UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Arthur Xavier Giffoni

PROJETO E IMPLEMENTAÇÃO DO MÓDULO DE COMUNICAÇÃO PARA SISTEMAS TUTORES INTELIGENTES

Uma abordagem formal usando Interaction Flow Modeling Language (IFML)

UBERLÂNDIA

2021

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Arthur Xavier Giffoni

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Orientador(a):Prof. Dr. Alexsandro Santos Soares
Universidade Federal de Uberlândia - UFU

Prof. Dr. Fabiano Azevedo Dorça
Universidade Federal de Uberlândia - UFU

Prof. Dr. Marcelo de Almeida Maia
Universidade Federal de Uberlândia - UFU

Uberlândia, Brasil
Junho de 2021

"Não confunda derrotas com fracasso nem vitórias com sucesso. Na vida de um campeão sempre haverá algumas derrotas, assim como na vida de um perdedor sempre haverá vitórias. A diferença é que, enquanto os campeões crescem nas derrotas, os perdedores se acomodam nas vitórias.

Roberto Shinyashiki

Agradecimentos

Agradeço aos meus pais pelo amor e apoio durante toda a caminhada. Aos meus irmãos pela amizade e carinho. À minha companheira que partilhou de todos os momentos na construção deste trabalho, pelo apoio e companheirismo. Aos meus tios e padrinhos por todo apoio, conselho e desejo de que eu fosse mais longe. Aos meus amigos e colegas que sempre estiveram ao meu lado confiando e apoiando. Ao meu orientador, por todo conhecimento que me foi transmitido e dedicação que conduziu todo o processo da construção deste. A todos aqueles que contribuíram de alguma forma e a mim mesmo por ter enfrentado desafios e noites em claros, seguindo em frente e realizado um trabalho que me deixa feliz por ter concluído, meu agradecimento.

Resumo

Este trabalho apresenta a modelagem de um módulo de comunicação e interface de um Sistema Tutor Inteligente (STI) usando a Interaction Flow Modeling Language (IFML). Para a especificação de tal módulo, foi realizada uma revisão sistemática da literatura sobre as interfaces utilizadas em diferentes STIs que adotou como critério de comparação as heurísticas de Nielsen. Visando compreender as vantagens e desvantagens da IFML, um sistema web para a gestão da aprendizagem foi modelado e implementado. Há claramente uma vantagem a favor de uma linguagem de modelagem quando comparada às práticas informais largamente empregadas no desenvolvimento de interfaces de comunicação de STIs.

Palavras-chave: Sistema Tutor inteligente; STI; Modulo de Comunicação; Modulo de Interface; Interação Humano-Computador; Interface de Usuário; Experiência de Usuário; Engenharia de Software; Linguagem de Modelagem; Linguagem de Modelagem de Fluxo de Interação; IFML; React.js; LMS; Ciência da Computação; Revisão Sistemática da Literatura.

Abstract

This work presents a modeling of a communication module and interface of an Intelligent Tutor System (STI) using an Interaction Flow Modeling Language (IFML). For a specification of such a module, a systematic review of the literature was carried out on the interfaces used in different STIs that adopted Nielsen's heuristics as a criterion for comparison. In order to understand the advantages and disadvantages of IFML, a web system for learning management was modeled and implemented. There is clearly an advantage in favor of a modeling language when compared to informal practices widely used in the development of STI communication interfaces.

Keywords: Intelligent Tutoring System; STI; Communication Module; Interface Module; Humancomputer interaction; User Interface; User experience design; Software Engineering; Modeling language; Interaction Flow Modeling Language; IFML; React.js; LMS; Computer Science; Systematic literature review;

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
CRUD	Create, Read, Update and Delete
CRUD	Creante, Read, Update and Delete
DOM	Document Object Model
EMF	Eclipse Modeling Framewor
ES6	ECMAScript 6
IFML	Interaction Flow Modeling Language
IHC	Interação Humano Computador
ITS	Intelligent Tutoring Systems
LMS	Learning Manage-ment Systems
MDWE	Model Driven Web Engineering
OMG	Object Management Group
ORM	Object Relational Mapping
RSL	Revisão Sistemática da Literatura
STI	Sistema Tutor Inteligente
UI	User Interface
URL	Uniform Resource Locator

Sumário

1 Introdução

Cada vez mais, o processo de ensino-aprendizagem vem sendo apoiado por recursos tecnológicos, que possibilita diferentes meios de comunicação entre indivíduo e máquina. A aplicação destes recursos na área educacional vem fortalecendo o laço entre inteligência artificial (IA) e o aprendizado personalizado de um indivíduo. As transformações do uso desses novos recursos, especialmente aqueles ligados à softwares inteligentes, instigam que se repense profundamente a maneira tradicional de se instruir um indivíduo.

