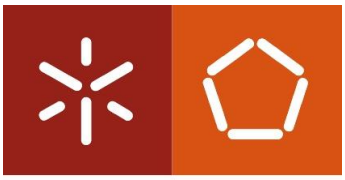

Universidade do Minho

Departamento de Informática

Laboratórios de Engenharia Informática
Projeto 93 - Carage

João Teixeira (A85504)
José Filipe Ferreira (A83683)
Miguel Solino (A86435)

22 de junho de 2021

Resumo

Ao longo deste relatório iremos descrever e documentar o processo de criação de uma plataforma digital Web e Mobile em modo PWA com a capacidade para gestão, compra e venda de automóveis com um componente de *machine learning* para calcular o valor do automóvel no momento da venda.

Conteúdo

1 Introdução **2**

2 Estudo de Mercado **3**

3 Backend **4**
3.1 Tecnologias Utilizadas 4
3.2 Base de dados 4
3.2.1 Conexão da Base de Dados 5
3.2.2 Users 5
3.2.3 Models 5
3.2.4 Cars 5
3.2.5 Maintenances 5
3.2.6 Ads 5
3.3 Autenticação do Utilizador 5
3.4 Comunicação com o Backend 5

4 Previsão de Preços **6**
4.1 Obtenção de Dados 6
4.2 Análise dos Dados 6
4.3 Preparação dos Dados 9
4.4 Aplicação dos Modelos e Resultados 10

5 Frontend **12**
5.1 Página Inicial 12
5.2 Dashboard 14
5.3 Carros 15
5.4 Carro 16
5.5 Mercado 17
5.6 Anúncio 18

6 Conclusão e Trabalho Futuro **19**

Capítulo 1

Introdução

O mercado automóvel e uma área em crescimento constante: Existem mais de 290 milhões de carros na União Europeia e no ano de 2019 foram vendidos mais de 15 milhões de automóveis novos. No entanto, gerir uma frota automóvel revela-se extremamente difícil. Organizar todas as despesas de cada carro e interpretar essa informação de forma útil requer o uso de ferramentas não intuitivas que tradicionalmente estão renegadas ao reino da contabilidade de uma empresa. Por isso, tanto uma pessoa que tenha mais do que um carro como uma pequena empresa tem imensas dificuldades em gerir os custos da sua frota automóvel.

Com a idade media dos automóveis a aumentar todos os anos, o mercado de carros usados encontra-se definitivamente em florescimento. Infelizmente, quando finalmente chega a altura de vender os carros para adquirir novas viaturas, estimar o preço de cada uma requer um conhecimento técnico e uma investigação cuidada por parte do proprietário que a maioria simplesmente não tem a disponibilidade para fazer.

A nossa *web app* propõe-se a resolver estes dois problemas numa plataforma só que junta a compra, venda e gestão de automóveis. Um utilizador pode adicionar os seus automóveis, associar de forma intuitiva despesas correntes e receber estatísticas sobre os custos totais de forma sucinta e clara. Quando o utilizador finalmente quiser vender um dos seus carros, um algoritmo de *machine learning* estima o valor de mercado para que seja possível vender o carro na plataforma de venda integrada ao melhor preço possível, mas mesmo assim competitivo com o mercado actual. Este mesmo algoritmo e usado para auxiliar os compradores a saberem se o preço do carro em que estão interessados em comprar esta dentro da media do mercado.

Ao longo deste relatório iremos descrever as tecnologias utilizadas e a estrutura deste aplicação revolucionaria colocando um foco no desenvolvimento do *machine learning* para previsão de preços.

Capítulo 2

Estudo de Mercado

Fazendo uma pesquisa de eventuais concorrentes descobrimos que existem algumas plataformas de vendas de carros usados no mercado. Exemplos destas São o StandVirtual, o auto.sapo.pt e o Olx. Porem, existem algumas falhas em todas estas plataformas. O Olx e uma plataforma generalista de venda de qualquer produto e, por isso, não inspira a confiança necessária para um comprador em busca de um carro usado. O auto.sapo.pt não apresenta qualquer tipo de previsão de preço tanto para o vendedor como para o comprador. O StandVirtual e a plataforma que mais se aproxima do nosso componente de venda de automóveis pois apresenta uma indicação a possíveis compradores se o preço do carro esta acima da media. No entanto, a data deste relatório, não apresenta qualquer tipo de sugestão de preço aquando da venda do carro.

No departamento de gestão de uma frota automóvel: as poucas aplicacoes moveis existentes tem interfaces gráficas desactualizadas e não existem *web apps* no mercado. Desta forma não existe nenhuma solução para gestão de automóveis que seja transversal a todos os dispositivos.

Por outro lado, não existe qualquer plataforma digital no mercado que funda a vertente de gestão e a vertente de compra e venda, fazendo com que este projeto apresente uma embalagem única de funcionalidades.

Capítulo 3

Backend

3.1 Tecnologias Utilizadas

O backend foi escrito na linguagem de programação Rust, escolhida pela sua capacidade de garantir correção do código aquando da compilação, sendo menos propicio a existência de bugs, aliada à boa documentação e existência de frameworks úteis ao projeto.

Para o motor da base de dados, escolhemos uma base de dados relacional *postgres*, visto que é o mais popular motor de base de dados utilizado atualmente.

