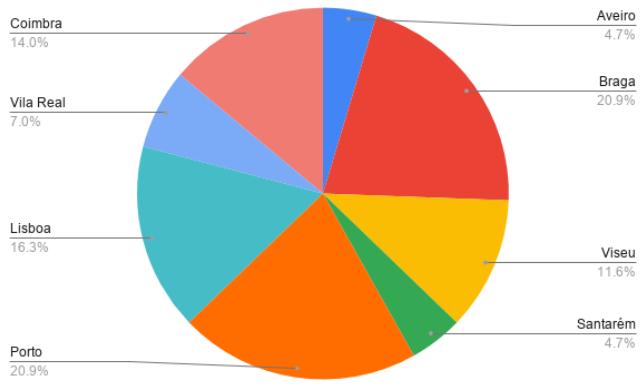


Figura 1: Cidades de residência de inquilinos

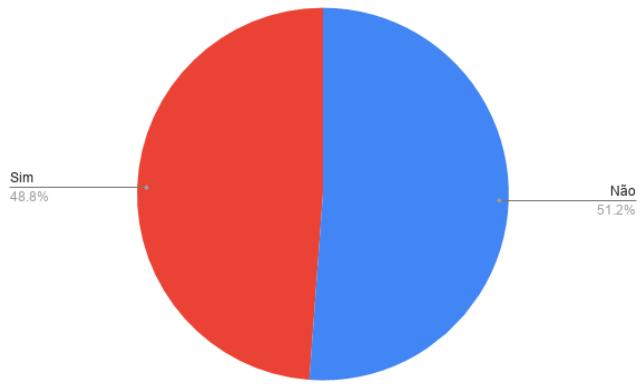


Figura 2: Percentagem de inquilinos que utilizaram plataformas online

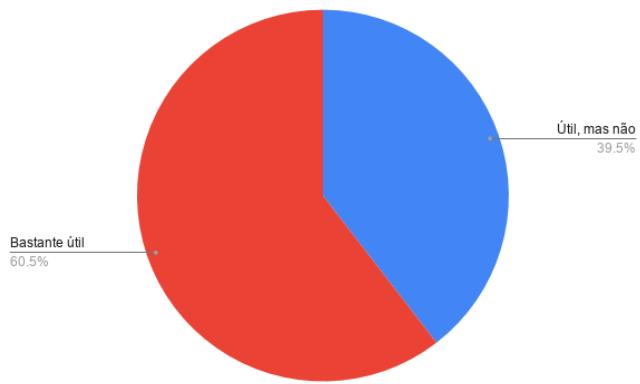


Figura 3: Utilidade da existência de um histórico associado a um inquilino

### 2.1.2 Senhorios

Relativamente aos inquéritos efetuados a senhorios, obtivemos um **total de 17 respostas**, sendo que a **média de idades** dos entrevistados é de **47 anos**, num **intervalo de idades** entre **31 e 67**.

Alguns dos dados recolhidos passaram pela cidade onde o senhorio possui imóveis para arrendamento, se já utilizou plataformas *online* para arrendamento dos seus imóveis e a utilidade de possuir um histórico de avaliação de um potencial inquilino.

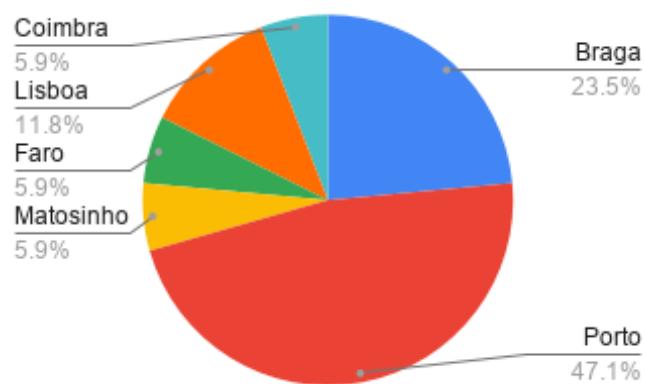


Figura 4: Cidades nas quais senhorios possuem imóveis

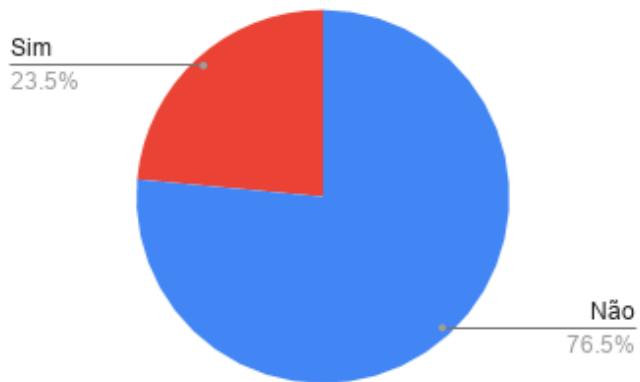


Figura 5: Percentagem de senhorios que utilizaram plataformas online

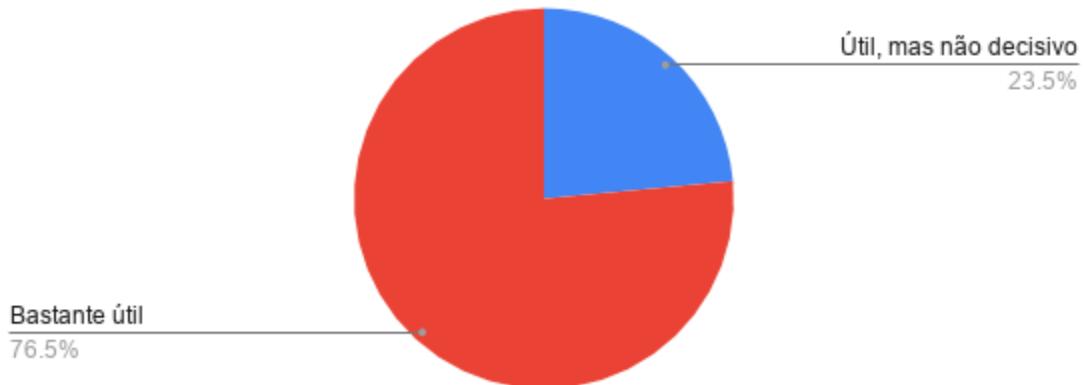


Figura 6: Utilidade da existência de um histórico associado a um inquilino

Outra questão relevante que colocámos no inquérito consistiu em obter as características que seriam mais interessantes estar no histórico de um inquilino, tendo a maioria das respostas consistido em: **assiduidade nos pagamentos, respeito pelo imóvel e a limpeza** por parte do inquilino.

## 2.2 Requisitos

Para efetuar o levantamento de requisitos considerámos as respostas obtidas por parte de utilizadores entrevistados e os objetivos por nós definidos, tendo obtido os seguintes resultados.

### 2.2.1 Funcionais

De modo a estruturar os requisitos funcionais, optámos por agrupar os requisitos levantados por 2 grupos relativos aos atores Inquilino e Senhorio.

#### Senhorio

1. **Criar conta na plataforma**, indicando o nome, email, morada, idade, nif, e password.
2. **Iniciar sessão** na plataforma, indicando email e password.
3. **Terminar sessão** na plataforma.

4. **Listar** os seus **imóveis**.
5. **Adicionar um imóvel** para arrendamento na plataforma, indicando a morada, tipologia, descrição, número de vagas, preço(s) por vaga, imagens, características.
6. **Remover um imóvel** da plataforma.
7. **Atualizar um imóvel**.
8. **Consultar candidaturas** de Inquilinos.
9. **Aceitar uma candidatura** de um Inquilino.
10. **Rejeitar uma candidatura** de um Inquilino.
11. **Avaliar um Inquilino** relativamente a diversas classificações como: ...
12. **Visualizar perfis de Inquilinos**.
13. **Visualizar o seu perfil**.
14. **Editar o perfil**.
15. **Apagar o perfil**.

## Inquilino

1. **Criar conta na plataforma**, indicando o nome, email, idade, sexo, profissão, nacionalidade, nif, e password, características.
2. **Iniciar sessão** na plataforma, indicando email e password.
3. **Terminar sessão** na plataforma.
4. **Procurar imóveis** para arrendar.
5. **Filtrar listagens** de imóveis para arrendar,
6. **Visualizar detalhes de um imóvel** tais como as características do mesmo, Inquilinos que já habitem o mesmo, e informações do Senhorio.
7. **Candidatar a uma vaga** disponível num imóvel.

8. **Consultar candidaturas** efetuadas.
9. **Cancelar candidaturas** efetuadas.
10. **Visualizar perfis de outros Inquilinos.**
11. **Avaliar Inquilinos** com quem tenha partilhado um imóvel em aspetos como: simpatia, respeito, arrumação/limpeza.
12. **Visualizar o seu perfil.**
13. **Editar o perfil.**
14. **Apagar o perfil.**

### 2.2.2 Não Funcionais

Relativamente à plataforma, consideramos que seria interessante que possua as seguintes características.

1. A plataforma deve estar funcional em diversos *browsers* como Chrome, Firefox, Edge e Safari.
2. O sistema deve efetuar validação de emails e NIF.
3. O sistema deve efetuar validação de passwords.
4. O sistema não deve permitir anúncios de imóveis sem imagens e detalhes completos.
5. O sistema deve ser fácil de utilizar por um utilizador não familiarizado.
6. O sistema deve respeitar a privacidade dos utilizadores, apenas divulgando dados necessários.
7. O sistema deverá possuir uma elevada disponibilidade, isto é, estar operacional pelo menos 99,95% do ano.
8. O sistema deverá suportar a utilização de um elevado número de utilizadores (pelo menos 500 Senhorios/Inquilinos).

9. O utilizador deverá conseguir visualizar páginas em menos de 3 segundos, incluindo os atrasos de rede.
10. O utilizador deverá conseguir realizar operações em menos de 3 segundos, incluindo os atrasos de rede.

## 2.3 Diagrama de Use Cases

O processo de desenvolvimento do Diagrama de Casos de Uso, a par dos Requisitos Funcionais apresentados na Secção 2.1.1, tratou-se de um processo iterativo e incremental, sempre com o intuito de elaborar uma plataforma com funcionalidades que realmente vão ao encontro das necessidades dos seus futuros utilizadores.

Após várias deliberações, chegamos ao Diagrama de Casos de Uso presente na figura 7. Através deste diagrama, é possível perceber que o sistema terá 2 atores: o Senhorio (*Landlord*) e o Inquilino (*Tenant*).

Tanto o *Landlord* como o *Tenant* têm como funcionalidades o **Registo** na plataforma, a **Autenticação** (login/logout), a possibilidade de **Editar o seu perfil** e **Apagar a conta** dentro da opção de **Visualizar o seu perfil**. Também podem **Visualizar perfil de um Tenant** e são capazes de **Avaliar um Tenant** (que no caso do ator ser o *Tenant* este avalia outros *Tenants* com os quais partilhou casa).

O *Landlord* não só é capaz de **Adicionar um novo imóvel** à plataforma, como **Remover o imóvel**. É também capaz de **Listar os seus imóveis** e **Atualizar detalhes** relevantes no caso de engano ou simplesmente porque necessita. Também tem acesso a uma **Lista de candidaturas** feitas pelos *Tenants* que pretendem ocupar uma vaga num dos seus imóveis, juntamente com a opção de **Aceitar ou Rejeitar a candidatura**.

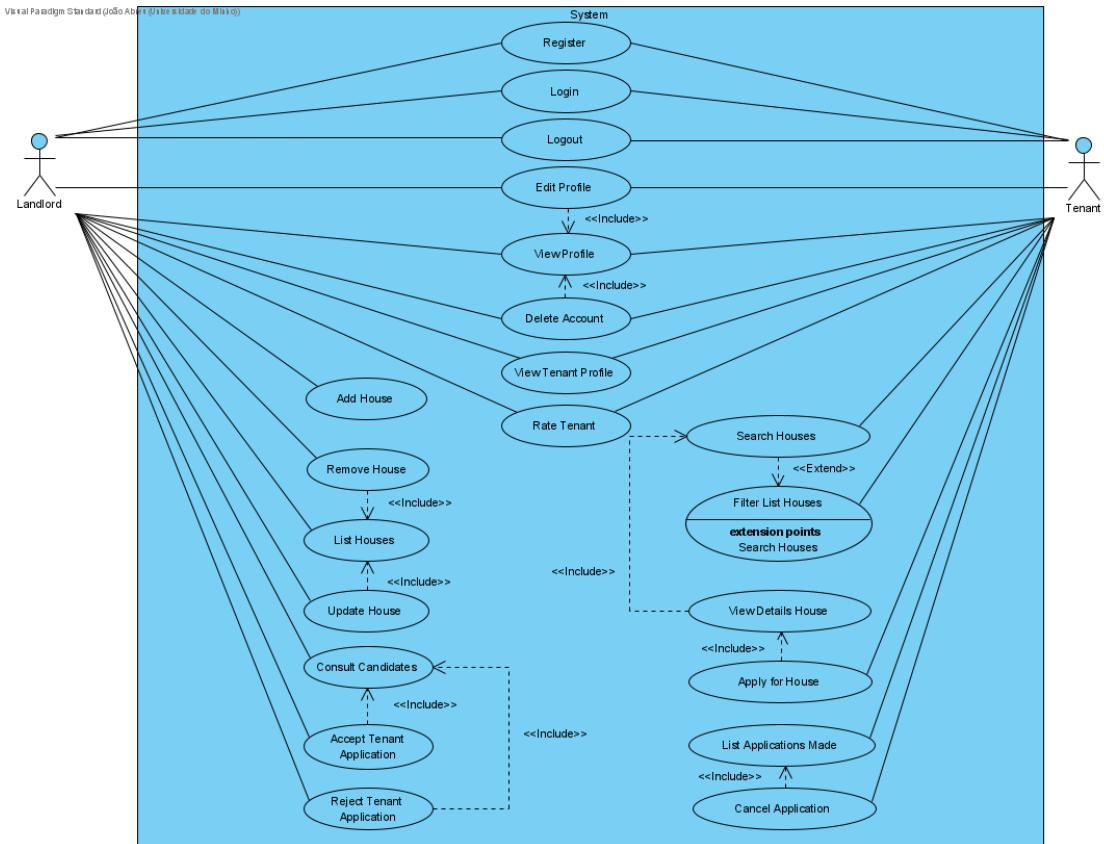


Figura 7: Diagrama de Use Cases

Por sua vez, o **Tenant** tem a possibilidade de se **Candidatar a um imóvel** mas necessita primeiro de **Procurar por imóveis** disponíveis na plataforma. É também capaz de **Filtrar a lista de imóveis** resultantes da procura e de **Ver mais detalhes sobre um imóvel**. Por fim, também pode **Listar candidaturas efetuadas** e a opção de **Cancelar uma candidatura** caso não esteja mais interessado em ocupar a vaga desse imóvel.

