Relatório de Projeto:

Amazon Clone

Relatório de Projeto: Amazon Clone

Tiago Batista Montenegro Galano

ESAS

Desenvolvimento de Aplicações Web – UFCD 10789

Braga, 30/06/2025

Relatório de Projeto:

Amazon Clone

Índice

Relatório de Projeto:

Amazon Clone

Índice de Imagens

Relatório de Projeto:

Amazon Clone

1. Introdução

1.1.Objectivo do Projecto

Este projecto tem como finalidade o desenvolvimento de uma aplicação web que reproduz funcionalidades essenciais da plataforma de e-commerce Amazon, utilizando a framework Django (Python).

A escolha do Django justifica-se pela sua integração com Python e a estrutura MTV, que separa dados, lógica e apresentação, facilitando manutenção e escalabilidade. O sistema contempla funcionalidades como registo de utilizadores, catálogo de produtos, carrinho de compras e sistema de pagamentos, aliado a um design responsivo e seguro.

1.2.Justificação

Este projecto permite superar a lacuna entre teoria e prática, promovendo o desenvolvimento de soluções tecnológicas aplicadas, num ambiente controlado. Alinha-se com os objectivos pedagógicos institucionais e aprofunda temas como usabilidade, segurança e escalabilidade.

2. Requisitos do Sistema

2.1 Requisitos Funcionais

O sistema em questão é uma aplicação web de comércio eletrônico, construída sobre o framework Django, que implementa funcionalidades fundamentais para a operação de uma plataforma similar à Amazon. A seguir, descrevem-se os perfis de usuários, suas permissões e as funcionalidades disponibilizadas, detalhando os diversos módulos que compõem o sistema.

Relatório de Projeto:
Amazon Clone

Perfis de Usuários e Funcionalidades:

1. Usuário Comum (Cliente)

Cadastro e Autenticação: Possibilidade de criar conta, realizar login e logout, e gerenciar informações pessoais via o módulo `accounts`.

Navegação e Visualização de Produtos: Visualização do catálogo, incluindo categorias e detalhes dos produtos, por meio do módulo `products`. Suporte à exibição de imagens armazenadas no módulo `media`.

Gerenciamento do Carrinho de Compras: Adição, alteração de quantidade e remoção de produtos no carrinho, através do módulo `cart`.

Finalização e Acompanhamento de Pedidos: Realização de pedidos via o módulo `orders`, incluindo fornecimento de informações para pagamento e envio, além da consulta do histórico e status dos pedidos.

2. Administrador

Gerenciamento de Produtos: Cadastramento, edição e exclusão de produtos e categorias, utilizando o módulo `products`.

Administração de Pedidos: Visualização e controle do status dos pedidos realizados na plataforma, via módulo `orders`.

Gestão de Usuários: Administração dos perfis de usuário (clientes e administradores) por meio do módulo `accounts`.

Módulos e suas funcionalidades detalhadas:

`accounts/` - Responsável pela gestão dos usuários, incluindo registro, autenticação, controle de permissões, e gerenciamento de perfis.

`products/` - Gestão do catálogo de produtos, englobando cadastro, edição, exclusão e organização por categorias, além da exibição das informações e imagens para os usuários.

`cart/` - Implementação do carrinho de compras, que permite aos usuários armazenar temporariamente os produtos selecionados, modificar quantidades e preparar a compra.

`orders/` - Controle do processo de pedidos, desde a finalização da compra até o acompanhamento do status, garantindo a comunicação do fluxo entre usuário e sistema.

`media/` - Armazenamento e gerenciamento dos arquivos de mídia, principalmente imagens associadas aos produtos, assegurando uma melhor apresentação visual da plataforma.

Funcionalidades Gerais do Sistema

Segurança: Implementação de autenticação e autorização, garantindo que cada usuário acesse somente as funcionalidades permitidas ao seu perfil.

Persistência e Integridade dos Dados: Uso de banco de dados relacional para armazenar informações dos usuários, produtos, carrinho e pedidos.