3.2 Base de dados

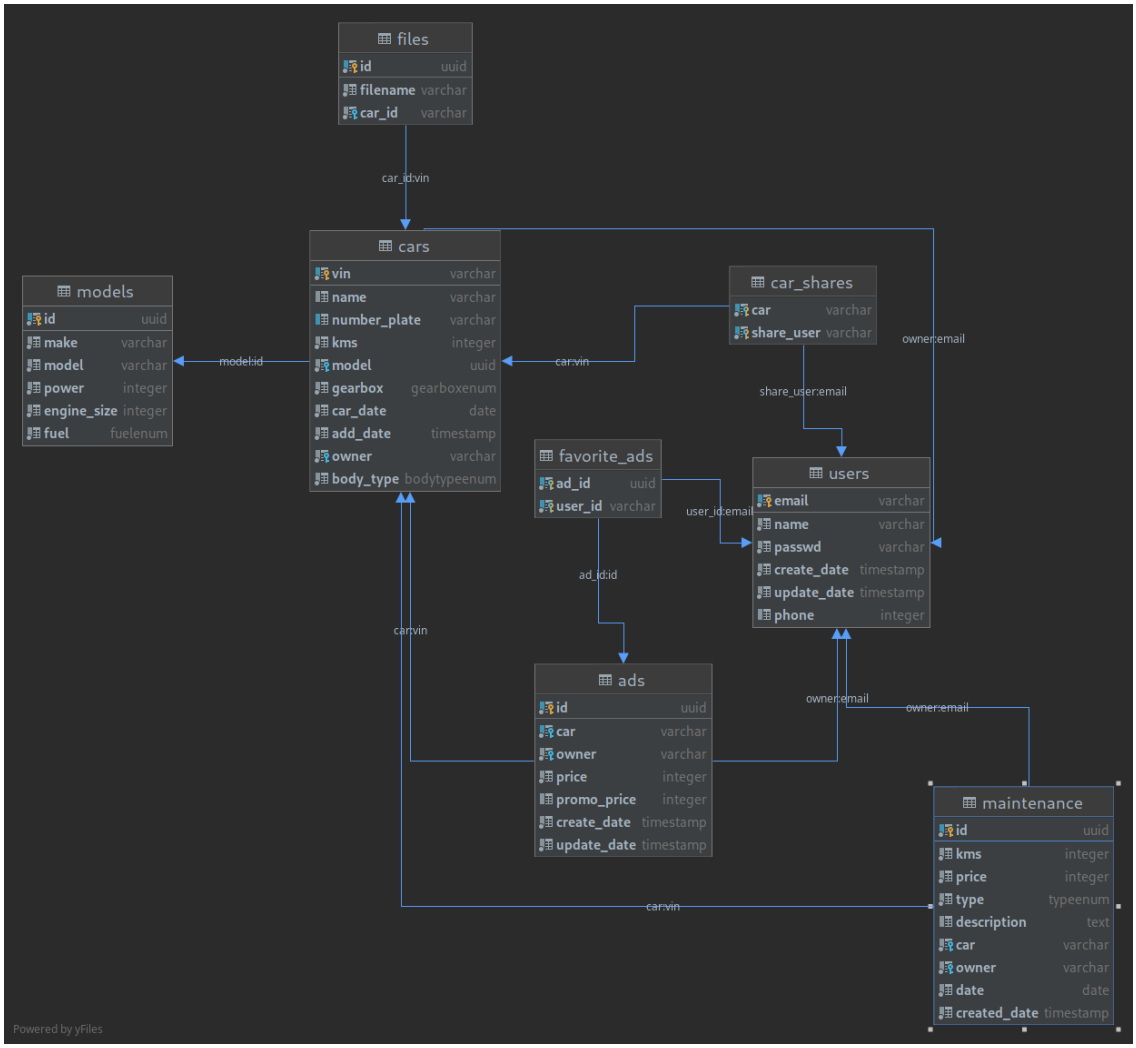

Figura 3.1: Modelo da base de dados

3.2.1 Conexão da Base de Dados

A conexão do backend à base de dados, é efetuada através da framework de *ORM (Object-Relational Mapping)* diesel, disponivel para a linguagem de programação Rust. Esta facilita a manipulação de dados na base de dados, pois apresenta uma abstração das queries SQL para código.

3.2.2 Users

Esta tabela contem a informação de todos os utilizadores registados no sistema. É aqui que são armazenadas as informações de contacto, bem como as suas credenciais de acesso.

3.2.3 Models

Esta tabela contem a informação dos modelos de veículos que podem inseridos no sistema. Visto que é uma informação que iria ser comum a muitos veículos, foi abstraída para esta tabela. É previamente populada, com base na informação presente no dataset recolhido para a previsão de preços, neste momento com mais de 4000 modelos distintos, para proporcionar uma melhor e mais facilitada experiência de inserção de carros por parte do utilizador final.

3.2.4 Cars

Esta tabela contem a informação especifica dos veículos do utilizador final, que consiste na quilometragem, tipo de caixa de velocidade, data da primeira matricula, matricula, tipo de automóvel, se é um carro, carrinha ou SUV, por exemplo.

3.2.5 Maintenances

Tabela que guarda a informação sobre despesas e manutenções de cada carro, nomeadamente, a quilometragem na qual foi efetuada, data e tipo de despesa, identificando se foi uma manutenção regular, de prevenção, mudança de pneus, combustível ou avaria.

3.2.6 Ads

Tabela que guarda a informação de um anuncio de um carro, contendo o identificador do veiculo ao qual se refere, ao utilizador que criou o anuncio, bem como o seu preço e data de inserção no sistema.

3.3 Autenticação do Utilizador

Por forma a simplificar a gestão do utilizador que faz os pedidos, implementamos um sistema de tokens, assente no standard *JSON Web Token* (*JWT*), que contém informação sobre o utilizador e uma assinatura digital que marca a autenticidade do mesmo, não dando assim acessos indevidos às páginas privadas do utilizador. Este token é gerado aquando da validação das credenciais no login, e deve estar presente no header do pedido *HTTP* para valiadção.

3.4 Comunicação com o Backend

Para a comunicação do exterior com o backend da aplicação, este expõe uma *REST API*, que apresenta endpoints para a manipulação dos dados guardados no sistema por parte do utilizador, de forma a manter a segurança e integridade da informação existente. Esta API funciona assente na framework rocket.

Capítulo 4

Previsão de Preços

Ao longo deste capitulo iremos descrever como desenvolvemos um algoritmo de *machine learning* com a capacidade de ter em conta o maior numero de factores possíveis para calcular o valor de mercado de um carro usado.

4.1 Obtenção de Dados

Para treinar este algoritmo precisamos de obter um *dataset* considerável de carros usados. Para tal usamos a maior plataforma de venda de carros usados em Portugal: o Standvirtual.

