

CTeSP Cibersegurança, Redes

e Sistemas Informáticos

Lousada – 2º Ano

Unidade Curricular

Programação II

Trabalho Prático

Alunos:

Rui Alexandre Borba Vitorino - 8190479

Luis Miguel da Silva de Sá – 8190448

Nelson de Jesus Jorge Laureano - 8190460

Docente:

Ana Filipe Barros Duarte

Índice

[Índice 2](#_Toc61946880)

[Índice de Figuras 3](#_Toc61946881)

[Introdução 4](#_Toc61946882)

[I. Objetivos do trabalho 5](#_Toc61946883)

[II. Breve explicação das tecnologias 6](#_Toc61946884)

[III. Configuração Inicial 8](#_Toc61946885)

[IV. Estrutura do Django 11](#_Toc61946886)

[i. Estrutura geral 11](#_Toc61946887)

[ii. Ficheiros importantes 12](#_Toc61946888)

[V. Ficheiro views.py 14](#_Toc61946889)

[VI. Classe Product 15](#_Toc61946890)

[iii. Estrutura da tabela Product 15](#_Toc61946891)

[iv. Código do ficheiro product.py 17](#_Toc61946892)

[VII. Classe Supplier 22](#_Toc61946893)

[v. Estrutura da tabela Supplier 22](#_Toc61946894)

[VIII. Conclusões 23](#_Toc61946895)

[IX. Referências 24](#_Toc61946896)

Índice de Figuras

[Figura 1 Criação de uma diretoria com o comando md 8](#_Toc61950500)

[Figura 2 Criação de um ambiente virtual 8](#_Toc61950501)

[Figura 3 Inserção da diretoria no IDE Visual Studio Code 9](#_Toc61950502)

[Figura 4 Seleção do ambiente virtual 9](#_Toc61950503)

[Figura 5 Criação de terminal no ambiente virtual 10](#_Toc61950504)

[Figura 6 Atualização do pip 10](#_Toc61950505)

[Figura 7 Instalação do Django 10](#_Toc61950506)

[Figura 8 URLs presentes no ficheiro mysite/urls.py 12](#_Toc61950507)

[Figura 9 Ficheiro index.html em uso (com alterações) 12](#_Toc61950508)

[Figura 10 Alguns conteúdos da base de dados 13](#_Toc61950509)

[Figura 11 URLs relativos aos ficheiros HTML 13](#_Toc61950510)

[Figura 12 Código da listagem de um produto e aviso de stock a 0 14](#_Toc61950511)

[Figura 13 Colunas presentes na tabela Product 15](#_Toc61950512)

[Figura 14 Dados presentes na tabela Product 16](#_Toc61950513)

[Figura 15 Métodos de obtenção de informação (nome produto e unidades em stock) 17](#_Toc61950514)

[Figura 16 Read de um produto 17](#_Toc61950515)

[Figura 17 Métodos principais da classe Product 18](#_Toc61950516)

[Figura 18 Criação e remoção de um produto 19](#_Toc61950517)

[Figura 19 Estrutura da tabela Supplier 22](#_Toc61950518)

[Figura 20 Métodos de listagem do código supplier.py 23](#_Toc61950519)

[Figura 21 Opção required no create.html 24](#_Toc61950520)

[Figura 22 Colocação do simbolo do euro e nome 24](#_Toc61950521)

[Figura 23 Nome do produto ao eliminar 25](#_Toc61950522)

[Figura 24 Alerta de stock a 0 25](#_Toc61950523)

[Figura 25 Criação de um produto 26](#_Toc61950524)

[Figura 26 Produto criado com sucesso (Cookies) 26](#_Toc61950525)

[Figura 27 Alterações com sucesso a um produto 27](#_Toc61950526)

[Figura 28 Verificação de estatísticas de dois produtos 28](#_Toc61950527)

[Figura 29 Listagem de informações com sucesso 28](#_Toc61950528)

[Figura 30 Remoção de um produto 29](#_Toc61950529)

[Figura 31 Remoção do product Chai com sucesso 29](#_Toc61950530)

Introdução

Presente neste relatório, encontra-se a resolução do trabalho prático proposto pela professora Ana Filipe Barros Duarte da unidade curricular de Ferramentas de Segurança Informática, da turma de 2º ano do CTeSP de Cibersegurança, Redes e Sistemas Informáticos.

O trabalho consiste na criação de várias classes e respetivos métodos, nomeadamente de um produto e um fornecedor, de modo a realizar certas operações em um website gerido pelo framework Django, atingindo-se assim o objetivo de gestão de produtos e o fornecedor associado.

Durante o relatório encontrar-se-á figuras ilustrativas com as operações de gestão a realizar, bem como uma explicação do código que fora necessário criar.

Na conclusão, encontra-se uma análise final do trabalho prático e eventual críticas e dificuldades encontradas durante a realização do mesmo.

# Objetivos do trabalho

Como indicado no enunciado do trabalho prático, usando o método CRUD (Create, Read, Update, Delete), ou seja, realizar qualquer etapa de gestão de dados relacionados com o negócio fictício usado como exemplo.

Existindo vários dados na base de dados a utilizar (de nome, Northwind), sendo os principais os dados relativos ao produto, fornecedor, e eventualmente, as entregas.

Resumindo, é possível gerar os seguintes tópicos com objetivos:

* Criação das classes necessárias (produto e fornecedor)
  + E criação dos respetivos métodos CRUD em cada classe
* Fazer uso do ficheiro views.py
* Uso correto do paradigma POO

Referenciando o ficheiro views.py, este que irá realizar uma ligação entre o website e os ficheiros necessários para qualquer operação, seja esta relativamente a um pedido HTTP para atualização de um produto, ou para qualquer outra operação disponível.

# Breve explicação das tecnologias

Django – Framework Web Python, que permite rapidamente lançar um website e respetiva implementação de aplicações no mesmo, de forma simples e segura, usando diversos modelos e princípios.

Framework – conjunto de códigos de baixo nível, escritos de forma a serem escaláveis, seguros e robustos, permitindo ao desenvolver final apenas se focar na criação de códigos de alto nível, como a criação e integração da sua aplicação no respetivo framework já existente (pe. website).

Python – linguagem de programação de alto nível, sem necessidade de compilação de código, suportando múltiplos paradigmas de programação como orientação a objetivos (POO), programação funcional e estruturada, e, devido à sua fácil leitura e escritura de código, é uma das mais utilizadas globalmente, em diversas áreas, desde aplicações, *scripts*, módulos de sistemas operativos.

SQL – em português, linguagem de consulta estruturada, utilizada na pesquisa e gestão de tabelas e dados presentes em uma base de dados relacional, e semelhante ao Python, devido à sua geral facilidade de utilização é utilizada globalmente, tendo a particularidade de apresentar o resultado e não o caminho para o mesmo.

Visual Studio Code – IDE grátis (em português, ambiente de desenvolvimento integrado) criado pela Microsoft, multiplataforma, que permite a edição de códigos em diversas linguagens de programação, suportando extensões e tornando o desenvolvimento de código completo em qualquer ambiente/linguagem.

Ambiente Virtual (Python) – de modo a facilitar o suporte a especificas bibliotecas, módulos, programas, requeridos pelas aplicações, estas que não se encontram na instalação padrão do Python, o uso de um ambiente virtual é a solução, este que é uma diretoria contendo uma instalação específica do Python e respetivos pacotes necessários (pe. Django e pip).

Chave Primária – uma chave primária é uma chave única que identifica em uma tabela um respetivo dado, semelhante e normalmente de nome, ID, ou seja, identificador.

Chave Estrangeira - é um identificador, isto é, uma chave primária de outra tabela que está agora na tabela atual, referenciando e interligando dados.

Classe –

Método -

# Configuração Inicial

Explicando e exemplificando a configuração inicial de um ambiente virtual, onde se poderia realizar um projeto utilizando o framework Django e respetivo website.

Começando em criar uma diretoria, onde se situará todos as diretorias e ficheiros necessários, de forma tradicional (botão direito do rato no ambiente de trabalho, criar pasta) ou com o seguinte comando no terminal de escolha, md “nome\_diretoria”.

