

DEPARTAMENTO DE TECNOLOGIAS

Mestrado Em INFORMÁTICA

Integração de Sistemas | 2023-2024

Docente: João Ventura



ESTGD – Relatório de época de
frequência

Gestão de Frotas

Camoin Pamante 22906

Sónia Pimentel 24691

Sumário

Objetivo.....	3
Modelação de dados	3
Tabelas	3
Modelo E-R.....	4
Tipos de relações.....	4
Models	4
Administração backend	6
Aplicação Web.....	6
Frontend.....	6
Estrutura frota conjunto de ficheiros .py	7
Funcionalidades	8
Utilizadores	8
Barra de navegação – Navebar com utilizador Administrador.....	10
Página About ou Acerca de Nós	13
Barra de navegação – Navebar com utilizador Funcionário.....	13
Media	15
Segurança.....	15
O código de resposta de estado de sucesso HTTP 204	15
Status=status.HTTP 400 BAD REQUEST	15
get_object_or_404.....	16
Utilização do Try catch.....	17
Encriptação da password.....	17
Bootstrap	17
Django API RestFul - Criar API do projeto.....	17
Comandos	17
Estrutura	18
VIEWS.PY GET/POST para PEDIDO	18
Class Listar Veiculos – Função GET/POST.....	19
Class CRUD Veículos – DELETE/PUT.....	19
Class Lista utilizadores – GET/POST	20
Class CRUD Utilizadores – DELETE/PUT	20
Serializers.py	20
Interface API.....	21
Considerações Finais	23

Objetivo

Este relatório foi realizado no contexto da unidade curricular de Integração de Sistemas do curso de Mestrado de Informática, com este projecto pretende-se a implementação de uma aplicação que inclua um modelo de dados, uma aplicação web e uma API Restful. Utilização de Django e.djangorest-framework. O tema do projecto é uma Plataforma de e-commerce, com utilizadores, veiculos e pedidos.

Modelação de dados

Tabelas

Tabela Utilizador Abstrata

Utilizador	Tipo de dados
Nome	Class abstrata

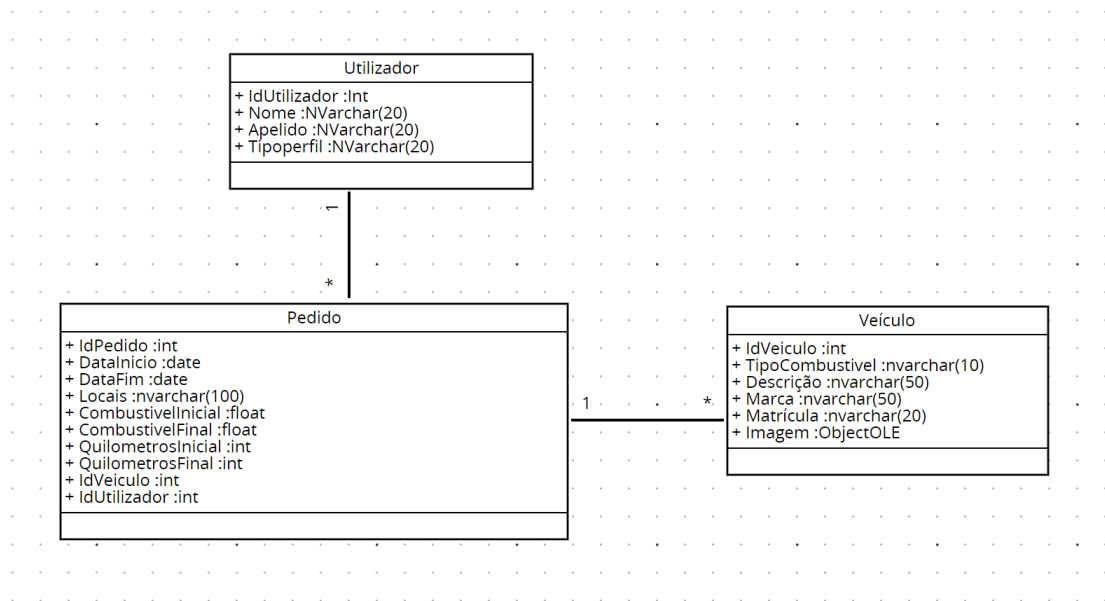
Tabela Pedido

IdPedido	Int
DataInicio	Date
DataFim	Date
Local	Char
Combustível_Inicial	Int
Combustível_Final	Int
Kilometro_Inicial	Int
Kilometro_Final	Int

Tabela Veículo

Veículo	Tipo de dados
Matricula	CharField
Marca	CharField
Tipo de Combustível	CharField
Descrição	TextField
Imagem	ImageField

Modelo E-R



Tipos de relações

Utilizador - Pedido

Relação de 1..* (1 utilizador faz muitos pedidos)

Pedido-Veículo

Relação de 1..* (1 pedido para muitos veículos)

Models

Ficheiro Models.py

```
class User(AbstractUser):
    def __str__(self):
        return f'{self.username}, {self.is_staff}'

class Veiculo(models.Model):
    matricula = models.CharField(max_length=50)
    marca = models.CharField(max_length=50)
    tipo_combustivel = models.CharField(max_length=50)
    description = models.TextField(max_length=100, blank=True,
    null=True, default='')
    image = models.ImageField(upload_to='uploads/veiculo')

class Pedido(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE,
    related_name='userpedidos')
    veiculo = models.ForeignKey(Veiculo, on_delete=models.CASCADE,
    related_name='veiculopedido')
```

```

data_inicio = models.DateField()
data_fim = models.DateField()
local = models.CharField(max_length=50)
combustivel_inicial = models.IntegerField()
combustivel_fim = models.IntegerField(default=0)
kilometro_inicial = models.IntegerField(default=0)
kilometro_final = models.IntegerField(default=0)

class Notification(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='usernotification')
    funcionario = models.CharField(max_length=20)
    mensagem = models.CharField(max_length=100)
    confirmacao = models.BooleanField(default=False)

