

Universidade da Beira Interior

Departamento de Informática



**Departamento de
Informática**

Nº 78 - 2020: *Sistema de Informação para Simulação de Seguros*

Elaborado por:

Pedro Daniel Pinho Moreira

Orientador:

Professor Doutor Nuno Pombo

7 de Setembro de 2020

Agradecimentos

A conclusão deste projeto, bem como da maior parte da minha vida académica, não seria possível sem o apoio da minha família, nomeadamente dos meus pais. Um enorme OBRIGADO a vocês por terem feito de tudo para que eu pudesse concretizar este sonho. Vocês são um grande exemplo de vida, como pessoa e ser humano, sem o vosso apoio, dedicação, força e presença os meus feitos nada seriam! Obrigado por estarem sempre presentes, mesmo a 200km de distância. Vocês são exemplo de que com trabalho e dedicação, tudo se alcança e espero poder-vos retribuir tudo que me proporcionaram. Amo-vos muito.

Em segundo lugar, quero agradecer ao meu apêndice, à minha namorada, Rita Correia, por ter escrito comigo estes últimos anos e pela sua teimosia em embarcarmos na aventura do *ERASMUS*, sem essa bagagem a minha visão do Mundo seria outra.

Ao meu professor orientador, Prof. Dr. Nuno Pombo, e ao Rui Cameira pela vossa dedicação, confiança e oportunidade depositada em mim. As vossas intervenções foram essenciais para o desfecho deste projeto e espero não vos ter dececionado.

Um grande obrigado aos amigos dos cursos de Engenharia Informática e Informática Web, que apesar das nossas divergências, formamos uma família. Obrigado ao corpo docente que me acompanhou nesta etapa e que sempre me desafiaram a ser melhor, a ver mais além e a continuar com esta sede de conhecimento.

Por fim, mas não menos importante, um grande obrigado a esta bela cidade, localizada na encosta da Serra da Estrela que me albergou e deu a conhecer o que de mais belo existe no interior do nosso país.

Dziękuję.

Conteúdo

| | |
|-------------------------------------------------------------|------------|
| Conteúdo | iii |
| Lista de Figuras | vii |
| Lista de Tabelas | ix |
| 1 Introdução | 1 |
| 1.1 Enquadramento | 1 |
| 1.2 Motivação | 1 |
| 1.3 Objetivos | 2 |
| 1.4 Organização do Documento | 2 |
| 2 Estado da Arte | 5 |
| 2.1 Introdução | 5 |
| 2.2 Interação Humano - Computador | 6 |
| 2.2.1 Interface Gráfica vs Linha de Comandos | 7 |
| 2.3 Otimização | 10 |
| 2.3.1 Plataformas Semelhantes | 10 |
| 2.4 Ferramentas de Automatização | 11 |
| 2.4.1 Vantagens no Uso de Ferramentas de Automatização . . | 11 |
| 2.4.2 Desvantagens no Uso de Ferramentas de Automatização | 11 |
| 2.4.3 Principais Ferramentas de Automatização de Testes . . | 12 |
| 2.5 Base de Dados | 12 |
| 2.5.1 Base de Dados vs Sistema de Ficheiros | 13 |
| 2.5.2 Modelos | 13 |
| 2.5.2.1 Modelo Hierárquico | 13 |
| 2.5.2.2 Modelo Rede | 14 |
| 2.5.2.3 Modelo Relacional | 15 |
| 2.6 Web | 16 |
| 2.7 Conclusões | 17 |
| 3 Tecnologias e Ferramentas Utilizadas | 19 |
| 3.1 Introdução | 19 |

| | | |
|----------|-----------------------------------------------------------|-----------|
| 3.2 | Linguagens Utilizadas | 20 |
| 3.2.1 | <i>Python</i> | 20 |
| 3.2.2 | <i>Structured Query Language</i> (SQL) | 21 |
| 3.3 | <i>SQLite</i> | 22 |
| 3.4 | <i>Qt</i> | 24 |
| 3.4.1 | <i>PyQt</i> | 24 |
| 3.5 | <i>Selenium</i> | 26 |
| 3.5.1 | <i>WebDriver</i> | 27 |
| 3.6 | <i>Integrated Development Environment</i> (IDE) | 28 |
| 3.7 | Conclusões | 29 |
| 4 | Engenharia de Software | 31 |
| 4.1 | Introdução | 31 |
| 4.2 | Levantamento de Requisitos | 32 |
| 4.2.1 | Requisitos Funcionais | 32 |
| 4.2.2 | Requisitos Não Funcionais | 38 |
| 4.2.3 | Requisitos de Domínio | 39 |
| 4.3 | Casos de Uso | 39 |
| 4.4 | Conclusões | 40 |
| 5 | Implementação e Testes | 41 |
| 5.1 | Introdução | 41 |
| 5.2 | <i>Learn PyQt</i> | 43 |
| 5.3 | Base de Dados | 43 |
| 5.3.1 | Diagrama Entidade-Associação | 43 |
| 5.4 | Implementação | 44 |
| 5.4.1 | Segurança | 44 |
| 5.4.2 | <i>Graphic User Interface</i> (GUI) | 47 |
| 5.4.3 | Base de Dados e Comandos | 50 |
| 5.4.3.1 | Criação da Base de Dados | 50 |
| 5.4.3.2 | Pedidos SQL | 51 |
| 5.4.4 | Automatização | 55 |
| 5.5 | Manual do Utilizador | 56 |
| 5.5.1 | Acesso | 56 |
| 5.5.2 | Página Principal | 57 |
| 5.5.3 | Simulação | 57 |
| 5.5.4 | Editar Entrada na Base de Dados (BD) | 59 |
| 5.5.5 | Remover Entrada na BD | 61 |
| 5.5.6 | Adicionar Entrada na BD | 62 |
| 5.6 | Testes | 63 |
| 5.6.1 | Testes Unitários | 63 |

| | |
|------------------------------------------|-----------|
| CONTEÚDO | v |
| 5.6.2 Testes Manuais | 65 |
| 5.6.2.1 Janelas de Confirmação | 66 |
| 5.7 Conclusões | 70 |
| 6 Conclusões e Trabalho Futuro | 71 |
| 6.1 Conclusões Principais | 71 |
| 6.2 Trabalho Futuro | 72 |
| Bibliografia | 73 |

Lista de Figuras

| | | |
|-----|-------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1 | Interação Humano - Computador. | 7 |
| 2.2 | À esquerda encontra-se um exemplo de uma GUI e à direita a mesma função mas através da <i>Command Line Interface</i> (CLI). | 8 |
| 2.3 | Página inicial do simulador da "Deco Proteste". | 10 |
| 2.4 | Página inicial do "Compare o Mercado". | 11 |
| 2.5 | Exemplo de um Modelo Hierárquico. | 14 |
| 2.6 | Exemplo de um Modelo Rede. | 15 |
| 2.7 | Exemplo de um Modelo Relacional. | 16 |
| 2.8 | Funcionalidades principais da <i>Web</i> | 17 |
| 3.1 | Logótipo do <i>Python</i> | 20 |
| 3.2 | Resultado do questionário na vertente de linguagens de programação. | 21 |
| 3.3 | Logótipo do <i>SQLite</i> | 22 |
| 3.4 | Comparação entre os três mais utilizados Sistemas de Gestão de Base de Dados <i>open-source</i> | 23 |
| 3.5 | Resultado do questionário na vertente de linguagem SQL. | 23 |
| 3.6 | Logótipo do <i>PyQt</i> | 25 |
| 3.7 | Arquitetura do <i>Selenium</i> | 26 |
| 3.8 | <i>Selenium WebDriver</i> | 27 |
| 3.9 | À direita encontra-se um exemplo de uma janela do <i>Visual Studio Code</i> (VSCode) e à esquerda o seu logótipo. | 29 |
| 4.1 | Casos de uso do utilizador. | 40 |
| 5.1 | Diagrama Entidade-Associação (DEA) implementado na BD. | 44 |
| 5.2 | Preenchimento da palavra-passe. | 56 |
| 5.3 | Janela de <i>login</i> | 57 |
| 5.4 | Página principal da aplicação. | 57 |
| 5.5 | Página de simulação. | 58 |
| 5.6 | Continuação da página de simulação. | 59 |
| 5.7 | Página de Definições. | 60 |
| 5.8 | Página de Gestão da BD. | 60 |
| 5.9 | Janela de edição da BD. | 61 |

| | | |
|------|------------------------------------------------------------------------------------------------------------------|----|
| 5.10 | Janela <i>pop-up</i> de confirmação de edição. | 61 |
| 5.11 | Janela de remoção da BD. | 62 |
| 5.12 | Janela <i>pop-up</i> de confirmação da remoção. | 62 |
| 5.13 | Janela de adição da BD. | 63 |
| 5.14 | Janela <i>pop-up</i> de confirmação da remoção. | 63 |
| 5.15 | Janela <i>pop-up</i> de insucesso na adição. | 63 |
| 5.16 | Execução do teste ao <i>login</i> | 65 |
| 5.17 | Dados a adicionar na BD. | 66 |
| 5.18 | Sucesso na adição de dados na BD. | 66 |
| 5.19 | À esquerda encontram-se as categorias existentes na BD antes da adição, e à direita após a adição. | 67 |
| 5.20 | Adição de uma marca já existente na BD. | 68 |
| 5.21 | Insucesso na adição de dados na BD. | 68 |
| 5.22 | Janela de edição de categoria na BD. | 68 |
| 5.23 | Sucesso na edição de dados na BD. | 69 |
| 5.24 | À esquerda estão descritas as categorias existentes na BD antes da edição, e à direita após a edição. | 69 |
| 5.25 | Janela de remoção de uma entrada na BD. | 70 |
| 5.26 | Janela de confirmação da remoção de uma entrada na BD. | 70 |

Lista de Tabelas

| | | |
|-----|------------------------------------------------------------------------|----|
| 2.1 | Tabela de comparação entre CLI e GUI. | 9 |
| 3.1 | Tabela de alguns módulos presentes no <i>PyQt</i> | 24 |
| 3.2 | Tabela de comparação entre as ferramentas do <i>Selenium</i> | 27 |
| 4.1 | Organização dos requisitos funcionais da aplicação. | 33 |
| 4.2 | Requisitos funcionais do <i>login</i> | 34 |
| 4.3 | Requisitos funcionais da página principal. | 34 |
| 4.4 | Requisitos funcionais da página de simulação de seguro. | 37 |
| 4.5 | Requisitos funcionais da página de definições. | 37 |
| 4.6 | Requisitos funcionais da página de gestão da BD. | 38 |
| 4.7 | Requisitos não funcionais da aplicação. | 39 |
| 4.8 | Requisitos de domínio da aplicação. | 39 |
| 5.1 | <i>Widgets</i> utilizados da classe <i>QtWidget</i> | 47 |
| 5.2 | <i>Layouts</i> utilizados na GUI. | 48 |
| 5.3 | <i>Windows</i> utilizadas na implementação da GUI. | 48 |

Lista de Excertos de Código

| | | |
|------|----------------------------------------------------------------------|----|
| 3.1 | Exemplo de uma janela com <i>PyQt</i> | 25 |
| 3.2 | Exemplo de uma pesquisa no motor de busca da <i>Google</i> | 28 |
| 5.1 | Função que efetua o <i>login</i> | 45 |
| 5.2 | Funções de cifra e decifra. | 45 |
| 5.3 | Função que gera chaves de cifra. | 46 |
| 5.4 | Classe GUI. | 49 |
| 5.5 | Criação da BD. | 50 |
| 5.6 | Preenchimento de uma tabela da BD. | 51 |
| 5.7 | Pedido à BD. | 51 |
| 5.8 | Pesquisa nas tabelas intermédias da BD. | 52 |
| 5.9 | Adição de um valor à BD. | 53 |
| 5.10 | Edição de um valor da BD. | 53 |
| 5.11 | Remoção de um valor da BD. | 54 |
| 5.12 | Pedido à BD feito pela GUI. | 54 |
| 5.13 | Mudança para o correto <i>iframe</i> | 55 |
| 5.14 | Automatização do <i>login</i> do <i>website</i> Caravela. | 55 |
| 5.15 | Função de teste do <i>login</i> | 64 |

Acrónimos

| | |
|---------------|--------------------------------------------------------------------|
| ACID | <i>Atomic, Consistent, Isolated, Durable</i> |
| ALLab | <i>Assisted Living Computing and Telecommunications Laboratory</i> |
| ANSI | <i>American National Standards Institute</i> |
| BD | Base de Dados |
| CERN | Organização Europeia para a Pesquisa Nuclear |
| CLI | <i>Command Line Interface</i> |
| DEA | Diagrama Entidade-Associação |
| GUI | <i>Graphic User Interface</i> |
| HTML | <i>HyperText Markup Language</i> |
| HTTP | <i>Hypertext Transfer Protocol</i> |
| IDE | <i>Integrated Development Environment</i> |
| ISO | <i>International Organization for Standardization</i> |
| MVC | <i>Model-View-Controller</i> |
| RDBMS | <i>Relational Database Management Systems</i> |
| SEQUEL | <i>Structured English Query Language</i> |
| SGBD | Sistema de Gestão de Base de Dados |
| SHA | <i>Secure Hash Algorithm</i> |
| SQL | <i>Structured Query Language</i> |
| SO | Sistema Operativo |
| TI | Tecnologias de Informação |
| UBI | Universidade da Beira interior |

UC Unidade Curricular

URL *Uniform Resource Locator*

VSCode *Visual Studio Code*

WWW *World Wide Web*

Glossário

Android É um Sistema Operativo móvel *open-source*, baseado em Linux. 12, 24

Appium É uma ferramenta de automatização de testes aplicados em páginas *Web* para dispositivos móveis e para aplicações híbridas no *Android* e *iOS*. O *software* é *open-source*. 12

C# Lê-se *C Sharp* e é uma linguagem de programação desenvolvida pela *Microsoft* nos anos 2000. 12, 26

Chrome É um *browser* gratuito e multi-plataforma, desenvolvido pela *Google* e lançado em 2008. 12, 26

Command Line Interface Deriva do vocabulário inglês *interface*, sendo a conexão entre duas máquinas de qualquer tipo, fornecendo assim um suporte para a comunicação em vários níveis. Neste caso, refere-se à conexão física e funcional entre o dispositivo e o utilizador, através da linha de comandos. vii, xiii, xv, 5

Edge É um *browser* gratuito e multi-plataforma, desenvolvido pela *Microsoft* e lançado em 2015. 12, 27

Graphic User Interface Semelhante à *Command Line Interface* diferindo no facto de o utilizador interagir de forma mais simplificada e com uma forte componente visual. iv, xiii, 5

HyperText Markup Language É uma linguagem utilizada na construção de páginas *Web*, na qual estas podem ser interpretadas pelo *browser*. xiii, 16

Integrated Development Environment É um editor de texto utilizado pelo programador para escrever, editar, formatar código, etc. Como exemplo temos o *Visual Studio Code*. iv, xiii, xvii, 20

JavaScript É uma linguagem de programação baseada na linguagem ECMAScript. Atualmente é a linguagem de programação mais utilizada em *Client-Side* nos *browsers*. 12, 55

Java É uma linguagem de programação orientada a objectos, desenvolvida pela *Sun Microsystems* na década de 90. Hoje pertence à empresa *Oracle*. 12, 26

Linux É um Sistema Operativo desenvolvido pelo programador finlandês Linus Torvalds, sendo este *open-source*. xvii, 12, 24, 26

Mozilla Firefox É um *browser* gratuito e multi-plataforma, desenvolvido pela *Mozilla Foundation* e lançado em 2004. 12, 26

MySQL É uma *Relational Database Management Systems* (RDBMS), lançada no ano 1995, que utiliza a linguagem SQL. Atualmente pertence à empresa *Oracle Corporation*. 22

PHP É uma linguagem de programação desenvolvida em 1995 por *Rasmus Lerdorf*, utilizada para o desenvolvimento de aplicações no lado do servidor. 26

PostgreSQL É uma RDBMS desenvolvida pela *PostgreSQL Global Development Group* e lançada no ano 1996, que utiliza a linguagem SQL. 22

PyQt É uma biblioteca da linguagens de programação *Python* que faz ligação para a biblioteca *Qt*. Atualmente encontra-se na versão 5 e suporta os Sistemas Operativos *Linux*, *macOS* e *Windows*. iv, vii, ix, xi, 19, 24–26, 41, 43, 47, 70, 71

Python É uma linguagem de programação de alto nível orientada a objetos e de forte tipagem, desenvolvida por Guido Van Rossum na década de 90. Hoje pertence à empresa *Python Software Foundation*. iv, 12, 19–22, 24, 26, 43, 50, 63, 70

Qt Ferramenta de desenvolvimento e *design* de interfaces gráficas, desenvolvida pela *Qt Company* e *KDAB* na década de 90. iv, 19, 24, 25

Ruby É uma linguagem de programação desenvolvida na década 90 por *Yukihiro "Matz" Matsumoto*. 12

SQLite É uma biblioteca escrita em C, que implementa uma base de dados SQL, tendo sido esta desenvolvida por D. Richard Hipp também como um *software open-source*. iv, vii, 19, 22–24

Safari É um *browser* padrão no *macOS*, desenvolvido pela *Apple* e lançado em 2003. 12, 27

Scripting Escrita de múltiplos *scripts* por parte do utilizador. 9, 21

Secure Hash Algorithm É uma família de funções de *hash* publicadas pelo Instituto Nacional de Padrões e Tecnologia (NIST). Atualmente incluem as famílias SHA-0, SHA-1 e SHA-2. xiii, 45

Selenium É uma ferramenta de automação de testes aplicados em páginas *Web*, com o propósito principal de as testar. É escrito em Java e pode ser executado em *Windows*, *Linux* e *macOS*, sendo também um *software open-source*. iv, vii, ix, 12, 19, 20, 26, 27, 55

Structured Query Language É uma linguagem padrão de base de dados utilizada para criar, manter, manipular e recuperar a base de dados relacional. iv, xiii, 12

Tricentis É uma linguagem de programação desenvolvida na década 90 por Yukihiro "Matz" Matsumoto. 12

Unified Functional Testing É uma ferramenta de automatização de testes desenvolvida pela *Micro Focus*. Faz uso de *VBscripts*. 12

Visual Basic Script É uma versão interpretada da linguagem *Visual Basic* utilizada para tarefas e construção de páginas HTML desenvolvida pela *Microsoft*. 12

Visual Studio Code O *Microsoft Visual Studio Code* é um *Integrated Development Environment* desenvolvido pela *Microsoft* e lançado em 2015, que contem um conjunto de ferramentas, mais concretamente extensões de desenvolvimento, projectadas para auxiliar os programadores a enfrentarem desafios complexos. vii, xiv, xv, 28

Windows É um Sistema Operativo desenvolvido pela *Microsoft*. xvii, 12, 24, 26

browser É um navegador de Internet, sendo este um programa de computador que permite a interação, por parte do utilizador, dos conteúdos presentes numa página *Web*. xv, xvi, 12, 16, 26, 27, 38

framework Conjunto de bibliotecas ou componentes que são utilizadas para criar uma base onde as aplicações podem ser construídas. Tem como principal objetivo resolver problemas recorrentes, permitindo ao programador focar-se na resolução de problemas ao invés de reescrever código. 19, 24, 26, 64

iOS É um Sistema Operativo móvel desenvolvido pela *Apple*, sendo este executado em todos os modelos *iPhone*, *iPad* e *iPod*. 12, 24

macOS É um Sistema Operativo desenvolvido e instalado por defeito nos produtos da *Apple*. xvi, xvii, 12, 24, 26

open-source É utilizado na informática para se referir a programas que, apesar de serem licenciados, a sua utilização é gratuita, podendo ser redistribuída e modificada. vii, xv–xvii, 12, 22, 23, 26

script Programa escrito para um determinado Sistema Operativo, de forma a automatizar a execução de tarefas que poderiam ser, de certa forma, executadas pelo utilizador. xvi, 7, 11, 12

software É a parte lógica do computador. Esta pode executar processos de manipulação, instruções de execução, redirecionamento e execução das actividades lógicas das máquinas. xv–xviii, 7, 13, 24, 29, 43

Base de Dados Conjunto de informação estruturada e relacionada entre si. iv, xiii, xviii, 2

Sistema de Gestão de Base de Dados É um *software* que permite fazer de forma eficaz e rápida a criação e manipulação da Base de Dados. xiii, 12

Sistema Operativo É um conjunto de programas (*software*) que permite fazer, de forma eficaz, a gestão dos recursos de um computador. Temos como exemplo a família *Windows*, as distribuições *Linux* e a família *macOS*. xiii, xv–xviii, 2

TCP/IP É um conjunto de protocolos de comunicação entre computadores ligados à rede. O nome TCP/IP surge da união entre dois protocolos: o TCP (*Transmission Control Protocol*) e o protocolo IP (*Internet Protocol*). 16

Capítulo

1

Introdução

*"Sometimes it is the people no one
can imagine anything of who do
the things no one can imagine."*

— Alan Turing —

1.1 Enquadramento

O presente projeto foi desenvolvido no âmbito da Unidade Curricular (UC) de Projeto, de 3º ano, sendo dirigido aos alunos dos cursos de Engenharia Informática e Informática Web da Universidade da Beira interior (UBI).

