UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GABRIEL HENRIQUE RIBAS

DESENVOLVIMENTO DA INTERFACE GRÁFICA PARA CRIAÇÃO DE OBJETOS DE APRENDIZAGEM NA FERRAMENTA FARMA

GUARAPUAVA

2025

GABRIEL HENRIQUE RIBAS

DESENVOLVIMENTO DA INTERFACE GRÁFICA PARA CRIAÇÃO DE OBJETOS DE APRENDIZAGEM NA FERRAMENTA FARMA

DEVELOPMENT OF THE GRAPHICAL INTERFACE FOR CREATING LEARNING OBJECTS IN THE FARMA TOOL

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientadora : Profª. Drª. Renata Luiza Stange Carneiro Gomes

Coorientador: Prof. Dr. Diego Marczal e Prof. Dr. Alex Sandro De Castilho

GUARAPUAVA

2025

GABRIEL HENRIQUE RIBAS

DESENVOLVIMENTO DA INTERFACE GRÁFICA PARA CRIAÇÃO DE OBJETOS DE APRENDIZAGEM NA FERRAMENTA FARMA

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 01/dezembro/2025

Profª. Renata Luiza Stange
Doutorado
Universidade Tecnológica Federal do Paraná – UTFPR

Prof. Diego Marczal
Doutorado
Universidade Tecnológica Federal do Paraná – UTFPR

Prof. Alex Sandro De Castilho
Doutorado
Universidade Tecnológica Federal do Paraná – UTFPR

Profª. Sediane Carmem Lunardi Hernandes
Doutorado
Universidade Tecnológica Federal do Paraná – UTFPR

GUARAPUAVA

2025

"Eu não disse que seria fácil; disse que valeria a pena." - (São João Bosco)

RESUMO

Com o avanço da tecnologia, a aplicação de ferramentas digitais em diversas áreas da educação tem se tornado cada vez mais frequente. A fim de auxiliar o educador, estruturas que possibilitam testar, avaliar e verificar o processo de ensino têm se mostrado fundamentais para proporcionar aos estudantes um maior aprofundamento nos conteúdos. Nesse contexto, este trabalho descreve o desenvolvimento de uma nova versão da ferramenta de Autoria para Remediação de Erros com Mobilidade na Aprendizagem (FARMA), com o objetivo de criar novas interfaces para a área de elaboração de Objetos de Aprendizagem (OA). Para isso, o projeto reformulou a estrutura da versão anterior, que utilizava uma arquitetura distribuída baseada em Application Programming Interface (API), adotando agora uma arquitetura monolítica. Essa mudança permitiu concentrar o desenvolvimento nas funcionalidades essenciais para a criação e gestão de um OA. Como resultado, foram implementadas novas áreas que facilitam a interação do educador com o processo de criação dos objetos, além de oferecer uma estrutura mais simples, organizada e de fácil manutenção, considerando possíveis expansões futuras.

Palavras-chave: objetos de aprendizagem; farma; arquitetura monolitica; sistemas educacionais.

ABSTRACT

With the advancement of technology, the use of digital tools in different areas of education has become increasingly common. To support educators, structures that allow testing, evaluating, and monitoring the teaching–learning process have proven essential to providing students with deeper engagement with the content. In this context, this work describes the development of a new version of the Authoring Tool for Error Remediation with Mobility in Learning (FARMA), aiming to create new interfaces for the design of Learning Objects (OA). To achieve this, the project restructured the previous version of the tool, which relied on a distributed architecture based on an Application Programming Interface (API), and adopted a monolithic architecture. This change enabled the development to focus on the essential functionalities required for the creation and management of a OA. As a result, new areas were implemented to facilitate educators' interaction with the authoring process, offering a simpler, more organized, and easily maintainable structure while considering possible future expansions.

Keywords: learning objects; farma; monolithic system; educational systems.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAGEM DE CÓDIGOS FONTE

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	Application Programming Interface
CD	Continuous Delivery
CI	Continuous Integration
CSS	Cascading Style Sheets
FAB	Floating Action Button
FARMA	Ferramenta de Autoria para Remediação de Erros com Mobilidade na Aprendizagem
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
OA	Objetos de Aprendizagem
PR	Pull Request
SMTP	Simple Mail Transfer Protocol
SVG	Scalable Vector Graphics

SUMÁRIO

1 INTRODUÇÃO

A criação de ferramentas tecnológicas educacionais tem o objetivo de facilitar e enriquecer o aprendizado, por meio de dispositivos como computadores, tablets e smartphones com acesso à *Internet*. Essas tecnologias possibilitam o acesso a informações e recursos, promovendo a integração de novas abordagens de ensino em diversas áreas do conhecimento. Assim, busca-se alcançar uma melhoria contínua no processo de ensino, garantindo qualidade na aprendizagem e contribuindo para o desenvolvimento pessoal e profissional (SOUSA, 2011).

As tecnologias educacionais têm impactado e transformado a educação de diversas maneiras. Uma dessas transformações é a possibilidade de personalização do ensino, que permite aos educadores adaptar os conteúdos e dinamizar suas atividades, de modo que os alunos aprendam de forma mais acessível e eficaz (DAMASCENO *et al.*, 2024). Recursos digitais, como plataformas de aprendizado adaptativo, jogos educativos e realidade aumentada, têm o potencial de aumentar o engajamento dos estudantes e aprimorar seu desempenho acadêmico.

Nesse cenário, um importante recurso utilizado para aprimorar as estratégias de ensino são os Objetos de Aprendizagem (OA). Esses objetos podem ser desenvolvidos em diferentes formatos e mídias, desde elementos simples, como uma animação ou apresentação de slides, até estruturas mais complexas, como simulações. Geralmente, são organizados em módulos reutilizáveis em diferentes contextos. Em outras palavras, cada OA é um módulo independente, com conteúdo autoexplicativo e autossuficiente, sem a necessidade de materiais complementares para seu entendimento (TAROUCO, 2014).

A Ferramenta de Autoria para Remediação de Erros com Mobilidade na Aprendizagem (FARMA) é uma plataforma de aprendizado adaptativo que tem como objetivo principal a criação de OA voltados à identificação e correção de erros cometidos pelos alunos, promovendo a remediação de dificuldades de aprendizagem (MARCZAL, 2014). A FARMA oferece um ambiente flexível e acessível, voltado à promoção do ensino-aprendizagem de Matemática, possibilitando a criação e visualização de OA. A versão atual da FARMA[1] passou por diversas melhorias desde a versão inicial desenvolvida por Marczal (2014), cujos avanços podem ser observados na linha do tempo apresentada nos capítulos posteriores.

Apesar dos progressos alcançados em versões anteriores, a FARMA ainda apresenta desafios relacionados à integração entre seus módulos e à dificuldade de manutenção decorrente do uso de uma arquitetura distribuída, baseada em *Application Programming Interface* (API). Essa abordagem, embora promova modularidade, pode gerar obstáculos ao gerenciamento e à evolução do sistema.

Neste contexto, este projeto busca dar continuidade e aprimorar os trabalhos anteriores, principalmente os trabalhos de de Vitek (2023) e Bonin (2024), propondo o redesenho das telas de criação de OA e a readequação da FARMA para uma arquitetura monolítica. Essa migração

[1] Disponível em https://farma.educacional.mat.br/, acessada em 17 de outubro de 2024

tem como propósito tornar o sistema mais simples, coeso e de fácil manutenção, garantindo maior estabilidade e eficiência no processo de desenvolvimento e uso da ferramenta.

1.1 Objetivos

Nesta seção, apresentam-se os objetivos que orientaram o desenvolvimento deste trabalho.

1.1.1 Objetivo geral

Desenvolver a interface gráfica para criação de Objetos de Aprendizagem (OA) na Ferramenta de Autoria para Remediação de Erros com Mobilidade na Aprendizagem (FARMA), substituindo o uso da *Application Programming Interface* (API) por uma arquitetura monolítica, a fim de facilitar a criação e a gestão dos objetos de aprendizagem.

1.1.2 Objetivos específicos

- Reestruturar a FARMA, substituindo a arquitetura distribuída baseada em API por uma arquitetura monolítica;
- Desenvolver a interface gráfica da área de criação de introdução ao conteúdo de um OA;
- Desenvolver a interface gráfica da área de criação de exercícios de um OA;
- Desenvolver a interface gráfica da área de criação de passos de solução do OA;
- Desenvolver a interface gráfica da área de definição de dicas do OA.

1.2 Justificativa

A migração de um sistema baseado em uma arquitetura distribuída para uma arquitetura monolítica pode trazer vantagens significativas para o desenvolvimento e a manutenção de sistemas, especialmente quando se busca reduzir a complexidade operacional e melhorar a experiência do usuário. De acordo com Bass (2013), a arquitetura de software representa o conjunto de estruturas necessárias para compreender e analisar um sistema, incluindo seus elementos, as relações entre eles e suas propriedades. Nesse sentido, a escolha da arquitetura é um fator determinante para garantir a eficiência, a manutenibilidade e a evolução de uma aplicação.

Conforme Richards (2024), as arquiteturas de software podem ser classificadas, de modo geral, em dois estilos principais — *monolítico e distribuído* —, a arquitetura monolítica consiste em uma única unidade de implementação, na qual todo o código da aplicação é desenvolvido e implantado de forma integrada. Já a arquitetura distribuída é composta por várias unidades de implementação que se comunicam por meio de protocolos de acesso remoto. Embora nenhum esquema de classificação seja perfeito, essa distinção é útil, pois todas as arquiteturas distribuídas compartilham desafios e problemas que não ocorrem em arquiteturas monolíticas. Essa diferenciação permite uma análise mais precisa das vantagens e desvantagens de cada abordagem.

Embora as arquiteturas distribuídas ofereçam benefícios como escalabilidade e alta disponibilidade, elas também apresentam desafios relacionados à confiabilidade da rede, à complexidade de integração entre múltiplos serviços e ao aumento dos custos de infraestrutura. Como destacam Richards (2024), os estilos arquiteturais distribuídos, como os baseados em serviços ou orientados a eventos, exigem mecanismos adicionais de orquestração e controle, o que pode gerar gargalos e aumentar a dificuldade de manutenção. Essas limitações tornam-se especialmente relevantes em contextos acadêmicos, nos quais os recursos de infraestrutura e o tempo de desenvolvimento são, em geral, restritos.

No caso da FARMA, o uso de uma API externa para comunicação entre os módulos de criação e visualização de Objetos de Aprendizagem (OA) tem se mostrado um fator que eleva a complexidade do sistema, dificultando sua manutenção e evolução. A dependência de múltiplos componentes distribuídos pode resultar em falhas de comunicação, inconsistências e latências, impactando negativamente o desempenho e a experiência do usuário.

A adoção de uma arquitetura monolítica, conforme discutido por Fowler (2006) e Newman (2021), favorece a simplicidade na manutenção e reduz a complexidade de infraestrutura, uma vez que todo o código se encontra em um único repositório e os componentes comunicam-se localmente. Essa abordagem permite maior controle sobre o ciclo de vida da aplicação, facilita o versionamento e torna o processo de depuração mais ágil. Além disso, conforme Kleppmann (2017), a comunicação interna em sistemas monolíticos tende a apresentar menor latência, já que elimina a sobrecarga de chamadas remotas típicas de sistemas distribuídos.

Paralelamente à mudança arquitetural, destaca-se a necessidade de oferecer uma interface gráfica (GUI) aprimorada para a FARMA. A ausência de uma interface integrada limita o potencial pedagógico da ferramenta, tornando o processo de criação e gestão dos OA menos intuitivo e acessível. Ao integrar a GUI diretamente na aplicação monolítica, é possível proporcionar uma interação mais fluida entre o usuário e o sistema, favorecendo a usabilidade e a adoção por professores e estudantes. Essa integração também reforça o princípio de coesão entre as camadas da aplicação, conforme preconizado por boas práticas de arquitetura de software.

Portanto, a atualização da FARMA justifica-se tanto pela necessidade de modernizar sua estrutura técnica quanto por fundamentos teóricos de engenharia de software. Ao migrar para

uma arquitetura monolítica e desenvolver uma interface gráfica dedicada, este trabalho busca reduzir a complexidade operacional, melhorar o desempenho do sistema e ampliar sua adoção, contribuindo para a eficácia e a sustentabilidade da FARMA no contexto educacional.

2 FARMA

A Ferramenta de Autoria para a Remediação de Erros com Mobilidade na Aprendizagem (FARMA) teve sua origem na tese de doutorado de Marczal (2014), a qual identificou a escassez de ferramentas educacionais voltadas ao monitoramento do processo de ensino e aprendizagem por meio da análise de tentativas e erros. Essa lacuna já havia sido apontada no estudo de Marczal (2011), que destacou a ausência de um gerenciador de Objetos de Aprendizagem (OA) capaz de acompanhar a evolução do aluno e utilizar seus erros como subsídios para o reforço do conteúdo e a promoção da remediação.

2.1 Linha do Tempo

A FARMA tem sido objeto de investigação em diversos estudos acadêmicos, abrangendo dissertações de mestrado, teses de doutorado, trabalhos de conclusão de curso, artigos científicos e, inclusive, premiações. Com o intuito de ilustrar essa evolução, foi elaborada uma linha do tempo que apresenta, de forma cronológica, os principais marcos do desenvolvimento da FARMA ilustrada na Tabela 1.

Tabela 1 – Representação da linha do Tempo FARMA (2009-2025)

2009	Objetos de aprendizagem generalizáveis para o currículo de Matemática do ensino médio (DIRENE, 2009) - Melhor Artigo do Workshop de Informática na Escola (WIE) - Sociedade Brasileira de Computação (SBC) (2º Lugar) [1]
2010	Um arcabouço que enfatiza a retroação a contextos de erro durante o acesso a conteúdos educacionais - Sociedade Brasileira de Computação (SBC) (3º Colocado cat. Dissertação)(MARCZAL, 2010)
2011	Classificação automática de erros de aprendizes humanos do processo de indução analítica (BAZZO, 2011)
2012	FARMA: Uma ferramenta de autoria para objetos de aprendizagem de conceitos matemáticos - Simpósio Brasileiro de Informática na Educação (SBIE)(MARCZAL, 2012)
2013	Arquitetura para remediação de erros baseada em múltiplas representações externas (LEITE, 2013)
2014	Farma: uma ferramenta de autoria para objetos de aprendizagem de conceitos matemáticos (MARCZAL, 2014)
2015	Sequenciamento adaptativo de exercícios baseado na correspondência entre a dificuldade da solução e o desempenho dinâmico do aprendiz - Sociedade Brasileira de Computação (SBC) (1º colocado na cat. Tese)(SILVA, 2015)
2015	Informática educacional e a mediação do erro na educação: um estudo teórico-crítico e uma proposta de instrumento computacional (KUTZKE, 2015)
2015	Avaliação do impacto da retroação na aprendizagem apoiada por uma ferramenta educacional (MOURA, 2015)

[1] Disponível em https://www.inf.ufpr.br/alexd/wie2009.pdf

- 2015 • Otimizando o processo de Ensino e Aprendizagem com a Arquitetura para Desenvolvimento de Objetos de Aprendizagem - ADOA (LEITE, 2015)
- 2016 • Metodologia e Software Educacional para a Investigação e Remediação de Erros Conceituais em Matemática (MARCZAL, 2016)
- 2017 • ATAUDIW - Uma ferramenta de autoria para auxiliar o uso da lousa digital interativa (COSTA, 2017)
- 2017 • EduTraceSys como objeto de aprendizagem na remediação de erros (TRIBECK, 2017)
- 2018 • Adaptabilidade de Objetos de Aprendizagem usando Calibragem e Sequenciamento Adaptativo de Exercícios (SILVA, 2018)
- 2019 • Visualização de Objetos de Aprendizagem para a Ferramenta de autoria FARMA (RAMOS, 2019)
- 2021 • Mudanças nas ideias prévias sobre ponto, reta e plano por meio da interação com a ferramenta FARMA (ARRUDA, 2021)
- 2022 • Desenvolvimento do Módulo de Estatísticas da Ferramenta de Autoria de Objetos de Aprendizagem FARMA (SANTOS, 2018)
- 2023 • Desenvolvimento do Módulo de Estatísticas da Ferramenta de Autoria de Objetos de Aprendizagem FARMA (VITEK, 2023)
- 2024 • Objetos de Aprendizagem com Feedbacks para a Autorregulação da Aprendizagem de Conceitos Matemáticos Necessários para o Cálculo Diferencial e Integral (CASTILHO, 2024)
- 2024 • Implementação de Uma Api Para Visualização E Interação Com Objetos De Aprendizagem Para Ferramenta De Autoria Farma (BONIN, 2024)

Além das contribuições históricas, a FARMA passou por várias versões, fruto de mestrados, doutorados e trabalhos de conclusão de curso, como FARMA ALG [2], FARMA CALC [3] e FARMA Reborn [4].

2.2 Objetos de Aprendizagem na FARMA

A FARMA tem como propósito principal disponibilizar um ambiente eficiente para o gerenciamento de erros dos estudantes, com ênfase no ensino de conceitos matemáticos. Na criação dos exercícios, são utilizados OA, de modo a proporcionar um ambiente mais flexível e adaptável à abordagem dos conteúdos. Dessa forma, professores e monitores de matemática podem ajustar o processo de ensino conforme as necessidades e objetivos específicos de cada aluno, favorecendo uma aprendizagem mais personalizada, dinâmica e eficaz.

Em um contexto geral, um OA pode ser qualquer tipo de mídia ou formato, desde algo simples até algo mais complexo. Na FARMA uma OA assume o papel de captar o erro de um estudante seguindo a seguinte estrutura:

[2] Disponível em http://farmaalg.c3sl.ufpr.br/users/sign_in
[3] Disponível em https://farma-calc.educacional.mat.br/
[4] Disponível em https://farma-reborn.educacional.mat.br/

1. **Introduções:** O principal objetivo da introdução é apresentar conceitos-chave que serão fundamentais para responder às questões propostas, utilizando imagens, exemplos e explicações para facilitar a compreensão. Dessa forma, a introdução pode ser organizada em partes menores, tornando o conteúdo mais acessível e favorecendo o entendimento do estudante.

2. **Exercícios**: Além da criação da introdução, o educador pode elaborar *exercícios* para avaliar o entendimento do aluno sobre o conteúdo apresentado. A estrutura de uma questão começa pelo título, que destaca o tema abordado. Em seguida, o enunciado explica o que é solicitado, podendo incluir imagens, links, textos ou outros recursos complementares para melhor contextualização.

3. **Passo de solução (questões)**: Com o enunciado do exercício definido, o educador pode elaborar as questões com o objetivo de verificar o nível de compreensão do estudante sobre o conteúdo trabalhado. Cada questão é composta por um título e um enunciado, que deve ser construído com base na descrição do exercício. Além disso, é possível incluir imagens e links que auxiliem na contextualização ou ilustração do problema proposto. A resposta esperada pode variar conforme o tipo de exercício, podendo ser um número, uma expressão aritmética ou uma expressão algébrica..

4. **Dicas (feedbacks)**: Após salvar a questão, é possível adicionar uma dica, elemento essencial para a remediação de erros dos estudantes. A dica pode ser configurada para aparecer após um determinado número de tentativas erradas, auxiliando o aluno no processo de aprendizagem.

Os OA podem ser segmentados em pequenos blocos, facilitando a assimilação do conteúdo pelo aprendiz e proporcionando uma experiência de aprendizado mais clara e eficaz. Na Figura 1 pode ser visualizado um esquema que demostra o fluxo de criação de um OA.

2.3 A FARMA Reborn

Nesta seção, será apresentada a versão FARMA Reborn [5], um projeto intitulado “Redesign e Refatoração da FARMA”, desenvolvido entre 2015 e 2018, com a colaboração de estudantes do curso de Graduação em Sistemas para Internet. O objetivo deste projeto foi melhorar a ferramenta, otimizando suas estruturas internas e interfaces, que haviam sido criadas no trabalho de Santos (2018). Mesmo que a versão FARMA Reborn tenha sido descontinuada a partir da criação deste projeto em 2025, suas contribuições para a estrutura inicial foram fundamentais. O novo projeto manteve a base do projeto desenvolvida anteriormente, adaptando-a para uma arquitetura monolítica. Além disso, diversas telas e elementos de design serviram como

[5] Disponível em https://farmareborn.educacional.mat.br/

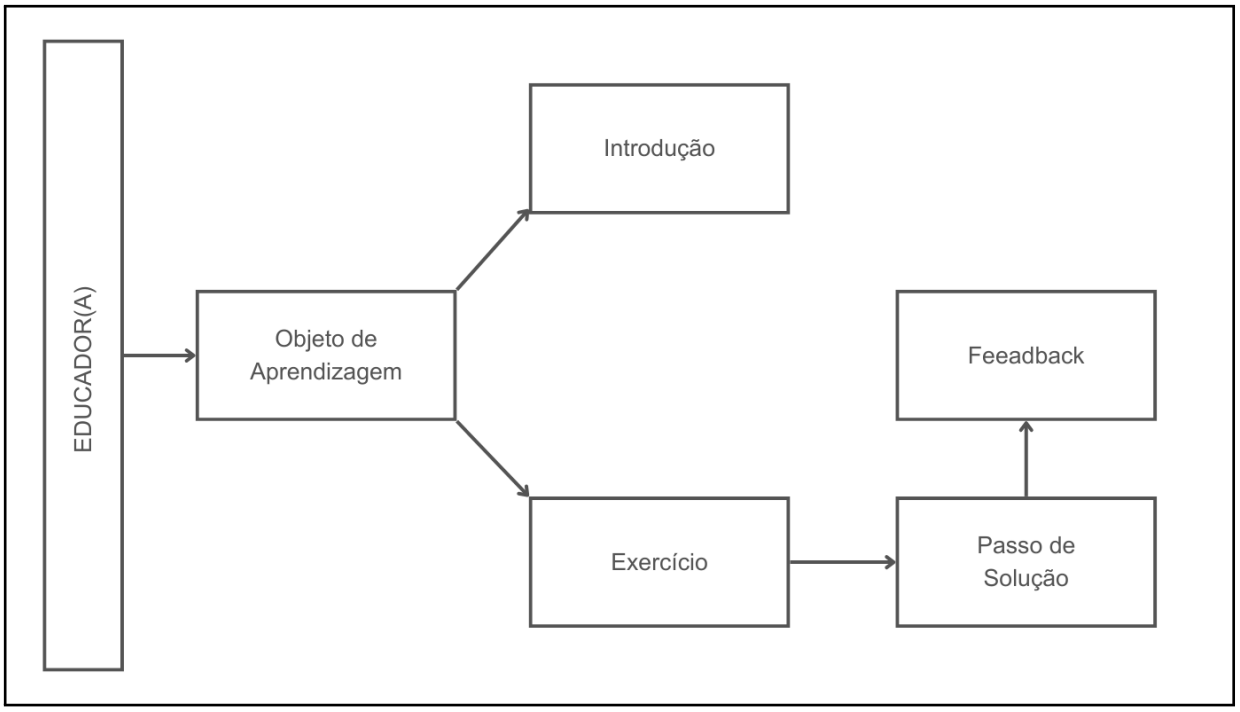

Figura 1 – Fluxo de criação de um Objeto de Aprendizagem.

referência e inspiração, influenciando diretamente a construção das interfaces apresentadas neste trabalho.

A Figura 2 exibe a página principal desta versão. Essa versão trouxe um design moderno, que serviu como base para o desenvolvimento deste trabalho.

Figura 2 – Página principal da FARMA Reborn

Na sequencia, são apresentadas as telas para a área de criação de OA. Na Figura 3, é possível observar a tela inicial para a visualização de um OA. Nesta tela, é possível observar a interface sem nenhum OA disponível. Para adicionar novos objetos de aprendizagem, o usuário deve selecionar o botão localizado no canto inferior direito, que aplica o conceito de Floating

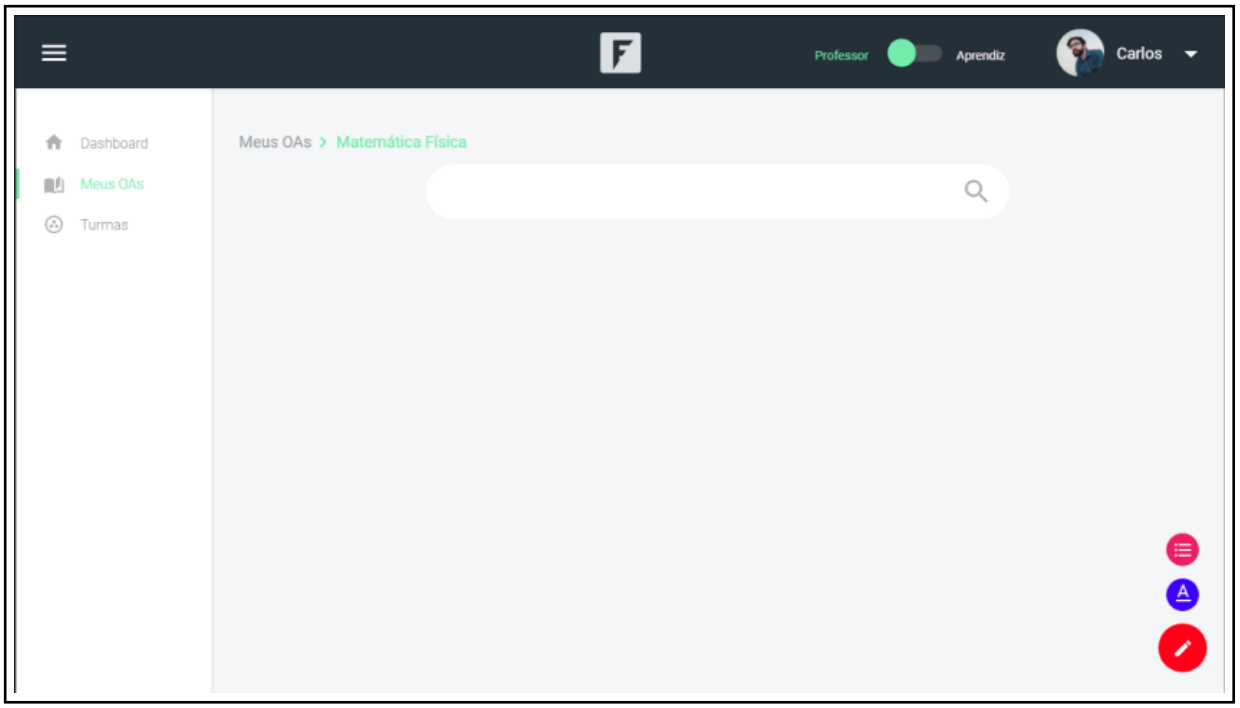

Figura 3 – Tela para visualização de OA FARMA Reborn

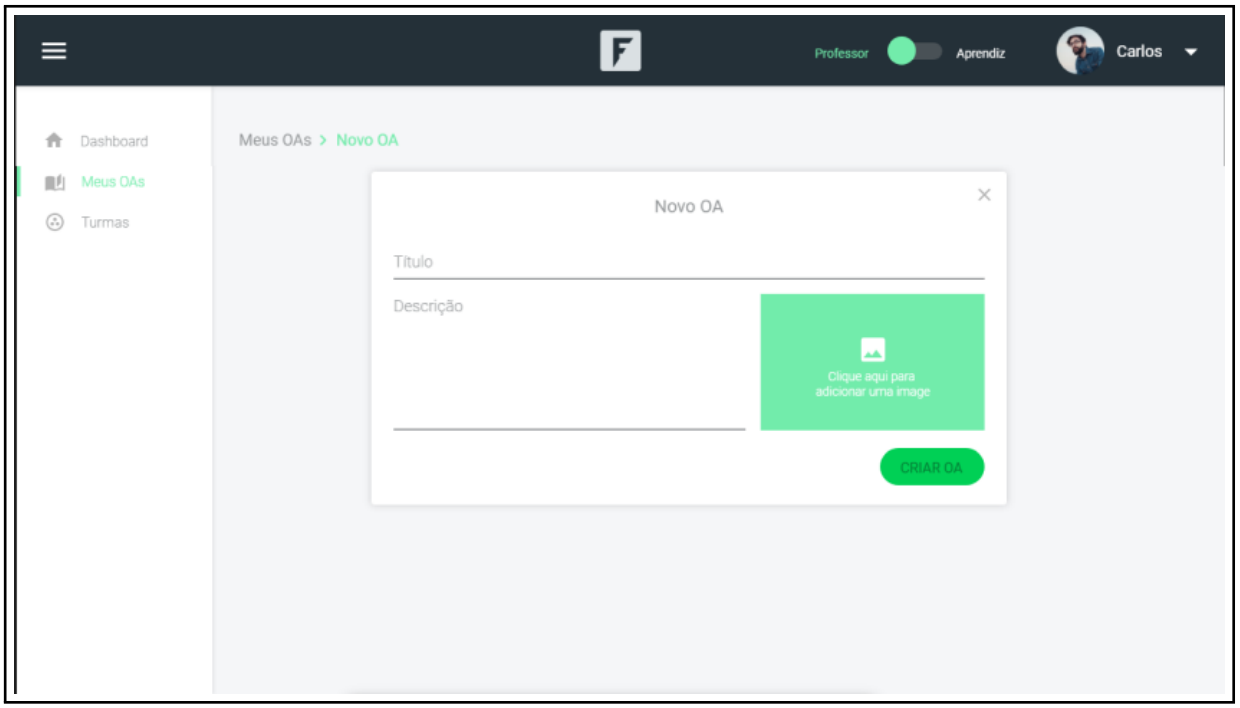

Figura 4 – Tela da area para criação de OA FARMA Reborn

Action Button (FAB). Ao ser acionado, o botão exibe dois ícones que representam os atalhos para as áreas de criação de Introdução e de Exercício.

