INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Editor de Formulários do E-goi

GONÇALO REAIS SANTOS DELFINO
Julho de 2019

POLITÉCNICO
DO PORTO

Editor de Formulários do E-goi

Gonçalo Reais Santos Delfino

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

Orientador: Nuno Bettencourt
Tutor: Ivo Pereira

Porto, 5 de Julho de 2019

Dedicatória

Dedicado a todos os que, das mais diversas maneiras, me ajudaram a completar esta etapa. A vossa ajuda foi fundamental.

Resumo

Para implementar uma funcionalidade num produto já existente, é necessário perceber se a funcionalidade é realmente necessária, definir o planeamento para a implementar, desenvolver a funcionalidade e, por fim, avaliar a solução de modo a perceber se cumpre os objetivos.

Todo este processo é descrito neste documento relativamente à funcionalidade de criação e gestão de formulários do produto E-goi. O E-goi consiste num produto de *marketing* digital multicanal que permite a importação de listas de contactos, envios automatizados de campanhas e formulários, entre outras funcionalidades relativas a *marketing* digital.

Está presente neste documento uma introdução ao projeto, identificando problemas do editor de formulários da E-goi e objetivos para o novo editor. É feita uma contextualização ao mundo do *marketing* digital e aos conceitos relativos ao projeto. São apresentadas e analisadas diversas soluções de criação e gestão de formulários assim como as tecnologias utilizadas no decorrer do projeto. Posteriormente é elaborada uma análise de valor que determina a validade e a necessidade da existência do projeto recorrendo a diversas técnicas diferentes. É feito o design de uma solução que alcança os objetivos propostos cumprindo os problemas, utilizando linguagens e tecnologias atuais assim como boas práticas de engenharia informática e de desenvolvimento de *software*. São também demonstrados os pormenores técnicos do desenvolvimento do novo editor de formulários do produto E-goi, apresentando a solução para os desafios da solução desenhada. De modo a validar a qualidade da solução desenvolvida, são feitos testes de usabilidade e de *software*, recorrendo a técnicas precisas e fundamentadas. Por último são tiradas conclusões relativas ao desenvolvimento de todo o projeto.

Palavras-chave: Formulários, Usabilidade, *Marketing* Digital

Abstract

In order to add a functionality to an existing product, one needs to understand if the functionality is really needed, set the planning to implement it, develop said functionality and ultimately evaluate the final result to understand if it meets the objectives and solves the problems.

This entire process is described in this document regarding the forms funcionality (creation and management of forms) of the E-goi product. E-goi is a multichannel digital marketing product that allows you to import contact lists, create and send automated campaigns and forms, among other functionality related to digital marketing.

This document includes an introduction to the project, identifying issues with the E-goi forms editor and goals for the new editor. The world of digital marketing and forms related concepts are presented to allow the reader to understand the whole project. Various solutions which allow forms creation and management are presented and analyzed as well as the technologies used during the project. Subsequently, a value analysis is elaborated determining need for the project using several different techniques. A solution that achieves the proposed objectives is designed, solving the identified problems with the actual editor. The details of implementation of the new forms editor of the E-goi product are exposed, showing the usage of good computer engineering and software development practices. In order to validate the quality of the developed solution, usability and software tests are implemented, using precise and well-founded techniques. Finally conclusions are drawn regarding the development of the whole project.

Agradecimentos

Em primeiro lugar, agradeço ao professor Nuno Bettencourt todos os conselhos que me deu e que permitiram que este documento existisse.

Seguidamente, agradeço à E-goi, representada pelo CEO Miguel Gonçalves, a oportunidade e condições para fazer o projeto.

Agradeço também a todos os colaboradores da E-goi pela ótima integração e ajuda ao longo de todo o projeto, em especial ao Ivo Pereira, Vitor Tavares, Tiago Temporin e a todos os Santos da empresa.

Agradeço a todos os amigos que a vida me trouxe, sem vocês tinha sido muito mais complicado.

Por último, o agradecimento mais especial, à minha família que permitiu que eu chegasse a este nível e sempre me apoiou em tudo.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Código

Lista de Acrónimos

AHP	Analytic Hierarchy Process.
API	Application Program Interface.
CSS	Cascading Style Sheets.
CTO	Chief Techonology Officer.
DDD	Domain Driven Design.
HTML	HyperText Markup Language.
HTML5	HyperText Markup Language 5.
HTTP	Hypertext Transfer Protocol.
HTTPS	Hyper Text Transfer Protocol Secure.
IP	Internet Protocol.
PDF	Portable Document Format.
REST	Representational State Transfer.
SMS	Short Message Service.
UML	Unified Modeling Language.

Capítulo 1

Introdução

Num mercado cada vez mais digital, onde a maioria das transações acontecem sem interação humana, a comunicação com o cliente é cada vez mais valorizada pelas empresas e organizações. Os departamentos de *marketing* e comunicação das empresas procuram maneiras otimizadas de manter o contacto com os seus clientes recorrendo a inúmeras abordagens como conversas de *chat*, contacto através de *e-mail*, redes sociais, utilização de formulários, entre outras [4]. Esta comunicação necessita de ser bem estruturada e personalizada para que seja eficaz mas não demasiado invasiva. Parte desta estruturação está relacionada com o aspeto e conteúdo das mensagens. Este projeto, apresenta a resposta do produto E-goi, relativamente a formulários, para esta necessidade emergente de comunicação.

1.1 Contextualização

Apesar de, por vezes, passarem despercebidos, os formulários estão presentes em diversos aspetos da nossa vida, quer de uma forma mais tradicional, quer num ambiente mais digital. Exemplos disso são a inscrição num clube desportivo, um boletim de voto ou uma inscrição numa rede social. Em cada um dos exemplos, assim como em qualquer formulário, o objetivo é a obtenção de informação sendo que, no ambiente empresarial, as equipas de *marketing* e comunicação utilizam esta técnica para diversos fins, tais como a angariação de clientes ou envio de *feedback* sobre um produto ou serviço [5].

Usualmente, os formulários são apresentados dentro de *landing pages*. Uma *landing page* consiste numa página, normalmente temática, que tem como objetivo levar o visitante a realizar uma ação como uma compra ou uma inscrição. Ambos os conceitos podem existir em separado ou em conjunto, ou seja, um formulário pode existir por si só ou ser integrado numa *landing page* assim como uma *landing page* pode conter um formulário ou não.

É importante definir e diferenciar os conceitos visitante, utilizador e cliente no âmbito deste projeto. Um visitante consiste numa pessoa que visualiza e preenche um formulário. Um utilizador consiste numa pessoa que utiliza o editor de formulários e o produto E-goi. Um cliente consiste numa pessoa que mantém contacto com uma instituição de um utilizador E-goi cujas informações podem já constar de uma lista de contactos ou não sendo que um cliente pode ser um visitante ou não.

Existem diversos produtos de *marketing* digital que, entre outras funcionalidades, disponibilizam a criação e gestão de formulários e *landing pages* que auxiliam os profissionais de *marketing* a alcançar os seus objetivos.

O E-goi [6] é um exemplo de um desses produtos. Sendo uma plataforma de automação de marketing digital multicanal, o E-goi permite a importação de listas de contactos, a criação e gestão de campanhas e formulários, assim como o seu envio através de diversos canais (como *e-mail*, Short Message Service (SMS), chamada de voz, notificações web, entre outros). O produto permite ainda automatizar processos consoante comportamento dos clientes, por exemplo, enviar uma SMS com uma promoção quando um cliente se inscreve através de uma campanha.

Este projeto encontra-se enquadrado no âmbito da tese de mestrado de Engenharia Informática (ramo de Sistemas Computacionais) do Instituto Superior de Engenharia do Porto em parceria com a empresa E-goi.

1.2 Apresentação da Empresa

A E-goi é uma empresa de *marketing* digital fundada por Miguel Gonçalves, em 2008 e sediada em Matosinhos, Portugal. A empresa conta com aproximadamente 100 colaboradores que trabalham diariamente para melhorar e manter um produto de *marketing* digital multicanal, o E-goi. Em Fevereiro de 2019, o produto conta com mais de quatrocentas mil contas criadas sendo já uma referência no mundo do *marketing* especialmente nos mercados português, brasileiro, espanhol, colombiano e da américa latina. A missão da empresa consiste na aproximação dos clientes e as marcas da forma mais fácil e direta possível, disponibilizando ferramentas para este fim.

O produto E-goi apresenta diversas funcionalidades desde importação, criação e gestão de listas de contactos, criação e envio de campanhas e a sua automatização, realização de ações automáticas consoante comportamento dos contactos (autobots), criação e gestão de formulários e *landing pages*, entre outras. Além de tudo isto, o E-goi conta com bastante integrações, tais como Facebook, Shopify, Wordpress, Magento, Zapier, Instapage, Unbounce, entre outras. Todas estas funcionalidades são oferecidas através da plataforma de uma forma *freemium*, ou seja, a plataforma permite uma utilização gratuita até certos limites (tais como número de contactos permitidos ou envios), sendo possível desbloquear todas as funcionalidades e limitações mediante um pagamento adicional.

1.3 Motivação

A existência desta dissertação é justificada com a presença de vários problemas relativos ao editor de formulários do E-goi, o editor EasyForms. Os problemas identificados foram reportados pelas equipas da empresa. Algumas das equipas da E-goi, nomeadamente a equipa de serviço de apoio ao cliente, departamento de *marketing* de pequenas e médias empresas e departamento de *marketing corporate* trabalham diariamente com a ferramenta tendo elementos que personificam os clientes E-goi. A natureza dos problemas identificados é funcional assim como não funcional e apresentam consequências negativas ao negócio.

O primeiro problema é de natureza não funcional e está relacionado com a forma como o editor se encontra implementado. O produto E-goi conta com mais de dez anos de existência e tem sofrido inúmeras alterações ao longo desses anos, quer a nível arquitetural como a nível de tecnologias. Desde 2016 que existe um esforço da empresa para migrar o produto para linguagens mais recentes, que necessitem de menor manutenção e que permitam uma

maior escalabilidade do sistema. Este problema traz algumas consequências à empresa tais como um aumento do custo de manutenção do editor, sendo necessário mais tempo para fazer alterações necessárias. Para além desse custo, a escalabilidade do sistema fica comprometida, devido à falta de novos padrões implementados por linguagens mais atuais e o acoplamento de diversas funcionalidades em monólitos complexos.

O segundo problema é de natureza não funcional e prende-se com a existência de um acoplamento entre as funcionalidades de formulários e *landing pages* no editor EasyForms, não sendo possível criar um formulário sem criar uma *landing page* e o inverso. A falta de clareza entre os dois conceitos gera confusão nos utilizadores, dificultando a gestão interna assim como a geração de estatísticas e métricas adequadas.

O terceiro problema é de natureza funcional e consiste na falta de funcionalidades, tanto dentro do editor como na análise de resultados. O editor não apresenta algumas funcionalidades que os utilizadores consideram importantes tais como a possibilidade de paginação, a possibilidade de criar formulários dinâmicos através de condições (por exemplo, ocultar campos do formulário dependendo de respostas dadas em campos anteriores) e a falta de alguns *widgets*.

O quarto problema é de natureza funcional e está relacionado com a análise de resultados do preenchimento dos formulários. Por ser uma funcionalidade demasiado permissiva, na medida em que permite liberdade quase total na construção dos formulários, é complicado gerar métricas úteis quer para os utilizadores finais quer para a empresa. Alguns exemplos destas métricas são, ao nível do utilizador final, o tempo de preenchimento do formulário de cada visitante e o tempo médio de preenchimento do formulário. Ao nível empresarial, algumas métricas em falta consistem no número de vezes que um *widget* é utilizado assim como no número de vezes que um *template* é utilizado. Estes fatores fazem com que a funcionalidade de criação e gestão dos formulários seja pouco usada pelos utilizadores E-goi assim como complicam a atualização quer do editor em si quer dos *templates* oferecidos, por falta de informação.

Os problemas funcionais refletem-se no elevado número de *tickets* existentes relativos ao editor de formulários do E-goi. Um *ticket* consiste numa dúvida ou reclamação enviada por escrito por um cliente para o serviço de apoio ao cliente de uma empresa ou organização. No decorrer do ano de 2018, a E-goi recebeu cerca de 3000 *tickets* relativos ao editor de formulários num universo de cerca de 40000 *tickets*, representando assim 7,5% da globalidade de *tickets* anuais. Adicionalmente ainda foram abertas 4700 conversas de *chat* também relativas à funcionalidade dos formulários representando 9% do total de conversas abertas em 2018. Ambas as abordagens (*tickets* e conversas de *chat*) constituem cerca de 8,3 % do total de pedidos de suporte em 2018. Estes números traduzem um grande esforço por parte da empresa para esclarecer e educar os seus utilizadores, algo que devia ser automático. Depois de uma análise aos *tickets* ficaram percetíveis alguns problemas evidentes que afetam os utilizadores E-goi e que, inclusivamente, fazem com que estes abandonem a plataforma e optem por criar os seus formulários noutras soluções concorrentes. Estes problemas estão relacionados com a usabilidade do editor em si assim como na dificuldade em encontrar opções e funcionalidades relativas aos de formulários.

Em suma, é possível listar os seguintes problemas relativos à funcionalidade de formulários da plataforma E-goi:

Complexidade para os utilizadores;

Arquitetura e tecnologias desatualizadas;

Acoplamento entre funcionalidade de formulários e *landing pages*;

Falta e inconsistência de funcionalidades;

Falta de métricas para análise de resultados.

A identificação de problemas é uma vertente de elevada importância num projeto e deve ser desenvolvida com o maior rigor possível. No entanto, é importante referir que, por ser feita numa fase prematura do projeto, esta identificação não é estanque e pode ser alterada ao longo do mesmo com a adição de novos problemas identificados.

1.4 Objetivos

O objetivo do projeto é implementar uma solução, integrada na plataforma E-goi, que permita a criação e gestão de formulários e solucione os problemas identificados. Para conseguir atingir este objetivo, é necessário subdividi-lo em sub-objetivos específicos e mais facilmente exequíveis.

Esta subdivisão permite definir abordagens para cada objetivo ao invés de aplicar uma solução complexa para o problema como um todo. Esta abordagem pode ser equiparada à popular expressão *"divide and conquer"*, dividindo assim o problema complexo em objetivos de dimensão mais reduzida, cuja solução é específica e mais facilmente atingível. Com este propósito, foram definidas tarefas para cada objetivo:

Identificar problemas no editor EasyForms:
: Para completar este objetivo é necessário estudar o editor EasyForms, implementar testes de usabilidade com utilizadores e perceber os problemas existentes na ótica do utilizador final;

Identificar soluções para os problemas identificados:
: Depois de identificados os problemas, é necessário perceber como os colmatar. Uma das formas possíveis é estudar o mercado, identificar soluções que resolvam os problemas identificados e perceber se essas soluções são aplicáveis no projeto;

Definir uma arquitetura que satisfaça os problemas identificados:
: Depois de perceber que medidas tomar para resolver os problemas identificados, é necessário desenvolver uma solução, validar a mesma com as partes interessadas e definir moldes para a implementação;

Implementar a arquitetura definida:
: Depois de todo o planeamento estar definido, é necessário implementar a solução final seguindo boas práticas de desenvolvimento de *software*, garantindo robustez, escalabilidade e baixa manutenção;

Integrar o sistema na plataforma E-goi:
: Adotar uma abordagem de integração continua do sistema na plataforma existente;

Testar a solução e avaliar o seu desempenho:
: Depois de implementado e integrado, é necessário testar o sistema para avaliar se efetivamente soluciona os problemas identificados, percebendo as suas limitações.

É esperado que o objetivo final seja atingido na sua totalidade seguindo a abordagem apresentada.

1.5 Metodologia

Neste documento é documentado o projeto para a criação do novo editor de formulários do produto E-goi. A metodologia usada neste projeto é baseada no artigo *Design Science Research Methodology in Computer Science and Information Systems* [7]. Este artigo apresenta diversas *guidelines* para o desenvolvimento de sistemas de informação e a metodologia utilizada segue essas *guidelines*.

Inicialmente é feita a contextualização e motivação do projeto assim como a definição dos objetivos do mesmo, seguindo assim a *guideline* 2, *"The Relevance of Problem"*.

Depois de definido o âmbito e a relevância do projeto, foi feita uma contextualização de conceitos relativos ao projeto necessários para a compreensão do mesmo.

Seguiu-se uma análise do estado da arte comparando soluções existentes relacionadas com o problema definido assim como o estudo de tecnologias relativas ao projeto, segundo a *guideline* 6, *"Design as a Search Process"*.

Após estudar as soluções e tecnologias existentes, foi feita uma análise de valor ao projeto que permitiu perceber a necessidade do mesmo, comprovando com mais exatidão a relevância do problema e seguindo a *guideline* 5, *"Research Rigo"*. Esta *guideline* pressupõe a utilização de métodos rigorosos para o desenvolvimento do projeto, tais como a utilização de métodos de decisão como Analytic Hierarchy Process [8] ou análise SWOT. Para que a solução a desenvolver fosse a mais adequada possível, foram desenvolvidos testes de usabilidade ao editor EasyForms, obtendo assim resultados de usabilidade representativos dos utilizadores reais do sistema, seguindo assim a *guideline* 6, *Design as a Search Process*.

Seguidamente foi definido o *design* da solução com base no estudo realizado e que cumprisse todos os objetivos definidos, seguindo a *guideline* 1, *"Design as an Artifact*, desenhando uma solução baseada em artefactos como a representação de diagramas ou técnicas de desenvolvimento.

Seguiu-se a implementação e integração da solução, cumprindo com o design definido assim como boas práticas de engenharia informática e desenvolvimento de *software*.

Depois de implementado, o novo sistema foi testado e validado de modo a perceber se cumpre com todos os objetivos definidos, de acordo com a *guideline* 3, *"The Design Evaluation"*. Esta última *guideline* pressupõe o uso de testes rigorosos como análise estatística e testes de usabilidade.

1.6 Estrutura do Documento

Este documento encontra-se dividido em Capítulos, cada um referente a uma parte específica do desenvolvimento do projeto constituídos por diversas Secções e Sub-Secções. Os capítulos presentes são os seguintes:

Introdução:
Neste capítulo é contextualizado o projeto, a empresa onde o projeto foi desenvolvido assim como apresentada a motivação e os objetidos do projeto. Por fim é apresentada a metodologia adotada para o desenvolvimento do projeto;

Contextualização:
Neste capítulo é apresentada uma contextualização de temas relativos ao projeto, necessários para compreender todo o documento. É também apresentado o produto E-goi de uma forma detalhada;

Estado da Arte:
Neste capítulo são apresentadas diversas soluções existentes para a criação de formulários seguida de uma análise comparativa. Também são apresentados todos os métodos, técnicas e tecnologias utilizadas ao longo do projeto;

Análise:
Neste capítulo é apresentado o levantamento de requisitos do sistema através da análise de um teste de usabilidade. É também feita uma análise de valor que confirma a necessidade do projeto assim como os moldes como este será desenvolvido;

Implementação:
Neste capítulo são apresentados todos os pormenores técnicos relevantes assim como a metodologia de desenvolvimento aplicada à implementação do projeto;

Experimentação e Avaliação:
Neste capítulo são apresentados os testes implementados à solução desenvolvida de modo a perceber a validade da solução;

Conclusões:
Neste capítulo são apresentadas as conclusões relativas ao projeto desenvolvido assim como o trabalho futuro a desenvolver para a continuação do projeto.

Capítulo 2

Contextualização

Para compreender um projeto na sua totalidade, é importante perceber os conceitos associados ao mesmo pois só assim é possível fundamentar decisões a tomar. O projeto descrito nesta dissertação contém alguns conceitos específicos que necessitam de ser explicitados. Neste capítulo são apresentados conceitos ligados ao *marketing* digital como listas de contactos, formulários e *landing pages* assim como uma visão detalhada do produto E-goi.

2.1 Listas de Contactos

Uma lista de contactos, no âmbito do projeto, numa listagem de informações sobre contactos (nome, apelido, número de telemóvel, *e-mail*, etc) que tem como objetivo permitir o envio automático de mensagens para toda a lista ou para segmentos da mesma.

É através das listas que as equipas de *marketing* conseguem, de uma forma automatizada, manter contacto com os seus clientes, o que torna as listas de contactos um bem bastante valioso para as empresas e organizações. Os produtos de *marketing* digital como o E-goi costumam permitir a importação de listas de contactos já existentes assim como a introdução de contactos individuais.

O número de contactos numa lista pode ascender aos milhares de contactos ou mesmo milhões e, como tal, é necessário existirem identificadores únicos para cada contacto. Podem ser definidos inúmeros campos para uma lista com informação que um contacto deve ter sendo que, tem de existir, pelo menos, um campo de valor único de modo a ser possível identificar o contacto de uma forma inequívoca. Os campos de uma lista consistem em informação que tem de estar presente em cada contacto, por exemplo, nome, apelido, plano de cliente, entre outros. No produto E-goi, os campos de uma lista podem ter os seguintes tipos: texto, número, telefone, telemóvel, *e-mail*, data e lista de opções.

Espera-se que uma lista de contactos não seja estática e tenha um crescimento para que as mensagens a enviar tenham o maior alcance possível por parte das entidades que as enviam. Para atingir esse objetivo, é usual a utilização de formulários. Através do preenchimento do formulário (de inscrição), é possível obter a informação necessária para criar um novo contacto na lista. Se o formulário for preenchido com informação já existente na lista de contactos, pode ser permitida a edição do contacto existente ou pode não ser aceite a inscrição.

2.2 Formulários

Os formulários são uma forma muito eficaz de obter informação e são bastante utilizados pelas equipas de *marketing* de empresas e organizações para angariação de contactos e informação relacionada.

2.2.1 Tipos de Formulários

Cada tipo de formulário tem objetivos diferentes e, por isso, deve ter opções diferentes. Na plataforma E-goi, onde os formulários têm uma ligação direta com uma lista de contactos, estes têm como objetivo adicionar, remover e editar contactos à lista de contactos. Paralelamente, os formulários E-goi podem ser utilizados para obter informação não relacionada com a lista e até mesmo permitir que os utilizadores consigam partilhar campanhas com outras pessoas. Através destes objetivos é possível categorizar os diversos tipos de formulários.

Formulário de Inscrição

Um formulário de inscrição tem como objetivo inscrever um visitante numa lista de contactos. Este tipo de formulário costuma requerer informações pessoais do visitante tais como nome, *e-mail*, contacto telefónico assim como um campo de valor único, que diferencie o contacto dos outros (usualmente esta diferenciação é feita pelo *e-mail*). Um formulário de inscrição dá também a possibilidade de confirmar a inscrição através do email (*double opt-in* [9]), sendo essa opção obrigatória em alguns países devido a leis de proteção de dados. Desta forma, existe um entrave que dificulta a inscrição de um contacto feita por terceiros, uma vez que é necessária a confirmação da inscrição através do *e-mail*.

Formulário de Edição

O formulário de edição tem como objetivo editar dados de um utilizador presente numa lista de contactos. Este formulário é bastante semelhante ao formulário de inscrição na medida em que pode requerer a mesma informação. Também é usual existir um *e-mail* de confirmação para que o utilizador apenas consiga editar os seus dados e não os de outra pessoa. Tratando-se de um formulário de edição, este costuma ser pré-preenchido com os dados atuais do contacto automaticamente. Este tipo de formulários é bastante utilizado quando é necessário acrescentar ou editar dados dos contactos da lista.

Formulário de Recomendação

O formulário de recomendação permite a um utilizador recomendar uma campanha a outra pessoa. Este tipo de formulários é bastante utilizado para angariar contactos através de contactos existentes, sendo usual oferecer uma recompensa aos contactos que indiquem novos contactos.

Formulário de Remoção

O formulário de remoção permite um utilizador remover o seu contacto da lista. Este formulário pode gerar o envio de um *e-mail* informativo para o utilizador removido e tem por norma a inclusão de um campo onde o utilizador pode apresentar o motivo da sua remoção, sendo usualmente um campo de introdução de texto ou uma lista de opções.

Questionário

O questionário é um tipo de formulário que, por omissão, não se encontra associado a uma lista. Este tipo de formulário tem como objetivo a obtenção direta de dados e não de contactos, sendo tipicamente usados para receber *feedback* de um produto ou serviço.

2.2.2 Edição de Formulários

A criação e construção de um formulário é determinante para a obtenção de um número elevado de respostas uma vez que um formulário muito extenso, com uma vertente gráfica fraca ou com conteúdo impróprio, pode intimidar um visitante e desencorajar a sua resposta.

Por este motivo, muitos dos editores para a criação de formulários tendem a recorrer a sistemas *drag & drop* simples que permitem criar formulários de uma maneira fácil e intuitiva. Estes sistemas incluem o conceito de *widget*, um elemento que pode ser arrastado para dentro de uma tela de modo a estruturar o formulário.

Existem diversos tipos de *widgets*, cada um com formas e finalidades diferentes tais como introdução de texto, seleção de opções, botões, imagens, entre outros. A quantidade e variedade de *widgets* que um editor disponibiliza é um aspeto importante a ser considerado, pois a inexistência de alguns *widgets* podem desencorajar o utilizador a usar o sistema. Cada *widget* costuma também ter opções específicas como cor do texto, tamanho, posição entre outras.

Quando não é utilizado um sistema *drag & drop*, os editores de criação de formulários tendem a apresentar uma tela onde é permitido adicionar campos ao formulário através de cliques, sem arrastar. Estas soluções tendem a apresentar menos opções gráficas ao utilizador e são menos utilizadas.

2.2.3 Utilização de Formulários

Depois de completar a criação do formulário, é necessário escolher o método para apresentar o mesmo para que este possa ser preenchido por clientes e potenciais clientes. Existem várias formas para utilizar um formulário, desde partilhar o mesmo através de um *link*, embeber o mesmo numa página *web* ou partilhar através de terceiros.

Os formulários alojados nas plataformas de criação são acedidos através de um *link*, personalizado ou não, gerado pela plataforma. Para partilhar este formulário, o utilizador apenas tem de partilhar o *link* do mesmo, sendo o visitante reencaminhado para o formulário quando acede ao *link*.

Um formulário cuja utilização consiste na sua integração numa página *web* pode ser embebido em vários formatos como código HyperText Markup Language (HTML) ou código Javascript e com vários formatos tais como *iframes*, janelas *pop-up*, código estático ou mesmo *QR code*. O código a embeber contém toda a estrutura e estilos do formulário assim como a ligação aos servidores para que as respostas sejam guardadas. É ainda possível partilhar um formulário recorrendo a serviços externos tais como Facebook, Twitter ou outras redes sociais. Os produtos de *marketing* digital costumam ter uma integração direta com este tipo de serviços garantindo uma partilha fácil e rápida.

2.2.4 Resultados de Formulários

Existem diversas métricas e estatísticas que podem resultar do preenchimento de formulários tais como o número de submissões, o número de vezes que o formulário foi visualizado, percentagem de visitantes que se inscrevem numa lista através de um formulário, entre outras. Os resultados de um formulário são importantes porque, além de conterem a informação introduzida pelos visitantes, traduzem a adesão e a taxa de conversão dos mesmos. Por esse motivo, é importante que uma ferramenta de criação de formulários seja capaz de apresentar métricas interessantes ao utilizador assim como apresentar as respostas de uma forma simples de ler, já que as submissões de um formulário podem atingir valores na ordem das dezenas de milhares.

2.3 Landing Pages

No seu *e-book* sobre *landing pages*, Oli Gardner [10], co-fundador da empresa Unbounce, define uma landing page como *"a standalone web page distinct from your main website, that has been designed with a single focused objective in mind"*. Através desta definição, podemos perceber que um visitante que seja reencaminhado para uma *landing page* vai encontrar um tema específico relacionado com uma marca mas não relacionado diretamente com o site da marca em questão. As *landing pages* podem ter diversos objetivos sendo os mais comuns a fidelização do cliente, ofertas e promoções ou reencaminhamento para páginas específicas.

Apesar de não ser obrigatório, a maioria das *landing pages* conta com a presença de um formulário de inscrição de modo a aumentar as listas de contactos das marcas. Também é comum existir algum tipo de oferta, como um *e-book* ou uma promoção, consoante a inscrição por parte dos visitantes.

Como é possível perceber através da figura 2.1, espera-se que uma *landing page* seja apelativa e contenha informação que leve o visitante a efetuar uma ação, seja essa ação o preenchimento de um formulário, uma compra ou um *download* de algo, por exemplo.

O propósito de colocar um formulário dentro de uma *landing page* consiste no incentivo à inscrição. O exemplo da figura 2.1 conta com uma frase aliciante e uma seta a apontar para o formulário, dando assim a sensação que o preenchimento do formulário vai resultar na mensagem escrita. Para que estas páginas cumpram o seu objetivo, é necessário que os editores permitam uma forte edição gráfica assim como a inclusão de diversos tipos de estruturas como formulários, cabeçalhos ou rodapés.

Figura 2.1: Exemplo de uma landing page [11]

2.4 E-goi

O produto E-goi consiste numa plataforma que, entre muitas funcionalidades, permite a criação e edição de formulários. A criação e edição de um formulário é feita no editor de formulários *drag & drop* chamado EasyForms. A contextualização relativa ao editor EasyForms é fundamental para perceber de uma forma mais exata os problemas identificados na Secção 1.3. A figura 2.2 apresenta o editor de formulários do produto E-goi.

Através da figura 2.2, é possível perceber que o editor consiste num sistema *drag & drop* onde é permitida a utilização de diversos *widgets*, listados na barra lateral esquerda. Os *widgets* podem ser arrastados para a zona de pré-visualização, permitindo assim ao utilizador montar o formulário. Os *widgets* representam os campos passíveis de serem inseridos no formulário tais como *input* de datas, *e-mail*, texto, números telefónicos, escolha de opções, *upload* de ficheiros assim como título, imagem, vídeo, botão, conteúdo externo e colunas. Estes *widgets* podem posteriormente ser mapeados com campos de uma lista de contactos fazendo com que as respostas do formulário sejam guardadas diretamente nessa lista de contactos. Não existe um controlo sobre este mapeamento dando assim liberdade excessiva ao utilizador e tornando o sistema complexo e confuso para este. É também possível perceber, através da figura 2.2, que para além de apresentar uma lista dos *widgets* existentes, é também possível editar as opções do *widget* selecionado na zona de pré-visualização do formulário, na aba "Item options"assim como as opções globais do formulário, na aba "Global colours and styles".

O E-goi dispõe de diversos tipos de formulários sendo apenas permitida a edição gráfica de formulários de inscrição e questionários, uma vez que os outros tipos de formulários disponíveis são formulários de sistema, iguais para todos os utilizadores. Os formulários de

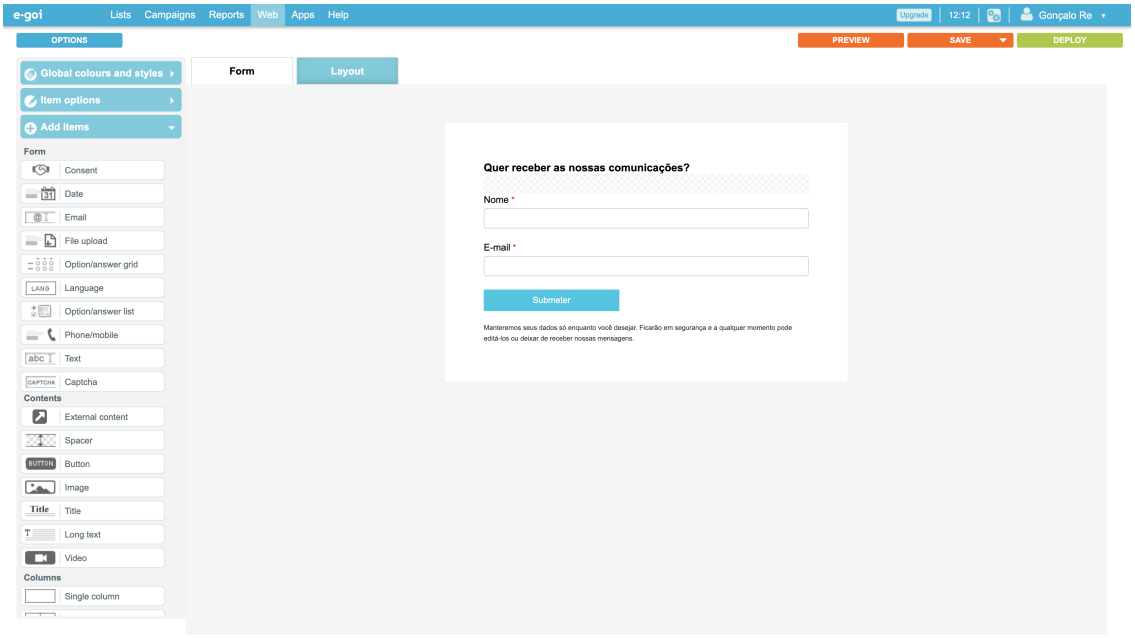

Figura 2.2: Criação de Formulários no editor EasyForms

sistema apresentam uma edição bastante reduzida sendo apenas permitida a alteração de texto exibida ao visitante do formulário.

Atualmente, os conceitos de formulário e *landing page* estão interligados, sendo impossível criar um sem o outro. Para conseguir utilizar um formulário e partilhar o mesmo com os visitantes, é necessário criar uma publicação sendo que, quando o formulário é criado, a publicação do tipo *link* é criada automaticamente. Para criar um formulário, é necessário indicar a lista de contactos a que este está associado podendo depois utilizar os campos dessa lista para montar o formulário, mapeando os *widgets* do editor com os campos da lista. É também possível criar novos campos no próprio editor. Os contactos criados através do formulário serão guardados na lista de contactos indicada.

Os utilizadores conseguem visualizar as respostas dos visitantes aos seus formulários individualmente, o número de submissões face ao número de visitas apresentando gráficos circulares e gráficos de submissões ao longo do tempo.

Por ser uma funcionalidade já existente no produto E-goi, é possível representar as entidades relacionadas com o editor EasyForms assim como as relações entre si. Desta forma, ficam claras as entidades que poderão ser alteradas ao longo do projeto facilitando assim o processo de desenvolvimento de uma nova solução. Para isso foi utilizada a linguagem Unified Modeling Language (UML), sendo escolhido um diagrama de domínio, apresentado na Secção 3.2.1.

