# **Relatório de Projeto**

# **Contra:**

# **The Contrarian Report**

**Introdução**

O presente relatório documenta o desenvolvimento da plataforma web "Contra: The Contrarian Report", realizado no âmbito das UFCDs 5421 e 5422 de Integração de Sistemas, do curso CET-9674 - Técnico Especialista em Tecnologia e Programação de Sistemas de Informação.

A plataforma Contra é um sistema de assinatura de conteúdos focado em relatórios e artigos sobre investimentos e trading. O sistema implementa diferentes níveis de acesso a conteúdos, baseado no tipo de utilizador e modalidade de assinatura. O projeto foi desenvolvido utilizando o framework Django, Python como linguagem de programação principal e MySQL como sistema de gestão de base de dados.

O objetivo principal foi criar uma aplicação web completa, incluindo sistema de autenticação, gestão de utilizadores, criação e categorização de artigos, e integração com sistemas de pagamento, seguindo as melhores práticas de desenvolvimento web moderno.

**Desenho da Estrutura**

**Arquitetura Geral**

A plataforma Contra segue uma arquitetura cliente-servidor baseada no padrão MVT (Model-View-Template) do Django:

+----------------+ +------------------+ +------------------+

| | | | | |

| Cliente | <--> | Servidor | <--> | BD |

| (Browser) | | (Django) | | (MySQL) |

| | | | | |

+----------------+ +------------------+ +------------------+

^

|

v

+----------——-----+

| |

| Serviços |

| Externos |

| (PayPal, |

| Google Auth) |

| |

+-------------——-+

**Componentes Principais**

#### **Cliente (Frontend)**

Embora o frontend não fosse um requisito principal do projeto, utilizou-se o framework Bootstrap para garantir uma interface responsiva e moderna. O cliente comunica com o servidor através de requisições HTTP.

#### **Servidor (Backend - Django)**

O núcleo da aplicação é desenvolvido em Django, que gere:

* Autenticação e autorização de utilizadores
* Lógica de negócio para assinaturas
* Gestão de artigos
* Integrações com serviços externos

#### **Base de Dados (MySQL)**

Armazena todos os dados da aplicação, incluindo:

* Informações dos utilizadores
* Detalhes das assinaturas
* Artigos e seu conteúdo
* Configurações do sistema

#### **Serviços Externos**

A plataforma integra-se com:

* PayPal: Para processamento de pagamentos das assinaturas

**Fluxo de utilizadores**

+---------------+

| |

| Utilizador |

| Anónimo |

| |

+--------------+

|

+-------------+--------------+

| |

+-------v--------+ +--------v-------+

| | | |

| Cliente | | Analista |

| | | |

+-------+--------+ +----------------+

|

+-------+--------+

| |

+---+ Standard +---+

| | | |

| +----------------+ |

| |

| +----------------+ |

| | | |

+---+ Premium +---+

| |

+----------------+

**Implementação**

**Esquema da Base de Dados**

O sistema implementa as seguintes tabelas principais:

#### **account\_customuser**

Extensão do modelo de utilizador padrão do Django para incluir campos específicos para a plataforma Contra, como tipo de utilizador (cliente ou analista).

#### **account\_customuser\_groups e account\_customuser\_user\_permissions**

Tabelas de relação do sistema de permissões do Django.

#### **client\_subscription**

Armazena as informações da assinatura de cada cliente, incluindo o tipo (standard ou premium), data de início e fim, e estado do pagamento.

#### **client\_planchoice**

Contém os diferentes planos de assinatura disponíveis, com detalhes de preço e características.

#### **writer\_article**

Armazena os artigos produzidos pelos analistas, incluindo título, conteúdo, data de publicação, autor e nível de acesso necessário.

