INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Novo Canal de Comunicação On-site Messaging

JOSÉ CARLOS TEIXEIRA MARINHO
outubro de 2021

POLITÉCNICO
DO PORTO

Novo Canal de Comunicação On-site Messaging

José Carlos Teixeira Marinho

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

Orientador: Fernando Jorge Ferreira Duarte
Supervisor: Ivo Pereira

Porto, 17 de outubro de 2021

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 Engenharia de Software, no Instituto Superior de Engenharia do Porto.

Este projeto, proposto pela empresa E-goi, tem como objetivo principal criar um novo canal de comunicação *on-site messaging*.

O E-goi é um produto de *marketing* digital multicanal, que permite ao cliente adicionar listas de contactos e enviar campanhas todos os dias, repartidas pelos vários canais de comunicação, dos quais se destacam E-mail, SMS, SmartSMS, Push, WebPush, Post social entre outros.

Nos dias de hoje, os proprietários de páginas *web*, procuram cada vez mais, formas de interagir com os visitantes dos seus *websites*. O *On-site Message* é um canal de comunicação que interage com os visitantes de *websites*, através de mensagens *pop-up*, com objetivo de estimular a navegação no mesmo, aumentando a chance de converter este visitante em cliente.

Atualmente no E-goi não existe nenhum canal de comunicação capaz de criar *on-site messages*. Assim, com este projeto, pretende-se desenvolver um novo canal de comunicação, integrado com o produto existente, que ofereça a possibilidade de criar, configurar e gerir mensagens a disponibilizar no site do cliente.

Palavras-chave: *Marketing* Digital, *Automation*, *On-site messages*

Abstract

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

This project, proposed by E-goi, aims to create a new on-site messaging communication channel.

E-goi is a multi-channel digital marketing product which allows the client to add contact lists and send campaigns every day, spread across several communication channels, including Email, SMS, SmartSMS, Push, WebPush and Social Post, among others.

Nowadays, the owners of web pages are increasingly looking for ways to interact with visitors to their websites. On-site messaging is a communication channel which interacts with website visitors through pop-up messages, with the aim of stimulating their browsing, increasing the chance of converting this visitor into a customer.

Currently, E-goi has no communication channel capable of creating on-site messages. Thus, with this project, we intend to develop a new communication channel, integrated with the existing product, that offers the possibility to create, configure and manage messages to be made available on the client's site.

Agradecimentos

Ao professor Fernando Jorge Duarte, pelo acompanhamento e conselhos que me deu ao longo do desenvolvimento deste documento.

À E-goi, pelas excelentes condições oferecidas para a realização deste projeto.

Ao engenheiro Ivo Pereira, pelos conselhos e pelas reuniões frequentes sobre o ponto de situação do documento.

Ao engenheiro Gonçalo Reais, pela ajuda prestada ao longo de todo o processo de desenvolvimento deste projeto, por mais do que um colega de trabalho, ser um amigo.

Aos meus amigos e, em especial, à minha família, pela força, motivação e apoio.

A todos um muito obrigado, sem vocês não teria sido possível.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Código

Capítulo 1

Introdução

1.1 Enquadramento/Contexto

O presente relatório foi elaborado no âmbito da unidade curricular de Tese / Dissertação / Estágio do Mestrado em Engenharia Informática, na área de especialização de Engenharia de Software, do Instituto Superior de Engenharia do Porto (ISEP). O projeto desenvolvido ocorreu a partir de uma proposta de estágio da empresa E-goi, com o principal objetivo de criar um novo canal de comunicação on-site messaging.

A plataforma de automação de *marketing* multicanal da E-goi envia milhares de campanhas todos os dias, repartidas pelos vários canais de comunicação, dos quais se destacam E-mail, SMS, SmartSMS, *Push*, *WebPush*, *Post* Social entre outros. O cliente moderno procura sobretudo serviços onde não há dependência de perfis, como *designer* ou programador, e que facilitem a integração de serviços no seu *website*. Na sua camada *Web*, o produto E-goi, para além de disponibilizar *landing pages*, disponibiliza formulários e *pop-ups* que podem ser embebidos nos *sites* dos clientes com o objetivo de captar contactos. Uma *landing page* é uma página feita para captar os dados do visitante por via dum formulário. Alguns clientes da E-goi, pretendem interagir com os seus clientes através de mensagens/campanhas que possam ser apresentadas no *site* durante a navegabilidade no mesmo, coisa que ainda não é possível utilizando o E-goi. Surgiu assim a necessidade e a oportunidade da criação de um novo canal de comunicação, o *On-site Message*. As *On-site Messages* superam o desempenho de *banners* nativos e ajudam empresas a guiar a navegação dos seus utilizadores para criar mais oportunidades de vendas [1]. Por este motivo, esta é uma funcionalidade fundamental a ser implementada no produto da E-goi.

1.2 Apresentação da Empresa

Fundada em 2008 por Miguel Gonçalves, a E-goi é uma empresa de *marketing* digital sediada em Matosinhos, Portugal. A empresa tem vindo a crescer bastante, muito fruto do trabalho que os mais de 100 colaboradores produzem diariamente. Opera em vários mercados, desde Portugal até à América Latina, contando com mais de quinhentas mil contas ativas na plataforma E-goi, o que a faz líder entre as várias multinacionais que abraçaram o movimento *phydigital* (união entre o mundo físico e digital) [2]. A sua missão consiste na criação de soluções de comunicação digital, acessíveis a todos os profissionais de *marketing* do mundo.

O produto E-goi consiste num SaaS (*Software as a Service*) de Automação de *Marketing* Multicanal. Oferece diversos canais, desde *email marketing*, SMS *marketing*, campanhas de voz, notificações *web* e *mobile push*, *landing pages*, entre outros, através dos quais os seus clientes podem fazer uma gestão eficiente do *marketing* das suas empresas. O E-goi

possui ainda integrações com várias ferramentas, tais como Shpoify, Facebook, Wordpress, Woocomerce, entre outras. Tudo isto é disponibilizado através da plataforma E-goi, onde o cliente pode subscrever vários planos, desde o Social One que é gratuito e fornece funcionalidades mais básicas, até ao Corporate, que tem todas as funcionalidades disponíveis, indicado para empresas de referência.

1.3 Motivação e Problema

O principal problema abordado nesta dissertação surge da necessidade dos clientes E-goi conseguirem interagir com os seus clientes através de mensagens e campanhas, que possam ser apresentadas na sua página web, no decorrer da navegação na mesma.

Atualmente no E-goi, os clientes apenas conseguem interagir com os seus clientes na sua página web através de *landing pages* ou formulários *pop-up*, por exemplo enviando mensagens ou até mesmo campanhas com códigos promocionais. Analisando o mercado, é possível aferir que muitas empresas concorrentes permitem aos seus clientes a criação de mensagens e campanhas, sem a obrigatoriedade de nelas conter formulários. É possível classificar este, como sendo um problema funcional, relacionado com a falta de um canal de comunicação, capaz de responder a estas funcionalidades. O produto E-goi possui uma grande variedade de canais de comunicação, que têm vindo a ser desenvolvidos ao longo do tempo pela empresa, respondendo sempre às necessidades dos seus clientes.

Outro problema existente, passa pela falta de uma ferramenta capaz de rastrear o comportamento de um visitante num *website* de um cliente e, consequentemente, despoletar mensagens *pop-up* (*on-site messages*), consoante esse comportamento na página *web*. Os canais atuais existentes no E-goi, apenas permitem despoletar *pop-ups* na forma de formulários.

Em suma, é possível listar os seguintes problemas em relação ao produto E-goi:

- Falta de um canal do comunicação *on-site messaging*;
- Falta de uma ferramenta capaz de despoletar *pop-ups*, consoante o comportamento de um visitante numa página *web*.

A identificação de problemas é fundamental para o desenvolvimento de um projeto e deve ser analisado cada um deles detalhadamente, de forma a responder a cada um positivamente.

1.4 Objetivos

O objetivo do projeto é implementar uma solução integrada na plataforma E-goi, criando um canal de comunicação que permita ao cliente criar, editar e gerir *onsite messages*.

Até atingir este objetivo final, várias tarefas foram definidas, o que é fundamental para tornar o trabalho mais eficiente e garantir que o objetivo final é cumprido:

- Análise e desenvolvimento de ferramentas para a criação, edição e gestão de mensagens. Para cumprir este objetivo, é necessário começar por analisar e estudar a concorrência, de modo a perceber que ferramentas existem atualmente e se estas são aplicáveis ao projeto;

- Modelação de arquitetura robusta e escalável de modo a poder ser adaptável a diferentes formas de distribuição de mensagens no cliente (pop-ups, banners, full-screen, entre outros);
- Implementação de funcionalidades específicas, tais como diferentes tipos de *display* e diferentes tipos de *triggers*. Devem ser definidas as funcionalidades fundamentais para o cliente e posteriormente implementar as mesmas.

1.5 Estrutura do Documento

Este documento encontra-se dividido em seis capítulos, sendo eles a Introdução, Contextualização, Estado da Arte, Análise de Valor, Design da Solução e Experimentação e Avaliação. Cada um destes capítulos é referente a uma parte específica do desenvolvimento do projeto.

O primeiro Capítulo divide-se em cinco secções: Enquadramento/Contexto, Apresentação da Empresa, Motivação, Objetivos e Estrutura do Documento. Este capítulo tem por objetivo apresentar, contextualizar e esclarecer o problema.

O segundo Capítulo é dividido em duas secções principais, a Contextualização e o Estado da Arte. Na Contextualização são apresentados temas relativos ao projeto, necessários para compreender todo o documento. No Estado da Arte é feita uma análise ao produto E-goi e são apresentadas várias soluções existentes para a criação de *on-site messages*, seguido-se uma análise comparativa entre elas.

O terceiro Capítulo diz respeito à Análise de Valor, onde é feita uma análise que confirma a necessidade do desenvolvimento do projeto, assim como os moldes como este será desenvolvido.

O quarto Capítulo apresenta o Desenho da Solução. Neste capítulo é feita uma análise aos atuais padrões do produto E-goi e de seguida são apresentados os artefactos que definem a solução, que servirão de base para a implementação da mesma.

O quinto Capítulo apresenta os pormenores técnicos relevantes, assim como a metodologia de desenvolvimento aplicada à implementação do projeto.

O sexto Capítulo diz respeito à Experimentação e Avaliação. Neste capítulo são apresentados e justificados os testes a implementar para avaliar a solução, assim como as grandezas a medir nas experiências definidas.

O sétimo e último Capítulo deste documento é a Conclusão. Neste Capítulo são refletidos os objetivos cumpridos e também o trabalho futuro a desenvolver para a continuação do projeto.

Capítulo 2

Contexto e Estado da Arte

Para uma melhor compreensão do projeto, é importante perceber os conceitos associados ao mesmo, pois só assim é possível explicar decisões tomadas. É igualmente necessário fazer um estudo do estado da arte, de modo a aprofundar o conhecimento em relação a uma área em específico e perceber os padrões utilizados e as soluções implementadas.

No âmbito desta dissertação, é inicialmente apresentada uma contextualização, relativamente à importância da comunicação *on-site*, seguido da apresentação do Matomo, que é a ferramenta que servirá de auxílio ao desenvolvimento deste projeto. Depois é apresentado um estudo relativamente ao produto E-goi e ainda uma avaliação de soluções concorrentes existentes, concluída com uma análise comparativa entre elas.

2.1 A importância da comunicação on-site

A comunicação é uma necessidade básica do ser humano, do homem social [3], é o processo de transmitir ideias entre indivíduos. O *marketing* omnicanal oferece aos clientes informações, produtos, serviços e suporte simultaneamente, por meio de dois ou mais canais de distribuição sincronizados de uma maneira contínua [4].

As empresas têm-se aproveitado das ferramentas digitais de *marketing* como uma maneira de criar vínculo, estabelecer comunicação e interação de maneira direta e rápida na mensagem partilhada com o público de interesse [5]. Os canais de comunicação auxiliam na construção e fixação de uma marca com os seus consumidores, intensificando as vendas e contribuindo para a compreensão do comportamento do público alvo. Através de canais de comunicação, as marcas constroem um relacionamento saudável com os seus clientes.

Atualmente, existem vários canais de comunicação pelos quais as empresas podem comunicar com os seus clientes. Entre todos os existentes, é possível destacar alguns deles como sendo os mais utilizados:

- Redes sociais: utilizadas como estratégia para afunilar o relacionamento com o cliente, pois permitem interação direta com o público, além de aumentar o aliciamento dele com a marca;
- Blogs: mais utilizado para disponibilizar informações educativas sobre diversos assuntos. Com os blogs, é possível construir um ciclo de publicações e criar novas maneiras de aproximação com a audiência;
- E-mail marketing: permite que a mensagem seja entregue de maneira personalizada para uma base de *leads* e clientes (este é um dos canais que o produto E-goi possui);

- Suporte: um suporte de qualidade é capaz de auxiliar e resolver os problemas de um cliente de um cliente, o que pode converter esse cliente num apoiante da marca.

On-site Messages

O *On-site Message* é um canal que pode ser utilizado por proprietários de *websites* para estes comunicarem com os seus visitantes quando estes entram no *site*. Este canal interage com visitantes que já estão na página, estimulando a navegação e podendo até direcionar o visitante exatamente para a página que o proprietário determina, aumentando as hipóteses de conversão deste visitante em cliente. Uma *on-site message* funciona como um *pop-up*, que se abre na página *web* consoante o comportamento do utilizador nela.

Na Figura 2.1, é possível ver um exemplo de como as *on-site messages* podem ser utilizadas, neste caso pela organização IKEA, que é especializada na venda de móveis domésticos.

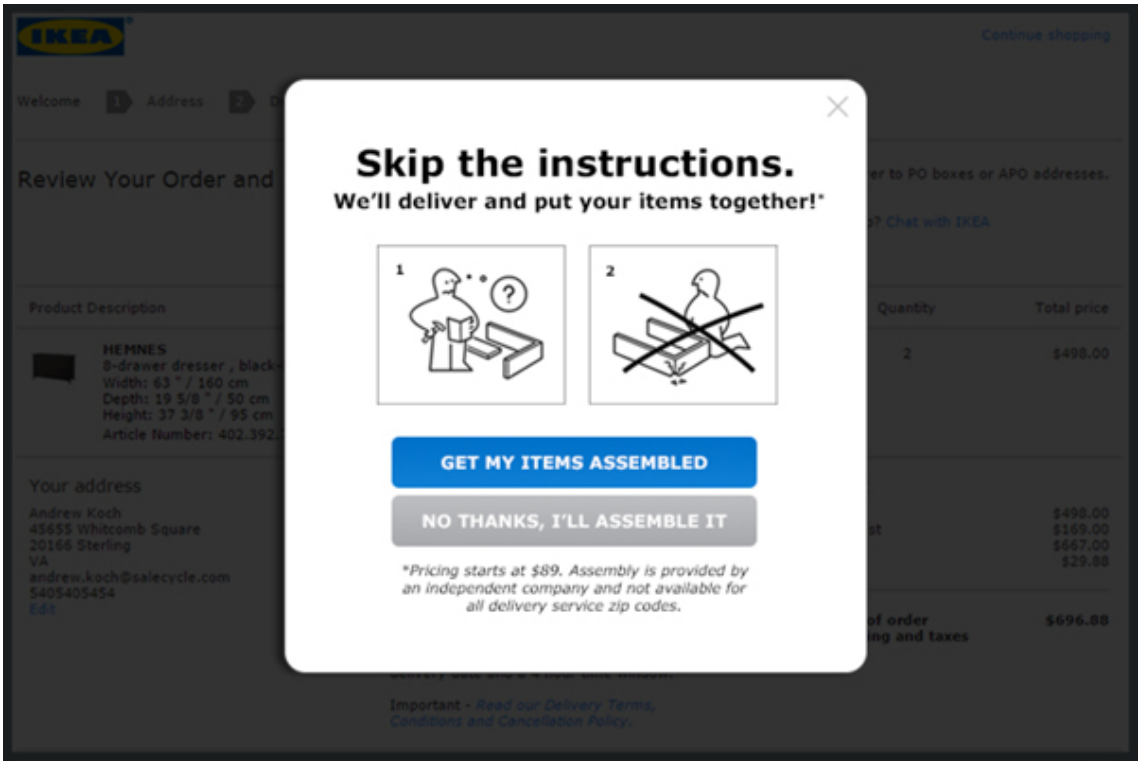

Figura 2.1: On-site Message na IKEA [6]

O facto de ter de montar os móveis em casa é suficiente para fazer alguns visitantes quererem abandonar a compra. Identificando isso como uma oportunidade de venda complementar perfeita, a IKEA pergunta aos cliente, antes do pagamento, se eles gostariam de ter os seus móveis montados após a entrega. Através de uma ilustração atrativa na mensagem, aludindo ao que pode surgir após a compra, a IKEA subtilmente destaca aos seus clientes que a despesa adicional pode diminuir um pouco o impacto da compra. Esta mensagem, útil e mostrada no local certo, poderá com certeza salvar alguns relacionamentos entre a organização e os seus consumidores.

Tipos de display

Há várias maneiras de apresentar as mensagens ao utilizador. Na Figura 2.2 são apresentadas alguns tipos de *display*, que vão desde notificação, *downbar, modal, sidepannel, corner* até *fullscreen cover*.

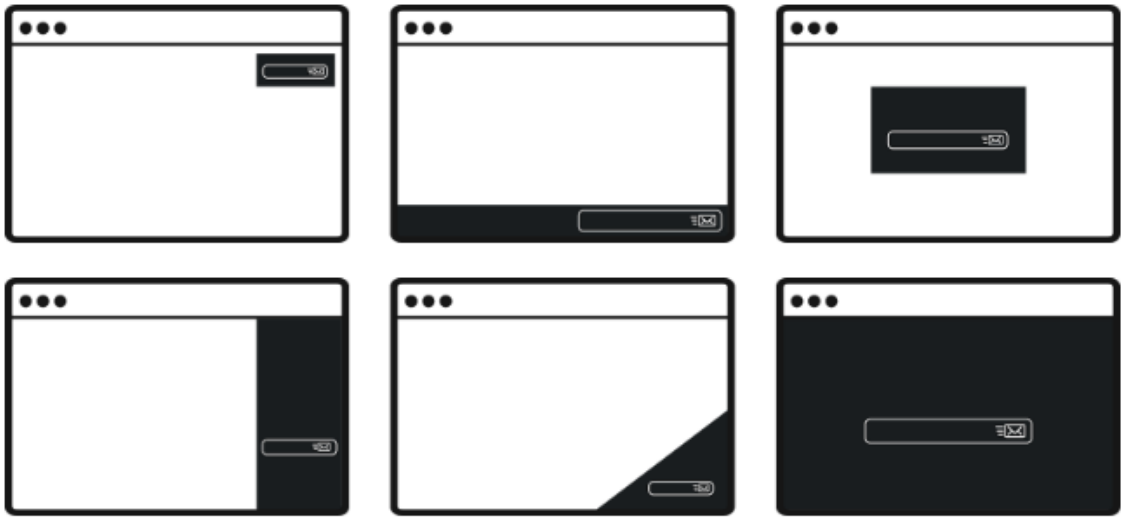

Figura 2.2: Tipos de display de on-site messages [7]

A melhor opção de exibição da mensagem, geralmente, será baseada na meta de conversão do proprietário da página *web*. Seja usando mensagens no *site* para fazer um anúncio, dar uma ajuda aos visitantes ou garantir uma venda, a sua intenção é sempre melhorar a experiência *online* do cliente e transformar mais visitantes em clientes. No exemplo da imagem 2.1, por ter algo que pode impedir a compra, é usada uma modal que impede o utilizador de efetuar outra ação. No caso de ser algo menos importante, talvez um *banner* com um aviso fosse suficiente.

2.2 Matomo

Da necessidade de uma ferramenta capaz de despoletar *pop-ups* consoante o comportamento de um visitante numa página, surge o Matomo [8]. O Matomo é uma ferramenta de análise de dados, criada em 2007, por Matthieu Aubry. Possui uma plataforma *open source web analytics*, usada por mais de 1,4 milhões de *websites* em mais de 190 países e acessível em mais de 50 idiomas. O objetivo desta ferramenta é fornecer aos utilizadores total controlo sobre os dados das suas páginas *web*. É uma alernativa de código aberto ao Google Analytics [9], igualmente poderosa e também respeitadora dos dados dos seus utilizadores. Atualmente no E-goi, o Matomo já utilizado para recolher dados sobre o comportamento dos visitantes das campanhas dos clientes E-goi. Quando um cliente envia uma campanha aos seus contactos com um link para o seu *site*, consegue saber as páginas que esses conctatos visitaram e o que fizeram nelas (por exemplo, se adicionaram um produto ao carrinho, se fizeram download de um documento, etc.).

Matomo Tag Manager

O Matomo Tag Manager é um *plugin* do Matomo que, semelhante ao modo como um CMS (Sistema de Gestão de Conteúdo) oferece flexibilidade para publicar conteúdo num *website*,

permite aos seus utilizadores incorporarem facilmente recursos de aplicações de terceiros no seu *website*, através de um Sistema de Gestão de Tags. Um gestor de tags permite gerir e unificar todos os *snippets* de rastreamento de *marketing* num só sítio. Esses *snippets* são normalmente código JavaScript ou HTML e permitem integrar vários recursos num *website* com apenas alguns cliques. Isto pode ser conseguido utilizando os seguintes componentes principais:

- Tags - Um pedaço de código (tipicamente JavaScript ou HTML) que pode ser executado no *site*. Na maioria das vezes, uma tag pode ser usada para enviar dados a terceiros (por exemplo, rastreamento de dados) ou para incorporar conteúdo de terceiros no *site* (por exemplo, *widgets* sociais ou pesquisas);
- *Triggers* - Define quando uma tag deve ser despoletada;
- Variáveis - Permite obter dados que podem ser utilizados em tags e *triggers*.

Ao criar uma *on-site message*, é criada uma tag que contém o HTML que é esperado que seja exibido. Existem vários *triggers*, como visualização, clique e outros (abordados na próxima secção), que permitem que ele seja exibido e que exiba a informação pretendida pelo utilizador.

O Matomo Tag Manager será utilizado neste projeto, como meio de rastreamento do comportamento de um visitante de um determinado *website*, despoletando *triggers* consoante esse comportamento e mostrando a *on-site message* correspondente.

2.2.1 Tipos de triggers

Uma das vantagens de recorrer ao Matomo como ferramenta de auxílio, é a quantidade de *triggers* que são possíveis definir para o despoletamento de uma tag. Nesta subsecção são descritos cada um deles.

Clique

Ativado quando há um clique num determinado elemento. Este *trigger* permite definir um elemento onde, quando o visitante o clica, é apresentado um *pop-up*

Alteração do histórico

Despoletado quando o URL atual é alterado. Este *trigger* permite definir um URL onde, quando o visitante o acede, é apresentado um *pop-up*.

Intenção de saída

Este *trigger* é despoletado quando o utilizador, possivelmente, está prestes a deixar o *site* e desloca o rato para fora da página *web* atual, por exemplo, em direção à caixa do endereço do navegador ou quaisquer outros botões. Isto pode ser útil para manter o visitante interessado e não perder o visitante, por exemplo, apresentando um *pop-up* de saída.

Deslocamento atingido

Este *trigger* é despoletado quando um utilizador faz um deslocamento ou redimensiona a janela atual do navegador. O *trigger* não é despoletado quando o utilizador faz um deslocamento dentro de um determinado elemento. É possível definir uma percentagem de deslocamento na página e quando o visitante atinge essa percentagem na página, é apresentando um *pop-up*.

Janela carregada

Este estágio de carregamento de uma página *web*, refere-se a quando a página está completamente carregada, de acordo com o navegador. Na maioria dos casos, isto significa que as imagens também estão carregadas e todos os estilos estão aplicados.

Evento personalizado

Permite que os programadores definam manualmente, quando este *trigger* deve ser despoletado, publicando um evento na camada de dados. Desta forma, por exemplo, pode executar determinadas ações quando um produto é adicionado ao carrinho ou quando um utilizador se autentica.

2.2.2 Prova de conceito

Com o objetivo de garantir que o Matomo é uma ferramenta viável para ter como solução, para um dos problemas deste projeto, foi realizada uma prova de conceito. Uma prova de conceito é um "conjunto de ações que permitem demonstrar que um produto ou projeto irá funcionar da forma pretendida, de modo a avaliar se é viável ou não a sua utilização ou comercialização"[10].

Para a realização desta prova, recorrendo ao Matomo Tag Manager, foram primeiramente criados todos os tipos de *trigger* mencionados na Secção 2.2.1, tal como é possível visualizar na Figura 2.3.

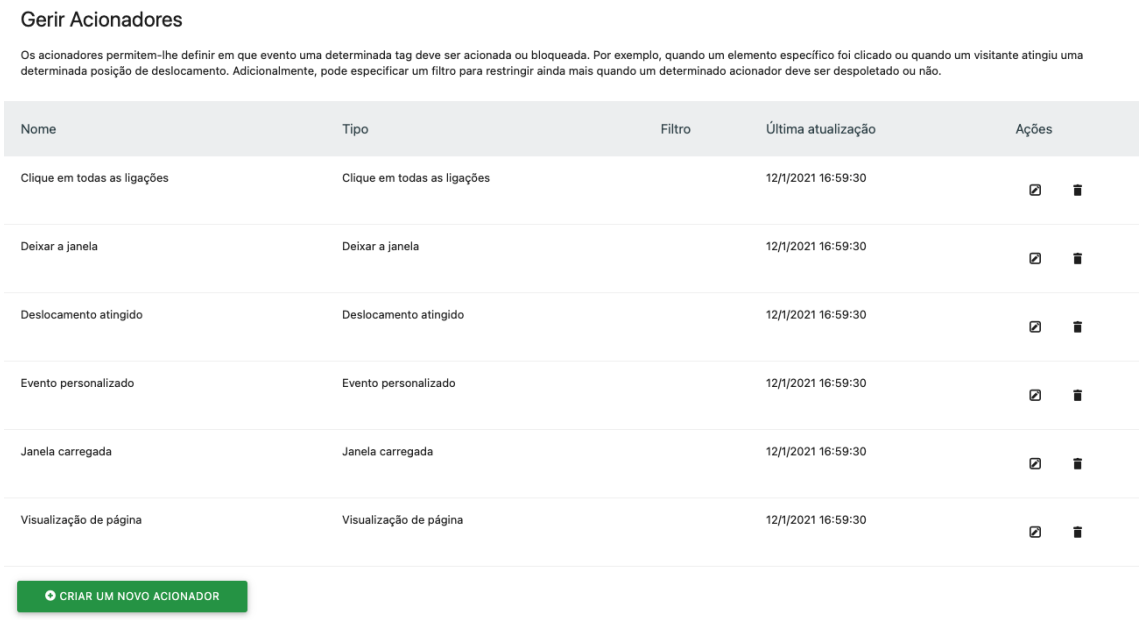

Figura 2.3: Triggers Matomo

De seguida, foi criada uma *tag*, com código HTML a ser executado no *site*, responsável por mostrar a mensagem na página, consoante o tipo de *trigger* que lhe é associado. Para testar o aparecimento da mensagem, foi utilizado um *website* em ambiente de testes. Na Figura 2.4, é possível analisar o resultado final desta prova de conceito.

