INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




One Widget

CLEMENTE MANUEL MACHADO MOREIRA
outubro de 2022

POLITÉCNICO
DO PORTO

One Widget

Clemente Manuel Machado Moreira

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Computacionais

Orientador: Paulo Maio
Supervisor: Ivo Pereira
Co-Orientador: Gonçalo Reais

Porto, 17 de outubro de 2022

Dedicatória

Aos meus pais e ao meu irmão por serem o meu maior apoio.

A todos os meus amigos e família por serem pessoas incríveis e por estarem sempre presentes.

Resumo

O presente documento visa descrever o processo de investigação e desenvolvimento do projeto realizado na empresa E-goi, no âmbito da unidade curricular Tese/Dissertação/Estágio, do Mestrado em Engenharia Informática, na área de especialização de Sistemas Computacionais, no Instituto Superior de Engenharia do Porto.

O projeto apresentado nesta dissertação consiste numa ferramenta que permite a configuração e gestão de canais já existentes na plataforma e a implementação de novos, num só widget que pode ser disponibilizado no website do cliente.

No entanto, sempre que se quer implementar uma nova funcionalidade num produto já existente, é necessário perceber se esta funcionalidade é realmente necessária e se esta trará benefícios ao produto.

Caso se verifique o valor, será então preciso definir o planeamento para a implementar, desenvolver a funcionalidade e, por fim, avaliar a solução de modo a perceber se cumpre os objetivos.

A descrição técnica deste documento descreve o processo de desenvolvimento da solução, resultante da análise do problema e levantamento dos requisitos de software.

Palavras-chave: Omnichannel Widget, Digital Marketing, Social Widgets

Abstract

This document aims to describe the research and development process of the project carried out at E-goi, in the scope of the course unit Thesis/Dissertation/Internship, of the MSc in Computer Engineering, in the specialization area of Computational Systems, at the Instituto Superior de Engenharia do Porto.

The project presented in this dissertation consists of a tool that allows the configuration and management of channels that already exist in the platform and some other new ones, condensating them in only one widget that can be integrated in the client's website. However, whenever someone wants to implement a new feature in a product that already exists, there's also the need to understand if that feature is actually necessary and if it brings befenits for the whole product.

In case real value is seen in this new feature, it will be necessary to define the planning to implement it, develop the functionality and, finally, evaluate the solution in order to understand if it meets the objectives.

The technical description of this document describes the solution development process, resulting from the problem analysis and software requirements gathering.

Agradecimentos

Em primeiro lugar, quero agradecer ao professor Paulo Maio pelo acompanhamento e por estar sempre disponível quando necessário.

Quero também agradecer à E-goi pela oportunidade e condições para a realização deste projeto.

Ao engenheiro Ivo Pereira, pelos conselhos e pelas reuniões frequentes de forma a acompanhar todo o processo e me motivar a cada reunião que tínhamos.

Ao engenheiro Gonçalo Reais, por toda a ajuda a apoio durante todo o processo, e por estar sempre presente e disponível para discutir qualquer assunto.

Por último, mas não menos importante, agradeço também a todos os meus amigos e família por todo o seu apoio e motivação, pois foi muito importante para concluir esta fase e sem eles o resultado não seria o mesmo.

Um muito obrigado a todos, sem vocês nada disto era possível!

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Código

Lista de Acrónimos

AHP	Analytic Hierarchy Process.
API	Application Programming Interface.
BO-UI	BackOffice User Interface.
CMS	Content Management System.
CR	Code Review.
CTO	Chief Technology Officer.
FEI	Front End Innovation.
FFE	Fuzzy Front End.
HOF	House Of Quality.
HTTPS	Hyper Text Transfer Protocol Secure.
IC	Índice de Consistência.
MVC	Model-View-Controller.
NCD	New Concept Development.
NPD	New Product Development.
OPaaS	Omnichannel Platform as a Service.
PM	Product Manager.
QA	Quality Assurance.
QFD	Quality Function Deployment.
RC	Razão de Consistência.
REST	Representational state transfer.
SaaS	Software as a Service.
SGBD	Sistema Gestor de Base de Dados.
SRE	Site Reliability Engineering.
SUS	System Usability Scale.
TL	Team Leader.
TMS	Tag Management System.
UXA	User Experience and Assurance.

Capítulo 1

Introdução

Este documento foi elaborado no âmbito da unidade curricular de Tese do Mestrado em Engenharia Informática, na área de especialização de Sistemas Computacionais, do Instituto Superior de Engenharia do Porto (ISEP).

Atualmente, encontramo-nos numa era cada vez mais digital em que a interação cliente-empresa é cada vez mais importante [1]. Consequentemente, esses avanços tecnológicos sentidos desde o inicio do século têm tornado o marketing digital cada vez mais importante para que uma empresa consiga chegar mais facilmente às pessoas e, com isso, atrair possíveis novos clientes e reter os já existentes [2].

O projeto desenvolvido começou a partir de uma proposta de estágio da empresa E-goi, com o principal objetivo de criar uma nova funcionalidade que permita a agregação de várias soluções para comunicação multicanal.

1.1 Contexto

A plataforma de marketing digital multicanal da E-goi envia milhares de campanhas todos os dias, estando esta distribuídas pelos diferentes canais de comunicação, dos quais se podem destacar os seguintes: E-mail, SMS, SmartSMS, Whatsapp Leads, WebPush, Click2call, etc.

O cliente moderno procura sobretudo serviços onde não há dependência de perfis, como designer ou desenvolvedor de software, e que facilitem a integração de serviços (tanto serviços E-goi como externos) no seu *website*. Por outro lado, a crescente multiplicidade de formas de comunicação associada à aposta dos clientes em soluções *online* resulta numa abundância de serviços que são integrados e carregados durante o acesso a um *website*. Condensando muitos desses serviços em apenas um *widget* permitirá a minimização/eliminação do ruído visual existente no *website*, tal como ilustrado nas soluções apresentadas na secção 2.3.

Para além disso, a possibilidade de adicionar ainda mais canais permite uma existência mais variada de funcionalidades. Surgiu assim a necessidade e a oportunidade da criação de uma nova *feature*, o One Widget.

1.2 Apresentação da empresa

A E-goi é uma empresa do ramo do *marketing* digital, fundada em 2008 por Miguel Gonçalves e sediada no concelho de Matosinhos. Ao longo dos últimos anos e com a crescente afluência no setor [3], a empresa tem vindo a fortalecer as suas equipas, encontrando-se

neste momento com mais de 100 colaboradores. A E-goi opera maioritariamente no mercado português e da América Latina, contando com mais de seiscentas mil contas criadas na plataforma. A empresa tem dois produtos principais, sendo que um deles, tal como a empresa, é denominado por E-goi (abordado na seção 2.1) e o outro é o Qero, que consiste numa plataforma de fidelização de clientes.

1.3 Motivação e Problema

Cada vez mais os utilizador procuram por soluções compactas e com diversas funcionalidades.

Atualmente, com a possibilidade do cliente E-goi integrar canais da plataforma (referidos na secção 2.2) no seu *website*, permite a que este possa captar e comunicar com os seus clientes com facilidade.

Estes canais encontram-se, normalmente, disponibilizados sob a forma de *widgets* que são elementos simples, sendo possível com que existam vários simultaneamente no mesmo *website* (p.e. social widgets, livechat, click2call, redirect widgets).

No caso da E-goi, existem aspetos a ser melhorados, tais como variedade e quantidade de produtos que podem ser utilizados num *website*, pelo que este trabalho tem como principal objetivo fornecer uma solução que permite disponibilizar esses produtos de uma forma mais organizada e numa quantidade/variedade maior.

Com base no que está descrito acima é possível verificar a existência de dois problemas principais na plataforma E-goi, estes sendo:

1. Impossibilidade de existir mais de dois canais sob a forma de *widget* num *website* do cliente. Mesmo que esse problema fosse resolvido possibilitando a existência de mais *widgets* para cada lado (esquerda e direita), isso causaria um ruído visual não muito apelativo;
2. Inexistência de alguns canais interessantes (p.e. social widgets, live chat, chatbot e até mesmo click2call), que podem ser integradas neste projeto.

1.4 Objetivos

O objetivo deste projeto é implementar uma solução na plataforma E-goi que permita gerir vários canais disponibilizados sob a forma de *widgets* e que seja simples e intuitiva. Isto será possível através da existência de um *widget* agregador que contém todos os outros e que seja possível a sua integração no *website* do cliente.

Para que este objetivo seja cumprido é necessário realizar as seguintes tarefas:

Análise e recolha de dados de uma ferramentas que permitam a criação, edição e gestão de *widgets* para agregação de canais. Para cumprir este objetivo, é necessário analisar e estudar algumas das soluções da concorrência, de modo a perceber se existe funcionalidades ainda não pensadas, mas que fariam sentido para o projeto;

Modelação de um design/arquitetura robusta de modo a poder ser adaptável às diferentes necessidades dos clientes;

Implementação de funcionalidades específicas, tais como costumização dos *widgets* e escolha dos canais disponíveis. As funcionalidades fundamentais devem ser definidas para que depois se possa começar o desenho e implementação das mesmas.

1.5 Metodologia de trabalho

Para a realização deste documento foi adotada uma metodologia que se baseia no artigo *Introduction to Design Science Research* [4]. Este apresenta uma enumeração de Atividades a serem realizadas durante a investigação.

Inicialmente, e tal como mencionado na Atividade 1 denominada *Problem identification and motivation* é necessário definir o problema e explicar a sua motivação, ou seja, a razão pelo qual a solução apresentada vai resolver esse problema.

De seguida e segundo a Atividade 2, *Define the objectives for a solution*, é necessário definir quais os objetivos da solução, de modo a cumprir com a motivação mencionada acima.

Na Atividade 3, *Design and Development*, é a fase onde a solução começa verdadeiramente a ser construída. De acordo com o que for definido anteriormente, é então desenhada a solução, tendo em conta o estudo do estado da arte feito previamente, de modo a que haja uma visão geral do que desenvolver e que irá servir como um guia para a próxima fase que é a de Implementação, fase esta em que é já é possível ver os frutos de todo o estudo que foi feito anteriormente.

A Atividade 4 e *5*, *Demonstration* e *Evaluation*, o primeiro que visa a demonstração do que o que foi implementado resolve realmente o problema inicialmente definido e o segundo que nos permite avaliar se os objetivos foram bem executados. Esta fase está relacionada com a Experimentação e Avaliação.

Por último, a Atividade 6, *Communication*, consiste em utilizar o melhor método para comunicar aos *stakeholders* de todo o trabalho realizado. Neste caso, tratar-se de uma tese de Mestrado e considerasse os arguentes como os *stakeholders*.

1.6 Estrutura do documento

O presente relatório encontra-se dividido em oito capítulos principais, cada um destes referente a uma parte especifica do desenvolvimento do projeto, sendo estes:

Introdução: Neste capítulo é dada uma introdução ao projeto, para que se possa começar a ter uma ideia geral do projeto e para que se entenda com clareza os capítulos que advêm. Paralelamente a isso, é necessário dar uma introdução à empresa onde o trabalho está a ser desenvolvido para que se possa perceber o contexto e propósito do mesmo.

Contextualização e Estado da Arte: Neste capítulo é feita uma contextualização de tópicos (conceitos de negócio) que não necessariamente estão referidos na Introdução, mas influenciam a maneira como este é desenhado e desenvolvido. É preciso então entender o contexto em que este projeto se encontra, para que nos capítulos seguintes as alterações tomadas façam sentido.

São ainda apresentadas e descritas soluções similares à que está a ser desenvolvida. Para além disso, são feitas comparações de *features* e dos propósitos de cada solução (análise comparativa), que serviram como um primeiro ponto de partida sobre o que faz sentido para o projeto ou não, ou até mesmo se faz sentido acrescentar uma *feature* que não exista em nenhum dos produtos analisados.

Análise: Neste capítulo é feita, primeiramente, uma análise de valor que nos irá permitir confirmar a necessidade do desenvolvimento deste projeto de acordo com os requisitos definidos anteriormente.

De seguida, é feito um levantamento de requisitos tendo em conta as necessidades do cliente. Posteriormente, são feitas decisões em conjunto com o Team Leader (TL), Product Manager (PM) e o Chief Technology Officer (CTO) para percebermos quais requisitos manter/modificar/eliminar, tendo em conta a sua relevância e custo/beneficio.

Design da Solução: Neste capítulo é apresentado o design desenvolvido até à data, não deixando de apresentar outras abordagens possíveis do problema, referindo os pontos fracos e pontos fortes de cada uma delas, adotando ciências e boas práticas de Engenharia Informática.

Implementação: Neste capitulo são apresentados os pormenores mais técnicos no desenvolvimento do projeto, assim como as metodologias usadas e alternativas que poderiam ter sido aplicadas.

Experimentação e Avaliação: Neste capitulo são realizados vários testes ao projeto, de modo a verificar a qualidade de vários parâmetros que são avaliados (p.e. usabilidade, performance).

Conclusão: Neste capitulo final são apresentadas conclusões relativamente ao trabalho realizado, mencionando também o trabalho futuro a ser realizado caso se queira progredir com o projeto. Há ainda referência aos objetivos e ao problema, resumindo como estes foram resolvidos e abordados.

Capítulo 2

Contextualização e Estado da Arte

Para um melhor enquadramento do projeto, é necessário perceber os conceitos associados ao mesmo, pois só assim é possível entender a motivação das decisões feitas ao longo do decorrer do trabalho.

São ainda apresentados conceitos relacionados com o *digital marketing*, tais como, captações e listas de contactos. Para além disso, também serão destacados algumas funcionalidades presentes no produto E-goi, como o Connected Sites e a utilização de software *open-source*, como é o caso do Matomo.

Posto isto e de forma a perceber melhor como poderá ser feito e implementado este projeto é necessário conhecer soluções que já existem no mercado, através de um estudo do estado da arte, e que têm em consideração o mesmo conceito base, que neste caso é a agregação de vários canais de comunicação.

, é feita ainda uma análise comparativa para haver uma ideias dos prós e dos contras de cada uma das soluções e qual a viabilidade de cada uma delas.

2.1 E-goi

O E-goi é um produto focado no marketing digital e tem como missão a criação de soluções para o auxilio no marketing digital, acessíveis a todos os profissionais de marketing do mundo.

Este produto é bastante complexo e encontra-se, até à data, em constante desenvolvimento e evolução. Para além disso, durante o processo de desenvolvimento de todas as componentes que constituem o E-goi são utilizadas várias tecnologias, tais como Java, Python, Angular, HTML, CSS, Javascript, PHP, entre outras.

Este software é comercializado baseado no modelo Software as a Service (SaaS) que consiste num modelo de subscrição (p.e. mensal, anual) e em que o sistema se encontra centralizado e fica à responsabilidade do fornecedor do *software*. Tal como mencionado na seção 1.1, a plataforma da E-goi disponibiliza diferentes canais de comunicação e para além disso permite também a captação de *leads* através de *landing pages*, *embedded forms*, *signup pages*, etc. Estas *leads* são armazenadas na conta do cliente e a partir daí este pode fazer todo o tipo de campanhas que pretender utilizando esses mesmos contactos.

2.1.1 Cliente vs Subscritor

Ao longo desta dissertação, os termos cliente e subscritor serão bastante frequentes, pelo que esta subsecção visa em esclarecer a diferença entre os dois. O termo cliente é utilizado para definir os utilizadores da plataforma E-goi e que usufruem dos componentes da mesma,

enquanto que subscritor refere-se aos utilizadores que por via de captação ou importação de contactos, acabaram na lista de contactos (secção 2.1.3) de um cliente.

2.1.2 Connected Sites

Connected Sites é uma funcionalidade presente na plataforma E-goi, que permite, através da inserção de um pequeno *script* no código fonte de uma página, a injeção de vários elementos (produtos E-goi) que tenham sido selecionados na página do Connected Sites.

Exemplificando com o caso de um utilizador que criou um formulário pop-up com os campos "nome"e "email". Considerando que o utilizador já seguiu os dois primeiros passos, sendo estes, a inserção de um domínio que se encontre ativo (figura 2.1) e a colocação do excerto de código (figura 2.2), fornecido pelo E-goi, no código fonte da página do cliente), este pode então selecionar o formulário criado inicialmente (figura 2.3).

Ao fazer essa seleção no Connected Sites, o formulário será então injetado na página e sempre que alguém aceder ao respetivo domínio, o formulário será carregado e exibido juntamente com a página (figura 2.4).

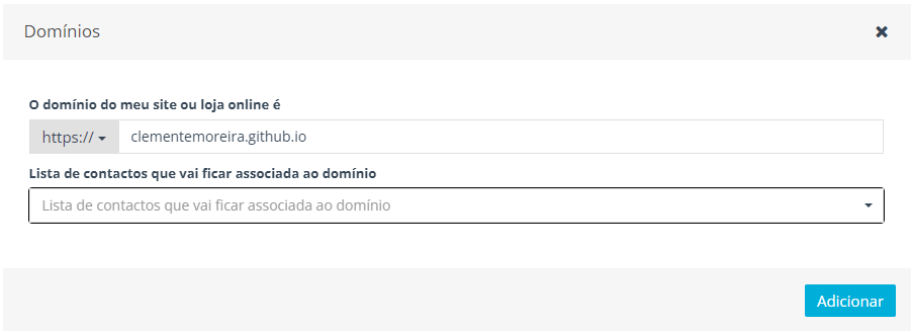


Figura 2.1: Adicionar domínio no Connected Sites

Figura 2.2: Apresentação do snippet para colocar no código fonte da página

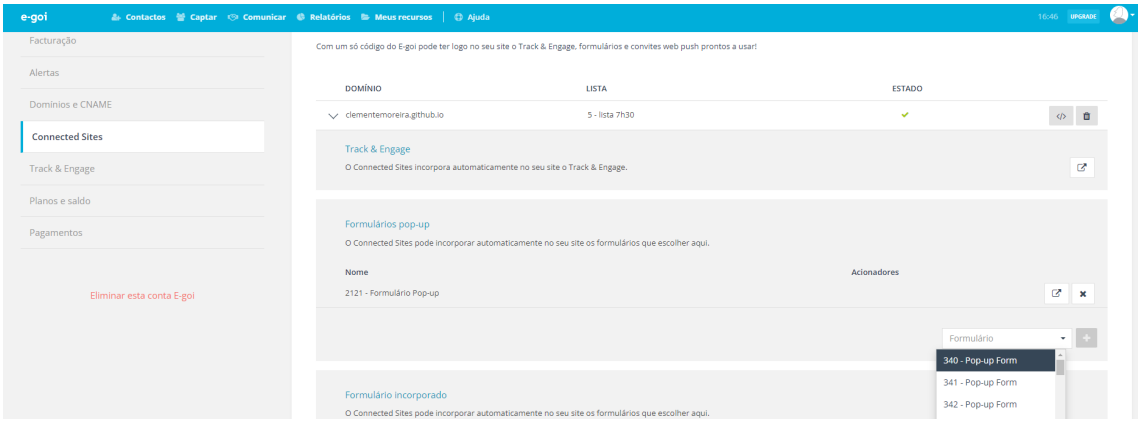


Figura 2.3: Escolher a formulário no Connected Sites

Figura 2.4: Pop-up aparece na página

2.1.3 Listas de Contactos

A lista de contactos é a componente onde podem ser encontradas todas as informação dos subscritores, obtida através do diferentes meios de captura (secção 2.2) de contactos. Para além disso, os contactos podem ser importados manualmente ou por Excel [5]/CSV caso já existam registos anteriores que não estejam na lista de contactos.

Com isto, é possível construir listas de contactos extensas para que, assim, as campanhas de comunicação (campanhas que permitem comunicar com o cliente, como por exemplo, campanhas de email e sms) tenham uma maior rede de contactos a que possam chegar (para os casos de campanhas de comunicação onde é necessário o uso de uma lista de contactos).

Com a possibilidade do número de contactos poder ascender aos milhares e até mesmo milhões é necessário existirem campos únicos, ou seja, um identificador que permita distinguir um subscritor do demais (p.e. o email).

Tags

Numa lista de contactos, as *tags* são utilizadas para segmentar os subscritores, isto é, as *tags* servem como forma de distinguir grupos de subscritores. Por exemplo, se um subscritor meteu que gosta de bananas, atribui-se a *tag* "bananas", se for de maçãs coloca-se a *tag* "maçãs". Na criação de uma campanha de maçãs, esta pode apenas ser enviada a quem tem a *tag* maçãs e o mesmo acontece para as bananas. Caso um subscritor tenha as duas *tags*, ambas as campanhas serão enviadas.

A configuração para a atribuição de *tags* é feita aquando da criação da campanha de captação (secção 2.2), mas também pode ser editada posteriormente. A atribuição de *tags* é, então, importante pois possibilita o envio de campanhas de comunicação, tendo em conta as *tags* que são atribuídas aos subscritores.

2.1.4 Matomo

Fundado em 2007, por Matthieu Aubry, o Matomo [6] é uma ferramenta de análise de dados que possui uma plataforma *open-source* para *web analytics* que corre num *webserver* PHP/MySQL. A junho de 2018, esta ferramenta era utilizada por cerca de 1 milhão e 400 mil *websites* espalhados por mais de 190 países e disponível em mais de 50 idiomas.

Auto-proclamada como sendo uma alternativa para o Google Analytics e visa proteger os dados e privacidade dos utilizadores, dando também total controlo aos utilizador sobre os dados que circulam nas suas páginas, respeitando também as regras de privacidade estabelecidas por regulamentos como o GDPR, LGPD, PECR, entre outros.

Atualmente, o E-goi utiliza o Matomo para recolher dados relativos a ações/comportamentos dos visitantes das campanhas dos cliente E-goi. Por exemplo, é possível saber onde os utilizadores interagiram mais na página, onde e quais as campanhas que tiveram maior número de visitas e adesões, etc. Com o Event Tracking disponibilizada pelo Matomo é possível calcular métricas para todo o tipo de elementos da página e, dessa forma, ter um melhor controlo e perceção do que se passa no *website*.

Matomo Tag Manager

Semelhante a como um Content Management System (CMS) traz toda um flexibilidade para publicar conteúdo num *website*, um Tag Management System (TMS), como o Matomo Tag Manager [7], permite facilmente incorporar recursos próprios ou de terceiros numa página, isto é, permite gerir e unificar todos os *snippets* de marketing e rastreamento num único lugar. Os *snippets* em si são tipicamente desenvolvidos em JavaScript ou HTML e permitem integrar funcionalidades no *website* tais como, *pop-ups* e *surveys*, *newsletter signups*, *social widget*, entre outros.

Para conseguir atingir esta finalidade é necessário ter em conta três componentes principais:

> **Tags**: *Snippet* que será incorporado na página, contendo informação não só visual, mas também funcional (eventos, ações) do conteúdo a ser carregado. Não confundir esta *tag* com a *tag* que é aplicada aos contactos e que está descrita na secção 2.1.3.
>
> **Trigger**: Define a condição ou condições em que uma *tag* deve ser despoletada. Por exemplo, no caso de o dono da página (cliente) querer que um *pop-up* apareça quando um utilizador carrega num certo elemento da página. Outro exemplo poderia ser caso o subscritor se encontrasse na página do cliente por mais de 30 segundos,

fazia aparecer uma *signup newsletter*. Com os *triggers*, existem imensas possibilidades e combinações que se adaptam às necessidades de cada um.

Variáveis: Permite obter dados que podem ser utilizados posteriormente por *tags* ou *triggers*. Por exemplo, no caso de existir uma *tag* que define uma modal de 100px por 100px, não existe responsividade e a janela terá esse tamanho independente do tamanho do ecrã do dispositivo que está a ser usado passar acessar a página. No caso do uso de uma variável, é possível saber, por exemplo, o tamanho do ecrã do dispositivo e ajustar o tamanho da modal de acordo com o dispositivo atual.

No caso da criação de um One Widget, é esperado que a *tag* contenha não só código HTML, como também JavaScript para ambos os aspectos visuais e comportamentais, respetivamente. Para além disso, o *trigger* que fará mais sentido neste caso é o de janela carregada, ou seja, no momento em que uma página é carregada, incluindo estilo e imagens. A decisão da escolha do *trigger* deve-se simplesmente ao facto do One Widget não ser um elemento condicional da página, ou seja, faz sentido o elemento estar sempre disponível do que existir uma condição para que este se torne disponível, ao contrário do que pode acontecer com os *pop-ups*, por exemplo.