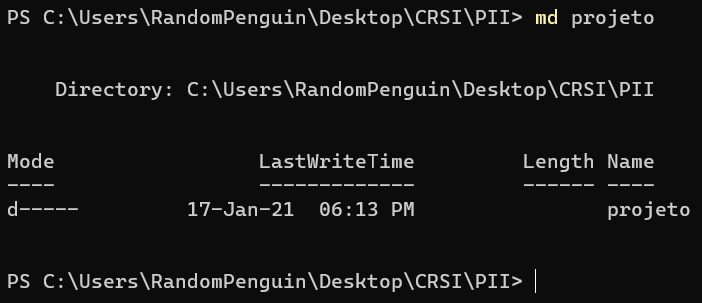


Figura 1 Criação de uma diretoria com o comando md

Procedendo, ir-se-á criar um ambiente virtual, aconselhado durante a unidade curricular, mas também como boa prática durante o desenvolvimento de um projeto, novamente e agora exclusivamente através da linha de comandos, com o executável do Python e o argumento -m venv “nome\_diretoria”.

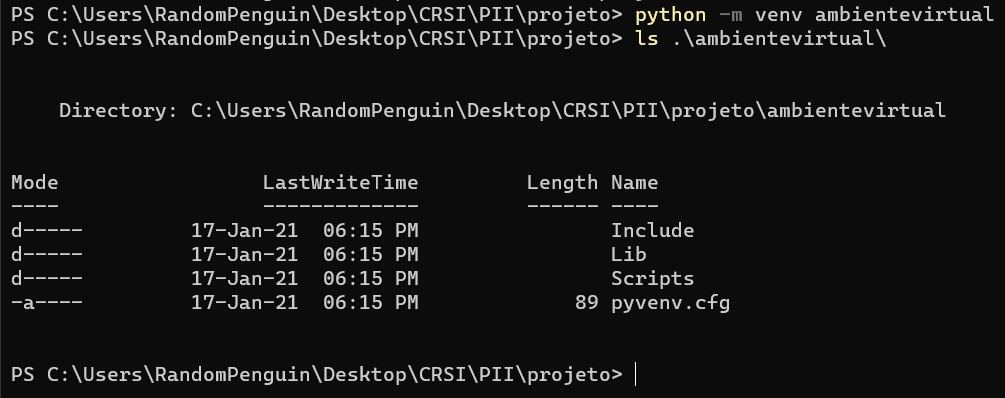


Figura 2 Criação de um ambiente virtual

Estando as configurações básicas criadas, é necessário, agora no editor de código, neste caso no Visual Studio Code, abrir a diretoria recentemente criada, usando o atalho de teclado (padrão) Ctrl + K, Ctrl + O ou navegando até à aba File > Open Folder….

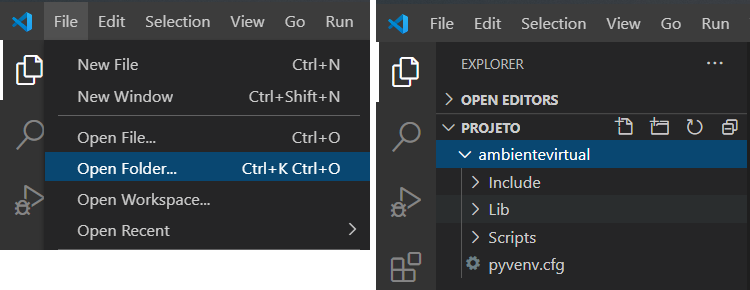


Figura 3 Inserção da diretoria no IDE Visual Studio Code

Como será possível observar na figura, o ambiente virtual encontra-se na diretoria, como de esperar, podendo-se assim proceder à escolha desse ambiente virtual e à instalação de pacotes necessários.

Na aba View, Command Pallete (ou atalho de teclado Ctrl + Shift + P, padrão) seleciona-se a opção Python Interpreter, e ir-se-á selecionar a opção que conterá o caminho da nossa diretoria e entre parenteses o nome do ambiente virtual.

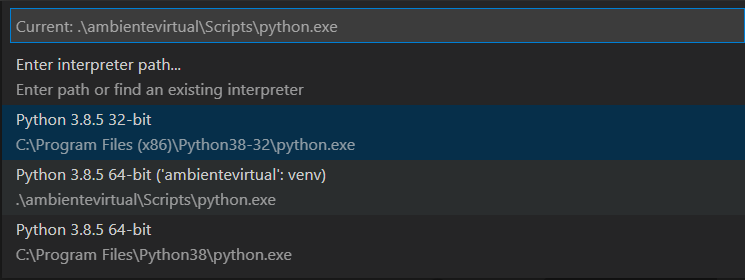


Figura 4 Seleção do ambiente virtual

Estando agora o ambiente virtual selecionado, é possível aceder a um terminal específico navegando até à aba Terminal, New Terminal (ou atalho de teclado, Ctrl + Shift + ç), e veremos que a verde, estará o nome do nosso ambiente virtual.

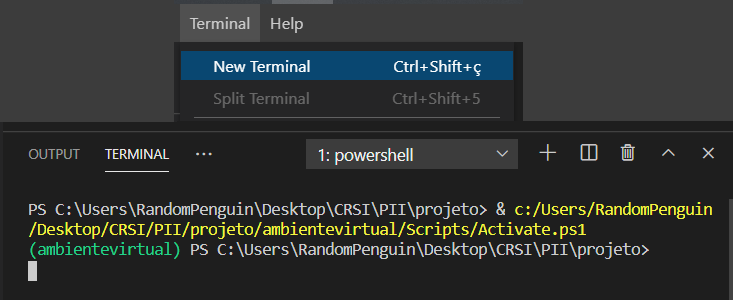


Figura 5 Criação de terminal no ambiente virtual

Aqui poder-se-á então instalar os pacotes, quase finalizando a configuração, mas como boa prática, a atualização do instalador (pip) é recomendada, usando o comando pip install --upgrade pip.

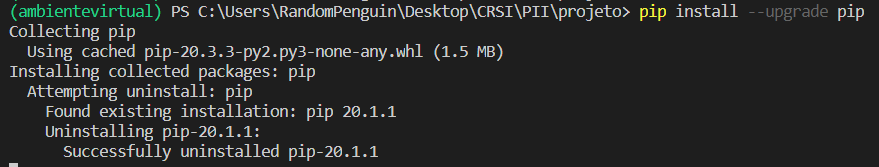


Figura 6 Atualização do pip

E procedendo à última etapa, a instalação do Django, com o comando pip install django.

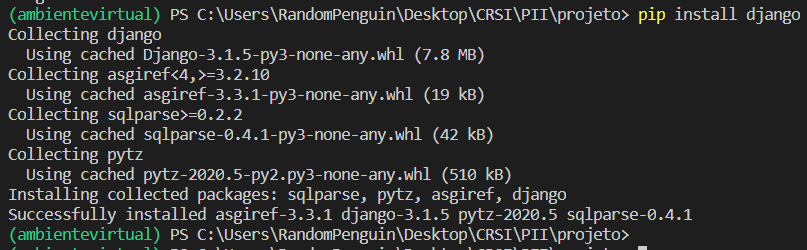


Figura 7 Instalação do Django

Após tudo isto poderemos começar a criar a nossa aplicação como irá ser demonstrado nos próximos tópicos.

# Estrutura do Django

## Estrutura geral

O Django, seja uma instalação de raiz, ou a instalação a utilizar fornecida pela docente da unidade curricular, apresenta uma estrutura, e é importante conhecê-la para simplesmente entender o funcionamento geral da aplicação, podendo assim criar código eficiente, e tal como um dos objetivos do Django indica, este fora feito para evitar a repetição de código e reutilizar métodos já criados.

Diretoria Função Ficheiros **importantes**

mysite Códigos principais urls.py

mysite/templates *Templates* HTML create.html

delete.html

index.html

read.html

statistics.html

update.html

db Base de dados Northwind.db

polls Códigos associados product.py

supplier.py

urls.py

views.py

env Ambiente Virtual *apenas a instalação do Django é importante*

## Ficheiros importantes

Começando com o ficheiro urls.py da diretoria principal do projeto, neste caso mysite, este que irá conter, como o nome indica, os URLs do website, já existindo no código as opções necessárias.

Foi apenas referido este ficheiro pois é uma parte importante de todos os projetos web com o Django.

