ISCTE-IUL
Licenciatura em Ciência de Dados

Projeto Django: PCK

Exercício realizado no âmbito da Unidade Curricular de Interfaces Web Para a Gestão de Dados do 3º ano da Licenciatura em Ciência de Dados

André Plancha, 105289
Andre_Plancha@iscte-iul.pt

Allan Kardec, 103380
aksrs@iscte-iul.pt

Rui Chaves, 104914
rfcs1@iscte-iul.pt

30 de outubro 2023
Versão 1.0.0

Descrição do projeto

Neste projeto, foi modelado e programado um website usando Django, consistindo num fórum dedicado para artistas e *performers* que queiram partilhar a sua dedicação à arte, partilhando conhecimentos, esperiências, demonstrações, e muito mais; Chamado de **PCK**. O website está devidido em tipos de artes diferentes, e os utilizadores podem criar uma conta para participarem no fórum.

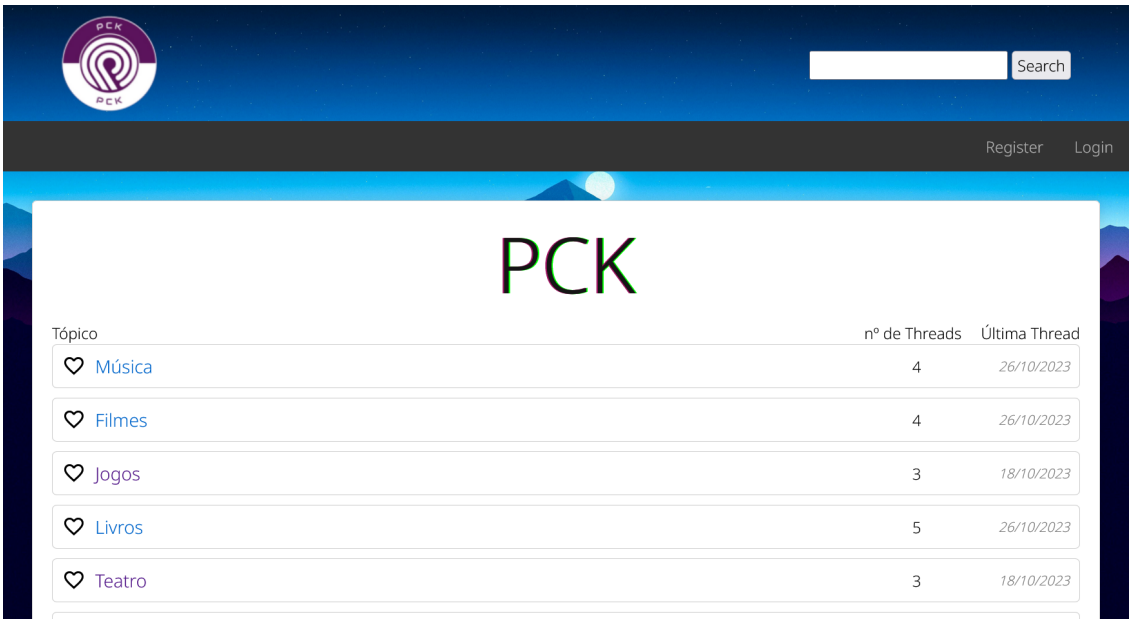

Figura 1: Página principal

Qualquer utilizador pode criar um novo Thread em qualquer Tópico, acompanhado de um Post, e qualquer utilizador pode vê-lo assim que publicado na lista de Threads disponivel dentro do Tópico. Um Thread é apenas composto por um título e um Post acompanhado, e qualquer utilizador por adicionar Posts, de forma a completá-lo. Isto pode ser em formato de pergunta resposta, solicitação e atendimento, demostração e feedback, criticas e contra criticas, etc.

