|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Ministério da Educação e Ciência**  **Escola Secundária de Avelar Brotero**  Ano letivo 2024 / 2025  **Curso Profissional de Técnico**  **de**  **Gestão e Programação de Sistemas Informáticos**  **Projeto**  **da**  **Prova de Aptidão Profissional**  ***Tema***  Gestão de Rede de Autocarros  31 de Dezembro de 2024   |  |  |  |  | | --- | --- | --- | --- | | **Nome do Autor:** | Sofia Miranda | | | | **Ano / Turma / Número** | 12º | PSI 1 | 25 | | **Professor Orientador:** | Pedro Costa  Susana Veiga | | | |

**Gestão de Rede de Autocarros (GRA)**

Uma imagem com arte

Descrição gerada automaticamente com confiança baixa

Uma imagem com Gráficos, roxo, Lilás, Saturação de cores

Descrição gerada automaticamente

Uma imagem com símbolo, Gráficos, clipart, Tipo de letra

Descrição gerada automaticamenteUma imagem com captura de ecrã, Gráficos, file, Azul elétrico

Descrição gerada automaticamente

Uma imagem com transporte, texto, veleiro, embarcação

Descrição gerada automaticamente****Uma imagem com círculo, símbolo, logótipo, Tipo de letra

Descrição gerada automaticamenteUma imagem com captura de ecrã, Gráficos, Retângulo, design gráfico

Descrição gerada automaticamenteUma imagem com amarelo, Retângulo, Gráficos, captura de ecrã

Descrição gerada automaticamenteUma imagem com Gráficos, Retângulo, design

Descrição gerada automaticamente****

[Título 4](#_Toc199500992)

[Introdução 5](#_Toc199500993)

[Descrição do projeto 6](#_Toc199500994)

[Fases de desenvolvimento do sistema de informação 7](#_Toc199500995)

[Planeamento 7](#_Toc199500996)

[Análise 8](#_Toc199500997)

[Especificações/restrições gerais do projeto: 9](#_Toc199500998)

[Desenho 10](#_Toc199500999)

[Base de Dados 11](#_Toc199501000)

[Fluxograma 25](#_Toc199501001)

[Aplicação 26](#_Toc199501002)

[WebSite 33](#_Toc199501003)

[WebSite atualizado 42](#_Toc199501004)

[Implementação 43](#_Toc199501005)

[Testes e instalação 45](#_Toc199501006)

[Objetivos do projeto 47](#_Toc199501007)

[Interesse e aplicabilidade do projeto 48](#_Toc199501008)

[Disciplinas envolvidas 49](#_Toc199501009)

[Saberes e competências profissionais incorporados no projeto 50](#_Toc199501010)

[Fases temporais de implementação do projeto 51](#_Toc199501011)

[Horas previstas na implementação do projeto 52](#_Toc199501012)

[Recursos Humanos envolvidos 53](#_Toc199501013)

[Recursos/Materiais envolvidos 54](#_Toc199501014)

[Parcerias com empresas/ instituições 55](#_Toc199501015)

[Reflexão sobre a exequibilidade do projecto 56](#_Toc199501016)

[Reflexão final sobre o trabalho a desenvolver 57](#_Toc199501017)

[Bibliografia 58](#_Toc199501018)

[Anexos 59](#_Toc199501019)

# Título

Gestão de Rede de Autocarros.

# Introdução

Este projeto é uma implementação de uma base de dados de uma empresa de rede de autocarro, realizada em MySQL Workbench, phpMyAdmin e Visual Studio e Visual Studio Code.

Este projeto foi desenvolvido utilizando as linguagens C#, para a aplicação e HTML, CSS, TypeScript e Angular para o desenvolvimento do site. Node.js é utilizado para a parte do *back-*end de ambos os programas para a passagem de parâmetros. A aplicação vai servir apenas para administradores, onde estes podem fazer a gestão dos autocarros e dos utilizadores e o site para clientes, onde os mesmos podem adicionar bilhetes ao carrinho, comprar os bilhetes, editar dados da conta, interagir com um mapa e ver rotas dos autocarros.

Neste trabalho serão apresentados os modelos da base de dados (ER, lógico e físico, todos num só desenho), a identificação dos dados que são pedidos, a parte do código feito e explicação do mesmo que será apresentado, entre outros aspetos.

# Descrição do projeto

O projeto baseia-se num site para clientes poderem comprar bilhetes e verem as rotas dos autocarros e uma aplicação para fazer a gestão dos autocarros e de todos os utilizadores.

Na aplicação, os administradores poderão gerir os autocarros e os clientes, como foi referido anteriormente. Na gestão dos autocarros, o administrador pode adicionar um novo autocarro para utilização, editar dados do autocarro, desativar um autocarro (como fora de serviço) e criar rotas para os autocarros. A criação de rotas vai funcionar para definir a distância de um lugar para o outro, calcular a hora e talvez conseguir calcular o preço mediante estes dados. Na gestão dos utilizadores, o administrador pode criar um novo utilizador (seja ele admin ou cliente), editar dados de um utilizador, desativar um utilizador e fazer a pesquisa dos utilizadores utilizando o nome.

No site, o cliente vai poder interagir com um mapa interativo e ver as rotas dos autocarros; adicionar bilhetes de viagens ao carrinho e fazer a compra dos mesmos; visualizar as compras já feitas; editar seus dados da conta, como também desativá-la e ver os locais por qual os autocarros da empresa passam. Caso o utilizador não tenha conta, também poderá criar uma.

Como foi referido, vai haver uma interação com um mapa, no site. Essa interação vai permitir ao utilizador ver a distância de um ponto ao outro (em linha reta).

A ideia do projeto é de uma rede de autocarros que só passa por Portugal, ou seja, é nacional.

# Fases de desenvolvimento do sistema de informação

## Planeamento

Esta fase corresponde à tomada de decisão de construir o sistema de informação ou renová-lo, após ter identificado as necessidades gerais da organização e as possíveis soluções. É costume considerar nesta fase o chamado estudo de viabilidade, ou seja, uma análise dos custos e benefícios em relação ao projeto.

O trabalho apresentado é uma implementação de uma base de dados de uma rede de autocarros em um site e uma aplicação, realizada, inicialmente em MySQL Workbench, Visual Studio, Visual Code e XAMPP (PHPMyAdmin). No entanto, com a alteração da linguagem, eu passei a utilizar o WSL em vez do XAMPP porque o WSL é uma funcionalidade do Windows que permite rodar um ambiente do linux sem iniciar o sistema operativo separadamente. Ele permite que eu tenha uma parte do computador à parte, sem ser afetada por programas ou definições do Windows.

Neste projeto, vou elaborar um programa, utilizando C# para fazer a gestão de todo o projeto e HTML, PHP, CSS E JS para a interação pública do mesmo. Para os administradores conseguirem fazer uma gestão mais simples sem ter que buscar informações ao servidor, como no site, a aplicação será utilizada para isto. Com a utilização do *framework* Angular, eu passei a utilizar o TypeScript ao invés de JavaScript e PHP, que antes juntava o servidor com o cliente, foi substituído por Angular no *front-end* e Node.js no *back-end*, separadamente.

Eu decidi fazer este projeto por ser uma ideia já utilizada em um dos meus módulos de programação e ter gostado do resultado final.

## Análise

Nesta fase, foram identificadas as necessidades do sistema e os requisitos específicos para o projeto de gestão de redes de autocarros. O sistema foi feito para atender às seguintes necessidades:

1. **Administração de Utilizadores:**
   * Criar, editar e desativar utilizadores (administradores ou clientes).
   * Pesquisar utilizadores pelo nome para uma gestão mais rápida.
2. **Gestão de Autocarros:**
   * Adicionar, editar e desativar autocarros.
   * Permitir que os administradores atualizem detalhes como capacidade, partida, destino e serviços oferecidos (Wi-Fi e ar-condicionado).
   * Criar, editar e desativar rotas.
3. **Interação com o Cliente:**
   * Um site para permitir aos clientes comprar bilhetes, visualizar rotas e interagir com mapas interativos.
   * Implementação de funcionalidades de login, registo e gestão de conta.

### **Especificações/restrições gerais do projeto:**

Abaixo encontram-se tipos de interações que os utilizadores (tanto administrador quanto cliente) podem realizar.

No site, um cliente poderá:

* Criar uma conta/registo;
* Adicionar bilhetes ao carrinho e fazer a compra dos mesmos;
* Visualizar as compras já feitas;
* Visualizar as rotas dos autocarros;
* Interagir com um mapa interativo;
* Editar dados da conta;
* Desativar a conta.

Na aplicação, um administrador poderá:

* Fazer a gestão de utilizadores;
  + Pesquisar utilizadores pelo nome;
  + Visualizar utilizadores existentes;
  + Criar um novo utilizador;
  + Editar um utilizador;
  + Desativar um utilizador.
* Fazer a gestão dos autocarros;
  + Visualizar autocarros existentes;
  + Adicionar um autocarro;
  + Editar um autocarro;
  + Desativar um autocarro;
  + Criar rotas para um autocarro;
  + Alterar e desativar rotas.

A aplicação terá uma restrição para o tipo de utilizador que está a fazer o login. A restrição permite que apenas administradores tenham acesso a aplicação.

# Desenho

Nesta fase é feita as decisões mais importantes para o trabalho, onde terei que analisar o projeto e organizar as ideias para conseguir implementá-las.

Este é o meu modelo físico e lógico da minha base de dados para o projeto. Nela, contém as entidades autocarro, utilizadores, tipoutilizador, carrinho e compras.

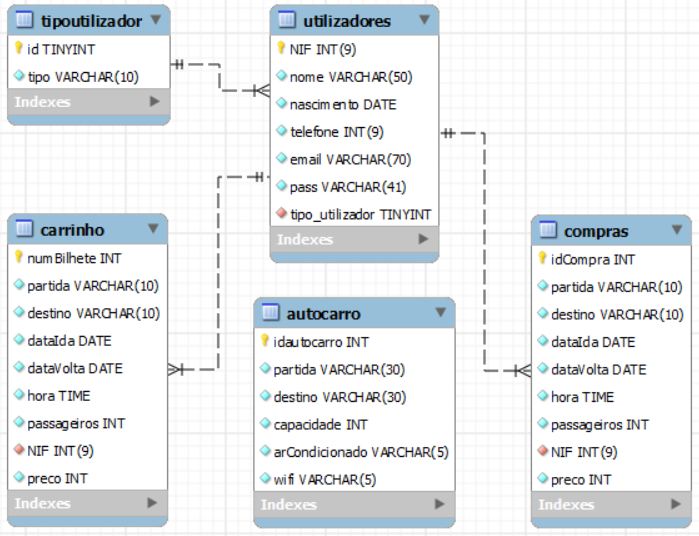


Figura 1 - Base de dados (inicial)

## Base de Dados

**Entidade - Autocarro**

IMAGEM

Figura 2 - Tabela “autocarro”.

A tabela “**autocarro**” foi criada para guardar todas as informações de um autocarro que foi inserido para estar disponível aos clientes.