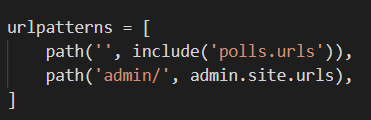


Figura 8 URLs presentes no ficheiro mysite/urls.py

Explicando, existe o URL principal, denominado pelas aspas ‘ ’, que irá conter as URLs presentes no ficheiro urls.py da diretoria polls (que irá conter os códigos a realizar), e o URL de administração do Django.

Na diretoria mysite/templates/*ficheiros*.html encontram-se os ficheiros de base de cada website escritos em HTML, nomeadamente das páginas a encontrar durante a criação, remoção, atualização, visualização de estatísticas sobre os produtos e também o website principal, de nome, index.html.

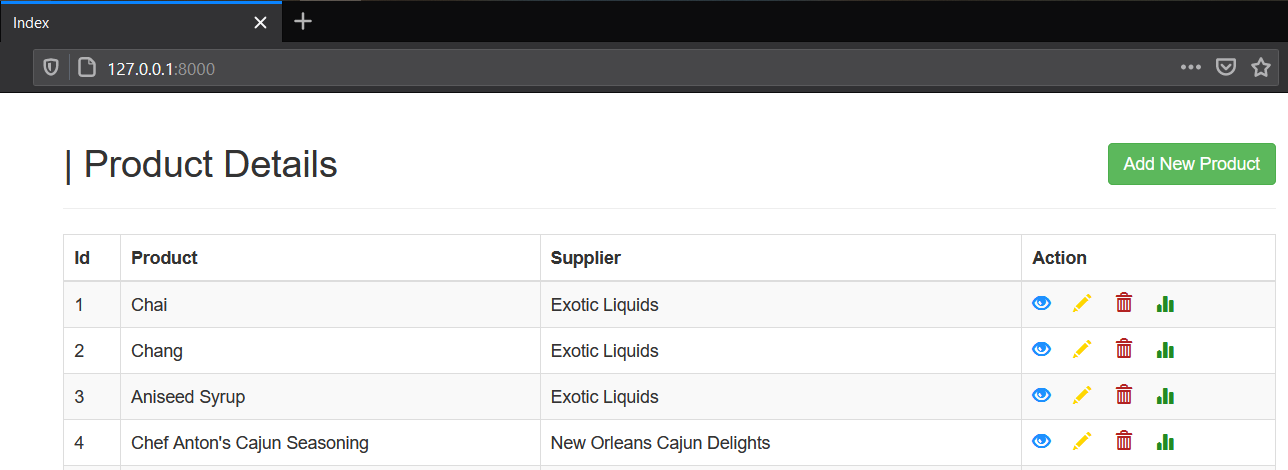


Figura 9 Ficheiro index.html em uso (com alterações)

Na diretoria de nome, db, encontra-se o ficheiro da base de dados a utilizar, esta que têm como nome, Northwind.db, e é bastante utilizada na criação de projetos de aprendizagem.

Com a ajuda de um website, referenciado mais em detalhe posteriormente no relatório, é possível visualizar alguns dos conteúdos que a base de dados contém.

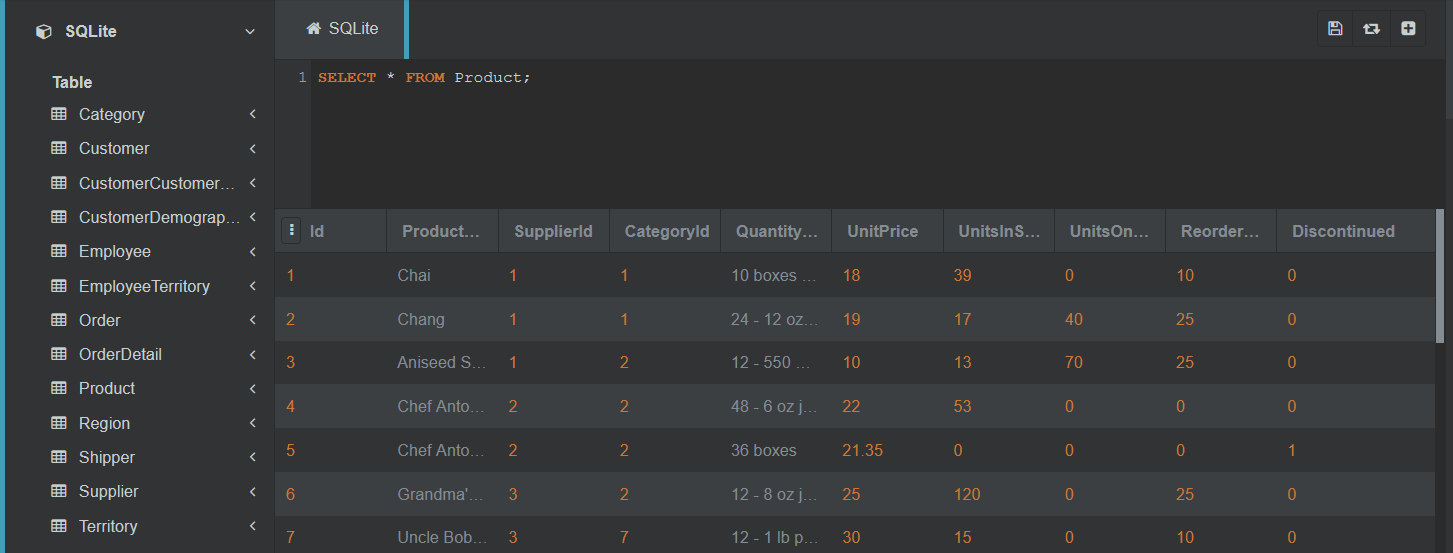


Figura 10 Alguns conteúdos da base de dados

Terminando, e chegando à parte mais importante, relativamente ao trabalho a desenvolver, na diretoria de nome polls, será onde os nossos ficheiros de códigos a criar serão colocados (product.py e supplier.py), contendo também o ficheiro views.py, este que irá interligar todos os códigos, seja HTML ou Python, de forma a disponibilizar as funcionalidades a desenvolver de forma simples e centralizada.

Nesta diretoria existe outro ficheiro de URLs, que irá fornecer as URLs mais “locais” para o website, indicando a localização dos ficheiros HTML e o respetivo URL de acesso, (pe 127.0.0.1:8000/read/ > read.html).



Figura 11 URLs relativos aos ficheiros HTML

# Ficheiro views.py

Embora as operações a realizar sejam efetuadas usando outros códigos, como product.py e supplier.py, é com o views.py que os dados são “transferidos” pelo website e colocados na sua correta forma para serem devidamente utilizados em qualquer método posterior.

É o caso do método index, que irá listar todos os produtos (e o nome do fornecedor associado) na página principal no website (index.html), usando o método list\_products() no ficheiro product.py.

Mencionando também o próximo método, read, este que irá listar várias informações sobre o produto, que semelhante ao método index, irá buscar essas informações aos respetivos métodos no ficheiro product.py.

Tendo sido feito uma alteração neste ficheiro e obviamente no ficheiro read.html, fora dos objetivos principais do trabalho, esta que aquando deteta que o stock do produto é zero (0), em vez de ser mostrado por exemplo “0 bananas” irá avisar “Stock is at 0”.

*\*O mesmo processo é realizado no método statistics e ficheiro statistics.html.*



Figura 12 Código da listagem de um produto e aviso de stock a 0

Relativamente a outras operações como criação, atualização, remoção e listagem de estatísticas, os comentários presentes no código foram respondidos, sem grandes alterações efetuadas.

# Classe Product

Passando então para a criação da primeira classe necessária e pretendida pelo trabalho, a classe de nome Product (em português, Produto), irá conter todos os dados relacionados com, obviamente o produto, tendo também, como chave estrangeira, o ID do fornecedor.

## Estrutura da tabela Product

É importante conhecer a estrutura da tabela product na base de dados, de modo a entender que dados irão ser geridos, e como esta se interliga com outros dados/tabelas.

