UNIVERSIDADE D
COIMBRA

Relatório do Projeto

André Couto (2019215056) Diogo Santiago (2019214326)

Base de Dados

Licenciatura em Matemática

Índice:

Diagrama ER

O nosso projeto tem como objetivo criar uma *web app* no *django* sobre canas de pesca, como tal construímos um diagrama entidade e relacionamento de forma a poder criar entidades com determinados atributos e relacioná-las da forma como pretendíamos.

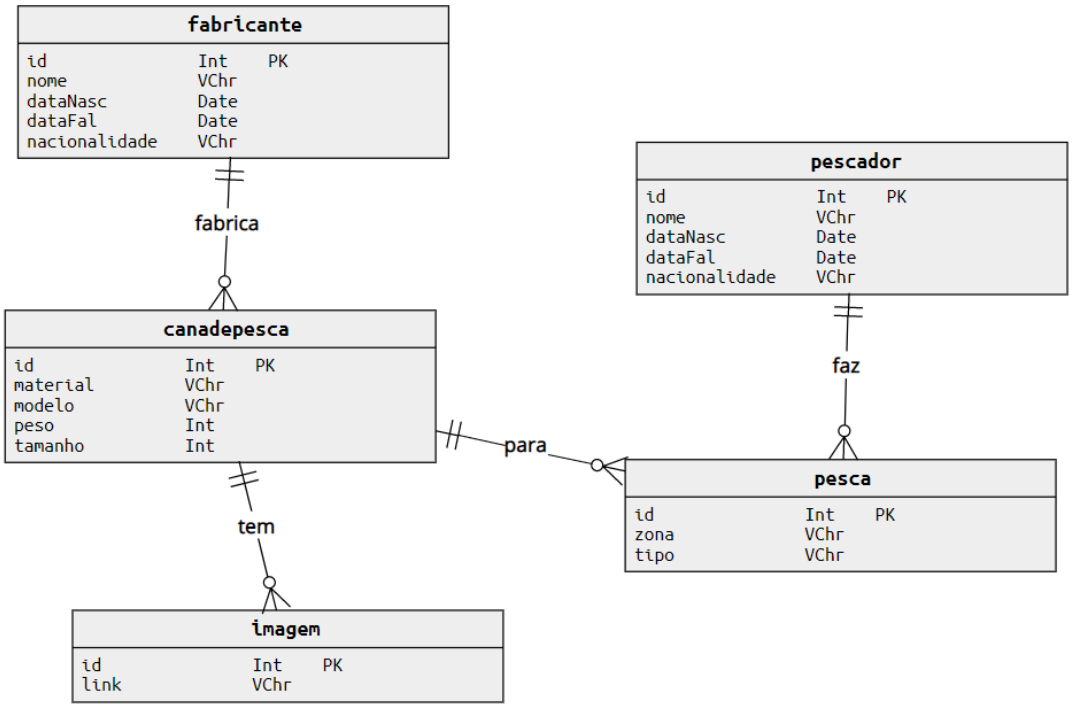
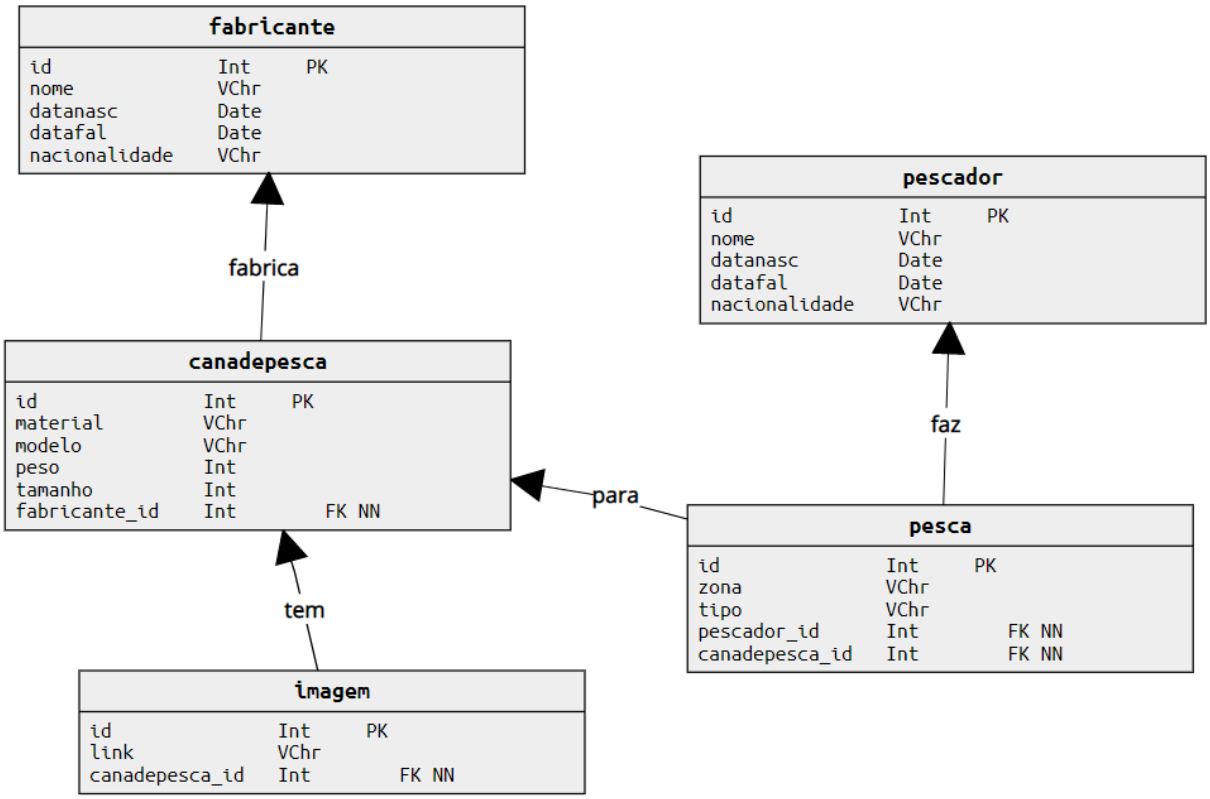