* Idautocarro (INT):

Este campo é para identificar cada autocarro, como por exemplo, o NIF de uma pessoa, sendo assim chave primária e de auto incremento.

* Número (INT):

Este campo é para mostrar o número de cada autocarro para sua identificação ao circular, sendo assim, do tipo inteiro.

* Capacidade (INT):

Este campo é para dizer a capacidade do autocarro. O número de lugares que este pode ocupar, sendo assim, do tipo inteiro.

* ArCondicionado (TINYINT):

Este campo é para dizer se o autocarro tem ar condicionado ou não. Para identificar, é utilizado o tipo tinyint que apenas usa zeros e uns, sendo 0 não e 1 sim.

* WiFi (TINYINT):

Este campo é para dizer se o autocarro tem WiFi ou não. Para identificar, também é utilizado o tipo tinyint, sendo 0 não e 1 sim.

* Servico (TINYINT):

Este campo serve para mostrar o serviço do autocarro, ou seja, para dizer se ele está em serviço ou fora de serviço. Para identificar, é utilizado o tipo tinyint, sendo 0 inativo e 1 ativo.

**Entidade - Tipo de utilizador**

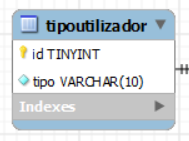
****

Figura 3 – Tabela “tipoutilizador”.

A tabela “**tipoutilizador**” foi criada para identificar se o utilizador que foi criado é administrador ou cliente.

* Id (TINYINT):

Este campo é para identificar o tipo de utilizador utilizando apenas 0 e 1. Se for 0, é administrador, se for 1, é cliente. Por isso, o campo é do tipo tinyint.

* tipo (VARCHAR(10)):

Este campo é para identificar o tipo de utilizador, contendo apenas o administrador e o cliente, como foi referido acima. Por isso este campo é do tipo varchar.

**Entidade - Atividades**

**IMAGEM**

A tabela “**atividades**” foi criada, com o mesmo propósito da tabela **“tipoutilizador”**, para identificar valores de sim/positivo/ativo e não/negativo/inativo.

* Id (TINYINT):

Este campo é para identificar o valor do campo com apenas 0 e 1. Se for 0. Por isso, o campo é do tipo tinyint.

* tipo (VARCHAR(10)):

Este campo é para identificar o valor do 0 e do 1. O zero é para qualquer valor negativo e o um para qualquer valor positivo, como foi referido acima. Por isso este campo é do tipo varchar.

**Entidade - Utilizadores**

IMAGEM

Figura 4 – Tabela “utilizadores”.

A tabela “**utilizadores**” foi feita para guardar os dados de cada utilizador, seja ele cliente ou administrador. A tabela vai guardar os dados ao fazer um registo ou a editar os dados da conta, qualquer uma das hipóteses.

* Id\_utilizador (INT 9):

Este campo é para identificar cada utilizador com um id único, para facilitar a busca de dados à base de dados, sendo assim chave primária, do tipo inteiro e com auto incremento.

* NIF (INT):

Este campo é para identificar cada utilizador, sendo assim, do tipo inteiro.

* Nome (VARCHAR(45)):

Este campo é para guardar o nome de cada pessoa, neste caso, do cliente, sendo assim do tipo varchar.

* Nascimento (DATE):

Este campo é para guardar a data de nascimento da pessoa, por isso o campo é do tipo date.

* Telefone (INT):

Este campo é para guardar o número de telemóvel da pessoa, sendo assim do tipo inteiro e limitado a nove caracteres.

* Localidade (VARCHAR 25):

Este campo é para guardar a localidade da pessoa, sendo assim do tipo varchar e limitado a vinte e cinco caracteres.

* Email (VARCHAR(70)):

Este campo é para guardar o email da pessoa, sendo assim do tipo varchar (este campo será importante).

* Password (VARCHAR(70)):

Este campo é para guardar a palavra-passe que foi definida pela pesssoa, sendo assim do tipo varchar e limitada a 41 caracteres.

* Tipo\_utilizador (TINYINT):

Este campo é para dizer qual é o tipo de utilizador. Apenas dirá se ele é administrador com o número 0 ou cliente com o número 1, sendo assim, do tipo tinyint e chave estrangeira da tabela “**tipoutilizador”.**

* Atividade (TINYINT):

Este campo é para mostrar de a conta do utilizador está ativa ou não. Para identificar, é utilizado o tipo tinyint, sendo 0 inativo e 1 ativo.

**Entidade -Carrinho**

IMAGEM

Figura 5 – Tabela “carrinho”.

A tabela “**carrinho**” foi criada para guardar os dados do bilhete que o cliente quer comprar, adicionando ao carrinho. Como é necessário ter alguma informação do cliente no bilhete, o NIF será o campo que ajudará na identificação do cliente que adicionou itens ao carrinho e posteriormente para o “recibo” da compra, quando esta estiver finalizada.

* Id\_produto (INT):

Este campo é para diferenciar os bilhetes uns dos outros, sendo assim do tipo inteiro, chave primária e com auto incremento.

* Id\_utilizador (INT):

Este campo é para identificar o os bilhetes que estão no carrinho de cada utilizador, sendo assim do tipo inteiro e chame estrangeira da entidade **utilizadores**.

* Id\_ponto\_partida (INT):

Este campo é para guardar o local de partida da viagem que a pessoa vai comprar o bilhete relacionada com a rota, sendo assim, do tipo inteiro e chave estrangeira da tabela **pontos\_rotas**.

* Id\_ponto\_chegada (INT):

Semelhante ao campo anterior, este campo é para guardar o local de destino da viagem que a pessoa vai comprar o bilhete, sendo assim, também do tipo inteiro e chave estrangeira da tabela **pontos\_rotas**.

* Preco (INT):

Este campo é para mostrar o preço total do bilhete, sendo assim do tipo inteiro. (eu ainda não decidi se este campo será do tipo inteiro ou do tipo decimal).

* Tipo\_viagem (VARCHAR 15):

Este campo é o tipo do bilhete, identificando se ele é apenas de ida ou se é de ida e volta, sendo assim do tipo varchar.

* DataIda (DATE):

Este campo é para guardar a data de ida da viagem, sendo assim do tipo date.

* DataVolta (DATE):

Semelhante ao campo anterior, este campo é para guardar a data de volta da viagem, apenas guarda caso o utilizador tenha selecionado a opção de “Ida e Volta” do site, sendo assim, também do tipo date.

* HoraIda (TIME):

Este campo é para guardar a hora em que a viagem de ida vai ser feita, sendo assim do tipo time.

* HoraVolta (TIME):

Este campo também é para guardar a hora em que a viagem vai ser feita, só que desta vez, para caso a pessoa selecione a opção “Ida e Volta” este campo guardará a hora da viagem de volta, sendo assim do tipo time.

* Passageiros (INT):

Este campo é para guardar o número de passageiros que irão na viagem, sendo assim do tipo inteiro.

Obs: Os campos “dataVolta” e “horaVolta” desta entidade não estão como NN (not null/não nulo) para caso a pessoa selecione apenas a opção de ida.

**Entidade - Compras**

**IMAGEM**

Figura 6 – Tabela “compras”.

A tabela “**compras**”, muito semelhante à tabela “**carrinho**”, também foi criada para guardar os dados bilhete que o cliente quer comprar, mas desta vez, guarda depois de ter finalizado a compra.

* Id\_compraRealizada (INT):

Este campo é para diferenciar as compras realizadas umas das outras, sendo assim do tipo inteiro, chave primária e com auto incremento.

* Id\_utilizador (INT):

Este campo é para identificar o os bilhetes que estão no carrinho de cada utilizador, sendo assim do tipo inteiro e chame estrangeira da entidade **utilizadores**.

* Id\_bilhete (INT):

Este campo vai servir para conseguir buscar os dados necessários que se encontram na tabela do carrinho (como a hora, a data e o preco do bilhete), sendo assim do tipo inteiro e chave estrangeira da tabela **carrinho**.

* Id\_ponto\_partida (INT):

Este campo é para guardar o local de partida da viagem que a pessoa vai comprar o bilhete relacionada com a rota, sendo assim, do tipo inteiro e chave estrangeira da tabela **pontos\_rotas**.

* Id\_ponto\_chegada (INT):

Semelhante ao campo anterior, este campo é para guardar o local de destino da viagem que a pessoa vai comprar o bilhete, sendo assim, também do tipo inteiro e chave estrangeira da tabela **pontos\_rotas**.

**Entidade – Pontos de rotas**

Figura - Tabela "pontos\_rotas"

A tabela “**pontos\_rotas**” foi criada para guardar locais que estarão relacionados aos autocarros criados na tabela “**autocarro**”

* Id\_ponto (INT):

Este campo é para diferenciar os locais uns dos outros, sendo assim do tipo inteiro, chave primária e com auto incremento.

* Idautocarro (INT):

Este campo é para identificar o autocarro que está relacionado a uma certa rota, sendo assim do tipo inteiro e chave estrangeira da tabela “**autocarro**”.

* Local (VARCHAR (100)):

Este campo é para identificar o nome do conselho de ida e de volta, sendo assim do tipo varchar.

* Latitude (DOUBLE):

Este campo é para mostrar a latitude do conselho em específico, de modo a identificar com mais precisão, cada um deles, sendo assim do tipo double.

* Longitude (DOUBLE):

Este campo é para mostrar a longitude do conselho em específico, de modo a identificar com mais precisão, cada um deles, sendo assim do tipo double.

* Hora\_partida (TIME):

Este campo é a hora que o autocarro relacionado irá partir de cada conselho.

Esta entidade funciona da seguinte forma:

Cada autocarro que está presente na tabela, relacionado com uma rota, mostra o seu local de destino e o seu local de chegada **final**. Cada autocarro tem, pelo menos, cinco conselhos relacionados, formando uma rota.

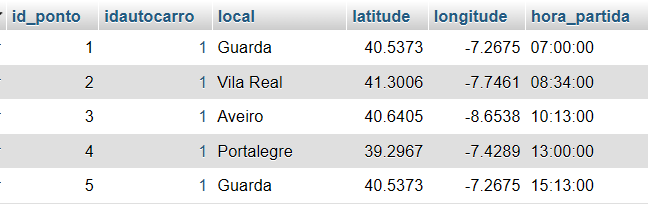


Figura - Rota do autocarro.

A rota funciona da seguinte forma:

* Primeiro local, é o início da rota, local de partida inicial.
* Os outros três locais fazem parte da rota, indo até o local de destino (o local mediano, ou seja, o do meio da rota) e voltando ao local de início.
* Último local, é o fim da rota, locar de destino.

**Tabela - pagamentos**

**IMAGEM**

A tabela “**pagamentos**” foi criada para guardar os dados de faturação do utilizador, caso ele queira manter os dados do seu cartão.

* Id\_pagamento (INT):

Este campo é para identificar cada cartão com um id único, sendo assim do tipo inteiro, chave primária e com autoincremento.