Este projeto consiste na criação de uma aplicação que permita aceder a páginas *Web* e simular de forma automatizada seguros automóveis. O trabalho foi realizado em parceria com a *Universalis Risk Management* no laboratório *Assisted Living Computing and Telecommunications Laboratory* (ALLab), situado no Departamento de Informática e Matemática da UBI.

O acesso à aplicação está restrito por uma *password*, a qual só é do conhecimento do respetivo utilizador. Após a validação do mesmo, o utilizador terá acesso a uma interface gráfica na qual poderá fazer a introdução dos dados necessários para fazer uma simulação do seguro automóvel.

1.2 Motivação

Sendo o tempo um bem essencial na vida do Ser Humano, é notável desde a Revolução Industrial, um aumento da automatização de processos do nosso

quotidiano. As Tecnologias de Informação (TI) andam de mãos dadas com essa crescente automatização, desde meros braços robóticos a carros autônomos.

A simulação de seguros é uma parte constituinte de qualquer seguradora, apresentando-se como um trabalho moroso e repetitivo devido à introdução exaustiva de dados semelhantes em diversas páginas *Web*. Assim sendo, torna-se bastante oportuno investigar sobre as soluções existentes para a automatização desta atividade de simulação.

1.3 Objetivos

Os objetivos principais estabelecidos para a elaboração do projeto são os seguintes:

- Contextualização com as tecnologias e linguagens utilizadas e consequente preparação do ambiente de desenvolvimento;
- Levantamento de requisitos com colaboração da *Universalis Risk Management*;
- Automatização dos *websites*;
- Criação da aplicação no Sistema Operativo (SO) *Linux*;
- Implementação de uma Base de Dados (BD);
- Migração da aplicação no SO *Windows*;
- Melhorias e testes.

1.4 Organização do Documento

De modo a refletir o trabalho que foi desenvolvido, este documento encontra-se estruturado nos seguintes capítulos:

1. O primeiro capítulo - **Introdução** - apresenta o projeto, a motivação, o enquadramento, os seus objetivos e a respetiva organização do documento.
2. O segundo capítulo - **Estado da Arte** - analisa diversas tecnologias e conceitos existentes na área das ferramentas de automatização, comparando-as de modo a entender a escolha final que foi implementada no projeto.

3. O terceiro capítulo - **Tecnologias Utilizadas** - descreve e pormenoriza os conceitos mais importantes no âmbito do desenvolvimento deste projeto, tal como as tecnologias que foram utilizadas.
4. O quarto capítulo - **Engenharia de Software** - apresenta os requisitos, diagramas e modelos utilizados para a construção da aplicação.
5. O quinto capítulo - **Implementação e Testes** - descreve a forma de implementação e como foram realizados os testes à aplicação.
6. O sexto capítulo - **Conclusões e Trabalho Futuro** - descreve as conclusões finais sobre o projeto e o trabalho que poderá ser realizado futuramente de forma a complementar, melhorar e estender todo o trabalho realizado até à data.

Capítulo

2

Estado da Arte

2.1 Introdução

Este capítulo irá abordar os elementos fundamentais para a elaboração de uma aplicação, as tecnologias de automatização existentes e uma breve comparação entre elas.

Um dos papéis fundamentais de qualquer aplicação passa pela boa interação desta com o utilizador, através de uma *Graphic User Interface* (GUI) ou de uma *Command Line Interface* (CLI), pelo que o seu bom desenvolvimento, tal como a praticabilidade da mesma por parte do utilizador, são em grande parte as responsáveis pelo uso ou desuso de um determinado *software*.

Este capítulo encontra-se organizado pelas seguintes secções:

1. Secção 2.2 - **Interação Humano - Computador** - descreve os conceitos de GUI e CLI.
2. Secção 2.2.1 - **Interface Gráfica vs Linha de Comandos** - compara as duas possibilidades de interação com a aplicação.
3. Secção 2.3 - **Otimização** - descreve a otimização necessária para a aplicação.
4. Secção 2.3.1 - **Plataformas Semelhantes** - estuda as aplicações semelhantes já existentes.
5. Secção 2.4 - **Ferramentas de Automatização** - descreve o conceito de automatização, apresentando algumas ferramentas nesse domínio.

6. Secção 2.4.1 - **Vantagens do Uso de Ferramentas de Automatização** - refere algumas vantagens na utilização de ferramentas de automatização.
7. Secção 2.4.2 - **Desvantagens do Uso de Ferramentas de Automatização** - refere algumas desvantagens na utilização de ferramentas de automatização.
8. Secção 2.4.3 - **Principais Ferramentas de Automatização de Testes** - compara algumas ferramentas de automatização de testes já existentes.
9. Secção 2.5 - **Base de Dados** - caracteriza o conceito de BD e revela a sua importância no contexto aplicacional.
10. Secção 2.5.1 - **Base de Dados vs Sistema de Ficheiros** - compara as duas possibilidades de armazenamento e acesso de informação.
11. Secção 2.5.2 - **Modelos** - expõe os três modelos clássicos de uma BD.
12. Secção 2.6 - **Web** - define o conceito em questão, referindo algumas funcionalidades fundamentais no seu domínio.
13. Secção 2.7 - **Conclusões** - trata das principais conclusões do capítulo.

2.2 Interação Humano - Computador

Com o avanço tecnológico e o aprimoramento da interação entre Humano e computador, existe uma crescente necessidade da existência de interfaces de fácil utilização no quotidiano de todos os utilizadores. Avaliando empiricamente, é correto manifestar a dependência crescente da tecnologia por parte das pessoas, sendo estas cada vez mais exigentes com a qualidade da mesma, e não admitindo perdas de tempo na utilização dos recursos tecnológicos [28].

Assim sendo, a necessidade da construção de uma interface amigável, bem projetada, compreensível, segura e funcional para o utilizador é fundamental em qualquer aplicação ou sistema computacional, pois permite que os mesmos se sintam satisfeitos e seguros ao realizar na mesma qualquer operação [10].

A interação entre ambas as partes é portanto um dos pontos cruciais de qualquer solução aplicacional que envolva a manipulação direta de um *software* por parte de um utilizador, sendo que é através desta interação que o mesmo pode tirar partido das funcionalidades da máquina ou programa.

Sabendo que a máquina não comunica na mesma linguagem do utilizador, é necessário a elaboração de uma língua comum aos dois, para que assim ambos possam comunicar de forma simples e correta. Esta interação, feita por meio de uma interface, está representada na figura 2.1.

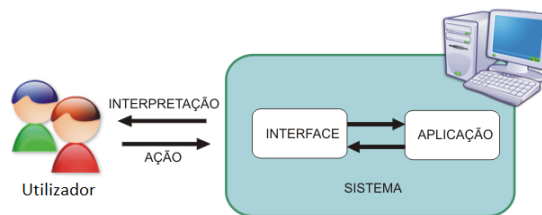


Figura 2.1: Interação Humano - Computador.

Esta interface pode ser uma simples CLI ou uma GUI, sendo que a comparação entre ambas será debatida na subsecção 2.2.1.

2.2.1 Interface Gráfica vs Linha de Comandos

Apesar de ambas as formas de interação Humano-Computador terem finalidades idênticas, como se pode observar na figura 2.2, através de uma GUI embutida no SO (neste caso o *Zorin*), o utilizador poderá proceder a uma representação visual mais apelativa e perceptível dos mecanismos e operações realizadas, bem como dos resultados obtidos. No caso da imagem em baixo representada (2.2), podemos observar o processo de realizar atualizações do *software*. Esta operação poderá ser executada através de *scripts* na CLI (como podemos ver na imagem do lado direito da mesma figura) embutida no mesmo SO ou com poucos cliques na GUI, tendo cada uma destas soluções os seus prós e contras. De forma a tornar essas divergências mais notáveis, a tabela 2.1 irá comparar, de forma aprofundada, as vantagens e desvantagens do uso de ambas.

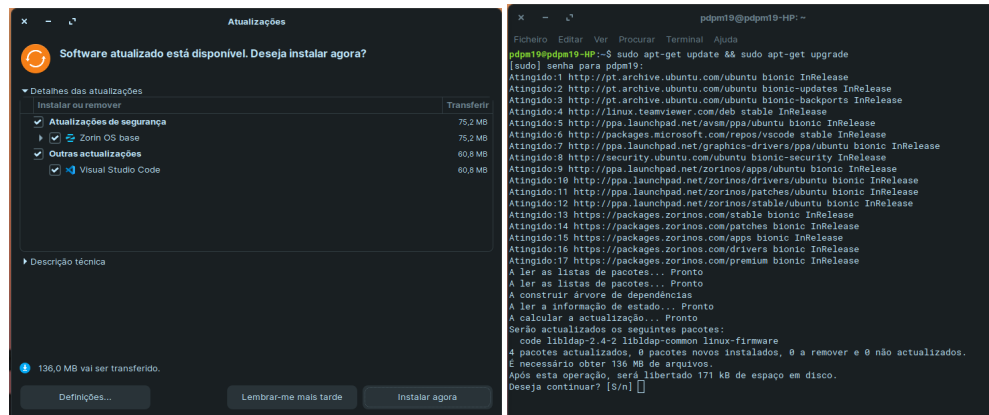


Figura 2.2: À esquerda encontra-se um exemplo de uma GUI e à direita a mesma função mas através da CLI.

| Tópico | CLI | GUI |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Facilidade | Novos utilizadores poderão achar a CLI mais difícil do que a GUI devido ao facto de ser necessário memorizar e estar familiarizado com os comandos necessários para executar a tarefa desejada. | Como a GUI tende a ser visualmente intuitiva, os utilizadores tendem também a aprender a utilizar a GUI mais rapidamente face à CLI. |
| Controlo | Apesar de os utilizadores terem acesso tanto a ficheiros como ao SO através do uso da CLI, para aqueles que utilizam de forma casual, torna-se menos <i>user-friendly</i> face à GUI. | A GUI oferece as mesmas capacidades que a CLI, só que de forma mais <i>user-friendly</i> , sendo utilizada em grande maioria por utilizadores menos experientes. |
| Rapidez | A CLI necessita apenas de um teclado para navegar pela interface, resultando assim numa maior rapidez e fluidez. | As GUIs modernas, apesar de serem eficientes e rápidas, requerem o manuseio de um rato, pelo que os utilizadores necessitam então de ir alternando entre escrever no teclado e mexer no rato, obtendo assim uma queda na rapidez e fluidez na execução das tarefas. |

| | | |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Multi-tarefas | Apesar de ser capaz de executar multi-tarefas, não apresenta a mesma facilidade e habilidade para os utilizadores de visualizarem múltiplas tarefas de uma só vez no ecrã. | Como as GUIs têm janelas, possibilitam aos utilizadores a capacidade de controlarem, manipularem e alternarem entre diversos programas e pastas em simultâneo. |
| Recursos | Uma máquina que utilize só a CLI necessita de muito menos recursos face a uma outra que recorra à GUI. | Uma GUI necessita de mais recursos da máquina devido ao facto de ter de carregar mais elementos, sendo estes maioritariamente fontes e ícones. As <i>drivers</i> de vídeo, rato (entre outras) necessitam também de ser carregadas. |
| <i>Scripting</i> | É difícil que utilizadores inexperientes criem <i>scripts</i> , devido à necessidade de conhecimento de comandos que a CLI apresenta a estes. | Através da documentação e comunidade existente relativa a determinada GUI, podem ser disponibilizados tutoriais e guias de como programar específicas funções. Os utilizadores conseguem assim criar <i>scripts</i> , mesmo sem conhecerem todos os comandos e a sua sintaxe. |
| Acesso remoto | Apesar de ser fazível, o acesso remoto através de uma CLI obriga o utilizador a dominar os comandos da mesma. | O acesso remoto através de uma GUI é algo simples e geralmente intuitivo, mesmo para quem tem pouca experiência. |
| Diversidade | Após o utilizador dominar a CLI, as mudanças ou atualizações futuras, não interferem, por norma, nos comandos já existentes. | Cada GUI tem diferentes <i>designs</i> e estruturas, quando se executam tarefas distintas. |

Tabela 2.1: Tabela de comparação entre CLI e GUI.

Pode-se então concluir que a GUI é usada em grande maioria por utilizadores casuais e inexperientes devido à sua característica principal de ser *user-friendly*, enquanto que a CLI é utilizada, na maioria, por utilizadores ex-

perientes de forma a estes serem mais eficientes e rápidos na execução de tarefas [19].

2.3 Otimização

Sendo o tempo um recurso finito, torna-se útil a implementação de aplicações que possam automatizar parte de certos processos.

A simulação de seguros em geral é um processo moroso, devido à quantidade de dados que têm que ser introduzidos sistematicamente e de forma repetida, para que assim seja possível satisfazer o cliente com um valor justo. Como tal é possível padronizar estes dados, de forma a que estes possam ser introduzidos uma única vez numa plataforma comum.

2.3.1 Plataformas Semelhantes

Na Internet é possível encontrar diversos simuladores destinados unicamente à seguradora detentora do *website*, pelo que existe um número reduzido destes que conseguem cruzar dados entre algumas seguradoras e apresentar valores finais ao utilizador. Com este propósito existem os simuladores de seguros da "Deco Proteste" e do "Compare o Mercado" (figuras 2.3 e 2.4) [27] [24].



Figura 2.3: Página inicial do simulador da "Deco Proteste".

Apesar destes *websites* fazerem a comparação e simulação de seguros com diversas seguradoras, têm como finalidade o seu uso por parte do cidadão comum e não um corretor de seguros. Como tal, este não poderá aplicar os bónus que as seguradoras lhe fazem, nem comparar apenas aquelas com as quais trabalha.



Figura 2.4: Página inicial do "Compare o Mercado".

2.4 Ferramentas de Automatização

Existe uma panóplia de ferramentas de automatização com finalidades distintas, que abrangem desde *marketing* até às redes sociais, passando por ferramentas de automatização de testes. Estas últimas foram utilizadas no desenvolvimento deste projeto, pelo facto de terem como objectivo o teste de páginas *Web*. Como tal, nesta secção serão apenas abordadas ferramentas de automatização de testes.

2.4.1 Vantagens no Uso de Ferramentas de Automatização

A automatização de tarefas apresenta diversas vantagens, tais como:

- **Poupar tempo** ao utilizador, pois as tarefas serão gravadas num *script* e poderão ser executadas diversas vezes. Caso seja necessário alterar algo no *script*, o mesmo pode ser feito com facilidade por parte do utilizador;
- **Execução rápida**, pois após a criação do *script*, a sua execução é muito mais rápida quando comparada com a execução manual do utilizador, tendo também como vantagem a possibilidade de se repetir várias vezes o mesmo *script*;
- **Feedback rápido**, tendo uma execução rápida, a obtenção de resultados também o é [18].

2.4.2 Desvantagens no Uso de Ferramentas de Automatização

O recurso a ferramentas de automatização apresenta também desvantagens, tais como:

- **Aplicação sem planos**, pois é necessário criar-se um plano ou estratégia a seguir, para que assim os resultados possam ter a finalidade pretendida;
- **Fiabilidade** no sentido em que o *script* pode falhar, não por conter um erro, mas devido a uma falha externa, pelo que o utilizador deverá ser capaz de distinguir entre falha e falso alarme.

2.4.3 Principais Ferramentas de Automatização de Testes

Para uma escolha ideal da ferramenta de automatização a utilizar num determinado contexto, é essencial conhecer um pouco sobre cada uma daquelas já existentes. Podemos então enfatizar as seguintes:

- **Appium**, que apesar de ser uma ferramenta *open-source*, é focada em testar aplicações *Android* e *iOS*, podendo no entanto ser executada em *Windows*;
- **Selenium**, esta é de uma ferramenta de testes portátil desenvolvida por Jason Huggins em 2004. É *open-source*, podendo ser executada em *Windows*, *Linux* e *macOS* e suportando um vasto leque de linguagens de programação, tais como *C#*, *Java*, *Python*, *Ruby*, entre outras, tornando-se assim uma ferramenta flexível;
- **Tricentis**, é uma ferramenta paga que suporta *JavaScript*, sendo que esta só executa em *Windows*;
- **Unified Functional Testing**, é uma ferramenta paga que suporta *Visual Basic Script* com os browsers *Chrome*, *Mozilla Firefox*, *Safari* e *Edge*, podendo ser executado em *Windows*.

2.5 Base de Dados

Uma BD é um conjunto de dados relacionados entre si e armazenados conjuntamente com o mínimo de redundância, para que assim possam servir para múltiplas aplicações. Para tal, os utilizadores recorrem a um Sistema de Gestão de Base de Dados (SGBD) ou a uma linguagem de programação com acesso à BD e que possibilite o envio de *queries Structured Query Language* (SQL).

Numa BD, a informação encontra-se estruturada, facilitando assim a utilidade e longevidade da mesma durante um maior período de tempo [9].

2.5.1 Base de Dados vs Sistema de Ficheiros

O acesso e armazenamento de informação, apesar de parecerem operações simples, apresentam um papel fundamental na fiabilidade e eficiência de todo o *software*. Torna-se assim necessário que estas operações utilizem a memória de forma inteligente e eficaz.

Inicialmente, a solução passava pelo uso de ficheiros, sendo esta uma forma de fácil implementação, mas apresentando as seguintes desvantagens:

- **Difícil manipulação de ficheiros**, principalmente quando é necessário o acesso concorrente a vários;
- **Baixa eficiência**, pois o facto de aplicar um filtro num ficheiro iria necessitar que o mesmo fosse carregado na sua totalidade em memória, podendo ainda ser necessário o carregamento de outros ficheiros;
- **Não existe abstração** entre a manipulação dos dados e a maneira como estes foram guardados;
- **Um registo mal guardado** pode comprometer todo o ficheiro e os que se relacionam com ele.

Estas desvantagens no uso de sistemas de ficheiros foram cruciais para o aparecimento de BDs e de SGBDs nos anos 60, onde a responsabilidade de gerir o acesso, persistência, manipulação e organização dos dados foi retirada da aplicação cliente, tornando-se assim a opção mais viável [21].

