Sistema Web para Controle de Estoque

Marco Antonio Borghetti[1]
Maria Isabel Wirth Marafon[2]
Vinícius Andrei Wille[3]
Roberson Junior Rodrigues Alves[4]

Trabalho apresentado ao curso de Ciência da Computação como parte dos requisitos para avaliação no componente curricular de Engenharia de Software II.

---

[1]Discente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
marco.borghetti@unoesc.edu.br

[2]Discente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
maria.im@unoesc.edu.br

[3]Discente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
vinicius.wille@unoesc.edu.br

[4]Docente do Curso de Ciência da Computação
Unoesc-Campus de São Miguel do Oeste
Rua Oiapoc, 2011. São Miguel do Oeste-SC
roberson.alves@unoesc.edu.br

RESUMO

A ausência de um sistema informatizado para controle de estoque ainda é uma realidade em diversas pequenas empresas, que recorrem a registros manuais sujeitos a falhas, atrasos e falta de visibilidade sobre a situação real dos produtos armazenados. Este artigo apresenta o desenvolvimento de uma aplicação web responsiva e de fácil uso para controle de estoque, construída com PHP, MySQL e o framework Yii2. A solução permite o cadastro de produtos, registro de movimentações de entrada e saída, consulta de estoque em tempo real, dashboard com dados agregados, e controle de acesso por autenticação de usuários. A proposta visa reduzir erros operacionais e aumentar a eficiência da gestão de inventário com um sistema leve, de rápida implantação e baixa curva de aprendizagem.

Palavras-chave: Controle de Estoque. Sistema Web. Pequenas Empresas. Yii2. PHP. MySQL. Auditoria de Inventário.

SUMÁRIO

1. INTRODUÇÃO

O controle de estoque é uma das atividades mais críticas dentro de uma organização comercial, independentemente do seu porte. O gerenciamento inadequado pode causar impactos financeiros significativos, como perdas por vencimento de produtos, falta de mercadorias para venda, ou excesso de itens estocados. Pequenas empresas, em especial, enfrentam desafios adicionais devido à escassez de recursos para adoção de soluções tecnológicas robustas.

Neste contexto, a digitalização dos processos de estoque surge como uma necessidade urgente. Este trabalho apresenta o desenvolvimento de um sistema web simples e funcional que permite o controle de estoque de forma eficiente e confiável, utilizando tecnologias amplamente conhecidas e acessíveis.

2. FUNDAMENTAÇÃO TÉCNICA

O desenvolvimento da aplicação utilizou princípios da programação orientada a objetos (POO), o que permitiu maior modularidade, reutilização de código e manutenção facilitada. A arquitetura do projeto segue o padrão MVC (Model-View-Controller), separando responsabilidades entre a lógica de negócios, a interface do usuário e a manipulação de dados.

A autenticação dos usuários foi implementada utilizando *password_hash()* e *password_verify()*, promovendo segurança no armazenamento de senhas. O controle de sessões foi aplicado para garantir que apenas usuários autenticados pudessem acessar funcionalidades sensíveis, como o cadastro de produtos ou movimentações de estoque.

A aplicação também faz uso de versionamento de banco de dados por meio de *migrations* e gestão de dependências através do *Composer*, o que facilita o deploy e a colaboração entre desenvolvedores.

3. OBJETIVOS

3.1 Objetivo Geral

Desenvolver um sistema web responsivo, acessível e de fácil utilização para controle de estoque em pequenas empresas.

3.2 Objetivos Específicos

- Permitir o cadastro, edição e exclusão de produtos.
- Registrar movimentações de entrada e saída com rastreabilidade.
- Exibir dashboards com dados de produtos e valor total em estoque.
- Restringir funcionalidades através de autenticação de usuário.
- Garantir usabilidade por meio de uma interface simples e intuitiva.

4. METODOLOGIA

A metodologia adotada para o desenvolvimento do sistema seguiu princípios da engenharia de software com foco em desenvolvimento incremental, priorizando entregas contínuas e testáveis. Utilizou-se uma abordagem prática com o apoio de ferramentas modernas e boas práticas de desenvolvimento web. A equipe optou por organizar o trabalho em etapas curtas e bem definidas, permitindo revisões constantes e ajustes durante o ciclo de desenvolvimento.