```

Admin

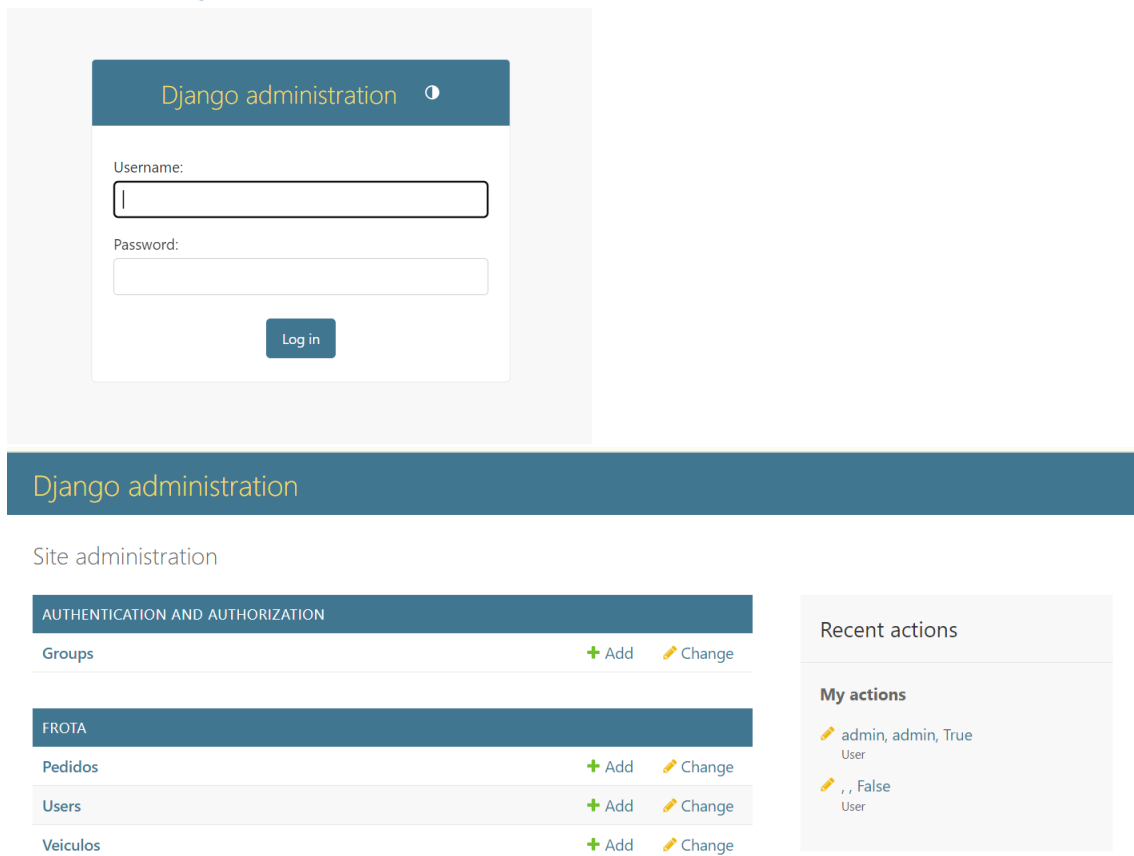
```

@admin.register(User)
class UserAdmin(admin.ModelAdmin):
    list_display = ('username', 'first_name', 'last_name', 'email',
'is_staff')

@admin.register(Pedido)
class ListAdmin(admin.ModelAdmin):
    list_display = ('user', 'data_inicio', 'data_fim', 'local',
'kilometro_inicial', 'kilometro_final',
'combustivel_inicial', 'combustivel_fim' )
@admin.register(Veiculo)
class ListAdmin(admin.ModelAdmin):
    list_display = ('matricula', 'marca', 'tipo_combustivel',
'description', 'image')

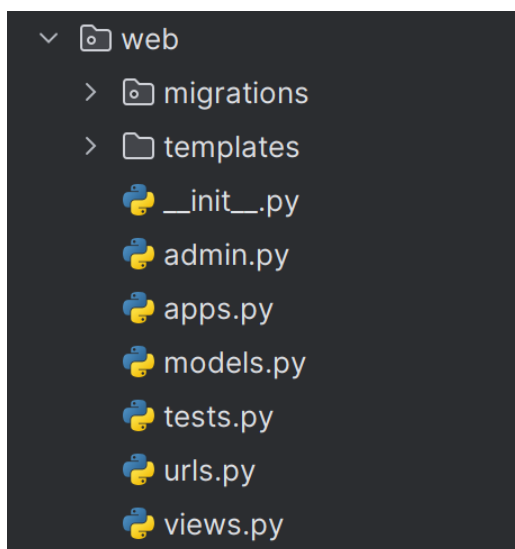
```