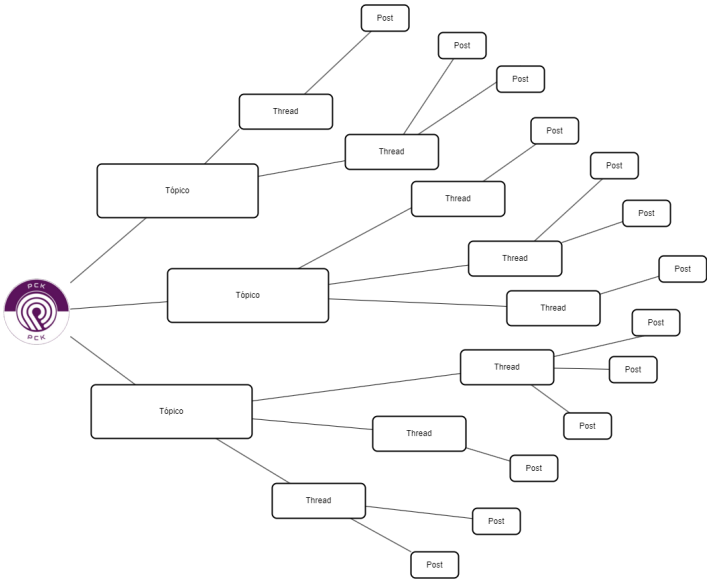

Figura 2: Formato do website

Principais funcionalidades

As funcionalidades tem as seguintes funcionalidades notáveis (sem orgem de importância):

1. O utilizador consegue navegar e pesquisar tópicos, threads e posts sem ter uma conta;

2. Ao registar, o utilizador pode imediatamente começar a criar threads e postar o que quiser em qualquer tópico;
3. Cada conta tem acesso a uma imagem de perfil (à escolha do utilizador), um username, e uma assinatura. Todas estas estão públicas e visíveis cada vez que o utilizador posta alguma coisa em qualquer tópico;
4. Cada utilizador pode mudar qualquer uma destas e a sua password assim que desejar;
5. Cada utilizador por "favoritar" tópicos, para que eles apareçam no top da página principal, para fácil navegamento;
6. Cada post pode ser reportado para um administrador;
7. Administradores podem participar também nas conversas sem revelar o seu status;
8. Administradores têm o poder de remover posts e threads, banir temporáriamente e permanentemente utilizadores, destacar (e tirar destaques de) threads e bloquear (e desbloquear) threads: destacar threads faz com que eles apareçam primeiro quando o utilizador está a navegar cada tópico, independente da data de publicação, e bloquear threads faz com que utilizadores não possam adicionar comentários ao thread, embora o seu conteudo ainda esteja acessível e interagível;
9. Cada utilizador pode remover o seu próprio post por qualquer motivo;
10. Administradores conseguem aceder uma página adicional (painel de administrador) onde podem visualizar reports que as pessoas fazem, e agir de acordo como acharem melhor acharem, tudo a partir do painel diretamente;
11. Cada utilizador pode reagir (e tirar reação) a um post sem queriarem um novo, usando reações em forma de *emojis* disponibilizados;
12. Cada post pode usufruir de diferentes estilos de texto, como *itálico*, **negrito**, links, `código`, ~~tachado~~, • pontos, "blocos de citação", e imagens externas.
13. Cada utilizador pode pesquisar por qualquer post.