Para a obtenção dos dados decidimos utilizar uma linguagem de scripting, neste caso Javascript, de forma a que este processo fosse direto e pouco demorado. Inicialmente foi necessária uma análise de como o StandVirtual esta organizado relativamente a componentes HTML. Munidos desta informação e fazendo uso da biblioteca *Cheerio* de Javascript conseguimos manusear o HTML completo de forma a retirar apenas as informações relevantes sobre os dados dos carros. Este script vai página a página, obtém a lista de carros nessa página e extrai, carro a carro, as informações respectivas. Devido ao rate limiting incutido pela api do standvirtual apenas conseguimos obter dois carros por segundo levando a que o processo todo demore aproximadamente oito horas.

4.2 Análise dos Dados

Começando por uma análise geral dos dados, as informações que extraímos de cada carro foram:

- **brand** - Marca;
- **model** - Modelo;
- **version** - Sub-modelo;
- **fuel** - Tipo de combustível;
- **month** - Mês de produção;
- **year** - Ano de produção;
- **km** - Quilómetros totais percorridos;
- **displacement** - Cilindrada do motor;
- **power** - Potência em cavalos;
- **gearbox** - Tipo de caixa;
- **doors** - Número de portas;
- **seats** - Número de lugares;
- **origin** - Nacional ou Importado;
- **price** - Preço de venda;