Atualmente, são encontrados disponíveis vários tipos de sistemas educacionais, destacando-se entre estes os chamados Sistemas Tutores Inteligentes (STIs). Estes sistemas possuem a capacidade de ensinar e se adaptar, de forma a adequarem sua estratégia de ensino às necessidades de aprendizagem de cada estudante. O Sistema rastreia o desempenho de um estudante numa determinada tarefa e oferece retornos e comentários personalizados durante sua realização. Ao coletar tais informações, o sistema pode realizar inferências sobre as deficiências particulares do aluno e assim adequar a rota de atividades. De acordo com WOOLF (2010), a arquitetura clássica de um sistema tutor inteligente possue os seguintes módulos: módulo do estudante, módulo do domínio, módulo pedagógico e módulo da comunicação.

Este trabalho dará enfoque ao módulo da comunicação, que é responsável pela apresentação do conteúdo e captura de informações do aluno (usuário). O módulo deve se atentar a apresentar as informações de forma rápida, agradável e com riquezas de recursos, não deixando o aluno entediado, como também deve executar o monitoramento das informações em segundo plano, não onerando o aluno com questionários excessivos. Neste trabalho também será executado uma pesquisa intensa na área de engenharia de software, com mais atenção na linguagem de modelagem utilizada. A partir desta, será realizado a modelagem de um modelo de comunicação utilizando a Linguagem de Modelagem de Fluxo de Interação ou Interaction Flow Modeling Language (IFML) para atender à modelagem do mesmo.

1.1 Questão de Pesquisa

A partir da análise de outros sistemas tutores, será levantado como são realizados a modelagem do modelo de comunicação e quais abordagens foram utilizadas no processo. Com isso, o trabalho levantará as seguintes questões:

- Existe um modelo detalhado do módulo de comunicação de um Sistema Tutor Inteligente (STI)?

- Se existe, qual técnica de modelagem web é utilizada para representar a arquitetura do modelo de comunicação?
- É possível usar métodos formais de modelagem web para modelar e implementar a interface gráfica, assim, um modelo de comunicação de um STI?
- A utilização do IFML para modelagem facilita o desenvolvimento de um sistema e a sua replicabilidade?
- Existe alguma implementação de um modelo de comunicação de um STI acessível para estudo?

1.2 Hipótese

A definição de uma arquitetura de software adequada para replicabilidade pode ser alcançada por meio da utilização de técnicas formais como IFML. Ser possível modelar um modelo de comunicação de um STI utilizando IFML. Ser possível a implementação gráfica de um STI a partir da utilização de métodos formais de modelagem web.

1.3 Objetivo Geral

Definir uma arquitetura para o modelo de comunicação de um sistema tutor inteligente utilizando as técnicas formais de modelagem de software IFML.

1.4 Objetivo Específico

1. Revisar a literatura acerca de Sistemas Tutores Inteligentes, Engenharia Web, Engenharia de Software e IFML.
2. Modelar o fluxo de navegação utilizando uma técnica formal.
3. Validar modelo da arquitetura por meio do desenvolvimento de um protótipo.
4. Analisar a replicabilidade do projeto a partir das modelagens utilizadas.

1.5 Justificativa

De forma geral, cabe apontar que os STIs não apresentam um modelo de comunicação e interface de forma definida e formalizada na literatura. Portanto, a replicabilidade na reimplementação de um sistema desses se torna complexa, tornando a colaboração de futuros trabalhos mais árduos.

1.6 Organização do Trabalho

O capítulo 2 introduz os principais conceitos sobre STI e quais técnicas utilizadas para a modelagem do projeto. Além disso, apresenta os conceitos de usabilidade e sobre a modelagem para aplicações web utilizando IFML.

O capítulo 3 contém a Revisão Sistemática da Literatura. Além da fundamentação da RSL, estão descritos o protocolo de pesquisa e os resultados obtidos através da pesquisa realizada.

O capítulo 4 apresenta os requisitos e casos de usos identificados, com o objetivo de auxiliar na definição de arquitetura e desenvolvimento do projeto.

