Ficha de Avaliação Final
Curso: UFCD 10793
UFCD/Módulo/Temática: UFCD 10793 - **Fundamentos de Python**
Ação: 10793_2/AT & 10793_5/N
Formador/a: Sandra Liliana Meira de Oliveira
Data: março de 2025
Nome do Formando/a:

O presente documento é composto por uma parte teórica e uma parte prática. A resolução da parte prática corresponde à realização da Ficha de Avaliação Final.

Introdução ao Django

Lê o enunciado referente ao projeto em Flask.

O **Django** é um framework web de alto nível para Python, concebido para fomentar o desenvolvimento rápido e limpo de aplicações web. Ao contrário do Flask (micro-framework), o Django segue o princípio "batteries-included", oferecendo por defeito um ORM robusto, sistema de autenticação, painel administrativo, templates, roteamento, e muito mais.

A estrutura dos projetos em Django é mais rígida que no Flask, sendo baseada em projetos e aplicações modulares. Esta organização contribui para maior escalabilidade e manutenção a longo prazo. O Django encoraja a separação clara entre lógica de negócio, apresentação e persistência de dados através do padrão **MTV (Model-Template-View)**.

Para aplicações como a gestão de utilizadores e picagens (entradas/saídas), o Django é ideal, uma vez que facilita a criação de modelos, rotas (views), templates HTML com Bootstrap, e a interação com a base de dados com segurança e rapidez.

Uma app em Django com o mesmo objetivo da app desenvolvida em Flask tem a seguinte estrutura:

Prática

Tutorial Passo a Passo: Gestão de Utilizadores e Picagens com Django

---

Passo 1: Criar e configurar o projeto Django

1. **Criar pasta do projeto e ativar ambiente virtual:**

```
mkdir django_picagens
cd django_picagens
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate
```

2. **Instalar o Django:**

```
pip install django
```

3. **Criar projeto Django:**

```
django-admin startproject gestao_picagens .
```

4. **Criar app principal:**

```
python manage.py startapp core
```

5. **Adicionar a app ao ficheiro settings.py:**

```
INSTALLED_APPS = [
  ...
  'core',
]
```

Passo 2: Criar os modelos no core/models.py

```
from django.db import models

class User(models.Model):
  name = models.CharField(max_length=100)
  email = models.EmailField(unique=True)

  def __str__(self):
    return self.name

class Picagem(models.Model):
  TIPOS = [('Entrada', 'Entrada'), ('Saída', 'Saída')]

  user = models.ForeignKey(User, on_delete=models.CASCADE, related_name='picagens')
  timestamp = models.DateTimeField(auto_now_add=True)
  tipo = models.CharField(max_length=10, choices=TIPOS)

  def __str__(self):
    return f'{self.user.name} - {self.tipo} - {self.timestamp}'
```

Passo 3: Migrar a base de dados

```
python manage.py makemigrations
python manage.py migrate
```

Passo 4: Criar formulários em core/forms.py

```
from django import forms
from .models import User

class UserForm(forms.ModelForm):
    class Meta:
        model = User
        fields = ['name', 'email']
```

Passo 5: Criar as views em core/views.py

```
from django.shortcuts import render, redirect, get_object_or_404
from .models import User, Picagem
from .forms import UserForm

# Home
def home(request):
    users = User.objects.all()
    return render(request, 'core/index.html', {'users': users})

# Adicionar utilizador
def add_user(request):
    form = UserForm(request.POST or None)
    if form.is_valid():
        form.save()
        return redirect('home')
    return render(request, 'core/add_user.html', {'form': form})

# Editar utilizador
def edit_user(request, user_id):
    user = get_object_or_404(User, id=user_id)
    form = UserForm(request.POST or None, instance=user)
    if form.is_valid():
        form.save()
        return redirect('home')
    return render(request, 'core/edit_user.html', {'form': form})

# Apagar utilizador
def delete_user(request, user_id):
    user = get_object_or_404(User, id=user_id)
    user.delete()
    return redirect('home')

# Registar picagem
def registar_picagem(request, user_id, tipo):
    user = get_object_or_404(User, id=user_id)
    if tipo in ["Entrada", "Saída"]:
        Picagem.objects.create(user=user, tipo=tipo)
    return redirect('home')
```

```
# Ver picagens
def listar_picagens(request):
    picagens = Picagem.objects.select_related('user').order_by('-timestamp')
    return render(request, 'core/picagens.html', {'picagens': picagens})
```