O processo foi orientado por práticas comuns da modelagem orientada a objetos e da divisão clara de responsabilidades entre back-end, front-end e banco de dados. Todas as decisões técnicas foram tomadas considerando a facilidade de manutenção, simplicidade para o usuário final e possibilidade de expansão futura do sistema.

4.1 Uso de Padrões de Projeto

Durante o desenvolvimento do sistema, foram identificados e aplicados dois padrões de projeto adequados ao contexto da aplicação: O primeiro padrão é o MVC (*Model-View-Controller***)**, que permite organizar o sistema em três camadas distintas: modelos (acesso aos dados), controladores (lógica de aplicação) e visões (interface com o usuário). Este padrão foi implementado com o suporte nativo do framework Yii2 e facilitou a separação de responsabilidades, melhorando a manutenção e a escalabilidade do sistema.

O segundo padrão utilizado foi o *Singleton*, aplicado implicitamente na conexão com o banco de dados. Através do Yii2, foi possível garantir que apenas uma instância da conexão (Yii::$app->db) fosse utilizada ao longo de toda a execução do sistema, otimizando o uso de recursos e evitando múltiplas conexões desnecessárias.

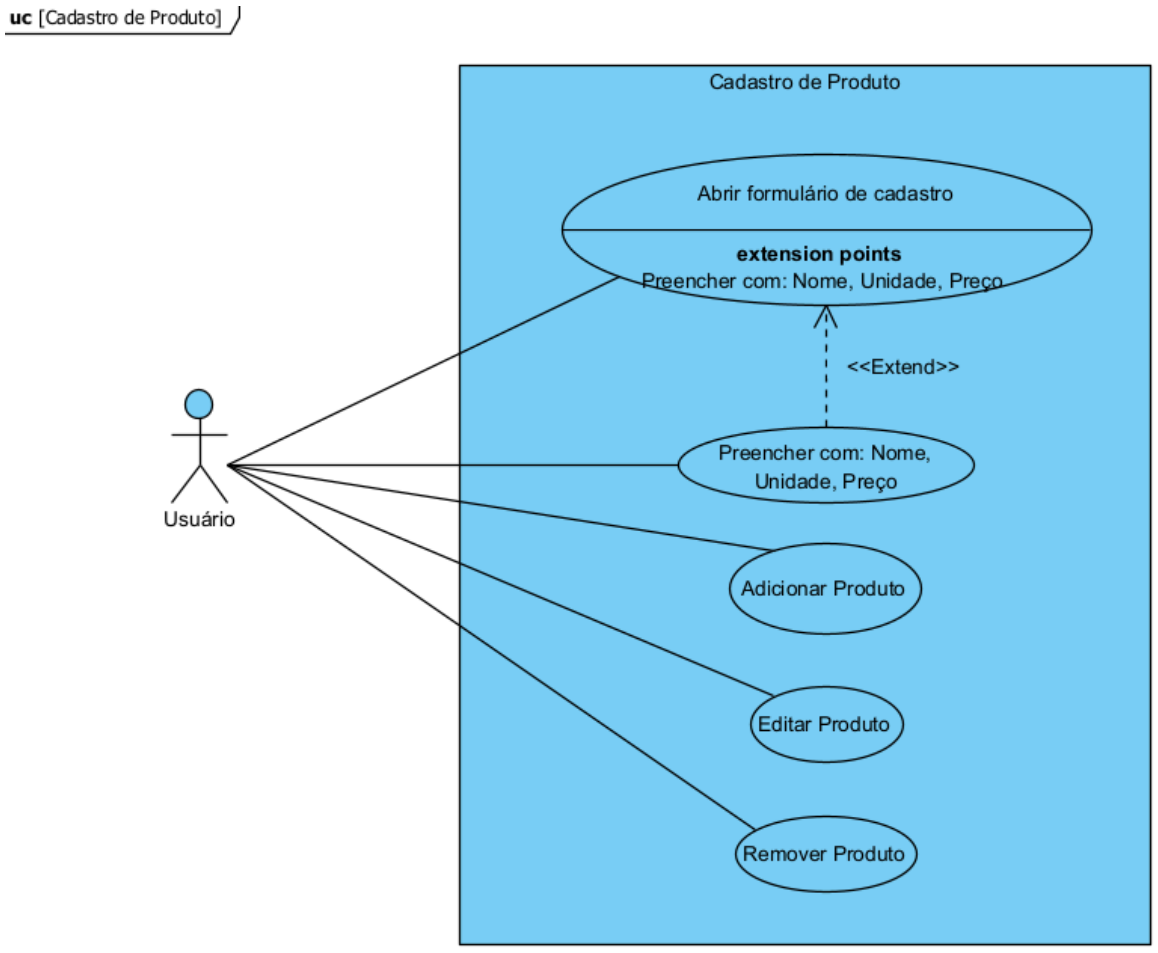
A escolha desses padrões se deu pela sua adequação ao contexto de uma aplicação web de pequeno porte, com foco em desempenho, organização e facilidade de manutenção. Ambos contribuíram diretamente para a estruturação sólida do código e a entrega de uma solução funcional e bem organizada.