* Nome\_cartao (VARCHAR(50)):

Este campo é para identificar o nome do titular do cartão, ou seja, o nome que está associado ao cartão registado, sendo assim do tipo varchar.

* Numero\_cartao (INT):

Este campo é para identificar o número do cartão, ou seja, sendo assim do tipo int.

* Validade (VARCHAR(10)):

Este campo é para identificar a validade do cartão, sendo assim do tipo varchar.

* Id\_utilizador (INT):

Este campo é para identificar o utilizador que está associado ao cartão, sendo assim do tipo inteiro e chave estrangeira da tabela “**utilizadores**”.

Obs: Eu defini como varchar porque o tipo date grava a data com dia, mês e ano, mas as validades dos cartões possuem apenas o mês e o ano.

**Ligação à base de dados**

Aqui será mostrada como eu fiz a ligação da base de dados com a aplicação e com o site.

**Aplicação**

Na aplicação, eu criei uma classe de ligação, onde insiro as informações da aplicação (XAMPP) que vou usar, como o servidor (localhost), o user, a password (que é nula, pois não existe, por isso a variável é aspas vazia), a porta, o nome da base de dados, entre outros…

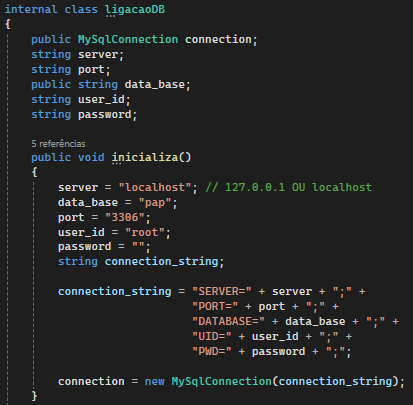
****

Figura 9 - Código da classe de ligação à base de dados.

Criei uma função booleana “open\_connection()” para abrir conexão com a base de dados. O código verifica se a conexão com a base de dados está fechada ou quebrada, tenta abri-la e, caso haja algum erro, ele exibe mensagens ao utilizador. Se a conexão for aberta com sucesso, retorna “*true*”; caso contrário, retorna “*false*”.

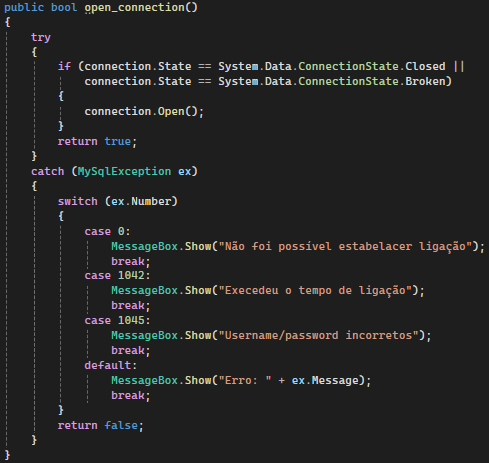


Figura 10 - Função "open\_connection()" .

Criei uma função para fechar a conexão da base de dados e um construtor para inicializar a classe. A função “close\_connection” fecha a conexão com a base de dados se esta se encontrar aberta, retorna “*true*” se for bem-sucedida e “*false”* se ocorrer uma exceção, exibindo uma mensagem de erro.

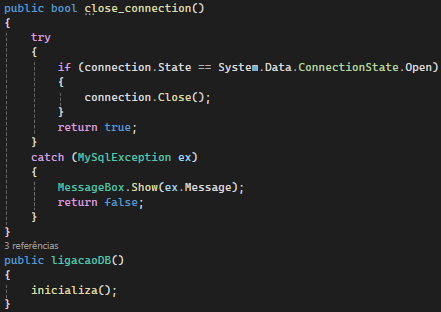


Figura 11 - Função “close\_connection” e construtor de inicialização.

**WebSite**

**Método antigo (com PHP)**

Como na aplicação, na ligação à base de dados eu insiro as informações da aplicação (XAMPP) que vou usar, como o servidor (localhost), o user e a password (que não há).

Eu guardo as informações fornecidas na variável “$con” para depois fazer a verificação da conexão. Caso esta falhe, exibe uma mensagem de erro e termina a execução do script. Criei outra variável ($escolheBD) para selecionar a base de dados (neste caso, a base de dados chamada “pap”) utilizando conexão já estabelecida com (pela variável $con). Caso esta falhe, também é exibida uma mensagem de erro e termina a execução se a base de dados não puder ser acessível.

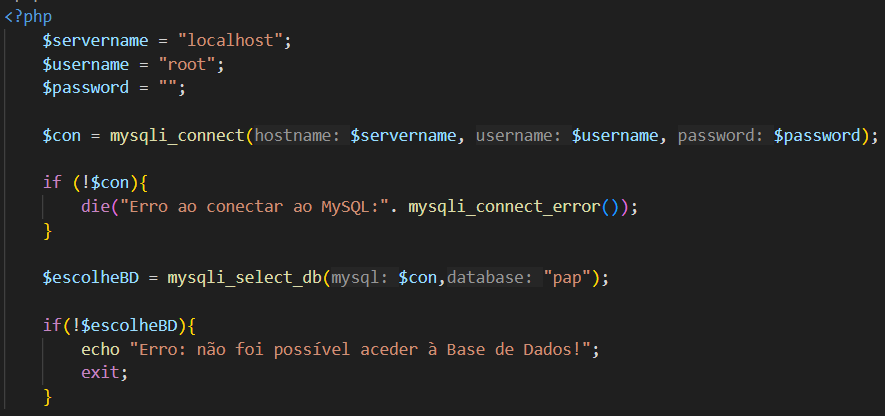


Figura 12 - Ligação à base de dados do site.

**Método novo (node.js)**

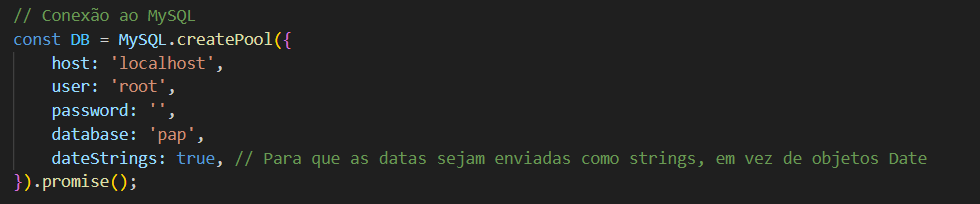


Figura - Ligação à base de dados com node.js.

Aqui é feita a ligação à base de dados quase da mesma maneira que era feita antes, em PHP.

Eu também criei um código para configurar o *middleware* de sessão. Esse código permite que o website **guarde sessões de utilizadores na base de dados,** em vez de apenas na memória, o que é mais seguro e persistente.

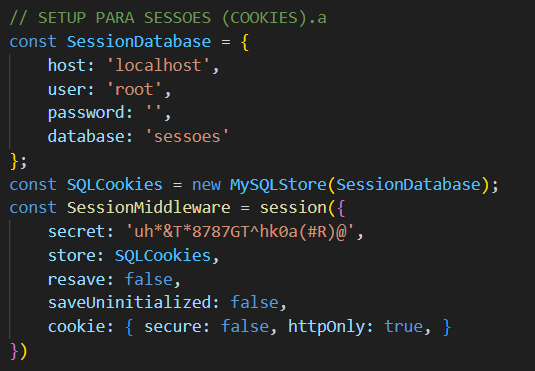


Figura - Configuração do middleware de sessão.

## Fluxograma

A seguir é apresentado um fluxograma para explicar como o sistema de login na aplicação funciona.

Uma imagem com texto, captura de ecrã, diagrama, Tipo de letra

Descrição gerada automaticamente

Figura 15 – Fluxograma do sistema de login da aplicação.

## Aplicação

Nesta fase, será apresentada as interações que a pessoa com acesso de administrador consegue realizar na aplicação. No entanto, a aplicação ainda não está finalizada.

Para o projeto, eu decidi criar uma marca chamada “**AutoBus**” para a companhia de autocarros.

**AutoBus**

Estrutura do código:

O código, no momento, está dividido em quatro partes, que constituem quatro páginas de de forms com mais uma classe que é a classe de ligação à base de dados (porém, o número estimado de páginas são seis).



Figura 16 - Páginas e classe da aplicação

Agora, com a aplicação finalizada, ela está dividida em seis partes com “sub-partes” entre elas, sem incluir a página de ligação à base de dados e as outras duas classes de busca de dados dos autocarros e dos utilizadores.

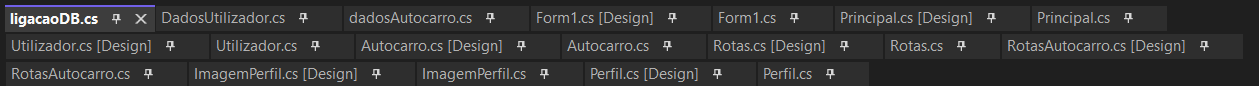


Figura - Páginas e classe da aplicação.

No início do programa, ao iniciá-lo, é nos mostrado o primeiro forms com uma página

de login. Ao fazer o login, aparece-nos o form principal. Na página principal, o administrador consegue escolher entre gerir os utilizadores ou os autocarros.

**Form 1 – Login**

“Form1” é a primeira página com a qual o utilizador começa a interagir na aplicação.

Uma imagem com texto, captura de ecrã, diagrama, file

Descrição gerada automaticamente

Figura 18 - Menu de login.

O utilizador tem um menu de login, onde ele tem de inserir o email e password para conseguir entrar e interagir com a página principal. Caso ele não preencha tudo ou deixe algum espaço em branco, aparecerá uma mensagem de erro “Todos os campos devem ser preenchidos” e terá de preencher novamente;

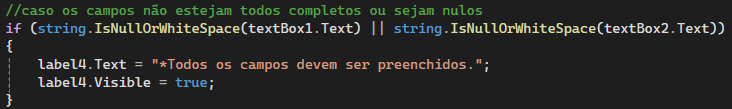


Figura 19 - Código da parte de verificação do login (espaço branco ou nulo)

Caso a password esteja incorreta, outra mensagem de erro “Password incorreta”;

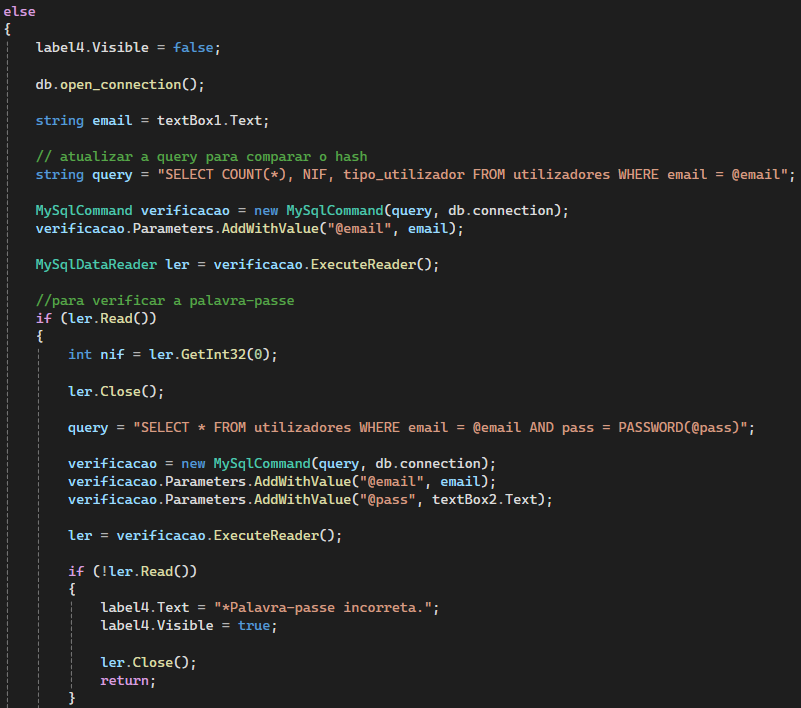


Figura 20 - Verificação da palavra-passe.