2.5.2 Modelos

A modelação é um ponto-chave de um SGBD, sendo que podemos encontrar os seguintes modelos básicos:

- Modelo Hierárquico;
- Modelo Rede;
- Modelo Relacional [34].

2.5.2.1 Modelo Hierárquico

Este modelo foi concebido de forma a adaptar o processamento sequencial, numa estrutura de segmentos (entidades-tipo), onde as entidades de baixo nível dependem das do nível acima, apresentando uma ligação unidirecional. Verifica-se assim a existência de redundância e problemas na introdução, eliminação e atualização dos dados. Este modelo funciona bem quando

as correspondências são de $N - 1$, pelo que as relações assemelham-se a um grafo, estando este representado na figura 2.5 [34] [6].

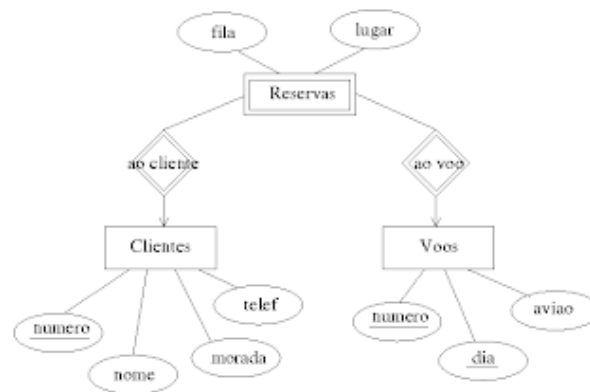


Figura 2.5: Exemplo de um Modelo Hierárquico.

2.5.2.2 Modelo Rede

Este modelo apresenta relações que funcionam em ambos os sentidos, contrariamente aquele apresentado na secção 2.5.2.1, que dispõe de relações unidireccionais. Temos que cada entidade (registo) corresponde a um *Record* (entidade-tipo), onde as pesquisas são efetuadas através do uso de apontadores. O seu uso torna a representação da BD perto da representação física, mas dificulta a independência dos dados.

Apesar dos problemas de redundância e as anomalias de inserção e eliminação não serem tão evidentes face ao modelo anterior, estes resultam de uma questão de normalização. Pode-se observar um exemplo de um Modelo Rede na figura 2.6 [34] [6].

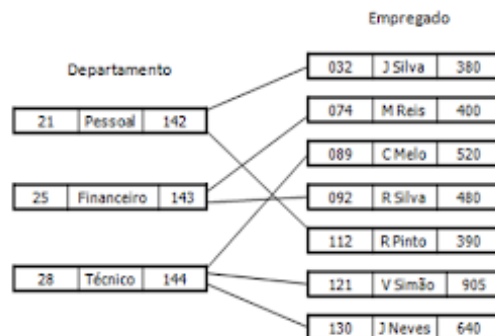


Figura 2.6: Exemplo de um Modelo Rede.

2.5.2.3 Modelo Relacional

Este é o mais simples em termos de conceção, sendo também o mais recente, pois a sua implementação só foi possível devido aos avanços tecnológicos na década de 70. O modelo relacional culmina as dificuldades e limitações dos modelos anteriores, ao mesmo tempo que torna a organização dos dados mais simples e flexível, recorrendo a tabelas para tal. A figura 2.7 apresenta um exemplo de um modelo relacional [34].

Os objetivos dos modelos relacionais passam por:

- Fornecer um elevado grau de independência dos dados;
- Simplificar o trabalho do administrador da BD;
- Apresentar uma definição comum dos dados, de forma simples, para que assim vários utilizadores de diversos níveis possam interatuar com a BD.

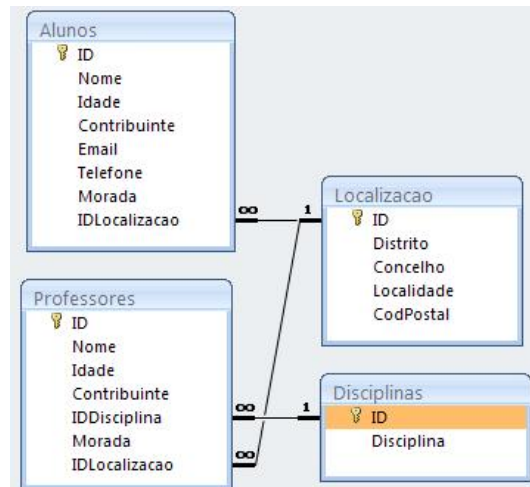


Figura 2.7: Exemplo de um Modelo Relacional.

2.6 Web

A *World Wide Web* (WWW) ou W3 é um serviço de informação baseado na partilha de documentos, onde a informação está armazenada no formato de *HyperText Markup Language* (HTML). A ideia foi desenvolvida por *Tim Berners-Lee*, um físico e investigador britânico, na década 80 enquanto trabalhava na Organização Europeia para a Pesquisa Nuclear (CERN) [12].

A *Web* acenta em três funcionalidades principais:

- ***Hypertext Transfer Protocol* (HTTP)**, que fica responsável por controlar as transferências de dados entre um servidor *Web* e o cliente, recorrendo para tal ao protocolo TCP/IP;
- ***Uniform Resource Locator* (URL)**, onde cada documento ou recurso tem um endereço único;
- **HTML**, estrutura padrão utilizada pelos documentos na *Web* [23].

Pode-se concluir então, que o URL é o *link* presente no *browser*, o qual serve para mostrar onde o item está localizado, sendo o HTTP o protocolo utilizado na transferência de dados entre computadores, e por fim o HTML que representa o código por trás das páginas *Web*. A boa relação entre estas três funcionalidades apresentadas acima e na figura 2.8 apresentam-se como a WWW [2].



Figura 2.8: Funcionalidades principais da *Web*.

2.7 Conclusões

Todos os conceitos abordados englobam-se, de alguma forma, na aplicação desenvolvida. Foi portanto necessário referir os conceitos de GUI e a importância da sua utilização por parte do utilizador, sendo este um aspecto relevante numa aplicação.

A comparação com plataformas já existentes foi útil, demonstrando que existe uma procura por tais plataformas.

No próximo capítulo iremos abordar em detalhe todas as ferramentas utilizadas em todas as fases do projeto desenvolvido.

Capítulo

3

Tecnologias e Ferramentas Utilizadas

3.1 Introdução

Para a realização deste projeto foram utilizadas diversas ferramentas e tecnologias. Desta forma serão abordadas com detalhe as ferramentas e tecnologias que suportam a base de dados, bem como a automatização e desenvolvimento da GUI.

Este capítulo encontra-se organizado nas seguintes secções:

1. Secção 3.2 - **Linguagens Utilizadas** - apresenta as linguagens utilizadas no desenvolvimento da aplicação.
2. Secção 3.2.1 - **Python** - descreve a linguagem de programação e a sua usabilidade.
3. Secção 3.2.2 - **SQL** - descreve a linguagem de pesquisa declarativa utilizada na BD.
4. Secção 3.3 - **SQLite** - refere o porquê da sua escolha e as suas principais características.
5. Secção 3.4 - **Qt** - descreve a *framework* para desenvolvimento de GUIs.
6. Secção 3.4.1 - **PyQt** - descreve a biblioteca utilizada e a sua importância para a realização deste trabalho.
7. Secção 3.5 - **Selenium** - descreve a ferramenta de automatização utilizada.

8. Secção 3.5.1 - **WebDriver** - apresenta a ferramenta *Selenium* utilizada.
9. Secção 3.6 - **Integrated Development Environment (IDE)** - refere qual foi o IDE utilizado no desenvolvimento da aplicação e as suas principais características.
10. Secção 3.7 - **Conclusões** - apresenta as conclusões sobre a preferência de todas as tecnologias e ferramentas apresentadas anteriormente.

3.2 Linguagens Utilizadas

Para a construção e implementação lógica e funcional da aplicação, recorreu-se ao *Python* para o desenvolvimento da GUI e automatização dos *websites* das seguradoras, enquanto que a pesquisa e consulta na BD ficou encarregue ao SQL.

Ambas as linguagens utilizadas foram abordadas em UCs anteriores, tendo sido aprofundado o conhecimento das mesmas na realização deste projeto.

3.2.1 *Python*

A linguagem de programação *Python*, estando o seu logótipo representado na figura 3.1, começou a ser desenvolvida por *Guido van Rossum*, no ano 1989, no *Centrum Wiskunde & Informatica* (CWI), situado na Holanda. Este foi, no entanto, publicado em 1991 e posteriormente desenvolvido pela *Python Software Foundation*.

O *Python* distingue-se das restantes linguagens de programação pela sua fácil interpretação de código (por parte do programador) e pela sua sintaxe simples que permite exprimir conceitos e algoritmos complexos, geralmente, em poucas linhas de código [29].



Figura 3.1: Logótipo do *Python*.

Podemos considerar, como principais vantagens do *Python*, as seguintes:

- **Simples e fácil de aprender**, devido à sua sintaxe, quantidade de documentação e comunidade;

- **Realização de testes**, onde apresenta variadas estruturas de teste integrados;
- Suporte a mecanismos de **Automatização** e **Scripting**;
- **Big Data e Data Science**, sendo amplamente utilizado nestas áreas [22].

Atualmente e apesar da versão mais recente do *Python*, até à data, ser a 3.8, foi utilizada a versão 3.6 para o desenvolvimento total deste projeto.

Um questionário realizado pelo *Stack Overflow* no mês de Fevereiro de 2020, demonstrou o interesse nesta linguagem de programação, tanto por parte dos programadores como das entidades patronais, colocando assim o *Python* em 4º lugar nas linguagens mais utilizadas, como se pode observar na figura 3.2 [32].

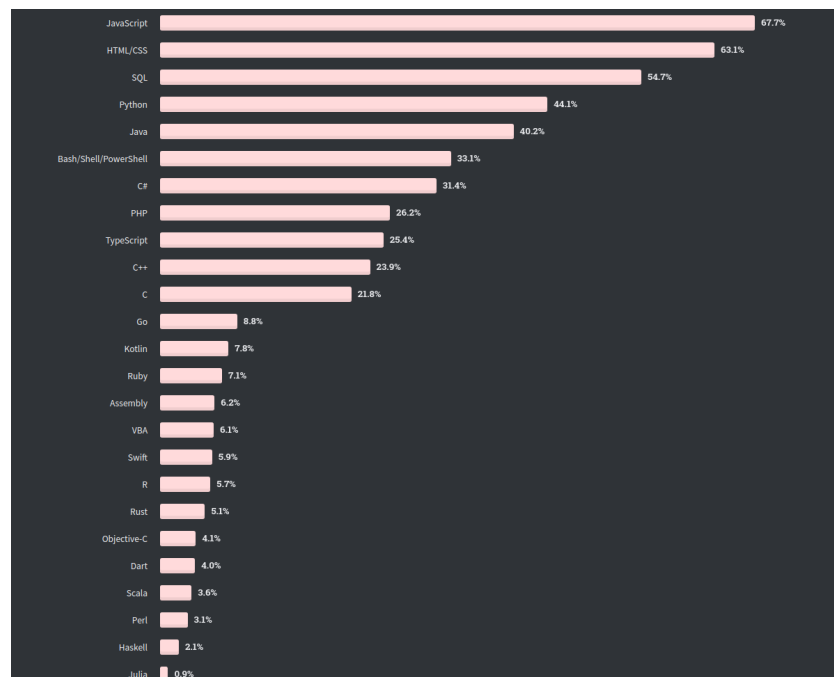


Figura 3.2: Resultado do questionário na vertente de linguagens de programação.

3.2.2 SQL

A linguagem *Structured English Query Language* (SEQUEL) começou a ser desenvolvida no ano de 1970 nos laboratórios da *IBM* pelos investigadores *Donald D. Chamberlin* e *Raymond F. Boyce*, com a finalidade de demonstrar

a viabilidade do modelo relacional, descrito na subsecção 2.5.2.3 do capítulo 2, proposto pelo Dr. *E. F Codd* na sua publicação intitulada de "*A Relational Model of Data for Large Shared Data Banks*" [25] [8].

Mais tarde passaria a chamar-se de SQL, sendo aceite como modelo padrão da linguagem *Relational Database Management Systems* (RDBMS), tendo sido padronizada pela *American National Standards Institute* (ANSI) em 1986 e pela *International Organization for Standardization* (ISO) em 1987 [5].

Para concluir, a linguagem SQL diferencia-se das outras linguagens de consulta à BD, na medida em que cada consulta especifica a forma do resultado, e não o caminho até ele.

3.3 SQLite

A *SQLite* é uma biblioteca *in-process, standalone, serveless, zero-configuration*, baseada em ficheiros, com transações *Atomic, Consistent, Isolated, Durable* (ACID) e *open-source* RDBMS que implementa uma BD SQL. Estando o seu logótipo representado na figura 3.3, esta é amplamente utilizada, contando com presença em variados projetos, tais como, *Adobe, Airbus, Apple, Bosch, Microsoft, Python* entre outros [1].



Figura 3.3: Logótipo do SQLite.

Esta biblioteca difere das restantes RDBMS pela sua portabilidade, fiabilidade, boa performance mesmo em ambientes com pouca memória e por ser *serveless*. Devido à sua característica de ser *serveless*, a BD é representada num único ficheiro, onde todos os processos que acedem à mesma têm obrigatoriamente de ler e escrever diretamente do ficheiro em disco, simplificando assim a instalação e configuração desta.

Em tom comparativo entre os três mais conhecidos SGBD *open-source* (*MySQL, PostgreSQL* e *SQLite*), podemos observar os seus aspetos positivos e menos positivo na figura 3.4, assim como quando é mais oportuno escolher um face aos restantes [11].

| Tópico | SQLite | MySQL | PostgreSQL |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| Vantagens | <ul style="list-style-type: none"> Necessidade de pouco espaço em disco; <i>User-friendly</i>; Portabilidade. | <ul style="list-style-type: none"> Velocidade; Replicação. | <ul style="list-style-type: none"> Conformidade face aos padrões SQL; Extensibilidade. |
| Desvantagens | <ul style="list-style-type: none"> Concorrência limitada; Não há diferenciação entre utilizadores. | <ul style="list-style-type: none"> Funcionalidades limitadas devido à preferência da velocidade; Licença e funcionalidades proprietárias. | <ul style="list-style-type: none"> Memória, pois por cada conexão cliente são alocados 10 MB. |
| Quando utilizar | <ul style="list-style-type: none"> Aplicações embutidas, ou seja único utilizador e aplicações móveis ou jogos; Substituição de acesso ao disco; Testes. | <ul style="list-style-type: none"> Operações distribuídas; <i>Websites</i> e aplicações baseadas em <i>web</i>. | <ul style="list-style-type: none"> Integração com outras ferramentas; Operações complexas. |
| Quando não utilizar | <ul style="list-style-type: none"> Elevada quantidade de dados por gerir; Elevados volumes de escrita. | <ul style="list-style-type: none"> Concorrência e elevada quantidade de informação; Conformidade com SQL necessária. | <ul style="list-style-type: none"> Velocidade é imperativa; Configurações simples. |

Figura 3.4: Comparação entre os três mais utilizados Sistemas de Gestão de Base de Dados *open-source*.

Mais uma vez, através dos resultados do questionário elaborado pelo *Stack Overflow*, podemos observar na figura 3.5 que os três SGBD abordados acima estão no topo dos mais utilizados pelos utilizadores do *Stack Overflow* [32].

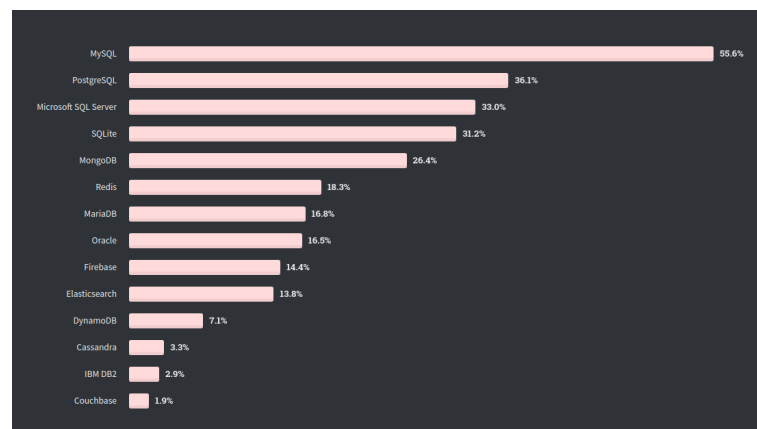


Figura 3.5: Resultado do questionário na vertente de linguagem SQL.

3.4 Qt

O Qt é uma *framework cross-plataform* baseada em C++ e que permite o desenvolvimento de GUIs para *Android*, *iOS*, *Windows*, *Linux* e *macOS*. Para além disso, inclui também abstrações de *sockets* de rede, *threads*, expressões regulares, BD SQL e um extenso conjunto de *widgets* para GUIs. As suas classes aplicam um sistema de sinal/*slot* para comunicarem entre objetos, tornando mais fácil criar componentes reutilizáveis de *software*.

Este inclui também o *Qt Designer* que é uma GUI para efeitos de *design* [33].

3.4.1 PyQt

O PyQt é um conjunto de módulos da linguagem *Python* que permite a ligação à *framework Qt* executando em todas as plataformas por esta suportadas.

Os módulos mais comumente utilizados estão representados na tabela 3.1 [31].

| Nome do módulo | Descrição |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>QtCore</i> | Contém as classes básicas, incluindo o mecanismo de sinal e <i>slot</i> , <i>threads</i> , expressões regulares e definições de aplicação e utilizador. |
| <i>QtGui</i> | Contém as classes para lidar com eventos, gráficos 2D, imagens, fontes e texto. |
| <i>QtSql</i> | Contém as classes que integram BD SQL. Inclui uma implementação do <i>SQLite</i> . |
| <i>QtWidgets</i> | Contém as classes para criar GUIs no estilo de <i>desktop</i> . |

Tabela 3.1: Tabela de alguns módulos presentes no PyQt.

O PyQt, estando o seu logótipo representado na figura 3.6, combina as vantagens do Qt e *Python*, dando origem a um programa com o poder do Qt e a simplicidade do *Python* [30] [16].

Figura 3.6: Logótipo do *PyQt*.

No excerto de código abaixo, podemos observar a criação de uma janela através do uso do *PyQt*, onde para tal, utilizamos os módulos *Widgets* e *Qt-Core*.

```
# Necessario caso a janela receba argumentos pelo terminal
import sys
# Imports do PyQt5
from PyQt5.QtWidgets import QApplication, QLabel, QMainWindow
from PyQt5.QtCore import Qt

# Subclasse QMainWindow que nos permite acrescentar conteudo a Janela
class MainWindow(QMainWindow):

    def __init__(self):
        super(MainWindow, self).__init__()
        # Nome da Janela
        self.setWindowTitle("Janela simples")
        # Localizacao no ecrã (200,200) e dimensoes da janela 600x300
        self.setGeometry(200, 200, 600, 300)

        # Criamos uma caixa de texto
        label = QLabel("Ola Mundo!")
        # Alinhamos o texto ao centro
        label.setAlignment(Qt.AlignCenter)

        # Atribuimos esse texto ao centro da janela
        # Ira manter-se sempre centrado mesmo quando e feito resize a
        # janela
        self.setCentralWidget(label)

if __name__ == "__main__":
    # E necessario 1 instancia QApplication, podemos passar sys.argv
    # para termos acesso aos argumentos passados pelo terminal
    app = QApplication([])

    # Criamos uma janela
    window = MainWindow()
    # Fazemos display da mesma, por defeito elas estao ocultas
```

```

window.show()
# Começa o loop de eventos
app.exec_()
# A execucao do codigo so chega aqui apos o loop acima ter terminado

```

Excerto de Código 3.1: Exemplo de uma janela com *PyQt*.