Usando o website SQL OnLine IDE[[1]](https://sqliteonline.com/), é possível observar alguns dados presentes bem como a estrutura da tabela:

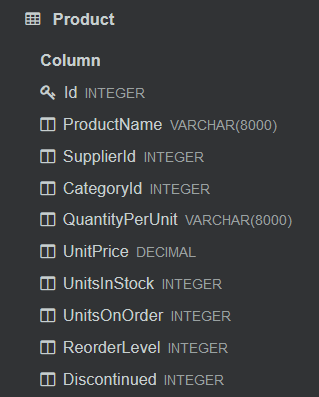


Figura 13 Colunas presentes na tabela Product

Verifica-se então na figura, que existem diversas colunas, desde o nome do produto, como uma chave estrangeira do fornecedor (que irá obviamente ser de bastante utilidade), colunas relacionadas com a quantidade e preço, outras com informações sobre possíveis entregas, e até mesmo uma chave estrangeira da categoria do produto.

Usando um comando SQL de forma a se obter todas as informações de todas as colunas (SELECT \*) da tabela Product (FROM Product), veremos alguns dados já presentes na base de dados e na respetiva tabela.

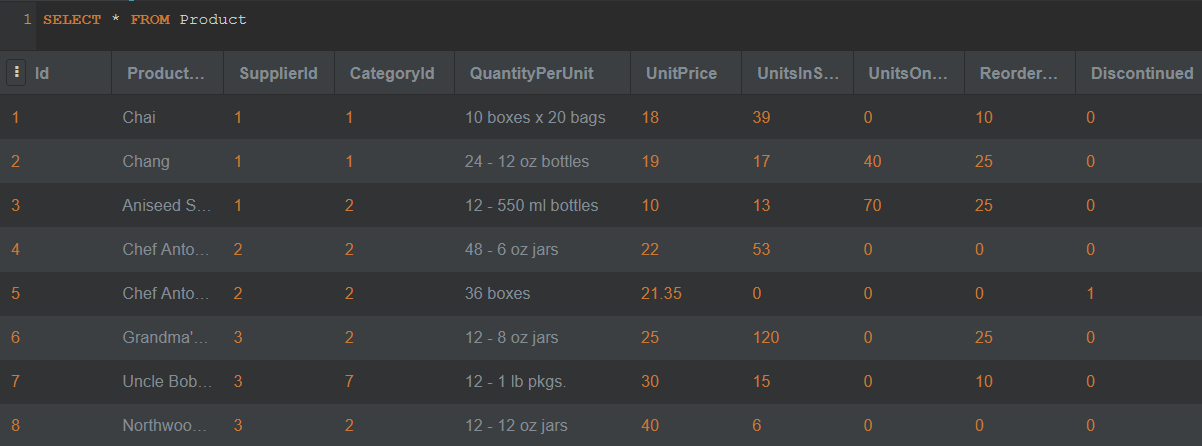


Figura 14 Dados presentes na tabela Product

Existindo agora uma noção básica da estrutura da tabela Product e eventuais ligações com outras tabelas (Supplier e Category), poder-se-á proceder à criação da classe em um novo ficheiro, que terá como nome product.py.

## Código do ficheiro product.py

Existindo vários métodos dentro da única classe de nome Product, estes que realizam as operações CRUD ou obtêm outras informações em particular, como o nome do produto, ou as unidades em stock.

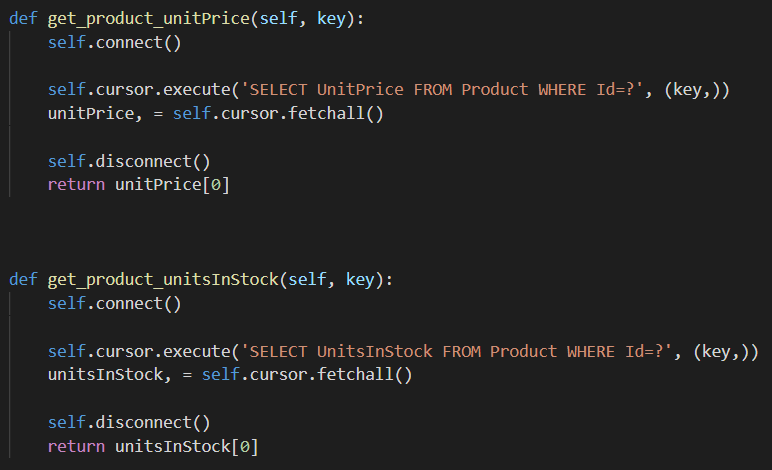
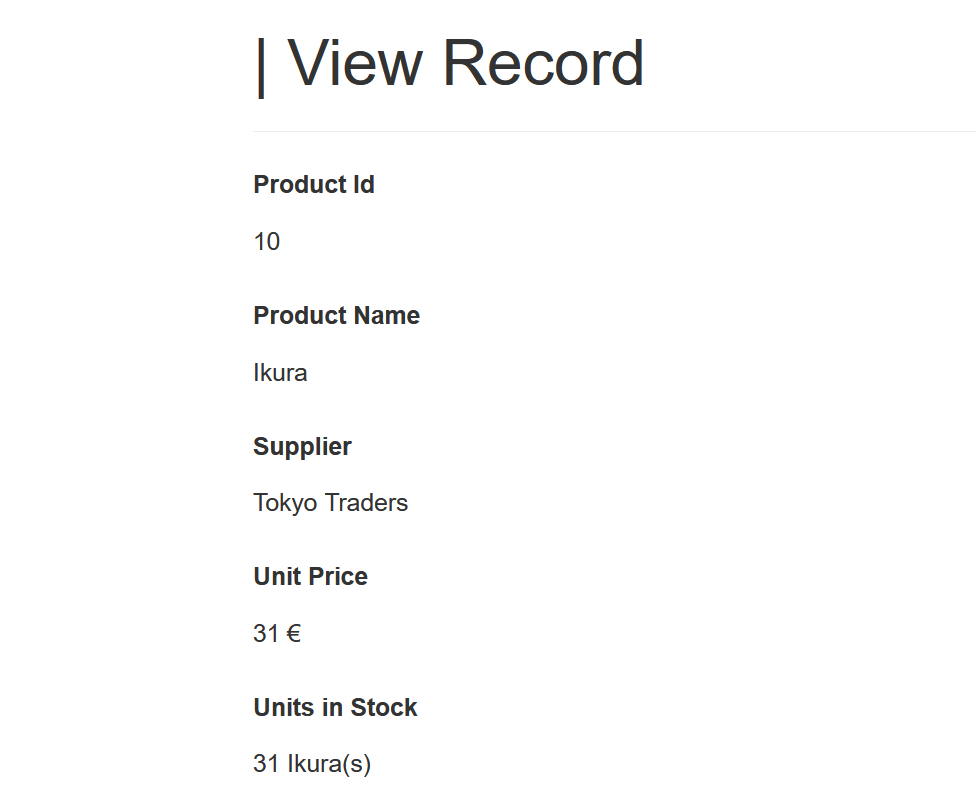


Figura 15 Métodos de obtenção de informação (nome produto e unidades em stock)

Aquando se necessita de obter informação, realiza-se um SELECT a certa coluna da respetiva tabela, onde o ID será o enviado pelos *forms* do Django, basicamente usando o ID do produto selecionado no website.



*Método get\_product\_unitPrice*

*Método get\_productName*

Figura 16 Read de um produto

Existindo chamadas de vários métodos durante qualquer operação, nomeadamente, os métodos de ligação e desconexão à base de dados, estes que estão presentes no início da classe, bem como a localização da base de dados no nosso sistema; e não esquecendo de importar o módulo sqlite3 (presente na primeira linha), de forma a se obter as funcionalidades de gestão da base de dados.

Outro método importante é o de listagem de produtos, este que irá realizar um INNER JOIN, isto é, uma junção de certa coluna de outra tabela, neste caso, SupplierID (da tabela Supplier) para a tabela Product, podendo assim listar o fornecedor, incluindo o seu nome, de certo produto.