Administração backend



Aplicação Web

Estrutura Web – Página principal



Frontend

Página Principal – index.html

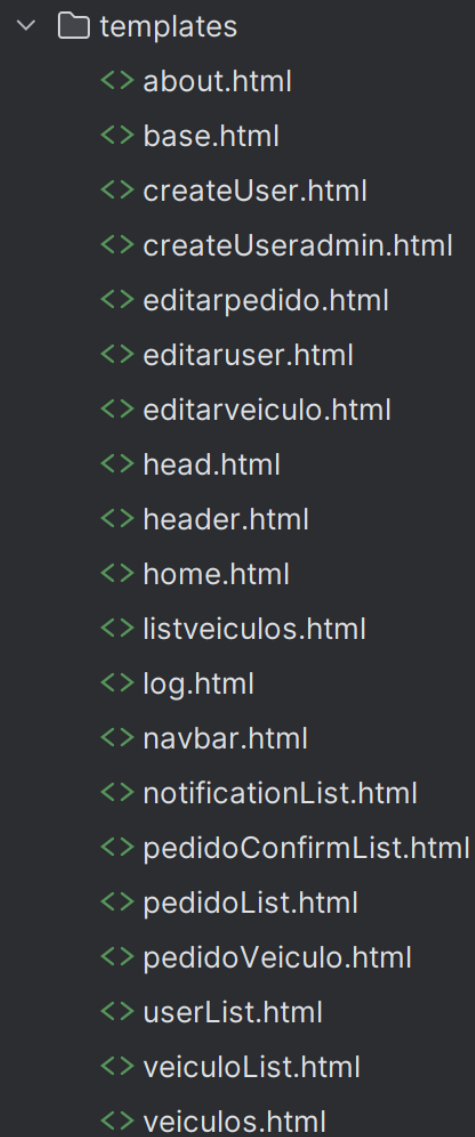
<http://127.0.0.1:8000>

✓ templates
<> index.html



Estrutura frota conjunto de ficheiros .py

```
> ESTGD
> frota
  > migrations
  > templates
  __init__.py
  admin.py
  apps.py
  models.py
  tests.py
  urls.py
  views.py
```



▼ templates

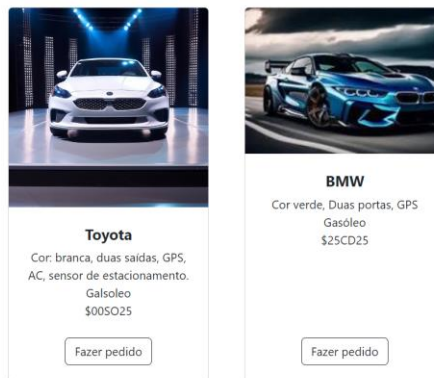
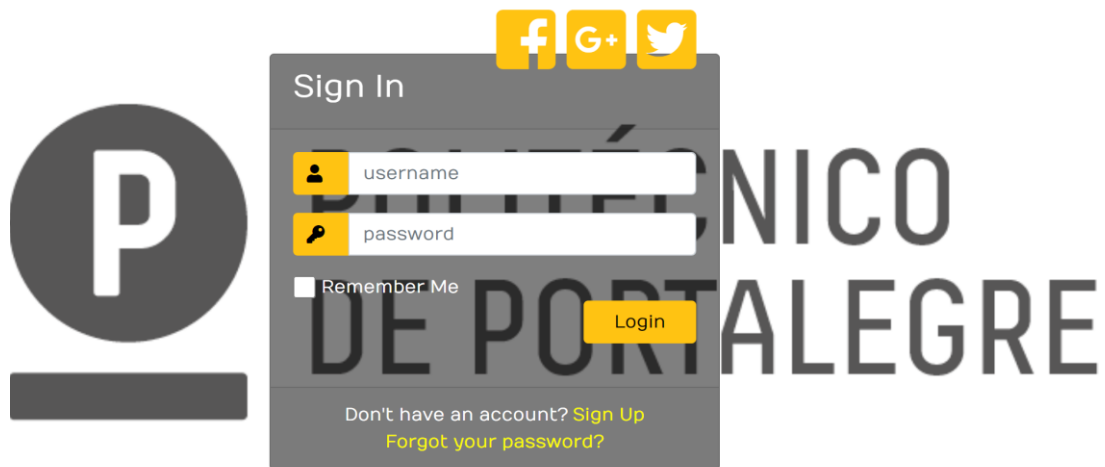
- <> about.html
- <> base.html
- <> createUser.html
- <> createUseradmin.html
- <> editarpedido.html
- <> editaruser.html
- <> editarveiculo.html
- <> head.html
- <> header.html
- <> home.html
- <> listveiculos.html
- <> log.html
- <> navbar.html
- <> notificationList.html
- <> pedidoConfirmList.html
- <> pedidoList.html
- <> pedidoVeiculo.html
- <> userList.html
- <> veiculoList.html
- <> veiculos.html

Funcionalidades

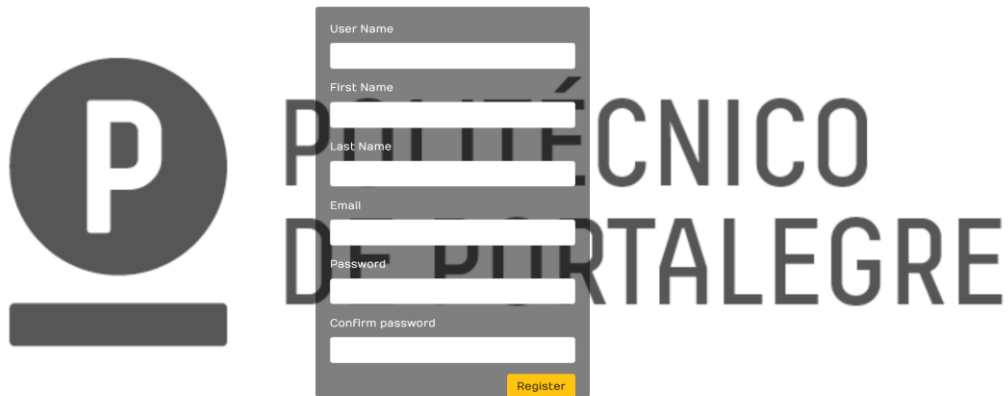
Utilizadores

Administrador e funcionários do Instituto Politécnico de Portalegre (IPP)

Sistema de Login com utilizador registado .




Registar utilizador.



User Name
First Name
Last Name
Email
Password
Confirm password
Register

Barra de navegação – Navebar com utilizador Administrador

Notificações- Permite listar notificações, eliminar notificações e procurar por funcionário.



ESTGD
Welcome to Frota dos Veículos

Frota Home About Notificações Listar Pedidos admin

Search Search

Funcionário	Mensagem	
admin	O funcionario sonia fez pedido do veiculo com destino à Setúbal	Eliminar
admin	O funcionario camoin fez pedido do veiculo com destino à Braga	Eliminar
admin	O funcionario liberata fez pedido do veiculo com destino à Braga	Eliminar

Listar– Lista utilizadores e veículos



ESTGD
Welcome to Frota dos Veículos

Frota Home About Notificações Listar Pedidos admin

- Utilizadores
- Veiculo

Listar Utilizadores – Permite listar os utilizadores e para cada utilizador (editar e eliminar) e um botão adicionar utilizador. Permite também fazer a pesquisa por username de utilizador.

