**EMPSYS: SISTEMA DE MONITORAMENTO DE MÁQUINAS**

João Vitor Pereira dos Santos (1), Maurilio Erik Batista Nóbrega de Souza Medeiros (2), Vinícius Heitor Siqueira Ribeiro (3). Orientador: Prof. Me. Ranieri Marinho. (1) 7-CCOMP-0034562, (2) 7-CCOMP-00343603, (3) 7-CCOMP-00333687.

**RESUMO**

O EMPSYS é uma aplicação web desenvolvida como parte do curso de Ciência da Computação. Seu objetivo é fornecer uma plataforma centralizada para o gerenciamento e monitoramento de empilhadeiras utilizadas por diferentes clientes em diversos locais. A aplicação é construída utilizando PHP 7.4, com o *framework Symfony* 4.4, e utiliza *SQLite* como banco de dados. A interface foi desenvolvida com *Bootstrap 4*, visando uma experiência amigável e acessível, especialmente para usuários com mais de 40 anos. O sistema é estruturado em torno de quatro entidades principais: Cliente, Local, Modelo Empilhadeira e Lista Empilhadeira, todas com funcionalidades completas de *CRUD*. Cada cliente pode possuir múltiplos locais e empilhadeiras, promovendo um controle eficiente do inventário. A interface prioriza uma navegação intuitiva, com boa acessibilidade e uso de cores suaves. O EMPSYS busca otimizar o controle de equipamentos industriais, oferecendo uma solução moderna, funcional e de fácil utilização para empresas que desejam monitorar suas máquinas com precisão e praticidade.

**Palavras-Chave:** Gerenciamento de empilhadeiras; *Symfony*; *Bootstrap*; Monitoramento de máquinas.

**1. Introdução**

O controle eficiente de equipamentos industriais é essencial para garantir a produtividade e a segurança em ambientes corporativos. Nesse contexto, o sistema EMPSYS: Sistema de Monitoramento de Máquinas é desenvolvido com o objetivo de fornecer uma solução web para o gerenciamento de empilhadeiras em empresas de diferentes portes [1].

A aplicação é construída utilizando a tecnologia PHP 7.4 em conjunto com o framework Symfony 4.4 [2], e faz uso do banco de dados SQLite para armazenar as informações das empilhadeiras, seus modelos, locais de operação e os clientes. A interface foi projetada com o framework Bootstrap 4, priorizando uma experiência de usuário acessível, especialmente para um público com mais de 40 anos, que muitas vezes encontra dificuldades em sistemas complexos. O sistema é estruturado em torno de quatro entidades principais: Cliente, Local, Modelo de Máquinas e Lista de Máquinas. Cada uma dessas entidades é implementada com operações de CRUD (Create, Read, Update, Delete), realizadas manualmente por meio de controllers e views próprias no Symfony. Essa escolha permite um maior controle sobre a lógica de negócios e personalização da interface de gestão, garantindo flexibilidade e adaptação às necessidades do usuário.

Além disso, o EMPSYS visa proporcionar uma plataforma intuitiva para o monitoramento de empilhadeiras, permitindo que os usuários possam registrar, editar e consultar informações sobre seus equipamentos, além de gerenciar locais de operação e gerar relatórios de forma simples e eficiente. A proposta do projeto é não apenas otimizar o gerenciamento de empilhadeiras, mas também fornecer uma interface clara e de fácil navegação, com um design focado em facilitar a leitura e a interação. Com isso, o EMPSYS se posiciona como uma solução prática e acessível para empresas que buscam maior controle sobre seu inventário de máquinas, contribuindo para a melhoria dos processos operacionais e a segurança no ambiente de trabalho.

Este documento é uma continuação do artigo do semestre anterior [5].

**2. Metodologia**

A metodologia utilizada no desenvolvimento do EMPSYS seguiu uma abordagem ágil [3], com foco na entrega incremental e adaptativa. Utilizou-se PHP 7.4 e Symfony 4.4. O banco SQLite foi escolhido por sua leveza e compatibilidade com protótipos. O Bootstrap 4 foi usado na interface, com foco em acessibilidade para usuários com mais de 40 anos. A estrutura padrão do Symfony foi mantida, com separação clara entre diretórios de entidades (src/Entity), controllers (src/Controller), templates (templates/) e rotas (config/routes). Essa organização favoreceu a manutenção e extensão do projeto. O Doctrine foi utilizado para o mapeamento objeto-relacional (ORM), com anotações nas entidades para definir relacionamentos, restrições e validações. Migrações foram geradas para versionamento do banco de dados e facilidade de replicação do ambiente.