## 2.4 Descrição dos Use Cases

Iremos de seguida proceder à especificação dos *Use Cases* mais relevantes previamente apresentados.

### 2.4.1 Adicionar Imóvel

Use Case	Adicionar Imóvel	
Autor	Senhorio	
Pré Condição	Ter Sessão Iniciada	
Fluxo Normal	Input Ator	Resposta Sistema
		1. Apresenta a página de adição de um imóvel
	2. Insere Nome do Imóvel	
	3. Insere Morada do Imóvel	
	4. Insere Número de Quartos do Imóvel	
	5. Insere Número de Quartos Disponíveis	
	6. Insere Número de Casas de Banho do Imóvel	
	7. Insere Preço por Quarto	
	8. Insere Descrição do Imóvel	
	9. Adiciona <i>Features</i> do Imóvel	
	10. Adiciona Imagens do Imóvel	
	11. Seleciona a Opção de Confirmação	
Exceção 1 Valores por preencher		12. Adiciona o Imóvel na Plataforma
	11.1 Apresenta mensagem de erro sobre parâmetros por preencher	
Pós Condição	Imóvel Adicionado na Plataforma	

Tabela 1: Especificação do Use Case Adicionar Imóvel

O ator Senhorio, de modo a poder alugar um imóvel, necessita de o inserir na plataforma. Para tal, necessita de aceder ao formulário de adição e preencher os dados referentes ao imóvel. Após o correto preenchimento de todos os campos poderá submeter a formulário, finalizando assim a adição do imóvel na plataforma.

#### 2.4.2 Ver Detalhes de Imóvel

Use Case	Ver Detalhes de Imóvel	
Autor	Inquilino	
Pré Condição	Ter Sessão Iniciada e Existirem Imóveis na Pesquisa	
Fluxo Normal	Input Ator	Resposta Sistema
	1. Seleciona opção de ver detalhes de imóvel	
		2. Apresenta página com detalhes do imóvel
Pós Condição	Visualizar Detalhes de Imóvel	

Tabela 2: Especificação do Use Case Ver Detalhes de Imóvel

Quando um Inquilino procura um Imóvel, é normal que pretenda conhecer todos os seus detalhes. Dado que numa pesquisa não são exibidos todos os detalhes dos Imóveis, existe a necessidade de criar uma página com essa informação. Ao consultar a página com os detalhes do Imóvel, o Inquilino passa a ter acesso a toda a informação e operações acerca do mesmo.

### 2.4.3 Ver Perfil de Inquilino

Use Case	Ver Perfil de Inquilino	
Autor	Inquilino ou Senhorio	
Pré Condição	Ter Sessão Iniciada	
Fluxo Normal	Input Ator	Resposta Sistema
	1. Seleciona opção de ver perfil de inquilino	2. Apresenta página com o perfil do inquilino
Pós Condição	Visualizar Perfil de Inquilino	

Tabela 3: Especificação do Use Case Ver Perfil de Inquilino

No caso de um Senhorio ou Inquilino pretender obter mais informações acerca de um dado Inquilino, torna-se importante a existência de um perfil. Ao aceder ao perfil de um Inquilino, qualquer Ator passa a ter acesso à informação pública referente ao Inquilino cujo perfil é consultado.

#### 2.4.4 Editar Perfil

Use Case	Editar Perfil	
Autor	Inquilino ou Senhorio	
Pré Condição	Ter Sessão Iniciada	
Fluxo Normal	Input Ator	Resposta Sistema
		1. «include» View Profile
	2. Seleciona opção de editar perfil	
		3. Apresenta a página do perfil com os campos possíveis a serem alterados
	4. Atualiza campos pretendidos	
	5. Seleciona a Opção de Confirmação	
		5. Atualiza os valores na Plataforma
Exceção 1 Valores por preencher		5.1 Apresenta mensagem de erro sobre parâmetros por preencher
	5.2 Insere valores em falta	
Pós Condição	Perfil Atualizado	

Tabela 4: Especificação do Use Case Editar Perfil

No caso de um Senhorio ou Inquilino se enganar ou simplesmente pretender atualizar os dados sobre o seu perfil, este deve ter o direito para tal. No entanto, nem todos os campos poderão ser livremente alterados, como por exemplo o NIF. Também não será permitido novos valores em branco.

#### 2.4.5 Avaliar Inquilino

Use Case	Avaliar Inquilino	
Autor	Inquilino ou Senhorio	
Pré Condição	Ter Sessão Iniciada e Inquilino Existe	
Fluxo Normal	Input Ator	Resposta Sistema
		1. «include» View Tenant Profile
	2. Seleciona opção de avaliar inquilino	
		3. Apresenta página de avaliação do inquilino
	4. Escolher avaliação	
		5. Adiciona a nova avaliação à Plataforma
Pós Condição	Inquilino com nova avaliação	

Tabela 5: Especificação do Use Case Avaliar Inquilino

A avaliação do Inquilino pode ser feita pelos 2 atores do sistema, pois cada um terá a sua versão que deve ser igualmente ponderada para a avaliação total do Inquilino.

#### 2.4.6 Listar Imóveis

Para o Senhorio ter acesso aos seus imóveis registados na plataforma este deve ter um acesso rápido à listagem dos mesmos. Aqui, poderá aceder a cada um dos seus imóveis e atualizá-los/apagá-los caso pretenda.

Use Case	Listar Imóveis	
Autor	Senhorio	
Pré Condição	Ter Sessão Iniciada	
Fluxo Normal	Input Autor	Resposta Sistema
	1. Seleciona opção de listar imóveis	
		2. Apresenta página com os seus imóveis
Pós Condição	Inquilino com nova avaliação	

Tabela 6: Especificação do Use Case Listar Imóveis

#### 2.4.7 Submeter Candidatura

O ator Inquilino, de maneira a ficar numa casa, necessita de fazer um pedido de candidatura ao Senhorio. Este poderá mais tarde aceitar ou rejeitar o pedido analisando o perfil do Inquilino.

Use Case	Submeter Candidatura	
Ator	Inquilino	
Pré Condição	Ter Sessão Iniciada e Existir Vaga no Imóvel	
Fluxo Normal	Input Ator	Resposta Sistema
	1. Seleciona uma casa e candidata-se	
		2. Apresenta confirmação da candidatura
		3. Regista novo pedido
Exceção 1 Vaga/Imóvel foi apagado		2.1 Apresenta mensagem de erro sobre a remoção da vaga/imóvel a qual seria feita a candidatura
		2.2 Candidatura é cancelada
Pós Condição	Imóvel com nova candidatura	

Tabela 7: Especificação do Use Case Submeter Candidatura

#### 2.4.8 Consultar Candidaturas

Use Case	Consultar Candidaturas	
Autor	Senhorio	
Pré Condição	Ter Sessão Iniciada	
Fluxo Normal	Input Autor	Resposta Sistema
	1. Seleciona opção de consultar candidaturas	2. Mostra candidaturas efetuadas aos seus imóveis
Pós Condição	Lista de candidaturas exibida	

Tabela 8: Especificação do Use Case Consultar Candidaturas

De forma a que o Senhorio saiba de uma maneira eficiente as candidaturas que recebeu e respetivo imóvel pretendido este deve poder aceder a uma lista com todas as candidaturas e, mais tarde, a partir daí, aceitar ou rejeitar os pedidos.

### 2.5 Modelos de Tarefas

Como acréscimo à especificação anterior dos casos de uso, decidiu-se criar alguns modelos de tarefas, para funcionalidades mais proeminentes, de forma a representar-se visualmente e de forma mais detalhada as tarefas a realizar, a sua ordem e a entidade que as realiza.

#### 2.5.1 Adicionar uma casa

Como se pode ver na figura 8, para adicionar uma casa, o *Landlord* começa por selecionar o botão “Add new House”, de seguida preencher os campos de informação por qualquer ordem, para, no fim, confirmar a submissão da casa.

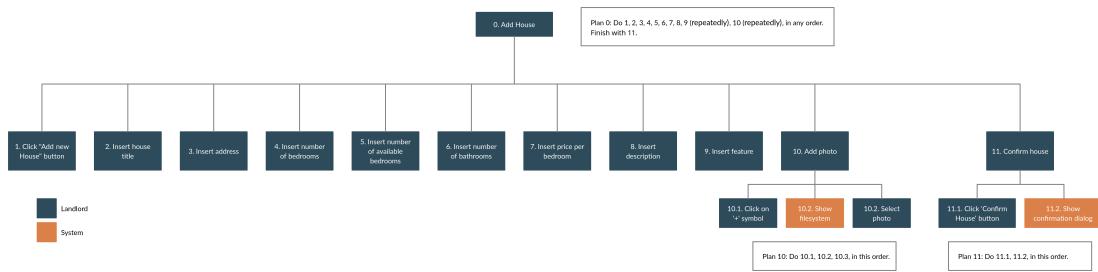


Figura 8: Modelo de Tarefas - Adicionar casa

### 2.5.2 Candidatar a uma casa

Como se pode ver na figura 9, para se candidatar a uma casa, o *Tenant* começa por escolher a casa a que se quer candidatar, selecionando da lista pré-concebida ou de uma lista filtrada de casas. Após selecionar a casa, basta carregar no botão “*Apply for house!*” e aguardar a confirmação por parte do Sistema.

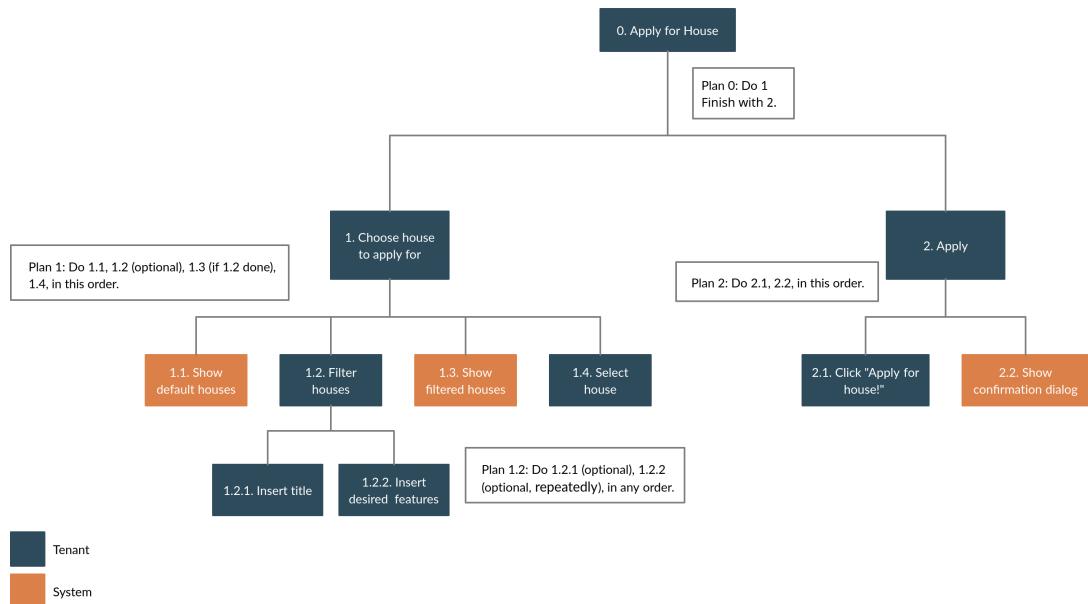


Figura 9: Modelo de Tarefas - Candidatar a casa

### 2.5.3 Aceitar uma candidatura de um inquilino

Como se pode ver na figura 10, para aceitar a candidatura de um inquilino a uma das suas casas, o *Landlord* começa por ir para a página das candidaturas, selecionando o botão “*Check applications for my Houses*”, escolhe a candidatura que quer avaliar/aceitar (selecionando da lista pré-concebida ou de uma lista filtrada) e, por fim, após selecionar a candidatura, carrega no botão “*Accept Application*”. Após aceite, o Sistema fornece uma mensagem de confirmação da operação.

