

Sistema de gestão de dados para uma oficina de automóveis

PROJETO TECNOLÓGICO



PROFESSORA: SOLANGE RIBEIRO

Tomás Neto 12º1C

## Agradecimentos

Todo o esforço, apresentado no presente relatório, não seria possível sem uma perseverança pessoal e coletiva dos meus familiares e amigos, devido às circunstâncias atuais, as quais nunca foram vividas anteriormente. No entanto, gostaria de deixar um especial agradecimento a todos os professores que me apoiaram, ao longo destes três anos do ensino secundário, por me terem apoiado e ajudado a crescer como pessoa, mas também na minha evolução a nível de conhecimentos, capacidades e competências científicas e práticas.

## Resumo

No âmbito da disciplina de Projeto Tecnológico, foi proposta a criação de um projeto utilizando as aprendizagens do curso de Informática Aplicada à Web. Deste modo, inspirado pela necessidade de um sistema de gestão de dados do meu pai para a sua oficina, decidi criar um site, com o objetivo de organizar os dados dos veículos, respetivos clientes e reparações.

Ao longo deste relatório, será apresentado todo o processo de análise, planeamento, arquitetura e implementação da base de dados, bem como todo o software de interação com os diversos utilizadores.

Numa primeira parte, aborda-se a definição do sistema, na qual se apresenta uma contextualização do mesmo e de seguida, exploram-se os requisitos necessários, casos de uso e base de dados.

Posteriormente, aborda-se o Modelo Concetual da base de dados, apresentado através do diagrama ER criado no “BrModelo”. Nesta fase, exploram-se as entidades, relacionamentos, atributos e cardinalidade que caracterizam a base de dados do sistema.

De seguida, apresenta-se as tecnologias utilizadas para a realização do projeto, como as linguagens de programação e *frameworks* utilizadas.

Na quarta parte, é explicado o código, que permite o funcionamento da aplicação *web* desenvolvida.

Por fim, é feita uma análise da integralidade do sistema de gestão de dados, de forma a garantir que os dados são devidamente ordenados de acordo com os requisitos do site.

## Índice

Agradecimentos .....	1
Resumo.....	2
Índice .....	3
Índice de Imagens.....	4
1.Introdução.....	5
1.1 Contextualização.....	5
1.2 Apresentação dos casos de estudo .....	5
1.3 Motivação e Objetivos .....	5
1.4 Estrutura do Relatório.....	5
2. Requisitos do Sistema .....	7
2.1 Contexto da aplicação do sistema.....	7
2.2 Fundamentação da implementação da base de dados .....	7
2.3 Análise da viabilidade do processo.....	8
2.4 Levantamento e Análise de Requisitos .....	8
3.Desenvolvimento da Base de Dados e do Sistema .....	9
3.1 Diagrama ER.....	9
3.2 Casos de Uso.....	10
3.3 Implementação Física .....	10
3.3.1 Linguagens de programação Utilizadas .....	10
3.3.2 Desenvolvimento do Código da aplicação.....	13
Conclusão.....	23
Bibliografia .....	24

## Índice de Imagens

Figura 1- Planeamento da construção do sistema .....	8
Figura 2 - Diagrama ER.....	9
Figura 3 - Casos de Uso .....	10
Figura 4 - Bibliotecas Importadas.....	13
Figura 5 – Tabela “clientes” Base de dados. ....	13
Figura 6 - Função de registo de clientes .....	14
Figura 7 - Formulário de registo de clientes .....	14
Figura 8 - Registo com sucesso de um cliente.....	15
Figura 9 - Lista de clientes.....	15
Figura 10 - Perfil do cliente .....	16
Figura 11 - Parte visual para o cliente .....	16
Figura 12 - Zona Utilizador/Administrador.....	17
Figura 13 - Lista de reparações existentes no Sistema .....	17
Figura 14 - Reparação Aberta.....	18
Figura 15 - Reparação Concluída .....	18
Figura 16 - Zona de impressão de documentos do Windows .....	19
Figura 17 - Área de gestão de dados do estabelecimento .....	19
Figura 18 - Código para fazer Update .....	20
Figura 19 - Lista de funcionários do estabelecimento .....	20
Figura 20 - Folha de registo de funcionários .....	21

## **1.Introdução**

De modo a introduzir o relatório do trabalho realizado, primeiramente, é feita uma pequena contextualização do projeto, seguida da apresentação, motivação e a definição dos objetivos do mesmo. Por fim, será feita a exposição da estrutura deste relatório.

### **1.1 Contextualização**

No âmbito da disciplina de Projeto Tecnológico, foi proposta a elaboração de um projeto de tema livre, para o qual foi escolhido o tema: Sistema de gestão de dados de uma oficina automóvel.

### **1.2 Apresentação dos casos de estudo**

O caso de estudo apresentado é um sistema de gestão de dados de uma oficina automóvel. Deste modo, reuni um conjunto de informações importantes que devem ser aplicadas no contexto da base de dados. Para tal, foram identificadas e caracterizadas as diversas entidades do modelo e definidos os relacionamentos entre as mesmas. Uma vez que o objetivo primordial deste trabalho é dar resposta às necessidades e interrogações do utilizador, foram construídas de diferentes *queries* que resultam das relações entre as entidades.

### **1.3 Motivação e Objetivos**

Nos dias de hoje em que o mundo está cada vez mais digital, o controlo e a gestão de informação podem ser os pontos chave para que uma organização consiga atingir o sucesso pretendido. Para que este seja alcançado, as bases de dados são ferramentas úteis e fundamentais em contexto organizacional. Neste sentido, surge este projeto, que tem como principal objetivo, de forma geral, fazer os registos relativos aos clientes e aos veículos numa oficina automóvel, para possibilitar o acesso mais rápido, detalhado e organizado à informação relativa às reparações.

### **1.4 Estrutura do Relatório**

Para além desta Introdução, este relatório está organizado da seguinte forma:

No Capítulo II, expõe-se o contexto da aplicação do sistema com uma fundamentação da necessidade de organizar e implementar uma base de dados na organização estudada.

No Capítulo III faz-se o levantamento dos requisitos e os casos de uso. Por fim, os mesmos são analisados e validados.

É também apresentada a modelação concetual do modelo, com a respetiva identificação e caracterização das entidades, dos relacionamentos e da associação dos atributos com as entidades e relacionamentos. Por fim, é explicado o diagrama ER que foi construído.

É ainda apresentado e explicado o código funcional da aplicação que vai permitir ver as informações de forma organizada.