A renderização de páginas foi feita com Twig, o mecanismo nativo do Symfony. Foram criados templates base (ex: base.html.twig) e templates filhos para reaproveitamento de estrutura visual. Componentes como navbar e rodapé foram reutilizados para garantir consistência visual e manutenção facilitada.

No desenvolvimento do sistema EMPSYS, é adotada uma estrutura baseada em operações CRUD (Create, Read, Update, Delete), amplamente utilizada em sistemas de informação que manipulam dados persistentes. As operações CRUD representam as quatro funcionalidades fundamentais de gerenciamento de registros em bancos de dados: criação, leitura, atualização e exclusão. Isso garante ao usuário a capacidade de controlar totalmente os dados do sistema, como o cadastro e manutenção de clientes, locais, empilhadeiras e seus respectivos modelos. Além disso, a arquitetura do sistema é construída seguindo o padrão MVC (Model-View-Controller), promovendo a separação de responsabilidades na aplicação e facilitando sua manutenção e escalabilidade. A utilização de controllers e views personalizados, proporciona maior controle sobre a lógica de apresentação e manipulação de dados, permitindo ajustes finos na usabilidade e adequação da aplicação ao perfil dos usuários finais. Realizou-se reuniões mensais e entregas incrementais, permitindo ajustes contínuos e maior alinhamento com as necessidades do cliente.

O desenvolvimento do sistema EMPSYS é realizado em um ambiente controlado e moderno, visando maior produtividade, portabilidade e compatibilidade com o ecossistema PHP/Symfony. É utilizado o WSL (Windows Subsystem for Linux), uma ferramenta que permite executar distribuições Linux diretamente em sistemas Windows. O ambiente também conta com o uso do Docker, uma plataforma de contêineres utilizada para isolar e padronizar os serviços da aplicação. Contêineres foram utilizados para simular o servidor PHP, o banco de dados SQLite e outros serviços auxiliares, garantindo que o comportamento do sistema se mantivesse consistente independentemente da máquina de desenvolvimento. Isso também facilita testes e replicações do ambiente em outras máquinas ou servidores. Como ambiente de desenvolvimento integrado (IDE), é utilizado o PhpStorm, uma ferramenta robusta da JetBrains com recursos avançados para desenvolvimento em PHP.

**3. Desenvolvimento**

O processo foi dividido em etapas: levantamento de requisitos, análise do sistema, implementação, testes e validação. Os requisitos foram definidos a partir de entrevistas com usuários e stakeholders, visando as funcionalidades essenciais: cadastro de clientes, locais, modelos de empilhadeiras e o inventário.

Foram desenvolvidas quatro entidades principais até o momento, com estrutura CRUD personalizada utilizando o padrão MVC [4]. Controllers e views próprios garantiram maior controle da lógica e usabilidade. O protótipo foi versionado com Git, utilizando repositório público no GitHub. Foram criadas branches para novas funcionalidades e correções, seguindo uma estratégia simplificada de Git Flow. Commits foram feitos com mensagens padronizadas, facilitando o rastreio da evolução do sistema e a colaboração entre desenvolvedores.

A interface foi construída com o framework Bootstrap 4, focando em responsividade e contraste de cores adequado, considerando o público-alvo com mais de 40 anos. Botões, formulários e fontes foram otimizados para leitura fácil e interação por dispositivos móveis. Testes unitários e de integração foram escritos utilizando a suíte de testes do Symfony (PHPUnit). Casos críticos como criação e edição de registros foram priorizados. Além disso, testes manuais com o público-alvo foram realizados para validação de usabilidade e fluxo. Todos os formulários foram protegidos contra CSRF e os dados de entrada foram validados tanto no frontend quanto no backend. Restrições de tipos, tamanhos e campos obrigatórios foram aplicadas diretamente nas entidades com o uso de validações do Symfony.