5. DIAGRAMAS

5.1 Casos de Uso

O diagrama de casos de uso demonstra as principais funcionalidades relacionadas ao gerenciamento de produtos e as interações realizadas pelo ator “Usuário”. Entre os casos de uso destacados estão: abrir o formulário de cadastro, preencher os campos obrigatórios (nome, unidade e preço), adicionar, editar e remover produtos. Utilizamos a notação <<extend>> para representar que o preenchimento dos dados estende a ação de abrir o formulário. Esse diagrama contribui para o mapeamento claro dos requisitos funcionais do sistema sob a perspectiva do usuário final, facilitando a validação com os stakeholders.

Figura 1: Diagrama de Casos de Uso.

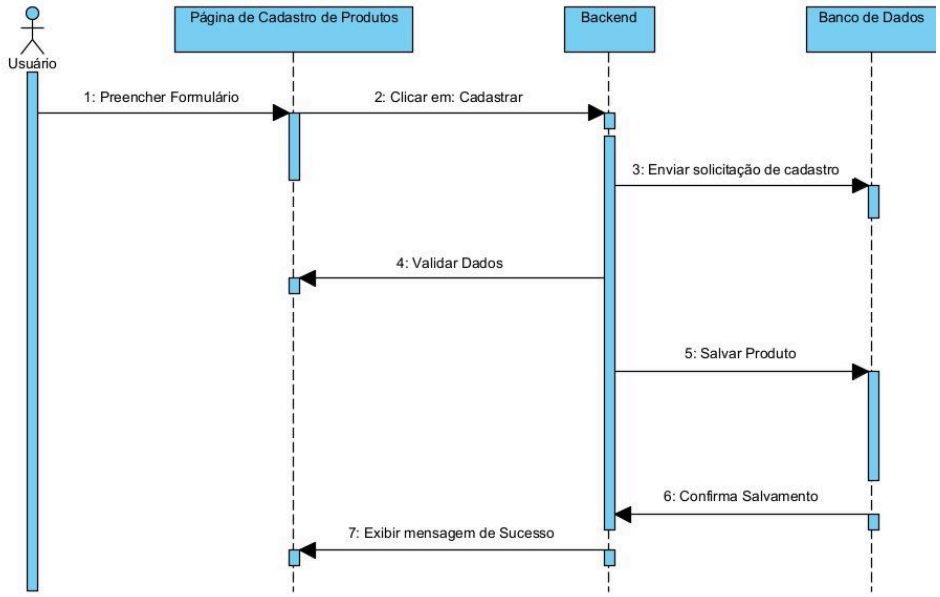


Fonte: Os autores.

5.2 Sequência

O diagrama de sequência ilustra o fluxo de execução do processo de cadastro de um produto no sistema. Ele apresenta a interação entre os principais elementos envolvidos: o usuário, a interface da página de cadastro, o backend e o banco de dados. A sequência inicia com o preenchimento do formulário pelo usuário, seguido pelo envio da requisição ao servidor. O backend valida os dados e executa a persistência no banco de dados. Após a confirmação, uma mensagem de sucesso é exibida na interface. Este diagrama reforça a lógica sequencial e síncrona do processo, garantindo clareza no entendimento das etapas executadas durante o cadastro.

Figura 2: Diagrama de Sequência.