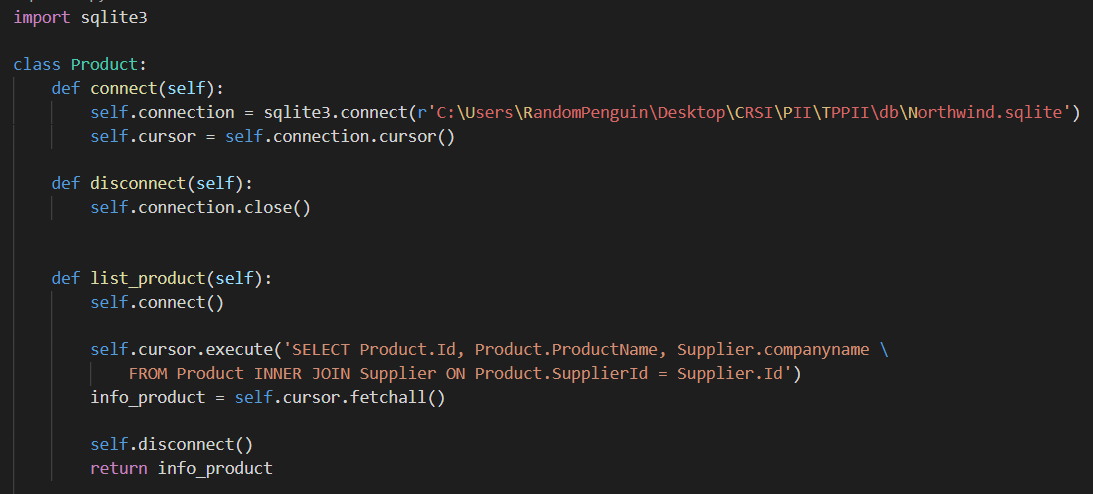


Figura 17 Métodos principais da classe Product

De modo a responder aos objetivos principais do trabalho, como a criação de um produto, criou-se os respetivos métodos, semelhante aos acima referidos, alterando apenas as queries SQL e os dados a serem passados do website para o método.

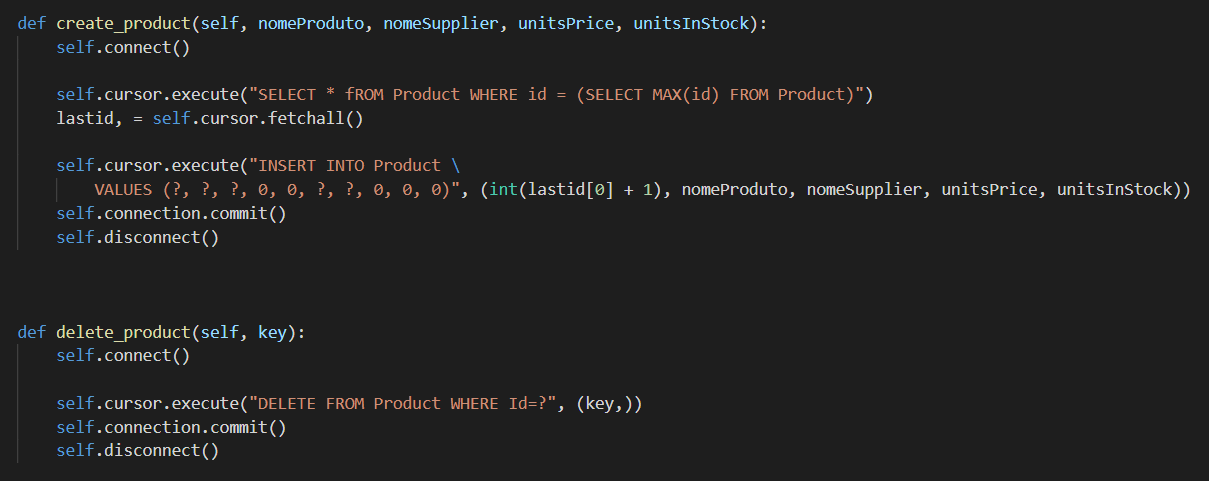
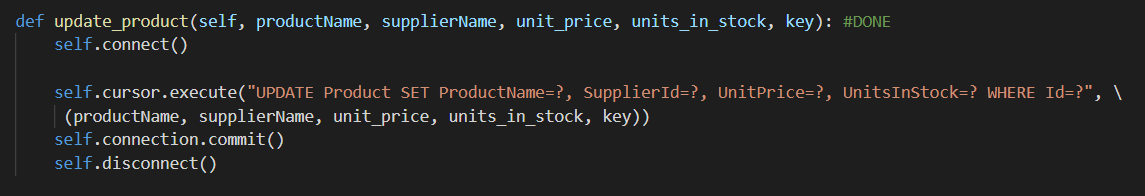


Figura 18 Criação e remoção de um produto

A query SQL usada para criar um produto, ou **qualquer outra operação que realize alterações** ao esquema ou aos dados presentes na base de dados requer um commit(), de forma a confirmar o que fora feito, de certa forma é um guardar alterações feitas a um ficheiro Word, por exemplo.

Um aspeto a referir durante a criação do produto, nomeadamente ao ID do novo produto, este que irá obter o último ID presente na coluna ProductID usando um query e acrescentando um valor; isto pois, devido a alguma motivo desconhecido não se conseguiu utilizar a função cursor.lastrowid.

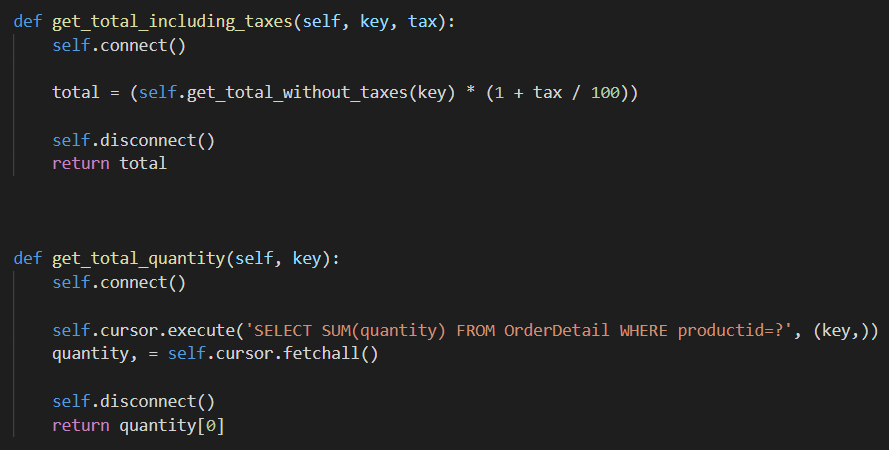
Já estando os métodos criados relativamente às operações de listagem, criação e remoção, faltará a operação de atualização de certo produto.



Este, que usando a query SQL UPDATE, irá atualizar os valores presentes no website, tais como nome, ID do fornecedor, preço por unidade e unidades em stock.

Tal como a operação de criação de um produto, existe a interligação de tabelas, obtendo através do ID do fornecedor o seu nome, sendo de fácil compreensão para o utilizador final. Esta ligação e respetiva query é efetuada no ficheiro views.py, este que já fora explicado, estando o código referente presente na figura.

Relativamente à última secção do ficheiro product.py, que irá criar certas estatísticas sobre o stock existentes e respetivos valores, consoante uma taxa em percentagem, e quantidades em várias entregas, novamente usando queries SQL com o SELECT.



No método de nome get\_total\_including\_taxes(), ir-se-á obter o valor retornado pelo método get\_total\_without\_taxes(), (não mostrado na figura) onde simplesmente se irá realizar uma multiplicação dos valores preço por unidade em stock, e então multiplicando pela taxa, obtendo-se assim o valor total com a respetiva taxa de 23%.

No método de nome get\_total\_quantity(), a query SQL já realiza a operação necessária, neste caso que seria somar (SUM) a quantidade de produtos relativos a várias entregas do produto em questão (usando o WHERE=Id), assim, evitou-se múltiplos queries à base de dados e eventual cálculo no código.

# Classe Supplier

Sendo esta classe e ficheiro de fácil criação, mesmo apresentando uma estrutura de maior dimensão comparando com a tabela produto, isto devido apenas ser necessário algumas informações em particular.

## Estrutura da tabela Supplier

Usando novamente o website SQLOnline é possível observar a estrutura da tabela e os campos que iremos necessitar, que serão o nome e o ID.

