NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

GABRIEL CAMPOS

Licenciatura em Engenharia de Sistemas e Informática

RELATÓRIO DE ATIVIDADE PROFISSIONAL

DISSERTAÇÃO PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA INFORMÁTICA

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Dezembro, 2023

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTAMENTO DE
INFORMÁTICA

RELATÓRIO DE ATIVIDADE PROFISSIONAL

DISSERTAÇÃO PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA INFORMÁTICA

GABRIEL CAMPOS

Licenciatura em Engenharia de Sistemas e Informática

Orientador: Pedro Medeiros
Professor Associado, Universidade NOVA de Lisboa

Júri

Presidente: Vasco Amaral
Professor Associado, Universidade NOVA de Lisboa

Arguente: Maria Cecília Gomes
Professora Auxiliar, Universidade NOVA de Lisboa

Vogal: Pedro Medeiros
Professor Associado, Universidade NOVA de Lisboa

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Dezembro, 2023

Relatório de Atividade Profissional
Dissertação para obtenção do Grau de Mestre em Engenharia Informática

Este documento foi gerado com o processador (pdf/Xe/Lua)LaTeX e o modelo NOVAthesis (v7.1.12) [48].

Para Ilka.

Agradecimentos

Agradeço a orientação e apoio do Professor Associado Pedro Medeiros, que me ajudou na elaboração deste relatório.

Quero também agradecer a todos os colegas com quem tive o privilégio de trabalhar, que me inspiraram e fizeram crescer profissional e pessoalmente.

”

«Procurai deixar o mundo um pouco melhor do que o encontrastes.»

— Baden-Powell, Escutismo para Rapazes
(Fundador do Escutismo)

Resumo

Pretende-se com este relatório descrever o percurso profissional do candidato, com vista à obtenção do grau de Mestre em Engenharia Informática, para licenciados pré-Bolonha. Tendo terminado a Licenciatura em Engenharia de Sistemas e Informática, pela Universidade do Minho, iniciou o seu percurso profissional no ano 2000 e esteve sempre envolvido em projetos de grande complexidade e envergadura. Na MobiComp, fez parte, desde o início, da equipa que desenvolveu o projeto myTMN cujo grau de exigência possibilitou consolidar os conhecimentos obtidos na formação académica. A passagem pela Microsoft foi outro marco na carreira do candidato, não só pela completa mudança de tecnologias, mas principalmente pela mudança de processos de desenvolvimento, que influenciou significativamente as suas competências e a perceção que tinha sobre o ciclo de vida de desenvolvimento de *software*. A mudança para a Critical Software teve como objetivo obter conhecimentos e experiências em áreas distintas. Até então, a experiência obtida focava-se essencialmente na computação móvel e aqui teve a oportunidade de trabalhar na área da banca, com processos BPM, e fazer integrações complexas com um *core* bancário. Na ASAP54, como arquiteto e líder da equipa de desenvolvimento, foi possível explorar, aprofundadamente, todas os aspetos do desenvolvimento de uma aplicação móvel de pesquisa visual de produtos de moda, com uma vertente forte de rede social, desde a conceção da ideia até à concretização e contribuir de forma fundamental para o sucesso do deste projeto. Tratando-se de uma equipa pequena e inexperiente, foi responsável por todas as fases do ciclo de vida do desenvolvimento e simultaneamente pela mentoria da equipa, partilhando o conhecimento e experiência através das metodologias ágeis *Scrum* e programação em par. Atualmente, na Smith Micro Software Inc., é líder técnico e responsável pela arquitetura do SafePath sendo também mentor da comunidade de prática dedicada à área alargada do servidor aplicacional. O percurso profissional do candidato tem sido pautado por uma contínua aprendizagem e pela partilha dos conhecimentos com os colegas de trabalho.

Palavras-chave: Engenharia Informática, Arquitetura, Desenvolvimento de *Software*, *Scrum*

Abstract

This report intends to describe the candidate's professional career, with the objective of obtaining the Master's degree in Computer Science and Engineering, for pre-Bologna graduates. Having finished his degree in Systems Engineering and Informatics at the University of Minho, he began his professional career in 2000 and has always been involved in projects of great complexity and scope. At MobiComp, he was part of the team that developed the myTMN project since the beginning, whose level of demand made it possible to consolidate the knowledge obtained in the academic studies. The time at Microsoft was another milestone in the candidate's career, not only for the complete change of technologies, but mainly for the change of development processes, which significantly influenced his skills and the perception he had about the software development life cycle. The move to Critical Software had the purpose of gaining knowledge and experience in different areas. Until then, the experience gained was essentially focused on mobile computing and here he had the opportunity to work in the banking business, with BPM processes, and make complex integrations with a banking core. At ASAP54, as an architect and leader of the development team, it was possible to explore, in depth, all aspects of the development of a mobile application for visual search of fashion products, with a strong aspect of a social network, from the conception of the idea to the implementation and contribute significantly to the success of this project. Being ASAP54 a small and inexperienced team, he was responsible for all phases of the development life cycle and simultaneously for mentoring the team, sharing knowledge and experience through the Scrum and pair programming agile methodologies. Currently, at Smith Micro Software Inc., he is the technical lead and responsible for the architecture of SafePath and also mentors the community of practice dedicated to the broader area of the application server development. The candidate's professional career has been guided by continuous learning and knowledge sharing with co-workers.

Keywords: Computer Science and Engineering, Architecture, Software Development, scrum

Índice

Índice de Figuras ix

Índice de Tabelas x

Siglas xii

1 Introdução 1

2 Percurso Profissional 2
- 2.1 MobiComp (2001 – 2008) [*Software Development Engineer*] . . . 2
- 2.2 Microsoft (2008 – 2011) [*Software Development Engineer II*] . . . 3
- 2.3 Critical Software (2011 – 2013) [*Project Engineer II*] . . . 3
- 2.4 ASAP54 (2013 – 2017) [*Lead Architect and Team Leader*] . . . 4
- 2.5 Smith Micro Software Inc. (2017 –) [*Principal Software Engineer*] . . . 5
- 2.6 ESTG — P.PORTO (2021 –) [*Assistente Convidado*] . . . 6

3 Projetos de Relevo 7
- 3.1 ASAP54 . . . 7
  - 3.1.1 Desenvolvimento . . . 8
  - 3.1.2 Método de Trabalho . . . 19
- 3.2 SafePath . . . 19
  - 3.2.1 Arquitetura Inicial . . . 20
  - 3.2.2 Otimização de Desempenho . . . 22
  - 3.2.3 Colunas JSON Tipadas . . . 23
  - 3.2.4 Reestruturação da API . . . 25
  - 3.2.5 Integração de Produtos Concorrentes . . . 28
  - 3.2.6 Outras Responsabilidades . . . 31

4 Conclusão 33
- 4.1 Competências de um Mestre em Engenharia Informática . . . 34

Bibliografia **37**

Apêndices

A ASAP54 **43**
A.1 Árvore de Categorias 43

B SafePath **47**
B.1 Modelo de Dados Simplificado 47
B.2 Categorias de Domínios Internet 48
B.3 Exemplo de Utilização do JSON Patch 50

Índice de Figuras

3.1 Componentes do Servidor Aplicacional. 10
3.2 Tipos de autenticação. 12
3.3 Subcomponentes do componente *Server*. 12
3.4 Estrutura de um subcomponente. 13
3.5 Funcionamento do *Search Engine*. 13
3.6 Estrutura da entidade *Event*. 14
3.7 Interações do componente ASAP54 Tracking Service (ATS). 15
3.8 Modelo de dados simplificado do Catálogo de Produtos. 16
3.9 Componentes do Catálogo de Produtos. 17
3.10 Diagrama de classes resumido do componente *Feed Ingestion*. 18
3.11 Visão de alto nível dos componentes. 21
3.12 Transformação da entidade *Device* para usar uma coluna JavaScript Object Notation (JSON) tipada. 24
3.13 Motor de execução de ações implícitas. 26
3.14 Motor de execução de comandos. 27
3.15 Estrutura dos recursos REpresentational State Transfer (REST). 28
3.16 Componentes principais da *Circle*. 29
3.17 Componentes principais após integração da *Circle*. 30

B.1 Modelo de dados simplificado. 47

Índice de Tabelas

3.1 Requisitos principais. 8

B.1 Categorias de Domínios Internet. 48

Índice de Listagens

B.1 Documento original. 50
B.2 Operações JSON Patch. 50
B.3 Documento final. 51

Siglas

ALS ASAP54 Location Service *(pp. 10, 14)*
AMS ASAP54 Messaging Service *(pp. 10, 14)*
API Application Programming Interface *(pp. 5, 9, 11, 12, 20, 22, 25, 28, 30)*
APNs Apple Push Notification service *(p. 14)*
ATS ASAP54 Tracking Service *(pp. ix, 10, 14, 15)*
AWS Amazon Web Services *(pp. 4–6, 23, 31)*

B2B2C Business to Business to Consumer *(p. 5)*
BPMN Business Process Model and Notation *(p. 4)*

CBIR Content-Based Image Retrieval *(p. 9)*
CCG Centro de Computação Gráfica *(p. 9)*
CSV Comma Separated Values *(p. 18)*

DDL Data Definition Language *(pp. 23, 24)*
DODS Data Object Design Studio *(p. 3)*

ETL Extract, Transform, Load *(pp. 14, 25)*

FTP File Transfer Protocol *(p. 18)*

GCM Google Cloud Messaging *(p. 14)*

HTTP Hypertext Transfer Protocol *(pp. 18, 25)*
HTTPS Hypertext Transfer Protocol Secure *(pp. 11, 21)*

IBM BPM IBM Business Process Manager *(p. 4)*
IP Internet Protocol *(pp. 10, 14)*

JSON JavaScript Object Notation *(pp. ix, 4, 11, 14, 18, 20, 24–27)*

REST REpresentational State Transfer *(pp. ix, 4, 11, 20, 25, 28)*

SQL Structured Query Language *(p. 23)*

UML Unified Modeling Language *(p. 4)*

WAP Wireless Application Protocol *(p. 2)*

XML Extensible Markup Language *(pp. 2, 18)*

XMLC XML compiler *(p. 3)*

1 Introdução

Este relatório tem por objetivo a obtenção do grau de Mestre em Engenharia Informática, para licenciados pré-Bolonha, no âmbito do Despacho n.º 20/2010 da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa. Será relatado o percurso profissional, descrevendo detalhadamente dois projetos de destaque, que demonstram a aplicação dos conhecimentos adquiridos durante a Licenciatura, bem como toda a experiência obtida ao longo da carreira profissional, toda ela na área da Engenharia Informática.

O relatório está dividido em três partes. Na primeira parte será descrito, pormenorizadamente, o percurso profissional. Na segunda parte serão apresentados dois projetos que, pela sua dimensão e complexidade e pelo envolvimento do candidato nesses mesmos projetos, permitem verificar a aplicação teórica e prática das competências da Engenharia Informática. A terceira parte focar-se-á no confronto das atividades descritas nas duas partes anteriores com as competências de um Mestre em Engenharia Informática.

2

Percurso Profissional

O primeiro contacto com o mundo profissional aconteceu ainda durante a Licenciatura, no contexto do Estágio Curricular, na Infopulse Portugal [26]. Aqui trabalhou na construção de um módulo genérico de exportação e importação de informação, em Extensible Markup Language (XML), para a solução Enterprise Extender. Findo o período de estágio, ainda trabalhou por alguns meses nesta empresa até terminar o projeto sob sua responsabilidade.

2.1 MobiComp (2001 – 2008) [*Software Development Engineer*]

A MobiComp [53], uma *startup* portuguesa sediada em Braga e fundada no ano 2000, fornecia soluções móveis que geravam valor para utilizadores e operadoras através da proteção, criação, partilha e descoberta de conteúdos móveis. Conhecida por alguns produtos de destaque como o MobileKeeper, Active mPortals, Mobile Banking e Mobile Messaging Hub, para além do desenvolvimento de produtos, também fornecia serviços e soluções à medida na área da computação móvel e telecomunicações.

Foi na área dos serviços que o candidato iniciou o seu percurso na MobiComp, em fevereiro de 2001, ao integrar, desde o início, a equipa que desenvolveu o myTMN [55]. O myTMN, era um portal desenvolvido para a operadora TMN — atualmente MEO — considerado a principal interface para os clientes particulares. Entre outras funcionalidades, permitia aos seus clientes enviar mensagens SMS, gerir o seu telemóvel, subscrever serviços, como notificações de notícias, desporto, entre outros. O acesso ao portal podia ser feito usando um navegador tradicional ou usando um navegador Wireless Application Protocol (WAP) [79]. Outro dos projetos de grande destaque foi o SMS Express [66], também desenvolvido para a TMN, que era uma plataforma para envio de SMS em massa, usado principalmente para estratégias de *marketing* ou para envio de alertas em grande escala. Para além de participar ativamente em diversos outros projetos, alguns relacionados com empresas de retalho, o candidato também fez parte da equipa de manutenção e suporte de todos os serviços e produtos fornecidos pela MobiComp.

Os servidores aplicacionais, de todos os projetos, usavam uma arquitetura de três camadas e eram desenvolvidos em Java [29], com recurso ao Lutris Enhydra [49] —

mais tarde, apenas Enhydra. O Enhydra incluía alguns componentes que eram usados extensivamente, tais como o XML compiler (XMLC) [80], para desenvolver a camada de apresentação, e o Data Object Design Studio (DODS) [17] que fazia o mapeamento de objetos para bases de dados relacionais. Para persistência de dados, era usado normalmente o motor de base de dados PostgreSQL [59], mas também era suportado o motor de base de dados Oracle [56], se o cliente final assim o desejasse. Quanto à utilização de recursos, houve o cuidado, desde o início, de se obter o melhor desempenho possível de todos os serviços, através da otimização do modelo de dados — estrutura das tabelas, tipos de dados, índices, etc. —, mas também através da otimização do servidor aplicacional — utilizando ferramentas de *profiling*, *cache*, *pool* de ligações, etc.

2.2 Microsoft (2008 – 2011) [*Software Development Engineer II*]

Em 2008, com a aquisição da MobiComp pela Microsoft [51], o foco voltou-se na totalidade para o MobileKeeper, que era uma solução cujas principais funcionalidades eram:

- salvaguardar e restaurar os dados de um telemóvel, incluindo restaurar para um telemóvel que funcionava com um sistema operativo diferente;
- encontrar um telemóvel perdido;
- bloquear e limpar remotamente um telemóvel.

Após a aquisição, o MobileKeeper tornou-se o serviço My Phone [50], oferecido pela Microsoft a todos os seus clientes.

Nesta fase, o candidato foi o principal responsável por converter o produto, assente na tecnologia Java e PostgreSQL/Oracle, para tecnologias Microsoft, nomeadamente .Net Framework [1] (C# [13]) e Microsoft SQLServer [52]. Se anteriormente havia a preocupação com o desempenho e fiabilidade da solução, no contexto da Microsoft, foi necessário preparar o My Phone para conseguir lidar com milhões de utilizadores, o que obrigou a uma análise e revisão profunda do código, mas igualmente importante, foi a revisão do modelo de dados e operações de base de dados.

Antes de o My Phone poder entrar em produção, foi necessário garantir que este passava com sucesso em todos os testes de segurança, testes de qualidade de serviço e testes de integridade, em conformidade com as regras rigorosas definidas pela Microsoft.

2.3 Critical Software (2011 – 2013) [*Project Engineer II*]

Em 2011, com a deslocalização das operações do My Phone para os Estados Unidos da América, um dos principais objetivos profissionais do candidato foi trabalhar numa área diferente da computação móvel, área em que trabalhara durante dez anos. É neste contexto que ingressa na Critical Software [16], uma empresa com sede em Coimbra e fundada em 1998. A Critical Software é especializada no fornecimento de soluções, serviços e

tecnologias fiáveis para sistemas de informação críticos, fornecendo soluções de *software* e tecnologias que protegem indivíduos, monitorizam a segurança dos equipamentos e garantem que os processos críticos para o negócio são realizados de forma segura e eficiente.

Enquanto esteve na Critical Software, o candidato participou no projeto de desenvolvimento de uma solução *front office*, altamente fiável, para uma das maiores instituições bancárias angolanas. Foi responsável por definir e modelar os processos de negócio em Business Process Model and Notation (BPMN) [12], utilizando o IBM Business Process Manager (IBM BPM) [23], e respetiva integração dos processos com o *core* bancário da instituição. Um dos requisitos principais foi garantir que todas as atividades, críticas para o sistema, lidavam corretamente com situações anómalas, evitando corrupção de informação. Para facilitar a utilização dos processos modelados em BPMN por qualquer aplicação externa, o candidato desenvolveu um módulo de comunicação genérico com o IBM BPM, assente em REST [60] e JSON [33], que permite a utilização e gestão dos processos no IBM BPM. Teve também um papel de destaque na identificação e resolução de problemas com a ferramenta IBM BPM e envio de sugestões de resolução dos mesmos para a IBM [22] para serem incluídos numa versão futura do produto.

2.4 ASAP54 (2013 – 2017) [*Lead Architect and Team Leader*]

Em 2013, foi convidado a juntar-se à ASAP54 [6], um projeto ainda em fase embrionária, como arquiteto e líder de equipa. A ASAP54 foi uma aplicação pioneira e revolucionária na abordagem que utilizava para permitir a descoberta e compra de produtos de moda. Era suportada por um motor de pesquisa que combinava tecnologia de reconhecimento de imagem com pesquisa de texto.

Como arquiteto e líder de equipa, foi responsável pelo planeamento, especificação e desenvolvimento, desde o início, de todos os serviços e da infraestrutura para a plataforma. Teve um papel bastante influente como evangelista e mentor, para alcançar o melhor desempenho de todos os membros da equipa. Introduziu e fomentou o uso da metodologia de desenvolvimento *Scrum* [62] por parte da equipa de desenvolvimento.

Outras atividades e responsabilidades de relevo foram:

- análise e desenho de todo o sistema em Unified Modeling Language (UML) [11];
- instalação e configuração de toda a infraestrutura numa plataforma de computação em nuvem (Amazon Web Services (AWS) [3]), garantindo alta disponibilidade, escalabilidade, segurança e tirando o melhor partido dos recursos para manter os custos controlados;
- implementação de um serviço de pesquisa de texto de alto desempenho baseado no Apache Solr [4];

- implementação de uma ferramenta de categorização automática de produtos, provenientes de fontes variadas e descritos em diversas línguas, suportada por um classificador de aprendizagem automática floresta aleatória;
- responsável pela equipa de suporte e manutenção.

2.5 Smith Micro Software Inc. (2017 –) [*Principal Software Engineer*]

Desde 2017, trabalha na Smith Micro Software [65], mais especificamente no produto SafePath [61]. O SafePath é um produto direcionado para a família, com um modelo de negócio Business to Business to Consumer (B2B2C) baseado em parcerias com operadoras móveis, composto por quatro áreas principais.

- **SafePath Family** Serviços de localização, controlo parental e monitorização de condução.
- **SafePath Home** Controlo parental e segurança de rede para dispositivos ligados à rede doméstica através de Wi-Fi.
- **SafePath Drive** Monitorização e análise de comportamento para incentivar hábitos de condução mais seguros.
- **SafePath IoT** Gestão e monitorização centralizada de dispositivos.

O objetivo inicial consistiu em garantir que o produto conseguisse suportar um milhão de contas ativas, quando então apenas suportava algumas dezenas de milhares de utilizadores. Para isso foi necessário rever todo o produto, desde redesenhar a Application Programming Interface (API), rever o modelo de dados, repensar as regras/lógica de negócio e arquitetar a instalação da infraestrutura na plataforma de computação em nuvem AWS — usando ferramentas de automação de aprovisionamento e configuração de recursos (AWS CloudFormation [10]).

Tem também um papel muito importante como influenciador e evangelista das melhores práticas de arquitetura e desenvolvimento de *software* em todo o ecossistema SafePath (servidor, aplicações móveis e aplicação *Web*).

Outras responsabilidades e atividades são:

- redefinir a arquitetura de todo o sistema;
- pesquisar, propor e implementar melhorias ao sistema, incluindo novas formas de trabalhar, introdução de novas tecnologias e revisão dos processos de desenvolvimento;
- análise de desempenho do servidor aplicacional e da base de dados;

- desenvolvimento e manutenção de uma ferramenta interna de automação da criação da infraestrutura na plataforma AWS (Cloud Deploy);
- desenvolvimento e manutenção de uma ferramenta interna de instalação automática de todos os serviços necessários ao funcionamento do SafePath;
- análise e especificação dos requisitos de integração do SafePath com serviços internos das operadoras móveis.

2.6 ESTG — P.PORTO (2021 –) [*Assistente Convidado*]

Desde 2021, como assistente convidado na Escola Superior de Tecnologia e Gestão do Politécnico do Porto, leciona a disciplina Fundamentos de Programação Orientada a Objetos no Curso Técnico Superior Profissional em Cibersegurança, Redes e Sistemas Informáticos e a componente prática laboratorial da disciplina Engenharia de Software I da Licenciatura em Engenharia Informática.

3

Projetos de Relevo

Neste capítulo serão descritos dois projetos que se destacam pela sua complexidade, pelo impacto direto que a atuação do candidato teve e pela dimensão do ecossistema do projeto.

3.1 ASAP54

A ASAP54 foi uma aplicação pioneira e revolucionária na abordagem que utilizava para permitir a descoberta e compra de produtos de moda. Era suportada por um motor de pesquisa que combinava tecnologia de reconhecimento de imagem com pesquisa de texto. A ideia surgiu da frustração sentida pela fundadora quando palavras não eram suficientes para descrever aquilo que procurava. Numa tentativa de explicar de forma simples o que se pretendia da aplicação, esta era também referida como *O Shazam da moda* [7, 8, 73], porque com uma fotografia era possível encontrar o produto em causa, ou produtos semelhantes, usando marcadores como cor, padrão, textura, categoria[1], entre outros.

Dependendo das condições da fotografia[2], os resultados podiam variar e para contornar essas limitações a pesquisa poderia ser refinada através de texto. A possibilidade de se poder usar texto para refinar os resultados, abriu outras possibilidades não consideradas inicialmente. Por exemplo, partindo de uma fotografia não necessariamente relacionada com produtos de moda, uma paisagem ou um quadro, era possível refinar por categoria e encontrar produtos com as mesmas cores, padrões e textura.

O modelo de negócio assentava principalmente na cobrança de uma comissão sobre as vendas de produtos do catálogo. O catálogo era constituído por alguns milhões de produtos, indexados através de parcerias com fornecedores externos[3], ver Catálogo de Produtos. Se os resultados da pesquisa não fossem satisfatórios, o utilizador tinha a possibilidade de pedir ajuda a uma equipa de estilistas especializadas que tentava encontrar o produto ou outros produtos semelhantes, baseando-se nos critérios de pesquisa definidos pelo utilizador. Para cativar e estimular o uso da aplicação, os utilizadores tinham a possibilidade

[1] Exemplos de categorias: sapatos, calças, relógios, etc.
[2] Luminosidade, perspetiva, foco.
[3] A ASAP54 tinha parcerias com dezenas de fornecedores, como a Farfetch, a Asos, a Zara, entre outros.

de partilharem as pesquisas, com resultados mais relevantes, diretamente na aplicação e também noutras redes sociais.

Neste projeto o candidato desempenhou as funções de arquiteto e líder de equipa. Como arquiteto, foi responsável pelo planeamento, especificação e desenvolvimento, desde o início, de todos os serviços e da infraestrutura para a plataforma, principalmente nas áreas servidor aplicacional e catálogo de produtos.

Como líder de equipa, foi responsável por organizar e mentorar a equipa de desenvolvimento, ainda muito inexperiente, apoiando-se em algumas metodologias ágeis, e definiu também os processos e diretrizes, que a equipa deveria seguir, sobre como estruturar e escrever código. Estes processos e diretrizes fomentaram a partilha de conhecimento, a aplicação das melhores práticas de desenvolvimento, a uniformização do código produzido e promoveram a manutenção do projeto a longo prazo.

3.1.1 Desenvolvimento

O facto de ter estado envolvido desde o início do projeto[4], permitiu ao candidato pôr em prática todos os conhecimentos obtidos até então, nomeadamente na análise do problema, levantamento dos requisitos, especificação, desenho da arquitetura, planeamento e implementação. A Tabela 3.1 lista os requisitos principais identificados inicialmente.

Tabela 3.1: Requisitos principais.

Categoria	Requisito
Autenticação	Autenticação com conta de redes sociais (Facebook e Twiter).
	Autenticação com conta local.
Catálogo de Produtos	Criar um catálogo de produtos usando como fonte os catálogos de fornecedores parceiros.
	Atualizar o catálogo periodicamente.
	Normalizar as diversas categorias dos fornecedores parceiros numa árvore de categorias comum.
	Categorização automática dos produtos.
	Categorização manual dos produtos para os quais a categorização automática não funcionou.
Pesquisa	Pesquisa por imagem com possibilidade de selecionar região de interesse.
	Refinar a pesquisa através de texto com preenchimento automático (*autocomplete*).
	Refinar a pesquisa usando um dos resultados como ponto de entrada para uma nova pesquisa.

[4]Foi inclusivamente o primeiro funcionário da ASAP54.

	Pedir a ajuda de estilistas especializadas para pesquisas cujos resultados não sejam satisfatórios.
	Comprar um produto resultante de uma pesquisa, garantindo a atribuição da compra à ASAP54.
Social	Publicar pesquisas e respetivos resultados na área pessoal do utilizador.
	Comentar e responder a comentários em pesquisas.
	Gostar e reagir a publicações.
	Partilhar pesquisas e respetivos resultados nas redes sociais, garantindo que as atribuições de potenciais compras são mantidas.
	Seguir outros utilizadores da aplicação.
Reconhecimento de Imagem	Criar um modelo que permita identificar cores, formas e texturas.
	Extrair marcadores de uma imagem que serão usados para indexar as imagens do catálogo e para pesquisar imagens nesse mesmo catálogo.

Após a identificação dos requisitos, o projeto foi decomposto em quatro áreas principais: servidor aplicacional, catálogo de produtos, motor de pesquisa visual — Content-Based Image Retrieval (CBIR) — e aplicações cliente — Android, iOS e *Web*. Inicialmente, enquanto a equipa não estava completamente formada, a análise e desenvolvimento do motor de pesquisa visual foram feitos em parceria com o Centro de Computação Gráfica (CCG) [9] e o desenvolvimento da aplicação cliente iOS foi feito em parceria com a iMobileMagic [25]. Este relatório focar-se-á essencialmente nas áreas servidor aplicacional e catálogo de produtos, incluindo a integração com o motor de pesquisa visual. Apesar de o candidato ter estado envolvido em todas as áreas, estas foram aquelas em que esteve diretamente envolvido e para as quais mais contribuiu.

3.1.1.1 Servidor Aplicacional

O termo "Servidor Aplicacional" pode ter vários significados, aproximadamente relacionados, mas aqui representa aquilo que numa arquitetura cliente-servidor se entende pelo servidor, com todos os seus diversos serviços. Esta área estava dividida em vários componentes, cada um responsável por uma atividade específica.

- ***Server*** — Componente principal e onde estavam definidas as API utilizadas pelas aplicações cliente. Foi o primeiro componente a ser criado, do qual parte dos restantes componentes foram sendo extraídos, à medida que novas funcionalidades eram acrescentadas, para evitar uma potencial arquitetura monolítica. O próprio nome

do componente é reflexo dessa implementação inicial, quando apenas existia um serviço que fornecia todas as funcionalidades.

- ***Search Engine*** — Componente que fornecia a funcionalidade de pesquisa, utilizando para isso o Catálogo de Produtos.

- **ASAP54 Location Service (ALS)** — Componente que permitia obter a localização geográfica de um endereço Internet Protocol (IP). Este componente era utilizado principalmente para filtrar dos resultados os produtos que não estavam disponíveis no país do utilizador e também para apresentar os preços na moeda local.

- **ASAP54 Messaging Service (AMS)** — Componente utilizado para o envio de notificações, quer notificações individuais para um determinado utilizador, quer notificações em massa, no contexto de campanhas.

- **ATS** — Componente utilizado para recolha de dados de utilização e posterior envio para uma plataforma externa de análise de dados (Tableau [68]).

A Figura 3.1 apresenta uma visão alto nível sobre como estes componentes interagem entre si. Como referido acima, esta não foi a estrutura inicial, mas ao fim de algumas iterações, e à medida que iam sendo acrescentadas novas funcionalidades, tornou-se necessário dividir o sistema em unidades menores.

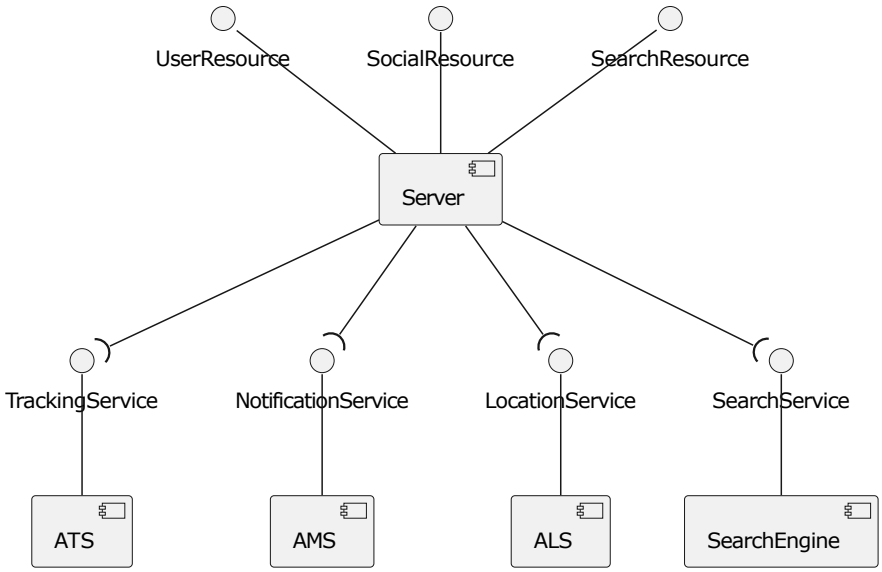


Figura 3.1: Componentes do Servidor Aplicacional.

Todos os componentes foram desenvolvidos como aplicações Java EE [30] — atualmente Jakarta EE [28] —, seguindo processos e estruturas semelhantes, que eram depois implantadas no Glassfish [20], a implementação de referência da plataforma Java EE. Das diversas especificações incluídas na plataforma Java EE, estas são as que foram utilizadas extensivamente neste projeto:

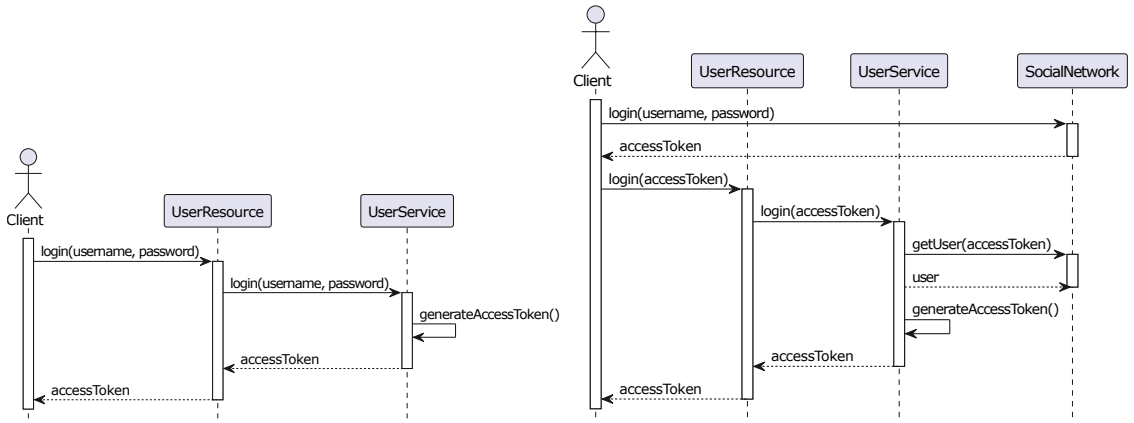
- *Streaming API for XML (StAX)* [36]
- *Java API for XML Processing (JAXP)* [37]
- *Java Architecture for XML Binding (JAXB)* [38]
- *Java API for XML-Based Web Services (JAX-WS)* [39]
- *Java Persistence (JPA)* [40]
- *Java API for RESTful Web Services (JAX-RS)* [41]
- *Enterprise JavaBeans (EJB)* [42]
- *Contexts and Dependency Injection for Java (CDI)* [43]
- *Java API for JSON Processing (JSON-P)* [44]
- *Java API for WebSocket* [45]
- *Java Transaction API (JTA)* [46]

Para persistência dos dados, foram explorados dois modelos, o modelo relacional (MySQL [54]) e o modelo de grafos de propriedades (OrientDB [57]). Para ajudar na decisão, foram feitos protótipos com ambos os modelos com resultados melhores para o modelo de grafos. Não obstante esses resultados, e apesar de a ASAP54 ter uma componente forte de rede social — área em que o modelo de grafos seria mais adequado —, a escolha recaiu sobre o modelo relacional. O que pesou mais nesta decisão foi o prazo para a apresentação pública da ASAP54 e também porque o modelo de dados relacional, tal como estava estruturado, permitia suportar alguns milhões de utilizadores registados, dependendo do tipo de perfil de utilização. O modelo de grafos não foi completamente descartado, tendo ficado como uma possível solução, caso a necessidade o justificasse.

Server

Estas são as principais características deste componente, identificadas desde as fases iniciais, e apesar de, durante os vários ciclos de desenvolvimento, o componente ter sofrido algumas reestruturações, a sua essência não mudou.

- **Comunicação** — A comunicação entre cliente e servidor usa REST com mensagens JSON. Para garantir tolerância à falha, o servidor não guarda informação de estado da comunicação (comunicação *Stateless*).
- **Segurança** — As API públicas estão expostas apenas por Hypertext Transfer Protocol Secure (HTTPS).



(a) Credenciais locais. (b) Credenciais de rede social.

Figura 3.2: Tipos de autenticação.

- **Autenticação e Autorização** — O acesso às API está restrito a clientes devidamente autenticados e autorizados, sendo que a autenticação pode ser feita através de uma conta local ou através de uma conta de uma rede social suportada (Facebook ou Twitter), ver Figura 3.2 — o uso de uma conta local carece de registo prévio. A implementação da autorização é baseada no RFC 6750 [71], prescindindo-se do *Resource Owner* e enviando-se as credenciais diretamente ao *Authorization Server*.

Figura 3.3: Subcomponentes do componente *Server*.

- **Estrutura** — Toda a lógica de negócio está implementada neste componente, estando contudo dependente das funcionalidades fornecidas pelos restantes componentes — é este componente quem faz a orquestração de todos os outros, como pode ser visto na Figura 3.1. Para melhor organização e para permitir uma manutenção evolutiva, o componente foi ele próprio dividido em diversos subcomponentes, pequenos em âmbito e responsáveis apenas por uma funcionalidade. A Figura 3.3 evidencia a

maioria desses subcomponentes — devido à quantidade de elementos, não foram desenhadas as ligações entre eles. Cada subcomponente segue uma estrutura comum, ver Figura 3.4, contendo uma entidade, com o mesmo nome que o subcomponente, e uma interface de negócio com a respetiva implementação. Em alguns deles, foram acrescentadas entidades e interfaces auxiliares internas, para manter a simplicidade dos elementos principais. Os restantes componentes do "Servidor Aplicacional" seguem uma estrutura semelhante.

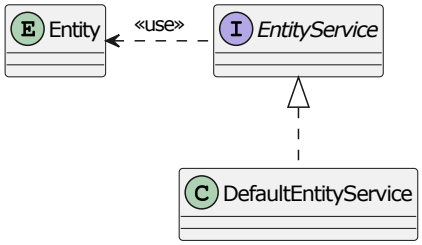


Figura 3.4: Estrutura de um subcomponente.

Search Engine

Este componente era responsável por receber os pedidos de pesquisa feitos pelas aplicações cliente e adaptá-los para poderem ser processados pelo Catálogo de Produtos. Nomeadamente, obtinha os dados da imagem e, usando o *Image Handler*, extraía os marcadores necessários para fazer a pesquisa. Adicionalmente, como era possível paginar os resultados de uma pesquisa, ou refinar a pesquisa com outros atributos opcionais, como cor, textura, padrão, categoria, marca etc., era também responsabilidade deste componente fazer a devida adaptação destes atributos em atributos reconhecíveis pelo Catálogo de Produtos, ver Figura 3.5.

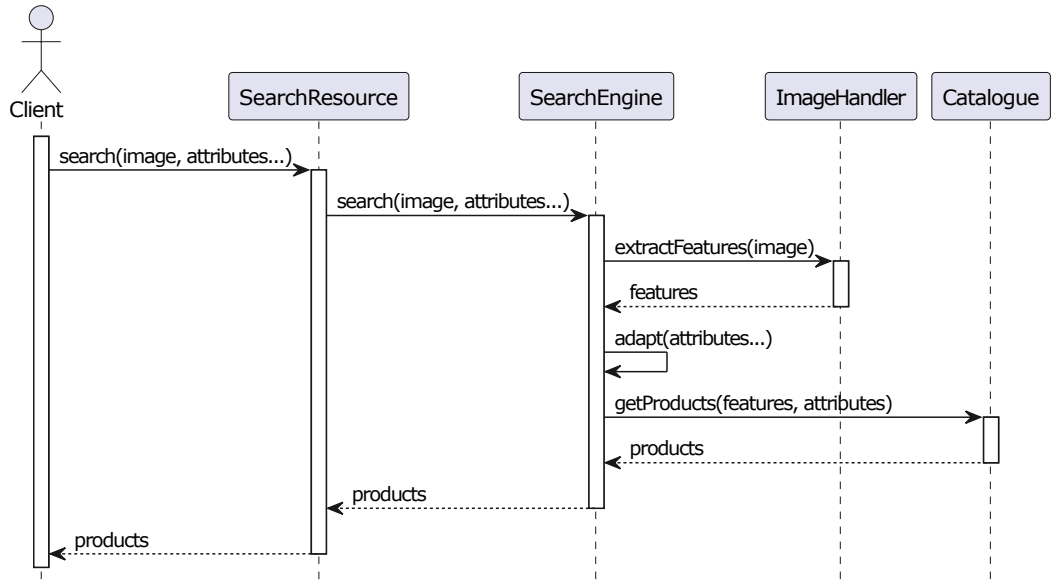


Figura 3.5: Funcionamento do *Search Engine*.

ALS

Este componente expunha uma API que, dado um endereço IP, devolvia o código do país associado a esse endereço. Por uma questão de eficiência, tinha uma base de dados local de endereços IP, atualizada periodicamente com dados fornecidos pelo serviço IP2Location [24]. Tendo em conta os requisitos, era suficiente obter o país, pelo que não era necessário guardar informação mais precisa, mantendo assim o tamanho da base de dados sob controlo.

AMS

Este componente fazia a ponte entre o componente *Server* e as plataformas de envio de notificações Apple Push Notification service (APNs) [5] e Google Cloud Messaging (GCM) [21]. Como funcionalidades, permitia o envio de notificações individuais para um determinado utilizador e o envio de notificações em massa, no contexto de campanhas.

ATS

Este componente era utilizado para recolha e persistência de todos os dados de utilização gerados pelas aplicações cliente e pelo componente *Server*. Os dados tinham uma estrutura genérica, com a informação comum persistida em campos específicos e a informação variável persistida num único campo do tipo JSON, ver Figura 3.6. Estes dados podiam ser

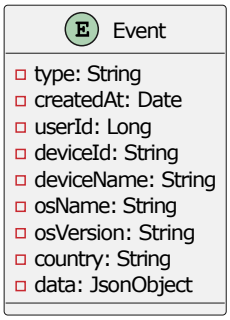


Figura 3.6: Estrutura da entidade *Event*.

analisados localmente, de forma básica, através de um portal *Web* construído para o efeito, mas o objetivo principal era o envio posterior para uma plataforma externa de análise de dados (Tableau [68]). Para envio dos dados para a plataforma externa, foi implementado um processo Extract, Transform, Load (ETL) [19], executado periodicamente, que enviava os todos os novos registos desde a sua última execução. A Figura 3.7 descreve as diversas interações do componente ATS.

3.1.1.2 Catálogo de Produtos

O catálogo foi pensado como uma entidade dinâmica, construído com os catálogos de fornecedores parceiros, com atualizações periódicas e automáticas. Para simplificar o processo de adicionar novos parceiros, privilegiou-se a utilização de programas de

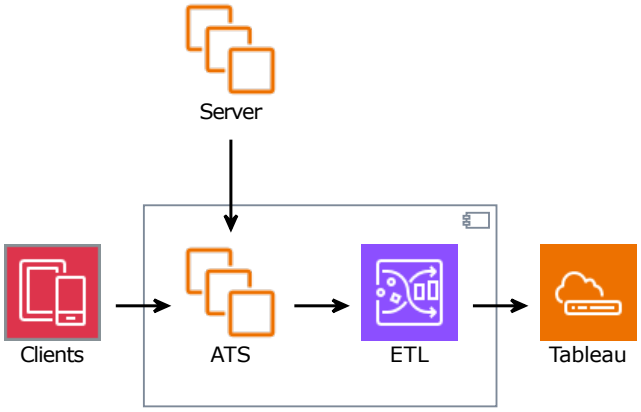


Figura 3.7: Interações do componente ATS.

afiliados [2]. Os programas de afiliados permitiam o acesso direto a centenas de parceiros, sem a necessidade de negociar individualmente com cada um deles — a ASAP54 tinha integração com quinze programas de afiliados diferentes. Durante a fase de análise, o catálogo chegou ter mais de seiscentos parceiros e mais de três milhões de produtos, mas como alguns não se adequavam à proposta da ASAP54, o catálogo foi revisto e vários parceiros removidos, o que eliminou também algum ruído nos resultados das pesquisas[5]. Após a revisão, o catálogo ficou com cerca de quatrocentos parceiros e cerca de um milhão de produtos.

Todos os produtos deviam ser categorizados consoante o seu tipo e género, contudo, cada programa de afiliados tinha a sua própria categorização, muitas vezes com nomes ou estruturas diferentes, o que dificultava grandemente a utilização do catálogo. Para além das diferenças entre programas de afiliados, havia também diferenças entre os parceiros do mesmo programa de afiliados. Para resolver este problema, foi necessário criar uma árvore de categorias normalizada que servisse as necessidades da ASAP54, ver resultado na secção Árvore de Categorias do Apêndice A. Foram também definidos os géneros que deveriam ser suportados: *male*, *female*, *unisex* e *unknown*.

Tendo em conta os requisitos identificados na Tabela 3.1, nomeadamente os respeitantes ao Catálogo de Produtos e à Pesquisa, foi escolhido o Apache Solr [4], doravante apenas Solr, para persistir todo o catálogo e fornecer as funcionalidades de pesquisa. Numa primeira abordagem, os produtos seriam persistidos numa base de dados relacional e a informação necessária para a pesquisa, texto e marcadores da imagem, seriam duplicados no Solr. Esta abordagem foi descartada, logo no início do desenvolvimento, e toda a informação dos produtos ficava persistida apenas no Solr, porque, para além aumentar o espaço necessário para o catálogo, implicava também um esforço acrescido para a manutenção do mesmo. A Figura 3.8 representa o modelo de dados de um produto do catálogo, aqui chamado *Document* — terminologia adotada pelo Solr —, de um programa de afiliados,

[5]Alguns destes parceiros tinham imagens de má qualidade e com muito ruído visual, o que introduzia erros na classificação das imagens.

aqui chamado *Provider*, e de um parceiro, aqui chamado *Merchant*. As entidades *Provider* e *Merchant* eram persistidas numa base de dados relacional.

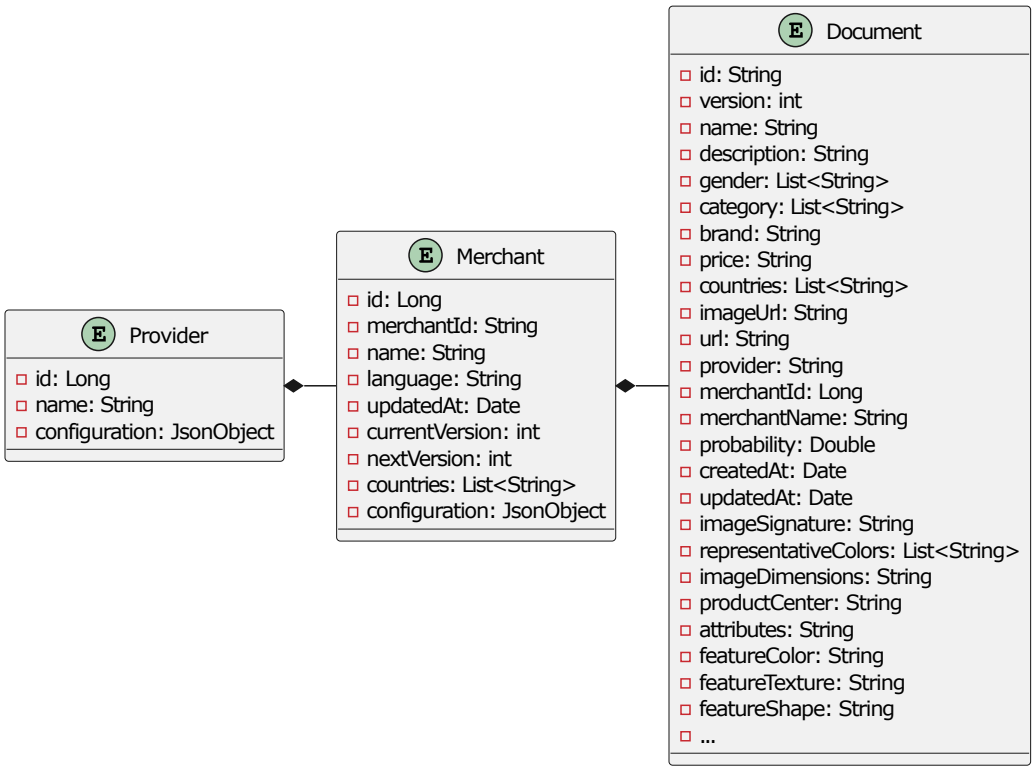


Figura 3.8: Modelo de dados simplificado do Catálogo de Produtos.

À semelhança da área Servidor Aplicacional, também esta área estava dividida em vários componentes, cada um responsável por uma atividade específica.

- ***Category Classifier*** — Componente que categorizava os produtos usando um classificador de aprendizagem automática floresta aleatória.
- ***Image Handler*** — Componente que extraía as propriedades das imagens como marcadores em formato de texto que poderiam ser usados tanto para indexar como procurar produtos no Catálogo de Produtos.
- ***Feed Ingestion*** — Componente responsável por atualizar o Catálogo de Produtos com os catálogos dos fornecedores parceiros.
- ***Catalogue*** — Componente que abstraía o acesso ao Solr e fornecia as funcionalidades de gestão dos produtos e de pesquisa.

A Figura 3.9 apresenta uma visão alto nível sobre como estes componentes interagem entre si.

Category Classifier

Para minimizar o esforço de manter o catálogo corretamente categorizado, foi desenvolvido um classificador de aprendizagem automática floresta aleatória, usando os algoritmos

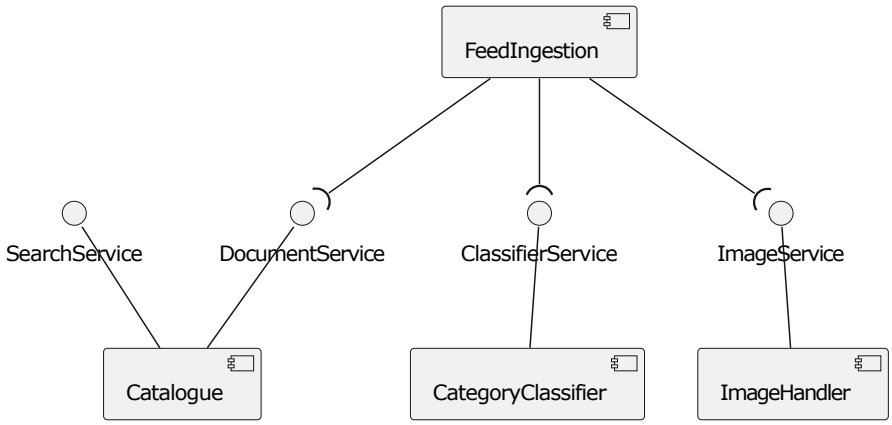


Figura 3.9: Componentes do Catálogo de Produtos.

disponibilizados pela biblioteca Weka [74]. Para treinar o modelo inicial, começou-se com o catálogo de um parceiro que se sabia estar bem categorizado, com informação de boa qualidade (nome, descrição, categoria original, etc.). Seguiu-se um processo iterativo de testar o modelo com catálogos de outros parceiros com informação de menor qualidade, ajustes aos parâmetros e novo treino. Para remover ruído e melhorar a precisão da classificação, acrescentaram-se várias palavras à lista de *stop words* convencional e criou-se um filtro para remover palavras que não tinham valor semântico, por exemplo, palavras com algarismos. A precisão da classificação era persistida com o produto, para facilitar verificações manuais dos resultados e mover produtos erradamente categorizados para a categoria correta — estas categorizações manuais serviam depois para melhorar o modelo, quando fosse executada uma nova iteração de treino. Este processo foi repetido para cada uma das línguas suportadas pelo Catálogo de Produtos: alemão, francês, inglês e português.

Após o modelo estar estabilizado, constatou-se que, com as primeiras mudanças de estação do ano, a precisão da classificação baixou. Após análise, concluiu-se que a diminuição da classificação estava relacionada com as alterações que os fornecedores parceiros faziam às descrições dos produtos na mudança de estação.

Image Handler

O *Image Handler* permitia extrair as propriedades das imagens como marcadores em formato de texto. No caso das imagens dos produtos do catálogo, as suas propriedades eram persistidas com os produtos para serem indexadas com os restantes atributos e no caso das imagens usadas nas pesquisas, as suas propriedades eram acrescentadas aos outros critérios de pesquisa.

Feed Ingestion

O *Feed Ingestion* era um processo periódico que atualizava o Catálogo de Produtos com os catálogos dos fornecedores parceiros — estes catálogos eram denominados *feed* pelos programas de afiliados, daí o nome do componente. Cada programa de afiliados definia

uma estrutura própria para os catálogos que disponibilizavam, comum a todos os seus parceiros, em formato JSON, XML ou Comma Separated Values (CSV), e normalmente disponibilizados por Hypertext Transfer Protocol (HTTP) ou File Transfer Protocol (FTP). Para além destas diferenças, havia também diferenças noutras propriedades que eram obrigatórias para o catálogo da ASAP54 e, em alguns programas afiliados, propriedades em falta. Para lidar com todas as variantes, foi criado um módulo de adaptadores que permitia acomodar as especificidades de cada programa de afiliados, ver Figura 3.10. Após a estrutura base ter sido criada, garantindo que todo o comportamento comum não era repetido, mas sim reutilizado, foi possível simplificar e diminuir o tempo necessário para adicionar novos programas de afiliados.

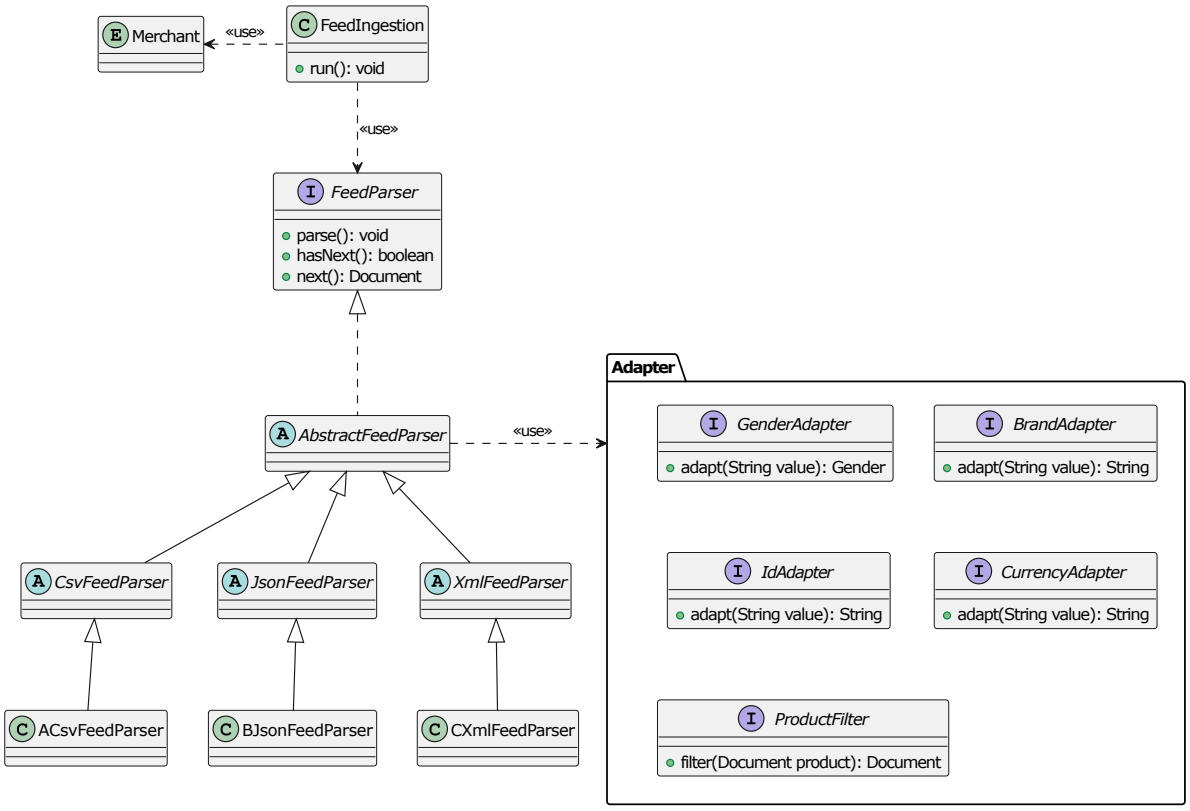


Figura 3.10: Diagrama de classes resumido do componente *Feed Ingestion*.

Catalogue

Este componente tinha a responsabilidade de abstrair os acessos ao Solr, nomeadamente as operações de gestão de documentos: criar, ler, atualizar e apagar, e também as operações de pesquisa com suporte para navegação facetada e preenchimento automático. Tendo em conta as especificidades da ASAP54, foi necessário estender o Solr, para adequar o cálculo da similaridade e da pontuação de um documento em relação aos critérios de pesquisa. Igualmente importante foi o cálculo dos pesos a dar a cada um dos critérios, para que parâmetros pouco relevantes influenciassem negativamente os resultados de pesquisa.

3.1.2 Método de Trabalho

Quando a ASAP54 iniciou atividade, a equipa era ainda muito pequena e com pouca experiência — alguns elementos ainda recém-formados —, mas o candidato fomentou desde o início um espírito colaborativo de trabalho. A estratégia do candidato passou por aplicar algumas metodologias ágeis — *Scrum* [62] e programação em par [58] —, e envolver a equipa, o máximo possível, em todas as fases do ciclo de vida do desenvolvimento, desde a análise à manutenção [77]. Mesmo com uma equipa pequena, o candidato definiu um conjunto de diretrizes sobre como estruturar e escrever código, para promover uma uniformização do código produzido e com isso aumentar a facilidade de manutenção a longo prazo, mesmo por pessoas que não tenham estado envolvidas no seu desenvolvimento. Igualmente importante foram as revisões de código, porque permitiram a partilha de conhecimento, a aplicação das melhores práticas na implementação e também eram uma garantia de que as diretrizes eram seguidas.

3.2 SafePath

O SafePath é um produto da Smith Micro Software Inc. que permite a operadoras móveis fornecerem soluções completas de segurança familiar. Dada a sua extensão, este relatório focar-se-á apenas no SafePath Family, uma das quatro áreas do produto. Esta área tem como funcionalidades principais: os serviços de localização, o controlo parental e a monitorização de condução. Quando o candidato se juntou à equipa do SafePath, o produto já tinha alguns anos de existência e estava instalado em várias operadoras móveis, de pequeno e médio porte, e havia perspetiva de o instalar também em operadoras de grande porte. Nesta categoria de operadoras, o número potencial de utilizadores passa das dezenas ou poucas centenas de milhares para os milhões de utilizadores.

Como *Principal Software Engineer*, o candidato foi responsável por supervisionar os aspetos técnicos do SafePath, nomeadamente:

- redefinir a arquitetura de todo o sistema e garantir a consistência entre todos os seus componentes;
- pesquisar, propor e implementar melhorias ao sistema, incluindo novas formas de trabalhar, introdução de novas tecnologias e revisão dos processos de desenvolvimento;
- garantir a manutenção e melhorar o desempenho do sistema;
- orientar e contribuir para o desenvolvimento técnico dos elementos da equipa;
- analisar e especificar os requisitos de integração do SafePath com serviços internos das operadoras móveis.

A lista abaixo descreve algumas das contribuições do candidato para este projeto que serão descritas nas secções seguintes.

- Otimização do desempenho do sistema para suportar pelo menos um milhão de contas ativas, quando então apenas suportava algumas dezenas de milhares.

- Introdução do conceito de colunas JSON tipadas cuja implementação permitiu diminuir, significativamente, as alterações necessárias ao esquema de base de dados durante o desenvolvimento de novas funcionalidades.

- Reestruturação da API com a introdução do método `PATCH` que permitiu remover a maioria dos métodos expostos por cada recurso REST e diminuir a quantidade de código necessária, na implementação de novas funcionalidades, tanto no servidor aplicacional como nas aplicações cliente.

- Definição da nova arquitetura e da estratégia de integração de dois produtos, concorrentes do SafePath, adquiridos pela empresa.

3.2.1 Arquitetura Inicial

Devido à experiência obtida noutros projetos, o objetivo principal foi garantir que o SafePath suportava pelo menos um milhão de utilizadores e, se possível, sem alterar a sua arquitetura. Antes de iniciar este processo, foi necessário conhecer a arquitetura e o modo de funcionamento de então. Tratava-se de uma arquitetura em tudo semelhante à utilizada no projeto ASAP54. A principal diferença é que o produto SafePath é considerado uma *Framework* [67], ou seja, não é instalado diretamente nos clientes finais, mas serve de base a todas as implementações[6], com pontos de extensão que permitem facilmente interligar com os sistemas das operadoras móveis como: criação de conta; autenticação e autorização; faturação e cobrança; etc. A Figura 3.11 apresenta uma visão de alto nível de alguns dos componentes. Destes, destacam-se os componentes *Common*, *External* e *ExternalAPI*, que são a peça fundamental para a classificação do SafePath como uma *Framework*. O componente *Common*, responsável por toda a lógica de negócio, usa várias estratégias para permitir que o SafePath possa ser adaptado às necessidades específicas de cada operadora móvel, nomeadamente inversão de controlo, através da injeção de dependência [27], método de modelo [72] e extensibilidade [18]. É nos componentes *External* e *ExternalAPI* que são concretizadas todas as funcionalidades necessárias para uma implementação, usando as estratégias definidas anteriormente. O componente *API* expõe uma interface RESTful usada pelas aplicações cliente e não pode ser alterada pelas implementações — salvo algumas extensões ao modelo de dados, em propriedades genéricas do tipo JSON. Para persistência dos dados, era usado o modelo relacional (MySQL [54]), para a maioria das funcionalidades, e o modelo não relacional chave-valor (Redis), para uma base de dados de categorização de domínios Internet[7], ver Categorias de Domínios Internet.

[6] Considera-se uma implementação, um projeto feito à medida para uma operadora móvel, que usa o SafePath como base.

[7] Estes dados eram usados pelo controlo parental para configurar e filtrar o acesso à Internet.

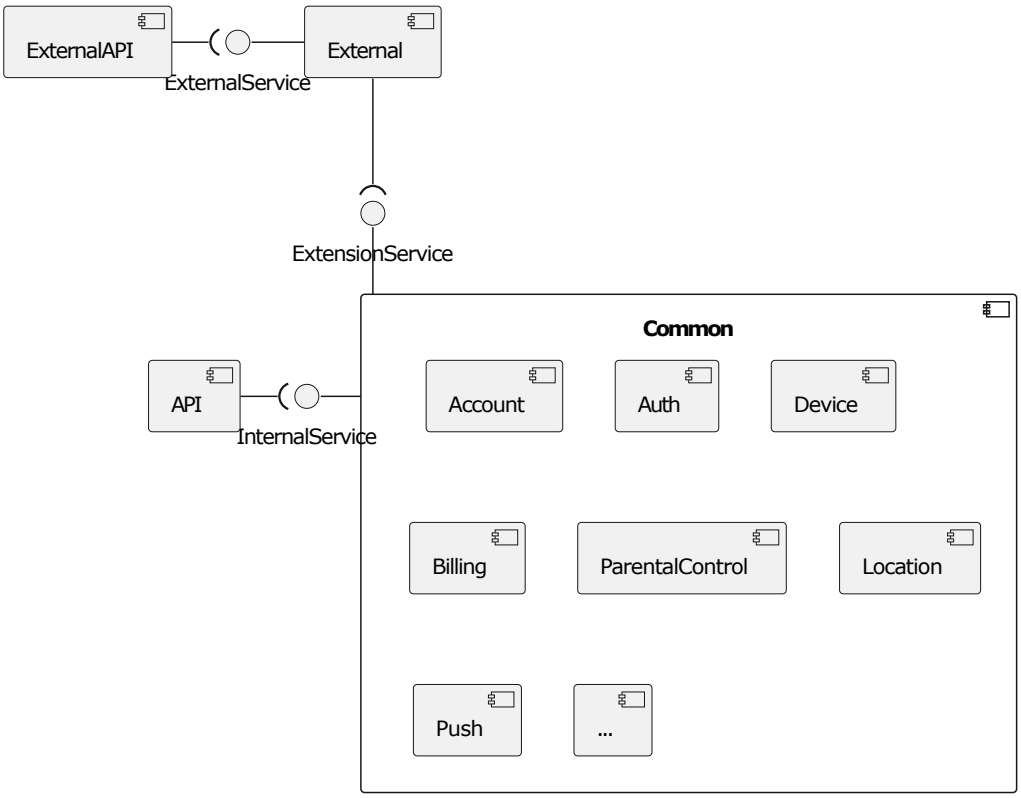


Figura 3.11: Visão de alto nível dos componentes.

Também aqui, todos os componentes eram desenvolvidos segundo a especificação Java EE [30] e para servidor de aplicações era usado o WildFly [78], outra implementação da plataforma Java EE. Das diversas especificações incluídas na plataforma Java EE, estas eram utilizadas extensivamente neste projeto:

- *Java Persistence (JPA)* [40]
- *Java API for RESTful Web Services (JAX-RS)* [41]
- *Enterprise JavaBeans (EJB)* [42]
- *Contexts and Dependency Injection for Java (CDI)* [43]
- *Java API for JSON Processing (JSON-P)* [44]
- *Java Transaction API (JTA)* [46]

A comunicação entre aplicações cliente e servidor é assente num protocolo sem estado (*Stateless*), feita exclusivamente por HTTPS e restrita a clientes devidamente autenticados e autorizados. A implementação da autorização é baseada no RFC 6750 [71], prescindindo-se do *Resource Owner* e enviando-se as credenciais diretamente ao *Authorization Server* — funcionalidade fornecida pelo subcomponente *Auth*.

3.2.2 Otimização de Desempenho

O primeiro passo, para otimizar o desempenho do SafePath, foi extrair métricas de utilização. Para simplificar a análise, foram extraídas apenas três métricas:

- tempo de processamento de cada pedido à API;
- em que pedidos à base de dados se desdobrava cada pedido à API;
- tempo de execução de cada pedido à base de dados.

Com estes dados, foi possível verificar que as aplicações cliente faziam vários pedidos repetidos e desnecessários à API, principalmente pedidos de leitura. Alguns referiam-se a dados que raramente alteravam e que passaram a ficar armazenados nas aplicações e refrescados periodicamente. Outros obtinham informação de várias entidades simultaneamente, o que originava vários pedidos à base de dados ou pedidos que cruzavam múltiplas tabelas o que aumentava o tempo total de processamento. Como nem sempre toda a informação obtida era necessária, e também porque não respeitavam as diretrizes de um serviço *RESTful* [76], os pedidos foram separados em pedidos menores, cada um relacionado apenas com uma entidade (*resource*). Apesar de estas duas alterações melhorarem o desempenho, os melhores resultados foram obtidos com alterações ao modelo de dados e revisão de todos os pedidos à base de dados. Os parágrafos abaixo resumem as ações realizadas para melhorar o desempenho da base de dados.

O modelo de dados foi completamente revisto, principalmente para melhorar o desempenho, mas também para melhorar a clareza, coerência e manutenção. As principais alterações foram:

- adicionar chave primária, em falta em algumas tabelas;
- normalizar os tipos de dados para garantir que colunas relacionadas tinham o mesmo tipo e tamanho;
- normalizar os nomes das tabelas e colunas para usarem a mesma convenção — esta alteração não teve impacto no desempenho, mas facilitou a compreensão e manutenção do modelo de dados;
- remover colunas que já não eram necessárias;
- garantir que todas as referências de integridade estavam definidas e tinham o índice correspondente;
- rever todas as consultas feitas à base de dados, começando por aquelas que eram executadas com maior frequência e as que tinham um tempo de execução maior — em alguns casos acrescentar índices em falta e/ou reescrever a consultas para tirar partido das otimizações feitas pelo motor de base de dados MySQL;

- manter em memória, no servidor aplicacional os resultados das consultas que raramente alteravam, para diminuir os pedidos feitos à base de dados;
- mover os dados binários, como imagens de perfil, da base de dados para um serviço externo, otimizado para responder a este tipo de requisitos — esta alteração implicou uma mudança na arquitetura inicial.

Apesar de ser desenvolvido de forma genérica, o SafePath era preferencialmente instalado na plataforma AWS e, para manter os custos baixos, havia também o objetivo de usar o mínimo possível de recursos com a infraestrutura. Após a aplicação destas alterações, o candidato trabalhou em conjunto com a equipa de testes, para ajudar a definir o processo de testes de carga automáticos que serviria para comprovar que o SafePath conseguia suportar o número de utilizadores pretendido, mantendo os custos com a infraestrutura sob controlo. Os testes de carga ajudaram a identificar outras áreas que careciam de otimização, como libertar memória e outros recursos quando não já não eram necessários, para evitar problemas com falta de memória e com falta de descritores de ficheiros.

O desempenho do SafePath passou, desde então, a ser um requisito permanente e faz parte das diretrizes de desenvolvimento que devem ser seguidas obrigatoriamente, principalmente nas alterações ao modelo de dados[8], mas também nas restantes áreas — cuidados com o uso de memória, uso de operações assíncronas quando se interage com serviços externos para libertar as ligações de rede o mais rápido possível, etc. Para se garantir que o desempenho não degrada à medida que são acrescentadas novas funcionalidades, os testes de carga são executados a cada nova versão.

Terminada a fase de otimização de desempenho, o candidato esteve envolvido em todas as fases do ciclo de vida do desenvolvimento de novas funcionalidades, tentando garantir que estas não iam contra a arquitetura estabelecida nem degradariam o desempenho. Após algumas iterações, foi possível observar potenciais melhorias que poderiam simplificar o desenvolvimento e a manutenção do SafePath.

3.2.3 Colunas JSON Tipadas

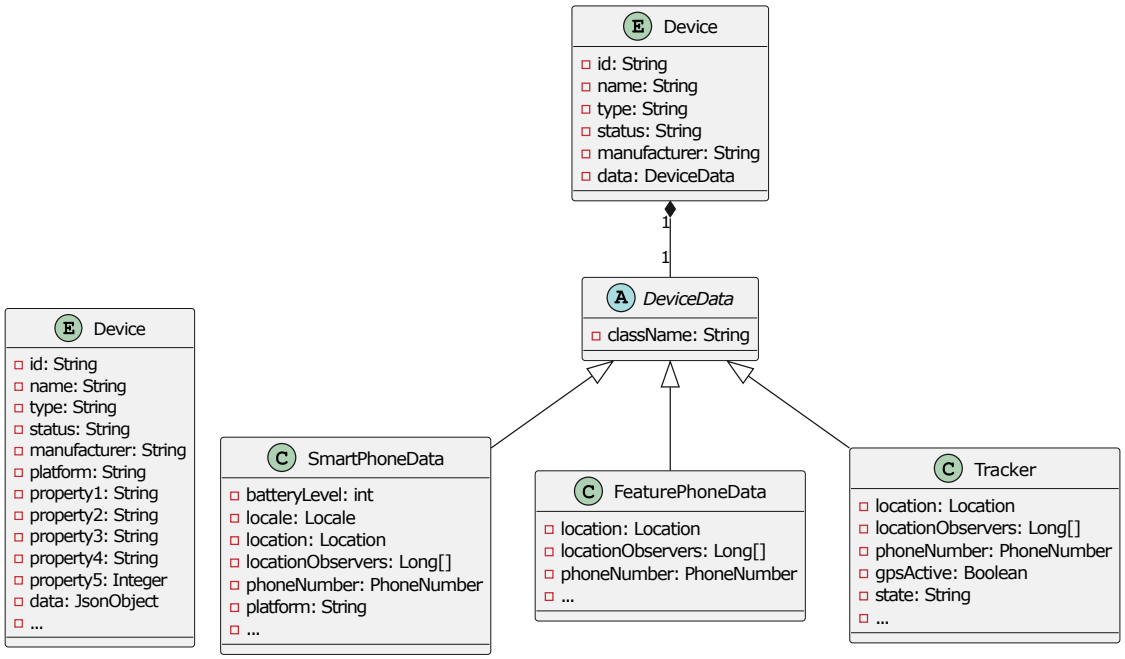
Observou-se que parte das novas funcionalidades envolviam adicionar ou remover propriedades ao modelo de dados, ver Modelo de Dados Simplificado, que correspondia a adicionar ou remover colunas em tabelas de base de dados. Este processo implicava fazer alterações em vários níveis, nomeadamente:

- alterar definição da tabela no esquema de base de dados — ficheiro Structured Query Language (SQL) com comandos Data Definition Language (DDL);
- criar procedimentos de atualização do esquema de base de dados para ambientes em utilização — um ou vários ficheiros SQL com comandos DDL;

[8]Tipo de dados, índices, referências de integridade e operações de consulta à base de dados.

- alterar entidade JPA para aplicar as alterações feitas ao esquema de base de dados;
- aplicar as alterações ao esquema de base de dados de cada ambiente em utilização — normalmente com paragem de serviço;
- instalar a nova versão em cada ambiente.

A maioria das novas colunas referiam-se à mesma tabela e estavam relacionadas com informação de estado dinâmica ou com colunas opcionais de tabelas genéricas. Essa tabela já tinha uma coluna genérica do tipo JSON que, no contexto desta melhoria, foi transformada numa coluna JSON tipada, e complementada com todas as propriedades de estado e propriedades opcionais. A Figura 3.12 demonstra esta transformação para a entidade *Device*[9], [10].



(a) Estado inicial. (b) Estado final.

Figura 3.12: Transformação da entidade *Device* para usar uma coluna JSON tipada.

Com esta alteração, o processo de adicionar ou remover propriedades ao modelo de dados, implica apenas alterar a classe ou capacidade correspondente ao dispositivo e instalar a nova versão em cada ambiente. Isto tornou o processo menos oneroso, principalmente porque deixa de ser necessário criar comandos DDL e aplicá-los em cada ambiente, minimizando o tempo de paragem do serviço e simplificando o processo de atualização.

[9] Na Figura 3.12(a), os nomes das propriedades (`property...`) são meramente indicativos.

[10] A Figura 3.12(b) representa apenas parcialmente a hierarquia de `DeviceData`. Como vários tipos de dispositivos podem partilhar comportamento, o SafePath tem o conceito de Capacidade, do inglês *Capability*, para lidar com o comportamento comum, por exemplo: *LocationCapability*, *PhoneCapability*, etc.

Depois da experiência positiva com a entidade *Device*, este modelo foi aplicado às entidades *PricePlan*, *Account*, *Profile* e *FamilyEvent* e mais tarde a quase todas as outras tabelas. Para além dos benefícios já descritos, outro efeito positivo de se ter maior estabilidade do esquema de base de dados entre versões, foi a diminuição de alterações necessárias ao processo ETL que envia os dados para um Armazém de Dados (*Data Warehouse*) [75].

3.2.4 Reestruturação da API

As alterações descritas na secção anterior permitiram simplificar o processo de desenvolvimento e a manutenção da API, mas à medida que se adicionavam novas funcionalidades, aumentava a sua complexidade. Por norma, cada recurso REST está associado a uma entidade do modelo de dados e expõe quatro métodos base, relacionados com a criação, leitura, alteração e remoção[11]. Destes, o método de leitura e o método de alteração padrão têm algumas desvantagens.

Quando se pretende ler apenas uma propriedade de uma entidade, o método de leitura padrão obriga a ler toda a entidade e quando se pretende alterar apenas uma propriedade de uma entidade, o método de alteração padrão obriga a enviar a entidade completa. Como se pode ver, o tamanho dos pedidos de leitura e alteração são, geralmente, maiores do que o necessário. No caso do método de alteração, há ainda um potencial problema de concorrência, quando dois pedidos simultâneos tentam alterar propriedades diferentes de uma mesma entidade, o segundo pedido a ser processado vai descartar as alterações feitas pelo primeiro.

Os impactos negativos do método de leitura, da forma como está especificado, não são muito significativos, mas foi possível minimizá-los através do uso de um apontador JSON para a propriedade pretendida[12]. Num caso normal do método de leitura, o pedido HTTP é da forma `GET /resource/id`, em que `resource` é o nome da entidade, e com a extensão para usar um apontador, o pedido HTTP é da forma `GET /resource/id/path/to/property`, em que `/path/to/property` representa o caminho para a propriedade pretendida — este elemento é opcional, se omitido, é devolvida a entidade completa.

Para minimizar os impactos negativos do método de alteração, foi usado o JSON Patch, tal como definido pelo RFC 6902 [31][13], que permite enviar apenas as alterações à entidade, em vez de enviar a entidade completa através do uso de seis operações possíveis: `add`, `remove`, `replace`, `move`, `copy` e `test`, ver Exemplo de Utilização do JSON Patch. Com o JSON Patch foi possível minimizar a perda de informação que podia ocorrer com alterações simultâneas à mesma entidade, mas quando se tratava de alterações a propriedades de tipo lista, continuava a ser possível a ocorrência de perda de informação.

[11] Estes métodos correspondem aos métodos HTTP `POST`, `GET`, `PUT` e `DELETE`.

[12] Um apontador JSON, definido pelo RFC 6901 [32], permite identificar um valor específico num documento JSON.

[13] O JSON Patch usa o método HTTP `PATCH`.

Após análise, o candidato implementou uma extensão ao RFC 6902 [31] que tirava partido da operação `test`[14]. A solução consiste em testar previamente se o valor do elemento da lista que se pretende alterar é igual ao último valor conhecido e se diferente, a operação falha, evitando-se assim a perda de informação[15]. A alternativa ao uso do JSON Patch, para enviar alterações parciais, era a criação de um método por cada propriedade que se pretendesse alterar individualmente, o que obrigava a alterações constantes à API e às correspondentes alterações nas aplicações cliente.

Alguns dos recursos já definiam métodos para alterar propriedades individualmente, mas com a introdução do JSON Patch todos esses métodos foram removidos, o que reduziu a quantidade de código que era necessário manter. Para além da remoção dos métodos descritos anteriormente, foram removidos também parte dos métodos `PUT`, privilegiando-se o uso do método `PATCH`.

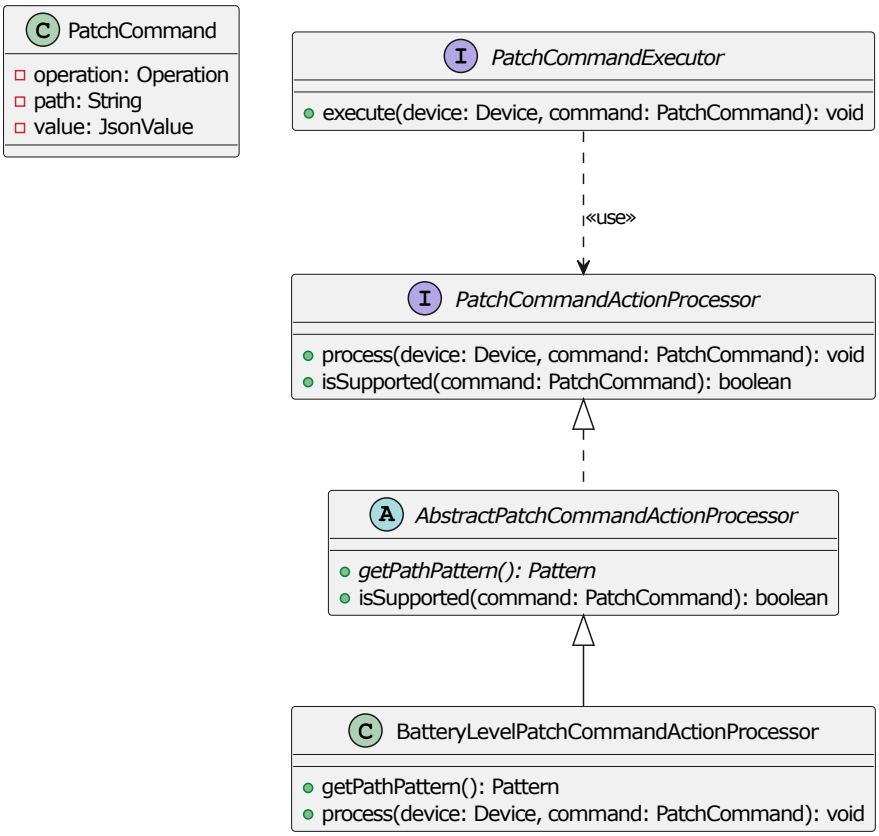


Figura 3.13: Motor de execução de ações implícitas.

O uso do método `PATCH` possibilitou outra simplificação, não antevista inicialmente, ao permitir simplificar o requisito de executar ações quando determinadas propriedades são alteradas. A forma tradicional de cumprir esse requisito seria enviar a entidade completa e

[14] A operação `test` permite testar se uma determinada propriedade tem determinado valor.

[15] Desde 2014 que se está a recolher informação para uma potencial revisão ao RFC 6902 [31]. Entre as propostas, estão duas que resolveriam o problema encontrado [35, 34]

verificar se determinada propriedade foi alterada ou através de um método específico, semelhante aos descritos anteriormente, que, para além de alterar a propriedade, executava também a ação pretendida. Como a especificação JSON Patch [31] define que cada operação tem de ter obrigatoriamente um atributo `path`, apontador para a propriedade relacionada com a operação, ver Operações JSON Patch, é possível usar esse atributo para identificar que propriedades foram alteradas. A Figura 3.13 demonstra como esta simplificação foi possível fazendo um pós-processamento do método `PATCH`. Cada ação que pretenda executar deve apenas implementar a interface `PatchCommandActionProcessor`, e será automaticamente injetada no motor de pós-processamento `PatchCommandExecutor`. Este motor é responsável por executar cada ação suportada pelo comando `PatchCommand` — normalmente, o atributo `path` do comando é usado para verificar se este pode ser processado por uma ação, pelo que a maioria das ações são subclasses de `AbstractPatchCommandActionProcessor`, que usa uma expressão regular para fazer a verificação.

Tomando como exemplo a propriedade `batteryLevel`, se quisermos ter como ação notificar todos os membros da família, sempre que o valor fique abaixo de 15%, bastará criar uma classe, como a `BatteryLevelPatchCommandActionProcessor`, que deverá suportar comandos cujo atributo `path` tenha o valor `/data/batteryLevel`. Esta estratégia foi

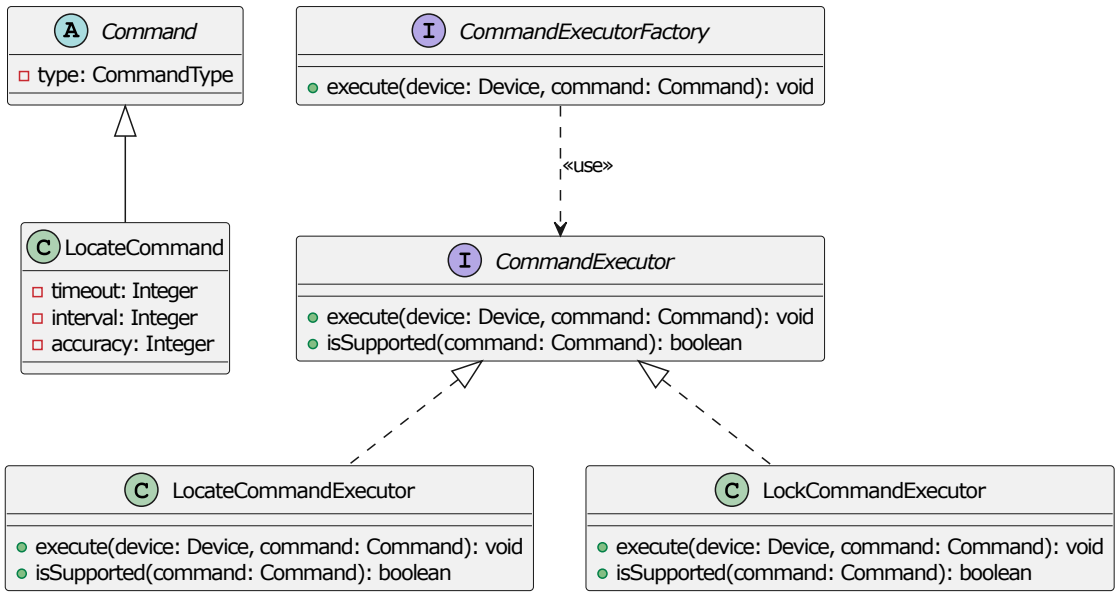


Figura 3.14: Motor de execução de comandos.

também aplicada para executar comandos explicitamente, ao contrário da execução de ações implícitas descritas anteriormente. Alguns exemplos concretos desses comandos são:

- *Locate* — usado para localizar um dispositivo;
- *Ring* — usado para tocar um dispositivo;

- *Lock* — usado para bloquear um dispositivo;
- *Wipe* — usado para limpar os dados de um dispositivo.

Como se pode ver na Figura 3.14, a implementação dos comandos explícitos é muito semelhante à implementação das ações implícitas[16].

Com a reestruturação, as alterações à API estabilizaram porque se tornou possível adicionar novas funcionalidades sem a alterar. A Figura 3.15 apresenta a estrutura de alguns dos recursos REST, cada um associado a uma entidade, onde se pode comprovar a estabilidade alcançada — a execução de ações implícitas é comum a todos os recursos, ao passo que a execução de comandos explícitos apenas é necessária para alguns recursos. Antes desta reestruturação, alguns recursos poderiam ter algumas dezenas de métodos.

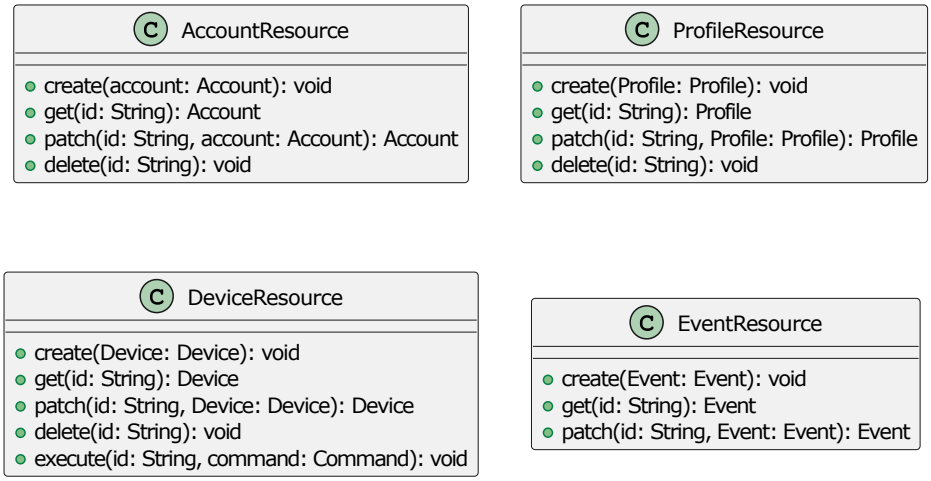


Figura 3.15: Estrutura dos recursos REST.

3.2.5 Integração de Produtos Concorrentes

Para melhorar o produto SafePath e, principalmente, para aumentar o número de clientes, a Smith Micro Software, Inc. adquiriu, em 2020, o negócio com operadoras móveis da Circle Media Labs, Inc. [63] (*Circle*) e, em 2021, adquiriu o negócio Family Safety da Avast [64] (*Ring*[17]). Com estas aquisições, o SafePath posicionou-se como líder global de serviços de segurança familiar para operadoras móveis. Neste contexto, o objetivo do candidato era o de integrar ambas as plataformas no SafePath, primeiro a *Circle* e depois a *Ring*, aproveitando o que de melhor tinha cada uma das soluções. Finalizada a integração, todas as implementações deveriam ser migradas para a nova plataforma e as implementações antigas descontinuadas.

[16]Já foi ponderado juntar as duas implementações numa única implementação, algo que poderá ser feito quando for necessário acrescentar nova funcionalidade a estes motores.

[17]*Ring* é o nome de código do Family Safety da Avast

3.2.5.1 *Circle*

Do produto da *Circle*, apenas se pretendia integrar os componentes relacionados com o controlo parental, ver Figura 3.16, os restantes componentes seriam descontinuados, quando todos os clientes fossem migrados para a nova plataforma. A estratégia de migração estava assente nos seguintes requisitos:

- a versão atual (*Circle*) e versão nova (SafePath) devem coexistir;
- a versão atual não deve ser alterada, tanto nas aplicações cliente como no servidor aplicacional;
- a funcionalidade de controlo parental utilizará exclusivamente o componente *ParentalControl* da *Circle* — o componente existente no SafePath será removido;
- a migração para a versão nova deve ser transparente para o utilizador — a sessão deve ser mantida e as definições devem ser todas migradas;
- deve haver interoperabilidade na funcionalidade de controlo parental — não será possível garantir a interoperabilidade completa, as alterações feitas com a versão nova serão visíveis na versão atual, mas as alterações feitas com a versão atual não serão visíveis na versão nova.

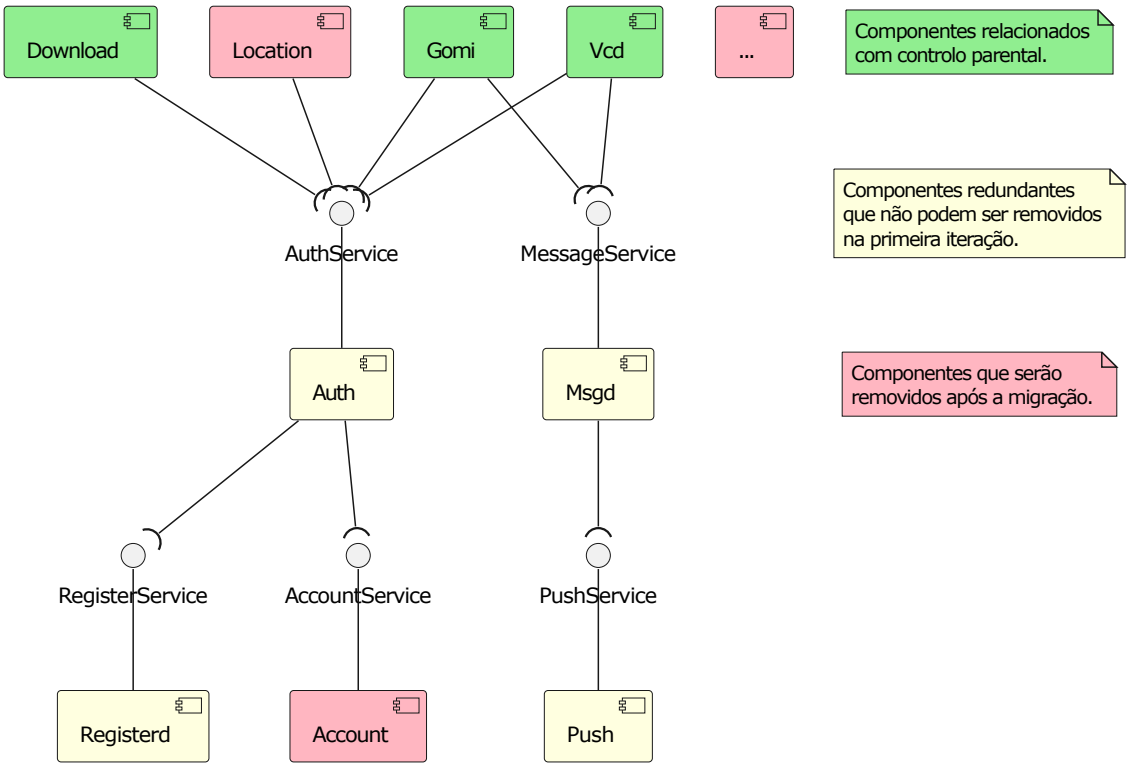


Figura 3.16: Componentes principais da *Circle*.

O produto da *Circle* usa exclusivamente a linguagem de programação Go[18] [70] e, por uma questão de simplicidade, deveria manter-se inalterado, pelo menos enquanto era preciso dar suporte à versão atual[19]. O esforço de integração foi dividido em duas fases. A primeira fase estava relacionada com o requisito de manter a sessão após a migração e consistiu em criar um módulo de autenticação e autorização de raíz que substituiria os componentes *Auth* do SafePath e da *Circle*. A segunda fase estava relacionada com a integração do componente de controlo parental da *Circle*.

Para simplificar o desenvolvimento e a manutenção, as aplicações cliente do SafePath continuariam a comunicar apenas com as API do SafePath, incluindo a funcionalidade de gestão do controlo parental e as aplicações cliente da *Circle* continuariam a comunicar diretamente com as API da *Circle*[20]. O SafePath servirá de intermediário e adaptará os pedidos recebidos das aplicações cliente para os enviar para o componente *ParentalControl* da *Circle*. A Figura 3.17 representa o resultado da integração da *Circle*, ainda com alguns componentes redundantes que serão extraídos para um módulo comum, à semelhança do que foi feito para os componentes *Auth*.

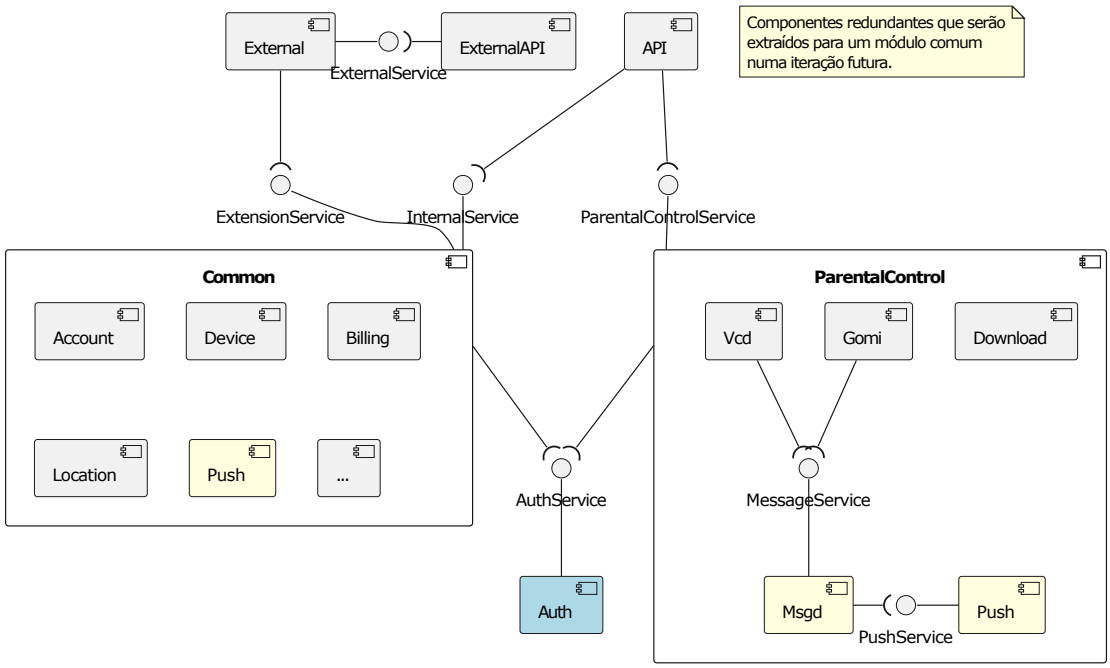


Figura 3.17: Componentes principais após integração da *Circle*.

3.2.5.2 *Ring*

Das funcionalidades suportadas pela *Ring*, ao contrário da *Circle*, nenhuma foi integrada diretamente no SafePath. Neste caso, após uma análise aprofundada das duas plataformas,

[18] Chegou a ser ponderada a migração para a linguagem Java, mas o único benefício, que seria ter uma linguagem de programação comum a todos os componentes, não compensava o esforço necessário.

[19] Foram feitas algumas alterações pontuais para facilitar a integração com o SafePath.

[20]

optou-se por complementar o SafePath com algumas funcionalidades que a *Ring* já tinha e que ainda não existiam no SafePath. A estratégia de integração consistiu em implementar de raíz essas funcionalidades, usando a implementação da *Ring* como base.

Mais do que trazer novas funcionalidades, a *Ring* serviu de inspiração para se melhorar a comunicação inter-serviços, que era maioritariamente assíncrona ao passo que no SafePath era maioritariamente síncrona. Com esta mudança de abordagem, começou-se a priviligiar o uso de filas de mensagens, que até então era usado apenas para comunicação com serviços externos ao SafePath, o que simplificou o desenvolvimento de novos serviços e, mais importante ainda, aumentou a tolerância à falha.

3.2.6 Outras Responsabilidades

O candidato tem um papel de liderança na manutenção e melhoria da consistência, extensibilidade e manutenibilidade da arquitetura geral do produto. Tem também uma voz ativa na identificação de problemas e oportunidades de melhoria contínua. Os items abaixo listam, de forma resumida, algumas das responsabilidades no projeto.

- Garantir que as novas propostas de funcionalidade não afetam negativamente o desempenho da plataforma.

- Garantir que as equipas de desenvolvimento implementam as novas funcionalidades de uma forma consistente com a arquitetura em uso. A arquitetura pode ser considerada um organismo vivo que evolui com os contributos da equipa, mas a cada evolução, tenta-se que todo o sistema mude completamente para a nova forma, para evitar inconsistências e promover uma melhor manutenção.

- Supervisionar a gestão da infraestrutura. A infraestrutura já passou por várias fases, começou por ser um processo manual, totalmente documentado para garantir que o processo de instalação é repetível. Como este processo de instalação era moroso e requeria um esforço grande de manutenção, foi feita a mudança para AWS CloudFormation [10], que permitia definir a infraestrutura como código, tendo resultado em grandes ganhos de produtividade, porque se minimizou a possibilidade de erro humano. Ainda que o AWS CloudFormation tenha apresentado grandes melhorias em relação ao processo manual, havia várias fases do processo que tinham de ser manuais e para se ultrapassarem estas limitações, todo o processo de criação e manutenção da infraestrutura foi mudado para Terraform [69] com Kubernetes [47].

- Esteve também envolvido na definição de processos e diretrizes transversais a todas as equipas, que se tornaram importantes para garantir coesão do processo de desenvolvimento, à medida que novas equipas se juntaram ao projeto.

- Orientar e trabalhar em conjunto com todos os membros da equipa SafePath. Um dos meios por excelência para promover a partilha de conhecimentos, tanto a nível

técnico como processual, são as revisões exaustivas aos pedidos de alteração de código.

- Mentor e dinamizador, desde 2018, da comunidade de prática [15, 14] inter-equipas, dedicada à área alargada do servidor aplicacional. Esta comunidade reúne semanalmente e discute temas gerais como melhores práticas, tecnologias novas que devam ser acompanhadas, apresentação de novas formas de trabalhar, etc., mas também discute temas relacionados com o SafePath, nomeadamente: como deverá ser implementada uma determinada funcionalidade, obtenção de consenso quando há desacordo sobre alguma implementação, definição de diretrizes relacionadas com a área do servidor aplicacional, entre outros.

4

Conclusão

O presente relatório descreve, de forma sucinta, o percurso profissional do candidato. Tal percurso denota, desde o início, rigor e excelência no trabalho realizado, resultado do grande impacto da experiência na MobiComp, empresa que influenciou significativamente o modo de raciocinar e abordar os problemas. Mesmo sendo uma empresa a dar os primeiros passos e apenas com recém-licenciados da Universidade do Minho, foi possível pôr em prática e aprofundar os conhecimentos, em várias das áreas de estudo, obtidos na formação académica (algoritmos e estruturas de dados, programação, bases de dados, redes, arquitetura e engenharia de software). Tal experiência facilitou o desenvolvimento da capacidade de compreensão e resolução de problemas em situações novas e não familiares e contribuiu também para o aprofundamento do trabalho em equipa.

Uma preocupação constante foi a aprendizagem contínua através de formações esporádicas patrocinadas pelas entidades empregadoras, da leitura regular de artigos técnicos e teóricos, da participação em formações e conferências em linha, e da troca de experiências com outras pessoas da área, quer da mesma empresa, quer de empresas diferentes. Mais recentemente, aceitou uma proposta para ser assistente convidado no ESTG — P.PORTO, de modo a aprofundar conceitos que tendem a ser menos utilizados no dia a dia das empresas, o que auxiliou o candidato a integrar conhecimentos, compreender e dominar o estado da arte em determinados campos da Engenharia Informática. Através desta aprendizagem contínua, foi possível obter as competências necessárias para a progressão profissional constante do candidato.

Outra preocupação é o cuidado contínuo com o trabalho produzido, pensando sempre que este pode servir, e regularmente serve, como referência para trabalhos futuros — aqui também se incluem as revisões regulares feitas a trabalho existente — o que promove um trabalho de excelência pautado pela melhoria contínua.

As funções desempenhadas têm munido o candidato de uma visão inovadora e crítica das soluções desenvolvidas; de autonomia profissional; de capacidade para comunicar e documentar as conclusões, os conhecimentos e os raciocínios subjacentes às soluções propostas; e competências para lidar com as diferentes facetas de um sistema informático complexo. Em praticamente todos os projetos, o candidato esteve envolvido em todas as

fases do ciclo de vida do desenvolvimento, mas com maior responsabilidade nos projetos ASAP54 e SafePath, onde foi o elemento da equipa com mais experiência. A lista abaixo descreve as contribuições em cada uma das fases do ciclo de vida do desenvolvimento tal como definidas para o SafePath — não varia muito do que é comummente preconizado pela indústria.

- **Planeamento** — Colaborar com os clientes finais, externos ou internos, definir o âmbito e analisar viabilidade técnica e operacional.
- **Análise de Requisitos** — Documentar e criar diagramas que ajudem à compreensão do que é pedido. Na maior parte das vezes, usando ferramentas como o Jira ou Redmine.
- **Desenho** — Nas empresas onde trabalhou, esta fase por vezes não existe explicitamente. Por vezes é feita em conjunto com a análise de requisitos, outras vezes em conjunto com o desenvolvimento ou então dividida entre estas duas fases.
- **Desenvolvimento** — Nesta fase o candidato contribuiu com: acompanhamento das equipas de desenvolvimento, programação em par, revisões exaustivas de código e, sempre que necessário, com código. Para além de garantir que os requisitos são cumpridos, garante também que as diretrizes são seguidas, que existem testes unitários e/ou de integração e que a manutenção e o desempenho não são comprometidos.
- **Testes** — As diretrizes internas de desenvolvimento dizem que antes de se submeter código para revisão por pares, para além dos testes unitários e/ou de integração, o autor precisa de testar as alterações num ambiente local ou remoto. Depois da revisão, as alterações devem ser testas por outra pessoa antes de serem submetidas para o repositório de código.
- **Implantação** — O processo de implantação e instalação teve um contributo grande do candidato, começando por ser uma sequência manual de passos propensa a falha humana, até ser um processo automatizado.
- **Manutenção** — O SafePath está continuamente em manutenção, com entregas periódicas, normalmente com novas funcionalidades, mas também com correções de potenciais problemas encontrados durante a utilização.

4.1 Competências de um Mestre em Engenharia Informática

Confrontando as competências obtidas ao longo do seu percurso profissional, descritas neste relatório, com os objetivos específicos do curso de Mestrado em Engenharia Informática, expressos no Artigo 3.º do Regulamento n.º 122/2010, o candidato considera que os cumpre na totalidade, tendo em conta que:

1. **alínea a)** — compreende e domina o estado da arte da Engenharia Informática, pelo trabalho desenvolvido como arquiteto e como mentor tecnológico das várias equipas com que trabalhou;

2. **alínea b)** — domina com profundidade a vertente de desenvolvimento de *software*, em todas as suas áreas, usando diferentes linguagens de programação e metodologias de desenvolvimento, com coordenação de várias equipas e interagindo com o cliente final, principalmente nas fases iniciais, mas também ao longo do projeto para garantir alinhamento de expetativas;

3. **alínea c)** — em todos os projetos em que esteve envolvido, teve de lidar com as diferentes facetas de um sistema informático complexo (infraestrutura, bases de dados, comunicação, *software*, utilizadores, etc.), com integração ou composição de múltiplos componentes, compreendendo de forma rigorosa cada um dos componentes e suas interações;

4. **alínea d)** — demonstra capacidade crítica, iniciativa e criatividade para resolver problemas complexos, ver ASAP54, mas também para simplificar processos existentes, como se verifica, por exemplo, em Colunas JSON Tipadas e Reestruturação da API;

5. **alínea e)** — realizou vários projetos de Engenharia Informática, alguns com papel de liderança técnica (ASAP54 e SafePath) e outros com contributos significativos, como os projetos desenvolvidos na MobiComp, Microsoft e Critical Software;

6. **alínea f)** — trabalhou sempre envolvido numa equipa ou como facilitador entre várias equipas, para além do papel de mentor, e com regularidade tem de apresentar objetivos, soluções e resultados a não especialistas e a utilizadores dos resultados do seu trabalho, nomeadamente aos clientes finais;

7. **alínea g)** — tratou dos aspetos de ética profissional, respeitando as normas e as leis que regem os países onde os projetos eram utilizados, nomeadamente os requisitos de privacidade dos utilizadores[1] e de propriedade intelectual;

8. **alínea h)** — como mentor da comunidade de prática inter-equipas, dedicada à área alargada do servidor aplicacional, teve frequentemente de relatar, investigar, sintetizar, apresentar e argumentar um tema técnico e de demonstrar capacidade de visão crítica das soluções desenvolvidas;

9. **alínea i)** — o percurso profissional foi pautado por uma aprendizagem contínua através: de formações esporádicas patrocinadas pelas entidades empregadoras; da leitura regular de artigos técnicos e teóricos; da participação em formações e conferências em linha; da troca de experiências com outras pessoas da área, quer

[1]Por exemplo, cifrando os dados privados quando são persistidos, protegendo o acesso aos dados e criando ferramentas de auditoria que garantem que os requisitos de privacidade são cumpridos.

da mesma empresa, quer de empresas diferentes; da docência como assistente convidado no ESTG — P.PORTO.

Bibliografia

[1] *.NET*. URL: https://dotnet.microsoft.com/en-us/ (acedido em 2023-06-08) (ver p. 3).

[2] *Affiliate Marketer: Definition, Examples, and How to Get Started*. URL: https://www.investopedia.com/terms/a/affiliate-marketing.asp (acedido em 2023-08-12) (ver p. 15).

[3] *Amazon Web Services*. URL: https://aws.amazon.com/ (acedido em 2023-06-08) (ver p. 4).

[4] *Apache Solr*. URL: https://solr.apache.org/ (acedido em 2023-06-08) (ver pp. 4, 15).

[5] *Apple Push Notification service*. URL: https://developer.apple.com/notifications/ (acedido em 2023-08-08) (ver p. 14).

[6] *ASAP54*. URL: https://web.archive.org/web/20140729045817/http://www.asap54.com/ (acedido em 2023-06-08) (ver p. 4).

[7] *ASAP54 - Get The Look ASAP - Vogue*. URL: https://www.vogue.co.uk/gallery/new-app-asap54-shazam-for-fashion (acedido em 2023-06-20) (ver p. 7).

[8] *ASAP54 - O Shazam da moda - Downshifting*. URL: https://downshifting.blogs.sapo.pt/asap54-o-shazam-da-moda-9811 (acedido em 2023-06-20) (ver p. 7).

[9] *ASAP54: Search and Shop for Fashion - CCG*. URL: https://web.archive.org/web/20220925043409/https://ccg.pt/my-product/asap54-search-and-shop-for-fashion/ (acedido em 2023-06-20) (ver p. 9).

[10] *AWS CloudFormation*. URL: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html (acedido em 2023-06-09) (ver pp. 5, 31).

[11] G. Booch, J. Rumbaugh e I. Jacobson. *Unified Modeling Language User Guide, The*. Second. Addison-Wesley Professional, 2005. ISBN: 9780321267979 (ver p. 4).

[12] *Business Process Model and Notation*. URL: https://www.bpmn.org/ (acedido em 2023-06-08) (ver p. 4).

[13] *C#*. URL: https://dotnet.microsoft.com/en-us/languages/csharp (acedido em 2023-06-08) (ver p. 3).

[14] *Communities of Practice*. URL: https://scaledagileframework.com/communities-of-practice/ (acedido em 2023-09-03) (ver p. 32).

[15] *Communities of Practice (CoP)*. URL: https://www.pmi.org/disciplined-agile/people/communities-of-practice (acedido em 2023-09-03) (ver p. 32).

[16] *Critical Software*. URL: https://www.criticalsoftware.com/ (acedido em 2023-06-08) (ver p. 3).

[17] *Data Modeling with DODS*. URL: https://web.archive.org/web/20010409085839/http://dods.enhydra.org/project/aboutProject/index.html (acedido em 2023-06-08) (ver p. 3).

[18] *Extensibility*. URL: https://en.wikipedia.org/wiki/Extensibility (acedido em 2023-08-22) (ver p. 20).

[19] *Extract, transform, load*. URL: https://en.wikipedia.org/wiki/Extract,_transform,_load (acedido em 2023-08-08) (ver p. 14).

[20] *GlassFish*. URL: https://glassfish.org/ (acedido em 2023-06-25) (ver p. 10).

[21] *Google Cloud Messaging*. URL: https://developers.google.com/cloud-messaging/ (acedido em 2023-08-08) (ver p. 14).

[22] *IBM*. URL: https://www.ibm.com/ (acedido em 2023-06-08) (ver p. 4).

[23] *IBM Business Process Manager*. URL: https://www.ibm.com/docs/en/bpm (acedido em 2023-06-08) (ver p. 4).

[24] *Identify Geographical Location and Proxy by IP Address*. URL: https://www.ip2location.com/ (acedido em 2023-08-08) (ver p. 14).

[25] *iMobileMagic*. URL: https://web.archive.org/web/20160504195700/http://imobilemagic.com/ (acedido em 2023-06-21) (ver p. 9).

[26] *Infopulse*. URL: https://web.archive.org/web/20010517014846/http://www.infopulse.pt/ (acedido em 2023-06-08) (ver p. 2).

[27] *Inversion of Control Containers and the Dependency Injection pattern*. URL: https://martinfowler.com/articles/injection.html (acedido em 2023-08-22) (ver p. 20).

[28] *Jakarta EE*. URL: https://jakarta.ee/ (acedido em 2023-06-25) (ver p. 10).

[29] *Java*. URL: https://en.wikipedia.org/wiki/Java_%28programming_language%29 (acedido em 2023-06-08) (ver p. 2).

[30] *Java EE at a Glance*. URL: https://www.oracle.com/java/technologies/java-ee-glance.html (acedido em 2023-06-25) (ver pp. 10, 21).

[31] *JavaScript Object Notation (JSON) Patch*. URL: https://datatracker.ietf.org/doc/html/rfc6902 (acedido em 2023-08-28) (ver pp. 25–27).

[32] *JavaScript Object Notation (JSON) Pointer*. URL: https://datatracker.ietf.org/doc/html/rfc6901 (acedido em 2023-08-28) (ver p. 25).

[33] *JSON (JavaScript Object Notation)*. URL: https://www.json.org/ (acedido em 2023-06-08) (ver p. 4).

[34] *JSON Patch 2 - Provide space for additional context/metadata*. URL: https://github.com/json-patch/json-patch2/issues/27 (acedido em 2023-09-03) (ver p. 26).

[35] *JSON Patch 2 - Value-based array operations*. URL: https://github.com/json-patch/json-patch2/issues/18 (acedido em 2023-09-03) (ver p. 26).

[36] *JSR 173: Streaming API for XML*. URL: https://jcp.org/en/jsr/detail?id=173 (acedido em 2023-08-06) (ver p. 11).

[37] *JSR 206: Java™ API for XML Processing (JAXP) 1.3*. URL: https://jcp.org/en/jsr/detail?id=206 (acedido em 2023-08-06) (ver p. 11).

[38] *JSR 222: Java™ Architecture for XML Binding (JAXB) 2.0*. URL: https://jcp.org/en/jsr/detail?id=222 (acedido em 2023-08-06) (ver p. 11).

[39] *JSR 224: Java™ API for XML-Based Web Services (JAX-WS) 2.0*. URL: https://jcp.org/en/jsr/detail?id=224 (acedido em 2023-08-06) (ver p. 11).

[40] *JSR 338: Java™ Persistence 2.2*. URL: https://jcp.org/en/jsr/detail?id=338 (acedido em 2023-08-06) (ver pp. 11, 21).

[41] *JSR 339: JAX-RS 2.0: The Java™ API for RESTful Web Services*. URL: https://jcp.org/en/jsr/detail?id=339 (acedido em 2023-08-06) (ver pp. 11, 21).

[42] *JSR 345: Enterprise JavaBeans™ 3.2*. URL: https://jcp.org/en/jsr/detail?id=345 (acedido em 2023-08-06) (ver pp. 11, 21).

[43] *JSR 346: Contexts and Dependency Injection for Java™ EE 1.1*. URL: https://jcp.org/en/jsr/detail?id=346 (acedido em 2023-08-06) (ver pp. 11, 21).

[44] *JSR 353: Java™ API for JSON Processing*. URL: https://jcp.org/en/jsr/detail?id=353 (acedido em 2023-08-06) (ver pp. 11, 21).

[45] *JSR 356: Java™ API for WebSocket*. URL: https://jcp.org/en/jsr/detail?id=356 (acedido em 2023-08-06) (ver p. 11).

[46] *JSR 907: Java™ Transaction API (JTA)*. URL: https://jcp.org/en/jsr/detail?id=907 (acedido em 2023-08-06) (ver pp. 11, 21).

[47] *Kubernetes*. URL: https://kubernetes.io/ (acedido em 2023-09-02) (ver p. 31).

[48] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (ver p. i).

[49] *Lutris Enhydra*. URL: https://web.archive.org/web/20010409181706/http://enhydra.enhydra.org/project/aboutProject/index.html (acedido em 2023-06-08) (ver p. 2).

[50] *Microsoft My Phone*. URL: https://web.archive.org/web/20090523151518/http://sn1-p2.myphone.microsoft.com/mkweb/MoreInfo.po?tsid=1242988210006&mkt=en-US (acedido em 2023-06-08) (ver p. 3).

[51] *Microsoft Reaches for Clouds and Deepens Mobile Possibilities With Planned MobiComp Acquisition*. URL: https://news.microsoft.com/2008/06/26/microsoft-reaches-for-clouds-and-deepens-mobile-possibilities-with-planned-mobicomp-acquisition/ (acedido em 2023-06-08) (ver p. 3).

[52] *Microsoft SQL Server*. URL: https://www.microsoft.com/en-us/sql-server/ (acedido em 2023-06-08) (ver p. 3).

[53] *MobiComp*. URL: https://web.archive.org/web/20050301080656/http://www.mobicomp.com/static/ (acedido em 2023-06-08) (ver p. 2).

[54] *MySQL*. URL: https://www.mysql.com/ (acedido em 2023-06-25) (ver pp. 11, 20).

[55] *myTMN*. URL: https://web.archive.org/web/20050713074709if_/http://www.mytmn.pt:80/web/mytmn.po (acedido em 2023-06-08) (ver p. 2).

[56] *Oracle Database*. URL: https://www.oracle.com/database/ (acedido em 2023-06-08) (ver p. 3).

[57] *OrientDB*. URL: https://orientdb.org/ (acedido em 2023-08-06) (ver p. 11).

[58] *Pair Programming*. URL: https://www.agilealliance.org/glossary/pairing/ (acedido em 2023-08-17) (ver p. 19).

[59] *PostgreSQL*. URL: https://www.postgresql.org/ (acedido em 2023-06-08) (ver p. 3).

[60] *Representational State Transfer (REST)*. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (acedido em 2023-06-08) (ver p. 4).

[61] *SafePath*. URL: https://www.smithmicro.com/safepath/ (acedido em 2023-06-09) (ver p. 5).

[62] *Scrum*. URL: https://www.agilealliance.org/glossary/scrum/ (acedido em 2023-08-17) (ver pp. 4, 19).

[63] *Smith Micro Acquires Operator Business from Circle Media Labs, Inc.* URL: https://www.smithmicro.com/press-releases/2020/02/13/smith-micro-acquires-operator-business-from-circle-media-labs-inc/ (acedido em 2023-08-31) (ver p. 28).

[64] *Smith Micro Completes Acquisition of Family Safety Mobile Business from Avast*. URL: https://www.smithmicro.com/press-releases/2021/04/19/smith-micro-completes-acquisition-of-family-safety-mobile-business-from-avast/ (acedido em 2023-08-31) (ver p. 28).

[65] *Smith Micro Software*. URL: https://www.smithmicro.com/ (acedido em 2023-06-09) (ver p. 5).

[66] *SMS Express*. URL: https://web.archive.org/web/20070623035652/https://smsexpress.tmn.pt/smsexpress-web/app/main/Login.po (acedido em 2023-06-08) (ver p. 2).

[67] *Software framework*. URL: https://en.wikipedia.org/wiki/Software_framework (acedido em 2023-08-21) (ver p. 20).

[68] *Tableau - Business Intelligence and Analytics*. URL: https://www.tableau.com/ (acedido em 2023-06-25) (ver pp. 10, 14).

[69] *Terraform by HashiCorp*. URL: https://www.terraform.io/ (acedido em 2023-09-02) (ver p. 31).

[70] *The Go Programming Language*. URL: https://go.dev/ (acedido em 2023-08-31) (ver p. 30).

[71] *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. URL: https://datatracker.ietf.org/doc/html/rfc6750 (acedido em 2023-08-03) (ver pp. 12, 21).

[72] *The Template Method Pattern*. URL: https://www.pmi.org/disciplined-agile/the-design-patterns-repository/the-template-method-pattern (acedido em 2023-08-22) (ver p. 20).

[73] *This Visual-Recognition App Lets You Shop the Streets - Entrepreneur*. URL: https://www.entrepreneur.com/leadership/this-visual-recognition-app-lets-you-shop-the-streets/233880 (acedido em 2023-06-20) (ver p. 7).

[74] *Weka 3: Machine Learning Software in Java*. URL: https://www.cs.waikato.ac.nz/ml/weka/ (acedido em 2023-08-13) (ver p. 17).

[75] *What Is a Data Warehouse?* URL: https://www.oracle.com/pt/database/what-is-a-data-warehouse/ (acedido em 2023-08-25) (ver p. 25).

[76] *What is REST - REST API Tutorial*. URL: https://restfulapi.net/ (acedido em 2023-08-24) (ver p. 22).

[77] *What Is the Software Development Life Cycle (SDLC)?* URL: https://www.upwork.com/resources/what-is-software-development-life-cycle (acedido em 2023-08-17) (ver p. 19).

[78] *WildFly*. URL: https://www.wildfly.org/ (acedido em 2023-08-23) (ver p. 21).

[79] *Wireless Application Protocol*. URL: https://technical.openmobilealliance.org/Affiliates/WAP.html (acedido em 2023-06-08) (ver p. 2).

[80] D. H. Young. *Enhydra XMLC Java Presentation Development*. Sams White Books, 2002. ISBN: 9780672322112 (ver p. 3).

A

ASAP54

A.1 Árvore de Categorias

```
Accessories
  Belts & Braces
  Glasses & Sunglasses
  Gloves
  Hair Accessories
  Hats
  Keyrings & Chains
  Scarves
  Ties
  Umbrellas
  Wallets & Purses

Bags
  Backpacks
  Clutches
  Luggage
  Make Up
  Shoulder Bags
  Totes

Beauty & Grooming
  Bath & Body
  Fragrances
  Hair Care
  Make Up
    Eyes
    Face
```

```
    Lips
    Nails
  Skin Care

Clothing
  Activewear
    Leggings
    Pants
    Shorts
    Tops
  All in One
    Co-ords
    Jumpsuits
    Playsuits
  Blazers
  Coats
  Dresses
    Bridal
    Maxi
    Midi
    Mini
  Hoodies & Sweatshirts
  Jackets
  Jeans
  Knitwear
    Cardigans
    Jumpers
  Leggings
  Maternity
  Plus Size
    Beachwear
    Dresses
    Outerwear
    Skirts
    Tops
    Trousers
  Shirts
  Shorts
  Skirts
    Maxi
    Midi
```

```
    Mini
  Suits
  Swimwear
    Beach Clothing
    Bikini
    Swim Shorts
    Swimsuits
  Tops
    Crop Top
    Long Sleeve
    Short Sleeve
    Strapless
  Trousers
  Underwear
    Boxers
    Bras
    Briefs
    Hosiery
    Lingerie Sets
    Nightwear & Robes
    Shapewear
  Waistcoats

Jewellery
  Body Jewellery
  Bracelets & Cuffs
  Brooches & Pins
  Charms
  Cufflinks
  Earrings
  Necklaces & Pendants
  Rings
  Watches

Kids

Lifestyle
  Home

Shoes
  Boots
```

```
    Ankle Boots - Flats
    Ankle Boots - Heels
    Tall Boots - Flats
    Tall Boots - Heels
  Brogues
    Flats
    Heels
  Espadrilles
    Flats
    Heels
  Flip Flops
  Mules
    Flats
    Heels
  Sandals
    Flats
    Heels
  Shoes
    Flats
    Heels
  Sneakers
  Wedges

Trash

Unsupported
```

B

SafePath

B.1 Modelo de Dados Simplificado

A Figura B.1 representa, de uma forma simplificada, as entidades principais do modelo de dados, para melhor se compreender alguns dos conceitos apresentados.

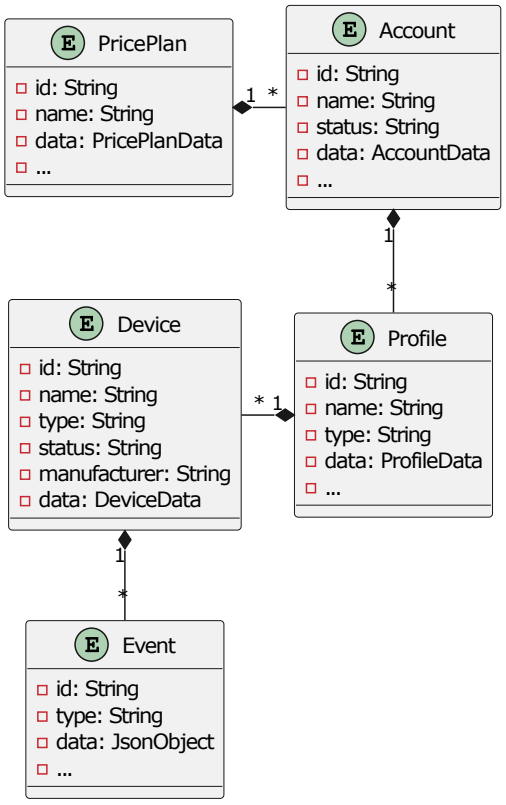


Figura B.1: Modelo de dados simplificado.

B.2 Categorias de Domínios Internet

Tabela B.1: Categorias de Domínios Internet.

Categoria	Descrição
ads	Advertisement.
adult	Some adult site from erotic to hard pornography.
aggressive	Aggressive sites.
astrology	Astrology
audio-video	Audio and video sites.
bank	Online banking.
bitcoin	Sites for bitcoin mining.
blog	Blogs sites.
celebrity	Sites about famous people, actors and magazines.
chat	Chat sites.
cheat	Sites which are designed to explain cheating on exams.
child	Any website allowed to children (less than 10 years old).
cleaning	Sites to disinfect, update and protect computers.
cooking	Sites for cooking.
cryptojacking	Mining site by hijacking.
dangerous-material	Sites which describe how to make bomb and some dangerous material.
dating	Dating and matching sites.
ddos	DDoS or Stresser sites
dialer	Dialer sites.
doh	Sites which provides DNS over HTTP service.
download	Sites which propose to download software.
drugs	Sites relative to drugs.
educational-games	Educational games sites.
filehosting	Websites which host files (pictures, video, etc.).
financial	Sites related to financial information.
forums	Forums sites.

gambling	Gambling and games sites, casino, etc.
games	Games sites (flash and online games).
hacking	Hacking sites.
jobsearch	Job searching sites.
lingerie	Sites for lingerie
malware	Any website which deliver malware.
manga	Any website related to manga and cartoons.
marketingware	Very special marketing sites.
mixed-adult	Websites which contain unstructured adult sections.
mobile-phone	Sites for mobile phone (rings, etc).
phishing	Phishing sites (same as malware category).
press	Any press (informational) sites.
radio	Internet radio sites.
reaffected	Websites which have been reaffected.
redirector	Some redirector sites, which are used to circumvent filtering.
remote-control	Site which allow remote control of user's dekstop.
sect	Sect related sites.
sexual-education	Websites which talk about sexual education and can be misdetected as porn.
shopping	Any shopping or seller center sites.
shortener	URL shortening sites
social-networks	Social networks sites.
sports	Sports.
stalkerware	Sites which sell spying software.
strict-redirector	Same as redirector, but with google, yahoo, and other cache/images search robots.
strong-redirector	Same as strict-redirector, but, for google, yahoo, we are only blocking some terms.
translation	Translation erlated sites.
update	Update sites for software or OS.
vpn	VPN sites.

warez	Warez sites.
webmail	Webmail sites.

B.3 Exemplo de Utilização do JSON Patch

Considerando o documento B.1 como o estado inicial, se se aplicarem as operações definidas em B.2, obtém-se como estado final o documento B.3. É de destacar a operação `test`, que permite testar se o elemento no índice `0` da propriedade `batteryLevel` ainda tem o valor `101`. Se tiver um valor diferente, nenhuma das operações definidas em B.2 são aplicadas e o documento permanece inalterado.

Listagem B.1: Documento original.

```
{
    "id": 1001,
    "name": "Xiaomi Mi 9",
    "type": "SmartPhone",
    "data": {
        "className": "com.smithmicro.safepath.device.DeviceData",
        "batteryLevel": 80,
        "manufacturer": "Xiaomi",
        "platform": "Android",
        "locationObservers": [
            101,
            203
        ]
    }
}
```

Listagem B.2: Operações JSON Patch.

```
[
    {
        "op": "replace",
        "path": "/data/batteryLevel",
        "value": 72
    },
    {
        "op": "test",
        "path": "/data/locationObservers/0",
        "value": 101
    },
    {
        "op": "replace",
        "path": "/data/locationObservers/0",
        "value": 202
    }
]
```

Listagem B.3: Documento final.

```
{
    "id": 1001,
    "name": "Xiaomi Mi 9",
    "type": "SmartPhone",
    "data": {
        "className": "com.smithmicro.safepath.device.DeviceData",
        "batteryLevel": 72,
        "manufacturer": "Xiaomi",
        "platform": "Android",
        "locationObservers": [
            202,
            203
        ]
    }
}
```