IPG Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE ESTÁGIO

Licenciatura em Engenharia Informática

Rui Miguel Gonçalves Agostinho

novembro | 2020

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

RELATÓRIO DE ESTÁGIO

GESTCOP – GESTÃO E CONTROLO DE OCORRÊNCIAS POLICIAIS

RUI MIGUEL GONÇALVES AGOSTINHO

RELATÓRIO PARA OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMÁTICA

Novembro de 2020

Agradecimentos

Com o culminar deste documento finda mais uma etapa do crescimento. Poderia dizer que neste momento “aprendi a andar” e pretendo evoluir ainda mais. Assim, será minha intenção manter-me atualizado em termos de conhecimentos técnicos e teóricos e continuar a aposta na formação como um dos pilares essenciais na construção do conhecimento.

Ao conjunto de pessoas que direta e indiretamente me fizeram crescer, evoluir, pela ajuda, dedicação, colaboração, disponibilidade, entrega deixo uma palavra de especial apreço e gratidão.

Começaria por agradecer à Escola Superior de Tecnologia e Gestão (ESTG) – Instituto Politécnico da Guarda (IPG) a oportunidade que me deu de alimentar a minha curiosidade pela informática, orientando-me formalmente neste caminho.

Um especial agradecimento ao Professor Doutor José Fonseca, orientador académico do Projeto de Estágio. Com as suas sábias palavras de incentivo, com as críticas construtivas e assertivas no momento certo e as sugestões de melhoria permitiram que este documento, assim como o trabalho desenvolvido pudessem assumir uma qualidade que considero relevante. Destaco igualmente a disponibilidade quer por telefone, via email e até presencialmente para me (des)orientar. Um muito obrigado.

A todos os docentes e não docentes do IPG, que ao longo destes anos me acompanharam nas várias unidades curriculares e extracurriculares.

Um reconhecido agradecimento à instituição de acolhimento de estágio, Polícia de Segurança Pública, pela oportunidade.

Um reconhecimento especial ao Sr. Superintendente José Leonardo, na qualidade de supervisor de estágio, por toda a ajuda, aconselhamento, tempo disponibilizado e orientação durante o estágio.

Ao meu amigo Filipe Caetano, que ao longo deste sinuoso caminho académico tanto contribuiu para o desenvolvimento, enriquecimento e crescimento do saber académico; ao meu amigo Paulo Silvestre, por toda a ajuda, dedicação e paciência demonstrada ao longo deste trabalho.

A todas as Pessoas Especiais, Amigos ou familiares, pois sem eles nada disto teria sido possível.

Por último, mas o mais importante, um inevitável agradecimento do fundo do coração, à minha esposa Sílvia, à minha filha Lara e aos meus pais, por todo o apoio, ânimo, perseverança, paciência, compreensão e por estarem sempre ao meu lado.

A todas estas pessoas o meu PROFUNDO E IMPERIOSO AGRADECIMENTO!!!

"O maior inimigo do conhecimento não é a ignorância, é a ilusão do conhecimento."

Stephen Hawking

Ficha de identificação

Aluno

Nome: Rui Miguel Gonçalves Agostinho

Número: 1009815

Curso: Engenharia Informática

Estabelecimento de Ensino

Instituição: Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

Morada: Rua Dr. Francisco Sá Carneiro, nº 50, 6300-559, Guarda

Telefone/Fax: 271 220 120/271 220 150

Instituição Acolhedora de Estágio

Instituição: Polícia de Segurança Pública

Departamento de Sistemas Informáticos e Comunicações

Morada: Largo da Penha de França, nº 1, 1199-010, Lisboa

Telefone/Fax: 218 111 000/218 147 705

Supervisor Institucional

Nome: José Manuel da Cruz Belo Pires Leonardo

Duração do Projeto

Inicio: 07 de abril de 2020 **Fim:** 01 de julho de 2020

Orientador do Projeto

Nome: José Carlos Coelho Martins Fonseca

Resumo

Este relatório descreve as tarefas e decisões tomadas para a elaboração do projeto desenvolvido em contexto de estágio efetuado na Polícia de Segurança Pública. O projeto surgiu no âmbito da unidade curricular, Projeto de Informática, do 3º ano, de Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda.

Até à data, a Polícia de Segurança Pública, regista todo o trabalho efetuado numa aplicação informática. Após uma análise a este sistema conseguem-se identificar algumas lacunas. Uma delas é o facto de ser uma aplicação pouco amigável para com o utilizador, outra deve-se ao facto de que este sistema não faz a gestão do registo das ocorrências reportadas à polícia, o que se traduz em procedimentos de registo pouco uniformes e por vezes com falhas na recolha de informação relevante para a resolução criminal de um determinado crime.

Procurando contribuir para a melhoria do registo de ocorrências e sua uniformidade, este projeto consiste na construção de uma aplicação *Web*, a GestCop, que servirá de protótipo para uma nova abordagem ao registo e gestão de ocorrências policiais na Polícia de Segurança Pública relativas à violência doméstica. No desenvolvimento do projeto foi usada a metodologia de desenvolvimento ágil, SCRUM. A aplicação foi concebida usando as *frameworks* Node.JS para o *backend* e React.JS – Redux para *frontend*, com recurso à base de dados MySQL.

Um projeto desta natureza por si só já é vasto e extenso, conjugando o facto da aprendizagem de novas tecnologias, originou que se perde-se muito tempo, levando a que se implementassem muito poucas funcionalidades do projeto.

Palavras-chave: Aplicação *Web*, Ocorrências policiais, *Web Services*, Node.JS, React.JS, Redux.

Abstract

This report describes the tasks and decisions taken to prepare the project developed in the context of an internship carried out at the *Polícia de Segurança Pública*. The project arose in the scope of the Informatics Project curricular unit, of the 3rd year of Computer Engineering of the *Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda*.

Up until now, the *Polícia de Segurança Pública*, record all the work done on a computer application. After analysing this system, it is possible identify some gaps, one of the breaches is due to the fact that it is a very unfriendly application with the user, another is due to the fact that this system not manage the occurrence record, which translates into uneven registration procedures and sometimes flaws in the collection of information relevant to the criminal resolution of a given crime.

The project consists to build a Web application, which will serve as a prototype, or as a base system for a new approach to the registration and management of police occurrences in the *Polícia de Segurança Pública*. For the development of the project, the agile development methodology, SCRUM, was used. The application was designed using the Node.JS frameworks for the backend and React.JS - Redux for the frontend, using the MySQL database.

A project of this nature is vast and never completely finished. However, the help system was developed based on a question / answer concept and the Wizard menu, which will support its development.

A project of this nature alone is already vast and extensive, combining the fact that learning new technologies has resulted in a lot of time being lost, leading to the implementation of very few features of the project.

Keywords: Web application, Police incidents, Web Services, Node.JS, React.JS, Redux

Índice Geral

Índice de figuras

Índice de tabelas

Lista de acrónimos

API	*Application Programming Interfaces*
BD	Bases de Dados
CORS	*Cross-Origin Resource Sharing*
CP	Código Penal
CPCJ	Comissão de Proteção de Crianças e Jovens
CPP	Código Processual Penal
CRUD	*Create, Read, Update, Delete*
CSRF	*Cross-Site Request Forgery*
CSS	*Cascading Style Sheets*
DN	Direção Nacional
DOM	*Document Object Model*
DSIC	Departamento de Sistemas de Informática e Comunicações
ESTG	Escola Superior de Tecnologia e Gestão
FIFO	*First In First Out*
HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
INML	Instituto Nacional de Medicina Legal
IPG	Instituto Politécnico da Guarda
JSON	JavaScript Objet Notation
JWT	*Json Web Token*
LDAP	*Lightweight Directory Access Protocol*
MAI	Ministério da Administração Interna
MP	Ministério Público
NPM	*Node Package Manager*
PC	Computador Pessoal
PSP	Polícia de Segurança Pública
REST	*Representational State Transfer*
RGPD	Regulamento Geral de Proteção de Dados

ROE	Registo de Ocorrência Escolar
RUP	*Rational Unified Process*
RVL	Risco de Violência Doméstica
SEI	Sistema Estratégico de Informação, Gestão e Controlo Operacional
SPA	*Single Page Aplication*
SQL	*Structured Query Language*
TIR	Termo de Identidade e Residência
UML	*Unified Modelling Language*
URL	*Uniform Resource Locator*
VSC	Visual Studo Code
XSS	*Cross-site Scripting*

1 Introdução

O presente relatório serve para descrever as atividades efetuadas no estágio realizado pelo aluno Rui Miguel Gonçalves Agostinho, no âmbito da unidade curricular Projeto de Informática, ministrada no terceiro ano da Licenciatura em Engenharia Informática, pela Escola Superior de Tecnologia e Gestão (ESTG) do Instituto Politécnico da Guarda (IPG).

O estágio decorreu no Departamento de Sistemas de Informação e Comunicações (DSIC) da Direção Nacional (DN) da Polícia de Segurança Pública (PSP), tendo como missão o desenvolvimento de um módulo para uma aplicação que efetue o registo de todas as denúncias e ocorrências policiais no âmbito de um processo criminal de violência doméstica e que pudesse ser utilizado em vários *browsers* e em vários dispositivos, tanto móveis como fixos.

1.1 Motivação/Enquadramento

A Unidade Curricular de Projeto de Informática do IPG fornece aos alunos duas metodologias distintas de avaliação, nomeadamente um projeto académico ou, em alternativa, a realização de um projeto em contexto de estágio.

Perante estas possibilidades optou-se pelo projeto em contexto de estágio. Esta oportunidade permitiu estabelecer uma ligação real e concreta com o mercado de trabalho, perceber e dirimir algumas das dificuldades com que se defrontam os profissionais, contudo também lançou bases para experiências únicas.

Como profissional da Polícia de Segurança Pública (PSP) tive a possibilidade de me envolver com a instituição através de uma nova realidade, de novos conhecimentos e distintos métodos de trabalho que considero de extrema relevância para a criação de um nível de

comprometimento superior, estabelecendo-se paulatinamente uma identidade profissional enquanto futuro Técnico Superior de Informática.

De forma humilde e extremamente motivada pretendo colocar à disposição do projeto os conhecimentos que adquiri ao longo do Curso de Engenharia Informática. No mesmo sentido venho para trabalhar, para fazer parte de uma equipa equilibrada e principalmente disponível para aprender.

Pretendo aliar características individuais, o conhecimento teórico, a prática profissional, o relacionamento interpessoal e o comprometimento institucional/organizacional no desenvolvimento de um desafio que considero ambicioso e deveras aliciante.

1.2 Caraterização sumária da instituição

A instituição acolhedora do estágio, assume a denominação de Polícia de Segurança Pública (PSP). Oficialmente, a história da instituição começou a 02 de julho de 1867, durante o reinado de D. Luís I que se criou em Portugal o Corpo de Polícia Civil de Lisboa e o Corpo de Polícia Civil do Porto. Em março de 1927, foram reestruturados e deram origem à PSP (Polícia de Segurança Pública, 2020).

A PSP rege-se por um regulamento próprio (Lei Orgânica da PSP), aprovado pela Lei nº 53/2007 de 31 de agosto.

Com base nesta Lei, a PSP é:

- Uma força de segurança uniformizada e armada, com natureza de serviço público e dotada de autonomia administrativa (artigo 1.º, nº1);
- Tem como missão assegurar a legalidade democrática, garantir a segurança interna e os direitos dos cidadãos, nos termos da Constituição e da Lei (artigo 1.º, nº2).

A PSP tem atribuída uma quantidade significativa de competências (artigo 3.º), que as coloca em prática por todo o território nacional (artigo 5.º, nº2), estando organizada hierarquicamente em todos os níveis da sua estrutura. Os colaboradores com funções policiais estão sujeitos à hierarquia de comando e os colaboradores sem funções policiais sujeito às regras gerais da hierarquia da função pública (artigo 1.º nº3) (Diário da República Eletrónico, 2020). No Anexo10.1, estão descriminadas todas as competências atribuídas à PSP.

Por forma a melhorar a eficiência e capacidade dos serviços, a PSP está dividida em diversos departamentos, todos eles sobre alçada e supervisão do Diretor Nacional (Polícia de Segurança Publica, 2020). No Anexo 10.2, está disponível o organigrama organizacional da PSP

O departamento onde decorreu o estágio é designado por Departamento de Sistemas de Informação e Comunicações (DSIC), e o principal objetivo deste departamento é gerir todos os sistemas de informação e de comunicações por forma a melhorar a produtividade e eficiência do modelo organizativo (Diário da República Eletrónico, 2020).

Para cumprir este objetivo, o DSIC encontra-se estruturalmente dividido em duas áreas: a área de Informática e a área de Comunicações. A área de Informática, por sua vez, ainda se encontra dividida em duas Divisões: a Divisão de Sistemas, onde decorreu o estágio e a Divisão de Infraestruturas.

1.3 Objetivos previstos

No desenrolar da atividade policial, a PSP, para além de garantir a segurança dos cidadãos, procede ao registo de todas as queixas e denúncias criminais, contraordenacionais ou de mera responsabilidade civil de que tem conhecimento direto ou que lhe são reportadas pelos cidadãos. Ainda no decorrer desta atividade, os elementos policiais, aquando de uma qualquer intervenção policial, efetuam um relatório de ocorrência.

As queixas, denúncias e relatórios de ocorrências que são qualificadas como crimes (Ex: crime de homicídio, ofensas à integridade física, roubo, etc.), são obrigatoriamente enviadas ao Ministério Público (MP) (Art. 241º e Art. 242º do Código Processo Penal (CPP)) sob a forma de Auto de notícia (Art. 241º e Art. 242º do CPP). As queixas, denúncias, e relatórios de ocorrência a nível contraordenacional (Ex: Mordedura de um canídeo, venda de produtos de forma ambulante, ruido, etc), são enviadas para a entidade competente da área territorial onde se verificou a infração (nos casos anteriores para a Câmara municipal), sob a forma de Auto de Notícia por Contraordenação.

Para registo, gestão e controlo destas ocorrências/denúncias policiais, a PSP tem ao seu dispor uma aplicação informática denominada Sistema Estratégico de Informação, Gestão e Controlo Operacional (SEI). Este dispositivo já se encontra implementado na PSP desde o ano de 2004, o que evidencia a sua maturidade, credibilidade e equilíbrio entre a informação que se introduz e a informação que produz depois de a trata.

Apesar do SEI possuir um nível de fiabilidade bastante elevado, não conseguiu acompanhar a evolução tecnológica, mesmo embora tenha tido inúmeras atualizações, o que faz com que apresente algumas limitações. Algumas destas limitações prendem-se pelo facto de, ao longo destes últimos anos, a tecnologia ter evoluído de forma exponencial, o que levou a que surgissem no mercado novas tecnologias cada vez mais rápidas e mais robustas, ficando o SEI com uma arquitetura em desuso.

Uma das limitações daqui decorrentes prende-se pelo facto da arquitetura do SEI estar completamente dependente do *browser* Internet Explorer, Versão 8, para funcionar. Atualmente, está a funcionar graças ao modo de compatibilidade da Versão 11 do *Internet Explorer*. Outra limitação prende-se com a portabilidade do sistema, pois não foi pensado para trabalhar em dispositivos móveis.

Após terem sido identificadas as principais melhorias a desenvolver e, tendo em conta a grande diversidade de ocorrências decorrentes da atividade policial, ficou definido que no âmbito do presente estágio, deveria ser criada uma nova aplicação *Web*, a GestCop, que servisse de base para um novo sistema ou servir de protótipo para melhoria do sistema atual. Como base inicial de trabalho, por se achar que os crimes de violência doméstica são crimes que envolvem um grande número de procedimentos no ato do registo da ocorrência/denúncia, decidiu-se que a GestCop deveria ser capaz de efetuar o registo destas ocorrências/denúncias.

Foi ainda decidido que para além do registo de ocorrências/denuncias de violência doméstica, a GestCop deverá ainda cumprir os seguintes requisitos:

- Completar as ocorrências/denúncias que se encontram por terminar;
- Emitir Autos de notícia de forma simples e o mais automatizada possível;
- Ser responsiva, ou seja, funcionar tanto em dispositivos móveis como em PCs;
- Funcionar pelo menos nos *browsers* mais populares: Microsoft Edge, Mozilla Firefox e Google Chrome;
- Deve ser mais interativa com o utilizador no que concerne ao registo de ocorrências, minimizado assim lacunas na recolha de informação proveniente das mais variadas fontes, como por exemplo: local do crime, contacto com testemunhas, procedimentos operacionais, etc.;

Há ainda a ambição, numa segunda fase da conceção da GestCop, da concretização do seguinte requisito: Recolher automaticamente dados através de uma câmara de filmar. Ou seja, pretende-se que com recurso a uma câmara de filmar, através de um varrimento ao cartão de cidadão ou de uma matrícula, obter dados correspondentes à pessoa do cartão do cidadão, ou da viatura associada àquela matrícula.

1.4 Estrutura do documento

Este documento encontra-se dividido em oito capítulos. Nesta secção do relatório encontra-se uma breve descrição do que pode encontrar em cada um deles.

O primeiro capítulo, **Introdução**, apresenta a instituição acolhedora do estágio, um pequeno resumo do projeto e os objetivos e problemas que motivaram a realização deste projeto.

O segundo capítulo, **Estado da Arte**, faz referência ao estudo efetuado sobre métodos e formas de registo de ocorrências/denúncias policiais de forma automática, evitando assim que se gaste tempo em desenvolver aplicações ou funcionalidades de aplicações já existentes no mercado. É ainda muito útil, no sentido de auxiliar na melhoria do desenvolvimento de novos conceitos e paradigmas no concerne ao registo de ocorrências/denuncias policias.

O terceiro capítulo, **Metodologia**, pretende-se caracterizar o processo de desenvolvimento de *software* ágil, definir a metodologia adotada, selecionar recursos, delinear e apresentar objetivos.

O quarto capítulo, **Análise de Requisitos**, é descrito o conjunto de processos que visam ilustrar os requisitos, tais como: diagrama de casos de uso, diagrama de contexto, diagramas de sequência, diagrama de classes.

O quinto capítulo, **Tecnologias Utilizadas**, apresenta uma breve descrição das ferramentas e tecnologias utilizadas, tanto no desenvolvimento da GestCop, nomeadamente no desenvolvimento da sua base de dados.

O sexto capítulo, **Implementação**, descreve a arquitetura, os mecanismos de segurança e a forma de implementação da GestCop.

O sétimo capítulo, **Verificação e Validação**, descreve os testes que foram realizados para que fosse garantido a segurança e o seu bom funcionamento.

Por fim nas **Conclusões**, são apresentadas as conclusões do trabalho realizado, os objetivos alcançados, bem como potenciais melhorias futuras.

Nos **Anexos** podemos encontrar figuras e tabelas que complementam o relatório. São ainda apresentados excertos de código utilizado durante o desenvolvimento da GestCop, de modo a facilitar a explicação das suas funções mais importantes.

2 Estado da arte

Após estarem definidos os principais objetivos para a conceção da aplicação, torna-se essencial realizar um estudo exploratório com o intuito de obter informação relevante para o seu planeamento e desenvolvimento. Pretende-se, desta forma, adquirir conhecimento de estudos e técnicas já utilizadas por outros investigadores, tentando replicar as boas práticas e os pontos positivos e melhorar os pontos menos positivos.

Após uma pesquisa na Internet por sistemas de registo de ocorrências policiais, foram analisadas várias aplicações semelhantes à aplicação que se pretende desenvolver, conseguindo-se evidenciar como mais interessantes para este estudo as seguintes: Sistema Estratégico de Informação, Gestão e Controlo Operacional (SEI) (Policia de Segurança Publica, 2020), Portal da Queixa Eletrónica (PQE), ambos desenvolvidos pelo Ministério da Administração Interna (MAI) (Ministério da Administração Interna, 2020) e Registo de Ocorrências Escolares (ROE) , desenvolvido pela Secretaria da Educação do Estado de São Paulo (Secretaria da Educação do Estado de São Paulo, 2020).

2.1 Sistema Estratégico de Informação, Gestão e Controlo Operacional - SEI

O SEI, é uma aplicação onde a PSP regista atualmente todo o tipo de ocorrências/denúncias. Nesta aplicação o elemento policial que regista a ocorrência ou denuncia tem a possibilidade de registar, consultar e adicionar informação ao relatório da ocorrência, ficando vedada a eliminação de qualquer incidência já registada. No SEI é possível fazer consulta de pessoas, locais, veículos ou objetos que tenham sido referenciados em qualquer tipo de ocorrência.

Para alem do registo de ocorrências, queixas e denuncias, o SEI tem ainda como funcionalidades principais:

- A gestão de recursos materiais e humanos, nomeadamente viaturas policiais, rádios, equipas policiais respetiva distribuição por áreas de serviço, entre outras;
- Admite a gestão de horários e assiduidade dos elementos com e sem funções policiais.

Na Figura 1 é demonstrada a interface de início do SEI. Por questões de segurança e de privacidade no que respeita à utilização dos dados, não é possível demostrar mais sobre este sistema informático.

Figura 1- Aplicação – SEI

Fonte: Sistema informático PSP

Nesta aplicação o acesso à informação e funcionalidades está distribuída por níveis acesso, ou seja, o acesso à informação e funcionalidades está diretamente ligada à função atribuída ou desempenhada. Um exemplo deste escalonamento é por exemplo um elemento policial que tem como função executar patrulhamento auto, não consegue aceder ao módulo de atribuição e gestão de escalas do efetivo policial.

É ainda de salientar que durante o registo de qualquer denuncia/ocorrência, este sistema não provê qualquer tipo de ajuda, deixando que seja o elemento policial que está a registar a denúncia/ocorrência a conduzir todo o processo, resultando muita das vezes em lapsos e falhas tanto na aquisição de informação junto da pessoa que está a denunciar a queixa, como na obtenção de meios de prova.

2.2 Registo de Ocorrências Escolares - ROE

O ROE (Secretaria da Educação do Estado de São Paulo, 2020), é uma aplicação desenvolvida no Brasil, destinada aos diretores das escolas, que permite o registo de todas as ocorrências, quer sejam de índole disciplinar ou criminal, que ocorram no estabelecimento de ensino que dirigem.

Neste sistema o diretor pode ainda visualizar, imprimir e editar os relatórios que foram registados.

Na Figura 2, é mostrado a *interface* gráfica do ROE, onde é possível visualizar em cima um menu *wizard* que ajuda no registo das ocorrências.

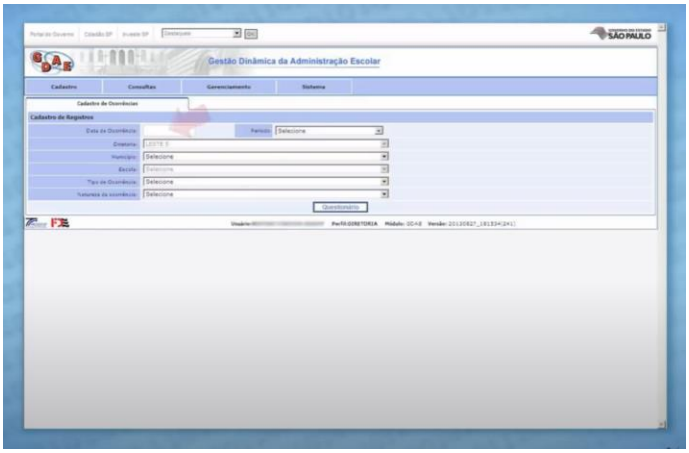


Figura 2 – Aplicação – ROE

Fonte: https://www.youtube.com/watch?v=CNx_gbSuPHY&t=96s

2.3 Portal da Queixa Eletrónica - PQE

O Portal da Queixa Eletrónica (Ministério da Administração Interna, 2020), foi concebido pelo Ministério da Administração Interna (MAI), para facilitar a apresentação de queixas e denúncias às Forças de Segurança de Portugal.

Permite registar queixas e denúncias dos crimes de: ofensas à integridade física simples, violência doméstica, roubo, furto, maus tratos, tráfico de pessoas, lenocínio, dano, burla, burla relativa a trabalho ou emprego, extorsão, danificação ou subtração de documento e notação técnica, danos contra a natureza, uso de documento de identificação ou de viagem alheio, poluição, auxilio à imigração ilegal, angariação de mão de obra ilegal e casamento de conveniência. Nesta aplicação não é possível fazer a denuncia de outros crimes alem destes (Ex: Violação, Rapto, Tráfico de Pessoas, etc.).

Podem apresentar queixa/denúncia qualquer pessoa singular que se identifique corretamente e preencha os campos solicitados no formulário. No final do registo é gerado um relatório que será submetido e enviado para a força de segurança competente. Após submissão não é possível efetuar qualquer tipo de alteração no relato da queixa ou denúncia.

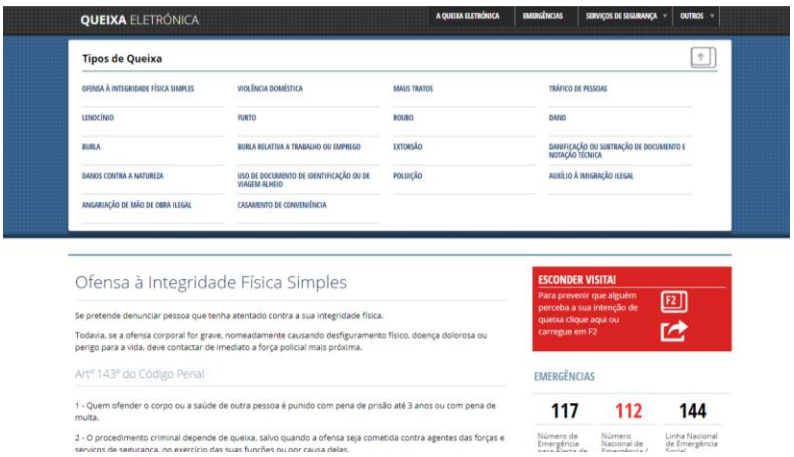


Figura 3 - Aplicação – Portal da Queixa Eletrónica

Fonte: https://queixaselectronicas.mai.gov.pt/SQE2013/default.aspx#tag=MAIN_CONTENT

2.4 Análise Crítica do Estado da Arte

Após uma breve análise destes sistemas, é possível concluir que existem duas grandes linhas orientadoras destes dispositivos eletrónicos. Uma vertente de utilização profissional (SEI e o ROE) e uma vertente para o utilizador comum (PQE), que autoriza a submissão de queixas ou denúncias junto das Forças de Segurança.

Com base na comparação das três aplicações foi elaborado um resumo onde se pode identificar as características que as une ou aproxima e o que as distingue.