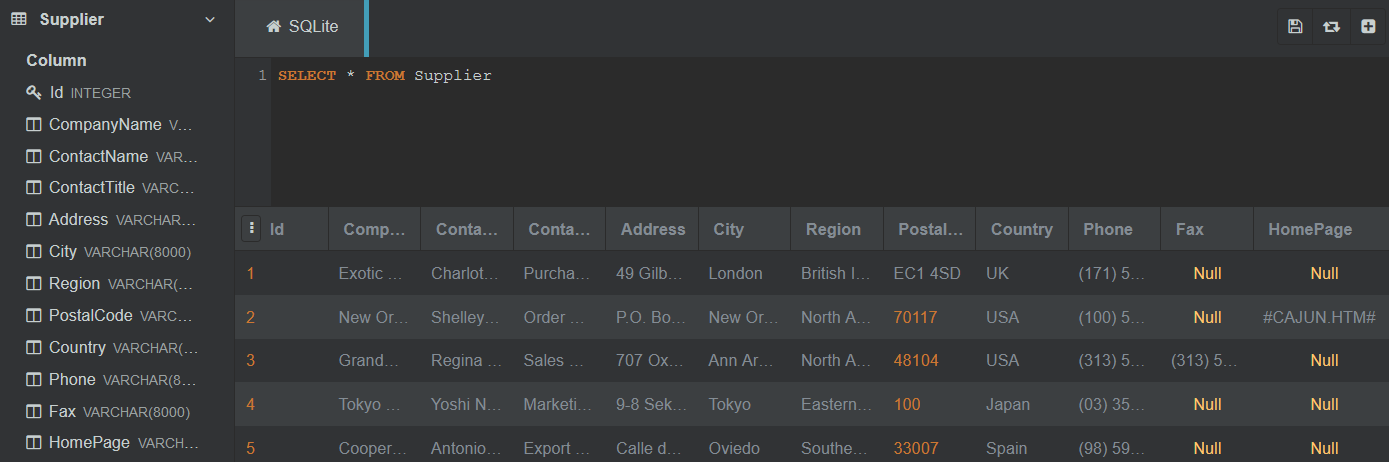


Figura 19 Estrutura da tabela Supplier

## Código do ficheiro supplier.py

Consequentemente o código do ficheiro e a classe é de simples e rápida criação, tendo apenas os métodos de ligação e desconexão à base de dados, e dois métodos, um de listagem geral dos dados de certo fornecedor e outro de listagem de um nome de certo fornecedor dado respetivo ID.

Sendo o último método o que poderá gerar mais confusão, contendo um LEFT JOIN para interligação do ID do produto com o ID do fornecedor associado, isto pois, em qualquer tópico do website é mostrado o nome do fornecedor em vez do seu ID, visto não ser fiável nem eficiente a utilização de IDs de inúmeros fornecedores ao contrário de o seu nome, principalmente para o utilizador final.

Explicando também o porquê da utilização de um LEFT JOIN e não um INNER JOIN como acontece em métodos anteriores, isto pois, apenas se necessita de dados da tabela “à esquerda” (Product) e de todos os dados da tabela “à direita” (Supplier); enquanto ao se utilizar um INNER JOIN, ir-se-á mostrar valores interligados entre as duas tabelas, como nome e ID associado.

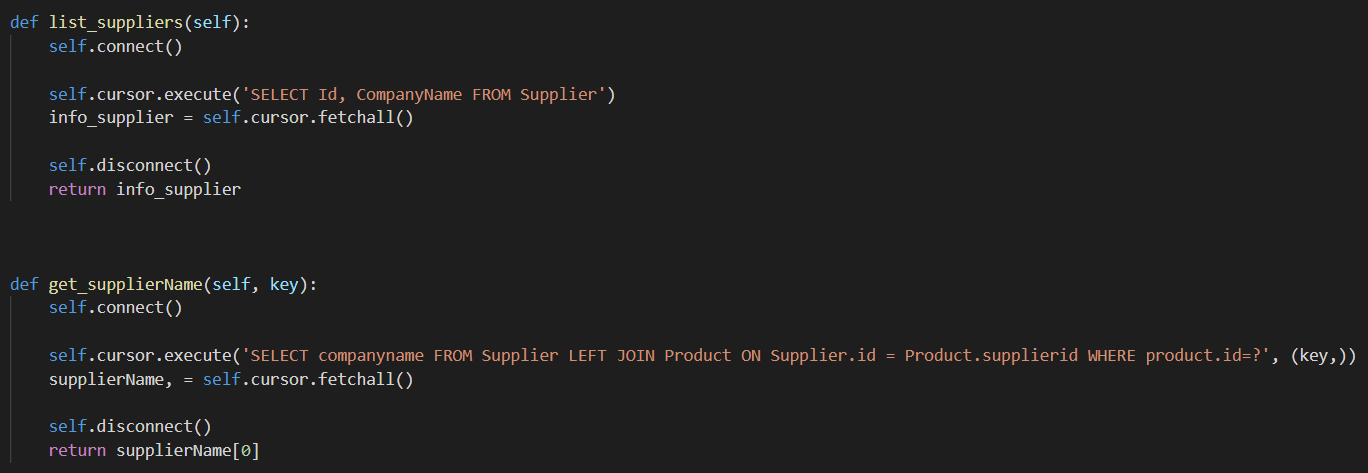


Figura 20 Métodos de listagem do código supplier.py

# Alterações realizadas

Algumas alterações foram realizadas devido a ter existindo tempo e principalmente curiosidade, sendo a mais básica a alteração das cores dos ícones do index.html.

Outra alteração fora a colocação de a opção required nos formulários HTML, desta forma ao se criar ou atualizar o produto é necessário que todos os valores sejam inseridos.

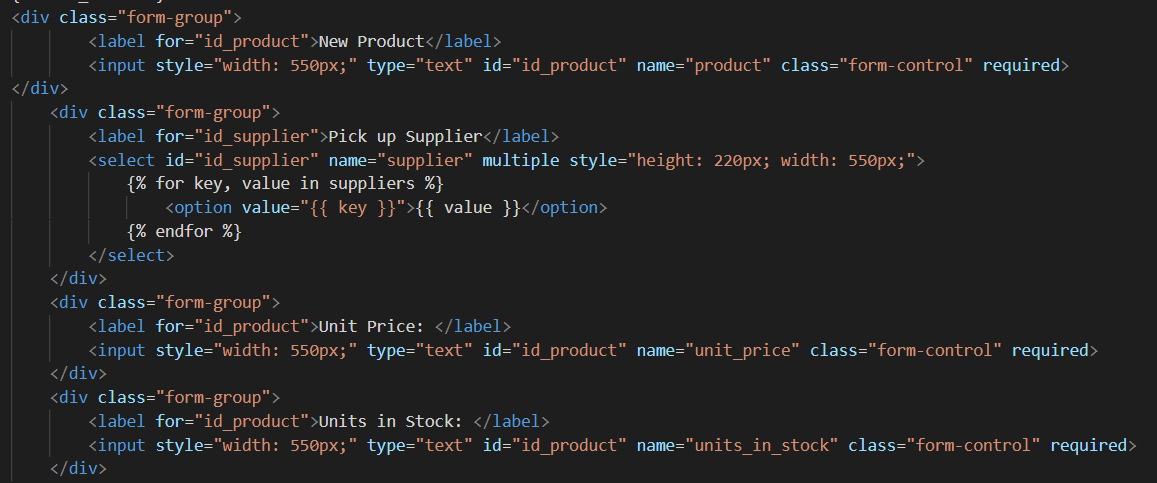


Figura 21 Opção required no create.html

Uma pequena alteração visual, que até pode passar despercebida, foi adição do símbolo euro e o nome do produto durante a visualização do mesmo, tendo sido efetuado a alteração também no ficheiro read.html.

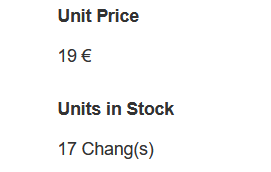


Figura 22 Colocação do simbolo do euro e nome

Como poderia ter sido notado, durante a remoção de um produto, o nome deste irá ser mostrado, evitando uma possível eliminação de um produto errado, confirmando o nome.