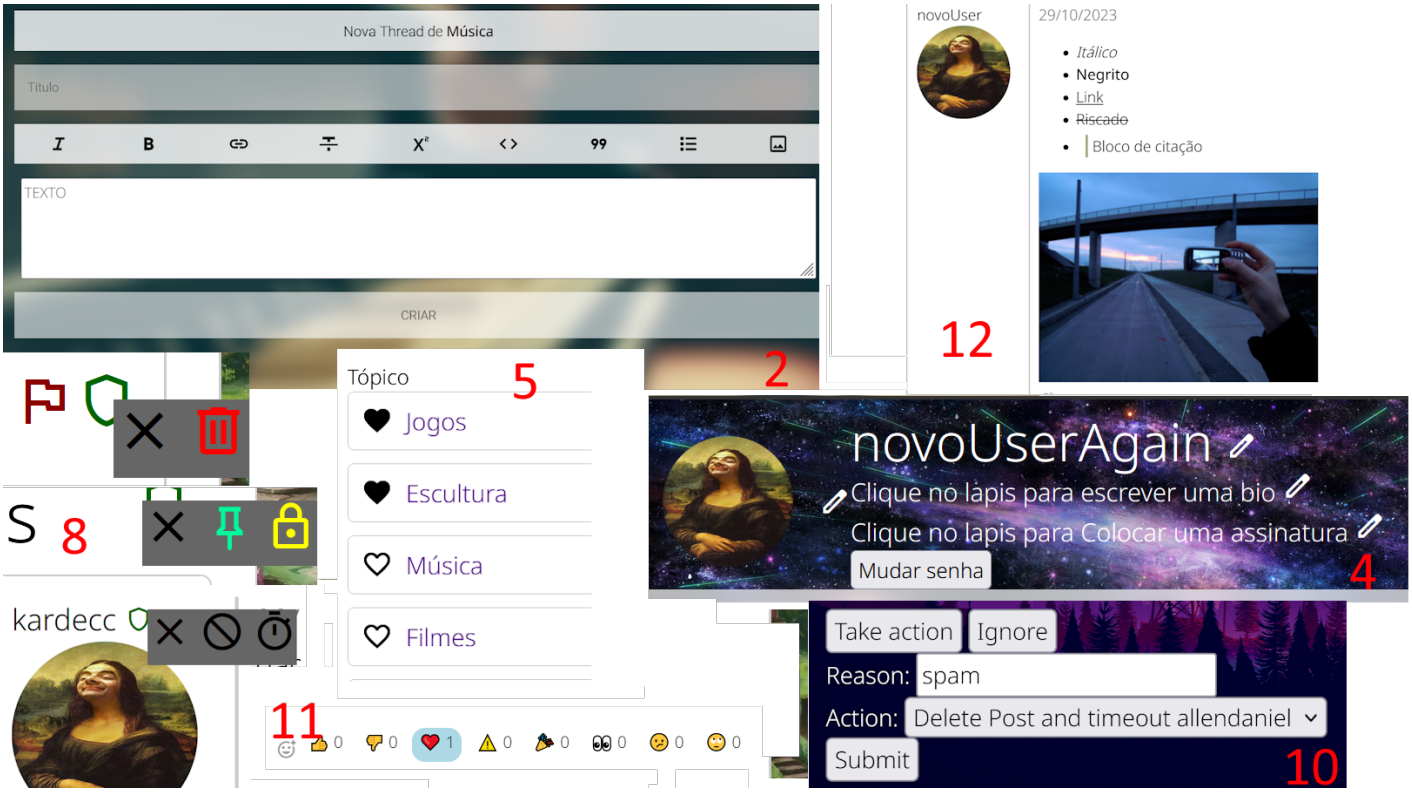

Figura 3: Principais funções

Decisões visuais

1. A página principal aprensenta o nº de Threads e a data da última thread, para um utilizador entender logo o tráfego de cada tópico;
2. Cada tópico tem uma imagem de fundo diferente relacionada com o tópico, para dar a ideia da diferença entre os tópicos;
3. O tópico "Meta" seria dedicado a tópicos sobre a própria funcionalidade do website, para os utilizadores poderem dar feedback sobre o funcionamento do PCK;
4. Administradores conseguem visualizar posts removidos para melhorar comunicação entre eles;
5. Todos os utilizadores têm acesso a uma "assinatura", servido de algo que é sempre acompanhado todos os posts como se fosse parte dela;
6. A assinatura de um utilizador é dinâmica em vêz de ser estática. Isto pode ser útil em alguns casos que um utilizador queira informar em posts antigos sobre alguma coisa nova;
7. Decidimos implementar pesquisa de threads, tópicos, e users, tudo com uma pesquisa apenas. Isto facilita a acessibilidade do website.

Decisões técnicas

Frontend

1. Para facilitar o desenvolvimento do projeto, decidimos usar um `base.html` (com ajuda da tag `block` do Django), onde vai estar `html` partilhado entre as várias páginas, como a importação de alguns ficheiros de estilo `css` e ficheiros de *scripts* externos
2. Para facilitar a execução de *scripts*, usámos jQuery[1], uma biblioteca de JavaScript para manipulação de elementos. Alguns plugins foram usados também em diversos sitios, sendo esses:
    - jQuery Modal de Kyle Fox[2] para criar popups interativos, usado para mudar as credenciais no perfil;
    - Um *fork* de jQuery.selection do madapaja e franck-eyraud[3], para facilitar inserção de estilos quando estamos a escrever posts.
3. Para fazer os estilos, decidimos usar Marked[4], uma biblioteca de js que analisa e compila MarkDown para HTML, deixando-nos a liberdade para escrever estilos diferentes. Usámos em conjunto de DOMPurify de Cure53[5], para ter a certeza que não haja problemas de segurança com inserção de MarkDown;
4. Para os backgrounds, usámos wallpapers de wallup.net. Para os efeitos de `glicthes`, usámos uma *demo* de Joan Leon[6]. Para os vários icones, usámos os icones disponibilizados pela Google[7]. Usámos também as fontes Noto Sans[8] e Roboto[9]. Finalmente, usámos um CSS Reset de Andy Bell[10].
5. Muitas das interações feitas durante o site foram feitas usando *Asynchronous JavaScript and XML* (ajax), (com a ajuda de jQuery), para poder disponibilizar ao utilizador várias funcionalidades sem necessidade de rerenderizar o website. Um exemplo óbvio do seu uso é nas reações, onde o utilizador pode reagir várias vezes sem ser interrompido.