Funcionalidades	SEI	ROE	PQE	GestCop
Aplicação responsiva	×	×	×	✓
Cross-Browser	×	✓	✓	✓
Ajuda na escrita do relato da ocorrência	×	×	×	✓
Aplicação Web	✓	✓	✓	✓
Possui aplicação nativa para dispositivos móveis	×	×	×	×
Controlo de nível de acesso aos dados	✓	✓	×	✓
Pesquisar, alterar relatórios existentes	✓	✓	×	✓
Consultar estado de pessoas, veículos, objetos, etc..	✓	×	×	✓
Ajuda no registo da ocorrência	×	×	✓	✓

Gestão de presenças	✓	×	×	×
Gestão de recursos e material	✓	×	×	×

Tabela 1 - Comparação de aplicações de gestão e registo de ocorrências

É de salientar que o SEI é a aplicação mais completa e com mais funcionalidades, no entanto, o ROE e o PQE são aplicações mais intuitivas, facilitando o registo de ocorrências.

No que concerne à portabilidade, o SEI é até ao momento um sistema limitado pois depende única e exclusivamente do *browser Internet Explorer* para funcionar.

Com o desenvolvimento desta nova aplicação pretende-se resolver os problemas de portabilidade e tornar o registo das queixas, denúncias e ocorrências num processo mais amigável com o elemento policial que está a interagir com o sistema. Pretende-se ainda que com este sistema construir um sistema de ajuda que vá guiando todo o processo de registo, minimizando assim falhas e lacunas

3 Metodologia

A metodologia científica visa o conhecimento, baseado em regras e normas bem definidas, com objetivos explícitos que permitem averiguar o rigor e as diretrizes de uma investigação. Assim deverão estar bem definidos a quem se dirige a investigação ou o desenvolvimento (participantes ou amostra), a forma como se irá avaliar ou desenvolver o conhecimento (instrumentos), deverão ainda estar bem definidos todos os passos e procedimentos para que se perceba se os critérios éticos e deontológicos foram respeitados como também o rigor da investigação ou desenvolvimento do produto, sendo ainda fundamental que futuros investigadores possam replicar o estudo partindo das mesmas condições (procedimentos) (Aragão & Neta, 2017; Peixoto, Silva, & Rocha, 2010).

De forma a cumprir estes objetivos durante o desenvolvimento da GestCop foram definidos:

1. **Participantes ou amostra** - Todos os elementos policiais que desejam registar uma ocorrência/denúncia de violência doméstica;
2. **Instrumentos** - Para cumprir este objetivo tornou-se necessário desenvolver mais conhecimento em duas áreas especificas:
    a. Registo de denuncias/ocorrências policias, em que se optou por recorrer à experiência profissional no âmbito da resolução e de registo de denuncias/ocorrências policiais, opiniões de outros elementos policiais, bem como através de análise de *software* já existente;
    b. No desenvolvimento de *software* optou-se por recorrer a conhecimentos adquiridos durante as aulas ministradas no curso, leitura e acompanhamento de livros técnicos sobre programação e desenvolvimento de *software*, bem como o recurso a pesquisas na Internet. No final do desenvolvimento da aplicação pretende-se avaliar este *software* recorrendo à utilização da aplicação por uma pequena amostra de elementos policiais, anotando os pontos fortes e fracos da aplicação

3. **Procedimentos –** Para tentar cumprir estes objetivos, vai-se tentar que todos os passos dados no desenvolvimento desta aplicação fiquem devidamente descritos, fundamentados e bibliografados neste relatório.

Com o passar dos anos, o exponencial desenvolvimento tecnológico conduziu a indústria de desenvolvimento de *software* a patamares nunca antes vistos, com metodologias, técnicas e instrumentos produzidos de forma complexa e multifacetada. Outrora a modelagem de *software* era muito morosa (modelo em cascata, incremental, espiral e *Rational Unified Process* - RUP) o que levava a que muitos projetos caíssem por terra. Para minimizar esta limitação foi criada a metodologia ágil (Pearson, 2011).

3.1 Metodologia de desenvolvimento utilizada

Aproveitando a vantagem de se desenvolver a aplicação junto do cliente final, a PSP, utilizando metodologias centradas no cliente, com a vantagem de se obter o *feedback* num curto espaço de tempo, optou-se por utilizar a metodologia ágil, mais concretamente o processo de desenvolvimento SCRUM para o desenvolvimento desta aplicação.

O ciclo de desenvolvimento de aplicações utilizando esta metodologia é baseado em fases: *Sprint Planning*, *Daily Scrum; Sprint Review;* e *Sprint Retrospective*.

3.2 Metodologia ágil SCRUM

O SCRUM é baseado no controlo empírico de processos e usa um método iterativo e incremental de forma a prevenir eventuais erros e aumentar o controlo de potenciais riscos (Sobrevilla, 2017; Lewis, 2015; Schwaber & Sutherland, 2013). Esta metodologia baseia os seus fundamentos no desenvolvimento de *software* em equipa como uma unidade integrada. Cada membro da equipa desempenha um papel específico tendo em conta um objetivo comum (Carvalho & Mello, 2012; Sobrevilla, 2017).

Esta metodologia assenta em três pilares fundamentais: transparência, inspeção e adaptação. A transparência significa que o processo de desenvolvimento do software deve ser assumindo como um padrão comum, e todos as partes envolvidas neste processo tenham a mesma conceção, definição e entendimento comum. A inspeção tem como objetivo inspecionar e avaliar todo o trabalho desenvolvido, permitindo assim que todas as partes envolvidas estejam mais atentos às variações e resultados indesejados. A adaptação é a plasticidade, a capacidade de se reorganizar tendo em conta o objetivo, onde é permitido um ajustamento ao longo do processo. Os processos de inspeção e adaptação deverão andar lado a lado, e como referido anteriormente poderão existir quatro eventos formais: *Sprint Planning, Daily Scrum*, *Sprint Review* e *Sprint Retrospective*, que traduzido de forma simplista e pouco rigorosa poderá ser entendido como planeamento, acompanhamento diário, a revisão e retrospetiva (Schwaber & Sutherland, 2013)

Algumas das características deste método passam pela flexibilidade de prazos e resultados, participação, cooperação e espírito empreendedor (Carvalho & Mello, 2012; Sobrevilla, 2017).

Ao utilizar esta metodologia, os desenvolvedores de *software* tentam resolver problemas adaptativos e complexos pretendendo gerar valor e conhecimento, sendo considerada uma metodologia que através da utilização de processos e técnicas bem definidas permitem gerir e desenvolver aplicações de uma forma eficaz e fácil. (Schwaber & Sutherland, 2013; Sobrevilla, 2017).

Para melhor compreender a aplicação do SCRUM ao processo de desenvolvimento da GestCop pode-se usar por base uma imagem do seu ciclo de vida (Figura 4).

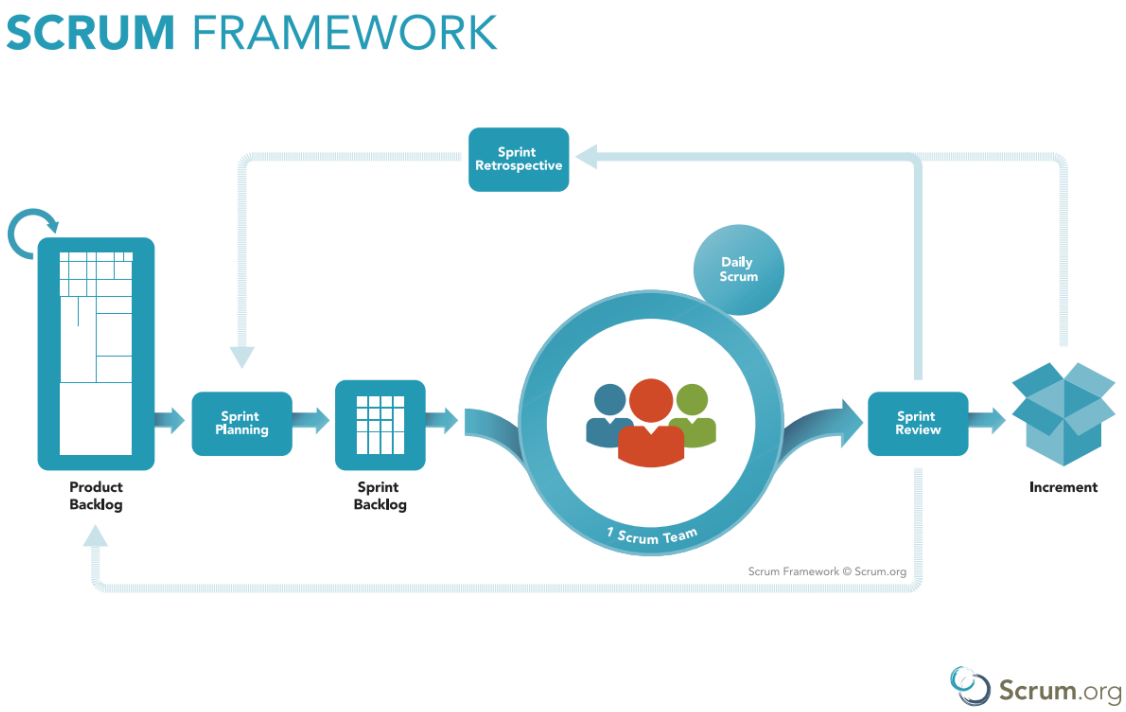


Figura 4 – Ciclo de vida de um processo de desenvolvimento de *software Scrum*

Fonte: http://www.scrumportugal.pt/scrum/

Antes da fase de execução propriamente dita, a *Scrum Team* foi formada pelos seguintes elementos:

- ***Product Owner*** – Neste projeto está representado pelo Superintendente José Leonardo, diretor do DSIC;
- ***Development Team*** – Por este projeto utilizar paradigmas de programação e ferramentas bastante recentes ao utilizado pela equipa de desenvolvedores do departamento de sistemas do DSIC, este projeto contou apenas com o meu trabalho para o desenvolvimento da aplicação;
- ***Scrum Master* –** Por só existir um *developer* o *Scrum Master* não fazia sentido de existir.

Após estar definida a *Scrum Team*, identificados os requisitos da aplicação, inicia-se a fase de ***Sprint Planning***. Nesta fase houve a necessidade reunir com o *Product Owner*, encarregado de dirigir o processo de desenvolvimento da aplicação.

No final desta reunião foi possível definir o seguinte:

- O nome da aplicação. Neste caso optou-se por batizar esta aplicação com o nome: GestCop – Sistema de Gestão e Controlo de Ocorrências Policiais;
- Os requisitos da aplicação ordenados por prioridade;
- A equipa de trabalho (*Developer Team*);
- A aplicação a implementar deveria ser executada através da *Web* para facilitar o acesso à aplicação;
- Como ferramentas de trabalho foi definido a utilização das seguintes bibliotecas *JavaScript*: para a construção da aplicação *backend* a utilização do Node.JS, para construção da aplicação de *frontend* a utilização do React.JS;
- Foi identificado como potencial risco o facto de ser uma aplicação muito extensa, levando a que se demorasse muito tempo na criação de uma aplicação completa. Foi então definido como requisito prioritário a criação de um único módulo da aplicação, referente às ocorrências de Violência Doméstica.

Após esta fase de planeamento formou-se a proposta de arquitetura de desenvolvimento mencionada na Figura 12, do subcapítulo – Arquitetura do Sistema.

Seguidamente foi definido como ***Sprint Backlog*** a construção de uma aplicação *Web* que fosse capaz de registar uma ocorrência ou denuncia de um crime de violência doméstica.

Após estar definido o *Sprint Backlog*, passa-se então à fase de desenvolvimento que é dividida em ciclos, denominados *sprints*. Neste caso foi definido o seguinte:

- Reuniões diárias com o *product owner*, no máximo de 15 minutos *(Daily Scrum)*, para discutir pequenos pontos da implementação;

- Foi definida uma reunião semanal com o Superintendente José Leonardo (*Sprint Review*), para apresentação dos novos trabalhos efetuados e das novas funcionalidades implementadas.

Por não existirem elementos policiais disponíveis para executarem a *Sprint Review* a aplicação será avaliada através de um conjunto de testes unitários efetuados por mim. Após a avaliação efetuada das funcionalidades já implementadas e de melhorias a realizar (*Sprint Retrospective*), caso haja necessidade, voltar-se-á à fase de *Sprint Planning*, com o objetivo de potenciar o rendimento do próximo *sprint*.

É expectável que por volta da trigésima semana, não existam alterações a fazer, estando a aplicação a funcionar corretamente e de acordo com os requisitos definidos.

Definida a metodologia a utilizar, segue-se a análise e recolha pormenorizada de requisitos do sistema.

4 Análise de requisitos

Para o desenvolvimento de sistemas informáticos deve proceder-se à recolha da informação relativa às funcionalidades expectáveis da aplicação por forma a responder às necessidades do utilizador. Esta recolha de informação deverá ter por base três critérios base, sendo eles critérios funcionais, não funcionais e facilidade de utilização (*usability*) (O'Neil, 2004).

No que concerne às técnicas utilizadas, dever-se-á atentar: às reuniões com o *Product Owner*, recurso a *Brainstorming* com grupos de elementos policiais, à minha experiência profissional no âmbito da resolução de ocorrências policiais e observação direta de amostras de documentos e relatórios.

4.1 Requisitos funcionais

Os requisitos funcionais num sistema informático representam um conjunto de serviços que se espera que o sistema disponibilize ao utilizador.

Feito este levantamento, em concordância com o *Product Owner*, foi definido que o *Sprint Backlog* inicial do sistema devia conter os seguintes requisitos:

1. Criar, alterar, consultar ocorrência;
    1.1. Criar, alterar e consultar as seguintes notificações:
        1.1.1. Estatuto de vítima;
        1.1.2. Preservação de imagens de videovigilância;
        1.1.3. Reconhecimento Fotográfico;
    1.2. Criar, editar, consultar os seguintes autos:
        1.2.1. Termo de identidade e residência (TIR)
        1.2.2. Auto de apreensão;
        1.2.3. Auto de entrega;
    1.3. Criar, editar, consultar avaliação de risco de vítima de violência doméstica;

1.4. Criar, editar, consultar locais de ocorrência;

1.5. Criar, editar, consultar pessoas;

1.6. Criar, editar, consultar objetos;

4.2 Requisitos não funcionais

Os requisitos não funcionais abrangem todas as características qualitativas do sistema, por exemplo, medidas de desempenho, tais como, tempos de resposta, volume de dados ou considerações de segurança (O'Neil, 2004).

De acordo com o levantamento de requisitos não funcionais é espectável que o sistema cumpra o seguinte:

- Deve ser um sistema executável em vários ambientes, quer seja a nível de software (Windows, Linux, Android, etc.), quer seja a nível de hardware (Computadores Pessoas, Portáteis, Smartphones, Tablets, etc.);
- Fácil na evolução e na manutenção;
- Ter grande disponibilidade e uma baixa taxa de falhas;
- Ter um bom desempenho;
- Ser um software seguro.

4.3 Requisitos de facilidade de utilização

Os requisitos de facilidade de utilização, garantem que existirá uma boa ligação entre o sistema e o utilizador recorrendo a sistemas de ajuda do próprio sistema (O'Neil, 2004).

Após a seleção e identificação dos requisitos, chegou-se à conclusão que o sistema necessitaria ainda de uma vertente intuitiva e interativa, ou seja, o sistema tem de ser fácil de aprender e manusear, levando a que o processo de registo da ocorrência/denúncia seja feito de um modo fácil, para prevenir a ocorrência de erros e lapsos esse registo deve de ser feito de uma forma mais interativa entre elemento policial e sistema.

Com o objetivo de apresentar os requisitos do sistema e a sua funcionalidade, utilizaram-se os diagramas de casos de uso. Estes diagramas têm a finalidade expor a informação de forma sectorial, graficamente enquadrada por forma a torná-la mais percetível e assim encontrar consensos entre o utilizador final e o analista do sistema (O'Neil, 2004).

Para simplificar a construção deste diagrama é necessário identificar quais são as entidades externas (Atores) a interagir no produto final.

4.4 Ator

Considerando que a nossa aplicação está a ser desenvolvida para utilização em meio policial e será uma aplicação de teste, não existindo diferenciação de quem a vai utilizar, só há o seguinte ator:

> Elemento policial – Utilizador que pode, mediante uma série de registos e buscas, registar, alterar, consultar a ocorrência. Pode ainda obter os mais variados autos e notificações em utilização até ao momento pela PSP em todo o território nacional, assim como após terminar o registo da ocorrência obter o respetivo Auto de Notícia.

4.5 Funcionamento da GestCop

Antes de proceder a implementação, é fundamental perceber o que é, e em que consiste o registo de ocorrências/denúncias, sejam elas criminais ou contraordenacionais.

O processo de registar uma ocorrência/denúncia consiste em documentar todo um conjunto de informação e tarefas importantes para que seja possível determinar se o crime aconteceu ou não, quem e em que circunstâncias cometeu o crime, proteger as vítimas e levar à responsabilização do(s) autor(es).

O registo de ocorrência/denúncias como já foi transcrito no Capítulo 1, tem como objetivo levar a notícia da existência de um crime ao MP, para que este possa investigar a existência do crime, determinar os seus agentes e a sua responsabilidade, bem como recolher provas para que se possa decidir acusar ou não (Pissarra, 2018).

Para se proceder ao registo de toda esta informação, é necessário recolher vários dados que se podem organizar nos seguintes grupos:

- Pessoas intervenientes no cenário criminal – suspeitos, lesados, vítimas, etc.;
- Objetos que tenham sido utilizados para o cometimento do crime – armas, veículos, objetos eletrónicos, etc.
- Locais onde o crime ocorreu ou que esteja diretamente relacionado com o crime;
- Processos efetuados para a recolha de indícios criminais – recolha de imagens de um sistema de videovigilância, fotografias recolhidas no local do crime, apreensão de objetos que tenham sido utilizados para o cometimento do crime, etc.;
- Tipo de elícito cometido – crime contra as pessoas, crime contra a propriedade, crime contra o estado, etc.
- Espaço temporal em que o crime foi cometido – data e hora a que foi cometido, se não souber ao certo tem de ficar registado um espaço temporal em que supostamente foi cometido o crime;
- Ações que legalmente e obrigatoriamente têm de ser desenvolvidas quando estamos perante um determinado ilícito criminal.
- Descrição de tudo o que se consegue apurar no local do crime.

Com o objetivo de demonstrar como estes conceitos estão interligados efetuou-se o fluxograma da Figura 5.

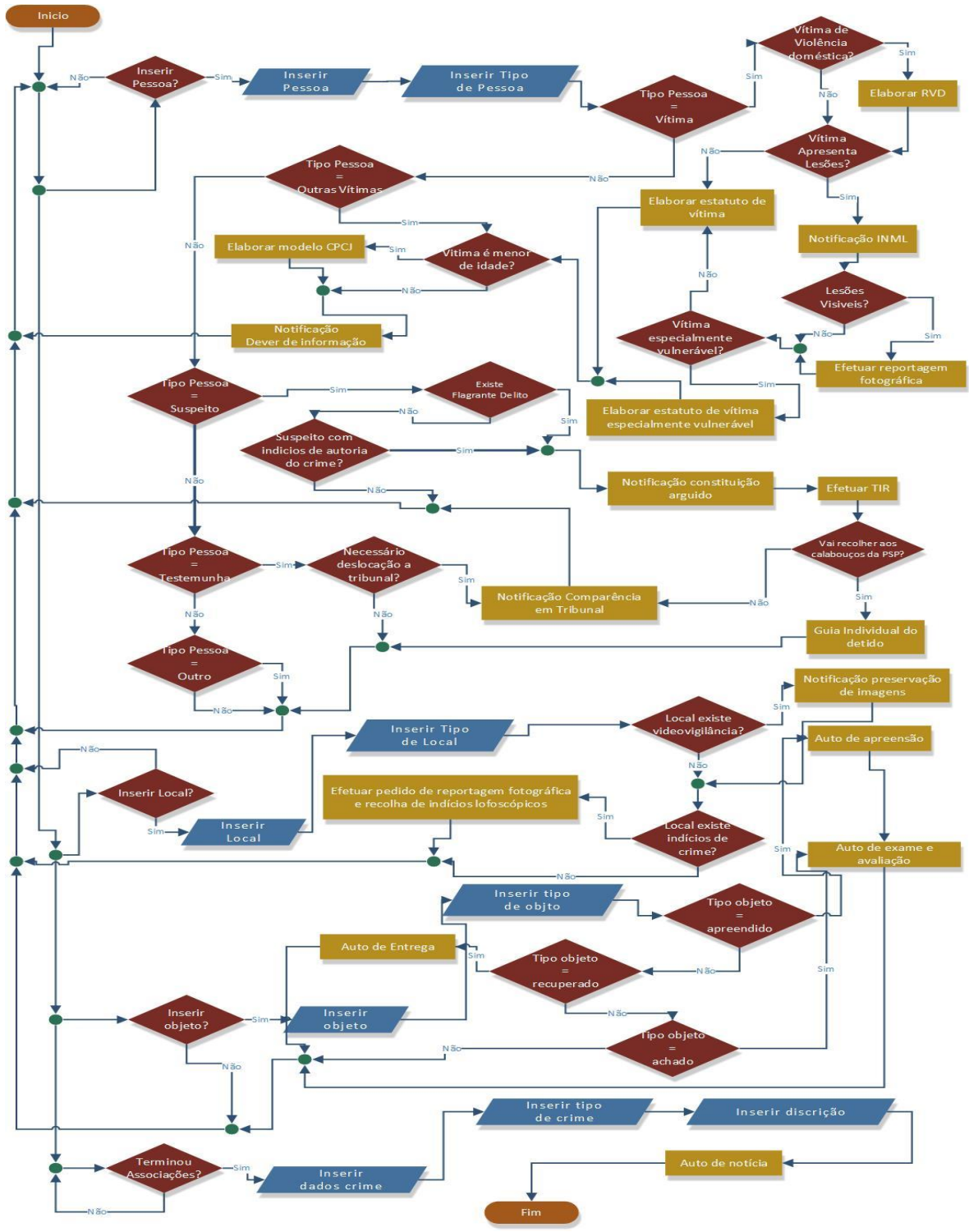


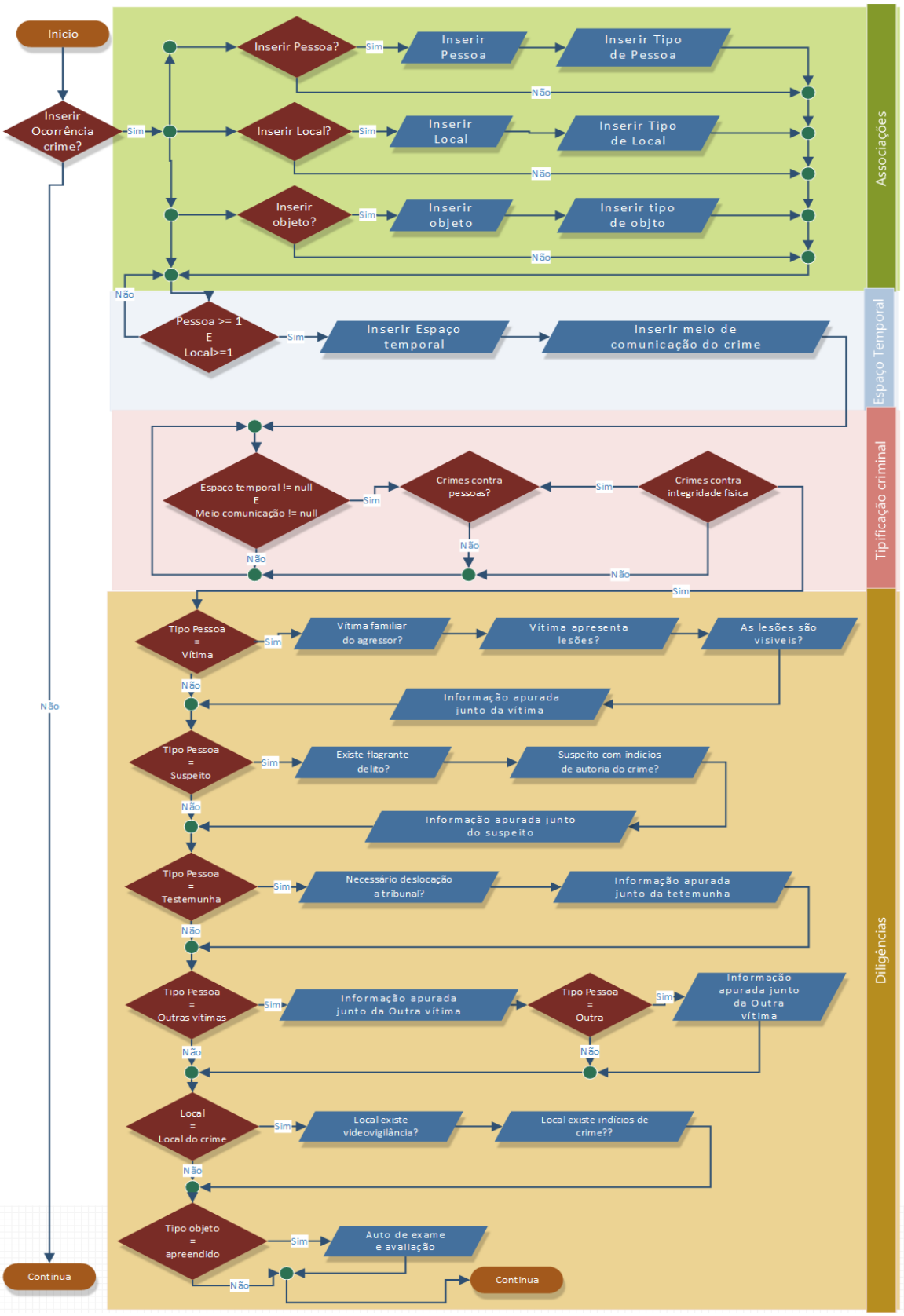
Figura 5 - Fluxograma de Inserir ocorrência/denúncia

Comos se pode observar, este é um processo deveras complexo e que pode levar a erros, omissões se o elemento policial não for bem guiado, tal como se pretende fazer com a GestCop. Para que este processo de registo se torne mais fácil e amigável com o elemento policial que está a registar a ocorrência/denúncia, pensou-se em dividir este registo em vários passos, recorrendo à conceção de um formulário do tipo *Wizard* aquando da sua implementação na GestCop.

Com base no fluxograma da Figura 5 e o conceito da aplicação funcionar com o menu *Wizard*, procedeu-se ao desenho do fluxograma da Figura 6 - Fluxograma de funcionamento e seccionamento do menu *Wizard*, ilustrativo da maneira do funcionamento da aplicação e do seccionamento dos passos do menu *Wizard.*

Este menu *Wizard*, encontra-se dividido por seis partes:

1. Associações – Nesta parte vão ser associados todas as pessoas, locais e objetos que estão diretamente relacionados com aquele crime.
2. Espaço temporal – Aqui será registado o espaço temporal em que supostamente ocorreu o crime;
3. Tipificação criminal – Nesta parte será registado o tipo de crime que foi cometido, bem como foi cometido;
4. Diligências – Aqui será registado todo o tipo de ações que foram feitas (algumas serão sugeridas pelo sistema) para a recolha de indícios criminais (Ex: recolha de indícios lofoscópicos (impressões digitais, recolha de ADN, etc.)), bem como de ações determinadas pelo CPP (Ex.: notificação do estatuto de vítima, notificação do direito de informação, etc.)
5. Descrições – Nesta parte será descrito tudo o que foi possível apurar do local do crime e da interação com as diversas pessoas envolvidas no cenário criminal;
6. Documentos – Aqui será a parte onde o elemento poderá fazer o *download* e a impressão de todos os documentos gerados automaticamente pelo sistema.