3.5 Selenium

O *Selenium* é uma *framework open-source* portátil de testes desenvolvida por *Jason Huggins* em 2004, com a finalidade de testar funcionalidades de aplicação baseadas na *Web*. Apresenta-se como uma *framework* multi-plataforma pelo facto de conter versões para *Linux*, *macOS* e *Windows*, suportando várias linguagens de programação, tais como *C#*, *Java*, *PHP* e *Python*.

Este subdivide-se em três ferramentas - *IDE*, *WebDriver* e *Grid* - estando estas retratadas na figura 3.7 [4].

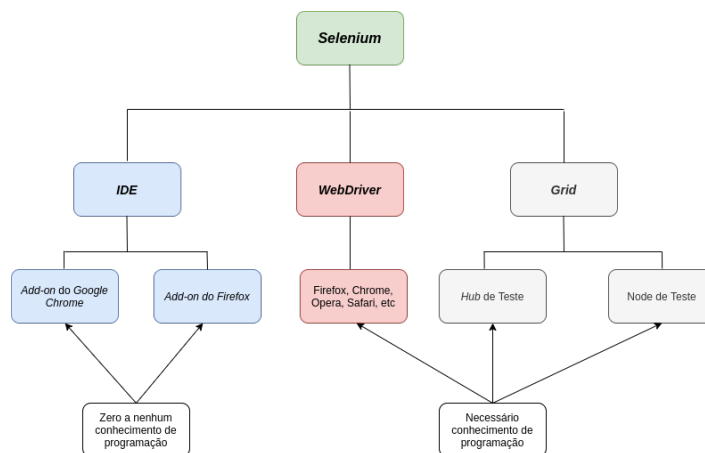


Figura 3.7: Arquitetura do *Selenium*.

Na tabela 3.2 podemos observar, de forma mais concisa, as características de cada ferramenta que completam a suite do *Selenium*.

| Tópico | <i>IDE</i> | <i>Webdriver</i> | <i>Grid</i> |
|-------------------------------------------------------------------|------------|------------------|-------------|
| Apresenta uma GUI | Sim | Não | Não |
| Compatível com <i>Linux</i> , <i>macOS</i> e <i>Windows</i> | Sim | Sim | Sim |
| Compatível com os <i>browsers Chrome</i> e <i>Mozilla Firefox</i> | Sim | Sim | Sim |

| | | | |
|-----------------------------------------------------------------------------------|-------|-------|-------|
| Compatível com outros <i>browsers</i> (<i>Edge</i> , <i>Safari</i> entre outros) | Não | Sim | Sim |
| Conhecimento de programação | Baixo | Médio | Médio |
| Aceita várias máquinas em paralelo | Não | Não | Sim |

Tabela 3.2: Tabela de comparação entre as ferramentas do *Selenium*.

Em 2019 o *Selenium* começou a aplicar os *standards* da W3C, possibilitando assim testes mais consistentes e estáveis em diferentes *browsers* [20].

3.5.1 WebDriver

Esta ferramenta do *Selenium* difere das restantes pelo facto de aceitar a execução de testes em diversos *browsers*, com variadas linguagens de programação numa mesma máquina, como se pode observar na figura 3.8 [3].

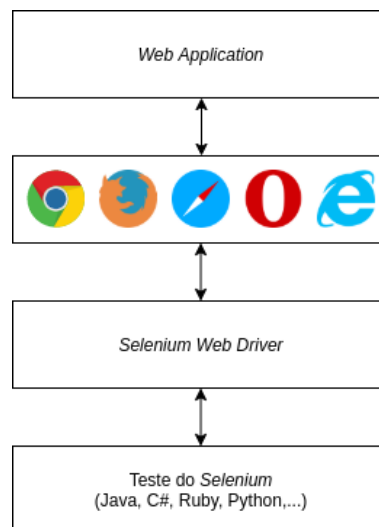


Figura 3.8: *Selenium WebDriver*.

No excerto de código abaixo podemos observar uma pesquisa no motor de busca da *Google*, utilizando o *Google Chrome*, de forma automatizada, onde para tal foi necessário identificar os campos e quais as ações que seriam tomadas.

```
import os
# Imports do Selenium
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.keys import Keys

# vai buscar o path do Chromedriver
path = os.path.join(os.getcwd(), 'chromedriver_linux')
# Browser driver, neste caso ChromeDriver do Linux
driver = webdriver.Chrome(path)
# Pagina que sera aberta na nova aba do browser
driver.get('https://google.pt')
# Ecra total
driver.maximize_window()

# Localiza o campo do motor de pesquisa
pesquisa = WebDriverWait(driver, 10).until(lambda driver: driver.
    find_element_by_name('q'))
# Envia dados
pesquisa.send_keys('Hello World!')
# Localiza o botao "Pesquisar"
submiter = WebDriverWait(driver, 10).until(lambda driver: driver.
    find_element_by_class_name('gNO89b'))
# Carrega nele
submiter.click()
```

Excerto de Código 3.2: Exemplo de uma pesquisa no motor de busca da Google.

3.6 IDE

O IDE utilizado para o desenvolvimento da aplicação foi o *Visual Studio Code* (VSCode), representado na figura 3.9, tendo sido este lançado em 2015 pela *Microsoft* [26].

Este IDE apresenta diversas vantagens, entre as quais:

- **Gratuito**, mesmo sendo um programa da *Microsoft* com o nome *Visual Studio*;
- **Open-source**, garantindo assim facilidade na criação de extensões e uma menor taxa de *bugs*;
- **Multi-Plataforma**;

- ***IntelliSense***, sendo esta uma assistência inteligente à codificação;
- **Integração com o *Git***;
- **Suporte para várias linguagens de programação** através do uso de extensões;
- **Funcionalidades para *debugging*** [7].

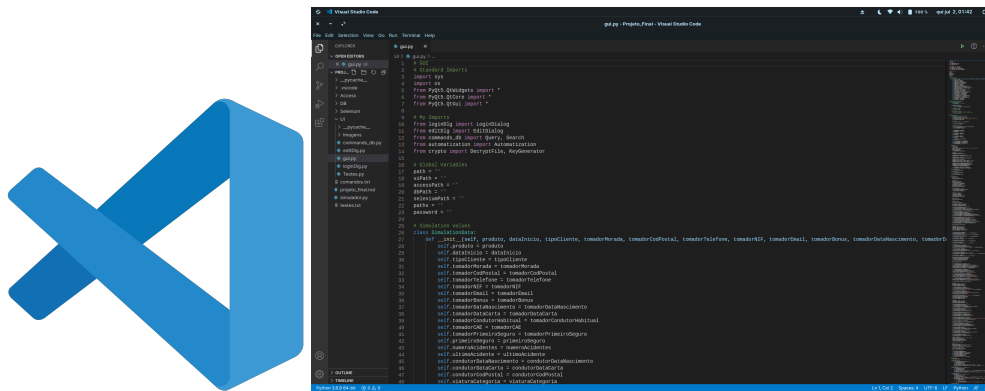


Figura 3.9: À direita encontra-se um exemplo de uma janela do VSCode e à esquerda o seu logótipo.

3.7 Conclusões

Neste capítulo foram caracterizadas e definidas todas as ferramentas e tecnologias utilizadas no desenvolvimento e implementação da aplicação. Todas elas foram essenciais para o cumprimento dos objetivos propostos para o trabalho.

Foram assim descritas as linguagens utilizadas para o desenvolvimento da aplicação, tanto na vertente de *frontend* como de *backend*, bem como os *softwares* utilizados referentes ao *IDE* e *SGBD*.

Engenharia de Software

4.1 Introdução

Neste capítulo irão ser abordados os mecanismos e funções do sistema no contexto da Engenharia de *Software* da aplicação desenvolvida, nomeadamente os requisitos funcionais, não funcionais e de domínio da mesma e o diagrama de casos de uso.

O processo de levantamento de requisitos de *software* é responsável por identificar todas as necessidades, funcionalidades, solicitações, propriedades, características ou restrições de um determinado sistema. Por outro lado, os diagramas de casos de uso representam possíveis utilizações do sistema por um determinado ator.

Este capítulo encontra-se organizado nas seguintes secções:

1. Secção 4.2 - **Levantamento de Requisitos** - apresenta um resumo e breve descrição do conceito de requisito funcional, não funcional e de domínio, bem como a sua importância para o desenvolvimento de um *software*.
2. Secção 4.2.1 - **Requisitos Funcionais** - descreve pormenorizadamente os requisitos funcionais da aplicação.
3. Secção 4.2.2 - **Requisitos Não Funcionais** - descreve pormenorizadamente os requisitos não funcionais da aplicação.
4. Secção 4.2.3 - **Requisitos de Domínio** - descreve pormenorizadamente os requisitos de domínio da aplicação.
5. Secção 4.3 - **Casos de Uso** - especificação dos casos de uso da aplicação por parte de um ator.

6. Secção 4.4 - **Conclusões** - reflexão das principais conclusões do capítulo.

4.2 Levantamento de Requisitos

No domínio do levantamento de requisitos na Engenharia de *Software*, podemos considerar 3 tipos de requisitos principais: funcionais, não funcionais e de domínio.

Os requisitos funcionais descrevem pormenorizadamente todas as funcionalidades e serviços do *software*, documentando como este deve reagir e comportar-se face a determinados *inputs* e situações, bem como o que não deve acontecer, sendo que não devem existir definições contraditórias [13].

Os requisitos não funcionais definem propriedades e restrições do *software* tais como a segurança, a performance, o espaço em disco e a usabilidade, sendo que estes podem ser referentes a partes individuais do sistema, ou a este como um todo. Se os requisitos não funcionais não forem cumpridos, o sistema torna-se inútil [13].

Por outro lado, os requisitos de domínio são intrínsecos ao cliente, sendo de difícil especificação. Estes podem ser novos requisitos funcionais ou até mesmo restringir os requisitos funcionais já existentes. A sua não satisfação, poderá fazer com que o sistema não opere adequadamente.

4.2.1 Requisitos Funcionais

Os requisitos funcionais são requisitos adjacentes às funcionalidades da aplicação, sendo que estes contêm as especificações claras, completas e numeradas das necessidades do *software* para um correto funcionamento e visualização gráfica do mesmo.

A tabela 4.1 explicita a ordem pela qual os requisitos foram encontrados e, conseqüentemente, abordados:

| ID Requisito | Nome do Requisito | Descrição do Requisito |
|--------------|-------------------|---------------------------------------------------------------------------------------------------------------|
| RF01 | <i>Login</i> | Contém todos os requisitos associados a RF01.X. O utilizador não tem que ter conta para aceder a esta página. |
| RF02 | Página Principal | Contém todos os requisitos associados a RF02.X. O utilizador tem de ter conta para aceder a esta página. |

| | | |
|------|------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| RF03 | Página de Simulação de Seguro de Viatura | Contém todos os requisitos associados a RF03.X. O utilizador tem que ter conta para aceder a esta página. |
| RF04 | Página de Definições | Contém todos os requisitos associados a RF04.X. O utilizador tem que ter conta para aceder a esta página. |
| RF05 | Página de edição dos dados na BD | Contém todos os requisitos associados a RF05.X. O utilizador tem que ter conta para aceder a esta página. |

Tabela 4.1: Organização dos requisitos funcionais da aplicação.

| ID Re- quisito RF01.x | Nome do Re- quisito | Descrição do Requisito |
|-----------------------------|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RF01.1 | Janela | Deverá conter uma caixa de texto para o preenchimento da palavra-passe (RF01.2) e dois botões abaixo do campo de preenchimento, um para cancelar a operação, com o nome "Cancel", e outro para verificar a palavra-passe com o nome "OK". A janela deverá conter o nome "Acesso", sem fundo de cor específica, e uma cruz para fechar a mesma, podendo esta encontrar-se à direita ou à esquerda do nome da janela. |

| | | |
|--------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RF01.2 | Palavra-passe | O preenchimento correto deste campo é obrigatório, abrangendo codificação alfanumérica, podendo conter letras capitalizadas ou não. O uso de caracteres especiais é permitido. Caso a palavra-passe introduzida esteja incorreta, deverá aparecer, na caixa de texto, o número de tentativas restantes que o ator terá direito, com a mensagem: "X tentativas restantes...", onde X representa um número inteiro entre 2 e 1. Os caracteres são representados por pontos, de forma a não ser possível a atores externos visualizarem a introdução de dados. Esta caixa de texto deverá ser precedida pelo texto "Palavra-passe:". Ao serem esgotadas as três tentativas consecutivas, a janela encerrará automaticamente, não carregando o resto da aplicação, por outro lado, se a palavra-passe for a correta, a janela encerrará e será carregado o resto da aplicação. |
|--------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Tabela 4.2: Requisitos funcionais do *login*.

| ID Re- quisito RF02.x | Nome do Re- quisito | Descrição do Requisito |
|-----------------------------|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RF02.1 | Janela | Deverá conter o nome "Simulador", assim como opções de minimizar e fechar no canto superior esquerdo ou direito (mas mantendo a coerência da seleção de lado do RF01.1). Deverá conter também o logótipo da <i>Universalis</i> centrado e botões para fazer a simulação de seguro de viatura com o nome "Viatura" e botão de definições com o nome "Definições" ou um ícone alusivo ao mesmo. Estes botões deverão estar centrados e alinhados em grelha. Deverá ser passível exercer <i>scroll</i> down, tornando assim a janela apta para as mais variadas resoluções de ecrãs disponíveis, assim como ser redimensionável. |

Tabela 4.3: Requisitos funcionais da página principal.

| ID Re- quisito RF03.x | Nome do Re- quisito | Descrição do Requisito |
|-----------------------------|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RF03.1 | Janela | Deverá manter as opções de dimensões e <i>scroll</i> da RF02.1, diferenciando no seu conteúdo. A janela será dividida em cinco campos, sendo estes "Dados Seguro" (RF03.2), "Dados Tomador" (RF03.3), "Dados Condu- tor" (RF03.4), "Dados Viatura" (RF03.5) e os botões, em grelha, de voltar e simular com os respectivos nomes "Voltar" e "Simular". |
| RF03.2 | Dados Seguro | Deverá conter uma opção <i>dropdown</i> com os parâmetros "AUTOM - AUTOMÓVEL" e "AUT2R - AUTOMÓVEL 2 RODAS", precedido do texto "Produto", um campo de seleção da data, sendo o valor por defeito a data atual, precedido do texto "Data Início Seguro:". Por baixo deverá conter dois <i>Radio Buttons</i> , seguidos dos textos "Individual" e "Colectivo", estando estes precedidos do texto "Tipo de Cli- ente:". |

| | | |
|--------|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RF03.3 | Dados Tomador | <p>Esta secção deverá estar separada verticalmente em duas colunas, uma delas contendo os parâmetros estáticos "Morada:", "Código Postal:", "Telefone:", "NIF:", "Email:", todos seguidos pelas suas respectivas caixas de texto e "Bónus:"seguido de uma <i>checkbox</i>, organizados seguindo um <i>layout</i> em formulário. A outra coluna deverá conter os parâmetros específicos do género de cliente. No caso de ser um cliente do género colectivo, irá apresentar o texto "CAE:"seguido de uma caixa de texto. No caso de ser um cliente do género individual, irá apresentar, seguindo um <i>layout</i> em formulário, dois campos de data, precedidos dos textos "Data de Nascimento:"e "Data Carta:"e o texto "Tomador é condutor habitual:"seguido de uma <i>checkbox</i>. Independentemente do género do cliente, deverá, por baixo, exibir o texto "1º Seguro Automóvel:"com dois <i>Radio Buttons</i>, à direita do texto, a formarem as respostas "Sim"e "Não". Dependendo da opção seleccionada, será exibido, em grelha, os textos "Seguro Desde:", "Data Último Sinistro:"seguidos de campos de data e uma opção <i>dropdown</i>, com os valores "Zero", "Um"e "Dois"precedida do texto "Nº de Sinistros nos Últimos 2 anos:".</p> |
| RF03.4 | Dados Condutor | <p>Seguindo um <i>layout</i> em grelha, esta secção, deverá exibir dois textos "Data Nascimento: e "Data Carta:"acompanhados por campos de data. Deverá também apresentar o texto "Código Postal:"seguido de uma caixa de texto. Estes campos só serão visíveis, se o tomador for individual e não for condutor habitual.</p> |

| | | |
|--------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RF03.5 | Dados Viatura | Nesta secção, deverão ser apresentados em grelha, da esquerda para a direita, dois <i>drop-downs</i> , um contendo as categorias e outro contendo os valores "Aluguer" e "Outro", precedidos dos textos "Categoria:" e "Natureza:" respetivamente. Por baixo deverá apresentar o texto "Matrícula Nova:" e dois <i>Radio Buttons</i> contendo, como resposta, "Sim" e "Não". De seguida deverá apresentar o texto "Matrícula:" seguido de um <i>dropdown</i> com as opções "Matrícula Nacional", "Importado" e "Matrícula Reboque", estes estarão acompanhados por uma caixa de texto. Por fim, nessa linha, deverá estar apresentado o texto "Data 1ª Matrícula:" seguido de um campo de data. Na última linha, irá apresentar os textos "Marca:", "Modelo:" e "Versão:" seguidos dos respetivos <i>drop-downs</i> , finalizando com um botão de <i>info</i> , com um ícone. |
|--------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Tabela 4.4: Requisitos funcionais da página de simulação de seguro.

| ID Re- quisito RF04.x | Nome do Re- quisito | Descrição do Requisito |
|-----------------------------|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RF04.1 | Janela | Deverá ser semelhante à janela descrita em RF02.1, apresentando como conteúdo botões com os nomes "Gerir Base de Dados" e "Voltar". Estes estarão apresentados seguindo um <i>layout</i> vertical. |

Tabela 4.5: Requisitos funcionais da página de definições.

| ID Re- quisito RF05.x | Nome do Re- quisito | Descrição do Requisito |
|-----------------------------|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RF05.1 | Janela | Deverá ser semelhante à janela descrita em RF02.1, apresentando como conteúdo botões com os nomes "Adicionar entrada", "Editar entrada", "Remover entrada" e "Voltar" estes estarão apresentados seguindo um <i>layout</i> vertical. |

Tabela 4.6: Requisitos funcionais da página de gestão da BD.

4.2.2 Requisitos Não Funcionais

Este tipo de requisitos são responsáveis por informar acerca do desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas na aplicação.

A tabela 4.7 descreve alguns requisitos não funcionais encontrados.

| ID Requi- sito | Nome do Requi- sito | Descrição do Requisito |
|-------------------|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RNF01 | Tamanho | A aplicação não deverá ter mais do que 250 MB. |
| RNF02 | Consumo | A aplicação não deverá consumir mais do que 200 MB de RAM. |
| RNF03 | Horário de funci- onamento | A aplicação deve estar disponível todos os dias e a todas as horas, excepto quando esta ou os <i>websites</i> a que acede, estão em períodos de manutenção. |
| RNF04 | <i>Browser</i> | O acesso à aplicação necessita de um <i>browser</i> moderno, como: <i>Microsoft Edge</i> versão 84 ou superior; <i>Google Chrome</i> versão 84 ou superior, <i>Chromium</i> versão 84 ou superior; <i>Mozilla Firefox</i> versão 45 ou superior; <i>Apple Safari</i> versão 9 ou superior. |
| RNF05 | Sistema Opera- tivo | O sistema operativo deve ser indiferente após a devida instalação dos pacotes necessários. |

| | | |
|-------|-----------------|-----------------------------------------------------------------------------------------------------------|
| RNF06 | Dispositivo | Será necessária ligação à Internet para efetuar a automatização dos <i>websites</i> . |
| RNF07 | Usabilidade | A aplicação deverá ser intuitiva e de fácil uso, não necessitando de formação prévia para o seu manuseio. |
| RNF08 | Disponibilidade | Deve poder ser utilizado em computador. |

Tabela 4.7: Requisitos não funcionais da aplicação.