Por último, no Capítulo IV são apresentadas as principais conclusões e são também indicadas algumas propostas de trabalho futuro.

## **2. Requisitos do Sistema**

### **2.1 Contexto da aplicação do sistema**

As oficinas automóveis são os estabelecimentos em que se realizam as operações de manutenção e reparação (mecânica, elétrica e eletrónica) de veículos automóveis (ligeiros e pesados). Inclui as atividades de lavagem, polimento, pintura, tratamento antiferrugem, reparação de componentes, substituição ou instalação (de pneus, para-brisas, vidros, rádios, jantes, etc.).

Estas oficinas automóveis têm de ter registado todos os detalhes de cada manutenção efetuada em determinado veículo, como por exemplo, as horas de trabalho realizadas num certo automóvel, a quantidade de peças e a respetiva identificação, o dono do veículo em questão, entre muitas outras questões, para questões financeiras, mas também para uma questão de gestão do estabelecimento.

### **2.2 Fundamentação da implementação da base de dados**

Ao longo dos anos, a tecnologia foi evoluindo como sabemos, mas muitas destes estabelecimentos continuam a registar os dados de clientes e veículos em formato de papel, o que traz várias consequências negativas. Assim, de modo a dar resposta a estes problemas, decidi criar um sistema que fosse resolver grande parte deles dos problemas de utilização de manuscritos, tais como perda de informação e desorganização.

Em primeiro lugar, a administração tem certa dificuldade em gerir a informação que é guardada que nem sempre é detalhada de forma correta e homogénea. Ao escrevermos num papel, dependendo de pessoa para pessoa, cada um tem o seu tipo de letra e forma de escrever, o que pode dificultar a perceção de certa informação. Quando é registado os dados de um veículo, todos os campos devem ser necessariamente preenchidos, como por exemplo, a matrícula, marca, modelo, entre outros dados. No caso de dados mais específicos, nem todas as pessoas preenchem todos os campos de informação necessários, por exemplo, o número de chassi, a potência do motor, a cilindrada, entre outras. Deste modo, ter a informação corretamente detalhada seria uma mais-valia para a administração ter os dados de forma perceptível.

Além disso, o papel ocupa um grande espaço de armazenamento e perde-se muito tempo a procurar registos, para não falar do alto custo da produção do papel:

1. São necessárias 12 árvores e cerca de 540 mil litros de água para produzir uma tonelada de papel;
2. Essa produção emite cerca de 1.5 toneladas de CO<sub>2</sub> para a atmosfera;
3. Apenas 37% do que é produzido é reciclado.

A produção de papel é um dos processos mais poluentes que existe. Além do mais, cada documento que precisa de ser transportado de um lugar a outro requer o uso de combustível, cuja queima também emite gases poluentes para o meio ambiente. Ou seja, usar menos papel ajuda a reduzir a quantidade de substâncias nocivas na atmosfera.



## 2.3 Análise da viabilidade do processo

O principal objetivo a alcançar com a construção desta base de dados consiste em reunir todas as informações relativas às oficinas automóveis realizados, de modo a auxiliar a administração na gestão dos dados dos seus clientes.

Deste modo, pretende-se guardar a informação dos clientes da oficina e dos respetivos veículos e as resparações dos mesmos.

O projeto decorerá de acordo com o planeamento expresso no seguinte diagrama de *Gantt*:

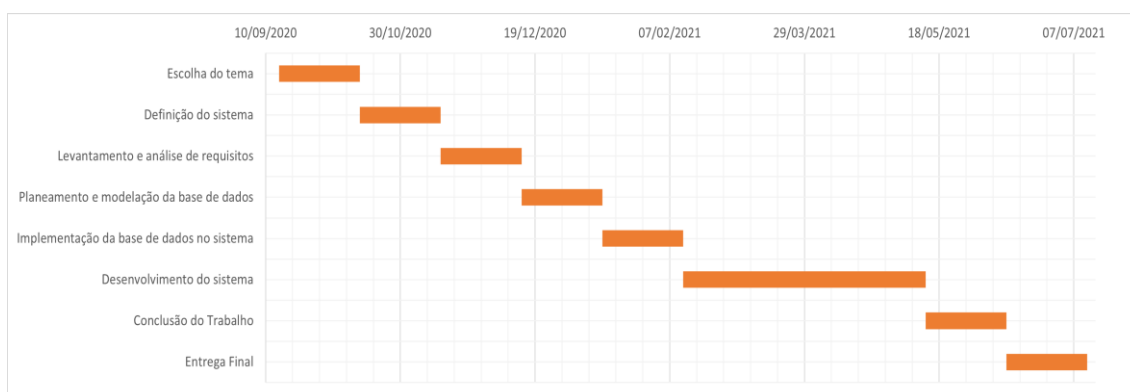


Figura 1- Planeamento da construção do sistema

## 2.4 Levantamento e Análise de Requisitos

Neste capítulo irei abordar quais os requisitos considerados para a concretização deste projeto.

De modo a ter uma perceção real do funcionamento da atividade de uma oficina automóvel, contactei uma oficina automóvel em Paços de Ferreira que me permitiu fazer o seguinte levantamento de requisitos:

1. Deve fazer CRUD (Create Read Update Delete) de clientes;
2. Deve fazer CRUD de administradores do sistema;
3. Deve fazer CRUD de veículos;
4. Deve fazer CRUD de cada reparação de cada veículo;
5. Deve fazer CRUD de dos dados do estabelecimento (contactos e morada);
6. Deve permitir imprimir as reparações;
7. Deve permitir contar quanto tempo demorou a reparação a ser concluída;
8. Deve ter um perfil de cada cliente onde mostra todas as informações referentes;
9. Deve ter um perfil de cada veículo onde mostra todas as informações referentes;
10. Deve ter um perfil de cada administrador onde mostra todas as informações referentes;
11. Deve ter permissões para diferentes tipos de utilizadores (Patrão, Gerente, Empregado).
12. Deve ter uma página de contactos e morada.
13. Deve ter um sistema de login de administradores;

### 3.Desenvolvimento da Base de Dados e do Sistema

#### 3.1 Diagrama ER

O desenvolvimento de uma base de dados pressupõe a construção de um diagrama ER (Entidade-Relacionamento), sendo necessário, para tal, seguir as seguintes etapas: a identificação das entidades e relacionamentos entre as mesmas; a associação dos atributos às entidades e relacionamentos; a identificação do domínio dos atributos; e, a determinação das chaves candidatas, primárias e estrangeiras.