Inicio
Inserir Ocorrência crime?
Sim
Não
Inserir Pessoa?
Sim
Inserir Pessoa
Inserir Tipo de Pessoa
Não
Inserir Local?
Sim
Inserir Local
Inserir Tipo de Local
Não
Inserir objeto?
Sim
Inserir objeto
Inserir tipo de objto
Não
Associações
Não
Pessoa >= 1 E Local>=1
Sim
Inserir Espaço temporal
Inserir meio de comunicação do crime
Espaço Temporal
Espaço temporal != null E Meio comunicação != null
Sim
Crimes contra pessoas?
Sim
Crimes contra integridade fisica
Não
Não
Não
Sim
Tipificação criminal
Não
Tipo Pessoa = Vítima
Sim
Vítima familiar do agressor?
Vítima apresenta lesões?
As lesões são visiveis?
Não
Informação apurada junto da vítima
Tipo Pessoa = Suspeito
Sim
Existe flagrante delito?
Suspeito com indícios de autoria do crime?
Não
Informação apurada junto do suspeito
Tipo Pessoa = Testemunha
Sim
Necessário deslocação a tribunal?
Informação apurada junto da tetemunha
Não
Tipo Pessoa = Outras vítimas
Sim
Informação apurada junto da Outra vítima
Tipo Pessoa = Outra
Sim
Informação apurada junto da Outra vítima
Não
Não
Local = Local do crime
Sim
Local existe videovigilância?
Local existe indícios de crime??
Não
Tipo objeto = apreendido
Sim
Auto de exame e avaliação
Não
Continua
Continua
Diligências

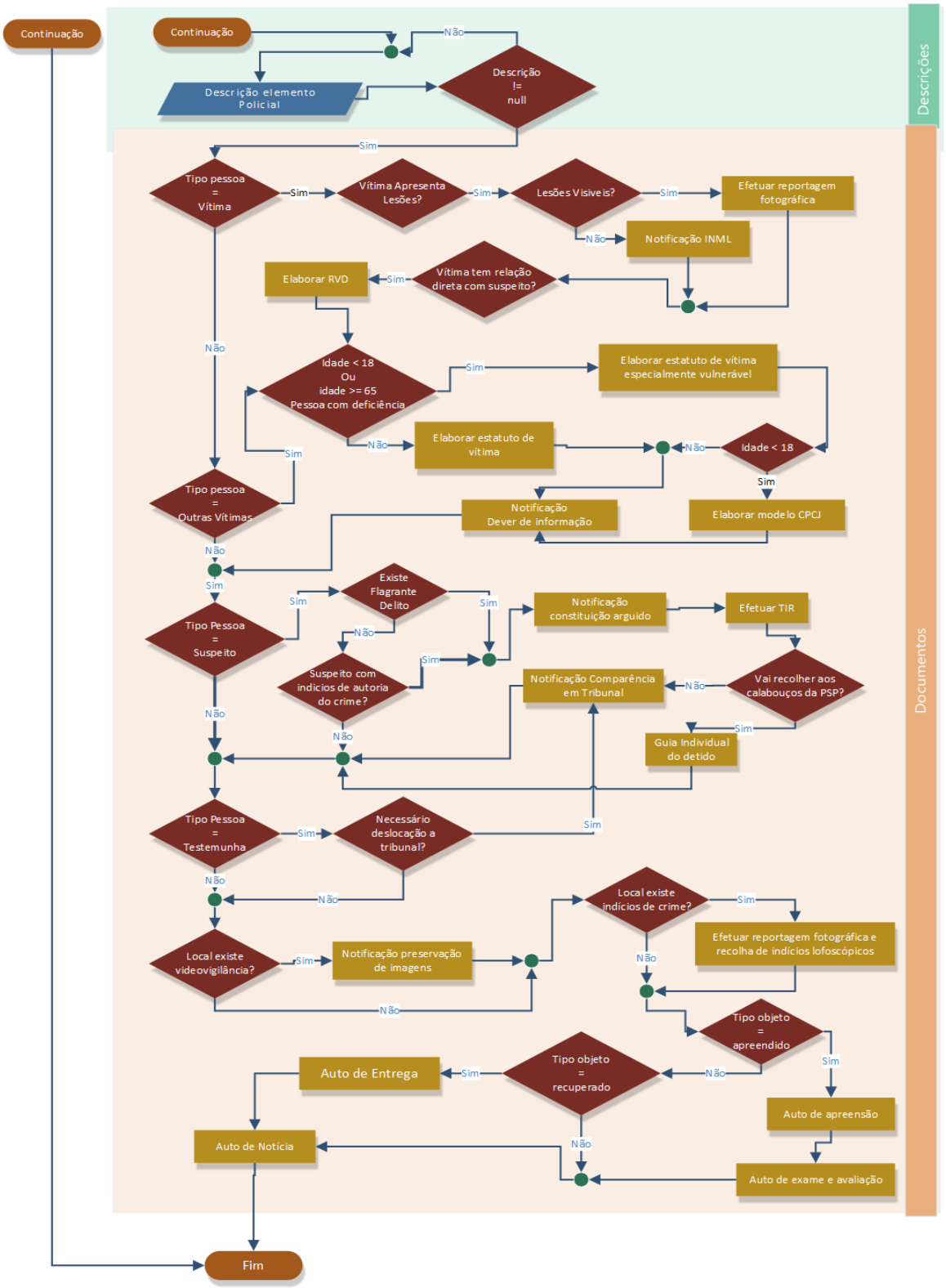


Figura 6 - Fluxograma de funcionamento e seccionamento do menu *Wizard*

4.6 Modelo Entidade-Relacionamento

Neste subcapítulo podemos ver na Figura 7 - Modelo ER, o modelo Entidade Relacionamento (ER) da aplicação GestCop. Este modelo representa o desenho conceptual da BD e mostra como as diferentes entidades e seus atributos se relacionam entre si.

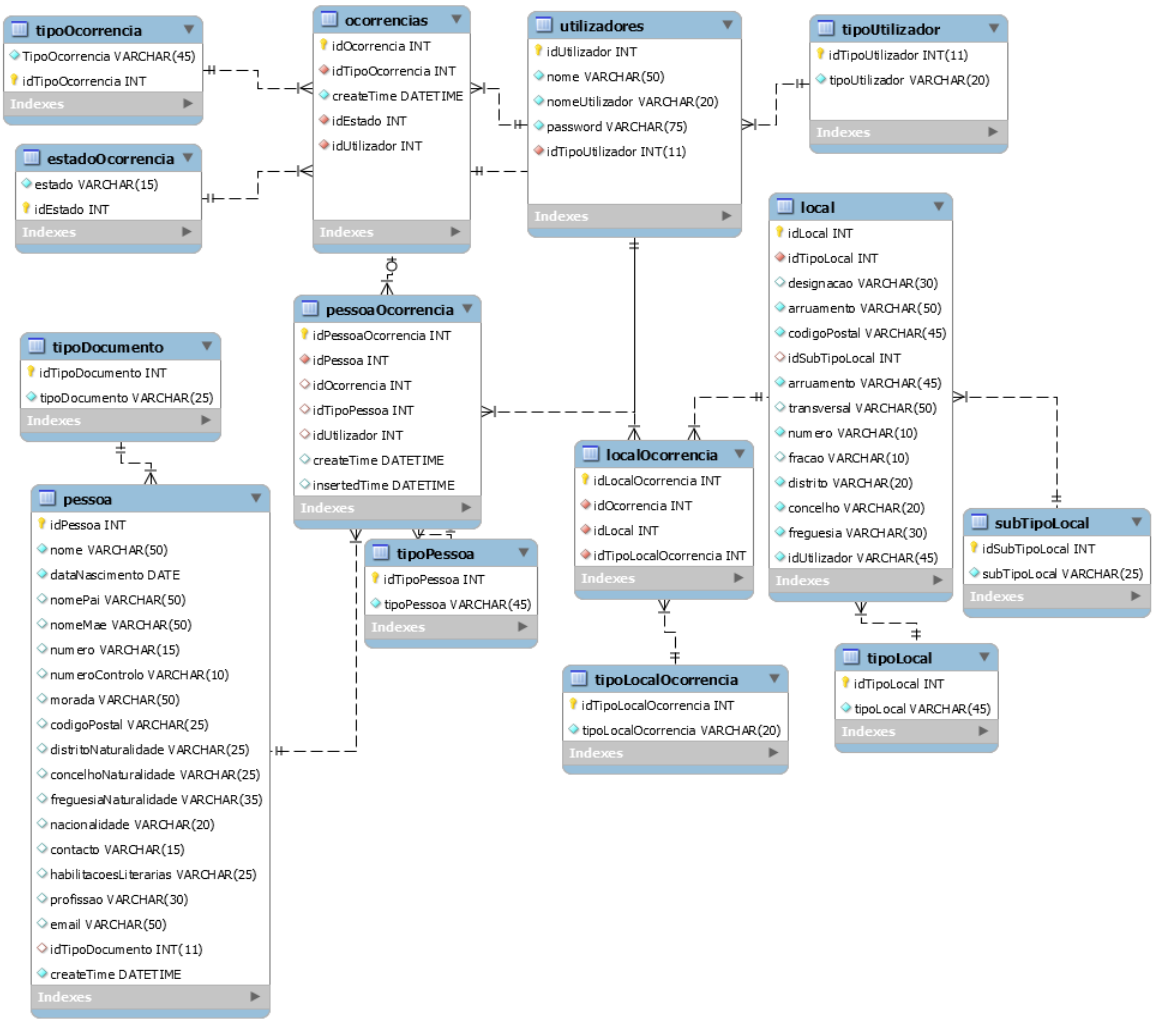


Figura 7 - Modelo ER

Uma das entidades mais importante deste modelo ER é a entidade Ocorrência. Esta entidade junta todos os dados essenciais para que seja possível inserir uma ocorrência/denúncia.

Ainda é possível visualizar, as duas tabelas de utilizadores do sistema, utilizadores e tipoUtilizador, que foram criadas com o intuito de registar os utilizadores da GestCop. Todas as tabelas restantes são utilizadas para armazenar a informação referente às associações de pessoas e locais.

Para o desenvolvimento da BD da GestCop, foi utilizada a metodologia de ir adicionando tabelas à medida que fosse necessário, por não se ter conseguido desenvolver toda a aplicação, não foram criadas mais tabelas.

5 Ferramentas e Tecnologias Utilizadas

O mundo da programação para a *Web* segue o modelo tradicional cliente-servidor conseguindo fazer uma divisão de responsabilidades por parte dos programadores para a *Web*, ou seja, este modelo é materializado em dois conceitos:

- ***Frontend*** – Parte que fica do lado do cliente da aplicação *Web*, isto é, a parte da aplicação *Web* que interage diretamente com utilizador (Queirós & Portela, 2018), que se encarrega da componente gráfica e desenho (Arciniegas, 2016);
- ***Backend* –** Parte que trabalha do lado do servidor, isto é, a parte que interage com as bases de dados e com os demais serviços (Queirós & Portela, 2018), mais ligado à programação e engenharia de *software* (Arciniegas, 2016).

Assim, o processo de construção de uma aplicação está interligado a estes dois conceitos, dando suporte aos dispositivos móveis e realizam a revisão dos elementos que permitem supervisionar a estrutura técnica e a facilidade de utilização, ou seja, a medida de qualidade da experiência que o utilizador tem quando interage com o produto ou sistema (Arciniegas, 2016).

Seguindo este modelo para o desenvolvimento desta aplicação e tendo em conta a popularidade, performance, e interação entre as mesmas, foi efetuado um breve estudo às tecnologias mais usadas atualmente no mercado.

Este estudo baseou-se essencialmente na análise de tabelas e gráficos de rankings elaborados pelas mais diversas empresas de análise estatística do setor de tecnologias e informação.

Através desta análise foi possível constatar o seguinte:

- Segundo o Website w3techs, é possível denotar que a linguagem de programação mais utilizada neste paradigma cliente-servidor é a linguagem PHP (W3teches, 2020). No anexo10.3.1, é apresentado um gráfico que demostra esta evidência. Interessa

referir que segundo empresa de análise RedMonk, a linguagem mais utilizada durante o primeiro trimestre de 2020 é a linguagem JavaScript (RedMonk, 2020). No Anexo 10.3.2 pode-se visualizar o gráfico que contem este ranking;

- Através da página Web, www.hotframeworks.com, é possível averiguar que a framework mais popular neste momento para a programação Frontend é a framework React que é implementada através do uso da linguagem JavaScript (Hotframeworks, 2020). No Anexo 10.3.3, encontra-se o gráfico representativo da popularidade da utilização de frameworks para frontend;
- Através da página, w3techs, é ainda possível averiguar que o Apache é o servidor Web mais utilizado (w3techs, 2020). No Anexo 10.3.4, encontra-se o gráfico demonstrativo desta realidade. Nesta página ainda é possível verificar que quando é necessário tratar um grande volume de informação o servidor mais utilizado é o Node que é implementado com recurso à linguagem JavaScript (w3techs, 2020). O Anexo 10.3.5, demonstra o gráfico correspondente a esta utilização;
- Ainda na página, w3techs, é possível aferir que a linguagem de marcação mais utilizada é HyperText Markup Language (HTML) (w3techs, 2020), a utilização de Cascading Style Sheets (CSS) (w3techs, 2020) e do BootSrap (w3techs, 2020) são utilizados para o desenvolvimento da grande maioria das aplicações Web. Nos Anexos 10.3.6, 10.3.7 e 10.3.8, é possível visualizar o gráfico demonstrativo destas realidades;

Com base neste estudo, foram definidas como tecnologias a utilizar para a concepção do fronted o ReactJS, HTML, CSS e BootStrap, para o desenvolvimento do backend foi definido Node.JS e MySql.

Feito o estudo das tecnologias a utilizar, prossegue-se nos próximos subcapítulos com uma breve descrição das ferramentas e tecnologias que serão utilizadas no desenvolvimento desta aplicação.

5.1 Ferramentas

Para o desenvolvimento de qualquer aplicação *Web* é necessário aplicar tecnologias e boas práticas de gestão de projetos. Por forma a aplicar estas tecnologias torna-se necessário a utilização de algumas ferramentas que suportem e apoiem o desenvolvimento.

Para o desenvolvimento e gestão desta aplicação tornou-se necessário a utilização das ferramentas que se passam a descrever.

5.1.1 Visual Studio Code

O Visual Studo Code (VSC) é um editor de código (Integrated Development Environment (IDE)), criado pela Microsoft em abril de 2015 e é suportado pelos principais sistemas operativos (Windows, Linux, Mac OS) (Portela & Queirós, 2018).

A escolha deste IDE deve-se ao facto de já estar familiarizado a trabalhar com esta ferramenta, ser de utilização gratuita, possuir a completação inteligente de código (*IntelliSense*) e de ter integração com o sistema de controlo de versões Git.

5.1.2 GIT

O Git é um sistema de controlo de versões, que permite arquivar ficheiros de um projeto (código-fonte e outros), registar alterações durante o desenvolvimento, desfazer alterações ou recuperar versões anteriores e separar troncos de desenvolvimento (ex.: produção/desenvolvimento). Este sistema permite ainda sincronizar o código-fonte entre diferentes computadores graças ao sistema de comunicação distribuída, facilitando assim a cooperação entre programadores (Portela & Queirós, 2018).

De forma a manter e agilizar todos o processo de escrita e manutenção do código da aplicação foi escolhido o Git como sistema de controlo de versões. A escolha desta ferramenta deve-se

ao facto de este ser um sistema de utilização gratuita, de código aberto e o mais utilizado a nível mundial.

5.1.3 GITHUB

O GitHub é um sistema de gestão de repositórios remotos que permite a hospedagem de código-fonte para repositórios) (Portela & Queirós, 2018). Estes repositórios podem ser armazenados de forma privada, ficando acessível só ao autor do código-fonte, ou de forma pública, ficando o código-fonte acessível a qualquer pessoa que queira visualizar ou fazer download.

A escolha da utilização do sistema GitHub, deve-se ao facto de ser uma ferramenta gratuita e por ser uma das ferramentas mais usadas a nível mundial. O repositório deste projeto encontra-se hospedado de forma privada em: https://github.com/RuiAgostinho/GestCop

5.1.4 Heroku

O Heroku permite desenvolver e implementar aplicações na *Web*. É uma plataforma como serviço, ou *Platform as a Service* (PaaS) (Portela & Queirós, 2018), que consegue disponibilizar uma infraestrutura (servidores, armazenamento e rede), e um conjunto de ferramentas de programação, serviços, Sistemas de Gestão de Bases de Dados (SGBD), entre outros, através de um serviço disponível na *cloud*. Através da utilização desta plataforma consegue-se evitar custos e complexidade inerentes à aquisição de licenças de *software* e infraestrutura, que seriam necessários para que a nossa aplicação fique disponível na Internet (Microsoft, 2020).

A seleção da utilização desta ferramenta deve-se ao facto de ser de muito fácil utilização, gratuita e conseguir executar as nossas aplicações *backend* em Node.JS e *frontend* em React.JS. A aplicação está disponível *online* em https://gestcop.herokuapp.com.

5.1.5 POSTMAN

O POSTman permite testar qualquer tipo de pedido HTTP (p. ex.: os métodos GET, POST, PUT, DELETE) (Portela & Queirós, 2018).

No desenvolvimento desta aplicação o POSTman foi utilizado como ferramenta de teste e *debugging* da aplicação *backend*, sendo escolhido por ser simples de utilizar, gratuito e muito completo.

5.1.6 MySQL Workbench

O MySQL Worckbench é uma ferramenta *open source* da Oracle e permite a modelação e administração de BDs MySQL, permite ainda o desenvolvimento de *Scripts* em linguagem SQL (Oracle, 2020).

O principal facto da utilização desta ferramenta deveu-se ao facto da BD desenvolvida para esta aplicação utilizar a linguagem MySQL e por ser uma aplicação robusta, gratuita e desenvolvida pela mesma empresa.

5.1.7 NPM

O NPM (Node Package Manager), é um utilitário de código aberto que serve para gerir pacotes (conjunto de ficheiros e pastas) JavaScript. Este utilitário é uma ferramenta utilizada para partilha ou reutilização de código JavaScript (Luís Abreu, 2016).

A gestão destes pacotes é feita através de um ficheiro designado por “package.json” que contem metadados que caracterizam o pacote (Luís Abreu, 2016). Dos diversos metadadosos possíveis de encontrar no ficheiro package.json, podemos encontrar metadados relativos ao nome e à versão de pacotes de terceiros instalados e que são fundamentais para a nossa aplicação funcionar.

Este gestor de pacotes foi utilizado para garantir que todos os pacotes essenciais ao funcionamento da nossa aplicação sejam importados ao instalar o GestCop, bastando para isso instalar o utilitário NPM e dentro da pasta do projeto executar o comando – npm install.

O principal motivo que levou à eleição deste gestor recaiu sobre o facto de ser um utilitário mais maduro em comparação com outros gestores de pacotes, e o facto de ter o apoio de uma grande comunidade de desenvolvedores de código.

5.2 Tecnologias - *Frontend*

Para o desenvolvimento da *interface* gráfica da aplicação, foram utilizadas as seguintes tecnologias.

5.2.1 HTML

O HTML (*Hyper Test Markup Language*), é uma linguagem de marcação (*markup*) de texto utilizada para descrever e definir o conteúdo de uma página *Web* (texto, imagens, etc.). Esta marcação HTML inclui elementos especiais denominados *tags* (ex: <html>, <head>, <h1>, etc.), que os navegadores *Web* se limitam a interpretar e a exibi-los de forma interativa para o utilizador final.

Esta linguagem utiliza ainda um conceito muito importante que permite a interligação de várias páginas, o hipertexto ou hiperligações (*links*). É através deste conceito que o utilizador através de um clique do rato pode visualizar páginas *Web* interrelacionados (Queirós & Portela, 2018).

Para o desenvolvimento desta aplicação optou-se por utilizar o HTML 5.2 como linguagem base das nossas páginas *Web*, pois este código consegue ser executado por qualquer *browser*

5.2.2 CSS

A CSS (*Cascading Style Sheets*) é uma linguagem de estilização *standard* e atualmente suportada por todos os *browsers.* É utilizada para adicionar estilos (ex.: cores, fontes, espaçamentos, animações, etc.) ao conteúdo de uma página *Web* (Queirós &Portela, 2018) Esta linguagem tem como principal atributo separar o formato do conteúdo, com o intuito de evitar falhas na apresentação (Lima, 2016).

Para se recorrer a estilizações personalizadas no desenvolvimento da nossa aplicação foi utilizada a CSS 3.2.

5.2.3 BootStrap

O BootStrap é uma *framework* JavaScript, HTML e CSS de código aberto (*open source*), e segue a filosofia *Mobile First* (Prates, 2015). Esta filosofia está assente sob a ideia de que se devem criar todas e quaisquer *interfaces* gráficas a pensar primeiro no ecrã dos dispositivos móveis e só à posteriori no ecrã dos PCs (Wroblewski, 2011).

O BootStrap, disponibiliza uma grande quantidade de componentes de interface (ex.: barras de navegação, botões *dropdown*, agrupamentos de botões, etc.), *plugins* (ex.: botões, abas, alertas, etc.), que foram pensados e concebidos para que o desenvolvimento de páginas *Web* responsivas seja feita de uma forma rápida e fácil.

Graças à facilidade de utilização destes componentes de interface e *plugins* recorreu-se à utilização do Bootstrap 4.4, para o desenvolvimento da nossa aplicação.

5.2.4 React.JS

O React.Js foi lançado em 2013, é uma *framework* JavaScript de código aberto desenvolvida pelo Facebook (Antonio, 2015). Além desta versão do React.JS, existe uma versão específica

para desenvolvimento de interfaces para dispositivos móveis, o React Native (Vipul & Sonpatki, 2016).

Esta *framework,* utiliza um conceito em que a aplicação é toda carregada pelo *browser* aquando da primeira requisição do utilizador, depois só são atualizados os componentes que mudam de estado, sem necessidade de recarregar toda a página. Este conceito é denominado por *Single Page Aplication* (SPA) (Ricardo Queirós & Filipe Portela, 2018).

Com a utilização deste conceito de SPA, pretende-se que o carregamento e a interação entre o utilizador e o sistema seja feito de uma forma mais rápida e fluida.

O React.JS utiliza ainda um conceito de blocos (em React.JS chamados componentes) fáceis de reutilizar, estender e manter, facilitando assim a construção da interface visual da aplicação. Estes componentes, para além de possuírem a parte visual que o utilizador vai ver, possuem ainda toda a parte lógica da programação e os dados que serão usados aquando do desenho das *interfaces* gráficas da aplicação.

O React baseia-se na ideia que a manipulação da *Document Object Model* (DOM), que é a árvore de elementos HTML de uma página *Web* que é criada pelo navegador quando a página é carregada, é uma operação muito complexa e que deve ser minimizada. Para reduzir ao máximo estas operações de manipulação da DOM, o React faz uso de uma DOM virtual que compara com a DOM real, no final só redesenha os componentes necessários para alcançar o novo estado, tornando assim a aplicação mais rápida e fluida (Vipul & Sonpatki, 2016).

Este processo de atualização e desenho da DOM real desenrola-se da seguinte forma:

1. Para cada objeto DOM real, o React cria um objeto Dom virtual igual como cópia livre. Este objeto virtual tem as mesmas propriedades do objeto real, mas não tem a capacidade de alterar o desenho da *interface* gráfica. É através desta copias que o React cria o seu próprio DOM virtual;

2. Quando ocorre uma mudança num componente o React compara o DOM virtual com o DOM virtual instantâneo. Este processo é chamado de “Comparação”;
3. Após o React calcular quais os objetos que sofreram alterações, o React atualiza esses objetos, e só esses objetos, na DOM real (CodeCademy, 2020) .

A Figura 8, apresenta um esquema visual deste processo de atualização de objetos na DOM real.

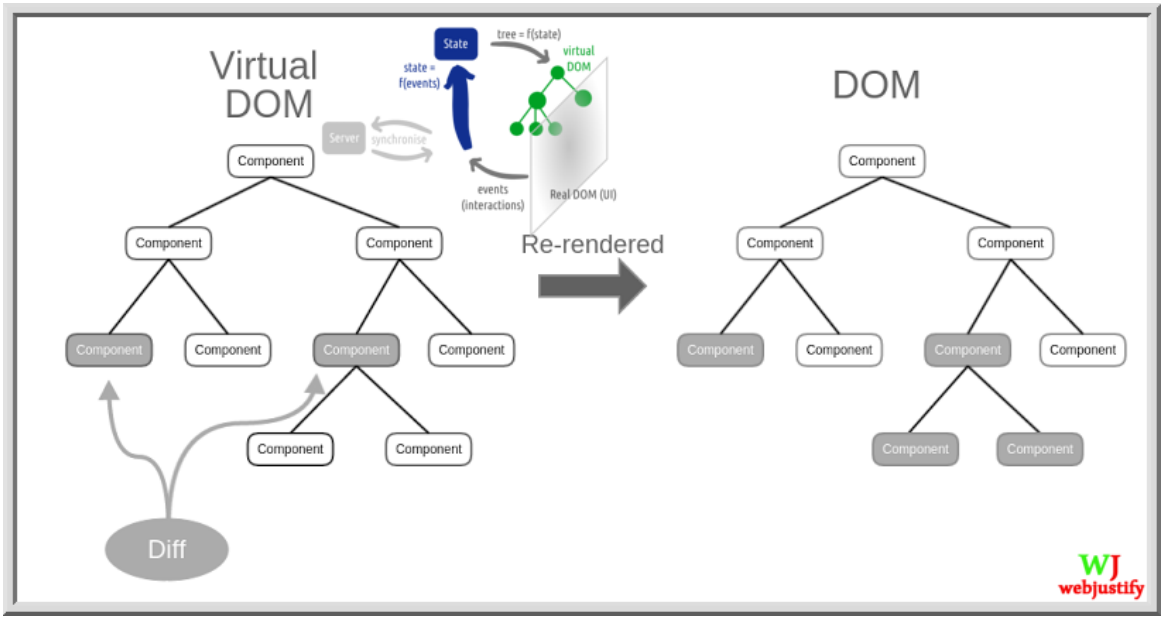


Figura 8 - Virtual DOM vs DOM

Fonte: https://danushkapereracom.wordpress.com/2019/02/28/react-js/

5.2.5 Redux

Com a utilização do Redux pretende-se resolver o problema da partilha de estados entre componentes do React. Esta partilha de estados do React está representada na Figura 9 do lado esquerdo, onde é possível visualizar que o estado de um componente é passado de

componente em componente (do componente pai para o componente filho), até chegar ao(s) componente(s) a que se destina(m) .

