Universidade do Minho

Mestrado em Engenharia Informática

Arquiteturas Aplicacionais
&
Sistemas Interativos Confiáveis

SicaVibe

Grupo 2

Ana Murta (PG50184)

Beatriz Oliveira (PG50942)

Gonçalo Soares (PG50393)

Joana Alves (PG50457)

Vicente Moreira (PG50799)

Junho 2023

Conteúdo

1 Introdução **9**
1.1 Contextualização . . . 9
1.2 Definição do Produto . . . 9
1.3 Motivação e Objetivos . . . 9

2 Utilizadores do Produto **10**
2.1 Hóspede Não Autenticado . . . 10
2.2 Hóspede Autenticado . . . 10
2.3 Funcionário . . . 10
2.4 Administrador . . . 10

3 Convenções de Nomenclatura e Definições **11**
3.1 Modelo de Domínio . . . 11
3.2 Definição de Entidades . . . 11

4 Âmbito do Produto **13**
4.1 Diagrama de *Use Cases* . . . 13
4.2 Atores . . . 13
4.3 Descrição dos *Use Cases* . . . 14
4.3.1 Login . . . 14
4.3.2 Registar . . . 14
4.3.3 Consultar Hotéis . . . 15
4.3.4 Consultar Tipo de Quarto . . . 15
4.3.5 Apagar Conta . . . 16
4.3.6 Editar Perfil . . . 16
4.3.7 Efetuar Reserva . . . 17
4.3.8 Cancelar Reserva . . . 18
4.3.9 Consultar Reservas . . . 18
4.3.10 Efetuar Check-In . . . 19
4.3.11 Alterar Estado do Quarto . . . 20

5 Técnicas de Levantamento de Requisitos **21**
5.1 Análise de Domínio . . . 21
5.2 *Personas* . . . 21
5.2.1 Administrador . . . 21
5.2.2 Funcionário . . . 21
5.2.3 Cliente . . . 22

6 Requisitos do Sistema **23**
6.1 Requisitos Funcionais . . . 23
6.1.1 Requisitos sobre o Sistema . . . 23
6.1.2 Requisitos sobre o Administrador . . . 29
6.1.3 Requisitos sobre o Funcionário . . . 32
6.1.4 Requisitos sobre o Hóspede Não Autenticado . . . 34
6.1.5 Requisitos sobre o Hóspede Autenticado . . . 35

6.2 Requisitos Não Funcionais . . . 37
6.2.1 Requisitos de Aparência . . . 37
6.2.2 Requisitos de Usabilidade . . . 38
6.2.3 Requisitos de Desempenho . . . 39
6.2.4 Requisitos de Escalabilidade . . . 41
6.2.5 Requisitos Operacionais . . . 42
6.2.6 Requisitos de Manutenção e Suporte . . . 42
6.2.7 Requisitos de Segurança . . . 43
6.2.8 Requisitos Culturais e Políticos . . . 45
6.2.9 Requisitos Legais . . . 45

7 Modelos de Tarefas **46**
7.1 Modelo 1: Hóspede autenticado fazer uma reserva . . . 46
7.2 Modelo 2: Efetuar Check In . . . 46
7.3 Modelo 3: Efetuar Check Out . . . 46
7.4 Modelo 4: Hóspede autenticado cancelar uma reserva. . . . 47

8 Protótipos de Baixa Fidelidade **48**
8.1 Página Home . . . 48
8.2 Página de Login . . . 49
8.3 Página de Registar . . . 49
8.4 Página de um Hotel . . . 50
8.5 Página de Perfil de um Utilizador . . . 51
8.6 Página de Editar Perfil de um Utilizador . . . 52
8.7 Página de Alterar Palavra-passe de um Utilizador . . . 52
8.8 Página de um Funcionário . . . 53
8.9 Página de um Administrador . . . 54
8.10 Página de uma Reserva . . . 56

9 Estratégia da Solução **59**
9.1 Padrão Arquitetural e Tecnologias . . . 59
9.2 *Servidor Aplicacional* . . . 60
9.2.1 Hibernate ORM . . . 60
9.2.2 Sping Boot Framework . . . 60
9.2.2.1 *OpenAPI & Swagger* . . . 60
9.2.3 Diagrama de Classes (PIM) . . . 62
9.2.3.1 Geração de Código . . . 62
9.2.4 Listagem de Endpoints Desenvolvidos . . . 63
9.2.4.1 No Auth . . . 63
9.2.4.2 Hospede . . . 64
9.2.4.3 Funcionário . . . 64
9.2.4.4 Administrador . . . 64
9.3 Interface Gráfica . . . 65
9.3.1 Estrutura da Solução . . . 65
9.3.1.1 *Views* . . . 65
9.3.1.2 Componentes . . . 65
9.3.2 Assets . . . 66

9.3.3 Hooks . . . 66
9.3.4 Layouts . . . 66
9.3.5 Services . . . 66
9.3.6 Stores . . . 66
9.3.7 Types . . . 66

10 *Deployment* **67**
10.1 Back-end . . . 67
10.1.1 Requisitos . . . 67
10.1.2 Base de Dados . . . 67
10.1.3 Servidor Aplicacional . . . 68
10.2 Front-end . . . 68
10.2.1 Servidor Web de Desenvolvimento . . . 68
10.2.2 Servidor Web de Produção . . . 69

11 Apresentação das Interfaces Finais **70**
11.1 Páginas de um Funcionário . . . 70
11.2 Páginas de um Administrador . . . 72
11.3 Páginas de um Hóspede . . . 75
11.4 Páginas de uma Reserva . . . 81

12 Análise de Usabilidade **83**
12.1 Princípios de Usabilidade . . . 83
12.1.1 *Learnability* . . . 83
12.1.2 *Flexibility* . . . 83
12.1.3 *Robustness* . . . 84
12.2 Heurísticas de Nielsen . . . 84
12.2.1 *Visibility of system status* . . . 84
12.2.2 *Match between system and the real world* . . . 85
12.2.3 *User control and freedom* . . . 85
12.2.4 *Consistency and standards* . . . 86
12.2.5 *Error prevention* . . . 86
12.2.6 *Recognition rather than recall* . . . 86
12.2.7 *Flexibility and efficiency of use* . . . 86
12.2.8 *Aesthetic and minimalist design* . . . 86
12.2.9 *Help users recognize and recover from errors* . . . 87
12.2.10 *Help and documentation* . . . 87

13 Conclusões e Trabalho Futuro **88**

14 Anexos **90**

Lista de Figuras

1 Modelo de Domínio. 11
2 Diagrama de Use Cases. 13
3 Diagrama de Tarefa 1 - Hóspede autenticado fazer uma reserva. 46
4 Diagrama de Tarefa 2 - Efetuar Check In. 46
5 Diagrama de Tarefa 3 - Efetuar Check Out. 46
6 Diagrama de Tarefa 4 - Hóspede autenticado cancelar uma reserva. 47
7 *Mockup* da Página Home. 48
8 *Mockup* da Página de Login. 49
9 *Mockup* da Página de Registar. 49
10 *Mockup* da Página de um Hotel. 50
11 *Mockup* da Página do Perfil de um Utilizador I. 51
12 *Mockup* da Página do Perfil de um Utilizador II. 51
13 *Mockup* da Página do Editar Perfil de um Utilizador. 52
14 *Mockup* da Página de Alterar Palavra-passe Perfil de um Utilizador. 52
15 *Mockup* da Página de um Funcionário. 53
16 *Mockup* da Página de um Funcionário II (Reservas). 53
17 *Mockup* da Página de um Administrador I. 54
18 *Mockup* da Página de um Administrador II (Reservas). 54
19 *Mockup* da Página de um Administrador III (Funcionários). 55
20 *Mockup* da Página de um Administrador IV (Promoções). 55
21 *Mockup* da Página de um Administrador V (Quartos). 56
22 *Mockup* da Página de Reserva I. 56
23 *Mockup* da Página de Reserva II. 57
24 *Mockup* da Página de Reserva III. 57
25 *Mockup* da Página de Reserva IV. 58
26 *Mockup* da Página de Reserva V. 58
27 Arquitetura Geral. 59
28 Exemplo *WebSite*. 61
29 Exemplo Documentação *Endpoint*. 61
30 Diagrama de Classes. 62
31 Modelo Relacional. 63
32 Página de um Funcionário. 70
33 Página de um Funcionário II ("*See Reservation*"). 70
34 *Pop up* de Confirmação de "*Check In*"do Perfil Funcionário. 71
35 *Pop up* de Confirmação de "*Check Out*"do Perfil Funcionário. 71
36 Página de um Administrador I ("*Reservations*"). 72
37 Página de um Administrador II ("*See Reservation*"). 72
38 Página de um Administrador III ("*Employees*"). 73
39 Página de um Administrador IV ("*Add Employee*"). 73
40 Página de um Administrador V ("*Rooms*"). 74
41 Página de um Administrador VI ("*Add New Room*"). 74
42 Página Inicial. 75
43 Página de um Hotel. 76
44 Página de *Login*. 77

45 Página de Registo. 77
46 Página de Perfil Hóspede. 78
47 *Pop up* de Confirmação de Eliminação de Perfil Hóspede. 78
48 *Pop up* de Confirmação de Eliminação de Reserva do Perfil Hóspede. 79
49 *Pop up* de Confirmação antes de permitir alterar a password do Perfil Hóspede. . . 79
50 Página de Editar Perfil Hóspede. 80
51 Página de Alterar Password. 80
52 Efetuar Reserva Passo 1. 81
53 Efetuar Reserva Passo 2. 81
54 Efetuar Reserva Passo 3. 82
55 Efetuar Reserva Passo 4. 82
56 Mensagem de confirmação de alteração de password com sucesso. 84
57 Mensagem de confirmação de edição do perfil com sucesso. 85
58 Mensagem de confirmação de registo de nova conta com sucesso. 85
59 Reservada marcada. 85
60 Reserva cancelada. 85
61 Erro no registo. 87

Lista de Tabelas

1 Especificação de *Use Case* Login. . . . 14
2 Especificação de *Use Case* Registar Hóspede. . . . 14
3 Especificação de *Use Case* Consultar Hotéis. . . . 15
4 Especificação de *Use Case* Consultar Tipo de Quarto. . . . 15
5 Especificação de *Use Case* Apagar Conta. . . . 16
6 Especificação de *Use Case* Editar Perfil. . . . 16
7 Especificação de *Use Case* Efetuar Reserva. . . . 17
8 Especificação de *Use Case* Cancelar Reserva. . . . 18
9 Especificação de *Use Case* Consultar Reservas. . . . 18
10 Especificação de *Use Case* Efetuar Check-In. . . . 19
11 Especificação de *Use Case* Alterar Estado do Quarto. . . . 20
12 Exemplo especificação de Requisitos. . . . 23
13 Requisito funcional 1. . . . 23
14 Requisito funcional 2. . . . 23
15 Requisito funcional 3. . . . 24
16 Requisito funcional 4. . . . 24
17 Requisito funcional 5. . . . 24
18 Requisito funcional 6. . . . 25
19 Requisito funcional 7. . . . 25
20 Requisito funcional 8. . . . 25
21 Requisito funcional 9. . . . 25
22 Requisito funcional 10. . . . 26
23 Requisito funcional 11. . . . 26
24 Requisito funcional 12. . . . 26
25 Requisito funcional 13. . . . 27
26 Requisito funcional 14. . . . 27
27 Requisito funcional 15. . . . 27
28 Requisito funcional 16. . . . 27
29 Requisito funcional 17. . . . 28
30 Requisito funcional 18. . . . 28
31 Requisito funcional 19. . . . 28
32 Requisito funcional 20. . . . 29
33 Requisito funcional 21. . . . 29
34 Requisito funcional 22. . . . 29
35 Requisito funcional 23. . . . 30
36 Requisito funcional 24. . . . 30
37 Requisito funcional 25. . . . 30
38 Requisito funcional 26. . . . 30
39 Requisito funcional 27. . . . 31
40 Requisito funcional 28. . . . 31
41 Requisito funcional 29. . . . 31
42 Requisito funcional 30. . . . 31
43 Requisito funcional 31. . . . 32
44 Requisito funcional 32. . . . 32

45 Requisito funcional 33. 32
46 Requisito funcional 34. 32
47 Requisito funcional 35. 33
48 Requisito funcional 36. 33
49 Requisito funcional 37. 33
50 Requisito funcional 38. 33
51 Requisito funcional 39. 34
52 Requisito funcional 40. 34
53 Requisito funcional 41. 34
54 Requisito funcional 42. 35
55 Requisito funcional 43. 35
56 Requisito funcional 44. 35
57 Requisito funcional 45. 36
58 Requisito funcional 46. 36
59 Requisito funcional 47. 36
60 Requisito funcional 48. 36
61 Requisito funcional 49. 37
62 Requisito não funcional 1. 37
63 Requisito não funcional 2. 37
64 Requisito não funcional 3. 38
65 Requisito não funcional 4. 38
66 Requisito não funcional 5. 38
67 Requisito não funcional 6. 39
68 Requisito não funcional 7. 39
69 Requisito não funcional 8. 39
70 Requisito não funcional 9. 39
71 Requisito não funcional 10. 40
72 Requisito não funcional 11. 40
73 Requisito não funcional 12. 40
74 Requisito não funcional 13. 40
75 Requisito não funcional 14. 41
76 Requisito não funcional 15. 41
77 Requisito não funcional 41
78 Requisito não funcional 16. 42
79 Requisito não funcional 17. 42
80 Requisito não funcional 18. 42
81 Requisito não funcional 19. 42
82 Requisito não funcional 20. 43
83 Requisito não funcional 21. 43
84 Requisito não funcional 22. 43
85 Requisito não funcional 23. 43
86 Requisito não funcional 24. 44
87 Requisito não funcional 25. 44
88 Requisito não funcional 26. 44
89 Requisito não funcional 27. 44
90 Requisito não funcional 28. 45
91 Requisito não funcional 29. 45

92 Requisito não funcional 30. 45

1 Introdução

Este trabalho prático foi realizado no âmbito das unidades curriculares Arquiteturas Aplicacionais e Sistemas Interativos Confiáveis, que pertencem ao perfil de Engenharia de Aplicações do Mestrado em Engenharia Informática da Universidade do Minho.

Durante a sua realização pretendeu-se desenvolver uma aplicação com as suas diversas camadas aplicacionais, mais especificamente, a camada de dados, da lógica de negócio e de apresentação. Para além disso, esta deve ser uma aplicação *web* e suportar grandes volumes de carga.

1.1 Contextualização

A indústria hoteleira enfrentou tempos desafiadores durante a pandemia de COVID-19, com as restrições de viagens e as medidas de distanciamento social que impactaram significativamente a procura pela mesma. No entanto, com o término gradual do *lockdown* e o progresso na vacinação, as pessoas começam a recuperar a sua liberdade e a sentir uma forte vontade de vivenciar novas experiências. Neste contexto, uma aplicação de reservas de quartos em hotéis desempenha um papel fundamental ao fornecer uma plataforma conveniente e segura para que os indivíduos possam planear e reservar as suas acomodações com antecedência, permitindo-lhes satisfazer o desejo de explorar o mundo novamente.

1.2 Definição do Produto

O produto é direcionado a um público que se sinta atraído pela liberdade de viajar e pela facilidade de reserva de alojamento.

Desta forma, o produto deverá providenciar o registo de novos utilizadores, bem como a autenticação dos mesmos. De seguida, deverá ser disponibilizado ao utilizador uma listagem e descrição breve dos hotéis disponíveis no sistema, e, caso selecione um destes, será apresentada toda a informação presente no sistema sobre o mesmo.

O utilizador deverá também ter acesso a um histórico de reservas efetuadas, conseguindo observar o seu estado. Para além disto, deverá conseguir cancelar reservas antes da data de *check-in* respetiva.

1.3 Motivação e Objetivos

O *SicaVibe* surge como resposta ao crescimento da procura por alojamento após o *lockdown* da pandemia de COVID-19. Desta forma, pretende-se com este produto oferecer uma alternativa móvel aos seus utilizadores, podendo estes realizar reservas desde o conforto de suas casas.

2 Utilizadores do Produto

Nesta secção, com o intuito de desenvolver, ainda mais, a fundamentação deste projeto, iremos, de seguida, apresentar todos os utilizadores relacionados com este, listando-se as funções de cada um.

2.1 Hóspede Não Autenticado

- **Nome:** Hóspede Não Autenticado
- **Função:** O Hóspede Não Autenticado possui funcionalidades reduzidas, podendo visualizar uma listagem com uma breve descrição dos vários hotéis presentes no sistema. Para além disso, pode também visualizar a informação de um hotel em específico, como por exemplo, os seus respetivos tipos de quartos.

2.2 Hóspede Autenticado

- **Nome:** Hóspede Autenticado
- **Função:** O Hóspede Autenticado, para além das funcionalidades do Hóspede Não Autenticado, poderá realizar reservas, alterar detalhes da sua conta pessoal e ainda aceder ao seu histórico de reservas.

2.3 Funcionário

- **Nome:** Funcionário
- **Função:** O Funcionário é responsável por realizar o *check-in* e/ou o *check-out* de uma dada reserva, podendo ainda marcar os vários quartos como ocupados (após o *check-in*), por limpar (após o *check-out*) e, por fim, como livres (após a limpeza).

2.4 Administrador

- **Nome:** Administrador
- **Função:** O Administrador é responsável pela gestão dos vários funcionários e dos quartos dos diversos hotéis, podendo também realizar as operações de *check-in* e *check-out*.

3 Convenções de Nomenclatura e Definições

Apresentamos, de seguida, o modelo de domínio deste sistema e as suas respetivas entidades, tendo esta informação sido desenvolvida considerando os dados apresentados nas secções anteriores.

3.1 Modelo de Domínio

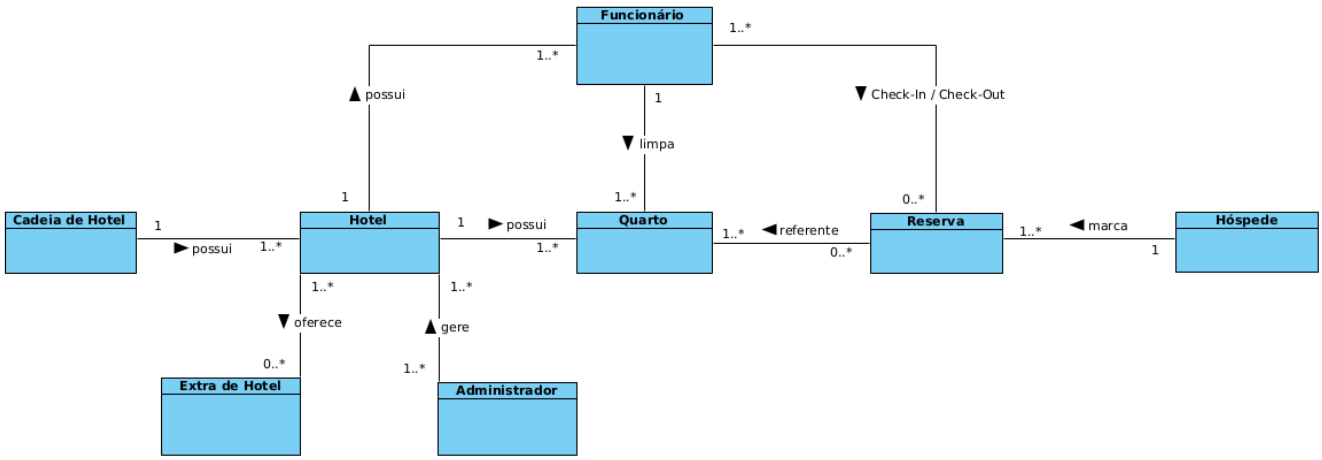


Figura 1: Modelo de Domínio.

3.2 Definição de Entidades

Nesta secção, são abordadas as diferentes entidades presentes no Modelo de Domínio apresentado na secção anterior (figura 1).

- **Hóspede Não Autenticado**: Utilizador visitante do sistema, cujo o objetivo da aplicação é levar estes a registarem-se, de forma a tirarem partido das restantes funcionalidades da aplicação.
- **Hóspede Autenticado**: Utilizador principal do sistema, podendo este ser capaz de realizar reservas e de aceder a outras funcionalidades da aplicação.
- **Funcionário**: Utilizador encarregue da limpeza de quartos, assim como o *check-in* e *check-out* das reservas do Hotel.
- **Administrador**: Utilizador encarregue da gestão geral do sistema, como o registo de novos funcionários e administradores.
- **Reserva**: Trata-se da principal atividade e funcionalidade do sistema. Esta é composta pela informação dos vários Quartos reservados num determinado Hotel, assim como a informação do Hóspede Autenticado que a realizou.
- **Quarto**: Entidade interna ao sistema, que representa um quarto físico num determinado Hotel, possuindo um determinado Tipo de Quarto. Os Hóspedes não possuem acesso a esta entidade.

- **Tipo de Quarto**: Representa os vários tipos de quartos possíveis dentro de um Hotel, podendo estes ser visualizados pelos Hóspedes. Cada Tipo de Quarto tem associado um conjunto de características específicas.
- **Serviço Extra**: Correspondem aos Serviços Extras presentes num dado Hotel, como o *spa*, o acesso ao parque de estacionamento, entre outros.
- **Hotel**: Entidade principal do sistema, composta por Quartos, Funcionários, Serviços Extra, pelas Reservas correspondentes e por outras informações.

4 Âmbito do Produto

4.1 Diagrama de *Use Cases*

De maneira a compreender melhor o contexto do sistema, vamos apresentar um diagrama de *Use Cases*. Neste vão ser explicitadas algumas das principais funcionalidades do sistema, bem como os atores do mesmo.

Neste diagrama, é ainda possível identificar quais as funcionalidades associadas a cada tipo de ator do sistema.

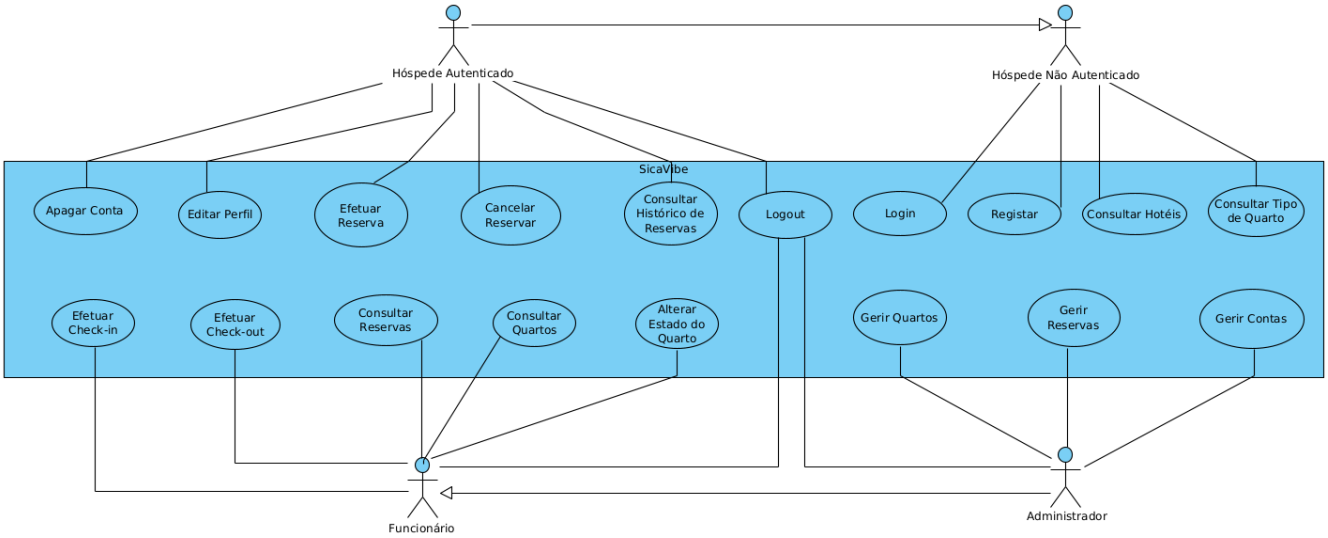


Figura 2: Diagrama de Use Cases.

4.2 Atores

Como é possível visualizar no diagrama anterior, o nosso sistema suporta quatro tipos de atores: o Hóspede Não Autenticado, o Hóspede Autenticado, o Funcionário e o Administrador.

- O **Hóspede Não Autenticado** é o utilizador visitante do nosso sistema, tendo funcionalidades reduzidas.
- O **Hóspede Autenticado** é o utilizador registado no nosso sistema, onde vai ter acesso a todas as funcionalidades que a plataforma oferece ao público-alvo.
- O **Funcionário** é o ator que tratará do *check-in* e *check-out* das reservas, assim como da gestão do estado dos quartos.
- O **Administrador** é responsável pela gestão do sistema, incluindo a gestão dos quartos, dos funcionários e das reservas presentes no mesmo.

4.3 Descrição dos *Use Cases*

Nesta secção iremos descrever o comportamento de cada *use case*, detalhando todos os passos necessários dos vários atores envolvidos até ao término do mesmo.

4.3.1 Login

O *use case* Login consiste na autenticação de um utilizador (hóspede, funcionário e administrador) que se encontra no sistema, sendo o seu *login* realizado através dos seguintes parâmetros: *email* e *password*. Os casos de erro acontecem quando o sistema deteta uma tentativa de autenticar um utilizador que não se encontra registado ou caso a *password* do utilizador não seja correspondente com a que existe no sistema.

USE-CASE:	Login	
Ator:	Utilizador	
Pré-Condição:	Ator não está autenticado	
Pós-Condição:	Ator está autenticado	
Cenário Normal	**Ator Input**	**System Response**
	2 - Fornece credenciais.	1- Pede credenciais de acesso. 3 - Valida credenciais. 4 - Informa que o utilizador está autenticado.
Exceção 1 : [credenciais erradas] (Passo 3)]		3.1 - Informa o utilizador do erro de credenciais. 3.2 - Volta ao passo 2

Tabela 1: Especificação de *Use Case* Login.

4.3.2 Registar

O *use case* Registar para o ator Hóspede consiste no registo de um hóspede, que ainda não se encontra no sistema. Os casos de erro acontecem quando o sistema deteta uma tentativa de inserir um hóspede que já se encontra registado ou caso alguns dos campos pessoais não cumpram os requisitos impostos pelo grupo.

USE-CASE:	Registar	
Ator:	Hóspede	
Pré-Condição:	Ator não está registado no sistema	
Pós-Condição:	Ator está registado	
Cenário Normal	**Ator Input**	**System Response**
	2 - Fornece informação.	1- Pede informação necessária ao registo. 3 - Valida informação. 4 - Informa que o utilizador está registado.
Exceção 1 : [informações inválidas] (passo 3)		3.1 - Informa o utilizador que as informações fornecidas são inválidas, especificando quais os campos errados. 3.2 - Volta ao passo 2.

Tabela 2: Especificação de *Use Case* Registar Hóspede.

4.3.3 Consultar Hotéis

O *use case* Consultar Hotéis para o ator Hóspede, consiste na visualização dos vários hotéis presentes na cadeia SicaVibe.

USE-CASE:	Consultar Hotéis	
Ator:	Hóspede	
Pré-Condição:		
Pós-Condição:	Ator consulta hotéis	
Cenário Normal	**Ator Input**	**System Response**
	1 - Acede página principal.	2 - Apresenta os vários hotéis da cadeia.

Tabela 3: Especificação de *Use Case* Consultar Hotéis.

4.3.4 Consultar Tipo de Quarto

O *use case* Consultar Tipo de Quarto para o ator Hóspede, consiste na visualização dos vários tipos de quarto presentes num determinado hotel da cadeia SicaVibe. Desta forma, este *use case* é feito no seguimento no anterior.

USE-CASE:	Consultar Tipo de Quarto	
Ator:	Hóspede	
Pré-Condição:		
Pós-Condição:	Ator consulta hotéis	
Cenário Normal	**Ator Input**	**System Response**
	1 - Acede página do hotel em específico.	2 - Apresenta as várias informações do hotel, incluindo os seus tipos de quartos.

Tabela 4: Especificação de *Use Case* Consultar Tipo de Quarto.

4.3.5 Apagar Conta

O *use case* Apagar Conta para o ator Hóspede, consiste na eliminação dos seus dados do sistema, incluindo os dados pessoais e as respetivas reservas feitas pelo mesmo.

USE-CASE:	Apagar Conta	
Ator:	Hóspede	
Pré-Condição:	Ator tem uma conta	
Pós-Condição:	Conta do ator eliminada	
Cenário Normal	**Ator Input**	**System Response**
	1 - Ator seleciona a opção de eliminar conta. 3- Confirma intenção.	2 - Confirma a intenção de eliminar conta. 4 - Sistema elimina conta do ator.

Tabela 5: Especificação de *Use Case* Apagar Conta.

4.3.6 Editar Perfil

O *use case* Editar Perfil para o ator Hóspede, consiste na modificação dos seus dados pessoais, que são passíveis de serem alterados (Nome, Morada e Número de Telemóvel e *Password*). Os casos de erro acontecem quando os dados introduzidos não são válidos segundo as diretivas impostas.

USE-CASE:	Editar Perfil	
Ator:	Hóspede	
Pré-Condição:	Ator tem uma conta	
Pós-Condição:	Ator edita o seu perfil	
Cenário Normal	**Ator Input**	**System Response**
	1 - Ator seleciona a opção de editar perfil 3 - Edita os campos que pretende alterar.	2 - Apresenta o formulário de edição de perfil 4 - Valida informações modificadas. 5 - Altera informações do ator.
Exceção 1 : [informações inválidas] (passo 4)		4.1 - Informa o utilizador que as informações fornecidas são inválidas, especificando quais os campos errados. 4.2 - Volta ao passo 3.

Tabela 6: Especificação de *Use Case* Editar Perfil.

4.3.7 Efetuar Reserva

O *use case* Efetuar Reserva para o ator Hóspede consiste na realização de uma reserva num dado Hotel da cadeia SicaVibe. Para isso, este terá de fornecer várias informações de modo a validar a sua reserva. Os casos de erro acontecem quando as opções escolhidas não são válidas ou não se encontram disponíveis, como por exemplo, o nº de quartos pedidos entre as datas definidas.