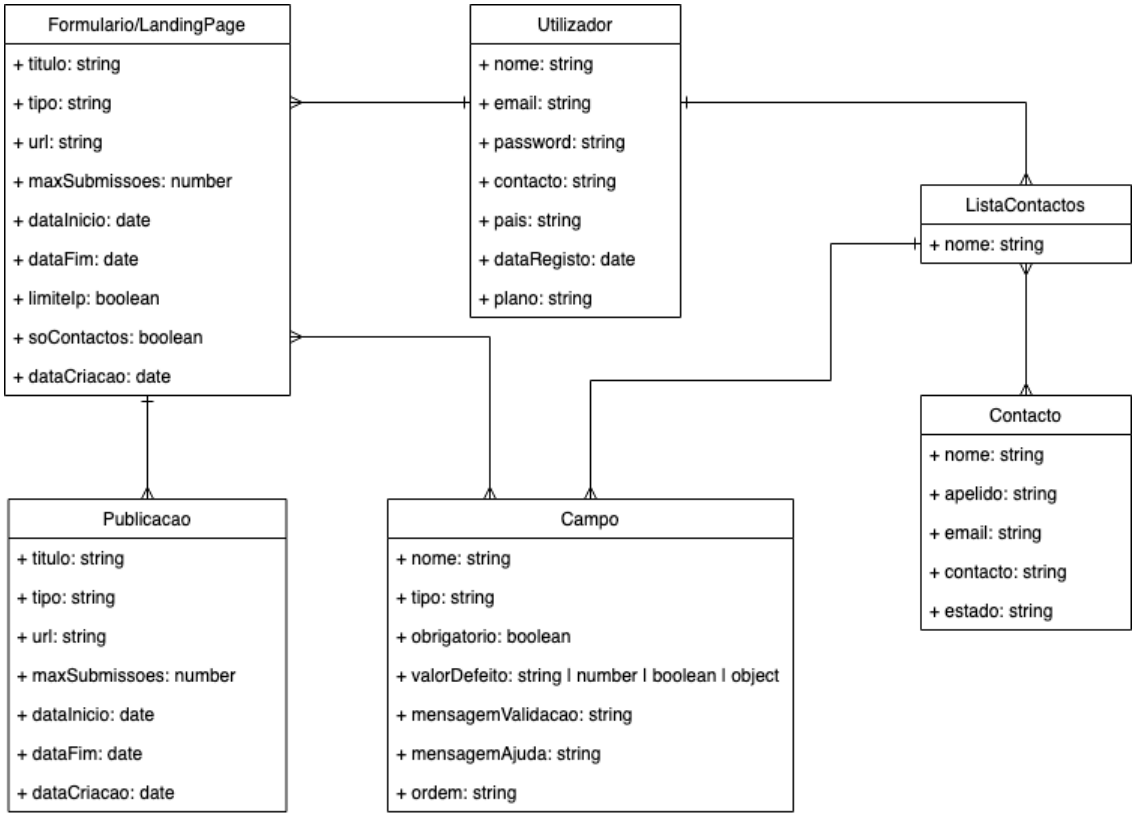

Figura 2.3: Diagrama de Domínio do editor EasyForms

Legenda:

Entidade	**Descrição**
Formulario/LandingPage	Conjunto de campos a ser preenchido pelos visitantes
Publicacao	Forma de utilização do formulário para ser partilhado ou embebido
Utilizador	Conta do produto E-goi que utiliza o sistema
Campo	Local onde é feito o *input* de dados de um formulário e que existe numa lista de contactos
ListaContactos	Lista com diferentes contactos que pode conter vários campos adicionais.
Contacto	Integrante de uma das listas de contactos de uma conta E-goi

Tabela 2.1: Definição das Entidades

Através da análise do diagrama representado na figura 2.3, podemos perceber a ligação entre as diversas entidades intervenientes no sistema. Um formulário/*landing page* é sempre criado por um utilizador e este pode criar diversos formulários/*landing pages*. Cada formulário/*landing page* pode conter diversos campos e ser publicado várias vezes. Um utilizador pode também possuir várias listas de contactos sendo que cada lista pertence a apenas um utilizador. As listas de contacto têm vários contactos e o mesmo contacto pode estar presente em várias listas, do mesmo utilizador ou não.

O diagrama de domínio espelha alguns comportamentos não desejados tais como a unificação do conceito formulário/*landing page* e a existência do conceito publicação. Como referido no secção 1.3 do Capítulo 1, atualmente, os conceitos de formulário e *landing page* estão interligados, sendo impossível criar um sem o outro, apesar de serem conceitos com configurações e finalidades diferentes. Um dos objetivos do projeto, identificados na Secção

1.4, é a separação destes conceitos para que possam ser vistos, criados e editados de forma independente. Esta divisão, uma vez que passará a ser possível criar formulários e *landing pages* de forma separados e com tipos próprios, faz com que o conceito de publicação perca o seu sentido.

O editor EasyForms encontra-se integrado no produto E-goi sendo necessária a compreensão da sua arquitetura. O produto E-goi tem diversas funcionalidades e diversos componentes, no entanto, serão apenas abordados os componentes necessários para o entendimento do projeto. Para este fim, foi utilizado um diagrama UML, o diagrama de componentes, apresentado na Secção 3.2.1.

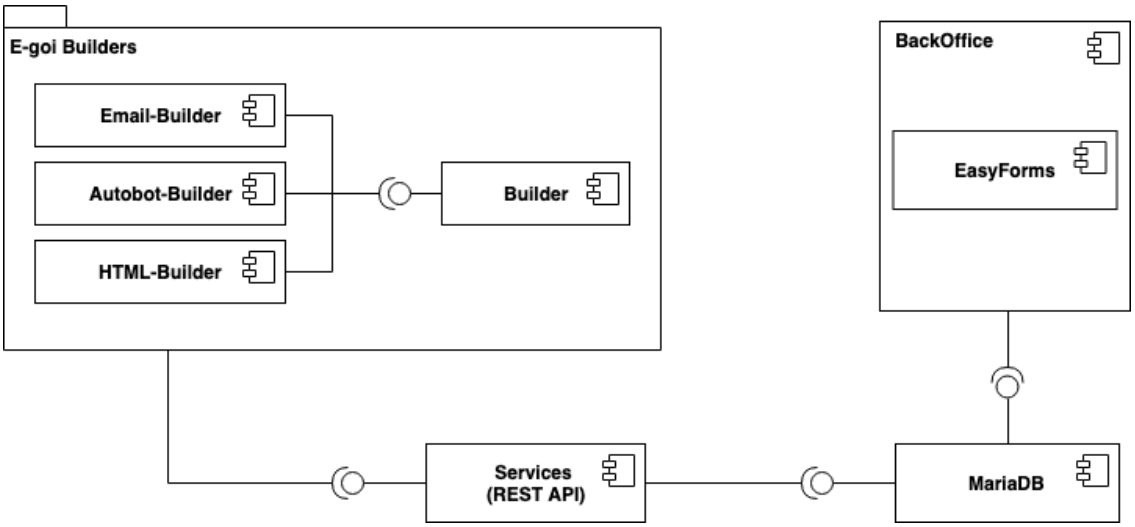

Figura 2.4: Diagrama de Componentes do editor EasyForms

Legenda:

Componente	Descrição
E-goi Builders	Componente que agrega todos os editores do produto E-goi
Email-Builder	Editor de emails do produto E-goi
Autobot-Builder	Editor de *autobots* do produto E-goi
HTML-Builder	Editor de HTML do produto E-goi
Builder	Pacote com lógica de renderização e definição de modelos estendido pelos editores
EasyForms	Editor de formulários do produto E-goi
BackOffice	Componente que contém lógica aplicacional e de interface para o utilizador do produto E-goi
MariaDB	Componente de persistência de dados
Services	API que comunica com a base de dados

Tabela 2.2: Definição dos Componentes

Como é possível perceber pelo diagrama da figura 2.4, o componente E-goi Builders contém todos os editores presentes no produto E-goi: o componente Email-Builder, o componente Autobot-Builder e o componente HTML-Builder. Todos estes componentes estendem o componente Builder. Os editores comunicam ainda com o componente Services que, por sua vez, comunica com o componente de persistência de dados MariaDB. Separado do E-goi Builders podemos constatar que existe o componente EasyForms, o editor de formulários,

que comunica com o componente BackOffice que por sua vez faz a persistência de dados no componente MariaDB.

Os três editores do produto E-goi consistem em interfaces gráficas, desenvolvidas em Angular (apresentado na Secção 3.2.2), que permitem aos seus utilizadores criar *e-mails* através de *drag & drop*, autobots (fluxos de automação para o envio de campanhas de *e-mail* ou outras funcionalidades) e campanhas de *e-mail* através da introdução e edição de código HTML. Todos os editores estendem o pacote Builder (também desenvolvido em Angular) que consiste na abstração de algumas funcionalidades comuns a todos os editores como o conceito de *widget* e os seus atributos gerais, menus presentes em todos os editores assim como estilos e predefinições que garantem familiaridade entre todos os editores. Uma vez que todo o componente E-goi Builders apenas possuí lógica para a interface dos diversos editores, é necessária lógica do lado de servidor que possa tratar e guardar a informação obtida pelos editores. Para esse efeito, o componente E-goi Builders comunica com o componente Services. Este componente trata-se de uma Representational State Transfer (REST) Application Program Interface (API) desenvolvida na linguagem de programação PHP [12] que contém grande parte da lógica de negócio e do tratamento de dados sendo a comunicação entre os dois componentes feita através do protocolo Hyper Text Transfer Protocol Secure (HTTPS). O componente Services comunica ainda com o componente MariaDB, responsável por toda a persistência de dados. Este componente é desenvolvido usando a tecnologia MariaDB [13], um sistema de gestão de bases de dados relacional com base em MySQL [14].

Através da análise do diagrama da figura 2.4, é também possível perceber que o editor EasyForms está contido no componente BackOffice. Este componente trata-se de um monólito complexo que contém diversas funcionalidades, incluindo o editor de formulários. Este componente é a base de todo o produto E-goi e tem existido um trabalho contínuo de simplificação do mesmo sendo que a migração e divisão do componente BackOffice em componentes mais pequenos e distribuídos tem sido uma preocupação para a empresa. O componente BackOffice é desenvolvido em várias linguagens de programação como PHP, HTML, Cascading Style Sheets (CSS), Javascript, SQL, entre outras, contendo lógica de negócio assim como interfaces e lógica de renderização. O componente EasyForms define toda a lógica, quer de negócio quer aplicacional, específica do editor de formulários. A persistência dos dados é feita no componente MariaDB.

A arquitetura do editor de formulários traz problemas de escalabilidade e manutenção, identificados na Secção 1.3. O facto de estar contido num componente pesado e bastante complexo, dificulta a manutenção uma vez que existem bastantes dependências internas. A escalabilidade também fica comprometida uma vez que é complexo conseguir aumentar o número de instâncias do componente EasyForms estando este dentro de um componente maior.

Um sistema informático apenas fica disponível se for implantado e, por isso, é importante perceber como implantar o sistema fisicamente, ou seja, como é que o software será distribuído pelas máquinas físicas e como é que estas comunicaram entre si. Para tal, foi utilizado um diagrama UML, o diagrama de implantação, para ilustrar a distribuição do editor EasyForms.

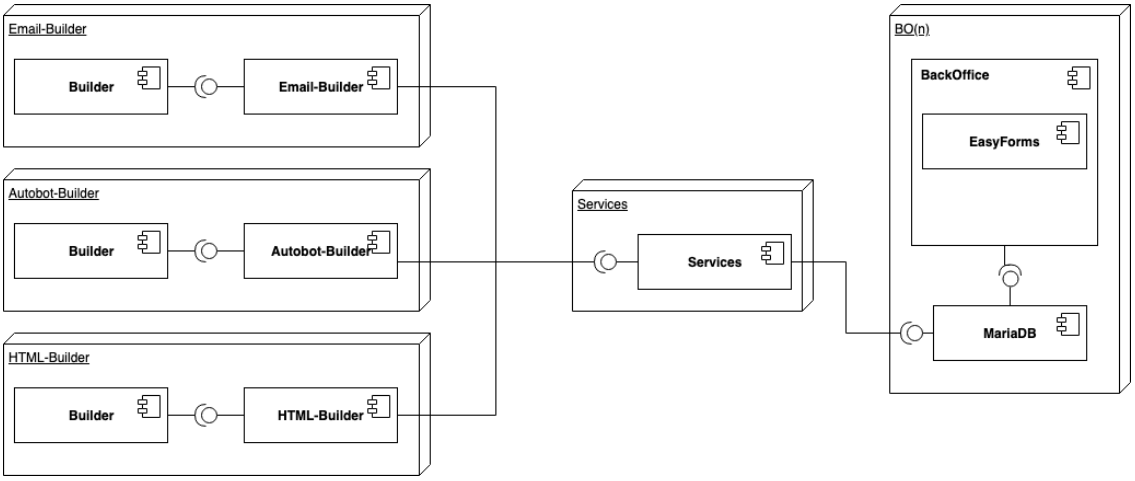

Figura 2.5: Diagrama de Implantação do Editor EasyForms

Através do diagrama representado na figura 2.5 é possível perceber como é que o sistema se encontra distribuído em diferentes máquinas físicas. Existe uma (ou mais, dependendo da necessidade) máquina para cada editor do produto E-goi, sendo que cada máquina contém uma instância do editor assim como uma instância do pacote Builder, que é estendido por cada editor existente (Email-Builder, HTML-Builder e Autobot-Builder). Existe também pelo menos uma máquina com uma instância do componente Services, que pode ser replicada consoante a necessidade do sistema. Por fim é possível perceber que existem N máquinas chamadas BO(N) que têm instâncias do monólito BackOffice, onde o atual editor de formulários (EasyForms) está incluído. Este modelo de implantação não garante uma boa escalabilidade do sistema uma vez que, se existir a necessidade de aumentar as instâncias do EasyForms, é necessário criar uma nova instância do BackOffice que conta com inúmeras funcionalidades para além do EasyForms, o que traduz um problema tanto de manutenção como de escalabilidade.

2.5 Sumário

Neste capítulo foram apresentados os conceitos de lista de contactos, formulários e *landing pages*. Depois de enumerados e definidos estes conceitos associados ao projeto, torna-se possível perceber o projeto de uma forma mais global assim como algumas escolhas feitas ao longo do mesmo. Foi também apresentada a contextualização ao produto E-goi, quer de um ponto de vista do editor gráfico EasyForms assim como do ponto de vista técnico.

No capítulo seguinte, serão listadas e analisadas as soluções existentes no mercado que resolvem os problemas identificados, as tecnologias utilizadas para os resolver assim como apresentados métodos e técnicas importantes para o desenvolvimento do projeto.

Capítulo 3

Estado da Arte

De modo a aprofundar o conhecimento relativamente a uma área em específico e perceber os padrões utilizados e as soluções implementadas, é necessário fazer um estudo do estado da arte. No âmbito desta dissertação, é apresentada uma comparação de ferramentas de *marketing* digital que apresentam a funcionalidade de criação e gestão de formulários fazendo uma análise comparativa de vários fatores. São também apresentados todos os métodos e tecnologias utilizados no âmbito deste projeto, quer utilizados no nível da análise quer utilizados no desenvolvimento de *software*.

3.1 Soluções para Criação de Formulários

Existem diversas abordagens para um editor de formulários, todas com as suas vantagens e desvantagens e, por isso, é necessário perceber no que diferem estas abordagens. Este passo é essencial para conseguir analisar ferramentas de criação e gestão de formulários e perceber as suas implementações.

Para obter uma amostra realista das soluções existentes hoje em dia, foram identificadas diversas ferramentas que permitem a criação e gestão de formulários. Os principais motivos para a escolha das ferramentas foram as funcionalidades e popularidade das ferramentas. Esta análise compara as ferramentas na sua globalidade e especificamente na funcionalidade de criação e gestão de formulários.

3.1.1 Mailchimp

O Mailchimp [15] é uma ferramenta de *marketing* digital criada em 2001 pelos dois membros da agência Rocket Science Group, Ben Chestnut e Dan Kurzius. Esta ferramenta permite a criação de campanhas de *e-mail*, anúncios *online*, *landing pages* e formulários assim como a importação de listas de contactos e automatização de envios.

O editor de formulários desta ferramenta, contrariamente ao habitual, não consiste num sistema *drag & drop* apresentando, no entanto, uma área de edição e uma barra com os *widgets* disponíveis no lado direito. Para adicionar um *widget* ao formulário, o utilizador apenas carrega no *widget* desejado e este é adicionado no fim do formulário sendo apresentadas as opções especificas do mesmo na barra lateral, como é possível ver na Figura 3.1, sendo possível reorganizar a ordem dos campos arrastando os mesmos.

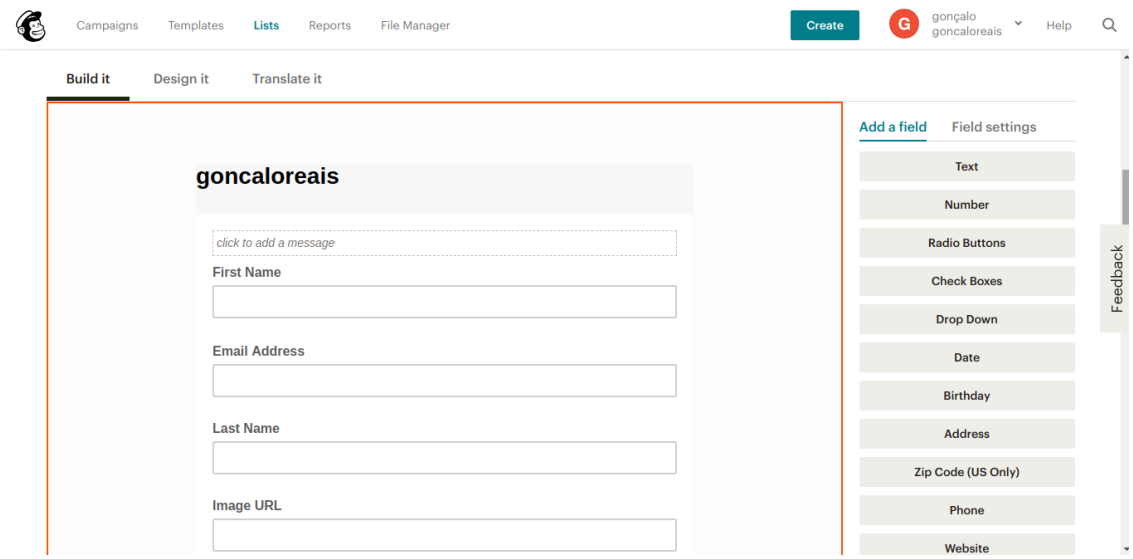

Figura 3.1: Criação de Formulários no Mailchimp

O editor de formulários encontra-se integrado na plataforma do Mailchimp, utilizando as listas de contactos para fazer o envio e a sua automação para vários clientes simultaneamente. É também possível criar novos campos na lista dentro do próprio editor, não sendo necessário abandonar o mesmo para criar um novo campo na lista de contactos.

Relativamente a opções, cada campo apresenta as suas opções específicas tais como o texto a apresentar no campo, se o campo é de preenchimento obrigatório ou não, se o campo é visível ou invisível, texto de ajuda, entre outras. Também é possível, na aba "Design It"(Figura 3.1), alterar opções gerais do formulário tais como a cor de fundo, cor de texto dos campos, cor do botão de submissão, entre outras opções gráficas. Ainda relativamente à vertente gráfica do formulário, não existe o conceito de colunas, sendo o formulário uma única coluna assim como não é possível fazer formulários com mais que uma página.

A plataforma apenas permite exportação de formulários *pop-up*, para serem integrados em páginas web assim como a integração de formulários em *landing pages*. Para simplificar este processo, a plataforma apresenta *templates* definidos que o utilizador pode editar e personalizar consoante as suas necessidades.

Relativamente aos envios, não é possível definir um limite de submissões do formulário assim como alterar o estado do formulário em si (ativo ou inativo). É possível ativar um *e-mail* de *double opt-in*, ou seja, um *e-mail* de confirmação de inscrição não sendo possível alterar o aspeto deste *e-mail*.

Relativamente a estatísticas do formulário, é possível perceber o número de visitas que um formulário obteve assim como a taxa de submissões desse formulário consoante as visitas.

3.1.2 123FormBuilder

O 123FormBuilder [16] consiste numa ferramenta de criação de formulários fundada em 2008 por Florin Cornianu e Tudor Bastea. Esta ferramenta não permite a importação de listas sendo o seu objetivo unicamente a criação de formulários.

O editor de formulários do 123FormBuilder consiste num sistema *drag & drop* e apresenta uma barra lateral com os *widgets* disponíveis e uma área editável onde os *widgets* são colocados, estruturando assim o formulário, como é possível ver na Figura 3.2. Quando um *widget* é selecionado ou arrastado, a barra lateral apresenta as opções específicas do *widget*.

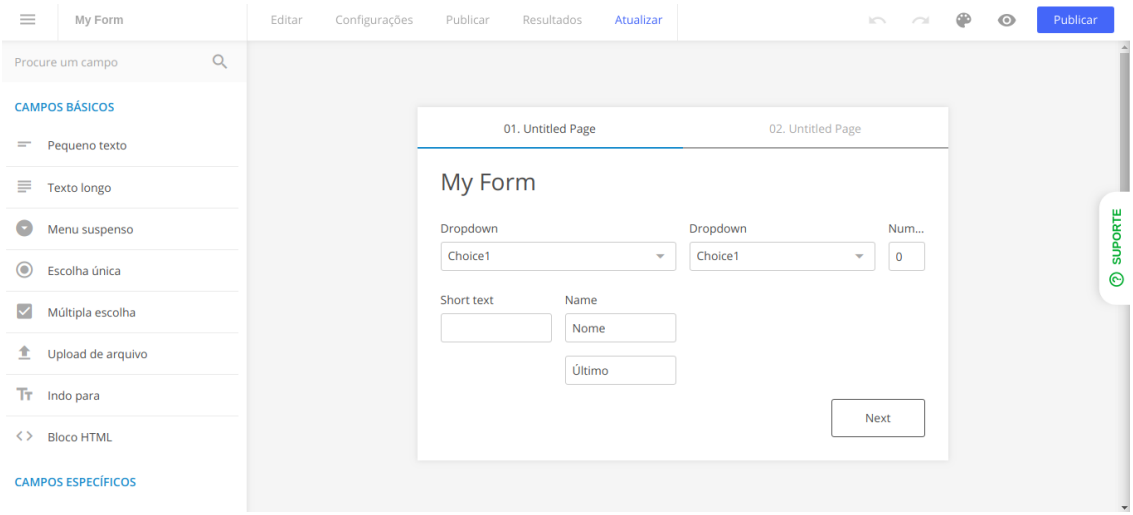

Figura 3.2: Criação de Formulários no 123FormBuilder

Por não permitir a criação de listas de contactos, esta plataforma de criação e gestão de formulários não conta com algumas funcionalidades presentes noutras ferramentas, nomeadamente, mapeamento de campos e criação de campos na lista. Paralelamente, é importante referir que esta plataforma permite a criação de cinco formulários de uma forma gratuita sendo possível adquirir um plano pago para acesso a mais formulários.

Relativamente a opções, o 123FormBuilder conta com opções gerais do formulário, tais como largura do formulário, cor de fundo, altura do cabeçalho, entre outras. Para além dessas, cada *widget* tem as suas opções específicas tais como texto a apresentar no campo, cor e tipos de letra do texto, valor predefinido, texto de ajuda, entre outros. Este editor permite os conceitos de colunas, permitindo dispor *widgets* lado a lado assim como o conceito de paginação, permitindo criar formulários com várias páginas.

No 123FormBuilder, é possível obter o código do formulário para o integrar numa página web assim como obter um link e até integrar o formulário em diversas plataformas externas tais como Shopify, Facebook, Wordpress, Unbounce, Twitter, entre outros.

Uma vez que a plataforma não permite a criação de listas, a funcionalidade *double opt-in* não faz sentido, porém, o 123FormBuilder permite que o utilizador solicite uma aprovação de resposta. Ou seja, o resultado da submissão apenas é contabilizado nas estatísticas quando o visitante confirma a submissão no seu *e-mail*.

Relativamente a estatísticas, esta plataforma permite listar todas as respostas obtidas, podendo estas serem exportadas assim como gerar relatórios através de filtros definidos pelo utilizador, como por exemplo respostas a uma pergunta em específico.

3.1.3 Formstack

O Formstack [17] consiste numa ferramenta de criação de formulários criada em 2006 por Ade Olonoh. O objetivo desta ferramenta é especificamente a criação de formulários não permitindo a importação de listas nem automatização de envios.

O Formstack conta com um editor de formulários *drag & drop* apresentando uma barra lateral com os *widgets* disponíveis e uma área de edição onde os *widgets* são colocados. Como na maioria dos editores, a barra lateral apresenta as opções específicas de um *widget* quando este é selecionado, como é possível perceber na Figura 3.3.

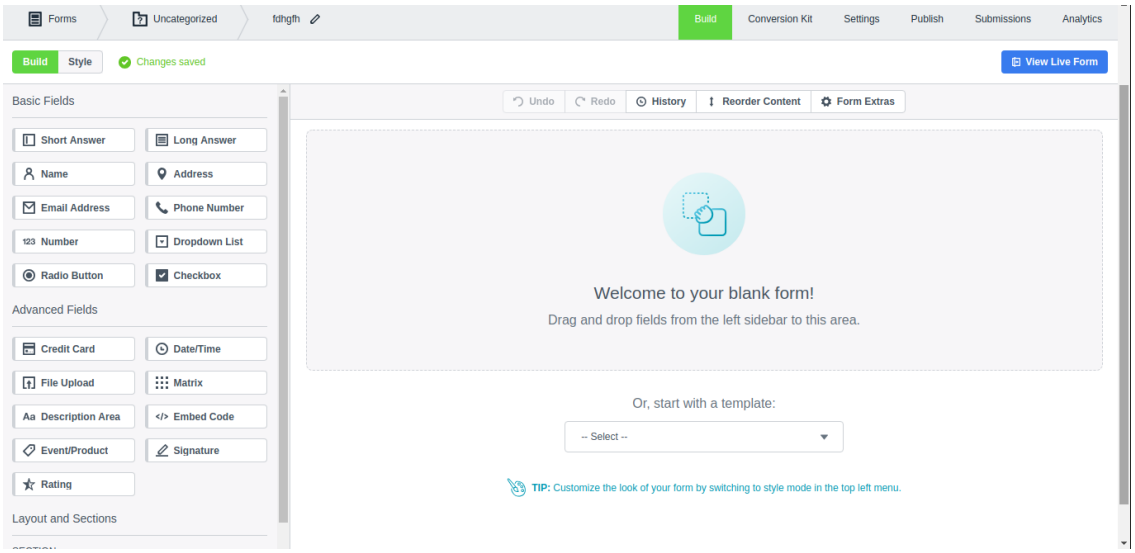

Figura 3.3: Criação de Formulários no Formstack

Esta plataforma não permite a criação de listas de contactos e, por esse motivo, não possui qualquer tipo de automação de envios sendo possível o envio de *e-mails* de confirmação de submissão assim como a alteração do texto a apresentar nesse *e-mail*.

Como as plataformas anteriores, o Formstack permite a edição de opções globais do formulário, quer opções gráficas como cor de fundo, largura do formulário, tipos e cores de letras, entre outras, assim como opções do formulário em si, nomeadamente linguagem do mesmo, estado, nome do formulário, entre outras.

Relativamente a publicações, o Formstack permite a exportação de formulários através de *links*, código para embeber em páginas *web* e diversas integrações nomeadamente com Facebook, Wordpress ou Drupal assim como em formato QR code.

O Formstack permite também visualizar uma lista com todas as submissões efetuadas assim como criar filtros específicos sobre as submissões. Para além disso, a plataforma apresenta também uma zona de estatísticas com a informação do número de visitas, número de submissões, percentagem de submissões, abandonos e percentagem de abandono durante o corrente dia, última semana ou último mês.

3.1.4 SendPulse

O SendPulse [18] é uma ferramenta de *online marketing* criada em 2015 por Konstantyn Makarov. Os utilizadores do SendPulse conseguem criar formulários, *newsletters*, campanhas, importar contactos e automatizar envios.

Relativamente ao editor de formulários, o SendPulse utiliza um sistema *drag & drop*, permitindo também a adição de *widgets* através do clique do rato. É então possível adicionar um *widget* arrastando o mesmo da barra lateral para o sítio pretendido na área de edição.

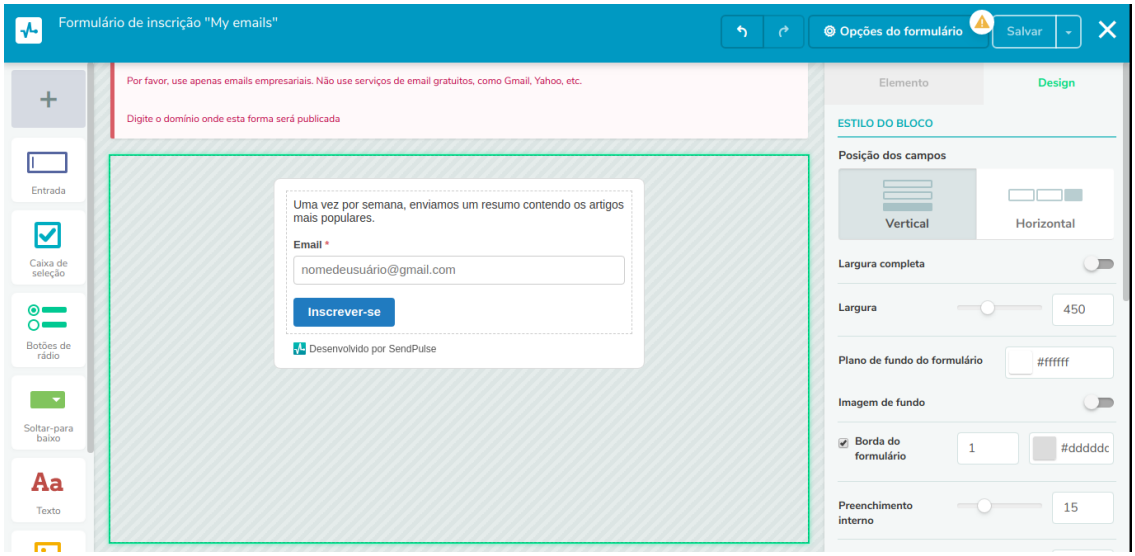

Figura 3.4: Criação de Formulários no SendPulse

Como a ferramenta permite a criação e importação de listas de contactos, é possível fazer um mapeamento dos campos do formulário com os campos da lista de contactos de modo a guardar as suas submissões diretamente na lista de contactos.

Mais uma vez, o SendPulse é uma ferramenta que apresenta opções globais do formulário como largura do mesmo ou espaçamento dos diversos campos, entre outras assim como opções específicas de cada campo como tipos e cores de letra, tamanho, alinhamento, entre outros. Não é permitida a criação de formulários com mais que uma página assim como o conceito de colunas, não sendo permitido dispor *widgets* lado a lado.

É possível definir uma página de agradecimento de sistema ou o reencaminhamento para uma página definida pelo utilizador.

Relativamente a estatísticas, apenas é possível perceber o número de visitas ao formulário. No entanto, é possível integrar funcionalidades externas como Google Analytics, por exemplo, de modo a controlar melhor as estatísticas dos formulários.

3.1.5 JotForm

O JotForm [19] é uma ferramenta de criação de formulários criada em 2006 por Aytekin Tank que permite a criação e gestão de formulários.

Esta ferramenta também utiliza um sistema de *drag & drop* para a criação dos seus formulários. O editor permite arrastar *widgets* de uma barra lateral para uma área de edição ou apenas clicar nos *widgets* para estes serem adicionados no fim dos *widgets* já existentes, como é possível ver na Figura 3.5.

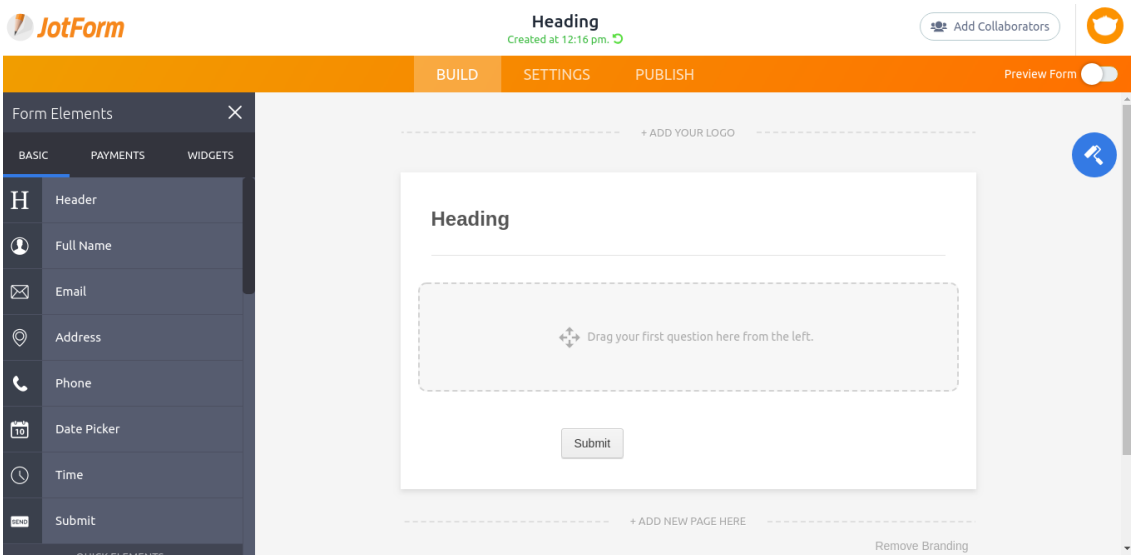

Figura 3.5: Criação de Formulários no JotForm

O JotForm é uma plataforma de criação e gestão de formulários que não permite a criação ou importação de listas de contactos e, por esse motivo, não permite automação de envios dos formulários. No entanto, a plataforma oferece a funcionalidade de *e-mails* de confirmação e *autoresponders* (*e-mails* automáticos de resposta) aos visitantes do formulário.

É possível editar toda a parte gráfica dos formulários na aba "BUILD", presente na Figura 3.5, tanto opções gerais dos formulários como opções específicas de cada *widget*.

Na aba "SETTINGS"é possível configurar todas as opções relativas ao formulário em si e não ao seu design, nomeadamente nome e estado do formulário, condições comportamentais, automatização de respostas, integrações com plataformas externas e a edição da página de agradecimento a aparecer depois de uma submissão.

