**Relatório de Projeto**

**Programação**

**Avaliação Periódica 3**

**Autores:**

Karine Aparecida Braga Florêncio

Nuno

**Data:** Janeiro de 2023

Resumo

Com o objetivo de a criar uma aplicação em Java para auxiliar uma oficina de automóveis foi desenvolvido um código com dezessete classes públicas de forma a praticar todo o conhecimento aprendido em nas aulas de programação. Durante o processo de criação foram realizados testes unitários de modo a encontrar e eliminar erros de programação. São cumpridos todos os requisitos inicialmente apresentados. Apesar de bem consolidada a Linguagem Java é observada que há formas mais sucintas de desenvolver o mesmo projeto, por exemplo, usando hierarquia e polimorfismo.

**Palavras-chave**

Algoritmos, Aplicação, Classes, Java para iniciantes, Java, Linguagem Java, Manual do Utilizador, Menu, Métodos, Oficina, Serviço, Utilizador, Veículo.

**Índice**

[Resumo ii](#_Toc123747569)

[Lista de Tabelas iv](#_Toc123747570)

[Lista de Figuras iv](#_Toc123747571)

[1. Introdução 1](#_Toc123747572)

[2. Objetivos e Metodologias 2](#_Toc123747573)

[2.1. Ferramentas e Tecnologias 4](#_Toc123747574)

[2.2. Planeamento 4](#_Toc123747575)

[3. Trabalho Desenvolvido 5](#_Toc123747576)

[3.1. Menu Desenvolvido 5](#_Toc123747577)

[3.2. Requisitos Implementados 6](#_Toc123747578)

[3.3. Classes 7](#_Toc123747579)

[3.4. Algoritmos 14](#_Toc123747580)

[3.5. Estruturas de Dados 15](#_Toc123747581)

[3.6. Armazenamento de Dados 15](#_Toc123747582)

[3.7. Procedimentos de Teste 15](#_Toc123747583)

[4. Conclusões 16](#_Toc123747584)

[4.1. Forças 16](#_Toc123747585)

[4.2. Limitações 16](#_Toc123747586)

[4.3. Trabalho Futuro 16](#_Toc123747587)

[5. Referências 17](#_Toc123747588)

[6. Anexos 18](#_Toc123747589)

[6.1. Manual do Utilizador 18](#_Toc123747590)

# Lista de Tabelas

[Tabela 1 - Requisitos cumpridos. 6](#_Toc123665263)

# Lista de Figuras

[Figura 1 - Diagrama de classes 7](#_Toc123747491)

[Figura 2 - Inicialização da aplicação. 18](#_Toc123747492)

[Figura 3 - Informações ao aceder a aplicação. 18](#_Toc123747493)

[Figura 4 - Autenticação. 19](#_Toc123747494)

[Figura 5 - Adicionar Utilizador. 19](#_Toc123747495)

[Figura 6 - Listar Utilizadores. 20](#_Toc123747496)

[Figura 7 - Ordenar Utilizadores. 20](#_Toc123747497)

[Figura 8 - Pesquisar Utilizador. 21](#_Toc123747498)

[Figura 9 - Remover Utilizador. 21](#_Toc123747499)

[Figura 10 - Alterar meu nome. 22](#_Toc123747500)

[Figura 11 - Alterar minha senha. 22](#_Toc123747501)

[Figura 12 - Consultar ações do sistema. 22](#_Toc123747502)

[Figura 13 - Gerir Veículos. 23](#_Toc123747503)

[Figura 14 - Adicionar Veículo. 24](#_Toc123747504)

[Figura 15 - Pesquisar Veículo. 24](#_Toc123747505)

[Figura 16 - Listar todos os Veículos. 25](#_Toc123747506)

[Figura 17 - Ordenar Veículos. 25](#_Toc123747507)

[Figura 18 - Listar Veículos de um Cliente. 26](#_Toc123747508)

[Figura 19 - Listar Veículos com serviço em aberto. 26](#_Toc123747509)

[Figura 20 - Listar Veículos que possuem uma peça. 27](#_Toc123747510)

[Figura 21 - Listar Veículos servidos após ano x. 27](#_Toc123747511)

[Figura 22 - Listar Veículos que gastou mais de x horas. 28](#_Toc123747512)

[Figura 23 - Gerir Veículo. 28](#_Toc123747513)

[Figura 24 - Adicionar Serviço. 29](#_Toc123747514)

[Figura 25 - Listar Serviços. 30](#_Toc123747515)

[Figura 26 - Gerir Serviço. 30](#_Toc123747516)

[Figura 27 - Adicionar Peça ao Serviço. 31](#_Toc123747517)

[Figura 28 - Adicionar Mecânico. 31](#_Toc123747518)

[Figura 29 - Listar Peças. 32](#_Toc123747519)

[Figura 30 - Listar Mecânicos. 32](#_Toc123747520)

[Figura 31 - Alterar custo. 32](#_Toc123747521)

[Figura 32 - Alterar descrição. 33](#_Toc123747522)

[Figura 33 - Gerir Peças. 33](#_Toc123747523)

[Figura 34 - Adicionar Peça. 34](#_Toc123747524)

[Figura 35 - Pesquisar Peça. 34](#_Toc123747525)

[Figura 36 - Listar peças com a mesma designação. 35](#_Toc123747526)

[Figura 37 - Listar peças de uma marca. 35](#_Toc123747527)

[Figura 38 - Listar peças de um fornecedor. 36](#_Toc123747528)

Lista de Acrónimos

*[Lista ordenada alfabeticamente]*

**NPC** Non-Player Character - é um personagem de jogo eletrônico que não pode ser controlado por um jogador.

**SDL** Simple DirectMedia Layer - é uma biblioteca de desenvolvimento de plataforma cruzada projetada para fornecer acesso de baixo nível a áudio, teclado, mouse, joystick e hardware gráfico via OpenGL e Direct3D.

# Introdução

Este trabalho pretende desenvolver uma aplicação com o objetivo de auxiliar uma oficina automóvel na gestão dos serviços realizados a veículos. Através da implementação de uma aplicação completa em linguagem **Java**, sem componente gráfica para interação com o utilizador, recorrendo ao paradigma de Programação Orientada a Objetos. Assim como o desenvolvimento e aplicação de conhecimentos e competências relacionadas com as temáticas abordadas na disciplina de programação, com especial incidência para:

* programação orientada a objetos;
* estruturas de dados;
* manipulação de ficheiros para armazenamento de dados de forma persistente;
* bem como o desenvolvimento de interfaces para interação com o utilizador (em modo texto).

# Objetivos e Metodologias

Tinhamos objetivo de

- inserir quadro das pontuações mais altas.

- server

## Ferramentas e Tecnologias

### Colaboração

Para trabalhar em cooperação usamos as ferramentas Git e GitHub.

**Git** é um sistema de controle de versão de código-fonte. Ele permite que se rastreie alterações em arquivos de código e reverta para versões anteriores se necessário. Git é amplamente utilizado por desenvolvedores de software para controlar o código-fonte de projetos de software, mas também pode ser usado para rastrear alterações em qualquer tipo de arquivo.

**GitHub** é um site que oferece serviços baseados em Git, incluindo armazenamento de código-fonte, rastreamento de problemas e gerenciamento de projetos. Ele é usado por milhões de desenvolvedores em todo o mundo para colaborar em projetos de software e hospedar projetos públicos e privados.

Para começar a usar Git e GitHub, é preciso instalar o Git em seu computador e criar uma conta no GitHub. Depois disso, pode criar um novo repositório (um lugar para armazenar o código) no GitHub e usar os comandos do Git para enviar (commit) alterações do código para o repositório. Também podemos usar o GitHub para trabalhar em projetos com outras pessoas, fazendo "pull requests" (solicitações de integração) para incluir alterações no código de outras pessoas em um repositório.

Aqui estão alguns dos comandos Git mais comuns para gerenciar o código:

* **git init**: Inicializa um novo repositório Git em um diretório existente.
* **git clone**: Clona um repositório Git existente em um novo diretório.
* **git add .**: Adiciona todos os arquivos ao índice (a área do Git onde as alterações são rastreadas).
* **git commit –m ‘mensagem’**: Salva as alterações no índice em um commit (uma versão do código) com o nome recebido entre aspas.
* **git push**: Envia os commits de um repositório local para um repositório remoto (como o GitHub).
* **git pull**: Puxa os commits de um repositório remoto para um repositório local.