2.2 Captação

Para que seja possível se obter uma rede robusta de subscritores interessados no produto é necessário, em primeiro lugar, fazer a captação dos mesmos, que pode ser feita através de diversos produtos e-goi, tais como, páginas de inscrição, formulários incorporados, formulários *pop-up*, whatsapp leads, etc.

A informação obtida (através do digital marketing) será bastante valiosa e não deve ser desperdiçada, pois dá a oportunidade de entender a perspetiva dos subscritores, e outras variáveis que possam estar envolvidas, em grande detalhe, ao contrário de outras áreas do negócio [8].

Esta componente é bastante importante pois também a possibilita a segmentação, ou seja, através da atribuição de *tags* (secção 2.1.3) aos subscritores que interagiram com uma certa campanha.

2.2.1 Formulários

Para aumentar o número de contactos são usados maioritariamente formulários (de inscrição) onde é possível obter a informação do utilizador e por seguinte adicioná-la na lista. Se o formulário preenchido já tiver informação (campo único) existente, então será enviado um email de confirmação, para ter a certeza de que quem quer alterar a informação é o próprio dono e não um impostor.

Existe ainda outro caso que se verifica quando uma conta foi removida da lista de contactos e volta a ser colocada (através da repetição do campo único), então denomina-se esse caso como reinscrição, em que é enviado novamente um email de confirmação, para a E-goi estar certa da autoria dessa informação.

2.2.2 Whatsapp Leads

Uma das principais formas de captação existentes na plataforma é o whatsapp leads que, na sua essência, representa um *widget* que se encontra na parte inferior do *website* e que ao ser clicado, abre um formulário (ao criar o whatsapp leads é possível definir os campos que o formulário terá e definir os obrigatórios). Depois de submetido, a informação do subscritor é captada e , posteriormente, este é redirecionado para o contacto do Whatsapp que foi definido no momento da criação desse *widget*.

Figura 2.5: Widget whatsapp

2.2.3 Webpush

Outra forma de captação disponível na plataforma E-goi são as captações para notificações *webpush*. Sendo que o *webpush* em si é uma forma de comunicação, é necessário de alguma forma obter interessados para receberem as notificações e é aí que a captação para *webpush* entra. Tal como pode ser visto na figura 2.6, este tipo de captação é representada por uma janela em que o subscritor ao clicar para receber notificações é feito um pedido ao Firebase para gerar um *token* para aquele site no *browser* do subscritor. Depois, esse *token* é guardado e é criado um contacto na lista com este. Quando é feito o envio de uma notificação webpush, é esse o *token* que é utilizado.

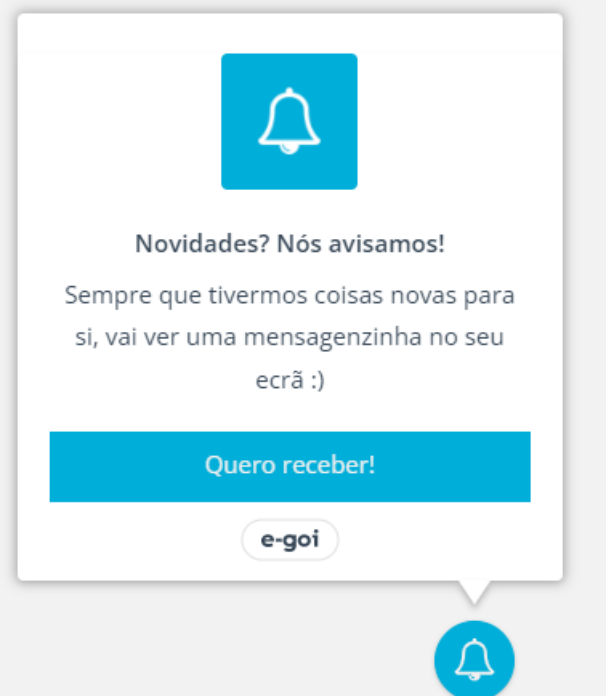


Figura 2.6: Widget webpush com confirmação

2.3 Soluções para criação de um agregador de canais

Apesar da agregação de vários canais de comunicação parecer um conceito simples, existem diversas variantes e caminhos que se podem tomar no desenvolvimento de um software com esse objetivo, portanto é necessário analisar diversas ferramentas que permitem a criação de um *widget* agregador de canais de comunicação e perceber qual a sua finalidade e o que foi implementado.

Para isso, foram identificadas diversas ferramentas (sem critério de seleção, a não ser terem pelo menos algo de diferente com as demais) que serão analisadas neste capítulo e, por fim, comparadas e a partir daí serão retiradas as conclusões tidas como pertinentes.

2.3.1 Elfsight

A empresa Elfsight [9] foi fundada no ano de 2012 pelos russos Vladimir Fedotov e Andrey Yusupov. Hoje em dia, a Elsight ajuda mais de um milhão de donos de *websites* a aumentar vendas, coletar *leads*, etc.

A Elfsight apresenta um plano grátis que suporta apenas um *website* e que permite cerca de 200 *views* por *widget*, ou seja, se for criado um *widget* agregador de canais de comunicação, este pode ser carregado no página um máximo de 200 vezes (por mês), independente de serem utilizadores que já tinham visto antes ou não. Para além disso, existem ainda os planos pagos que funcionam para um número ilimitado de *websites*, sendo que ainda têm direito a um serviço de instalação grátis e o logótipo da Elfsight é removido. Quanto maior o valor do plano, mais prioritário é o suporte ao cliente e maior o número de *views* que um *widget* pode ter, chegando a um máximo de cinco milhões de *views*.

Dos produtos oferecidos pela Elfsight, destacam-se dois, sendo que um deles é o que será usado para análise comparativa com o One Widget. Em primeiro lugar, analisou-se o All-In-One-Reviews que permite ao utilizador colocar um *widget* no seu *website* para que o cliente possa, posteriormente a uma compra, deixar a sua avaliação em diversas plataformas que são disponibilizadas pelo All-In-One-Reviews, como por exemplo, Airbnb, Edmunds e OpenTable (tal como pode ser visto na figura 2.9).

De seguida e mais importante para esta dissertação temos o All-In-One-Chat que permite agregar vários canais de comunicação num só *widget*. Isto permite ao utilizador escolher entre os diferentes canais de comunicação (facebook messenger, telegram, viber, etc, tal é ilustrado na figura 2.8) sendo que, o único problema é que estes canais têm todos o mesmo propósito, ou seja, o apoio ao cliente, daí o nome All-In-One-Chat.

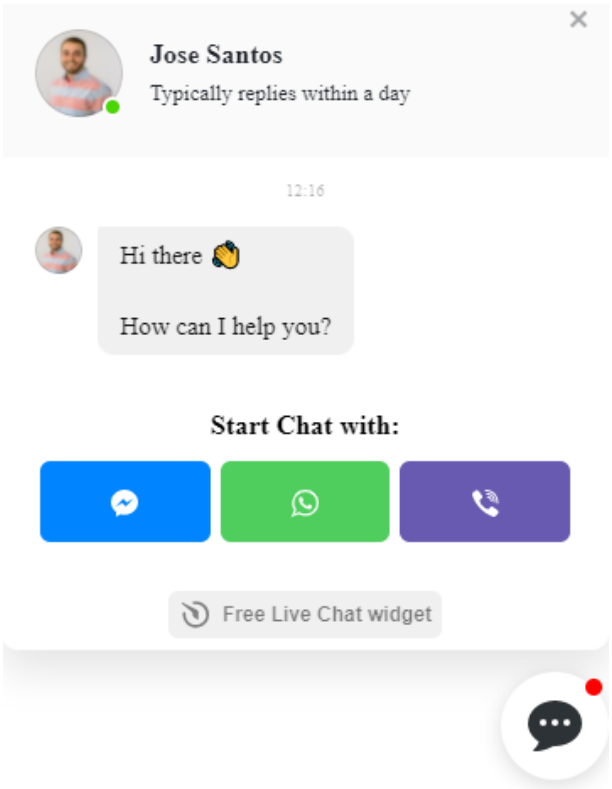


Figura 2.7: Elfsight

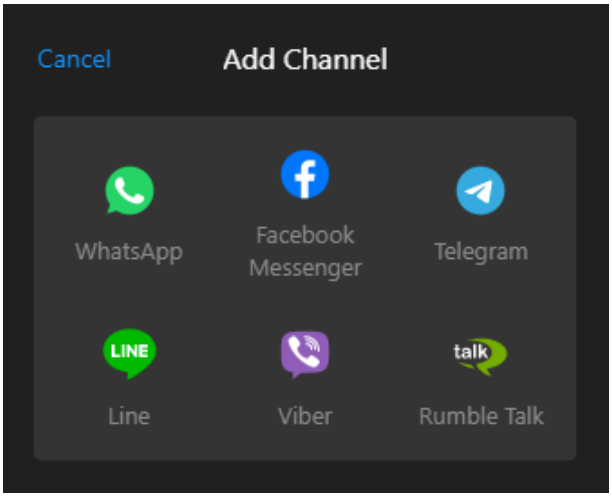


Figura 2.8: Canais do Elfsight All-in-One-Chat

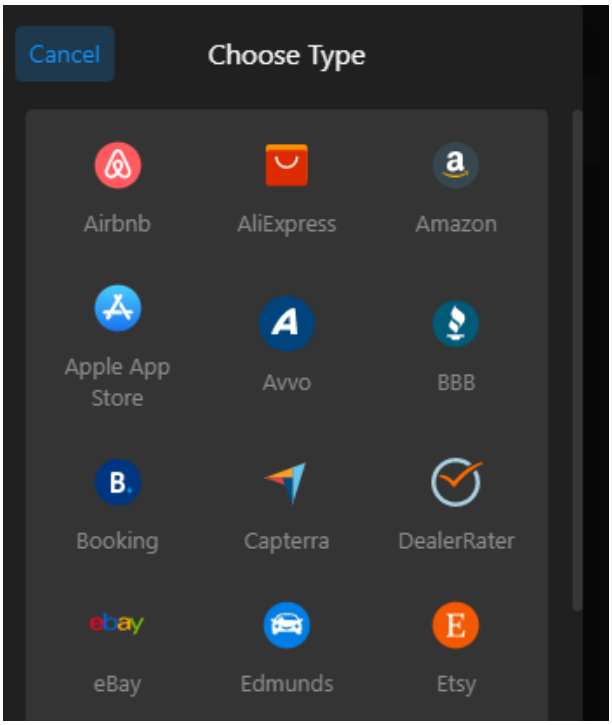


Figura 2.9: Aplicações do Elfsight All-in-One-Reviews

2.3.2 MessageBird

Fundado em Janeiro de 2011 por Robert Vis, o MessageBird [10] é um Omnichannel Platform as a Service (OPaaS) que visa em facilitar a comunicação entre negócios e os seus clientes a partir de diversos canais.

O MessageBird disponibiliza um plano grátis com todos os canais de comunicação para até 50 contactos diferente (com conversas ilimitadas com os mesmos). No primeiro plano pago apenas é alterado o número máximo de contactos diferentes que podemos ter por mês. Já no segundo e terceiro planos existem mais benefícios, como por exemplo, acesso a APIs,

custom email e chamadas *live* com um especialista em *Onboarding*, etc. Como o terceiro plano é *custom*, o preço irá variar dependendo do máximo de contactos por mês necessários para o negócio.

De todos os produtos analisados, o *Omnichannel Widget* da MessageBird é o mais complexo. Tal como o *All-In-One-Chat* visto na subsecção 2.3.1, o produto da MessageBird é também destinado ao suporte ao cliente, apesar deste ser baseado em agentes, dando um aspecto mais "profissional"ao produto. Tanto a gestão destes agentes (feita por um gestor), como a gestão das mensagens de clientes (feita pelos agentes) são feitas no próprio *website* da MessageBird.

O fluxo funciona desta forma:

1. O utilizador interage com o *widget* (este que se encontra ilustrado na figura 2.10);
2. O utilizador seleciona um dos canais de comunicação;
3. O utilizador é redirecionado para esse canal de comunicação, caso este não seja controlada pelo MessageBird (p.e. Facebook Messenger, Whatsapp, Instagram, etc). Caso o canal escolhido permita começar a conversa sem redirecionamento (normalmente estes canais de comunicação não precisam de autenticação), então o utilizador pode conversar de imediato;;
4. No instante do envio da mensagem todos os agentes receberão a mensagem na lista de espera;
5. O primeiro agente a responder será o responsável pela conversa. O agente terá a informação sobre o canal que o utilizador está a utilizar e sobre alguma informação básica que esse canal ofereça (p.e. com um utilizador que mande mensagem pelo Messenger, será possível saber o nome de utilizador);
6. Logo que a conversa termine, o agente pode dar esse ticket como terminado.

A parte que torna o MessageBird diferenciado, para além do suporte baseado em agentes, tal como ilustrado na figura 2.11, é o facto de existir um livechat próprio da marca e ainda que os agentes conseguem responder a tickets, provenientes dos diversos canais, através da própria plataforma do MessageBird (sendo que existe uma ligação que é feita posteriormente para associar as contas dos diferentes canais à plataforma, para que exista autorização para obter a *inbox* de cada canal).

Por fim, existe ainda uma lista de contactos que pode ser vista por todos os agentes, que contém todos os utilizadores que iniciaram uma conversa. É possível importar contactos (através de um ficheiro em formato CSV) para essa lista e adicionar contactos de forma manual. Estes contactos, após a exploração da plataforma, são meramente ilustrativos e não têm qualquer função.

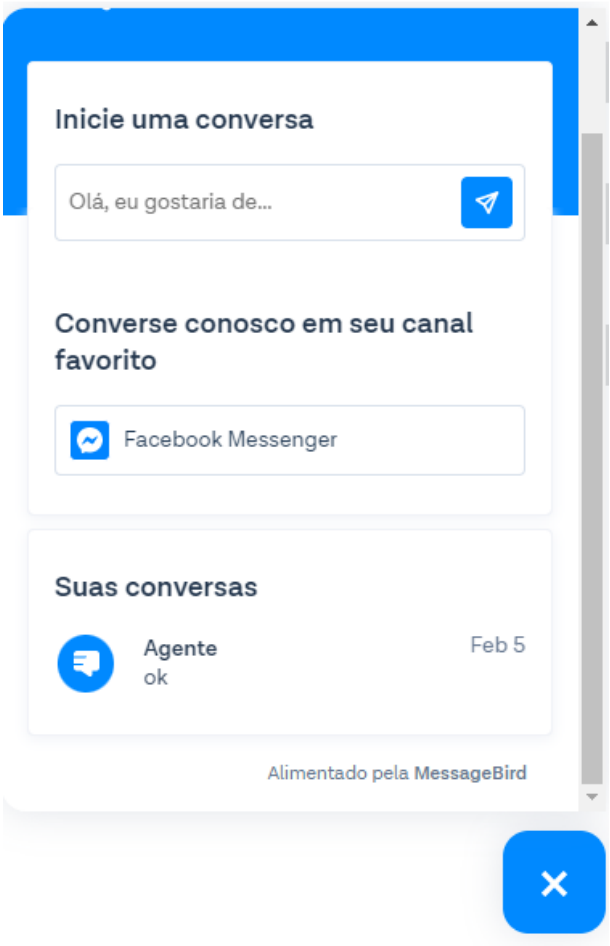


Figura 2.10: MessageBird

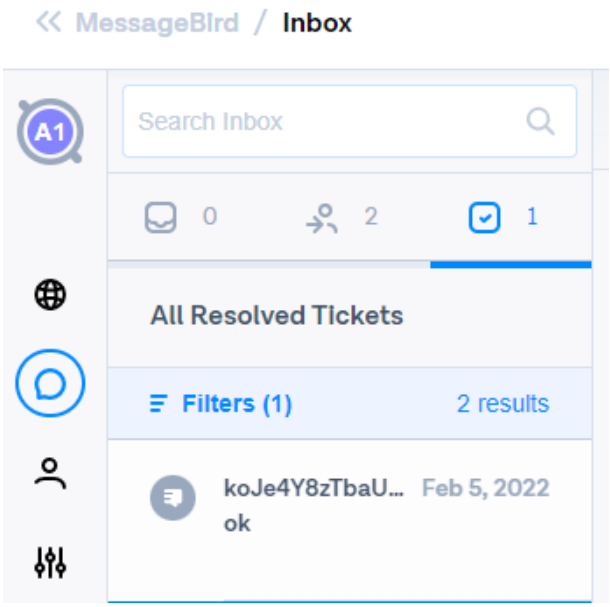


Figura 2.11: Inbox de um Agente no MessageBird

2.3.3 Chaty

Premio foi fundada em Agosto de 2018 por Tomer Aharon e Gal Dubinski e é uma empresa especializada em construir plugins (sendo Chaty [11] um deles) para o WordPress, Shopify, entre outros.

Para este *plugin* existem três planos (Basic, Plus e Agency) que apenas são distinguidos pela quantidade de *websites* em que podem ser inseridos (1,5 e 50, respectivamente). Para além disso, o pagamento pode ser feito tendo em conta três espaços temporais diferentes, sendo estes, 1 ano, 2 anos e permanente. A vantagem deste produto é que qualquer um dos planos que seja escolhido durante uma certa quantidade de tempo é que logo que faça a primeira compra, o produto fica na posse do comprador para sempre. O único problema é que não haverá atualizações e suporte ao software caso este deixe de ser pago.

O Chaty permite agregar diversos tipos de canais sob a forma de *widget* (ao contrário dos anteriores que permitiam apenas agregar canais com o objetivo do suporte ao cliente) sendo estes:

1. Comunicação (Email, Sms, Skype, Instagram, etc);
2. GPS (Waze, Google Maps);
3. Canais externos/Páginas;
4. Lançar um Poptin [12] *popup*.

Uma funcionalidade interessante para análise e que é apresentada por esta solução é a de *click tracking*, ou seja, a possibilidade de monitorizar as cliques que são feitos nos *widget*, usando o Google Analytics. Também existe a possibilidade de mostrar diferentes canais para diferentes dispositivos (*mobile* e *desktop*), ou até mesmo de mostrar o agregador em apenas um dos dispositivos.

Figura 2.12: Chaty

2.3.4 Widg.io

Widg.io [13] faz parte de um grupo chamado Agendas Group, fundado em Janeiro de 2021 e com uma politica de plataforma "*no code*", tendo em conta uma audiência sem capacidade de programação e que, por isso, fazem com que os seus produtos sejam intuitivos e fáceis de utilizar.

Existem dois tipos de planos que pode ser escolhidos, sendo que o primeiro é relativo ao uso de apenas um *widget* em que existe o plano grátis e três planos pagos, diferenciando os pagos apenas no número de *views* do *widget*. No outro plano de múltiplos *widgets*, para além das diferenças descritas para os planos de um *widget*, existe também um diferencial no número de *widgets* disponíveis para os quatro planos presentes, sendo que neste não existe nenhum gratuito. Todos os planos pagos podem ser usados num número ilimitado de *websites* (mas tendo em conta que o número de *views* é acumulado), para além de terem o logótipo da Widg.io removido e terem suporte para a instalação. Por fim, os pacotes Lite (os mais baratos de cada tipo) têm apenas suporte via email, enquanto que os restantes pacotes pagos têm também suporte via *livechat*.

A Widg.io oferece produtos muito semelhantes ao Elfsight, contudo a configuração não é tão complexa. Tal como se pode observar na figura 2.13, até o próprio design do All-In-One-Chat é idêntico, tal como as próprias funcionalidades, porém a configuração e personalização do *widget* agregador é mais simples.

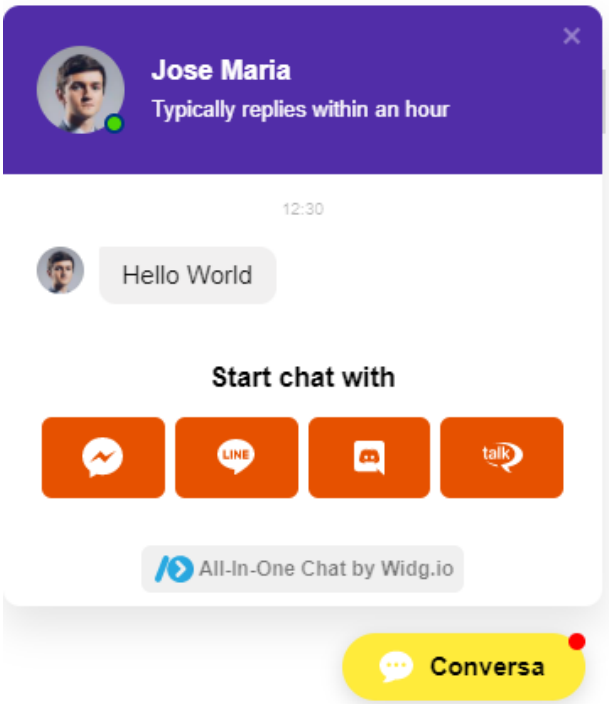


Figura 2.13: Widg.io

2.3.5 Análise comparativa

Tabela 2.1: Análise Comparativa

	Elfsight	Widg.io	MessageBird	Chaty
Regista Contactos			X	
Agrega diferentes tipos de canais				X
Sem restrições de uso	X	X	X	
Tracking das ações do utilizador				X
Existência de triggers	X		X	X

Como é possível observar na tabela 2.1, podemos verificar que a ferramenta da Widg.io é a que contempla menos dos pontos escolhidos para análise comparativa, tendo em conta os restantes agregadores. Começando pelo registo de contactos que é feito unicamente pelo MessageBird, mas que esta lista, tal como descrito na secção 2.3.2, tem um carácter somente ilustrativo.

De seguida, é possível verificar que apenas o Chaty tem a funcionalidade de agregar diferentes canais, isto é, para além de existirem os canais de *chat*, existem também canais que apresentam rotas ou localizações e canais de redirecionamento de links (estes links sendo customizáveis), o que tendo em conta este tema, trás maior versatilidade ao produto.

Por conseguinte, é de notar apenas o Chaty tem restrições de uso, visto que este é um *plugin* de WordPress e que, por isso, não será possível a sua integração em qualquer *website*. Já os restantes, usam todas a mesma estratégia de o cliente adicionar um pequeno excerto de código ao código fonte da página e dessa forma é permitida a injeção do *widget* na página.

É possível também observar que o *plugin* Chaty é o único que apresenta *tracking* das ações do utilizador, por exemplo, quantos *widgets* foram carregados nas páginas dos utilizadores ou então quantos utilizadores clicaram no *widget* agregador, etc. Estas métricas ajudam o gerente do *website* a perceber certas variáveis que podem ser importantes saber (p.e. percentagem de utilizadores que clicaram num certo *widget*, ou então, percentagem de utilizadores que clicaram mas que não interagiram posteriormente, etc).

Por último, verificou-se que a existência de *triggers* é linea em todos os produtos excepto no Widg.io. Estes *triggers* variam em complexidade, desde o simples "Mostrar após n segundo" de permanência no *website*, como existe no Elfsight e no Chaty, até complexos fluxos como o MessageBird disponibiliza (p.e. *Inbox Flow*).

Capítulo 3

Análise

Neste capítulo é realizada uma análise de valor ao projeto, tentando perceber se este se trata de um projeto válido e qual a melhor solução para o implementar.Para além disso, é realizada uma análise ao projeto em mãos, apresentando os requisitos que devem ser cumpridos para garantir o sucesso do projeto.

3.1 Análise de valor

Com o desenvolvimento de um novo produto, é necessário fazer uma análise e avaliação que vise a comprovar a viabilidade de um projeto, sendo que o maior foco é a qualidade do mesmo, mas também tendo em consideração se o custo/benefício da solução faz sentido para o caso em especifico [14].

3.1.1 Processo de Inovação

Com a constante evolução no mundo tecnológico, torna-se essencial inovar constantemente produtos e/ou processos que criem valor para uma organização e respetivos clientes.

Inovação é mais do que simplesmente ter boas ideias e obter vantagem estratégica, é o processo de transformá-las para que possam ter uso prático. Empresas inovadoras, tipicamente, conseguem um crescimento mais forte e são mais bem sucedidas do que aquelas que não inovam. Posto isto, não é grande surpresa que a inovação seja uma das principais características associadas ao sucesso [15].

Como referido no livro *The PDMA Toolbook 3 for New Product Development* [16], o processo de inovação é um processo segue o fluxo desde a criação à comercialização de um certo produto.

Este é, normalmente, divido em três partes (ilustradas na figura 3.1):

- *Discovery* (Fuzzy Front End (FFE)/Front End Innovation (FEI));
- *Development*;
- *Commercialization*.