ESTGD

Welcome to Frota dos Veículos

FrotaHomeAboutNotificaçõesListarPedidosadmin

Adicionar

Search

Search

ID	Username	Email	Ação	
1	admin	admin@gmail.com	Editar	Eliminar
2	sonia	sonia@gmail.pt	Editar	Eliminar
36	camoin	antoniopamantec@gmail.com	Editar	Eliminar
37	liberata	antoniopamantec@gmail.com	Editar	Eliminar
46	liberatas	antoniopamantec@gmail.com	Editar	Eliminar

Listar Veículos – Permite listar todos os veículos e para cada veículo (editar e eliminar) e um botão adicionar veículo. Permite também fazer a pesquisa por marca de veículo.

ESTGD

Welcome to Frota dos Veículos

FrotaHomeAboutNotificaçõesListarPedidosadmin

Adicionar

Search

Search

Marca	Matricula	Descrição	tipo_Combustivel		
Toyota	00SO25	Cor: branca, duas saídas, GPS, AC, sensor de estacionamento.	Galsoleo	Editar	Eliminar
BMW	25CD25	Cor verde, Duas portas, GPS	Gasóleo	Editar	Eliminar

Listar Pedidos – Não confirmados ou Confirmados

ESTGD

Welcome to Frota dos Veículos

FrotaHomeAboutNotificaçõesListarPedidosadmin

Não confirmados

Confirmados

Adicionar

Search

Search

Marca	Matricula	Descrição	tipo_Combustivel		
Toyota	00SO25	Cor: branca, duas saídas, GPS, AC, sensor de estacionamento.	Galsoleo	Editar	Eliminar
BMW	25CD25	Cor verde, Duas portas, GPS	Gasóleo	Editar	Eliminar

Listar Pedidos – Permite listar os pedidos não confirmados e para cada pedido permite (confirmar,editar e eliminar).

ESTGD

Welcome to Frota dos Veiculos

Frota

Home

About

Notificações

Listar

Pedidos

admin

ID	Funcionário	Marca de veiculo	Destino	Data de Inicio	Data Final	Ação
8	sonia	Toyota	Setúbal	April 24, 2024	May 3, 2024	<div>Confirmar</div> <div>Editar</div> <div>Eliminar</div>
9	sonia	Toyota	Setúbal	April 24, 2024	April 4, 2024	<div>Confirmar</div> <div>Editar</div> <div>Eliminar</div>
10	sonia	Toyota	Setúbal	April 30, 2024	May 1, 2024	<div>Confirmar</div> <div>Editar</div> <div>Eliminar</div>
11	sonia	Toyota	Setúbal	April 30, 2024	May 1, 2024	<div>Confirmar</div> <div>Editar</div> <div>Eliminar</div>
12	sonia	Toyota	Setúbal	April 30, 2024	May 2, 2024	<div>Confirmar</div> <div>Editar</div> <div>Eliminar</div>

Listar Pedidos – Permite listar os pedidos confirmados e para cada pedido permite (editar e eliminar).

ESTGD

Welcome to Frota dos Veiculos

Frota

Home

About

Notificações

Listar

Pedidos

admin

ID	Funcionário	Marca de veiculo	Destino	Data de Inicio	Data Final	Ação
1	admin	Toyota	Setúbal	April 29, 2024	May 1, 2024	<div>Editar</div> <div>Eliminar</div>
5	sonia	Toyota	Braga	April 30, 2024	April 19, 2024	<div>Editar</div> <div>Eliminar</div>
6	sonia	Toyota	Braga	April 30, 2024	April 30, 2024	<div>Editar</div> <div>Eliminar</div>
7	sonia	Toyota	Setúbal	April 30, 2024	April 26, 2024	<div>Editar</div> <div>Eliminar</div>
19	camoin	Toyota	Braga	April 30, 2024	May 1, 2024	<div>Editar</div> <div>Eliminar</div>
20	liberata	Toyota	Braga	April 30, 2024	May 1, 2024	<div>Editar</div> <div>Eliminar</div>

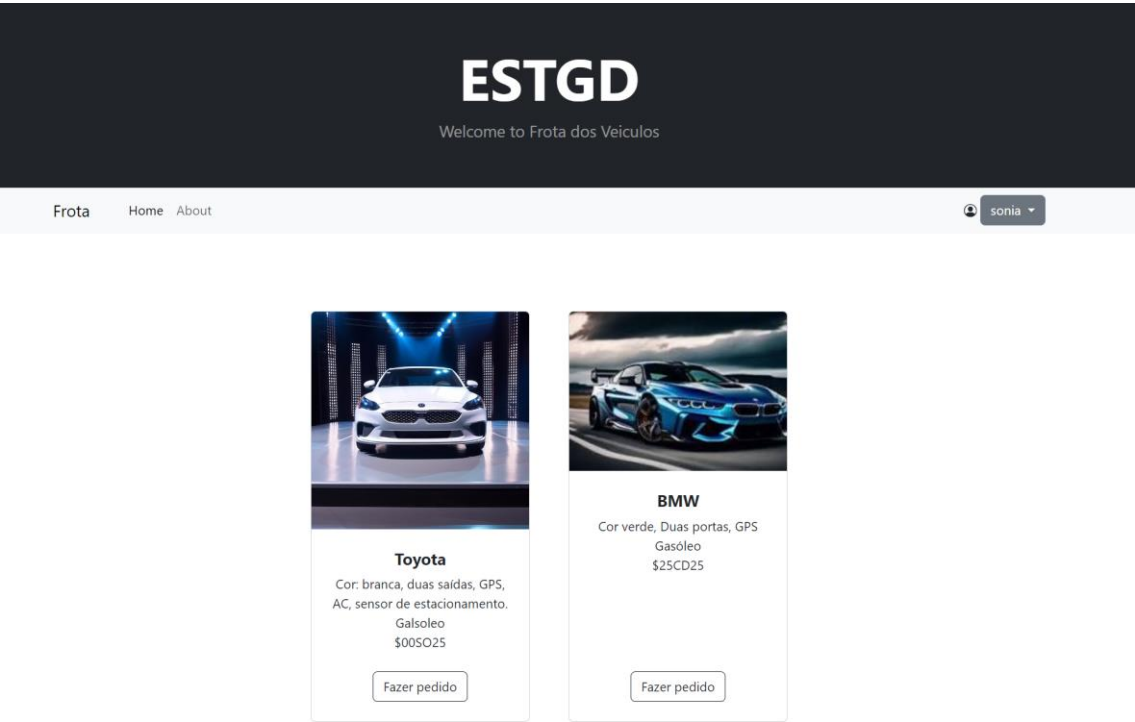
Botão Logout (sair)direciona para página principal index.html.