Na aba "PUBLISH"é possível definir todas as opções relativas à publicação e exportação do formulário. Para além da opção do *link* direto, a plataforma permite a exportação de código a embeber em páginas *web*, envio via *e-mail*, *download* do formulário no formato Portable Document Format (PDF), integração com plataformas externas e ainda no formato *QR code*.

A plataforma permite ainda listar todas as submissões assim como gerar relatórios em diversos formatos com a informação relativa ao número de submissões, percentagem de submissões consoante as visitas, entre outras.

3.1.6 Wufoo

O Wufoo [20] permite a criação de formulários e foi criado em 2006 pela SurveyMonkey. O objetivo desta ferramenta é a criação de formulários não permitindo a importação de contactos nem a automação de envios.

A ferramenta apresenta um editor de formulários *drag & drop* permitindo arrastar *widgets* de uma barra lateral para uma área de edição assim como adicionar *widgets* ao formulário através de um clique.

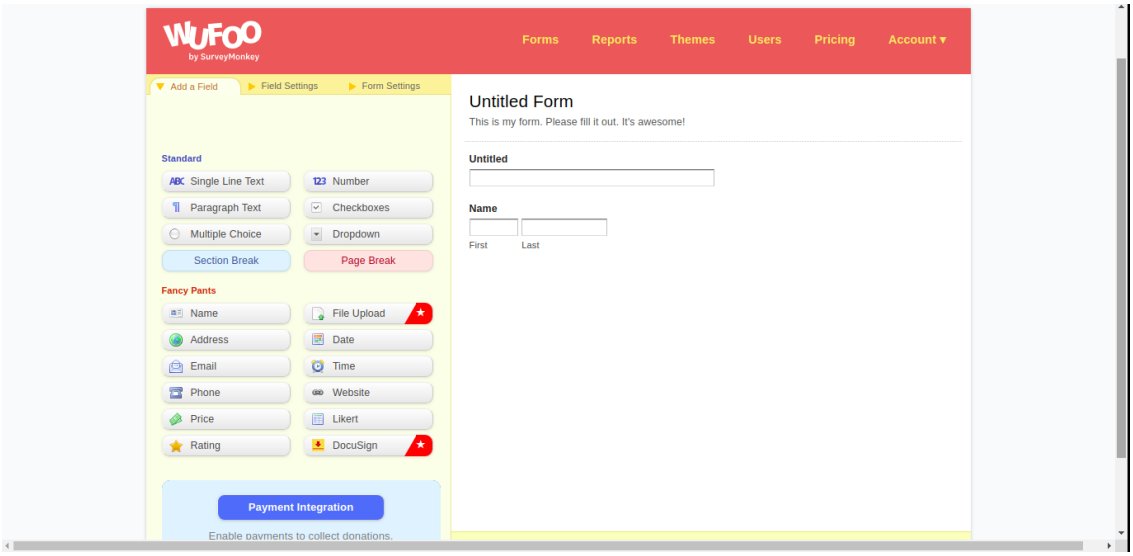

Figura 3.6: Criação de Formulários no Wufoo

Como é possível ver na Figura 3.6, o editor apresenta três abas: "Add a field", "Field Settings"e "Form Settings". Como habitualmente acontece, a primeira aba tem como objetivo apresentar todos os *widgets* existentes no editor, a segunda apresenta as opções específicas do campo selecionado e a terceira apresenta todas as opções relacionadas com o formulário em si e não com a edição gráfica do mesmo.

Nas definições gerais do formulário (terceira aba), o utilizador pode editar o título e descrição do formulário, língua, alinhamento dos *widgets*, opções de *tracking*, tais como o uso de ferramentas externas, opções de confirmação, nomeadamente o envio de um *e-mail* de confirmação e definição da mensagem e opções de envio tais como limitar envios por Internet Protocol (IP), entre outras.

Relativamente à análise de estatísticas e respostas, é possível listar todas as submissões efetuadas assim como aplicar filtros às mesmas. É também possível criar relatórios personalizados em vários formatos com filtros específicos, por formulário.

3.1.7 Google Forms

O Google Forms [21] é a ferramenta de criação de formulários da Google, criada em 2012. Esta ferramenta não permite a importação de listas de contactos permitindo adicionar contactos para os quais o formulário será enviado.

Como é possível verificar na Figura 3.7, o Google Forms possui uma abordagem um pouco diferenciada das soluções analisadas anteriormente, não tendo aplicado uma abordagem *drag & drop* no seu editor de formulários. Para adicionar um novo campo no formulário é necessário clicar no botão de adicionar presente na *toolbox* lateral e, posteriormente, definir o tipo de campo.

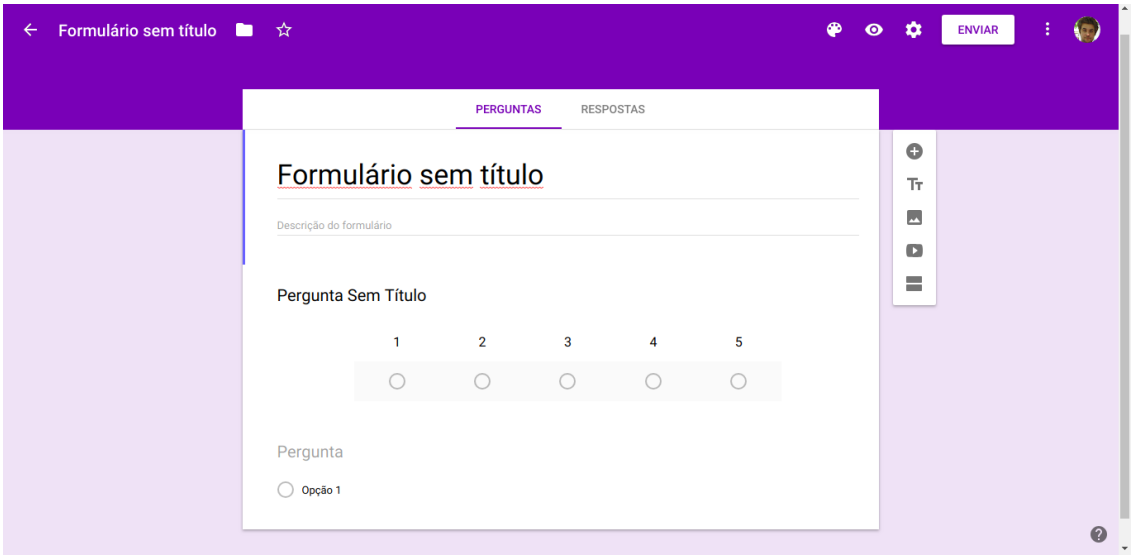

Figura 3.7: Criação de Formulários no Google Forms

Relativamente às opções do editor, não existe o conceito de colunas, não sendo permitido a apresentação de campos lado a lado no formulário. É permitido, no entanto, a criação de formulários com mais de uma secção e definir condições entre as diferentes secções. As opções gráficas são bastante limitadas em comparação a outras soluções sendo apenas possível alterar a cor do tema do formulário.

Existem ainda definições específicas do formulário quer relativas a submissões, nomeadamente recolher os endereços de *e-mail* dos visitantes que submetem uma resposta assim como requerer o inicio de sessão de modo a limitar a uma resposta por *e-mail*.

É possível enviar o formulário através de *e-mail*, obter um *link* direto para o mesmo ou ainda o código que pode ser embebido numa página *web*.

Como é possível ver na Figura 3.7 existe uma aba de "Respostas"onde é possível visualizar uma lista de todas as respostas ao formulário assim como definir se o formulário deve ou não aceitar mais submissões. Para além destas opções é ainda permitida a exportação das respostas em vários formatos.

3.1.8 Análise Comparativa

Após uma análise às alternativas existentes no mercado, é importante fazer uma análise comparativa, de modo a perceber os pontos onde as ferramentas diferem e a justificação para essa diferenciação.

Através da análise, é possível dividir as ferramentas em dois grandes grupos: ferramentas integradas em plataformas que permitam a criação e importação de listas de contactos, como o E-goi, Mailchimp e SendPulse, e ferramentas que apenas permitem a criação e gestão de formulários, como o 123FormBuilder, Formstack, JotForm, Wufoo e Google Forms.

O estudo destas alternativas permitiram perceber que existem funcionalidades presentes na maioria dos editores, tenham eles integração com listas de contactos ou não, tais como o número de subscrições através de um formulário assim como o número de visitas do

mesmo, a funcionalidade de ativação do *double opt-in*, a exportação do formulário em diferentes formatos, a existência de opções globais, quer gráficas quer comportamentais, do formulário assim como a existência de *templates* predefinidos. Podemos afirmar que estas são funcionalidades básicas que qualquer editor de formulários deve incluir.

Por outro lado, existem funcionalidades apenas presentes nos editores que contemplam uma integração com a lista. No âmbito desta dissertação, o novo editor terá essa integração e, por isso, é necessário perceber quais são as funcionalidades específicas deste tipo de editores mais comuns. Através da análise podemos perceber que a criação de campos da lista novos no próprio editor, o mapeamento dos campos do formulário com os campos da lista assim como a alteração dos estados dos formulários são funcionalidades bastante comuns neste tipo de editores.

Outro facto percetível através desta análise, é o facto dos editores do Mailchimp e SendPulse serem bastante mais fechados do que, por exemplo, os editores do JotForm, Wufoo ou 123FormBuilder. Esta diferença deve-se à finalidade dos formulários. Uma vez que os formulários dos primeiros editores referidos têm como objetivo a manipulação de uma lista de contactos, é expectável que apresentem mais limitações comparativamente a editores que apenas guardarão os dados, sem os inserirem numa lista. Esta diferença é percetível através da análise de algumas funcionalidades, como se constata na Tabela 3.1:

	Com Contactos			**Sem Contactos**				
	E-goi	Mailchimp	SendPulse	123Formbuilder	Formstack	JotForm	Wufoo	Google Forms
Conceito de Colunas	x			x	x			
Paginação				x		x	x	
Exportação do Formulário	x			x	x	x	x	
Lógica entre Campos				x	x		x	x

Tabela 3.1: Análise Comparativa de Ferramentas

As indicações (x) na Tabela 3.1 traduzem se a ferramenta em questão possui a funcionalidade em questão e as colunas "Com Contactos"traduzem as plataformas que têm integração com listas de contactos enquanto que as colunas "Sem Contactos"traduzem as plataformas que não têm integração com listas de contactos e cujo único propósito é a criação e gestão de formulários.

Através da Tabela 3.1 podemos perceber que existe uma maior liberdade para o utilizador, relativamente às quatro funcionalidades analisadas, nas ferramentas que não apresentam uma integração com a lista pelos motivos referidos anteriormente.

Relativamente à exportação de formulários, conseguimos perceber que, ferramentas mais completas como Mailchimp e SendPulse não permitem a exportação de formulários. Isto deve-se ao facto de estas ferramentas, habitualmente, terem a funcionalidade de criação e gestão de *landing pages* permitindo assim importar os formulários para páginas que podem ser partilhadas em massa através das listas de contactos.

Dentro do seu espectro, o podemos considerar que o editor de formulários do E-goi oferece bastante liberdade aos seus utilizadores, especialmente em funcionalidades que outros editores não oferecem, por exemplo o conceito de colunas, exportação de formulários, configuração do *e-mail* de *double opt-in*, entre outros. Esta liberdade traz constrangimentos ao sistema, tornando-o confuso para o utilizador e dificultando a implementação tanto a nível arquitetural como a nível técnico da funcionalidade de criação e gestão de formulários.

3.2 Métodos, Técnicas e Tecnologias

Ao longo de todo o desenvolvimento do projeto existiu um estudo constante de métodos, técnicas e tecnologias à medida das necessidades. Nesta secção são agrupadas todas os conceitos estudados para levar a cabo as diferentes etapas do projeto apresentando informação relevante sobre cada conceito.

3.2.1 Análise

Nesta subsecção são apresentados todos os conceitos estudados relativamente à Análise do projeto (Capítulo 4). São apresentados os conceitos New Concept Development, o padrão utilizado para o desenvolvimento da ideia e do produto, Analytic Hierarchy Process, um método matemático de apoio à decisão, Business Model Canvas, uma técnica utilizada para avaliar os custos e benefícios de um produto, Testes de Usabilidade, testes utilizados para medir a usabilidade de um sistema, Análise de Requisitos, técnica utilizada para o levantamento de requisitos de um sistema e, por fim, UML, uma linguagem utilizada para o desenvolvimento de diagramas.

New Concept Development

O processo de desenvolvimento de um produto possui diferentes etapas tais como Fuzzy Frond End, New Product Development e Comercialização. As etapas New Process Development e Comercialização possuem terminologia própria, bem definida e universal, contrariamente ao passo Fuzzy Front End. O conceito New Concept Development é criado em 2002 por Peter A. Koen com o objetivo de resolver a falta de uma linguagem comum e bem definida na vertente de Fuzzy Frond End do processo de desenvolvimento [22]. O modelo é representado por três partes fundamentais:

O motor:
: mobiliza os cinco elementos chave do modelo representando a vertente executiva da empresa e todas as decisões internas da empresa;

Os cinco elementos chave:
: traduzem as fases necessárias para transformar uma oportunidade num conceito e avançar para a próxima fase do processo de inovação (Identificação da Oportunidade, Análise da Oportunidade, Geração de Ideias, Seleção de Ideias e Definição do Conceito);

Fatores Influenciadores:
: camada onde assenta o todo o modelo e que traduz todos os fatores externos e não controlados pela empresa que podem afetar o processo de inovação.

Aplicando este conceito, é possível seguir uma metodologia estruturada e bem definida para a primeira etapa do processo de desenvolvimento, como é descrito na Secção 4.2.2.

Analytic Hierarchy Process

Este método foi criado por Tomas L. Saaty em 1980 [8] e é um dos métodos mais conhecidos de apoio à decisão. O seu conceito principal consiste em desconstruir a ideia em fragmentos cada vez mais pequenos de modo a tonar a sua análise mais fácil. Para atingir esse objetivo, o método divide-se em três passos principais: Divisão Hierárquica, Definição de Prioridades e Consistência.

A divisão hierárquica do Analytic Hierarchy Process (AHP) consiste na definição do objetivo, dos critérios de resolução do problema e, por último, nas alternativas que resolvem o problema. Dispondo estes conceitos numa hierarquia percebemos a relação entre eles sendo que no topo da pirâmide está o objetivo da decisão, seguido dos critérios a utilizar e, por último, as opções consideradas.

Depois da divisão hierárquica, é necessário definir prioridades entre os diferentes pares de critérios escolhidos. Esta priorização é baseada na opinião sendo, por isso, subjetiva, tornando-se então necessário calcular o índice de consistência de modo a validar a decisão. Para fazer a priorização de critérios foi utilizada a tabela que apresenta a relação entre cada par de critérios utilizando a escala de Saaty [23]. A escala de Saaty consiste numa escala numerada de 1 a 9 onde 1 corresponde a uma igualdade de importância entre critérios, onde 3 corresponde a uma fraca diferença de importância entre critérios e 9 corresponde a uma diferença de importância absoluta.

Para finalizar a utilização do método, é necessário garantir a consistência da decisão, uma vez que a priorização dos critérios é baseada em opinião e é subjetiva. Para isso são calculados os índice de consistência e a razão de consistência. Desta forma é possível medir a consistência dos julgamentos relativamente a grandes amostras. Se o resultado do cálculo do rácio de consistência for menor que 0.1, consideram-se os critérios consistentes.

Para calcular o rácio de consistência é primeiro necessário calcular o índice de consistência que é definido pela seguinte formula:

$$\frac{(\lambda_{max} - n)}{(n-1)}$$

onde n corresponde à ordem da matriz e λ_{max} corresponde ao maior valor próprio da matriz resultado da multiplicação da matriz comparativa de critérios multiplicada pela matriz de prioridades relativas.

Depois de todos os cálculos é possível tirar conclusões e tomar uma decisão com base nos resultados obtidos. Este método foi utilizado na Secção 4.2.2 para tomar uma decisão relativa à abordagem para a implementação do editor de formulários.

Business Model Canvas

O Business Model Canvas foi apresentado por Alex Osterwalder na sua tese de doutoramento *The Business Model Ontology - A Proposition In A Design Science Approach*, em 2004 [24]. Na sua tese de doutoramento, Osterwalder define um modelo de negócio como *"business model is a representation of how a company buys and sells goods and services and earns money"* [24]. Todos os aspetos relativos à venda de produtos ou serviços estão contemplados no Canvas que o autor apresenta, sendo divididos em blocos. No âmbito deste projeto, o

Business Model Canvas foi aplicado a uma funcionalidade e não a um produto ou serviço: o editor de formulários. O Business Model Canvas é utilizado na Secção 4.2.4 com o objetivo de perceber como será introduzido o novo editor de formulários, perceber os seus custos e as fontes de retorno associadas.

Testes de Usabilidade

Uma das maneiras mais eficazes de avaliar a usabilidade de um sistema é através de um teste de usabilidade. No seu livro *Usability Testing Essentials: Ready, Set... Test!* [25], Carol M. Barnum define testes de usabilidade como *"When I refer to usability testing, I mean the activity that focuses on observing users working with a product, performing tasks that are real and meaningful to them"*. Através desta definição é possível perceber que um teste de usabilidade consiste na observação de utilizadores a levar a cabo tarefas reais num sistema. O resultado de um teste de usabilidade mede a qualidade da usabilidade de um produto através da execução de tarefas reais.

Segundo Jakob Nielsen [26], para executar testes de usabilidade, são necessários utilizadores representativos da população que irá usar ou que usa o sistema a ser testado. Apesar de não existir uma maneira totalmente correta para escolher populações para testes de usabilidade, no artigo *234 Tips and Tricks for Recruiting Users as Participants in Usability Studies* [27], Nielsen aborda o tópico dando dicas e conselhos para levar a cabo este processo. São abordados diversos temas tais como as características dos utilizadores, a existência de agências especializadas em recrutar a população à qual será feito o teste, os custos que podem estar associados a este recrutamento assim como alguns documentos modelo para levar a cabo os testes de usabilidade. É importante levar estes tópicos em consideração de modo a tomas decisões mais acertadas quer para a qualidade dos testes quer para o custo para a organização.

Uma das formas de reduzir o custo de recrutamento é utilizar recursos humanos das próprias instituições para testar o sistema em questão. Quando confrontados com esta questão, Joseph S. Dumas e Janice C. Redish, no seu livro *A Practical Guide to Usability Testing* [28], fazem uma análise comparativa entre os prós e os contras da utilização de funcionários de uma instituição para levar a cabo testes de usabilidade dentro da mesma. *"The most serious reason for not using company employees, however, is still that employees may not represent the actual users. Even if the employees are new to the particular product, they may know too much about similar products."*. Desta forma, é possível perceber os prós e os contras da utilização de recursos humanos de uma empresa num teste de usabilidade relacionado com a mesma.

Na Secção 4.1.1 é definido e implementado um teste de usabilidade que tem como objetivo perceber, a um nível global, a usabilidade do editor EasyForms. Na Secção 7.2.2, de modo a conseguir fazer uma comparação, é implementado novamente o mesmo teste de usabilidade ao novo editor de modo a comparar as respostas de ambos e tirar conclusões relativamente à evolução.

Análise de Requisitos

A definição de requisitos é um passo essencial para definir como é que um sistema será implementado. No seu livro *Mastering the Requirements Process: Getting Requirements*

Right, Suzanne Robertson e James Robertson [29] escreveram *"Requirements are what the software product, or hardware product, or service, or whatever you intend to build is meant to do and to be."*. Esta afirmação define os requisitos como a missão e a descrição de um produto, seja ele *software* ou não. O levantamento de requisitos é um passo importante em engenharia de *software* uma vez que define a maneira como todo o *software* será desenvolvido, desde as funcionalidades do mesmo como as condições nas quais este será desenvolvido.

Existem dois tipos de requisitos: requisitos funcionas e requisitos não funcionais. Suzanne Robertson e James Robertson [29] definem requisitos funcionais como *"A functional requirement describes an action that the product must take if it is to be useful to its operator..."*. Esta definição traduz o conceito de requisito funcional como uma função que o sistema tem de fazer. Já os requisitos não funcionais são definidos como *"Colloquially speaking, NFRs have been referred to as "-ilities" (e.g., usability) or "-ities" (e.g., integrity), i.e., words ending with the string "-ility" or "-ity."* por Lawrence Chung e Julio Leite [30], no seu livro *On Non-Functional Requirements in Software Engineering*, definindo assim requisitos não funcionais como requisitos que traduzem características do sistema que não são funcionalidades do mesmo, tais como linguagens de programação a utilizar, questões de usabilidade ou limitações de vários tipos (temporais, visuais, entre outras).

Os requisitos funcionais consistem nas funcionalidades que o sistema terá de cumprir e, por isso, é necessário que sejam claros e entendidos por todos. Mike Cohn [31], no seu artigo *Advantages of User Stories for Requirements*, apresenta uma forma de representar os requisitos funcionais: *user stories*. *"User stories emphasize verbal communication. Written language is often very imprecise, and there's no guarantee that a customer and developer will interpret a statement in the same way."* é a frase onde Cohn explica o aumento da eficácia da comunicação entre desenvolvedores e *stakeholders* utilizando *user stories*. Cohn também demonstra como usar esta técnica através de alguns exemplos. Para garantir que todas as *user stories* são desenvolvidas com os mesmos critérios, foi necessário utilizar o método S.M.A.R.T. [32]. A utilização deste método garante que as *user stories* são *Specific* (específicas), *Measurable* (mensuráveis), *Achievable* (concretizáveis), *Realistic* (realistas) e *Time-based* (com limites temporais definidos). Esta técnica foi utilizada na Secção 4.1.2 para definir os requisitos funcionais do novo editor de formulários do produto E-goi.

Os requisitos não funcionais consistem em características que o sistema tem de cumprir. No seu artigo *"On Non-Functional Requirements"*, Martin Glinz [33] explica *"Although the term 'non-functional requirement' has been in use for more than 20 years, there is still no consensus in the requirements engineering community what non-functional requirements are and how we should elicit, document, and validate them. On the other hand, there is a unanimous consensus that nonfunctional requirements are important and can be critical for the success of a project."*. Não existe uma forma *standard* de fazer o levantamento de requisitos não funcionais apesar de existirem vários métodos para normalizar este processo. Um destes métodos chama-se FURPS, criado por Robert Grady [34] e explicado por Namita Malhotrae e Shefali Pruthi no seu artigo *An Efficient Software Quality Models for Safety and Resilience*. O modelo FURPS+ é um acrónimo para *Functionality* (funcionalidade), *Usability* (usabilidade), *Reliability* (confiabilidade), *Performance* (desempenho) e *Supportability* (suportabilidade) onde o + assume a função de outros requisitos que não se encaixem em qualquer um dos outros princípios. O levantamento de requisitos não funcionais associados ao âmbito deste projeto é definido na Secção 4.1.2.

Unified Modeling Language

O UML é uma linguagem *standard* para o desenvolvimento de diagramas [35]. Criada por Grady Booch, James Rumbaugh e Ivar Jacobson [36] no seu livro *Unified Modeling Language User Guide*, é utilizada mundialmente por profissionais de diversas áreas de negócio distintas para modelar processos de negócio, desenvolvimento de *software*, entre outros. Tenta unificar a comunicação entre profissionais de diversas áreas através da definição de diagramas específicos, com linguagem própria e objetivos distintos. Existem diversos elementos que podem figurar num diagrama UML tais como atores, pacotes, classes, entidades, interfaces, entre outros. Os diversos diagramas consistem na interligação destes elementos consoante a sua relação.

Seguindo a metodologia descrita na Secção 1.5, o desenho de uma solução passa pelo desenvolvimento de artefactos que representem e demonstrem o sistema a ser desenvolvido, sendo que artefacts podem consistir em diagramas UML.

No seu livro *Software Engineering 3*, Scott Millett e Nick Tune [37] definem o conceito de Domain Driven Design (DDD) como *"Domain Driven Design is a process that aligns your code with the reality of your problem domain."*. Seguindo esta abordagem, é importante que exista um artefacto que defina todos os conceitos e definições do domínio. Para a criação deste artefacto pode ser considerada a linguagem UML, utilizando um diagrama de componentes.

Len Bass, Paul Clements e Rick Kazman [36], no seu livro *Software Architecture in Practice* definem arquitetura de software como *"Software Architecture encompasses the structures of large software systems. The architectural view of a system is abstract, distilling away details of implementation, algorithm, and data representation and concentrating on the behavior and interaction of "black box"elements."*. Através desta definição, é possível perceber que para definir a arquitetura de um *software*, é necessário um artefacto que apresente o sistema de uma forma abstrata, apresentando os componenentes do sistema e a interligação entre eles, independentemente de pormenores de implementação como linguagens de programação ou protocolos de comunicação. Para apresentar este artefacto, pode ser utilizado um diagrama de componentes UML.

Estes foram apenas dois exemplos de utilização de diagramas UML. A linguagem UML tem diversas funcionalidades e pode ser utilizada nas áreas mais distintas para a criação de artefactos. No âmbito deste projeto, foram utilizados os seguintes diagramas UML:

Diagrama de Sequência:
: Este diagrama define como é que a lógica do sistema flui consoante diferentes ações. No âmbito do projeto, este diagrama foi utilizado para representar as *user stories* definidas na Secção 4.1.2 e para apresentar o funcionamento do novo editor de formulários na Secção 6.2;

Diagrama de Domínio:
: Este diagrama define uma representação do sistema nas suas diferentes entidades e as relações entre si. No âmbito do projeto, este diagrama foi utilizado na Secção 5.1 para representar uma visão abstrata de todo o sistema atual assim como a nova implementação do editor de formulários, definindo as diversas entidades relacionadas com o sistema e a sua ligação;

Diagrama de Componentes:
: Este diagrama define os componentes existentes no sistema e a sua comunicação entre si. No âmbito do projeto, este diagrama foi utilizado na Secção 5.2 para representar a arquitetura do sistema, contemplando todos os componentes e a maneira como estes comunicam;

Diagrama de Implantação:
: Este diagrama representa o modo com um sistema é implementado, apresentando as diferentes máquinas e a sua comunicação. No âmbito do projeto, este diagrama foi utilizado na Secção 5.3 para representar a implantação do sistema, representando cada máquina com o *software* correspondente e a sua interligação.

Os diagramas UML ajudam a definir de uma forma inequívoca como funcionará o sistema quer para que pessoas menos técnicas consigam compreender o funcionamento assim como pessoas que dominem a linguagem saibam como desenvolver o sistema.

3.2.2 Desenvolvimento de Software

Nesta subsecção são apresentados todos os conceitos estudados relativamente ao desenvolvimento de *software*. É apresentada a *framework* Angular, utilizada ao longo do projeto para o desenvolvimento do editor, Git, o sistema de controlo de versões utilizado, Integração Contínua, metodologia de automação utilizada, Angular Unit Testing, utilizado para testes unitários assim como SonarQube, utilizado para análise ao código desenvolvido.

Angular

O Angular consiste numa *framework open-source* de *front-end* desenvolvida pela Google desde 2010 surgindo como uma versão melhorada da *framework* AngularJS [38], implementada em TypeScript [39]. Os diferentes editores do produto E-goi são desenvolvidos usando esta *framework* assim como o pacote Builder, referenciado no Capítulo 5. Apesar de ser consideravelmente complexa e verbosa, a utilização de Angular traz bastantes vantagens quer de *performante* como de implementação tais como a utilização de *framworks* de gestão de estados reativas (como NgRx), divisão do código em componentes reutilizáveis e documentação de qualidade.

A escolha desta *framework* prende-se com o facto de ser uma opção já conhecida pela empresa, que cumpre os requisitos de escalabilidade e facilidade de manutenção de código definidos assim como pela comunidade ativa existente em torno da mesma. Para além disso, Angular conta com diversos módulos e componentes já implementados que podem ser utilizados e editados de forma livre tais como o Angular Material [40], uma biblioteca que implementa componentes (como modais e janelas colapsáveis) e comportamentos (como *drag & drop* e animações). Para além disso, a *framework* conta também com módulos de *routing* e testes, utilizados no âmbito deste projeto.

É importante perceber o funcionamento interno da *framework* para que seja possível aplicar a mesma ao âmbito deste projeto. Angular apresenta três conceitos fundamentais cuja compreensão é necessária para entender a divisão do código no projeto: módulos, componentes e serviços. Um módulo em Angular [1] consiste num *container* que contém código relacionado com uma certa funcionalidade, parte do domínio ou qualquer outra lógica que

isole uma parte do código. Uma aplicação Angular tem obrigatoriamente pelo menos um módulo, usualmente denominado AppModule, que é iniciado com a aplicação. Um módulo pode importar outros módulos e declarar diversos componentes que existem dentro do próprio módulo. Estas definições são feitas através da utilização do *decorator* @NgModule, que consiste numa função que recebe diferentes parâmetros, como é possível verificar no seguinte excerto de código.

```
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
```

Listing 3.1: Exemplo Módulo Angular [1]

O objecto enviado como parâmetro da função contém o atributo *imports* que permite importar outros módulos, o atributo *providers* que permite a criação de serviços, o atributo *declarations* que permite declarar componentes dentro do contexto do módulo, o atributo *exports* que permite exportar componentes a serem usados outros módulos e ainda o atributo *bootstrap* que identifica o módulo que iniciará a aplicação.

Um componente em Angular [2] consiste numa classe que representa uma parte da interface. Os componentes são criados através de *decorators*, tal como os módulos, que recebem parâmetros e identificam a classe criada, como podemos ver no seguinte excerto de código.

```
@Component({
  selector:    'app-hero-list',
  templateUrl: './hero-list.component.html',
  providers: [ HeroService ],
  stylesUrl: './hero-list.component.css',
})
export class HeroListComponent implements OnInit {
constructor() { }
  ngOnInit() {

  }
}
```

Listing 3.2: Exemplo Componente Angular [2]

No excerto é possível perceber que um componente recebe um objeto com vários atributos tais como o atributo *selector* que define como é que o componente será invocado na interface, o atributo *templateUrl* que especifica o caminho do ficheiro HTML a renderizar com o componente, o atributo *providers* que definem os serviços a serem injetados no componente e o atributo *stylesUrl* que especifica o caminho do ficheiro CSS com os estilos a serem aplicados ao componente. Os componentes em Angular são isolados entre si e, por isso, os estilos aplicados a um componente são apenas aplicado a esse componente sendo também possível definir estilos globais para a aplicação. Um componente comunica com a interface através de métodos específicos como, por exemplo, o método OnInit que é chamado quando o componente é iniciado.

Um serviço em Angular [3] consiste numa classe que pode conter métodos e atributos. Um serviço pode, por exemplo, devolver mensagens na consola do *browser* como podemos ver no seguinte excerto de código.

```
@Injectable({})
export class Logger {
  log(msg: any)   { console.log(msg); }
  error(msg: any) { console.error(msg); }
  warn(msg: any)  { console.warn(msg); }
}
```

Listing 3.3: Exemplo Componente Angular [3]

Através do uso do *decorator* @Injectable, é possível injetar instâncias dos serviços nos componentes que podem usar os seus métodos. A gestão de instâncias dos serviços, para que a mesma instância possa ser usada por componentes diferentes, é controlada através da *Dependency Injection* [41] do Angular.

Git

Desenvolvido pelo criador do sistema operativo Linux, Linus Torvalds, Git [42] é um dos sistema de controlo de versões mais utilizados para o desenvolvimento de *software*. Esta ferramenta funciona como um repositório de ficheiros guardando os diversos estados dos mesmos assim como adicionando códigos específicos a cada versão. As mudanças são feitas de uma forma local e posteriormente enviadas para o servidor onde ficam a ser visíveis a todos os colaboradores do repositório. Existe o conceito de ramos (*branches*) independentes entre si sendo que um ramo é criado sempre a partir de outro já existente. As alterações feitas num ramo não se refletem noutro ramo sem acontecer uma junção de ramos (*merge*).

Existem diversas maneiras de utilizar o sistema Git, utilizando ferramentas como GitLab, GitHub ou BitBucket. A utilização de Git é referido na Secção 6.1.

Integração Contínua

No seu livro *"Continuous Integration"*, Paul Duvall sugere *"If you would like to run frequent integration builds so that it becomes a nonevent on your project - including compilation, rebuilding your database, executing automated tests and inspections, deploying software and receiving feedback - Continuous Integration can help"* [43]. Através desta citação, podemos perceber que integração contínua permite a execução de diversas ações, como compilações de projetos e execução de testes, de uma forma automática. Paul Duvall explica ainda que estes processos podem ser automatizados quando é adicionado código novo a um repositório Git, por exemplo. Através do uso de integração contínua, é possível criar novas funcionalidades, colocá-las em repositórios, executar testes automáticos, compilar a aplicação e apresentar a nova funcionalidade aos clientes de uma forma contínua.

Existem diversas ferramentas que permitem a automação de processos de testes, compilação, *deploy*, entre outras, utilizando o conceito de integração contínua, como por exemplo Jenkins, Cruise Control e Integrity. A utilização de integração contínua no âmbito do projeto é descrita na Secção 6.1

Angular Unit Testing

A *framework* Angular, utilizada para o desenvolvimento do projeto, disponibiliza um módulo de testes, o Angular Testing. Este módulo recorre à *framework* Jasmine [44] para a implementação de testes unitários aos diferentes componentes.

Um teste unitário testa uma unidade isolada de código, testando o comportamento da mesma através da execução de código e comparação de resultados. Os resultados são definidos no próprio teste assim como os valores necessários para a execução do código. Através dos testes unitários é possível perceber se o código a ser testado efetivamente executa o que é suposto, garantindo assim que todo o código desenvolvido executa as funcionalidades esperadas.

A utilização deste módulo para testes unitários prende-se com o facto ser disponibilizada pela *framework* Angular, facilitando assim a integração com o projeto e simplificando a utilização e implementação de testes. Outro fator determinante consistiu na utilização do módulo para testes de outros projetos desenvolvidos em Angular existentes na empresa conseguindo assim utilizar *know-how* presente na empresa.

SonarQube

Criado em 2007, o SonarQube consiste numa *framework open-source* [45] que permite a execução automática de testes ao código desenvolvido detetando erros de compilação, más práticas de programação e falhas de segurança em mais de 20 linguagens de programação diferentes. Desenvolvida pela empresa SonarSource, esta ferramenta permite a integração com diversos ambientes de desenvolvimento de código.

A utilização desta ferramenta já é uma prática na empresa, estando integrada com a ferramenta Jenkins, apresentada no âmbito deste projeto, e executa testes de uma forma automática. A utilização desta ferramenta encontra-se descrita na Secção 6.1.