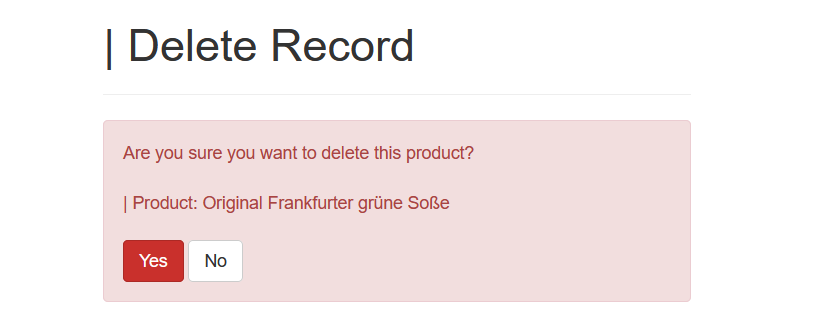


Figura 23 Nome do produto ao eliminar

Por último, adicionou-se nos ficheiros views.py, read.html e statistics.html, código para alterar o utilizador caso o stock de um produto seja zero (0).

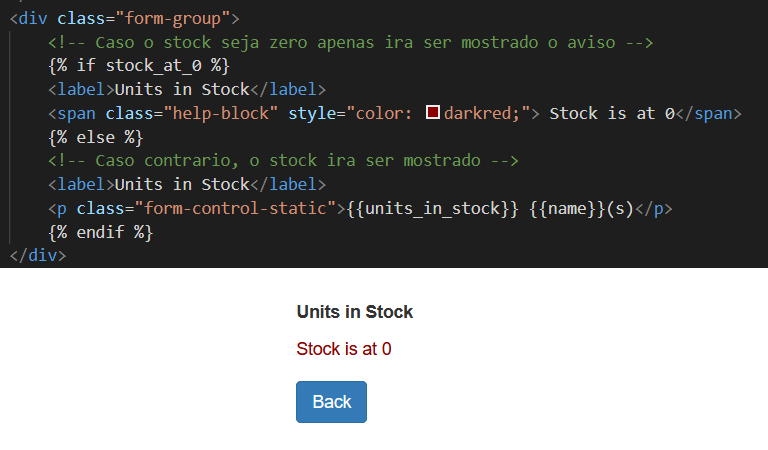


Figura 24 Alerta de stock a 0

# Verificação de resultados

Verificando todas as operações pretendidas para realização, bem como as pequenas alterações.

Começando pela criação de um produto, e visualização do mesmo criado.

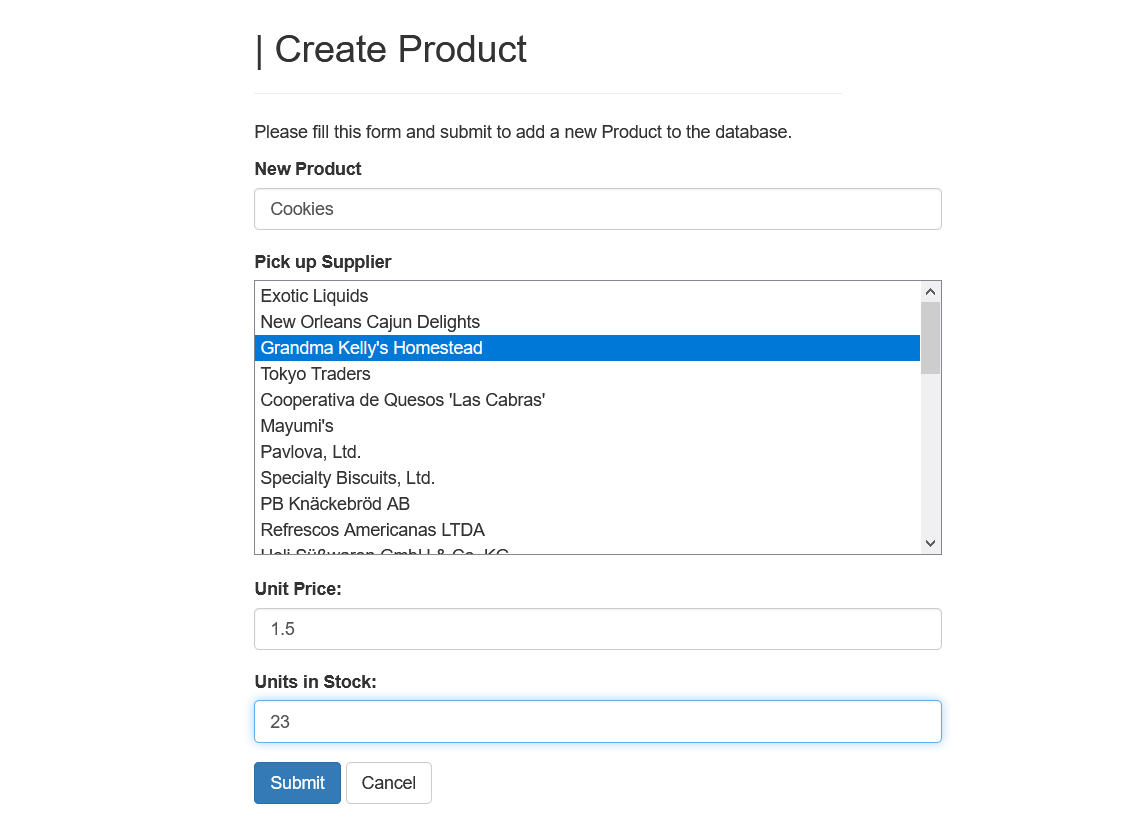


Figura 25 Criação de um produto

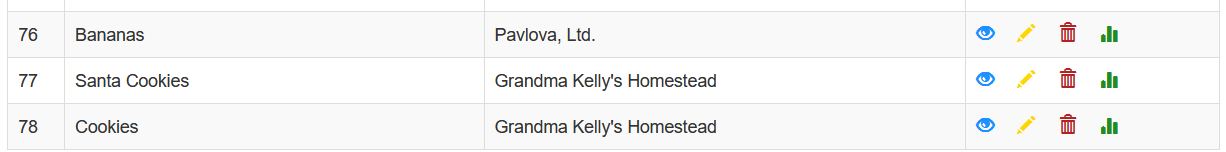


Figura 26 Produto criado com sucesso (Cookies)

Atualizando um produto, usando também o exemplo para a verificação da alteração de requerimento de alteração de campos (o que também acontece durante a criação de um produto, sendo necessário a inserção de todos os campos indicados).