USE-CASE:	Efectuar Reserva	
Ator:	Hóspede	
Pré-Condição:	Ator autenticado	
Pós-Condição:	Ator realiza uma reserva	
Cenário Normal	**Ator Input**	**System Response**
	1 - Seleciona o Hotel onde pretende efetuar a reserva. 2 - Seleciona a opção de efetuar reserva. 4 - Preenche as datas de entrada e saída para a reserva. 7 - Seleciona os quartos. 10 - Seleciona serviços extra. 13 - Confirma reserva.	3 - Apresenta formulário para escolha de datas. 5 - Valida as datas. 6 - Apresenta formulário para escolha de quartos. 8 - Valida quartos escolhidos. 9 - Apresenta formulário para escolha de serviços extra. 11 - Valida serviços extra. 12 - Apresenta resumo da reserva. 14 - Efetua Reserva.
Exceção 1 : [datas inválidas] (passo 5)		5.1 - Informa o utilizador que as datas fornecidas são inválidas. 5.2 - Volta ao passo 4.
Exceção 2 : [nº quartos inválidos] (passo 8)		8.1 - Informa o utilizador que o nº de quartos fornecidos é inválido. 8.2 - Volta ao passo 7.
Exceção 3 : [serviços extra inválidos] (passo 11)		8.1 - Informa o utilizador que o nº de quartos fornecidos é inválido. 8.2 - Volta ao passo 7.

Tabela 7: Especificação de *Use Case* Efetuar Reserva.

4.3.8 Cancelar Reserva

O *use case* Cancelar Reserva para o ator Hóspede, consiste no cancelamento de uma reserva que ainda não foi iniciada, até 24 horas antes do seu suposto início. Os casos de erro acontecem quando a reserva não pode ser cancelada, isto é, no caso de já faltarem menos de 24 horas para o seu início.

USE-CASE:	Cancelar Reserva	
Ator:	Hóspede	
Pré-Condição:	Ator autenticado	
Pós-Condição:	Ator cancela a reserva	
Cenário Normal	**Ator Input**	**System Response**
	1 - Seleciona histórico de reservas. 3 - Seleciona Cancelar Reserva.	2 - Apresenta histórico de reservas. 4 - Valida se operação é válida. 5 - Cancela Reserva.
Exceção 1 : [reserva não pode ser cancelada] (passo 4)		4.1 - Informa o utilizador que a reserva não pode ser cancelada. 4.2 - Volta ao passo 2.

Tabela 8: Especificação de *Use Case* Cancelar Reserva.

4.3.9 Consultar Reservas

O *use case* Consultar Reservas para o ator Funcionário, consiste na consulta de todas as reservas efetuadas no Hotel onde este está empregado.

USE-CASE:	Consultar Reservas	
Ator:	Funcionário	
Pré-Condição:	Ator autenticado	
Pós-Condição:	Ator consulta as reservas	
Cenário Normal	**Ator Input**	**System Response**
	1 - Ator seleciona a opção de consultar reservas.	2 - Apresenta a listagem das várias reservas no Hotel.

Tabela 9: Especificação de *Use Case* Consultar Reservas.

4.3.10 Efetuar Check-In

O *use case* Efetuar Check-In para o ator Funcionário, consiste na confirmação de início de uma dada reserva. Este ato é realizado na receção do Hotel, onde o Hóspede fornece os dados necessários para identificação da reserva, sendo depois confirmado o *check-in*. Os casos de erro ocorrem quando não existe nenhuma reserva com os dados introduzidos, ou a reserva já se encontra *checked-in*.

Desta forma, o *use case* Efetuar Check-Out é similar a este, apenas alterando o estado da reserva para confirmar a sua completude e a alteração do estado dos quartos reservados para o estado por limpar.

USE-CASE:	Efetuar Check-In	
Ator:	Funcionário	
Pré-Condição:	Ator autenticado	
Pós-Condição:	Ator realiza o check-in	
Cenário Normal	**Ator Input**	**System Response**
	1 - Seleciona opção de pesquisar reserva. 3 - Filtra as reservas a partir de dados do Hóspede. 5 - Seleciona Reserva. 7 - Seleciona opção de *Check-In*.	2 - Apresenta as diversas reservas presentes no hotel. 4 - Apresenta lista de reservas filtrada. 6 - Apresenta informações da reserva. 8 - Valida opção escolhida. 9 - Estado da Reserva é alterado para *checked-in* e o estado dos quartos alterado para ocupados.
Exceção 1 : [reserva não pode ser cancelada] (passo 8)		8.1 - Informa o ator que a reserva não pode ser cancelada. 8.2 - Volta ao passo 6.

Tabela 10: Especificação de *Use Case* Efetuar Check-In.

4.3.11 Alterar Estado do Quarto

O *use case* Alterar Estado do Quarto do ator Funcionário consiste na alteração do estado dos quartos do Hotel onde este trabalha, nomeadamente para livre, ocupado ou por limpar.

USE-CASE:	Alterar Estado do Quarto	
Ator:	Funcionário	
Pré-Condição:	Ator autenticado	
Pós-Condição:	Ator altera o estado do quarto	
Cenário Normal	**Ator Input**	**System Response**
	1 - Ator seleciona a opção de consultar quartos.	2 - Apresenta lista de quartos do hotel.
	3 - Seleciona quarto.	4 - Apresenta informações do quarto.
	5 - Altera estado do quarto.	6 - Confirma alteração do estado do quarto.

Tabela 11: Especificação de *Use Case* Alterar Estado do Quarto.

5 Técnicas de Levantamento de Requisitos

A criação de um sistema para a gestão de uma cadeia de hotéis, tal como qualquer outro serviço, requer uma análise detalhada dos requisitos envolvidos, a fim de garantir que a solução desenvolvida atenda às necessidades específicas do negócio. Com isto em mente, recorremos a diversas técnicas de levantamento de requisitos, como a introspeção, a *Brainstorming*, a análise do domínio e as *Personas*. No entanto, é de destacar que a segunda técnica, *Brainstorming*, foi utilizada contudo num carácter mais oral, principalmente, durante as reuniões que realizamos enquanto grupo.

5.1 Análise de Domínio

Para a análise de domínio, o método utilizado e escolhido por toda a equipa foi a análise de outros *sites* de hotelaria, em concreto, *sites* que possibilitassem a funcionalidade de reserva de quartos. Assim, toda a equipa aprofundou os seus conhecimentos no entendimento da dinâmica de reservas de quartos assim como os passos necessários para executá-las com sucesso.

5.2 *Personas*

Criamos três *Personas*, uma referente a cada utilizador do sistema, com o intuito de facilitar não só o levantamento de requisitos, tal como já referimos, mas também para facilitar, posteriormente, a modelação das diversas interfaces, já que a obtenção de conhecimento relativo a cada utilizador existente influencia os objetivos de usabilidade e de desenho.

5.2.1 Administrador

Maria Santos, 35 anos, Lisboa

A Maria, que de criança tinha o sonho de um dia ter o seu próprio hotel, trabalha atualmente na área da administração. É uma profissional experiente, que supervisiono a gestão das várias propriedades hoteleiras de uma cadeia de hotéis.

Para além de garantir todas as funções que, normalmente se associam ao gerente de um hotel, como garantir que todas as operações do hotel estão a funcionar sem problemas, desde a receção e ao atendimento aos hóspedes até as equipas responsáveis pela limpeza, pelo restaurante e pelos eventos. A Maria também supervisiona as reservas, a disponibilidade dos quartos, a contratação de novos funcionários e, ainda, é responsável pela procura de soluções para maximizar a satisfação dos clientes e os lucros do hotel, como a criação de promoções.

Assim sendo, esta administrativa valoriza uma plataforma de gestão hoteleira que seja intuitiva, abrangente, que facilite a sua supervisão das diferentes operações do hotel e da disponibilidade de quartos e que ao mesmo tempo seja simples e direta.

5.2.2 Funcionário

Pedro Rodrigues, 28 anos, Coimbra

O Pedro que desde de sempre que adorou viajar, conhecer novas pessoas, novos lugares, mas também adorava a sua cidade, ajudar as pessoas e sobretudo mostrar a sua cidade aos turistas, a amigos e familiares de fora. Atualmente, é um dos rececionista de um hotel de luxo em Coimbra, a sua cidade Natal.

No seu dia a dia, é um profissional atencioso e prestativo, responsável por receber os hóspedes, fazer os *check-in* e os *check-out*, por lidar com os pagamentos dos hóspedes e por fornecer informações relativas ao hotel e à região.

Dada a sua necessidade de lidar com uma grande quantidade e variedade de informações e da sua necessidade de estar sempre atualizado relativamente à disponibilidade dos quartos do hotel, aos serviços oferecidos e aos eventos locais, o Pedro valoriza uma aplicação de gestão hoteleira que seja fácil de utilizar, com uma interface intuitiva. Para além disso, valoriza a rapidez da aplicação, para momentos em que existam vários clientes em espera, durante a realização de operações como o registo e a visualização de reservas, o *check-in* e o *check-out*.

5.2.3 Cliente

Gabriela Mendes, 32 anos, Porto

A Gabriela, uma executiva de uma empresa multinacional, é uma cliente exigente e bem-sucedida, que está acostumada a viajar tanto a negócios, como em lazer. Durante as suas estadias nos diversos hotéis que frequenta, esta executiva valoriza a qualidade, o conforto, a diversidade, a privacidade e o atendimento excecional. Devido muitas vezes à pouca antecedência que tem conhecimento das suas viagens, a Gabriela opta pela realização das reservas das suas estadias online, pela sua facilidade e rapidez.

Assim sendo, a Gabriela espera que o serviço de reservas e o próprio web site da cadeia de hotéis proporcione uma experiência tranquila e personalizada desde o momento da reserva até o ao *Check Out*. Ela pretende encontrar uma aplicação intuitiva e amigável, que ofereça recursos acessíveis e fáceis para operações como a reserva de quartos e a existência de detalhes completos sobre os quartos e os serviços disponibilizados pelo hotel. Para além disso, a Gabriela também valoriza a disponibilidade de informação relativa às estadias anteriores e espera que a aplicação seja uma extensão do serviço de alta qualidade que ela pretende receber durante a sua estadia nesta cadeia de hotéis.

6 Requisitos do Sistema

Nesta fase, procedemos com a definição e formalização da estrutura e do comportamento da aplicação. Para isso, iremos definir os requisitos funcionais e não funcionais. Estando concluídas estas ações, inicializaremos a criação dos modelo de tarefas e a prototipagem da interface gráfica.

Para o levantamento de requisitos, utilizamos como forma de representação, a *requirement shell* do modelo de *Volere*, de forma a descrever concisamente os requisitos. No entanto, decidimos não utilizar desta representação, a secção dos Conflitos, uma vez que tratando-se, ainda, da modelação da aplicação, não é possível conhecer quais os conflitos criados na implementação de um dado requisito.

Requirement **Type**:
Description:
Rationale:
Originator:
Fit Criterion:
Priority:
History:

Tabela 12: Exemplo especificação de Requisitos.

6.1 Requisitos Funcionais

6.1.1 Requisitos sobre o Sistema

Requirement #1 **Type**: 9
Description: O sistema deve disponibilizar a funcionalidade de reserva de quartos.
Rationale: Principal funcionalidade da aplicação.
Originator: Introspecção
Fit Criterion: Os hóspedes autenticados realizam reservas no sistema.
Priority: Must
History: 30/03/2023

Tabela 13: Requisito funcional 1.

Requirement #2 **Type**: 9
Description: O sistema deve permitir a autenticação de utilizadores já registados no sistema.
Rationale: É necessário saber quem acede ao sistema.
Originator: Introspecção
Fit Criterion: Acessibilidade do sistema.
Priority: Must
History: 30/03/2023

Tabela 14: Requisito funcional 2.

Requirement #3 **Type**: 9
Description: O sistema deve distinguir o tipo de utilizador autenticado.
Rationale: Deve ser apresentada a informação correta e adequada a cada tipo de utilizador.
Originator: Introspecção
Fit Criterion: A interface apresentada relaciona-se com o tipo de utilizador autenticado.
Priority: Must
History: 30/03/2023

Tabela 15: Requisito funcional 3.

Requirement #4 **Type**: 9
Description: Um quarto tem associado um tipo, que deverá obrigatoriamente ser 'Single Room', 'Twin Room', 'Deluxe Room' ou 'Presidential Suite'.
Rationale: Personalização da experiência dos hóspedes através da oferta de serviços variados.
Originator: Introspecção
Fit Criterion: Os hóspedes realizam reservas de acordo com os diversos tipos de quartos.
Priority: Must
History: 30/03/2023

Tabela 16: Requisito funcional 4.

Requirement #5 **Type**: 9
Description: Os utilizadores do sistema devem pertencer a um das quatro categorias: 'Hóspede', 'Funcionário' ou 'Administrador'.
Rationale: Permite que o sistema seja utilizado por vários tipos de utilizadores.
Originator: Introspecção
Fit Criterion: Todos os utilizadores conseguem aceder ao sistema.
Priority: Must
History: 30/03/2023

Tabela 17: Requisito funcional 5.

Requirement #6 **Type**: 9
Description: As reservas têm associadas um estado. Em que este pode corresponder a 'Marcada', 'A Decorrer', 'Terminada' ou 'Cancelada'.
Rationale: Agilizar a filtragem e a identificação das reservas no sistema.
Originator: Introspecção
Fit Criterion: Todas as reservas presentes no sistema encontram-se num destes estados.
Priority: Must
History: 30/03/2023

Tabela 18: Requisito funcional 6.

Requirement #7 **Type**: 9
Description: Um quarto tem obrigatoriamente um dos 3 estados: 'Livre', 'Ocupado' e 'Por Limpar'.
Rationale: Facilitar o processo de filtragem dos quartos.
Originator: Introspecção
Fit Criterion: Todos os quartos presentes no sistema têm um destes estados.
Priority: Must
History: 30/03/2023

Tabela 19: Requisito funcional 7.

Requirement #8 **Type**: 9
Description: Os Funcionários devem ter associados um estado tal como: 'Ativo', 'Inativo' ou 'Em Pausa'.
Rationale: Controlo dos horários dos funcionários.
Originator: Introspecção
Fit Criterion: Atualização do estado dos funcionários consoante o seu horário de trabalho.
Priority: Must
History: 30/03/2023

Tabela 20: Requisito funcional 8.

Requirement #9 **Type**: 9
Description: A aplicação deve permitir o registo de novos Hóspedes.
Rationale: É essencial para o aumento do alcance e utilização da aplicação.
Originator: Introspecção
Fit Criterion: Após a introdução e validação dos dados do utilizador, um novo Hóspede é criado na base de dados.
Priority: Must
History: 30/03/2023

Tabela 21: Requisito funcional 9.

Requirement #10 **Type**: 9
Description: A aplicação deve permitir ao utilizador escolher múltiplos quartos na mesma reserva.
Rationale: Sendo esta uma funcionalidade principal do sistema, é fundamental que seja garantida. O mesmo hóspede pode querer reservar mais quartos para a família e/ou amigos.
Originator: Introspecção
Fit Criterion: Reserva de quartos.
Priority: Must
History: 30/03/2023

Tabela 22: Requisito funcional 10.

Requirement #11 **Type**: 9
Description: O sistema deve disponibilizar uma lista dos tipos de quartos disponíveis no momento da reserva.
Rationale: Informar os utilizadores dos tipos de quartos livres no período da reserva.
Originator: Introspecção
Fit Criterion: Listar os tipos de quartos após a escolha das datas para a reserva.
Priority: Should
History: 30/03/2023

Tabela 23: Requisito funcional 11.

Requirement #12 **Type**: 9
Description: As reservas realizadas deverão apresentar o mínimo de informação.
Rationale: O utilizador querer saber o mínimo de pormenores das reservas, como por exemplo, a sua data e tipo de quarto escolhido.
Originator: Introspecção
Fit Criterion: Consultar histórico de reservas lista de todas as reversas dos utilizadores.
Priority: Should
History: 30/03/2023

Tabela 24: Requisito funcional 12.

Requirement #13 **Type**: 9
Description: O sistema deve permitir escolher extras durante a marcação da reserva.
Rationale: Permitir ao utilizador assinalar desde início os extras que pretende ter durante a sua estadia.
Originator: Introspecção
Fit Criterion: Reserva de quartos.
Priority: Should
History: 30/03/2023

Tabela 25: Requisito funcional 13.

Requirement #14 **Type**: 9
Description: A aplicação deve possibilitar o pagamento da reserva através de 2 métodos de pagamento distintos, Multibanco e MBWay.
Rationale: Aumentar o alcance de utilizadores, visto haver, atualmente, uma grande diversidade de preferências nos métodos de pagamento.
Originator: Introspecção
Fit Criterion: Método de pagamento.
Priority: Could
History: 30/03/2023

Tabela 26: Requisito funcional 14.

Requirement #15 **Type**: 9
Description: O sistema apenas deve permitir efetuar o pagamento da reserva no *check-out*.
Rationale: Facilitar o processo de pagamento.
Originator: Introspecção
Fit Criterion: Pagamento no *check-out*.
Priority: Could
History: 30/03/2023

Tabela 27: Requisito funcional 15.

Requirement #16 **Type**: 9
Description: Aquando do registo de um utilizador, o sistema tem de enviar um código de confirmação de email para o candidato a hóspede.
Rationale: Garantir que o email introduzido é o email correto.
Originator: Introspecção
Fit Criterion: O código é um valor aleatório de 10 dígitos que deve ser enviado por email ao utilizador para o mesmo confirmar o seu email.
Priority: Could
History: 30/03/2023

Tabela 28: Requisito funcional 16.

Requirement #17 **Type**: 9
Description: Quando um novo hóspede se regista, a aplicação deve oferecer-lhe uma promoção.
Rationale: Aumentar a aderência de utilizadores à aplicação.
Originator: Introspecção
Fit Criterion: Após o registo de um novo utilizador, é lhe oferecido uma promoção na sua primeira reserva.
Priority: Could
History: 30/03/2023

Tabela 29: Requisito funcional 17.

Requirement #18 **Type**: 9
Description: A aplicação deve ter uma página de ajuda com resposta às perguntas que podem surgir durante a utilização da mesma.
Rationale: Ajudar os utilizadores durante o uso da aplicação.
Originator: Introspecção
Fit Criterion: A aplicação deve ter sempre disponível a página Ajuda para o utilizador.
Priority: Could
History: 30/03/2023

Tabela 30: Requisito funcional 18.

Requirement #19 **Type**: 9
Description: O sistema deve permitir filtrar o histórico de reservas por datas e horas, e pelos estados de reserva.
Rationale: Oferecer uma visualização mais organizada das reservas do utilizador.
Originator: Introspecção
Fit Criterion: Filtrar histórico de reservas do utilizador e consultar reservas.
Priority: Could
History: 30/03/2023

Tabela 31: Requisito funcional 19.

Requirement #20 **Type**: 9
Description: O sistema deve permitir efetuar pesquisas.
Rationale: Oferecer mais rapidez para atingir o objetivo pretendido com a utilização do sistema.
Originator: Introspecção
Fit Criterion: Permitir ao utilizador pesquisar por hotel, por tipo de quarto e por valor máximo que pretende usar para a estadia.
Priority: Could
History: 30/03/2023

Tabela 32: Requisito funcional 20.

Requirement #21 **Type**: 9
Description: O sistema pode disponibilizar uma lista com os eventos a acontecer em cada hotel.
Rationale: Informar os utilizadores dos eventos prestes a acontecer.
Originator: Introspecção
Fit Criterion: Disponibilização dos eventos.
Priority: Could
History: 30/03/2023

Tabela 33: Requisito funcional 21.

6.1.2 Requisitos sobre o Administrador

Requirement: #22 **Type**: 9
Description: O administrador deve ter acesso às reservas efetuadas e a toda a informação que estejam associadas às mesmas.
Rationale: A gestão de cada hotel implica a gestão das diversas reservas.
Originator: Introspecção
Fit Criterion: O administrador consegue aceder à informação do sistema.
Priority: Must
History: 30/03/2023

Tabela 34: Requisito funcional 22.

Requirement: #23 **Type**: 9
Description: O administrador deverá conseguir registar novos funcionários, tendo de referir qual o Hotel que estes irão integrar.
Rationale: Permitir adicionar novos elementos à equipa de um hotel.
Originator: Introspecção
Fit Criterion: O administrador consegue inserir novos funcionários na base de dados.
Priority: Must
History: 30/03/2023

Tabela 35: Requisito funcional 23.

Requirement: #24 **Type**: 9
Description: O administrador deve ter a capacidade de modificar uma reserva.
Rationale: Permitir uma gestão facilitada das reservas.
Originator: Introspecção
Fit Criterion: O administrador consegue alterar as informações relativas a uma reserva.
Priority: Should
History: 30/03/2023

Tabela 36: Requisito funcional 24.

Requirement: #25 **Type**: 9
Description: O Administrador deve conseguir modificar a informação relativa a um quarto, para tal deve fornecer o ID/número do mesmo.
Rationale: O Administrador gere os quartos.
Originator: Introspecção
Fit Criterion: O Administrador modifica a informação dos quartos.
Priority: Should
History: 30/03/2023

Tabela 37: Requisito funcional 25.

Requirement: #26 **Type**: 9
Description: O Administrador deverá conseguir apagar contas relativas a antigos Funcionários, tendo de referir qual o seu nome ou ID.
Rationale: Permitir no caso de diminuição da equipa, que o sistema seja atualizado.
Originator: Introspecção
Fit Criterion: O Administrador consegue eliminar Funcionários da base de dados.
Priority: Could
History: 30/03/2023

Tabela 38: Requisito funcional 26.

Requirement: #27 **Type**: 9
Description: O Administrador deve conseguir retirar quartos a um dado Hotel, sendo apenas necessário fornecer o ID/número do mesmo.
Rationale: O Administrador gere os quartos.
Originator: Introspecção
Fit Criterion: O Administrador se necessário retira quartos de um Hotel.
Priority: Could
History: 30/03/2023

Tabela 39: Requisito funcional 27.

Requirement: #28 **Type**: 9
Description: O Administrador deve conseguir adicionar novos quartos a um dado Hotel. Sendo necessário fornecer a informações, como o nome do quarto, a descrição, a ocupação, o seu Hotel, o preço base e ainda, imagem do quarto.
Rationale: O Administrador gere os quartos.
Originator: Introspecção
Fit Criterion: O Administrador insere, caso exista, novos quartos.
Priority: Could
History: 30/03/2023

Tabela 40: Requisito funcional 28.

Requirement: #29 **Type**: 9
Description: O Administrador deve conseguir criar uma promoção. Para tal, deve fornecer informações como a data de início e de fim, a descrição, o valor, o hotel em que se realiza e o tipo de quarto associado.
Rationale: Facilitar e dinamizar o processo de criação de promoções.
Originator: Introspecção
Fit Criterion: O Administrador cria promoções.
Priority: Could
History: 30/03/2023

Tabela 41: Requisito funcional 29.

Requirement: #30 **Type**: 9
Description: O Administrador deve conseguir modificar uma promoção.
Rationale: Dinamização de tudo o que seja relacionado com as promoções. O Administrador faz a gestão das mesmas.
Originator: Introspecção
Fit Criterion: O Administrador modifica promoções.
Priority: Could
History: 30/03/2023

Tabela 42: Requisito funcional 30.

6.1.3 Requisitos sobre o Funcionário

Requirement: #31 **Type**: 9
Description: O Funcionário deverá poder modificar o estado das reservas.
Rationale: Dinamizar dos diferentes estados das reservas.
Originator: Introspecção
Fit Criterion: O Funcionário modifica o estado das reservas.
Priority: Must
History: 30/03/2023

Tabela 43: Requisito funcional 31.

Requirement: #32 **Type**: 9
Description: O Funcionário deverá ter acesso à listagem das várias reservas do hotel.
Rationale: Permitir a agilização das reservas.
Originator: Introspecção
Fit Criterion: O Funcionário consegue ver as reservas existentes para um dado dia.
Priority: Must
History: 30/03/2023

Tabela 44: Requisito funcional 32.

Requirement: #33 **Type**: 9
Description: O Funcionário deverá poder realizar o check-out de um cliente.
Rationale: Facilitar o processo de check-out.
Originator: Introspecção
Fit Criterion: O check-out é realizado pelo Funcionário.
Priority: Must
History: 30/03/2023

Tabela 45: Requisito funcional 33.

Requirement: #34 **Type**: 9
Description: O Funcionário deverá poder realizar o *check-in* de um cliente.
Rationale: Facilitar o processo de check-in e permitir o acesso aos quartos pelo hóspedes.
Originator: Introspecção
Fit Criterion: Realização do *check-in*.
Priority: Must
History: 30/03/2023

Tabela 46: Requisito funcional 34.

Requirement: #35 **Type**: 9
Description: O Funcionário deverá poder alterar o estado de um quarto.
Rationale: Um Funcionário deve, ao realizar o check-in, alterar o estado do quarto dos hóspedes para ocupado. Da mesma forma, ao realizar o check-out deve alterar o estado do mesmo para que seja possível a sua limpeza.
Originator: Introspecção
Fit Criterion: O Funcionário modifica o estados dos diferentes quartos.
Priority: Should
History: 30/03/2023

Tabela 47: Requisito funcional 35.

Requirement: #36 **Type**: 9
Description: O funcionário poderá consulta a lista de todos os quartos, obtendo a informação relativa à sua localização (o seu número).
Rationale: Automatizar e facilitar o processo que dá conhecimento dos quartos por limpar.
Originator: Introspecção
Fit Criterion: O funcionário tem acesso e consulta a lista dos quartos.
Priority: Should
History: 30/03/2023

Tabela 48: Requisito funcional 36.

Requirement: #37 **Type**: 9
Description: O Funcionário poderá estender o tempo da estadia/reserva de um Hóspede.
Rationale: Permitir modificações e extensões das reservas no sistema.
Originator: Introspecção
Fit Criterion: O Funcionário estende reservas, consoante a disponibilidade do hotel.
Priority: Could
History: 30/03/2023

Tabela 49: Requisito funcional 37.

Requirement: #38 **Type**: 9
Description: O Funcionário deverá ter acesso à lista de todos os quartos.
Rationale: Para suas diversas tarefas, o Funcionário necessita de ter acesso à listagem de todos os quartos.
Originator: Introspecção
Fit Criterion: O Funcionário consulta a lista de quartos.
Priority: Could
History: 30/03/2023

Tabela 50: Requisito funcional 38.

Requirement: #39 **Type**: 9
Description: Os funcionários poderão acrescentar observações acerca da limpeza do quarto (por exemplo, avarias ou mobílias danificadas).
Rationale: Agilizar a comunicação entre os diversos constituintes do sistema e a documentação relativa à exposição de problemas.
Originator: Introspecção
Fit Criterion: Introdução de observações.
Priority: Could
History: 30/03/2023

Tabela 51: Requisito funcional 39.

6.1.4 Requisitos sobre o Hóspede Não Autenticado

Requirement: #40 **Type**: 9
Description: O Hóspede não autenticado poderá consultar a lista dos Hotéis disponíveis para reserva dentro da cadeia de Hoteís SicaVibe.
Rationale: O agente tem acesso a uma variedade de hotéis para efetuar uma melhor escolha.
Originator: Introspecção
Fit Criterion: Todos os hotéis da SicaVibe são apresentados.
Priority: Must
History: 30/03/2023

Tabela 52: Requisito funcional 40.

Requirement: #41 **Type**: 9
Description: O Hóspede não autenticado poderá consultar informações acerca de um Hotel da cadeia SicaVibe. Estas devem conter o nome do Hotel, uma ou mais imagens do hotel e os seus arredores, uma descrição da caracterização do hotel, o endereço deste, os tipos de quartos que disponibiliza para reservas e os serviços extra presentes neste Hotel.
Rationale: O agente ficará informado acerca do Hotel onde poderá efetuar uma reserva.
Originator: Introspecção
Fit Criterion: As informações especificadas são apresentadas.
Priority: Must
History: 30/03/2023

Tabela 53: Requisito funcional 41.

Requirement: #42 **Type**: 9
Description: O Hóspede não autenticado poderá consultar informações acerca de um tipo de quarto da cadeia SicaVibe. Este deverá conter informação do nome do tipo de quarto, uma ou mais imagens do quarto, uma descrição da caracterização do quarto, o número de pessoas indicado pelo quarto e o o preço base por noite do quarto.
Rationale: O agente ficará informado acerca do tipo de quarto ao qual poderá efetuar uma reserva.
Originator: Introspecção
Fit Criterion: As informações especificadas são apresentadas.
Priority: Must
History: 30/03/2023

Tabela 54: Requisito funcional 42.

6.1.5 Requisitos sobre o Hóspede Autenticado

Requirement: #43 **Type**: 9
Description: O Hóspede autenticado durante o processo de reserva, deverá obter uma lista de quartos disponíveis para reserva durante as datas indicadas nesta, assim como a quantidade de quartos disponíveis.
Rationale: O agente fica informado da disponibilidade dos quartos.
Originator: Introspecção
Fit Criterion: A lista de quartos disponíveis é apresentado.
Priority: Must
History: 30/03/2023

Tabela 55: Requisito funcional 43.

Requirement: #44 **Type**: 9
Description: O Hóspede autenticado durante o processo de reserva, deve conseguir visualizar o preço total final da sua reserva.
Rationale: O agente fica informado da sua reserva.
Originator: Introspecção
Fit Criterion: O preço final da reserva é apresentado.
Priority: Must
History: 30/03/2023

Tabela 56: Requisito funcional 44.

Requirement: #45 **Type**: 9
Description: O Hóspede autenticado poderá consultar a lista de reservas efetuadas, sendo que estas podem estar terminadas, a ocorrer, canceladas ou por abrir.
Rationale: O agente gere as suas reservas.
Originator: Introspecção
Fit Criterion: A lista é apresentada.
Priority: Must
History: 30/03/2023

Tabela 57: Requisito funcional 45.

Requirement: #46 **Type**: 9
Description: O Hóspede autenticado poderá cancelar uma reserva, desde que o cancelamento seja efetuado 24h antes do dia de abertura da reserva.
Rationale: O agente gere as suas reservas.
Originator: Introspecção
Fit Criterion: A reserva é cancelada.
Priority: Should
History: 30/03/2023

Tabela 58: Requisito funcional 46.