Com a implementação de uma arquitetura Redux é implementada uma *Store* (sítio onde é gerido o estado global da aplicação), e é esta *Store* que faz a atualização dos estados dos componentes afetados. Na Figura 9 do lado direito está representada esta forma de atualização de estados (Maruta, 2018).

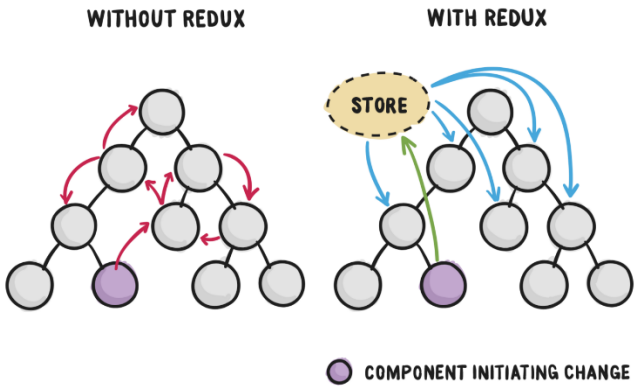


Figura 9 - Controlo de estado React e React-Redux

Fonte: https://medium.com/reactbrasil/iniciando-com-redux-c14ca7b7dcf

A arquitetura Redux foi desenvolvida baseada em três grandes princípios (Khuat, 2018):

1. O estado da aplicação é guardado num único objeto, designado por *Store*. Esta *Store* é um objeto JavaScript que centraliza o estado de todos os componentes da aplicação e torna mais fácil a gestão e passagem de estado para os componentes da aplicação;
2. O estado da aplicação é inalterável. A única forma de alterar um estado é através de *Actions*;
3. *Reducers* especificam como é que a *Action* vai modificar o estado do(s) componente(s). *Reducers* são funções JavaScript que evoluem o estado da aplicação. Estas evoluções podem agir em parte ou em todo o estado do componente. Estes *Reducers* podem ser combinados e reutilizados.

A Figura 10, demonstra o fluxo de uma evolução de estado utilizando Redux. Esta evolução de estado é despoletada quando o Elemento Policial executa uma *Action* através de uma interação com um elemento HTML (ex.: clicar no botão adicionar para adicionar uma pessoa). Esta *Action*, vai acionar um Middleware chamado Dispacher, que faz a ligação entre a *Action* despoletada e o *Reducer* pretendido. Aquando do acionamento deste *Reducer* a *Store* vai evoluir de estado, o que leva a que os componentes que foram afetados por esta mudança de estado evoluam, o que faz com que ao evoluírem de estado e tenham de ser redesenhados na *interface* gráfica.

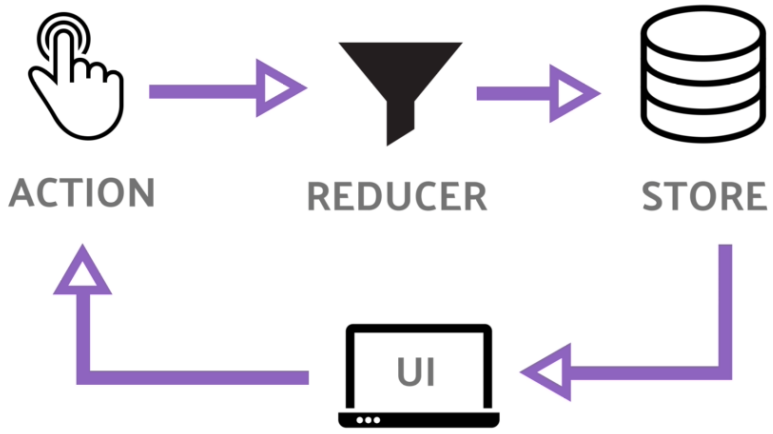


Figura 10 - Evolução de estado Redux

Fonte: https://itnext.io/integrating-semantic-ui-modal-with-redux-4df36abb755c

5.2.6 Outros módulos

Para que o desenvolvimento desta aplicação fosse mais rápido e eficaz, recorreu-se ainda a vários outros módulos desenvolvidos por outras pessoas ou empresas. Segue-se uma breve descrição dos módulos utilizados.

Para o desenvolvimento do layout da aplicação utilizaram-se os seguintes módulos:

- **Fontawesome** – Para a utilização de ícones e componentes gráfico pré elaborados através de estilos CSS (https://fontawesome.com/);

- **AdminLTE, Bootstrap e Jquery** – Para a construção do layout em si (https://fontawesome.com/);
- **Toastr** – Para aprestação de mensagens ao utilizador (https://www.npmjs.com/package/toastr);
- **History** – Para o redireccionamento de páginas (https://www.npmjs.com/package/history);
- **Axios** – Cliente HTTP para fazer requisições ao *Backend* (https://www.npmjs.com/package/axios);
- **Redux-Form** – Para facilitar a obtenção de dados vindos dos formulários (https://www.npmjs.com/package/redux-form);
- **Redux-multi** – Usado para que através de uma *action* do Redux seja possível interagir com vários *Reducers* do Redux (https://www.npmjs.com/package/redux-multi);
- **Redux-promise** – Para gestão de ações assíncronas (https://www.npmjs.com/package/redux-promise);
- **Redux-thunk** – Permite criar encadeamentos de ações (https://www.npmjs.com/package/redux-thunk);

5.3 Tecnologias - *Backend*

Para a modelação e controlo da informação proveniente do *frontend*, informação esta que foi introduzida pelo utilizador, foi desenvolvido o *backend* recorrendo as seguintes tecnologias:

5.3.1 Node.JS

O Node.JS, é um ambiente de *runtime* JavaScript, que permite a execução de código JavaScript fora do *browser* (Luís Abreu, 2016), ou seja, através desta *runtime* consegue-se desenvolver e executar aplicações escritas em JavasScript do lado do servidor.

A escolha desta *framework* deveu-se ao facto desta *framework* tirar o máximo partido das técnicas utilizadas no motor de *runtime* JavaScrit V8 da Google, o que faz com que a execução do código seja feita de uma forma simples, extremamente rápida, e com uma grande performance. (Luís Abreu, 2016).

A arquitetura do Node.JS é uma arquitetura *Single-Thread* que recorre ao ciclo de eventos (*event loop*) para facilitar a criação de aplicações fiáveis e escaláveis. Esta abordagem à utilização de ciclos de eventos é suportada por bibliotecas que permitem a realização de operações de entrada e saída (I/O) de forma assíncrona, ou seja, por exemplo uma simples consulta na base de dados não resulta num bloqueio do sistema, em vez disso o Node.JS notifica a aplicação de que já tem os dados disponíveis através da geração de um evento. Este comportamento faz com que o Node.JS, não tenha bloqueios (Luís Abreu, 2016).

Para controlar o processamento de eventos, o Node.JS recorre a um ciclo de eventos que é gerido como uma fila *First In First Out* (FIFO), ou seja, o primeiro evento a chegar é o primeiro a ser processado. Este ciclo de eventos é automaticamente iniciado assim que a aplicação Node.JS é iniciada.

O Node.JS recorre à utilização de um modulo especial chamado libuv, que em conjunto com a lógica do Node.JS utiliza para realizar as operações assíncronas e gerir uma *Thread Pool* libuv. Esta *Thread Pool* é composta por quatro *Pools* que são utilizadas quando existem tarefas muitos pesadas para serem tratadas no *loop* de eventos. Operações de I/O, operações de abertura e fecho de conexões, operações de BD, entre outras são exemplos de operações pesadas para serem tratadas neste *loop* de eventos (AritaSen, 2020).

Quando a *Tread Pool* conclui uma tarefa, é chamada uma função de retorno (função de *callback*), a qual trata os erros (se existirem), ou executa outra operação que tem a executar. Esta função de *callback* é enviada para a fila de eventos. Quando a fila de chamadas está vazia, o evento passa pela fila de eventos e envia o retorno para a fila de chamadas (AritaSen, 2020).

A Figura 11, representa a arquitetura de uma *event loop* existente num servidor Node.Js

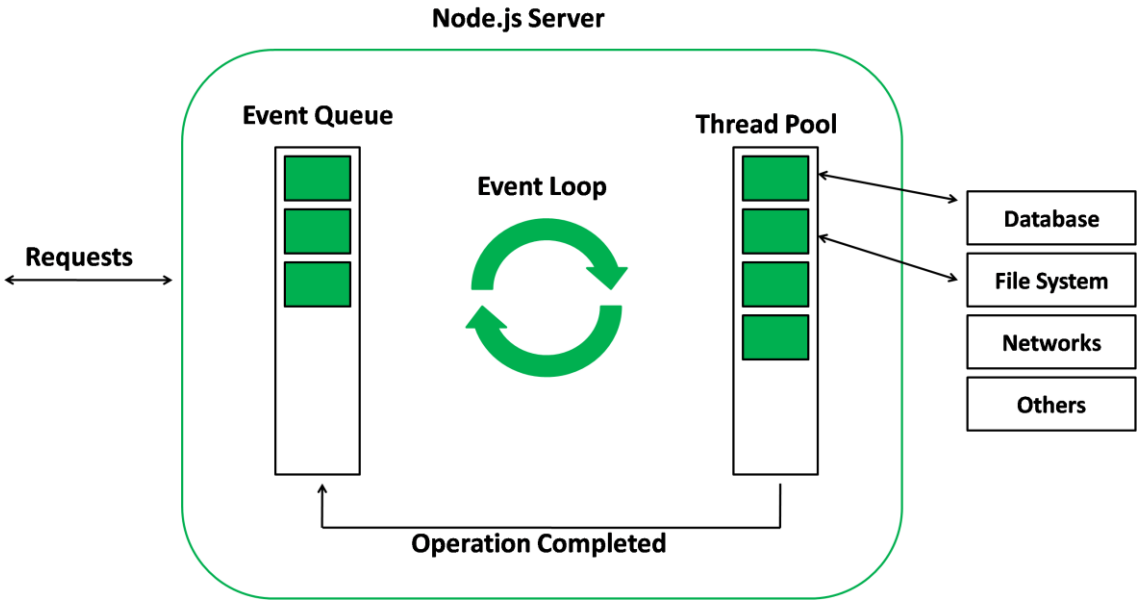


Figura 11 - Representação do *loop* de eventos num servidor Node.Js

Fonte: https://www.geeksforgeeks.org/node-js-event-loop/

Graças ao sistema de módulos que o Node.JS implementa, torna-se fácil a organização e gestão de código. Este conceito de módulos permite reutilizar código durante o desenvolvimento da aplicação. Podemos encontrar três tipos de módulos (Abreu, 2016):

- Módulos core – São módulos base que o Node.JS disponibiliza e que em caso de conflito são sempre carregados em detrimento de outros módulos. Um exemplo disso é o modulo de leitura de ficheiros (módulo FS).
- Módulos pasta – São módulos criados pelo próprio programador, normalmente organizado dentro de pastas.

- Módulos de terceiros – Estes módulos normalmente encontram-se organizados dentro da pasta *NODE_MODULES* que é controlada pelo utilitário NPM.

Para o desenvolvimento desta aplicação foi necessário o uso dos seguintes módulos de terceiros:

- **Bcrypt** – Utilizado para em conjunto com uma chave pré-definida pelo programador cifrar e comparar a senha introduzida pelos utilizadores;
- **Body-Parser** – É responsável em converter o corpo das requisições para vários formatos. No nosso caso para JavaScript Objet Notation (JSON);
- **Cross-Origin Resource Sharing (CORS)** – Controla o acesso a recursos da aplicação;
- **DotEnv** – Usado para criar as nossas variáveis de ambiente;
- **Express** – Gere requisições de endereços URL (*Uniform Resource Locator*);
- **HTTP Status** – Utilizado para interagir com códigos de estados HTTP;
- **JSON Web Token** – Responsável por gerar o *token* para autenticação de utilizadores;
- **JWT Simple** – Utilizado para codificar/descodificar a assinatura do *token*;
- **Mysql** – Driver responsável pelas requisições e ligação à base de dados;
- **Passport** – Módulo que gere a autenticação de utilizadores;
- **Passport JWT** – Módulo que gere a autenticação de utilizadores mediante a utilização de *tokens* (escolhida para esta aplicação);
- **Sanitize** – Usado para remover código HTML inserido por utilizadores da aplicação.

5.3.2 MySQL

O MySQL é um sistema de gestão de Bases de Dados (SGBD) relacionais, *Open-Source* adquirido e mantido pela Oracle. Este sistema é reconhecido pela sua velocidade, fácil uso, segurança devido ao uso de SSL, capacidade de *multi-threading*, entre outras características que o levam a que seja um dos sistemas de gestão de Bases de dados relacionais mais utilizados no desenvolvimento de aplicações *Web* (DuBois, 2013).

6 Implementação

Identificado o problema a resolver, feito o estudo da análise de requisitos, determinada a metodologia a adotar, após se ter determinado um conjunto de ferramentas e tecnologias a utilizar, pretende-se neste capítulo descrever alguns dos passos tomados para tornar a GestCop em realidade.

6.1 Arquitetura da GestCop

Um sistema informático é composto por vários elementos e a forma como está projetada a sua arquitetura ajuda a compreender como esses elementos estão organizados, integrados e interligados entre si.

Para fazer a interligação entre os vários componentes, a aplicação GestCop foi concebida assente numa arquitetura de *Web Services.* Esta arquitetura é baseada numa tecnologia que permite usufruir de vários serviços disponíveis em outras aplicações *Web* e que podem estar a ser executados em diferentes plataformas, permitindo assim que diversos programas possam interagir uns com os outros. A razão pela qual foi escolhida esta arquitetura para implementação nesta aplicação recaiu sobre o facto de se poder construir a aplicação de uma forma mais flexível, modular e eficiente.

Ao utilizar a arquitetura de *Web Services* conseguiu-se separar a aplicação em duas grandes aplicações, uma aplicação de *frontend*, implementada com recurso à biblioteca React.JS, e uma aplicação de *backend,* implementada com recurso à biblioteca Node.JS. A aplicação de *frontend* é a aplicação responsável pela *interface visual* da aplicação e pela recolha de dados do utilizador, a aplicação de *backend* é responsável por implementar as regras de negócio e efetuar as operações necessárias na BD. Estas duas aplicações ficarão alojadas na plataforma HEROKU, onde ficaram disponíveis para acesso dos utilizadores via *Web*. Na Figura 12, pode observar a arquitetura do sistema desenvolvido.

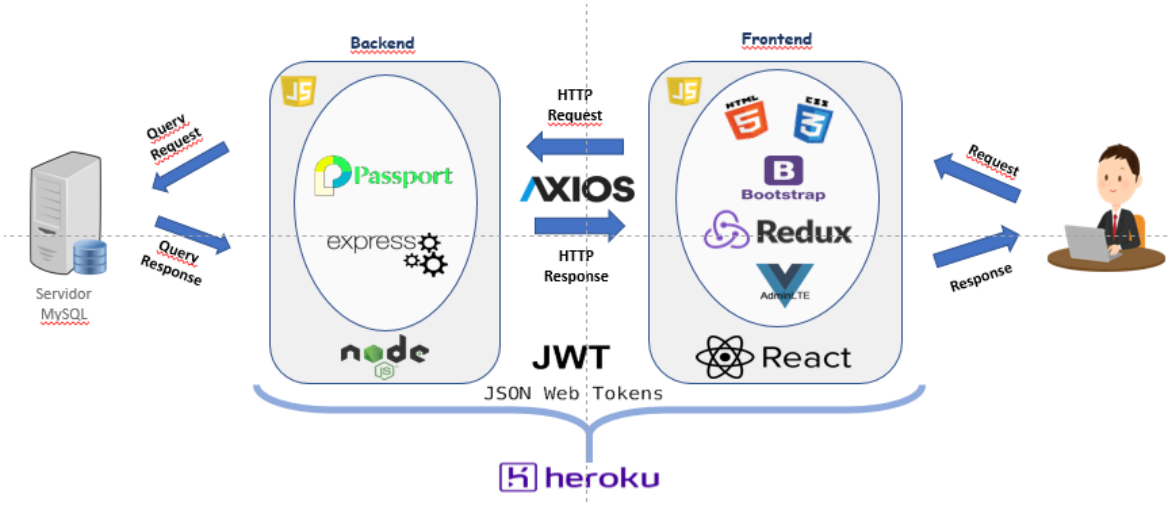


Figura 12 - Arquitetura do Sistema

De forma a simplificar a comunicação entre estas duas aplicações utilizou-se o protocolo de comunicações *Representational State Transfer* (REST), implementado através da utilização do módulo AXIOS. Este protocolo permite que a aplicação de *frontend*, através de do protocolo TCP/IP consiga efetuar pedidos HTTP para aceder ou modificar recursos à aplicação de *backend*, esta por sua vez envia uma resposta com os dados pretendidos ou em caso de alteração de recursos uma mensagem de sucesso ou falha na concretização desta alteração.

Para a implementação do protocolo REST foi necessário cumprir três requisitos:

1. Definição de um método HTTP que vai definir a operação a ser realizada. Estes métodos podem ser do tipo GET, POST, PUT e DELETE, que representam o conjunto de operações que se pretende efetuar no *backend*: o método GET para consultar dados, POST para enviar dados, PUT para fazer alterações e DELETE para apagar um determinado dado;
2. Um cabeçalho onde o *frontend* passa a informação para o *backend*;
3. Um URL (*endpoint*) com o caminho para o recurso a ser utilizado no *backend.*

Na Figura 13 e no Anexo 10.5.1, está representada a implementação de uma solicitação do tipo POST ao *backend*. Esta implementação foi efetuada com recurso à função POST disponível na biblioteca AXIOS, que automaticamente define o campo do *Request Method* existente no cabeçalho do pedido HTTP como POST, e através dos parâmetros é lhe passado o URL (*endpoint*) e os dados que se deseja transmitir para o *backend*.

No Anexo 10.4, apresenta-se o cabeçalho do pedido HTTP que foi originado através da utilização da função apresentada na Figura 13. Analisando este anexo, consegue-se visualizar entre muitos outros campos, o *Request Method* definido como POST, o URL (*endpoint*) definido como https://gestcop.herokuapp.com/ocorrencia/id, no *Request Payload* a informação {id:2}, que no fundo, é a informação que se pretende transmitir para o *backend*.

```
export function procuraOcorrenciaId(id){
    return dispatch => {
        console.log("entrei na action procura ocorrencia por id "+id)
        axios.post("ocorrencia/id", {'id': id})
        .then(resp => {
            console.log(resp.data)
            dispatch([{type: "OCORRENCIA_STATE", payload: resp.data}])
        })
        .then(()=>{
            history.push('/crime')
        })
        .catch(e => {
            console.log(e)
            e.response.data.erros.forEach(
                erros => toastr.error('Erro', erros))

        })
    }
}
```

Figura 13 - Exemplo Solicitação POST ao *Backend*

Como foi dito acima, por cada solicitação HTTP do *frontend* espera-se sempre uma resposta do *backend*. Na Figura 13, é possível verificar que após a solicitação HTTP a função espera que seja devolvido os dados vindos do *backend*, ou caso ocorra um erro apresenta uma mensagem de erro. No Anexo 10.5.1 pode visualizar outra requisição do *frontend*, e no Anexo 10.5.2 a resposta a esse pedido.

Após estar determinado o protocolo que permite efetuar a comunicação entre *frontend* e *backend* torna-se necessário estabelecer o padrão do formato da troca de dados e informações entre as duas aplicações. Neste tipo de aplicações, está padronizado que o formato destas mensagens devem fazer recurso aos formatos padrão XML ou JSON. Na aplicação GestCop tomou-se como padrão o uso da linguagem JSON por ser um formato leve no momento da troca de dados e por ser um formato fácil de escrever e ler.

Na Figura 14 está representado uma mensagem trocada entre o *frontend* e o *backend* da aplicação GestCop. Através desta figura é possível ver que o formato deste tipo de mensagens é simples e clara de entender, pois recorre ao envio de uma coleção (objeto) de pares nome/valor, estes objetos podem ainda ser transmitidos sob a forma de Vetores, Arrays, listas ou sequências de objetos.

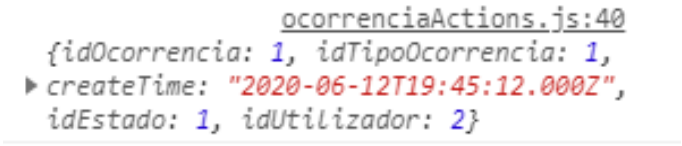


Figura 14 - Mensagem no formato JSON

Para que esta interligação entre o *frontend* e o *backend* se proceda de forma segura, utilizou-se um mecanismo de autenticação de utilizador denominado, JSON Web Token (JWT), com recurso à implementação da biblioteca Passport (Passportjs, 2020).

Este mecanismo de autenticação processa-se da seguinte forma:

1. O elemento policial insere o nome de utilizador e a *password* nos campos do formulário de *login;*
2. A aplicação de *backend* valida o nome de utilizador e a *password*. Se o nome de utilizador e a *password* tiverem sido inseridas corretamente, devolve uma mensagem para o *frontend* com um Token que codifica os dados do elemento policial;
3. No *frontend* este Token é guardado no LocalStorage do *Browser;*

4. Cada vez que o *frontend* faz um pedido ao *backend* envia o Token no campo *Authorization* do cabeçalho do pedido HTTP. Se o *backend* validar o Token, efetua as operações que tem a efetuar (ex.: aceder a dados provenientes da BD), envia os dados pretendidos ou a mensagem de erro ou sucesso da operação para o *frontend.* Na Figura 15 está um exemplo de um Token transmitido no pedido HTTP efetuado com o recurso à função da Figura 13;
5. Por sua vez, cada vez que o elemento policial pretende aceder a uma página que requer autenticação, o *frontend* envia uma mensagem com um pedido de autenticação ao *backend*, só depois de validado este pedido pelo *backend* é que o elemento policial acede à página pretendida.

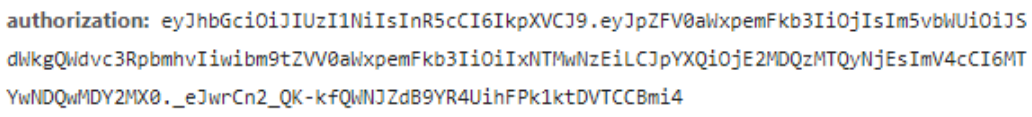


Figura 15 - Exemplo de Token no cabeçalho de um pedido HTTP

6.2 React-Redux

Para que seja possível implementar o Redux, é necessário implementar quatro novos conceitos (Maruta, Iniciando com Redux em 9 passos, 2018):

1. *Store* – É o local onde está armazenado o estado geral da aplicação, ela é imutável, apenas evolui de estado. Cada aplicação Redux tem apenas uma *Store* e é baseada no conceito – Única fonte de verdade (*Single Source of True*). No Anexo 10.7.1, pode-se observar como está implementada na aplicação desenvolvida;

2. *Actions* – São simples funções que ao serem executadas acionam os *Reducers*. Como exemplo temos na aplicação desenvolvida, a ação, entrarAction, que quando o utilizador clicar no botão de Entrar do formulário de Entrada na aplicação, vai enviar

ao *Reducer*, entrarReducer, os dados inseridos nos campos do formulário. Pode observar-se como foi implementada esta ação no Anexo 10.7.2

3. *Reducers* – Os *Reducers*, recebem e tratam os dados enviados pelas *Actions*¸que por sua vez a enviam à *Store*. Na aplicação desenvolvida temos como exemplo o *Reducer*, entrarReducer, que recebe a informação vinda da actionEntrar, que por sua vez a vai guardar na *Store* (no objeto utilizador). No Anexo 10.7.3, pode-se visualizar como foi implementado este *Reducer* na aplicação.

4. Ligação entre os componentes do React com o Redux – Para que sejam disparadas as *Actions* e haver a evolução de estados da *Store* é necessário que o componente seja ligado ao Redux. No Anexo 10.7.4 pode ver-se como o nosso componente, Entrar está ligado ao Redux graças à função *connect*.

6.3 Segurança do sistema

Os mecanismos de segurança são cada vez mais importantes nos dias de hoje. Por trabalhar informação sensível torna-se premente que os dispositivos de segurança assumam os mais elevados padrões. Existem medidas de segurança quer ao nível do utilizador, com a certificação individualizada; quer através de permissões e níveis de acessos que preenchem os critérios ligados com a necessidade de conhecer e a função desempenhada. A robustez do sistema tem igualmente de estar preparada para ataques externos, pois como se referiu trabalha-se diariamente com informação confidencial que obrigatoriamente necessita de ser protegida (Portela & Queirós, 2018).

Vários são os autores que alertam para os perigos cibernéticos, salientando que deverá existir uma grande interação e colaboração interdisciplinar. Estes autores fazem a distinção entre a lei de proteção de dados e o conhecimento (por vezes escasso), dos programadores no que concerne às questões de ordem jurídico-legal (Falco, et. al., 2019).

Por outro lado, estes dispositivos são criados para serem utilizados no âmbito da sua profissão, contudo não lhe é exigido que sejam *experts* em conhecimentos informáticos, tendo a sua maioria conhecimentos na ótica do utilizador. Desta forma, torna-se ainda mais importante que as medidas de segurança se encontrem em pleno funcionamento para que não permita ataques, por exemplo, da engenharia social, ou seja, a exploração dos pontos mais fracos da segurança – os utilizadores - através da exploração da informação ou o descarregar de *software* malicioso (Bullée, Montoya, Pieters, Junger. & Hartel, 2018).

De acordo com os especialistas em segurança, a sociedade atual quanto mais dependente estiver da tecnologia maior será a ameaça da engenharia social para qualquer sistema de segurança. Apraz referir, que os problemas de segurança são tratados normalmente por engenheiros informáticos como situações técnicas, sendo criticada por alguns autores, acrescentando estes, que se está a desvalorizar ou a subestimar o fator humano e em contexto organizacional este deverá ser um fator a ter em conta. Os erros cometidos são normalmente não intencionais e poderão ter origem no utilizador pela utilização deficiente e inadequada, ou induzidos externamente (normalmente mais gravosos), pois poderão deixar o sistema permeável (Bullée, et al., 2018).

Com o objetivo de bloquear este tipo de ataques foram implementados os seguintes mecanismos de segurança:

6.3.1 SQL e Node.js *injection*

De acordo com Chang (2009), grande parte do *software* destinado a proteger contra-ataques maliciosos são insuficientes.

