



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





# Uma breve introdução ao sistema

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



# Uma breve introdução ao sistema

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



# Uma breve introdução ao sistema

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



# Uma breve introdução ao sistema

- Por que usar o FamacoCheck?
  - 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.



# Uma breve introdução ao sistema

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





# Uma breve introdução ao sistema

- **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 (JOSIMARA)

## ➤ 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.





# Integração das tecnologias


The screenshot displays a GitHub pull request interface. A modal window is open, titled "All checks have passed", indicating that four successful checks were performed. The checks listed are:


- Laravel CI / laravel-tests (pull\_request)**: Successful in 42s. A "Details" link is provided.
- Code scanning results / SonarCloud**: Successful in 4s - No new alerts in code changed by this pull request. A "Details" link is provided.
- GitGuardian Security Checks**: Successful in 1s - No secrets detected. A "Details" link and a "Verified" badge are present.
- SonarCloud Code Analysis**: Successful in 14s - Quality Gate passed. A "Details" link is provided.

The background shows a pull request from "silviosmsantos/feature/search-interactions" to "develop". Other pull requests are visible in the list below, including one for "delete-interactions".






# DigitalOcean servidor de hospedagem e deploy

 **orca-app**

[first-project](#) / [orca-app-9k2ie.ondigitalocean.app](#) 


[Create](#) [Actions](#)


 web-flow's [deployment](#) went live at 08:36:48 


[Live App](#) 

[Overview](#) [Insights](#) [Activity](#) [Runtime Logs](#) [Console](#) **[Settings](#)**

Components:

 App

 farmacocheck...

 db-mysql-nyc3...





# Integração contínua e deploy

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

← Laravel CI

✓ Merge pull request #64 from silviosmsantos/develop #97

Summary

Jobs

- ✓ laravel-tests

Run details

- Usage
- Workflow file

Triggered via push 28 minutes ago

Status: Success

Total duration: 1m 1s

Artifacts: —

silviosmsantos pushed → ebc668f main

laravel.yml

on: push

✓ laravel-tests 51s

# 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!

