ANÁLISE DA EFICIÊNCIA E PRATICIDADE NA IMPLEMENTAÇÃO DE OPERAÇÕES CRUD UTILIZANDO A LINGUAGEM PYTHON E DJANGO: UM ESTUDO SOBRE DESEMPENHO E FLEXIBILIDADE EM FRAMEWORKS WEB

Antônio Fernandes De Santana Neto
Manoel Dantas Macedo Filho
Universidade Tiradentes - UNIT

25/08/2024

Abstract

Este artigo explora a eficiência e praticidade na implementação de operações CRUD (Create, Read, Update, Delete) utilizando a linguagem Python e o framework Django no desenvolvimento de aplicações web. Através de uma análise detalhada, são discutidos os aspectos técnicos que tornam o Django uma escolha robusta para a criação e gestão de sistemas de persistência de dados. Além de fornecer um guia prático, o artigo examina a flexibilidade e o desempenho do Django na execução de operações CRUD, destacando como sua arquitetura permite a rápida implementação e escalabilidade de aplicações web. Este estudo busca capacitar desenvolvedores a compreender as vantagens técnicas e as considerações práticas na escolha de tecnologias para projetos que demandam alta eficiência e manutenibilidade.
Palavras chave: CRUD, Python, Django, Desenvolvimento Web, Persistência de Dados, Eficiência, Flexibilidade.

This article explores the efficiency and practicality of implementing CRUD (Create, Read, Update, Delete) operations using the Python language and the Django framework in the development of web applications. Through a detailed analysis, the technical aspects that make Django a robust choice for creating and managing data persistence systems are explained. In addition to providing a practical guide, the article examines the flexibility and performance of Django in executing CRUD operations, highlighting how its architecture allows for the rapid implementation and scalability of web applications. This study aims to enable developers to understand the technical advantages and practical considerations in choosing technologies for projects that require high efficiency and maintainability.
Keywords: CRUD, Python, Django, Web Development, Data Persistence, Efficiency, Flexibility.

1 Introdução

No desenvolvimento de aplicações web, a manipulação eficiente de dados é fundamental para garantir a integridade e a funcionalidade dos serviços oferecidos. As operações CRUD (Create, Read, Update, Delete) representam a base do gerenciamento de dados, permitindo que informações sejam criadas, lidas, atualizadas e excluídas de maneira organizada e eficaz. Segundo a Mozilla Developer Network, "CRUD é um acrônimo para as maneiras de operar dados armazenados, e representa as quatro funções básicas do armazenamento persistente."

O Django, um dos frameworks mais populares para Python, se destaca por oferecer um conjunto robusto de ferramentas que facilitam a implementação dessas operações em qualquer aplicação web. Este artigo tem como objetivo geral analisar como o Django, aliado ao Python, permite a construção eficiente e flexível de um sistema CRUD, com foco na rapidez de desenvolvimento e na adaptabilidade às necessidades específicas de diferentes projetos.

Já os Objetivos Específicos transitam entre os seguintes interesses: "Demonstrar a implementação prática de um CRUD utilizando Django, explorando as etapas de criação, leitura, atualização e exclusão de dados"; "Avaliar como o Django facilita a integração de regras de negócio complexas, mantendo a estrutura do código clara e organizada"; "Analisar a eficiência do Django em termos de desempenho e facilidade de manutenção, especialmente no contexto de aplicações escaláveis"; e "Investigar a flexibilidade do Django na adaptação de sistemas CRUD para diferentes cenários de aplicação, desde pequenas soluções até sistemas de grande porte.

A Justificativa na escolha de Python e Django para este estudo se dá pela ampla adoção dessas tecnologias na indústria de desenvolvimento web, motivada por sua combinação de simplicidade e poder. Python é reconhecido por sua clareza e facilidade de aprendizado, enquanto o Django se destaca por sua abordagem completa, que inclui um vasto conjunto de funcionalidades prontas para uso. A análise apresentada neste artigo visa preencher uma lacuna na literatura técnica, ao oferecer tanto um guia prático quanto uma análise aprofundada das capacidades do Django, especialmente em comparação com outros frameworks. Dessa forma, o artigo busca capacitar desenvolvedores a escolherem as ferramentas mais adequadas para implementar sistemas CRUD que sejam ao mesmo tempo eficientes, escaláveis e fáceis de manter.