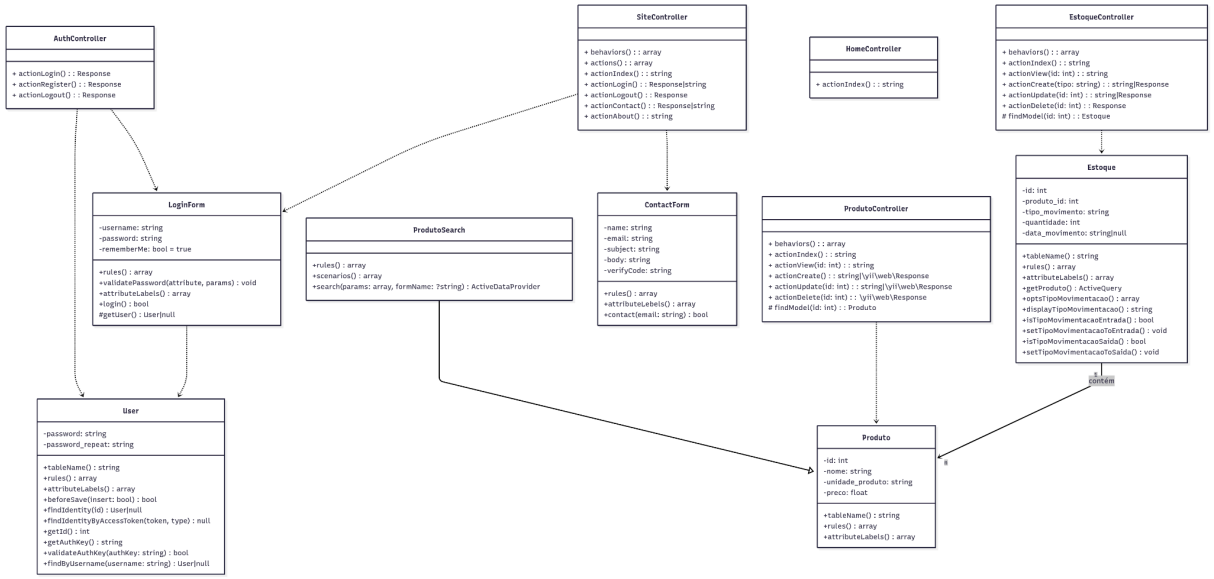


Fonte: Os autores.

5.3 Classes

O diagrama de classes representa a estrutura principal da aplicação, evidenciando as classes mais relevantes, seus atributos, métodos e relacionamentos. Ele mostra claramente a adoção do padrão de arquitetura MVC, com classes responsáveis pelo controle de fluxo (Controllers), regras de negócio e validações (Models), além de formulários auxiliares como LoginForm e ContactForm. Também evidencia a separação entre Produto, Estoque e User, com seus respectivos métodos de manipulação de dados e interações com os controladores. Esse diagrama foi essencial para a organização e compreensão das responsabilidades de cada componente dentro do sistema.

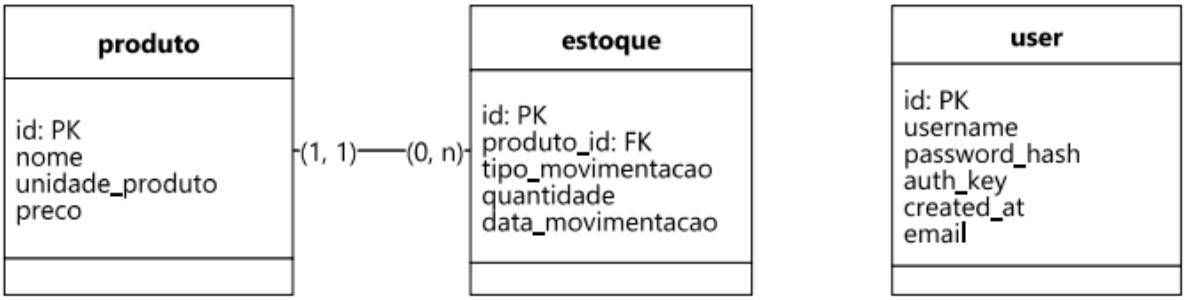
Figura 3: Diagrama de Classes



Fonte: Os autores.

