

# ISPGAYA

instituto superior politécnico

Licenciatura em Engenharia Informática

Projeto de Engenharia Informática em Contexto Empresarial

3.º ano

Leonor Carneiro Machado

## GoIZAPY

DESENVOLVIMENTO DE PLUGIN DE INTEGRAÇÃO  
ENTRE E-GOI E WHATSAPP

Relatório de estágio orientado pelo Professor Doutor Mário Jorge Dias  
Lousã e apresentada à Escola Superior de Ciência e Tecnologia

Junho de 2025



## Dedicatória

À minha família e ao meu namorado,  
por nunca esperarem que eu fosse sempre forte,  
mas por me mostrarem, mesmo assim,  
que eu era.

*"A força não provém da capacidade física, mas da vontade indomável."*

Mahatma Gandhi

## Agradecimentos

Agradeço à minha família e ao meu namorado, pelo apoio incondicional e constante incentivo ao longo de todo o percurso.

À E-goi, em especial à equipa MMA, pela integração, profissionalismo e espírito colaborativo. Ao gestor de projeto, Alberto Vasconcelos, pela oportunidade de estágio e acompanhamento ao longo da sua concretização. Ao Professor Daniel Alves, pelas sessões de orientação do relatório, ao meu padrinho de estágio, João Silva, pela prontidão e auxílio prestado sempre que necessário, e à Paula Adams, pelo desenvolvimento do protótipo da interface da extensão.

Ao ISPGAYA e, em particular, ao Professor Mário Lousã, pela orientação, disponibilidade e dedicação. Estendo ainda o meu agradecimento a todos os docentes da licenciatura de Engenharia Informática, pelo contributo na minha formação.

Aos colegas, hoje amigos, que contribuíram para tornar esta jornada académica mais leve, partilhada e memorável.

## Resumo

Este relatório apresenta o desenvolvimento da extensão GoiZapy, no contexto do estágio curricular na empresa E-goi. O projeto tem como objetivo integrar o WhatsApp Web com a plataforma E-goi através de uma extensão para o navegador Google Chrome, permitindo a captura automatizada de contactos (nome, número de telefone e email) diretamente do DOM do WhatsApp e a sua sincronização com a API pública da E-goi. A solução viabiliza a criação e atualização de contactos, associação a listas e aplicação de tags que ativam fluxos de automação.

A arquitetura da extensão segue o modelo definido pelo Manifest V3, com separação entre *content scripts* e *service workers*, garantindo isolamento funcional e segurança. O desenvolvimento adota uma abordagem iterativa, suportada por tecnologias como SvelteKit, TypeScript, Vite e o SDK oficial da E-goi.

Os testes asseguram a estabilidade da extensão, a integração funcional com a API da E-goi e a conformidade com os requisitos técnicos estabelecidos.

**Palavras-chave:** WhatsApp Web, E-goi, extensão de navegador, Google Chrome, DOM, API pública, automação de marketing, captura de contactos, Manifest V3, integração de sistemas.

## Abstract

This report presents the development of the GoiZapy extension, carried out as part of a curricular internship at the company E-goi. The project aims to integrate WhatsApp Web with the E-goi platform through a Google Chrome browser extension, enabling the automated capture of contact data (name, phone number, and email) directly from the WhatsApp DOM and its synchronization with E-goi's public API. The solution enables the creation and update of contacts, association with lists, and the application of tags that trigger automation flows.

The extension's architecture follows the model defined by Manifest V3, with a clear separation between content scripts and service workers, ensuring functional isolation and security. Development follows an iterative approach, supported by technologies such as SvelteKit, TypeScript, Vite, and the official E-goi SDK.

Testing confirms the stability of the extension, its effective integration with E-goi's API, and full compliance with the defined technical requirements.

**Keywords:** WhatsApp Web, E-goi, browser extension, Google Chrome, DOM, public API, marketing automation, contact capture, Manifest V3, systems integration.

## **Lista de abreviaturas e siglas**

2FA	Two-Factor Authentication
API	Application Programming Interface
CI/CD	Continuous Integration / Continuous Deployment
CSP	Content Security Policy
DELETE	Hypertext Transfer Protocol DELETE Method
DOM	Document Object Model
FS	Forward Secrecy
GET	Hypertext Transfer Protocol GET Method
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I&D	Investigação e Desenvolvimento
JSON	JavaScript Object Notation
MMA	Marketing Messages & Automation
PA	Product Academy
PATCH	Hypertext Transfer Protocol PATCH Method
PME	Pequenas e Médias Empresas
POST	Hypertext Transfer Protocol POST Method
PUT	Hypertext Transfer Protocol PUT Method
REST	Representational State Transfer
SaaS	Software as a Service
SDK	Software Development Kit
SMS	Short Message Service
SOP	Same Origin Policy
SPA	Single Page Application

TLS	Transport Layer Security
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XSS	Cross-Site Scripting

## Índice

---

Dedicatória .....	1
Agradecimentos .....	2
Resumo .....	3
Abstract .....	4
Lista de abreviaturas e siglas .....	5
1. Introdução .....	13
1.1. Contexto e motivação .....	13
1.2. Objetivos .....	14
1.3. Estrutura do documento .....	15
2. A empresa.....	16
2.1. Apresentação da empresa .....	16
2.2. Missão, visão e cultura organizacional.....	17
2.3. Orientações de integração .....	17
2.4. Estrutura organizacional .....	18
2.5. Equipa MMA – Marketing Messages & Automation.....	19
2.6. Ferramentas digitais e tecnologias de suporte .....	20
3. Projeto GoiZapy .....	21
3.1. Fundamentos teóricos.....	21
3.1.1. WhatsApp.....	21
3.1.1.1. Diferenças entre contas pessoais e comerciais .....	21
3.1.1.2. Modelo client-side .....	22
3.1.1.3. Acesso e manipulação do DOM.....	22
3.1.1.4. Considerações técnicas e limitações de integração .....	23
3.1.2. API da E-goi.....	24
3.1.2.1. Modelo funcional .....	24
3.1.2.2. Arquitetura RESTful.....	24
3.1.2.3. Formato de dados.....	25
3.1.2.4. Autenticação .....	25
3.1.2.5. Comunicação segura .....	25
3.1.2.6. Limites de utilização .....	26

3.1.2.7.	SDK.....	27
3.1.2.8.	Tratamento de respostas.....	27
3.1.2.9.	Recursos utilizados .....	28
3.1.2.10.	Integração com mecanismos de automações.....	28
3.1.3.	Extensões Chrome .....	29
3.1.3.1.	Extensões de navegador .....	29
3.1.3.2.	Histórico e evolução das especificações .....	30
3.1.3.3.	Arquitetura de extensões.....	31
3.1.3.4.	Execução Client-Side e interação com o DOM.....	31
3.1.3.5.	Segurança e permissões .....	32
3.1.3.6.	Considerações éticas e de privacidade .....	33
3.1.3.7.	Limitações do modelo de extensões .....	33
3.2.	Fundamentos práticos.....	34
3.2.1.	Projetos semelhantes.....	34
3.2.2.	Requisitos do sistema .....	35
3.2.2.1.	Requisitos funcionais .....	35
3.2.2.2.	Requisitos não funcionais.....	37
3.2.3.	Metodologia .....	37
3.2.4.	Descrição funcional da extensão .....	38
3.2.4.1.	Login .....	38
3.2.4.2.	Adicionar contacto .....	39
3.2.4.3.	Procurar contacto.....	40
3.2.4.4.	Segmentar contactos .....	41
3.2.4.5.	Gestão de listas de contactos.....	43
3.2.4.6.	Gestão da lista de supressão .....	44
3.2.4.7.	Gestão de Tags.....	46
3.2.5.	Implementação técnica .....	47
3.2.6.	Design da interface da extensão.....	51
3.2.7.	Testes e validações.....	52
3.2.8.	Cronograma e desenvolvimento .....	53
4.	Conclusão .....	55
4.1.	Resultados obtidos .....	55
4.2.	Dificuldades e soluções .....	56
4.3.	Melhorias futuras .....	56
4.4.	Contributo pessoal e aprendizagens .....	57
	Glossário .....	61

Apêndices .....	66
-----------------	----

## Índice de figuras

---

<i>Figura 1 - Fluxograma de Atividades na Plataforma E-goi (adaptado de Reis, 2018, p. 14).....</i>	16
<i>Figura 2 - Estrutura Organizacional da E-goi (Fonte: elaboração própria).....</i>	19
<i>Figura 3 - Fluxo de Comunicação entre Componentes (Fonte: elaboração própria).....</i>	24
<i>Figura 4 - Automação E-goi Acionada por Trigger de Tag (Fonte: plataforma E-goi).....</i>	29
<i>Figura 5 - Diagrama de Casos de Uso da Extensão GoiZapy (Fonte: elaboração própria) .....</i>	36
<i>Figura 6 - Ecrã de Autenticação da Extensão GoiZapy (Fonte: elaboração própria) .....</i>	39
<i>Figura 7 - Menu Principal da Extensão (Fonte: elaboração própria).....</i>	39
<i>Figura 8 - Ecrã de Adição de Contacto com Sincronização via WhatsApp Web (Fonte: elaboração própria).</i>	40
<i>Figura 9 - Ecrã de Procura de Contacto com Resultados Listados (Fonte: elaboração própria).....</i>	41
<i>Figura 10 - Ecrã de Edição de Contacto (Fonte: elaboração própria).....</i>	41
<i>Figura 11 - Ecrã de Segmentação de Contactos e Ações em Massa (Fonte: elaboração própria).....</i>	42
<i>Figura 12 - Confirmação de Ação Aplicada (Fonte: elaboração própria).....</i>	42
<i>Figura 13 - Ecrã Inicial de Gestão de Listas de Contactos (Fonte: elaboração própria) .....</i>	43
<i>Figura 14 - Formulário de Criação de Lista de Contactos (Fonte: elaboração própria) .....</i>	43
<i>Figura 15 - Ecrã de Edição de Lista de Contactos (Fonte: elaboração própria) .....</i>	43
<i>Figura 16 - Ecrã Inicial de Gestão de Listas de Supressão (Fonte: elaboração própria) .....</i>	44
<i>Figura 17 - Formulário para Inserção de Contactos na Lista de Supressão (Fonte: elaboração própria) .....</i>	44
<i>Figura 18 - Ecrã de Remoção de Contactos da Lista de Supressão (Fonte: elaboração própria).....</i>	45
<i>Figura 19 - Ecrã Inicial de Gestão de Tags (Fonte: elaboração própria).....</i>	46
<i>Figura 20 - Formulário de Criação de Tag (Fonte: elaboração própria) .....</i>	46
<i>Figura 21 - Ecrã de Edição de Tag (Fonte: elaboração própria) .....</i>	46
<i>Figura 22 - Diagrama de Componentes da Extensão GoiZapy (Fonte: elaboração própria) .....</i>	49
<i>Figura 23 - Diagrama de sequência da operação de sincronização de contacto na extensão GoiZapy (Fonte: elaboração própria).....</i>	50
<i>Figura 24 - Comparação entre o Protótipo da Interface Desenvolvido em Figma e a Implementação Final da Extensão GoiZapy .....</i>	51
<i>Figura 25 - Cronograma de Desenvolvimento do Projeto Goizapy (Fonte: elaboração própria).....</i>	54

## Índice de apêndices

---

Apêndice 1- Especificação do Caso de Teste para extração de dados do DOM do WhatsApp Web (Fonte: elaboração própria) .....	66
Apêndice 2- Especificação do Caso de Teste para comunicação entre content script e service worker (Fonte: elaboração própria) .....	67
Apêndice 3- Especificação do Caso de Teste para edição de contacto na API da E-goi (Fonte: elaboração própria) .....	68
Apêndice 4- Especificação do Caso de Teste para criação manual de contacto com tags (Fonte: elaboração própria) .....	69
Apêndice 5- Especificação do Caso de Teste para criação automática de contacto com dados do WhatsApp (Fonte: elaboração própria) .....	70
Apêndice 6- Especificação do Caso de Teste para criação de listas de contactos (Fonte: elaboração própria) .....	71
Apêndice 7- Especificação do Caso de Teste para edição e remoção de listas (Fonte: elaboração própria) .....	72
Apêndice 8- Especificação do Caso de Teste para adição à lista de supressão (Fonte: elaboração própria) .....	73
Apêndice 9- Especificação do Caso de Teste para remoção de contactos da lista de supressão (Fonte: elaboração própria) .....	74
Apêndice 10- Especificação do Caso de Teste para criação de tags personalizadas (Fonte: elaboração própria) .....	75
Apêndice 11- Especificação do Caso de Teste para edição e remoção de tags (Fonte: elaboração própria) .....	76
Apêndice 12- Especificação do Caso de Teste para segmentação de contactos e ações em massa (Fonte: elaboração própria) .....	77
Apêndice 13- Especificação do Caso de Teste para saudação e redirecionamento no menu (Fonte: elaboração própria) .....	78
Apêndice 14- Especificação do Caso de Teste para introdução e validação da API Key (Fonte: elaboração própria) .....	79
Apêndice 15- Especificação do Caso de Teste de segurança e permissões (Fonte: elaboração própria) .....	80
Apêndice 16- Especificação do Caso de Teste de desempenho no navegador (Fonte: elaboração própria) .....	81
Apêndice 17- Especificação do Caso de Teste em ambiente real com WhatsApp Web (Fonte: elaboração própria) .....	82
Apêndice 18- Especificação do Caso de Teste de build e empacotamento (Fonte: elaboração própria) .....	83

## Índice de anexos

---

Anexo 1- Proposta de projeto (Fonte: E-goi - equipa de MMA) .....	85
Anexo 2-Protótipo de interface do login (Fonte: prototipagem interna - equipa de MMA) .....	86
Anexo 3-Protótipo de interface do menu (Fonte: prototipagem interna - equipa de MMA) .....	86
Anexo 4-Protótipo de interface do ecrã adicionar contacto (Fonte: prototipagem interna - equipa de MMA) .....	87
Anexo 5- Protótipos de interface dos ecrãs de procura e edição de contacto (Fonte: prototipagem interna - equipa de MMA) .....	88
Anexo 6- Protótipo de interface do ecrã de segmentação de contactos e aplicação de ações em massa (Fonte: prototipagem interna - equipa de MMA) .....	90
Anexo 7- Protótipos de interface dos ecrãs de gestão da lista de supressão (Fonte: prototipagem interna - equipa de MMA) .....	91
Anexo 8- Protótipos de interface dos ecrãs de gestão de listas de contactos (Fonte: prototipagem interna - equipa de MMA) .....	92
Anexo 9- Protótipos de interface dos ecrãs de gestão de tags (Fonte: prototipagem interna - equipa de MMA) .....	93

## 1. Introdução

A primeira secção deste relatório apresenta o enquadramento conceptual e tecnológico do projeto de estágio curricular, focando o problema identificado, os objetivos definidos, a abordagem metodológica adotada e os recursos tecnológicos envolvidos no desenvolvimento da solução. A análise parte do contexto atual do WhatsApp Web enquanto canal de comunicação empresarial e da ausência de integração com a plataforma E-goi. É ainda introduzido o projeto GoiZapy, concebido como resposta a essa limitação, através da criação de uma extensão para o navegador Google Chrome.

### 1.1. Contexto e motivação

O WhatsApp é uma aplicação de comunicação que permite a troca de mensagens de texto, voz e conteúdos multimédia em tempo real. A versão Web estende estas funcionalidades ao ambiente de *desktop*, facilitando a gestão de interações (Jannah, 2023). Originalmente desenvolvido para uso pessoal, tem vindo a ser adotado pelas empresas como meio de comunicação direta com os seus clientes, sendo utilizado em contextos de atendimento, apoio pós-venda, envio de conteúdos promocionais e campanhas (Tenório et al., 2020).

A utilização do WhatsApp Web no marketing relacional é sustentada por fatores como a acessibilidade da plataforma, a informalidade do canal e a percepção de proximidade social. De acordo com Zarouali et al. (2021), os utilizadores tendem a confiar nas marcas que comunicam via WhatsApp, considerando-o um ambiente reservado, familiar e propício à partilha de informação. No entanto, estas interações permanecem frequentemente dissociadas de sistemas estruturados de gestão de contactos ou de plataformas de automação de marketing, dificultando a sua integração em estratégias mais abrangentes.

A plataforma E-goi, especializada em automação de marketing multicanal, oferece funcionalidades para gestão de contactos, listas, campanhas, segmentações e notificações. Através da sua *API REST*, é possível integrar sistemas externos e automatizar tarefas como a criação e atualização de contactos ou a atribuição de *tags* e *triggers*. Segundo a própria documentação, a *API* “foi criada para ser simples, robusta e fácil de usar em qualquer linguagem de programação” (E-goi, 2024b), proporcionando flexibilidade na integração com aplicações externas.

Contudo, a E-goi não dispõe, à data, de uma solução nativa que permita capturar dados diretamente do WhatsApp Web. Tal como referido na proposta do projeto, disponível no Anexo 1, “atualmente, não existe uma solução integrada entre o WhatsApp Web e o E-

goi que permita automatizar estas interações, gerindo leads diretamente no WhatsApp e conectando-as aos fluxos de automação da E-goi” (E-goi, 2024a, p. 1).

Neste contexto, a extensão de navegador surge como solução técnica viável para estabelecer essa ligação. A arquitetura das extensões Chrome permite injetar *scripts* que acedem ao conteúdo da página web, extraem dados visíveis no *DOM* e os transmitem, de forma controlada, para serviços externos. Como defendido por Liu et al. (2020), a separação de processos e permissões no modelo das extensões garante um ambiente seguro e flexível para a implementação de funcionalidades específicas, sem alterar a aplicação de origem.

O projeto GoiZapy propõe uma solução de integração entre o WhatsApp Web e a plataforma E-goi, através do desenvolvimento de uma extensão para o navegador Google Chrome. Esta integração visa eliminar tarefas manuais, reduzir a dispersão da informação e permitir a articulação direta entre conversação digital e estratégias de marketing automatizado.

## 1.2. Objetivos

O presente projeto tem como objetivo central o desenvolvimento de uma extensão para o navegador Google Chrome, concebida para capturar contactos diretamente do WhatsApp Web e integrá-los na plataforma E-goi.

Para a concretização deste objetivo, foram definidas as seguintes funcionalidades:

- **Captura de dados no WhatsApp Web** - extrair automaticamente nomes, números de telefone e endereços de email a partir da estrutura *DOM* das conversas, através de *scripts* injetados pela extensão.
- **Integração com a API da E-goi** - sincronizar os dados recolhidos com a plataforma, utilizando chamadas *REST* para criação ou atualização de contactos.
- **Gestão de contactos, listas e tags** - implementar funcionalidades para criação, edição e eliminação de contactos, listas e *tags*, incluindo ações isoladas ou em lote.
- **Automação com base em triggers** - possibilitar a atribuição de *tags* que ativem fluxos automatizados na E-goi, como campanhas de seguimento.

Todos as funcionalidades foram implementadas e integradas numa solução, validada em ambiente de testes com a API da E-goi.

### **1.3. Estrutura do documento**

Este relatório encontra-se dividido em quatro secções. A primeira enquadra o projeto, apresentando a motivação e os objetivos. A segunda descreve a entidade de acolhimento, destacando a sua estrutura organizacional e contexto técnico. A terceira desenvolve o projeto GoiZapy, integrando os fundamentos teóricos com a implementação realizada. Por fim, a quarta secção apresenta as conclusões e propõe direções para desenvolvimento futuro.

## 2. A empresa

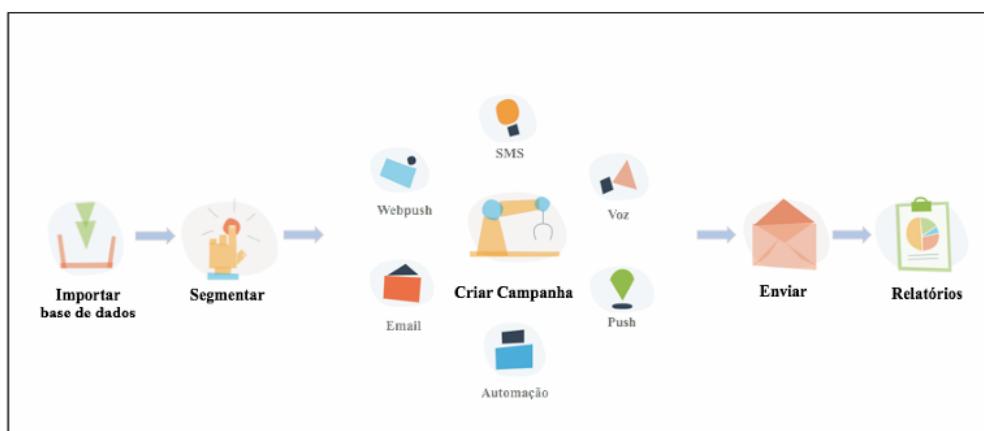
A segunda secção deste relatório apresenta a organização de acolhimento do estágio curricular, fornecendo um enquadramento institucional, técnico e organizacional da E-goi. São abordados aspectos como a missão da empresa, a sua estrutura interna, os principais departamentos e as tecnologias adotadas no contexto empresarial, com ênfase na equipa em que o projeto foi desenvolvido.

A informação apresentada baseia-se em documentação oficial disponibilizada pela empresa, nomeadamente o Manual do Sistema de Gestão Integrado (E-goi, 2023a) e o Manual de Funções (E-goi, 2023b).

### 2.1. Apresentação da empresa

A E-goi é uma empresa tecnológica portuguesa, fundada em 2005 e sediada em Matosinhos, Porto. Opera no setor do marketing digital, com foco na automatização de comunicações multicanal através de uma plataforma própria disponibilizada em modelo *Software as a Service (SaaS)* (E-goi, 2023a).

A plataforma permite planejar, executar e monitorizar campanhas em canais como email, SMS, notificações *push*, chamadas de voz e redes sociais. Integra ainda funcionalidades como *landing pages*, fluxos automáticos, testes A/B, funis de vendas, recuperação de carrinhos, gestão de contactos e análise comportamental (E-goi, 2023a). O percurso típico de utilização é representado na Figura 1.



**Figura 1** - Fluxograma de Atividades na Plataforma E-goi (adaptado de Reis, 2018, p. 14).

Segundo documentação interna, trata-se de “uma plataforma de automação de marketing omnicanal que integra funcionalidades diversas, adaptadas às necessidades dos utilizadores” (E-goi, 2023a, p. 3). Aplica-se a setores como *e-commerce*, retalho, imobiliário, financeiro, agências de marketing digital e *startups*, ajustando-se às exigências específicas de cada um.

