Criação e (Re)utilização de Código Próprio e de Terceiros
Django Web Framework

1. Introdução

O desenvolvimento de aplicações web modernas exige a capacidade de criar funcionalidades do zero, bem como a habilidade de reutilizar e integrar código existente de forma eficiente. A reutilização de código, seja através de bibliotecas ou frameworks, reduz o esforço de desenvolvimento, melhora a qualidade do produto final e permite focar no que é realmente inovador. Além disso, frameworks como o Django oferecem uma infraestrutura robusta e flexível que simplifica a implementação de aplicações escaláveis e seguras.

Nesta atividade laboratorial, o desafio será criar uma aplicação web funcional que permita a gestão de tópicos de discussão, algo semelhante ao que encontramos em fóruns ou plataformas colaborativas. Esta aplicação deverá permitir que utilizadores publiquem tópicos, interajam através de comentários, e visualizem discussões organizadas. Para isso, será necessária dominar conceitos como a criação de modelos, rotas, interfaces e a utilização do ORM (Object-Relational Mapping) do Django para gerir a base de dados.

Adicionalmente, será incentivada a integração de bibliotecas de terceiros para adicionar funcionalidades ou melhorar a experiência do utilizador. O objetivo é proporcionar uma experiência prática que espelhe o processo de desenvolvimento de software no mundo real, onde o aproveitamento de ferramentas externas é essencial para aumentar a eficiência e reduzir redundâncias. Assim, espera-se que o estudante aprenda a desenvolver código do zero, bem como a integrar soluções já existentes, avaliando a sua qualidade e adequação ao projeto.

2. Objetivos e competências a desenvolver

Ao concluir esta atividade laboratorial, os estudantes deverão ser capazes de:

- Desenvolver uma aplicação web funcional utilizando o Django, implementando operações de CRUD (Create, Read, Update, Delete) de forma eficiente.
- Reutilizar e integrar bibliotecas de terceiros no projeto, avaliando a sua adequação e contribuindo para a escalabilidade e qualidade do código.
- Projetar e implementar bases de dados relacionais utilizando o ORM do Django, garantindo consistência e integridade dos dados.
- Adotar boas práticas de programação, incluindo modularidade, documentação, e testes unitários, para assegurar um código robusto e de fácil manutenção.
- Gerir o ciclo completo de desenvolvimento de uma aplicação web, desde o design inicial até à entrega final, incluindo configuração de ambientes e utilização de sistemas de controlo de versão.

3. Instruções

A atividade será dividida em seis etapas:

Etapa 1: Configuração inicial

1. Instale e configure um ambiente virtual Python para o projeto.

2. Instale o Django (versão mínima 4.0) e qualquer dependência necessária.
3. Configure uma nova aplicação Django chamada `topics`.

Etapa 2: Estrutura básica

1. Crie modelos para representar os seguintes elementos:
   - **Tópicos:** título, descrição, data de criação, autor.
   - **Comentários:** conteúdo, data de criação, autor, tópico relacionado.

```
from django.db import models
from django.contrib.auth.models import User


class Topic(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    author = models.ForeignKey(User, on_delete=models.CASCADE, related_name='topics')

    def __str__(self):
        return self.title


class Comment(models.Model):
    topic = models.ForeignKey(Topic, on_delete=models.CASCADE, related_name='comments')
    text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    author = models.ForeignKey(User, on_delete=models.CASCADE, related_name='comments')

    def __str__(self):
        return f"Comments on {self.topic.title} by {self.author.username}"
```

2. Configure a base de dados utilizando o ORM do Django.
3. Crie um superuser para aceder ao painel de administração do Django.

Etapa 3: Funcionalidades obrigatórias

1. **Listagem de tópicos:** Página inicial que apresenta todos os tópicos criados, com título e descrição.
2. **Detalhes de um tópico:** Página que apresenta os detalhes de um tópicos e os comentários associados.
3. **Criar tópicos:** Formulário para criar novos tópicos.
4. **Adicionar de comentários:** Formulário para adicionar comentários a um tópico existente.
5. **Editar e eliminar tópicos:** Permitir a edição e eliminação de tópicos e comentários (apenas pelo autor).