Caso o email esteja incorreto ou a conta não exista, outra mensagem de erro “Email está incorreto ou a conta não existe”. Caso tente entrar com uma conta que seja de um utilizador cliente, aparece outra mensagem de erro a dizer que é apenas para administradores;



Figura 21 - Verificação do email e do tipo de utilizador.

**Principal - Página principal**

Esta é a página principal, a qual o administrador tem a maior interação no programa.

*Uma imagem com texto, veículo, Veículo terrestre, roda

Descrição gerada automaticamente*

Figura 22 - Página principal.

Nesta página, o utilizador tem as seguintes interações iniciais: sair e aceder ao menu (como mostrado na imagem acima). Ao aceder ao menu, o administrador pode selecionar utilizadores ou autocarros.

Uma imagem com texto, veículo, Veículo terrestre, Meio de transporte

Descrição gerada automaticamente

Figura 23 - Menu.

**Utilizadores**

Ao selecionar esta opção, o administrador vai poder fazer a gestão dos utilizadores. Será possível criar um novo utilizador, editar os dados de um utilizador ou desativar um utilizador. Também é possível fazer pesquisa de utilizadores ao pesquisar pelo nome.

Uma imagem com texto, captura de ecrã, software, diagrama

Descrição gerada automaticamente

Figura 24 – Opção “Utilizadores”.

Na parte de editar os utilizadores, os únicos campos que são inalteráveis são: o NIF da pessoa, o email e a palavra-passe. Visto que o NIF é a chave primária e tem que ser única e o email e a password são necessárias para efetuar o login.

**Autocarros**

Ao selecionar esta opção, o administrador vai poder fazer a gestão dos autocarros. Será possível adicionar um novo autocarro (para que ele esteja disponível para utilização), editar os dados de um autocarro ou desativar um autocarro (para dizer que está fora de serviço).

*Uma imagem com texto, captura de ecrã, número, software

Descrição gerada automaticamente*

Figura 25 – Opção “Autocarros”.

Quando selecionado qualquer uma das opções, aparece outras opções de gestão dentro do “tema” escolhido, seja ele utilizadores ou autocarros. Ao clicar, por exemplo, em editar é aberta uma janela (form) com os campos do utilizador/autocarro para editar. Foi criado uma página para cada opção, como foi referido anteriormente.

As páginas criar e editar tem dois painéis cada. Um é para os utilizadores e o outro para os autocarros. Caso esteja selecionada a opção dos autocarros, por exemplo, e o administrador clicar em editar, o painel dos autocarros ficará visível para administrador conseguir fazer a edição dos dados. Isto funciona para o “criar” também.

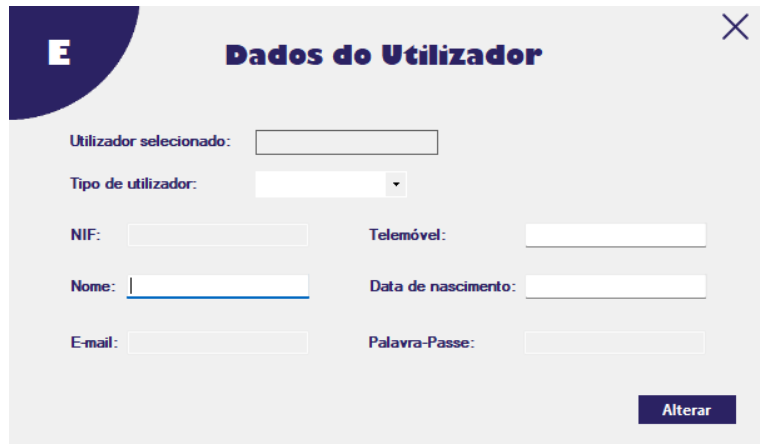
**

Figura 26 - Editar "autocarros" .

Figura 27 - Editar "utilizadores"

Cada painel tem uma letra no canto superior esquerdo para identificar se está na página de editar ou criar, já que ambas são muito parecidas, fiz isso para distinguir pois pensei que seria mais fácil e assim não haveria confusões.

Figura 28 - Identificador de página (editar). Figura 29 - Identificador de página (adicionar/criar).

## WebSite

Durante o desenvolvimento do projeto, optei por refazer completamente o website utilizando tecnologias, ou seja, linguagens e *frameworks* mais modernos. A versão inicial do site foi criada com HTML, CSS, JavaScript e PHP, como será apresentado a seguir, com *printscreens* e explicações detalhadas. No entanto, posteriormente, desenvolvi uma nova versão utilizando Angular para o *front-end* e Node.js no *back-end* que comunica com a base de dados através de *end-points* definidos mantendo a base de dados em MySQL e a estrutura do site em HTML e CSS. Esta decisão permitiu uma separação clara entre cliente e servidor, maior organização do código e uma experiência de utilização mais fluida.

**WebSite (versão antiga (PHP))**

No site, temos a página principal, que é comum para quando qualquer pessoa entra no site e será um pouco modificada para quando alguém faz um login (as alterações ainda estão em processo).

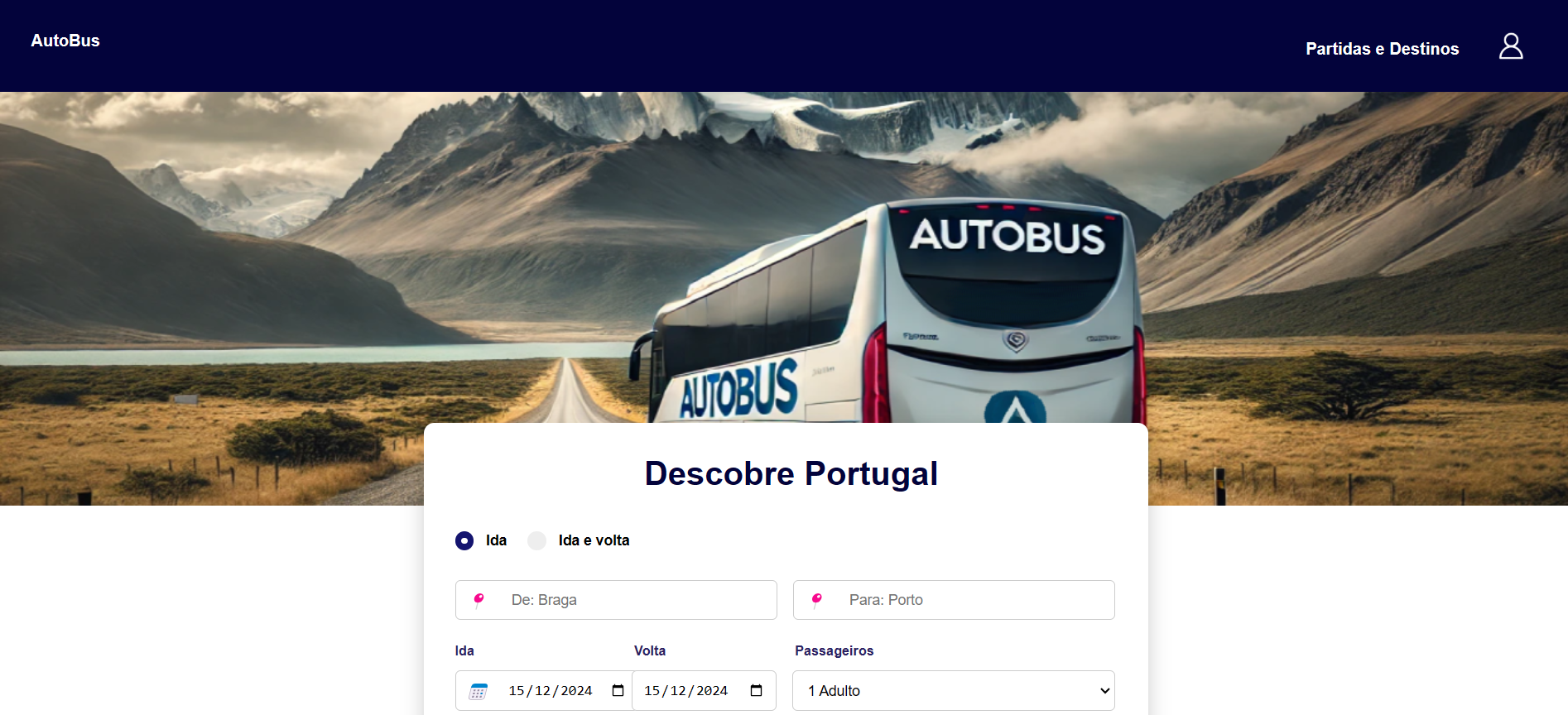
**

Figura 30 - Site (parte principal).

Ao clicar no icon de utilizador, aparece uma caixa com um pequeno formulário para preencher o email e a password, caso o utilizador já tenha uma conta criada.

** **

Figura 31 - Ícone do botão para fazer o login. Figura 32 - Formulário de login.

Caso a pessoa não tenha uma conta, ela pode criar uma ao clicar em “Criar uma” e é redirecionada à página do formulário de registo.