Para trabalhar com branch (ramificações) em um projeto:

* **git branch**: Lista as branches do repositório ou cria uma nova branch.
* **git checkout**: Muda para uma branch existente ou cria uma nova branch e muda para ela.
* **git merge**: Mescla duas branches juntas.

Esses são apenas alguns exemplos dos muitos comandos que podemos usar com o Git. Existem muitos outros comandos disponíveis e cada um tem suas próprias opções e parâmetros. Para obter mais informações sobre os comandos Git, consulte a documentação do Git ou um tutorial online.**[][]**

O projeto realizado por no desenvolvimento deste relatório pode ser encontrado ao acessar <https://github.com/MrNoino/MGC-TP3.git>

### Desenvolvimento

Para a realização deste trabalho necessitamos de 2 semanas de aulas e horário de atendimento com o professor, além de extras horas, totalizando uma média de 20 horas semanais cada desenvolvedor.

Neste trabalho foram usadas as seguintes bibliotecas para a linguagem de programação Python:

As bibliotecas **sys**, **random**, **time**, **datetime**, **\_thread**, **pickle** e **socket** são bibliotecas padrão do Python o que significa que elas já estão disponíveis para usar sem a necessidade de instalar nada, só precisa incluir um *import* no início do código para poder usá-las. A biblioteca **sys** fornece acesso a alguns objetos e funções relacionados ao interpretador Python. A biblioteca **random** fornece funções para gerar números aleatórios. A biblioteca **time** fornece funções para trabalhar com tempo e data. A biblioteca **datetime** fornece classes para trabalhar com data e hora, é muito poderosa e é a biblioteca recomendada para qualquer tarefa de manipulação de data e hora no Python. A biblioteca **\_thread** fornece um módulo de baixo nível para trabalhar com threads em Python, é menos poderosa e mais complexa de usar do que outras bibliotecas de threads disponíveis para o Python, como o threading. A biblioteca **pickle** fornece um módulo para serializar e desserializar objetos Python, significa que você pode usar o pickle para converter objetos Python em uma representação de sequência de bytes, que pode ser armazenada em um arquivo ou transmitida pela rede, e depois recriar os objetos a partir dessa representação. A biblioteca **socket** fornece uma interface de baixo nível para criar e usar sockets de rede, podendo usa-la para criar clientes e servidores que se comunicam através da rede usando diferentes protocolos de rede, como o TCP e o UDP.**[][]**

**Pygame** é uma biblioteca de jogos para Python. Ela fornece uma ampla variedade de recursos para criar jogos 2D, incluindo gráficos, sons, entrada do usuário e sistemas de colisão. A Pygame é uma das bibliotecas mais populares para criar jogos em Python devido à sua simplicidade e à grande quantidade de recursos disponíveis. A Pygame adiciona funcionalidade à excelente biblioteca SQL, permitindo que você crie jogos completos e programas multimídia na linguagem python. A Pygame é altamente portátil e roda em quase todas as plataformas e sistemas operacionais. A instalação pode ser feita usando o gerenciador de pacotes Python pip. O Comando ***pip*** é automaticamente instalado após a instalação do Python e o comando instala tudo no ambiente Python. Abra o terminal e digite o seguinte comando:

**pip install pygame**

## Planeamento

O projeto foi planejado por etapas de desenvolvimento, sendo elas e suas cargas horárias:

1. Adaptação Git/GitHub – 6 horas.
2. Criação das classes – 3 horas.
3. Gerir NPC’s, Threads – 15 horas.
4. Correção de desafios encontrados. – 8 horas.
5. Animações. – 4 horas.
6. Menu / Teclas de atalhos. – 3 horas.
7. Implementar servidor. – 10 horas.
8. Elaborar relatório – 6 horas.
9. Elaborar manual do utilizador – 4 horas.

# Trabalho Desenvolvido

A aplicação é capaz de gerir Utilizadores, Veículos, Serviços e Estoque de peças de uma oficina. Isto é, adicionar, pesquisar, remover, listar são algumas das principais funcionalidades.

É necessário autenticação/registo para acesso de todas as funcionalidades da aplicação, só é possível registos de novos utilizadores a partir do registo por meio de um utilizador já cadastrado. Qualquer utilizador consegue manipular qualquer funcionalidade da aplicação.

## Classes

### Game

Classe de execução e interacção com o utilizador.

##### Métodos:

**main**: void – recupera dados dos ficheiro, inicializa a aplicação e armazena os dados em ficheiro.

**executar**: Utilizador – autenticação do utilizador e executa o menu de utilizadores da aplicação. Onde é possível adicionar, atualizar, remover, listar, ordenar e pesquisar utilizadores. Método de organização do código. Utiliza diversos métodos públicos e privados da classe.

### Entity

Representa um Sprite da aplicação.

##### Atributos:

**\_image** – Imagem do sprite.

**rect** – Objeto que armazena as coordenadas retangulares do sprite.

##### Métodos:

**setImage**(image) – Atualiza a imagem do sprite.

**getPostion**() – Fornece o ponto central do sprite.

**setPosition**(position, update) – Atualiza a posição do sprite de acordo com a position recebida alterada pelo update.

**move**(position) – Movimenta o sprite de acordo com o valor recebido.

**draw**(surface) – Imprime a imagem do sprite na posição que este se encontra.

### Enemy

Representa um NPC do jogo.

##### Atributos:

**\_speed\_animation** – velocidade em que a animação ocorre, quanto menor mais rápido.

**\_value** – valor usado para definir o index no array de imagens para realizar a animação de movimento do NPC.

**\_value\_dead** – valor usado para definir o index no array de imagens para realizar a animação de termino de um NPC.

**\_images** – array de imagens do NPC.

##### Métodos:

**setPosition**(position) – Atualiza a posição do NPC para 50 pixels a frente.

**getFinal**(): boolean – Verifica se um NPC chegou ao fim da tela, ganhando assim o jogo.

**move**(velocity) – Movimenta o NPC de acordo com o valor recebido e atualiza a animação.

**animatedead**(): boolean – Realiza a animação para finalizar o NPC.

### Player

Representa um veículo da aplicação.

##### Atributos:

**\_username** – nome do jogador.

**\_score** – pontuação do jogador.

**\_moving** – define se é permitido a movimentação do jogador.

**\_speed\_animation** – velocidade em que a animação ocorre, quanto menor mais rápido.

**\_value** – valor usado para definir o index no array de imagens para realizar a animação de movimento do jogador.

**\_value\_shoot** – valor usado para definir o index no array de imagens para realizar a animação de movimento de tiro de um jogador.

**\_value\_dead** – valor usado para definir o index no array de imagens para realizar a animação de termino de um jogador.

**\_images** – array de imagens do jogador.

##### Métodos:

**shoot**(all\_shots) – Realiza um disparo caso o jogador pressionar o ‘ESPAÇO’ e tiver tiros disponíveis, e verifica se um dos tiros passou do fim da tela e recoloca como disponível para o jogador.

**incrementScore**(quantity) – atualiza a pontuação do jogador com o valor recebido.

**notMove**() – tira a permissão do jogador se mover.

**move**() – Movimenta o jogador de acordo com a tecla precionada e atualiza a animação.

**animatedead**(): boolean – Realiza a animação para finalizar o jogador e devolve True para o jogo terminar.

.

### Shot

Representa uma lista de veículos da aplicação.

##### Atributos:

**\_speed\_animation** – velocidade em que a animação ocorre, quanto menor mais rápido.

**\_value** – valor usado para definir o index no array de imagens para realizar a animação de movimento do jogador.

**\_value\_shoot** – valor usado para definir o index no array de imagens para realizar a animação de movimento de tiro de um jogador.

**\_value\_dead** – valor usado para definir o index no array de imagens para realizar a animação de termino de um jogador.

**\_images** – array de imagens do jogador.

##### Métodos:

**pesquisar**(): boolean - Pesquisa a existência de um veículo com chassis ou matrícula iguais as recebidas. Assim sabe-se os valores estão disponíveis para uso e cadastro de novo veículo.

**listarEmProgresso**(): String - Lista os veículos com serviço em estado a decorrer ou concluído. Utiliza o método emProgresso() da classe Veiculo.