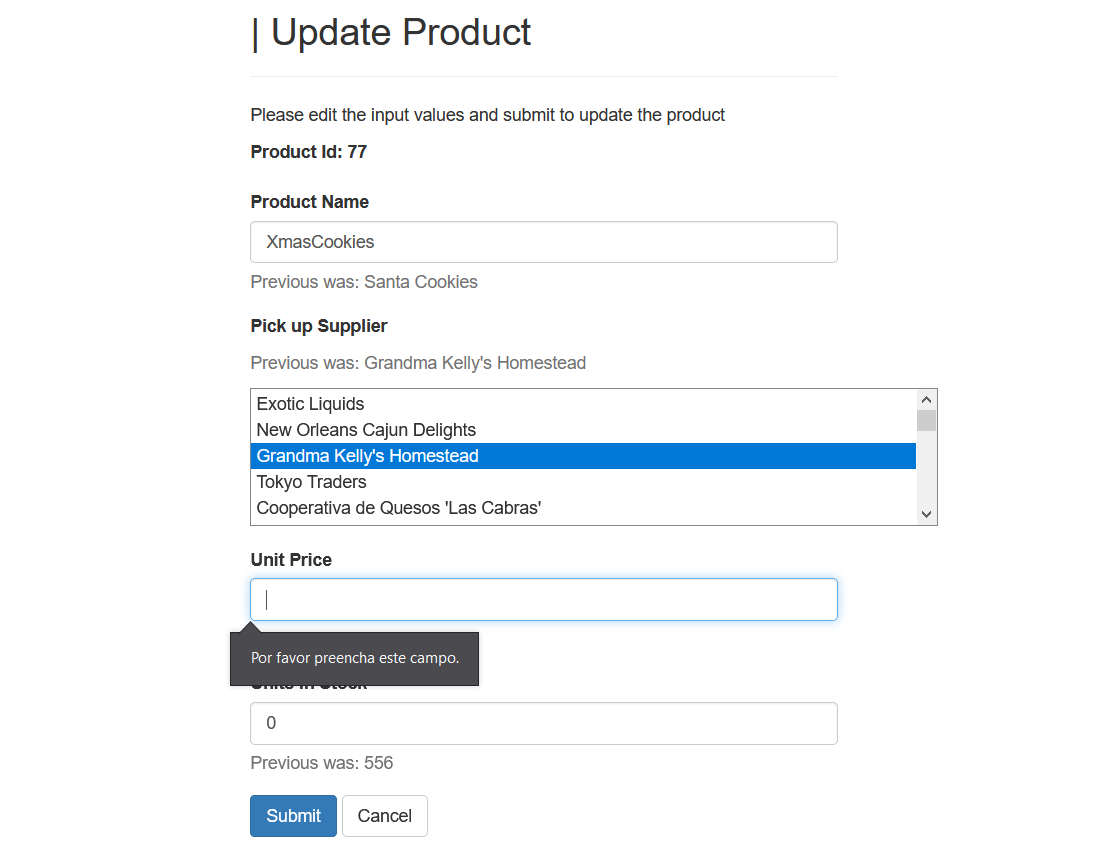


Figura 27 Requerimento de inserção de informações

Durante a alteração de um produto (ou a criação de um mesmo), caso o código detete que valores incorretos estão a ser introduzidos, como por exemplo, texto no preço unitário, após submeter, o seguinte pop-up é apresentado, e a página será atualizada automaticamente (não submetendo as informações).

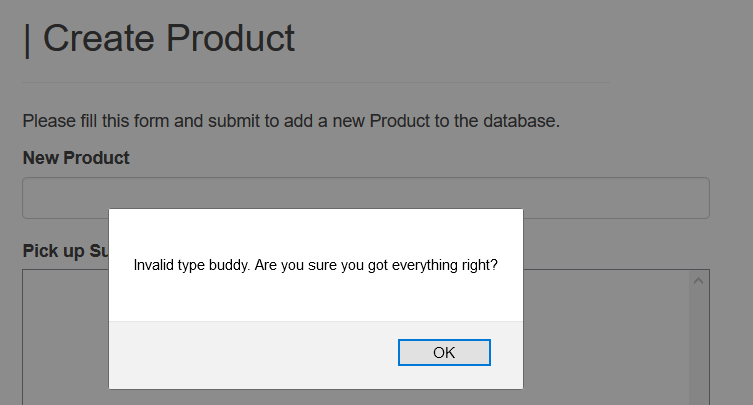


Figura 28 Pop-up de aviso de tipo inválido de informação

Após a alteração, visto que o stock será 0, e o nome fora alterado, as modificações irão ser mostradas nas estatísticas, incluindo o aviso de stock a 0.

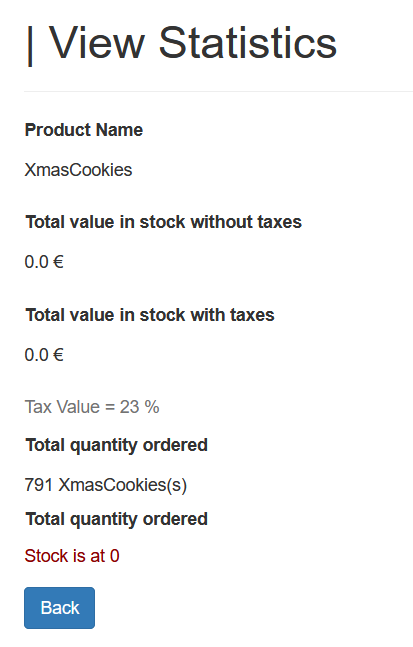


Figura 29 Alterações com sucesso a um produto

Aproveitando para a visualização das estatísticas de produtos, seja no produto recentemente criado ou em qualquer outro, e sem esquecer o primeiro objetivo, a listagem de informações sobre o produto.

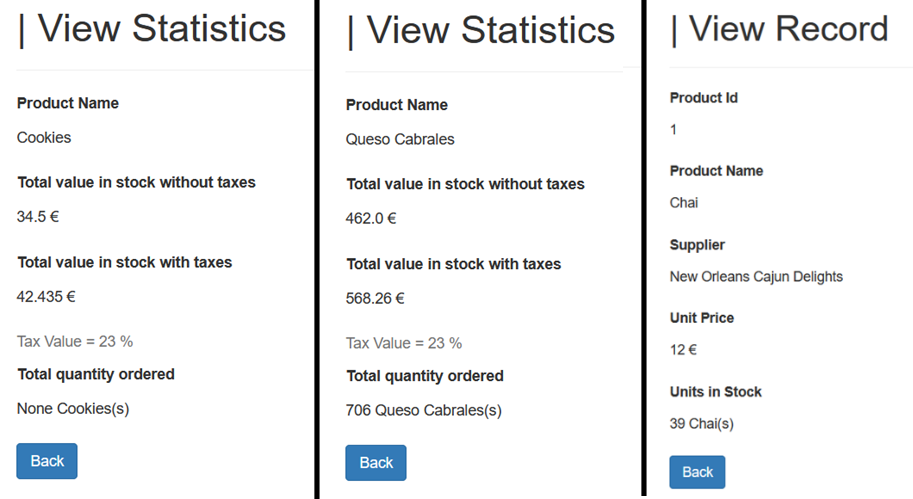


Figura 30 Verificação de estatísticas e informações de produtos

Faltando apenas a confirmação da operação de eliminação de produtos.

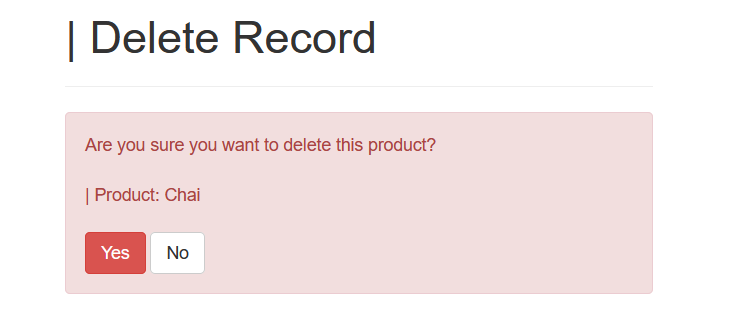


Figura 32 Remoção de um produto

Verifica-se então que o produto de nome Chai, foi eliminado com sucesso.

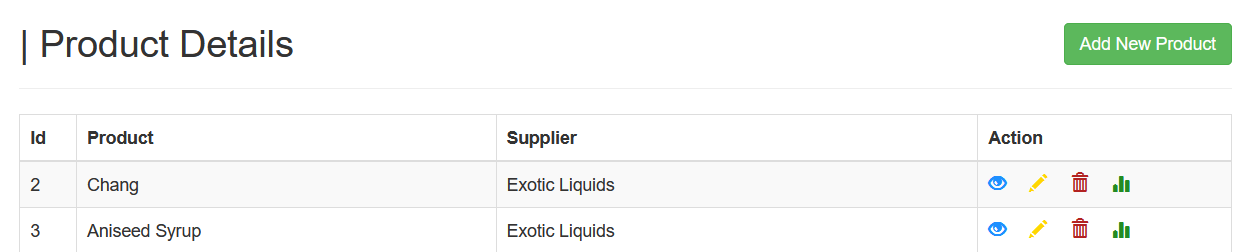


Figura 33 Remoção do product Chai com sucesso

# Conclusões

Este trabalho permitiu assimilar todos os conhecimentos obtidos durante as aulas da unidade curricular, bem como explorar um pouco o framework Django, durante a tentativa de eliminação de vários produtos (que infelizmente não fora concluída com sucesso), e certamente cementar o conhecimento sobre classes e o seu objetivo, eficácia na linguagem de programação Python, e até mesmo juntar tecnologias de base de dados com código Python e HTML.

Tendo todos os elementos do grupo contribuído para a realização do mesmo, concluindo todos os tópicos individualmente e centralizando várias funcionalidades à versão final, desta forma existiu contribuição equalizada e integração de cada elemento.

# Referências

Todo o material utilizado durante a realização do trabalho foi disponibilizado pelo docente e disponível no Moodle

Link One Drive com as máquinas virtuais:

<https://ipppt-my.sharepoint.com/:u:/g/personal/8190479_estg_ipp_pt/EWgo4SAS689JrPFlMN_pFHIB-La9CnZ3ej9EdXLN_knPjA?e=Pcr4QU>