Como resultado final obtivemos 46713 carros com um total de 14 colunas e tamanho de ficheiro 5 MiB, existindo alguns valores em falta.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46713 entries, 0 to 46712
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   brand         46712 non-null  object
 1   model         46712 non-null  object
 2   version       42170 non-null  object
 3   fuel          46712 non-null  object
 4   month         46712 non-null  object
 5   year          46712 non-null  float64
 6   km            46712 non-null  float64
 7   displacement  45945 non-null  float64
 8   power         46712 non-null  float64
 9   gearbox       46495 non-null  object
 10  doors         46279 non-null  float64
 11  seats         38798 non-null  float64
 12  origin        28086 non-null  object
 13  price         46712 non-null  float64
dtypes: float64(7), object(7)
memory usage: 5.0+ MB
```

Figura 4.1: Informações gerais do dataset obtido

Com o objetivo de ter os melhores resultados possíveis, passamos também por um processo de análise mais profundo. Ou seja, avaliamos quais São os factores que mais afectam o preço com base nas correlacoes representadas através de heatmaps.

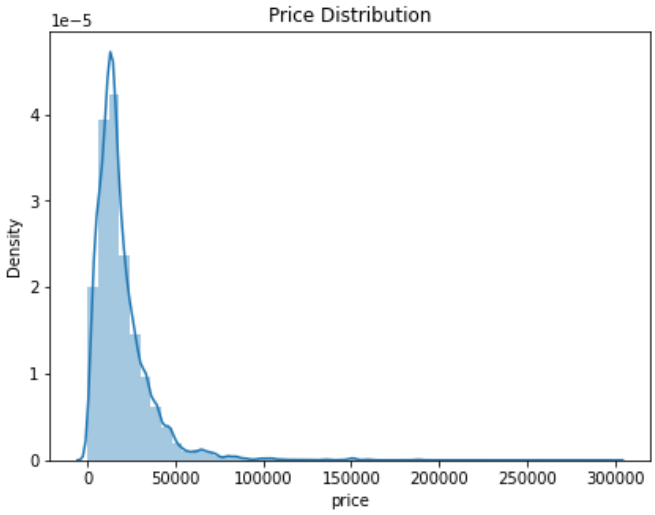

Figura 4.2: Densidade dos preços

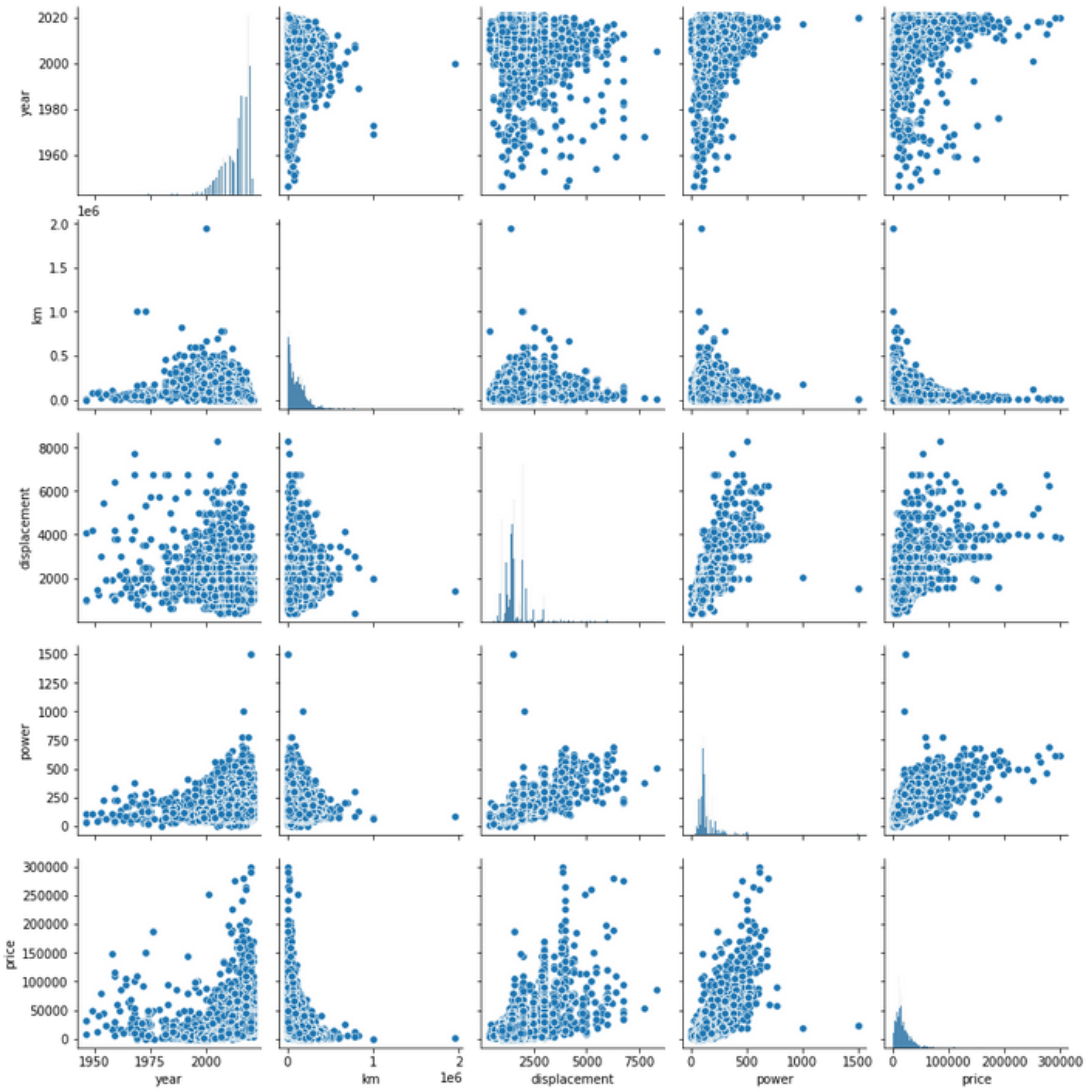

Figura 4.3: Correlação dos dados com mais impacto

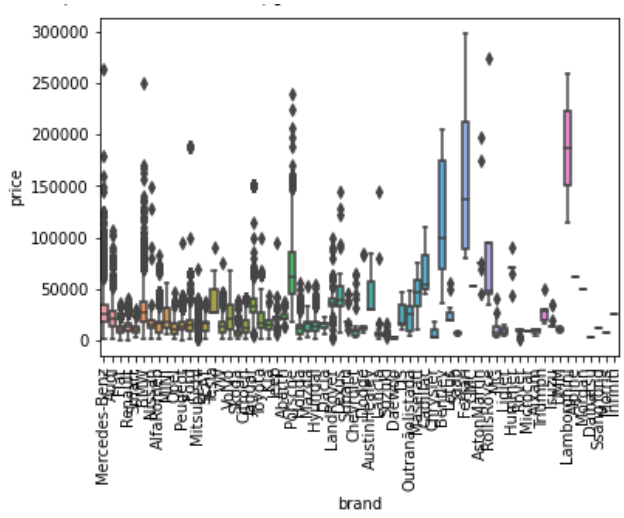

Figura 4.4: Distribuição dos preços pelas marcas

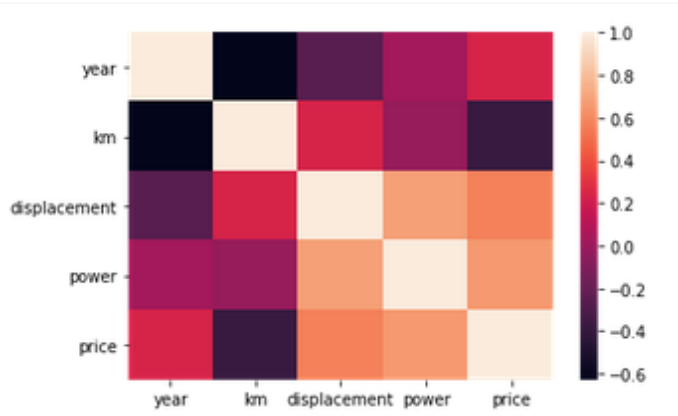

Figura 4.5: Correlação em Heatmap dos dados com mais impacto

Visualizando estes dois últimos gráficos chegamos à conclusão que os atributos com maior relevância para o nosso modelo são: brand, fuel, year, km, displacement, power e gearbox. Infelizmente, nem todos os gráficos analisados por nós são legíveis e por isso não achamos relevante colocar. Isto deve-se a alguns valores em falta e por maior parte dos campos serem nominais com valores únicos quase na sua totalidade.

4.3 Preparação dos Dados

Com uma análise feita e sabendo quais os dados que são obrigatórios usarmos, passamos à preparação dos mesmos para mais à frente aplicarmos o modelo. Como referido anteriormente, para uma melhora dos resultados foram separadas as colunas com maior impacto no preço, resultando no seguinte processo aplicado nos dados:

- **Remoção de atributos** - Remoção das colunas dos dados com menos impacto no preço sobrando as referidas na secção anterior;

- **Conversão dos dados** - Tendo em conta que vários dos dados são nominais e estes algoritmos não suportam valores não-numéricos, foi feita uma conversão dos dados originando colunas do tipo "boolean";
- **Remoção de valores em falta** - Como se trata de previsão de preços e o nosso objetivo é obter os melhores resultados possíveis, não podemos inventar valores que alterem a exactidão das previsões. Assim sendo removemos todas as linhas que continham valores em falta.

Abaixo estão presentes pedaços de código executados no Google Colab referentes a cada uma das etapas do processo.

```
df.drop(['model'],axis=1,inplace=True)
df.drop(['version'],axis=1,inplace=True)
df.drop(['seats'],axis=1,inplace=True)
df.drop(['doors'],axis=1,inplace=True)
df.drop(['month'],axis=1,inplace=True)
df.drop(['origin'],axis=1,inplace=True)
```

Figura 4.6: Remoção das colunas com pouca relevância

```
df_dummy = pd.get_dummies(car_test['fuel'])
car_test = pd.concat([car_test, df_dummy], axis = 1)
car_test.drop('fuel', axis = 1, inplace=True)

df_dummy = pd.get_dummies(car_test['gearbox'])
car_test = pd.concat([car_test, df_dummy], axis = 1)
car_test.drop('gearbox', axis = 1, inplace=True)

df_dummy = pd.get_dummies(car_test['brand'])
car_test = pd.concat([car_test, df_dummy], axis = 1)
car_test.drop('brand', axis = 1, inplace=True)
```

Figura 4.7: Conversão dos dados nominais para númericos