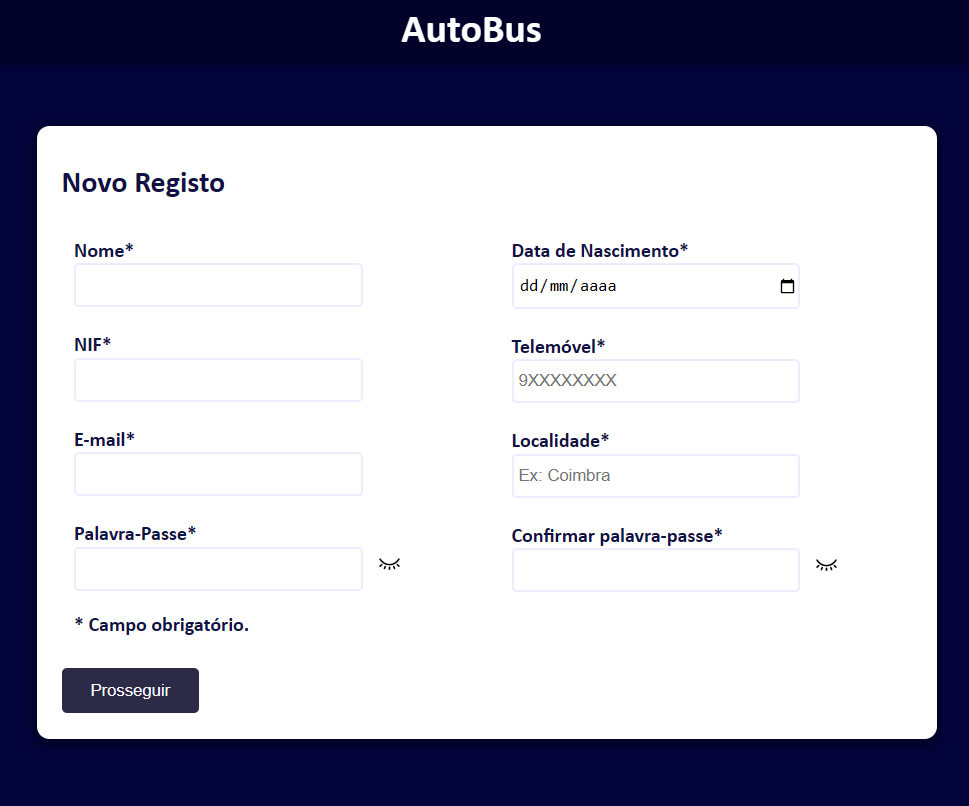
**

Figura 33 - Formulário de registo.

Quando preenchido o formulário, a pessoa é redirecionada novamente para a página principal. Após o login feito, a pessoa já pode efetuar compras, ver o carrinho, editar dados da conta e coisas do género.

Para o formulário de registo, eu criei diversas restrições. No geral, as validações são: nome não poder conter números ou caracteres especiais; número de telemóvel e nif não podem conter letras e nem caracteres especiais; não poder ser menor de 18 anos (por ser um site em que se pode efetuar compras, não é recomendado ser para menores de 18 anos); o email tem que ter o formato necessário e as validações da palavra-passe.

Como foi referido, eu fiz algumas validações porém maior parte delas foi feita em HTML utilizando o atributo “pattern” (embora ele não seja 100% eficaz, eu utilizei para validações simples).

A validação da password, por ser a mais complexa, eu tive que fazer em JavaScript e fiz também em HTML. Para esta validação, eu fiz uma “caixa” animada para verificar em “tempo real”.

Abaixo, encontram-se o código todo que foi feito para a parte da palavra-passe. Tanto a parte da validação quando a parte da animação da caixa, *scroll* e visualizar a password.

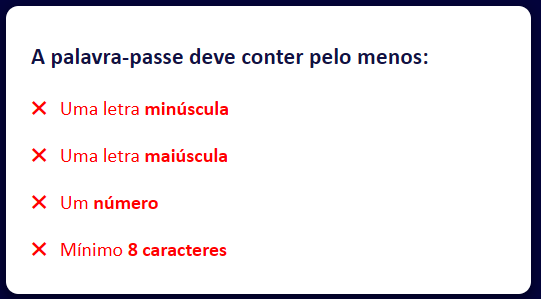
**

Figura 34 – Caixa de confirmação da palavra-passe.

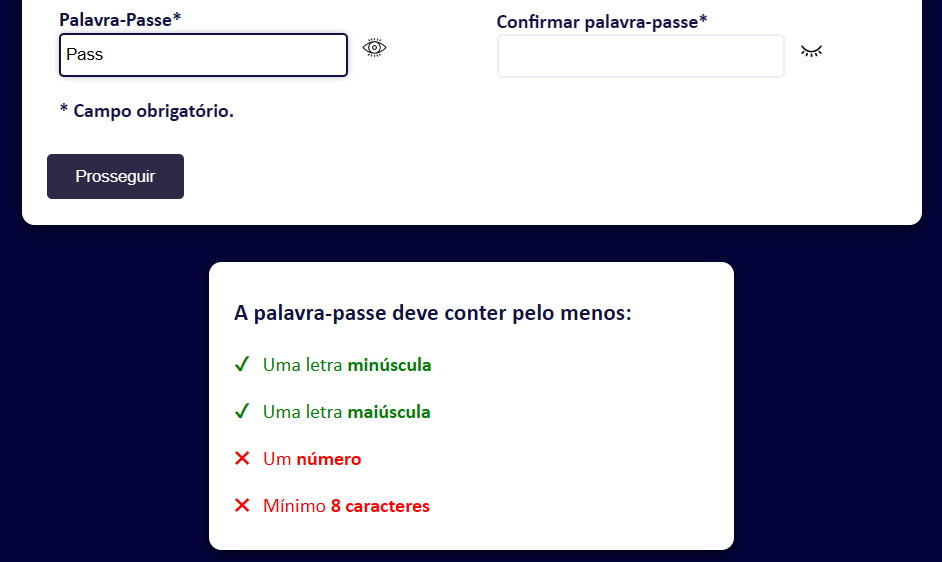


Figura 35 - Palavra-passe a ser confirmada.

Como as outras validações são iguais (a validação de conter número e de ter no mínimo 8) eu só coloquei o *printscreen* da letra maiúscula e minúscula, mas todas funcionam da mesma forma.

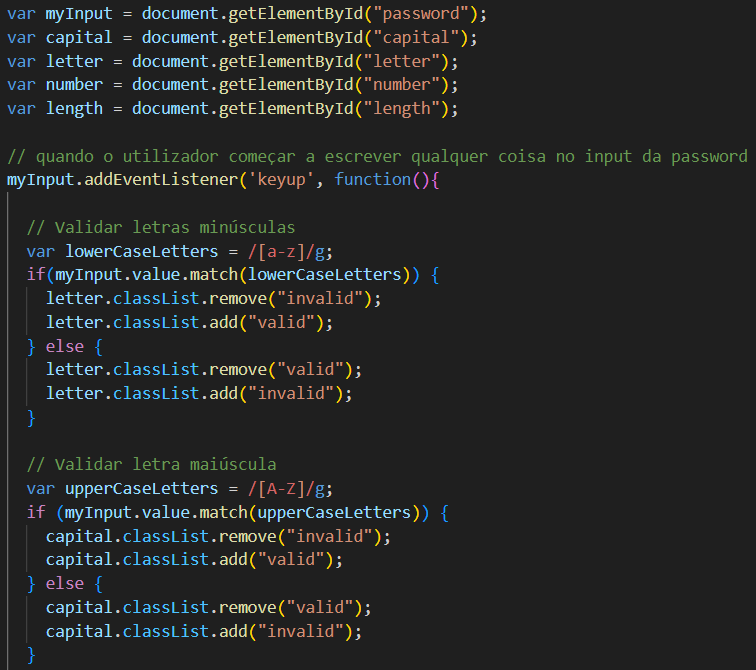
**

Figura 36 – Validação em “tempo real”.

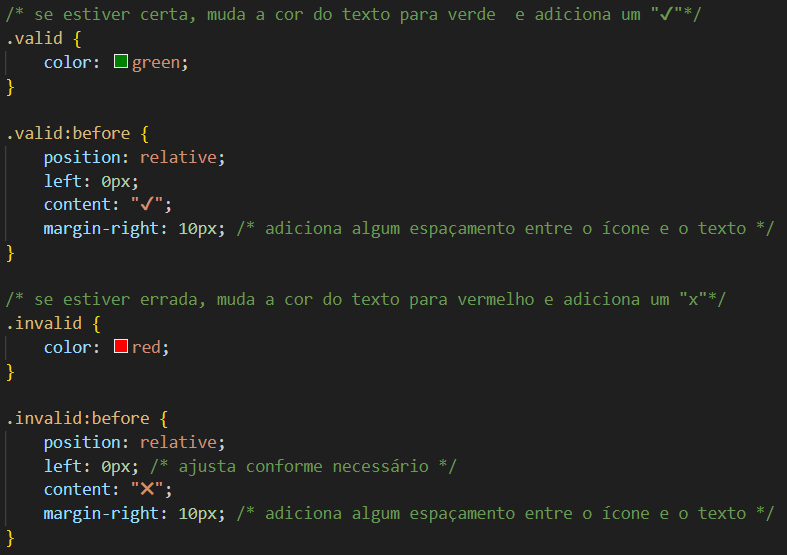


Figura 37 - Parte do estilo da confirmação da palavra-passe (CSS).

Nas imagens acima, a caixa de validação está abaixo da caixa do formulário. Isto porque, ao clicar na parte de inserir a palavra-passe a caixa aparece em baixo com uma pequena animação de *scroll* para baixo, como foi referido anteriormente.



Figura 38 - Função da animação de scroll.

Criei também dois botões para visualizar a password.

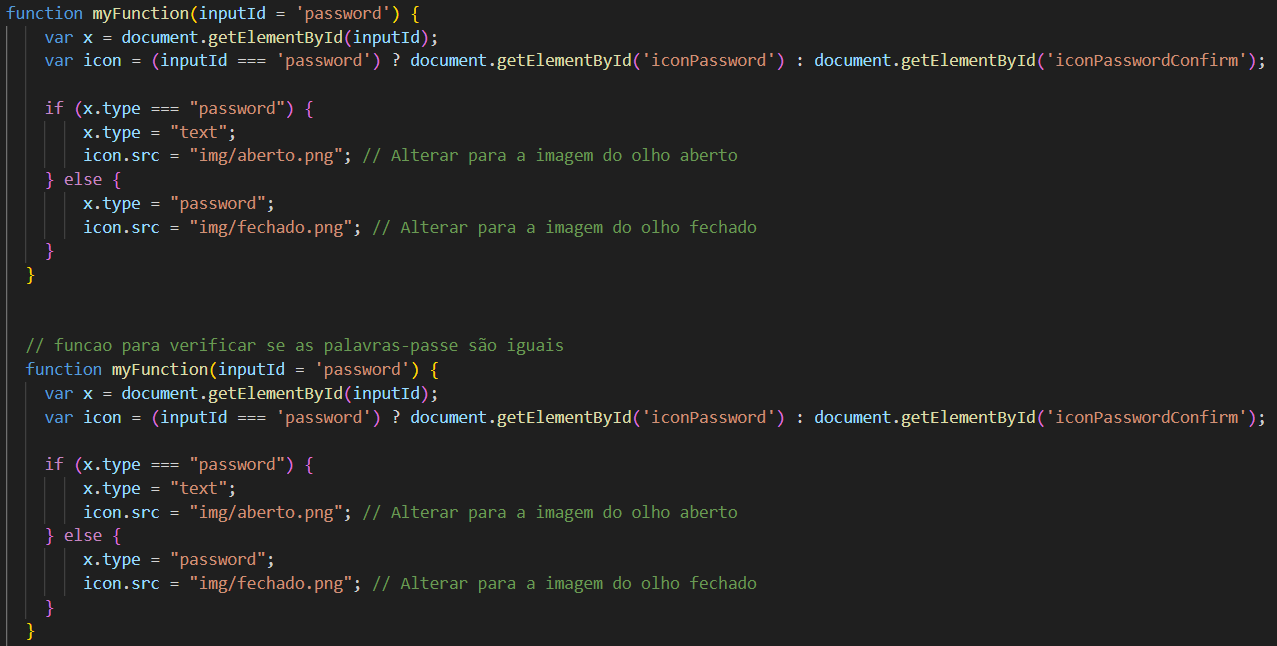
**

Figura 39 - Visualizar a palavra-passe.