Página About ou Acerca de Nós



Barra de navegação – Navebar com utilizador Funcionário

O utilizador funcionário pode aceder a página principal “home” onde pode efetuar o pedido de veículo no botão “Fazer Pedido”. Botão Logout (sair) direciona para página index.html.



Botão “Fazer Pedido” aparecerá uma página com o formulário a ser preenchido pelo utilizador e fazer pedido de veículo.

Veiculos

Welcome to Frota dos Veiculos

Frota

Home

About

sonia

Data do início

dd/mm/aaaa

Data Final

dd/mm/aaaa

Local

Combustivel Inicial

Combustivel final

Kilometro inicial

Kilometro final

Submit

Botão Notificações onde o utilizador pode verificar as notificações relativas ao seu pedido.

ESTGD

Welcome to Frota dos Veiculos

Frota


Home

About

sonia

Sair


Notificação



Toyota

Cor: branca, duas saídas, GPS, AC, sensor de estacionamento. Galsoleo \$00SQ25

Fazer pedido



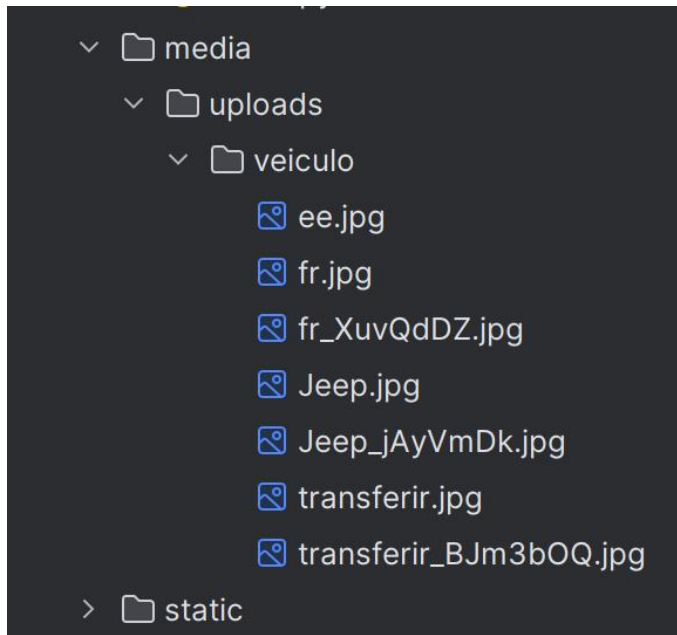
BMW

Cor verde, Duas portas, GPS Gasóleo \$25CD25

Fazer pedido

14

Media



Segurança

O código de resposta de estado de sucesso HTTP 204

No Content indica que um pedido foi bem sucedido, mas que o cliente não precisa de navegar para fora da sua página atual.

```
1 usage
class crudESTGDAPIView(APIView):
    7 usages (7 dynamic)
    def delete(self, request, pk):
        pedido = get_object_or_404(Pedido, pk=pk, user=request.user)
        pedido.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

Status=status.HTTP 400 BAD REQUEST

O código de status de resposta do HyperText Transfer Protocol (HTTP) 400 Bad Request indica que o servidor não pode ou não irá processar a solicitação devido a algo que é percebido como um erro do cliente (por exemplo, sintaxe de solicitação malformada, enquadramento de mensagem de solicitação inválida).

```

return Response(serializer.data)