Este tipo de modelo é uma mais valia para apresentar e validar o modelo junto do utilizador, uma vez que tem uma forte apresentação gráfica como a seguinte:

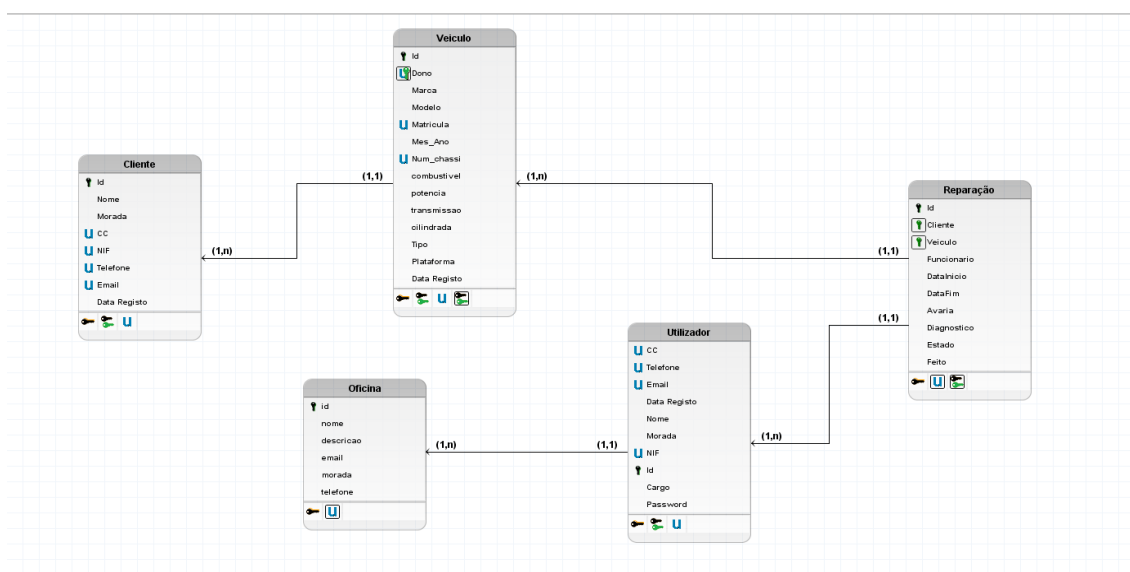


Figura 2 - Diagrama ER

## 3.2 Casos de Uso

Para ajudar a perceber o que o utilizador do site poderá fazer, foi-me pedido que desenhasse um diagrama de Casos de Uso que mostrará todas as permissões e funções do site:

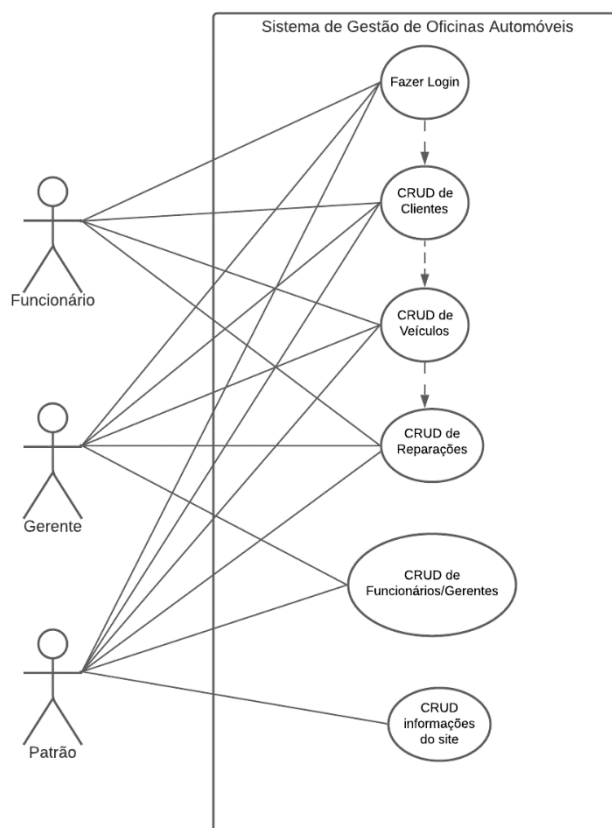


Figura 3 - Casos de Uso

## 3.3 Implementação Física

### 3.3.1 Linguagens de programação Utilizadas

Para o desenvolvimento desta aplicação utilizei duas linguagens de programação e uma de marcação. Utilizei também um framework para me ajudar e facilitar a programação.

As Linguagens Utilizadas são:

#### Python

Python é uma linguagem de programação de alto nível, interpretada por scripts, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por

Guido van Rossum em 1991. Atualmente, possui um modelo de desenvolvimento comunitário, aberto e gerido pela organização sem fins lucrativos Python Software Foundation. Apesar de várias partes da linguagem possuírem padrões e especificações formais, a linguagem, como um todo, não é formalmente especificada. O padrão de facto é a implementação CPython.

A linguagem foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Prioriza a legibilidade do código sobre a velocidade ou expressividade. Combina uma sintaxe concisa e clara com os recursos poderosos da sua biblioteca padrão e por módulos e frameworks desenvolvidos por terceiros.

Python é uma linguagem de propósito geral de alto nível, multiparadigma, suporta o paradigma orientado a objetos, imperativo, funcional e procedural. Possui tipagem dinâmica e uma de suas principais características é permitir a fácil leitura do código e exigir poucas linhas de código quando comparado ao mesmo programa em outras linguagens.

### **JavaScript**

JavaScript (frequentemente abreviado como JS) é uma linguagem de programação interpretada estruturada, por scripts em alto nível com tipagem dinâmica fraca e multiparadigma (protótipos, orientado a objeto, imperativo e, funcional). Juntamente com HTML e CSS, o JavaScript é uma das três principais tecnologias da World Wide Web. JavaScript permite criar páginas da Web interativas e, portanto, é uma parte essencial das aplicações web. A grande maioria dos sites usa, e todos os principais navegadores têm um mecanismo JavaScript dedicado para executá-lo.

É atualmente a principal linguagem para programação client-side em browsers web. É também bastante utilizada do lado do servidor através de ambientes como o node.js.