2 Fundamentação Teórica

A eficiência e a praticidade na implementação de operações CRUD são aspectos cruciais no desenvolvimento de sistemas web modernos, especialmente em contextos onde a escalabilidade e a manutenibilidade são requisitos indispensáveis. Esses dois conceitos, embora distintos, estão profundamente interligados quando se trata da escolha de tec-

nologias e frameworks para o desenvolvimento web.

2.1 Eficiência no Desenvolvimento Web

A eficiência no desenvolvimento web refere-se à capacidade de um sistema em executar operações de maneira rápida e com baixo consumo de recursos. No contexto das operações CRUD, essa eficiência é medida pela rapidez com que o sistema pode processar solicitações de criação, leitura, atualização e exclusão de dados. Além disso, a eficiência também envolve o uso otimizado de recursos como memória, CPU e I/O (entrada e saída), que são críticos em sistemas que necessitam de alta disponibilidade e baixo tempo de resposta.

O Django, como um framework de alto nível para Python, oferece uma série de ferramentas e otimizações que contribuem para essa eficiência. A utilização de ORM (Object-Relational Mapping), por exemplo, permite que desenvolvedores manipulem bancos de dados de forma mais intuitiva e eficiente, sem a necessidade de escrever consultas SQL complexas. Estudos como o de Silva et al. (2020) destacam que a abstração proporcionada pelo ORM do Django pode reduzir significativamente o tempo de desenvolvimento e minimizar erros relacionados à manipulação direta do banco de dados.

2.2 Praticidade e Flexibilidade no Uso de Frameworks

Praticidade no desenvolvimento refere-se à facilidade e rapidez com que os desenvolvedores podem implementar funcionalidades, adaptar o sistema a novos requisitos e manter o código ao longo do tempo. Um framework é considerado prático quando ele reduz a complexidade do código, promove a reutilização e facilita a integração de novas funcionalidades sem comprometer a estrutura existente do sistema.

O Django se destaca por sua abordagem "batteries-included", que significa que ele vem com uma série de funcionalidades pré-configuradas e prontas para uso. Isso inclui um sistema de autenticação robusto, um painel administrativo automático e ferramentas para a criação de APIs RESTful. Essa característica permite que desenvolvedores concentrem seus esforços na lógica de negócios específica da aplicação, ao invés de reinventar soluções comuns.

Além disso, a flexibilidade do Django é outro ponto chave. O framework permite que as operações CRUD sejam personalizadas para atender a requisitos específicos, seja pela modificação de comportamentos padrão ou pela integração de pacotes externos. O uso de mixins e viewsets no Django REST Framework, por exemplo, facilita a criação de APIs customizadas, que podem ser ajustadas para oferecer o equilíbrio ideal entre desempenho e funcionalidade.

2.3 Versatilidade e Aplicabilidade em Diferentes Cenários

Uma das grandes vantagens do Django é a sua versatilidade, que permite que aplicações desenvolvidas com esse framework sejam adaptadas para uma ampla variedade de cenários e regras de negócio. Essa versatilidade é possível graças à arquitetura modular do Django, que permite que os desenvolvedores adicionem ou removam funcionalidades conforme necessário, ajustando a aplicação às demandas específicas de diferentes projetos.

Por exemplo, uma aplicação CRUD desenvolvida em Django pode ser facilmente adaptada para atender a diferentes setores, como e-commerce, educação, saúde, entre outros. A principal estrutura de dados e operações permanece consistente, enquanto as regras de negócio específicas podem ser customizadas através da implementação de modelos personalizados, formulários, e lógica de backend. Esse nível de customização é facilitado pelo uso de Django signals, middleware, e a robusta API de modelagem.

Além disso, o painel administrativo nativo do Django é uma ferramenta poderosa que torna a aplicação facilmente auditável e gerenciável. Esse painel permite que os administradores monitorem e gerenciem o sistema de maneira intuitiva, o que é especialmente útil em ambientes onde a integridade e a segurança dos dados são críticas. A capacidade de auditar e ajustar rapidamente as operações através do painel administrativo torna o Django uma escolha atrativa para projetos que exigem flexibilidade e adaptabilidade.

Essa capacidade de adaptação é crucial para garantir que a aplicação possa evoluir com as necessidades do negócio, permitindo que o mesmo código base seja reutilizado em diferentes contextos com ajustes mínimos. A robustez do Django, combinada com sua flexibilidade, possibilita a criação de soluções que são ao mesmo tempo específicas para um contexto e escaláveis para atender novas demandas.