3.3 Sumário

Neste capítulo foram apresentadas e analisadas diversas ferramentas que contam com a funcionalidade de criação de formulários existentes no mercado, percebendo assim as diferenças entre elas. Foram também apresentadas métodos e tecnologias estudados para a análise e desenvolvimento deste projeto.

No capítulo seguinte é demonstrada uma análise de valor à oportunidade assim como a definição da abordagem para a solução dos problemas identificados.

Capítulo 4

Análise

Os clientes estão constantemente à procura de produtos novos, melhorados continuamente e que acrescentem valor [46] e cabe às empresas lutar pela diferenciação e inovação dos seus produtos. Este facto representa custos para uma empresa ou organização e, por essa razão, existem processos de inovação que tentam garantir que as empresas ou organizações não alocam os seus recursos em ideias desnecessárias. Neste capítulo será desenvolvida a análise do projeto através da implementação de um teste de usabilidade e definição de requisitos da nova solução. É também feita uma análise de valor com o objetivo de validar a existência do mesmo e definindo os moldes para o desenvolvimento da solução.

4.1 Engenharia de Requisitos

No âmbito do projeto, o primeiro passo consiste na definição de requisitos do novo editor. Para isso, é necessário o estudo do editor EasyForms, de modo a perceber os seus pontos fortes e os seus defeitos e listar os requisitos atualmente presentes assim como os requisitos a acrescentar com base nas falhas identificadas.

4.1.1 Análise de Usabilidade

Na Secção 3.2.1 é apresentado o conceito de teste de usabilidade como uma das maneiras mais eficazes de avaliar a usabilidade de um sistema. De modo a avaliar o editor EasyForms, foi definido um teste de usabilidade, definida a população representativa dos utilizadores do editor, assim como recolhidos e analisados os resultados do teste. Desta forma é possível identificar os problemas no editor EasyForms, cumprindo assim um dos objetivos propostos na Secção 1.4.

Definição do Teste de Usabilidade

Com o objetivo de avaliar a usabilidade, foi desenvolvido o teste apresentado no Apêndice A. Este teste pretende identificar os pontos fortes e pontos fracos do editor de formulários do E-goi, o EasyForms. O teste consiste numa lista de tarefas reais e num questionário associado às tarefas realizadas. Através dos resultados deste teste é possível avaliar a usabilidade do editor a um nível geral assim como perceber quais os pontos onde este sucede ou onde falha.

O teste é realizado de forma individual, num ambiente isento de distrações e sempre com a presença de um acompanhante que apresenta o teste e esclarece o participante sobre

os moldes do mesmo. O ambiente no produto E-goi e as configurações da conta E-goi necessárias para a realização do teste são previamente prontas para que o participante apenas se foque na resolução das tarefas.

As tarefas propostas abrangem a funcionalidade de criação e gestão de formulários de forma global dando sempre alguma liberdade ao participante para testar opções que não constam do teste. O participante é encorajado a partilhar as suas dúvidas e linhas de pensamento sendo que o acompanhante não pode responder às mesmas, apontando-as e anexando aos resultados do teste.

Depois da conclusão das tarefas, os participantes são convidados a responder a um questionário de avaliação do editor. Este questionário usa a escala de Likert Forced Choice [47] que consiste numa escala 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 de existir um número par de opções limita a escolha do participante, impossibilitando-o de escolher um ponto médio. Desta forma, o participante tem sempre de tomar uma decisão mais positiva ou mais negativa. As questões foram baseadas na System Usability Scale, um questionário criado por John Brooke [48] que tem como objetivo medir a usabilidade de um sistema, assim como nas dez heurísticas de Nielsen, um conjunto de princípios criado por Jakob Nielsen [28] para aumentar a usabilidade de um sistema.

Caracterização da População

A caracterização da população para um teste de usabilidade é um passo importante, como é possível perceber na Secção 3.2.1. No âmbito desta dissertação, por se tratar de um produto a melhorar e não ser algo completamente novo, é importante que os participantes do teste tenham algum conhecimento sobre o sistema para que possam opinar construtivamente sobre funcionalidades que possam ser mais ocultas ou demasiado avançadas para o novo utilizador. É importante, no entanto, perceber a opinião de utilizadores que não tenham experiência de modo a analisar o comportamento de pessoas inexperientes a utilizar o sistema.

Para fins estatísticos e aplicação de fórmulas matemáticas, o tamanho mínimo da população é de 30 elementos. Sendo a diversidade um fator importante no âmbito deste teste, a população foi dividida em duas partes iguais tendo sido selecionadas 15 pessoas familiarizadas com o sistema (internas à empresa) assim como 15 pessoas sem qualquer experiência com o mesmo (externas à empresa). As idades dos participantes variam entre 19 até 48 sendo que 17 dos participantes são do género masculino e 13 pessoas são do género feminino.

Resultados do Teste de Usabilidade

Depois de executado o teste de usabilidade, é necessário interpretar e avaliar os resultados. No gráfico seguinte, são apresentados todos os resultados para o questionário de usabilidade.

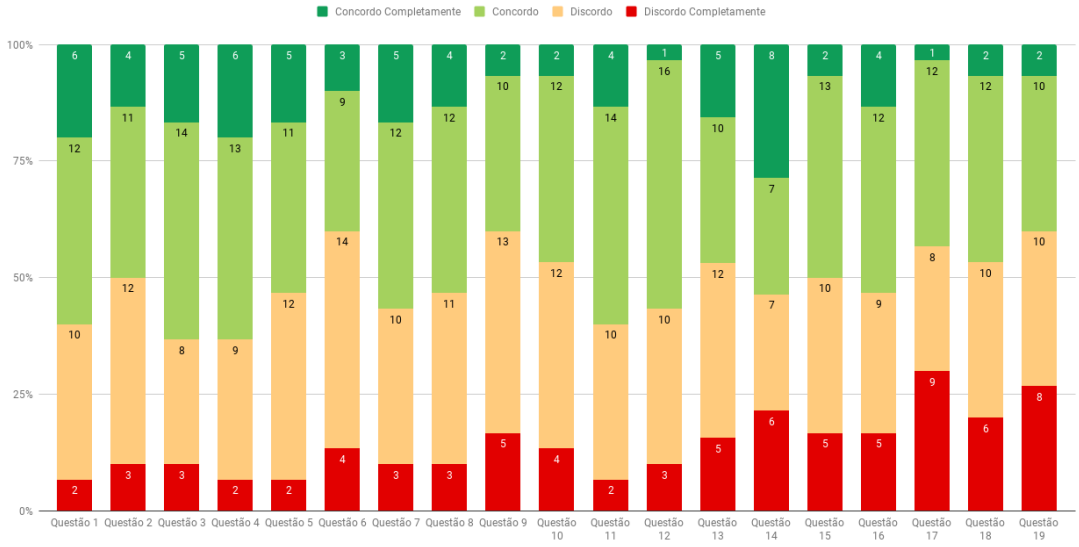

Figura 4.1: Resultados do Teste de Usabilidade ao Editor EasyForms

1. "Apesar de possuir bastante funcionalidade, o sistema é lento e pouco apelativo."
2. "O sistema tem bastantes opções."
3. "funciona bem apesar de ser um bocado lento"
4. "consegui construir um formulário com muita personalização"
5. "O editor apresenta bastantes falhas ao nível de consistência. É necessário sair do editor em si para fazer algumas funções, as diversas opções estão demasiado dispersas."

Figura 4.2: Comentários do Teste de Usabilidade ao Editor EasyForms

No gráfico da figura 4.1 estão representadas as dezanove perguntas do questionário de usabilidade através das colunas e as respostas dos trinta participantes do teste a cada questão através das cores. Na figura 4.2 são apresentados os comentários adicionais feitos na última pergunta do questionário. Para facilitar a análise, as questões podem ser agrupadas por tópicos sendo as questões 1 e 3 relacionadas com a importância da funcionalidade, as questões 2, 4, 7, 9, 10 e 15 relacionadas com a complexidade do editor, as questões 5, 6, 8, 11 e 12 relacionadas com a consistência do editor, as questões 18 e 19 e, por fim, as questões 13, 14, 16 e 17 relacionadas com a recuperação de erros feitos pelo utilizador. Analisando o gráfico da figura 4.1 podemos tirar as seguintes conclusões relativas ao estado do editor de formulários EasyForms.

Relativamente à importância da funcionalidade, as respostas revelam que a funcionalidade é importante para a população uma vez que a maioria concorda que usaria o sistema frequentemente assim como que o sistema é fácil de utilizar, coincidindo com a análise de valor desenvolvida na Secção 4.2.3. Esta informação valida a necessidade de desenvolver um novo editor de formulários.

Relativamente à complexidade do editor, de acordo com as respostas à questão 2, constata-se que o sistema possui uma complexidade alta, uma vez que metade dos participantes respondeu afirmativamente à questão. É importante perceber o nível de complexidade de um sistema uma vez que este pode afastar os utilizadores reais do mesmo, fazendo-os

procurar outras alternativas menos complexas que ofereçam a mesma funcionalidade. Nas respostas às questões 4 e 7, percebe-se que a maioria dos participantes considera que não precisaria de suporte técnico assim como que a maioria das pessoas aprenderia a utilizar o sistema facilmente, apesar de cerca de 40% dos participantes assumir o contrário. De acordo com as respostas à questão 9, percebemos a maioria dos participantes não se sentiu confiante a utilizar o sistema, o que revela alguma inconsistência no mesmo. Na questão 10 é possível perceber que os utilizadores não necessitaram de aprender novos conceitos ou adquirir novas informações para utilizar o sistema uma vez que a grande maioria das repostas são negativas. As respostas da questão 15 são igualmente distribuídas uma vez que metade dos participantes afirma que foi fácil realizar tarefas no sistema enquanto que a outra metade considera o contrário. As respostas a estas questões provam que o sistema, apesar da sua funcionalidade e valor para o utilizador final, é complexo em termos de usabilidade e não dá a confiança necessária aos seus utilizadores. Os resultados destas questões corroboram o problema identificado na Secção 1.3 relativo à complexidade para os utilizadores.

Relativamente à consistência do editor e das suas funcionalidades, as respostas da questão 5 permitem entender que, quando questionados relativamente à integração das diversas funcionalidades do sistema, a maioria da população considera que todas as funcionalidades do sistema estão bem integradas apesar da margem de aprovação ser reduzida. Relativamente às resposta da questão 6, é possível perceber que a maioria dos participantes concorda que o sistema é inconsistente. Relativamente às respostas da questão 8, a maioria dos participantes afirma que o sistema não é confuso apesar de cerca de 45% da população achar o contrário. Relativamente às questões 11 e 12, os participantes revelaram que os utilizadores se sentiram em controlo sobre o sistema assim como o seu estado durante todo o teste. Aliado às respostas do questionário, os participantes do teste teceram comentários que enfatizam as conclusões tais como o comentário número 3 e 5. Apesar de ser uma funcionalidade do produto E-goi, estas respostas demonstram alguma inconsistência do editor e, com isso, a necessidade de fazer alterações ao mesmo.

Relativamente ao design e interação com o editor, as respostas à questão 18 demonstram que o design gráfico da aplicação é pobre ou confuso, segundo os participantes do teste. Relativamente à questão 19, é possível perceber que os participantes consideram o sistema lento e pouco responsivo. Aliado a estas respostas, o comentário 1 e 3 são bastante representativos da opinião da população do teste demonstrando uma interface pobre e pouco apelativa. A interface gráfica é a ligação do sistema com o utilizador e, por isso mesmo, é essencial que seja bem desenvolvida para que o utilizador obtenha a melhor experiência possível.

Relativamente aos erros feitos pelos utilizadores e à sua recuperação, a questão 13 revela que o sistema não é eficiente a prever erros do utilizador uma vez que metade dos participantes responderam negativamente à questão. Relativamente à questão 14, as respostas demonstram que os participantes sentem que não necessitaram de memorizar demasiada informação para conseguir utilizar o sistema convenientemente. As respostas à questão 16 demonstram que o sistema pode não estar devidamente documentado, na opinião da população que efetuou o teste, uma vez que metade da população tem uma opinião diferente da outra metade. Relativamente à questão 17, é possível perceber que o sistema não é eficiente a oferecer *feedback* ao utilizador quando este comete algum erro. Erros feitos pelo utilizador, não sendo corrigidos, podem ser um motivo que leva o mesmo a abandonar um sistema ou a ter uma má experiência.

De forma global, através do teste de usabilidade, foi possível perceber que o sistema é útil

para os seus utilizadores e possui as funcionalidades que estes necessitam (questões 1 e 3). No entanto, estas funcionalidades não possuem uma acessibilidade adequada ou não são suficientemente consistentes (questões 2, 6, 8 e 16), quer na sua apresentação no editor (questões 18 e 19), quer na funcionalidade em si (questões 15 e 5). De modo a resolver os problemas identificados, é necessário fazer o levantamento dos requisitos funcionais e não funcionais do sistema para que possa ser desenvolvido um novo sistema.

4.1.2 Levantamento de Requisitos Funcionais

Depois de identificar os problemas do editor de formulários EasyForms, é possível definir os requisitos que a nova solução terá de cumprir de modo colmatar os mesmos. De modo a fazer o levantamento de requisitos funcionais, foram desenvolvidas *user stories* cuja descrição se encontra na Secção 3.2.1. De modo a simplificar a perceção dos cenários de sucesso das *user stories*, recorre-se à linguagem UML, utilizando um diagrama de sequência que representa a interação entre o utilizador e o sistema.

Criaram-se os seguintes requisitos funcionais para a nova solução:

User Story 1: Criar Formulário

Como utilizador do editor de formulários, eu quero criar um formulário para ser listado na minha lista de formulários.

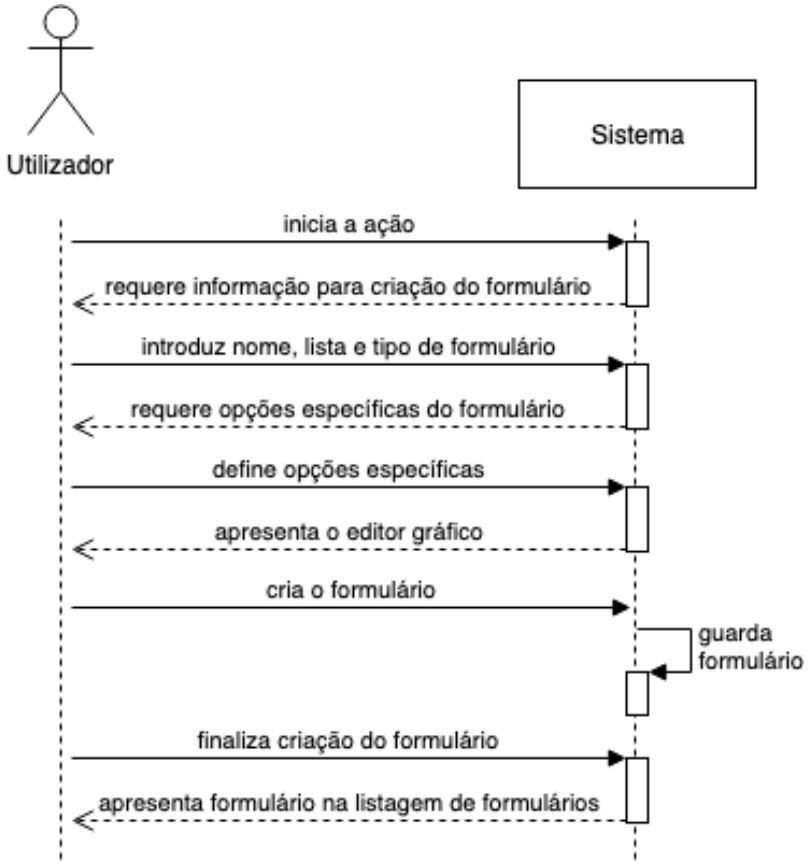

Figura 4.3: Cenário de Sucesso do US1

Para que este caso de sucesso se suceda, o utilizador necessita de fornecer um nome válido para o formulário, definir a lista à qual o formulário está associado e o tipo de formulário que deseja criar. O sistema guarda automaticamente o formulário de uma forma periódica, garantindo assim que o utilizador não perde todo o trabalho desenvolvido caso aconteça algo de anormal.

User Story 2: Editar Formulário

Como utilizador do editor de formulários, eu quero editar o meu formulário num ambiente gráfico para que este possa ser posteriormente enviado ou exportado.

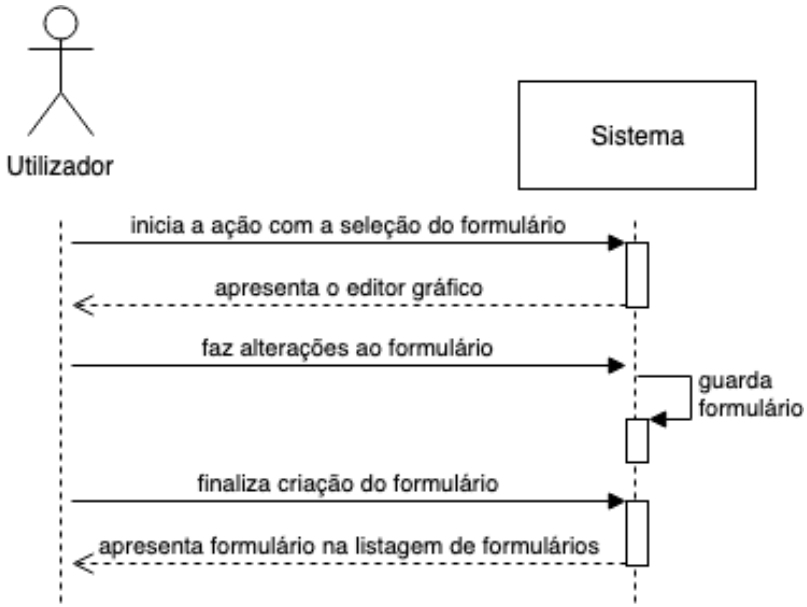

Figura 4.4: Cenário de Sucesso do US2

Para que este caso de sucesso se suceda, o utilizador necessita de indicar a seleção de um formulário existente para que o editor o possa apresentar. O utilizador pode modificar,acrescentar ou eliminar campos do formulário ou opções globais do mesmo. À semelhança da *User Story 1*, o sistema guarda automaticamente o formulário de uma forma periódica.

User Story 3: Acrescentar um Campo Existente

Como utilizador do editor de formulários, eu quero adicionar campos ao meu formulário que estejam mapeados com os campos presentes na lista para que as respostas ao formulário sejam guardadas.

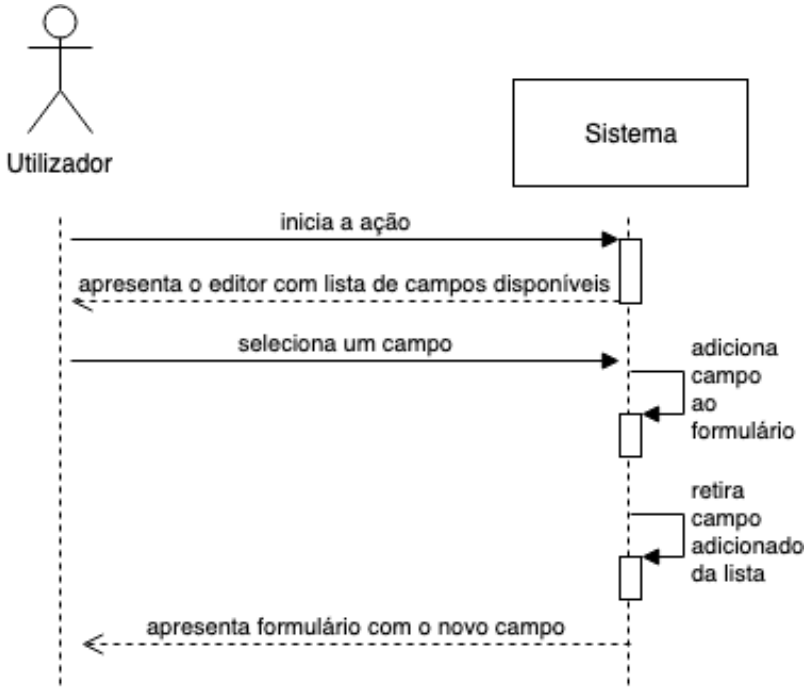

Figura 4.5: Cenário de Sucesso do US3

Para que este caso de sucesso se suceda, o editor apresenta uma listagem dos campos presentes na lista de contactos selecionada na criação do formulário adicionando ao formulário um campo do tipo de campo selecionado.

User Story 4: Acrecentar Campos Extra

Como utilizador do editor de formulários, eu quero adicionar campos extra ao formulário, criando um campo novo na lista e mapeando-o com o campo do formulário para que este possa armazenar as respostas dos visitantes.

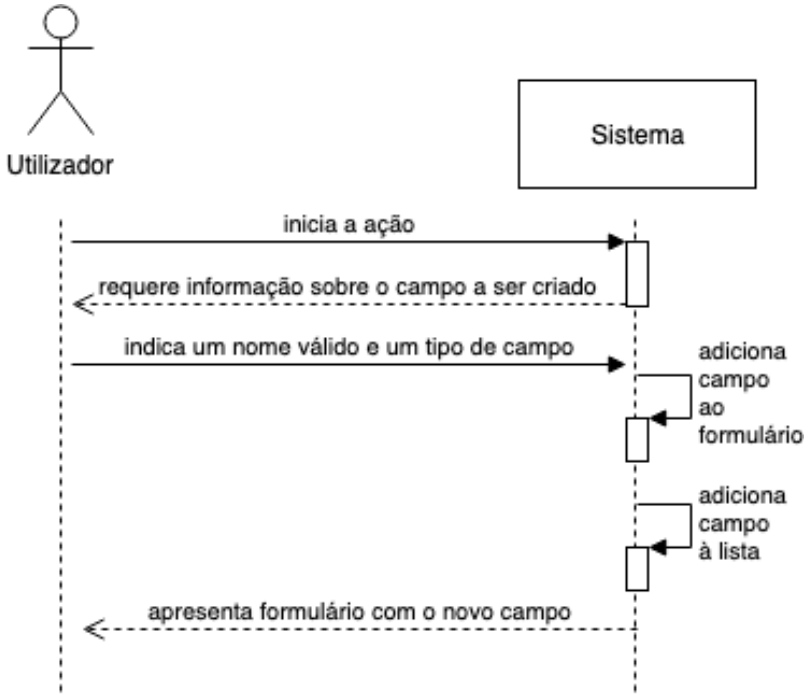

Figura 4.6: Cenário de Sucesso do US4

Para que este caso de sucesso se suceda, o utilizador indica o nome e o tipo de campo a adicionar, o sistema adiciona o campo ao formulário assim como à lista de contactos.

User Story 5: Definir Estilos Globais

Como utilizador do editor de formulários, eu quero definir e alterar estilos do meu formulário de uma forma global para que a alteração seja rápida.

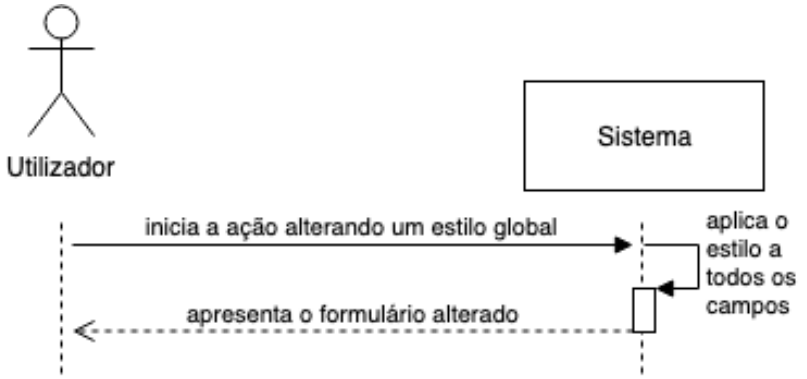

Figura 4.7: Cenário de Sucesso do US5

Para que este caso de sucesso se suceda, o utilizador altera um estilo global que pode consistir numa alteração de cor, posicionamento do campos do formulário, entre outras. O editor efetua a alteração e apresenta o formulário com a mesma aplicada.

User Story 6: Definir Opções Específicas por Campo

Como utilizador do editor de formulários, eu quero definir opções específicas para um tipo de campo do meu formulário para que estas sejam refletidas no formulário final.

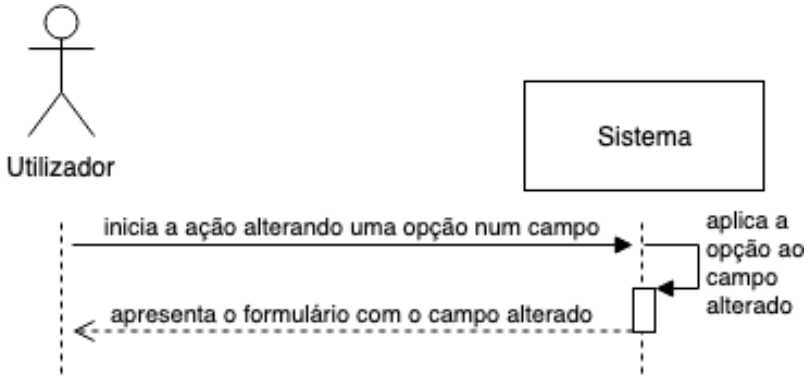

Figura 4.8: Cenário de Sucesso do US6

Para que este caso de sucesso se suceda, o utilizador altera uma opção específica de um campo que pode consistir na alteração do texto de ajuda, inclusão de um ícone, posição do campo, entre outros. O editor efetua a alteração e apresenta o formulário com as alterações feitas no campo editado.

User Story 7: Guardar Alterações do Formulário

Como utilizador do editor de formulários, eu quero guardar as alterações que o fiz ao formulário, quer sejam alterações gráficas como alterações de opções para que estas se reflitam no formulário final.

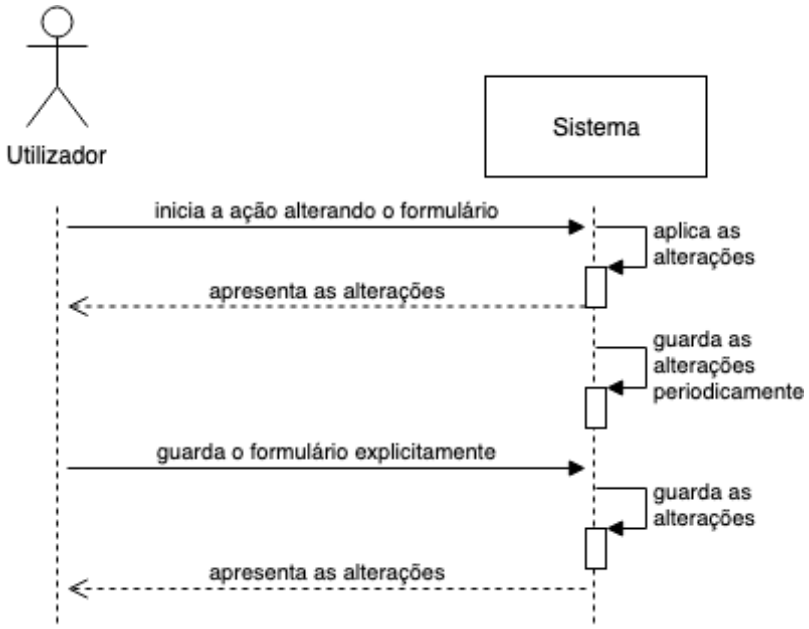

Figura 4.9: Cenário de Sucesso do US7

Para que este caso de sucesso se suceda, o utilizador pode explicitamente comandar o sistema para guardar as alterações feitas. No entanto, para que o trabalho do utilizador seja sempre salvaguardardo, o sistema guarda as alterações de uma forma periódica.

User Story 8: Exportar Formulário

Como utilizador do editor de formulários, eu quero conseguir exportar o meu formulário para conseguir colocá-lo noutros ambientes, como numa página *web*, por exemplo.

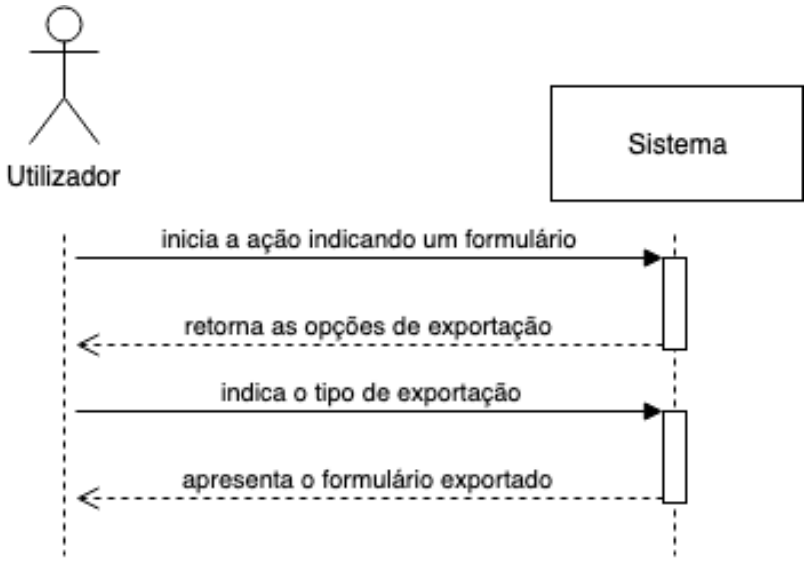

Figura 4.10: Cenário de Sucesso do US8

Para que este caso de sucesso se suceda, o utilizador escolhe o formulário para exportar, decide o modo de exportação e recebe o formulário exportado no formato escolhido.

Levantamento de Requisitos Não Funcionais

Aplicando os princípios apresentados na Secção 3.2.1 relativos ao levantamento de requisitos não funcionais ao âmbito deste projeto, foram identificados os seguintes requisitos não funcionais:

Funcionalidade:

- O sistema deve comunicar com o produto E-goi através de uma API;
- Nenhuma ação deve ser destrutiva, quer dentro do editor quer relativamente aos formulários em si;
- O sistema deve manter as funcionalidades base de todos os *widgets* existentes no editor de formulários EasyForms;

Usabilidade:

- As opções do formulário devem ser contextuais ao tipo de formulário escolhido;
- O editor deve adaptar-se para os diferentes tipos de formulários;

Confiabilidade:

- O sistema deve fazer gravações automáticas periódicas;
- O sistema deve informar o utilizador de qualquer mudança de estado quer do próprio sistema quer do formulário;

Desempenho:

O tempo de espera para a criação de um formulário ser aceitável para o utilizador;

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:

O sistema deve ser desenvolvido usando a *framework* Angular [49], a linguagem de programação PHP e o sistema de gestão de base de dados MariaDB;

A implementação gráfica dos campos do formulário deve ser implementada com a *framework* Bootstrap;

4.2 Análise de Valor

Para desenvolver um novo produto ou funcionalidade é necessária uma análise interna, relativa às condições para o desenvolvimento e custos *versus* benefícios assim como externa relativa ao estado do mercado ou à necessidade do produto ou funcionalidade. Nesta secção é apresentado todo o processo de inovação desenvolvido para a escolha da abordagem a implementar.

4.2.1 Processo de Inovação

O conceito inovação é bastante abrangente uma vez que, no ambiente empresarial, pode traduzir a criação de um produto completamente novo assim como a atualização ou adição de funcionalidades a um produto existente ou até mesmo uma mudança no processo de fabrico do produto [50]. No entanto, em todos os casos apresentados, a inovação tem como principal objetivo acrescentar valor, quer seja à organização, quer seja ao cliente [51].

O processo de inovação, apresentado na Secção pode ser relativo a um produto ou à própria organização e pode ser dividido em três partes sequenciais:

- Fuzzy Frond End;
- New Product Development;
- Comercialização.

Este processo tenta definir o ciclo de vida de uma ideia desde a sua criação até à comercialização de um produto, passando pela sua implementação. O processo é sequencial, ou seja, espera-se que apenas cheguem à fase de comercialização ideias e produtos cujo valor exista efetivamente, provados no resto do processo. Cada passo do processo tem objetivos específicos e dependentes dos passos anteriores.

O termo Fuzzy Front End foi cunhado por Reinertsen e Smith em 1991 [52]. Consiste no primeiro passo do processo de inovação e o seu objetivo é a criação de um conceito baseado numa oportunidade. Para atingir esse objetivo é necessário perceber e identificar as oportunidades existentes, validar as mesmas, gerar ideias que acrescentam valor baseadas

nas oportunidades identificadas, selecionar ideias e, por fim, definir o conceito que será implementado na fase seguinte.

O conceito de New Product Development, apresentado na Secção 3.2.1 consiste na implementação dos conceitos criados no passo anterior do processo de inovação. O objetivo deste passo é transformar o conceito num produto, fazer análises de mercado, perceber como chegar aos clientes assim como testar o produto e fazer possíveis alterações.

Por fim segue-se a Comercialização cujo objetivo é vender o produto criado (ou alterado). Neste passo, as ideias já foram transformadas em produtos que já foram testados e aprovados e, por isso, segue-se a fase de produção e comercialização do produto.

4.2.2 New Concept Development

Aplicando o modelo de Peter A. Koen, New Concept Development, é possível traduzir uma oportunidade num conceito, seguindo todos elementos chave do modelo, percebendo assim se a oportunidade traz valor à empresa ou organização de uma forma sistemática e bem definida.

Identificação da Oportunidade

As oportunidades para inovar podem surgir das mais variadas formas, por exemplo, a inexistência de um produto no mercado pode traduzir uma oportunidade assim como a implementação de novas funcionalidades em produtos existentes . As oportunidades podem estar relacionadas com fatores internos ou externos.

Os problemas identificados na secção 1.3 representam uma oportunidade para melhorar o produto E-goi, nomeadamente com a implementação de um novo editor de formulários, gerando mais utilização da funcionalidade ou mesmo do produto.

Uma oportunidade por si só pode não representar qualquer acréscimo de valor à empresa ou organização e, por isso, torna-se necessário efetuar uma análise e tirar conclusões quanto à mesma.

Análise da Oportunidade

O objetivo da análise da oportunidade consiste em identificar os pontos fortes e os pontos fracos da mesma e tirar conclusões relativamente ao valor que esta acrescenta ao produto assim como o custo a pagar para o seu desenvolvimento. Existem várias ferramentas que auxiliam esta análise de uma oportunidade, nomeadamente a análise SWOT.