Na Figura 4, é possível observar a tela de criação de um OA. Para realizar o cadastro, é necessário inserir um título para o OA, uma descrição e, opcionalmente, uma imagem — que pode ser pré-visualizada no canto direito do campo de descrição. Após o preenchimento do formulário, o usuário deve clicar no botão “Criar OA” para concluir o processo de criação do OA.

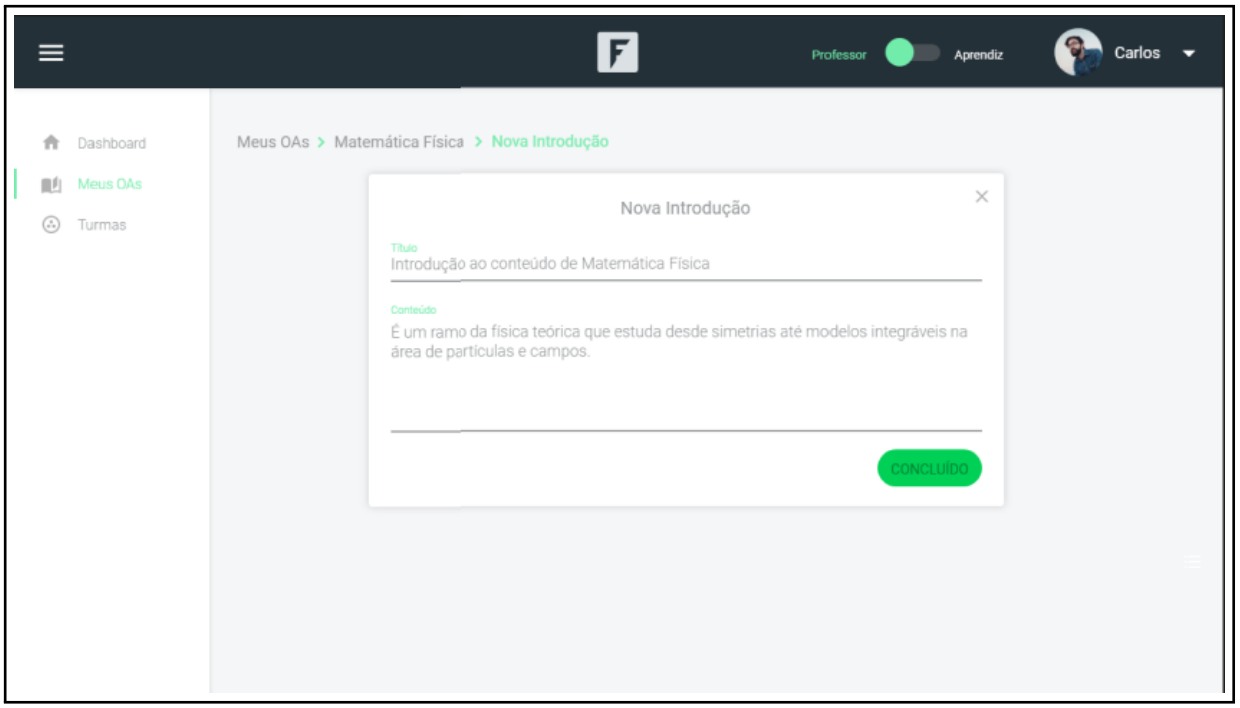

Figura 5 – Tela para criação de Introdução de um OA na FARMA Reborn

Na Figura 5, é possível observar a tela de criação de uma introdução em um OA. Para realizar esse processo, é necessário inserir um título, seguido da descrição obrigatoriamente. Em seguida, o educador deve selecionar o botão “Concluído”, finalizando assim a criação da introdução do OA.

Na Figura 6, é apresentada a tela de criação de um novo exercício dentro de um OA. O processo de criação segue uma estrutura semelhante à utilizada na introdução, sendo obrigatório o preenchimento dos campos de título e descrição. Por fim, deve-se selecionar o botão “Concluído” para finalizar o processo de criação do exercício.

Na Figura 7, é possível observar a tela de criação de um novo passo de solução/questão de um OA. Assim como nas telas anteriores, o educador deve inserir um título e uma descrição — campos obrigatórios —, seguidos da resposta esperada e da opção “Considerar ordem da resposta”, representada por um checkbox. Essa opção permite verificar se o estudante seguiu corretamente a sequência esperada na resolução do exercício. Por fim, deve-se selecionar o botão “Criar questão”, que realiza o envio do passo de solução/questão.

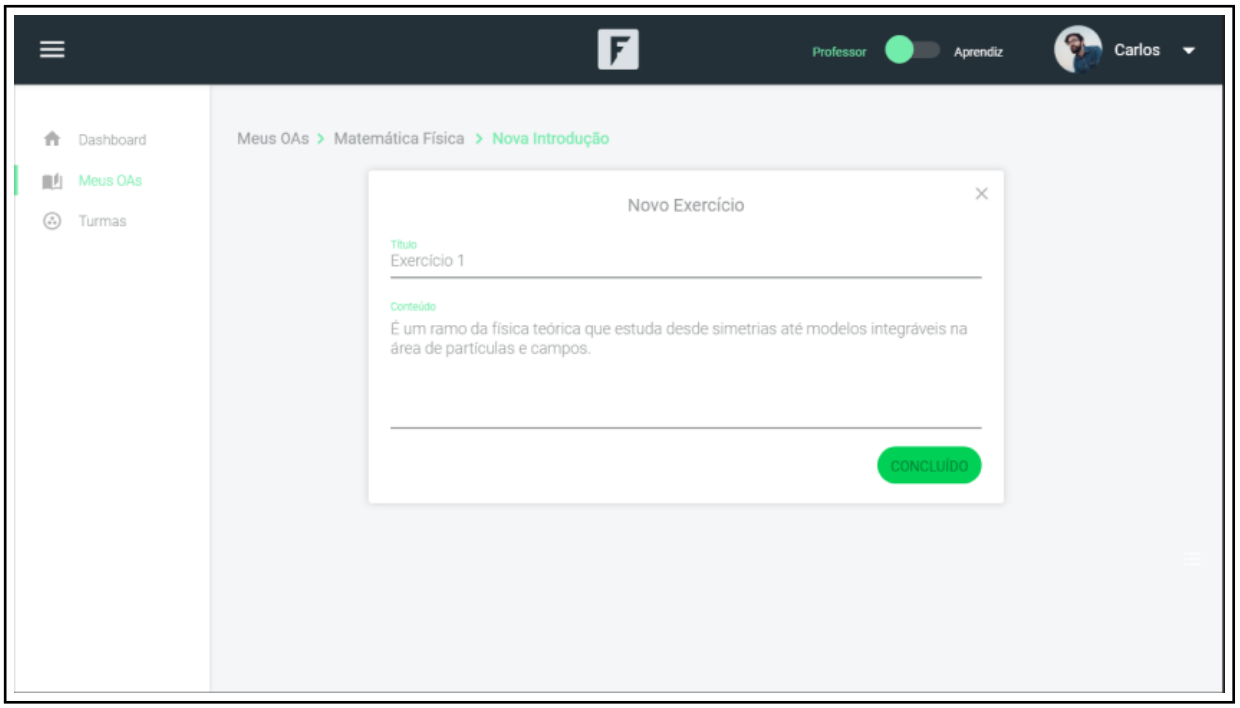

Figura 6 – Página para criação de Exercicios de um OA na FARMA Reborn

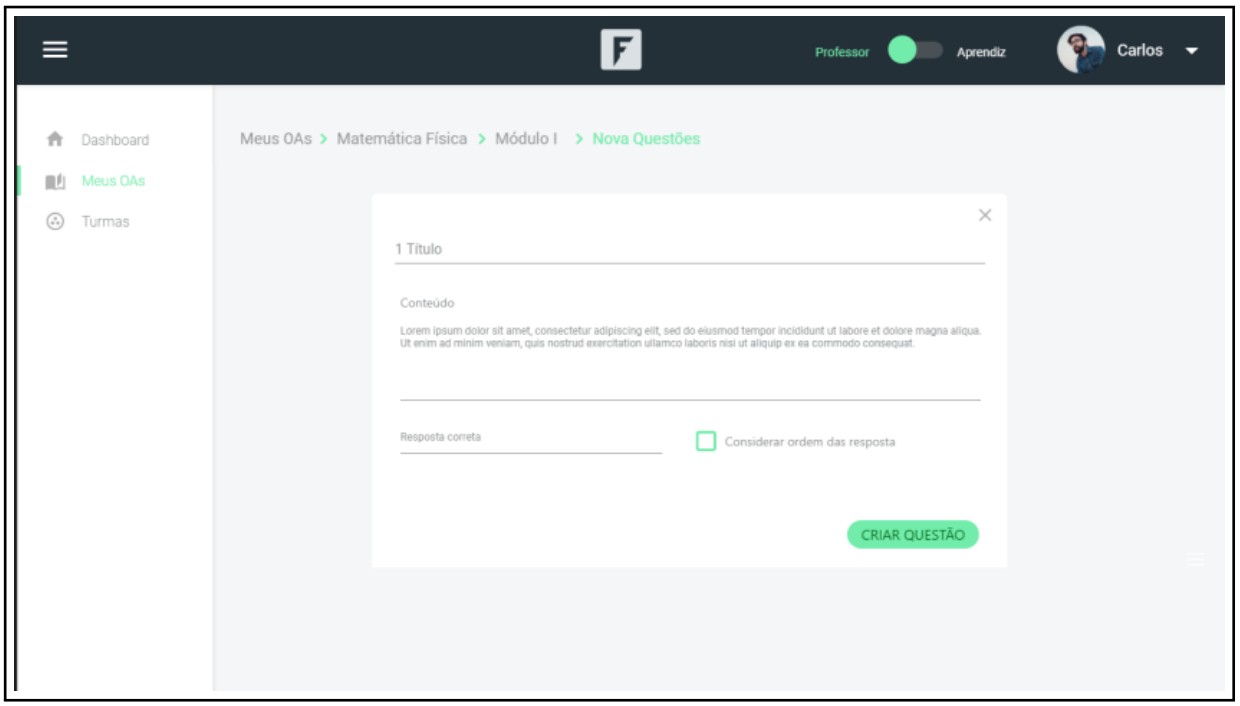

Figura 7 – Página para criação de Questões de um Objetos de Aprendizagem FARMA Reborn

3 MATERIAIS E MÉTODOS

Este projeto teve o objetivo desenvolver uma interface gráfica para a criação de OA, seguindo uma arquitetura monolítica, buscando alternativas de tecnologias diferentes das abordadas no trabalho de Vitek (2023).

Para o desenvolvimento do *Frontend* no trabalho anterior foi utilizado *React Next*, com auxilio do *Framework CSS Tailwind*, opções essas, que trouxeram dificuldades em seu trabalho por conta de sua complexidade e uso de uma API para renderização de dados dinâmicos. Após analisar as dificuldades apresentadas em Vitek (2023), foi avaliada alternativas tecnológicas para o desenvolvimento da nova interface gráfica para criação de OA, com o intuito de se beneficias de uma arquitetura monolítica.

3.1 Materiais

Neste projeto, foram aplicadas as seguintes ferramentas e tecnologias que abrangem desde o design até a implementação e o gerenciamento do código. A seleção desses recursos tem como objetivo otimizar o processo de desenvolvimento e assegurar a qualidade do produto final.

- **ClickUP** [1]: Software de gestão de projetos que oferece recursos como tarefas, bate-papo, quadros brancos, planilhas e colaboração em documentos, tudo integrado em uma plataforma única.
- **Figma** [2]: Ferramenta para a criação de protótipos interativos das telas da interface, possibilitando a visualização e testes do fluxo de navegação e da experiência do usuário antes da implementação.
- **VsCode** [3]: É um editor de código-fonte aberto, leve e altamente extensível, desenvolvido pela Microsoft. Ele oferece suporte a diversas linguagens de programação, possui recursos avançados como depuração integrada, controle de versão com Git, realce de sintaxe, IntelliSense para autocompletar código e um vasto ecossistema de extensões.
- **Tailwindcss** [4]: É um framework de CSS utilitário que permite construir interfaces modernas de forma rápida, diretamente noHTML.
- **Postgresql** [5]: É um sistema de banco de dados objeto-relacional de código aberto, reconhecido por sua confiabilidade, robustez de recursos e alto desempenho.

[1] Disponível em https://clickup.com/pt-BR
[2] Disponível em https://www.figma.com/pt-br/
[3] Disponível em https://code.visualstudio.com/
[4] Disponível em https://tailwindcss.com/
[5] Disponível em hhttps://www.postgresql.org/

- **Docker** [6]: É um conjunto de soluções de Plataformas que empregam virtualização no nível do sistema operacional para distribuir software em unidades chamadas contêineres.
- **Ruby on Rails** [7]: Framework web completo e poderoso que acelera o desenvolvimento de aplicações web, utilizando a linguagem *Ruby*. Facilita a criação de código bem estruturado, escalável e de fácil manutenção.
- **Hotwire** [8]: Framework padrão do Rails para criação de aplicações web interativas. Ele combina o Turbo(navegação dinâmica) e Estimulus(interação no FrontEnd).
- **GitHub** [9]: Plataforma de hospedagem de código-fonte e colaboração para projetos de software. Possibilita o controle de versão do código, facilitando o acompanhamento das alterações, a colaboração entre desenvolvedores e a resolução de conflitos.
- **Git** [10]: É um sistema de controle de versão distribuído, gratuito e de código aberto, projetado para gerenciar projetos de qualquer tamanho com rapidez e eficiência.

3.2 Métodos

Para execução desse projeto de uma nova interface gráfica para criação de OA, foi proposto os seguintes passos para desenvolvimento:

3.2.1 Levantamento de requisitos

Antes de iniciar o desenvolvimento, foi fundamental compreender o sistema para que pudéssemos levantar os requisitos necessários. Esse entendimento foi crucial para atender às demandas de desenvolvimento da FARMA. Para organizar esses requisitos e o progresso do desenvolvimento, foi utilizado o quadro *Kanban*, uma ferramenta de gestão de projetos que facilita a visualização do trabalho e otimiza a eficiência. O quadro *Kanban* define as seguintes etapas do processo: No *Backlog*, as tarefas aguardam priorização e inclusão no fluxo de trabalho ativo; *Ready*, são tarefas que estão prontas para serem iniciadas; *In Progress*, tarefas em fase de desenvolvimento; *In Review*, tarefas em fase de revisão dos responsáveis e, *Done* que são tarefas finalizadas.

Dessa forma, cada requisito foi subdividido em tarefas menores, o que facilita o gerenciamento e o acompanhamento do progresso. Na Figura 8, é possível observar a representação

[6] Disponível em https://www.docker.com/
[7] Disponível em https://rubyonrails.org/
[8] Disponível em https://www.hotrails.dev/
[9] Disponível em https://github.com/
[10] Disponível em https://git-scm.com/

dessa estrutura por meio da ferramenta ClickUp, que será utilizada para o gerenciamento das tarefas deste projeto.

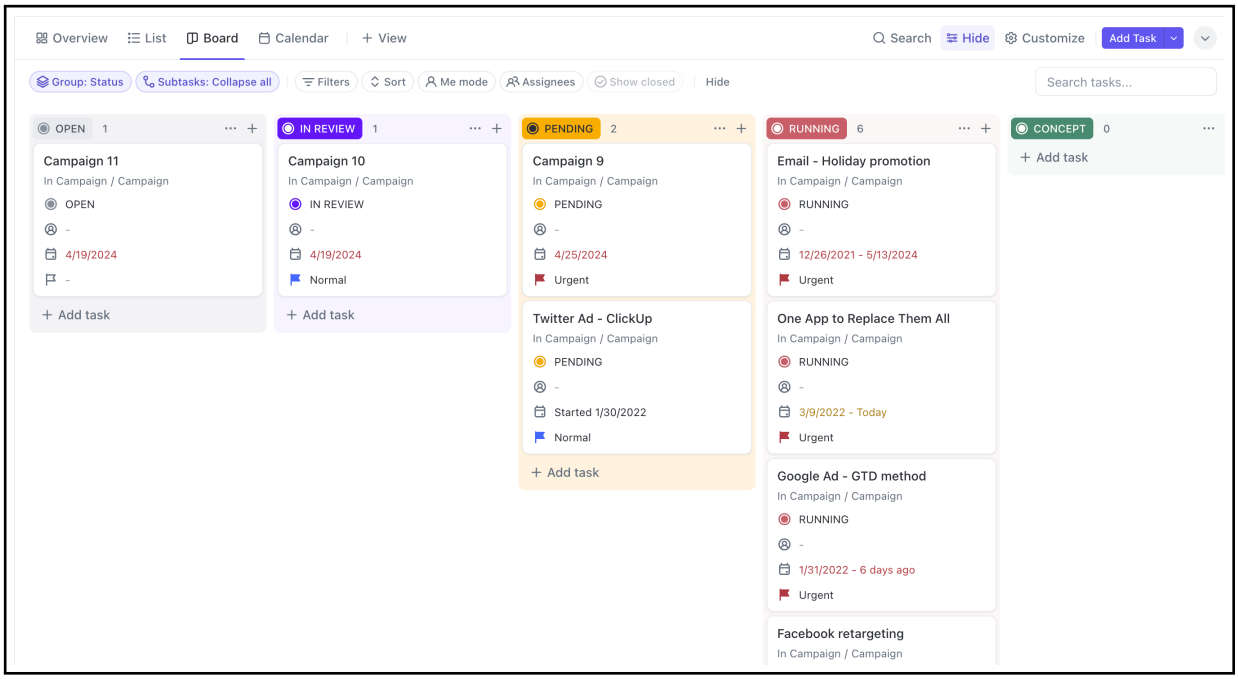

Figura 8 – Quadro Kanban no ClickUP

3.2.2 Definição de histórias e cenários

No processo inicial de levantamento de requisitos, o Backlog foi utilizado para inserir as histórias de usuário, com o objetivo de definir as tarefas a serem desenvolvidas, baseadas nas necessidades dos usuários. O termo "história de usuário" refere-se a uma funcionalidade específica, que pode incluir um ou mais cenários que demonstram diferentes formas de utilizá-la. Para construir essas histórias, utilizamos a estrutura padrão: **Dado** para identificar o estado atual, **Quando** para indicar a ação a ser realizada e **Então** para descrever a consequência dessa ação. Com essa abordagem, podemos extrair as tarefas necessárias para dar continuidade ao desenvolvimento. Na Figura 9, podemos observar um modelo de historia de usuário.

Como educador, **quero** criar um novo OA **para** que eu possa disponibilizar conteúdos interativos para os alunos.
Dado que sou um educador,
Quando eu acessar a plataforma e optar por criar um novo OA,
Então deverei ter a opção de inserir conteúdos interativos.

Figura 9 – Modelo de história de usuário

3.2.3 Prototipação para levantar os requisitos de interface do usuário

Foram desenvolvidas telas interativas que demonstram o funcionamento da criação de OA na FARMA. Seguindo a paleta de cores e a tipografia utilizadas em trabalhos anteriores, como por exemplo em (LARA, 2017). Para a criação dessas telas fpo ultilizado a ferramenta *Figma*, para criação de interfaces web, mobile, entre outros. A prototipação interativa no *Figma* traz múltiplos benefícios para o desenvolvimento da nova plataforma. (AWARI., 2023), tais como:

- **Antecipação visual:** Permite que os *stakeholders (partes interresadas)* conheçam a interface da plataforma antes da implementação, promovendo uma comunicação mais clara e alinhamento de expectativas.
- **Avaliação de usabilidade:** Possibilita simular a experiência do usuário na plataforma, identificando dificuldades e validando soluções antes do desenvolvimento.
- **Ajustes e otimização:** Facilita a adaptação e melhoria do design com base no feedback dos stakeholders, assegurando que a plataforma atenda às demandas dos usuários.

3.2.4 Organização do fluxo de desenvolvimento

O GitLab Flow é um modelo de fluxo de trabalho que organiza as ramificações (branches) do código e facilita a colaboração entre desenvolvedores. Ele divide as branches em diferentes tipos:

- **Main:** Representa a versão estável do código em produção.
- **Staging:** Usada para testes finais antes da produção.
- **Production:** Espelha o ambiente de produção e é usada exclusivamente para implantações.
- **Feature:** Criadas a partir de Main para desenvolver novas funcionalidades.
- **Hotfix/BugFix:** Criadas a partir de Production (para Hotfix) ou Staging (para correção de bugs).

Esse modelo ajuda a estruturar o desenvolvimento e facilita a gestão de mudanças no código.

3.2.5 Verificação e Validação do Software

A verificação e a validação do software constituem etapas fundamentais no processo de desenvolvimento, garantindo que o sistema implementado atenda às especificações definidas e aos requisitos dos usuários. Esses processos asseguram tanto a conformidade do código com o projeto quanto o comportamento correto do sistema em um ambiente real de uso.

No contexto deste trabalho, a verificação e validação foram conduzidas por meio da execução de diferentes tipos de testes automatizados, cada um com foco em uma camada específica da aplicação:

- **Testes unitários:** Avaliam o funcionamento de partes isoladas do sistema, como modelos e regras de negócio, assegurando que cada unidade opere corretamente de forma independente.
- **Testes de controladores:** Verificam o comportamento das ações e fluxos definidos nos controladores, garantindo que as rotas, redirecionamentos e mensagens de retorno sejam executados conforme o esperado.
- **Testes de sistema:** Simulam a interação real do usuário com a aplicação, por meio da interface web, validando fluxos completos de uso e a integração entre os diversos componentes do sistema.

Esses níveis de teste complementam-se, promovendo uma cobertura abrangente sobre o comportamento do software — desde a verificação de unidades isoladas até a validação do sistema em funcionamento integrado. Tal abordagem contribui para assegurar a qualidade, a confiabilidade e a estabilidade da aplicação, em consonância com os princípios de desenvolvimento orientado a testes.

3.2.6 Entrega e Implantação

No desenvolvimento ágil, a metodologia CI/CD (Integração Contínua e Entrega Contínua) foi adotada para otimizar os ciclos de desenvolvimento, proporcionando maior velocidade e eficiência. O CI automatiza a integração de alterações no código-fonte em um repositório central, executando testes automatizados à medida que novas alterações são feitas. Já o CD automatiza a liberação do código validado para ambientes de *staging (teste)* ou produção. Isso permite lançamentos mais frequentes e seguros de novas funcionalidades e melhorias.

4 ANÁLISE E PROJETO

Nesta seção, serão apresentados os resultados preliminares alcançados. Inicialmente, serão descritas as histórias de usuário desenvolvidas com o objetivo de levantar os requisitos necessários para o desenvolvimento da área de criação de OA. A partir dessas informações, as telas foram elaboradas na ferramenta Figma, cada uma representando uma etapa do processo de criação de OA.

4.1 História de usuário

Para o desenvolvimento da nova versão da plataforma, as histórias de usuário coletadas e detalhadas a seguir representaram os requisitos funcionais do sistema. Essas histórias serviram como guia para o processo de desenvolvimento, assegurando que a plataforma atendesse de forma eficaz às necessidades e expectativas dos usuários.

4.1.1 Criação e configuração dos objeto de aprendizagem

As histórias de usuário a seguir ilustram como os educadores irão criar e configurar os OA dentro da plataforma FARMA.

Feature: Criação de um Objeto de Aprendizagem
Como educador, **quero** criar um novo OA **para** que eu possa disponibilizar conteúdos interativos para os alunos.
Dado que sou um educador
Quando eu acessar a plataforma e optar por criar um novo OA
Então deverei ter a opção de inserir conteúdos interativos.

Feature: Inserir titulo e descrição de um Objeto de Aprendizagem
Como educador, **quero** inserir um título e uma descrição no OA **para** contextualizar o conteúdo apresentado.
Dado que estou criando um OA
Quando eu preencher os campos de título e descrição
Então esses dados devem ser salvos e exibidos corretamente.

Feature: Inserir componentes gráficos em um Objeto de Aprendizagem

Como educador, **quero** adicionar imagens, vídeos e gráficos ao OA **para** tornar a aprendizagem mais visual e interativa.

Dado que estou editando um OA

Quando eu fizer o upload de mídias

Então elas devem ser armazenadas e exibidas corretamente no OA.

Feature: Salvar informações não finalizadas de um Objeto de Aprendizagem

Como educador, **quero** poder salvar o OA como rascunho **para** que eu possa editá-lo antes de publicá-lo.

Dado que estou criando ou editando um OA

Quando eu optar por salvar como rascunho

Então o sistema deve armazená-lo sem disponibilizá-lo para os alunos.

Feature: Editar informações de um Objeto de Aprendizagem

Como educador, **quero** editar um OA já criado **para** atualizar o conteúdo quando necessário.

Dado que tenho um OA salvo

Quando eu acessar a opção de edição e fizer alterações

Então as modificações devem ser aplicadas e salvas corretamente.

Feature: Excluir informações de um Objeto de Aprendizagem

Como educador, **quero** excluir um OA **para** aqueles casos que ele não seja mais relevante ou precise ser removido.

Dado que tenho um OA criado

Quando eu selecionar a opção de exclusão

Então o OA deve ser removido do sistema.