Como uma linguagem multiparadigma, o JavaScript suporta estilos de programação orientados a eventos, funcionais e imperativos (incluindo orientado a objetos e prototype-based), apresentando recursos como fechamentos (closures) e funções de alta ordem comumente indisponíveis em linguagens populares como Java e C++. Possui APIs para trabalhar com texto, matrizes, datas, expressões regulares e o DOM, mas a linguagem em si não inclui nenhuma E/S, como instalações de rede, armazenamento ou gráficos, contando com isso no ambiente host em que está embutido.

### **HTML**

HTML (abreviação para a expressão inglesa HyperText Markup Language, que significa: "Linguagem de Marcação de Hipertexto") é uma linguagem de marcação utilizada na construção de páginas na Web. Os documentos HTML podem ser interpretados por browsers. A tecnologia é fruto da junção entre os padrões HyTime e SGML.

HyTime é um padrão para a representação estruturada de hipermídia e conteúdo baseado em tempo. Um documento é visto como um conjunto de eventos concorrentes dependentes de tempo (como áudio, vídeo, etc.), ligados por hiperligações (em inglês: hyperlink e link). O padrão é independente de outros padrões de processamento de texto em geral.

SGML é um padrão de formatação de textos. Não foi desenvolvido para hipertexto, mas tornou-se conveniente para transformar documentos em hiper-objetos e para descrever as ligações.

## **CSS**

Cascading Style Sheets (CSS) é um mecanismo para adicionar estilo (cores, fontes, espaçamento, etc.) a um documento web.

O código CSS pode ser aplicado diretamente nas tags ou ficar contido dentro das tags <style>. Também é possível, em vez de colocar a formatação dentro do documento, criar um link para um arquivo CSS que contém os estilos. Assim, quando se quiser alterar a aparência dos documentos vinculados a este arquivo CSS, basta modifica-lo.

Com a variação de atualizações dos browsers, o suporte ao CSS pode variar. A interpretação dos navegadores pode ser avaliada com o teste Acid2, que se tornou uma forma base de revelar quão eficiente é o suporte de CSS, fazendo com que a nova versão em desenvolvimento do Firefox seja totalmente compatível a ele, assim como o Opera já é. O Doctype informado, ou a ausência dele, determina o quirks mode ou o strict mode, modificando o modo como o CSS é interpretado e a página desenhada.

## **FLASK**

Flask é um pequeno framework web escrito em Python. É classificado como um microframework porque não requer ferramentas ou bibliotecas particulares, mantendo um núcleo simples, porém, extensível. Não possui camada de abstração de base de dados, validação de formulário ou quaisquer outros componentes onde bibliotecas de terceiros pré-existentes fornecem funções comuns. No entanto, o Flask oferece suporte a extensões que podem adicionar recursos do aplicativo como se fossem implementados no próprio Flask. Existem extensões para mapeadores objeto-relacional, validação de formulário, manipulação de upload, várias tecnologias de autenticação aberta e várias ferramentas comuns relacionadas ao framework.

Aplicações que utilizam o framework Flask incluem a própria página da comunidade de desenvolvedores: o Pinterest e o LinkedIn.

### 3.3.2 Desenvolvimento do Código da aplicação

No desenvolvimento da aplicação, tive primeiro que importar as seguintes bibliotecas:

```
from flask import Flask, redirect, render_template, request, flash
from flask_bcrypt import Bcrypt
from flask_login import LoginManager, login_user, UserMixin, login_required, current_user, logout_user
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime, date
from flask_mail import Mail, Message
import time
```

Figura 4 - Bibliotecas Importadas

De seguida defini o Código da Base de Dados da seguinte forma:

```
class Clients(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    nome = db.Column(db.String(100), nullable = False)
    morada = db.Column(db.String(500), nullable = False, default="Nao inserido")
    cc = db.Column(db.String(10), nullable = False, default="Nao inserido")
    nif = db.Column(db.String(20), nullable = False, default="Nao inserido", unique=True)
    telefone = db.Column(db.Integer, nullable = False, default="Nao inserido", unique=True)
    email = db.Column(db.String(100), nullable = False, default="Nao inserido", unique=True)
    data_registo = db.Column(db.String(20), nullable = False, default=datetime.now().strftime("%d/%m/%Y"))

    def __repr__(self):
        return 'Cliente ' + str(self.id)
```

Figura 5 – Tabela “clientes” Base de dados.

Todas as entidades foram assim definidas, são classificadas como classes e recebem um argumento que faz com sejam uma tabela da Base de Dados. Todos os campos da tabela são definidos como na imagem como por exemplo o “id” ou o “nome”. Os argumentos que são passados para cada campo têm uma justificação:

nullable = False -> Significa que aquele valor tem que existir obrigatoriamente;  
default = “Não Inserido” -> Significa que se o valor não for inserido através de um registo, esse campo receberá o valor “Não Inserido”

Para registar um cliente, precisei de desenvolver o Código seguinte:

```
#Register Client
@app.route("/admindashboard/addclient", methods=['GET','POST'])
@login_required
def addclient():
    if request.method == "POST":

        nome = request.form['nome']
        cc = request.form['cc']
        nif = request.form['nif']
        morada = request.form['morada']
        telemovel = request.form['telemovel']
        email = request.form['clientemail']

        checkemail = db.session.query(Clients.query.filter(Clients.email == email).exists()).scalar()
        checkcc = db.session.query(Clients.query.filter(Clients.cc == cc).exists()).scalar()
        checknif = db.session.query(Clients.query.filter(Clients.nif == nif).exists()).scalar()
        checktelemovel = db.session.query(Clients.query.filter(Clients.telefone == telemovel).exists()).scalar()

        if not checkemail and not checkcc and not checknif and not checktelemovel:
            new_register = Clients(nome = nome, morada = morada, cc = cc, nif = nif, telefone = telemovel, email = email )
            db.session.add(new_register)
            db.session.commit()
            flash('Cliente registado com sucesso.','success')
            return redirect("/admindashboard/addclient")

        else:
            flash('Cliente já existe.','danger')
            return redirect("/admindashboard/addclient")

    return render_template("Software/pages/forms/formAddClient.html")
```

Figura 6 - Função de registo de clientes

@app.route("/admindashboard/addclient", methods=['GET','POST']) -> serve para especificar qual o caminho/link que eu preciso de estar para aceder ao formulário de registo de clients, ou seja, o seguinte:

Figura 7 - Formulário de registo de clientes

@login\_required -> serve para apenas deixar utilizadores autenticados entrar na página do formulário, esta função necessita da biblioteca “flask\_login”.