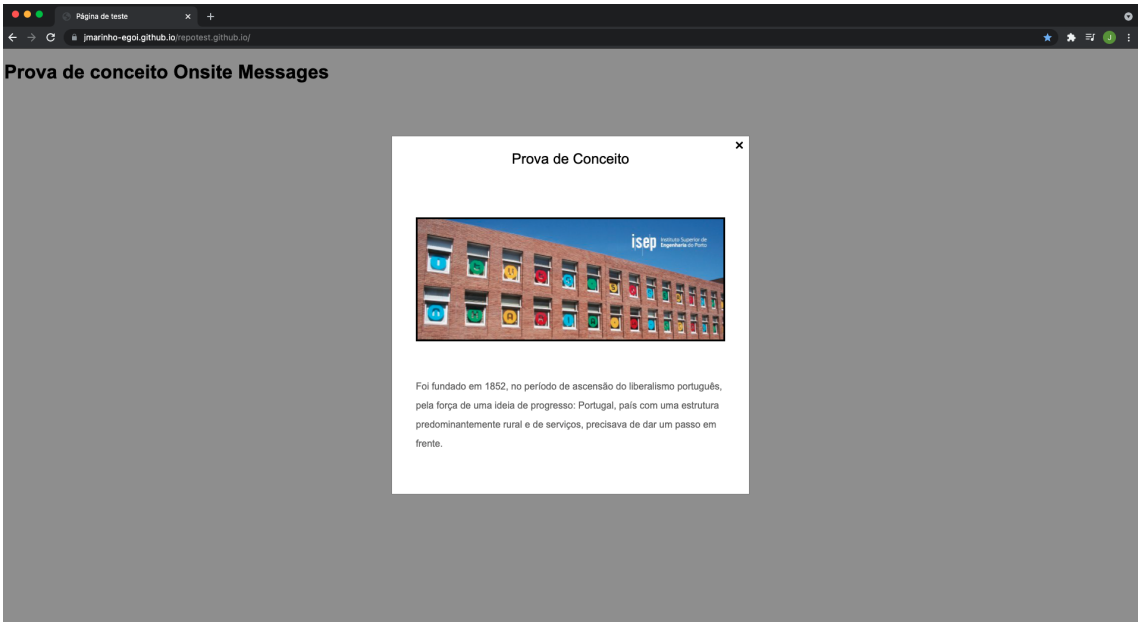

Figura 2.4: Prova de conceito numa página web

O Matomo disponibiliza ainda documentação de suporte onde lista todas as funções que podem ser chamadas e documenta os parâmetros e links para exemplos de cada chamada nos vários formatos [11].

Assim, é possível concluir que o Matomo é uma ferramenta viável a ser utilizada como solução para resolver a falta de uma ferramenta capaz de despoletar *pop-ups*, consoante o comportamento de um visitante numa página web. A mensagem aparece em formato *pop-up* na página *web*, após o *trigger* definido na *tag* ser despoletado, tal como esperado. O facto de o Matomo já ser utilizado no E-goi com resultados positivos, viabiliza ainda mais a escolha desta ferramenta para resolver o problema em questão.

2.3 Análise do E-goi

O E-goi é um produto bastante complexo, em constante desenvolvimento e evolução. A sua solução utiliza várias tecnologias, tais como PHP, Python, Angular, Javascript, HTML, CSS, entre outras. Dependendo dos objetivos de cada componente, são aplicadas as tecnologias mais apropriadas.

Easygoi

O Easygoi é a ferramenta mais antiga do E-goi para criação e edição de formulários *pop-up*. Esta ferramenta está desenvolvida em jQuery, utilizando JavaScript, PHP, HTML e CSS. Com o Easygoi é possível estruturar o formulário através de um editor *drag and drop*, onde é permitida a utilização de diversos *widgets*, desde *input* de data, email, texto, número telefónico, escolha de opções (*multiselect*), *upload* de ficheiro, assim como título, imagem, vídeo, botões, entre outros, tal como se pode ver na Figura 2.5.

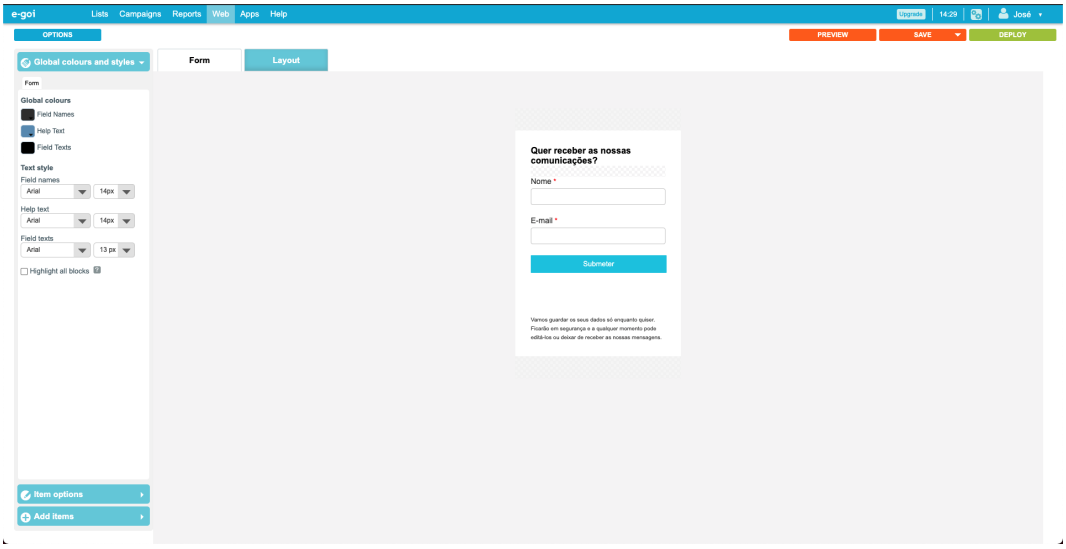

Figura 2.5: Editor Easygoi

Relativamente às condições de aparecimento do formulário, é possível configurar:

- Posição (centro, cimo, direita, fundo, esquerda);
- Momento em que o formulário aparecerá (ao ver o cimo da página, ao passar o meio da página, ao chegar ao fundo da página e ainda é possível personalizar relativamente à percentagem de ocupação na página);
- Intervalo de tempo (em segundos) antes de aparecer;
- Voltar a mostrar o formulário se o visitante fechar o mesmo (são configuráveis os números de dias até voltar a apresentar).

No final, é gerado um código HTML (http e https) com um *script* pronto a ser inserido na página do cliente.

Email Builder

O Email Builder é uma ferramenta do E-goi que surgiu com base no Easygoi, com o objetivo de disponibilizar ao cliente um criador e editor de *emails*. Veio renovar o fluxo de criação de uma campanha no E-goi, tornando esse processo mais atrativo, mais intuitivo e mais fácil para o cliente.

Posteriormente foi criado um novo builder, o Landing Page Builder, que, migrando algumas funcionalidades do Easygoi e baseado no fluxo de criação do Email Builder, possibilita ao cliente criar as suas *landing/signup pages*. Na Figura 2.6 está representado o editor do Landing Page Builder.

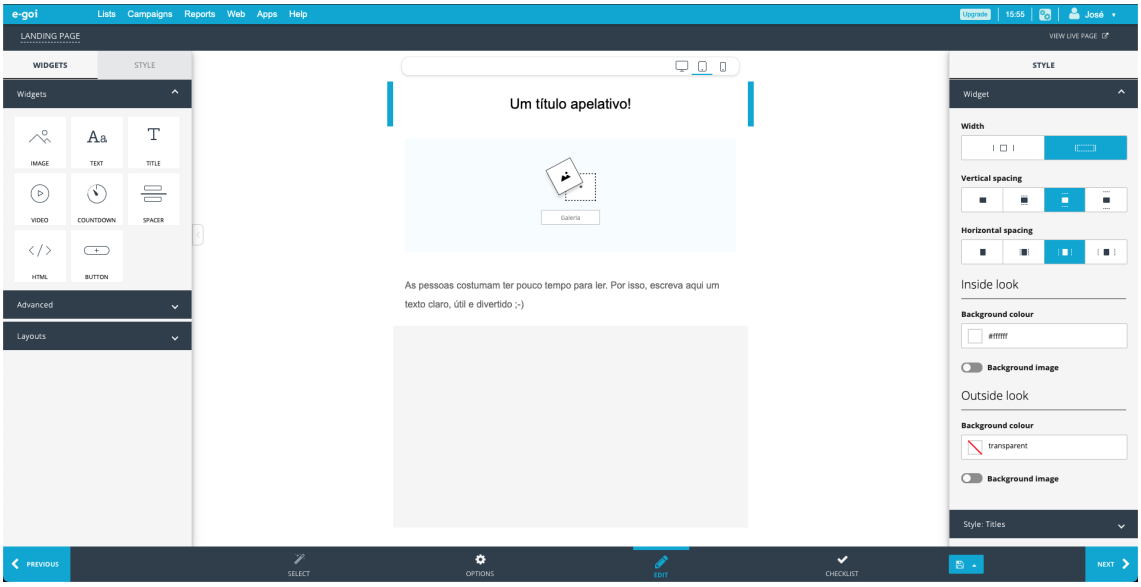

Figura 2.6: Editor Landing Page Builder

Com o criador de Landing Pages do E-goi, o cliente pode facilmente adicionar colunas, elementos de texto, imagens, vídeo e muito mais. Para tal, basta arrastar e soltar os *widgets* para a página (*drang and drop*).

Em comparação com o Easygoi, esta ferramenta utiliza ferramentas mais recentes, que permitem mais facilmente desenvolver novas funcionalidades, tornando o produto mais escalável e mais estável.

O que se pretende para este projeto, é um novo canal de comunicação com a capacidade de criar, editar e configurar *on-site messages*. O fluxo de criação será baseado nos editores já existentes, devido às suas integrações, questões gráficas e maior facilidade de desenvolvimentos. Será possível criar mensagens independentes de formulários, o que até agora não era possível.

2.4 Soluções existentes

No início deste projeto, foi levantado um estudo da concorrência que são plataformas que permitem os seus utilizadores criarem *on-site messages* e adiciona-las ao seu *website*, controlando o aparecimento através de *triggers*.

2.4.1 MoEngage

A empresa MoEngage foi fundada em 2014 por ex-alunos do IIT Kharagpur [12], Raviteja Dodda e Yashwanth Kumar.

A oferta da MoEngage [13] está baseada em 3 planos (*Starter, Growth, Enterprise*). As *on-site messages* estão disponíveis em todos os planos sendo que no plano *Starter* estão limitadas a 10000 clientes ativos por mês. Esta ferramenta permite aos seus clientes criarem várias interações com os seus clientes através do seu *website*, desde campanhas de Email, WebPush, In-app, SMS, Connector, Flow e *On-site Messages*.

As *On-site Messages* apresentam múltiplas opções de *triggers*, desde condicionadas ao carregamento da página, condicionadas ao comportamento do utilizador e condicionadas a comportamento de saída.

Relativamente às mensagens, estas são configuráveis a partir de um editor apresentado do lado direito da página do utilizador, tal como é possível verificar na Figura 2.7. É possível definir o seu título e corpo da mensagem e um endereço de redirecionamento. Pode ainda ser adicionado um ícon ou imagem à mensagem e também botões personalizáveis. As mensagens podem aparecer em diferentes formatos, pop-up, banner ou CTA.

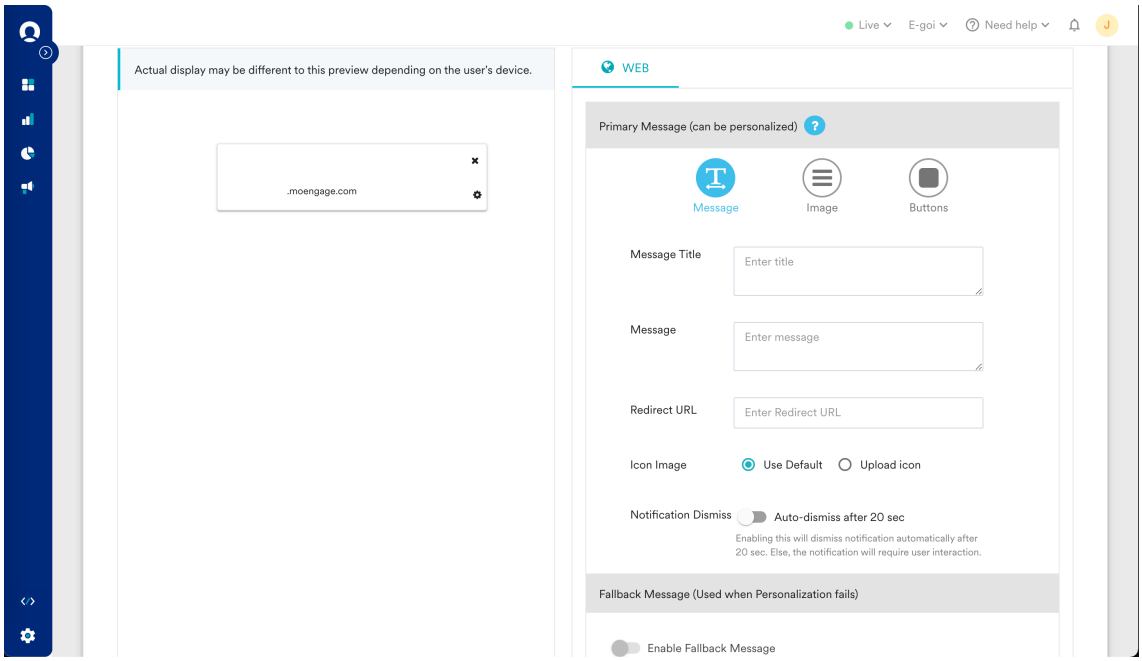

Figura 2.7: On-site Messages no Moengage

Esta ferramenta permite ainda a apresentação da mensagem de forma segmentada. Ou a todos os visitantes, ou a visitantes que cumpram determinadas condições, ou a visitantes que estejam já identificados e segmentados.

Para finalizar, é também possível definir limites, como por exemplo, o intervalo de duração da mensagem, número máximo de vezes que o mesmo cliente vê a mensagem, ou ainda o auto fecho da mensagem, após um tempo definido pelo utilizador.

2.4.2 WisePops

Liderada por Benjamin Cahen, a WisePops consiste numa ferramenta que permite aos seus clientes exibirem uma mensagem a qualquer segmento dos visitantes da sua página *web*, com apenas alguns cliques, sem a necessidade de nenhum *developer* [14]. O WisePops capacita as equipas de *Marketing* e Produto com uma nova maneira simples, escalonável e mensurável de se comunicar no seu *website*.

A oferta da WisePops baseia-se em 3 planos (*Basic, Expert e Pro*). A diferença na variação dos preços, está baseada em *features* e limites, nomeadamente de *sites* disponíveis e *page views*.

A criação de *pop-ups* baseia-se em *templates* que podem ser customizados e adaptados num editor *drag and drop*, desde estilos, posição e conteúdo. Relativamente aos *triggers*, apresentados na Figura 2.8, é possível definir diferentes tipos de *triggers*, relacionados com a entrada/saída do *site*, baseados na posição de *scroll* no *site*, associado a clique num botão específico, associado a *hoover*, ou com objetivos definidos (por exemplo, de URL para URL).

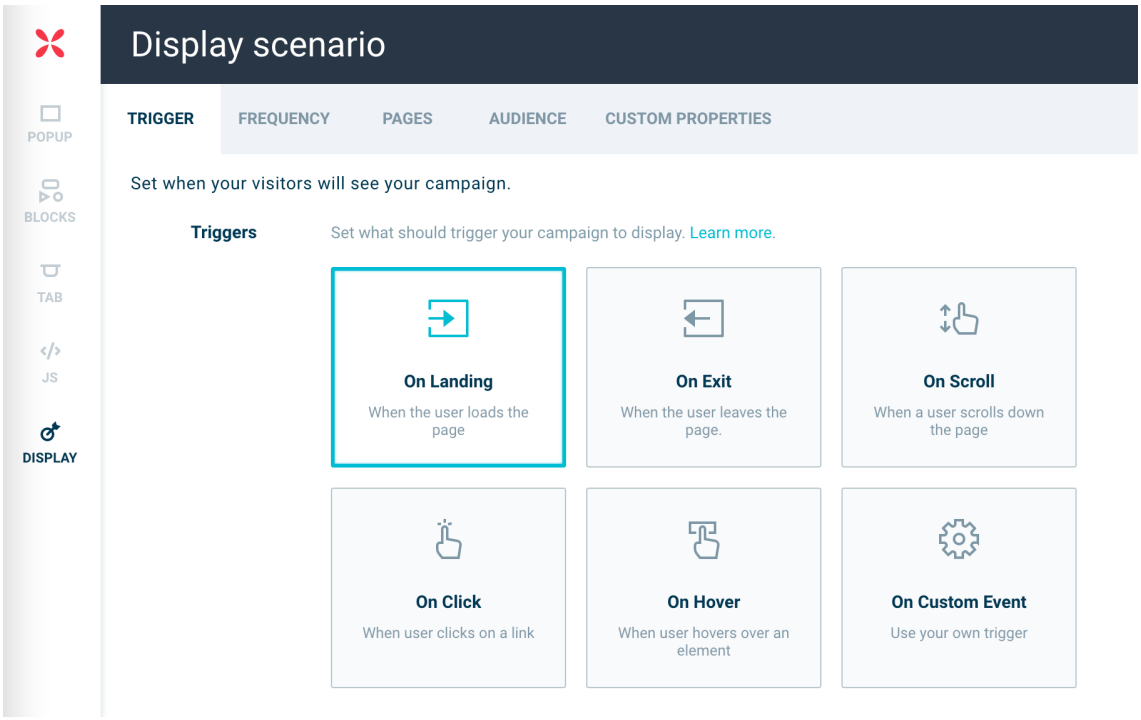

Figura 2.8: On-site Messages na WisePops

É possível segmentar o aparecimento da mensagem com base em diferentes critérios, dispositivo, frequência de visita, tipo de tráfego (como por exemplo determinado UTM, *browser* e localização). É ainda possível determinar diferentes tipos de limites, como por exemplo, as datas em que a mensagem está ativa e o tempo de espera até voltar a apresentar a um mesmo visitante.

2.4.3 GetSiteControl

A GetSiteControl [15] é uma ferramenta que permite a um proprietário de um *site*, adicionar fácil e rapidamente elementos que atraem os visitantes do *site* e ajudam a aumentar as conversões de visitantes em clientes, através de *widgets* inteligentes de otimização. Pertence e é operada pela Getwebcraft Limited [16], uma empresa registada no Chipre. Existem apenas 11 pessoas a trabalhar no desenvolvimento da GetSiteControl, projeto que se iniciou em 2014 e a primeira versão foi lançada em Junho do mesmo ano.

A oferta da GetSiteControl baseia-se em 3 planos (Small, Medium e Large), baseado no número de *views*.

A configuração é bastante simples e rápida. Está baseada na construção do pop-up com recurso a um editor *drag and drop*, partindo da escolha de um *layout* tipo (desde *Modal, Fullscreen, Slide-in, Bar* e *Panel*) e posicionamento na página, seguido da definição do conteúdo da mensagem. Permite também segmentar a apresentação do pop-up e definir

condições de aparecimento (num determinado URL, dependendo da região onde o visitante se encontra, intenção de saída, percentagem de *scroll* na página, inatividade, tempo na página e no site e ainda num determinado período de tempo), tal como é apresentado na Figura 2.9.

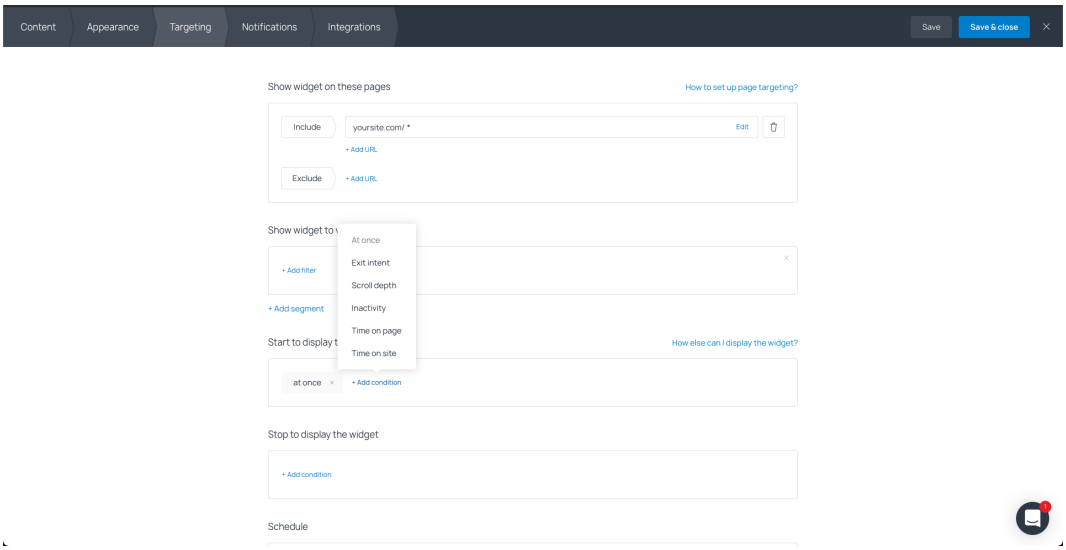

Figura 2.9: On-site Messages na GetSiteControl

2.4.4 HelloBar

Fundada em 2011, a HelloBar [17] é uma empresa sediada em San Diego, Califórnia, nos Estados Unidos da América, especializada na captura de *leads* e otimização da taxa de conversão de visitantes em *websites*.

A HelloBar baseia a sua oferta em 4 planos (Starter, growth, Premium, elite). As diferenças entre plano baseiam-se em *features* e em visitas.

Esta ferramenta permite a criação de diferentes tipos de mensagens *on-site*, desde *hello bars*, *page takeovers*, modais, *sliders*, *alert bells* e *alert icons*. Permite a função de ativar/desativar o *pop-up* sem ser necessário remover o código do *site*.

O primeiro passo para o utilizador criar uma *on-site message*, é definir um objetivo (captar contactos, recuperar carrinhos, enviar mensagem, entre outros). Depois é necessário escolher o formato do *pop-up*, tal como é apresentado na Figura 2.10. É possível editar o contéudo da mensagem num editor *drag and drop*. De seguida, definem-se as condições de *triggering*, que vão desde um tempo definido pelo utilizador, pecentagem de scroll ou um evento costumizado, sendo possível definir limites (após fechar não mostrar novamente antes de um determinado número de dias). No *targeting*, permite ainda segmentação de acordo com a geolocalização, *browser* e *device* (só em *mobile* por exemplo).

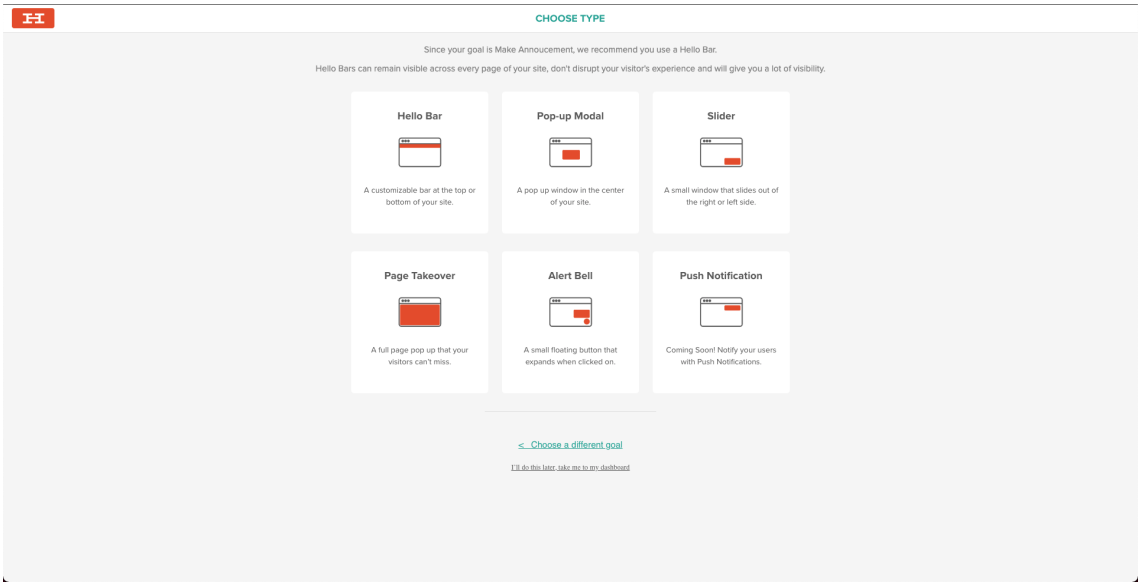

Figura 2.10: On-site Messages na HelloBar

2.4.5 Mailmunch

A Mailmunch [18] é uma empresa Norte Americana, fundada em 2014 e especializada em *Email Marketing*, que ajuda os seus clientes a aumentar a sua base de clientes, fornecendo ferramentas para criar formulários de assinatura opcionais, totalmente personalizáveis.

A oferta da Mailmunch, limita-se a *pop-up forms*, não dispondo de *on-site messaging*. Os formulários podem ser totalmente personalizados nos mínimos detalhes, através de um editor *drag and drop*, incluindo cor, fonte, posicionamento da página e até mesmo o seu tempo de pop-up. É possível criar *pop-ups, slide boxes* e *top bars*, com diferentes tipos de *trigger*, como por exemplo mostrar/não mostrar quando o URL é um determinado definido pelo utilizador, intenção de saída, não mostrar na primeira visita e baseado em tempo de permanência ou frequência de visita. Na Figura 2.11 está apresentado o passo de definição de comoportamento.