```
df_cars.dropna(axis=0,inplace=True)
```

Figura 4.8: Remoção das linhas com dados em falta

4.4 Aplicação dos Modelos e Resultados

Tendo em mãos os dados preparados, passamos à fase de treino e teste do dataset para cada algoritmo. Visto que tradicionalmente este tipo de situacoes São tratadas por algoritmos de regressão foi por ai que começamos a nossa pesquisa do melhor modelo para prever de preços dos automóveis.

Inicialmente experimentamos tres algoritmos distintos: Linear Regression, Lasso Regressione Ridge Regression. Curiosamente estes 3 apresentam resultados muito semelhantes e bastante abaixo das expectativas. Com uma pesquisa um pouco mais profunda descobrimos um outro tipo de modelo denominado de XGBoost (Regressor), que para surpresa nossa, foi o que nos ofereceu de longe os melhores resultados, sendo eles muito superiores aos três anteriores.

	Model	Training Score	Test Score	Mean Absolute Error	Mean Squared Error	Root Mean Squared Error
0	Linear Regression	0.733382	0.73197	4687.22	7.5414e+07	8684.12
1	Lasso Regression	0.732974	0.731542	4689.58	7.55344e+07	8691.05
2	Ridge Regression	0.732985	0.731169	4691.42	7.56393e+07	8697.08
3	XGBoost (Regressor)	0.928431	0.915887	2604.61	2.36663e+07	4864.81

Figura 4.9: Comparação entre pontuações e erros dos algoritmos usados