## **2.2. Missão, visão e cultura organizacional**

A missão da E-goi é “criar as melhores soluções de comunicação digital, acessíveis a todos, garantindo o sucesso do cliente” (E-goi, 2023a, p. 3). A visão centra-se em “imaginar um mundo em que as marcas conhecem, entendem, e comunicam na perfeição com a sua audiência, com respeito, e criando cada vez mais valor para todas as pessoas” (E-goi, 2023a, p. 3). A meta da organização é “ser uma referência na área da comunicação eficaz e das tecnologias da informação” (E-goi, 2023a, p. 3).

A cultura organizacional baseia-se na valorização dos colaboradores, na promoção da motivação, do trabalho em equipa e no respeito pela diversidade. A empresa compromete-se com um ambiente seguro, inclusivo e orientado para o desenvolvimento contínuo (E-goi, 2023a).

## **2.3. Orientações de integração**

A E-goi implementa um processo de integração aplicável a todos os novos elementos da organização, incluindo estagiários, com o objetivo de assegurar uma adaptação rápida e alinhada com a cultura da empresa.

O acolhimento inclui a apresentação do plano de *onboarding*, a designação de um padrinho ou madrinha, a formalização legal e a entrega dos recursos necessários ao desempenho da função. Este processo foi validado pela experiência direta durante o estágio.

O percurso formativo inicial contempla:

- Formação específica da função, direcionada para as atividades do cargo ou estágio;
- Formações basilares, transversais à organização e centradas nos valores e metodologias da E-goi;
- *Product Academy (PA)*, focada nas funcionalidades da plataforma.

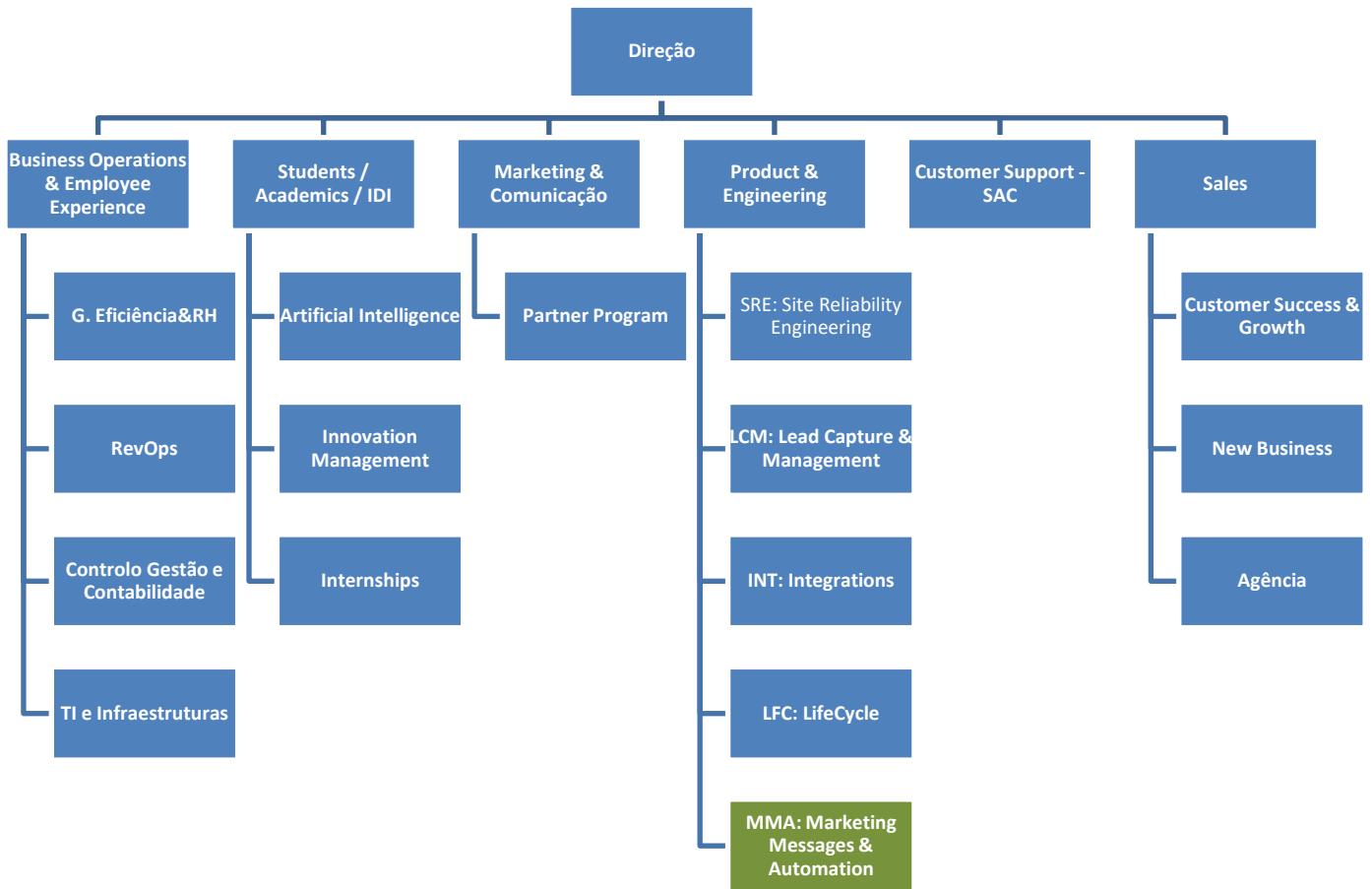
## 2.4. Estrutura organizacional

A E-goi estrutura-se segundo um modelo funcional, com departamentos especializados que asseguram a coordenação das operações e o cumprimento dos objetivos estratégicos. A gestão de topo orienta a estratégia global e promove a articulação entre equipas, valorizando uma cultura colaborativa, assente na comunicação interna e na melhoria contínua (E-goi, 2023b).

A organização reconhece como partes interessadas a direção, os colaboradores, clientes, fornecedores, parceiros institucionais, entidades reguladoras e concorrentes, sendo as suas necessidades e expectativas regularmente monitorizadas.

Os principais departamentos, conforme descrito no Manual de Funções e representados na Figura 2, são:

- ***Business Operations & Employee Experience*** - gere as áreas administrativa, financeira e de recursos humanos.
- ***Students / Academics / I&D*** - promove parcerias académicas e lidera iniciativas de investigação e inovação.
- ***Marketing & Comunicação*** - coordena a comunicação interna, externa e institucional da marca.
- ***Product & Engineering*** - desenvolve e mantém a plataforma, alinhando visão estratégica e tecnologia.
- ***Customer Success (SAC)*** - assegura o apoio ao cliente e a fidelização no segmento PME.
- ***Sales*** - supervisiona as equipas comerciais e acompanha clientes empresariais.



**Figura 2 - Estrutura Organizacional da E-goi (Fonte: elaboração própria).**

## 2.5. Equipa MMA – Marketing Messages & Automation

A equipa MMA (*Marketing Messages & Automation*) integra o departamento de *Product & Engineering* da E-goi e é responsável pelo desenvolvimento e manutenção das funcionalidades da plataforma relacionadas com a criação, gestão e automação de mensagens de marketing. As suas atividades centram-se na implementação de ferramentas para campanhas multicanal, segmentação de públicos, definição de fluxos automáticos e personalização da comunicação.

O presente estágio foi realizado no âmbito desta equipa, com base numa experiência prática integrada nas dinâmicas de desenvolvimento funcional da plataforma.

## 2.6. Ferramentas digitais e tecnologias de suporte

A E-goi adota um conjunto de ferramentas digitais para apoiar o desenvolvimento de software, a comunicação interna e a gestão de projetos. Entre as principais tecnologias utilizadas, destacam-se:

- **Bitrix24** - plataforma de colaboração usada para gestão de recursos internos, comunicação entre equipas, organização de tarefas e acesso a documentação interna.
- **Jira** - ferramenta de gestão de projetos, com metodologias ágeis (*Kanban* e *Scrum*), utilizada para organização de tarefas e acompanhamento de desenvolvimento.
- **GitLab** - sistema de controlo de versões e integração contínua, suportando *pipelines CI/CD* e colaboração entre equipas técnicas.
- **SonarQube** - utilizado para análise de qualidade e segurança do código.
- **Docker** - ferramenta de virtualização e gestão de ambientes de desenvolvimento.
- **Zend Framework, PHP, Python, JavaScript, AngularJS** - principais tecnologias empregues no desenvolvimento da plataforma.

### 3. Projeto GoiZapy

A presente secção organiza-se em duas componentes complementares: fundamentos teóricos e fundamentos práticos. A componente teórica fornece o enquadramento técnico e conceptual que justifica e suporta as decisões tomadas durante o desenvolvimento da extensão. Já a componente prática documenta todas as atividades realizadas no âmbito do estágio, com foco na implementação e desenvolvimento da extensão GoiZapy.

#### 3.1. Fundamentos teóricos

Nesta secção são considerados o funcionamento do WhatsApp Web, explorando o seu modelo de execução *client-side*, a manipulação do *DOM* e as limitações inerentes à ausência de uma *API* oficial. A arquitetura das extensões Chrome, com destaque para os seus componentes estruturais, as restrições impostas pelo *Manifest V3* e as implicações de segurança associadas à integração com plataformas externas. E a interface da *API* da E-goi, cuja abordagem *RESTful*, autenticada por *API Key*, permite a gestão programática de contactos, listas, *tags* e supressões. Este enquadramento permite compreender a viabilidade da solução, os constrangimentos operacionais enfrentados e os critérios que orientaram a definição da arquitetura da extensão.

##### 3.1.1. WhatsApp

O presente tópico aborda os princípios técnicos do WhatsApp Web que fundamentam a sua integração com a extensão GoiZapy, nomeadamente as diferenças entre contas pessoais e comerciais, o modelo de execução client-side, a estrutura do DOM e as limitações inerentes à ausência de interfaces públicas de integração. Estes elementos são importantes para compreender como a extensão acede e extrai dados diretamente da interface do utilizador, superando os constrangimentos impostos pela inexistência de suporte oficial a sistemas externos.

###### 3.1.1.1. Diferenças entre contas pessoais e comerciais

O WhatsApp disponibiliza duas variantes de conta para utilização no WhatsApp Web: a pessoal e a empresarial. Ambas operam sobre a mesma interface Web (<https://web.whatsapp.com>), com diferenças subtils no plano funcional e estrutural.

As contas pessoais destinam-se ao uso individual e são limitadas a interações sociais não comerciais. Já as contas comerciais, registadas através da aplicação WhatsApp Business, permitem a exibição de informações adicionais como nome da empresa, descrição, localização, horário de funcionamento e catálogo de produtos, visíveis na

área de "*Detalhes de Contacto*", conforme documentado na própria plataforma (WhatsApp, n.d.).

Apesar de partilharem a mesma estrutura geral da interface, o WhatsApp introduz diferenças nos atributos HTML e metainformação associada a cada tipo de conta (Schaffner et al., 2024). Este detalhe é importante para ferramentas que operam diretamente sobre o *DOM*, como é o caso do GoiZapy, ainda que, do ponto de vista funcional, ambas as contas sejam tratadas da mesma forma no WhatsApp Web (Rastogi & Hendler, 2017).

### **3.1.1.2. Modelo client-side**

O WhatsApp Web adota um modelo de execução integralmente *client-side*, em que toda a lógica de apresentação e interação ocorre no navegador do utilizador. A aplicação é disponibilizada como uma interface web dinâmica, que se comunica com os servidores do WhatsApp através de *WebSockets*, mantendo uma sessão sincronizada com o dispositivo móvel (Esiyok et al., 2023).

Todo o conteúdo das conversas, incluindo mensagens, nomes de contacto e metainformação, é renderizado diretamente no *DOM*, tornando-se acessível em tempo real a partir da estrutura HTML da página (Somé, 2019). Esta abordagem elimina a necessidade de renderização no servidor, delegando ao cliente a responsabilidade pelo ciclo de atualização e apresentação dos dados.

A interface do WhatsApp Web não emprega mecanismos de ofuscação dos elementos *DOM*, o que permite que extensões como o GoiZapy acedam programaticamente aos dados exibidos, observando mutações estruturais da interface e extraíndo informação relevante para posterior integração com sistemas externos (Schaffner et al., 2024; Somé, 2019).

### **3.1.1.3. Acesso e manipulação do DOM**

O *Document Object Model (DOM)* representa a estrutura hierárquica de uma página web, permitindo que *scripts* acedam, leiam e modifiquem elementos HTML e os respetivos atributos em tempo real. No modelo de execução do WhatsApp Web, toda a interface da aplicação é gerada e atualizada dinamicamente no lado do cliente, sem renderização do lado do servidor (Esiyok et al., 2023). A interface é construída com HTML, estilizada com CSS e controlada por JavaScript, com os dados das conversas e contactos inseridos no *DOM* à medida que o utilizador interage com a aplicação. Elementos como nomes, números de telefone, ícones de estado e mensagens são

injetados no *DOM* e podem ser identificados por seletores específicos (Rastogi & Hendler, 2017).

Do ponto de vista da engenharia de software, extensões desenvolvidas para o Google Chrome têm acesso ao *DOM* de páginas web através de *content scripts*, que operam num contexto isolado, mas partilham a árvore *DOM* da página (Somé, 2019). Estes *scripts* são executados diretamente no ambiente da aplicação web e permitem capturar ou observar alterações no conteúdo apresentado, sem modificar o comportamento original da página.

No caso do WhatsApp Web, a ausência de mecanismos de ofuscação e a visibilidade pública de grande parte da estrutura da interface tornam possível a monitorização e extração de dados. Este recurso permite detetar alterações no *DOM*, como a abertura de uma nova conversa ou o carregamento de informações de contacto, e responder automaticamente a eventos visuais.

A relevância desta capacidade torna-se evidente em soluções como a extensão GoiZapy, que se baseia na leitura do *DOM* para obter dados estruturais da interface do WhatsApp Web.

#### **3.1.1.4. Considerações técnicas e limitações de integração**

Apesar da sua ampla adoção, o WhatsApp Web não foi concebido para suportar integrações com sistemas externos. A ausência de uma *API* pública acessível diretamente através da interface Web limita a interoperabilidade com ferramentas de automação ou CRMs. Além disso, a estrutura dinâmica da interface impõe desafios ao reconhecimento estável de elementos *DOM*, uma vez que seletores CSS podem ser alterados em atualizações da plataforma (Schaffner et al., 2024).

Do ponto de vista técnico, não existem pontos de extensão nativos, e qualquer extração de dados requer observação contínua do *DOM* e adaptação a eventuais alterações de layout. Esta ausência de suporte oficial impõe restrições à confiabilidade da solução e exige estratégias de engenharia que assegurem resiliência e compatibilidade com alterações futuras do *front-end* da aplicação (Esiyok et al., 2023).

Tais limitações justificam a abordagem do GoiZapy, baseada em *content scripts* e observadores estruturais, como mecanismo de acesso indireto e controlado à informação apresentada no cliente.

### 3.1.2. API da E-goi

O tópico apresenta os fundamentos da API da E-goi relevantes para a compreensão da sua aplicação no projeto GoiZapy, incluindo o modelo de funcionamento, os formatos de dados suportados, os mecanismos de autenticação e comunicação, os limites de utilização, a integração do SDK, os recursos manipulados pela extensão e a articulação com os fluxos de automação disponibilizados pela plataforma.

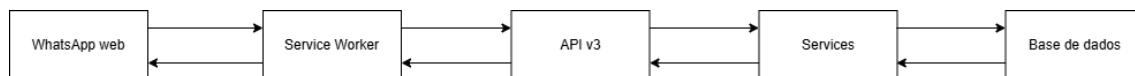
#### 3.1.2.1. Modelo funcional

A API da E-goi (v3) permite a integração direta com os serviços da plataforma, incluindo gestão de contatos, envio de campanhas (e-mail, SMS, *push*), automações e segmentações (E-goi, 2024b). Toda a documentação técnica, rotas, parâmetros, exemplos e respostas, está disponível publicamente e é atualizada conforme alterações na plataforma (E-goi, 2024b).

A API atua como camada de interface entre sistemas externos e os serviços internos da E-goi. As requisições são redirecionadas para <https://services.e-goi.com/api>, onde são processadas por serviços aplicacionais que interagem com a base de dados.

No projeto GoiZapy, a extensão coleta dados diretamente do *DOM* do WhatsApp Web e os envia ao *service worker*, que formata e encaminha os dados para a API da E-goi via chamadas *REST*. Essa abordagem mantém a separação entre interface e lógica de integração.

A Figura 3 representa a arquitetura e os fluxos de comunicação entre os componentes.



**Figura 3 - Fluxo de Comunicação entre Componentes** (Fonte: elaboração própria).

#### 3.1.2.2. Arquitetura RESTful

A API v3 da E-goi segue o modelo *REST*, estruturado em torno de recursos identificáveis por *URIs* e manipuláveis através de métodos HTTPS. Os principais recursos expostos incluem contactos, listas, *tags* e listas de supressão, acessíveis sob o domínio principal <https://api.egoiaapp.com>.

A interação é realizada por meio de chamadas assíncronas, com a seguinte correspondência semântica entre métodos e operações:

- **GET** - Leitura de dados existentes.
- **POST** - Criação de novos recursos.
- **PATCH** - Alteração parcial de atributos.

- **PUT** - Substituição completa do recurso.
- **DELETE** - Remoção com base no identificador.

### **3.1.2.3. Formato de dados**

A API da E-goi comunica por HTTPS, utilizando o formato **JSON (JavaScript Object Notation)** para a serialização dos dados transmitidos. Este formato, baseado em pares chave-valor, é apropriado para chamadas assíncronas e processamento em ambientes web. Cada requisição inclui o cabeçalho **Content-Type: application/json**, que define explicitamente o tipo de conteúdo enviado ao servidor (E-goi, 2024b).

As respostas seguem a mesma estrutura, devolvendo objetos JSON que contêm o estado da operação, eventuais mensagens de erro e dados associados ao recurso processado. Este modelo uniforme de entrada e saída facilita o tratamento programático no projeto GoiZapy, onde as respostas são validadas e processadas assincronamente no contexto isolado do *service worker*.

### **3.1.2.4. Autenticação**

O acesso à API da E-goi requer autenticação baseada em chave privada (**API Key**), incluída no cabeçalho **Authorization** com o formato **Apikey {chave}** (E-goi, 2024b). Este mecanismo assegura que apenas clientes autenticados podem aceder aos recursos expostos, permitindo rastrear e controlar cada interação com a plataforma.

A chave é gerada na área de administração da conta E-goi e é tratada como credencial sensível, uma vez que concede permissões equivalentes às do utilizador autenticado.

No projeto GoiZapy, a autenticação é realizada no *service worker*, que encapsula a lógica de comunicação com a API. Cada requisição **fetch()** inclui a chave de autenticação no cabeçalho da mensagem, garantindo isolamento entre a camada de interface (DOM) e o canal de comunicação externo.

### **3.1.2.5. Comunicação segura**

A API da E-goi exige comunicação exclusivamente via HTTPS, suportando as versões 1.2 e 1.3 do protocolo **TLS (Transport Layer Security)**, conforme especificado na documentação oficial (E-goi, 2024b). A versão TLS 1.3 é recomendada por oferecer menor latência na negociação e maior segurança criptográfica.

As cifras suportadas incluem:

- TLS\_AES\_128\_GCM\_SHA256
- TLS\_AES\_256\_GCM\_SHA384
- TLS\_CHACHA20\_POLY1305\_SHA256

Estas configurações garantem *forward secrecy (FS)*, impedindo a exposição de dados em cenários de interceção ou adulteração. A plataforma rejeita qualquer requisição não encriptada enviada via HTTP.

No projeto GoiZapy, todas as trocas de dados entre o *service worker* e a API da E-goi decorrem sob este canal encriptado.

### **3.1.2.6. Limites de utilização**

A API da E-goi aplica restrições de utilização para preservar a estabilidade e controlar a carga distribuída entre clientes. Estes limites encontram-se organizados em duas categorias: limites de taxa (*rate limits*), atribuídos por cliente; e limites de conta (*account limits*), aplicados globalmente a todas as requisições autenticadas com a mesma chave de API (E-goi, 2024b).

As respostas da API incluem cabeçalhos específicos que informam, em tempo real, o estado das quotas:

- **X-RateLimit-Limit** - número máximo de requisições permitidas na janela temporal atual.
- **X-RateLimit-Remaining** - número de requisições ainda disponíveis.
- **X-RateLimit-Reset** - tempo restante (em segundos) até à renovação da quota.

No caso dos limites de conta, são utilizados os seguintes cabeçalhos:

- **X-Account-Limit** - total de requisições permitido na janela global.
- **X-Account-Remaining** - número de chamadas restantes.
- **X-Account-Reset** - tempo até à reposição da quota da conta.

O código HTTP 429 *Too Many Requests* é devolvido sempre que qualquer um destes limites é ultrapassado, bloqueando temporariamente novas requisições até ao reinício do intervalo.

O respeito pelos limites de utilização é, por isso, uma parte integrante da lógica de comunicação com a API.

### **3.1.2.7. SDK**

A utilização do SDK oficial da E-goi no projeto GoiZapy teve como objetivo otimizar a integração com a *API v3*, eliminando a necessidade de construir manualmente requisições HTTP. O SDK abstrai operações como a serialização de objetos em *JSON*, a definição de cabeçalhos e a validação prévia dos dados a enviar (E-goi, 2024b).

A integração foi realizada no ambiente SvelteKit, com instalação via npm e importação nos módulos responsáveis pela comunicação com a API.

A execução do SDK foi confinada ao contexto do *service worker*, assegurando o isolamento da lógica de rede e conformidade com as restrições do *Manifest V3*. Esta delimitação evita a persistência desnecessária de processos e restringe os privilégios dos *scripts* da interface.

### **3.1.2.8. Tratamento de respostas**

A extensão GoiZapy implementa um mecanismo de validação sistemática das respostas devolvidas pela *API* da E-goi, assegurando a deteção imediata de falhas e a interpretação adequada dos resultados. Os códigos de estado tratados incluem:

- **200 OK** - resposta padrão para operações de leitura bem-sucedidas.
- **201 Created** - indica a criação de um novo recurso, como um contacto.
- **400 Bad Request** - erro de validação no *payload* enviado ou nos parâmetros definidos.
- **401 Unauthorized** - credenciais ausentes ou inválidas no cabeçalho de autenticação.
- **429 Too Many Requests** - excedido o limite de requisições permitido na janela temporal.

### 3.1.2.9. Recursos utilizados

A extensão GoiZapy interage com a *API* da E-goi utilizando autenticação por *API key*, permitindo identificar o utilizador autenticado e apresentar o seu nome na interface da extensão.

A nível funcional, a extensão utiliza os seguintes recursos da *API*:

- **Contactos** - criação, edição, eliminação, segmentação e ações em massa.
- **Listas** - criação, edição e remoção.
- **Tags** - criação, edição, remoção e associação a contactos.
- **Listas de supressão** - adição e remoção de contactos.