No gráfico da análise SWOT figuram as forças, fraquezas, oportunidades e ameaças de uma ideia. As forças identificam os pontos fortes da ideia, os motivos pelos quais a ideia é adiciona valor. As fraquezas identificam os pontos fracos como o custo de produção, dificuldades de implementação, entre outras. As oportunidades identificam os fatores externos que podem contribuir para o sucesso da ideia. Por último, as ameaças identificam fatores externos que podem fazer com que a ideia não suceda, como por exemplo, concorrência, leis externas, constrangimentos do mercado.

Figura 4.11: Análise SWOT

Relativamente à oportunidade de remodelar o editor de formulários do produto E-goi, gerou-se a análise SWOT apresentada na figura 4.11. Esta análise tem como objetivo perceber se a oportunidade identificada traz valor à empresa assim como se o esforço necessário para a concretizar é recompensado pelos ganhos que a concretização da mesma pode gerar.

Relativamente às forças foram identificados cinco pontos chave relativamente ao desenvolvimento do novo editor de formulários. O facto de pertencer ao produto E-goi confere ao novo editor algumas vantagens relativamente aos concorrentes, como por exemplo, o envio do formulário para listas de contactos predefinidas. A maioria dos editores de formulários disponíveis atualmente não se encontram integrados num produto de *marketing* não permitindo a criação, gestão e importação de listas de contactos. Desta forma, os utilizadores ficam impossibilitados de enviar formulários (tendo de os exportar e posteriormente colocar numa página web, por exemplo) ou necessitam de introduzir todos os contactos pretendidos para cada formulário. Além disso, os resultados dos formulários não são automaticamente adicionados às listas de contactos (para formulários de inscrição), dando assim vantagem ao produto E-goi.

Outra vantagem relativamente à integração no E-goi é a utilização de outras funcionalidades sendo exemplo disso a funcionalidade Track & Engage. Esta funcionalidade tem a capacidade de reconhecer visitantes de um formulário através de *cookies* podendo fazer um preenchimento automático dos campos de um formulário caso o contacto já exista na lista de contactos E-goi. Esta funcionalidade traduz uma vantagem perante a maioria dos concorrentes.

A possibilidade de desenvolver um novo editor de formulários vai permitir simplificar o editor EasyForms que gera bastantes dúvidas aos utilizadores, comprovadas através de *tickets*.

Através de estudos de usabilidade e testes será possível implementar um sistema mais amigável para o utilizador mantendo as funcionalidades existentes e acrescentando novas que possam fazer sentido ou ser uma necessidade para o utilizador.

O editor de formulários EasyForms encontra-se implementado com linguagens desatualizadas e sem aplicação de boas práticas de engenharia informática. A manutenção deste componente é cara para a empresa uma vez que, por não estar desenvolvido de uma forma modular assim como utilizar linguagens pouco usadas atualmente, traz dificuldades aos colaboradores. A possibilidade de implementar o novo editor de formulários com linguagens mais atuais, que permitam a implementação de novas funcionalidades de uma forma ágil e assim como a divisão do editor em módulos que permitam facilmente detetar falhas e colmatar as mesmas traduz valor acrescentado à empresa.

Relativamente às fraquezas, foram identificadas duas grandes fraquezas relacionadas com a oportunidade identificada. O desenvolvimento de um novo editor traduz um custo considerável para a empresa uma vez que é necessário efetuar análises ao editor EasyForms, desenvolver uma solução que colmate todos os problemas desenvolvidos, implementar e testar a mesma. O custo pode não compensar o valor que o novo editor poderá trazer e, por isso, é algo a ter em conta na tomada de decisão e na definição do conceito.

A outra fraqueza identificada foi a dificuldade de integração do novo editor no produto. Qualquer integração de uma nova funcionalidade com um sistema existente pode traduzir um risco, especialmente se a funcionalidade for desenvolvida de uma maneira totalmente independente. É importante perceber se o esforço para integrar o novo editor é compensado pelo valor que este poderá trazer ou se é mais sensato remodelar o editor existente, uma vez que já se encontra perfeitamente integrado com todo o sistema.

As oportunidades e as ameaças consistem em factores externos à empresa, ou seja, não controlados pela mesma. No entanto, é importante perceber as oportunidades e ameaças existentes de modo a existir uma resposta rápida e eficaz por parte da empresa aos fatores externos. Relativamente às oportunidades identificadas, destacam-se duas. Com o desenvolvimento de um novo editor de formulários, espera-se que a utilização desta funcionalidade aumente consideravelmente por se tratar de um editor mais simples e mais funcional. Outro fenómeno que pode acontecer é o aumento da utilização do E-goi, atraindo utilizadores que não utilizavam a mesma devido ao editor de formulários. Estes aumentos de utilização não dependem da organização mas são objetvos da mesma e as ações tomadas visam alcançar esses objetivos.

Relativamente às ameaças identificadas, existe uma relação direta com a concorrência. O aparecimento de novos concorrentes que tenham editores e produtos mais completos podem ser uma ameaça ao editor de formulários assim como concorrentes com sistemas específicos para a edição de formulários. É importante perceber as ameaças relacionadas com a concorrência de modo a melhorar continuamente o produto para que os utilizadores do E-goi não sintam a necessidade de abandonar a mesma e optar por outras soluções.

Por último, é importante também manter uma atenção constante na vertente legislativa, nomeadamente em relação à proteção de dados. Alterações legislativas podem tornar ilegais algumas práticas ou funcionalidades do produto e, por esse motivo, é fulcral que exista uma atenção continua de modo a conseguir alterar essas práticas ou funcionalidades.

Em suma, através da análise SWOT torna-se perceptível que esta oportunidade é válida e pode trazer bastante valor à empresa, quer a nível externo com o aumento de utilização do

produto quer a nível interno com a facilidade de manutenção e escalabilidade. Para além disso, a concretização da oportunidade acrescenta também valor ao cliente, uma vez que simplifica a criação e edição de formulários sem comprometer funcionalidades já existentes no E-goi e no próprio editor. Estas vantagens compensam os custos que a concretização da ideia possa gerar à empresa e, por isso, segue-se a geração das ideias que poderão aproveitar a oportunidade identificada.

Geração de Ideias

Demonstrada a validade e as vantagens da oportunidade através da sua análise, foi necessário gerar soluções que colmatem os problemas identificados, que simplifiquem os processos internos e que aumentem a competitividade do produto no mercado. Efetuou-se um trabalho de pesquisa de possíveis soluções que cumprissem todos os pontos identificados na oportunidade. Após discussão interna surgiram duas possíveis abordagens que abarcam todos os pontos identificados:

- A reconstrução do editor de formulários EasyForms;
- A implementação de um novo editor de formulários utilizando funcionalidades já existentes no E-goi, nomeadamente o pacote "builder";
- A implementação de um novo editor de formulários independente do sistema E-goi, posteriormente integrado.

O pacote "builder"foi desenvolvido internamente e tem como objetivo normalizar a criação de novos editores em Angular. Consiste num pacote que contém lógica de renderização assim como a definição dos *widgets* utilizados nos atual editor de *e-mail* do produto E-goi. A utilização deste pacote traz alguns constrangimentos relativamente ao desenvolvimento da solução mas permite que a integração no sistema E-goi seja facilitada.

Apesar de serem diferentes, todas as ideias apresentadas resolvem os problemas identificados. No entanto, as qualidades e defeitos das soluções diferem e, por isso, torna-se necessário fazer uma análise segundo alguns critérios chave e, posteriormente, decidir qual a solução a seguir.

Seleção de Ideias

Depois de identificadas as ideias que colmatam os problemas existentes e aproveitam oportunidades identificadas, é necessário perceber especificamente as suas vantagens e desvantagens para que possa existir uma tomada de decisão. Os recursos alocados para a implementação de cada ideia podem variar e, por esse motivo, é necessário perceber se o retorno da implementação de cada ideia compensa o esforço realizado.

Por ser uma decisão crucial no processo de inovação, as empresas e organizações utilizam métodos de apoio à tomada de decisão. Um dos exemplos desses métodos é o AHP, que utiliza uma abordagem matemática e racional [8]. Como abordado no Capítulo 3, o método divide-se em três etapas sequenciais.

A primeira destas etapas consiste na **Divisão Hierárquica**. Aplicando a hierarquia do método ao âmbito desta dissertação, é possível definir o objetivo como do desenvolvimento de um editor de formulários que colmate os problemas identificados.

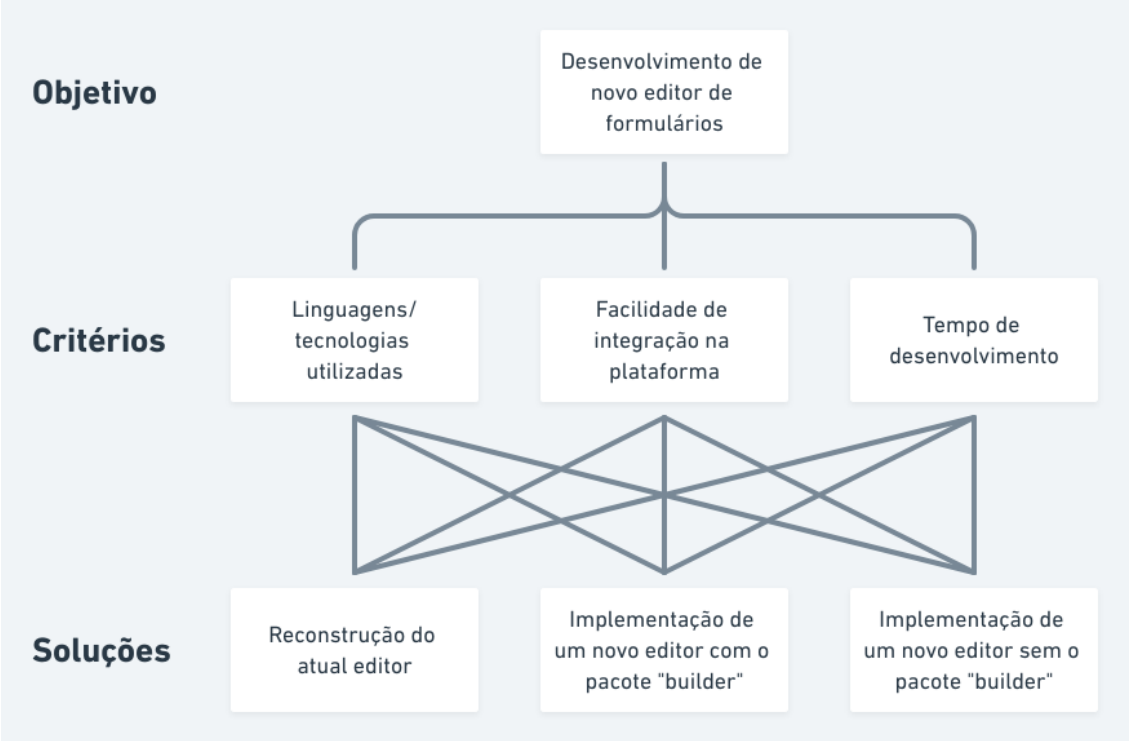

Figura 4.12: Estrutura hierárquica do Analytic Hierarchy Process

Para atingir este objetivo, e como referido anteriormente, os três critérios mais importantes consistem na linguagem e tecnologias utilizadas, na facilidade com a integração no E-goi e o tempo de desenvolvimento da solução.

As opções identificadas na Secção 4.2.2 são as opções que figuram neste processo por atingirem o objetivo definido sendo elas a reconstrução do editor de formulários EasyForms utilizando as tecnologias presentes no mesmo, desenvolver um editor de formulários novo sem utilizar o pacote "builder"ou desenvolver um editor de formulários novo utilizando o pacote "builder".

A segunda etapa consiste do método AHP consiste na **Definição de Prioridades**. Aplicando a etapa de definição de prioridades do AHP, são definidas as prioridades e utilizada a escala de Saaty [23].

Sendo:

- A: Tecnologias e linguagens utilizadas no editor;
- B: Facilidade de integração no produto E-goi;
- C: Tempo de desenvolvimento do editor.

	A	B	C	Prioridade Relativa
A	1	0.33	4	0.28
B	3	1	5	0.62
C	0.25	0.20	1	0.09

Tabela 4.1: Estrutura hierárquica de decisão

Interpretando a tabela é possível perceber que a integração é um pouco mais importante que a tecnologias/linguagem utilizadas e consideravelmente mais importante que o tempo de desenvolvimento. Também é perceptível que as tecnologias são consideravelmente mais importantes que o tempo de desenvolvimento.

Através da prioridade relativa, calculada através da média das linhas da matriz normalizada, conseguimos perceber que o critério mais importante se trata da integração no produto E-goi, seguido do critério de tecnologias e linguagens utilizadas e, por último, o tempo de desenvolvimento do editor.

Depois de priorizar os critérios, é necessário priorizar as soluções relativamente a cada critério e para isso foram utilizadas três tabelas semelhantes à tabela anterior.

Sendo:

- X: Fazer um novo editor de formulários sem utilizar o pacote "builder";
- Y: Fazer um novo editor de formulários utilizando o pacote "builder";
- Z: Reconstruir o editor EasyForms.

	X	Y	Z	Prioridade Relativa
X	1	1	9	0.47
Y	1	1	9	0.47
Z	0.11	0.11	1	0.05

Tabela 4.2: Priorização de Soluções relativamente ao critério Tecnologias/-Linguagem

	X	Y	Z	Prioridade Relativa
X	1	0.14	0.14	0.07
Y	7	1	2	0.36
Z	7	2	1	0.57

Tabela 4.3: Priorização de Soluções relativamente ao critério Integração

	X	Y	Z	Prioridade Relativa
X	1	0.14	7	0.23
Y	7	1	7	0.70
Z	0.14	0.14	1	0.07

Tabela 4.4: Priorização de Soluções relativamente ao critério Tempo de Desenvolvimento

Nas três tabelas é possível perceber a relação entre as diferentes soluções relativamente aos critérios definidos assim como as suas prioridades relativas. Sendo que relativamente ao critério de tecnologias e linguagens as hipóteses A e B são bastante mais convenientes que a hipótese C. No entanto, relativamente ao critério de integração, a hipótese C é a mais benéfica sendo bastante mais conveniente que a opção A e um pouco mais conveniente que

a opção B. Relativamente ao critério do tempo de desenvolvimento, a opção B é a opção mais benéfica relativamente às outras opções com um grau de diferença elevado.

Dispondo as prioridades relativas dos critérios e das soluções em matrizes próprias, obtemos as seguintes matrizes:

	A	B	C
X	0.47	0.07	0.23
Y	0.47	0.36	0.70
Z	0.05	0.57	0.07

Tabela 4.5: Matriz de prioridades relativas das soluções

A	0.28
B	0.62
C	0.09

Tabela 4.6: Matriz de prioridades relativas dos critérios

Multiplicando ambas as matrizes, é possível obter o peso global para cada solução relativamente aos critérios:

	Peso Global
A	0.20
B	0.43
C	0.37

Tabela 4.7: Matriz de Pesos Globais

Por fim, o último passo do método AHP consiste na garantia de **Consistência**. Utilizando a formula relativa à consistência do AHP:

$$\frac{(\lambda_{max} - n)}{(n-1)}$$

onde $n = 3$.

Fazendo o cálculo para a consistência, é obtida a seguinte matriz:

A	0.88
B	1.95
C	0.29

Como referido, λ_{max} é igual ao maior valor próprio da matriz. O calculo do valor próprio é igual à média dos valores obtidos na matriz dividido pelos valores da matriz de prioridades relativas dos critérios:

$$\lambda_{max} = \frac{0.88/0.28 + 1.95/0.62 + 0.29/0.09}{3} = 3.09$$

O aplicando a fórmula do índice de consistência, obtemos o seguinte resultado:

$$IC = \frac{(\lambda_{max} - n)}{(n-1)} = \frac{(3.09 - 3)}{(3-1)} = 0.04$$

Depois de calculado o índice de consistência, é possível calcular o rácio de consistência através da seguinte formula:

$$RC = \frac{IC}{IR}$$

onde IC corresponde ao Índice de Consistência e IR corresponde ao Índice Aleatório.

O Índice Aleatório é obtido através de uma tabela própria consoante o valor de n.

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IR	0	0	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

Tabela 4.8: Tabela de Índice Aleatório

Sendo n = 3, o cálculo do rácio de consistência é dada pela seguinte função:

$$RC = \frac{0.04}{0.58} = 0.07$$

Como RC < 0.1, consideram-se os dados consistentes e o resultado final do processo de AHP válido.

Conclusões

Uma vez que foi provada a consistência dos critérios, é possível considerar os resultados do processo AHP válidos. Os valores obtidos foram:

- Solução X: 0.20;
- Solução Y: 0.43;
- Solução Z: 0.37

Uma vez que a solução Y obteve maior resultado, a implementação de um novo editor de formulários utilizando o pacote "builder"pode ser considerada a melhor solução para o problema identificado.

Definição de Conceito

Depois de utilizar o AHP para decidir qual das três ideias seria implementada, é importante definir o conceito. O conceito para o novo editor de formulários do produto E-goi consiste num sistema *drag & drop* que permite criar um formulário arrastando *widgets* para uma área de edição. A interface tem de ser simples e intuitiva para o utilizador e resolver todos os problemas identificados anteriormente e detetados através dos estudos de usabilidade e testes com utilizadores.

4.2.3 Valor da Solução

Valor, Valor para o Cliente, Valor Percecionado

Para que se entenda a importância de um produto ou serviço para o cliente, é importante definir conceitos relacionados com o valor de um produto tais como valor, valor para o cliente e valor percecionado.

O valor de um produto consiste na relevância que este tem para o cliente e no que o cliente está disposto a pagar por ele. Uma vez que a maioria das receitas de uma organização advêm dos seus clientes, torna-se fulcral que as organizações se preocupem com o valor dos seus produtos ou serviços. Esta preocupação é cada vez mais relevante e traduz-se na mudança e continua atualização de produtos ou serviços assim como de todo o processo de desenvolvimento dos mesmos, sendo estas as duas principais maneiras de aumentar o valor de um produto ou serviço.

O valor para o cliente consiste no valor que o cliente dá ao produto ou serviço consoante a sua perceção pessoal. O mesmo produto ou serviço pode ter valores diferentes para clientes diferentes consoante a sua utilização e experiência do mesmo. Um utilizador que obtenha uma boa experiência com um produto ou serviço tende a querer voltar a repetir a experiência assim como tende a considerar que o valor desse produto ou serviço é elevado. Aplicando este conceito ao projeto, oferecendo um produto mais simples e intuitivo ao utilizador que não perde as funcionalidades do editor EasyForms, é expectável que o cliente sinta um aumento no valor do produto.

O valor percecionado consiste no resultado da diferença entre os benefícios e os sacrifícios para o cliente [53]. No âmbito to projeto, é possível considerar os benefícios da solução o aumento de métricas de resultados dos formulários, o aumento da usabilidade do editor e o aumento de funcionalidades do editor. Por outro lado é possível também considerar a aprendizagem de utilização do novo editor e o esforço para a criação de formulários como sacrificos para o cliente. O valor percecionado é variável de cliente para cliente pois os pesos dos benefícios e sacrifícios é medido de uma forma pessoal.

Proposta de Valor

Uma proposta de valor visa apresentar os motivos pelos quais um cliente deve adquirir um produto ou serviço e quais os benefícios que este irá acrescentar ao cliente. Deve ser explicito no que consiste o produto ou serviço, no valor obtido pelo cliente assim como no que é que o produto ou serviço simplificam a vida do cliente.

No âmbito do projeto, a proposta de valor consiste num editor de formulários integrado no produto E-goi que ofereça mais usabilidade aos seus utilizadores comparativamente ao editor EasyForms sem perder as funcionalidades existentes. Esta solução não só vai aumentar as funcionalidades do editor (como métricas e a criação de novos *widgets*) assim como a usabilidade com um novo editor permitindo assim ao utilizador criar formulários de uma forma mais rápida e adequada, simplificando todo o processo de criação.

4.2.4 Business Model Canvas

A utilização do Business Model Canvas, apresentado na Secção 3.2.1, permite perceber como é que os diversos aspetos relativos à venda de um produto ou serviço se interligam.

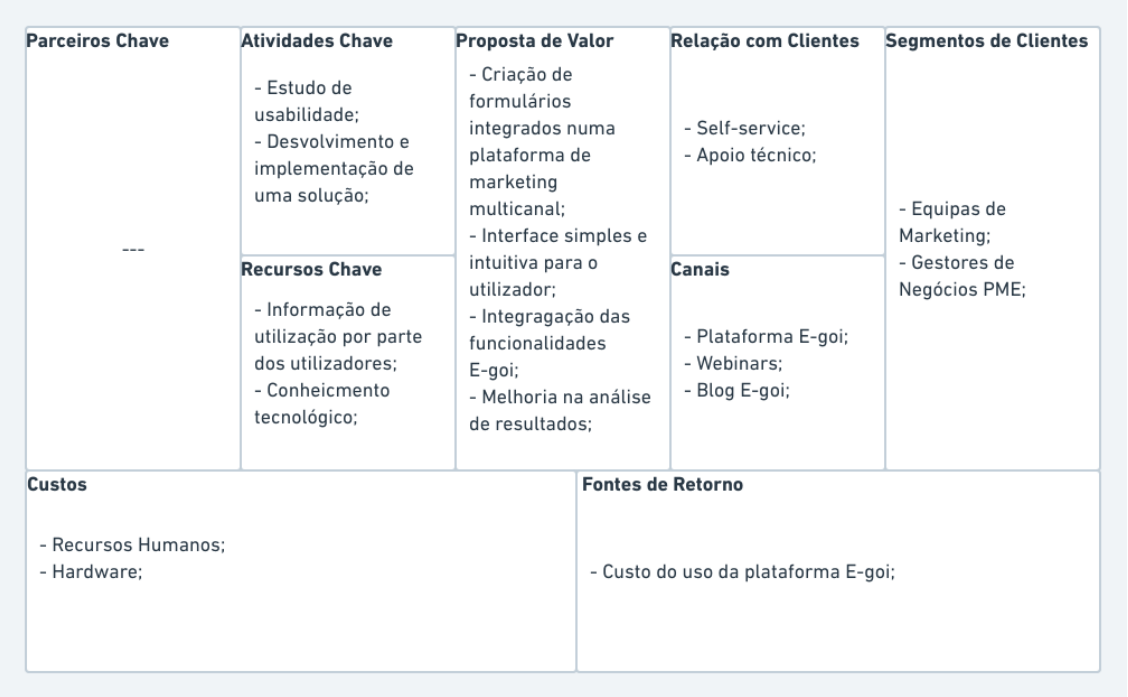

Figura 4.13: Business Model Canvas

Como se constata na imagem, cada bloco possui informação sobre uma parte do negócio:

Parceiros Chave:
: Este bloco tem como objetivo listar todos os parceiros externos à organização que são essenciais para a concretização da ideia num produto ou serviço. No âmbito deste projeto, uma vez que se trata de uma funcionalidade para o produto da empresa, não existem parceiros chave sendo a E-goi a manter exclusivamente o produto e consequentemente as suas funcionalidades;

Atividades Chave:
: Este bloco visa perceber as atividades que darão origem ao produto ou serviço e que são essenciais para atingir esse objetivo. No âmbito deste projeto, as atividades chave consistem em fazer estudos de usabilidade e testes com utilizadores de modo a perceber qual a melhor forma de implementar o editor assim como desenvolver uma arquitetura para o mesmo e implementar a mesma;

Recursos Chave:
: Este bloco tem como objetivo listar todos os recursos sem os quais é impossível concretizar a ideia num produto ou serviço. No âmbito deste projeto, os recursos necessários para implementar o novo editor de formulários são a informação de utilização por parte dos utilizadores assim como o conhecimento tecnológico necessário para efetivamente implementar o editor;

Proposta de Valor:
: Este bloco consiste na apresentação dos motivos pelos quais o cliente escolhe usar o produto ou serviço em detrimento de outros. No âmbito deste projeto, a proposta de valor consiste na funcionalidade de criar e editar formulários integrada num produto de *marketing* digital multicanal e com as suas funcionalidades garantindo que o editor apresenta uma interface simples e intuitiva para o utilizador e que permita uma fácil e útil análise de métricas;

Relação com os Clientes:
: Este bloco explicita como será a relação entre a organização e o cliente relativamente ao produto em questão. No âmbito deste projeto, espera-se que os clientes sejam o mais autónomos possíveis na utilização do editor garantindo assistência técnica sempre que necessária;

Canais:
: Este bloco define os canais de comunicação e venda do produto ou serviço. No âmbito deste projeto, o único canal de venda do produto é o E-goi, uma vez que o editor é uma funcionalidade do mesmo. Relativamente aos canais de comunicação, serão usados os canais de comunicação do produto E-goi, nomeadamente *webinars* (vídeos explicativos das funcionalidades do produto) e o *blog* E-goi, onde são colocadas as atualizações do produto.

Segmentos de Clientes:
: Este bloco explicita os tipos de clientes que se espera que comprem um produto ou serviço. No âmbito deste projeto, os clientes esperados para esta funcionalidade do E-goi são equipas de marketing de grandes organizações e gestores de marketing de negócios de pequenas e médias empresas. Estes são os dois segmentos de clientes do próprio produto E-goi.

Custos:
: Este bloco visa perceber os custos inerentes à implementação da ideia. No âmbito deste projeto, existem os custos relativos a recursos humanos assim como hardware, quer para desenvolvimento do editor quer para alojamento do mesmo.

Fontes de Retorno:
: Este bloco explicita todas as fontes de retorno de um produto ou serviço. No âmbito deste projeto, espera-se que todo o retorno consista no pagamento dos clientes para a utilização do produto. O editor poderá trazer fontes de retorno indiretas, uma vez que apenas acrescenta valor a um produto já existente.

4.2.5 Cadeia de Valor de Porter

Em 1985, Michael Porter apresenta o conceito de cadeia de valor como uma divisão das diversas atividades de uma organização e a maneira como estas interagem para gerar valor [54] a uma organização, representada na Figura 4.14. O objetivo desta ferramenta de gestão é ilustrar as diversas atividades levadas a cabo pela empresa, categorizando-as de modo a perceber a organização como partes pequenas que funcionam em conjunto ao invés de ver a organização como um todo. Aplica-se o conceito "Dividir para conquistar", desconstruindo a organização e as suas atividades em partes mais pequenas e mais facilmente compreendidas e analisadas.

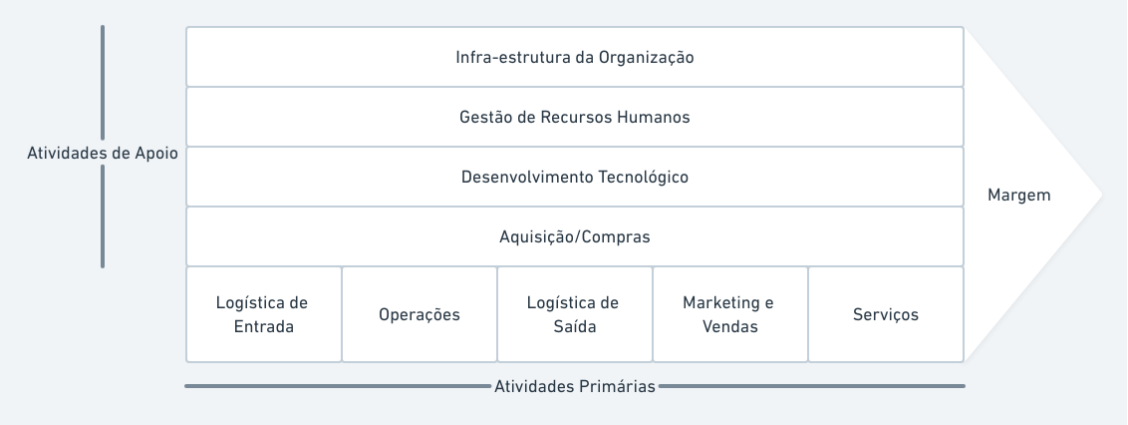

Figura 4.14: Cadeia de Valor de Porter

Como é possível perceber pela Figura 4.14, a organização e as suas diversas atividades estão divididas em duas grandes áreas: Atividades Primárias, Atividades de Apoio. As atividades primárias referem-se às atividades de produção dos produtos ou serviços da empresa assim como a venda e o suporte aos mesmos. As atividades de apoio garantem o funcionamento da organização assim como a produção dos produtos ou serviços e as condições para as atividades primárias serem desenvolvidas.

As atividades primárias deste modelo encontram-se divididas em cinco partes:

Logística de Entrada:
: Contempla todas as atividades de receção, armazenamento e controlo de inventário e inputs de produtos e serviços da organização. No âmbito desta dissertação, e por se tratar de um produto intangível, as atividades de logística de entrada serão a recolha de inputs por parte dos utilizadores, quer nos estudos realizados no planeamento do projeto quer nos possíveis estudos realizados ao longo do tempo de modo a manter a funcionalidade atual;

Operações:
: Contempla todas as atividades relacionadas com a conversão da matéria-prima (inventários, inputs de utilizadores, etc) em produtos. Relativamente a este projeto, a implementação e manutenção do novo editor de formulários assim como toda o produto E-goi serão as atividades principais;

Logística de Saída:
: Contempla todas as atividades de recolha, armazenamento e distribuição dos produtos e serviços aos clientes. Relativamente ao âmbito desta dissertação e por ser um produto web e intangível, as atividades de manutenção tanto do produto assim como a sua continua atualização são as únicas atividades de Logística de Saída;

Marketing e Vendas:
: Contempla todas as atividades de venda e promoção dos produtos e serviços da organização. Relativamente ao projeto, a realização de tutoriais, campanhas e vídeos promocionais são as maneiras de chegar ao cliente e por isso são as principais atividades desta parte da cadeia de valor. Uma vez que o produto E-goi pode ser utilizada gratuitamente, campanhas de angariação de clientes ou de adição de funcionalidades pagas podem também ser atividades a ter em consideração nesta vertente da cadeia;

Serviços:
: Contempla todas as atividades de apoio ao produto ou serviço. Exemplo disso, e no âmbito desta dissertação, são as atividades de apoio ao cliente, manutenção de sistemas e *hardware* que mantenham o E-goi operacional assim como atividades da organização que garantam o correto funcionamento das mesmas;

Relativamente às atividades secundárias, podem ser analisadas quatro partes:

Infra-estrutura da Organização:
: Contempla todas as atividades de gestão que têm como objetivo manter as operações diárias, como por exemplo administração, departamento financeiro, departamento de qualidade, atividades de planeamento, entre outras;

Gestão de Recursos Humanos:
: Contempla todas as atividades relacionadas com recursos humanos, nomeadamente colaboradores da organização, quer atividades relacionadas com contratação de mais colaboradores como a manutenção das condições dos colaboradores atuais. Organização de estágios, realização de formações internas, reajuste de salários consoante o avanço na carreira, prémios adicionais são exemplos de atividades de gestão de recursos humanos. Quando bem implementadas, estas ações garantem a motivação dos colaboradores aumentando o valor de cada um para a organização;

Desenvolvimento Tecnológicos:
: Contempla todas as atividades de desenvolvimento de processos e do desenvolvimento dos próprios produtos ou serviços da empresa. A atualização de metodologias de trabalho utilizadas assim como tecnologias (*software* e *hardware*, por exemplo) utilizadas são atividades importantes para aumentar a produtividade e qualidade dos produtos e serviços da organização;

Aquisição/Compras:
: Contempla todas as compras e aquisições a serem feitas para garantir a manutenção da organização assim como produtos e serviços da mesma. Exemplos disso são a aquisição de software e hardware, tanto para os colaboradores como para os produtos (quando aplicável).

A margem representada no modelo apresentado por Porter consiste na diferença entre o valor que o cliente paga para a obtenção de um produto ou serviço e o custo de produção dos produtos ou serviços aliados aos custos de manutenção da organização. A rentabilidade de um produto e consequentemente de uma organização é medida desta forma. O modelo da cadeia de valor tem a forma de uma seta em direção à margem uma vez que a cadeia de valor tem como objetivo final aumentar o valor para a organização e para os seus clientes.

Através da análise de toda a cadeia de valor é possível perceber que todas as atividades, ainda que divididas, estão relacionadas e, em conjunto, formam a organização. Relativamente ao caso da E-goi, e identificadas as atividades relativas à funcionalidade em questão, concluí-se que a implementação do novo editor de formulários é viável e exequível.

4.3 Sumário

Neste capítulo foi descrito todo o processo de engenharia de requisitos, implementando testes de usabilidade ao editor de formulários EasyForms para perceber, junto de utilizadores reais

do sistema o que alterar. Através desta análise, foi possível definir os requisitos (funcionais e não funcionais) que o novo editor terá de cumprir. Foi também descrito todo o processo de inovação, desde a identificação da oportunidade à definição do conceito, percebendo o valor que o desenvolvimento do conceito pode trazer à empresa. Foram geradas ideias que colmatassem a oportunidade identificada, foi escolhida uma ideia que foi transformada num conceito. O conceito foi analisado através de diferentes métodos e foram avaliados os custos e os ganhos da sua implementação.

Depois de concluir que é vantajoso para a empresa implementar o conceito definido, é importante desenhar uma solução que cumpra todos os objetivos e colmate as falhas.

Capítulo 5

Desenho

Para um produto ou funcionalidade ser implementado, é necessário um planeamento composto por diversas etapas. O objetivo do planeamento é definir como é que a solução será desenvolvida com base nos problemas identificados e no estado da arte. Como referido na Secção 1.5, o desenho de uma solução deve gerar artefactos, feitos com métodos rigorosos e testados. Para isso, foram desenvolvidos os diferentes artefactos que definem o novo editor, servindo como base para a implementação da solução.

5.1 Domínio

O domínio de uma aplicação consiste na definição de todos os conceitos e entidades relacionados com o problema que está a ser resolvido. Para que as novas alterações se possam refletir na solução a desenvolver, é importante remodelar o diagrama de domínio. Utilizando novamente a linguagem UML, é possível desenvolver um diagrama de componentes que represente o domínio do novo editor de formulários.

O diagrama de domínio proposto na figura 5.1 é bastante similar ao diagrama apresentado na figura 2.3 da Secção 2.4 com algumas exceções. A entidade Formulário/LandingPage foi separada em duas entidades independentes, com opções próprias. Ambas as novas entidades, Formulario e LandingPage estão ligadas à entidade Utilizador, uma vez que este pode criar vários formulários ou várias *landing pages*. As entidades estão também interligadas entre si, visto que uma *landing page* continua a poder conter um formulário. A funcionalidade não é eliminada mas deixa de ser obrigatório que uma página tenha um formulário. Outra alteração que se pode constatar é o desaparecimento da entidade Publicacao. Como referido no na Secção 2.4, esta funcionalidade deixa de ter sentido uma vez que existem tipos quer de formulários quer de *landing pages*. Por exemplo, ao criar um formulário do tipo *embedded* deixa de ser necessária uma publicação, o tipo de publicação é definido quando é feita a escolha do tipo de formulário.