Requirement: #47 **Type**: 9
Description: O Hóspede autenticado poderá editar informações do seu perfil.
Rationale: O agente controla a sua informação pessoal.
Originator: Introspecção
Fit Criterion: As novas informações são atualizadas.
Priority: Should
History: 30/03/2023

Tabela 59: Requisito funcional 47.

Requirement: #48 **Type**: 9
Description: O Hóspede autenticado poderá eliminar a sua conta, apagando toda a sua informação pessoal do sistema.
Rationale: O agente controla a sua informação pessoal.
Originator: Introspecção
Fit Criterion: A conta do agente é apagado do sistema.
Priority: Should
History: 30/03/2023

Tabela 60: Requisito funcional 48.

Requirement: #49 **Type**: 9
Description: O Hóspede autenticado poderá extender uma reserva que esteja a ocorrer, sendo necessário indicar uma nova data de saída e, após confirmação da disponibilidade dos quartos e a apresentação do novo preço final, a extensão é confirmada.
Rationale: O agente estende a sua reserva.
Originator: Introspecção
Fit Criterion: A reserva é estendida.
Priority: Could
History: 30/03/2023

Tabela 61: Requisito funcional 49.

6.2 Requisitos Não Funcionais

6.2.1 Requisitos de Aparência

Requirement: #1 **Type**:
Description: A aplicação deverá ser atrativa para utilizadores experientes na área hoteleira.
Rationale: Facilitar a transição dos utilizadores de outros sites para a aplicação.
Originator: Introspecção
Fit Criterion: A aplicação segue a mesma estrutura que outras aplicações na área.
Priority: Should
History: 30/03/2023

Tabela 62: Requisito não funcional 1.

Requirement: #2 **Type**:
Description: O sistema deverá ser simples, organizado e atrativo para utilizadores que nunca tiveram contacto prévio com este género de aplicações.
Rationale: Utilizadores que não tenham conhecimento na área devem preferir esta aplicação como principal plataforma de reservas.
Originator: Introspecção
Fit Criterion: Simplicidade e atratividade do produto.
Priority: Should
History: 30/03/2023

Tabela 63: Requisito não funcional 2.

6.2.2 Requisitos de Usabilidade

Requirement: #3 **Type**:
Description: Ao fim de 10 minutos, 90% dos utilizadores devem ser capazes de realizar a principal funcionalidade do sistema.
Rationale: Os utilizadores não precisam de possuir conhecimento prévio sobre a aplicação para a conseguir utilizar.
Originator: Introspecção
Fit Criterion: Utilização facilitada do serviço.
Priority: Should
History: 30/03/2023

Tabela 64: Requisito não funcional 3.

Requirement: #4 **Type**:
Description: Os meios de pagamento devem ser acompanhados de imagens alusivas aos mesmos.
Rationale: Facilitar a identificação de todos os meios disponíveis para estas operações.
Originator: Introspecção
Fit Criterion:
Priority: Could
History: 30/03/2023

Tabela 65: Requisito não funcional 4.

Requirement: #5 **Type**:
Description: A aplicação deve permitir uma procura facilitada para os diversos tipos de quartos.
Rationale: Disponibilizar uma interatividade facilitada ao utilizador.
Originator: Introspecção
Fit Criterion: Interatividade.
Priority: Won't
History: 30/03/2023

Tabela 66: Requisito não funcional 5.

Requirement: #6 **Type**:
Description: Numa primeira interação com a aplicação são apresentados tutoriais aos Hóspedes.
Rationale: Ajudar Hóspedes principiantes a usar a aplicação.
Originator: Introspecção
Fit Criterion: Serão apresentadas caixas de texto com explicações de como usar as funcionalidades da aplicação.
Priority: Won't
History: 30/03/2023

Tabela 67: Requisito não funcional 6.

6.2.3 Requisitos de Desempenho

Requirement: #7 **Type**:
Description: O sistema deverá ser preciso na realização de transações.
Rationale: Evitar o tratamento de valores monetários incorretos.
Originator: Introspecção
Fit Criterion: As transações são efetuadas sem erros.
Priority: Must
History: 30/03/2023

Tabela 68: Requisito não funcional 7.

Requirement: #8 **Type**:
Description: Os cálculos relativos aos valores das reservas deverão ser efetuados corretamente.
Rationale: Evitar transferências monetárias de valores incorretos.
Originator: Introspecção
Fit Criterion: Todas as reservas apresentam o valor total de pagamento correto.
Priority: Must
History: 30/03/2023

Tabela 69: Requisito não funcional 8.

Requirement: #9 **Type**:
Description: O sistema deverá ter capacidade para armazenar grandes volumes de dados.
Rationale: O sistema terá de manusear a informação de um número elevado de quartos, hóspedes e funcionários.
Originator: Introspecção
Fit Criterion: O sistema armazena, no mínimo, 30 quartos e 20 utilizadores.
Priority: Must
History: 30/03/2023

Tabela 70: Requisito não funcional 9.

Requirement: #10 **Type**:
Description: O sistema deverá estar ativo cerca de 98% do tempo.
Rationale: Maximização da disponibilização do site, dos seus lucros e da sua utilização.
Originator: Introspecção
Fit Criterion: Disponibilidade do serviço para o utilizador.
Priority: Must
History: 30/03/2023

Tabela 71: Requisito não funcional 10.

Requirement: #11 **Type**:
Description: O sistema deverá ser tolerante a falhas.
Rationale: Permitir a utilização do sistema mesmo quando alguns recursos falham.
Originator: Introspecção
Fit Criterion: Tolerância a falhas.
Priority: Should
History: 30/03/2023

Tabela 72: Requisito não funcional 11.

Requirement: #12 **Type**:
Description: O utilizador deverá ter acesso rápido às funcionalidades do sistema.
Rationale: Manter a simplicidade de acesso às funcionalidades da aplicação.
Originator: Introspecção
Fit Criterion: O utilizador deve, no máximo de 3 cliques, chegar às funcionalidades principais como efetuar reservas e registo.
Priority: Should
History: 30/03/2023

Tabela 73: Requisito não funcional 12.

Requirement: #13 **Type**:
Description: Qualquer pedido ao website deve ser respondido em menos de 1 segundo, em 90% dos casos. Os restantes casos, devem ser respondidos em 1,5 segundos.
Rationale: O tempo de espera entre as conexões deve ser o mais breve possível.
Originator: Introspecção
Fit Criterion: Tempo de resposta do serviço.
Priority: Should
History: 30/03/2023

Tabela 74: Requisito não funcional 13.

Requirement: #14 **Type**:
Description: O sistema deve ser capaz de suportar um grande nível de operações sem restrições.
Rationale: Assegurar a resiliência da aplicação.
Originator: Introspecção
Fit Criterion: Suporte para um grande número de operações.
Priority: Should
History: 30/03/2023

Tabela 75: Requisito não funcional 14.

Requirement: #15 **Type**:
Description: O tempo de atualização do serviço deverá ser feito de minuto a minuto.
Rationale: Manter a informação o mais atualizada possível contribui para uma interação agradável.
Originator: Introspecção
Fit Criterion: Atualização da informação.
Priority: Should
History: 30/03/2023

Tabela 76: Requisito não funcional 15.

6.2.4 Requisitos de Escalabilidade

Requirement: # **Type**: 12
Description: A solução deve ser capaz de suportar um crescimento anual de 7% de novos utilizadores.
Rationale: Espera-se que o serviço, com o passar do tempo, seja mais utilizado e mais popular.
Originator: Introspecção
Fit Criterion: Evolução do serviço.
Priority: Should
History: 30/03/2023

Tabela 77: Requisito não funcional .

Requirement: #16 **Type**:
Description: A solução deve ser capaz de suportar o decréscimo de utilizadores em períodos noturnos.
Rationale: Espera-se que o serviço seja mais utilizado durante o dia.
Originator: Introspecção
Fit Criterion: Utilização dos recursos em períodos noturnos.
Priority: Should
History: 30/03/2023

Tabela 78: Requisito não funcional 16.

6.2.5 Requisitos Operacionais

Requirement: #17 **Type**:
Description: A aplicação deverá ser capaz de ser executada em qualquer *browser*/plataforma *web*.
Rationale: As preferências dos utilizadores não impedem o uso da aplicação.
Originator: Introspecção
Fit Criterion: Adaptabilidade da solução para os diversos browsers.
Priority: Should
History: 30/03/2023

Tabela 79: Requisito não funcional 17.

Requirement: #18 **Type**:
Description: A aplicação deve estar disponível tanto como um *website*, como uma aplicação *mobile*.
Rationale: Dinamização da utilização do serviço para todos os envolventes.
Originator: Introspecção
Fit Criterion: Portabilidade da aplicação.
Priority: Won't
History: 30/03/2023

Tabela 80: Requisito não funcional 18.

6.2.6 Requisitos de Manutenção e Suporte

Requirement: #19 **Type**:
Description: O sistema deverá manter informação de estado.
Rationale: Permitir ao utilizador fazer refresh ou fechar a página sem perder os seus dados.
Originator: Introspecção
Fit Criterion: Manter a informação de estado.
Priority: Must
History: 30/03/2023

Tabela 81: Requisito não funcional 19.

Requirement: #20 **Type**:
Description: A manutenção deverá ser feita no horário de menor tráfego.
Rationale: Minimizar o transtorno aos utilizadores do sistema.
Originator: Introspecção
Fit Criterion: Horário de Manutenção.
Priority: Could
History: 30/03/2023

Tabela 82: Requisito não funcional 20.

Requirement: #21 **Type**:
Description: Todo o código desenvolvido deve ser devidamente comentado.
Rationale: Facilitar o entendimento e funcionamento do código por todos os elementos responsáveis pelo desenvolvimento de software.
Originator: Introspecção
Fit Criterion: Inserção de comentários no código.
Priority: Could
History: 30/03/2023

Tabela 83: Requisito não funcional 21.

Requirement: #22 **Type**:
Description: A inclusão de uma nova funcionalidade deve ser feita durante a noite no horário local.
Rationale: Evitar a sobrecarga do sistema e a introdução do erros no serviço.
Originator: Introspecção
Fit Criterion: Horário de introdução de funcionalidades.
Priority: Won't
History: 30/03/2023

Tabela 84: Requisito não funcional 22.

6.2.7 Requisitos de Segurança

Requirement: #23 **Type**:
Description: O acesso à informação da aplicação deverá ser restringido dependendo dos privilégios do utilizador.
Rationale: Garantir que não ocorre vazamento de informação nem acesso não autorizado da mesma.
Originator: Introspecção
Fit Criterion: É apresentada a informação a cada utilizador de acordo com os seus privilégios na aplicação.
Priority: Must
History: 30/03/2023

Tabela 85: Requisito não funcional 23.

Requirement: #24 **Type**:
Description: O sistema deve prevenir a introdução de dados incorretos, através da validação dos inputs, por exemplo dados de autenticação e formulários de reservas.
Rationale: Validação e segurança do sistema.
Originator: Introspecção
Fit Criterion: O utilizador deve errar no máximo 3 vezes ao preencher o mesmo input.
Priority: Must
History: 30/03/2023

Tabela 86: Requisito não funcional 24.

Requirement: #25 **Type**:
Description: A comunicação entre o servidor de dados do sistema e os clientes deve ser criptografada.
Rationale: Segurança dos dados.
Originator: Introspecção
Fit Criterion: Assegurar a segurança dos dados.
Priority: Must
History: 30/03/2023

Tabela 87: Requisito não funcional 25.

Requirement: #26 **Type**:
Description: Deve ser mantida uma backup de todos os dados relativos ao sistema.
Rationale: Evitar a perda de dados relevantes para o serviço.
Originator: Introspecção
Fit Criterion: Segurança dos dados.
Priority: Must
History: 30/03/2023

Tabela 88: Requisito não funcional 26.

Requirement: #27 **Type**:
Description: O sistema deverá armazenar todas as passwords com o recurso a hash.
Rationale: Confiabilidade dos utilizadores no sistema e segurança do mesmo.
Originator: Introspecção
Fit Criterion: Segurança das passwords.
Priority: Must
History: 30/03/2023

Tabela 89: Requisito não funcional 27.

6.2.8 Requisitos Culturais e Políticos

Requirement: #28 **Type**:
Description: A aplicação deve estar disponível em múltiplas linguagens.
Rationale: Dinimização da utilização da aplicação.
Originator: Introspecção
Fit Criterion: Internacionalização da aplicação.
Priority: Should
History: 30/03/2023

Tabela 90: Requisito não funcional 28.

6.2.9 Requisitos Legais

Requirement: #29 **Type**:
Description: Uma reserva só deve ser validada e efetuada com sucesso, caso o utilizador tenha mais de 18 anos.
Rationale: Pessoas com menos de 18 anos, não têm legalidade para fazer uma reserva.
Originator: Introspecção
Fit Criterion:
Priority: Must
History: 30/03/2023

Tabela 91: Requisito não funcional 29.

Requirement: #30 **Type**:
Description: Deverão ser cumpridas todas as leis/normas legais relativas à área em que a aplicação se encontra disponível.
Rationale: As normas de funcionamento têm de ser respeitadas.
Originator: Introspecção
Fit Criterion: Termos legais existentes que devem ser respeitados.
Priority: Must
History: 30/03/2023

Tabela 92: Requisito não funcional 30.

7 Modelos de Tarefas

Para o trabalho prático foram desenvolvidos apenas três modelos de tarefas fundamentais: "Fazer uma reserva", "Efetuar Check In"e "Efetuar Check Out". Estes modelos têm como objetivo ilustrar a sequência de ações que o sistema e o utilizador devem realizar para executar uma determinada tarefa. Além disso, estes também fornecem informações sobre a ordem e as diferentes abordagens que podem ser adotadas. Todos os modelos aqui presentes foram também colocados nos Anexos de modo a facilitar a sua leitura.

7.1 Modelo 1: Hóspede autenticado fazer uma reserva

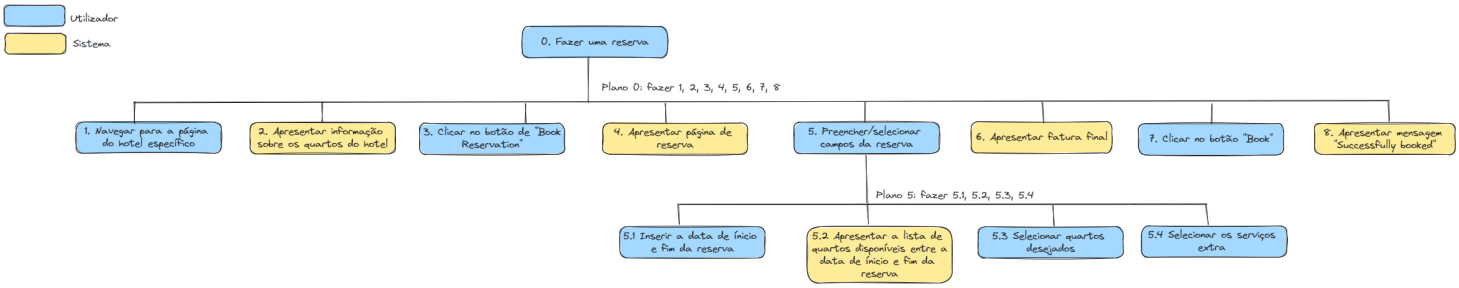


Figura 3: Diagrama de Tarefa 1 - Hóspede autenticado fazer uma reserva.

7.2 Modelo 2: Efetuar Check In

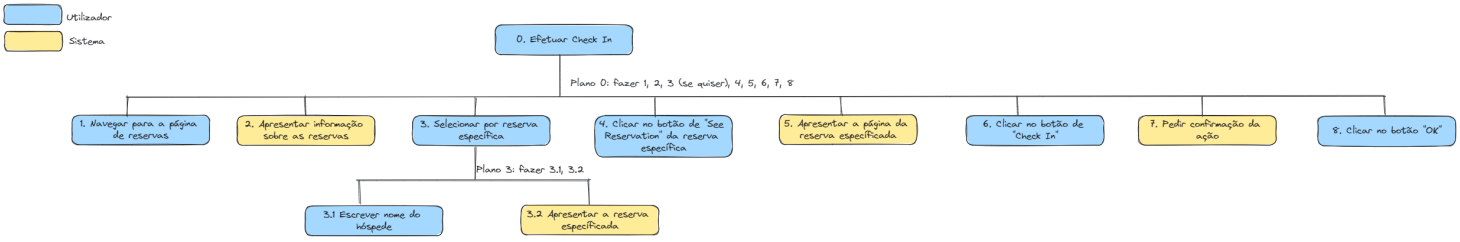


Figura 4: Diagrama de Tarefa 2 - Efetuar Check In.

7.3 Modelo 3: Efetuar Check Out

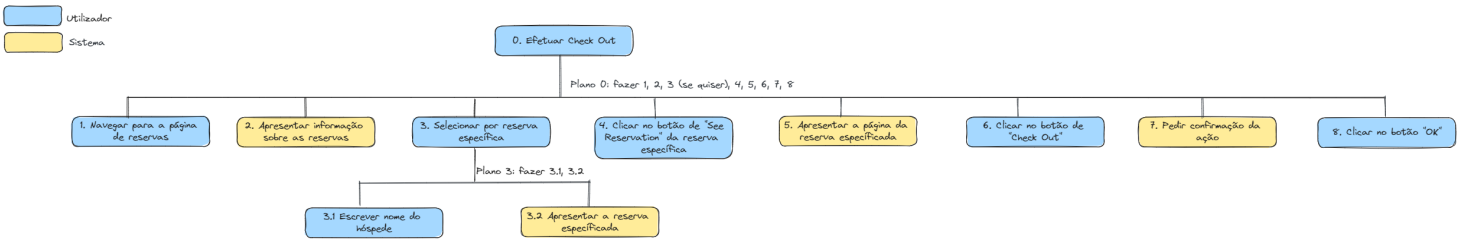


Figura 5: Diagrama de Tarefa 3 - Efetuar Check Out.

7.4 Modelo 4: Hóspede autenticado cancelar uma reserva.

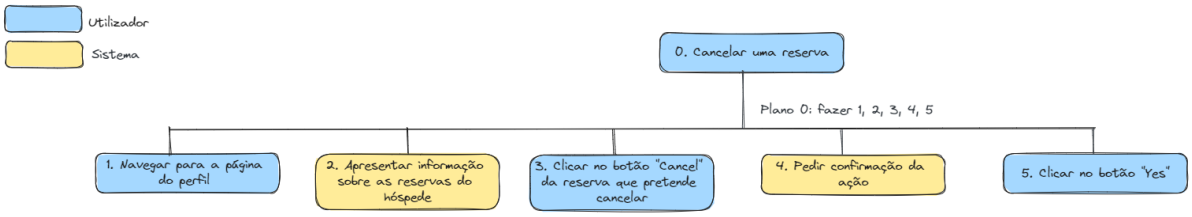


Figura 6: Diagrama de Tarefa 4 - Hóspede autenticado cancelar uma reserva.

8 Protótipos de Baixa Fidelidade

A prototipagem permite economizar tempo e recursos, além de proporcionar a exploração das diversas alternativas para o design da interface. Desta forma, e com o intuito de verificar a usabilidade e detetar possíveis problemas, foram criados *Mockups* das páginas da interface esperadas na implementação, utilizando para isso a ferramenta **Figma**.

8.1 Página Home

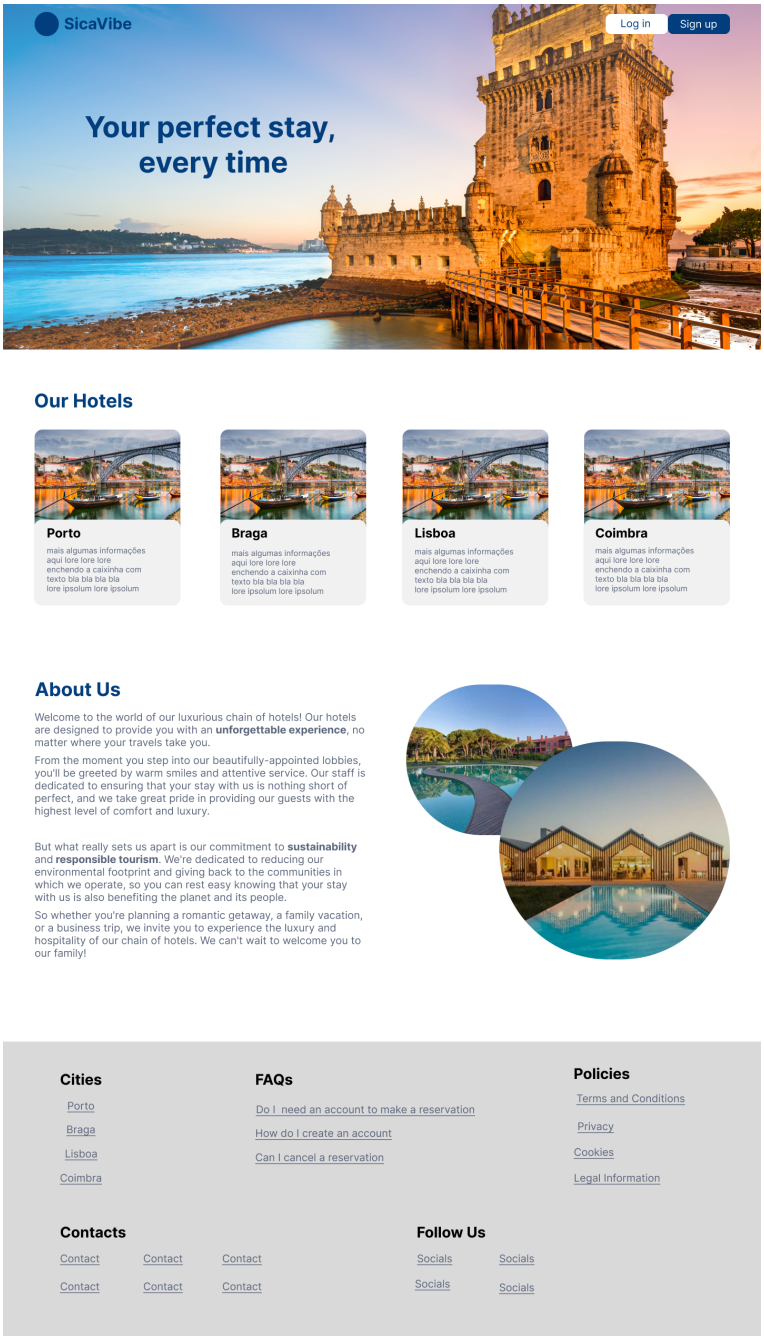


Figura 7: *Mockup* da Página Home.

8.2 Página de Login

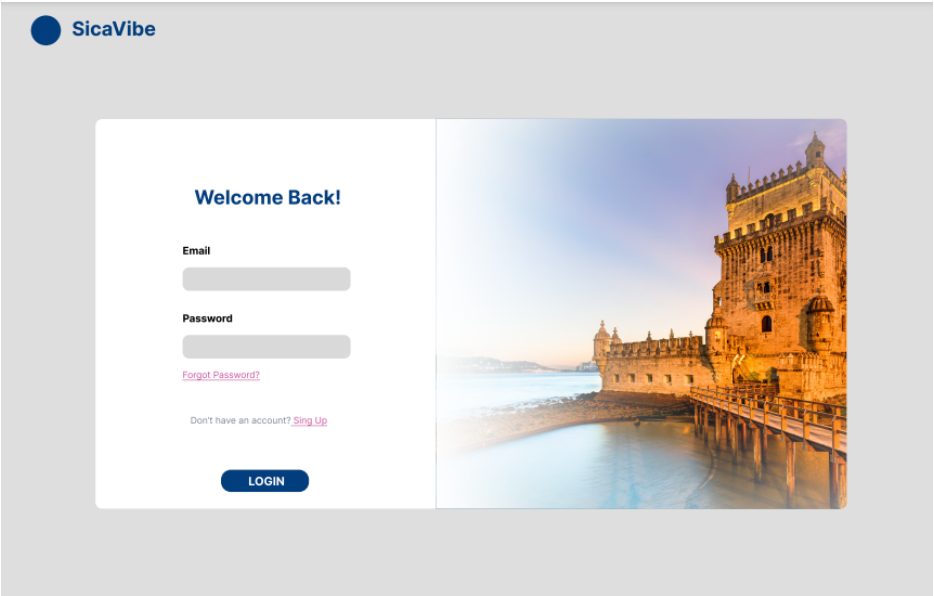


Figura 8: *Mockup* da Página de Login.

8.3 Página de Registar

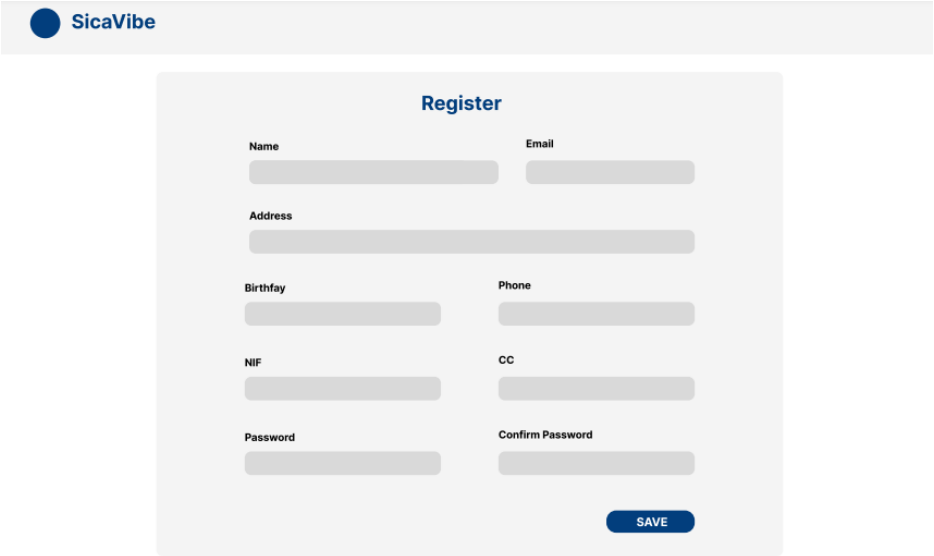


Figura 9: *Mockup* da Página de Registar.

8.4 Página de um Hotel

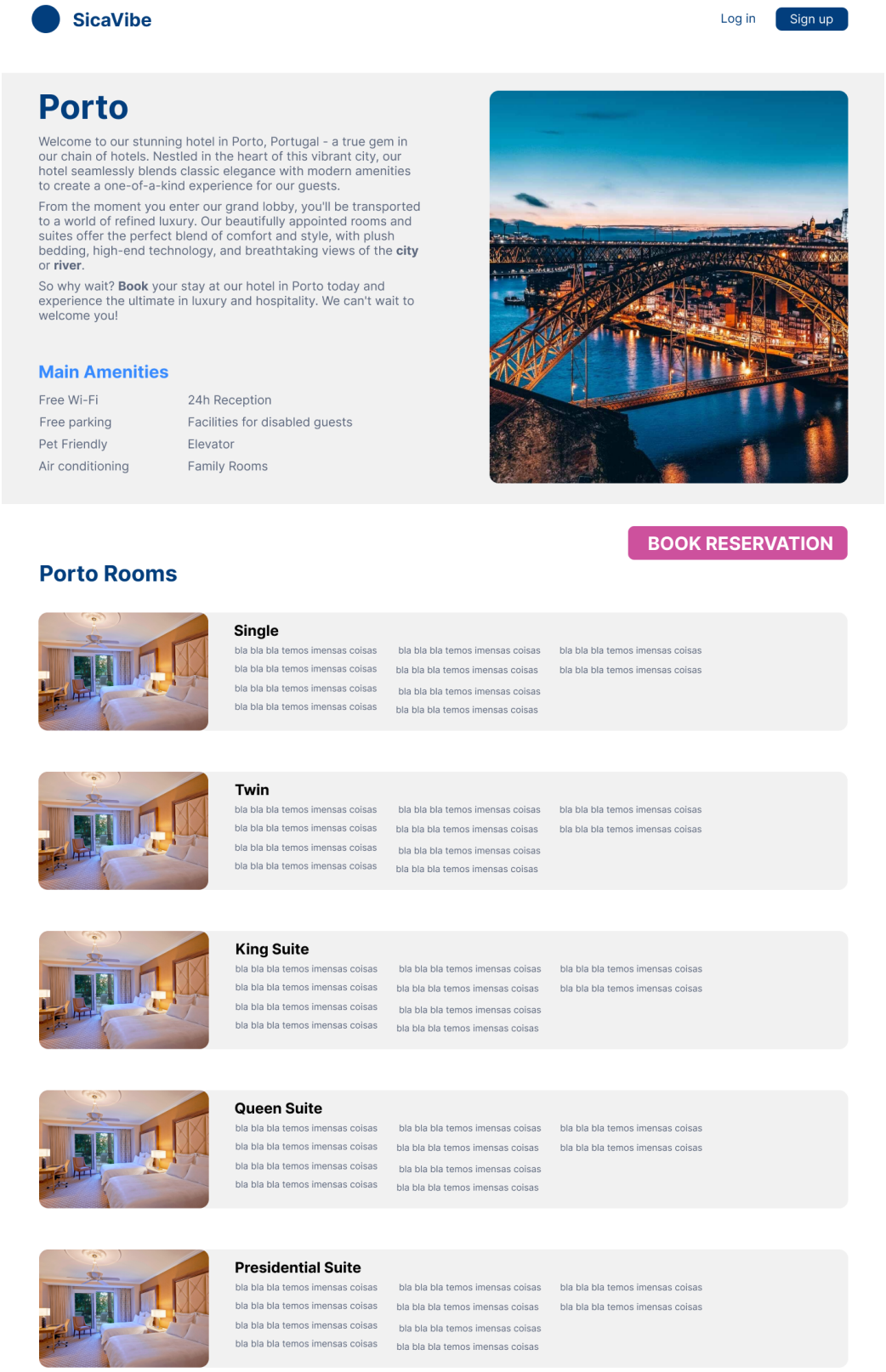


Figura 10: *Mockup* da Página de um Hotel.