4.2.3 Requisitos de Domínio

Este tipo de requisitos trata de todo o tipo de informação relativa ao domínio da aplicação e dos seus processos. São específicos da área do cliente e como tal são de difícil especificação.

A tabela 4.8 descreve alguns requisitos de domínio encontrados.

| ID Requisito | Nome do Requisito | Descrição do Requisito |
|--------------|-------------------|--------------------------------------------------------------------------------|
| RD01 | Linguagem | Saber falar e escrever Português. |
| RD02 | Acesso à Internet | Saber conectar o computador à Internet. |
| RD04 | Instalação | Saber instalar os programas necessários para o bom funcionamento da aplicação. |
| RD03 | Funcionamento | O utilizador deverá saber executar uma aplicação no computador. |

Tabela 4.8: Requisitos de domínio da aplicação.

4.3 Casos de Uso

Nesta subsecção será apresentado o diagrama de casos de uso considerado durante o desenvolvimento da aplicação. Este diagrama representa uma possível utilização da aplicação por parte de um ator, demonstrando assim a interação do sistema com o utilizador.

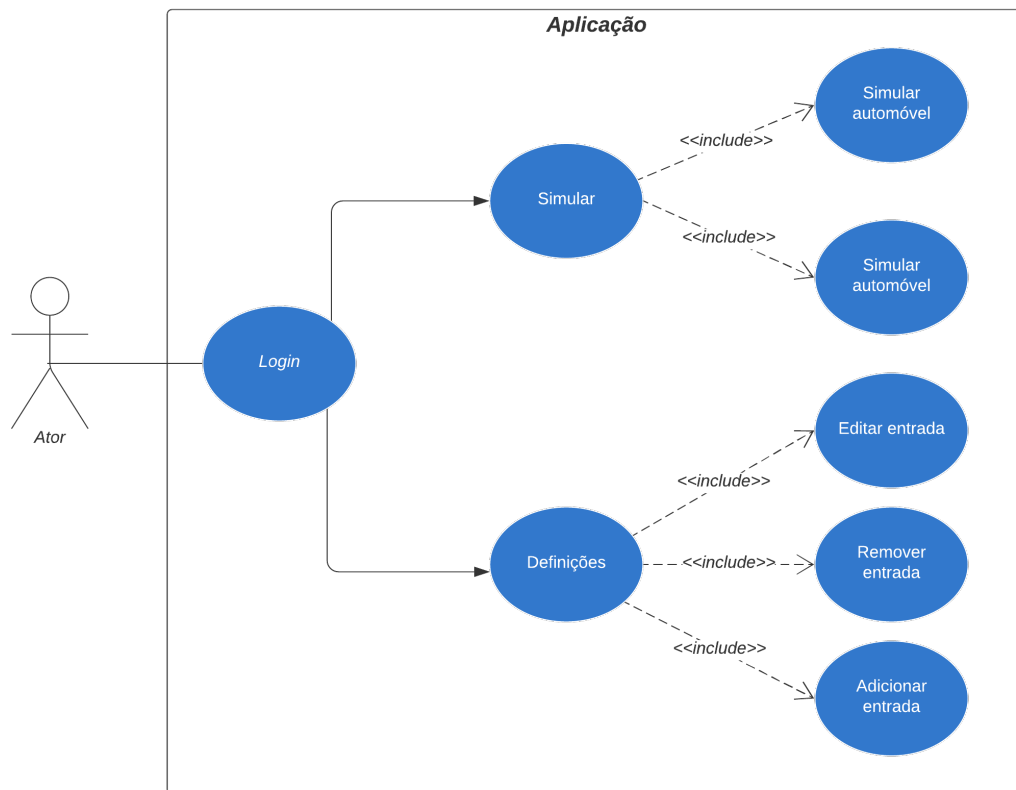


Figura 4.1: Casos de uso do utilizador.

Neste diagrama (figura 4.1) podemos observar as possíveis interações que um utilizador, com acesso à aplicação, pode ter. Esta interação passa pela possibilidade de fazer novas simulações, tanto de veículos de quatro rodas como de veículos de duas rodas, adicionar, remover ou editar entradas nas tabelas da BD da aplicação.

4.4 Conclusões

Com os requisitos e casos de uso encontrados, é possível idealizar o funcionamento da aplicação de forma eficiente, eficaz e segura.

O seguinte capítulo irá abordar a forma de implementação da aplicação.

Capítulo

5

Implementação e Testes

5.1 Introdução

Neste capítulo irão ser abordados os detalhes de implementação que sustentaram o desenvolvimento da aplicação, sendo que no capítulo 3 foram descritas as ferramentas necessárias para tal.

Numa fase inicial será apresentado o livro onde retirei os conceitos básicos do funcionamento e sintaxe do *PyQt*.

De seguida irá ser descrita a BD que foi necessária para o desenvolvimento da aplicação, bem como partes de código relevantes.

Irá também ser apresentado o manual de utilizador juntamente com as respetivas capturas de ecrã.

Por último serão demonstrados os testes realizados após a implementação da aplicação.

Este capítulo encontra-se dividido da seguinte maneira:

- Secção 5.2 - ***Learn PyQt*** - apresenta o livro que auxiliou a aprendizagem de conceitos básicos sobre *PyQt*.
- Secção 5.3 - **Base de Dados** - introdução à BD implementada na aplicação.
- Secção 5.3.1 - **Diagrama Entidade-Associação** - demonstração e explicação do modelo de dados utilizado.
- Secção 5.4 - **Implementação** - descrição pormenorizada do funcionamento da aplicação.
- Secção 5.4.1 - **Segurança** - demonstração e explicação pormenorizada do funcionamento da segurança implementada na aplicação.

- Secção 5.4.2 - **GUI** - descrição e explicação da GUI.
- Secção 5.4.3 - **Base de Dados e Comandos** - demonstração e explicação da implementação da BD.
- Secção 5.4.3.1 - **Criação da Base de Dados** - apresentação e explicação dos passos necessários para a criação da BD.
- Secção 5.4.3.2 - **Pedidos SQL** - apresentação e explicação da criação dos pedidos SQL.
- Secção 5.4.4 - **Automatização** - demonstração e explicação do funcionamento da automatização de um *website* de uma seguradora.
- Secção 5.5 - **Manual do Utilizador** - descrição pormenorizada do funcionamento da aplicação na perspectiva utilizador.
- Secção 5.5.1 - **Acesso** - forma de aceder à aplicação.
- Secção 5.5.2 - **Página Principal** - demonstra a página principal da aplicação.
- Secção 5.5.3 - **Simulação** - demonstração e explicação do processo da criação de uma simulação.
- Secção 5.5.4 - **Editar Entrada na BD** - demonstração e explicação da edição de valores na BD.
- Secção 5.5.5 - **Remover Entrada na BD** - demonstração e explicação da remoção de valores na BD.
- Secção 5.5.6 - **Adicionar Entrada na BD** - demonstração e explicação da adição de valores na BD.
- Secção 5.6 - **Testes** - introdução aos testes realizados na aplicação.
- Secção 5.6.1 - **Testes Unitários** - apresentação dos testes unitários realizados.
- Secção 5.6.2 - **Testes Manuais** - apresentação dos testes realizados manualmente, na implementação da aplicação.
- Secção 5.6.2.1 - **Janelas de Confirmação** - descrição das janelas de confirmação obtidas através dos testes manuais.
- Secção 5.7 - **Conclusões** - apresenta as principais conclusões do capítulo.

5.2 Learn PyQt

Após a escolha da linguagem de programação *Python*, foi necessário pesquisar formas de criar uma GUI sendo que para tal foi fundamental a procura de material que desse a conhecer melhor o *PyQt*. Nesse processo o *website Learn PyQt* foi um ponto crucial, pois deu-me acesso a um livro e a 4 horas de vídeos tutorial que o complementaram [14] [15].

Através destes foi-me possível adquirir conhecimento sobre sinais e eventos, *widgets*, *layouts*, caixas de diálogo, uso do modelo *Model-View-Controller* (MVC) e execução de tarefas de forma concorrente através de *multithreading*.

Todo este material foi muito útil, estando bem classificado devido à simplicidade e forma de explicar que o instrutor, *Martin Fitzpatrick*, demonstra nos seus vídeos. O *Martin Fitzpatrick* é um engenheiro de *software* que conta com mais de 8 anos de experiência a desenvolver aplicações recorrendo ao *PyQt*.

5.3 Base de Dados

A BD é, neste projeto, um elemento auxiliar ao utilizador, onde foi referida a sua importância em qualquer projeto no capítulo 2 na subsecção 2.5. Nesta encontra-se toda a informação referente aos possíveis produtos, categorias, marcas, modelos e versões de veículos.

5.3.1 Diagrama Entidade-Associação

Um Diagrama Entidade-Associação (DEA) é uma técnica de modelização de dados de uma BD nas áreas da engenharia de *software* e sistemas de informação. Para tal, apresenta a estrutura lógica de uma BD, incluindo as relações e restrições que determinam como os dados devem de ser guardados e acedidos.

Num DEA, definem-se três conceitos chave, sendo estes:

- **Entidade:** algo do mundo real que pode ser definido por um conjunto de atributos. A sua definição passa pela identificação dos elementos e de um conjunto de atributos comuns do mundo real que estamos a analisar;
- **Atributos:** propriedade ou característica de uma entidade;
- **Associação:** relacionamento de duas ou mais entidades.

Algumas vantagens deste modelo são a descoberta de problemas lógicos e de implementação, a sua construção faseada *Top-Down* e a facilidade de comunicação entre informáticos e utilizadores.

A figura 5.1 descreve a versão final do modelo de dados implementado na aplicação.

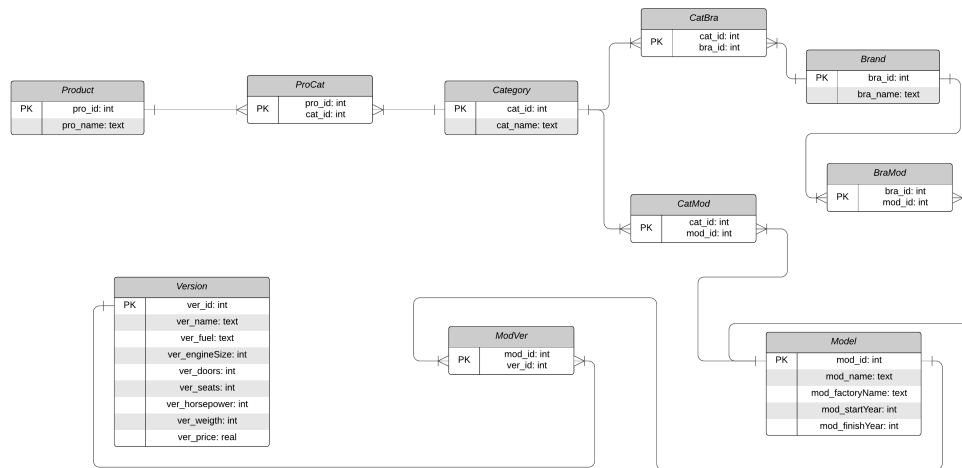


Figura 5.1: DEA implementado na BD.

5.4 Implementação

Nesta secção serão abordados, ao pormenor, os detalhes a implementação da aplicação. Para tal foi necessário, numa fase inicial, um levantamento dos parâmetros a padronizar, tendo sido utilizados diversos *websites* de seguradoras distintas.

5.4.1 Segurança

A aplicação apresenta vários níveis de segurança, não permitindo assim o acesso de terceiros à mesma. O nível mais baixo de segurança é a implementação de um *login* com **verificação de uma palavra-passe**, onde o utilizador terá que **introduzir a palavra-passe correta** num campo de texto que apresenta visualmente os caracteres como bolas, estando este processo representado na figura 5.2. O excerto de código abaixo (5.1) demonstra como é feita a verificação da palavra-passe. A variável global *hash* contém um valor pré-calculado do valor de *hash* da palavra-passe com um *salt*, também pré-calculado. Quando o utilizador preenche o campo de texto respetivo à

palavra-passe e clica no botão "Ok", como representado na figura 5.2, a função *Login* é chamada. A função tem acesso à variável global *hash* e compara-a com o output da função *KeyGenerator*, que por sua vez recebe como *input* o valor de *hash*, *Secure Hash Algorithm* (SHA)-256, do texto contido no *passwordField*. Se for verificada a igualdade, a aplicação irá prosseguir, caso contrário o utilizador perderá uma das três tentativas consecutivas, sendo o texto contido na variável *passwordField* substituído pelo texto "X tentativas restantes...", onde X é o número de tentativas restantes, como representado na figura 5.3. Esgotando as três tentativas consecutivas a aplicação ir-se-á fechar.

```
# Valor pre-calculado de hash da palavra-passe
hash = b'jd3bxQcHGO4F5ksoqFpsLEpbOL_d6wgTb_5RPm6dhJs='

# Verifica se a palavra introduzida e a correta, o utilizador tera
numberTries
def Login(self):
    global hash
    # KeyGenerator(modulo, password)
    if KeyGenerator(1, DigestSHA256(self.passwordField.text())) == hash:
        self.accept
    elif self.numberTries < 2 :
        self.numberTries = self.numberTries + 1
        self.passwordField.setText('')
        self.passwordField.setPlaceholderText('%d tentativas restantes
        ... ' % (3 - self.numberTries))
    else:
        sys.exit()
```

Excerto de Código 5.1: Função que efetua o *login*.

Outro nível de segurança, foi a **cifra dos acessos** aos *websites* das seguradoras, onde para tal a cifra e decifra dos mesmos ficou ao encargo da classe *Fernet*. O excerto de código abaixo (5.2) representa as funções de cifra e decifra de um ficheiro, onde para tal as funções irão receber o *path* do ficheiro a manipular, assim como a chave de cifra simétrica do mesmo. A chave de cifra será o *output* da função *KeyGenerator*.

```
# Cifra um ficheiro
def EncryptFile(filePath: str, key: bytes):
    with open(filePath, 'rb') as f:
        data = f.read()
    f.close()
    os.remove(filePath)
    fernet = Fernet(key)
    encrypted = fernet.encrypt(data)
    new_filePath = filePath + '.enc'
```

```

print(new_filePath)

with open(new_filePath, 'wb') as f:
    f.write(encrypted)
f.close()

# Decifra um ficheiro
def DecryptFile(filePath: str, key):
    with open(filePath, 'rb') as f:
        data = f.read()
    f.close()
    fernet = Fernet(key)
    decrypted = fernet.decrypt(data)

    return decrypted.decode()

```

Excerto de Código 5.2: Funções de cifra e decifra.

Por fim, a função *KeyGenerator*, presente no excerto de código 5.3, recebe o modo de operação, sendo 1 para quando está a **tratar do acesso à aplicação** e 2 para quando está a **gerar a chave de cifra simétrica** para a **cifra e decifra** do ficheiro que contém os acessos. A variável *password* será o valor de *hash* da palavra-passe válida introduzida pelo utilizador. Independentemente do modo a utilizar, a função recorre ao algoritmo PBKDF2 (*Password-Based Key Derivation 2*) que irá utilizar o algoritmo de *hash* SHA-256 e um *salt* para obter um valor de *hash*. Por fim esse valor será passado para a variável *key*.

```

def KeyGenerator(mode: int, password: bytes):
    # Trata da chave para a aplicacao
    if mode == 1:
        salt = b'\x0c<!\xdd\xc7%\x89\xb0\xd7\x88\x01D\x81U\xa2\x03'
        kdf = PBKDF2HMAC(algorithm=hashes.SHA256(), length=32, salt=salt,
            iterations=100000, backend=default_backend())

        key = base64.urlsafe_b64encode(kdf.derive(password))
    # Trata da chave de cifra e decifra do ficheiro de acessos
    elif mode == 2:
        salt = b'\x15\r\xcf\xed9\r7\xf53\x85\xe90\x91\x9cu\xa4'
        kdf = PBKDF2HMAC(algorithm=hashes.SHA256(), length=32, salt=salt,
            iterations=100000, backend=default_backend())

        key = base64.urlsafe_b64encode(kdf.derive(password))
    else:
        sys.exit()
    return key

```

Excerto de Código 5.3: Função que gera chaves de cifra.

5.4.2 GUI

O desenvolvimento da GUI utilizou as classes do *PyQt*, mais especificamente as classes ***QtWidgets*** e ***QtGui***.

A tabela 5.1 apresenta os *widgets* utilizados, bem como uma pequena descrição do seu propósito.

| Nome | Finalidade |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>QCheckBox</i> | Mostra uma caixa selecionável podendo conter um texto ao seu lado. |
| <i>QComboBox</i> | Apresenta uma lista no modo <i>dropdown</i> . |
| <i>QDateEdit</i> | Mostra um campo de introdução de texto numérico, permitindo as seguintes disposições da data: dd-mm-aaaa, aaaa-mm-dd, entre outras, onde "dd" representa o dia, "mm" o mês e "aaaa" o ano. |
| <i>QLabel</i> | Apresenta um texto não interativo. Esse texto será passado como <i>String</i> quando se cria um objeto. |
| <i>QLineEdit</i> | Semelhante ao <i>QDateEdit</i> , mas permite a introdução de caracteres alfanuméricos. |
| <i>QPushButton</i> | Mostra um botão que apresenta um texto passado como <i>String</i> . |
| <i>QScrollArea</i> | Possibilita a introdução de <i>scroll</i> vertical e horizontal. |
| <i>QSpinBox</i> | Permite a introdução, por parte do utilizador, de valores inteiros. |
| <i>QStackedWidget</i> | Possibilita a criação de vários <i>widgets</i> filho que estarão sobrepostos entre si. |
| <i>QWidget</i> | Componente básica da GUI onde o utilizador pode interagir com. Esta pode conter várias outros <i>widgets</i> dentro de si. |

Tabela 5.1: *Widgets* utilizados da classe *QWidget*.

A gestão organizacional dos *widgets* foi feita através de *layouts*. A tabela 5.2 apresenta os *layouts* utilizados na aplicação bem como uma pequena descrição acerca da sua finalidade.

| Nome | Finalidade |
|--------------------|--------------------------------------------------------------|
| <i>QHBoxLayout</i> | Possibilita a organização de <i>widgets</i> horizontalmente. |
| <i>QVBoxLayout</i> | Permite a disposição vertical dos <i>widgets</i> . |

| | |
|--------------------|------------------------------------------------------------------------------------------|
| <i>QGridLayout</i> | Concede a disposição dos <i>widgets</i> em grelha, ou seja, é a junção das duas de cima. |
| <i>QFormLayout</i> | Possibilita a organização de <i>widgets</i> em formulário. |

Tabela 5.2: *Layouts* utilizados na GUI.

Por fim, para a apresentação dos *widgets* referidos acima, foi necessário implementar janelas (*windows*). A tabela 5.3 apresenta os diferentes géneros de janelas utilizadas na GUI.

| Nome | Finalidade |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>QMainWindow</i> | Cria a janela principal da aplicação, dando asas à criação do conteúdo da GUI. Esta pode ser povoada por uma barra de menu, <i>toolbars</i> , <i>widgets</i> centrais e uma barra de estado. |
| <i>QDialog</i> | Permite a criação de pequenas janelas, comumente utilizadas para definições, preferências ou funções fora da principal janela da aplicação. Estas janelas ficam sobre a janela principal bloqueando o avanço da mesma. |
| <i>QMessageBox</i> | Apresenta um texto a informar o utilizador de uma situação, podendo conter um texto informativo e um texto detalhado onde explica o porquê da sua ativação. |

Tabela 5.3: *Windows* utilizadas na implementação da GUI.

