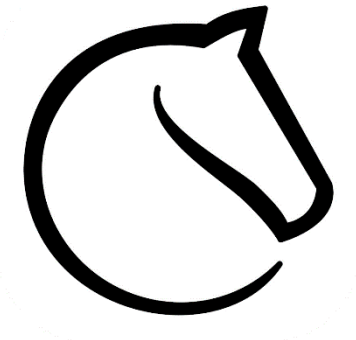
SITE E-COMMERCE: CHESSPOWERED.NET

Técnico de Gestão e Programação de Sistemas Informáticos, 12º51

PROGRAMAÇÃO E SISTEMAS DE INFORMAÇÃO

PROVA DE APTIDÃO PROFISSIONAL



ELABORADO POR:

FRANCISCO PERESTRELO PONTES

Índice

1 Índice de Imagem

Não foi encontrada nenhuma entrada do índice de ilustrações.

2 Introdução

Este projeto será um site genérico criado para negócios que utilizem a internet e websites como a sua principal forma de comércio.

Os softwares utilizados para a criação do projeto foram:

- Para o desenvolvimento do website será usado ***Django***, esta é uma framework web escrita em python que é capaz de processar front-end e back-end, tudo numa única applicação.
- Para fazer *deploy* do site foi usada a plataforma ***Heroku***.
- Para a criação de diagramas foi utilizado a aplicação ***app.diagrams.net***.

Neste caso o tema do trabalho será a venda de cursos de aprendizagem de xadrez, e material para xadrez competitivo como relógios, peças e tabuleiros.

Nenhum destes materiais existem mesmo na loja, serão apenas imagens stock.

Depois disto serão apresentados os objetivos gerais do projeto, mesmo que alguns destes não se encontrem completos.

Após os objetivos gerais, o desenvolvimento de todos os componentes do web-site, base de dados e o deploy do web-site será explicado na secção de Planeamento do relatório.

3 Objetivos Gerais

Estes serão os objetivos gerais do projeto que poderão mudar ao longo do seu desenvolvimento:

- Ter um sistema de autenticação de modo a distinguir as páginas de admins, clientes e visitantes do site.
- Criar uma base de dados que gere a venda de produtos.
- Ter uma Front-end elaborada que funcione em mobile e desktop.
- Ter um sistema de compra de produtos que automaticamente atualiza faturas, vendas e stock de inventário na compra de um item, este sistema tem de conter uma simulação de compra de produtos e um cesto de compras.
- Ter uma página de contactos que permita a comunicação do site com uma pessoa designada, neste caso será o e-mail de tal pessoa.
- Ter uma página que permita jogar xadrez contra uma máquina.
- Permitir a impressão de faturas guardadas na base de dados.
- Fazer o deploy do site na plataforma ***Heroku.***

4 Tabela de atividades

TABELA DE ATIVIDADES

Descrição	Atividade	Duração	Atividades precedentes
Análise de requisitos	A	1 dia	-
Planeamento do DER	B	1 dia	A
Criação de tabelas	C	1 dia	B
Planeamento do layout do site (master page)	D	4 dias	C
Criação de uma página de contactos que manda e-mails automaticamente	E	1 dia	D
Criação de forms para introdução de dados	F	2 dias	D
Criação de um sistema de cesto de compras	G	1 dia	F
Criação de uma simulação de uma página de compras	H	2 dias	G
Front-end	I	3 dias	D
Sistema de autenticação	J	2 dias	I

Gerenciamento de páginas visíveis para utilizadores certificados	K	2 dias	J
Tabuleiro	L	3 dias	K
Engine de Xadrez	M	7 dias	L
Fazer o deploy do website para a plataforma Heroku	N	1 dia	L, K

