

**Relatório Projeto Final**

**Licenciatura em Engenharia Informática 3º ano, 1º semestre**

**Realizado em Base de Dados II Por**

**20223 – Alexandre Moreira**

**20255 - Carlos Silva**

**17852 – Gonçalo Marques**

**Relatório Projeto Final**

**Licenciatura em Engenharia Informática 3º ano, 1º semestre**

**Realizado em Base de Dados II Por**

**20223 – Alexandre Moreira**

**20255 - Carlos Silva**

**17852 – Gonçalo Marques**

**Docentes: Paulo Tomé, Pedro Martins, Paulo Costa Entidade: Base de Dados 2**

**Índice**

[1. Projeto 6](#_Toc144848038)

[1.1 Estrutura 6](#_Toc144848039)

[2 Diagrama 14](#_Toc144848040)

[3 Base de Dados 16](#_Toc144848041)

[3.1 PostgreSQL 16](#_Toc144848042)

[3.1.1 Encomendas 16](#_Toc144848043)

[3.1.2 Itens Encomendas 17](#_Toc144848044)

[3.2 MongoDB 17](#_Toc144848045)

[3.2.1 Utilizadores 18](#_Toc144848046)

[3.2.2 Produtos 18](#_Toc144848047)

[3.2.3 Carrinho 19](#_Toc144848048)

[3.2.4 Promoções 19](#_Toc144848049)

[3.2.5 Tipos de Produtos 19](#_Toc144848050)

[3.3 Funções 20](#_Toc144848051)

[3.3.1 GetOrdersByCustomer() 20](#_Toc144848052)

[3.3.2 GetOrder() 20](#_Toc144848053)

[3.3.3 getTop5MostSoldProductsByUser() 21](#_Toc144848054)

[3.3.4 getMostSoldProductByType() 21](#_Toc144848055)

[3.3.5 getMostSoldProductByPartner() 22](#_Toc144848056)

[3.3.6 getSoldProductByPartner() 22](#_Toc144848057)

[3.3.7 getOrdersByOnePartner() 23](#_Toc144848058)

[3.4 Procedimentos 23](#_Toc144848059)

[3.4.1 Orders\_insert() 23](#_Toc144848060)

[3.4.2 OrderItems\_insert() 24](#_Toc144848061)

[3.4.3 Update\_OrderStatus() 24](#_Toc144848062)

[3.4.4 UpdateAll\_OrderStatus() 25](#_Toc144848063)

[3.4.5 Orders\_delete() 25](#_Toc144848064)

[3.5 Trigger 25](#_Toc144848065)

[3.6 Views 27](#_Toc144848066)

[3.6.1 getMostPopularProduct() 27](#_Toc144848067)

[3.6.2 getMostPopularProductThisWeek() 27](#_Toc144848068)

[3.6.3 getTop5MostSoldProduct() 27](#_Toc144848069)

[3.6.4 getUsersWithMoreOrdersAndHowMany() 28](#_Toc144848070)

[3.6.5 getUsersOrdersAndHowMany() 28](#_Toc144848071)

[3.6.6 ordersView() 28](#_Toc144848072)

[3.6.7 getOrderStatusFalse() 29](#_Toc144848073)

[3.6.8 getCountOrderStatusFalse() 29](#_Toc144848074)

[4 Aplicação Web 30](#_Toc144848075)

[4.1 Utilizadores 30](#_Toc144848076)

[4.2 Autenticação de utilizadores 31](#_Toc144848077)

[4.2.1 Funções 32](#_Toc144848078)

[4.3 Navbar 34](#_Toc144848079)

[4.4 HomePage 34](#_Toc144848080)

[4.4.1 Utilizadores não autenticados 34](#_Toc144848081)

[4.4.2 Clientes 35](#_Toc144848082)

[4.4.3 Comerciais 36](#_Toc144848083)

[4.4.4 Parceiros 37](#_Toc144848084)

[4.4.5 Administradores 38](#_Toc144848085)

[4.5 Listagem de produtos 39](#_Toc144848086)

[4.6 Criação de produtos 41](#_Toc144848087)

[4.6.1 Adicionar tipo de produto 41](#_Toc144848088)

[4.6.2 Adicionar um novo produto 42](#_Toc144848089)

[4.7 Alteração de produtos 43](#_Toc144848090)

[4.8 Encomendas 44](#_Toc144848091)

[4.9 Carrinho 46](#_Toc144848092)

[4.9.1 Funções do carrinho 46](#_Toc144848093)

[4.10 Promoções 48](#_Toc144848094)

[5 Conclusão 50](#_Toc144848095)

## Projeto

No âmbito da cadeira de Base de Dados II, foi no pedido para desenvolvermos loja digital. A mesma esta dividida em duas partes: consulta dos produtos e aquisição dos mesmos. Para esta última, o utilizador necessita de efetuar login com as suas respetivas credenciais para continuar as operações. Estas duas etapas diferem apenas na informação a que os utilizadores têm acesso, isto é, no caso da consulta de produtos, todos os utilizadores têm acesso aos produtos disponibilizados no site, no caso da aquisição dos mesmos, como o utilizador tem de colocar os seus dados pessoais, apenas este tem acesso a estes.

É de notar que os produtos existentes na aplicação podem ser de duas origens: da própria loja ou de um parceiro externo, sendo que no caso deste último, o preço é estipulado previamente.

Cada utilizador pode aceder ao estado das suas compras, desde que esteja autenticado, estejam estas ainda em andamento ou já efetuadas. Por último, existe ainda uma entidade responsável por adicionar, editar e eliminar produtos.

Para o desenvolvimento deste projeto, utilizamos tecnologias como *PostgreSQL*, *MongoDB*, *Django* e a livraria *Pymongo.*

## Estrutura

Este relatório encontra-se dividido em cinco partes. No primeiro capítulo, fazemos uma contextualização do trabalho proposto assim como uma definição dos objetivos do mesmo.