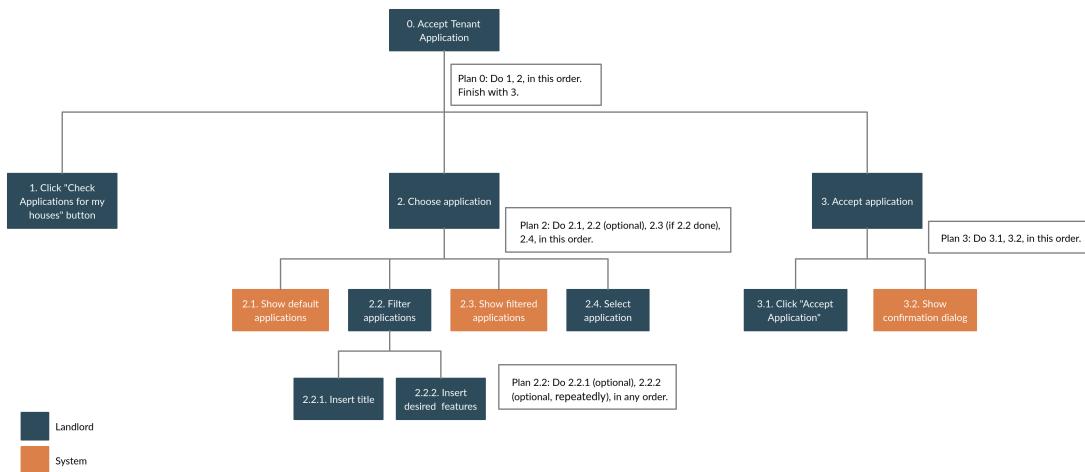


Figura 10: Modelo de Tarefas - Aceitar candidatura de inquilino

## 2.6 Diagrama de Classes

De forma a ser possível representar a conceitualização do sistema: as suas entidades e relações, desenvolveu-se um diagrama de classes. Como se pode ver na figura 11, existem três principais entidades: *Tenant*, *Landlord* e *House*. Ao nível das entidades que representam relações as mais notórias são:

- **LandlordEvaluation:** os *landlords* avaliam os seus *tenants*.
- **TenantEvaluation:** os *tenants* avaliam os outros *tenants* com quem partilham casa.
- **Application:** os *tenants* candidatam-se a casas.
- **RentHistory:** os *tenants* foram aceites e estão a morar numa casa alugada.

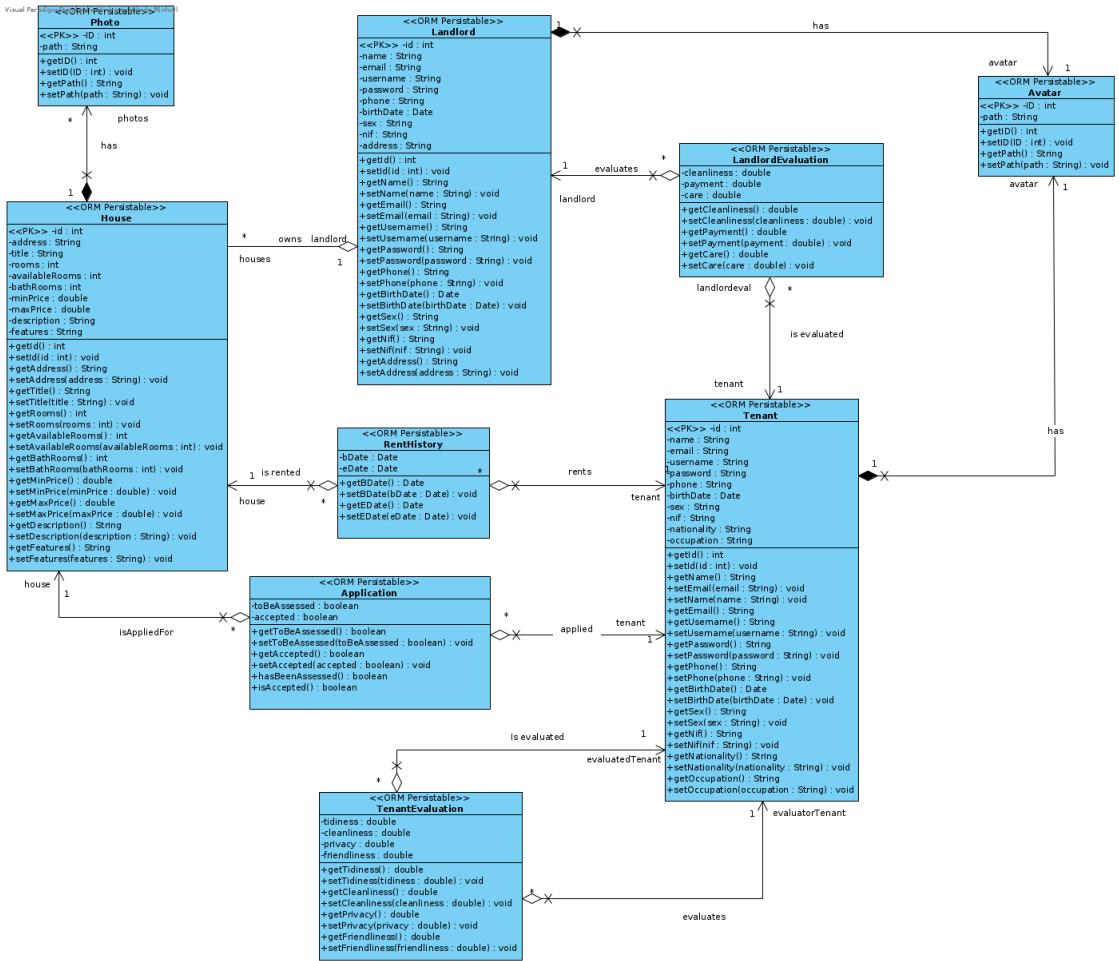


Figura 11: Diagrama de Classes

O diagrama presente na figura 11 já se encontra numa fase final, em que já está mais interligado com a tecnologia usada, dadas as anotações de persistências tanto ao nível das entidades como das relações.

## 2.7 Proposta de Interfaces

Para desenvolvemos as propostas de interfaces utilizámos o software *Figma* por considerarmos que era fácil de trabalhar e pela sua capacidade de gerir projetos, permitindo o desenvolvimento por vários utilizadores em simultâneo. As propostas de interfaces possuem grande utilidade, como por exemplo, orientar o código que precisamos de desenvolver a nível de *frontend* da *Roomie*, assim como também

permitir ter uma melhor ideia da forma como os utilizadores irão interagir com o sistema permitindo desenvolver os diagramas de tarefas e as especificações do uses cases com um maior grau de detalhe. Para além disso, é possível interagir diretamente com os protótipos no seguinte [link](#).

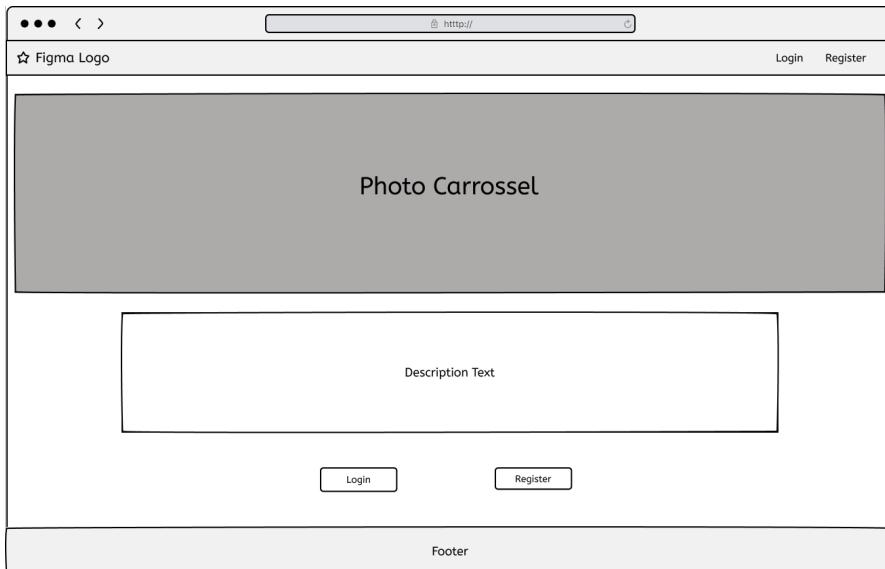


Figura 12: Proposta de interface para a página principal

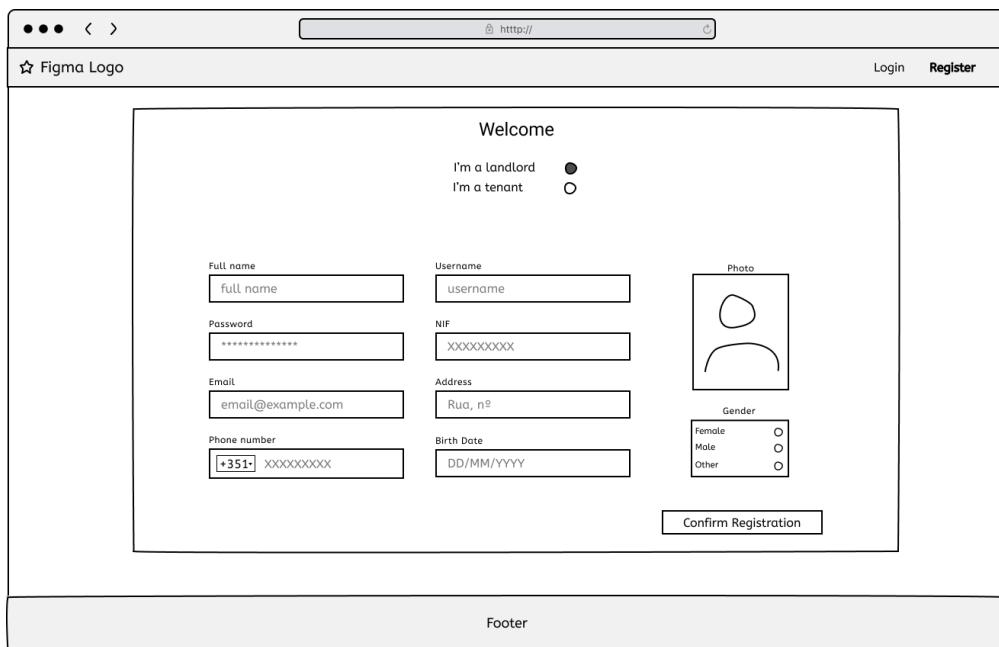


Figura 13: Proposta de interface para o *login*



A wireframe of a web browser window showing a registration interface. The header includes a Figma logo, a search bar with placeholder text 'http://', and navigation buttons. The top right has 'Login' and 'Register' links. The main content area is titled 'Welcome' and contains two radio buttons: 'I'm a landlord' (selected) and 'I'm a tenant'. A large empty rectangular box occupies most of the center. The footer is a light gray bar.

Figura 14: Proposta de interface para o primeiro momento do processo de registo para senhorios e inquilinos



A wireframe of a web browser window showing a registration form. The header is identical to Figure 14. The main content area is titled 'Welcome' and contains two radio buttons: 'I'm a landlord' (selected) and 'I'm a tenant'. Below this are several input fields: 'Full name' (text box), 'Username' (text box), 'Photo' (placeholder image of a person's face), 'Password' (text box with asterisks), 'NIF' (text box with asterisks), 'Address' (text box), 'Email' (text box), 'Rua, nº', 'Phone number' (text box), '+351- XXXXXXXX', 'Birth Date' (text box), 'Gender' (radio buttons for Female, Male, Other), and a 'Confirm Registration' button. A large empty rectangular box is at the bottom. The footer is a light gray bar.

Figura 15: Proposta de interface para o registo do senhorio

The registration form interface is titled "Welcome". It includes fields for "full name" (placeholder: "full name"), "Password" (placeholder: "\*\*\*\*\*"), "Email" (placeholder: "email@example.com"), "Phone number" (placeholder: "+351- XXXXXXXX"), "Nationality" (placeholder: "Portuguese"), "Username" (placeholder: "username"), "NIF" (placeholder: "XXXXXX"), "Birth Date" (placeholder: "DD/MM/YYYY"), "Occupation" (placeholder: "Student"), and "Gender" (radio buttons for "Female", "Male", and "Other"). There is also a placeholder "Photo" with a user icon. A "Confirm Registration" button is at the bottom right.

Figura 16: Proposta de interface para o registo do inquilino



Figura 17: Proposta de interface para a confirmação do registo

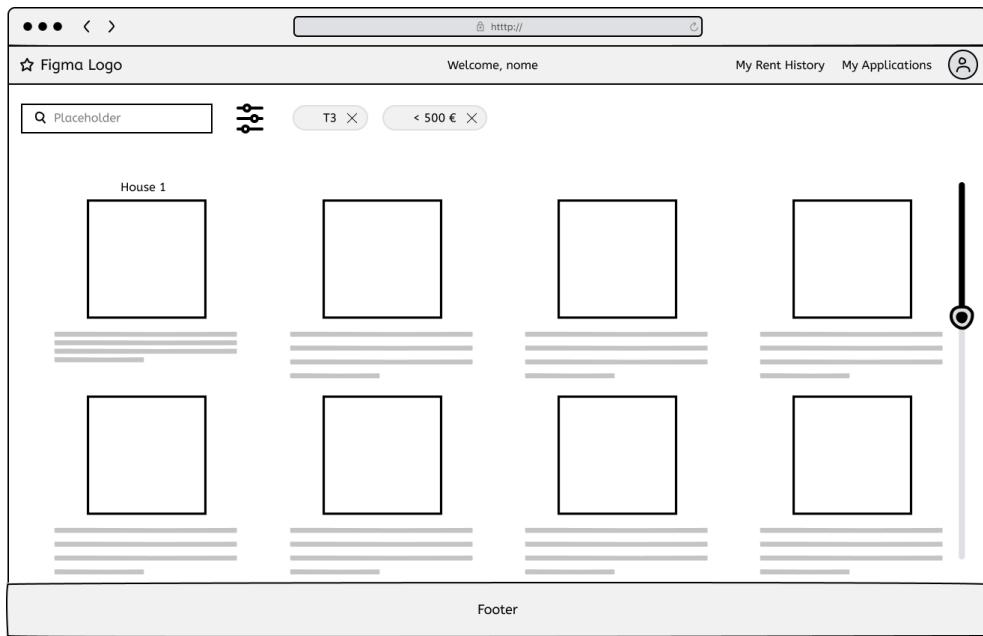


Figura 18: Proposta de interface para a página principal do inquilino



Figura 19: Proposta de interface para a página principal do senhorio

This wireframe shows the user interface for a tenant's profile page. At the top, there is a header with the Figma logo, the user's name 'Welcome, nome', and navigation links for 'My Rent History' and 'My Applications'. Below the header is a large placeholder for a user profile picture. To the right of the picture are several input fields for personal information: 'Full name' (text input), 'NIF' (text input), 'Email' (text input), 'Phone number' (text input), 'Nationality' (text input), 'Username' (text input), 'Password' (password input), 'Birth Date' (text input), and 'Occupation' (text input). There are also two buttons: 'Edit Password' and 'Edit'. Below these fields is a section titled 'My Ratings' containing four circular icons labeled 'Feature1', 'Feature2', 'Feature3', and 'Feature4', each accompanied by a five-star rating scale. To the right of this section is a sidebar titled 'My Rent History' which lists three entries with date ranges and 'View More' buttons. At the bottom of the page is a footer.