**listar**(Peca): String - Lista todos os veículos com uma determinada peça. Utiliza o método hasPeca(Peca) da classe Veiculo.

**listar**(Pessoa): String - Lista todos os veículos pertencente a uma determinada pessoa.

**pesquisarTempo**(int): String – Reúne uma lista dos veículos que tem o tempo superior ou igual a um valor. Utiliza o método getTempoMaximo() da classe Veiculo.

**pesquisarAno**(int): String - Reúne uma lista dos veículos que tem a data de entrada maior que um determinado ano+”0000”.

### Servico

Representa um serviço realizados pela oficina da aplicação.

##### Métodos:

**criaCodigo**(): int – Cria um valor inteiro com o formato AAAAMMDDX, onde A são algarismos de um ano, M são algarismos de um mês, D são algarismos de um dia e X um número inteiro que é a quantidade de serviços existentes na aplicação.

**hasPeca**(Peca): boolean - Verifica se uma peça está presente na lista de peças do serviço. Utiliza o método pesquisar(int) da classe GerePecas.

### GereServicos

Representa uma lista de objetos da classe Servico.

##### Métodos:

**emProgresso**(): boolean - Verifica a existência de um serviço na lista com estado a decorrer ou concluído.

**tempoMaximo**(): int - Verifica o tempo máximo despendido em um serviço na lista.

**hasPeca**(Peca): boolean - Verifica a existência de uma peça em um dos serviços da list. Utiliza o método hasPeca(Peca) da classe Servico.

## Algoritmos

Os principais e mais comuns algoritmos da aplicação foram desenvolvidos de acordo com os apresentados durante as aulas de Programação.

### Adicionar

Para adicionar um objeto a uma lista de objetos usamos o método **add** da classe **ArrayList**. Precisamos garantir anteriormente que a lista de objetos foi instanciada/inicializada.

### Pesquisar

Para pesquisar um objeto específico em uma lista de objetos, devemos receber um valor único identificador do objeto, invocar o método **enumeration** da classe **Collections** para que se organize cada objeto de maneira contável, e para cada objeto encontrado comparamos a informação recebida com a informação do objeto atual, retornando objeto encontrado ou **null** caso não se encontre objeto. Precisamos garantir anteriormente que a lista de objetos foi instanciada/inicializada.

### Remover

Para remover um objeto de uma lista de objeto é necessário fazer uso do método **pesquisar** para localizar o objeto desejado. Caso o objeto for identificado é chamado o método **remove** da classe **ArrayList**.

### Listar

Para listar todos os objetos da lista é necessário invocar o método **enumeration** da classe **Collections** para que se organize cada objeto de maneira contável, e cada objeto encontrado adicionamos em uma única String que será retornada para assim representar a lista de objetos. Precisamos garantir anteriormente que a lista de objetos foi instanciada/inicializada.

### Ordenar

Para ordenar uma lista de objetos devemos seguir os seguintes passos:

1. Garantir que o objeto é comparável, ou seja, implementa **Comparable** e contém o método **compareTo** em sua classe.
2. Precisamos garantir que a lista de objetos foi instanciada/inicializada e que tem tamanho suficiente para ordenação.
3. Invocamos o método **sort** da classe **Collections** que ordena um **ArrayList**, classe da lista.

### Quantidade

Este algoritmo tem a mesma funcionalidade e faz uso do método **size** da classe **ArrayList**.

## Procedimentos de Teste

Foram adotados diversos testes unitários não automatizados a vários componentes isolados da aplicação para a identificação de erros. Houveram testes unitários com intuito de testar cada funcionalidade da aplicação, comportamento dos métodos e classes da aplicação. E com diferentes valores de leitura, sendo algum destes não esperado inicialmente pelo programador e corrigidos ao longo da fase de testes.

# Conclusões

Após a concretização deste trabalho posso concluir que o conhecimento teórico e prático em relação a linguagem Java foram bem consolidados, contudo é necessária prática de outros componentes que não foram desenvolvidos neste projeto.

Existindo outros métodos para a realização deste trabalho, eu escolhi aplicar aquilo que foi desenvolvido em sala de aula e em consulta com o professor.

Por fim posso dizer que consegui cumprir com todos os requisitos e funcionalidades do trabalho.

Este projeto desenvolveu uma aplicação para auxiliar uma oficina de automóveis na gestão dos serviços realizados a veículos, assim como registos de peças, mecânico, clientes e veículos na sua “base de dados”. A aplicação não está pronta para acesso de vários threads, ou seja não é sincronizada para múltiplos acessos.

## Forças

O projeto foi desenvolvido a facilitar a interação com o utilizador, disponibiliza uma interface em modo texto onde o utilizador interage e controla a aplicação. As entradas de dados por parte do utilizador são testadas e protegidas contra erros ou falhas que possam surgir. Há a apresentação de mensagens informativas por toda aplicação, sabendo quais ações são realizadas com sucesso ou insucesso. Há ainda o desenvolvimento do menu e separação das possíveis ações por categorias e níveis, com intuito de facilitar a localização do utilizador na aplicação.

## Limitações

A maior dificuldade encontrada durante a solução do código foi o estranhamento a uma nova Linguagem, não ter facilidades com componentes que Java disponibiliza como objeto Date, biblioteca Time, Interface Map, entre outros.

## Trabalho Futuro

Para o futuro é necessário arriscar mais em componentes novos, desenvolver o código aplicando hierarquia e polimorfismo essencial na Linguagem Java. Estudar a manipulação de dados persistente e como otimizar o código.

# Referências

*Página da Internet:*

1. *API - Overview (Java Platform SE 8 ).* Acedido em Dezembro 2022. Disponível em <https://docs.oracle.com/javase/8/docs/api/>
2. Replit. «Programação» de Marco Veloso. Acedido em Dezembro 2022. Disponível em <https://replit.com/@mveloso/Programacao#Main.java>
3. Baeldung. (2018). *Introduction to JavaDoc*. Acedido em Dezembro 2022. Disponível em <https://www.baeldung.com/javadoc>
4. *How to Write Doc Comments for the Javadoc Tool.* Acedido em janeiro de 2023. Disponível em [https://www.oracle.com/technical-resources/articles-/java/javadoc-tool.html](https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html)
5. *Documentation with Javadoc. .* Acedido em janeiro de 2023. Disponível em <http://www.drjava.org/docs/user/ch10.html>

# Anexos

## Manual do Utilizador

### Acesso

##### Primeiro acesso

No primeiro acesso a aplicação é realizado o registo do primeiro utilizador. É pedido o login, a senha e o nome do mesmo. E então iniciada a aplicação, exibindo o menu de gestão.

O primeiro acesso acontece quando o ficheiro de dados não existe.

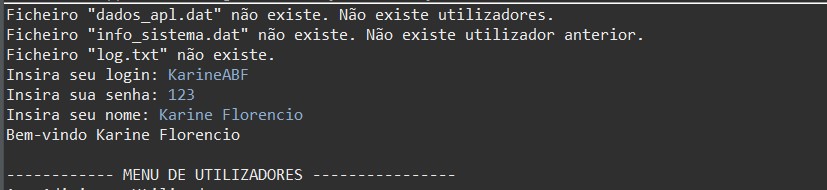


Figura 2 - Inicialização da aplicação.

##### Acessos posteriores

Nos acessos posteriores é exibido uma mensagem de sucesso de recuperação de dados e informa quantos utilizadores estão registados na aplicação, quantas execuções houveram da aplicação e o ultimo utilizador a acessar a aplicação.

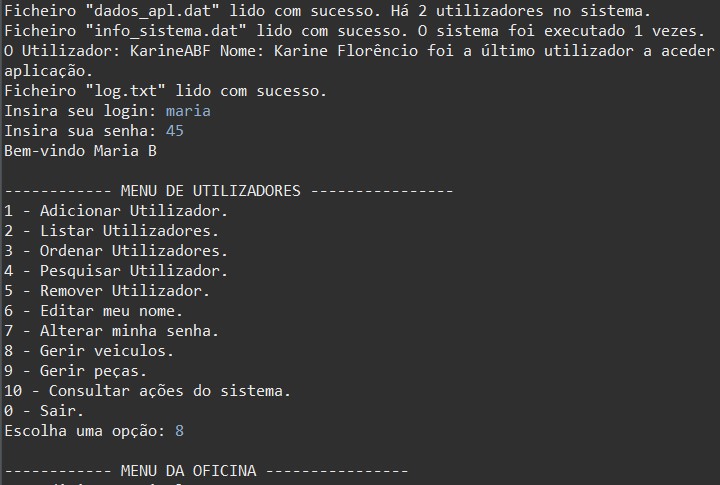


Figura 3 - Informações ao aceder a aplicação.