8.5 Página de Perfil de um Utilizador

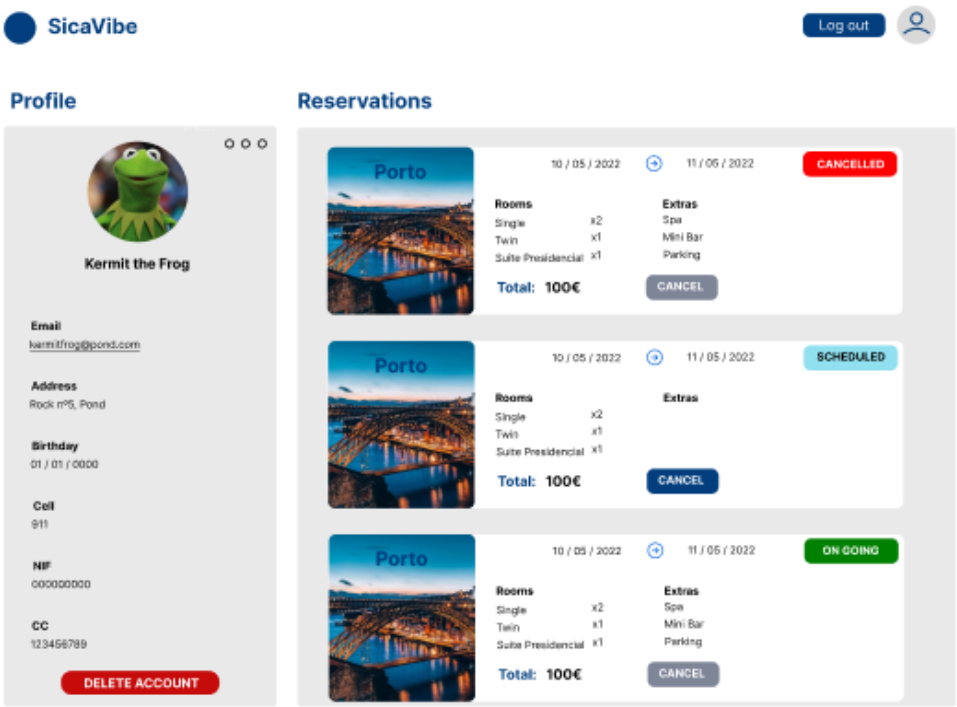


Figura 11: *Mockup* da Página do Perfil de um Utilizador I.

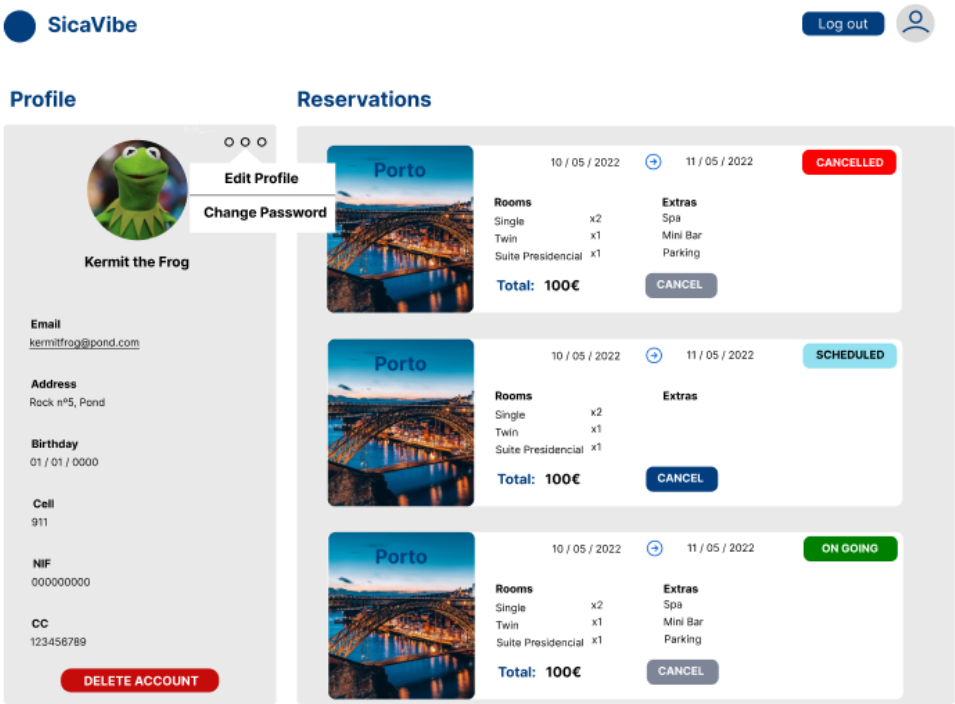


Figura 12: *Mockup* da Página do Perfil de um Utilizador II.

8.6 Página de Editar Perfil de um Utilizador

Figura 13: *Mockup* da Página do Editar Perfil de um Utilizador.

8.7 Página de Alterar Palavra-passe de um Utilizador

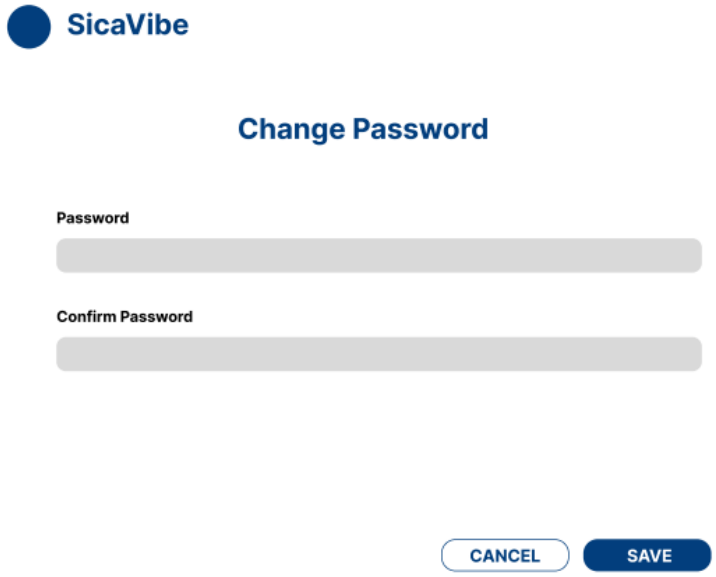


Figura 14: *Mockup* da Página de Alterar Palavra-passe Perfil de um Utilizador.

8.8 Página de um Funcionário

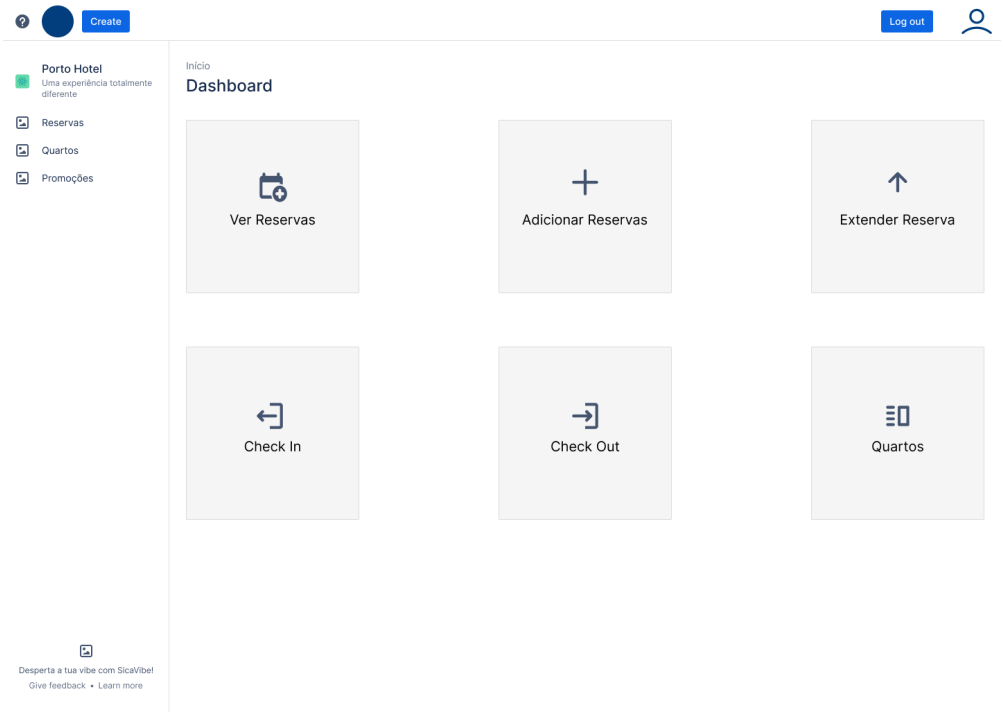


Figura 15: *Mockup* da Página de um Funcionário.

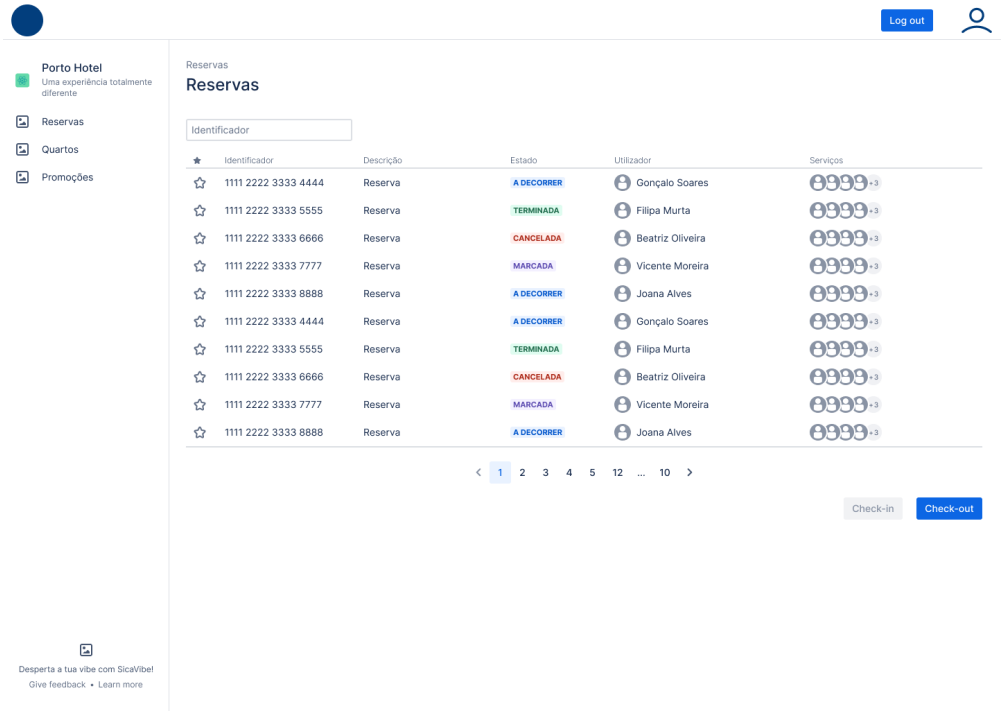


Figura 16: *Mockup* da Página de um Funcionário II (Reservas).

8.9 Página de um Administrador

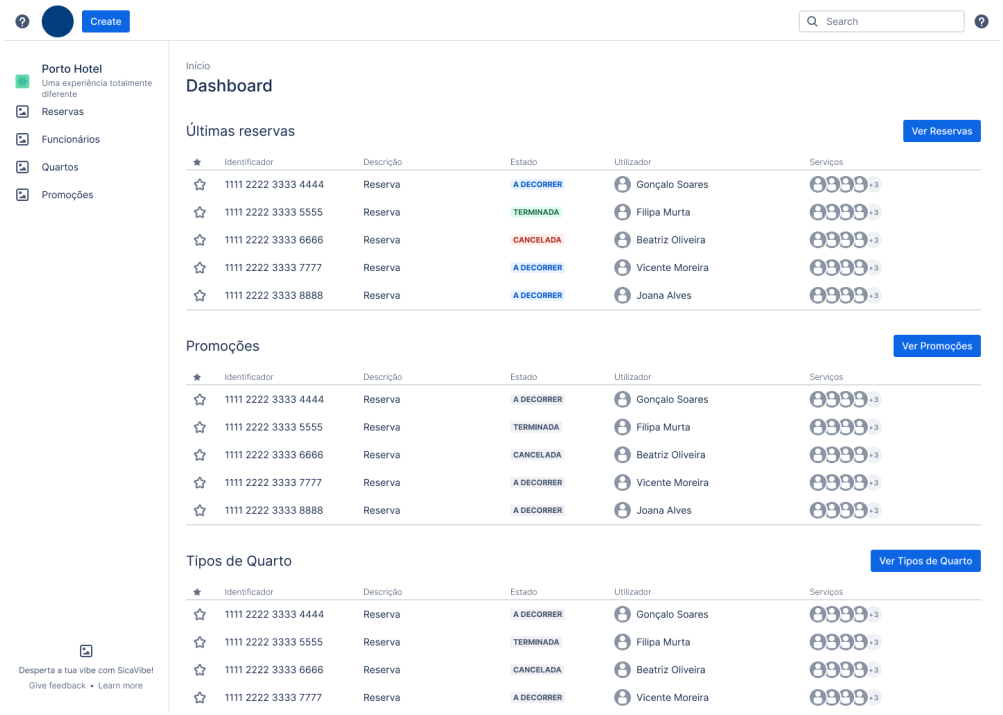


Figura 17: *Mockup* da Página de um Administrador I.

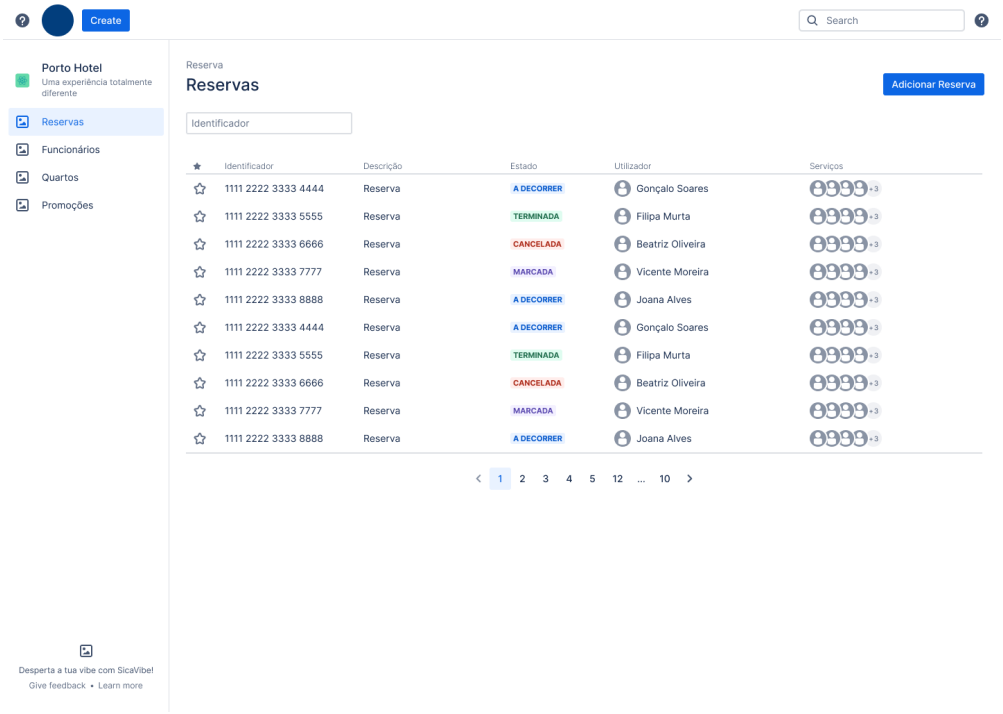


Figura 18: *Mockup* da Página de um Administrador II (Reservas).

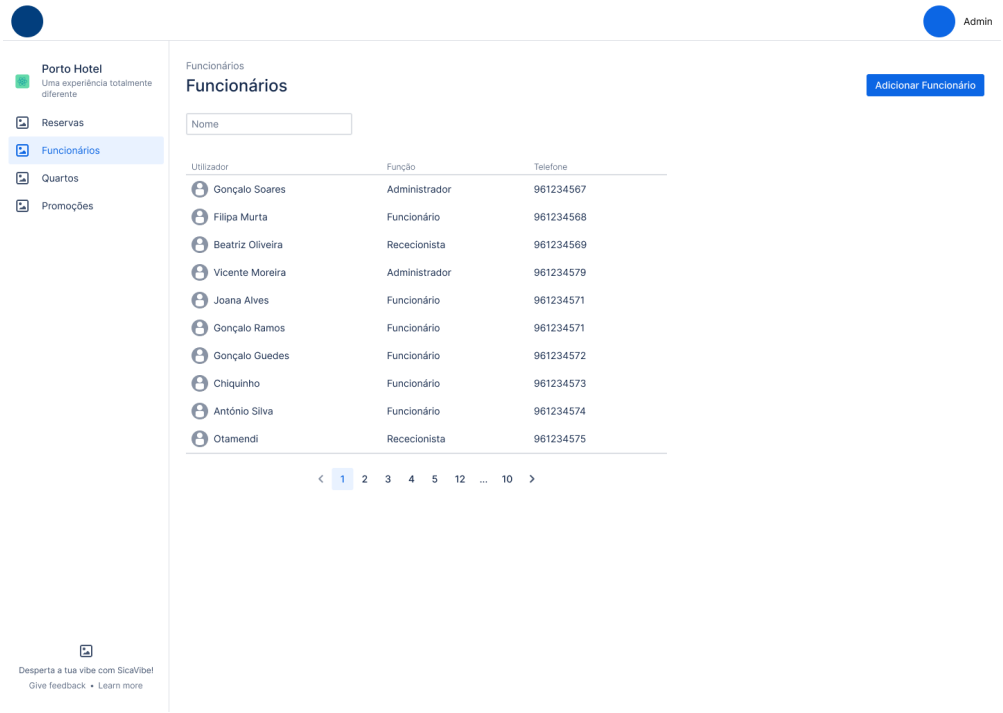


Figura 19: *Mockup* da Página de um Administrador III (Funcionários).

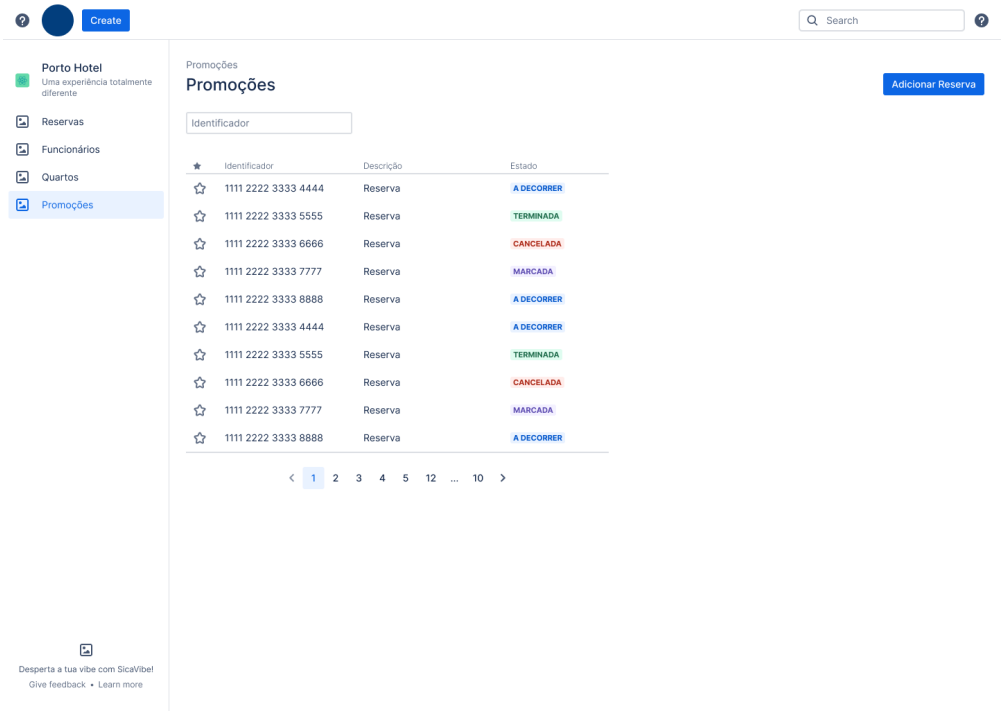


Figura 20: *Mockup* da Página de um Administrador IV (Promoções).

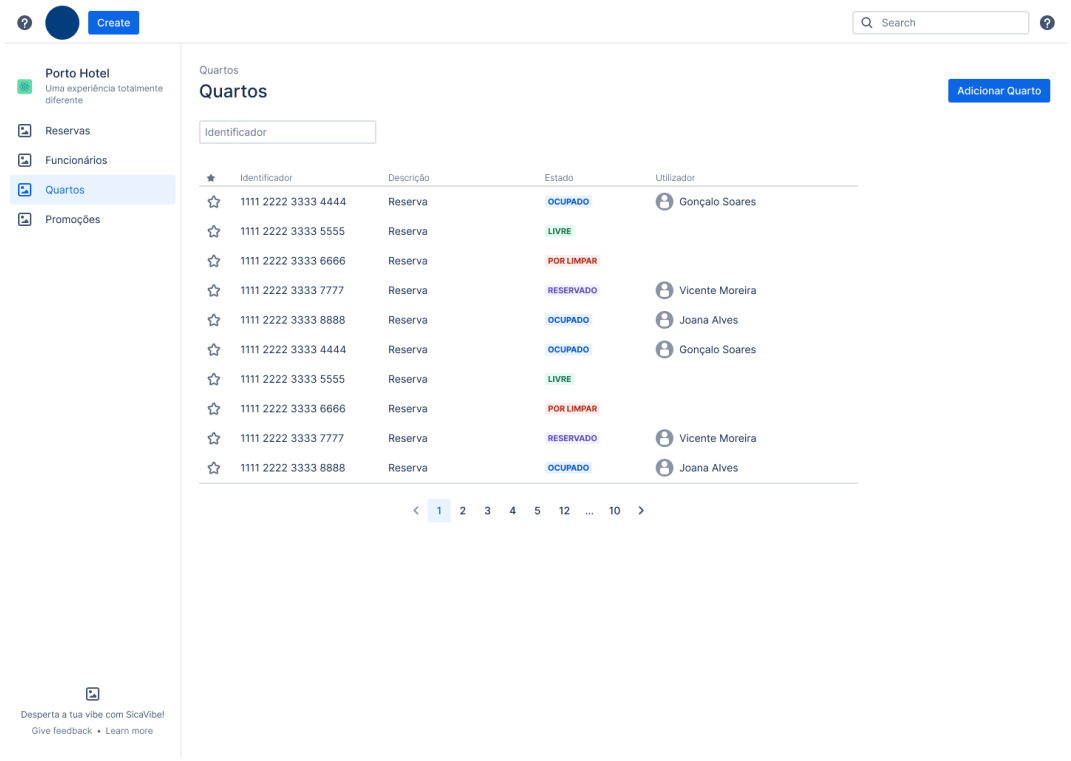


Figura 21: *Mockup* da Página de um Administrador V (Quartos).

8.10 Página de uma Reserva

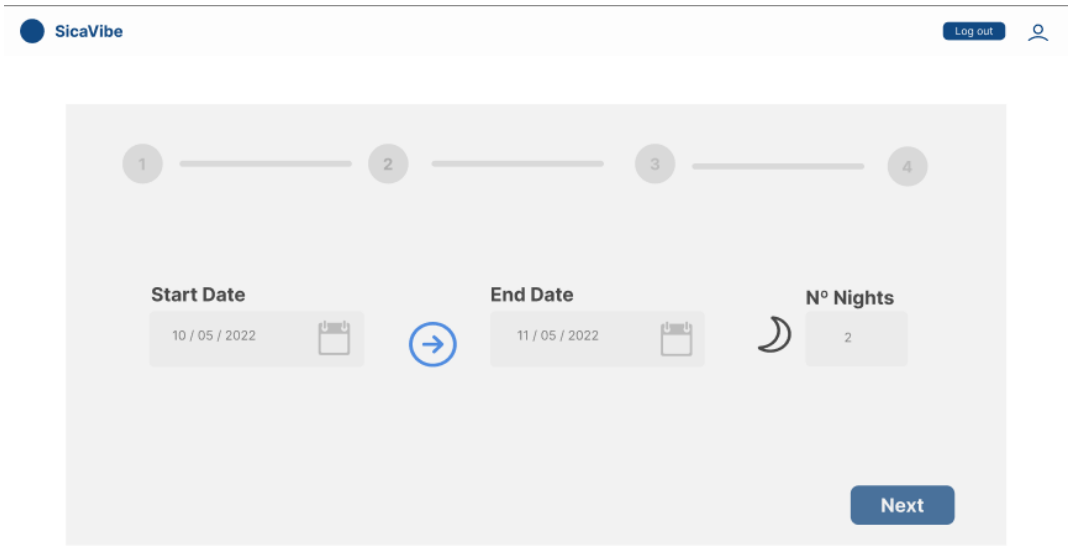


Figura 22: *Mockup* da Página de Reserva I.

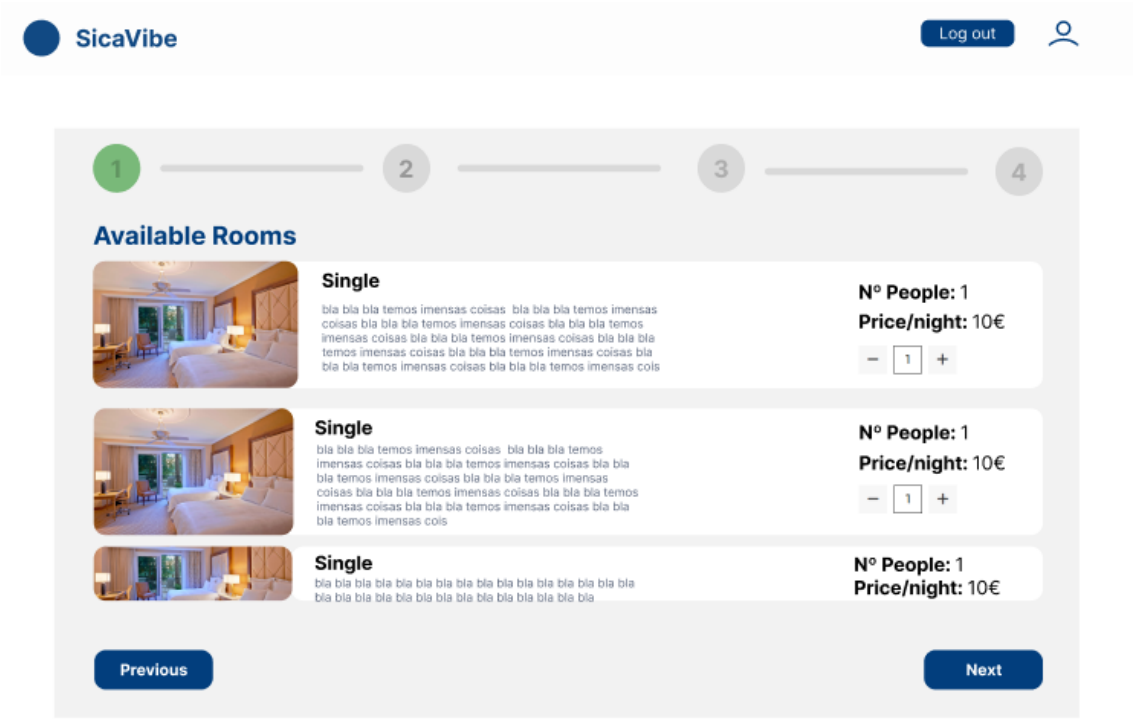


Figura 23: *Mockup* da Página de Reserva II.

Figura 24: *Mockup* da Página de Reserva III.

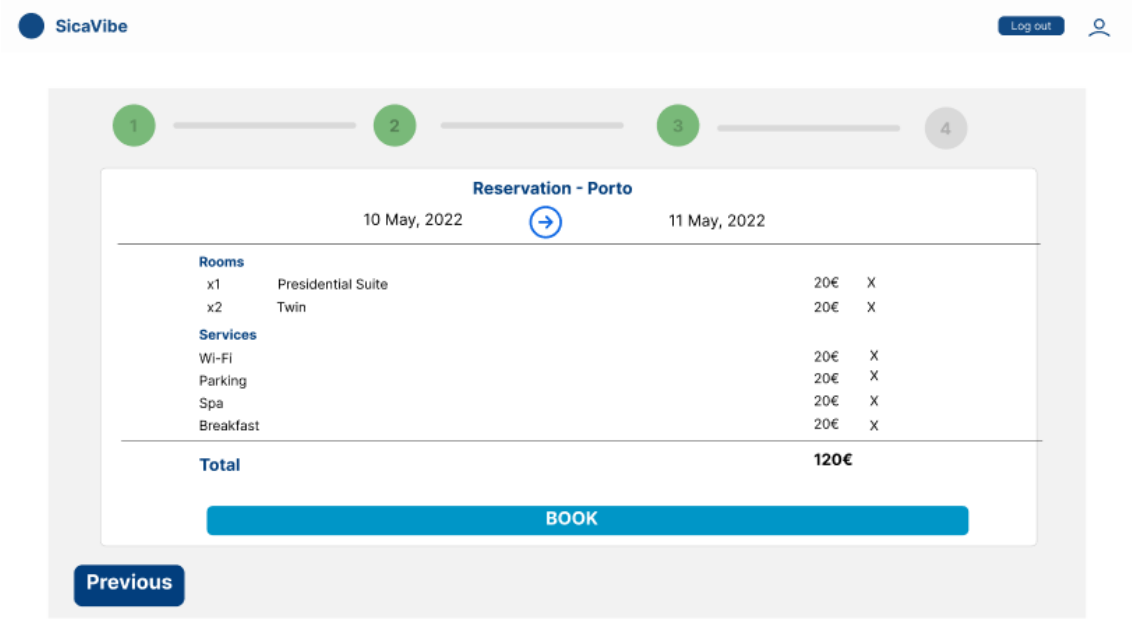


Figura 25: *Mockup* da Página de Reserva IV.

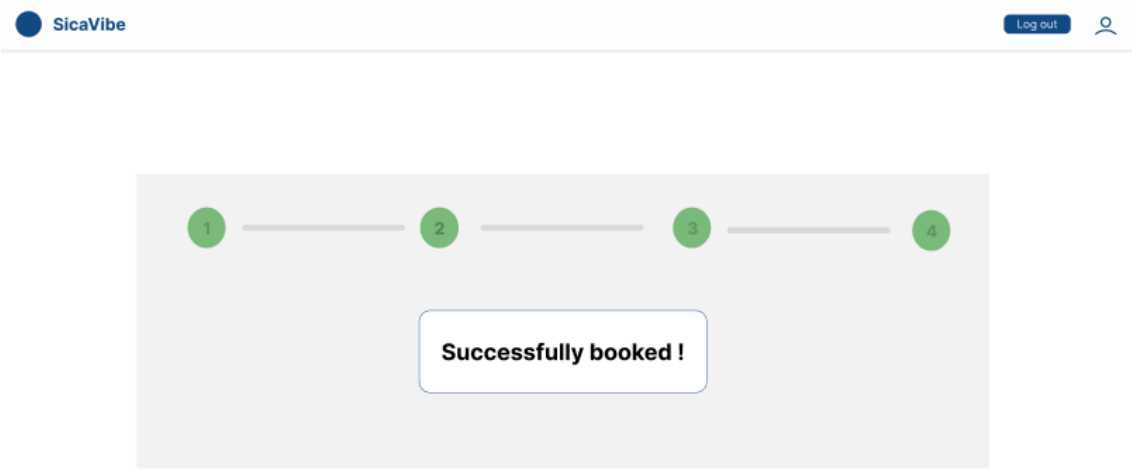


Figura 26: *Mockup* da Página de Reserva V.

9 Estratégia da Solução

Nesta secção, discutimos a solução planeada e implementada pelo grupo de forma a cumprir os vários requisitos já delimitados. Para além disso, é possível encontrar as justificações relativas às várias decisões efetuadas, detalhando as vantagens e as desvantagens de cada solução utilizada.

9.1 Padrão Arquitetural e Tecnologias

Para a arquitetura do sistema, a equipa decidiu criar uma divisão forte entre as componentes da interface gráfica (***frontend***) e as restantes componentes relativas à lógica de negócio e ao modelo de dados do sistema, sendo que estas entidades comunicam através de chamadas HTTP entre os vários *endpoints* disponibilizados no servidor aplicacional (***backend***). Acreditamos que esta decisão favorece o sistema visto que facilita o desenvolvimento deste pela modularização das suas componentes e permitindo o desenvolvimento independente de cada uma das componentes do sistema.

Para o desenvolvimento do servidor aplicacional, a equipa decidiu utilizar as seguintes tecnologias e *frameworks* com o objetivo de padronizar a aplicação desenvolvida, assim como tirar proveito das vantagens de eficiência e facilidade de uso que estas tecnologias fornecem. O modo de funcionamento destas tecnologias será explicado em detalhe das nas secções seguintes.

- **Base de Dados: *MySQL* -** Base de Dados relacional popular, com várias ferramentas e suporte de desenvolvimento.
- **ORM: *Hibernate* -** Baseado na experiência obtida nas aulas práticas, assim como as vantagens de facilidade de uso e comunicação com a base de dados.
- **Web Framework: *Spring Boot* -** *Framework* base do servidor, responsável pelo controlo de dependências e gestão dos *controllers* que permitem a criação dos vários *endpoints*.

No desenvolvimento da camada de interface da aplicação, a equipa utilizou a *framework* ***Vue.js*** em junção com outras bibliotecas desta como o *Pinia* e o *Vuetify*. Esta decisão foi feita devido à capacidade nativa do *Vue* de criar interfaces altamente interativas e *responsive* às limitações de tamanho de ecrã.

De seguida, apresentamos um pequeno diagrama representativo do sistema como um todo, com a divisão entre camadas e sub-camadas e a tecnologia utilizada em cada uma destas.

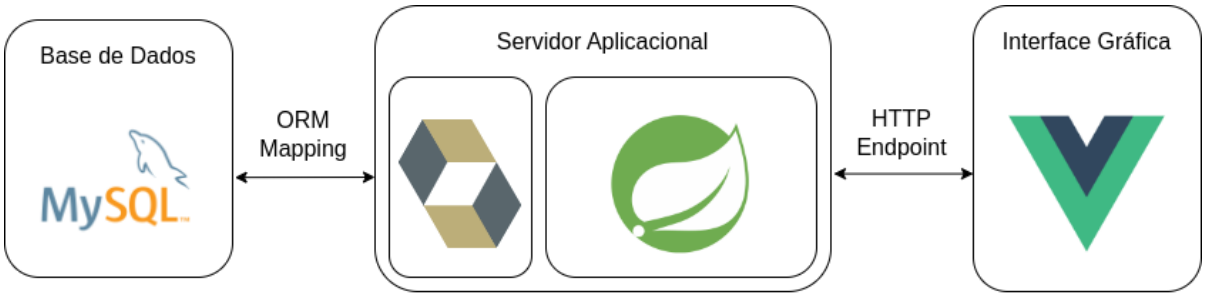


Figura 27: Arquitetura Geral.

9.2 *Servidor Aplicacional*

Para o desenvolvimento do servidor aplicacional, desenvolvemos o mesmo recorrendo à linguagem de programação *Java*. Desta forma, tínhamos à nossa disposição várias ferramentas e tecnologias que agilizavam este processo de desenvolvimento.

Resumidamente, este será responsável pela gestão de todos os dados do sistema, pelo controlo da lógica entre estes e pela disponibilização dos vários *endpoints* onde serão efetuadas as chamadas ao sistema. Este pode ser "divido"nos seus subconjuntos, sendo estes: *Hibernate* e *Spring Boot*.

9.2.1 Hibernate ORM

As ferramentas de ***Object-Relational Mapping***, *ORM*, são extensamente úteis para a programação do servidor aplicacional do sistema, visto que estas retiram carga de desenvolvimento no que toca às funcionalidades de comunicação e de troca de informação com a Base de Dados. Já que esta ferramenta é capaz de efetuar a ligação com Base de Dados, pesquisar e recolher dados desta e efetuar transações e *commits* de forma transparente para o programador, tendo a grande vantagem que este poderá trabalhar com as várias classes num formato **POJO** definido.

Para este acesso transparente, é necessário recorrer às classes ***Data Access Object***, *DAO*, dos vários objetos com o intuito de realizar a pesquisa de uma instância de um objeto específico ou de uma lista destes. Depois de recolhidos, estes podem ser manipulados como ***POJOs***, sendo que qualquer alteração a estes podem ser gravados na Base de Dados através, novamente, do ***Data Access Object*** referente a essa classe.

9.2.2 Sping Boot Framework

O *Spring Boot* é uma *framework opensource* para servidores aplicacionais **Java** e permite a criação de aplicações *Web* com base na plataforma **Java EE**, sendo esta uma adição ao modelo de **Enterprise JavaBeans (EJB)**. Esta é também conhecida pelo vasto suporte a diversas dependências e a possibilidade de adição de módulos e de componentes a este, tendo esta *framework* módulos capazes de fazer a gestão da segurança da aplicação e da comunicação com Bases de Dados.

No nosso projeto, utilizamos esta *framework* pela sua capacidade de gerir e desenvolver uma ***REST API*** através da criação de vários ***Rest Controllers*** responsáveis por atender pedidos **HTTP** em vários *endpoints* criados. Já estes são responsáveis pelo *parsing* dos *headers* e *bodys* dos pedidos *HTTP*, assim como, pela realização das pesquisas ou das mudanças necessárias ao sistema, quer pelo uso de *DAO's* como mencionado anteriormente, ou pela execução de funções auxiliares destas funções. Estes *controllers* também são responsáveis pela gestão dos possíveis erros, mais precisamente, são responsáveis por reencaminhar estes para a função de tratamento de erro, que irá gerar uma página com a informação acerca deste.

9.2.2.1 *OpenAPI & Swagger*

Um dos módulos extra adicionados a esta *framework* é a ***OpenAPI*** com a integração **Swagger**, sendo que este módulo trabalha com os **Rest Controllers** definidos pela equipa de desenvolvimento e gera a documentação de forma automática sobre os vários **endpoints** presentes no servidor aplicacional. Este módulo também suporta o uso de anotações nas funções de *endpoints* de forma a que o programador possa detalhar os *endpoints* explicitamente, adicionar os parâmetros

necessários e os formatos de resposta e, ainda, organizar estes endpoints em categorias na página gerada.

Para além da geração desta página, foi adicionada uma funcionalidade de redirecionamento automático a esta quando um utilizador acede à **landing page** do servidor aplicacional. Apresentamos aqui uma imagem para exemplificar a documentação gerada, sendo que os vários *endpoints* serão detalhados na sua secção correspondente.

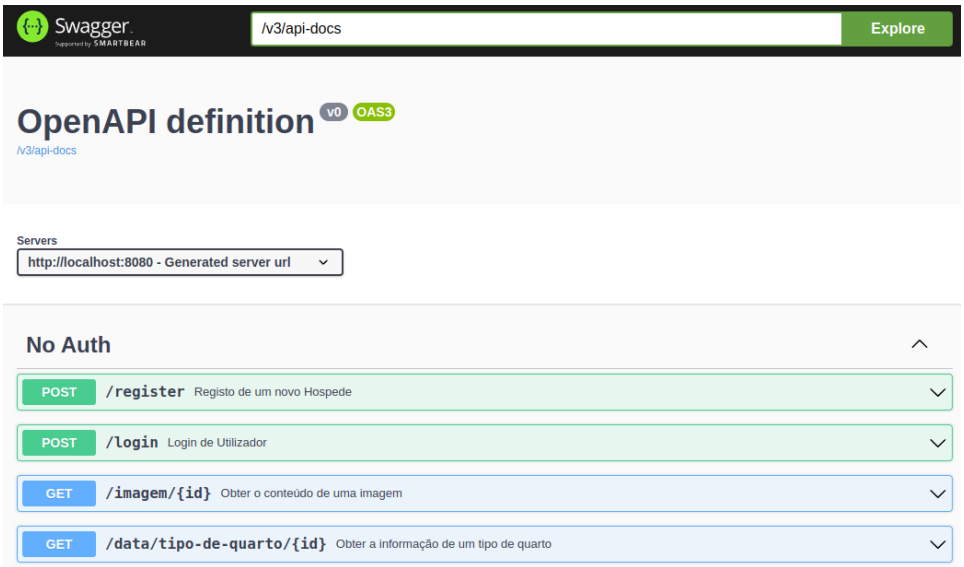


Figura 28: Exemplo *WebSite*.

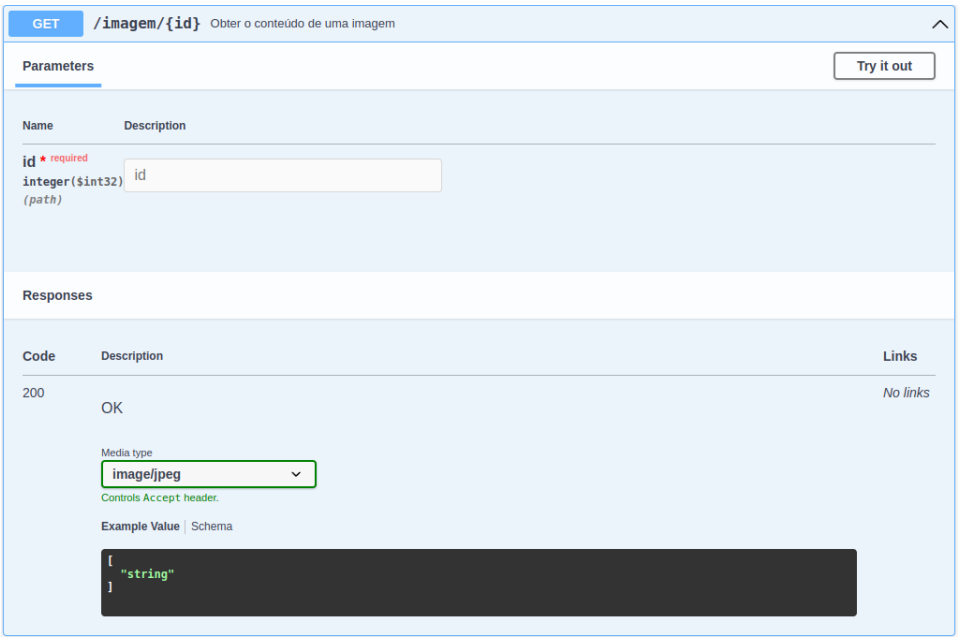


Figura 29: Exemplo Documentação *Endpoint*.

9.2.3 Diagrama de Classes (PIM)

Nesta secção vamos apresentar o diagrama de classes (PIM) desenvolvido de modo a albergar a estrutura do sistema, assim como todas as relações entre as entidades presentes no mesmo. Pelo que apresentamos, de seguida, o diagrama desenvolvido:

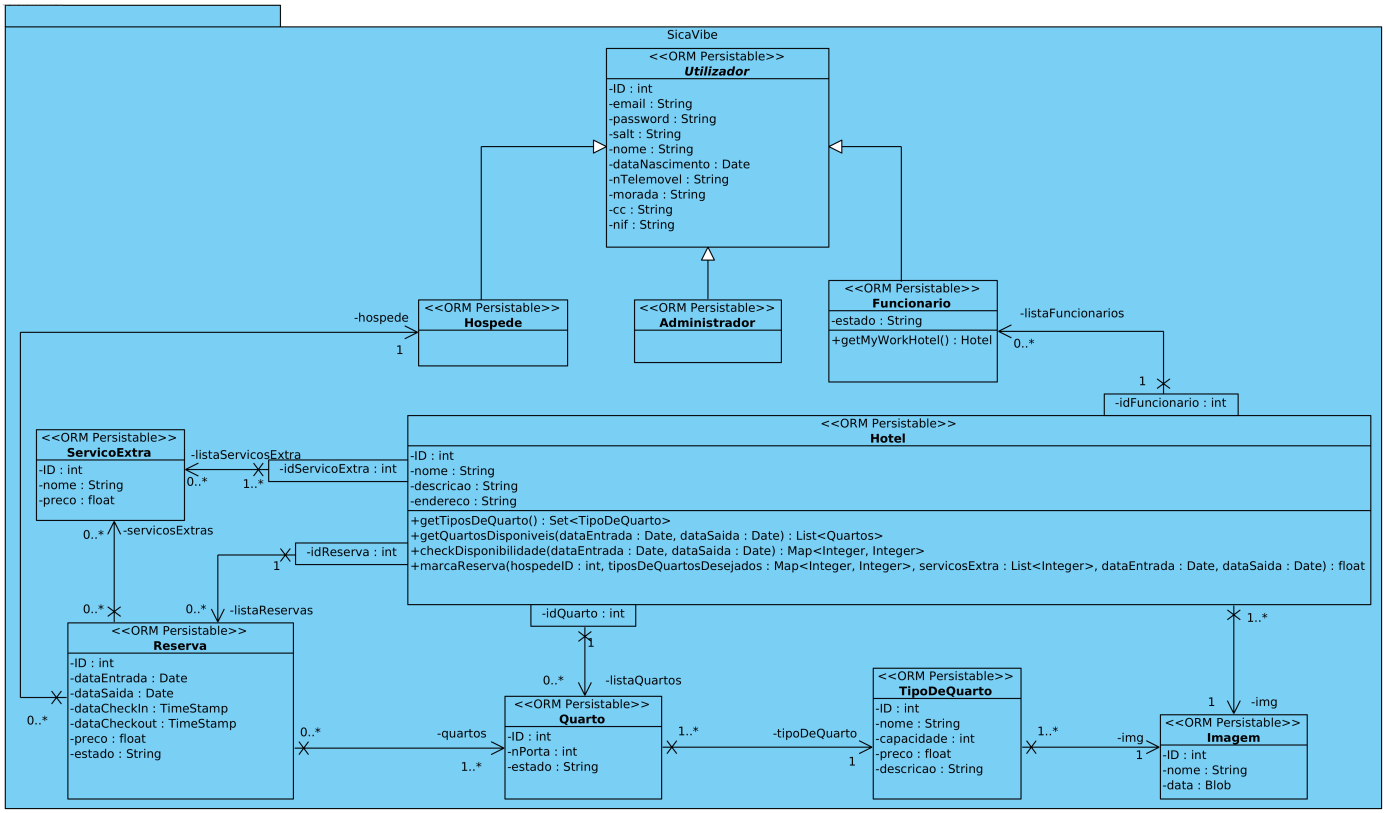


Figura 30: Diagrama de Classes.

9.2.3.1 Geração de Código

Após a criação do diagrama apresentado acima, o próximo passo consistiu na geração do código de persistência através do *Visual Paradigm*. Desta forma, com a sua utilização, geramos o *mapping* a ser efetuado, e todas as classes necessárias para este *mapping*, de forma a que apenas fosse necessário utilizar as classes *Java* como *POJO*.

Deste modo, após a correta geração com base nos diversos tutoriais fornecidos nas aulas práticas, obtivemos um modelo relacional da nossa base de dados, assim como um ficheiro com as instruções SQL para a criação da mesma.

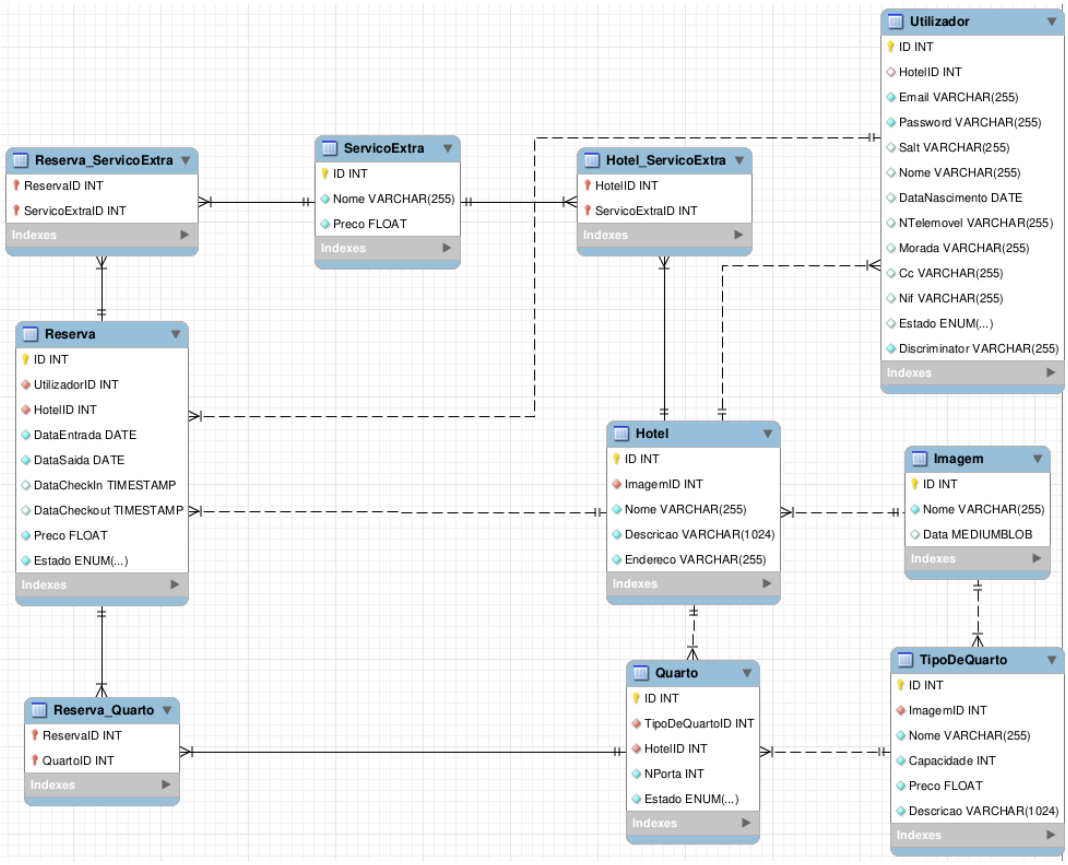


Figura 31: Modelo Relacional.

9.2.4 Listagem de Endpoints Desenvolvidos

Nesta secção, listamos todos os *endpoints* desenvolvidos e presentes no Servidor Aplicacional, seguidos de uma breve descrição do seu papel no sistema. Estes *endpoints* estão divididos por categorias, onde cada uma representa o ator ou o nível de autenticação necessária para efetuar essa ação.

9.2.4.1 No Auth

- **POST: /register -** Registo de um novo Hóspede;
- **POST: /login -** *Login* de Utilizador;
- **GET: /imagem/{id} -** Obter conteúdo de uma Imagem;
- **GET: /data/tipo-de-quarto/{id} -** Obter a informação de um Tipo de Quarto;
- **GET: /data/servico-extra/{id} -** Obter a informação de um Serviço Extra;
- **GET: /data/hotel/{id} -** Obter a informação de um Hotel;
- **GET: /data/hotel/{id}/tipos-de-quartos -** Tipos de Quartos disponíveis num Hotel;
- **GET: /data/tipo-de-quarto/{id}/servicos-extra -** Serviços Extra disponíveis no Hotel;
- **GET: /data/hoteis -** Obter a informação simples de todos os Hotéis;

9.2.4.2 Hospede

- **POST: /hospede/reservation-cancel -** Cancelar uma Reserva;
- **POST: /hospede/make-reservation -** Marcar uma Reserva;
- **POST: /hospede/edit-account -** Editar perfil de um Hóspede;
- **POST: /hospede/delete-account -** Eliminar conta de um Hóspede;
- **GET: /hospede -** Obter informação de um Hóspede;
- **GET: /hospede/reservations -** Obter Reservas de um Hóspede;
- **GET: /hospede/check-availability -** Verificar a disponibilidade de Quartos;

9.2.4.3 Funcionário

- **POST: /funcionario/set-quarto-state -** Alterar o Estado Do Quarto para LIVRE;
- **POST: /funcionario/alter-reservation -** Alterar o Estado da Reserva;
- **GET: /funcionario -** Obter informação de um Funcionário;
- **GET: /funcionario/reservation -** Obter informação de uma Reserva;
- **GET: /funcionario/list-reservations -** Obter lista de Reservas;
- **GET: /funcionario/list-quartos -** Listar Quartos de um Hotel;

9.2.4.4 Administrador

- **POST: /admin/register-func -** Registo de um novo Funcionário;
- **POST: /admin/register-admin -** Registo de um novo Administrador;
- **GET: /admin -** Obter informação de um Administrador;
- **GET: /admin/reservation -** Obter informação de uma Reserva;
- **GET: /admin/get-user-list -** Obter lista de Hóspedes;
- **GET: /admin/get-reservation-lists -** Obter lista de Reservas;
- **GET: /admin/get-quartos-list -** Obter lista de Quartos;

9.3 Interface Gráfica

No que diz respeito à camada de apresentação, tal como já foi referido anteriormente, utilizamos a *framework* ***Vue.js***. Esta camada é completamente independente das outras duas camadas, mais especificamente, da camada da lógica de negócio e da camada de dados, já que para obter todas as informações necessárias, como por exemplos a lista de hotéis existentes, são realizados pedidos *HTTP* aos diversos *endpoints* existentes.

Com o intuito de facilitar o desenvolvimento da interface gráfica e também por necessidades que surgiram ao longo do próprio desenvolvimento, recorremos a algumas bibliotecas, tais como o ***Vuetify***, o ***vue-router***, o ***Pinia***, o ***Axios***, entre outras.

O *Vuetify* corresponde a uma biblioteca *UI* de componentes em *Vue* e como tal, ao longo deste desenvolvimento, tirámos partido de alguns dos seus componentes. No entanto, também existiram situações em que não os utilizamos, já que pelas suas características, propriedades e comportamento que ofereciam, não correspondia ao desejado pelo grupo. O *vue-router* foi utilizado para a criação e definição das diversas rotas da aplicação.

Relativamente aos pedidos *HTTP* aos *endpoints* para obtenção das informações necessárias, estes foram executados com recurso à biblioteca *axios*, que permite definir o *endpoint* a que se pretende realizar o pedido, o método a utilizar, mais especificamente, se trata de um *GET* ou de um *POST* e permite também receber a resposta proveniente do pedido efetuado. Já para a gestão do estado ao longo das diversas páginas da aplicação, utilizamos a biblioteca *Pinia*, uma biblioteca para *Vue* que trata do armazenamento e da estrutura de gestão do estado.

9.3.1 Estrutura da Solução

No sentido de tirar partido do *Vue* e de todas as suas vantagens, seguimos a estruturação de um projeto *Vue* para a solução obtida. Ou seja, fizemos uma divisão em diretorias, em que temos uma diretoria onde é possível encontrar todas os componentes necessários que foram criados, uma diretoria que trata das rotas, outra que contém os *layouts* relativos às interfaces dos diferentes utilizadores, uma outra que contém as páginas criadas e que faz uso dos componentes criados, etc.

9.3.1.1 *Views*

Em relação às páginas criadas, estas podem ser divididas da seguinte forma:

1. As páginas que podem ser acedidas por um Hóspede Não Autenticado, que correspondem à página inicial, à página específica de um Hotel, ao registo e ao *Login*;

2. As páginas que podem ser acedidas por um Hóspede Autenticado, que incluem as páginas de um Hóspede Não Autenticado, mais a página de perfil, por exemplo;

3. As páginas relativas ao Funcionário;

4. As páginas relativas ao Administrador;

9.3.1.2 Componentes

Relativamente aos componentes criados, de alguma forma são componentes bastante específicos para os utilizadores em questão, no sentido, em que por exemplo, no caso da *Navbar*, a referente aos Hóspedes é distinta da dos Funcionários e da dos Administradores. E mesmo dentro destas

próprias categorias, existem mais distinções, ou seja, a *Navbar* relativa a um Hóspede Autenticado oferece funcionalidades diferentes da de uma relativa a um Hóspede Não Autenticado.

9.3.2 Assets

Nesta componente foram guardados todos as imagens ou ícones que são necessários para o funcionamento da aplicação.

9.3.3 Hooks

Nesta diretoria e componente, é possível encontrar todas as funções que interajam com a reatividade da aplicação.

9.3.4 Layouts

Tal como o nome refere, aqui encontram-se os *layouts* utilizados pelos vários tipos de utilizador.

9.3.5 Services

Nesta componente estão todos os serviços necessários para o funcionamento da aplicação. Estes serviços podem ser tanto locais, sendo por exemplo funções de formatação de *strings*, como serviços remotos. Um exemplo dos serviços remotos utilizados são os serviços do *backend*. Também estão aqui incluídos serviços como a descodificação de um *token* JWT, o armazenamento no *localstorage* entre outros.

9.3.6 Stores

Aqui encontra-se a lógicas de gestão de estado de global na aplicação. Neste caso, apenas foi utilizada uma *store* para manter as informações do utilizador.

9.3.7 Types

Finalmente, temos a diretoria *types*, onde se encontram todos os tipos de dados utilizados globalmente pela aplicação.

10 *Deployment*

Para o ***deployment*** da solução desenvolvida, é necessária uma máquina com o sistema operativo *Ubuntu 22.04 LTS* e é recomendado um mínimo de 16GB de memória RAM e 8 CPU's. Tanto a componente de *frontend* como a de *backend* irão operar na mesma máquina.

10.1 Back-end

10.1.1 Requisitos

Para a compilação e execução da aplicação serão necessários os seguintes requisitos de *software* na máquina local.

- *MySQL Server*;
- *JRE* (Ver. 17 ou superior);
- *Maven* (Ver. 3.6 ou superior);

Estes poderão ser instalados através dos seguintes comandos.

```
sudo apt-get update
sudo apt-get install openjdk-17-jre maven mysql-server -y
```

Por último é preciso clonar o projeto desenvolvido via *GitHub* através de:

```
git clone git@github.com:VicShadow/SicaVibe.git
cd SicaVibe
```

10.1.2 Base de Dados

Antes de criar e popular os dados da Base de Dados, é necessário carregar as imagens do Servidor Aplicacional para uma pasta "segura" do MySQL, sendo necessário executar os seguintes comandos:

```
sudo mkdir /var/lib/mysql-files/SicaVibeImgs
sudo cp SicaVibeApp/scripts/imgs/* /var/lib/mysql-files/SicaVibeImgs/
```

De seguida, é preciso gerar o utilizador que o Servidor Aplicacional irá utilizar para aceder à Base de Dados, assim como gerar as várias tabelas da Base de Dados e, opcionalmente, populá-las. Estes três processos podem ser executados via os seguintes comandos:

```
sudo mysql < SicaVibeApp/scripts/DBSicaVibeCreate.ddl
sudo mysql < SicaVibeApp/scripts/DBUserCreate.sql
sudo mysql < SicaVibeApp/scripts/DBPopulate.sql
```

10.1.3 Servidor Aplicacional

Para o Servidor Aplicacional, é necessário navegar para a diretoria **SicaVibeApp** e compilar o projeto via *Maven*:

```
cd SicaVibeApp
mvn clean package
```

Depois de compilado, o ficheiro executado final poderá ser encontrado na pasta 'target', sendo este um ficheiro de extensão *.jar* que poderá ser executado através do comando:

```
java -jar target/SicaVibeApp-0.0.1-SNAPSHOT.jar
```

Quando o Servidor Aplicacional iniciar, poderá ser visitada a documentação dos **endpoints** do servidor aplicacional via *browser* (http://localhost:8080/).

10.2 Front-end

Para o *front-end* foi utilizada a *framework* Vue.js. Esta possui características bastante favoráveis para um *deployment* de um balanceador de carga, como o facto de ser *client-sided* e *stateless*, ou seja, não há estado entre este e o *back-end*.

Estas características permitem a implementação de várias instâncias sob um balanceador de carga, que distribui o tráfego entre as diferentes instâncias. É de destacar que o balanceador de carga é totalmente independente do servidor a que acede, uma vez que não há necessidade de armazenar estado. Desta forma, o cliente acederia ao *front-end* pelo endereço IP fornecido pelo balanceador de carga, ou seja, por um único ponto de acesso. O balanceador por si só iria redirecionar os pedidos para um simples servidor que apenas retorna os ficheiros (html, css, js, imagens) que assim são pedidos.

É importante notar que como o balanceador é um ponto único de acesso, este também pode acabar por ser um ponto único de falha. Deste modo, isto pode levar a que o sistema se encontre indisponível mesmo quando existem vários servidores prontos a receber pedidos. Evidentemente, esta não é uma situação benéfica e pode ser facilmente resolvida através da implementação de vários balanceadores de carga.

Finalmente, também foi considerada a utilização de vários mecanismos de *caching*, de modo a que as páginas fossem mantidas em *cache*, diminuindo, assim o tempo de resposta e, consequentemente, melhorando a performance.

10.2.1 Servidor Web de Desenvolvimento

Para iniciar o Servidor Web de desenvolvimento é necessário navegar para a diretoria **SicaVibeWeb** e correr os seguintes scripts utilizando a ferramenta *yarn*:

```
cd SicaVibeApp
yarn install
yarn build
```

Quando o Servidor Web iniciar, poderá ser visitado via *browser* (http://localhost:5173/).

10.2.2 Servidor Web de Produção

Para iniciar um Servidor Web de Produção basta compilar o projeto e criar um servidor simples capaz de devolver conteúdos web (como páginas html, css, js, imagens).

Para compilar o projeto pode-se utilizar os seguintes comandos:

```
cd SicaVibeApp
yarn install
yarn build
```

Depois, para iniciar um simples servidor web podemos utilizar a seguinte combinação de comandos:

```
cd SicaVibeApp
npm install -g serve
serve -s dist -l 80
```

Deste modo, o serviço pode ser exposto encontrando-se este em modo de produção.

11 Apresentação das Interfaces Finais

11.1 Páginas de um Funcionário

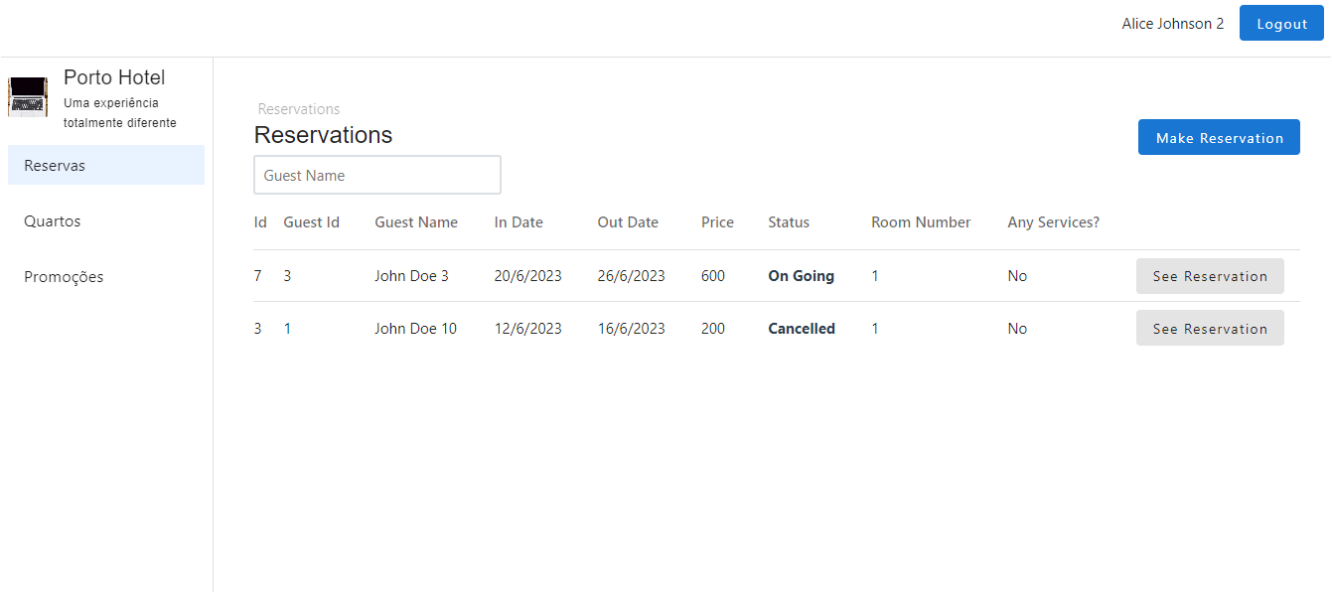


Figura 32: Página de um Funcionário.

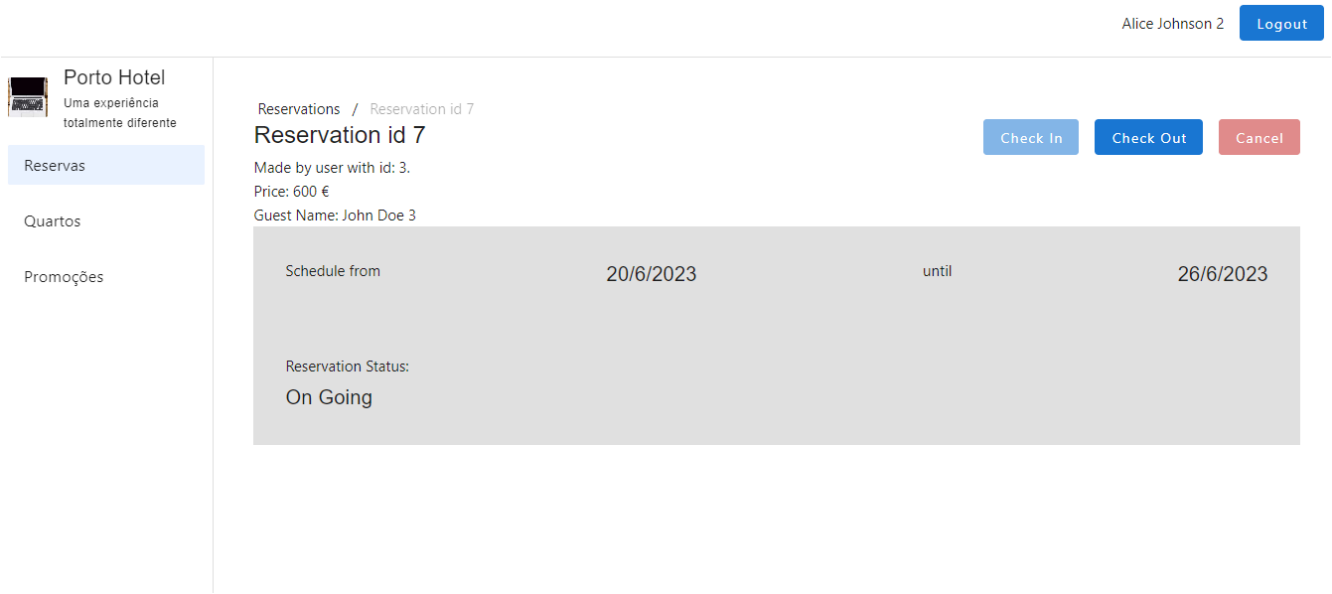


Figura 33: Página de um Funcionário II ("*See Reservation*").

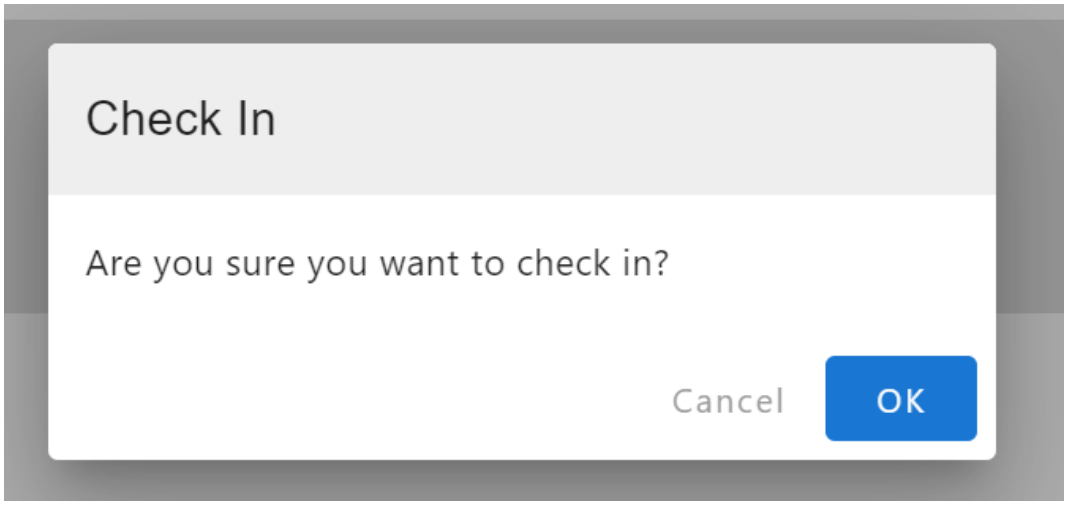


Figura 34: *Pop up* de Confirmação de "*Check In*" do Perfil Funcionário.

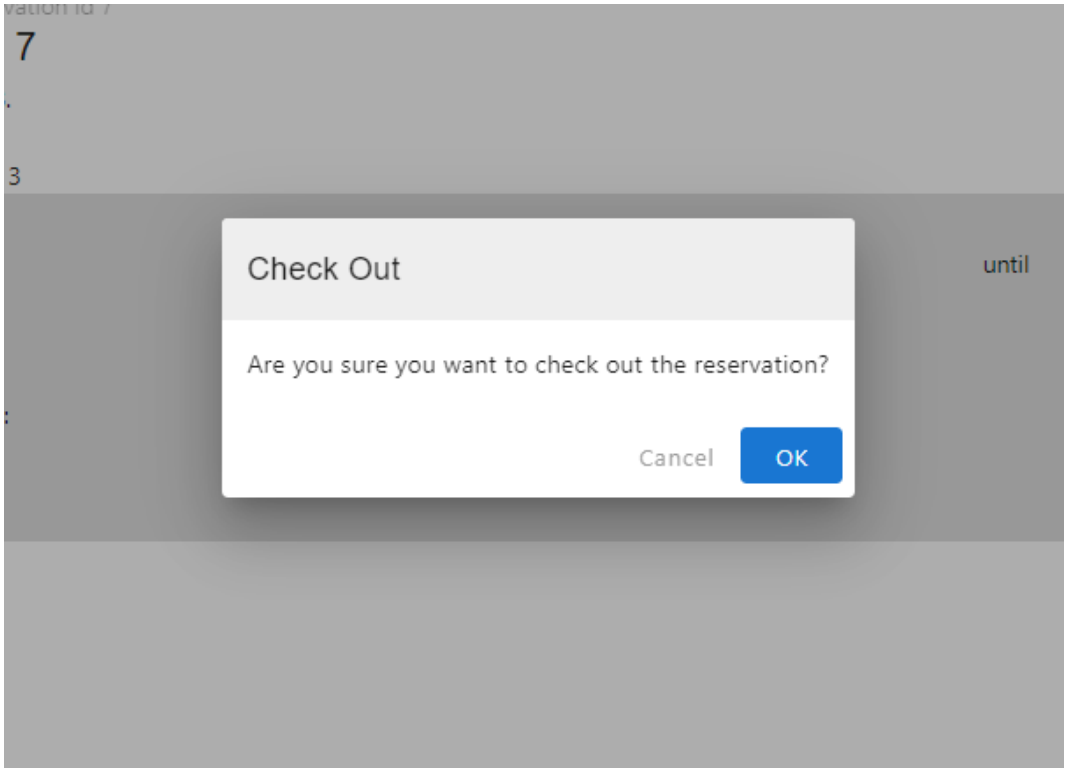


Figura 35: *Pop up* de Confirmação de "*Check Out*" do Perfil Funcionário.

11.2 Páginas de um Administrador

Jane Smith 2 Logout

SicaVibe
Management

Reservations

Employees

Rooms

Reservations

Filter: Hotel ID

SicaVibe Porto

Id	Guest Id	Guest Name	In Date	Out Date	Price	Status	Room Number	Any Services?	
3	1	John Doe 10	12/6/2023	16/6/2023	200	**Cancelled**	1	No	See Reservation
7	3	John Doe 3	20/6/2023	26/6/2023	600	**On Going**	1	No	See Reservation

SicaVibe Faro

Id	Guest Id	Guest Name	In Date	Out Date	Price	Status	Room Number	Any Services?	
12	5	John Doe 5	10/6/2023	14/6/2023	410	**Done**	1	Yes	See Reservation
2	1	John Doe 10	12/6/2023	16/6/2023	210	**Done**	1	Yes	See Reservation
1	1	John Doe 10	23/6/2023	30/6/2023	365	**Done**	2	Yes	See Reservation

Figura 36: Página de um Administrador I ("*Reservations*").

Jane Smith 2 Logout

SicaVibe
Management

Reservations

Employees

Rooms

Reservations / Reservation id 7

Reservation id 7

Made by user with id: 3.
Price: 600 €
Guest Name: John Doe 3

Schedule from 20/6/2023 until 26/6/2023

Reservation Status:
On Going

Figura 37: Página de um Administrador II ("*See Reservation*").

Jane Smith 2 Logout

SicaVibe
Management

Reservations

Employees

Rooms

Employees

ADD EMPLOYEE

ID	Hotel Name	Name	Email	Phone Number
12	SicaVibe Faro	Alice Johnson 1	funcionario1@example.com	111111111
13	SicaVibe Porto	Alice Johnson 2	funcionario2@example.com	111111111
14	SicaVibe Lisboa	Alice Johnson 3	funcionario3@example.com	111111111
15	SicaVibe Coimbra	Alice Johnson 4	funcionario4@example.com	111111111
16	SicaVibe Porto	Alice Johnson 5	funcionario5@example.com	111111111
17	SicaVibe Lisboa	Alice Johnson 6	funcionario6@example.com	111111111

Figura 38: Página de um Administrador III (”*Employees*”).

Jane Smith 2 Logout

SicaVibe
Management

Reservations

Employees

Rooms

Employees

ADD EMPLOYEE

ID	Hotel Name	Phone Number
12	SicaVibe Faro	111111111
13	SicaVibe Porto	111111111
14	SicaVibe Lisboa	111111111
15	SicaVibe Coimbra	111111111
16	SicaVibe Porto	111111111
17	SicaVibe Lisboa	111111111

Add New Employee

Name

Email

Address

Birthday

dd/mm/aaaa

Phone Number

NIF

CC

Password

Confirm Password

Hotel:

Cancel Register

Figura 39: Página de um Administrador IV (”*Add Employee*”).

Jane Smith 2 Logout

SicaVibe
Management

Reservations

Employees

Rooms

Rooms

Rooms

Add New Room

Hotel Id

Hotel id: 1

Id	TypeId	TypeName	DoorNumber	Status
1	1	Single Room	101	LIVRE
2	1	Single Room	102	LIVRE
3	1	Single Room	103	POR_LIMPAR
4	1	Single Room	104	POR_LIMPAR
7	2	Twin Room	105	LIVRE
8	2	Twin Room	106	LIVRE
5	1	Single Room	201	LIVRE

Figura 40: Página de um Administrador V (”*Rooms*”).

Jane Smith 2 Logout

SicaVibe
Management

Reservations

Employees

Rooms

Rooms

Rooms

Add New Room

Hotel Id

Hotel id: 1

Add New Room

Room Details

Room Description

Cancel Add Room

Id	TypeId			Status
1	1			LIVRE
2	1			LIVRE
3	1			POR_LIMPAR
4	1	Single Room	104	POR_LIMPAR
7	2	Twin Room	105	LIVRE
8	2	Twin Room	106	LIVRE
5	1	Single Room	201	LIVRE

Figura 41: Página de um Administrador VI (”*Add New Room*”).

11.3 Páginas de um Hóspede

Figura 42: Página Inicial.

SicaVibe

LOGIN SIGN UP

Faro

Explore the enchantment of SicaVibe Faro, a contemporary hotel nestled in the heart of charming Faro, Portugal. Immerse yourself in modern comfort and local authenticity as you discover the captivating beauty of the Algarve region. Relax in our stylish and well-appointed rooms, designed to provide a tranquil haven after a day of exploration. Indulge in delectable cuisine at our on-site restaurant, showcasing a fusion of local and international flavors. Unwind and socialize in our inviting lounge area or take a refreshing dip in our sparkling outdoor pool. With a prime location near pristine beaches, historic sites, and lively markets, SicaVibe Faro offers the perfect gateway to discover the Algarves treasures. Immerse yourself in the coastal charm, from the scenic Ria Formosa to the captivating old town. Experience the vibrant energy and warm hospitality of Faro at SicaVibe, where modern comfort meets the authentic charm of the Algarve.

Main Amenities

- Parking
- Pool Access
- Laundry Service
- Spa
- Breakfast

BOOK A RESERVATION

Faro Rooms

Deluxe Room

Indulge in our deluxe room, where luxury meets relaxation. Experience spacious comfort, elegant decor, and upscale amenities. Unwind in style, enjoy the plush bedding, and rejuvenate in the lavish en-suite bathroom. Escape to a haven of tranquility in our deluxe room.

Capacity: 3 people

Price per Night: 150€

Presidential Suite

Indulge in luxury with our suite room, a haven of elegance and sophistication. Enjoy spacious accommodations, a separate living area, and upscale amenities. Retreat to a plush bedroom, relax in style, and experience the pinnacle of comfort and refinement in our exquisite suite room.

Capacity: 4 people

Price per Night: 200€

Twin Room

Our twin room offers comfort and functionality with a cozy ambiance. Rest peacefully on a comfortable bed, enjoy modern amenities, and refresh in the en-suite bathroom. Experience a delightful blend of comfort and convenience in our standard room.

Capacity: 2 people

Price per Night: 100€

Single Room

Welcome to our cozy single room, a comfortable retreat designed for solo travelers. Enjoy a restful nights sleep on a cozy bed, relax in a well-appointed space, and make use of modern amenities. Experience a tranquil and convenient stay in our inviting single room.

Capacity: 1 people

Price per Night: 50€

Figura 43: Página de um Hotel.

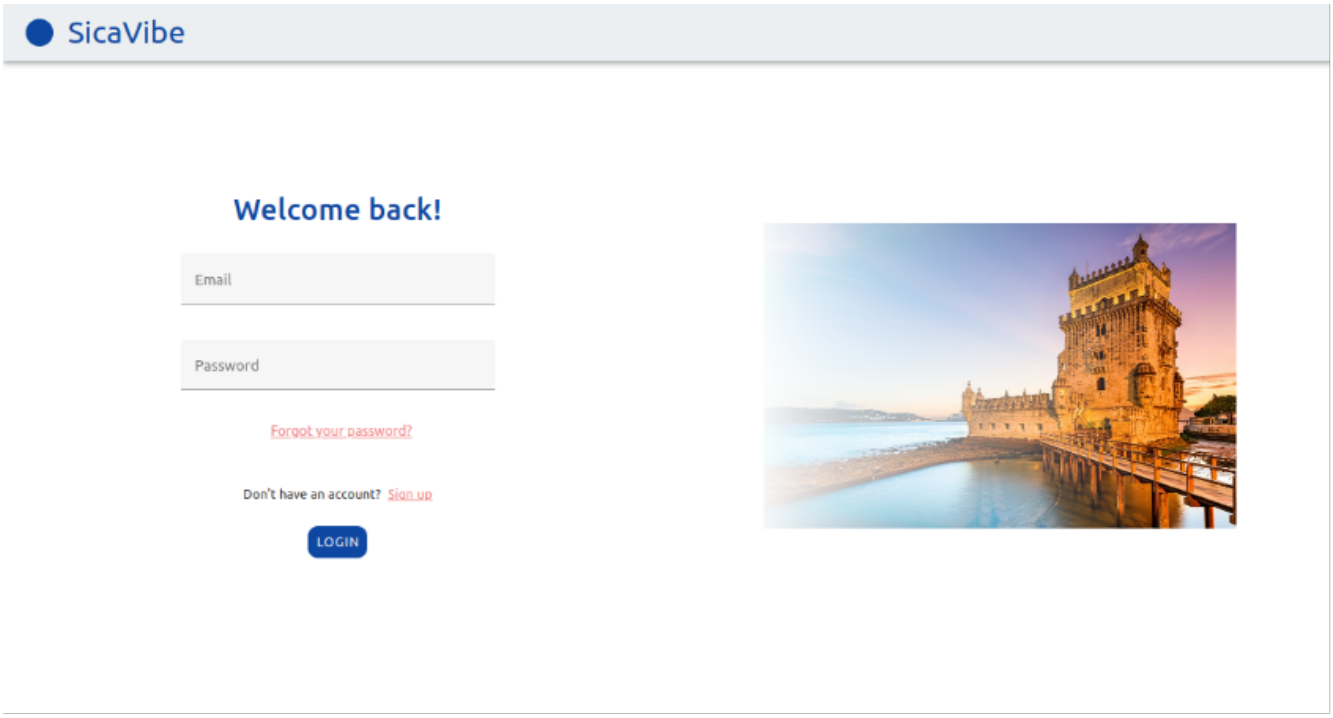


Figura 44: Página de *Login*.

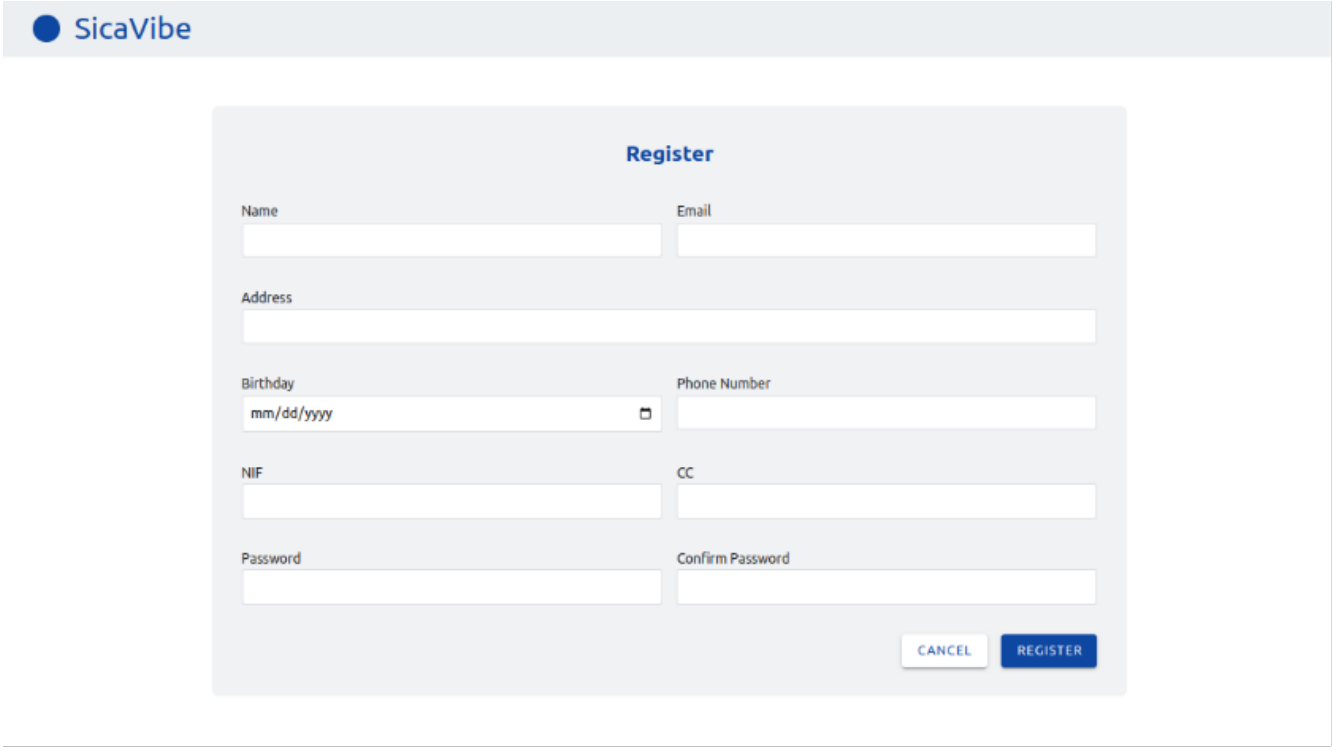


Figura 45: Página de Registo.

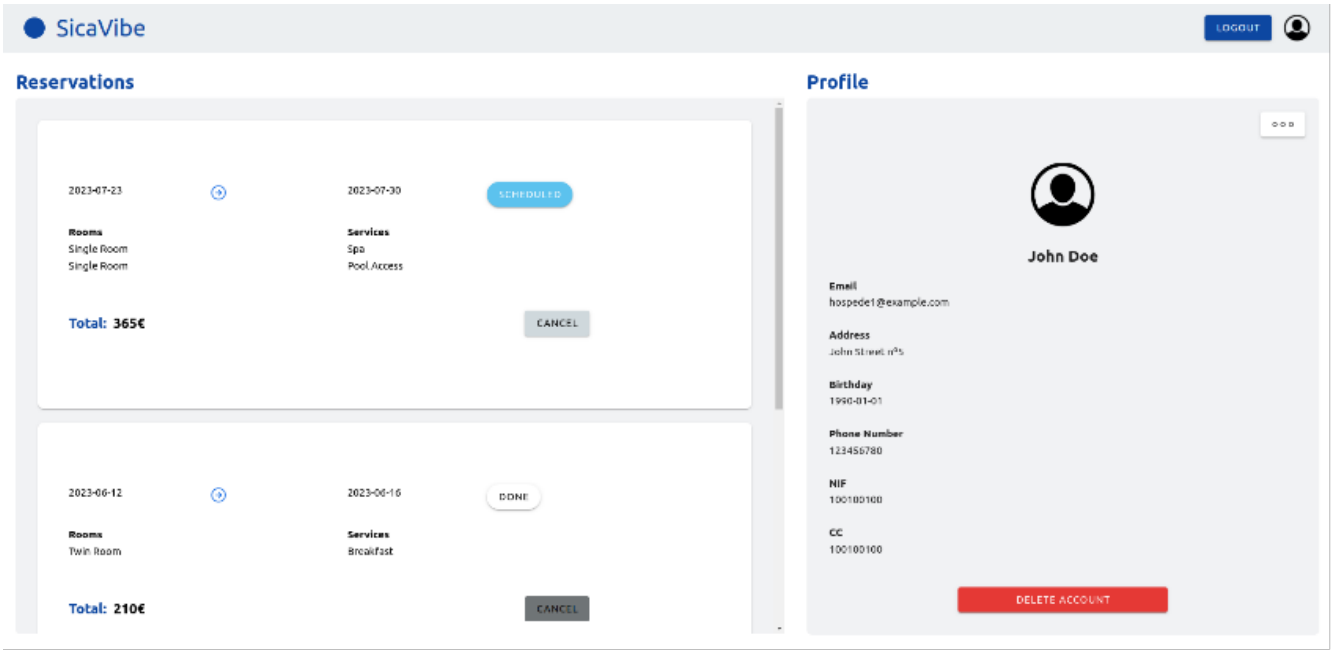


Figura 46: Página de Perfil Hóspede.

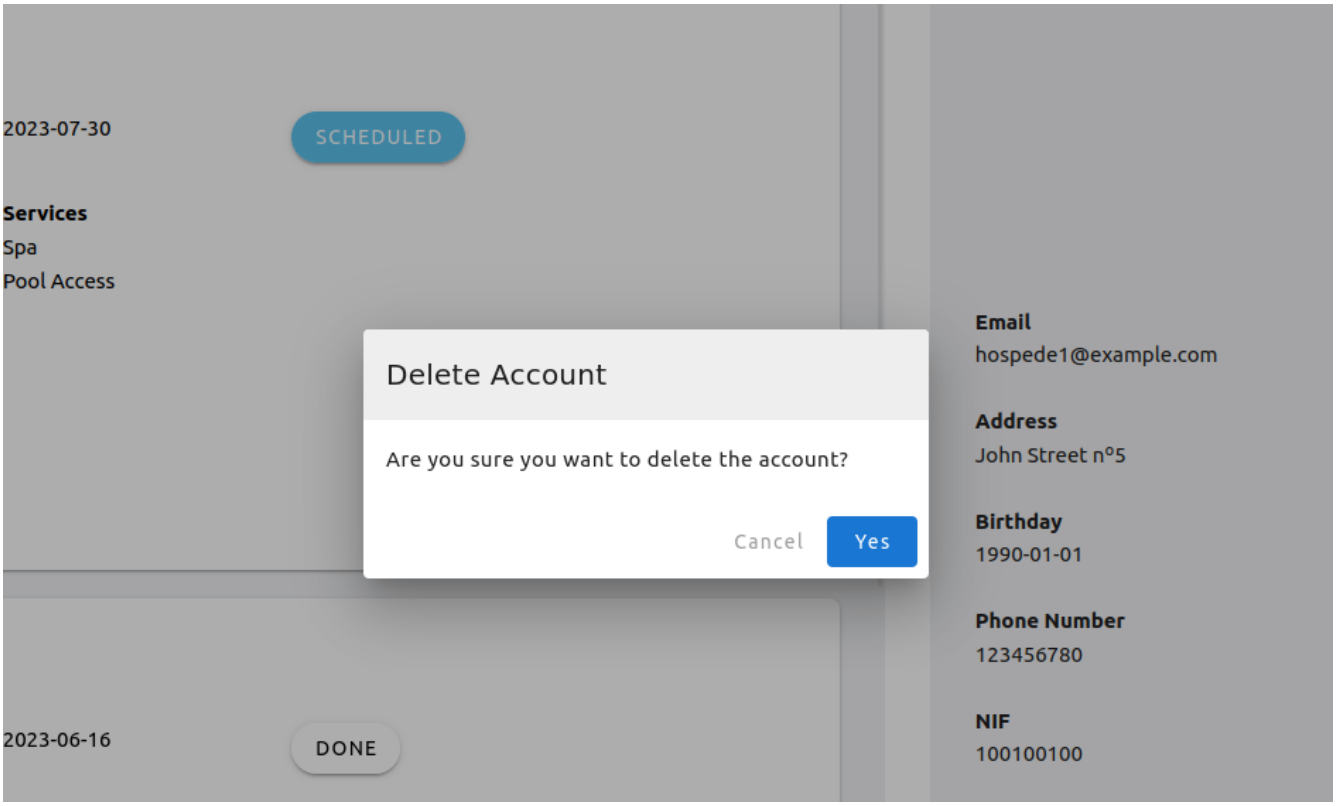


Figura 47: *Pop up* de Confirmação de Eliminação de Perfil Hóspede.

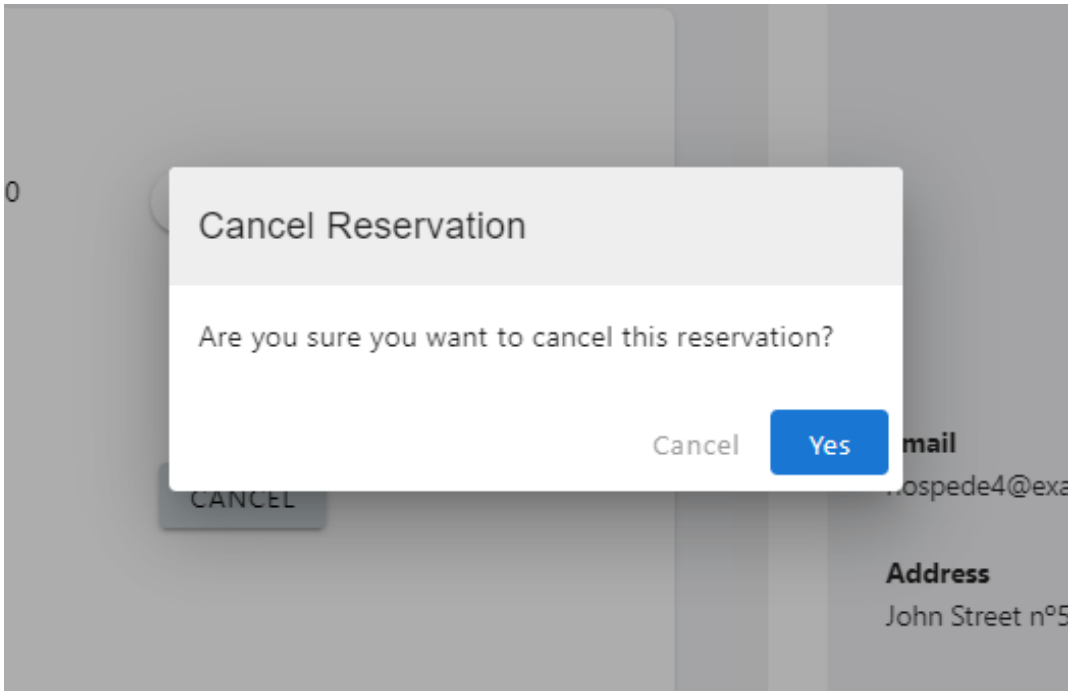


Figura 48: *Pop up* de Confirmação de Eliminação de Reserva do Perfil Hóspede.

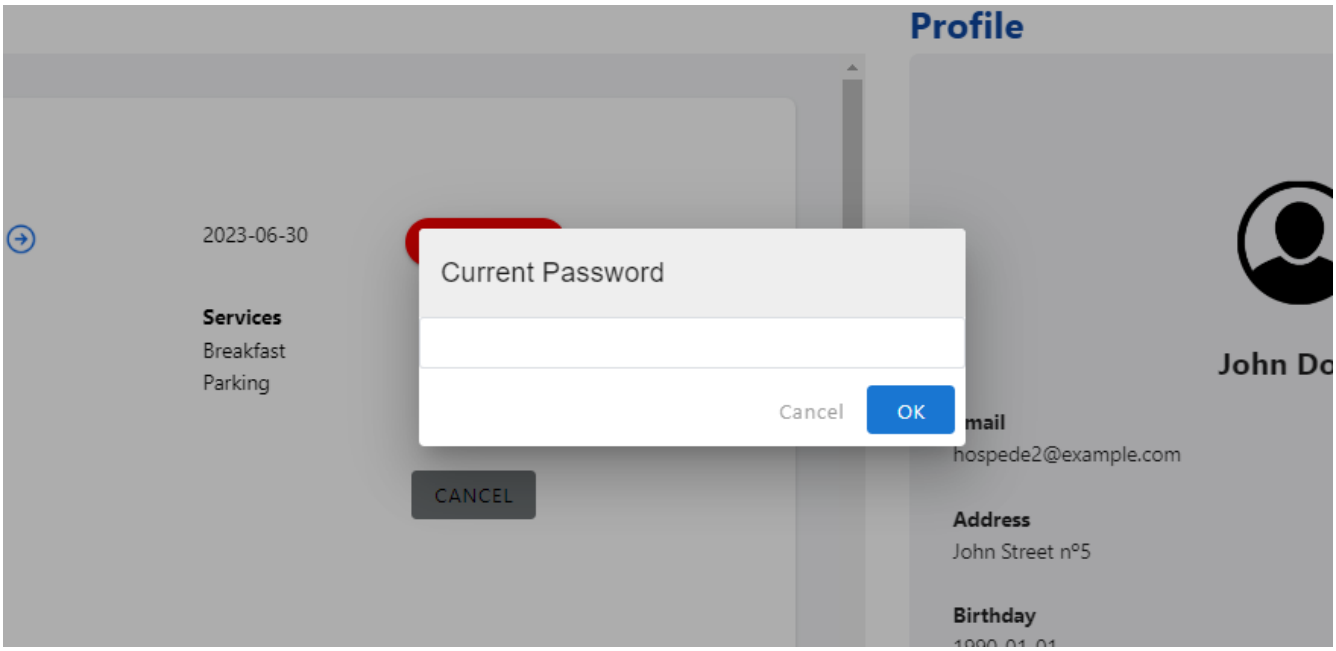


Figura 49: *Pop up* de Confirmação antes de permitir alterar a password do Perfil Hóspede.

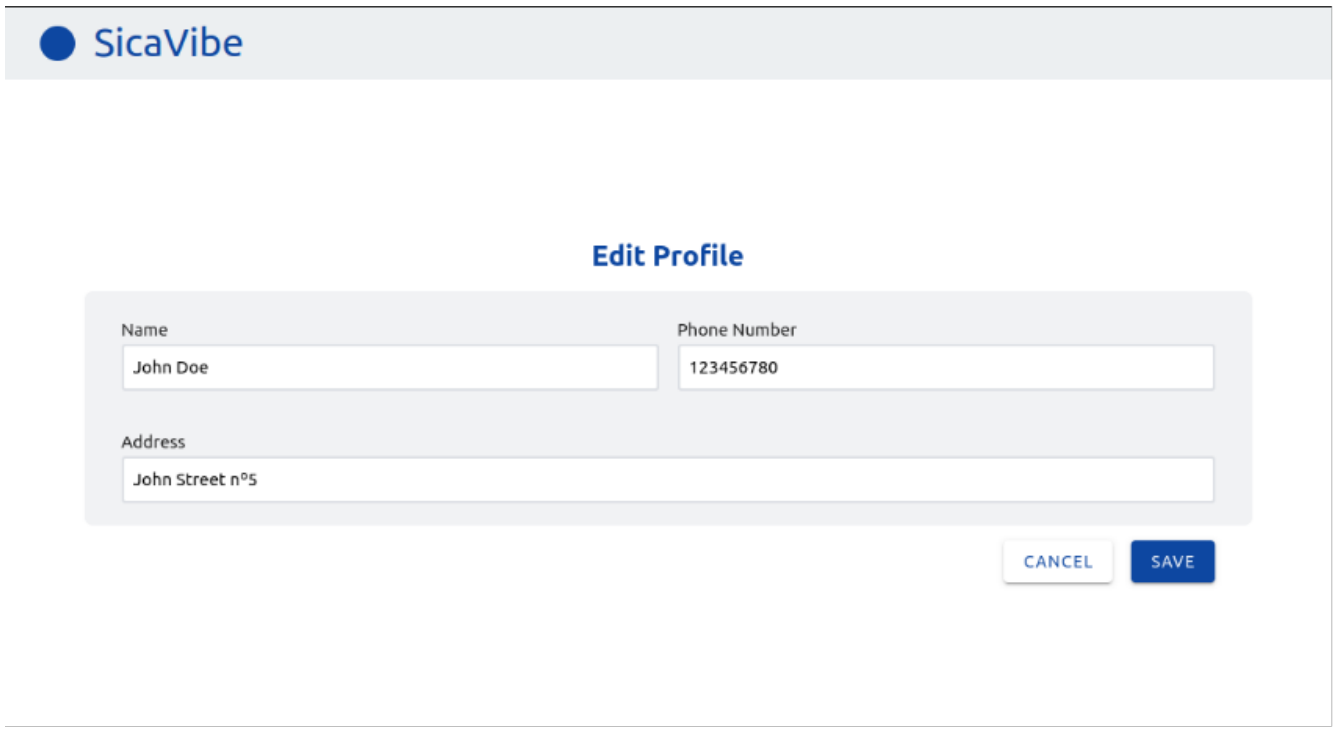


Figura 50: Página de Editar Perfil Hóspede.

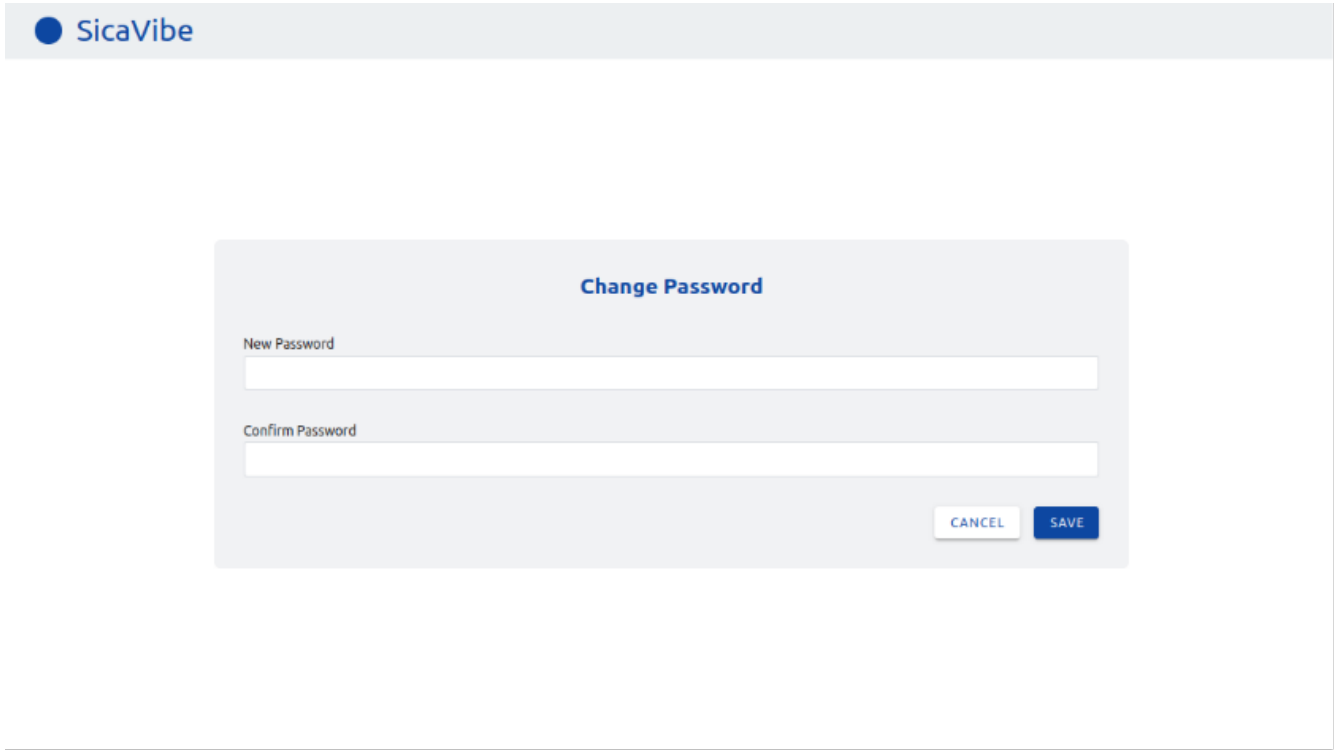


Figura 51: Página de Alterar Password.

11.4 Páginas de uma Reserva

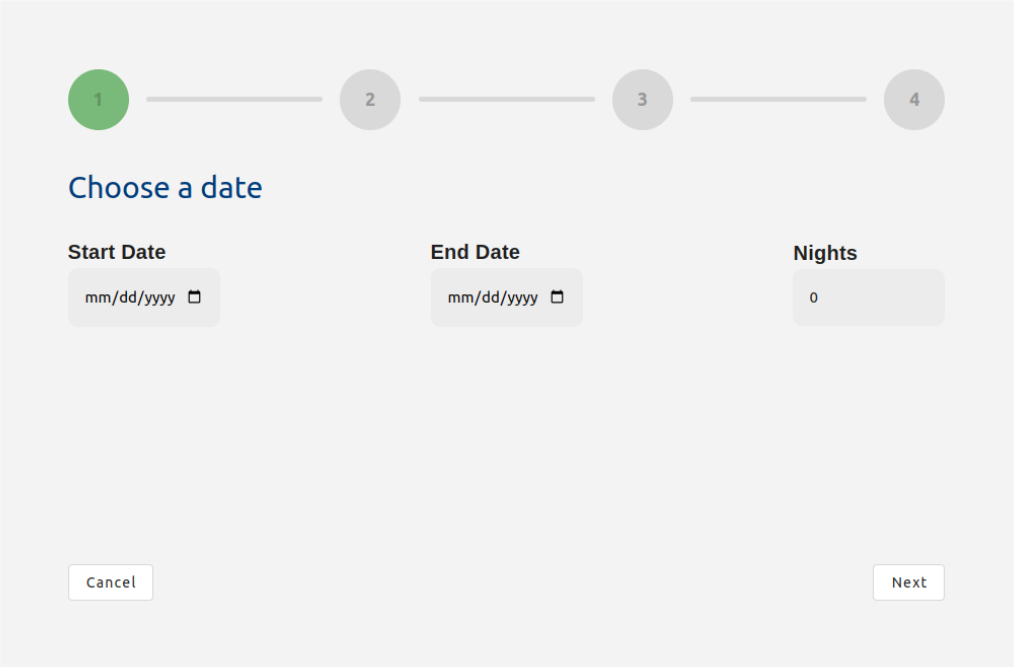


Figura 52: Efetuar Reserva Passo 1.

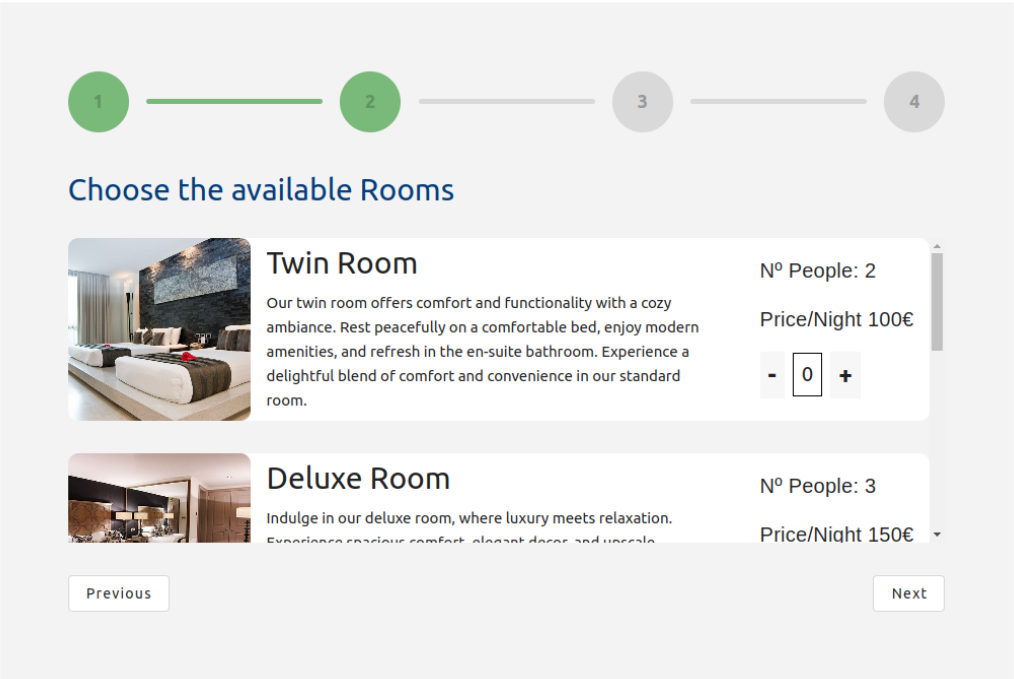


Figura 53: Efetuar Reserva Passo 2.

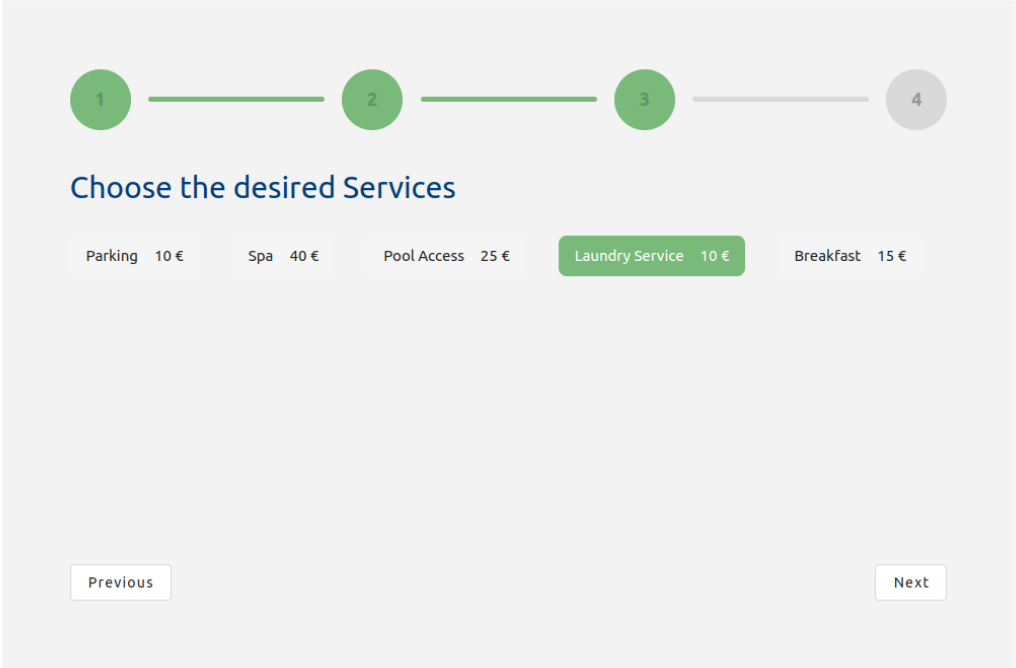


Figura 54: Efetuar Reserva Passo 3.

Figura 55: Efetuar Reserva Passo 4.

12 Análise de Usabilidade

12.1 Princípios de Usabilidade

12.1.1 *Learnability*

Este princípio é baseado na facilidade com que novos utilizadores podem iniciar uma interação eficaz e alcançar a sua máxima performance. Este contém as seguintes características: ***Predictability***, ***Synthesizability***, ***Familiarity***, ***Generalizability*** e ***Consistency***.

Relativamente à ***Predictability***, são múltiplos os exemplos desta propriedade que se podem observar ao longo desta aplicação. A título de exemplo, consideremos a figura 41, em que um administrador adiciona um novo quarto. A execução do botão *Add Room* ou do *Cancelar* executa a ação esperada pelo utilizador. Da mesma forma, podemos considerar os botões, *Login* e *Sign Up*, da *Página Inicial*, que tal como os nomes referem, dirigem o utilizador para os formulários de *login* e de *sign up*/registo.

Para a propriedade ***Synthesizability***, quando um hóspede edita o seu perfil, este é, em seguida, redirecionado de volta para a sua página de perfil, percebendo imediatamente a alteração. Destaca-se que este redirecionamento sucede após a apresentação de uma mensagem de confirmação do sucesso da alteração dos dados, em caso de sucesso. De forma semelhante, quando o utilizador altera a sua palavra-passe com sucesso, uma mensagem de sucesso é lhe apresentada e este é redirecionado para a sua página de perfil.

No caso da ***Familiarity***, esta propriedade também está presente no registo de novos hóspedes e funcionários, uma vez que trata de um formulário simples, utilizado por diversas aplicações.

Por último, a propriedade ***Consistency*** foi sempre considerada, com cuidado, na aplicação, no design da mesma em vários aspetos, como por exemplo, no registo de novos funcionários e hóspedes, na adição de quartos, entre outros.

12.1.2 *Flexibility*

Este princípio diz respeito à diversidade de formas de troca de informações entre o utilizador e o sistema. Este envolve as seguintes propriedades: ***Dialogue Initiative***, ***Multithreading***, ***Task Migrability***, ***Substitutivity*** e ***Customizability***.

Quanto à propriedade ***Dialogue Initiative***, temos que o utilizador controla o fluxo do programa, como exemplo podemos considerar o caso em que o hóspede cria uma nova reserva. Ora, tendo em conta as figuras 52, 53, 54 e 55 que representam o fluxo de criar uma nova reserva, podemos perceber que o modal exibido está de acordo com as ações do hóspede, isto é, durante o processo de realização de uma reserva, o utilizador consegue voltar para trás no decorrer do mesmo. Para além deste, ainda existem também outros casos em que o modal é apresentado conforme o resultado das ações dos utilizadores, bloqueando o ecrã.

Em relação à *Multithreading*, esta propriedade é possível de observar no caso das listas dos quartos e de reservas, uma vez que o utilizador consegue filtrar as mesmas pelo hotel ou pela reserva/hotel, respetivamente. Para além disto, o grupo reconhece que existe, ainda, espaço para melhoria neste ponto, através da adição da possibilidade de ordenar as listas e da possibilidade do hóspede também conseguir filtrar o seu histórico de reservas.

Por fim, as propriedades ***Task Migrability***, ***Substitutivity*** e ***Customizability*** não foram aplicadas, visto que as páginas correspondem a conteúdos estáticos.

12.1.3 *Robustness*

O princípio da ***Robustness*** diz respeito ao nível de suporte fornecido ao utilizador para determinar a realização bem sucedida e para avaliar o comportamento orientado para os objetivos. Esta contém as seguintes propriedades: ***Observability***, ***Recoverability***, ***Responsiveness*** e ***Task Conformance***.

Relativamente à propriedade ***Observability***, esta está presente em várias secções, uma vez que é possível saber o estado interno do sistema. Por exemplo, nas listas de quartos, de reservas e histórico de reservas, que estão sempre atualizadas. Ou, ainda, nas mensagens de sucesso ou de erro quando uma ação executada por um utilizador é ou não possível.

Quanto à ***Recoverability***, esta aplica-se nos caso de registo, da edição de perfil e da alteração da *password*. Caso o utilizador se enganee no preenchimento destes formulários é lhe permitido reescrever os mesmo campos (*forward recovery*), uma vez que a ação não é executada até os mesmos estarem todos de acordo com as normas estipuladas. Para além destes casos, esta propriedade também é observada durante o processo de realização de uma reserva, uma vez que é permitido ao utilizador voltar atrás no momento em que está a efetuar a reserva, podendo alterar alguma informação. A propriedade ***Responsiveness*** está presente em todas em todas as páginas.

Por último, a propriedade de ***Task Conformance*** pode ser observada nos caso das listas dos quartos e de reservas, devido à possibilidade do utilizador filtrar as mesmas, respetivamente, pelo hotel ou pela reserva/hotel. Tal como já referimos numa das propriedades anteriores, é possível melhorar esta propriedade.

12.2 Heurísticas de Nielsen

12.2.1 *Visibility of system status*

Esta heurística diz respeito à "visibilidade do estado do sistema", ou seja, se o sistema mantém os utilizadores informados sobre o que se passa, através de *feedback* apropriado.

De acordo com o resultado da ação executada pelo utilizador, são lhe apresentadas mensagens de confirmação, como por exemplo, a mensagem após a edição dos seus dados de perfil, ou após a alteração da sua *password* ou depois do registo de um novo hóspede, como podemos observar pelas seguintes imagens:

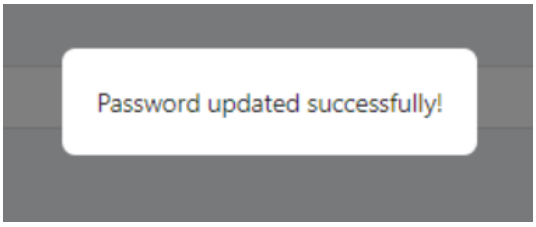


Figura 56: Mensagem de confirmação de alteração de password com sucesso.

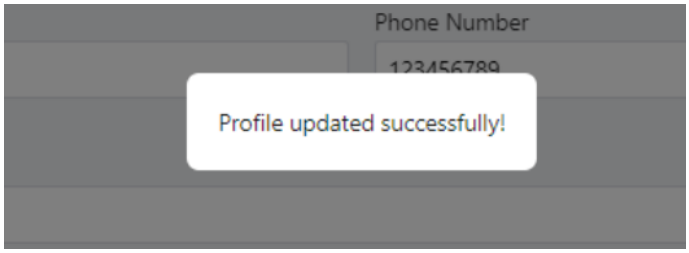


Figura 57: Mensagem de confirmação de edição do perfil com sucesso.

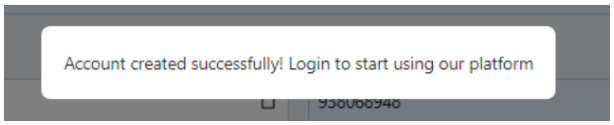


Figura 58: Mensagem de confirmação de registo de nova conta com sucesso.

É também relevante comentar sobre a ação de cancelar uma reserva executada pelo hóspede, uma vez que apesar de não ser enviada uma mensagem de confirmação, é possível observar a alteração do seu estado, como podemos ver pelas figuras 59 e 60, em que a primeira corresponde ao estado da reserva antes do seu cancelamento e a segunda trata-se do estado da reserva após o seu cancelamento.

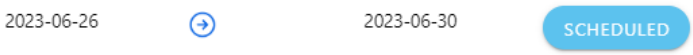


Figura 59: Reservada marcada.

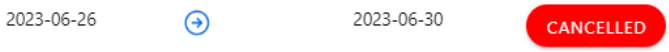


Figura 60: Reserva cancelada.

12.2.2 *Match between system and the real world*

Esta heurística é referente ao facto de que a aplicação *web* corresponda ao mundo real, ou seja, que utilize uma linguagem e uma estruturação da interface familiares aos utilizadores alvo. Na figura 55 é possível observar que a fatura da reserva tem o mesmo formato ou muito semelhante ao de uma fatura de um hotel. Para além disto, esta heurística também afirma que a informação deve ser apresentada de uma forma natural e lógica, sendo que isto está presente no processo de efetuar um reserva.

12.2.3 *User control and freedom*

Por sua vez, esta heurística diz respeito aos utilizadores executarem frequentemente ações por engano e pelo que se deve fornecer "saídas de emergência"claramente marcadas. Um exemplo da utilização desta são os menus de confirmação de eliminação de uma conta e cancelamento de uma

reserva, que aparecem quando o hóspede quer eliminar a sua conta ou cancelar uma reserva. Neste menus, existe a opção de "Cancel"que permite ao utilizador fazer "undo"caso o mesmo decida não prosseguir com essa ação. Na figura 51 é possível observar um exemplos desses menus.

12.2.4 *Consistency and standards*

Esta heurística refere que se devem seguir as convenções da plataforma e da indústria, para além de que as palavras, situações, ou ações distintas devem ter significados distintos. Consequentemente, é possível observar nas diversas páginas criadas, a referência a expressões e a ações relacionadas com o contexto hoteleiro deste produto. Para além disto, os botões *Cancel* na aplicação estão com uma cor esbatida ou até mesmo da cor de fundo, o que é consistente na maioria das aplicações. Destacamos, ainda, o facto dos botões referentes a operações como *Cancel* ou *Preview* estarem posicionados no lado esquerdo, enquanto os botões como *Next* estarem posicionados no lado direito do ecrã.

12.2.5 *Error prevention*

Esta heurística afirma que boas mensagens de erro são importantes, mas ainda mais é evitar a ocorrência de problemas. Esta heurística é utilizada no preenchimento dos formulários. Os campos *Phone Number*, *NIF* e *CC* não permitem que o utilizador escreva mais números do que o número permitido para cada campo. Sendo que este número para cada campo varia de acordo com a sua informação, isto é, os campos *Phone Number* e *NIF* só permite que escreva até 9 números e o campo *CC* até 8 números.

12.2.6 *Recognition rather than recall*

Esta heurística afirma que a informação necessária deve ser visível ou facilmente recuperável de modo a minimizar a carga de memória do utilizador. Um exemplo da utilização desta heurística é encontrado na página de edição do perfil do hóspede (figura 51), uma vez que as suas informações antigas permanecem visíveis. Deste modo, o hóspede pode apenas editar o campo que pretende sem precisar de lembrar das suas informações antigas.

12.2.7 *Flexibility and efficiency of use*

Esta heurística aborda a criação e a existência de atalhos para acelerar a experiência dos utilizadores experientes e que estejam ocultos dos utilizadores principiantes.

De acordo com esta diretiva, temos os exemplos, em que tanto um Funcionário ou um Administrador que pretendem ver uma reserva em específico, podem apenas na página principal clicarem no botão respetivo da reserva.

12.2.8 *Aesthetic and minimalist design*

Esta heurística refere que as interfaces não devem conter informação que seja irrelevante ou raramente necessária. O grupo considera que a aplicação cumpre este requisito, uma vez que apresenta um design simples, transmitindo ao utilizador apenas a informação relevante de uma forma clara.