Figura 20: Proposta de interface para o perfil do próprio inquilino

This wireframe shows the user interface for a landlord's profile page. At the top, there is a header with the Figma logo, the user's name 'Welcome, nome', and navigation links for 'Search', 'Check Applications', 'Add New House', and 'My Houses'. Below the header is a large placeholder for a user profile picture. To the right of the picture are several input fields for personal information: 'Full name' (text input), 'NIF' (text input), 'Email' (text input), 'Phone number' (text input), 'Username' (text input), 'Password' (password input), 'Address' (text input), 'Birth Date' (text input), and 'Gender' (radio buttons for Female, Male, and Other). There are also two buttons: 'Edit Password' and 'Edit'. To the right of these fields is a sidebar titled 'Houses' which lists three entries with house details like 'Name', 'Location', and 'Available Slots', each with a 'View More' button. At the bottom of the page is a footer.

Figura 21: Proposta de interface para o perfil do próprio senhorio

This wireframe shows a user profile page for a tenant. At the top, there's a placeholder for the Figma logo. The navigation bar includes links for 'Welcome, nome', 'Search', 'Check Applications', 'Add New House', 'My Houses', and a user icon. On the left, there's a large circular placeholder for a profile picture. Below it, a 'Gender' section offers options: 'Female' (selected), 'Male', and 'Other'. To the right, there's a form for personal information: 'Full name' (text input 'full name'), 'Email' (text input 'email@example.com'), 'Username' (text input 'username'), 'Phone number' (text input '+351- XXXXXXXX'), 'Birth Date' (text input 'DD/MM/YYYY'), 'Nationality' (text input 'Portuguese'), and 'Occupation' (text input 'Student'). Below this, a 'Ratings' section displays four circular icons labeled 'Feature1', 'Feature2', 'Feature3', and 'Feature4', each accompanied by a five-star rating scale.

Figura 22: Proposta de interface para o perfil de outros inquilinos

This wireframe shows a user profile page for a landlord. The layout is similar to Figure 22, with the Figma logo at the top, a navigation bar with 'Welcome, nome', 'My Rent History', 'My Applications', and a user icon, and a large circular profile picture placeholder. The 'Gender' section is identical. The personal information form includes 'Full name' (text input 'full name'), 'Phone number' (text input '+351- XXXXXXXX'), 'Birth Date' (text input 'DD/MM/YYYY'), 'Username' (text input 'username'), 'Address' (text input 'Rua, n°'), and 'Email' (text input 'email@example.com'). To the right, there's a 'Houses' section showing three house cards. Each card displays a placeholder image, 'Name', 'Location', 'Available Slots', and a 'View More' button. A 'More' button is located at the bottom of this section. A footer bar is at the bottom of the page.

Figura 23: Proposta de interface para o perfil de outros senhorios

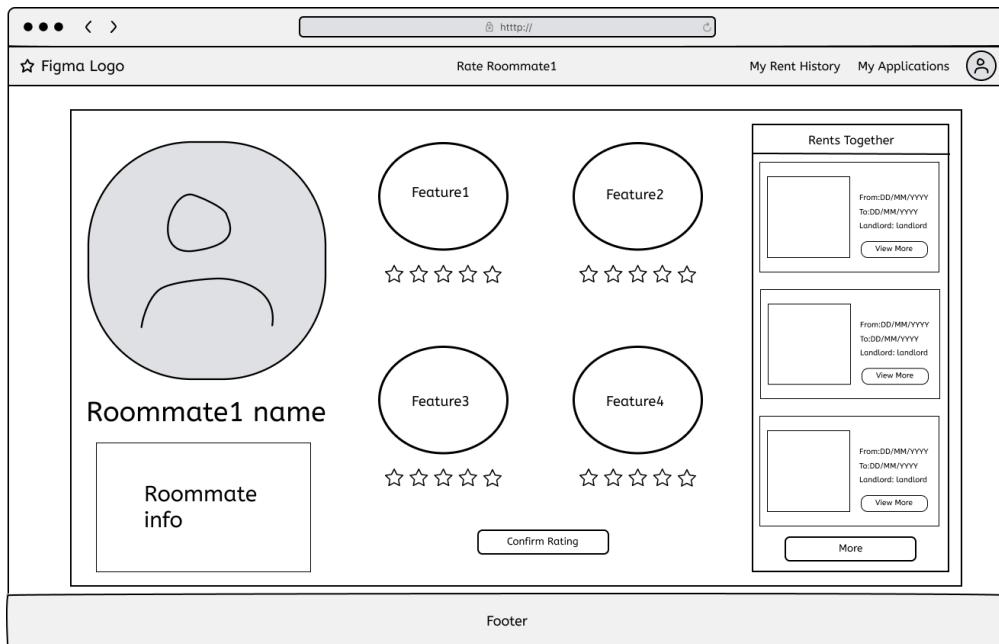


Figura 24: Proposta de interface para um inquilino avaliar inquilinos

(a) A interface do senhorio para avaliar inquilinos será semelhante mas com diferentes aspectos para serem avaliados

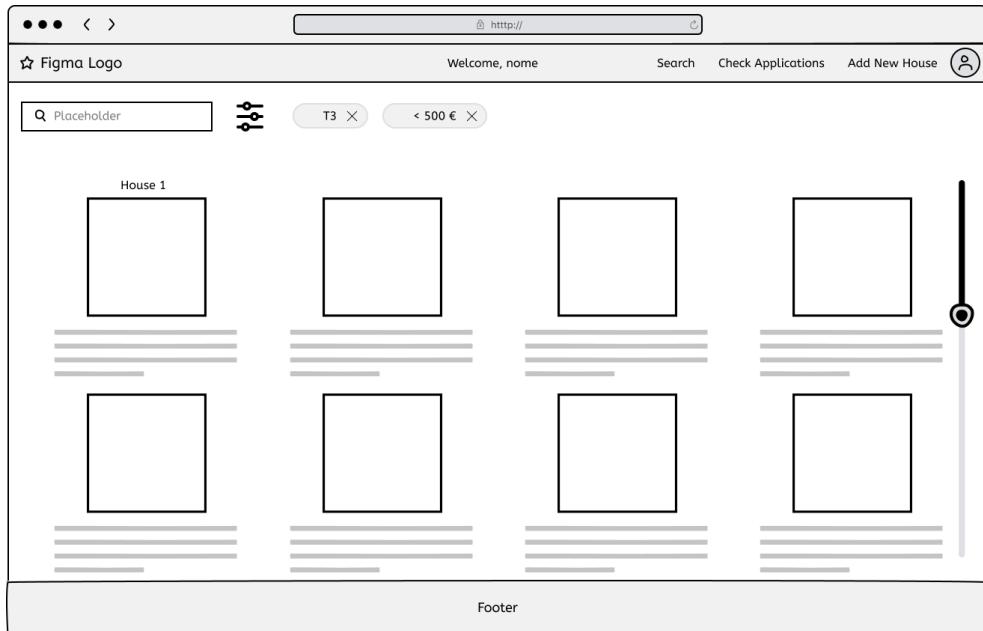


Figura 26: Proposta de interface para procurar imóveis por parte de um senhorio

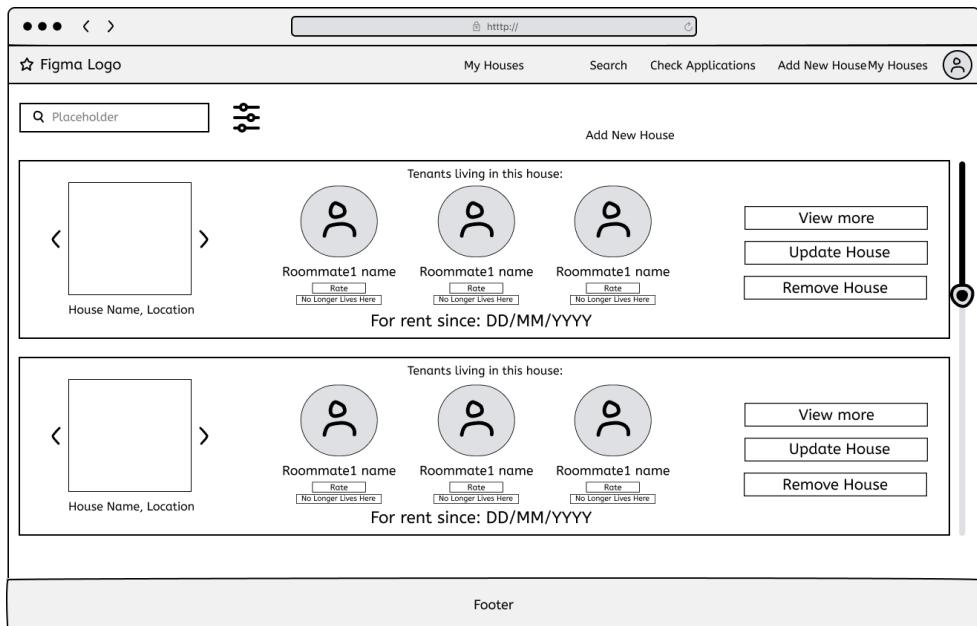


Figura 27: Proposta de interface para um senhorio poder ver as suas casas

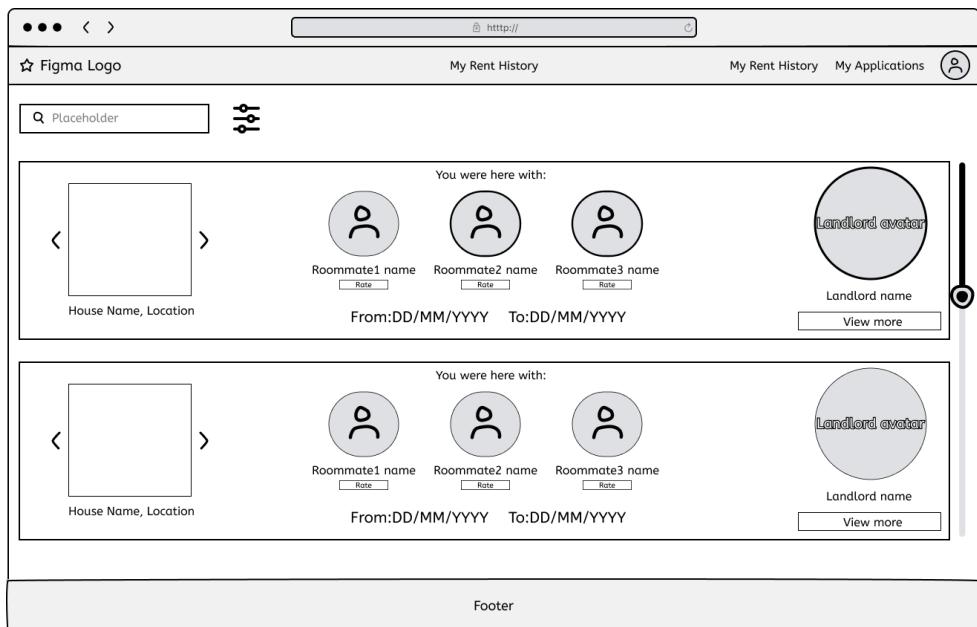


Figura 28: Proposta de interface para um inquilino poder ver o seu historial de arrendamentos