4.1.2 Criação de exercícios, introdução, passos de soluções e dicas

As histórias de usuário a seguir ilustram como os educadores irão criar as introduções, exercícios, passos de solução e dicas dentro da plataforma FARMA.

Feature: Criação de Introdução

Como educador, **quero** criar introdução dentro do OA **para** contextualizar e relembrar conceitos dos alunos.

Dado que estou criando uma Introdução

Quando eu acessar a opção de adicionar introdução

Então devo poder inserir e configurar a introdução dentro do OA.

Feature: Personalizar introdução

Como educador, **quero** personalizar a introdução criada **para** atender às necessidades específicas do conteúdo.

Dado que estou criando uma introdução

Quando eu editar o conteúdo da da introdução

Então as modificações devem ser salvas e exibidas corretamente para os alunos.

Feature: Criação de Exercicio

Como educador, **quero** criar exercícios dentro do OA **para** reforçar o aprendizado dos alunos.

Dado que estou criando um OA

Quando eu acessar a opção de adicionar exercícios

Então devo poder inserir e configurar questões dentro do OA.

Feature: Personalizar enunciado de exercicios

Como educador, **quero** personalizar os enunciados dos exercícios **para** atender às necessidades específicas do conteúdo.

Dado que estou criando um exercício

Quando eu editar o enunciado do exercício

Então as modificações devem ser salvas e exibidas corretamente para os alunos.

Feature: Adicionar Passo de solução
Como educador, **quero** adicionar passo de solução aos estudantes **para** avaliar seus conhecimentos a partir do enunciado do exercicio. **Dado** que estou criando um passo de solução **Quando** eu preencher o formulario de criação **Então** ele deve ser exibida na área de vizualização de passos de soluções criados

Feature: Adicionar dicas
Como educador, **quero** adicionar dicas às perguntas **para** ajudar os alunos a encontrar o caminho correto em caso de erro. **Dado** que estou configurando um exercício **Quando** eu inserir uma dica **Então** ela deve ser exibida após o aluno exceder o numero de tentativas.

Feature: Configurar numeros de tentativas para o acionamente de dicas
Como educador, **quero** configurar um número de tentativas por questão **para** que após exceder esse valor, aparecer uma dica. **Dado** que estou criando um exercício **Quando** eu definir um limite de tentativas **Então** o sistema deve apresentar uma dica para os alunos.

4.2 Prototipação de Telas

A seguir, são apresentadas as interfaces de usuário desenvolvidas com base na coleta dos requisitos de interface, conforme as histórias de usuários descritas no Seção 4.1, além de sugestões de interfaces para trabalhos futuros.

4.2.1 Interface de criação de Objeto de Aprendizagem

A seguir, serão apresentadas as interfaces desenvolvidas com base nas histórias de usuários, ilustrando o processo de criação do OA.

Na Figura 10, é possível visualizar a tela inicial, onde são exibidas as OA's criadas. No caso apresentado, como nenhuma OA foi criada, a mensagem "Nenhum OA foi encontrado"é exibida. Ao lado da mensagem, há um campo de busca, útil quando há um grande número de

OA's cadastradas e o usuário deseja encontrá-las rapidamente pelo nome. No canto inferior direito da tela, há um botão FAB para adicionar uma nova OA.

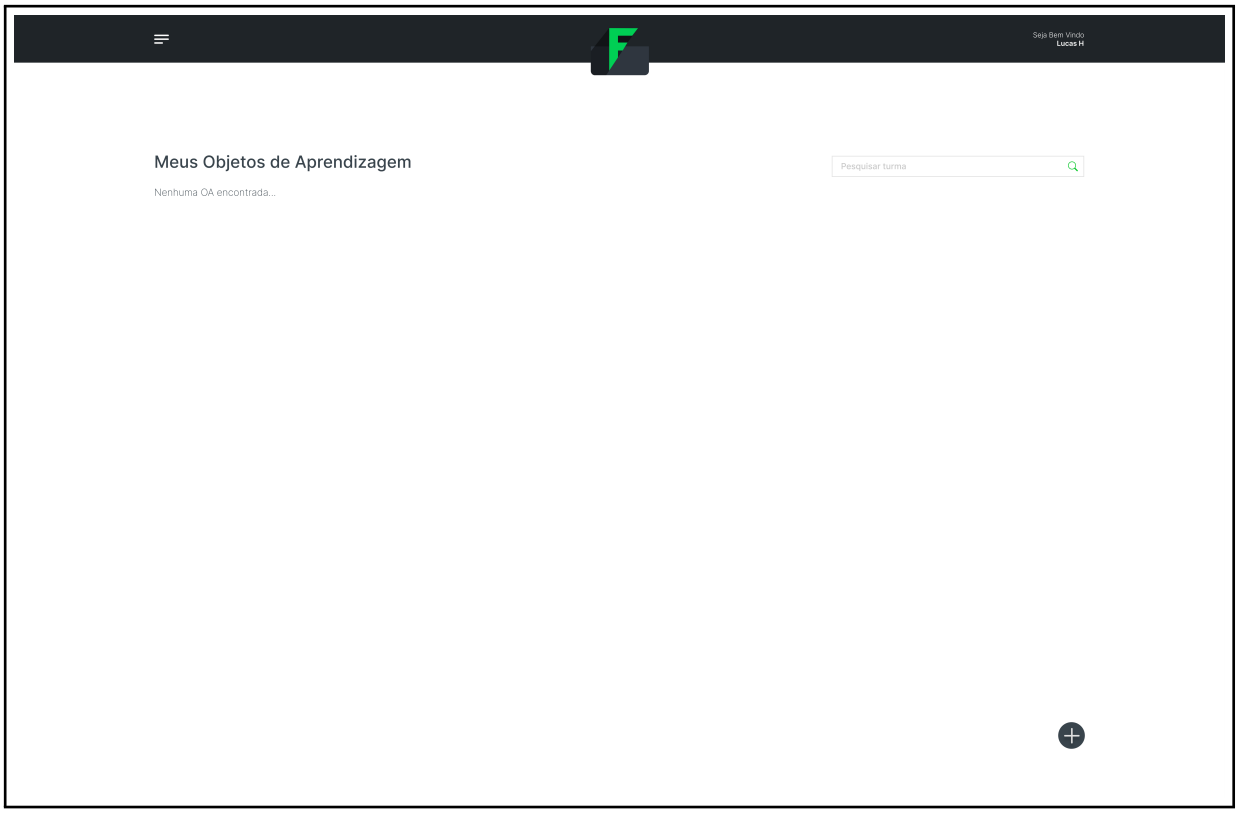

Figura 10 – Tela inicial para visualização de Objetos de Aprendizagem na FARMA (autoria própria)

Na Figura 11, é possível observar a tela de criação de uma OA. Nela, o usuário deve fornecer o título e a descrição da OA, podendo também adicionar uma imagem. Todos os campos, exceto a imagem, são obrigatórios. Na parte superior da tela, há duas opções de navegação: a seta à esquerda permite voltar à página anterior, enquanto o ícone de casa direciona o usuário para a tela inicial da FARMA.

Na Figura 12, vemos a mesma tela apresentada na Figura 10, porém agora com uma OA criada após passar pelo processo descrito na Figura 11. Nesta tela, é possível observar um componente exibindo a imagem inserida, o título e a descrição da OA. Na parte inferior direita, há um botão de acesso que permite visualizar a OA criada.

Na Figura 13, é possível visualizar a tela que exibe os componentes da OA criada. Neste exemplo, não há introduções, exercícios ou dicas cadastrados, portanto, a mensagem "Nenhuma informação encontrada"é exibida. Esses componentes podem ser adicionados utilizando os três botões localizados no canto inferior direito: o primeiro botão adiciona uma nova introdução, o segundo permite criar um novo exercício, e o último botão adiciona uma dica.

Nas Figuras 14, 15 e 16, observamos uma estrutura semelhante para a criação de diferentes componentes da OA. Na criação de uma introdução, é necessário adicionar um título, o conteúdo e uma imagem (*opcional*). Em seguida, na tela de criação de exercícios, é possível

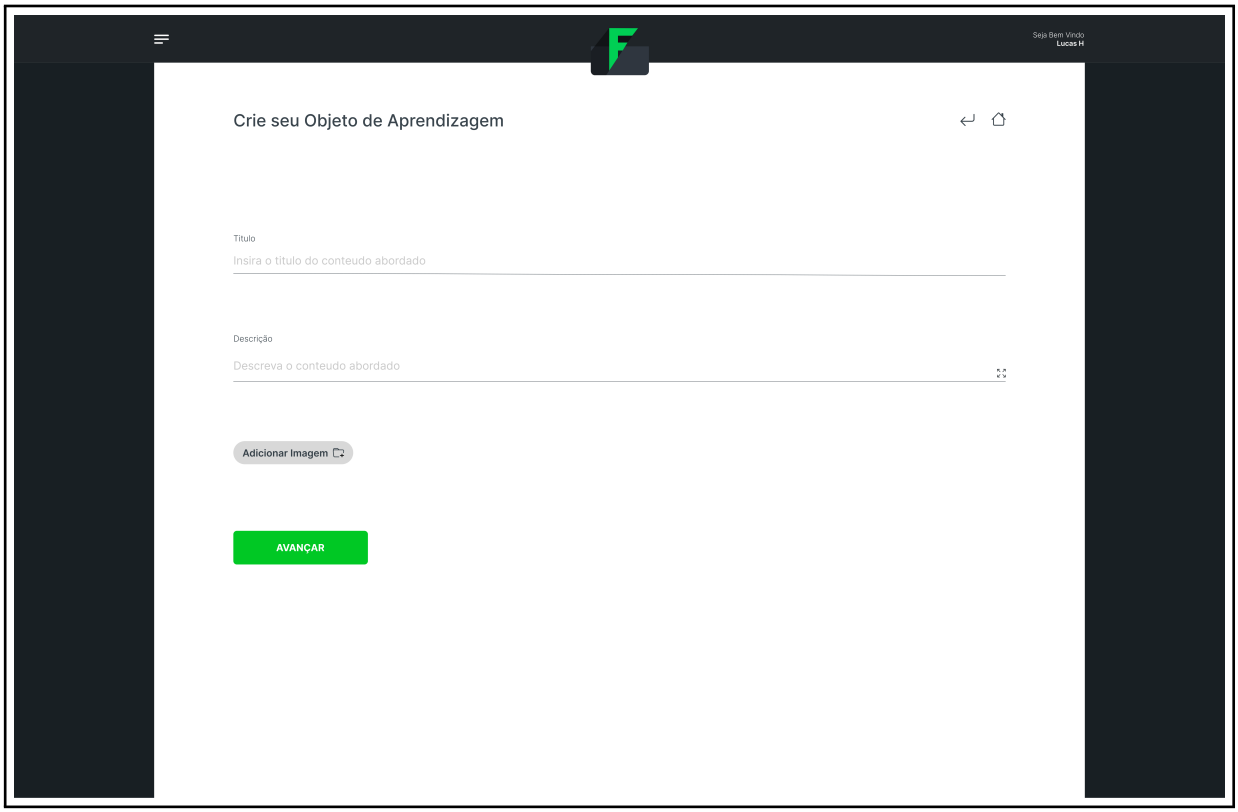

Figura 11 – Tela inicial para criação de Objetos de Aprendizagem na FARMA (autoria própria).

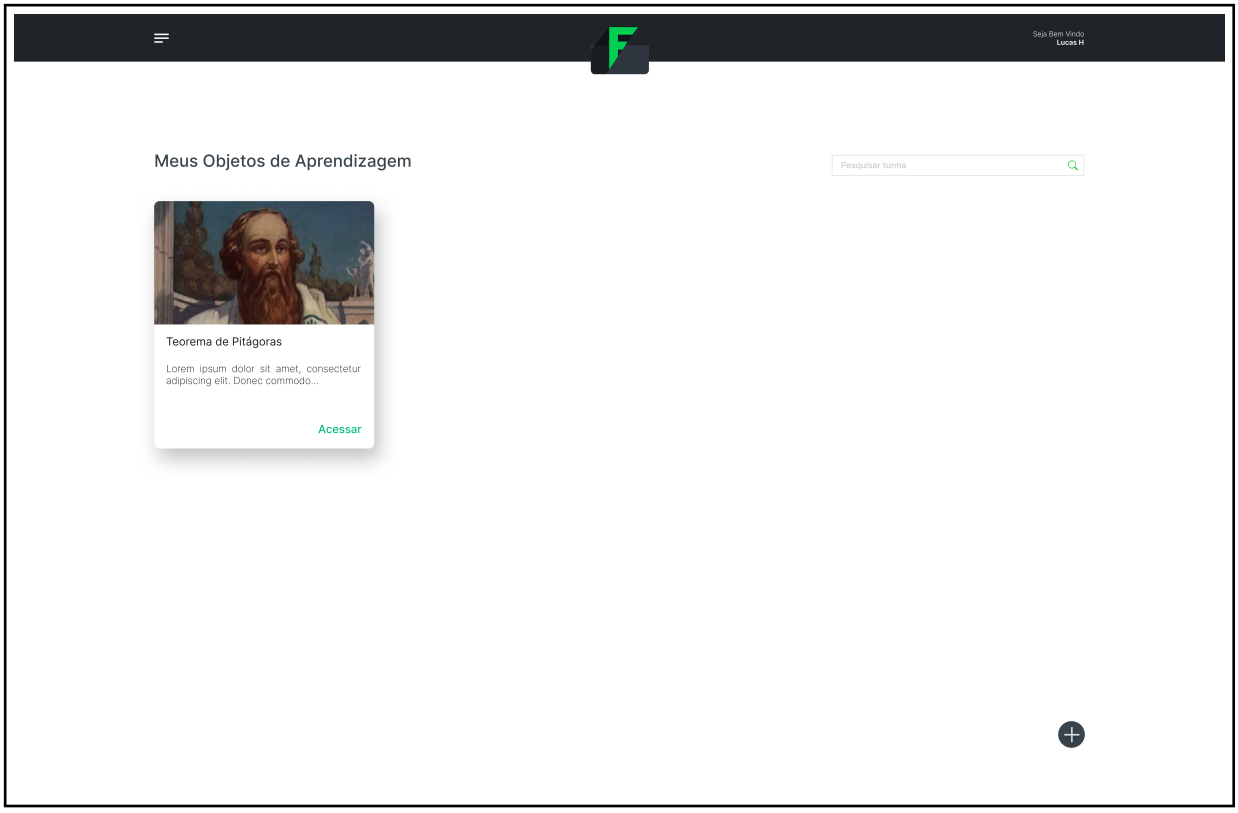

Figura 12 – Tela inicial para visualização de Objetos de Aprendizagem já criada na FARMA (autoria própria)

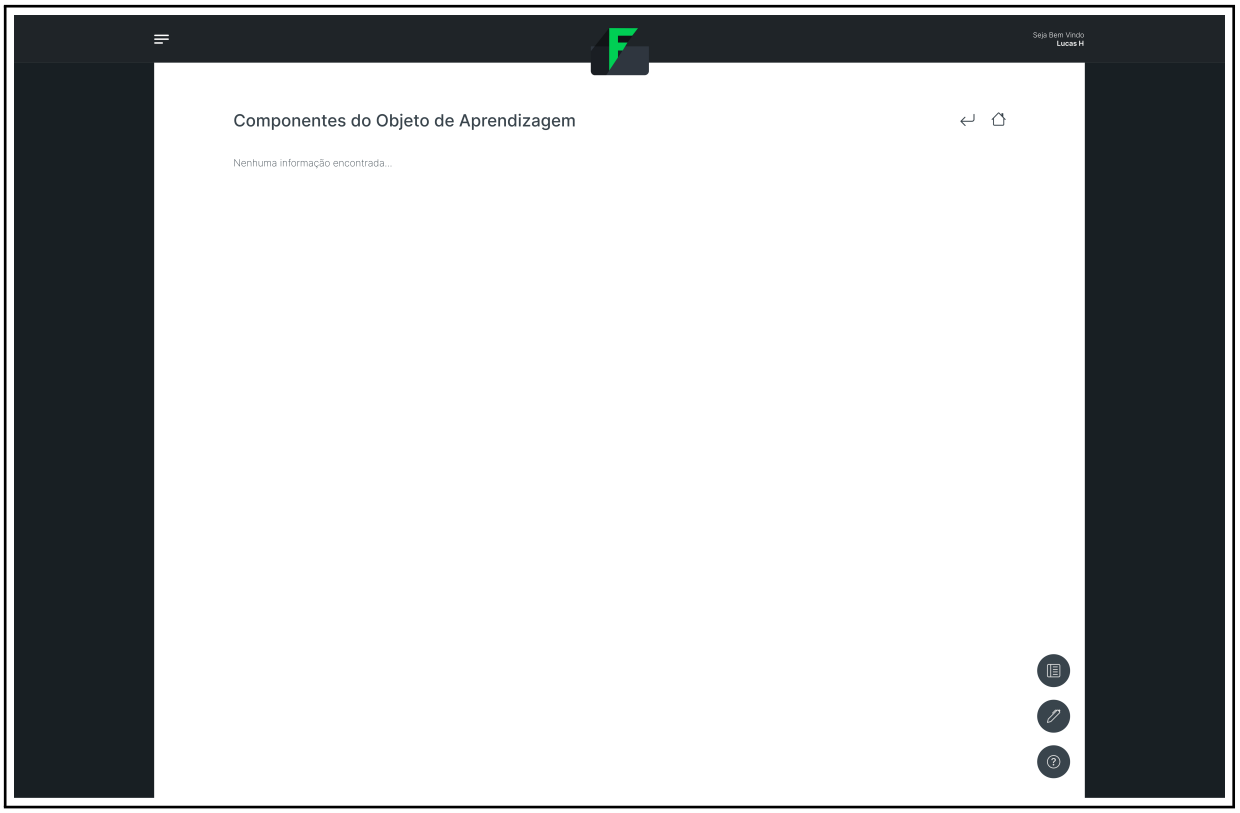

Figura 13 – Tela para visualização de componentes de um Objetos de Aprendizagem (autoria própria)

inserir um título, uma descrição do exercício, uma imagem opcional e, por fim, a resposta esperada. Por último, na tela de criação de dicas, o usuário pode adicionar o texto da dica e definir o número de tentativas necessárias para que a dica seja acionada.

Na Figura 17, vemos uma tela semelhante à Figura 13, mas com os componentes já criados. Agora, cada componente é exibido com seu título, descrição e opções específicas. No canto superior direito, o ícone à esquerda refere-se à opção de editar, enquanto o ícone à direita permite excluir o componente. Além disso, há um novo botão acima, simbolizado por um "olho", que serve para pré-visualizar o componente. Esta funcionalidade de pré-visualização pode ser vista nas próximas figuras.

Na Figura 18, é apresentada a tela de pré-visualização da interface do OA. Nessa área, o educador pode visualizar como o OA será exibido aos alunos, permitindo identificar possíveis falhas ou inconsistências antes de sua publicação. A pré-visualização inclui todos os elementos cadastrados, como introdução, exercício, passos de solução e dicas.

Na Figura 19, observa-se a tela de login do sistema. O layout apresenta um fundo escuro composto por linhas onduladas, que conferem um aspecto moderno à interface. Ao centro, encontra-se o formulário de autenticação, destacando visualmente a área destinada ao acesso do usuário. Esse formulário é composto pelos campos para inserção de e-mail e senha previamente cadastrados, seguido pelo botão para envio das credenciais. Além disso, há um link que permite redirecionar o usuário à página de recuperação de senha.

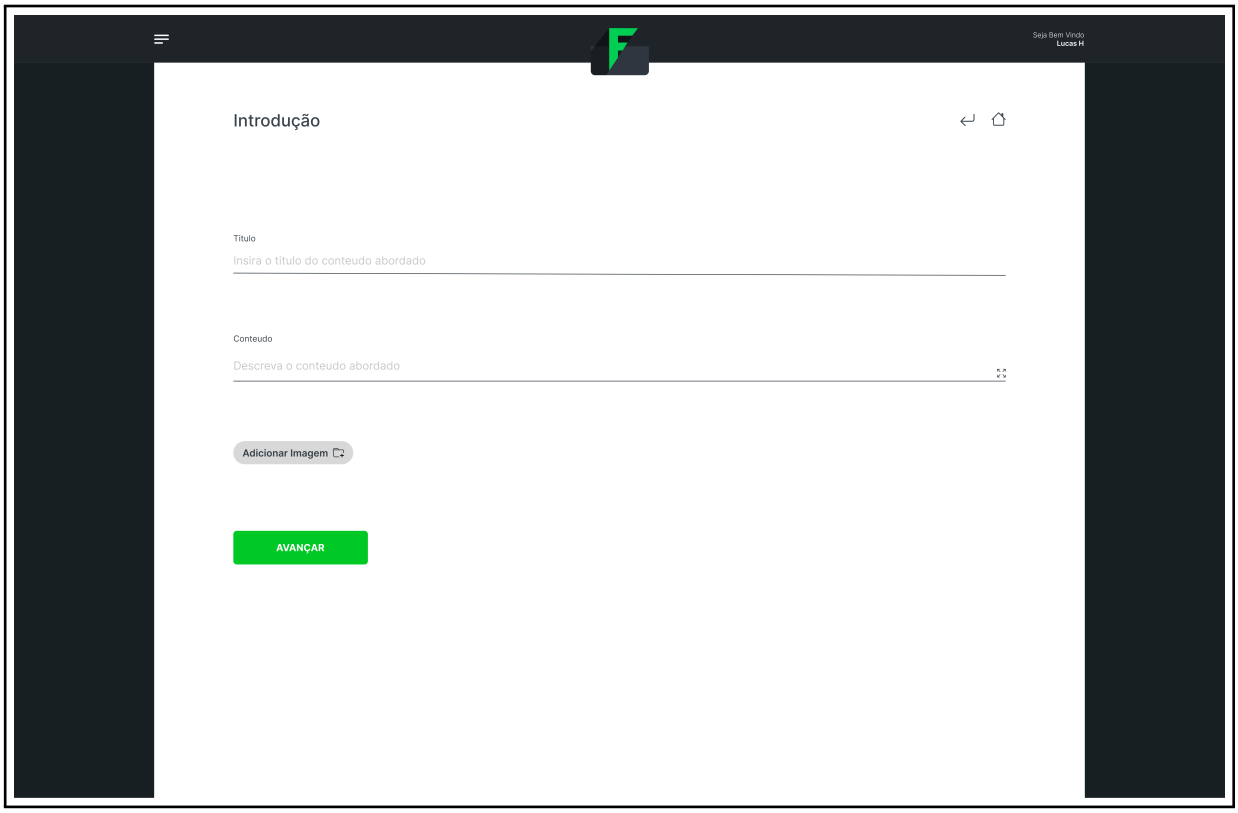

Figura 14 – Tela para criação de introdução de um Objetos de Aprendizagem (autoria própria)

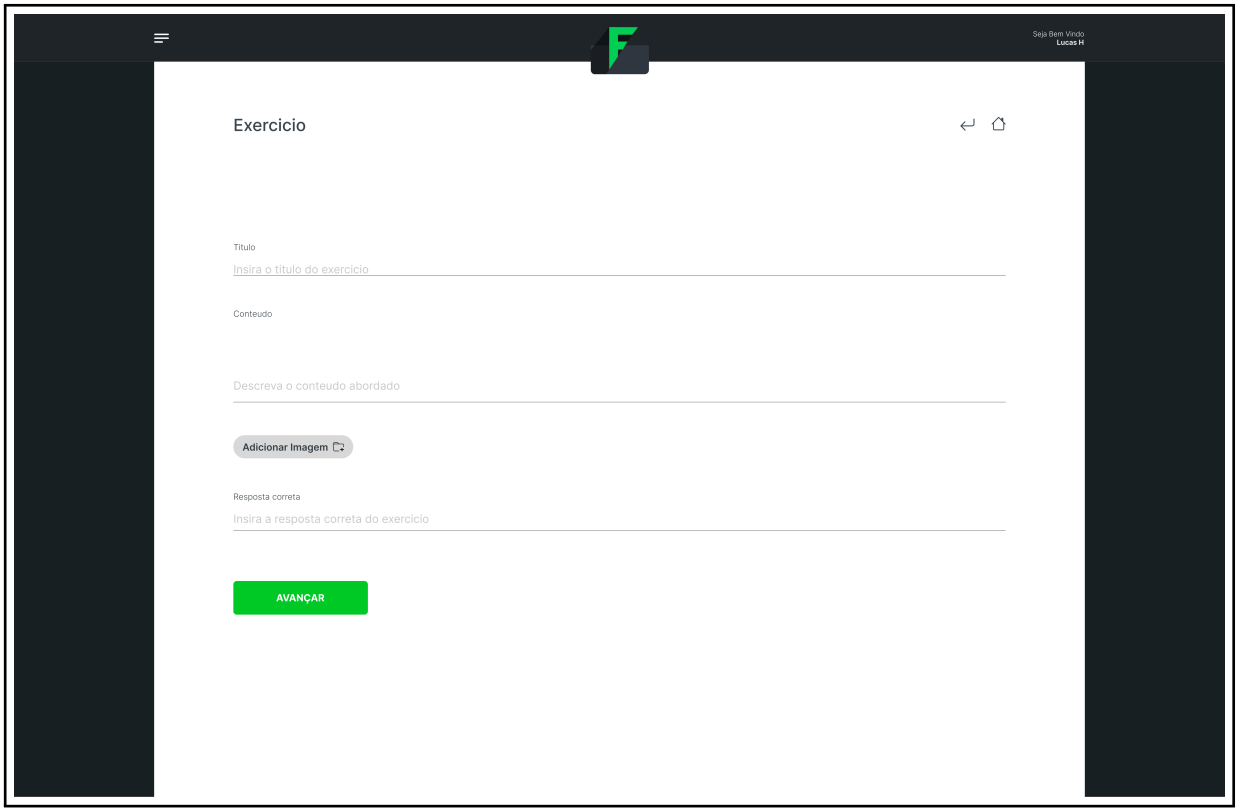

Figura 15 – Tela para criação de exercício de um Objetos de Aprendizagem (autoria própria)