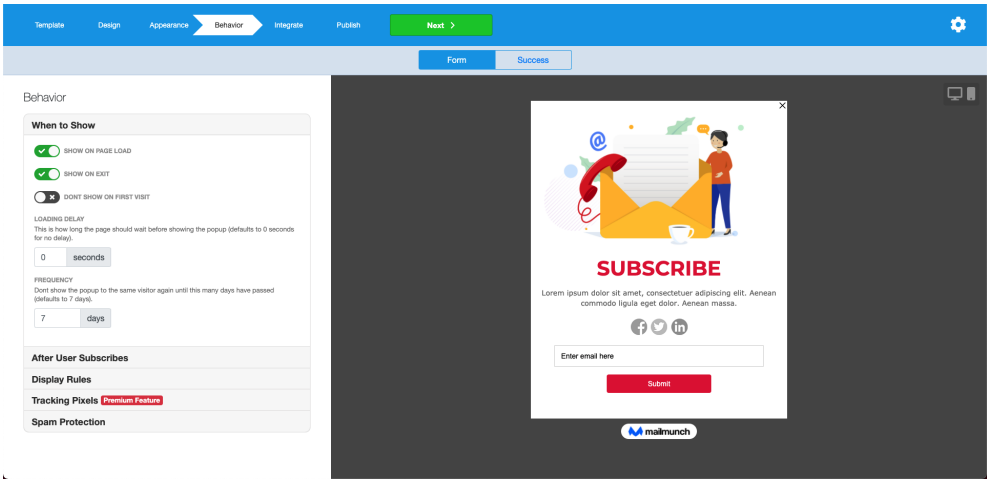

Figura 2.11: On-site Messages na Mailmunch

A Mailmunch pode ser integrada em qualquer *website* e funciona juntamente com todas as principais plataformas de *Email Marketing*.

2.4.6 OptiMonk

A OptiMonk [19] é uma empresa de origem Húngara fundada em 2014, especializada na geração de *leads*, tecnologia de intenção de saída e otimização de conversões. A sua solução permite aos seus utlizadores converterem os visitantes do seu *website* em clientes através de *on-site messages*, apresentadas quando estes mostram intenção de deixar a página.

A oferta da Optimunk baseia-se em 5 planos *(free, essential, growth, premium, master)*. A alteração de planos está baseada em *features* e *pageviews*.

O primeiro passo para criar um *pop-up*, passa por escolher entre diferentes tipos de aparecimento da mensagem, a ideal para o objetivo a alcançar (*pop-up, fullscreen, sidemessage, nanobar, lucky wheel, scratchcard, pick a gift*). São baseados em modelos, que podem ser editados num editor *drag and drop*. Permite a definição de diferentes tipos de *trigger*, que podem ser combinados entre si, tal como é apresentado na Figura 2.12. Alguns *triggers* possíveis são: intenção de saída em *website*, após um evento *JavaScript* predefinido, frequência de visita, tempo de permanência na página, percentagem de *scroll* na página e após *click* numa área específica na página.

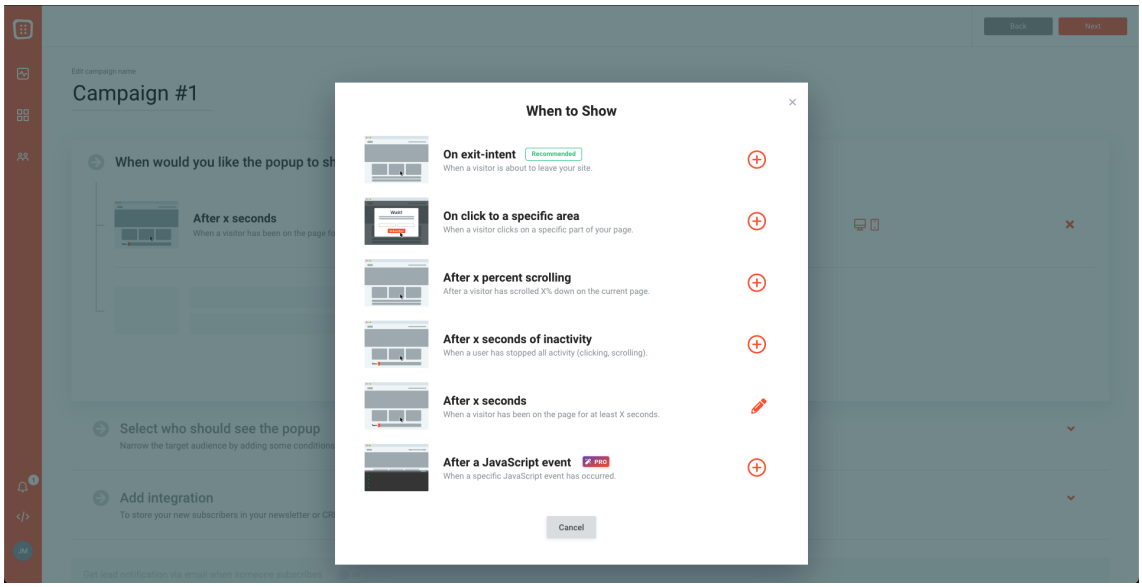

Figura 2.12: On-site Messages na OptiMonk

É ainda possível definir a quem deve ser apresentada a mensagem, com base em regras de estado do carrinho de compras, como por exemplo, valor no carrinho e número de produtos selecionados, se um determinado URL contém algo específico definido pelo utilizador, deteção de ferramentas de *adblock* e ainda de acordo com *cookies*.

2.4.7 Sleeknote

A Sleeknote [20] começou a partir de duas empresas: Twami, fundada por Mogens e Patrick, e a encarnação original da Sleeknote, fundada por Emil e Daniel. Com experiência em *e-commerce* e visão semelhante, as duas empresas uniram-se para um único propósito, ajudar

as marcas de *e-commerce* a converter os visitantes dos seus *sites* em clientes, sem prejudicar a experiência do utilizador.

A oferta da Sleeknote baseia-se em 5 planos (*basic, plus, pro, premium, enterprise*). A diferença de planos está assente em *page views*.

As mensagens são criadas com base em casos de uso específicos. O primeiro passo para criar uma *on-site message*, passa por escolher o tipo de campanha, desde captação de *leads*, anúncios de produtos, apoio ao cliente e questionário de *feedback*. Mediante o caso de uso é dado a escolher ao utilizador o tipo de *display* em que se quer apresentar a mensagem (*slide-in, pop-up* ou *sleek bar* e um *template* predefinido. Tudo isto é personalizável num editor *drang and drop*.

Após definir o aspeto da mensagem, podem ser definidos diferentes triggers, tal como é apresentado na Figura 2.13. São exemplos de *triggers*: tempo de permanência na página, percentagem de *scroll* na página, quando o visitante mostra intenção de saída, quando o visitante clica numa área específica e após um evento *JavaScript* predefinido. Permite ainda alguns *triggers* mais avançados, como UTM, novas visitas, *query detection, referrals*, entre outros.

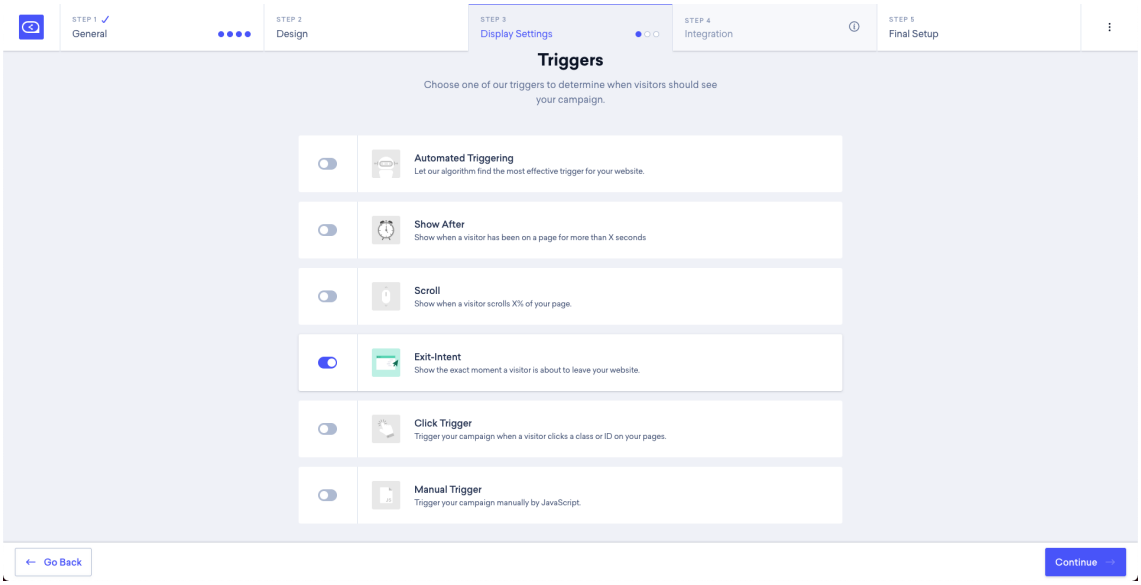

Figura 2.13: On-site Messages na Sleeknote

2.4.8 Optinmonster

Fundada em 2013, a Optinmonster tem crescido ao longo dos anos com um único objetivo: construir uma tecnologia poderosa de nível empresarial para ajudar as empresas a aumentar a sua base de clientes e receitas [21].

A oferta da Optinmonster baseia-se em 4 planos (*basic, plus, pro, growth*). A diferença de planos está assente em upgrade de features, quantidade de *sites* suportados e *page views*.

Toda a oferta de *pop-ups* está associada ao uso de um formulário. Esta ferramenta permite aos seus clientes criarem os seus formulários para campanhas através de um editor *drag and drop*, visualmente otimizados para as taxas de conversão mais altas.

O primeiro passo para a criação de um formulário *pop-up*, é escolher o tipo de *display*, desde *slide-in, sidebar, fullscreen, floating bar*, entre outros, e um *template*. De seguida, é possível configurar diferentes tipos de *triggers*. São exemplo: intenção de saída, *scroll*, inatividade/permanência, tempo na página, agendamento numa determinada data. A Figura 2.14 apresenta exemplos de *triggers*.

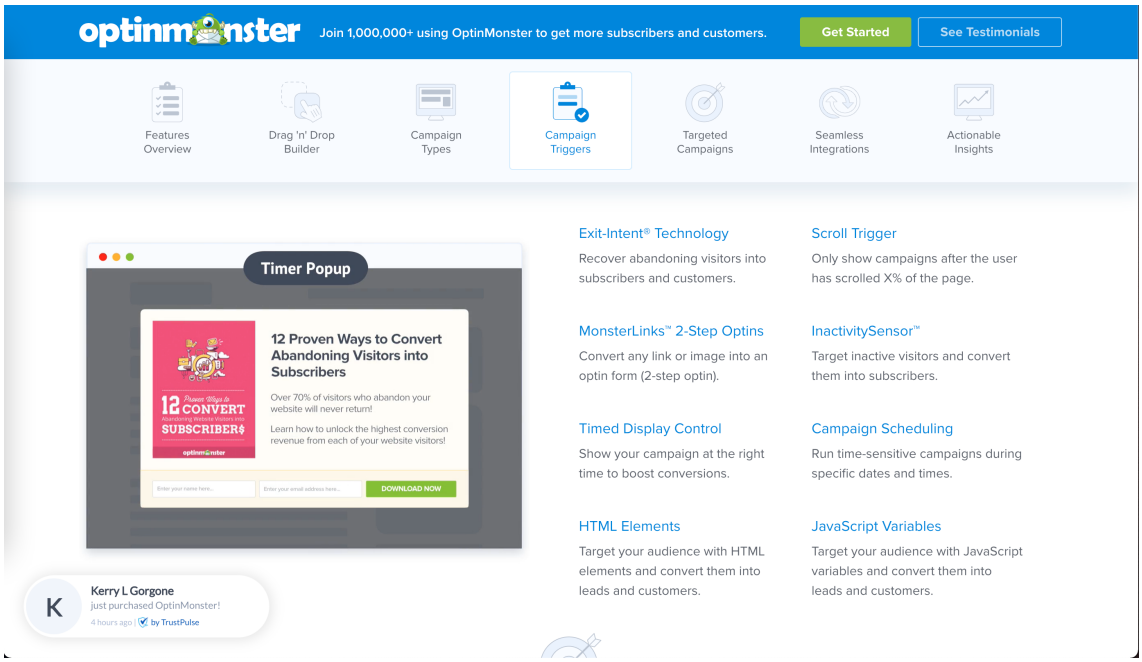

Figura 2.14: On-site Messages na Optinmonster

É ainda possível segmentar a que visitantes do *site* é apresentado o formulário *pop-up* com base em *referral*, geolocalização, *retargeting, adblock detection*, e tipo de *device*.

2.4.9 Socital

A Socital foi fundada em 2016 por Theo Vasileiadis e encontra-se atualmente sediada em Londres. É uma empresa que possui uma ferramenta SaaS para campanhas *on-site*, especializada em *e-commerce* [22]. O seu produto ajuda os prorpietários de lojas online a recolher dados e interagir mais com os seus clientes e transformar visitantes desconhecidos em clientes.

A oferta da Socital baseia-se em 5 planos (*accelarate, growth, Pro, Pro plus* e *corporate*). A diferença de preços está baseada na quantidade de novas *leads* por mês.

Esta ferramenta permite a criação de vários tipos de mensagens *on-site*, desde *Pop-up, Bar, Scroll Box, Slide Up* e *Slide Tab*, totalmente personalizáveis num editor que apresenta todos os widgets disponíveis.

Relativamente ao comportamento da mensagem, permite a definição de diferentes *triggers*, baseados em tempo de permanência na página, percentagem de *scroll* na página e intenção de saída, tal como é apresentado na Figura 2.15.

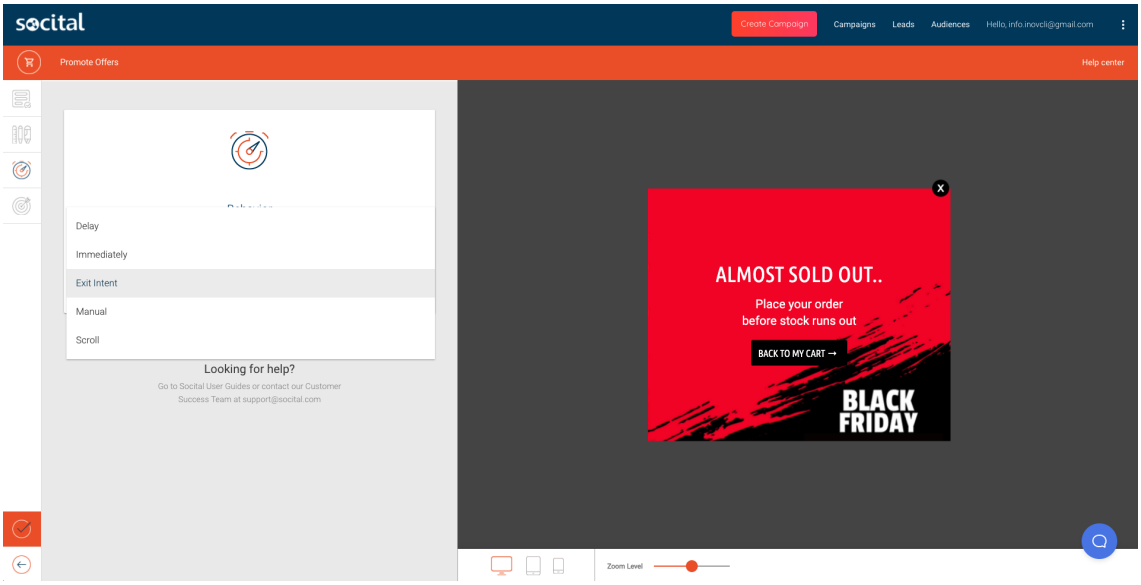

Figura 2.15: On-site Messages na Socital

É possível ainda ter diferentes tipos de *targeting*, como por exemplo, mostrar mensagem a visitantes identificados/primeiros visitantes e visita de um determinado URL.

2.5 Análise comparativa

Depois de uma análise às soluções existentes no mercado, é importante perceber em que aspetos as ferramentas diferem, fazendo-se, para isso, uma análise comparativa.

Analisando as ferramentas, é possível verificar que todas se podem caracterizar pelos seguintes aspetos: planos de subscrição, tipos de aparecimento da mensagem, tipos de *trigger* e ainda tipos de *target*. Para este projeto serão explorados mais ao detalhe os tipos de *trigger*, uma vez que é uma necessidade maior para a organização.

Por este estudo, foi possível determinar os principais tipo de *triggers* utilizados no mercado, sendo eles: intenção de saída, percentagem de *scroll*, com base no URL da página, tempo de permanência na página, clique num determinado elemento, frequência de visita, janela carregada e ainda eventos personalizados. Cada um destes tipos está explicitado no Capítulo 2, na secção de Tipos de Trigger.

Uma vez que o Matomo será a ferramenta que servirá de base ao projeto para rastrear a atividade de um visitante num *website* e responsável por tratar estes *triggers*, é importante verificar que através dela é possível definir todos os tipos de *triggers* identificados.

Todas as ferramentas analisadas são especializadas em *e-commerce* e têm como objetivo captar *leads* e otimizar a taxa de conversão de visitantes de *websites* em clientes, ou seja, ajudar os proprietários de *websites* a interagir com os visitantes dos mesmos e transforma-los em clientes.

Através desta análise foi possível verificar que nem todas as ferramentas possuem o canal de *on-site messaging*. A Mailmunch e a Optinmonster não oferecem aos seus clientes a possibilidade de criar um *pop-up* sem a utilização de um formulário, contudo, possuem tipos de *triggers* fulcrais para análise.

Tabela 2.1: Comparação de triggers entre ferramentas (1/2)

	MoEngage	**WisePops**	**GetSiteControl**	**HelloBar**	**Mailmunch**
Intenção de saída	X	X			X
Percentagem de scroll	X	X	X	X	
Com base no URL			X	X	X
Tempo de permanência			X		X
Clique	X	X	X		
Frequência de visita		X			X
Janela Carregada	X	X			
Evento personalizado	X	X	X	X	

Tabela 2.2: Comparação de triggers entre ferramentas (2/2)

	OptiMonk	**Sleeknote**	**Optinmonster**	**Socital**	**Matomo**
Intenção de saída	X	X	X	X	X
Percentagem de scroll	X	X	X	X	X
Com base no URL	X			X	X
Tempo de permanência	X	X	X	X	X
Clique	X				X
Frequência de visita	X				X
Janela Carregada	X				X
Evento personalizado	X	X	X		X

Para uma mais fácil perceção desta análise comparativa, foram elaboradas duas tabelas onde é possível identificar todos os tipos de *triggers* explorados para cada uma das ferramentas analisadas.

Na primeira coluna estão identificados 8 tipos de *triggers* e na primeira linha todas as ferramentas analisadas. A indicação (X) revelam se a ferramenta possui ou não o tipo de *trigger* em questão.

Através das Tabelas 2.1 e 2.2, é possível perceber que os *triggers* mais utilizados são o de intenção de saída, percentagem de *scroll* e evento personalizado. Isto deve-se ao facto de serem os comportamentos mais habituais de um visitante de um *website*. O *trigger* de tempo de permanência também apresenta uma elevada utilização, justificada pelo mesmo facto.

É possível ainda verificar, que, excluindo o Matomo, a ferramenta que oferece menos possibilidades de escolha aos seus clientes é a HelloBar e, em contrapartida, a que oferece o maior número de *triggers* é a OptiMonk, sendo assim identificada como a principal concorrente. Contudo, através da análise à última coluna da Tabela 2.2, é viável a utilização do Matomo, uma vez que oferece todos os tipos de *triggers* identificados, o que garante que o novo canal de comunicação *on-site messaging* do E-goi oferecerá todas essas escolhas aos seus clientes.

2.6 Estado da Arte em Tecnologias Relevantes

Git

O Git [23], criado por Linus Torvalds, é um dos sistemas de controlo de versões mais utilizados para o desenvolvimento de *software*. Projetado para lidar com tudo, desde pequenos projetos a projetos de grandes dimensões com velocidade e eficiência, o Git permite aos seus utilizadores guardarem os seus ficheiros, adicionando um código diferente para cada versão. Todas as mudanças são feitas localmente e podem ser depois enviadas para o servidor, onde ficam disponíveis para todos os colaboradores do repositório. Utilizando o conceito de *branches*, é possível criar vários ramos independentes entre si a partir de um ramo principal, por exemplo para separar os ficheiros com o código que vai para produção, dos ficheiros com código de desenvolvimento de novas funcionalidades. Para haver uma junção de dois ramos, é feito um *merge*.

Existem diversas maneiras de utilizar o sistema Git, utilizando ferramentas como GitLab, GitHub ou BitBucket. A utilização de Git é referido na Secção 5.1.

Integração Contínua

A integração contínua é uma prática de desenvolvimento de *software* onde os membros de uma equipa juntam as suas alterações de código num repositório central. Cada integração é verificada por uma construção automatizada, incluindo testes, para detectar erros de integração o mais rápido possível [24]. A ausência de integração contínua, dificulta a junção das alterações de código, tornando esse processo muito demorado, podendo ainda resultar em acumulações de erros por longos períodos de tempo. Estes fatores dificultam uma distribuição de atualizações rápida para os clientes.

Com a integração contínua, os desenvolvedores utilizam com frequência um repositório partilhado usando um sistema de controlo de versões, como o Git. Antes de cada envio de

código para o repositório, é possivel executar testes unitários locais como uma camada de verificação extra, anterior à integração. Um serviço de integração contínua cria e executa automaticamente testes unitários nas novas alterações de código para destacar imediatamente todos os erros [25].

Esta abordagem reduz significativamente os problemas de integração e permite que uma equipa desenvolva *software* com maior eficiência e rapidez.

No âmbito deste projeto, a utilização de integração contínua é mencionada na Secção 5.1.

Cookies

Uma das principais tecnologias, que facilita o direcionamento comportamental são as *cookies*. Um cookie é um pequeno pedaço de código incorporado num *site* que coloca um identificador exclusivo no navegador do utilizador quando ele visita o *site* de um determinado anunciante [26]. Esta *cookie* permite ao anunciante rastrear o comportamento de navegação do visitante da sua página *web*, guardando informações sobre a sessão do visitante.

As *cookies* de um navegador são identificadas e lidas por pares “nome-valor”. Eles informam as *cookies* para onde devem ser enviadas e quais os dados que devem ser recuperados. O navegador armazena a *cookie* localmente para relembrar o par "nome-valor” que a identifica. Se um visitante voltar a esse *site*, o navegador irá recuperar os dados das suas sessões anteriores [27].

Capítulo 3

Análise

Após a contextualização do problema, é necessário realizar uma análise mas detalhada ao problema em si. Neste capítulo será feita uma análise ao problema em questão e serão definidos os requisitos da nova solução. Será também avaliado o valor da solução final, de modo a identificar a existência do mesmo e definindo os moldes para o desenvolvimento da solução.

3.1 Engenharia de Requisitos

A Engenharia de Requisitos, conhecida como a primeira fase do processo de Engenharia de *Software*, é considerada a principal tarefa do desenvolvimento de *software* [28]. Uma Engenharia de Requisitos eficaz é essencial para o resto do processo de desenvolvimento de *software*. A Engenharia de Requisitos também é um fator crucial que influencia a produtividade e a qualidade do produto final.

O modelo FURPS (*Functionality, Usability, Reliability, Performance, Supportability*), proposto por Robert Grady e Hewlett-Packard, decompõe características em duas categorias diferentes de requisitos, os requisitos funcionais e os requisitos não funcionais [29]. Os requisitos funcionais representam o que o software faz, em termos de tarefas e serviços, ou seja, as suas Funcionalidades. Os requisitos não funcionais representam um conjunto de características que o sistema deve cumprir, desde Usabilidade, Confiabilidade, Desempenho e Suportabilidade.

3.1.1 Requisitos Funcionais

Neste sub-capítulo, serão apresentados detalhadamente os requisitos funcionais deste projeto. De modo a tornar mais percetível a sua compreensão, quando necessário o requisito é acompanhado por um *Short Sequence Diagram* (SSD) que o ilustra. Um SSD é um diagrama que representa a interação entre o Utilizador e o Sistema.

Em todos eles, o Ator Principal e as Partes Interessadas é o Utilizador da Plataforma E-goi, sendo do seu interesse garantir que é possível criar *on-site messages* e fazer uma completa gestão das mesmas.

UC-1: Criar On-site Message

"Como utilizador deste Sistema, eu quero criar uma *on-site message* para ser listada nas minhas Páginas."

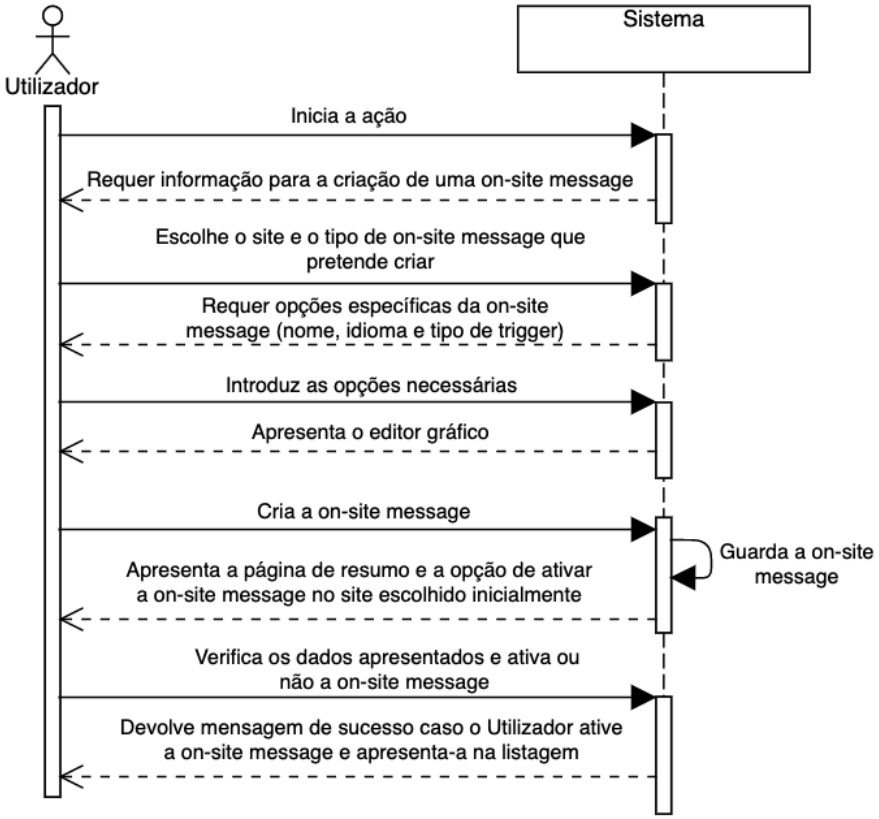

Figura 3.1: SSD do UC1

Para que este caso de uso, que pode ser considerado o principal, se suceda, podemos considerar o seguinte fluxo de eventos:

1. O utilizador inicia a ação de criação de uma *on-site message*.
2. O sistema requer as primeiras informações necessárias para criação de uma on-site message.
3. O utilizador define o domínio onde deseja que a *on-site* apareça e o seu tipo (caixa pop-up, barra lateral, *banner* suspenso, ecrã total ou *slider*).
4. O sistema requer as opções específicas da *on-site message*.
5. O Utilizador define um nome válido, um idioma e um tipo de *trigger*.
6. O sistema apresenta o editor gráfico.
7. O utilizador cria a sua *on-site message*.
8. O sistema guarda automaticamente a *on-site* de uma forma periódica, garantindo assim que o utilizador não perde todo o trabalho desenvolvido caso aconteça algo de anormal.
9. O sistema apresenta a página de resumo, que contém toda a informação definida pelo utilizador até ao momento e permite ativar a *on-site* no domínio definido.
10. O utilizador verifica os dados apresentados e ativa, ou não, a sua on-site message.