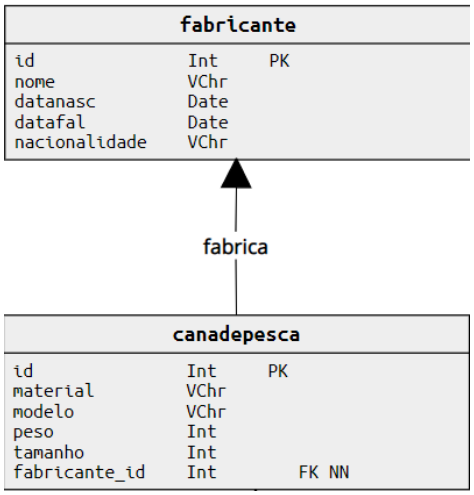
Diagrama Físico

A passagem para diagrama físico é feita de forma automática no *onda.dei.uc.pt* e é a partir deste diagrama que vamos construir a nossa models.py, assim como o views e templates.



Models.py

Como tínhamos dito anteriormente, para cada entidade no diagrama físico criamos uma *class* no models.py. Por exemplo, olhemos para as tabelas do *fabricante* e da *canadepesca*:



Então, no ficheiro models.py vamos ficar com:

```python
from django.db import models

class Fabricante(models.Model):
    nome = models.CharField(max_length=20, null=False, blank=False)
    datanasc = models.CharField(max_length=200, null=True, blank=True)
    datafal = models.CharField(max_length=200, null=True, blank=True)
    nacionalidade = models.CharField(max_length=200, null=True, blank=True)
    def __str__(self):
        return self.nome
    def isAlive(self):
        return datafal==False

class Canadepesca(models.Model):
    fabricante = models.ForeignKey(Fabricante, on_delete=models.CASCADE)
    material = models.CharField(max_length=200, null=True, blank=True)
    modelo = models.CharField(max_length=200, null=True, blank=True)
    peso = models.IntegerField(null=True, blank=True)
    tamanho = models.IntegerField(null=True, blank=True)
    def __str__(self):
        return self.fabricante.nome+"-"+self.modelo
```

Repetimos assim o processo para todas as tabelas e obtemos o nosso ficheiro models.py concluído.

Views e templates

Ora o ficheiro views.py e a pasta templates são aqueles que de certa forma constroem aquilo que nós vemos quando acedemos à nossa app.