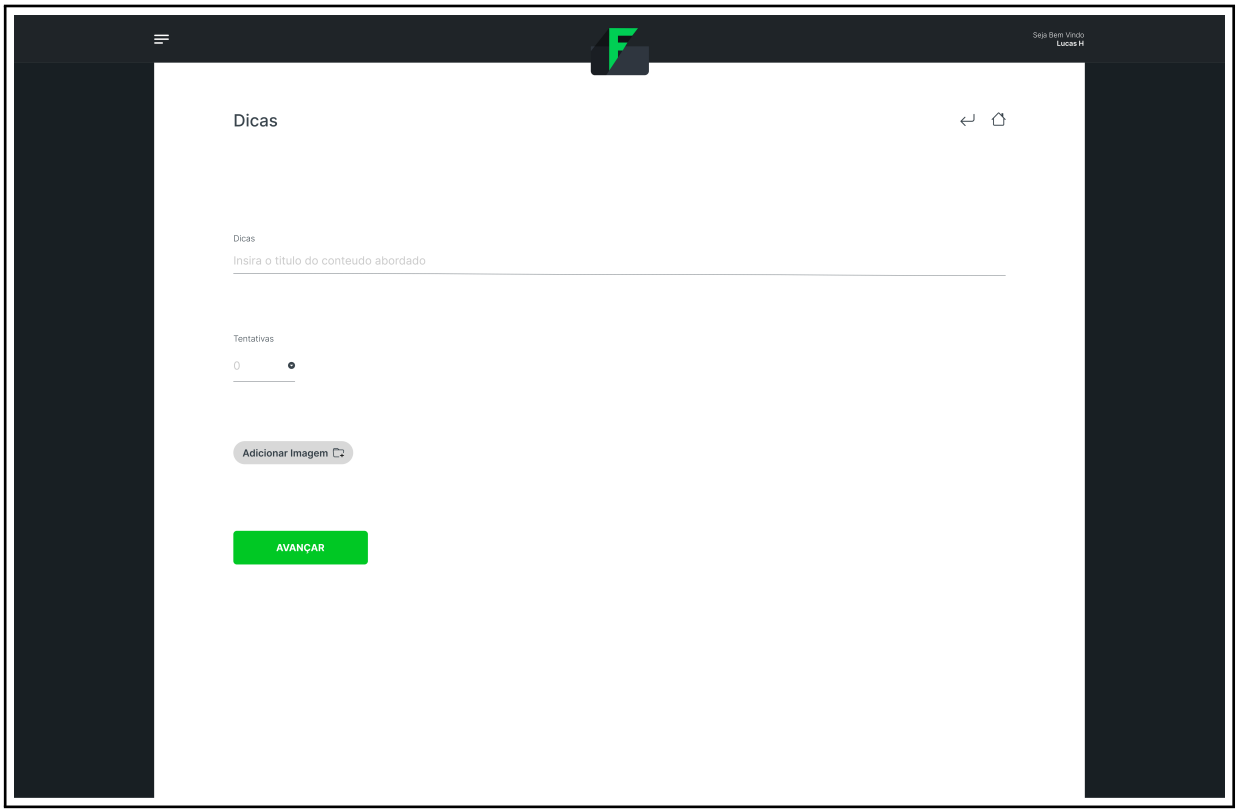

Figura 16 – Tela para criação de Dicas de um Objetos de Aprendizagem (autoria própria)

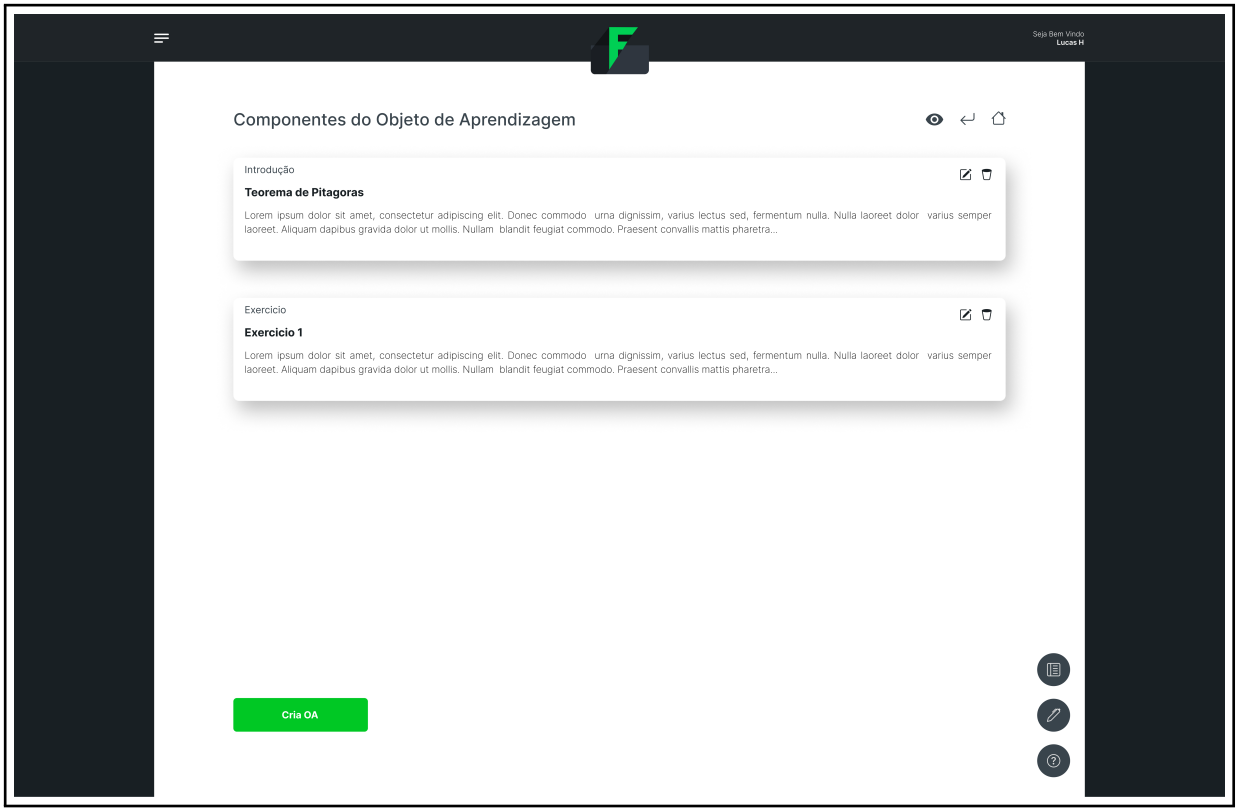

Figura 17 – Tela para visualização de componentes de um Objetos de Aprendizagem, agora criados (autoria própria)

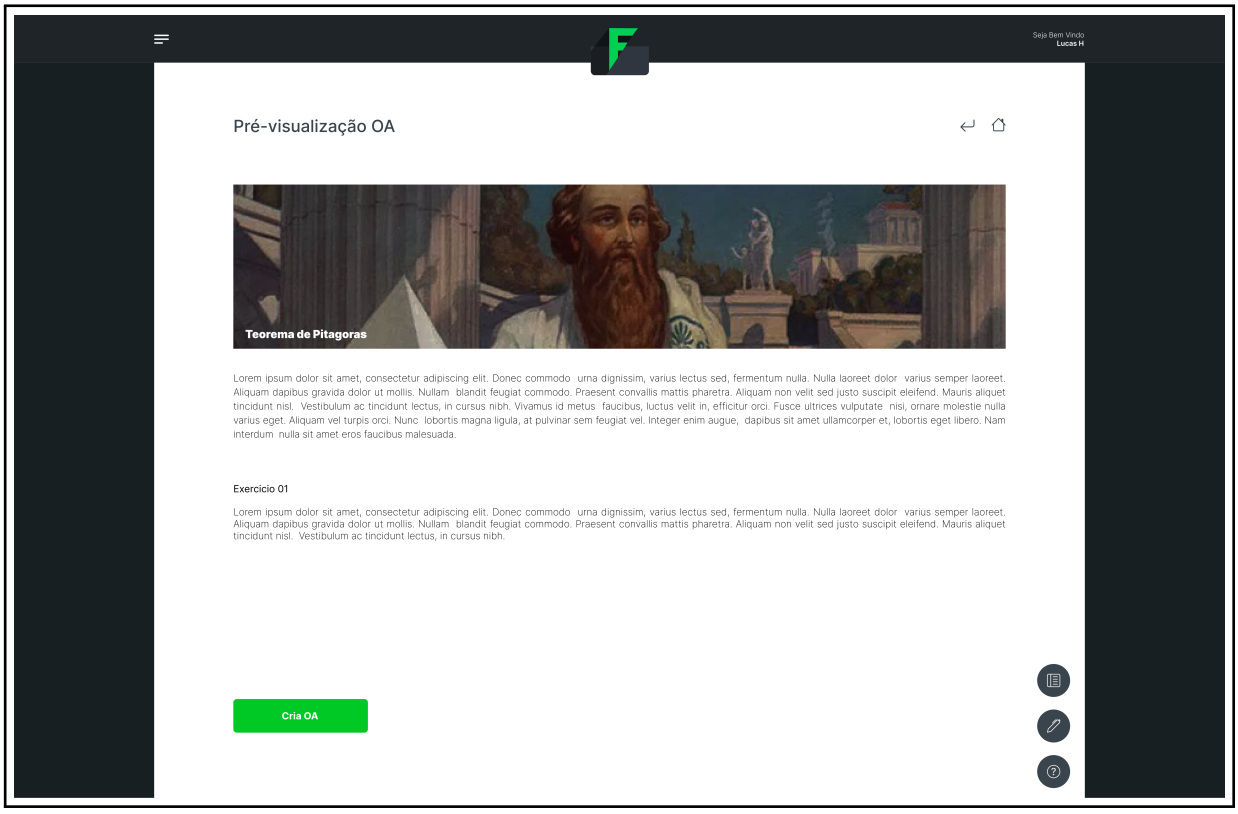

Figura 18 – Tela para pré-visualização do Objeto de Aprendizagem(autoria própria)

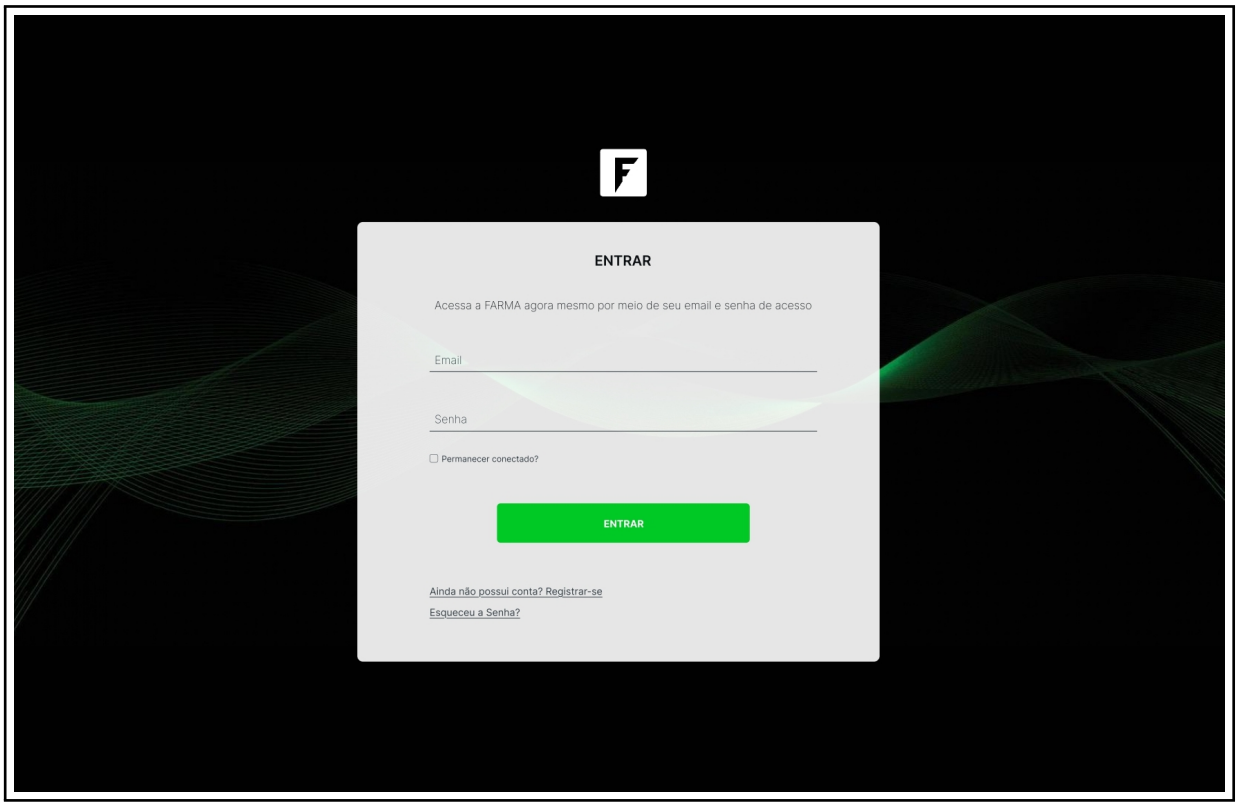

Figura 19 – Tela de login da nova versão da FARMA (autoria própria)

Na Figura 20, é apresentada a tela de registro da plataforma FARMA. A interface possui um fundo escuro composto por linhas onduladas, conferindo um aspecto moderno e coerente com a identidade visual do sistema. Ao centro, encontra-se o formulário de cadastro, estruturado para orientar o usuário no processo de criação de sua conta. O formulário é composto pelos campos: Nome Completo, E-mail, Senha e Repetir Senha, garantindo que o usuário informe seus dados corretamente e reduza erros de digitação. Logo abaixo, há o botão de ação “Registrar-se”, destacado em verde para indicar a continuidade do fluxo. Também são disponibilizados links para redirecionamento às páginas de login e recuperação de senha, oferecendo suporte aos usuários que já possuem conta ou que necessitam redefinir sua credencial de acesso.

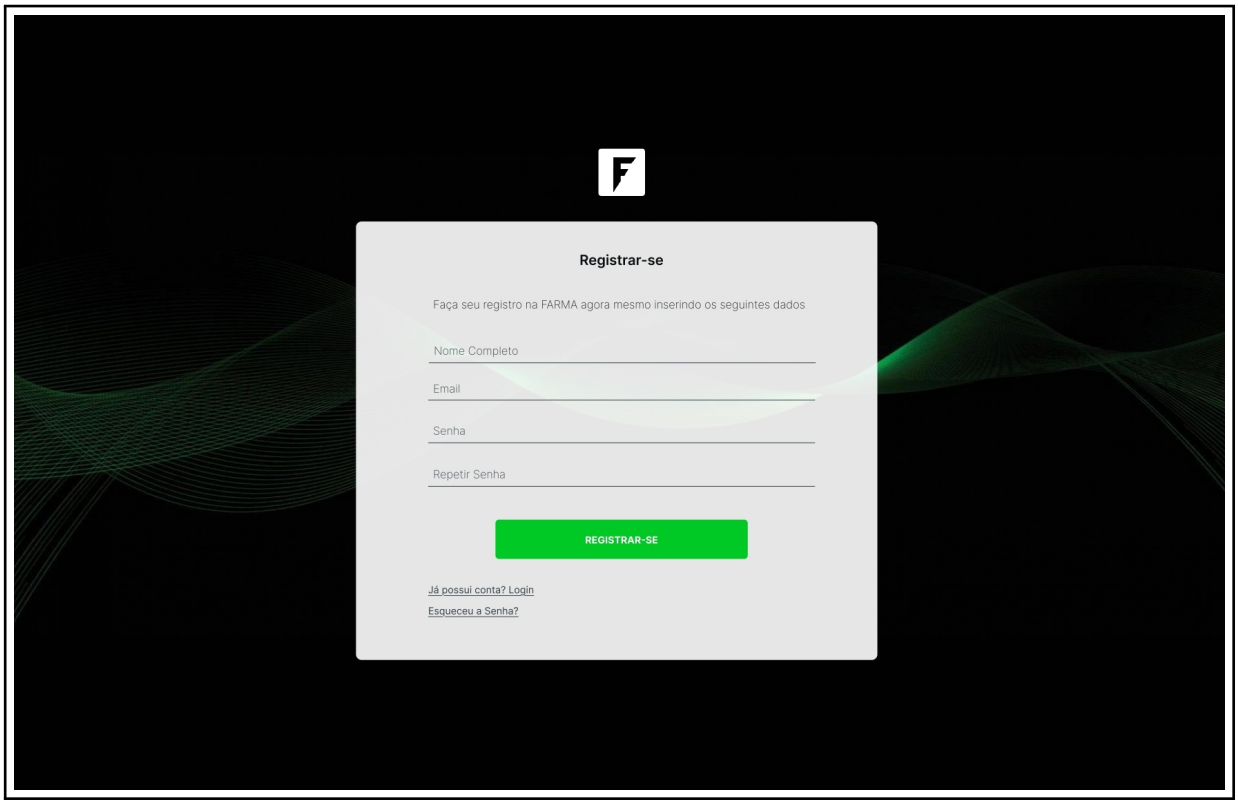

Figura 20 – Tela de criação de conta da nova versão da FARMA (autoria própria)

Na Figura 21, observa-se a página inicial da plataforma FARMA. A interface apresenta um layout dividido em duas seções: à esquerda, o conteúdo textual institucional; à direita, uma imagem ilustrativa que representa o contexto educacional e tecnológico do sistema. O destaque inicial é dado ao título “FARMA: Ferramenta de Autoria para a Remediação de Erros com Mobilidade na Aprendizagem”, seguido de um subtítulo que reforça sua proposta voltada à criação de objetos de aprendizagem para o ensino de Matemática. Logo abaixo, encontra-se o botão “Acessar”, que conduz o usuário ao ambiente de autenticação da ferramenta. Na parte superior, é exibido o menu compacto e o logotipo da FARMA, organizando a navegação de forma simples e objetiva.

Figura 21 – Tela home da nova versão da FARMA (autoria própria)

Por fim, a Figura 22 apresenta a proposta de tela de acesso à conta do usuário. Embora esse recurso ainda não tenha sido implementado na versão atual da plataforma, ele foi planejado como parte da evolução do sistema e poderá ser desenvolvido em trabalhos futuros, ampliando as funcionalidades disponíveis ao usuário.

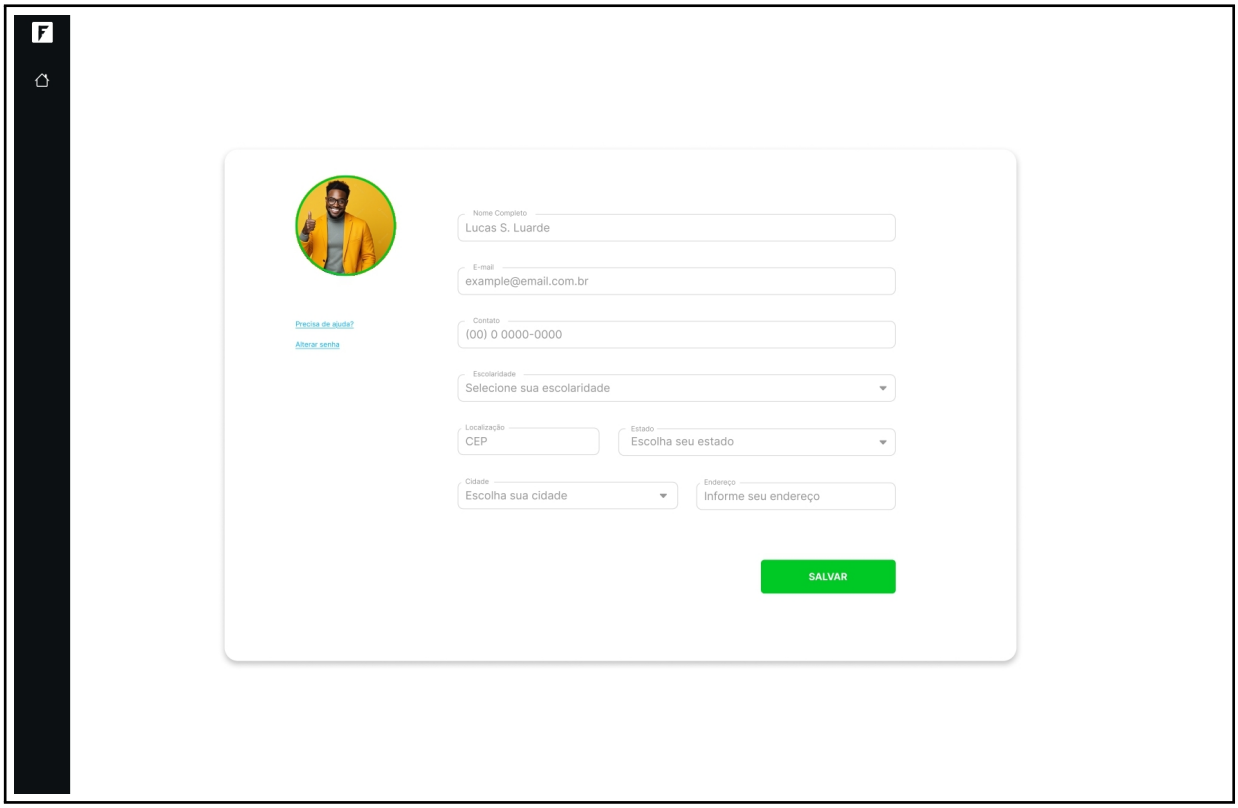

Figura 22 – Tela de acesso as informações de usuário da nova versão da FARMA (autoria própria)

5 DESENVOLVIMENTO DO SISTEMA

Neste capítulo, apresenta-se o processo de desenvolvimento do trabalho, descrevendo de forma detalhada as etapas de construção, implementação e testes. São discutidas as principais dificuldades enfrentadas e as modificações realizadas ao longo do processo, bem como as justificativas para os ajustes efetuados. Por fim, apresentam-se os resultados alcançados ao término de cada história implementada.

5.1 Escopo de Desenvolvimento

Com base na análise dos requisitos, nas histórias de usuário e na prototipação das telas, definiu-se o escopo inicial prioritário para a criação da nova interface gráfica da FARMA. Para isso, foi conduzida uma sessão colaborativa de pontuação entre orientadores e orientando, na qual cada funcionalidade foi avaliada de acordo com os seguintes critérios:

- Complexidade técnica — considerando dependências e integrações necessárias;
- Impacto operacional — avaliando o nível de urgência para retomada das atividades;
- Esforço de implementação.

Todas as funcionalidades foram subdivididas em tarefas menores. O desenvolvimento de cada uma seguiu três etapas principais: implementação da interface, integração da lógica de funcionamento e realização dos testes. As funcionalidades selecionadas para a primeira iteração receberam pontuações consensuais, estabelecendo um roteiro inicial de desenvolvimento, conforme Tabela 2.

Tabela 2 – Priorização de funcionalidades para desenvolvimento inicial

Funcionalidades	Pontuação
Criação de Objeto de Aprendizagem	5
Criação de Introdução do Objeto de Aprendizagem	4
Criação de Exercicios de Objeto de Aprendizagem	4
Criação de Passos de solução	4
Criação de dicas	4
Desenvolvimento da Página Inicial	4
Cadastro de usuario	5
Autenticação de usuario	5
Recuperação de senha de usuario	7
Seleção ao tipo de usuario	2
Integração TinyMCE (Editor de texto)	8
Visualização de informação do usuario	5

No entanto, algumas tarefas precisaram ser reavaliadas. Embora inicialmente estivessem previstas como contribuições para trabalhos futuros, durante o desenvolvimento tornou-se

necessário antecipar a implementação de determinadas funcionalidades consideradas prioritárias. Assim, essas tarefas foram incorporadas ao escopo e implementadas ainda neste trabalho (por exemplo: Página Inicial, Área de Autenticação e Área de Registro).

5.2 Configuração do TailwindCSS

Para o desenvolvimento da interface gráfica, utilizou-se a estrutura de um projeto previamente existente, originalmente configurado como uma API. Dessa forma, algumas configurações iniciais do ambiente já estavam implementadas. No entanto, como o objetivo deste trabalho era o desenvolvimento da interface visual do sistema, tornou-se necessária a configuração de ferramentas específicas para o *front-end*.[1]

Entre essas ferramentas, optou-se pela utilização do *framework* TailwindCSS[2], um utilitário CSS que possibilita a criação de interfaces modernas e responsivas de forma ágil, por meio da aplicação direta de classes CSS aos elementos HTML. A escolha pelo TailwindCSS justificou-se por sua flexibilidade, facilidade de personalização e compatibilidade com o *framework* Ruby on Rails.

A configuração do TailwindCSS seguiu as etapas descritas na documentação oficial da ferramenta, sendo implementada conforme os passos apresentados a seguir:

- **Instalação de tailwindcss:** Para a configuração primeiramente foi feito a instalação da *gem tailwind-rails* e em seguida a instalação do *TailwindCSS* no projeto, no qual gerou as configurações iniciais do projeto. Para isso seguir os comandos citados na documentação (ver Código 1):

```
#Instalação da gem tailwind-rails
./run bundle add tailwind-rails //version: 4.3

#Instalação do tailwind no projeto
./run rails tailwindcss:install //version: 4.1.11
```

Código 1 – Comandos de instalação Tailwind

- **Inclusão do arquivo de estilos na aplicação**: Para garantir o carregamento automático do `TailwindCSS` em todas as páginas, adicionou-se a seguinte linha no elemento <head> do layout principal (ver Código 2):

[1] Front-end: parte de um site ou aplicativo com a qual o usuário interage diretamente, incluindo elementos visuais como botões, menus, imagens e textos.

[2] Disponível em https://tailwindcss.com/

```
#farma/app/views/layout/shared/_head.html.erb
<head>
    <%= stylesheet_link_tag "tailwind", "data-turbo track": "
reload" %>
</head>
```

Código 2 – Configuração do cabeçalho com Tailwind CSS

5.3 Implementação das Funcionalidades

Nas próximas seções, serão apresentadas as funcionalidades desenvolvidas em cada etapa do processo de desenvolvimento, organizadas em *features*. Cada *feature* corresponde a uma tarefa implementada, cuja relevância e contribuição para a evolução da FARMA serão discutidas ao longo do capítulo.

5.3.1 Feature: Desenvolvimento da Página Inicial

A primeira atividade realizada após a configuração do ambiente foi o desenvolvimento da Página Inicial. Inicialmente prevista para trabalhos futuros, essa funcionalidade revelou-se prioritária durante a revisão do planejamento, uma vez que constitui o ponto inicial de interação entre o usuário e o sistema. Como a versão anterior da FARMA operava apenas como API, tal página não existia, tornando-se indispensável com a adoção da nova arquitetura monolítica.

O design da Página Inicial tomou como referência o protótipo desenvolvido no Figma (Figura 21). No entanto, ao longo do processo de implementação, ajustes foram realizados com base nas sugestões da orientadora e dos coorientadores, resultando em um layout mais moderno, responsivo e alinhado à nova identidade visual da ferramenta. Para a estilização dessa interface, utilizou-se o *framework* TailwindCSS, previamente configurado no projeto. Os principais elementos definidos incluem uma breve apresentação sobre a FARMA, com o objetivo de contextualizar novos usuários quanto às finalidades e funcionalidades da plataforma, seguida por botões de acesso às áreas de registro e autenticação.

Além disso, a Página Inicial incorpora uma barra de navegação contendo a logo da FARMA e links de acesso rápido para as seguintes seções:

- **Sobre:** Apresenta uma descrição mais detalhada sobre a história, os objetivos e a evolução da FARMA;
- **Equipe:** Exibe os principais colaboradores responsáveis pela concepção e pelo desenvolvimento da ferramenta;
- **Pesquisa:** Reúne os trabalhos acadêmicos e estudos produzidos com base na plataforma (visualizáveis na linha do tempo apresentada na Seção 2.1);

- **Prêmios:** Destaca conquistas e reconhecimentos obtidos pela FARMA ao longo dos anos.

Embora algumas dessas áreas ainda não estejam completamente implementadas, a construção da Página Inicial estabelece a base estrutural necessária, funcionando como ponto de partida para que futuros colaboradores possam expandir e detalhar as seções planejadas.

Por fim, observa-se no canto inferior direito da interface uma visualização em escala reduzida que evidencia o comportamento da Página Inicial em dispositivos móveis, demonstrando a aplicação efetiva de princípios de responsividade no design. A versão final da tela pode ser consultada na Figura 23.

Figura 23 – Página Inicial da nova versão da FARMA

5.3.2 Feature: Autenticação e Registro de Usuário

Após a implementação da Página Inicial, o sistema passou a disponibilizar ao usuário o acesso à área de autenticação, permitindo o registro e o login por meio de e-mail e senha previamente cadastrados. Para viabilizar essa funcionalidade, foi necessário compreender e integrar a estrutura oferecida pela `gem Devise`, amplamente utilizada no ecossistema Ruby on Rails. Essa ferramenta abstrai grande parte da complexidade inerente à construção de um mecanismo de autenticação, fornecendo soluções robustas e seguras para cadastro, autenticação, recuperação de senha e gerenciamento de sessões.