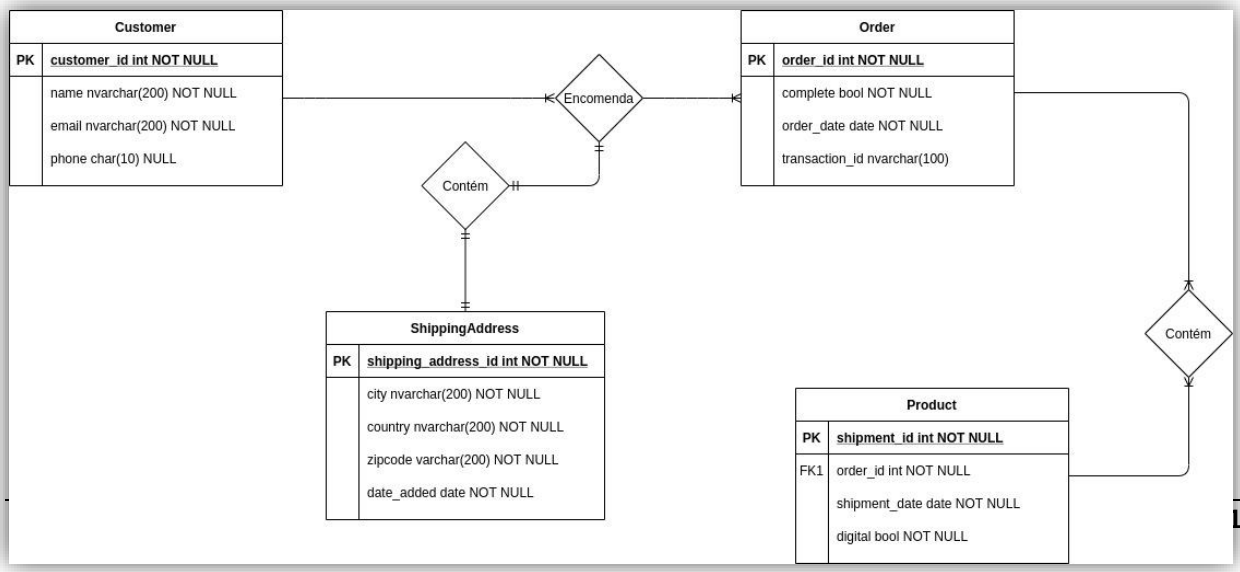
5 Planeamento

5.1 Base de dados

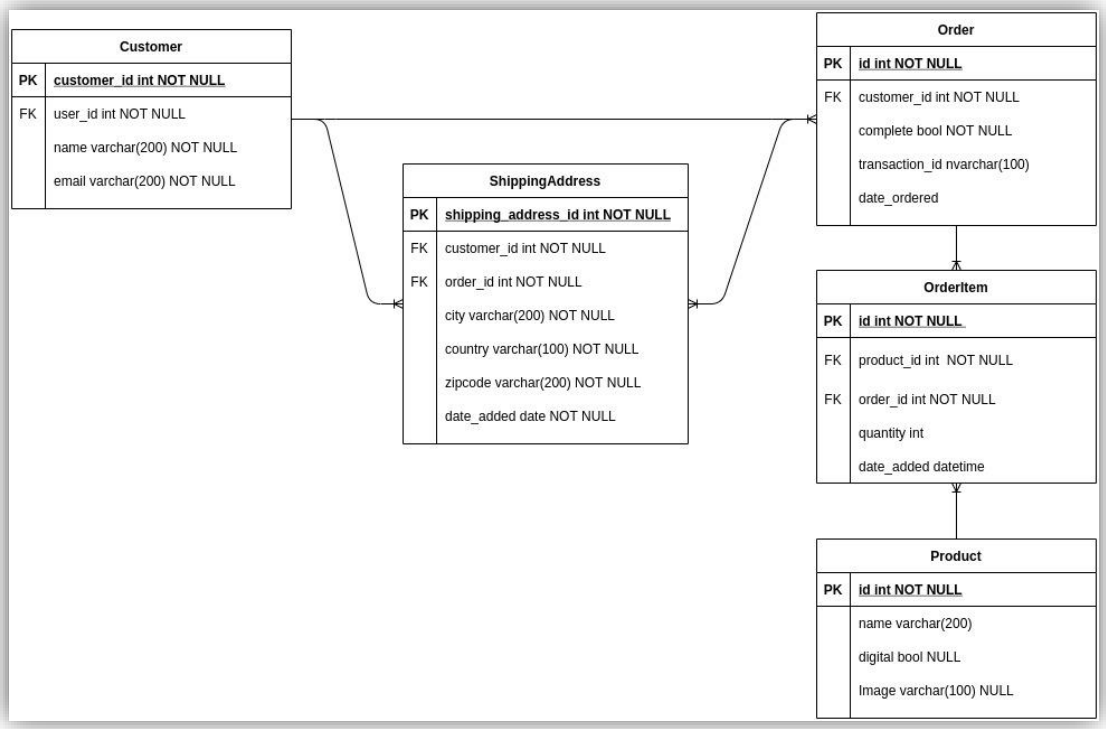
Esta parte do relatório demonstrará o progresso da base de dados e explicará a estrutura da base de dados.