Por exemplo, no views.py vamos contar quantos elementos de cada entidade temos.

```python
def index(request):
    template = loader.get_template('PTpesca/index.html')
    context = {
        'numFabricantes':Fabricante.objects.all().count(),
        'numCanasdepesca':Canadepesca.objects.all().count(),
        'numPescadores':Pescador.objects.all().count(),
        'numPescas':Pesca.objects.all().count(),
        'numImagens':Imagem.objects.all().count(),
    }
    return HttpResponse(template.render(context, request))
```

Visualmente a página inicial é definida a partir do index.html como podemos verificar em seguida:

```
<html>
<head>
<meta charset="UTF-8">
<title>Base de Dados : Canas de Pesca</title>
</head>
<body>
<h1>Base de Dados : Canas de Pesca</h1>
<h2><a href="/PTpesca/fabricantes"> {{numFabricantes}} Fabricantes</a></h2>
<h2><a href="/PTpesca/pescadores"> {{numPescadores}} Pescadores</a></h2>
<h2><a href="/PTpesca/canasdepesca"> {{numCanasdepesca}} Canas de Pesca</a></h2>
<h2><a href="/PTpesca/pescas"> {{numPescas}} Pescas</a></h2>
<h2><a href="/PTpesca/imagens"> {{numImagens}} Imagens de Canas de Pesca</a></h2>
```

E nesta vão aparecer os números definidos na views.py.

Portanto são assim definidos todos os nossos templates em primeiro lugar no views.py como podemos ver por exemplo no caso dos fabricantes.

No views.py temos:

```
def Fabricantes(request):
    template = loader.get_template('PTpesca/fabricantes.html')
    items = Fabricante.objects.order_by('nome')[0:]
    context = {
        'fabricantes':items
    }
    return HttpResponse(template.render(context, request))

def FabricanteDetalhes(request, fabricante_id):
    template = loader.get_template('PTpesca/fabricanteDetalhes.html')
    try:
        myFabricante = Fabricante.objects.get(pk=fabricante_id)
        myFabricanteCanasdepesca = Canadepesca.objects.filter(fabricante = fabricante_id)
        context = {'fabricante' : myFabricante, 'canasdepesca' : myFabricanteCanasdepesca}
    except Fabricante.DoesNotExist:
        raise Http404("Nao existe fabricante")

    return HttpResponse(template.render(context, request))
```

E nos respetivos ficheiros html vamos ter:

```
<a href="/PTpesca/">Home</a></p>
{% if fabricantes %}
    <ul>
    {% for Fabricante in fabricantes %}
        <li><a href="/PTpesca/fabricantes/{{Fabricante.id}}/">{{ Fabricante.nome }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>Nao ha registo de fabricantes.</p>
{% endif %}
```

```
<a href="/PTpesca/">Home</a></p>
<h1>{{fabricante.nome}} Detalhes</h1>
<h3>Nacionalidade: {{fabricante.nacionalidade}}</h3>
<h3>Data de Nascimento: {{fabricante.datanasc}}</h3>
<h3>Data de Falecimento: {{fabricante.datafal}}</h3>
<h3>Canas de Pesca fabricadas:</h3>
{% if canasdepesca %}
    <ul>
    {% for Canadepesca in canasdepesca %}
        <li><a href="/PTpesca/canasdepesca/{{Canadepesca.id}}/">{{Canadepesca.modelo}}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>Sem registo de canas fabricadas.</p>
{% endif %}
```

Mais uma vez repetimos o mesmo processo para todas as entidades que queremos na nossa app, definindo sempre no views.py e detalhando a forma como queremos visualizar nos ficheiros html dos templates.

Inserir dados

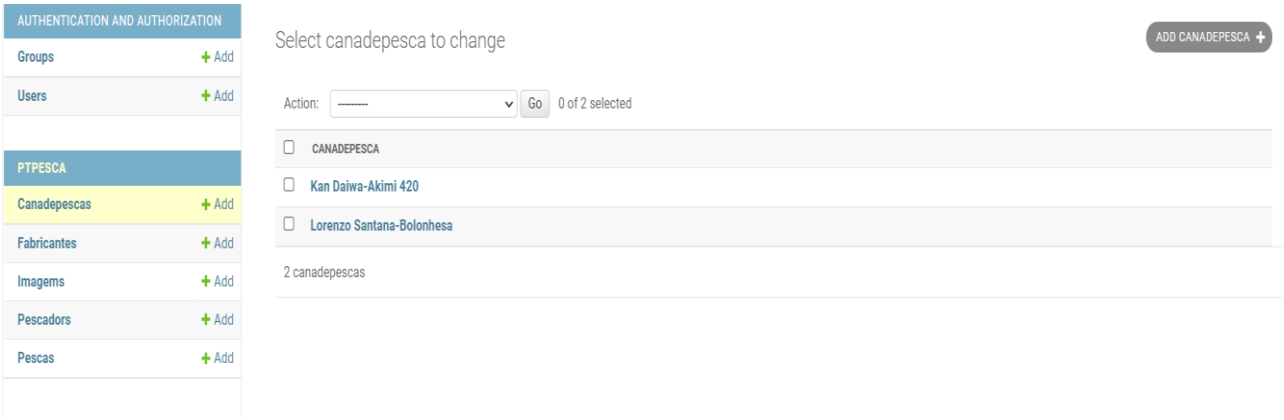
Para inserir os dados temos duas formas possíveis, ou a partir do *Query Tool* do *PgAdmin* na base de dados PTpesca ou usando o admin do django.