A maioria dos utilizadores ainda usa uma versão de base de dados relacional, ligada ao SQL (*Structured Query Language*). A grande vantagem prende-se com a manutenção da informação sobre as pessoas e as suas características, podendo fornecer informação longitudinal e dados estatísticos (Chowkwanyun, 2019).

O SQL *injection* utiliza a técnica de injetar código SQL como sendo texto num campo de introdução de dados de um qualquer formulário, aproveitando assim a oportunidade para que código SQL injetado seja interpretado pelo interpretador de destino (o Servidor) como uma instrução SQL. Este tipo de ataques tem como principais objetivos: destruir a base de dados, perda ou corrupção de dados, bloqueio de acesso ao proprietário dos dados e acesso indevido aos dados da aplicação (Portela & Queirós, 2018).

Para evitar este tipo de ataques podem ser utilizados dois métodos:

- O **connection.escape** do *package* MySql, que fornece funcionalidades para validar corretamente a entrada do utilizador, como por exemplo, à utilização segue-se um pedaço de código onde é utilizado este método;

- Na construção desta aplicação foi escolhido e utilizado o método de parametrização de instruções SQL. Existe a função de pesquisa de utilizadores pelo nome usando este método (Anexo 10.8).

O SQL não é a única maneira de efetuar ataques *injection*. O Node.js também é alvo deste tipo de ataques. Um método utilizado para concretizar este tipo de ataques é quando o programador recorre à função eval(), que é uma função de avaliação do conteúdo que lhe foi passado como parâmetro. Ao utilizar esta função podem ser injetadas instruções maliciosas, que com a utilização desta função acabam por ser executadas (Portela & Queirós, 2018).

Para que a aplicação não fique vulnerável a este tipo de ataques optou-se por nunca utilizar a função eval().

6.3.2 Quebras de autenticação e gestão de sessões

Este tipo de ataques utiliza fugas ou falhas na autenticação de utilizadores e na gestão de sessões. Os ataques ocorrem quando é extraída informação sobre os dados da autenticação,

conseguidos de forma ilícita, apoderando-se a pessoa que comete este crime das credencias de acesso de um determinado utilizador acedendo à aplicação.

A Figura 16, é ilustrativa do que foi referido anteriormente.

Figura 16 - Ataque de autenticação e gestão de sessões

Fonte: https://gbhackers.com/broken-authentication-and-session-management/

Para que a aplicação que estamos a desenvolver não fique exposta a este tipo de ataques foram utilizados os seguintes métodos:

- Proteção de *password* através da utilização do *package* **Bcrypt** para cifrar a password recorrendo a um *hash* forte. No Anexo 10.9, pode ver-se como foi implementada este método de segurança;
- Para evitar que fosse intercetado o ID de Sessão e a execução de *scripts* através de *Cookies*, não foram utilizados Cookies nem Sessões, sendo feita a autenticação de utilizadores recorrendo ao modulo **Passport** e utilizando tokens *Json Web Token* (JWT) para transmitir de forma segura informação entre o *frontend* e o *backend*. Ao

longo do Anexo10.10, pode ver-se a implementação da atribuição de tokens aos utilizadores, verificação e validação do Token no *frontend* e no *backend*.

6.3.3 *Cross Site Scripting* (XSS)

Esta vulnerabilidade aproveita o facto de o utilizador ter confiança no *site* que deseja usar e ocorre quando um atacante, injeta código (*Scripts*) malicioso no *browser* do cliente ou na aplicação web, de forma a conseguir capturar os dados inseridos pelo utilizador. Este tipo de vulnerabilidade pode ser implementado de duas formas (Ricardo Queirós & Filipe Portela, 2018):

1. De forma não persistente (ou refletida), em que o atacante necessita que seja executado um trecho de código no *browser* do utilizador para assim depois poder aceder ao conteúdo da informação transferida entre o site e o utilizador. Na Figura 17, está ilustrado um ataque deste tipo;

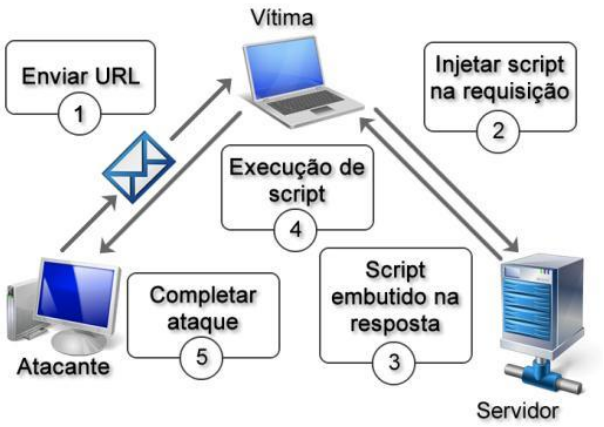


Figura 17 - Ataque não persistente XSS

Fonte: https://www.researchgate.net/figure/Figura-1-Exemplo-de-um-Ataque-XSS-nao-persistente-No-passo-1-o-atacante-envia-um_fig1_299344993

2. De forma persistente (ou armazenado), em que o atacante injeta código malicioso no *site* de destino ficando guardado na base de dados, ficheiros, etc. Neste tipo de ataque,

normalmente o utilizador é levado a clicar num *link* malicioso da página para correr esse código, que vai enviar para o atacante os dados do utilizador (Portela & Queirós, 2018). Na Figura 18 está representado um ataque deste género.

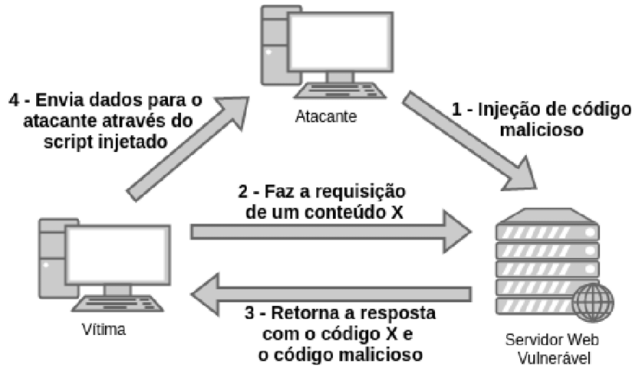


Figura 18 - Ataque Persistente XSS

Fonte: https://www.researchgate.net/figure/Figura-1-Representacao-de-ataque-XSS-armazenado-32-Ataques-XSS-Refletido-O-XSS_fig1_319205710

O desenvolvimento da aplicação parte do princípio que toda a informação inserida deve ser tratada como mal-intencionada, por esta razão optou-se por validar, proteger (*escape*) e higienizar (*sanitize*) todas as entradas de dados dos utilizadores, recorrendo ao uso do *package* **Sanitize.** A utilização deste *package* previne que seja introduzido código malicioso pelo utilizador (Ricardo Queirós & Filipe Portela, 2018).

De forma a permitir a manipulação e troca de dados apenas de fontes confiáveis, neste caso do domínio "herokuapp.com", recorreu-se ao *package* **helmet**. No Anexo 10.11.2, pode ver-se que o sistema só permite o acesso de recursos vindos da plataforma Heroku.

6.3.4 *Cross-Origin Resource Sharing* (CORS)

Este tipo de ataques utiliza os cabeçalhos (*headers*) HTTP para informar o *browser* sobre se uma aplicação *Web*, em execução numa origem (domínio), tem ou não permissão para aceder aos recursos externos que tenham origem num domínio diferente. A Figura 19, é ilustrativa

da ambiguidade do sistema, onde o utilizador atende os pedidos tendo por base origens distintas

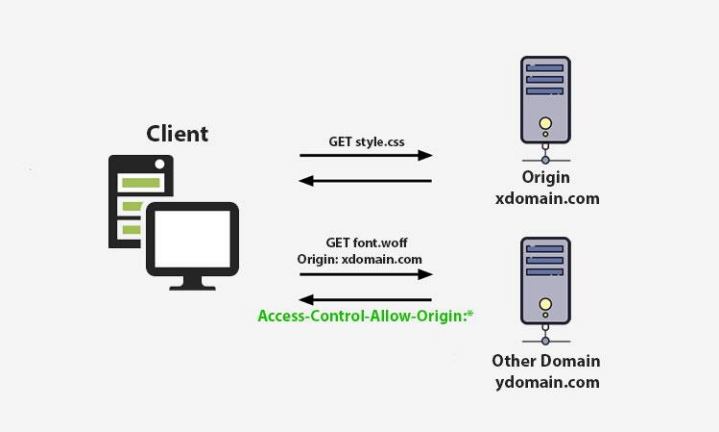


Figura 19 - Ataque CORS

Fonte: https://softaox.info/cross-origin-resource-sharing-cors/

Para impedir que este tipo de ataques ocorra, foi utilizado o *package* CORS, conforme se pode ver no Anexo 10.12, devidamente configurado para só permitir pedidos vindos do domínio https://gestcop.herokuapp.com/, que é o domínio do *frontend* da GestCop.

6.4 Interfaces do sistema

A presente aplicação servirá de protótipo e só terá um género de utilizador que terá acesso a todos os recursos do sistema. Para a conceção das mais variadas interfaces, foi necessário debater com alguns elementos do departamento de sistemas a forma mais adequada de como a informação iria ser exposta. Após ter bem assentes e estruturados os objetivos, foi possível prosseguir com o desenvolvimento das interfaces do sistema.

Com o intuito de se ter uma visão global dos componentes implementados na implementação da aplicação foi efetuado o organigrama apresentado na Figura 20.

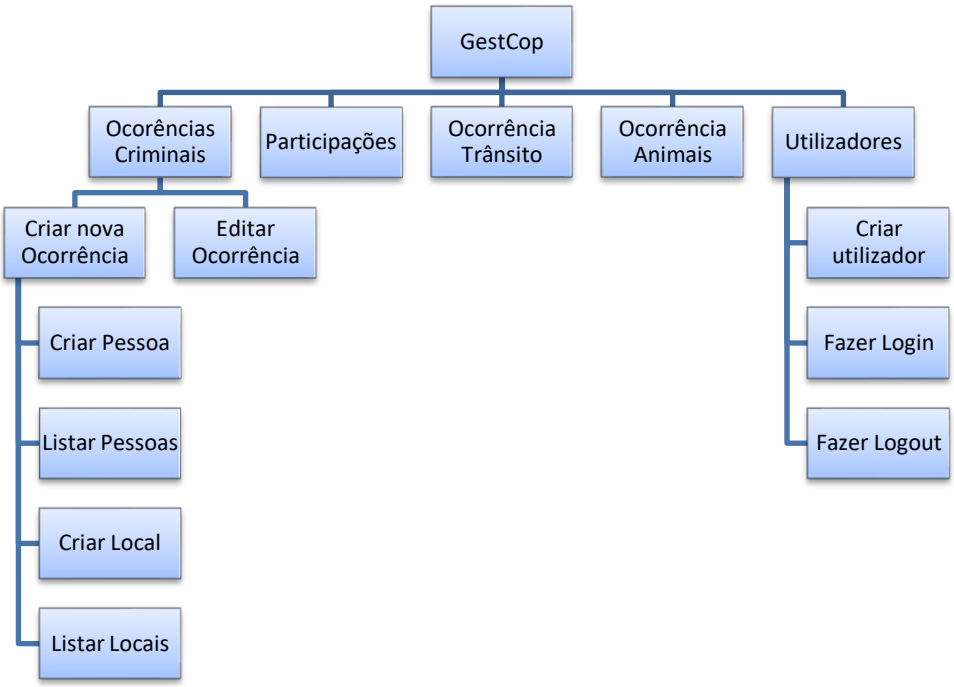


Figura 20 - Organigrama dos componentes da aplicação

6.4.1 Interface de Boas Vindas

Esta interface é a interface principal do sistema, ou seja, é a primeira interface que aparece ao utilizador quando o utilizador acede à aplicação. É através desta *interface* que o utilizador ao clicar num dos botões de Entrar é reencaminhado para a página de *Login.* A Figura 21, demostra esta *interface* gráfica

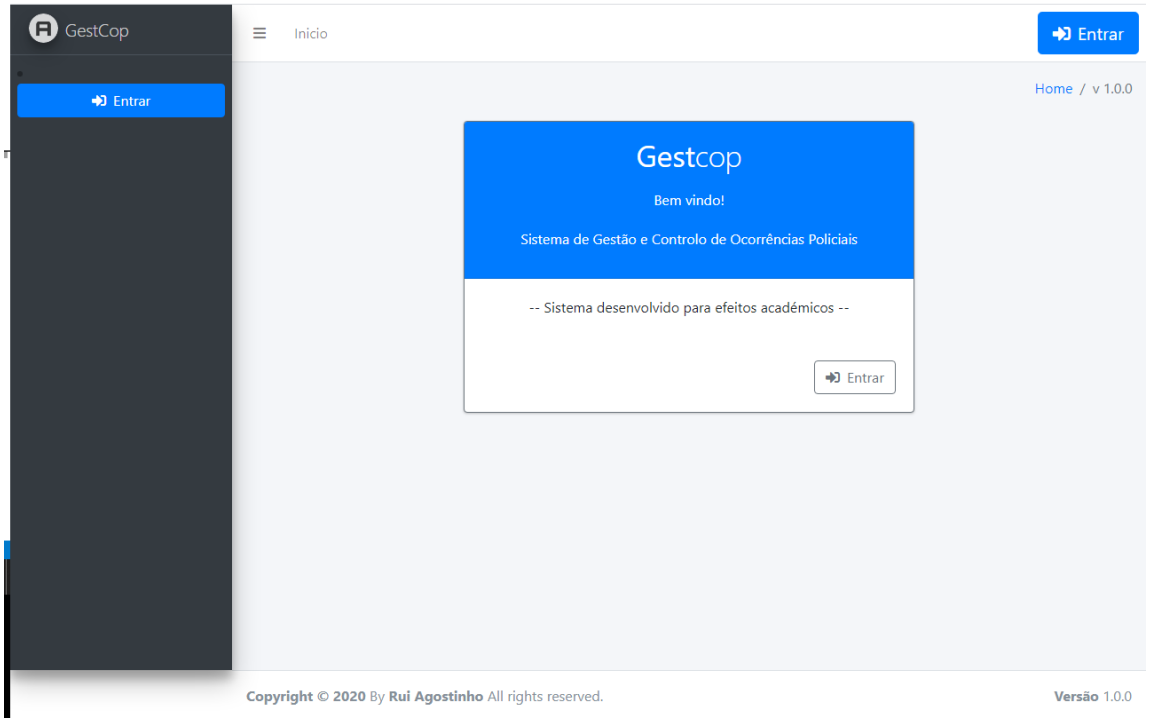


Figura 21 - *Interface* de Boas Vindas

6.4.2 Interface de *Login* da aplicação

Como foi descrito no capítulo designado por "Segurança do Sistema", é necessário que o utilizador esteja perfeitamente autenticado e autorizado para puder aceder aos recursos da aplicação. Para a autenticação dos utilizadores foi criada a interface de *Login* para validar as credenciais e os privilégios que têm no sistema.

Na Figura 22, pode visualizar-se a página de *Login* da aplicação, onde o utilizador tem de inserir o nome e a *password*. Se o nome de utilizador e a *password* forem iguais às definidas aquando da criação do utilizador no sistema, o utilizador é redirecionado para a página *Default*, caso contrário é exibida a mensagem de erro "Utilizador inválido".

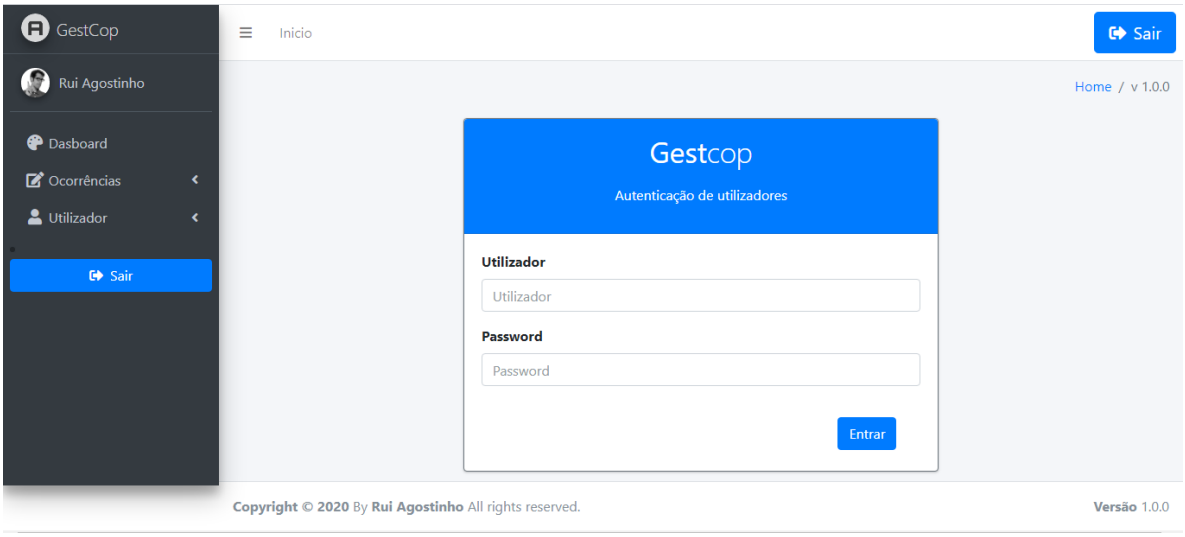


Figura 22 - Interface de Login

6.4.3 Interface Registo de Utilizadores

Como referido acima, a GestCop vai servir de sistema de demonstração, como não está ligada a nenhuma ferramenta de gestão de utilizadores, no entanto a GestCop permite que sejam adicionados novos utilizadores, bastando para isso o elemento policial aceder ao menu Utilizador existente na barra lateral e clicar em adicionar. A Figura 23, demostra esta página de registo de utilizadores.

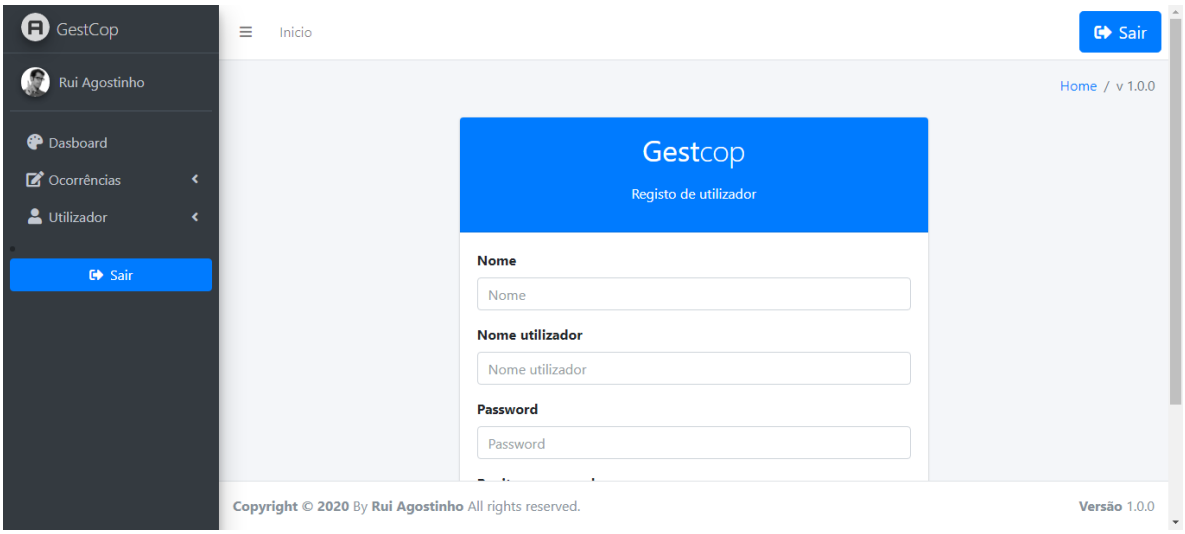


Figura 23- Interface Adicionar Utilizador

6.4.4 Interface *Dashboard*

Após o elemento policial introduzir um nome de utilizador e password válido é encaminhado para a sua área de trabalho *Dashboard*. A Figura 24, demonstra a área de trabalho de um utilizador de teste.

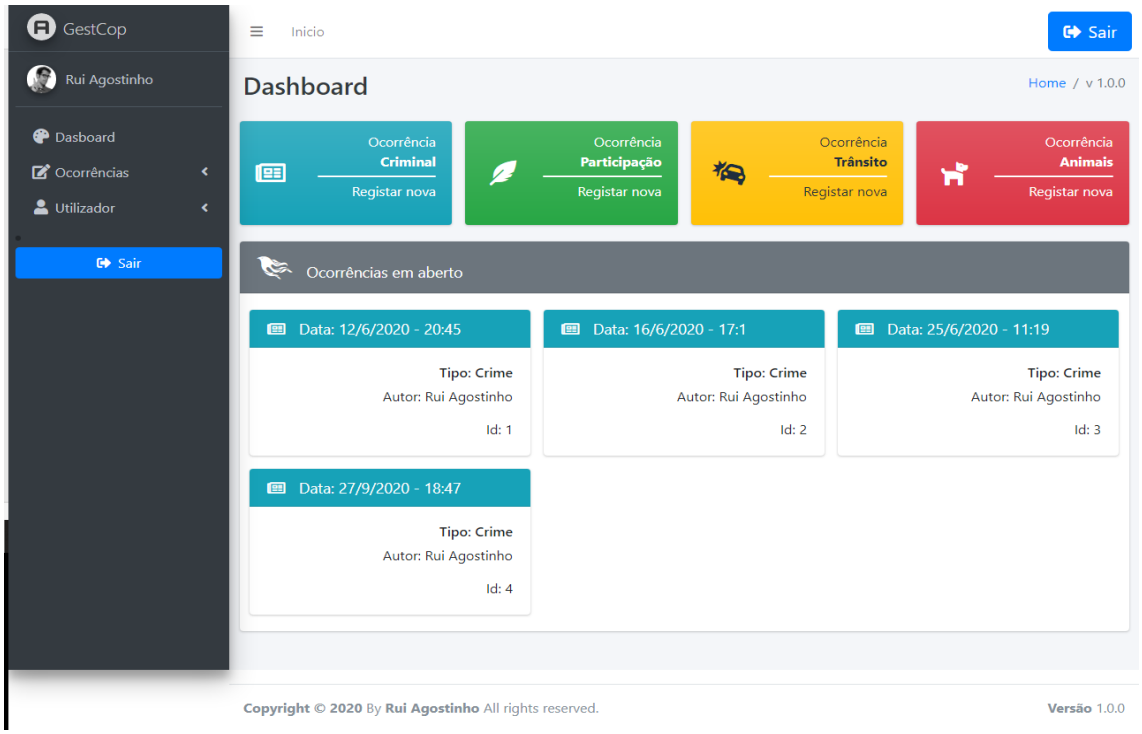


Figura 24 - Página Dashboard

O elemento policial, consegue através desta página dar início ao registo das mais variadas ocorrências/denúncias possíveis, ou visualizar todas as ocorrências/denúncias que tem para terminar.

6.4.5 Interface Associações

Após o elemento policial ter clicado em registar nova ocorrência crime, é redirecionado para a página de Associações, e dá início do registo da ocorrência/denúncia. A Figura 25, demonstra a página de Associações em que é possível visualizar ao cimo o menu *Wizard* que guiará este processo, os botões de inserção de pessoas, locais e objetos. É ainda possível visualizar uma lista com as associações que se vão adicionando.

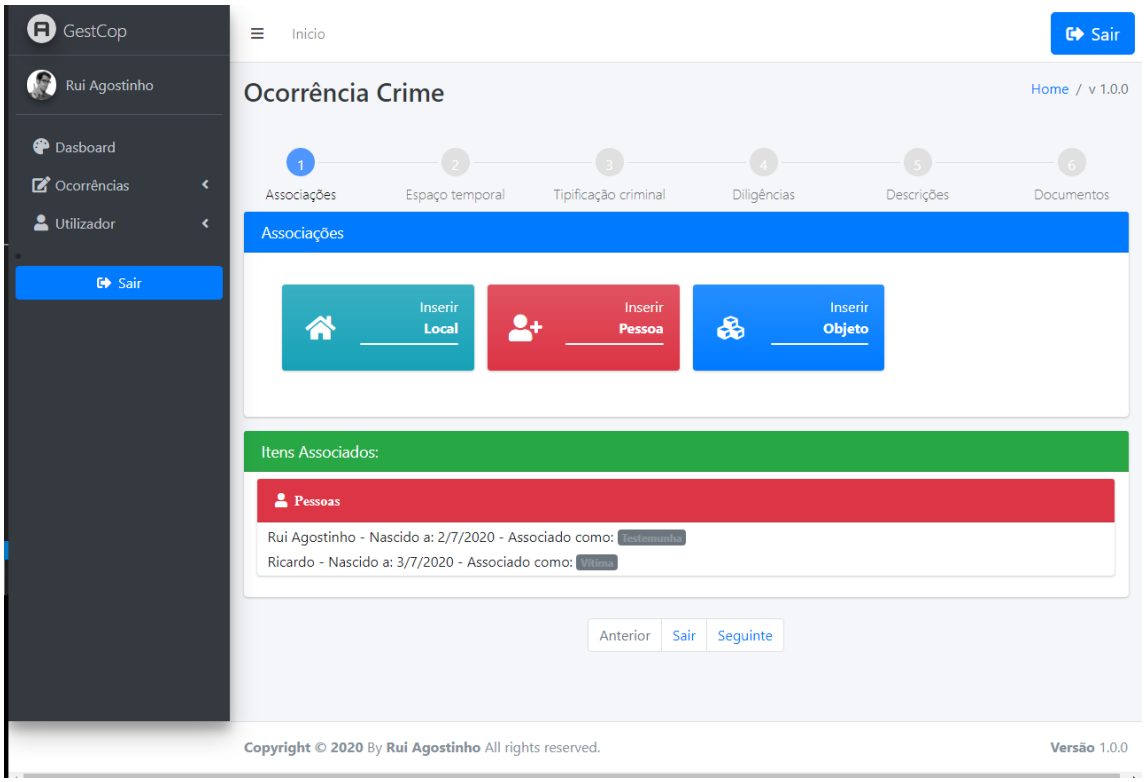


Figura 25 - Página de Associações

Nesta página, o elemento policial poderá efetuar todas as associações necessárias, ou seja, nesta página vai associar tudo aquilo que seja necessário para determinar quem são os autores, lesados ou vítimas do crime, deverá ainda associar objetos e locais de importância para que sirvam como meios de prova do crime (Ex.: Durante o ato de execução do crime esteve envolvida uma arma de fogo que à posteriori foi possível apreender. Esta arma deve ser associada como objeto nesta página).

6.4.6 Interface Associar Pessoa

Para associar uma pessoa, o elemento policial clica no botão inserir pessoa e será reencaminhado para a página inserir pessoa. A Figura 26, demonstra a página com o respetivo formulário de inserção de dados para poder inserir pessoas.