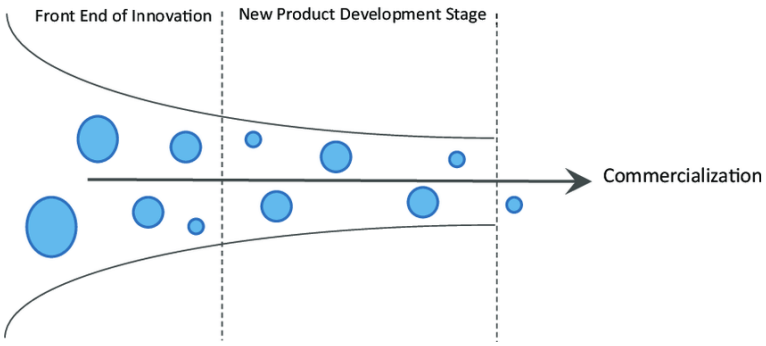


Figura 3.1: Fluxo New Product Development (NPD) [17]

A fase de *Discovery* tem como objetivo permitir às equipas de NPD gerarem e avaliarem várias oportunidades de ideias para novos produtos, sem que exista uma grande quantidade de recursos investidos nessas ideias. Para além disso, problemas técnicos e não técnicos podem ser abordados com antecedência, o que permite evitar atrasos e problemas mais significativos posteriormente.

Na segunda fase, o *Development*, envolve a passagem dos requisitos e especificações do produto para um design final, que é então convertido num produto pronto para comercialização. Esta fase também envolve o teste do produto, tanto internamente quanto com os clientes.

Por fim segue-se a fase de *Comercialization*, cujo objetivo é formular e executar o lançamento do produto final [18].

3.1.2 New Concept Development Model

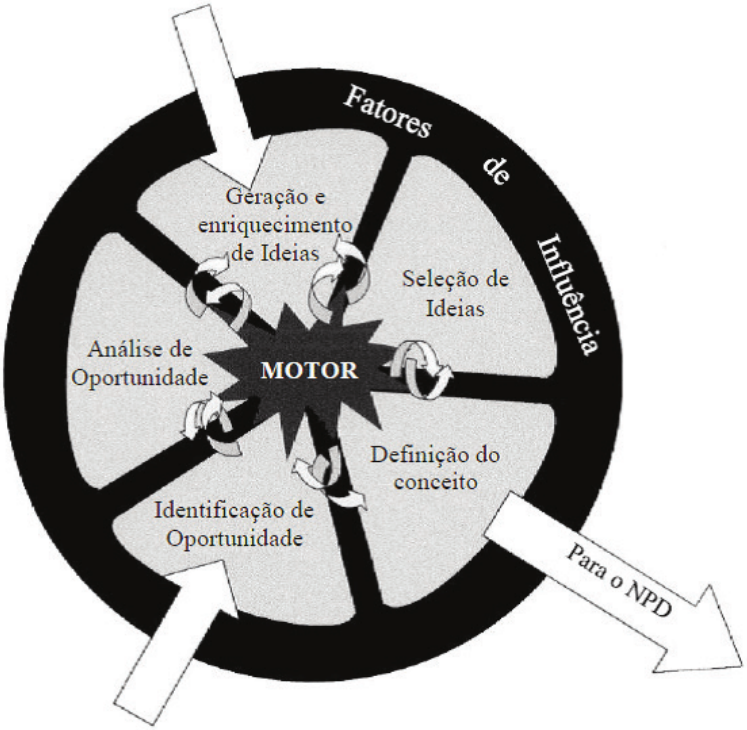


Figura 3.2: Modelo New Concept Development (NCD) [19]

Este modelo, ilustrado na figura 3.2, divide o FEI em três elementos principais:

O motor (*Engine*): Representa o centro do modelo é constitui a liderança, cultura e estratégias de negócio de uma organização. O número de inovações bem-sucedidas depende do quão bom é o motor.

A roda (*Wheel*): Representa os cinco elementos de atividades FEI, como a identificação da oportunidade, análise da oportunidade, geração de ideias, seleção de ideias e definição do conceito. O modelo NCD tem uma forma circular para dar a entender que as ideias podem circular entre os cinco elementos. Um projeto pode começar ou com a identificação da oportunidade, ou com a geração de ideias e logo que o conceito seja definido é possível passar para o processo de desenvolvimento.

O aro (*Rim*): Representa os fatores influenciadores, que neste caso referem-se a fatores externos que influenciam o *Engine* e os cinco elementos (*Wheel*) do modelo NCD, tais como clientes, concorrentes, fornecedores, regulamentos governamentais, tecnologia, entre outros [20].

Nesta próxima fase serão apresentados, de uma forma sistemática e bem definida, os cinco elementos de atividades FEI, sendo assim possível entender se a oportunidade traz valor à organização.

Identificação da oportunidade

A identificação de oportunidades pode ocorrer, principalmente, de duas maneiras, sendo que a primeira é a inexistência de um produto no mercado e que através de estudos (p.e.

estudo de mercado) se conclua que a ideia inicial é realmente uma oportunidade que deve ser aproveitada. No outro caso, pode ser a implementação/alteração de funcionalidades em produtos existentes. Estas alterações visam em colmatar, para além das fraquezas do produto (fator interno), as possíveis ameaças (fator externo) que possam existir.

A partir dos problemas identificados na secção 1.3, podem ser exploradas oportunidades para melhorar o produto E-goi, neste caso, com a implementação de um *widget* agregador de canais. Apesar de ser uma oportunidade válida, esta pode não acrescentar valor ao produto e, consequentemente, à empresa. Por isso, é necessário analisar a proposta estabelecida e tirar conclusões quanto à mesma.

Análise da Oportunidade

Numa análise de oportunidade é necessário entender se uma oportunidade tem o que é preciso para que se desenvolva em algo mais que uma ideia. Para traduzir a identificação de oportunidades em negócios específicos e oportunidades de tecnologia são necessárias informações adicionais, o que envolve fazer avaliações precoces e frequentemente de tecnologia e mercado que são tidas como incertas [21].

Para auxiliar neste processo, é necessário identificar os pontos fortes e os pontos fracos da oportunidade e, por conseguinte, é necessário tirar conclusões relativamente ao valor que será entregue ao produto e se o custo/beneficio compensa o avanço. Neste caso em especifico, será utilizada a análise SWOT.

A análise SWOT (abreviação para *strengths*, *weaknesses*, *opportunities*, *threads*) é uma ferramenta de estratégia de negócios que permite avaliar, neste caso, se uma ideia é viável de acordo com a SWOT feita. Existem considerações a ter tanto para ambientes internos e externos na análise do projeto.

"Forças"e "fraquezas"estão relacionadas internamente, isto é, são fatores que apenas estão dependentes da organização e em que existe controlo sobre os mesmos. A primeira representa as razões da ideia adicionar valor à empresa/produto, sendo que as fraquezas representam os pontos negativos da implementação do projeto.

Em relação às externas, ou seja, fatores que não podem ser controlados pela organização, temos as “oportunidades” que representam os fatores que podem, eventualmente, servir em benefício da entidade. Por outro lado, “ameaças” representam situações que podem gerar problemas para a entidade. (ref teoli2021dac)

Figura 3.3: Análise SWOT

Como é possível observar na figura 3.3, nas forças temos que o produto encontra-se integrado numa plataforma de marketing digital o que, querendo ou não, é uma grande força para o projeto visto que ter uma grande plataforma onde este projeto será integrado torna o produto mais robusto e adiciona valor não só a este projeto, como também ao próprio produto E-goi.

Para além disso, este agregador como contém produtos E-goi, que na sua essência representam canais com funções distintas, o que torna a escolha de *widgets* mais diversificada e não tão focada em apenas um tipo de canal (ao contrário dos All-In-One Chat, que focam em canais de comunicação ligados ao apoio ao cliente).

Como última força e através do uso do Matomo, como referido na secção 2.1.4, é possível saber métricas relativas ao uso do One Widget, o que trás um grande valor ao produto pois é possível ter um maior controlo sobre o One Widget, algo que os concorrentes destacados no capítulo 2.3 não possuem.

Para as fraquezas, tem-se que apesar de existir uma diversidade de canais com diferentes funções, não existe grande variedade de canais com a mesmo função, por exemplo, imaginemos que um One Widget tem um Whatsapp *leads* (referido na secção 2.2.2), mas como o subscritor não tem Whatsapp, provavelmente irá evitar o uso, que no final do contas não é esse o objetivo pretendido.

Posto isso, se existissem mais canais do tipo (p.e. Messenger Leads, Telegram Leads, etc) haveria maior possibilidade de escolha por parte do subscritor e, consequentemente, o aumento das chances de interação com o produto.

No caso das oportunidades, identificou-se que uma delas seria a facilidade na expansão de futuros canais E-goi, visto que a solução atual já terá a lógica da integração de produtos E-goi, o que torna a integração de futuros produtos E-goi mais fácil.

A outra oportunidade seria que os canais disponibilizados no One Widget teriam um aumento de utilização por parte dos clientes, visto que posteriormente à existência do One Widget era apenas possível utilizar um número limitado de canais sob a forma de *widget*, problema este que é colmatado com esta solução tecnológica.

No que diz respeito às ameaças, temos que a alteração do software externo utilizado para este projeto (p.e. Matomo) que pode afetar o comportamento do One Widget e com isto levar a que tenham de ser feitas alterações ao que foi desenvolvido inicialmente. Outra ameaça que pode também afetar o fluxo é a alteração da legislação, neste caso mais focado na legislação para a proteção de dados, pois como este trabalho envolve dados de utilizadores é necessário estar constantemente atento, de modo a intervir atempadamente (p.e. atualizar a produto para que esteja de acordo com a lei).

Por fim e tal como qualquer em qualquer negócio, é essencial ter um produto/serviço que possa competir com os demais. É importante perceber as ameaças relacionadas com a concorrência de modo a melhorar constantemente o produto para que os utilizadores do E-goi não sintam a necessidade de abandonar a mesma e optar por outras soluções.

Concluindo, foi possível perceber, através da análise SWOT, que esta oportunidade pode trazer valor à empresa, ao fornecer mais uma ferramenta que tem propósito e utilidade. Terá ainda um papel importante para aqueles clientes que necessitam de utilizar múltiplos canais na sua página. Posto isto, o custo de desenvolvimento compensará se tivermos em conta as vantagens que esta ferramenta trará e por isso será necessário, agora, proceder para a geração de ideias e definir a que melhor se adequa à situação.

Geração de ideias

A geração de ideias diz respeito à criação, desenvolvimento e amadurecimento de uma ideia concreta. A geração de ideias pode ser descrita como um processo evolutivo e iterativo onde as ideias são construídas, destruídas, combinadas, reformuladas, modificadas, etc [21]. Estas ideias podem passar por diversas iterações e mudanças, conforme esta passe pelos outros quatro elementos da Roda (*Wheel*) do modelo NCD.

Após a análise da oportunidade e da análise de soluções já existentes (encontradas na seção 2.3), foram discutidas possíveis soluções que estivessem em concordância com as análises feitas anteriormente e a partir daí, sugiram as seguintes abordagens:

- Agregador de canais feitos de raíz.
- Agregador de canais E-goi e integrações parceiras.
- Agregador de apenas canais E-goi.

Apesar destas abordagens serem diferentes, todas elas resolvem os problemas identificados, sendo que é necessário decidir por qual destas soluções optar. Para isso é necessário proceder à seleção da ideia, considerando diferentes critérios.

Seleção de ideias

Quase nunca faltam boas ideias e o verdadeiro problema é quais ideias perseguir a fim de alcançar o maior valor para o negócio. Infelizmente, não há processo único que garantirá uma boa seleção. Frequentemente, a seleção de ideias é baseada em julgamento, emoções ou "instinto" do avaliador. Para além disso, proceder a uma análise financeira e estimativas de receitas futuras, nesta fase inicial, é tipicamente má ideia. [21]

No entanto, é necessário utilizar uma método que ajude no auxilio da seleção de ideia. Neste caso o método escolhido foi o Analytic Hierarchy Process (AHP).

Analytic Hierarchy Process

O AHP foi desenvolvido na década de 1970 por Thomas L. Saaty e foi extensivamente estudado a partir dessa época. Nos dias de hoje, este método é utilizado para tomar decisões em diversos cenários complexos, em que pessoas colaboram de modo a encontrar a melhor solução e onde percepções humanas e julgamentos possuem repercussão de longo prazo [22]. Abaixo encontram-se as fases definidas para o modelo.

Fase 1: Construção da árvore hierárquica de decisão

Esta árvore encontra-se estruturada em níveis hierárquicos, o que permite uma melhor análise e compreensão do mesmo. Para isso, temos o objetivo principal no topo, sendo este ramificado em diferentes critérios que serão usados na avaliação de cada solução, estas que se encontram no fundo. A figura 3.4 ilustra a estrutura hierárquica para este projeto.

Para atingir o objetivo de desenvolver um agregador de canais é necessário avaliar as soluções referidas na secção 3.1.2 seguindo quatro critérios, sendo estes as funcionalidades existentes (neste caso, os canais de cada solução), a facilidade de integração na plataforma E-goi, a usabilidade de cada uma das soluções propostas e o seu custo de desenvolvimento.

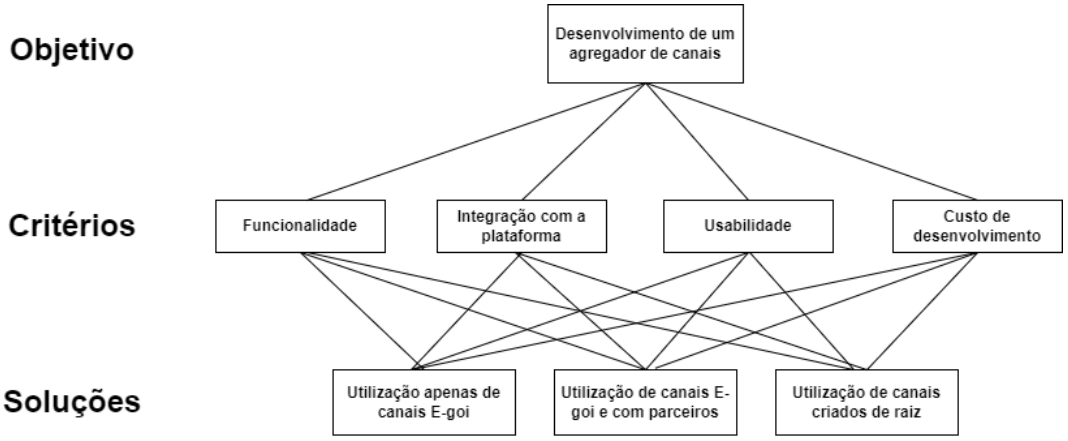


Figura 3.4: Estrutura Hierárquica do método AHP [23]

Fase 2: Prioridade relativa dos critérios

Definida a estrutura hierárquica do AHP, é agora necessário comparar cada critério, par a par, sendo necessário a elaboração de uma matriz de comparação utilizando a escala fundamental de Saaty [24], que se encontra no apêndice A.1.

Depois de construir a estrutura hierárquica, cada critério deve fazer uma comparação, par a par, criando-se uma matriz de decisão quadrada. Nessa matriz, o critério representará, a partir de uma escala predefinida, a sua preferência entre os elementos comparados. A escala predefinida que será utilizada para comparar os critérios par a par, será a Escala Fundamental de Saaty [24], que varia de 1 a 9, tal como demonstrado no apêndice A.1.

Dados os seguintes critérios:

- A - Funcionalidade;
- B - Integração com o E-goi;
- C - Usabilidade;
- D - Custo de desenvolvimento.

	A	B	C	D
A	1	3	4	7
B	1/3	1	2	5
C	1/4	1/2	1	3
D	1/7	1/5	1/3	1

Tabela 3.1: Matriz de comparação dos critérios

Tal como ilustrada na matriz 3.1, os critérios foram comparados em pares e dessa forma foi possível obter a matriz de comparação. A próxima etapa é normalizar a matriz e calcular a prioridade relativa, que se faz através do cálculo da média de todos os valores das comparações de um critério com os demais.

	A	B	C	D	Prioridade relativa
A	84/145	30/47	6/11	7/16	0.55
B	28/145	10/47	3/11	5/16	0.25
C	21/145	5/47	3/22	3/16	0.14
D	12/145	2/47	1/22	1/16	0.06

Tabela 3.2: Matriz normalizada de prioridade relativa

Na tabela 3.2 é possível verificar que, que a prioridade e importância das funcionalidades (canais de cada solução) são o critério mais importante, o que faz sentido pois todo o projeto envolve os próprios canais. Já o custo de desenvolvimento foi tido como o critério menos importante o que, novamente, faz sentido pois neste projeto o custo não é um fator muito importante pois não acarreta custos extra para a empresa, sendo a maior problemática o tempo que levará a desenvolver.

Fase 3: Avaliar a consistência das prioridades relativas

A próxima etapa consiste em calcular a Razão de Consistência (RC) para medir o quanto os julgamentos do avaliador foram consistentes em relação a grandes amostras de juízos, estas que são completamente aleatórias. Primeiramente é necessário a valor de λ_{max}.

Considerando a equação:

$$Ax = \lambda_{max} \tag{3.1}$$

onde A representa a matriz comparativa, x o vetor das prioridades relativa e $n = 4$, então:

$$\begin{bmatrix} 1 & 3 & 4 & 7 \\ 1/3 & 1 & 2 & 5 \\ 1/4 & 1/2 & 1 & 3 \\ 1/7 & 1/5 & 1/3 & 1 \end{bmatrix} \begin{bmatrix} 0.55 \\ 0.25 \\ 0.14 \\ 0.06 \end{bmatrix} \cong \lambda_{max} \begin{bmatrix} 0.55 \\ 0.25 \\ 0.14 \\ 0.06 \end{bmatrix} \iff \begin{bmatrix} 2.28 \\ 1.01 \\ 0.59 \\ 0.24 \end{bmatrix} \cong \lambda_{max} \begin{bmatrix} 0.55 \\ 0.25 \\ 0.14 \\ 0.06 \end{bmatrix} \iff$$

$$\iff \lambda_{max} = mean\{(2.28/0.55), (1.01/0.25), (0.59/0.14), (0.24/0.06)\} \approx 4.1$$

Figura 3.5: Cálculo do λ_{max}

Com a valor de λ_{max} calculado é possível, então, fazer a cálculo do Índice de Consistência (IC).

Dada a expressão:

$$[H]IC = \frac{\lambda_{max} - n}{n - 1} \tag{3.2}$$

E sabendo que $\lambda_{max} \approx 4.1$ e $n = 4$, temos:

$$[H]IC = \frac{4.1 - 4}{4 - 1} \approx 0.03 \tag{3.3}$$

Para finalizar o cálculo do RC será necessária a seguinte formula:

$$[H]RC = \frac{IC}{IR} \tag{3.4}$$

De acordo com a tabela disponível no anexo A.2, para 4 critérios o valor de IR deve ser 0.90. Para além disso, sabemos que IC é igual a 0.03, então:

$$[H]RC = \frac{0.03}{0.90} \approx 0.03 \tag{3.5}$$

Uma vez que $0.03 < 0.1$ conclui-se que os critérios estão consistentes e que o processo pode continuar para a próxima fase.

Fase 4: Construção da matriz de comparação paritária

Dada a consistência dos critérios na fase anterior, é necessário avaliar as soluções propostas em relação a cada critério.

Sendo assim, considera-se as seguintes soluções:

- A1 - Agregador apenas de canais E-goi;
- B1 - Agregador de canais E-goi e de parceiros;
- C1 - Agregador de canais criados de raiz;

	A1	B1	C1	Prioridade relativa
A1	1	1/3	1/2	0.17
B1	3	1	2	0.50
C1	2	1/2	1	0.33

Tabela 3.3: Tabela relativa ao critério "Funcionalidade"

	A1	B1	C1	Prioridade relativa
A1	1	4	3	0.62
B1	1/4	1	1/2	0.24
C1	1/3	2	1	0.14

Tabela 3.4: Tabela relativa ao critério "Integração com a plataforma"

	A1	B1	C1	Prioridade relativa
A1	1	3	2	0.54
B1	1/3	1	1/2	0.16
C1	1/2	2	1	0.30

Tabela 3.5: Tabela relativa ao critério "Usabilidade"

	A1	B1	C1	Prioridade relativa
A1	1	4	7	0.54
B1	1/4	1	5	0.25
C1	1/7	1/5	1	0.07

Tabela 3.6: Tabela relativa ao critério "Custo de desenvolvimento"

A partir da análise das tabelas de comparação paritária, é possível verificar que em relação ao critério da funcionalidade, a Solução B1 é claramente mais prioritária. Em relação a todos os outros critérios, a solução A1 é a dominante.

Fase 5: Escolha da alternativa

Por fim, na escolha da alternativa, apenas é necessário fazer a multiplicação da matriz das prioridades das soluções em relação a cada um dos critérios com o vetor das prioridades dos critérios.

A solução pode ser calculada da seguinte forma:

$$\begin{bmatrix} 0.17 & 0.62 & 0.54 & 0.54 \\ 0.50 & 0.24 & 0.16 & 0.25 \\ 0.33 & 0.14 & 0.30 & 0.07 \end{bmatrix} \begin{bmatrix} 0.55 \\ 0.25 \\ 0.14 \\ 0.06 \end{bmatrix} = \begin{bmatrix} 0.36 \\ 0.38 \\ 0.26 \end{bmatrix}$$

Figura 3.6: Cálculo das prioridade

Com este último cálculo, as prioridades das soluções ficam as seguintes:

	Prioridade
A1	0.36
B1	0.38
C1	0.28

Tabela 3.7: Tabela relativa à prioridade final de cada solução

Tendo agora os resultados da tabela 3.7 é possível concluir que a opção ideal para avançar é a B1 (**Agregador de canais E-goi e de parceiros**).

Definição do conceito

Resta agora o último elemento da roda (*Wheel*) do modelo NCD, a definição do conceito.

O conceito para o One Widget é de uma agregador de canais E-goi e integração com canais de organizações parceiras. A interface necessita de ser simples e intuitiva, sendo que a requisição de informação necessária, para a criação de um One Widget com quaisquer canais, tem de ser pedida de uma forma a que qualquer utilizador consiga perceber. Com isto, é essencial colmatar os problemas identificados inicialmente na secção 1.3.

3.1.3 Valor da solução

De modo perceber o valor da solução, é preciso entender a importância de um produto/serviço para o cliente. Para isso, é importante definir os conceitos de valor do produto e valor percecionado.

Valor do produto, Valor Percecionado

Valor do produto está relacionado com o preço do produto e é influenciado pela sua qualidade. O valor do produto aumenta de forma diretamente proporcional às suas vantagens, em relação à concorrência, e diminui de maneira igual em relação às desvantagens.

O valor percecionado é o beneficio derivado do próprio produto e é medido através do quanto um cliente está disposto a pagar (por exemplo, valor percecionado = benefícios percecionados, sendo que estes benefícios são comparados com os produtos concorrentes). Este valor é influenciado pelas necessidades, expectativas, experiências passadas, entre outros, pelo que é um valor que difere de pessoa para pessoa [25].

3.1.4 Business Model Canvas

O Business Model Canvas [26] é uma ferramenta de gestão estratégica e que permite que um negócio seja completamente visualizado em uma única página.

Isso é feito por meio de um quadro, dividido em nove componentes que cobrem as quatro áreas consideradas as principais de um negócio, que são clientes, oferta, infraestrutura e viabilidade financeira.

Este modelo está ilustrado na Figura 3.7.