def post(self, request):
    request.data['user'] = request.user.id
    serializer = PedidoSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

```

get_object_or_404

É uma função de atalho fornecida pelo Django, que é usada para obter um objeto de uma base de dados utilizando uma gestão de modelo e gerar uma exceção `Http404` se o objeto não for encontrado. É também utilizado nas visualizações para obter um único objeto com base numa consulta e lidar com o caso em que o objeto não existe.

← → ↺ ⓘ 127.0.0.1:8000/api/pedidos/jhkgjhg

Page not found (404)

Request Method: GET
Request URL: http://127.0.0.1:8000/api/pedidos/jhkgjhg

Using the URLconf defined in `ESTGD.ur1s`, Django tried these URL patterns, in this order:

1. admin/
2. frota/
3. api/ pedidos/
4. api/ pedidos/<int:pk>/
5. api/ veiculos/
6. api/ veiculos/<int:pk>/
7. api/ utilizador/
8. api/ utilizador/<int:pk>/
9. api/ login/
10. web/
11. api/auth
12. ^media/(?P<path>.*)\$

The current path, `api/pedidos/jhkgjhg`, didn't match any of these.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

← → ↺ ⓘ 127.0.0.1:8000/frota/find/?foo=ghghghghghg

Page not found (404)

No User matches the given query.

Request Method: GET
Request URL: http://127.0.0.1:8000/frota/find/?foo=ghghghghghg
Raised by: `frota.views.find`

Using the URLconf defined in `ESTGD.ur1s`, Django tried these URL patterns, in this order:

1. admin/
2. frota/ [name='home']
3. frota/ login/ [name='login']
4. frota/ logout_app/ [name='logout']
5. frota/ about/ [name='about']
6. frota/ veiculoregister/ [name='veiculoregister']
7. frota/ listveiculos/ [name='listveiculos']
8. frota/ registeruser/ [name='registeruser']
9. frota/ fazerpedido/<int:veiculo_id>/<int:id_user>/ [name='fazerpedido']
10. frota/ registrarpedido/<int:veiculo_id>/ [name='registrarpedido']
11. frota/ veiculolist/ [name='veiculolist']
12. frota/ pedidolist/ [name='pedidolist']
13. frota/ editarveiculo/<int:pk>/ [name='editarveiculo']
14. frota/ eliminarveiculo/<int:pk>/ [name='eliminarveiculo']
15. frota/ veiculoeditado/<int:pk>/ [name='veiculoeditado']
16. frota/ userList/ [name='userList']
17. frota/ editaruser/<int:pk>/ [name='editaruser']
18. frota/ editarpedido/<int:pk>/ [name='editarpedido']
19. frota/ eliminaruser/<int:pk>/ [name='eliminaruser']
20. frota/ eliminarpedido/<int:pk>/ [name='eliminarpedido']
21. frota/ usereditado/<int:pk>/ [name='usereditado']
22. frota/ find/ [name='find']

The current path, `frota/find/`, matched the last one.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

Utilização do Try catch

O try-catch é uma estrutura fundamental para lidar com exceções (erros) que podem ocorrer durante a execução do código.

Em termos de segurança, o try-catch oferece uma maneira de controlar o fluxo do programa quando um erro acontece, em vez de simplesmente deixar o programa falhar completamente. Isso ajuda a garantir que o programa possa lidar com condições inesperadas sem quebrar ou fornecer mensagens de erro confusas para o utilizador final.

Controle de Erros: O try-catch permite a captura exceções específicas e tome medidas apropriadas para lidar com elas, seja registrando o erro ou informando o utilizador.

Encripatação da password

```
Hashed_password = make_password(password)
```

```
User= User(....., password=hashed-password)
```

Bootstrap

Utilização do um template para tabelas

<https://getbootstrap.com/docs/5.3/content/tables/#overview>

Utilização de template para aplicação web e página principal index.html.

<https://startbootstrap.com/theme/business-casual>

Django API RestFul- Criar API do projeto

Comandos

```
python manage .py Startapp api
```

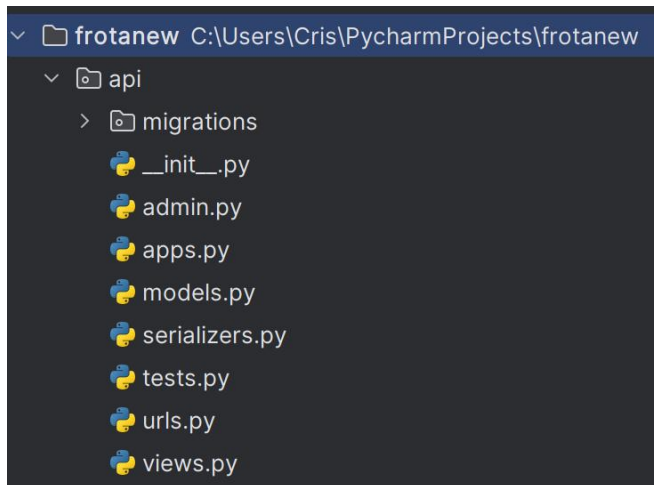
criar ficheiro serializers

adicionar no projeto frota – URLS.py adicionar o caminho path

Criar import no ficheiro serializers.py

Instalação do django rest_framework serializers

Estrutura



VIEWS.PY GET/POST para PEDIDO

Class Pedido - Função definição GET/POST (pedido)

```
class ListESTGDAPIView(APIView):
    def get(self, request):
        pedido = Pedido.objects.filter(user=request.user.id)
        serializer = PedidoSerializer(pedido, many=True)
        return Response(serializer.data)

    def post(self, request):
        request.data['user'] = request.user.id
        serializer = PedidoSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status.HTTP_201_CREATED)
        return Response(serializer.errors,
                        status=status.HTTP_400_BAD_REQUEST)
```

Class veículo – função GET/PUT

```
class VeiculoAPIView(APIView):
    def get(self, request, pk):
        veiculo = get_object_or_404(Veiculo, pk=pk, user=request.user)
        serializer = VeiculoSerializer(veiculo, many=False)
        return Response(serializer.data)

    def put(self, request, pk):
        veiculo = get_object_or_404(Veiculo, pk=pk)
        serializer = VeiculoSerializer(veiculo, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors,
                        status=status.HTTP_400_BAD_REQUEST)
```

Função definição DELETE/PUT PEDIDO

```
class crudESTGDAPIView(APIView):
    def delete(self, request, pk):
```

```

        pedido = get_object_or_404(Pedido, pk=pk, user=request.user)
        pedido.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)

    def put(self, request, pk):
        pedido = get_object_or_404(Pedido, pk=pk, user=request.user)
        serializer = PedidoSerializer(pedido, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors,
                        status=status.HTTP_400_BAD_REQUEST)

```

DELETE

Função definição DELETE Veiculo

```

def delete(self, request, pk):
    veiculo = get_object_or_404(Veiculo, id=pk, user=request.user)
    veiculo.delete()
    return Response(status=status.HTTP_204_NO_CONTENT)

```

Class Listar Veiculos – Função GET/POST

```

class ListVeiculoESTGDAPIView(APIView):
    def get(self, request):
        veiculo = Veiculo.objects.all()
        serializer = VeiculoSerializer(veiculo, many=True)
        return Response(serializer.data)

    def post(self, request):
        request.data['user'] = request.user.id
        serializer = VeiculoSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors,
                        status=status.HTTP_400_BAD_REQUEST)

```

Class CRUD Veículos – DELETE/PUT