No excerto de código abaixo (5.4), pode-se verificar a criação de uma janela que recebe, quando iniciada, os valores de largura, comprimento e o nome a apresentar. As funções base para a **criação da janela** consistem na *setGeometry*, a qual irá permitir aplicar os valores de largura e comprimento, a função *setWindowIcon* que irá colocar o **ícone** na barra de ferramentas do SO do utilizador e a *setWindowTitle* que coloca o **nome** na janela.

As páginas presentes na aplicação devem-se ao uso de um *widget* central do formato *stacked*, que permite assim a sobreposição de diversos *widgets* secundários, possibilitando a visualização simultânea de apenas um *widget* secundário. A variável *stacked* aplica esse género de *widget*, sendo que irá conter três *widgets* secundários sobrepostos: **página principal** da aplicação, **simulação de seguro** de viatura e **definições**.

Existindo a possibilidade de alterar as dimensões da janela, a variável *scroll* aplica uma área de *scroll*, permitindo assim ao utilizador visualizar *widgets* que possam ter ficado escondidos pelo redimensionamento da janela.

De seguida são criados os *widgets* secundários, finalizando com a sua adição ao *widget stacked* através do uso da função *addWidget*. Esta ordem de adição será importante para saber qual o índice de cada um dos *widgets* secundários.

Por fim a função *setCentralWidget* irá colocar o *widget scroll* como *widget* central na janela.

```
# imports necessarios
from PyQt5.QtWidgets import QApplication, QWidget, QStackedWidget,
    QScrollArea, QVBoxLayout,
from PyQt5.QtGui import QIcon
# GUI
class GUI(QMainWindow):
    def __init__(self, width, heigth, windowTitle):
        super(GUI, self).__init__()
        self.setGeometry(800, 600, width, heigth)
        iconPath = uiPath + path + 'Imagens' + path + 'Universalis-RM-
            icon.png'
        self.setWindowIcon(QIcon(iconPath))
        self.setWindowTitle(windowTitle)

        # Fonte dos cabecalhos
        self.headline = QFont("Arial", 14, QFont.Bold)

        # Principal stacked (Homepage-Vehicle-Health)
        self.stacked = QStackedWidget()

        # Scrollable
        self.scroll = QScrollArea()
        self.scroll.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)
        self.scroll.setWidgetResizable(True)
        self.scroll.setWidget(self.stacked)

        self.homepageWidgets = QWidget()
        self.homepageLayout = QVBoxLayout()
        self.HomepageUI()

        self.vehicleWidgets = QWidget()
        self.vehicleLayout = QVBoxLayout()
        self.VehicleUI()

        self.healthWidgets = QWidget()
        self.healthLayout = QVBoxLayout()
        self.HealthUI()

        self.settingsStacked = QStackedWidget()
        self.SettingsUI()
```

```
self.stacked.addWidget(self.homepageWidgets)
self.stacked.addWidget(self.vehicleWidgets)
self.stacked.addWidget(self.healthWidgets)
self.stacked.addWidget(self.settingsStacked)
self.setCentralWidget(self.scroll)
```

Excerto de Código 5.4: Classe GUI.

5.4.3 Base de Dados e Comandos

A criação da BD e os pedidos SQL ficaram encarregues ao *Python*, através do seu módulo *sqlite3*. É possível observar-se, na subsecção 5.4.3.1, as funções que permitiram a criação da mesma. Por sua vez, na subsecção 5.4.3.2, pode-se visualizar a criação de funções que permitiram a execução de pedidos (*queries*) SQL.

5.4.3.1 Criação da Base de Dados

O excerto de código 5.5 permite visualizar a **criação da BD** através da função **sqlite3.connect**. Após a criação da mesma, é necessário a implementação de um objeto do tipo **Cursor**, o qual nos irá permitir percorrer as linhas de um conjunto de resultados. Em seguida podemos então criar as tabelas, utilizando para tal as funções do *Cursor*, mais especificamente a função *execute*, a qual possibilita a introdução de **comandos SQL no Python**.

```
# imports necessarios
import sqlite3
import os

# Diretoria atual
workingDirectory = os.getcwd()
dbPath = os.path.join(workingDirectory, 'project.db')

conn = sqlite3.connect(dbPath)
c = conn.cursor()

# Cria as tabelas
def CreateDB():
    global c
    # Tabelas principais
    c.execute('''CREATE TABLE Product
                (pro_id INTEGER PRIMARY KEY NOT NULL,
                 pro_name TEXT NOT NULL)''')
    c.execute('''CREATE TABLE Category
```

```

        (cat_id INTEGER PRIMARY KEY NOT NULL,
         cat_name TEXT NOT NULL) '''
# Tabela intermedia
c.execute('''CREATE TABLE ProCat
         (product_id INTEGER NOT NULL,
          category_id INTEGER NOT NULL,
          PRIMARY KEY(product_id, category_id))''')
```

Excerto de Código 5.5: Criação da BD.

Tendo sido as tabelas criadas, podemos então povoar as mesmas, sendo possível observar no excerto de código 5.6, o preenchimento da tabela *Product*. Para que este preenchimento seja gravado, é necessário a execução da função **commit**, a qual irá atualizar o conteúdo das tabelas.

```

# Preenche a tabela
def FillsProductDB():
    global c, conn
    c.execute("INSERT INTO Product(pro_name) VALUES ('AUTOM – AUTOMVEL
              ')")
    c.execute("INSERT INTO Product(pro_name) VALUES ('AUT2R – AUTOMVEL
              2 RODAS')")
    conn.commit()
```

Excerto de Código 5.6: Preenchimento de uma tabela da BD.

5.4.3.2 Pedidos SQL

As *queries* SQL ficaram por sua vez encarregues a funções que fazem a comunicação à BD. No excerto de código 5.7, é possível observar-se, através da função **Query**, a criação de um pedido *SELECT*. Para tal, é **necessário receber** o parâmetro a retornar, o nome da tabela à qual irá fazer a *query*, o parâmetro a restringir e o(s) valor(es) dessa restrição. Devido ao facto desta função ser utilizado em diversas partes deste projeto, tornou-se necessário a implementação de uma variável, opção, quando é efetuada a chamada a esta.

```

# Faz uma query a BD e retorna a mesma
def Query(headerOutput, tableName, restrictor, headerInput, operator,
          option):
    global c
    ret = []
    # Remove a 1a posicao
    restrictor.pop(0)
    if option == 1:
        query = 'SELECT ' + headerOutput + ' FROM ' + tableName
    else:
```

```

if headerOutput != 'pro_name' and len(restrictor) == 0:
    return None
if len(restrictor) == 0:
    query = 'SELECT ' + headerOutput + ' FROM ' + tableName
elif len(restrictor) == 1:
    query = 'SELECT ' + headerOutput + ' FROM ' + tableName + \
        ' WHERE ' + headerInput + ' = "' + str(restrictor[0]) + \
        '"'
else:
    query = 'SELECT ' + headerOutput + ' FROM ' + tableName + ' \
        WHERE '
    # Gets the last value of the restrictor
    lastValuePox = len(restrictor)-1
    for value in range(len(restrictor)):
        if value == lastValuePox:
            query = query + headerInput + ' = "' + \
                str(restrictor[value]) + '"'
        else:
            query = query + headerInput + ' = "' + \
                str(restrictor[value]) + '" ' + operator + ' '
if headerOutput == '*':
    for row in c.execute(query):
        # 0 -> id, 1 -> name, 2-> ....
        ret.append(row[1:])
else:
    for row in c.execute(query):
        ret.append(row[0])
return ret

```

Excerto de Código 5.7: Pedido à BD.

A pesquisa nas tabelas intermédias ficou ao encargo da função **Search**, representada pelo excerto de código abaixo (5.8, a qual irá receber, quando chamada, o valor de *ID* da entrada na BD e o seu respetivo nome.

```

def Search(index, name_id):
    global c
    if name_id == 'product_id':
        restrictor = [ ' ', str(index) ]
        return (Query('category_id', 'ProCat', restrictor, name_id, ' ',
            0))
    elif name_id == 'category_id':
        restrictor = [ ' ', str(index) ]
        return (Query('brand_id', 'CatBra', restrictor, name_id, ' ', 0))
    elif name_id == 'brand_id':
        restrictor = [ ' ', str(index) ]
        return (Query('model_id', 'BraMod ', restrictor, name_id, ' ', 0)
        )
    elif name_id == 'model_id':

```



```

    restrictor = [ ' ', str(index)]
    return (Query('version_id', 'ModVer', restrictor, name_id, ' ',
0))
# Erro
else:
    return -3

```

Excerto de Código 5.8: Pesquisa nas tabelas intermédias da BD.

Por sua vez, a adição, edição e remoção de valores das principais tabelas da BD ficaram encarregues às funções **Add**, **Edit** e **Remove** respetivamente. Podemos visualizar no excerto de código 5.9, a formulação de uma *query* com o intuito de adicionar um novo parâmetro a uma tabela da BD, sendo que para tal, esta necessita de receber o nome da tabela e os respetivos valores a adicionar. Por sua vez, no excerto de código 5.10, pode-se observar a função *Edit*, a qual necessita de receber o nome da tabela da BD, os valores a alterar e o respetivo *ID* da entrada. Por fim, no excerto de código 5.11, a função *Delete()* necessita de receber, aquando chamada, o nome da tabela da BD e o *ID* da entrada a remover.

```

def Add(tableName, inputs):
    if tableName == 'Category' and len(inputs) == 1:
        query = 'INSERT INTO Category(cat_name) VALUES (?);'
    elif tableName == 'Brand' and len(inputs) == 1:
        query = 'INSERT INTO Brand(bra_name) VALUES (?);'
    elif tableName == 'Model' and len(inputs) == 4:
        query = 'INSERT INTO Model(mod_name, mod_factoryName,
        mod_startYear, mod_finishYear) VALUES(?,?,?,?);'
    elif tableName == 'Version' and len(inputs) == 8:
        query = 'INSERT INTO Version(ver_name, ver_fuel, ver_engineSize,
        ver_doors, ver_seats, ver_horsepower, ver_weighth, ver_price
        ) VALUES(?,?,?,?,?,?,?,?);'
    else:
        return -1
    c.execute(query, inputs)
    conn.commit()

```

Excerto de Código 5.9: Adição de um valor à BD.

```

def Edit(tableName, inputs, id):
    if tableName == 'Category' and len(inputs) == 1:
        query = 'UPDATE Category SET cat_name = ? WHERE cat_id = ' + str
(id) + ' ';
    elif tableName == 'Brand' and len(inputs) == 1:
        query = 'UPDATE Brand SET bra_name = ? WHERE bra_id = ' + str(id
) + ' ';
    elif tableName == 'Model' and len(inputs) == 4:

```

```

        query = 'UPDATE Model SET mod_name = ?, mod_factoryName = ?,
                mod_startYear = ?, mod_finishYear = ? WHERE mod_id = ' + str
                (id) + ';'
    elif tableName == 'Version' and len(inputs) == 8:
        query = 'UPDATE Version SET ver_name = ?, ver_fuel = ?,
                ver_engineSize = ?, ver_doors = ?, ver_seats = ?,
                ver_horsepower = ?, ver_weighth = ?, ver_price = ? WHERE
                ver_id = ' + str(id) + ';'
    else:
        return -1
    c.execute(query, inputs)
    conn.commit()

```

Excerto de Código 5.10: Edição de um valor da BD.

```

def Delete(tableName, id):
    if tableName == 'Category':
        query = 'Delete FROM Category WHERE cat_id = ?;'
    elif tableName == 'Brand':
        query = 'Delete FROM Brand WHERE bra_id = ?;'
    elif tableName == 'Model':
        query = 'Delete FROM Model WHERE mod_id = ?;'
    elif tableName == 'Version':
        query = 'Delete FROM Version WHERE ver_id = ?;'
    else:
        return -1
    c.execute(query, (id,))
    conn.commit()

```

Excerto de Código 5.11: Remoção de um valor da BD.

Devido à implementação das funções acima retratadas, mais concretamente as funções **Search** e **Query**, todos os pedidos feitos pela GUI podem ser representados pelo excerto de código abaixo (5.12). Nele é possível observar-se, numa primeira fase, o uso da função **Query**, a qual permite a procura do *ID* da categoria atualmente seleccionada. De seguida, será pesquisado na tabela intermédia respetiva (*CatBra*), todos os *IDs* de marcas possíveis. Por fim, e através do uso da função **Query**, serão adicionados, à variável *vehicleBrandField*, os nomes das possíveis marcas.

```

def QueryBrands(self):
    if (self.vehicleCategoryField.currentIndex() != 0):
        self.vehicleBrandField.clear()
        self.vehicleModelField.clear()
        self.vehicleVersionField.clear()
        self.categoryID = Query('cat_id', 'Category', ['', self.
            vehicleCategoryField.currentText()], 'cat_name', '', 0)

```

```
brandsIDFromCategory = Search(self.categoryID[1], '
    category_id')
self.vehicleBrandField.addItem(Query('bra_name', 'Brand',
    brandsIDFromCategory, 'bra_id', 'OR', 0))
```

Excerto de Código 5.12: Pedido à BD feito pela GUI.

5.4.4 Automação

Através da utilização do **Selenium** foi possível automatizar os *websites* de diversas seguradoras, sendo necessário para tal a identificação dos parâmetros obrigatórios para que assim a sua automatização seja bem sucedida. Inicialmente, e devido à existência de *iframes* em certos *websites*, a localização dos parâmetros não estava a ser concluída com sucesso. Pode-se observar no excerto de código 5.13 como foi culminado esse problema.

O excerto de código 5.14 apresenta a função *execute_script*, a qual executa um comando em *JavaScript* para que o *browser* abra uma nova aba, sendo necessário a mudança da aba automatizada pelo *Selenium* com a função *switch_to_window*.

A localização dos parâmetros a serem automatizados faz-se através do *XPath* dos mesmos, recorrendo à função *find_element_by_xpath*. Após a localização com sucesso dos parâmetros serão passados os valores de acesso à página pela função *send_keys* seguida do clique no botão que faz submissão com a função *click*.

```
iframeElement = WebDriverWait(driver, 10).until(
    lambda driver: driver.find_element_by_xpath(iframeXPath))
driver.switch_to.frame(iframeElement)
```

Excerto de Código 5.13: Mudança para o correto *iframe*.

```
def Caravela(driver, user: str, password: str, simulationData,
    windowHandler: int):
    driver.execute_script(
        "window.open('http://egis.caravelaseguros.pt/eGIS/logon.jsp');")
    driver.switch_to_window(driver.window_handles[windowHandler])

    userFieldXPath = '/html/body/div/table/tbody/tr/td[2]/form/table/
        tbody/tr[3]/td/input'
    passwordFieldXPath = '/html/body/div/table/tbody/tr/td[2]/form/table
        /tbody/tr[5]/td/input'
    submitFieldXPath = '/html/body/div/table/tbody/tr/td[2]/form/table/
        tbody/tr[6]/td/input'
```

```
userElement = WebDriverWait(driver, 10).until(  
    lambda driver: driver.find_element_by_xpath(userFieldXPath))  
passwordElement = WebDriverWait(driver, 10).until(  
    lambda driver: driver.find_element_by_xpath(passwordFieldXPath))  
submitElement = WebDriverWait(driver, 10).until(  
    lambda driver: driver.find_element_by_xpath(submitFieldXPath))  
  
# Envia o user e password para a pagina  
userElement.send_keys(user)  
passwordElement.send_keys(password)  
submitElement.click()
```

Excerto de Código 5.14: Automatização do *login* do *website* Caravela.

5.5 Manual do Utilizador

Nesta secção irá ser abordado o funcionamento da aplicação do ponto de vista do utilizador. O seu funcionamento é bastante simples e intuitivo, não exibindo grande dificuldade para um utilizador habituado a manusear um computador.

Irão ser demonstrados os procedimentos de *login*, como é efetuada a simulação e introdução de dados e por fim a edição e remoção das entradas da BD da aplicação.

5.5.1 Acesso

Para um utilizador, basta introduzir a respetiva palavra-passe de acesso à aplicação seguida do clique no botão com o texto "OK", como representados na figura 5.2. Se a palavra-passe introduzida não for a correta, o utilizador terá mais duas tentativas, como demonstrado na figura 5.3.

A introdução de três palavras-passe consecutivas incorretas, fará com que a aplicação não prossiga, sendo encerrada de forma automática.

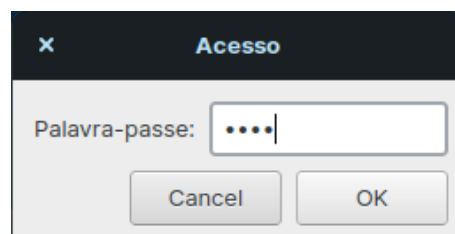
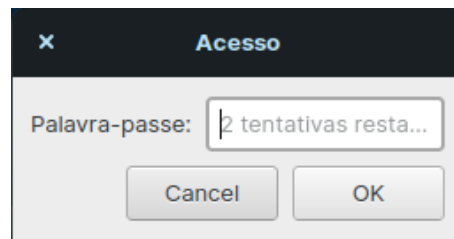


Figura 5.2: Preenchimento da palavra-passe.

Figura 5.3: Janela de *login*.

5.5.2 Página Principal

Após a introdução da palavra-passe válida, o utilizador terá acesso à página principal da aplicação, representada na figura 5.4, podendo iniciar uma simulação de seguro de viatura ou gerir os dados introduzidos na BD, se assim o desejar.

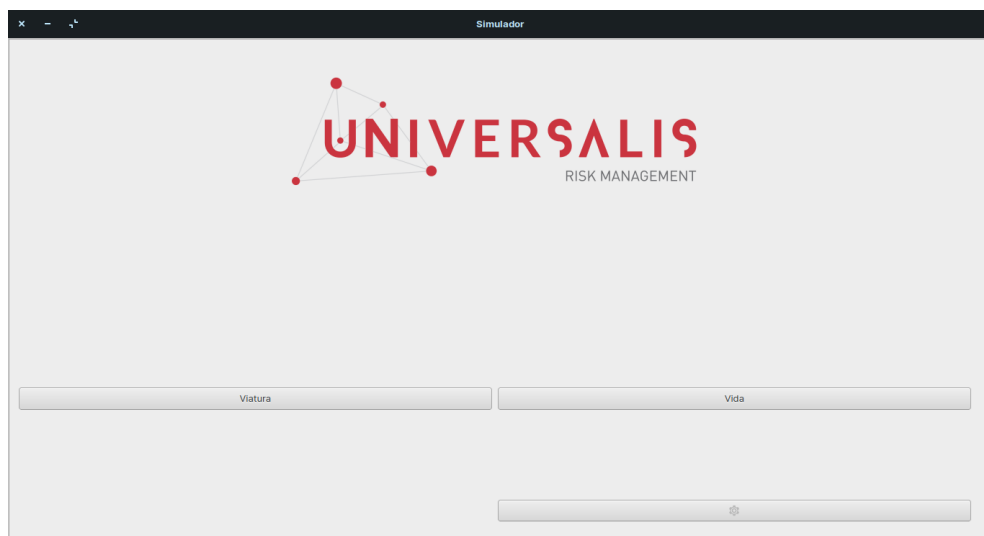


Figura 5.4: Página principal da aplicação.