2.4 Comparação com Outros Frameworks

Embora o Django seja amplamente reconhecido por sua eficiência e praticidade, é importante considerar como ele se compara a outros frameworks. Frameworks como Flask (para Python) e Laravel (para PHP) são frequentemente mencionados em comparações devido à sua simplicidade e flexibilidade. No entanto, a escolha entre Django e outros frameworks deve ser guiada pelos requisitos específicos do projeto. O Django é geralmente preferido em projetos que demandam uma estrutura mais rígida e que precisam escalar rapidamente, enquanto o Flask pode ser mais adequado para aplicações menores e mais flexíveis.

A análise teórica apresentada aqui estabelece a base para a discussão prática que será abordada na metodologia deste estudo. Ao entender os princípios de eficiência, praticidade, e versatilidade no uso de frameworks como Django, desenvolvedores podem fazer escolhas mais informadas que impactam diretamente a qualidade e o sucesso de seus projetos.

3 Metodologia

Este artigo tem como foco a implementação de um sistema de gerenciamento de usuários que abrange todas as operações CRUD (Create, Read, Update, Delete) de maneira visual e de fácil compreensão. Para facilitar a interação com o sistema, será criado um "superusuário" que possui permissões administrativas completas, permitindo a gestão de outros usuários dentro da aplicação. Essa abordagem é fundamental para garantir que os administradores possam realizar operações essenciais, como a criação de novos usuários, a leitura de informações, a atualização de dados existentes e a exclusão de usuários que não sejam mais necessários.

O desenvolvimento do sistema seguirá uma metodologia estruturada, começando pela configuração do ambiente de desenvolvimento com Python e Django. Inicialmente, será necessário instalar as dependências necessárias e configurar o banco de dados, que servirá como a base para a persistência de dados. Em seguida, serão definidas as rotas e as views que permitirão a interação do usuário com o sistema, garantindo que todas as operações CRUD sejam implementadas de maneira eficiente e intuitiva.

Além disso, o sistema contará com a implementação de formulários que facilitarão a entrada de dados e a validação das informações fornecidas pelos usuários. Os formulários serão projetados para serem amigáveis, garantindo que os administradores tenham uma experiência fluida ao inserir ou modificar dados. Assim, será possível validar as informações em tempo real e fornecer feedback imediato, o que é crucial para a manutenção da integridade dos dados.

Embora este artigo não aborde a criação de "grupos de usuários", é importante destacar que essa funcionalidade é facilmente integrável ao sistema. A adição de grupos de usuários permitirá uma auditoria e um controle de permissões mais granulares, possibilitando a segmentação de acessos e a definição de diferentes níveis de permissão para cada grupo. Essa flexibilidade é um dos grandes benefícios do uso do Django, pois permite que o sistema se adapte às necessidades específicas de cada aplicação, garantindo a escalabilidade e a robustez necessárias para um gerenciamento eficaz de dados.

4 Materiais e Métodos

4.1 Sistema Operacional

Neste artigo, foi utilizado o sistema operacional Linux, especificamente a versão Debian 5.10.0-18-amd64 (SMP Debian 5.10.140-1, lançado em 02 de setembro de 2022, arquitetura x86-64).

4.2 Python 3.11.9

Foi utilizada a linguagem de programação Python, na versão 3.11.9.

5 Implementação

5.1 Criaçao do diretório

Para fins de organização, foi criado uma pasta chamada "crud" e posteriormente adentrado na mesma, como ilustra a imagem a seguir:

```
antonio@debian:~$ mkdir crud
antonio@debian:~$ cd crud
```

5.2 Ambiente Virtual

Também para fins de organização, será criado um ambiente virtual onde instalaremos o Django e todas suas respectivas dependências.

```
antonio@debian:~/crud$ python3 -m venv venv
antonio@debian:~/crud$ ls
venv
```

Depois de criar o ambiente virtual, iremos ativar o mesmo:

```
antonio@debian:~/crud$ source venv/bin/activate
(venv) antonio@debian:~/crud$
```