Tendo em conta estes resultados, decidimos utilizar como algoritmo final o XGBoost, obtendo previsão de preços bastante próximos dos reais.

	price	Predicted_Price
0	11950.0	10266.849609
1	11997.0	11772.080078
2	14249.0	13122.179688
3	5650.0	6689.709961
4	24890.0	23214.849609
5	19500.0	21675.599609

Figura 4.10: Cinco carros com o preço real e previsto resultante da aplicação do algoritmo

Capítulo 5

Frontend

Relativamente à componente do frontend, esta foi desenvolvida utilizando a framework Next.js e foram implementadas maior parte das funcionalidades propostas. É possível o utilizador criar uma conta na página inicial como também efetuar o login.

Na dashboard consegue-se visualizar um resumo geral de todos os dados da sua conta, como número de carros, anúncios, despesas, etc.

Depois na página de carros é possível criar um carro como aceder a um dos criados ou partilhados. Na própria página de um carro é possível adicionar manutenções, colocar à venda, partilhar o carro e pedir uma previsão ou uma estimativa de quanto é que o carro vale atualmente.

Na página de mercado é possível ver todos os anúncios existentes na plataforma e também a possibilidade de os filtrar por certos atributos.

Existe também a página de favoritos onde irão aparecer os anúncios favoritos e a página de perfil onde é possível editar o nome, password ou número de telemóvel.