11. O sistema devolve uma mensagem de sucesso, caso o utilizador tenha ativado a *on-site* e apresenta-a na listagem.

UC-2: Editar On-site Message

"Como utilizador deste Sistema, eu quero abrir o editor gráfico e fazer alterações a uma *on-site message*."

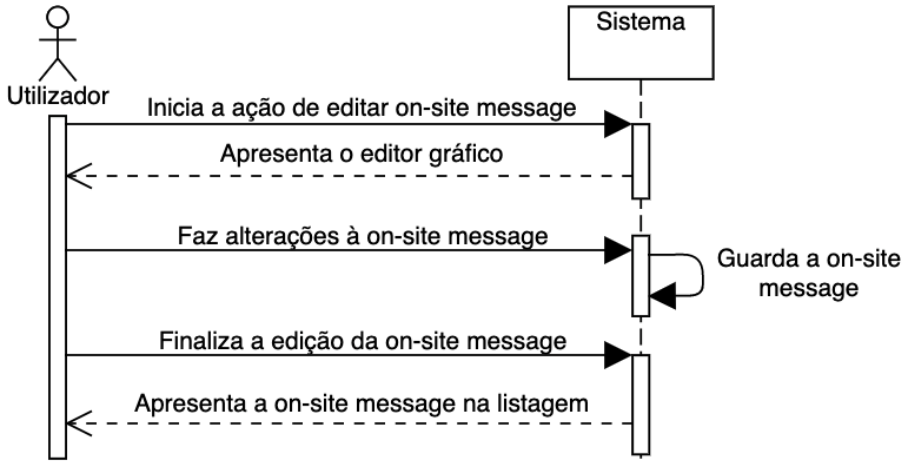

Figura 3.2: SSD do UC2

Para que este caso de uso, que pode ser considerado o principal, se suceda, podemos considerar o seguinte fluxo de eventos:

1. O utilizador inicia a ação de editar uma *on-site message*.
2. O sistema apresenta o editor gráfico com a última versão guardada da *on-site* escolhida pelo utilizador.
3. O utilizador faz as alterações pretendidas.
4. À semelhança da UC-1, o sistema guarda automaticamente o formulário de uma forma periódica.
5. O utilizador finaliza a edição da *on-site message*.
6. O sistema apresenta a listagem, com a versão da *on-site* atualizada.

UC-3: Consultar/Editar opções da On-site Message

"Como utilizador deste Sistema, eu quero verificar as opções da *on-site message* e edita-las, se necessário."

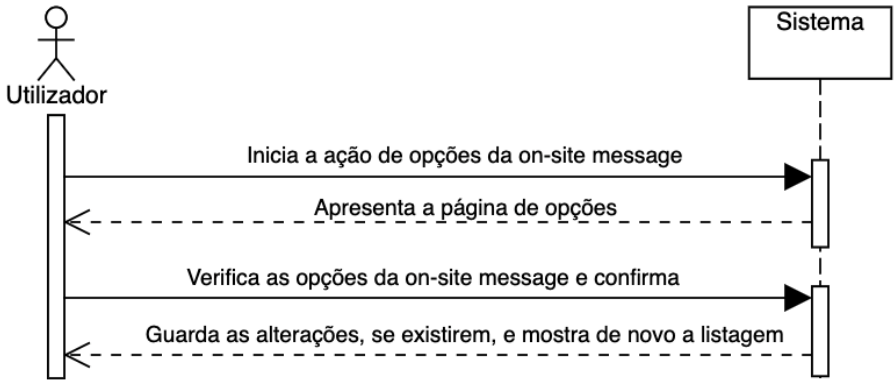

Figura 3.3: SSD do UC3

Para que este caso de uso, que pode ser considerado o principal, se suceda, podemos considerar o seguinte fluxo de eventos:

1. O utilizador inicia a ação de opções da *on-site message*.
2. O sistema apresenta a página de opções.
3. O utilizador verifica as opções definidas, podendo alterar o nome, o idioma e o tipo de *trigger*.
4. O sistema guarda as alterações, se existirem, e mostra de novo a listagem.

UC-4: Antever On-site Message

"Como utilizador deste Sistema, eu quero antever uma *on-site message*, de modo a perceber como vai aparecer em diferentes dispositivos."

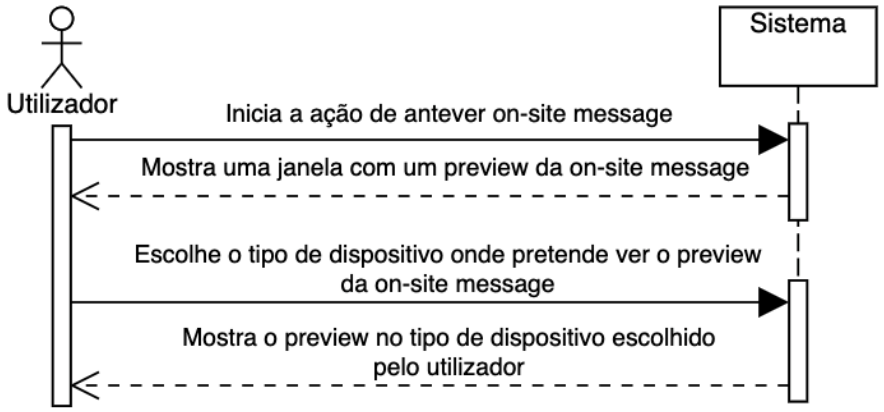

Figura 3.4: SSD do UC4

Para que este caso de uso, que pode ser considerado o principal, se suceda, podemos considerar o seguinte fluxo de eventos:

1. O utilizador inicia a ação de antever a *on-site message*.

2. O sitema apresenta uma janela com um *preview* da *on-site* num computador por *default* e permite alterar o tipo de dispositivo onde o utilizador pretende pré-visualizar.
3. O utilizador verifica o *preview* da *on-site message* e pode escolher outro tipo de dispositivo (iPhone ou iPad, na horizontal ou vertical).
4. Mostra o *preview* no tipo de dispositivo escolhido pelo utilizador.

UC-5: Apagar On-site Message

"Como utilizador deste Sistema, eu quero apagar uma *on-site message* criada."

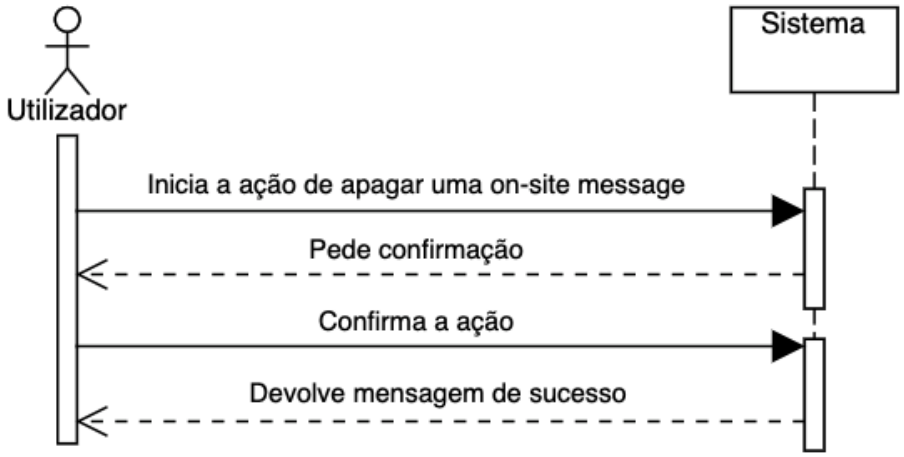

Figura 3.5: SSD do UC5

Para que este caso de uso, que pode ser considerado o principal, se suceda, podemos considerar o seguinte fluxo de eventos:

1. O utilizador inicia a ação de apagar uma *on-site message*.
2. O sistema pede confirmação.
3. O utilizador confirma.
4. O sistema elimina a *on-site message* selecionada.

UC-6: Consultar resultados de uma On-site Message

"Como utilizador deste Sistema, eu quero consultar os resultados que a minha *on-site message* está a ter em termos de visitas, visitas únicas e cliques."

Para que este caso de uso, que pode ser considerado o principal, se suceda, podemos considerar o seguinte fluxo de eventos:

1. O utilizador inicia a ação de consulta de resultados da *on-site message*.
2. O sistema apresenta a página de resultados com todos os dados (visitas, visitas únicas e cliques) relativos à *on-site* escolhida pelo utilizador.

Para além dos requisitos já enunciados, é possível identificar mais três, que dizem respeito à parte pública do sistema, ou seja, aquilo que acontece no ecrã do dispositivo dos visitantes das páginas *web* dos utilizadores. Uma vez que, nestes casos, a interação do utilizador com o sistema era bastante simples, apenas foi feita uma descrição textual dos requisitos.

UC-7: Despoletar trigger na página web do Utilizador

"Como utilizador deste sistema, eu quero que a minha *on-site message* apenas apareça no ecrã do visitante da minha página quando o *trigger* que defini despoletar."

O visitante de uma página *web* que contenha nela uma *on-site message*, deve ver a mesma, apenas quando o seu comportamento na página despoletar o *trigger* de ativação definido pelo utilizador. Esse *trigger* pode ser qualquer um dos tipos mencionados na Subsecção 2.2.1, desde que tenha sido definido nas opções da *on-site* pelo utilizador.

UC-8: Personalizar mensagens usando códigos de personalização

"Como utilizador deste sistema, eu quero conseguir personalizar as minhas mensagens caso o visitante da minha página já esteja na lista de contactos associada à minha *on-site message*."

Através de um sistema de *cookies*, explicado na Secção 2.6, é possível o utilizador identificar se o visitante da sua página é novo ou se já está na lista de contactos associada à *on-site message*. Ao criar uma *on-site message*, o utilizador escreve a sua mensagem e através de um código de personalização consegue tornar a comunicação com os seus visitante dinâmica. Por exemplo, na mensagem "Bem-vindo !fname", "!fname"é o código de personalização e se o visitante for conhecido, o código será substituído pelo nome do mesmo, originando uma mensagem, como por exemplo, "Bem-vindo João".

3.1.2 Requisitos Não Funcionais

Neste sub-capítulo, serão apresentados requisitos não funcionais do sistema, de acordo com o sistema FURPS+, onde o sinal de mais (+) representa restrições que possam existir, como restrições de design ou de implementação [30].

Usabilidade

A usabilidade avalia os requisitos com base na experiência do utilizador ao usar interface do sistema (por exemplo estética geral, consistência e documentação) [30]. Para este projeto foram definidos os seguintes requisitos de usabilidade:

- O editor gráfico deve adaptar-se para os diferentes tipos de *on-site messsage*.
- O sistema deve garantir a responsividade de uma *on-site message* quando esta aparece no ecrã do dispositivo do visitante. As suas dimensões devem ser adaptadas conforme o tamanho do ecrã onde aparecem, desde monitores de computador, de *tablets* e de telemóveis.

Confiabilidade

A confiabilidade diz respeito à disponibilidade do sistema, à sua previsibilidade, precisão e tempo médio entre falhas [30]. Para este projeto foram definidos os seguintes requisitos de confiabilidade:

- O sistema deve informar o utilizador de qualquer mudança de estado quer do próprio sistema quer da *on-site messsage*, no decorrer da sua utilização.

Desempenho

O desempenho do sistema é avaliado pela sua velocidade de processamento, tempo de resposta, rendimento e eficiência. Para este projeto foram definidos os seguintes requisitos de desempenho:

- O tempo de espera entre cada página do fluxo de criação de uma *on-site messsage* deve ser aceitável para o utilizador.

Suportabilidade

A capacidade de suporte está relacionada com extensibilidade, adaptabilidade, compatibilidade, facilidade de manutenção, instabilidade [30]. Para este projeto foram definidos os seguintes requisitos de suportabilidade:

- O sistema deve ser desenvolvido de forma a ser facilmente escalável.
- O sistema deve ser suportado pelo produto E-goi estando totalmente integrado no mesmo.

Outros

Relativamente a restrições de implementação, tem-se que:

- O sistema deve ser desenvolvido usando a framework Angular e AngularJS, a linguagem de programação PHP e o sistema de gestão de base de dados MariaDB.
- O sistema deve seguir a mesma lógica de design de interface gráfica do produto E-goi.

3.2 Análise de Valor

Num mundo tão acelerado, o desenvolvimento do produto deve ser transformado num processo de aprendizagem contínuo e iterativo, com foco no valor do cliente [31]. "A Análise de Valor pode ser definida como um processo de revisão sistemática que é aplicada aos projetos de produtos existentes, para avaliar a função do produto exigida por um cliente, para atender aos seus requisitos com o menor custo, consistente com o desempenho especificado e a confiabilidade necessária"[32]. Ou seja, é uma abordagem organizada para melhorar a lucratividade de aplicações.

3.2.1 Processo de Inovação

Segundo Nick Skillicorn, inovar é transformar uma ideia nova numa solução que agrega valor pela perspectiva de um cliente. Contudo, ser diferente não significa necessariamente ser capaz de criar valor. Para que tal aconteça, é necessário que a inovação se constitua como vantagem competitiva. Por isso, é importante procurar condições em que a inovação se transforma numa vantagem competitiva que permita às empresas consubstanciar, de facto, a sua capacidade diferenciadora, em valor [33].

O processo de inovação apresenta uma das maiores oportunidades para melhorar o processo geral de inovação [34]. Todo o processo de inovação pode ser dividido em três partes: Fuzzy Front End, New Product Development e Comercialização.

O Fuzzy Front End (FFE) é o primeiro passo do processo de inovação. O FFE é um componente crítico do processo de inovação. A qualidade do trabalho de front-end é decisiva para o sucesso da inovação, pois aqui as oportunidades de inovação são identificadas, analisadas e trabalhadas. As escolhas feitas no FFE determinarão, em última instância, quais opções de inovação podem ser consideradas para desenvolvimento e comercialização [35].

O New Product Development (NPD) obedece a um conjunto sistemático (pesquisa científica) de etapas organizadas, cujo objetivo é converter uma consideração de produto em

produto acabado tangível. O NPD inicia-se com a partir de uma oportunidade de mercado e culmina com a fabricação, venda e entrega de um produto [36].

Por fim, segue-se a última etapa, a Comercialização. Esta etapa tem por objetivo vender o produto final criado, devidamente testado e aprovado.

3.2.2 New Concept Development

O New Concept Development (NCD) foi desenvolvido por Peter Koen com o objetivo de fornecer um vocabulário para a compreensão das atividades que ocorrem no processo de inovação [37]. O modelo é representado por três partes fundamentais, tal como é apresentado na Figura 3.6.

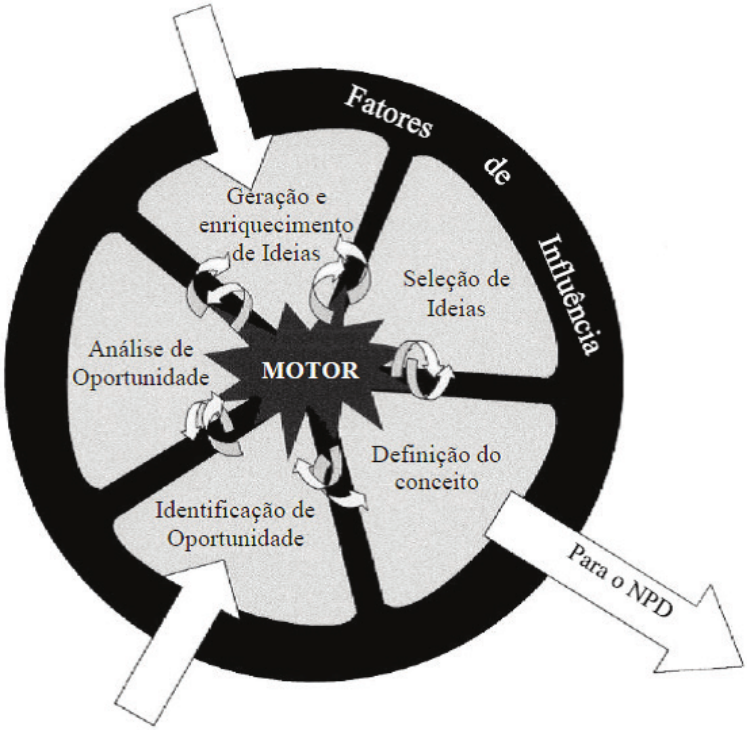

Figura 3.6: Modelo New Concept Development [38]

A primeira parte é o motor, ou centro do modelo, que é responsável pela visão, estratégia e cultura, que impulsiona o processo de inovação. A segunda parte, é a parte interna do modelo e define os cinco elementos de atividade do front end. São eles, a identificação de oportunidade, análise de oportunidade, a geração e enriquecimento de ideias, seleção de ideias e por fim a definição do conceito. A terceira e última parte do modelo diz respeito aos fatores externos de influência, não controláveis pela empresa que podem influenciar o processo de inovação.

A forma circular do NCD, pretende sugerir que as ideias devem fluir e iterar entre todos os cinco elementos.

Identificação da Oportunidade

A identificação da oportunidade define o mercado ou a área de tecnologia, na qual a empresa pode querer participar. Por exemplo, a oportunidade pode ser uma resposta de curto prazo a

uma ameaça competitiva, uma possibilidade de avanço para obter vantagem competitiva ou um meio de simplificar as operações, acelerá-las ou reduzir seus custos. Pode ser uma direção totalmente nova para o negócio ou uma atualização de um produto existente. Também pode ser uma nova plataforma de produto, um novo processo de produção, uma nova oferta de serviço ou uma nova abordagem de marketing ou vendas [39].

Os problemas identificados na Secção 1.3 do Capítulo 1, representam uma oportunidade de melhoramento do produto E-goi, através da criação de um novo canal de comunicação *on-site messaging*, que irá trazer uma nova oferta para o cliente.

Análise da Oportunidade

Uma oportunidade é avaliada para confirmar que vale a pena segui-la. Informações adicionais são necessárias para traduzir a identificação de oportunidades em negócios específicos e oportunidades de tecnologia, o que envolve fazer avaliações precoces e frequentemente incertas de tecnologia e mercado. A análise de oportunidade é a fase onde se avalia a capacidade e a competência do negócio e deve fazer parte de um processo iterativo [40]. Uma análise completa deve incluir:

- Enquadramento estratégico: determinação de como a oportunidade se encaixa no mercado da empresa e nos pontos fortes, fracos e ameaças da tecnologia.
- Avaliação do segmento de mercado: descrição detalhada do segmento de mercado, mostrando o porquê de este representar uma oportunidade, tal como é apresentado na Secção 2.1.
- Análise da concorrência: determinação dos principais concorrentes, no segmento de mercado identificado, tal como é apresentado na Secção 2.4.
- Avaliação do cliente: determinação das necessidades principais do cliente que não estão a ser atendidas pelos produtos atuais.

Enquadramento Estratégico

O enquadramento estratégico é o primeiro passo para realizar uma análise de oportunidade consistente e completa. Para determinar os pontos fortes e fracos, as oportunidades e as ameaças da tecnologia, foi elaborada uma análise SWOT, apresentada na Figura 3.7.

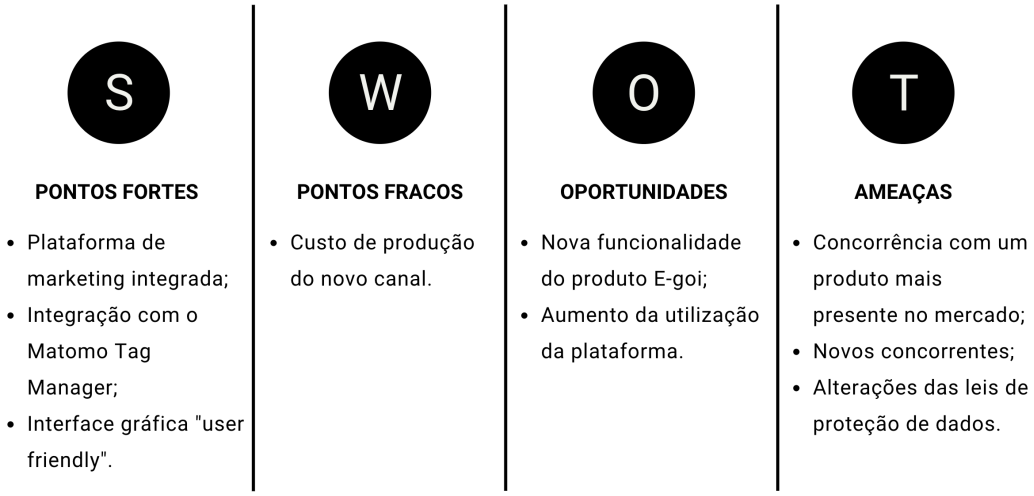

Figura 3.7: Análise SWOT

Relativamente aos pontos fortes, foram identificados três pontos fulcrais, referentes ao novo canal de comunicação *on-site messaging*. O facto de ser integrado com a plataforma de marketing do E-goi, traz várias vantagens ao cliente, como por exemplo, o envio de mensagens *on-site* para as suas listas de contactos predefinidas. A integração com o Matomo Tag Manager, é outro ponto que traz vantagens ao cliente, uma vez que permite definir condições de aparecimento das mensagens através de *triggers* variados, tal como é apresentado na Secção 2.2. O último ponto forte identificado, foi a interface gráfica *user friendly*. De modo a proporcionar ao cliente uma experiência de utilização positiva, a interface gráfica do novo canal de comunicação, será bastante intuitiva e fácil de manipular.

O único ponto fraco identificado, foi o custo de produção do novo canal. O desenvolvimento de um novo canal de comunicação, implica uma análise aos canais existentes atualmente, construir uma solução que responda aos objetivos do novo canal, implementá-la e testá-la. Este custo, pode não compensar o valor que o novo canal de comunicação poderá trazer e, por isso, é algo a ter em conta na tomada de decisão e na definição do conceito.

Em relação às oportunidades que o novo canal de comunicação trará, foram identificados dois pontos chave. Será uma nova funcionalidade do produto E-goi no mercado, o que consequentemente, aumentará a utilização da plataforma, por parte dos clientes atuais e ainda atrair novos clientes.

Por fim, foram identificadas três ameaças. Concorrentes com um produto mais presente no mercado, representam uma ameaça, uma vez que já possuem um produto mais consistente e completo. Estando o mercado das *on-site messages* em constante crescimento, o surgimento de novos concorrentes, poderá constituir também uma ameaça. Por fim, alterações legislativas sobre a proteção de dados, poderam tornar impraticáveis algumas funcionalidades do novo canal.

Geração de Ideias

O elemento de geração de ideias, diz respeito ao nascimento, desenvolvimento e maturação de uma ideia concreta. A geração de ideias é evolucionária. As ideias são construídas, destruídas, combinadas, remodeladas, modificadas e atualizadas. Uma ideia pode passar por muitas iterações e mudanças à medida que é examinada, estudada, discutida e desenvolvida em conjunto com outros elementos do modelo *New Concept Development* [41].

Após uma pesquisa de possíveis soluções, que respondessem corretamente aos pontos identificados na oportunidade e de uma discução interna, surgiram três possíveis abordagens para a solução:

- A criação de um novo canal de comunicação *on-site messaging* com recurso ao Matomo;
- A reconstrução do Easygoi;
- A criação de um novo canal de comunicação *on-site messaging* sem recurso ao Matomo;

Todas as abordagens definidas diferem entre si, contudo, resolvem os problemas identificados. É necessário fazer uma análise de cada uma, de maneira a decidir a melhor solução, tendo por base, alguns critérios chave.

Seleção de Ideias

A seleção de ideias é fundamental para o sucesso futuro do negócio. Depois de identificadas as ideias capazes de resolver o problema, é necessário perceber as vantagens e desvantagens de cada uma, com o objetivo de tomar uma decisão final. Para auxiliar a tomada de decisão, foi utilizado o método AHP.

O método AHP (Analytic Hierarchy Process) foi desenvolvido por Tomas L. Saaty no início da década de 70 e é o método de multicritério mais utilizado e conhecido no apoio à tomada de decisão [42].

É baseado em cinco etapas, sendo elas a divisão hierárquica, a definição de prioridades relativas, a avaliação da consistência das prioridades relativas, a construção da matriz de comparação paritária para cada critério e por fim, a escolha da alternativa.

Divisão hierárquica

No método AHP, o problema é estruturado em níveis hierárquicos, o que facilita a melhor compreensão e avaliação do mesmo. Para a aplicação desta metodologia, é necessário que tanto os critérios quanto as alternativas possam ser estruturadas de forma hierárquica, sendo que, o primeiro nível da hierarquia corresponde à definição do objetivo, o segundo aos critérios e o terceiro às alternativas de soluções para o problema. A Figura 3.8 apresenta a estrutura hierárquica do método AHP.