As alterações efetuadas resolvem o problema identificado na Secção 1.3 relativo ao acoplamento da funcionalidade de formulários e da funcionalidade de *landing pages*. Com estas alterações ao modelo de domínio, é possível iniciar o desenvolvimento do novo sistema corrigindo e melhorando os aspetos menos positivos detetados no editor EasyForms.

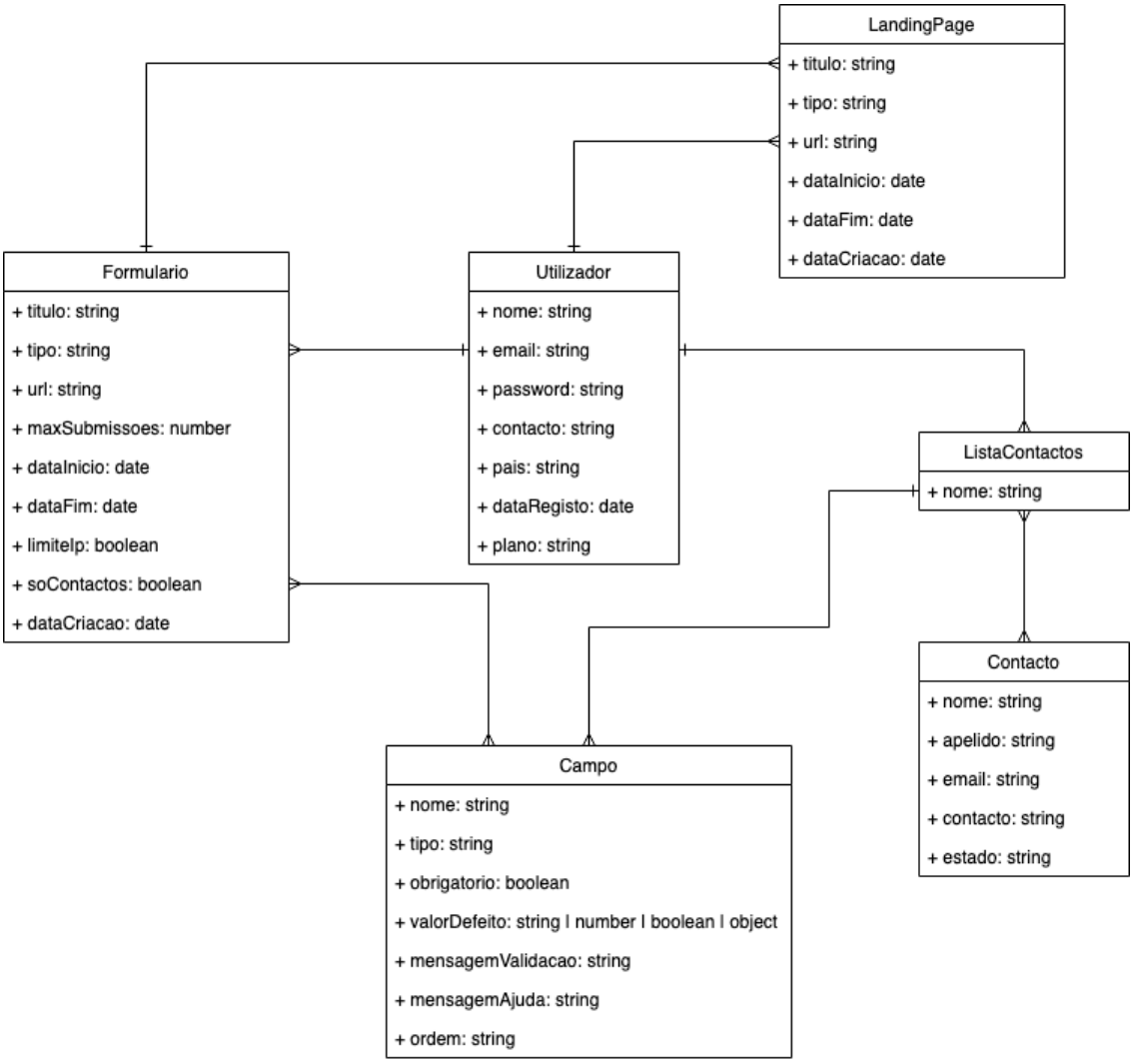

Figura 5.1: Diagrama de Domínio do Novo Editor de Formulários

5.2 Arquitetura

Da mesma forma que é indispensável fazer uma planta para construir um edifício, é indispensável fazer arquitetura de um *software* para o desenvolver. Através desta analogia, é possível perceber que a arquitetura em engenharia de *software* tem como objetivo desenhar a estrutura do *software* sem qualquer tipo de preocupação relativa a limitações de implementação ou tecnologias, focando apenas no comportamento dos diversos componentes. Para desenvolver este artefacto, foi usada, novamente, a linguagem UML, desta vez com um diagrama de componentes.

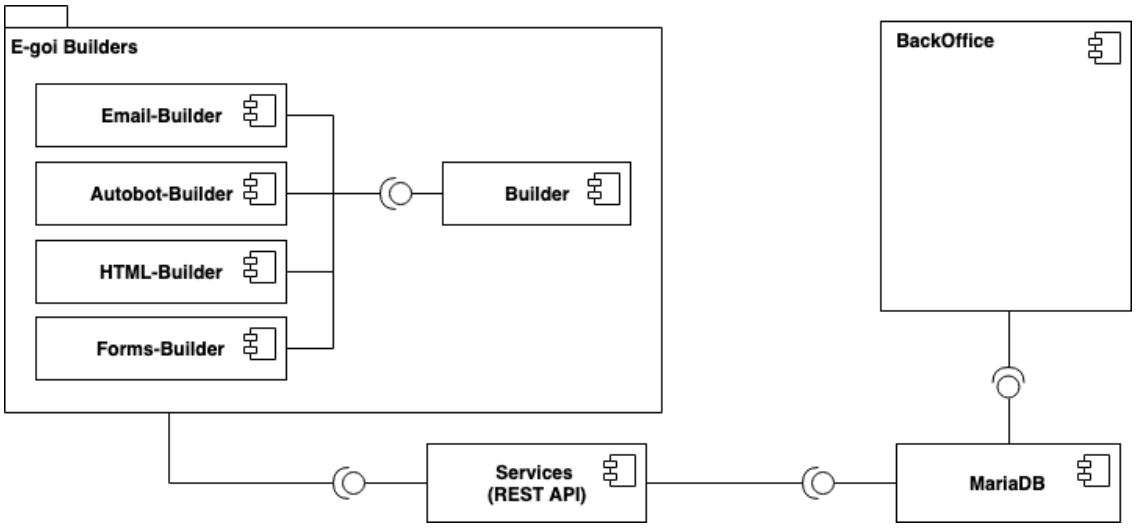

Figura 5.2: Diagrama de Componentes do Novo Editor de Formulários

A arquitetura proposta na figura 2.4 traz algumas alterações que resolvem os problemas identificados na Secção 1.3 relativos à manutenção e escalabilidade do sistema. Através da arquitetura proposta, é possível perceber que o componente EasyForms deixa de existir, sendo retirado do componente BackOffice e dividido entre dois componentes: o novo componente Forms-Builder e o componente Services. Na arquitetura, toda a lógica de interface e visualização do editor de formulários passa a estar presente no componente Forms-Builder, desenvolvido em Angular (apresentado na Secção 3.2.2. Este componente contém a implementação do editor e da *user interface* que permite desenvolver formulários. Toda a lógica de negócio associada ao novo editor de formulários passa a estar contida no componente Services, tendo sido acrescentados todos os métodos necessários para as diferentes funções do editor de formulários.

A remoção da funcionalidade de formulários do componente BackOffice corrobora o esforço da empresa para a divisão do monólito em funcionalidades independentes, como é descrito na Secção 2.4. A arquitetura garante escalabilidade, aumentando as instâncias dos componentes Forms-Builder e Services consoante as necessidades e a utilização do editor. A manutenção do sistema fica também facilitada uma vez que o editor está repartido por diversos componentes, sendo mais fácil acrescentar funcionalidades assim como manter as funcionalidades existentes, quer a nível de interface quer a nível de servidor. Estas alterações colmatam o problema identificado na Secção 1.3 relativos a escalabilidade e manutenção.

Todos os componentes não mencionados nesta secção mantém-se inalterados relativamente ao diagrama de componentes apresentado na figura 5.2 da Secção 2.4.

5.3 Implantação

A implantação de um projeto de *software* consiste na instalação do mesmo nas máquinas de modo a disponibilizar o sistema aos utilizadores. Esta é um passo bastante importante para as organizações, uma vez que as máquinas utilizadas para a instalação do *software* consistem em custos para as empresas e organizações. Para apresentar o artefacto que define a implantação do sistema, foi usada, novamente, a linguagem UML, desta vez com um diagrama de implantação.

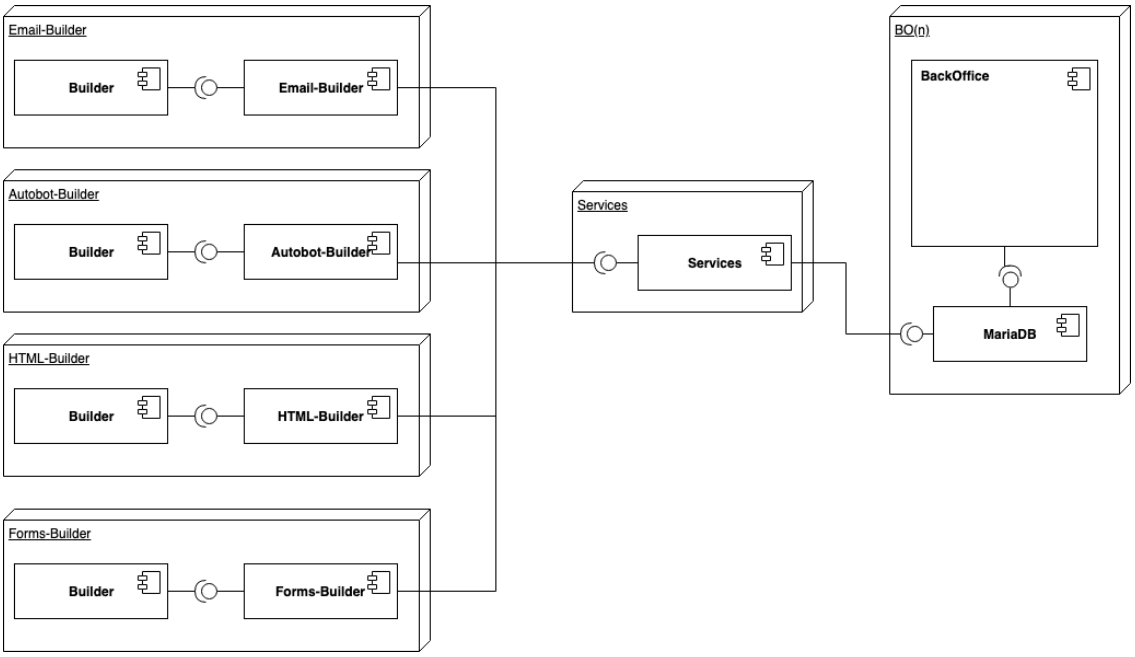

Figura 5.3: Diagrama de Implantação do Novo Editor de Formulários

A gestão das máquinas a serem utilizadas e o número de instâncias dos diferentes componentes em cada máquina é um ponto bastante importante para garantir que não são utilizados recursos desnecessários. A nova implantação resolve os problemas identificados removendo a funcionalidade do editor (EasyForms) do componente BackOffice, criando o componente Forms-Builder que estende o componente Builder e é independente de qualquer outro componente. Desta forma, é possível ter uma máquina dedicada ao componente Forms-Builder, sendo possível escalar o número de instâncias consoante a necessidade.

5.4 Serviços

Para que as alterações que um utilizador produz na interface de um sistema sejam persistidas, é necessário efetuar uma ligação entre a interface e o servidor que contém toda a lógica de negócio e a persistência de dados. Como apresentado na Secção 5.1, o componente Services contém serviços que comunicam com a base de dados do produto E-goi e disponibiliza *endpoints* estabelecer conexão com os serviços. A funcionalidade do editor de formulários já existe e, consequentemente, os serviços necessários para a funcionalidade também já estão implementados, no entanto, para que o novo editor funcionasse, foi necessário fazer algumas alterações aos serviços existentes.

As alterações efetuadas nos serviços consistiram, maioritariamente, na alteração do modelo de dados guardado, uma vez que o modelo de formulário foi alterado com a utilização do novo editor. Desta forma, os serviços estão adaptados a trabalhar com ambos os conceitos de formulários, para que a transição possa ser feita de uma forma suave mantendo os dois editores ativos ao mesmo tempo e, posteriormente, eliminar a funcionalidade do antigo editor. Estas alterações são especificadas na Secção 5.1.

Foram usados diversos serviços existentes no componente Services:

GET /forms - Obter a listagem de todos os formulários de um utilizador;

POST /forms - Criar um novo formulário;

PUT /forms/:id - Editar o formulário com o id especificado no URL;

DELETE /forms/:id - Apagar o formulário com o id especificado no URL;

GET /common/prefixes - Obter a listagem de indicativos para apresentar nos campos do tipo telemóvel e telefone;

GET /lists/:id/fields - Obter a listagem de todos os campos da lista cujo id é especificado no URL.

Todos os pedidos são feitos por Hypertext Transfer Protocol (HTTP), o que permite a utilização de *headers*. Para qualquer pedido é necessário um *header* com o *token* de autenticação, de forma a garantir segurança no acesso aos serviços.

5.5 Design de Mock-Ups

Como foi possível perceber através do teste de usabilidade realizado ao editor EasyForms (na Secção 4.1.1), a maioria dos problemas identificados estão relacionados com a interface gráfica do editor de formulários. Foi necessário desenvolver uma nova interface gráfica que, para além de resolver os problemas identificados, cumprisse com as especificações definidas no levantamento de requisitos do novo sistema (tanto os requisitos funcionais identificados na Secção 4.1.2 como os requisitos não funcionais identificados na Secção 4.1.2). Para a construção da interface foram desenvolvidos *mock-ups* não funcionais que permitissem perceber como é que o editor se iria comportar assim como o seu aspeto gráfico. O desenvolvimento de *mock-ups* trata-se de um processo bastante rápido que permite a alteração do design de uma forma simples, garantindo assim que a interface resolve os problemas identificados cumprindo com o levantamento de requisitos.

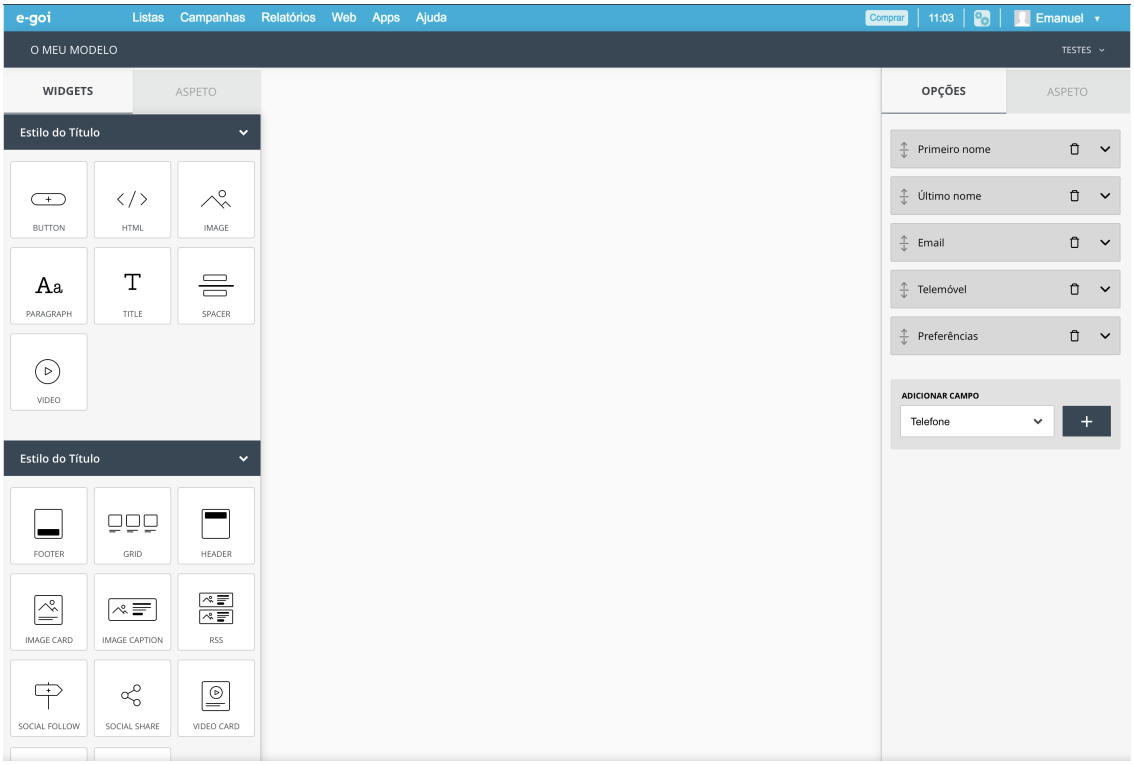

Figura 5.4: Mock-up 1 - Editor com todas as opções dos campos colapsadas

Como se pode constatar na figura 5.4, a abordagem escolhida para o editor consiste numa zona de *widgets* do lado esquerdo, uma zona de pré-visualização no centro do editor e uma zona de opções do lado direito. A zona de pré-visualização tem como propósito renderizar os *widgets* adicionados através de *drag & drop* assim como as alterações feitas aos *widgets* adicionados na barra lateral direita. Na barra lateral esquerda, estão todos os *widgets* que podem ser usados na construção do formulário [1]. Na barra lateral direita são apresentadas todas as opções do *widget* selecionado, sendo que a alteração destas opções é refletida na zona de pré-visualização.

[1] Os *widgets* apresentados podem não ser definitivos.

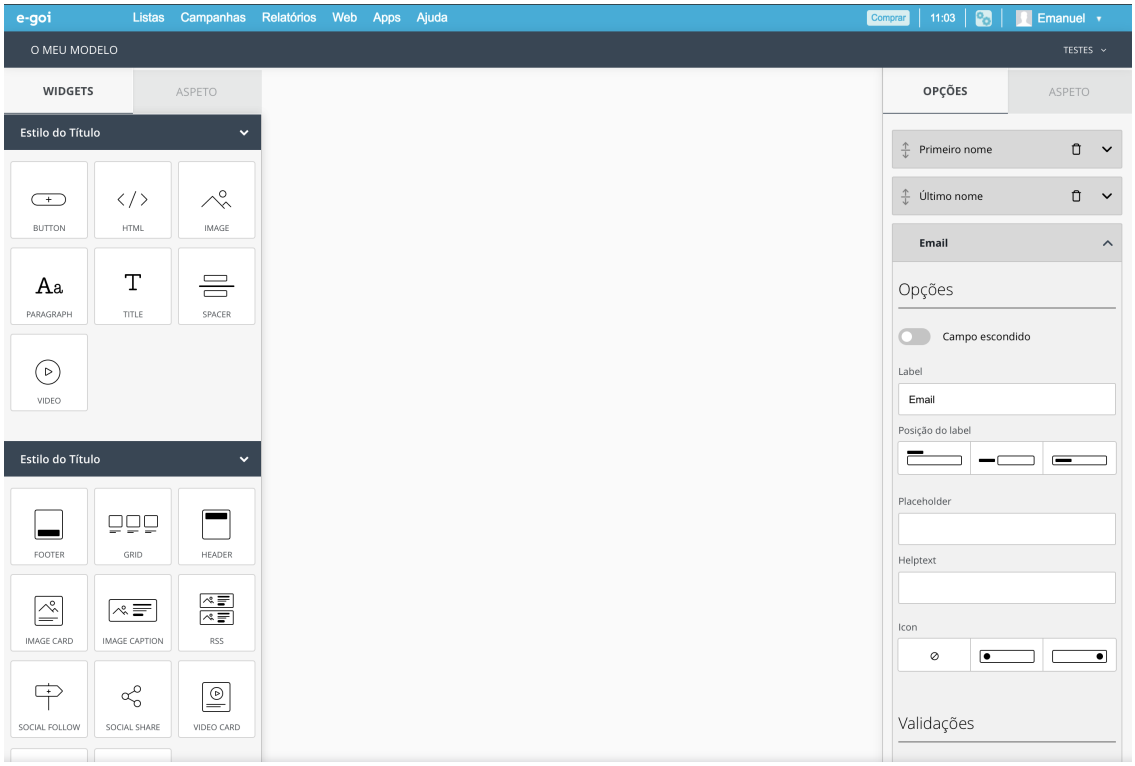

Figura 5.5: Mock-up 2 - Editor com todas as opções do campo abertas

De forma a simplificar uma futura integração com um editor de *landing pages*, os formulários foram implementados como *widgets*. Desta forma, como qualquer outro *widget*, as opções do formulário aparecem na barra lateral direita e as suas alterações refletem-se na pré-visualização.

As opções do *widget* formulário consistem em duas partes distintas: a área para adicionar campos ao formulário (cumprindo com a *User Story* 3 [4.5]) e a área com as opções específicas de cada campo, como é possível ver na figura 5.5. Os campos passíveis de serem adicionados ao formulário consistem nos campos que existem na lista de contactos selecionada antes de iniciar a edição do formulário. O mapeamento dos campos do formulário e dos campos da lista de contactos é automático, diminuindo o esforço ao utilizador que, no editor EasyForms, necessitava de fazer o mapeamento explicitamente. Este foi um dos problemas identificados que é resolvido com a nova interface. As opções dos campos presentes no formulário são apresentadas numa lista de janelas colapsáveis onde é possível mover e apagar assim como editar todas as opções específicas do mesmo (cumprindo com a *User Story* 6 [4.8]). É também possível criar campos na lista dentro do editor, escolhendo a opção "Campo Extra"na lista de campos existentes e fornecendo o tipo de campo e o nome do mesmo (cumprindo com a *User Story* 4 [4.6]).

Uma vez que um campo de um formulário pode ter diversos tipos, identificados na Secção 2.4, é necessário perceber as opções de cada tipo de campo, definidas na seguinte tabela.

Tipo de Campo	Opções
Todos os campos	Texto identificador do campo Posição do texto identificador Texto de Ajuda *Placeholder* Valor por defeito Campo obrigatório Campo escondido
Texto	Ser caixa de texto Tamanho da caixa de texto
Email	Mostrar icon de email Posição do icon de email
Telemóvel	Mostrar icon de email Posição do icon de email
Telefone	Mostrar icon de email Posição do icon de email
Data	Limite inferior de data ("antes de") Limite superior de data ("depois de")
Número	Limite inferior de número Limite superior de número
Lista de Opções	Modo de apresentação das opções (*select*, *radio buttons* ou *checkbox*) Listagem das opções existentes e edição de opções

Tabela 5.1: Tipos de Campos e Respetivas Opções

Na tabela 5.1 podemos perceber todos os campos a ser implementados assim como as suas opções, quer as opções comuns a todos os campos assim como as opções específicas de cada um. Todo o design das opções segue os *standards* e o design da *framework* Bootstrap, de modo a cumprir os requisitos não funcionais definidos. Para além das opções definidas em cada tipo de campo, existem também opções gerais do formulário (cumprindo com a *User Story* 5 [4.7]) sendo elas:

- Espaçamento vertical entre campos;
- Espaçamento horizontal dos campos à margem do formulário;
- Estilos do botão de submissão do formulário;
- Posição do botão de submissão do formulário;

De forma a validar os *mock-ups*, foi realizado um *focus group* [55] composto por cinco pessoas: dois programadores, um designer, um especialista em *user experience* e o Chief Techonology Officer (CTO) da empresa. O facto do grupo ser constituído por pessoas com áreas de especialidades diferentes garante uma discussão mais construtiva uma vez que cada elemento tem uma visão diferente consoante as competências. O *focus group* consistiu na avaliação da interface desenvolvida, moderada por um investigador. O investigador colocou diversas questões aos participantes fomentando a discussão e apontando as principais ideias debatidas. No fim, o grupo definiu consensualmente as conclusões atingidas assim como as alterações e/ou preocupações necessárias a ter em consideração relativamente à proposta do design para o novo editor de formulários.

A principal preocupação detetada estava relacionada com o campo "Lista de Opções". Neste campo, por se tratar do tipo de campo com opções mais complexas, é necessário explicar a listagem das opções existentes nesse campo. Estas opções são definidas na criação do campo na lista de contactos e, por serem específicas da lista e não do formulário, podem ser usadas em diversos formulários.

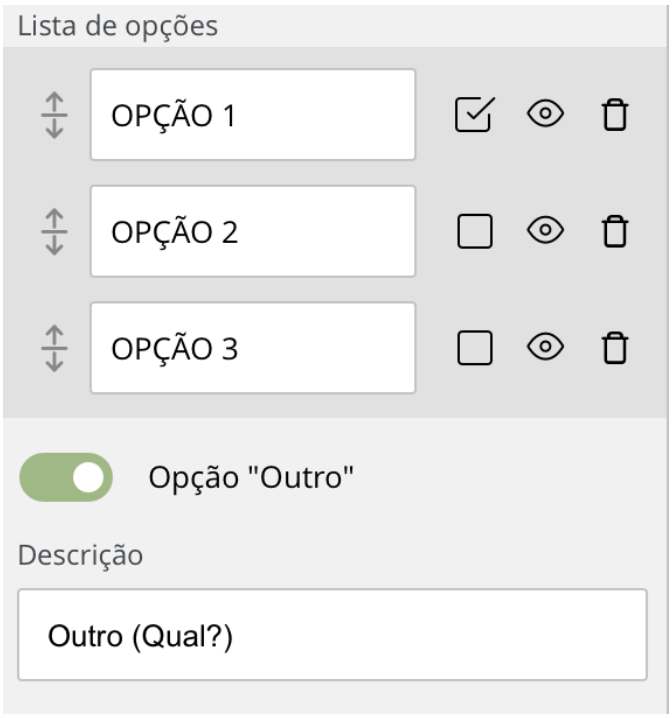

Figura 5.6: Listagem de Opções do tipo de campo "Lista de Opções"

A figura 5.6 mostra a listagem das opções existentes e edição de opções do campo "Lista de Opções". Como é possível perceber pela figura, é possível alterar a ordem das opções, editar as opções na caixa de texto, fazer com que a opção seja selecionada por omissão, esconder ou eliminar a opção.

A preocupação detetada no *focus group* prende-se com o facto das mudanças (edição ou remoção) feitas às opções afetarem não apenas o formulário a ser editado mas assim como qualquer outro formulário que usasse o mesmo campo da lista. Para tornar esta questão mais clara para o utilizador, foi retirada a funcionalidade de remover uma opção (por ser destrutiva) e foram criadas janelas específicas para a edição de opções que avisam o utilizador das consequências das alterações feitas às opções da "Lista de opções".

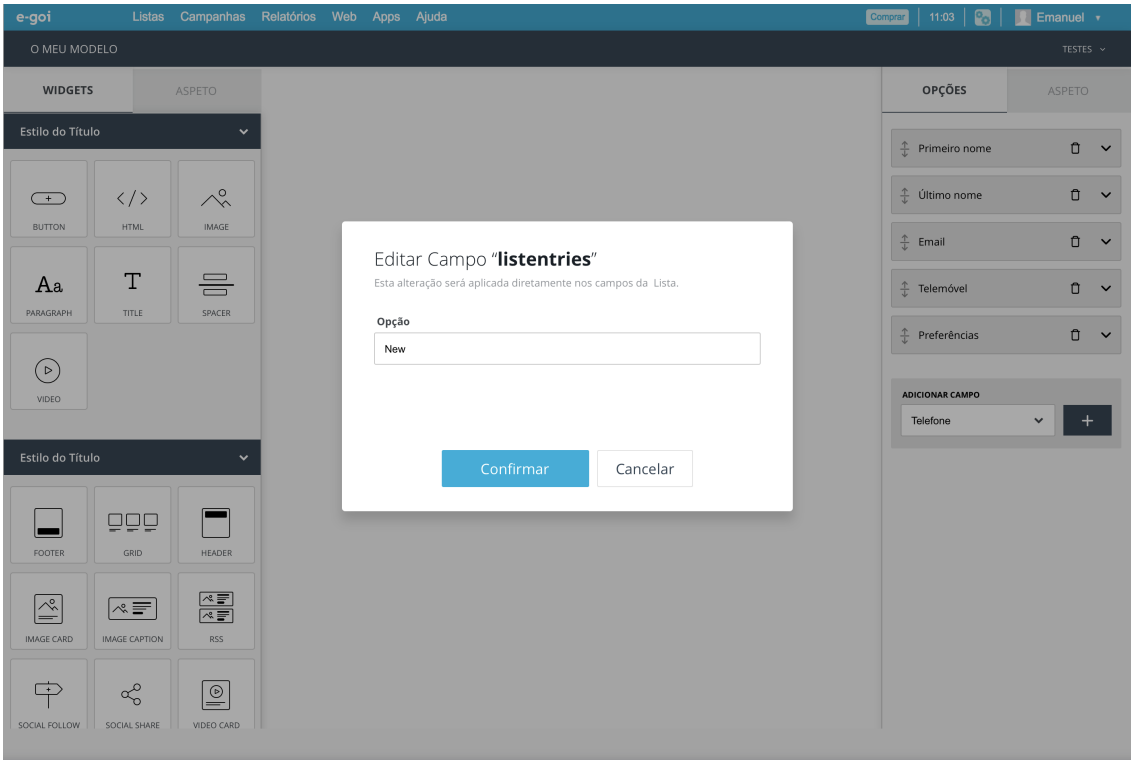

Figura 5.7: Janela de edição de opções do campo "Lista de Opções"

Através da janela representada figura 5.7 garante-se que o utilizador conhece as consequências da ação que pretende efetuar. Foi criada uma janela igual para a adição de opções motivada pelas conclusões do *focus group* adicionando a funcionalidade ao utilizador para adicionar uma nova opção à "Lista de Opções"ficando também explicito que, ao efetuá-lo, será adicionada a opção a todos os formulários que utilizem o campo da lista.

5.6 Sumário

Neste capítulo foi apresentado o desenho da solução, quer técnica como de interface gráfica, que resolve os problemas identificados. Foi apresentada toda a vertente técnica do mesmo (arquitetura do sistema, modelação do domínio e implantação do sistema). A solução desenhada resolve os problemas identificados na Secção 1.3 assim como nos testes implementados na Secção 4.1.1, validando e cumprindo assim a Análise de Valor desenvolvida na Secção 4.2 Também foram definidos os serviços que o editor necessitará de utilizar assim como a interface que o editor terá para os utilizadores finais.

Segue-se a implementação do sistema, baseado em todo o planeamento efetuado neste capítulo, onde serão explicados os pormenores técnicos.

Capítulo 6

Implementação

Todo o planeamento e análise de um sistema culmina na sua implementação, seguida de uma análise contínua ao sistema e manutenção do sistema. Neste capítulo é apresentada a implementação do novo editor de formulários do E-goi, baseado no planeamento e análise desenvolvidos nos capítulos anteriores deste documento. Este capítulo apresenta a metodologia de desenvolvimento, a implementação da interface gráfica do editor e as alterações feitas aos serviços necessários.

6.1 Metologia de Desenvolvimento

Para que uma equipa consiga trabalhar em conjunto, é necessário que exista uma metodologia de desenvolvimento comum a todos os elementos. A E-goi implementa algumas estratégias para manter a organização entre os mais de 10 programadores da empresa. Apesar do âmbito deste projeto ser maioritariamente individual, foram seguidas as práticas correntes na empresa para a organização e estruturação do desenvolvimento de *software*.

A E-goi utiliza a ferramenta Git [42], apresentada na Secção 3.2.2, como sistema de controlo de versões para todos os seus projetos e produtos. A ferramenta que permite a utilização de Git na E-goi é a aplicação GitLab [56] instalada num servidor interno. Esta ferramenta permite a integração com outro tipo de ferramentas, nomeadamente ferramentas de integração contínua.

Para cada projeto existem pelo menos três ramos (*branches*), um ramo onde é feito o desenvolvimento e que contém código inacabado, um ramo onde são feitos diversos testes e um ramo de produção onde está presente o código disponível para o utilizador final. O fluxo consiste na implementação de funcionalidades no ramo de desenvolvimento, seguido de testes às funcionalidades no ramo de testes e disponibilização da funcionalidade no ramo de produção. Para isto é feito *merge* entre os diversos ramos, uma operação que permite juntar as alterações feitas num ramo a outro ramo diferente. Apesar de existir apenas um programador neste projeto, foi implementada a mesma lógica de modo a que, no futuro, seja fácil integrar novos membros na equipa e continuar o desenvolvimento do projeto.

Para disponibilizar o seu produto para os utilizadores assim como para testar todo o produto de uma forma contínua (através de integração contínua, descrita na Secção 3.2.2), a E-goi utiliza a ferramenta Jenkins [57] num servidor interno para gerir todos os seus projetos. Integrado com o GitLab, a ferramenta Jenkins, disponibiliza a automação de testes e compilação do código desenvolvido. Sempre que existe novo código no ramo de testes, a ferramenta

Jenkins despoleta testes automáticos ao código fazendo também uma compilação do projeto, garantindo assim a funcionalidade do código através da execução de testes unitários, apresentados na Secção 7.1. Quando entra código novo no ramo de produção, são despoletados novamente os testes automáticos assim como o *deploy* da aplicação, distribuindo assim o código das novas funcionalidades para todos os clientes.

Utilizando esta metodologia, torna-se mais simples a gestão de todo o código desenvolvido assim como a garantia da sua qualidade, sendo garantida a análise à existência de *bugs* e más práticas de programação, gestão de testes e garantias de percentagens mínimas de cobertura de testes. Estas métricas são apresentadas, de uma forma mais detalhada, na Secção 7.2.1. Também simplifica os processos internos das equipas uma vez que todos os elementos seguem o mesmo fluxo para a criação e manutenção de código.