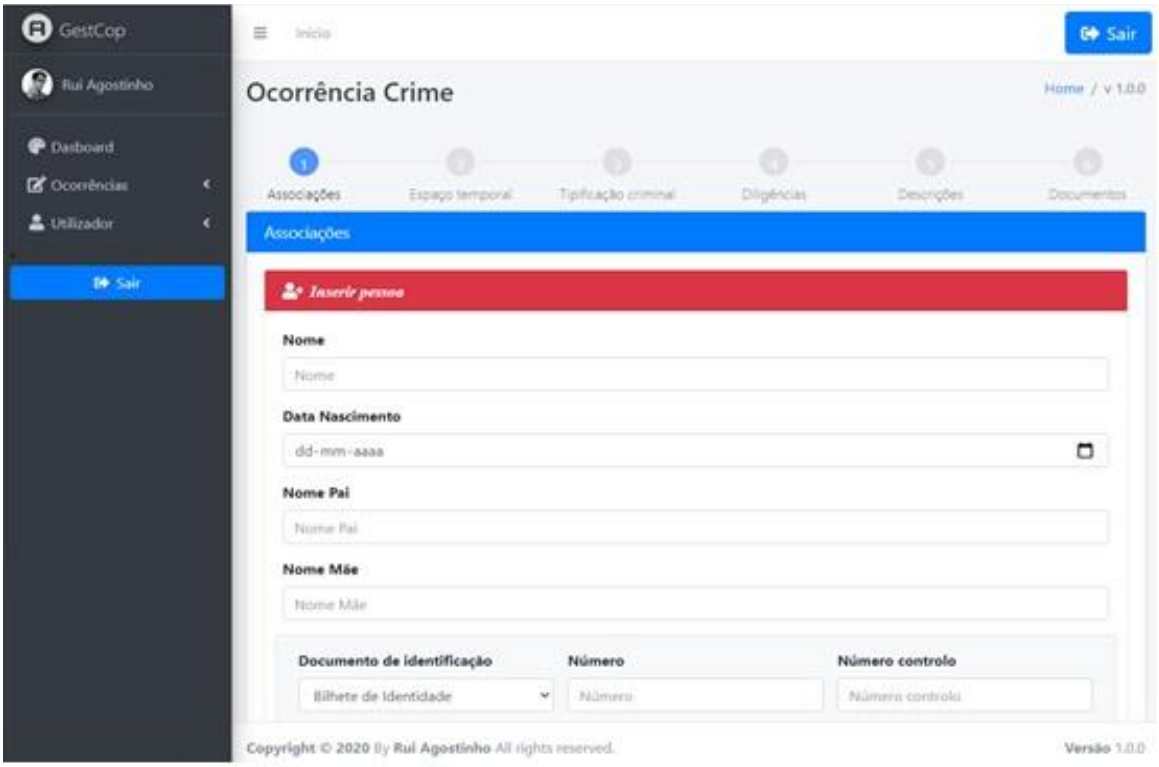


Figura 26 - Interface inserir pessoas

Nesta página o elemento policial insere os respetivos dados referentes à pessoa que está a adicionar, após o términus desta inserção clica em avançar. Durante a inserção destes dados, o sistema, de forma automática vai validando os dados inseridos, se os dados não estiverem devidamente preenchidos o sistema mostra uma mensagem de erro e não avança.

6.4.7 Interface Tipo de Associação de Pessoa

Após o elemento policial ter clicado no botão avançar do respetivo formulário é reencaminhado para a página do tipo de associação que se pretende. A Figura 27, demostra a página onde o elemento policial escolhe o tipo de associação pretendida.

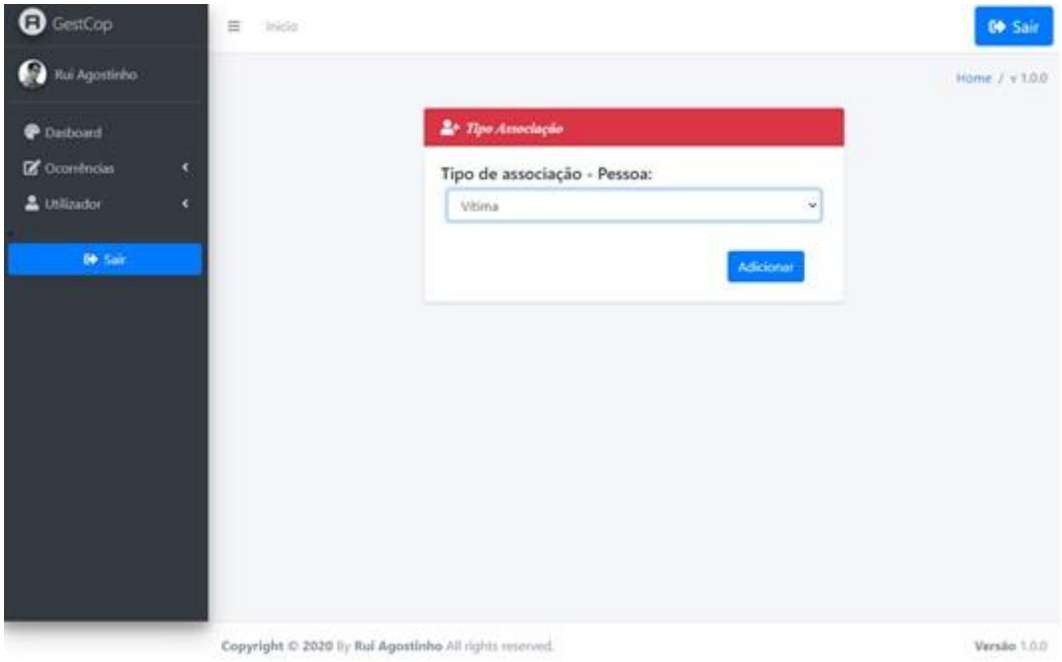


Figura 27 – Interface Tipo de associação de pessoa

Após o elemento policial ter selecionado o tipo de associação que pretende, clica no botão adicionar, o sistema reencaminha o elemento policial novamente para a página de associações.

É relevante salientar que no âmbito das associações é possível ter a mesma pessoa, local ou objeto associados múltiplas vezes, mas com tipificações diferentes, ou seja, podemos ter por exemplo, a pessoa "A" associada como suspeito e vítima ao mesmo tempo. Um exemplo desta situação é por exemplo: temos uma individuo "A" que afirma que foi agredido por "B", mas por sua vez "B" afirma que ele é que foi agredido por "A", nestas situações os indivíduos terão de ser adicionados com ambas as tipificações.

6.4.8 Interface Associar Local

Para associar um local, o elemento policial clica no botão inserir local e será reencaminhado para a página inserir local. Na Figura 28, demonstra a página com o respetivo formulário de inserção de dados para poder inserir pessoas.

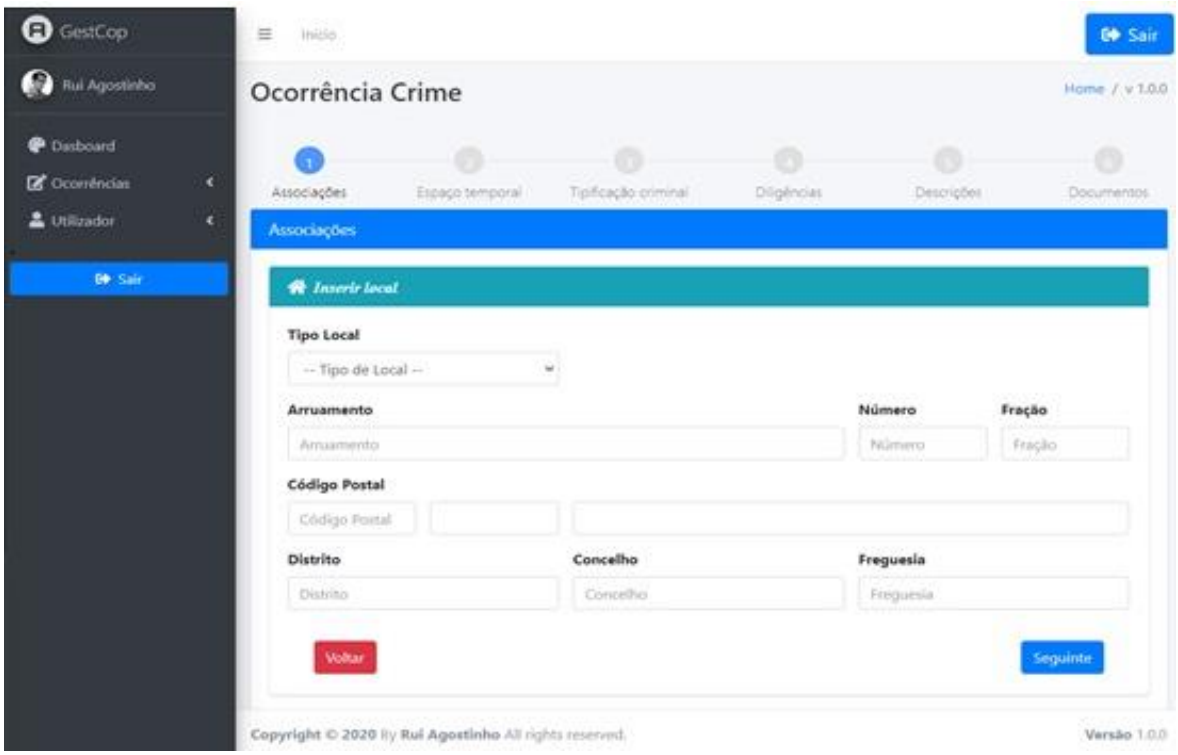


Figura 28– Interface inserir local

Nesta página o elemento policial insere os dados referentes ao local que está a adicionar, após terminar esta inserção clica em avançar. Durante a inserção destes dados, o sistema, de forma automática vai validando os dados inseridos, se os dados não estiverem devidamente preenchidos o sistema mostra uma mensagem de erro e não avança.

6.4.9 Interface Tipo de Associação de Local

Após o elemento policial ter clicado no botão avançar é reencaminhado para a página do tipo de associação que pretende. A Figura 29, demostra a página onde o elemento policial escolhe o tipo de associação pretendida.

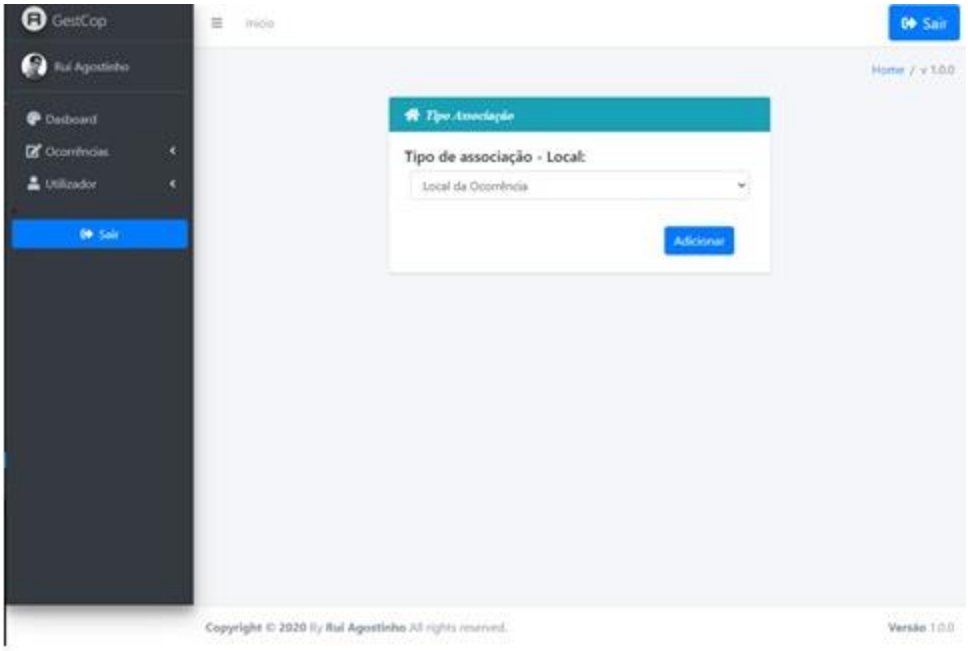


Figura 29 -Interface Tipo de associação de local

Depois de efetuadas todas as associações, o elemento policial clica em seguinte e é reencaminhado para a página de Espaço temporal.

Neste ponto torna-se importante referir que a página, Espaço Temporal, assim como outras não forma implementadas, mas seguidamente faz-se uma breve descrição de como se tenha idealizado o processo de inserção de uma ocorrência.

6.5 Interfaces não implementadas

Na página, Espaço Temporal, o elemento policial introduz, a data e a hora a que ocorreu o ilícito criminal e o modo como teve conhecimento do ilícito criminal (Ex.: através de outro cidadão, via rádio policial, telefone, etc).

Nesta parte é importante salientar que está disponível uma data e hora inicial e uma data e hora final. Este facto deve-se ao facto de muitas das vezes não se conseguir saber o dia e a hora a que ocorreu o ilícito criminal, ficando por esse motivo registado um espaço temporal.

Após a inserção de datas e horas válidas (validação feita pelo sistema) e do modo como teve conhecimento do ilícito criminal, o elemento policial clica em seguinte e o sistema reencaminha-o para a página Tipificação criminal.

Nesta parte é importante salientar que como o módulo que se deseja implementar está relacionado com a registo de uma ocorrência/denúncia de violência doméstica, o elemento policial deverá selecionar no campo "Tipo de crime", o tipo "crimes contras as pessoas" e depois no campo "subtipo" em "ofensas à integridade física". Depois de ter estas opções selecionadas o elemento policial entra num sistema de registo baseado em perguntas e respostas em que de acordo com algumas respostas o sistema vai encaminhado o processo de registo.

O sistema baseado em perguntas e respostas foi pensado com o objetivo de uniformizar e melhorar o processo de recolha de provas (Artigo 124º do CPP) durante o processo de registo da ocorrência/denúncia. Este processo torna-se muito importante pois previne a ocorrência de lapsos e perca de provas importantes para determinar se existiu o ilícito criminal, e se existiu, apurar o(s) seu(s) autor(es).

Como na tipificação do crime na página de "Tipificação criminal" foi selecionado o subtipo de crime "Crimes contra a integridade física" o sistema vai efetuar os seguintes passos:

1. Caso tenha sido inserido uma vítima no sistema, o sistema questiona qual é o relacionamento que a vítima tem com o pressuposto suspeito?

2. O elemento policial responde com umas das opções: cônjuge ou ex-cônjuge, pessoa com quem matem ou tenha mantido relação de namoro ou uma relação análoga à dos cônjuges, progenitor ou descendente comum em primeiro grau, pessoa de quem dependa por motivos financeiros, deficiência, doença ou gravidez (Artigo. 152º número 1 do Código Penal (CP)), ou nenhuma.

3. O sistema questiona se existem lesões visíveis na(s) vítima(s);

4. O elemento policial responde clicando em "Sim" ou "Não";

    4.1. Se o elemento policial clicou em "Sim", o sistema exibe um alerta para a necessidade de efetuar reportagem fotográfica (registo fotográfico de todas as lesões visíveis);

    4.2. O elemento policial clica em "OK". O sistema passa para o ponto 5;

    4.3. Se o elemento policial clicou em não, passa para o ponto 5

5. O sistema questiona se a(s) vítima(s) possui(em) lesões não visíveis;

6. O elemento policial clica em "Sim" ou "Não"

    6.1. Se o elemento policial clicou em "Sim", o sistema exibe um alerta para a necessidade de a vítima ser sujeita a exame médico e para isso deve ser notificada para comparecer no Instituto Nacional de Medicina Legal (INML);

    6.2. O elemento policial clica em "Ok". O sistema passa para o ponto 7;

    6.3. Se o elemento policial clicar em "Não" o sistema passa para o ponto 7;

7. O sistema pergunta se o elemento policial assistiu à agressão (Flagrante delito (Artigo 256º do CPP));

8. O elemento policial clica em “Sim” ou Não;

    8.1. Se o elemento policial clicar em “Sim” e o sistema detetar uma pessoa associada como suspeito, o sistema vai questionar o elemento policial se o suspeito está na sua presença;

    8.2. O elemento policial clica em “Sim” ou “Não”;

        8.2.1. Se o elemento policial clicar em “Sim”, o sistema exibe um alerta para a detenção do suspeito.

        8.2.2. Se o elemento policial clicar em “Não”, o sistema passa para o ponto 9;

    8.3. Se o elemento policial clicar em “Não”, o sistema passa para o ponto 9,

9. O elemento policial clicar em seguinte e é reencaminhado para a página “Diligências”.

Na página “Diligências” o sistema mostra um conjunto de tarefas que o elemento policial tem de efetuar de acordo com o registo da ocorrência/denúncia que se pretende.

Neste caso, como estamos a efetuar o registo de uma ocorrência/denúncia de violência doméstica é necessário fazer as seguintes diligências:

1. Se foram associadas pessoas como vítimas na página de associações é obrigatório que o elemento policial preencha o formulário para o cálculo do grau de Risco de Violência Doméstica (RVD);

2. Se estiverem associadas pessoas menores de idade é obrigatório o elemento policial elaborar um formulário para posterior envio à Comissão de Proteção de Crianças e Jovens (CPCJ) em risco;

3. Se foram associadas pessoas suspeitas e existiu flagrante delito, é obrigatório preencher o formulário do guia individual do detido.

4. Se foram associados objetos como apreendidos, é obrigatório preencher formulário de "Auto de Exame e Avaliação".

Após as tarefas sugeridas pelo sistema nesta página estarem terminadas o elemento policial clica em seguinte e o sistema reencaminha o elemento policial para a página de descrição.

Na página de descrição o elemento policial relata tudo o que conseguiu apurar no local do crime, bem como o mencionado pelas vítimas e testemunhas. No final desta descrição o elemento policial clica em seguinte e é reencaminhado para a página "Documentação".

Na página "Documentação" o elemento procede ao download e impressão de toda a documentação relacionada com o registo da ocorrência/denúncia, nomeadamente: RVD, notificação para a vitima se deslocar ao INML, notificação para reportagem fotográfica, notificação para preservação de imagens de videovigilância, notificação de constituição de arguido, Termo de Identidade e Residência (TIR), guia individual do detido, estatuto de vítima, estatuto de vítima especialmente vulnerável, notificação de comparência em tribuna, Auto de exame e avaliação.

Para terminar o registo da ocorrência/denúncia o elemento clica em terminar.

7 Testes de verificação

Para um bom desenvolvimento e garantia do cumprimento dos requisitos analisados no Capítulo Análise de Requisitos, é necessário realizar os mais variados tipos de testes para encontrar possíveis falhas, com o intuito de serem avaliadas e corrigidas. Foi com este objetivo que durante o seu desenvolvimento foram executados um conjunto de testes, nomeadamente, testes unitários, testes de segurança e testes de validação de entrada de dados.

Neste ponto, importa referir que alguns dos testes efetuados à segurança da GestCop são apenas algumas possibilidades de alguns tipos de ataques.

Seguem-se alguns dos testes efetuados.

7.1 Testes unitários

Estes testes foram sendo feitos à medida que a aplicação ia sendo desenvolvida, serviam no fundo para testar se aplicação ia cumprindo tudo aquilo que se pretendia.

7.2 Teste de segurança SQL *Injection*

Com este teste pretendeu-se aceder à aplicação, para isso foi injetando um pequeno pedaço de código SQL (' OR 1=1; --) no campo onde o utilizador deveria introduzir o seu nome de utilizador, fazendo com que o sistema ignorasse a password. Como se pode verificar na Figura 30, o sistema passou com sucesso este teste.

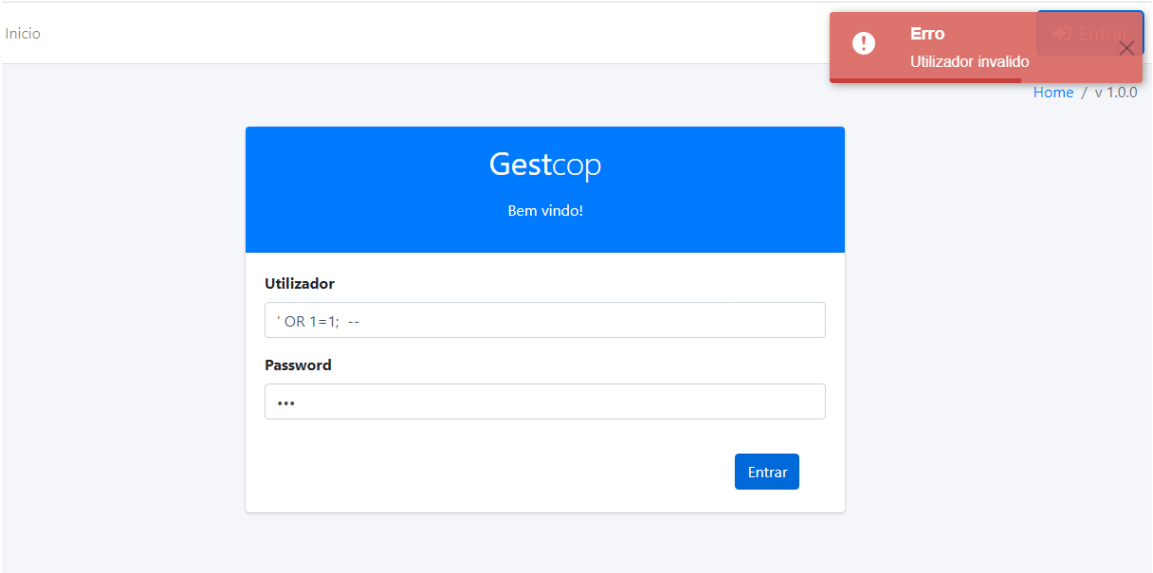


Figura 30 - Teste de SQL Injection

7.3 Teste de segurança - Ataque XSS

Neste teste pretendeu-se injetar o Script num dos campos de *Login* do utilizador para testar se a higienização dos campos da entrada do utilizador estava a funcionar.

```
<script> alert('hacked')</script>
```

O sistema conseguiu passar no teste convertendo o Script em:

%3Cscript%3Ealert%28%27hack%27%29%3C/script%3E

7.4 Teste segurança - CORS

Com teste CORS pretendeu-se verificar se o sistema estava a bloquear ou não pedidos tenham vindo da plataforma herokuapp.com. Para efetuar este teste utilizou-se a ferramenta POSTman para simular o registo de um utilizador, como se pode ver na Figura 31. Com este

teste é possível visualizar que o sistema passou no teste, pois respondeu que o pedido não estava autorizado e por esse motivo não inseriu o utilizador na BD.

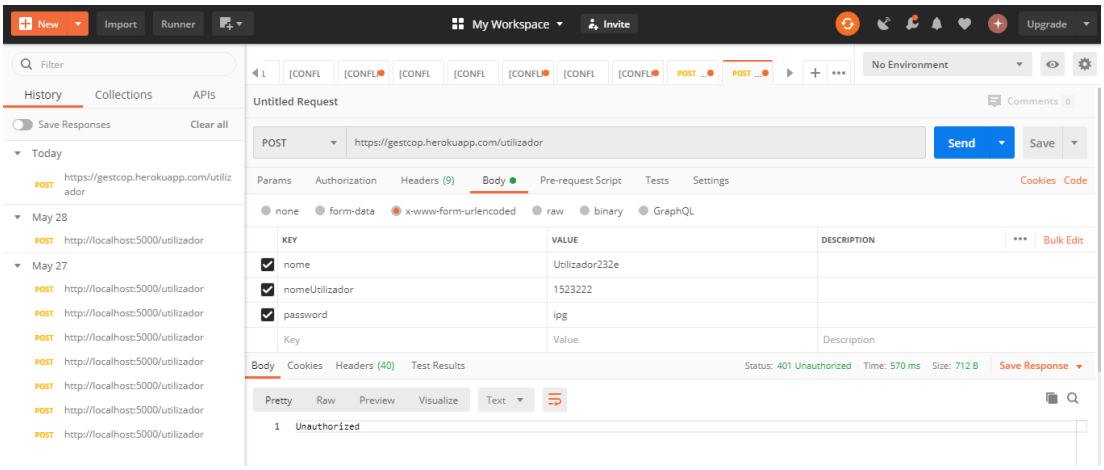


Figura 31 - Teste segurança – CORS

7.5 Teste de segurança - Autenticação

Neste teste pretendeu-se testar se o sistema deixava aceder a funcionalidades sem que existisse um utilizador autenticado. Para efetuar o teste foi necessário que nenhum utilizador estivesse autenticado no sistema. Após confirmar esta premissa tentou-se aceder ao endereço: https://gestcop.herokuapp.com/dashboard. O sistema reencaminhou o pedido para a página de *Login.*

7.6 Testes de validação de entrada de dados

Não menos importante é a validação da informação introduzida pelo utilizador, esta validação tem de garantir que os dados introduzidos pelo utilizador preenchem os requisitos que a

aplicação necessita e que seja garantida a consistência dos dados que são enviados para a BD.

A Figura 32 demonstra um teste de validação de dados aquando da inserção de uma pessoa no sistema, esta validação garante que o elemento policial introduz dados válidos ao inserir uma pessoa, ou seja, o nome, nome do pai e o nome da mãe têm de ter pelo menos 3 carateres, não podem conter caracteres numéricos, a data de nascimento tem de ser inferior à data atual, o número do cartão de cidadão também tem de ser válido, etc.