Estas operações garantem a integração dos dados captados com os fluxos automatizados da plataforma E-goi.

### 3.1.2.10. Integração com mecanismos de automações

A extensão GoiZapy integra-se com os mecanismos de automação da plataforma E-goi através da atribuição de *tags* aos contactos captados. Estas *tags* funcionam como *triggers*, ou eventos de entrada, que ativam automaticamente fluxos predefinidos na plataforma, como campanhas de email, SMS ou notificações *push*.

O modelo seguido é *event-based*, em que a alteração do estado de um recurso (por exemplo, a adição de uma *tag*) desencadeia uma reação automática, sem intervenção manual. Esta abordagem é suportada nativamente pela E-goi, que associa cada automação a um ou mais eventos disparadores. No caso do GoiZapy, a extensão inclui as *tags* diretamente no objeto de contacto enviado via POST /contacts, garantindo que a automação é iniciada no momento da sincronização.

Este mecanismo permite uma articulação fluida entre a captura de dados no WhatsApp Web e a execução de estratégias de marketing automatizado, assegurando continuidade no ciclo de relacionamento com o cliente, sem necessidade de processamento adicional na extensão. A Figura 4 ilustra um exemplo de automação configurada na E-goi, onde a adição da *tag* "campanha", (ID 10), a um contacto dispara automaticamente o envio de uma campanha SMS (ID 37).



**Figura 4** - Automação E-goi Acionada por Trigger de Tag (Fonte: plataforma E-goi).

### 3.1.3. Extensões Chrome

O presente tópico explora os fundamentos técnicos das extensões Chrome que sustentam a integração com o projeto GoiZapy, com foco na estrutura funcional, mecanismos de isolamento, regras de segurança, evoluções normativas e limitações estruturais que condicionam o seu funcionamento. Estes elementos são importantes para compreender como a extensão foi concebida para interagir com o ambiente do navegador, extraír dados de forma controlada e comunicar com serviços externos.

#### 3.1.3.1. Extensões de navegador

As extensões de navegador são aplicações modulares concebidas para ampliar as funcionalidades nativas dos navegadores, permitindo a personalização da experiência do utilizador e a integração com serviços externos. No caso do Google Chrome, estas extensões atuam como camadas adicionais de funcionalidade que interagem com o ambiente do utilizador, sem comprometer a estrutura base do navegador ou da aplicação web subjacente (Somé, 2019).

Constituem ferramentas versáteis, aplicadas em diversos contextos, desde o bloqueio de anúncios e preenchimento automático de formulários até à análise e manipulação de páginas web em tempo real. Esta flexibilidade torna-as particularmente adequadas a cenários onde se pretende captar, transformar ou redirecionar informação visível no navegador, como é o caso da extensão GoiZapy, que atua sobre a interface do WhatsApp Web.

O desenvolvimento de extensões está regulamentado por um conjunto de especificações técnicas que garantem a sua interoperabilidade, segurança e

desempenho. No ecossistema Chrome, este enquadramento é definido no manifesto da extensão, um ficheiro declarativo que estabelece os limites operacionais da aplicação, as permissões concedidas e os contextos em que pode atuar (Agarwal, 2022).

A sua importância estratégica para aplicações web justifica a crescente atenção da comunidade científica e dos organismos reguladores. Segundo Pantelaios e Kapravelos (2024), as extensões são simultaneamente pontos de extensão legítima e potenciais vetores de risco, exigindo uma modelação precisa das suas capacidades. Esta dualidade explica a adoção progressiva de normas mais restritivas, como o *Manifest V3*, que visa reforçar o controlo sobre o comportamento das extensões e minimizar o seu impacto na segurança do utilizador (Google Chrome Developers, 2024).

### **3.1.3.2. Histórico e evolução das especificações**

O ciclo de evolução das extensões Chrome, iniciado com o *Manifest V1*, foi marcado por uma grande flexibilidade programática, mas também por fragilidades evidentes em termos de segurança e controlo (Google Chrome Developers, 2024). O *Manifest V2* surgiu para mitigar parte desses riscos, impondo limites mais rigorosos ao uso de permissões e restringindo *APIs* suscetíveis a abuso.

No entanto, a execução contínua de *background scripts*, o uso de código dinâmico e o carregamento remoto de bibliotecas tornaram-se vulnerabilidades no ecossistema, afetando tanto a segurança dos utilizadores quanto o controlo do navegador (Agarwal, 2022).

O *Manifest V3* foi concebido para resolver essas falhas, introduzindo um modelo declarativo que transfere para o navegador o controlo sobre operações sensíveis. Entre as mudanças destacam-se a substituição dos *background scripts* por *service workers* com duração limitada, a eliminação do suporte a código dinâmico e a adoção de uma *Content Security Policy (CSP)* mais rigorosa, que restringe execuções não autorizadas (Pantelaios & Kapravelos, 2024).

O projeto GoiZapy foi desenvolvido diretamente com base no *Manifest V3*, uma vez que este modelo já impõe restrições técnicas e elimina dependências dinâmicas. Esta decisão garante conformidade com os requisitos de segurança, publicação e regulação aplicáveis às extensões de navegador (Pantelaios & Kapravelos, 2024; Agarwal, 2022).

### 3.1.3.3. Arquitetura de extensões

A arquitetura das extensões do Google Chrome assenta num modelo modular, orientado por princípios de segurança, isolamento e segmentação funcional. No centro desta estrutura encontra-se o ficheiro *manifest.json*, que define o escopo da extensão, declara os scripts utilizados, as permissões requeridas, os domínios autorizados e a versão da especificação adotada, sendo atualmente obrigatória a utilização do *Manifest V3* (Google Chrome Developers, 2024).

De forma geral, as extensões estruturam-se em dois componentes:

- **Content Scripts** - Scripts executados no contexto da página web alvo, com acesso direto ao DOM mas isolados do ambiente JavaScript da aplicação. Esta separação impede a interferência com variáveis internas da página, garantindo a segurança do utilizador e a estabilidade da aplicação de origem (Somé, 2019).
- **Service Workers** - unidades de controlo reativas responsáveis pela lógica assíncrona, comunicação com APIs externas e gestão de eventos internos. Estes componentes não mantêm estado persistente e são ativados exclusivamente em resposta a eventos definidos, em conformidade com o modelo de execução do *Manifest V3* (Agarwal, 2022).

A comunicação entre *content scripts* e *service workers* ocorre por mensagens explícitas (*chrome.runtime.sendMessage*), mantendo o isolamento de contexto e o controlo de privilégios entre domínios.

Este modelo adapta-se bem a aplicações dinâmicas *client-side*, como o WhatsApp Web, permitindo extrair dados visuais sem interferir na lógica interna. No GoiZapy, garante-se a separação entre a captura no *DOM* e a integração com a *API* da E-goi.

### 3.1.3.4. Execução Client-Side e interação com o DOM

Os navegadores web atuam como ambientes de execução para aplicações *client-side*, permitindo que toda a lógica de interação e renderização da interface ocorra no lado do utilizador. Este modelo baseia-se no acesso ao *Document Object Model (DOM)*, uma representação hierárquica da estrutura da página, manipulável através de *APIs* JavaScript integradas no navegador (Agarwal, 2022).

O WhatsApp Web enquadra-se neste paradigma, operando exclusivamente no cliente. Como aplicação de *front-end*, toda a interface e lógica de visualização são carregadas no navegador, o que torna os seus elementos acessíveis a *scripts* externos, sem necessidade de comunicação direta com o servidor da aplicação (Nayak et al., 2024). Esta característica torna o WhatsApp Web particularmente compatível com a atuação de extensões, que podem inspecionar e manipular o *DOM* em tempo real.

Apesar da política de segurança *Same Origin Policy (SOP)*, que restringe o acesso entre scripts de origens distintas, os *content scripts* de extensões Chrome operam como exceção controlada. Injetados diretamente no contexto da página, estes scripts têm acesso total ao *DOM*, sem partilhar o escopo de execução dos *scripts* internos da aplicação, o que garante isolamento lógico e segurança (Pantelaios & Kapravelos, 2024).

No contexto do projeto GoiZapy, esta arquitetura permite a extração de dados, como contactos presentes nas conversas, diretamente da interface do WhatsApp Web, sem modificação do código de origem. Esta abordagem preserva a integridade da aplicação analisada e cumpre as diretivas de privilégio mínimo recomendadas para extensões de navegador (Agarwal, 2022; Nayak et al., 2024).

### 3.1.3.5. Segurança e permissões

A segurança em extensões Chrome é regida por um modelo baseado no princípio do privilégio mínimo (*Least Privilege Principle*), segundo o qual cada componente deve operar apenas com as permissões estritamente necessárias à sua função. Esta abordagem, reforçada com a introdução do *Manifest V3*, visa limitar a superfície de ataque e reduzir os riscos associados a acessos indevidos, execução de código não autorizado ou fuga de dados sensíveis (Agarwal, 2022; Nayak et al., 2024).

Todas as permissões requeridas são declaradas explicitamente no ficheiro *manifest.json*, que distingue entre *permissions*, para funcionalidades internas como rede ou armazenamento, e *host\_permissions*, que definem os domínios com os quais a extensão está autorizada a interagir. No caso de aplicações que atuam sobre páginas específicas, como o WhatsApp Web, a indicação explícita do domínio <https://web.whatsapp.com> é obrigatória, garantindo que a extensão só opera no contexto autorizado (Google Chrome Developers, 2024).

Outro elemento importante do modelo de segurança é a *Content Security Policy (CSP)*, que limita a execução de scripts a origens autorizadas e bloqueia código dinâmico não declarado. Esta política previne ataques *Cross-Site Scripting (XSS)* e impede a execução de código remoto (Pantelaios & Kapravelos, 2024).

Além disso, a arquitetura imposta pelo *Manifest V3* promove o isolamento entre contextos, impedindo que os *content scripts*, que interagem com o *DOM*, tenham acesso a objetos partilhados, como *localStorage*, *cookies* ou variáveis da aplicação web. Esta separação garante que a extensão não interfere com o comportamento da página e que *scripts* de terceiros não consigam manipular a extensão (Somé, 2019).

No projeto GoiZapy, este modelo permite uma integração segura com o WhatsApp Web, limitando as permissões ao estritamente necessário e garantindo que nenhuma informação sensível (como dados de utilizador ou *localStorage*) seja acedida ou armazenada. A extensão opera sem persistência local de dados e sem execução contínua, alinhando-se com as práticas definidas pela Chrome Web Store e pela documentação oficial de extensões da Google. A lógica de integração com a API da E-goi decorre de forma segura no *service worker*, enquanto a observação do DOM é realizada em contexto isolado, respeitando todas as diretivas de segurança definidas pelo *Manifest V3*.

#### **3.1.3.6. Considerações éticas e de privacidade**

O desenvolvimento de extensões que manipulam dados pessoais, como números de telefone captados a partir do WhatsApp Web, exige rigor ético e alinhamento com as normas legais aplicáveis à proteção de dados. Extensões deste tipo devem seguir princípios fundamentais como a minimização de tratamento, evitando a recolha excessiva de informações, a não retenção local, garantindo que dados sensíveis não permaneçam no dispositivo do utilizador, e a proibição de partilha não autorizada com terceiros, respeitando integralmente os limites do consentimento informado (Kariyaa et al., 2021).

No projeto GoiZapy, os dados extraídos são imediatamente transmitidos, de forma autenticada e segura, para a *API* oficial da E-goi, sem qualquer forma de armazenamento intermédio ou persistente no ambiente local da extensão. A arquitetura foi concebida para operar de modo não intrusivo, interagindo exclusivamente com elementos explicitamente autorizados no manifesto, assegurando que não ocorre interferência indevida na aplicação alvo nem exposição adicional de informações sensíveis. Este alinhamento traduz um compromisso não apenas com os requisitos técnicos, mas também com a responsabilidade social e ética no desenvolvimento de software.

#### **3.1.3.7. Limitações do modelo de extensões**

O modelo definido pelo *Manifest V3* impõe restrições que impactam diretamente projetos como o GoiZapy, sobretudo no que diz respeito à gestão de fluxos dinâmicos. Funcionalidades como *reCAPTCHA*, autenticação em duas etapas (2FA) e *tokens* de acesso não podem ser tratados no contexto das extensões, pois dependem de *scripts* dinâmicos, *iframes* embutidos e lógicas interativas geradas em tempo real, elementos

expressamente bloqueados pelas políticas de segurança e pela *Content Security Policy* (*CSP*) reforçada no *Manifest V3*.

Estas limitações conduziram a decisões técnicas estratégicas no desenvolvimento do GoiZapy, nomeadamente a adoção exclusiva da *API* pública v3 da E-goi. Esta opção assegura compatibilidade com o ambiente restritivo das extensões Chrome, garantindo o cumprimento das exigências declarativas impostas pelo *Manifest V3* e permitindo uma comunicação segura, estável e controlada, sem recorrer a fluxos interativos ou manipulações dinâmicas incompatíveis com o modelo.

### 3.2. Fundamentos práticos

Esta secção abrange a definição dos requisitos operacionais, o desenho arquitetural da extensão com a separação entre *content scripts*, *service worker* e interface construída em SvelteKit, bem como o desenvolvimento das principais funcionalidades de captura, gestão e sincronização de dados. Integra também a comunicação com a *API* da E-goi, permitindo operações em tempo real sobre recursos como contactos, listas, *tags* e supressões. Complementarmente, contempla os testes de usabilidade, o acompanhamento do desenvolvimento por ciclos iterativos e o alinhamento visual com a identidade da plataforma. Esta componente demonstra a transição do projeto da sua conceção teórica para uma solução funcional e tecnicamente integrada.

#### 3.2.1. Projetos semelhantes

No panorama atual de integrações entre plataformas de CRM e mensageiros instantâneos, existem soluções que partilham objetivos similares ao GoiZapy. Um exemplo notório é o WhatStation, uma extensão desenvolvida pela RD Station que também atua sobre o WhatsApp Web com o intuito de capturar informações de contacto e facilitar o registo de leads no CRM da empresa. Esta ferramenta permite, por exemplo, adicionar um contacto diretamente no RD Station a partir de uma conversa no WhatsApp, com a possibilidade de etiquetar o lead e iniciar fluxos de automação personalizados (RD Station, 2024a).

A documentação oficial da plataforma destaca que o objetivo da integração é reduzir o tempo operacional gasto em tarefas manuais e garantir que leads capturados via WhatsApp não fiquem fora dos processos de nutrição e qualificação (RD Station, 2024b). Essa abordagem está em linha com o modelo funcional do GoiZapy, embora a extensão se distinga pela simplicidade e pela integração direta com a API REST da E-goi, sem dependência de vários formulários ou camadas de configuração no cliente.

Tal como salientado pela RD Station: “A integração com o WhatsApp permite centralizar as interações comerciais e não perder oportunidades por falta de sincronização entre os canais de comunicação” (RD Station, 2024c). O GoiZapy retoma essa premissa com uma proposta centrada na automatização passiva, sem necessidade de interface adicional, otimizando assim o tempo de resposta e a fiabilidade da informação capturada.

### **3.2.2. Requisitos do sistema**

A extensão GoiZapy foi concebida a partir de especificações técnicas organizadas em requisitos funcionais e não funcionais. Estes requisitos asseguram a coerência entre os objetivos do projeto e as restrições impostas pelo ambiente de execução, garantindo a integração correta com o WhatsApp Web e a API da E-goi, bem como a conformidade com as diretrizes do *Manifest V3*. A sua definição orientou a arquitetura da solução e permitiu estabelecer critérios de fiabilidade, segurança e desempenho aplicáveis ao desenvolvimento de extensões Chrome.

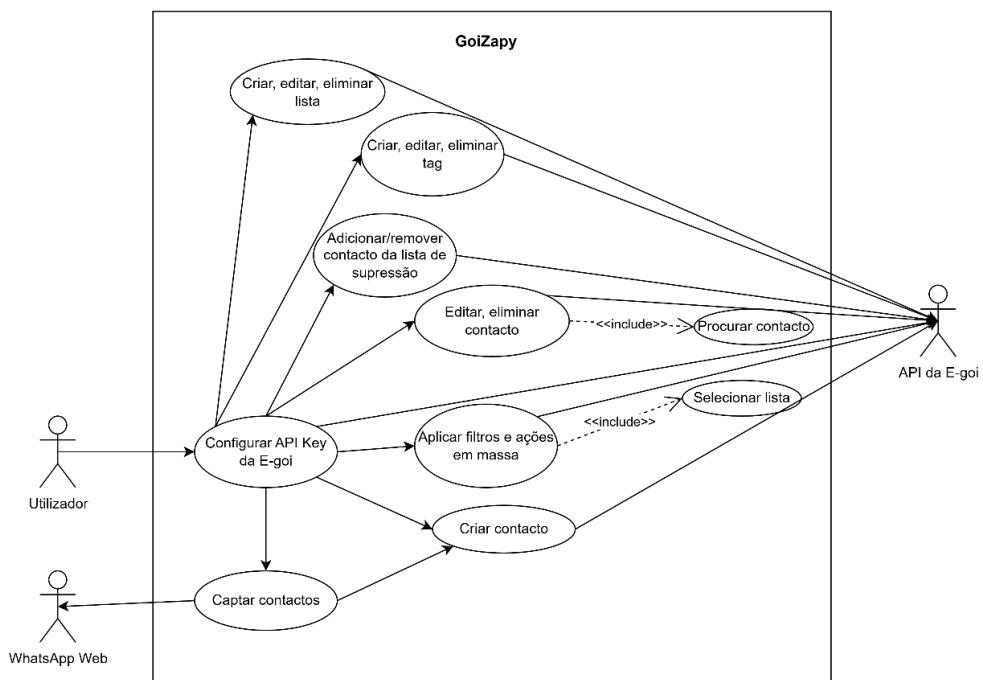
#### **3.2.2.1. Requisitos funcionais**

Os requisitos funcionais definem o comportamento esperado da extensão no seu ambiente de execução e descrevem as funcionalidades importantes que garantem o cumprimento dos objetivos do projeto. No contexto do GoiZapy, estes requisitos foram delineados para assegurar a integração automática entre o WhatsApp Web e a API da E-goi, com o mínimo de intervenção do utilizador:

- **Extração automática de dados de contacto** - identificação e recolha, em tempo real, de nomes, números de telefone e email visíveis na interface do WhatsApp Web.
- **Observação de alterações na interface** - deteção de mutações no DOM que sinalizem a abertura ou atualização de conversas, acionando os mecanismos de captura.
- **Validação local da informação** - verificação da estrutura e consistência dos dados recolhidos, evitando redundâncias, omissões ou falhas na integração.
- **Transmissão segura dos dados** - envio estruturado das informações captadas para a API REST da E-goi.
- **Funcionamento passivo e contínuo** - operação da extensão em segundo plano, permitindo ao utilizador navegar livremente no WhatsApp Web.

- **Adição de contactos** - permite adicionar contactos por extração direta do WhatsApp ou manualmente, através da interface da extensão.
- **Consulta, edição e eliminação de contactos** - os contactos registados podem ser pesquisados, atualizados ou removidos diretamente na extensão.
- **Visualização e gestão de listas** - integra funcionalidades para consultar os elementos de uma lista da E-goi, com suporte a filtros e ações em massa, como a atribuição de *tags*.
- **Gestão de tags** - inclui criação, edição (nome e cor), aplicação e remoção de tags, possibilitando segmentações dinâmicas e automações com base em triggers definidos na E-goi.
- **Administração de listas** - permite criar, editar ou eliminar listas, com controlo sobre o nome interno e o nome público associados.
- **Gestão da lista de supressão** - permite adicionar ou remover emails e números de telefone da lista de supressão, assegurando que os contactos indesejados são corretamente excluídos dos fluxos de comunicação.

Para suportar a definição dos requisitos funcionais, apresenta-se o diagrama de casos de uso da extensão (Figura 5), que representa as principais interações entre o utilizador e as funcionalidades disponibilizadas. Este modelo contribui para a identificação dos comportamentos esperados e serve de base à especificação técnica subsequente.



**Figura 5** - Diagrama de Casos de Uso da Extensão GoiZapy (Fonte: elaboração própria).

### 3.2.2.2. Requisitos não funcionais

Os requisitos não funcionais estabelecem os critérios técnicos que garantem a integridade da solução no seu contexto de execução. No caso do GoiZapy, estes requisitos visam assegurar conformidade com as normas do ecossistema Chrome, desempenho estável e uma arquitetura coerente com os princípios de segurança e isolamento:

- **Conformidade com o *Manifest V3*** - utilização exclusiva de *service workers* e aplicação das regras de *Content Security Policy (CSP)*.
- **Separação de responsabilidades** - segmentação da lógica em módulos distintos, *content script* para acesso ao *DOM* e *service worker* para comunicação com a *API* externa.
- **Segurança baseada no princípio do privilégio mínimo** - solicitação apenas das permissões estritamente necessárias à operação da extensão, conforme declarado no ficheiro *manifest.json*.
- **Execução não intrusiva** - minimização do impacto no desempenho do navegador e ausência de alterações visuais na interface do WhatsApp Web.
- **Adaptação a alterações estruturais** - tolerância a mudanças no layout ou nos seletores da interface do WhatsApp Web, com base em lógica reativa e observadores dinâmicos.
- **Desempenho otimizado** - a extensão deve manter um tempo de resposta reduzido, idealmente inferior a um segundo, entre a deteção de novos dados e o envio para a API, sem comprometer a fluidez da navegação.

### 3.2.3. Metodologia

O desenvolvimento do projeto GoiZapy seguiu uma abordagem iterativa, estruturada em ciclos sucessivos de pesquisa, implementação e validação, com foco na integração com a API da E-goi. As escolhas técnicas assentaram em princípios de modularidade, coesão estrutural e conformidade com a arquitetura das extensões Chrome.

A seleção das ferramentas e *frameworks* utilizados foram orientadas por critérios de desempenho, manutenibilidade e compatibilidade com o modelo de execução de extensões Chrome.

- **SvelteKit** - estrutura a interface da extensão, permitindo a criação de componentes reativos com grau de otimização, adequados ao ambiente restrito das extensões Chrome (Svelte Docs; Trentbrew, 2023).
- **TypeScript** - assegura tipagem estática, deteção antecipada de erros e maior previsibilidade na evolução do código.
- **Vite** - gera o sistema de *build* do projeto, proporcionando tempos de compilação reduzidos, recarregamento rápido em desenvolvimento e uma estrutura de ficheiros simples, adequada ao contexto das extensões Chrome (Plain English, 2022).
- **Manifest V3** - define o modelo de execução da extensão, impondo a utilização de *service workers*, bloqueio de código dinâmico e regras de segurança explícitas através de *Content Security Policy (CSP)* (Google Chrome Developers, 2024).
- **SDK oficial da E-goi** - integra-se diretamente no *service worker*, facilitando o consumo da *API REST v3* para operações como gestão de contactos, listas, *tags* e lista de supressão (E-goi, 2024b; Plain English, 2023).