No segundo capítulo, explicar-se-á estrutura da base de dados.

No terceiro capítulo, encontra-se as diversas funções utilizadas dentro da própria base de dados.

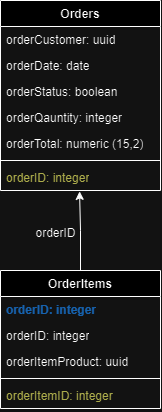
No quarto capítulo, encontra-se a implementação do projeto, os seus requisitos e a devida explicação de como o mesmo opera.

No quarto e último capítulo, encontrar-se a conclusão, explicando as dificuldades encontradas ao longo do projeto assim como a maneira como foram ultrapassadas e algumas reflexões do grupo.

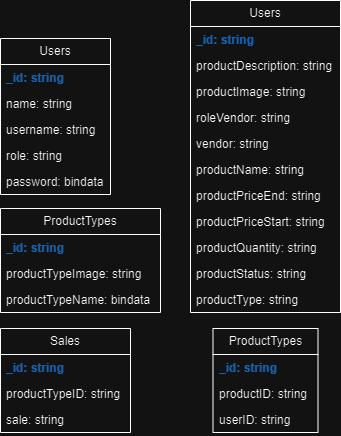
# Diagrama

A primeira coisa decidida sobre o projeto foi a estrutura do mesmo, e em seguida apresenta se a mesma.

Como em *PostgreSQL* apenas criamos as tabelas relacionais transacionais, é apresentado o diagrama das mesmas:



Já em *MongoDB*, criou-se as seguintes *collections* que são referentes a operações não transacionais.



# Base de Dados

De modo a dar suporte ao *website*, utilizamos os motores de base de dados *PostgreSQL* e *MongoDB*. Desta forma, todas as operações que envolvam transações foram implementadas em *PostgreSQL*, isto é, encomendas e respetivas linhas da encomenda, sendo que este último faz referência aos itens existentes em cada encomenda.

De forma a controlar os utilizadores, produtos, promoções, carrinho implementámos as *collections* necessárias para o efeito em *MongoDB*.

## PostgreSQL

PostgreSQL é um sistema de gestão de base de dados relacional que utiliza tabelas e modelos como forma de armazenamento dos dados. O PostgreSQL oferece funcionalidades avançadas de segurança, desempenho e extensibilidade que o tornam ideal para o desenvolvimento de projetos como este.

### Encomendas

De forma a armazenar as encomendas criadas pelos clientes, criou-se a tabela “*Orders”* em que se armazenam dados referentes ao utilizador que fez a encomenda (id do mesmo), a data em que foi feita, o total a pagar, a quantidade de produtos e o estado da encomenda.

A screenshot of a computer code

Description automatically generated

### Itens Encomendas

Para especificar os produtos presentes em cada uma das encomendas realizadas pelos clientes, criou-se a tabela *“OrderItems”*, que contém um ID próprio, o ID da encomenda a que se refere, o ID de cada produto presente em cada linha e o seu preço.

A white background with black text

Description automatically generated

## MongoDB

O MongoDB é um sistema de gestão de base de dados NoSQL de código aberto. Ao contrário dos sistemas de base de dados relacionais tradicionais, que armazenam dados em tabelas com esquemas predefinidos, o MongoDB guarda dados em formato de documentos JSON flexíveis, o que o torna adequado para guardar dados semiestruturados ou não estruturados.

Desta forma, consegue obter-se mais velocidade e disponibilidade no manuseio da base de dados, porém, é necessário utilizar linguagens de baixo nível (python, por exemplo).

### Utilizadores

No caso dos utilizadores, adicionamos a respetiva *collection* no ato da criação de conta. Para o efeito, registamos qual o nome do utilizador, o *username* (identificação pela qual faz login) e a respetiva *password*. Caso seja um administrador a criar a conta, este pode escolher que tipo de conta que pretende criar.

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

Descrição gerada automaticamente

### Produtos

Relativamente aos produtos, armazenamos na *collection* correspondente, alguns dados como o nome do produto, preço, tipo de produto, entre outros dados.

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

Descrição gerada automaticamente

### Carrinho

Para o cliente poder comprar vários items ao mesmo tempo em vez de os comprar 1 a 1, decidimos desde cedo que a criação de um carrinho era necessária. Na respetiva *collection* são guardados dados relativamente ao cliente em questão e aos produtos selecionados pelo mesmo.

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

Descrição gerada automaticamente

### Promoções

Cada promoção esta associada a um tipo de produtos facilitando assim a criação de promoções para vários produtos do mesmo tipo.

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

Descrição gerada automaticamente

### Tipos de Produtos

Para podermos organizar os produtos assim como dar lhes os respetivos descontos criamos a *collection* *productType*.

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

Descrição gerada automaticamente

## Funções

Neste subcapítulo, são apresentadas as funções criadas no *PostgreSQL*, de forma a obter os dados desejados das tabelas existentes.

### GetOrdersByCustomer()

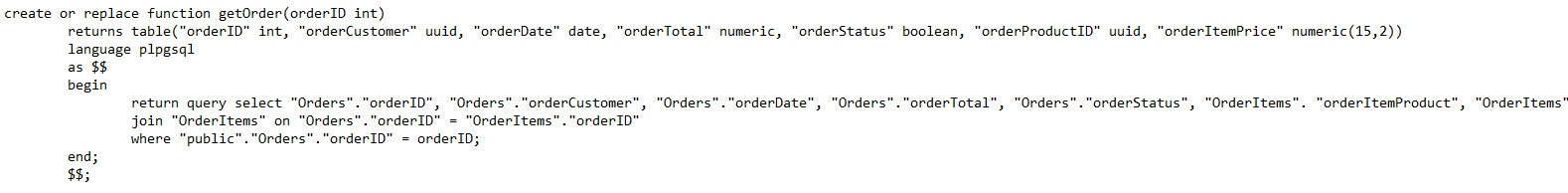
A função apresentada em baixo recebe, por parâmetro, o *id* do cliente do qual se pretende analisar as encomendas e retorna, através de uma query, as encomendas feitas pelo mesmo.