5.4 Modelo Entidade-Relacionamento

O diagrama entidade-relacionamento apresenta a estrutura lógica do banco de dados utilizado no sistema de controle de estoque. Ele é composto por três entidades principais: **produto, estoque** e **user.**



Fonte: Os autores.

A entidade **produto** armazena informações básicas sobre os itens cadastrados, como nome, unidade_produto (ex: UN, ME, KG, LT) e preco. Cada produto possui um identificador único (id), representado como chave primária (PK). A entidade **estoque** é responsável por registrar as movimentações de entrada e saída dos produtos. Ela possui uma chave estrangeira (produto_id) que referencia a entidade produto, além dos campos tipo_movimentacao, quantidade e data_movimentacao. A cardinalidade expressa que um produto pode estar associado a nenhuma ou várias movimentações (0,n), mas toda movimentação pertence

obrigatoriamente a um único produto (1,1). Já a entidade **user** representa os usuários do sistema, contendo atributos como username, password_hash, auth_key, email e created_at.

Essa tabela é utilizada no processo de autenticação e controle de acesso às funcionalidades sensíveis do sistema. Esse modelo relacional foi a base para a construção das *migrations* no Yii2, garantindo a integridade referencial entre as tabelas e permitindo consultas consistentes entre produtos e suas respectivas movimentações no estoque.

6. DESENVOLVIMENTO

A linguagem de programação escolhida foi o PHP na versão 7.4, amplamente utilizada no desenvolvimento web. Para organizar e estruturar o código, adotamos o framework Yii2, que além de ser leve e de fácil configuração, oferece suporte nativo ao padrão MVC (*Model-View-Controller*), facilitando a separação de responsabilidades e a escalabilidade do projeto. O banco de dados utilizado foi o MySQL, por sua confiabilidade e compatibilidade com o ambiente de desenvolvimento escolhido, o XAMPP. As dependências foram gerenciadas via Composer.

Na camada de apresentação, utilizamos HTML e CSS com o auxílio do Bootstrap, garantindo responsividade e boa usabilidade em diferentes tamanhos de tela. O Chart.js foi utilizado na construção dos gráficos de movimentação do dashboard, permitindo uma visualização amigável dos dados.

A estrutura do projeto foi organizada em diretórios funcionais: controladores responsáveis pela lógica da aplicação, modelos com regras de validação e acesso ao banco de dados, e views contendo as interfaces renderizadas. Scripts de inicialização, configurações de banco e *migrations* foram estruturados de forma clara, permitindo fácil replicação do ambiente.

O desenvolvimento seguiu as seguintes etapas práticas: inicialmente, modelamos os dados com base nos requisitos. Em seguida, criamos as migrations que estruturam o banco de dados. Posteriormente, desenvolvemos os CRUDs de produtos e movimentações, integrando formulários e tabelas com validação de dados. A seguir, implementamos o layout com Bootstrap e criamos o dashboard informativo com cards e gráficos. Na reta final, foi implementado o sistema de autenticação com hashes de senha e sessões PHP, restringindo o

acesso às rotas principais. Por fim, realizamos testes manuais e ajustes com base no feedback de uso.

Essa abordagem proporcionou um fluxo de trabalho eficiente, com entregas frequentes e funcionais. O uso das ferramentas modernas, como o Yii2, Composer e Bootstrap, contribuiu para um desenvolvimento rápido, organizado e com foco na experiência do usuário final.

7. Validação e Testes

Como parte do processo de desenvolvimento da aplicação, foram realizados testes manuais e validações constantes em cada etapa. A equipe optou por uma abordagem prática, testando os módulos logo após sua implementação para garantir seu correto funcionamento e detectar eventuais falhas de forma antecipada. Isso incluiu o cadastro e edição de produtos, registro de movimentações e verificação dos dados refletidos no dashboard.

7.1 Testes Unitários

Os testes unitários foram desenvolvidos utilizando a ferramenta PHPUnit, amplamente adotada para projetos em PHP. As classes ProdutoTest e EstoqueTest foram criadas na pasta tests/unit/models, com o objetivo de verificar o comportamento isolado de métodos e validações dos modelos principais do sistema.