### 3.2.4. Descrição funcional da extensão

O presente tópico descreve as principais funcionalidades da extensão GoiZapy, evidenciando como se operacionalizam, na interface, os requisitos funcionais previamente definidos. Cada subtópico corresponde a uma área funcional da solução, sendo apresentada com base na experiência de utilização, nas ações disponíveis para o utilizador e nos mecanismos de interação com a *API* da E-goi. Esta descrição visa complementar a especificação técnica com uma perspetiva aplicada, ilustrando a lógica operacional da extensão.

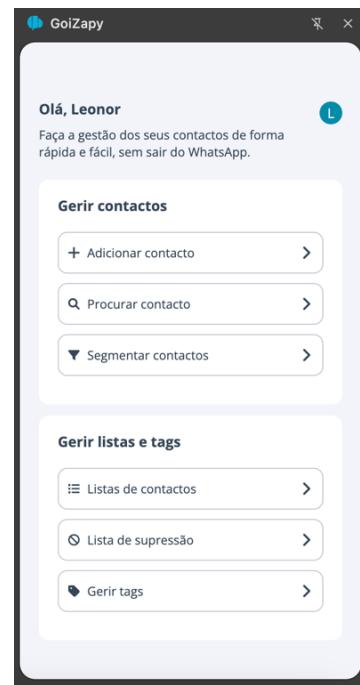
#### 3.2.4.1. Login

O processo de autenticação da extensão inicia-se com a introdução da chave de API fornecida pela E-goi (Figura 6). Esta credencial é obrigatória para estabelecer a comunicação com a *API v3* e desbloquear o acesso às funcionalidades da extensão. A chave é validada por meio de uma chamada HTTPS ao serviço de verificação de conta. Em caso de validação bem-sucedida, é armazenada localmente com *chrome.storage.local* e configurada na instância do SDK (*egoisdk.ApiClient*).

Sempre que a extensão é carregada, é verificada automaticamente a existência de uma chave previamente guardada. Se presente, é efetuada uma validação junto do serviço de conectividade da E-goi. Se a chave for considerada válida, o utilizador é redirecionado para o menu principal (Figura 7), caso contrário, a interface permanece bloqueada até nova autenticação.



**Figura 6 - Ecrã de Autenticação da Extensão GoiZapy** (Fonte: elaboração própria).



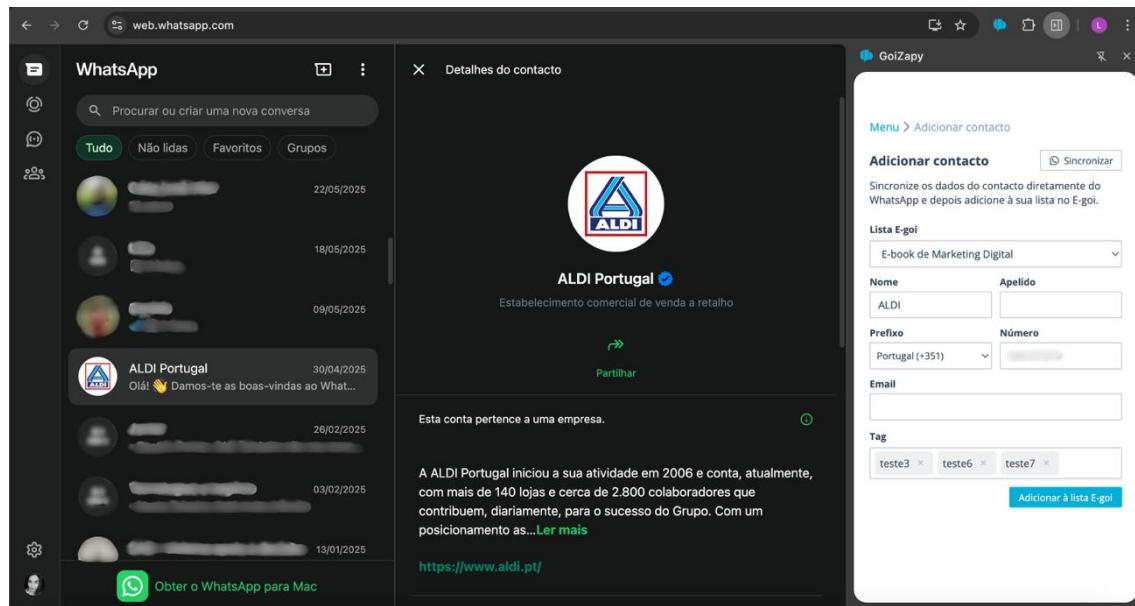
**Figura 7 - Menu Principal da Extensão GoiZapy** (Fonte: elaboração própria).

### 3.2.4.2. Adicionar contacto

A funcionalidade de adição de contacto permite registar novos elementos numa lista da E-goi, tanto manualmente como por captação automatizada a partir do WhatsApp Web (Figura 8). A interface apresenta um formulário com campos para nome, apelido, número de telefone, email, lista de destino e *tags* associadas.

No modo manual, o utilizador insere os dados diretamente e seleciona a lista e as *tags* disponíveis, previamente carregadas através da *API*. Alternativamente, pode acionar a opção “*Sincronizar*”, que ativa a *content script* responsável por extrair os dados do contacto atualmente em foco no WhatsApp. Este *script* analisa a estrutura do *DOM*, identifica os campos relevantes (nome, número de telefone e, quando presente, email) e preenche automaticamente o formulário da extensão.

Após submissão, a extensão executa a criação do contacto na lista selecionada. Em caso de conflito, como duplicação de email ou número, o utilizador é notificado com uma mensagem de erro.



**Figura 8 - Ecrã de Adição de Contacto com Sincronização via WhatsApp Web (Fonte: elaboração própria).**

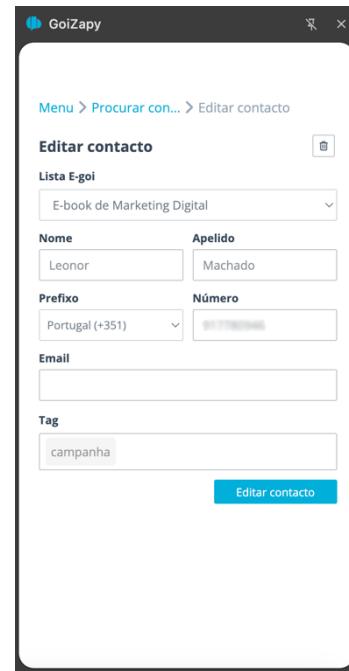
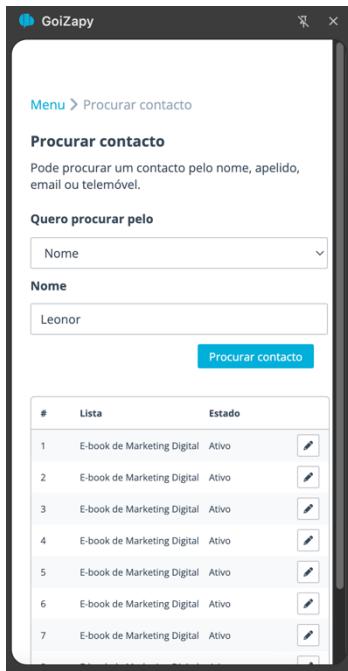
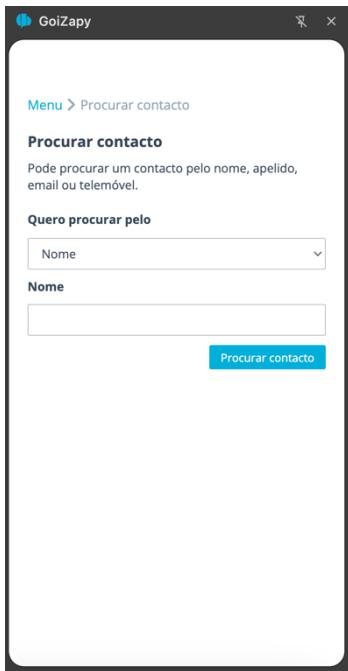
### 3.2.4.3. *Procurar contacto*

A extensão permite localizar contactos existentes através de pesquisa por nome, apelido, número de telemóvel ou email. O utilizador seleciona o critério num menu suspenso e insere o valor a consultar no campo associado (cf. Figura 9).

Ao submeter a pesquisa, a extensão percorre todas as listas da conta E-goi, executando requisições com o parâmetro definido. Os resultados são apresentados em tabela, com indicação da lista, estado do contacto e opção de edição.

A edição é iniciada através do ícone de ação. A interface de detalhe (cf. Figura 10) carrega os dados do contacto, permitindo alterações nos campos principais. A atualização executa chamadas à API para modificar os dados e redefinir as *tags* associadas.

A eliminação é realizada por desativação lógica do contacto na lista selecionada, com validação prévia do utilizador.



**Figura 9** - Ecrã de Procura de Contacto com Resultados Listados (Fonte: elaboração própria).

**Figura 10** - Ecrã de Edição de Contacto (Fonte: elaboração própria).

### 3.2.4.4. Segmentar contactos

A funcionalidade de segmentação permite aplicar filtros avançados sobre os contactos de uma lista E-goi, com base em atributos como nome, apelido, email, número, estado, data de inscrição, *tags* ou associação a automações.

Após selecionar a lista, o utilizador define os critérios de filtragem através de condições combináveis (E/OU) e operadores lógicos (igual, contém, entre outros). A extensão processa os dados localmente após consulta à API, retornando os contactos que satisfazem os critérios definidos (cf. Figura 11).

Os resultados são apresentados em tabela, e sobre eles podem ser aplicadas ações em massa, como a associação ou remoção de *tags*. A adição de *tags* permite a organização segmentada dos contactos, como também pode desencadear campanhas automatizadas, por exemplo, o envio de SMS, ao ativar *triggers* previamente definidos na plataforma E-goi. Estas operações são executadas por meio de chamadas

autenticadas à API, utilizando os identificadores dos contactos segmentados (cf. Figura 11).

Uma mensagem de confirmação indica o sucesso da ação executada (cf. Figura 12).

The figure consists of three side-by-side screenshots of a software application window titled "GoiZapy".

- Screenshot 1:** Shows the "Segmentar contactos" (Segment contacts) screen. It includes a "Lista de contactos" dropdown menu set to "Escolha uma lista" (Select a list). A descriptive text box says: "Segmente os seus contactos e aplique ações em massa como, por exemplo, adicionar uma tag." Below it is a "Nova pesquisa" (New search) section with fields for "Corresponde" (Matches), "a todas as condições" (All conditions), "Quando" (When), "Data de inscrição" (Registration date), and a dropdown set to "igual" (Equal) with the value "07/06/2025". There are also fields for "Nome" (Name) and "igual" (Equal) with the value "Leonor".
- Screenshot 2:** Shows the continuation of the segmentation process. It includes a "Ações em Massa" (Mass actions) section with a dropdown menu set to "Selecionar uma ação" (Select an action) and a "Aplicar" (Apply) button. Below it is a "Resultado" (Result) table with columns "Nome" (Name), "Email" (Email), and "Telemóvel" (Mobile). One row is shown: "Leonor Machado".
- Screenshot 3:** Shows a confirmation dialog box. The title bar says "GoiZapy". The message inside the box reads: "A extensão GoiZapy indica" (The GoiZapy extension indicates) and "Ação aplicada com sucesso!" (Action applied successfully). There is an "OK" button at the bottom.

**Figura 11 - Confirmação de Ação Aplicada (Fonte: elaboração própria).**

**Figura 12 - Ecrã de Segmentação de Contactos e Ações em Massa (Fonte: elaboração própria).**

### 3.2.4.5. Gestão de listas de contactos

A funcionalidade de gestão de listas permite ao utilizador criar, editar e eliminar listas de contactos associadas à conta E-goi, diretamente através da extensão. Cada lista é caracterizada por dois atributos, o nome interno, utilizado como identificador técnico, e o nome público, apresentado nas interfaces visuais e comunicações.

Ao aceder à interface de gestão (cf. Figura 13), o utilizador pode optar por criar uma lista ou editar uma existente. A criação de uma lista (Figura 14) exige a introdução de ambos os nomes, interno e público, sendo a informação transmitida à plataforma E-goi para que a nova lista seja registada e fique disponível para utilização em ações como campanhas, segmentações ou automações.

A edição de listas previamente criadas (cf. Figura 15) permite modificar os seus atributos identificadores de forma direta. Após seleção da lista, o utilizador pode ajustar os dados e aplicar as alterações, que são imediatamente refletidas na conta E-goi. Adicionalmente, está disponível a opção de eliminar listas, com a devida confirmação do utilizador, assegurando a remoção segura da estrutura correspondente.

**Figura 13**

Ecrã Inicial de Gestão de Listas de Contactos

**Figura 14**

Ecrã de Edição de Lista de Contactos

**Figura 15**

Formulário de Criação de Lista de Contactos

**Figura 13** - Ecrã Inicial de Gestão de Listas de Contactos (Fonte: elaboração própria).

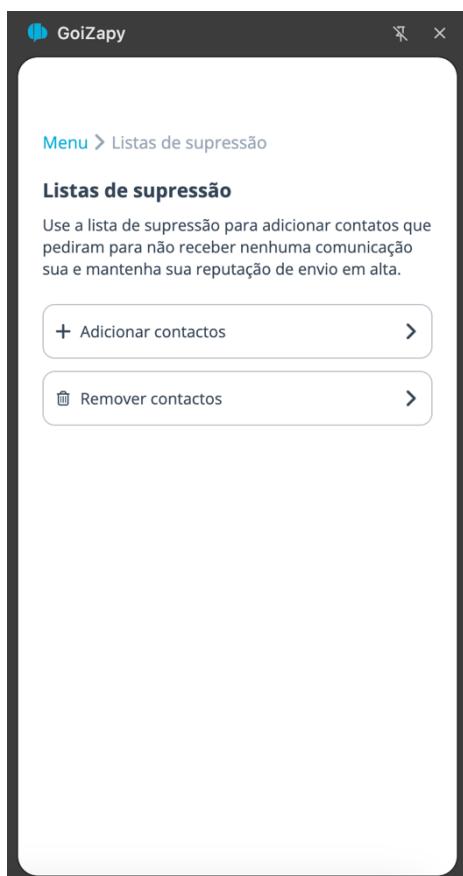
**Figura 15** - Formulário de Criação de Lista de Contactos (Fonte: elaboração própria).

**Figura 14** - Ecrã de Edição de Lista de Contactos (Fonte: elaboração própria).

### 3.2.4.6. Gestão da lista de supressão

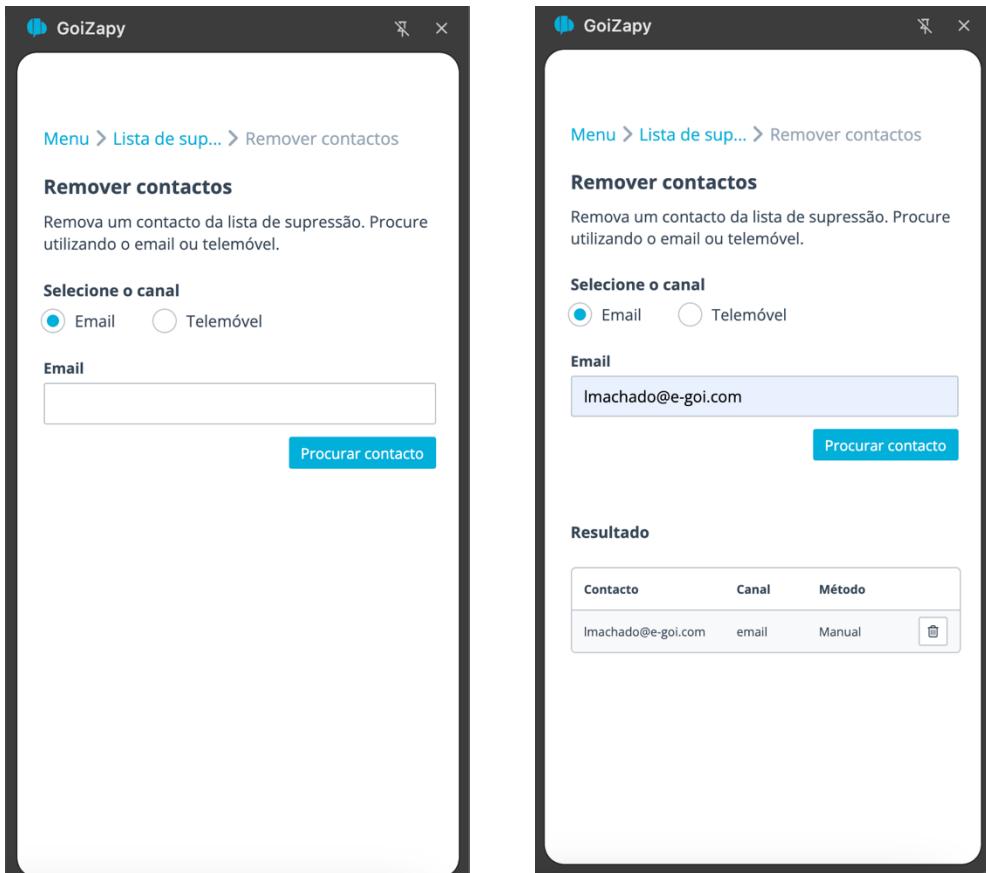
A partir da interface (cf. Figura 16), o utilizador pode aceder às opções para adicionar ou remover contactos da lista de supressão. A adição de um contacto (Figura 17) pode ser feita com base no email ou número de telemóvel. Após a submissão dos dados, o contacto é registado na lista de supressão associada à conta E-goi, ficando automaticamente excluído de futuras campanhas e comunicações. Esta operação é particularmente relevante para salvaguardar a reputação de envio e evitar envios não autorizados.

A funcionalidade de remoção (cf. Figura 18) permite reativar contactos previamente suprimidos. O utilizador pesquisa o registo por canal (email ou telemóvel) e, se encontrado, pode confirmar a sua eliminação da lista. A ação é refletida de imediato na conta E-goi, permitindo a reintegração do contacto nas operações de comunicação.



**Figura 16** - Ecrã Inicial de Gestão de Listas de Supressão (Fonte: elaboração própria).

**Figura 17** - Formulário para Inserção de Contactos na Lista de Supressão (Fonte: elaboração própria).



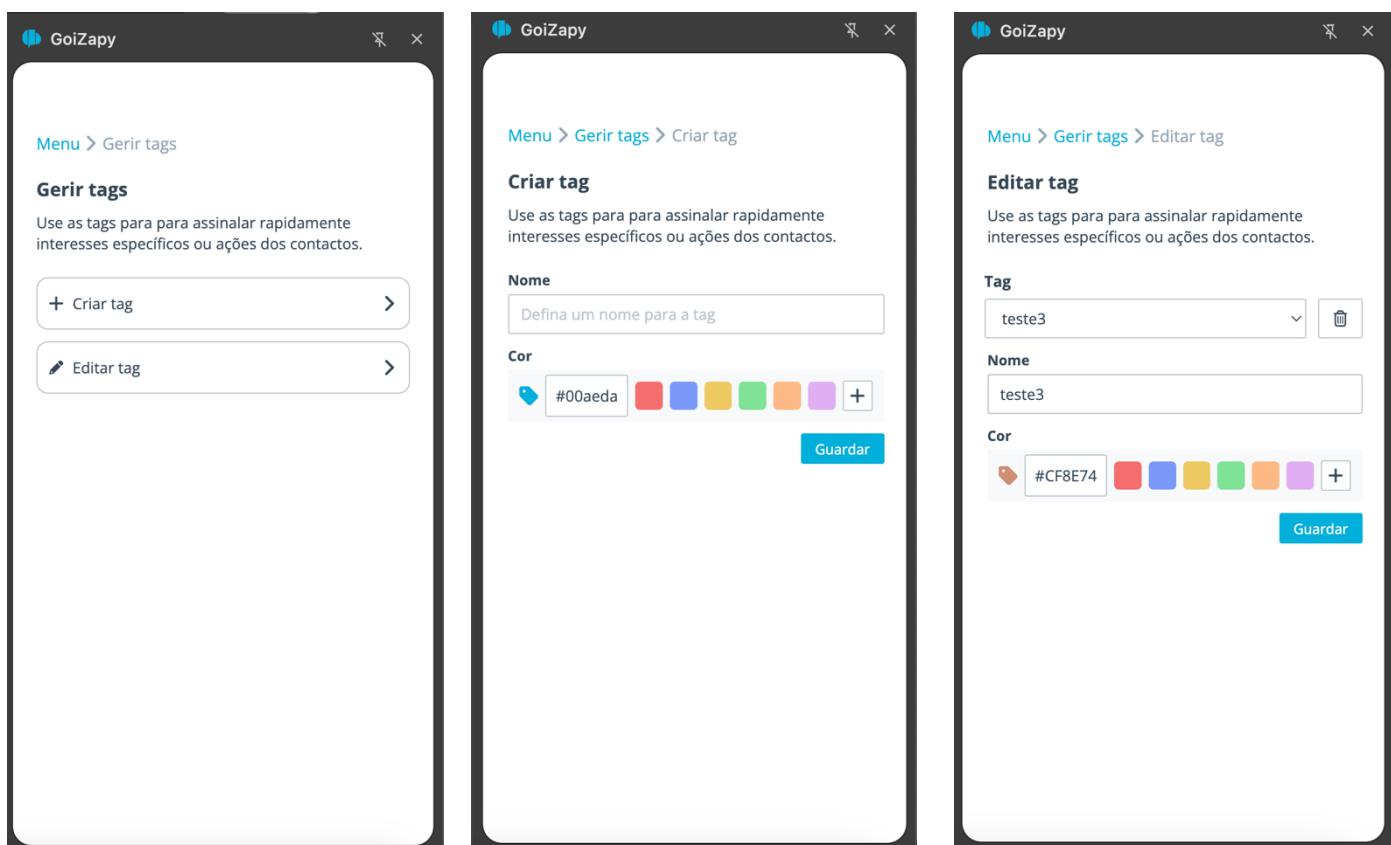
**Figura 18** - Ecrã de Remoção de Contactos da Lista de Supressão (Fonte: elaboração própria).

### 3.2.4.7. Gestão de Tags

A extensão disponibiliza um módulo para a gestão de *tags*, facilitando a criação, edição e eliminação de etiquetas associadas à conta E-goi. As *tags* funcionam como elementos de segmentação e podem ser utilizadas como gatilhos para a execução de automações na plataforma, como o envio de campanhas personalizadas.

A interface apresenta duas funcionalidades principais (cf. Figura 19): a criação de novas *tags*; e a edição de *tags* existentes. Para criar uma *tag* (cf. Figura 20), o utilizador deve definir um nome e escolher uma cor representativa a partir de uma paleta predefinida ou personalizada. Após submissão, a *tag* fica imediatamente disponível para associação a contactos ou utilização em campanhas.