De seguida, a função faz uma verificação do método que foi utilizado, se for “POST”, significa que queremos enviar dados para a Base de Dados e por isso ele executa o Código dentro do “If”. Se o método for “GET” significa que apenas queremos receber os dados da página e portanto ele retorna o ficheiro .html do formulário de registo de clients.

Dentro do ciclo “If”, o algoritmo faz uma verificação na base de dados da existência dos dados enviados, caso os dados já sejam repetidos vai enviar uma mensagem a dizer que

o utilizador já existe, senão forem repetidos ele regista na base de dados o novo cliente como na imagem seguinte:

The screenshot shows a web application interface for client registration. The title bar includes 'Início', 'Ajuda', and 'Nova Reparação'. The main content area is titled 'Folha de registo clientes' and displays a green success message: 'Cliente registado com sucesso.' Below this is a form titled 'Dados Cliente' with fields for Name, Address, N° Cartão Cidadão, NIF, Telephone, and Email. A 'Registar Cliente' button is at the bottom. The footer contains copyright information: 'Copyright © 2020 Tomás Neto 12º1c Colégio de São Gonçalo - Amarante. All rights reserved.'

Figura 8 - Registo com sucesso de um cliente

Por conseguinte, o novo cliente deve ser mostrado numa lista de clientes e ter um perfil com todos os dados inseridos no seu registo.

The screenshot shows a web application interface for a list of clients. The title bar includes 'Início', 'Ajuda', and 'Nova Reparação'. The main content area is titled 'Lista Clientes' and displays a table with columns: #, Nome, Morada, CC, NIF, Telefone, Email, and Opções. The table contains one entry for 'Tomás Neto'. The footer contains copyright information: 'Copyright © 2020 Tomás Neto 12º1c Colégio de São Gonçalo - Amarante. All rights reserved.'

#	Nome	Morada	CC	NIF	Telefone	Email	Opções
1	Tomás Neto	Portugal	12345679	132465798	987654321	tomas@email.com	

Figura 9 - Lista de clientes

Nesta foto conseguimos ver a lista de clientes e algumas opções:

A opção com o botão com um “i” de “informação” que redireciona para o perfil do cliente:

A opção vermelha com um caixote do lixo serve para eliminar o registo do cliente.



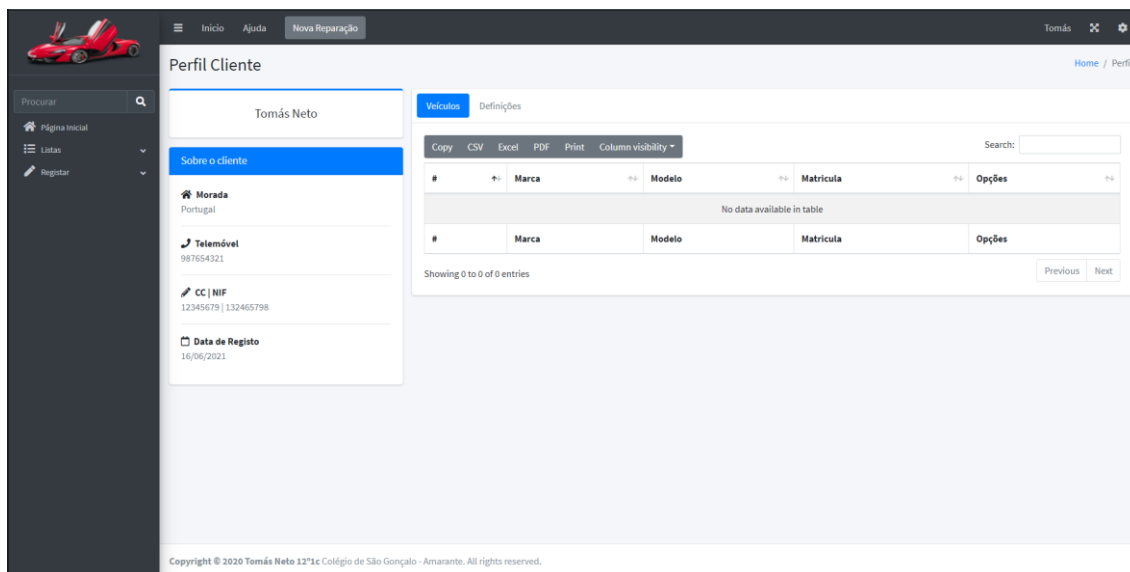


Figura 10 - Perfil do cliente

O perfil do cliente deve mostrar todos os dados relativos ao cliente em questão como também todos os seus veículos registados no Sistema.

Todo o site está definido desta forma, tem tanto uma parte visual para o cliente:



Figura 11 - Parte visual para o cliente

Como também tem a parte do utilizador/administrador:

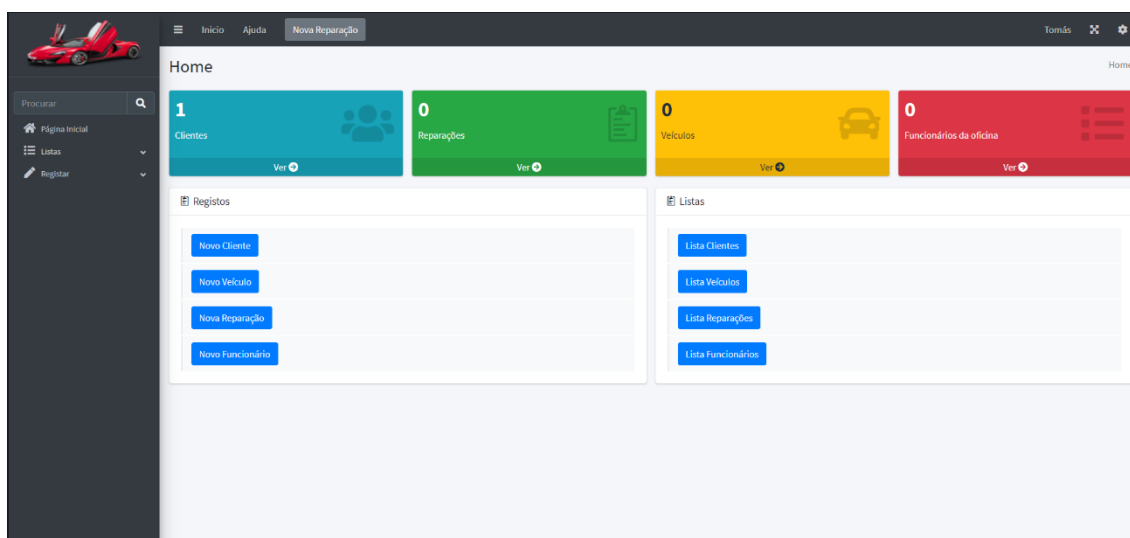


Figura 12 - Zona Utilizador/Administrador

É também possível visualizar todas as reparações que foram efetuadas no Sistema como também é possível imprimi-las.