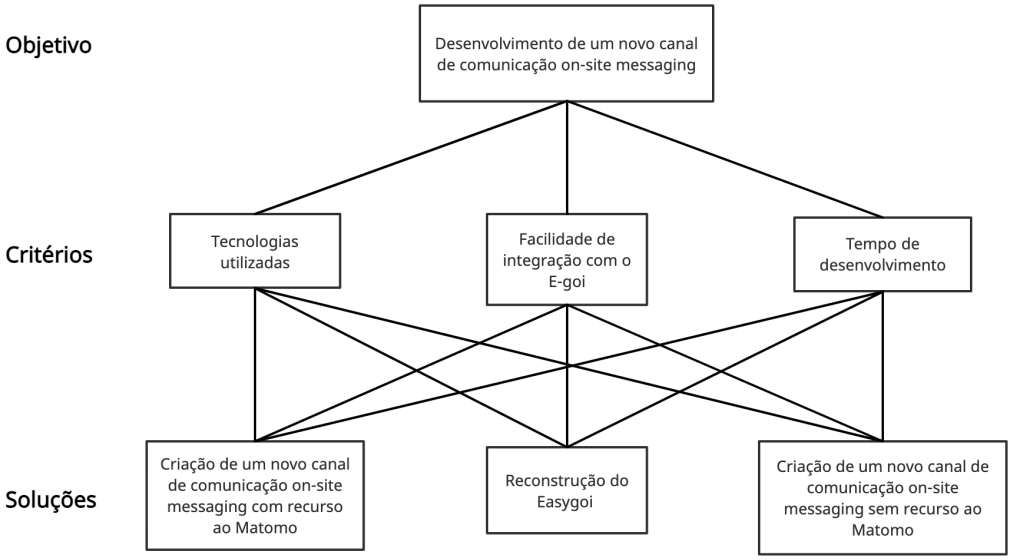

Figura 3.8: Estrutura hierárquica do AHP

O objetivo definido, foi o desenvolvimento de um novo canal de comunicação *on-site messaging*. Para atingir este objetivo, foram identificados três critérios, sendo eles as tecnologias utilizadas, a facilidade de integração com o E-goi e ainda o tempo de desenvolvimento. As alternativas para solucionar o problema, são concordantes com as opções definidas na Secção 3.2.2, sendo elas a criação de um novo canal de comunicação *on-site messaging*, com recurso ao Matomo, a reconstrução do Easygoi ,e por fim, a criação de um novo canal de comunicação *on-site messaging*, sem recurso ao Matomo.

Definição de Prioridades Relativas

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 [43], que varia de 1 a 9, tal como demonstrado no Anexo A.1.

Tendo os seguintes critérios:

- A: Tecnologias utilizadas
- B: Facilidade de integração com o E-goi
- C: Tempo de desenvolvimento

Tabela 3.1: Matriz de comparação dos critérios

	A	B	C	Prioridade Relativa
A	1	3	5	0.63
B	0.33	1	3	0.26
C	0.2	0.33	1	0.11

As prioridades relativas, têm por objetivo identificar a ordem de importância de cada critério. Para tal, é calculada a média aritmética dos valores de cada linha da matriz normalizada.

Através da Tabela 3.1, é possível perceber que o critério das tecnologias utilizadas é bastante mais importante que a facilidade de integração e o tempo de desenvovimento. A facilidade de integração com o E-goi, é ligeiramente mais importante do que o tempo de desenvolvimento.

Definidas as prioridades relativas, a próxima etapa é calcular a Razão de Consistência (RC), para medir o quanto os julgamentos foram consistentes em relação a grandes amostras de juízos completamente aleatórios.

Avaliação da consistência das prioridades relativas

Para calcular a Razão de Consistência (RC), é necessário primeiro, obter o valor de λ_{max},

que representa o maior valor próprio da matriz presente na Tabela 3.1, obtido a partir da seguinte equação:

$$Ax = \lambda_{max} x$$

Sendo A a matriz e:

$$x = (0.63; 0.26; 0.11)$$

Então:

$$\begin{bmatrix} 1 & 3 & 5 \\ 0.33 & 1 & 3 \\ 0.2 & 0.33 & 1 \end{bmatrix} \begin{bmatrix} 0.63 \\ 0.26 \\ 0.11 \end{bmatrix} \cong \lambda_{max} \begin{bmatrix} 0.63 \\ 0.26 \\ 0.11 \end{bmatrix} \Leftrightarrow \begin{bmatrix} 1.95 \\ 0.79 \\ 0.32 \end{bmatrix} \cong \lambda_{max} \begin{bmatrix} 0.63 \\ 0.26 \\ 0.11 \end{bmatrix} \Leftrightarrow$$

$$\Leftrightarrow \lambda_{max} = average\{1.95/0.63, 0.79/0.26, 0.32/0.11\} = 3.04$$

Para calcular o Índice de Consistência (IC), sendo n o número de critérios, tem-se:

$$IC = \frac{\lambda_{max} - n}{n-1} = \frac{3.04 - 3}{3-1} = 0.02$$

Depois de obter o valor do Índice de Consistência, é possível calcular a Razão de Consistência. Para isso, é necessário identificar o valor do Índice aleatório (IR), que segundo o Anexo A.2, para 3 critérios é 0.58. Assim:

$$RC = \frac{IC}{IR} = \frac{0.02}{0.58} = 0.03 < 0.1$$

Uma vez que o RC é inferior a 0.1, os julgamentos são confiáveis porque estão afastados, para o conforto de aleatoriedade. Neste caso os resultados obtidos apresentam valores consistentes.

Construção da matriz de comparação paritária para cada critério

Confirmada a consistência das prioridades relativas, é necessário priorizar as soluções relativamente a cada critério e para isso foram utilizadas três tabelas semelhantes à Tabela 3.1.

Tendo as seguintes soluções:

- X: Criação de um novo canal de comunicação on-site messaging com recurso ao Matomo
- Y: Reconstrução do Easygoi
- Z: Criação de um novo canal de comunicação on-site messaging sem recurso ao Matomo

Tabela 3.2: Matriz de comparação paritária de tecnologias utilizadas

	X	Y	Z	Prioridade Relativa
X	1	9	3	0.65
Y	0.11	1	0.14	0.06
Z	0.33	7	1	0.29

Tabela 3.3: Matriz de comparação paritária de facilidade de integração

	X	Y	Z	Prioridade Relativa
X	1	2	0.33	0.27
Y	0.5	1	0.5	0.19
Z	3	2	1	0.54

Tabela 3.4: Matriz de comparação paritária de tempo de desenvolvimento

	X	Y	Z	Prioridade Relativa
X	1	5	3	0.63
Y	0.2	1	0.33	0.11
Z	0.33	3	1	0.26

A partir da análise das tabelas, é possível verificar que relativamente ao critério das tecnologias utilizadas, a solução X, é claramente a mais prioritária. Já em relação ao critério da facilidade de integração, a solução Z, é a mais prioritária. Por fim, relativamente ao critério do tempo de desenvolvimento, a solução X, é a mais prioritária.

Escolha da alternativa

Terminadas todas as etapas anteriores, obtemos as seguintes matrizes:

Tabela 3.5: Matriz de prioridades relativas das soluções

	A	B	C
X	0.65	0.27	0.63
Y	0.06	0.19	0.11
Z	0.29	0.54	0.26

	Prioridade Relativa
A	0.63
B	0.26
C	0.11

Tabela 3.6: Matriz de prioridades relativas dos critérios

Nesta última etapa, obtemos as prioridades compostas das alternativas, multiplicando as duas matrizes. A matriz de prioridades compostas é apresentada na Tabela 3.7.

Tabela 3.7: Matriz de prioridade composta

	Prioridade Composta
X	0.55
Y	0.10
Z	0.35

Conclusões

O valores finais obtidos para cada solução foram:

- Solução X: 0.46;
- Solução Y: 0.10;
- Solução Z: 0.44.

Conclui-se, que a melhor solução, será adotar a Solução X, ou seja, a criação de um novo canal de comunicação *on-site messaging*.

Definição do Conceito

A definição de conceito é o elemento final do *New Concept Development*. Depois de utilizar o método AHP, para determinar a melhor abordagem para o problema, é importante realizar uma definição do conceito.

O conceito para o novo canal de comunicação *on-site messaging* do E-goi, consiste numa interface gráfica intuitiva, simples e fácil de utilizar, que permitirá ao cliente criar as suas *on-site messages*, através de um editor *drag-and-drop* e definir o comportamento das mesmas com recurso a variados *triggers*.

3.2.3 Valor da Solução

Valor, Valor para o Cliente, Valor Percecionado

O valor foi definido em diferentes contextos teóricos, como necessidade, desejo, interesse, padrão/critérios, crenças, atitudes e preferências. A criação de valor, é fundamental para qualquer negócio, e qualquer negócio consiste em trocar algum bem ou serviço tangível ou intangível e ter o seu valor aceite e recompensado por clientes, seja dentro da empresa ou rede colaborativa ou fora dela [44].

O valor para o cliente consiste no valor que o cliente dá ao produto consoante a sua precessão. Oferecer valor para o cliente é, de acordo com Phillip Kotler, algo relacionado com a criação de razões genuínas para que ele opte pelo consumo de um produto ou serviço. Relativamente ao projeto apresentado nesta dissertação, oferecendo um produto mais simples e intuitivo ao utilizador, que lhe permita criar livremente as suas mensagens *on-site*, é expectável que o cliente sinta um aumento no valor do produto.

O valor percecionado, é a avaliação dos clientes sobre os benefícios de um produto ou serviço e a sua capacidade de atender às suas necessidades e expectativas [45]. No âmbito deste projeto, é possível classificar esta solução como um benefício para o cliente, uma vez que consiste numa nova funcionalidade para o E-goi, que permitirá criar *on-site messages* de forma intuitiva e original.

Proposta de Valor

Uma proposta de valor é sobre o cliente, mas para uso interno da empresa. Deve definir exatamente o que a organização pretende fornecer para a vida do cliente. Define a forma como as organizações trabalham, concentrando as suas atividades no melhor atendimento aos clientes, ao mesmo tempo que o fazem com lucro [46].

No âmbito deste projeto, a proposta de valor, consiste num canal de comunicação *on-site messaging* integrado no produto E-goi, com recurso ao Matomo, o que permitirá ao cliente definir condições de aparecimento das mensagens. Todo o fluxo de criação das mensagens será simples e intuitivo, de modo a proporcionar uma boa experiência de utilização ao cliente. Esta solução vai criar uma nova funcionalidade no produto E-goi.

3.2.4 Business Model Canvas

O Business Model Canvas é uma ferramenta de gestão estratégica, que permite desenvolver e esboçar modelos de negócio novos ou existentes numa única página. O modelo Canvas aplicado ao tema desta dissertação está apresentado na Figura 3.9.

<table>
<tr>
<td rowspan="2">Parceiros Chave

------</td>
<td>Atividades Chave

- Desenvolvimento e implementação de uma solução;
- Estudo de usabilidade.</td>
<td rowspan="2">Proposta de Valor

- Criação de um canal de comunicação on-site messaging integrado numa plataforma de marketing multicanal;

- Interface gráfica "user friendly"</td>
<td>Relação com Clientes

- Serviço de apoio ao cliente.</td>
<td rowspan="2">Segmentos de Clientes

- E-Commerce;
- Bloggers;
- Marketing Automóvel;
- Agências de Marketing Digital;
- Marketing Imobiliário;
- Clínicas e Profissionais de Saúde e Bem-Estar;
- Turismo e Agências de Viagens;
- Marketing Financeiro;
- Retalho.</td>
</tr>
<tr>
<td>Recursos Chave

- Conhecimento tecnológico.</td>
<td>Canais

- Plataforma E-goi;
- Blog E-goi.</td>
</tr>
<tr>
<td colspan="3">Custos

- Hardware;
- Equipas de trabalho.</td>
<td colspan="2">Fontes de Retorno

- Uso da plataforma E-goi.</td>
</tr>
</table>

Figura 3.9: Business Model Canvas

Este modelo é dividido em 9 elementos, sendo eles:

- Parceiros Chave: Rede de fornecedores e parceiros essenciais que garantem o funcionamento do Modelo de Negócio. No âmbito deste projeto, não existem parceiros chave, sendo a E-goi a manter exclusivamente o produto e consequentemente as suas funcionalidades;
- Atividades Chave: As coisas mais importantes que a empresa deve fazer de forma constante, para que o Modelo de Negócio funcione. No âmbito deste projeto, foram identificadas duas atividades chave, o desenvolvimento e implementação de uma solução e o estudo de usabilidade da mesma.
- Recursos Chave: São os ativos fundamentais necessários, para fazer o Modelo de Negócio funcionar. Para este projeto, foi apenas definido um recurso chave, o conhecimento tecnológico, necessário para implementar a solução.
- Proposta de Valor: Este bloco representa os pacotes de produtos e serviços que geram valor para os segmentos de clientes específicos. A proposta de valor, no âmbito deste projeto, consiste na criação de um canal de comunicação *on-site messaging* integrado numa plataforma de marketing multicanal, que é o E-goi, e também numa interface gráfica *"user friendly"*.
- Relação com Clientes: Este bloco refere-se aos tipos de relacionamentos, que uma empresa estabelece com os seus segmentos. No âmbito deste projeto, é garantido um serviço de apoio ao cliente, capaz de responder a todas as dúvidas do cliente sobre o produto.
- Canais: Descreve quais os caminhos pelos quais a empresa comunica e entrega valor para o cliente. Foram identificados dois canais como meios de comunicação do produto, sendo eles a plataforma e o blog do E-goi, onde serão publicadas informações sobre o produto.

- Segmentos de Clientes: Diferentes grupos de pessoas ou organizações, que a empresa pretende servir, com necessidades ou comportamentos comuns, claramente definidos. No âmbito deste projeto, são vários os segmentos de clientes alvo do produto, como por exemplo, bloggers, agências de marketing digital, e-commerce, entre outros.
- Custos: Descreve todos os principais custos embutidos na operação do Modelo de Negócio. Os custos identificados para este projeto, são relativos a equipas de trabalho, assim como hardware, quer para desenvolvimento do canal, quer para alojamento do mesmo.
- Fontes de Retorno: Representa as possibilidades de geração de dinheiro que a empresa pode obter com cadas segmento de clientes. A fonte de retorno é o uso da plataforma E-goi, através de planos de subscrição da mesma.

3.3 Sumário

Neste capítulo foram inicialmente identificados os requisitos funcionais e não funcionais que a solução deste projeto terá que cumprir. De seguida, foi feita uma análise ao valor da solução, descrevendo todo o processo de inovação, desde a identificação da oportunidade à definição do conceito, de modo a perceber o valor que o desenvolvimento deste projeto pode trazer à empresa.

Depois de analisar e confirmar que o desenvolvimento deste projeto trás valor à empresa, é fundamental desenhar uma solução que cumpra os requisitos e objetivos definidos.

Capítulo 4

Desenho da Solução

O desenho da solução é uma etapa fundamental para planear o que irá ser implementado, com base 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 novo canal de comunicação, que servirão de base para a implementação da solução.

4.1 Domínio

O primeiro passo para o desenvolvimento de um sistema de *software*, é compreender, claramente, o conteúdo do domínio em questão.

De forma a facilitar a compreensão do problema em análise, foi elaborado um diagrama do modelo do domínio, que ilustra os conceitos relacionados com o problema e relações entre si e que é apresentado na Figura 4.1.

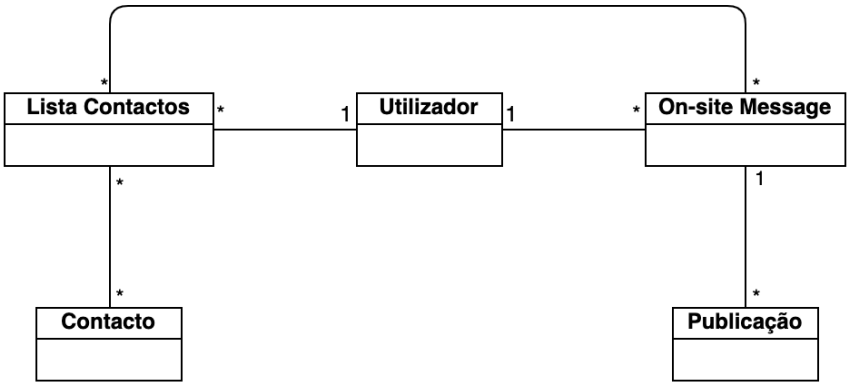

Figura 4.1: Diagrama do modelo do domínio da solução.

O utilizador representa um cliente E-goi que utiliza o sistema e tem a possibilidade de criar várias listas de contactos e também várias *on-site messages*. A *on-site message* contém informações tais como o nome da *on-site*, o idioma e o aspeto gráfico. Tem sempre associada uma lista de contactos definida pelo utilizador e pode ser publicada em uma ou mais páginas *web*, definidas também pelo utilizador. O contacto é um integrante de uma ou mais listas de contactos de uma conta E-goi, podendo visualizar mensagens personalizadas pelo utilizador, uma vez que pertence a uma lista de contactos. No contexto deste projeto, um contacto representa o visitante da página *web* de um utilizador. A publicação tem o domínio onde

será apresentada a *on-site message* e também toda a informação que é necessária, como *triggers* e limitações.

4.2 Arquitetura

A arquitetura de *software* é um conceito abstrato que trata da relação entre o mapeamento dos componentes de um *software* e os detalhes que são levados em conta na hora de implementar esses elementos. É o conjunto das principais decisões técnicas de design tomadas a respeito do *software*. Os elementos devem estar dispostos da melhor forma possível, garantindo que o utilizador terá uma experiência otimizada [47].

O produto E-goi apresenta uma arquitetura baseada em componentes. Como tal, o sistema é dividido em componentes, o que diminui a sua complexidade e permite que cada componente seja focado numa funcionalidade ou num conjunto restrito de funcionalidades semelhantes. Deste modo, estas funcionalidades podem ser reutilizadas em diversas aplicações através do acesso ao componente [48].

Segundo John Cheesman [49], os princípios fundamentais de uma arquitetura baseada em componentes 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.

Este modelo de arquitetura trás alguns benefícios para o sistema, tais como:

- Facilidade de manutenção e atualização: o facto do sistema ser construído a partir de componentes, facilita a localização de modificações e atualizações [50];
- Redução de custos: O uso do componente de terceiros, permite a redução do custo do desenvolvimento e manutenção;
- Fácil desenvolvimento: Implementar novos componentes, bem como a funcionalidade definida pela interface, permite desenvolvimento sem impacto em outros componentes do sistema [51];
- Reutilização: a reutilização de componentes é um meio de agilizar o desenvolvimento e manutenção, o que também ajuda na redução de custos [51].

Foi elaborado um diagrama de componentes, que representa a arquitetura do sistema, onde será implementada a solução.

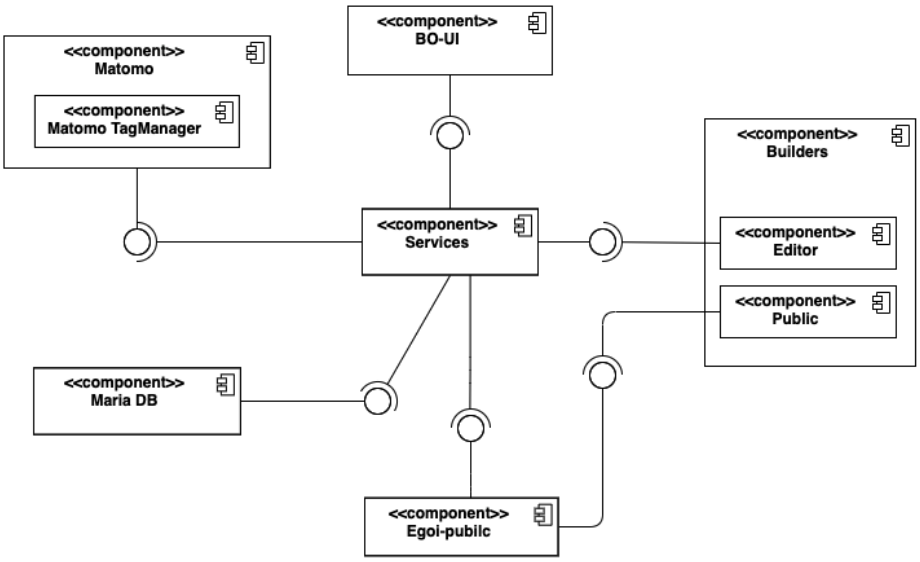

Figura 4.2: Diagrama de componentes

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

Tabela 4.1: Definição dos Componentes

Componentes	**Descrição**
BO-UI	Componente responsável pela interface gráfica da plataforma do E-goi
Matomo	Componente que contém o Matomo TagManager
Matomo TagManager	Plugin do Matomo que permite aos seus utilizadores incorporarem recursos de aplicações de terceiros no seu website
Builders	Componente que agrega o editor gráfico e a parte pública das páginas e formulários do produto E-goi
Editor	Editor gráfico de páginas e formulários do produto E-goi
Public	Parte pública que contém toda a lógica de renderização de páginas e formulários na página do cliente
Services	API que comunica com todos os componentes desde a base de dados até à interface gráfica
Egoi-public	API responsável por comunicar com o component Public do Builders e com o Services
Maria DB	Componente de persistência 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 REST API (*Representational State Transfer Application Program Interface*) desenvolvida na linguagem de programação PHP [52], que utiliza pedidos HTTP (*Hyper Text Transfer Protocol*) 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 [53], um sistema de gestão de bases de dados relacionais, rápido, escalável e robusto, criado pelos desenvolvedores do MySql [54].

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 a sua *on-site message*, 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 [55], 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 das *on-site messages*.

O Builders é um componente composto por outros dois componentes, o Editor e o Public, ambos desenvolvidos em Angular [56]. O Editor é a parte gráfica do editor de *on-sites* e será onde o utilizador poderá criar as suas *on-site messages* através de um sistema de *drag and drop*. Este Editor comunica com o Services para guardar toda a informação obtida pelo editor gráfico e também para ler informação, caso se trate de uma edição de uma página existente. O Public é a parte pública das *on-site messages*, que comunica com o componente Egoi-public, que 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 e personalização de mensagens para contactos já inseridos na lista de contactos do utilizador.

Arquitetura Alternativa

Uma alternativa à arquitetura escolhida como solução para o problema, seria desenvolver uma nova ferramenta, a partir do zero, que substituísse 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.

Este diagrama representa uma alternativa possível para resolver o problema. O componente do Matomo, é substituído por um novo componente, representado na Figura 4.3 como Trigger Service. Este novo componente, seria, tal como o Matomo, um meio de rastreamento do comportamento de um visitante de um determinado *website*, despoletando *triggers* consoante esse comportamento, mas seria uma ferramenta desenvolvida de raiz.

Contudo, esta solução, traz alguns obstáculos ao desenvolvimento do projeto. Atualmente, o Matomo, já se encontra integrado com o E-goi e, tal como concluido na Secção 2.5, é uma ferramenta bastante completa e viável, capaz de fornecer todas as funcionalidades necessárias para o desenvolvimento do projeto. Por estas razões, não se justifica desenvolver uma ferramenta nova, com as mesmas funcionalidades que o Matomo, sendo de referir ainda, que o tempo de desenvolvimento do projeto ficaria comprometido, dada a elevada dificuldade e complexidade de implementação que esta ferramenta traria.

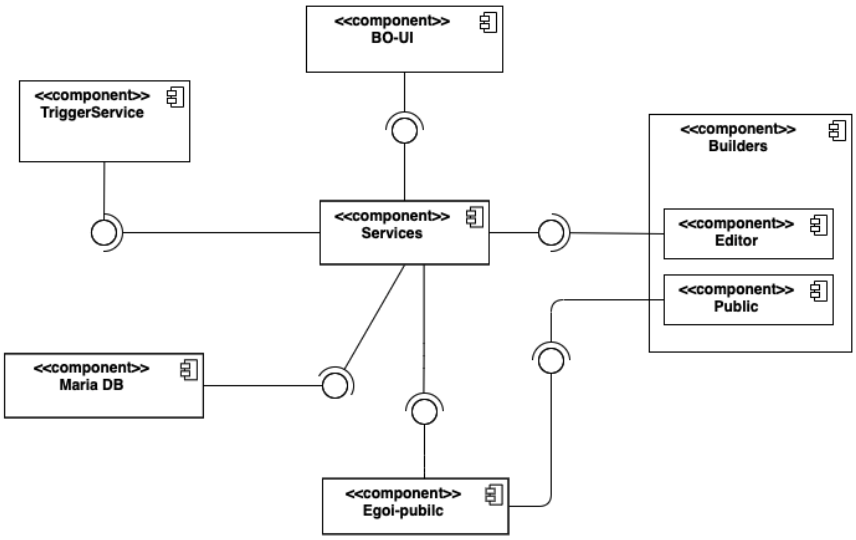

Figura 4.3: Diagrama de componentes da arquitetura alternativa

4.3 Implantação

A 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. Os diagramas de implantação são usados na modelagem dos aspectos físicos de um sistema orientado a objetos [57].

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.

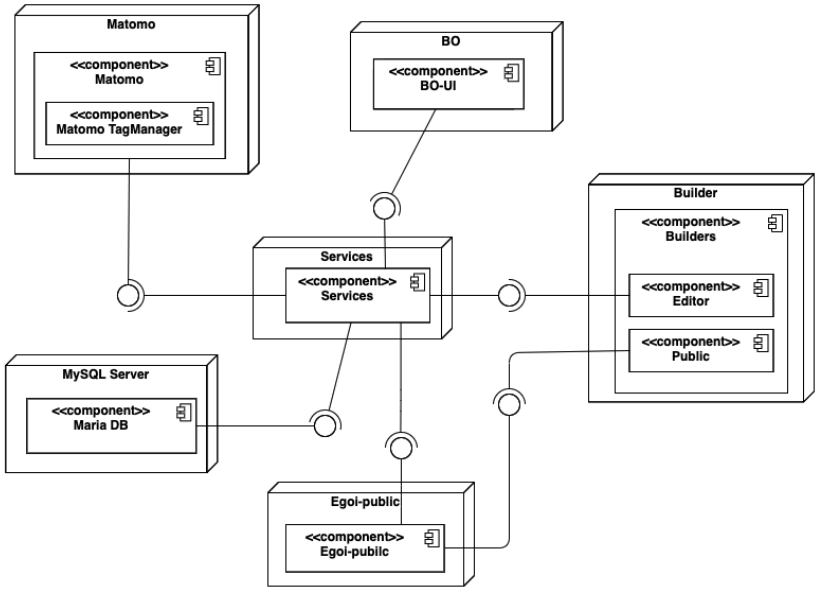

Figura 4.4: Diagrama de Implantação

4.4 Fluxo de Utilização

Nesta secção é apresentado o principal fluxo de utilização deste projeto 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.