A estrutura de testes foi configurada via phpunit.xml, e a execução foi realizada com o comando:

bash

```
./vendor/bin/phpunit --colors=always
```

Estes testes permitiram garantir, por exemplo, que o sistema rejeita cadastros com dados inconsistentes e que os cálculos de valores de estoque são executados corretamente.

7.2 Testes de Funcionalidade

Os testes funcionais foram realizados utilizando a ferramenta Katalon Recorder, permitindo a simulação do uso real do sistema por meio da gravação e reprodução de ações no navegador. Foram testadas funcionalidades como:

- Cadastro e edição de produtos;
- Registro de movimentações de entrada e saída;
- Login e logout de usuários;
- Navegação pelas páginas protegidas por autenticação.

Os scripts gravados foram exportados e arquivados para reuso em futuras validações. Esses testes ajudaram a comprovar que o sistema atende aos requisitos definidos, funcionando conforme o esperado em situações reais.

7.3 Testes de Carga

Para validar o desempenho da aplicação sob uso intensivo, foram executados testes de carga utilizando a ferramenta Apache JMeter. O plano de testes incluiu múltiplas requisições simultâneas aos principais endpoints da aplicação, simulando acessos em massa ao sistema.

Foram gerados gráficos de desempenho (Summary Report e Resultados Visuais) que demonstraram boa estabilidade, sem queda significativa no tempo de resposta, mesmo em cenários com grande volume de requisições. Isso confirma que o sistema é capaz de atender à demanda esperada em seu contexto de uso — pequenas e médias empresas.

8. RESULTADOS OBTIDOS

O sistema entregue apresentou um conjunto sólido de funcionalidades que atendem aos principais requisitos definidos no início do projeto. Entre os principais resultados, destaca-se a implementação do módulo de cadastro de produtos, onde é possível informar o nome, a unidade de medida (como UN, ME, KG ou LT) e o preço em reais com formatação adequada. Além disso, foi desenvolvido um módulo de movimentações de estoque, no qual o operador pode registrar entradas e saídas de produtos, escolhendo o item em um campo de seleção, definindo a quantidade e a data da movimentação.

Outro destaque foi o dashboard informativo, que apresenta em tempo real o total de produtos cadastrados, o valor financeiro estimado do estoque e um gráfico interativo de movimentações ao longo do tempo, utilizando a biblioteca Chart.js. A segurança e o controle de acesso também foram considerados com a criação de um sistema de autenticação de usuários, com telas de login e cadastro, uso de hash para senha e restrição de rotas. Por fim, a interface responsiva foi desenvolvida com o framework Bootstrap, contendo elementos de

navegação claros, labels explicativos, breadcrumbs e botões compactos, que garantem uma boa experiência de uso em diferentes dispositivos.

9. CONSIDERAÇÕES FINAIS

O sistema desenvolvido se mostrou eficiente em resolver os principais desafios enfrentados por pequenas empresas no controle de estoque. Sua interface simples, aliada às funcionalidades essenciais, proporciona ganhos em precisão, agilidade e organização, sem exigir infraestrutura complexa ou conhecimento técnico avançado por parte do usuário.

Para trabalhos futuros, recomenda-se:

- Exportação de relatórios (PDF/Excel);
- Níveis de permissão por usuário;
- Integração com sistemas de vendas ou ERPs.

10. APRENDIZADOS E EXPERIÊNCIA DO GRUPO

Durante o desenvolvimento do projeto, cada integrante do grupo teve a oportunidade de aplicar e aprimorar conhecimentos práticos em PHP, banco de dados, estruturação de projetos com MVC, autenticação de usuários e front-end responsivo. Além do desenvolvimento técnico, houve avanço significativo na organização do trabalho em equipe, divisão de tarefas e no uso de ferramentas de versionamento e testes. A entrega do projeto possibilitou vivenciar na prática a criação de uma solução real com foco na usabilidade e impacto direto em um problema cotidiano de gestão.

11. REFERÊNCIAS

BALLOU, Ronald H. *Logística empresarial*. 1. ed. São Paulo: Atlas, 2006.

LAUDON, Kenneth C.; LAUDON, Jane P. *Sistemas de informação gerenciais*. São Paulo: Pearson, 2012.

YII Framework. *Yii2 Documentation*. Disponível em: https://www.yiiframework.com/doc. Acesso em: jun. 2025.