Teste Prático - Desenvolvedor Django

Parte 1: Modelagem de Banco de Dados

1. Crie um modelo `Post` que tenha os seguintes campos:
    - Título (CharField)
    - Conteúdo (TextField)
    - Data de publicação (DateTimeField)
    - Autor (ForeignKey para o modelo User do Django)
2. Crie um modelo `Comment` com os seguintes campos:
    - Autor (ForeignKey para o modelo User do Django)
    - Conteúdo (TextField)
    - Data de criação (DateTimeField)
    - Relacione o modelo `Comment` com o modelo `Post` de modo que um post possa ter vários comentários.

Parte 2: Views, Templates e API

3. Crie uma view que liste todos os posts ordenados pela data de publicação. A view deve renderizar um template que exiba o título, o autor e a data de publicação de cada post, incluindo a lista de comentários associados.
4. Crie uma view que mostre os detalhes de um post específico, incluindo a lista de comentários associados.
5. Crie um template base que inclua um bloco para o conteúdo principal e um bloco para o título da página. Os templates das views devem herdar deste template base.
6. Crie um formulário para adicionar um novo post, incluindo campos para o título, conteúdo e autor.
7. Implemente operações CRUD (Create, Read, Update, Delete) para os modelos `Post` e `Comment` através de uma API RESTful utilizando Django Rest Framework.

Parte 3: Autenticação, Autorização e Segurança

8. Implemente autenticação para garantir que apenas usuários autenticados possam criar, atualizar ou excluir posts e comentários.
9. Crie uma política de autorização personalizada para garantir que um usuário só possa editar ou excluir seu próprio post ou comentário.
10. Lidar com exceções e erros de forma adequada nas views e API.
11. Implemente práticas de segurança em todo o código.

Parte 4: Cache, Otimização e Testes Avançados

12. Implemente caching para a lista de posts utilizando o cache do Django para melhorar o desempenho da view.
13. Otimize consultas usando `select_related` e `prefetch_related` conforme necessário para evitar problemas de N+1.
14. Escreva testes para todas as funcionalidades, incluindo testes para a API, autenticação, autorização e otimizações.
15. Utilize mocks e fixtures para simular diferentes cenários e estados nos testes.

Observações

- Utilize o Django 3.x ou versão mais recente.
- Certifique-se de que o código esteja bem organizado, seguindo as melhores práticas de desenvolvimento Django.
- Inclua comentários no código sempre que necessário para explicar escolhas ou decisões.
- Fornecer um arquivo README explicando como executar o projeto localmente e como os testes podem ser executados.