5.1.1 DER

Este é o Diagrama de entidades e relações, a primeira versão do planeamento da base de dados:



5.1.2 Estrutura final da base de dados



5.1.3 Queries de criação das tabelas

Todas as tabelas foram criadas usando Django sem SQL direto na aplicação.

Em Django todas as tabelas são representadas por uma classe de python, isto permite-nos manipular os dados de formas mais flexíveis.

Uma dessas formas é a inclusão de métodos nestas classes, algo que foi usado na aplicação.

Em todas as classes existe um método chamado __str__, este apenas retorna o campo que identifica a classe, pode ser um nome ou o id do objeto.

Há a referir também que existe uma inexistência de chaves primárias nas tabelas, isto é, porque por padrão, Django inclui uma chave primária id em todas tabelas, este campo é um campo de número inteiro simples que se auto-incrementa.

5.1.3.1 Producto

```python
class Product(models.Model):
    name = models.CharField(max_length=200)
    price = models.FloatField()
    digital = models.BooleanField(default=False, null=True, blank=True)
    image = models.ImageField(null=True, blank=True)

    @property
    def imageURL(self):
        try:
            url = self.image.url
        except:
            url=""
        return url

    def __str__(self):
        return self.name
```

Esta classe representa um produto em stock na loja.

Esta classe tem um método específico, imageURL(), este retorna o caminho para o campo image da classe, permitindo-nos renderizar a imagem associada a cada produto.

5.1.3.2 Cliente

```python
class Customer(models.Model):
    user = models.OneToOneField(User, null=True, blank=True, on_delete=models.CASCADE)
    name = models.CharField(max_length=200, null=True)
    email = models.CharField(max_length=200)

    def __str__(self):
        return self.name
```

Esta classe é responsável por guardar informação específica à conta de cada utilizador.

Algo especial desta classe é que tem uma relação de um para um com “User”, isto é uma classe de Django que representa uma conta de utilizador, que podera ou não ser criada pelo mesmo.

Para clarificação Django, por padrão, já tem uma extensão embebida para gerir utilizadores.

5.1.3.3 Encomenda

```python
class Order(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.SET_NULL, null=True, blank=True)
    date_ordered = models.DateTimeField(auto_now_add=True)
    complete = models.BooleanField(default=False)
    transaction_id = models.CharField(max_length=100, null=True)

    @property
    def get_cart_items(self):
        orderitems = self.orderitem_set.all()
        total = sum([item.quantity for item in orderitems])
        return total

    @property
    def get_cart_total(self):
        orderitems = self.orderitem_set.all()
        total = sum([item.get_total for item in orderitems])
        return total

    @property
    def shipping(self):
        shipping = False
        orderitems = self.orderitem_set.all()
        for i in orderitems:
            if j.product.digital == False:
                shipping = True
        return shipping

    def __str__(self):
        return str(self.id)
```

Esta classe representa um carrinho de um cliente.

Existem três funções essenciais à aplicação:

- Get_cart_items:
  - Retorna a lista dos itens na encomenda.
- Get_cart_total:
  - Retorna o custo total da encomenda.
- Shipping:
  - Retorna se a encomenda irá precisar de ser encomendada ou se é constituída de itens digitais (os quais não precisam de ser encomendados).