Na etapa de autenticação é pedido login, este deve ter sido registado anteriormente na aplicação, caso o login pertença a um utilizador é pedido a password. Se a password for incorreta ou o login não pertencer a um utilizador registado a aplicação informa o utilizador e é reiniciado a etapa de autenticação.

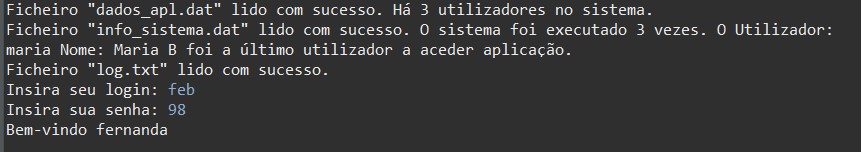


Figura 4 - Autenticação.

Estando login e password corretos é exibida a mensagem de inicialização da aplicação, seguido do menu de utilizadores.

### Menu de Gestão

O menu é organizado com uma função por opção, duas das opções levam a outro menu. Após o termino da ação escolhida pelo utilizador o menu é apresentado automaticamente, sendo encerrado apenas a escolha da opção 0 que encerra a aplicação.

##### Adicionar Novo Utilizador

A primeira opção tem a função de registar um novo utilizador na aplicação. É pedido um login que deve ser único, caso já exista um utilizador registado com este login é informado ao utilizador seguido de uma mensagem de erro, caso contrário, é pedido a password e o nome do utilizador a registar, ao sucesso da ação é exibida uma mensagem informativa.

Uma imagem com texto

Descrição gerada automaticamente

Figura 5 - Adicionar Utilizador.

##### Listar ou Ordenar Utilizadores

A opção 2 e 3 apenas exibe a lista de todos os utilizadores registados na aplicação.

Uma imagem com texto

Descrição gerada automaticamente

Figura 6 - Listar Utilizadores.

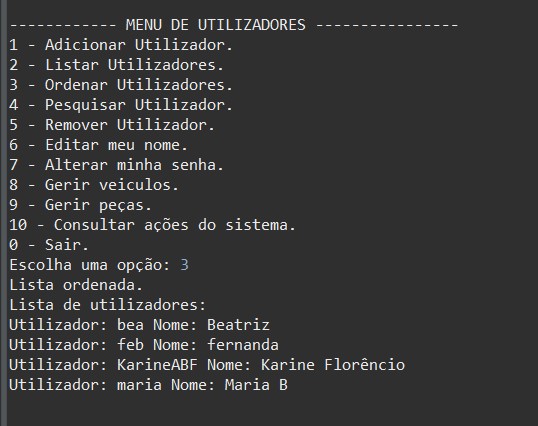


Figura 7 - Ordenar Utilizadores.

A opção 3 apresenta a lista ordenada, ou mensagem informando erro na ordenação.

##### Pesquisar Utilizador

A quarta opção tem a função de pesquisar um utilizador, é pedido o login a pesquisar e apresentado informações do utilizador pesquisado ou mensagem informando que não existe utilizador com o login digitado.

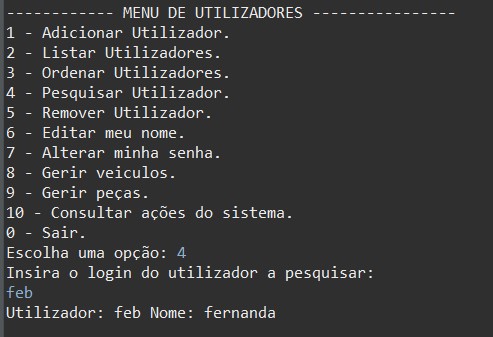


Figura 8 - Pesquisar Utilizador.

##### Remover Utilizador

A quinta opção é a ação de remover um utilizador da aplicação. É pedido o login do utilizador a ser removido, devendo ser diferente do login do utilizador logado no momento da ação. É apresentado uma mensagem quanto ao sucesso da operação.

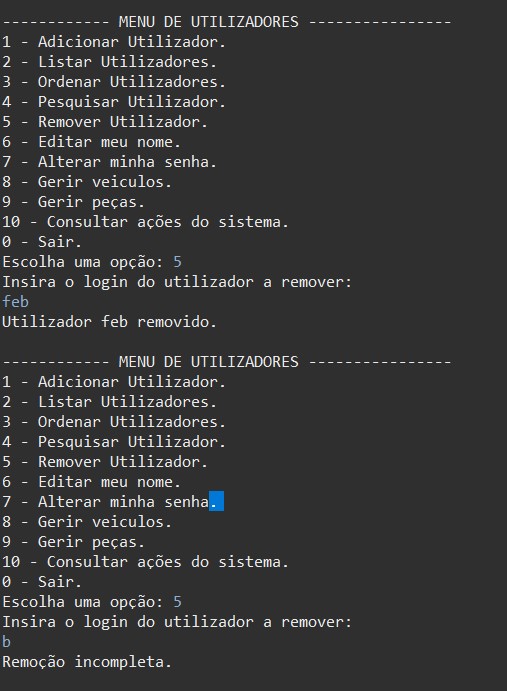


Figura 9 - Remover Utilizador.

##### Alterar nome/palavra-passe

As opções 6 e 7 são atualizações das informações do utilizador, este é permitido alterar o próprio nome ou password.

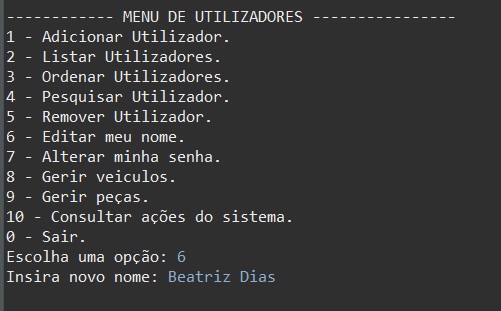


Figura 10 - Alterar meu nome.

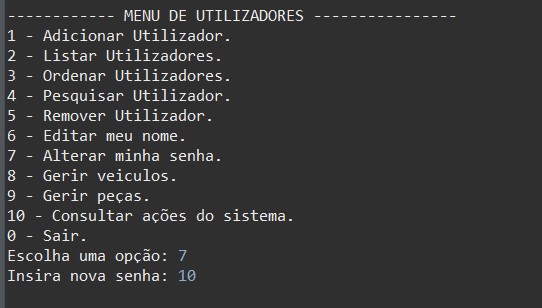


Figura 11 - Alterar minha senha.

##### Consultar ações da aplicação

A decima e ultima opção do primeiro menu exibe o log de ações, onde cada linha refere a um <Utilizador><Ação> na aplicação. Está ordenado de maneira que a log mais recente está no final da lista e o inicio da aplicação é o primeiro item listado.

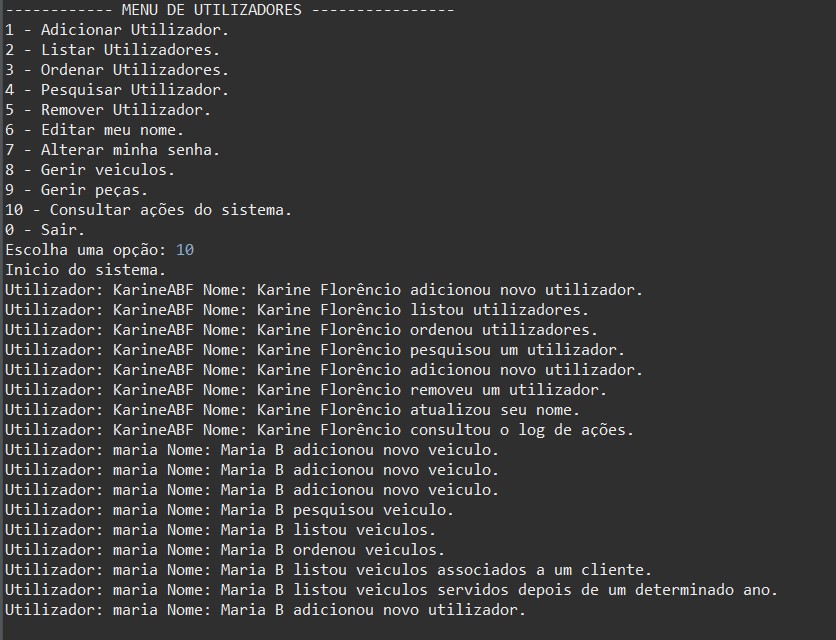


Figura 12 - Consultar ações do sistema.