No modo de edição (Figura 21), é possível selecionar uma etiqueta existente, modificar os seus atributos (nome e cor) e guardar as alterações. A interface também disponibiliza uma opção para remoção definitiva da *tag*, mediante confirmação.



**Figura 19** - Ecrã Inicial de Gestão de Tags (Fonte: elaboração própria).

**Figura 20** - Formulário de Criação de Tag (Fonte: elaboração própria).

**Figura 21** - Ecrã de Edição de Tag (Fonte: elaboração própria).

### 3.2.5. Implementação técnica

A extensão GoiZapy é composta por três blocos principais: o *content script* responsável pela extração de dados, o *service worker* que centraliza a lógica de integração com a *API* da E-goi e a interface gráfica desenvolvida com SvelteKit.

A configuração base está definida no ficheiro *manifest.json*, localizado na pasta *static*. Este ficheiro declara as permissões da extensão, os scripts que devem ser executados, os domínios permitidos e a *Content Security Policy* de acordo com o *Manifest V3*. O *manifest* identifica também o *content script* (*whatsapp-content.js*) e o *background script* (*background.js*) como componentes ativos da extensão.

A captura de dados é realizada pelo ficheiro *whatsapp-content.js*, que funciona como *content script*. É injetado na página [web.whatsapp.com](https://web.whatsapp.com) e executado no contexto do *DOM*. O *script* utiliza um *MutationObserver* para detetar alterações na interface do WhatsApp Web. Quando uma nova conversa é aberta, o script extrai nome, número de telefone e email do contacto. Estes dados são enviados para o *service worker* através de *chrome.runtime.sendMessage*.

O ficheiro *background.ts*, compilado como *background.js*, define o *service worker* da extensão. Este componente recebe os dados enviados pelo *content script* e pela interface. Após validação, encaminha os dados para a *API* da E-goi. O envio é feito por meio do SDK oficial, as operações incluem criação de contactos, aplicação de *tags* e associação a listas. A autenticação é feita através de uma chave *API* incluída nas chamadas HTTP.

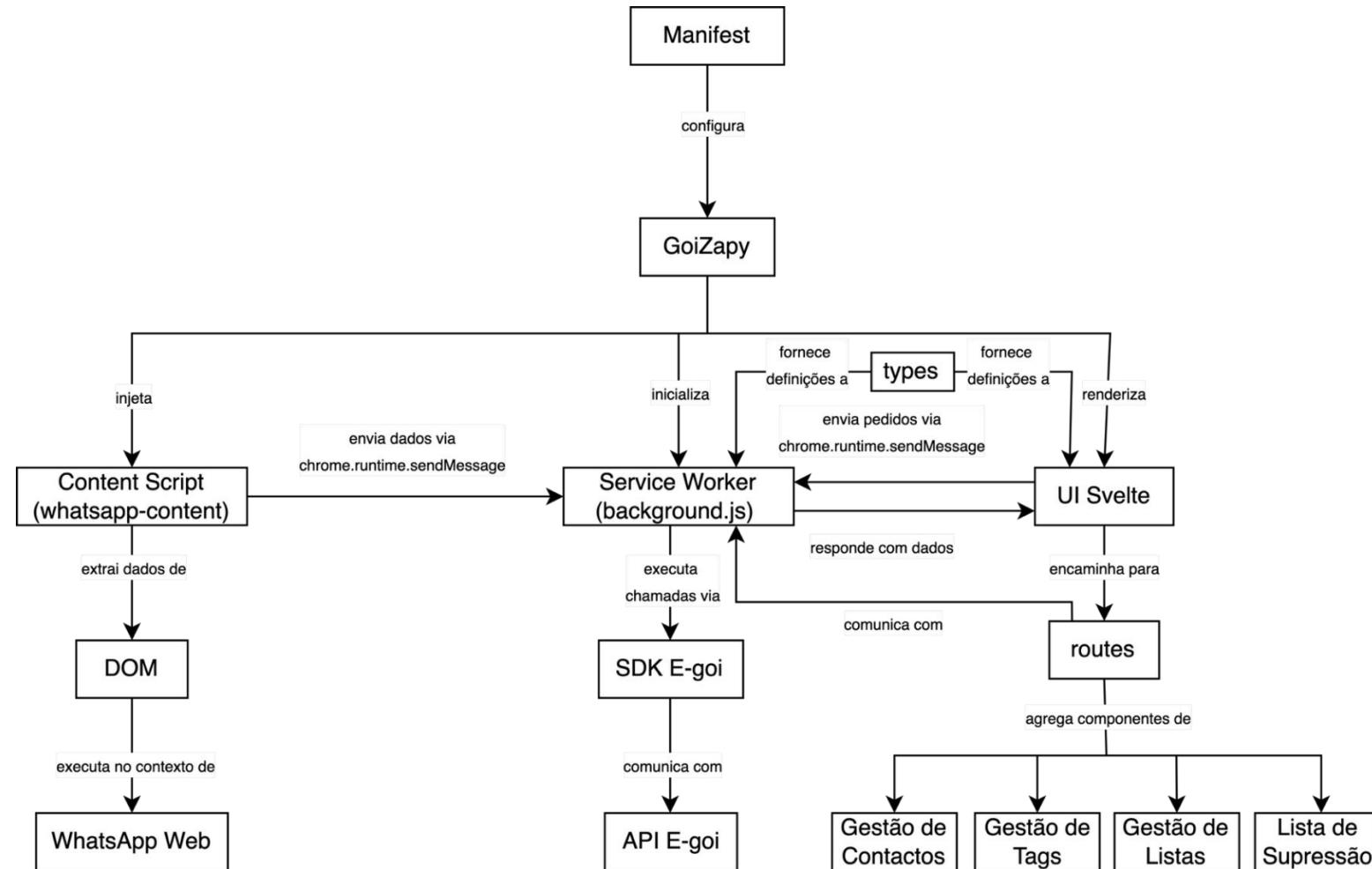
A interface da extensão é desenvolvida com SvelteKit e está organizada em subpastas dentro de *src*. Cada funcionalidade tem a sua própria pasta, com um componente *.svelte* para a interface e um ficheiro *.ts* com a lógica funcional.

A definição de tipos para comunicação com a *API* da E-goi está centralizada no ficheiro *egoisdk.d.ts*. Este ficheiro fornece as estruturas esperadas nos pedidos e nas respostas, permitindo validação em tempo de compilação.

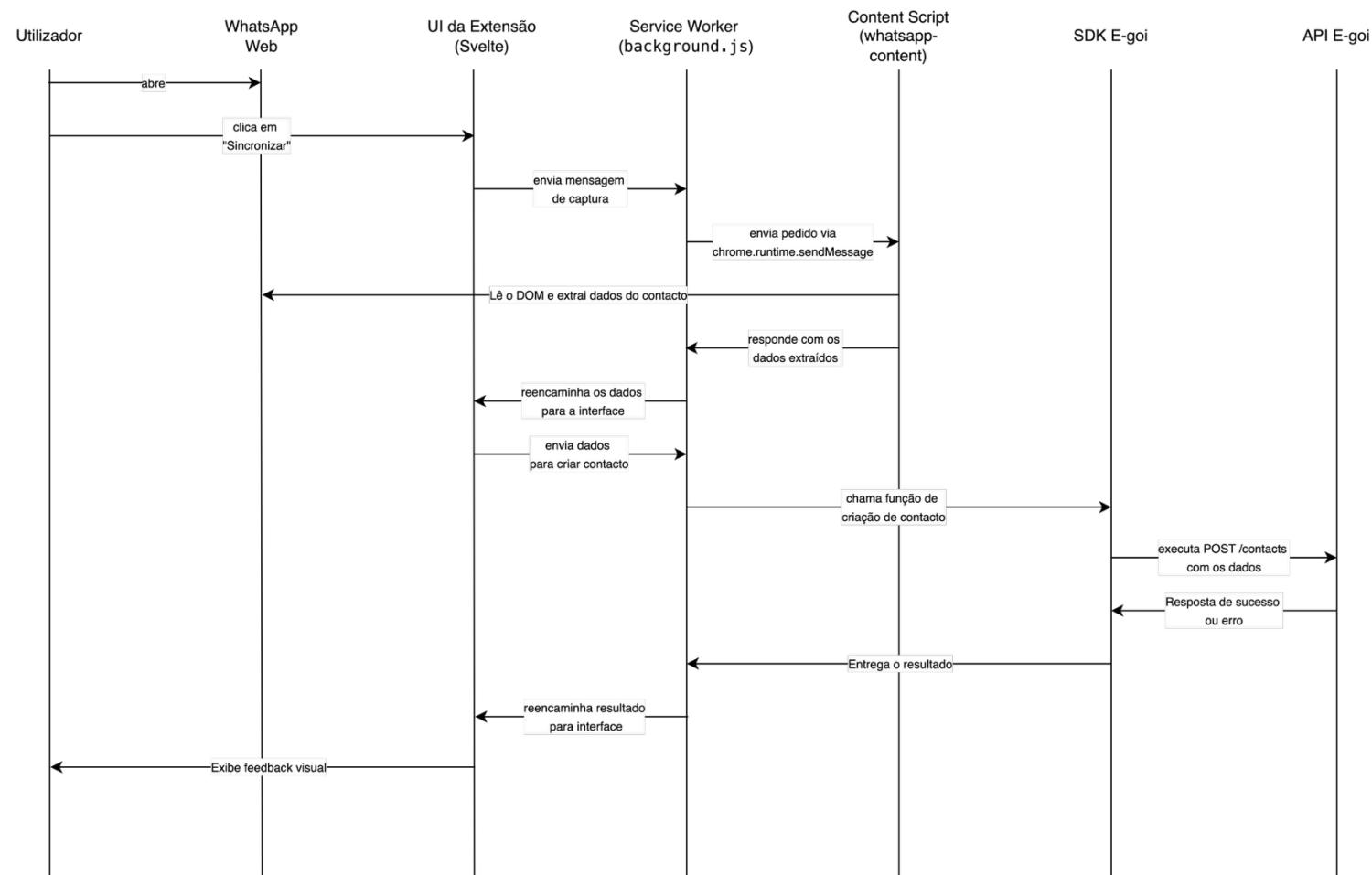
Os recursos estáticos da extensão estão agrupados na pasta *static*. Esta pasta contém o *manifest.json*, o *content script*, os ícones, as fontes e as imagens. A compilação do projeto é feita com *Vite*. Os ficheiros de saída são colocados na pasta *build*, incluindo os *bundles* da interface, do *content script* e do *service worker*.

A estrutura técnica do projeto e os fluxos de comunicação entre os seus componentes estão representados na Figura 22. O processo completo de sincronização de um

contacto, desde a ação do utilizador até ao registo na *API* da E-goi, encontra-se descrito na Figura 23.



**Figura 22** - Diagrama de Componentes da Extensão GoiZapy (Fonte: elaboração própria).



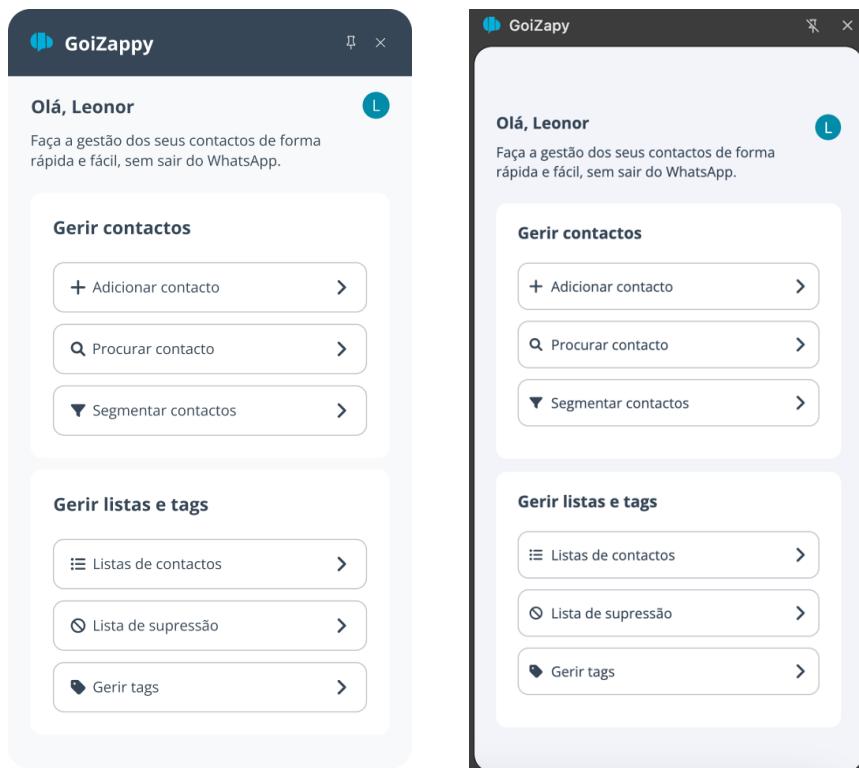
**Figura 23** - Diagrama de sequência da operação de sincronização de contacto na extensão GoiZapy (Fonte: elaboração própria).

### 3.2.6. Design da interface da extensão

A construção visual da extensão GoiZapy seguiu a identidade gráfica da E-goi, assegurando uniformidade visual e integração com o seu ecossistema digital. A interface aplica os parâmetros visuais definidos oficialmente pela E-goi, incluindo paletas de cores institucionais, hierarquias tipográficas, espaçamentos, estilos de botões e iconografia (E-goi, s.d.).

O protótipo, desenvolvido em Figma por Paula Adams (equipa MMA), encontra-se documentado nos Anexos 2 a 9. Este modelo serve de base para a implementação da interface, definindo a organização e o estilo de menus, painéis, listagens e componentes interativos. A fidelidade ao modelo visual é ilustrada na Figura 24, onde se observa a equivalência entre o protótipo em Figma e a interface final da extensão.

Todos os elementos visuais da extensão mantêm alinhamento total com os padrões gráficos da E-goi, resultando numa interface limpa, funcional e imediatamente reconhecível para os utilizadores habituados ao ambiente visual da plataforma.



**Figura 24** - Comparação entre o Protótipo da Interface Desenvolvida em Figma e a Implementação Final da Extensão GoiZapy  
(Fonte: elaboração própria).

### 3.2.7. Testes e validações

A extensão Goizapy encontra-se validada através de um conjunto de testes funcionais, técnicos e de integração, organizados nos Apêndices 1 a 18. Cada teste apresenta objetivos, entradas definidas, saídas esperadas e dependências explícitas.

- **Extração e comunicação interna (cf. Apêndices 1 e 2):** O *content script* deteta alterações no *DOM* do WhatsApp Web e extrai com o nome, número e email. A comunicação com o *service worker* ocorre com integridade estrutural e latência inferior a 1 segundo.
- **Interações com a API da E-goi (cf. Apêndices 3 a 14):** As funcionalidades de criação, edição e remoção de contactos, listas, *tags* e supressões geram chamadas HTTP com estrutura válida e respostas consistentes. A interface reflete corretamente os dados transmitidos e recebidos.
- **Autenticação e navegação cf. (cf. Apêndices 13 e 14):** A *API Key* é lida e validada em ambiente seguro. A navegação entre interfaces ocorre de forma fluida, com exibição personalizada do nome do utilizador e redirecionamento correto.
- **Segurança (cf. Apêndice 15):** A extensão mantém permissões restritas utilizando apenas HTTPS, impede *scripts inline* e evita conteúdo dinâmico executável. Não se identificam violações da *Content Security Policy*.
- **Desempenho (cf. Apêndice 16):** O uso de memória permanece abaixo de 100 MB, a utilização de CPU não ultrapassa os 30% em carga e a latência das chamadas API mantém-se abaixo de 500 ms. A interface preserva a responsividade (<50 ms de atraso).
- **Validação em ambiente real (cf. Apêndice 17):** A extensão executa com estabilidade no WhatsApp Web real, mantendo consistência funcional, integridade visual e sem falhas técnicas.
- **Build e empacotamento (cf. Apêndice 18):** O processo de *build* produz corretamente o diretório com todos os ficheiros e *bundles* necessários, garantindo compatibilidade imediata com o Chrome.

Todos os testes indicam conformidade funcional, técnica e visual. A extensão opera dentro dos parâmetros definidos, com segurança, desempenho e integração assegurados.

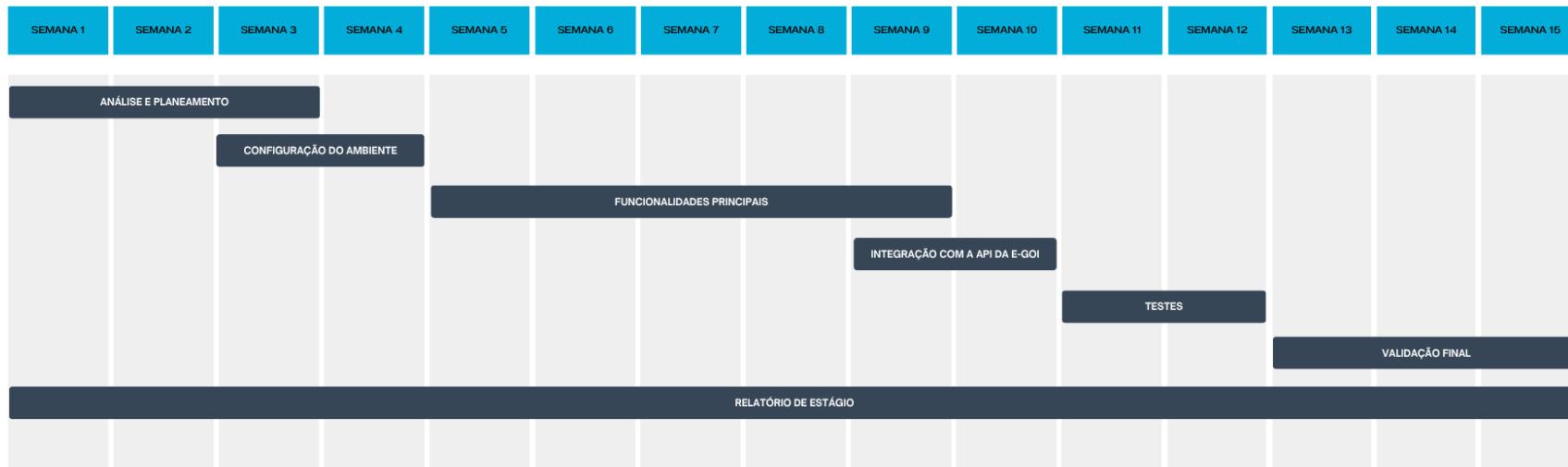
### 3.2.8. Cronograma e desenvolvimento

O desenvolvimento do projeto Goizapy decorreu ao longo de 15 semanas, seguindo uma abordagem incremental e iterativa. As tarefas técnicas evoluem de forma contínua, acompanhadas em paralelo pela elaboração do relatório de estágio, assegurando a documentação com o progresso funcional. Na Figura 25 apresenta-se o cronograma do projeto, estruturado por fases semanais que cobrem desde o planeamento até à validação final da extensão.

Durante as semanas 1 e 3, decorreu a fase de análise e planeamento, onde são definidos os requisitos funcionais, estudada a *API* da E-goi e estabelecida a arquitetura da extensão, composta por *content script*, *service worker* e interface. Em simultâneo, é iniciada a estruturação do relatório de estágio, com a introdução e definição dos objetivos. Nas semanas 3 e 4, realizou-se a configuração do ambiente de desenvolvimento, através da integração de TypeScript, Svelte, definição de permissões no ficheiro *manifest.json* e preparação do contexto de execução da extensão no Chrome. Esta fase dá suporte ao enquadramento técnico do relatório.

Entre as semanas 5 e 9, desenvolveu-se as funcionalidades principais, com foco na extração de dados do *DOM* do WhatsApp Web, comunicação entre *scripts* e autenticação com a *API Key*. A interface gráfica começa a ganhar forma, permitindo os primeiros fluxos de interação funcional. Esta implementação é acompanhada da respetiva documentação técnica no relatório. A integração com a *API* da E-goi ocupa as semanas 9 a 10, com a implementação de operações *CRUD* sobre contactos, listas, *tags* e listas de supressão. Esta integração garante o envio correto de requisições e a sincronização dos dados com a interface, sendo igualmente documentada na secção funcional do relatório.

Nas semanas 11 e 12, foram conduzidos testes técnicos, abrangendo aspectos funcionais, desempenho, segurança e comunicação. As validações incluem conformidade com a *Content Security Policy*, tempos de resposta da interface e uso de recursos do navegador. As semanas 13 a 15 são dedicadas à revisão do código, validação em ambiente real, com simulação de interações no WhatsApp Web, verificação dos fluxos completos e análise da consistência visual com o protótipo. O relatório de estágio é revisto e concluído.



**Figura 25** - Cronograma de Desenvolvimento do Projeto Goizapy (Fonte: elaboração própria)

## 4. Conclusão

A quarta e última secção do relatório sintetiza os principais resultados alcançados com o desenvolvimento do projeto GoiZapy, destacando o grau de concretização dos objetivos propostos e a profundidade da solução implementada. São identificadas as principais dificuldades enfrentadas durante o processo, bem como as estratégias adotadas para as ultrapassar. Em seguida, apresentam-se potenciais melhorias, com destaque para a escalabilidade da captação de dados. Por fim, é refletido o contributo pessoal da autora no projeto, com ênfase nas aprendizagens técnicas e metodológicas adquiridas em contexto real de desenvolvimento.

### 4.1. Resultados obtidos

O projeto culmina no desenvolvimento de uma extensão para o Google Chrome que estabelece uma integração direta entre o WhatsApp Web e a plataforma E-goi, permitindo a extração de dados de contacto e a sua sincronização com a API da E-goi. A solução responde ao objetivo de interligar sistemas de comunicação digital com ferramentas de automação de marketing, colmatando a inexistência de uma solução nativa que articule estes dois domínios (E-goi, 2024a).

A extensão implementa mecanismos de leitura do *DOM* para recolha de dados em tempo real, acompanhados de operações autenticadas de criação, atualização e organização de contactos, listas, *tags* e exclusões. A inclusão programática de *tags* viabiliza a entrada imediata de novos contactos em fluxos automatizados, permitindo o acionamento direto de campanhas a partir da interface do WhatsApp Web.

A arquitetura, baseada em *content scripts* e *service workers*, cumpre as restrições do *Manifest V3*, assegurando isolamento funcional, segmentação lógica e conformidade com as diretivas de segurança. A interface, desenvolvida com SvelteKit, disponibiliza controlo integral sobre os recursos da conta E-goi, com operações *CRUD* completas e comunicação validada em tempo real.