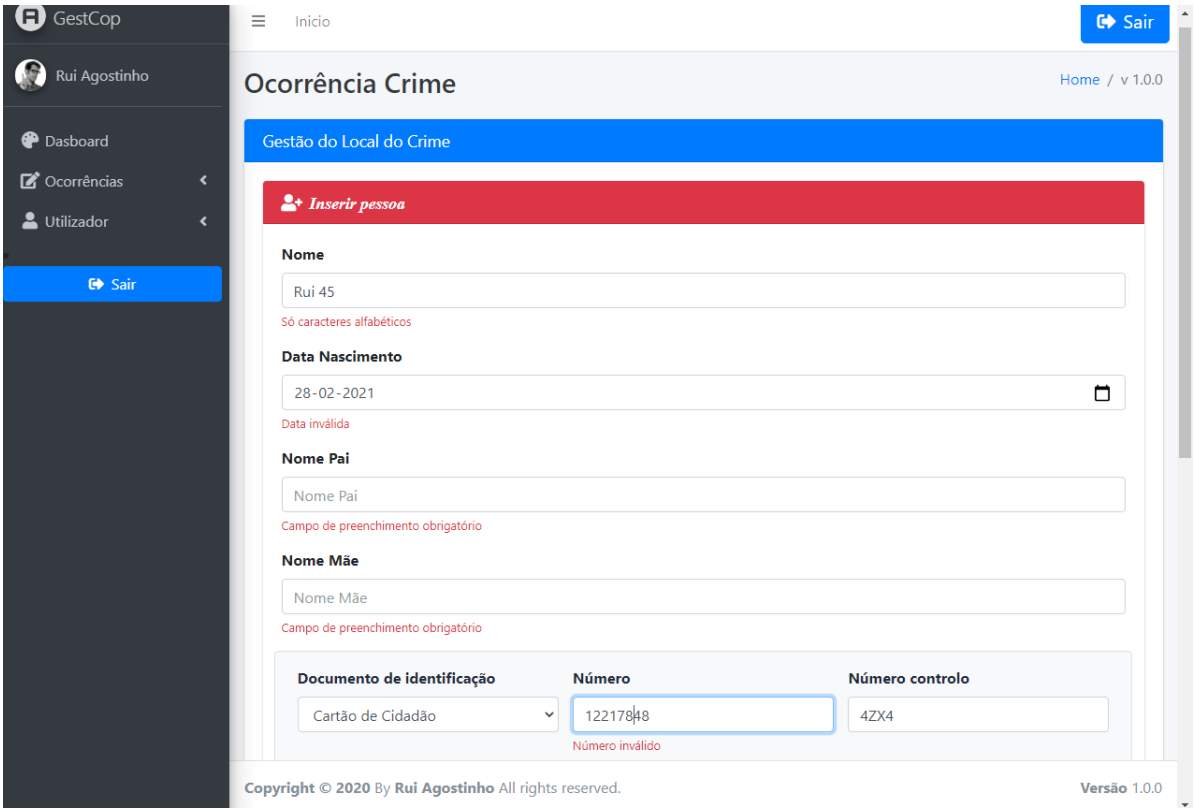


Figura 32 - Teste de validação de entrada de dados

8 Conclusões

Com este trabalho foi desenvolvida uma aplicação *Web*, a GestCop, cujo principal objetivo é demonstrar uma nova abordagem ao registo de ocorrências policiais. Com esta nova abordagem pretendia-se que o registo de ocorrências policiais fosse um processo mais intuitivo, uniformizado, automatizado e que se reduzisse ao máximo a ocorrência de erros durante este processo.

Durante o processo de desenvolvimento deste trabalho, evidencia-se a necessidade de um bom planeamento e desenho da aplicação antes do início da sua implementação. Durante o desenvolvimento desta aplicação foram efetuados grandes ajustes neste desenho e planeamento, levando a que se perdesse muito tempo nestas revisões.

Para o desenvolvimento deste projeto recorreu-se à utilização de tecnologias recentes e à utilização de alguns módulos desenvolvidos por outras entidades. Por estas tecnologias não terem sido abordadas com detalhe durante o curso, foi necessário efetuar um estudo aprofundado das várias funcionalidades que estas disponibilizam, originando um nível de dificuldade de desenvolvimento bastante elevado.

Com o desenvolvimento deste projeto foram atingidos alguns dos objetivos inicialmente propostos, destacando-se entre eles, a implementação da arquitetura do sistema, a implementação do menu *Wizard* e uma pequena parte dos passos necessários para um registo de ocorrências mais intuitivo.

8.1 Trabalho futuro

É de salientar que o tempo de estágio é um tempo muito curto para aprender novas tecnologias e desenvolver uma aplicação completa, ficando ainda por cumprir muitos dos objetivos inicialmente previstos.

De entre estes objetivos por cumprir destacam-se: a emissão de autos de notícia e das demais notificações afetas à atividade policial, o envio dos emails para as diversas entidades e o sistema de leitura de cartões de identificação recorrendo à utilização de uma *webcam* ou da câmara do telemóvel.

9 Bibliografia

Antonio, C. d. (2015). *Pro React.* New York: Apress.

AritaSen. (13 de 03 de 2020). *Node.js | Loop de eventos*. Obtido de GeeksforGeeks: https://www.geeksforgeeks.org/node-js-event-loop/

Arnold, J. (09 de 02 de 2017). *Fetch vs. Axios.js for making http requests*. Obtido de medium: https://medium.com/@thejasonfile/fetch-vs-axios-js-for-making-http-requests-2b261cdd3af5

Carvalho & Mello. (2012). *Aplicação do método ágil scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica. - Gestão Produtos.*

CodeCademy. (15 de 10 de 2020). *React: The Virtual DOM*. Obtido de CodeCademy: https://www.codecademy.com/articles/react-virtual-dom

Diário da República Eletrónico. (08 de 04 de 2020). *Diário da República Eletrónico*. Obtido de DRE - Lei 53/2007: https://dre.pt/home/-/dre/641142/details/maximized

Diário da República Eletrónico. (09 de 04 de 2020). *Diário da República Eletrónico*. Obtido de DRE - Despacho 6158/2017: https://dre.pt/home/-/dre/107677747/details/maximized

DuBois, P. (2013). *MySQL* . United States of America: Developer's Library.

Filipe Portela, Ricardo Queirós. (2018). *Introdução ao Desenvolvimento Moderno Para a Web.* Lisboa: FCA.

Hotframeworks. (15 de 10 de 2020). *Find your new favorite web framework*. Obtido de Hotframeworks: https://hotframeworks.com/#rankings

Jacobson, S. &. (2011). *USE-CASE 2.0 The Guide to Succeeding with Use Cases.* Londres: Ivar Jacobson International.

Jesus, L. (06 de 12 de 2016). SCRUM – Um breve resumo. pp. https://www.linkedin.com/pulse/scrum-um-breve-resumo-leandro-jesus. Obtido de Revista Pensar: http://revistapensar.com.br/tecnologia/pasta_upload/artigos/a64.pdf

JUSTIÇA.GOV.PT. (18 de 10 de 2017). *Investigação Criminal*. Obtido em 07 de 10 de 2020, de JUSTIÇA.GOV.PT: https://justica.gov.pt/Justica-Criminal/Investigacao-Criminal

Khuat, T. (2018). *Developing a frontend application.* Leppävaara: Laurea University of Applied Sciences.

Lewis, J. (2015). Introduction to the special issue on usability and user experience: Methodological evolution. *The International Journal of Human Resource Management*, 555-556.

Lópes-Campos, Salvatore & Manfredi. (2014).

Luís Abreu. (19 de 02 de 2016). *Node.JS Construção de Aplicações Web.* Lisboa: FCA. Obtido em 01 de 05 de 2020, de https://pt.wikipedia.org/wiki/React_(JavaScript): https://pt.wikipedia.org/wiki/React_(JavaScript)

Marques, E. (2005). *Faculdade de Ciencias do porto*. Obtido de Universidade Porto: https://www.dcc.fc.up.pt/~edrdo/aulas/bd/teoricas/bd_modelo_er.pdf

Maruta, R. (14 de 02 de 2018). *Iniciando com Redux em 9 passos*. Obtido em 05 de 06 de 2020, de Medium: https://medium.com/reactbrasil/iniciando-com-redux-c14ca7b7dcf

Maruta, R. (14 de 02 de 2018). *Iniciando com Redux em 9 passos*. Obtido de Medium: https://medium.com/reactbrasil/iniciando-com-redux-c14ca7b7dcf

Microsoft. (15 de 10 de 2020). *O que é PaaS?* Obtido de Microsoft: https://azure.microsoft.com/pt-pt/overview/what-is-paas/

Ministério da Administração Interna. (05 de 03 de 2020). *Queixa Eletónica*. Obtido de Queixa Eletrónica: https://queixaselectronicas.mai.gov.pt/SQE2013/default.aspx#tag=MAIN_CONTENT

Ministério da Administração Interna. (14 de 04 de 2020). *Queixa Eletrónica*. Obtido de Queixa Eletrónica: https://queixaselectronicas.mai.gov.pt/SQE2013/default.aspx#tag=MAIN_CONTENT

O'Neil, M. N. (2004). *Fundamentos de UML.* Lisboa: FCA - Editora de Informática.

Oracle. (2020). *MySQL::MySQL Workbench*. Obtido em 23 de 06 de 2020, de MySQL: https://www.mysql.com/products/workbench/

Passportjs. (15 de 11 de 2020). *passport-jwt*. Obtido de Passportjs: http://www.passportjs.org/packages/passport-jwt/

Pearson. (14 de 06 de 2011). *Software engineering / Ian Sommerville. — 9th ed.* Boston: Addison-Wesley. Obtido em 24 de 06 de 2019, de https://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_destaques&DESTAQUESdest_boui=354227526&DESTAQUESmodo=2

Peixoto, M., Silva, M. & Rocha, C. (2010). *Aprendizagem e metacognição no ensino de metodologia científica. Ensaio Pesquisa em Educação em Ciências.*

Pissarra, M. (12 de 01 de 2018). *O processo Penal*. Obtido em 06 de 10 de 2018, de JUP: https://www.juponline.pt/politica/artigo/24787/o-processo-penal.aspx

Polícia de Segurança Pública. (08 de Abril de 2020). *PSP - História*. Obtido de PSP: https://www.psp.pt/Pages/sobre-nos/quem-somos/historia.aspx

Polícia de Segurança Publica. (08 de Abril de 2020). *PSP - Missão*. Obtido de PSP - Missão: https://www.psp.pt/Pages/sobre-nos/quem-somos/o-que-e-a-psp.aspx

Policia de Segurança Publica. (02 de 03 de 2020). *SEI*. Obtido de Sistema Estratégico de informação, Gestão e Controlo Operacional.

Prates, R. (2015). *BootStrap 3.3.5.* São Paulo - Brasil: Novatec Editora, Ltda.

RedMonk. (27 de 07 de 2020). *The RedMonk Programming Language Rankings: junho de 2020*. Obtido de RedMonk: https://redmonk.com/sogrady/2020/07/27/language-rankings-6-20/?utm_source=rss&utm_medium=rss&utm_campaign=language-rankings-6-20

Ricardo Queirós & Filipe Portela. (2018). *Introdução ao Desenvolvimento Moderno para a web.* Lisboa: FCA.

Ricardo Queirós & Filipe Portela. (2018). *Introdução ao Desenvolvimento Moderno para a web.* Lisboa: FCA.

Ricardo Queirós & Filipe Portela. (2018). *Introdução ao Desenvolvimento Moderno para a web.* Lisboa: FCA.

Schwaber & Sutherland. (2013). *The Scrum Guide - The definitive guide to Scrum: The rules of the game. Share-Alike do Creative Commons.*

Secretaria da Educação do Estado de São Paulo. (06 de 03 de 2020). *Registos de Ocorrências Escolares*. Obtido de Sistema de Proteção Escolar: http://www.educacao.sp.gov.br/spec/registros-de-ocorrencias-escolares-roe/

Secretaria da Educação do Estado de São Paulo. (14 de 04 de 2020). *Secretaria do Estado da Educação*. Obtido de Secretaria do Estado da Educação: http://portalnet.educacao.sp.gov.br/

Sistema Nacional de Saúde. (2018). *Retrato da saúde 2018*. (Sistema Nacional de Saúde) Obtido em 24 de 06 de 2019, de www.sns.gov.pt: https://www.sns.gov.pt/retrato-da-saude-2018/

Sobrevilla, G. H.-E. (2017). Aplicando Scrum y prácticas de ingeniería de software para la mejora continua del desarrollo de un sistema ciber-físico. *Revista Electrónica de Computación, Informática, Biomédica y Electrónica*, 1-15.

Vipul A M & Prathamesh Sonpatki. (2016). *ReactJS by Example - Building Modern Web*. Birmingham: Packt Publishing.

W3teches. (15 de 10 de 2020). *Usage statistics of server-side programming languages for websites*. Obtido de W3teches: https://w3techs.com/technologies/overview/programming_language

w3techs. (15 de 11 de 2020). *Usage statistics and market share of Bootstrap for website*. Obtido de w3techs: https://w3techs.com/technologies/details/js-bootstrap

w3techs. (15 de 11 de 2020). *Usage statistics of Embedded CSS for websites*. Obtido de w3techs: https://w3techs.com/technologies/details/ce-embeddedcss

w3techs. (13 de 11 de 2020). *Usage statistics of markup languages for websites*. Obtido de w3techs: https://w3techs.com/technologies/overview/markup_language

w3techs. (15 de 10 de 2020). *Usage statistics of Node.js*. Obtido de w3techs: https://w3techs.com/technologies/details/ws-nodejs

w3techs. (15 de 10 de 2020). *Usage statistics of web servers*. Obtido de Usage statistics of web servers: https://w3techs.com/technologies/overview/web_server

Wikipédia. (22 de 11 de 2019). *Metodologia (Engenharia de Software)*. (Instituto Nacional de Estatística) Obtido em 24 de 06 de 2019, de Wikipédia: https://pt.wikipedia.org/wiki/Metodologia_(engenharia_de_software)

Wroblewski, L. (2011). *Mobile First.* New York: Mandy Brown.

Zapata, C. &. (2008). Generación del diagrama de secuencias de uml 2.1.1 desde esquemas preconceptuales. *EIA*(10), 89-103.

10 Anexos

10.1 Atribuições PSP - Artigo 3.º da Lei nº 53/2007

1 - Em situações de normalidade institucional, as atribuições da PSP são as decorrentes da legislação de segurança interna e, em situações de excepção, as resultantes da legislação sobre a defesa nacional e sobre o estado de sítio e de emergência.

2 - Constituem atribuições da PSP:

a) Garantir as condições de segurança que permitam o exercício dos direitos e liberdades e o respeito pelas garantias dos cidadãos, bem como o pleno funcionamento das instituições democráticas, no respeito pela legalidade e pelos princípios do Estado de direito;

b) Garantir a ordem e a tranquilidade públicas e a segurança e a protecção das pessoas e dos bens;

c) Prevenir a criminalidade em geral, em coordenação com as demais forças e serviços de segurança;

d) Prevenir a prática dos demais actos contrários à lei e aos regulamentos;

e) Desenvolver as acções de investigação criminal e contra-ordenacional que lhe sejam atribuídas por lei, delegadas pelas autoridades judiciárias ou solicitadas pelas autoridades administrativas;

f) Velar pelo cumprimento das leis e regulamentos relativos à viação terrestre e aos transportes rodoviários e promover e garantir a segurança rodoviária, designadamente através da fiscalização, do ordenamento e da disciplina do trânsito;

g) Garantir a execução dos actos administrativos emanados da autoridade competente que visem impedir o incumprimento da lei ou a sua violação continuada;

h) Participar no controlo da entrada e saída de pessoas e bens no território nacional;

i) Proteger, socorrer e auxiliar os cidadãos e defender e preservar os bens que se encontrem em situações de perigo, por causas provenientes da acção humana ou da natureza;

j) Manter a vigilância e a protecção de pontos sensíveis, nomeadamente infra-estruturas rodoviárias, ferroviárias, aeroportuárias e portuárias, edifícios públicos e outras instalações críticas;

l) Garantir a segurança nos espectáculos, incluindo os desportivos, e noutras actividades de recreação e lazer, nos termos da lei;

m) Prevenir e detectar situações de tráfico e consumo de estupefacientes ou outras substâncias proibidas, através da vigilância e do patrulhamento das zonas referenciadas como locais de tráfico ou consumo;

n) Assegurar o cumprimento das disposições legais e regulamentares referentes à protecção do ambiente, bem como prevenir e investigar os respectivos ilícitos;

o) Participar, nos termos da lei e dos compromissos decorrentes de acordos, tratados e convenções internacionais, na execução da política externa, designadamente em operações internacionais de gestão civil de crises, de paz, e humanitárias, no âmbito policial, bem como em missões de cooperação policial internacional e no âmbito da União Europeia e na representação do País em organismos e instituições internacionais;

p) Contribuir para a formação e informação em matéria de segurança dos cidadãos;

q) Prosseguir as demais atribuições que lhe forem cometidas por lei.

3 - Constituem ainda atribuições da PSP:

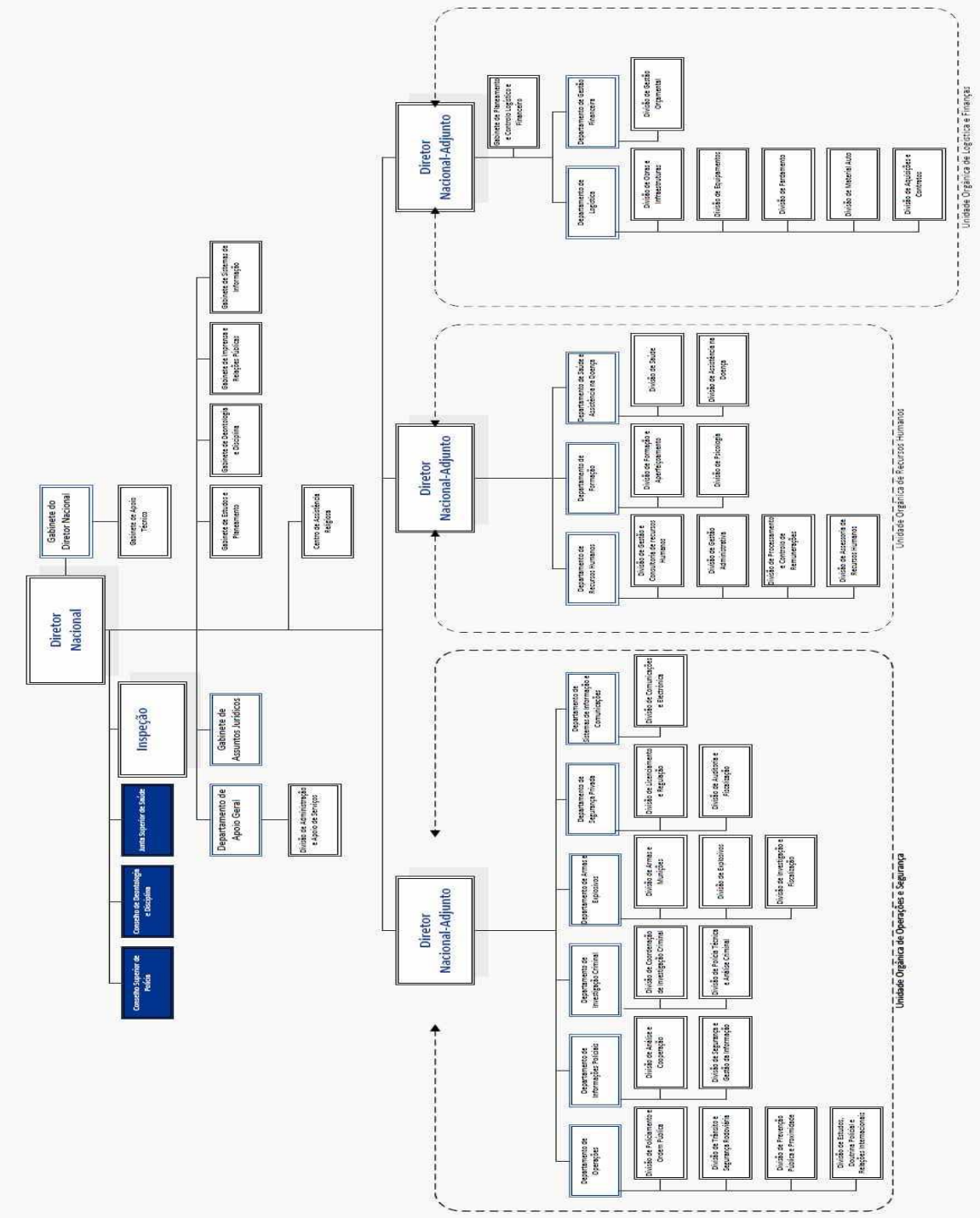
a) Licenciar, controlar e fiscalizar o fabrico, armazenamento, comercialização, uso e transporte de armas, munições e substâncias explosivas e equiparadas que não pertençam ou se destinem às Forças Armadas e demais forças e serviços de segurança, sem prejuízo das competências de fiscalização legalmente cometidas a outras entidades;

b) Licenciar, controlar e fiscalizar as actividades de segurança privada e respectiva formação, em cooperação com as demais forças e serviços de segurança e com a Inspecção-Geral da Administração Interna;

c) Garantir a segurança pessoal dos membros dos órgãos de soberania e de altas entidades nacionais ou estrangeiras, bem como de outros cidadãos, quando sujeitos a situação de ameaça relevante;

d) Assegurar o ponto de contacto permanente para intercâmbio internacional de informações relativas aos fenómenos de violência associada ao desporto.

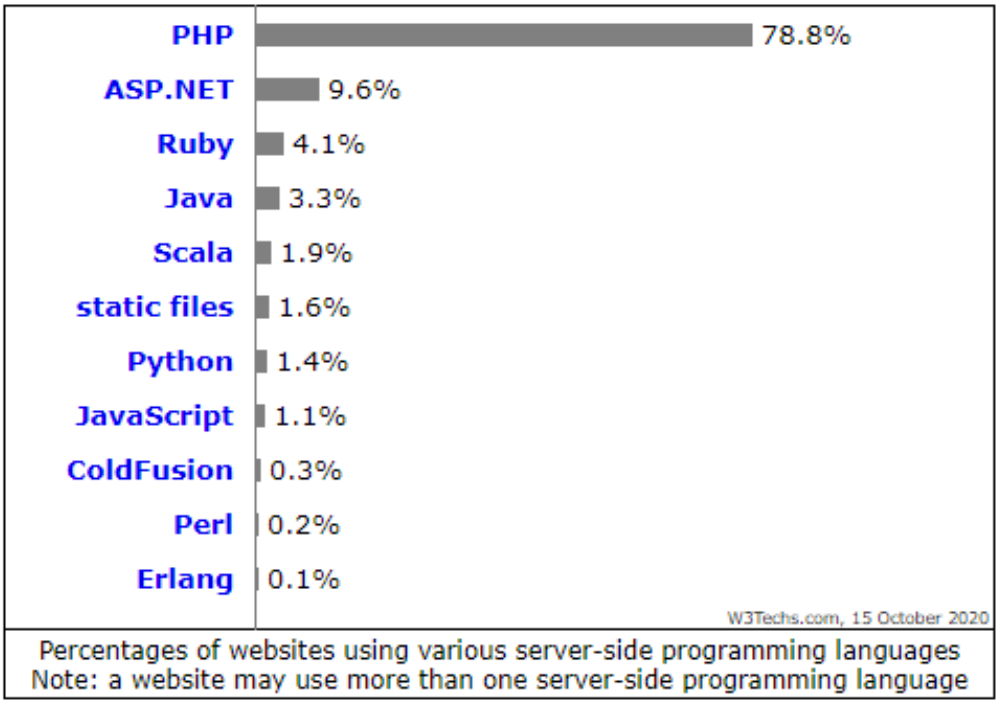
10.2 Organograma da PSP



Fonte: https://www.psp.pt/Pages/sobre-nos/quem-somos/o-que-e-a-psp.aspx

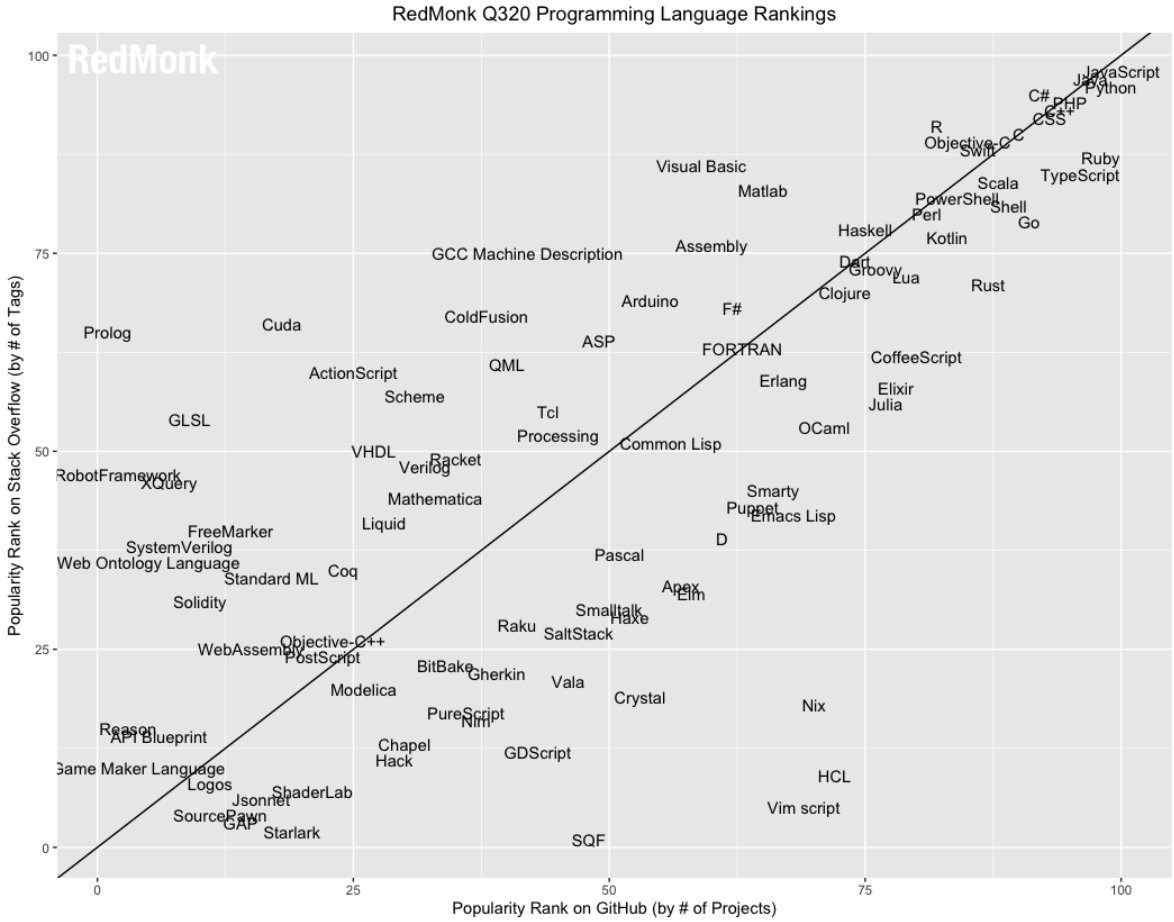
10.3 Gráficos estatísticos – Linguagens de programação