### Gerir Veículos

A opção 8 do menu de utilizadores leva ao menu da oficina. O menu da oficina é organizado com uma função por opção, uma das quais levam ao menu do veículo. Após o termino da ação escolhida pelo utilizador o menu é apresentado automaticamente, sendo encerrado apenas a escolha da opção 0 que retorna ao menu de utilizadores da aplicação.

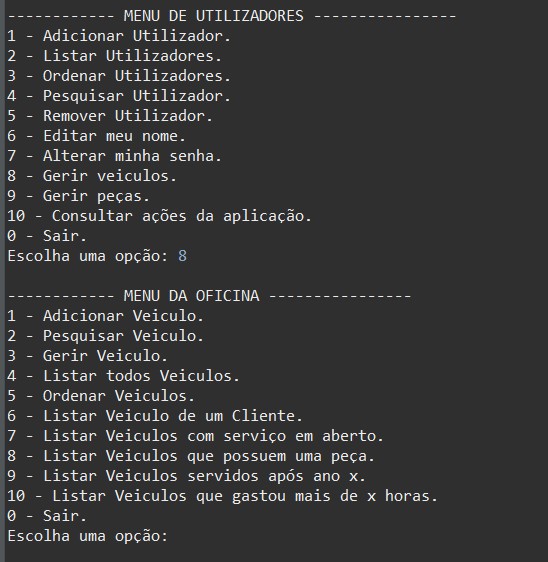


Figura 13 - Gerir Veículos.

##### Adicionar Veículo

A opção 1 do menu da oficina tem a função de adicionar um veículo ao registo de veículos da oficina. O primeiro passo desta ação é a recolha da matrícula e número de chassis do carro para a verificação se um destes já pertence a um veículo registado, caso exista é informado ao utilizador e a ação é interrompida. O segundo passo é a identificação do cliente dono do carro, é pedido o NIF do cliente e pesquisado em registo o cliente, caso este esteja em registo suas informações estão preenchidas, caso contrario é sugerido adicionar o cliente em registo e então é pedido o nome do cliente, sua morada e o telefone que só poderá ser preenchido com número não cadastrados por outras pessoas registadas. O terceiro e ultimo passo para adicionar um veículo no registo da oficina é a coleta de outros dados do veículo, como marca, modelo, ano de fabricação do veículo, data de entrada e conclusão na oficina, ao sucesso da ação é exibida uma mensagem informativa.

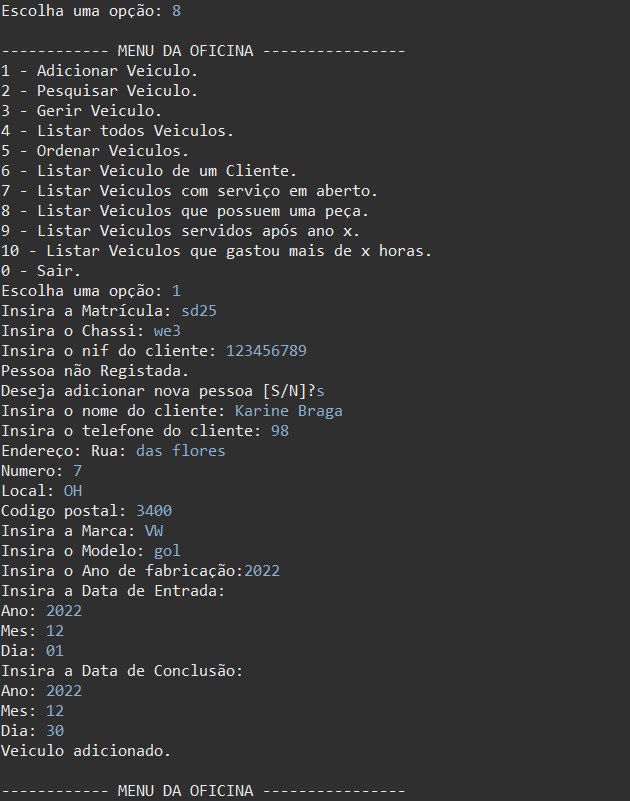


Figura 14 - Adicionar Veículo.

##### Pesquisar Veículo

A segunda opção do menu da oficina tem a função de pesquisar um veículo, é pedido a matrícula a pesquisar e apresentado informações do veículo pesquisado ou mensagem informando que não existe veículo com a matrícula digitada.

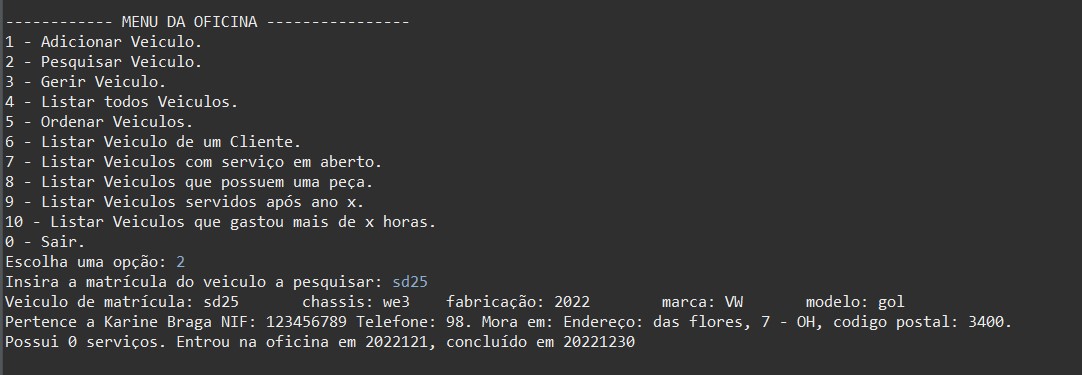


Figura 15 - Pesquisar Veículo.

##### Listar/Ordenar Veículos

A opção 4 e 5 apenas exibe a lista de todos os veículos registados na aplicação.

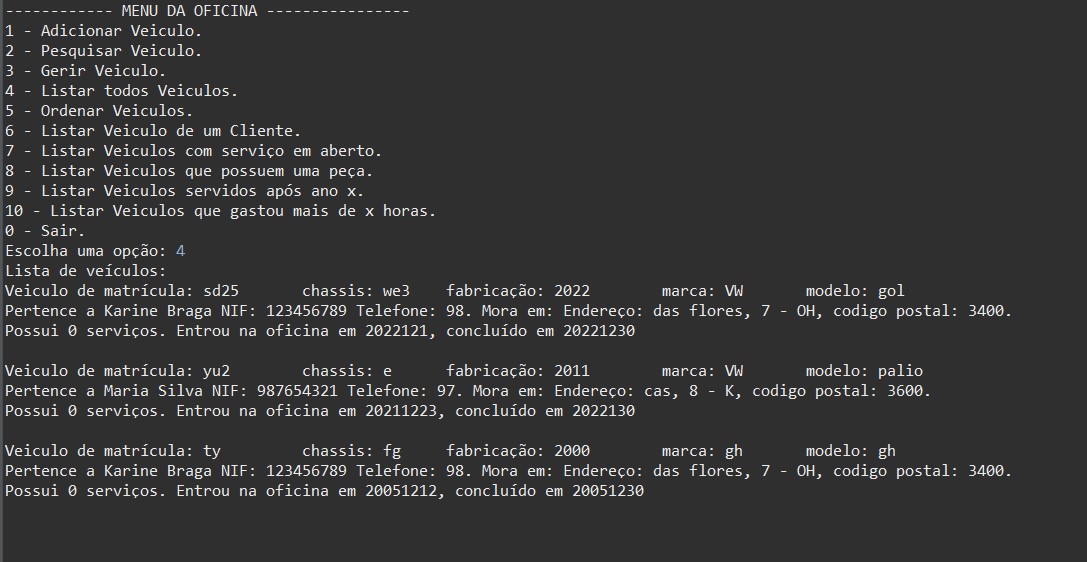


Figura 16 - Listar todos os Veículos.

