UNIVERSIDADE DO OESTE DE SANTA CATARINA UNOESC – CAMPUS SÃO MIGUEL DO OESTE CURSO DE CIÊNCIA DA COMPUTAÇÃO

FERNANDO CAMILO SCHNEIDER

JEAN CARLO TORAL

JOÃO ULISSES PORTO ALEGRE CIRIACO TEIXEIRA

TASK MANAGER

São Miguel do Oeste - SC

2025

UNIVERSIDADE DO OESTE DE SANTA CATARINA UNOESC – CAMPUS SÃO MIGUEL DO OESTE CURSO DE CIÊNCIA DA COMPUTAÇÃO

FERNANDO CAMILO SCHNEIDER

JEAN CARLO TORAL

JOÃO ULISSES PORTO ALEGRE CIRIACO TEIXEIRA

TASK MANAGER

Trabalho apresentado ao Componente Curricular de Programação III do curso de Ciência da Computação da Universidade do Oeste de Santa Catarina – Unoesc.

São Miguel do Oeste - SC

2025

RESUMO

Este artigo detalha o desenvolvimento de uma aplicação de lista de tarefas, motivada pela necessidade de organizar ideias de forma independente e ágil. O projeto explora a aplicação de frameworks modernos, utilizando o CodeIgniter por sua leveza e agilidade, e seguindo o padrão arquitetural MVC (Model-View-Controller) para estruturação do código. A interação com o banco de dados PostgreSQL foi simplificada pelo Mapeamento Objeto-Relacional (ORM), enquanto o uso de Docker e Docker Compose garantiu um ambiente de desenvolvimento consistente e portátil. O processo de desenvolvimento abordou desde a modelagem do banco de dados e a implementação das funcionalidades (cadastro e login de usuários, registro, visualização, exclusão e marcação de status de tarefas) até a configuração do ambiente e a execução de testes com PHPUnit. Desafios como a inicialização do projeto, problemas de conexão com o banco de dados, estruturação de arquivos, execução de queries e questões de cache foram enfrentados e superados, enriquecendo o aprendizado. Em suma, o trabalho demonstra os benefícios da utilização de frameworks para a construção de software eficiente e de fácil manutenção.

Palavras-chave: Task Manager. CodeIgniter. MVC. Docker. Desenvolvimento Web.

SUMÁRIO

1 INTRODUÇÃO

No cenário atual de desenvolvimento web, a utilização de frameworks tornou-se indispensável para a criação de aplicações escaláveis, seguras e eficientes. Eles fornecem uma estrutura organizada, bibliotecas de código reutilizáveis e padrões de design que aceleram o processo de desenvolvimento e promovem a boa prática. Este artigo detalha uma aplicação de lista de tarefas, escolhida como um projeto prático para a organização de ideias de forma fácil, independente e rápida. Para isso, exploramos conceitos como o padrão arquitetural MVC (Model-View-Controller) para organizar o código, o Mapeamento Objeto-Relacional (ORM) para simplificar a interação com o banco de dados, e a Containerização com Docker para garantir um ambiente de desenvolvimento consistente. Durante o processo de desenvolvimento, enfrentamos desafios significativos, como o entendimento aprofundado do framework e a aplicação correta das lógicas de programação, além da dificuldade em resolver problemas específicos, como as marcações de tarefas completas. Apesar dessas adversidades, este projeto prático demonstra como um framework pode ser empregado para construir uma ferramenta útil, seguindo as orientações para artigos científicos.

2 A APLICAÇÃO DE LISTA DE TAREFAS

A aplicação de lista de tarefas, desenvolvida como um projeto de estudo de frameworks, permite aos usuários gerenciar suas atividades diárias. As funcionalidades principais incluem:

- Cadastro de Usuários: Implementação de um módulo de registro que permite aos novos usuários criar credenciais de acesso, gerenciando a persistência de dados de autenticação e a integridade do banco de dados de usuários.
- Login de Usuários: Módulo de autenticação que valida as credenciais do usuário e estabelece uma sessão segura, garantindo o acesso autorizado às funcionalidades personalizadas e aos dados específicos do usuário.
- Registro de Tarefas: Funcionalidade para a persistência de novas entidades de tarefa no sistema, capturando dados como descrição e status inicial, e associando-as ao usuário autenticado.
- Visualização de Tarefas: Exibição dinâmica da coleção de tarefas persistidas no banco de dados, apresentando-as ao usuário de forma estruturada para facilitar o gerenciamento.
- Exclusão de Tarefas: Operação para a remoção permanente de registros de tarefas do banco de dados, mantendo a integridade referencial e o controle de acesso baseado no usuário.
- Marcação de Status: Funcionalidade para a atualização do estado de uma tarefa (e.g., de "pendente" para "concluída"), implicando a modificação de atributos específicos no registro persistido.