Passo 6: Definir URLs

gestao_picagens/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('core.urls')),
]
```

core/urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('add/', views.add_user, name='add_user'),
    path('edit/<int:user_id>/', views.edit_user, name='edit_user'),
    path('delete/<int:user_id>/', views.delete_user, name='delete_user'),
    path('picagem/<int:user_id>/<str:tipo>/', views.registar_picagem, name='picagem'),
    path('picagens/', views.listar_picagens, name='picagens'),
]
```

Passo 7: Criar templates HTML com Bootstrap

Colocar os **templates** em core**/templates/core/:**

layout.html (base)

```
<!DOCTYPE html>
<html lang="pt">
<head>
    <meta charset="UTF-8">
    <title>Gestão de Picagens</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="{ % url 'home' % }">Início</a>
            <a class="nav-link text-white" href="{ % url 'add_user' % }">Adicionar Utilizador</a>
            <a class="nav-link text-white" href="{ % url 'picagens' % }">Picagens</a>
        </div>
    </nav>
    <div class="container mt-4">
```

```
    {% block content %}{% endblock %}
  </div>
</body>
</html>
```

index.html

```
{% extends 'core/layout.html' %}
{% block content %}
<h1 class="text-center">Utilizadores</h1>
<a class="btn btn-success mb-3" href="{% url 'add_user' %}">Adicionar Utilizador</a>
<table class="table">
  <thead>
    <tr>
      <th>ID</th>
      <th>Nome</th>
      <th>Email</th>
      <th>Ações</th>
    </tr>
  </thead>
  <tbody>
    {% for user in users %}
    <tr>
      <td>{{ user.id }}</td>
      <td>{{ user.name }}</td>
      <td>{{ user.email }}</td>
      <td>
        <a href="{% url 'edit_user' user.id %}" class="btn btn-primary">Editar</a>
        <a href="{% url 'delete_user' user.id %}" class="btn btn-danger">Apagar</a>
        <a href="{% url 'picagem' user.id 'Entrada' %}" class="btn btn-success">Entrada</a>
        <a href="{% url 'picagem' user.id 'Saída' %}" class="btn btn-warning">Saída</a>
      </td>
    </tr>
    {% endfor %}
  </tbody>
</table>
{% endblock %}
```

add_user.html

```
{% extends 'core/layout.html' %}
{% block content %}
<h2 class="text-center">Adicionar Utilizador</h2>
<form method="post" class="container">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit" class="btn btn-primary">Adicionar</button>
</form>
{% endblock %}
```

edit_user.html

```
{% extends 'core/layout.html' %}
{% block content %}
<h2 class="text-center">Editar Utilizador</h2>
<form method="post" class="container">
```

```
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit" class="btn btn-primary">Atualizar</button>
</form>
{% endblock %}
```

picagens.html

```
{% extends 'core/layout.html' %}
{% block content %}
<h1 class="text-center">Registo de Picagens</h1>
<table class="table">
  <thead>
    <tr>
      <th>Nome</th>
      <th>Tipo</th>
      <th>Data e Hora</th>
    </tr>
  </thead>
  <tbody>
    {% for p in picagens %}
    <tr>
      <td>{{ p.user.name }}</td>
      <td>{{ p.tipo }}</td>
      <td>{{ p.timestamp }}</td>
    </tr>
    {% endfor %}
  </tbody>
</table>
{% endblock %}
```

Passo 8: Testar a aplicação

python manage.py runserver

Abrir no browser: http://127.0.0.1:8000/

Temos agora um projeto Django funcional: com base de dados, CRUD de utilizadores, picagens, integração com Bootstrap, e estrutura modular.