6.2 Interface Gráfica

Como descrito no Capítulo 1, o principal objetivo deste projeto é a implementação de um novo editor de formulários e, para tal, foram desenvolvidos *mock-ups* no Capítulo 5 que servem como base para a implementação da interface gráfica do editor. Para além dos *mock-ups*, no Capítulo 5 também são definidos os requisitos que o editor tem de cumprir, quer funcionais, quer não funcionais. Para o desenvolvimento da interface gráfica foi utilizada a *framework* Angular, apresentada na Secção 3.2.2, cumprindo um dos requisitos não funcionais apresentados na Secção 4.1.2. Para implementar a interface do editor, foi necessário dividir o *mock-up* produzido em módulos e componentes assim como perceber quais os serviços necessários para o editor funcionar corretamente. A divisão dos módulos é apresentada na figura 6.1.

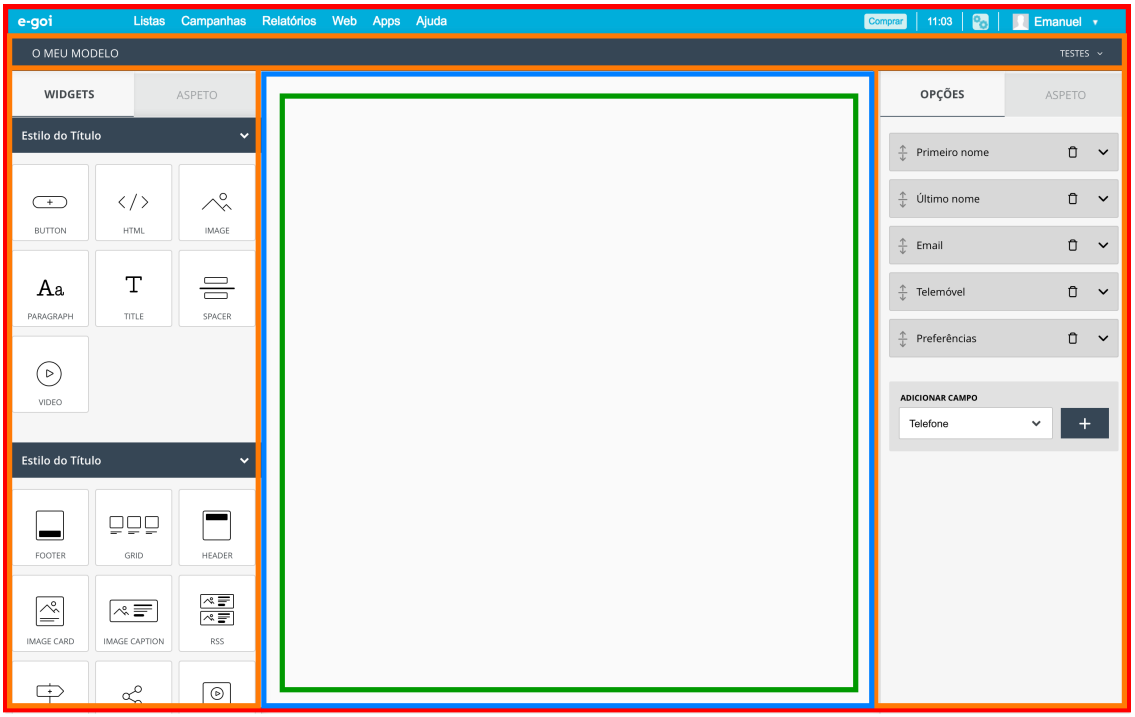

Figura 6.1: Divisão de Mock-up em Módulos

Para implementar o editor, foram implementados quatro módulos Angular:

AppModule (representado a vermelho) - módulo que inicia a aplicação e que importa todos os outros módulos;

EditorModule (representado a laranja) - módulo que contém as barras laterais do editor;

PreviewModule (representado a azul) - módulo que contém a zona de pré-visualização do formulário;

BootstrapModule (representado a verde) - módulo que contém todos os componentes que podem ser inseridos num formulário (campos do formulário, colunas, linhas, entre outros).

Os módulos foram agrupados por funcionalidades sendo que cada módulo tem uma responsabilidade específica. Enquanto que os módulos consistem numa divisão mais abstrata, os componentes contêm o código para renderizar as diferentes partes que compõem a interface. Os componentes existem dentro dos seus módulos podendo, no entanto, ser reaproveitados. Na figura 6.2 é possível perceber os diferentes componentes existentes na interface.

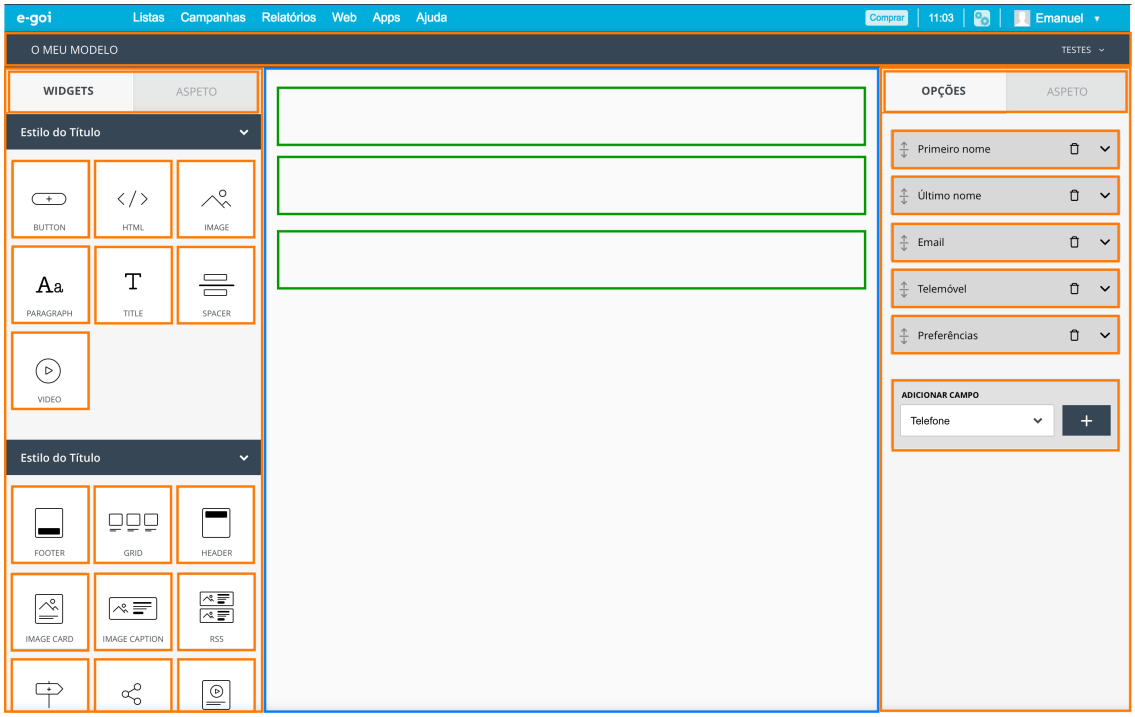

Figura 6.2: Divisão de Mock-up em Componentes

Em cada módulo, existem diversos componentes apresentados na seguinte lista:

AppModule:

- **AppComponent** - componente que contém o *iframe* que apresenta todos os outros componentes;

EditorModule (representado a laranja):

- **EditorComponent** - componente que contém o *iframe* que apresenta as duas barras laterais e a barra de topo do formulário;

ToolbarComponent - componente que contém a barra de topo;

ControlAttributesComponent - componente que contém a listagem dos widgets disponíveis (aba "WIDGETS"da barra lateral esquerda);

GlobalStyleComponent - componente que contém a listagem dos estilos globais do formulário (aba "ASPETO"da barra lateral esquerda);

WidgetOptionsComponent - componente que contém a listagem das opções do *widget* selecionado (aba "OPÇÕES"da barra lateral direita);

WidgetStyleComponent - componente que contém a listagem dos estilos do *widget* selecionado (aba "OPÇÕES"da barra lateral direita);

BuilderAttributeRenderComponent - componente que contém as diferentes opções e estilos dos *widgets*;

AddFieldDialogComponent - componente que contém a modal para adicionar um campo ao *widget* formulário;

EditFieldDialogComponent - componente que contém a modal para editar um campo ao *widget* formulário;

PreviewModule (representado a azul):

PreviewComponent - componente que contém o *iframe* que renderiza os diferentes campos do formulário;

BootstrapModule (representado a verde) [1]:

InputCellphoneComponent - componente que contém o campo do tipo "telemóvel";

InputDateComponent - componente que contém o campo do tipo "data";

InputEmailComponent - componente que contém o campo do tipo "email";

InputNumberComponent - componente que contém o campo do tipo "número";

InputOptionsComponent - componente que contém o campo do tipo "lista de opções";

InputPhoneComponent - componente que contém o campo do tipo "telefone";

InputTextComponent - componente que contém o campo do tipo "texto";

Uma vez que os principais componentes do EditorModule e PreviewModule (EditorComponent e PreviewComponent, respetivamente) consistem em *iframes*, eles comunicam através de uma funcionalidade do HyperText Markup Language 5 (HTML5) denominada postMessage. Esta função permite criar emissores e recetores de mensagens, facilitando assim a troca de dados entre os dois *iframes*. Esta troca de mensagens entre ambos os componentes é fundamental para garantir consistência do sistema, por exemplo, garantir que ao adicionar um *widget* ao formulário, o PreviewComponent está pronto para o apresentar evitando assim erros inesperados.

Paralelamente ao uso da funcionalidade postMessage, uma vez que esta só funciona entre janelas ou *iframes*, tornou-se necessário implementar outra maneira de transmitir dados entre

[1] Campos do formulário não representados na imagem.

os diferentes componentes de modo a manter todo o sistema coerente e consistente. Para isso utilizou-se NgRx, uma *framework* que permite a gestão de estados de forma reativa [58]. NgRx consiste numa *store* que guarda o estado da aplicação, modificando-o e gerando novos estados consoante alterações feitas na aplicação. Para isso, é criado o estado inicial quando é iniciada a aplicação assim como ações que modificam este estado sendo que cada vez que existe uma alteração (adicionar um widget, por exemplo) é disparada uma ação para a *store*. Ao receber essa ação, a *store* altera o seu estado gerando um novo estado. Esta ferramenta é bastante usada em aplicações Angular para garantir consistência em toda a aplicação e para questões de *debugging* de código.

Para entender como é que o novo estado gerado pela *store* se reflete na interface é necessário perceber o conceito de Observables [59]. O conceito de Observable permite a transmissão de mensagens de uma forma assíncrona entre consumidores e produtores. A transmissão de mensagens é feita através de subscrições, ou seja, um consumidor subscreve um produtor e começa a receber os dados emitidos por este. Sempre que o produtor emitir novos dados, os subscritores recebem esses dados podendo atuar sobre eles até anularem a subscrição de forma explicita.

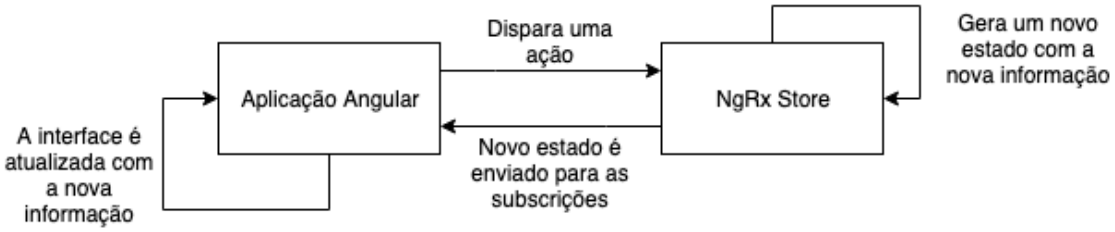

Figura 6.3: NgRx Store

Na figura 6.3 é possível perceber como é que a aplicação interage com a *store* e como é que as alterações são feitas na interface. Quando a aplicação é carregada, os componentes subscrevem a *store* estando assim prontos para receber novas informações para renderizar. Sempre que existe alguma ação no editor, é disparada uma ação com o tipo de ação e os dados necessários para fazer as alterações. A *store* atualiza-se e gera um novo estado, emitindo este estado para todos os subscritores. Os componentes, subscritos à *store* recebem os novos dados e atualizam a interface através do método de deteção de alterações do Angular. Desta forma, uma vez que a *store* é instanciada dentro do *browser* do cliente, as alterações não dependem de pedidos ao servidor garantindo assim velocidade e fluidez do sistema. Quando possível, os pedidos ao servidor são feitos de uma forma paralela às alterações na *store*.

Seguindo o princípio de responsabilidade única, de modo a que os componentes apenas sejam responsáveis pela renderização na interface, foi criado um serviço Angular que é responsável por disparar ações para a *store* assim como fazer toda a manipulação de dados necessária para que a ação seja concretizada.

Para perceber a interação entre os diversos que compõe o editor, utilizou-se um diagrama UML de sequência, apresentado na Secção 3.2.1 que demonstra a utilização do editor de formulários.

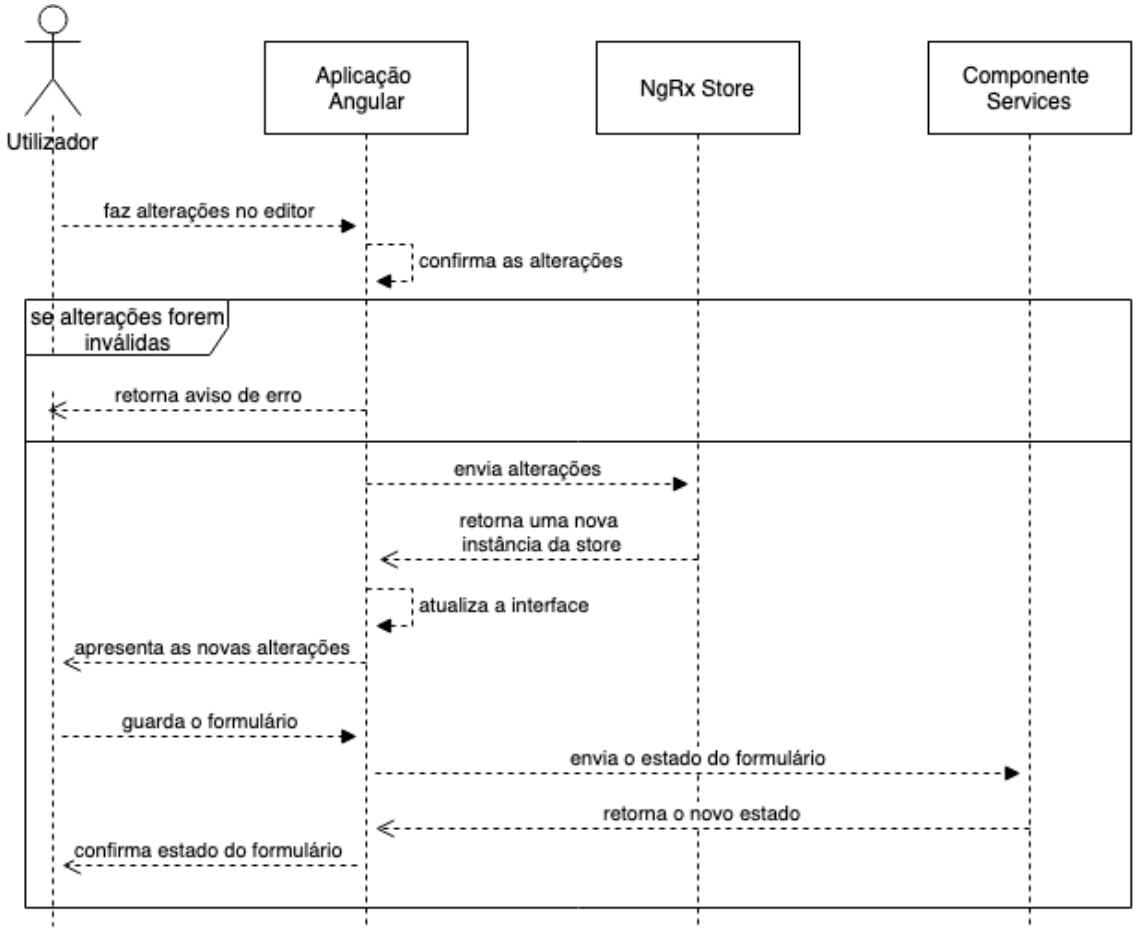

Figura 6.4: Funcionamento do Editor de Formulários

Como é possível perceber através da análise do diagrama de sequência apresentado na figura 6.4, o utilizador realiza uma alteração no editor que valida a mesma. No caso da edição ser inválida, como por exemplo a inserção de uma letra numa opção numérica, o editor retorna uma mensagem de erro ao editor, ajudando-o a ultrapassar o erro. Este comportamento resolve um dos problemas identificados através do teste de usabilidade apresentado na Secção 4.1.1, relativos à recuperação de erros por parte o utilizador. Caso a alteração seja válida, o fluxo normal é iniciado, enviando a alteração para a NgRx Store. Esta é atualizada e emite um novo estado, que será recebido pelo editor através da uma subscrição, como explicado anteriormente. A interface é atualizada consoante o novo estado recebido. Como definido na *User Story 7*, definida na Secção 4.1.2, o utilizador pode explicitamente guardar o seu formulário apesar de o editor fazer isso de uma forma implícita periodicamente. Ambas as formas resultam num pedido para o componente Services, que guarda as alterações na camada de persistência de dados e retorna o novo estado do formulário que é apresentada ao utilizador.

Em suma, toda a implementação da parte gráfica do editor gerou a criação de 23 componentes Angular, inseridos em 4 módulos Angular que comunicam entre si. Foram geradas 5300 linhas de código sendo que cerca de 3700 dessas linhas são desenvolvidas em TypeScript, 800 em CSS e 800 em HTML.

6.3 Serviços

O componente Services consiste numa REST API desenvolvida em PHP, utilizando a *framework* Zend Framework [60], que disponibiliza diversos *endpoints* que permitem interagir com o sistema E-goi. Como referido na Secção 5.1 foram alteradas as estruturas do modelo formulário de modo a implementar o novo editor.

Como referido, o componente Services comunica através do protocolo HTTP, disponibilizando recursos para que os seus clientes possam fazer pedidos. Os serviços alterados consistem no método POST do recurso /forms assim como o método PUT do recurso /forms/:id. Para perceber as alterações feitas é necessário perceber como funcionam estes dois serviços atualmente.

A informação necessária para criar um formulário consiste na seguinte tabela:

Atributo	Descrição	Tipo de Dados
id	id do formulário	string
html	HTML do formulário	string
internalName	nome único interno do formulário	string
externalName	nome externo do formulário	string
listId	id da lista associada ao formulário	string
userId	id do utilizador que cria o formulário	string
dateEnd	data de fim da atividade do formulário	date
dateStart	data de inicio da atividade do formulário	date
ipLimit	definição de limitação de resposta por IP	boolean
justContacts	definição de limitação de resposta apenas por contactos	boolean
maxSubmits	definição do número máximo de submissões	number
channel	canal pelo qual o formulário foi exportado	string
created	data em que o formulário foi criado	date
updated	última data em que o formulário foi atualizado	date
createdBy	id do utilizador que criou o formulário	string
updatedBy	id do utilizador que fez a última atualização ao formulário	string

Tabela 6.1: Atributos para Criação de um Formulário

Uma vez que o conceito de formulários será desacoplado do conceito de *landing page*, apesar de poder existir uma ligação entre os dois (um formulário estar contido numa *landing page*), é necessário acrescentar atributos que definam esta ligação. Por este motivo, é necessário fazer uma ligação entre as duas entidades tendo sido acrescentado o atributo *idLandingPage* à entidade formulário e o atributo *idForm* à entidade *landing page*. Para representar esta ligação no modelo de dados, foi necessária a criação de uma tabela intermédia entre as duas entidades, exibida na figura 6.5.

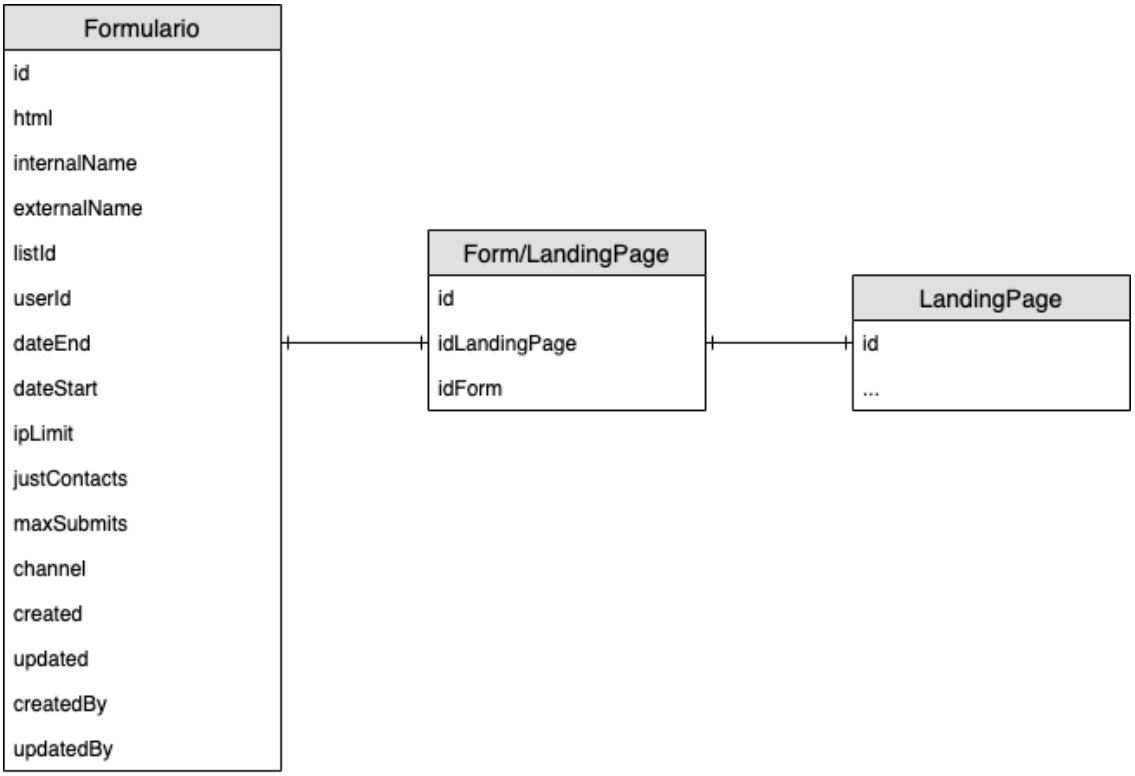

Figura 6.5: Nova Tabela Criada na Camada de Persistência

Com a criação da tabela Form/LandingPage, presente na figura 6.5, é possível ligar um formulário com diversas *landing pages* garantindo que o modelo relacional da base de dados é respeitado.

Atualmente, o editor abre formulários através do HTML guardado na base de dados. Este processo é complexo uma vez que o editor necessita de converter o HTML para conhecer quais os *widgets* presentes em cada formulário e fazer a renderização dos mesmos. Para evitar este processo, foi criada uma nova tabela na base de dados com o objetivo de guardar uma lista de todos os *widgets* de cada formulário.

A nova tabela guarda as seguintes informações:

Atributo	**Descrição**	**Tipo de Dados**
id	id do *widget*	string
formId	id do formulário ao qual o *widget* pertence	string
widgetType	tipo do *widget*	string
sort	ordem do *widget* no formulário	number
config	objeto JSON que contém todas as definições do *widget*	object
created	data em que o formulário foi criado	date
createdBy	id do utilizador que criou o formulário	string

Tabela 6.2: Atributos para Criação de um Widget de um Formulário

É possível guardar *widgets* na nova tabela fazendo um pedido HTTP com o método POST para o recurso /formsbuilder, passando toda a informação necessária como o corpo do pedido. Uma vez que um formulário consiste num *widget*, todas as informações específicas dos campos do formulário são guardados nesta tabela. Uma vez que cada cliente E-goi possui uma base de dados própria, problema da existência de um número elevado de entradas nesta tabela fica controlado, uma vez que é esperado que um utilizador E-goi não crie um número exagerado de *widgets*.

6.4 Sumário

Neste capítulo foi explicado todo o desenvolvimento da solução, demonstrando todos os passos utilizados para implementar o editor de formulários. Na primeira secção do capítulo são apresentados os pormenores de implementação relativos ao desenvolvimento da interface enquanto que a segunda secção foca a parte dos serviços e no seu desenvolvimento.

Depois do desenvolvimento de uma solução, é necessário levar a cabo experiências e testes que comprovem a qualidade e validem a solução. No capítulo seguinte são apresentados todos as experiências e testes implementados assim como demonstrada a análise de resultados.

Capítulo 7

Experimentação e Avaliação

Depois de desenvolver qualquer produto é necessário avaliar o mesmo de modo a perceber se os problemas identificados foram mitigados assim como se a solução produzida tem qualidade suficiente. É expectável que existam avaliações ao longo do processo de desenvolvimento mas existem algumas grandezas que apenas são passiveis de ser avaliadas quando o produto está desenvolvido. Este capítulo tem como objetivo avaliar a solução desenvolvida tanto a nível técnico como a nível de usabilidade definindo experiências e testes e analisando os resultados dos mesmos.

7.1 Experiências e Testes

Uma experiência ou teste consiste na medição ou avaliação de uma grandeza, utilizando uma metodologia específica de modo a validar ou refutar uma hipótese. O primeiro passo para um teste é a definição das grandezas a avaliar, das hipóteses e das metodologias de avaliação.

7.1.1 Grandezas a Avaliar

Os problemas identificados na Secção 1.3 e na Secção 4.1.1 têm duas naturezas distintas, uns estão relacionados com a qualidade do *software* e outros relacionados com a usabilidade do sistema. Por este motivo, as grandezas a avaliar dividem-se nos resultados dos testes de *software* e nos resultados dos testes de usabilidade feitos a cada editor. A avaliação destas grandezas permite confirmar se os objetivos foram atingidos e, para isso, é necessária a definição das hipóteses sobre as grandezas definidas.

7.1.2 Hipóteses

Uma hipótese consiste num enunciado que se quer provar através de testes estatísticos, utilizando grandezas identificadas [61]. No âmbito do projeto, foram definidas duas hipóteses que refletem a resolução dos problemas identificados sendo elas o sucesso de 100% dos testes de *software* assim como o aumento da usabilidade do novo editor de formulários do E-goi de 25% face ao editor antigo.

A primeira hipótese, se verificada, garante que o código desenvolvido para o novo editor de formulários tem a qualidade suficiente para ser integrado no produto, garantindo a fácil manutenção do mesmo assim como a aplicação de boas práticas de engenharia informática

e desenvolvimento de *software*. A segunda hipótese verifica que aconteceu o aumento de usabilidade esperado.

Depois de definidas, as hipóteses são testadas utilizando metodologias de avaliação de modo a serem confirmadas ou refutadas.

7.1.3 Metodologias de Avaliação

As metodologias de avaliação consistem no modo como se verificarão, ou não, as hipóteses definidas [62]. Existem diversas metodologias de avaliação que dependem das hipóteses e das grandezas, sendo necessário escolher as metodologias mais adequadas para uma melhor obtenção de resultados.

A qualidade técnica da solução é medida através de testes de *software*, nomeadamente testes unitários e testes de integração que são executados através de ferramentas específicas para o efeito. Esta metodologia é direta e não necessita de tratamento estatístico ou qualquer tipo de estudo.

Relativamente à usabilidade do novo editor, é possível perceber que é difícil quantificar esta grandeza, uma vez que o conceito de usabilidade é baseado na experiência pessoal de cada individuo e não existe uma escala para o definir. No entanto, é possível obter dados quantitativos através da implementação de testes de usabilidade sendo usual a utilização desse tipo de dados para testes estatísticos. Para provar a hipótese relativa à usabilidade serão feitos testes de hipóteses [63] para calcular a diferença de usabilidade de ambos os editores e perceber assim a margem de crescimento da usabilidade do novo editor de formulários.

7.2 Resultados

O culminar da realização de testes para avaliar hipóteses definidas é a análise de resultados. É através da análise de resultados que é validada a solução apresentada sendo possível averiguar se os objetivos foram atingidos, o trabalho futuro e outras questões que necessitem de atenção ou trabalho adicional.

7.2.1 Testes de Software

Para avaliar a qualidade do *software* desenvolvido, foram utilizadas duas ferramentas para testar os componentes assim como a qualidade do código desenvolvidos. As ferramentas utilizadas foram a funcionalidade de testes da *framework* Angular [64] assim como a plataforma SonarQube [45], apresentadas na Secção 3.2.2 e na Secção 3.2.2, respetivamente.

A funcionalidade de testes do Angular permite implementar testes unitários recorrendo à *framework* Jasmine [44]. Cada componente gerado para o novo editor de formulários conta com um ficheiro de testes com a extensão *.spec.ts* através do qual o Angular inicia os testes unitários. Cada ficheiro contém a definição do teste assim como os valores de entrada e os valores de saída.

```
1 describe('Component: Input-Text Component', () => {
2     beforeEach(() => {
3         TestBed.configureTestingModule({
```

```
4             imports: [BootstrapModule, MaterialModule, HttpModule,
    MatDialogModule, BrowserAnimationsModule]
5             declarations: [InputTextComponent],
6         });
7     });
8
9     it('should create: InputTextComponent', () => {
10        const fixture = TestBed.createComponent(InputTextComponent);
11        const inputText = fixture.debugElement.componentInstance;
12        expect(inputText).toBeTruthy();
13    });
14
15    it('should have type 'input-text'', () => {
16        const fixture = TestBed.createComponent(InputTextComponent);
17        const inputText = fixture.debugElement.componentInstance;
18        expect(inputText.type).toEqual('input-text');
19    });
20});
```

Listing 7.1: Exemplo Teste Unitário

O teste apresentado no extrato de código cria um componente *InputTextComponent*, importando todas as dependências deste componente na função *beforeEach*. Seguidamente o componente é instanciado, é testado se o tipo do componente corresponde ao valor correto, neste caso "input-text". Este teste é replicado para todos os componentes gerados de modo a testar a criação dos mesmos e a atribuição de valores, garantindo assim que estes funcionam como esperado.

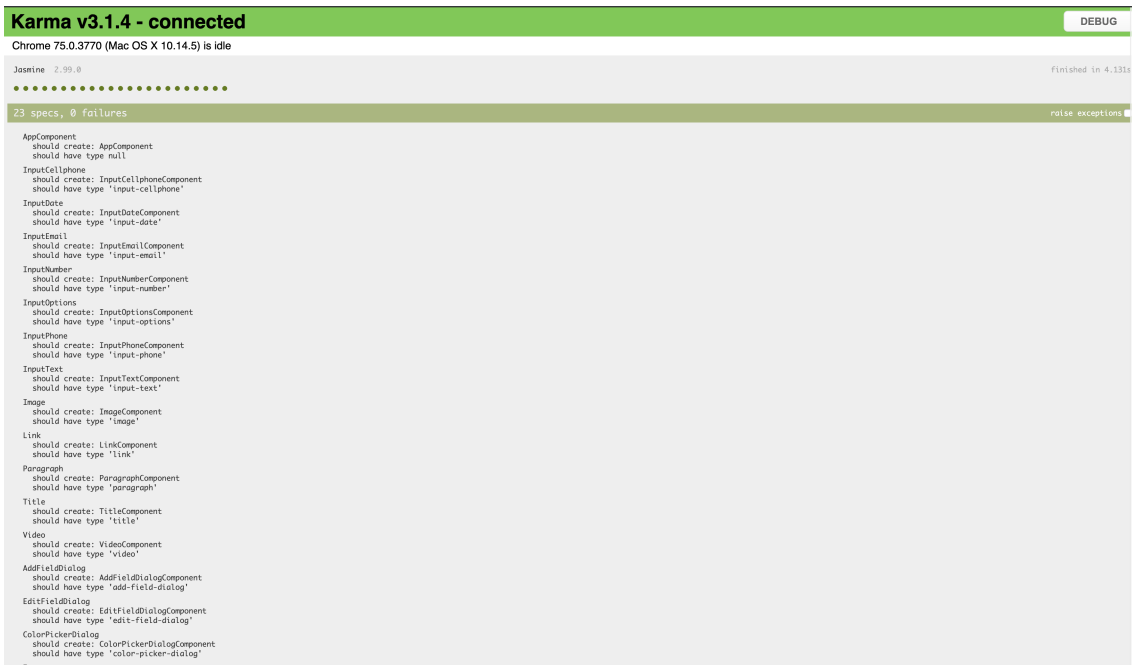

Figura 7.1: Resultados Testes Unitários

Como é possível perceber pela figura 7.1, podemos perceber que todos os componentes gerados foram testados e que todos os testes foram bem sucedidos, garantindo assim a funcionalidade esperada para cada componente.

Para testar a qualidade do código, foi utilizada a *framework* SonarQube. Os testes são feitos ao código existente no repositório de uma forma automática sendo gerado um relatório. O relatório contém informações relevantes sobre o código tais como *bugs* e vulnerabilidades, *code smells* [1] assim como duplicação de código. Para além disso, também conta com informação sobre o projeto como o número total de linhas e a percentagem de linguagens de programação presentes no projeto.

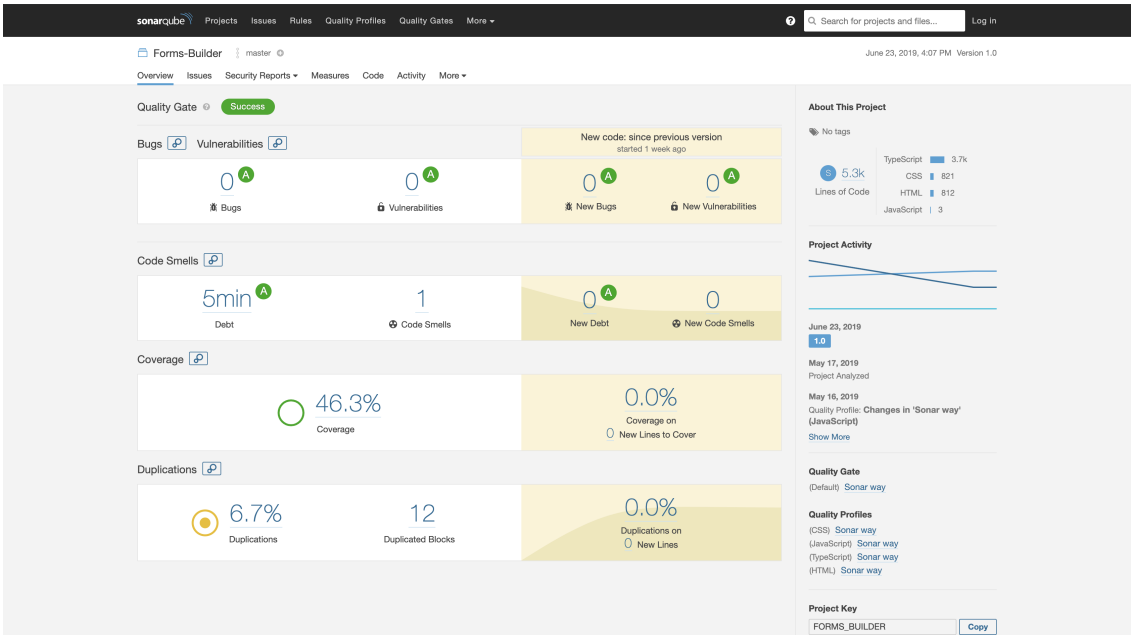

Figura 7.2: Resultados SonarQube

Como é possível constatar na figura 7.2, no relatório executado para este projeto, é possível verificar que o código desenvolvido está livre de *bugs*, apenas contém um caso que pode ser considerado má prática de desenvolvimento de *software* e possui alguma repetição de código. É importante reduzir ao máximo a repetição de código através do uso de funções genéricas melhorando assim a qualidade do mesmo e possivelmente melhorando a performance geral do sistema.

