

## App Install & Go

Roberto Filipe Manso Barreto  
(nrº 21123, regime diurno)

Orientação de  
Luís Gonzaga Martins Ferreira

LICENCIATURA EM ENGENHARIA EM SISTEMAS INFORMÁTICOS  
ESCOLA SUPERIOR DE TECNOLOGIA  
INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE

## **Identificação do aluno**

Roberto Filipe Manso Barreto  
Aluno número 21123, regime diurno  
Licenciatura em Engenharia em Sistemas Informáticos

## **Orientação**

Luís Gonzaga Martins Ferreira

## **Informação sobre o Estágio**

Motorline Eletrocelos S.A  
Travessa do Sobreiro, 29 Rio Côvo (Sta. Eugénia) 4755-474 Barcelos  
Eng. Helder Remelhe

## Resumo

Este documento trata o processo de análise, especificação e implementação da solução Install&Go. Esta vem resolver o problema de comunicação entre a empresa Motorline e os seus técnicos, dado que, na eventualidade de um problema estes deverão telefonar para a empresa, o que gera sobrecarga do sistema.

Para a resolução deste problema foi desenvolvido uma plataforma de fórum, onde as empresas podem registar os seus técnicos. Aqui, podem expor questões, onde técnicos externos de outras empresas, e/ou técnicos Motorline poderão prestar auxílio. Se um técnico possuir um problema já resolvido, este poderá pesquisar pela solução dada no fórum.

O desenvolvimento desta solução proveu a aquisição de novas capacidades tecnológicas como o desenvolvimento *cross-platform* e as suas *frameworks*, sendo que destas foi explorado o *Flutter*. Através da elaboração desta aplicação foi possível, para além das competências mencionadas anteriormente, assimilar capacidades como análise, especificação de projetos e comunicação com clientes. Por fim, foi desenvolvida completamente a solução conforme as necessidades e expectativas do cliente.



## Abstract

This document describes the analysis, specification and development process of the Install&Go solution. This solution solves the communication problem between Motorline and their professionals, since that, in the event of a problem, they must call the company, which generates an overload.

To solve this problem a forum platform was developed, where companies can register their professionals. Here they can submit questions, so that other professionals from other companies and/or Motorline can provide assistance. If a professional has a problem that has already been solved, he can search for the solution in the forum

The development of this solution provided the acquisition of new technological skills such as cross-platform development and its frameworks, of which Flutter was explored. Through the elaboration of this application it was possible, besides the competences previously mentioned, to assimilate skills such as analysis, project specification and communication with clients. At the end, the solution was completely developed according to the client's needs and expectations.



## Agradecimentos

Em primeiro lugar, gostaria de agradecer à minha família, com destaque à minha mãe, ao meu padrinho, aos meus avós e à minha namorada, que com todo o carinho orientaram-me nesta caminhada.

Também, gostaria de salientar um caloroso agradecimento à empresa Motorline, uma vez que, fizeram-me sempre sentir um membro da equipa, com destaque ao supervisor Helder Remelhe que sempre esteve disponível para eventuais dúvidas que surgissem no desenvolvimento do projeto.

Por fim, mas não menos importante, gostaria de enfatizar toda a orientação, disponibilidade, conversa e ensinamentos proporcionados pelo professor doutor Luís Ferreira e também aos meus colegas de curso Henrique Cartucho e João Castro que mais que colegas, são amigos para a vida.



# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Objetivos . . . . .	2
1.2	Contexto . . . . .	2
1.3	Planificação do trabalho . . . . .	3
1.4	Estrutura do documento . . . . .	4
<b>2</b>	<b>Estado da arte</b>	<b>5</b>
2.1	Ferramentas de trabalho utilizadas . . . . .	5
2.2	Plataforma Tecnológica . . . . .	6
2.2.1	Web Scraper . . . . .	6
2.2.1.1	Selenium . . . . .	6
2.2.2	Serviços Backend . . . . .	7
2.2.2.1	Serviços RestFull e SOAP . . . . .	7
2.2.2.2	NodeJS . . . . .	8
2.2.2.3	TypeScript . . . . .	8
2.2.2.4	PostgreSQL . . . . .	9
2.2.2.5	Logs e Logging . . . . .	9
2.2.2.6	Morgan . . . . .	9
2.2.2.7	Gestão de <i>emails</i> . . . . .	10
2.2.2.8	Agendamento de tarefas . . . . .	10
2.2.2.9	Encriptação de <i>passwords</i> . . . . .	10
2.2.2.10	Encriptação de configurações do servidor . . . . .	11
2.2.2.11	Firebase . . . . .	12
2.2.2.12	Axios . . . . .	12
2.2.3	<i>Frontend</i> . . . . .	13
2.2.3.1	Desenvolvimento cross-platform . . . . .	13
2.2.3.2	Flutter . . . . .	14
2.2.3.3	Dart . . . . .	14

2.2.3.4	Links de aplicações . . . . .	15
2.2.4	Qualidade de código . . . . .	16
2.2.4.1	Design patterns . . . . .	16
2.2.4.2	Documentação . . . . .	16
2.2.4.3	Typedoc . . . . .	16
2.2.4.4	Swagger . . . . .	17
2.2.4.5	Testes de código . . . . .	17
<b>3</b>	<b>Análise e especificação</b>	<b>19</b>
3.1	Modelo de negócio . . . . .	19
3.2	Objetivos de negócio . . . . .	19
3.3	Estudo de Soluções existentes . . . . .	20
3.4	Domínio de aplicação do sistema . . . . .	20
3.5	Operações a realizar no sistema . . . . .	21
3.6	Descrição dos intervenientes . . . . .	24
3.7	Partes Interessadas . . . . .	24
3.8	Condições Específicas . . . . .	24
3.9	Esquematização do conteúdo das páginas . . . . .	25
3.9.1	Autenticação e página Inicial . . . . .	26
3.9.2	Fórum . . . . .	27
3.9.3	Criar nova tópico . . . . .	27
3.9.4	Detalhes de tópicos . . . . .	28
3.9.5	Pesquisa de tópicos . . . . .	28
3.9.6	Notificações . . . . .	29
3.9.7	Perfil . . . . .	29
3.9.8	Gestão de recursos humanos . . . . .	30
3.10	<i>User Stories</i> . . . . .	31
3.11	Casos de uso . . . . .	34
3.11.1	Especificação de casos de uso . . . . .	36
3.11.1.1	Especificação de caso de uso de listagem de tópicos . . . . .	36
3.11.1.2	Especificação de caso de uso de criar novo tópico . . . . .	37
3.11.1.3	Especificação de caso de uso de pesquisar tópicos por escrito	38
3.11.1.4	Especificação de caso de uso de ver finalizar tópico . . . . .	38
3.11.1.5	Especificação de caso de uso de selecionar melhor resposta	39
3.11.1.6	Especificação de caso de uso de eliminar tópico . . . . .	40

3.11.1.7 Especificação de caso de uso de alterar visibilidade de um tópico . . . . .	40
3.11.1.8 Especificação de caso de uso gostar de um tópico . . . . .	41
3.11.1.9 Especificação de caso de uso gostar de um comentário . . . . .	41
3.11.1.10 Especificação de caso de uso de comentar tópico . . . . .	42
3.11.1.11 Especificação de caso de uso ativar conta . . . . .	43
3.11.1.12 Especificação de caso de uso configurar notificações . . . . .	44
3.11.1.13 Especificação de caso de uso registar técnico . . . . .	44
3.11.2 Diagramas de casos de uso . . . . .	45
3.11.2.1 Casos de uso Fórum . . . . .	45
3.11.2.2 Casos de uso de pesquisar tópicos . . . . .	45
3.11.2.3 Casos de uso ver detalhes de tópico . . . . .	46
3.11.2.4 Casos de uso ver comentários . . . . .	46
3.11.2.5 Casos de uso ativação de conta . . . . .	47
3.11.2.6 Casos de uso perfil . . . . .	47
3.11.2.7 Casos de uso notificações . . . . .	48
3.11.2.8 Casos de uso gestão de recursos humanos . . . . .	48
3.12 Diagrama Entidade Relação . . . . .	49
3.12.1 Escolha de diagrama de entidade relação . . . . .	50
3.12.2 Dicionário de termos . . . . .	51
3.13 Diagrama de Classes . . . . .	55
3.14 Mockups . . . . .	56
3.14.1 Página Inicial . . . . .	56
3.14.2 Autenticação - Login e Registo . . . . .	57
3.14.3 Autenticação - Ativação e Confirmação de conta . . . . .	58
3.14.4 Página inicial fórum . . . . .	59
3.14.5 Página de detalhes de um tópico . . . . .	60
3.14.6 Página de criação de um tópico . . . . .	61
3.14.7 Página de notificações . . . . .	62
3.14.8 Página de perfil de utilizador . . . . .	63
3.14.9 Página de gestão de recursos humanos . . . . .	64
3.14.10 Página de perfil de técnico registado . . . . .	65
3.14.11 Página de registo de novo técnico . . . . .	66
3.15 Diagramas de atividades . . . . .	67
3.15.1 Diagrama de atividades página inicial . . . . .	67
3.15.2 Diagrama de atividades página de perfil . . . . .	67

3.15.3 Diagrama de atividades página inicial do fórum . . . . .	68
3.15.4 Diagrama de atividades página de criação de tópico . . . . .	69
3.15.5 Diagrama de atividades página de detalhes do tópico . . . . .	70
3.15.6 Diagrama de atividades páginas de autenticação . . . . .	71
3.15.7 Diagrama de atividades registar técnico . . . . .	72
3.15.8 Diagrama de atividades confirmar conta . . . . .	72
3.16 Diagramas de estados . . . . .	73
3.16.1 Diagrama de estados criação de tópico . . . . .	73
3.16.2 Diagrama de estados responder a tópico . . . . .	74
3.16.3 Diagrama de estados autenticação e validação de conta . . . . .	75
3.17 Diagrama de sequência . . . . .	76
3.17.1 Diagrama de sequência Login e ativação de conta . . . . .	76
3.17.2 Diagrama de sequência Registo e ativação de conta . . . . .	77
3.17.3 Diagrama de sequência registo de técnicos . . . . .	78
3.18 Arquitetura de sistema . . . . .	79
3.18.1 Arquitetura de funcional . . . . .	79
3.18.2 Arquitetura de componentes . . . . .	80
3.18.3 Tabela de endpoints . . . . .	81
<b>4 Trabalho desenvolvido</b>	<b>83</b>
4.1 Web scraper . . . . .	83
4.1.1 Implementação web scraper . . . . .	83
4.1.1.1 Implementação no website . . . . .	88
4.1.1.2 Melhoria de implementação . . . . .	89
4.1.1.3 Diagrama de base de dados final . . . . .	92
4.1.1.4 Armazenamento de dados . . . . .	92
4.2 Backend . . . . .	93
4.2.1 Organização do projeto . . . . .	93
4.2.2 Definição de rotas base . . . . .	94
4.2.3 Middlewares . . . . .	94
4.2.3.1 Idioma de Comunicação . . . . .	94
4.2.3.2 Autenticação . . . . .	95
4.2.3.3 Validação de <i>Role</i> . . . . .	96
4.2.3.4 Logging de erros . . . . .	96
4.2.3.5 Logging de pedidos com Morgan . . . . .	96
4.2.4 Controllers . . . . .	97

4.2.4.1	Estruturação dos controllers . . . . .	97
4.2.4.2	Execução da lógica de negócio . . . . .	97
4.2.4.3	Validação dos dados . . . . .	98
4.2.4.4	Formulação da resposta . . . . .	98
4.2.4.5	Processamento de erros . . . . .	98
4.2.5	Envio de <i>emails</i> . . . . .	99
4.2.6	Agendamento de tarefas . . . . .	99
4.2.7	Cifra de <i>passwords</i> . . . . .	100
4.2.7.1	Cifra de configurações do servidor . . . . .	100
4.2.8	Documentação Typedoc . . . . .	101
4.2.9	Documentação Swagger . . . . .	102
4.3	Frontend . . . . .	103
4.3.1	Organização do projeto . . . . .	103
4.3.2	Processo de aprendizagem . . . . .	104
4.3.3	Extensions . . . . .	104
4.3.4	Handlers . . . . .	104
4.3.5	Providers . . . . .	105
4.3.6	Helpers . . . . .	105
4.3.7	Gestão de utilizadores . . . . .	106
4.3.8	Fórum . . . . .	107
4.3.8.1	Filtragem de tópicos . . . . .	108
4.3.8.2	Carregamento de tópicos . . . . .	109
4.3.8.3	Detalhes de tópico . . . . .	110
4.3.9	Firestorage . . . . .	111
4.3.10	Apresentação de Imagens . . . . .	112
4.3.11	Apresentação de Imagens em carrossel . . . . .	113
4.3.12	Carregamento de Imagens . . . . .	113
4.3.13	Vídeos . . . . .	114
4.3.14	Links . . . . .	114
4.3.15	Notificações . . . . .	115
4.3.16	Permissões . . . . .	116
4.3.17	Ios . . . . .	117
<b>5</b>	<b>Análise de resultados</b>	<b>119</b>
5.1	Tarefas . . . . .	119
5.2	Base de dados . . . . .	120

5.3 Aplicação final . . . . .	121
5.4 Opinião do cliente . . . . .	121
<b>6 Conclusão e futuras implementações</b>	<b>123</b>
<b>7 Anexos</b>	<b>125</b>

# **Lista de Figuras**

1.1	Planificação de <i>sprints</i> . . . . .	3
2.1	Funcionamento dos <i>Dynamic Links</i> em <i>Android</i> (Firebase, 2023b) . . . . .	15
3.1	Diagrama de contexto da aplicação . . . . .	20
3.2	Esquema de organização de páginas do <i>software</i> . . . . .	25
3.3	Esquema de organização das páginas de autenticação e página inicial . . .	26
3.4	Esquema de organização da página de fórum . . . . .	27
3.5	Esquema de organização da página de criação de tópicos . . . . .	27
3.6	Esquema de organização da página de detalhes de tópico . . . . .	28
3.7	Esquema de organização da página de pesquisa de tópicos . . . . .	28
3.8	Esquema de organização da página de notificações . . . . .	29
3.9	Esquema de organização da página de perfil . . . . .	29
3.10	Esquema de organização da página de gestão de recursos humanos . . . .	30
3.11	Diagrama de casos de uso de fórum . . . . .	45
3.12	Diagrama de casos de uso de pesquisa de tópicos . . . . .	45
3.13	Diagrama de casos de uso de detalhes de tópico . . . . .	46
3.14	Diagrama de casos de uso de ver comentários . . . . .	46
3.15	Diagrama de casos de uso de ativação de conta . . . . .	47
3.16	Diagrama de casos de uso de perfil . . . . .	47
3.17	Diagrama de casos de notificações . . . . .	48
3.18	Diagrama de casos de uso de recursos humanos . . . . .	48
3.19	Diagrama Entidade Relação base de dados <i>Install&amp;Go</i> . . . . .	49
3.20	Diagrama Entidade Relação alternativo . . . . .	50
3.21	Diagrama de classes <i>Install&amp;Go</i> . . . . .	55
3.22	Página inicial do fórum . . . . .	56
3.23	Autenticação - Login e Registo . . . . .	57
3.24	Autenticação - Ativação e Confirmação de conta . . . . .	58
3.25	Página inicial do fórum . . . . .	59

3.26	Página de detalhes de tópico do software . . . . .	60
3.27	Página de criação de tópico . . . . .	61
3.28	Página de notificações . . . . .	62
3.29	Página de perfil de utilizador . . . . .	63
3.30	Página de gestão de recursos humanos . . . . .	64
3.31	Página de perfil de técnico registado . . . . .	65
3.32	Página de registo de novo técnico . . . . .	66
3.33	Diagrama de atividades de página inicial da aplicação . . . . .	67
3.34	Diagrama de atividades de página de perfil . . . . .	67
3.35	Diagrama de atividades de página inicial do fórum . . . . .	68
3.36	Diagrama de atividades de página de criação de tópico . . . . .	69
3.37	Diagrama de atividades de página de detalhes do tópico . . . . .	70
3.38	Diagrama de atividades de página de validação de conta . . . . .	71
3.39	Diagrama de atividades de página de registar técnico . . . . .	72
3.40	Diagrama de atividades de página de confirmar conta de técnico . . . . .	72
3.41	Diagrama de estados de criar tópico . . . . .	73
3.42	Diagrama de estados de criar tópico . . . . .	74
3.43	Diagrama de estados de autenticação e validação de conta . . . . .	75
3.44	Diagrama de sequência de login e ativação de conta . . . . .	76
3.45	Diagrama de sequência de registo e validação de conta . . . . .	77
3.46	Diagrama de sequência de registo de técnicos . . . . .	78
3.47	Arquitetura do sistema . . . . .	79
3.48	Arquitetura do funcional . . . . .	79
3.49	Arquitetura de componentes . . . . .	80
3.50	Listagem de serviços documentados . . . . .	82
4.1	Estrutura dos dados obtidos . . . . .	84
4.2	Página geral de produtos . . . . .	85
4.3	Página de produtos de uma subcategoria . . . . .	85
4.4	Página de produtos de uma subcategoria distinta . . . . .	86
4.5	Página de detalhes de produto, secção inicial . . . . .	86
4.6	Página de detalhes de produto, secção de informações . . . . .	87
4.7	Página de detalhes de produto, secção de videos . . . . .	87
4.8	Resposta obtida aquando o pedido ao <i>url</i> da página <i>web</i> . . . . .	88
4.9	Urls com erro primeira interação . . . . .	89
4.10	Exemplo de página de acessórios . . . . .	89

4.11	Indicação das páginas com falha . . . . .	90
4.12	Exemplo de página de produto incomum . . . . .	90
4.13	Exemplo de página de produto com subprodutos . . . . .	91
4.14	Exemplo de página de serviço . . . . .	91
4.15	atualização da base de dados . . . . .	92
4.16	Exemplo de página de produto incomum . . . . .	93
4.17	Comportamento de middlewares . . . . .	94
4.18	Utilização de tokens . . . . .	95
4.19	Autenticação - Login e Registo . . . . .	98
4.20	Documentação <i>typedoc</i> . . . . .	101
4.21	Documentação de classe <i>typedoc</i> . . . . .	101
4.22	Documentação swagger . . . . .	102
4.23	Exemplo de documentação de serviço . . . . .	102
4.24	Organização do projeto . . . . .	103
4.26	Aviso de login a conta sem acesso . . . . .	106
4.28	Filtragem do forum . . . . .	108
4.29	Carregamento de tópicos . . . . .	109
4.30	Destaque de mensagens . . . . .	110
5.1	Organização de tarefas final . . . . .	119
5.2	Base de dados inicial . . . . .	120
5.3	Base de dados final . . . . .	120



# Listas de Tabelas

3.1	Tabela de requisitos funcionais . . . . .	21
3.2	Tabela de requisitos não funcionais . . . . .	24
3.3	Tabela de <i>user stories</i> . . . . .	31
3.4	Tabela de casos de uso . . . . .	34
3.5	Tabela de especificação de caso de uso de listagem de tópicos do técnico . .	36
3.6	Tabela de especificação de caso de uso login . . . . .	37
3.7	Tabela de especificação de caso de uso de pesquisa por escrito . . . . .	38
3.8	Tabela de especificação de caso de uso de finalizar tópico . . . . .	38
3.9	Tabela de especificação de caso de uso de selecionar melhor resposta . . . .	39
3.10	Tabela de especificação do caso de uso de eliminar tópico . . . . .	40
3.11	Tabela de especificação de caso de uso de alteração de visibilidade de um tópico . . . . .	40
3.12	Tabela de especificação de caso de uso de gostar de um tópico . . . . .	41
3.13	Tabela de especificação de caso de uso de gostar de comentário . . . . .	41
3.14	Tabela de especificação de caso de uso de comentar um tópico . . . . .	42
3.15	Tabela de especificação de caso de uso ativação de conta . . . . .	43
3.16	Tabela de especificação de caso de uso de configuração de notificações . .	44
3.17	Tabela de especificação de caso de uso de registar técnico . . . . .	44
3.18	Dicionário de termos da base de dados . . . . .	51
3.19	Tabela de endpoints . . . . .	81



# Siglas & Acrónimos

**API** Application Programming Interface. 7, 79, 95, 105, 109

**JSON** JavaScript Object Notation. 7, 17, 84, 89, 95, 98, 102, 115

**MVC** Model View Controller. 16, 93, 103

**RF** Requisitos Funcionais. 21

**RNF** Requisitos Não Funcionais. 21

**UC** Use Cases. 34

**US** User Stories. 31



# 1. Introdução

Este projeto intitulado *Install & Go*, refere-se a uma aplicação para *smartphone* direcionada a todos os clientes e técnicos Motorline, com o propósito de agilizar todo o processo de resolução de problemas e de acesso aos produtos. Para alcançar estes objetivos, a aplicação conta com um catálogo para todos os utilizadores e também com um fórum para os clientes e técnicos Motorline.

No início de estágio foi indicado pelo supervisor da empresa a subdivisão do projeto em duas grandes componentes. Também este projeto foi desenvolvido com uma colega que ficou encarregue de desenvolver o *frontend* da componente do catálogo. Já o desenvolvimento completo do *backend*, aquisição do catálogo de produtos do *website* da empresa e o *frontend* para o fórum, gestão de utilizadores e autenticação ficaram a cargo do autor deste documento.

O fórum da aplicação é uma plataforma que permite aos clientes e técnicos criar tópicos para realizar questões e/ou expor problemas para a comunidade. Estes tópicos podem ser comentados, onde também é possível realizar uma comunicação.

O suporte *backend* da aplicação foi realizado através de um conjunto de serviço, sendo inicialmente apenas necessário para fórum. Contudo, foi posteriormente acrescentado suporte para o catálogo de produtos.

## 1.1 Objetivos

A plataforma do fórum da aplicação deverá permitir que a comunidade partilhe os seus tópicos. Para isso, o utilizador necessitará conseguir criar estes com a indicação do problema e uma descrição do mesmo, com a possibilidade de anexar imagens, assim como referenciar o produto, para facilitar a identificação e resolução do problema.

A comunidade deverá também ter a possibilidade de comentar os tópicos e comunicar nesta secção. O responsável pelo tópico necessitará ter a possibilidade de o colocar em privado caso tenha como objetivo que apenas técnicos Motorline respondam ao mesmo. Este também deverá ter a seu dispor a possibilidade de indicar quando o tópico estiver finalizado e qual a melhor resposta que obteve. Os utilizadores deverão também conseguir gostar de tópicos e comentários para destacar os mesmos perante a restante comunidade.

A comunidade deverá conseguir ver os tópicos em destaque, os mais recentes, os seus tópicos, os que não estão finalizados e os técnicos Motorline necessitarão de conseguir ver os tópicos privados existentes. A comunidade deverá ter a possibilidade de pesquisar por tópicos com assuntos específicos, pesquisa por nome e por produto. Para filtragem de pesquisa a comunidade deverá também conseguir selecionar o tipo de tópico.

## 1.2 Contexto

O projeto foi desenvolvido na empresa Motorline Eletrocelos S.A daqui em diante designada Motorline, esta empresa é especializada na produção e comercialização de automatismos para portas e portões, sistemas de controlo de acessos, sistemas de segurança, entre outros produtos relacionados com o setor da automação.

Este projeto vem por este meio resolver o problema de comunicação com o cliente, que atualmente para solucionar as suas questões tem de contactar a assistência técnica que por vezes pode estar sobrecarregada, pelo que deverão preencher um formulário para expor a sua questão.

Com esta solução os clientes e técnicos Motorline poderão procurar ou expor os seus problemas com a comunidade onde têm a possibilidade de ser respondidos, o que torna o processo de resolução de problemas mais ágil.

## 1.3 Planificação do trabalho

Para obter uma visão geral do projeto e uma previsão de finalização foi realizada uma planificação expectável de tarefas.



Figura 1.1: Planificação de *sprints*

## 1.4 Estrutura do documento

Este documento contém a descrição de todo o processo de engenharia de *software*, desenvolvimento e pensamento sobre o problema em mãos, sendo estes pontos divididos sobre diversos tópicos:

1. Estado da Arte, onde é abordado as ferramentas utilizadas e tecnologias exploradas.
2. Análise e especificação, onde é abordado o estado da arte, descrito o modelo de negócio e apresentada toda a engenharia de *software*
3. Trabalho desenvolvido, onde é descrito todo o processo de desenvolvimento do projeto.
4. Análise de resultados, onde é discutido os resultados obtidos.
5. Conclusão e trabalho futuro, onde é abordada a conclusão sobre o projeto e também futuras implementações que poderiam ser realizadas.

## 2. Estado da arte

Para a organização de todo o trabalho a desenvolver, visto que este é dividido com mais uma colega, foi utilizada a técnica de desenvolvimento ágil, através da qual foi possível organizar todas as tarefas entre os elementos de desenvolvimento do projeto.

### 2.1 Ferramentas de trabalho utilizadas

Este documento foi desenvolvido com a ferramenta *LaTeX*, uma vez que, permite a aplicação de um conjunto de regras pré-definidas, o que evita a preocupação com estas durante a escrita. Para o auxílio com referências bibliográficas foi utilizado o *Mendeley*, visto que, fornece um conjunto de funcionalidades de ajudas na investigação, assim como para importação para *LaTeX* em formato *Bibtex*.

A organização de todas as tarefas, foi realizada na ferramenta *Github Projects*. Esta dispõe de funções que permitem ligar um projeto a um repositório de *Github*, onde se consegue personalizar completamente todo o projeto e parâmetros das tarefas que resulta numa organização minuciosa.

O *Microsoft Excel* foi utilizado para a engenharia de *software* onde foram descritos os requisitos do projeto, *user stories* e também a especificação de casos de uso. Esta ferramenta também foi utilizada para a organização de reuniões com o cliente e redação de tópicos a abordar.

Para o desenvolvimento do *design* do *software* foi utilizada a ferramenta *Figma*, que permite o *design* de todas as componentes tendo em conta as reais dimensões de um dispositivo. Esta ferramenta dispõe de funções para criar apresentações interativas que conseguem demonstrar o comportamento da aplicação como resultado final, dando também suporte à implementação.

O *Draw.io* foi utilizado para os desenhos das arquiteturas do projeto tendo revelado grande auxílio, uma vez que, permite uma grande liberdade ilustrativa. Esta ferramenta permite conectar com o *github* o que proporciona a facilidade de guardar projetos e ter acesso a partir de qualquer dispositivo.

A engenharia de *software* foi realizada através da utilização *Visual Studio Paradigm*, esta é uma ferramenta muito completa que contém modelos e regras para a engenharia de *software*. Esta tornou-se um grande recurso no desenvolvimento da base de dados, dado que é possível desenhar o modelo e exportar para um ficheiro de criação de base de dados.

## 2.2 Plataforma Tecnológica

### 2.2.1 Web Scraper

*Web scraping* é terminologia dada à "(...)construction of an agent to download, parse, and organize data from the web in an automated manner(...)"(vanden Broucke & Baesens, 2018). O grande problema com *web scraping* é que poderá ser considerado ilegal e é facilmente detetado. Tendo em conta este problema surgiram duas grandes formas de realizar *web scraping*. A forma mais comum de *web scraping* é realizar um pedido para obter uma página web e ler esta, sendo assim um processo rápido e simples.

A segunda forma de realizar *web scraping* é através da simulação da ação humana com da abertura do navegador programáticamente, pesquisa pela página desejada, descarregar e daí ler os dados. Este torna-se um processo lento e complexo.

A grande diferença entre estas duas formas é a velocidade, visto que a segunda forma tem de esperar que o navegador inicie, de seguida terá de esperar que a página carregue e apenas após este processo se poderá ler os dados.

#### 2.2.1.1 Selenium

O *Selenium* é uma ferramenta "(...)for a range of tools and libraries that enable and support the automation of web browsers(...)"(Selenium, 2023), esta provém "(...)extensions to emulate user interaction with browsers(...)"(Selenium, 2023). Na sua base esta "(...)is WebDriver, an interface to write instruction sets that can be run interchangeably in many browsers(...)"(Selenium, 2023). Esta ferramenta provém também a possibilidade de escalar com *multi threading*, o que permite abrir diversas janelas do navegador simultaneamente e obter dados destas o que diminui drasticamente o tempo de execução, esta funcionalidade não foi explorada devido a limitações de *hardware*, mas seria uma importante implementação futura.

## 2.2.2 Serviços Backend

Para a realização da integração entre a aplicação *frontend* e os dados, foi necessário desenvolver uma *Application Programming Interface (API)* para dar suporte a todos os serviços necessários para a aplicação. Uma *API* "*(...)exposes a set of data and functions to facilitate interactions between computer programs and allow them to exchange information. (...)"* (Massee, 2011). Estas ferramentas apesar de serem "*(...)designed to work with other programs, they're mostly intended to be understood and used by humans writing those other programs(...)"*"(Jin et al., 2018).

### 2.2.2.1 Serviços RestFull e SOAP

Os serviços *RestFull* "*(...)expose data as resources and use standard HTTP methods to represent Create, Read, Update, and Delete (CRUD) transactions against these resources(...)"*(Jin et al., 2018), sendo que a resposta é no formato *JavaScript Object Notation (JSON)* "*(...)due to its simplicity and ease of use with JavaScript, JSON has become the standard for modern APIs(...)"*(Jin et al., 2018).

Já *SOAP* é "*(...)XML-based envelope for the information being transferred, and a set of rules for translating application and platform-specific data types into XML representations(...)"*(Snell et al., 2002) sendo que a resposta é realizada em *XML* leva a que "*(...)It relies heavily on XML standards like XML Schema and XML Namespaces for its definition and function(...)"*(Snell et al., 2002). A utilização de *XML* permite que "*(...)two applications, regardless of operating system, programming language, or any other technical implementation detail, may openly share information using nothing more than a simple message encoded in a way that both applications understand(...)"*(Snell et al., 2002).

Por fim foi decidido utilizar *RestFull* devido a ser "*(...)much lighter compared to SOAP. It does not require formats like headers to be included in the message, like it is required in SOAP architecture(...)"*(Halili & Ramadani, 2018). Outra maior valia é que "*(...)it parses JSON, a human readable language designed to allow data exchange and making it easier to parse and use by the computer. It is estimated to be at around one hundred times faster than XML(...)"*(Halili & Ramadani, 2018).

### 2.2.2.2 NodeJS

Para o desenvolvimento do projeto *backend* foi escolhido *NodeJS*, este surgiu quando "(...)the original developers took JavaScript, something you could usually only run inside the browser, and they let it run on your machine as a standalone process(...)"(Mead, 2018) isto significa que é possível correr código "(...)JavaScript outside the context of the browser(...)"(Mead, 2018) .

Para correr *JavaScript* a nível de servidor *web* e a nível de computador pessoal é utilizado o mesmo motor, "(...)it's called the V8 JavaScript runtime engine(...)"(Mead, 2018), este é "(...)an opensource engine that takes JavaScript code and compiles it into much faster machine code. And that's a big part of what makes Node.js so fast(...)"(Mead, 2018) .

*NodeJs* permite a utilização de bibliotecas externas "(...)by providing the Node Package Manager(...)"(Mead, 2018), este provém ao programador "(...)to easily install, manage, and even provide your own modules for a rapidly grown and well-maintained open source repository(...)"(Mead, 2018).

*NodeJS* foi escolhido para o *backend*, uma vez que, permite a utilização de *TypeScript* para o desenvolvimento e por ser vastamente utilizado, o que dá acesso a diversas fontes de informação para a resolução de problemas, assim como, para o auxílio ao desenvolvimento.

### 2.2.2.3 Typescript

O *TypeScript* "(...)is a bit unusual as a language in that it neither runs in an interpreter (as Python and Ruby do) nor compiles down to a lower-level language (as Java and C do).(...)"(Vanderkam, 2019) isto porque este "(...)compiles to another high-level language, JavaScript(...)"(Vanderkam, 2019), isto faz com que o *TypeScript* seja visto como "(...)a superset of JavaScript in a syntactic sense(...)"(Vanderkam, 2019).

Todos os programas *JavaScript* "(...)are TypeScript programs, but the converse is not true(...)"(Vanderkam, 2019), "(...)This is because TypeScript adds additional syntax for specifying types(...)"(Vanderkam, 2019). O sistema de tipagens do *TypeScript* tem como objetivo "(...)to detect code that will throw an exception at runtime, without having to run your code(...)"(Vanderkam, 2019). "(...)The type checker cannot always spot code that will throw exceptions, but it will try(...)"(Vanderkam, 2019).

Esta linguagem foi escolhida para o *backend*, visto que, assegura as tipagens, o que proporciona um maior nível de segurança quando se trabalha com dados recebidos, assim como, a agilização do processo de programação devido à capacidade de prever a maioria dos erros de código.

### 2.2.2.4 PostgreSQL

*PostgreSQL* "(...) is an open source object relational database management system (...)"(Juba et al., 2015). Esta "(...) emphasizes extensibility(...)"(Juba et al., 2015) o que permite a utilização de extensões desenvolvidas pela comunidade, mas "(...) Also, there are several extensions to access, manage, and monitor PostgreSQL clusters, such as pgAdmin III(...)"(Juba et al., 2015). Esta ferramenta é de aprendizagem simples devido ao facto de "(...) it complies with ANSI SQL standards(...)"(Juba et al., 2015), o que leva a que, qualquer indivíduo com conhecimentos prévios em *SQL* consiga facilmente aprender esta tecnologia.

*PostgreSQL* foi escolhido para a base de dados, dado que, a empresa já utiliza vastamente esta tecnologia, mas também porque é *open-source* e não existem custos associados para este tipo de utilização. A funcionalidade de extensões foi utilizada para implementar *id's*, que utilizam a estrutura *uuid*, o que dificulta o ataque aos dados, uma vez que, é difícil de prever os valores de *id's*.

### 2.2.2.5 Logs e Logging

*Logs* "(...) are a very useful source of information for computer system resource management (printers, disk systems, battery backup systems, operating systems, etc.), user and application management (login and logout, application access, etc.), and security(...)"(Chuvakin et al., 2012). Um Log é "(...) what a computer system, device, software, etc. generates in response to some sort of stimuli. What exactly the stimuli are greatly depends on the source of the log message(...)"(Chuvakin et al., 2012), ou seja, perante um estímulo desejado, um *log* poderá ser gerado. Os dados dos *logs* são "(...) the intrinsic meaning that a log message has(...)"(Chuvakin et al., 2012), o que significa que estes contêm apenas dados relevantes ao objetivo do *log*. *Logging* é o nome que se dá ao processo de geração de *logs*.

Neste contexto *logging* poderá ser utilizado para realizar a monitorização de pedidos e erros. Estas informações poderão até auxiliar na toma de decisões sobre o *software* e em quais funcionalidades colocar mais atenção.

### 2.2.2.6 Morgan

*Morgan* é uma biblioteca que permite extrair dados de um pedido, assim como a criação de *logs*. Este atua como um *middleware* do servidor, que recebe qual o tipo de *log* a ser escrito, estes estão definidos na biblioteca. Os principais dados obtidos por este são, a data e hora do pedido, o tipo, o serviço, os dados recebidos, a resposta devolvida e também a descrição do sistema utilizado. Com estes dados é possível saber que plataforma é mais utilizada no *software*, quais as horas de maior utilização e quais os serviços mais executados. Estes dados permitem direcionar mais recursos para uma indicada plataforma e/ou serviço, assim como escolher os melhores horários de manutenção dos servidores.

### 2.2.2.7 Gestão de *emails*

O envio de *emails* para os utilizadores, foi desenvolvido através da biblioteca *Node-mailer*, que permite a utilização de um servidor de *SMTP*. Esta ferramenta foi escolhida devido a ser uma das mais utilizadas para este tipo de necessidade, o que permite que exista mais informação sobre a mesma que auxilia a resolução e identificação de erros.

Para desenvolver o conteúdo dos *emails* foi utilizada a ferramenta *Tabular Email*, esta permite realizar o *design* do conteúdo de um *email*, sendo possível exportar para *html*. A maior dificuldade desta ferramenta é que não permite a utilização de acentuação, e visto que o *html* é gerado por uma máquina este torna-se complicado de navegar e traduzir.

### 2.2.2.8 Agendamento de tarefas

Um requisito deste projeto foi o envio diário de *emails* com o relatório de notificações ao final do dia. Primeiramente, para realizar o agendamento de tarefas, foi feita uma análise das ferramentas existentes para a realização deste tipo de ações. Deste modo, foram encontradas o *Cronetab* e o *Node-Cron*. A grande diferença entre estas duas ferramentas é que o *Cronetab* funciona a nível de servidor, sendo que, o funcionamento tem por base "*...run this command at this time on this date...*"(Linux, 2023), este comando poderá por exemplo executar um código para enviar *emails*. Já o *Node-Cron* trata-se de uma biblioteca de *NodeJs* "*...in pure JavaScript for node.js based on GNU crontab...*"(merencia, 2023), este permite o fácil agendamento de tarefas de forma programática, assim como a indicação do código a ser executado sem necessidade de criar comandos.