**Implementação do Sistema de Utilizadores**

O sistema implementa três tipos de utilizadores:

1. **Anónimos**: Não registados, com acesso limitado à plataforma
2. **Clientes**: Utilizadores registados que podem aceder a:

* Artigos gratuitos (todos os clientes)
* Artigos standard (com assinatura standard ou superior)
* Artigos premium (apenas com assinatura premium)

1. **Analistas**: Utilizadores que produzem conteúdo e determinam o nível de acesso aos seus artigos

**Implementação de uma View em Django**

A implementação de uma view em Django segue um processo sistemático que interliga vários componentes do framework:

#### **Criação do Modelo de Dados (models.py)**

#### **Criação da View (views.py)**

#### **Criação do Template (templates/article/detail.html)**

#### **Configuração das URLs (urls.py)**

#### **Registo no Django Admin (admin.py)**

**Interação com a Base de Dados sem SQL Direto**

Uma das grandes vantagens do Django é o seu ORM (Object-Relational Mapping), que permite interagir com a base de dados sem escrever código SQL direto. Isto traz benefícios significativos:

1. **Abstração da base de dados**: O mesmo código funciona com diferentes sistemas de gestão de base de dados (MySQL, PostgreSQL, SQLite).
2. **Segurança**: O ORM previne automaticamente ataques de injeção SQL, pois os parâmetros são sempre sanitizados.
3. **Produtividade**: Reduz significativamente a quantidade de código necessário para operações CRUD.
4. **Manutenção**: Facilita a alteração do modelo de dados, pois o ORM gere automaticamente as migrações necessárias.

Exemplo de como o Django ORM substitui o SQL direto:

# Em SQL direto seria algo como:

# SELECT \* FROM writer\_article WHERE id = 5;

# Com Django ORM:

article = Article.objects.get(id=5)

# Em SQL seria:

# INSERT INTO writer\_article (title, content, author\_id, article\_type)

# VALUES ('Novo Artigo', 'Conteúdo...', 1, 'premium');

# Com Django ORM:

new\_article = Article(

title='Novo Artigo',

content='Conteúdo...',

author=request.user,

article\_type='premium'

)

new\_article.save()

**Conclusão**

**Resultados Alcançados**

O projeto Contra: The Contrarian Report foi desenvolvido com sucesso, implementando as funcionalidades principais definidas nos requisitos:

* Sistema de autenticação (email/password)
* Diferentes tipos de utilizadores (anónimo, cliente, analista)
* Sistema de assinaturas (standard e premium)
* Categorização de artigos por nível de acesso
* Integração com sistema de pagamento PayPal

**Melhorias Futuras**

Embora o projeto atenda aos requisitos principais, algumas funcionalidades complementares poderiam ser implementadas no futuro:

* Suporte a outras formas de pagamento (Stripe)
* Internacionalização completa do site
* Sistema de recomendação de artigos baseado em interesses do utilizador
* Implementação de uma API para acesso por aplicações móveis
* Melhorias na interface do utilizador e experiência de utilização

**Aprendizagens e Reflexões**

O desenvolvimento deste projeto proporcionou aprendizagens valiosas sobre:

* Integração de sistemas heterogéneos (Django + serviços externos)
* Implementação de sistemas de autenticação seguros e extensíveis
* Modelação de dados para aplicações web complexas
* Boas práticas de desenvolvimento em Django

A utilização do Django como framework principal mostrou-se uma escolha acertada pela sua robustez, segurança e produtividade. A integração com o ORM do Django eliminou a necessidade de escrever código SQL direto, o que acelerou significativamente o desenvolvimento.

**Críticas e Considerações**

Uma limitação do projeto foi o tempo disponível para implementação, numa altura em que diversos projetos estão a ser elaborados em simultâneo, que impediu o desenvolvimento de algumas funcionalidades do âmbito alargado.

A implementação de uma arquitetura de microserviços poderia também ser considerada para futuros desenvolvimentos, permitindo maior escalabilidade e manutenção independente dos diferentes componentes do sistema.

Em conclusão, o projeto Contra representou uma oportunidade valiosa de aplicação prática dos conhecimentos adquiridos nas UFCDs de Integração de Sistemas, resultando num sistema funcional que atende às necessidades especificadas.