```

class crudVeiculoESTGDAPIView(APIView):
    def delete(self, request, pk):
        veiculo = get_object_or_404(Pedido, pk=pk,
user=request.user)
        veiculo.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)

    def put(self, request, pk):
        veiculo = get_object_or_404(Veiculo, pk=pk)
        serializer = VeiculoSerializer(veiculo, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)

```

```

        return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)

```

Class Lista utilizadores – GET/POST

```

class ListUtilizadorESTGDAPIView(APIView):
    def get(self, request):
        utilizador = User.objects.all()
        serializer = UserSerializer(utilizador, many=True)
        return Response(serializer.data)

    def post(self, request):
        request.data['user'] = request.user.id
        serializer = UserSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data,
status.HTTP_201_CREATED)
        return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)

```

Class CRUD Utilizadores – DELETE/PUT

```

class crudUtilizadorESTGDAPIView(APIView):
    def delete(self, request, pk):
        utilizador = get_object_or_404(User, pk=pk)
        utilizador.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)

    def put(self, request, pk):
        utilizador = get_object_or_404(User, pk=pk)
        serializer = UserSerializer(utilizador, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)

```

Serializers.py

Class Veículo;User;Pedido

```

class VeiculoSerializer(serializers.ModelSerializer):
    class Meta:
        model = Veiculo
        fields = '__all__'

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = '__all__'

class PedidoSerializer(serializers.ModelSerializer):
    class Meta:
        model = Pedido
        fields = '__all__'

```

Interface API

<http://127.0.0.1:8000/api>

Django REST framework

Username:

Password:

Log in

Tipo de Utilizador	Ações
Administrador	GET;POST;PUT;DELETE (Utilizador;Veículos;Pedidos)
Funcionário	GET;POST;PYUT;DELETE (Pedido)

Listar veículos (POST/GET)

POST – Colocar inserir veículos

GET – Obter todos os veículos

PUT – Atualizar dados dos veículos

DELETE – Apagar Veículo

Django REST frameworkadmin, True

List Veiculo Estgdapi

OPTIONSGET

GET /api/veiculos/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[
  {
    "id": 5,
    "matricula": "253095",
    "marca": "Toyota",
    "tipo_combustivel": "Gasol\u00edo",
    "description": "175 cm de largura; 8 A altura corresponde a 60% da largura; 8 A estrutura \u00c9 radial (R); 8 A jante \u00c9 de 14 polegadas; 8 Admite at\u00e9 387 kg de peso; i",
    "image": "/media/uploads/veiculo/transferir.jpg"
  },
  {
    "id": 6,
    "matricula": "905A00",
    "marca": "Ford",
    "tipo_combustivel": "Gasol\u00edo",
    "description": "Car azul dois entradas",
    "image": "/media/uploads/veiculo/fr_kuvQ002.jpg"
  }
]
```

Media type: application/json

Content:

POST

Listar utilizadores (POST/GET)

POST – Colocar inserir utilizadores

GET – Obter todos os utilizadores

PUT – Atualizar dados dos utilizadores

DELETE – Apagar utilizador

Django REST framework

admin, True

List Utilizador Estgdapi

OPTIONS

GET

GET /api/utilizador/

HTTP 200 OK

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
[
  {
    "id": 1,
    "password": "pbkdf2_sha256$7200005$0gJ6ubwIAZImhfvIhkAA3c$6ydl8Z1ThR/JoQA4q9/H2Apf0FmaZCUB5A2z8YyoIPV=",
    "last_login": "2024-04-20T13:38:41.381535Z",
    "is_superuser": true,
    "username": "frota",
    "first_name": "",
    "last_name": "",
    "email": "frota@gmail.com",
    "is_staff": false,
    "is_active": true,
    "date_joined": "2024-04-17T18:43:49Z",
    "groups": [],
    "user_permissions": []
  },
  {
    "id": 2,
    "password": "pbkdf2_sha256$7200005$0gJ6ubwIAZImhfvIhkAA3c$6ydl8Z1ThR/JoQA4q9/H2Apf0FmaZCUB5A2z8YyoIPV=",
    "last_login": "2024-04-29T16:38:51.659046Z",
    "is_superuser": false,
    "username": "sonia",
    "first_name": "",
    "last_name": "",
    "email": "sonia@gmail.pt",
    "is_staff": false,
    "is_active": true,
    "date_joined": "2024-04-17T18:44:44.377348Z",
    "groups": [],
    "user_permissions": []
  },
  {
    "id": 4,
    "password": "pbkdf2_sha256$7200005$a4ThwpFtCNownd4ysXsHhps$10mn3YtntvViq5bmIcymRLqnrod+w9STxwQbA/f48ZY=",
    "last_login": "2024-04-29T16:42:50.181559Z",
    "is_superuser": true,
    "username": "admin",
    "first_name": "admin",
    "last_name": "admin",
    "email": "admin@gmail.pt",
    "is_staff": true,
    "is_active": true,
    "date_joined": "2024-04-20T12:10:59Z",
    "groups": [],
    "user_permissions": []
  },
  {
    "id": 5,
    "password": "pbkdf2_sha256$7200005$3Qs5rrCFwGuwPKhoBARwt6$0taN0k55/W06+xxdHE11+CmlF6baxGrsk/95JjWfIENQ=",
    "last_login": "2024-04-20T15:52:33.956981Z",
    "is_superuser": false,
    "username": "camoioipamante",
    "first_name": "Camoioin ",
    "last_name": "Pamante ",
    "email": "antonioipamantec@gmail.com",
    "is_staff": false,
    "is_active": true,
    "date_joined": "2024-04-20T15:52:33.925410Z",
    "groups": [],
    "user_permissions": []
  },
  {
    "id": 8,
```

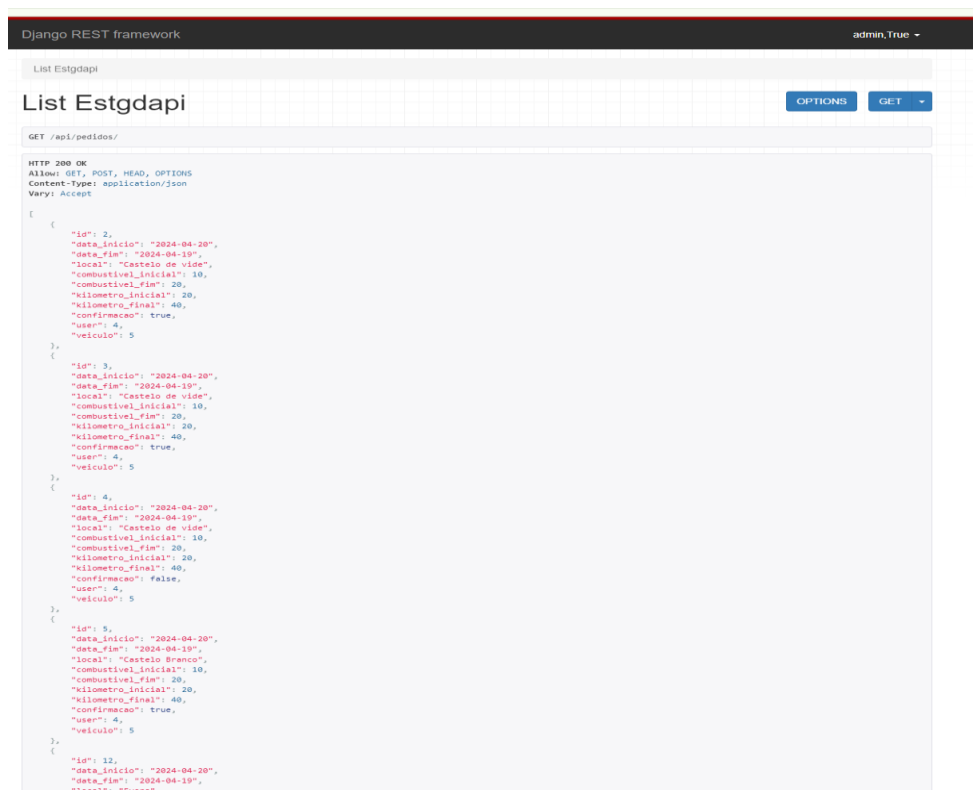
Listar Pedidos (POST/GET)

POST – Colocar inserir pedidos

GET – Obter todos os pedidos

PUT – Atualizar dados dos pedidos

DELETE – Apagar pedido



The screenshot shows the Django REST framework interface for the 'List Estgdapi' endpoint. The URL is 'GET /api/pedidos/'. The response is an HTTP 200 OK with the following headers: 'Allow: GET, POST, HEAD, OPTIONS', 'Content-Type: application/json', and 'Vary: Accept'. The response body is a JSON array of 5 objects, each representing an order. The first four objects have 'confirmacao' set to true, and the fifth object has 'confirmacao' set to false. The objects contain fields for 'id', 'data_inicio', 'data_fim', 'local', 'combustivel_inicial', 'combustivel_fim', 'quilometro_inicial', 'quilometro_fim', 'user', and 'veiculo'.

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
  {
    "id": 2,
    "data_inicio": "2024-04-20",
    "data_fim": "2024-04-19",
    "local": "Castelo de vide",
    "combustivel_inicial": 10,
    "combustivel_fim": 20,
    "quilometro_inicial": 20,
    "quilometro_fim": 40,
    "confirmacao": true,
    "user": 4,
    "veiculo": 5
  },
  {
    "id": 3,
    "data_inicio": "2024-04-20",
    "data_fim": "2024-04-19",
    "local": "Castelo de vide",
    "combustivel_inicial": 10,
    "combustivel_fim": 20,
    "quilometro_inicial": 20,
    "quilometro_fim": 40,
    "confirmacao": true,
    "user": 4,
    "veiculo": 5
  },
  {
    "id": 4,
    "data_inicio": "2024-04-20",
    "data_fim": "2024-04-19",
    "local": "Castelo de vide",
    "combustivel_inicial": 10,
    "combustivel_fim": 20,
    "quilometro_inicial": 20,
    "quilometro_fim": 40,
    "confirmacao": false,
    "user": 4,
    "veiculo": 5
  },
  {
    "id": 5,
    "data_inicio": "2024-04-20",
    "data_fim": "2024-04-19",
    "local": "Castelo Branco",
    "combustivel_inicial": 10,
    "combustivel_fim": 20,
    "quilometro_inicial": 20,
    "quilometro_fim": 40,
    "confirmacao": true,
    "user": 4,
    "veiculo": 5
  },
  {
    "id": 12,
    "data_inicio": "2024-04-20",
    "data_fim": "2024-04-19",
    "local": "Evora",
    "combustivel_inicial": 10,
    "combustivel_fim": 20,
    "quilometro_inicial": 20,
    "quilometro_fim": 40,
    "confirmacao": true,
    "user": 4,
    "veiculo": 5
  }
]
```

Considerações Finais

Deparámo-nos com alguns desafios que enfrentamos durante o desenvolvimento trabalho de Integração de Sistemas, como por exemplo, alguma complexidade com o desenrolar das ideias que iam surgindo, e também com falhas a nível do software “Pycharm” a meio do processo ambos tivemos de reinstalar o programa e também na partilha de commits no “GitHub”. Este trabalho foi todo desenvolvido por ambos, trabalhámos na aplicação web e na API Restful.

Relativamente aos resultados alcançados, é de destacar como pontos positivos:

Uma aplicação funcional;

Sistema login;

Confirmação dos pedidos de veículo com notificações para o utilizador funcionário;

Permite pesquisa por nome;

As áreas que ainda podem ser melhoradas são:

Campo pesquisa por exemplo pesquisar por letra;

Efetuar um sistema de login para a API RestFul;

Passagem do atributo imagem pelo API RestFul.

Salientar a importância de considerações éticas e de segurança ao integrar sistemas, garantindo a proteção adequada dos dados e o cumprimento de regulamentações relevantes, como a GDPR (Regulamentação Geral de Proteção de Dados).