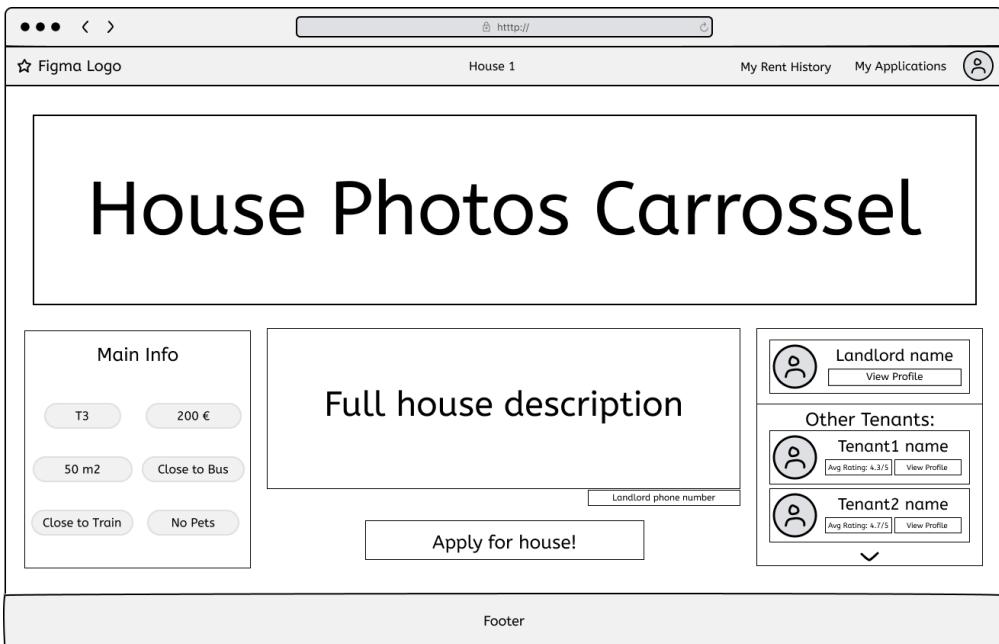


Figura 29: Proposta de interface para um inquilino poder ver um determinado imóvel

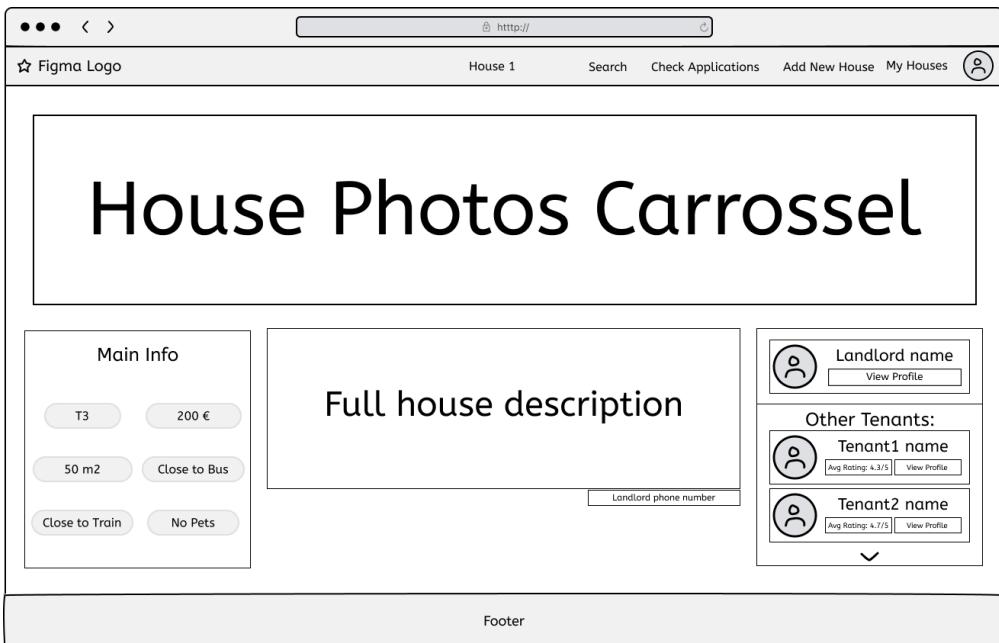


Figura 30: Proposta de interface para um senhorio poder ver um determinado imóvel

Enter info for the new house

House title House title	Description	Photos
Address Street, nº		
Number of Bedrooms 3		
Available Bedrooms 180€	Features <span>No Pets X</span> <span>Close to Bus X</span> <span>+</span>	
Number of Bathrooms 1		
Price per Bedroom 180€		

Confirm House

Footer

Figura 31: Proposta de interface para um senhorio adicionar um imóvel

Change house info

House title House title	Description	Photos
Address Street, nº		
Number of Bedrooms 3		
Available Bedrooms 180€	Features <span>No Pets X</span> <span>Close to Bus X</span> <span>+</span>	
Number of Bathrooms 1		
Price per Bedroom 180€		

Edit House

Footer

Figura 32: Proposta de interface para um senhorio editar um imóvel



Figura 33: Proposta de interface para um senhorio confirmar a adição de uma casa

(a) Para confirmar a edição da casa e mais algumas confirmações que possam vir a haver no sistema serão muito semelhantes a esta

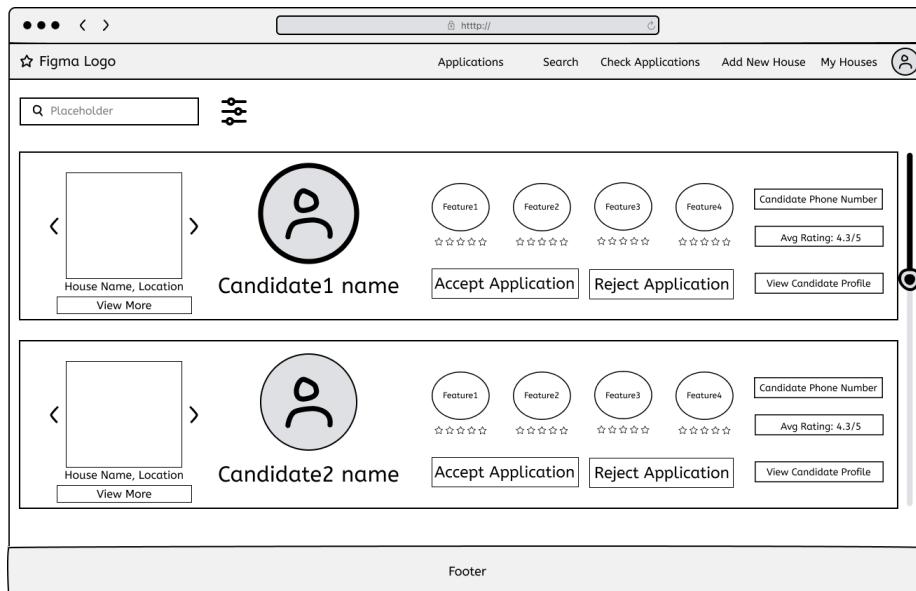


Figura 35: Proposta de interface para um senhorio verificar as candidaturas aos seus imóveis

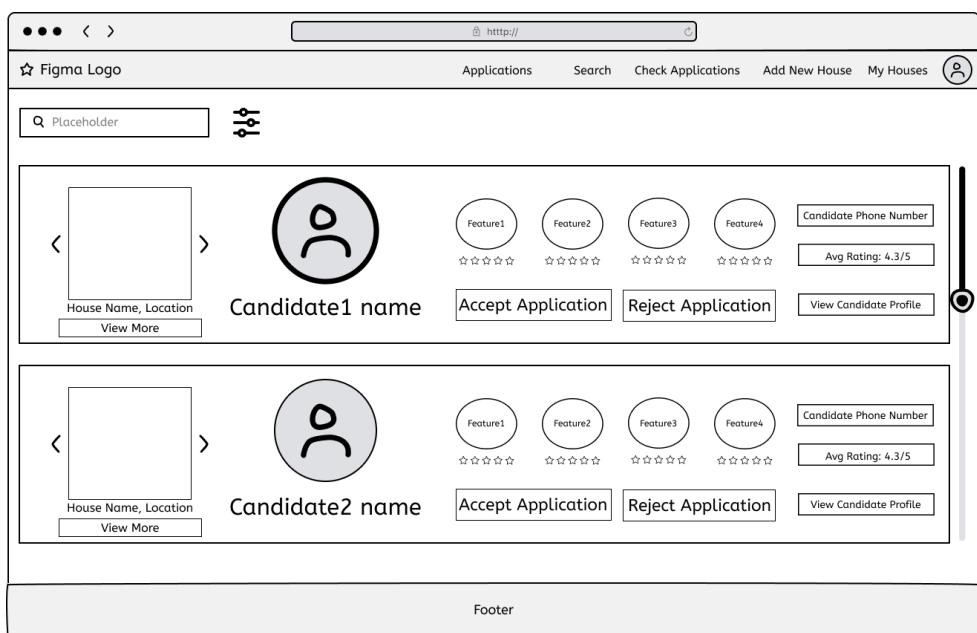


Figura 36: Proposta de interface para um inquilino verificar as suas candidaturas

## 3 Implementação

Após terminar a etapa de Modelação, o nosso foco voltou-se para o desenvolvimento do sistema, começando pela escolha das tecnologias a usar e passando pela implementação do sistema usando as mesmas.

### 3.1 Frontend

Para a construção do *Frontend* da nossa aplicação, optámos pela utilização da *framework* **Vue.js**. Esta escolha deve-se ao facto desta *framework* ter uma curva de aprendizagem pouco acentuada e ao facto de existir bastante documentação de suporte, bem como a existência de componentes que facilitam certos aspetos da implementação (*vue-router*, *vue-formulate*, etc).

#### 3.1.1 Componentes

Uma das filosofias do **Vue.js** é a utilização de componentes **reutilizáveis** de modo a minimizar a escrita de código repetido e facilitar a manutenção do mesmo. No nosso caso, implementámos diversas componentes, tais como: **Navbars**, **Footers**, **Cards**, **Carroséis**, e outros elementos presentes em diversas páginas.

#### 3.1.2 Views

As **Views** correspondem às diversas páginas a que o utilizador tem acesso, sendo que são também componentes em si. A única diferença entre uma **View** e uma **Componente** prende-se pela questão de uma *View* não poder ser reutilizada noutra *View*, algo que uma Componente consegue. Algumas das principais *Views* por nós implementadas foram: a **Página Inicial**, **perfis dos utilizadores**, páginas de **pesquisa de casas**, páginas com os **detalhes de uma casa**, formulários de adição de casas e utilizadores, entre outras. No caso dos formulários, recorremos a um módulo bastante útil chamado de **vue-formulate**, que nos permitiu obter os dados dos formulários com facilidade e validar os mesmos do lado do cliente.

### 3.1.3 Routes

Uma parte essencial na nossa aplicação é a capacidade de navegar entre páginas. Para o efeito utilizámos um módulo disponível para **Vue.js** chamado de **vue-router**.

No **Router** da nossa aplicação definimos todas as rotas disponibilizadas ao Utilizador, sendo que tivemos o cuidado de definir mecanismos de proteção para as rotas e para o sistema como um todo. A primeira dessas proteções foi o redirecionamento para uma **página de erro 404** quando o Utilizador tenta aceder a uma rota inexistente. As restantes proteções que adicionámos consistem em verificações feitas em determinadas rotas antes de permitir que um Utilizador aceda à página. AS verificações feitas consistem em saber se o Utilizador está autenticado, se está autenticado e é do tipo *Landlord* ou se está autenticado e é do tipo *Tenant*. Deste modo conseguimos evitar que Utilizadores não autenticados acedam a páginas que não pretendemos, e evitámos que Utilizadores do tipo *Landlord* acedam a páginas exclusivas a Utilizadores do tipo *Tenant*, e evitámos que Utilizadores do tipo *Tenant* acedam a páginas exclusivas a Utilizadores do tipo *Landlord*.

### 3.1.4 API

O *frontend* por si só não apresenta toda a funcionalidade pretendida, sendo que é necessário integrar o *backend*. Para esse efeito, é necessário fazer invocações à nossa **API REST**. Para efetuarmos as invocações à nossa API, e dado que é muito simples a utilização de **JavaScript** puro com **Vue.js**, optámos pela utilização do **axios** para esse efeito. Outro fator que pesou nesta decisão foi a nossa experiência na utilização do mesmo. Dado que alguns dos pedidos feitos à API requerem autenticação, aquando o início de sessão, guardámos o *token* devolvido nos *headers* do *axios*, facilitando a utilização do token. Aquando ao término de sessão, removemos os valores guardados nos *headers*. Deste modo, somos capazes de integrar a nossa API no *frontend* de uma maneira simples e intuitiva.

## 3.2 Backend

Para o backend, construímos uma **Rest API** que irá satisfazer os pedidos por parte do *frontend*. Esta arquitetura foi escolhida para facilitar a independência dos componentes sem perder a facilidade de acesso à lógica de negócio. Sendo assim, utilizámos a *framework* **Spring Boot** para a construção de **Rest Controllers** de forma a fazer o mapeamento do pedido e assim obter a resposta adequada. A razão por detrás da escolha desta camada de lógica de negócio deve-se, essencialmente, à sua extensibilidade e rápido processo de desenvolvimento.

### 3.2.1 *Controllers*

Em Spring, os *HTTP Requests* são manipulados por **Rest Controllers**. A divisão que achámos mais adequada foi a separação por entidades do sistema. Assim foram criados os seguintes *controllers*:

- **AdminController** - *Controller* que manipula operações de **criação** e **remoção** da base de dados do sistema;
- **ApplicationController** - *Controller* responsável por responder às operações das candidaturas. Dentro destas temos a opção de **criar/apagar** uma candidatura por parte do inquilino, e **aceitar/rejeitar** uma candidatura por parte do senhorio.
- **AuthController** - *Controller* que satisfaz os pedidos de autenticação do sistema (**login/logout**).
- **EvaluationController** - As operações relacionadas com as avaliações dos inquilinos são mapeadas neste controlador.
- **HouseController** - *Controller* que trata de pedidos CRUD relacionados com um imóvel.
- **LandlordController** - *Controller* que responde às operações relacionadas com senhorios, sendo estas a **criação/remoção** dos mesmos, **atualização** de dados pessoais, apresentação do conjunto de imóveis registados no sistema e candidaturas feitas para estes, realizadas pelos inquilinos.