O Model (Modelo) representa a estrutura dos dados e as regras de negócio do sistema, sendo implementado por meio das entidades do projeto, que são gerenciadas com o uso do Doctrine ORM no Symfony. A View (Visualização) corresponde à camada responsável pela interface com o usuário, desenvolvida utilizando o mecanismo de templates Twig, em conjunto com o framework Bootstrap, para garantir uma interface limpa, responsiva e de fácil navegação. Por fim, o Controller (Controlador) é responsável por processar as requisições dos usuários, interagir com os modelos e repassar os dados processados para as views. [Figura 4].

A integração com Docker, o suporte a Symfony, os recursos de autocomplete, refatoração, debug e análise de código ajudam a acelerar o desenvolvimento e evitar erros. A combinação dessas ferramentas – WSL, Docker e PhpStorm – permite a criação de um ambiente de desenvolvimento moderno, eficiente e alinhado com boas práticas de engenharia de software. O uso do WSL proporciona acesso a um terminal Linux nativo, facilitando o uso de ferramentas como Composer, Symfony CLI, e comandos do sistema operacional sem a necessidade de uma máquina virtual.

Os requisitos funcionais definem as ações que o sistema deve executar para atender às necessidades dos usuários. Eles especificam as funcionalidades essenciais, como cadastro, visualização, edição e inativação dos itens de cada entidade. A aplicação está projetada para proporcionar uma experiência segura e intuitiva, contemplando os seguintes requisitos funcionais:

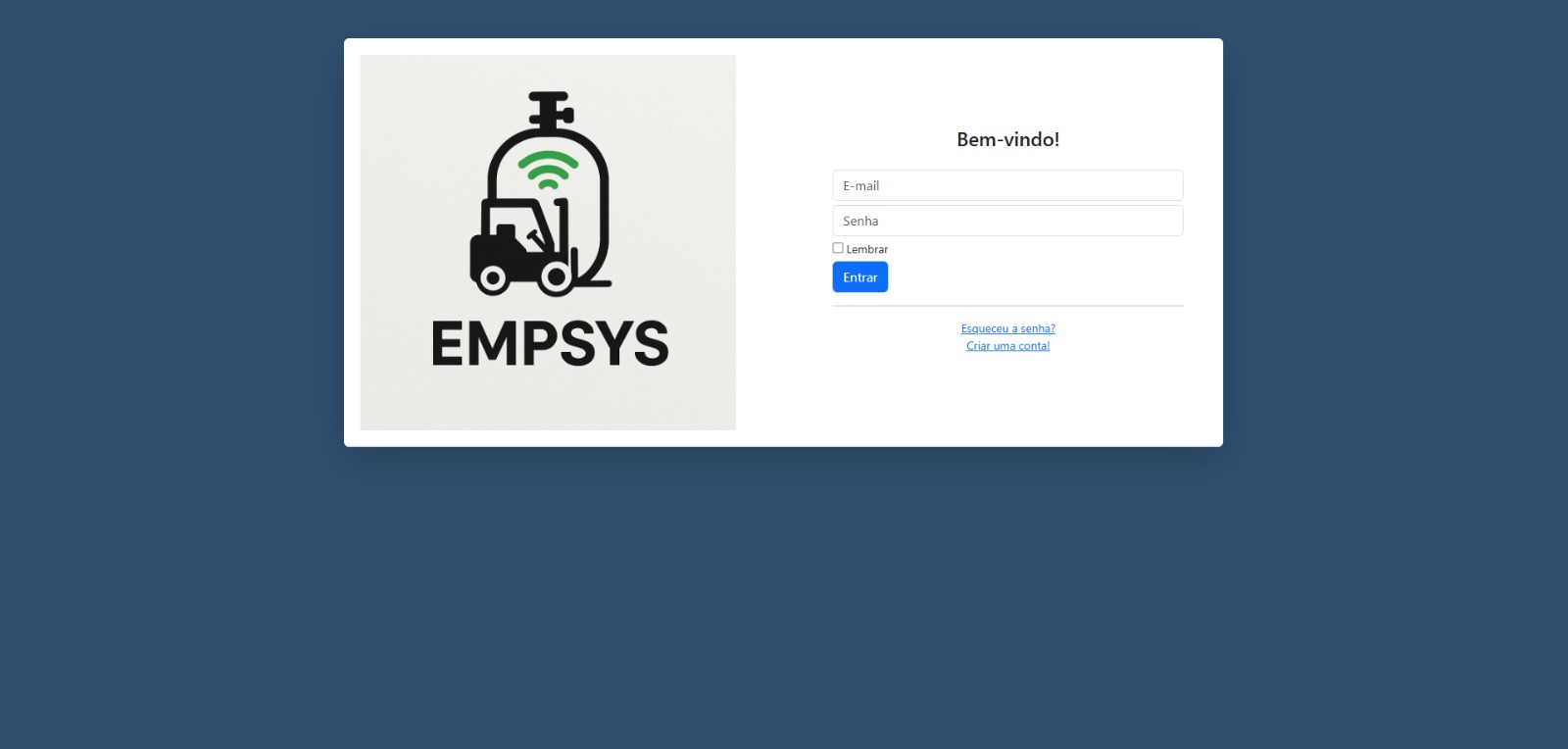
**Tabela 1. Requisitos Funcionais**

|  |  |
| --- | --- |
| **Requisitos Funcionais** | **Descrição** |
| **RF01 - Cadastro de novos itens (das entidadades: Cliente, Local, ModeloEmpilhadeira e ListaEmpiladeira)** | O sistema permitirá que o usuário com permissão adequada possa registrar novos dados para cada entidade do sistema:  - Cliente: Cadastro de um novo cliente, com nome, CNPJ/CPF, dados de contato, etc;  - Local: Cadastro de um novo local de operação (ex: galpão, armazém), vinculado a um cliente;  - ModeloEmpilhadeira: Registro de um novo modelo de empilhadeira com nome, fabricante, capacidade, tipo de energia, etc;  - ListaEmpilhadeira: Cadastro de uma nova empilhadeira no inventário, associando-a a um cliente, um local e um modelo;  Todos os campos obrigatórios serão validados no momento da criação, e o sistema exibirá mensagens de erro em caso de dados inválidos ou incompletos. |
| **RF02 – Visualização de objetos existentes (das entidadades: Cliente, Local, ModeloEmpilhadeira e ListaEmpiladeira)** | O sistema oferecerá interfaces para consulta dos registros cadastrados, com exibição em listas (tabelas) ou páginas detalhadas:  - Visualização de todos os clientes, locais, modelos e empilhadeiras cadastrados. [Figura 3];  - Filtros por campos específicos (ex: cliente por nome, empilhadeira por modelo);  - Possibilidade de visualizar os detalhes completos de um registro individual (ex: empilhadeira com local, cliente e modelo associados);  - Paginação e ordenação para facilitar a navegação entre os registros; |
| **RF03 – Edição de objetos existentes (das entidadades: Cliente, Local, ModeloEmpilhadeira e ListaEmpiladeira)** | O sistema permitirá que o usuário altere dados previamente cadastrados, com formulários similares aos de criação:  Exemplo: corrigir o nome de um cliente, trocar o local de uma empilhadeira, atualizar os dados de um modelo.  Os dados editados serão validados novamente.  Alterações serão no banco de dados e exibidas corretamente após a atualização. |
| **RF04 – Exclusão/Inativação de objetos existentes (das entidadades: Cliente, Local, ModeloEmpilhadeira e ListaEmpiladeira)** | O sistema deve permitir a remoção ou inativação de registros:  Exemplo: marcar um cliente como "inativo" ao invés de removê-lo, impedindo associação futura, mas mantendo o histórico de empilhadeiras já cadastradas com ele.  Deve haver confirmação por parte do usuário antes da exclusão ou inativação.  O sistema não deve permitir a exclusão de registros que estejam sendo usados em outras entidades (ex: não pode excluir um modelo que está vinculado a empilhadeiras ativas). |