É importante referir que a estimativa de preços é usada em duas páginas: a página do próprio carro como foi referido anteriormente e na página de um anúncio. Na página do carro esta previsão é dada num intervalo que o carro poderá valer. Já na página de anúncios a previsão de preços é utilizada de uma forma diferente, quando o carro está mais barato, mais caro ou dentro do preço previsto é indicado por baixo do preço dado pelo vendedor essa previsão.

Abaixo estão presentes maior parte das páginas existentes na nossa aplicação.

5.1 Página Inicial

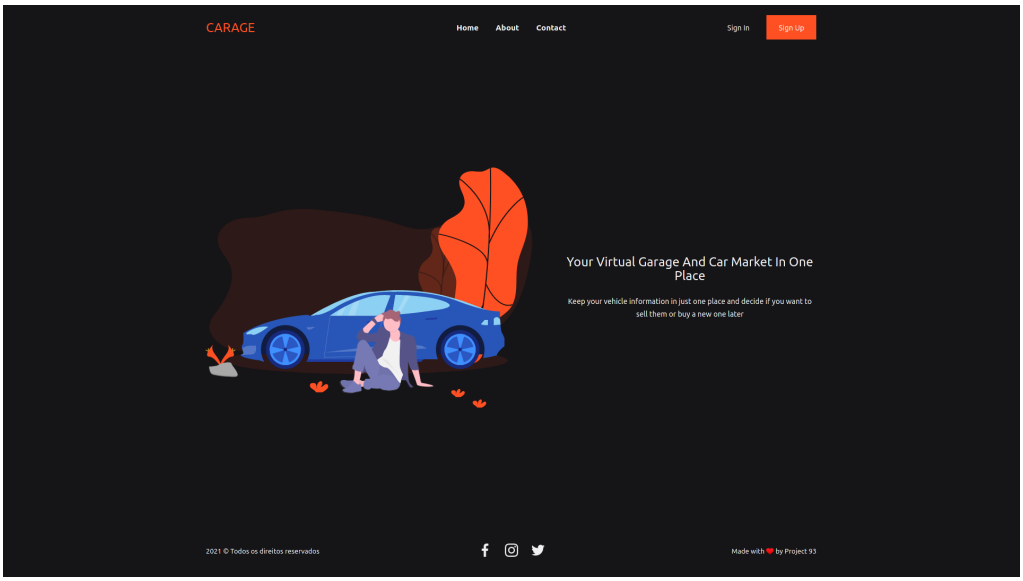

Figura 5.1: Página inicial desktop

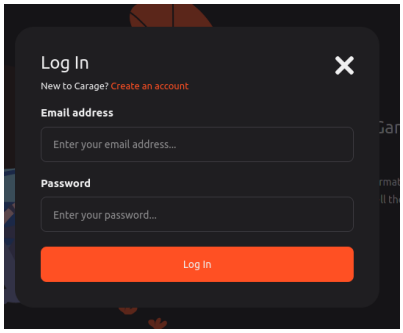

Figura 5.2: Form de login

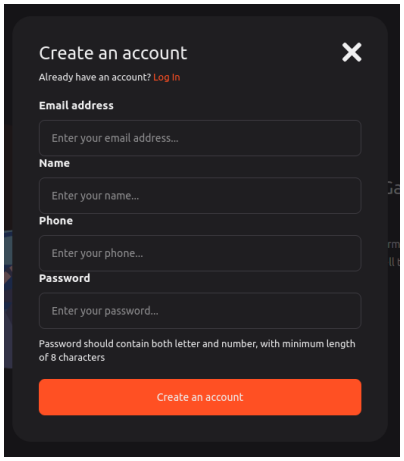

Figura 5.3: Form de registo

Figura 5.4: Página inicial mobile

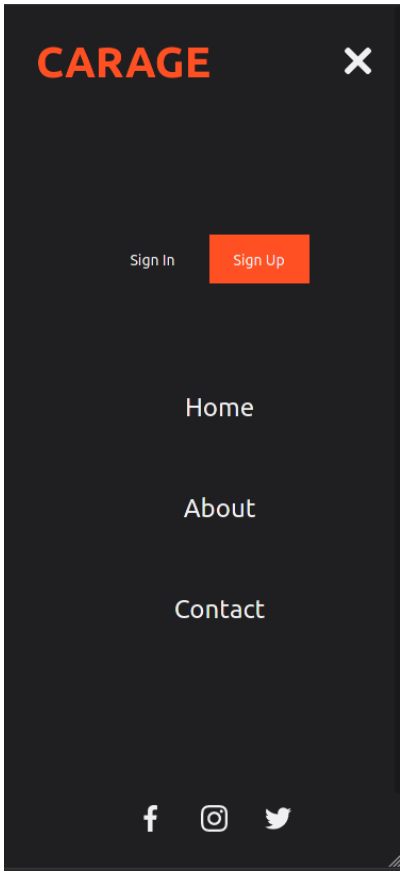

Figura 5.5: Menu da página inicial mobile

5.2 Dashboard

Figura 5.6: Dashboard Desktop

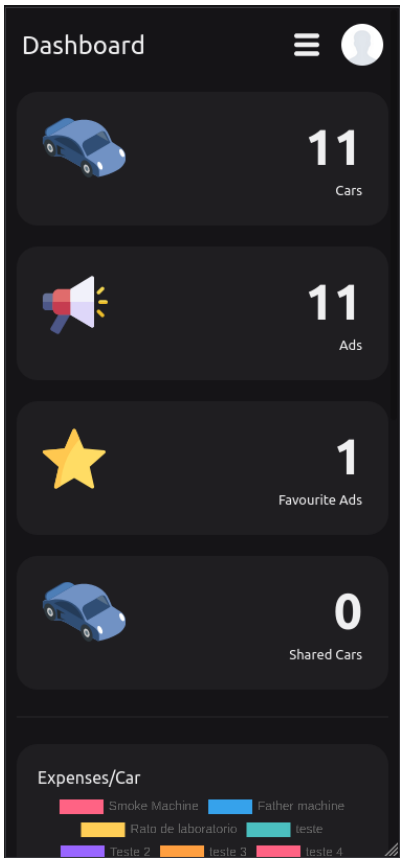

Figura 5.7: Dashboard Mobile

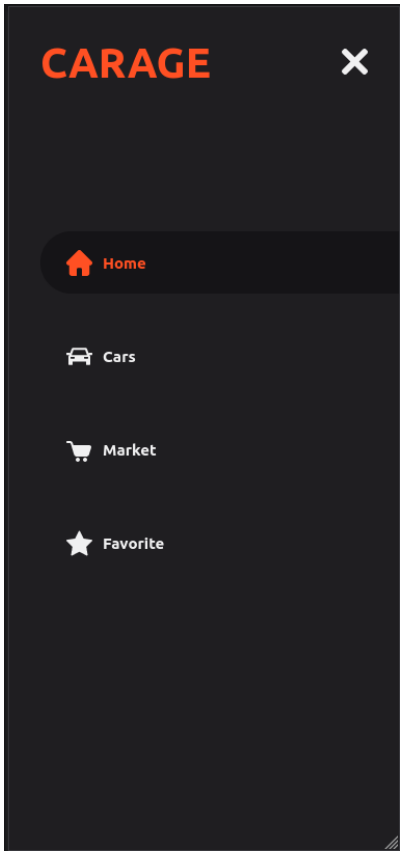

Figura 5.8: Navbar da Dashboard Mobile

5.3 Carros

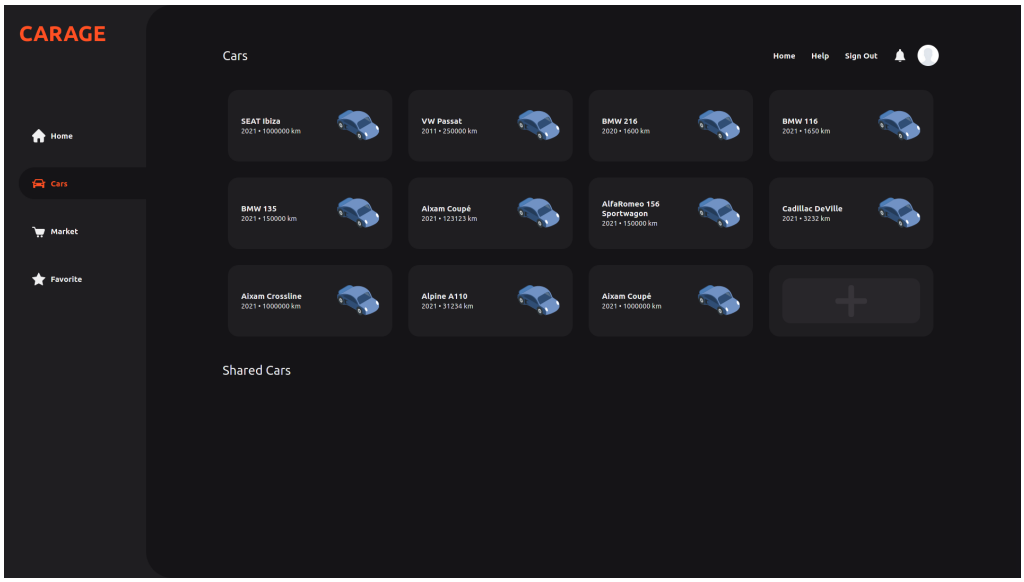

Figura 5.9: Página dos Carros Desktop

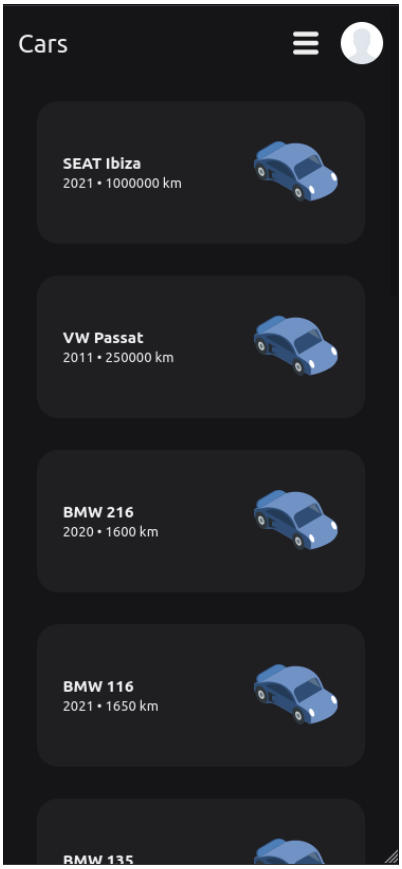

Figura 5.10: Página dos Carros Mobile

5.4 Carro

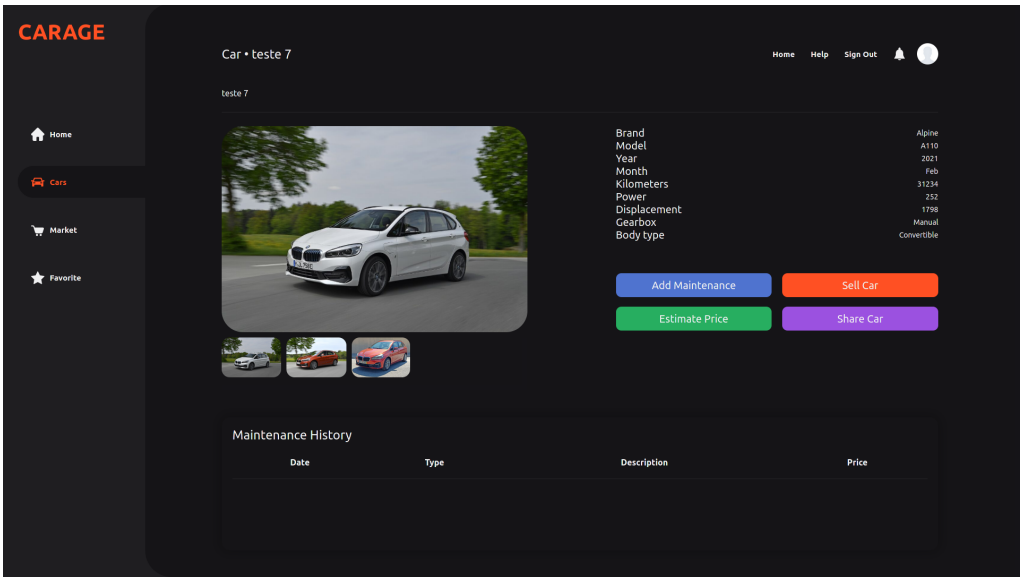

Figura 5.11: Página de um Carro Desktop