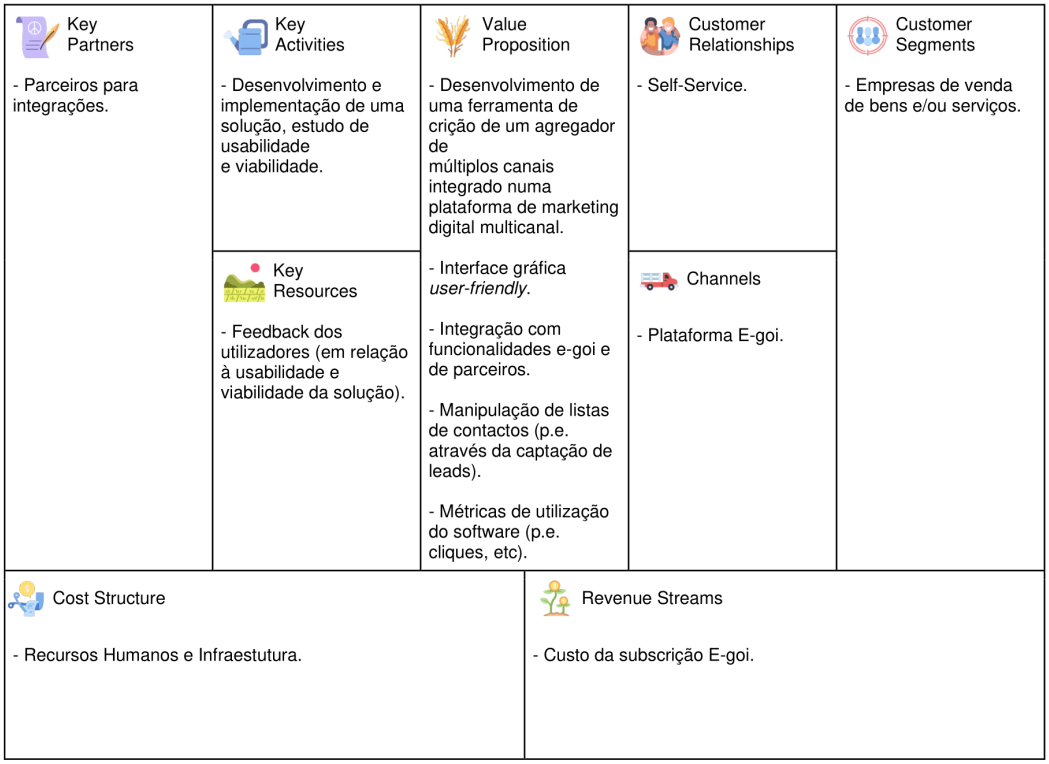


Figura 3.7: Business Model Canvas

3.1.5 Desenvolver a pensar no cliente

Um produto com qualidade visa atender às necessidades do cliente. Este, portanto, pode ser considerado como um produto ergonómico - um produto adaptado às capacidades e limitações humanas.

Na área de tecnologia de qualidade, uma série de métodos foram desenvolvidos para tentar simplificar e tornar o desenvolvimento do produto mais eficiente. O método Quality Function Deployment (QFD) é um método bem conhecido e sistemático que se baseia na ideia de adaptar a tecnologia às pessoas [27].

No âmbito do projeto atual, decidiu-se elaborar um QFD), para que seja possível estabelecer uma relação entre o que o cliente necessita e as características de qualidade do produto a ser elaborado, priorizando estas características.

Quality Function Deployment

Existem 4 níveis para este processo, sendo que será apenas abordado o primeiro. Numa primeira fase, são definidas as necessidades do cliente, traduzindo-as em características de qualidade do produto em análise.

Depois de uma discussão com elementos da empresa, chegou-se a uma conclusão que as principais necessidades do cliente seriam:

- **Fácil acesso:** Cliente necessita de ter o acesso facilitado ao recurso, utilizando o menor número de cliques possíveis;
- **Boa performance:** O sistema tem de apresentar boa performance pois só assim, o cliente consegue ter uma boa experiência de uso;
- **Funcionalidades relevantes:** As funcionalidades têm de ser relevantes para o cliente;
- **Interface intuitiva:** Para um cliente uma interface tem de ser fácil de usar e não demorar muitas vezes até que este saiba o percurso todo que tem de fazer para realizar uma certa ação.

Como estas características não tem todas a mesma importância e prioridade, é necessário fazer uma estimativa da importância de cada uma. Para isso, as necessidades são avaliadas de 1 a 5 e no fim dessa avaliação, é possível calcular a percentagem de importância de cada necessidade. Esses valores encontram-se junto às necessidades do cliente na figura 3.8.

De seguida é necessário definir o que será comparado com as necessidades do clientes, que neste caso, são conceitos considerados importantes para este projeto tais como:

- **Usabilidade**;
- **Tempo de processamento**;
- **Criar um One Widget**;
- **Associar um domínio**;
- **Relatório de métricas**;
- **Estabilidade**.

Estes conceitos devem ser avaliados de acordo com as necessidades, sendo que os critérios e os próprios conceitos estão ilustrados na figura 3.8. Estas relações entre conceitos podem apenas ser avaliados com 0, 1, 3 e 9, sendo que 9 representa que uma necessidade tem uma forte relação com o conceito.

Agora, falta definir os concorrentes e avaliá-los de acordo as necessidades dos clientes. São também avaliados de 1 a 5, sendo que 5 representa que esse produto aborda a necessidade de maneira exemplar. Por último, é necessário o calcular o rácio de importância de cada conceito e com isso, é possível finalizar com o cálculo da importância percentual de cada conceito.

Figura 3.8: House Of Quality (HOF)

De acordo Com o QFD desenvolvido e com a análise da figura 3.8, pode-se concluir que:

- Foi dado maior importância às funcionalidades relevantes de que a todo o resto;
- Como seria de esperar a ação principal do projeto obteve a maior pontuação, sendo que o segundo mais prioritário é o tempo de processamento;
- O tempo de processamento e a estabilidade obtiveram correlação negativa, ou seja, quando um aumenta o outro diminui, proporcionalmente.

3.2 Engenharia de Requisitos

A aplicação da Engenharia de Requisitos é, geralmente, discutida como uma fase inicial do processo de desenvolvimento de software. No entanto, sabe-se que o conhecimento dos requisitos requer um esforço contínuo no refinamento progressivo das exigências contidas nas regras de negócio das organizações e em atendimento às vontades e ao querer dos *stakeholders*, durante todo o ciclo de vida do software [28].

Para a classificação dos requisitos será utilizado o Modelo FURPS+. Este acrónimo, corresponde a *Functionality*, *Usability*, *Reliability*, *Performance*,*Supportability* e que em português representa Funcionalidade, Usabilidade, Fiabilidade, Desempenho e Suporte. O "+" em FURPS+ serve para relembrar que também deve haver preocupações/restrições de design, implementação, interface e físicas [29]. Portanto, os requisitos funcionais representam

a Funcionalidade, enquanto que os não funcionais representam todos os outros conceitos referidos no modelo FURPS+.

3.2.1 Requisitos Funcionais

Os requisitos funcionais descrevem explicitamente as funcionalidades e serviços do Sistema, mas também como este deve reagir e se comportar em determinadas situações [30]. Para melhor contextualização dos requisitos funcionais, este encontram descritos abaixo como User Stories.

RF - 01: Criar um One Widget

Descrição:

Como Utilizador (cliente), devo poder criar um One Widget.

Prioridade:

Alta.

Pré-Condições:

- Utilizador deve estar autenticado na plataforma (E-goi).

Fluxo de Eventos:

1. O Utilizador inicia a ação de criação do One Widget;
2. O Sistema apresenta a página de opções e requer uma lista de contactos, idioma e nome interno para o One Widget;
3. O Utilizador define essa informação;
4. O Sistema apresenta a página de edição requer canais (Obrigatório) e opções de customização (Não obrigatório, visto que as opções de customização, como posição, cor e ícone, entre outros, já têm valores por defeito);
5. O Utilizador define os canais (juntamente com a informação indispensável para cada canal, se necessário);
6. O Sistema apresenta a página de resumo, que contém todas as configurações feitas neste.
7. O utilizador pode optar por definir já o domínio em que quer que o One Widget apareça (Não Obrigatório) e confirma (Obrigatório).
8. O Sistema cria o One Widget e apresenta uma mensagem de sucesso e redireciona para as listagens de One Widgets.

RF - 02: Editar os canais e customização de um One Widget

Descrição: Como Utilizador, devo poder editar os canais e customização de um One Widget.

Prioridade:

Alta.

Pré-Condições:

- Utilizador deve estar autenticado na plataforma;
- Utilizador deve ter criado pelo menos um One Widget.

Fluxo de Eventos:

1. O Utilizador inicia a ação de editar o One Widget;
2. O Sistema redireciona o Utilizador para a página de edição;
3. O Utilizador finaliza a edição dos canais e customização do One Widget;
4. O Sistema edita o One Widget e apresenta uma mensagem de sucesso e redireciona para as listagens de One Widgets.

RF - 03: Editar as opções de um One Widget

Descrição:

Como Utilizador, devo poder editar as opções de um One Widget.

Prioridade:

Alta.

Pré-Condições:

- Utilizador deve estar autenticado na plataforma;
- Utilizador deve ter criado pelo menos um One Widget.

Fluxo de Eventos:

1. O Utilizador inicia a ação de editar opções do One Widget;
2. O Sistema redireciona o Utilizador para a página de opções;
3. O Utilizador finaliza a edição das opções do One Widget;
4. O Sistema edita o One Widget e apresenta uma mensagem de sucesso e redireciona para as listagens de One Widgets.

RF - 04: Apagar um One Widget

Descrição:

Como Utilizador, devo poder apagar um One Widget.

Prioridade:

Alta.

Pré-Condições:

- Utilizador deve estar autenticado na plataforma;
- Utilizador deve ter criado pelo menos um One Widget.

Fluxo de Eventos:

1. O Utilizador inicia a ação de apagar o One Widget;

2. O Sistema apresenta uma mensagem de confirmação;
3. O Utilizador confirma;
4. O Sistema apresenta uma mensagem de sucesso.

RF - 05: Antever um One Widget

Descrição:

Como Utilizador, devo poder antever um One Widget.

Prioridade:

Média.

Pré-Condições:

- Utilizador deve estar autenticado na plataforma;
- Utilizador deve ter criado pelo menos um One Widget.

Fluxo de Eventos:

1. O Utilizador inicia a ação de antever o One Widget;
2. O Sistema mostra uma *preview* (utilizador pode escolher entre um certo número de dispositivos);

RF - 06: Ver um relatório de resultados de um One Widget

Descrição:

Como Utilizador, devo poder ter acesso a um relatório de resultados do One Widget.

Prioridade:

Alta.

Pré-Condições:

- Utilizador deve estar autenticado na plataforma;
- Utilizador deve ter criado pelo menos um One Widget.

Fluxo de Eventos:

1. O Utilizador inicia a ação de resultados do One Widget;
2. O Sistema mostra o relatório de métricas do One Widget;

RF - 07: Descarregar o relatório de resultados de um One Widget

Descrição:

Como Utilizador, devo poder ter descarregar o relatório de resultados do One Widget.

Prioridade:

Baixa.

Pré-Condições:

- Utilizador deve estar autenticado na plataforma;
- Utilizador deve ter criado pelo menos um One Widget.

Fluxo de Eventos:

1. O Utilizador inicia a ação de resultados do One Widget;
2. O Sistema mostra o relatório de métricas do One Widget;
3. O Utilizador inicia a ação de descarregar o relatório de resultados do One Widget;
4. O Sistema cria o ficheiro pdf e este é descarregado;

RF - 08: Permitir utilizar canais E-goi já criados

Descrição:

Como Utilizador, devo poder adicionar ao meu One Widget um canal E-goi criado previamente.

Prioridade:

Alta.

Pré-Condições:

- Utilizador deve estar autenticado na plataforma;
- Utilizador deve ter pelo menos um canal E-goi criado, compatível com o One Widget.

Fluxo de Eventos:

1. O Utilizador cria/edita um One Widget;
2. O Sistema mostra uma lista de canais E-goi já criados;
3. O Utilizador escolhe o canal;
4. O Sistema guarda essa informação;

RF - 09: Permitir criar canais E-goi na página de edição de um One Widget

Descrição:

Como Utilizador, devo poder adicionar ao meu One Widget um novo canal E-goi criado a partir da página de edição.

Prioridade:

Média.

Pré-Condições:

- Utilizador deve estar autenticado na plataforma.

Fluxo de Eventos:

1. O Utilizador cria/edita um One Widget e inicia a ação de criar um novo canal;
2. O Sistema mostra uma lista de canais E-goi disponíveis para criar;

3. O Utilizador escolhe o canal;
4. O Sistema abre um *iframe* com o editor desse canal;
5. O Utilizador finaliza o canal;
6. O Sistema guarda o canal na base de dados e associa-o a esse One Widget;

RF - 10: Adicionar o domínio na página de resumo do One Widget

Descrição:

Como Utilizador, devo poder adicionar o domínio do meu *website* na página de resumo do One Widget.

Prioridade:

Alta.

Pré-Condições:

- Utilizador deve estar autenticado na plataforma.

Fluxo de Eventos:

1. O Utilizador cria/edita um One Widget e inicia a ação de criar um novo canal;
2. O Sistema mostra uma lista de canais E-goi disponíveis para criar;
3. O Utilizador escolhe o canal;
4. O Sistema abre um *iframe* com o editor desse canal;
5. O Utilizador finaliza o canal;
6. O Sistema guarda o canal na base de dados e associa-o a esse One Widget;

RF - 11: Associar um One Widget ao domínio através do Connected Sites

Descrição:

Como Utilizador, devo poder ter associar um One Widget ao meu domínio, através do Connected Sites.

Prioridade:

Alta.

Pré-Condições:

- Utilizador deve estar autenticado na plataforma;
- Utilizador deve ter criado pelo menos um One Widget.

Fluxo de Eventos:

1. O Utilizador acede ao Connected Sites;
2. O Sistema mostra os domínios existentes para aquela conta;
3. O Utilizador escolhe a domínio que pretende associar;

4. O Sistema apresenta a lista de produtos E-goi que podem ser associados;
5. O Utilizador seleciona "One Widget"e escolhe qual dos One Widgets existentes quer associar;
6. O Sistema associa aquele One Widget ao domínio pretendido;

Os requisitos funcionais apresentados acima traduzem as funcionalidades que o Sistema deve apresentar para atingir o seu propósito. A tabela 3.8 contém todos os requisitos funcionais especificados previamente.

ID	Requisito Funcional
RF-01	Criar um One Widget
RF-02	Editar os canais e customização de um One Widget
RF-03	Editar as opções de um One Widget
RF-04	Apagar um One Widget
RF-05	Antever um One Widget
RF-06	Ver a relatório de resultados de um One Widget
RF-07	Descarregar o relatório de resultados
RF-08	Utilizar um produto E-goi já criado anteriormente
RF-09	Criar um produto E-goi na página de edição do One Widget
RF-10	Adicionar um domínio na página de resumo do One widget
RF-11	Associar um One Widget ao domínio através do Connected Sites

Tabela 3.8: Tabela de Requisitos Funcionais

Com os requisitos funcionais definidos, estes podem ser esquematizados num diagrama de casos de uso, como ilustrado na 3.9.

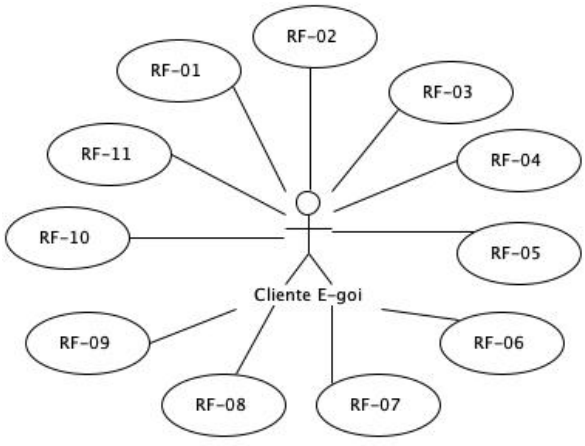


Figura 3.9: Diagrama de Casos de Uso

3.2.2 Requisitos Não Funcionais

Seguindo a ordem do modelo FURPS+ e visto que a *Functionality* já foi analisada nos requisitos funcionais3.2, é agora necessário analisar os restantes conceitos, que definem

propriedades e restrições do Sistema. Estes conceitos enquadram-se nos requisitos não funcionais que são abordados nesta secção.

Usabilidade

A usabilidade trata de características como a aparência e consistência de uma interface gráfica [29]. Neste projeto foram identificados os seguintes requisitos de usabilidade:

- O Sistema deve garantir que exista responsividade e correto posicionamento, tantos dos *widgets*, como o conteúdo que estes exibem ao serem clicados;
- O Sistema deve garantir que o One Widget se adapte ao número de *widgets* que possuí, para que não existam desformatações.

Fiabilidade

A fiabilidade refere-se a características de Sistema como a disponibilidade (tempo de *uptime* do Sistema), precisão com cálculos do Sistema e a capacidade que o Sistema tem de se recuperar de falhas [29]. Foram constatados os seguintes requisitos relativos à fiabilidade:

- O Sistema deve fazer gravações automáticas periódicas;
- O Sistema deve suportar tolerância a falhas.

Desempenho

O Desempenho caracteriza-se pelo tempo de resposta, taxa de transferência, tempo de recuperação, tempo de inicialização, entre outros [29]. Os requisitos a destacar para esta vertente são:

- O Sistema deve ter um tempo de espera adequado na transição de página, no fluxo de criação;
- O Sistema deve ter um tempo aceitável entre a seleção do One Widget no Connected Sites e o aparecimento deste no *website* do cliente.
- Em caso de falha, o Sistema deve ter um tempo de recuperação aceitável.

Suporte

O Suporte é caracterizado pela adaptabilidade, manutenibilidade, compatibilidade, escalabilidade, entre outros [29]. No suporte, há que destacar:

- O Sistema deve ter capacidade de expansão, nomeadamente, permitir a adição de novos canais a qualquer momento.
- O Sistema deve ser suportado pela plataforma E-goi, visto que será um produto da mesma.
- O Sistema deve garantir que todos os canais provenientes de parcerias têm compatibilidade com todas as funcionalidades do One Widget.

Outros(+)

Relativamente a restrições de implementação, tem-se que:

- O Sistema deve ser desenvolvido usando a *framework* Angular, a linguagem de programação PHP e o Sistema Gestor de Base de Dados (SGBD) MariaDB;
- A comunicação entre a componente do One Widget e a base de dados deve ser feita através da API da E-goi - *Services*.

Capítulo 4

Desenho da Solução

O desenho da solução é uma etapa fundamental para planear o que irá ser implementado, com base no que foi apresentado na engenharia de requisitos.

Para tal, foi feita uma análise aos padrões e regras arquiteturais atuais do produto E-goi e posteriormente desenvolvidos os artefactos que definem o One Widget e que servirão de base para a implementação da solução.

4.1 Arquitetura do Sistema

Nesta secção é realizada a análise da abordagem arquitetural para o projeto, bem como, algumas alternativas para se apresentar um desenho arquitetural completo. A descrição da análise e desenho foi baseada no Modelo 4+1 [31].

A arquitetura de software lida com abstração, decomposição e composição, estilo e estética. Este modelo é composto por cinco vistas que permitem abordar separadamente as preocupações de várias partes interessadas da arquitetura. As cinco vistas são projetadas usando um processo de desenvolvimento iterativo centrado na arquitetura e orientado por cenário [31]:

- **Vista Lógica:** A vista lógica suporta os requisitos funcionais, ou seja, o que o sistema deve oferecer em termos de funcionalidades aos utilizadores. O sistema é decomposto num conjunto de abstrações, retiradas do domínio do problema, em forma de classes.
- **Vista de Implementação:** A vista de implementação foca-se na organização dos módulos do software no ambiente de desenvolvimento. O software é agrupado em pequenos módulos ou subsistemas.
- **Vista de Processos:** A vista de processos tem em consideração alguns requisitos não funcionais como o desempenho e a disponibilidade. Aborda problemas de concorrência e distribuição, integridade do sistema e as principais abstrações da vista lógica que se enquadram na arquitetura de processos.
- **Vista Física:** A vista física foca-se primeiramente nos requisitos não funcionais do sistema como a disponibilidade, confiabilidade, desempenho e escalabilidade. É nesta vista que são esquematizados os componentes físicos do sistema.
- **Vista de Cenários:** A vista de cenários descreve os casos de uso ou cenários que representam a quinta vista. Os cenários descrevem sequências de interações entre objetos e processos. Esta vista já se encontra representada na secção 3.2.1, mais concretamente na figura 3.9.

4.1.1 Vista lógica

O domínio de uma aplicação consiste na definição de todos os conceitos e entidades relacionados com o problema que se pretende resolver.

O primeiro passo para o desenvolvimento de um sistema de software é compreender, de forma clara, o contexto ou domínio em questão.

De forma a facilitar essa compreensão, foi elaborado um modelo do domínio, que ilustra as classes conceituais relacionadas com o problema e as respetivas relações e cardinalidades, que se encontra ilustrado na figura 4.1.

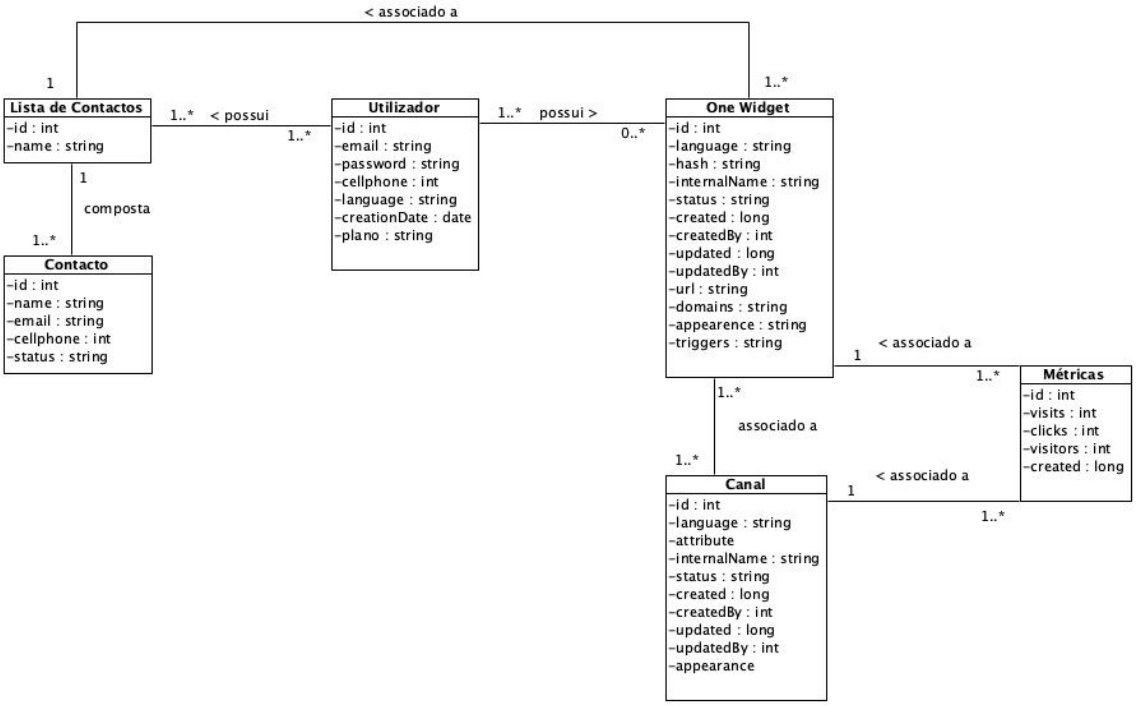


Figura 4.1: Diagrama do modelo de domínio

Neste diagrama, o utilizador representa um cliente E-goi que utiliza o sistema e tem a possibilidade de criar várias listas de contactos e também vários One Widgets.

O One Widget contém informações tais como o nome interno, o idioma, o aspeto gráfico, entre outros. Este tem sempre associado uma lista de contactos definida pelo utilizador e ter várias métricas, estas que são calculadas em específicos intervalos de tempo.

O contacto representa o visitante da página web de um utilizador, que ao submeter a sua informação é adicionado a uma lista de contactos. Um contacto pode pertencer a uma lista de contacto podendo este contacto interagir com os One Widgets disponibilizados previamente pelo utilizador.

O One Widget está ainda associado a vários canais, estes que também têm as suas métricas, que são definidas separadamente do One Widget.

4.1.2 Vista de implementação

A vista de implementação ilustra o sistema no ponto de vista do programador. Esta vista é possível ser representada através de um diagrama de componentes.

A E-goi já apresenta uma arquitetura baseada em componentes para o seu produto. Isto é o sistema é representado e dividido em componentes com o objetivo de reduzir a complexidade deste, permitindo que cada componente tenha a sua função e estas não se misturem. Deste modo, estas funcionalidades podem ser reutilizadas em diversas aplicações através do acesso ao componente responsável por essa função.