**Tabela 2. Requisitos Não Funcionais**

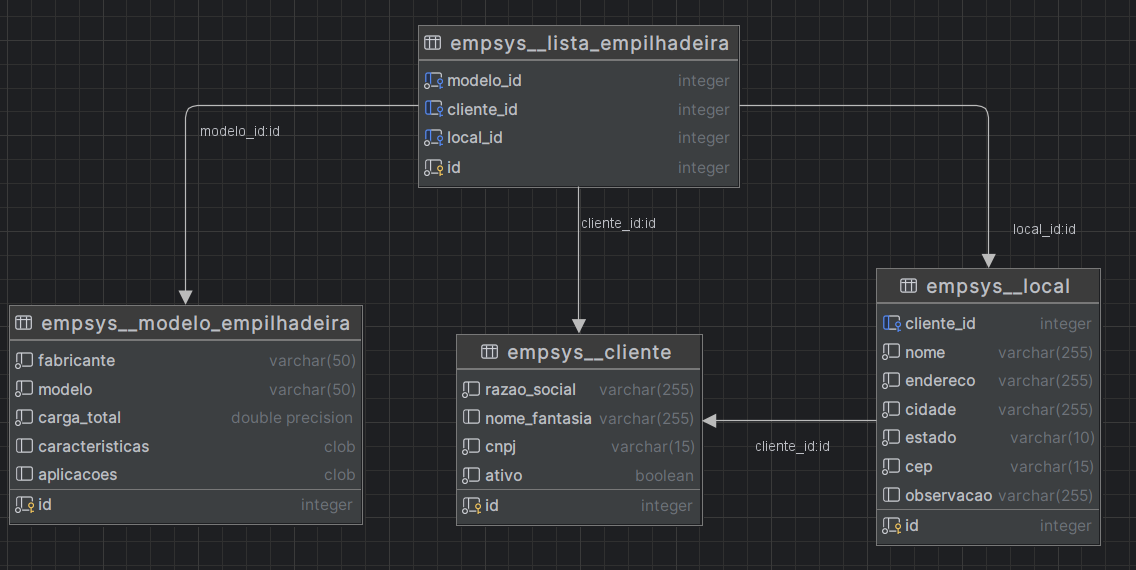
|  |  |
| --- | --- |
| **Requisitos Não Funcionais** | **Descrição** |
| **RF01 – Ambiente de execução** | O sistema deve ser implantado em servidores com sistema operacional Linux (desenvolvimento feito em Debian). O servidor deve ter suporte para PHP 7.4, Symfony 4.4. |
| **RF02 – Visualização de objetos existentes (das entidadades: Cliente, Local, ModeloEmpilhadeira e ListaEmpiladeira)** | O sistema oferecerá interfaces para consulta dos registros cadastrados, com exibição em listas (tabelas) ou páginas detalhadas:  - Visualização de todos os clientes, locais, modelos e empilhadeiras cadastrados. [Figura 3];  - Filtros por campos específicos (ex: cliente por nome, empilhadeira por modelo);  - Possibilidade de visualizar os detalhes completos de um registro individual (ex: empilhadeira com local, cliente e modelo associados);  - Paginação e ordenação para facilitar a navegação entre os registros; |
| **RNF02 - Navegadores compatíveis** | O sistema deve operar nos navegadores Google Chrome, Mozilla Firefox, Microsoft Edge, Safari e Opera. |
| **RFN03 - Backup** | O sistema realizará backups automáticos diariamente. |