Esta aplicação serve como um excelente ponto de partida para entender como os frameworks facilitam a implementação de funcionalidades CRUD (Create, Read, Update, Delete) em um ambiente web.

3 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os conceitos teóricos que embasam o projeto, incluindo uma discussão aprofundada sobre a justificativa do framework, os conceitos de MVC, o Mapeamento Objeto-Relacional e a containerização.

Optou-se pelo CodeIgniter devido à sua natureza leve e rápida, ideal para projetos de médio porte que precisam ser desenvolvidos e implantados rapidamente. Ele oferece simplicidade e bom desempenho, sendo uma alternativa eficiente para quem busca agilidade sem a complexidade de frameworks maiores. Seu sistema ORM nativo é suficiente para as necessidades básicas de manipulação de dados da aplicação.

O MVC é um padrão de arquitetura que divide a aplicação em três partes para organizar o código: o Model cuida dos dados e da lógica de negócios, interagindo com o banco de dados (ex: uma classe Tarefa); a View é o que o usuário vê, ou seja, a interface gráfica (ex: arquivos HTML para exibir a lista de tarefas); e o Controller age como um intermediário, recebendo as requisições do usuário, processando-as e decidindo qual Visão mostrar.

No projeto, o ORM nativo do CodeIgniter foi fundamental para simplificar a interação com o banco de dados. Em vez de escrever comandos SQL complexos para cada operação, utilizaram-se os métodos do ORM para realizar as ações de criar, ler, atualizar e deletar tarefas. Isso agilizou o desenvolvimento e tornou o código de manipulação de dados mais limpo e fácil de manter.

Docker e Docker Compose foram essenciais para criar um ambiente de desenvolvimento padronizado e sem conflitos. Utilizou-se o Docker para empacotar a aplicação PHP, o servidor web e o banco de dados PostgreSQL em "contêineres" separados. O Docker Compose permitiu definir e gerenciar facilmente esses múltiplos contêineres juntos, garantindo que o ambiente de desenvolvimento fosse consistente e facilmente replicável para todos os membros da equipe.

4 ARQUITETURA E TECNOLOGIAS

O projeto baseou-se em uma arquitetura robusta, utilizando as seguintes tecnologias.

4.1 Framework PHP

O núcleo da aplicação foi desenvolvido utilizando o framework CodeIgniter, que serviu como a espinha dorsal do projeto. Ele ajudou a organizar o código, separando as preocupações em Modelos, Views e Controladores (MVC). Isso permitiu gerenciar o roteamento, o processamento de dados e a comunicação com

o banco de dados. A escolha do framework trouxe recursos de segurança e um padrão de trabalho que facilitou a criação e a manutenção do código.

4.2 Docker e Docker Compose

Para garantir que a aplicação funcionasse da mesma forma em qualquer computador, utilizou-se o Docker e o Docker Compose. O Docker permitiu "empacotar" a aplicação PHP junto com todas as suas dependências em um contêiner isolado. O Docker Compose foi usado para orquestrar e conectar diferentes partes do ambiente, como o contêiner da aplicação e o do banco de dados PostgreSQL, simplificando a configuração e o gerenciamento.

4.3 Banco de Dados PostgreSQL

O PostgreSQL foi escolhido como o banco de dados para armazenar todas as informações da aplicação, como os dados de usuários e as tarefas. Trata-se de um sistema de gerenciamento de banco de dados robusto e confiável, que lida bem com várias operações ao mesmo tempo e é compatível com os padrões SQL, sendo a escolha ideal para o projeto.

4.4 Ferramentas Adicionais

- Composer: Essencial para o projeto, o Composer foi utilizado para gerenciar as dependências do PHP, automatizando a instalação e a atualização de todas as bibliotecas de código necessárias para o funcionamento do framework.
- PHPUnit: Para garantir a qualidade e a confiabilidade da aplicação, utilizou-se o PHPUnit, uma ferramenta de testes automatizados que permitiu criar e executar testes para verificar se as diferentes partes do código estavam funcionando como esperado.

5 IMPLEMENTAÇÃO DO PROJETO

O processo de desenvolvimento da aplicação de lista de tarefas abrangeu desde a concepção do esquema de banco de dados até a execução do ambiente.