A screenshot of a computer code

Description automatically generated

### GetOrder()

Dado o *id* de uma encomenda, esta retorna os dados correspondentes à mesma.



### getTop5MostSoldProductsByUser()

A função seguinte retorna quais os cinco produtos mais vendidos de um determinado utilizador. Sendo que o *id* desse utilizador é passado como parâmetro.

*A screenshot of a computer code

Description automatically generated*

### getMostSoldProductByType()

Esta função recebe como parâmetro um array com os *ids* dos produtos associados a um determinado tipo de produtos e retorna o *id* daquele que é mais comprado pelos clientes (mais vendido pela loja).

A screenshot of a computer code

Description automatically generated

### getMostSoldProductByPartner()

A seguinte função retorna quais o cinco produtos mais comprados pelos clientes, tendo em conta o parceiro que o vende, para isso, esta recebe como parâmetro um array de *ids de* produtos e, após isso, faz uma query de forma a retornar quais os produtos mais comprados pelos clientes e o número de vezes que foram comprados.

A computer code with black text

Description automatically generated

### getSoldProductByPartner()

Para saber qual o produto mais vendido por parceiro, criou-se a seguinte função que, recebendo como parâmetro um array de *ids de* produtos retorna qual os produto mais vendido pelo parceiro e o número de vezes que foi comprado.

A computer code with black text

Description automatically generated

### getOrdersByOnePartner()

Recebendo como parâmetro um array de *ids* de produtos, retorna os produtos e respetivo preço a pagar.

A screenshot of a computer code

Description automatically generated

## Procedimentos

### Orders\_insert()

Para que seja possível inserir encomendas nas tabelas criadas em PostgresSQL, foi criado o procedimento Orders\_insert() que recebe como variáveis de entrada um array de ID’s de produtos, o ID do cliente em causa e um array com o preço de cada produto. Neste caso é inserida uma nova linha na tabela “Orders” e faz-se uma consulta para obter o ID dos últimos dados inseridos na mesma. Após isso, é chamado outro procedimento (OrderItems\_insert()).

A computer screen shot of a code

Description automatically generated

### OrderItems\_insert()

O procedimento OrderItems\_insert(), por sua vez, recebe como variáveis de entrada o orderID criado anteriormente, o array de ID’s de produtos e o um array com o preço dos mesmos. No procedimento existe um ciclo for que irá percorrer o array de ID’s de produtos e irá, para cada elemento do mesmo, inserir novos dados na tabela “OrderItems”.

A screen shot of a computer code

Description automatically generated

### Update\_OrderStatus()

O procedimento Update\_OrderStatus() recebe por parâmetro um array de ID’s de encomenda. Existe um ciclo for que percorre o array e para cada linha que contenha este ID irá alterar a variável “OrderStatus” para true. Isto serve para alterar o estado de uma encomenda.

A screenshot of a computer code

Description automatically generated

### UpdateAll\_OrderStatus()

Este procedimento foi criado para atualizar todas as encomendas cuja variável “orderStatus” seja “false”, para “true”. Isto servirá para alterar o estado de todas as encomendas.

A screenshot of a computer program

Description automatically generated

### Orders\_delete()

Este procedimento permite a eliminação de uma “Order” conforme o orderID fornecido.

A screenshot of a computer code

Description automatically generated

## Trigger

Este trigger (“Orders\_insert\_trigger”) foi criado para atuar após a inserção de dados na tabela “OrderItems”. Neste caso, quando são inseridos dados na tabela, é executada a função “OrdersItem\_insertTrigger()”. Esta função retorna “trigger” e tem duas variáveis: “totalProductItem” e “totalPrice”. A variável “totalPrice” guarda a soma do preço de todos os produtos inseridos na tabela “OrderItems” em que o ID da encomenda seja igual ao ID da nova encomenda. A variável “totalProductItem” guarda o número total de itens inseridos na tabela “OrderItems” cujo ID da encomenda é igual ao ID da nova encomenda. Por fim são atualizados os campos “orderTotal” e “orderQuantity” na tabela “Orders” com os valores das variáveis “totalPrice” e “totalProductItem” respetivamente.

A computer screen shot of a computer code

Description automatically generated

## Views

### getMostPopularProduct()

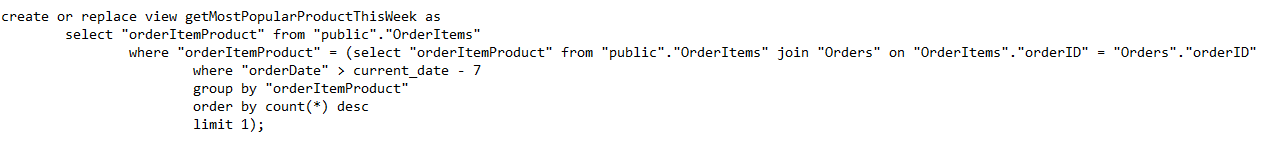
A view “getMostPopularProduct()” apresenta o produto mais vendido de sempre.

A white background with black text

Description automatically generated

### getMostPopularProductThisWeek()

A view apresentada apresenta qual o produto mais vendido no espaço de uma semana.



### getTop5MostSoldProduct()

A view “getTop5MostSoldProduct()” apresenta os 5 produtos mais vendidos de sempre.

A black text on a white background

Description automatically generated

### getUsersWithMoreOrdersAndHowMany()

A view “getUsersWithMoreOrdersAndHowMany()” apresenta o cinco utilizadores com mais encomendas feitas e a quantidade das mesmas.