**Figura 1. Página de login da aplicação feita em HTML.**

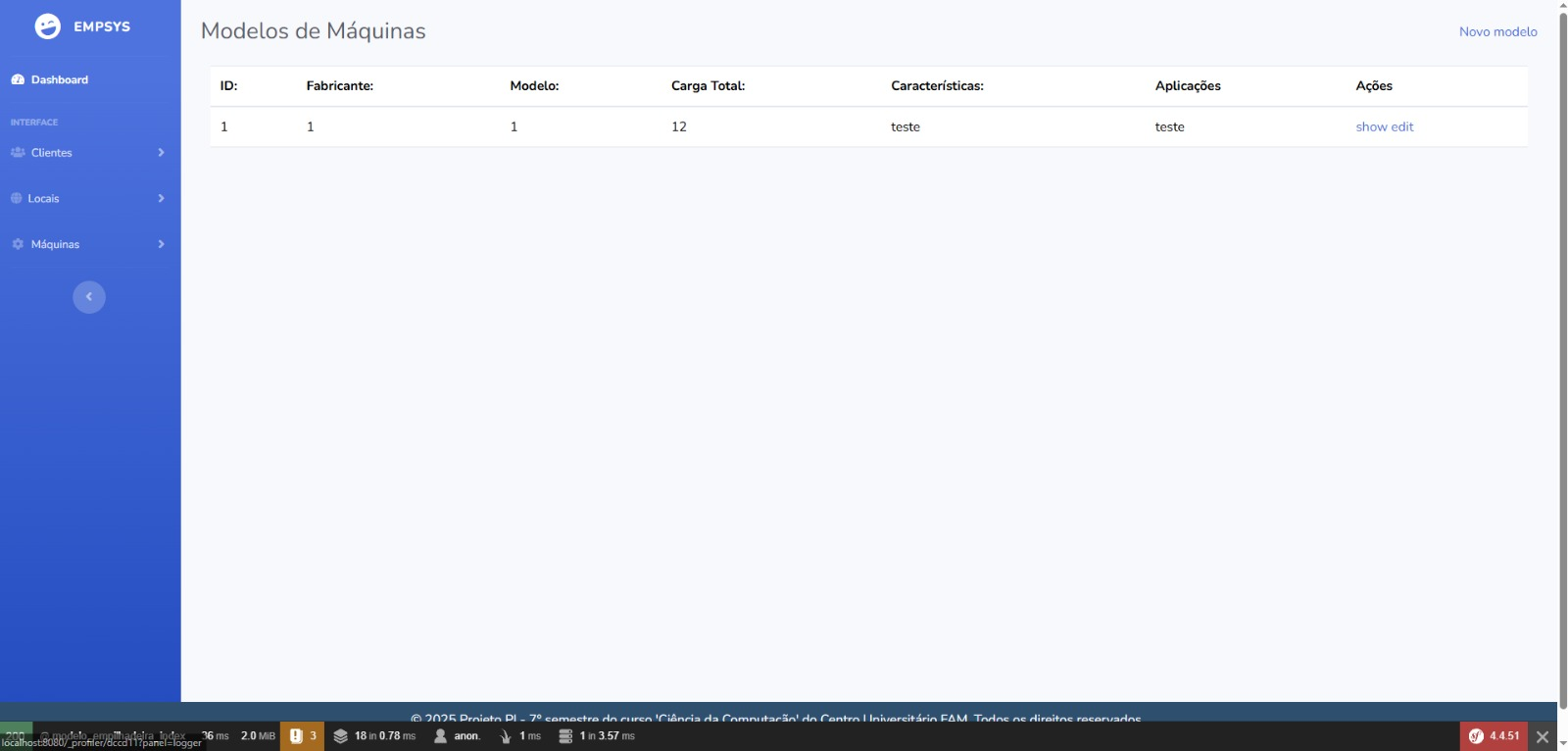


A aplicação EMPSYS está sendo desenvolvida com base em um planejamento estruturado. Nesse contexto, o diagrama de entidade e relacionamento (Figura 2) permite definir claramente as entidades, relacionamentos, atributos e seus identificadores, garantindo um armazenamento adequado e otimizado das informações. O DER foi elaborado na aplicação DataGrip, uma poderosa ferramenta multiplataforma para bancos de dados relacionais e NoSQL.