Interface Responsiva: Adaptação da interface para diferentes dispositivos, proporcionando uma experiência de navegação fluida.

Gerenciamento de Mídia: Suporte ao upload e exibição de imagens dos produtos para melhor visualização.

3. Desenvolvimento do Projeto

3.1 Arquitetura do Sistema

A aplicação segue a arquitetura típica de projetos desenvolvidos com o framework Django, que dota o padrão Model-View-Template (MVT). Essa arquitetura promove a separação clara entre a lógica de negócio, a apresentação e o gerenciamento dos dados, facilitando manutenção e escalabilidade.

Estrutura de Pastas e Módulos Principais: O sistema está organizado em múltiplos módulos (apps), cada um responsável por uma funcionalidade específica, conforme descrito abaixo:

- `accounts/`: Gerencia o cadastro, autenticação e gerenciamento dos usuários.
- `products/`: Contém os modelos, views e templates relacionados ao catálogo de produtos.
- `cart/`: Implementa a funcionalidade de carrinho de compras.
- `orders/`: Controla o processamento e acompanhamento dos pedidos.
- `media/`: Diretório destinado ao armazenamento dos arquivos de mídia, como imagens dos produtos.
- `amazon_clone/`: Diretório principal da aplicação, contendo configurações gerais, URLs e inicialização.

Organização dos Arquivos: Cada módulo possui uma estrutura padrão do Django, com subpastas para `models.py` (definição do modelo de dados), `views.py` (lógica de controle), `templates/` (arquivos HTML), `forms.py` (formularios) e `urls.py` (rotas específicas).

Interação com a Base de Dados: A aplicação utiliza o ORM (Object-Relational Mapping) do Django para abstrair o acesso ao banco de dados. Os modelos definidos em `models.py` de cada app representam as tabelas no banco de dados relacional. O ORM permite consultas, inserções, atualizações e deleções através de métodos Python, sem necessidade de escrever SQL manualmente.

Lógica de Autenticação: A autenticação dos usuários é gerida pelo módulo `accounts`, utilizando os mecanismos nativos do Django, como `django.contrib.auth`. Isso inclui a criação e verificação de credenciais, controle de sessões, além de permissões baseadas em grupos e perfis para diferenciar usuários comuns de administradores. A aplicação implementa formulários de login, logout e registro, além de proteção de rotas para garantir o acesso controlado às funcionalidades.

3.2 Design da Base de Dados

O modelo de dados da aplicação é organizado em diversas tabelas que representam as entidades principais do sistema e seus relacionamentos, conforme descrito abaixo:

Tabelas Principais:

- **Usuário (`User`):** Representa os clientes e administradores da plataforma, com campos para informações pessoais, credenciais de acesso e permissões.
- **Produto (`Product`):** Contém informações sobre os produtos disponíveis, incluindo nome, descrição, preço, quantidade em estoque, e referência para as imagens armazenadas.
- **Categoria (`Category`):** Estrutura para classificação dos produtos em categorias, permitindo navegação e filtragem.
- **Carrinho (`Cart` e `CartItem`):** Representa o carrinho de compras temporário dos usuários. A tabela `Cart` identifica o carrinho vinculado ao usuário, enquanto `CartItem` registra os produtos adicionados com suas respectivas quantidades.
- **Pedido (`Order`):** Armazena informações dos pedidos realizados, como data, status, valor total e vínculo ao usuário.
- **Item do Pedido (`OrderItem`):** Registra os produtos e quantidades específicos associados a cada pedido.

Relacionamentos:

- Um **usuário** pode possuir vários **pedidos** (relação um-para-muitos).
- Um **pedido** contém vários **itens de pedido** (relação um-para-muitos).
- Um **produto** pode pertencer a uma **categoria** (relação muitos-para-um).
- Um **carrinho** está associado a um único **usuário**, e contém vários **itens de carrinho**.