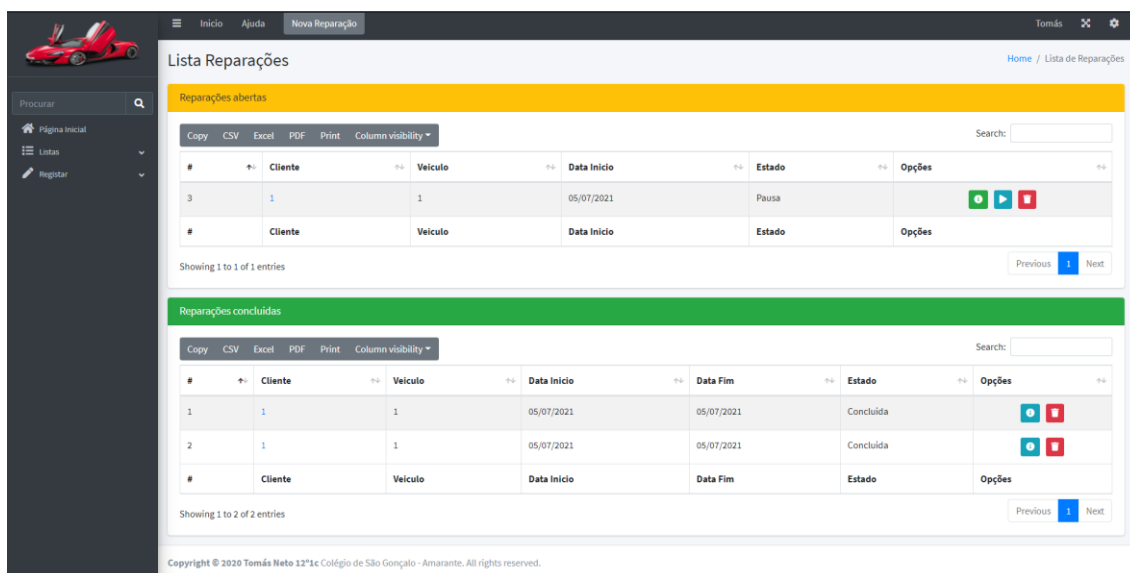


Figura 13 - Lista de reparações existentes no Sistema

Na figura acima podemos ver a página que mostra todas as reparações do Sistema. Temos as reparações Abertas, ou seja, as que ainda não foram concluídas, e temos as reparações concluídas.

Uma reparação quando é registada recebe o estado de “Aberta” ou “Concluída” conforme o decisão do utilizador. Se for escolhida como aberta ela terá o seguinte aspeto:

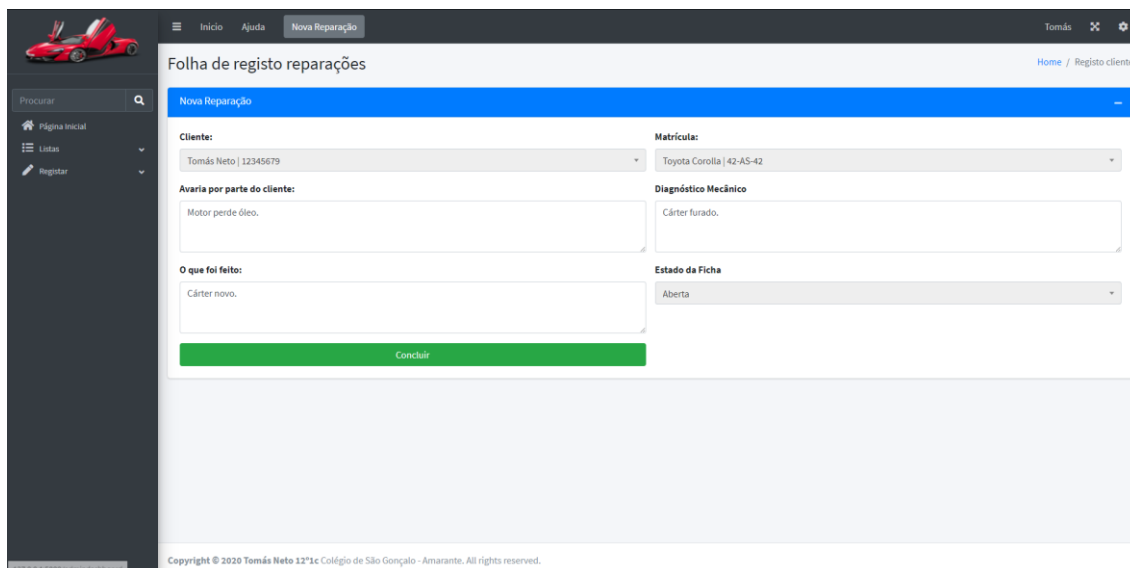


Figura 14 - Reparação Aberta

Aqui é onde o mecânico irá registar todos os detalhes relativos à reparação. Depois de todos os detalhes serem escritos a reparação deve ser concluída clicando em “Concluir”.

Depois de clicar no botão, a reparação terá o seguinte formato:

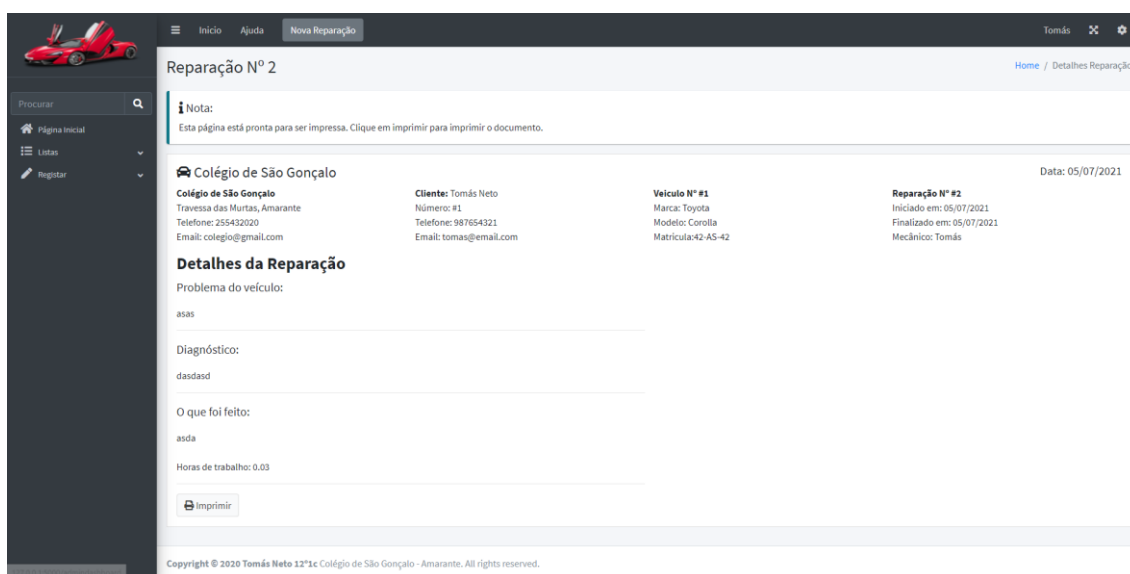


Figura 15 - Reparação Concluída

Nesta página Podemos ver todos os dados relativos à reparação como as informações da oficina, do cliente, do veículo e da reparação que clicando em “Imprimir” o documento é redirecionado para aplicação do Windows para imprimir:

**Colégio de São Gonçalo**

Colégio de São Gonçalo  
Travessa das Murtras, Amarante  
Telefone: 255432020  
Email: colegio@gmail.com

**Cliente:** Tomás Neto

Número: #1  
Telefone: 987654321  
Email: tomas@email.com

**Veículo Nº #1**

Marca: Toyota  
Modelo: Corolla  
Matrícula: 42-AS-42

**Reparação Nº #2**

Iniciado em: 05/07/2021  
Finalizado em: 05/07/2021  
Mecânico: Tomás

Data: 05/07/2021

**Detalhes da Reparação**

Problema do veículo:

asas

Diagnóstico:

dadasd

O que foi feito:

asda

Horas de trabalho: 0.03

Imprimir

1 sheet of paper

Destino: Microsoft Print to PDF

Páginas: Todos

Cor: Cor

Mais opções

Imprimir Cancelar

Figura 16 - Zona de impressão de documentos do Windows

Ainda na zona do utilizador/administrador existe uma área para o patrão do estabelecimento onde ele alterar os dados dos estabelecimento como a morada, contactos e até o nome.

Inicio Ajuda Nova Reparação

Tomás

**Dados Oficina**

Home / Dados Oficina

**Oficina**

**Nome:** Colégio de São Gonçalo

**Morada:** Travessa das Murtras, Amarante

**Descrição:** Escola Secundária

**Telefone:** 255432020

**Email:** colegio@gmail.com

Guardar

Copyright © 2020 Tomás Neto 12\*1c Colégio de São Gonçalo - Amarante. All rights reserved.

Figura 17 - Área de gestão de dados do estabelecimento

Para fazer o update dos dados utilizei o seguinte Código:

```
#Página Dados da Oficina
app.route("/admindashboard/oficina", methods=['GET', 'POST'])
@login_required
def oficina():
    if current_user.cargo == 'Patrao' or current_user.cargo == 'Gerente':
        oficina = Oficina.query.first()

        if request.method == "POST":
            nome = request.form['nome']
            morada = request.form['morada']
            descricao = request.form['descricao']
            telefone = request.form['telefone']
            email = request.form['email']
            dono = "Patrao"

            if oficina:
                Oficina.query.filter(Oficina.id == 1).update(dict(nome = nome, morada = morada, descricao = descricao, telefone = telefone, email = email))
            else:
                new_register = Oficina(id = 1, nome = nome, morada = morada, descricao = descricao, telefone = telefone, email = email)
                db.session.add(new_register)

            db.session.commit()

            flash('Dados guardados com sucesso!', 'success')
            return redirect("/admindashboard/oficina")

        return render_template("Software/pages/forms/oficina.html", oficina = oficina)
    else:
        return redirect("/admindashboard")
```

Figura 18 - Código para fazer Update

Aqui é comparado o método para aceder à página e faz a mudança na base de dados. Altera os velhos dados pelos novos inseridos no formulário e redireciona para a mesma página mostrando uma mensagem de sucesso.

Por fim foi feito ainda uma área que também só pode ser acedida pelo patrão ou gerente onde é possível controlar todos os funcionários que trabalham na oficina e também registar novos.

#	Nome	Morada	Cargo	CC	Telefone	Email	Opções
2	Gilberto	Portugal	Gerente	12336963	913654789	gilberto@email.com	

Figura 19 - Lista de funcionários do estabelecimento

Numa outra parte é possível registar novos funcionários que também podem fazer login no sistema uma vez que sejam registados por um “Gerente” ou pelo “Patrão”.