A opção 5 apresenta a lista ordenada por matrícula, ou mensagem de erro na ordenação.

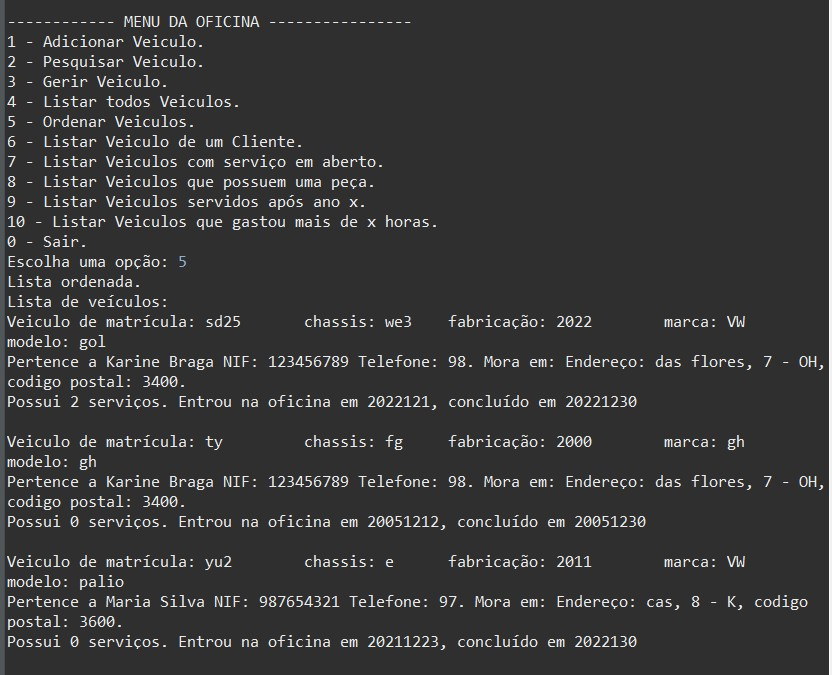


Figura 17 - Ordenar Veículos.

##### Filtrar Veículos

Existem diversas opções para filtrar a listagem de veículos, nos próximos pontos encontram se as filtrações disponíveis.

* ***Cliente***

A sexta opção lista apenas os veículos de um cliente. Portanto é necessário identificar qual o cliente é desejado para realizar a pesquisa, por isso é feito a recolha do NIF do cliente. Caso o cliente não esteja registado ou o cliente não tenha veículos registados é apresentada uma mensagem informativa.

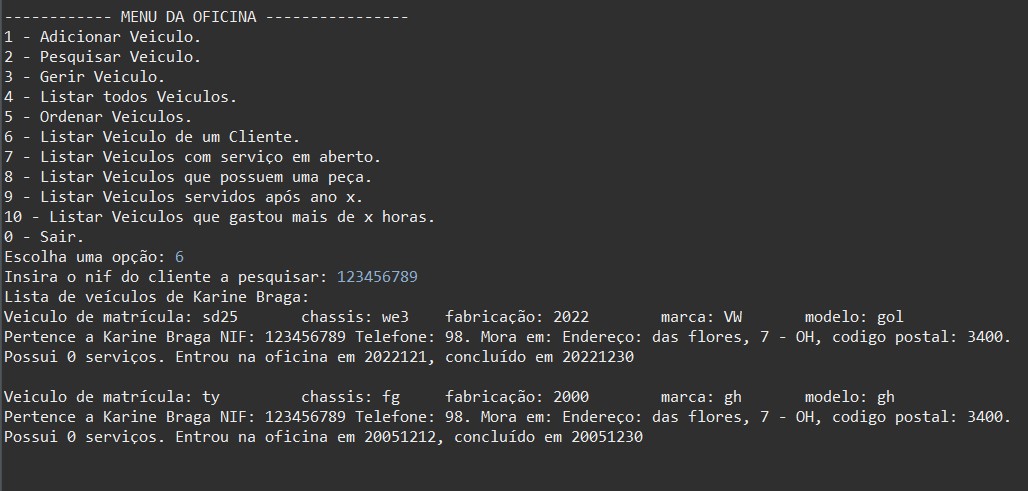


Figura 18 - Listar Veículos de um Cliente.

* ***Progresso***

A opção 7 lista apenas os veículos com serviço “a decorrer” ou “concluído”. Caso não houver veículos nesse estado é exibida uma mensagem informando o utilizador.

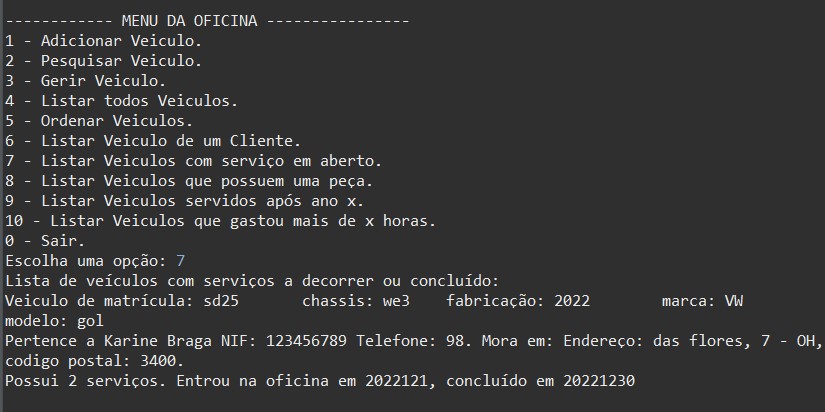


Figura 19 - Listar Veículos com serviço em aberto.

* ***Peça***

A opção 8 lista apenas os veículos que contém uma peça. Portanto é necessário identificar qual a peça é desejada para realizar a pesquisa, é pedido ao utilizador o código de identificação da peça. Caso a peça não esteja registada ou nenhum veículo registado contém a peça é apresentada uma mensagem informativa.

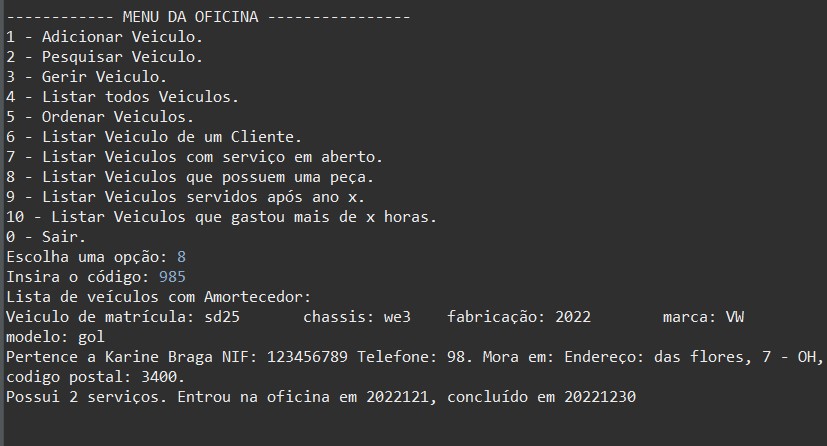


Figura 20 - Listar Veículos que possuem uma peça.

* ***Ano***

A nona opção lista apenas os veículos servidos após um determinado ano. É pedido ao utilizador o ano de interesse. Caso nenhum veículo registado tenha sido servido após o ano digitado é apresentada uma mensagem informando.