- `RentHistoryController` - *Controller* cuja função é concluir um aluguer, adicionando ao histórico de aluguerares da casa o respetivo inquilino.
- `TenantController` - *Controller* que responde às operações relacionadas com inquilinos, sendo estas a **criação/remoção** dos mesmos, **atualização** de dados pessoais, apresentação do conjunto de candidaturas efetuadas juntamente com o seu histórico de aluguerares.

### 3.2.2 Services

Para além dos *controllers*, optámos por adicionar uma nova camada à estrutura do *backend* com objetivo de auxiliar na gestão dos pedidos recebidos nos controladores. Este auxílio é feito **encapsulando a lógica de negócio** dos pedidos, simplificando assim os métodos dos controladores, sendo que nestes apenas teremos a preocupação de construir o objeto *response* do pedido efetuado. Chamadas de métodos de **acesso à base de dados** também serão feitos no contexto dos serviços.

### 3.2.3 Segurança

De maneira a prevenir o uso indevido da API, todos os mapeamentos (com exceção do *login* e registo) são filtrados com o objetivo de perceber se o pedido HTTP possui um *token* e se este é certificado. Depois de validar o *token*, o pedido HTTP pode assim ser “enviado” para o respetivo *controller*. Para a autenticação dos utilizadores, recorremos ao módulo Spring Security que nos permitiu aproveitar de mecanismos como JWT Tokens. Para além de autenticação foram também aplicados mecanismos de validação de autorização de forma a garantir que cada utilizador apenas tem acesso ao que lhe é suposto.

### 3.2.4 Documentação

Para a documentação da API, optámos por tirar partido da biblioteca Java chamada `springdoc-openapi`, que automatiza todo este processo para projetos de Spring Boot. Gera a documentação examinando a aplicação em *build time* inferindo a semântica da API baseando-se nas configurações do Spring, na estrutura de

classes e nas várias anotações que podem existir. A biblioteca gera documentação automaticamente em páginas formatadas em JSON / YAML e HTML. Exemplos das mesmas estarão presentes abaixo.

The screenshot shows the Swagger UI interface for the "Roomie's Rest API". At the top, it displays the API title "Roomie's Rest API" with version "1.1.0" and "OAS3". Below the title, it shows the license "MIT". The main content area is titled "tenant-controller". It lists several endpoints:

- DELETE** /tenants/{id}
- GET** /tenants/{id}
- GET** /tenants/{id}/rentHistory
- GET** /tenants/{id}/rating
- GET** /tenants/{id}/avatar
- GET** /tenants/{id}/applications

Figura 37: Página Inicial Documentação

The screenshot shows the Swagger UI interface for the "POST /auth/login" endpoint. It includes the following sections:

- Parameters**: No parameters listed.
- Request body**: Required, application/json schema. Example value:

```
{
  "email": "string",
  "password": "string"
}
```
- Responses**:
  - Code**: 200 **Description**: OK. **Links**: No links.
  - Media type: application/json (selected). Controls Accept header.
  - Example Value | Schema:

```
{
  "token": "string"
}
```

Figura 38: Documentação para Login

### 3.3 Base de Dados

Nesta subdivisão será abordada a camada de dados e como é feito o acesso aos mesmos. Para facilitar a gestão desta camada utilizamos a ORM **Hibernate**. Esta escolha deve-se ao facto de não só ter sido uma ORM estudada no contexto da unidade curricular Arquiteturas Aplicacionais, como também por ser uma ferramenta conhecida pelo seu alto desempenho e confiabilidade. O facto de ser bastante completa proporciona uma redução de trabalho na administração do acesso aos dados. Uma vez que existem ferramentas, como *Visual Paradigm*, que ajudam na tarefa de modelar e implementar os modelos, optámos por esta abordagem. O diagrama de classes foi anotado de maneira a indicar à ferramenta as entidades e ligações a persistir. Com o diagrama anotado, basta selecionar o sistema desejado, algumas configurações e a ferramenta é capaz de gerar os ficheiros. No fim deste processo, todo o acesso ao servidor **PostgreSQL** é feito através de DAO's e gerido pelo Hibernate. PostgreSQL foi escolhido para servidor de base de dados do projeto por ser *Open Source*, apresentar características de elevada escalabilidade e disponibilidade e forte adoção pelo tecido empresarial na atualidade.

A figura 39 mostra o diagrama EER gerado através no Visual Paradigm.

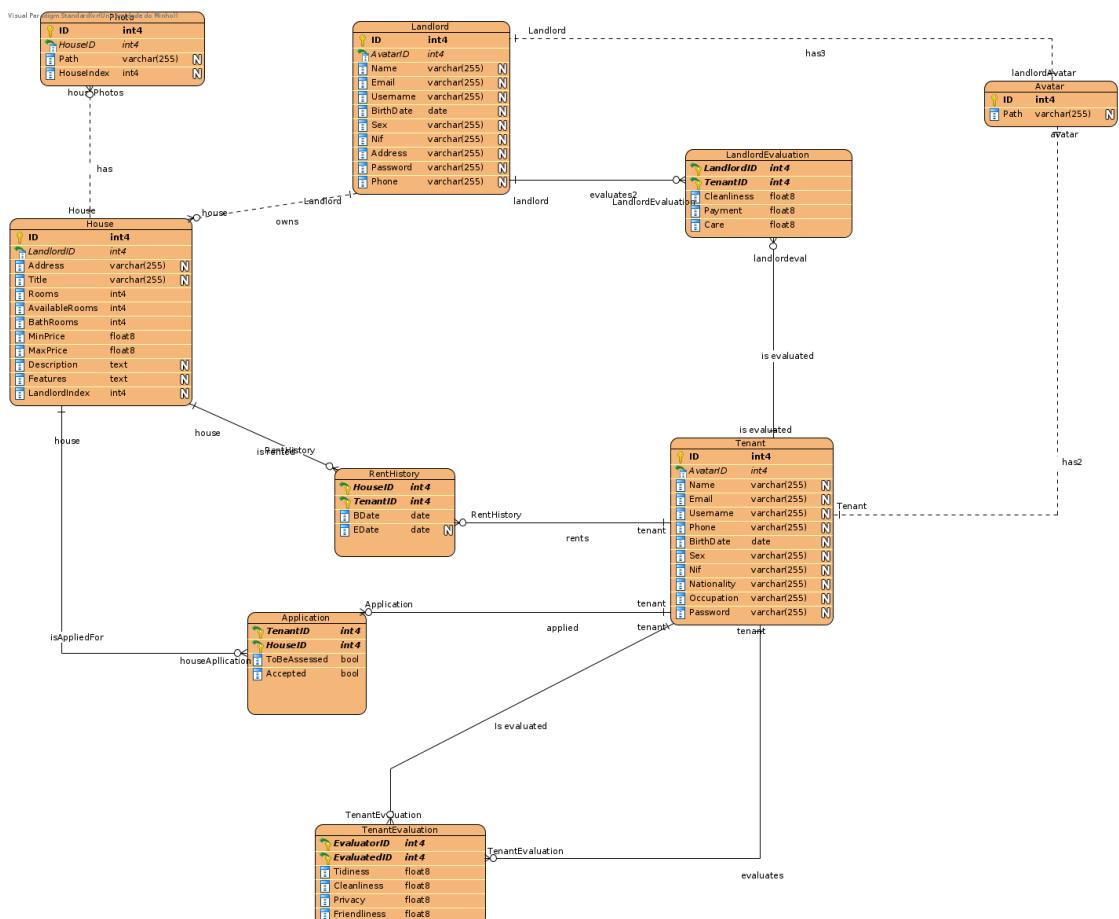


Figura 39: Diagrama EER

## 4 Deployment

Após terminada a fase de desenvolvimento, iniciou-se a fase de *deployment*, de forma a permitir que a aplicação esteja disponível para os utilizadores. De forma a alcançar uma instalação simples, adaptável, versátil e facilmente escalável, decidiu-se utilizar ferramentas de aprovisionamento de infraestruturas e gestão de configurações (**Ansible**, em conjunto com a *Google Cloud Platform*) e orquestração de *containers* (**Docker** e **Docker Swarm**).

O *Ansible* é uma ferramenta que permite efetuar o aprovisionamento remoto de máquinas de um modo escalável, previsível, e totalmente automático. É, portanto, uma ferramenta extremamente útil para efetuar a instalação dos componentes necessários para correr esta aplicação, e portanto, foi esta ferramenta que foi utilizada para instalar algumas das ferramentas necessárias (como a *Docker Engine*), e também para fazer a criação e manutenção das máquinas virtuais e discos na *Google Cloud Platform* (GCP).

A escolha de utilizar **Docker** visa simplificar e agilizar os processos de automatização da instalação de toda a infraestrutura, visto que a conteinerização permite ter ambientes isolados e leves para cada uma das camadas da aplicação.

Contudo, a simples utilização de *Docker* não cumpre por completos todos os requisitos de orquestração de serviços. É nesta parte que entra o *Docker Swarm*. O objetivo desta ferramenta é a orquestração de serviços, e através da mesma conseguimos, com recurso a ficheiros *YAML*, construir toda a nossa infraestrutura de serviços, desde a criação dos mesmos, a especificação do número de instâncias que cada serviço deve ter, *restart policy* dos *containers*, entre outras coisas.

Para além disso, de forma a ter um *Docker Environment* mais eficiente, após concluir as imagens de cada um dos componentes foram construídas (fazendo o seu *build*) e publicadas no registo público do *Docker*, [Docker Hub](#), (através do comando *docker push*), para que, quando fossem necessárias, ser apenas preciso fazer o respetivo *pull*.

Desta forma, a figura 40 representa arquitetura planeada para aplicar no *deployment* da aplicação **Roomie**, que é constituída por três grandes camadas,

todas elas assentes em um ou mais containers Docker:

- Web Servers (3 instâncias) com o objetivo de fazer balanceamento de carga para os App Servers e servir conteúdo estático (imagens, HTML, CSS, JS, etc).
- App Servers (3 instâncias) com a responsabilidade de lidar com toda a parte lógica da aplicação. De ressalvar que estes *containers* partilham um volume externo NFS onde são guardadas as imagens dos utilizadores e casas.
- Base de Dados (PostgreSQL - 1 instância) com a responsabilidade de guardar toda a informação necessária da aplicação.

A camada mais superficial é a única que comunica com o exterior, e é acedida de forma única, visto que o Docker Swarm as exporta com um só e faz o respetivo balanceamento entre as várias instâncias dos web servers.

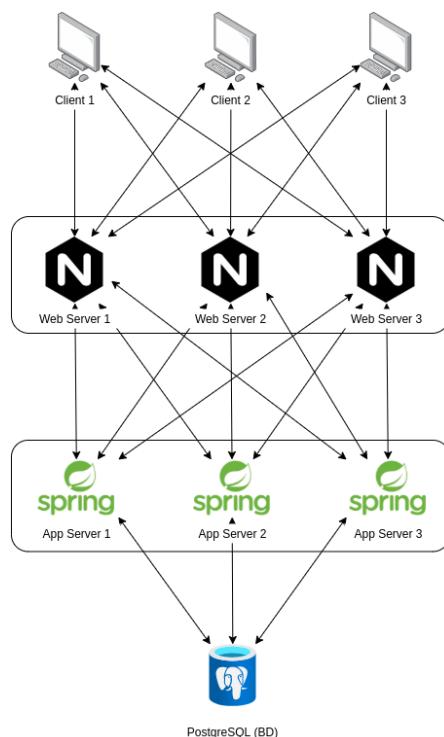


Figura 40: Arquitetura de *Deployment*

No anexo A, disponibiliza-se a especificação YAML do Docker Swarm/Stack.

## 5 Análise de Carga

### 5.1 Considerações Iniciais

Testar e analisar a capacidade do sistema disponibilizado é uma tarefa crucial para antecipar e corrigir possíveis problemas e falhas, conhecer a capacidade efetiva do mesmo e identificar *bottlenecks* antes de tornar o sistema disponível a centenas, milhares, dezenas de milhares ou milhões de utilizadores.

Assim, e considerando a natureza da aplicação desenvolvida, decidiu-se implementar testes de carga para os dois atores do sistema, simulando uma utilização do mesmo para cada um dos atores, focando o cerne dos testes em operações possivelmente mais custosas para o sistema, quer pela sua particular complexidade de execução ou pela elevada recorrência que serão executadas.

De modo a evitar todo e qualquer tipo de distorção de performance, sobrecarga adicional e não necessária, e garantir resultados o mais representativos e fidedignos possível, os testes forma realizados a partir de uma máquina na pertence ao *cluster* de *deployment* e essa máquina manteve-se constante ao longo da execução de toda a suite dos testes.

### 5.2 Cenários de Teste

Para a realização dos testes utilizou-se uma ferramenta de *benchmarking* de serviços web em Python: **Locust**. A principal razão desta escolha deveu-se ao facto da ferramenta permitir implementar os cenários de teste em código e não através de uma interface como é o caso de outras ferramentas (por exemplo, JMeter).

De referir que todos os testes foram executados após fazer um pré-populamento do sistema com:

- 50 Landlords;
- 50 Tenants;
- 500 Casas.

Deste modo, e através da utilização deste ferramenta, foram implementados dois cenários de teste, um para cada ator do sistema:

- **Landlord**: registar, autenticar, adicionar entre 1 a 6 casas, listar casas existentes no sistema, listar candidaturas às suas casas, ver o perfil de candidatos a casas.
- **Tenant**: registar, autenticar, pesquisar por casas, candidatar-se a casas (entre 1 a 6), listar as suas candidaturas a casas, cancelar uma das suas candidaturas.

Por fim, referir que os testes foram executados para um número crescente de utilizadores simulados (100, 250, 500, 750 e 1000) de forma a melhor observar qual a capacidade de resposta do sistema, bem como os seus limites.