5.1.3.4 OrderItem

```python
class OrderItem(models.Model):
    "Represents a item that is in the cart of a user"
    product = models.ForeignKey(Product, on_delete=models.SET_NULL, null=True)
    order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)
    quantity = models.IntegerField(default=0, null=True, blank=True)
    date_added = models.DateTimeField(auto_now_add=True)

    @property
    def get_total(self):
        "Gets the total price of a cart item, according to its price and quantity"
        return self.product.price * self.quantity
```

Esta classe representa um item no carrinho do utilizador e faz a relação entre a encomenda e o produto da loja.

Existe uma função essencial à aplicação:

- Get_total:
  - Retorna o preço total de um item da encomenda de acordo com a quantidade e preço do produto.

5.1.3.5 Shipping Address

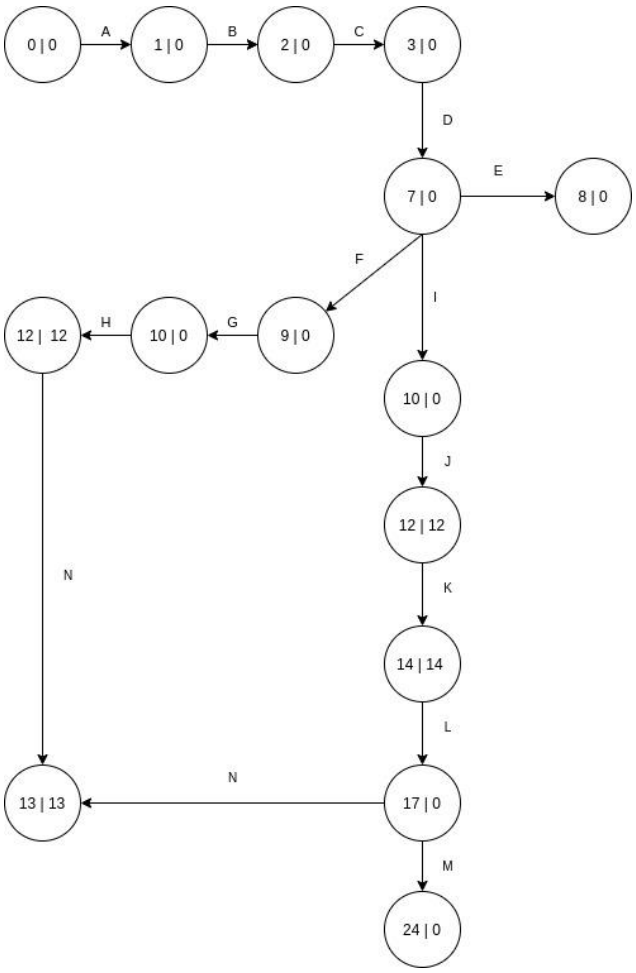
```python
class ShippingAddress(models.Model):
    "Represents a shipping adress of a user and a order"
    customer = models.ForeignKey(Customer, on_delete=models.SET_NULL, null=True, blank=True)
    order = models.ForeignKey(Order, on_delete=models.SET_NULL, blank=True, null=True)
    address = models.CharField(max_length=200, null=True)
    city = models.CharField(max_length=200, null=True)
    zip_code = models.CharField(max_length=200, null=True)
    date_added = models.DateTimeField(auto_now_add=True)
    country = models.CharField(max_length=100, null=True)

    def __str__(self):
        return self.address
```

Esta classe representa um endereço de entregas exclusivo a cada utilizador.

5.1.3.6 Cliente

5.2 Rede PERT

A rede PERT ajuda-nos a determinar o caminho crítico do projeto e ajuda também a manter a direção e tarefas pendentes do projeto em linha:



Caminho crítico:

- A, B, C, D, F, G, M, N;

Este é o caminho que temos de percorrer para que o projeto esteja minimamente pronto no menor tempo possível.

5.3 Layout de páginas

5.4 Urls

6 Conclusão

7 Web-Grafia

https://github.com/p1ng07/pap_site/

https://docs.djangoproject.com/en/3.0/