10.3.1 Ranking de linguagens de programação – Lado Servidor



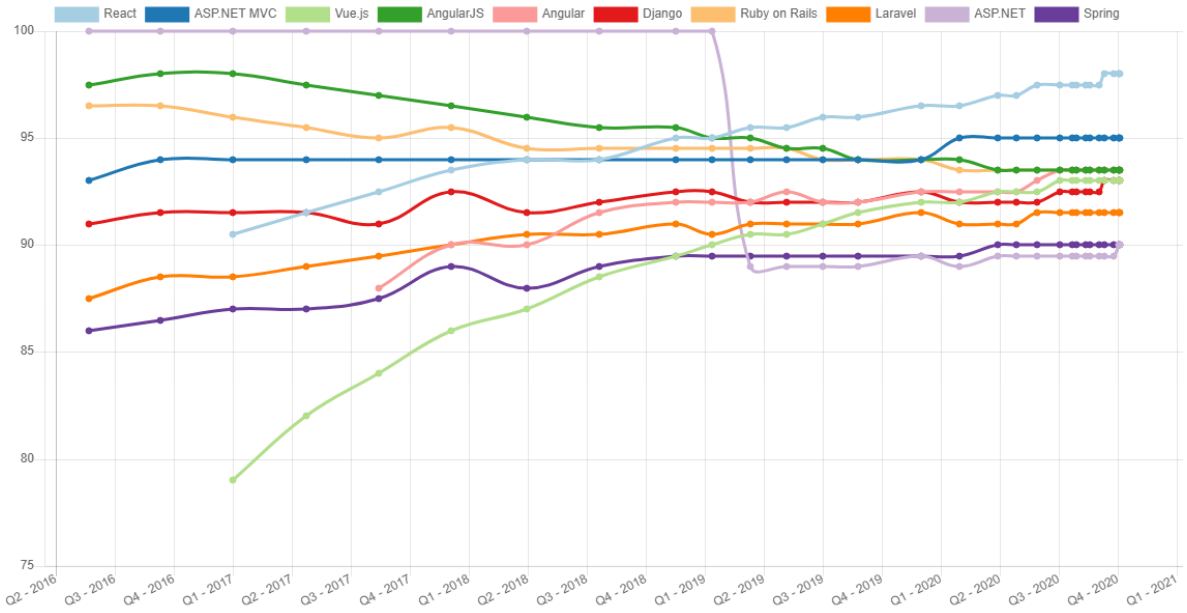
Fonte: https://redmonk.com/sogrady/files/2020/07/lang-rank-q320-wm.png

10.3.2 Ranking de linguagens de programação – Lado Servidor 2



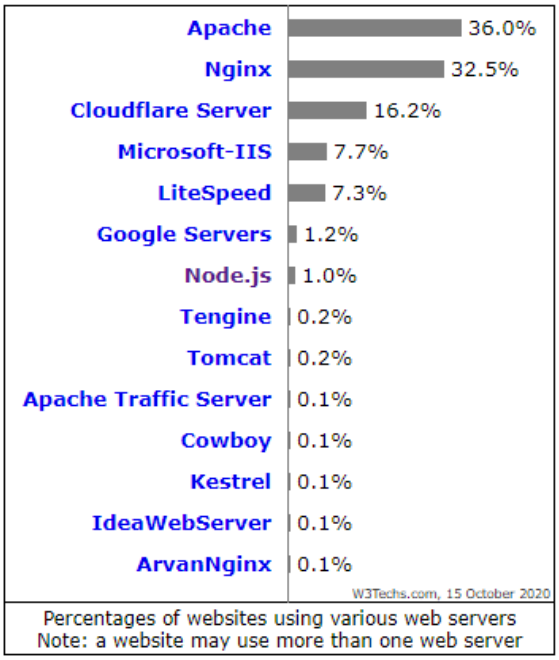
Fonte: https://redmonk.com/sogrady/files/2020/07/lang-rank-q320-wm.png

10.3.3 Ranking de utilização Frameworks



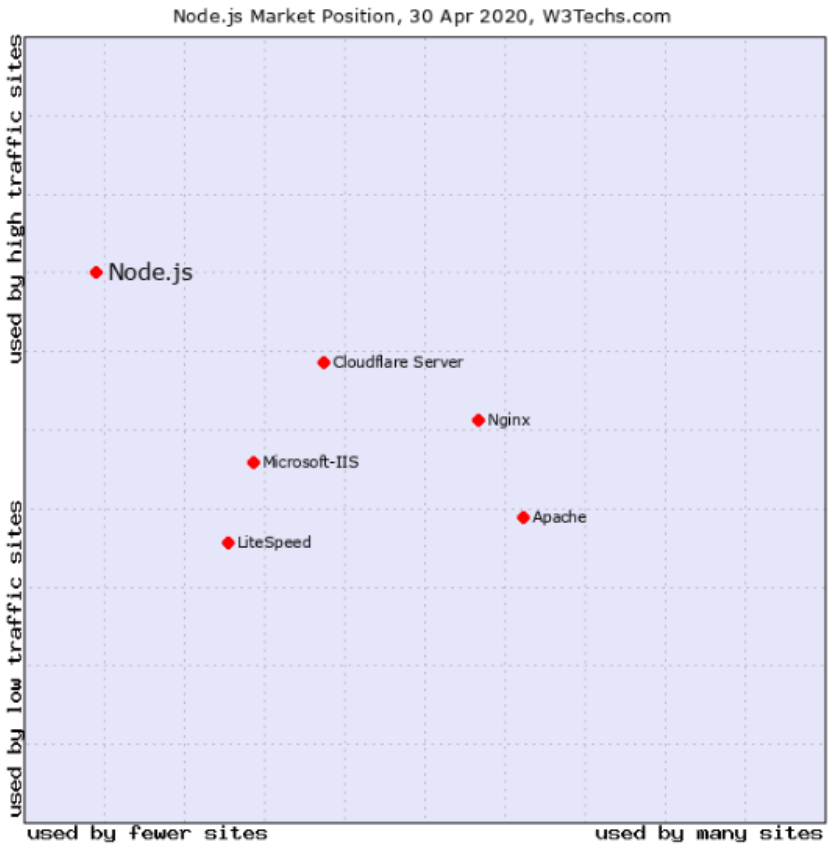
Fonte: https://hotframeworks.com/#rankings

10.3.4 Ranking de utilização Web Servers



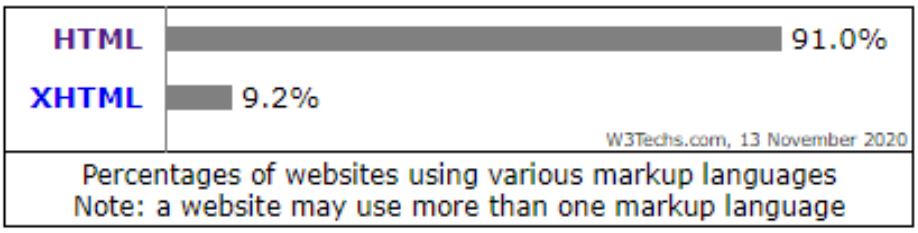
Fonte: https://w3techs.com/technologies/overview/web_server

10.3.5 Ranking de utilização Web Servers – Tratamento de grande volume de dados



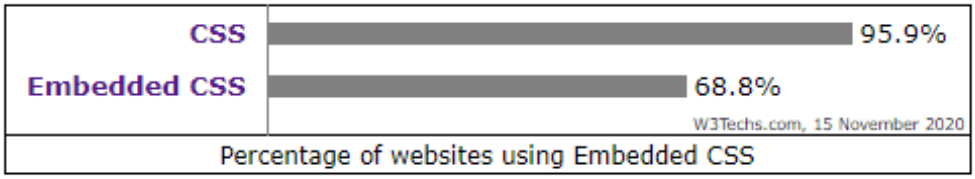
Fonte: https://w3techs.com/technologies/details/ws-nodejs

10.3.6 Ranking HTML



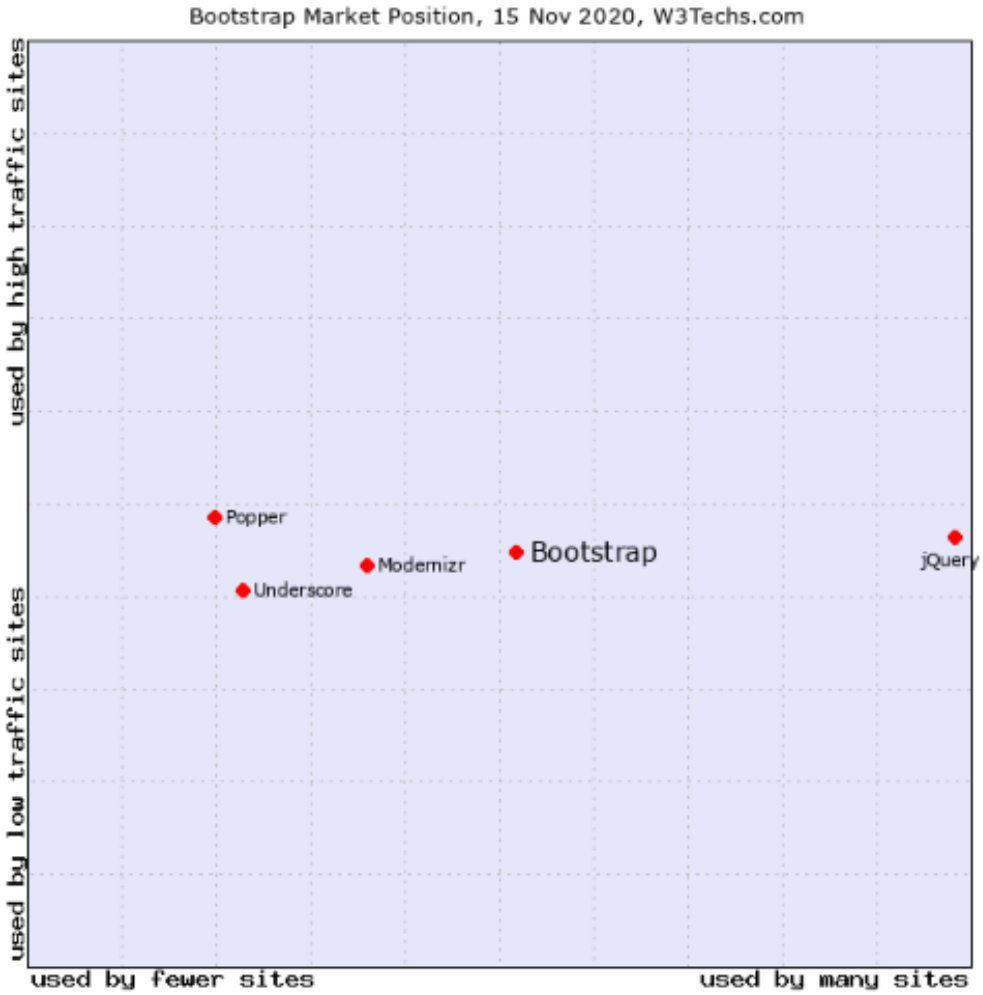
Fonte: https://w3techs.com/technologies/overview/markup_language

10.3.7 Ranking CSS



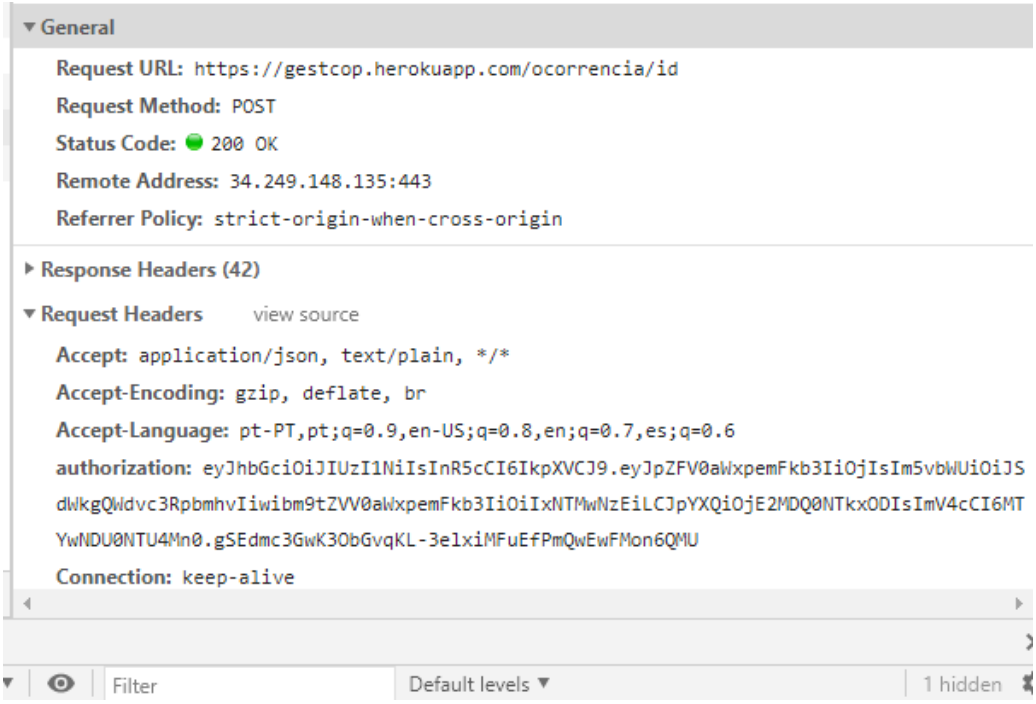
Fonte: https://w3techs.com/technologies/details/ce-embeddedcss

10.3.8 Ranking Bootstrap



Fonte: https://w3techs.com/technologies/details/js-bootstrap

10.4 Cabeçalho HTTP



10.5 Protocolo REST

10.5.1 Requisição *frontend* para *backend*

```
export function getTiposDocumentos(){
   return dispatch => {axios.post('/tipoDocumentos')
    .then(resp => {
            //console.log('tipos de documento '+resp.data)
            dispatch({ type: 'SET_DOCUMENTOS', payload: resp.data})
    }).catch(e => {
        console.log(e)
        e.response.data.erros.forEach(
            erros => toastr.error('Erro', erros))
    })
    }
}
```

10.5.2 Resposta *backend* para *frontend*

```
router.post('/', auth.authenticate('jwt',{ session: false }), (req, res, don
e) => {
    TiposDocumentos.getAll(function (erro, result) {
        if (erro) {
            return res.status(400).send({erros: ["Erro de base de dados"] })
        } else {
            console.log(result)
            return res.status(200).send(result)
        }
    })
})
```

10.6 Configuração de *MidlleWares*

```
import React from 'react';
import ReactDOM from 'react-dom';

import {createStore, combineStore, combineReducers, applyMiddleware} from 'r
edux'
import {Provider} from 'react-redux'
//para acionar vaias actions
import multi from 'redux-multi'
//promise na action
import promise from 'redux-promise'
//
import thunk from 'redux-thunk'


import App from './App';
import reducers from './reducers/reducers'

const devTools = window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTO
OLS_EXTENSION__()
const store = applyMiddleware(thunk, multi, promise)(createStore)(reducers,
devTools)

ReactDOM.render(
    <Provider store={store}>
            <App />
    </Provider>

, document.getElementById('root'));
```

10.7 Implementação React-Redux - Entrar

10.7.1 Store

```
import {combineReducers} from 'redux'

import EntrarReducer from '../views/entrar/entrarReducer/entrarReducer'
import UtilizadorReducer from '../views/utilizador/utilizadorReducer'
import ocorrenciaReducer from '../views/ocorrencias/ocorrenciaReducer'
import OcorrenciasAbertas from '../views/dashboard/dashboardReducer'

import {reducer as toastrReducer} from 'react-redux-toastr'



import {reducer as formReducer} from 'redux-form'

const rootReducer = combineReducers({
    //utilizador : ()=>({ utilizador :{nome:"Rui Agostinho", nomeUtilizador:
 "153071", password:"", idTipoUtilizador:"2", token:false}}),
    form: formReducer,
    toastr: toastrReducer,
    utilizador : EntrarReducer,
    validToken : EntrarReducer,
    user: UtilizadorReducer,
    ocorrencia: ocorrenciaReducer,
    ocorrenciasAbertas: OcorrenciasAbertas,
    //local: LocalReducer

})

export default rootReducer
```

10.7.2 *Actions* de Entrar

```
import { toastr } from 'react-redux-toastr'
import axios from 'axios'
import {history} from '../../../history'



export function login(values) {
    return submit(values, '/login')
}
export function signup(values) {
    return submit(values, '/signup')
}

function submit(valores, url) {
    return dispatch => {
        axios.post(url, valores)
            .then(resp => {
                toastr.success('Sucesso', `Bem vindo ${resp.data.nome}`)
                //console.log(url)
                //console.log(resp.data)
                dispatch(
                    { type: 'USER_FETCHED', payload: resp.data }
                )
            }).then(()=>{
                history.push('/dashboard')
            })
            .catch(e => {
                console.log(e)
                e.response.data.erros.forEach(
                    erros => toastr.error('Erro', erros))

            })
    }
}

export function logout() {
    return { type: 'TOKEN_VALIDATED', payload: false }
}
```

```
export function validateToken(token) {
    console.log('entrei na validação do token')
    return dispatch => {
        if (token) {
            axios.post('/validateToken', { token })
                .then(resp => {
                    console.log(resp.data.valid)
                    dispatch({ type: 'TOKEN_VALIDATED', payload: resp.data.v
alid })
                })
                .catch(e => dispatch({ type: 'TOKEN_VALIDATED', payload: fal
se }))
        } else {
            dispatch({ type: 'TOKEN_VALIDATED', payload: false })
        }
    }
}
```

10.7.3 *Reducer* de Entrar

```
const userKey = 'GestCop'
const INITIAL_STATE = {
    //nomeUtilizador: "153071", password:"sdsd",
    utilizador: JSON.parse(localStorage.getItem(userKey)),
    validToken: false
}
export default (state = INITIAL_STATE, action) => {
    switch (action.type) {
        case 'TOKEN_VALIDATED':
            if (action.payload) {
                //console.log('entrei na validação do tokem reducer')
                return { ...state, validToken: true }
            } else {
                localStorage.removeItem(userKey)
                return { ...state, validToken: false, utilizador: null }
            }
        case 'USER_FETCHED':
            localStorage.setItem(userKey, JSON.stringify(action.payload))
            return { ...state, utilizador: action.payload, validToken: true
}
        default:
            return state
    }
}
```

10.7.4 *Component* de Entrar

```
import React, {Component} from 'react'

import {connect} from 'react-redux'
import {bindActionCreators} from 'redux'
//import {changeValueEmail, changeValuePassword} from './entrarActions/entra
rActions'
//import {login} from '../../auth/authActions'
import {login} from './entrarActions/entrarActions'
import { Field, reduxForm } from 'redux-form'

import {toastr} from 'react-redux-toastr'

import ContentHeader from '../../components/template/contentHeader/contentHe
ader'
import Content from '../../components/template/content/content'

import Input from '../../common/form/input'

class Entrar extends Component{

    constructor(props){
        super(props)
    }

    onSubmit(values){
       this.props.login (values)
    }

    render(){

        const {handleSubmit} = this.props
        return(
            <div>
                <ContentHeader titulo="" small="v 1.0.0"/>
                    <div className="row">
                        <div className="  col-lg-3 col-xs-12 "></div>

                            <div className="  col-lg-6 col-xs-12 ">
                            <Content>
```

```
                                <div className="card card-primary">
                                    <div className="card-header">
                                        <div className="login-
logo"><b> Gest</b>cop</div>
                                            <div className="login-box-body">
                                            <p className="login-box-
msg">Bem vindo!</p>
                                        </div>
                                    </div>
                                    {/* <!-- /.card-header --> */}

                                    <form onSubmit={handleSubmit(v => this.o
nSubmit(v))}>
                                        <div className="card-body">
                                            <Input nome="nomeUtilizador" lab
el="Utilizador" type="texte" className="form-
control" placeholder= "Nome de Utilizador" required="true"/>
                                            <Input nome="password" label="Pa
ssword" type="password" className="form-
control" placeholder= "Password" required="true"/>
                                        </div>
                                        {/* <!-- /.card-body --> */}
                                        <button type="submit" className="btn
 btn-primary float-right mr-5 mb-4">Entrar</button>

                                    </form>
                                </div>
                            </Content>
                        </div>
                    <div className="col-xs-1 col-sm-1 col-md-2 col-lg-3 col-
xl-4"></div>
                </div>
            </div>
        )
    }
}
Entrar = reduxForm({form: 'entrarForm',
                    })(Entrar)
const mapStateToProps = state => ({nomeUtilizador: state.utilizador.nomeUtil
izador, password: state.utilizador.password})
const mapDispatchToProps = dispatch => bindActionCreators({login}, dispatch)
export default connect(mapStateToProps, mapDispatchToProps)(Entrar)
```

10.8 Parametrização de instruções SQL utilizada

```
        getNomeUtilizador: function (nomeUtilizador, done) {
            const queryString = 'select nome, nomeUtilizador, password, idTip
            oUtilizador from utilizadores where nomeUtilizador = ? LIMIT 1;'
             connection.query(queryString, [nameUser], function (erro, utiliz
             ador, fields) {
              if (erro) {
                console.log('erro de pesquisa na base de dados ', erro)
                return done(erro, utilizador)
              } else if (!utilizador){
                return done(null, false)
              }
              console.log(utilizador[0])
              return done(null, utilizador[0])
            })
}
```

10.9 Cifra da password

```
router.post('/', (req, res) => {
  console.log("entrei na rota utilizador")
  if (req.body.nomeUtilizador && req.body.password) {
    const password = req.body.password
    Utilizador.getNomeUtilizador(req.body.nomeUtilizador, function (erro, ut
ilizador) {
      if (erro) {
        return res.status(400).send({ erros: ["Erro de base de dados"] })
      } else if (utilizador) {
        return res.status(400).send({ erros: ['Utilizador já existente'] })
      } else {
        const passwordHash = bcrypt.hashSync(password, salt)
        Utilizador.insertUtilizador(req.body.nome, req.body.nomeUtilizador,
passwordHash, function (erro, result){
          if (erro) {
            return res.status(400).send(erro)
          } else if (result) {
            return res.status(200).send({ msg: ['Utilizador inserido com suc
esso'] })
          }
        })
      }
    })

  }
}
```

10.10 Utilização Tokens JWT

10.10.1 Atribuição de token a utilizador

```
router.post('/', (req, res) =>{
  console.log("entrei no login")
  if(req.body.nomeUtilizador && req.body.password){
      const nomeUtilizador = req.body.nomeUtilizador
      const password = req.body.password
      const CHAVE = process.env.SENHA
      Utilizador.getNomeUtilizador(nomeUtilizador, function (erro, utilizado
      r) {
        if (erro){
            res.status(400).send({erros:["Erro no acesso à BD"]})
            return done(erro)
        } else if (!utilizador || !password) {
            res.status(400).send({erros:["Utilizador invalido"]})
        }
        else {
          console.log("Utilizador de rota" +utilizador)
          console.log(utilizador.password)
          console.log(password)
          if (utilizador && bcrypt.compareSync(password,utilizador.password)
) {
            const jwtPayload = {idUtilizador : utilizador.idUtilizador}
            const jwtToken = jwt.sign({idUtilizador : utilizador.idUtilizado
r}, CHAVE, {
              expiresIn: "1 day"
            })
           return res.json({
                  nome : utilizador.nome,
                  nomeUtilizador: utilizador.nomeUtilizador,
                  idTipoUtilizador: utilizador.idTipoUtilizador,
                  token: jwtToken}
            )
          } else {
            res.status(400).send({erros:["Password errada"]})
          }
        }
      })
  }
})
```

10.10.2 Verificação de Foken de Utilizador – backend

```
const CHAVE = process.env.SENHA
    const jwtSession = process.env.JWTSESSION
    const Utilizador = require('../models/utilizadores')
    const opts = {}
    opts.secretOrKey = CHAVE
    //vai extrair o token do cabeçalho
    opts.jwtFromRequest =  ExtractJwt.fromHeader('authorization')

    module. exports = function (){
        const strategy = new JwtStrategy (opts,
        function (jwtPayload, cb) {
            Utilizador.getIdUtilizador(jwtPayload.idUtilizador,  (error, uti
lizador) => {
                if (error){
                    console.log('Erro ao obter dados da base de dados')
                    const erro = {erro :['Erro ao obter dados da base de dad
os']}
                    return cb(erro)
                } else if (!utilizador) {
                    console.log(`Utilizador inválido ${jwtPayload.idUtilizad
or}`)
                    const erro = {erro: [`Utilizador inválido ${jwtPayload.i
dUtilizador}`]}
                    return cb(null, false, erro)
                } else {
                    console.log('Utilizador encontrado, validação da passwor
d')
                    console.log(utilizador)
                    if (jwtPayload.password === utilizador.password){


                        console.log(`sucesso ao autenticar : ${utilizador.no
meUtilizador} ID: ${utilizador.idUtilizador} Tipo:${utilizador.idTipoUtiliza
dor}`)
                        cb(null, utilizador)
                    } else {
                        const erro = {erro: ['Password inválida']}
                        console.log('Password Invalida')
                        done(null, false, erro)
                    }
                }
```

```
            }
        )
    })




    passport.use(strategy);
    return {
        initialize: function() {
            return passport.initialize();
        },
        authenticate: function() {
            return passport.authenticate("jwt", jwtSession);
        }
    }

}

function tokenAutenticação (token, cb){
    if(!token) {
        const erro = ({erro: ['Não existe Token']})
        return cb(null, false, erro)
    }
    jwt.verify(token, CHAVE, function(err, decoded) {
        if(err) {
            const erro = ({erro: ['Falha na autenticação do token']})
            return cb(null, false, erro)
        } else {
            console.log('Utilizador autenticado')
        }
    })
}
```

10.10.3 Verificação de Foken de Utilizador – frontend

```
class PrivateRoute extends Component {
    componentWillMount() {

        if (this.props.utilizador.utilizador.token) {
            //console.log('token do utilizador ' + this.props.utilizador.tok
en)
            this.props.validateToken(this.props.utilizador.utilizador.token)
        }
    }
    render() {
        const { utilizador,  validToken} = this.props.utilizador
        //console.log(validToken)
        if (utilizador && validToken) {
            axios.defaults.headers.common['authorization'] = utilizador.toke
n
            return <Route {...this.props}/>
        } else if (!utilizador && !validToken) {
            return <Redirect to ='/entrar'/>
        } else {
            return false
        }
    }
}
```

10.11 Prevenção de ataques XSS

10.11.1 Utilização e implementação da biblioteca Sanitize

```
app.use(require('sanitize').middleware);
```

10.11.2 Utilização e implementação da biblioteca Helmet

```
const helmet = require('helmet')
app.use(helmet({
    defaultSrc: ["'self"],
    acriptSrc: ['*.herokuapp.com'],
    styleSrc: ["'unsafe-inline'"],
    imgSrc: ['*.herokuapp.com'],
    connectSrc: ["'none'"],
    fontSrc:[],
    objectSrc: [],
    mediaSrc: [],
    frameSrc: []
}))
```

10.12 Prevenção de ataques CORS

```
module.exports = function (req, res, next) {

    let i=0
    let notFound =1
    const referer = req.get('Referer');
    const permittedLinker = ['localhost','127.0.0.1', 'https://gestcop.herokuapp.com/']

    if (referer){
        while ((i<permittedLinker.length) && (notFound)){
            notFound = (referer.indexOf(permittedLinker[i]) === -1 )
            i++
        }
    }
    if (notFound){
        res.send("Pedido não autorizado")
    } else{
        res.header('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE')
        res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-
Type, Accept, Authorization')
        next()
    }
}
```