De acordo com John Cheesman [32], existem 3 princípios fundamentais de uma arquitetura que se baseia em componentes, e estes são:

- Unificação dos dados e da função: um componente de software consiste em valores dos dados e em funções que processam estes dados. Esta colocação natural das dependências entre os dados e a função melhora a coesão;
- Encapsulamento: um componente de software é isolado de outros componentes pelas suas funcionalidades, o que reduz as dependências e o acoplamento entre os diversos componentes do sistema;
- Identidade: cada componente possui uma identidade única.

Foi elaborado um diagrama de componentes, que representa a arquitetura do sistema, onde será implementada a solução que se encontra ilustrado na Figura 4.2.

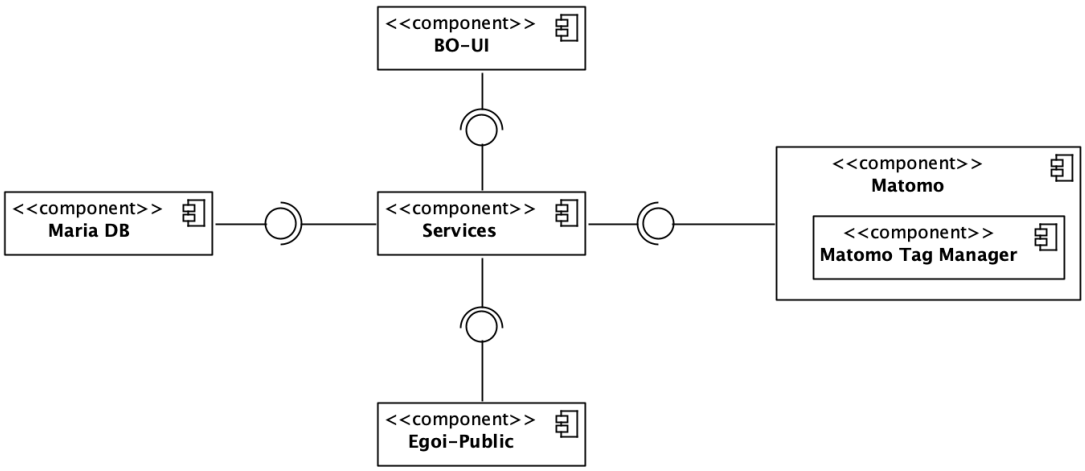


Figura 4.2: Diagrama de Componentes do sistema

Na Tabela 4.1 é possível consultar a legenda da Figura 4.2.

Tabela 4.1: Tabela de Componentes

Componentes	**Descrição**
Matomo	Componente que contem o Matomo Tag Manager
BO-UI	Componente responsavel pela interface grafica da plataforma do E-goi
Matomo TagManager	*Plugin* do Matomo que permite aos seus utilizadores incorporarem recursos de aplicacoes de terceiros no seu *website*
Services	API que comunica com todos os componentes desde a base de dados ate a interface grafica
Egoi-public	API pública responsavel por comunicar com o componente Services
Maria DB	Componente de persistencia de dados

Pela análise ao diagrama presente na Figura 4.2, é possível perceber que todos os componentes comunicam com o componente principal, o Services. Este componente é uma Representational state transfer (REST) Application Programming Interface (API) desenvolvida na linguagem de programação PHP [33], que utiliza pedidos Hyper Text Transfer Protocol Secure (HTTPS) para aceder, utilizar e manipular dados. O Services contém a maior parte da lógica de negócio do produto e de tratamento de dados. Para persistir dados, comunica com o componente MariaDB, desenvolvido utilizando a tecnologia MariaDB [34], um sistema de gestão de bases de dados relacionais, rápido, escalável e robusto, criado pelos desenvolvedores do MySql[35].

O componente do Matomo é uma API que contém o Matomo Tag Manager, um *plugin* integrado com a plataforma E-goi, que para este projeto será utilizado como meio de rastreamento do comportamento de um visitante de um determinado *website*, despoletando *triggers* consoante esse comportamento. O Services comunica com o Matomo sempre que o utilizador ativa, altera ou desativa o One Widget, enviando toda a informação necessária, desde tipo de *trigger*, condições de aparecimento e limitações de visualização, que será persistida na base de dados do Matomo.

O BO-UI é uma aplicação desenvolvida em AngularJS [36], que contém a lógica de grande parte da interface gráfica do produto E-goi. Este componente comunica com o Services para receber e enviar dados através de pedidos HTTP. Neste componente será desenvolvida a interface gráfica de gestão dos One Widgets.

O Egoi-public contém toda a lógica do que acontece quando um visitante abre uma página, desde registo de dados (cliques, visitas e visitas únicas), redirecionamento após clique em botões, entre outros.

Arquitetura alternativa

Uma possível alternativa à arquitetura escolhida como solução para o problema, seria desenvolver uma nova componente que fosse capaz de substituir o Matomo.

Seguindo a estrutura arquitetural do sistema, é possível construir um diagrama de componentes da alternativa, tal como é apresentado na Figura 4.3

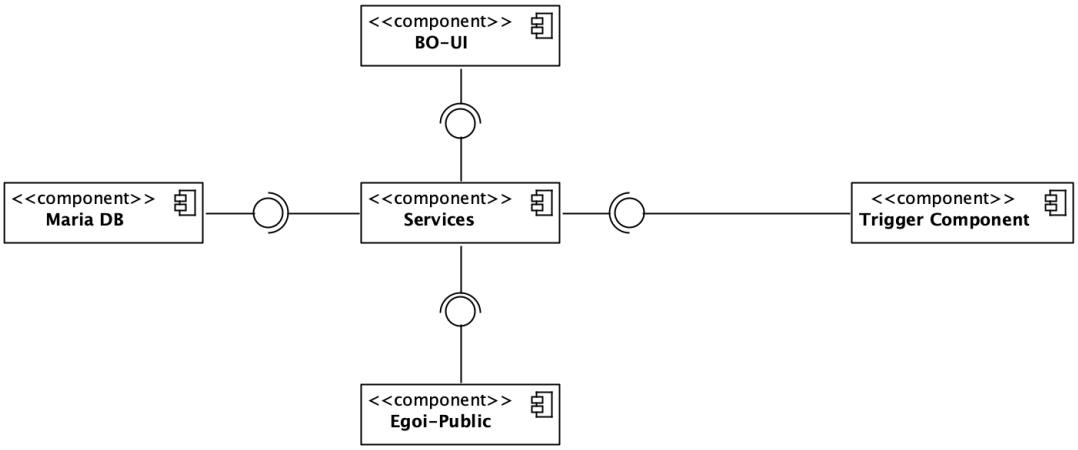


Figura 4.3: Diagrama de Componentes Alternativo do sistema

Visto que a E-goi já tem propriedade dos componentes anteriores, o único componente que faria sentido alterar, neste contexto, seria a integração com o Matomo que seria substituído por um componente feito de raiz com o mesmo propósito, que neste caso é um meio de rastreamento do comportamento de um visitante de um determinado *website*, despoletando *triggers* consoante esse comportamento.

Como já existem produtos E-goi integrados com o Matomo e esta ferramenta é bastante completa e viável, para além de satisfazer as necessidades atuais do produto e por isso não se justifica o desenvolvimento de um componente do zero, tanto no ponto de vista do projeto, como do próprio produto E-goi. A vantagem do uso desta arquitetura seria a possibilidade de um maior controlo de dados e facilidade de adicionar novas *features* para este componente, caso necessárias.

4.1.3 Vista Física

A vista física/implantação da arquitetura de um projeto de software diz respeito à distribuição da instalação dos componentes do software por máquinas que vão disponibilizar o sistema aos utilizadores. Para representar a implantação do sistema deste projeto, foi elaborado um diagrama de implantação, representado na Figura 4.4.

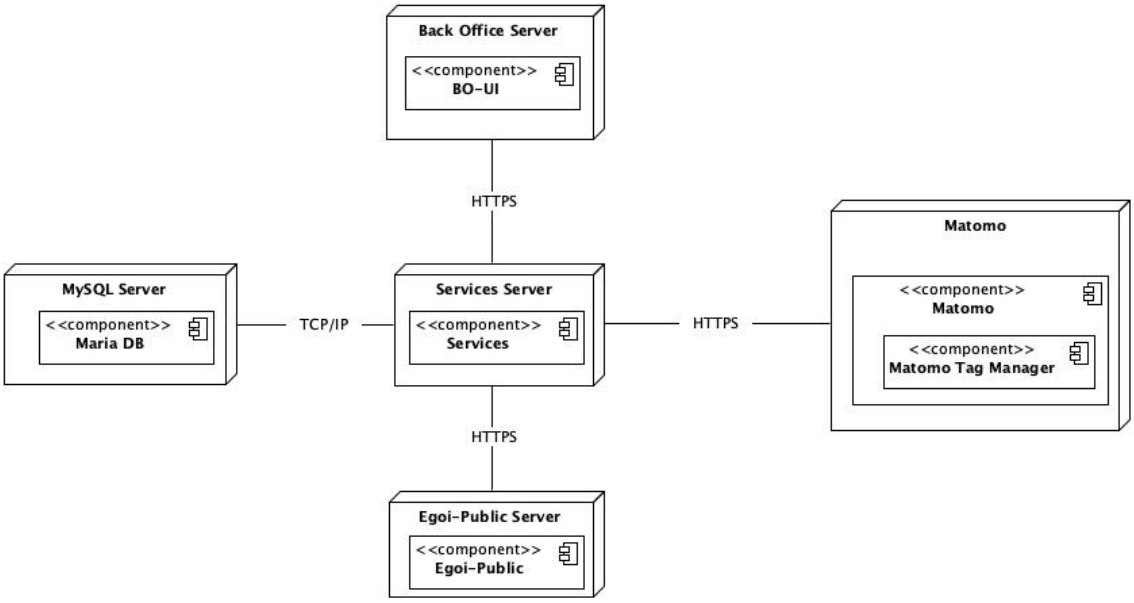


Figura 4.4: Diagrama de Implantação do sistema

Pela análise ao diagrama da Figura 4.4, é possível verificar que cada componente se encontra alojado na sua respetiva máquina servidor, por exemplo, a máquina Services está inteiramente dedicada ao componente Services. A comunicação feita entre a API interna do E-goi (Services) e a base de dados (MariaDB) utiliza o protocolo TCP/IP, enquanto que as restantes comunicações são feitas através do protocolo HTTPS.

4.1.4 Vista de processos

Nesta secção é apresentado o principal processo/fluxo de utilização deste projeto (criação de um One Widget) e também como os componentes interagem entre si. Para isso, foi desenhado um diagrama de sequência, que descreve como, e em qual ordem, os processos se executam.

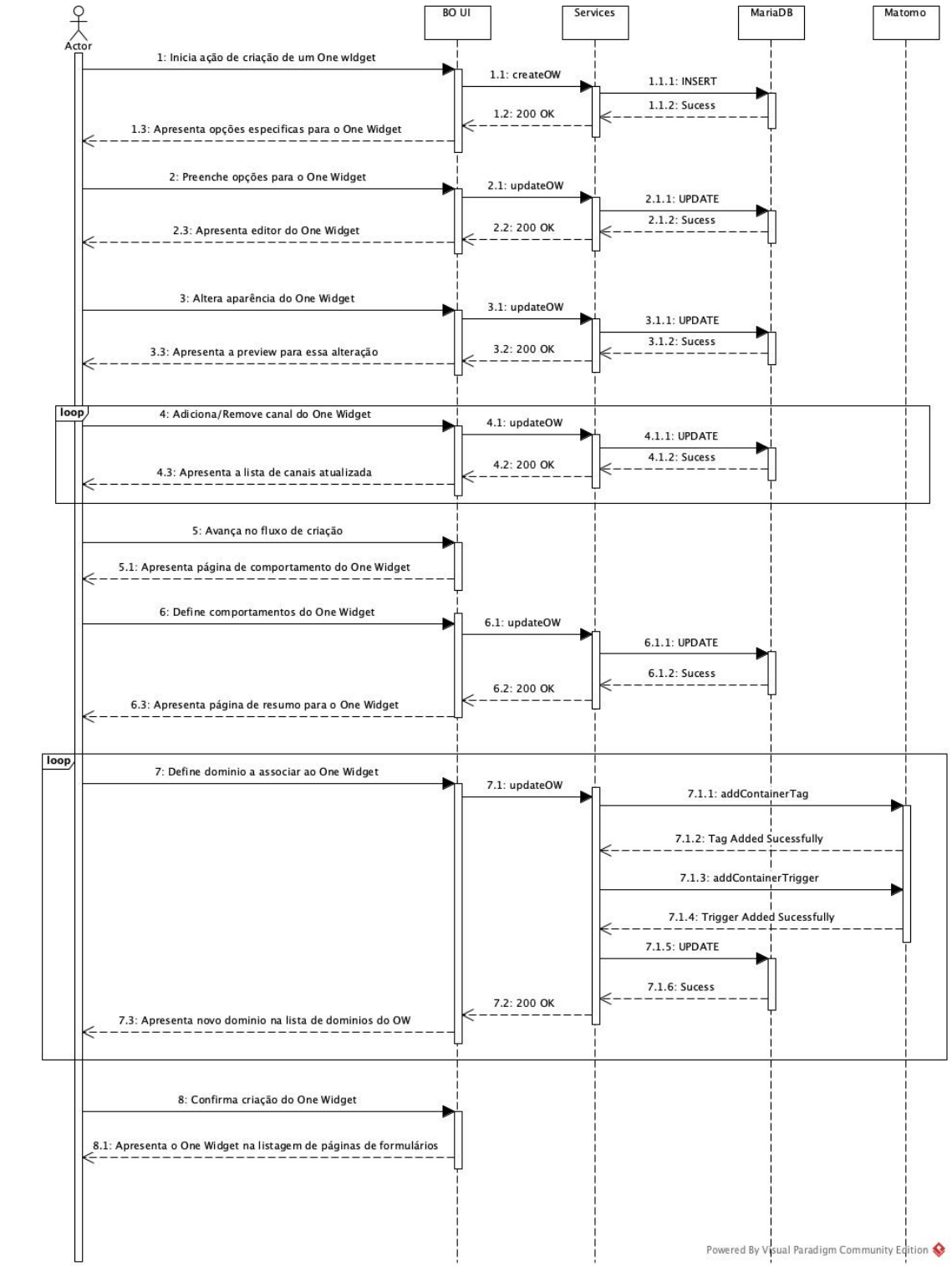


Figura 4.5: Diagrama de Sequência para o RF - 01

O diagrama de sequência ilustrado na figura 4.5 representa as interações entre componentes realizadas durante o RF-01 (Criar um one Widget), caso estas interações sejam realizadas com sucesso.

O diagrama é constituído por um *Actor*, que neste caso representa o cliente E-goi e ainda por 4 *lifelines* que representam o interface gráfico (BO-UI), a API interna da E-goi (Services), a base de dados onde é armazenada a informação relativa a um One Widget (MariaDB) e, por fim, o componente externo responsável pelos *triggers* (Matomo).

Quando o cliente inicia a ação de criar um One Widget, a UI faz um pedido ao Services, que insere os dados relativos a esse One Widget na base de dados. De seguida, o cliente define opções específicas (nome, idioma) relativas, sendo que é feito um *update* à informação na base de dados e a interface apresenta ao utilizador o editor do One Widget.

No Editor, é possível alterar a aparência do One Widget, esta que funciona da mesma forma que as opções, ou seja, faz apenas um *update* na base de dados da aparência. O interface gráfico apresenta ao cliente uma preview da aparência conforme esta seja alterada.

Para além disso, ainda é possível adicionar e remover canais, sendo que é feito novamente um *update* à base de dados para que tal aconteça, associando ou desassociando X canal ao One Widget criado. Esta sequência pode acontecer diversas vezes, dependo do número de canais que são removidos e adicionados. Conforme exista a remoção ou adição de canais, a própria lista será atualizada. É preciso ter em conta que para adicionar/remover canais, este precisam de ser criados primeiro, mas como este fluxo é apenas referente à criação de um One Widget, as sequências para esse caso não são exibidas.

De seguida, haverá um redirecionamento para a página dos comportamentos em que serão definidos pelo cliente os *triggers* para o One Widget e de seguida estes serão guardados, existindo novamente um *update* na base de dados para registar esta alteração

Depois dos comportamentos definidos, é apresentada a página dos domínios, onde podem ser adicionados todos os domínios em que o cliente quer que este One Widget apareça. Ao definir um domínio, o One Widget fica ativo para aquele domínio pois é feito o envio ao matomo tanto do *trigger*, como da tag que corresponde aos mesmos. Para que o One Widget apareça no site, apenas é necessário colocar o script apresentado na Figura 2.2 no próprio código fonte do *website* do cliente.

Por fim, existe a confirmação da criação do One Widget e o cliente é redirecionado para a página com a listagem de One Widgets.

4.2 Recursos

Para tornar possível a comunicação entre a interface gráfica e o servidor que contém toda a lógica do negócio, é necessário definir recursos que possibilitem a gestão e persistência de dados.

Atualmente, no produto E-goi, o componente Services, sendo responsável por toda a lógica do negócio, disponibiliza vários recursos compostos por *endpoints*, que são rotas que estabelecem a conexão entre a interface e o servidor e assim possibilitam a troca de dados entre vários componentes.

Neste projeto, tal como apresenta a Figura 4.4, o componente Services é responsável por estabelecer a ligação à base de dados, comunicando com todos os componentes da solução e gerindo todo o fluxo de dados. Assim, seguindo a arquitetura e tecnologias já existentes no produto E-goi, foi criado um novo recurso composto por vários *endpoints* essenciais no componente Services:

RECURSO api/forms/onewidget - Recurso responsável por agregar todos os *endpoints* para a gestão do One Widget;

GET /forms/onewidget - Obter todas os One Widgets de um utilizador;

GET /forms/onewidget/:id - Obter um One Widget com um determinado id;

POST /forms/onewidget - Criar um One Widget;

PUT /forms/onewidget/:id - Atualizar um One Widget com um determinado id;

DELETE /forms/onewidget/:id - Eliminar um One Widget com um determinado id;

GET /forms/onewidget/:id/count - Obter todos os contadores relativos às métricas de um One Widget;

POST /forms/onewidget/:id/count - Adiciona uma nova entrada para um contador;

GET /forms/onewidget/:id/channel - Obter todos os canais presente no One Widget de um utilizador;

GET /forms/onewidget/:id/channel/:id - Obter um canal com determinado id e presente no One Widget de um utilizador;

POST /forms/onewidget/:id/channel - Adiciona um canal a um One Widget;

PUT /forms/onewidget/:id/channel/:id - Atualiza um canal, com determinado id, de um One Widget;

DELETE /forms/onewidget/:id/channel/:id - Eliminar um canal, com determinado id, de um One Widget;

GET /forms/onewidget/:id/channel/:id/count - Obter todos os contadores relativos às métricas de um canal de um One Widget;

POST /forms/onewidget/:id/channel/:id/count - Adiciona uma nova entrada para um contador de um canal que está associado a um One Widget;

Sendo o Services uma REST API em que todos os pedidos são feitos por HTTPS, é possível recorrer à utilização de *headers*, que permitem definir uma chave de segurança adicional a todos os pedidos, o que garante a segurança de todos os acessos aos recursos.

4.3 Mock-ups

Para o desenvolvimento da interface gráfica da solução, recorreu-se ao desenho de *mock-ups*, que representam um modelo em escala real da solução final.

A E-goi é uma empresa dividida em várias equipas, tendo cada equipa uma responsabilidade diferente. Uma das equipas existente é a de User Experience and Assurance (UXA), responsável por desenvolver os *mock-ups* das novas funcionalidades a serem desenvolvidas, assim como testar e validar a solução final de cada uma (sejam funcionalidades novas ou resolução de bugs). Todos os *mock-ups* para a interface gráfica da solução final deste projeto foram desenvolvidos juntamente com a equipa de UXA da E-goi (Exceto o da figura 4.6).

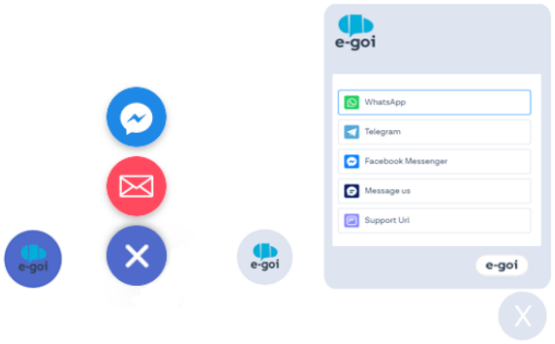


Figura 4.6: Mock-up inicial do One Widget

A figura 4.7 ilustra o *mock-up* inicial das duas alternativas desenvolvidas e por conseguinte apresentadas ao CTO e à equipa de UXA. Depois de uma breve reunião, decidiu-se adotar o design da direita por ser capaz de guardar mais canais sem que cause tanto ruído visual. No design final foi removido o icon " e-goi " que se encontra abaixo do grupo de canais.

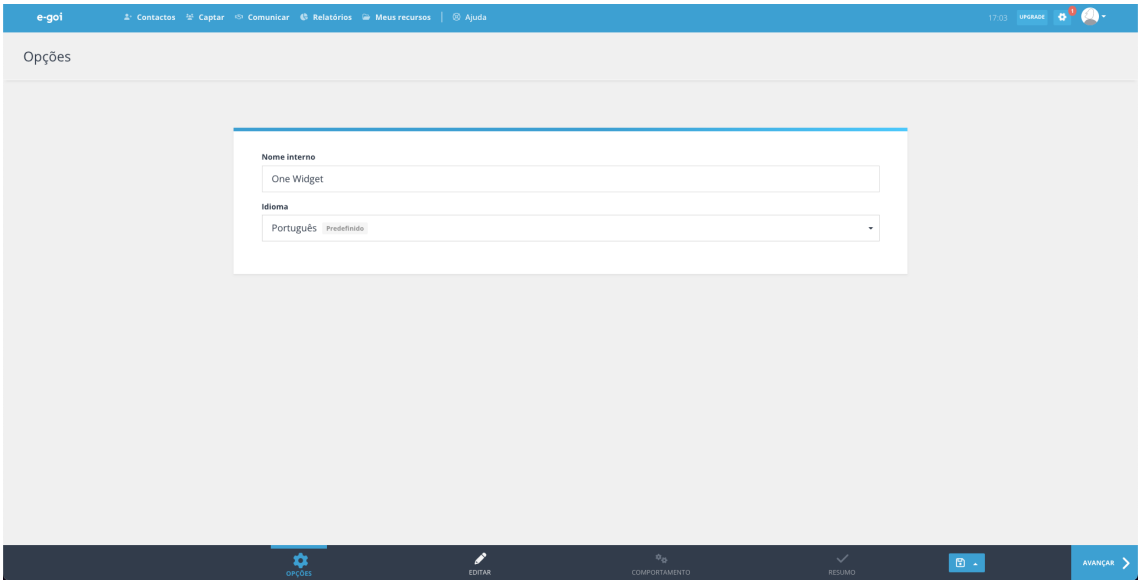


Figura 4.7: Página das opções

Na figura 4.7 está representado o *mock-up* da página de opções de um One Widget. A página de opções é o primeiro passo do fluxo de criação de um One Widget. Como é possível verificar, a interface gráfica apresenta dois campos para o utilizador preencher, sendo eles o nome interno e o idioma.

O nome interno é apresentado como um input de texto, onde o utilizador define o nome do seu One Widget. O idioma é uma *select-box* composta pelas opções Português, Português do Brasil, Inglês e Espanhol, que são os idiomas suportados pelo produto E-goi. No fundo da página de opções, existe um *footer* que permite ao utilizador regressar à página anterior ou avançar para a página seguinte.