5.5.3 Simulação

O utilizador terá a possibilidade de efetuar simulações de seguro de viatura. Para tal terá que ter acesso à aplicação, efectuando o respetivo *login* como descrito no capítulo 5 na subsecção 5.5.1, obtendo por conseguinte acesso à página principal da aplicação, como demonstrado na subsecção anterior. Após ter acesso à página principal da aplicação (figura 5.4) deverá cli-

car no botão com o texto "Viatura", sendo redirecionado para uma janela que conterá os dados necessários para uma simulação de seguro de viatura, representados nas figuras 5.5 e 5.6, onde o seu preenchimento na totalidade não será obrigatório, mas aconselhável para o bom funcionamento da aplicação.

Para finalizar, o utilizador terá que clicar no botão com o texto "Simular", o qual fará com que o navegador abra e preencha de forma automatizada os *websites* a que aplicação tem acesso.

The screenshot displays a web application titled "Simulador". It is organized into several sections with form fields:

- Dados Seguro:** Includes a "Produto" dropdown menu, a "Data Inicio Seguro:" date field (set to 12/08/2020), and "Tipo de Cliente:" radio buttons for "Individual" (selected) and "Colectivo".
- Dados Tomador:** Includes text input fields for "Morada:", "Código Postal:", "Telefone:", "NIF:", and "Email:". It also has a "Bónus:" checkbox, a "Data de Nascimento:" date field (01/01/2000), a "Data Carta:" date field (01/01/2000), and a "Tomador é condutor habitual:" checkbox.
- 1º Seguro Automóvel:** Includes radio buttons for "Sim" and "Não", a "Seguro Desde:" date field (01/01/2000), a "Data Último Sinistro:" date field (01/01/2000), and an "Nº Sinistros nos Últimos 2 anos:" dropdown menu (set to Zero).
- Dados Condutor:** Includes "Data Nascimento:" and "Data Carta:" date fields (both 01/01/2000) and a "Código Postal:" text input field.

Figura 5.5: Página de simulação.

The screenshot shows a web application window titled "Simulador". It contains several sections of input fields:

- Top Section:** Fields for "Código Postal:", "Telefone:", "NIF:", and "Email:". To the right, there is a "Data Carta:" dropdown menu set to "01/01/2000" and a checkbox for "Tomador é condutor habitual:".
- Insurance Section:** A checkbox for "Bónus:" and a label "1º Seguro Automóvel:". Below this, there are radio buttons for "Sim" and "Não".
- Insurance Dates and Claims:** Fields for "Seguro Desde:" (01/01/2000), "Data Último Sinistro:" (01/01/2000), and "Nº Sinistros nos Últimos 2 anos:" (Zero).
- Dados Condutor:** Fields for "Data Nascimento:" (01/01/2000), "Data Carta:" (01/01/2000), and "Código Postal:".
- Dados Viatura:** Fields for "Categoria:", "Natureza:" (Aluguer), "Matricula Nova:" (radio buttons for Sim and Não), "Matricula Nacional:" (dropdown), "Data 1ª Matricula:" (01/01/2000), "Marca:", "Modelo:", and "Versão:".

At the bottom of the form are two buttons: "Voltar" and "Simular".

Figura 5.6: Continuação da página de simulação.

5.5.4 Editar Entrada na BD

O utilizador terá a possibilidade de editar as entradas contidas nas tabelas da BD, sendo que para tal terá que ter acesso à aplicação, efectuando o respetivo *login* como descrito no capítulo 5 na subsecção 5.5.1, obtendo por conseguinte acesso à página principal da aplicação. Após ter acesso à página principal (figura 5.4) terá que clicar no botão com o ícone de definições, o qual irá redirecionar o utilizador para a página respetiva, como demonstrado na figura 5.7. Neste momento, o utilizador terá que clicar no botão com o texto "Gerir Base de Dados", onde irá visualizar a página de gestão da BD, como demonstrada na figura 5.8.

Por fim, o utilizador, terá que clicar no botão com o texto "Editar entrada", que abrirá uma janela *pop-up* como demonstrado na figura 5.9. A janela *pop-up* será constituída por dois campos *dropdown*, um campo de edição e um botão de confirmação com o texto "Editar".

Para efetuar uma edição, o utilizador terá que selecionar uma das possibilidades do primeiro campo *dropdown*, seguida da seleção de uma possibilidade do segundo campo *dropdown*. Consoante as respetivas seleções, o campo de edição será preenchido com os dados atuais da entrada, pelo que basta agora alterar os parâmetros pretendidos.

Assim sendo, basta clicar no botão "Editar". A confirmação de uma edição com sucesso será representada pela janela *pop-up* com a mensagem "Entrada editada na Base de Dados com sucesso", como podemos observar na figura

5.10.

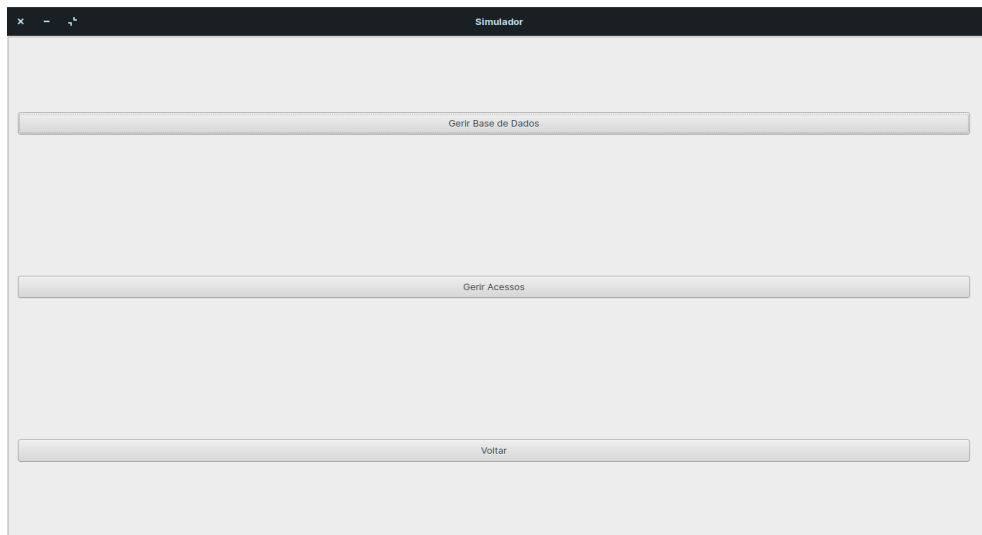


Figura 5.7: Página de Definições.

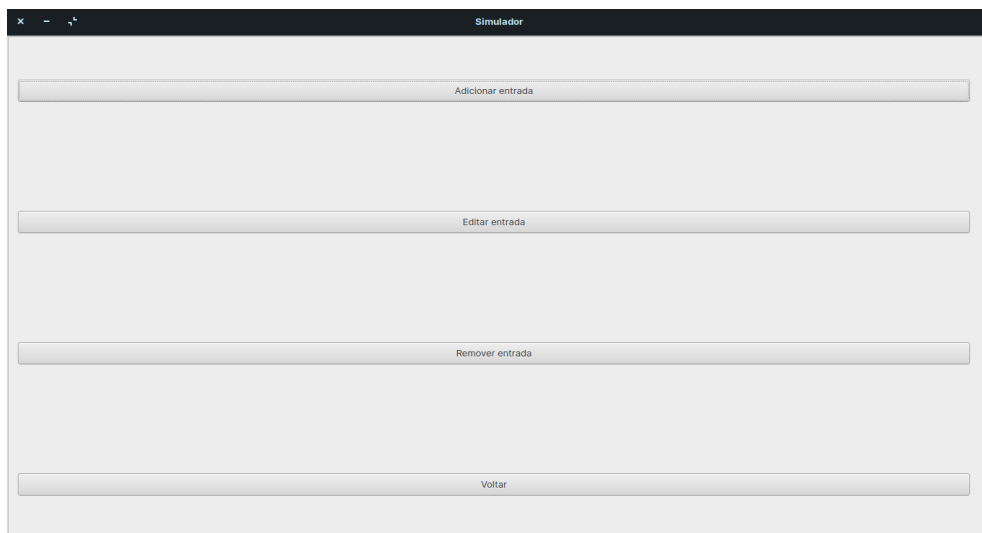


Figura 5.8: Página de Gestão da BD.

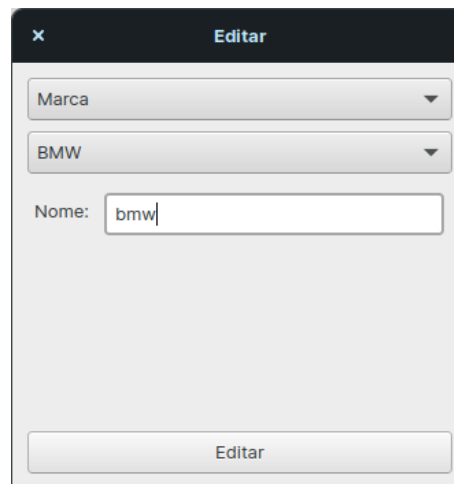
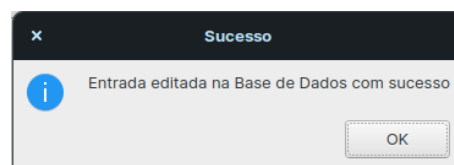
A screenshot of a web application window titled "Editar" with a close button (X) in the top-left corner. The window contains two dropdown menus: the first is labeled "Marca" and has "BMW" selected; the second is labeled "Nome:" and has "bmw" entered. At the bottom of the window is a button labeled "Editar".

Figura 5.9: Janela de edição da BD.

A screenshot of a small "Sucesso" (Success) pop-up window. It features a blue information icon (i) on the left and the text "Entrada editada na Base de Dados com sucesso" in the center. An "OK" button is located in the bottom-right corner.Figura 5.10: Janela *pop-up* de confirmação de edição.

5.5.5 Remover Entrada na BD

Para efetuar uma remoção de dados da BD, o utilizador terá que efetuar os passos semelhantes aos apresentados na subsecção 5.5.4, sendo que chegando à página de gestão da BD, representada pela figura 5.8, deverá clicar no botão com o texto "Remover entrada".

Após o clique, o utilizador visualizará uma janela *pop-up* com dois campos *dropdown* e um botão com o texto "Remover", semelhante à figura 5.11. Nessa janela o utilizador deverá selecionar uma das opções do primeiro campo *dropdown*, seguido da seleção de uma das opções do segundo campo *dropdown*.

Para finalizar a remoção, deverá clicar no botão com o texto "Remover", sendo que como confirmação da remoção, irá aparecer uma janela *pop-up* semelhante à da figura 5.12.

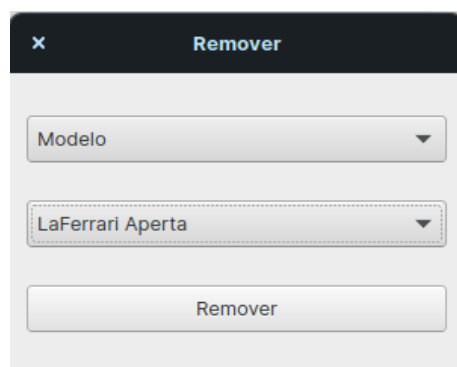


Figura 5.11: Janela de remoção da BD.

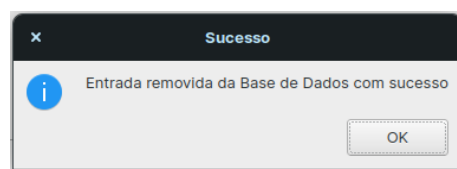


Figura 5.12: Janela *pop-up* de confirmação da remoção.

5.5.6 Adicionar Entrada na BD

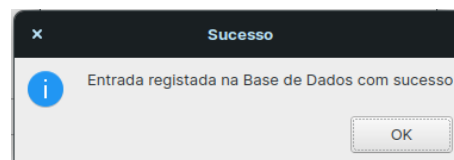
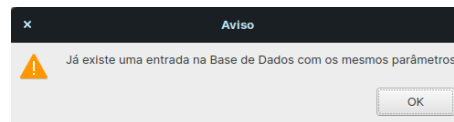
Para efetuar a adição de dados na BD, o utilizador terá que efetuar os passos semelhantes aos apresentados na subsecção 5.5.4, sendo que chegando à página de gestão da BD (figura 5.8) deverá clicar no botão com o texto "Adicionar entrada".

Após a seleção, o utilizador deverá visualizar uma janela *pop-up* semelhante à representada na figura 5.13. Nessa janela o utilizador deverá selecionar uma das possíveis opções no campo *dropdown* e preencher os campos abaixo do mesmo.

Para concluir a adição, deverá clicar no botão com o texto "Adicionar". Consoante haja sucesso ou insucesso na operação, o utilizador deverá visualizar uma janela *pop-up*. Esta será semelhante à representada na figura 5.14 como confirmação da operação efetuada com sucesso, ou a janela 5.15 como confirmação da operação efetuada com insucesso.

A janela de adição da BD, intitulada "Adicionar", possui um menu suspenso "Versão" no topo. Abaixo dele, há dois campos de texto para "Nome:" e "Lugares:". Seguem-se dois campos para "Combustível:" e "Cavalagem:". Depois, dois campos para "Cilindrada:" e "Peso:". Por fim, dois campos para "Portas:" e "Preço:". No rodapé da janela, há um botão "Adicionar".

Figura 5.13: Janela de adição da BD.

Figura 5.14: Janela *pop-up* de confirmação da remoção.Figura 5.15: Janela *pop-up* de insucesso na adição.

5.6 Testes

Foram realizados vários testes à aplicação, tanto ao nível do utilizador como ao nível da programação. Para este efeito foram realizados testes unitários com recurso ao módulo *unittest* do *Python* e testes manuais no decorrer da implementação do projeto.

5.6.1 Testes Unitários

A classe *unittest* é um módulo de testes para *Python*. Este módulo permite-nos formar testes, mais especificamente testar partes isoladas de código, de

modo a evitar falhas e erros quando estas partes comunicam entre si. Para tal suporta automatização de testes, agregação de testes em coleções e independência dos testes em relação à *framework* [17].

O uso deste módulo permite-nos testar os casos de limite, sendo que no excerto de código 5.15, é possível visualizar parte da função de teste. Com recurso ao *unittest* foi possível testar os casos para quando o utilizador introduz três vezes seguidas uma palavra-passe vazia, uma palavra-passe constituída só por caracteres ou só numérica. Sabendo que a aplicação só deverá prosseguir se a palavra-passe introduzida for a correta e que após a introdução de três palavras-passe incorretas seguidas fechar-se-á, é esperado que o resultado das operações a testar dê uma exceção, mais especificamente um *SystemExit*.

```
import unittest

class TestLogin(unittest.TestCase):
    def test_empty(self):
        global paths, hash
        app = QApplication([])
        log = LoginDialog(paths)
        log.passwordField.setText('')
        log.Login()
        log.Login()
        with self.assertRaises(SystemExit):
            log.Login()

    def test_only_chars(self):
        app = QApplication([])
        log = LoginDialog(paths)
        log.passwordField.setText('helloworld')
        log.Login()
        log.Login()
        with self.assertRaises(SystemExit):
            log.Login()

    def test_only_numbers(self):
        app = QApplication([])
        log = LoginDialog(paths)
        log.passwordField.setText('0123456789')
        log.Login()
        log.Login()
        with self.assertRaises(SystemExit):
            log.Login()

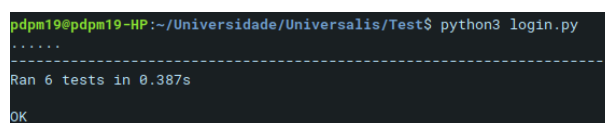
if __name__ == "__main__":
```

```
unittest.main()
```

Excerto de Código 5.15: Função de teste do *login*.

Na figura 5.16, pode-se constatar o resultado do teste para seis casos de limite, sendo estes:

- **Palavra-passe vazia**, simulando um caso em que o utilizar precisa só no botão com o texto "OK";
- **Palavra-passe constituída só com caracteres pertencentes ao alfabeto**, testando um caso em que o utilizador preenche o campo só com letras do alfabeto;
- **Palavra-passe constituída só por caracteres numéricos**, caso semelhante ao de cima, mas só com caracteres numéricos;
- **Palavra-passe constituída por caracteres alfanuméricos**, representa a junção dos dois casos acima, com a diferença de que a palavra-passe será constituída simultaneamente por ambos os géneros de caracteres;
- **Palavra-passe constituída por caracteres alfanuméricos e com caracteres especiais**, simulando o caso acima com a adição de caracteres especiais;
- **Palavra-passe correta**, representado o caso de uso em que o utilizador introduz a palavra-passe correta.



```
pdpm19@pdpm19-HP:~/Universidade/Universalis/Test$ python3 login.py
.....
-----
Ran 6 tests in 0.387s
OK
```

Figura 5.16: Execução do teste ao *login*.

5.6.2 Testes Manuais

Durante o desenvolvimento da aplicação foram realizados diversos testes manuais de forma a testar algumas funcionalidades.

A aplicação conta com várias janelas *pop-up* com mensagens de confirmação de sucesso ou insucesso, que irão aparecer quando o utilizador faz a gestão dos dados da BD.

5.6.2.1 Janelas de Confirmação

Estas janelas devem aparecer quando uma operação na BD da aplicação é terminada, independentemente do seu sucesso ou insucesso. Na figura 5.17, podemos observar o nome da categoria que será criada. A sua adição à tabela da BD será feita com sucesso, visto não existir nenhuma categoria com o mesmo nome (figura 5.18).


A janela de confirmação intitulada "Adicionar" possui um cabeçalho escuro com um ícone de fechar (X) e o título "Adicionar". O corpo da janela contém um menu suspenso rotulado "Categoria" com uma seta para baixo. Abaixo dele, há um campo de texto rotulado "Nome:" com o valor "Máquina Militar" inserido. No rodapé, há um botão "Adicionar".

Figura 5.17: Dados a adicionar na BD.

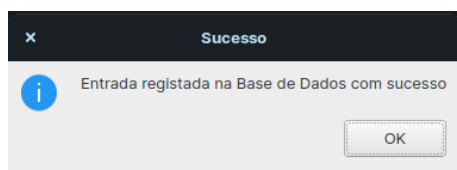
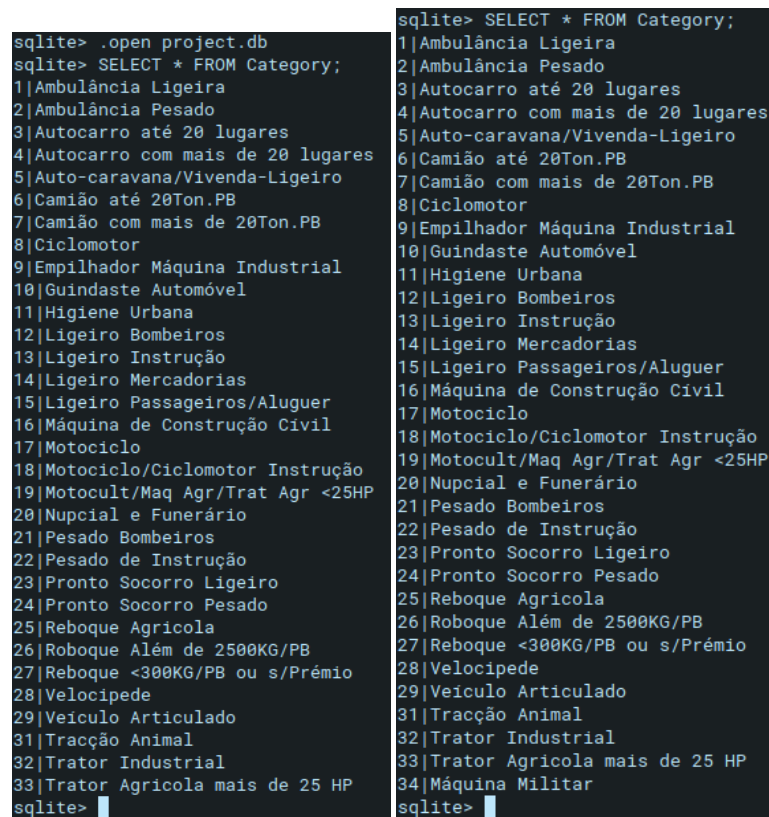
A janela de confirmação intitulada "Sucesso" possui um cabeçalho escuro com um ícone de fechar (X) e o título "Sucesso". O corpo da janela contém um ícone de informação (i) em um círculo azul, seguido pelo texto "Entrada registada na Base de Dados com sucesso". No rodapé, há um botão "OK".