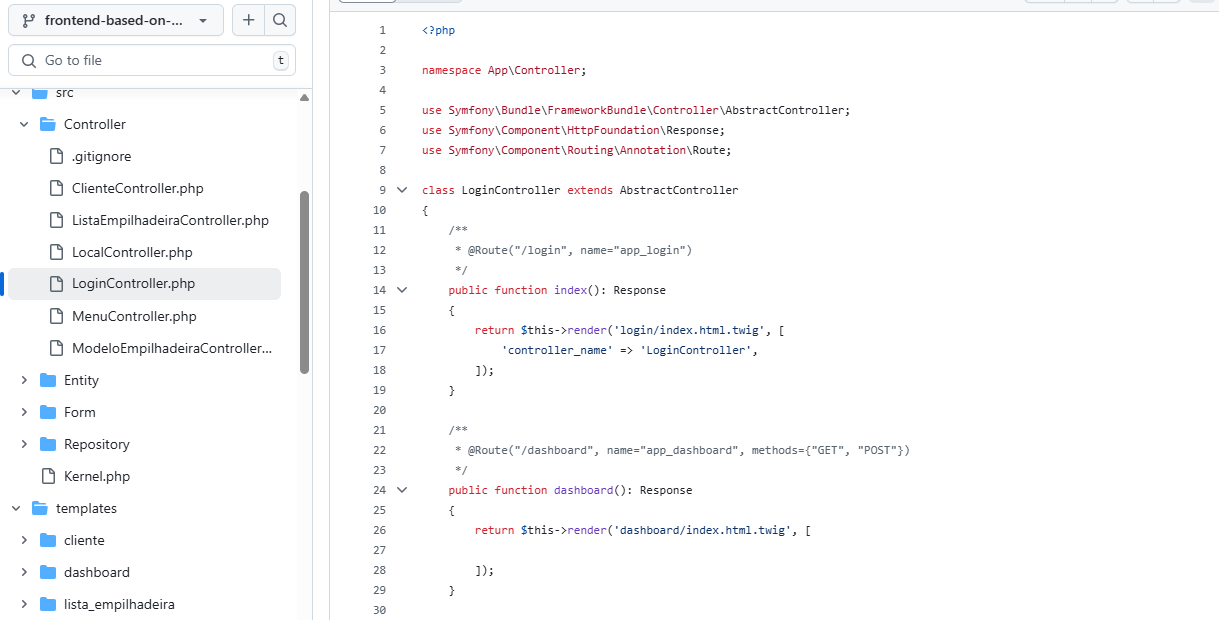
**Figura 2. DER do banco de dados.**



**Figura 3. Visualização do inventário de máquinas.**



**Figura 4. Código em PHP de um *Controller* na arquitetura *MVC.***



**4. Considerações Finais**

O EMPSYS – Sistema de Monitoramento de Máquinas – encontra-se em fase ativa de desenvolvimento, tendo concluído com sucesso uma etapa fundamental: a implementação das funcionalidades principais de cadastro e gerenciamento das entidades Cliente, Local, Modelo de Empilhadeira e Lista de Empilhadeiras. Essas funcionalidades foram desenvolvidas com uso de tecnologias como PHP 7.4, Symfony 4.4 e SQLite, dentro de uma arquitetura baseada em MVC, com apoio do ORM Doctrine e interface construída com Bootstrap 4. O sistema já oferece uma base sólida para o controle de equipamentos industriais, promovendo maior organização, rastreabilidade e eficiência no gerenciamento de empilhadeiras. O ambiente de desenvolvimento com WSL, Docker e PhpStorm contribuiu para um processo estruturado, reprodutível e alinhado às boas práticas da engenharia de software. Apesar dos avanços, o projeto segue em desenvolvimento. Estão previstas para as próximas etapas a implementação de autenticação de usuários, controle de permissões, geração de relatórios, aprimoramentos na interface com foco em usabilidade, e futuramente, a integração com sensores de monitoramento em tempo real. Além disso, funcionalidades de segurança, como backups automáticos e validação reforçada de dados, estão sendo planejadas para garantir a confiabilidade e estabilidade do sistema.

Dessa forma, o EMPSYS avança como uma solução promissora, com aplicação prática e potencial de evolução contínua. A cada nova entrega, busca-se aproximar ainda mais o sistema das necessidades reais do ambiente corporativo, reforçando seu papel como ferramenta de apoio à gestão operacional de máquinas.

**5. Referências**

[1] Gestão da manutenção preventiva de empilhadeiras [...] utilizando uma ferramenta IOT integrada a um software. Disponível em: https://www.researchgate.net/publication/355040572

[2] SYMFONY DOCS. The Symfony Framework. Disponível em: https://symfony.com/doc/current/index.html. Acesso em: 08 maio 2025.

[3] SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo: Pearson, 2019.

[4] FOWLER, Martin. Patterns of Enterprise Application Architecture. Addison-Wesley, 2003.

[5] SISTEMA DE MONITORAMENTO DE EMPILHADEIRAS: CRISTOVÃO EMPILHADEIRAS LTDA. Disponível em: https://docs.google.com/document/d/1cVvu8YLnDIoj0rSOpd-0kqJV5Tz\_0a8M/edit?usp=sharing&ouid=111362897086435197517&rtpof=true&sd=true. Acesso em 21/05/2025.