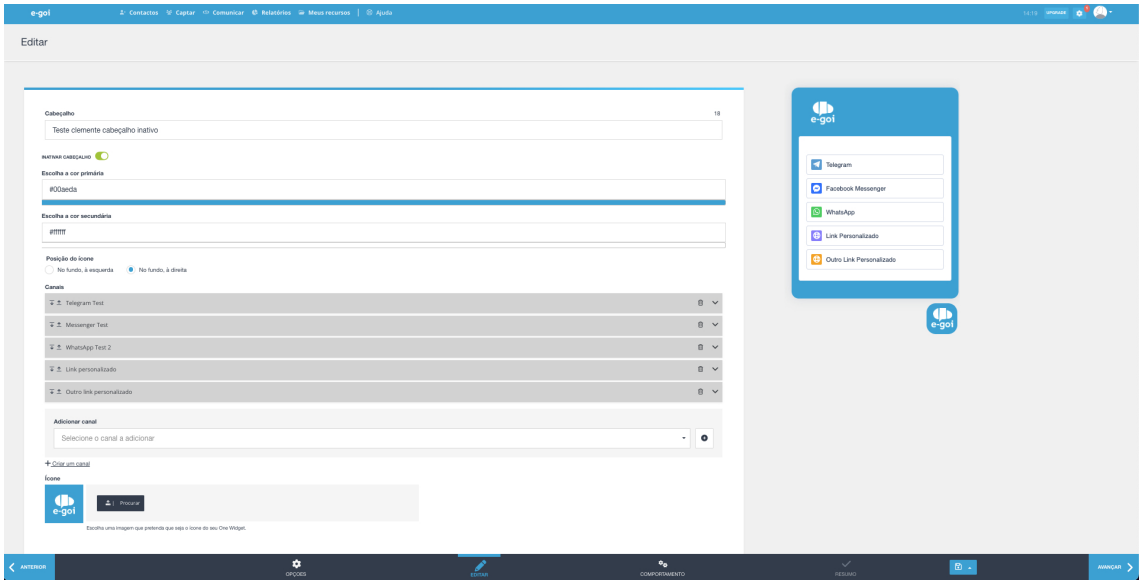


Figura 4.8: Página da Edição

A figura 4.8 representa a página de edição do One Widget, página esta que mostra um *preview* de como o One Widget fica ao longo das alterações que são feitas. Inicialmente é pedido o cabeçalho, que será o texto que ficará acima dos canais presentes no One Widget. Isto serve caso o cliente queira passar uma mensagem como " Fale Connosco! ", por exemplo. De seguida, é possível personalizar cor, posição, canais e ícone que farão parte do One Widget a ser exibido ao cliente.

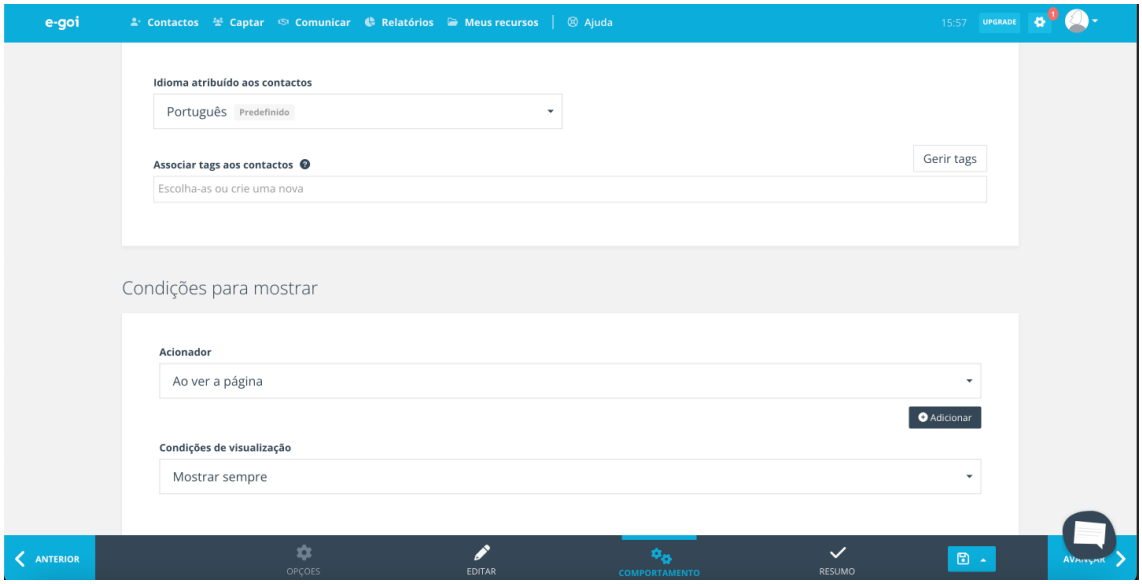


Figura 4.9: Página do comportamento

Na figura 4.9 está representado o *mock-up* da página do comportamento de um One Widget. Nesta página é pedido ao utilizador que selecione o idioma e *tags* (identificador para grupos de subscritores) que pretende adicionar aos contactos que sejam adicionados através do One Widget.

Para além disso, é pedido ainda o acionador que é o tipo de *trigger* que o utilizador pretende que faça o One Widget aparecer no ecrã da sua página web. É apresentada uma *select-box* com os tipos: ao ver a página, ao percorrer a página, ao sair da página, ao fim de um certo tempo na página e ainda ao ver determinado URL. Por último, é apresentada outra *select-box* para o utilizador definir uma condição de visualização do One Widget, com as opções: mostrar sempre e ainda mostrar num intervalo de datas.

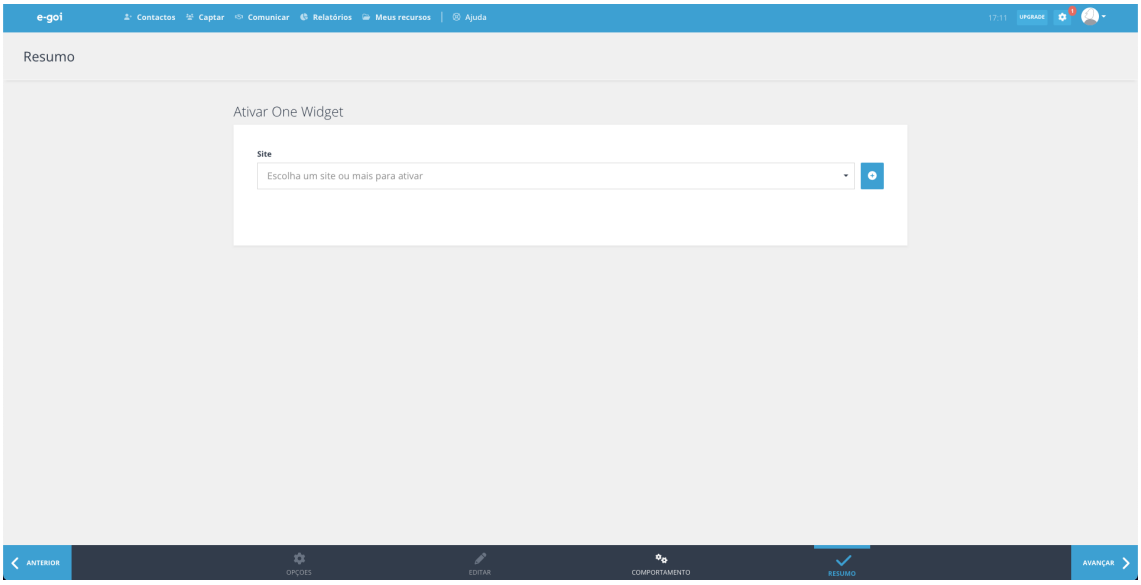


Figura 4.10: Página de resumo

Na figura 4.10 está representado o *mock-up* da página de resumo de um One Widget. Neste página apenas existe uma *select-box* em que o cliente pode selecionar o domínio ou domínios em que o One Widget aparecerá.

Canais

Depois de um dialogo com os interessados, nomeadamente CTO e equipa de UXA, decidiu-se que os canais que seriam aplicados inicialmente ao projeto seriam os canais sociais, webpush e com link personalizado. Para o Whatsapp e webpush, eles terão o mesmo design de como ilustrado na secção 2.2, figuras 2.5 e 2.6, respetivamente. Os novos canais sociais como Messenger, Viber e Telegram seguem a mesma lógica de estrutura do Whatsapp, havendo apenas a alteração de temas de cores e ícones. Estes serão, assim, adaptados ao One Widget, como mostra na preview da figura 4.8. O link personalizado será apenas um botão terá como função o redirecionamento para uma URL que fique a cargo o cliente.

Capítulo 5

Implementação da Solução

Neste capítulo é apresentada a implementação da solução, com base no desenho definido no capítulo anterior. Para introduzir este capitulo, são descritas as metodologias utilizadas para o desenvolvimento do projeto, seguido de uma abordagem da implementação das partes mais relevantes, para cada componente da arquitetura da solução.

5.1 Metodologia de Desenvolvimento

A qualidade do desenvolvimento de software e muito importante para garantir que o produto final satisfaz os requisitos do cliente. Para que o trabalho seja organizado e coeso, a(s) equipa(s) de desenvolvimento precisam de adotar métodos de trabalho que possibilitem que tal acontece. Apesar deste trabalho ser de âmbito individual, é necessário seguir as praticas da empresa de modo a que o trabalho de integração desta implementação no produto global seja o mais facilitado possível.

Começando pelo sistema de controlo de versões, a E-goi utiliza a ferramenta Git [37] em conjunto com a aplicação GitLab [38] que é bastante similar ao GitHub [39], mas com a particularidade do primeiro permitir que os *developers* armazenem o código nos próprios servidores, em vez de servidores de terceiros, o que dá um maior controlo de todo o código dos projetos do produto. Para cada projeto existem 4 *branches* principais, uma onde é feito o desenvolvimento e de seguida o Code Review (CR) pelo *team leader*, outra onde são feitos os testes ao código desenvolvido pelo equipa de Quality Assurance (QA). Depois do código passar nos testes, é ainda enviada para a *branch release*, onde o código será revisto novamente, mas desta vez pela equipa de Site Reliability Engineering (SRE) e caso este passe, será enviada no dia do *deploy* para a *branch master*, onde o código desenvolvido fica disponível em produção.

De modo a que haja uma integração continua (*continuous delivery*) dos projetos, a E-goi utiliza a ferramenta Jenkins [40]. O Jenkins é um servidor de automação de código aberto independente, que pode ser usado para automatizar todos os tipos de tarefas relacionadas a construção, teste e entrega ou implantação de software. Esta alocado num servidor interno e integrado com o GitLab, o que permite gerir todos os projetos do produto com maior facilidade e agilidade. Sempre que e enviado código novo da *branch* de desenvolvimento para a de testes, ou da de testes para a de *release* e da *branch* de *release* para a de produção, o Jenkins começa por analisar toda a sintaxe do código, procedendo depois para a compilação do projeto. De seguida, despoleta testes automáticos ao código através de uma integração com o SonarQube [41]. Por fim, e realizado o *deploy*, operação que deixa o código disponível para o utilizador, tanto em ambientes de testes como de produção, dependendo de qual *branch* o *build* é iniciado.

Em suma, utilizando estes processos para o desenvolvimento de software, torna-se mais simples a gestão de todo o código desenvolvido assim como a garantia da sua qualidade, sendo ainda garantida a analise a existência de bugs e mas praticas de programação, gestão de testes e garantias de percentagens mínimas de cobertura de testes. Também simplifica os processos internos das equipas uma vez que todos os elementos seguem o mesmo fluxo para a criação e manutenção de código.

5.2 Camada de Dados

A camada de persistência do produto E-goi é uma base de dados relacional, ou seja, define relações sob a forma de tabelas, estruturadas recorrendo à linguagem de programação SQL. A linguagem de programação SQL permite armazenar, manipular e recuperar dados em bases de dados relacionais. Este tipo de base de dados é o mais indicado para projetos em que existe uma grande correlação entre tabelas e elevada quantidade de transações de dados, como é o caso deste projeto. Na Figura 5.1 está representada a camada de dados para este projeto.

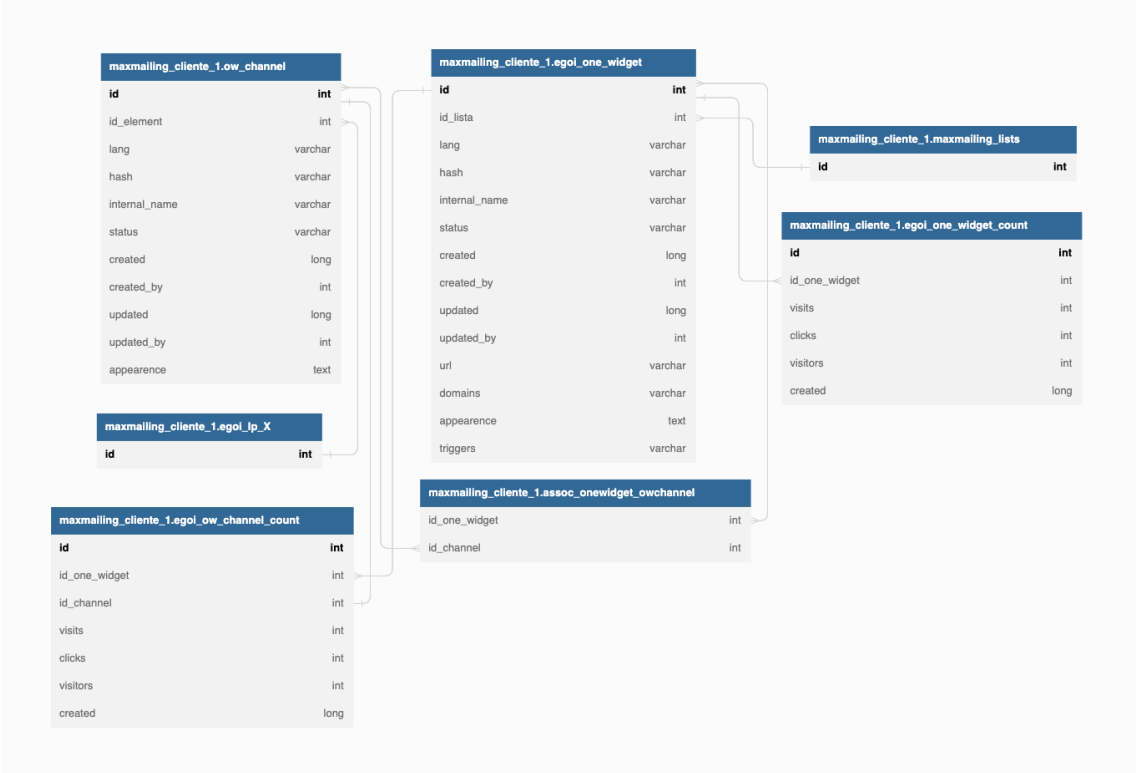


Figura 5.1: Camada de Dados

Na tabela 5.1 é possível consultar a legenda da figura 5.1

Tabela 5.1: Definição das colunas das tabelas da Camada de Dados

Coluna	**Descrição**
id	identificador principal do elemento
id_lista	identificador da lista de contactos associada ao elemento
lang	idioma do elemento
hash	*hash* do elemento (identificador utilizado para pesquisas na DB)
internal_name	nome do elemento
created	data de criação do elemento
created_by	id do utilizador que criou o elemento
updated	data da ultima alteração do elemento
updated_by	id do utilizador que alterou o elemento pela ultima vez
url	link público do elemento
domains	dominios onde o elemento está ativo
appearence	dados relativos à aparência do elemento
triggers	tipo de *triggers* e condições de visualização do elemento
id_one_widget	id do One Widget associado ao elemento
id_channel	id do canal associado ao elemento
visits	número de visitas de um elemento
clicks	número de cliques de um elemento
visitors	número de visitantes de um elemento

Na figura 5.1 é possível observar as tabelas pertencentes ao projeto e as suas respetivas colunas, assim como a sua ligação à parte do modelo de dados já existente na E-goi (que são as tabelas representadas apenas com o campo id).

A tabela central do projeto é a egoi_one_widget que é onde será armazenada toda a informação sobre todos os One Widgets criados para uma conta (sendo que maxmailing_cliente_1 é referente à conta do cliente de id 1).

Como já relatado na secção 4.1.1, um One Widget esta associado a uma lista de contactos, daí a relação de *one-to-one* entre a tabela das listas de contactos (maxmailing_lists) e a tabela do One Widget.

A tabela egoi_lp_X representa a tabela que contém a informação sobre um canal criado, onde 'X' representa o tipo do canal. Um exemplo será um whatsapp leads que ao ser criado, será inserido na tabela egoi_lp_whatsapp. Caso este também fique associado a um One Widget, durante a criação/edição do mesmo, essa informação será adicionada na tabela ow_channel.

Como um One Widget pode ter um ou mais canais associados e um canal pode pertencer a um ou mais One Widgets, a relação entre a tabela do One Widget e a tabela dos canais (ow_channel) será de *many-to-many*. Neste tipo de relação entre tabelas, faz sentido criar uma tabela de associação para que seja mais fácil, neste caso, saber quais os canais que pertencem a um One Widget.

Por ultimo, tanto os canais, como o próprio One Widget, precisam de ter uma tabela para as métricas e daí existirem as tabelas egoi_one_widget_count (métricas para os One

Widgets) e egoi_ow_channel_count (métricas para os canais). Cada canal e cada One Widget têm uma ou mais entradas para métricas, pois estas são coletadas periodicamente.

5.3 Camada de Negócio

O componente Services é uma REST API desenvolvida em PHP e utiliza a *framework* Laminas [42], antes denominada de Zend Framework [43], que é uma biblioteca de código aberto que possibilita a utilização de *endpoints* para que todo o sistema E-goi interaja entre si. Uma vez que permite a comunicação com vários componentes e gere todos os dados necessários, pode ser considerado como sendo a camada de lógica de negócio do sistema.

5.3.1 Recursos

Foram implementados todos os *endpoint* referidos na Secção 4.2 na classe *OneWidgetResource*, *OneWidgetChannelResource* e *CountResource*. Esta classes foram criadas para controlar a entrada e saída de dados da base de dados do sistema, disponibilizando recursos para que os clientes possam fazer pedidos através da interface gráfica do sistema. Cada *endpoint* possui um método próprio responsável por tratar a informação recebida. Como exemplo, podemos analisar os métodos responsáveis pelo tratamento da informação associados aos *endpoints* destinados ao One Widget:

GET /forms/onewidget - método getList() que devolve a lista de todos os One Widgets do utilizador;

GET /forms/onewidget/:id - método get($id) que devolve o One Widget com o $id recebido;

POST /forms/onewidget - método create($data) que cria um One Widget com base nos dados recebidos na variável $data;

PUT /forms/onewidget/:id - método update($id, $data) que atualiza o One Widget com base nos dados recebidos na variável $data, onde o id tem o valor da variável $id;

DELETE /forms/onewidgets/:id - método delete($id) que elimina um One Widget, onde o id tem o valor da variável $id.

5.3.2 Integração com o Matomo Tag Manager

O Matomo disponibiliza uma API que, há semelhança do Services, utiliza o protocolo HTTP como protocolo de comunicação.

Para fazer pedidos ao Matomo, foi utilizado o método makeMatomoRequest que se encontra ilustrado no *code listing* 5.1. Este método recebe como parâmetros os dados a enviar, o id do cliente em questão e o domínio que será afetado com estas alterações.

```
...
private function makeMatomoRequest($data, $clientId, $domain){
    if (empty($this->clientHttp)) {
        $this->clientHttp = new Client();
    }

    $data = array_merge($data, [
        'module' => 'API',
        'format' => 'JSON',
        'idSite' => $clientId,
        'domain' => preg_replace('/^(?:https?:\/\/)?/i', '', trim(
    $domain, '/'))
    ]);

    $this->clientHttp->setUri($this->matomoUri);
    $this->clientHttp->setMethod('GET');
    $this->clientHttp->setParameterGet($data);
    $this->clientHttp->setHeaders($this->matomoAuth);
    $this->clientHttp->setOptions(array(
        'timeout' => 30
    ));

    return json_decode($this->clientHttp->send()->getBody(), true);
}
...
```

Listing 5.1: Método que faz o pedido ao Matomo

Atualmente, no sistema E-goi, cada domínio do utilizador possui um *container* único associado, onde é possível definir *tags* e *triggers*.

Para adicionar *tags* e *triggers* de um One Widget ao Matomo e consequentemente ativar um One Widget, foi implementado o método oneWidgetTagSetup no TagManagerService que, tal como o nome indica, é o responsável pelas operações no Matomo Tag Manager.

Quando o utilizador ativa um One Widget, o componente BackOffice User Interface (BO-UI) faz um pedido ao Services através do método POST e *endpoint* /matomo/tagmanager/-configs. Este pedido é depois processado no *MatomoConfigsController*, onde é chamado o método *oneWidgetTagSetup*.

Este método adiciona a *tag* que, posteriormente, é enviada para o Matomo.

```
...
public function oneWidgetTagSetup($clientId, $id, $domain) {
    ...
    $this->addContainerTag (
        $clientId,
        $domain,
        'oneWidgetContainer',
        ['customHtml' => $script ],
        $trigger,
        'CustomHtml',
        $fireLimit,
        $startDate,
        $endDate
    );
    ...
}
...
private function addContainerTag($clientId, $domain, $name, $parameters,
    $trigger, $type, $fireLimit, $startDate, $endDate) {

    $data = [
        'method' => 'TagManager.addContainerTag',
        'idContainer' => $this->\textit{container}['idcontainer'],
        'name' => $name,
        'parameters' => $parameters,
        'trigger' => $trigger,
        'type' => $type,
        'fireLimit' => $fireLimit,
        'startDate' => $startDate,
        'endDate' => $endDate
    ];

    return $this->makeMatomoRequest($data, $clientId, $domain);
}
...
```

Listing 5.2: Métodos adicionam uma tag ao Matomo

No caso do code listing 5.3, o processo é o mesmo que o das *tags*, excepto que o conteúdo a ser enviado ao Matomo não é uma *tag*, mas sim um *trigger*.

```
...
public function oneWidgetTagSetup($clientId, $id, $domain) {
    ...
    $this->addContainerTrigger(
        $clientId,
        $domain,
        $trigger['type'],
        $trigger['name'],
        $trigger['parameters']
    );
    ...
}
...
private function addContainerTrigger($clientId, $domain, $type, $name,
    $parameters)
    {
    $data = [
        'method' => 'TagManager.addContainerTrigger',
        'idContainer' => $this->\textit{container}['idcontainer'],
        'type' => $type,
        'name' => $name,
        'parameters' => $parameters
    ];

    return $this->makeMatomoRequest($data, $clientId, $domain);
}
...
```

Listing 5.3: Métodos adicionam um trigger ao Matomo

Para obter o *snippet* que é necessário para colocar no site do cliente para que seja possível disponibilizar o One Widget é necessário fazer uma chamada ao método *getOneWidgetCode* em que é necessário o parâmetro appCode, que neste caso é a hash do próprio One Widget.

Para além disso, é colocado nesse script o ficheiro para que tem toda a lógica de necessário para fazer exibir um One Widget no *website* do cliente. A *Code List* 5.4 ilustra como foi desenvolvido esse método para obtenção do snippet.

```
...
public function getOneWidgetCode($appCode)
    {

        $file = $this->getServiceLocator()->get('config')['egoiPublic']['onewidget']['jsFile'];

        $script = "
            <script type=\"text/javascript\">
                var _egoiwp = _egoiwp || {};
                (function(){
                _egoiwp.code = \"$appCode\";
                var d=document, g=d.createElement('script'), s=d.getElementsByTagName('script')[0];
                g.type='text/javascript';
                g.defer=true;
                g.async=true;
                g.src=u+'$file';
                s.parentNode.insertBefore(g,s);
                })();
        ";

        $script .= "</script>";

        return $script;
    }
...
```

Listing 5.4: Método para obter o snippet de um One Widget

Depois de ser feito o envio dos *triggers* e *tags* para o Matomo através do método apresentados anteriormente, o *website* do cliente já permite que o One Widget apareça. Mas, para que isso aconteça, é necessário que o *script* que está a correr no site do cliente, através do *snippet* (que é o referido na *Code List* 5.4) que é fornecido cliente, receba informação do Matomo quando os *triggers* forem atingidos.

A seguir o *script* faz um pedido à API publica da E-goi (E-goi Public) a pedir o One Widget designado. Esta, por sua vez, faz um pedido à API interna (Services) da E-goi a pedir essa mesma informação. O Services retorna essa informação ao E-goi Public, este, que por sua vez faz a construção do One Widget com os dados recebidos e depois devolve essa informação. O One Widget construido é, então devolvido e exibido tal como ilustra a figura 5.2.

Figura 5.2: One Widget - ícone

Caso um utilizador do site do cliente clique no One Widget, este exibe os canais disponiveis para o utilizador escolher, tal como mostra a figura 5.3.