Figura 5.18: Sucesso na adição de dados na BD.

A listagem das entradas contidas na tabela de categorias da BD representadas na figura 5.19, comprova o sucesso na adição da nova entrada.



```

sqlite> .open project.db
sqlite> SELECT * FROM Category;
1|Ambulância Ligeira
2|Ambulância Pesado
3|Autocarro até 20 lugares
4|Autocarro com mais de 20 lugares
5|Auto-caravana/Vivenda-Ligeiro
6|Camião até 20Ton.PB
7|Camião com mais de 20Ton.PB
8|Ciclomotor
9|Empilhador Máquina Industrial
10|Guindaste Automóvel
11|Higiene Urbana
12|Ligeiro Bombeiros
13|Ligeiro Instrução
14|Ligeiro Mercadorias
15|Ligeiro Passageiros/Aluguer
16|Máquina de Construção Civil
17|Motociclo
18|Motociclo/Ciclomotor Instrução
19|Motocult/Maq Agr/Trat Agr <25HP
20|Nupcial e Funerário
21|Pesado Bombeiros
22|Pesado de Instrução
23|Pronto Socorro Ligeiro
24|Pronto Socorro Pesado
25|Reboque Agrícola
26|Reboque Além de 2500KG/PB
27|Reboque <300KG/PB ou s/Prémio
28|Velocipede
29|Veiculo Articulado
31|Tracção Animal
32|Trator Industrial
33|Trator Agrícola mais de 25 HP
sqlite>

sqlite> SELECT * FROM Category;
1|Ambulância Ligeira
2|Ambulância Pesado
3|Autocarro até 20 lugares
4|Autocarro com mais de 20 lugares
5|Auto-caravana/Vivenda-Ligeiro
6|Camião até 20Ton.PB
7|Camião com mais de 20Ton.PB
8|Ciclomotor
9|Empilhador Máquina Industrial
10|Guindaste Automóvel
11|Higiene Urbana
12|Ligeiro Bombeiros
13|Ligeiro Instrução
14|Ligeiro Mercadorias
15|Ligeiro Passageiros/Aluguer
16|Máquina de Construção Civil
17|Motociclo
18|Motociclo/Ciclomotor Instrução
19|Motocult/Maq Agr/Trat Agr <25HP
20|Nupcial e Funerário
21|Pesado Bombeiros
22|Pesado de Instrução
23|Pronto Socorro Ligeiro
24|Pronto Socorro Pesado
25|Reboque Agrícola
26|Reboque Além de 2500KG/PB
27|Reboque <300KG/PB ou s/Prémio
28|Velocipede
29|Veiculo Articulado
31|Tracção Animal
32|Trator Industrial
33|Trator Agrícola mais de 25 HP
34|Máquina Militar
sqlite>

```

Figura 5.19: À esquerda encontram-se as categorias existentes na BD antes da adição, e à direita após a adição.

A tentativa de adição de uma marca já existente, neste caso "BMW" (figura 5.20), tem como resultado esperado a apresentação da janela com uma mensagem de erro (figura 5.21).

A screenshot of a dialog box titled "Adicionar" with a close button (X) in the top-left corner. It contains a dropdown menu labeled "Marca" and a text input field labeled "Nome:" with the text "BMW" entered. At the bottom, there is a button labeled "Adicionar".

Figura 5.20: Adição de uma marca já existente na BD.

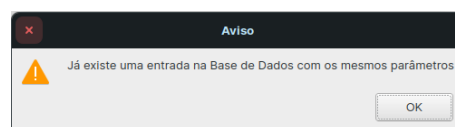
A screenshot of a dialog box titled "Aviso" with a close button (X) in the top-left corner. It features a yellow warning triangle icon and the text "Já existe uma entrada na Base de Dados com os mesmos parâmetros". An "OK" button is located at the bottom right.

Figura 5.21: Insucesso na adição de dados na BD.

A edição de valores de um entrada na BD também exibe uma janela *pop-up* com a confirmação da realização da tarefa com sucesso ou insucesso. Neste caso, na edição representada na figura 5.22, o resultado é a confirmação da edição com sucesso (figura 5.23).

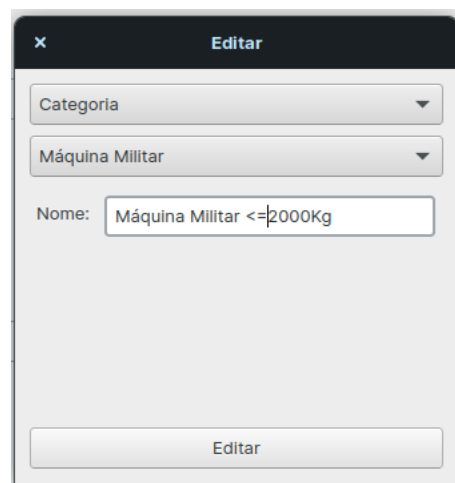
A screenshot of a dialog box titled "Editar" with a close button (X) in the top-left corner. It contains two dropdown menus: "Categoria" and "Máquina Militar". Below them is a text input field labeled "Nome:" with the text "Máquina Militar <=2000Kg" entered. At the bottom, there is a button labeled "Editar".

Figura 5.22: Janela de edição de categoria na BD.

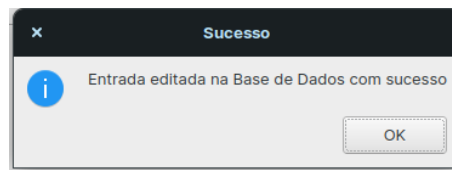


Figura 5.23: Sucesso na edição de dados na BD.

Pode-se observar a edição com sucesso representada na figura 5.24.

| | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> sqlite> SELECT * FROM Category; 1 Ambulância Ligeira 2 Ambulância Pesado 3 Autocarro até 20 lugares 4 Autocarro com mais de 20 lugares 5 Auto-caravana/Vivenda-Ligeiro 6 Camião até 20Ton.PB 7 Camião com mais de 20Ton.PB 8 Ciclomotor 9 Empilhador Máquina Industrial 10 Guindaste Automóvel 11 Higiene Urbana 12 Ligeiro Bombeiros 13 Ligeiro Instrução 14 Ligeiro Mercadorias 15 Ligeiro Passageiros/Aluguer 16 Máquina de Construção Civil 17 Motociclo 18 Motociclo/Ciclomotor Instrução 19 Motocult/Maq Agr/Trat Agr <25HP 20 Nupcial e Funerário 21 Pesado Bombeiros 22 Pesado de Instrução 23 Pronto Socorro Ligeiro 24 Pronto Socorro Pesado 25 Reboque Agrícola 26 Reboque Além de 2500KG/PB 27 Reboque <300KG/PB ou s/Prémio 28 Velocipede 29 Veículo Articulado 31 Tracção Animal 32 Trator Industrial 33 Trator Agrícola mais de 25 HP 34 Máquina Militar sqlite> </pre> | <pre> sqlite> SELECT * FROM Category; 1 Ambulância Ligeira 2 Ambulância Pesado 3 Autocarro até 20 lugares 4 Autocarro com mais de 20 lugares 5 Auto-caravana/Vivenda-Ligeiro 6 Camião até 20Ton.PB 7 Camião com mais de 20Ton.PB 8 Ciclomotor 9 Empilhador Máquina Industrial 10 Guindaste Automóvel 11 Higiene Urbana 12 Ligeiro Bombeiros 13 Ligeiro Instrução 14 Ligeiro Mercadorias 15 Ligeiro Passageiros/Aluguer 16 Máquina de Construção Civil 17 Motociclo 18 Motociclo/Ciclomotor Instrução 19 Motocult/Maq Agr/Trat Agr <25HP 20 Nupcial e Funerário 21 Pesado Bombeiros 22 Pesado de Instrução 23 Pronto Socorro Ligeiro 24 Pronto Socorro Pesado 25 Reboque Agrícola 26 Reboque Além de 2500KG/PB 27 Reboque <300KG/PB ou s/Prémio 28 Velocipede 29 Veículo Articulado 31 Tracção Animal 32 Trator Industrial 33 Trator Agrícola mais de 25 HP 34 Máquina Militar <=2000Kg sqlite> </pre> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Figura 5.24: À esquerda estão descritas as categorias existentes na BD antes da edição, e à direita após a edição.

Por fim, na remoção da entrada representada na figura 5.25, temos como resultado a apresentação da janela de confirmação do sucesso da operação (figura 5.26).



Figura 5.25: Janela de remoção de uma entrada na BD.

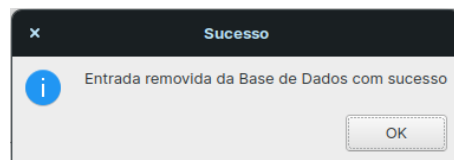


Figura 5.26: Janela de confirmação da remoção de uma entrada na BD.

5.7 Conclusões

Neste capítulo foi abordada a implementação da aplicação realizada ao longo do último semestre.

Inicialmente, e com a ajuda da plataforma *Learn PyQt* e do seu livro *Create GUI Applications with Python & Qt5*, foi-me possível acentuar, de forma rápida, a minha curva de aprendizagem no que toca ao desenvolvimento de GUIs com *Python* e *PyQt*.

O desenvolvimento da GUI foi um processo contínuo, de forma a esta ser o mais completa possível. Este foi o processo mais demorado e sujeito a alterações durante a sua implementação.

Devido à aplicação conter vários tópicos interessantes que em conjunto formam a sua totalidade, como por exemplo questões de segurança e automatização, estes foram explicados de forma aprofundada.

Foi também descrito um manual do utilizador, recorrendo a capturas de ecrã da sua execução, de forma a tornar o manuseamento da aplicação o mais *user-friendly* possível.

Por fim, foram elaborados testes à aplicação, tanto unitários como manuais, tendo estes sido descritos da melhor maneira possível.

Capítulo

6

Conclusões e Trabalho Futuro

Este capítulo tem como objetivo a apresentação das principais conclusões do projeto face aos objetivos inicialmente propostos. Serão referidas possíveis melhorias da aplicação e o trabalho futuro que poderá ser realizado.

6.1 Conclusões Principais

O projeto foi realizado durante o segundo semestre, o que condicionou um pouco a minha curva de aprendizagem, visto que não possuía qualquer conhecimento sobre o funcionamento do *PyQt*.

Na minha opinião, o projeto atendeu a quase todos os requisitos indicados na proposta inicial do projeto, incluindo o foco na automatização e otimização da introdução de dados em páginas *Web*.

Foi um trabalho que a nível académico, profissional e mesmo até pessoal me fez crescer imenso, tendo conseguido alcançar as metas por mim propostas.

É de referir que existe sem dúvida várias coisas a melhorar, e acredito que se houvesse mais tempo, o resultado seria ainda melhor.

Por fim, este projeto permitiu-me compreender a importância e crescimento que as áreas da informática, nomeadamente da automatização, tem vindo a ter.

A criação desta aplicação permite agilizar as operações de simulação de seguros nas corretoras de seguros, facilitando assim o seu processo e a redução da carga de trabalho ao seu utilizador.

6.2 Trabalho Futuro

Devido ao imenso tempo que tive a criar de raiz a GUI, não atingi a meta de gerir os acessos pela GUI, bem como o desenvolvimento da simulação de seguro de vida, sendo estas objetivos extras.

Como trabalho futuro, seria interessante começar por implementar a gestão de acessos e simulação de seguro de vida, referidos anteriormente e aplicar *threads* na parte de automatização da inserção de dados no *browser*.

Num futuro seria também interessante colocar a aplicação num servidor, sendo esta utilizada pelos clientes. Estes teriam direito também a fazer a sua simulação. Esta autonomia por parte do cliente iria permitir, ao corretor de seguros, uma diminuição extra da sua carga de trabalho.

Bibliografia

- [1] About SQLite. [Online] <https://www.sqlite.org/about.html>. Último acesso a 2 de Julho de 2020.
- [2] Como funciona o HTTP, URL, Link e WWW em um site? [Online] <http://monge.com.br/blog-detalle-seo-consultoria-web-sites/como-funciona-o-http-url-link-e-www-em-um-site/17>. Último acesso a 7 de Junho de 2020.
- [3] What is Selenium WebDriver? Difference with RC . [Online] <https://www.guru99.com/introduction-webdriver-comparison-selenium-rc.html>. Último acesso a 2 de Julho de 2020.
- [4] SeleniumHQ Browser Automation, 2019. [Online] <https://selenium.dev/>. Último acesso a 2 de Julho de 2020.
- [5] Allyn Grey de Almeida Lima. Padrão SQL e sua Evolução, 2019. [Online] <http://www.ic.unicamp.br/~geovane/mo410-091/Ch05-PadraoSQL-art.pdf>. Último acesso a 2 de Julho de 2020.
- [6] Ana Ribeiro. Ana Ribeiro GBD, 2013. [Online] <http://anaferreira403.blogspot.com/2013/10/os-tres-principais-modelos-conceptuais.html>. Último acesso a 1 de Maio de 2020.
- [7] ByLearn. 11 Motivos para migrar para o VS Code, 2019. [Online] <https://medium.com/@bylearn/11-motivos-para-migrar-para-o-vs-code-5b9574a057f5>. Último acesso a 2 de Julho de 2020.
- [8] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–389, 1970.
- [9] José Coelho. Introdução à base de dados, 2011. [Online] <https://repositorioaberto.uab.pt/bitstream/10400.2/3462/1/Introdu%C3%A7%C3%A3o%20%C3%A0%20Base%20de%20Dados.pdf>. Último acesso a 1 de Maio de 2020.

-
- [10] Devmedia. Introdução à Interface Homem-Máquina, 2012. [Online] <https://www.devmedia.com.br/introducao-a-interface-homem-maquina/24013>. Último acesso a 28 de Julho de 2020.
- [11] Mark Drake. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems, 2019. [Online] <https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems>. Último acesso a 2 de Julho de 2020.
- [12] Elaine Martins. O que é World Wide Web?, 2008. [Online] <https://www.tecmundo.com.br/web/759-o-que-e-world-wide-web-.htm>. Último acesso a 1 de Maio de 2020.
- [13] Eduardo Figueiredo. Requisitos Funcionais e Requisitos Não Funcionais. [Online] https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/req-funcional-rnf_v01.pdf. Último acesso a 5 de Agosto de 2020.
- [14] Martin Fitzpatrick. *Create Simple GUI Applications, with Python Qt5*. Leanpub, 2019.
- [15] Martin Fitzpatrick. The complete PyQt5 tutorial - Create GUI applications with Python, 2019. [Online] <https://www.learnpyqt.com/>. Último acesso a 30 de Abril de 2020.
- [16] Python Software Foundation. PyQt5 5.14.2, 2020. [Online] <https://pypi.org/project/PyQt5/>. Último acesso a 1 de Abril de 2020.
- [17] The Python Software Foundation. The Not So Short Introduction to \LaTeX , 2013. [Online] <https://docs.python.org/3/library/unittest.html>. Último acesso a 22 de Agosto de 2020.
- [18] Amir Ghahrai. Test Automation Advantages and Disadvantages, 2020. [Online] <https://devqa.io/test-automation-advantages-and-disadvantages/>. Último acesso a 1 de Maio de 2020.
- [19] Computer Hope. Command line vs. GUI, 2018. [Online] <https://www.computerhope.com/issues/ch000619.htm>. Último acesso a 1 de Maio de 2020.

- [20] Juan Jose. Selenium 4 is now W3C compliant. What does this mean?, 2019. [Online] <https://medium.com/@juanba48/selenium-4-is-now-w3c-compliant-what-does-this-mean-ceb44\de2d29b>. Último acesso a 2 de Julho de 2020.
- [21] Maria Inês Coelho. A História das Bases de Dados - O Início, 2019. [Online] <https://pplware.sapo.pt/software/a-historia-das-bases-de-dados-o-inicio/>. Último acesso a 1 de Maio de 2020.
- [22] Mastertech. 10 motivos para você aprender Python, 2018. [Online] <https://www.hostgator.com.br/blog/10-motivos-para-voce-aprender-python/>. Último acesso a 2 de Julho de 2020.
- [23] MDN contributors. World Wide Web, 2019. [Online] https://developer.mozilla.org/pt-PT/docs/Gloss%C3%A1rio/World_Wide_Web. Último acesso a 1 de Maio de 2020.
- [24] Compare o Mercado. Comparador Nº1 em Seguro Automóvel, 2018. [Online] <https://www.compareomercado.pt/simulacao-seguro-automovel>. Último acesso a 27 de Agosto de 2020.
- [25] Oracle. History of SQP, 2003. [Online] https://docs.oracle.com/cd/B12037_01/server.101/b10759/intro001.htm. Último acesso a 2 de Julho de 2020.
- [26] Pedro Pinto. Visual Studio Code: O melhor editor para programadores?, 2017. [Online] <https://pplware.sapo.pt/software/visual-studio-code-melhor-editor-programadores/>. Último acesso a 2 de Julho de 2020.
- [27] Deco Proteste. Seguro automóvel: qual o mais barato?, 2020. [Online] <https://www.deco.proteste.pt/auto/seguro-automovel/simule-e-poupe/seguro-automovel-qual-o-mais-barato?landingpage>. Último acesso a 27 de Agosto de 2020.
- [28] Ana Paula Santos. A Importância da Interação Humano-Computador, 2012. [Online] <http://tiqx.blogspot.com/2012/02/compreenda-importancia-da-interacao.html>. Último acesso a 28 de Julho de 2020.

-
- [29] SohomPramanick. History of Python. [Online] <https://geeksforgeeks.org/history-of-python/>. Último acesso a 7 de Junho de 2020.
- [30] Riverbank Computing. Modules. [Online] <https://riverbankcomputing.com/software/pyqt/intro>. Último acesso a 6 de Agosto de 2020.
- [31] Riverbank Computing. What is PyQt? [Online] https://www.riverbankcomputing.com/static/Docs/PyQt5/module_index.html. Último acesso a 7 de Agosto de 2020.
- [32] Stack Overflow. 2020 Developer Survey, 2020. [Online] <https://insights.stackoverflow.com/survey/2020>. Último acesso a 1 de Julho de 2020.
- [33] The Qt Company. One framework. One codebase. Any platform. [Online] <https://www.qt.io/>. Último acesso a 6 de Agosto de 2020.
- [34] Universidade Católica do Porto. Modelização da informação - Os vários Modelos de dados, 2019. [Online] <https://www.porto.ucp.pt/nonio/nonio/Tabelas%20e%20consultas%20-%20MSACCESS/ContAnalise.htm>. Último acesso a 1 de Maio de 2020.