Os testes realizados confirmam a estabilidade da extensão em contexto de produção, com latência reduzida, consumo otimizado de recursos e tolerância a alterações na estrutura do WhatsApp Web. A solução demonstra aderência integral aos requisitos definidos, garantindo um fluxo contínuo entre a captação de dados e a automatização de ações de marketing, conforme previsto na conceção do projeto.

#### 4.2. Dificuldades e soluções

A principal limitação decorreu da inexistência de uma *API* pública no WhatsApp Web, o que obrigou à extração de dados diretamente a partir da estrutura *DOM* da interface. Esta abordagem requer a implementação de mecanismos de deteção reativa de mutações no *DOM* e a verificação contínua da consistência dos seletores utilizados, face à natureza dinâmica da aplicação.

Em paralelo, o *Manifest V3* impõe restrições estruturais, nomeadamente a eliminação de *scripts* de *background* persistentes e a proibição de execução de código dinâmico. A arquitetura da extensão teve, por isso, de ser redesenhada com base em *service workers*, comunicação assíncrona entre contextos isolados e segmentação de responsabilidades.

Inicialmente, prevêu-se a utilização da API interna (*services*) da E-goi como camada de integração, dado o seu grau de abstração e suporte a fluxos internos da plataforma. Contudo, a impossibilidade de incluir elementos dinâmicos ou *scripts* externos no contexto da extensão inviabilizou essa abordagem. Esta limitação exigiu a reformulação da estratégia de integração e a adoção da *API* pública v3, compatível com as diretrizes de segurança do *Manifest V3* e adequada à comunicação direta a partir do *service worker*.

A integração com a *API* v3 implicou, adicionalmente, a gestão de quotas de utilização, autenticação segura e tratamento de falhas, exigindo lógica de controlo e tolerante a erros. A solução implementada baseia-se em ciclos iterativos de validação modular e ajustamento progressivo, assegurando estabilidade, conformidade e adaptabilidade às limitações impostas pelo ecossistema envolvido.

#### 4.3. Melhorias futuras

Prevê-se a introdução de um mecanismo de captação em massa, capaz de percorrer automaticamente várias conversas no WhatsApp Web e extrair contactos de forma sequencial, eliminando a dependência da deteção manual por interação. Esta funcionalidade visa maximizar a cobertura e acelerar o processo de aquisição de dados, especialmente em cenários com elevado volume de comunicações.

Em complemento, propõe-se a verificação imediata da existência de cada contacto nas diferentes listas da conta E-goi, através de chamadas prévias à API.

A implementação destas melhorias requer o reforço da camada lógica de integração, com suporte a requisições assíncronas em lote, gestão de respostas e sincronização incremental com os recursos existentes. Adicionalmente, a geração de relatórios,

discriminando contactos novos, duplicados ou rejeitados, poderá fornecer visibilidade operacional sobre o processo de captação.

Estas evoluções visam aumentar a autonomia da extensão, reduzir a carga manual e alinhar a sua operação com as exigências de escalabilidade e precisão próprias de contextos empresariais.

#### **4.4. Contributo pessoal e aprendizagens**

A participação no desenvolvimento do projeto GoiZapy constituiu uma oportunidade para aplicar, de forma integrada e orientada a objetivos, competências adquiridas ao longo da licenciatura em Engenharia Informática. O envolvimento direto em todas as fases, desde a definição da arquitetura até à integração com a *API* da E-goi, possibilitou uma experiência prática alinhada com requisitos técnicos exigentes e com a realidade de projetos aplicados.

Do ponto de vista formativo, destaca-se a aprendizagem de uma *framework* até então não explorada, *Svelte*, bem como o domínio do seu ecossistema, nomeadamente a integração com *TypeScript*, a gestão de estado reativo e a organização modular de interfaces. Paralelamente, foi consolidada experiência no desenvolvimento de extensões para navegador, com foco no cumprimento do *Manifest V3*, e na leitura e extração de dados diretamente a partir do *DOM* de aplicações *client-side*.

A experiência exigiu ainda a implementação de mecanismos assíncronos de comunicação entre contextos isolados, o consumo estruturado de *APIs REST* e a validação das operações em ambiente real. A capacidade de adaptação a limitações técnicas e a autonomia na tomada de decisões revelaram-se importantes para a resolução de problemas concretos durante o desenvolvimento.

O projeto traduziu-se, assim, numa consolidação técnica, permitindo a produção de uma solução funcional e segura com o contexto profissional em que se insere.

## Referências bibliográficas

- Agarwal, S. (2022). *Helping or hindering? How browser extensions undermine security*. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22).
- E-goi. (2023a). *Manual do Sistema de Gestão Integrado – Versão 3* [Documento interno]. E-goi, Lda.
- E-goi. (2023b). *Manual de Funções – Versão 3* [Documento interno]. E-goi, Lda.
- E-goi. (2024a). *Proposta de estágio/projeto em ambiente empresarial: Projeto 05 – GoiZapy – Desenvolvimento de Plugin de Integração entre E-goi e WhatsApp* [Documento interno]. Escola Superior de Ciência e Tecnologia.
- E-goi. (2024b). *E-goi API v3 documentation*. <https://developers.e-goi.com/api/v3/>
- E-goi. (s.d.). *E-goi Design System Documentation*. Recuperado de <https://docs.egoiaapp.com/?path=/docs/development-bo-ui-docs--docs>
- Esiyok, I., Berrang, P., Cohn-Gordon, K., & Künemann, R. (2023). *Accountable JavaScript code delivery*. arXiv.
- Google Chrome Developers. (2024). *Manifest V3 documentation*. Recuperado de <https://developer.chrome.com/docs/extensions/mv3/>
- Jannah, R. (2023). *Utilization of WhatsApp Business in marketing strategy to increase the number of sales through direct interaction with customers*. Syntax Idea, 5(4), 489–491.
- Kariyaa, A., Savino, G.-L., Stellmacher, C., & Schöning, J. (2021). *Understanding Users' Knowledge about the Privacy and Security of Browser Extensions*. Proceedings of the Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021). Disponível em: <https://www.usenix.org/conference/soups2021/presentation/kariyaa>
- Nayak, A., Khandelwal, R., Fernandes, E., & Fawaz, K. (2024). *Experimental security analysis of sensitive data access by browser extensions*. In

Proceedings of the ACM Web Conference 2024 (WWW '24), May 13–17, 2024, Singapore. ACM.

Pantelaios, D., & Kapravelos, A. (2024). *Browser extensions under the microscope: Understanding malicious behavior and Manifest V3 migration*.

Plain English. (2022). *Making a Chrome extension with Svelte*. Recuperado de <https://javascript.plainenglish.io/making-a-chrome-extension-with-svelte-2fefb3769c>

Plain English. (2023). *Fetch data in Chrome Extension V3*. Recuperado de <https://javascript.plainenglish.io/fetch-data-in-chrome-extension-v3-2b73719ffc0e>

Rastogi, N., & Hendler, J. (2017). *WhatsApp security and role of metadata in preserving privacy*. In Proceedings of the 12th International Conference on Cyber Warfare and Security (ICCWS).

RD Station. (2024a). *Adicionar extensão WhatStation*. Recuperado de [https://ajuda.rdstation.com/s/article/Adicionar-extens%C3%A3o-What-Station?language=pt\\_BR](https://ajuda.rdstation.com/s/article/Adicionar-extens%C3%A3o-What-Station?language=pt_BR)

RD Station. (2024b). *Integração com o WhatsApp Web*. Recuperado de [https://ajuda.rdstation.com/s/article/Integra%C3%A7%C3%A3o-com-o-WhatsApp-Web?language=pt\\_BR](https://ajuda.rdstation.com/s/article/Integra%C3%A7%C3%A3o-com-o-WhatsApp-Web?language=pt_BR)

RD Station. (2024c). *Como integrar RD Station com WhatsApp via Pluga*. Recuperado de <https://blog.pluga.co/rd-station-whatsapp-tutorial/>

Reis, J. F. (2018). *Implementação de um sistema de CRM para otimização de processos internos* [Dissertação de mestrado, Faculdade de Engenharia da Universidade do Porto]. Repositório Aberto da Universidade do Porto.

Schaffner, B., Brohn, A., Chee, J., Feng, K. J., & Chetty, M. (2024). *Designing and testing a mobile application for collecting WhatsApp chat data while preserving privacy*. arXiv.

Somé, D. F. (2019). *EmPoWeb: Empowering web applications with browser extensions*. arXiv.

Svelte. (s.d.). *SvelteKit documentation*. Recuperado de <https://svelte.dev/docs/kit/routing>

Tenório Junior, N. N., Pereira, K. R. F., Oliveira, M., Menegassi, C. H. M., & Bento, T. P. (2020). A gestão do conhecimento no marketing digital: uma investigação no uso do WhatsApp. *Revista Inteligência Competitiva*, 9(4), 39–47.

Trentbrew. (2023). *Svelte 5 Chrome extension example*. Recuperado de <https://github.com/trentbrew/svelte5-chrome-extension>

WhatsApp. (n.d.). *WhatsApp Web*. <https://web.whatsapp.com/>

Zarouali, B., Brosius, A., Helberger, N., & de Vreese, C. H. (2021). WhatsApp marketing: A study on WhatsApp brand communication and the role of trust in self-disclosure. *International Journal of Communication*, 15, 252–276.

## Glossário

2FA	Autenticação em duas etapas, aumenta a segurança de acesso.
API Key	Chave de autenticação utilizada para acesso seguro a APIs.
API REST	Interface que permite a comunicação entre sistemas usando o protocolo HTTP e dados em JSON.
Automação de Marketing	Processo de envio automatizado de comunicações com base em comportamentos ou regras definidas.
Carrinho Abandonado	Situação em que um utilizador adiciona produtos a um carrinho de compras online, mas não conclui a compra.
Chrome Extensions APIs	Conjunto de interfaces de programação disponibilizadas pelo navegador Google Chrome para criar extensões com acesso controlado a funcionalidades do browser.
CI/CD	Práticas de integração e entrega contínuas utilizadas em ciclos de desenvolvimento de software.
Client-side	Modelo de execução em que a lógica de uma aplicação ocorre no navegador do utilizador, sem processamento no servidor.
Content Script	Script inserido por uma extensão no contexto de uma página web, com acesso ao DOM mas isolado do código interno da aplicação.
CSP	Política de segurança que restringe a execução de conteúdo web potencialmente perigoso.

DOM	Estrutura hierárquica de uma página web, manipulável via scripts no navegador.
E-goi	Plataforma multicanal de automação de marketing, especializada na gestão de contactos, campanhas e fluxos automatizados.
Event-based	Modelo baseado em eventos, no qual mudanças de estado ou ações específicas disparam automaticamente reações ou fluxos.
Extensão Chrome	Aplicação instalada no navegador Chrome que adiciona funcionalidades às páginas web.
FS	Propriedade criptográfica que impede a recuperação de sessões passadas mesmo que as chaves atuais sejam comprometidas.
Funil de Vendas	Representação do percurso do consumidor desde o primeiro contacto com a marca até à conversão final.
GoiZapy	Extensão desenvolvida para integrar contactos do WhatsApp Web com a plataforma E-goi.
I&D	Conjunto de atividades voltadas à criação e evolução de produtos, processos ou serviços.
JSON	Formato de troca de dados baseado em texto, estruturado em pares chave-valor.
Landing Page	Página web desenvolvida com objetivo de conversão, acedida através de campanhas.
Manifest V3	Terceira versão do ficheiro de manifesto usado em extensões Chrome, definindo permissões, scripts e funcionalidades.

Marketing Relacional	Estratégia centrada na criação de relações duradouras e personalizadas com clientes.
MutationObserver	API JavaScript usada para detetar alterações no DOM.
Push Notification	Notificação enviada diretamente para o dispositivo do utilizador (mobile ou browser), mesmo que não esteja a usar a aplicação.
Rate Limit	Límite imposto ao número de requisições que podem ser feitas a uma API num determinado intervalo de tempo.
Requisição HTTP	Pedido enviado por um cliente (como um navegador) a um servidor, usando métodos como GET, POST, PUT, DELETE e PATCH.
SaaS	Modelo de distribuição de software baseado na nuvem, acessível via Internet.
Script	Código executado automaticamente para realizar tarefas específicas numa página web.
SDK	Conjunto de ferramentas de desenvolvimento que facilita a integração com uma plataforma específica, como a API da E-goi.
Segmentação	Divisão de contactos em grupos com características ou interesses comuns, para comunicação dirigida.
Service Worker	Script de fundo em extensões Chrome que gera operações assíncronas, comunicação com APIs externas e eventos, com duração limitada e ativação baseada em eventos com ciclo de vida efémero.
SMS	Serviço de envio de mensagens de texto por telemóvel.

SOP	Regra de segurança que restringe interações entre origens diferentes.
SvelteKit	<i>Framework</i> baseado em Svelte para o desenvolvimento de aplicações web dinâmicas e rápidas.
Tags	Etiquetas associadas a contactos para organização e ativação de ações automáticas.
Teste A/B	Técnica de experimentação que compara duas versões de conteúdo para determinar qual gera melhores resultados.
TLS	Protocolo de segurança para comunicação encriptada pela Internet, garantindo confidencialidade e integridade dos dados.
Triggers	Ações automáticas iniciadas por eventos específicos ou pela aplicação de tags.
TypeScript	Linguagem de programação que estende o JavaScript com tipagem estática.
URI	Identificador uniforme de recursos, usado para localizar ou referenciar elementos na web.
Vite	Ferramenta de build e bundling para aplicações web modernas, focada em desempenho e experiência de desenvolvimento.
WebSocket	Protocolo de comunicação bidirecional que permite troca contínua de dados entre cliente e servidor em tempo real.
WhatsApp Business	Versão comercial do WhatsApp, direcionada a empresas, com funcionalidades adicionais como catálogo de produtos e horário de funcionamento.

WhatsApp Web

Versão do WhatsApp acessível via navegador, que permite gerir conversas no desktop.

XSS

Tipo de vulnerabilidade que permite a execução de scripts maliciosos no navegador de outro utilizador.

## Apêndices

### Caso de Teste 1: extração de dados do DOM do WhatsApp Web

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-DOM-001
<b>Objetivo:</b>	Validar a capacidade do content script para detetar, extrair e validar corretamente os dados de contacto presentes no DOM do WhatsApp Web.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. Aceder ao WhatsApp Web.</li> <li>b. Abrir uma ou mais conversas com contactos no painel de detalhes do WhatsApp.</li> <li>c. Alternar entre diferentes conversas.</li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. Nome, número do contacto e email são corretamente extraídos do DOM.</li> <li>b. A deteção reativa é confirmada com base em alterações estruturais (MutationObserver).</li> <li>c. Conteúdos incompletos, campos vazios ou com estrutura inválida são ignorados.</li> <li>d. O número de telefone é validado segundo o formato internacional (ex: +351XXXXXXXX).</li> </ul>	
<b>Outros</b> O WhatsApp Web deve estar carregado com pelo menos um contacto visível na interface (painel de detalhes).	
<b>Dependências</b> Este teste está diretamente relacionado com a comunicação entre o content script e o service worker (GOIZAPY-CT-MSG-002) e com a posterior submissão dos dados extraídos para a API da E-goi (GOIZAPY-CT-API-003).	

**Apêndice 1-** Especificação do Caso de Teste para extração de dados do DOM do WhatsApp Web (Fonte: elaboração própria)

**Caso de Teste 2: comunicação entre content script e service worker**

<b>ESPECIFICAÇÃO DE CASO DE TESTE</b>	
<b>Identificador:</b>	GOIZAPY-CT-MSG-002
<b>Objetivo:</b>	Verificar se a troca de mensagens entre o content script e o service worker ocorre corretamente, sem perdas ou atrasos.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. O content script extrai dados do DOM do WhatsApp Web.</li> <li>b. Envia um payload com nome e número para o service worker via chrome.runtime.sendMessage.</li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. O service worker recebe o payload sem erro de estrutura.</li> <li>b. É emitida uma resposta de sucesso ou erro, conforme o estado do processamento.</li> <li>c. O tempo entre o envio e a resposta é inferior a 1 segundo.</li> </ul>	
<b>Outros</b> Requer a execução simultânea do content script e do service worker no ambiente controlado da extensão.	
<b>Dependências</b> Este teste depende da extração correta dos dados pelo content script (GOIZAPY-CT-DOM-001) e é pré-requisito para testes que envolvem envio de dados à API da E-goi, como GOIZAPY-CT-API-003 e GOIZAPY-CT-API-005.	

**Apêndice 2-** Especificação do Caso de Teste para comunicação entre content script e service worker (Fonte: elaboração própria)

### Caso de Teste 3: edição de contacto na API da E-goi

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-API-003
<b>Objetivo:</b>	Validar a edição de um contacto existente através da API da E-goi, incluindo atualização de dados pessoais, remoção e reaplicação de tags.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. Contacto já existente é carregado na interface através da URL com parâmetro data (via onMount).</li> <li>b. O utilizador ativa o modo de edição e modifica dados como nome, apelido, email ou número de telemóvel.</li> <li>c. São selecionadas novas tags e confirmada a intenção de guardar.</li> <li>d. Requisições executadas:           <ul style="list-style-type: none"> <li>• PATCH /lists/{list_id}/contacts/{contact_id} com o corpo base</li> <li>• POST /lists/{list_id}/contacts/actions/detach-all-tags</li> <li>• POST /lists/{list_id}/contacts/actions/attach-tag para cada tag selecionada</li> </ul> </li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. A API responde com código HTTP 200/204 em cada chamada.</li> <li>b. Os dados alterados do contacto (nome, número, email) são persistidos corretamente na plataforma.</li> <li>c. Todas as tags anteriores são removidas com sucesso e substituídas pelas novas seleções.</li> <li>d. A interface apresenta mensagem de sucesso e retorna ao modo de visualização.</li> </ul>	
<b>Outros</b> NA	
<b>Dependências</b> Requer execução bem-sucedida de GOIZAPY-CT-API-014 (armazenamento e validação da API Key) e GOIZAPY-CT-MSG-002 (comunicação entre content script e service worker). É base para GOIZAPY-CT-API-004 e GOIZAPY-CT-API-005.	

**Apêndice 3-** Especificação do Caso de Teste para edição de contacto na API da E-goi (Fonte: elaboração própria)

### Caso de Teste 4: criação manual de contacto com tags

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-API-004
<b>Objetivo:</b>	Verificar a funcionalidade de criação de contacto manualmente na API da E-goi através da interface da extensão, incluindo a associação de tags.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ol style="list-style-type: none"> <li>O utilizador preenche os campos de nome, apelido, telemóvel (com indicativo), email e seleciona a lista E-goi.</li> <li>O utilizador seleciona uma ou mais tags.</li> <li>Clica no botão “Adicionar à lista E-goi”.</li> <li>O corpo do pedido é enviado com os dados estruturados para o endpoint POST /contacts.</li> </ol>	
<b>Especificação de Saídas</b> <ol style="list-style-type: none"> <li>A API responde com código 201 (Created) e retorna o contact_id do novo contacto.</li> <li>A interface apresenta a mensagem "Contacto criado com sucesso!".</li> <li>As tags selecionadas são corretamente associadas via POST /contacts/actions/attach-tag.</li> <li>A interface reinicia os campos para novo registo.</li> </ol>	
<b>Outros</b> <ul style="list-style-type: none"> <li>O teste valida também o comportamento em caso de conflito de email (erro 409), alertando o utilizador.</li> <li>Requer API Key válida armazenada localmente. Depende do carregamento bem-sucedido de listas, países e tags.</li> </ul>	
<b>Dependências</b> <p>Depende do teste GOIZAPY-CT-API-003 (integração geral com API) para garantir que os endpoints estão operacionais. Depende da validação da API Key através do GOIZAPY-CT-API-014. Requer também o carregamento de listas, países e tags.</p>	

**Apêndice 4-** Especificação do Caso de Teste para criação manual de contacto com tags  
(Fonte: elaboração própria)

## Caso de Teste 5: criação automática de contacto com dados do WhatsApp

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-API-005
<b>Objetivo:</b>	Validar a criação automática de um contacto na E-goi com base nos dados captados diretamente do WhatsApp Web, garantindo a correta extração, estruturação, envio e resposta da API.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. O utilizador clica no botão “Sincronizar” na interface da extensão.</li> <li>b. É enviado um pedido ao content script ativo no WhatsApp Web para obter nome, número e email.</li> <li>c. Os dados extraídos do DOM do WhatsApp são usados para preencher automaticamente o formulário.</li> <li>d. Após confirmação do utilizador, o botão “Adicionar à lista E-goi” envia os dados para a API REST da E-goi:           <ul style="list-style-type: none"> <li>• POST /lists/{list_id}/contacts</li> <li>• POST /lists/{list_id}/contacts/actions/attach-tag</li> </ul> </li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. O contacto é criado com os dados extraídos do WhatsApp.</li> <li>b. As tags selecionadas são associadas ao contacto criado.</li> <li>c. Caso o contacto já exista (ex.: conflito de email), é exibido alerta apropriado.</li> <li>d. A interface apresenta mensagens de sucesso, erro de duplicação ou falhas de rede.</li> </ul>	
<b>Outros</b> <ul style="list-style-type: none"> <li>• O contacto é criado corretamente sem intervenção manual do utilizador nos campos.</li> <li>• Dados extraídos do DOM do WhatsApp são corretos e completos (nome, número e, se disponível, email).</li> <li>• O tempo entre clique e criação deve ser inferior a 3 segundos (em condições normais).</li> <li>• Tags são aplicadas sem falhas</li> </ul>	
<b>Dependências</b> <p>Este teste integra os testes de comunicação entre scripts (GOIZAPY-CT-MSG-002), criação de contacto via API (GOIZAPY-CT-API-003) e deteção no DOM do WhatsApp Web (GOIZAPY-CT-DOM-001).</p>	

**Apêndice 5-** Especificação do Caso de Teste para criação automática de contacto com dados do WhatsApp (Fonte: elaboração própria)

### Caso de Teste 6: criação de listas de contactos

<b>ESPECIFICAÇÃO DE CASO DE TESTE</b>	
<b>Identificador:</b>	GOIZAPY-CT-API-006
<b>Objetivo:</b>	Verificar a criação de uma nova lista de contactos na conta E-goi do utilizador, garantindo a correta submissão de dados via API e a atualização do estado da interface.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. O utilizador insere os valores nos campos "Nome público" e "Nome interno" através do formulário da extensão.</li> <li>b. Ao clicar no botão "Adicionar", é acionada a função createList().</li> <li>c. A extensão envia uma requisição HTTP POST para o endpoint de criação de listas da API da E-goi, contendo os dados introduzidos pelo utilizador.</li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. A API responde com status 201 Created e os dados da nova lista.</li> <li>b. A nova lista aparece imediatamente na listagem da interface (recarregamento com loadLists()).</li> <li>c. Se os campos estiverem vazios, o pedido não é enviado.</li> <li>d. Em caso de erro da API, é apresentado um alerta ao utilizador.</li> </ul>	
<b>Outros</b> <ul style="list-style-type: none"> <li>• Lista criada com nomes válidos é persistida na conta E-goi.</li> <li>• Campos obrigatórios validados antes do envio (evita POST com strings vazias).</li> <li>• Feedback visível e imediato ao utilizador (sucesso ou erro).</li> <li>• A nova lista aparece na interface sem necessidade de recarregar a página manualmente.</li> </ul>	
<b>Dependências</b> Depende da validação da API Key no GOIZAPY-CT-API-014 e do funcionamento do endpoint de criação de listas.	