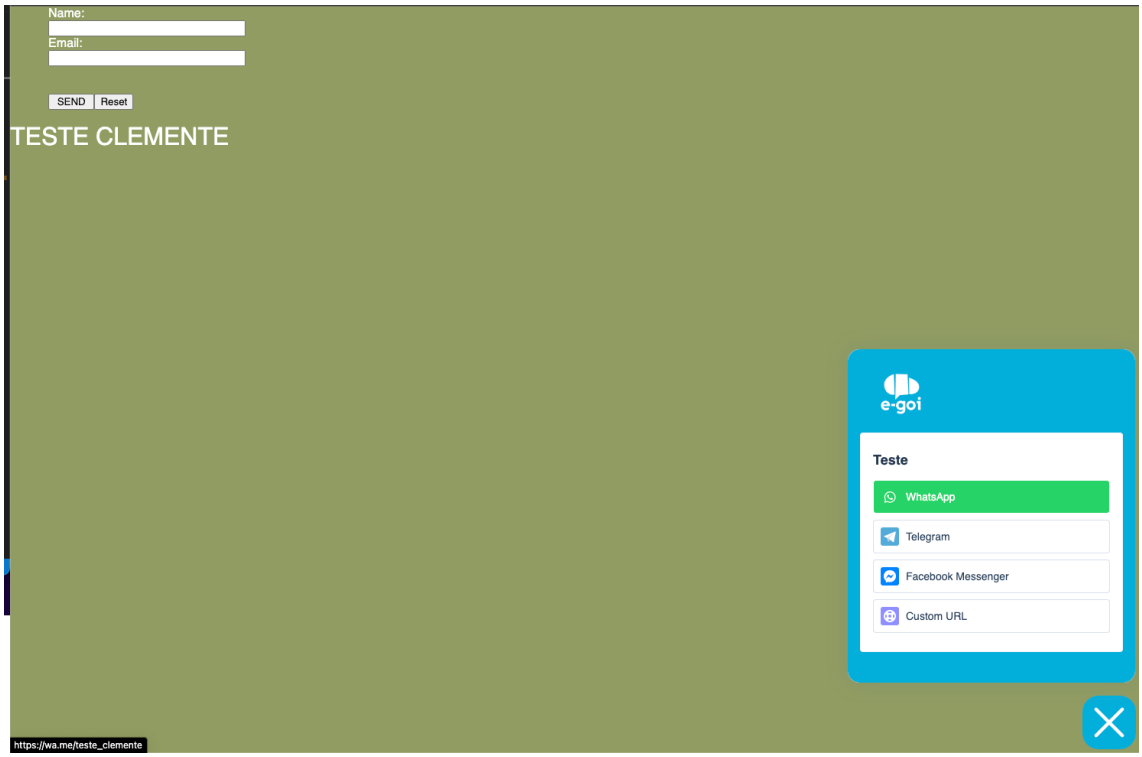


Figura 5.3: One Widget - lista

5.4 Camada de Apresentação

O componente que representa a camada de apresentação do sistema é o BO-UI. A camada de apresentação é aquela que interage diretamente com o utilizador, tem a responsabilidade de mostrar informação ao utilizador e enviar a informação introduzida pelo utilizador para a camada de negócio.

Fluxo de gestão

Toda a interface gráfica de gestão dos One Widgets, foi implementada no componente BO-UI, com base nos *mock-ups* apresentados na Secção 4.3. Para o desenvolvimento desta interface gráfica, foi utilizada a *framework* AngularJS[36], uma vez que todo o projeto do componente BO-UI, já se encontra desenvolvido a partir da mesma.

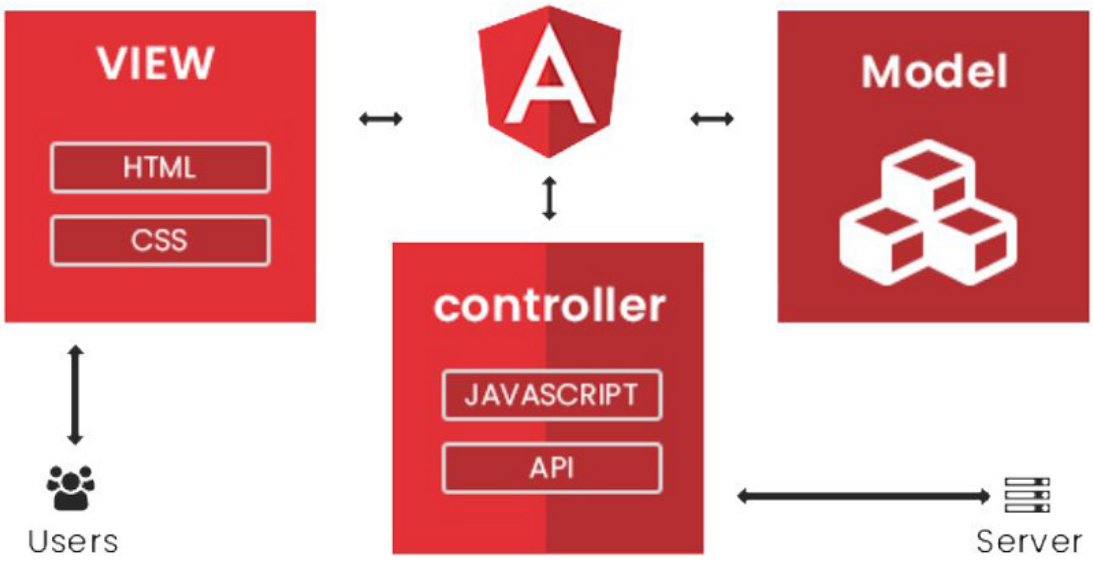


Figura 5.4: Arquitetura MVC do AngularJS

Uma aplicação AngularJS tem por base uma arquitetura Model-View-Controller (MVC), tal como é apresentado na Figura 5.4. O *model* é parte do conceito MVC, representando os dados manipulados pelo utilizador, direta ou indiretamente. O AngularJS mantém o *model* armazenado num objeto intrínseco da arquitetura da *framework*, chamado *scope*. A *view* é a interface gráfica com o qual o utilizador interage e manipula dados da aplicação, utilizando uma instância do *scope* disponibilizada na *view*, por meio de um *controller*. Existem ainda serviços que podem ser implementados para encapsular toda a lógica de operações existente, sendo compartilhados por um ou mais *controllers* [44].

Foram criados vários ficheiros no projeto BO-UI e também alterados vários já existentes, para implementar todas as funcionalidades da solução final. Cada *mock-up* apresentado na Secção 4.3 tem o seu próprio ficheiro *controller*, desenvolvido em JavaScript e a respetiva *view* desenvolvida em HTML e CSS. O fluxo de gestão de um One Widget, manipula os dados do mesmo, utilizando o objeto *scope*. Sempre que o utilizador muda de página no fluxo, é feito um pedido HTTP ao componente Services, que atualiza a informação do One Widget na base de dados do sistema.

Capítulo 6

Experimentação e Avaliação

Ao longo do processo de desenvolvimento, foram realizados testes ao projeto, de modo a verificar que o seu desenvolvimento cumpria com os objetivos definidos. Contudo, existem grandezas que só são possíveis de avaliar quando toda a implementação está concluída. Este capítulo tem por objetivo avaliar a qualidade do software da solução final desenvolvida e também a sua usabilidade, através de experiências e testes, e analisando o resultado dos mesmos.

6.1 Métricas de Avaliação

A primeira métrica de avaliação da solução tem como fator de avaliação, para este caso, a qualidade do código do software desenvolvido, de modo a entender se este está apto para estar em produção. A segunda métrica definida, diz respeito à usabilidade da solução final. Sendo um projeto em que o utilizador interage com uma interface gráfica, é importante avaliar se a sua experiência de utilização é positiva. Para isso, foi definida a satisfação do utilizador, como segundo fator de avaliação.

6.2 Metodologias de Avaliação

Definidas as métricas, é altura de escolher as metodologias que serão utilizadas para avaliar a qualidade do software. Para a avaliação da métrica de qualidade do código software desenvolvido, foram realizados testes de integração e uma análise ao relatório da *framework* SonarQube. Para avaliar a métrica de satisfação do utilizador foi realizado um inquérito de satisfação e usabilidade, tendo por base a System Usability Scale (SUS) de John Brooke [45]. O resultado final pode variar entre 0 e 100.

6.3 Especificação da Hipótese

No âmbito deste projeto, foram definidas duas hipóteses, para verificar o cumprimento das métricas enunciadas na secção 6.1.

Para a verificação da qualidade do software, tem-se a seguinte hipótese:

- O código desenvolvido para o One Widget, tem a qualidade suficiente para ser integrado no produto E-goi.

Para a satisfação do utilizador tem-se a seguinte hipótese:

- O resultado final tem de ser igual ou superior a 75.

6.4 Resultados

Depois de concluir os testes para avaliar as hipóteses definidas anteriormente, é altura de analisar os resultados obtidos. Nesta secção são apresentados os resultados de cada teste, juntamente com uma análise aos mesmos, o que permitirá aferir se os objetivos foram atingidos e as melhorias que podem ser aplicadas ao sistema.

6.4.1 Testes de Software

De maneira a que se possa avaliar a qualidade de um software, é necessário realizar testes de integração e testes de qualidade de código.

Os testes de integração foram realizados nas diferentes rotas de comunicação de forma a verificar que todos os módulos funcionam de forma correta juntos.

Para isso, foi utilizada a ferramenta Postman [46] que permite não só a execução de pedidos a APIs, mas também a execução de testes para as mesmas.

O objetivo destes testes, consistiu em cobrir os métodos da camada de negócio implementados, que interagem diretamente com a base de dados do sistema.

Na figura 6.1, estão representados os testes para os *endpoints* relacionados com a classe *OneWidgetResource* mencionados na secção 5.3.1.

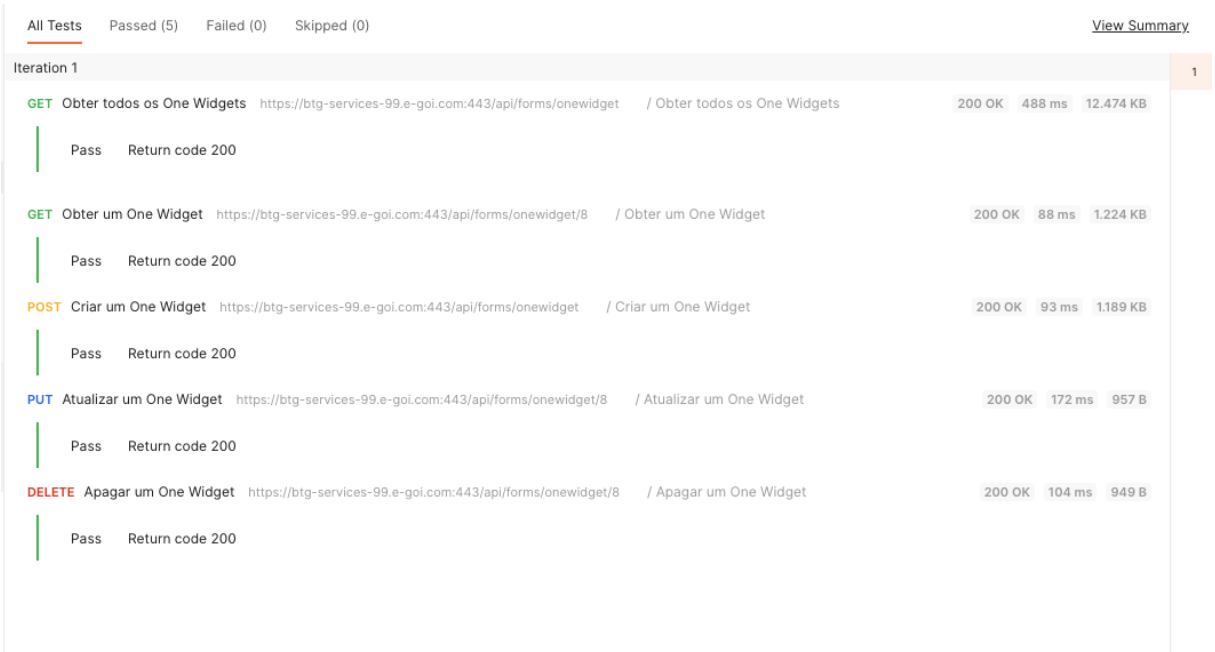


Figura 6.1: Testes de integração - Postman

Na figura 6.1 é possível observar os resultados dos testes realizados. Para que os resultados fossem satisfatórios, era necessário que, para todos os pedidos, fosse retornado o status code 200. Para o protocolo HTTP, 200 significa sucesso na resposta ao pedido efetuado. Pelo a análise dos resultados, podemos concluir que todos os *endpoints* passaram com sucesso.

No que diz respeito aos testes à qualidade do código, foi utilizada a plataforma SonarQube [41]. Esta é uma ferramenta automática de revisão de código para detetar *bugs*, vulnerabilidades e más práticas de programação em projetos locais ou que se encontrem em repositórios. Essa informação é exibida num relatório final e pode ser exibida tal como ilustra a figura 6.2.

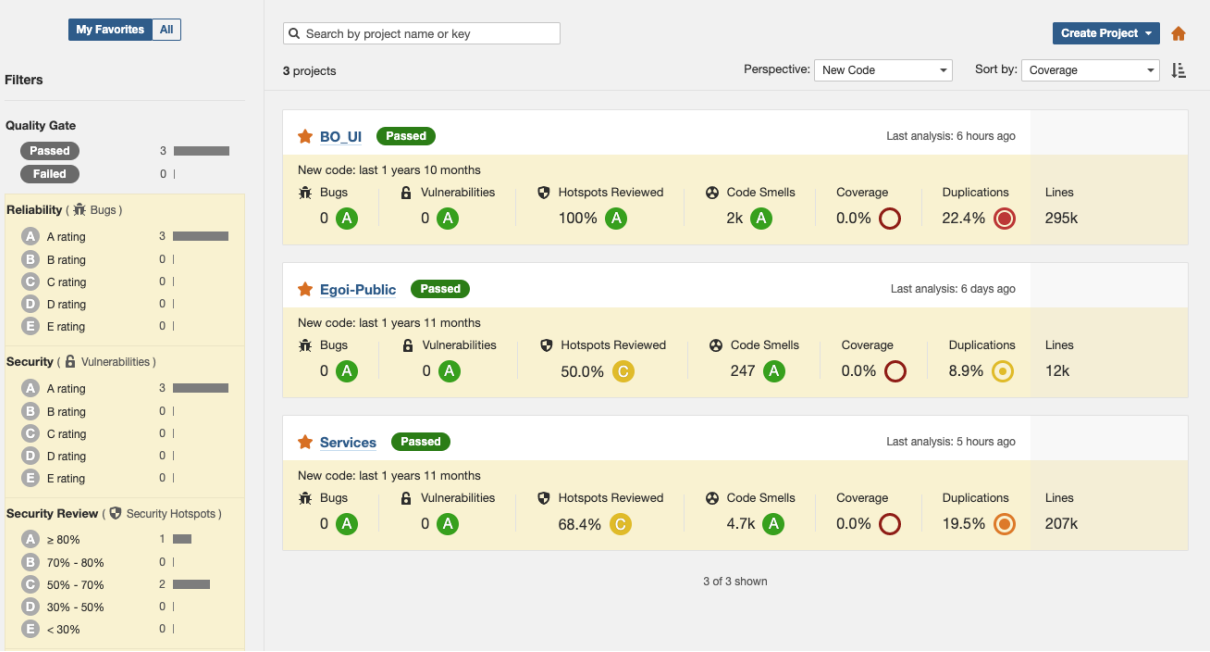


Figura 6.2: Teste dos recursos do One Widget

Neste relatório do SonarQube dos 3 projetos em que o One Widget faz parte, é possível observar que em todos eles passaram e apesar de a duplicação de código ser um problema visível, não existem *bugs*, nem vulnerabilidades detetadas.

O problema com este relatório é que exibe os resultados globais do novo código para cada um dos projetos, mas como é de esperar, o One Widget não é o único componente incluído nesta nova cobertura do novo código. Para resolver esse problema, basta aceder aos projetos e procurar pelos ficheiros do One Widget. Para cada projeto, o ficheiro que diz respeito ao One Widget encontra-se selecionado a azul e pode ser visto no Apêndice B.

6.4.2 Testes de Usabilidade

De modo a conseguir avaliar a satisfação do utilizador face à solução desenvolvida, foi realizado um teste de usabilidade, que tal como referido na secção 6.2, é baseado na SUS de John Brooke.

Estudar a usabilidade do software permite-nos saber:

- Se os utilizadores conseguem atingir os seus objetivos;
- Quanto esforço e recursos são necessários para atingir esses objetivos;
- Se a experiência foi satisfatória.

Inicialmente, foi feita uma breve apresentação da solução e das suas funcionalidades a alguns membros da empresa (cerca de 36 elementos) ligados ao marketing e vendas, UXA e também *developers*. Todos os envolvidos ou não tinham conhecimento prévio da solução, ou se tinham, esse conhecimento era tão baixo ao ponto de não influenciar este teste.

O inquérito é constituído por 10 itens (de Q.1 a Q.10), cada uma delas pode ser avaliada de 1 a 5. A figura 6.3 representa o gráfico de colunas empilhadas, com todos os resultados do inquérito realizado.

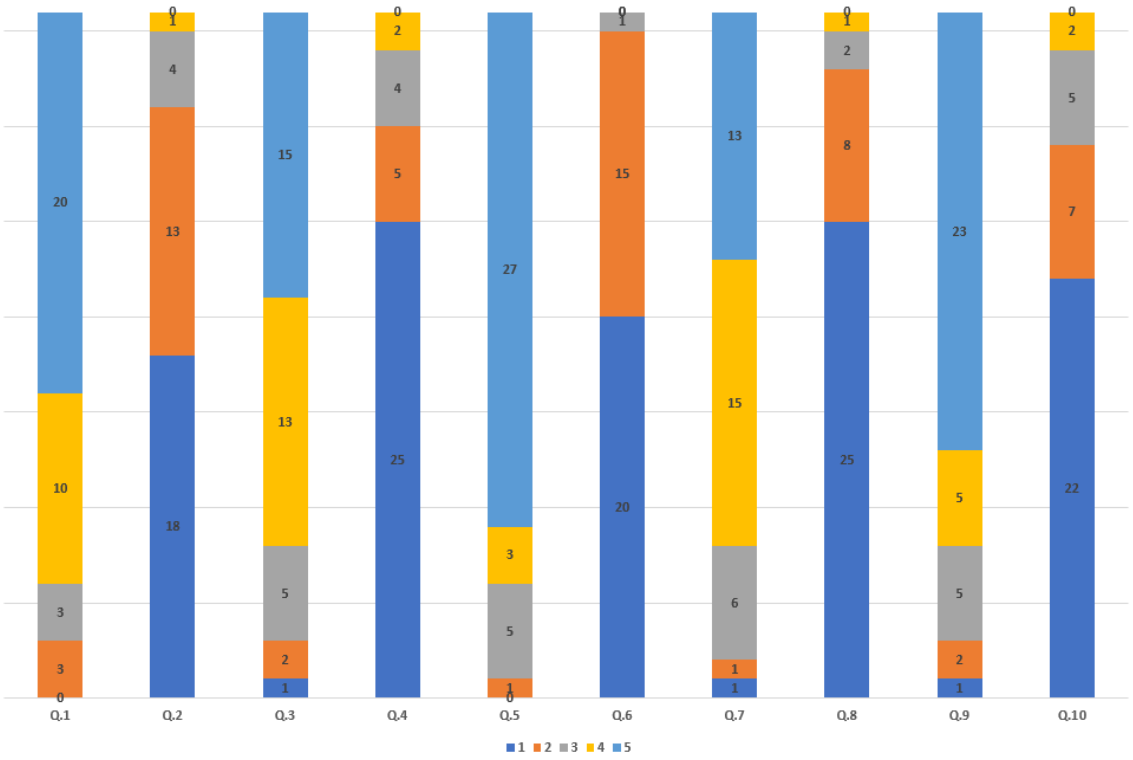


Figura 6.3: Resultados do teste de usabilidade

Para ser possível tirar conclusões sobre o inquérito, é necessário saber que:

- O resultado final varia entre 0 e 100;
- n representa a média da valor de cada resposta;
- Por cada resposta, o resultado pode variar entre 0 e 4;
- Para todas as opções que se encontram em números pares, o cálculo deve ser feito seguindo a expressão 5-n.
- Para todas as opções que se encontram em números ímpares, o calculo deve ser feito seguindo a expressão n-1.

A tabela 6.1 representa a classificação média de cada afirmação, com base nas expressões apresentadas nos últimos dois pontos.

Tabela 6.1: Média dos resultados para cada afirmação

Afirmação	1	2	3	4	5	6	7	8	9	10
Classificação	3.2	3.3	3.1	3.6	3.6	3.5	3.1	3.6	3.3	3.4

Dada a equação:

$$R = (n_1 + n_2 + n_3 + ... + n_y) * 2.5 \tag{6.1}$$

E sabendo que y representa o número de questão e R o resultado final do SUS, temos que:

$$R = 82.25 \tag{6.2}$$

Segundo James R. Lewis [47], de acordo com o resultado final do teste de usabilidade de um software, é dada uma nota, tal como ilustrado no Apêndice C.2, sendo que 68 (nota C) representa o meio das variações de resultados das avaliações.

Tendo em conta o resultado final obtido de 82.25, é então, possível concluir que a nota final é um A. Esta nota pode ser considerada positiva, visto que o One Widget se encontra em fase de testes e não é um produto maduro e encontra-se na sua fase inicial de testes.

6.5 Resultados do Produto

Depois de terminada a implementação da solução, o produto foi disponibilizado para testes na plataforma de desenvolvimento do E-goi. Isto servirá para que qualquer colaborador da empresa possa testar à vontade o novo produto, para que, na eventualidade de existir algum *bug*, este possa ser corrigido.

Como ainda não está nos planos da empresa o lançamento do produto neste trimestre, não é possível obter métricas sobre os resultados de evolução de utilização do novo produto por parte dos clientes. No entanto, isto permitirá com o produto amadureça, visto que, podem surgir sugestões de como melhorar o produto que não foram pensadas inicialmente e que tornará este produto ainda mais robusto.

Capítulo 7

Conclusão

O projeto documentado, teve como objetivo, o desenvolvimento de um agregador de canais, disponibilizando-os em apenas um só widget. Neste capítulo, todo o trabalho desenvolvido é analisado, abordando os objetivos alcançados e o trabalho futuro a realizar.

7.1 Objetivos alcançados

Na fase inicial deste projeto foram definidos objetivos para a solução final, enunciados na Secção 1.4, que são os seguintes:

1. Análise e desenvolvimento de ferramentas para a criação, edição e gestão *widgets* para agregação de canais;
2. Modelação de um design/arquitetura robusta de modo a poder ser adaptável às diferentes necessidades dos clientes;
3. Implementação de funcionalidades específicas, tais como personalização dos *widgets* e escolha dos canais disponíveis.

O objetivo número 1 foi divido em duas partes distintas. Inicialmente foi realizado um estudo do estado da arte das soluções existentes, apresentado na Secção 2.3, com a finalidade de perceber as ferramentas existentes atualmente no mercado e se estas seriam aplicáveis ao projeto.

Feita esta análise, foram levantadas várias funcionalidades que permitem o utilizador criar, editar e gerir os One Widgets, tal como apresentado na Secção 3.2.

Toda a implementação da solução final está descrita no Capítulo 5.

O objetivo número 2 foi atingido através de um desenho arquitetural coeso e consistente, definido no Capítulo 4. Com a utilização de boas práticas, o processo de desenvolvimento tornou-se mais simples e facilitado.

Por último, objetivo número 3 foi atingido através da implementação do sistema seguindo a arquitetura e *mock-ups* definida no Capítulo 4. Foram implementados vários tipos de personalização no One Widget, não só a nível de canais, mas também a nível de aparência (as cores, ícone e posição na página a serem os mais relevantes).

No Capítulo 6 foi avaliado o trabalho final e foi possível perceber que todas as hipóteses foram cumpridas. Todos os objetivos definidos inicialmente foram cumpridos, tendo sido desenvolvido um produto bastante consistente. Por estes motivos o projeto revela-se bem sucedido.

7.2 Trabalho futuro

Relativamente ao trabalho futuro, prevê-se que sejam feitas melhorias contínuas às funcionalidades do produto e também à sua usabilidade. É necessário atender sempre às necessidades do cliente, de modo a manter a sua satisfação. Foram implementados diferentes canais, mais podem ser implementados ainda mais, com diversos propósitos, de modo a tornar o produto mais robusto e atrativo.

Bibliografia

[1] Lawrence B. Chonko Rebecca A. VanMeter Douglas B. Grisaffe. «Of "Likes" and "Pins": The Effects of Consumers' Attachment to Social Media». Em: *Journal of Interactive Marketing* 32 (nov. de 2015), pp. 70–88. url: https://doi.org/10.1016/j.intmar.2015.09.001.