A black text on a white background

Description automatically generated

### getUsersOrdersAndHowMany()

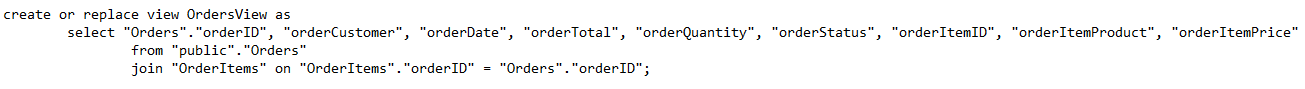
A view “getUsersOrdersAndHowMany() apresenta todas as encomendas feitas por cada utilizador de um só produto e quantas vezes o encomendou.

A black text on a white background

Description automatically generated

### ordersView()

A view “ordersView()” apresenta todos os dados sobre todas as encomendas feitas desde sempre.



### getOrderStatusFalse()

A view “getOrderStatusFalse” apresenta quais as encomendas que ainda não foram expedidas para a casa dos clientes, ou seja, que têm o “orderStatus” a false.

A black text on a white background

Description automatically generated

### getCountOrderStatusFalse()

A view “getCountOrderStatusFalse()” apresenta o número de encomendas cujo estado é “false”, ou seja, que ainda não foram expedidas.

*A black text on a white background

Description automatically generated*

# Aplicação Web

## Utilizadores

Este projeto apresenta diversos tipos de usuários, cada um com funções específicas, a fim de organizar o site de forma adequada. Estes utilizadores são:

**Administrador**: O administrador é responsável pela gestão dos produtos e dos usuários do comércio. Além disso, ele tem acesso a uma página de monitoramento com estatísticas do site.

**Comercial de Tipo 1:** Os comerciais são membros da empresa que gerência o site. Os comerciais de tipo 1 podem listar e gerenciar produtos pertencentes à empresa, realizar ações como alterar produtos, criar promoções e excluir produtos.

**Comercial de Tipo 2:** À semelhança dos comerciais de tipo 1, estes também são membros da empresa que gerem o site, mas em contraste, os comerciais de tipo 2 têm acesso a um painel de estatísticas que mostra os produtos mais vendidos, os tipos de produtos mais populares e as promoções ativas.

**Parceiros**: Os parceiros são usuários responsáveis pela gestão de seus próprios produtos. Eles podem adicionar produtos de sua autoria ao site após se registarem como parceiros, permitindo que outros usuários comprem seus produtos.

**Clientes**: A maioria dos usuários são clientes. Eles têm acesso a uma página com recomendações personalizadas, mostrando os produtos mais populares no site, os mais vendidos na semana e as melhores promoções. Os clientes também podem fazer compras no site.

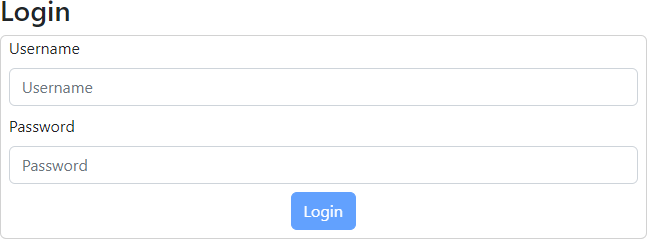
Este sistema de usuários e suas respetivas funções ajuda a manter o site organizado e oferecer uma experiência personalizada aos diferentes tipos de usuários.

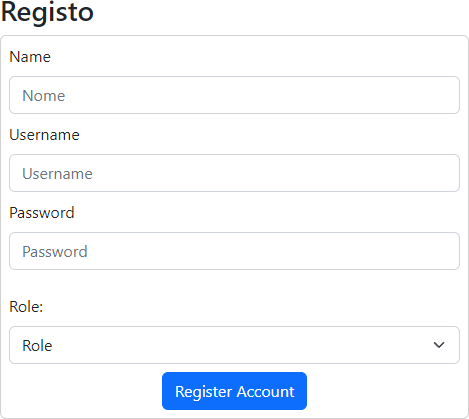
## Autenticação de utilizadores

Para proteção de dados e gestão de encomendas, criou-se um sistema de autenticação de utilizadores.

Assim, qualquer utilizador que esteja autenticado no *website*, consegue efetuar a compra de qualquer que seja o produto disponível no site, bem como consegue acompanhar as suas compras, estejam estas ainda em fase de aquisição ou terminadas.

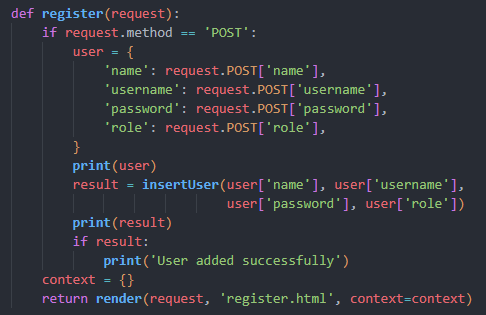
Nas imagens seguintes verifica-se quais os formulários utilizados para o login e registo de utilizadores, respetivamente.





### Funções

Para que estes formulários sejam apresentados, criaram-se as seguintes views:





Como é possível verificar na [figura](#_bookmark71) acima, a função baseia-se num método que, dado o nome do utilizador, o *username* por ele escolhido, palavra-passe e a *role* do utilizador, adiciona-se o mesmo à base de dados.

Caso exista algum erro na criação da conta, o utilizador mantém-se na página de registo, caso contrário, é redirecionado para a página principal, já com o *login* efetuado.

Já na função de *login,* faz-se uma query à base de dados e, caso exista um utilizador com aquelas credenciais, é, à semelhança do que acontece com a criação de conta, redirecionado para a página principal. Caso não exista uma conta com as credenciais inseridas, é apresentada uma mensagem ao utilizador.