O capítulo 5 detalha a arquitetura proposta e o processo de desenvolvimento do projeto, bem como apresenta os modelos UML e IFML do projeto.

Por fim, o capítulo 6 fornece uma análise das conclusões obtidas das contribuições alcançadas e exibe os trabalhos futuros a serem realizados.

2 Fundamentação Teórica

Como apontado anteriormente, neste capítulo aborda os aspectos teóricos relacionados à proposta deste trabalho, possibilitando uma melhor compreensão do tema. Nesse sentido, o capítulo estrutura-se da seguinte forma: a seção 2.1 apresenta uma definição de Sistemas Tutores Inteligentes e a arquitetura de seus principais módulos componentes; a seção 2.2 apresenta conceitos sobre usabilidade de software; 2.3 discorre sobre heurísticas de Nielsen; os paradigmas de modelagem de software são discutidos na seção 2.3; nas seções 2.4 e 2.5, são discutidos, respectivamente sobre a modelagem para aplicações web e sobre a ferramenta IFML que será utilizada para modelar; 2.6 apresenta o processo de modelagem de design de uma aplicação; a seção 2.7 apresenta sobre os conceitos de arquitetura REST; nas seções 2.8 e 2.9, são discutidos, respectivamente sobre a modelagem de dados e sobre a estruturação de um dicionário de dados; 2.10 apresenta conceitos do modelo UML e 2.11 apresenta detalhes da arquitetura MVC.

2.1 Sistemas Tutores Inteligentes

Segundo FOWLER (1991), os Sistemas Tutores Inteligentes (STIs) são programas de computador que possuem propósitos educacionais e que utilizam técnicas de Inteligência Artificial (IA). Eles podem simular o processo do pensamento humano, por meio da IA, para auxiliar um estudante no processo de ensino-aprendizagem em um determinado domínio ou em tomadas de decisões. Genericamente, os STIs conseguem representar separadamente a matéria que se ensina e as estratégias para ensiná-la. Além disso, possuem a capacidade de mapear as características específicas de cada estudante. SOTTILARE (2015) aponta algumas características ideias de um STI como:

- **Autorregulação**: suporte ao aprendizado tanto de indivíduos quanto de times.
- **Adaptabilidade**: modelagem das instruções de acordo com as necessidades e preferências do(s) usuário(s).
- **Exatidão**: utilização de métodos de instruções precisos.
- **Usabilidade**: acessibilidade à usuários com diferentes necessidades.
- **Disponibilidade**: disponível ao usuário sempre que for de sua vontade.

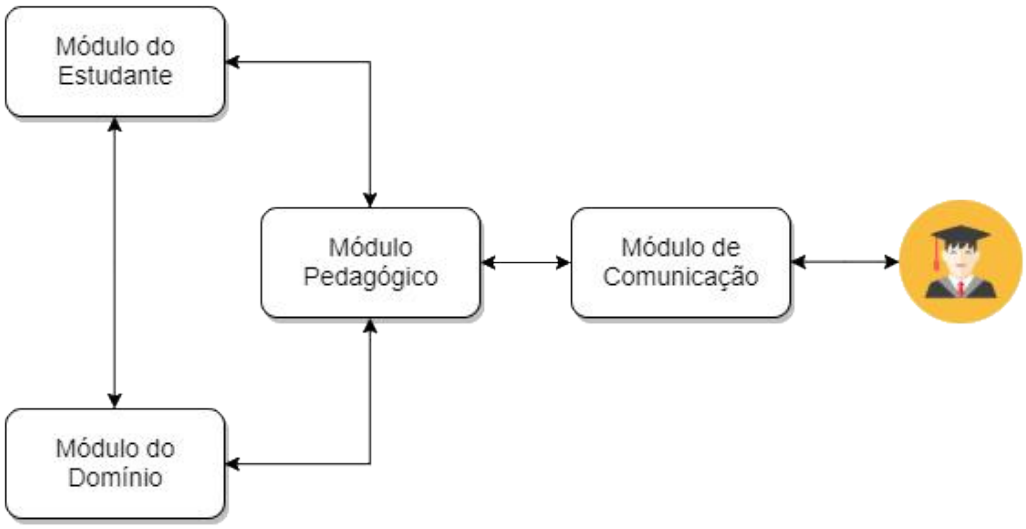
No aspecto de Usabilidade, SOTTILARE (2015) ainda defende que o STI deve ser adaptável ao estilo de aprendizagem do aluno, ou seja, se ele prefere que o conteúdo seja apresentado em forma de texto ou em forma de vídeo. Dessa forma, MORAIS; SCHAAB;

JAQUES (2017) argumentam que a principal forma de comunicação entre o usuário e um programa de computador, assim como STIs, é por meio da interface gráfica do usuário. Sendo assim, SEFFRIN (2015) salienta que, a interface de comunicação de um STI deve ser um módulo bem planejado e de fácil manipulação, favorecendo o processo de comunicação entre tutor e aluno.