Para a criação dos formulários, adotou-se a `gem Simple Form`, selecionada pela sua capacidade de gerar automaticamente campos HTML de maneira padronizada. Essa esco-

lha simplificou o processo de desenvolvimento e favoreceu a legibilidade e a manutenção das `Views`, resultando em um código mais organizado e coerente.

A exemplo da Página Inicial, a estilização das interfaces de autenticação foi realizada com o uso do *framework* Tailwind CSS, que disponibiliza classes utilitárias pré-definidas e facilita a personalização visual. Essa abordagem permitiu manter a consistência estética entre os diferentes formulários, contribuindo para uma experiência de usuário mais uniforme.

A seguir, apresenta-se uma descrição detalhada do processo de desenvolvimento das telas de autenticação e registro.

O protótipo desenvolvido no Figma e utilizado como referência para a página de registro de usuários encontra-se ilustrado na Figura 19. A partir da análise desse modelo, iniciou-se a implementação da interface, estruturando os elementos visuais e aplicando a estilização correspondente.

Diferentemente da primeira *Pull Request* (PR), na qual o uso da `Simple_Form` ainda não havia sido incorporado, esta etapa contou com uma refatoração significativa, resultando em melhorias tanto na organização do código quanto na apresentação visual da página. O trecho apresentado a seguir demonstra a evolução obtida, evidenciando a diferença entre a implementação inicial e a versão refatorada com o uso de `Simple_Form` (ver Código 3).

```
#Versão 1   PR - Sem Simple Form
<%= f.email_field :email, placeholder: "Email", class: "text-sm w
-full px-4 py-2 border border-gray-400" %>

#Versão 2   PR - Usando Simple Form
<%= f.email_field :email, type: :email %>
```

Código 3 – Comparação da 1º para 2º PR

Para a estilização, foram aplicados conceitos de componentização — prática que possibilita a reutilização de elementos recorrentes em diferentes partes do sistema. Essa abordagem reduz duplicações de código, melhora a organização geral do projeto e assegura a manutenção de um padrão visual e estrutural entre todos os formulários da aplicação.

A área de registro foi desenvolvida com uma imagem de fundo composta por linhas verdes onduladas, que conferem leve sensação de movimento ao layout. Na parte superior do formulário, posicionou-se o logotipo da FARMA, seguido de um texto informativo que identifica a seção como o espaço destinado ao cadastro de novos usuários. Em seguida, foram organizados os campos de entrada — `nome`, `email`, `password` e `password_confirmation`. Logo abaixo, encontra-se o botão `Registrar Usuário` e, por fim, um link de redirecionamento para a área de autenticação, destinado aos usuários que já possuem cadastro.

A ação pós-registro foi configurada para redirecionar o usuário diretamente à área de escolha de perfil, evitando a necessidade de realizar um novo login imediatamente após o cadastro. Assim, as credenciais só precisam ser reenviadas quando a sessão for encerrada. Esse

comportamento é definido pelo método privado `after_sign_up_path_for`, localizado no `controller` responsável pela tela de registro, conforme apresentado a seguir (ver Código 4):

```
def require_no_authentication
    return unless user_signed_in?

    redirect_to users_choose_profile_path, alert:I18n.t('devise.
   failure.already_authenticated')
end
```

Código 4 – Codigo de redirecionamento para area principal

A versão final da interface pode ser visualizada na Figura 24.

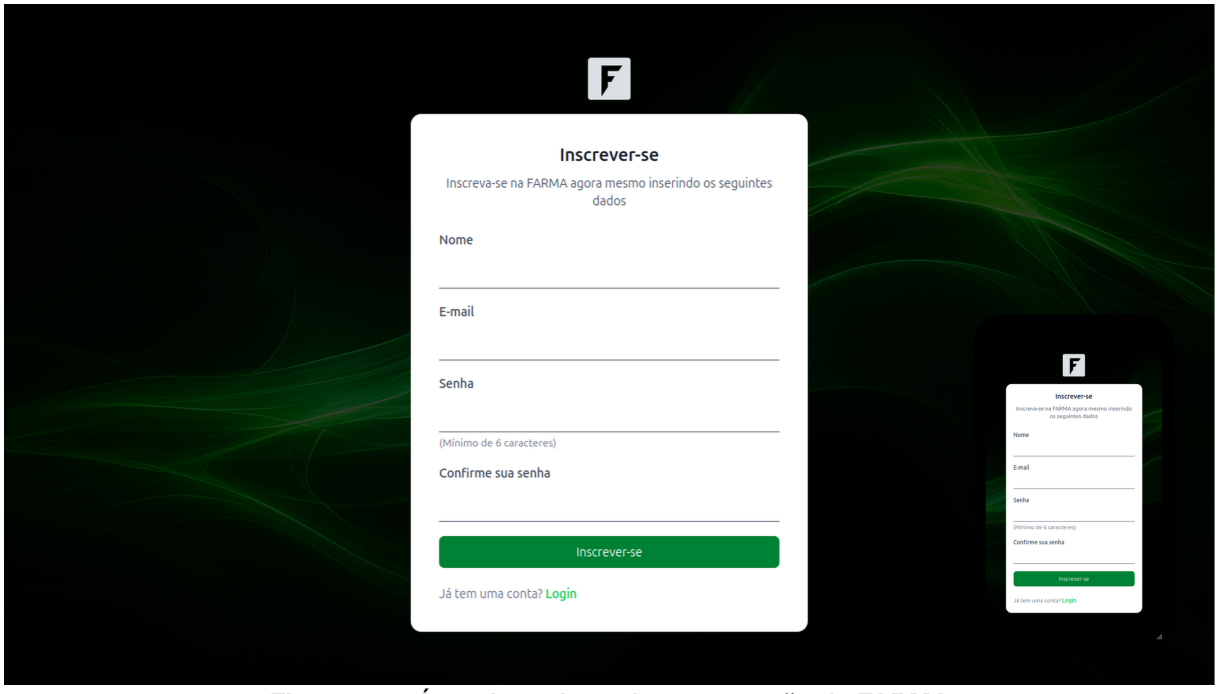

Figura 24 – Área de registro da nova versão da FARMA

Para a área de autenticação, foi adotado o mesmo padrão visual e estrutural desenvolvido para a área de registro. O protótipo da tela, elaborado durante a fase de projeto, serviu como base para a implementação e pode ser visualizado na Figura 25.

Diferentemente da primeira PR da área de registro, nesta etapa o `Simple Form` já foi aplicado desde o início, proporcionando uma melhoria significativa tanto no aspecto visual quanto na organização estrutural do código.

A estilização seguiu o mesmo padrão estabelecido anteriormente: um fundo com linhas verdes onduladas, transmitindo uma leve sensação de movimento. No topo do formulário foi inserido o logotipo da FARMA e, em seguida, os campos de `e-mail` e `senha`. Abaixo desses campos, foram posicionados o link para recuperação de senha, o botão de `Login` e, por fim, um redirecionamento para a área de registro, destinado aos usuários que ainda não possuem conta. A versão final da interface pode ser observada na Figura 25.

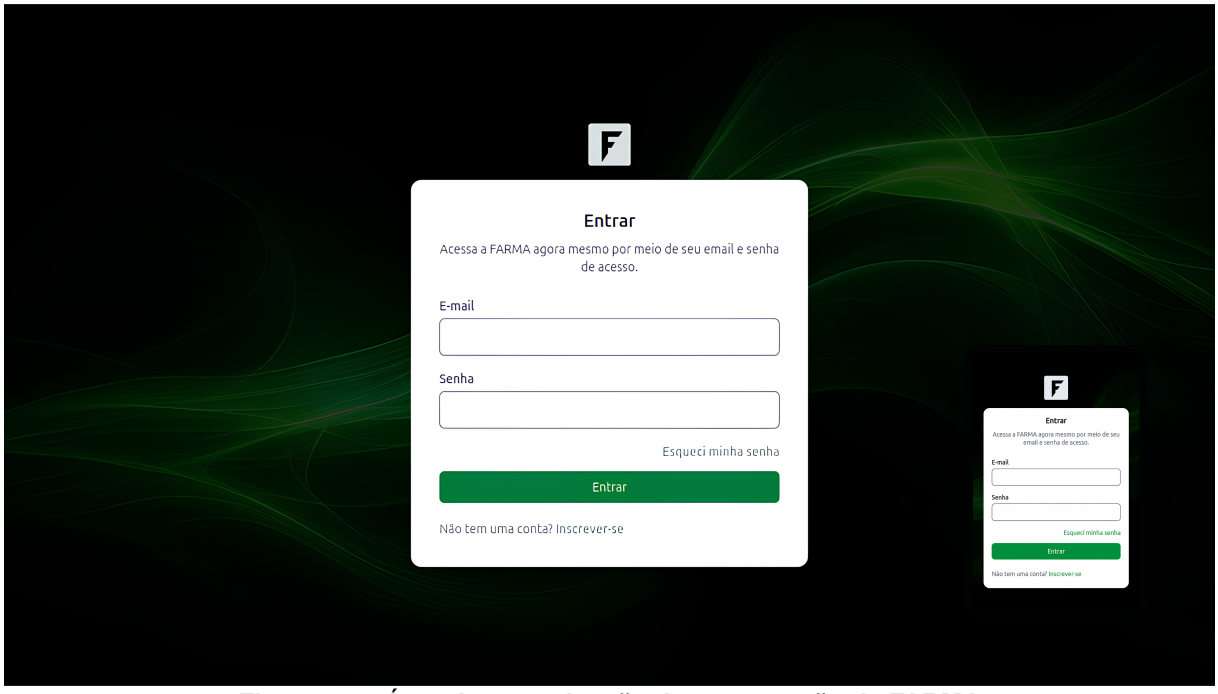

Figura 25 – Área de autenticação da nova versão da FARMA

Em ambas as figuras, observa-se, no canto inferior direito, uma versão reduzida que demonstra o comportamento das áreas de autenticação e de registro em dispositivos móveis, evidenciando a aplicação dos princípios de design responsivo na interface.

5.3.3 Feature: Recuperação de Senha

Considerando a possibilidade de o usuário esquecer sua senha de acesso, foi implementada na aplicação a funcionalidade de recuperação de senha. Essa funcionalidade foi desenvolvida utilizando o componente Action Mailer, fornecido pelo framework Ruby on Rails, que possibilita o envio de e-mails de forma configurável, podendo operar tanto de maneira síncrona (imediata) quanto assíncrona (em background, por meio de filas de processamento).

O `Action Mailer` funciona de forma semelhante a um `controller`, porém é responsável exclusivamente pelo envio de mensagens de e-mail. As classes de `mailers` herdam de `ActionMailer::Base` e, por convenção, localizam-se no diretório `appmailers`. O exemplo a seguir apresenta a estrutura básica da classe principal de `mailers` utilizada na aplicação, responsável por definir o remetente padrão e o layout dos e-mails(ver Código 5):

```
class ApplicationMailer < ActionMailer::Ba
se
    layout 'mailer/application'
    default from: 'contato@farma.educacional.mat.br'
end
```

Código 5 – Configuração Application Mailer

Para o correto funcionamento do envio de e-mails, é necessário configurar o servidor SMTP, responsável por gerenciar a comunicação com o provedor de e-mail. Essa configuração foi realizada no arquivo `config/initializers/mailer.rb`, conforme o exemplo abaixo. Nesse arquivo, são definidos os parâmetros essenciais, como domínio, endereço, porta, credenciais e método de autenticação (ver Código 6):

```
config.action_mailer.smtp_settings = {
    domain: mailer&.domain,
    address: mailer&.address,
    host: mailer&.host,
    port: mailer&.port,
    user_name: mailer&.user_name,
    password: mailer&.password,
    authentication: :login
}
```

Código 6 – Configuração de inicialização Mailer

Durante o desenvolvimento e os testes, foi utilizada a ferramenta Mailtrap [3], uma plataforma voltada para o ambiente de desenvolvimento que permite capturar e visualizar e-mails de teste sem enviá-los a destinatários reais. Dessa forma, é possível validar com segurança o conteúdo, o formato e os dados do e-mail (como remetente, destinatário e links), evitando o envio indevido de mensagens durante a fase de testes.

Na aplicação, o Mailtrap foi configurado como servidor SMTP, utilizando a chave de autenticação fornecida pela plataforma. Assim, quando o usuário solicita a redefinição de senha, o sistema envia o e-mail de recuperação para o host do Mailtrap. A partir da interface web da ferramenta, é possível visualizar o e-mail recebido, inspecionar seu conteúdo em HTML e confirmar se o link de redefinição está funcional. Essa simulação pode ser observada na Figura 25.

A tela de recuperação de senha não estava inicialmente prevista no escopo do projeto. Contudo, por se tratar de um elemento essencial do fluxo de autenticação, optou-se por incluí-la, mantendo o mesmo padrão visual aplicado nas telas de registro e login. O design adotado utiliza um fundo preto com linhas verdes onduladas, a logo da FARMA posicionada no topo do formulário e um campo de entrada para o e-mail cadastrado. Abaixo do campo, há um link de retorno para a tela de login e, ao final, um botão para envio do formulário. A versão final dessa interface pode ser visualizada na Figura 26.

Após o envio do formulário, o sistema processa a solicitação, gerando um token temporário associado ao usuário e enviando o link de redefinição de senha por e-mail (via Mailtrap, em ambiente de teste). Ao acessar esse link, o usuário é redirecionado para a tela de definição de nova senha, na qual deve informar e confirmar o novo código de acesso.

[3] Disponível em https://mailtrap.io/

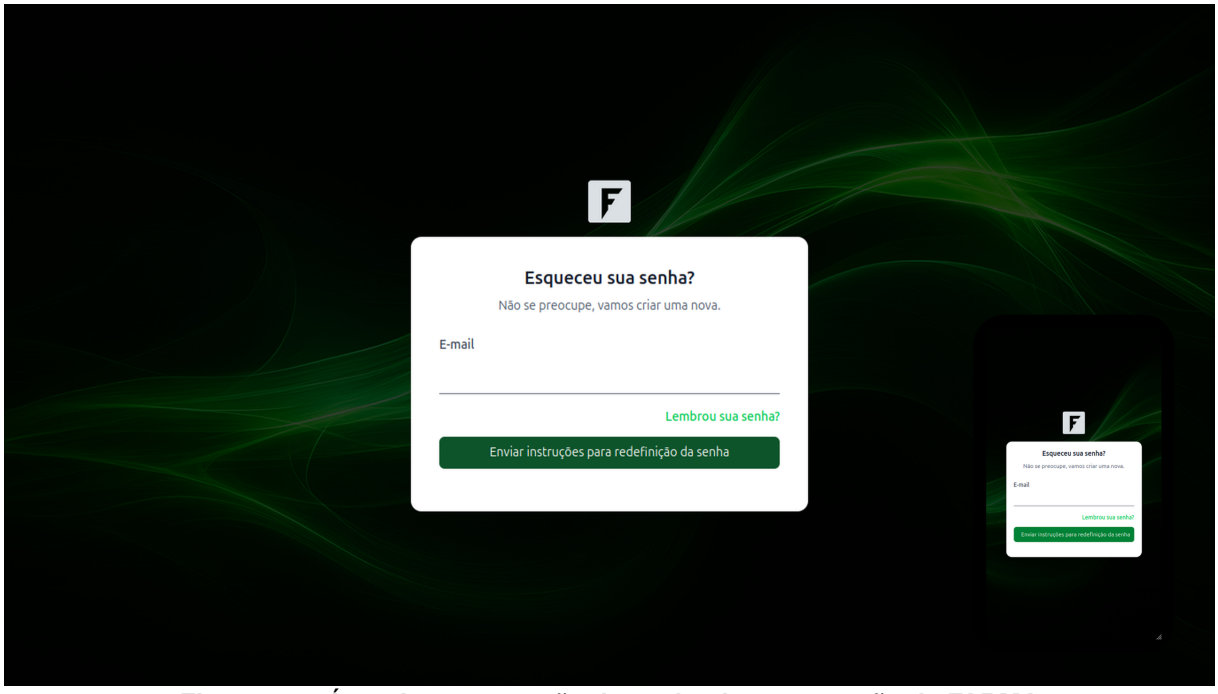

Figura 26 – Área de recuperação de senha da nova versão da FARMA

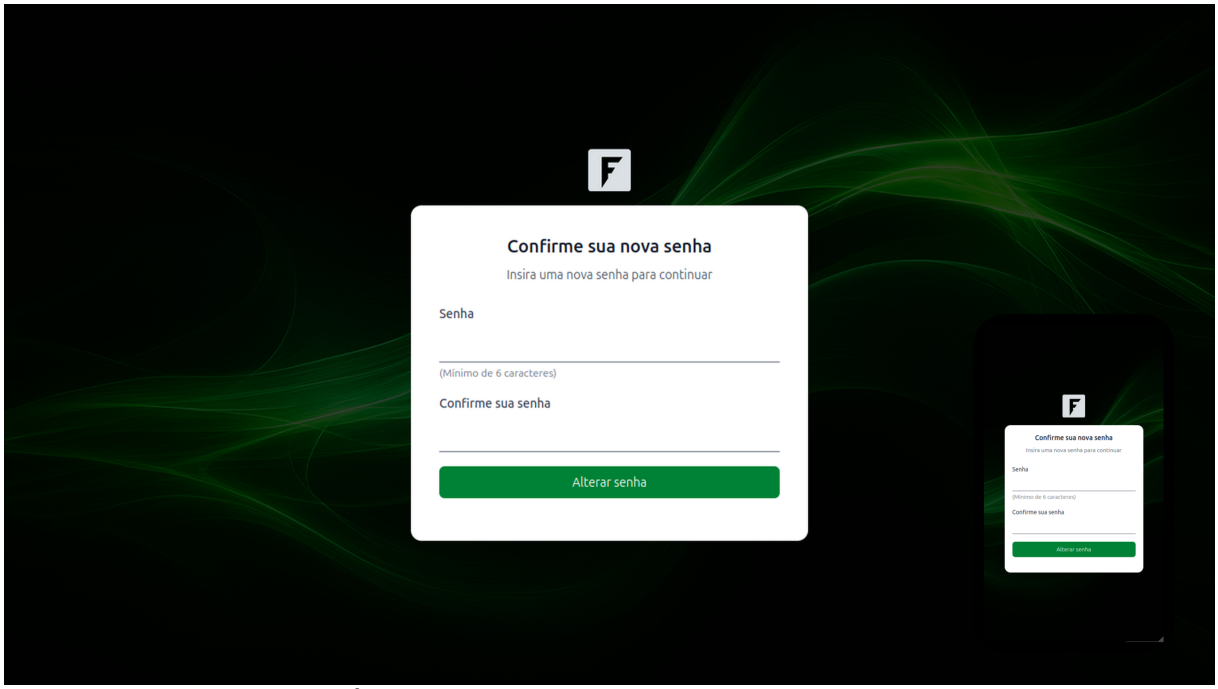

Figura 27 – Área de alteração de senha da nova versão da FARMA

Essa tela mantém o mesmo padrão visual das demais páginas de autenticação, apresentando fundo preto com linhas verdes onduladas, a logo da FARMA no topo e um formulário contendo dois campos: nova senha e confirmação da nova senha, seguidos de um botão para envio das informações. Esse fluxo proporciona uma experiência consistente e intuitiva ao usuário, além de reforçar a segurança do processo de autenticação. A versão final dessa interface pode ser visualizada na Figura 27.

5.3.4 Feature: Seleção Perfil de Usuário

Conforme citado por Marczal (2014), o uso de tecnologias amplia significativamente as formas de interação entre professores e alunos, especialmente com o apoio do computador e da Internet. Com base nesse princípio, definiu-se que o sistema deveria possuir áreas distintas para docentes e discentes, permitindo que os educadores criem seus OAs e que os alunos possam acessá-los de maneira organizada.

A partir dessa definição, foi desenvolvida uma tela de escolha de perfil, que formaliza essa separação entre alunos e educadores. Os educadores têm o papel de desenvolver os OAs, com o objetivo de transmitir determinado conteúdo e avaliar o aprendizado dos alunos por meio de exercícios e atividades, incluindo remediação de erros com dicas. Já os alunos têm a função de acessar os OAs que lhes foram atribuídos, revisando o conteúdo e acompanhando seu próprio progresso. Dessa forma, o usuário pode selecionar o perfil que deseja utilizar, de acordo com sua finalidade na ferramenta.

A tela de seleção de perfil segue o mesmo padrão visual adotado nas telas de registro e autenticação, uma vez que ainda integra o processo de entrada no sistema. Nesse estágio, o usuário ainda não possui acesso à plataforma propriamente dita, sendo necessário, primeiramente, definir o tipo de perfil para, então, acessar as funcionalidades da ferramenta. O fundo da tela mantém o estilo característico, com cor preta e linhas verdes onduladas. Na parte superior, encontra-se o logotipo da FARMA, e logo abaixo há dois botões: um destinado ao acesso como estudante e outro como educador, sendo cada um representado por um ícone que simboliza seu respectivo perfil. A versão final dessa tela pode ser visualizada na Figura 28.

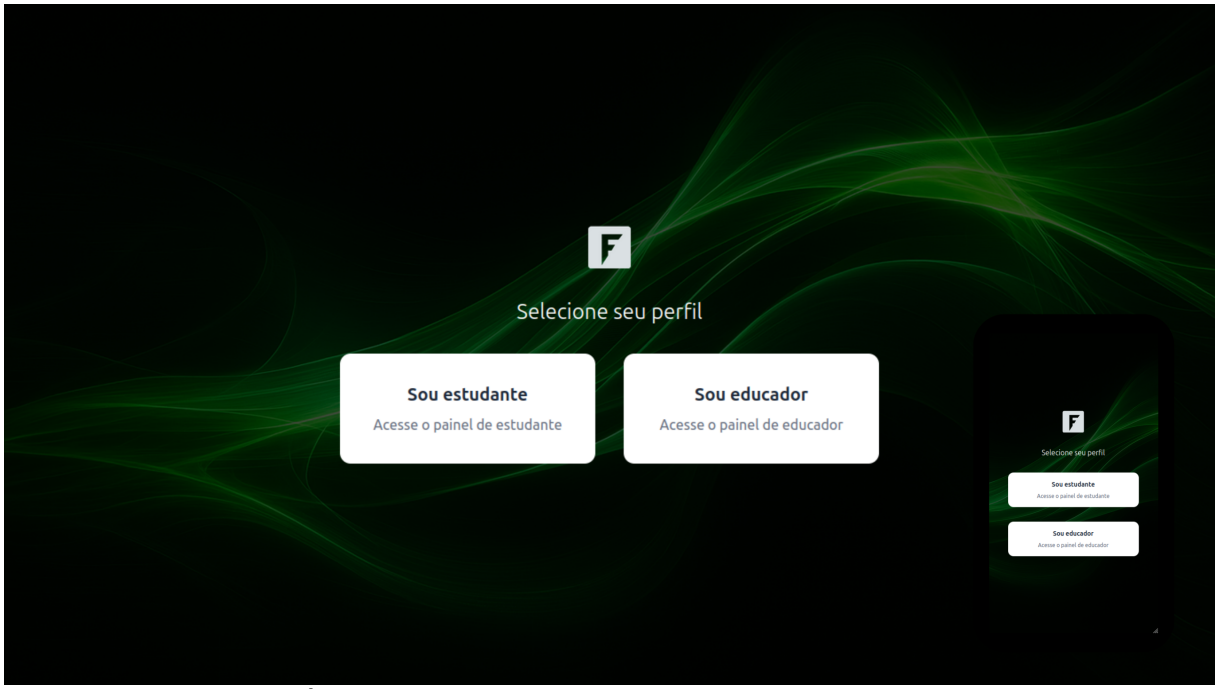

Figura 28 – Área de seleção de tipo de usuário da nova versão da FARMA

Até o momento do desenvolvimento deste trabalho, a área destinada ao estudante redireciona para uma página simples, sem estilização, exibindo apenas a mensagem “Olá, entrei como estudante” e um link para retornar à tela de seleção de perfil. Essa limitação se deve ao fato de que o foco principal do desenvolvimento foi a área do educador, responsável pela criação dos Objetos de Aprendizagem, que constitui o propósito central deste trabalho.

5.3.5 Feature: Dashboard Perfil Educador

Após a autenticação do usuário e a seleção do perfil de educador, o sistema redireciona para a tela inicial da ferramenta, denominada *Dashboard*, apresentada na Figura 29. No canto inferior direito da figura, observa-se uma versão reduzida que ilustra o comportamento responsivo da interface em dispositivos de menor tamanho de tela.

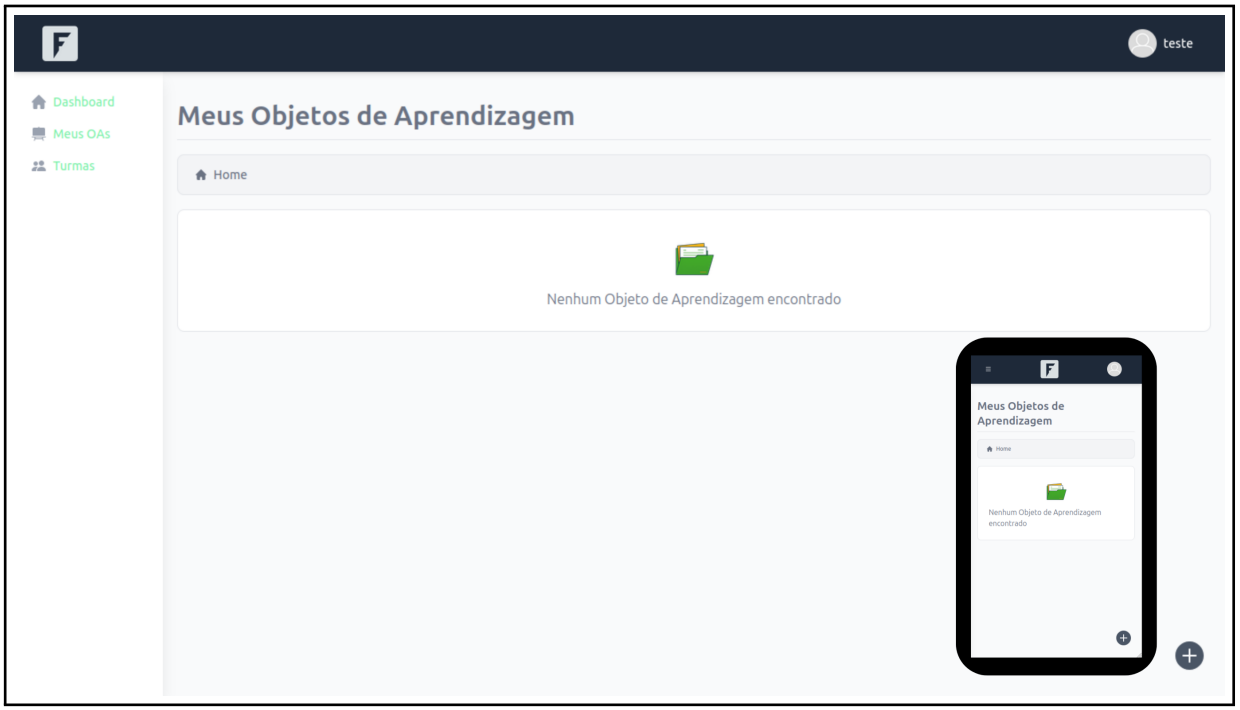

Figura 29 – Dashboard da nova versão da FARMA

Nessa etapa, foi definido um *layout* padrão, aplicado a toda a interface do sistema. Esse *layout* segue um estilo amplamente adotado em sistemas web e organiza a interface em elementos principais que garantem consistência, facilidade de navegação e usabilidade. Os componentes que compõem o *Dashboard* são:

- **Cabeçalho**: localizado na parte superior da interface, apresenta a logomarca da aplicação no canto esquerdo e, no canto direito, a imagem e o nome do usuário. Este componente também funciona como um botão que ativa um menu suspenso (*dropdown*), exibindo informações do perfil (nome e e-mail) e oferecendo links para acesso à área de perfil, alternância entre os perfis de educador e aluno, além da opção de logout.