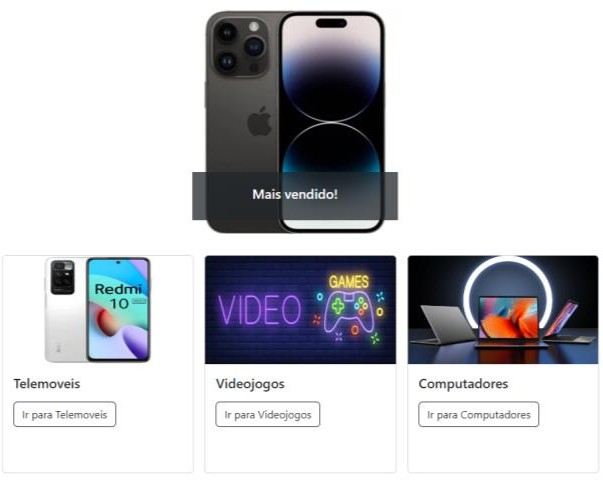
## Navbar

Na implementação do *website* implementou-se uma barra de navegação (*navbar*) de forma a facilitar a navegação dos utilizadores pelo *website*. Cada tipo de utilizador terá acesso uma nav bar diferente que ira ter disponível todas as possibilidades que os mesmos podem necessitar

## HomePage

Como dito no anteriormente, os vários tipos de utilizadores vão ter *Homepages* diferentes. Para isso, fazemos algumas verificações na *view* correspondente à página principal (Index) e, tendo em conta o tipo de utilizador, assim são apresentadas as informações pretendidas.

### Utilizadores não autenticados

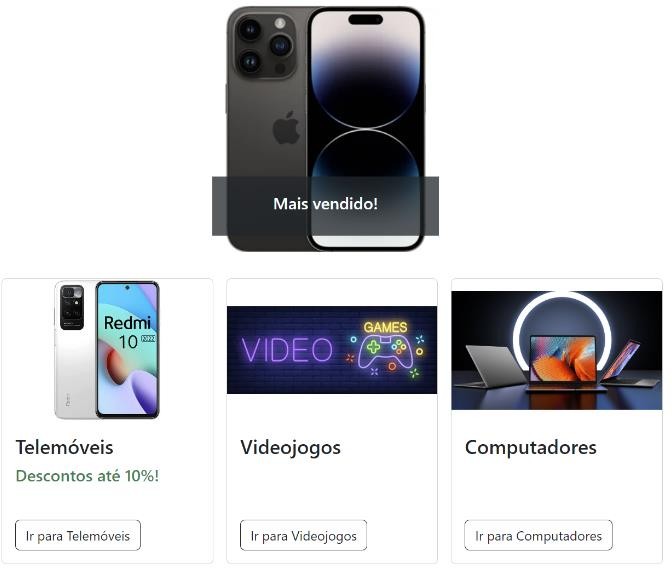




Como é possível visualizar na imagem acima apresentada, após verificação de que o utilizador é cliente, são feitas várias querys à base de dados de modo a obter os produtos mais vendidos, quer seja de sempre, quer seja na própria semana e qual a melhor promoção atualmente disponível. A par disso, é ainda feita uma query para se apresentarem todos os tipos de produtos disponíveis no *website*.

### Clientes

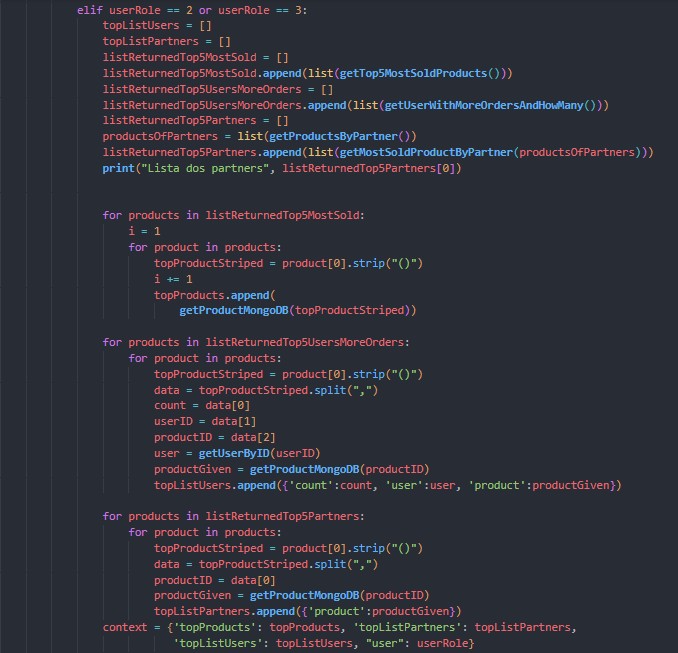
Como se pode verificar, para os clientes da loja, aparecem visíveis as promoções.



### Comerciais

**A screenshot of a computer

Description automatically generated**

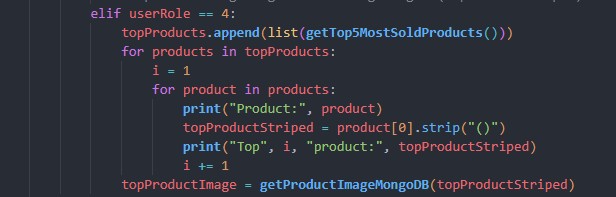


Já nesta parte da função, fazemos referência aos comerciais. Nesta parte, pretende-se apresentar uma espécie de *dashboard* com algumas informações do *website*, assim fazem-se queries à base de dados para se obter os utilizadores com mais encomendas, os produtos mais vendidos por parceiro e os produtos mais vendidos.

### Parceiros

**A screenshot of a phone

Description automatically generated**



Quanto aos parceiros, na sua página principal, são apresentados quais dos seus produtos mais se vendem no *website*.

### Administradores

**A close-up of a screen

Description automatically generated**

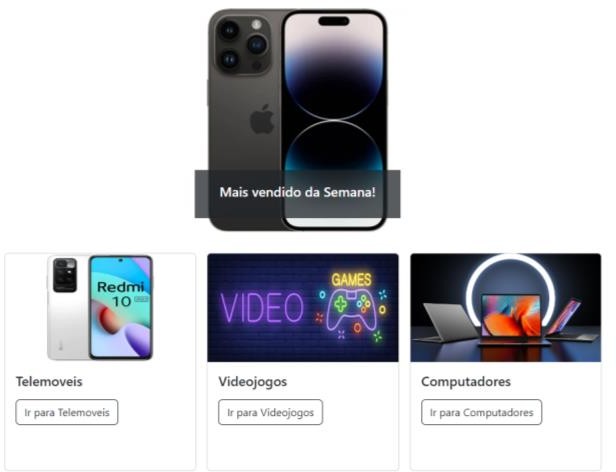


Neste caso, os administradores terão acesso aos utilizadores que utilizam o seu *website*. Para o efeito, são feitas queries à base de dados, usando as funções especificadas previamente, de forma a obter estas informações.

## Listagem de produtos

Como já se referiu, para efetuar a compra de um produto, o utilizador deve estar registado e autenticado no *website*, porém, qualquer que seja o utilizador, consegue visualizar os produtos disponíveis no site (como demonstra a figura seguinte).

Esta será a página referente à apresentação dos produtos ou, *HomePage* tanto dos clientes como dos utilizadores que não estejam autenticados. É de notar que, nesta página, são apresentados os tipos de produtos existentes na loja. Para aceder aos produtos em si, deve clicar no cartão com o tipo de produto.



Acedendo ao tipo de produto pretendido, a página de apresentação será semelhante, porém, apenas estarão disponíveis produtos relacionados entre si e com o tipo previamente selecionado.

A screenshot of a phone

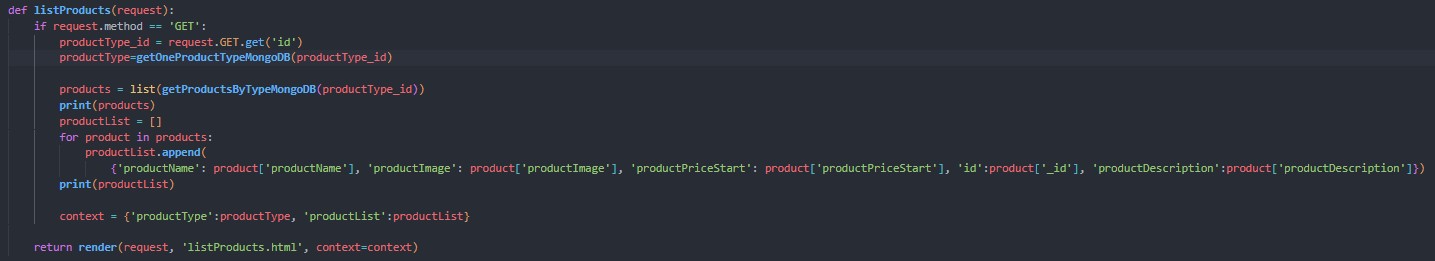
Description automatically generated

Por fim, para aceder a um produto e ver as suas características com mais pormenor, basta clicar na imagem do mesmo, o que o irá redirecionar para a página do produto que é apresentado individualmente:

A screenshot of a product

Description automatically generated

Para listar os produtos, agrupados por tipo de produto, fez-se uso da *view* listProducts, onde se faz uma query à base de dados de forma a receber quais os tipos de produtos existentes para venda, apresentando-os na página. É de notar que se usa o id do tipo de produto, passado no url da página.



Para terminar a visualização de produtos, temos a apresentação individual do produto, juntamente com as suas características mais próprias, cuja função é a seguinte:

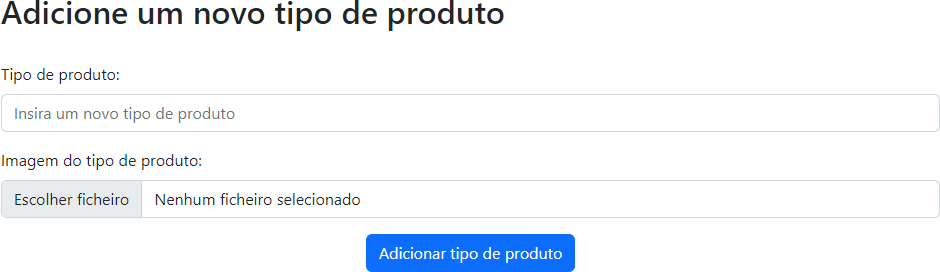


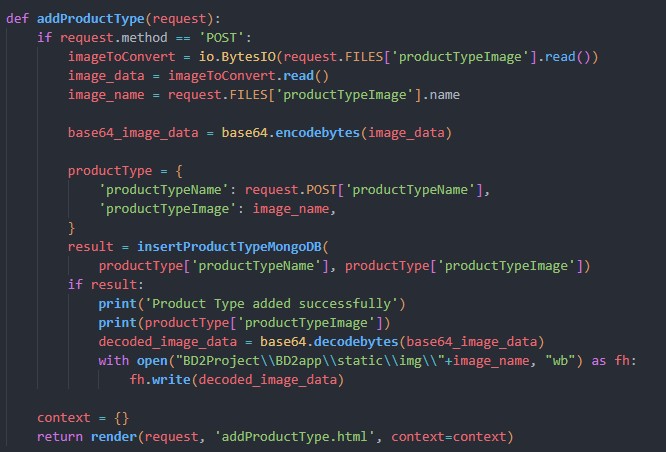
## Criação de produtos

Criaram-se funções para que os utilizadores com as devidas permissões o consigam fazer. Assim, as funções criadas são relativas à criação de tipos de produtos. A primeira, resume- se à inserção de um nome e imagem para o tipo de produtos que se deseja adicionar.