É importante manter mecanismos de revisão de código e testes unitários de modo a garantir a qualidade do *software*, permitindo assim a adição de funcionalidades de uma forma simplificada assim como facilitar a manutenção do código. Desta forma, as próximas equipas que precisem de interagir com o projeto têm o seu trabalho facilitado uma vez que o código é bem estruturado e de qualidade.

7.2.2 Testes de Usabilidade

Como já referido, de modo a conseguir fazer uma avaliação das melhorias do novo editor, foi realizado o mesmo teste de usabilidade ao editor EasyForms e ao novo editor. O teste relativo ao novo editor foi aplicados à mesma população do teste relativo ao editor EasyForms, para ser possível perceber a evolução do editor sem interferências externas. Todos os participantes testaram o editor EasyForms, podendo assim ter uma opinião mais consistente

[1] Más práticas de programação.

relativa à evolução do novo editor face ao editor EasyForms. Nas figuras 7.3 e 7.4 são apresentados os resultados do teste feito ao novo editor de formulários, seguindo a mesma lógica aplicada nos resultados do teste ao editor EasyForms (figura 4.1) onde cada coluna representa uma coluna do formulário e as diferentes áreas coloridas representam o número de respostas, legendadas na figura.

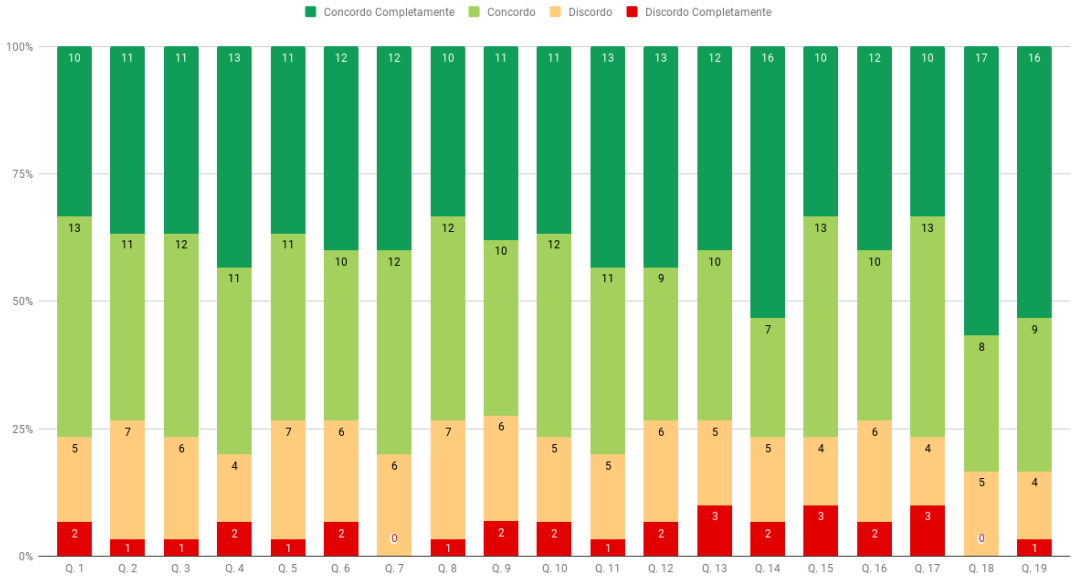

Figura 7.3: Resultados do Teste de Usabilidade ao Novo Editor

1. "O editor é rápido e simples de entender."
2. "Apesar de estar habituado ao editor com drag and drop, foi relativamente simples construir um formulário"
3. "o design e muito melhor mas tenho pena que o modo de fazer o formulario nao seja drag and drop"

Figura 7.4: Comentários do Teste de Usabilidade ao Novo Editor

Analisando os resultados, é possível perceber que de uma forma qualitativa, houve uma melhoria positiva do editor de formulários. Esta análise, comparativamente com os resultados do teste feito ao editor EasyForms, permite perceber em que pontos houve uma maior evolução assim como os pontos onde é necessário continuar o trabalho desenvolvido. Para uma análise mais clara, foi calculado o aumento percentual da média para cada questão, apresentado na tabela 7.1.

Questão	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Aumento	10,98%	21,05%	14,81%	14,46%	16,46%	29,58%	21,52%	18,18%	30,43%	27,78%
Questão	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	
Aumento	20,00%	24,00%	22,87%	24,02%	25,00%	22,67%	38,46%	45,71%	51,52%	

Tabela 7.1: Aumento Percentual da Média das Questões

Relativamente às questões 1 e 3, é possível constatar que a funcionalidade é importante para a população assim como que se tornou mais relevante com a implementação do novo editor, com um aumento da média em cerca de 10%. Este aumento pode traduzir uma maior utilização e necessidade do novo editor face ao editor EasyForms.

A complexidade do editor é avaliada através das questões 2, 4, 7, 9, 10 e 15.

As respostas da questão 2 demonstram um aumento de média de cerca de 21%, o que representa uma diminuição da complexidade do editor. No entanto, aproximadamente 25% da população ainda considera o editor complexo.

Através das respostas às questões 4 e 7, é possível perceber que a maioria dos utilizadores não necessitaria de suporte técnico para usar o sistema assim como a maioria dos novos utilizadores aprenderia a utilizar o sistema de uma forma simples. O aumento da média de aproximadamente 14% e 21% destas questões demonstram uma simplificação do sistema.

Relativamente à questão 9, percebe-se um aumento de confiança na utilização do sistema comprovado com o aumento de cerca de 30% da média das respostas. As respostas à questão 10 mostram que menos de 25% dos participantes considerou ter de aprender novos conceitos para a utilização do sistema, com um ligeiro aumento da média das respostas.

Relativamente à questão 15, as respostas permitem perceber que a grande maioria dos participantes considerou fácil realizar tarefas no sistema.

As respostas a estas questões permitem perceber uma diminuição da complexidade do sistema assim como um aumento de confiança na utilização do mesmo, estimulando assim a fidelização do utilizador e o incentivo a voltar a usar o sistema.

As questões 5, 6, 8, 11 e 12 estão relacionadas com a consistência do editor, tanto relativa às funcionalidades como ao uso geral do mesmo.

A questão 5 apresenta um aumento da média das respostas de cerca de 16% provando assim que a população considera que existiu uma melhoria na integração das funcionalidades do sistema ainda que cerca de 25% da população tenha uma opinião contrária.

A questão 6 encontra-se relacionada com a consistência do sistema e as suas respostas permitem perceber que a maioria dos participantes considera o sistema consistente tendo um aumento na média das respostas de cerca de 29%.

As respostas à questão 8 permitem perceber que a maioria dos participantes considera que o sistema não é confuso. Relativamente às questões 11 e 12, pode concluir-se que os participantes sentiram que tiveram controlo sobre o sistema (com um aumento de 20% de respostas positivas) e que conheceram o seu estado (com um aumento de 24% de respostas positivas). As respostas a estas questões demonstram um aumento da consistência do editor assim como das suas funcionalidades. A consistência de um sistema é essencial para que os seus utilizadores sintam confiança a usar e não tenham receio de depender do mesmo.

O design do editor, assim como a interação com o utilizador são avaliadas através das respostas às questões 18 e 19.

A questão 18 está relacionada com a interface gráfica do sistema e os resultados revelam um aumento da média das respostas de 45%, traduzindo assim a satisfação dos participantes relativamente à interface do editor.

As respostas da questão 19 permitem concluir que o novo editor é mais rápido e responsivo através do aumento da média de respostas de cerca de 51%. Um aumento na rapidez e resposta do editor traduz uma melhor experiência para o utilizador final incentivando-o a reutilizar o sistema.

As questões 13, 14, 16 e 17 estão relacionadas com a recuperação de erros feitos pelo utilizador.

As respostas à questão 13 revelam um aumento na eficiência do sistema relativamente à prevenção de erros através de um aumento da média das respostas de cerca de 23%.

A questão 14 está relacionada com a informação que o utilizador necessita de memorizar para usar o sistema de uma forma eficiente sendo que se constata uma melhoria de cerca de 24% na média das respostas da questão.

As respostas da questão 16 demonstram que a maioria dos participantes considera o sistema bem documentado.

Por último, as respostas da questão 17 revelam que o sistema é eficiente a fornecer *feedback* ao utilizador quando este comete algum tipo de erro. A prevenção e correção de erros é uma parte importante de qualquer sistema uma vez que um erro não entendido pelo utilizador pode levá-lo a desistir do sistema.

De uma forma global, constata-se que existiram diversas melhorias com a implementação do novo editor a nível de usabilidade, quer a nível de interação com o utilizador como ao nível da consistência do sistema.

Através das respostas de ambos os questionários, é possível fazer cálculos que demonstrem o crescimento médio da usabilidade do editor.

Uma vez que todas as respostas do questionário variam numa escala igual (1 a 4) e todas as respostas têm uma ponderação igual, é possível descobrir a média de crescimento através dos valores médios de crescimento por questão.

Somando todos os valores das médias de cada questão e dividindo pelas 19 questões o resultado consiste num crescimento médio de 25,32%. Este crescimento é coincidente com um dos objetivos propostos na Secção 1.4.

Para testar estatisticamente o crescimento obtido, é necessário realizar um teste de hipóteses.

Uma das hipóteses definidas no âmbito deste projeto consiste no aumento de 25% da usabilidade geral do sistema. O teste vai permitir refutar ou validar a hipótese estatística. Para todos os testes realizados é assumido um grau de confiança de 95%.

Será feito um teste comparativo entre duas amostras sendo elas as médias das respostas do questionário ao editor EasyForms e as médias das respostas do questionário ao novo editor. O teste adequado seria Two Sample Independent T-Test [65]. No entanto, este teste tem pré-requisitos para ser utilizado de forma correta sendo um deles a homogeneidade de variância das amostras. Através do teste F-Test [66], é possível perceber que existe uma diferença significativa na variância das duas amostras e, por isso, não se pode confirmar a homogeneidade das mesmas. Esta conclusão é obtida através da análise do $p - value = 0,06345$. Como $p - value$ é maior que $0,05$, não é possível aferir a homogeneidade das variâncias.

Uma vez que os pré-requisitos para a realização do T-Test não foram garantidos, utilizou-se uma variante não paramétrica deste teste, nomeadamente o Two Sided Sign-Test [67] onde $\mu_{EasyForms}$ traduz as médias das respostas dos resultados do teste realizado ao editor EasyForms e μ_{novo} traduz a média das respostas dos resultados do teste realizado ao novo editor.

$$H_0 : \mu_{EasyForms} <= \mu_{novo}$$
$$H_1 : \mu_{novo} > \mu_{EasyForms}$$

Feito o teste, é obtido o $p - value = 3,047e^{-14}$, logo a hipótese H_0 é rejeitada (uma vez que o $p - value < 0,05$), provando assim que a média das respostas do questionário de usabilidade do novo editor é superior à média de respostas do questionário relativo ao editor EasyForms. Para perceber se a margem de crescimento foi de 25%, é necessário calcular o valor de crescimento médio relativamente ao editor EasyForms que corresponde a 25%. A média total de respostas dos questionários de usabilidade do editor EasyForms é de 2,5. Calculando $2,5 * 0,25$, é obtido o resultado de 0,625, obtendo assim o valor de 25% da média dos resultados.

Fazendo um novo teste de hipóteses para testar o crescimento, é novamente usado um Sign Test onde $\mu_{EasyForms}$ traduz as médias das respostas dos resultados do teste realizado ao editor EasyForms e μ_{novo} traduz a média das respostas dos resultados do teste realizado ao novo editor.

$$H_0 : \mu_{novo} - \mu_{EasyForms} <= 0,625$$
$$H_1 : \mu_{novo} - \mu_{EasyForms} > 0,625$$

Feito o teste, é obtido o $p - value = 0.9165$, logo a hipótese H_0 não pode ser rejeitada. Através do teste é também obtido o crescimento mínimo, para um intervalo de confiança de 95%, tendo sido obtido o valor 0,533. Calculando o valor de percentagem, é possível garantir estatisticamente que a usabilidade do novo editor é 21,32%. A diferença de cerca de 4% para o objetivo pode ser justificada pelo uso de médias e a possível existência de *outliers* nas respostas dos questionários ou pelo facto de efetivamente não ter sido alcançado o objetivo.

7.3 Sumário

Neste capítulo são apresentadas todas as formas de avaliação do projeto para provar que o mesmo colmata os problemas identificados. Através desta avaliação é possível perceber os resultados do trabalho realizado assim como tirar conclusões e definir trabalho futuro. Foi realizado um teste estatístico para confirmar a evolução do formulário assim como apresentados os resultados dos testes ao *software* desenvolvido.

Depois de validada a solução, é importante tirar conclusões da mesma assim como definir o trabalho futuro para de modo a melhorar ou/e a continuar o projeto. No capítulo seguinte são apresentadas as conclusões do projeto assim como apresentado o trabalho futuro do mesmo.

Capítulo 8

Conclusões

Como apresentado no Capítulo 1, foi seguida uma metodologia para o projeto baseada no artigo *"Design Science Research Methodology"* [7]. A metodologia proposta para o projeto contém diversas etapas desde o planeamento, com a definição de objetivos baseados em problemas identificados, assim como o desenvolvimento e implementação de uma solução que atingisse os objetivos até à definição e implementação de testes que validassem a solução. Através dos testes e avaliação é possível perceber se o projeto foi bem sucedido, se cumpre com os requisitos definidos e se resolve os problemas identificados.

8.1 Objetivos Alcançados

Na fase inicial do projeto, foram definidos objetivos para o mesmo sendo necessário avaliar se estes foram alcançados. Os objetivos definidos no âmbito deste projeto encontram-se definidos na Secção 1.4 e são os seguintes:

1. Identificar problemas no atual editor de formulários;
2. Identificar soluções para os problemas identificados;
3. Definir uma arquitetura que satisfaça os problemas identificados;
4. Implementar a arquitetura definida;
5. Integrar o sistema na plataforma E-goi;
6. Testar a solução e avaliar o seu desempenho;

O objetivo número 1 foi atingindo através de testemunhos vindos de diferentes elementos das várias equipas da E-goi assim como através da implementação de testes de usabilidade ao editor de formulários EasyForms, apresentados na Secção 4.1.1. Estas duas abordagens, aliadas entre si, permitiram identificar os seguintes problemas:

1. Complexidade para os utilizadores;
2. Arquitetura e tecnologias desatualizadas;
3. Acoplamento entre funcionalidade de formulários e *landing pages*;
4. Falta de funcionalidades;
5. Falta de métricas para análise de resultados.

O objetivo número 2 foi atingido identificando uma solução para cada problema identificado. O problema número 1 foi resolvido através de uma interface mais amigável e intuitiva para os utilizadores, apresentada na Secção 5.5. A solução para o problema número 2 consistiu na escolha da *framework* Angular, apresentada na Secção 3.2.2, assim como no desenvolvimento de uma nova arquitetura escalável e de fácil manutenção, apresentada no Capítulo 5. O problema número 3 foi eliminado através da separação dos conceitos de formulários e *landing pages* em entidades distintas e independentes como apresentado no Capítulo 5. A abordagem para o problema número 4 consistiu na identificação de funcionalidades e a sua implementação como é possível perceber ao longo de todo o Capítulo 5. Por último, o problema número 5 foi resolvido através da nova arquitetura que permite gerar métricas individuais e específicas quer para formulários quer para *landing pages*.

O objetivo número 3 foi atingido através do desenvolvimento de todo o processo descrito no Capítulo 5 quer no desenvolvimento da arquitetura para o desenvolvimento de *software*, quer da arquitetura física das máquinas envolvidas no projeto assim como no desenvolvimento da nova interface e alteração dos serviços existentes no produto E-goi.

A abordagem para o objetivo número 4 consistiu na implementação do sistema seguindo a arquitetura desenvolvida através das linguagens e tecnologias definidas e apresentadas no Capítulo 3. O desenvolvimento contou com o uso de boas práticas de engenharia informática e desenvolvimento de *software* como demonstrado ao longo do Capítulo 6. A utilização de boas práticas simplifica o trabalho de manutenção que será necessário para manter o projeto funcional.

Relativamente ao objetivo número 5, foi feita uma integração continua (apresentada na Secção 3.2.2) do projeto na plataforma E-goi garantindo assim compatibilidade do sistema ao longo de todo o desenvolvimento e fase de testes. Desta forma, ficando assim simplificada a implantação do projeto dentro do produto E-goi.

Por último, o objetivo número 6 foi atingido recorrendo a várias técnicas, definidas no Capítulo 7. Estas técnicas consistiram na implementação de testes de *software* assim como na realização de testes de usabilidade ao novo editor de formulários (apresentada na Secção 7.2.2 cujos resultados foram comparados com os resultados do mesmo teste para o editor atual. Os resultados foram analisados empiricamente e estatisticamente de modo a perceber e quantificar a evolução existente e validar todas as hipóteses definidas.

Apesar de todos os objetivos terem sido alcançados, apenas se poderá comprovar o sucesso do projeto com a utilização do editor por utilizadores reais. As populações dos testes (definida na Secção 4.1.1 desenvolvidos são apenas uma representação dos reais utilizadores do produto E-goi e existem métricas que apenas poderão ser obtidas após o lançamento do novo editor para os clientes E-goi. Um desses exemplos, referido no Capítulo 1 é a diminuição de *tickets* relativos ao editor de formulários provenientes dos clientes e utilizadores do produto E-goi. Através dos testes implementados, espera-se uma diminuição do número de pedidos de suporte mas este facto apenas poderá ser comprovado com a utilização real do sistema.

Através dos testes realizados no Capítulo 7, foi possível perceber um crescimento de, pelo menos, 21% relativo à usabilidade. Esta é uma das maiores provas do valor, identificado na Secção 4.2, que o novo editor traz ao produto E-goi, quer relativamente ao valor para o cliente como o valor para a empresa.

Os objetivos identificados foram atingidos, os problemas existentes foram colmatados e os testes implementados validam a solução e, por estes motivos, o projeto revela-se bem sucedido.

8.2 Trabalho Futuro

Apesar do sucesso geral do projeto e da sua implementação, existem tarefas a executar de modo a concluir definitivamente o projeto e a conseguir manter o mesmo.

Como já referido, é essencial que o editor seja testado por utilizadores reais de modo a fazer uma avaliação continua do mesmo e melhorá-lo de acordo com essa informação. Se um produto ficar completamente estático durante o seu percurso de vida, eventualmente acabará por fracassar. O sistema foi desenvolvido seguindo boas práticas de desenvolvimento de *software* de modo a que seja simples a adição de funcionalidade assim como a manutenção.

Uma vez que o crescimento comprovado foi apenas de 21 %, é importante continuar a trabalhar para que esta percentagem aumente. O trabalho deve ser contínuo, analisando o *feedback* dos utilizadores, através de *tickets* ou outras formas. Um investimento contínuo na funcionalidade aumentará o valor do produto e garantirá a fácil manutenção do mesmo durante mais tempo.

Existem funcionalidades que não foram implementadas, tais como a possibilidade de paginação do formulário e campos com lógica condicional. Outra funcionalidade que pode ser adicionada é a ajuda na construção do formulário através de dicas e informação relevante, como por exemplo a percentagem de respostas do formulário consoante o número de campos presentes no mesmo. Estas funcionalidades não foram implementadas por falta de tempo e por opção da empresa. A sua futura implementação vai adicionar valor ao produto e ao cliente final e é importante que sejam desenvolvidos.

É também necessário perceber, quando for disponibilizado para os utilizadores finais, o nível de uso do editor de modo a conseguir escalar de uma forma adequada, alocando recursos físicos de um modo dinâmico. Esta alocação de recursos dinâmica poupa recursos que possam não estar a ser utilizados assim como alerta para uma necessidade do aumento de instâncias do editor a funcionar.

Por último, implícito a qualquer *software*, o sistema vai precisar de manutenção contínua, quer seja a correção de erros como o aumento de funcionalidades. O projeto integrará uma das equipas da E-goi que o manterá, seguindo o fluxo atual de desenvolvimento e integração contínuos.

Bibliografia

[1] Google. *Introduction to modules*. https://angular.io/guide/architecture-modules. Acedido em Outubro 2018.

[2] Google. *Introduction to components*. https://angular.io/guide/architecture-components. Acedido em Outubro 2018.

[3] Google. *Introduction to services and dependency injection*. https://angular.io/guide/architecture-services. Acedido em Outubro 2018.

[4] Jennifer Rowley. «Just another channel? Marketing communications in e-business». Em: *Marketing Intelligence and Planning* 22 (2004), pp. 24–41.

[5] Kabu Khadka e Soniya Maharjan. «CUSTOMER SATISFACTION AND CUSTOMER LOYALTY». Em: (nov. de 2017).

[6] E-goi. *E-goi*. http://www.e-goi.pt. Acedido em Setembro 2018. 2003.

[7] Desmond Bisandu. «Design Science Research Methodology in Computer Science and Information Systems». Em: (nov. de 2016).

[8] Thomas L. Saaty. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. McGraw-Hill, 1980.

[9] Emailmanager. *O que significa "opt in", "soft opt in" e double opt in" no email marketing?* https://www.emailmanager.com/pt/blog/1/1425/o-que-significa-opt-in-soft-opt-in-e-double-opt-in-no-email-marketing.html. Acedido em Outubro 2018.

[10] Oli Gardner. «The Ultimate Guide To Landing Page Optimization». http://unbounce.com/photos/ultimate-guide-lpo.pdf. Acedido em Dezembro 2018. 2012.

[11] Jay Chambers. *Email Landing Page: Time To Be DEBT FREE!* https://www.flickr.com/photos/jaychambers/4735120621. Acedido em Novembro 2018. 2010.

[12] The PHP Group. *PHP*. https://php.net/. Acedido em Outubro 2018.

[13] MariaDB Foundation. *About MariaDB*. https://mariadb.org/about/. Acedido em Outubro 2018.

[14] Oracle. *About MySQL*. https://www.mysql.com/about/. Acedido em Outubro 2018.

[15] The Rocket Science Group. *Mailchimp*. http://www.mailchimp.com. Acedido em Outubro 2018. 2001.

[16] 123FormBuilder. *123FormBuilder*. http://www.123formbuilder.com. Acedido em Outubro 2018. 2008.

[17] Formstack. *Formstack*. http://www.formstack.com. Acedido em Outubro 2018. 2006.

[18] SendPulse. *SendPulse*. http://www.sendpulse.com. Acedido em Outubro 2018. 2015.

[19] JotForm. *JotForm*. http://www.jotform.com. Acedido em Outubro 2018. 2006.

[20] SurveyMonkey. *Wufoo*. http://www.wufoo.com. Acedido em Outubro 2018. 2006.

[21] Google. *Google Forms*. http://www.google.com/forms. Acedido em Outubro 2018. 2012.

[22] Paul Belliveau, Abbie Griffin e Stephen Somermeyer. *The PDMA ToolBook 1 for New Product Development*. New York: John Wiley e Sons, 2004, pp. 5–35.
[23] Thomas L. Saaty. *Decision Making With the Analytic Hierarchy Process*. Vol. 1. 2008, pp. 83–98.
[24] Alexander Osterwalder. «The Business Model Ontology - A Proposition In A Design Science Approach». Tese de doutoramento. Ecole des Hautes Etudes Commerciales, 2004.
[25] C.M. Barnum. *Usability Testing Essentials: Ready, Set...Test!* Elsevier Science, 2010.
[26] Jakob Nielsen. *Recruiting Test Participants for Usability Studies*. `https://www.nngroup.com/articles/recruiting-test-participants-for-usability-studies/`. Acedido em Fevereiro 2019. 2003.
[27] Deborah Hinderer Sova e Jakob Nielsen. «234 Tips and Tricks for Recruiting Users as Participants in Usability Studies». `https://media.nngroup.com/media/reports/free/How_To_Recruit_Participants_for_Usability_Studies.pdf`. Acedido em Fevereiro 2018. 2013.
[28] J. Nielsen. *Usability Engineering*. Elsevier Science, 1994.
[29] Suzanne Robertson e James Robertson. *On Non-Functional Requirements in Software Engineering*. Springer, 2012.
[30] Chung L. e do Prado Leite J.C.S. *Mastering the Requirements Process: Getting Requirements Right*. Addison-Wesley, 2009.
[31] Mike Cohn. «Advantages of User Stories for Requirements». Em: *InformIT Network* (out. de 2004).
[32] Robert Bogue. «Use S.M.A.R.T. goals to launch management by objectives plan». Em: *TechRepublic* (fev. de 2018).
[33] Martin Glinz. «On Non-Functional Requirements». Em: *15th IEEE International Requirements Engineering Conference* (out. de 2007).
[34] R. B. Grady. *Practical software metrics for project management and process improvement*. Prentice Hall, 1992.
[35] uml-diagrams. *The Unified Modeling Language*. `https://www.uml-diagrams.org/`. Acedido em Março 2019.
[36] Grady Booch, James Rumbaugh e Ivar Jacobson. *Unified Modeling Language User Guide, The, 2nd Edition*. Addison-Wesley Professional, 2005.
[37] Scott Millett e Nick Tune. *Software Engineering 3*. John Wiley e Sons, 2015.
[38] Google. *AngularJS*. `https://angularjs.org/`. Acedido em Março 2019.
[39] Microsoft. *TypeScript*. `https://www.typescriptlang.org/`. Acedido em Março 2019.
[40] Google. *Angular Material*. `https://material.angular.io/`. Acedido em Dezembro 2018.
[41] Google. *Introduction to services and dependency injection*. `https://angular.io/guide/architecture-services`. Acedido em Outubro 2018.
[42] git-scm. *Git*. `https://git-scm.com/`. Acedido em Outubro 2018.
[43] Paul Duvall. *Continuous Integration*. Pearson Education, 2007. isbn: 9788131722916. url: `https://books.google.pt/books?id=YPlgL1aILqIC`.
[44] Jasmine. *Jasmine Framework*. `https://jasmine.github.io/`. Acedido em Março 2019.
[45] SonarQube. *SonarQube Documentation*. `https://docs.sonarqube.org/latest/`. Acedido em Março 2019.
[46] Jose Nicolas e Jose Cardona Mora. «CONTINUOUS IMPROVEMENT STRATEGY». Em: *European Scientific Journal* 10 (dez. de 2014), pp. 117–126.

[47] I. Elaine Allen e Christopher A. Seaman. «Likert Scales and Data Analyses». Em: (jul. de 2007).
[48] P.W. Jordan et al. *Usability Evaluation In Industry*. Taylor & Francis, 1996.
[49] Google. *Angular*. `https://angular.io/`. Acedido em Janeiro 2019. 2010.
[50] P. Drucker e J. Maciariello. *Innovation and Entrepreneurship*. London: Routledge, 2015.
[51] Michael G. Jacobides, Thorbjørn Knudsen e Mie Augier. «Benefiting from innovation: Value creation, value appropriation and the role of industry architectures». Em: *Research Policy* 35 (out. de 2006), pp. 1200–1221.
[52] P. Smith e D. Reinertsen. *Developing Products in Half the Time*. Vol. 12. MCB UP Ltd, 1991, pp. 18–22.
[53] P. Kotler e K. L. Keller. *Creating Customer Value, Satisfaction, and Loyalty*. Pearson Education, 2009.
[54] M.E. Porter. *Competitive Advantage: Creating and Sustaining Superior Performance*. Free Press, 1985.
[55] Isabel Soares Silva, Ana Veloso e José Keating. «Focus group: Considerações teóricas e metodológicas». Em: *Revista Lusofona de Educacao* (set. de 2014).
[56] GitLab. *GitLab*. `https://about.gitlab.com/`. Acedido em Outubro 2018.
[57] Jenkins. *Jenkins*. `https://jenkins.io/`. Acedido em Outubro 2018.
[58] NgRx. *What is NgRx?* `https://ngrx.io/docs`. Acedido em Outubro 2018.
[59] Google. *Observables*. `https://angular.io/guide/observables`. Acedido em Outubro 2018.
[60] Zend. *About Zend Framework*. `https://framework.zend.com/about`. Acedido em Março 2019.
[61] J. Neyman e E. S. Pearson. «On the Problem of the Most Efficient Tests of Statistical Hypotheses». Em: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 231 (1933).
[62] H. Cramér. *Mathematical Methods of Statistics*. Princeton Landmarks in Mathematics and Physics. Princeton University Press, 1999.
[63] Rafael Izbicki. «Classes de Testes de Hipóteses». Em: (jun. de 2010).
[64] Google. *Angular Testing*. `https://angular.io/guide/testing`. Acedido em Março 2019.
[65] Gopal K. Kanji. *100 STATISTICAL TESTS 3rd Edition*. Sage Publications, 2006.
[66] Thomas K. Tiemann. *Introductory Business Statistics*. 2010.
[67] Christian Akrong Hesse, Ezekiel Nortey e John Ofosu. *INTRODUCTION TO NON-PARAMETRIC STATISTICAL METHODS*. Jan. de 2018.

Apêndice A

Teste de Usabilidade – E-goi: Editor de Formulários

Este teste de usabilidade tem como objetivo perceber quais os principais pontos fortes e pontos fracos do atual sistema de criação e edição de formulários. Cada sessão deverá ser feita num ambiente isolado de distrações.

Caracterização do Participante:

Número do Participante:
Idade:
Já utilizou o E-goi?:

Explicação da Sessão:

Espera-se que o participante comunique as suas dúvidas e linhas de pensamento para garantir que está a conseguir (ou não) completar o teste de uma forma fundamentada e não por tentativa erro ou outro método não fundamentado. Também é expectável que o moderador não ajude, de forma nenhuma, o participante para que o teste seja isento e que aponte as questões que o participante levanta durante o teste.

O objetivo da primeira tarefa é focar as vantagens e limitações do editor de formulários e, por esse motivo, todas as tarefas adicionais que seriam obrigatórias (criar uma lista, definir campos extra numa lista, etc) já se encontram definidas previamente. Isso é explicado ao participante na descrição das tarefas propostas.

O objetivo das restantes tarefas é perceber se as funcionalidades e localização das opções do formulário de edição e remoção são claras para o utilizador sendo que não é necessário qualquer tipo de setup para esta tarefa.

Estrutura da Sessão:

Explicação e motivação para este teste de usabilidade: aprox. 2 minutos;
Explicação da tarefa proposta: aprox. 5 minutos;
Execução das tarefas: aprox. 25 minutos;
Questionário: aprox. 15 minutos

Tempo total: aprox. 50 minutos

Tarefas Propostas:

1. Crie um novo formulário de inscrição com o layout "Large", escolhendo a lista "teste usabilidade" e a língua portuguesa ("Portuguese"). Espera-se que experimente esta ferramenta livremente apesar da utilização dos seguintes widgets ser obrigatória: Name, Email, Date, Option List, Three Columns e Button. Pode utilizar qualquer outro widget livremente e alterar as suas propriedades. Os campos do formulário Name, Email, Date devem estar devidamente mapeados com os campos respetivos da lista. Deve também abrir as opções do formulário no canto superior esquerdo e testar livremente as mesmas. Quando terminar, publique o formulário através de um QR code.

2. Depois de preencherem um formulário de remoção (formulário de unsubscribe), os utilizadores podem receber um email com uma mensagem escrita por si. Ative essa opção e altere o Assunto (Subject) do e-mail assim como a mensagem (Plain Text). A mensagem deverá apresentar a seguinte estrutura: "Caro [nome completo do utilizador], foi removido da nossa lista com o nome [nome da lista]. Foi bom ter contado com o seu apoio desde [data da subscrição do utilizador] e esperamos que volte no futuro.

3. Quando os utilizadores carregam no link de uma campanha para serem removidos da lista, são reencaminhados para uma página que contém um formulário de feedback. Altere a questão que é apresentada aos utilizadores para "Caro [primeiro nome do utilizador], a sua opinião é fundamental, indique o motivo pelo qual se quis remover da nossa lista.". Seguidamente adicione dois motivos: "Já não tenho interesse" e "Inscrevi-me por engano".

Nas tarefas 2 e 3, é suposto que o participante introduza o código que apresente a informação dentro dos parênteses retos.

Questionário:

1. Usaria este sistema frequentemente	1	2	3	4
2. Considero o sistema simples no seu todo				
3. Considero o sistema fácil de usar				
4. Considero que não precisaria de suporte técnico para conseguir usar o sistema				
5. Considero que todas as funções do sistema estão bem integradas				
6. Considero que o sistema é consistente				
7. Considero que a maioria das pessoas aprenderia a usar este sistema facilmente				
8. Não considero o sistema confuso				
9. Senti-me confiante a usar o sistema				
10. Considero que não tive de aprender muita coisa antes de conseguir usar o sistema				
11. Senti que soube sempre qual o estado do sistema				
12. Senti que tinha o controlo enquanto usei o sistema				
13. Senti que o sistema previu possíveis erros que eu podia ter feito				
14. Senti que não tive de memorizar muita informação para conseguir usar o sistema de uma forma eficiente				
15. Senti que foi fácil executar as tarefas no sistema				
16. Senti que o sistema estava bem documentado e era auto-explicativo				
17. Senti que quando errei, o sistema me deu feedback de qualidade que me ajudou a ultrapassar o erro				
18. Achei o design do sistema minimalista e apelativo				
19. Achei o sistema suficientemente rápido e responsivo				

Tabela A.1: Análise Comparativa de Ferramentas

Legenda: 1 – Discordo Totalmente 2 – Discordo 3 – Concordo 4 – Concordo Totalmente

Comentário/Sugestão adicional: