FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Gestão dinâmica do processo de atendimento à vítima na APAV

Susana Isabel do Vale Ventura de Sousa



Mestrado Integrado em Engenharia Informática e Computação

Orientador: António Miguel Pontes Pimenta Monteiro

Co-orientador: André Monteiro de Oliveira Restivo

14 de Julho de 2017

© Susana Ventura, 2017

Gestão dinâmica do processo de atendimento à vítima na APAV

Susana Isabel do Vale Ventura de Sousa

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Alexandre Miguel Barbosa Valle de Carvalho

Arguente: Helena Cristina Coutinho Duarte Rodrigues

Vogal: António Miguel Pontes Pimenta Monteiro

14 de Julho de 2017

Resumo

A utilização de Sistemas de Informação (SI) numa organização altera significativamente a sua produtividade, uma vez que permite a automatização e uniformização de processos e a centralização de dados. No entanto, em muitos casos, os SI são pouco ágeis e pouco versáteis, e implementar uma nova funcionalidade, ou alterar o modo de funcionamento para melhor adaptar o sistema às necessidades reais da organização, são tarefas complexas.

A Associação Portuguesa de Apoio à Vítima (APAV) é uma instituição particular de solidariedade social, que tem como missão prestar apoio às vítimas de crime e aos seus familiares e amigos. Faz parte do processo de atendimento e apoio a recolha, registo e tratamento de informação relacionada com os utentes da instituição e situações de vitimação.

Os sistemas informáticos de suporte ao atendimento atualmente disponíveis na APAV, apresentam problemas de funcionamento, interoperabilidade e limitações que dificultam o trabalho dos colaboradores e impossibilitam que a instituição possa melhorar o seu serviço.

O projeto apresentado nesta dissertação consiste no planeamento e parcial implementação de uma nova aplicação *web* com vista na centralização de dados, flexibilidade de gestão e simplificação do processo de atendimento. A centralização de dados passa por unificar a informação relativa aos atendimentos e aos processos de apoio, atualmente armazenada e tratada em dois sistemas distintos; a flexibilidade de gestão é obtida através da implementação de um sistema de edição de formulários de interface dinâmica, o que permite a manutenção dos formulários de atendimento independente de recursos externos, de acordo com as necessidades da instituição; a simplificação do processo de atendimento passa pela implementação de mecanismos que automatizam tarefas, até agora executadas manualmente pelos colaboradores da APAV, e pelo fornecimento de um conjunto de ferramentas úteis para a organização do seu trabalho.

O desenvolvimento da referida aplicação está associado a um estudo dos aspetos mais relevantes relacionados com alojamento de aplicações *web*, segurança e confidencialidade de dados, tecnologias e conceitos atuais de desenvolvimento de aplicações *web* e técnicas de implementação de formulários dinâmicos geridos pelo utilizador.

O sistema apresentado consiste no primeiro protótipo funcional daquele que será, no futuro, o novo SI de suporte ao atendimento à vítima na APAV. A fase de validação contou com a participação dos utilizadores finais da plataforma (colaboradores da APAV), que realizaram vários testes de usabilidade, de modo a avaliar se o sistema realmente corresponde às necessidades e quais os aspetos que poderão ser melhorados no futuro.

Abstract

Information Systems (IS) can make a considerable impact on a company's productivity, because it facilitates the automation and uniformization of processes and data centralization. However, in many scenarios, IS are not agile enough to implement new functionalities or accommodate workflow changes to better adapt to the real needs of the organization.

The Portuguese Association for Victim Support (APAV) is a private charitable organization that supports crime victims, as well as their family and friends. Currently, APAV uses software support systems to care for the victims. These systems have functionality problems, interoperability flaws, and limitations that make it difficult to collect the information needed, ultimately preventing APAV from improving its victim care process.

This dissertation presents the planning and partial implementation of a new web system that provides data centralization, management flexibility and care process simplification. The new system centralizes data by bringing care process information that is currently stored in two separate systems, it provides management flexibility with a dynamic forms user managed system which allows APAV to change forms used in the victim care service without the need of external resources, and it simplifies the care process by providing a set of tools for work and time organization and automating tasks currently done manually by APAV's employees.

The development of this new application is associated with a study of relevant aspects regarding the web applications hosting, security and confidentiality of data, modern technologies for web applications' development, and dynamic forms' implementation techniques.

The presented system is the first functional prototype of what will be the new IS for victim care process at APAV. At the validation phase, the platform's end users performed several usability tests to assess whether the system meets the indicated requirements and what aspects could be improved in the future.

Agradecimentos

Quero agradecer todos aqueles que contribuíram para o desenvolvimento deste projeto e desta dissertação, em especial:

Aos meus orientadores, o Professor Doutor Miguel Pimenta Monteiro e o Professor Doutor André Restivo, pelas indicações e pelo apoio;

Aos colaboradores da APAV por toda a disponibilidade e simpatia, em especial ao dr. João Lázaro, à Deolinda Santos e aos colaboradores do GAV Porto que participaram nos testes de validação do projeto;

Aos meus colegas e amigos, pela ajuda e por não me deixarem sentir sozinha, em especial à Ana, ao Pedro, ao Mira, ao Luís, ao Sandim, ao Maia, ao JP, ao Amorim, ao Carlos e à Raquel;

Ao Hugo, por todo o apoio, paciência e carinho que demonstra sempre, e por me fazer acreditar mais em mim;

Aos meus pais, Fernando e Dalila, e aos meus irmãos, Nuno e André, pela confiança, entusiasmo e força que me transmitem. O orgulho que sentem em mim não me deixa nunca desistir;

À Nala e à Yuki, por serem tolas e me fazerem rir;

Finalmente, agradeço ao resto da minha família, avós, tios e primos, pelo apoio e por serem a melhor família onde eu poderia ter crescido.

Susana Ventura

"In a gentle way, you can shake the world."

Mahatma Gandhi

Conteúdo

1 Introdução **1**
1.1 Contexto . . . 1
1.2 Motivação . . . 1
1.3 Projeto e objetivos . . . 2
1.4 Estrutura do documento . . . 3

2 Alojamento e segurança das aplicações *web* **5**
2.1 Ambientes *Cloud* . . . 5
2.2 Segurança e confidencialidade de dados . . . 6
2.3 Vulnerabilidades e ataques . . . 7
2.3.1 OWASP Top 10 2017 . . . 8
2.4 Replicação da base de dados . . . 13
2.5 Controlo de acessos . . . 14
2.5.1 Autenticação . . . 14
2.5.2 Autorização . . . 17
2.5.3 Imputabilidade (*audit logs*) . . . 20
2.6 Algoritmos de *hashing* e encriptação de dados . . . 20
2.6.1 Armazenamento de *passwords* . . . 21
2.7 Sumário . . . 22

3 Interfaces dinâmicas e gestão de formulários pelo utilizador **25**
3.1 Tecnologias e conceitos para o desenvolvimento de aplicações *web* com interface dinâmica . . . 25
3.1.1 *Single Page Applications* (SPA) . . . 26
3.1.2 DOM virtual . . . 26
3.1.3 *Web components* . . . 27
3.1.4 TypeScript . . . 27
3.1.5 ECMAScript 6 . . . 28
3.2 Gestão de formulários dinâmicos . . . 30
3.2.1 Bases de dados relacionais e NoSQL . . . 31
3.2.2 Formulários *web* geridos pelo utilizador . . . 33
3.2.3 Trabalhos relacionados . . . 34
3.3 Sumário . . . 37

4 Processo de atendimento na APAV **39**
4.1 A Associação Portuguesa de Apoio à Vítima . . . 39
4.2 Plataforma de Processos de Apoio Online . . . 40
4.3 Atendimento em GAV e em LAV . . . 41

4.4 Análise do problema . . . 43
4.4.1 Plataforma PAO . . . 43
4.4.2 Guião LAV . . . 45
4.5 Sumário . . . 46

5 Plataforma de Gestão de Atendimentos 47
5.1 Introdução . . . 47
5.2 Atores e casos de utilização . . . 48
5.3 Escolha de tecnologias . . . 50
5.3.1 *Layouts* responsivos . . . 50
5.3.2 *Front-end*: Páginas dinâmicas . . . 52
5.3.3 *Back-end* . . . 54
5.3.4 Tecnologias selecionadas . . . 55
5.4 Arquitetura do sistema . . . 56
5.4.1 Considerações sobre segurança . . . 58
5.4.2 Módulos . . . 59
5.5 Administração . . . 59
5.5.1 Gestão de formulários . . . 59
5.5.2 Monitorização de atividade . . . 66
5.5.3 Lista PAR . . . 67
5.6 Atendimentos . . . 67
5.6.1 Pesquisa dos processos mais prováveis com base num atendimento . . . 69
5.6.2 Relação com atendimentos anteriores . . . 72
5.7 Processos de apoio . . . 74
5.7.1 Ligação entre atendimentos e processos de apoio . . . 75
5.7.2 Análise de risco . . . 75
5.8 Estatística . . . 75
5.9 Área pessoal do utilizador . . . 76
5.10 Sumário . . . 77

6 Validação 79
6.1 Metodologia . . . 79
6.2 Métricas de avaliação . . . 80
6.2.1 Métricas quantitativas . . . 80
6.2.2 Métricas subjetivas . . . 80
6.3 Análise dos resultados . . . 80
6.3.1 Características dos participantes . . . 81
6.3.2 Análise quantitativa . . . 81
6.3.3 Análise qualitativa . . . 83
6.4 Sumário . . . 85

7 Conclusões e trabalho futuro 87

Referências 91

A Plano de Testes de Validação 99
A.1 Gestão de formulários . . . 99
A.1.1 Cenários . . . 99
A.2 Atendimentos e processos . . . 102

A.3 Área do utilizador 104
A.3.1 Cenários 105

B Questionários de validação qualitativa e respostas dos utilizadores 107
B.1 Questionários 107
B.1.1 Gestão de formulários 107
B.1.2 Áreas de atendimentos e processos 108
B.1.3 Área pessoal do utilizador 110
B.2 Respostas 111
B.2.1 Área de administração de formulários e campos de formulários 111
B.2.2 Áreas de atendimentos e processos 113
B.2.3 Área pessoal do utilizador 115

Lista de Figuras

2.1 *Cloud computing* pelo NIST [AKV15] 6
2.2 Utilização da técnica replicação para fins de escalabilidade [Oraa] 13
2.3 Método de autenticação gráfica proposto em [WWSB06] 15
2.4 Método de autenticação com o sistema Clef 17
2.5 Processo abstrato de autorização com Auth 2.0 [SBCQ12] 19

3.1 *Timeline* das tecnologias de desenvolvimento *web* [Eri16] 26
3.2 Categorias de NoSQL [Cha13] 32
3.3 Problemas na edição de formulários Google 34
3.4 Esquema de tabelas, proposto em [MMNL14], para os metadados de formulários dinâmicos 35

4.1 Vista de processo na Plataforma de Processo de Apoio Online 42
4.2 Script Red Monitor, utilizado nos atendimentos da LAV 42

5.1 Diagrama de casos de utilização 49
5.2 Padrão *Model-View-View-Model* [Mic17] 53
5.3 Arquitetura geral do sistema 56
5.4 Modelo de dados relativo à gestão de formulários 60
5.5 Interface de edição da especificação de campos de formulário em YAML 61
5.6 Interface gráfica de edição da especificação de campos de formulário 62
5.7 Processo de criação de um campo de formulário 63
5.8 Vista do separador de Dados Gerais de um formulário, com histórico de edições . 64
5.9 Fluxo do processo da edição das opções e das secções de um formulário 65
5.10 Exemplo de definição de uma regra de visibilidade de um campo num formulário 66
5.11 Listagem da atividade dos utilizadores 67
5.12 Vista de um novo atendimento 68
5.13 Listagem das respostas de atendimentos anteriores de um dado processo de apoio 73
5.14 Vista do processo de apoio de um utente 74
5.15 *Dashboard* da área pessoal do utilizador 76
5.16 Edição de eventos de calendário da área pessoal do utilizador 77

6.1 Resultados quantitativos relativos à área de gestão de formulários e de campos de formulário 82
6.2 Resultados quantitativos relativos à área de gestão de formulários e de campos de formulário 82
6.3 Resultados quantitativos relativos à área pessoal do utilizador 83
6.4 Resultados qualitativos relativamente ao grau de facilidade de utilização e satisfação dos utilizadores na área de gestão de formulários e campos de formulário . . 84

6.5 Resultados qualitativos relativamente ao grau de facilidade de utilização e satisfação dos utilizadores na área de atendimentos e processos 84
6.6 Resultados qualitativos relativamente ao grau de facilidade de utilização e satisfação dos utilizadores na área pessoal do utilizador 85

Lista de Tabelas

5.1 Atores do sistema 48
5.2 Comparação entre soluções de pesquisa *full-text* 72

Abreviaturas e Símbolos

AES	*Advanced Encryption Standard*
AJAX	Asynchronous JavaScript And XML
APAV	Associação Portuguesa de Apoio à Vítima
API	*Application Programming Interface*
BD	Base de dados
CAPTCHA	*Completely Automated Public Turing test to tell Computers and Humans Apart*
CNPD	Comissão Nacional de Proteção de Dados
CSS	*Cascading Style Sheets*
CSV	*Comma-Separated Values*
DAC	*Discretionary Access Control*
DBMS	*Database Management System*
DES	*Data Encryption Standard*
DOM	*Document Object Model*
GAV	Gabinetes de Apoio à Vítima
HTML	*HyperText Markup Language*
HTTP	Hypertext Transfer Protocol
JSON	*JavaScript Object Notation*
LAV	Linha de Apoio à Vítima
MAC	*Mandatory Access Control*
NASA	*National Aeronautics and Space Administration*
ORM	*Object-relational mapping*
OTP	*One Time Password*
OWASP	*Open Web Application Security Project*
PAO	Processo de Apoio Online
PAR	Processos de Acesso Restrito
PIN	*Personal Identification Number*
PLAGA	Plataforma de Gestão de Atendimentos
RDBM	*Relational Database Management System*
REST	*Representational State Transfer*
RGPD	Regulamento Geral de Proteção de Dados
RSA	*Rivest-Shamir-Adleman*
SMS	*Short Message Service*
SPA	*Single Page Application*
SQL	*Structured Query Language*
SSH	*Secure Shell Protocol*
SSL	*Secure Sockets Layer*
TAV	Técnico de Apoio à Vítima
UI	*User Interface*
UO	Unidade Orgânica

URL *Uniform Resource Locator*
USB *Universal Serial Bus*
XML *eXtensible Markup Language*

Capítulo 1

Introdução

1.1 Contexto

A utilização de Sistemas de Informação (SI) numa organização altera significativamente a sua produtividade, uma vez que permite a automatização e uniformização de processos e a centralização de dados, levando à redução de custos e maior agilidade do negócio. No entanto, em muitos casos, os SI são pouco ágeis e pouco versáteis, e como tal, implementar uma nova funcionalidade ou alterar o modo de funcionamento para melhor adaptar o sistema às necessidades reais da organização, e aos seus processos de negócio, são tarefas complexas [BVM13].

A Associação Portuguesa de Apoio à Vítima (APAV) é uma instituição portuguesa de solidariedade social que presta apoio a vítimas de crime. De entre os serviços que oferece constam a Linha de Apoio à Vítima, onde o apoio é prestado através da via telefónica, e a rede nacional de Gabinetes de Apoio à Vítima, constituída por vários gabinetes localizados em diferentes pontos do país, onde o apoio é prestado presencialmente [Ass]. Segundo a informação que consta no relatório anual de estatísticas da APAV, em 2016 a instituição trabalhou com 12.450 processos, apoiou 9.347 vítimas de crime e realizou 35.411 atendimentos, representando este número um aumento de 8.1% dos atendimentos entre 2014 e 2016 [Ass17a].

Atualmente, a APAV utiliza dois sistemas que suportam a recolha e armazenamento dos dados relativos ao processo de apoio: a plataforma de Processos de Apoio Online, que consiste numa aplicação web onde é guardada a informação relativa aos processos das vítimas, e o Script RedMonitor, um programa instalado localmente nos computadores da sala de atendimento da Linha de Apoio e que contém formulários para registo dos dados das chamadas telefónicas.

1.2 Motivação

Considerando a importância do trabalho da APAV e a natureza sensível dos dados com que a associação trabalha, é fundamental que exista um Sistema de Informação de suporte confiável,

uniformizado, seguro, acessível, eficiente e prático de utilizar. Neste contexto, o impacto de problemas como a perda de informação ou acessos não autorizados aos dados é alto, representando risco para as vítimas e também consequências legais para a instituição [Ass14].

Como a APAV procura melhorar continuamente os seus serviços e o apoio que presta às vítimas de crime, é importante que o sistema informático de suporte seja flexível o suficiente para permitir eventuais alterações, nomeadamente a nível da gestão (criação, edição e consulta) dos formulários de atendimento. O objetivo é diminuir ao máximo a necessidade de recurso a programadores e serviços externos para realizar a manutenção que a APAV normalmente pratica na sua gestão de atendimentos.

A produção dos relatórios estatísticos anuais é uma componente fundamental do trabalho da associação, pois reflete para o exterior a atividade desenvolvida, à qual recorrem investigadores, meios de comunicação social, políticos, entre outros [Ass16b]. Assim, é do interesse da APAV que exista um sistema de gestão de dados que permita a exportação da informação segundo diferentes dimensões relevantes para análise, de forma a facilitar o trabalho dos técnicos de estatística. É importante que o sistema esteja devidamente expurgado de inconsistências, permitindo evitar ao máximo a limpeza manual da base de dados, como é feita atualmente [Ass16b].

1.3 Projeto e objetivos

O projeto apresentado nesta dissertação consiste no desenvolvimento de um primeiro protótipo funcional daquele que será o novo sistema informático de suporte ao atendimento da APAV. Este sistema consiste numa plataforma *web*, e deve permitir, de uma forma eficaz, a realização dos atendimentos, a gestão dos processos de apoio, a monitorização da atividade dos utilizadores e a administração dos formulários de atendimento às vítimas.

Assim, com este projeto e dissertação pretendem-se cumprir os seguintes objetivos:

- Estudar aspetos relacionados com o desenvolvimento e segurança das aplicações *web*, e construção de formulários com interface dinâmica;
- Planear uma plataforma de apoio ao atendimento que permita resolver os problemas evidenciados pela APAV nesse processo de atendimento;
- Desenvolver um protótipo funcional com foco nas áreas de administração de formulários e realização de atendimentos;
- Efetuar uma primeira validação junto dos futuros utilizadores para verificar se o sistema desenvolvido vai, de facto, ao encontro das suas necessidades.

Espera-se que a plataforma desenvolvida e apresentada nesta dissertação seja o primeiro protótipo de um sistema que venha a permitir aos técnicos e gestores da APAV melhorar de forma considerável a gestão dos atendimentos a vários níveis, nomeadamente a nível do aumento de eficiência do processo, flexibilidade de gestão, possibilidades de acesso e controlo de segurança.

1.4 Estrutura do documento

Para além da Introdução, este documento é constituído por mais seis capítulos e dois apêndices.

No capítulo 2, são abordados aspetos sobre o alojamento de aplicações *web* na *cloud*, e questões de segurança e confidencialidade de dados.

No capítulo 3, são apresentadas as formas e tecnologias atuais de desenvolvimento de aplicações *web*, bem como o estado da arte relativo à gestão de formulários dinâmicos pelo utilizador.

No capítulo 4, é apresentado, de forma detalhada, o problema a que se procurou dar resposta.

No capítulo 5, são apresentados os detalhes de implementação da Plataforma de Gestão de Atendimentos, o novo sistema criado para gerir o processo de atendimento à vítima na APAV.

No capítulo 6, são apresentados os testes de usabilidade conduzidos junto dos futuros utilizadores da PLAGA, de forma a poder validar o trabalho desenvolvido.

As conclusões finais e o trabalho futuro a desenvolver estão apresentados no capítulo 7.

Os apêndices A e B apresentam, respetivamente, o plano de validação efetuado e os resultados completos dos questionários de validação.

Capítulo 2

Alojamento e segurança das aplicações *web*

Neste capítulo são abordados aspetos relacionados com o alojamento de aplicações *web* na *cloud*, nomeadamente os tipos de serviços disponíveis. Considerando o objetivo final do projeto, são também apresentadas questões de segurança e confidencialidade de dados, como as vulnerabilidades possíveis e formas de prevenção, controlo de acessos e métodos de encriptação de dados.

2.1 Ambientes *Cloud*

A computação na *cloud* é uma variante da arquitetura cliente-servidor, onde milhares de clientes usam a mesma infraestrutura a grande escala [CKS+11]. Consiste num modelo para publicar, de forma partilhada, recursos e serviços poderosos com baixo custo e configuráveis conforme as necessidades, requerendo um esforço mínimo de gestão por parte dos clientes [MG11].

A definição de *cloud computing*, publicada pelo National Institute of Standards and Technology, possui uma alta taxa de aceitação na comunidade [FSG+14]. Segundo esta definição [MG11], os serviços *cloud* podem ser classificados como *Software as a Service* (SaaS), *Infrastructure as a Service* (IaaS) e *Platform as a Service* (PaaS), e disponibilizados em ambientes privados, comunitários, públicos ou híbridos. Esta tecnologia deve apresentar cinco características fundamentais (Figura 2.1):

- *On-demand self-service* — Os clientes podem gerir os serviços da *cloud* através de uma interface web, sem necessidade de interação pessoal com o fornecedor dos serviços;
- *Broad network access* — Os serviços e os dados presentes na *cloud* devem ser acessíveis através de protocolos e mecanismos *standard*;
- *Resource pooling* — Os recursos físicos e virtuais são dinamicamente atribuídos aos utilizadores de acordo com as suas necessidades. Normalmente, os clientes não controlam a

localização específica desses recursos, embora possam especificar uma localização de alto nível como o país ou região;

- *Rapid elasticity* — Para que o utilizador tenha uma sensação de disponibilidade de recursos ilimitada, estes são libertados ou aprovisionados de forma rápida e "elástica";
- *Measured service* — A utilização dos recursos pode ser monitorizada, controlada e transmitida, tanto ao fornecedor do serviço como ao utilizador.

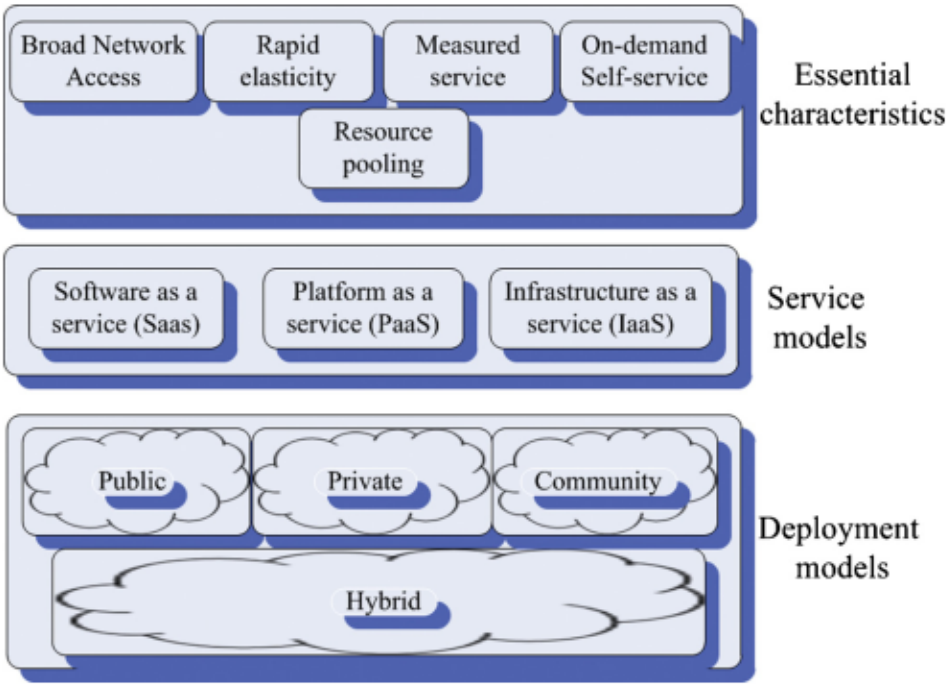


Figura 2.1: *Cloud computing* pelo NIST [AKV15]

A computação na *cloud* é atualmente considerada um dos paradigmas mais dominantes da área de Tecnologias de Informação (TI) [AYKM14, AKV15, CKS+11, KKMR16].

2.2 Segurança e confidencialidade de dados

As bases de dados pessoais e o tratamento de dados associado assumem um papel fundamental na boa gestão de qualquer empresa/entidade pública, com particular relevância no domínio do relacionamento com os seus clientes/utentes dos serviços [Ass14].

Em Portugal, o tratamento de dados pessoais está sujeito a determinadas regras de proteção da privacidade, definidas pela Comissão Nacional de Proteção de Dados (CNPD) [1]. A CNPD é a autoridade nacional de controlo de dados pessoais, e é responsável por controlar e fiscalizar o cumprimento da regulamentação relativa à proteção de dados.

[1] https://www.cnpd.pt/

Os dados recolhidos por uma entidade podem ser classificados como dados sensíveis ou dados não sensíveis. Dados sensíveis são, por exemplo, dados referentes à vida privada, à saúde ou vida sexual, e o seu tratamento é apenas autorizado em casos excecionais [Ass14, Per17].

Nos casos em que é efetuada recolha de dados sensíveis, a entidade fica incumbida de respeitar certas obrigações, entre as quais pôr em prática medidas técnicas e organizacionais para proteger os dados contra a destruição acidental ou ilícita, a perda acidental, a alteração, a difusão ou o acesso não autorizados e qualquer forma de tratamento ilícito [Ass14].

Em 2016 foi publicado o Novo Regulamento Geral de Proteção de Dados (RGPD), que será aplicável a partir de Maio de 2018 a todas as entidades públicas ou privadas que realizem operações com dados de pessoas singulares. Destacam-se algumas regras importantes deste regulamento a ter em conta, que podem estar relacionadas com os sistemas informáticos utilizados pelas entidades de suporte ao tratamento de dados [Per17]:

- As empresas devem salvaguardar o ciclo de vida dos dados, de forma a poderem localizar e eliminar dados dos clientes/utentes quando necessário, caso estes o requeiram;
- No caso de ocorrência de violação dos dados pessoais, ou *data breaches*, estas devem ser comunicadas em 72h à CNPD;
- Os sistemas informáticos devem possuir mecanismos de autenticação e rastreio dos utilizadores;
- Devem ser efetuados testes para deteção de vulnerabilidades e devem ser definidas políticas de *backup* e recuperação de dados;
- Devem existir mecanismos de bloqueio de dispositivos comprometidos.

No caso do não cumprimento destas regras, a entidade fica sujeita a penalizações monetárias severas.

2.3 Vulnerabilidades e ataques

A probabilidade de possíveis ataques para violação de dados aumenta em aplicações *web*, já que são sistemas acessíveis através da *internet*, potencialmente por qualquer pessoa. Assim, é importante conhecer quais as vulnerabilidades a que uma aplicação *web* poderá estar sujeita, e que mecanismos existem para evitar a exploração dessas vulnerabilidades.

Entende-se por vulnerabilidade uma fraqueza numa aplicação, que pode ser explorada por um atacante de modo a que este consiga acesso não autorizado aos dados da aplicação, consiga danificar ou comprometer de algum modo o seu conteúdo [Mica].

São inúmeras as formas possíveis de ataque a aplicações web, pelo que é necessário analisá-las, de forma a perceber exatamente que tipo de fraquezas poderão conter. Esta análise deve ser feita nos níveis de lógica (processo de interação entre a aplicação e os dados), dados (formas de acesso, formatos e locais de armazenamento, modo como os dados são combinados) e interface

(funcionalidades que podem permitir ataques). É importante ter também em conta restrições de acessos aos dados, como permissões de escrita e leitura, verificação do tipo esperado de dados e validação de dados de *input* [Mue15].

Aplicações alojadas na *cloud* poderão ficar sujeitas a ataques devido a vulnerabilidades existentes nesse serviço. Assim, no caso de ser possível alojar o projeto em servidores locais com implementação dos protocolos de segurança recomendados, essa opção é preferível [Mue15].

2.3.1 OWASP Top 10 2017

A Fundação OWASP (Open Web Application Security Project) [2] é uma organização internacional focada em melhorar a segurança das aplicações *web*, disponibilizando de forma gratuita várias ferramentas, documentos e fóruns sobre a área. O OWASP Top Ten Project é um dos projetos desenvolvidos, e nesse âmbito é publicada uma lista tri-anual das dez vulnerabilidades mais graves para as organizações. São apresentadas a seguir, de forma breve, as vulnerabilidades indicadas na listagem de 2017 [OWA17], a mais recente até à data, bem como o impacto que podem causar nas organizações. Para cada uma é ainda indicada a forma de prevenção.

1. *Injection* — Consiste na inserção de informação maliciosa que é executada pelo interpretador como parte de um comando ou *query*, permitindo que o atacante obtenha acesso não autorizado aos dados. O impacto deste ataque nas organizações é muito alto, uma vez que pode resultar em perda ou corrupção de dados, negação de acessos, perda de credibilidade da organização, ou mesmo permitir que o atacante obtenha permissões de administrador e controle completamente o sistema.

   Vulnerabilidades associadas A aplicação incorpora o *input* dos utilizadores diretamente nos comandos a serem executados pelo interpretador. As *queries* SQL, XPath e NoSQL são exemplos de locais que podem estar sujeitos a este ataque.

   Prevenção Toda a utilização de interpretadores deve ser feita de modo a separar os dados inseridos pelo cliente e os comandos/*queries* a serem executados, utilizando uma uma API que forneça uma interface parametrizada, ou seja, que faça a verificação do *input* do cliente e o trate de forma literal, e não como parte do comando/*query* a ser executado.

2. *Broken authentication and session management* — Consiste na exploração de falhas de implementação das funções de autenticação ou gestão de sessão, de forma a personificar os utilizadores e violar a privacidade. O atacante passa a poder efetuar qualquer ação permitida ao utilizador legítimo, sendo que, no caso de se tratar de uma conta com altos privilégios, pode comprometer severamente a informação.

[2] https://www.owasp.org/

Vulnerabilidades associadas As credenciais dos utilizadores não estão devidamente encriptadas e protegidas; a sessão do utilizador é exposta no URL; a sessão não tem um *timeout* associado ou utiliza *tokens* que não são refrescados regularmente, mantendo-se sempre válida; as credenciais dos utilizadores são enviadas através de ligações não seguras.

Prevenção Deve ser utilizado um sistema de autenticação e gestão de sessão que tenha sido altamente testado e cumpra com os requisitos recomendados para a proteção deste tipo de ataque, como por exemplo a biblioteca ESAPI (OWASP Enterprise Security API), que providencia uma série de interfaces e controlos para a implementação de uma gestão segura de sessões e mecanismo de autenticação de utilizadores.

3. *Cross-site scripting (XSS)* — O atacante executa *scripts* no *browser* da vítima, de forma a poder obter a sua sessão, alterar a página ou redirecionar o utilizador para páginas maliciosas.

   Vulnerabilidades associadas Não é feita uma validação do *input* do cliente no lado do servidor, nem é efetuado o *escaping* [3] desses dados antes de os incluir na página, permitindo que sejam tratados como conteúdo ativo; os dados maliciosos são guardados no servidor sem validação, podendo depois ser apresentados aos utilizadores.

   Prevenção Nunca incluir dados provenientes do *input* do utilizador diretamente nos ficheiros HTML, CSS ou em *scripts*. Efetuar validações de comprimento, formato e caracteres utilizados, e recorrer a codificadores que efetuem o devido *escape* do *input*.

4. *Broken access control* — O sistema de controlo de acessos não está devidamente bem implementado, e permite que utilizadores sem os níveis de permissão adequados consigam aceder a determinados recursos. Neste caso, os atacantes são os próprios utilizadores, que alteram determinado parâmetro para acederem a recursos que, em teoria, não poderiam aceder, mas o sistema concede esse acesso.

   Vulnerabilidades associadas Não é feita verificação de papéis/permissões dos utilizadores no acesso a todos os recursos da aplicação.

   Prevenção Todos os acessos diretos a recursos devem conter uma verificação de autorização de acesso. Devem ser utilizadas referências indiretas aos objetos por utilizador ou sessão, de forma a prevenir que os atacantes tentem utilizar esses valores para acesso aos recursos.

[3] *Data escaping* é uma técnica que consiste em reescrever determinado texto de forma a tornar esse texto irreconhecível para o interpretador.

5. *Security misconfiguration* — As configurações e definições de segurança mal implementadas, em qualquer nível da aplicação, permitem que o atacante tenha acesso a recursos não autorizados desprotegidos, podendo levar a que o sistema fique completamente comprometido sem que os administradores tenham conhecimento.

   Vulnerabilidades associadas Utilização de *software* desatualizado; funcionalidades, como portas, serviços, páginas, contas e privilégios, ativadas desnecessariamente; contas com *password* definida por defeito, sem ter sido alterada; utilização de *frameworks* sem a configuração segura correta.

   Prevenção Deve ser construída uma arquitetura de sistema que providencie uma separação segura e efetiva entre componentes. As configurações utilizadas no ambiente de desenvolvimento devem ser semelhantes às utilizadas em produção, mas com *passwords* diferentes.

6. *Sensitive data exposure* — Muitas aplicações não protegem devidamente os dados, o que pode levar o atacante a conseguir alterar ou roubar esses dados. Este problema é especialmente grave no caso dos dados mais sensíveis, como credenciais de autenticação ou dados de contas bancárias.

   Vulnerabilidades associadas Não é efetuada encriptação dos dados mais sensíveis, ou é efetuada com algoritmos fracos e facilmente quebráveis.

   Prevenção Devem ser utilizados algoritmos de encriptação *standard* considerados seguros para guardar os dados e para efetuar as comunicações. A função de *autocomplete* nos formulários deve ser desativada nos campos de dados mais sensíveis, e a *cache* das páginas que contêm esses formulários e dados deve também ser desativada.

7. *Insufficient attack protection* — Muitas aplicações não implementam mecanismos de deteção e resposta a ataques manuais e/ou automáticos, o que impede os donos das aplicações de conseguirem tomar medidas rápidas de defesa.

   Vulnerabilidades associadas A aplicação está exposta a possíveis ataques e não implementa mecanismos de deteção, resposta ou bloqueio automático a esses ataques.

   Prevenção Implementar um sistema de *logging* com notificações, implementar mecanismos de bloqueio automático de pedidos ou endereços IP em determinadas situações, monitorizar a atividade dos utilizadores.

8. *Cross site request forgery (CSRF)* — Este tipo de ataque força o *browser* de uma vítima, autenticada numa aplicação, a enviar pedidos HTTP forjados com os dados da sessão para o atacante. A partir daí, o atacante pode forçar pedidos à aplicação que, por não implementar a devida proteção, os executa por os considerar legítimos.

   Vulnerabilidades associadas Confiar apenas da identidade do utilizador autenticado para efetuar pedidos, sem requerer que o utilizador autorize uma ação específica.

   Prevenção Deve ser gerado um *token* imprevisível que seja, no mínimo, único por sessão, e seja incluído nos pedidos HTTP de forma a que a aplicação possa reconhecer os pedidos legítimos (CSRF *token*).

9. *Using components with known vulnerabilities* — O atacante explora vulnerabilidades existentes em componentes, conhecidos como *frameworks* ou bibliotecas. Essas vulnerabilidades podem ser identificadas de forma manual ou automática. Todas as aplicações que utilizem esses componentes ficam expostas ao ataque.

   Vulnerabilidades associadas Utilização de *frameworks* e componentes desatualizados, com vulnerabilidades.

   Prevenção Deve-se manter o *software* atualizado e certificar que as versões dos componentes utilizados não são vulneráveis. Existem *websites* que listam componentes com vulnerabilidades conhecidas e que podem ser utilizados para facilitar esta investigação, como *http://cve.mitre.org/* ou *https://nvd.nist.gov/home.cfm*.

10. *Unprotected APIs* — Frequentemente as aplicações recorrem a APIs externas, que podem conter vulnerabilidades. Os atacantes podem explorar as vulnerabilidades dessas APIs para conseguir atacar as aplicações. A ferramentas de teste de segurança são mais difíceis de utilizar neste caso, pois frequentemente as APIs não disponibilizam UI e utilizam estruturas de dados e protocolos complexos.

    Vulnerabilidades associadas Utilização de APIs com vulnerabilidades, sem verificações suficientemente fortes de segurança e controlo de acessos.

    Prevenção Utilizar comunicações seguras entre a aplicação e as APIs, assegurar um esquema forte de autenticação nas APIs, implementar mecanismos que impeçam a utilização indevida das APIs.

Na listagem precedente, OWASP Top 10 2013, são ainda indicadas as seguintes vulnerabilidades nas posições 4, 7 e 10, respetivamente:

1. *Insecure direct object references* — Acesso a dados não autorizados através de referências de objetos, como ficheiros, diretórios ou chaves utilizadas nas bases de dados, expostas pelo programador e acessíveis sem controlo.

   Vulnerabilidades associadas O acesso aos recursos é permitido direta ou indiretamente, sem se verificar se o utilizador possui as permissões necessárias.

   Prevenção Controlar os acessos às referências dos objetos e utilizar referências indiretas por sessão ou utilizador, de forma a impedir que os atacantes consigam acessos diretos aos recursos não autorizados. Por exemplo, ao apresentar ao utilizador uma listagem de opções provenientes da base de dados, deve-se utilizar um mapeamento entre os valores a selecionar e as chaves da BD correspondentes.

2. *Missing function level access control* — Consiste na execução de pedidos maliciosos, não controlados, forjados pelos atacantes, permitindo que estes obtenham acesso a funcionalidades sem a devida autorização.

   Vulnerabilidades associadas Verificações de autenticação e autorização em falta no lado do servidor; acesso, na interface, a funções não autorizadas; verificações, no lado do servidor, que se baseiam apenas na informação fornecida pelo utilizador.

   Prevenção A aplicação deve ter um módulo de autorização consistente, que seja fácil de analisar e que seja invocado por todas as funções de negócio. Devem ser verificadas as condições de acesso a uma determinada função, tendo em conta o estado atual de execução no *workflow*.

3. *Unvalidated redirects and forwards* — Se uma página efetuar redirecionamentos para outros *websites* sem a devida validação, os atacantes podem substituir o URL correto por um malicioso, por exemplo: *http://www.example.com/redirect.jsp?url=evil.com*. Os *sites* maliciosos podem tentar instalar *malware* ou utilizar técnicas de *phishing* [4] para obter dados privados dos utilizadores.

   Vulnerabilidades associadas Permitir redirecionamentos não validados.

   Prevenção Os redirecionamentos devem ser evitados. No entanto, caso sejam utilizados, o URL não deve incluir dados fornecidos pelo utilizador ou, caso inclua, deve ser feita validação no lado do servidor. Os parâmetros introduzidos pelos utilizadores devem ser mapeados e não utilizados diretamente.

[4] Técnicas de fraude *online*, na categoria de *social engineering*, utilizadas para tentar roubar informação pessoal dos utilizadores

2.4 Replicação da base de dados

> *"Replication is the process of copying and maintaining database objects in multiple databases that make up a distributed database system."* [Orab]

O processo de replicação é geralmente realizado de forma a que os dados de um servidor *master* sejam copiados para servidores *slaves* (réplicas). Esta técnica pode ser utilizada para aumentar tolerância a falhas [WPS+00], uma vez que existem cópias dos dados acessíveis a partir de diferentes locais (redundância). Se o servidor *master* falhar, os servidores *slaves* podem ser utilizados em substituição [Oraa]. É também possível recorrer à replicação para aumentar a *performance* em situações de escalabilidade, gerindo o tráfego de acessos ao distribuir operações de leitura pelas réplicas, e apenas utilizar o servidor *master* para operações de escrita, como ilustrado na Figura 2.2.

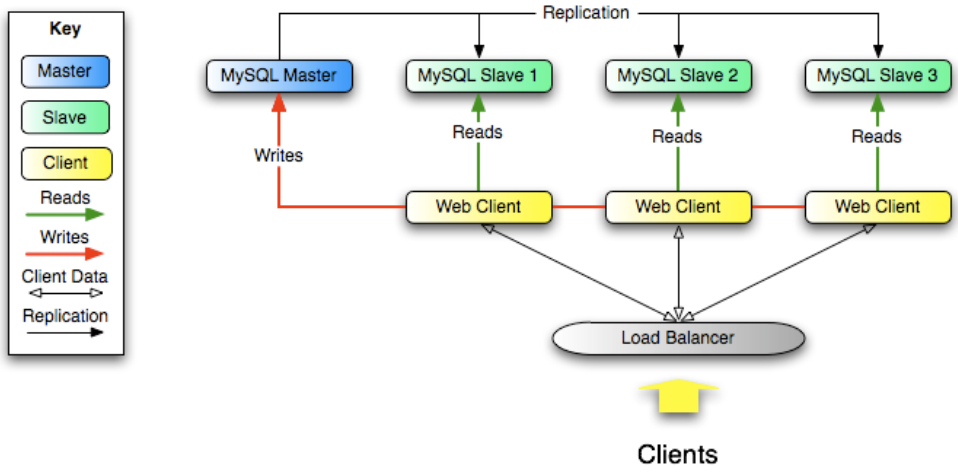


Figura 2.2: Utilização da técnica replicação para fins de escalabilidade [Oraa]

Esta técnica pode também ser utilizada para auxiliar o processo de *backup* dos dados, que é mais facilmente realizado se os dados do servidor estiverem estáveis (sem a ocorrência de alterações ativas no momento). Um cenário comum é replicar os dados do servidor *master* para um servidor *slave*, que é depois desconectado do sistema para ser realizado o *backup* [Oraa]. Os sistemas de gestão de base de dados (DBMS) fornecem mecanismos para efetuar replicação consoante os objetivos.

Existe também a possibilidade de haver mais do que um servidor *master*, de forma a que as operações de escrita possam ser efetuadas em todos esses servidores e não apenas num. Esta técnica é chamada replicação *multi-master* ou *master-master*. A vantagem deste processo é que as cópias dos dados contêm sempre a versão mais recente, mas existem maiores dificuldades de sincronização e desafios de consistência.

2.5 Controlo de acessos

Um sistema de controlo de acessos permite saber que utilizadores acedem ao sistema (autenticação), o que é que cada utilizador pode fazer no sistema (autorização) e quais são as ações efetivamente realizadas pelos utilizadores no sistema (imputabilidade, ou *audit logs*).

2.5.1 Autenticação

A autenticação é uma das características de segurança mais importantes num sistema, principalmente em sistemas *cloud* [AKC16]. O método de autenticação mais utilizado é através de *passwords* textuais, uma vez que é um método muito simples de implementar e utilizar, e apresenta um baixo custo associado. No entanto, é também um método vulnerável a diversos ataques, como ataques *brute-force*, *key-loggers*, *shoulder surfing* e *social engineering* [AKC16]. Para tentar resolver este problema surgiram vários métodos de autenticação alternativos, como a autenticação multi-fator, autenticação com recurso a entidades terceiras, *smart cards*, *scans* biométricos e *passwords* gráficas.

Smart Cards Vários autores propõem o recurso a *smart cards* para tornar a autenticação do utilizador mais segura [TLL12, HCC10, CKS+11], uma vez que estes cartões permitem a utilização de *passwords* complexas, que podem ser conjugadas com outros detalhes de identificação do utilizador [Mue15]. No entanto, são necessários leitores especiais para os cartões e para utilizar estes sistemas, e existe o risco de as pessoas os perderem ou de os deixarem acessíveis aos atacantes. Existem já teclados com leitores de cartões incorporados; em Portugal, há *software* disponível para a leitura do cartão de cidadão[5]. A Secura Key [6] é uma empresa que desenvolve esta tecnologia, permitindo muitos tipos diferentes de *smart cards*.

Alternativamente, é também possível utilizar *pens* USB para realizar a autenticação de forma semelhante: a *pen* contém as *passwords* ou *tokens* de acesso à aplicação, e o acesso é permitido ao conectar a *pen* ao computador. Uma das vantagens desta opção é o custo mais baixo em relação aos *smart cards*. No entanto, mantém-se o problema da possibilidade de perda da *pen* ou de que esta possa ser utilizada por atacantes [Mue15].

***Scans* biométricos** A autenticação por *scans* biométricos recorre a características biológicas dos utilizadores. São exemplos deste tipo de autenticação o reconhecimento da íris, reconhecimento facial, reconhecimento de impressão digital e reconhecimento por voz [Rou14]. Estes sistemas apresentam as desvantagens de necessitarem de dispositivos que efetuem a recolha dos dados biométricos, e de poderem efetuar um reconhecimento incorreto [AKC16, XYO05]. Por exemplo, o reconhecimento por voz apresenta dificuldades em ambientes com ruído, o reconhecimento facial é sensível às variações de luz e os leitores de impressões digitais podem ser enganados com impressões falsas [HDCP08].

[5] http://www.altronix.pt/produtos/teclados-cherry/teclado-cherry-g83-6644-cartao-de-cidadao-
[6] http://securakey.com/

***Passwords* gráficas** Baseando-se em estudos que demonstram que o ser humano consegue recordar consideravelmente melhor imagens do que texto [She67], as *passwords* gráficas apresentam uma alternativa interessante às textuais, sendo também de mais fácil utilização nos dispositivos móveis [SWKW13]. Neste método, o utilizador clica em imagens que tenha escolhido durante o processo de registo (técnica baseada em reconhecimento), ou reproduz algum registo gráfico que criou ou selecionou na fase de registo (técnica baseada em memória) [XYO05]. Um grande problema deste método é o ataque *shoulder surfing*[7].

Em [WWSB06] os autores desenvolvem um método de *passwords* gráficas que dificulta este ataque: o utilizador, para se autenticar, deve reconhecer objetos e clicar na forma geométrica gerada por esses objetos. Os autores sugerem utilizar muitos objetos para tornar o ecrã muito povoado e dificultar o reconhecimento, como ilustrado na Figura 2.3. Uma desvantagem deste processo é exigir muito tempo ao utilizador.

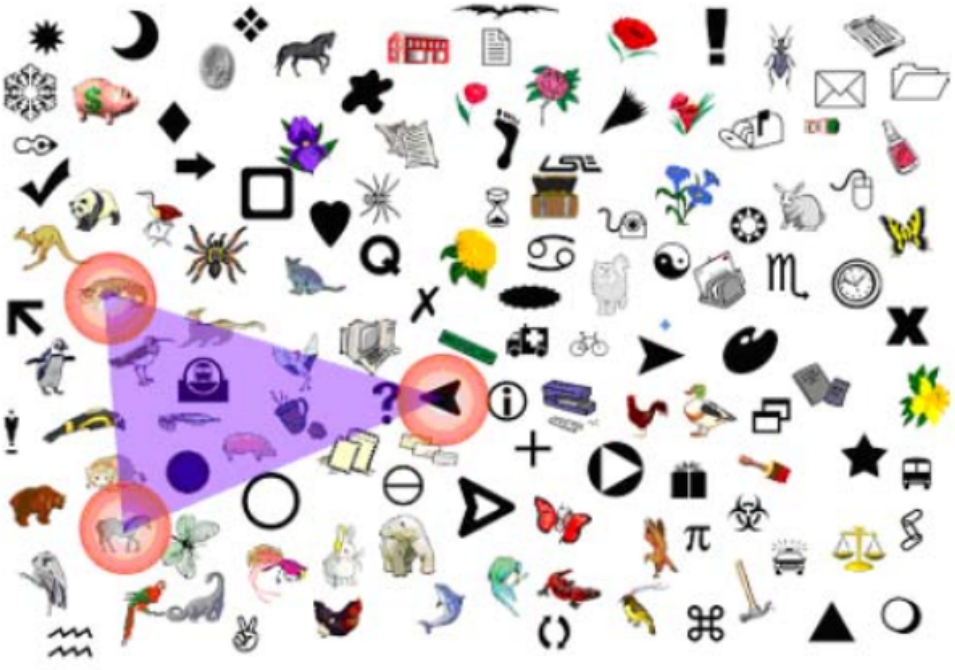


Figura 2.3: Método de autenticação gráfica proposto em [WWSB06]

Passfaces [8] é uma tecnologia em que o código de autenticação do utilizador consiste numa sequência de fotografias de rostos de pessoas, baseando-se no facto de o ser humano conseguir reconhecer muito mais facilmente rostos do que outro tipo de figuras. No entanto, em [DMR04] os autores verificam que os utilizadores tendem a escolher rostos de pessoas da mesma raça e a seguir determinados padrões que tornam as sequências previsíveis.

Use Your Illusion (UYI) é um mecanismo de autenticação em que a imagem que o utilizador escolhe no processo de registo é depois distorcida para ser reconhecida no processo de *login*. O

[7] Técnica utilizada para obter informação pessoal, como *passwords*, espreitando por cima do ombro da vítima
[8] http://www.passfaces.com/

algoritmo apresenta alguns pontos fracos, como a dependência do ponto de distorção ótimo da imagem, uma vez que a distorção de uma imagem com cores mais brilhantes e formas mais bem definidas preserva mais informação do que a distorção de uma imagem com cores menos contrastantes e formas mais difusas, e o facto de alguns utilizadores conseguirem reconhecer a imagem original a partir de uma imagem distorcida, mesmo sem terem sido expostos à original [HDCP08].

Em [SWKW13] são exploradas e analisadas, de forma detalhada, diferentes estratégias de autenticação gráfica em dispositivos móveis sob o ponto de vista da usabilidade, segurança, *design* e capacidade dos *smartphones*. Para essa análise, os autores fizeram implementações dos projetos TAPI [CR10], que utiliza imagens parciais, MIBA [RSWW13], que suporta *multi-touch* para permitir a seleção simultânea de diferentes pontos da imagem, Pass-Go [TA08], baseado no jogo chinês Go, e UYI, já apresentado anteriormente. Foi concluído que todos os mecanismos apresentam as suas vantagens e desvantagens.

Passphrases Apesar da existência de todas estas alternativas, uma vez que as *passwords* textuais são as mais utilizadas, deve-se encontrar métodos para aumentar a sua segurança na medida do possível. Um método interessante é a utilização de *passphrases* [Mue15]. Esta técnica consiste em utilizar palavras ou frases que, apesar de serem facilmente memorizáveis para o utilizador devido ao significado que apresentam, contêm letras, números e símbolos, o que torna a *password* mais complexa. É um exemplo a frase "*I luv fl0w3rs!*". Convém realçar que, para que a *password* seja realmente difícil de adivinhar, não seja relacionada com a aplicação, vida pessoal do utilizador ou local de trabalho.

Tokens Neste contexto, entende-se por *token* um código para ser utilizado apenas uma vez numa autenticação, designado por One-Time Password (OTP). Este método é muito utilizado com *smartphones*, em que é enviado o código por SMS ou é utilizada a câmara do dispositivo para fazer o reconhecimento de uma imagem, de modo a realizar a autenticação. Tem-se com exemplo deste tipo de implementação o Whatsapp [9], em que é enviada uma SMS com um PIN numérico (no caso de o utilizador querer iniciar sessão no *smartphone*) ou criado um *QRcode* para sincronização da conta com o computador. Outros exemplos são a Illiri API [10], em que é enviado um som para ser tocado no *smartphone* do utilizador para que este seja autenticado no computador; e o sistema Clef [11], que realiza a autenticação através da correspondência de um padrão em imagem (Figura 2.4).

Autenticação multi-fator A autenticação multi-fator consiste em conjugar dois ou mais métodos de autenticação, com o objetivo de tornar o processo de autenticação mais forte [LM16]. Cada método diferente adicionado corresponde a uma nova camada de autenticação, e portanto é mais uma barreira para um possível atacante. O conceito por detrás deste método é que a legitimidade de um indivíduo é verificada através de três componentes essenciais: 1. algo que ele sabe (por

[9] https://www.whatsapp.com
[10] https://www.programmableweb.com/api/illiri
[11] https://getclef.com/

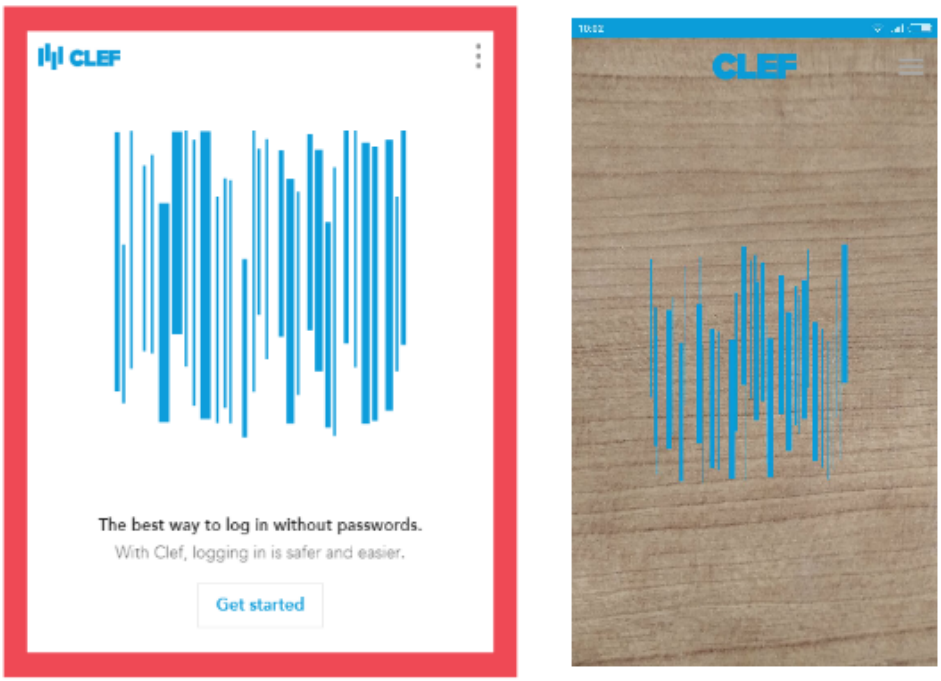


Figura 2.4: Método de autenticação com o sistema Clef

exemplo, *password* textual); 2. algo que ele tem (por exemplo, um *smart card*); 3. algo que ele é (através de *scan* biométrico) [LM16].

A variante mais utilizada é a autenticação em 2 fatores (2FA), que consiste na adição de um OTP ao primeiro método de autenticação. Por exemplo, num sistema bancário, após o utilizador ter efetuado autenticação com uma *password* textual, pode ser realizada novamente a autenticação com um *token* para efetuar uma transação monetária.

2.5.2 Autorização

Os mecanismos de autorização representam também um papel fundamental na proteção dos dados de um sistema. Estes mecanismos podem ser classificados nos seguintes tipos [AYKM14]:

- *Mandatory Access Control* (MAC) — Os objetos são classificados com um determinado nível de segurança, consoante o tipo de informação neles contida. O controlo de acessos a estes objetos é feito por uma entidade central (por exemplo, o sistema operativo), e não pode ser alterado pelos utilizadores;

- *Discretionary Access Control* (DAC) — Neste modelo, as entidades responsáveis pelos objetos podem conceder, ou restringir acesso aos mesmos, com base na identificação dos utilizadores, ou pertença dos utilizadores a grupos. É o modelo mais utilizado nos sistemas operativos comerciais, como UNIX e Windows, uma vez que apresenta mais facilidade de utilização e flexibilidade, comparativamente ao modelo MAC [AYKM14, Yer13];

- *Role Based Access Control* (RBAC) — Um utilizador tem um ou mais papéis (*roles*), e o acesso aos recursos é feito com base nesses papéis;
- *Task-Role Based Access Control* (T-RBAC) — É uma variante do modelo RBAC, em que, para além de os utilizadores terem papéis, são atribuídas permissões nesses papéis a tarefas que os utilizadores podem efetuar. Este modelo apresenta grande flexibilidade, mas é necessário ter o cuidado de impedir a atribuição de um mesmo utilizador a papéis mutuamente exclusivos, e definir bem que permissões estão associadas a cada papel, para não haver violação da política de acessos. Em sistemas de grande complexidade este modelo pode tornar-se consideravelmente difícil de utilizar;
- *Attribute Based Access Control* (ABAC) — Baseia-se num conjunto de atributos associados ao autor do pedido, ou ao recurso a ser acedido. Um atributo pode ser, por exemplo, o local ou a hora de início de trabalho de um utilizador.

Protocolo OAuth 2.0 O OAuth 2.0 é um protocolo *standard* de autorização para a concessão de acesso limitado aos dados das contas dos utilizadores às aplicações, através de um serviço HTTP. A autenticação do utilizador é delegada ao serviço onde a conta do utilizador é hospedada, e a autorização para o acesso à informação é concedido às aplicações [SBCQ12]. Para este protocolo são definidos quatro papéis:

- *Resource Owner* — Refere-se à entidade que autoriza a aplicação a aceder à sua conta, indicando o tipo de acesso, como escrita e/ou leitura de dados, e o tipo de informação que pode ser acedida, como por exemplo *email*, nome, morada, etc;
- *Client* — Refere-se à aplicação que requer o acesso à conta do utilizador;
- *Resource Server* — Servidor onde estão armazenados os dados das contas dos utilizadores e onde é exposta uma API para o acesso a esses dados;
- *Authorization Server* — É o servidor onde é validada a identidade do utilizador, e que apresenta uma *interface* onde este aprova ou recusa o pedido de autorização do acesso aos seus dados.

A Figura 2.5 ilustra, de forma abstrata, o processo de autorização utilizando o protocolo OAuth 2.0, onde os quatro papéis descritos anteriormente interagem. O cliente começa por pedir autorização de acesso aos recursos ao utilizador, sendo que este pedido pode ser direto, ou através do intermediário do servidor de autorização. O cliente recebe depois a credencial representativa da autorização dada pelo utilizador (*authorization grant*), utilizando-a para se autenticar no servidor de autenticação e obter o *token* de acesso. Utilizando esse *token*, o cliente requer acesso aos recursos protegidos, ao servidor onde estes estão alojados; caso o *token* seja válido, o acesso é permitido. Pode ainda ser utilizado, opcionalmente, um *refresh token* para obter um novo *token* de acesso quando o atual expira ou se torna inválido.

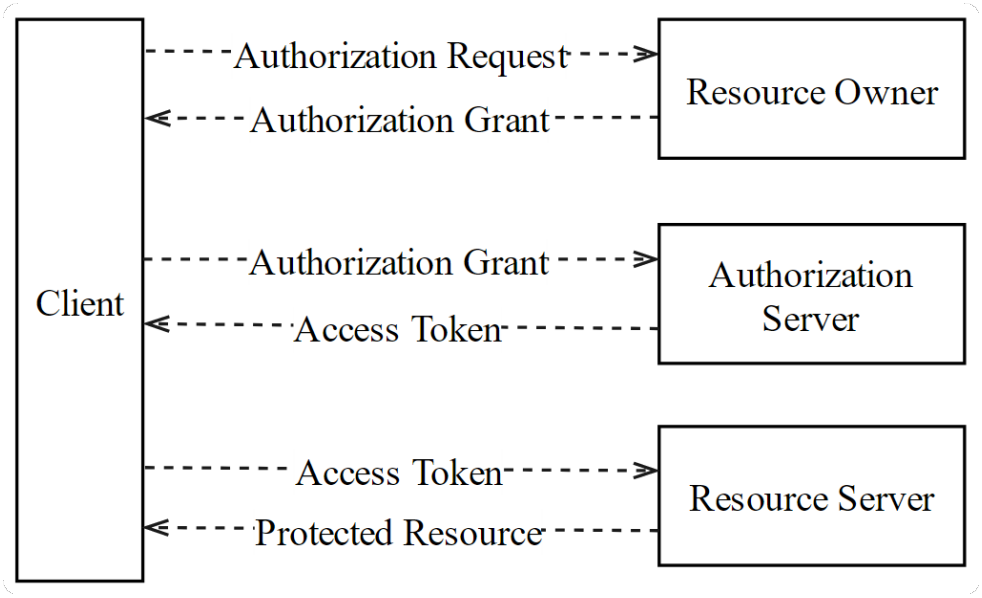


Figura 2.5: Processo abstrato de autorização com Auth 2.0 [SBCQ12]

De acordo com a especificação, por forma a um cliente receber um *token* de acesso, pode autenticar-se no servidor com diferentes tipos de credenciais (*authorization grant*), dependendo do caso de uso, como indicado de seguida [SBCQ12, LM16]:

- *Authorization Code* — O código de acesso é obtido através de um servidor intermediário de autenticação. O utilizador é redirecionado para um servidor de autenticação, que depois o volta a redirecionar para a aplicação cliente. Este processo permite que as credenciais do utilizador não sejam partilhadas com o cliente, uma vez que são substituídas pelo *token* como credencial de acesso. Este tipo de autenticação é normalmente utilizado nas aplicações que utilizam recursos provenientes de um servidor. Por exemplo, é possível integrar serviços do Slack, Facebook ou Twitter com este método;

- *Implicit* — Este método está otimizado para ser utilizado quando o cliente é um *browser* e utiliza linguagem de *scripting*, como Javascript. Neste caso, é fornecido diretamente ao cliente um *token* de acesso, após a autorização do *resource owner*. O *token* fica exposto do lado do cliente, o que pode causar problemas de segurança;

- *Resource Owner Password Credentials* — São utilizadas as credenciais do utilizador para obter um *token* de acesso. Por essa razão, este método só deve ser utilizado quando existe um grau de confiança elevado do utilizador em relação à aplicação cliente. Este método, apesar de necessitar das credenciais do utilizador num primeiro pedido, evita que a aplicação cliente tenha de armazenar essas credenciais, pois estas são trocadas pelo *token* para os pedidos seguintes;

- *Client Credentials* — As credenciais do cliente são tipicamente utilizadas quando a aplicação cliente, e não o utilizador final, é o *resource owner*.

2.5.3 Imputabilidade (*audit logs*)

Os *audit logs* são especialmente importantes para saber que contas estão associadas a determinadas ações, de modo a perceber se o sistema está a ser corretamente utilizado, reconstruir cronologicamente as ações efetuadas antes ou depois de determinado evento, detetar intrusos no sistema através de, por exemplo, tentativas de *login* falhadas, tentativas de *login* fora do horário normal, ou de locais improváveis, ou acessos a determinados ficheiros, e detetar eventuais problemas no sistema que precisem de ser resolvidos [Spa06].

2.6 Algoritmos de *hashing* e encriptação de dados

De forma a proteger determinada informação, como *passwords*, recorre-se normalmente a algoritmos não reversíveis (funções de *hash*), que são algoritmos que codificam determinado conteúdo para um texto de tamanho fixo. Uma boa função de *hash* não deve produzir o mesmo resultado a partir de duas fontes diferentes, mesmo que próximas (colisão), devendo para isso implementar um bom algoritmo de dispersão. Como estes algoritmos são de muito difícil reversão, são geralmente usados para codificar *passwords*, uma vez que apenas é necessário comparar o *hash* da *password* introduzida pelo utilizador, na fase de autenticação, com o *hash* da *password* que foi registada no processo de registo, para verificar se a autenticação é válida. De forma semelhante, são também utilizados para verificar a integridade de ficheiros. São exemplos destes algoritmos o MD5 e SHA.

Para a encriptação de dados recorre-se a algoritmos reversíveis, que podem ser utilizados para diminuir a probabilidade de o conteúdo protegido ser acedido e lido por atacantes.

> "*Encryption is a method for a user to securely share data over an insecure network or storage site.*" [BSW10]

> "*Cryptography deals with the actual securing of digital data. It refers to the design of mechanisms based on mathematical algorithms that provide fundamental information security services.*" [MS13]

É apresentada a seguir uma breve comparação entre alguns algoritmos reversíveis de encriptação:

- *Data Encryption Standard* (DES) — Este algoritmo foi proposto nos anos 70 como método *standard* para a proteção de dados. Encripta blocos de 64 bits com chave de 56 bits (+ 8 bits de paridade), e é vulnerável a ataques *brute-force* [MS13].

- *Advanced Encryption Standard* (AES) — Este algoritmo sucedeu ao DES como método *standard*. Encripta blocos de 128 bits e utiliza chaves privadas de 128, 192 ou 256 bits. É muito rápido, eficiente e, até ao momento, considerado muito pouco vulnerável [MS13, RSKM13].

- *Rivest-Shamir-Adleman* (RSA) — Surgiu em 1978, é baseado em aritmética modular e utiliza chaves feitas com números primos muito altos ($> 10^{100}$), sendo bastante pesado. Ao contrário dos dois algoritmos anteriores, que são sistemas de chave privada (simétricos), o RSA é um sistema de chave pública (assimétrico), e portanto pode ser utilizado para gerar certificados digitais (por exemplo, utilizado no protocolo SSH[12] [BS01]).

2.6.1 Armazenamento de *passwords*

Não basta apenas guardar as *passwords* com *hashing*, pois, desse modo, são ainda muito vulneráveis a vários ataques, sendo os mais comuns [Res17, Def16]:

- *Brute Force* e *Dictionary Attack* — É feito o *hashing* de diferentes combinações de caracteres, correspondendo a possíveis *passwords*, de modo a adivinhar qual a combinação correta. No caso do ataque *brute force*, são executadas todas as possíveis combinações de caracteres até um determinado comprimento; o *dictionary attack* utiliza um ficheiro que contém palavras, frases e *passwords* comuns a testar (dicionário). De forma a tornar o processo mais eficiente, o conteúdo do dicionário é também modificado (por exemplo, a *password* "123" pode ser alterada para "a23" e é testada essa hipótese);

- *Lookup Tables* e *Rainbow Tables* — As *passwords* e os respetivos *hashes* são pré-processados a partir de um dicionário de *passwords* e guardados numa tabela de *lookup*. o que permite que, em muito pouco tempo, possam ser testados centenas de *hashes*. As *rainbow tables* são mais lentas do que as *lookup*, mas permitem o armazenamento de um maior número de resultados;

- *Reverse Lookup Tables* — Neste caso, os *hashes* são agrupados por utilizadores que utilizam a mesma *password*, logo, que têm *hashes* iguais associados. É comum vários utilizadores definirem a mesma *password*, o que faz com que esta técnica seja eficaz.

Não há forma de evitar completamente os ataques *dictionary* ou *brute force*, mas é possível torná-los menos eficientes, por exemplo, através de técnicas como bloqueio temporário da conta após algumas tentativas de *login* seguidas, ou da utilização de um CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart). O CAPTCHA é uma ferramenta que permite saber se determinado pedido ou ação é realizado com recurso a mecanismos automáticos ou por um humano, através de um teste que, em teoria, apenas humanos conseguem passar. Um exemplo muito utilizado é o reCAPTCHA [13] da Google.

A forma de prevenir os ataques com *lookup tables* e *rainbow tables*, que apenas funcionam bem porque todos os *hashes* das *passwords* são produzidos exatamente da mesma forma, é tornar cada *hash* aleatório, adicionando, a cada *password* antes de correr a função de *hashing*, um texto aleatório (*salt*). Desta forma serão produzidos resultados completamente diferentes, mesmo que a *password* original seja a mesma. Este método previne o *reverse lookup* e obriga o atacante a gerar

[12] *Secure Shell*, protocolo de encriptação para a implementação de ligações seguras através da rede.
[13] https://www.google.com/recaptcha/

uma tabela para cada conta, o que torna estas formas de ataque impraticáveis. De realçar que, para que esta técnica funcione, o *salt* não pode ser reutilizado nem deve ter um comprimento pequeno, de modo a que não possam ser geradas *lookup tables* para todos os *salts* possíveis [Def16].

2.7 Sumário

A computação na *cloud* é atualmente considerada um dos paradigmas mais dominantes da área de Tecnologias de Informação (TI), e permite que os utilizadores tenham acesso a serviços de alta qualidade com baixo custo. Estes serviços podem ser classificados como *Software as a Service* (SaaS), *Infrastructure as a Service* (IaaS) e *Platform as a Service* (PaaS), e disponibilizados em ambientes privados, comunitários, públicos ou híbridos. Devido às suas vantagens, muitas organizações têm vindo a migrar os seus projetos para a *cloud*.

As bases de dados pessoais e o tratamento de dados associado assumem um papel fundamental na boa gestão de qualquer empresa/entidade pública. Em Portugal, o tratamento de dados pessoais está sujeito a regras de proteção da privacidade definidas pela Comissão Nacional de Proteção de Dados (CNPD). É fundamental que uma aplicação *web* implemente mecanismos de segurança que evitem a exploração de vulnerabilidades por parte dos atacantes, de forma a impedir que estes consigam acesso não autorizado aos dados da aplicação, consigam danificar ou comprometer de algum modo o seu conteúdo. Existe muita informação disponível acerca das vulnerabilidades possíveis de uma aplicação *web*, como devem ser evitadas e como é que as aplicações devem ser testadas para diminuir a probabilidade da existência destas vulnerabilidades. A Fundação OWASP é um exemplo de uma organização focada em melhorar a segurança do *software*, disponibilizando de forma gratuita várias ferramentas, documentos e fóruns sobre a área.

Apesar de todos os cuidados no desenvolvimento de uma aplicação *web*, é muito provável que ocorra algum problema. Uma forma de aumentar a resiliência a erros é a replicação da base de dados. Existem diferentes métodos de replicação, sendo os mais utilizados a replicação *master-slave* e *master-master*.

É importante também proceder ao controlo de acessos num sistema, pois permite saber que utilizadores acedem ao sistema (autenticação), o que é que cada utilizador pode fazer no sistema (autorização) e quais são as ações efetivamente realizadas pelos utilizadores no sistema (imputabilidade, ou *audit logs*). O método de autenticação mais utilizado é através de *passwords* textuais, embora seja também o mais vulnerável a ataques. Existem diversas formas de fortalecer o método de autenticação, como a autenticação multi-fator, autenticação com recurso a entidades terceiras, *smart cards*, *scans* biométricos e *passwords* gráficas. Quanto aos mecanismos de autorização, podem ser classificados como *Mandatory Access Control*, *Discretionary Access Control*, *Role Based Access Control*, *Task-Role Based Access Control* e *Attribute Based Access Control*.

É ainda importante considerar a encriptação dos dados e das comunicações. Os algoritmos reversíveis permitem que o conteúdo cifrado possa ser descodificado, sendo utilizados para encriptar dados, de forma a diminuir a probabilidade de serem obtidos por atacantes. Os algoritmos

não reversíveis (funções de *hash*), por não permitirem a descodificação do resultado, são utilizados para, por exemplo, representar *passwords* e verificar a integridade de ficheiros.

Mesmo com *hashing*, as *passwords* podem ainda ser vulneráveis a ataques *brute force*, *dictionary*, *rainbow tables*, entre outros. Devem ser utilizados métodos como a utilização de CAPTCHA e *hashing* com *salt* como forma de prevenção.

Capítulo 3

Interfaces dinâmicas e gestão de formulários pelo utilizador

Neste capítulo, aborda-se a forma como as aplicações *web* são atualmente desenvolvidas, com tecnologias e conceitos-chave modernos. É também apresentado o estado da arte relativo à gestão de formulários dinâmicos pelo utilizador.

3.1 Tecnologias e conceitos para o desenvolvimento de aplicações *web* com interface dinâmica

A forma como as aplicações *web* são desenvolvidas sofreu uma grande evolução desde o seu aparecimento, em 1989. No início dos anos 90 as páginas eram documentos estáticos, apresentados em HTML básico. As linguagens *server-side* de *scripting*, como a PHP, vieram permitir o carregamento de páginas com conteúdo dinâmico (dependente da interação do utilizador) a partir do servidor, embora ainda fosse preciso recarregar as páginas para visualizar as alterações no cliente. Com o aparecimento de JavaScript, em 1995, começou a ser possível implementar maior dinamismo nas páginas HTML, pois permitiu manipular a DOM e provocar alterações visuais sem necessidade de alterar a página a partir do servidor. Em 1999, o termo *Ajax* (*Asynchronous JavaScript And XML*) surgiu como a representação de um grupo de tecnologias utilizadas para construir aplicações *web* dinâmicas, que comunicam com o servidor a partir de pedidos HTTP e alteram a interface de acordo com a nova informação, sem necessidade de recarregar a página [Goo12, UD07]. Os objetos JSON (JavaScript Object Notation) vieram facilitar esta comunicação, pois permitiram a troca de informação com objetos de formato leve, de notação simples, e independentes da linguagem utilizada [1] [W3Sb].

[1] A notação JSON utiliza a sintaxe JavaScript, mas o formato é apenas texto, que pode ser lido e utilizado como formato de dados por qualquer linguagem de programação.

Atualmente, a forma como o *software web* é desenvolvido inclui diversas ferramentas, conceitos e abordagens que podem ser bastante complexas, mas que permitem o desenvolvimento de aplicações poderosas e de agradável utilização [Eri16] (Figura 3.1).



Figura 3.1: *Timeline* das tecnologias de desenvolvimento *web* [Eri16]

Nesta secção são abordadas algumas tecnologias e conceitos-chave atuais relevantes para o desenvolvimento de aplicações *web* com interface dinâmica e, mais especificamente, para o desenvolvimento deste projeto.

3.1.1 *Single Page Applications* (SPA)

"*A Single Page Application is a web application that requires only a single page loading in a web browser.*" [Sot14].

Numa implementação de SPA, o *browser* carrega a DOM completa uma única vez, e depois, com recurso a JavaScript e a técnicas como AJAX, carregamento parcial, *data binding*, utilização da DOM virtual e *routing*, são efetuadas atualizações de partes da aplicação conforme o necessário, de forma eficiente. Apesar de a aplicação ser carregada uma única vez, pode ser facilmente dada a sensação ao utilizador de navegação entre páginas diferentes, mas de uma forma mais interativa e rápida, já que apenas certas partes do documento são atualizadas. Assim, operações complexas, que antes eram desenvolvidas em *back-end*, do lado do servidor, passam a ser executadas do lado do cliente, aumentando consideravelmente o dinamismo da aplicação, e enriquecendo a experiência do utilizador [Sot14].

3.1.2 DOM virtual

A DOM (*Document Object Model*) é uma abstração do texto HTML, constituída por objetos estruturados em árvore [Kra15]. A DOM nunca foi otimizada para criar interfaces dinâmicas, e para aplicações que contêm páginas com um elevado número de elementos (por exemplo, Facebook, Twitter ou Pinterest), a manipulação da DOM com JavaScript e bibliotecas como jQuery apresenta problemas de eficiência [Fre16].

A DOM virtual é uma abstração mais leve da DOM real. As aplicações que utilizam esta solução interagem com a DOM virtual para efetuar as alterações necessárias, e só depois é que essas alterações são guardadas na DOM real, o que aumenta de forma considerável a *performance* [Fre16, Kra15].

3.1.3 *Web components*

Web components são uma série de APIs para plataformas *web*, que permitem criar novas *tags* HTML que representam componentes customizados, reutilizáveis e que podem ser utilizados com qualquer biblioteca JavaScript ou *framework* que trabalhe com HTML [Web]. Os *web components* são baseados nas seguintes especificações:

- *Custom elements* — Fundamentos para o desenho e utilização de novos tipos de elementos da DOM;
- *Shadow* DOM — A *Shadow* DOM providencia um encapsulamento da DOM e estilos CSS num *web component*, de modo a que permaneçam separados da DOM do documento principal [Moz17]. Os estilos CSS aplicados dentro do *scope* de um *web component* apenas se aplicam aos elementos desse componente. A especificação da *Shadow* DOM define como se deve utilizar o encapsulamento nos *web components*;
- HTML *imports* — Inclusão e reutilização de documentos HTML dentro de outros documentos HTML;
- HTML *template* — Como declarar fragmentos que podem ser instanciados apenas quando necessário, em *runtime*, e não quando a página carrega.

O Excerto de código 3.1 ilustra um exemplo de utilização de *web components*.

```
1 <div class="user">
2     <user-pic></user-pic>
3     <user-info></user-info>
4     <share-options></share-options>
5 </div>
```

Excerto de Código 3.1: Exemplo de utilização de *web components*.

3.1.4 TypeScript

TypeScript [2] é uma linguagem desenvolvida pela Microsoft, fortemente tipada e orientada a objetos, que é baseada em e compila para JavaScript [Tut]. O JavaScript que é gerado após a compilação do TypeScript pode ser utilizado em qualquer código JavaScript, e qualquer ficheiro

[2] https://www.typescriptlang.org/

JavaScript pode ser compilado com outros ficheiros TypeScript. Como TypeScript é uma linguagem orientada a objetos, suporta classes, interfaces, herança, etc. Como é uma linguagem compilada, ao contrário de JavaScript, que é interpretada, é possível obter erros sintáticos antes de executar o *script*.

O Excerto de código 3.2 apresenta um exemplo de código escrito em TypeScript, e no Excerto de código 3.3 é mostrado o código JavaScript que é produzido [Micb].

```
1 class Greeter {
2     greeting: string;
3     constructor (message: string) {
4         this.greeting = message;
5     }
6     greet() {
7         return "Hello, " + this.
              greeting;
8     }
9 }
```

Excerto de Código 3.2: Código em TypeScript

```
1 var Greeter = (function () {
2     function Greeter(message) {
3         this.greeting = message;
4     }
5     Greeter.prototype.greet =
          function () {
6         return "Hello, " + this.
              greeting;
7     };
8     return Greeter;
9 })();
```

Excerto de Código 3.3: Código em JavaScript

3.1.5 ECMAScript 6

A linguagem JavaScript implementa uma especificação *standard* definida pela ECMA International [3], a ECMA-262 [Via15]. ECMAScript é o nome oficial da linguagem JavaScript, e está atualmente na versão 7 (ECMAScript 2016), embora a maioria dos *browsers* ainda não suportem essa versão [W3Sa].

A versão 5 é a versão mais recente com total suporte pelos *browsers*. A versão 6 (ECMAScript 2015) representou um *update* significativo em relação à anterior, e introduziu funcionalidades muito úteis para os programadores, entre as quais:

- *Arrow Functions* — As *arrows* são uma sintaxe para a declaração de funções anónimas de forma mais simples, através de "=>". Se uma função deste tipo for declarada dentro de outra função, o valor do objeto "this" (contexto) mantém-se igual, ao contrário do que acontecia nas versões anteriores, em que cada função definia o seu próprio contexto. O Excerto de código 3.4 mostra um exemplo de uma *arrow function*.

[3] *European Computer Manufacturers Association* (http://www.ecma-international.org)

```
function Person(){
  this.age = 0;

  setInterval(() => {
    this.age++; // |this| properly refers to the person object
  }, 1000);
}

var p = new Person();
```

Excerto de Código 3.4: Exemplo da utilização de *arrow functions* com ECMAScript 6

- Classes — É possível utilizar classes, que suportam construtores, métodos estáticos e polimorfismo. A utilização de classes está demonstrada no Excerto de código 3.5.

```
class Cat {
  constructor(name) {
    this.name = name;
  }

  speak() {
    console.log(this.name + ' makes a noise.');
  }
}

class Lion extends Cat {
  speak() {
    super.speak();
    console.log(this.name + ' roars.');
  }
}
```

Excerto de Código 3.5: Exemplo da utilização classes com ECMAScript 6

- *Let* e *Const* — A *keyword* "let"vem substituir a anterior *keyword* para declaração de variáveis "var". "const" serve para declarar uma variável que não pode ser alterada dentro do *scope* da função.

- *Generators* — São funções que definem um algoritmo iterativo e gerem o seu próprio estado, permitindo "pausar" a função quando é obtido determinado valor, e passar esse valor à próxima iteração. As funções deste tipo podem ser especialmente úteis para a sincronização de chamadas assíncronas, nos casos em que determinada chamada necessita de um valor proveniente do resultado de uma chamada anterior, por exemplo. O Excerto de código 3.6 ilustra uma situação deste tipo.

```
1    var myGen = function*(){
2        var friends = yield $.get('/api/friends');
3        var profile = yield $.get('/api/profile');
4
5        //this only runs when getFriends and getProfile calls are done
6        console.log(friends, profile);
7    }
```

Excerto de Código 3.6: Exemplo da utilização *generators* com ECMAScript 6

Como ainda há muitos *browsers* que não suportam a ECMAScript 6, e de forma a permitir a utilização da nova sintaxe, pode-se recorrer ao Babel [4], que é um *transpiler* de conversão de código-fonte em ECMA 6 para ECMA 5. Atualmente já existe uma versão do Babel que converte também ECMAScript 7, mas encontra-se ainda em versão *alpha*.

3.2 Gestão de formulários dinâmicos

Os formulários *web* são um mecanismo comum a que as organizações recorrem para a recolha de dados *online*, de forma a alimentar bancos de informação para utilização futura [GHS05]. Depois do planeamento inicial do projeto e desenvolvimento da aplicação que contém os formulários, pode surgir a necessidade de efetuar alterações, de forma a acompanhar a evolução das necessidades da organização. No entanto, devido à estrutura pré-definida e estática dos formulários, as organizações ficam dependentes de programadores que possam efetuar as alterações [GHS05]. Se estas forem muito frequentes, a organização pode enfrentar altos custos de manutenção e atrasos prejudiciais nos processos.

Assim, poderá ser conveniente implementar uma solução de gestão de formulários dinâmicos pelos utilizadores. Neste contexto, entende-se por formulários dinâmicos formulários com as seguintes características:

1. O conteúdo dos formulários, nomeadamente campos e secções (ou outros), pode ser alterado pelo utilizador, não sendo os formulários objetos de estrutura previamente definida, programada na aplicação;

2. À medida que são registadas respostas nos campos do formulário, são mostrados/ocultados outros campos e/ou secções (ou outro tipo de conteúdo) conforme regras previamente definidas pelo utilizador;

Existem na literatura estudos e soluções propostas para este problema, sendo apresentadas algumas delas na secção 3.2.2. Uma vez que a implementação desta funcionalidade poderá implicar a inexistência de uma estrutura bem definida dos formulários, é também estudada a viabilidade de utilização de sistemas NoSQL na secção 3.2.1.

[4] https://babeljs.io/

3.2.1 Bases de dados relacionais e NoSQL

Os sistemas de bases de dados tradicionais (RDBMS), como MySQL, Postgres ou Oracle, implementam um modelo relacional de dados em estruturas tabulares bem definidas (relações), baseadas na álgebra relacional. Estes sistemas permitem o acesso simultâneo de diferentes utilizadores à base de dados, possuem funcionalidades de controlo de privilégios e autorizações, e providenciam ferramentas de manutenção que permitem testar, reparar e efetuar o *backup* e a reposição das bases de dados [Sol15]. Os RDBMS suportam SQL para a manipulação dos dados, e privilegiam a segurança e consistência dos dados.

O conceito de NoSQL refere-se a sistemas não relacionais de bases de dados [GHTC13]. Este tipo de sistemas surgiu devido ao desejo de desenvolver sistemas ágeis e altamente escaláveis sem um esquema de dados pré-definido, e à necessidade de acessos muito rápidos e simples aos dados (baixa latência, baixo *over-head* e sem indexação complexa) [Cha13]. Podem-se considerar quatro categorias de NoSQL (Figura 3.2):

- *Key-value* — Representa o modelo mais simples, em que o armazenamento é feito através de pares chave-valor. É apropriado para situações em que apenas é necessário aceder a um conjunto de dados através de um valor único usado como chave, apresentando alta eficiência no armazenamento dos dados, com valores muito baixos de latência nas operações de escrita, e fácil escalabilidade [GHTC13, Cha13]. Podem ser classificados como *in-memory key-value stores* (os dados são mantidos em memória, como no sistema Redis [5]) ou *persistent key-value stores* (os dados são mantidos em disco, como no sistema Riak [6]). Casos de uso incluem a implementação de *cache* de objetos, o armazenamento da informação associada à sessão do utilizador e o armazenamento de informação associada a carrinhos de compras;

- *Column* — Cada linha é identificada por uma chave e contém famílias de colunas, podendo cada linha conter famílias diferentes. Cada família de colunas atua como chave para as colunas que estão nela contidas. Este tipo de base de dados é normalmente utilizado quando o modelo *key-value* não é suficiente (a estrutura de dados é mais complexa), e há necessidade de armazenar grandes quantidades de dados [Dig14]. Cassandra [7] e HBase [8] são dois exemplos de sistemas desta categoria. Casos de uso incluem *feeds* de atividade, filas de mensagens e *caching* [Cha13];

- *Document* — Este modelo pode ser considerado um modelo *column-based* mais complexo, uma vez que não limita o número de objetos aninhados, permitindo armazenar estruturas arbitrárias muito complexas (documentos) [Dig14]. A maior parte dos DBMS deste tipo

[5] https://redis.io/
[6] http://basho.com/products/riak-ts/
[7] http://cassandra.apache.org/
[8] https://hbase.apache.org/

representam os documentos em JSON ou algum formato derivado [GHTC13]. Como permitem indexar os documentos pela chave primária e pelo conteúdo é possível efetuar interrogações ao nível do conteúdo e não apenas por uma chave, como no modelo *key-value*. Este modelo é útil quando há necessidade de construção rápida de esquemas não estruturados de dados [Cha13]. São exemplos de DBMS desta categoria o CouchDB [9] e o MongoDB [10];

- *Graph* — É um modelo que estrutura os dados em árvore, com nós que se ligam entre si através de relações. Este modelo simplifica muito a programação em aplicações com informação relacionada de forma complexa, como por exemplo redes sociais ou sistemas de inferência médica [Cha13]. São exemplos desta categoria o InfiniteGraph [11] e o Neo4j [12].

Tipo	Exemplos	Modelo	Casos de uso
Key-value	redis riak	Key | Binary data	- Quantidades massivas de dados simples - *Caching* - Mudanças rápidas de dados
Column	cassandra HBASE	Key | CF1: C1 | CF1: C2 | CF1: C3 | CF1: C4	- *Caching* - Activity feeds - Filas de mensagens
Document	CouchDB mongoDB	Key | Document (collection of K-V)	- Construção rápida de esquemas não estruturados de dados
Graph	InfiniteGraph Neo4j	Key | Properties → Key | Properties	- Dados complexos e relacionados: ex. redes sociais

Figura 3.2: Categorias de NoSQL [Cha13]

Existem muitas comparações na comunidade entre esquemas relacionais e não relacionais, e diversas opiniões. Pode-se afirmar que as maiores vantagens dos esquemas NoSQL são a alta *performance* e eficiência (baixa latência), escalabilidade e flexibilidade [GHTC13]. No entanto, estes sistemas apresentam ainda algumas limitações, nomeadamente a nível de:

- Segurança — Comparativamente aos sistemas relacionais, as soluções NoSQL apresentam um nível menor de segurança, por exemplo, devido à falta de mecanismos de autenticação e mecanismos insuficientes de autorização [Cha13];

- Consistência — Apesar de a inexistência de uma estrutura pré-definida permitir uma grande flexibilidade e rapidez de desenvolvimento, pode causar problemas de consistência de dados,

[9] http://couchdb.apache.org/
[10] https://www.mongodb.com/
[11] http://www.objectivity.com/products/infinitegraph/
[12] https://neo4j.com/

especialmente em aplicações muito complexas. Existem vários relatos de empresas que optaram por utilizar NoSQL em projetos que, com o tempo, se tornaram inconsistentes e muito difíceis de manter [Cha13];

- Maturidade e Suporte — As soluções NoSQL são mais recentes do que as RDBMS e, consequentemente, menos maduras. É muito mais provável que existam erros e problemas em ferramentas NoSQL do que nas habituais ferramentas de bases de dados relacionais, e o suporte da comunidade é menor [Wei15].

- Mecanismos *standard* de interrogações — Apesar de algumas soluções permitirem interrogações *SQL-like* (como MongoDB, Neo4j), é frequente ser necessário utilizar código específico para a solução escolhida, o que dificulta migrações de soluções [GHTC13].

3.2.2 Formulários *web* geridos pelo utilizador

Atualmente existem diferentes ferramentas *online* disponíveis que permitem aos utilizadores criarem e gerirem os seus próprios formulários. São exemplos o Google Forms [13], TypeForms [14], FormAssembly [15], Zoho [16], JotForm [17] e Wufoo [18]. Cada um destes recursos apresenta diferenças a nível de interface, opções disponíveis, visualização dos resultados e gestão dos dados. São analisados, de seguida, os formulários Google Forms e TypeForms em maior detalhe, pois são duas das ferramentas mais conhecidas e utilizadas [Gua15].

Google Forms Os formulários da Google (Google Forms) representam um bom exemplo e ponto de partida em termos de interação do utilizador e a criação de formulários dinâmicos. É possível especificar restrições de *input* como campos obrigatórios, número de caracteres mínimo/máximo num campo de texto ou número mínimo/máximo de opções a selecionar. Permite também dividir as questões em secções, e especificar que secção deve ser apresentada com base nas respostas dadas. A forma como este sistema lida com as edições dos formulários apresenta alguns problemas na visualização dos resultados, como ilustrado na Figura 3.3.

Neste exemplo, foi criada uma pergunta com as opções "A" e "B", tendo sido depois assinalada a "B"como resposta. De seguida, esta opção foi removida, tendo sido mantidas as respostas existentes no formulário. Ao adicionar novamente a "B", e ainda uma terceira opção com o mesmo título, a visualização de resultados apresenta 1 resposta na primeira opção "B" (50%) e uma resposta na segunda opção "B" (50%), apesar de, efetivamente, apenas existir uma resposta no total. Ou seja, o sistema mapeia a visualização de resultados através do título das respostas, embora não exista nenhuma validação a nível de repetições durante o processo de edição.

[13] https://www.google.com/forms/about/
[14] https://www.typeform.com/
[15] https://www.formassembly.com/
[16] https://www.zoho.com/
[17] https://www.jotform.com/
[18] https://www.wufoo.com/

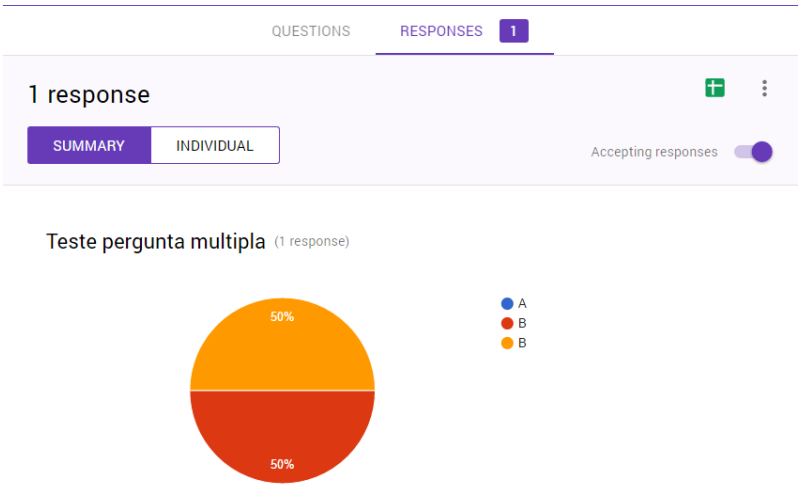


Figura 3.3: Problemas na edição de formulários Google

TypeForms Outro exemplo de criação de formulários dinâmicos é o sistema disponibilizado pela TypeForm. Estes formulários, ao contrário dos da Google, permitem incluir campos de *upload* de ficheiros. O processo de criação e configuração dos formulários é mais longo e menos simples do que os formulários os Google Forms. O sistema permite apresentar métricas de visitas dos formulários e permite integração com Google Analytics [19]. Foi efetuado um teste semelhante ao descrito anteriormente, e neste caso a informação devolvida na análise de resultados foi consistente. Depois de ter duas opções "B" no formulário e uma resposta com a primeira opção "B", fez-se uma nova submissão de resposta com a segunda opção "B", e foi dada a indicação de que tinham sido dadas duas respostas "B" (mesmo sendo opções diferentes). Depois, ao editar a segunda opção "B", alterando o seu nome para "C", o resultado apresentado foi de duas respostas "B" e zero respostas "C": a resposta original é preservada, independentemente da edição feita no formulário.

3.2.3 Trabalhos relacionados

Em [MMNL14] os autores apresentam um sistema que permite aos utilizadores definirem a estrutura de formulários através de uma interface. Os autores propõem a criação de um esquema de base de dados relacional para armazenar as definições (metadados) dos formulários. Esse esquema é apresentado na Figura 3.4.

Existe uma separação entre este esquema e o esquema de tabelas dos dados que são efetivamente inseridos pelos utilizadores, ou seja, as respostas aos formulários. Este modelo permite manter várias versões do mesmo formulário, registando todas as alterações. Se o utilizador acrescentar campos a formulários existentes com respostas, esses formulários apresentam as respostas

[19] https://www.google.com/analytics/

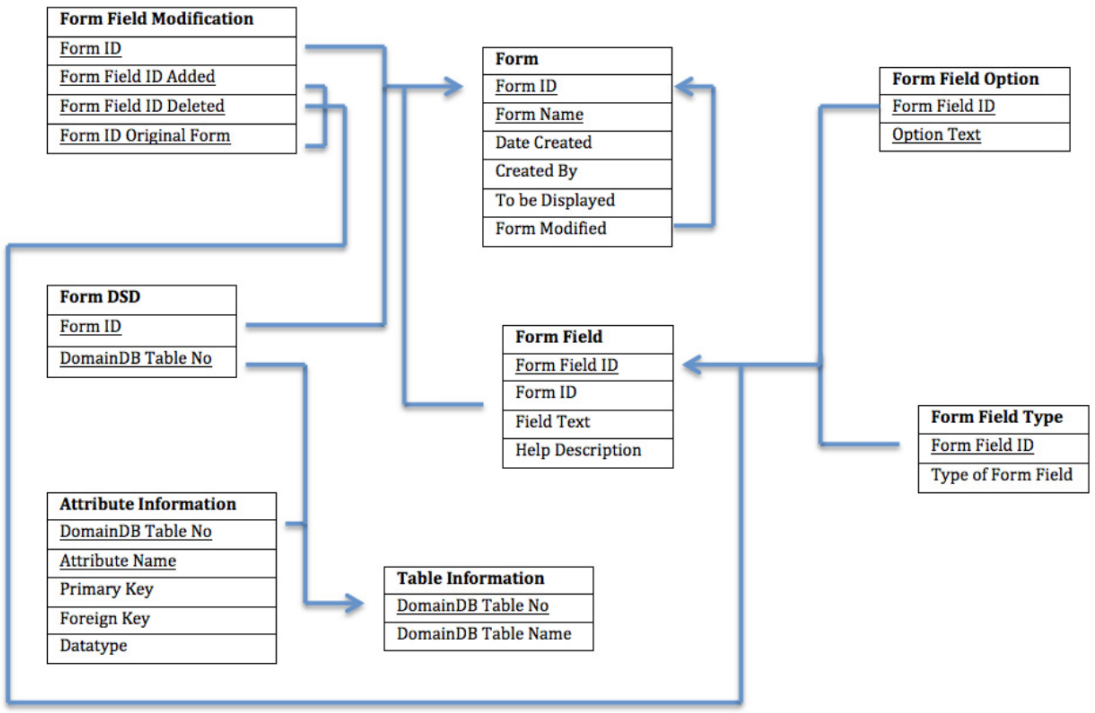


Figura 3.4: Esquema de tabelas, proposto em [MMNL14], para os metadados de formulários dinâmicos

com os novos campos preenchidos com valores *default*. Se o utilizador eliminar um determinado campo de um formulário, o campo não é removido da base de dados, mas sim ocultado. É feito um mapeamento entre as chaves de identificação dos formulários e dos campos, e os valores que são efetivamente apresentados na interface aos utilizadores. Os utilizadores podem criar formulários a partir de *templates* pré-definidos. No entanto, o modelo apenas prevê campos de formulários dos tipos *radio buttons*, *checkboxes*, *textboxes*, *dropdowns* e botões. O processo de criação de formulários exige aos utilizadores algum conhecimento de bases de dados, dado que é pedida informação sobre relações entre formulários, respetiva cardinalidade (1:1, 1:N e M:N) e atributos extra envolvidos. Na consulta de dados, é também pedido ao utilizador que indique o tipo de seleção que deseja fazer, nomeadamente operações *select*, *aggregation*, *group by* e *having*. O modelo não suporta o controlo de *input* do utilizador nas respostas (não é possível especificar, por exemplo, que um determinado campo é obrigatório).

Em [KPZ12] os autores propõem uma *framework* para simplificar a implementação de formulários geridos pelos utilizadores. A *framework* prevê a criação de formulários com vários tipos de campos e com alguma validação (campos obrigatórios), e também a reutilização de campos em diferentes formulários. São apresentados os fluxos recomendados nos processos de criação, apresentação e preenchimento dos formulários. Os autores não referem como lidar com a edição de formulários com respostas dadas.

Em [GHS05] os autores apresentam um sistema de criação de formulários dinâmicos para

```
<elementLibrary>
    <LibraryElement id="libId70">
        <elementName><![CDATA[Subject]]></elementName >
        <displayName><![CDATA[Subject/Title/topic]]></displayName >
        <elementType>string</elementType >
        <elementBriefDescription>
            <![CDATA[Topic of the Lesson Learned]]>
        </elementBriefDescription >
        <detailedDescription>
            <![CDATA[The Subject or Title reflects the primary topic of the Lesson
            Learned. A carefully written Subject should
            contain some of the Topic(s) that will help
            other users find this lesson in the
            future.]]>
        </detailedDescription >
        <presentationStyle><![CDATA[mystyle.xsl]]></presentationStyle>
        <approved><![CDATA[true]]></approved>
    </LibraryElement >
    ...
</elementLibrary >
```

Excerto de Código 3.7: Exemplo XML para definir elementos do formulário proposto em [GHS05].

serem utilizados pelos cientistas e engenheiros da NASA, de forma a submeterem lições aprendidas relacionadas com os seus trabalhos e atividades. Uma vez que, consoante a atividade, os formulários podiam sofrer variações, surgiu a necessidade de permitir que os utilizadores os personalizassem. No sistema é disponibilizada uma biblioteca de elementos que podem ser utilizados para construir os diferentes formulários. Esses elementos são definidos em XML, como ilustrado no excerto de código 3.7.

Os utilizadores podem criar novos elementos e submetê-los no sistema. Cada submissão necessita de ser aprovada por um administrador que verifica, de forma manual, se o elemento a adicionar não é equivalente a outro já existente. Todos os dados relativos às respostas colecionadas são armazenados num repositório XML, e é utilizado XPath para permitir a funcionalidade de pesquisa de informação.

Em [HT01] é apresentado um sistema para gerar formulários a partir de metadados armazenados em tabelas relacionais, utilizando a plataforma ColdFusion [20]. As respostas são armazenadas numa tabela que tem como chave primária um identificador de sistema e um identificador da opção associada à resposta dada por um utilizador. Quando um formulário é modificado, todas as associações entre o formulário e conteúdo são eliminadas para simplificar o processo. Depois, os novos elementos são associados ao formulário, sendo repostas as respostas já fornecidas pelos utilizadores através do campo *name* de cada opção. Assim, se forem removidas questões/opções que já estão associadas a respostas, estas são recuperadas.

[20] http://www.adobe.com/pt/products/coldfusion-family.html

3.3 Sumário

A forma como as aplicações web são desenvolvidas sofreu uma grande evolução desde o seu aparecimento. Inicialmente as páginas web eram documentos estáticos, obtidos integralmente através do servidor; mais tarde, começaram a surgir tecnologias (JavaScript, AJAX) que permitiram a atualização de informação nas páginas sem necessidade de as recarregar completamente, aumentando assim a qualidade da experiência de utilização. Nasceu assim o conceito de Single Page Applications (SPA). Atualmente, são muitas as tecnologias disponíveis na área, e são utilizadas técnicas como DOM virtual ou *web components* para implementar aplicações altamente dinâmicas e eficientes.

Os formulários web são um mecanismo comum a que as organizações recorrem para a recolha de dados online, de forma a alimentar bancos de informação para utilização futura. No entanto, devido à sua tradicional estrutura pré-definida, as organizações ficam frequentemente dependentes de programadores para alterarem os formulários quando necessário. Se as alterações forem muito frequentes, a organização pode enfrentar altos custos de manutenção e atrasos prejudiciais nos processos. Existe assim interesse em implementar a possibilidade de manipulação de formulários dinâmicos pela organização. Uma vez que isto pode implicar uma estrutura desconhecida à partida, surge o interesse de investigar soluções NoSQL, sendo consideradas quatro categorias: *Key-value*, *Column*, *Document* e *Graph*. Existem também soluções na literatura para a implementação de formulários dinâmicos utilizando bases de dados relacionais, em situações em que a consistência e segurança dos dados é prioritária.

Capítulo 4

Processo de atendimento na APAV

Neste capítulo é feita uma apresentação da APAV e do seu trabalho, evidenciando determinados problemas com que a instituição atualmente se depara e que motivaram a realização do projeto apresentado nesta dissertação.

4.1 A Associação Portuguesa de Apoio à Vítima

A Associação Portuguesa de Apoio à Vítima (APAV) é uma instituição particular de solidariedade social, que tem como missão prestar apoio de qualidade, confidencial e gratuito às vítimas de crime e aos seus familiares e amigos [Ass16b]. A instituição, de âmbito nacional, foi fundada em 1990 e a sua sede localiza-se em Lisboa; presta atualmente apoio a vítimas de mais de 40 tipos de crime diferentes.

Importa introduzir aqui os conceitos de vítima direta e vítima indireta, uma vez que estes termos serão mencionados ao longo do restante documento. Vítima direta refere-se à pessoa que sofreu um atentado à sua integridade física e/ou mental, um sofrimento de ordem moral ou uma perda material [Ass16b]; vítimas indiretas são a família próxima ou as pessoas a cargo da vítima direta, e as pessoas que tenham sofrido um prejuízo ao intervirem para prestar assistência às vítimas em situação de carência, ou para impedir a vitimação [Ass16b]. Em 2016, a APAV registou 12.450 processos de apoio com atendimentos nesse ano, e um total de 35.411 atendimentos (atendimentos subsequentes - acompanhamento de casos), nos quais se identificam 9.347 vítimas diretas [Ass17b].

A APAV presta quatro tipos de apoio [Ass16b] através de diferentes serviços, entre os quais se destacam a Linha de Apoio à Vítima - LAV (apoio através da via telefónica) e a rede nacional de Gabinetes de Apoio à Vítima - GAV (apoio presencial):

- Apoio prático e informações genéricas — informação sobre o crime, risco, diligências e aspetos práticos, tipos de apoios prestados pela APAV, recursos de apoio existentes na comunidade e articulação com outros serviços;

- Prestação de informação jurídica — informação sobre os direitos da vítima, etapas de determinados processos judiciais, questões financeiras;
- Apoio psicológico — alívio e melhoria dos sintomas, redução do desconforto e comportamentos disfuncionais, melhoria das capacidades de julgamento da realidade, reforço da auto-estima, maximização da autonomia, restabelecimento do equilíbrio psicológico;
- Apoio social — articulação com serviços de habitação social, casa de abrigo e ação social local.

Em 2016, 80,5% dos contactos feitos à APAV foram realizados através da via telefónica, e 29,1% de forma presencial, tendo os restantes realizado-se através de e-mail, redes sociais, fax e por carta [Ass17b].

O projeto apresentado nesta dissertação foca-se no processo de atendimento da Linha de Apoio à Vítima e dos Gabinetes de Apoio à Vítima, e no sistema utilizado para armazenar toda a informação recolhida sobre os atendimentos nestes serviços.

4.2 Plataforma de Processos de Apoio Online

Um processo de apoio é "um conjunto de diligências (atendimentos, contactos com outras instituições, etc), que visa contribuir para a prevenção e minimização dos efeitos da vitimação" [Ass16b]. A informação relativa aos processos de apoio das vítimas é reunida num sistema interno da APAV, chamado Plataforma de Processo de Apoio Online (PAO). É muito importante que a informação que consta na plataforma seja consistente, atualizada e completa, pois é com esses dados que os técnicos de apoio conseguem acompanhar devidamente as vítimas, e é a partir dessa informação que é feita a monitorização da qualidade dos serviços da APAV e são gerados relatórios estatísticos [Ass16b]. Estes relatórios são depois consultados por investigadores, meios de comunicação social, agentes políticos, entre outros [Ass16b], e por isso devem refletir a atividade da associação de forma precisa.

A plataforma PAO é utilizada pelos gestores de gabinete, técnicos e voluntários que realizam atendimentos. Os acessos são controlados a partir de uma outra plataforma (BDLena) dedicada à gestão de utilizadores, onde é mantido registo de todas as pessoas associadas à APAV. Os nomes de utilizador e *passwords* de acesso ao PAO são gerados para os utilizadores da BDLena que reúnem os requisitos necessários para poderem realizar atendimentos (ciclo de formações completo, por exemplo).

Os utilizadores podem ter um de dois níveis de acesso: administrador ou utilizador normal. Os administradores podem monitorizar, de forma limitada, os acessos e a atividade dos utilizadores da plataforma e do sistema da LAV: data/hora de *login* e *logout*, e algumas ações relacionadas com os processos (nove tipos de ações, entre as quais atualizar, consultar, criar).

A plataforma permite também a exportação de dados no formato CSV para tratamento estatístico. Em alguns casos, o utilizador pode filtrar esses dados por ano, mês e gabinete. As opções

de exportação incluem a exportação de todos os processos, do total de crimes por gabinete, ou a exportação de processos relacionados com determinado crime.

No PAO, os processos de apoio consistem numa série de formulários que seguem uma estrutura fixa pré-definida, comum a todos os tipos de crime [Ass16b]:

1. Atendimentos — Registo de informações genéricas sobre o processo e sobre os atendimentos realizados no âmbito do processo de apoio, sendo relevante para os responsáveis pela produção dos relatórios estatísticos, de monitorização e de qualidade, conseguirem perceber quantos atendimentos o processo tem, bem como a duração de cada um;

2. Dados Vítima/Utente — Registo de dados de caráter socioeconómico acerca da vítima/utente. Serve também para registar informação acerca do primeiro contacto do/a utente/vítima com a APAV, bem como da existência ou inexistência de crime;

3. Vitimação — Registo da informação relacionada com a vitimação, como tipo de vitimação, local/locais do(s) crime(s), existência de recurso a arma, informação relativa a queixas/denúncias, pedidos de indemnização, entre outras;

4. Autor do Crime — Registo dos dados pessoais e de carácter socioeconómico do(s) autor(es) do(s) crime(s);

5. Apoio Prestado — Registo de informação acerca do apoio prestado à vítima/utente antes e depois do contacto com a APAV;

6. Anexos — Descrição de todas as informações complementares referentes ao processo de apoio, sendo o formulário constituído por vários campos de texto (não se destina ao *upload* de ficheiros).

Na Figura 4.1 são apresentadas duas vistas de um processo no PAO, mais especificamente das fichas "Dados vítima/utente"e "Atendimentos".

A plataforma PAO permite ainda o cálculo do grau de risco nos casos de mulheres vítimas de violência doméstica nos relacionamentos íntimos heterossexuais. O grau de risco é um valor numérico inteiro, inserido numa escala pré-definida e calculado com base nas respostas dadas pelas vítimas a determinadas questões. Esse valor serve de referência para os técnicos decidirem de que forma devem acompanhar a vítima e que ações devem tomar. Por exemplo, no caso de um grau de risco menor que 8, o técnico de apoio deve efetuar a primeira reavaliação do risco até 60 dias; no caso de um grau de risco maior que 18, esta reavaliação deve ser realizada até 24 a 48 horas, uma vez que representa risco extremo; em situações de risco extremo e severo existem ainda indicações internas de procedimentos a efetuar, como fazer denúncia à autoridades policiais [Ass16b].

4.3 Atendimento em GAV e em LAV

O atendimento na LAV é prestado a partir das instalações da sede em Lisboa. Para auxiliar o processo de atendimento, os técnicos utilizam um *software* chamado Script Red Monitor



(a) Ficha "Dados vítima/utente"

(b) Ficha "Atendimentos"

Figura 4.1: Vista de processo na Plataforma de Processo de Apoio Online

(Figura 4.2), doravante designado por Guião LAV, que permite preencher os formulários pré-definidos de entrevista. Este programa foi desenvolvido pela empresa REDITUS [1], e integra com o software de VoIP Bria [2] para que sejam iniciados formulários em branco quando são recebidas novas chamadas.



Figura 4.2: Script Red Monitor, utilizado nos atendimentos da LAV

Assim que o técnico dá por encerrado o atendimento telefónico no Guião LAV, a informação aí recolhida é transferida de forma automática para o PAO, sendo indicado ao técnico o número de

[1] http://www.reditus.pt/
[2] http://www.counterpath.com/bria/

processo atribuído a esse atendimento. Os problemas que existem na transferência de dados entre os dois sistemas serão analisados na secção 4.4.

O atendimento presencial em gabinete é prestado pelos técnicos nos diversos gabinetes existentes no país, sendo nesse caso toda a informação recolhida em papel, e a inserção dos dados na plataforma PAO é efetuada manualmente mais tarde.

4.4 Análise do problema

Nesta secção são apresentados os problemas que atualmente existem na plataforma PAO e no Guião LAV.

4.4.1 Plataforma PAO

A plataforma PAO apresenta vários problemas em diferentes domínios, nomeadamente na interação entre os utilizadores e o sistema, no controlo de acessos e monitorização da atividade, e na lógica de organização e gestão dos processos e atendimentos.

Sistema antigo e não responsivo A plataforma de PAO é um sistema muito antigo e encontra-se bastante desatualizado em relação às necessidades atuais da APAV. Apresenta uma interface pouco intuitiva e não responsiva aos diferentes tamanhos de ecrã, impossibilitando a sua utilização em dispositivos móveis, fora do posto fixo, o que será necessário em alguns casos. Por estas razões, os técnicos em gabinete optam por registar toda a informação sobre os atendimentos em papel, e só depois a transcrevem para a plataforma. Isto resulta em trabalho extra para os técnicos, e em dificuldades relacionadas com a legibilidade das notas registadas durante os atendimentos, resultado da rapidez de escrita.

***Logout* automático obrigatório** Quando um utilizador acede à plataforma, é iniciado um contador de 1 hora, ao fim da qual é executado o *logout* automático do utilizador por razões de segurança. Por vezes o utilizador necessita de continuar a trabalhar no sistema durante mais tempo, mas não existe forma de prolongar o contador. Assim, quando faltam poucos minutos até ao *logout* automático, o utilizador tem de interromper o seu trabalho, guardá-lo e voltar a aceder à sua conta para o continuar; no caso de estar a editar um processo deve pesquisá-lo novamente, o que nem sempre é uma tarefa fácil. Se o utilizador não estiver atento, pode mesmo acontecer perder todo o trabalho que estava a efetuar, por não existir gravação automática.

Níveis de acesso limitados O PAO é utilizado por pessoas com vários tipos de funções diferentes: administradores, gestores de gabinete, técnicos de apoio e voluntários, e pessoal do secretariado. Os gestores, técnicos e voluntários estão alocados a um gabinete (domínio), e devem poder apenas aceder à informação desse domínio. No entanto, atualmente a plataforma suporta apenas dois níveis de acesso, administrador e utilizador normal, o que não é de todo adequado ao cenário real e às necessidades de segurança.

Monitorização insuficiente Embora exista a possibilidade de os administradores monitorizarem os acessos dos utilizadores, a informação que é disponibilizada neste âmbito revela-se insuficiente para as necessidades da APAV, pois não apresenta o nível de detalhe necessário à análise pretendida. Por exemplo, não é possível indicar o percurso de um dado processo, que é importante nos casos de encaminhamento de processos para diferentes gabinetes ou serviços.

Não distinção entre vitimação direta e vitimação indireta A distinção entre vitimação direta e indireta é importante tanto a nível estatístico como para perceber o tipo de apoio que é necessário dar às vítimas. As vítimas indiretas não devem ter processos atribuídos, sendo apenas indicadas no processo da vítima direta. Pode acontecer que uma vítima indireta num processo possa vir a tornar-se vítima direta mais tarde, e nesse caso deve ser então criado o processo para essa vítima. No PAO todas as vítimas são tratadas como vítimas diretas.

Impossibilidade de relacionamento de processos Esta questão está diretamente relacionada com o ponto anterior: no caso em que uma vítima indireta de um processo se torna vítima direta, deve ser criado um processo para essa vítima. Esse novo processo apresenta uma relação com o anterior, que deve ser indicada. Uma vez que o sistema não distingue os dois tipos de vitimação, também não permite este relacionamento.

Inexistência de histórico de atendimentos/evolução dos processos Ao longo do acompanhamento de uma vítima com processo, podem ser realizados diversos atendimentos, em diferentes serviços. Em cada atendimento o técnico recolhe informação que depois passa para o processo no PAO. No entanto, não existe forma de saber que informação foi recolhida em cada atendimento, pois os dados são simplesmente atualizados no processo. Desta forma é muito fácil haver perda de informação importante.

Processos não adaptados aos diferentes tipos de crime A estrutura dos diferentes formulários que constituem o processo é fixa, estática e comum a todos os tipos de crime:

- Todos os campos são apresentados no formulário, fazendo com que este seja desnecessariamente longo, pois há campos que apenas deveriam aparecer em determinadas situações. Por exemplo, o campo "Que tipo de arma?" apenas deveria ser mostrado se a resposta ao campo "Na prática do crime houve recurso a alguma arma?" fosse "Sim";
- Existem questões que, apesar de não estarem previstas nos formulários atuais (recorde-se que estes formulários não podem ser alterados), têm de ser colocadas às vítimas. Deste modo, para além de ocorrerem ocasionais esquecimentos, os técnicos são obrigados a registar as respostas num campo genérico de anotações.

Necessidade de registo dos atendimentos *a posteriori* É importante que os atendimentos realizados fiquem corretamente registados para fins estatísticos e de monitorização. Atualmente é

necessário efetuar esse registo após a realização do atendimento, indicando a data/hora de início e fim, e o técnico responsável. Pode acontecer que o técnico não registe corretamente os dados, ou não faça o registo de todo, por motivo de esquecimento.

4.4.2 Guião LAV

O Guião LAV representa um formulário de atendimento mais atualizado em relação aos formulários de processo no PAO. No entanto, existem também problemas com este sistema, e com a ligação, à plataforma PAO, dos dados que são nele recolhidos.

Impossibilidade de acesso remoto Por vezes os técnicos recebem chamadas telefónicas para apoio fora do horário de trabalho na sede. Uma vez que o sistema Guião LAV é uma aplicação local, de interface não responsiva, instalada apenas nos computadores da sede da APAV, os técnicos não conseguem utilizá-lo noutros dispositivos ou lugares. Nessas situações optam por registar a informação num formulário construído na Formsite [3].

Problemas de funcionamento sem mecanismo de recuperação de erros Por vezes o Guião LAV bloqueia durante as chamadas, levando à perda de algumas respostas. Para contornar esta situação os técnicos registam a informação num documento MS Word, o que não é prático. Se o técnico já tiver desligado uma chamada mas ainda não tiver concluído a inserção dos dados no Guião, e entretanto receber uma nova chamada via Bria, o sistema do Guião inicia automaticamente um novo formulário e toda a informação do anterior é perdida.

Integração incompleta e parcialmente incorreta com o PAO Embora a informação que é registada no Guião LAV seja automaticamente transferida para o PAO, esta transferência não é completa nem totalmente correta, pois apenas são passadas as respostas às perguntas de carácter obrigatório e de algumas de caráter não obrigatório. Este facto obriga a que o técnico tenha depois de aceder ao processo para complementar e ordenar manualmente a informação que aí consta.

Limitação a nível da gestão A APAV procura melhorar continuamente o seu atendimento. Para isso pode ser necessário alterar os formulários de atendimento pré-definidos, ou adicionar novos formulários para poder ser possível prestar apoio a outros tipos de crime. No entanto, o sistema atual não permite estas funcionalidades. Acresce ainda que, como já referido, os formulários apresentam-se muito extensos e complexos, sem qualquer tipo de adequação dinâmica aos contextos. Isto faz com que os técnicos tenham dificuldade em utilizar o Guião.

[3] https://www.formsite.com/

4.5 Sumário

A APAV é uma instituição de solidariedade social que presta apoio confidencial e gratuito às vítimas de crime e aos seus familiares e amigos, através de diferentes serviços. O projeto apresentado nesta dissertação foca-se nos serviços de atendimento em gabinete (GAV) e de atendimento pela via telefónica (LAV).

Para registar a informação que provém dos diversos atendimentos, a APAV utiliza dois sistemas: a Plataforma de Processos de Apoio Online (PAO), que consiste numa aplicação web onde são armazenados os processos das vítimas, e o Guião LAV, uma aplicação local instalada nos computadores da sala de atendimentos da sede, que consiste num formulário/guião com questões a serem colocadas pelos técnicos de apoio às vítimas.

A plataforma PAO apresenta vários problemas em diferentes domínios, nomeadamente na interação entre os utilizadores e o sistema, no controlo de acessos e monitorização da atividade, e na lógica de organização e gestão dos processos e atendimentos. Embora o Guião LAV represente um formulário de atendimento mais atualizado em relação aos formulários de processo no PAO, existem problemas a nível da ligação dos dados recolhidos no sistema ao PAO, problemas de funcionamento e limitações na gestão do guião.

Capítulo 5

Plataforma de Gestão de Atendimentos

Neste capítulo é apresentada a Plataforma de Gestão de Atendimentos (PLAGA), o novo sistema criado, no âmbito desta dissertação, para gerir o processo de atendimento à vítima na APAV.

5.1 Introdução

A criação da Plataforma de Gestão de Atendimentos (PLAGA) surgiu a partir da necessidade de resolver os problemas descritos no capítulo 4, e de acrescentar outras funcionalidades não existentes até ao momento. No início do projeto, a APAV indicou alguns pontos importantes a ter em conta no desenho do sistema:

1. Os utilizadores devem poder efetuar atendimentos e consultar os processos;
2. Os formulários de atendimento devem poder ser editáveis e apresentar dinamismo (mostrar/ocultar campos com base em respostas anteriores; definir próxima secção com base em respostas da secção atual);
3. O sistema deve estar preparado para utilizadores com diferentes papéis e permissões;
4. Deve ser possível monitorizar a atividade dos utilizadores de forma detalhada;
5. Deve haver suporte à distinção entre vítimas diretas e vítimas indiretas;
6. Deve poder ser criado um processo para uma vítima indicada como vítima indireta noutro processo, e estabelecer a relação entre o processo de origem e o novo;
7. Deve ser possível encaminhar processos para diferentes gabinetes, e seguir o percurso desses processos;
8. Deve existir uma área pessoal do utilizador, onde este deve poder gerir alertas e eventos de calendário;

9. Deve existir uma lista de processos com maior restrição de acessos para os casos de utentes que, por exemplo, são figuras públicas (Lista de Processos de Acesso Restrito - PAR).
10. Num mesmo processo devem poder ser indicados vários crimes e vários autores. É possível que a vítima possa ter mais do que um processo associado, embora não seja comum;
11. É possível que uma vítima se torne autor de crime, e vice-versa, pelo que nos processos deve ser apresentada essa indicação, se for o caso;
12. Futuramente, deve ser possível gerar documentos para um processo, parcialmente preenchidos com os dados da vítima;
13. O sistema deve ser construído de forma a que possam ser exportados dados para tratamento estatístico, segundo a perspetiva de processo, atendimento, vitimação direta, vitimação indireta, tipo de crime, GAV/UO (gabinete/unidade orgânica) e data;
14. Deve ser possível utilizar o sistema em *tablets*, para realizar atendimentos nos gabinetes e na situações de atendimento pelo telefone, fora do recinto da LAV;
15. Se for conveniente, a estrutura dos processos existente no PAO poderá ser alterada.

Considerando o tempo disponível para o desenvolvimento do projeto, ficou estabelecido que as áreas que iriam ser mais focadas no desenvolvimento do primeiro protótipo funcional, descrito nesta dissertação, seriam a gestão dos formulários de atendimento, a área pessoal do utilizador e o processo de atendimentos. O desenho do sistema não foi elaborado todo à partida, tendo-se optado por uma metodologia mais ágil de desenvolvimento, recebendo *feedback* rápido da APAV, novos requisitos, ou alteração dos requisitos existentes.

5.2 Atores e casos de utilização

Um diagrama UML de casos de utilização ajuda a perceber, de forma rápida, que tipos de utilizadores (atores) interagem com o sistema, quais as suas relações, e os principais casos de uso (requisitos funcionais) [Mic15].

São identificados seis atores diferentes, descritos na Tabela 5.1.

Tabela 5.1: Atores do sistema

Ator	**Descrição**
Utilizador	Utilizador da plataforma
Secretariado	Utilizador com funções de secretariado
Voluntário	Utilizador com funções de técnico de apoio e voluntário
Normal	Utilizador com funções de gestores de gabinete e assessores técnicos
Administrador	Utilizador com funções de administração da plataforma
Super administrador	Utilizador com acesso a dados restritos e permissão para gerir os formulários de atendimento

Na Figura 5.1 é apresentado o diagrama UML dos casos de utilização da PLAGA.

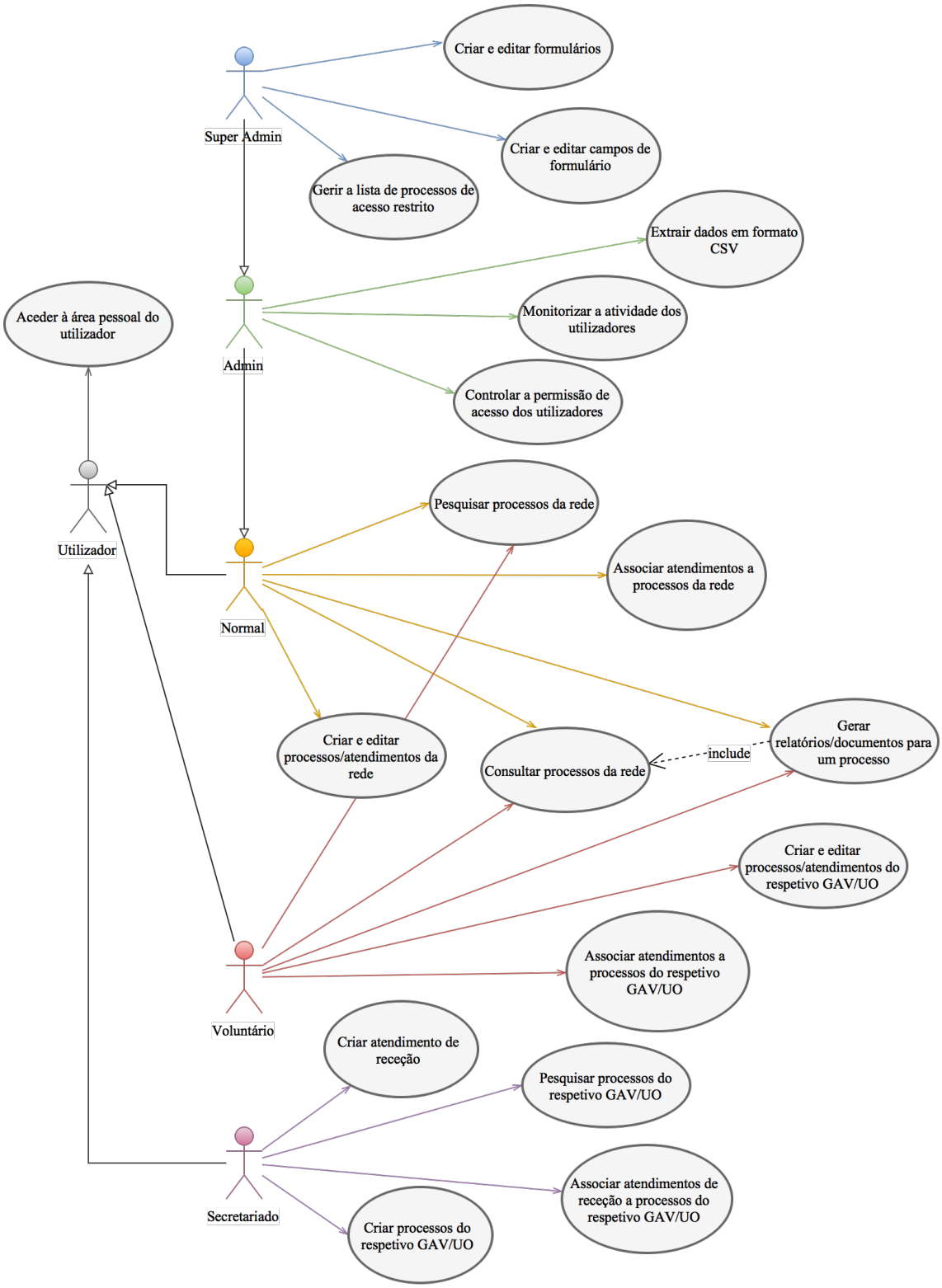


Figura 5.1: Diagrama de casos de utilização

Os utilizadores Secretariado apenas criam atendimentos de receção; podem criar novos processos quando necessário, e associar o seu atendimento a esses processos.

O utilizadores com papel de Voluntário podem consultar todos os processos da rede, mas apenas criar e editar os processos associados ao seu gabinete/unidade orgânica (GAV/UO); estes utilizadores podem também gerar documentos e relatórios associados a um processo.

Os utilizadores do tipo Normal podem pesquisar, consultar e editar todos os processos/atendimentos da rede; tal como os utilizadores Voluntário, podem também gerar documentos e relatórios associados a um processo.

Os utilizadores com papel de Administrador têm acesso a todas as funcionalidades permitidas aos utilizadores do tipo Normal, e podem ainda extrair a informação armazenada na PLAGA, em formato CSV, para posterior tratamento estatístico; têm também permissão para monitorizar a atividade dos utilizadores e controlar (atribuir/revogar) as permissões de acesso dos utilizadores do sistema.

Os Super Administradores têm acesso a todas as funcionalidades permitidas aos Administradores, e podem ainda gerir a lista de Processos de Acesso Restrito, e gerir os formulários de atendimento.

Todos os atores podem aceder à área pessoal do utilizador.

5.3 Escolha de tecnologias

Nesta secção são apresentadas e comparadas diferentes tecnologias possíveis para o desenvolvimento do projeto, a nível de *layouts* responsivos, *front-end* para páginas dinâmicas e *back-end*. Na secção 5.3.4 são indicadas as tecnologias selecionadas.

5.3.1 *Layouts* responsivos

Um dos requisitos indicados, foi a construção de uma aplicação que pudesse ser utilizada tanto em *desktop* como em *tablets*. Hoje em dia existem muitas ferramentas para o desenvolvimento de *layouts* responsivos, ou seja, *layouts* que se adaptam aos diferentes tamanhos de ecrã. Estas ferramentas funcionam à base de um sistema de grelha (*grid*), utilizando *breakpoints*, que representam diferentes valores de tamanho que os elementos da página devem tomar consoante os tamanhos dos ecrãs. As *frameworks* Bootstrap [1], Foundation [2], Semantic UI [3] e Avalanche [4] são das ferramentas deste género mais populares hoje em dia, e por isso foram analisadas.

Bootstrap Foi criada em 2010 pela Twitter [One16]. Utiliza um sistema de grelha de 12 colunas que podem ser adicionadas dentro de linhas. Para obter responsividade utilizam-se os *breakpoints* 'xs' (largura $<$ 768px), 'sm' (largura $>=$ 768px), 'md' (largura $>=$ 992px) e 'lg' (largura $>=$

[1] http://getbootstrap.com/
[2] http://foundation.zurb.com/
[3] https://semantic-ui.com/
[4] http://colourgarden.net/avalanche/

1200px), associados ao número de colunas que um elemento deve ocupar, de forma a que esses valores sejam escolhidos consoante o tamanho do ecrã. O Excerto de código 5.1 apresenta um exemplo da utilização destes *breakpoints*.

```
<div class="row">
    <div class="col-sm-12 col-md-6"></div>
</div>
```

Excerto de Código 5.1: Exemplo da utilização de *breakpoints* com Bootstrap

Atualmente, a versão estável da *framework* é a 3.3.7, mas existe já disponível a versão 4.0.0 alpha.6, que apresenta algumas alterações significativas, como o suporte de *Flexbox*[5] por defeito, mudança a nível de suporte de *browsers*, alteração do pré-processador CSS Less para Sass e alteração de componentes [Boo].

Esta *framework* apresenta alto nível de popularidade e bom suporte de desenvolvimento, tanto através da documentação oficial, como pela comunidade, existindo diversos *templates* disponíveis e integrações com outras ferramentas.

Foundation Foi desenvolvida inicialmente como ferramenta interna da ZURB, uma empresa norte-americada de *design*, mas em 2011 foi disponibilizada ao público [One16]. Embora possa ser considerara a *framework* mais popular a seguir à Bootstrap [Sta], o seu nível de utilização é bastante menor [One16]. O sistema de grelha é constituído por 12 colunas, com uma sintaxe muito semelhante à da Bootstrap *('small', 'medium', 'large')*, mas a interface é responsiva mesmo sem a adição destas classes.

Semantic UI Esta *framewok* disponibiliza mais de 20 temas de customização, entre os quais Semantic UI, Google Material, GitHub e Bootstrap. O sistem de grelha é constituído por 16 colunas e as classes de manipulação são muito *user-friendly*, de fácil leitura. O Excerto de código 5.2 exemplifica a utilização destas classes para criar um componente que ocupa 3 colunas.

```
<div class="three wide column">
```

Excerto de Código 5.2: Exemplo da utilização de classes para a responsividade dos elementos com Semantic UI

A Semantic UI integra bem com as *frameworks* Meteor, Ember.js, React e Angular.js [Sta].

Avalanche É um sistema de grelha responsivo para a construção de interfaces, que apresenta características interessantes, como a otimização de seletores através da verificação de duplicações ou redundâncias, *breakpoints* orientados à utilização em dispositivos móveis e mais legíveis

[5] *Layout* presente no CSS3 que permite posicionar os elementos de forma mais eficiente e flexível do que os *layouts* tradicionais, mesmo quando o tamanho é dinâmico ou desconhecido [Coy16].

('*thumb*', '*handheld*' e '*desk*'), possibilidade de aninhamento infinito de componentes, e escolha da nomenclatura de tamanho das colunas, de entre as possibilidades *fraction* (1/4), *percentage* (25) ou *fragment* (1-of-4) [Tom].

5.3.2 *Front-end*: Páginas dinâmicas

Com vista ao desenvolvimento de uma aplicação dinâmica, *user-friendly*, e especialmente considerando os requisitos de construção de formulários dinâmicos, tornou-se importante recorrer a uma ferramenta que facilitasse esse trabalho. As *frameworks* AngularJS [6], ReactJS [7] e VueJS [8], são das *frameworks* JavaScript atualmente mais populares para a implementação de aplicações *web* com interfaces dinâmicas [Kor17, Buc17], e por isso foram investigadas.

AngularJS Esta *framework* foi criada pela Google em 2010, e representou uma revolução no modo como se desenvolvem páginas dinâmicas [Mal16]: trabalha à base de modelos e controladores JavaScript (implementa o padrão MVC[9]), e permite efetuar o *binding* de dados aos elementos HTML através de diretivas e expressões.

> *Data-binding is an automatic way of updating the view whenever the model changes, as well as updating the model whenever the view changes. This is awesome because it eliminates DOM manipulation from the list of things you have to worry about.* [Goo]

O *binding* é feito nos dois sentidos, o que permite uma separação completa entre os modelos de dados e a interação do utilizador no *layout*. Esta ferramenta possui também um grande suporte para testes automáticos [Goo].

Apesar destas facilidades, a AngularJS apresenta alguns problemas de *performance*, obriga os programadores a seguirem determinada estrutura de organização, e considera-se que a sintaxe é muito complexa [Vue17, Mal17]. Devido a estas questões, e perante a necessidade de acompanhar a evolução do desenvolvimento das aplicações *web* e as funcionalidades/otimizações que novas ferramentas trouxeram para o mercado, a Google reescreveu a *framework* e lançou-a numa nova versão, no final de 2016, chamada Angular (versão 2) [Vue17].

A Angular apresenta muitas alterações, tanto a nível de funcionalidades disponibilizadas, como no seu funcionamento interno [Mal16]. Na nova versão, a *framework* passou a trabalhar com *web components* em vez de controladores, e a especificação de diretivas foi bastante simplificada em relação à versão anterior. A Angular é disponibilizada com uma série de *packages* de raiz, muito úteis, como validação de formulários ou *package* de definição de *routes*. Em Angular é utilizada a linguagem TypeScript [Mul15].

ReactJS Esta biblioteca é desenvolvida pela Facebook e foi lançada pela primeira vez em 2013 [Fac17]. Trabalha com uma DOM virtual, e utiliza também o conceito de *web components*, em que cada

[6] https://angularjs.org/
[7] https://facebook.github.io/react/
[8] https://vuejs.org/
[9] *Model-View-Controller*

componente deve implementar uma função *render()*, responsável pelo *output* da interface. No entanto, existe a alternativa de utilizar a linguagem JSX, que permite escrever código HTML em JavaScript, e elimina a necessidade de implementar as funções *render* para cada elemento. No entanto, é necessário utilizar uma sintaxe diferente do que é habitual em HTML, como por exemplo a utilização do atributo `className` em vez de `class` para atribuir uma classe CSS a um elemento, já que `class` é uma *keyword* reservada de JavaScript (Excerto de código 5.3).

```
1 const element = (
2     <p className="test">
3         This is an example!
4     </p>
5 );
```

Excerto de Código 5.3: Exemplo de declaração de um elemento com a linguagem JSX

Assim, a organização e estruturação de código em React é bastante diferente da de Angular, pois em Angular há separação do código dos *templates* e do código Typescript, e em React, com JSX, essa separação já não existe. Em React, ao contrário de Angular, é necessário instalar o *router* ou outros *packages* disponibilizados pela comunidade, o que pode ser uma desvantagem porque o tipo de suporte de atualizações poderá não ser tão bom como em Angular, em que os *packages* são distribuídos pela equipa de desenvolvimento oficial [Sch17]. Um aspeto que poderá ser considerado uma vantagem da React, em relação à Angular, é que em React não é necessário construir a aplicação como uma SPA, podendo-se utilizar as funcionalidades de React associadas a determinados elementos numa página [Sch17].

VueJS Esta *framework*, atualmente na versão 2, surgiu em 2014 e desde então tem vindo a ganhar grande popularidade na comunidade. Os criadores tiveram AngularJS e ReactJS como inspiração, procurando utilizar os pontos fortes e corrigir as limitações destas ferramentas, resultando numa *framework* leve, eficiente e fácil de utilizar [Vue17]. A *framework*, que utiliza *web components*, segue, embora não estritamente, o padrão arquitetural MVVM (Figura 5.2).

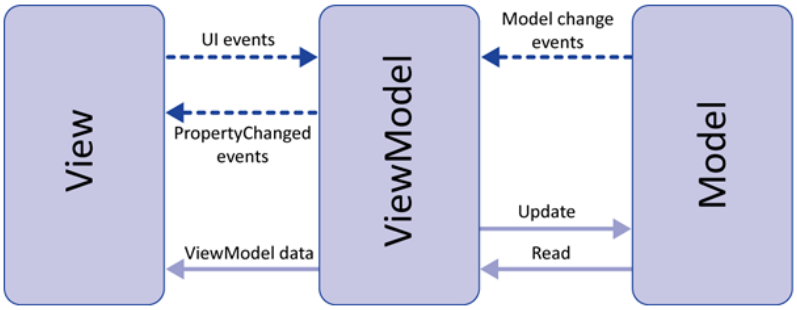


Figura 5.2: Padrão *Model-View-View-Model* [Mic17]

Este padrão pretende isolar a interface do utilizador da lógica de negócio, sendo constituído por um *Model* (representação da entidade de negócio), uma *View* (interface do utilizador) e um *ViewModel* (ponte entre a *View* e o *Model*). As operações de lógica são efetuadas com o *Model*, independentemente da forma como são depois apresentadas ao utilizador na *View*. O *ViewModel* utiliza os dados retornados do *Model* e molda-os no formato requerido pela *View*, atualizando-a [Mic17].

À semelhança de React, a VueJS utiliza a DOM virtual, embora com uma implementação menos pesada e, consequentemente, de maior eficiência [Vue17]. A forma como o código é estruturado em Vue é bastante interessante, pois existe, num mesmo ficheiro, a separação entre código HTML (*template*), código JavaScript (*script*) e código CSS (*style*). No Excerto de código 5.4, é exemplificada essa estrutura.

```
<template>
    <p>Hello World!</p>
</template>

<script>
    export default{
        data(){
            return {
                //...
            }
        }
    }
</script>

<style>
    p { color: blue; }
</style>
```

Excerto de Código 5.4: Estrutura de código em VueJS

5.3.3 *Back-end*

No início do projeto, foi decidido que o *back-end* seria desenvolvido em PHP, pois é uma linguagem muito popular e conhecida, bem documentada, e suportada por muitos provedores de alojamento. Assim, foram estudadas duas *frameworks* MVC, Laravel [10] e CodeIgniter [11].

Laravel É ativamente mantida, tendo sido a última versão (5.4) lançada muito recentemente, suportando versões de PHP superiores à 5.6.4. Esta ferramenta apresenta funcionalidades interessantes, como migrações (permite migrar o esquema de base de dados para diferentes sistemas),

[10] https://laravel.com/
[11] https://codeigniter.com/

algumas implementações de segurança a nível da *middleware* (como proteção contra ataque CSRF e autenticação), várias funções utilitárias, suporte para *routing* e inclui ORM [12] para manipular os dados [Otw]. Dada a sua popularidade, existem várias implementações de componentes e grande suporte por parte da comunidade. Esta ferramenta utiliza o Composer [13] para gerir as dependências, o que facilita a gestão de recursos, especialmente em projetos grandes. Os ficheiros JavaScript podem ser geridos utilizando a API Laravel Mix [14], que permite o controlo de versões e compilação de vários ficheiros num único ficheiro. Suporta também interface de linha de comandos. A Laravel providencia também uma *framework* que facilita muito a escrita e a execução de testes automáticos.

CodeIgniter É de instalação e utilização muito simples, apresenta alto nível de *performance*, e foi durante algum tempo a *framework* com maior popularidade na comunidade PHP. No entanto, a sua manutenção foi praticamente inexistente por alguns anos, e considera-se que se tornou numa ferramenta desatualizada e com menos recursos quando comparada à Laravel [Cun16]. Por ser muito mais simples que a Laravel, poderá ser útil em projetos pequenos, de baixa complexidade.

5.3.4 Tecnologias selecionadas

Com base na informação descrita na secção anterior, foram selecionadas as tecnologias de desenvolvimento, apresentadas a seguir.

***Backend* em PHP com Laravel** Optou-se pela *framework* Laravel v5.4, não só pelas razões apresentadas na secção 5.3.3, mas também por esta apresentar uma excelente documentação de suporte, possuir uma comunidade muito ativa que disponibiliza diversos *packages* de extensões úteis, e ser distribuída com a *framework* VueJS já configurada. A Laravel utiliza o Composer para gerir os *packages* PHP, e a npm [15] para os *packages* JavaScript.

Páginas dinâmicas com VueJS Para o desenvolvimento *front-end*, optou-se pela VueJS, que, apesar de ser uma *framework* recente, tem vindo a mostrar grande potencial. Em termos de *performance*, a Vue apresenta valores muito interessantes nos testes de *benchmarks*, sendo mais eficiente na maioria dos testes, quando comparada às restantes *frameworks* avaliadas [Kra16]. A ótima documentação disponível, o alto nível de popularidade, o estilo de estruturação de código e a utilização de *web components*, foram também fatores que contribuíram para esta escolha. Outra vantagem foi o facto de a *framework* ser já distribuída juntamente com a Laravel por defeito.

Destacam-se ainda os seguintes recursos JavaScript utilizados:

- Axios [16] — É um cliente HTTP, com proteção contra ataques XSRF;

[12] Object-relational Mapping: é uma técnica que permite a manipulação das tabelas da base de dados através de objetos.
[13] https://getcomposer.org/
[14] https://laravel.com/docs/5.4/mix
[15] https://www.npmjs.com/
[16] https://github.com/mzabriskie/axios

- Lodash [17] — É uma biblioteca com funções utilitárias, que facilita o trabalho com *arrays*, *strings*, *objects*, etc;
- Laravel Mix [18] — É uma API que abstrai a configuração do Webpack [19], através de métodos simples;
- Vue Router [20] — É o *router* oficial para a VueJS.

No desenvolvimento deste projeto foi utilizado ECMAScript 6.

***Layouts* responsivos com Bootstrap** Para a construção dos *layouts* responsivos optou-se pela *framework* Bootstrap, pois é bastante completa (HTML, CSS e JavaScript), facilmente customizável, simples de utilizar, e é disponibilizada de raiz com a Laravel. Visto a versão 4 ser ainda versão alpha, foi utilizada a versão 3.3.7.

5.4 Arquitetura do sistema

Na Figura 5.3 é apresentado o diagrama da arquitetura geral do sistema.

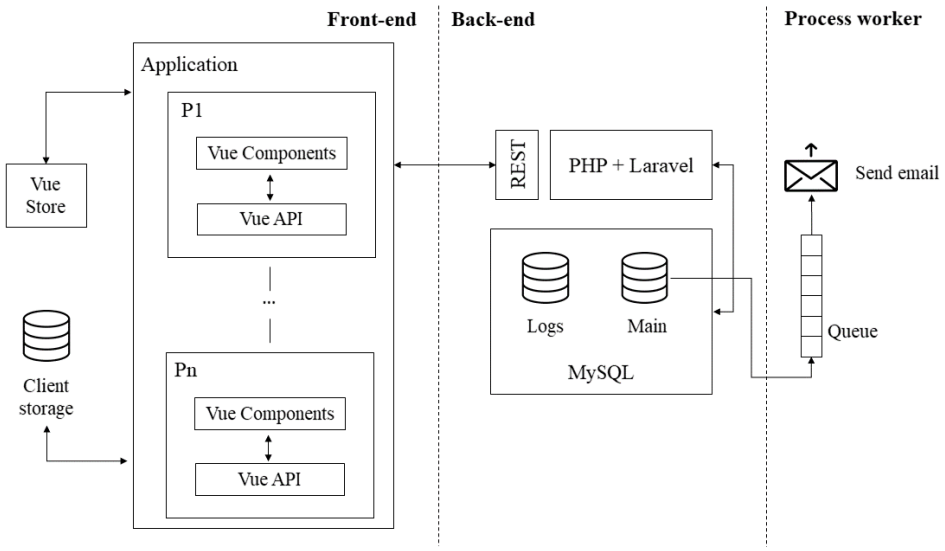


Figura 5.3: Arquitetura geral do sistema

Front-end A Laravel disponibiliza um *templating engine* bastante poderoso, chamado Blade, que permite a construção de sistemas tradicionais "*server-side rendered*" de forma simples, e

[17] https://lodash.com/
[18] https://laravel.com/docs/5.4/mix
[19] Ferramenta para compilação de módulos (https://webpack.github.io/)
[20] https://github.com/vuejs/vue-router

otimizada para melhorar a experiência do utilizador. Simples, porque disponibiliza uma série de diretivas que permitem escrever código muito mais legível, quando comparado com PHP *vanilla*, como @foreach, @unless, @forelse (entre outras). Mais otimizada, porque utiliza mecanismos de *including* de *templates* parciais e *caching*, permitindo o recarregamento parcial do documento, e tornando assim o *rendering* da página mais rápido.

Deste modo, optou-se por não implementar uma aplicação "SPA" na sua totalidade, no sentido em que existem módulos (páginas) que são carregados a partir do servidor com recurso a *blade templates*, mas onde são incluídos Vue *components*, com a API JavaScript, de forma a obter maior dinamismo nas páginas.

Em Vue, os *web components* apenas efetuam comunicação direta entre componentes-pais e componentes-filhos. A restante comunicação é assegurada a partir do sistema de emissão de eventos, onde um componente pode registar determinado evento, ficando a aguardar que outro componente o emita.

No entanto, quando a complexidade da aplicação aumenta, o sistema de eventos e interações entre componentes torna-se difícil de gerir. Recorreu-se assim a um método de registo de objetos de forma global e partilhada, a Store, onde as alterações efetuadas nos objetos aí guardados são propagadas a todos os componentes da aplicação, e as alterações efetuadas em cada componente são notificadas à Store. A Vuex é a biblioteca oficial para a implementação da Store partilhada, providenciado funcionalidades como *time travel* e separação de ficheiros. No entanto, para este projeto, dado que o estado partilhado da aplicação é simples, apenas se pretendendo um sistema de partilha de dados, e como a biblioteca Vuex apresenta alguma complexidade de utilização, optou-se por uma alternativa mais leve, a Vue-Stash [Tar17].

É ainda considerado um sistema de *client storage* (HTML5 Web Storage) para salvaguardar os dados introduzidos pelos utilizadores, por exemplo, quando editam os formulários.

Back-end Optou-se por utilizar bases de dados relacionais, MySQL, para implementar o projeto. Existem duas bases de dados que alimentam o sistema: a base de dados Main armazena os dados da aplicação, e a base de dados Logs destina-se ao armazenamento dos dados relacionados com a atividade dos utilizadores.

O sistema foi construído segundo o estilo arquitetural REST (Representational State Transfer), sendo utilizado JSON para a comunicação entre o *front-end* e *back-end*.

Process worker Existe um processo responsável por gerir as tarefas de envio de notificações por email, pendentes numa fila (*queue*). A existência deste processo permite gerir, de forma otimizada, notificações agendadas para determinada data e acelerar os pedidos *web* à aplicação, já que estas tarefas, que consomem tempo, estão delegadas ao processo. A *queue* é armazenada na base de dados Main, mas poderá mais tarde passar a ser guardada em Redis [21], de forma a otimizar a *performance*.

[21] Redis é um sistema de armazenamento de dados estruturados em memória, o que faz com que seja extremamente eficiente para implementar filas de mensagens ou dados de *cache*.

5.4.1 Considerações sobre segurança

A segurança dos dados é, em qualquer aplicação *web*, um fator importante a ter em conta. No caso da PLAGA, devido à natureza dos dados sensíveis que são armazenados e tratados, este aspeto torna-se ainda mais significativo.

Esta secção não pretende apresentar todos os mecanismos de segurança normalmente implementados numa aplicação *web*, como por exemplo certificação SSL, proteção contra ataques CSRF, ou proteção contra ataques *brute-force* (foram já feitas referências a possíveis ataques e modos de prevenção no capítulo 2), mas sim fazer algumas considerações, relacionadas com segurança, mais específicas para o caso da PLAGA.

Autenticação Apesar de ser utilizado o sistema de autenticação tradicional, através de *passwords* textuais, para determinadas áreas da aplicação de acesso mais restrito, será implementado um método de autenticação 2FA, em que o utilizador apenas pode aceder a essas áreas depois de, para além de indicar uma *password* textual válida, apresentar também um código, que recebe através de SMS. O Nexmo [22] é um exemplo de um serviço que providencia uma API para esta funcionalidade.

Sessão com *timeout* A sessão de um utilizador, que efetuou *login* na plataforma, não deve ser mantida indefinidamente, pois existe o risco de outra pessoa, que não o utilizador autenticado, aceder ao dispositivo e à informação da conta desse o utilizador. No PAO atual, existe um mecanismo de *timeout*, com a duração máxima de 1 hora, em que, ao fim desse tempo, é efetuado o *logout* automático do utilizador no sistema.

Na PLAGA, o *logout* automático só será efetuado se o utilizador não interagir com o sistema por mais do que alguns minutos (o valor exato de tempo poderá ser configurável pelo Super Administrador), evitando assim que o utilizador tenha de ser obrigado a interromper o seu trabalho.

Encriptação da base de dados Mesmo com a implementação de mecanismos de segurança a nível da aplicação, os dados armazenados na base de dados podem ser comprometidos por diferentes razões: podem ainda existir vulnerabilidades na aplicação, que permitem acesso indevido aos dados; no caso de a aplicação ser alojada por um *hosting provider*, os administradores desse alojamento podem aceder aos dados; ou atacantes com acesso aos servidores podem roubar os dados a partir do disco [PRZB12].

Uma solução que pode ser utilizada para prevenir estes ataques é a encriptação dos dados sensíveis na base de dados, e utilizar um algoritmo para os descodificar ao nível do cliente (fidedigno). No entanto, os recursos computacionais e de armazenamento do cliente podem não permitir este processo, especialmente no caso de grandes volumes de dados, de forma eficiente [PRZB12].

[22] https://www.nexmo.com/

O CryptDB [23] é uma ferramenta que pode ser utilizada para resolver este tipo de problema, funcionando com um *proxy*, por onde passam todas as *queries* SQL, que armazena uma chave secreta para reescrever essas *queries*, de modo a que possam ser utilizadas com os dados encriptados na BD; quando o servidor retorna os resultados, o *proxy* desencripta a informação e envia-a à aplicação. Os dados sensíveis nunca estão acessíveis na sua forma original, mas sempre encriptados.

5.4.2 Módulos

Considerando os problemas a serem solucionados, indicados no capítulo 4, e os pontos indicados na secção 5.1 deste capítulo, o sistema foi idealizado em cinco módulos principais: Administração, Atendimentos, Processos de apoio, Estatística e Área pessoal do utilizador, sendo, respetivamente, apresentados o desenho e detalhes de implementação mais relevantes nas secções 5.5, 5.6, 5.7, 5.8 e 5.9.

5.5 Administração

Na área de administração são disponibilizadas funcionalidades de gestão dos formulários de atendimento, monitorização da atividade dos utilizadores, e a gestão da lista de Processos de Acesso Restrito (PAR).

5.5.1 Gestão de formulários

O protótipo implementado permite a criação e edição de campos de formulário e de formulários, a atribuição de um formulário a um serviço e a consulta da atividade dos utilizadores num dado formulário ou campo.

De forma a evitar que o utilizador necessite de voltar a criar um campo já existente ao longo da edição dos formulários, optou-se pela existência de um repositório de campos, onde estão disponíveis todos os campos existentes e que podem ser utilizados nos formulários.

Na Figura 5.4 está representado o modelo de dados implementado para a área de gestão dos formulários.

A entidade Form representa um formulário criado para um serviço (Service), podendo o serviço ser a Linha de Apoio à Vítima, a Receção, o Gabinete de Apoio à Vítima, ou outro. A entidade FormField representa um campo de formulário. Tanto os formulários como os campos de formulário são identificados por um código único. Os formulários guardam ainda um número de versão, que corresponde à versão da edição do utilizador. É mantido o registo das datas de criação, atualização e eliminação dos formulários e campos, bem como dos utilizadores que realizaram essas ações. Para os campos de formulário é ainda guardada a *label*, que corresponde ao título do campo, de modo a facilitar a pesquisa por título.

Optou-se por guardar a estrutura de definição dos formulários e dos campos numa coluna *definition*, em formato JSON. A decisão foi tomada com vista a facilitar a manipulação e definição

[23] https://css.csail.mit.edu/cryptdb/#Software

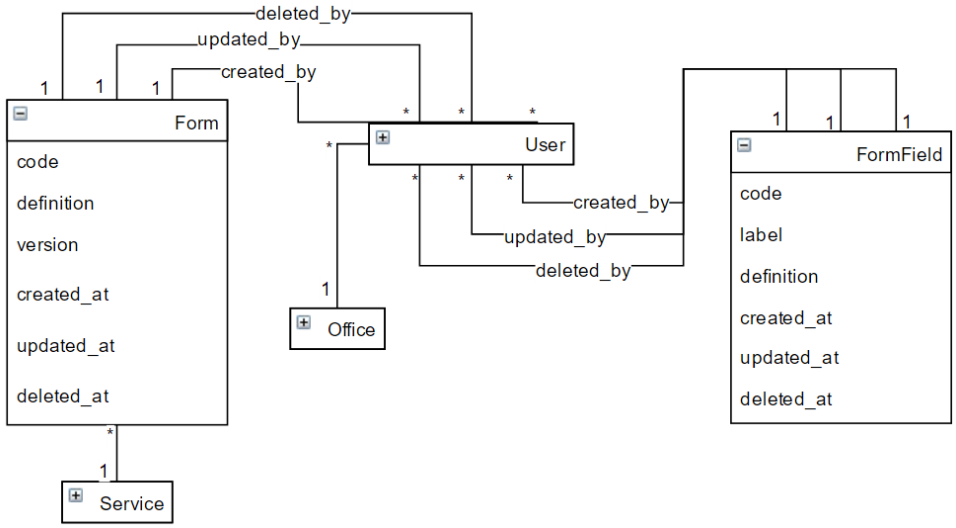


Figura 5.4: Modelo de dados relativo à gestão de formulários

dos formulários e dos campos, uma vez que a sua estrutura poderá ser frequentemente modificada, especialmente no caso dos campos de formulário, em que pode existir alteração dos elementos que compõem um campo de determinado tipo, e a adição de novos tipos. Esta solução é viável porque não serão realizadas consultas ao nível dos campos (os diferentes atendimentos, respostas aos formulários, serão exportados, no formato CSV, para tratamento estatístico posterior), mas apenas se pretende que o conteúdo total dos formulários seja apresentado.

Uma alternativa de modelação, segundo uma perspetiva mais relacional, passaria por adaptar a solução proposta em [MMNL14] e apresentada no capítulo 2, onde os autores definiram uma estrutura de tabelas de suporte à definição dos formulários, com as tabelas Form, FormField, FormFieldOption e FormFieldType (entre outras).

Campos de formulário A estrutura dos campos de formulário varia consoante o tipo de campo que é definido. Atualmente, o sistema suporta nove tipos diferentes: Radio Button, Checkbox, Select (único), Select (múltiplo), Texto Curto, Texto Longo, Data, Hora/Minuto, Data e Hora/Minuto.

Nos tipos de campo de resposta de seleção, é indicada uma listagem de respostas possíveis (*answers*); no caso dos tipos Checkbox e Radio Button, é possível indicar também se deve existir a opção outro (*opt_other*), que consiste num campo de marcação com um campo de texto livre. A *version* indica a versão da estrutura do tipo do campo. O Excerto de código 5.5 exemplifica a estrutura de um campo do tipo Radio Button.

```
{
  "label": "Como se sente com o atendimento prestado?",
  "code": "opiniao_atendimento_prestado",
  "type": {
    "code": "RADIO",
    "text": "Radio button",
    "version": "1"
  },
  "answers": [
    {
      "code": "satisfeito",
      "label": "Satisfeito"
    },
    {
      "code": "insatisfeito",
      "label": "Insatisfeito"
    },
  ],
  "opt_other": "YES",
  "help": null,
  "required": false
}
```

Excerto de Código 5.5: Exemplo da definição de um campo do tipo Radio Buttonbasicstyle

Inicialmente, foi integrado um editor (Ace Editor [24]) de texto para aceitar as especificações dos formulários e dos campos de formulário, feitas pelo utilizador, no formato YAML (Figura 5.5).

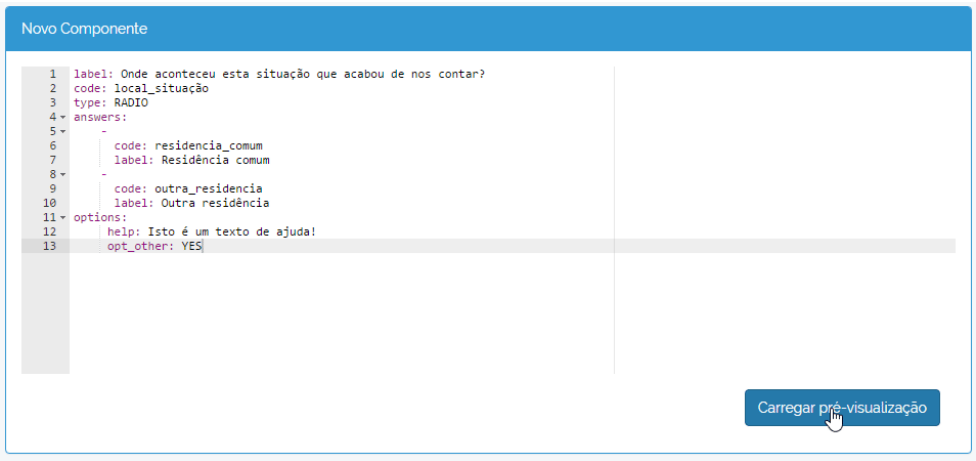


Figura 5.5: Interface de edição da especificação de campos de formulário em YAML

A opção por esse formato deveu-se ao facto de ser um formato de relativamente fácil manipulação por pessoas não ligadas à área da programação, e por ser facilmente conversível para

[24] https://ace.c9.io/

JSON. Como os campos de formulário apresentam uma estrutura simples, a sua edição através deste método apresenta baixa complexidade. No entanto, a definição da estrutura dos formulários é muito maior, e tornar-se-ia muito complicado para o utilizador geri-la a partir do editor de texto. Por isso, optou-se, mais tarde, por implementar um editor gráfico, que apresenta vantagens tanto a nível da facilidade de utilização, como da validação do *input* do utilizador (Figura 5.6).

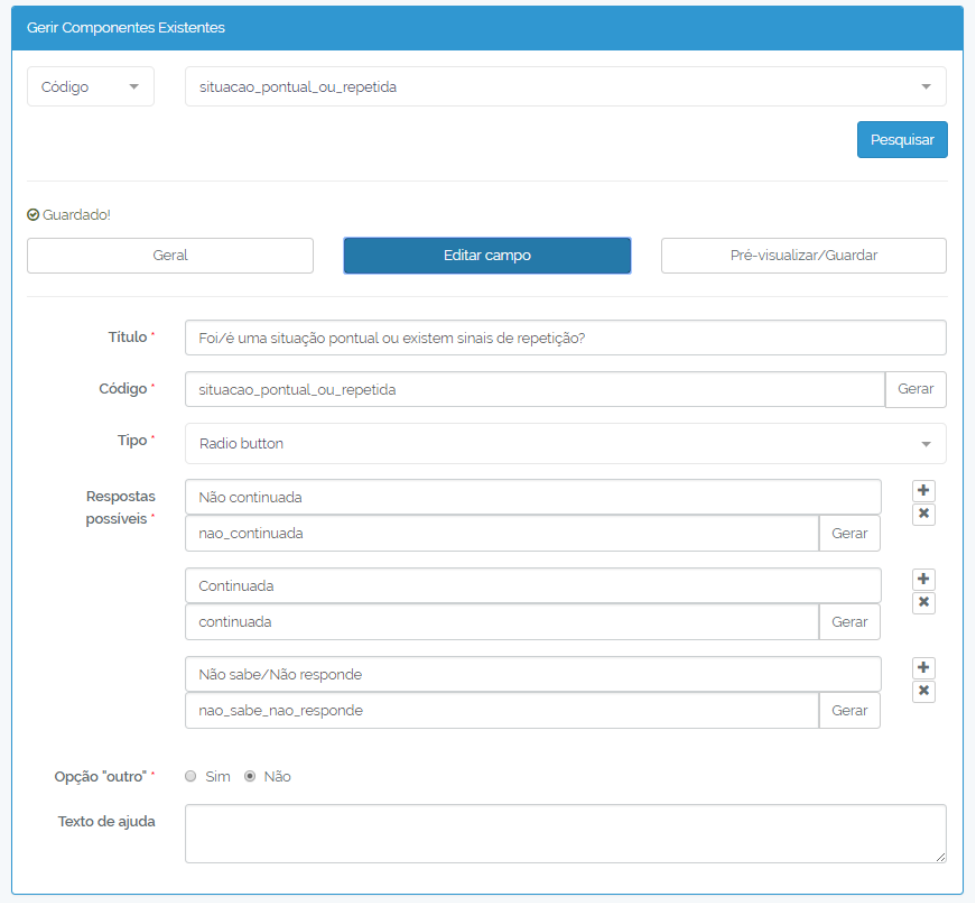


Figura 5.6: Interface gráfica de edição da especificação de campos de formulário

O processo final de criação de um campo de formulário está ilustrado na Figura 5.7. O código de um campo, indicado pelo utilizador, deve ser único na base de dados. No caso de edição de um campo, e se este já estiver a ser utilizado em formulários com respostas registadas, o sistema pede ao utilizador que indique um novo código, passando a ser tratado como um novo campo.

Antes de guardar o campo, tanto na fase de criação como de edição, o utilizador deve verificar se a especificação do campo é válida e qual o aspeto visual final (pré-visualização). No caso de existirem erros, a pré-visualização não é mostrada e o utilizador é impedido de guardar esse campo.

Permitir ao utilizador a gestão completa dos campos dos formulários leva a possíveis problemas relacionados com a consulta dos dados e tratamento estatístico: por exemplo, numa situação

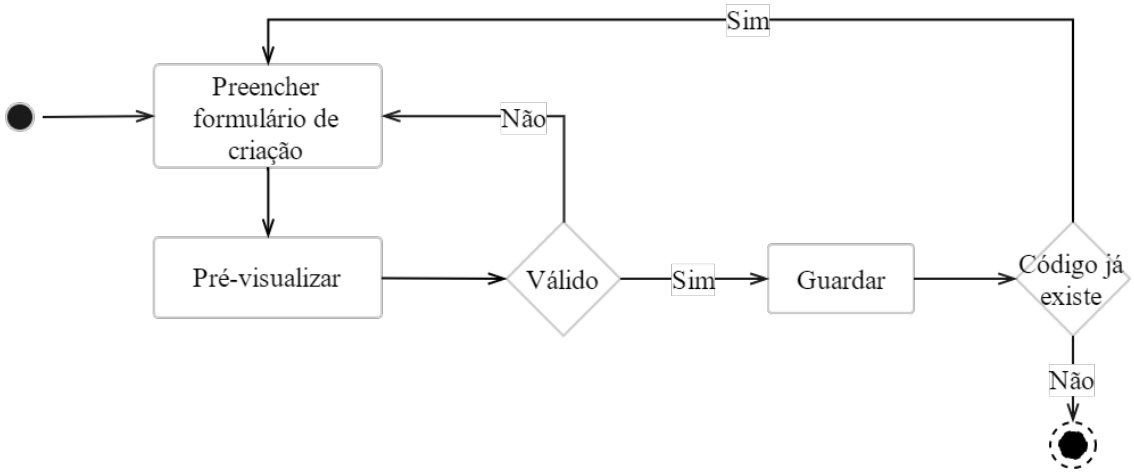


Figura 5.7: Processo de criação de um campo de formulário

em que o utilizador crie um campo morada, do tipo Texto Curto, e, depois de serem registadas respostas nesse campo, o alterar para o tipo Texto Longo, será obrigado a criar um novo campo, por exemplo com o código morada_1. A partir daí, poderão existir respostas ao campo morada_1 e ao campo morada, o que dificulta depois a análise estatística, pois, apesar de o significado de ambos os campos ser o mesmo, a exportação de dados originará duas colunas distintas. Assim, os utilizadores devem efetuar testes e planear bem os formulários, antes de os utilizarem em atendimentos "reais", evitando ao máximo este tipo de situações.

Este problema torna-se mais crítico com campos que estão na base de muitas consultas e agregações, como por exemplo campos de seleção dos crimes ou dos tipos de apoio prestados. Nestes casos, optou-se por criar tabelas que armazenem esta informação, e permitir apenas um campo "Tipo de Crime", ou "Apoio Prestado", onde o utilizador só pode escolher quais as opções visíveis. Esta decisão foi também tomada para permitir a ligação das respostas dos campos à folha de rosto dos processos de apoio (ver secção 5.7). O conteúdo destas tabelas é gerido numa área à parte.

Formulários O processo de criação e edição dos formulários é semelhante ao dos campos, mas, neste caso, o sistema de versões é gerido automaticamente pelo sistema, sendo atribuídos códigos com a forma <nome do serviço>_<número da versão>. Se o utilizador editar um formulário sem respostas registadas, a versão mantém-se e são gravadas as alterações efetuadas; se o formulário já tiver sido respondido, é criada outra versão, que passa a ser tratada como outro formulário independente do anterior.

Em cada edição de um formulário (e campo), são os utilizadores que indicam quais as alterações efetuadas, de forma a manter um histórico de edições completo, que os ajude a perceber o "percurso" de evolução de um formulário (Figura 5.8). O objetivo deste histórico é, então, mais

do que manter um sistema de *logging* da atividade na plataforma, disponibilizar um controlo detalhado de versões para os utilizadores (como acontece, por exemplo, no GitHub [25]).

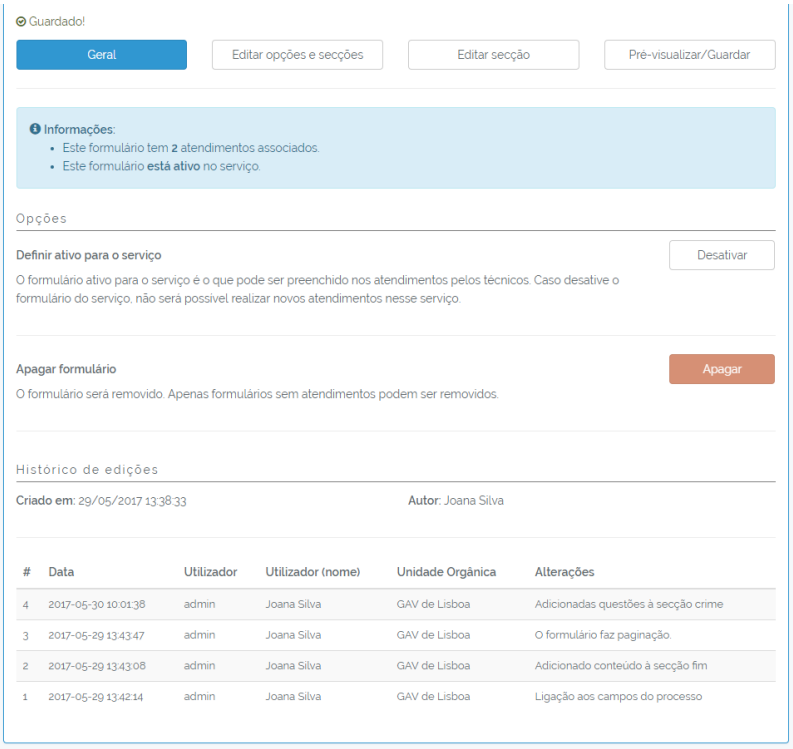


Figura 5.8: Vista do separador de Dados Gerais de um formulário, com histórico de edições

A edição de um formulário engloba duas partes principais: a edição de opções e secções existentes, e a edição de cada secção específica (Figura 5.9). Antes de poder editar uma secção, o utilizador deve criá-la na listagem de secções disponíveis do formulário, sendo cada secção identificada por um código único. As opções do formulário incluem a opção de apresentação de secções de forma corrida, ou por páginas, e qual a primeira secção que deve ser apresentada. Em cada secção, o utilizador pode adicionar campos de formulários, que podem ser campos de texto livre ou campos previamente adicionados ao repositório de campos. Para a edição dos campos de texto foi incorporado o editor MediumEditor [26], que permite uma série de formatações para personalizar o conteúdo.

Para cada campo, o utilizador deve indicar se este é sempre visível, ou se a sua visibilidade depende das respostas a outros campos. Se for este o caso, devem ser construídas regras de visibilidade, que seguem uma forma específica:

1. Ação — A ação pode ser mostrar ou ocultar, e define o que acontece ao campo se a regra for avaliada como verdadeira;

[25] https://github.com/
[26] https://github.com/yabwe/medium-editor

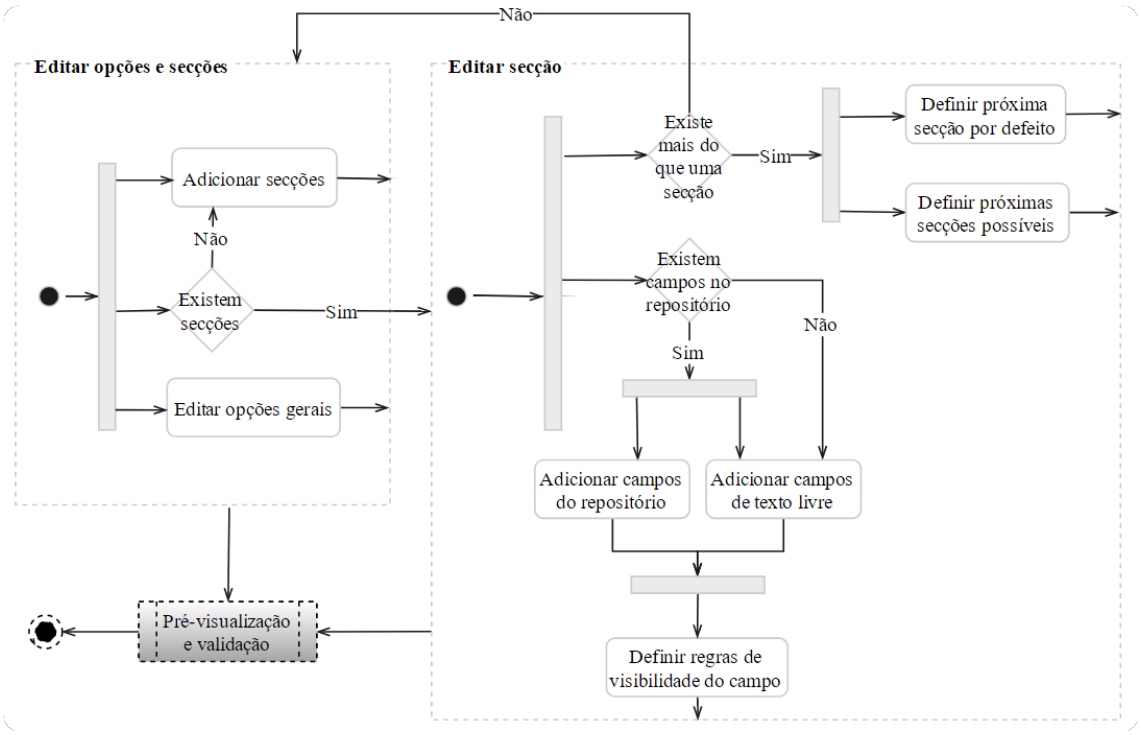


Figura 5.9: Fluxo do processo da edição das opções e das secções de um formulário

2. Quando — Pode assumir o valor todos ou algum, e indica se a regra é avaliada como verdadeira quando todos os critérios são verdadeiros, ou pelo menos um critério é verdadeiro (respetivamente);

3. Critérios — Listagem de critérios a serem avaliados, com os seguintes elementos:

   A Campo — Código de outro campo da secção, que é escolhido a partir da listagem de campos existentes;

   B Quando — Pode tomar os valores é ou não é, e indica se a resposta do utilizador ao campo deve ser igual ou diferente do valor indicado no critério;

   C Valor — Valor a ser comparado com a resposta do utilizador ao campo. No caso de campos de seleção, os valores disponíveis são apresentados; no caso de campos de texto longo/curto, pode ser indicado qualquer texto. Podem ser selecionados ou indicados vários valores num único critério, sendo que o sistema avalia o critério como verdadeiro se alguma das respostas do utilizador corresponder a pelo menos um valor indicado (a separação de valores nos campos de texto é feita através de vírgula).

   Por exemplo, na situação ilustrada na Figura 5.10, o campo seria mostrado quando a regra **(situacao_pontual_ou_repetida = nao_continuada OU continuada) E nome_utente != 'luisa'** fosse verdadeira.



Figura 5.10: Exemplo de definição de uma regra de visibilidade de um campo num formulário

Para além de adicionar campos a uma secção, o utilizador pode indicar qual a secção que deve aparecer a seguir a essa, em qualquer circunstância, e quais as secções que devem aparecer consoante determinadas regras (estas regras seguem um esquema semelhante às dos campos). É a partir destas especificações que o sistema ordena e apresenta as secções disponíveis na pré-visualização do formulário.

Na pré-visualização de um formulário, ao contrário do que acontece com os campos de formulário, o utilizador pode guardar um formulário com avisos, ou seja, situações que não são consideradas erros, pois permitem completar a pré-visualização, mas que poderão causar um funcionamento incorreto. Um exemplo será um campo que define uma regra de visibilidade baseada num campo que ainda não foi adicionado à secção, o que fará com que essa regra seja sempre avaliada como falsa.

Cada campo de formulário contém ainda uma opção "Código campo PAO", que serve para fazer a ligação desse campo à folha de rosto dos processos de apoio. A forma como esta ligação é feita será explicada na secção 5.7.

5.5.2 Monitorização de atividade

A aplicação guarda informação, na base de dados de Logs, da atividade dos utilizadores. Essa informação é apresentada, numa página, aos utilizadores com permissão de administração (Figura 5.11). Atualmente, são apresentadas as ações de Login/Logout, Consulta/Edição de campo de formulário, Consulta de processo, Consulta de atendimento e Consulta/Edição de formulário. Futuramente serão adicionadas mais ações à listagem, bem como opções de filtro dos dados.

Para além da informação armazenada na base de dados, o sistema guarda também outros dados adicionais em ficheiros de texto, como o IP, URL, método HTTP e *referrer*. Esta informação não será, à partida, consultada muitas vezes (serve para salvaguardar os casos em que ocorre algum problema, e são necessários dados adicionais para perceber a sua origem), e por isso não é armazenada na base de dados, nem mostrada ao utilizador na plataforma. Para a funcionalidade de

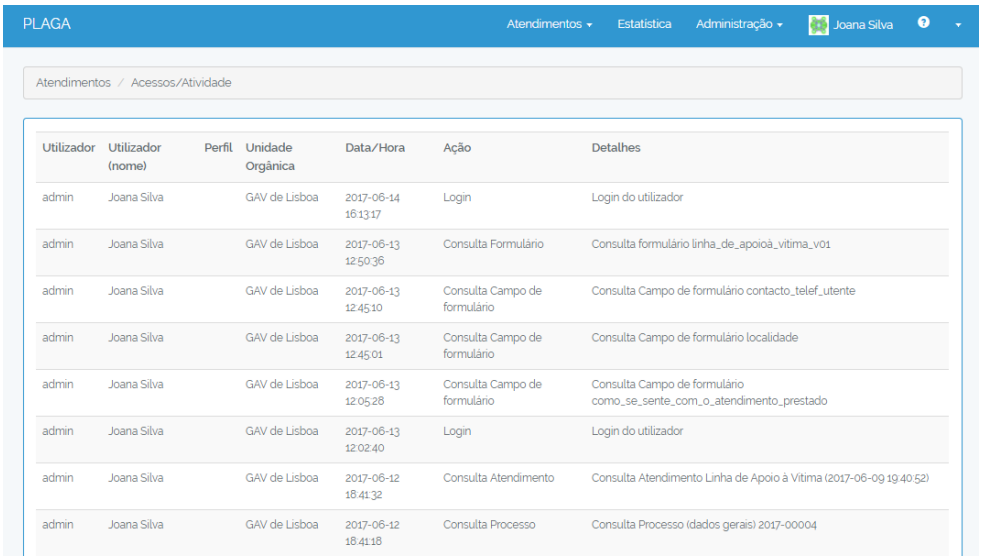


Figura 5.11: Listagem da atividade dos utilizadores

logging recorreu-se à biblioteca Monolog [27], que providencia uma série de *handlers* que facilitam a gravação da informação em diferentes formatos.

Os *logs* são categorizados em níveis (informação, aviso, erro, crítico, emergência, etc). No futuro será implementado um sistema de notificação no caso de registo de *logs* com níveis de alarme.

5.5.3 Lista PAR

Existem processos que, por pertencerem a utentes que são, por exemplo, figuras públicas (artistas, políticos, apresentadores, etc), e que por isso têm maior probabilidade de serem alvo de possíveis consultas não autorizadas e fugas de informação, devem possuir um acesso mais restrito.

A APAV considera estes processos como fazendo parte de um conjunto especial que agrega numa lista, a que deu o nome de Lista PAR (Lista de Processos de Acesso Restrito). Apenas os utilizadores com permissão de Super Administrador podem gerir esta lista, podendo indicar quais os utentes que dela constam e quais os utilizadores da PLAGA que poderão consultar os processos desses utentes. Estes processos são excluídos de todas as listagens e apenas podem ser acedidos através de consulta direta (pelo número de processo), e com a inserção de um código, enviado através de SMS para o utilizador, caso este possua permissão de acesso.

5.6 Atendimentos

Quando um utilizador inicia um novo atendimento, é carregado o formulário que estiver atualmente definido como o ativo para o serviço pretendido. A área de preenchimento do formulário

[27] https://github.com/Seldaek/monolog

pode ser dividida em três secções (Figura 5.12):

- Menu de secções do formulário (1) — Listagem das secções visíveis; importante para permitir que o técnico possa navegar diretamente até à secção adequada, para registar determinada informação, não necessitando de preencher as secções por ordem;
- Formulário (2) — Formulário de atendimento; corresponde à área de registo da informação;
- Área de processos existentes (3) — Lista dos processos existentes e que poderão corresponder ao processo do utente.

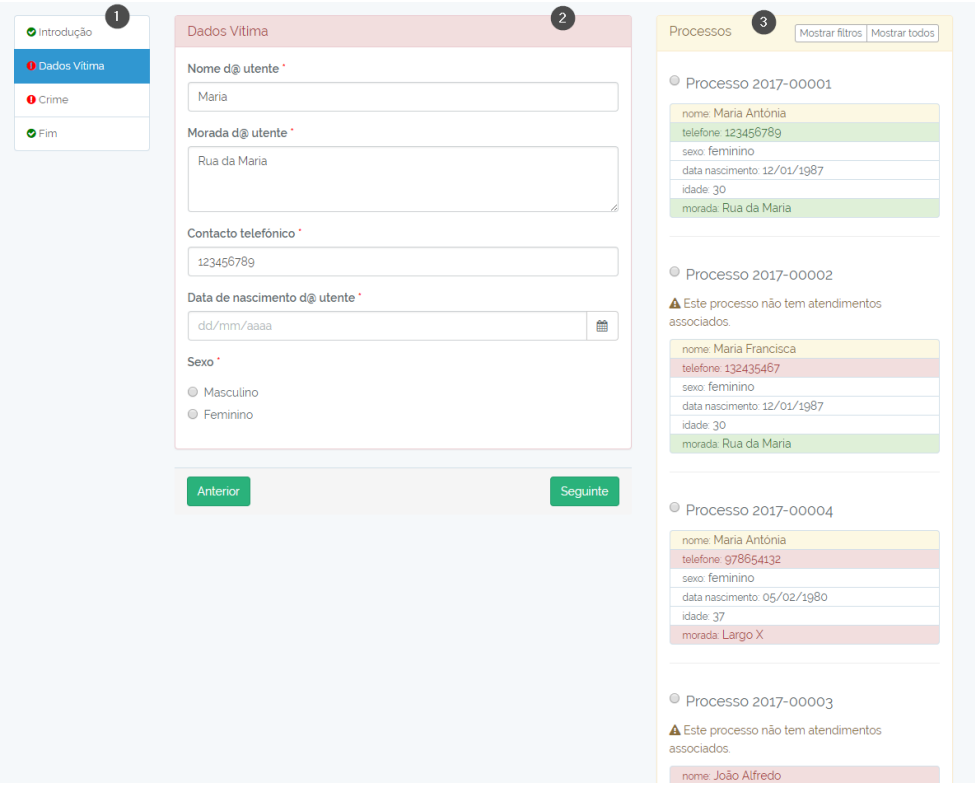


Figura 5.12: Vista de um novo atendimento

Com a lista de processos existentes, pretende-se evitar que o técnico, após realizar um atendimento, necessite de pesquisar o processo ao qual esse atendimento deve ficar anexado. No PAO, os utilizadores encontram um processo pesquisando pelo nome do utente, pelo gabinete de atendimento, data, e técnico responsável, mas nem sempre é fácil encontrar o processo pretendido (por exemplo, em casos de utentes anónimos). Na PLAGA foi implementado um algoritmo que lista os cinco processos que apresentam maior probabilidade, com base nas correspondências entre os dados inseridos no atendimento e os dados que constam no processo, de pertencerem ao utente a quem o atendimento está a ser realizado.

5.6.1 Pesquisa dos processos mais prováveis com base num atendimento

Quando um Administrador constrói um formulário, pode indicar, para um determinado campo do formulário, a que campo do processo de apoio ele corresponde, a partir de uma lista pré-definida (ver secção 5.7).

À medida que o formulário de atendimento é preenchido, e no caso de serem preenchidos campos associados aos dados dos processos de apoio, o sistema procura a informação nos processos existentes e ordena-os conforme o maior grau de probabilidade, considerando a informação registada no atendimento. Para o cálculo da probabilidade, são contabilizados *total matches*, em que a informação inserida no atendimento é igual à que consta no processo, *partial matches*, em que foi feita uma correspondência parcial de informação, ainda que não totalmente correta, e *no matches*, em que a informação do atendimento e a do processo é completamente diferente. Para o utilizador, os *total matches* são indicados a verde, os *partial matches* a amarelo e os *no matches* a vermelho. Na Figura 5.12 é ilustrada uma situação em que foram inseridos dados, no formulário de atendimento, correspondentes a campos dos processos (nome, morada e contacto telefónico do utente), e o sistema ordenou os processos existentes por maior grau de probabilidade.

Para efeitos de pesquisa dos processos, nem todos os campos de ligação atendimento-processo são tidos em conta: apenas são considerados dados "nome", "telefone", "sexo", "data de nascimento", "idade"e "morada", pois são estes que mais contribuem para a identificação do utente.

O Excerto de código 5.6 apresenta o método de pesquisa e ordenação dos processos.

```
1  function searchPossibleProcessesByFormAnswers($answers){
2
3    $searchable_data = filterSearchableData($answers);
4    $results = searchWithPartialMatches($searchable_data);
5    calculateScore($searchable_data, $results);
6    sortByScore($results);
7
8    //collect additional data ...
9
10     return $results;
11 }
```

Excerto de Código 5.6: Função de pesquisa e ordenação de processos com base em respostas a atendimentos

A função começa por filtrar as respostas registadas no atendimento pelo utilizador, de modo a considerar apenas os dados pré-definidos como os identificativos do utente. Depois, efetua a pesquisa na base de dados, utilizando o mecanismo de *full text search* do MySQL para os campos nome e morada. De forma a acelerar as pesquisas, foram criados índices *full text* nestes campos. Para todos os outros campos é feita comparação binária (*total match* ou *no match*).

O mecanismo de *full-text search* funciona utilizando a sintaxe `MATCH(col1,col2,...)`

`AGAINST (expr [search_modifier])`, em que `MATCH` aceita uma lista de nomes de colunas, separados por vírgulas, das colunas onde pesquisar, e `AGAINST` aceita o texto que se pretende encontrar e o modo de pesquisa. Existem quatro tipos de pesquisa *full-text* (ou modos) diferentes [Orag]:

- *Natural Language* — Este é o modo utilizado por defeito na *full-text search*, onde a função `MATCH` realiza uma pesquisa de um texto, segundo a perspetiva de linguagem natural, nas colunas indicadas. Para cada linha na tabela é calculado um *score*, que indica a relevância da linha, ou seja, uma medida de semelhança entre o texto a ser pesquisado e as colunas indicadas. O valor de relevância é um valor *float* não negativo, e é calculado com base na frequência da ocorrência das palavras do texto pesquisado nas colunas [Orac];

- *Full-text Stopwords* — É também um modo de pesquisa em linguagem natural, mas tem em conta uma lista pré-definida de palavras, que devem ser ignoradas por não representarem valor em termos semânticos (proposições e determinantes, por exemplo) [Oraf];

- *Boolean* — Neste modo é utilizado um conjunto de regras, sob a forma de operadores, aplicado ao texto a pesquisar. Por exemplo, o operador '+' indica que determinada palavra deve estar presente no resultado, enquanto que o operador '-' indica que a palavra não pode constar do resultado. O valor de relevância é calculado com base no algoritmo TD-IDF, que utiliza um sistema de pesos, considerando a frequência em que uma determinada palavra aparece numa linha e a frequência em que a palavra aparece em todas as linhas, para atribuir valores de *score* [Orad].

- *Search with Query Expansion* — Neste modo, primeiro é realizada uma pesquisa em linguagem natural; depois, as palavras retornadas e ordenadas por maior relevância são adicionadas ao texto a ser pesquisado, e é feita novamente outra pesquisa em linguagem natural. Este modo é útil em situações em que o utilizador pesquisa, por exemplo, "*database*", mas o que quer realmente encontrar é "MySQL"ou "Oracle", ou seja, o texto de pesquisa não é explícito o suficiente porque há uma relação semântica implícita (e que o mecanismo de pesquisa não deteta). Assim, no exemplo apresentado, a primeira pesquisa a partir de "*database*" iria encontrar documentos com as palavras "*database*" e "MySQL"; a segunda pesquisa já incluiria estas palavras, o que resulta em encontrar documentos que incluam a palavra "MySQL"mesmo que não contenham a palavra "*database*" [Orae].

Para a pesquisa dos processos, foram conjugados os modos de pesquisa *Boolean* e *Natural Language* com um sistema de pesos, de modo a controlar qual a importância que determinado *score* tem no resultado final (por exemplo, o valor do resultado com o operador '+', que indica que uma palavra tem de estar no resultado final, é mais relevante do que o resultado de pesquisa aproximada pela *Natural Language*).

Depois da seleção dos processos possíveis, é calculado um valor de *score* final, com base no número de *full matches*, *partial matches* e *no matches* dos campos de cada formulário. O algoritmo utilizado neste cálculo é indicado em pseudo-código no Algoritmo 1.

Algoritmo 1: Algoritmo em pseudo-código de cálculo do *score* de cada processo para posterior ordenação

```
1  function calculateScore;
2  foreach possible processes do
3      score_name = 0;
4      score_addr = 0;
5      full_matches_count = 0;
6      partial_matches = []; no_matches_count = 0;
7      if has name to evaluate then
8          add name to partial matches;
9          score_name = previously calculated score_name;
10     end
11     if has address to evaluate then
12         add address to partial matches;
13         score_addr = previously calculated score_addr;
14     end
15     foreach proccess searchable data do
16         if data is not in partial matches then
17             if data is full match then
18                 increment full_matches_count;
19             end
20             else if data is no match then
21                 increment no_matches_count;
22             end
23         end
24     end
25     total_score = full_matches_count + count(partial_matches) - no_matches_count +
         score_name + score_addr
26 end
```

A relevância de cada processo aumenta com o aumento do número de *matches*, e diminui com o aumento do número de *no matches* encontrados. No caso de existirem valores de *score* provenientes da *full-text search*, estes são adicionados. O valor de *score* final é utilizado para ordenar a lista de processos e apresentar os cinco processos mais prováveis.

O algoritmo de cálculo de *score* indicado apenas tem em conta o número de *matches* de cada tipo. Poderá ser interessante aplicar um sistema de pesos a cada campo, de modo a priorizar os mais relevantes (por exemplo, um *total match* do campo "sexo" poderá não ser tão importante como um *total match* do campo "telefone").

Para além do mecanismo de pesquisa *full-text* do MySQL, foram analisadas outras soluções possíveis: *Sphinx*[28], *Algolia*[29] e *ElasticSearch*[30]. O resultado dessa análise é apresentado na Tabela 5.2.

Tabela 5.2: Comparação entre soluções de pesquisa *full-text*

	Sphinx	**Algolia**	**ElasticSearch**
Modo de instalação	Pacote de instalação	Serviço	Pacote de instalação Serviço
Local de armazenamento	Local	Remoto	Local (pacote instalação) Remoto (serviço)
Gratuito	✓	✗	✓ se local ✗ se remoto
Open source	✓	✗	✓
Popularidade [31]	Média	Alta	Muito alta
Funcionalidades principais	- Indexação da base de dados SQL - Extensões para suporte da sintaxe SQL - Alto nível de customização e controlo	- Tolerância a falhas do utilizador na escrita - Estatísticas de pesquisa - Muito fácil de implementar/integrar - Suporte de várias línguas (incluindo PT) - Suporte de sinónimos - Obtenção de resultados muito rápida	- RESTful API e JSON - Comunidade ativa - Obtenção de resultados muito rápida - Extensões de suporte de deteção de anomalias utilizando {machine learning}, análise de performance, entre outros

A opção pela utilização da funcionalidade de pesquisa *full-text* do MySQL foi tomada, considerando o volume baixo de dados pesquisáveis (ordem dos 15.000), o tempo disponível para desenvolvimento, o preço dos diferentes serviços, e o facto de a aplicação ser armazenada em *shared hosting*, o que não permite instalação dos pacotes das soluções. Futuramente poder-se-á implementar outra solução de pesquisa, conforme as necessidades.

De modo a diminuir o âmbito da pesquisa e a acelerar a obtenção de resultados, são disponibilizados filtros por gabinete, data, tipo de utente (vítima, autor do crime, outro) e técnico responsável, à semelhança do que já existe no PAO atual.

5.6.2 Relação com atendimentos anteriores

Quando o utilizador seleciona um dos processos apresentados na listagem, o sistema oculta os restantes processos sugeridos, e passa a considerar apenas o processo selecionado. A partir daí, o sistema procura, em todos os atendimentos associados a esse processo, qual o atendimento mais recente que contém determinado campo respondido, e apresenta essa listagem, de modo a

[28] http://sphinxsearch.com
[29] https://www.algolia.com/
[30] https://www.elastic.co/

que o utilizador possa visualizar as respostas existentes, e utilizá-las no novo atendimento, não necessitando de voltar a colocar a questão ao utente (Figura 5.13).

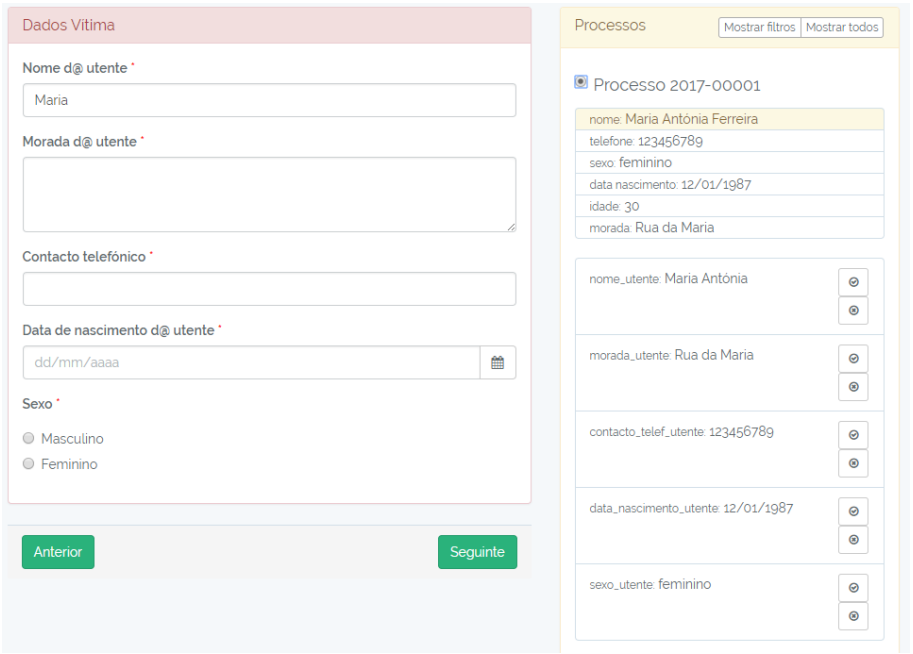


Figura 5.13: Listagem das respostas de atendimentos anteriores de um dado processo de apoio

A ligação entre os dados do novo atendimento e dos atendimentos anteriores é feita através do código dos campos e da secção. Para cada secção apresentada, o sistema procura atendimentos anteriores que contenham a mesma secção, e nessa secção os campos de igual código aos que existem na secção apresentada. A listagem de respostas resultado pode não conter todas as respostas provenientes do atendimento anterior mais recente, mas apresenta sempre as respostas mais recentes existentes.

No formulário, podem existir campos de resposta obrigatória. Cada secção com campos obrigatórios não respondidos é apresentada a vermelho; caso contrário, a verde. O utilizador pode navegar entre secções livremente, mas apenas pode submeter o novo atendimento depois de ter respondido a todos os campos obrigatórios.

Quando o técnico termina o atendimento, pode escolher se o quer associar ao processo selecionado (caso tenha selecionado algum), ou deixar o atendimento sem processo associado. Neste caso, o atendimento ficará registado numa lista de atendimentos sem processos, e poderá, mais tarde, ser associado a um processo. Atendimentos associados a processos nesta fase, poderão também ser depois reassociados a processos diferentes (para casos em que, por exemplo, o técnico se engana a fazer a associação durante a realização do atendimento).

5.7 Processos de apoio

A estrutura de fichas, fixa e comum a todos os tipos de crime, é uma das questões que, no PAO, dificulta e limita a manipulação da informação que consta nos processos apoio. Assim, procedeu-se a uma alteração da estrutura de organização: na PLAGA, os processos de apoio (Figura 5.14) passam a ser constituídos por:

- Dados Gerais — Uma ficha de dados gerais estática, onde constam os dados que são comuns a todos os processos, independentemente do tipo de vitimação, como os dados da vítima, crime(s) cometido(s), dados dos autores do(s) crime(s), apoio prestado. É nesta ficha que são também apresentados os processos relacionados, caso existam;

- Atendimentos — Listagem de todos os atendimentos realizados e anexados ao processo; cada atendimento consiste no formulário de atendimento que foi preenchido, e por isso apresenta toda a informação necessária e útil, de acordo com o tipo de vitimação.

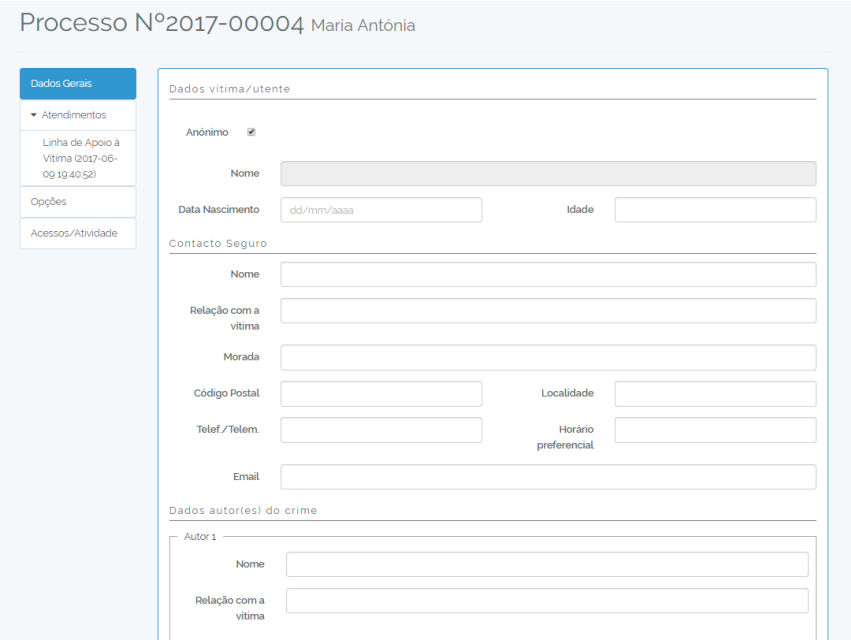


Figura 5.14: Vista do processo de apoio de um utente

A ficha de dados gerais pretende dar uma visão geral e rápida da situação do utente a que o processo diz respeito. Com a listagem de atendimentos efetuados pode-se consultar a informação completa e, ao mesmo tempo, manter facilmente um histórico de atendimentos. Em cada atendimento, na listagem, é apresentado o serviço no qual foi realizado, e a data. Na vista de consulta é indicada também a duração do atendimento e o técnico responsável.

São ainda disponibilizados separadores de opções, como "Fechar processo", ou "Encaminhar processo", e de indicação da atividade dos utilizadores, relacionada com a consulta e edição do

processo (visível apenas para administradores). Futuramente será também implementada uma funcionalidade de reposição de dados das diferentes edições.

Será ainda disponibilizada uma área para gerar documentos PDF, com *templates* pré-definidos, parcialmente preenchidos com os dados do utente.

5.7.1 Ligação entre atendimentos e processos de apoio

Durante a criação dos formulários de atendimento, os utilizadores indicam, em cada campo, se a informação que será nele registada deve ser guardada em algum campo da folha de dados gerais do processo, ao qual o atendimento será anexado. Os utilizadores definem essa ligação, selecionando os campos de processo disponíveis a partir de uma listagem Código Campo PAO. Por exemplo, num campo de formulário de código nome e código de campo PAO nome_vitima_direta, a informação registada no campo Nome no atendimento será depois, no processo, apresentada no campo Nome da secção "Dados da vítima/utente". Este mapeamento é importante, não só para evitar que o técnico tenha de preencher manualmente os campos dos dados gerais do processo, mas também para permitir a criação/edição de entradas de tabelas relacionais, como victim, ou crime_author, que facilitam posteriores consultas de informação.

Quando o atendimento é submetido e anexado a um processo, o sistema indica os campos do processo que podem ser preenchidos/atualizados com a nova informação, proveniente desse atendimento, e o valor atual desses campos; o utilizador escolhe depois quais é que devem passar para os dados gerais. Atualmente este mecanismo ainda não está implementado e faz parte do trabalho futuro a realizar.

5.7.2 Análise de risco

A análise de risco consiste numa classificação da situação da vítima, de forma a perceber que tipo de apoio ela deve receber, e que medidas devem ser tomadas por parte dos técnicos.

O grau de risco é um valor numérico, inteiro, e é calculado com base nas respostas da vítima a um formulário pré-definido, onde cada questão tem um valor associado, que é depois somado ou subtraído com base numa dada fórmula. O preenchimento deste formulário é feito separadamente do atendimento, e será disponibilizado em cada processo.

5.8 Estatística

O módulo de estatística não foi desenvolvido neste protótipo. Este módulo irá disponibilizar opções de exportação de dados no formato CSV, à semelhança do que já acontece no PAO atual. Poderá também apresentar algumas opções de consultas diretas na plataforma, conforme as necessidades. Para além da extração da informação da PLAGA, pretende-se que o sistema indique, de forma clara:

1. Quantos processos novos foram abertos num GAV por dia, semana e mês;

2. Quantos atendimentos foram realizados num GAV por dia, semana e mês;
3. Quantos atendimentos foram realizados por processo num GAV;
4. Quantos processos em acompanhamento existem num GAV.

5.9 Área pessoal do utilizador

A área pessoal do utilizador (Figura 5.15) reúne um conjunto de ferramentas para auxiliar o trabalho individual, dentro da instituição, de cada colaborador.

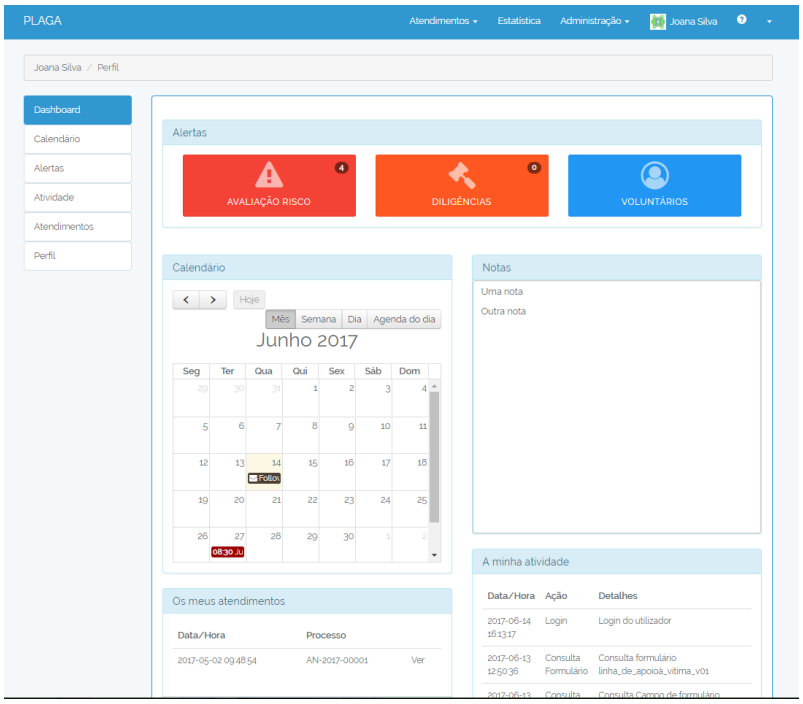


Figura 5.15: *Dashboard* da área pessoal do utilizador

Uma das dificuldades com que atualmente os colaboradores da APAV se deparam, é a gestão de todas as tarefas que têm de realizar e eventos a que têm de comparecer. Há relatos de colaboradores que afirmam utilizar até três agendas diferentes para registar tudo o que precisam. Para facilitar essa tarefa, é disponibilizado um calendário onde os utilizadores podem gerir os seus eventos. É possível configurar os eventos, de forma a que seja enviado um email a relembrar o evento no dia marcado, e é também possível indicar os utilizadores associados ao evento, de forma a que este apareça também no seu calendário (no caso de eventos com várias pessoas, como reuniões). O utilizador pode escolher um tipo de evento pré-definido, ou com título livre (Figura 5.16). A cada tipo de evento é associada uma cor, de modo a tornar mais fácil e rápida a leitura do calendário. Para a apresentação da interface do calendário foi utilizada a biblioteca JavaScript FullCalendar [32].

[32]https://fullcalendar.io/

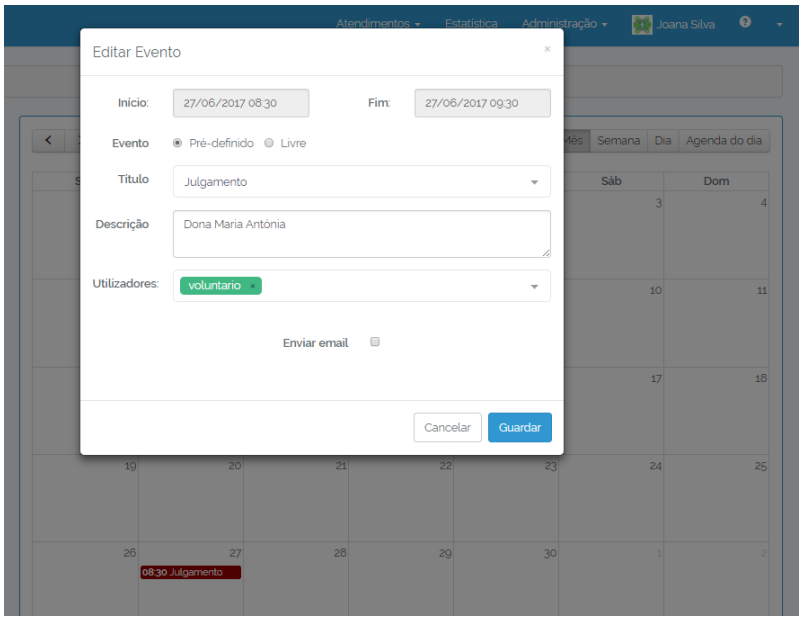


Figura 5.16: Edição de eventos de calendário da área pessoal do utilizador

Para além dos eventos de calendário, existem listagens de alertas automáticos, criados pelo sistema e organizados em categorias, para as seguintes situações:

- Avaliação/reavaliação de risco periódica não realizada;
- Diligência não realizada;
- Processo, com um risco associado classificado como extremo, sem nenhuma intervenção por 30 ou mais dias;
- Processo com determinados campos não preenchidos.

O sistema de alertas não foi implementado neste protótipo.

Para além do calendário e dos alertas, o utilizador tem também disponível uma listagem dos atendimentos a que está associado, pode consultar o registo da sua atividade na plataforma, e ainda registar notas num editor de texto.

5.10 Sumário

Neste capítulo foram apresentados os detalhes de desenho e implementação da Plataforma de Gestão de Atendimentos.

A utilização de *frameworks* e bibliotecas permite um desenvolvimento mais rápido das aplicações. Devido à alta taxa de utilização de dispositivos móveis, hoje em dia são várias as ferramentas disponíveis que permitem o desenvolvimento de *layouts* responsivos, como é o caso da Bootstrap, Foundation, Semantic UI e Avalanche. Para auxiliar o desenvolvimento de interfaces dinâmicas,

de forma aumentar a eficiência e melhorar a estruturação do código, podem ser utilizadas ferramentas como AngularJS, ReactJS e VueJS. A nível de *back-end*, as *frameworks* CodeIgniter e Laravel são as mais utilizadas com PHP, sendo a Laravel mais moderna, tendo granjeado grande popularidade na comunidade. As tecnologias utilizadas no desenvolvimento deste projeto foram escolhidas com base numa análise destas *frameworks*.

Para dar resposta às necessidades da APAV, o sistema foi idealizado em cinco módulos principais:

1. Administração — Funcionalidades de gestão dos formulários de atendimento e monitorização da atividade dos utilizadores;
2. Atendimentos — Área de dedicada aos novos atendimentos, onde os utilizadores podem preencher os formulários de atendimento de um dado serviço;
3. Processos de apoio — Funcionalidades relacionadas com os processos de apoio das vítimas;
4. Estatística — Funcionalidades de exportação e consulta dos dados da plataforma, para efeitos estatísticos;
5. Área pessoal do utilizador — Funcionalidades úteis a todos os utilizadores, como agendamento de eventos/alertas.

O desenvolvimento do primeiro protótipo funcional focou-se mais na área de gestão de formulários, criação de novos atendimentos e na área pessoal do utilizador. Apesar de ter sido abordado e planeado, o módulo de Estatística não foi implementado.

Foi implementado um repositório de campos de formulário, onde os utilizadores podem gerir campos para utilizar nos formulários. Inicialmente a especificação dos formulários e dos campos era feita através da edição de uma estrutura em YAML, mas posteriormente foi implementada uma interface gráfica, mais vantajosa tanto a nível de facilidade de utilização, como de validação do *input* do utilizador. Os formulários são dinâmicos, sendo possível especificar regras de visibilidade para cada campo, consoante as respostas registadas noutros campos, e quais as secções do formulário visíveis.

No preenchimento de novos atendimentos, os utilizadores têm acesso a um sistema de pesquisa e ordenação automática de processos existentes, mais prováveis de pertencerem ao utente a quem está a ser realizado o atendimento. O objetivo é evitar que os técnicos necessitem de pesquisar o processo para anexar o atendimento *a posteriori*, como acontece atualmente.

Na área do utilizador, é disponibilizado um calendário onde os utilizadores podem agendar eventos, uma listagem de alertas automáticos para determinadas situações pré-definidas, e ainda um editor para notas.

Foi ainda apresentada a nova estrutura proposta de processos, que apresenta grandes diferenças comparativamente à do PAO atual. A nova estrutura permite manter um histórico dos atendimentos realizados, e é mais legível e adequada aos diferentes tipos de crime.

Capítulo 6

Validação

Neste capítulo são apresentados os testes de usabilidade conduzidos junto dos futuros utilizadores da PLAGA, de forma a poder validar o trabalho desenvolvido, percebendo se o sistema vai de encontro às necessidades dos utilizadores, e de que forma será possível melhorá-lo.

6.1 Metodologia

Foi elaborado um plano de validação para testar os três componentes principais desenvolvidos: gestão de formulários e campos de formulário, área pessoal do utilizador e criação e consulta de atendimentos. O plano é constituído por diversos cenários de teste para cada área, entendendo-se por cenário de teste um conjunto de tarefas a serem realizadas pelo utilizador [Ass13b]. De forma a poder posteriormente comparar o percurso escolhido pelos utilizadores para completarem o cenário com o percurso ideal, este último foi também indicado. O plano de validação é apresentado no Anexo A.

Os testes foram realizados nos serviços de gabinete da APAV no Porto, e cada sessão durou aproximadamente 30 minutos. No início de cada módulo foi feita uma pequena introdução aos utilizadores sobre as funcionalidades que iriam ser testadas, embora sem grandes detalhes, pois pretendia-se que os testes fossem realizados da forma mais intuitiva e independente possível (ver Anexo A). Após cada utilizador completar os cenários, foi-lhes pedido que respondessem a questionários relativos às áreas testadas, tendo-se procedido depois à análise das respostas dadas.

A interação dos utilizadores com o sistema foi captada com recurso ao Camtasia Recorder [1]. Com base na observação desta interação, foram recolhidos dados relativos a métricas de avaliação que permitem analisar a taxa de sucesso/insucesso na realização de cada cenário. Os questionários e as métricas de avaliação foram definidos com base nas recomendações indicadas na plataforma *usability.gov*, cujo conteúdo é gerido pela Divisão de Comunicação Social do Departamento de Saúde e Serviços Humanos dos Estados Unidos da América [Ass16a].

[1] http://discover.techsmith.com/camtasia-brand-desktop/

6.2 Métricas de avaliação

Nesta secção são indicadas as métricas tidas em conta para a formulação dos cenários de validação e dos questionários.

6.2.1 Métricas quantitativas

As métricas quantitativas referem-se a dados mensuráveis, como duração de execução das tarefas do cenário ou taxa de sucesso [Ass13a]. Para este plano de validação optou-se pelos seguintes parâmetros:

- Cenário Completo — Um cenário de teste é considerado completo se o utilizador consegue com sucesso efetuar as tarefas pedidas nesse cenário;
- Cenário com erro não crítico — Erros não críticos são desvios do percurso preferencial na realização do cenário, mas que não impedem o utilizador de o completar com sucesso; por exemplo, comportamentos exploratórios do sistema, como carregar na opção do menu errada, são considerados erros não críticos;
- Cenários com erro crítico — Considera-se que existe um erro crítico no caso de o utilizador não conseguir completar o cenário;

6.2.2 Métricas subjetivas

As métricas subjetivas incluem questões relacionadas com o grau de satisfação dos utilizadores enquanto utilizam o sistema, pontos positivos, pontos negativos, recomendações e *feedback* relativo à facilidade de utilização [Ass13a]. Os dados referentes a estas métricas foram recolhidos através dos questionários realizados aos utilizadores após completarem os cenários de teste.

As perguntas dos questionários foram elaboradas de forma a avaliar os seguintes pontos principais:

- O grau de satisfação do utilizador enquanto trabalhava na plataforma;
- A facilidade do utilizador em trabalhar na plataforma;
- Se o utilizador considerou a plataforma intuitiva;
- A comparação entre o novo sistema e o sistema PAO;
- O que é que o utilizador gostou mais, o que gostou menos e sugestões de alterações;

Os questionários completos podem ser consultados no Anexo B.

6.3 Análise dos resultados

Nesta secção são analisados os resultados dos testes de validação. As respostas dos utilizadores podem, à semelhança dos questionários, ser consultadas no Anexo B.

6.3.1 Características dos participantes

Para os casos de teste definidos foram considerados dois papéis: administrador, com permissões de gestão dos formulários e campos de formulário, e voluntário, com permissões de criação e consulta dos atendimentos e processos de apoio. Tanto o administrador como o voluntário têm permissão para aceder à área pessoal do utilizador.

Estiveram disponíveis sete participantes, todos colaboradores da APAV. Tendo em conta as funções que desempenham na associação, cinco pessoas utilizaram a plataforma na qualidade de voluntários e duas na qualidade de administrador.

Todos os participantes são atualmente utilizadores do sistema PAO. Dos que estão responsáveis pelos atendimentos, um deles faz atendimentos na receção, e os restantes em gabinete.

Assim, foram recolhidas duas respostas ao formulário relativo à gestão de formulários, cinco respostas ao formulário sobre os atendimentos e sete respostas ao formulário sobre a área pessoal do utilizador.

6.3.2 Análise quantitativa

Nesta secção são apresentados e analisados os resultados dos testes de validação. Os diferentes cenários numerados no plano de validação são identificadas nos gráficos por C<N>, em que C é o número do cenário. Os gráficos apresentados nesta secção, nas Figuras 6.1, 6.2 e 6.3, apresentam, para cada cenário, o número de utilizadores que o completaram com sucesso, com erro não crítico ou que não o completaram (erro crítico).

Administração de formulários e campos de formulário Os resultados relativos a esta área estão apresentados na Figura 6.1. Os utilizadores conseguiram realizar a maior parte dos cenários de validação de gestão dos formulários de forma independente e com facilidade, tendo apresentado algumas dificuldades nas tarefas relacionadas com a edição dos formulários (C6, C7, C8). O cenário que se revelou mais complicado foi o C6, onde era pedido que fosse adicionada uma nova secção ao formulário, e um novo campo a essa secção. Apesar disso, todos os cenários foram completados, não tendo sido registados erros críticos.

Área de atendimentos e processos Os resultados relativos a esta área estão apresentados na Figura 6.2. A maioria dos utilizadores conseguiu realizar os cenários apresentados sem erros, com exceção das tarefas do cenário C4, onde era pedido que fossem preenchidos campos do novo atendimento com as respostas indicadas num atendimento anterior. Isto deveu-se ao facto de a ligação entre os campos do formulário do novo atendimento e do atendimento anterior ser feita através do código da pergunta, o que dificultou a identificação dos campos corretos; para além disso, os ícones utilizados não foram suficientemente intuitivos para a realização do cenário em questão. Também foram demonstradas algumas dificuldades no cenário C6, uma vez que os utilizadores não conseguiram deduzir facilmente o caminho até à nova página que deviam visitar. Não houve ocorrência de erros críticos.

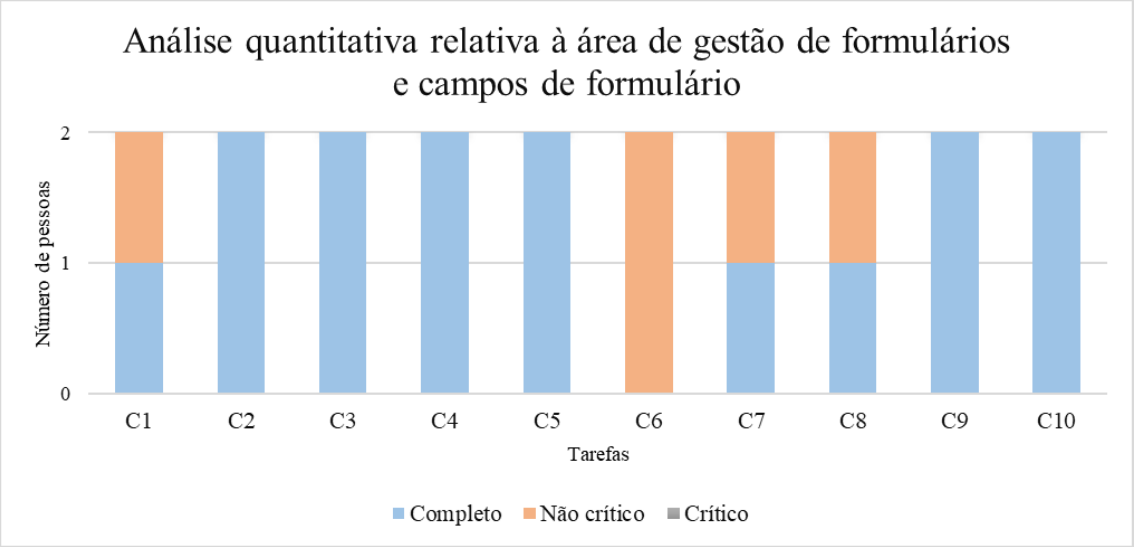


Figura 6.1: Resultados quantitativos relativos à área de gestão de formulários e de campos de formulário

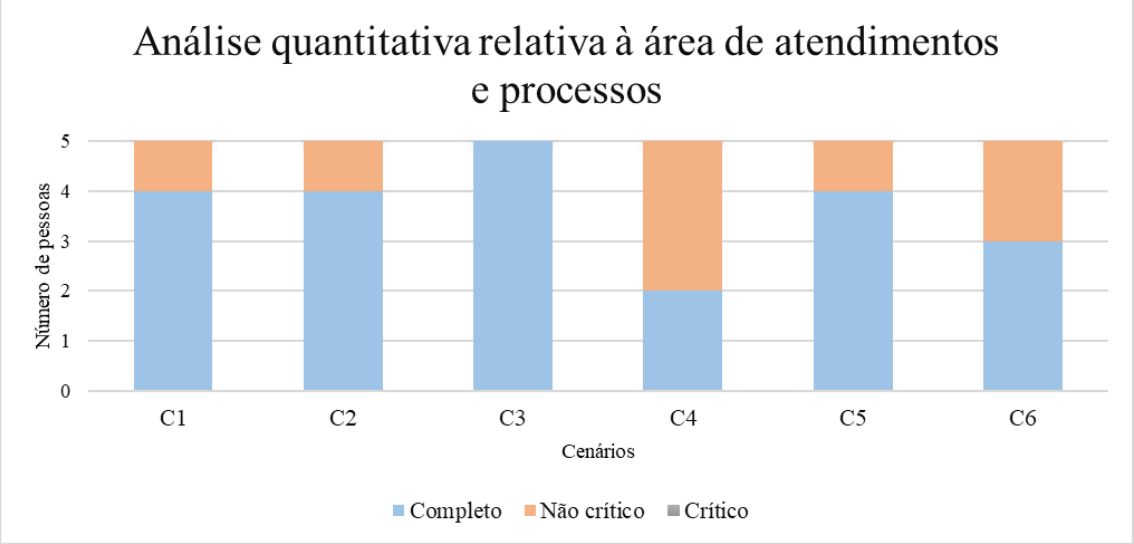


Figura 6.2: Resultados quantitativos relativos à área de gestão de formulários e de campos de formulário

Área pessoal do utilizador Os resultados relativos a esta área estão apresentados na Figura 6.3. O cenário onde os utilizadores tiveram mais dificuldade foi o C2, onde era pedido que fossem alteradas a hora e duração de um evento de calendário. A dificuldade deveu-se ao facto de o sistema não permitir a edição manual da hora e data dos eventos, e requerer que o utilizador mova e redimensione o evento de modo a obter a data/hora de início e a duração desejadas.

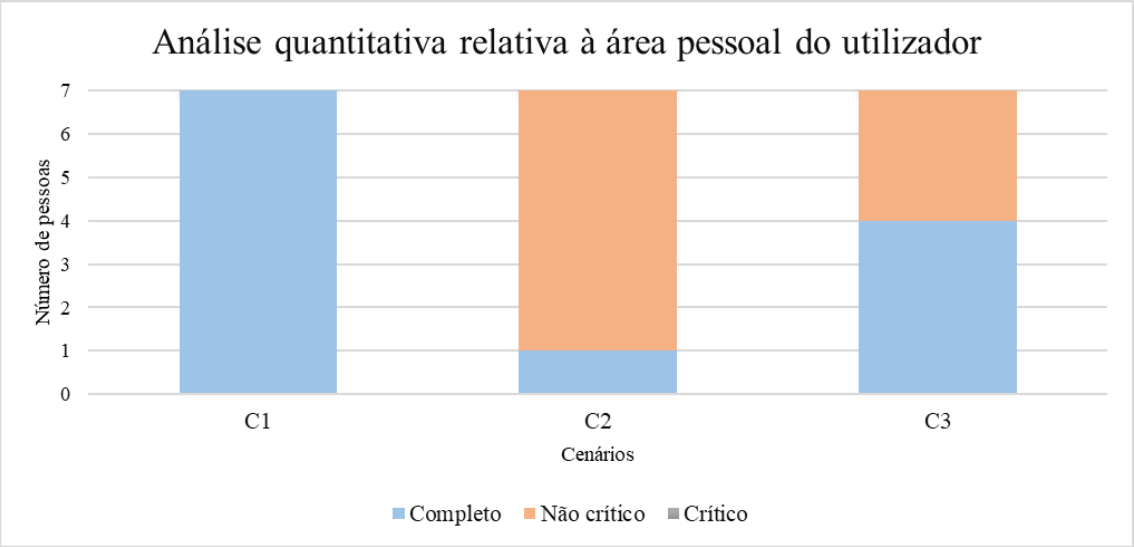


Figura 6.3: Resultados quantitativos relativos à área pessoal do utilizador

6.3.3 Análise qualitativa

A partir das respostas dos utilizadores aos questionários, efetuou-se a respetiva análise. Para cada área de estudo, foi pedido aos utilizadores que classificassem o grau de facilidade e satisfação que sentiram durante a realização das tarefas pedidas numa escala de 1 a 5, onde 1 significa que o utilizador discorda totalmente com a afirmação, e 5 significa que o utilizador concorda totalmente com a afirmação. De forma a analisar mais facilmente os resultados, foram elaborados gráficos com as médias das classificações para cada afirmação. Estes gráficos são apresentados nas Figuras 6.4, 6.5 e 6.6.

Administração de formulários e campos de formulário Os resultados relativos a esta área estão apresentados na Figura 6.4. Os utilizadores sentiram-se muito confortáveis e satisfeitos ao utilizar a aplicação, embora tenha existido uma maior dificuldade de utilização na gestão dos formulários em relação aos campos de formulários. Esta dificuldade é compreensível, pois a edição de formulários envolve mais componentes do que a edição de campos, e é por si só uma tarefa mais complexa.

Ambos os utilizadores apontaram a interface intuitiva e agradável como ponto positivo. O facto de não ser necessário conhecimentos de programação para a criação dos formulários foi também indicado como sendo uma mais valia.

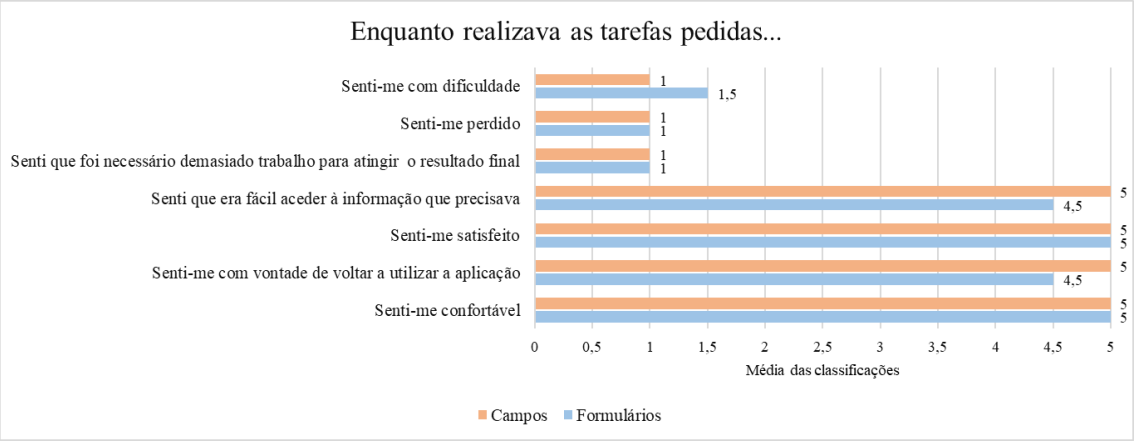


Figura 6.4: Resultados qualitativos relativamente ao grau de facilidade de utilização e satisfação dos utilizadores na área de gestão de formulários e campos de formulário

Foi sugerido que os códigos dos campos de formulário fossem omitidos quando se faz a importação dos campos no formulário, e em vez deles se utilizasse o título do campo.

Área de atendimentos e processos Os resultados relativos a esta área estão apresentados na Figura 6.5. O grau de satisfação dos utilizadores foi muito elevado; apesar de terem sentido alguma dificuldade em utilizar a plataforma, indicaram que a conseguiriam utilizar bem após um período de ambientação.

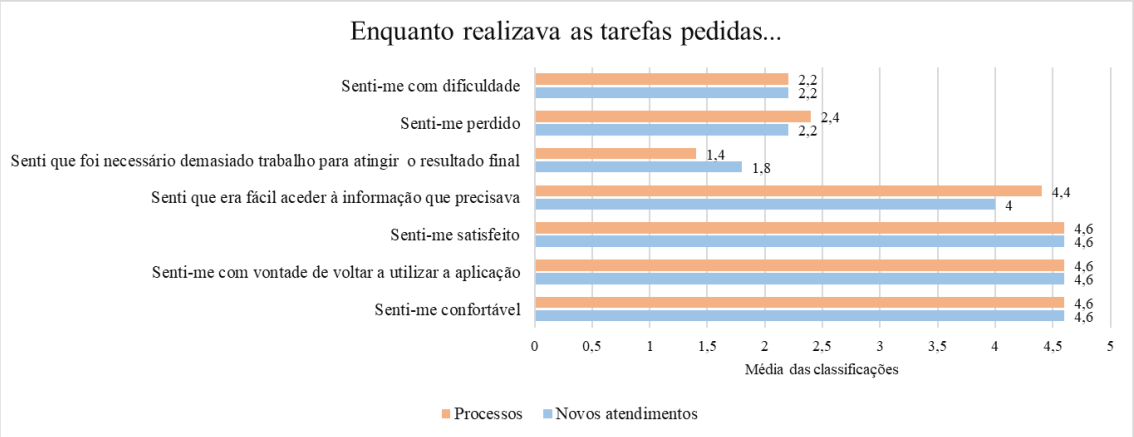


Figura 6.5: Resultados qualitativos relativamente ao grau de facilidade de utilização e satisfação dos utilizadores na área de atendimentos e processos

Nos questionários, os utilizadores indicaram ainda que, comparativamente ao sistema PAO atual, a aplicação PLAGA é muito menos confusa, mais agradável, útil, organizada, e permite aceder a uma maior quantidade de informação.

As funcionalidades salientadas como mais positivas foram a indicação automática de processos existentes com base nas respostas dadas nos novos atendimentos (e possível associação dos

atendimentos a estes processos), e a possibilidade de preencher automaticamente a informação nos novos atendimentos com base em respostas anteriores.

Relativamente aos pontos que requerem melhoria, os utilizadores indicaram as cores utilizadas e a interface do mecanismo de ligação da informação dos atendimentos anteriores aos novos.

Foi sugerido que, no caso de atendimentos subsequentes de um processo, o formulário do novo atendimento fosse logo preenchido com as respostas recolhidas em atendimentos anteriores.

Área pessoal do utilizador Os resultados relativos a esta área estão apresentados na Figura 6.6. O grau de satisfação dos utilizadores foi elevado, e no geral não sentiram muita dificuldade em utilizar o sistema.

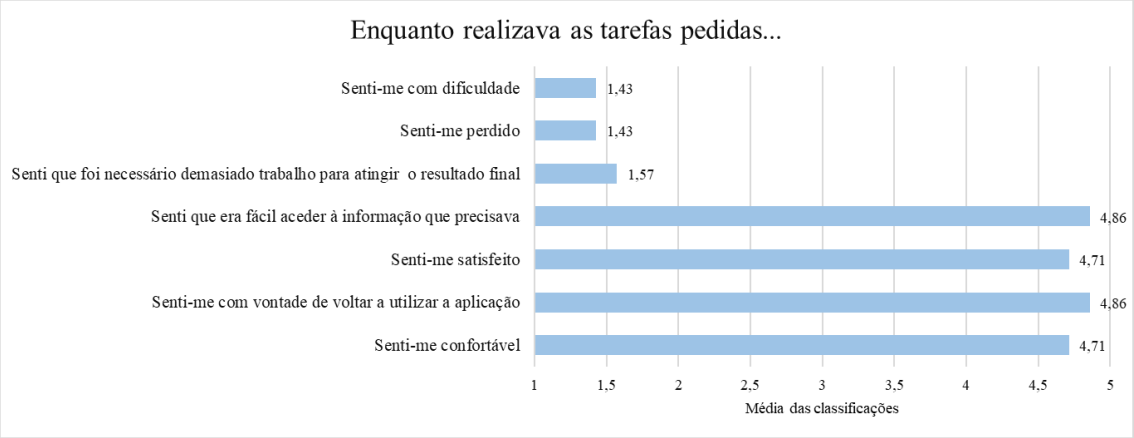


Figura 6.6: Resultados qualitativos relativamente ao grau de facilidade de utilização e satisfação dos utilizadores na área pessoal do utilizador

Foram realçados como aspetos positivos a possibilidade de agendamento de eventos e a existência de alertas [2]. Como pontos pontos negativos, os utilizadores mencionaram o facto de apenas ser possível alterar a data e hora dos eventos arrastando e redimensionando o evento no calendário.

6.4 Sumário

Os resultados analisados na Secção 6.3.2 permitiram concluir que, no geral, a aplicação PLAGA apresenta uma interface intuitiva para os utilizadores, que não demonstraram grandes dificuldades em realizar as tarefas pedidas nos cenários de validação.

O *feedback* dos utilizadores, analisado na Secção 6.3.3, foi muito positivo. Todos indicaram que o sistema PLAGA irá ser muito mais útil e agradável de utilizar do que o atual sistema PAO. Mostraram-se todos entusiasmados e com vontade de o voltar a utilizar.

Os questionários permitiram recolher também sugestões de melhorias da plataforma que serão tidas em conta no futuro.

[2] Apesar de esta funcionalidade não ter sido implementada, foi construído e incorporado na página um *layout* estático, para efeitos de demonstração

Capítulo 7

Conclusões e trabalho futuro

O atendimento à vítima na APAV é um processo complexo, que envolve diversos serviços e recursos, em diferentes pontos do país. Os sistemas informáticos que a APAV tem atualmente disponíveis para suporte a este processo, não satisfazem as necessidades da instituição e apresentam problemas de funcionamento e interoperabilidade que exigem aos colaboradores uma grande carga de trabalho extra de correção de erros e registo repetido de informação.

Nesta dissertação é apresentada a nova aplicação *web*, a Plataforma de Gestão de Atendimentos (PLAGA), que irá, no futuro, substituir os sistemas atuais, providenciando um armazenamento centralizado da informação, maior controlo de gestão, e automatização de determinadas tarefas.

Foi necessário conhecer bem o domínio do problema e compreender a forma como a APAV atualmente trabalha, para poder desenhar um novo sistema que de facto fosse de encontro às necessidades da instituição, e pudesse apoiar verdadeiramente o trabalho dos colaboradores. Um dos objetivos principais era poder permitir a gestão dos formulários de atendimento por parte dos administradores da APAV, ou seja, implementar uma forma de os administradores criarem e editarem formulários dinâmicos que possam ser utilizados pelos técnicos de apoio nos atendimentos às vítimas, nas muito diversas tipificações dos vários crimes tratados.

A implementação de funcionalidades de gestão pelo utilizador de formulários dinâmicos tem de garantir que os dados armazenados se mantêm consistentes, independentemente das alterações dos formulários. Algumas das soluções tipicamente utilizadas, como a GoogleForms, revelaram apresentar problemas a esse nível, embora normalmente nesses casos, as ferramentas sejam apenas utilizadas para a construção de questionários pontuais simples. No caso da APAV, possíveis inconsistências podem ter um impacto negativo alto, tanto a nível do trabalho interno da instituição, como a nível da produção dos relatórios estatísticos, que são fonte de informação para diversas entidades externas. Foram estudadas diferentes abordagens possíveis a esta implementação, desde soluções de modelação relacional [MMNL14, HT01] a modelos não relacionais [KPZ12, GHS05]. A abordagem definida para este projeto passou pela construção de um modelo híbrido, em que as

estruturas dos formulários e dos campos de formulário são definidas como objetos JSON para permitir maior flexibilidade, mas onde determinados campos, relativos a "dados *core*"(por exemplo, tipos de crime), são alimentados por dados armazenados em tabelas relacionais. A consistência da informação é também garantida através de um sistema de versões de edição dos formulários e dos campos.

Apesar de existirem estudos na literatura sobre a implementação de formulários dinâmicos geridos pelo utilizador, este projeto introduz uma abordagem que difere das soluções propostas já existentes. O estudo sobre os diferentes aspetos de segurança a ter em conta no desenvolvimento de aplicações *web*, a análise de diferentes tecnologias que facilitam a implementação destas aplicações e as considerações feitas sobre a solução proposta, poderão ser úteis a quem pretender, no futuro, desenvolver sistemas com características semelhantes.

Durante o tempo de realização desta dissertação foi implementado o primeiro protótipo funcional, com maior foco nas funcionalidades relacionadas com os atendimentos, gestão de formulários e área pessoal do utilizador.

O *backend* da aplicação foi desenvolvido em PHP, com recurso à *framework* MVC Laravel, que tem vindo a ganhar grande popularidade nos últimos tempos e que disponibiliza uma boa infraestrutura base, com alguns mecanismos de segurança já implementados, e um conjunto de recursos úteis para a construção de aplicações *web*. A comunidade, muito ativa, disponibiliza também vários *packages* de extensões que podem ser integrados com a ferramenta. Para a implementação de páginas dinâmicas e *user-friendly*, recorreu-se a VueJS, uma *framework* Javascript orientada a componentes e desenhada para construir Single Page Applications. A *framework* Bootstrap foi utilizada na construção dos *layouts* para implementar uma interface responsiva, de modo a que a aplicação possa ser utilizada tanto em *desktop* como em *tablets*.

Para efeitos de validação do projeto, foram conduzidos testes de usabilidade junto de colaboradores da APAV, no Porto, onde estiveram presentes pessoas com diferentes funções dentro da instituição. Era esperado que os colaboradores oferecessem alguma resistência ao novo sistema, por ser bastante diferente daqueles que estão habituados a utilizar, mas tal não aconteceu e o *feedback* obtido foi muito positivo. Uma das funcionalidades implementadas mais importantes e valorizadas pelos utilizadores com permissão de realização de atendimentos, foi a pesquisa automática de processos existentes com base nas respostas a um atendimento. Pela parte dos administradores, a possibilidade de edição dos formulários de atendimento foi muito valorizada, pois permitirá o melhoramento contínuo do serviço de atendimento da APAV.

Como a PLAGA é ainda um primeiro protótipo funcional, está já acordado desenvolvimento futuro até a completa implementação de todos os requisitos. Existem também alguns aspetos na implementação atual que poderão ser melhorados, nomeadamente:

- Atualmente, a edição da data e hora dos eventos de calendário apenas pode ser feita através de *drag and drop* do evento, na interface do calendário. Os utilizadores indicaram que seria preferível poder alterar esses campos de forma manual, editando diretamente os valores;

- A estrutura dos formulários construída permite apenas definir, para cada secção, qual a próxima secção que deve aparecer consoante as respostas registadas. Haverá necessidade de permitir que sejam mostradas várias secções em consequência de determinadas respostas, para ser possível, por exemplo, poder responder às questões das secções relativas aos tipos de crime conforme a seleção de mais do que um tipo;

- Apenas é possível definir regras de visibilidade dos campos de formulário, numa secção, com base nas respostas dadas noutros campos da mesma secção. Este aspeto será alterado para oferecer maior flexibilidade de configuração dos formulários aos administradores;

- Uma vez que foi utilizado o código dos campos de formulário para os identificar numa secção, apenas é possível adicionar campos com códigos distintos a cada secção. Poder-se-á alterar este aspeto, de modo a poderem ser adicionados campos repetidos numa mesma secção;

- Podem existir casos em que seja útil criar grupos de campos (por exemplo, quando existem vários autores do crime, existir o grupo "Dados do autor" dentro de uma secção "Autores do crime", que possa ser adicionado várias vezes). A estrutura de formulário proposta não contempla este caso, pelo que será um ponto a alterar futuramente;

- Os testes de usabilidade permitiram perceber que a interface para a ligação das respostas dos atendimentos anteriores aos novos atendimentos necessitará de ser melhorada, pois os utilizadores tiveram algumas dificuldades em relacionar os campos através do seu código;

- É importante definir um plano de prevenção e resposta a casos de perda ou roubo dos dados. Esse plano poderá incluir o *backup* (e teste desse *backup*) dos dados de forma periódica, e funcionalidade de *reset* fácil de todas as *passwords* dos utilizadores da plataforma, no caso de estas terem sido comprometidas.

O grande objetivo deste projeto era planear uma solução informática que possibilitasse à APAV gerir de forma mais flexível, fácil e completa, o seu processo de atendimento à vítima, e implementar um primeiro protótipo funcional que permitisse testar algumas funcionalidades-chave junto dos utilizadores finais, como a criação de atendimentos, a gestão dos formulários dinâmicos de atendimento, e a gestão de eventos. Com a Plataforma de Gestão de Atendimentos, este objetivo pode ser considerado cumprido com alto grau de satisfação por parte dos colaboradores da instituição.

Para finalizar, é de realçar o impacto positivo que a realização deste projeto teve na autora, tanto pela oportunidade de adquirir e melhorar competências na área de desenvolvimento *web*, como também pelo facto de o sistema implementado representar uma ferramenta valiosa de suporte ao trabalho de uma instituição de cariz social, tão importante como é a APAV.

Os URLs indicados em nota de rodapé ao longo deste documento e na listagem de referências bibliográficas foram acedidos entre outubro de 2016 e junho de 2017.

Referências

[AKC16] Abderrahim Abdellaoui, Younes Idrissi Khamlichi e Habiba Chaoui. A Novel Strong Password Generator for Improving Cloud Authentication. *Procedia Computer Science*, 85:293–300, 2016. URL: http://search.ebscohost.com/login.aspx?direct=true{&}AuthType=ip,sso,url,uid{&}db=edselp{&}AN=S1877050916305841{&}lang=pt-br{&}site=eds-live{&}scope=site{&}authtype=sso, doi:10.1016/j.procs.2016.05.236.

[AKV15] Mazhar Ali, Samee U. Khan e Athanasios V. Vasilakos. Security in cloud computing: Opportunities and challenges. *Information Sciences*, 305:357–383, 2015. URL: http://www.sciencedirect.com/science/article/pii/S0020025515000638, doi:http://dx.doi.org/10.1016/j.ins.2015.01.025.

[Ass] Associação Portuguesa de Apoio à Vítima. APAV - Contactos. URL: http://www.apav.pt/apav_v3/index.php/pt/contactos.

[Ass13a] Assistant Secretary for Public Affairs. Planning a Usability Test. 2013. URL: https://www.usability.gov/how-to-and-tools/methods/planning-usability-testing.html.

[Ass13b] Assistant Secretary for Public Affairs. Scenarios. 2013. URL: https://www.usability.gov/how-to-and-tools/methods/scenarios.html.

[Ass14] Associação para a Promoção e Desenvolvimento da Sociedade da Informação. *O Tratamento de Dados Pessoais em Portugal*. APDSI, 2014.

[Ass16a] Assistant Secretary for Public Affairs. About Us, 2016. URL: https://www.usability.gov/about-us/index.html.

[Ass16b] Associação Portuguesa de Apoio à Vítima. Manual de procedimentos, 2016.

[Ass17a] Associação Portuguesa de Apoio à Vítima. Estatísticas APAV - Relatório anual 2016, 2017. URL: http://www.apav.pt/apav_v3/images/pdf/Estatisticas_APAV_Relatorio_Anual_2016.pdf.

[Ass17b] Associação Portuguesa de Apoio à Vítima. Estatísticas APAV - Relatório anual 2016, 2017. URL: http://www.apav.pt/apav_v3/images/pdf/Estatisticas_APAV_Relatorio_Anual_2016.pdf.

[AYKM14] Younis A. Younis, Kashif Kifayat e Madjid Merabti. An access control model for cloud computing. *Journal of Information Security and Applications*, 19:45 – 60, 2014. URL: http://search.ebscohost.com/login.

aspx?direct=true&AuthType=ip,sso,url,uid&db=edselp&AN=S2214212614000222&lang=pt-br&site=eds-live&scope=site&authtype=sso.

[Boo] Bootstrap. Migrating to v4. URL: https://v4-alpha.getbootstrap.com/migration/.

[BS01] Daniel J. Barrett e Richard E. Silverman. Algorithms used by ssh. In *SSH, The Secure Shell: The Definitive Guide*, chapter 3.9. O'Reilly, 2001.

[BSW10] Dan Boneh, Amit Sahai e Brent Waters. Functional encryption: Definitions and challenges, 2010.

[Buc17] Craig Buckler. Best JavaScript Frameworks, Libraries and Tools to use in 2017, 2017. URL: https://www.sitepoint.com/top-javascript-frameworks-libraries-tools-use/.

[BVM13] João Paulo Baptista, João Varajão e Fernando Moreira. *Função Sistemas de Informação nas organizações : realidade, desafios e oportunidades do uso de arquiteturas empresariais*, pages 155–159. Almedina, 2013. URL: http://hdl.handle.net/1822/26252.

[Cha13] Akmal B. Chaudhri, 2013. URL: https://speakerdeck.com/abchaudhri/considerations-for-using-nosql-technology-on-your-next-it-project-1.

[CKS+11] A. J. Choudhury, P. Kumar, M. Sain, H. Lim e H. Jae-Lee. A strong user authentication framework for cloud computing. In *2011 IEEE Asia-Pacific Services Computing Conference*, pages 110–115, Dec 2011. doi:10.1109/APSCC.2011.14.

[Coy16] Chris Coyier. A complete guide to flexbox, 2016. URL: https://css-tricks.com/snippets/css/a-guide-to-flexbox/.

[CR10] Jonathan Citty e Hutchings Dugald Ralph. Tapi: Touch-screen authentication using partioned images, 2010.

[Cun16] Victor Hugo Cunha. Codeigniter x laravel - prós e contras de cada framework. *rbtech*, 2016. URL: http://dev.rbtech.info/codeigniter-vs-laravel-melhor/.

[Def16] Defuse Security. Secure Salted Password Hashing - How to do it Properly, 2016. URL: https://crackstation.net/hashing-security.htm{#}attacks.

[Dig14] DigitalOcean, 2014. URL: https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models.

[DMR04] Darren Davis, Fabian Monrose e Michael K. Reiter. On user choice in graphical password schemes. *Proceedings of the 13th Usenix Security Symposium*, 2004.

[Eri16] Mark Erikson. The (R)Evolution of Web Development, 2016. URL: http://blog.isquaredsoftware.com/presentations/2016-10-revolution-of-web-dev.

[Fac17] Facebook Inc. React, 2017. URL: https://facebook.github.io/react/.

[Fre16] Tony Freed. What is virtual dom. *Cardlife Blog*, 2016. URL: https://medium.com/cardlife-app/what-is-virtual-dom-c0ec6d6a925c#.6xi9tpj0r.

[FSG+14] Diogo A. B. Fernandes, Liliana F. B. Soares, João V. Gomes, Mário M. Freire e Pedro R. M. Inácio. Security issues in cloud environments: a survey. *International Journal of Information Security*, 13(2):113–170, 2014. URL: http://dx.doi.org/10.1007/s10207-013-0208-7, doi:10.1007/s10207-013-0208-7.

[GHS05] Swapna Gupta, J. Scott Hawker e Randy K. Smith. Acquiring and delivering lessons learned for nasa scientists and engineers: A dynamic approach. In *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2*, ACM-SE 43, pages 370–375, New York, NY, USA, 2005. ACM. URL: http://doi.acm.org/10.1145/1167253.1167341, doi:10.1145/1167253.1167341.

[GHTC13] Katarina Grolinger, Wilson A. Higashino, Abhinav Tiwari e Miriam Am Capretz. Data management in cloud environments: Nosql and newsql data stores. *J. Cloud Comput.*, 2(1):49:1–49:24, December 2013. URL: http://dx.doi.org/10.1186/2192-113X-2-22, doi:10.1186/2192-113X-2-22.

[Goo] Google Inc. Angularjs. URL: https://angularjs.org/.

[Goo12] Google Chrome. Infographic: Evolution of Browsers and the Web, 2012. URL: http://www.evolutionoftheweb.com/#/evolution/day.

[Gua15] Matthew Guay. The 14 Best Online Form Builders for Every Task - The Ultimate Guide to Forms and Surveys, 2015. URL: https://zapier.com/learn/forms-surveys/best-online-form-builder-software/.

[HCC10] Min-Shiang Hwang, Song-Kong Chong e Te-Yu Chen. Dos-resistant id-based password authentication scheme using smart cards. *The Journal of Systems & Software*, 83(SI: Top Scholars):163 – 172, 2010. URL: http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,sso,url,uid&db=edselp&AN=S0164121209001952&lang=pt-br&site=eds-live&scope=site&authtype=sso.

[HDCP08] Eiji Hayashi, Rachna Dhamija, Nicolas Christin e Adrian Perrig. Use your illusion: Secure authentication usable anywhere. In *Proceedings of the 4th Symposium on Usable Privacy and Security*, SOUPS '08, pages 35–45, New York, NY, USA, 2008. ACM. URL: http://doi.acm.org/10.1145/1408664.1408670, doi:10.1145/1408664.1408670.

[HT01] Daniel J. Helm e Bruce W. Thompson. An approach for totally dynamic forms processing in web-based applications. *ICEIS*, 2001.

[KKMR16] B. Hari Krishna, S. Kiran, G. Murali e R. Pradeep Kumar Reddy. Security issues in service model of cloud computing environment. *Procedia Computer Science*, 87:246–251, 2016. URL: http://www.sciencedirect.com/science/article/pii/S1877050916304963, doi:http://dx.doi.org/10.1016/j.procs.2016.05.156.

[Kor17] Eugeniya Korotya. 5 Best JavaScript Frameworks in 2017, 2017. URL: https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282.

[KPZ12] K. Kroflin, S. Prstačić e M. Zagar. Framework for implementation of complex dynamic web forms. In *2012 Proceedings of the 35th International Convention MIPRO*, pages 421–423, May 2012.

[Kra15] Bartosz Krajka. The difference between Virtual DOM and DOM, 2015. URL: http://reactkungfu.com/2015/10/the-difference-between-virtual-dom-and-dom/.

[Kra16] Stefan Krause. JS web frameworks benchmark – Round 4, 2016. URL: http://www.stefankrause.net/wp/?p=316.

[LM16] Jonathan LeBlan e Tim Messerchmidt. *Identity and Data Security for Web Development*. O'Reilly Media, Inc., second early release edition, 2016.

[Mal16] Ravi Malhotra. Angularjs 2: Official release from google team. *ValueCoders*, 2016. URL: http://www.valuecoders.com/blog/industry-news/angularjs-2-official-release-google-team/.

[Mal17] Ravi Malhotra. Vue.js is good, but is it better than angular or react? *ValueCoders*, 2017. URL: http://www.valuecoders.com/blog/technology-and-apps/vue-js-comparision-angular-react/.

[MG11] Peter Mell e Timothy Grance. Nist definition of cloud computing, 2011.

[Mica] Micosoft. URL: https://msdn.microsoft.com/en-us/library/cc751383.aspx.

[Micb] Microsoft. Playground · TypeScript. URL: http://www.typescriptlang.org/play/.

[Mic15] Microsoft. UML Use Case Diagrams: Guidelines, 2015. URL: https://msdn.microsoft.com/en-us/library/dd409432.aspx.

[Mic17] Microsoft. Implementing the model-view-viewmodel pattern, 2017. URL: https://msdn.microsoft.com/en-us/library/ff798384.aspx.

[MMNL14] Kunal Malhotra, Shibani Medhekar, Shamkant B. Navathe e M. D. David Laborde. Towards a form based dynamic database schema creation and modification system. *Advanced Information Systems Engineering (9783319078809)*, page 595, 2014. URL: http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,sso,url,uid&db=edb&AN=98827058&lang=pt-br&site=eds-live&scope=site&authtype=sso.

[Moz17] Mozilla Developer Network. Shadow DOM - Web Components, 2017. URL: https://developer.mozilla.org/en-US/docs/Web/Web_Components/Shadow_DOM.

[MS13] Prerna Mahajan e Abhishek Schdeva. A study of encryption algorithms aes, des and rsa for security. *Global Journal of Computer Science and Technology*, 13(15), 2013.

[Mue15] John Paul Mueller. *Security for Web Developers*. O'Reilly Media Inc., 1 edition, 2015.

[Mul15] Eelco Muller. Angular 2 vs. Angular 1: Key Differences, 2015. URL: https://dzone.com/articles/typed-front-end-with-angular-2.

[One16] One Small Step. Bootstrap vs foundation: Que framework devo usar? *OneSmallStep*, 2016. URL: http://www.onesmallstep.pt/artigo/bootstrap-vs-foundation-que-framework-devo-usar.

[Oraa] Oracle. URL: https://dev.mysql.com/doc/refman/5.7/en/replication-solutions-scaleout.html.

[Orab] Oracle. Database replication. URL: https://docs.oracle.com/cd/A64702_01/doc/server.805/a58227/ch_repli.htm.

[Orac] Oracle. MySQL :: MySQL 5.7 Reference Manual :: 12.9.1 Natural Language Full-Text Searches. URL: https://dev.mysql.com/doc/refman/5.7/en/fulltext-natural-language.html.

[Orad] Oracle. MySQL :: MySQL 5.7 Reference Manual :: 12.9.2 Boolean Full-Text Searches. URL: https://dev.mysql.com/doc/refman/5.7/en/fulltext-boolean.html.

[Orae] Oracle. MySQL :: MySQL 5.7 Reference Manual :: 12.9.3 Full-Text Searches with Query Expansion. URL: https://dev.mysql.com/doc/refman/5.7/en/fulltext-query-expansion.html.

[Oraf] Oracle. MySQL :: MySQL 5.7 Reference Manual :: 12.9.4 Full-Text Stopwords. URL: https://dev.mysql.com/doc/refman/5.7/en/fulltext-stopwords.html.

[Orag] Oracle. MySQL 5.7 Reference Manual - 12.9 Full-Text Search Functions. URL: https://dev.mysql.com/doc/refman/5.7/en/fulltext-search.html.

[Otw] Taylor Otwell. Laravel. URL: https://laravel.com/.

[OWA17] OWASP. Top 10 2017, 2017. URL: https://www.owasp.org/index.php/Top_10_2017-Top_10.

[Per17] FIlipe Pereira. Proteção de dados nas empresas: novo regulamento - obrigações e boas práticas, 2017.

[PRZB12] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich e Hari Balakrishnan. CryptDB. *Communications of the ACM*, 55(9):103, sep 2012. URL: http://dl.acm.org/citation.cfm?doid=2330667.2330691, doi:10.1145/2330667.2330691.

[Res17] André Restivo. Web Security, 2017. URL: https://web.fe.up.pt/{~}arestivo/presentation/security/{#}48.

[Rou14] Margaret Rouse, 2014. URL: http://searchsecurity.techtarget.com/definition/biometric-authentication.

[RSKM13] Aditya Rayarapu, Abhinav Saxena, N.Vamshi Krishna e Diksha Mundhra. Securing files using aes algorithm. *International Journal of Computer Science and Information Technologies*, 4(3):433–435, 2013.

[RSWW13] Daniel Ritter, Florian Schaub, Marcel Walch e Michael Weber. Miba: Multitouch image-based authentication on smartphones. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 787–792, New York, NY, USA, 2013. ACM. URL: http://doi.acm.org/10.1145/2468356.2468497, doi:10.1145/2468356.2468497.

[SBCQ12] G Sadasivan, JN Brownlee, B Claise e J Quittek. *The OAuth 2.0 Authorization Framework*. RFC Editor, 2012. URL: https://tools.ietf.org/html/rfc6749.

[Sch17] Maximilian Schwarzmüller. Angular vs React.js vs Vue.js - My Thoughts!, 2017. URL: https://www.youtube.com/watch?v=KMX1mFEmM3E.

[She67] Roger N. Shepard. Recognition memory for words, sentences, and pictures. *Journal of Verbal Learning and Verbal Behavior*, 6(1):156–163, feb 1967. URL: http://linkinghub.elsevier.com/retrieve/pii/S0022537167800677, doi:10.1016/S0022-5371(67)80067-7.

[Sol15] Deborah Lee Soltesz. The advantages of a relational database management system, 2015. URL: https://www.techwalla.com/articles/the-advantages-of-a-relational-database-management-system.

[Sot14] Caleb Sotelo. Evolution of the Single Page Application, 2014. URL: http://paislee.io/evolution-of-the-single-page-application/.

[Spa06] George Spafford. The importance of audit logs. *Datamation Magazine*, 2006. URL: http://www.datamation.com/columns/article.php/3578916/The-Importance-of-Audit-Logs.htm.

[Sta] StackShare. Bootstrap vs Foundation vs Semantic UI Comparison. URL: https://stackshare.io/stackups/bootstrap-vs-foundation-vs-semantic-ui.

[SWKW13] Florian Schaub, Marcel Walch, Bastian Könings e Michael Weber. Exploring the design space of graphical passwords on smartphones. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, SOUPS '13, pages 11:1–11:14, New York, NY, USA, 2013. ACM. URL: http://doi.acm.org/10.1145/2501604.2501615, doi:10.1145/2501604.2501615.

[TA08] Hai Tao e Carlisle Adams. Pass-go: A proposal to improve the usability of graphical passwords. *I. J. Network Security*, 7(2):273–292, 2008. URL: http://dblp.uni-trier.de/db/journals/ijnsec/ijnsec7.html#TaoA08.

[Tar17] Simon Tarchichi. Simple state management, simpler than Vuex, 2017. URL: http://vuetips.com/simple-state-management-vue-stash.

[TLL12] Woei-Jiunn Tsaur, Jia-Hong Li e Wei-Bin Lee. An efficient and secure multi-server authentication scheme with key agreement. *The Journal of Systems & Software*, 85:876 – 882, 2012. URL: http://search.ebscohost.com/

login.aspx?direct=true&AuthType=ip,sso,url,uid&db=edselp&AN=S0164121211002810&lang=pt-br&site=eds-live&scope=site&authtype=sso.

[Tom] Tom Hare. Avalanche. URL: http://colourgarden.net/avalanche/.

[Tut] TutorialsPoint. TypeScript Overview. URL: https://www.tutorialspoint.com/typescript/typescript_overview.htm.

[UD07] Chris. Ullman e Lucinda. Dykes. *Beginning Ajax*. Wiley Pub, 2007. URL: http://www.wrox.com/WileyCDA/Section/id-303217.html.

[Via15] José Viana. ECMAScript 2015: a nova versão do JavaScript. *Revista Programar*, dec 2015. URL: https://www.revista-programar.info/artigos/ecmascript-2015-nova-versao-do-javascript/.

[Vue17] VueJS. Comparison with other frameworks, 2017. URL: https://vuejs.org/v2/guide/comparison.html#Angular-1.

[W3Sa] W3Schools. JavaScript Versions. URL: https://www.w3schools.com/js/js_versions.asp.

[W3Sb] W3Schools. JSON Introduction. URL: https://www.w3schools.com/js/js_json_intro.asp.

[Web] WebComponents.org. Web components: Introduction. URL: https://www.webcomponents.org/introduction.

[Wei15] Marty Weiner. Learn to stop using shiny new things and love mysql. *Pinterest Blog*, 2015. URL: https://engineering.pinterest.com/blog/learn-stop-using-shiny-new-things-and-love-mysql.

[WPS+00] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme e G. Alonso. Database replication techniques: a three parameter classification. In *Proceedings 19th IEEE Symposium on Reliable Distributed Systems SRDS-2000*, pages 206–215, 2000. doi:10.1109/RELDI.2000.885408.

[WWSB06] Susan Wiedenbeck, Jim Waters, Leonardo Sobrado e Jean-Camille Birget. Design and evaluation of a shoulder-surfing resistant graphical password scheme. *Proceedings of the Working Conference: Advanced Visual Interfaces (9781595933539)*, page 177, 2006. URL: http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,sso,url,uid&db=edb&AN=84181835&lang=pt-br&site=eds-live&scope=site&authtype=sso.

[XYO05] Suo Xiaoyuan, Zhu Ying e G.S. Owen. Graphical passwords: a survey. *21st Annual Computer Security Applications Conference (ACSAC'05)*, page 472, 2005. URL: http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,sso,url,uid&db=edb&AN=81090479&lang=pt-br&site=eds-live&scope=site&authtype=sso.

[Yer13] Johnathan Yerby, 2013. URL: https://youtu.be/kGpAdbBudOU.

REFERÊNCIAS

Anexo A

Plano de Testes de Validação

A.1 Gestão de formulários

Utilizadores

- admin

Condições iniciais

- Utilizador com login feito na plataforma
- Página inicial: admin/gestao_formularios
- Formulário em vigor para a linha de apoio à vítima: linha_apoio_a_vitima_02

Informação para os utilizadores *a priori*

- Podem ser criados e geridos formulários para diferentes serviços. Cada formulário é constituído por secções, e cada secção apresenta campos (as questões);
- Cada campo é associado a um formulário através de um código único identificativo desse campo, a partir de um banco de campos de formulários gerido pelo utilizador;
- É possível definir regras para mostrar/ocultar campos com base em respostas do utilizador a outros campos de uma secção;
- É possível definir regras para indicar qual a próxima secção com base em respostas do utilizador numa dada secção.

A.1.1 Cenários

Cenário 1 Consultar os dados gerais do campo de formulário de código "localidade".
Percurso preferencial

1. Clicar em "Campos"
2. Clicar em “Gerir existentes”
3. Selecionar o código “Localidade”
4. Visualizar o separador “Geral”

Cenário 2 Indicar qual a última alteração feita por um utilizador no campo “localidade”

Percurso preferencial

1. Consultar, no separador “Geral”, o histórico de edições
2. Indicar o primeiro elemento da tabela de edições

Cenário 3 Criar o seguinte campo:

- Título — Como teve conhecimento dos nossos serviços?
- Código — conhecimento_servicos
- Tipo — *checkbox*
- Opções: PJ (código: pj), Amigo/Conhecido (código: amigo_conhecido), Comunicação Social (código: comunicacao_social)
- Deve apresentar o campo "outro"

Percurso preferencial

1. Clicar, na secção “Campos”, em “Novo”
2. Preencher título “Como teve conhecimento dos nossos serviços?”
3. Preencher código “conhecimento_servicos”
4. Selecionar tipo “*checkbox*”
5. Adicionar respostas possíveis, gerando os códigos respetivos
6. Selecionar “sim” na “opção outro”
7. Pré-visualizar o campo
8. Guardar

Cenário 4 Pré-visualizar o formulário "linha_de_apoio_a_vitima_01"

Percurso preferencial

1. Clicar em “Formulários”
2. Clicar em “Gerir existentes”
3. Selecionar o formulário de código “linha_de_apoio_a_vitima_01”
4. Clicar em “Pré-visualizar/Guardar”
5. Visualizar o formulário

Cenário 5 O formulário “linha_de_apoio_a_vitima_01” apresenta todas as secções seguidas, sem paginação. Editar o formulário, de forma a que pagine as secções.

Percurso preferencial

1. Clicar em “Editar opções e secções”
2. Selecionar “sim” em “Paginar”
3. Pré-visualizar

Cenário 6 Adicionar a secção “Encerramento de contacto” (código “sec_encerramento_contacto”), com o campo “conhecimento_servicos”.

Percurso preferencial

1. Adicionar, na listagem de secções, a secção com título “Encerramento de contacto” e código “sec_encerramento_contacto”
2. Clicar em “Editar secção”
3. Selecionar o código da secção criada
4. Clicar em “Novo campo”
5. Selecionar o código “conhecimento_servicos”

Cenário 7 Adicionar um campo de texto a seguir ao campo “conhecimento_servicos”, a negrito, com o texto “Obrigada”. Esse texto só deve aparecer caso a resposta ao campo “conhecimento_servicos” não foi vazia.

Percurso preferencial

1. Clicar no ícone “+” no campo existente
2. Clicar no botão para alterar o tipo de campo para texto
3. No editor, clicar em "B"

4. Escrever “Obrigada”
5. Selecionar “Não” em “Sempre visível”
6. Selecionar: Ação “mostrar”, Quando “todos”, Campo “conhecimento_servicos”, Condição “NÃO É”, Valor <vazio>
7. Pré-visualizar e testar

Cenário 8 Editar o formulário, de forma a que apareça a secção “encerramento_contacto” caso o campo “Tem mais alguma questão?” na secção “crime” seja “não”.

Percurso preferencial

1. Clicar em “Editar secção”
2. Selecionar a secção “Crime”
3. Carregar em “Nova próxima secção”
4. Selecionar: Próxima secção “encerramento_contacto”, Quando “todos”, Campo “tem_mais_questoes”, Condição “é”, Valor “não”
5. Pré-visualizar e testar

Cenário 9 Guardar o formulário, indicando como alterações “edição de teste”.

Percurso preferencial

1. Escrever na mensagem “edição de teste”
2. Guardar

Cenário 10 Ativar o formulário linha_de_apoio_a_vitima_01 para o serviço.

Percurso preferencial

1. Selecionar formulário “linha_de_apoio_a_vitima_01”
2. No separador “Geral”, clicar em “Ativar”

A.2 Atendimentos e processos

Utilizadores

- voluntario

Condições iniciais

- Utilizador com login feito, na página atendimentos/novos_atendimentos;
- Formulário ativo para a linha de apoio à vitima: linha_de_apoio_a_vitima_02.

Informação para os utilizadores *a priori*

- O sistema irá apresentar, enquanto o novo atendimento é preenchido, os processos existentes que têm mais probabilidade de corresponder ao utente em questão, com base nas respostas dadas.

Cenário 1 A partir da página atendimentos/novos_atendimentos, iniciar um novo atendimento para a Linha de Apoio à Vítima

Percurso preferencial

1. Selecionar o serviço "Linha de apoio à vítima"

Cenário 2 Preencher o questionário. A vítima forneceu os seguintes dados:

- Relatou imediatamente a situação
- O seu nome é Maria Antónia, não sendo este o seu primeiro contacto telefónico com a APAV
- Contacto telefónico é 123456789
- Mora na Rua da Maria
- Nasceu a 12/01/1987
- Foi vítima de violência doméstica num relacionamento heterossexual

Percurso preferencial

1. Preencher os dados indicados.

Cenário 3 Verificar os processos existentes sugeridos pela plataforma e selecionar o processo mais provável

Percurso preferencial

1. Consultar a listagem de processos
2. Selecionar o processo "2017-00001"

Cenário 4 Preencher os campos "Foi/é uma situação pontual ou existem sinais de repetição?" e "Tem filhos a seu cargo?" a partir da resposta do atendimento anterior, indicado no processo selecionado.

Percurso preferencial

1. Na secção "Dados Vítima", clicar nos botões de ligar a resposta antiga ao processo atual dos campos pedidos.

Cenário 5 Submeter o novo atendimento.

Percurso preferencial

1. Clicar em “Submeter”
2. Confirmar que o atendimento deve ser associado ao processo "2017-00001"

Cenário 6 Consultar o processo "2017-00001": dados gerais e atendimentos.

Percurso preferencial

1. Na barra superior, clicar em “Atendimentos”
2. Clicar em “Processos de apoio”
3. Escrever o código de processo "2017-00001"
4. Clicar em "Pesquisar"ou carrega na tecla *Enter*
5. Ver os dados gerais do processo
6. Ver os atendimentos do processo

A.3 Área do utilizador

Utilizadores

- admin
- voluntario

Condições iniciais

- Último ecrã do teste de admin ou voluntario, consoante o caso.

Informação para os utilizadores *a priori*

- Existe uma *dashboard* com calendário, alertas, notas, e indicação da atividade e atendimentos associados ao utilizador;
- É possível criar e gerir eventos de calendário.

A.3.1 Cenários

Cenário 1 A partir da página da *dashboard*, ir para a página do calendário. Na página do calendário, criar um novo evento para todo o dia 10/06/2017. Este evento deve ter o título "Elaborar" e descrição "relatório X". No dia do evento, o utilizador deve receber um alerta via email.

Percurso preferencial

1. Clicar em "Dashboard"
2. Clicar em "Calendário"
3. Na vista "Mês", clicar no dia "10/06/2017".
4. Selecionar o título "Elaborar", escrever na descrição "relatório X" e marcar o campo "Enviar email"
5. Guardar

Cenário 2 Alterar o evento criado de modo a que este ocorra apenas das 9h às 10h no mesmo dia.

Percurso preferencial

1. Clicar na vista "Semana"
2. Arrastar o evento de forma a que comece às 9h
3. Guardar
4. Alterar a duração do evento para acabar às 10h, redimensionando-o
5. Guardar

Cenário 3 Consultar a agenda do dia 10/06/2017 e editar o evento "Elaborar: relatório X", alterando o título para "Rever"

Percurso preferencial

1. Clicar em "Agenda do dia"
2. Clicar no evento "Elaborar: relatório X"
3. Clicar em Editar
4. Alterar o tipo de título para "Livre"
5. Escrever no título "Rever"
6. Guardar

Anexo B

Questionários de validação qualitativa e respostas dos utilizadores

Neste anexo são apresentadas todas as questões realizadas aos utilizadores após efetuarem os testes de validação, bem como as respostas individuais. O questionário é apresentado na secção B.1. As respostas são apresentadas na secção B.2.

B.1 Questionários

B.1.1 Gestão de formulários

B.1.1.1 Gestão de campos de formulários

1. Enquanto realizava as tarefas pedidas,

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Senti-me confortável

   B Senti-me com dificuldade

   C Senti-me satisfeito

   D Senti que era fácil aceder à informação que precisava

   E Senti que foi necessário demasiado trabalho para atingir o resultado final

   F Senti-me perdido

   G Senti-me com vontade de voltar a utilizar a aplicação

2. O que gostou mais?

3. O que gostou menos?

4. Sugestões de alterações

B.1.1.2 Gestão de formulários

1. Enquanto realizava as tarefas pedidas,

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Senti-me confortável

   B Senti-me com dificuldade

   C Senti-me satisfeito

   D Senti que era fácil aceder à informação que precisava

   E Senti que foi necessário demasiado trabalho para atingir o resultado final

   F Senti-me perdido

   G Senti-me com vontade de voltar a utilizar a aplicação

2. Que outras ferramentas para construção de formulários dinâmicos já utilizou?

   A Google Forms

   B Formsite

   C Type-Forms

   D Outro (qual?)

3. Comparativamente às ferramentas para construção de formulários utilizadas, considera que a edição de formulários na PLAGA é:

   (escolher valor de 1 a 5, conforme a legenda)

   A Muito mais difícil de utilizar (1)

   B Muito mais fácil de utilizar (5)

4. O que gostou mais?

5. O que gostou menos?

6. Sugestões de alterações

B.1.2 Áreas de atendimentos e processos

B.1.2.1 Novos atendimentos

1. Enquanto realizava as tarefas pedidas,

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Senti-me confortável

B Senti-me com dificuldade

C Senti-me satisfeito

D Senti que era fácil aceder à informação que precisava

E Senti que foi necessário demasiado trabalho para atingir o resultado final

F Senti-me perdido

G Senti-me com vontade de voltar a utilizar a aplicação

2. Como classifica o sistema de procura de processos existentes com base nas respostas do formulário de atendimento?

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Útil

   B Rápido

   C Intuitivo

3. Como classifica o sistema de preenchimento do formulário de atendimento com base em respostas anteriores?

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Útil

   B Rápido

   C Intuitivo

4. O que gostou mais?

5. O que gostou menos?

6. Sugestões de alterações

B.1.2.2 Processos

1. Enquanto realizava as tarefas pedidas,

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Senti-me confortável

   B Senti-me com dificuldade

   C Senti-me satisfeito

   D Senti que era fácil aceder à informação que precisava

   E Senti que foi necessário demasiado trabalho para atingir o resultado final

F Senti-me perdido

G Senti-me com vontade de voltar a utilizar a aplicação

2. Como classifica a nova organização dos processos (dados gerais + listagem de atendimentos realizados), em comparação com o atual PAO?

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Mais confuso

   B Mais agradável

   C Mais útil

   D Mais organizado

   E Permite aceder a mais informação

3. O que gostou mais?

4. O que gostou menos?

5. Sugestões de alterações

B.1.3 Área pessoal do utilizador

1. Qual o seu perfil?

   A Gestor

   B TAV

2. Enquanto realizava as tarefas pedidas,

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Senti-me confortável

   B Senti-me com dificuldade

   C Senti-me satisfeito

   D Senti que era fácil aceder à informação que precisava

   E Senti que foi necessário demasiado trabalho para atingir o resultado final

   F Senti-me perdido

   G Senti-me com vontade de voltar a utilizar a aplicação

3. O que gostou mais?

4. O que gostou menos?

5. Sugestões de alterações

B.2 Respostas

Foram obtidas 2 respostas relativas à Gestão de formulários e campos de formulário, 5 respostas relativas às Áreas de atendimentos e processos, e 7 respostas relativas à Área pessoal do utilizador.

Para cada questão do formulário, nos casos de perguntas que pediam valores de classificação, são indicadas as diferentes respostas separadas por vírgula; nas perguntas de resposta aberta, são indicadas as respostas individuais linha a linha.

B.2.1 Área de administração de formulários e campos de formulários

B.2.1.1 Gestão de campos de formulários

1. Enquanto realizava as tarefas pedidas,

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Senti-me confortável — 5, 5

   B Senti-me com dificuldade — 1, 1

   C Senti-me satisfeito — 5, 5

   D Senti que era fácil aceder à informação que precisava — 5, 5

   E Senti que foi necessário demasiado trabalho para atingir o resultado final — 1, 1

   F Senti-me perdido — 1, 2

   G Senti-me com vontade de voltar a utilizar a aplicação — 5, 5

2. O que gostou mais?

   - Interface *user-friendly*.

   - A facilidade e usabilidade da aplicação e do interface. Clean e agradável!

3. O que gostou menos?

   - Nada a apontar

   - Nada a assinalar

4. Sugestões de alterações

   - Nada a apontar.

   - Até ao momento, a aplicação está conforme as necessidades da APAV pelo que não há nada a assinalar.

B.2.1.2 Gestão de formulários

1. Enquanto realizava as tarefas pedidas,

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Senti-me confortável — 5, 5

   B Senti-me com dificuldade — 1, 2

   C Senti-me satisfeito — 5, 5

   D Senti que era fácil aceder à informação que precisava — 4, 5

   E Senti que foi necessário demasiado trabalho para atingir o resultado final — 1, 1

   F Senti-me perdido — 1, 1

   G Senti-me com vontade de voltar a utilizar a aplicação — 5, 5

2. Que outras ferramentas para construção de formulários dinâmicos já utilizou?

   - Google Forms (2x)

   - Formsite

   - Access e Visual Basic

3. Comparativamente às ferramentas para construção de formulários utilizadas, considera que a edição de formulários na PLAGA é:

   - Muito mais fácil de utilizar (2x)

4. O que gostou mais?

   - Interface user-friendly, imagem agradável

   - Da facilidade de criação e alteração sem necessidade de grandes conhecimentos em programação.

5. O que gostou menos?

   - Nada a apontar.

   - Nada a assinalar

6. Sugestões de alterações

   - Nada a apontar.

   - Se possível, ao invés de mostrar o nome dos campos do formulário ao utilizador final, mostrar a questão que diz respeito a esses mesmos campos.

B.2.2 Áreas de atendimentos e processos

B.2.2.1 Novos atendimentos

1. Enquanto realizava as tarefas pedidas,

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Senti-me confortável — 4, 4, 5, 5, 5

   B Senti-me com dificuldade — 2, 3, 3, 2, 1

   C Senti-me satisfeito — 4, 4, 5, 5, 5

   D Senti que era fácil aceder à informação que precisava — 4, 4, 5, 5, 4

   E Senti que foi necessário demasiado trabalho para atingir o resultado final — 3, 3, 1, 1, 1

   F Senti-me perdido — 3, 2, 3, 1, 2

   G Senti-me com vontade de voltar a utilizar a aplicação — 4, 4, 5, 5, 5

2. Como classifica o sistema de procura de processos existentes com base nas respostas do formulário de atendimento?

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Útil — 4, 4, 5, 5, 5

   B Rápido — 4, 4, 5, 4, 5

   C Intuitivo — 4, 4, 5, 3, 5

3. Como classifica o sistema de preenchimento do formulário de atendimento com base em respostas anteriores?

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Útil — 4, 4, 5, 4, 5

   B Rápido — 4, 5, 5, 4, 5

   C Intuitivo — 4, 5, 5, 4, 5

4. O que gostou mais?

   - Procura de perguntas às quais já tenham sido obtidas respostas anteriormente.

   - A possibilidade de procura automática dos processos existentes, através dos diversos itens.

   - A relação dos processos já existentes.

   - Funcionalidades necessárias reunidas no mesmo programa, logo, maior eficácia preenchimento automático da informação; associação de processos

5. O que gostou menos?

   - Nada.

   - Cores poderão ser alteradas, mais dinâmicas, para perceção dos diversos campos. Botão submeter ser no topo.

   - Ligação das respostas ao novo formulário (apresentação)

   - Ligação das respostas já dadas ao novo atendimento - pouco intuitivo.

   - Nada.

6. Sugestões de alterações

   - Ligação das informações preenchidas e os processos anteriores.

   - As evidenciadas supra: cores e localização "botão submeter".

   - Nenhuma.

   - Um processo mais intuitivo relativamente ao item apontado anteriormente.

   - Nenhuma.

B.2.2.2 Processos

1. Enquanto realizava as tarefas pedidas,

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Senti-me confortável — 4, 4, 5, 5, 5

   B Senti-me com dificuldade — 2, 3, 2, 2, 2

   C Senti-me satisfeito — 4, 4, 5, 5, 5

   D Senti que era fácil aceder à informação que precisava — 4, 4, 5, 5, 4

   E Senti que foi necessário demasiado trabalho para atingir o resultado final — 2, 2, 1, 1, 1

   F Senti-me perdido — 2, 3, 3, 2, 2

   G Senti-me com vontade de voltar a utilizar a aplicação — 4, 4, 5, 5, 5

2. Como classifica a nova organização dos processos (dados gerais + listagem de atendimentos realizados), em comparação com o atual PAO?

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Mais confuso — 1, 1, 1, 1, 1

   B Mais agradável — 4, 4, 5, 5, 5

   C Mais útil — 4, 4, 5, 5, 5

   D Mais organizado — 4, 4, 5, 5, 5

E Permite aceder a mais informação — 4, 4, 5, 5, 5

3. O que gostou mais?

   - Do acesso a informações obtidas anteriormente no âmbito daquele processo.

   - A atualização automática.

   - Acesso a maior informação (mais itens de pesquisa)

   - Tudo

   - Possibilidade de ver informação existente em cada atendimento

4. O que gostou menos?

   - Nada

   - A necessidade de repetição das mesmas questões por atendimento, quando já poderão estar inseridas.

   - Nada

   - Nada a apontar

   - Nada

5. Sugestões de alterações

   - Nenhuma

   - Nenhuma

   - Previamente, questionar se a temática mantém-se

   - Nenhuma

   - Nenhuma

B.2.3 Área pessoal do utilizador

1. Qual o seu perfil?

   Gestor (x2)

   TAV (x5)

2. Enquanto realizava as tarefas pedidas,

   (classificar de 1 a 5, sendo 1 = Discordo totalmente e 5 = Concordo totalmente)

   A Senti-me confortável — 4, 5, 4, 5, 5, 5

   B Senti-me com dificuldade — 2, 1, 3, 1, 1, 1

   C Senti-me satisfeito — 4, 5, 4, 5, 5, 5

   D Senti que era fácil aceder à informação que precisava — 5, 5, 4, 5, 5, 5

E Senti que foi necessário demasiado trabalho para atingir o resultado final — 2, 1, 3, 1, 1, 1

F Senti-me perdido — 2, 1, 2, 1, 2, 1

G Senti-me com vontade de voltar a utilizar a aplicação — 5, 5, 4, 5, 5, 5

3. O que gostou mais?

   - Avisos das reavaliações de risco

   - Interface; possibilidade de filtrar no imediato os processos semelhantes já existentes

   - Da possibilidade de calendário, com o aviso por email.

   - De fácil compreensão

   - Calendário e alertas

   - Da usabilidade e da aparência

   - Do calendário

4. O que gostou menos?

   - Não poder alterar diretamente as horas dos eventos

   - Nada a apontar

   - Forma de editar o evento.

   - Parte da edição dos eventos

   - Nada

   - Nada a assinalar

   - Nada

5. Sugestões de alterações

   - Nenhuma

   - Nada a apontar

   - Estudar a questão de editar o evento.

   - Nada

   - Título e evento parecem referir-se a itens diferentes; talvez seja melhor dar-lhes o mesmo nome

   - O utilizador final deveria ver as questões associadas aos campos das tabelas e não o contrário, se possível

   - Poder editar a data do evento manualmente