A modelagem do banco de dados envolveu a projeção de duas tabelas principais: `users` para armazenar dados de autenticação e `tasks` para as tarefas individuais, com chave estrangeira para o usuário. A gestão dessas estruturas foi realizada através do sistema de migrations do CodeIgniter.

Para a interação com o banco de dados, o sistema ORM do CodeIgniter permitiu manipular os dados como objetos PHP, evitando a escrita de consultas SQL diretas e simplificando as operações CRUD.

O projeto seguiu rigorosamente o padrão arquitetural MVC, organizando os arquivos em diretórios específicos (`app/Controllers`, `app/Models`, `app/Views`), o que contribuiu para a modularidade e a facilidade de manutenção.

A configuração do ambiente foi simplificada pelo uso de Docker. O processo incluiu a criação de um arquivo `.env` para variáveis de ambiente, a inicialização dos

contêineres com `docker-compose up -d`, a instalação de dependências com `composer install` e a execução das migrations com `php spark migrate`.

6 RESULTADOS

A aplicação final permite um fluxo de usuário completo e funcional. Após iniciar os contêineres Docker, o usuário pode acessar a aplicação via navegador e realizar as seguintes ações:

- Cadastro e Login: Um sistema de autenticação seguro permite que novos usuários se registrem e usuários existentes acessem suas contas.
- Gerenciamento de Tarefas: Uma vez logado, o usuário é direcionado para a página principal, onde pode adicionar novas tarefas. A lista exibe todas as tarefas cadastradas, permitindo sua exclusão por meio de um ícone de lixeira ou a marcação de seu status (concluída/pendente) através de um checkbox. As alterações são refletidas em tempo real na interface e persistidas no banco de dados.

7 REFLEXÃO SOBRE DESAFIOS

Durante o desenvolvimento, diversos desafios foram encontrados e superados:

- Inicialização do Projeto: Dificuldades iniciais na configuração do ambiente Docker.
- Conexão com o Banco de Dados: Falhas de comunicação entre a aplicação e o PostgreSQL.
- Estruturação de Arquivos: Problemas de localização e carregamento de arquivos pelo framework.
- Execução de Queries: Erros pontuais em operações de banco de dados, mesmo com o uso do ORM.
- Problemas de Cache: Comportamento inesperado da aplicação ao ser executada em ambientes fora do desenvolvimento principal.

A superação desses obstáculos não apenas resolveu questões técnicas, mas também aprofundou a compreensão sobre as ferramentas utilizadas e as boas práticas de desenvolvimento web.

8 LINK PARA O VÍDEO E REPOSITÓRIO NO GITHUB

Vídeo: https://youtu.be/SZGj055Ch3k
GitHub: https://github.com/fernandocschneider/prog3-projeto-frameworks

9 CONCLUSÃO

Este projeto demonstrou a construção de uma aplicação de lista de tarefas, um esforço prático para aplicar e compreender os benefícios dos frameworks

modernos. A escolha do CodeIgniter pela sua agilidade, aliada à organização do padrão MVC, otimizou o ciclo de desenvolvimento. A utilização do ORM simplificou a interação com o PostgreSQL, enquanto a containerização com Docker assegurou a consistência do ambiente. O sucesso na implementação de todas as funcionalidades propostas e a superação dos desafios técnicos reforçam a eficácia da abordagem baseada em frameworks. Conclui-se que o trabalho proporcionou um aprendizado valioso sobre os fluxos de desenvolvimento web, consolidando o conhecimento sobre a construção de software robusto e de fácil manutenção.

REFERÊNCIAS

CODEIGNITER. CodeIgniter User Guide. CodeIgniter Foundation, 2025. Disponível em: https://codeigniter.com/user_guide. Acesso em: 29 abr. 2025.

DOCKER. Docker Documentation. Docker, Inc., 2025. Disponível em: https://docs.docker.com/. Acesso em: 4 mai. 2025.

GOMES, I. Entendendo o Padrão MVC em Aplicações Web. Medium, 8 fev. 2021. Disponível em: https://medium.com/@ivan.n.gomes/entendendo-o-padr%C3%A3o-mvc-em-aplica%C3%A7%C3%B5es-web-a0e28f3a8b41. Acesso em: 4 mai. 2025.

MODEL-VIEW-CONTROLLER. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2023. Disponível em: https://pt.wikipedia.org/wiki/Model-View-Controller. Acesso em: 4 mai. 2025.