3.3 Funcionalidades Implementadas

Durante o desenvolvimento da plataforma, foram implementadas funcionalidades essenciais, focadas na experiência do utilizador e gestão administrativa. O gerenciamento de usuários, via módulo accounts, integra os mecanismos de autenticação nativos do Django (django.contrib.auth), com formulários personalizados para registo, login, edição de perfil e recuperação de senha, garantindo acesso seguro e permissões diferenciadas.

O catálogo de produtos, no módulo products, permite criação, edição e exclusão de itens e categorias, com dados armazenados e exibidos por templates responsivos e integração ao diretório media para upload de imagens. Essa funcionalidade é central, pois é o principal ponto de interação do utilizador para busca e seleção de produtos.

O carrinho de compras, no módulo cart, usa sessões para manter os itens adicionados, permitindo modificações antes da finalização. O módulo orders consolida dados do carrinho e utilizador para registrar e acompanhar pedidos, assegurando transparência e confiança no processo.

Para a gestão, há ferramentas administrativas para controle de produtos, pedidos e usuários, protegidas por sistema de permissões que limita alterações a administradores autorizados. Também foi integrado o gerenciamento de mídia, permitindo upload e exibição de imagens, melhorando a atratividade e a experiência do utilizador.

3.4 Interface de Utilizador

A interface do sistema foi concebida com ênfase na usabilidade, simplicidade e acessibilidade, proporcionando uma navegação intuitiva tanto para clientes como para administradores. A aplicação utiliza templates em HTML e CSS, com apoio de frameworks como Bootstrap, permitindo que o design se adapte a diferentes dispositivos, como computadores, tablets e smartphones, o que garante uma boa experiência em contextos variados.

A navegação é clara e direta, com menus bem definidos para as principais áreas do sistema, como catálogo, carrinho, perfil e administração.

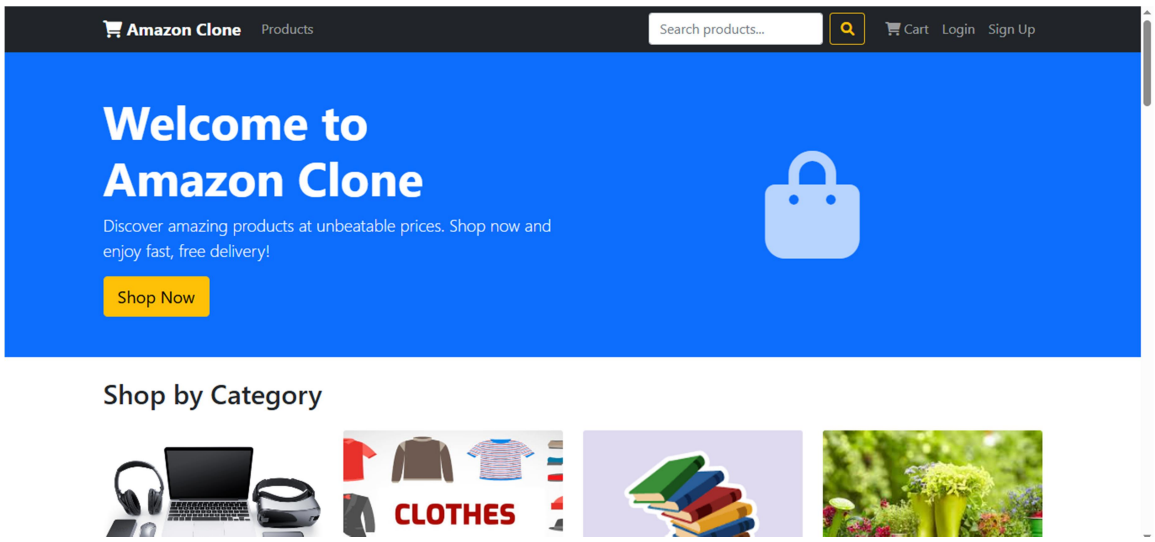

Ilustração 1- Página home do site

4. Testes e Validação

4.1 Testes Funcionais

Para assegurar o correto funcionamento da aplicação, foram realizados testes funcionais manuais ao longo do desenvolvimento, utilizando o ambiente local do Django. Os testes abrangeram as principais funcionalidades, como cadastro, login e recuperação de senha,

garantindo um acesso seguro. Também foram verificados o correto funcionamento do catálogo, com exibição e edição de produtos, e do carrinho de compras, validando a adição, remoção e persistência de itens durante a navegação.