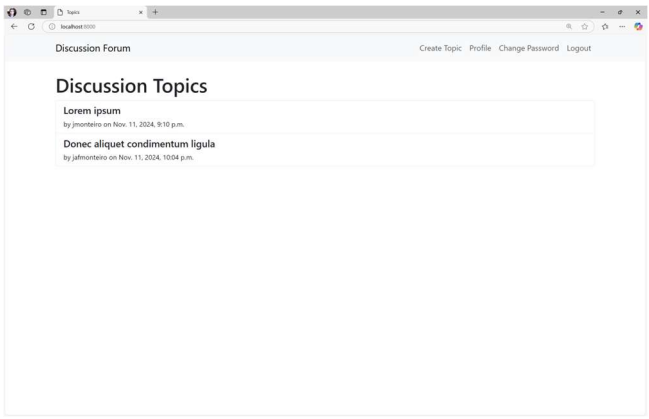

Figura 1. Listagem de tópicos.

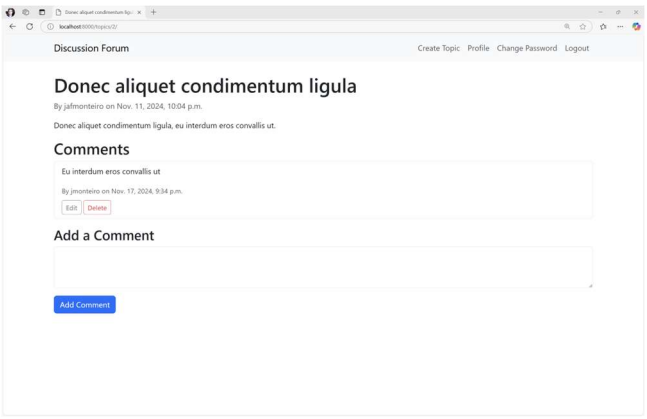

Figura 2. Detalhes de um tópico.

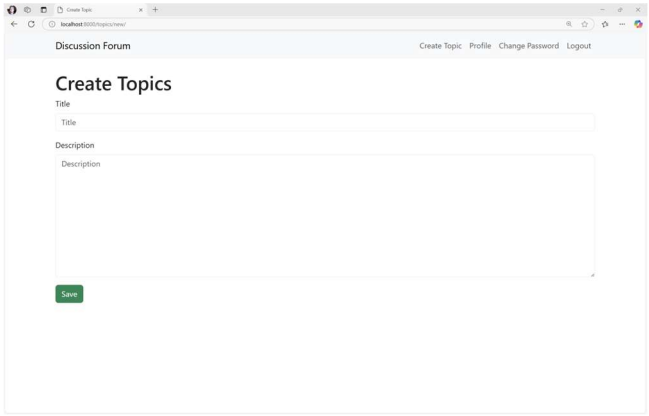

Figura 3. Criação de tópicos.

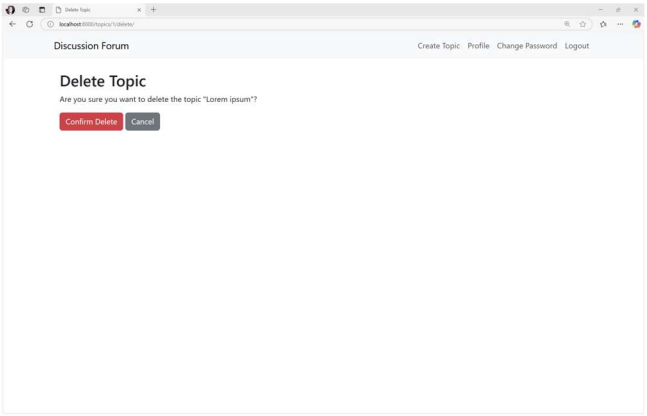

Figura 4. Eliminar um tópico de discussão.

Etapa 4. Utilização de bibliotecas de terceiros

1. Utilize pelo menos uma biblioteca externa, como o `django-crispy-forms` para melhorar o design dos formulários ou outra ferramenta, justificando a seleção.

Etapa 5. Testes e documentação

1. Escreva testes unitários para verificar o funcionamento das principais funcionalidades.
2. Documente o código seguindo boas práticas, incluindo comentários relevantes e um ficheiro `README.md` explicando o funcionamento da aplicação.

Etapa 6. Submissão

1. Entregue o código num repositório Git (GitHub, GitLab, ou outro) com um histórico de commits coerente.
2. Inclua instruções claras no `README.md` para instalação e execução do projeto.

Bom trabalho!