**Apêndice 6-** Especificação do Caso de Teste para criação de listas de contactos (Fonte: elaboração própria)

### Caso de Teste 7: edição e remoção de listas

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-API-007
<b>Objetivo:</b>	Verificar se a funcionalidade de edição e remoção de listas de contactos da API E-goi é executada corretamente através da extensão.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b>	
a. O utilizador seleciona uma lista existente através do campo dropdown. b. Os campos "Nome público" e "Nome interno" são editados manualmente. c. A função saveEdit() é acionada ao clicar no botão “Guardar alterações”. d. A função deleteList() é acionada ao clicar no botão de eliminar, após confirmação. e. As requisições são enviadas à API da E-goi utilizando os métodos PATCH e DELETE.	
<b>Especificação de Saídas</b>	
a. A lista selecionada é atualizada com os novos dados introduzidos, sem erros de validação. b. Caso a lista seja removida, ela desaparece da interface e da resposta da API. c. O utilizador é notificado em caso de sucesso ou falha da operação. d. A lista de opções é atualizada após as alterações.	
<b>Outros</b>	
<ul style="list-style-type: none"> <li>O sistema exige a presença de uma API Key válida armazenada localmente para autenticar as requisições.</li> </ul>	
<b>Dependências</b>	
Depende do carregamento correto das listas (GOIZAPY-CT-API-006) e da API Key válida previamente armazenada (GOIZAPY-CT-API-014).	

**Apêndice 7-** Especificação do Caso de Teste para edição e remoção de listas (Fonte: elaboração própria)

### Caso de Teste 8: adição à lista de supressão

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-API-008
<b>Objetivo:</b>	Verificar se a adição de contactos à lista de supressão é efetuada corretamente através da API da E-goi, utilizando os canais email e telemóvel.
<b>Autor(es)</b>	Leonor Carneiro Machado
<p><b>Especificação de Entradas</b></p> <p>a. O utilizador seleciona o tipo de supressão (email ou telemóvel) através do botão de opção (radio button).</p> <p>b. Introduz um ou mais contactos válidos no campo de entrada.</p> <p>c. Clica no botão “Adicionar”, acionando a função addSuppression().</p> <p>d. A função prepara o payload com os valores inseridos e envia-o à API da E-goi utilizando o método POST.</p>	
<p><b>Especificação de Saídas</b></p> <p>a. A API responde com um código de sucesso (200 OK ou 201 Created).</p> <p>b. Os contactos são adicionados à lista de supressão e refletidos na interface após recarregamento.</p> <p>c. É exibida uma mensagem de confirmação visível (“Contacto adicionado à lista com sucesso!”).</p> <p>d. Em caso de falha (ex.: dados inválidos, API key em falta, erros de rede), é apresentada uma mensagem de erro ao utilizador.</p>	
<p><b>Outros</b></p> <ul style="list-style-type: none"> <li>O campo aceita vários contactos separados por vírgulas, com sanitização automática.</li> </ul>	
<p><b>Dependências</b></p> <p>Requer validação da API Key (GOIZAPY-CT-API-014) e funcionamento da função de carregamento das supressões.</p>	

**Apêndice 8-** Especificação do Caso de Teste para adição à lista de supressão (Fonte: elaboração própria)

### Caso de Teste 9: remoção de contactos da lista de supressão

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-API-009
<b>Objetivo:</b>	Verificar se a funcionalidade de remoção de contactos da lista de supressão está a funcionar corretamente, com pesquisa prévia e chamada à API da E-goi.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. O utilizador seleciona o tipo de canal (email ou telemóvel).</li> <li>b. Introduz o valor a pesquisar (email ou número) e clica em “Procurar contacto”, acionando a função <code>procurarSuppression()</code>.</li> <li>c. Se forem encontrados resultados, o utilizador clica no botão de remoção associado, ativando a função <code>removeSuppression(id)</code>.</li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. A pesquisa retorna os contactos encontrados, visíveis em tabela.</li> <li>b. Ao confirmar a remoção, o contacto é eliminado da lista de supressão através de um pedido <code>DELETE</code> à API.</li> <li>c. É apresentada uma mensagem de sucesso (“O contacto foi removido da lista de supressão.”).</li> <li>d. A interface é atualizada com os resultados atualizados (contacto removido).</li> </ul>	
<b>Outros</b> <ul style="list-style-type: none"> <li>• NA</li> </ul>	
<b>Dependências</b> Depende da remoção prévia de supressões (GOIZAPY-CT-API-008) e da autenticação garantida via GOIZAPY-CT-API-014.	

**Apêndice 9-** Especificação do Caso de Teste para remoção de contactos da lista de supressão  
(Fonte: elaboração própria)

**Caso de Teste 10: criação de tags personalizadas**

<b>ESPECIFICAÇÃO DE CASO DE TESTE</b>	
<b>Identificador:</b>	GOIZAPY-CT-API-010
<b>Objetivo:</b>	Verificar se a criação de uma tag personalizada funciona corretamente, com envio dos dados à API da E-goi e apresentação de feedback na interface.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. O utilizador insere um nome para a nova tag no campo de texto.</li> <li>b. Selecciona uma cor predefinida ou personalizada para a tag.</li> <li>c. Ao clicar no botão “Guardar”, é chamada a função createTag().</li> <li>d. Um pedido POST é enviado para <a href="https://api.egoapp.com/tags">https://api.egoapp.com/tags</a> com os dados name e color.</li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. A API responde com código HTTP 201 (sucesso na criação) ou erro.</li> <li>b. Em caso de sucesso:           <ul style="list-style-type: none"> <li>• A lista de tags é recarregada via loadTags().</li> <li>• Os campos são limpos (nome e cor voltam ao estado inicial).</li> <li>• É exibida uma mensagem de sucesso temporária ("A sua tag foi criada").</li> </ul> </li> <li>c. Em caso de falha:           <ul style="list-style-type: none"> <li>• É exibido um alert() com a indicação de erro.</li> <li>• A consola apresenta detalhes do erro, se aplicável.</li> </ul> </li> </ul>	
<b>Outros</b> <ul style="list-style-type: none"> <li>• A cor é representada no formato hexadecimal.</li> <li>• A tag não é criada se o campo “nome” estiver vazio ou só com espaços.</li> </ul>	
<b>Dependências</b> Depende da validação da API Key (GOIZAPY-CT-API-014) e é seguido por GOIZAPY-CT-API-011 para manutenção de tags.	

**Apêndice 10-** Especificação do Caso de Teste para criação de tags personalizadas (Fonte: elaboração própria)

## Caso de Teste 11: edição e remoção de tags

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-API-011
<b>Objetivo:</b>	Validar a edição e remoção de tags existentes através da extensão, assegurando a correta comunicação com a API da E-goi.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b>	
<p>a. O utilizador seleciona uma tag existente no campo &lt;select&gt;.</p> <p>b. Os campos de nome e cor são automaticamente preenchidos com os dados da tag.</p> <p>c. O utilizador pode:</p> <ul style="list-style-type: none"> <li>• Editar os campos e clicar em “Guardar”, ativando a função saveEdit(tagId).</li> <li>• Eliminar a tag, clicando no botão com o ícone de lixo, o que ativa a função deleteTag(tagId) após confirmação.</li> </ul>	
<b>Especificação de Saídas</b>	
<p>a. A função saveEdit() envia um pedido PUT para /tags/{tag_id} com os novos valores.</p> <ul style="list-style-type: none"> <li>• Se res.ok, a lista é atualizada e é exibida uma mensagem de sucesso por 3 segundos.</li> <li>• Em caso de erro, um alert exibe a mensagem e o erro técnico é registado na consola.</li> </ul> <p>b. A função deleteTag() envia um pedido DELETE para /tags/{tag_id}.</p> <ul style="list-style-type: none"> <li>• Se res.ok, a tag é removida e a mensagem de sucesso é exibida.</li> <li>• Em caso de erro, o alert mostra os detalhes da falha retornada pela API.</li> </ul>	
<b>Outros</b>	
<ul style="list-style-type: none"> <li>• O botão de eliminar fica desativado até que uma tag seja selecionada.</li> <li>• A cor da tag é representada em formato hexadecimal (#RRGGBB).</li> <li>• A API Key é lida de chrome.storage.local.</li> </ul>	
<b>Dependências</b>	
Depende da criação prévia de tags (GOIZAPY-CT-API-010) e da autenticação via GOIZAPY-CT-API-014.	

**Apêndice 11-** Especificação do Caso de Teste para edição e remoção de tags (Fonte: elaboração própria)

**Caso de Teste 12: segmentação de contactos e ações em massa**

<b>ESPECIFICAÇÃO DE CASO DE TESTE</b>	
<b>Identificador:</b>	GOIZAPY-CT-API-012
<b>Objetivo:</b>	Validar a segmentação de contactos com filtros dinâmicos e a aplicação de ações em massa (adicionar/remover tag), com comunicação correta com a API da E-goi.
<b>Autor(es)</b>	Leonor Carneiro Machado
<p><b>Especificação de Entradas</b></p> <p>a. O utilizador seleciona uma lista de contactos.</p> <p>b. O utilizador define filtros com base em campos como: data de inscrição, estado, nome, email, telemóvel, tag ou automação.</p> <p>c. O utilizador aplica ações em massa (adicionar/remover tag) aos contactos filtrados.</p> <p>d. O sistema recupera os dados da API:</p> <ul style="list-style-type: none"> <li>• /lists</li> <li>• /tags</li> <li>• /automations</li> </ul>	
<p><b>Especificação de Saídas</b></p> <p>a. Os contactos filtrados são exibidos em uma tabela.</p> <p>b. Ao aplicar uma ação:</p> <ul style="list-style-type: none"> <li>• Para adicionar tag:           <ul style="list-style-type: none"> <li>• É enviado um POST para /lists/{list_id}/contacts/actions/attach-tag com o corpo: { contacts: [IDs], tag_id: }</li> </ul> </li> <li>• Para remover tag: • É enviado um POST para /lists/{list_id}/contacts/actions/detach-tag com estrutura idêntica.</li> </ul> <p>c. Se a ação for bem-sucedida:</p> <ul style="list-style-type: none"> <li>• Uma mensagem de sucesso é exibida ("Ação aplicada com sucesso!").</li> </ul> <p>d. Em caso de falha:</p> <ul style="list-style-type: none"> <li>• Um alert exibe o erro.</li> <li>• O erro é registado na consola.</li> </ul>	
<p><b>Outros</b></p> <ul style="list-style-type: none"> <li>• O sistema suporta vários grupos de filtros com operadores lógicos (E/OU).</li> <li>• Filtros podem ser adicionados/removidos dinamicamente.</li> <li>• Os filtros são aplicados localmente após a recuperação da lista de contactos.</li> </ul>	
<p><b>Dependências</b></p> <p>Depende de GOIZAPY-CT-API-006 (listas), GOIZAPY-CT-API-010 (tags) e autenticação via GOIZAPY-CT-API-014.</p>	

**Apêndice 12-** Especificação do Caso de Teste para segmentação de contactos e ações em massa (Fonte: elaboração própria)

### Caso de Teste 13: saudação e redirecionamento no menu

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-API-013
<b>Objetivo:</b>	Verificar se o nome do utilizador é corretamente carregado e exibido, e se o redirecionamento por botão funciona corretamente.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. A extensão é iniciada e o componente é montado (onMount).</li> <li>b. A função obterNomeUtilizador() é chamada.</li> <li>c. O utilizador interage com os botões do menu:           <ul style="list-style-type: none"> <li>• “Adicionar contacto”</li> <li>• “Procurar contacto”</li> <li>• “Segmentar contactos”</li> <li>• “Listas de contactos”</li> <li>• “Lista de supressão”</li> <li>• “Gerir tags”</li> </ul> </li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. Se a resposta da API (res.ok) for positiva:           <ul style="list-style-type: none"> <li>• O nome do primeiro utilizador (first_name) é exibido na saudação: Olá, {nome}</li> <li>• A primeira letra do nome é usada no avatar-circle.</li> </ul> </li> <li>b. Se a resposta falhar:           <ul style="list-style-type: none"> <li>• O erro é registado na consola com console.error.</li> </ul> </li> <li>c. Ao clicar num botão do menu:           <ul style="list-style-type: none"> <li>• O utilizador é redirecionado para a rota correspondente, por exemplo:               <ul style="list-style-type: none"> <li>◦ /create-contact.html</li> <li>◦ /tag.html</li> <li>◦ etc.</li> </ul> </li> </ul> </li> </ul>	
<b>Outros</b> <ul style="list-style-type: none"> <li>• O redirecionamento é feito via window.location.href.</li> <li>• A API key é lida do chrome.storage.local.</li> </ul>	
<b>Dependências</b> <p>Requer o carregamento inicial do utilizador autenticado (GOIZAPY-CT-API-014) e o correto funcionamento da navegação entre interfaces (como GOIZAPY-CT-API-004, GOIZAPY-CT-API-012).</p>	

**Apêndice 13-** Especificação do Caso de Teste para saudação e redirecionamento no menu  
(Fonte: elaboração própria)

## Caso de Teste 14: introdução e validação da API Key

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-API-014
<b>Objetivo:</b>	Validar a introdução, validação e persistência da API Key, bem como o redirecionamento para a página principal em caso de sucesso.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. A extensão é aberta e o componente é montado.</li> <li>b. É procurada automaticamente uma API Key previamente armazenada em chrome.storage.local.</li> <li>c. O utilizador insere manualmente uma API Key e confirma com o botão “Iniciar sessão”.</li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. Se for encontrada uma API Key local válida:           <ul style="list-style-type: none"> <li>• É feito um GET /ping com a API Key.</li> <li>• Se o resultado for positivo, o utilizador é redirecionado para o menu.</li> </ul> </li> <li>b. Se a API Key for inserida manualmente:           <ul style="list-style-type: none"> <li>• É feita a validação com GET /my-account.</li> <li>• Se for válida, a chave é armazenada localmente e o utilizador é redirecionado para o menu.</li> <li>• Em caso de falha, é exibido um alert com a mensagem de erro.</li> </ul> </li> </ul>	
<b>Outros</b> <ul style="list-style-type: none"> <li>• O SDK da E-goi é atualizado com a chave válida após validação.</li> </ul>	
<b>Dependências</b> <p>Este teste é a base para todos os restantes testes que requerem API Key válida, incluindo GOIZAPY-CT-API-003, GOIZAPY-CT-API-006, GOIZAPY-CT-API-010.</p>	

**Apêndice 14-** Especificação do Caso de Teste para introdução e validação da API Key (Fonte: elaboração própria)

## Caso de Teste 15: segurança e permissões

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-API-015
<b>Objetivo:</b>	Verificar se a extensão cumpre requisitos mínimos de segurança no armazenamento, permissões e comunicação com a API.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. A extensão é instalada num ambiente de teste com Chrome DevTools ativos.</li> <li>b. É aberto e analisado o ficheiro manifest.json.</li> <li>c. A API Key é introduzida manualmente no formulário de login.</li> <li>d. Funcionalidades da extensão são utilizadas normalmente.</li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. As permissões declaradas em manifest.json estão limitadas a "storage" e "activeTab", sem excessos.</li> <li>b. A API Key é guardada apenas em chrome.storage.local, sem presença em localStorage ou sessionStorage.</li> <li>c. Todas as chamadas à API são feitas via HTTPS.</li> <li>d. Nenhum script inline é detetado no DOM após carregamento das páginas.</li> <li>e. Não são utilizados métodos como eval(), Function() ou injecções JS inseguras.</li> <li>f. A extensão funciona corretamente sem erros ou bloqueios gerados pela Content Security Policy.</li> </ul>	
<b>Outros</b> <ul style="list-style-type: none"> <li>• As chamadas de rede são monitorizadas via aba Network do DevTools.</li> <li>• O conteúdo do DOM é verificado através do Inspector.</li> <li>• O armazenamento local é inspecionado na aba Application &gt; Storage.</li> </ul>	
<b>Dependências</b> Este teste complementa os testes de autenticação (GOIZAPY-CT-API-014) e é necessário para garantir segurança nas interações realizadas nos restantes testes da API.	

**Apêndice 15-** Especificação do Caso de Teste de segurança e permissões (Fonte: elaboração própria)

## Caso de Teste 16: desempenho no navegador

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-API-016
<b>Objetivo:</b>	Avaliar o impacto da extensão Goizapy no desempenho do navegador durante o uso normal.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. A extensão é instalada e ativada (Google Chrome em modo developer).</li> <li>b. As funcionalidades principais são executadas: login, criação de contacto, pesquisa, edição de tags/listas.</li> <li>c. Ferramentas como Chrome DevTools (aba Performance e Memory) estão ativas para análise.</li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. O tempo entre a alteração do DOM e a execução da ação esperada (chamada à API) é inferior a 100 ms.</li> <li>b. O consumo de memória da extensão no Chrome não excede 100 MB durante uso regular.</li> <li>c. A utilização da CPU mantém-se abaixo de 10% em estado ocioso e abaixo de 30% durante ações intensas.</li> <li>d. O tempo médio de resposta das chamadas à API (latência) é inferior a 500 ms para operações típicas (GET, POST, PATCH, DELETE).</li> <li>e. A UI da extensão mantém-se responsiva (&lt;50 ms de atraso nas interações visuais).</li> </ul>	
<b>Outros</b> <ul style="list-style-type: none"> <li>• A medição do tempo DOM, ação é feita com PerformanceObserver e logs em console.time().</li> <li>• O Chrome Task Manager é usado para monitorizar recursos em tempo real.</li> <li>• As chamadas de rede são analisadas na aba Network para medir latência e tempo total.</li> </ul>	
<b>Dependências</b> <p>Este teste está ligado ao desempenho das funcionalidades principais (GOIZAPY-CT-API-003 a GOIZAPY-CT-API-012) e à observação de impacto do content script (GOIZAPY-CT-DOM-001) no navegador.</p>	

**Apêndice 16-** Especificação do Caso de Teste de desempenho no navegador (Fonte: elaboração própria)

## Caso de Teste 17: ambiente real com WhatsApp Web

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	GOIZAPY-CT-REAL-017
<b>Objetivo:</b>	Validar o funcionamento global da extensão GoiZapy em ambiente real de utilização, simulando o comportamento típico do utilizador final.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. A extensão é instalada manualmente no Chrome via chrome://extensions.</li> <li>b. O WhatsApp Web é aberto e autenticado com um número.</li> <li>c. A extensão é ativada no contexto do WhatsApp Web com contactos reais.</li> <li>d. São realizadas ações como:           <ul style="list-style-type: none"> <li>• Criação e edição de contactos</li> <li>• Atribuição de tags</li> <li>• Interacção com listas e supressões</li> </ul> </li> <li>e. O comportamento visual é comparado com o protótipo Figma.</li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. Todos os elementos funcionais são exibidos corretamente.</li> <li>b. As chamadas à API da E-goi ocorrem como esperado, com respostas válidas.</li> <li>c. Não ocorrem erros visíveis nem falhas críticas no content script ou service worker.</li> <li>d. O layout da interface mantém consistência com o design aprovado no Figma.</li> </ul>	
<b>Outros</b> <ul style="list-style-type: none"> <li>• A DevTools do Chrome (aba Network e Console) é usada para monitorizar chamadas e erros.</li> <li>• O teste é feito em ambiente com rede real e sem cache forçada.</li> </ul>	
<b>Dependências</b> Este teste consolida os testes funcionais da extensão (GOIZAPY-CT-DOM-001 até GOIZAPY-CT-API-016), verificando a integração final e a coerência com o protótipo visual.	