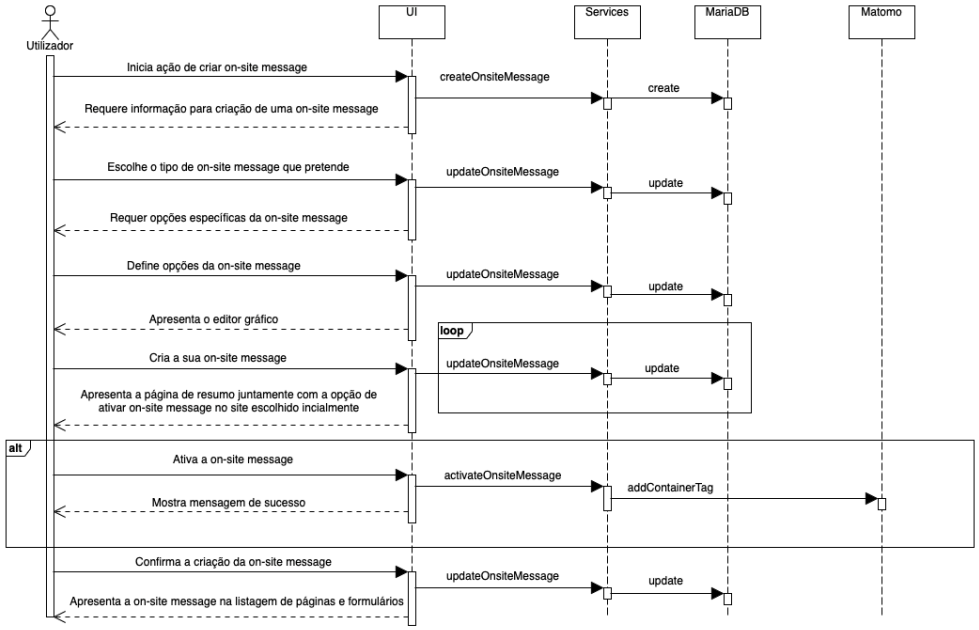

Figura 4.5: Diagrama de Sequência

A Figura 4.5 ilustra o diagrama de sequência que contém todas as interações do utilizador com o sistema e dos diferentes componentes entre si. O diagrama foi simplificado para tornar mais simples a sua compreensão, sendo que o componente UI representa a interface gráfica do componente BO-UI e também do editor gráfico Builders. O componente Services representa todas as classes intervenientes no processamento lógico do negócio e do tratamento de dados. O componente MariaDB representa a base de dados onde são guardadas todas as informações relativas a uma *on-site message* e o componente Matomo representa a API externa utilizada pelo sistema E-goi, responsável por garantir que os *triggers* são despoletados.

Quando o utilizador inicia a ação de criar uma *on-site message*, a UI faz um pedido ao Services, que cria uma *on-site message* na base de dados. De seguida, o utilizador escolhe o tipo de *on-site* que pretende criar e depois define opções específicas (nome, idioma, tipo de *trigger* e condição de visualização). Ambas as ações processam um *update* à *on-site* gravada inicialmente. No passo seguinte, a UI apresenta o editor gráfico de páginas e formulários, onde o utilizador cria a sua *on-site message* através de um editor *drag and drop*, que grava automaticamente periodicamente o estado atual da edição. No passo seguinte, o utilizador tem a possibilidade de ativar a *on-site message* no seu domínio. Caso o faça, o Services comunica com o Matomo, que adiciona a tag com o aspeto, tipo de *trigger* e condição de visualização da *on-site message* ao container do respetivo domínio. Por último, o utilizador confirma a criação da *on-site message* e é redirecionado para a listagem de páginas e formulários, onde já estará a *on-site message* criada.

4.5 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 6 *endpoints* essenciais no componente Services:

GET /forms/onsite - Obter todas as *on-site messages* de um utilizador;

GET /forms/onsite/:id - Obter uma *on-site message* com um determinado id;

POST /forms/onsite - Criar uma *on-site message*;

PUT /forms/onsite/:id - Atualizar uma *on-site message* com um determinado id;

PUT /forms/onsite/widget/:id - Atualizar os *widgets* que compõe uma *on-site message*;

DELETE /forms/onsite/:id - Eliminar uma *on-site message* com um determinado id.

Sendo o Services uma REST API em que todos os pedidos são feitos por HTTP, 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.

4.6 Design de 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 UXA (*User Experience and Assurance*), responsável por desenvolver os *mock-ups* das novas funcionalidades a serem desenvolvidas, assim como testar e validar a solução final de cada uma (sejam *features* novas ou resolução de *bugs*). Todos os *mock-ups* para a interface gráfica da solução final deste projeto foram desenvolvidos pela equipa de UXA da E-goi.

Na Figura 4.6 está representado o *mock-up* da página de opções de uma *on-site* message. A página de opções é o segundo passo do fluxo de criação de uma *on-site message*, e responde às necessidades do requisito funcional Consultar/Editar opções da On-site Message, definido na subsecção 3.1.1. Como é possível verificar, a interface gráfica apresenta vários campos para o utilizador preencher, sendo eles o nome interno, o idioma, o acionador e a condição de visualização. O nome interno é apresentado como um *input* de texto, onde o utilizador define o nome da sua *onsite message*. 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. O acionador é o tipo de *trigger* que o utilizador pretende que faça a sua *on-site message* aparecer no ecrã da sua página *web*. É apresentada uma *select-box* com os tipos: ao ver a página, ao percorrer a página (este é o exemplo apresentado na Figura 4.6,

que requer ainda a percentagem de página percorrida para que o *trigger* seja despoletado), 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 da *on-site message*, com as opções: mostrar sempre, mostrar uma vez por cada visita, mostrar uma vez por dia, mostrar apenas uma vez e ainda mostrar num intervalo de datas. No fundo da página de opções, está desenhado um *footer* que permite ao utilizador regressar à página anterior ou avançar para a página seguinte.

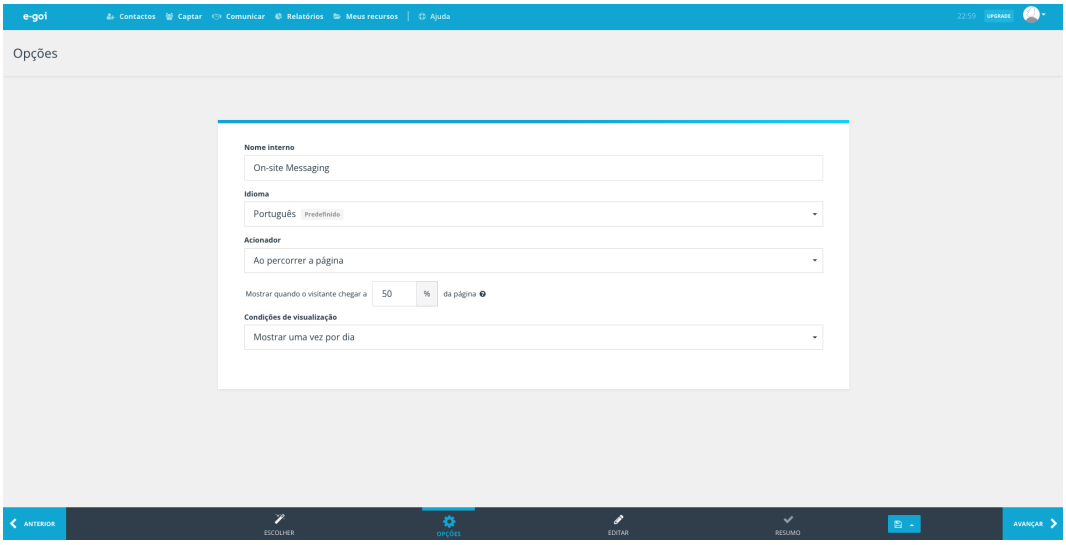

Figura 4.6: Mock-up da página de opções

Na Figura 4.7 está representado o *mock-up* da página que contém o editor gráfico de páginas e formulários do produto E-goi. O editor deve mostrar o espaço disponível para editar, em conformidade com o tipo de *on-site message* escolhido pelo utilizador. Na Figura 4.7 é apresentado o exemplo do editor gráfico para o tipo *banner*. O utilizador tem disponível a opção de configurações no canto superior esquerdo, que abre uma modal com duas opções, que para este tipo de *on-site* são o tamanho (pequeno, médio ou grande) da *on-site* e a posição (cimo da página ou fundo da página) onde quer que apareça na página *web*. No canto superior direito é possível editar o nome da *on-site message*. Do lado esquerdo da página, estão todos os *widgets* que o utilizador pode adicionar à sua *on-site*, que se encontra no centro da página, através do sistema *drag and drop*, que vão desde imagens, texto, vídeos, produtos, botões, entre outros. Do lado direito, é possível ao utilizador editar os estilos dos *widgets* que adicionou. No fundo da página de edição, está desenhado um *footer* que permite ao utilizador regressar à página anterior ou avançar para a página seguinte. Este *mock-up* mostra como é possível cumprir o requisito funcional UC-2 Editar On-site Message, definido na subsecção 3.1.1.

Os restantes *mock-ups* desenvolvidos para este projeto encontram-se apresentados no Anexo B.

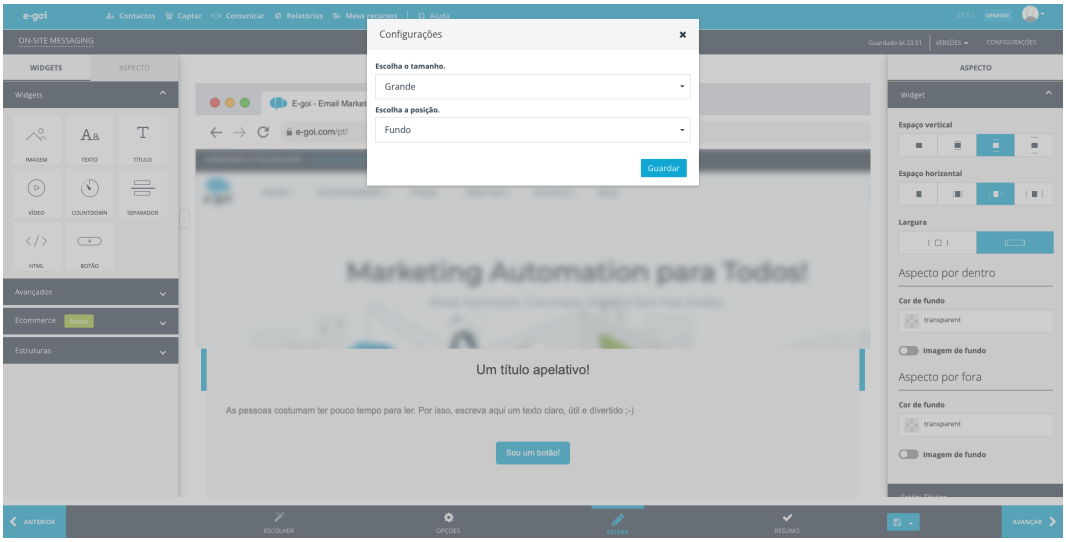

Figura 4.7: Mock-up da página do editor gráfico

4.7 Sumário

Neste capítulo foi apresentado todo o desenho da solução final, desde a modelação do domínio, a arquitetura dos componentes do sistema, juntamente com uma arquitetura alternativa, a implantação, o fluxo de utilização na criação de uma *on-site message*, novos recursos necessários para a criação e gestão das *on-site messages* e ainda dois *mock-ups* que ilustram a interface gráfica da solução final.

O capítulo seguinte refere-se à implementação da solução, onde é apresentado todo o desenvolvimento da solução a partir do desenho descrito neste capítulo.

Capítulo 5

Implementação da Solução

Neste capítulo é apresentada a implementação da solução, com base no desenho definido no capítulo anterior. São inicialmente descritas as metodologias utilizadas para o desenvolvimento do projeto, seguido de uma apresentação técnica de toda a implementação em cada componente da arquitetura da solução.

5.1 Metodologia de Desenvolvimento

A qualidade do desenvolvimento de *software* é muito importante para garantir que o produto final satisfaz os requisitos do cliente. Para que uma equipa de desenvolvedores consiga trabalhar em sintonia e com eficiência, necessita de adotar métodos de trabalho que possibilitem essas condições. Para o desenvolvimento deste projeto, mesmo sendo maioritariamente de caráter individual, foram postas em prática as metodologias da utilizadas na empresa.

Como sistema de controlo de versões dos projetos do produto, a E-goi utiliza a ferramenta Git, apresentada na Secção 2.6. Este sistema é utilizado pela E-goi através da aplicação GitLab [58], que permite aos seus utilizadores armazenarem o código dos seus projetos em servidores próprios, o que facilita a gestão de todo o código dos projetos do produto. Cada projeto possui 3 ramos base, uma onde é feito todo o desenvolvimento de código, onde são feitos os testes ao código desenvolvido e onde é colocado o código final, disponível para o utilizador. Para o realização deste projeto, todo o código foi inicialmente desenvolvido no ramo de desenvolvimento, seguidamente enviado para o ramo de testes e, sendo aprovado no ramo de testes, foi então enviado para o ramo de produção, ficando assim disponível para o utilizador final. O código é enviado de um ramo para outra através de um *merge*, que junta as alterações feitas num ramo a outro ramo diferente.

De modo a fazer uma integração contínua do projeto, descrita na Secção 2.6, a E-goi utiliza a ferramenta Jenkins. O Jenkins é um servidor de automação de código aberto independente, que pode ser usado para automatizar todos os tipos de tarefas relacionadas à construção, teste e entrega ou implantação de *software* [59]. Está 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 é enviado código novo do ramo de desenvolvimento para a de testes, ou da de testes 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, ferramenta também mencionada na Seccção 2.6. Por fim, é realizado o *deploy*, operação que deixa o código disponível para o utilizador.

Em suma, a utilização destas metodologias, facilita o processo de desenvolvimento, desde a gestão do código desenvolvido, a garantia da sua qualidade, o uso de boas práticas de programação, a ausência de *bugs* e a cobertura de testes. Estas métricas são apresentadas mais detalhadamente na Secção 6.4.

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.

Para que fosse possível persistir todos os dados relativos às *on-site messages*, foi criada uma nova tabela na base de dados, a **egoi_lp_onsite**. Na Figura 5.1 está representada a nova tabela criada e as suas respetivas colunas, assim como a sua ligação à parte do modelo de dados já existente.

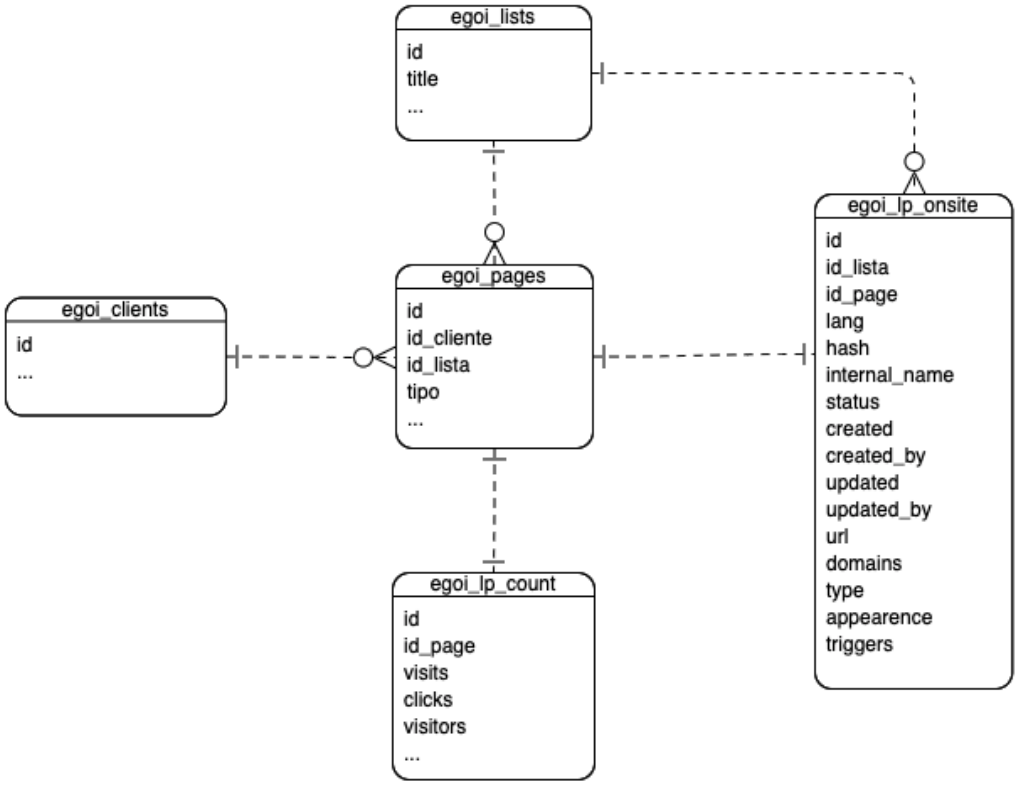

Figura 5.1: Camada de Persistência

Na Tabela 5.1 é possível consultar a legenda da Figura 5.1.

Tabela 5.1: Definição das colunas da tabela egoi_lp_onsite

Coluna	**Descrição**
id	id da on-site message
id_lista	id da lista onde a on-site message está associada
id_form	id da tabela maxmailing_forms correspondente à on-site message
lang	Idioma da on-site message
internal_name	Nome único da on-site message
status	Estado da on-site message (pode ser ativo ou inativo)
created	Data em que a on-site message foi criada
created_by	id do utilizador que criou a on-site message
updated	Data em que a on-site message foi atualizado
updated_by	id do utilizador que atualizou a on-site message pela última vez
url	Link público da on-site message
domains	Domínios onde a on-site message está ativada
type	Tipo de on-site message
appearence	Dados relativos à aparência da on-site message
triggers	Tipo de trigger e condições de visualização da on-site message

A tabela egoi_clients contém as informações de todos os utilizadores de uma conta E-goi, identifica por um id, que pode ter uma ou mais páginas ou formulários a si associados. A tabela que representa todas as páginas e formulários da conta do cliente é a egoi_pages. Esta tabela é global e serve de listagem para todas as páginas e formulários do produto E-goi, havendo a ela associada uma tabela de cada tipo de página e formulário. Cada elemento da tabela possui, entre outros, um id de identificação, um id cliente e id lista ao qual está associado e um tipo que define o tipo de página e formulário, que pode ser 'form', 'signup', 'landing', 'embed', 'whatsapp' e, no caso deste projeto, 'onsite'. A tabela egoi_pages está também ligada à tabela egoi_lists, o que significa que cada página e formulário está associado a uma única lista. A tabela egoi_lp_count contém todos os registos de número de visitas, cliques e visitas únicas de cada página e formulário.

A tabela de persistência de dados relativos a uma *on-site message* é a **egoi_lp_onsite**, que na tabela egoi_pages representa uma página do tipo 'onsite'. Todos os dados guardados nesta tabela, são necessários aos componentes que envolvem esta solução, sendo o componente Services, o responsável por garantir a persistência dos mesmos na base de dados do sistema.

O componente Services é também responsável por enviar o tipo de *trigger*, as condições de visualização e o código HTML da *on-site message*. Este código vai ser persistido no componente Matomo, que possui uma base de dados interna cuja estrutura não é conhecida.

5.3 Camada de Negócio

O componente Services é uma REST API desenvolvida em PHP e utiliza a *framework* Zend Framework [60], 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 5 dos *endpoints* referidos na Secção 4.5 na classe OnsiteMessageController, criada 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:

- GET /forms/onsite - getList(), que devolve a lista de todas as *on-site messages* do utilizador;
- GET /forms/onsite/:id - get($id), que devolve a *on-site message* com o $id recebido;
- POST /forms/onsite - create($data), que cria uma *on-site message* com base nos dados recebidos na variável $data;
- PUT /forms/onsite/:id - update($id, $data), que atualiza a *on-site message* com o id recebido com os dados recebidos na varável $data;
- DELETE /forms/onsite/:id - delete($id), que elimina a *on-site message* com o id recebido.

Foi também criada a classe OnsiteMessageWidgetController, para implementar o *endpoint* PUT /forms/onsite/widget/:id, que é responsável por atualizar os *widgets* de uma *on-site message* na base de dados, sempre que o utilizador realiza uma edição gráfica.

Foram implementados métodos de filtragem para garantir que a informação recebida nas variáveis está dentro dos formatos definidos, bem como um *transformer*, que transforma os dados obtidos da base de dados, mapeando-os num *array*, simplificando o seu tratamento na camada de apresentação, como é possível verificar no excerto de código 5.1.

```
public function transform($data)
    {
        return [
            'id' => $data->getId(),
            'listId' => $data->getListId(),
            'internalName' => $data->getInternalName(),
            'formLang' => $data->getFormLang(),
            'created' => $data->getCreated(),
            'createdBy' => $data->getCreatedBy(),
            'updated' => $data->getUpdated(),
            'updatedBy' => $data->getUpdatedBy(),
            'hash' => $data->getHash(),
            'formId' => $data->getFormId(),
            'type' => $data->getType(),
            'url' => $data->getUrl(),
            'domains' => $data->getDomains(),
            'appearance' => $data->getAppearance(),
            'triggers' => $data->getTriggers()
        ];
    }
```

Listing 5.1: Transformador de dados em PHP

Os transformadores contêm a lógica de negócios para alterar o formato de um modelo para o que for necessário para a saída de dados. O desacoplamento dessa lógica do modelo

permite mudanças de esquema discretas e garante um local confiável para todas as lógicas de operações de formatação de dados.

5.3.2 Construção do aspeto da On-site Message

Foi implementado um serviço OnsiteMessageService, que faz a ligação entre o controlador e a base de dados, onde se encontra a lógica de processamento de negócios principal, que é chamado pelo OnsiteMessageController para obter ou atualizar o modelo de dados. É neste serviço que é construída toda a aparência de uma *on-site message*. Para essa construção, foi utilizada a linguagem de marcação HTML, utilizada para a construção de páginas *web*, uma vez que é a linguagem aceite pelo Matomo para despoletar conteúdo numa página *web*. Foram implementados 5 tipos de *on-site message*: caixa pop-up, barra lateral, *banner* suspenso, ecrã total e *slider*. Para cada tipo de *on-site*, foi desenvolvido dinamicamente código HTML, que recebe da interface gráfica a aparência definida pelo utilizador.

No excerto de código 5.2, é apresentado um exemplo de como é aplicada a aparência de uma *on-site message* do tipo caixa pop-up. No tipo caixa pop-up, o utilizador define no editor gráfico, o tamanho que pretende que a sua *on-site message* tenha, podendo escolher entre pequeno, médio ou grande. A partir da escolha do utilizador, as dimensões são guardadas em pixeis na variável $appearance e depois recebidas pela função getModalScript, que as aplica ao estilo CSS da *on-site message*. CSS é um mecanismo que permite adicionar estilo a um documento *web*.

```
1  public function getModalScript($url, $appearance, $hash)
2  {
3          ...
4      .onsiteModal-content-'.$hash.' {
5           background-color: #fefefe;
6           position: fixed;
7           border: 1px solid #888;
8           width:'.$appearance['width'].'px;
9           height:'.$appearance['height'].'px;
10          overflow: hidden;
11       }
12         ...
13 }
```

Listing 5.2: Exemplo de estilo css aplicado a uma on-site message

A variável $hash é adicionada ao nome da classe CSS, para garantir que tem um nome único, de forma que não haja conflitos de estilos com outras classes que possam ter o mesmo nome.

Ainda nos estilos CSS da *on-site message*, foram implementadas *media queries*, um recurso do CSS que permite a renderização de conteúdo para se adaptar a diferentes condições, como a resolução do ecrã do dispositivo onde é apresentado o conteúdo da *on-site message*. Com esta solução, é possível garantir a responsividade em qualquer tipo de dispositivo, o que permite cumprir um dos requisitos de usabilidade da Secção 3.1.2. No excerto de código 5.3, é apresentado um exemplo de uma *media query* para adaptar o tamanho da *on-site* a dispositivos com largura inferior a 800 pixeis.

```
public function getModalScript($url, $appearance, $hash)
{
        ...
    @media (max-width: 800px) {
        .onsiteModal-content-'.$hash.'{
            max-width: 50%;
            max-height: 60%;
        }
    }
        ...
}
```

Listing 5.3: Exemplo de media query aplicada a uma on-site message

O corpo do código HTML, é onde está todo o conteúdo de um documento HTML, como títulos, parágrafos, imagens, *links*, tabelas, listas, entre outros. Para a construção da *on-site message* do tipo caixa pop-up, foi desenvolvido o código apresentado no excerto 5.4.

```
public function getModalScript($url, $appearance, $hash)
{
        ...
    <div id="modalOnsite-'.$hash.'" class="onsiteModal-'.$hash.'">
        <div class="onsiteModal-content-'.$hash.'">
            <span class="closeOnsiteModal-'.$hash.'">&times;</span>
            <iframe id="iframeOnsite-'.$hash.'" src="'.$url.'"
            style="border: 0; margin: 0; width: 100%; height: 100%;">
            </iframe>
        </div>
    </div>
        ...
}
```

Listing 5.4: Exemplo de estilo

Uma define uma divisão ou secção do documento HTML, usada como um *container* para elementos HTML. Para o tipo de *on-site* caixa pop-up foram criadas duas *divs*, a primeira que contém todo o conteúdo em torno da caixa pop-up na página *web* e a segunda, que está dentro da primeira, tem o conteúdo dentro da caixa pop-up.

A caixa pop-up possui um , que permite adicionar o elemento X ao canto superior direito da caixa, que aquando do clique do mesmo, fecha a *on-site message*. Possui também um <iframe>, que renderiza o conteúdo do url recebido na variável $url da função getModalScript, que contém a *on-site message* criada pelo utilizador no editor gráfico.

O componente Public do Builders, representado na Figura 4.2, possui uma aplicação Angular que é carregada e responsável por renderizar os *widgets* da *on-site message*. O url está sempre direcionado para a máquina onde está a correr o servidor do Builders. O acesso ao url retorna a aplicação Angular, que é carregada dentro do <iframe> e que faz um pedido HTTP ao *endpoint* GET /page/widget/:id do componente Services, para retornar os *widgets* da *on-site message* e renderiza-los, mostrando a *on-site* dentro da caixa pop-up.

Na Figura 5.2 é apresentado um exemplo de como uma *on-site message* do tipo caixa pop-up aparece numa página *web*.

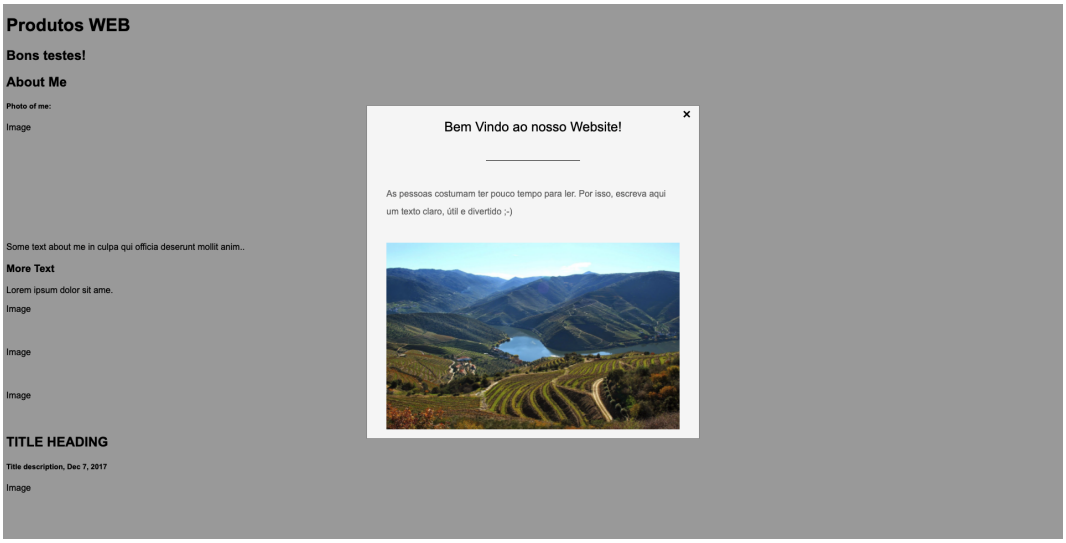

Figura 5.2: Exemplo de uma on-site do tipo caixa pop-up

5.3.3 Integração com o Matomo

O Matomo disponibiliza uma API que, há semelhança do Services, utiliza o protocolo HTTP para todas as chamadas às suas funções. Uma vez que atualmente o Matomo já se encontra integrado com o sistema E-goi, existe uma classe *controller* onde as chamadas à API do Matomo são feitas, a MtmConfigsController. O serviço TagManagerService trata toda a lógica de processamento de dados entre a base de dados do sistema E-goi e o *controller*.

Atualmente, no sistema E-goi, cada domínio do utilizador possui um *container* único associado, onde é possível definir tags e *triggers*. As tags contêm o código HTML a ser executado numa página *web* e os *triggers* contêm o tipo de ação do visitante de uma página *web* que faz com que a tag mostre o código HTML no ecrã do visitante.

Para adicionar tags e *triggers* de uma *on-site message* ao Matomo e consequentemente ativar uma *on-site message*, foi implementada a função onsiteMessageTagSetup($clientId, $onsiteMessageId, $domain), no TagManagerService. Esta função recebe como parâmetros o id do utilizador, o id da *on-site message* que o utilizador pretende ativar e o domínio onde esta deve ser ativada.

Quando o utilizador ativa uma *on-site message*, o componente BO-UI faz um pedido HTTP ao Services através do *endpoint* POST /matomo/tagmanager/configs, enviando o id do utilizador, o id da *on-site* e o domínio. O pedido chega ao MtmConfigsController, que faz uma chamada à função onsiteMessageTagSetup do TagManagerService, enviando-lhe os dados que recebeu do componente BO-UI.

```php
public function onsiteTagSetup($clientId,$formId,$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->container['idcontainer'],
        'type' => $type,
        'name' => $name,
        'parameters' => $parameters
    ];

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

Listing 5.5: Função que permite adicionar trigger ao Matomo

No exemplo apresentado no excerto de código 5.5, é possível verificar como é adicionado um *trigger* ao Matomo. A função onsiteTagSetup faz uma chamada à função addContainerTrigger, enviando o id do utilizador, o domínio, o tipo de *trigger* e os parâmetros que contêm informações adicionais ao tipo de *trigger*. Por exemplo, se for o *trigger* de deslocamento atingido, os parâmetros possuem a percentagem de deslocamento. De seguida, a função addContainerTrigger constrói o *array* $data, onde define o método do Matomo ao qual vai fazer o pedido, o id do *container* onde deve ser adicionado o *trigger* e o tipo, nome e parâmetros do *trigger*. Por fim, retorna o pedido ao Matomo, que adiciona os dados recebidos ao domínio do utilizador.

O passo seguinte a adicionar o *trigger* ao Matomo, é adicionar a tag. No excerto de código 5.6 é possível verificar como é que uma tag é adicionada ao Matomo. A função onsiteTagSetup faz uma chamada à função addContainerTag, enviando o id do utilizador, o domínio, o nome da tag, o código HTML da *on-site message* a ser executado (todo o processo da sua construção apresentado na Secção 5.3.2), o *trigger* adicionado anteriormente, o tipo de tag, o limite de execução da tag (por exemplo, ilimitado ou uma vez por dia), a data de início de ativação da tag e a data em que a tag deve ser desativada. De seguida, a função addContainerTag, tal como no exemplo 5.5, constrói o *array* $data com todos os dados recebidos e retorna o pedido ao Matomo, que adiciona por completo a *on-site message* ao domínio do utilizador, finalizando o processo.

```php
public function onsiteTagSetup($clientId,$formId,$domain)
{
    ...
    $this->addContainerTag(
        $clientId,
        $domain,
        'OnsiteMessage',
        ['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->container['idcontainer'],
        'name' => $name,
        'parameters' => $parameters,
        'trigger' => $trigger,
        'type' => $type,
        'fireLimit' => $fireLimit,
        'startDate' => $startDate,
        'endDate' => $endDate
    ];
    return $this->makeMatomoRequest($data,$clientId,$domain);
}
```

Listing 5.6: Função que permite adicionar uma tag ao Matomo

5.3.4 Códigos de personalização

Para que o utilizador consiga personalizar mensagens utilizando os códigos de personalização do sistema E-goi, o primeiro requisito é o visitante da sua página já estar inscrito na lista ao qual a *on-site message* está associada. Para ser inscrito na lista, o visitante tem de ter submetido um formulário do produto E-goi, que pertença à mesma lista da *on-site message*. Sendo um elemento da lista de contactos, o visitante tem a ele associado uma *cookie* permite identifica-lo.

```javascript
public readSubscriberCookie(): string {
    const cookie = document.cookie.split('; ').find(row => row.
    startsWith('eg_subs'));
    return cookie ? cookie.split('=')[1] : null;
}
```

Listing 5.7: Função que permite obter cookie do visitante

A leitura da *cookie* de um visitante é feita pela função apresentada no excerto de código 5.7, implementada no serviço do componente Builders. O formato da *cookie* possui o nome 'eg_subs' e uma chave, que é o id do visitante.

Há vários códigos de personalização possíveis de utilizar, como por exemplo, !fname, !email, !telephone, !birth_date, entre outros. O utilizador pode escrever uma mensagem de texto na sua *on-site message* com um dos códigos. Por exemplo "Bem-vindo à nossa página !fname. Sabemos que festeja o seu aniversário no dia !birth_date...". A aplicação Angular do Builders, carregada pelo url da *on-site*, vai renderizar todo o conteúdo da mesma e, ao verificar que na mensagem contém um código de personalização, através da *cookie*, obtém o id do visitante. Com essa informação, faz um pedido HTTP ao componente Egoi-public, responsável por receber todos os pedidos da parte pública das *on-site messages*, para obter toda a informação relativa à *on-site* e, caso seja um visitante já existente na lista, retorna também os seus dados. O Egoi-public, por sua vez, efetua um pedido HTTP ao Services com o id da *on-site message* e do visitante e recebe como resposta a informação desejada. Tendo os dados do visitante, a mensagem do utilizador é apresentada com os códigos substituídos pela respetiva informação. No exemplo abordado, o código !fname seria substituído pelo nome do visitante e o !birth_date pela sua data de aniversário. No caso de não existir informação ou não existir forma de identificar o visitante, os códigos são removidos e mantém-se a mensagem definida pelo utilizador, para que a experiência do visitante seja melhor e não fique afetada caso seja um visitante que não pertence à lista.

5.3.5 Registo de Resultados

O utilizador tem a possibilidade de consultar resultados referentes a visitas, visitas únicas e cliques das suas *on-site messages*. As visitas únicas são visitas cujo visitante não é identificado pela *cookie*. Para que isso seja possível, é necessário guardar esses dados cada vez que um visitante vê uma *on-site message*. Sempre que o url de uma *on-site message* é carregado, o componente Builders faz um pedido HTTP ao componente Egoi-public que regista a ação de visita. Para que a visita seja guardada na base de dados do sistema, o Egoi-public faz um pedido HTTP ao componente Services com o id da *on-site message* visitada. Por sua vez, o Services, como responsável pela comunicação com a base de dados, regista a visita, incrementando o valor na coluna de visitas da respetiva *on-site message*, na tabela egoi_lp_count.

5.4 Camada de Apresentação

Os componentes que representam a camada de apresentação do sistema são o BO-UI e o Builders. 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.

5.4.1 Adaptação do Editor Gráfico

Um dos requisitos de usabilidade desta solução, é garantir que o editor gráfico ajusta o seu espaço de edição em conformidade com o tipo de *on-site message*. Assim, o utilizador consegue editar a sua *on-site message*, tendo sempre perceção do espaço que vai ocupar e da posição onde esta vai ficar na página *web*. Para o utilizador conseguir definir os tamanhos da sua *on-site* foi implementada uma opção de configurações de tamanho, onde pode escolher os vários tamanhos disponíveis para o tipo de *on-site* que selecionou incialmente. Foi também adicionado uma imagem de fundo ao editor, com o objetivo de simular uma página *web*, para dar a entender ao utilizador o resultado mais aproximado possível do aspeto da *on-site message* no seu *site*.

Na Figura 5.3 é possível perceber o comportamento esperado do editor gráfico para o tipo caixa pop-up.

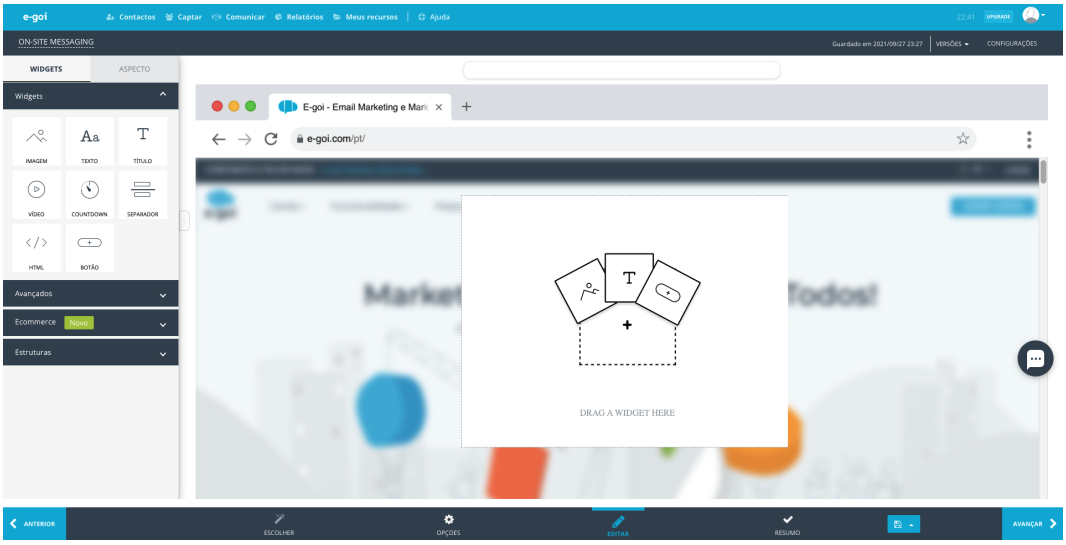

Figura 5.3: Exemplo do editor gráfico para o tipo caixa pop-up

Quando o utilizador avança para o passo de edição, o componente Editor do Builders, faz um pedido HTTP ao Services, para obter o tipo de *on-site message* definido inicialmente pelo utilizador e adapta as dimensões do espaço de edição em conformidade.

5.4.2 Fluxo de Gestão

Toda a interface gráfica de gestão das *on-site messages*, foi implementada no componente BO-UI, com base nos *mock-ups* apresentados na Secção 4.6. Para o desenvolvimento desta interface gráfica, foi utilizada a *framework* AngularJS [55], uma vez que todo o projeto do componente BO-UI, já se encontra desenvolvido a partir da mesma.

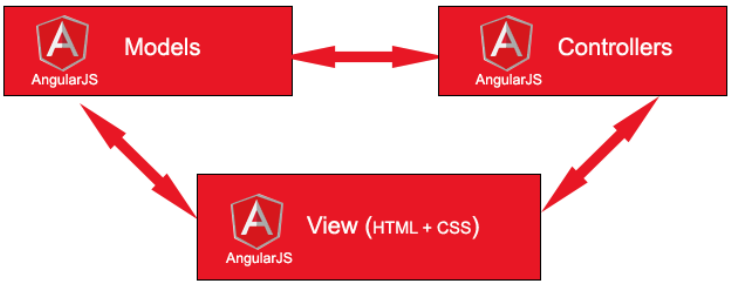

Figura 5.4: Arquitetura AngularJS

Uma aplicação AngularJS tem por base uma arquitetura MVC (*Model-View-Controller*), 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* [61].

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 4.6 tem o seu próprio ficheiro *controller*, desenvolvido em JavaScript e a respetiva *view* desenvolvida em HTML e CSS. O fluxo de gestão de uma *on-site message*, manipula os dados da mesma, 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 da *on-site* na base de dados do sistema.

5.5 Sumário

Neste capítulo foi explicada toda a implementação da solução, desde a camada de dados até à camada de apresentação. Inicialmente foi descrita toda a metodologia utilizada para o desenvolvimento do projeto. De seguida, foi descrito o desenvolvimento da solução em cada camada do produto E-goi, passando pela nova tabela criada na base de dados do sistema e as suas relações com as tabelas já existentes na camada de dados. Foi também abordada a construção de uma *on-site message* e a sua integração com o Matomo na camada de negócio. Por fim, foi descrito o desenvolvimento da interface gráfica na camada de apresentação.

Terminada a implementação, é necessário realizar algumas experiências e testes à solução final, com o objetivo de avaliar a qualidade da mesma. Todos as experiências e avaliações são apresentadas no capítulo seguinte.

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 origem nos problemas apresentados na Secção 1.3 e nos objetivos definidos na Secção 1.4. Tem-se como fator de avaliação, para este caso, a qualidade do *software* desenvolvido. 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 Especificação da Hipótese

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

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

- O código desenvolvido para o novo canal de comunicação, tem a qualidade suficiente para ser integrado no produto E-goi.

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

- A média das classificações das respostas do inquérito de satisfação e usabilidade é superior a 3.5.

6.3 Metodologias de Avaliação

Definidas as métricas e as suas respetivas hipóteses, é altura de escolher metodologias, para avalia-las. Para a avaliação da métrica de qualidade do *software* desenvolvido, foram realizados testes de integração e uma análise ao relatório da *framework* SonarQube [62]. Para avaliar a métrica de satisfação do utilizador, foi realizado um inquérito de satisfação e usabilidade, que segue os requisitos do sistema e que permite perceber se os utilizadores

efetivamente conseguiram utilizar o produto sem qualquer problema, a colaboradores internos da empresa e também a elementos externos.

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

Para avaliar a qualidade do *softwate* do sistema desenvolvido, foram realizados testes de integração e testes de qualidade ao código.

Os testes de integração são testes feitos às rotas de comunicação entre as diferentes partes do módulo do sistema, para mostrar que todos os módulos funcionam corretamente juntos. Para a realização destes testes foi utilizada a plataforma Postman [63]. O Postman é uma ferramenta que dá suporte à documentação dos pedidos feitos por uma API e possui um ambiente para a documentação, execução de testes de APIs e pedidos em geral.

O objetivo destes testes, consistiu em cobrir os métodos da camada de negócio implementados, que fazem chamadas à base de dados do sistema. Para isso, foram executados testes para todos os *endpoints* referidos na Secção 5.3.1.

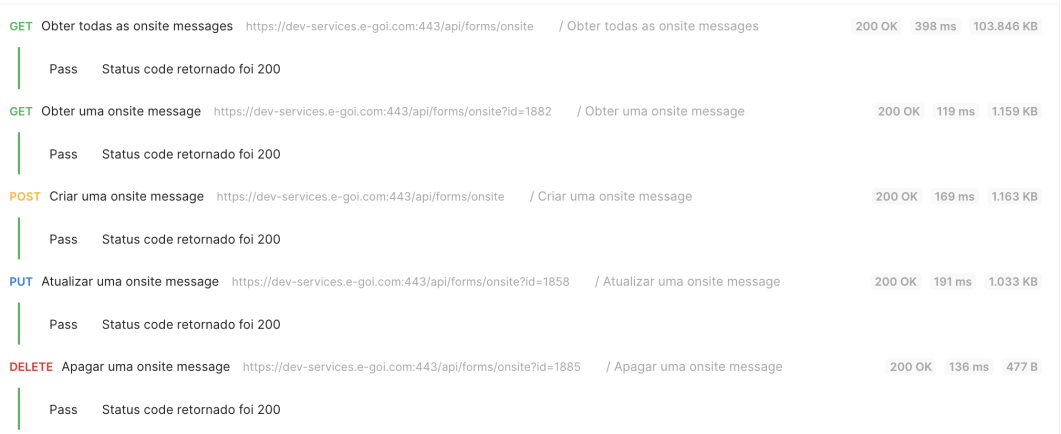

Figura 6.1: Resultado dos testes de integração no Postman

Na Figura 6.1 são apresentados os resultados dos testes de integração realizados. Para que todos os testes passassem o código de resposta retornado deveria ser 200. O código HTTP 200 é a resposta de estado de sucesso, que indica que o pedido foi bem sucedido. Pela análise dos mesmos, é possível concluir que todos passaram com sucesso.

Para a realização dos testes à qualidade do código desenvolvido, foi utilizada a *framework* SonarQube [62]. O SonarQube é uma ferramenta automática de revisão de código para detectar *bugs*, vulnerabilidades e más práticas de programação. Os testes são feitos de forma automática ao código existente no repositório e é gerado um relatório de resultados.

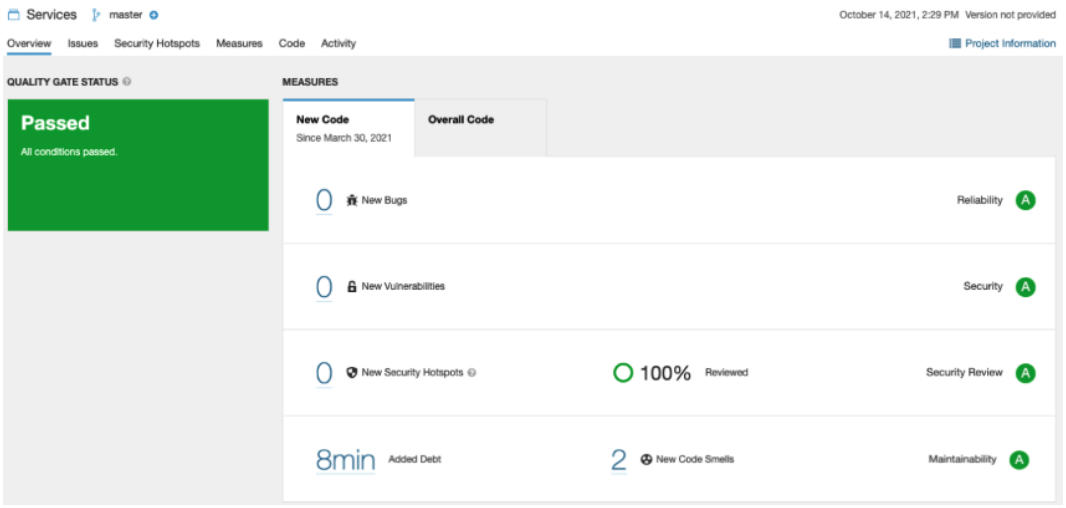

Figura 6.2: Relatório de resultados do SonarQube

Na Figura 6.2 está representado o relatório do SonarQube do projeto Services, onde foi desenvolvida a camada de negócio da solução. Pela sua análise, é possível constatar que o código desenvolvido para este projeto está livre de *bugs* e vulnerabilidades e apenas contém 2 casos em que pode ser considerado existir má prática de desenvolvimento de *software*.

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 teve por base um inquérito apresentado no Anexo C. Inicialmente, foi feita uma apresentação da solução e das suas funcionalidades a alguns membros da empresa e a convidados externos, via *zoom* **??** e de seguida, foi-lhes pedido que respondessem ao inquérito de satisfação e usabilidade. A amostra total foi de cerca de 30 elementos, maioritariamente constituída por elementos da equipa de *marketing* e vendas e também a elementos externos à empresa, para garantir que pessoas sem conhecimento prévio da solução também conseguem usar o produto.

Este questionário utiliza uma escala baseada na escala Likert-Scale [64], que varia entre 1 a 4, sendo 1 correspondente a "Discordo totalmente", 2 correspondente a "Discordo", 3 correspondente a "Concordo"e 4 correspondente a "Concordo totalmente". O facto do número de opções de resposta ser um número par, obriga os elementos da amostra a tomar uma decisão mais positiva ou mais negativa, o que facilita a avaliação, uma vez que não há respostas neutras. Foram realizadas cerca de 15 questões, baseadas nos requisitos funcionais e não funcionais do sistema e na satisfação geral do utilizador, sendo elas:

1. Facilidade em criar uma *on-site message* e vê-la listada.
2. Facilidade em ativar uma *on-site message* no site.
3. Facilidade em editar as opções de uma *on-site message*.
4. Facilidade em editar a aparência de uma *on-site message* através do editor gráfico.
5. Facilidade em antever uma *on-site message* listada.
6. Facilidade em apagar um *on-site message* listada.

7. Facilidade em ver a *on-site message* no site segundo o trigger que definiu.
8. Facilidade em personalizar mensagens de texto através dos códigos de personalização.
9. Facilidade em utilizar o editor gráfico para diferentes tipos de on-site messages.
10. Facilidade em visualizar a *on-site message* em qualquer tipo de dispositivo, não apresentando problemas de responsividade.
11. Facilidade em reconhecer alterações de estado do sistema.
12. O tempo de espera entre a passagem de uma página do fluxo para outra é aceitável.
13. Considera o sistema simples e fácil de utilizar.
14. Está satisfeito com o novo canal de *on-site messaging*.
15. Recomendaria o novo canal de *on-site messaging* a outras pessoas.

No Figura 6.5 é apresentado um gráfico de colunas empilhadas, com todos os resultados do inquérito realizado.

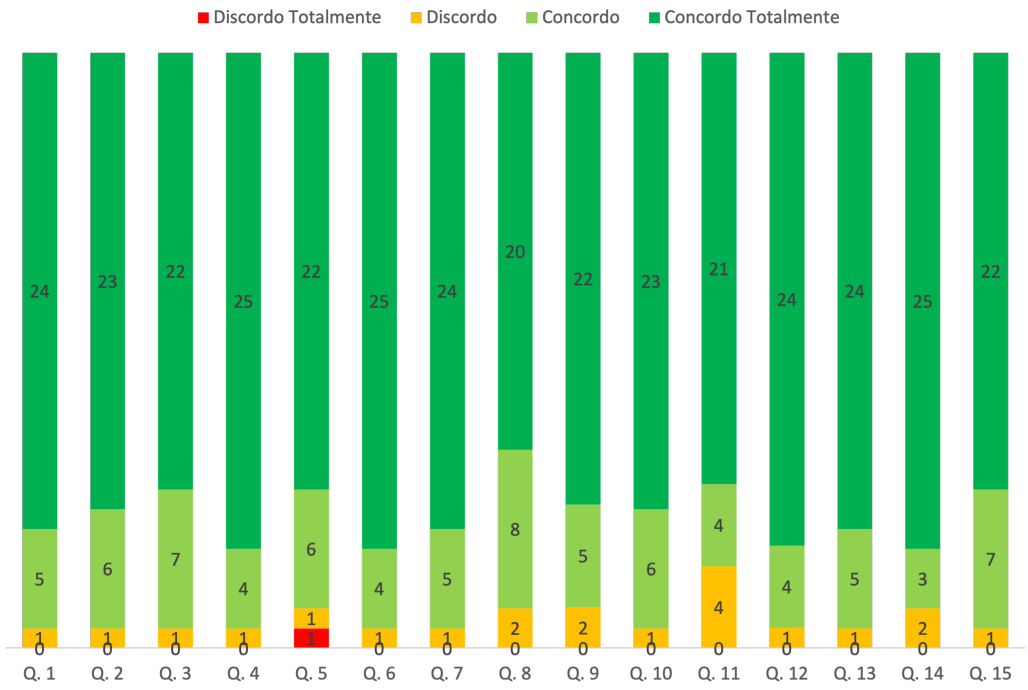

Figura 6.3: Resultados do inquérito de satisfação e usabilidade

Cada coluna do gráfico apresentado representa uma questão do inquérito e as diferentes respostas dos 30 participantes ao mesmo, com cores diferentes. Para facilitar a análise, as questões podem ser agrupadas por temas, sendo as questões de 1 a 8, relacionadas com a facilidade de utilização das funcionalidades, as questões de 9 a 12, relacionadas com a facilidade em reconhecer o cumprimento dos requisitos não funcionais e por fim, as questões de 13 a 15, relacionadas com a satisfação geral do utilizador.

Relativamente às questões da facilidade de utilização das funcionalidades do sistema, pela análise do gráfico, é possível verificar que as respostas foram maioritariamente "Concordo totalmente", havendo apenas uma resposta "Discordo totalmente", na questão 5, que aborda

a facilidade de antever uma *on-site message*. Com esta informação, é possível concluir que, para a grande maioria dos participantes, foi fácil interagir com as funcionalidades do sistema desenvolvido.

Em relação às questões da facilidade em reconhecer o cumprimento dos requisitos não funcionais do sistema, é possível verificar que as respostas também foram maioritariamente "Concordo totalmente", mas que a questão 11, que aborda a facilidade em reconhecer alterações do estado do sistema, apresenta um total de 4 participantes com a resposta "Discordo".

Relativamente às questões de satisfação geral de utilização do sistema, é possível concluir que a maioria dos participantes se encontra satisfeito com a solução desenvolvida, havendo uma maioria clara de respostas com a opção "Concordo totalmente".

Para calcular a classificação média de cada questão aplicou-se a seguinte fórmula:

$$\frac{x_1 w_1 + x_2 w_2 + x_3 w_3 \ldots x_n w_n}{\text{Número total de respostas}}$$

Em que x representa a contagem de respostas por opção de resposta e y o peso da posição classificada.

Tabela 6.1: Tabela de classificações médias de cada questão

Q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C	3.8	3.7	3.7	3.8	3.6	3.8	3.8	3.6	3.6	3.7	3.6	3.8	3.8	3.8	3.7

Na Tabela 6.1 são apresentadas todas as questões do questionário e a sua respetiva classificação média de resposta. Pela sua análise, é possível aferir que a média em cada resposta foi superior a 3.5, que foi a meta inicialmente definida. A média mais baixa obtida, foi de 3.6 e a mais alta foi de 3.8, havendo pouca variação entre cada questão, o que mostra a consistência geral da solução desenvolvida. Através destes resultados, podemos concluir que os objetivos quanto à satisfação e usabilidade do sistema foram cumpridos.

6.5 Resultados do Produto

Depois de terminada a implementação da solução, o produto foi disponibilizado ao cliente na plataforma E-goi em Julho de 2021. Foram recolhidos dados das primeiras 12 semanas de utilização do novo canal de *on-site messaging*, dados esses referentes à adesão de clientes pagantes e a visualizações de *on-site messages*.

Na primeira semana que o produto esteve disponível na plataforma E-goi, cerca de 7 clientes pagantes utilizaram o produto. Nas 4 semanas seguintes, o número de clientes pagantes subiu de 7 para 39. No final das primeiras 12 semanas, foram registados cerca de 59 clientes pagantes a utilizar o novo canal de *on-site messaging*.

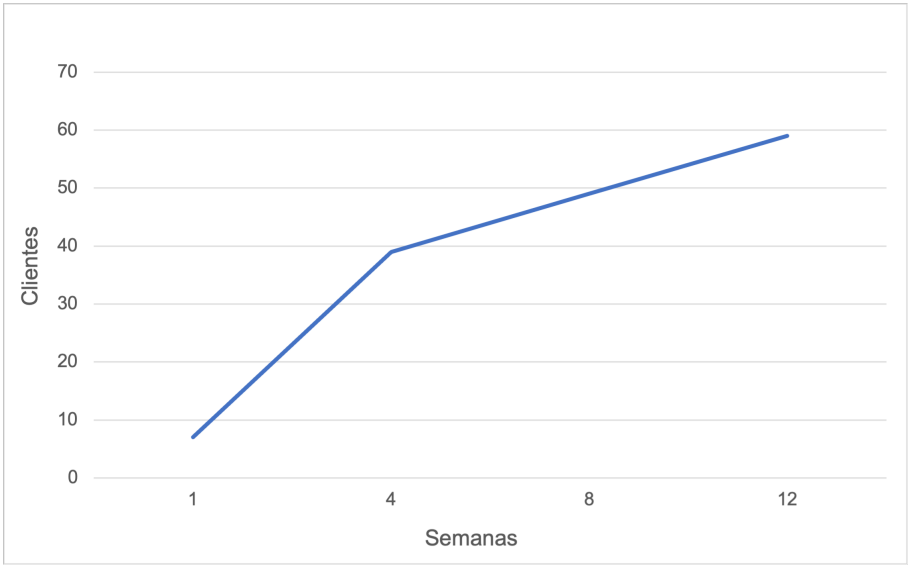

Figura 6.4: Evolução do número de clientes pagantes nas primeiras 12 semanas

Na Figura 6.4 é apresentado um gráfico que representa a evolução da adesão de clientes pagantes ao produto desenvolvido neste projeto nas primeiras 12 semanas que esteve disponível. Pela análise da linha do gráfico, é possível verificar que houve um grande aumento do número de clientes entre a primeira e quarta semana, o que significa que o produto teve um impacto positivo nos clientes E-goi. Da quarta semana até ao final do terceiro mês, não se verifica um crescimento tão elevado, contudo o número de clientes pagantes continuou a crescer.

Relativamente às visualizações de *on-site messages* por parte de visitantes de páginas *web*, na primeira semana houve um total de 150 visualizações. Nas 4 semanas seguintes o número de visualizações crescer para 543 e no final das 12 semanas registaram-se cerca de 3309 visualizações de *on-site messages*.

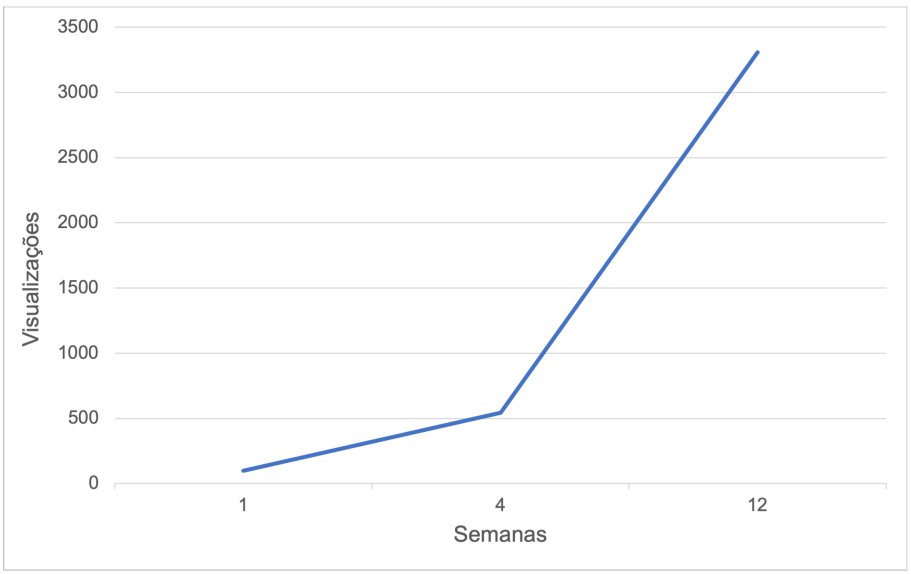

Figura 6.5: Evolução das visualizações nas primeiras 12 semanas

Na Figura 6.5 é apresentado um gráfico que ilustra o crescimento do número de visualizações ao longo das primeiras 12 semanas que o produto esteve disponível. Contrariamente ao número de clientes pagantes, o número de visualizações da primeira para a quarta semana não teve um crescimento tão elevado. Contudo, da quarta semana até ao final do terceiro mês, foi registado um grande aumento de visualizações de *on-site messages*.

Pelos resultados obtidos, é possível concluir que o produto foi bastante bem aceite pelos clientes E-goi, uma vez que se regista um crescimento contínuo do número de clientes pagantes. O facto de haver um crescimento de clientes pagantes a utilizar o produto e um aumento significativo de visualizações, mostra que o produto trouxe realmente valor para a empresa e que os objetivos para o qual foi desenvolvido foram cumpridos.

Capítulo 7

Conclusão

O projeto documentado, teve como objetivo, o desenvolvimento de um canal de comunicação *on-site messaging*. 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 de *on-site messages*.
2. Modelação de arquitetura robusta e escalável, de modo a poder ser adaptável a diferentes formas de distribuição de mensagens no cliente.
3. Implementação de funcionalidades específicas, tais como, diferentes tipos de *display* e *triggers*.

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 2.4, com a finalidade de perceber as ferramentas existentes atualmente no mercado e se estas seriam aplicáveis ao projeto. Feita esta análise, foram desenvolvidas várias funcionalidades que permitem o utilizador criar, editar e gerir as suas *on-site messages*, tal como apresentado na Secção 3.1. 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. Assim, foi possível desenvolver várias formas do utilizador distribuir as suas *on-site messages* aos seus clientes.

Por último, objetivo número 3 foi atingido através da implementação do sistema seguindo a arquitetura definida no Capítulo 4. Foram implementados vários tipos de *display*, caixa pop-up, barra lateral, *banner*, ecrã total e *slider*. Foram também implementados diferentes tipos de *triggers*, desde visualização, percentagem de *scroll*, intenção de saída, tempo de permanência na página e ainda com base no URL da página.

No Capítulo 6 foi avaliado o trabalho final e foi possível perceber que todas as hipóteses foram cumpridas. Foram ainda analisados os resultados da utilização do produto nas primeiras 12 semanas que esteve disponível para o cliente, apresentando um crescimento contínuo de utilização e consequentemente resultados bastantes positivos.

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 tipos de *trigger*, mas podem ser ainda implementados outros, como por exemplo o *trigger* de clique e de evento personalizado, enunciado na Secção 2.2.1, referente aos tipos de *trigger* suportados pelo Matomo. Podem também ser melhoradas as configurações de tamanho e posição de uma *on-site message*, de modo a dar mais liberdade ao utilizador.

O projeto integrará uma das equipas de desenvolvimento de produto da E-goi, que será responsável por garantir a sua manutenção e desenvolvimento futuro, seguindo o fluxo atual de trabalho e integração contínua.

Bibliografia

[1] Autor Carolina Silveira. *Guia do On-site Message: canal de marketing digital converte até 4x mais*. Abr. de 2019. url: http://blog.socialminer.com/people-marketing/guia-do-on-site-message-canal-de-marketing-digital-converte-ate-4x-mais/ (acedido em 16/12/2020).

[2] Jornal Público. *A fronteira entre o "online"e "offline"acabou: e-goi lidera a integração total das empresas 2020*. Dez. de 2020. url: https://www.publico.pt/2020/12/30/estudiop/artigo/fronteira-online-offline-acabou-egoi-lidera-integracao-total-empresas-1944565 (acedido em 29/12/2020).

[3] E. Díaz Bordenave Juan. *O que é comunicação*. Brasiliense, 1982. (Acedido em 09/02/2021).

[4] Venkatesh Shankar e Tarun Kushwaha. «Omnichannel marketing: Are cross-channel effects symmetric?» Em: *International Journal of Research in Marketing* (2020). (Acedido em 09/02/2021).

[5] Piedley Macedo Saraiva. «Marketing Digital: A Utilização Das Mídias Sociais como um Canal de Comunicação no Varejo de Moda de Barbalha-CE / Digital Marketing: The use of Social Media as a Communication Tool in the Fashion Retail Market in Barbalha, Ceará». Em: *ID on line REVISTA DE PSICOLOGIA* 13.44 (2019), pp. 486–507. (Acedido em 09/02/2021).

[6] *7 Excellent On-Site Messages That Increase Conversion*. url: https://www.salecycle.com/blog/strategies/7-awesome-on-site-messages-that-increase-conversion/ (acedido em 20/01/2021).

[7] *How retailers can use site conversion messages*. url: https://www.salecycle.com/blog/strategies/retailers-can-use-site-conversion-messages/ (acedido em 20/01/2021).

[8] *Matomo's History*. Jun. de 2020. url: https://matomo.org/history/ (acedido em 30/01/2021).

[9] *Google Analyitics*. url: https://analytics.google.com/analytics/web/provision/#/provision (acedido em 30/01/2021).

[10] Infopédia. *Prova: Definição ou significado de prova no Dicionário Infopédia da Língua Portuguesa*. url: https://www.infopedia.pt/dicionarios/lingua-portuguesa/prova?express=prova%20de%20conceito.

[11] *Reporting API Reference*. url: https://developer.matomo.org/api-reference/reporting-api.

[12] Indian Institute of Technology Kharagpur. url: http://www.iitkgp.ac.in/ (acedido em 20/01/2021).

[13] MoEngage. url: https://www.moengage.com/company/ (acedido em 20/01/2021).

[14] WisePops. *About us*. url: https://wisepops.com/about-us/ (acedido em 20/01/2021).

[15] *About us*. url: https://getsitecontrol.com (acedido em 20/01/2021).

[16] url: https://opencorporates.com/companies/cy/HE330173 (acedido em 21/01/2021).

[17] url: https://www.linkedin.com/company/hellobar/about/ (acedido em 21/01/2021).

[18] *MailMunch*. url: https://www.getapp.pt/software/107087/mailmunch (acedido em 21/01/2021).
[19] *Exit-Intent Popup Builder, Tool, and Software*. url: https://www.optimonk.com/ (acedido em 21/01/2021).
[20] *About Sleeknote: Engage Visitors with On-Site Messages*. Jan. de 2021. url: https://sleeknote.com/about (acedido em 29/01/2021).
[21] *About OptinMonster - Lead Generation Software for Marketers*. url: https://optinmonster.com/about/ (acedido em 29/01/2021).
[22] *Socital: onsite campaigns for ecommerce*. url: https://www.linkedin.com/company/socital/about/ (acedido em 30/01/2021).
[23] *About*. url: https://git-scm.com/about.
[24] Martin Flower. «Continuous Integration». Em: (2006).
[25] A. J. Russo. *PT*. 2003. url: https://aws.amazon.com/pt/devops/continuous-integration/.
[26] Arslan Aziz e Rahul Telang. «What Is a Digital Cookie Worth?» Em: *SSRN Electronic Journal* (2016). doi: 10.2139/ssrn.2757325.
[27] Kaspersky. *What are Cookies?* Jan. de 2021. url: https://www.kaspersky.com/resource-center/definitions/cookies.
[28] Romi Satria Wahono. «Analyzing Requirements Engineering Problems». Em: (2003), pp. 55–58.
[29] Adnan Rawashdeh e Bassem Matalkah. «A New Software Quality Model for Evaluating COTS Components». Em: *Journal of Computer Science* 2.4 (2006), pp. 373–381. doi: 10.3844/jcssp.2006.373.381.
[30] Bayer Jaroslav, Jan Hana Bydžovská e Géryk. «TOWARDS COURSE PREREQUISITES REFINEMENT». Em: *Journal of Computer Science* (2012), p. 26. url: https://is.muni.cz/publication/977628/IMEA_2012_Sbornik.pdf#page=28.
[31] David Hughes e Don Chafin. «Turning New Product Development into a Continuous Learning». Em: (). (Acedido em 10/02/2021).
[32] Nick Rich, Matthias Holweg e Dipl.Wirtschaftsing.(FH) MSc. «INNOREGIO: dissemination of innovation and knowledge management techniques». Em: (jan. de 2000), pp. 2–3. (Acedido em 10/02/2021).
[33] José Neves Adelino. «O Ponto de Partida: a Procura de Valor e a Inovação». Em: *Sociedade Portuguesa de Inovação* (1999).
[34] Peter Koen et al. «PROVIDING CLARITY AND A COMMON LANGUAGE TO THE "FUZZY FRONT END"». Em: ().
[35] Peter A. Koen, Heidi M. J. Bertels e Elko Kleinschmidt. «Managing the Front End of Innovation, Results From a Three-Year Study». Em: *Research-Technology Management* (2014), p. 34. (Acedido em 17/02/2021).
[36] Marcio Cardoso Machado. «Gestão do Processo de Desenvolvimento de Produtos: uma abordagem baseada na criação de valor». Em: *Editora Atlas* (jan. de 2007). (Acedido em 17/02/2021).
[37] *Peter Koen*. url: http://frontendinnovation.com/fei/what-is-the-new-concept-development-ncd-model (acedido em 17/02/2021).
[38] Pierry Teza et al. «Direcionadores do processo de inovação: o papel da estratégia, liderança e cultura». Em: *Navus - Revista de Gestão e Tecnologia* (2013), pp. 77–88. doi: 10.22279/navus.2013.v3n2.p77-88.156. (Acedido em 17/02/2021).
[39] P. Koen et al. «Fuzzy Front End: Effective Methods, Tools, and Techniques». Em: (2002), p. 15. (Acedido em 17/02/2021).

[40] P. Koen et al. «Fuzzy Front End: Effective Methods, Tools, and Techniques». Em: (2002), pp. 17–18. (Acedido em 17/02/2021).

[41] P. Koen et al. «Fuzzy Front End: Effective Methods, Tools, and Techniques». Em: (2002), pp. 19–20. (Acedido em 17/02/2021).

[42] Cristiano Souza Martins, Daniela de Oliveira Souza e Magno da Silva Barros. «O Uso do Método de Análise Hierárquica (AHP) na Tomada de Decisões Gerenciais: Um Estudo de Caso». Em: *Simpósio Brasileiro de Pesquisa Operacional, 41* (2009), pp. 1778–1788. url: http://www.din.uem.br/~ademir/sbpo/sbpo2009/artigos/55993.pdf (acedido em 20/02/2021).

[43] Zhiyong Zhang, Xinbao Liu e Shanlin Yang. «A Note on the 1-9 Scale and Index Scale In AHP». Em: *Communications in Computer and Information Science Cutting-Edge Research Topics on Multiple Criteria Decision Making* (2009), pp. 630–634. (Acedido em 20/02/2021).

[44] Susana Nicola, Eduarda Pinto Ferreira e J.J. Pinto Ferreira. «International Journal of Information Technology Decision Making». Em: (2012), pp. 661–703. (Acedido em 28/02/2021).

[45] Carol M. Kopp. *Understanding Perceived Value*. Dez. de 2020. url: https://www.investopedia.com/terms/p/perceived-value.asp (acedido em 02/03/2021).

[46] Jaka Lindič e Carlos Marques Da Silva. «Value proposition as a catalyst for a customer focused innovation». Em: *Management Decision* 49.10 (2011), pp. 1694–1708. doi: 10.1108/00251741111183834. (Acedido em 02/03/2021).

[47] Cristian Picoli, Cristian Picoli e Cristian Picoli. *Arquitetura de software: Saiba o que é e qual a sua importância*. Mar. de 2021. url: https://atmdigital.com.br/saiba-o-que-e-arquitetura-de-software-e-qual-a-sua-importancia/ (acedido em 03/03/2021).

[48] Renato Gonalves de Oliveira. *Desenvolvimento baseado em componentes - Revista Java Magazine 110*. url: https://www.devmedia.com.br/desenvolvimento-baseado-em-componentes-revista-java-magazine-110/26550 (acedido em 04/03/2021).

[49] John Cheesman e John Daniels. *UML components: a simple process for specifying component-based software*. Addison-Wesley, 2000.

[50] Itana Maria de Souza Gimenes e Elisa Hatsue Moriya Huzita. *Desenvolvimento baseado em componentes: conceitos e técnicas*. Ciência Moderna, 2005.

[51] *Arquitetura baseada em Componentes*. Out. de 2010. url: https://marcobaccaro.wordpress.com/2010/10/05/arquitetura-baseada-em-componentes/ (acedido em 04/03/2021).

[52] *Hypertext Preprocessor*. url: https://www.php.net/.

[53] *MariaDB*. url: https://mariadb.org/about/.

[54] *MySQL*. url: https://www.mysql.com/.

[55] *AngularJS*. url: https://angularjs.org/.

[56] *Angular*. url: https://angular.io/.

[57] url: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deplo%20yment-diagram/.

[58] *GitLab*. url: https://about.gitlab.com/.

[59] *Jenkins*. url: https://www.jenkins.io/doc/.

[60] *Zend Framework*. url: https://framework.zend.com/about.

[61] Michael Henrique Pereira. *AngularJS: Uma abordagem prática e objetiva*. Novatec Editora, 2014.

[62] *SonarQube*. url: https://docs.sonarqube.org/latest/.

[63] *Postman*. url: `https://www.postman.com/`.
[64] Tomoko Nemoto e David Beglar. «Developing Likert-Scale Questionnaires». Em: (2014).

Apêndice A

Método AHP

A.1 Escala Fundamental de Saaty

Nivel de importância	Definição	Explicação
1	Igual importânica	As duas atividades contribuem igualmente para o objetivo
3	Fraca importância	A experiência e o julgamento favorecem levemente uma atividade em relação à outra
5	Forte importância	A experiência e o julgamento favorecem fortemente uma atividade em relação à outra
7	Muito forte importância	Uma atividade é muito fortemente favorecida em relação a outra
9	Importância absoluta	A evidência favorece uma atividade em relação a outra com o mais alto grau de certeza
2,4,6,8	Valores intermediários	Quando se procura uma condição de compromisso entre duas definições

Figura A.1: Escala Fundamental de Saaty

A.2 Índice aleatório (IR)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

Figura A.2: Valores de IR para matrizes quadradas de ordem n

Apêndice B

Design de Mock-Ups

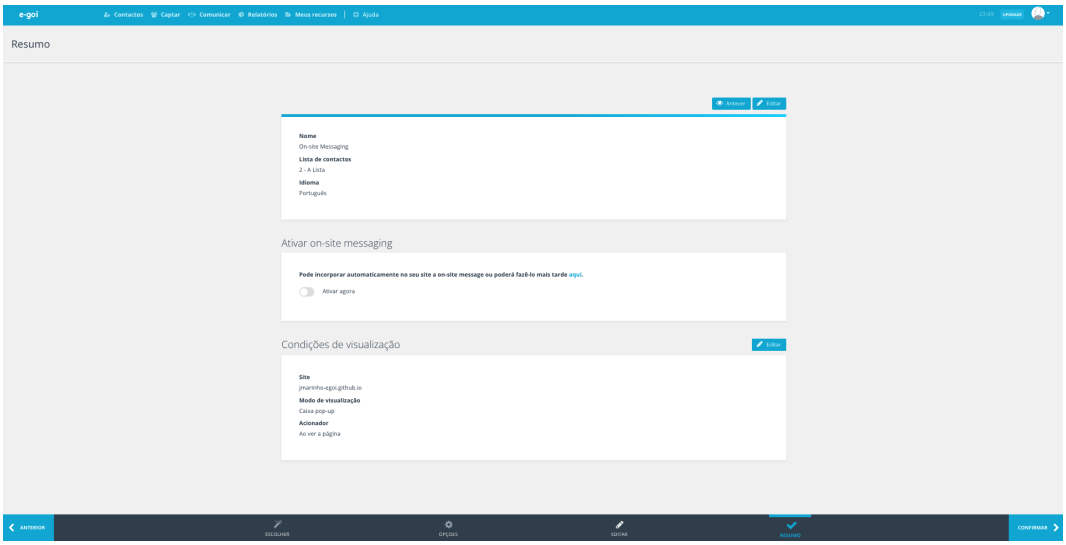

Figura B.1: Mock-up da página template

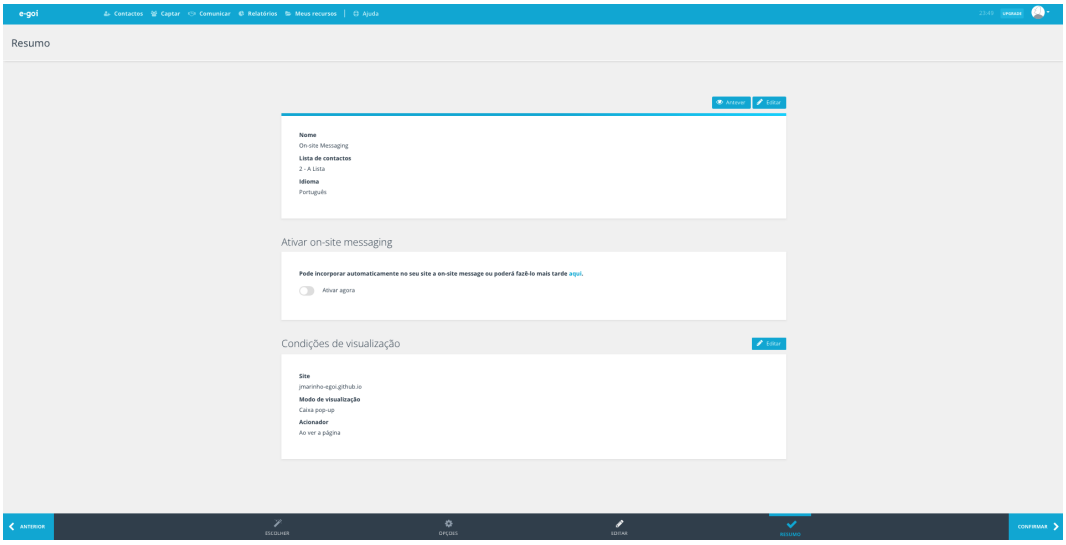

Figura B.2: Mock-up da página de resumo

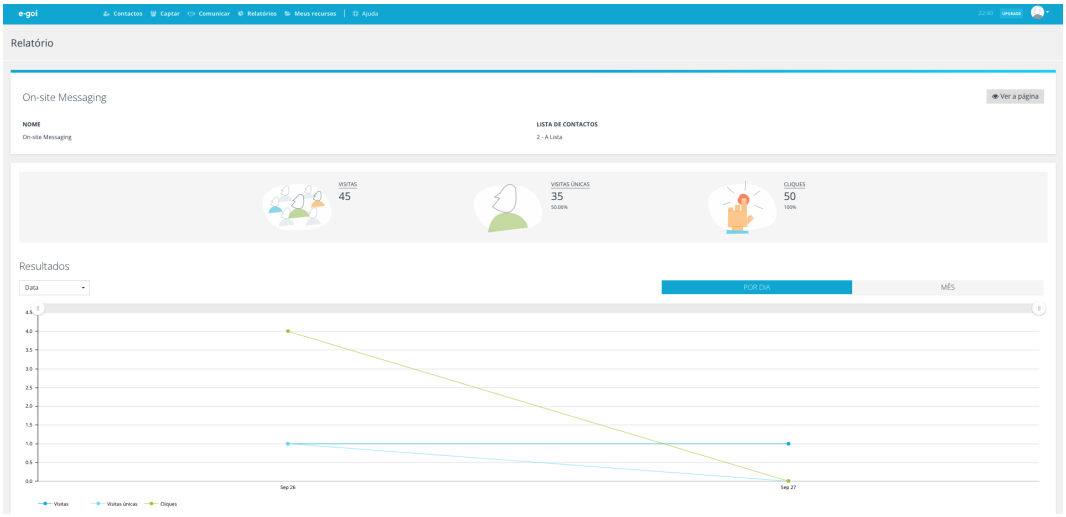

Figura B.3: Mock-up da página de resultados

Apêndice C

Inqérito de Satisfação e Usabilidade

Inquérito de satisfação e usabilidade

Este inquérito foi desenvolvido no âmbito da Unidade Curricular TMDEI no decorrer da Tese de Mestrado do aluno José Marinho e tem por objetivo obter o feedback do nível de satisfação e usabilidade do novo canal de comunicação on-site messaging da E-goi.
Responda a todas as questões de 1 a 4, sendo que 1 significa "Discordo totalmente" e 4 "Concordo totalmente".
Todas as respostas serão anónimas e usadas apenas para fins académicos.
Obrigado pela disponibilidade.

1. Facilidade em criar uma on-site message e vê-la listada.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

2. Facilidade em ativar uma on-site message no site.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

3. Facilidade em editar as opções de uma on-site message.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

4. Facilidade em editar a aparência de uma on-site message através do editor gráfico.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

5. Facilidade em antever uma on-site message listada.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

6. Facilidade em apagar um on-site message listada.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

7. Facilidade em ver a on-site messag no site segundo o trigger que definiu.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

8. Facilidade em personalizar mensagens de texto através dos códigos de personalização.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

9. Facilidade em utilizar o editor gráfico para diferentes tipos de on-site messages.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

10. Facilidade em visualizar a on-site message em qualquer tipo de dispositivo, não apresentando problemas de responsividade.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

11. Facilidade em reconhecer alterações de estado do sistema.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

12. O tempo de espera entre a passagem de uma página do fluxo para outra é aceitável.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

13. Considera o sistema simples e fácil de utilizar.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

14. Está satisfeito com o novo canal de on-site messaging.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯

15. Recomendaria o novo canal de on-site messaging a outras pessoas.

Marcar apenas uma oval.

1	2	3	4
◯	◯	◯	◯