Eu também criei uma caixa de mensagem de pop-up personalizado. Esta serve para dizer se as palavras-passe coincidem ou não. Caso estejam diferentes, é mostrada a caixa de mensagem.



Figura 40 – Mensagem de palavras-passe diferentes.

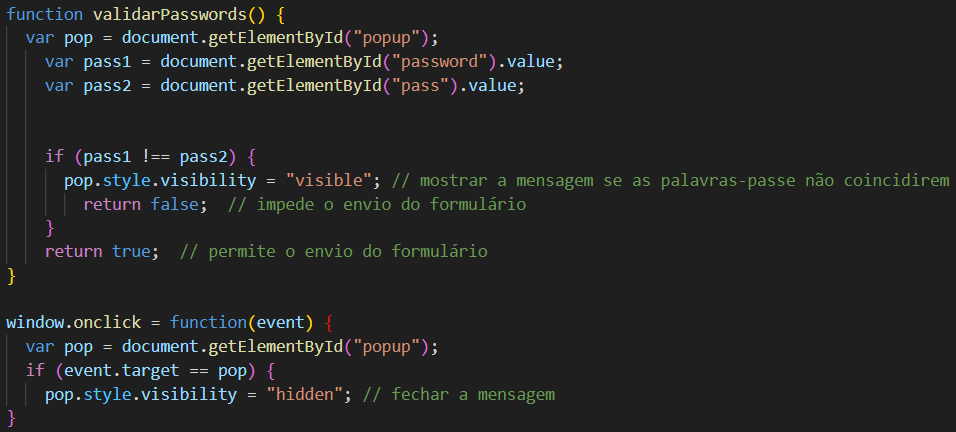


Figura 41 – Código da caixa de mensagem.

Para eu conseguir inserir a password encriptada na base de dados, tive que fazer manualmente no código para que, quando criado o registo, a password esteja logo encriptada.

Também tive que definir o tipo de utilizador (cliente ou administrador) porque toda a conta que é criada no site, é conta cliente.

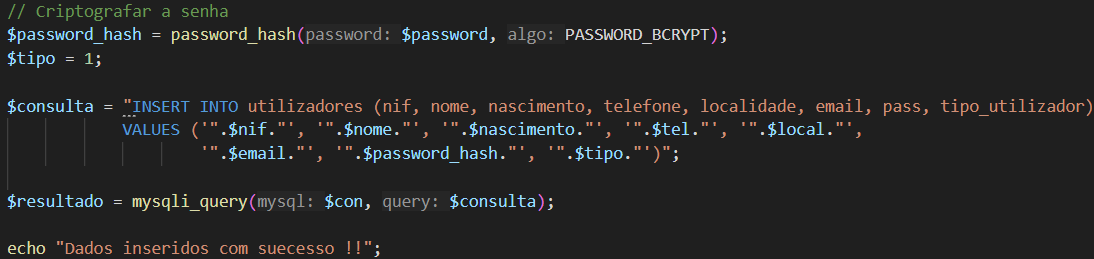


Figura 42 - Password encriptada e definição do tipo de utilizador.

A validação da idade, uma coisa que também é bem importante, foi um pouco complexa a se fazer. Eu inseri um input do tipo date para a pessoa inserir a data de nascimento e mediante isso fazer o cálculo do ano em que estamos menos a data de nascimento da pessoa para obter a idade dela. Caso a pessoa tenha menos de 18 anos, não será possível criar um registo no site.

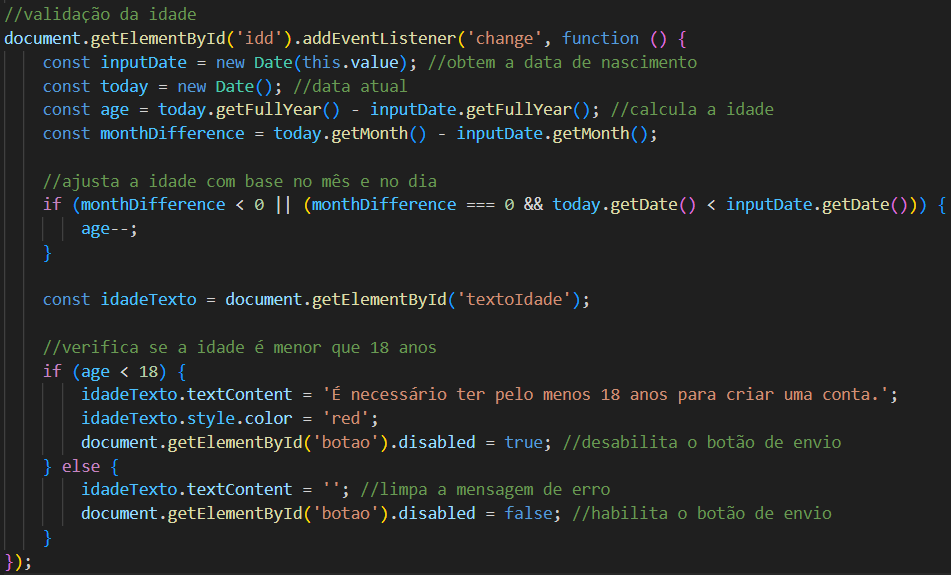


Figura 43 - Validação da idade em JavaScript.

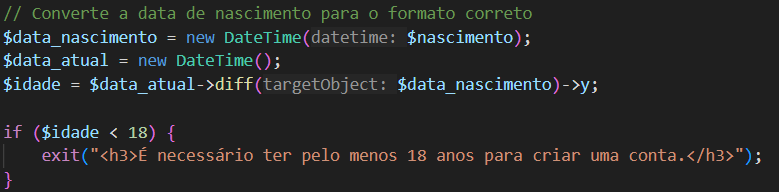


Figura 44 - Validação da idade em PHP.

No canto superior direito do site, também encontramos um botão de “Partidas e Destinos”. Ao clicar neste botão, será mostrado os diversos lugares/cidades por onde os autocarros desta companhia passa. No entanto, ainda não está feito.

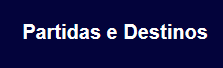


Figura 45 - Botão de "Partidas e Destinos".

### **WebSite (versão recente (Angular e Node.js))**

Aqui será apresentada a versão mais recente e final do site, ou seja, já com a alteração de linguagens e frameworks.

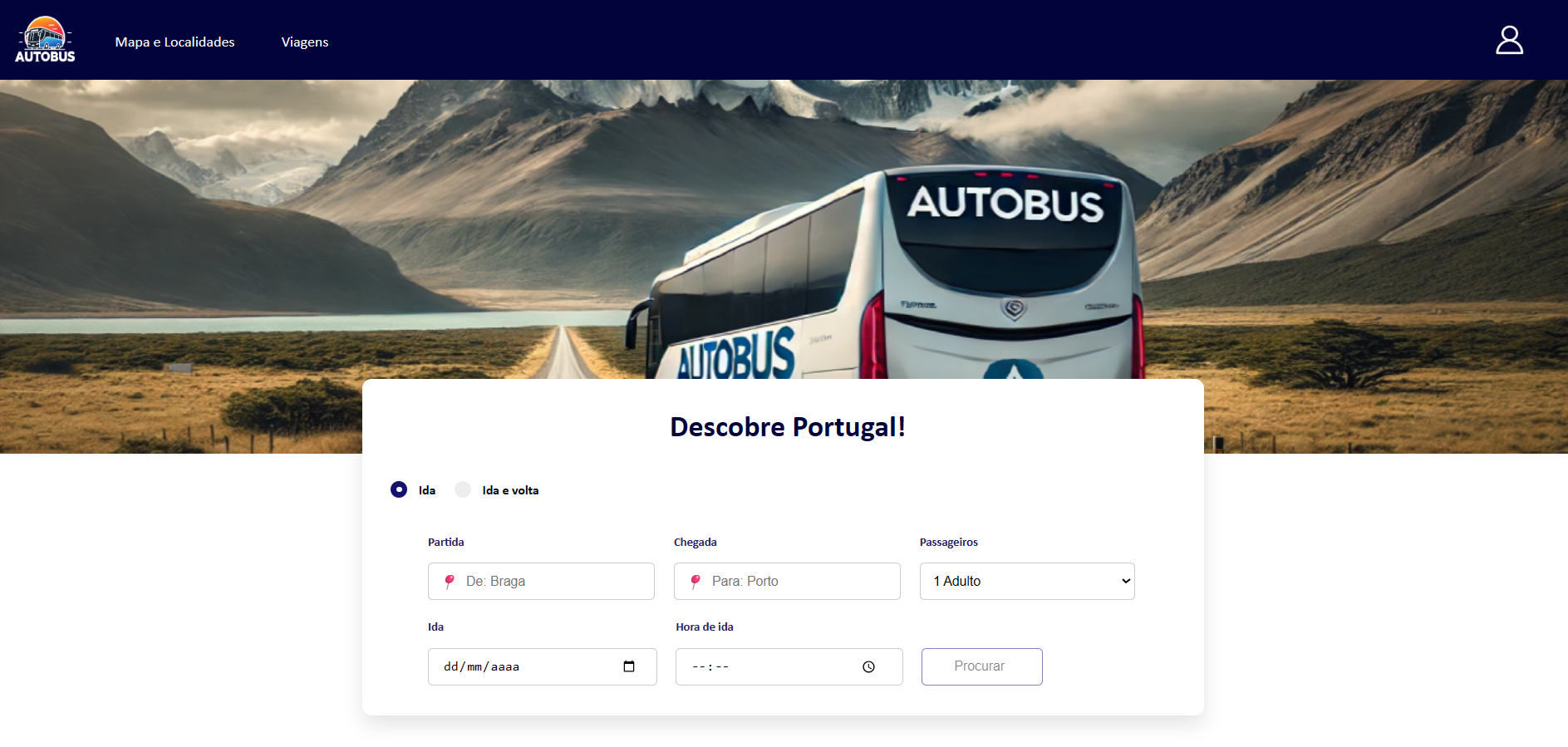


Figura - Página principal e inicial do website.

A parte principal do site é assim, como mostra a imagem. No canto superior esquerdo é possível ver opções de interação como “Mapas e localidades” e “Viagens” que são duas páginas, uma com informações do site e o mapa interativo e a outra com a gestão da compra dos bilhetes. Já no canto superior direito, um ícone de utilizador para efetuar o login.

Como foi explicado anteriormente (já que as funcionalidades são quase as mesmas), ao clicar no ícone de utilizador, aparece uma caixa com um pequeno formulário para preencher o email e a password, caso o utilizador já tenha uma conta criada.



