Desafio de Desenvolvimento Laravel - Vaga Júnior

Objetivo: Desenvolver uma aplicação web simples utilizando Laravel, focando nas boas práticas de desenvolvimento, segurança e organização de código.

Requisitos:

1. **Autenticação Completa:**
   - Sistema de **cadastro de novos usuários** (com validação de dados).
   - Sistema de **login** (com validação de credenciais).
   - Redirecionamento apropriado após login/cadastro.
2. **Gerenciamento de Usuários (Apenas para Usuários Autenticados):**
   - **Cadastro de Usuário:** Permitir que um usuário autenticado cadastre novos usuários no sistema.
   - **Listagem de Usuários:** Exibir uma lista paginada de todos os usuários cadastrados.
     - **Filtro:** A listagem de usuários deve permitir um **filtro por nome e/ou e-mail**.
   - **Edição de Usuário:** Permitir a edição de informações de um usuário existente (ex: nome, email).
   - **Deleção de Usuário:** Permitir a exclusão de um usuário.

Requisitos Técnicos Obrigatórios:

1. **Laravel Framework:** Utilizar a versão 10 do Laravel.
2. **Validação de Requisições (Form Request):** Todas as validações de entrada de dados (cadastro, login, gerenciamento de usuários) devem ser implementadas utilizando `Form Request`.
3. **Envio de E-mail:**
   - Ao cadastrar um novo usuário (tanto pelo próprio usuário quanto por um administrador), deve ser disparado um e-mail de boas-vindas para o e-mail cadastrado.
   - A mensagem de boas-vindas deve ser simples e informativa.
   - Configurar um driver de e-mail local (ex: Mailtrap, MailHog) para testes.
4. **Frontend (Bootstrap):** A interface da aplicação deve ser construída utilizando o framework CSS **Bootstrap** para um layout responsivo e agradável.
5. **Princípio DRY (Don't Repeat Yourself):** O código deve seguir o princípio DRY, evitando duplicação de lógica e código. Utilizar recursos como `Blade Components`, `Traits`, etc., quando apropriado.
6. **Eloquent ORM:** Toda a interação com o banco de dados deve ser feita através do Eloquent ORM. Não é permitido o uso de `DB::raw()` ou `Query Builder` para operações CRUD principais.

7. **Performance:** A aplicação deve ser desenvolvida com a performance em mente. Considerar:
    - Uso eficiente de queries Eloquent (evitar N+1 problem).
    - Otimização de assets (se necessário).
    - Configuração do cache do Laravel (se aplicável para otimização de rotas/config).
8. **Layout Avaliado:** O design geral e a usabilidade da interface serão avaliados. O layout deve ser limpo, intuitivo e responsivo.

Diferenciais (Será um bônus):

1. **Single Action Controllers:** Utilização de Single Action Controllers (`__invoke`) para as ações que se encaixam nesse padrão (ex: `StoreUserController`, `UpdateUserController`).
2. **Camada de Serviço (Service Layer):** Implementação de uma camada de serviço para encapsular a lógica de negócio complexa, separando-a dos controllers (ex: `UserService` para as operações de CRUD de usuário).
3. **Testes Automatizados:** Implementação de testes unitários ou de feature para as funcionalidades principais (ex: testes para cadastro, login, listagem de usuários).

Instruções para Entrega:

1. O código deve ser entregue em um repositório Git (GitHub, GitLab, Bitbucket).
2. O repositório deve conter um arquivo `README.md` com as seguintes informações:
    - Instruções detalhadas para rodar a aplicação localmente (passos para instalação, configuração do `.env`, migrações, etc.).
    - Uma breve descrição das decisões de arquitetura e design tomadas.
    - Quaisquer observações importantes sobre o projeto.
3. A aplicação deve ser funcional e livre de erros.
4. O **link do repositório público** deve ser enviado para o e-mail contato@dinamk.com.br com o assunto **Teste Vaga Junior**.

Critérios de Avaliação:

- **Funcionalidade:** Todas as funcionalidades obrigatórias devem estar implementadas e funcionando corretamente.
- **Boas Práticas Laravel:** Uso correto do Eloquent, Form Requests, Blade, Middlewares.
- **Segurança:** Prevenção de vulnerabilidades comuns (CSRF, SQL Injection, XSS).
- **Performance:** A aplicação deve apresentar um bom desempenho.
- **Design e Usabilidade:** Qualidade do layout e da experiência do usuário.
- **Aplicação de Diferenciais:** Pontos extras para a implementação de Single Action Controllers, Camada de Serviço e Testes.

Boa sorte!