**Apêndice 17-** Especificação do Caso de Teste em ambiente real com WhatsApp Web (Fonte: elaboração própria)

**Caso de Teste 18: build e empacotamento**

<b>ESPECIFICAÇÃO DE CASO DE TESTE</b>	
<b>Identificador:</b>	GOIZAPY-CT-BUILD-018
<b>Objetivo:</b>	Garantir que o processo de build da extensão GoiZapy gera corretamente os ficheiros finais necessários para execução em modo de desenvolvimento no Google Chrome.
<b>Autor(es)</b>	Leonor Carneiro Machado
<b>Especificação de Entradas</b> <ul style="list-style-type: none"> <li>a. Execução do comando de build configurado no projeto (npm run build).</li> <li>b. Verificação do conteúdo gerado no diretório /dist.</li> <li>c. Instalação manual da extensão no Chrome a partir do diretório /dist.</li> </ul>	
<b>Especificação de Saídas</b> <ul style="list-style-type: none"> <li>a. O diretório /dist é gerado corretamente, com todos os ficheiros esperados.</li> <li>b. Os seguintes bundles estão presentes e funcionais:           <ul style="list-style-type: none"> <li>• Interface de utilizador (ficheiros HTML, JS, CSS)</li> <li>• Content Script</li> <li>• Service Worker</li> </ul> </li> <li>c. A extensão é carregada em chrome://extensions, sem erros.</li> <li>d. Nenhum erro é registado na consola ao iniciar a extensão no Chrome.</li> <li>e. O manifest.json está corretamente formatado e reconhecido pelo navegador.</li> </ul>	
<b>Outros</b> <ul style="list-style-type: none"> <li>• É validado que o manifest_version, permissions e content_scripts estão corretamente declarados.</li> </ul>	
<b>Dependências</b> <p>Este teste depende da configuração correta de build (ex: vite.config.js) e garante que os scripts centrais (content script, service worker e interface) usados nos testes GOIZAPY-CT-DOM-001, GOIZAPY-CT-MSG-002 e GOIZAPY-CT-API-003 são corretamente empacotados.</p>	

**Apêndice 18-** Especificação do Caso de Teste de build e empacotamento (Fonte: elaboração própria)

## Anexos

### Proposta de Estágio/Projeto da E-goi



#### PROPOSTA DE ESTÁGIO/PROJETO EM AMBIENTE EMPRESARIAL

<b>Entidade proponente</b>	E-goi
<b>Proponente(s)</b> «indicar forma de contacto (por exemplo, e-mail, telefone, ...)»	<a href="mailto:internships@e-goi.com">internships@e-goi.com</a> <a href="mailto:dalves@e-goi.com">dalves@e-goi.com</a>
<b>Título do estágio/projeto</b>	Projeto 05 - GoiZapy - Desenvolvimento de Plugin de Integração entre E-goi e WhatsApp
<b>Objetivos</b>	A integração eficaz entre plataformas de comunicação e ferramentas de automação de marketing é essencial para as empresas que pretendem otimizar a gestão de leads e campanhas. Atualmente, não existe uma solução integrada entre o WhatsApp Web e o E-goi que permita automatizar estas interações, gerindo leads diretamente no WhatsApp e conectando-as aos fluxos de automação da E-goi. Este projeto visa desenvolver um plugin que une estas duas plataformas, potenciando as funcionalidades do WhatsApp Business Platform e da E-goi.
<b>Descrição resumida do trabalho a desenvolver</b>	<b>Informações sobre a E-goi</b> "De Matosinhos para o mundo!" Este é o lema da E-goi, plataforma de automação de marketing omnicanal. Começando pelo email, em 2002, hoje uma conta E-goi permite criar campanhas de comunicação em mais de uma dezena de canais: SMS, SmartSMS, Voz, Push, WebPush, Landing Pages, entre outros. Com mais de 700 mil utilizadores e 800 milhões de comunicações efetuadas anualmente, em todo o mundo, a E-goi é uma referência nos mercados nacional e internacional. Considerada uma das TOP 25 melhores empresas para trabalhar em Portugal em 2023. <b>Condições Oferecidas aos Estagiários</b> A E-goi valoriza o bem-estar e a motivação dos seus estagiários, proporcionando um ambiente acolhedor e dinâmico que estimula a criatividade e a aprendizagem. Durante o estágio, garantimos uma orientação próxima e uma mentorização de excelência por parte de colaboradores experientes, que acompanham todas as áreas tecnológicas e oferecem apoio dedicado na redação do relatório ou tese, contribuindo para o sucesso académico e profissional dos estagiários. Oferecemos um posto de trabalho completo e uma bolsa de estágio mensal. Além disso, os estagiários podem usufruir gratuitamente de pequeno-almoço, almoço e lanches nas nossas instalações. A nossa cultura organizacional é familiar e descontraída, promovendo um ambiente de proximidade e sinergia entre equipas. Para momentos de descompressão, disponibilizamos uma sala de jogos, reforçando o equilíbrio entre produtividade e bem-estar. Localizados numa zona privilegiada, junto à
<b>Outras considerações</b>	

**ISPGAYA**   
instituto superior politécnico  
Escola Superior de Ciência e Tecnologia  
**LICENCIATURA EM ENGENHARIA INFORMÁTICA**  
Unidade Curricular de Projeto de EI em Contexto Empresarial  
2024/2025

praia e à estação de metro, proporcionamos o cenário ideal para conciliar trabalho e qualidade de vida. Reconhecemos e valorizamos o desempenho demonstrado durante o estágio, existindo uma forte possibilidade de continuidade através de contrato. Esta abordagem reflete o nosso compromisso com a retenção de talento e com a evolução profissional, reforçando a aposta na formação e desenvolvimento de carreiras.

**Anexo 1- Proposta de projeto (Fonte: E-goi - equipa de MMA)**

## Protótipo de interface do menu



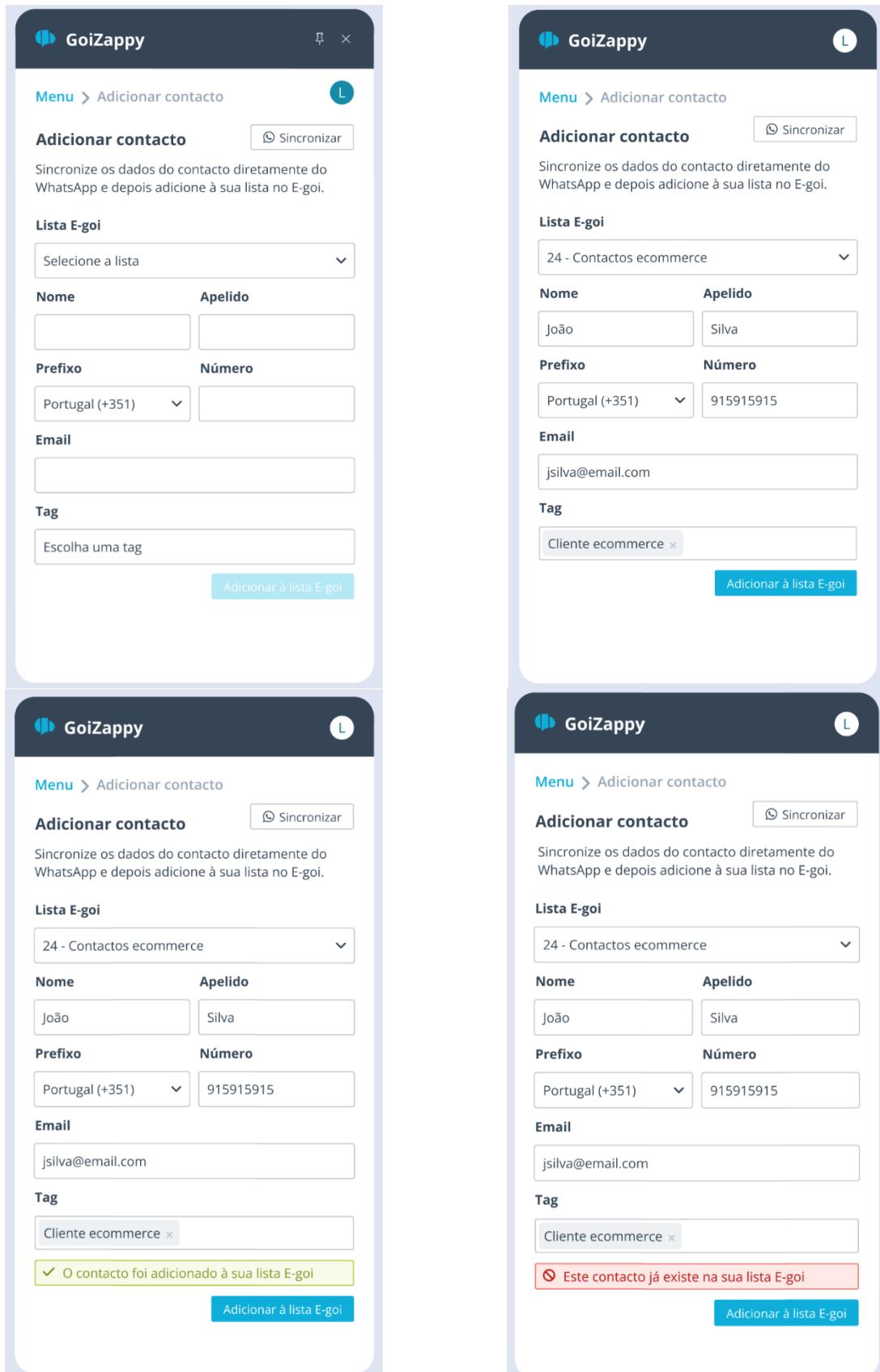
**Anexo 2-Protótipo de interface do login**  
(Fonte: prototipagem interna - equipa de MMA)

O protótipo de interface do menu da GoiZappy é dividido em três telas laterais. Todas as telas têm uma barra superior com o logo "GoiZappy" e uma barra de navegação com ícones para "L" (Logout) e "X" (Fechar).

- Tela esquerda:** Título "Olá, Leonor". Subtítulo: "Faça a gestão dos seus contactos de forma rápida e fácil, sem sair do WhatsApp." Seção "Gerir contactos": "+ Adicionar contacto", "Procurar contacto", "Segmentar contactos". Seção "Gerir listas e tags": "Listas de contactos", "Lista de supressão", "Gerir tags".
- Tela central:** Título "Olá, Leonor". Subtítulo: "Faça a gestão dos seus contactos de forma rápida e fácil, sem sair do WhatsApp." Seção "Gerir contactos": "+ Adicionar contacto", "Procurar contacto", "Segmentar contactos". Seção "Gerir listas e tags": "Listas de contactos", "Lista de supressão", "Gerir tags".
- Tela direita (menu lateral):** Botões: "Configurações", "Ajuda" (destacado em azul), "Sair".

**Anexo 3-Protótipo de interface do menu** (Fonte: prototipagem interna - equipa de MMA)

## Protótipo de interface do ecrã adicionar contacto



The image displays four wireframe prototypes of the 'Adicionar contacto' (Add Contact) screen for the GoiZappy application. Each prototype shows a different state of the form fields:

- Prototype 1 (Top Left):** Shows a blank contact entry form. Fields include: 'Lista E-goi' dropdown ('Selecionar lista'), 'Nome' and 'Apelido' inputs, 'Prefixo' and 'Número' inputs ('Portugal (+351)'), 'Email' input, 'Tag' input ('Escolha uma tag'), and a blue 'Adicionar à lista E-goi' button.
- Prototype 2 (Top Right):** Shows a contact entry form with populated fields: 'Lista E-goi' dropdown ('24 - Contactos ecommerce'), 'Nome' ('João') and 'Apelido' ('Silva') inputs, 'Prefixo' and 'Número' inputs ('Portugal (+351)', '915915915'), 'Email' ('jsilva@email.com'), 'Tag' ('Cliente ecommerce'), and a blue 'Adicionar à lista E-goi' button.
- Prototype 3 (Bottom Left):** Shows a contact entry form with populated fields: 'Lista E-goi' dropdown ('24 - Contactos ecommerce'), 'Nome' ('João') and 'Apelido' ('Silva') inputs, 'Prefixo' and 'Número' inputs ('Portugal (+351)', '915915915'), 'Email' ('jsilva@email.com'), 'Tag' ('Cliente ecommerce'), and a green success message ('✓ O contacto foi adicionado à sua lista E-goi'). The 'Adicionar à lista E-goi' button is blue.
- Prototype 4 (Bottom Right):** Shows a contact entry form with populated fields: 'Lista E-goi' dropdown ('24 - Contactos ecommerce'), 'Nome' ('João') and 'Apelido' ('Silva') inputs, 'Prefixo' and 'Número' inputs ('Portugal (+351)', '915915915'), 'Email' ('jsilva@email.com'), 'Tag' ('Cliente ecommerce'), and a red error message ('⌚ Este contacto já existe na sua lista E-goi'). The 'Adicionar à lista E-goi' button is blue.

**Anexo 4-Protótipo de interface do ecrã adicionar contacto (Fonte: prototipagem interna - equipa de MMA)**

## Protótipos de interface dos ecrãs de procura e edição de contacto

The image displays three wireframe prototypes for the GoiZappy application, showing the search and edit contact interfaces.

**Protótipo 1: Procurar contacto**

This screen shows the search interface. It includes a header with the GoiZappy logo and a user icon. Below the header, the text "Menu > Procurar contacto" is displayed. The main area is titled "Procurar contacto" and contains the instruction "Pode procurar um contacto pelo nome, apelido, email ou telemóvel." A section titled "Quero procurar pelo" contains a dropdown menu set to "Nome" and a text input field with the placeholder "Nome". A "Procurar contacto" button is located below the input field. The entire interface is enclosed in a rounded rectangle.

**Protótipo 2: Procurar contacto**

This screen shows the search results. It has a similar header and navigation bar. The main area is titled "Procurar contacto" and contains the same search instructions. Below this, a section titled "Quero procurar pelo" shows a dropdown menu set to "Nome" with the value "Leonor" selected, and a text input field with the same value. A "Procurar contacto" button is present. Below the search fields, a table displays search results:

Nome	Lista	Estado
Leonor Machado	24 - Lista ecommerce	Ativo
Leonor Santos	36 - Lista Loja 1	Ativo
Leonor Martins	43 - Lista Natal	Ativo

A total count of "Total: 3/150" is shown above the table. The entire interface is enclosed in a rounded rectangle.

**Protótipo 3: Editar contacto**

This screen shows the contact editing interface. It includes a header with the GoiZappy logo and a user icon. Below the header, the navigation path "Menu > Procurar con... > Editar contacto" is displayed. The main area is titled "Editar contacto" and contains a trash can icon. A section titled "Lista E-goi" shows a dropdown menu set to "24 - Lista Ecommerce". Below this, there are two sets of input fields: "Nome" (Leonor) and "Apelido" (Machado), and "Prefixo" (Portugal (+351)) and "Número" (915915917). Further down are fields for "Email" (lmachado@email.com) and "Tag" (Cliente ecommerce). A "Guardar alterações" button is located at the bottom right. The entire interface is enclosed in a rounded rectangle.

**Anexo 5-** Protótipos de interface dos ecrãs de procura e edição de contacto (Fonte: prototipagem interna - equipa de MMA)

## Protótipo de interface do ecrã de segmentação de contactos e aplicação de ações em massa

The image displays three wireframe prototypes of a user interface for contact segmentation and mass actions. The interface is titled "GoiZappy" and includes a "Menu" button and a "L" icon.

**Top Left Prototype:**

- Section:** Segmentar contactos
- Description:** Segmenta os seus contactos e aplique ações em massa como, por exemplo, adicionar uma tag.
- Listas de contactos:** Escolha uma lista (dropdown menu).

**Top Right Prototype:**

- Section:** Segmentar contactos
- Description:** Segmenta os seus contactos e aplique ações em massa como, por exemplo, adicionar uma tag.
- Listas de contactos:** 24 - Contactos ecommerce (dropdown menu).
- Nova pesquisa:** A search panel with fields for conditions and operators (AND/OR), and a "Adicionar grupo" button.

**Bottom Prototype:**

- Section:** Segmentar contactos
- Description:** Segmenta os seus contactos e aplique ações em massa como, por exemplo, adicionar uma tag.
- Listas de contactos:** 24 - Contactos ecommerce (dropdown menu).
- Nova pesquisa:** A search panel with fields for conditions and operators (AND/OR), and a "Adicionar grupo" button.
- Buttons:** Limpar pesquisa (Clear search) and Filtrar contactos (Filter contacts).

O protótipo apresenta duas telas de interface de usuário para a plataforma GoiZappy.

**Tela 1: Segmentar contactos**

Aqui, o usuário pode definir critérios para segmentar uma lista de 24 contatos ecommerce. Os critérios são:

- Não corresponde a uma das condições
- Quando Telemóvel não contém 915

O resultado da pesquisa mostra três contatos ativos:

Nome	Lista	Estado
Carlos Machado	24 - Lista ecommerce	Ativo
João Santos	36 - Lista Loja 1	Ativo
Paulo Martins	43 - Lista Natal	Ativo

**Tela 2: Segmentar contactos**

Aqui, o usuário pode aplicar ação em massa a 3 contatos que receberam uma tag. As opções de ação são:

- Ações do contacto
- Qual é a ação?
- Escolha uma ação

O resultado mostra que uma tag foi adicionada a 3 contatos:

Nome	Lista	Estado
Carlos Machado	24 - Lista ecommerce	Ativo
João Santos	36 - Lista Loja 1	Ativo
Paulo Martins	43 - Lista Natal	Ativo

**Anexo 6-** Protótipo de interface do ecrã de segmentação de contactos e aplicação de ações em massa (Fonte: prototipagem interna - equipa de MMA)

## Protótipos de interface dos ecrãs de gestão da lista de supressão

The image displays four wireframe prototypes of the GoiZappy mobile application interface, arranged in a 2x2 grid. Each prototype represents a different screen related to managing suppression lists.

- Top Left (Screen 1):** Shows the 'Lista de supressão' (Suppression List) screen. It includes a header with the GoiZappy logo and a back arrow. Below the header, the title 'Listas de supressão' is displayed, followed by a descriptive text: 'Use a lista de supressão para adicionar contactos que pediram para não receber nenhuma comunicação sua e mantenha sua reputação de envio em alta.' Two main buttons are shown: '+ Adicionar contactos' and 'Remover contactos'.
- Top Right (Screen 2):** Shows the 'Adicionar contactos' (Add contacts) screen. It includes a header with the GoiZappy logo and a back arrow. Below the header, the title 'Adicionar contactos' is displayed, followed by the sub-instruction 'Adicione contactos à lista de supressão utilizando o email ou telemóvel.' A section titled 'Selecionar o canal' (Select channel) shows two radio buttons: 'Email' (selected) and 'Telemóvel'. Below this is a 'Email' input field and a 'Adicionar' (Add) button.
- Bottom Left (Screen 3):** Shows the 'Remover contactos' (Remove contacts) screen. It includes a header with the GoiZappy logo and a back arrow. Below the header, the title 'Remover contactos' is displayed, followed by the sub-instruction 'Remova um contacto da lista de supressão. Faça a busca utilizando o email ou telemóvel.' A section titled 'Selecionar o canal' shows two radio buttons: 'Email' (selected) and 'Telemóvel'. Below this is an 'Email' input field and a 'Buscar contacto' (Search contact) button.
- Bottom Right (Screen 4):** Shows the 'Resultado' (Result) screen after a search. It includes a header with the GoiZappy logo and a back arrow. Below the header, the title 'Remover contactos' is displayed, followed by the sub-instruction 'Remova um contacto da lista de supressão. Faça a busca utilizando o email ou telemóvel.' A section titled 'Selecionar o canal' shows two radio buttons: 'Email' (selected) and 'Telemóvel'. Below this are sections for 'Prefixo' (Prefix) set to 'Portugal (+351)' and 'Número' (Number) set to '917917917', with a 'Buscar contacto' (Search contact) button. At the bottom, a table titled 'Resultado' shows one contact entry:

Nome	Lista	Estado
Leonor Machado	24 - Lista ecommerce	Ativo

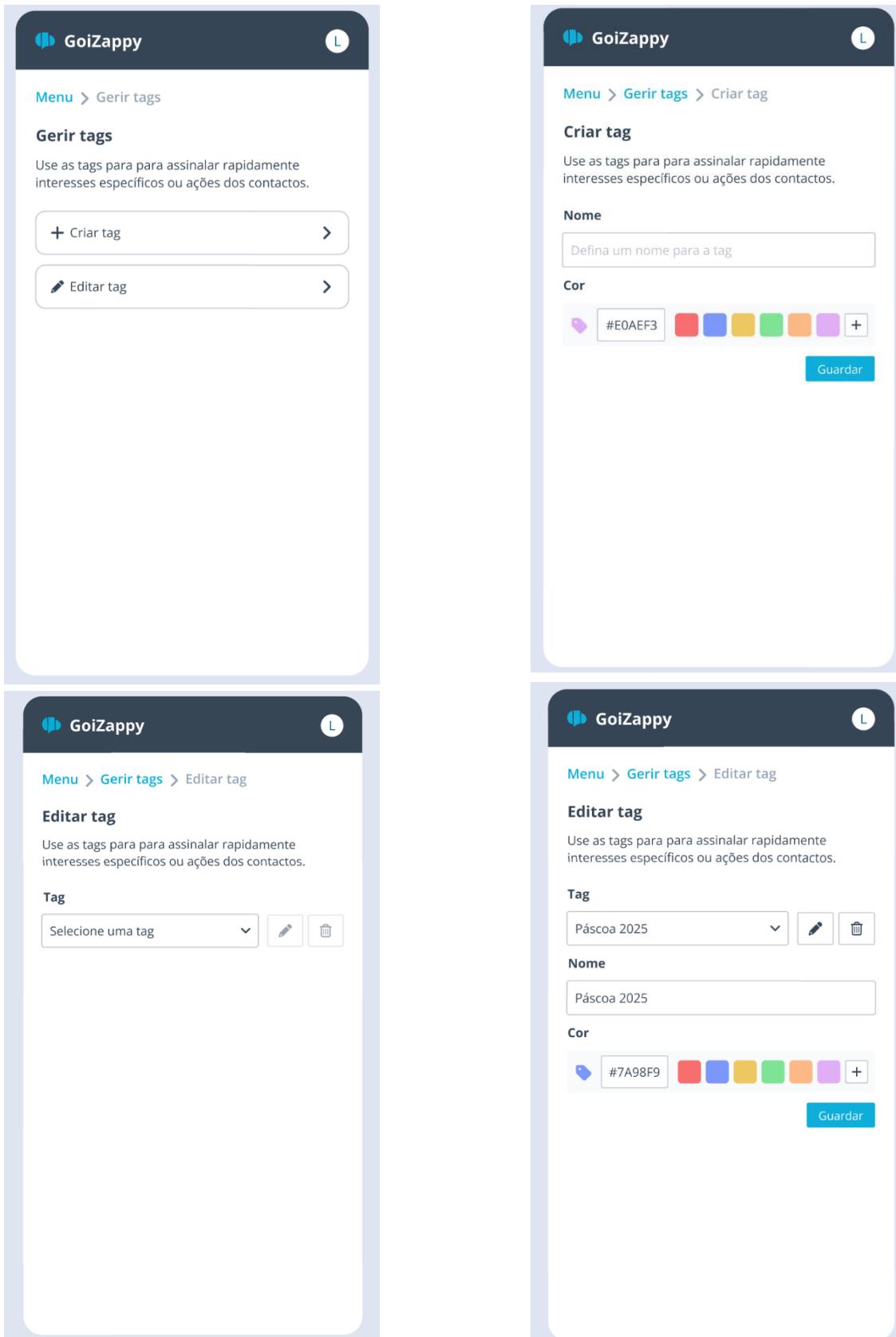
**Anexo 7- Protótipos de interface dos ecrãs de gestão da lista de supressão (Fonte: prototipagem interna - equipa de MMA)**

## Protótipos de interface dos ecrãs de gestão de listas de contactos

The image displays four wireframe prototypes of the GoiZappy application interface, arranged in a 2x2 grid. Each prototype shows a dark header bar with the 'GoiZappy' logo and a profile icon.

- Top Left (List View):** Shows the 'Lista de contactos' screen. It includes a sub-header 'Listas de contactos', a note about creating new lists or managing existing ones, and two buttons: '+ Criar lista de contactos' and 'Editar lista de contactos'.
- Top Right (Create List):** Shows the 'Criar lista de contactos' screen. It includes a sub-header 'Criar lista de contactos', a note about defining public and internal names, and fields for 'Nome público' and 'Nome interno'. A blue 'Adicionar' button is at the bottom right.
- Bottom Left (Edit List):** Shows the 'Editar lista de contactos' screen. It includes a sub-header 'Editar lista de contactos', a note about selecting a list to edit, and a dropdown menu labeled 'Escolha uma lista' with a delete icon.
- Bottom Right (Edit List Details):** Shows the 'Editar lista de contactos' screen with detailed fields. It includes a 'Lista' dropdown set to 'Lista Páscoa 2025', 'Nome público' field containing 'Lista Páscoa 2025', 'Nome interno' field containing 'Páscoa 2025', and a blue 'Guardar alterações' button at the bottom right.

**Anexo 8-** Protótipos de interface dos ecrãs de gestão de listas de contactos (Fonte: prototipagem interna - equipa de MMA)

**Protótipos de interface dos ecrãs de gestão de tags**

**Anexo 9-** Protótipos de interface dos ecrãs de gestão de tags (Fonte: prototipagem interna - equipa de MMA)