A hora de execução do código de envio de *emails* poderá variar e necessitar de reprogramação, pelo que, foi optada a utilização do *Node-Cron*, uma vez que, facilita a utilização e agiliza o processo de reprogramação de horas de envio do relatório de notificações.

### 2.2.2.9 Encriptação de *passwords*

Para garantir a segurança das *passwords* dos utilizadores é necessário encriptar estas, a encriptação poderá ser realizada manualmente ou com o auxílio de ferramentas, a grande diferença é que manualmente poderá não se obter uma encriptação tão segura como com o auxílio de uma ferramenta. Sendo assim, foi decidido utilizar uma ferramenta para encriptação de *passwords*, a ferramenta escolhida foi *Bcrypt*. Esta foi escolhida devido a ser vastamente utilizada e tem por base a *Hash Bcrypt*, esta "*...uses a variant of the Blowfish encryption algorithm's keying schedule, and introduces a work factor, which allows you to determine how expensive the Hash function will be...*"(Hale, 2023) isto permite que esta acompanhe a lei de *Moore*, pois "*...As computers get faster you can increase the work factor and the Hash will get slower...*"(Hale, 2023). Esta ferramenta oferece um conjunto de métodos dos quais foram utilizados os de *Hash* e de *compare*. O método de *Hash* permite através de um valor, chamado *salt*, que não indica a complexidade a aplicar sobre a encriptação, sendo de seguida devolvida a *password* encriptada. O método *compare* permite comparar uma *password* encriptada com uma *password* não encriptada, e devolve verdadeiro ou falso conforme as *passwords* sejam iguais ou não.

### 2.2.2.10 Encriptação de configurações do servidor

Com o objetivo de garantir um nível de segurança maior foram realizadas pesquisas sobre as principais falhas de segurança no *NodeJs*. Nestas, foi descoberto que as principais formas de ataque são as bibliotecas de *malware* e o ataque direto com o objetivo de obter dados de acesso a servidores que se encontram nos ficheiros de configuração.

Por norma, nas metodologias mais recentes de desenvolvimento de *software*, é sugerido que se coloque todas as configurações de servidores num ficheiro à parte, devido à praticidade de gerir estes dados, mas esta leva a um nível de segurança mais baixo, uma vez que, se alguém conseguir acesso a este ficheiro, consegue obter todos os dados de configuração de servidores. Neste projeto foi utilizado o ficheiro *env* para este fim, este no momento de iniciar o servidor é utilizado para carregar todas as variáveis para o ambiente do mesmo. Sendo assim qualquer um com acesso ao ficheiro ou às variáveis de ambiente poderá ver todas as configurações do servidor.

A solução mais indicada para este problema é a encriptação do ficheiro *env* e das variáveis de ambiente. A biblioteca mais utilizada para este objetivo é a *Secur-Env*, visto que, esta permite realizar a encriptação de um ficheiro com a indicação de uma *password*. A *password* deverá ser indicada no processo de inicialização do servidor de forma a ser possível ao mesmo desencriptar o ficheiro, sendo que a gestão das variáveis cifradas passa então a estar encarregue desta biblioteca.

Embora exista esta solução, continuam a haver possibilidades de ataque, uma vez que, é possível ver o histórico do terminal do servidor, o que permite obter a *password* escrita. Para resolver este problema é indicada a biblioteca *Readline*, pois esta possui o modo de *password* que apaga o histórico do terminal sempre que utilizado. Esta contém a vertente assíncrona, *Readline* e a vertente síncrona, *Readline-Sync*. Neste projeto, foi utilizada a versão síncrona da biblioteca visto que o objetivo é o servidor apenas inicie após a indicação da *password* sem nenhum serviço a correr em simultâneo.

### 2.2.2.11 Firebase

*Firebase* é uma solução que foi comprada pela *Google* em 2014. O seu objetivo é "*(...)to provide the tools and infrastructure that you need to build great apps(...)*"(Moroney, 2017), esta alcança este objetivo através da oferta de serviços pré configurados, sendo que "*(...)many of the technologies are available at no cost(...)*"(Moroney, 2017).

*Firestorage* também conhecida como *cloud storage*, é um serviço que dispõe "*(...)a simple API that is backed up by Google Cloud Storage(...)*"(Moroney, 2017), que permite guardar e transmitir até um gigabyte de ficheiros de forma gratuita.

*Cloud messaging* é também um serviço do *Firebase* que permite "*(...)to reliably deliver messages at no cost(...)*"(Moroney, 2017). Este garante que "*Over 98% of connected devices receive these messages in less than 500ms(...)*"(Moroney, 2017). *Cloud messaging* permite a utilização de diversas formas de envio de notificações como "*(...)driven by analytics to pick audiences, or using topics or other methods(...)*"(Moroney, 2017).

*Dynamic links* é um serviço do *Firebase* que permite a criação de "*(...)links to an app that contain context about what you want the end user to see in the app(...)*"(Moroney, 2017).

Esta ferramenta foi escolhida devido à sua capacidade de fornecer serviços pré configurados de forma gratuita, o que evita o gasto monetário e a alocação de tempo para a configuração de servidores durante o desenvolvimento.

### 2.2.2.12 Axios

Para ser possível realizar pedidos a outros serviços externos como *Firebase*, é necessário utilizar uma biblioteca capaz do mesmo. Para isso foi optado por utilizar *Axios*. Esta é "*(...)a promise-based HTTP Client for node.js(...)*"(Axios, 2023) que "*(...)uses the native node.js http module(...)*"(Axios, 2023). Esta disponibiliza um conjunto de métodos para a realização de pedidos a serviços externos, assim como a configuração total dos mesmos.

### 2.2.3 *Frontend*

Um dos requisitos do projeto é o desenvolvimento do *frontend* com a utilização de *Flutter*, visto que a empresa no seu trabalho diário já utiliza esta ferramenta. Deste modo, foi fulcral a aprendizagem desta ferramenta e da sua linguagem de programação o Dart.

#### 2.2.3.1 Desenvolvimento cross-platform

O desenvolvimento de aplicações *cross-platform* ou multi-plataforma, consiste no desenvolvimento de uma aplicação para diversas plataformas e este pode ser realizado de diversas formas, mas as principais formas conhecidas são *WebView*, nativo e outras abordagens.

As *frameworks* nativas são "(...)the most stable choice for mobile application development..."(Mainkar & Giordano, 2019) e dispõem de um grande comunidade e leque de aplicações desenvolvidas. O que torna estas *frameworks* estáveis é o facto de "(...)the app in this framework talks directly to the system..."(Mainkar & Giordano, 2019). Todo o desenho no ecrã é realizado através de o que é chamado de *OEM components* que são disponibilizados pela *framework* mas não permitem customização total. A grande vantagem desta abordagem é o facto de se o objetivo do projeto é o desenvolvimento para *iOS* e *Android*, então "(...)you need to learn two different languages..."(Mainkar & Giordano, 2019), porque estas são utilizadas para "(...)write two different apps with the same functionalities..."(Mainkar & Giordano, 2019) o que significa que "(...)every modification must be duplicated on both platforms..."(Mainkar & Giordano, 2019).

Uma outra abordagem para o desenvolvimento para diversas plataformas através de uma única base de código é o *WebView*. "(...)Cordova-, Ionic-, PhoneGap-, and WebView-based frameworks in general are good examples of cross-platform frameworks..."(Mainkar & Giordano, 2019), mas o grande problema desta abordagem é a "(...)lack in performance..."(Mainkar & Giordano, 2019) pois esta é composta por um processo intermédio chamado *WebView* que renderiza código *HTML*, isto significa que "(...)the app is basically a website..."(Mainkar & Giordano, 2019). Esta abordagem acrescenta também o componente de ponte que realiza o "(...)switch between JavaScript to the native system..."(Mainkar & Giordano, 2019) para obter acesso aos serviços nativos.

Um concorrente à tecnologia mencionada na secção (2.2.3.2) é o *React Native*, este assim como as *frameworks* nativas "(...)heavily relies on OEM components..."(Mainkar & Giordano, 2019) e "(...)expands the bridge concept in the WebView systems, and uses it not only for services, but also to build widgets..."(Mainkar & Giordano, 2019), isto leva a grandes problemas em termos de performance devido a que "(...)a component may be built hundreds of times during an animation, but due to the expanded concept of the bridge, this component may slow down to a great extent..."(Mainkar & Giordano, 2019).

### 2.2.3.2 Flutter

*Flutter* é uma *framework* desenvolvida pela *Google*, de inicio "*(...)was an experiment, as the developers at Google were trying to remove a few compatibility supports from Chrome, to try to make it run smoother(...)*"(Mainkar & Giordano, 2019), por fim, acabaram por descobrir que "*(...)they had something that rendered 20 times faster than Chrome did and saw that it had the potential to be something great(...)*"(Mainkar & Giordano, 2019). Em suma, *Google* desenvolveu "*(...)a layered framework that communicated directly with the CPU and the GPU in order to allow the developer to customize the applications as much as possible(...)*"(Mainkar & Giordano, 2019).

Para o *Flutter* tudo é um *widget*, "*(...)Orientation, layout, opacity, animation... everything is just a widget(...)*"(Mainkar & Giordano, 2019), isto permite que os utilizadores "*(...)choose composition over inheritance, making the construction of an app as simple as building a Lego tower(...)*"(Mainkar & Giordano, 2019). Todos estes *widgets* oficiais estão identificados no catálogo de *widgets* do *Flutter*. Como tudo no *Flutter* é composto por *widgets* "*(...)the more you learn how to use, create, and compose them, the better and faster you become at using Flutter(...)*"(Mainkar & Giordano, 2019).

A abordagem ao *cross-platform* realizada pelo *Flutter* é baseada em "*(...)AOT (Ahead Of Time) instead of JIT (Just In Time) like the JavaScript solutions(...)*"(Mainkar & Giordano, 2019) mostradas anteriormente. Esta também permite a conversação direta com o cpu sem necessidade de ponte e "*(...)does not rely on the OEM platform(...)*"(Mainkar & Giordano, 2019). Esta faculta que "*(...)custom components to use all the pixels in the screen(...)*"(Mainkar & Giordano, 2019), o que significa que "*(...)the app displays the same on every version of Android and iOS(...)*"(Mainkar & Giordano, 2019). Esta também utiliza "*(...)Platform Channels to use the services(...)*"(Mainkar & Giordano, 2019), o que leva a que "*(...)if you need to use a specific Android or iOS feature, you can do it easily(...)*"(Mainkar & Giordano, 2019).

### 2.2.3.3 Dart

*Dart* é a linguagem de programação utilizada pela *framework* *Flutter*, esta é "*(...)a general purpose programming language(...)*"(Bracha, 2015) que foi desenhada para ser "*(...)familiar to the vast majority of programmers(...)*"(Bracha, 2015). Esta linguagem é "*(...)purely object-oriented(...)*"o que significa que "*(...)all values in a Dart program manipulate at run time are objects(...)*"(Bracha, 2015), até tipos básicos como números e booleanos, esta é também "*(...)class-based, optionally typed(...)*"(Bracha, 2015). Esta é opcionalmente tipada, o que significa que a decisão de utilizar tipagens cai sobre o programador, mas no caso de *Flutter*, na sua versão mais recente é recomendado a utilização de tipagens de variáveis. Por fim, esta "*(...)supports mixin-based inheritance and actor-style concurrency(...)*"(Bracha, 2015).

### 2.2.3.4 Links de aplicações

Existem diferentes tipos de *links* sendo que para *mobile* é utilizado os *App Links*, *Deep Links* e os *Dynamic Links*.

Os *App Links* são "(...) web links that use the HTTP and HTTPS schemes(...)"(Developers, 2023), estes possuem também um atributo extra chamado *autoVerify*. Este atributo permite a uma aplicação "(...)to designate itself as the default handler of a given type of link(...)"(Developers, 2023), isto permite que "(...)app opens immediately if it's installed(...)"(Developers, 2023). O grande problema é que estes *links* não permitem o redirecionamento do utilizador para uma parte específica da aplicação e é necessário dispor de um domínio próprio.

Os *Deep Links* são "(...) URIs of any scheme that take users directly to a specific part of your app(...)"(Developers, 2023), o grande problema deste tipo de *links* é que se os utilizadores não dispuserem da aplicação instalada no dispositivo, este irá falhar e não permite a customização de comportamento.

Já os *Dynamic Links*, desenvolvidos pela *Firebase*, assim como os *Deep Links* "(...)if a user opens a Dynamic Link on iOS or Android, they can be taken directly to the linked content in your native app(...)"(Firebase, 2023a), mas para além disto, este permite que "(...)if a user opens the same Dynamic Link in a desktop browser, they can be taken to the equivalent content on your website(...)"(Firebase, 2023a), ou seja, este permite a customização de comportamento de *links* para diversas situações e em caso do utilizador não dispor da aplicação instalada, este permite que "(...)the user can be prompted to install it; then, after installation, your app starts and can access the link(...)"(Firebase, 2023a). Visto que este é o comportamento desejado pelo cliente da aplicação, então foi decidido utilizar esta abordagem.

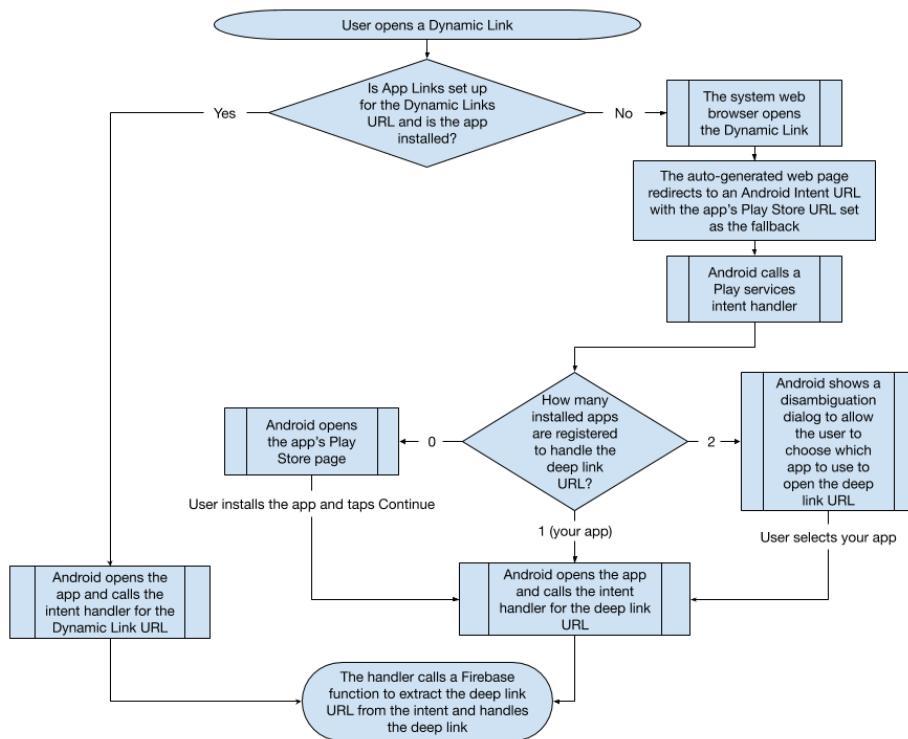


Figura 2.1: Funcionamento dos *Dynamic Links* em *Android* (Firebase, 2023b)

## 2.2.4 Qualidade de código

A qualidade do código desenvolvido é de extrema importância para possibilitar e facilitar a manutenção da solução desenvolvida, assim como a melhoria da segurança da mesma. Esta permite a percepção de objetivo de código e a estruturação do mesmo de acordo com normas estabelecidas perante a comunidade. A qualidade de código tem como objetivo diminuir a complexidade, pois nem sempre documentação e estruturação é o suficiente para o código ser de qualidade, sendo que, este deverá ser simplificado para ser interpretável. Todos estes pontos poderão ser implementados através do uso de *design patterns*, documentação e testes de código.

### 2.2.4.1 Design patterns

Os *design patterns* "*(...)names, abstracts, and identifies the key aspects of a common design structure that make it useful for creating a reusable object-oriented design(...)*"(Gamma et al., 2009). Estes identificam "*(...)the participating classes and instances, their roles and collaborations, and the distribution of responsibilities(...)*"(Gamma et al., 2009).

O *Model View Controller (MVC)* consiste em "*(...)three kinds of objects(...)*"(Gamma et al., 2009), o *Model* que é "*(...)the application object(...)*"(Gamma et al., 2009), a *View* que é a "*(...)screen presentation(...)*"(Gamma et al., 2009) do *Model* e por fim o *Controller* que "*(...)defines the way the user interface reacts to user input(...)*"(Gamma et al., 2009). Antes do *MVC* "*(...)user interface designs tended to lump these objects together(...)*"(Gamma et al., 2009), o *MVC* "*(...)decouples them to increase flexibility and reuse(...)*"(Gamma et al., 2009), através de "*(...)establishing a subscribe/notify protocol between them(...)*"(Gamma et al., 2009).

### 2.2.4.2 Documentação

Para ser possível manter todo o projeto desenvolvido, foi criada documentação a diferentes níveis, sendo esta desenvolvida a nível de serviços com a explicação do objetivo do serviço e que dados este recebe, como também a nível de código sendo explicado o código desenvolvido em cada *script* existente. Para estes níveis de documentação foram utilizadas diferentes ferramentas, para documentação de serviços, foi utilizada a ferramenta *swagger* e para a documentação de código foi utilizado o *typedoc*.

### 2.2.4.3 Typedoc

*Typedoc* é um "*(...)documentation generator for TypeScript(...)*"(TypeDoc, 2023), uma ferramenta "*(...)which reads your TypeScript source files, parses comments contained within them, and generates a static site containing documentation(...)*"(TypeDoc, 2023). Esta utiliza chaves específicas para detetar as informações e criar categorias para melhor organizar toda a documentação. Esta documentação possibilita a interligação entre si mesma, o que permite ao visualizador seguir todo o processo.

#### 2.2.4.4 Swagger

*Swagger* é uma ferramenta "(...) built around the OpenAPI Specification(...)"(SmartBear, 2023) que ajudam com "(...) design, build, document and consume REST APIs(...)"(SmartBear, 2023), o *OpenAPI* "(...) is an extit API description format for REST APIs(...)"(SmartBear, 2023) que poderá ser "(...) written in YAML or JSON(...)"(SmartBear, 2023). Esta permite gerar documentação a nível de serviços que é acessível a partir do mesmo servidor com indicação de uma rota, o que evita outro servidor para hospedar a documentação. Esta documentação poderá ser gerada a partir de comentários a nível de código ou a partir de um ficheiro em formato *JSON* ou *yaml*, como mencionado anteriormente. Este ficheiro poderá ser mantido manualmente ou automaticamente. Neste projeto foi decidido manter este manualmente em *JSON*.

#### 2.2.4.5 Testes de código

Aquando o fim do desenvolvimento de cada serviço é necessário testar este para verificar se a funcionalidade encontra-se de acordo com o desejado e/ou existem erros de código. Para realizar estes testes poderão ser utilizadas ferramentas de auxílio ou então poderão ser realizados manualmente. O grande problema dos testes manuais é a exaustividade, uma vez que, são longos e propensos a erros, pelo que, são realizados em menor escala. Para os testes deste projeto foram utilizadas ferramentas de auxílio, sendo as ferramentas escolhidas *mocha* e *chai*.

*Mocha* é uma *framework* de testes *JavaScript* que permite teste assíncrono." (...) *Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases(...)"*(Foundation, 2023). Esta ferramenta foi escolhida devido à sua simplicidade e à capacidade de testar código *TypeScript* para além de *javascript*.

*Chai* é "(...) a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework(...)"(Library, 2023), neste caso esta foi utilizada em conjunto com *Mocha*. A utilização de *BDD* permite a utilização de uma "(...) expressive language & readable style(...)"(Library, 2023), já *TDD* é "(...) more classical feel(...)"(Library, 2023). Para tornar os testes de código mais interpretáveis foi explorado o estilo de testes *BDD*.

A realização de testes de código foi muito importante, visto que, com esta ferramenta foi possível encontrar erros de lógica de negócio, bem como, erros de código tanto a nível da formulação de respostas como a nível de código.



# **3. Análise e especificação**

Na análise e especificação está contida a descrição do modelo de negócio, onde serão identificados os principais problemas do modelo atual e será indicada a necessidade de implementação do *software*. Nesta análise estarão identificados os objetivos de negócio e o principal impacto deste *software* neste modelo de negócio. Nesta análise estará também contida a engenharia de *software* com toda a especificação do projeto em mãos.

## **3.1 Modelo de negócio**

Este projeto surgiu com a necessidade da Motorline melhorar a comunicação e a experiência das empresas clientes e técnicos. Atualmente sempre que um técnico possui uma questão este deverá contactar o suporte técnico o que pode levar a sobrelocação do sistema, ou então este deverá se juntar ao grupo do Facebook de técnicos e colocar neste a sua questão. Sempre que é necessário um manual de produto ou sempre que o utilizador pretender ver o catálogo este deverá deslocar-se ao *site* da Motorline e procurar o produto no catálogo onde não existe um acesso rápido. Como este processo acaba por não ser prático para o utilizador, foi decidido suprimir o problema com o projeto *Install & Go*. Com estas questões em mente surgiu então a ideia de implementar um fórum de resolução de problemas.

## **3.2 Objetivos de negócio**

Este *software* visa minimizar o problema acima descrito, visto que, são efetuadas muitas questões por parte dos técnicos que são comuns entre si. Deste modo, surgiu a ideia de implementar um fórum onde o técnico poderá pesquisar por questões semelhantes à sua, e encontrar a solução sem necessidade de contactar o suporte técnico. O técnico poderá também expor a sua questão com o anexo de imagens, o que facilita o processo de resolução da questão. Para existir partilha de conhecimento este poderá responder a tópicos, pois caso encontre alguma questão que já tenha resolvido ajudará outro técnico. O técnico também poderá alterar a visibilidade do seu tópico caso deseje que apenas técnicos Motorline visualizem a sua questão.

A empresa poderá realizar as mesmas ações que o técnico, e também criar contas para os seus técnicos e realizar a gestão destas sendo importante permitir apagar e impedir acesso à conta em caso de necessidade.

### 3.3 Estudo de Soluções existentes

Para ser possível entender todo o negócio onde o projeto se encaixa foi realizado um estudo que engloba outras soluções do mesmo ramo. Estas soluções investigadas foram indicações do supervisor de estágio, pois este já tinha realizado um estudo da situação do negócio previamente. As soluções investigadas foram, *FixAndGo*, *My VDS* e *Ultimate Cardin*. A solução *FixAndGo* foi indicada devido à capacidade de calcular a possibilidade de instalar um motor com base em medidas indicadas pelo utilizador. Já a solução *My VDS* foi mencionada devido ao facto de dispor de documentação sobre os seus produtos como manuais, vídeos e uma forma de comunicação com a equipa de suporte. Por fim, a solução *Ultimate Cardin* foi utilizada como exemplo de *design* a não seguir, devido a ser confuso, esta foi também utilizada como exemplo de organização de design do ecrã de detalhes de produto.

### 3.4 Domínio de aplicação do sistema

Com o diagrama da Figura 3.1 é possível visualizar todos os atores do *software* e as suas ações, assim como os sistemas envolvidos na aplicação e as funções. Destes é possível identificar que este *software* contém três atores principais, o Utilizador que é um utilizador sem sessão iniciada, o Técnico, um utilizador com sessão iniciada, já a Empresa é uma empresa cliente da Motorline. Também é possível visualizar os diferentes sistemas integrados no projeto, como Servidor Motorline onde serão obtidas informações do catálogo de produtos, Servidor *Install & Go* onde estão todas as funções de suporte ao *software*, o Servidor de Imagens onde serão guardadas todas as imagens da aplicação e por fim o Servidor de *Email* que enviará *email* com o código de validação de conta para os clientes assim que se registarem no *software*.

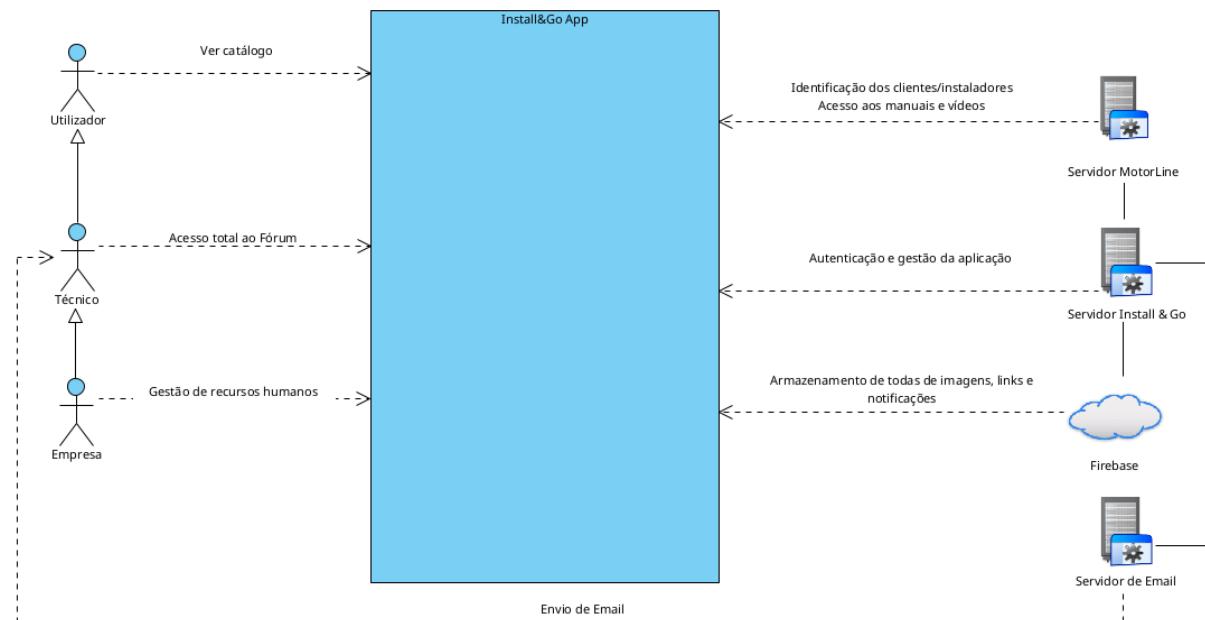


Figura 3.1: Diagrama de contexto da aplicação

## 3.5 Operações a realizar no sistema

A primeira tarefa a realizar no desenvolvimento desta etapa do projeto foi o levantamento dos Requisitos Funcionais (RF) (Tabela 3.1) e Requisitos Não Funcionais (RNF), tendo sido posteriormente validados com cliente. Este levantamento de requisitos funcionais, foi realizado num conjunto de reuniões realizadas com os clientes do projeto.

Tabela 3.1: Tabela de requisitos funcionais

#	Descrição	Fonte	Data
	Autenticação		
RF01	O Utilizador deverá conseguir visualizar o catálogo	Helder Remelhe	2/13/2023
RF02	A Empresa deverá conseguir realizar o registo na aplicação	Helder Remelhe	2/13/2023
RF03	O Técnico deverá conseguir realizar o login na aplicação	Helder Remelhe	2/13/2023
RF04	O login deverá ser realizado utilizando número de contribuinte e <i>password</i>	Helder Remelhe	2/13/2023
RF05	Assim que o registo é realizado, a motor-line deverá validar a conta sendo posteriormente enviado um <i>email</i> para a empresa ativar e utilizar a conta	Rafael Viana	2/13/2023
RF06	O Técnico deverá conseguir pedir reenvio de código de verificação de conta	Rafael Viana	2/27/2023
RF07	Os Técnicos deverão ser identificados como técnicos certificados ou técnicos oficiais	Helder Remelhe	2/27/2023
	Fórum		
RF08	O Técnico deverá conseguir aceder ao fórum e realizar operações	Helder Remelhe	2/13/2023
RF09	O Técnico deverá conseguir visualizar os tópicos mais recentes	Brainstorming	2/14/2023
RF10	O Técnico deverá conseguir visualizar os tópicos em destaque	Brainstorming	2/14/2023
RF11	O Técnico deverá conseguir visualizar os tópicos por responder	Brainstorming	2/14/2023
RF12	O Técnico deverá conseguir acessar aos tópicos privados do fórum	Helder Remelhe	2/13/2023
RF13	O Técnico deverá conseguir pesquisar por tópicos referentes a um assunto desejado	Brainstorming	2/14/2023

Continua na página seguinte

Tabela 3.1: Tabela de requisitos funcionais (Continuação)

RF14	O Técnico deverá conseguir pesquisar por tópicos referentes a um produto desejado	Brainstorming	2/14/2023
RF15	O Técnico deverá conseguir pesquisar por tópicos referentes a um produto por código QR	Brainstorming	2/14/2023
RF16	O Técnico deverá conseguir visualizar os seus tópicos	Brainstorming	2/14/2023
Criar Tópico			
RF17	O Técnico deverá conseguir criar tópicos para expor a sua questão	Helder Remelhe	2/13/2023
RF18	O Técnico deverá conseguir colocar o seu tópico público ou privado para assim apenas outras empresas conseguirem ver	Helder Remelhe	2/13/2023
RF19	O Técnico deverá conseguir indicar tipo de tópico para agilizar a identificação do mesmo	Helder Remelhe	2/14/2023
RF20	O Técnico deverá conseguir indicar o produto referente ao tópico para agilizar a identificação da sua questão	Brainstorming	2/14/2023
RF21	O Técnico deverá conseguir anexar imagens ou vídeos ao tópico em questão para agilizar a comunicação e identificação do seu problema	Helder Remelhe	2/13/2023
Gestão de tópico			
RF22	O Técnico deverá conseguir indicar qual a melhor resposta ao seu tópico	Helder Remelhe	2/14/2023
RF23	O Técnico deverá conseguir indicar que o tópico se encontra finalizado quando o seu problema se encontrar resolvido	Helder Remelhe	2/13/2023
RF24	O Técnico deverá conseguir remover o seu tópico	Brainstorming	2/14/2023
RF25	O Técnico deverá conseguir alterar a visibilidade do seu tópico	Helder Remelhe	2/13/2023
Tópicos			
RF26	O Técnico deverá conseguir ver todas as respostas ao tópico	Helder Remelhe	2/13/2023
RF27	O Técnico deverá conseguir gostar do tópico para dar destaque ao mesmo	Brainstorming	2/14/2023
RF28	O Técnico deverá conseguir remover uma resposta que colocou em um tópico	Brainstorming	2/14/2023

Continua na página seguinte

Tabela 3.1: Tabela de requisitos funcionais (Continuação)

	Respostas a Tópicos		
RF29	O Técnico deverá conseguir comentar tópicos	Helder Remelhe	2/13/2023
RF30	O Técnico deverá conseguir responder a comentários de tópicos	Helder Remelhe	2/13/2023
RF31	O Técnico deverá conseguir anexar imagens e videos à sua resposta	Helder Remelhe	2/13/2023
RF32	O Técnico deverá conseguir gostar de alguma resposta para dar destaque a esta resposta	Helder Remelhe	2/13/2023
	Perfil		
RF33	O Técnico deverá conseguir alterar o seu <i>email</i>	Helder Remelhe	2/27/2023
RF34	O Técnico deverá conseguir alterar imagem de perfil	Brainstorming	2/27/2023
RF35	O Técnico deverá conseguir alterar nome	Brainstorming	2/27/2023
RF36	O Técnico deverá ser identificado como Técnico oficial ou certificado	Helder Remelhe	2/28/2023
	Notificações		
RF37	O Técnico deverá conseguir receber notificações por <i>email</i> e/ou push	Rafael Viana	2/27/2023
RF38	O Técnico deverá conseguir alterar o tipo de notificação entre relatório diário e notificação direta para ambos os tipos	Rafael Viana	2/27/2023
RF39	O Técnico deverá conseguir ver as suas notificações	Helder Remelhe	2/27/2023
RF40	O Técnico deverá conseguir apagar as suas notificações	Helder Remelhe	2/27/2023
	Gestão de recursos humanos		
RF41	A Empresa deverá conseguir criar contas para os seus técnicos	Brainstorming	2/13/2023
RF42	Assim que a conta de técnico for criada, este deverá receber um <i>email</i> para registrar as restantes informações e ativar a conta	Helder Remelhe	2/27/2023
RF43	A Empresa deverá conseguir impedir acesso a uma conta de técnico	Helder Remelhe	2/27/2023
RF44	A Empresa deverá conseguir remover uma conta de técnico da empresa	Helder Remelhe	2/27/2023

## 3.6 Descrição dos intervenientes

O projeto envolve um conjunto de intervenientes, sendo estes, o utilizador, a empresa e o técnico. Estes desempenham um papel fundamental e podem realizar diferentes ações.

O utilizador, sem sessão iniciada terá apenas acesso ao catálogo de produtos.

O técnico conseguirá realizar as mesmas ações que o utilizador, mas este ator conseguirá também ter acesso total ao fórum e se for técnico oficial tem também acesso a questões privadas. O fórum permite expor questões, com auxílio de imagens, a ligação da questão a uma categoria e um produto para facilitar a sua resolução. As questões poderão ser públicas, ou então privadas, para apenas técnicos oficiais conseguirem ver. Assim que o técnico estiver satisfeito com a questão poderá indicar a melhor resposta obtida ao problema sendo esta destacada e a tópico finalizada.

O técnico pode realizar pesquisas por questões, onde evita um telefonema ou o preenchimento de um formulário para contactar um técnico oficial. Com estas pesquisas poderá responder a outras questões. As respostas podem conter imagens anexadas e responder a outras respostas para manter comunicação. O técnico poderá destacar tópicos e respostas de tópicos com um *like*.

A empresa pode realizar a gestão de contas de técnicos, onde cria, impede acesso e elimina a conta em caso de necessidade.

## 3.7 Partes Interessadas

Este projeto foi proposto pelo supervisor de estágio, na empresa Motorline representada por Rafael Viana e André Viana, as partes interessadas.

## 3.8 Condições Específicas

Ao longo de um conjunto de reuniões foram levantados os requisitos não funcionais do projeto representados na Tabela 3.2.

Tabela 3.2: Tabela de requisitos não funcionais

#	Tipo	Descrição
RNF01	Cultural	O <i>software</i> deverá suportar vários idiomas (prioritariamente, português e inglês)
RNF02	Configuração	A falha dos servidores, implica a inutilidade total do fórum da aplicação
RNF03	Conexão	Necessário o uso de dados móveis ou WIFI
RNF04	Segurança	O <i>software</i> tem de ser seguro para proteger os dados do consumidor
RNF05	Desenvolvimento	A aplicação deverá ser desenvolvida com a utilização da tecnologia <i>Flutter</i>

### 3.9 Esquematização do conteúdo das páginas

Para ser possibilitar a percepção dos dados necessários para alimentar o *software*, o que apresentar em cada página e também como navegar entre os ecrãs da aplicação foi então desenhado um esquema (Figura 3.2).

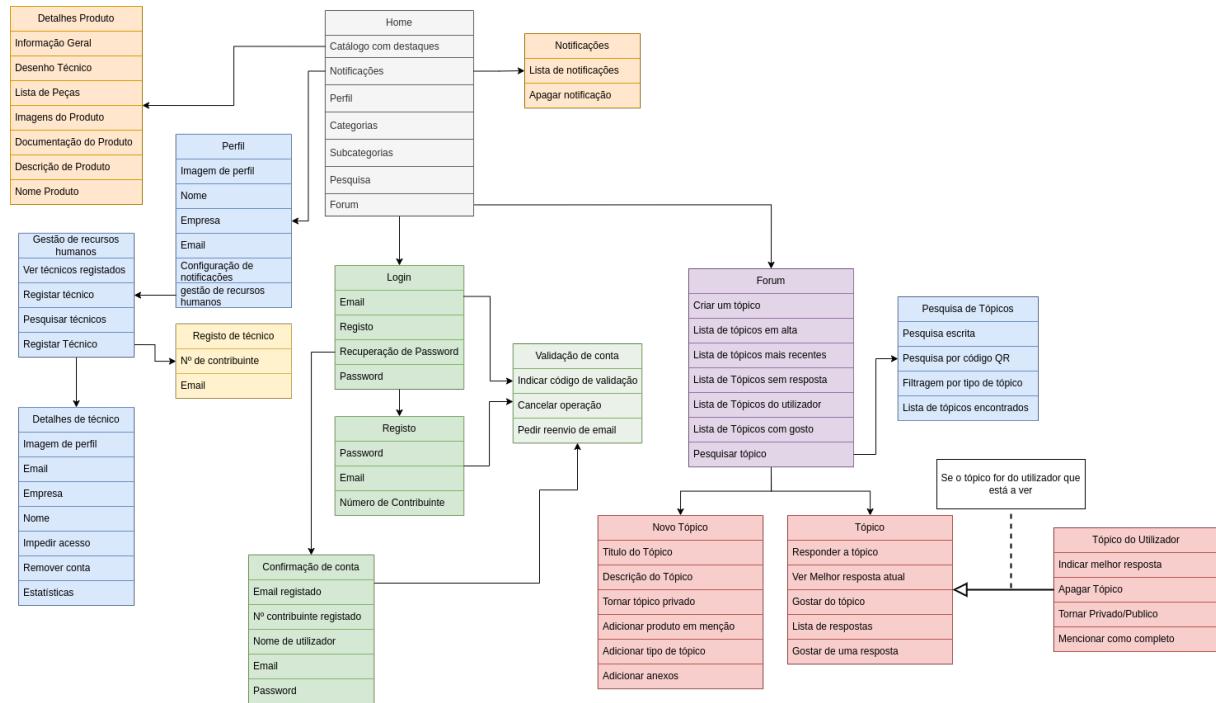


Figura 3.2: Esquema de organização de páginas do *software*

### 3.9.1 Autenticação e página Inicial

Através deste esquema é possível perceber que do ecrã principal, o utilizador tem acesso ao catálogo de produtos e ao fórum, neste pode também realizar o *login* e o *logout* que redirecionam para os respetivos ecrãs.

No ecrã de *login* é necessário o utilizador indicar o número de contribuinte e a *password*, neste ele pode também pedir recuperação de *password* e/ou redirecionar para o registo onde necessitará de número de contribuinte, *password* e *email* para o realizar.

Em caso de o utilizador não possuir a conta ativa, este será encaminhado para o ecrã de validação conta em que deverá indicar o código de validação, cancelar a operação e pedir o reenvio do código de validação.

Em caso de se tratar de um técnico que necessita de confirmar a sua conta, este conseguirá ver as informações registadas, introduzir o seu nome, alterar o seu *email* e *password*.

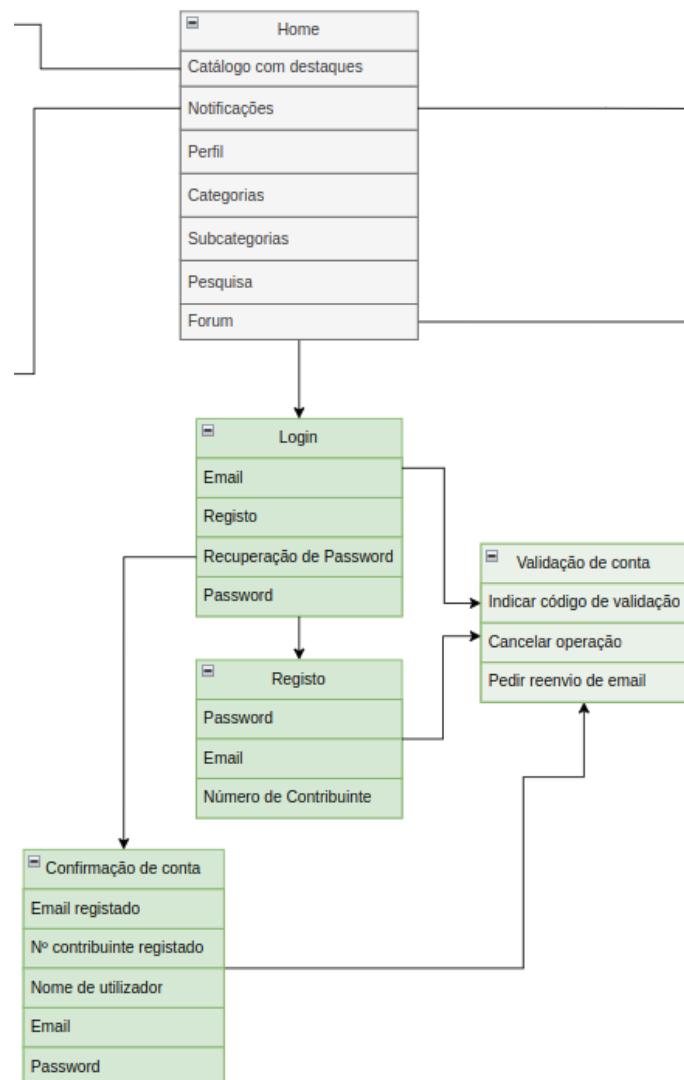


Figura 3.3: Esquema de organização das páginas de autenticação e página inicial

### 3.9.2 Fórum

Através do ecrã inicial o utilizador pode direcionar-se para o ecrã do fórum. Neste ecrã, poderá pesquisar por tópicos, ou então aceder a tópicos em alta, mais recentes ou sem resposta. O técnico consegue também criar e ver as seus tópicos.

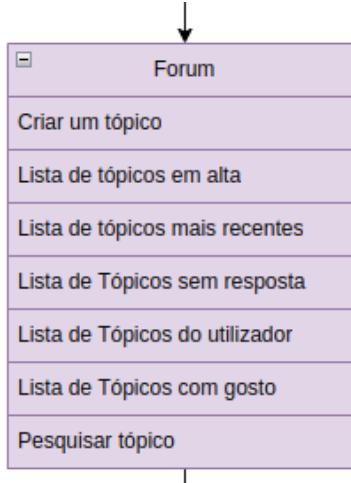


Figura 3.4: Esquema de organização da página de fórum

### 3.9.3 Criar nova tópico

Quando o técnico decide criar uma tópico, este tem de indicar o título e a descrição, de seguida poderá indicar se é privado ou não, o tipo de tópico, o produto referente e adicionar anexos.

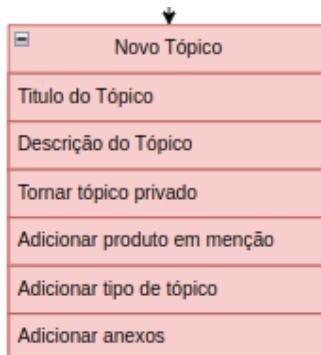


Figura 3.5: Esquema de organização da página de criação de tópicos

### 3.9.4 Detalhes de tópicos

O técnico pode também ver os detalhes do tópico, responder, gostar e gostar de uma resposta. Caso esta seja do mesmo, este pode indicar a melhor resposta, apagar a tópico, tornar pública ou privada e indicar como completa.

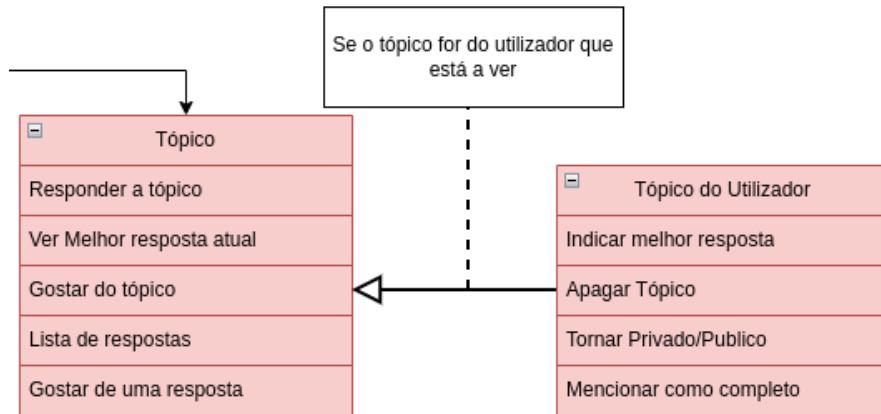


Figura 3.6: Esquema de organização da página de detalhes de tópico

### 3.9.5 Pesquisa de tópicos

A página de pesquisa permite ao técnico procurar por tópicos específicos tanto por nome como por produto. Para além da pesquisa o utilizador pode também realizar a filtragem dos tópicos por tipo e categoria.

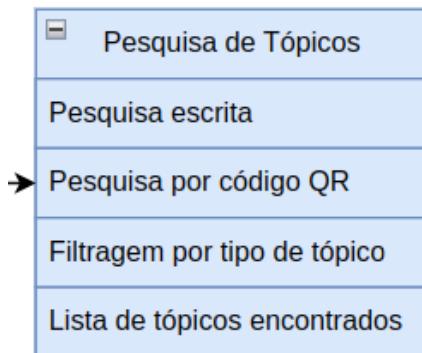


Figura 3.7: Esquema de organização da página de pesquisa de tópicos

### 3.9.6 Notificações

A página de notificações permite ao técnico visualizar as suas notificações, assim como também apagar.

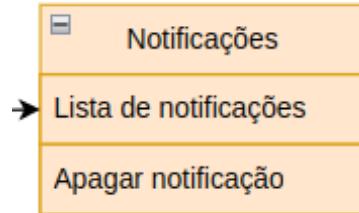


Figura 3.8: Esquema de organização da página de notificações

### 3.9.7 Perfil

A página de perfil de técnico permite visualizar as suas informações, assim como alterar o seu *email* e configurar as notificações. Caso se trate de uma empresa a visualizar o seu perfil, esta poderá ter acesso à gestão de recursos humanos, para gerir os seus técnicos.

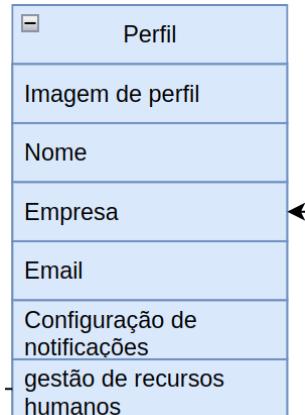


Figura 3.9: Esquema de organização da página de perfil

### 3.9.8 Gestão de recursos humanos

A página de gestão de recursos humanos permite à empresa gerir todos os seus técnicos registados e criar contas. Assim que a empresa seleciona um técnico, esta vê o seu perfil, com estatísticas.

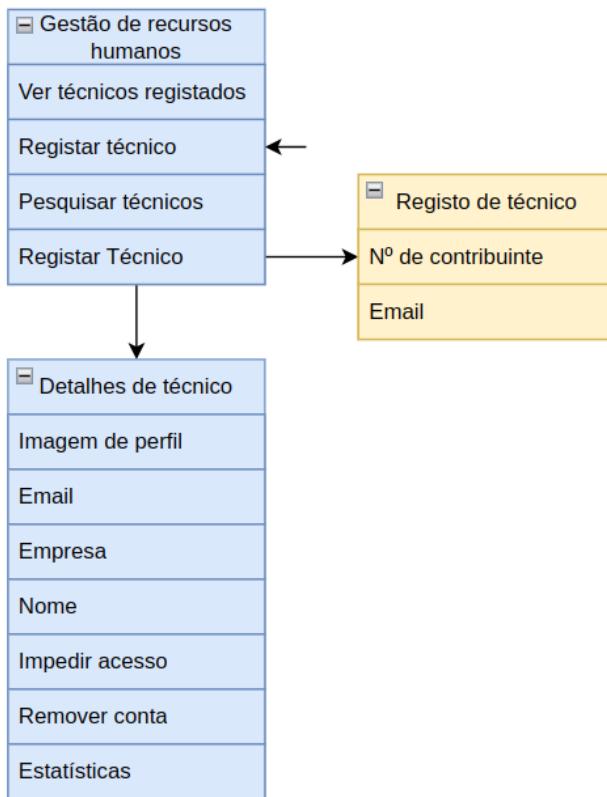


Figura 3.10: Esquema de organização da página de gestão de recursos humanos

## 3.10 User Stories

Antes de desenvolver os casos de uso foram criadas *User Stories (US)* para ser possível descrever os objetivos ao realizar uma ação.

Tabela 3.3: Tabela de *user stories*

#	Autor	Descrição
Autenticação		
US01	Utilizador	Eu como Utilizador, quero conseguir utilizar a aplicação sem realizar o login
US02	Empresa	Eu como Empresa, quero conseguir realizar o registo na aplicação
US03	Técnico	Eu como Técnico, quero conseguir realizar o login na aplicação utilizando o número de contribuinte e <i>password</i>
US04	Técnico	Eu como Técnico, quero conseguir pedir reenvio de código de ativação de conta caso eu não receba o código
US05	Técnico	Eu como Técnico, quero conseguir ser identificado como tal na aplicação
Fórum		
US06	Técnico	Eu como Técnico, quero conseguir acessar ao fórum
US07	Técnico	Eu como Técnico, quero conseguir visualizar os tópicos mais recentes, de forma a conseguir ver os mais falados no dia atual
US08	Técnico	Eu como Técnico, quero conseguir visualizar os tópicos em destaque, de forma a ver quais são mais falados
US09	Técnico	Eu como Técnico, quero conseguir ver os meus tópicos de forma a conseguir aceder a estes facilmente
US10	Técnico	Eu como Técnico, quero conseguir visualizar os tópicos por responder, de forma a conseguir ajudar alguém com maior facilidade
US11	Técnico	Eu como Técnico, quero conseguir visualizar os tópicos privados
US12	Técnico	Eu como Técnico, quero conseguir realizar filtragem de tópicos por tipo
US13	Técnico	Eu como Técnico, quero conseguir pesquisar por um tópico relativo a um assunto de forma a obter a solução
US14	Técnico	Eu como Técnico, quero conseguir pesquisar por um tópico relativo a um produto de forma a encontrar questões comuns a este
US15	Técnico	Eu como Técnico, quero conseguir pesquisar por código QR de um produto de forma a encontrar tópicos referentes ao mesmo mais facilmente

Continua na página seguinte

Tabela 3.3: Tabela de *user stories* (Continuação)

	Criar Tópico	
US16	Técnico	Eu como Técnico, quero conseguir criar tópicos de forma a conseguir expor questões
US17	Técnico	Eu como Técnico, quero conseguir indicar se o meu tópico é público ou privado, de forma a conseguir respostas de qualquer cliente, ou apenas de técnicos
US18	Técnico	Eu como Técnico, quero conseguir indicar o tipo de tópico em que o este se enquadra de forma a facilitar a sua identificação
US19	Técnico	Eu como Técnico, quero conseguir indicar o produto referente ao tópico para facilitar a identificação do mesmo
US20	Técnico	Eu como Técnico, quero conseguir anexar imagens ao tópico de forma a facilitar a comunicação e identificação do problema
	Gestão de Tópico	
US21	Técnico	Eu como Técnico quero conseguir indicar qual a melhor resposta ao meu tópico de forma a facilitar o encontro da solução do problema a outros clientes ou técnicos com o mesmo problema
US22	Técnico	Eu como Técnico quero conseguir indicar que o tópico se encontra finalizado quando o problema está resolvido
US23	Técnico	Eu como Técnico quero conseguir remover o meu tópico
US24	Técnico	Eu como Técnico quero conseguir alterar a visibilidade do meu tópico
	Tópicos	
US25	Técnico	Eu como Técnico, quero conseguir ver todas as respostas a um tópico
US26	Técnico	Eu como Técnico, quero conseguir gostar de um tópico caso o ache relevante
US27	Técnico	Eu como Técnico, quero conseguir apagar uma resposta minha
	Respostas a Tópicos	
US28	Técnico	Eu como Técnico, quero conseguir comentar um tópico de forma a dar a minha resposta
US29	Técnico	Eu como Técnico, quero conseguir responder a um comentário de forma a comunicar
US30	Técnico	Eu como Técnico, quero conseguir anexar imagens ao meu comentário
US31	Técnico	Eu como Técnico, quero conseguir gostar de um comentário caso ache este relevante

Continua na página seguinte

Tabela 3.3: Tabela de *user stories* (Continuação)

	Perfil	
US32	Técnico	Eu como Técnico quero conseguir alterar o meu <i>email</i>
US33	Técnico	Eu como Técnico quero conseguir alterar a minha imagem de perfil
US34	Técnico	Eu como Técnico quero conseguir alterar o meu nome
	Notificações	
US35	Técnico	Eu como Técnico quero conseguir receber notificações por <i>email</i> e/ou push de forma a manter-me atualizado das minhas questões
US36	Técnico	Eu como Técnico quero conseguir alterar o tipo de notificação que recebo entre relatório diário e tempo real
US37	Técnico	Eu como Técnico quero conseguir apagar as minhas notificações de forma a evitar aglomeração
	Gestão de recursos humanos	
US38	Empresa	Eu como Empresa quero conseguir criar conta para os meus técnicos utilizarem o fórum
US39	Empresa	Eu como Empresa quero conseguir impedir acesso a uma conta de técnico
US40	Empresa	Eu como Empresa quero conseguir remover uma conta de técnico em caso de este já não pertencer à empresa

### 3.11 Casos de uso

Para transformar as *user stories* em ações e especificar todas as reações do sistema com o qual o ator interage foram desenvolvidos casos de uso (*Use Cases (UC)*).

Tabela 3.4: Tabela de casos de uso

#	User Story	Autor	Nome	Descrição
Criar tópico				
UC 1.0	US15	Técnico	Criar novo tópico	Criação de um novo tópico no fórum
Pesquisa de tópicos				
UC 1.1	US11	Técnico	Pesquisar tópicos específicos	Pesquisar por tópicos no fórum
UC 1.1.1	US12 e US13	Técnico	Pesquisa escrita	Pesquisar tópicos por assunto
UC 1.1.2	US14	Técnico	Pesquisa por código QR	Pesquisar tópicos referentes a um produto
Listagens de tópicos				
UC 1.2	US05	Técnico	Ver tópicos	Ver listagens de tópicos do fórum
Detalhes de tópico				
UC 1.3	US08	Técnico	Selecionar tópico	Ver detalhes de um tópico selecionado
UC 1.3.1	US21	Técnico	Finalizar tópico	Finalizar um tópico para indicar que está respondido
UC 1.3.2	US20	Técnico	Selecionar melhor resposta	Selecionar a melhor resposta do tópico
UC 1.3.3	US22	Técnico	Eliminar tópico	Eliminar um tópico do fórum
UC 1.3.4	US23	Técnico	Alterar visibilidade do tópico	Alterar a visibilidade de um tópico entre público e privado
UC 1.3.5	US28	Técnico	Comentar o tópico	Comentar um tópico
UC 1.3.6	US25	Técnico	Gostar de tópico	Gostar de um tópico
Comentários				
UC 1.3.7	US24	Técnico	Ver comentários	Ver comentários do tópico
UC 1.3.7.1	US27	Técnico	Apagar comentário	Apagar comentário de um tópico
UC 1.3.7.2	US29	Técnico	Responder a comentário	Responder a um comentário de um tópico

Continua na página seguinte

Tabela 3.4: Tabela de casos de uso (Continuação)

UC 1.3.7.3	US31	Técnico	Gostar de comentá- rio	Gostar de um comentário
Ativação de conta				
UC 1.4	-	Técnico	Ativação de conta	Ativar conta de cliente
UC 1.4.1	US04	Técnico	Pedir reenvio de có- digo de ativação	Pedir reenvio de <i>email</i> de có- digo de ativação
Perfil				
UC 1.5	US31 - US32 - US33	Técnico	Ver Perfil	Ver perfil de utilizador
Notificações				
UC1.6	US34 e US36	Técnico	Ver notificações	Ver todas as notificações
UC1.7	US34 e US35	Técnico	Configuração de notificações	Configurar o modo e tipo de notificações a receber
Gestão de recursos humanos				
UC1.8	US37	Empresa	Registar Técnico	Registar conta de técnico da empresa
UC1.9	US38	Empresa	Impedir acesso a técnico	Registar conta de técnico da empresa
UC1.10	US39	Empresa	Remover conta de técnico	Registar conta de técnico da empresa

### 3.11.1 Especificação de casos de uso

Para demonstrar todas as interações entre os atores e o sistema, assim como todas as ações e fluxos possíveis, foram realizadas especificações de casos de uso.

#### 3.11.1.1 Especificação de caso de uso de listagem de tópicos

Através da listagem de tópicos é possível visualizar todas as listagens que o utilizador poderá visualizar, sendo que, o técnico oficial consegue para além destas listagens, ver os seus tópicos e os tópicos privados.

Tabela 3.5: Tabela de especificação de caso de uso de listagem de tópicos do técnico

Caso de Uso	Ver listagem de tópicos	
Descrição	Ver a listagem de tópicos existentes no fórum por diversas categorias	
Autor	Técnico	
Pré-condição	-	
Pós-condição	-	
	Autor	Sistema
Fluxo Principal	1-Ver tópicos populares	
		2-Lista de tópicos populares
Fluxo Alternativo(A1)	1-Ver tópicos mais recentes	
		2-Lista de tópicos mais recentes
Fluxo Alternativo(A2)	1-Ver meus tópicos	
		2-Lista de tópicos do técnico
Fluxo Alternativo(A3)	1-Ver tópicos por responder	
		2-Lista de tópicos por responder

### 3.11.1.2 Especificação de caso de uso de criar novo tópico

Aquando a criação de um tópico, um técnico poderá realizar diversas ações sendo que obrigatoriamente terá de indicar o título, descrição e tipo de tópico, para além desta informação o técnico poderá anexar imagens, referenciar um produto e indicar a visibilidade.

Tabela 3.6: Tabela de especificação de caso de uso login

Caso de Uso	Criar novo tópico	
Descrição	Criação de um novo tópico no fórum	
Autor	Técnico	
Pré-condição	Clicar em adicionar novo tópico	
Pós-condição	-	
	Autor	Sistema
Fluxo Principal	1-Indicar o título do tópico	
	2-Indicar a descrição do tópico	
	3-Indicar se o tópico é privado	
	4-Indicar o tipo do tópico	
	5-Indicar o produto referido no tópico	
	6-Adicionar imagens de anexo	
	7-Confirmar a criação do tópico	
		8-Verificar se título está inserido
		9-Verificar se descrição está inserida
		10-Inserir novo tópico no fórum
Fluxo Alternativo(A1)	1-Cancelar a criação do tópico	
Fluxo Alternativo(A2)	1-Indicar o título do tópico	
	2-Indicar se o tópico é privado	
	3-Confirmar a criação do tópico	
		4-Verificar se título está inserido
		5-Verificar se descrição está inserida
		6-Erro descrição em falta

### 3.11.1.3 Especificação de caso de uso de pesquisar tópicos por escrito

Assim que um técnico deseje pesquisar por um assunto específico de tópico este poderá realizar uma pesquisa escrita onde conseguirá realizar filtragem por tipo e categoria de tópico.

Tabela 3.7: Tabela de especificação de caso de uso de pesquisa por escrito

Caso de Uso	Pesquisa por escrita	
Descrição	Pesquisar por tópicos no fórum	
Autor	Utilizador	
Pré-condição	Selecionar pesquisa de fórum	
Pós-condição	-	
	Ator	Sistema
Fluxo Principal	1-Pesquisar assunto	
		2-Lista de tópicos do assunto
	3-Filtrar por tipo	3-Filtragem de tópicos do assunto por tipo
Fluxo Alternativo(A1)	1-Pesquisar assunto	
		2-Lista de tópicos do assunto

### 3.11.1.4 Especificação de caso de uso de ver finalizar tópico

Quando um técnico encontra-se satisfeito com a solução do problema este poderá indicar que o tópico está finalizado, o que é sinalizado para outros técnicos.

Tabela 3.8: Tabela de especificação de caso de uso de finalizar tópico

Caso de Uso	Finalizar tópico	
Descrição	Finalizar um tópico para indicar que está respondido	
Autor	Técnico	
Pré-condição	Clicar no tópico desejado	
Pós-condição	Alterar tópico para finalizado	
	Ator	Sistema
Fluxo Principal	1-Clicar em finalizar tópico	
		2-Alterar tópico para finalizado
Fluxo Alternativo(A1)	-	-

### 3.11.1.5 Especificação de caso de uso de selecionar melhor resposta

Sempre que o técnico encontrar uma resposta no seu tópico que se destaca na solução da sua questão, este poderá colocar esta resposta como a melhor resposta do tópico. Caso já exista uma melhor resposta, esta automaticamente é removida e a nova é colocada como melhor resposta.

Tabela 3.9: Tabela de especificação de caso de uso de selecionar melhor resposta

Caso de Uso	Selecionar melhor resposta	
Descrição	Selecionar a melhor resposta do tópico	
Autor	Técnico	
Pré-condição	Clicar no tópico desejado	
Pós-condição	Alterar a resposta para melhor resposta	
	Autor	Sistema
Fluxo Principal	1-Clicar em melhor resposta	
		2-Verificar se já existe uma melhor resposta - Não
		3- Colocar a resposta como melhor resposta do tópico
Fluxo Alternativo(A1)	1-Clicar em melhor resposta	
		2-Verificar se já existe uma melhor resposta - Sim
		3- Verificar se a resposta existente é a mesma selecionada - Não
		4-Alterar melhor resposta
Fluxo Alternativo(A2)	1-Clicar em melhor resposta	
		2-Verificar se já existe uma melhor resposta - Sim
		3- Verificar se a resposta existente é a mesma selecionada - Sim
		4-Remover melhor resposta

### 3.11.1.6 Especificação de caso de uso de eliminar tópico

O técnico sempre que desejar poderá eliminar o tópico, o que permite remover do fórum e não volta a ser apresentado.

Tabela 3.10: Tabela de especificação do caso de uso de eliminar tópico

Caso de Uso	Eliminar tópico	
Descrição	Eliminar um tópico do fórum	
Autor	Técnico	
Pré-condição	Clicar no tópico desejado	
Pós-condição	Remoção do tópico	
	Autor	Sistema
Fluxo Principal	1-Clicar em remover tópico	
		3-Remover o tópico
Fluxo Alternativo(A1)	-	-

### 3.11.1.7 Especificação de caso de uso de alterar visibilidade de um tópico

Quando um técnico pública um tópico este pode desejar alterar a sua visibilidade para apenas técnicos oficiais ou todos os técnicos.

Tabela 3.11: Tabela de especificação de caso de uso de alteração de visibilidade de um tópico

Caso de Uso	Alterar visibilidade do tópico	
Descrição	Alterar a visibilidade de um tópico entre público e privado	
Autor	Técnico	
Pré-condição	Clicar no tópico desejado	
Pós-condição	Alterar visibilidade do tópico	
	Autor	Sistema
Fluxo Principal	1-Clicar em alterar visibilidade	
		2-Inverter visibilidade do tópico
Fluxo Alternativo(A1)	-	-

### 3.11.1.8 Especificação de caso de uso gostar de um tópico

O técnico sempre que encontra um tópico que identifica como útil, este poderá colocar *like* o que gera destaque.

Tabela 3.12: Tabela de especificação de caso de uso de gostar de um tópico

Caso de Uso	Gostar do tópico	
Descrição	Gostar de um tópico	
Autor	Técnico	
Pré-condição	Clicar no tópico desejado	
Pós-condição	Alterar gostos do tópico	
	Autor	Sistema
Fluxo Principal	1-Clicar em gosto	
		2-Verificar se o gosto já existe - Não
		3-Aumentar gosto ao tópico
Fluxo Alternativo(A1)	1-Clicar em gosto	
		2-Verificar se o gosto já existe - Sim
		3-Remover gosto do tópico

### 3.11.1.9 Especificação de caso de uso gostar de um comentário

Sempre que um técnico identificar um comentário como útil este poderá colocar *like* o que gera destaque.

Tabela 3.13: Tabela de especificação de caso de uso de gostar de comentário

Caso de Uso	Gostar de comentário	
Descrição	Gostar de um comentário	
Autor	Técnico	
Pré-condição	Clicar no tópico desejado	
Pós-condição	-	
	Autor	Sistema
Fluxo Principal	1-Clicar em gosto	
		2-Verificar se o gosto já existe - Não
		3-Aumentar gosto ao comentário
Fluxo Alternativo(A1)	1-Clicar em gosto	
		2-Verificar se o gosto já existe - Sim
		3-Remover gosto do tópico

### 3.11.1.10 Especificação de caso de uso de comentar tópico

Sempre que um técnico encontra um tópico sobre uma questão que poderá ajudar, este consegue responder através de um comentário, onde este também poderá adicionar imagens.

Tabela 3.14: Tabela de especificação de caso de uso de comentar um tópico

Caso de Uso	Comentar o tópico	
Descrição	Comentar um tópico	
Autor	Técnico	
Pré-condição	Clicar no tópico desejado	
Pós-condição	Inserir a resposta no tópico	
	Autor	Sistema
Fluxo Principal	1-Indicar a descrição da resposta	
	2-Anexar Imagem	
	3-Confirmar a resposta	
		4-Inserir novo comentário no tópico
Fluxo Alternativo(A1)	1-Indicar a descrição da resposta	
	2-Confirmar a resposta	
		3-Inserir novo comentário no tópico
Fluxo Alternativo(A2)	1-Cancelar a criação do tópico	

### 3.11.1.11 Especificação de caso de uso ativar conta

O técnico para ativar a sua conta deverá introduzir o código de ativação correto, caso contrário não será possível ativar.

Tabela 3.15: Tabela de especificação de caso de uso ativação de conta

Caso de Uso	Ativar conta	
Descrição	Ativar conta de aplicação	
Autor	Técnico	
Pré-condição	-	
Pós-condição	-	
	Autor	Sistema
Fluxo Principal	1-Inserir código de validação	
	2-Validar conta	
		3-Verificar se o código está correto-Sim
		4-Validar conta
Fluxo Alternativo(A1)	1-Cancelar Ativação de conta	
Fluxo Alternativo(A2)	1-Inserir código de validação	
	2-Validar conta	
		3-Verificar se o código está correto-Não
		4-Código de validação incorreto

### 3.11.1.12 Especificação de caso de uso configurar notificações

As notificações da aplicação poderão ser personalizadas, o que possibilita escolher entre *email*, *push* e ambos, para além destas configurações, é também possível personalizar o tipo de notificação para cada método, seja relatório diário de todas as notificações ou notificações em tempo real.

Tabela 3.16: Tabela de especificação de caso de uso de configuração de notificações

Caso de Uso	Configuração de notificações	
Descrição	Configuração de notificações do técnico	
Autor	Técnico	
Pré-condição	-	
Pós-condição	-	
	Ator	Sistema
Fluxo Principal	1-Indicar preferência de receção de notificações	
	2-Indicar tipo de receção de notificações	
Fluxo Alternativo(A1)	1-Ver notificações	

### 3.11.1.13 Especificação de caso de uso registar técnico

Sempre que uma empresa deseja realizar o registo de técnicos em seu nome, esta poderá indicar o nº contribuinte e *email*, com isto, este receberá um *email* para confirmar o registo de conta.

Tabela 3.17: Tabela de especificação de caso de uso de registar técnico

Caso de Uso	Registar técnico	
Descrição	Registar conta de técnico da empresa	
Autor	Empresa	
Pré-condição	-	
Pós-condição	-	
	Ator	Sistema
Fluxo Principal	1-Indicar o nºcontribuinte	
	2-Indicar <i>email</i>	
		3-Registrar técnico
Fluxo Alternativo(A1)	-	-

### 3.11.2 Diagramas de casos de uso

Para ser possível visualizar graficamente todas as ações que os atores conseguem realizar e para melhorar a comunicação com as partes interessadas do projeto, foram desenvolvidos diagramas de casos de uso.

#### 3.11.2.1 Casos de uso Fórum

Na Figura 3.11, é possível visualizar o diagrama de casos de uso para o fórum. Neste, o técnico poderá ver as listagens de tópicos em destaque e tópicos mais recentes. Caso seja um técnico oficial terá acesso aos tópicos privados e os seus tópicos. O técnico poderá também pesquisar por tópicos, dos quais ele terá a possibilidade de selecionar um para visualizar. O técnico conseguirá, além disso, criar um novo tópico, onde se dirigirá para a criação de tópicos. Aqui, o técnico será obrigado a inserir um título, descrição e tipo do tópico para o criar, mas poderá inserir imagens e indicar produto referente. Para finalizar a criação o técnico conseguirá confirmar ou cancelar o processo.

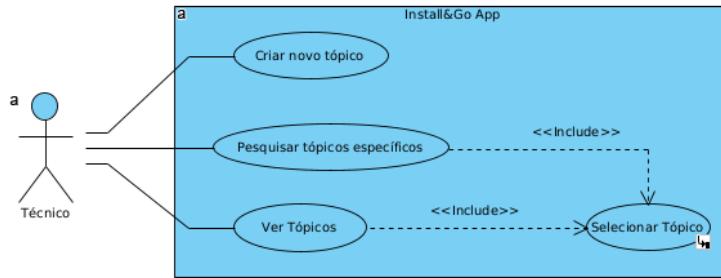


Figura 3.11: Diagrama de casos de uso de fórum

#### 3.11.2.2 Casos de uso de pesquisar tópicos

O técnico poderá realizar a pesquisa por tópicos específicos, esta será ser realizada por escrito onde indica o assunto a pesquisar e poderá ser filtrada.

O técnico terá também a possibilidade de pesquisar por código QR de produto, uma vez que, o servidor da Motorline esteja desenvolvido para tal.

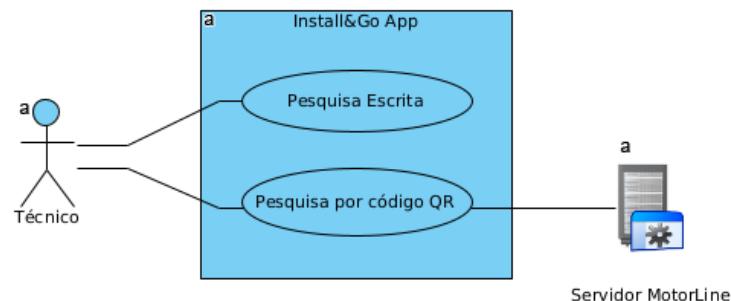


Figura 3.12: Diagrama de casos de uso de pesquisa de tópicos

### 3.11.2.3 Casos de uso ver detalhes de tópico

Assim que um técnico seleciona um tópico, é movido para os detalhes, onde consegue visualizar os detalhes, responder e, caso seja o seu tópico, consegue finalizar, selecionar a melhor resposta, remover a melhor resposta, eliminar e alterar a visibilidade do tópico.

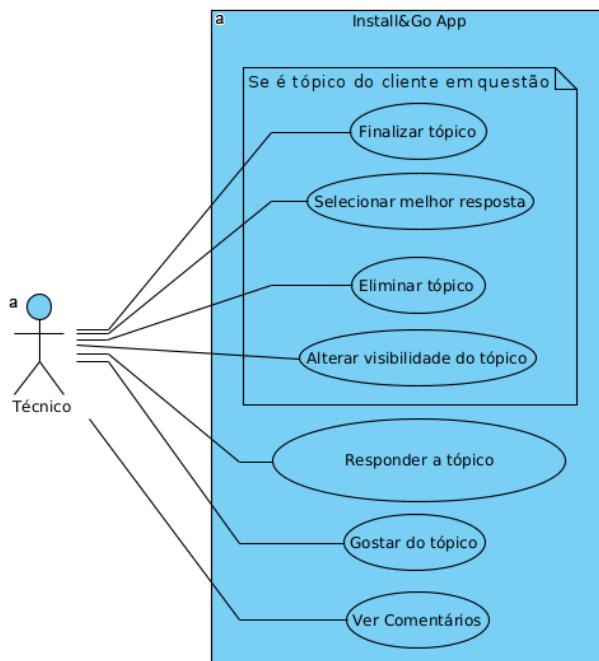


Figura 3.13: Diagrama de casos de uso de detalhes de tópico

### 3.11.2.4 Casos de uso ver comentários

O técnico quando decide visualizar os comentários consegue responder e gostar de uma resposta ou comentário, caso este seja seu ainda o consegue apagar.

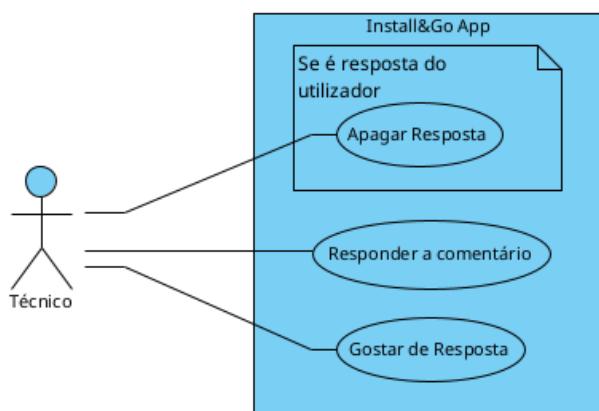


Figura 3.14: Diagrama de casos de uso de ver comentários

### 3.11.2.5 Casos de uso ativação de conta

Assim que uma conta é confirmada, um *email* de ativação é enviado para técnico e esta deverá ser ativada. Para isto, o código deverá ser indicado pelo técnico para se proceder à ativação da conta. Este, poderá em caso de necessidade, pedir o reenvio do código de ativação, o qual será gerado novamente e reenviado.

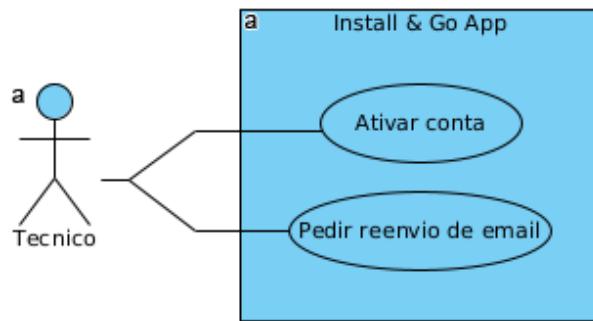


Figura 3.15: Diagrama de casos de uso de ativação de conta

### 3.11.2.6 Casos de uso perfil

Sempre que o técnico desejar alterar alguma informação, este poderá alterar o seu nome, *email* e imagem de perfil.

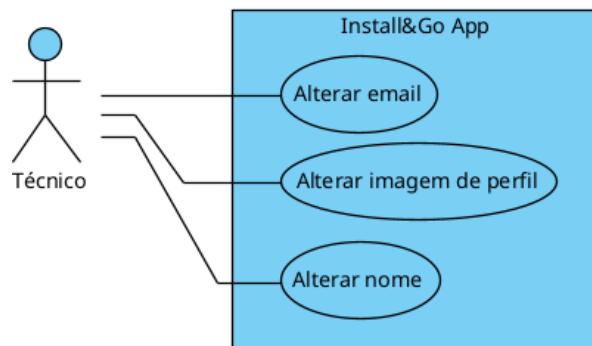


Figura 3.16: Diagrama de casos de uso de perfil

### 3.11.2.7 Casos de uso notificações

Sempre que o técnico desejar ver as suas notificações, poderá selecioná-las, também dispõe da possibilidade de alterar a configuração das notificações, para apenas as receber por *email* ou push, ou então, ambas. Terá também a possibilidade de personalizar cada método, para receber um relatório diário de notificações ou então, notificações em tempo real.

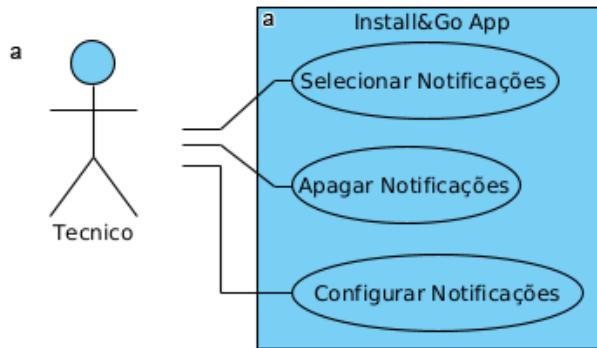


Figura 3.17: Diagrama de casos de notificações

### 3.11.2.8 Casos de uso gestão de recursos humanos

Uma empresa poderá registar contas para os seus técnicos no seu nome, com a indicação do *email* e nº contribuinte. Esta poderá também impedir acesso a estas contas ou remover completamente a conta da aplicação.

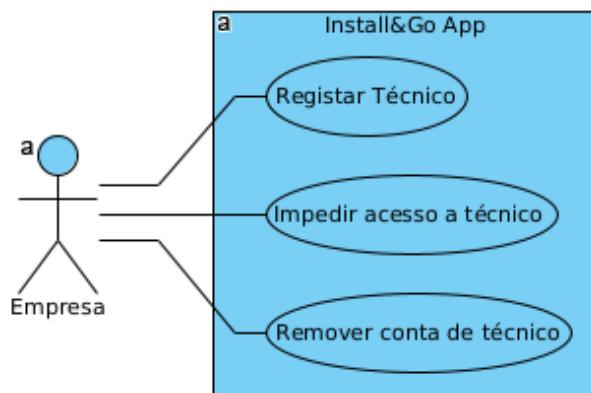


Figura 3.18: Diagrama de casos de uso de recursos humanos

### 3.12 Diagrama Entidade Relação

O software Install&Go é suportado por uma base de dados relacional esquematizada de acordo com as necessidades do projeto. Para garantir que os tipos dos atributos se encontram corretos, foram criados novos tipos, já para simular a criação de *id's* do tipo *uuid* foi criado um *stored procedure* o que resulta na tabela de cor verde na Figura 3.19.

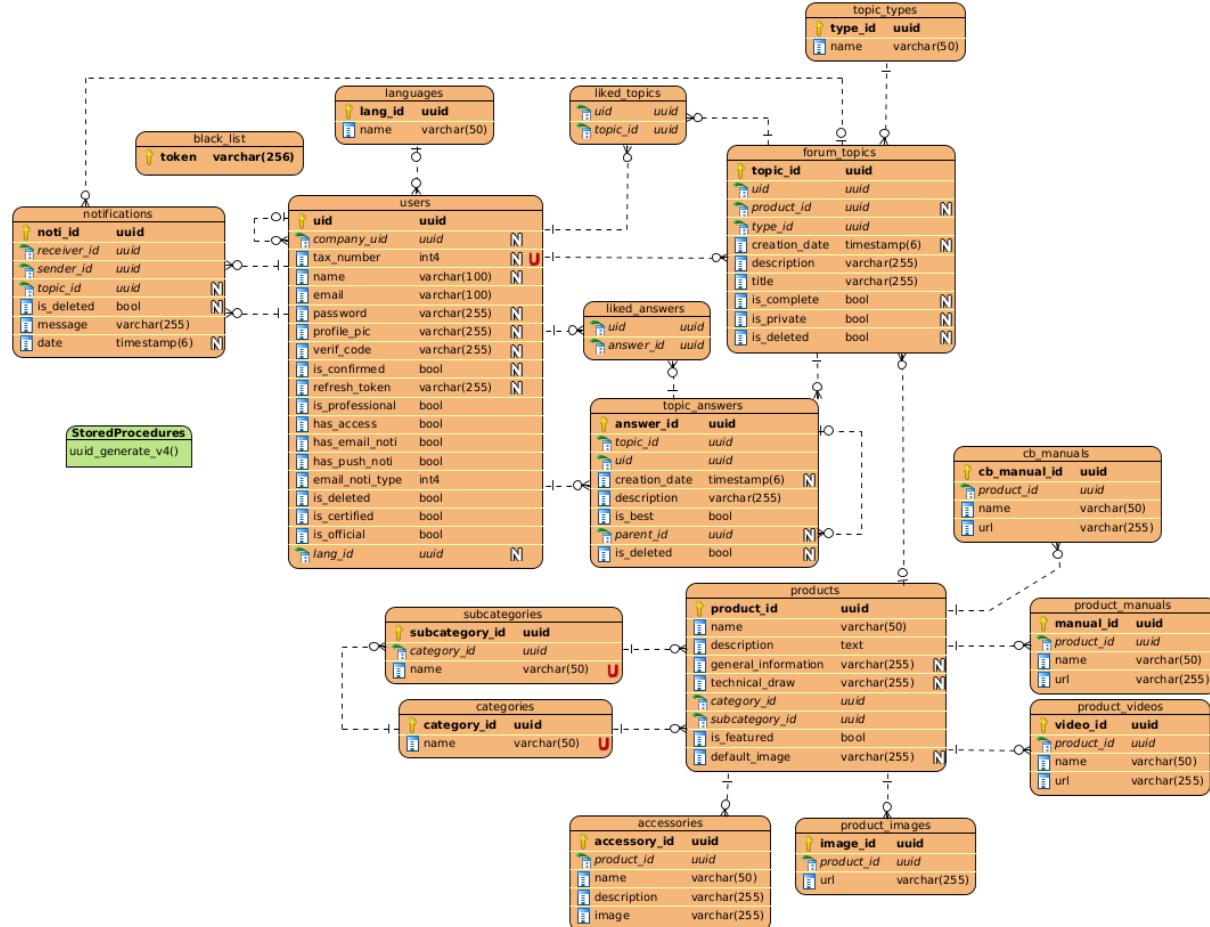


Figura 3.19: Diagrama Entidade Relação base de dados Install&Go

### 3.12.1 Escolha de diagramma de entidade relação

Durante o desenvolvimento do diagramma de entidade relação, surgiu a opção de separar as empresas dos seus técnicos em duas tabelas como exemplificado na Figura 3.20. Neste sempre que se deseja, por exemplo, obter o utilizador que criou um tópico é necessário verificar se o *uid* contido é de uma empresa ou de um técnico e apenas de seguida se obter o utilizador que criou o tópico, sendo este um exemplo entre os demais do mesmo tipo. Tendo em conta este problema foi decidido optar pelo diagramma da Figura 3.19.

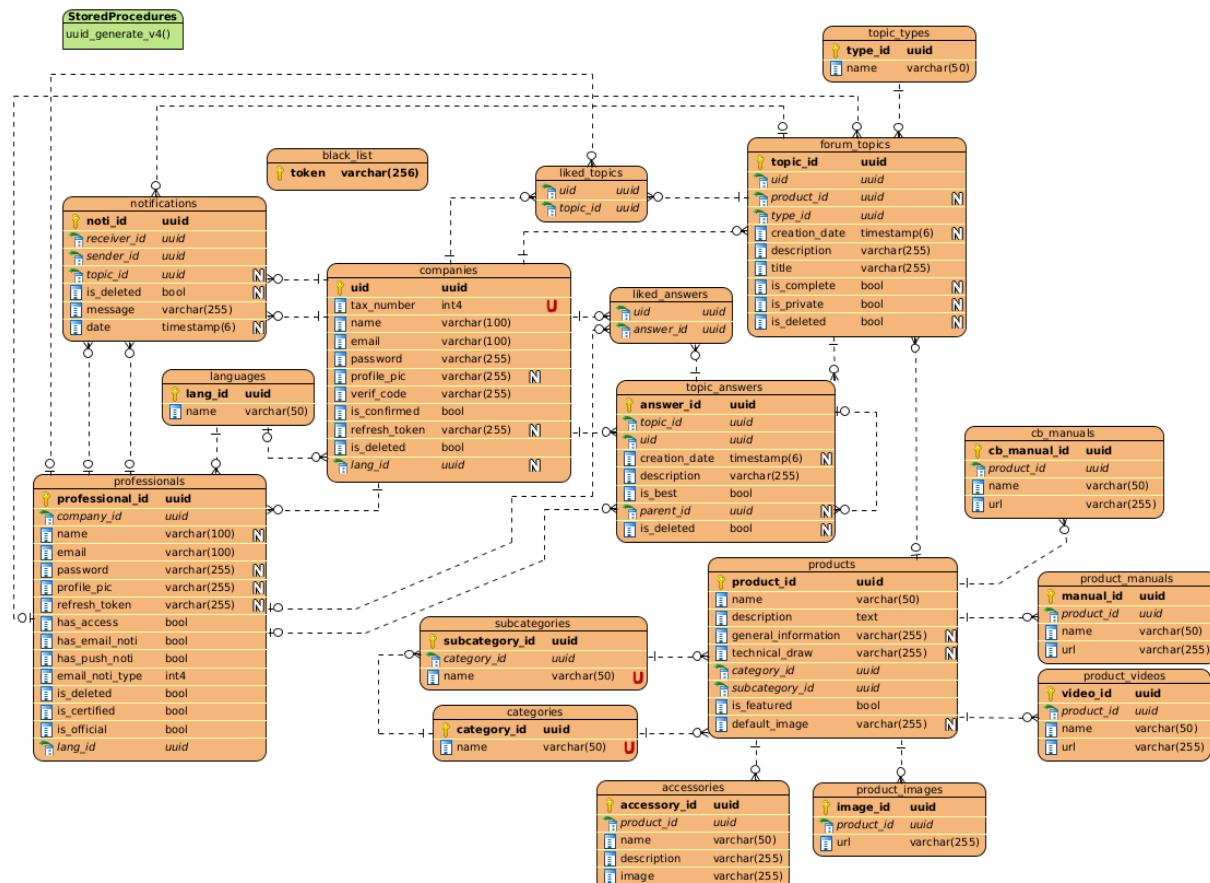


Figura 3.20: Diagramma Entidade Relação alternativo

### 3.12.2 Dicionário de termos

Com intuito de alcançar o propósito de cada tabela e atributo foi criado um dicionário de termos para a base de dados(Tabela 3.18).

Tabela 3.18: Dicionário de termos da base de dados

Tabela	Descrição	Atributos	Descrição
users	Tabela encarregue de guardar todos os dados referentes aos utilizadores da aplicação	uid	<i>id</i> do utilizador
		company_id	<i>id</i> da empresa referente ao técnico
		tax_number	Número de contribuinte do utilizador
		name	Nome do utilizador
		email	Email do utilizador
		password	Password do utilizador
		profile_pic	Imagen de perfil do utilizador
		verif_code	Código de verificação do utilizador
		is_confirmed	Verificação de se o código está confirmado
		refresh_token	Token de refresh
		is_professional	Verificação se é profissional
		has_access	Verificação se tem acesso à conta
		has_email_noti	Verificação se ativou notificações de email
		has_push_noti	Verificação se ativou notificações push
		email_noti_type	Tipo de notificação de email
		push_noti_type	Tipo de notificação push
		is_deleted	Verificação se a conta se encontra apagada
		is_certified	Verificação se é um técnico certificado
		is_official	Verificação se é um técnico oficial

Continua na página seguinte

Tabela 3.18: Dicionário de termos da base de dados (Continuação)

		lang_id	<i>id</i> da linguagem preferencial do utilizador
black_list	Tabela que guarda os tokens a bloquear	token	Token a bloquear
liked_topics	Tabela encarregue de guardar todos os tópicos que foram gostados pelo utilizador	uid	<i>id</i> do utilizador
		topic_id	<i>id</i> do tópico
liked_answers	Tabela encarregue de guardar todas as respostas que receberam gosto do utilizador	uid	<i>id</i> do utilizador
		answer_id	<i>id</i> da resposta
topic_types	Tabela encarregue de guardar os tipos de tópico existentes	type_id	<i>id</i> do tipo de tópico
		name	nome do tipo de tópico
notifications	Tabela encarregue de guardar todas as notificações do técnico	noti_id	<i>id</i> da notificação
		receiver_id	Receptor da notificação
		sender_id	Emissor da notificação
		topic_id	<i>id</i> do tópico em caso de estar referente a um tópico
		is_deleted	Verificação se a notificação está apagada
		message	Mensagem da notificação
		date	Data de emissão da notificação

Continua na página seguinte

Tabela 3.18: Dicionário de termos da base de dados (Continuação)

forum_topics	Tabela encarregue de guardar todos os tópicos existentes na aplicação	topic_id	<i>id</i> do tópico
		uid	<i>id</i> do dono do tópico
		product_id	Produto referente ao tópico
		ttype_id	<i>id</i> do tipo referente ao tópico
		creation_date	Data de criação do tópico
		description	Descrição do tópico
		title	Titulo do tópico
		is_complete	Verificação se o tópico está finalizado
		is_private	Verificação se o tópico é privado
		is_deleted	Verificação se o tópico está apagado
topic_answers	Tabela encarregue de guardar todas as respostas a um tópico	answer_id	<i>id</i> da resposta
		topic_id	<i>id</i> do tópico
		uid	<i>id</i> do dono da resposta
		creation_date	Data de criação da resposta
		description	Descrição da resposta
		is_best	Verificação se é a melhor resposta
		parent_id	<i>id</i> da resposta pai
		is_deleted	Verificação se o topico se encontra apagado
categories	Tabela encarregue de guardar todas as categorias de produtos existentes	category_id	<i>id</i> da categoria
		name	Nome da categoria
subcategories	Tabela encarregue de guardar as subcategorias de produtos existentes	subcategory_id	<i>id</i> da subcategoria
		category_id	<i>id</i> da categoria
		name	Nome da subcategoria

Continua na página seguinte

Tabela 3.18: Dicionário de termos da base de dados (Continuação)

cb_manuals	Tabela encarregue de guardar os manuais de utilização das placas de controlo	cb_manual_id	<i>id</i> do manual
		product_id	<i>id</i> do produto
		name	Nome da placa de controlo
		url	Url do manual
products	Tabela encarregue de guardar as informações dos produtos do catálogo da empresa	product_id	<i>id</i> do produto
		name	Nome do produto
		description	Descrição do produto
		general_information	Url da informação geral do produto
		technical_draw	Url do desenho técnico do produto
		category_id	<i>id</i> da categoria de produto
		subcategory_id	<i>id</i> da subcategoria de produto
		is_featured	Verificação se o produto é um destaque
		default_image	Imagen do produto por omissão
product_manuals	Tabela encarregue de guardar os manuais de utilização dos produtos	cb_manual_id	<i>id</i> do manual
		product_id	<i>id</i> do produto
		name	Nome do manual
		url	Url do manual
product_videos	Tabela encarregue de guardar os videos de cada produto	cb_manual_id	<i>id</i> do video
		product_id	<i>id</i> do produto
		name	Nome do video
		url	Url do video
languages	Tabela encarregue de guardar as linguagens suportadas pela aplicação	lang_id	<i>id</i> da linguagem
		name	Nome da linguagem

### 3.13 Diagrama de Classes

A fim de prever e organizar o *software* foi desenvolvido um diagrama de classes (Figura 3.21) que permite visualizar cada classe que se espera conter no *software*, assim como também os seus atributos e métodos.

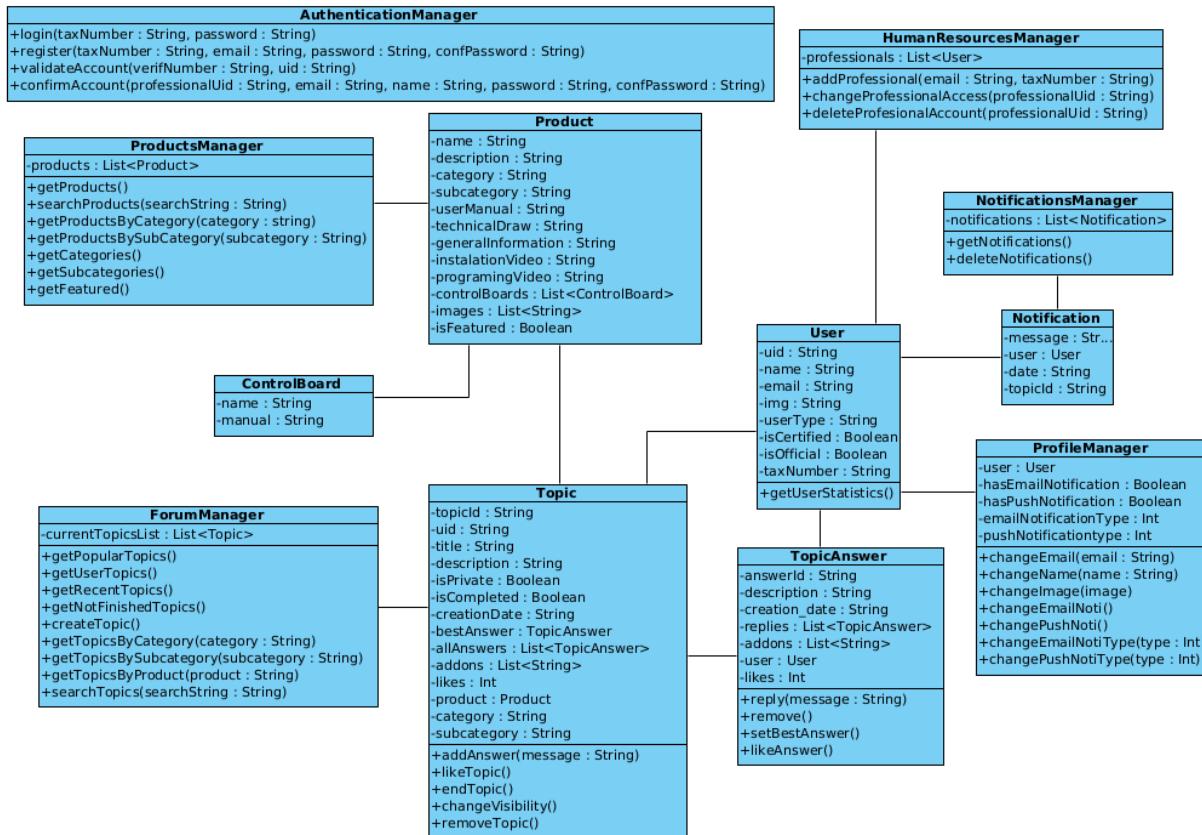


Figura 3.21: Diagrama de classes Install&Go

## 3.14 Mockups

Com o propósito de criar um *design* para seguir e apresentar às partes interessadas antes de iniciar a fase de desenvolvimento, então foram realizadas *mockups* do *design* da aplicação. Este *design* foi iterativamente revisto pelo cliente e ajustado até alcançar o estado final.

### 3.14.1 Página Inicial

A página inicial da aplicação, dá ao utilizador a possibilidade de navegar pelos produtos do catálogo, filtrar por categorias e subcategorias, assim como realizar uma pesquisa rápida e por fim navegar para o fórum. Caso um técnico esteja com sessão iniciada este poderá visualizar o *icon* de notificações e a sua imagem de perfil.

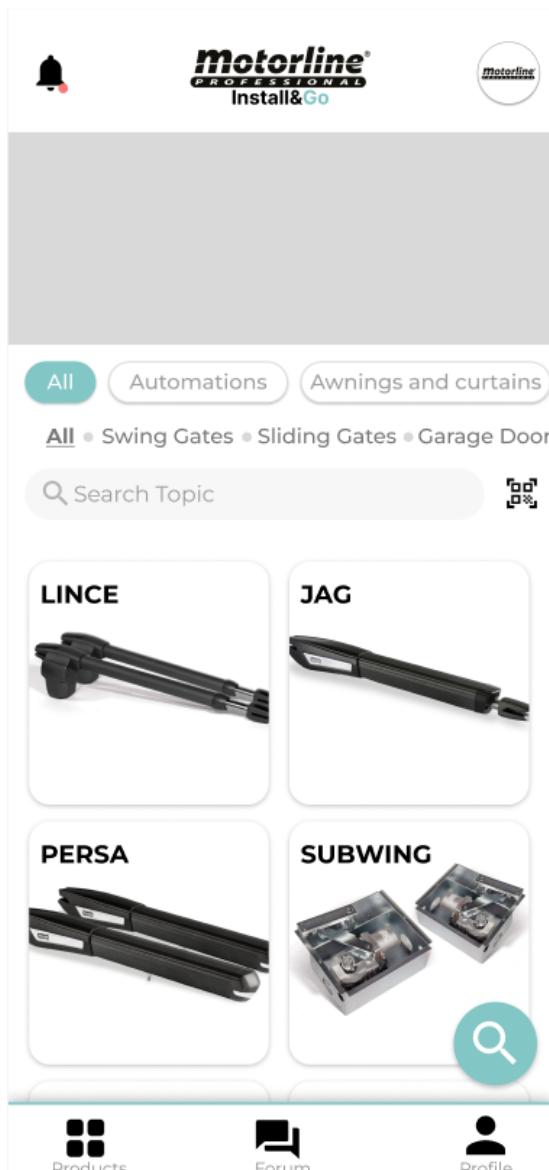


Figura 3.22: Página inicial do fórum

### 3.14.2 Autenticação - Login e Registo

Na autenticação, é possível iniciar sessão e registo, mas a empresa é a única entidade poderá realizar o registo no *software*.

**(a) Página de login**

**(b) Página de registo**

Figura 3.23: Autenticação - Login e Registo

### 3.14.3 Autenticação - Ativação e Confirmação de conta

Na autenticação também existe a página de confirmação de conta. Um técnico que tem a conta recentemente adicionada poderá confirmar o registo, indicar as informações finais e por fim será direcionado para a página de ativação onde terá de colocar o código de ativação enviado para o *email*, esta página também será aberta caso um técnico realize o *login* com uma conta que não foi ativada ou sempre que um registo é finalizado.

**(a) Página de confirmação de conta**

**(b) Página de ativação de conta**

Figura 3.24: Autenticação - Ativação e Confirmação de conta

### 3.14.4 Página inicial fórum

O técnico assim que se dirige ao fórum entrará na página inicial. Esta página permite navegar entre as diferentes listagens de tópicos acessíveis ao técnico, pesquisar, filtrar por tipo e criar um novo tópico.

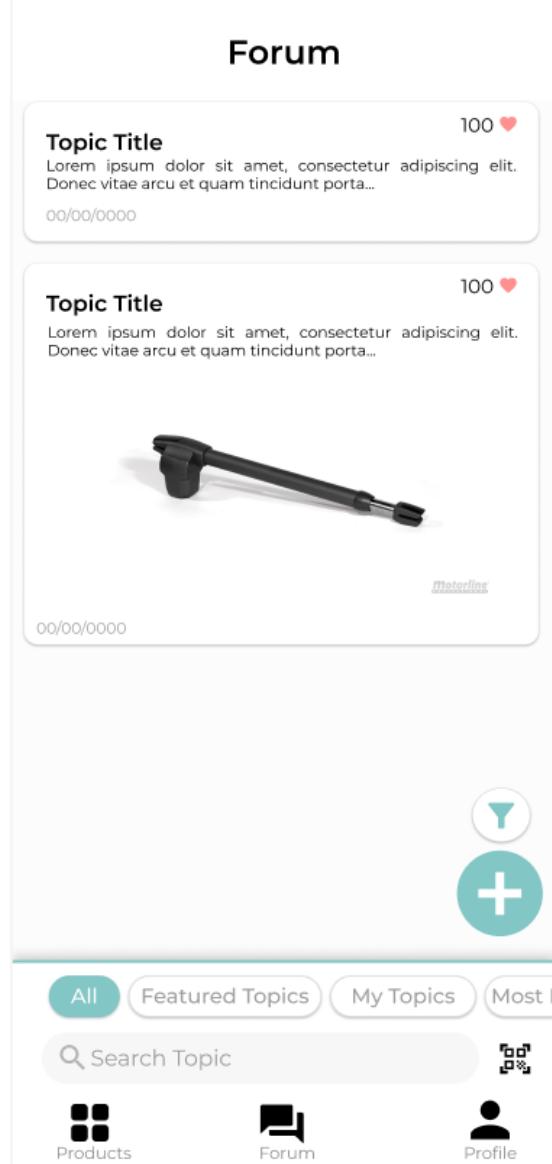


Figura 3.25: Página inicial do fórum

### 3.14.5 Página de detalhes de um tópico

Assim que o técnico seleciona um tópico, será encaminhado para a página de detalhes, onde é indicado o nome do proprietário, a imagem de perfil, a hora de criação, a quantidade de gostos, o título, a descrição, as imagens e os comentários. É possível gostar do tópico, gostar de comentários, comentar o tópico e outros comentários.

Se o tópico for do técnico que está a visualizar, poderá também concluir, eliminar e alterar a sua visibilidade.

**(a) Página de detalhes de um tópico**

**(b) Página de detalhes de um tópico  
do técnico**

Figura 3.26: Página de detalhes de tópico do software

### 3.14.6 Página de criação de um tópico

Quando um técnico inicia a criação de um tópico, é obrigado a inserir o título e a descrição. Já a indicação da visibilidade, do tipo de tópico, do produto referente e de imagens é opcional. A qualquer momento poderá cancelar ou confirmar a ação.

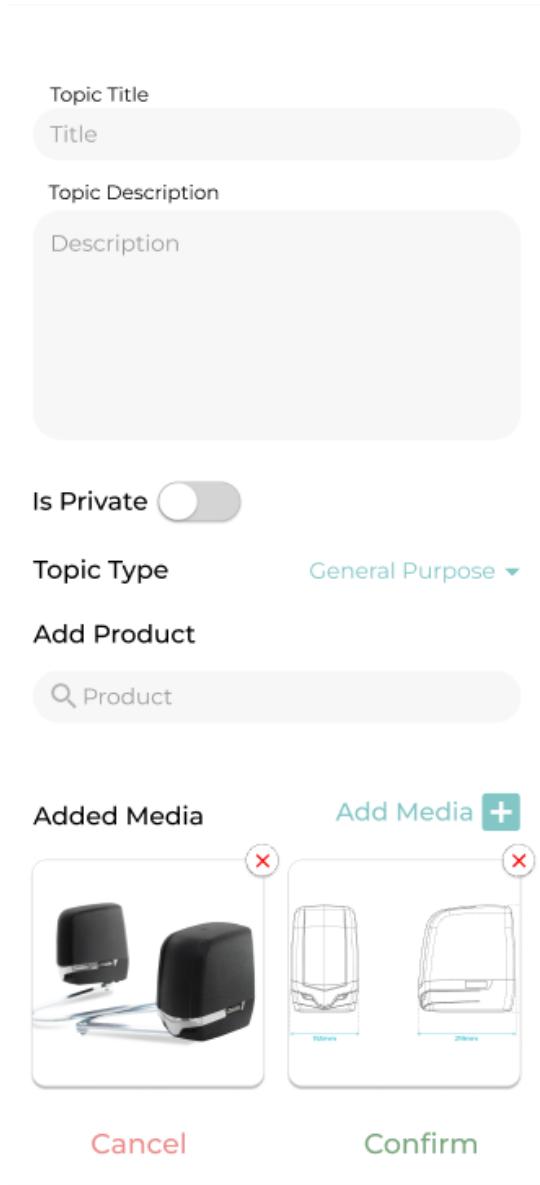


Figura 3.27: Página de criação de tópico

### 3.14.7 Página de notificações

Um técnico sempre que desejar tem como opção visualizar as suas notificações. Neste ecrã, é possível ver todas com a identificação de quem enviou, qual a descrição e a data de receção. O técnico, também tem como escolha apagar se assim desejar.

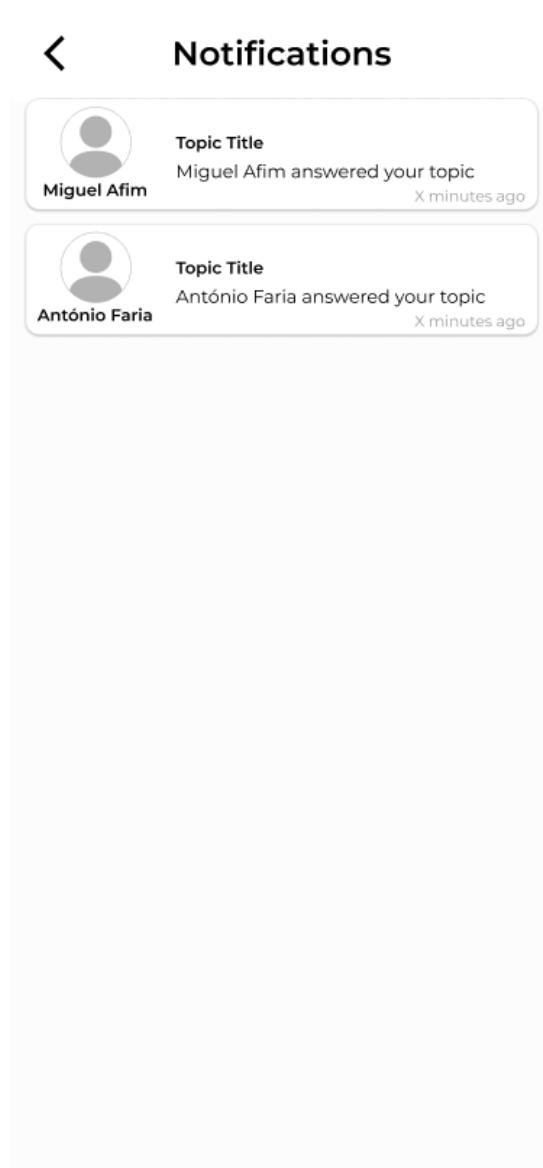


Figura 3.28: Página de notificações

### 3.14.8 Página de perfil de utilizador

O técnico, sempre que desejar alterar as suas informações, tem a possibilidade de modificar o *email*, a imagem de perfil e a configuração das notificações com indicação dos métodos e tipo a receber.

Caso uma empresa entre no perfil, esta visualizará um botão para aceder à gestão de recursos humanos.

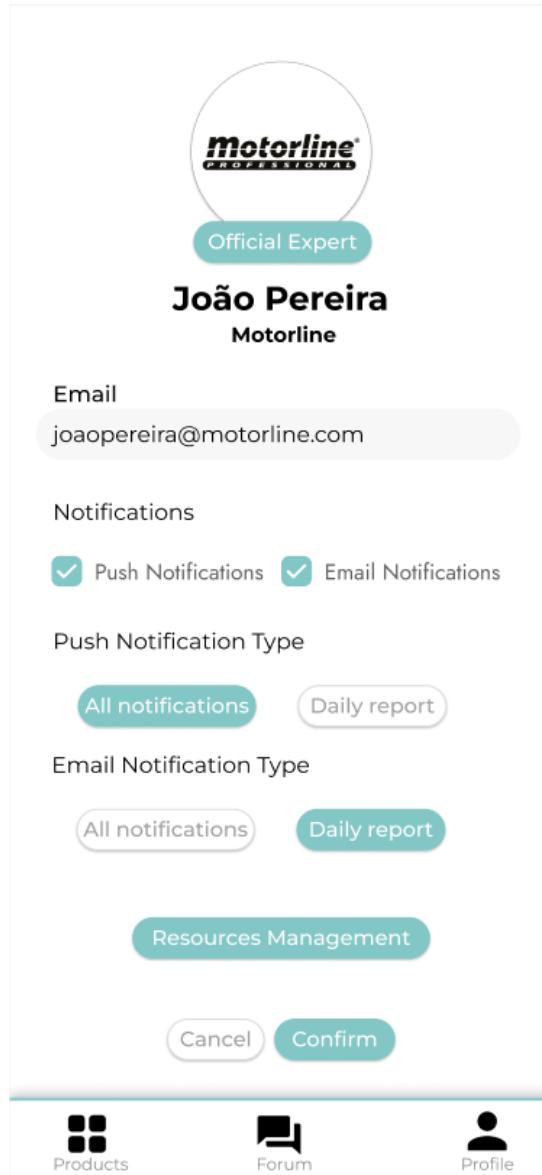


Figura 3.29: Página de perfil de utilizador

### 3.14.9 Página de gestão de recursos humanos

Na página de gestão de recursos humanos, apenas acessível a empresas, é possível registar novos técnicos, pesquisar e gerir os que já se encontram registados através dos seus perfis.

The screenshot shows a table titled "Resources Management" with two columns: "Tax Number" and "Name". Both columns contain the value "Nome Técnico" repeated eight times. Below the table is a large teal circular button with a white plus sign. At the bottom of the page are three navigation icons: "Products" (grid icon), "Forum" (speech bubble icon), and "Profile" (person icon). A search bar with the placeholder "Search Person" is also present.

Tax Number	Name
999999999	Nome Técnico

Figura 3.30: Página de gestão de recursos humanos

### 3.14.10 Página de perfil de técnico registado

O perfil de técnico, apenas acessível para empresas, permite visualizar as estatísticas, as informações e permite impedir acesso à conta ou então remover da plataforma.

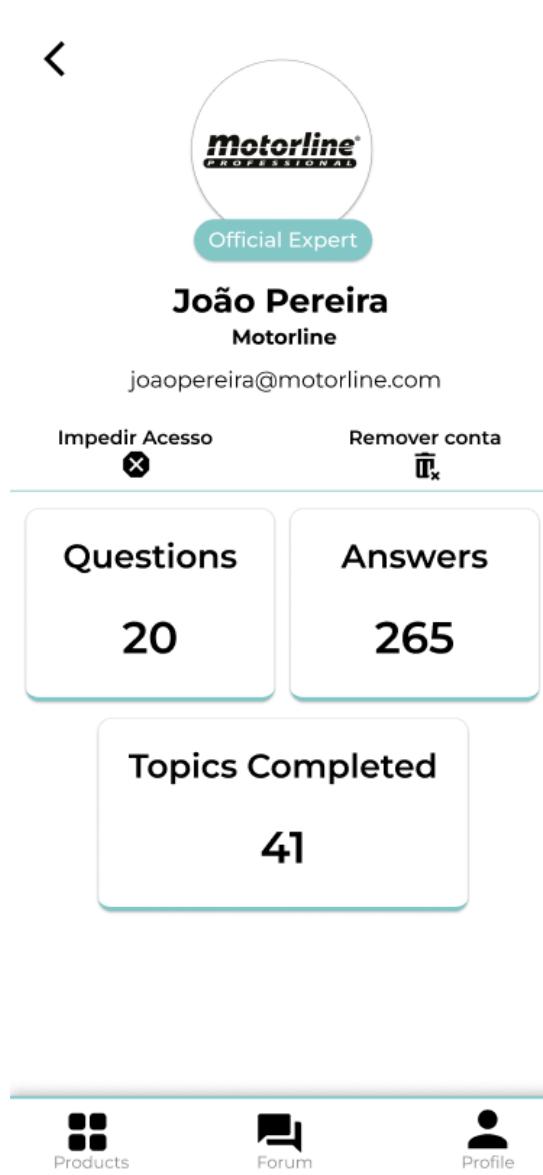


Figura 3.31: Página de perfil de técnico registado

### 3.14.11 Página de registo de novo técnico

Sempre que uma empresa deseja registar um novo técnico, esta deverá indicar o nome, o *email* e tipo de técnico.

The screenshot shows a registration form titled "Add Professional". It includes fields for "Name" and "Email", each with a placeholder text input. Below these are two checkboxes: one checked for "Certified Professional" and one unchecked for "Official Professional". A green "Register" button is at the bottom.

Name	Email
Name	Email

Certified Professional    Official Professional

**Register**

Figura 3.32: Página de registo de novo técnico

## 3.15 Diagramas de atividades

O detalhe de forma simples das ações do ator nos diferentes ecrãs foi realizado em diagramas de atividades.

### 3.15.1 Diagrama de atividades página inicial

Da página inicial da aplicação é possível deslocar para o fórum, para as notificações, para o perfil e realizar operações do catálogo.