Figura - Formulário de login.

Eu criei uma restrição para caso a palavra-passe ou o email estiverem errados, aparece uma mensagem de erro.

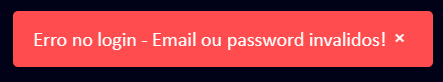


Figura - Mensagem de erro.

Para a mensagem de erro eu criei um componente, já que ela não é utilizada apenas nesta parte de login, para conseguir utilizá-la em outras páginas sem ter o trabalho de estar a criar novas caixas de mensagem.

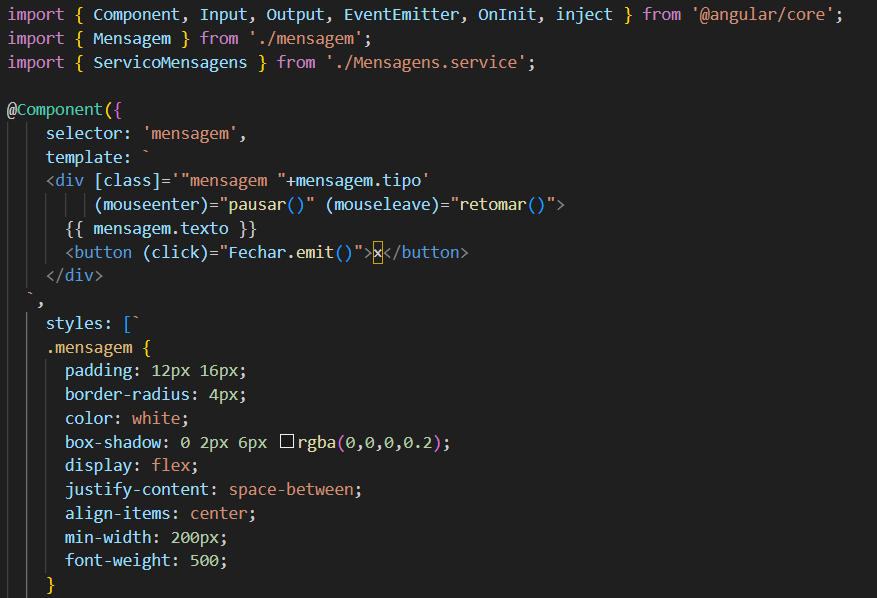


Figura - Parte do componente da mensagem.

**Este componente serve para mostrar mensagens temporárias** (tipo alertas de sucesso, erro, etc.), com a opção de fechar manualmente e desaparecer sozinho após 3 segundos.

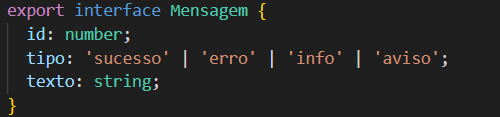


Figura - Estrutura da mensagem



Figura - Temporizador da mensagem

Do mesmo jeito que fiz um componente para a mensagem, eu também fiz outros componentes pra outras funcionalidades que são reutilizadas em outras páginas ou muito complexas (por mais que seja apenas utilizada em uma, como por exemplo, o mapa). Como por exemplo para a topbar, o footer, o formulário de pesquisa de viagens que é utilizado em duas páginas, entre outros.

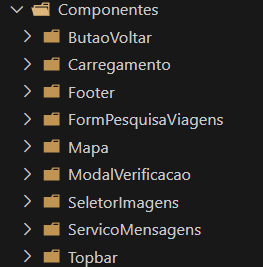


Figura - Componentes criados para o website.

Caso a pessoa não tenha uma conta, ela pode criar uma ao clicar em “Criar uma” e é redirecionada à página do formulário de registo.

A página de criar conta tem as mesmas funcionalidades que havia antes, no entanto, com node.js.

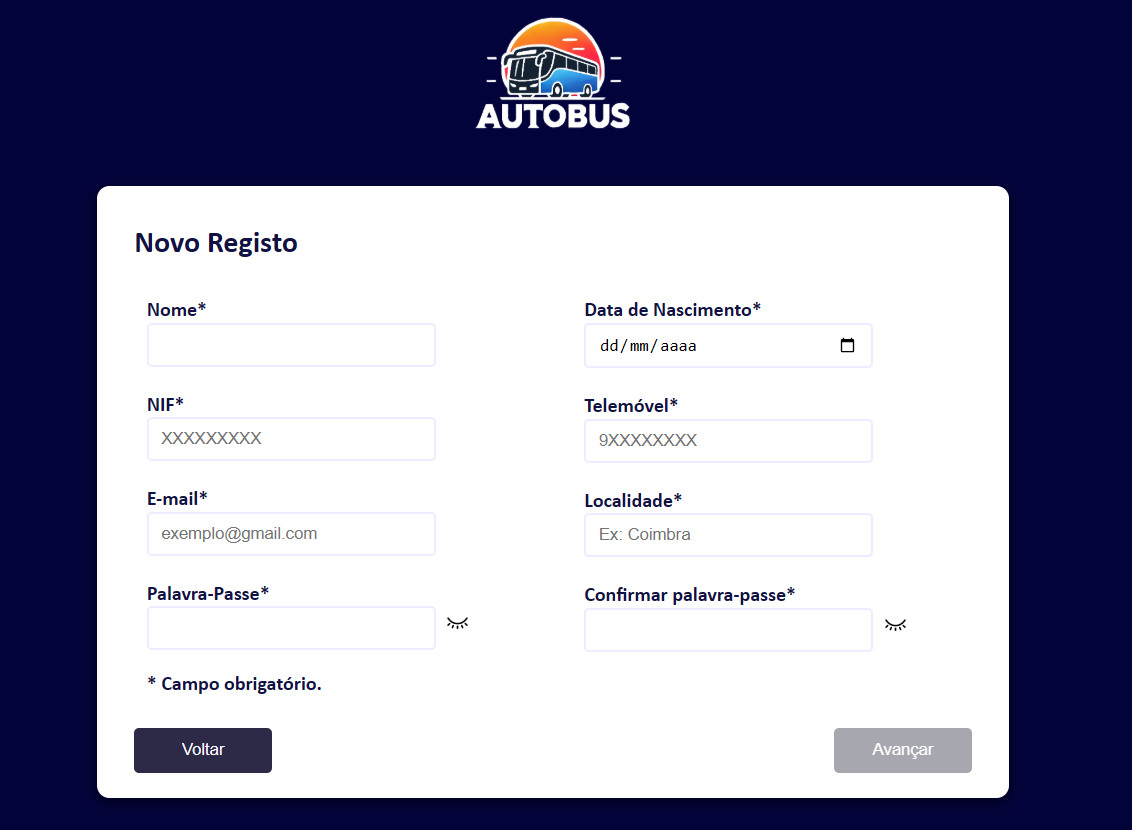


Figura - Página de criar conta.

Como havia antes, ainda há as restrições na password como em outros campos também, no entanto, são mostrados de diferentes formas.

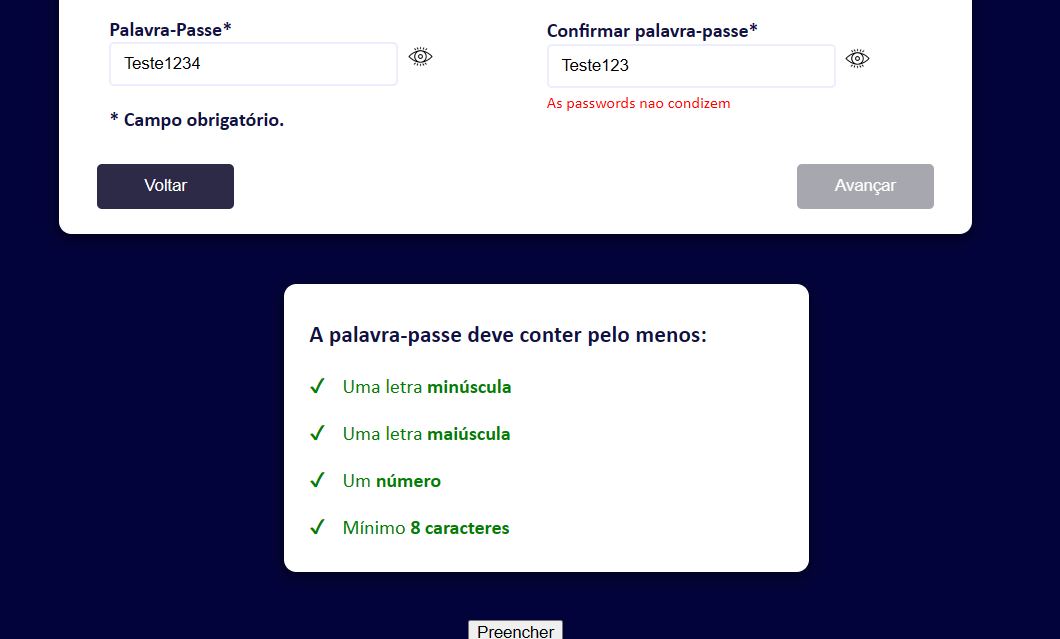


Figura - Restrições.

O botão de avançar só fica ativo quando todos os campos estiverem preenchidos, as palavras-passe estiverem corretas e não haver campos que já existem na base de dados, como o NIF ou o email.

O botão preencher foi criado temporariamente (apenas para testes) para criar um utilizador com dados aleatórios, só para todos os campos ficarem preenchidos e eu conseguir testar os próximos passos após preencher o formulário.

Ao clicar no botão “avançar”, presente no formulário o utilizador é redirecionado à um modal de foto de perfil, caso queira colocar alguma em sua conta.

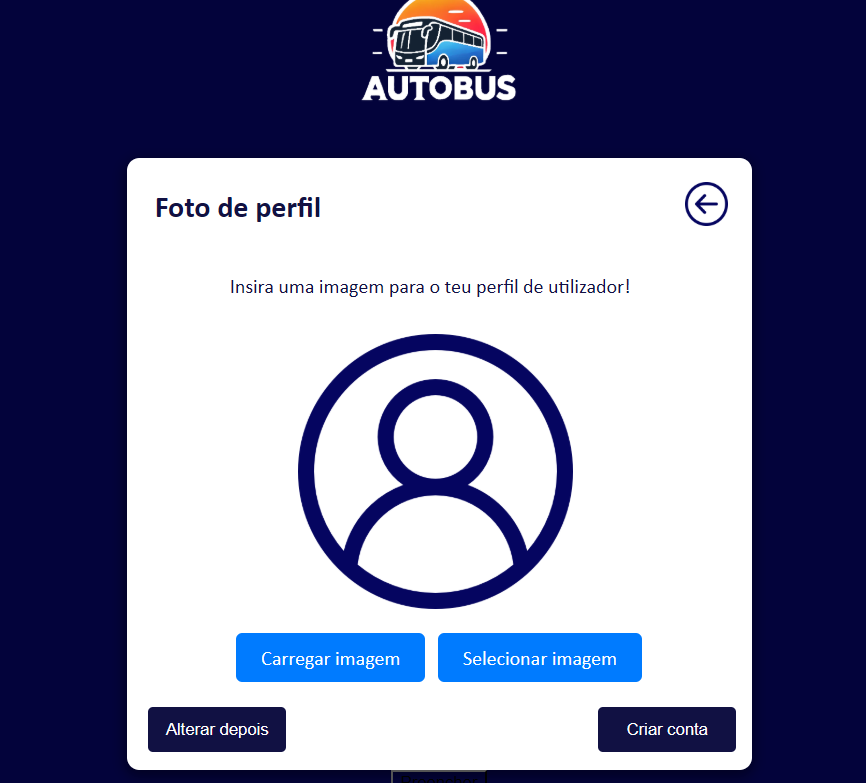


Figura - Modal foto de perfil.