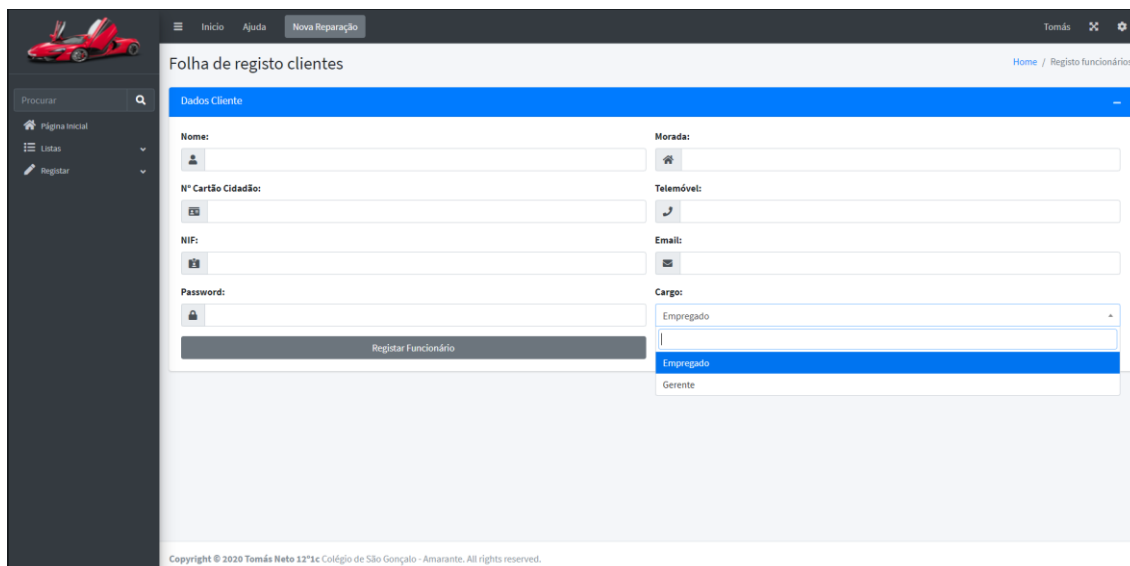
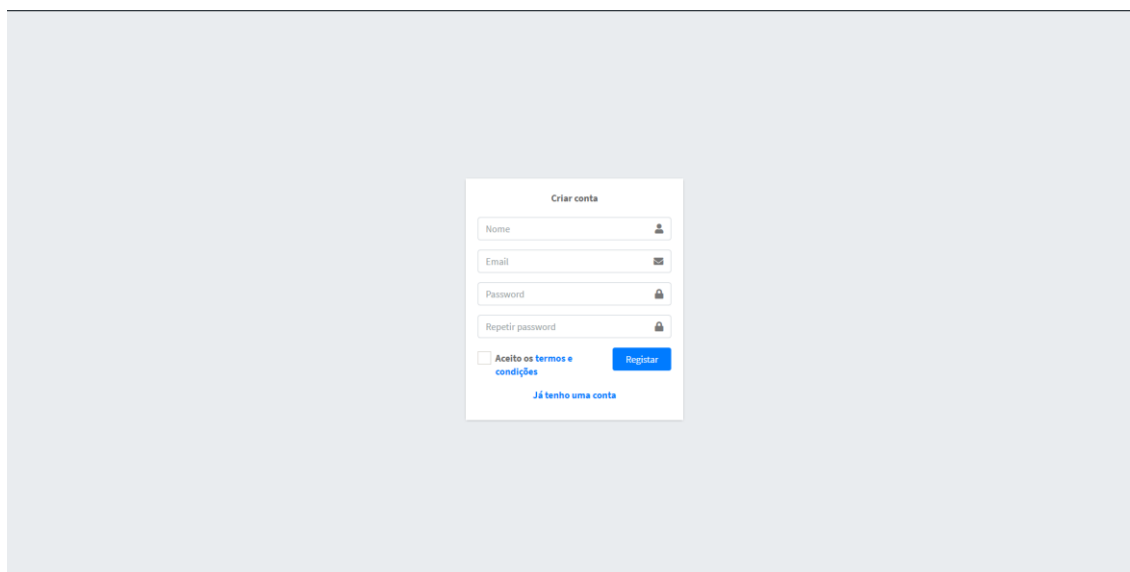


Figura 20 - Folha de registo de funcionários

É possível fazer um registo primário para um primeiro acesso à aplicação web através do seguinte formulário:



Este formulário só pode ser acedido caso não exista ainda um único utilizador registado na Base de Dados como mostra o código a seguir:

```
@app.route("/register", methods=['GET', 'POST'])
def register():
    oficina = Oficina.query.first()
    exists = db.session.query(users.query.filter(users.id == 1).exists()).scalar()

    if exists:
        return redirect("/")
    else:

        if request.method == 'POST':

            nome = request.form['usr_username']
            email = request.form['usr_email']
            password = request.form['usr_pwd']
            passwordconf = request.form['usr_repwd']

            checkemail = db.session.query(users.query.filter(users.email == email).exists()).scalar() #Verifica se email existe

            if checkemail == False and password == passwordconf:

                hash_password = bcrypt.generate_password_hash(password).decode('utf-8')

                new_register = users(nome = nome, email = email, password = hash_password)
                #new_oficina = oficina()
                db.session.add(new_register)
                #db.session.add(new_oficina)
                db.session.commit()
                return redirect("/login")

            else:
                flash('Email em uso ou passwords incorretas!')
                #return '<center><script>alert("Erro: Email em uso ou passwords incorretas!")</script> '
                return redirect("/register")

        return render_template("Software/pages/examples/register.html", oficina=oficina)
```

Neste código todas as informações são confirmadas, verificadas se já existem e se estão corretas, caso estejam as passwords corretas e o email ainda não existir, então é registado na base de dados um novo utilizador que poderá fazer login.

Por código *default* da base de dados, se não especificarmos um cargo para o utilizador ele recebe automaticamente o valor de “Patrao” e será então o patrão do estabelecimento.

Após este registo ter sido efetuado, sempre que alguém tentar acessar esta página será redirecionado para a página origem do site.

Todas as partes aqui apresentadas formam então o meu projeto de gestão de oficinas automóveis.

## Conclusão

O objetivo da realização deste trabalho consistiu em analisar, planejar, modelar e implementar um sistema de base de dados para oficinas automóveis.

Apesar de a dimensão do projeto ser pequena, deparei-me com alguns problemas que acabaram por ser todos resolvidos,

Em primeiro lugar, através do diálogo com o funcionário de uma oficina automóvel, que utilizei para obter mais informações sobre o contexto do caso de estudo escolhido, entendi de imediato o desafio colossal que é entender as necessidades do cliente. De facto, no contexto real, é extremamente difícil compreender a dimensão de um problema, e só estudo aprofundado, aliado a uma forte resiliência, permite combater todas as ambivalências que derivam do diálogo com os clientes. Deste modo, a identificação clara dos requisitos relevantes para o problema, assim como a validação de todas as fases com o utilizador, são condições *sine qua non* para o sucesso da base de dados a implementar.

Em suma, concluí que as bases de dados apresentam um papel fundamental no armazenamento de informação, mas que a sua implementação pode constituir um processo complexo. Além disso, concluí que os objetivos pretendidos foram alcançados, uma vez que apresentei uma base de dados funcional, capaz de auxiliar a vertente das oficinas automóveis.



## **Bibliografia**

<https://pt.wikipedia.org/wiki/Python>

[https://pt.wikipedia.org/wiki/Cascading\\_Style\\_Sheets](https://pt.wikipedia.org/wiki/Cascading_Style_Sheets)

[https://pt.wikipedia.org/wiki/Flask\\_\(framework\\_web\)](https://pt.wikipedia.org/wiki/Flask_(framework_web))

<https://www.youtube.com/watch?v=Qr4QMBUPxWo>

<https://pt.wikipedia.org/wiki/HTML>

<https://www.w3schools.com>