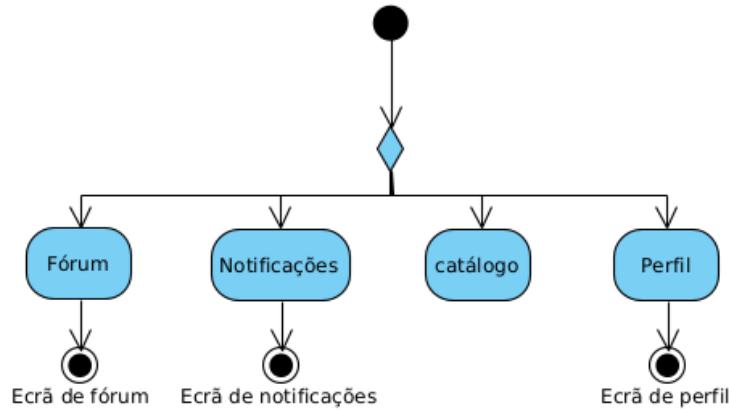


Figura 3.33: Diagrama de atividades de página inicial da aplicação

### 3.15.2 Diagrama de atividades página de perfil

Na página de perfil, é possível alterar a imagem, o nome, o *email*, selecionar os métodos e os tipos de notificação a receber. Caso uma empresa veja o perfil esta poderá, além das operações acima mencionadas, gerir os recursos humanos onde é encaminhada para o ecrã de gestão de recursos humanos.

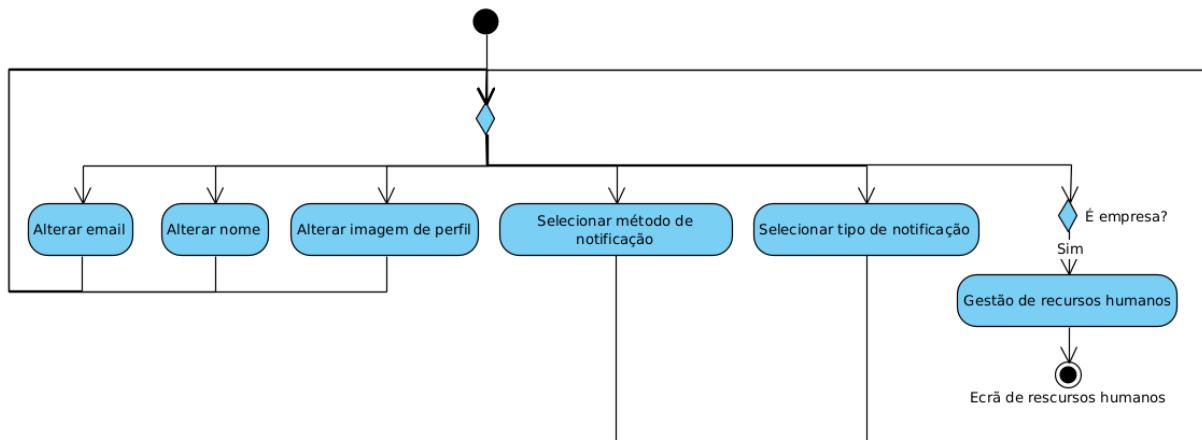


Figura 3.34: Diagrama de atividades de página de perfil

### 3.15.3 Diagrama de atividades página inicial do fórum

Na página inicial do fórum o técnico poderá selecionar um dos tipos de pesquisa, escrita ou código QR, filtrar por tipo, ver as listagens de tópicos em destaque, mais recentes, por responder, os seus tópicos e criar um novo tópico. Estas listagens poderão ser filtradas por tipo e sobre as mesmas tem a possibilidade de selecionar um tópico o que o redirecionará para o ecrã de detalhes.

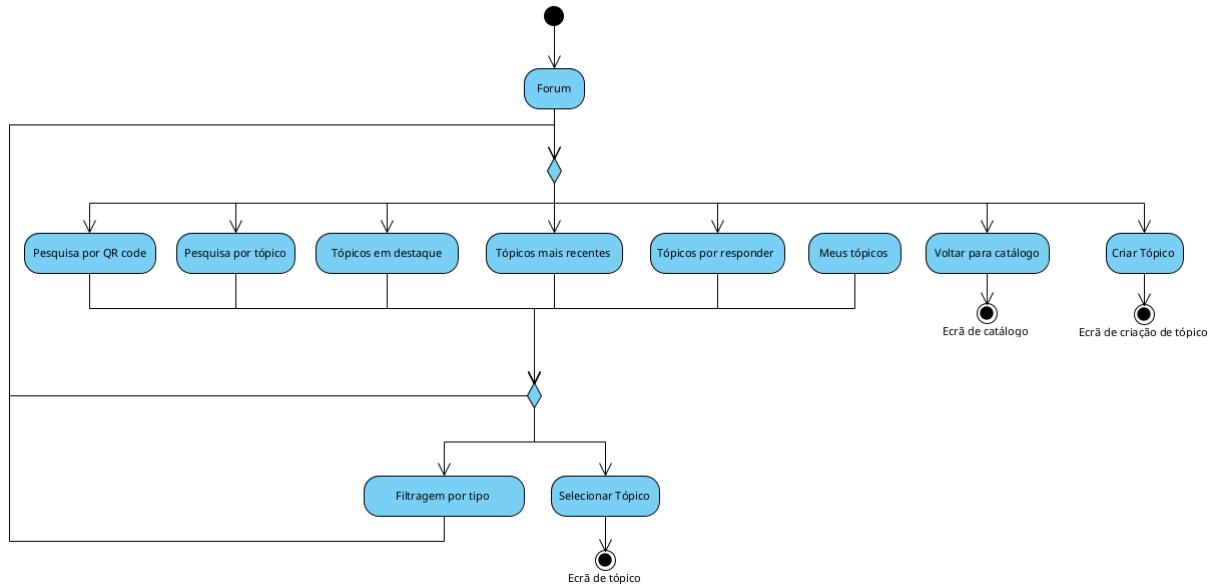


Figura 3.35: Diagrama de atividades de página inicial do fórum

### 3.15.4 Diagrama de atividades página de criação de tópico

Quando o técnico decide criar um tópico, obrigatoriamente tem de indicar o título, a descrição e o tipo do tópico. Por predefinição a visibilidade deste é pública, mas o técnico poderá alterar. Facultativamente o técnico poderá indicar o produto referente ao tópico, assim como anexar e remover imagens. A qualquer momento, o técnico poderá confirmar a criação do tópico, quando esta ação inicia, é realizada uma verificação do título e da descrição para concluir se estão preenchidos. Caso estes dados não estejam preenchidos é indicado ao técnico que as informações estão em falta, caso contrário este volta para o ecrã anterior.

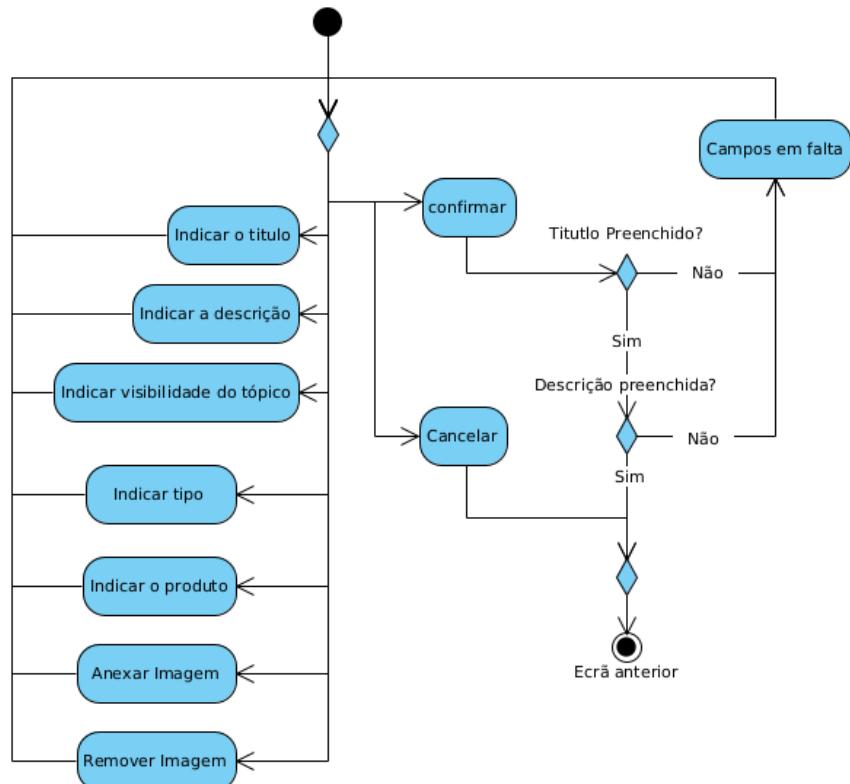


Figura 3.36: Diagrama de atividades de página de criação de tópico

### 3.15.5 Diagrama de atividades página de detalhes do tópico

Assim que o técnico seleciona um tópico, este poderá visualizar todos os comentários, apagar um caso seja seu, gostar do tópico e/ou de uma resposta, comentar e responder a um comentário. Caso o tópico seja do técnico, este poderá também alterar a visibilidade, marcar como concluído ou remover. A qualquer momento, o técnico tem como possibilidade retroceder para o ecrã anterior.

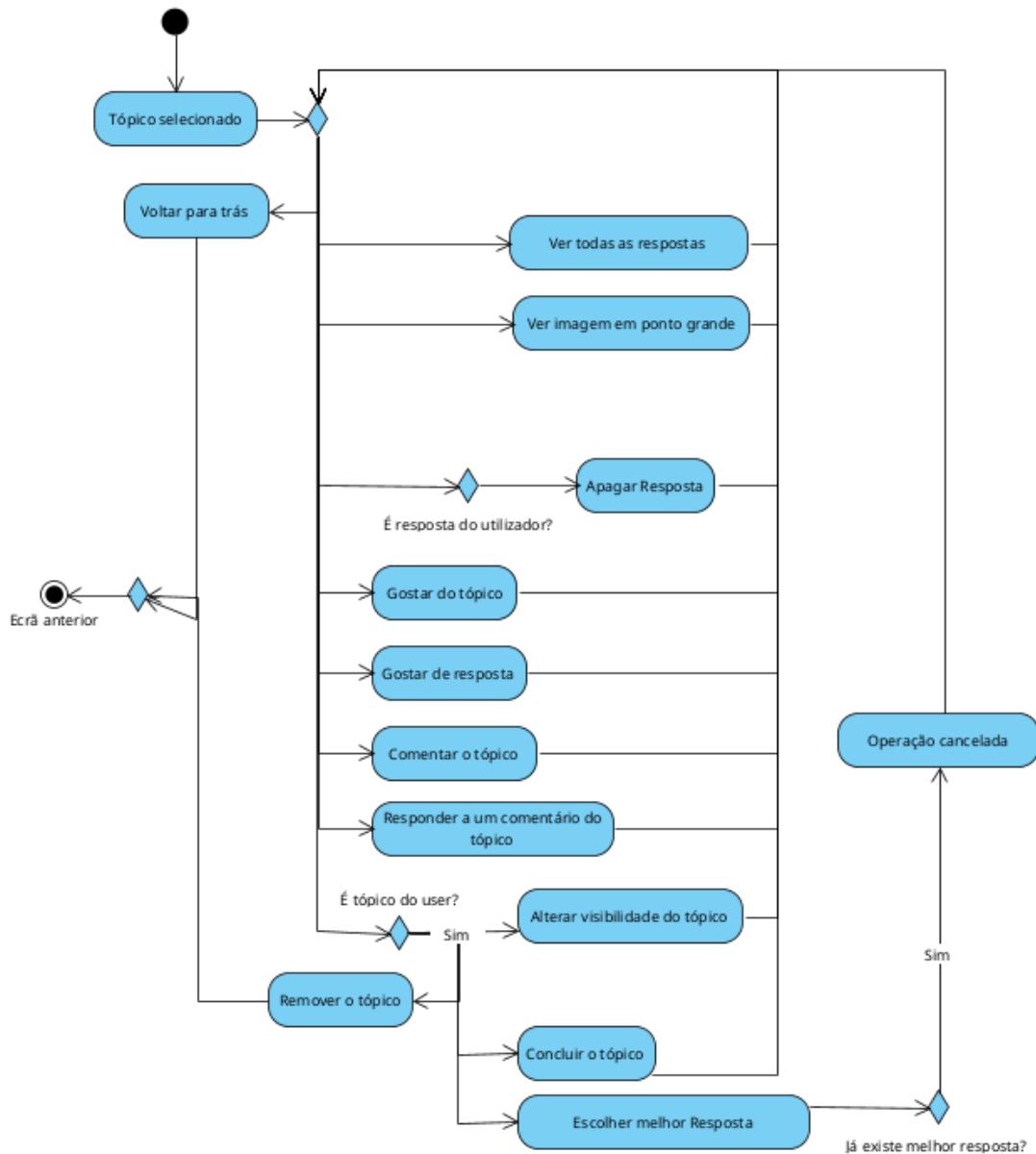


Figura 3.37: Diagrama de atividades de página de detalhes do tópico

### 3.15.6 Diagrama de atividades páginas de autenticação

Para realizar a ativação da conta do técnico, assim que este realiza o registo, confirmação da conta ou o login com uma conta que não se encontra ativa, este é encaminhado para o ecrã de ativação da conta. Neste ecrã poderá cancelar e indicar o código de ativação. Se o código estiver errado, o técnico deverá inserir-lo novamente. Por outro lado, se inserir um código correto a conta será validada e o técnico ficará autenticado. Também, em caso de necessidade, o proprietário da conta terá como opção pedir o envio de um novo código de ativação.

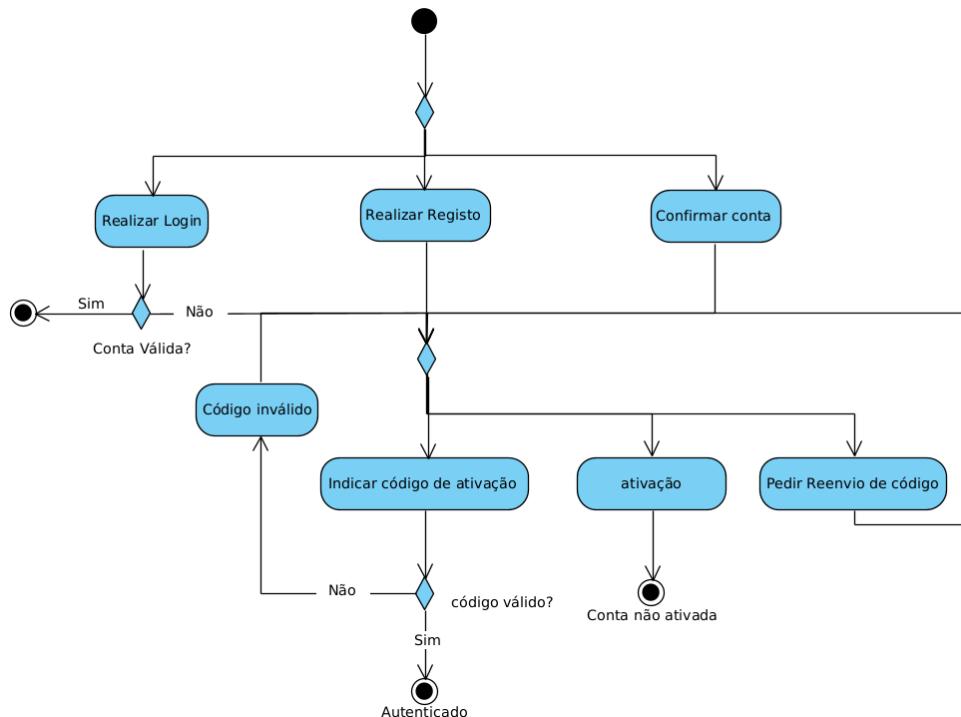


Figura 3.38: Diagrama de atividades de página de validação de conta

### 3.15.7 Diagrama de atividades registar técnico

Assim que uma empresa inicia o registo de um técnico, esta é redirecionada para a página de registo do técnico. Nesta página, terá de indicar o nome, o *email* e o tipo de técnico. Por fim será capaz de confirmar o registo da conta e automaticamente a empresa é movida para a página anterior.

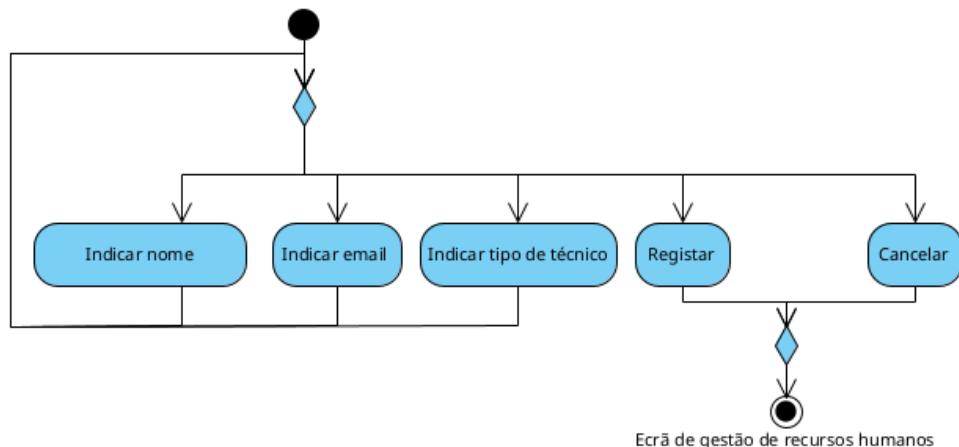


Figura 3.39: Diagrama de atividades de página de registar técnico

### 3.15.8 Diagrama de atividades confirmar conta

Quando uma conta de técnico é registada, um *email* de confirmação é enviado para o técnico. Assim que este o recebe, deverá clicar em confirmar a conta. A partir desta ação, este move-se para a página de confirmação da conta. Nesta página, o técnico poderá alterar o seu *email*, indicar o nome, a *password* e a confirmação da *password*. Todo o processo termina quando o botão de registrar for pressionado.

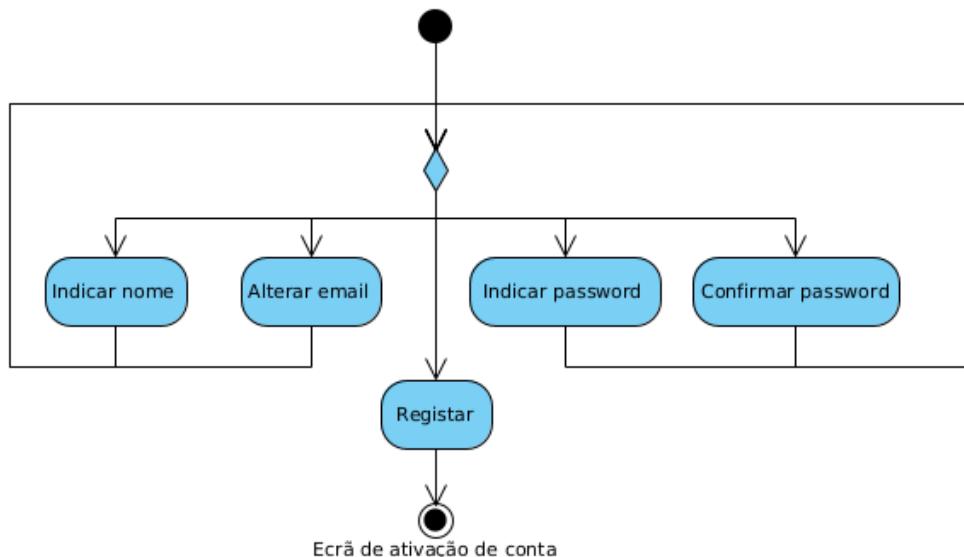


Figura 3.40: Diagrama de atividades de página de confirmar conta de técnico

## 3.16 Diagramas de estados

Para especificar os principais processos do projeto foram desenvolvidos diagramas de estados, com o objetivo de demonstrar o processo de criação de um tópico do fórum por parte de um técnico, o processo de aceder e responder a um tópico e o login com ativação de conta, visto que, estas interações são as de maior significância e regradas no *software*.

### 3.16.1 Diagrama de estados criação de tópico

Com o diagrama de estados de criação de tópico é pretendido demonstrar o processo por parte de um técnico. Assim sendo, primeiramente terá de estar autenticado, caso não esteja, será encaminhado para autenticação. De seguida criará um tópico. Após preencher os campos desejados este poderá confirmar. Caso confirme, é verificado se o tópico possui título. Caso não possua, é determinado como inválido o que levará o técnico a preencher os dados em falta. Contudo, se o título estiver preenchido, é verificado se possui descrição e tipo. Na condição de não possuir descrição ou tipo, é seguido o mesmo fluxo que o caso anterior. Caso contrário é criado um novo tópico. Por fim, na hipótese de o técnico não desejar confirmar o tópico, este poderá cancela-lo, o que o determina como cancelado.

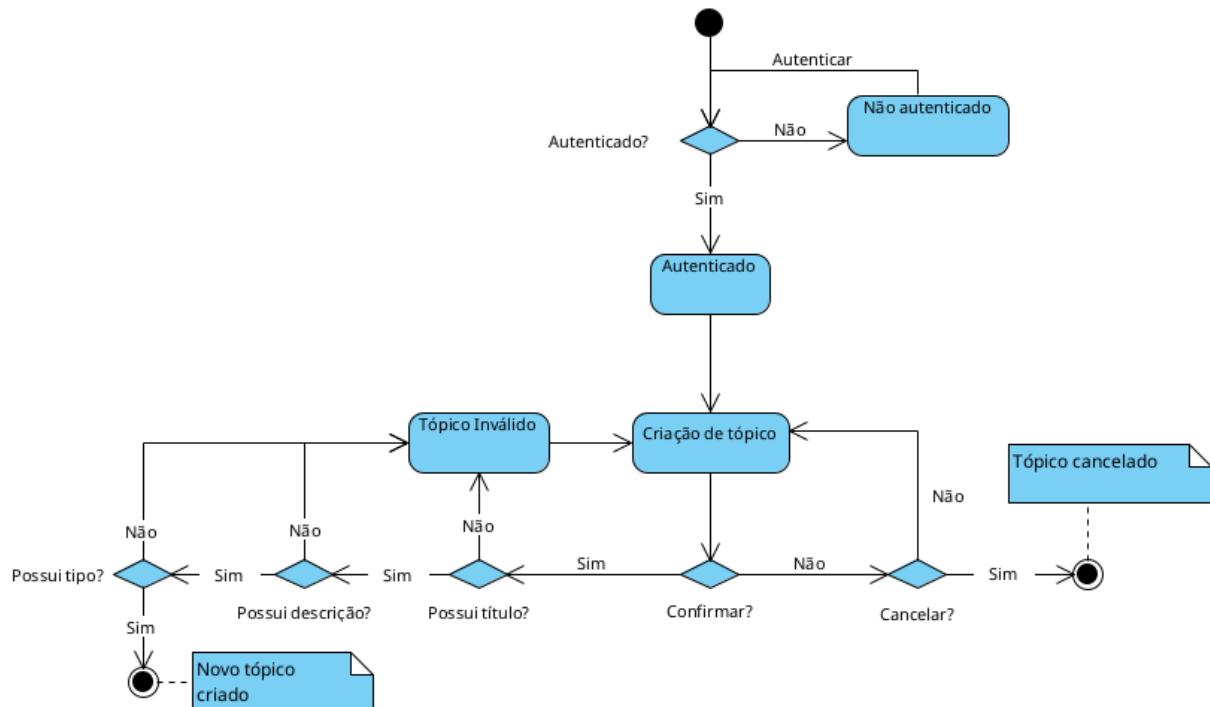


Figura 3.41: Diagrama de estados de criar tópico

### 3.16.2 Diagrama de estados responder a tópico

Com o diagrama de estados de responder a tópico é pretendido demonstrar o processo de seleção e de responder a um tópico por parte de um técnico. Assim sendo, primeiramente terá de estar autenticado, caso não esteja, será encaminhado para autenticação. Após a autenticação, o técnico irá por predefinição ver os tópicos em destaque. Nesta listagem selecionará um tópico o que permite responder. Depois da criação do comentário, este poderá confirmar e caso confirme ficará criado, caso contrário ficará cancelado.

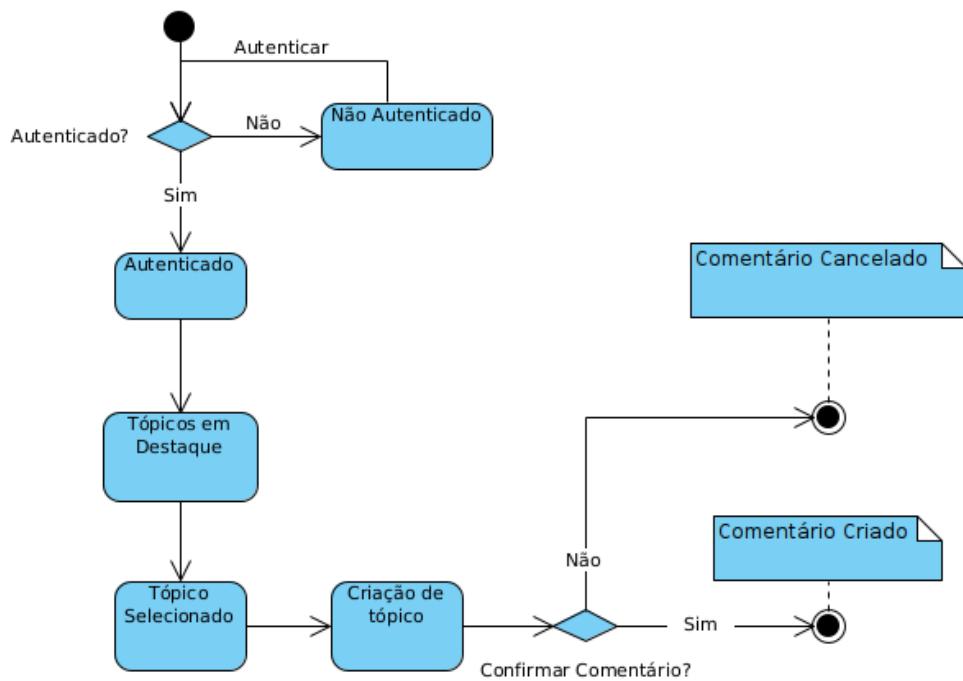


Figura 3.42: Diagrama de estados de criar tópico

### 3.16.3 Diagrama de estados autenticação e validação de conta

Aquando a realização do *login*, o técnico indicará as suas credenciais. Todavia, se não estiverem corretas, a autenticação será determinada como incorreta. Caso as credenciais estejam corretas e a conta válida, o técnico ficará automaticamente autenticado. Contudo, na condição de não conter uma conta válida, esta deverá ser validada e para isso, este terá de inserir o código de validação. No caso de o código estar correto, a conta será validada e o técnico ficará autenticado. Caso contrário o código será inválido e este necessitará indicar o código de validação novamente.

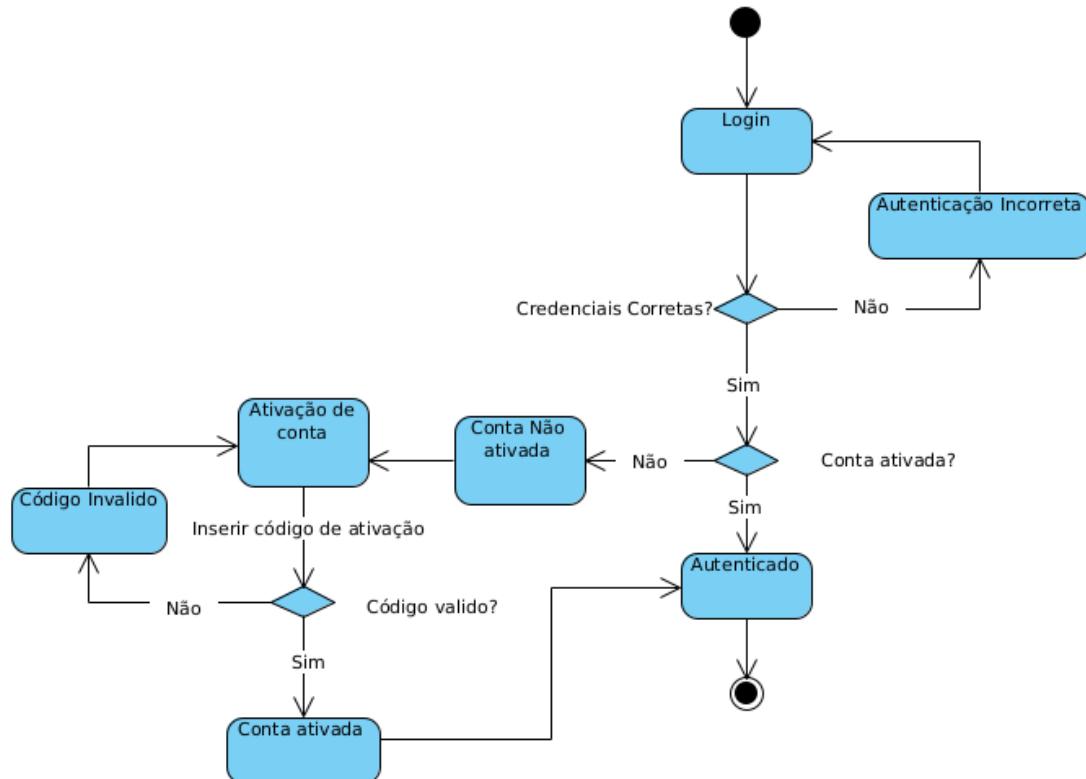


Figura 3.43: Diagrama de estados de autenticação e validação de conta

## 3.17 Diagrama de sequência

A realização da autenticação, ativação e confirmação da conta requerem procedimentos extras e seguem determinadas regras. Deste modo, foi necessário elaborar diagramas de sequência para especificar as interações com o sistema.

### 3.17.1 Diagrama de sequência Login e ativação de conta

O objetivo do diagrama da figura 3.44 consiste em demonstrar, que assim que o técnico realiza o login, é necessário verificar as credenciais. Se estas estiverem incorretas, este receberá uma mensagem de erro. Contudo, caso estas estejam válidas e a conta estiver ativada o técnico ficará autenticado.

Se porventura o técnico colocar as credenciais corretas, mas a conta não estiver ativada, este irá realizar a ativação da conta, onde poderá enviar o código de ativação. Se eventualmente estiver correto, a conta será ativada, caso contrário, este receberá uma mensagem de erro. Porém, o técnico também terá como hipótese cancelar a ativação da conta e/ou pedir um novo *email* de ativação, onde será pedido um novo código ao servidor e este automaticamente será gerado e enviado.

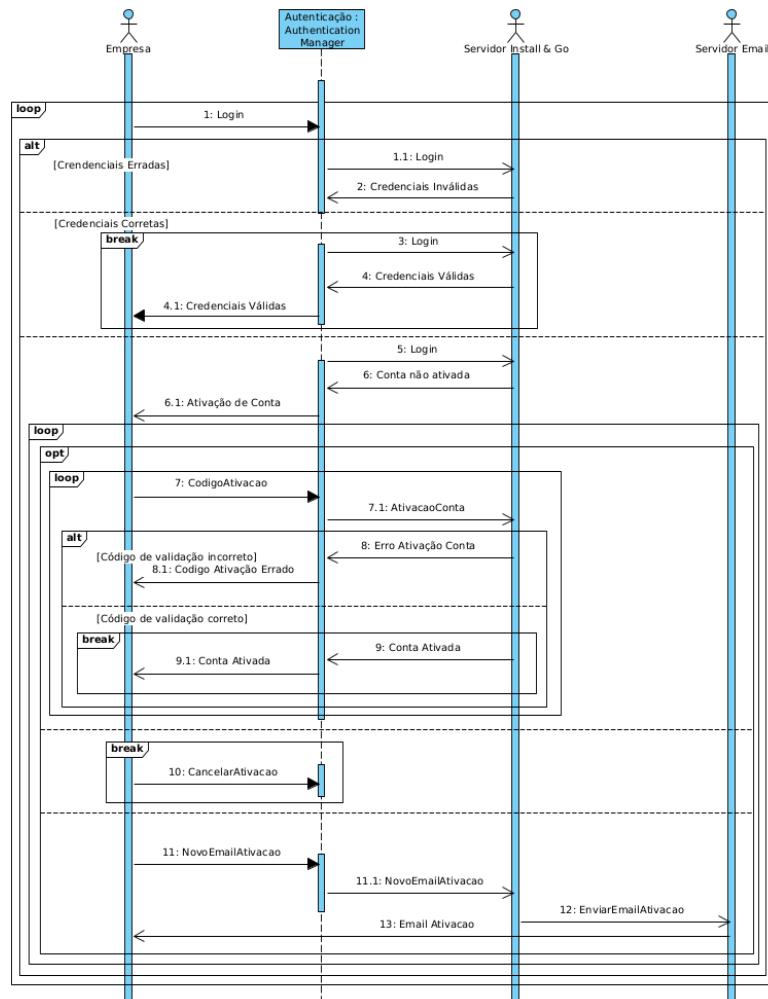


Figura 3.44: Diagrama de sequência de login e ativação de conta

### 3.17.2 Diagrama de sequência Registo e ativação de conta

Através do diagrama da figura 3.45 é possível perceber que quando uma empresa realiza o registo, este será enviado para o servidor, o qual registará a empresa com uma conta não ativada. Esta conta será validada pela Motorline, de seguida é gerado um código de ativação e por fim é enviado para o *email* de registo. Após o registo ou o clique em ativar a conta no *email* recebido, a empresa será encaminhada para a validação de conta e toda a validação ocorre conforme o processo mencionado no capítulo 3.17.1.

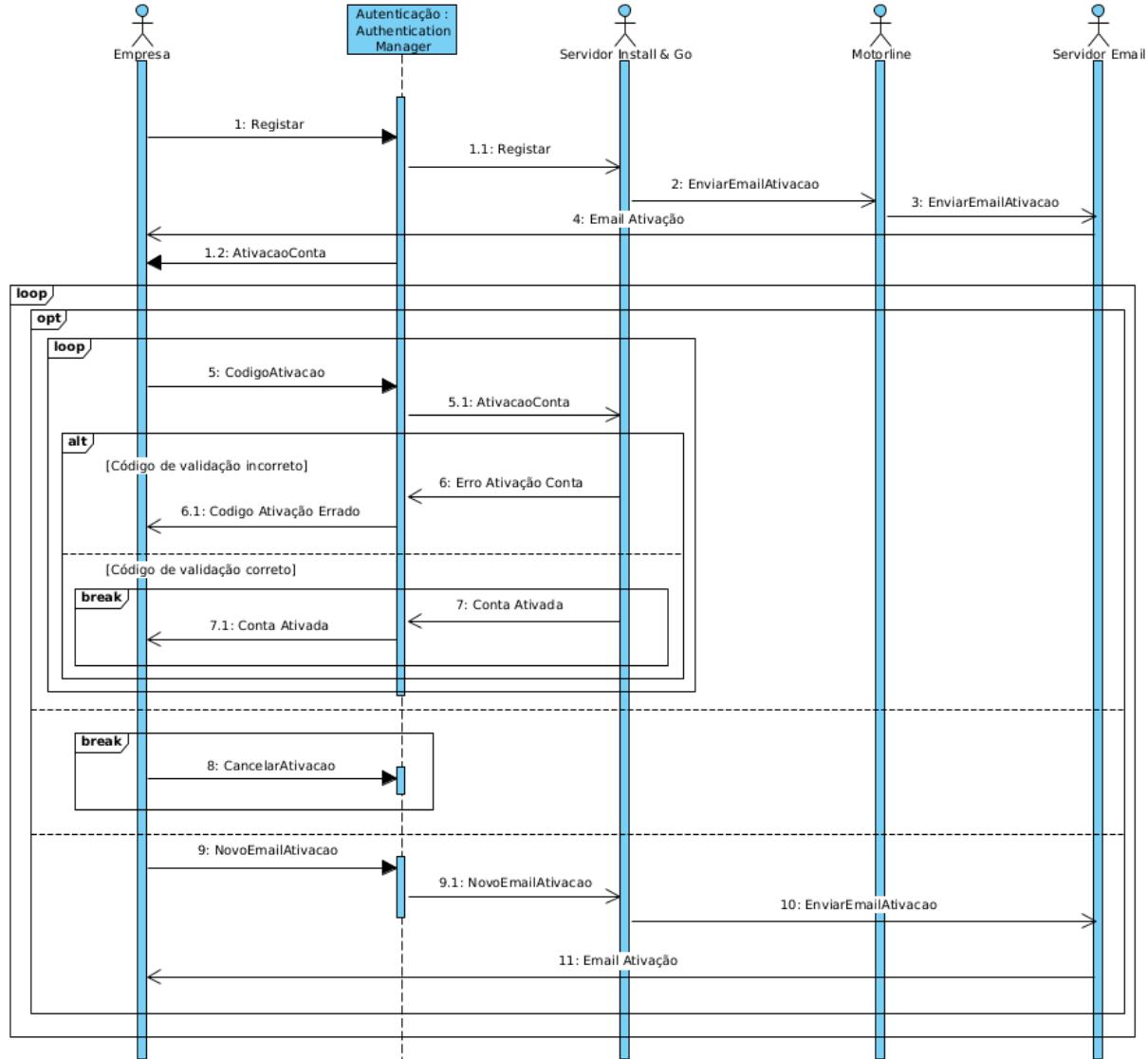


Figura 3.45: Diagrama de sequência de registo e validação de conta

### 3.17.3 Diagrama de sequência registo de técnicos

O diagrama da figura 3.46 tem como propósito dar a perceber que quando uma empresa deseja registar um técnico esta introduzirá os seus dados no sistema, o que leva estes a serem enviados e a conta criada. Posteriormente, um código de ativação é gerado e enviado para o técnico poder ativar a sua conta, este processo segue a mesma orientação mencionada no capítulo 3.17.1.