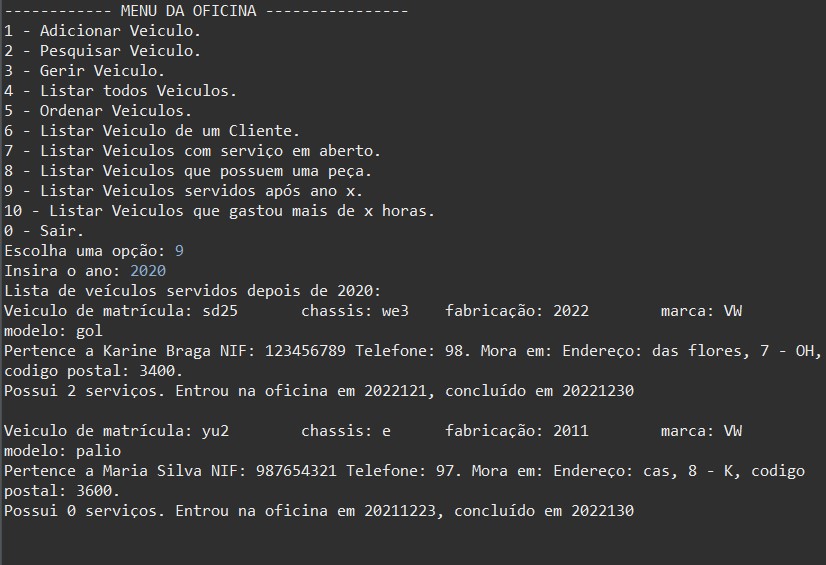


Figura 21 - Listar Veículos servidos após ano x.

* ***Tempo***

A opção 10 lista apenas os veículos que despenderam mais tempo de serviço. É pedido ao utilizador o tempo mínimo em horas para realizar a pesquisa. Caso nenhum veículo registado tenha o tempo despendido maior é apresentada uma mensagem informando.

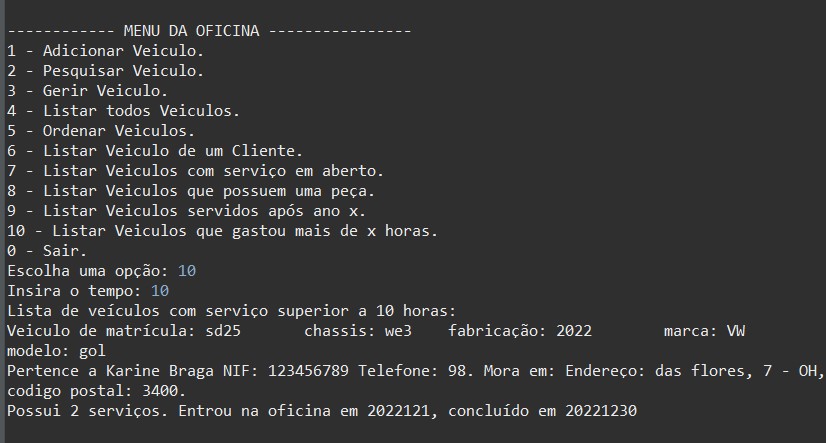


Figura 22 - Listar Veículos que gastou mais de x horas.

### Gerir Veículo

A opção 3 do menu da oficina leva ao menu do veículo, entretanto há a necessidade de identificar qual veículo iremos gerir, essa identificação ocorre através da coleta da matrícula do veículo, caso o veículo seja identificado o menu do veículo é apresentado, caso contrário é apresentada uma mensagem de erro e retorna ao menu da oficina.

Uma imagem com texto

Descrição gerada automaticamente

Figura 23 - Gerir Veículo.

O menu do veículo é organizado com uma função por opção, uma das opções levam ao menu do serviço. Após o termino da ação escolhida pelo utilizador o menu é apresentado automaticamente, sendo encerrado apenas a escolha da opção 0 que retorna ao menu da oficina .

##### Adicionar Serviço

A opção 1 do menu do veículo tem a função de adicionar um serviço ao veículo anteriormente indicado. O primeiro passo desta ação é a identificação do mecânico responsável pelo serviço, é pedido o NIF do mecânico e pesquisado em registo, caso este já tenha registo suas informações estão preenchidas, caso contrário é sugerido adicionar o mecânico em registo e então é pedido o nome, a morada e o telefone que só poderá ser preenchido com número não cadastrados por outras pessoas registadas (como exemplificado na *Figura 28 - Adicionar Mecânico.*). O segundo passo é preencher outros dados sobre o serviço, como a data, tempo despendido, custo, descrição, tipo e estado. Ao sucesso da ação é exibida uma mensagem informativa.

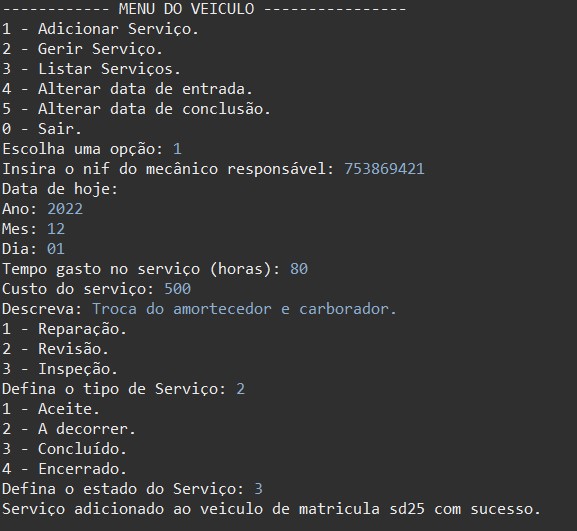


Figura 24 - Adicionar Serviço.

##### Alterar Data de Entrada/Data de Conclusão

As opções 4 e 5 são atualizações das informações próprias do veículo, este é permitido alterar a data de entrada e a data de conclusão.

##### Listar Serviços

A opção 3 do menu do veículo lista todos os serviços do veículo.

Uma imagem com texto

Descrição gerada automaticamente

Figura 25 - Listar Serviços.

### Gerir Serviço

A segunda opção do menu do veículo tem a função de gerir um serviço, logo há a necessidade de identificar qual serviço iremos gerir, essa identificação ocorre através da coleta do código identificador do serviço, caso o serviço seja identificado é apresentado as informações atuais do serviço seguido do menu do serviço, caso contrário é apresentada uma mensagem de erro e retorna ao menu do veículo.

Uma imagem com texto

Descrição gerada automaticamente

Figura 26 - Gerir Serviço.

O menu do serviço é organizado com uma função por opção. Após o termino da ação escolhida pelo utilizador o menu é apresentado automaticamente, sendo encerrado apenas a escolha da opção 0 que retorna ao menu do veículo.

##### Adicionar Peça

A opção 1 do menu do serviço tem a função de adicionar uma peça a um serviço anteriormente indicado. Portanto, é pedido ao utilizador o código de identificação da peça, se a peça existir em registo é apresentado uma mensagem de sucesso da ação, caso contrário é perguntado ao utilizador se este deseja adicionar uma nova peça ao registo, essa ação será exemplificada na secção *Adicionar Peça*. Caso a peça não for adicionada ao serviço é apresentado uma mensagem informando.

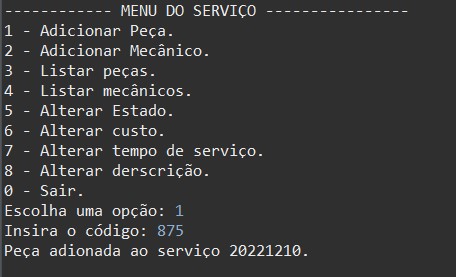


Figura 27 - Adicionar Peça ao Serviço.

##### Adicionar mecânico

A opção 2 do menu do serviço tem a função de adicionar um mecânico a um serviço anteriormente indicado. Portanto, é pedido ao utilizador o NIF do mecânico, se o mecânico existir em registo é apresentado uma mensagem de sucesso da ação, caso contrário é perguntado ao utilizador se este deseja adicionar uma nova pessoa ao registo, a operação de adicionar um mecânico é a mesma que vimos anteriormente em *Adicionar Serviço*. Caso o mecânico não for adicionada ao serviço é apresentado uma mensagem informando.

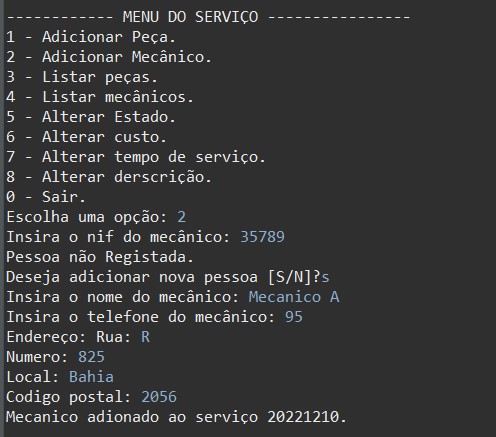


Figura 28 - Adicionar Mecânico.