No primeiro caso apenas temos que colar no *Query Tool* os dados do ficheiro Dados.txt que tem como corpo:

```
insert into "PTpesca_fabricante" (nome, nacionalidade, datanasc, datafal)
values
('Lorenzo Santana', 'Espanha', '10/06/1968', '11/08/2021'),
('Kan Daiwa', 'Japão', '13/02/1948', null);


insert into "PTpesca_pescador" (nome, nacionalidade, datanasc, datafal)
values
('Andre Couto', 'Portugal', '11/01/2001', null),
('Diogo Santiago', 'Portugal', '17/08/2001', null);

insert into "PTpesca_canadepesca" (fabricante_id, material, modelo, peso, tamanho)
values
((select id from "PTpesca_fabricante" where nome = 'Lorenzo Santana'), 'carbono', 'Bolonhesa', 382, 147),
((select id from "PTpesca_fabricante" where nome = 'Kan Daiwa'), 'carbono preto', 'Akimi 420', 610, 420);

insert into "PTpesca_imagem" (canadepesca_id, link)
values
((select c.id from "PTpesca_canadepesca" c, "PTpesca_fabricante" f  where  f.id = c.fabricante_id and c.modelo='Bolonhesa' and f.nome='Lorenzo Santana'), 'https://www.daiwa.pt/uploads/prod
((select c.id from "PTpesca_canadepesca" c, "PTpesca_fabricante" f  where  f.id = c.fabricante_id and c.modelo='Akimi 420' and f.nome='Kan Daiwa'), 'https://www.formulapesca.com/11296-larg

insert into "PTpesca_pesca" (pescador_id, canadepesca_id, zona, tipo)
values
(
	(select id from "PTpesca_pescador" where nome='Andre Couto'),
	(select c.id from "PTpesca_canadepesca" c, "PTpesca_fabricante" f  where  f.id = c.fabricante_id and c.modelo='Akimi 420' and f.nome = 'Kan Daiwa'),
	'Esposende','Costeira'
),
(
	(select id from "PTpesca_pescador" where nome='Diogo Santiago'),
	(select c.id from "PTpesca_canadepesca" c, "PTpesca_fabricante" f  where  f.id = c.fabricante_id and c.modelo='Bolonhesa' and f.nome = 'Lorenzo Santana'),
	'Figueira da Foz','Em alto mar'
);
```

No segundo caso temos que criar um *superuser* na linha de comandos de modo a podermos inserir dados a partir do admin do django:

```
C:\Users\andre\Dropbox\pesca>python manage.py createsuperuser
Username (leave blank to use 'andre'): pesca
Email address:
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```



Apresentação final da app

Depois de tudo feito obtivemos então a nossa app que tem o seguinte aspeto:

Base de Dados : Canas de Pesca

2 Fabricantes

2 Pescadores

2 Canas de Pesca

2 Pescas

2 Imagens de Canas de Pesca

Quando clicamos por exemplo em Canas de Pesca obtemos o resultado:

Home

- Lorenzo Santana-Bolonhesa
- Kan Daiwa-Akimi 420

E agora escolhemos a cana cujo modelo é *Akimi 420:*

Home

Kan Daiwa-Akimi 420

Material: carbono preto

Modelo: Akimi 420

Peso em gramas: 610

Tamanho em centimetros: 420

Autoavaliação

Apesar do projeto não ser excessivamente elaborado conseguimos realizar todas as tarefas pretendidas e a nossa *web app* está perfeitamente funcional o que nos deixa convincentes que a nossa nota deverá ser entre 70%-80%.

A forma como realizamos o projeto foi maioritariamente em conjunto, o que foi de facto útil pois era mais fácil detetar os erros que um ou outro estivesse a cometer e portanto não dividimos tarefas em específico, como tal achamos que ambos merecemos a mesma avaliação.

Quanto às horas de dedicação ao projeto este já se encontra a ser realizado desde que começamos a aprender *django* nas aulas práticas, sendo que nesta última semana dedicamos cerca de 3 a 4 horas por dia, contudo, quando vimos o resultado final, achamos que valeu a pena!