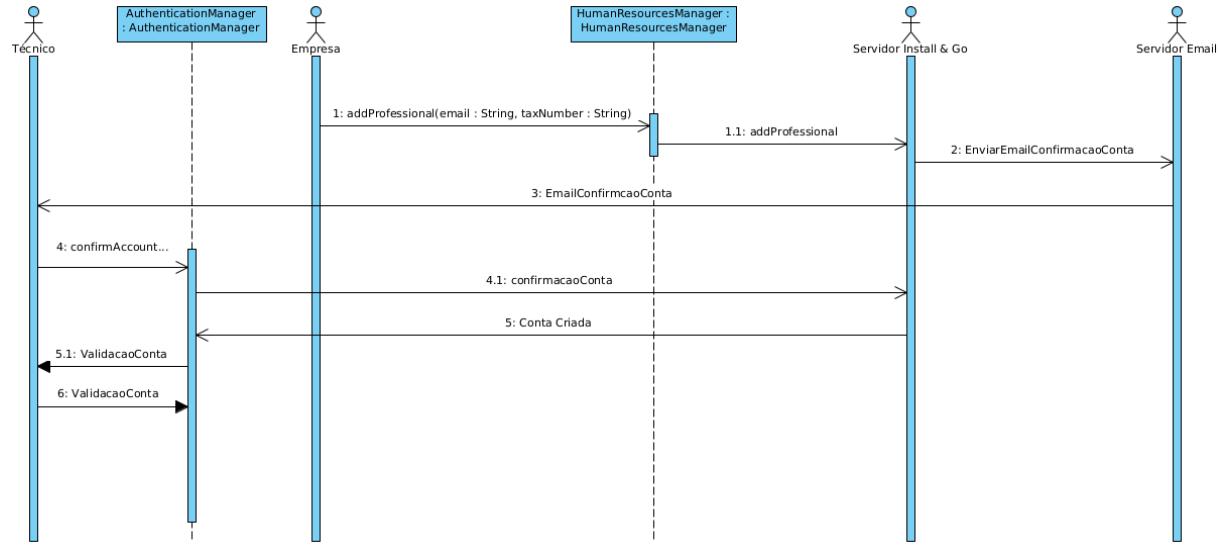


Figura 3.46: Diagrama de sequência de registo de técnicos

## 3.18 Arquitetura de sistema

A figura 3.47 expõe a arquitetura do sistema que indica as principais componentes. Entre estas, é possível visualizar a aplicação *frontend*, que realiza pedidos a uma aplicação *backend* e espera respostas. O *backend* é composto por uma *api rest* que recebe e responde aos pedidos e por uma base de dados a qual recebe *queries* e devolve dados para a *api rest*.

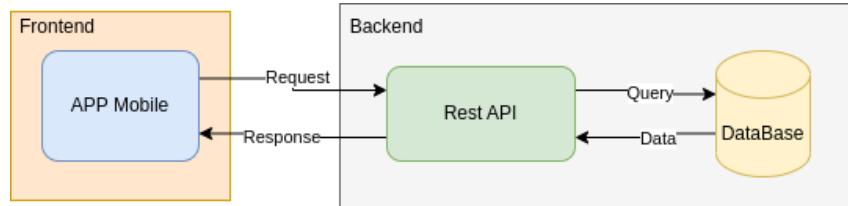


Figura 3.47: Arquitetura do sistema

### 3.18.1 Arquitetura de funcional

A especificação da implementação da *api rest* foi realizada através de uma arquitetura de *backend* (Figura 3.48). Aqui, é possível visualizar que sempre que a *API* recebe um pedido, este é redirecionado primeiramente para o *router*. O *router* tem como função identificar a rota referente ao pedido e deslocar para os respetivos *middlewares*.

Os *middlewares* têm como objetivo realizar todas as ações necessárias antes de proceder à execução do código de rota. Os *middlewares* existentes são o *SessionToken Validator*, valida a sessão do utilizador a realizar o pedido, de forma similar, o *middleware RefreshTokenValidator*, valida a sessão principal do utilizador e por fim, o *middleware RoleValidator*, valida se o utilizador que efetua o pedido tem cargos suficientes para tal. Na eventualidade de não existir nenhum impedimento, o pedido é direcionado para o *controller*.

O *controller* extraí os dados do pedido, verifica se os dados obrigatórios existem e cumprem as regras de negócio e encaminha o pedido para o serviço. O serviço, em caso de necessidade, irá proceder à interação com a base de dados, esta realiza diversas ações como, obter, atualizar, apagar e inserir dados. Por fim, após todo o código de serviço ser executado, a resposta é formulada e devolvida para o utilizador.

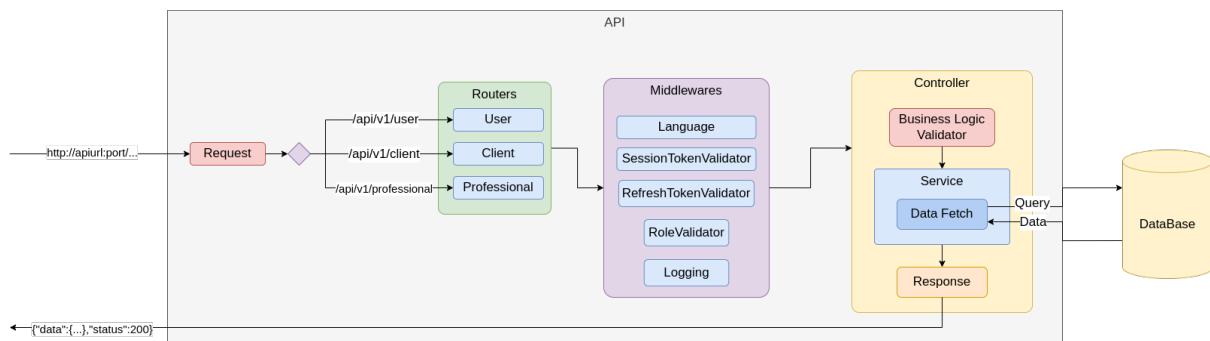


Figura 3.48: Arquitetura do funcional

### 3.18.2 Arquitetura de componentes

A arquitetura de componentes contém todos os serviços que deverão ser implementados na *api frontend*, com a identificação dos atores que poderão realizar estes pedidos. Esta foi desenvolvida após uma análise de todos os dados necessários para suporte do *frontend*.

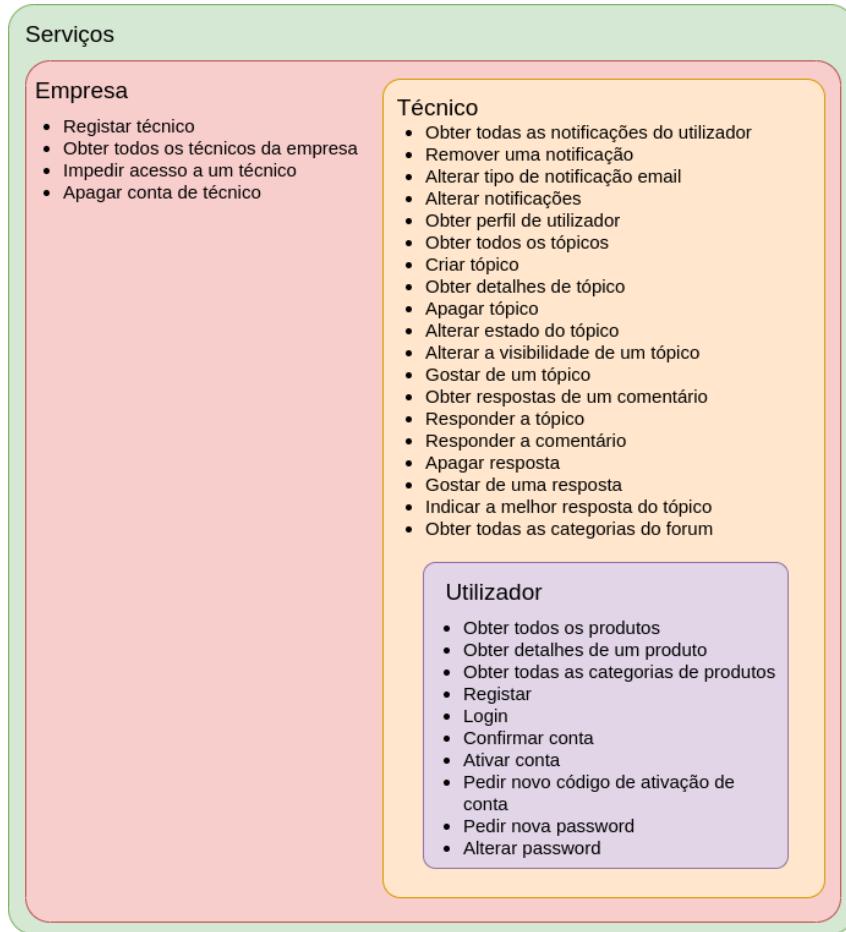


Figura 3.49: Arquitetura de componentes

### 3.18.3 Tabela de endpoints

Com o propósito de evitar colisões de *endpoints* durante a implementação, foi desenvolvida uma tabela de *endpoints*. Esta possui dados semelhantes à arquitetura de componentes (Figura 3.49), mas para cada serviço é indicada a rota e o método a utilizar.

Tabela 3.19: Tabela de endpoints

Serviço	Autor	Rota	Método
Obter informações do utilizador	Cliente	baseurl/client/:uid	GET
Realizar login	Utilizador	baseurl/login	POST
Realizar registo	Utilizador	baseurl/register	POST
Esquecimento de password	Utilizador	baseurl/forgot-password	GET
Ativação de conta	Cliente	baseurl/client/:uid/activate	POST
Reenvio de código de ativação de conta	Cliente	baseurl/client/:uid/new-code	GET
Obter tópicos em destaque	Cliente	baseurl/client/topics/featured	GET
Obter tópicos mais recentes	Cliente	baseurl/client/topics/latest	GET
Obter tópicos por responder	Cliente	baseurl/client/topics/to-answer	GET
Obter tópicos do utilizador	Cliente	baseurl/client/topics	GET
Obter tópicos privados	Técnico	baseurl/professional/topics/private	GET
Gostar de um tópico	Cliente	baseurl/client/topics/:topicId/like	PUT
Adicionar resposta a tópico	Cliente	baseurl/client/topics/:topicId/answer	POST
Adicionar resposta a outra resposta	Cliente	baseurl/client/answers/:answerId/	POST
Gostar de uma resposta	Cliente	baseurl/client/answers/:answerId/like	PUT
Marcar tópico como completo	Cliente	baseurl/client/topics/:topicId/completed	PUT
Remover Tópico	Cliente	baseurl/client/topics/:topicId/	DELETE
Alterar visibilidade do tópico	Cliente	baseurl/client/topics/:topicId/visibility	PUT

Continua na página seguinte

Tabela 3.19: Tabela de endpoints (Continuação)

Adicionar melhor resposta do tópico	Cliente	baseurl/client/topics/:topicId/answers/:answerId/best-answer	PUT
Adicionar novo tópico	Cliente	baseurl/client/topics/	POST

The screenshot shows a Swagger UI interface with two main sections: **Categories** and **Auth**.

**Categories:**

- GET /categories** Get categories
- POST /categories** Add categories
- POST /subcategories** Add subcategories

**Auth:**

- POST /register** Register a company
- POST /login** Login
- GET /new-session** New session token
- PUT /confirm** Account confirmation
- PUT /activate** Account activation
- PUT /new-password** New password
- PUT /set-password** Set up new password
- PUT /profesisonal/logout** Logout
- PUT /new-code** Get new activation code

Figura 3.50: Listagem de serviços documentados

# 4. Trabalho desenvolvido

## 4.1 Web scraper

Após uma reunião com o cliente foi compreendido que o catálogo de produtos Motorline não se encontra num servidor. Estes dados apenas estão diretamente no *website* da empresa, sendo assim, foi determinado a necessidade de criar um *web scraper*.

Durante a reunião concluiu-se que o *web scraper* iria apenas correr uma vez por mês para ser evitada a sobrelocação do servidor. Para agilizar a realização do *web scraper* foi entregue pela empresa a estrutura do *website* a seguir para serem obtidas as informações.

### 4.1.1 Implementação web scraper

A implementação e testagem do *web scraper*, sem sobrelocação do servidor, foi alcançada, uma vez que, préviamente foi descarregado todo o *website* localmente, o que permitiu simular o catálogo.

Para a implementação do *web scraper* optou-se por uma abordagem mais simples. Esta abordagem, consiste na realização de um pedido para ser obtida a página *web*, ler e guardar os dados.

A linguagem utilizada foi *python* devido à facilidade de lidar com abundantes quantidades de dados. O tratamento dos dados das páginas *web* foi realizado com o auxílio da biblioteca *bs4*, também conhecida como *beautiful soup*, esta permite alimentar com uma página *web* e de seguida realizar pesquisas sobre esta com base em *tags* e atributos dos elementos.

Após um estudo do catálogo foi compreendido que dados seriam necessários, sendo estes:

1. Categorias e subcategorias de produtos;
2. Produtos de cada categoria e subcategoria;
3. Documentação dos produtos;
4. Imagens e vídeos dos produtos;

Para guardar estes dados foi utilizado um dicionário, que contém como chaves as categorias de produtos. Para cada categoria contém mais um dicionário com as subcategorias de produtos. Cada categoria possui uma lista de produtos, sendo cada produto representado por um dicionário, que abrange como chaves os seus atributos. A utilização de dicionários e listas para guardar estes dados deve-se a que o objetivo será guardar estes em *JSON* e a transformação é simplificada com a utilização destas estruturas devido à proximidade com a estrutura *JSON*.

```
{
  "produtos": [
    {
      "nome": "nome do produto",
      "categoria": "categoria do produto",
      "subcategoria": "subcategoria do produto",
      "imagem-amostra": "imagem de amostra do produto",
      "imagens": [
        "links de imagens de produtos"
      ],
      "descricao": "descrição do produto",
      "documentacao": [
        {
          "nome": "Nome da documentação",
          "url": "link da documentação"
        }
      ],
      "placas-controlo": [
        {
          "nome": "nome da placa de controlo",
          "url": "documentação placa de controlo"
        }
      ],
      "informacao-geral": "imagem de informação geral",
      "desenho-tecnico": "imagem de desenho técnico",
      "videos": [
        {
          "name": "nome do vídeo",
          "url": "link do vídeo"
        }
      ]
    },
    "categorias": {
      "nome_categoria": {
        "nome_subcategoria": [
          {
            "nome": "nome produto",
            "link": "link página do produto",
            "imagem-amostra": "link imagem de amostra do produto"
          }
        ]
      }
    }
  ]
}
```

Figura 4.1: Estrutura dos dados obtidos

Após uma análise da estrutura do *website* foi constatado que a página geral dos produtos possui todas as categorias, assim como, as subcategorias com *urls* para as páginas que contém todos os produtos das subcategorias.

Sendo assim, em primeiro lugar percebeu-se que cada conjunto(categoria, subcategorias) é uma secção, pelo que, são obtidas todas as secções das categorias. Para cada uma destas secções, é obtido o título que equivale ao nome da categoria e também todos os elementos clicáveis. Estes, correspondem às subcategorias que contêm o nome e um *url*, que redireciona para a página de produtos da subcategoria.



Figura 4.2: Página geral de produtos

Posto isto, já é possível identificar cada categoria e subcategoria, assim como, o *url* da página de produtos para cada subcategoria. Mas, após alguma análise dos dados foi percebido que estas não contêm acentuação visto que, existe uma formatação no *website*. Para resolver este problema, foram pesquisadas ferramentas capazes de corrigir erros ortográficos. Pelo que, descobriu-se que a biblioteca mais utilizada em *python* para resolver este problema é a biblioteca *spellchecker*. Esta ferramenta, por sua vez, é a mais utilizada dado à sua capacidade de correção dos erros ortográficos em diversas linguagens. Por fim, sempre que uma categoria e subcategoria é obtida, antes de ser guardada, é corrigida.

Aquando a finalização do processo anterior, cada *url* é aberto e a partir disto, são obtidos os *urls* de produtos e imagens de amostra dos produtos. Em cada página, foram obtidos todos os elementos pressionáveis existentes, sendo que cada um refere-se a um produto. Para obter o nome do produto correspondente, foi utilizado o nome contido no *url* do elemento, pois todos os produtos seguem a mesma estrutura, sendo esta, /produtos/nome-produto. Como em *urls* não é permitido utilizar acentuação e espaços, todos os nomes foram corrigidos com a mesma ferramenta de correção ortográfica mencionada anteriormente.

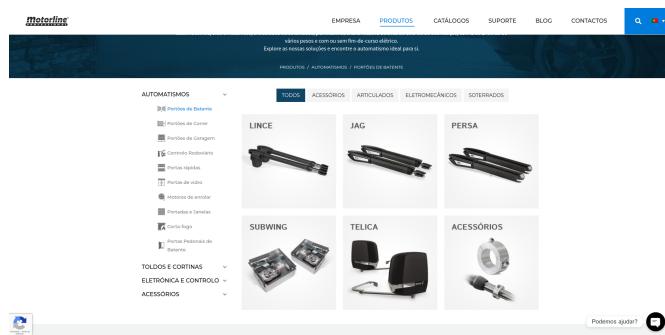


Figura 4.3: Página de produtos de uma subcategoria

Neste momento, depois de correr o código foi deduzido que existem algumas páginas de produtos em que este não conseguia obter dados, pelo que, um erro era atirado. Para compreender exatamente em que páginas de produtos surgia este erro, sempre que um erro era detetado, este *url* era adicionado a uma nova chave do dicionário mencionado anteriormente. Esta chave, tem o nome *misses* e contém todos os *urls* em que algum erro aconteceu. Então, nesta ocasião percebeu-se que nem todas as páginas de produtos são iguais, contudo, após uma reunião com o cliente este expôs que existem páginas de produtos e de detalhes de produtos que são muito diferentes das restantes.

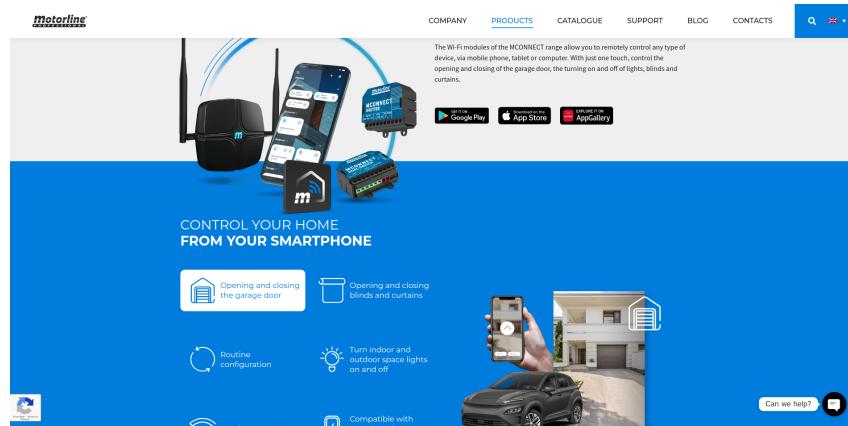


Figura 4.4: Página de produtos de uma subcategoria distinta

Com o objetivo de não atrasar o restante projeto foi determinado primeiramente, que todos os produtos que contêm páginas semelhantes deveriam ser obtidos. Para cada página de produto, foi obtido o título que corresponde ao nome do produto e o elemento que contém o *id* produto-descrição, que corresponde à sua descrição. As imagens dos produtos são disponibilizadas através de *urls* na secção da galeria do produto, pelo que, são obtidos todos os seus *urls*.

A documentação dos produtos pode ser disponibilizada através de *urls* para os manuais, ou, com uma lista *dropdown* com todos os manuais em formato *url*, o que permite, serem obtidos através de todos os *urls* da secção de documentação, assim como, os nomes e todas as opções do *dropdown* se este existir.

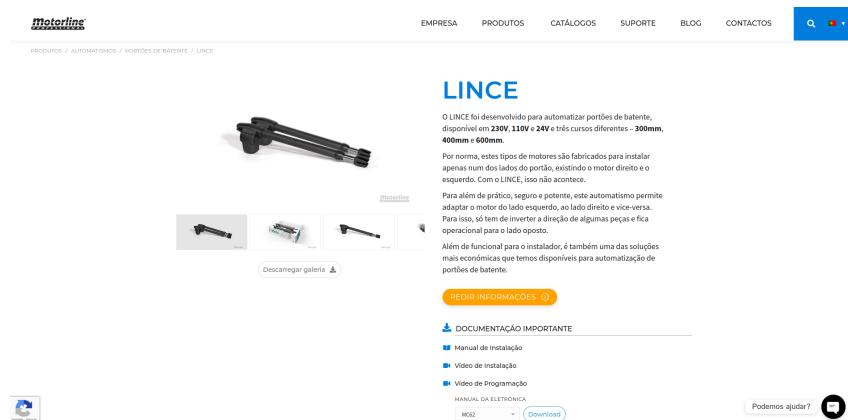


Figura 4.5: Página de detalhes de produto, secção inicial

As imagens de desenho técnico e informação geral encontram-se disponíveis na secção correspondente ao nome de cada uma, caso existam, são obtidas as imagens e os seus *urls*.

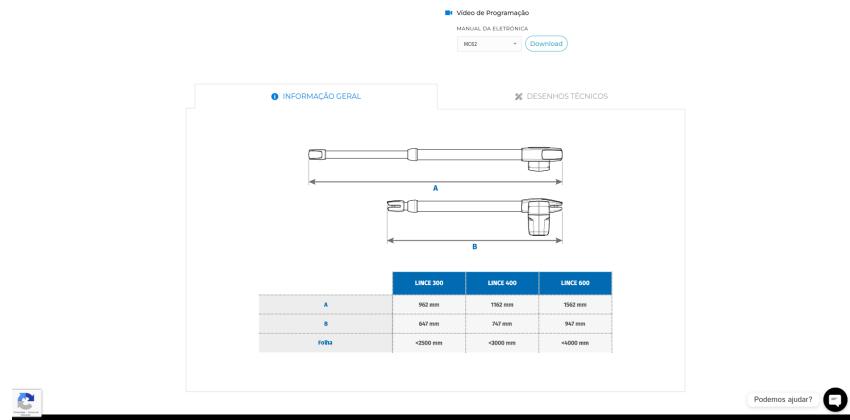


Figura 4.6: Página de detalhes de produto, secção de informações

Os vídeos de produtos estão disponíveis na secção de vídeos, sendo que, cada uma contém o nome e o vídeo. Estes são demonstrados através da utilização de um elemento *iframe*, este contém um *url* para o vídeo, mas após a tentativa de visualização deste *url*, constatou-se que não é possível obter o vídeo a partir deste. Sendo assim, foi investigada a plataforma *vimeo*. Esta, contém todos os vídeos de produtos, pelo que, para cada um é gerado um *id* único. Este vídeo poderá ser acedido através do *url* geral da plataforma seguido do *id*. O *id*, está colocado no elemento *iframe*, pelo que, é obtido e acrescentado ao *url* da plataforma, o que permite guardar todos os vídeos de produtos.

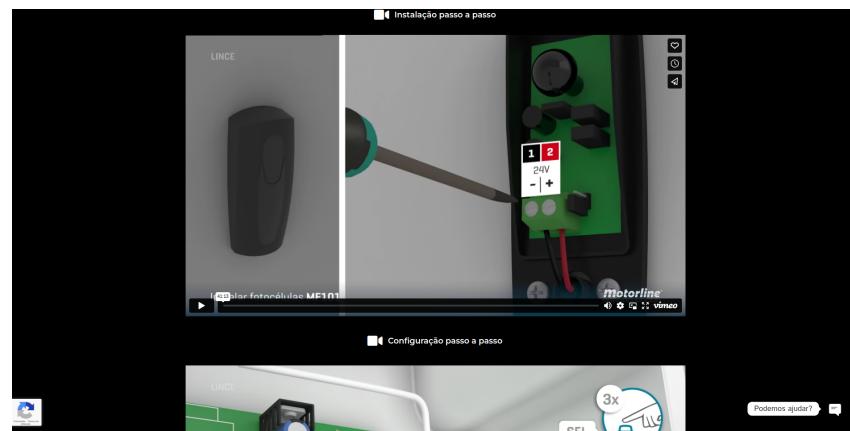


Figura 4.7: Página de detalhes de produto, secção de videos

#### 4.1.1.1 Implementação no website

Logo que se verificou que pelo menos 80% dos produtos totais eram obtidos, decidiu-se testar no *website*. Para isto, foi utilizada a biblioteca *requests*, com a qual é realizado um pedido *GET* a cada *url* necessário para obter a página *web*. Assim que o código foi corrido e a resposta analisada compreendeu-se que o *website* bloqueia este tipo de solução como é indicado na figura 4.8

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
<p>Additionally, a 403 Forbidden
error was encountered while trying to use an ErrorDocument to handle the request.</p>
</body></html>
```

Figura 4.8: Resposta obtida aquando o pedido ao *url* da página *web*

Através da resposta obtida, foi compreendida a necessidade de alteração da abordagem, visto que, a anterior não poderia ser utilizada. Opcionalmente seria possível seguir a abordagem de simular a ação humana através da abertura de um navegador programaticamente e pesquisar pelo *url* desejado.

Após uma investigação descobriu-se que existem ferramentas que permitem controlar o dispositivo onde correm, mas estas impedem a utilização enquanto se encontram a correr. Também, foram descobertas ferramentas que apenas recebem o navegador a utilizar e abrem uma nova janela deste para realizar a pesquisa. O processo de obter os produtos é demorado, pelo que, optou-se pela segunda opção, uma vez que, seria possível continuar com o trabalho em paralelo com o processo de obter dados. Sendo assim, a ferramenta mais recomendada para realizar esta operação é a biblioteca *Selenium*.

Com a utilização da biblioteca *Selenium* indicou-se qual o navegador a utilizar. Neste caso foi escolhido o *chrome*, dado que, encontrava-se instalado no dispositivo. Após ser indicado qual o navegador a utilizar, é necessário para cada página referir qual o elemento a esperar que carregue, pois, por vezes existem elementos que demoram mais tempo a carregar do que a página. A página carregada poderá não conter o elemento a obter, pelo que foi implementado um tempo de espera máximo de 5 segundos, assim que este tempo expira, a operação é cancelada e o *url* é adicionado à lista de *urls* com erros.

#### 4.1.1.2 Melhoria de implementação

Depois de completo o processo, foi decidido melhorar a implementação com a resolução dos erros encontrados em *urls* específicos. Para perceber exatamente quais os *urls* que possuem erros, foram direcionados os dados obtidos para um ficheiro *JSON*. Sendo assim, os *urls* com erros eram os indicados na figura 4.9.

```
"misses": [
    "https://motorline.pt/produto/acessorios-para-portoes-de-correr/",
    "https://motorline.pt/produto/acessorios-portoes-garagem/",
    "https://motorline.pt/produto/zuma/",
    "https://motorline.pt/produto/sigma-x/",
    "https://motorline.pt/produto/acessorios-barreira-eletromecanica/",
    "https://motorline.pt/produto/acessorios-para-portas-de-vidro/",
    "https://motorline.pt/produto/adaptador-tub/",
    "https://motorline.pt/produto/acessorios-motores-enrolar/",
    "https://motorline.pt/produto/cortina-corta-fogo-flama/",
    "https://motorline.pt/produto/acessorios-para-toldos/",
    "https://motorline.pt/produto/mbn25/",
    "https://motorline.pt/produto/acessorios-controlo-de-acessos/",
    "https://motorline.pt/produto/stop/",
    "https://motorline.pt/produto/acessorios-fotocelulas/"
]
```

Figura 4.9:Urls com erro primeira interação

Posteriormente a uma primeira análise percebeu-se que grande maioria os erros provêm de *urls* de acessórios de produtos, isto deve-se ao facto destes encontrarem-se na página de produtos de subcategoria e serem tratados como um produto, sendo assim, sempre que se trata de um url de acessórios seria necessário correr código para obter dados de acessórios ao invés de detalhes de produtos. Deste modo, compreendeu-se que nos *urls* a palavra accessórios está sempre contida, o que permite que sempre que esta é detetada num *url*, seja corrido o código referente a obter de acessórios. Para desenvolver este código foi primeiramente analisada a página de acessórios de produtos(Figura 4.10), esta contém para cada acessório um elemento do tipo artigo o qual detém uma imagem, titulo e descrição. Esta descrição por vezes possui *urls* para os produtos aos quais este acessório se refere. Sempre que estes *urls* são detetados, os nomes dos produtos são guardados para futuramente realizar a ligação entre os acessórios e os produtos, dado que, não existem produtos com nomes iguais.



Figura 4.10: Exemplo de página de acessórios

Na iteração seguinte determinou-se que a quantidade de *urls* com erros diminuiu, mas mesmo assim existiam produtos com erro, pelo que, estes foram analisados e percebeu-se que o erro ocorria, visto que, por vezes as páginas não continham os vídeos ou imagens de documentação. Para resolver este problema, o código foi alterado para somente obter estes dados se os elementos existirem na página. Após correr novamente o código foi compreendido que a quantidade de falhas obtidas diminuiu drasticamente (Figura 4.11), mas mesmo assim, ainda existiam quatro falhas a ocorrer e após uma análise foi determinado que estas ocorriam devido a uma página de subcategoria de produtos conter um serviço, um produto conter uma página de detalhes de produto com sub produtos, existir uma página de adaptadores de produtos e um produto conter uma página de detalhes diferente das demais.