##### Listar Peça/Mecânico

As opções 3 e 4 do menu do serviço tem a função de listar todos as peças ou mecânicos de um serviço de um veículo.

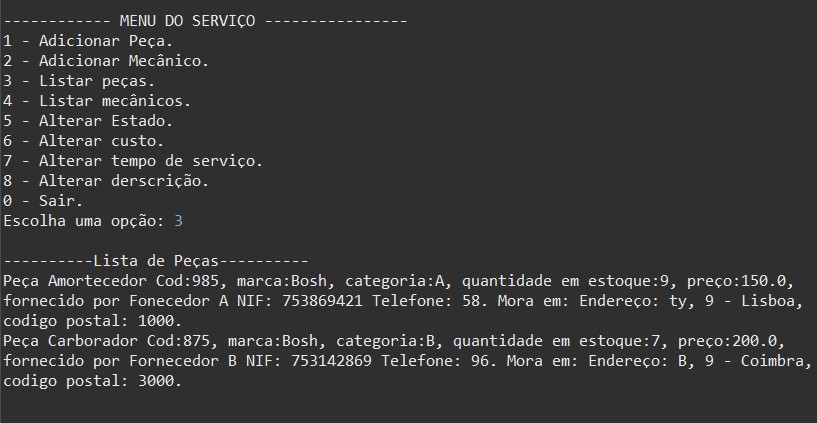


Figura 29 - Listar Peças.

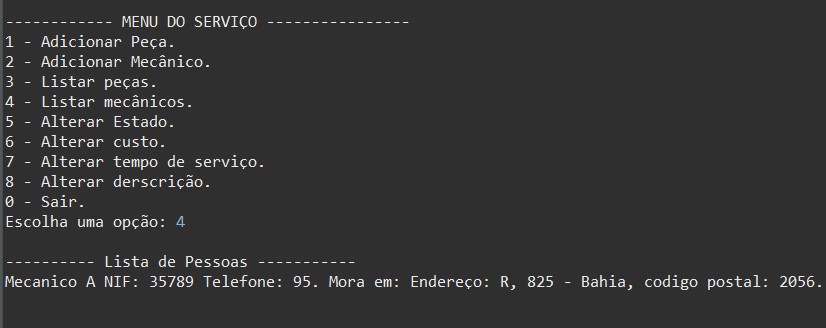


Figura 30 - Listar Mecânicos.

##### Alterar dados do Serviço

As opções 5, 6, 7 e 8 tem função de atualizar dados do serviço. Sendo eles: estado, custo, tempo e descrição respetivamente.

Uma imagem com texto

Descrição gerada automaticamente

Figura 31 - Alterar custo.

Uma imagem com texto

Descrição gerada automaticamente

Figura 32 - Alterar descrição.

### Gerir Peças

A opção 9 do menu de utilizadores leva ao menu do estoque. O menu do estoque é organizado com uma função por opção. Após o termino da ação escolhida pelo utilizador o menu é apresentado automaticamente, sendo encerrado apenas a escolha da opção 0 que retorna ao menu de utilizadores da aplicação.

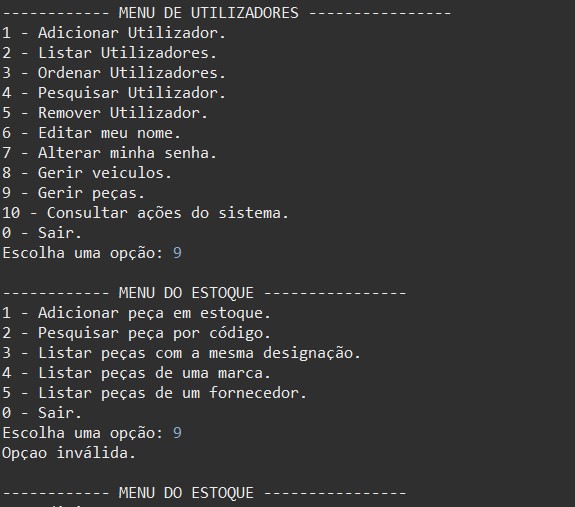


Figura 33 - Gerir Peças.

##### Adicionar Peça

A opção 1 tem a função de adicionar uma peça ao registo de peças da aplicação. O primeiro passo desta ação é a recolha do código de identificação da peça, para que se verifique se o mesmo está disponível para novo registo, caso não esteja disponível é apresentado uma mensagem de erro ao utilizador. Apos confirmação do utilizador é iniciado o processo de coleta de dados da peça, como designação, marca, categoria, quantidade em estoque, preço, e fornecedor. Para a identificação do fornecedor é pedido o NIF do fornecedor e pesquisado em registo, caso este esteja em registo suas informações estão preenchidas, caso contrario é sugerido adicionar o fornecedor em registo e então é pedido o nome, a morada e o telefone que só poderá ser preenchido com número não cadastrados por outras pessoas registadas. Ao final é apresentado uma mensagem informando o sucesso ou insucesso da ação.

Uma imagem com texto

Descrição gerada automaticamente

Figura 34 - Adicionar Peça.

##### Pesquisar Peça

A segunda opção do menu do estoque tem a função de pesquisar uma peça, é pedido o código de identificação da peça a pesquisar e apresentado informações da peça pesquisada ou mensagem informando que não existe peça com o código digitado.

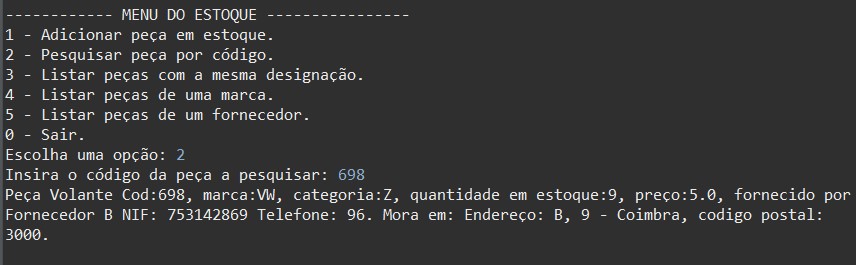


Figura 35 - Pesquisar Peça.

##### Filtrar Peças

Existem diversas opções para filtrar a listagem de peças, nos próximos pontos encontram se as filtrações disponíveis.

* ***Designação***

A terceira opção lista apenas as peças com um dado nome. Por isso é feito a recolha da designação a pesquisar. Caso não exista pecas com tal designação é apresentada uma mensagem informativa.

Uma imagem com texto

Descrição gerada automaticamente

Figura 36 - Listar peças com a mesma designação.

* ***Marca***

A opção 4 lista apenas as peças de uma marca. Para isso é feito a coleta da marca a pesquisar. Caso não exista pecas com tal marca é apresentada uma mensagem informativa.

Uma imagem com texto

Descrição gerada automaticamente

Figura 37 - Listar peças de uma marca.

* ***Fornecedor***

A quinta opção lista apenas as peças de um fornecedor. Portanto é necessário identificar qual o fornecedor é desejado para realizar a pesquisa, por isso é feito a recolha do NIF do fornecedor. Caso o fornecedor não esteja registado ou o fornecedor não tenha veículos registados é apresentada uma mensagem informativa (não é necessária a inserção de um novo fornecedor para realizar a pesquisa).

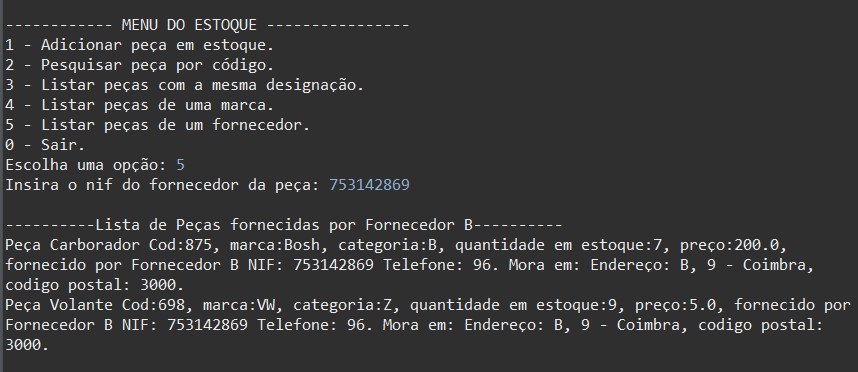


Figura 38 - Listar peças de um fornecedor.