Eu também criei um modal de imagens predefinidas do site para serem selecionadas.

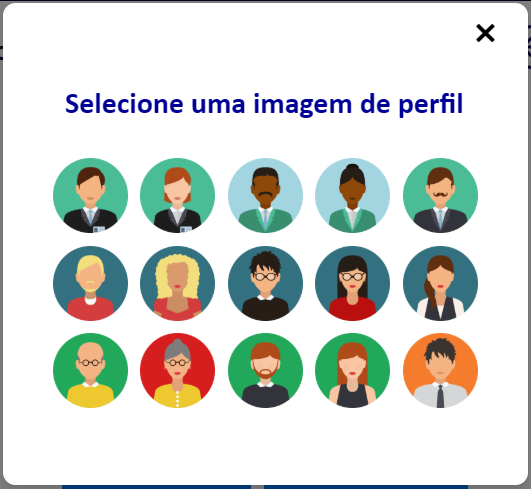


Figura - Seleção de imagens predefinidas.

## Implementação

A fase de implementação incluiu os seguintes passos:

1. **Criação da Base de Dados:**
   * **Utilização do PHPMyAdmin para a criação da base de dados;**
   * Utilização do MySQL Workbench para criar o desenho da base de dados;
   * Criação de tabelas e definição de chaves primárias e estrangeiras para garantir a integridade referencial.
2. **Desenvolvimento da Aplicação:**
   * Construção de formulários em C# para permitir o login de administradores e gestão de utilizadores e autocarros.
   * Criação de uma interface intuitiva para que os administradores possam executar as ações mencionadas.
3. **Desenvolvimento do Site:**
   * Uso de HTML, CSS, JS e PHP para criar um ambiente interativo para os clientes.

Integração de mapas interativos e visualização de rotas, funções para gerir compras e para gerir a própria conta. A nova versão do site foi desenvolvida com:

* **Angular** para o frontend (interface do utilizador),
* **Node.js** para o backend (servidor e lógica de negócio),
* **APIs REST** para comunicação entre o frontend e a base de dados,
* **MySQL** como sistema de gestão da base de dados (mantido da versão anterior).

A utilização de HTML e CSS se mantém, exceto o PHP e o JavaScript.

* + Esta alteração foi feita com o objetivo de tornar o projeto mais modular, mais organizado e flexível, garantindo um desempenho mais eficiente e uma melhor experiência de utilização.

1. **Testes:**
   * Cada funcionalidade foi implementada e testada em separado antes de ser integrada ao sistema completo (nem tudo que foi mencionado ainda foi feito, pois o projeto ainda está em progresso).

# Testes e instalação

**Testes**  
Os testes desempenham um papel fundamental para garantir o correto funcionamento do projeto. Por isso, foram abordados:

1. **Testes Unitários**
   * Cada módulo foi testado isoladamente para verificar a sua funcionalidade, como o sistema de login, gestão de utilizadores e gestão de autocarros.
2. **Testes de Integração**
   * Após a validação dos módulos individuais, realizou-se a integração entre a aplicação e o site, garantindo que a base de dados foi corretamente sincronizada entre ambos.
3. **Teste Funcionais**
   * O sistema foi avaliado em cenários reais simulados, como a criação de utilizadores, compra de bilhetes e consulta de rotas.
4. **Testes de Interface**
   * Testou-se a experiência do utilizador para assegurar que tanto o site quanto a aplicação oferecem uma navegação intuitiva e sem falhas.

Embora nenhum dos dois (aplicação e site) estejam completos, estas fases de testes têm de ser feitas e vão sendo feitas ao longo do progresso do projeto.

**Instalação**

O processo de instalação do projeto foi definido de forma a ser eficiente e seguro, utilizando as seguintes ferramentas:

* XAMPP;
* Visual Studio;
* Visual Studio Code;
* PHP MyAdmin;
* MySQL Workbench.

# Objetivos do projeto

O objetivo do projeto é mostrar todos os conhecimentos e competências que aprendi durante estes três anos que foi adquirido nas diversas disciplinas e conseguir implementá-las neste projeto. Pretende-se mostrar um site e uma aplicação que gere utilizadores e autocarros de forma fácil e acessível.

# Interesse e aplicabilidade do projeto

O desenvolvimento deste projeto irá me proporcionar o desenvolvimento e aplicação de conhecimentos que obtive durante estes três anos nas várias disciplinas, nomeadamente, Programação e Sistemas de Informação e Redes de Comunicação.

Este projeto não será usado em nenhuma empresa e nem terá qualquer tipo de fins lucrativos, pois foi feito por uma ideia pessoal e não para algo de fora.

# Disciplinas envolvidas

* Programação e Sistemas de Informação;
* Redes de Comunicação;

Embora estas sejam as disciplinas envolvidas, nem todas são “utilizadas” no projeto. Redes de Comunicação faz parte do site (onde aprendi HTML, PHP, CSS e JS) e Programação e Sistemas de Informação faz parte da aplicação, onde aprendi C# e muitas de suas utilidades.

# Saberes e competências profissionais incorporados no projeto

Durante o desenvolvimento deste projeto, foram aplicados diversos saberes e competências adquiridos ao longo do curso. As linguagens e tecnologias utilizadas foram:

* **C#** para o desenvolvimento da aplicação de gestão, utilizada pelos administradores;
* **SQL** para a criação e gestão da base de dados em MySQL;
* **HTML, CSS, JavaScript e PHP**, na versão inicial do site;
* **Angular** e **Node.js**, utilizados na versão final do site — tecnologias que aprendi de forma autónoma, por iniciativa própria, com o objetivo de melhorar a estrutura e modernizar o projeto;
* **JSON e APIs REST**, para a comunicação entre o frontend (Angular) e o backend (Node.js);
* Ferramentas como **Visual Studio**, **Visual Studio Code**, **XAMPP**, **PHPMyAdmin** e **MySQL Workbench**.

A utilização destas tecnologias permitiu aplicar competências como o planeamento e implementação de bases de dados relacionais, o desenvolvimento de interfaces de utilizador, a programação orientada a objetos, a criação de APIs e a utilização de *frameworks* modernas no desenvolvimento web.

# Fases temporais de implementação do projeto

O projeto terá as seguintes fases:

* Planeamento do projeto - até dia 15 de Novembro;
* Desenvolvimento da Base de Dados – até dia 25 de Novembro;
* Desenvolvimento da Aplicação – até dia 15 de Janeiro;
* Desenvolvimento site – até dia 30 de Janeiro;
* Fase de testes – até dia 25 de Fevereiro;

# Horas previstas na implementação do projeto

De acordo com todas as horas previstas para a implementação do projeto são aproximadamente um total de 400 a 450 horas. No entanto, por ser uma estimativa, não signifique que eu ocupe todas as horas ou precise de mais. E como prova da estimativa, as diferentes fases temporais previstas na implementação do projeto estão divididas pelas seguintes horas:

* Desenvolvimento da Base de Dados (72 horas);
* Desenvolvimento da Aplicação (100 horas);
* Desenvolvimento do Site (200 horas);
* Testes (40 horas).

# Recursos Humanos envolvidos

Os recursos humanos envolvidos para a realização deste projeto são:

* A desenvolvedora do projeto (eu);
* O diretor de curso José Carlos Martins;
* O professor de Programação e Sistemas de Informação Pedro Costa;
* A professora de Redes e Comunicação Susana Veiga;
* A diretora de turma Cristina Pires;
* O professor de Sistemas Operativos Jorge Sousa;
* Colegas de turma e amigos de fora.

# Recursos/Materiais envolvidos

Para estes protejo são precisos alguns materiais para que sejam implementadas toda a informação dita acima dentro das datas de requerimento. Os materiais são:

* Um computador com Windows 11 (no meu caso, eu utilizo o meu portátil HP Victus com o sistema operativo Windows 11);
* Tablet gráfico de desenho (HUION) com aplicação Clip Studio e Clip Studio Paint;
* Rato (neste caso usei o que eu já tinha, Logitech G502 HERO);
* Visual Studio 2022 Community;
* Visual Studio Code;
* XAMPP;
* MySQL Workbench;
* Microsof Office 365;
* Notepad++;
* Photopea;
* Lucidchart.;
* GitHub.

# Parcerias com empresas/ instituições

Para este projeto, não tive nenhuma parceria com empresa ou instituições, porque a ideia da aplicação e do site de gestão de autocarros foi reutilizada de um módulo de programação (especificamente o módulo 15), onde eu fiz um trabalho final com uma aplicação, também acompanhado de um relatório e uma apresentação. Portanto, por ter sido reutilizado, eu decidi aprimorar o que já tinha feito, tendo uma base em mente e fazer daquilo algo melhor.

# Reflexão sobre a exequibilidade do projeto

Este projeto é prático, com aplicação útil na vida real, embora não se torne algo grande e que terá utilização no futuro. Durante estes anos eu adquiri todo o conhecimento que será necessário para o desenvolvimento deste projeto e mais alguns de fora. No entanto, será necessário gerir bem o tempo para garantir a conclusão deste sistema funcional.

# Reflexão final sobre o trabalho a desenvolver

Este projeto vai pôr à prova todo o meu conhecimento em programação e, não só, como também o meu crescimento pessoal. Além de proporcionar prática e lógica em programação, também ajudou a reforçar as minhas habilidades de organização, autonomia e comunicação. O projeto permitiu-me apresentar um produto final com utilidade prática, incentivando a superação de desafios e dificuldades.

Durante o desenvolvimento, tomei a iniciativa de aprender tecnologias novas — como Angular e Node.js — que não foram lecionadas nas aulas, mas que considerei importantes para melhorar a qualidade e a estrutura do meu site. Esta aprendizagem autónoma representou um desafio adicional, mas também um grande contributo para o enriquecimento pessoal e técnico do projeto.

# Bibliografia

* Guião de Projeto de P.A.P (ficheiro em .pdf, disponibilizado aos alunos pela plataforma Moodle);
* Site de informação: <https://www.w3schools.com/> : [Refsnes Data, 1999];
* Loja online de viagens: <https://www.flixbus.pt/percursos-autocarros>.

# Anexos

**Anexo 1**

* Victus HP Laptop
  + Processador – Intel Core i5-11400H (2.70GHz);
  + RAM - 8,00 GB.