### 5.2.1 Landlord

Para o ator Landlord, como já referido, foi escolhido o seguinte cenário de teste:

- Pedido POST de registo;
- Pedido POST de autenticação;
- Pedido(s) POST para registrar casa(s);
- Pedido GET para listar as suas casas;
- Pedido GET para listar as candidaturas às suas casas;
- Pedido GET para ver o perfil de Tenant(s).

Na tabela 9 encontra-se um resumo dos resultados do respetivo teste.

#Utilizadores	<i>Throughput</i> (req/s)	Tempo Resposta (mediana)(ms)	Erro (%)
100	48.6	74.49	0.0
250	79.16	88.24	0.0
500	83.77	99.65	0.0
750	81.51	113.31	0.0
1000	77.99	140.65	0.7

Tabela 9: Resumo dos resultados do cenário de teste para o Landlord

Analizando os dados da tabela 9, conforme o número de **Utilizadores** aumenta, o **Throughput** da aplicação aumenta também, até alcançarmos o patamar dos 750 Utilizadores, sendo o seu valor máximo alcançado com 500 Utilizadores. Relativamente aos **Tempos de Resposta**, estes mantêm-se dentro de padrões aceitáveis aquando do aumento de Utilizadores. A **Taxa de Erro** nos pedidos efetuados, mantém-se nula até chegarmos aos 1000 Utilizadores, apesar da mesma ser extremamente baixa.

Para o cenário de testes apresentado, o sistema possui uma capacidade de resposta de aproximadamente 500 a 600 Utilizadores.



Figura 41: Evolução do Número de Utilizadores



Figura 42: Total de Pedidos por Segundo

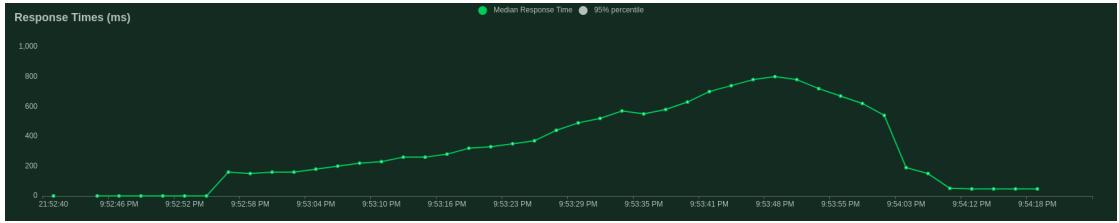


Figura 43: Evolução dos Tempos de Resposta

Analizando a evolução dos tempos de resposta (figura 41) e comparando com a evolução do número de utilizadores (figura 43), é visível que quando o número de utilizadores estabiliza, os tempos de resposta começam a diminuir.

### 5.2.2 Tenant

Para o ator Tenant, como já referido, foi escolhido o seguinte cenário de teste:

- Pedido POST de registo;
- Pedido POST de autenticação;
- Pedido GET para pesquisar/listar casas;
- Pedido(s) POST para se candidatar a casa(s);
- Pedido GET para listar as suas candidaturas;
- Pedido DELETE para cancelar uma das suas candidaturas.

Na tabela 10 encontra-se um resumo dos resultados do respetivo teste.

# Utilizadores	<i>Throughput</i> (req/s)	Tempo Resposta (mediana) (ms)	Erro (%)
100	41.44	140.44	0.0
250	72.2	130.77	0.0
500	93.6	150.46	0.0
750	87.8	197.03	0.0
1000	79.3	300.21	0.0

Tabela 10: Resumo dos resultados do cenário de teste para o Tenant

Analizando os dados da tabela 10, conforme o número de **Utilizadores** aumenta, o **Throughput** da aplicação aumenta também, até alcançarmos o patamar dos 750 Utilizadores, sendo o seu valor ótimo alcançado com 500 Utilizadores. Relativamente aos **Tempos de Resposta**, estes mantêm-se dentro de padrões aceitáveis aquando do aumento de Utilizadores. A **Taxa de Erro** nos pedidos efetuados, mantém-se sempre nula.

Para o cenário de testes apresentado, o sistema possui uma capacidade de resposta de aproximadamente 500 a 600 Utilizadores.

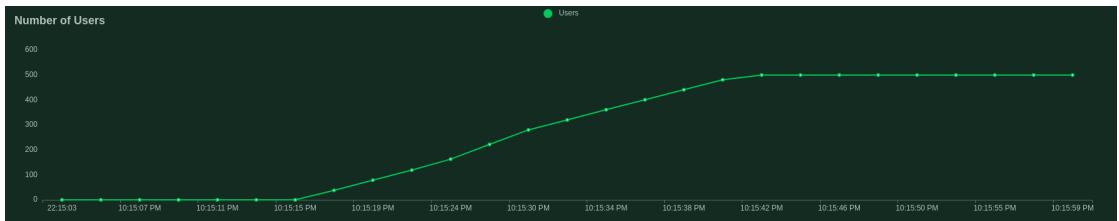


Figura 44: Evolução do Número de Utilizadores

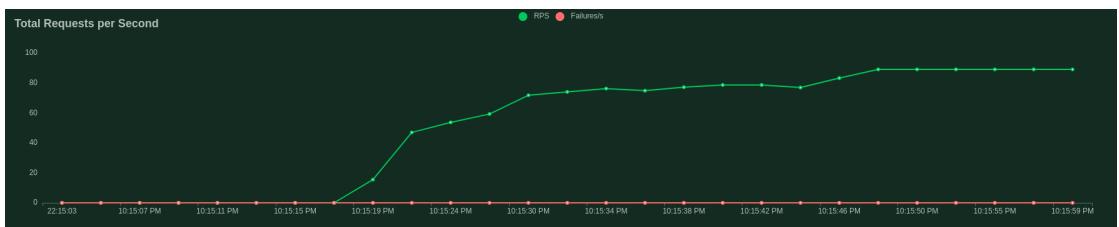


Figura 45: Total de Pedidos por Segundo

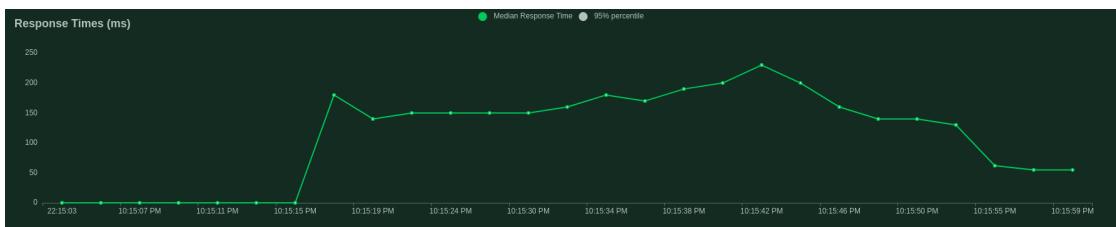


Figura 46: Evolução dos Tempos de Resposta

Analisisando a evolução dos tempos de resposta (figura 44) e comparando com o a evolução do número de utilizadores (figura 46) e do total de pedidos por segundo (figura 45), é visível que quando o número de utilizadores estabiliza, os tempos de resposta começam a diminuir e o total de pedidos por segunda também estabiliza.

Refletindo sobre os resultados obtidos, e face aos objetivos por nós traçados, podemos afirmar que nos encontramos satisfeitos com a performance do sistema, sendo que o mesmo suporta confortavelmente entre 500 a 600 utilizadores em simultâneo, nas diversas operações.

## 6 Análise de Usabilidade

Após terminado, e mesmo durante, o desenvolvimento da plataforma, torna-se imperativo documentar as interfaces, bem como efetuar uma avaliação da qualidade das mesmas. Com esse intuito, iremos neste capítulo apresentar o resultado das interfaces, bem como, recorrer aos **Princípios de Usabilidade** e às **Heurísticas de Nielsen** para melhor entender a qualidade, e acima de tudo, **usabilidade da plataforma desenvolvida**.

### 6.1 Apresentação da Interface

Por forma a não tornar o relatório demasiado extenso, optámos por apresentar apenas algumas das interfaces mais relevantes do projeto.

Print das navbars

Antes de falar das interfaces em si é importante analisar o componente que está presente em todas elas, sendo esse componente a barra de navegação que nos permite andar pelas páginas de uma forma muito intuitiva e rápida assim como dar *logout* ou visitar o perfil clicando na foto do utilizador que lá aparece.

Print da página de search do tenant

Na página de pesquisa de casas, que é semelhante tanto nos inquilinos como nos senhorios, podemos verificar que é mostrado ao utilizador um conjunto de casas organizado por páginas que permite uma melhor navegação pelos imóveis existentes, para além dessa paginação temos ainda uma barra de pesquisa que nos permite aplicar vários tipos de filtragem como o nome da casa, a localização, preços ou número de quartos, sendo que também achamos importante ter um botão que nos permitisse nessa barra de pesquisa apagar os filtros já existentes para facilitar o seu uso.

Print da vista de uma casa por parte do tenant

Já na visualização da página de uma casa por parte de um inquilino, semelhante à do senhorio mas sem o botão para se candidatar à casa, optamos por usar um carrossel de imagens para que fosse intuitivo ver as fotografias disponíveis as-

sim como destacar as informações mais importantes para quem esta a ver. Temos também uma área que nos mostra que é o senhorio daquele imóvel assim como os potenciais inquilinos que lá morem para que os inquilinos possam saber de antemão os seus possíveis futuros colegas de casa.

#### Print das applications do landlord e do modal

Uma vez que o inquilino se candidate a uma casa o senhorio irá receber na sua página uma candidatura na qual é facilmente identificável o candidato através da sua fotografia de perfil assim como de um conjunto de classificações feitas tanto por outros senhorios como antigos colegas de casa do candidato o que facilita a decisão ao senhorio na hora de decidir aceitar ou não a candidatura. Quando tiver tomado a decisão o senhorio pode optar por uma das opções e irá ser-lhe apresentado uma pequena caixa de diálogo para confirmar a sua decisão e assim que isso confirmar dá-mos o processo de integração do inquilino na casa como concluído.

#### Print das applications do tenant

Os inquilinos podem verificar se o senhorio já viu a sua candidatura através da página de aplicações acima que permite saber se a candidatura foi aceite, rejeitada ou ainda está em análise assim como um pequeno carrossel de fotos da casa para que esta seja mais facilmente identificada.

#### Print do renthistory dos tenants

É também possível aos inquilinos ver em casas já estiveram e com quem, assim como ser capaz de avaliar os inquilinos com quem partilharam cada uma dessas casas e ver de quando a quando lá estiveram para que essa informação seja mais fácil de obter.

#### Print do perfil do inquilino visto pelo próprio

Outra página que achamos importante é vista do perfil do inquilino visto pelo próprio dado que permite editar os seus dados, alterar a palavra-passe sendo que esta mostra uma nova área de interação para que se possam introduzir as mudanças requeridas pelo utilizador. Para além disso está presente um menu lateral que permite ver até às três casas mais recentes onde o inquilino esteve assim

como abrir a sua página de histórico de rendas para que possa ver a informação completa.

## 6.2 Princípios de Usabilidade

Antes de passarmos a uma visão mais formal das métricas de avaliação da usabilidade do nosso sistema, utilizando as Heurísticas de *Nielsen*, podemos realizar uma avaliação prévia da usabilidade do nosso sistema através dos Princípios de Usabilidade. Contrariamente a outros métodos de avaliação de uma interface, estes princípios fornecem uma visão mais genérica e menos autoritária de boas práticas de *design*, servindo assim como um bom ponto de partida.

Assim sendo, nesta fase do trabalho, vamos tentar expor algumas características de cada um dos princípios que acreditamos ter maior notoriedade na versão final da nossa plataforma web.

### 6.2.1 *Learnability*

Este princípio visa avaliar a facilidade com a qual novos utilizadores conseguem começar a interagir eficientemente com a plataforma idealizada, por forma a maximizar a performance das interações.

- **Predictability:** Por forma a promovermos a habilidade de predição dos resultados de determinada interação, optámos por adicionar modais capazes de guiar os utilizadores. Estes modais estão presentes em várias operações, desde a submissão das candidaturas a uma casa, onde irão ilustrar ao utilizador a consequência imediata de submeter ou cancelar a candidatura, como na operação de remover a própria submissão, entre outras. Desta forma, qualquer utilizador conseguirá, com segurança, prever o resultado da ação que pretende realizar. Para além disso, sempre que uma ação se encontra disponível, optámos por fornecer indicações claras, através da atualização dos botões disponibilizados. Por exemplo, caso um inquilino realize a submissão da sua candidatura, o botão irá ser atualizado para constatar esse facto e sinalizar que a candidatura já foi submetida.

- **Familiarity:** Um grande foco que optámos por manter durante a idealização da nossa plataforma foi o conceito de familiaridade, ou seja, a noção de que conhecimento de experiências, sistemas ou plataformas familiares ao utilizador deverá servir como base para a sua interação com a nossa aplicação *web*. Nesse sentido, temos várias ferramentas, que sendo baseadas nessas mesmas experiências, serão capazes de beneficiar dessa familiaridade. Começando pela criação de perfis, optámos por utilizar um *Callendar Picker* para a inserção das datas de nascimento que simula a utilização de um calendário de papel. Outro exemplo, são os mecanismos de pesquisa e paginação que implementámos na página de pesquisa de casas, que sendo baseados em implementações semelhantes muito comuns na maioria de plataformas *web* atualmente utilizadas, também beneficiam deste conhecimento pré-existente.