### Adicionar tipo de produto

Para adicionar um novo tipo de produto, o utilizador vê a seguinte página:



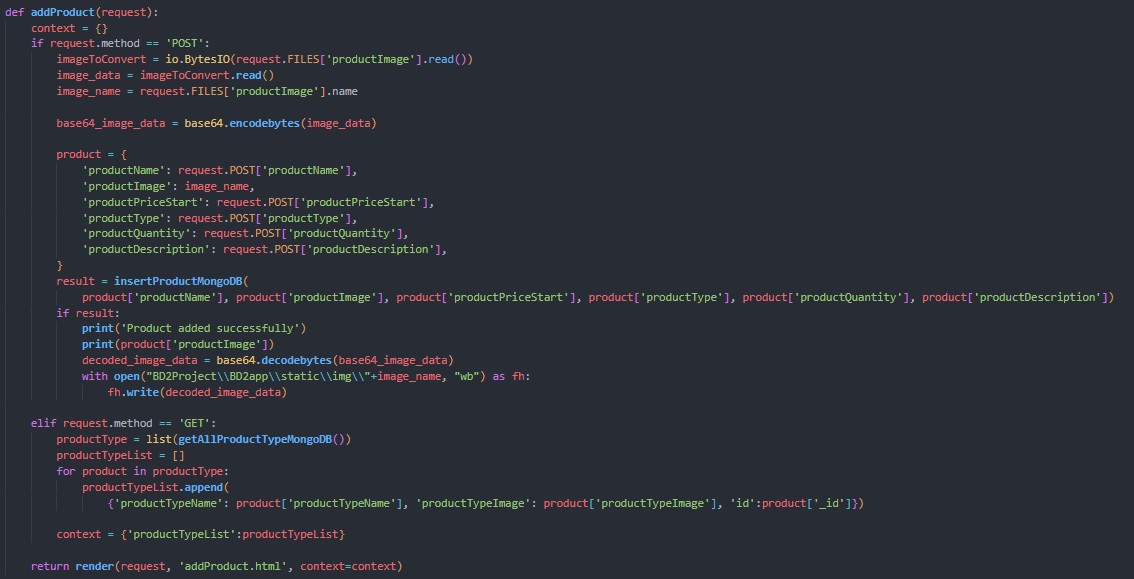
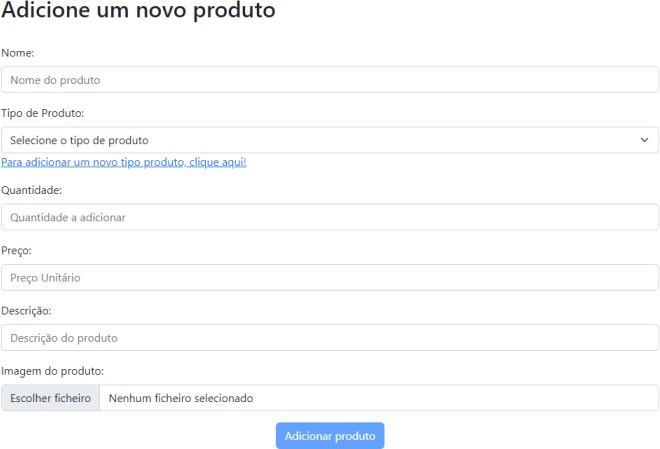


Como é possível visualizar na figura acima apresentada, a função consiste num método *POST* em que, dados o nome e a imagem alusivos ao tipo de produto, adiciona-se o mesmo à base de dados.

### Adicionar um novo produto

Já para adicionar novos produtos à loja, deve indicar-se qual o nome, o tipo a que se associa, a quantidade disponível, o preço unitário, uma pequena descrição e, por fim, uma imagem do mesmo.





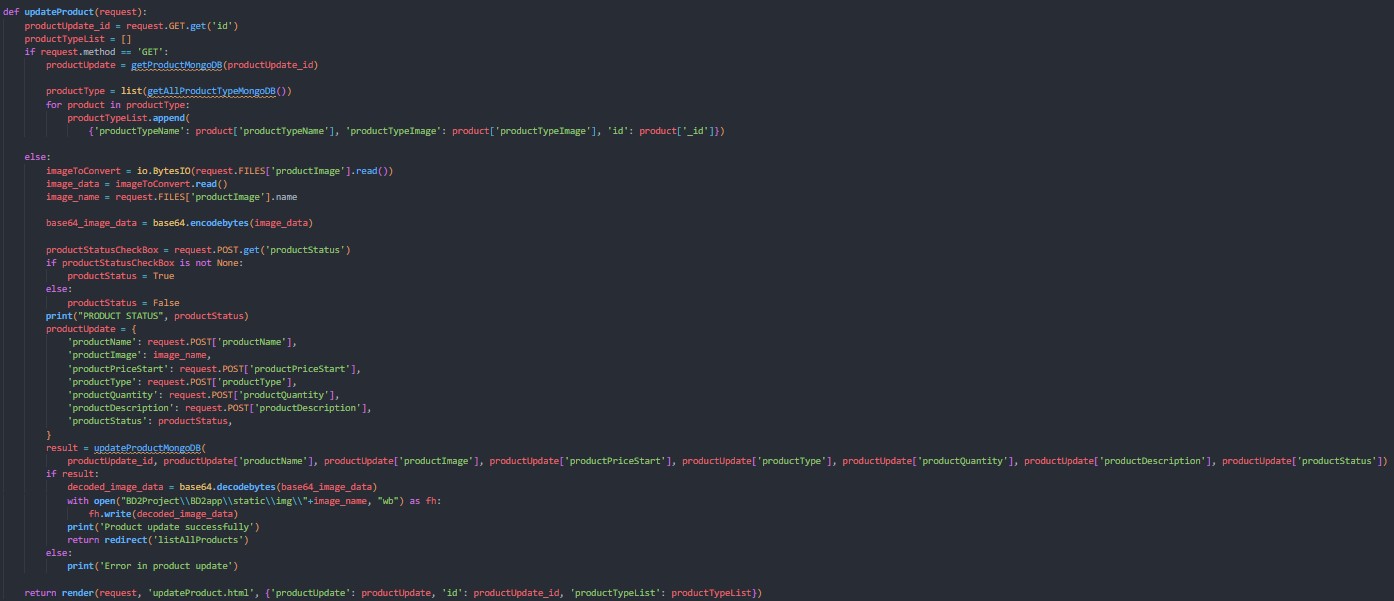
À semelhança do que acontece na criação de novos tipos de produto, é feito uma nova adição à tabela dos produtos com os devidos campos preenchidos relativamente às informações do produto.

