Relatório Técnico - MVP de Blog

Contexto do Projeto

O projeto é um MVP (Produto Mínimo Viável) de um blog, desenvolvido para um processo seletivo. O sistema permite a criação e visualização de posts, com funcionalidades diferenciadas para usuários logados e deslogados. As principais tecnologias utilizadas foram:

- PHP Laravel: Framework para desenvolvimento do backend.
- Slim: Micro-framework para construção da API.
- Bootstrap: Framework CSS para estilização e design responsivo.
- MySQL (phpMyAdmin): Banco de dados relacional.
- JavaScript (JS) e CSS: Linguagens para interação e estilização adicionais.

Arquitetura do Banco de Dados

1. Estrutura do Banco de Dados:

- Banco de Dados: blog_ps

Tabelas:

- tb_posts: Armazena os posts do blog.
    - Campos:
        - id_post (int): Identificador único do post.
        - nm_post (varchar(150)): Descrição do post.
        - img_post (longblob): Imagem do post, armazenada em formato binário.
        - dt_post (date): Data do post.
        - cd_usuario (int): Identificador do usuário que criou o post.
    - Índices:
        - Chave primária em id_post.
        - Chave estrangeira FK_post_usuario em cd_usuario, referenciando id_usuario na tabela tb_usuario.
- tb_usuario: Armazena informações dos usuários.
    - Campos:
        - id_usuario (int): Identificador único do usuário.

        - nm_usuario (varchar(40)): Nome do usuário.
        - nm_email (varchar(30)): Email do usuário.
        - cd_senha (varchar(10)): Senha do usuário.
        - img_perfil (varchar(200)): URL da imagem de perfil do usuário.
    - Índices:
        - Chave primária em id_usuario.

Dados de Exemplo:

- tb_posts:
    - (5, 'descricao', 0x696d6167656d, '2024-08-21', 1)
    - (6, 'sdifygiwgyeffshuifdihuofsdohuji', 0x7465737465, '2024-08-13', 2)
    - (7, 'bom dia gente!', '', '2024-08-23', 11)
- tb_usuario:
    - (1, 'Arthur', 'artu@email.com', '123123', '')
    - (2, 'isabelle', 'isabelle@email', '12345', 'vazia')
    - (11, 'Lucas', 'lucas@email.com', '123456', '')

Organização do Código

- database/: Contém o arquivo SQL de dump do banco de dados e o print das tabelas.
- api_blog/: Diretórios e funcionalidades do Slim Framework para a construção da API.
- blog/: Diretórios e funcionalidades do Laravel para o desenvolvimento do site.
- readme.md: Descrição do projeto para o GitHub.

Resiliência

- Tratamento de Erros: Implementação de tratamento de erros para chamadas à API e operações de banco de dados. Mensagens de erro são geradas e exibidas quando necessário.
- Validação: Validação de dados antes do armazenamento e exibição para evitar erros e inconsistências.

Performance

- Boas Práticas de Banco de Dados:
    - Índices: Uso de índices nas tabelas para melhorar a velocidade das consultas.
    - Consultas Otimizadas: Consultas SQL são preparadas para evitar injeções de SQL e melhorar a performance.
- Cache: Considerações sobre uso de cache para melhorar a performance das respostas da API.

Segurança

- Proteção Contra Injeção de SQL: Uso de PDO para consultas SQL preparadas e parâmetros vinculados.
- Proteção de Imagens: Codificação base64 para armazenar imagens, evitando problemas de injeção.
- Segurança de Sessão: Implementação de controle de acesso baseado em sessão para funcionalidades de criação e visualização de posts.

Simultaneidade

- Escalabilidade: Preparação para escalar o sistema em caso de aumento significativo no número de requisições. Considerações sobre uso de técnicas de balanceamento de carga e otimização de consultas.
- Concorrência: Uso de práticas para garantir a integridade dos dados e minimizar problemas em cenários de alta concorrência.

Obs.: A funcionalidade base64 para exibição de imagens está um pouco bugada ao exibir .jpeg do banco de dados, então as imagens podem ser renderizadas mas não exibidas.