### 6.2.2 *Flexibility*

Este segundo princípio, *Flexibility*, lida com a variedade de maneiras como a informação será trocada entre utilizador e sistema.

- **Dialogue Initiative:** O nosso sistema está configurado de modo a fornecer o máximo de controlo possível ao utilizador, ou seja, todas as *features* disponibilizadas podem ser acedidas independentemente da experiência do mesmo, sendo apenas exclusivas ao tipo de conta. Nesse sentido, de acordo com esta característica, vimo-nos na necessidade de criar meios de comunicação entre o sistema e utilizador. Estes métodos variam, porém, acreditamos que os mecanismos de filtragem e pesquisa implementados na página de seleção de casas será o melhor exemplo deste tipo de característica.

### 6.2.3 *Robustness*

Finalmente, o princípio de *Robustness* lida com o nível de apoio ou suporte que o sistema providencia ao utilizador por forma a maximizar o número de ações realizadas com sucesso e manter comportamentos orientados ao objetivo dos mesmos.

- **Observability:** Quanto a esta característica, optámos por nos focar na

possibilidade do utilizador interagir com o máximo de funcionalidades em qualquer etapa da interação. Nesse sentido, optámos por disponibilizar *navigation bars*, acessíveis na maioria das páginas, como também fornecer o máximo de meios de interação nas próprias páginas (botões, *interactive pics*, etc.)

- **Responsiveness:** Qualquer operação disponibilizada pelo nosso sistema será, na vasta maioria dos casos, realizada com tempos de resposta ínfimos. Assim sendo, os nossos utilizadores irão manter um contacto responsivo e com a plataforma, o que providenciará um maior conforto sobre as interações realizadas.

## 6.3 Heurísticas de Nielsen

O principal objetivo das *Heurísticas de Nielsen* consiste na possibilidade de um conjunto de peritos analisarem a interface desenvolvida relativamente a 10 tópicos essenciais, com o objetivo de perceber o nível de usabilidade da mesma.

As Heurísticas a seguir apresentadas foram consideradas no desenvolvimento da interface. Dada a natureza iterativa do processo de *design* da interface do sistema, consideramos que com sucessivas atualizações, fomos capazes de criar uma *interface* com boa usabilidade. De seguida iremos apresentar as *Heurísticas* e fazer uma breve associação à nossa *interface*.

### 6.3.1 *Visibility of System Status*

Relativamente a esta Heurística, o nosso sistema apesar de não possuir operações demoradas, apresenta ao utilizador informação acerca do que acontece no sistema sob a forma de confirmações exibidas em *modals* (Figura 47) e sob a forma de alertas. Em operações como atualizações de perfis ou casas, submissões de candidaturas ou criação de contas, o utilizador sabe que a operação teve sucesso dado que é exibido um *modal* com o resultado da operação.

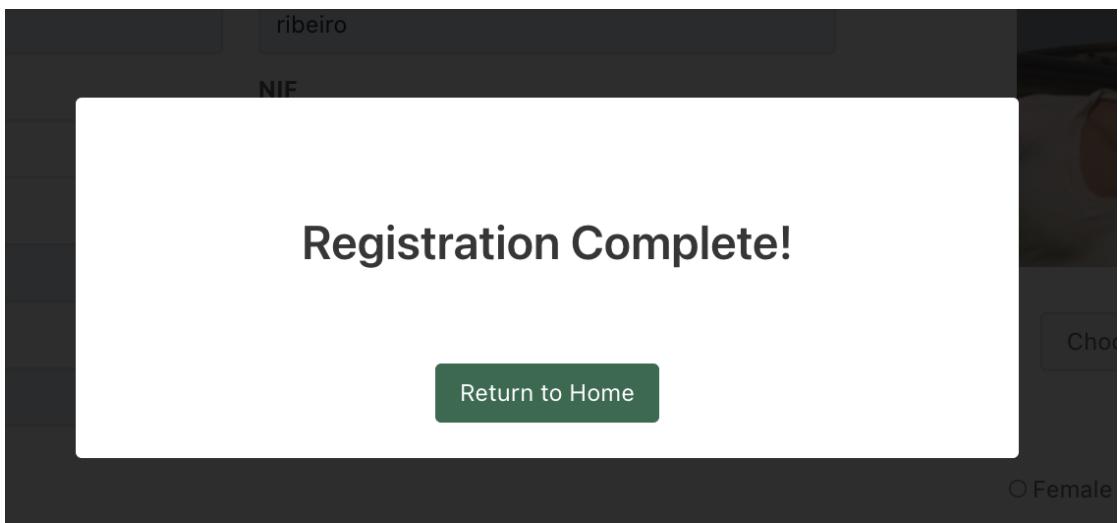


Figura 47: Exemplo de um *modal*

### 6.3.2 *Match between system and the real world*

Com esta Heurística, pretendemos que o utilizador possa utilizar conhecimentos previamente adquiridos ou experiências do seu dia-a-dia como referência na utilização do nosso sistema. Com esse objetivo, sempre que possível, utilizámos componentes que o utilizador já conhece, nomeadamente a utilização de um *Calendar Picker* para a seleção de datas (Figura 48). Este componente, por exemplo, simula bastante bem a utilização de um calendário em papel, permitindo que um utilizador inexperiente o utilize com relativa facilidade.

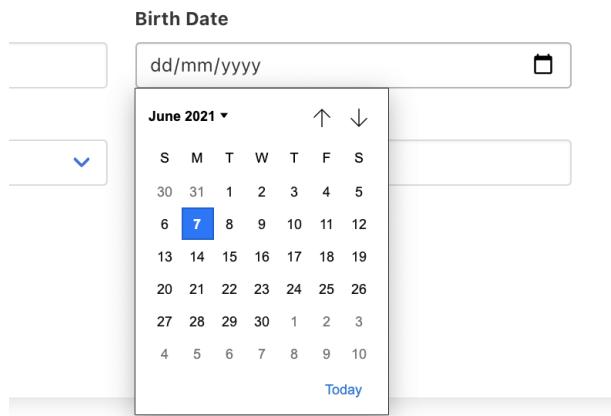


Figura 48: Exemplo de um *calendar picker*

### 6.3.3 User control and freedom

Com esta Heurística, pretende-se guiar o utilizador pela nossa *interface*, e informar o utilizador do ponto onde se encontra. Relativamente a este ponto, considerámos que a nossa interface se foca mais em guiar o utilizador para as páginas que pretende e não é tão clara relativamente a informar o utilizador acerca da página onde se encontra e como pode retroceder.

### 6.3.4 Consistency and standards

Relativamente a esta Heurística, pretendemos que o utilizador possa encontrar elementos que estão presentes noutros *sites* por exemplo, e que não tenha que adivinhar se elementos diferentes têm ações semelhantes. Neste aspeto, considerámos que tivemos sucesso dado que sempre que possível tentámos manter um nível de consistência na nossa *interface* e tentámos ao máximo seguir normas e padrões utilizados.

### 6.3.5 Error prevention

Tal como o nome indica, com esta Heurística pretendemos que a nossa *interface* consiga prevenir a ocorrência de erros. Para o efeito, adicionámos elementos como a validação de *inputs*, indicando qual o problema no *input* (Figura 49), ou a proteção de rotas no sistema (não permitir que senhorios accedam a rotas de inquilinos e inquilinos a rotas de senhorios), ou ainda, confirmar a intenção de realizar ações definitivas como remoções de *tentants* de casas ou rejeição de candidaturas.

NIF  
123456  
Invalid NIF!

Figura 49: Exemplo de *input validation*

### 6.3.6 Recognition rather than recall

Com esta Heurística, pretende-se que um utilizador não necessite de se lembrar de informação para utilizar o sistema, mas sim, possa reconhecer e ver a

informação através da *interface*. No caso do nosso sistema, não existe propriamente a necessidade de um utilizador necessitar de se lembrar de informação, dado que pode aceder a imóveis, históricos ou formulários de maneira simples e intuitiva.

### 6.3.7 Flexibility and efficiency of use

Relativamente a esta Heurística, pretende-se que seja possível que um utilizador experiente consiga acelerar a sua interação com o sistema através do uso de *shortcuts*. Nessa vertente, o nosso sistema não possui *shortcuts* dado que considerámos que a colocação de páginas mais comuns na *navbar* cumpre este objetivo, dada a simplicidade da plataforma e as restrições de tempo para o desenvolvimento.



Figura 50: Exemplo de uma *navbar*

### 6.3.8 Aesthetic and minimalist design

Nesta Heurística pretende-se avaliar a simplicidade e o minimalismo da *interface* desenvolvida. Nessa vertente, procurámos sempre manter um *design* simples e minimalista na nossa *interface* (Figura 51), tentando sempre evitar sobrecarregar o utilizador com informação desnecessária e tentando expor o conteúdo de uma forma simples e comprehensível.

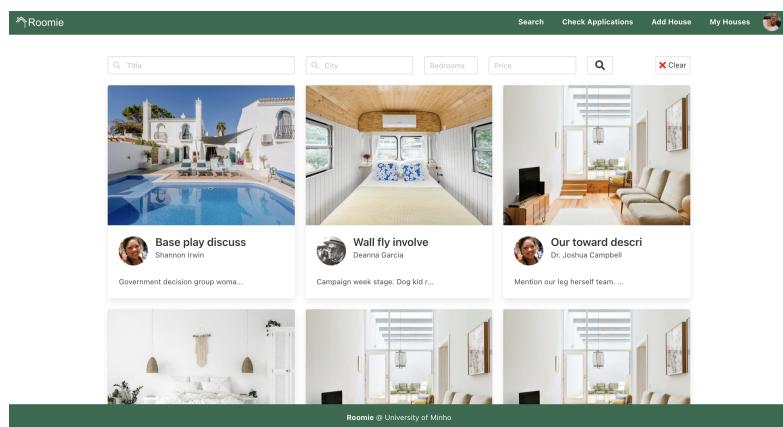


Figura 51: Página de pesquisa de imóveis

### **6.3.9 *Help users recognize and recover from errors***

Tal como o nome indica, esta Heurística pretende avaliar se um sistema informa o utilizador acerca de erros que possam ocorrer e se permite ao utilizador executar ações para recuperar do mesmo. Relativamente ao reconhecimento de erros, tentámos apresentar um *modal* (Figura 47) com uma mensagem simples, no entanto, a nossa *interface* não fornece métodos para o utilizador recuperar de um erro.

### **6.3.10 *Help and documentation***

Por fim, com esta Heurística pretende-se que uma *interface* seja simples o suficiente para ser utilizada sem documentação, mas, idealmente, seja disponibilizada. Dado que a nossa aplicação é relativamente simples de utilizar, e, dadas as restrições temporais, não nos foi possível criar uma página de documentação da plataforma para os utilizadores, sendo essa uma tarefa essencial para uma iteração futura.

## 7 Conclusões e Trabalho Futuro

No contexto da evolução das nossas capacidades como estudantes de engenharia informática, a realização deste projeto permitiu-nos consolidar a aprendizagem das UCs de Arquiteturas Aplicacionais e Sistemas Interativos, mais concretamente, as metodologias para planear e desenvolver interfaces úteis e fáceis de utilizar por todo e qualquer utilizador, bem como desenvolver sistemas capazes de satisfazer os requisitos de um projeto, enquanto mantendo uma arquitetura organizada, preparada para evolução futura e que é pensada para ser capaz de responder a elevada carga de utilização.

Para a execução deste projeto foi necessário utilizar tecnologias de desenvolvimento que a maioria do grupo não se encontrava familiarizada, pelo que também isso foi um desafio e necessitou de um trabalho de pesquisa e exploração considerável, permitindo aumentar o nosso conhecimento e alargar a nossa perspetiva relativamente a este tipo de tecnologias.

Analizando em retroespetiva, o grupo considera que foi possível cumprir os objetivos e requisitos a que se propôs, apesar de considerar que, como é normal, existem pontos que podiam ser melhorados e evoluídos. Alguns destes tópicos relacionam-se com a criação de funcionalidades que permitam os *tenants* e *landlords* mais facilmente comunicarem, desenvolvendo um mecanismo de *chat* interno ao sistema, bem como melhorar o sistema de *ratings* de modo a torná-lo ainda mais útil à decisão, de ambos os lados (*tenants* e *landlords*).

Por fim, no que toca ao *deployment*, o grupo também considera que, apesar de se encontrar satisfeito com o que foi obtido, seria possível melhorar acrescentando replicação na camada de base de dados (através de uma topologia *master-slave*), de forma a tornar o sistema mais resiliente a falhas, bem como aumentar a sua disponibilidade e performance, principalmente nas operações de leitura.

## A Docker Swarm/Stack: Especificação

```
version: "3.8"

services:
  web-server:
    image: vascoalramos/roomie-web
    ports: "80:80"
    depends_on:
      - postgres
      - app-server
    deploy:
      replicas: 3
      restart_policy:
        condition: any

  app-server:
    image: vascoalramos/roomie-app
    ports: "8083:8083"
    depends_on:
      - postgres
    volumes:
      - /nfs/roomie/images:/usr/src/app/images:z
      - /nfs/roomie/uploads:/usr/src/app/uploads:z
    deploy:
      replicas: 3
      restart_policy:
        condition: any

  postgres:
    image: postgres
    ports: "5433:5432"
    environment:
      - POSTGRES_DB=roomie
      - POSTGRES_USER=roomie
      - POSTGRES_PASSWORD=passw0rd
    command: [ "-c", "max_connections=1000" ]
    volumes:
      - /nfs/roomie/pg:/var/lib/postgresql/data/:z
    deploy:
      replicas: 1
      restart_policy:
        condition: any

volumes:
  database-data:
```

Código 1: Docker Swarm/Stack: Especificação