---

[1] https://jQuery.com/
[2] https://www.jQuerymodal.com/
[3] https://github.com/evan-dickinson/jQuery.selection
[4] https://github.com/markedjs/marked
[5] https://github.com/cure53/DOMPurify/
[6] https://codepen.io/nucliweb/pen/QEreae
[7] https://fonts.google.com/icons

Backend

A nossa base de dados está feita da seguinte forma:

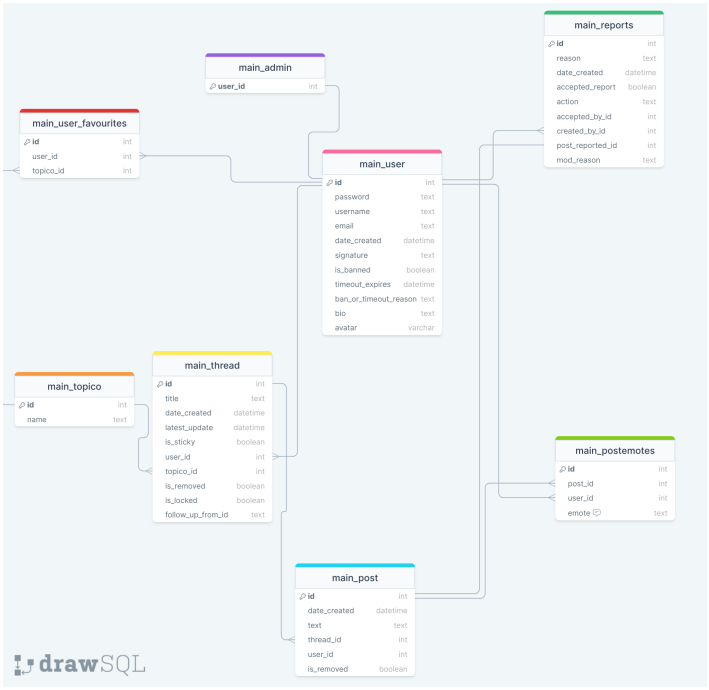

Figura 4: Esquema dos modelos

Nota-se que nós fizemos várias fizemos várias verificações de integridade nas nossas funções de que forma a que apenas informações desejadas sejam inseridas, de forma a que sejam diminuidas ações não desejadas ou previstas por parte dos utilizadores.

Pesquisa

Para pesquisa, nós usámos *Haystack* de Daniel Lindsley[11], uma biblioteca que serve para ligar uma biblioteca de pesquisa com o Django. Nós usámos Whoosh de Matt Chaput[12], uma implementação de pesquisa puramente em python, que facilita a sua implementação neste caso. Da forma que nós implementámos, assim que que um novo post, user, ou thread é adicionado a base de dados, novos documentos necessário para o funcionamento do whoosh são criados logo. Isto podia ser um problema se o website tivesse um grande volume de dados, mas para o nosso propósito é perfeito. O Whoosh usa *full-text indexing*, e o Haystack supporta a biblioteca, portanto a integração destas duas foi feita muito facilmente através da documentação oficial.

Desenvolvimento

Para uma facilidade em alteração futura, e para separar as funções dos nossos ficheiros, *quase* todas as funções que tocam na base de dados encontram-se no nosso `models.py`, para se houver alguma mudança na base de dados que requer mudanças nas consultas, as nossas `views.py` não precisam de mudar tanto.

Para facilitar o desenvolvimento, usámos `django-browser-reload`, de Adam Johnson, para quando um ficheiro no nosso projeto alterá-se, o browser automaticamente fazer refresh.

[8]https://fonts.google.com/noto/specimen/Noto+Sans
[9]https://fonts.google.com/specimen/Roboto
[10]https://andy-bell.co.uk/a-more-modern-css-reset/
[11]https://github.com/django-haystack/django-haystack
[12]https://github.com/mchaput/whoosh/