12.2.9 *Help users recognize and recover from errors*

Esta heurística afirma que as mensagens de erro devem ser numa linguagem simples (sem códigos de erro), indicando o problema e a possível solução. Um exemplo da aplicação desta heurística é no preenchimento dos formulários, quando é apresentada uma mensagem de erro no campo específico. Por exemplo, quando o utilizador insere dados incorretos no registo, não o permitindo de terminar a ação com sucesso, as mensagens de erros são exibidas

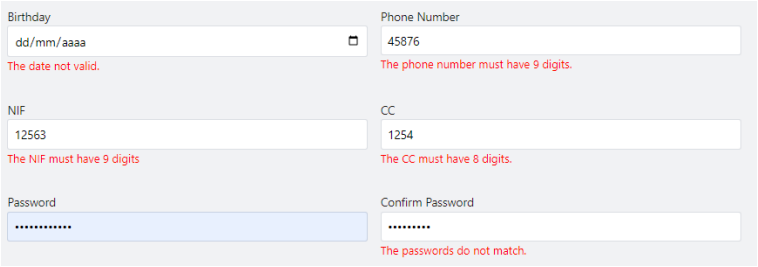


Figura 61: Erro no registo.

12.2.10 *Help and documentation*

Nesta heurística afirma-se que um sistema não deve necessitar de qualquer explicação ou informação adicional. No entanto, existem casos em que pode ser necessário.

Enquanto grupo consideramos que por se tratar de uma aplicação simples, que transmite de forma clara a informação, julgamos que não existia a necessidade de fornecer documentação e/ou uma página de ajuda.

13 Conclusões e Trabalho Futuro

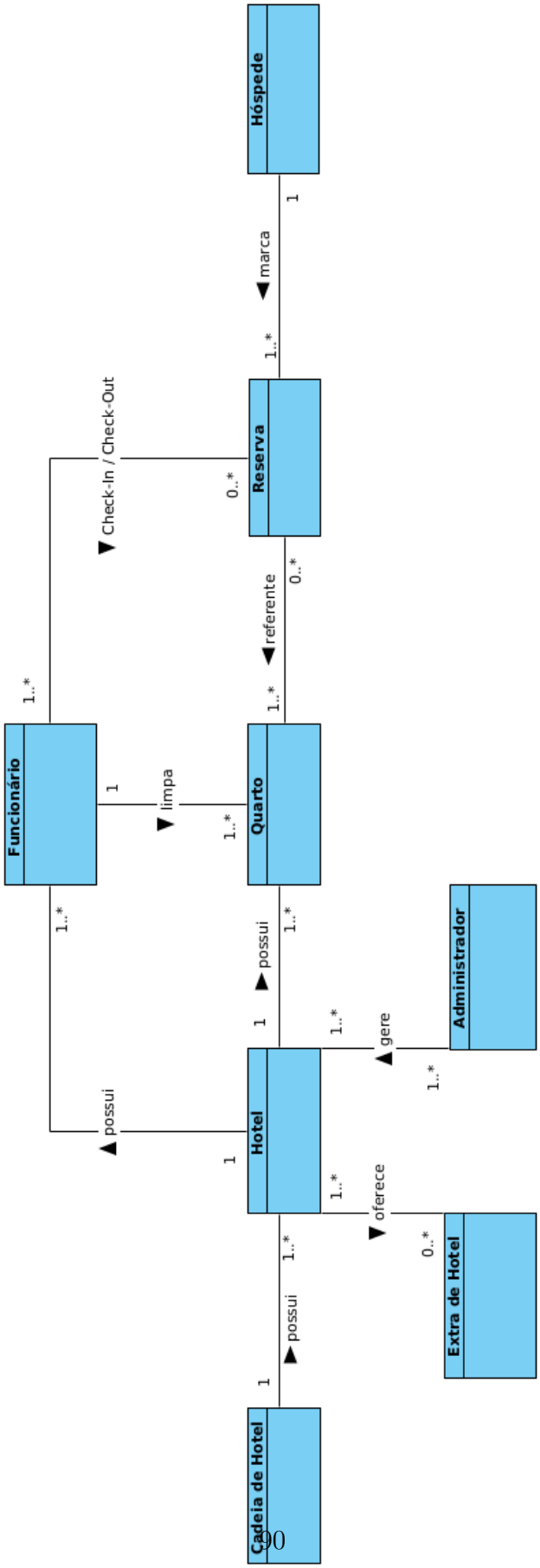
A realização deste trabalho prático possibilitou não só uma melhor consolidação dos conceitos lecionados nas aulas teóricas e práticas, como também possibilitou a aplicação dos mesmos num projeto de maior escala, comparativamente aos realizados nas aulas. Para além disso, permitiu uma imersão mais aprofundada na framework *Vue* e aumentar os conhecimentos relativos a esta, já que nem todos os elementos do grupo tinham uma grande experiência com esta *framework*.

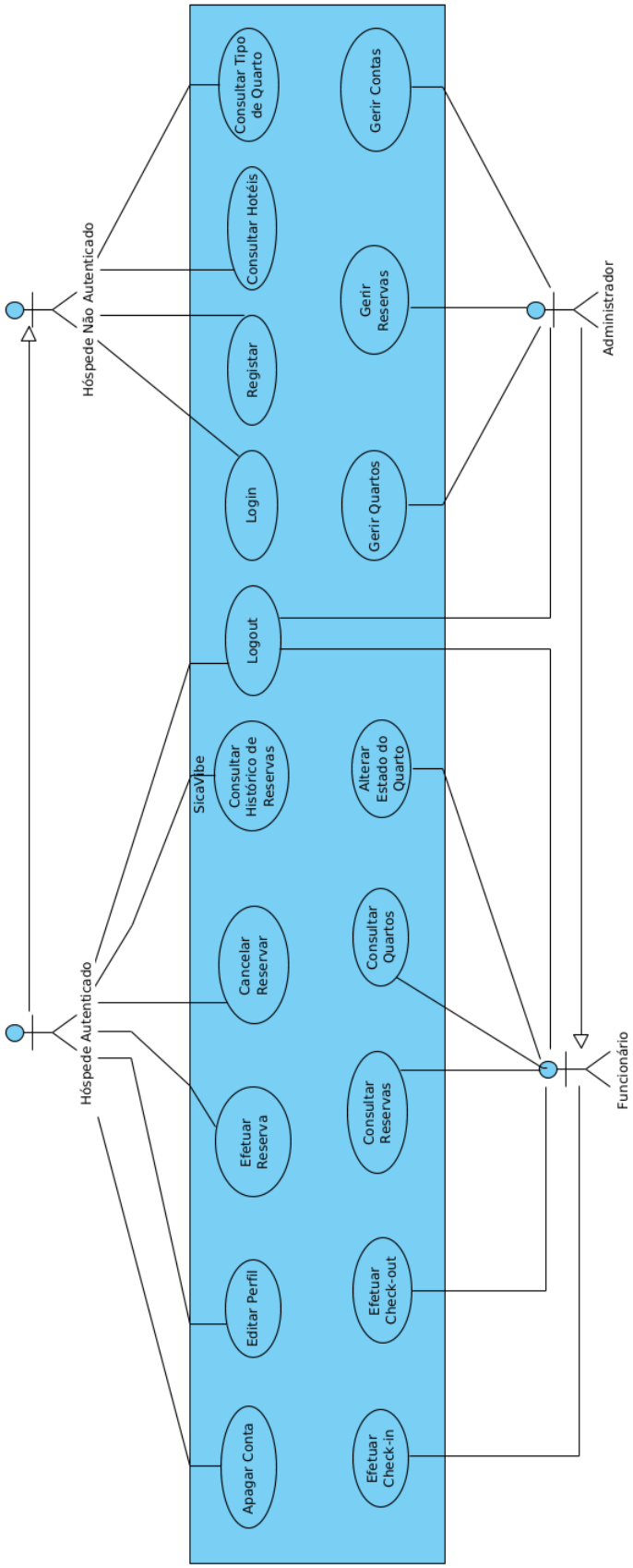
No que toca ao **fronted**, um dos motivos pelos quais não foram atingidos todos os objetivos e requisitos da aplicação, foi a inexperiência dos vários elementos do grupo com tecnologias *web* e, principalmente, com a *framework Vue.js*. No entanto, conseguimos ultrapassar todas as dificuldades e tendo em as conta, consideramos, enquanto grupo, que atingimos um produto final razoável.

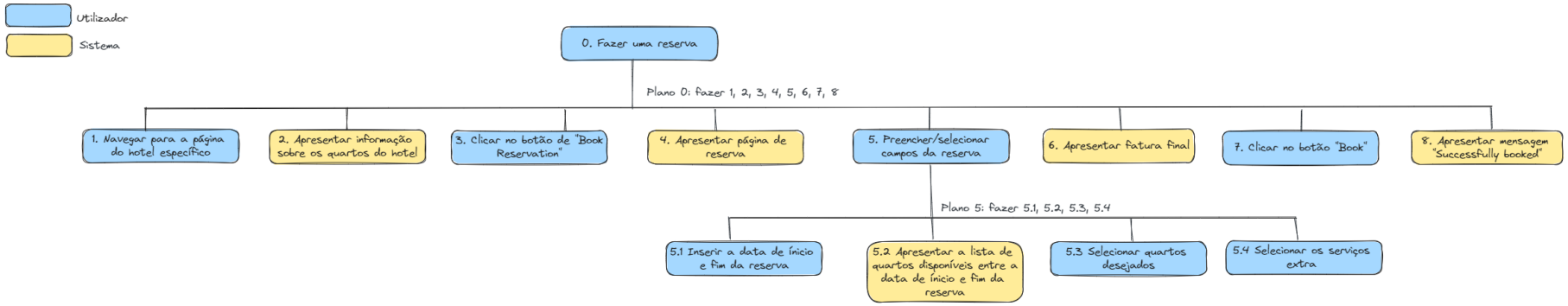
Relativamente ao **backend**, consideramos que foram atingidos todos os objetivos impostos pelo grupo, tendo sido obtido um servidor aplicacional funcional e facilmente extensível ou modificável para mais funcionalidades.

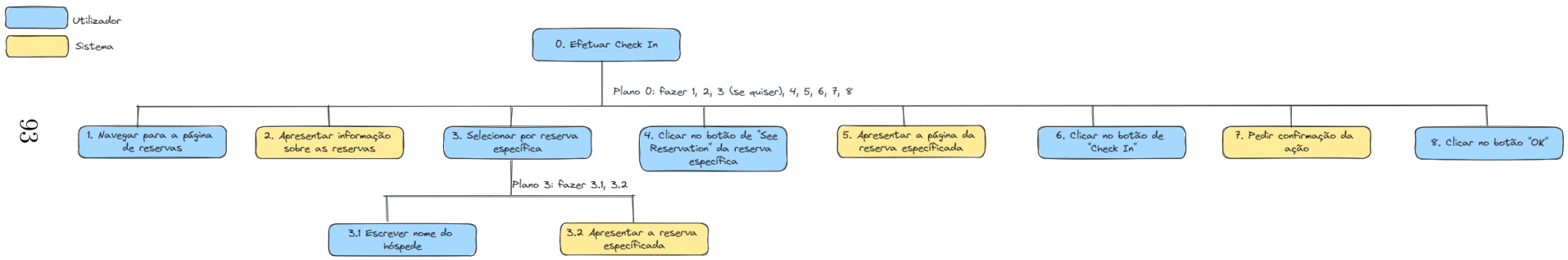
Em suma, no que diz respeito ao trabalho desenvolvido como um todo, consideramos que, apesar dos diversos desafios com que nos deparamos durante a sua resolução, o resultado obtido é satisfatório. Existindo, contudo e de forma clara, alguns aspetos que podem ser melhorados. Como trabalho futuro, o grupo gostaria de implementar os requisitos classificados como *Could*, entre outros.

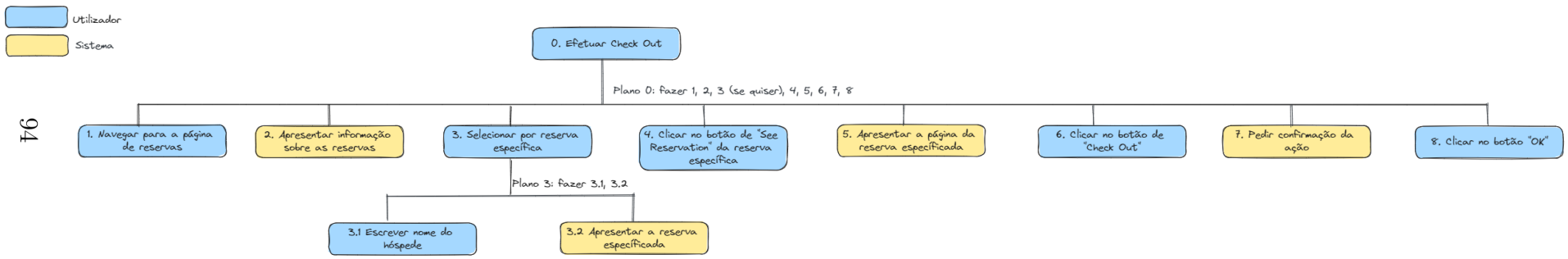
14 Anexos

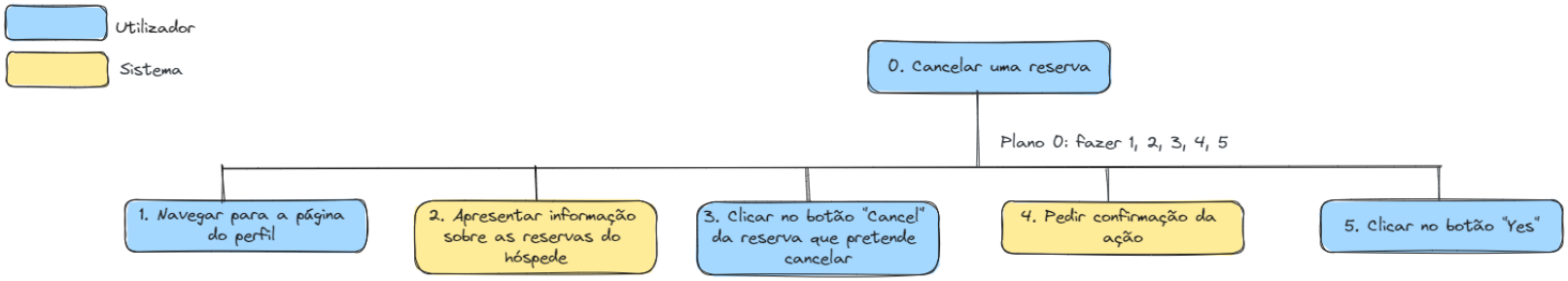

Hóspede
1
marca
1..*
Reserva
0..*
Check-In / Check-Out
1..*
Funcionário
1
limpa
1..*
Quarto
0..*
referente
1..*
1..*
possui
1..*
1
Hotel
1..*
gere
1..*
Administrador
1
possui
1..*
1..*
oferece
0..*
Extra de Hotel
1
Cadeia de Hotel

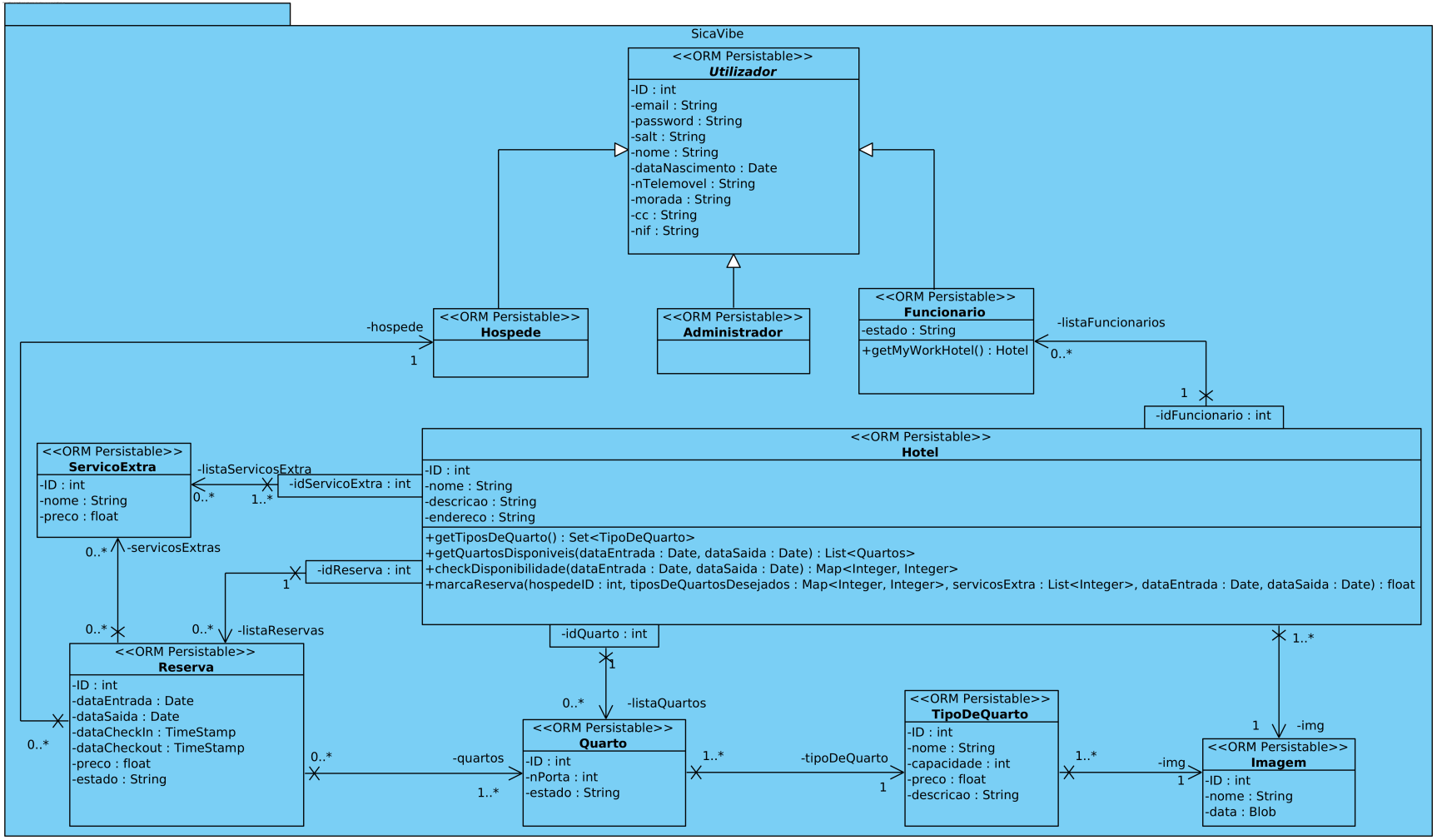

Hóspede Não Autenticado
Hóspede Autenticado
Administrador
Funcionário
SicaVibe
Consultar Tipo de Quarto
Consultar Hotéis
Registar
Login
Logout
Consultar Histórico de Reservas
Cancelar Reservar
Efetuar Reserva
Editar Perfil
Apagar Conta
Gerir Contas
Gerir Reservas
Gerir Quartos
Alterar Estado do Quarto
Consultar Quartos
Consultar Reservas
Efetuar Check-out
Efetuar Check-in


Utilizador
Sistema
0. Fazer uma reserva
Plano 0: fazer 1, 2, 3, 4, 5, 6, 7, 8
1. Navegar para a página do hotel específico
2. Apresentar informação sobre os quartos do hotel
3. Clicar no botão de "Book Reservation"
4. Apresentar página de reserva
5. Preencher/selecionar campos da reserva
6. Apresentar fatura final
7. Clicar no botão "Book"
8. Apresentar mensagem "Successfully booked"
Plano 5: fazer 5.1, 5.2, 5.3, 5.4
5.1 Inserir a data de ínicio e fim da reserva
5.2 Apresentar a lista de quartos disponíveis entre a data de ínicio e fim da reserva
5.3 Selecionar quartos desejados
5.4 Selecionar os serviços extra


Utilizador
Sistema
0. Efetuar Check In
Plano 0: fazer 1, 2, 3 (se quiser), 4, 5, 6, 7, 8
1. Navegar para a página de reservas
2. Apresentar informação sobre as reservas
3. Selecionar por reserva específica
4. Clicar no botão de "See Reservation" da reserva específica
5. Apresentar a página da reserva específicada
6. Clicar no botão de "Check In"
7. Pedir confirmação da ação
8. Clicar no botão "OK"
Plano 3: fazer 3.1, 3.2
3.1 Escrever nome do hóspede
3.2 Apresentar a reserva específicada


Utilizador
Sistema
0. Efetuar Check Out
Plano 0: fazer 1, 2, 3 (se quiser), 4, 5, 6, 7, 8
1. Navegar para a página de reservas
2. Apresentar informação sobre as reservas
3. Selecionar por reserva específica
4. Clicar no botão de "See Reservation" da reserva específica
5. Apresentar a página da reserva específicada
6. Clicar no botão de "Check Out"
7. Pedir confirmação da ação
8. Clicar no botão "OK"
Plano 3: fazer 3.1, 3.2
3.1 Escrever nome do hóspede
3.2 Apresentar a reserva específicada


Utilizador
Sistema
0. Cancelar uma reserva
Plano 0: fazer 1, 2, 3, 4, 5
1. Navegar para a página do perfil
2. Apresentar informação sobre as reservas do hóspede
3. Clicar no botão "Cancel" da reserva que pretende cancelar
4. Pedir confirmação da ação
5. Clicar no botão "Yes"


SicaVibe
<<ORM Persistable>>
Utilizador
-ID : int
-email : String
-password : String
-salt : String
-nome : String
-dataNascimento : Date
-nTelemovel : String
-morada : String
-cc : String
-nif : String
<<ORM Persistable>>
Hospede
<<ORM Persistable>>
Administrador
<<ORM Persistable>>
Funcionario
-estado : String
+getMyWorkHotel() : Hotel
-hospede
1
-listaFuncionarios
0..*
1
-idFuncionario : int
<<ORM Persistable>>
Hotel
-ID : int
-nome : String
-descricao : String
-endereco : String
+getTiposDeQuarto() : Set<TipoDeQuarto>
+getQuartosDisponiveis(dataEntrada : Date, dataSaida : Date) : List<Quartos>
+checkDisponibilidade(dataEntrada : Date, dataSaida : Date) : Map<Integer, Integer>
+marcaReserva(hospedeID : int, tiposDeQuartosDesejados : Map<Integer, Integer>, servicosExtra : List<Integer>, dataEntrada : Date, dataSaida : Date) : float
<<ORM Persistable>>
ServicoExtra
-ID : int
-nome : String
-preco : float
-listaServicosExtra
0..*
1..*
-idServicoExtra : int
0..*
-servicosExtras
1
-idReserva : int
0..*
0..*
-listaReservas
<<ORM Persistable>>
Reserva
-ID : int
-dataEntrada : Date
-dataSaida : Date
-dataCheckIn : TimeStamp
-dataCheckout : TimeStamp
-preco : float
-estado : String
0..*
-idQuarto : int
1
0..*
-listaQuartos
<<ORM Persistable>>
Quarto
-ID : int
-nPorta : int
-estado : String
0..*
-quartos
1..*
1..*
-tipoDeQuarto
1
<<ORM Persistable>>
TipoDeQuarto
-ID : int
-nome : String
-capacidade : int
-preco : float
-descricao : String
1..*
-img
1
<<ORM Persistable>>
Imagem
-ID : int
-nome : String
-data : Blob
1..*
1
-img

Utilizador
ID INT
HotelID INT
Email VARCHAR(255)
Password VARCHAR(255)
Salt VARCHAR(255)
Nome VARCHAR(255)
DataNascimento DATE
NTelemovel VARCHAR(255)
Morada VARCHAR(255)
Cc VARCHAR(255)
Nif VARCHAR(255)
Estado ENUM(...)
Discriminator VARCHAR(255)
Indexes
Imagem
ID INT
Nome VARCHAR(255)
Data MEDIUMBLOB
Indexes
TipoDeQuarto
ID INT
ImagemID INT
Nome VARCHAR(255)
Capacidade INT
Preco FLOAT
Descricao VARCHAR(1024)
Indexes
Hotel_ServicoExtra
HotelID INT
ServicoExtraID INT
Indexes
Hotel
ID INT
ImagemID INT
Nome VARCHAR(255)
Descricao VARCHAR(1024)
Endereco VARCHAR(255)
Indexes
Quarto
ID INT
TipoDeQuartoID INT
HotelID INT
NPorta INT
Estado ENUM(...)
Indexes
ServicoExtra
ID INT
Nome VARCHAR(255)
Preco FLOAT
Indexes
Reserva_ServicoExtra
ReservaID INT
ServicoExtraID INT
Indexes
Reserva
ID INT
UtilizadorID INT
HotelID INT
DataEntrada DATE
DataSaida DATE
DataCheckIn TIMESTAMP
DataCheckout TIMESTAMP
Preco FLOAT
Estado ENUM(...)
Indexes
Reserva_Quarto
ReservaID INT
QuartoID INT
Indexes

```
# Criar VM com as seguintes caracteristicas:
# Tipo: E2
# SO: Linux Ubuntu 22.04 LTS x86/64
# Disco: SSD 500GB
# Memoria: 16GB
# vCPU: 8
# Zona: us-central1-a

# NOTA: DEVEM ESTAR LOGADOS NA CONTA DO GOOGLE CLOUD
# CASO NÃO ESTEJAM, DEVEM EXECUTAR O COMANDO:
# £ gcloud auth login

# NOTA: TAMBÉM É NECESSÁRIO TER A CHAVE SSH ADICIONADA NO COMPUTE ENGINE

$vmName = "sicavibe-app"
$imageFamily = "ubuntu-2204-lts"
$imageProject = "ubuntu-os-cloud"
$bootDiskSize = "64GB"
$bootDiskType = "pd-ssd"
$customMemory = "16GB"
$customCpu = "8"
$zone = "us-central1-a"
$scopes = "https://www.googleapis.com/auth/cloud-platform"
$network = "address=34.28.141.169,network-tier=PREMIUM,stack-type=IPV4_ONLY,
    subnet=default"

# Criação da VM
Write-Host -NoNewline "Creating VM..."
gcloud compute instances create $vmName `
    --image-family $imageFamily `
    --image-project $imageProject `
    --boot-disk-size $bootDiskSize `
    --boot-disk-type $bootDiskType `
    --custom-memory $customMemory `
    --custom-cpu $customCpu `
    --zone $zone `
    --scopes $scopes `
    --network-interface $network
Write-Host "done."

# Get vm's ip
Write-Host -NoNewline "Getting VM's IP..."
$vmIP = gcloud compute instances describe $vmName --format="get(networkInterfaces[0]
    .accessConfigs[0].natIP)"
Write-Host "$vmIP"

# Wait for the instance to be in the "RUNNING" state
```

```
Write-Host "Waiting for the instance to start..."


while ($true) {
    #gcloud compute ssh £vmName --command="echo instance now up"
        --strict-host-key-checking=no 2>$null
    $sshOutput = ssh -o "StrictHostKeyChecking=no" $vmIP -C
        "echo VM is up" #2>£null

    if ($LASTEXITCODE -eq 0) {
        break
    }

    Write-Host "($LASTEXITCODE) Instance not up yet, waiting 5 seconds..."
    Start-Sleep -Seconds 5
}

$sshInstallBasics = @"
sudo apt-get update
echo 'Sleeping for 1s'
sleep 1
sudo apt-get update
echo 'Sleeping for 6s'
sleep 6
sudo apt-get install openjdk-17-jre maven mysql-server -y
"@

$sshSetupGitHubUp = @"
gcloud secrets versions access 1 --secret="id_github" > ~/.ssh/id_rsa
sudo chmod 600 .ssh/id_rsa
gcloud secrets versions access 1 --secret="id_github_pub" > ./.ssh/id_rsa.pub
ssh-keyscan -t rsa github.com >> ~/.ssh/known_hosts
git config --global core.sshCommand 'ssh -o UserKnownHostsFile=/dev/null
    -o StrictHostKeyChecking=no'
git clone git@github.com:VicShadow/SicaVibe.git
"@

$sshSetupBD = @"
sudo mkdir /var/lib/mysql-files/SicaVibeImgs
sudo cp ~/SicaVibe/SicaVibeApp/scripts/imgs/* /var/lib/mysql-files/SicaVibeImgs/
sudo mysql < ~/SicaVibe/SicaVibeApp/scripts/DBSicaVibeCreate.ddl
sudo mysql < ~/SicaVibe/SicaVibeApp/scripts/DBUserCreate.sql
sudo mysql < ~/SicaVibe/SicaVibeApp/scripts/DBPopulate.sql
"@

$sshSetupMaven = @"
mvn clean package -f ~/SicaVibe/SicaVibeApp/pom.xml
```

```
mv ~/SicaVibe/SicaVibeApp/target/SicaVibeApp-0.0.1-SNAPSHOT.jar ./SicaVibeApp.jar
mv ~/SicaVibe/SicaVibeApp/scripts/startApp.sh .
sudo chmod +x startApp.sh
"@

$sshSetupNPM = @"
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install nodejs -y
echo 'Sleeping for 6s'
sleep 6
sudo npm install --global yarn
yarn --cwd ~/SicaVibe/SicaVibeWeb/
echo 'VITE_BACKEND_IP=$vmIP
VITE_BACKEND_PORT=8080
VITE_BACKEND_URL=http://`$VITE_BACKEND_IP:`$VITE_BACKEND_PORT' >
    ~/SicaVibe/SicaVibeWeb/.env
"@

$sshStartSicaVibe = @"
yarn --cwd ~/SicaVibe/SicaVibeWeb/ dev --host & ./startApp.sh
"@



Write-Host "Setting up VM..."
ssh -o "StrictHostKeyChecking=no" $vmIP -C $sshInstallBasics
ssh -o "StrictHostKeyChecking=no" $vmIP -C $sshSetupGitHubUp
ssh -o "StrictHostKeyChecking=no" $vmIP -C $sshSetupBD
ssh -o "StrictHostKeyChecking=no" $vmIP -C $sshSetupMaven
ssh -o "StrictHostKeyChecking=no" $vmIP -C $sshSetupNPM
Write-Host "Done. Starting App..."
ssh -o "StrictHostKeyChecking=no" $vmIP -C $sshStartSicaVibe
```