- **Menu lateral**: destinado a proporcionar acesso rápido e intuitivo às principais seções do sistema. O menu disponibiliza links diretos para a página inicial (*Dashboard*), para a listagem de Objetos de Aprendizagem (OAs) criados e para as turmas cadastradas pelo educador, contribuindo para maior praticidade e agilidade na navegação.

- **Trilhas de navegação (*breadcrumbs*)**: à medida que o usuário acessa diferentes páginas, aumenta a profundidade da navegação, o que pode dificultar o retorno a áreas específicas. Para resolver esse problema, foram implementadas trilhas de navegação, que exibem o caminho percorrido e permitem retornar facilmente a páginas anteriores, mantendo o fluxo de interação mais organizado e compreensível durante o processo de criação e gerenciamento de OAs.

Após a definição do *layout* padrão, iniciou-se a etapa de estilização da página inicial. Essa tela apresenta uma mensagem de boas-vindas — “Bem-vindo à FARMA” — seguida da listagem dos seis OAs criados ou modificados mais recentemente. O objetivo dessa página é proporcionar acesso rápido aos objetos em desenvolvimento recente, enquanto os demais permanecem disponíveis na seção “Meus OAs”, acessível por meio do menu lateral.

O trecho de código responsável por ordenar e limitar a exibição dos OAs recentes foi implementado no `controller` da área inicial da ferramenta, conforme o exemplo abaixo (ver Código 7):

```
class Educators::HomeController < Educators::BaseController
    def dashboard
        @los = current_user.los
            .includes(:picture_attachment)
            .order(updated_at: :desc) # Ordena pela última edição
   realizada
            .limit(6) # Limita a apresentação a até 6 OAs
    end
end
```

Código 7 – Codigo para ordenação de OA na Dashboard

5.3.6 Feature: Área de Criação de Objetos de Aprendizagem (OA)

Após acessar o sistema, o próximo passo do educador consiste na criação de um OA, seguido do desenvolvimento de seus elementos complementares, como `introduções, exercícios, passos de solução e feedbacks`.

Inicialmente, foi implementada a área de visualização geral dos OAs, que exibe uma listagem ordenada por data de criação. O trecho de código a seguir ilustra a lógica de ordenação utilizada (ver Código 8):

```ruby
class Educators::LosController < Educators::BaseController
    def index
        @los = current_user.los
        .includes(:picture_attachment)
        .order(created_at: :desc) # Ordena por data de criação
    end
end
```

Código 8 – Codigo para ordenação de OA na area de visualização principal

Além disso, a tela conta com um botão flutuante, seguindo o conceito de `Floating Action Button` (FAB). Posicionado no canto inferior direito da interface, esse botão apresenta comportamento de menu suspenso (*dropdown*), permitindo, neste contexto, apenas a criação de um novo OA. A aparência e a posição do botão podem ser observadas na Figura 30.

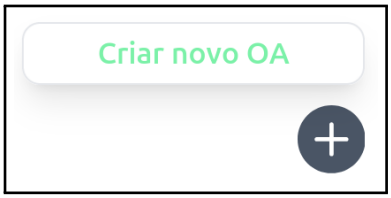

Figura 30 – Botão flutuante de criação de introdução e exercício na nova versão da FARMA

Ao selecionar a opção de criação, o educador é redirecionado ao formulário correspondente. Assim como nas demais áreas da aplicação, foi utilizado o `Simple Form`, ferramenta que simplifica a estrutura do formulário, tornando o código mais legível e de fácil manutenção. O formulário é apresentado dentro do *layout* padrão aplicado em todo o sistema.

O formulário contém os seguintes campos:

- **Título**: obrigatório, identifica o OA;
- **Descrição simples**: obrigatório, resumo do conteúdo;
- **Imagem ilustrativa**: opcional, com pré-visualização;
- **Disponível para alunos**: define se o OA está acessível aos estudantes;
- **Permitir cópia**: permite que outros usuários dupliquem o OA.

Ao submeter o formulário, caso todas as informações obrigatórias estejam válidas, o usuário é redirecionado para a área "`Meus OAs`", onde poderá visualizar o objeto recém-criado. A Figura 31 apresenta o formulário inicial de criação de OA, enquanto a Figura 32 exibe esse mesmo formulário devidamente preenchido, já próximo do envio. Por fim, a Figura 33 mostra a visualização de um OA criado e disponível para acesso. Em todas as figuras, observa-se ainda uma versão reduzida da interface, evidenciando o comportamento responsivo da aplicação em dispositivos com telas menores.

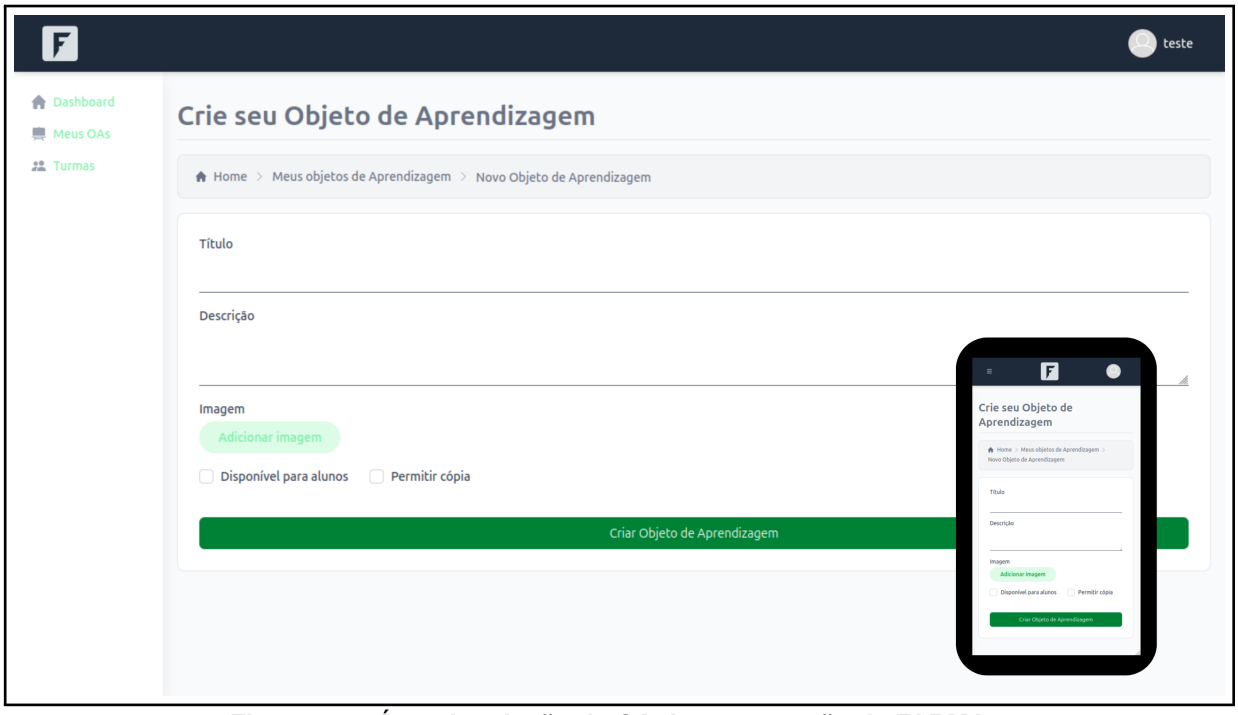

Figura 31 – Área de criação de OA da nova versão da FARMA

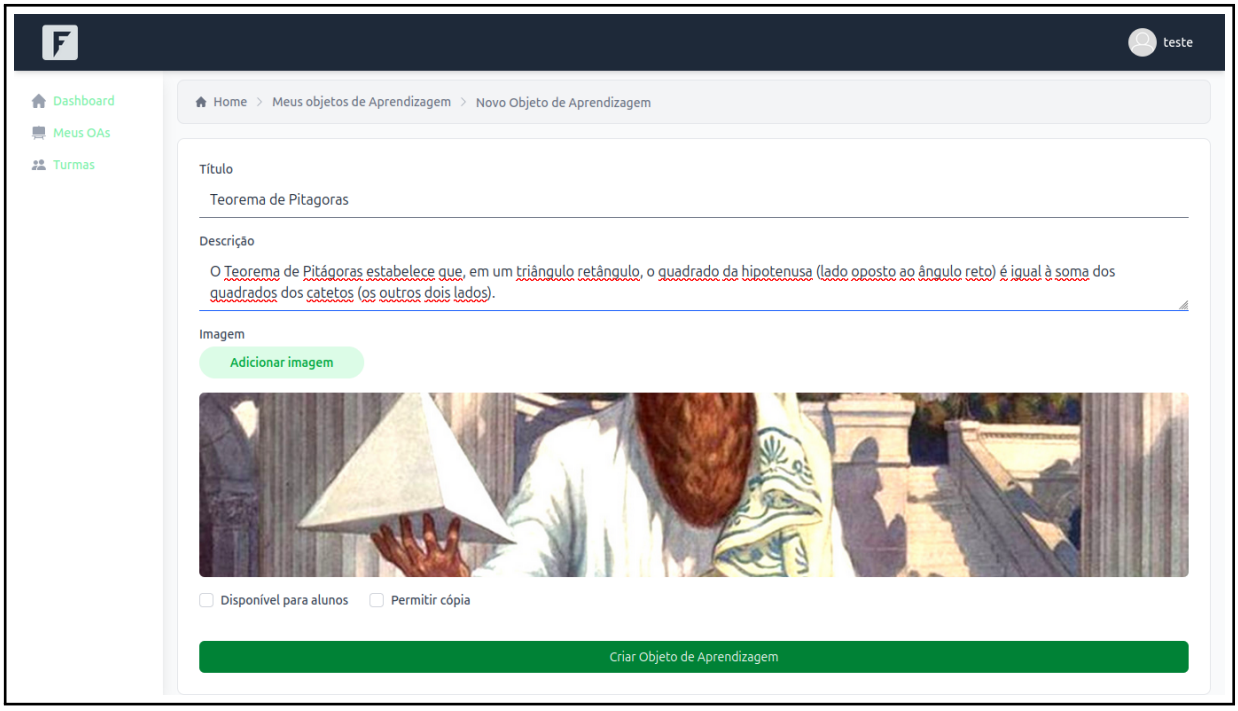

Figura 32 – Formulário de criação de OA preenchido na nova versão da FARMA

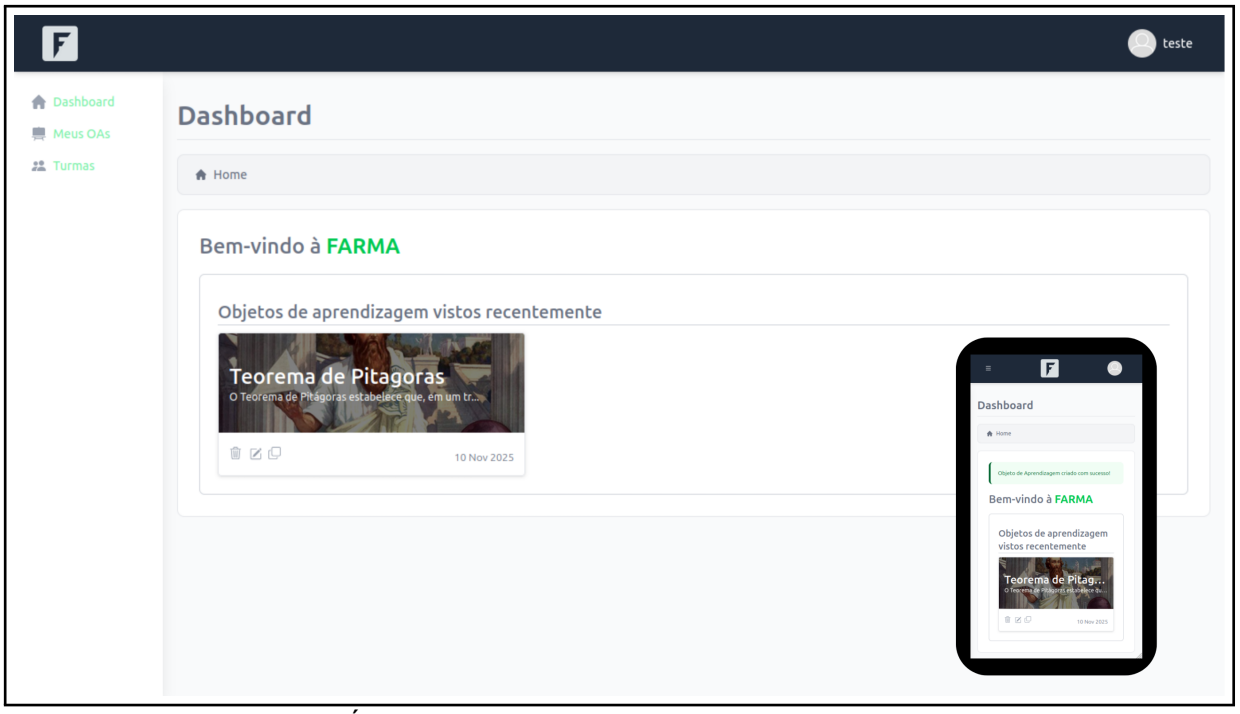

Figura 33 – Área de vizualização de OA na nova versão da FARMA

Um ponto importante diz respeito à ausência de *upload* de imagem pelo educador. Como apenas os campos de `título` e `descrição` são obrigatórios, tornou-se necessário definir um comportamento padrão para os casos em que nenhuma imagem é anexada. Nesses cenários, a aplicação renderiza automaticamente uma imagem genérica, conforme apresentado no método a seguir (ver Código 9):

```
def picture_url
  if picture.attached
    Rails.application.routes.url_helpers.rail _blob_path(picture,
   only_path: true) //se houver, renderiza a imagem que o usuario
   inseriu
  else
    ActionController::Base.helpers.asset_path('bg/default_lo.png') //
    senão, renderiza a imagem padrão
  end
end
```

Código 9 – Codigo aplicação de imagem padrão ao OA

Esse mecanismo garante a consistência visual da aplicação e evita erros de renderização, assegurando que sempre haja uma imagem exibida — seja ela personalizada ou padrão.

5.3.7 Feature: Acesso ao Objeto de Aprendizagem (OA)

Ao acessar um OA, o usuário é direcionado para a área de exposição das informações previamente cadastradas. Nessa página, além de visualizar os dados do OA, o educador pode cadastrar e gerenciar as `introduções` e `exercícios` associados.

A interface foi desenvolvida com foco na organização e na padronização visual. O título e a descrição cadastrados são exibidos na parte superior da página, com limitação do número de caracteres para garantir melhor legibilidade e manter um layout harmônico. O trecho de código a seguir exemplifica essa limitação (ver Código 10):

```
<%= @lo.description.truncate(300) %>
<%= @lo.title.truncate(50) %>
```

Código 10 – Limitação de caracteres na apresentação de titulo OA

Além dessas informações textuais, a página também apresenta a imagem associada ao OA. Caso nenhuma imagem tenha sido cadastrada, uma imagem padrão é exibida.

Outro elemento importante da interface é o botão FAB, responsável por facilitar o acesso às ações principais. Nesse contexto, o FAB disponibiliza as opções de cadastro de introdução e exercício vinculados ao OA, proporcionando uma navegação mais intuitiva e eficiente para o educador.

A Figura 34 apresenta o resultado final da página de acesso ao OA na nova versão do sistema FARMA.

Figura 34 – Area de vizualização de dados cadastrados de OA na nova versão da FARMA

5.3.8 Feature: Criação de Introdução

Com o objetivo de oferecer ao estudante uma base teórica antes da realização dos exercícios, a funcionalidade de criação de `introduções` foi desenvolvida para auxiliar na construção do raciocínio e na consolidação dos fundamentos necessários para a execução das atividades. O usuário pode criar mais de uma introdução, possibilitando uma abordagem gradual e estruturada antes de avançar para os exercícios correspondentes.

Durante o processo de criação, o formulário segue o mesmo padrão visual e estrutural utilizado na criação de um OA. O layout mantém a consistência estética da aplicação, e o formulário é composto pelos seguintes campos:

- **Título**: campo obrigatório;
- **Descrição**: campo obrigatório que, até o momento do desenvolvimento desta *feature*, ainda não contava com a integração do editor TinyMCE;
- **Publicar**: campo *checkbox* utilizado para definir se a introdução estará disponível assim que o OA for publicado;

A Figura 35 apresenta a página de criação de Introdução na nova versão do sistema FARMA, exibindo a estrutura do formulário e sua organização visual. Já a Figura 36 mostra um exemplo desse formulário devidamente preenchido, ilustrando como os dados são distribuídos antes do envio.

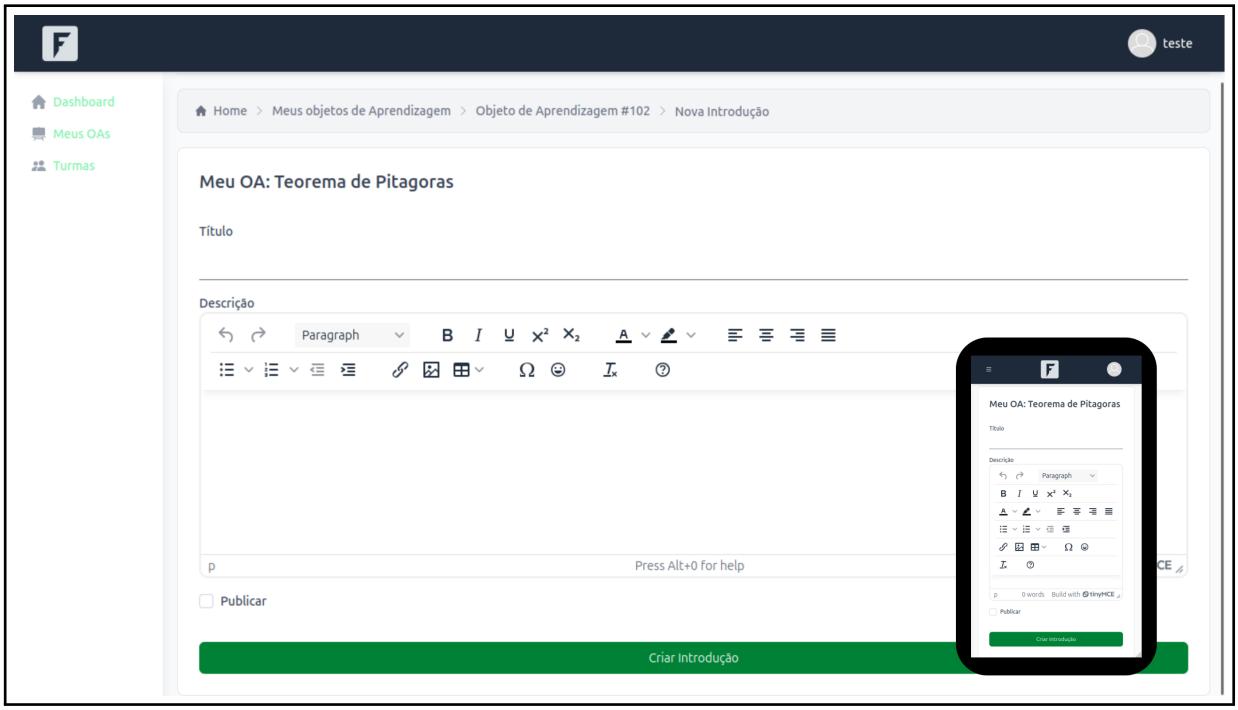

Figura 35 – Página de criação de Introdução na nova versão da FARMA

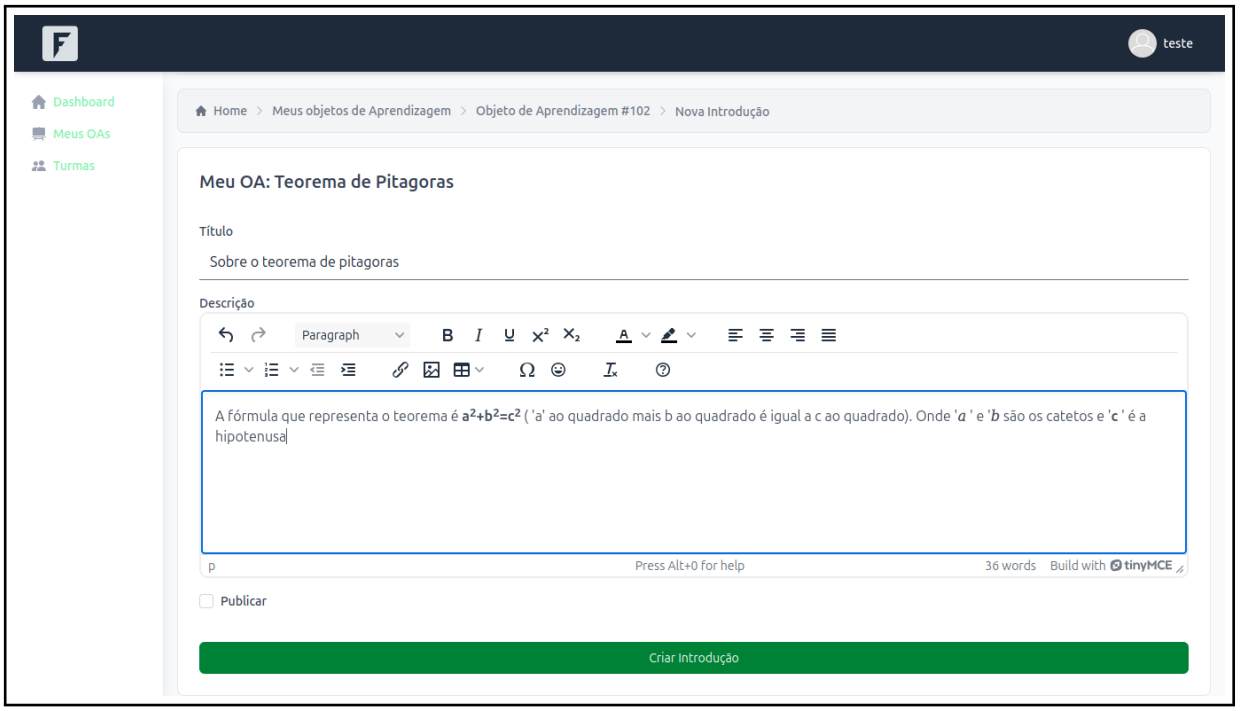

Figura 36 – Exemplo de formulário de criação de Introdução do OA preenchido

Caso a criação da introdução seja concluída com êxito, o usuário é redirecionado para a área de visualização do OA, onde já é possível observar a introdução cadastrada. O design desse componente foi desenvolvido para destacar visualmente seu conteúdo e evitar confusão com os exercícios, uma vez que ambos coexistem na mesma página.

Para isso, cada introdução apresenta seu título precedido pela palavra “Introdução”, deixando explícita sua função dentro do OA. Em seguida, exibe-se a descrição correspondente. Ambos os campos — título e descrição — possuem limitação de caracteres, aplicada por meio da função `truncate()`, com o objetivo de preservar a legibilidade e manter o padrão estético da interface. A Figura 37 ilustra o resultado após a criação da introdução.

A Figura 37 ilustra o resultado após a criação da introdução.

5.3.9 Feature: Criação de Exercício

Após criar a introdução ao conteúdo, o educador pode prosseguir para a elaboração dos exercícios diretamente a partir da área de visualização do OA. Essa ação é realizada por meio do botão FAB, que permite a criação de um ou mais exercícios, cada um deles podendo incluir um ou mais passos de solução. O exercício tem como objetivo apresentar um enunciado e, posteriormente, detalhar os passos necessários para sua resolução.

A interface de criação de exercícios segue o mesmo padrão visual adotado na criação de introduções, assegurando consistência e familiaridade na experiência do usuário. No topo da página, é exibida a identificação do OA ao qual o exercício será vinculado. Em seguida, apresenta-se o formulário, composto pelos seguintes campos:

Figura 37 – Apresentação da introdução após a criação

- **Título**: campo obrigatório referente ao nome do exercício;
- **Descrição**: campo obrigatório que, até o momento do desenvolvimento desta *feature*, ainda não dispunha da integração com o editor *TinyMCE*;
- **Publicar**: campo do tipo *checkbox*, utilizado para indicar se o exercício deverá estar disponível assim que o OA for publicado;

A Figura 38 apresenta a página de criação de Exercício na nova versão do sistema FARMA, evidenciando a estrutura do formulário e sua organização visual. Já a Figura 39 mostra um exemplo desse formulário devidamente preenchido, ilustrando como os campos são distribuídos e exibidos antes do envio.

Após a criação bem-sucedida do exercício, o usuário é redirecionado para a área de visualização do OA, onde já é possível observar o exercício recém-cadastrado. Assim como nas introduções, o título do exercício é precedido pela palavra “Exercício:”, com o intuito de oferecer maior clareza e distinguir visualmente o tipo de conteúdo exibido.

O componente de exercício foi estilizado de forma semelhante ao da introdução. No topo, apresenta-se o `título`, seguido da `descrição`; no canto inferior esquerdo, encontra-se um botão que direciona o usuário à área de visualização dos passos de solução vinculados ao exercício.