[2] Afrina Yasmin, Sadia Tasneem, Kaniz Fatema et al. «Effectiveness of digital marketing in the challenging age: An empirical study». Em: *International journal of management science and business administration* 1.5 (2015), pp. 69–80.

[3] Aklima Rahman Mitul. «Rise of Digital Marketing and Its Impact in Logistics & Supply Chain». Em: *Master's dissertation for the degree in Procurement and Supply Management* (ago. de 2020), pp. 1–21. url: http://dspace.bracu.ac.bd/xmlui/handle/10361/15140.

[4] Alexander Maedche Jan vom Brocke Alan Hevner. «Introduction to Design Science Research». Em: *Design Science Research. Cases* (set. de 2020), pp. 1–13. url: https://www.researchgate.net/publication/345430098_Introduction_to_Design_Science_Research.

[5] *Microsoft Excel – software de Folha de Cálculo: Microsoft 365*. url: https://www.microsoft.com/pt-pt/microsoft-365/excel.

[6] *The google analytics alternative that protects your data*. Fev. de 2022. url: https://matomo.org/.

[7] *Tag manager - analytics platform*. Set. de 2021. url: https://matomo.org/docs/tag-manager/.

[8] Simon-Kingsnorth. *Digital Marketing Strategy. An integrated approach to online marketing*. Kogan Page, 2016. isbn: 9780749474706.

[9] Mathias Gwadanican. *80+ website widgets - to grow your business*. Set. de 2021. url: https://elfsight.com/.

[10] *An omnichannel communications platform, built for Global Scale*. url: https://www.messagebird.com/.

[11] *Chaty: Wordpress Chat Plugin, whatsapp chat button, Messenger and contact WordPress plugin*. Fev. de 2022. url: https://premio.io/downloads/chaty/.

[12] *Smart website pop ups: Free exit intent popups amp; inline forms*. Dez. de 2021. url: https://www.poptin.com/.

[13] *Widgets amp; plugins for websites*. url: https://www.widg.io/.

[14] Lawrence D Miles. *Techniques of value analysis and engineering*. Miles Value Foundation, 2015.

[15] Joe Tidd e Keith Pavitt. «Managing Innovation: Integrating Technological, Market And Organizational Change». Em: (jan. de 2011).

[16] Abbie Griffin e Stephen Somermeyer. *The PDMA Toolbook 3 for New Product Development*. Jan. de 2002. doi: 10.1002/9780470209943.

[17] Roberto Flores e Ruth León. «Towards a reference framework and characterization of Advanced Design, a design culture for strategic designers». Em: *Strategic Design Research Journal* 11 (jul. de 2018). doi: 10.4013/sdrj.2018.113.06.

[18] Serdar S. Durmuşoğlu e Gloria Barczak. «The use of information technology tools in new product development phases: Analysis of effects on new product innovativeness, quality, and market performance». Em: *Industrial Marketing Management* 40.2 (2011). Special issue on Service-Dominant Logic in Business Markets, pp. 321–330. issn: 0019-8501. doi: https://doi.org/10.1016/j.indmarman.2010.08.009. url: https://www.sciencedirect.com/science/article/pii/S0019850110001446.

[19] Pierry Teza et al. «Direcionadores do processo de inovação: o papel da estratégia, liderança e cultura». Em: *Navus: Revista de Gestão e Tecnologia* 3 (nov. de 2013). doi: 10.18815/navus.v3i2.156.

[20] Ismail Mounir e Mickaël Gardoni. «FRONT-END DE MÉTRICAS DE INOVAÇÃO: PERGUNTA DE PESQUISA E REVISÃO DE LITERATURA». Em: *Anais do Congresso Internacional de Conhecimento e Inovação – ciki* 1.1 (nov. de 2020). doi: 10.48090/ciki.v1i1.913. url: https://proceeding.ciki.ufsc.br/index.php/ciki/article/view/913.

[21] Jouko Myllyoja. *Water business is not an Island: Assessing the market potential of environmental innovations*. Jan. de 2009, pp. 30–34.

[22] Ricardo Viana Vargas e PMP IPMA-B. «Utilizando a programação multicritério (Analytic Hierarchy Process-AHP) para selecionar e priorizar projetos na gestão de portfólio». Em: *PMI Global Congress*. Vol. 2009. sn. 2010.

[23] *Diagrams.net - free flowchart maker and diagrams online*. url: https://app.diagrams.net/.

[24] Ricardo Peculis e Derek Rogers. «A Task Model of Software Intensive Acquisitions: Integrated Tactical Avionics System for a Maritime Helicopter Case Study». Em: (fev. de 2022).

[25] Sebastian Barney, Aybüke Aurum e Claes Wohlin. «A product management challenge: Creating software product value through requirements selection». Em: *Journal of Systems Architecture* 54.6 (2008). Selection of best papers from the 32nd EUROMICRO Conference on 'Software Engineering and Advanced Applications' (SEAA 2006), pp. 576–593. issn: 1383-7621. doi: https://doi.org/10.1016/j.sysarc.2007.12.004. url: https://www.sciencedirect.com/science/article/pii/S1383762107001348.

[26] Alexander Osterwalder. «The business model ontology a proposition in a design science approach». Em: 2004.

[27] Karin Bergquist e John Abeysekera. «Quality function deployment (QFD) — A means for developing usable products». Em: *International Journal of Industrial Ergonomics* 18.4 (1996), pp. 269–275. issn: 0169-8141. doi: https://doi.org/10.1016/0169-8141(95)00051-8. url: https://www.sciencedirect.com/science/article/pii/0169814195000518.

[28] Edna Pacheco Zanlorenci e Robert Carlisle Burnett. «A Abordagem de Engenharia de Requisitos em Software Legado.» Em: *WER*. 2003, pp. 270–284.

[29] Peter Eeles. «Capturing architectural requirements». Em: *IBM Rational developer works* (2005).

[30] Eduardo Figueiredo. «Requisitos Funcionais e Requisitos Não Funcionais». Em: *Icex, Dcc/Ufmg* (2011).

[31] P.B. Kruchten. «The 4+1 View Model of architecture». Em: *IEEE Software* 12.6 (1995), pp. 42–50. doi: 10.1109/52.469759.

[32] John Cheesman. *UML components: A simple process for specifying component-based software*. Jan. de 1970. url: https://www.abebooks.com/9780201708516/UML-Components-Simple-Process-Component-Based-0201708515/plp.

[33] *What is php?* url: https://www.php.net/manual/en/intro-whatis.php.
[34] *About mariadb server*. Set. de 2022. url: https://mariadb.org/about/.
[35] *About mysql*. url: https://www.mysql.com/about/.
[36] *Superheroic JavaScript MVW framework*. url: https://angularjs.org/.
[37] *About*. url: https://git-scm.com/about.
[38] *The One DevOps platform*. url: https://about.gitlab.com/.
[39] *Build software better, together*. url: https://github.com/about.
[40] url: https://www.jenkins.io/doc/.
[41] *About*. url: https://www.sonarqube.org/about/.
[42] *About*. url: https://getlaminas.org/about/.
[43] a Rogue Wave Company Zend. url: https://framework.zend.com/about.1.html.
[44] Michael Henrique Pereira. *AngularJS: Uma abordagem prática e objetiva*. Novatec Editora, 2014.
[45] John Brooke. «SUS: A quick and dirty usability scale». Em: *Usability Eval. Ind.* 189 (nov. de 1995).
[46] *About postman*. url: https://www.postman.com/company/about-postman/.
[47] James Lewis e Jeff Sauro. «Item Benchmarks for the System Usability Scale». Em: 13 (mai. de 2018), pp. 158–167.
[48] Mehrbakhsh Nilashi et al. «An application expert system for evaluating effective factors on trust in B2C WebsitesTrust, security, ANFIS, fuzzy logic, rule based systems, electronic commerce». Em: *Engineering* 03.11 (2011), pp. 1063–1071.

Apêndice A

Metodologia AHP

A.1 Escala Fundamental de Saaty

Intensity of Importance	Definition	Explanation
1	**Equal importance**	**The two components contribute equally to the objective**
2	Weak importance	
3	**Moderate importance**	**Experience and judgement slightly favour one component over the other**
4	Moderate plus importance	
5	**Strong importance**	**Experience and judgement strongly favour one component over the other**
6	Strong plus importance	
7	**Very strong importance**	**One component is favoured very strongly over another; its dominance demonstrated in practice**
8	Very, very strong importance	
9	**Extreme importance**	**The evidence favouring one component over another is of the highest possible order of affirmation**

Figura A.1: Escala Fundamental de Saaty [24]

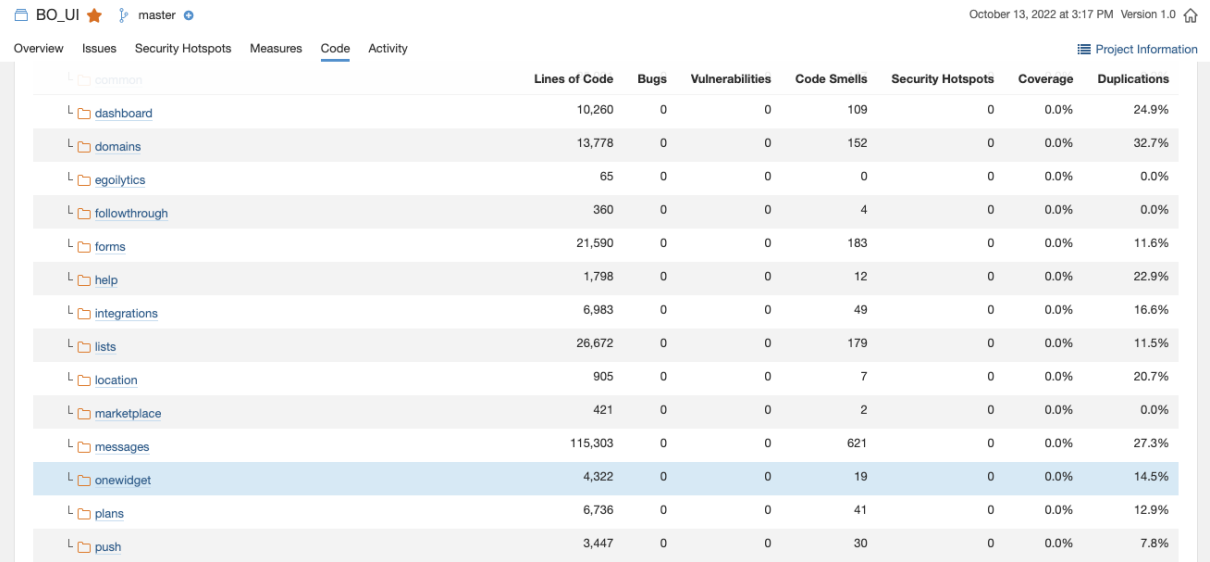
A.2 Valores para o índice de consistência aleatória

n	1	2	3	4	5	6	7	8	9	10
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

Figura A.2: Valores para o IC [48]

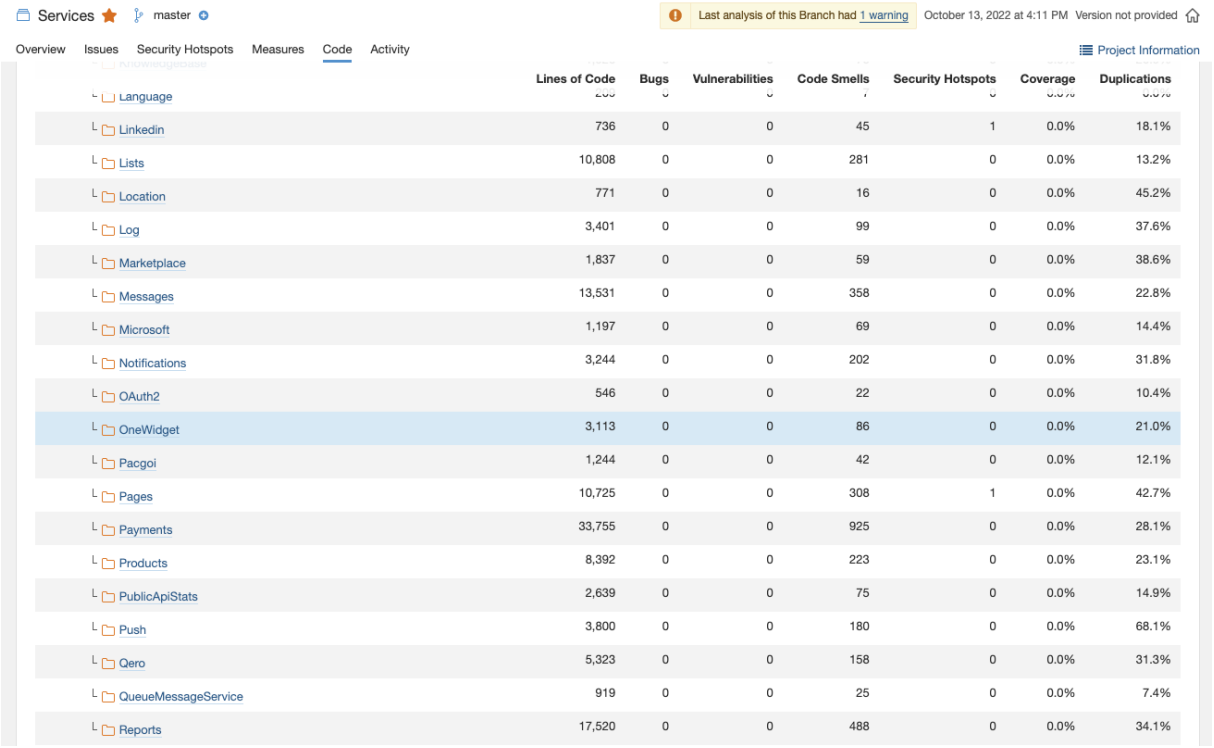
Apêndice B

Resultados do SonarQube - One Widget



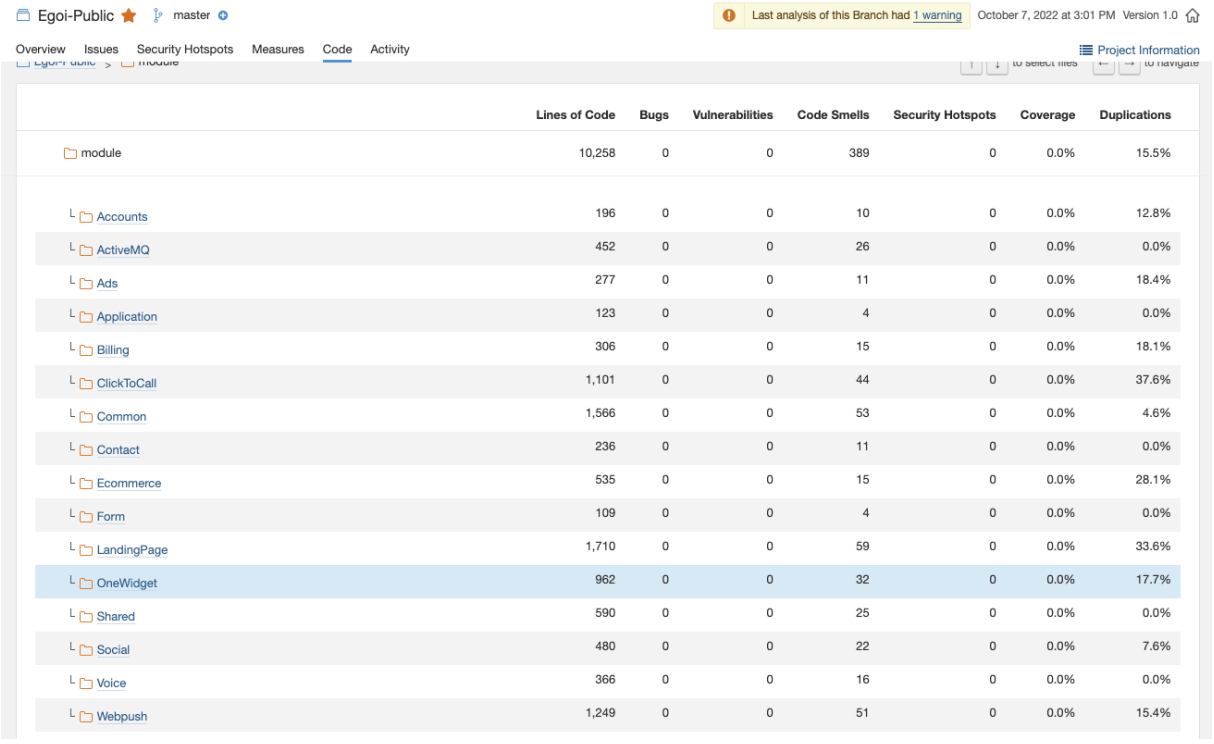
	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
dashboard	10,260	0	0	109	0	0.0%	24.9%
domains	13,778	0	0	152	0	0.0%	32.7%
egoilytics	65	0	0	0	0	0.0%	0.0%
followthrough	360	0	0	4	0	0.0%	0.0%
forms	21,590	0	0	183	0	0.0%	11.6%
help	1,798	0	0	12	0	0.0%	22.9%
integrations	6,983	0	0	49	0	0.0%	16.6%
lists	26,672	0	0	179	0	0.0%	11.5%
location	905	0	0	7	0	0.0%	20.7%
marketplace	421	0	0	2	0	0.0%	0.0%
messages	115,303	0	0	621	0	0.0%	27.3%
onewidget	4,322	0	0	19	0	0.0%	14.5%
plans	6,736	0	0	41	0	0.0%	12.9%
push	3,447	0	0	30	0	0.0%	7.8%

Figura B.1: Resultados SonarQube - One Widget - BO-UI



	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
Language							
Linkedin	736	0	0	45	1	0.0%	18.1%
Lists	10,808	0	0	281	0	0.0%	13.2%
Location	771	0	0	16	0	0.0%	45.2%
Log	3,401	0	0	99	0	0.0%	37.6%
Marketplace	1,837	0	0	59	0	0.0%	38.6%
Messages	13,531	0	0	358	0	0.0%	22.8%
Microsoft	1,197	0	0	69	0	0.0%	14.4%
Notifications	3,244	0	0	202	0	0.0%	31.8%
OAuth2	546	0	0	22	0	0.0%	10.4%
OneWidget	3,113	0	0	86	0	0.0%	21.0%
Pacgoi	1,244	0	0	42	0	0.0%	12.1%
Pages	10,725	0	0	308	1	0.0%	42.7%
Payments	33,755	0	0	925	0	0.0%	28.1%
Products	8,392	0	0	223	0	0.0%	23.1%
PublicApiStats	2,639	0	0	75	0	0.0%	14.9%
Push	3,800	0	0	180	0	0.0%	68.1%
Qero	5,323	0	0	158	0	0.0%	31.3%
QueueMessageService	919	0	0	25	0	0.0%	7.4%
Reports	17,520	0	0	488	0	0.0%	34.1%

Figura B.2: Resultados SonarQube - One Widget - Services



	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
module	10,258	0	0	389	0	0.0%	15.5%
Accounts	196	0	0	10	0	0.0%	12.8%
ActiveMQ	452	0	0	26	0	0.0%	0.0%
Ads	277	0	0	11	0	0.0%	18.4%
Application	123	0	0	4	0	0.0%	0.0%
Billing	306	0	0	15	0	0.0%	18.1%
ClickToCall	1,101	0	0	44	0	0.0%	37.6%
Common	1,566	0	0	53	0	0.0%	4.6%
Contact	236	0	0	11	0	0.0%	0.0%
Ecommerce	535	0	0	15	0	0.0%	28.1%
Form	109	0	0	4	0	0.0%	0.0%
LandingPage	1,710	0	0	59	0	0.0%	33.6%
OneWidget	962	0	0	32	0	0.0%	17.7%
Shared	590	0	0	25	0	0.0%	0.0%
Social	480	0	0	22	0	0.0%	7.6%
Voice	366	0	0	16	0	0.0%	0.0%
Webpush	1,249	0	0	51	0	0.0%	15.4%

Figura B.3: Resultados SonarQube - One Widget - E-goi public

Apêndice C

Teste de Usabilidade

C.1 System Usability Scale de John Brooke

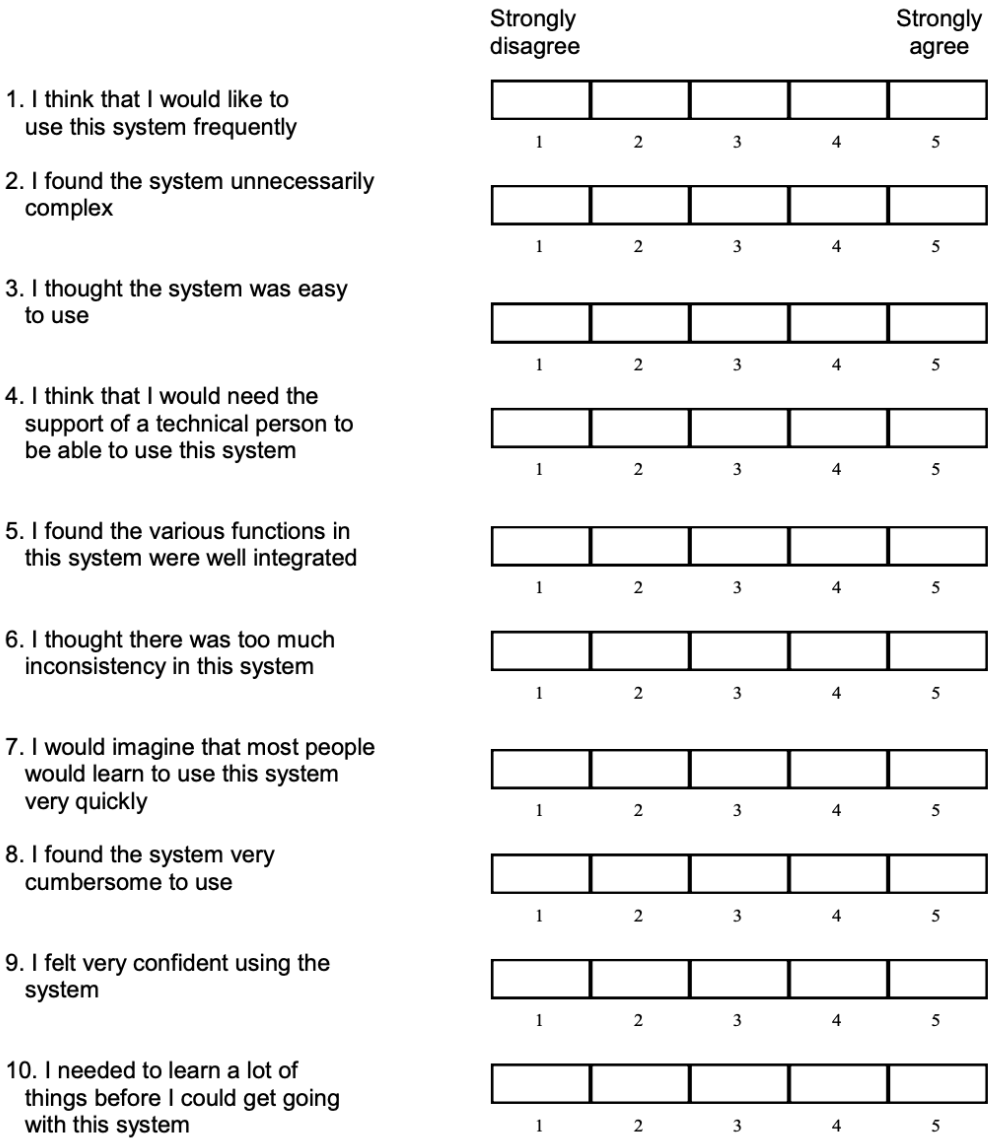


Figura C.1: System Usability Scale

C.2 Tabela de avaliação de usabilidade

Grade	SUS	Percentile range
A+	84.1 - 100	96 - 100
A	80.8 - 84.0	90 - 95
A-	78.9 - 80.7	85 - 89
B+	77.2 - 78.8	80 - 84
B	74.1 - 77.1	70 - 79
B-	72.6 - 74.0	65 - 69
C+	71.1 - 72.5	60 - 64
C	65.0 - 71.0	41 - 59
C-	62.7 - 64.9	35 - 40
D	51.7 - 62.6	15 - 34
F	0 - 51.6	0 - 14

Figura C.2: Tabela de avaliação de usabilidade