```
"misses": [
    "https://motorline.pt/produtos/electronica-e-controlo/casa-inteligente/",
    "https://motorline.pt/produto/zuma/",
    "https://motorline.pt/produto/adaptador-tub/",
    "https://motorline.pt/produto/cortina-corta-fogo-flama/"
]
```

Figura 4.11: Indicação das páginas com falha

A página de adaptadores de produtos segue uma estrutura similar à dos acessórios, pelo que foi a primeira a ser abordada e resolvida através do código de obter detalhes de acessórios. Neste sentido este foi alterado para executar sempre que a palavra acessórios ou adaptadores encontra-se no *url*.

A resolução do problema de existirem serviços e subprodutos implica uma alteração no diagrama de entidade relação, pelo que, o problema do produto que contém uma página de detalhes diferente das demais foi resolvido primeiro. Este produto para além da dificuldade de ser uma página completamente diferente, as informações encontram-se espalhadas (Figura 4.12), o que leva a que estas tenham de ser combinadas para construir os detalhes do produto. Após obter-se estes dados foi determinado que as imagens do produto têm dados escritos que não se encontram na imagem original. Para a solução deste problema o cliente recomendou obter com *screenshots* e guardar num servidor de imagens.



Trata-se de uma cortina construída de forma compacta em perfis lacados, garantindo assim um maior aproveitamento do vão de passagem, sem alterar a estética do espaço. Rigorosamente testada, esta passou com distinção nos seguintes testes CE.

CERTIFICAÇÃO E QUALIDADE

Podemos ajudar?

Figura 4.12: Exemplo de página de produto incomum

O problema de existirem produtos com subprodutos foi resolvido através da alteração da base de dados, à qual foi adicionada uma nova ligação sobre si mesma na tabela produtos, o que permite que um subproduto possua uma ligação com produto principal. Após ser feita esta alteração, o código para obter os dados do catálogo foi alterado. Em primeiro lugar, foram obtidos todos os dados do produto principal e de seguida os dados específicos a cada subproduto, o que leva a que a organização do produto principal seja igual aos restantes, mas, com um dado extra com os subprodutos.

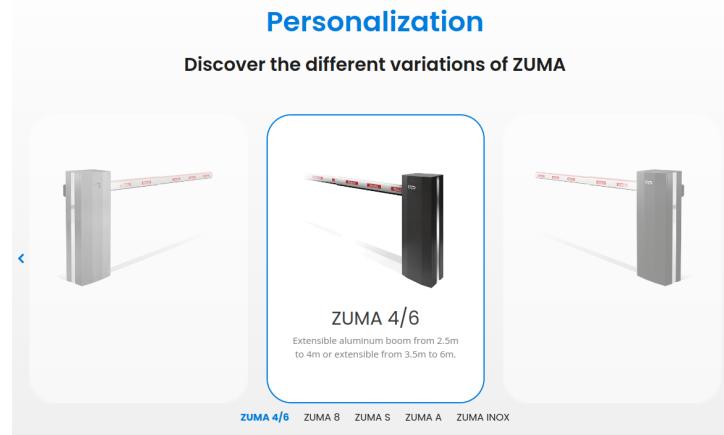


Figura 4.13: Exemplo de página de produto com subprodutos

O último problema a solucionar é a existência de serviços, pelo que, iniciou-se pela análise deste tipo de produto. Este possui descrição e imagens como os produtos, mas também vídeos direcionados a plataformas diferentes, produtos do serviço, registo de atualizações do serviço, planos de pagamento e cada plano contém diversas ofertas. Através da estrutura de um serviço foi adicionado à base de dados as tabelas de serviço, planos do serviço, ofertas dos planos, vídeos do serviço, informações de atualizações dos serviços, imagens do serviço e a ligação à tabela produtos, o que permite a identificação dos produtos do serviço. Aqui, foi identificada a necessidade de manter guardado os *links* para todas as imagens e vídeos na própria base de dados, visto que, existem imagens que não sabem ainda qual é o *id* do produto referente, pelo que não seria possível no futuro obtê-las sem alteração manual, o que levou a que estas tabelas também existam para os produtos. De seguida compreendeu-se que existem dados que não são necessários na descrição, o cliente recomendou guardar estes dados como *screenshots*.

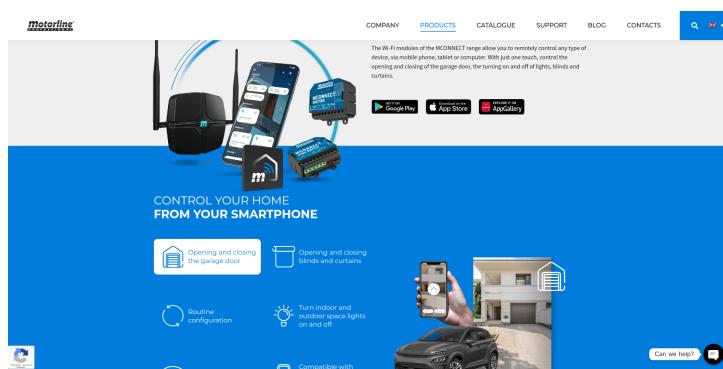


Figura 4.14: Exemplo de página de serviço

#### 4.1.1.3 Diagrama de base de dados final

Após as alterações mencionadas à base de dados, o seu diagrama final encontra-se na Figura 4.15

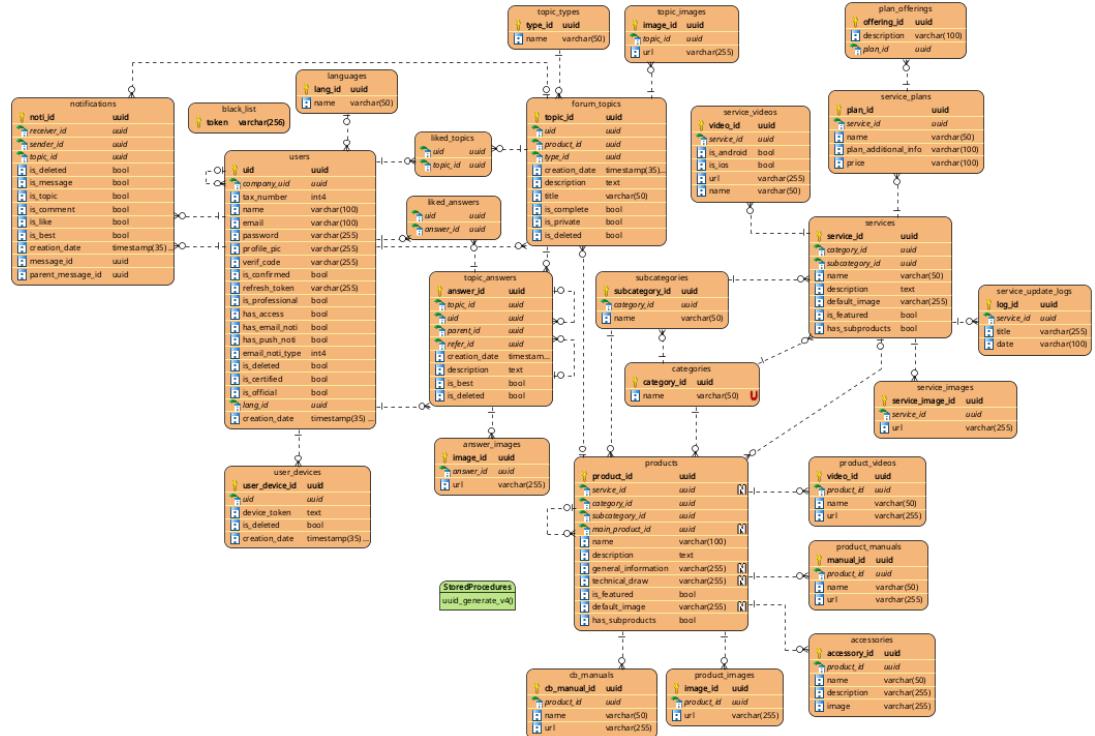


Figura 4.15: atualização da base de dados

#### 4.1.1.4 Armazenamento de dados

Após obter-se os dados dos produtos, é necessário guardar na base de dados para serem disponibilizados ao *backend*. Para realizar esta operação existiam duas opções, criar um serviço para inserir produtos e realizar um pedido a este serviço, ou então, conectar diretamente o *web scraper* à base de dados. Não seria de grande interesse conectar diretamente à base de dados, pelo que, foi decidido criar um serviço que recebe um produto e o insere. O grande problema que surgiu com esta solução é que os pedidos ocorrem de forma sequencial, mas com pouco tempo de espera, o que levou a que o limite máximo de conexões com a base de dados fosse extrapolada. Isto acontece porque para cada serviço que recebe uma chamada é desenvolvida uma nova conexão à base de dados, todas as operações são realizadas e por fim a conexão é terminada, mas enquanto estas operações estão a decorrer, o servidor poderá receber mais pedidos, o que leva a que mais conexões sejam criadas, o que atinge assim rapidamente o limite de conexões da base de dados. Como solução para este problema foi acrescentado uma espera de 0.5 segundos a cada pedido. A inserção de serviços decorreu com o mesmo processo.

A inserção de categorias decorre através do envio das categorias a inserir num *array*, o que leva a que estas sejam inseridas todas num pedido. As subcategorias, visto que, não se sabe o *id* da categoria referente, foi utilizado o nome da categoria, pois este é único, sendo assim, é enviado o nome da categoria e as subcategorias referentes. Todas são inseridas com a referência para a sua categoria.

## 4.2 Backend

Após o desenvolvimento do *webscraper*, procedeu-se com o desenvolvimento do suporte *backend* do projeto.

### 4.2.1 Organização do projeto

Antes de iniciar a implementação definiu-se a estrutura do projeto a seguir. *MVC* foi a estrutura escolhida, uma vez que, é a mais comum e bem estabelecida. Sendo assim, a organização do projeto seguiu a seguinte estrutura:

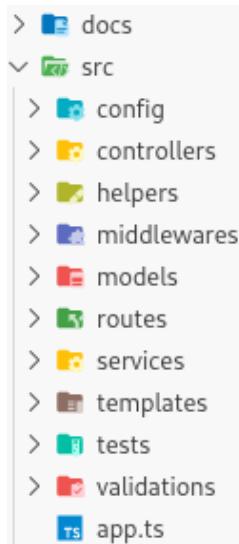


Figura 4.16: Exemplo de página de produto incomum

- **docs** - Documentação gerada;
- **src** - Base de todo o projeto;
- **config** - Ficheiros de configuração do projeto;
- **controllers** - Controladores para cada pedido;
- **helpers** - Ficheiros com funções gerais utilizadas regularmente;
- **middlewares** - Ficheiros com os middlewares da api;
- **models** - Classes criadas para representação de base de dados e outras entidades;
- **routes** - Rotas existentes;
- **services** - Serviços para cada pedido;
- **templates** - Templates de *email* a serem enviados;
- **tests** - Testes de código realizados;
- **validations** - Validações a realizar do modelo de negócio e dos dados;
- **app** - Ficheiro de início do projeto;

### 4.2.2 Definição de rotas base

Após ser definida a estrutura do projeto foram estruturadas as rotas base a existir, estas referem-se a cada ator. Para uma melhor organização das rotas e da aplicação de regras, foram definidos 3 *routers*, *user*, para utilizadores sem sessão, *professional*, para técnicos, e *company* para empresas. Para indicar qual o *router* a utilizar em cada pedido foi definido que:

- **http://baseurl:port/professional** - Encaminhar para *router* de técnicos;
- **http://baseurl:port/company** - Encaminhar para *router* de empresas;
- **Restantes** - Encaminhar para *router* de utilizadores;

### 4.2.3 Middlewares

Um *middleware* comporta-se como uma ligação entre duas partes e permite executar código.

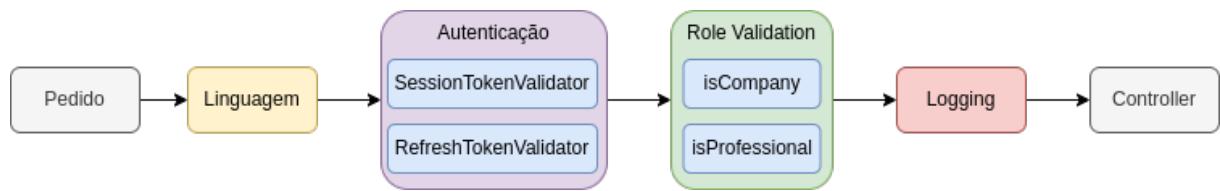


Figura 4.17: Comportamento de middlewares

#### 4.2.3.1 Idioma de Comunicação

O bem essencial para uma boa comunicação entre duas partes, é a utilização da mesma linguagem, pelo que, foi necessário perceber qual a linguagem a utilizar quando se responde a um pedido. Para este fim, foi desenvolvido um *middleware*, cujo objetivo é verificar se existe a chave *language* no cabeçalho do pedido. Caso esta exista, é obtida a linguagem e guardada nas variáveis locais do pedido. Na eventualidade desta não existir, a aplicação dará uma resposta em português por omissão. Este valor poderá ser futuramente alterado de forma simples.

#### 4.2.3.2 Autenticação

A autenticação dos utilizadores foi implementada através de *Json Web Token*. Este tipo de autenticação, tem por base a utilização de *tokens* com tempo de expiração, o que significa que enquanto o *token* estiver válido, o utilizador poderá realizar pedidos. Assim que o *token* expirar, este terá de se autenticar novamente para obter um novo *token*. A utilização de *tokens* permite assegurar que os pedidos são realizados com *tokens*, gerados pela *API*, através de uma chave de assinatura de *token*, o que impede a utilização de *tokens* gerados por utilizadores.

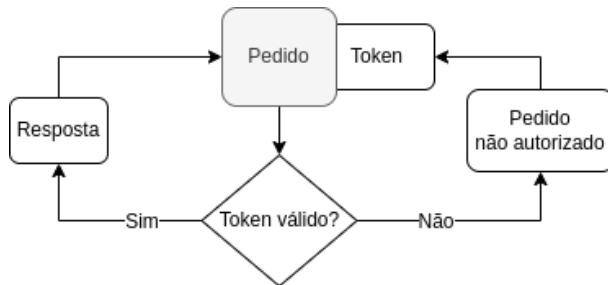


Figura 4.18: Utilização de tokens

A maior valência da utilização da técnica de autenticação mencionada anteriormente é a segurança. Isto acontece porque estes *tokens* têm geralmente uma duração muito curta, como por exemplo, quinze minutos. Sempre que um *token* de sessão expira o utilizador necessita realizar novamente o *login*, o que poderá tornar a utilização da aplicação imprática.

A solução deste problema sem a perda de segurança significativa veio pelo meio da utilização de *tokens* de duração maior em conjunto com os *tokens* de duração curta. Enquanto o *token* de grande duração se encontrar válido, novos *tokens* de curta duração são gerados para o utilizador, o que leva a que o utilizador nunca perca a sua sessão. Estes *tokens* de grande duração têm por nome *refresh tokens* e os *tokens* de curta duração têm por nome *session tokens*. O utilizador sempre que termina a sessão o *refresh tokens* deverá ser apagado.

Sempre que um utilizador realiza um pedido, o seu *token* de sessão deverá ser validado. Caso este seja válido, o seu *refresh token* deverá ser validado e apenas após esta verificação o utilizador estará autenticado. Na eventualidade de o *token* de sessão ou de *refresh* estiverem expirados, este estará sem autorização para realizar o pedido, mas, poderá solicitar um novo *token* de sessão enquanto o seu *refresh token* estiver válido. Isto acontece sem realizar novamente o *login* e sem o utilizador perceber.

Além das funcionalidades atrás mencionadas é possível também associar dados em formato *JSON* a um *token*. Esta funcionalidade foi utilizada para enviar o *id* e o cargo do utilizador a qual pertence este *token*.

#### 4.2.3.3 Validação de *Role*

Com a finalidade de garantir que apenas utilizadores com cargos suficientes conseguem realizar as ações regradas, foi desenvolvido um *middleware* que valida se o utilizador que realizou o pedido tem permissões para o mesmo. Este *middleware*, interliga-se com o *middleware* anterior, pois, o cargo do utilizador em questão é enviado no *token*, pelo que, é obtido e realizada uma comparação com o cargo desejado. Para isto foram criados dois *middlewares*s diferentes, um valida o cargo das empresas e o outro o cargo dos técnicos. As empresas podem realizar operações de técnicos, dado que, no *middleware* de técnicos é verificado se o *token* corresponde a um utilizador empresa, ou a um utilizador técnico. Já no *middleware* de validação de empresa é verificado se o utilizador tem cargo de empresa.

#### 4.2.3.4 Logging de erros

A identificação dos erros no servidor é difícil, uma vez que, os erros são tratados e os dados referentes não são guardados ou utilizados. Para resolver este problema foi determinado que sempre que um erro que não é customizado é detetado é registado um *log*, este contém informações sobre o pedido como data e hora, dados recebidos, a descrição original do erro e serviço referente.

Para implementar esta solução foi em primeiro lugar criado um *middleware* que sempre que deteta erros dentro do serviço executa um método. Este método, por sua vez, obtém os dados referentes ao pedido realizado, sendo estes a data e hora, os dados recebidos e o serviço pedido. Após se obter os dados do pedido é obtida a mensagem do erro e estes dados acrescentados numa nova linha no ficheiro de erros do servidor.

#### 4.2.3.5 Logging de pedidos com Morgan

Para realizar o *logging* de pedidos a serviços foi utilizado o *Morgan*. Esta ferramenta precisa da indicação do ficheiro onde escrever os *logs*, o que leva à utilização de um ficheiro chamado *log*. O *Morgan*, também precisa da indicação do tipo de *log* a realizar. Estes tipos são indicados pela ferramenta, o utilizado foi o *combined*, este é o tipo de *log* mais completo, visto que, é o que obtém mais dados, sendo estes, a data e hora do pedido, o tipo, o serviço, os dados recebidos, a resposta devolvida e também a descrição do sistema utilizado.

### 4.2.4 Controllers

Assim que um pedido consegue ultrapassar todos os *middlewares* sem ser impedido, este é redirecionado para um *controller*.

#### 4.2.4.1 Estruturação dos controllers

Para evitar a variação de código destes *controllers* em termos de estrutura, foi decidido desenhar uma estrutura de *controller* e aplicar perante o demais código. Esta segue as seguintes etapas:

1. Obter dados do pedido
2. Validar se os dados obrigatórios são obtidos
3. Validar o pedido perante o modelo de negócio
4. Executar a lógica do pedido
5. Formular a resposta e enviar
6. Em caso de erro este deverá ser capturado e processado para enviar um erro para o utilizador

Esta estrutura será sempre aplicada, pois, foram utilizados *snippets* de código que permitem criar um modelo de estrutura, apenas é necessário escrever a palavra-chave e toda a estrutura é aplicada, pelo que, é necessário de seguida efetuar as alterações perante o contexto.

#### 4.2.4.2 Execução da lógica de negócio

A execução da lógica de negócio passa por direcionar os dados para a ação correta. Esta ação geralmente resulta numa operação de base de dados. Inicialmente foi desenvolvida toda a validação de código e todas as operações de base de dados diretamente na execução da lógica de negócio. Após uma revisão desta organização de código com o professor orientador, foi decidido separar estas funcionalidades. Daí surgiu a componente de validação de dados, a de operações de base de dados e a de lógica de negócio, que implementa a componente de operações de base de dados. Sendo assim, para evitar que estas operações sobre a base de dados estejam em conjunto com o direcionamento dos dados, foram criados modelos para cada tabela. Cada modelo contém um conjunto de operações sobre a tabela correspondente. Estas operações, estão contidas sobre métodos que podem receber dados para executar na operação e devolver a resposta da mesma.

#### 4.2.4.3 Validação dos dados

A validação dos dados é necessária para evitar erros a nível de servidor com dados em falta e também para aplicar as regras de negócio. Para realizar estas validações, é necessário em primeiro lugar verificar se todos os dados são recebidos, de seguida, são enviados para um *validator*. O *validator* executa todas as verificações necessárias a nível de regras de negócio, na possibilidade de alguma regra não ser cumprida, é atirado um erro.

#### 4.2.4.4 Formulação da resposta

Como mencionado no capítulo 4.2.3.1, o bem mais importante numa boa comunicação é a utilização da mesma linguagem, pelo que, a resposta do servidor deverá utilizar a linguagem indicada pelo utilizador. Para realizar esta tradução foi utilizado o mesmo conceito que se usa em *Android*. Neste é desenvolvido um ficheiro que contém um conjunto de chaves e a cada corresponde um texto. Cada tradução tem de conter estas chaves para que seja possível obter o texto correto para cada uma. Sendo assim, foi utilizado um ficheiro *JSON* que dispõem das chaves das linguagens suportadas. A cada uma corresponde a um conjunto de outras chaves que com todas as traduções necessárias. Esta abordagem permite que de forma fácil, futuramente seja possível adicionar outras linguagens ao servidor.

```
"pt/PT": {
    "unauthorized_request": "Pedido não autorizado",
    "jwt_key_missing": "Chave jwt em falta",
    "undefined_token": "Token não definido",
    "invalid_language": "Linguagem inválida",
    "invalid_request": "Pedido inválido",
```

(a) Página de login

