

Interações medicamentosas descomplicadas



Componentes

- Caio Alberto Nunes Marques
- Carlos Daniel Freire Fernandes
- Josimara Silva de Lima
- Leandra Lauana Izidio Ferreira
- Silvio Martins Santos

Sumário da apresentação

- 1. Uma breve introdução ao sistema
- 2. Arquitetura do projeto
- 3. Práticas de desenvolvimento
- 4. Tecnologias utilizadas
- 5. Integração contínua e deploy (CI/CD)
- 6. Testes
- 7. Demo do projeto



- Introdução O que é o FarmacoCheck
 - Ferramenta para consulta de interações medicamentosas.
 - Desenvolvida para apoiar profissionais de saúde na tomada de decisões baseadas em dados.



- Objetivo O que buscamos com o FarmacoCheck?
 - Identificar interações medicamentosas prejudiciais.
 - Reduzir riscos associados a combinações de medicamentos.



- Por que usar o FarmacoCheck?
 - Base de Dados Estruturada
 - Permite consultar interações medicamentosas e contraindicações;
 - informações detalhadas sobre os medicamentos e interações.



- Por que usar o FarmacoCheck?
 - Sistema de Perfis
 - **Super Admin**: Controle total do sistema.
 - Admin: Atualização de dados (exceto usuários).
 - São profissionais da área da saúde tais como:
 - Farmacêuticos, Bioquímicos e Biomédicos.
 - Médicos (para a primeira versão): Consulta às interações.



- Por que usar o FarmacoCheck?
 - Cobertura de Medicamentos
 - Cobertura de medicamentos amplamente distribuídos pelo Sistema Único de Saúde.
 - Infraestrutura Tecnológica
 - Desenvolvido em PHP, Laravel e MySQL.



- Impacto desejado com a Aplicação:
 - Solução para desafios da polifarmácia.
 - Auxílio na tomada de decisões clínicas.
 - Contribuição direta para a segurança do paciente e no sistema de saúde.



Arquitetura do projeto

Model-View-Controller (MVC);

MVC é um padrão de arquitetura/design que separa uma aplicação em 3 componentes lógicos:

- Model;
- View;
- Controller.

Model

Model é a camada que possui a lógica da aplicação, sendo responsável pelas regras de negócios, pela persistência no banco de dados e pelas classes de entidades.

Ele processa as requisições recebidas do **Controller** e retorna as respostas correspondentes.

View

Camada de visualização, responsável pela interação entre usuário e sistema.

A View não contém lógica de negócios; todo o processamento ocorre na camada **Model**, e as respostas são enviadas à View pelo **Controller**.

Controller

Controller é a camada intermediária que organiza e gerencia a comunicação entre o Model e a View.

Ele processa as requisições da View, chama os métodos do Model e repassa os resultados para a View.

Vantagens

- Separação clara entre:
 - lógica de negócios;
 - lógica de interface do usuário;
 - lógica de entrada;
- Desenvolvimento mais rápido;
- Promove manutenabilidade da aplicação;
- Facilita testes independentes.



Práticas de desenvolvimento

- Modelo incremental
 - Entrega de pequenos incrementos, cada um adicionando funcionalidade ao sistema.
- Objetivo: Buscar redução do tempo para entrega de partes funcionais do sistema.



Consultas

Medicamentos

Interações

Usuários

Algumas tecnologias utilizadas

- MySQL;
- PHP e TallStack;
- DigitalOcean;
- Sonar cloud;
- Postman.











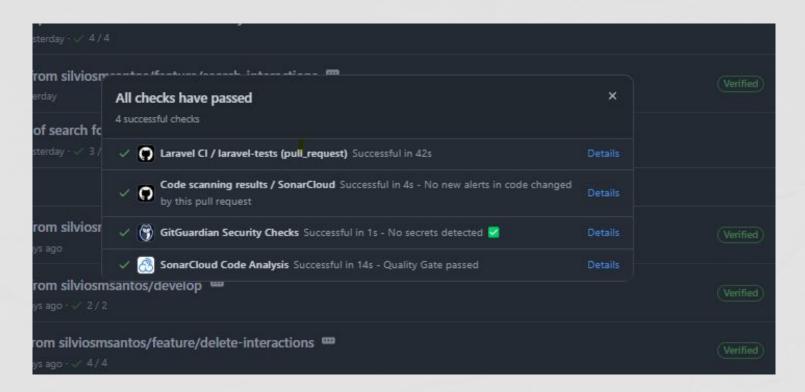


sonarcloud &



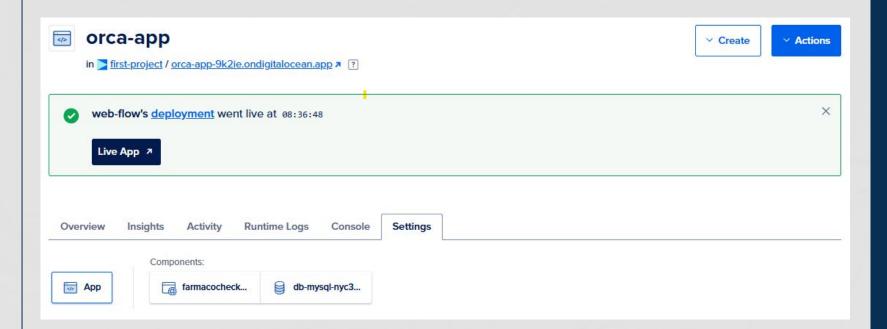


Integração das tecnologias





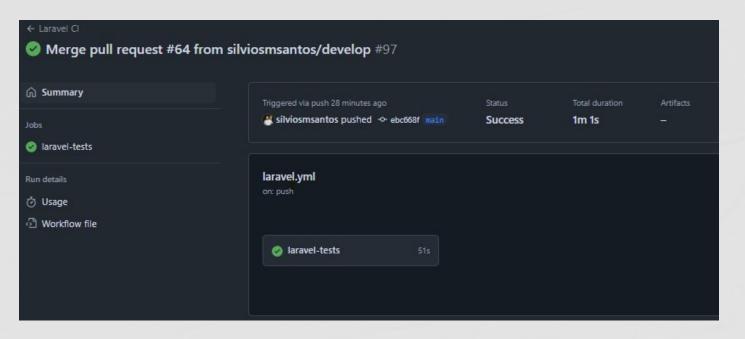
DigitalOcean servidor de hospedagem e deploy





Integração contínua e deploy

- GitHub Actions para automação de testes e deploy contínuo.
 - Pipeline de CI/CD configurado.



Testes

Testes Automatizados

- Testes de unidade para verificar a lógica interna das classes.
- Testes de integração e sistema parte do funcionamento dos componentes integrados.

Outros testes

- **♦** K6;
- Testes funcionais exploratórios.





Demo

Agora, vamos à demonstração do projeto!



Obrigado!