5.3 Django

5.3.1 Instalação

Após configurar e acessar o ambiente virtual, instala-se o Django

```
(venv) antonio@debian:~/crud$ pip install django
Collecting django
  Downloading Django-4.2.14-py3-none-any.whl (8.0 MB)
     |████████████████████████████████| 8.0 MB 81 kB/s
Collecting asgiref<4,>=3.6.0
  Using cached asgiref-3.8.1-py3-none-any.whl (23 kB)
Collecting sqlparse>=0.3.1
  Downloading sqlparse-0.5.1-py3-none-any.whl (44 kB)
     |████████████████████████████████| 44 kB 297 kB/s
Collecting typing-extensions>=4
  Downloading typing_extensions-4.12.2-py3-none-any.whl (37 kB)
Installing collected packages: typing-extensions, sqlparse, asgiref, django
Successfully installed asgiref-3.8.1 django-4.2.14 sqlparse-0.5.1 typing-extensions-4.12.2
(venv) antonio@debian:~/crud$
```

5.3.2 Criação da aplicação

Agora com o Django instalado, basta criar o projeto:

```
(venv) antonio@debian:~/crud$ django-admin startproject crud
(venv) antonio@debian:~/crud$
```

5.3.3 Migrations

Com a aplicação já criada, precisamos modelar o banco de dados que armazenará as informações. O Django já traz um recurso para facilitar todo o processo de modelagem, chamado de "Migrations", que basicamente são arquivos estáticos que conseguem salvar os processos de alterações nas tabelas cronologicamente. "Migrations são a maneira do Django de propagar as alterações feitas em seus Models (adicionando um campo, excluindo um modelo, etc.) em seu esquema de banco de dados." (Documentação oficial do Django). Estes "Models" são arquivos criados manualmente pelo desenvolvedor para modular alguma tabela do banco de dados, todavia existem algumas que já vem por padrão criadas, dentre elas, o Model de "User" (usuário). Então podemos aplicar os arquivos de migrations que já vem com o framework e teremos o banco de dados modularizados para auditar usuários.

```
(venv) antonio@debian:~/crud$ python3 crud/manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
(venv) antonio@debian:~/crud$
```

5.3.4 Super Usuário

Quando todas as aplicações dos arquivos de Migrations forem aplicados, poderemos criar o "super usuário" que auditará toda e qualquer informação do sistema. Neste caso foi criado com o nome de usuário "root" e senha "root".

```
(venv) antonio@debian:~/crud$ python3 crud/manage.py createsuperuser
Username (leave blank to use 'antonio'): root
Email address: root@root.com
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
(venv) antonio@debian:~/crud$
```

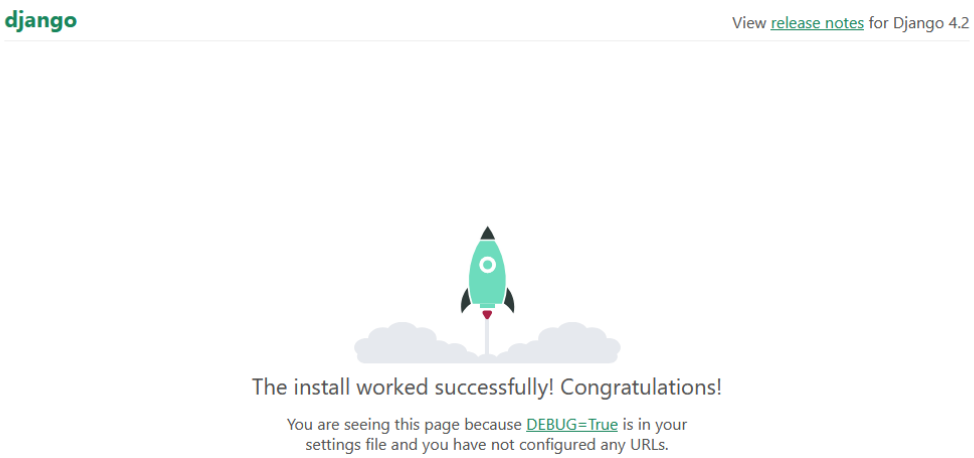
5.3.5 Executar a Aplicação

Depois de seguir todos os passos anteriores, a aplicação está pronta para ser utilizada. Após executada, estará disponível em seu endereço local na porta 8000 por padrão.