```
"en/UK": {
    "unauthorized_request": "Unauthorized Request",
    "jwt_key_missing": "Jwt key missing",
    "undefined_token": "Token undefined",
    "invalid_language": "Invalid communication language",
    "invalid_request": "Invalid Request",
```

(b) Página de registo

Figura 4.19: Autenticação - Login e Registo

O suporte a este ficheiro foi elaborado com uma operação que recebe a chave e a linguagem desejada. Este devolve o texto traduzido, dado que, na formulação da resposta a operação é executada com a indicação da chave do texto a enviar e a linguagem desejada, sendo que este é devolvido para o utilizador.

#### 4.2.4.5 Processamento de erros

Como não é de interesse enviar erros do próprio servidor para o utilizador, foi decidido controla-los. Para isso foi concebido um erro customizado com base no erro da própria linguagem. Este recebe por parâmetro o código da tradução da mensagem. Esta abordagem permite evitar que sempre que um erro é lançado o sistema pare. Contudo, sempre que um erro é lançado pela base de dados, erro de código ou de biblioteca, o original é chamado, o que levou a que sempre que é detetado é devolvida uma mensagem de "erro de servidor", isto evita a que dados sensíveis e desnecessários para o utilizador sejam devolvidos.

#### 4.2.5 Envio de *emails*

Como mencionado no capítulo 2.2.2.7, não é permitido a utilização de acentuação na escrita de *emails* no *Tabular*, pelo que, primeiramente todos os problemas foram resolvidos. Para permitir que os dados dos *emails* sejam personalizáveis, como por exemplo, dados de utilizador e *link* para clicar, estes *emails* são colocados em métodos que recebem por parâmetro os dados para serem colocados dentro do *html* do *email*, este método, por fim, devolve em *string* o conteúdo a enviar.

Para o envio de *emails* é necessário um servidor e um serviço de envio. Para vias de teste foi utilizado um servidor gratuito de *email* hospedado por *Mailjet*. Após a fase de testes foi alterado para o servidor de *email* da empresa, o que permitiu que seja identificado como empresa Motorline.

O serviço de envio de *emails* foi utilizado o *Nodemailer*. A configuração do servidor de *emails* do *Nodemailer* é realizada através de uma chamada ao objeto do servidor com a indicação das configurações que estão no ficheiro *.env*. Esta chamada ao servidor, por sua vez, devolve um objeto que fornece métodos para enviar *emails*, sendo este, criado e utilizado sempre que se deseja enviar *emails* no servidor.

Para evitar que sempre que se deseja enviar um *email* seja necessário indicar todos os dados de configuração do servidor, foi elaborado um método que cria e devolve o objeto do servidor sempre que necessário. Sendo assim, sempre que se deseja enviar um *email* é primeiramente obtido o objeto do servidor, de seguida é invocado o método para obter o conteúdo *html* do *email* e por fim, é utilizado o objeto do servidor para chamar o método de envio de *emails*, no qual, se terá de indicar o *email* do destinatário, o assunto e o conteúdo.

#### 4.2.6 Agendamento de tarefas

Com a utilização da ferramenta *Node-Cron*, foi inicialmente programado para enviar o relatório de notificações, todos os dias, às 22 horas. Esta configuração foi realizada através da indicação da programação de horário de envio, para isso, é utilizada a estrutura segundo, minuto, hora, dia do mês, mês, dia da semana. Como o objetivo é enviar às 22 horas, os segundos e minutos foram indicados como 0 e as horas foram indicadas como 22, já o restante foi indicado com o símbolo "\*", que indica que o processo deverá ocorrer em todas as instâncias dos restantes valores, o que significa, todos os dias. A configuração final obtida foi "0 0 22 \* \* \*".

Quando o método do agendamento é invocado, são obtidos todos os utilizadores com a configuração de relatório de notificações ativa. Em primeiro lugar, para cada um destes, são obtidas todas as notificações do dia, de seguida é criado o objeto de servidor e invocado o método para obter o conteúdo de notificações através da indicação de todas as notificações do utilizador. Por fim, é enviado o *email* para o utilizador com todas as suas notificações e um *link* para aceder rapidamente ao ecrã de notificações.

### 4.2.7 Cifra de *passwords*

Aquando o registo de clientes é indicada a *password*, pelo que, esta deverá ser guardada sobre cifragem. Para isto, é utilizada a biblioteca *Bcrypt*. No processo de registo, antes do envio dos dados para a base de dados, é invocado o método de cifra da *password* com a indicação do *salt* com valor de oito. Após a execução é obtida a *password* cifrada e enviada para a base de dados em conjunto com os restantes dados.

#### 4.2.7.1 Cifra de configurações do servidor

Para a cifra das configurações do servidor, foi executado o comando de cifra da biblioteca *secure-env*, com a indicação da *password*. Como resultado foi criado o ficheiro cifrado. O ficheiro original deverá ser eliminado ou, em caso de necessidade, guardado num local seguro para evitar que durante algum ataque seja obtido os dados do mesmo.

Após cifrar o ficheiro, o servidor foi configurado para quando inicia pedir ao utilizador para indicar a *password* do ficheiro. Para esta configuração foi utilizada a biblioteca *Readline-Sync* no modo de *password*. Quando o utilizador indica a *password*, na eventualidade de esta estar errada, o servidor irá falhar, pois a biblioteca de cifra tentará decifrar o ficheiro de configurações e falhará não concluindo o processo de inicialização. Caso contrário, este continuará o processo, no qual, o primeiro passo é a utilização da *password*, para decifrar o ficheiro de configurações e carregar as variáveis de ambiente.

### 4.2.8 Documentação Typedoc

Para a documentação de código foi utilizado *typedoc*. Durante a implementação do *typedoc* foi detetado que a categorização da documentação não estava funcional devido a um problema encontrado pelos programadores da ferramenta. Deste modo reduziu-se a versão para uma com a funcionalidade ativa, mas esta, não é compatível com a versão mais atualizada de *TypeScript*, pelo que, não foi possível explorar esta funcionalidade. Para contornar o problema foi explorada outra funcionalidade menos utilizada, esta permite converter qualquer documentação em módulos, estes podem ser categorizados, o problema é que é criado um modelo genérico do código, o qual não permite a fácil identificação das tipagens de *scripts*. Estes módulos permitem também a categorizar, o que leva a um maior nível de organização da documentação.

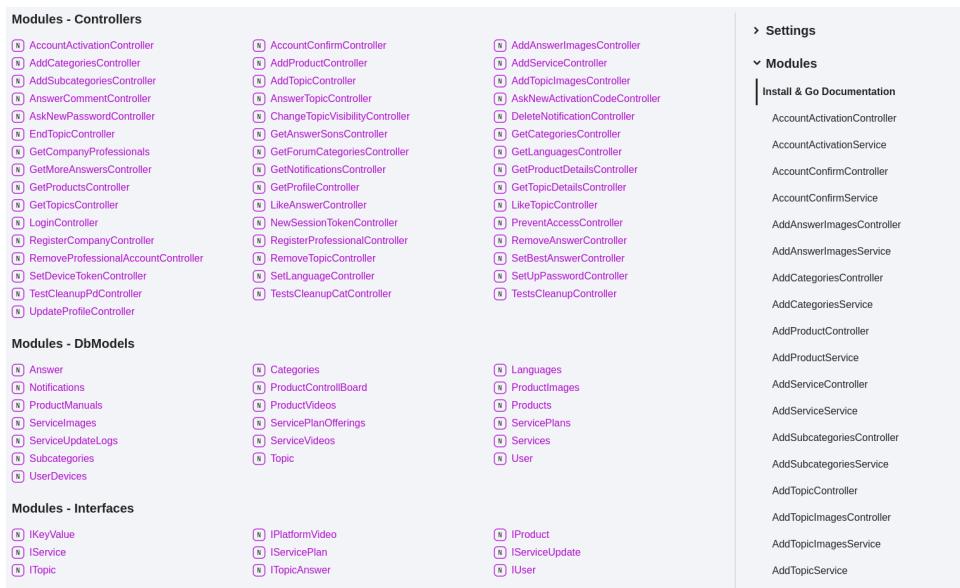


Figura 4.20: Documentação *typedoc*

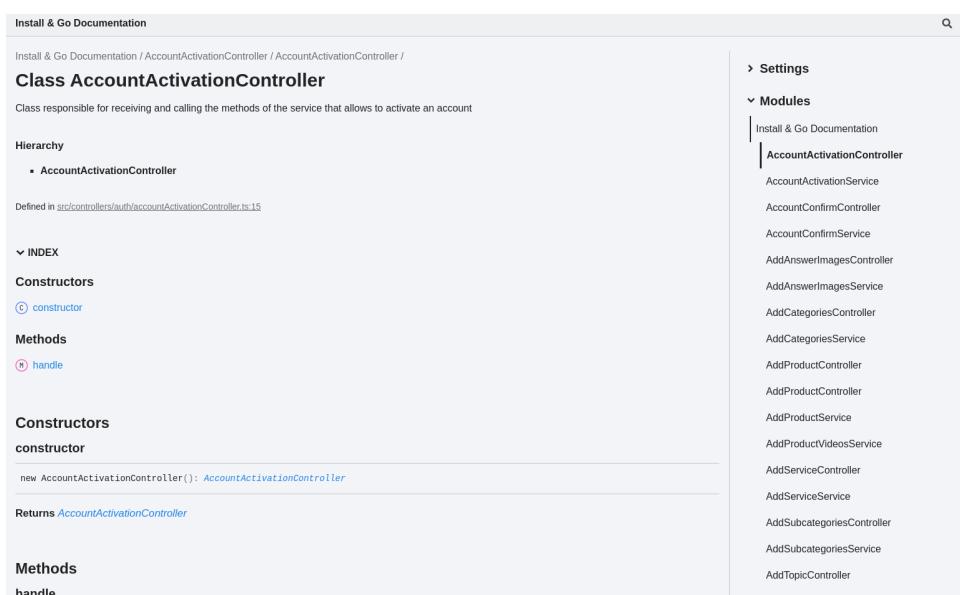


Figura 4.21: Documentação de classe *typedoc*

#### 4.2.9 Documentação Swagger

Apesar do *swagger* disponibilizar a funcionalidade de gerar automaticamente documentação a partir de comentários de código, foram encontrados alguns problemas com esta funcionalidade, o que levou a que não fosse possível gerar a documentação. Portanto, optou-se por manter a documentação manualmente com o ficheiro *JSON*. Esta ferramenta oferece diversas funcionalidades como autenticação, definição de estruturas de dados para os serviços e exemplos de respostas. Estas funcionalidades foram exploradas o que gerou um bom suporte de documentação para qualquer utilizador.

Figura 4.22: Documentação swagger

Figura 4.23: Exemplo de documentação de serviço

## 4.3 Frontend

Após o desenvolvimento do suporte *backend* prosseguiu-se para o desenvolvimento *frontend* do projeto.

### 4.3.1 Organização do projeto

Tal como o *backend*, o modelo a seguir no *frontend* foi o *MVC*.

Como recomendado de boas práticas de código limpo da *framework*, as cores do tema da aplicação foram colocadas num ficheiro separado para garantir a fácil troca do tema da aplicação. Outras aplicações de boas práticas de código limpo foram, sempre que possível, particionar o código das páginas em vários *widgets*, para deste modo, facilitar a navegação e também, a criação de *widgets* reutilizáveis que evitam a repetição de código e agilizam o desenvolvimento. Por fim, a estrutura do projeto foi organizada da seguinte forma:

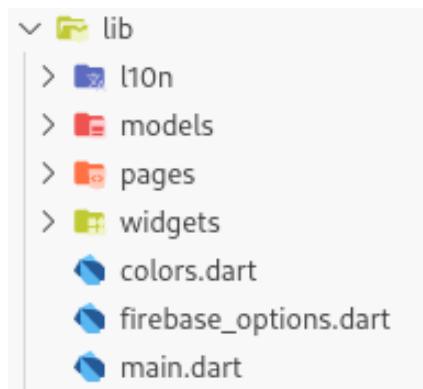


Figura 4.24: Organização do projeto

- **l10n** - Traduções da aplicação;
- **models** - Modelos de classes como *handlers*, *helpers*, *providers*, entre outros;
- **pages** - Páginas da aplicação;
- **widgets** - *Widgets* referentes às páginas;

### 4.3.2 Processo de aprendizagem

De forma a realizar a aprendizagem da ferramenta foi então procedido para a produtora da mesma, a *Google*, esta dispõe de uma lista de *workshops* e projetos a seguir para aprender as bases da ferramenta, sendo a lista a seguinte:

1. MDC-101 no Flutter: noções básicas dos componentes do Material Design | Google Codelabs
2. MDC-102 no Flutter: estrutura e layout do Material Design | Google Codelabs
3. MDC-103 Flutter: temas do Material Design com cores, formas, elevação e tipo | Google Codelabs
4. MDC-104 Flutter: componentes avançados do Material Design | Google Codelabs
5. Apps adaptáveis no Flutter | Google Codelabs

Após a realização destes *workshops*, foi possível interiorizar como funciona a base desta ferramenta e também encontrar fontes para pesquisa de *widgets* da comunidade a nível gráfico e funcional, estes provaram ser de grande auxílio no desenvolvimento da aplicação *frontend*.

### 4.3.3 Extensions

A linguagem de programação *Dart*, assim como outras linguagens orientadas a objetos, permite alterar e adicionar métodos aos objetos base da linguagem. Para realizar estas ações foram criadas extensões do objeto, neste caso, uma extensão para o objeto *string* que permite capitalizar um texto.

### 4.3.4 Handlers

Os *handlers* são porções de código que possibilitam a execução de ações perante um evento, como por exemplo, a realização de uma ação num clique no ecrã. Neste caso, os *handlers* foram utilizados para detetar o estado da aplicação e realizar ações diante destes estados. Os estados da aplicação referem-se a se a aplicação se encontra em primeiro plano, segundo plano, a resumir ou então desligada. Com estes *handlers* é possível alterar o funcionamento da aplicação de acordo com estes estados, como por exemplo, tratar de notificações da aplicação.

### 4.3.5 Providers

Os *providers* são classes criadas para ajudar com comunicações externas, neste caso chamadas à *API*. Estes foram desenvolvidos conforme os diversos modelos de dados que se recebem, como por exemplo, uma tópico do fórum. Um *provider* oferece, na presença de um modelo, um conjunto de métodos para as diferentes chamadas necessárias à *API*. Como o exemplo anterior, numa tópico existem métodos para eliminar, adicionar, editar e obter dados, sendo que, cada método terá os seus requisitos do serviço que invoca.

Estes *providers* automaticamente detetam a linguagem da aplicação e realizam o pedido ao serviço com a utilização dessa linguagem.

### 4.3.6 Helpers

Os *helpers*, como o próprio nome indica, são classes que ajudam com a realização de tarefas. Neste caso, os *helpers* foram utilizados para o auxílio de mensagens de notificações. Estas mensagens deveriam conter o nome do utilizador e a ação, traduzida na linguagem da aplicação, pelo que, foi criada a classe de *helper* de notificação, que contém um método estático para obter a mensagem de uma notificação segundo a sua ação.

### 4.3.7 Gestão de utilizadores

Um dos requisitos da aplicação é que apenas as empresas têm a possibilidade de registar-se, pelo que, apenas estas poderão registar os seus técnicos. Dado este requisito foi elaborada a página de gestão de utilizadores, apenas acessível às empresas. Nesta página poderão pesquisar pelos seus técnicos ou registar novos técnicos, sendo que, têm a obrigatoriedade de indicar o nome, *email* e o tipo de técnico. Outras ações que as empresas poderão realizar, é a visualização do perfil de um técnico com a possibilidade de bloquear o acesso ou até apagar a conta do mesmo, sendo que, esta ação é irreversível.



(a) Aviso de impedir acesso à conta

(b) Aviso de remover conta

Sempre que um utilizador sem acesso ou com a conta apagada efetua o *login*, receberá uma mensagem de erro que menciona que não possui acesso à conta impedindo a continuação do processo.

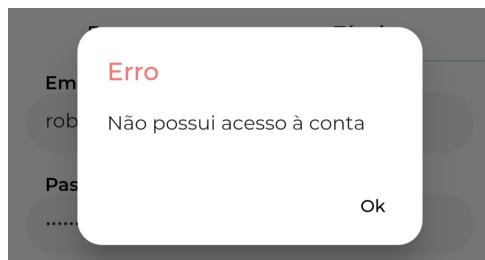
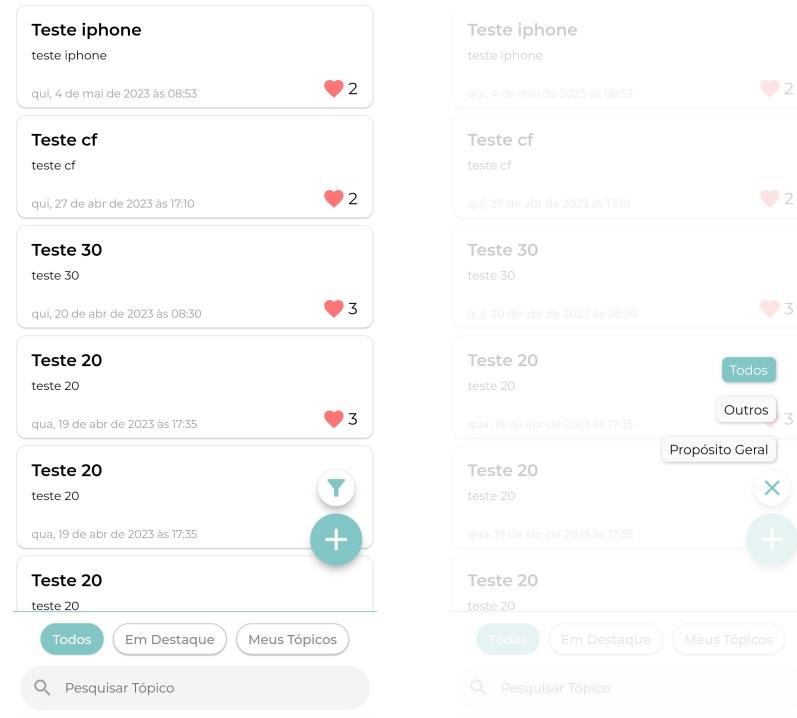


Figura 4.26: Aviso de login a conta sem acesso

### 4.3.8 Fórum

O desenvolvimento da página do fórum trouxe diversas dificuldades, entre elas, a gestão de filtros, pesquisas, a mostragem de tópicos e atualização das mesmas.



#### 4.3.8.1 Filtragem de tópicos

O grande problema com a filtragem dos tópicos é que existem 3 tipos de filtros, o de categoria de tópico, o de tipo de tópico e o de pesquisa.

Se algum filtro for alterado, os seguintes deverão ser novamente executados para garantir que todos permanecem aplicados. Inicialmente, este tipo de filtragem não era realizado, o que levava a diversos problemas, tais como, na realização de uma pesquisa, esta não era efetuada sobre os tópicos filtradas, o que levava a que a pesquisa fosse efetuada por todas os tópicos.

Outro problema detetado foi a troca de categorias, que por vezes, os filtros adicionavam-se, o que levava a que estes não apresentassem tópicos.

Sendo assim, foram criados métodos para auxílio na filtragem, tais como, uma prioridade, onde cada método invoca o método seguinte de forma encadeada. Primeiramente aplica-se o filtro de categoria, de seguida este envia o resultado para o método de filtragem por tipo e por fim, se existir algum tipo de pesquisa, os tópicos provenientes do método anterior serão filtrados.

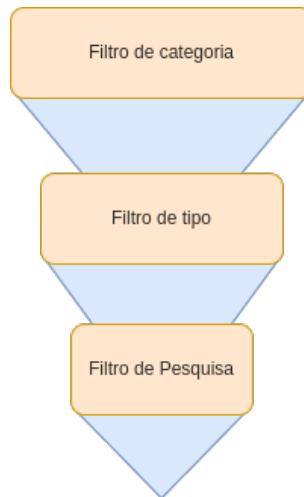


Figura 4.28: Filtragem do forum

#### 4.3.8.2 Carregamento de tópicos

Inicialmente o carregamento de tópicos fazia-se por inteiro, desde carregamento de todos os tópicos, até todos os dados dos mesmos, visto que, não existiam muitos tópicos este não seria um problema para a *API*, mas, conforme os testes foram realizados a quantidade de tópicos existentes foi gradualmente aumentando, pelo que, foi possível visualizar o tempo de demora da resposta do servidor a aumentar, assim como, o desempenho da aplicação no fórum a piorar.

A resolução deste problema proveio com uma técnica de *sliding window*, na qual os tópicos mantêm-se carregados, sendo que, a própria *framework* consegue através da lista retirar de renderização os tópicos que o utilizador não consegue ver.

Esta solução foi implementada através da utilização de três valores, quantidade de tópicos a obter, índice inicial e data do primeiro tópico. O valor de quantidade de tópicos a obter, inicialmente dez, permite limitar a quantidade de tópicos que a *API* irá processar, o que reduz o tempo de resposta. O índice inicial, indica qual o índice do primeiro elemento que se deseja obter da lista. A data do primeiro tópico, mantém uma referência temporal para obter tópicos, o que garante que a lista que está a ser a visualizada é sempre a mesma.

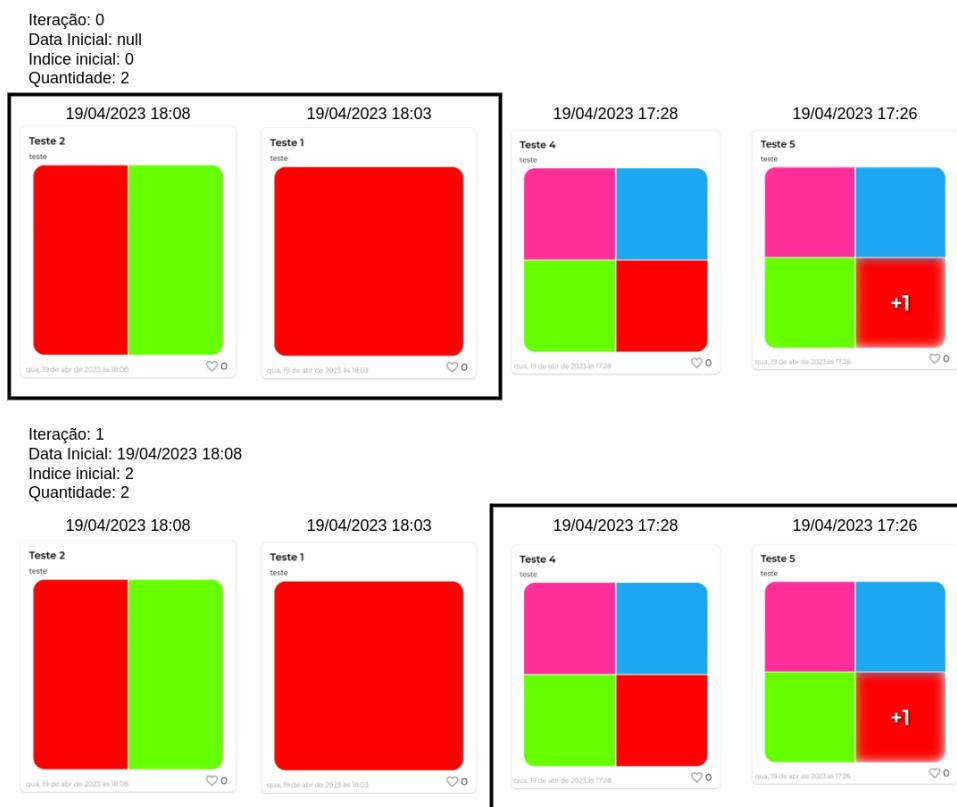


Figura 4.29: Carregamento de tópicos

Também foi reduzida a quantidade de dados carregados por cada tópico, pelo que, apenas os comentários diretos à tópico são carregados e não as respostas a estes, o que contribuiu para a melhoria do desempenho.

Por fim, sempre que o utilizador alcança o fim da lista de tópicos este poderá deslizar para carregar mais tópicos.

#### 4.3.8.3 Detalhes de tópico

A página de detalhes de tópico sofreu os mesmos problemas que a página anterior, o que levou à necessidade de aplicar a mesma solução sobre os comentários de tópico e sobre as respostas aos mesmos. Sendo assim são carregados os primeiros dez comentários e por fim, demonstrado ao utilizador quantos mais existem que poderá carregar, sendo carregados dez de cada vez.

Estes comentários podem conter respostas, sendo que estas conseguem ser carregadas também dez de cada vez e o utilizador consegue esconder ou mostrar estas.

Outro problema que surgiu no desenvolvimento da página de detalhes de tópico foi o destaque de um comentário. Este foi uma grande dificuldade, pois, com a nova implementação as mensagens não estão carregadas no momento do destaque da mensagem, pelo que, é necessário procurar a mesma nos comentários carregados, o que expande as respostas do comentário que contêm a resposta a destacar.

O destacamento das mensagens também continha um erro. Sempre que algo no ecrã é atualizado, este recarregava a animação de destaque, como solução este código foi movido para apenas ser executado no momento de inicialização do ecrã após todos os elementos se encontrarem devidamente carregados.

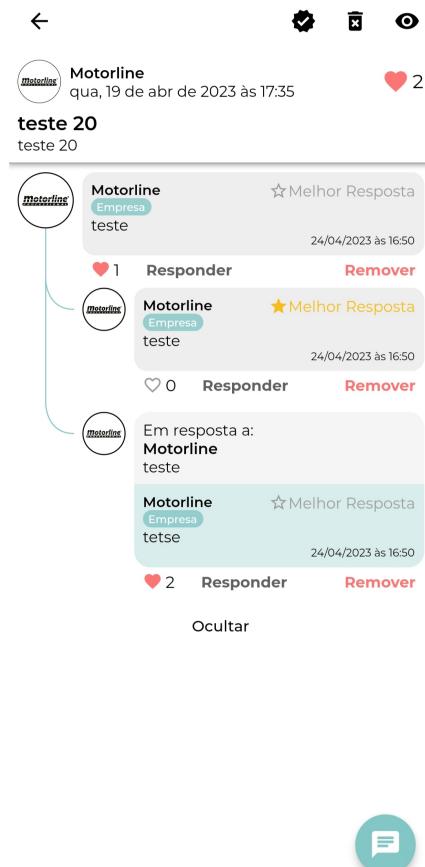


Figura 4.30: Destaque de mensagens

### 4.3.9 Firestore

A produtora do *Flutter* e do *Firebase* é a mesma, neste sentido, disponibilizou recursos que facilitam a utilização desta ferramenta pelo *Flutter*. Sendo assim, todas as imagens e vídeos de utilizadores, tópicos e comentários são guardados diretamente da aplicação para o *Firestore*, assim como, o acesso às mesmas é realizado diretamente.

Para permitir este tipo de acesso o *Firebase* disponibiliza uma ferramenta que possibilita, que através do terminal se realize a configuração da ligação entre o projeto e o servidor do *Firebase*, sendo que, no final apenas é necessário importar a biblioteca do serviço do *Firebase* que se deseja e invocar a classe do mesmo para serem realizadas ações.

Os ficheiros são organizados conforme o seu contexto. Para utilizadores, existe a pasta utilizadores, para tópicos, existe a pasta tópicos e para comentários, existe a pasta comentários.

A pasta utilizadores, como cada utilizador, apenas contém uma imagem, então são guardadas com o nome do *id* do utilizador e na eventualidade de já existir é substituída. No caso de tópicos e comentários, como podem conter várias imagens e vídeos, estes são guardados em pastas com os ficheiros referentes, que têm como nome os *id's* dos tópicos ou comentários.

The screenshot shows the Firebase Storage interface with the title "Storage". At the top, there are tabs for "Files", "Rules", "Usage", and "Extensões Novo". Below the tabs, there is a URL "gs://install-and-go.appspot.com" and a blue button "Fazer upload do arquivo" (Upload file). A table lists three folders: "Products/", "topics/", and "users/". Each folder has a checkbox, a name, a size (Tamanho), a type (Tipo), and a last modified date (Última modificação).

	Name	Tamanho	Tipo	Última modificação
<input type="checkbox"/>	Products/	—	Pasta	—
<input type="checkbox"/>	topics/	—	Pasta	—
<input type="checkbox"/>	users/	—	Pasta	—

(a) Raiz do Firestore

The screenshot shows the Firebase Storage interface with the title "Storage". At the top, there are tabs for "Files", "Rules", "Usage", and "Extensões Novo". Below the tabs, there is a URL "gs://install-and-go.appspot.com > topics" and a blue button "Fazer upload do arquivo" (Upload file). A table lists three subfolders under "topics": "1dddf308-9fd5-41ac-8d17-b2d04 10486fd/", "256c6ac4-7ce4-495c-a42a-97cd2 270f3e3/", and "2c77661f-ded5-488b-943c-d35e6 67b9194/". Each folder has a checkbox, a name, a size (Tamanho), a type (Tipo), and a last modified date (Última modificação).

	Name	Tamanho	Tipo	Última modificação
<input type="checkbox"/>	1dddf308-9fd5-41ac-8d17-b2d04 10486fd/	—	Pasta	—
<input type="checkbox"/>	256c6ac4-7ce4-495c-a42a-97cd2 270f3e3/	—	Pasta	—
<input type="checkbox"/>	2c77661f-ded5-488b-943c-d35e6 67b9194/	—	Pasta	—

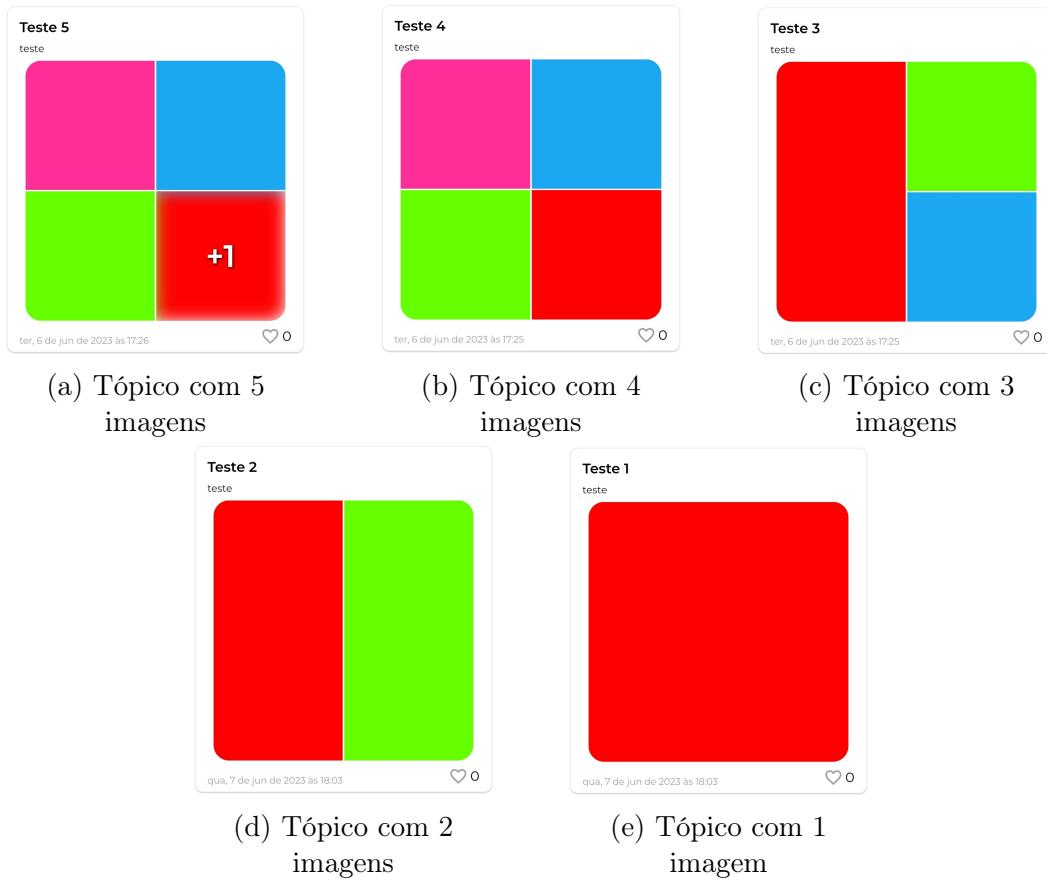
(b) Pasta topics do Firestore

#### 4.3.10 Apresentação de Imagens

A apresentação das imagens é definida em 2 níveis, o nível de pré-visualização, por exemplo, a miniatura da imagem de uma tópico no fórum e o nível do detalhe, onde é possível visualizar a imagem em ponto grande e realizar *zoom*.

Para a apresentação da miniatura da imagem foi decidido apresentar até quatro imagens, sendo que, acima de quatro imagens serão apenas demonstradas três, o que significa que a quarta imagem indicaria quantas mais existem para ser visualizadas.

Para a implementação da apresentação das miniaturas das imagens utilizou-se a biblioteca *staggered\_grid\_view*, que permite organizar imagens em grelha. Neste contexto desejava-se estruturar as imagens em diferentes aspectos e disposições, pelo que, esta biblioteca permite indicar quantas colunas e linhas existem na grelha ao criar o agrupamento de imagens. Deste modo, decidiu-se que quando são duas imagens, estas dividem a grelha, quando são três imagens, a primeira divide metade da grelha e as outras duas dividem a outra metade, quando são quatro ou mais, as quatro separam a grelha por igual. Quando existem mais do que quatro imagens, optou-se por colocar um filtro de desfoco sobre a última imagem e por cima desta quantas mais imagens existem para apresentar.



### 4.3.11 Apresentação de Imagens em carrossel

A apresentação das imagens deverá permitir que o utilizador visualize em ponto grande e realize diversas ações sobre estas. Para isso experimentaram-se diversas bibliotecas, mas nunca se alcançou o comportamento desejado, sendo assim, decidiu-se criar o próprio carrossel de imagens, sendo que, o próprio *Flutter* já disponibiliza um *widget* para tal.

O ponto de maior dificuldade para este processo foi a implementação de *zoom*, visto que, o *Flutter* não dispõe de *widgets* para tal. Por isso foi necessário primeiramente detetar gestos com o detetor de gestos da ferramenta e aplicado um *zoom* sobre o centro do gesto. Os gestos aceites foram o de pinça e o gesto de duplo clique.

O grande problema com esta solução é, uma vez que, é permitido um *scroll* horizontal de imagens, os gestos por vezes poderão não funcionar corretamente, principalmente o gesto de pinça que se efetuado na horizontal poderá resultar num *scroll*. Para resolver tal problema definiu-se que quando dois dedos são reconhecidos no ecrã, a navegação horizontal fica bloqueada e, assim que, estes são retirados, a navegação horizontal é ativada novamente.

### 4.3.12 Carregamento de Imagens

O carregamento de imagens do dispositivo poderá ser realizado por meio da seleção da galeria. Para realizar esta seleção, em primeiro lugar, foi testada a biblioteca *image\_picker*, contudo, esta biblioteca utiliza o seletor de ficheiros do dispositivo. Este seletor de ficheiros permite a seleção de qualquer tipo de ficheiros o que faz com que seja necessário um conjunto de outras verificações, para garantir que apenas as imagens são selecionadas. Isto levaria a uma possível perda do desempenho e da qualidade na experiência de utilização.

Sendo assim, de seguida foi testada a biblioteca *advanced\_image\_picker*, todavia, o problema desta biblioteca é que não permite selecionar vídeos. Posteriormente decidiu-se experimentar a biblioteca *wechat\_asset\_picker*. Esta cria uma página própria para seleção de imagens e vídeos diretamente da galeria. Também permite indicar o limite máximo de seleção e os tipos de ficheiros que o utilizador poderá selecionar, para que apenas, os tipos aceites sejam demonstrados. Por fim, o único ponto desvantajoso é que não é possível traduzir o botão de confirmação da seleção de ficheiros.

Após carregar os ficheiros para a memória, estes são enviados para o *firestorage*.

### 4.3.13 Vídeos

A maior dificuldade detetada nos vídeos foi a necessidade de um comportamento diferente nestes quando se encontram no ecrã de visualização de imagens e vídeos, visto que, ao contrário das imagens, os vídeos necessitam de um reprodutor, sendo sempre necessário verificar qual o tipo de ficheiro antes de carregar o *widget* do mesmo.

Para resolver o problema de utilizar um reprodutor testou-se uma biblioteca que permite a utilização do reprodutor nativo do dispositivo, ou seja, *Android* utilizaria o reprodutor do *Android* e *iOS* utilizaria o reprodutor de *iOS*. O grande problema com esta solução é que o reprodutor de *Android* tem os botões completamente brancos, sem nenhum tipo de fundo para os destacar, o que significa que se um vídeo branco for visualizado, o utilizador não conseguirá visualizar os botões do reprodutor.

Deste modo decidiu-se desenvolver um reprodutor próprio. Após a implementação de diversas funções como, pausar, resumir, avançar 5 segundos e recuar 5 segundos, esconder e apresentar os botões, existiam dois grandes problemas, demonstrar o vídeo em ponto grande, voltar para o mesmo tempo do vídeo em ponto pequeno e também o comportamento do reprodutor não ser completamente fluido.

Depois de uma vasta pesquisa compreendeu-se que o reprodutor do *iOS* resolia os problemas do reprodutor do *Android* através da colocação de um fundo nos botões do reprodutor. Através da biblioteca *appinio* é possível utilizar e configurar o reprodutor nativo de *iOS* em *Android*. Sendo assim, a utilização do reprodutor de *iOS* em ambos os sistemas operativos resolveria o problema. Este reprodutor permitiu a resolução de um problema menor, a visualização de vídeos em ponto grande, sendo que, dependendo da orientação do vídeo, o reprodutor altera a orientação da aplicação automaticamente voltando à orientação vertical, uma vez que, termina a visualização do vídeo em ponto grande.

### 4.3.14 Links

Uma das funcionalidades necessárias da aplicação é a utilização de *links*. Para isto, a programação *mobile* oferece duas soluções, *App Links*, *Deep Links* e *Dynamic Links*. Como mencionado na secção de tecnologias foi decidido implementar a solução de *Dynamic Links* da *Firebase*.

Para implementar esta solução, primeiramente a nível de *backend* foi necessário gerar os *links*, para isto, existem duas opções, implementação do *Firebase* no próprio *backend* ou então uma chamada ao *Firebase* com a utilização de uma chave de pedido. Em primeiro lugar foi testada a implementação do *Firebase* no próprio *backend*, contudo, esta implementação surgiu com alguns problemas, uma vez que, existem diversas configurações específicas necessárias, sendo então recomendado pelos colegas de trabalho a chamada ao *Firebase* dada à sua simplicidade.

Sendo assim, para a realização de chamadas ao *Firebase* foi utilizado o *Axios*. Este permitiu realizar um pedido com o método *POST* para o serviço de *Dynamic Links* do *Firebase*, com indicação do prefixo do projeto. Para permitir a reutilização deste código foi colocado num método onde é chamado com indicação do *link* desejado e os dados a enviar. O *link* é utilizado para, por exemplo, como numa página *web*, indicar qual página se deseja direcionar o utilizador, já os parâmetros, assim como em um *url web*, são

enviados através do próprio *link*, pelo que, estes dados são colocados na *string* do *link* o que permite a configuração perante diversas situações. Por fim, a *Firebase* retorna o *link* criado e este é colocado no *email* desejado.

Para a implementação do *frontend* foi necessário importar a biblioteca de *Dynamic Links* do *Firebase* e de seguida no código de inicialização da aplicação colocar um método para em caso de a aplicação ser aberta a partir de um *link*, este o ler. Quando este método lê o *link*, extraí a página indicada e a lista de parâmetros recebidos. Deste modo, o utilizador é redirecionado para a página do *link* com os dados recebidos.

Aquando o teste da implementação, diversas tentativas de abertura de *links* foram realizadas, mas, contudo, sem sucesso. A grande dificuldade desta implementação foi os *links* dinâmicos não permitem realizar *debug*, sendo que, ou funcionará na totalidade, ou não funcionará, o que leva a que seja complicado identificar *bugs*. A documentação do serviço foi de grande auxílio, uma vez que, estava em falta a indicação do nome do pacote da aplicação para *Android* e *iOS*. Após a colocação destes dados, tudo seguiu em completo funcionamento.

#### 4.3.15 Notificações

A implementação das notificações revelou ser a de maior dificuldade, visto que, surgiram diversos imprevistos. Para isto foi utilizado o serviço de notificações do *Firebase*. Este surge como os *links*, em duas secções, primeiramente *backend* e de seguida *frontend*.

A nível do *backend* foi necessário utilizar *Axios* para realizar um pedido ao serviço de notificações do *Firebase*. Mas assim como na criação dos *links* o serviço não indica quaisquer informações sobre erros, apenas o dispositivo poderá ou não receber a notificação.

Em primeiro lugar foi utilizado o conteúdo a enviar indicado pela documentação do serviço, mas, apenas retornava uma mensagem de erro. Posteriormente foi pesquisado outras implementações de outros utilizadores e testadas, mas, novamente surgia um erro, pelo que, decidiu-se utilizar o conteúdo indicado pelo professor de programação de dispositivos móveis, tendo este funcionado sem qualquer indicação de erro.

No conteúdo da notificação é enviado em formato *JSON* a mensagem a apresentar na notificação e como estas serão sempre referentes a tópicos ou comentários é indicado o *id* do tópico, do comentário e em caso de necessidade o *id* do comentário pai.

Este processo de notificação foi aproveitado para direcionar os mesmos dados para notificação de *email* em caso do utilizador possuir ativo as notificações por *email*, sendo gerado um *link* dinâmico com os dados na notificação.

A nível do *frontend* foi importada a biblioteca do serviço de notificações do *Firebase*, sendo esta implementada conforme a documentação. Como é necessário detetar notificações no iniciar da aplicação, durante a utilização e quando esta encontra-se em segundo plano aproveitaram-se estas deteções para implementar o direcionamento do utilizador para as páginas referentes às notificações, com a utilização dos dados recebidos.

O grande problema detetado nas notificações é que o *icon* não era apresentado corretamente, sendo que ou era demonstrado um quadrado escuro, ou nenhum *icon*. A biblioteca de notificações do *Firebase* não permite a customização do *icon*, pelo que foi decidido utilizar a biblioteca *flutter\_local\_notifications*. Esta permite a total customização das notificações, na qual é enviado o *icon* desenhado para a aplicação. Mesmo assim, as no-

tificações continuavam com o mesmo erro, pelo que, decidiu-se realizar uma pesquisa e percebeu-se que *Android* possui um novo sistema para os *icons* das notificações, deste modo, é necessário transformar estes em preto ou branco com fundo transparente e de seguida tratados pelo próprio *Android*.

Sendo assim foi realizada a transformação e novamente alterados os *icons* das notificações. Após um novo teste compreendeu-se que mesmo assim apenas o *icon* da notificação expandida teria sido alterado. Em seguida a uma nova pesquisa percebeu-se que *Android* necessita de dois *icons* de notificação para aplicar a ambas as situações, tendo sido resolvido o problema em questão.

Nesta implementação apenas um problema continuou sem resolução, a abertura de notificações quando a aplicação encontra-se terminada. Aqui foram testadas várias soluções, mas após a leitura da documentação e das soluções de outros utilizadores, compreendeu-se que o *Flutter* não permite a reconfiguração do comportamento das notificações quando a aplicação está terminada. O *Flutter* possui como futura implementação a permissão de reconfiguração do comportamento do *click* neste tipo de notificação, mas, de momento não dispõe de solução.

#### 4.3.16 Permissões

Para garantir o acesso à galeria, às notificações e à *internet* é necessário pedir permissão ao utilizador.

Para pedir permissão à galeria, foi alterado o *android manifest* que pede permissão ao utilizador assim que for necessário aceder à galeria. Sendo assim, sempre que é executado algum código de acesso à galeria, pela primeira vez, o utilizador receberá um alerta a pedir permissão de acesso, sendo que, em caso de recusa não será possível aceder à galeria.

Para acesso à *internet* realizou-se o mesmo, mas, esta permissão é pedida no ato de inicialização da aplicação em conjunto com as permissões de notificações, contudo, as permissões de notificações são encarregues da biblioteca do *Firebase*.

#### 4.3.17 Ios

Após o desenvolvimento completo da aplicação para dispositivos *Android*, foi proposto pela Motorline testar como esta se comportava num ambiente *iOS*, para isso, esta disponibilizou acesso a um dispositivo móvel *apple*, um computador, uma conta de programador e um colega de trabalho que já possuía experiência de desenvolvimento *iOS* com *Flutter*.

Para a compilação, primeiramente foi necessário configurar a ferramenta *XCode*. Esta configuração foi realizada em conjunto com o colega de trabalho.

Depois do processo de configuração, o projeto foi compilado para *iOS*. Nesta primeira compilação toda a aplicação funcionava corretamente, mas, o colega de trabalho em questão indicou algumas configurações de *design* comuns em *iOS*, como por exemplo, os botões de cancelar e confirmar se encontrarem no topo do ecrã e a explicação de como lidar com navegação, uma vez que, *iOS* não dispõe de função de voltar para trás. Outros problemas encontrados foram as notificações e o *links* de aplicação.

Para a resolução do problema de navegação foram acrescentados botões de navegação em todas as páginas necessárias. Já para implementar a sugestão dos botões de cancelar e confirmar foi declarado que se o dispositivo em que a aplicação está a correr for um dispositivo *apple*, estes botões estarão no topo da página, caso contrário, ficarão no fundo.

A resolução do problema de notificações proveio de uma pesquisa sobre qual seria a possível fonte do problema, sendo detetado que as permissões poderiam não estar configuradas. Para realizar a configuração foi necessário atribuir as permissões de *Android* para *iOS*, contudo, estas foram implementadas através de um ficheiro com o nome de *info.plist*.

Posteriormente a esta adição, o colega de trabalho indicou que no envio do pedido de notificações para o *Firebase* é necessário também indicar as configurações de *iOS*. Deste modo foi procurado na documentação como se envia as configurações de *iOS*. Com a utilização destas configurações foi alterado o pedido de notificações e de *links* dinâmicos, visto que, é um pedido do mesmo estilo.

Para além das alterações anteriores foi necessário também configurar as notificações e *links* no *XCode*. Depois desta configuração foi testado e todas as notificações, *links* de *emails* e permissões funcionaram como planeado.



# 5. Análise de resultados

O objetivo do projeto é o desenvolvimento de uma aplicação *mobile* com foco na área do fórum, pelo que, este deve permitir o registo de empresas, gestão de utilizadores, notificações, comunicações por *email*, gestão de conta e notificações do utilizador.

## 5.1 Tarefas

A nível de tarefas, o esperado, como mencionado no capítulo 1.3, seria terminar o projeto no final de março, o que não se tornou realidade, visto que, diversos imprevistos ocorreram no caminho, sendo o principal o atraso no desenvolvimento do *webscraper*. Este atraso derivou da constante atualização do catálogo, o que levava a alterações no conteúdo do *website*, sendo necessário alterar o código de leitura a cada alteração. Outro grande problema encontrado foi a alteração de alguns requisitos durante o desenvolvimento, o que resultou em mais tarefas e por si, mais tempo de desenvolvimento.

Estes imprevistos e dificuldades no desenvolvimento levaram a um atraso de um mês e uma semana, estava previsto concluir o projeto em março, mas, contudo, apenas foi finalizado no início de maio.

Apesar de todas as dificuldades, todas as tarefas previstas foram concluídas.

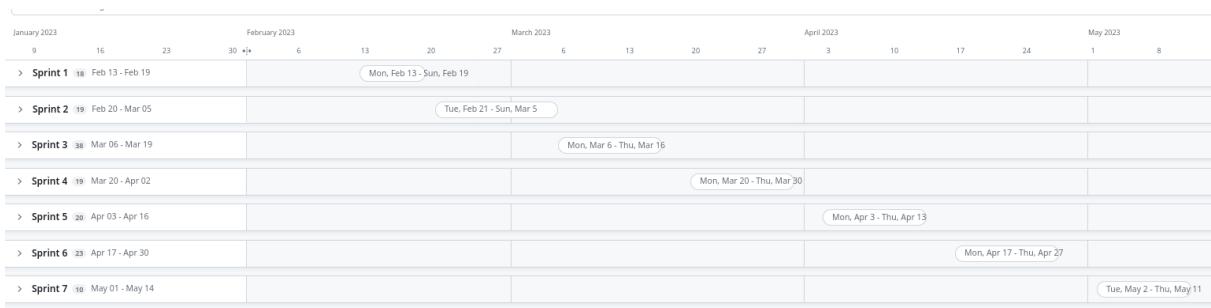


Figura 5.1: Organização de tarefas final

## 5.2 Base de dados

Um dos pontos que sofreu mais alterações durante o desenvolvimento do projeto foi a base de dados, pois, com alteração dos requisitos e do catálogo dos produtos foi necessário realizar várias alterações até alcançar a versão final.

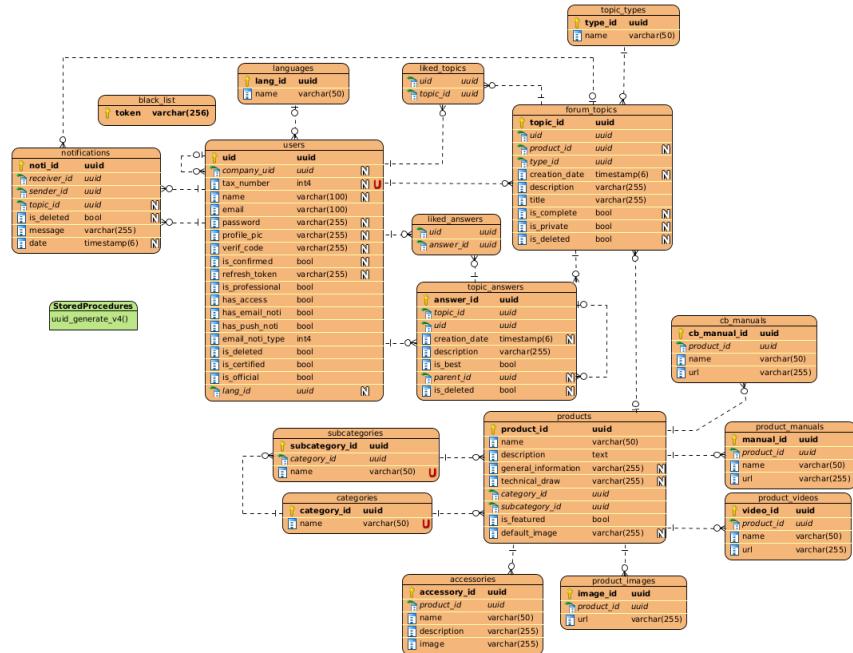


Figura 5.2: Base de dados inicial

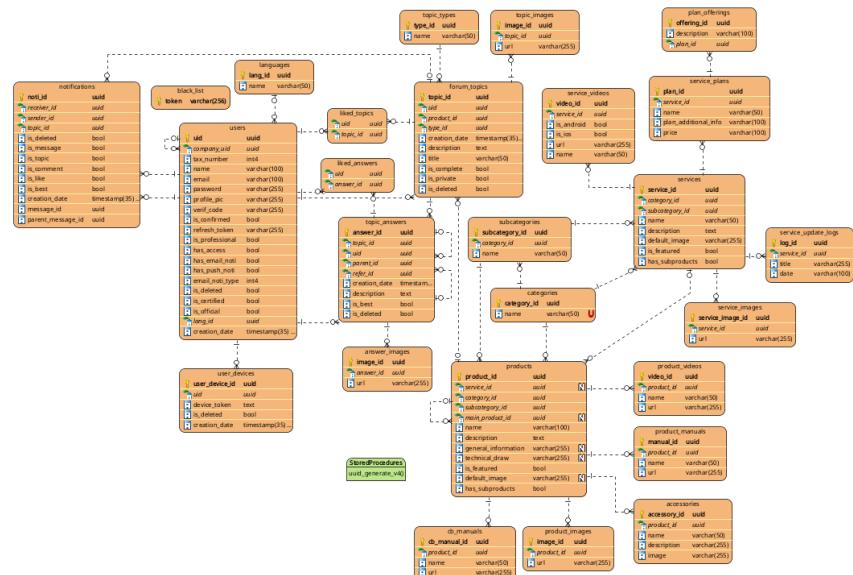


Figura 5.3: Base de dados final

## 5.3 Aplicação final

Já na aplicação final, todos os parâmetros estipulados foram alcançados, sendo apenas alguns pontos alterados, como por exemplo, as categorias de tópicos, que foram alteradas de forma a que as categorias mais importantes fossem apresentadas.

## 5.4 Opinião do cliente

Após o desenvolvimento do projeto foi apresentado ao cliente o resultado final para este realizar uma verificação se tudo encontrava-se conforme os requisitos e expectativas.

Na apresentação, o cliente dispôs de uma opinião positiva, com a indicação de que a aplicação encontrava-se conforme as expectativas e requisitos. O único ponto em falta é a testagem da solução em relação a erros e desempenho, sendo que, só poderá ser realizada após a aplicação ser lançada, dado que, não foi possível realizar testes em larga escala.



## 6. Conclusão e futuras implementações

O desenvolvimento deste projeto resultou numa solução completa preparada para uma primeira versão de teste no mercado. Este desenvolvimento conteve grandes dificuldades que permitiram adquirir novas capacidades e assimilar conhecimentos obtidos.

Entre as diversas dificuldades encontradas é importante destacar a forma como se comunica com o cliente e a importância de uma conversa onde este explica o processo de negócio da empresa, pois, a maioria dos problemas encontrados referentes a especificação do *software* poderiam ter sido resolvidas se esta conversa tivesse sido realizada no início do projeto. Já no desenvolvimento do *software*, as principais dificuldades encontradas foram a implementação dos *links* para abrir um ecrã de uma aplicação, visto que, não funcionam como os *links* na *web*, notificações, uma vez que, tinham de enviar dados específicos para o dispositivo e a implementação de *iOS*. Esta tecnologia nunca tinha sido explorada devido a limitações da produtora, que obrigam a que seja utilizado um dispositivo *Apple* para o desenvolvimento *iOS*.

Apesar das diversas dificuldades, estas agregaram para novas capacidades, assim como, a exploração de novas tecnologias. A nível da segurança foi explorada a segurança das variáveis de ambiente do *backend*, uma vez que, não teria sido possível ainda explorar estas tecnologias. A nível de *frontend* foi possível explorar desenvolvimento *cross-platform*, tendo sido esta uma experiência completamente diferente, visto que, é necessário a todo o momento pensar em ambas as plataformas de desenvolvimento. Associado ao desenvolvimento *cross-platform* foi aprendida uma nova linguagem de programação, o *Dart* e uma nova ferramenta de desenvolvimento, o *Flutter*, acompanhado de uma nova abordagem ao desenvolvimento de *software* através da utilização de *widgets*. Para além destas novas tecnologias, o *frontend* permitiu explorar funcionalidades, tais como, obter imagens e vídeos da galeria do dispositivo, reprodução de vídeos e abertura de um ecrã da aplicação através das notificações e *links*.

Futuramente seria de grande importância disponibilizar uma versão de testes da aplicação, para assim obter o *feedback* dos utilizadores sobre formas de melhorar a experiência de utilização da aplicação e a resolução de eventuais erros. Para além destas melhorias, seria relevante o desenvolvimento de um reprodutor de vídeo próprio, em vez da utilização do reprodutor de *iOS*. A implementação de funcionalidades como descarregar imagens de outros utilizadores e editar imagens antes de carregar para a plataforma, também seria notável, dado que, forneceria uma melhor comunicação entre utilizadores.

Estas novas competências adquiridas e todo o aprendizado com as dificuldades encontradas, assim como os erros cometidos agregaram para a experiência em toda a área de desenvolvimento de *software*, sendo este um dos pontos mais críticos no mercado de trabalho.



## **7. Anexos**

Com o objetivo de demonstrar em maiores dimensões todas as figuras que são de difícil percepção neste documento, assim como toda a análise de e especificação de software que não foi possível demonstrar de forma a evitar a extensão do documento e também especificações realizadas que acabaram por não ser implementadas, está em conjunto com este documento, um outro com todos os anexos referentes.



# Bibliografia

- Axios. 2023. Getting started | axios docs. <https://axios-http.com/docs/intro>. [Abril-2023].
- Bracha, Gilad. 2015. The dart programming language.
- vanden Broucke, Seppe & Bart Baesens. 2018. *Practical web scraping for data science*. Apress. doi:10.1007/978-1-4842-3582-9.
- Chuvakin, Dr. Anton A., Kevin J. Schmidt & Christopher Philips. 2012. Logging and log management.
- Developers, Android. 2023. Handling android app links. <https://developer.android.com/training/app-links#android-app-links>. [Abril-2023].
- Firebase. 2023a. Firebase dynamic links | firebase documentation. <https://firebase.google.com/docs/dynamic-links>. [Abril-2023].
- Firebase. 2023b. Operating system integrations | firebase dynamic links. <https://firebase.google.com/docs/dynamic-links/operating-system-integrations>. [Abril-2023].
- Foundation, OpenJS. 2023. Mocha - the fun, simple, flexible javascript test framework. <https://mochajs.org/>. [Abril-2023].
- Gamma, Erich, Richard Helm, Ralph Johnson & John Vlissides. 2009. *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.
- Hale, Coda. 2023. How to safely store a password. <https://codahale.com/how-to-safely-store-a-password/>. [Abril-2023].
- Halili, Festim & Erenis Ramadani. 2018. Web services: A comparison of soap and rest services. *Modern Applied Science* 12. 175. doi:10.5539/mas.v12n3p175.
- Jin, Brenda, Saurabh Sahni & Amir Shevat. 2018. *Designing web apis*. O'Reilly Media, Inc.
- Juba, Salahaldin, Andrey Volkov & Achim Vannahme. 2015. *Learning postgresql : create, develop, and manage relational databases in real-world applications using postgresql*. Packt Publishing Ltd.
- Library, Chai Assertion. 2023. Chai. <https://www.chaijs.com/>. [Abril-2023].

- Linux. 2023. crontab(5) - linux manual page. <https://man7.org/linux/man-pages/man5/crontab.5.html>. [Abril-2023].
- Mainkar, Prajyot & Salvatore Giordano. 2019. *Google flutter mobile development quick start guide : Get up and running with ios and android mobile app development.* Packt Publishing Ltd.
- Masse, Mark. 2011. *Rest api design rulebook.* O'Reilly Media, Inc.
- Mead, Andrew. 2018. *Learning node.js development : learn the fundamentals of node.js, and deploy and test node.js applications on the web.* Packt Publishing Ltd.
- merencia. 2023. node-cron - npm. <https://www.npmjs.com/package/node-cron>. [Abril-2023].
- Moroney, Laurence. 2017. *The definitive guide to firebase.* Apress. doi:10.1007/978-1-4842-2943-9.
- Selenium. 2023. The selenium browser automation project | selenium. <https://www.selenium.dev/documentation/>. [Abril-2023].
- SmartBear. 2023. Swagger specification | documentation | swagger. <https://swagger.io/docs/specification/>. [Abril-2023].
- Snell, James., Doug. Tidwell & Pavel. Kulchenko. 2002. *Programming web services with soap.* O'Reilly & Associates.
- TypeDoc. 2023. Overview | typedoc. <https://typedoc.org/guides/overview/>. [Abril-2023].
- Vanderkam, Dan. 2019. Effective typescript 62 specific ways to improve your typescript.