A criação de uma área específica para o gerenciamento de exercícios e de seus respectivos passos de solução foi planejada com o objetivo de preservar a organização da interface. Como as `introduções` e os `exercícios` já coexistem no ambiente de visualização do OA, a inclusão de um terceiro componente nesse mesmo espaço poderia gerar sobrecarga visual

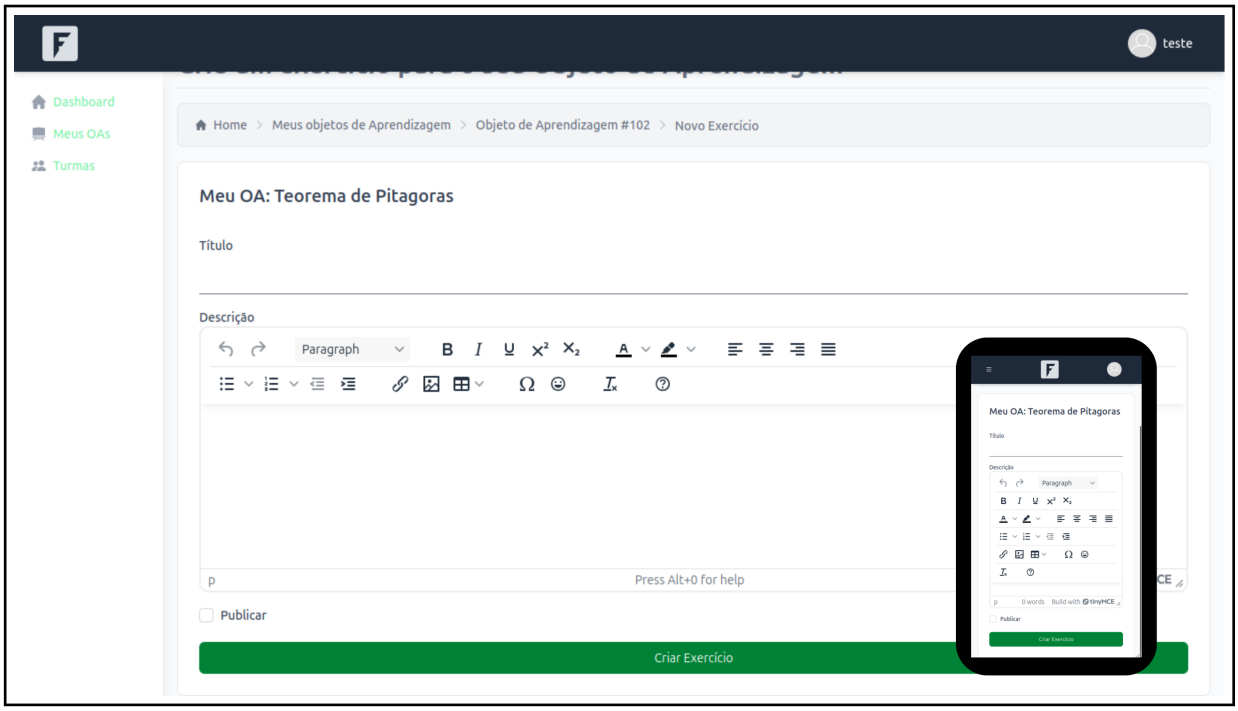

Figura 38 – Página de criação de Exercício na nova versão da FARMA

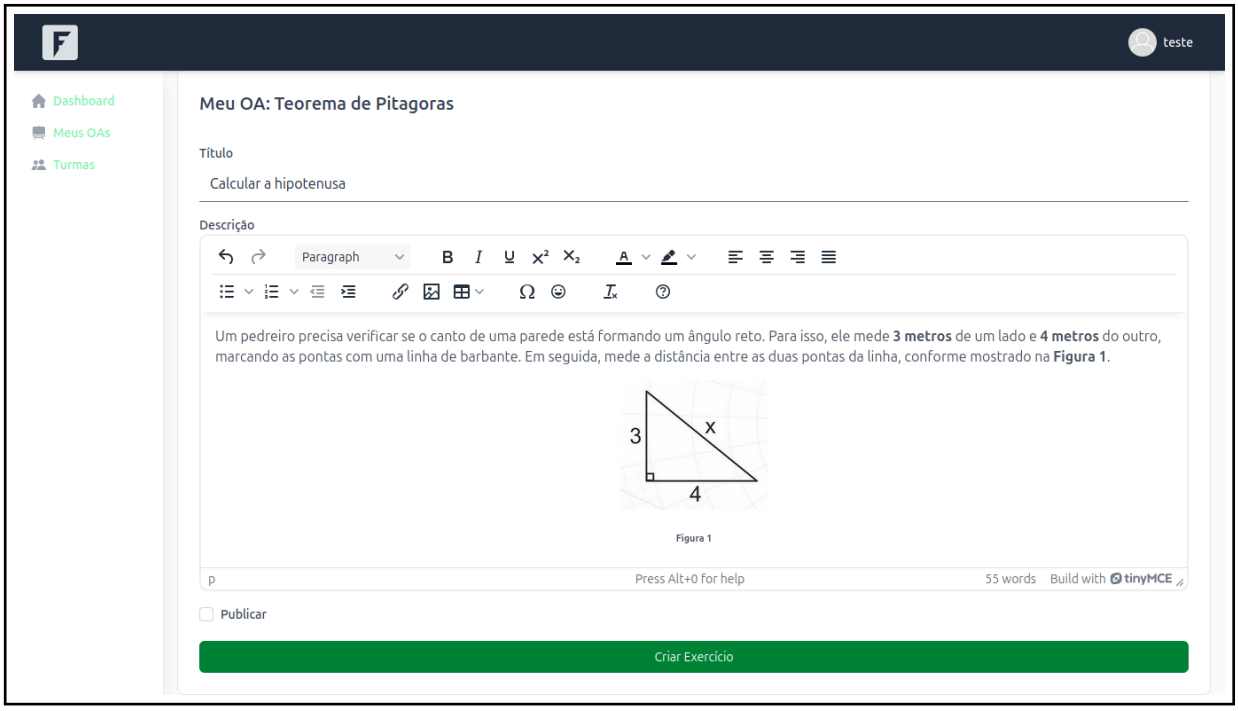

Figura 39 – Exemplo de formulario de criação de Exercicio do OA preenchido

e dificultar a compreensão por parte do usuário. Dessa forma, o desenvolvimento de uma área dedicada à gestão das questões e de seus feedbacks mostrou-se a solução mais adequada, promovendo maior clareza e melhor usabilidade.

A Figura 40 ilustra o resultado final do componente de Exercício na nova versão do sistema FARMA.

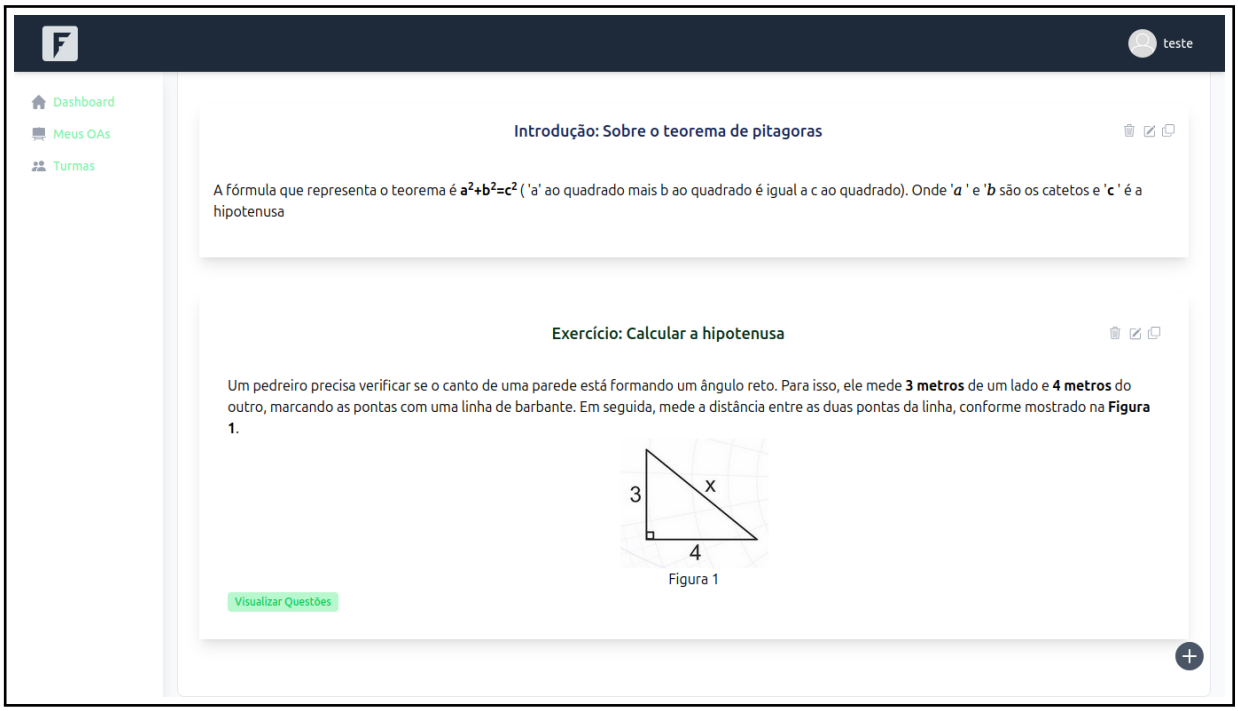

Figura 40 – Apresentação de exercicio após a criação

5.3.10 Feature: Criação de Passos de Solução

Após a criação da introdução do exercício, o usuário passa a ter acesso a um botão localizado no canto inferior esquerdo da tela, conforme demonstrado na Figura 40. Esse botão direciona para a área de visualização dos *Passos de Solução*, que podem ser um ou mais, e têm como objetivo avaliar o conhecimento do estudante sobre o conteúdo abordado na introdução do OA.

Essa tela apresenta, na parte superior, o título do OA em questão, com o propósito de deixar explícito ao usuário o contexto em que está navegando, evitando possíveis confusões. Caso ainda não existam passos cadastrados, é exibida a mensagem “Nenhum exercício encontrado”. A Figura 41 demonstra essa tela inicial da área de visualização dos Passos de Solução.

Para criar um novo Passo de Solução, o usuário deve clicar no botão FAB, localizado no canto inferior direito da tela, e selecionar a opção “Criar Passo de Solução”. Essa ação redireciona o usuário ao formulário de criação.

O formulário apresenta os seguintes campos:

- **Título**: campo obrigatório;

Figura 41 – Acesso à área de visualização de Passos de Solução na nova versão da FARMA

- **Descrição**: campo obrigatório que, até o momento do desenvolvimento desta *feature*, ainda não contava com a integração do editor *TinyMCE*;
- **Resposta esperada**: campo destinado ao valor considerado correto para esse passo da solução;
- **Número de casas decimais**: campo utilizado para definir a quantidade de casas decimais que devem ser consideradas na avaliação da resposta do estudante;

Ao finalizar o preenchimento, o usuário deve acionar o botão “Criar Passo de Solução” para salvar o registro. A Figura 42 apresenta o formulário de criação dessa etapa, enquanto a Figura 43 mostra um exemplo do formulário devidamente preenchido, ilustrando a disposição dos campos antes do envio.

Após o envio do formulário, o sistema exibe a tela de visualização do Passo de Solução criado. Essa interface segue o mesmo padrão visual adotado nas páginas de Introdução e Exercício, apresentando o título definido pelo usuário — precedido da expressão “Passo de Solução” — seguido da descrição correspondente. Esse layout pode ser observado na Figura 44.

5.3.11 Feature: Criação de Dicas

Após a criação do Passo de Solução, o usuário pode acessar, no canto superior direito da interface, um ícone em formato de lâmpada. Ao selecioná-lo, torna-se possível criar uma dica ou um feedback, cujo objetivo é oferecer remediação ao erro identificado no desempenho do estudante. A Figura 45 apresenta o local destinado à elaboração dessas orientações.

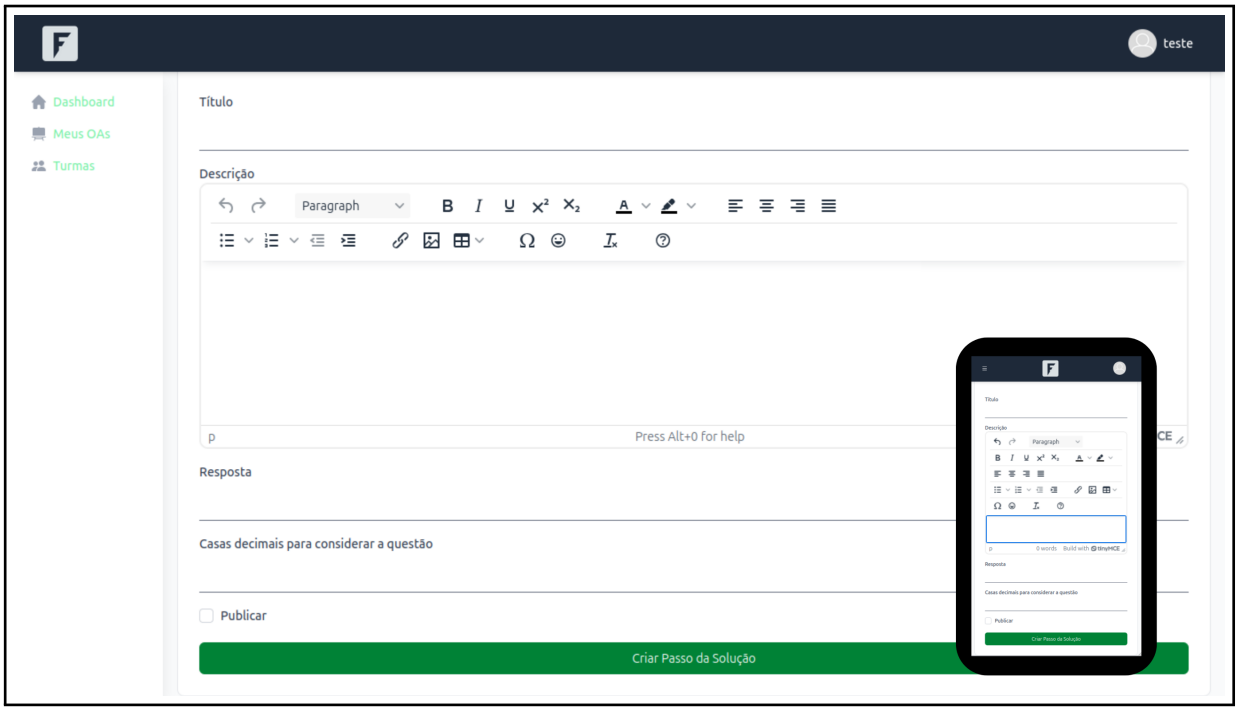

Figura 42 – Formulário de criação de Passo de Solução

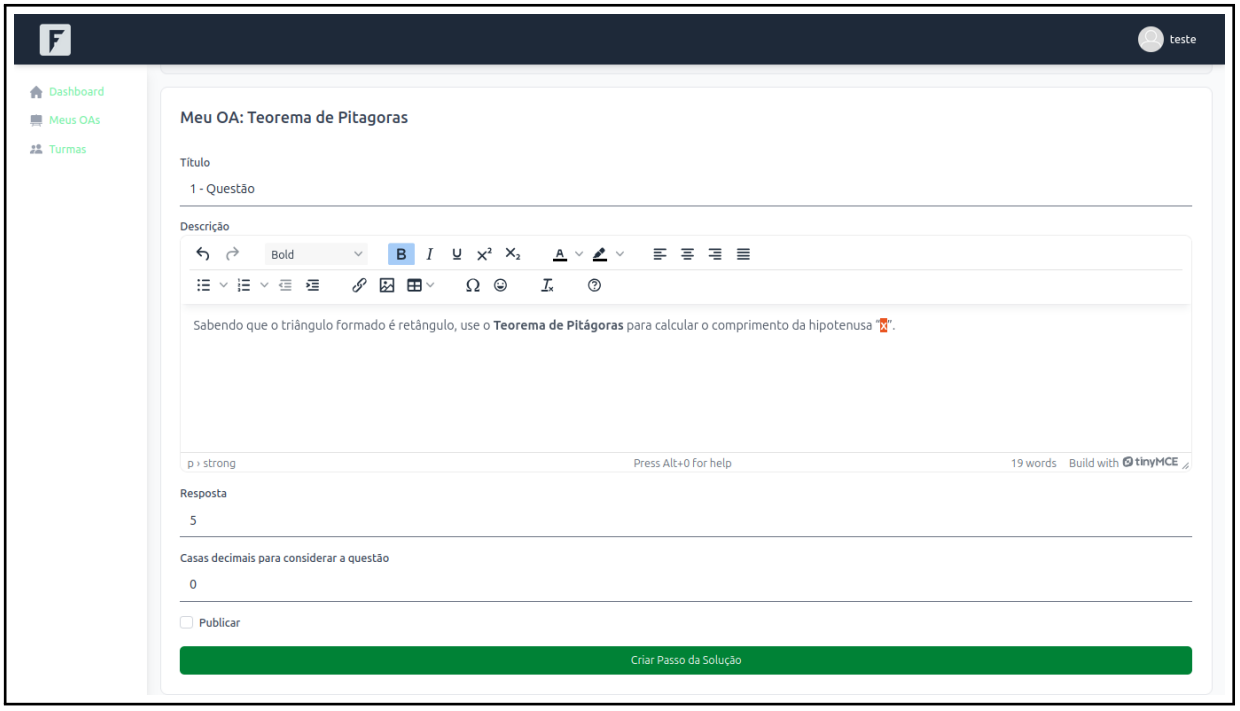

Figura 43 – Exemplo de formulario de criação de Passo de solução do OA preenchido

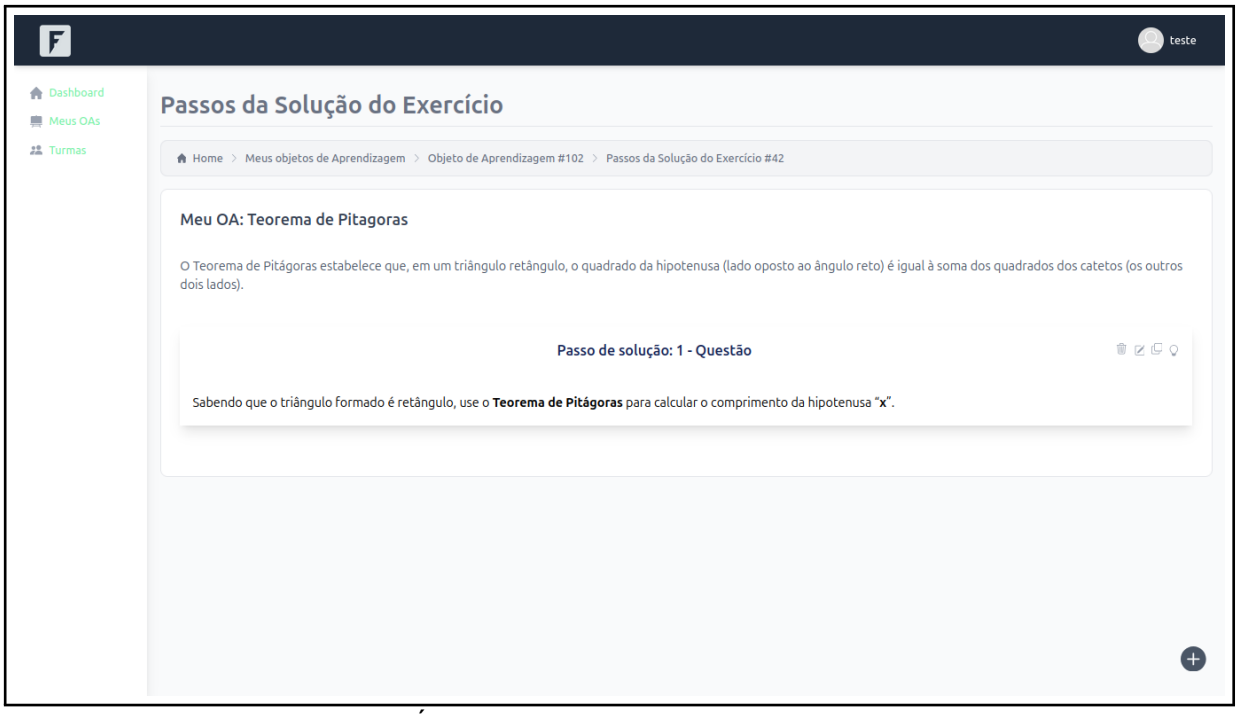

Figura 44 – Área de visualização de Passos de Solução

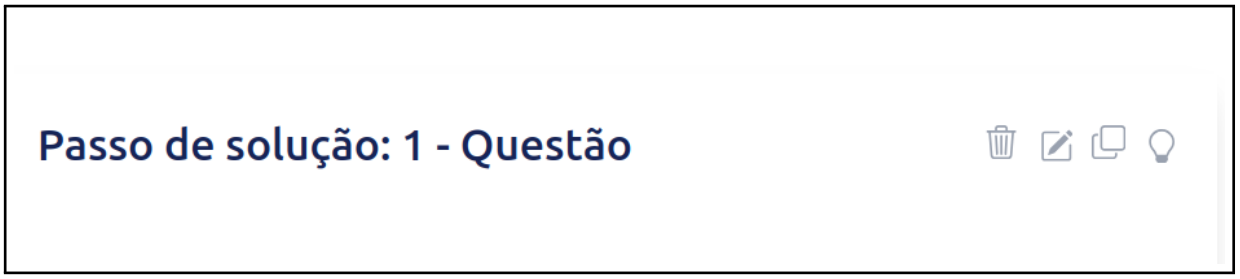

Figura 45 – Icone criação de dicas e feedbacks

Ao acessar essa área, o usuário visualiza um formulário composto pelos campos de título, descrição, número de tentativas e modo de apresentação dos feedbacks — que pode ocorrer de forma sequencial ou condicionada à quantidade de erros cometidos. O formulário é concluído por meio de um botão de envio, responsável por registrar a dica criada no sistema. A Figura 46 ilustra a interface correspondente a essa funcionalidade.

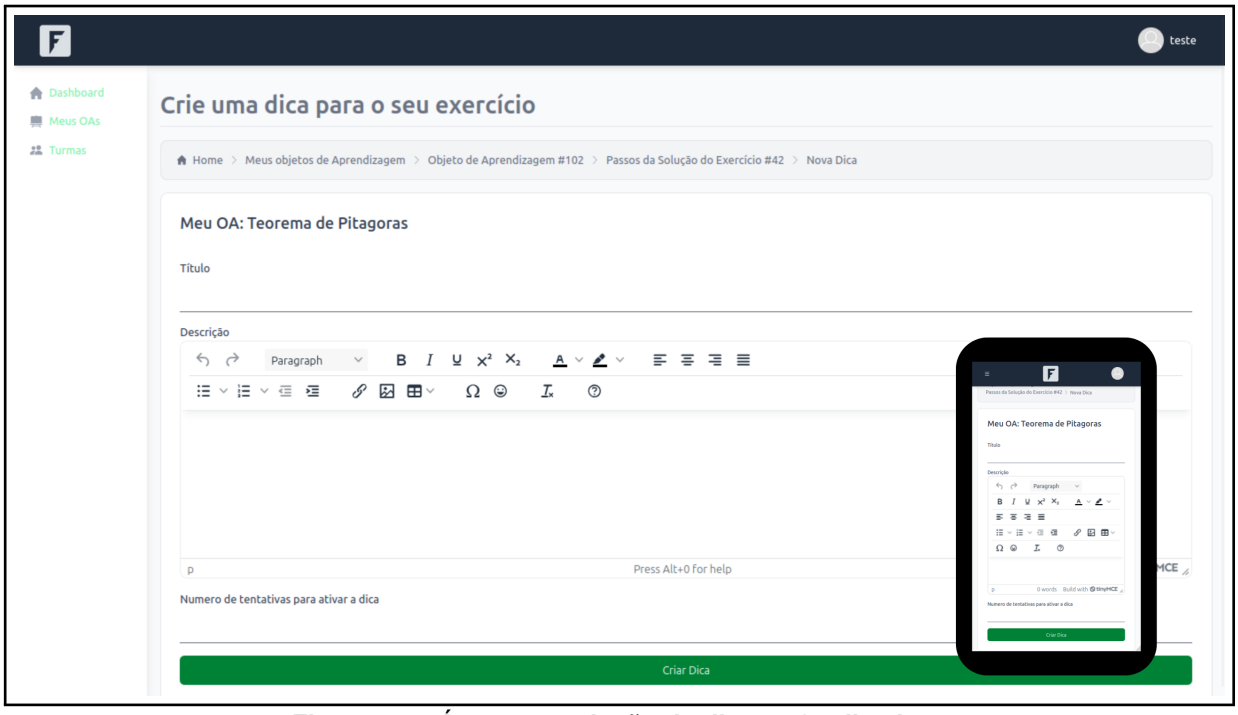

Figura 46 – Área para criação de dicas e feedbacks

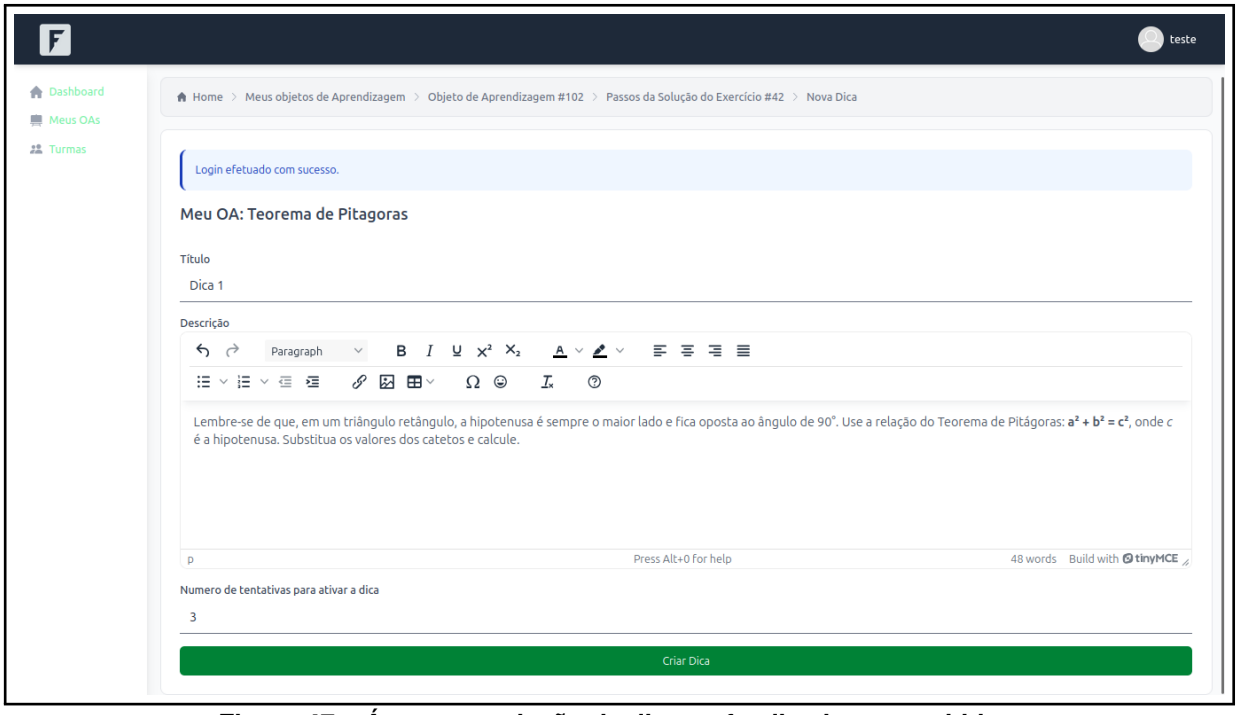

Figura 47 – Área para criação de dicas e feedbacks preenchida

5.4 Integração do TinyMCE

Conforme ilustrado nas figuras anteriores que apresentam as áreas de criação e edição de Objetos de Aprendizagem — incluindo as interfaces de elaboração de **Exercício**, **Introdução**, **Passo de Solução** e **Dicas** — a implementação do TinyMCE já está incorporada a esses formulários, permitindo a demonstração completa do fluxo de construção de cada elemento. É importante destacar, entretanto, que a integração desse editor ocorreu somente após a finalização dessas etapas, decisão tomada em conjunto com os orientadores para garantir eficiência no desenvolvimento e evitar retrabalho nas fases iniciais do projeto.

O TinyMCE é um editor de texto rico (WYSIWYG — *What You See Is What You Get*) de código aberto que possibilita a criação e edição de conteúdo formatado de maneira visual e intuitiva, sem a necessidade de escrita direta em HTML. Sua adoção foi definida durante as reuniões de orientação, considerando sua simplicidade de uso, ampla compatibilidade com o framework *Ruby on Rails* e facilidade de customização, fatores que o tornam uma solução adequada ao contexto deste projeto.