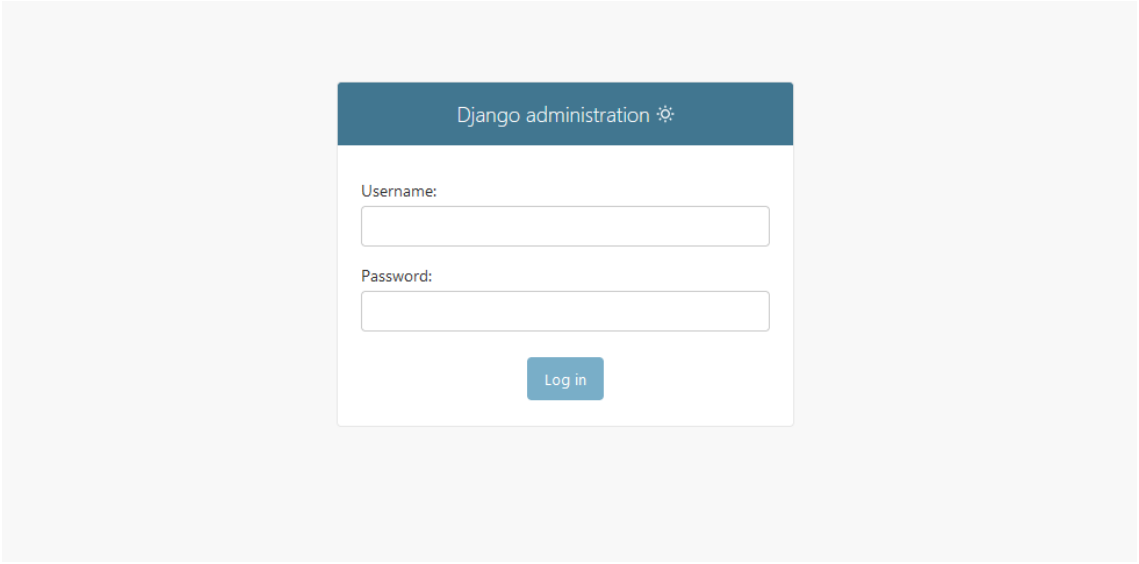
```
(venv) antonio@debian:~/crud$ python3 crud/manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 21, 2024 - 03:24:44
Django version 4.2.14, using settings 'crud.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

6 Resultados

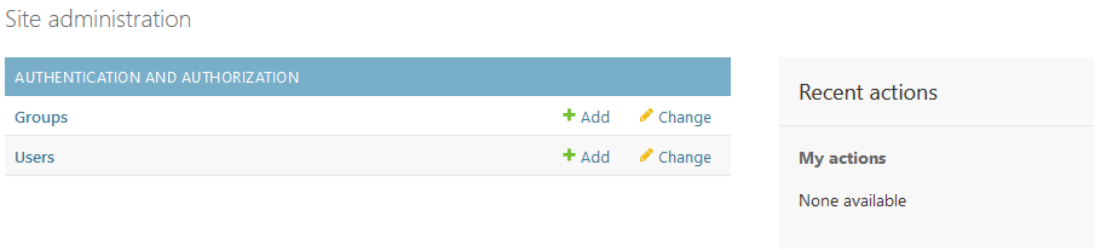
Os resultados incluem um aplicativo web funcional que permite realizar operações de criação, leitura, atualização e exclusão de usuários. Após completar todo o processo de implementação, a aplicação deve estar disponível em "http://localhost:8000". Acessando o endereço indicado este é o resultado:



Acessando a rota "/admin" temos o seguinte resultado:



Utilizando o acesso anteriormente criado para o super usuario, é possivel efetuar o login e acessar o ambiente admnistrativo.



7 Conclusão

Em conclusão, este artigo demonstrou a viabilidade e a simplicidade de implementar persistência de dados utilizando Python e Django. A abordagem passo a passo proporcionou uma compreensão clara das etapas necessárias para construir uma aplicação web básica.

A implementação de CRUD é fundamental em praticamente qualquer aplicação web, e dominar essa habilidade com Django permite a criação de soluções eficientes e escaláveis.

8 Referências

1. MOZILLA DEVELOPER NETWORK. MDN Web Docs: Django Overview. 2024. Disponível em: `https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django`. Acesso em: 20 jul. 2024.

2. DJANGO SOFTWARE FOUNDATION. Django Documentation. 2024. Disponível em: `https://docs.djangoproject.com/en/5.1/`. Acesso em: 17 jul. 2024.

3. RAMALHO, Luciano. Python Fluente: Programação Clara, Concisa e Eficaz. 2. ed. Rio de Janeiro: Novatec Editora, 2022.

4. Debian Linux. Debian Linux 5.10.0-18-amd64. Disponível em: `https://packages.qa.debian.org/l/linux.html`. Acesso em: 18 jul. 2024.

5. Python Software Foundation. Python 3.11.9 Release. Disponível em: `https://www.python.org/downloads/release/python-3119/`. Acesso em: 18 jul. 2024.