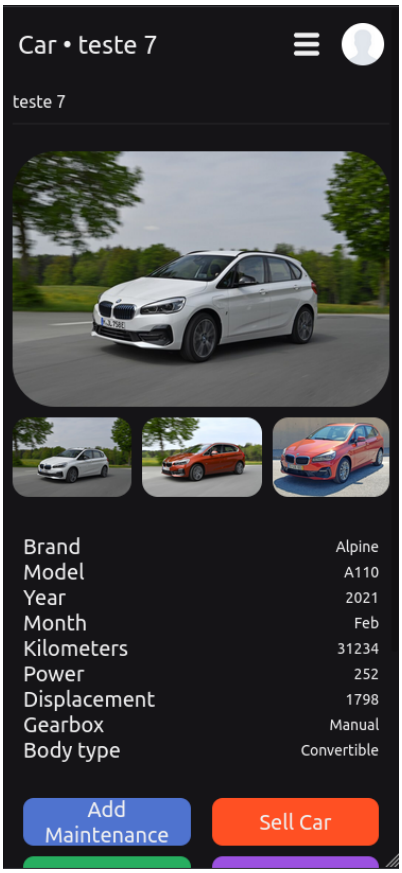

Figura 5.12: Página de um Carro Mobile

5.5 Mercado

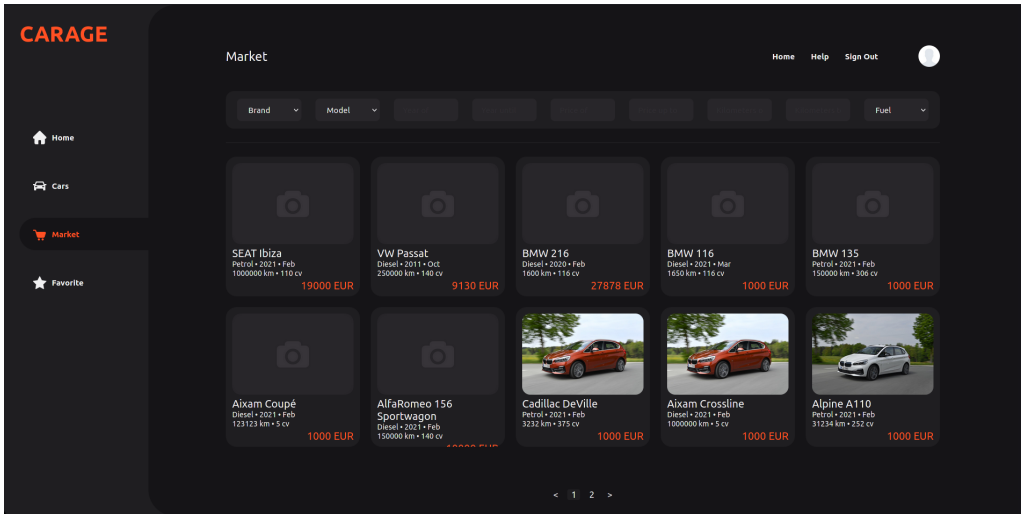

Figura 5.13: Página do Mercado Desktop

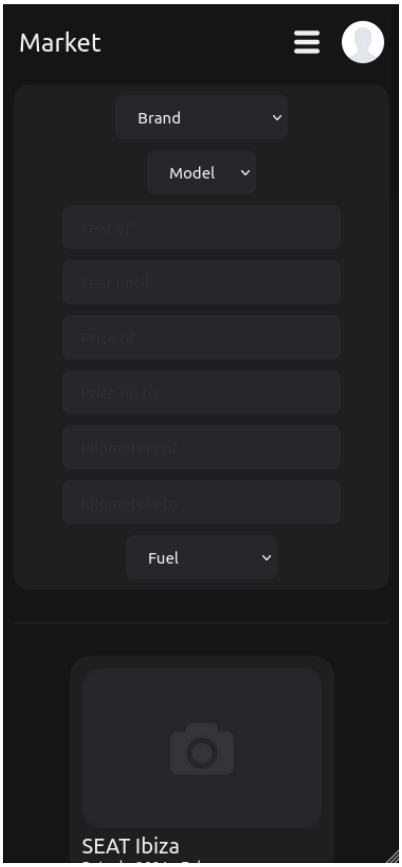

Figura 5.14: Página do Mercado Mobile

5.6 Anúncio

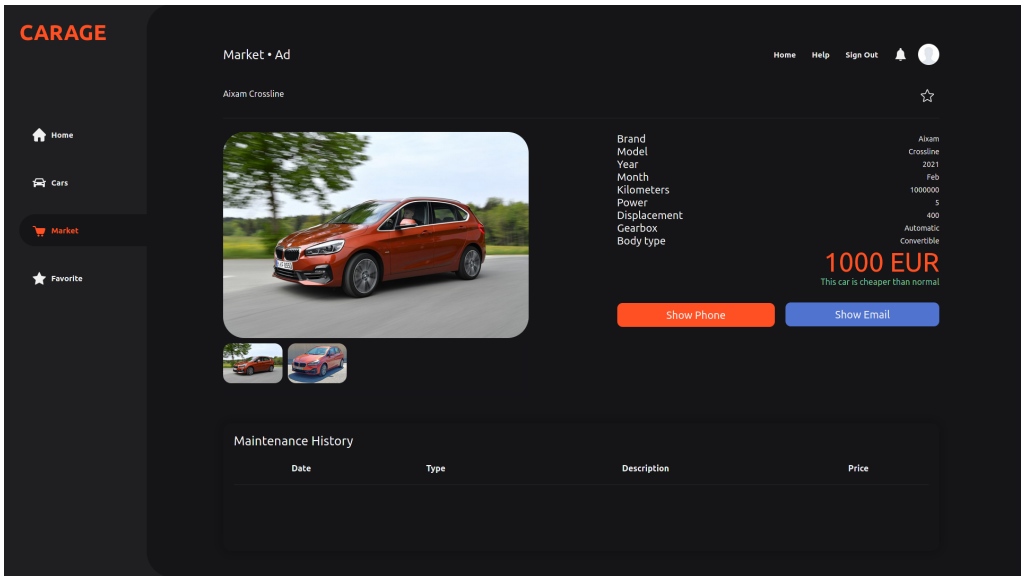

Figura 5.15: Página de um Anúncio Desktop

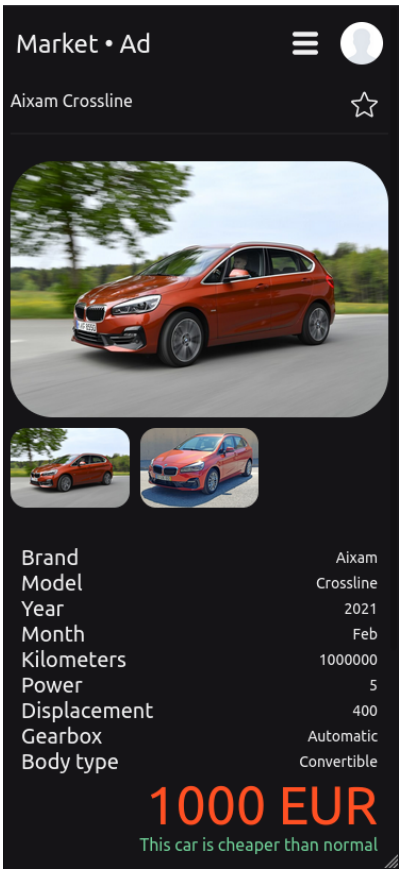

Figura 5.16: Página de um Anúncio Mobile

Capítulo 6

Conclusão e Trabalho Futuro

Conseguimos cumprir todos os requisitos a que nos propusemos inicialmente conseguindo realizar com sucesso uma plataforma de gestão compra e venda de automóveis intuitiva e simples de usar acessível a qualquer pessoa.

Como trabalho futuro gostaríamos de permitir a introdução de um carro apenas com base na sua matricula e associar fotos ou ficheiros de facturas aos consumos do carro. Para tal seria necessário expandir as funcionalidades do armazenamento de ficheiros criado de forma a ser mais genérico e flexível. Também gostaríamos de apresentar gráficos na pagina de cada carro semelhantes aos da dashboard. Se este projecto fosse de facto ser utilizado por um numero elevado de utilizadores também seria necessário fazer um deployment na cloud, criar uma página de contactos e ajuda funcional e ter um guia de utilizacao da plataforma quando se entra pela primeira vez.