Para integrar o TinyMCE ao sistema, o primeiro passo consistiu na instalação da `gem`[4] correspondente, por meio da execução do seguinte comando (ver Código 11):

```
./run add gem 'tinymce-rails'
```

Código 11 – Comando para instalação TinyMCE

Essa `gem` disponibiliza os arquivos JavaScript e recursos necessários para a inicialização do editor dentro do ambiente Rails, sem a necessidade de dependências externas.

Em seguida, foi criado o arquivo de configuração em `app/javascript/controllers/tinymce_controller.js`, contendo a estrutura básica de inicialização do editor (ver Código 13):

A configuração apresentada estabelece parâmetros essenciais para o funcionamento do editor, incluindo:

- **Altura do editor** (`height`), ajustada para garantir uma área de escrita proporcional e visualmente equilibrada;
- **Desativação da barra de menu** (`menubar: false`), contribuindo para uma interface mais limpa e objetiva;
- **Padronização do estilo do conteúdo** (`content_style`), assegurando consistência tipográfica no texto exibido;

[4] Em projetos Ruby, uma *gem* é um pacote de código distribuído pela comunidade, utilizado para adicionar funcionalidades ou bibliotecas ao sistema de forma modular.

```
tinymce.init({
  target: this.element,
  license_key: "gpl",
  height: 300,
  menubar: false,
  relative_urls: false,
  remove_script_host: false,
  document_base_url: window.location.origin + "/",

  content_style: "body { color: #4B5563; }", // Cor padrão do texto

  toolbar: [
    // Elementos exibidos na barra de ferramentas
  ],

  plugins: [
    // Plugins necessários para o funcionamento dos recursos
  ],

  images_upload_url: "/educators/uploader/image", // Rota para upload
    de imagens
})
```

Código 12 – Configuração para o tinyMCE

- **Especificação de *plugins* e ferramentas**, responsáveis por funcionalidades como formatação, inserção de imagens, listas, tabelas e edição de expressões matemáticas;
- **Configuração da rota de upload de imagens**, permitindo que o usuário insira figuras diretamente no corpo do texto de forma integrada ao sistema.

Para garantir que o TinyMCE fosse carregado em todas as páginas necessárias, foi adicionada a seguinte linha no arquivo de *layout* principal, localizado em `app/views/layout/shared/_head.html.erb` (ver Código 13):

```
<%= tinymce_assets "data-turbo-track": "reload" %>
```

Código 13 – Carregamento do TinyMCE

Essa diretiva garante que os recursos de JavaScript e CSS do TinyMCE sejam corretamente incluídos no cabeçalho da aplicação, bem como recarregados sempre que ocorrer uma transição de página.

A integração do TinyMCE desempenha um papel central na criação dos elementos do OA, ao permitir que os usuários elaborem conteúdos matemáticos e científicos de maneira visual, estruturada e acessível. O editor possibilita a formatação de equações, símbolos especi-

ais, expressões matemáticas e textos explicativos — funcionalidades essenciais para materiais destinados ao ensino de Matemática e Física.

Além disso, sua adoção contribui significativamente para a usabilidade do sistema, tornando a interface mais intuitiva e favorecendo a experiência de professores e estudantes que interagem com a plataforma.

5.5 Atualizações e Melhorias

Nesta seção, são apresentadas as principais melhorias e atualizações técnicas implementadas na aplicação que tiveram como propósito otimizar a navegação e a coerência visual da interface, além de reforçar boas práticas de desenvolvimento, como a modularização, a reutilização de componentes e a padronização de elementos de interface.

5.5.1 Implementação de Trilhas de Navegação

Com o intuito de proporcionar uma navegação mais fluida e intuitiva, foram implementadas trilhas de navegação, também conhecidas como *breadcrumbs*. Essa funcionalidade tem como propósito indicar ao usuário em qual parte do sistema ele se encontra e quais caminhos percorreu até chegar à página atual. Dessa forma, o recurso permite uma navegação secundária mais intuitiva e fluida, facilitando o retorno a etapas anteriores ou o acesso a seções relacionadas. Em síntese, os *breadcrumbs* exibem a hierarquia do sistema desde a página inicial até a página atual, promovendo melhor orientação e usabilidade.

No contexto da FARMA, à medida que o usuário avança pelas etapas de criação de um OA, desde a introdução até a elaboração de dicas, torna-se essencial explicitar o fluxo de navegação. Assim, a implementação das trilhas de navegação reforça a noção de progressão e localização dentro do ambiente de autoria.

A estrutura principal dos *breadcrumbs* é representada pela classe definida em `app/controller/concerns/breadcrumbs.rb`, responsável por inicializar o nome da trilha e, opcionalmente, o seu link de destino (ver Código 14):

```
class Breadcrumb
    def initialize(name, path)
        @name = name
        @path = path
    end
end
```

Código 14 – Inicialização Breadcrumb

No mesmo arquivo, o módulo `Breadcrumbs` centraliza a lógica genérica de funcionamento das trilhas de navegação. Esse módulo redefine o método `render`, assegurando que os

breadcrumbs sejam carregados automaticamente antes da renderização de cada página (ver Código 16):

```
    def render(*args)
        default_breadcrumbs # Define o primeiro item da trilha
        set_breadcrumbs if respond_to?(:set_breadcrumbs, true) #
   Adiciona os breadcrumbs específicos de cada controller
    end
```

Código 15 – Modulo breadcrumbs - renderização

A rota padrão das trilhas é inicializada no arquivo `appcontrollereducatorsbase-controller.rb`, por meio de um método privado que adiciona o primeiro item da trilha, correspondente à página inicial do ambiente do educador (ver Código 16):

```
    def default_breadcrumbs
        add_breadcrumb I18n.t('educators.home.breadcrumbs'),
   educators_root_path
    end

    def add_lo_breadcrumbs(name, path)
        add_breadcrumb name, path
    end

    def set_breadcrumbs
        case action_name.to_sym
        when :new, :create
        add_lo_breadcrumbs(:new)
        when :edit, :update
        add_lo_breadcrumbs(:edit)
    end
end
```

Código 16 – Modulo breadcrumbs - renderização

Por fim, as trilhas de navegação são renderizadas e exibidas na interface da aplicação. A Figura 48 ilustra a forma como os *breadcrumbs* são apresentados, permitindo ao usuário visualizar claramente o caminho percorrido dentro do sistema.

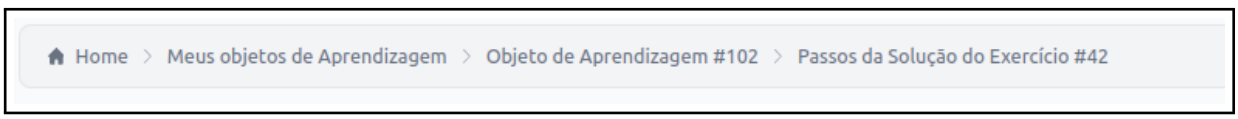

Figura 48 – Exemplo de *Breadcrumbs* renderizados

5.5.2 Padronização dos Títulos das Páginas

Com o objetivo de assegurar maior coerência e clareza na navegação do sistema, foi implementado o `helper`[5] `full_title`, responsável por padronizar a exibição dos títulos ao longo da aplicação. Essa funcionalidade garante que cada página apresente um título descritivo e consistente, alinhado tanto ao seu conteúdo quanto à identidade geral do sistema. Nas `Views`, o título de cada página é definido por meio da seguinte estrutura (ver Código 17):

```
<% provide(:title, t('.title')) %>
<title><%= full_title(content_for(:title)) %></title>
```

Código 17 – Estrutura para renderização de titulos

O `helper full_title` recebe como parâmetros o título específico da página e o título basen - geralmente, o nome da aplicação — retornando ambos combinados no formato:

`"Título da Página | Nome da Aplicação"`

Caso nenhum título seja definido na *view*, o método exibe apenas o nome base. A implementação é apresentada a seguir (ver Código 18):

```
def full_title(page_title = '', base_title = t('app.name'))
        page_title.blank? ? base_title : "#{page_title} | #{
   base_title}"
end
```

Código 18 – Codigo para renderização de titulos

Essa abordagem garante uma estrutura uniforme para os títulos em todas as seções da plataforma, contribuindo para uma navegação mais intuitiva e para uma melhor identificação das abas abertas no navegador. Além disso, reforça a consistência visual e semântica da aplicação, tornando a experiência de uso mais organizada e profissional.

5.5.3 Padronização de Ícones

Com o propósito de promover consistência visual e reutilização de componentes gráficos em toda a aplicação, foi desenvolvido o *helper* `IconsHelper`. Esse recurso centraliza a definição e o uso de ícones em formato SVG, permitindo sua inserção nas *views* sem repetição de código e sem necessidade de dependência de bibliotecas externas.

[5] Em aplicações Ruby on Rails, helpers são métodos auxiliares utilizados para encapsular lógicas que podem ser reutilizadas nas *views*, contribuindo para organização, clareza e redução de código duplicado.

A estrutura principal do `helper` está localizada no arquivo `apphelpersicons_helper.rb`, que declara o módulo `IconsHelper`. Em seu interior, encontra-se o dicionário `ICONS`, responsável por armazenar os ícones utilizados na aplicação em formato SVG, cada um identificado por uma chave simbólica. Essa abordagem facilita a manutenção, melhora a organização e garante a padronização dos elementos visuais ao longo do sistema (ver Código 19):

```
module IconsHelper
    ICONS = {
    edit: '<svg ...></svg>',
    trash: '<svg ...></svg>',
    home: '<svg ...></svg>',
    # outros ícones omitidos para brevidade
    }.freeze

    def icon(name, options = {})
        classes = options[:class] || 'w-4 h-4'
        icon = ICONS[name]&.sub('<svg', "<svg class='#{classes}'") ||
    ''
        sanitize_svg(icon)
    end
end
```

Código 19 – Estrutura para renderizar icones

O método `icon` é responsável por renderizar o SVG correspondente ao ícone solicitado. Ele recebe dois parâmetros: o nome simbólico do ícone (por exemplo, `:edit`, `:trash`, `:home`) e um conjunto de opções opcionais, como classes CSS adicionais para controle de tamanho, margem ou cor. O método substitui dinamicamente o atributo de classe dentro do SVG e retorna o código sanitizado, garantindo segurança e compatibilidade com o HTML. A utilização desse `helper` nas `Views` é feita de forma simples e declarativa, conforme o exemplo a seguir (ver Código 20):

```
<%= icon(:solution_step, class: "mr-2") %>
```

Código 20 – Exemplo de icone sendo aplicado

Esse comando insere o ícone definido pela chave `:solution_step` acompanhado da classe `mr-2`, responsável por adicionar espaçamento à direita. Tal implementação contribui para uma interface mais padronizada e de fácil manutenção, permitindo a reutilização de ícones em diferentes partes do sistema sem duplicação de código. De modo geral, o `IconsHelper` reforça a coerência estética da aplicação, melhora a legibilidade das interfaces e facilita a expansão futura do conjunto de ícones, bastando adicionar novos elementos ao dicionário `ICONS`.

5.6 Realização dos Testes

Durante o processo de desenvolvimento da aplicação, foi fundamental realizar testes em cada funcionalidade implementada. Para garantir a confiabilidade e o correto funcionamento do sistema, foram elaborados três tipos principais de testes: unitários, de controladores e de sistema. Os testes foram implementados utilizando o Minitest[6] e o Capybara[7].

5.6.1 Testes Unitários

Os testes unitários tiveram como objetivo verificar o funcionamento de partes isoladas do sistema, tais como modelos e regras de negócio específicas. Esses testes visam garantir que cada unidade de código opere corretamente de forma independente, evitando que falhas locais comprometam o funcionamento global da aplicação.

No contexto deste projeto, os testes unitários englobaram tanto validações quanto métodos personalizados e comportamentos internos das classes. Para tornar a escrita dos testes mais concisa e legível, foi utilizada a `gem Shoulda Matchers` [8]. Essa ferramenta proporciona uma sintaxe mais expressiva para testar validações do *ActiveRecord*, reduzindo redundâncias e tornando o código de teste mais fácil de manter.

O Código 21 apresenta um exemplo de teste unitário aplicado ao modelo `Lo`. Nesse caso, verifica-se que um OA não pode ser criado sem título e descrição, assegurando a consistência dos dados armazenados no sistema:

```
class LoTest < ActiveSupport::TestCase
  context 'validations' do
    should validate_presence_of(:title)
    should validate_presence_of(:description)
  end
end
```

Código 21 – Teste unitário do modelo `Lo`.

[6] Minitest é o *framework* de testes nativo do Ruby on Rails, caracterizado por sua simplicidade e rapidez. Disponível em: https://guides.rubyonrails.org/testing.html

[7] Capybara é uma ferramenta de automação utilizada para testes de aceitação em aplicações web, permitindo simular interações reais de usuários. Disponível em: https://github.com/teamcapybara/capybara

[8] Shoulda Matchers é uma coleção de *matchers* inspirados na abordagem do RSpec e compatíveis com Minitest e RSpec. A ferramenta facilita a escrita de testes para modelos e controladores no Rails, fornecendo uma sintaxe enxuta para testar validações e associações. Disponível em: https://thoughtbot.com/open-source/shoulda-matchers

5.6.2 Testes de Controladores

Os testes de controladores tiveram como objetivo verificar o correto funcionamento das ações definidas nos controladores da aplicação. Esse tipo de teste assegura que as rotas retornem o HTTP *status* apropriado, que redirecionamentos sejam executados corretamente e que mensagens de *flash* [9] sejam apresentadas de maneira adequada, sem envolver a interação direta do usuário com a interface.

O Código 22 apresenta um exemplo de teste aplicado ao controlador responsável pela gestão dos OA. No primeiro teste, `'should get /educators/los'`, verifica-se se a rota `educatorslos` responde corretamente com uma mensagem HTTP de sucesso. Já no teste `'should create /los/new'`, é verificado se o processo de criação de um novo OA é realizado com sucesso, incluindo o incremento na contagem de registros, o redirecionamento adequado e a exibição da mensagem de confirmação ao usuário:

```
class Educators::LosControllerTest < ActionDispatch::IntegrationTest
  test 'should get /educators/los' do
    get educators_los_path
    assert_response :success
  end

  test 'should create /los/new' do
    assert_difference('Lo.count', 1) do
      post educators_los_path, params: { lo: { title: 'Novo OA',
   description: 'Descrição OA' } }
    end

    assert_redirected_to educators_root_path
    assert_equal I18n.t('educators.los.create.success'), flash[:
   success]
  end
end
```

Código 22 – Teste de controlador para o gerenciamento de `Lo`.

5.6.3 Testes de Sistema

Por fim, os testes de sistema compõem a camada mais ampla de verificação, uma vez que simulam a interação real do usuário por meio do navegador. Utilizando o *Capybara*, esses testes reproduzem fluxos completos de uso, navegando entre páginas, preenchendo formulários, acionando botões e verificando o conteúdo apresentado na interface.

[9] No Ruby on Rails, o *flash* é um mecanismo de mensagens temporárias usado para fornecer feedback ao usuário após uma ação.

Essa abordagem possibilita identificar falhas decorrentes da integração entre diferentes componentes da aplicação, assegurando que o sistema opere de maneira consistente, integrada e estável em um cenário próximo ao real.

O Código 23 apresenta um teste de sistema que simula a criação de um novo OA. Nele, o *Capybara* é empregado para percorrer a interface de cadastro, preenchendo os campos necessários, anexando uma imagem, selecionando as opções disponíveis e submetendo o formulário. Ao final, o teste confirma se a criação foi concluída com sucesso, verificando a mensagem de feedback exibida e garantindo que o novo OA esteja devidamente listado na interface:

```
class Educators::LosControllerTest < ActionDispatch::IntegrationTest
  should 'successfully create a new LO' do
    fill_in :lo_title, with: 'Novo OA'
    fill_in :lo_description, with: 'Descrição do OA'

    attach_file 'lo_picture',
                Rails.root.join('test/fixtures/files/default_lo.png')
   ,
                visible: :all

    check :lo_accessible
    check :lo_duplicable

    click_on I18n.t('educators.los.new.submit')

    assert_current_path educators_root_path

    assert_text I18n.t('educators.los.create.success')

    assert_text 'Novo OA'
    assert_text 'Descrição do OA'
  end
end
```

Código 23 – Teste de sistema para o gerenciamento de Lo.

5.7 Métricas Gerais do Desenvolvimento

Ao longo do processo de desenvolvimento, a ferramenta FARMA passou por uma série de aprimoramentos progressivos, envolvendo ajustes estruturais, validações constantes e revisões incrementais. Durante esse período, foram realizadas 13 atividades principais, registradas em 153 commits, que resultaram em 8.970 alterações no repositório — considerando 5.083 linhas inseridas e 3.887 removidas. Esse volume de modificações evidencia um ciclo ativo de evolução do software e um cuidado contínuo com a melhoria das funcionalidades.

Do ponto de vista da arquitetura interna, o sistema alcançou um nível expressivo de organização e modularidade. Foram desenvolvidas 21 interfaces distintas, sustentadas por 13 controladores, além de 5 componentes reutilizáveis e 14 partials, os quais contribuem para a padronização visual e a manutenção do código.

Para garantir confiabilidade e reduzir a ocorrência de falhas, o projeto também incorporou 26 testes automatizados, distribuídos entre testes de controladores e de sistema, assegurando a verificação das principais operações da aplicação.

Em relação à composição tecnológica, o código-fonte apresenta predominância da linguagem Ruby (71,9%), seguida por HTML (24,5%), Shell (1,9%) e outros recursos complementares (1,7%). Esses dados foram obtidos a partir das métricas disponibilizadas pelo próprio GitHub e refletem o perfil técnico do desenvolvimento realizado.

6 CONSIDERAÇÕES FINAIS

A Ferramenta de Autoria para Remediação de Erros com Mobilidade na Aprendizagem (FARMA) foi concebida com o propósito de testar, avaliar e corrigir erros cometidos pelos estudantes por meio de Objetos de Aprendizagem (OA). Sua *versão original*[1] apresentava uma interface desatualizada e com limitações no uso cotidiano.

Posteriormente, foi desenvolvida uma versão baseada em uma *arquitetura de API*, que introduziu um conjunto de melhorias estruturais. No entanto, essa abordagem elevou a complexidade do sistema, o que poderia gerar dificuldades de manutenção e eventuais impactos no desempenho.

Diante desse contexto, o presente trabalho teve como objetivo projetar e implementar uma nova interface gráfica para a FARMA, acompanhada da migração de uma arquitetura distribuída para uma arquitetura monolítica. Essa escolha exigiu a reorganização da área de criação de OA, com foco na revisão dos fluxos e na simplificação das interações realizadas pelos educadores durante a elaboração dos materiais.

Com as alterações implementadas, a FARMA passou a operar em uma estrutura mais simples de manter e com interações reorganizadas para o usuário. A migração para a arquitetura monolítica contribuiu para uma gestão mais direta do sistema, enquanto a revisão da interface buscou tornar os processos de criação de OA mais claros e consistentes.

Por fim, além dos aspectos técnicos desenvolvidos, este trabalho também contribuiu ao reunir e sistematizar o histórico da FARMA, resultando na Linha do Tempo apresentada na Seção 2.1. Esse levantamento envolveu a consulta a documentos, publicações e versões anteriores da ferramenta, permitindo consolidar informações que estavam dispersas entre diferentes pesquisas, implementações e registros institucionais. A elaboração dessa linha histórica possibilita uma compreensão mais clara da trajetória da FARMA desde 2009, contextualizando as sucessivas reimplementações e destacando o papel deste projeto na continuidade e organização desse percurso.

Para a continuidade do desenvolvimento da FARMA envolve a expansão das funcionalidades e o aperfeiçoamento da estrutura existente. Um dos próximos passos é a implementação da visualização dos OA pelo estudante, etapa que complementa o escopo deste trabalho, voltado principalmente à área de criação e organização de conteúdos pelo professor. A visualização estudantil é necessária para completar o ciclo de utilização da ferramenta.

Além disso, algumas seções — como a criação de turmas e a área do usuário — não foram concluídas devido às delimitações de escopo e cronograma. Embora parcialmente representadas na interface, essas funcionalidades ainda precisam ser implementadas e estruturadas, constituindo possibilidades de continuidade em trabalhos futuros.

Essas evoluções podem ampliar as aplicações da FARMA no contexto educacional e fortalecer seu papel no apoio ao diagnóstico e à remediação de erros.

[1] Disponível em https://farma.educacional.mat.br/

REFERÊNCIAS

ARRUDA, S. Mudanças nas ideias prévias sobre ponto, reta e plano por meio da interação com a ferramenta farma. **Revista Brasileira de Ensino de Ciência e Tecnologia**, 2021.

AWARI., S. **As Vantagens E Desvantagens Do Figma: Tudo O Que Você Precisa Saber Sobre Essa Ferramenta De Design**. [*S.l.*]: https://awari.com.br/ as-vantagens-e-desvantagens-do-figma-tudo-o-que-voce-precisa-saber-sobre-essa-ferramenta-de-design, 2023.

BASS, L. **Software architecture in practice**. Addison-Wesley, 2013. Disponível em: https://www.amazon.com.br/Software-Architecture-Practice-Len-Bass/dp/0321815734.

BAZZO, G. **CLASSIFICAÇÃO AUTOMÁTICA DE ERROS DE APRENDIZES HUMANOS DO PROCESSO DE INDUÇÃO ANALÍTICA**. 2011. Tese (Doutorado) — Universidade Federal do Paraná, Curitiba - Paraná, 2011.

BONIN, D. Implementação de uma api para visualização e interação com objetos de aprendizagem para ferramenta de autoria farma. 2024.

CASTILHO, A. **Objetos de Aprendizagem com Feedbacks para a Autorregulação da Aprendizagem de Conceitos Matemáticos Necessários para o Cálculo Diferencial e Integral**. 2024. Tese (Doutorado) — Universidade Federal do Paraná, Curitiba - Paraná, 2024.

COSTA, D. **ATAUDIW - Uma Ferramenta de Autoria para Auxiliar o Uso da Lousa Digital Interativa**. 2017. Tese (Doutorado) — Universidade Federal de Goiás - Regional Jataí Instituto de Ciências Exatas e Tecnológicas (ICET), Goias, 2017.

DAMASCENO, M. d. G. d. A. *et al.* Potencializando a aprendizagem: A contribuiÇÃo da neurociÊncia e das tecnologias educacionais. p. 2409–2422, jul. 2024. Disponível em: https://periodicorease.pro.br/rease/article/view/14983.

DIRENE, A. **Objetos de aprendizagem generalizáveis para o currículo de Matemática do ensino médio**. 2009. Tese (Doutorado) — Universidade Federal do Paraná, Guarapuava - Paraná, 2009.

FOWLER, M. **Patterns of Enterprise Application Architecture**. Bookman, 2006. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9788577800643/pageid/19.

KLEPPMANN, M. **Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems**. O REILLY, 2017. Disponível em: https://www.amazon.com.br/Designing-Data-Intensive-Applications-Martin-Kleppmann/dp/1449373321.

KUTZKE, A. **Informática educacional e a mediação do erro na educação : um estudo teórico-crítico e uma proposta de instrumento computacional**. 2015. Tese (Doutorado) — Universidade Federal do Paraná, Curitiba - Paraná, 2015.

LARA, D. **FRedesign e Refatoração da Ferramenta de Autoria para a Remediação de Erros com Mobilidade na Aprendizagem (FARMA)**. 2017. Tese (Doutorado) — Universidade Tecnologica Federal do Paraná, Guarapuava - Paraná, 2017.

LEITE, M. **Arquitetura para remediação de erros baseada em múltiplas representações externas**. 2013. Tese (Doutorado) — Universidade Federal do Paraná, Curitiba - Paraná, 2013.

LEITE, M. Otimizando o processo ensino e aprendizagem com a arquitetura para desenvolvimento de objetos de aprendizage. **Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE 2015)**, 2015.

MARCZAL, D. **UM ARCABOUÇO QUE ENFATIZA A RETROAÇÃO A CONTEXTOS DE ERRO DURANTE O ACESSO A CONTEÚDOS EDUCACIONAIS**. 2010. Tese (Doutorado) — Universidade Federal do Paraná, Curitiba - Paraná, 2010.

MARCZAL, D. **Classificação automática de erros de aprendizes humanos do processo de indução analítica**. 2011. Tese (Doutorado) — Universidade Tecnologica Federal do Paraná, Guarapuava - Paraná, 2011.

MARCZAL, D. Farma: Uma ferramenta de autoria para objetos de aprendizagem de conceitos matemáticos. **C3SL – Departamento de Informática – Universidade Federal do Paraná (UPFR)**, 2012.

MARCZAL, D. **FARMA: Uma ferramenta de autoria para objetos de aprendizagem de conceitos matemáticos**. 2014. Tese (Doutorado) — Universidade Federal do Paraná, Curitiba - Paraná, 2014.

MARCZAL, D. Metodologia e software educacional para a investigação e remediação de erros conceituais em matemática. **Revista Brasileira de Informática na Educação**, 2016.

MOURA, V. **AVALIAÇÃO DO IMPACTO DA RETROAÇÃO NA APRENDIZAGEM APOIADA POR UMA FERRAMENTA EDUCACIONAL**. 2015. Tese (Doutorado) — Universidade Federal do Paraná, Curitiba - Paraná, 2015.

NEWMAN, S. **Building Microservices**. O REILLY, 2021. Disponível em: https://www.amazon.com/Building-Microservices-Designing-Fine-Grained-Systems/dp/1491950358.

RAMOS, J. Visualização de objetos de aprendizagem para a ferramenta de autoria farma. 2019.

RICHARDS, M. **Fundamentos da arquitetura de Software: Uma abordagem de engenharia**. Alta Books, 2024. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9788550819754/epubcfi/6/2[%3Bvnd.vst.idref%3Dcover]!/4/2/2%4050:2.

SANTOS, E. **Projeto de uma nova interface gráfica para a ferramenta de autoria de objetos de aprendizagem matemáticos FARMA**. 2018. Tese (Doutorado) — Universidade Tecnológica Federal do Paraná, Guarapuava - Paraná, 2018.

SILVA, R. **SEQUENCIAMENTO ADAPTATIVO DE EXERCÍCIOS BASEADO NA CORRESPONDNCIA ENTRE A DIFICULDADE DA SOLUÇÃO E O DESEMPENHO DINÂMICO DO APRENDIZ**. 2015. Tese (Doutorado) — Universidade Federal do Paraná, Curitiba - Paraná, 2015.

SILVA, R. Adaptabilidade de objetos de aprendizagem usando calibragem e sequenciamento adaptativo de exercícios. **Revista Brasileira de Informática na Educação – RBIE**, 2018.

SOUSA, R. **Tecnologias Digitais na Educação**. SciELO, 2011. Disponível em: https://static.scielo.org/scielobooks/6pdyn/pdf/sousa-9788578791247.pdf.

TAROUCO, L. **Objetos de aprendizagem : teoria e prática**. Evangraf, 2014. Disponível em: http://hdl.handle.net/10183/102993.

TRIBECK, P. Edutracesys como objeto de aprendizagem na remediação de erros. **Espacios**, 2017.

VITEK, I. Desenvolvimento da Área destinada À criação de objetos de aprendizagem para ferramenta de autoria farma. 2023.