## Alteração de produtos

Para alterar produtos à loja, na página de exibição, já aparecem alguns campos preenchidos, como o caso do nome, preço, quantidade e descrição, pelo que os utilizadores sevem sempre especificar o que querem alterar. Para isso, basta apagar o conteúdo do campo e inserir o que desejam.

A screenshot of a computer

Description automatically generated



## Encomendas

De forma a aceder às encomendas, o utilizador deve encontrar-se autenticado no *website.* Tendo isso em conta, ao aceder à parte das encomendas, o utilizador encontra a seguinte página:

A screenshot of a calendar

Description automatically generated

Sendo que a *view* relativa a essa página é apresentada na imagem a seguir. Esta consiste numa simples query à base de dados, recebendo assim uma lista com todas as encomendas do cliente em questão. São apresentados o produto comprado, o preço do mesmo e a data de compra.



O utilizador tem ainda possibilidade de visualizar informações mais detalhadas das suas encomendas, acedendo a “detalhes”. Após isso, a página de apresentação será deste género:

A screenshot of a computer

Description automatically generated

## Carrinho

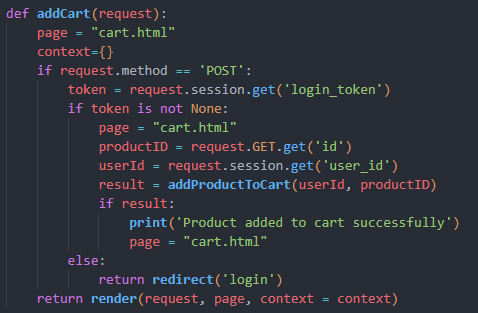
E finalmente implementamos um carrinho no *website*.

A screenshot of a computer

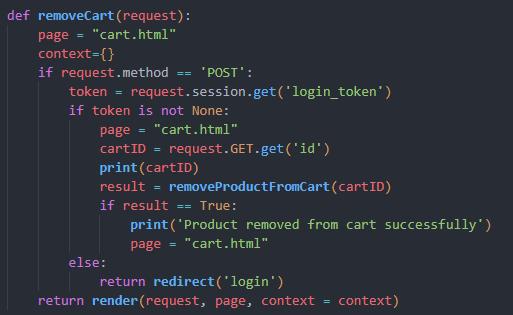
Description automatically generated

### Funções do carrinho

A função de adicionar um produto ao carrinho faz uso do método *POST*, adicionando à tabela correspondente ao carrinho as informações do produto e do utilizador que pretende comprar o produto.



A função de remover produtos do carrinho, tem por base passar o id do carrinho e, após isso, faz uso de uma função previamente definida e, apaga os itens existentes no carrinho.



Por fim, a função de listar o conteúdo do carrinho baseia-se num método *GET* onde se vão buscar à base de dados, todos os produtos existentes no carrinho correspondente ao cliente em questão. O id do cliente é passado no *url* da página.

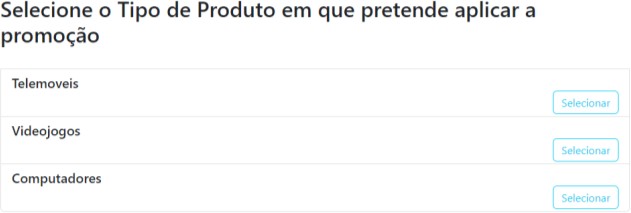


## Promoções

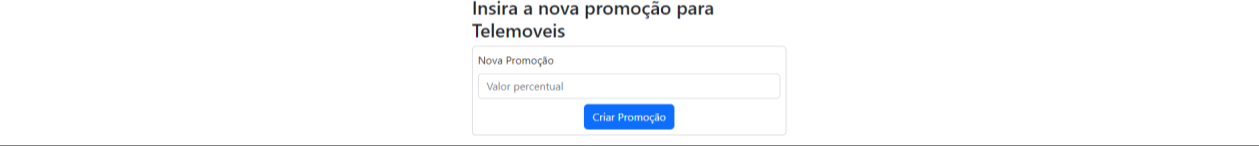
Para evitar problemas, as paginas seguintes só podem ser acedidas por alguns utilizadores . Assim, tendo por base a página apresentada a seguir, os utilizadores podem controlar quais os produtos a que querem adicionar uma nova promoção.



Para adicionar uma nova promoção, deve clicar-se no respetivo botão, obtendo a seguinte página:



Por fim, a página final para adição de uma nova promoção é a seguinte



As funções seguintes foram criadas de forna a auxiliar o bom funcionamento das páginas anteriormente referenciadas*.*



Na imagem anteriormente apresentada, consegue visualizar-se as várias funções relativas às promoções. Para a criação de uma promoção, faz-se um *insert* na tabela **Sales** da promoção que se pretende realizar. Para se obter as promoções, os tipos de produto consoante a promoção, a promoção por tipos de produto e a maior promoção, faz-se uma query à tabela **Sales** consoante a necessidade especificada. Por fim, para se eliminarem registos na tabela, faz-se uso do *delete.*

# Conclusão

Com a realização deste trabalho conseguimos expandir os nossos conhecimentos vistos que muito do que usamos na criação deste site envolveu muita pesquisa externa

Conseguimos de forma bem-sucedida completar muitas das implementações, embora algumas tenha se provado mais desafiantes que outras conseguimos concluir o trabalho da maneira que imaginamos.

É também de mencionar que foi a primeira vez que trabalhamos com algumas das ferramentas como Python e Django, aumentando assim os nossos conhecimentos e ao mesmo tempo criando um desafio.

Em suma, pode dizer-se que foi um projeto bastante desafiador e que superou algumas das expectativas do grupo, tanto em matéria de base de dados como em produção de código em outras linguagens.