4.2 Testes de Usabilidade

Para avaliar a usabilidade da aplicação, foram realizados testes que demonstraram facilidade em localizar produtos, adicionar itens ao carrinho e concluir pedidos, indicando que a interface é bem organizada e intuitiva.

4.3 Testes de Segurança

O sistema foi submetido a testes de segurança focados na proteção de dados e controle de acesso. As senhas foram armazenadas com criptografia segura (hashing), impedindo o uso de texto plano. Também foram verificadas as permissões de acesso, garantindo que apenas usuários autorizados pudessem aceder a áreas restritas.

5. Desafios e Soluções

5.1 Desafios Técnicos

Durante o desenvolvimento da aplicação, surgiram diversos desafios. Um dos principais foi garantir que o carrinho de compras mantivesse seu estado ao longo da navegação, especialmente para usuários não autenticados, o que exigiu um bom controle de sessões.

5.2 Soluções Encontradas

Para resolver os desafios enfrentados, utilizaram-se sessões do Django para armazenar os dados do carrinho e manter o estado mesmo após o login.

6. Resultados

6.1 Funcionalidade Completa

Após a finalização do desenvolvimento, todas as funcionalidades propostas foram testadas e confirmadas. O sistema permite cadastro e autenticação segura de usuários, exibe um catálogo de produtos bem estruturado com categorias e imagens, e possibilita a gestão eficiente do carrinho de compras. A área administrativa permite o controle de produtos, pedidos e utilizadores com segurança, e a interface responde bem em diferentes dispositivos, mantendo boa usabilidade.

6.2 Usabilidade e Design

A estrutura clara, o design responsivo e a facilidade em localizar produtos e concluir compras contribuíram para uma navegação eficiente e agradável.

6.3 Desempenho do Sistema

Em ambiente de testes local, a aplicação apresentou desempenho consistente, com tempo de resposta adequado nas operações principais. O ORM do Django garantiu boa performance nas consultas ao banco de dados, e o uso de cache e otimização de imagens reduziu os tempos de carregamento. Ainda assim, é recomendável realizar testes de carga em ambiente de produção para validar a escalabilidade sob maior demanda.

7. Conclusão

7.1 Conclusões Finais

O projeto alcançou os objetivos propostos inicialmente, entregando uma plataforma funcional de comércio eletrônico que permite o gerenciamento completo do catálogo, processo de compra e administração do sistema. A aplicação demonstrou ser uma solução viável para

pequenos e médios negócios que desejem disponibilizar seus produtos online de forma eficiente e segura.

7.2 Lições Aprendidas

Durante o desenvolvimento, foi possível aprofundar conhecimentos técnicos em Django, modelagem de dados, desenvolvimento web responsivo e segurança da informação.

7.3 Melhorias Futuras

Para aprimorar ainda mais o sistema, sugerem-se as seguintes melhorias:

- Inclusão de filtros avançados no catálogo, como pesquisa por preço, avaliações e disponibilidade.
- Integração com gateways de pagamento para processar transações financeiras diretamente na plataforma.
- Automatização dos testes funcionais e de usabilidade para garantir a manutenção da qualidade em futuras atualizações.
- Desenvolver um sistema de pagamento robusto.
- Desenvolver um algoritmo de conversão de pagamentos.

Essas melhorias podem ampliar significativamente a funcionalidade e a competitividade do sistema no mercado.

Bibliografia

1. **Django Software Foundation.** Django Documentation. *Django.* [Online] [Citação: 30 de 06 de 2025.] https://docs.djangoproject.com/.

2. ChatGPT. *ChatGPT.* [Online] [Citação: 30 de 06 de 2025.] https://chatgpt.com/c/682c8777-5424-800f-8e4f-0c9729af0a1a.