2.1.1 Arquitetura

Existem várias abordagens específicas em relação à arquitetura de um STI, no entanto WOOLF (2010) discuti sobre uma arquitetura clássica, representada na Figura 1. Tal arquitetura é composta por quatro componentes presentes na maioria dos STIs, o módulo do domínio, o módulo do estudante, o módulo pedagógico e o módulo da comunicação. É importante destacar que esta arquitetura pode ser adaptada de acordo com o domínio e as necessidades especificas observadas em uma aplicação.

Figura 1 – Arquitetura Clássica de um Sistema Tutor Inteligente



Fonte: WOOLF (2010)

O **Módulo do Domínio** representa o conhecimento do especialista em um determinado domínio e contém as medidas de aprendizado e o desempenho do aluno. Este módulo é utilizado para avaliar o progresso do estudante em relação as expectativas (SOTTILARE, 2015). Com isso, o módulo combina essas informações com recomendações estratégicas do Módulo Pedagógico para executar uma ação inteligente direcionada ao aluno. É fundamental estruturar de forma organizada o conhecimento com o objetivo de viabilizar a escalabilidade do domínio e uso eficiente dos recursos (DORÇA, 2004).

O **Módulo do Estudante** armazena todas as informações individuais relevantes sobre o estudante, tais como as suas preferências, o nível de conhecimento, os objetivos e o

histórico de navegação, assim possibilita uma experiência personalizada. Logo, partir desse modelo e do conteúdo a ser ensinado, o sistema deve ser capaz de inferir a melhor estratégia de ensino a ser utilizada (JUNIOR; FREITAS et al., 2018). O histórico de aprendizagem do estudante é acessado pelo módulo pedagógico durante o processo de seleção dos objetos de aprendizagem mais adequados para seu sequenciamento de conteúdo (DORÇA, 2004).

O **Módulo Pedagógico**, por sua vez, determina as estratégias de ensino mais adequadas ao estudante com base nas informações contidas no sistema, no Módulo do Domínio e no Módulo do Estudante (DORÇA, 2004). Tais estratégias são responsáveis por determinar quando intervir na aprendizagem, com base no conhecimento presumido do estudante e suas ações. Estas ainda devem ser personalizadas conforme as necessidades individuais: para um estudante pode ser necessário estimular a sua motivação, enquanto um outro pode necessitar de alguma ajuda cognitiva ou treinar habilidades básicas (JUNIOR; FREITAS et al., 2018). Este módulo também é o responsável por realizar as correções dos exercícios resolvidos pelo estudante, selecionar as atividades e exemplos para exibição , retorno de informações sobre o seu desempenho e avaliação do progresso no domínio do conhecimento do sistema (CRUZ et al., 2009).

Por fim, o **Módulo de Comunicação**, o qual é o foco desse trabalho, é responsável pela interação entre estudantes e computadores, como interfaces gráficas, agentes animados ou mecanismos de diálogo (WOOLF, 2010). Não há um grau de complexidade definido para a sua implementação, entretanto, é importante considerar que uma interface de usuário bem desenhada pode impactar significativamente na clareza com a qual o estudante absorve as instruções e informações retornadas pelo sistema (BUTZ; HUA; MAGUIRE, 2006).

2.1.2 Modulo de Comunicação

Como descrito na seção anterior, SOTTILARE (2015) argumenta que uma característica fundamental dos Sistemas Tutores Inteligentes é a sua capacidade de se adaptar a cada aluno que utiliza o sistema. Dessa maneira o Módulo de Comunicação se torna essencial na experiência do aluno, pois tem a responsabilidade de apresentar a interface de interação de forma adaptativa, ou seja, caso um aluno tenha mais facilidade ou gosto em estudar por meio de vídeos, o STI deve apresentar um roteiro de estudo com maior predominância de vídeos e a interface deve ser dinâmica para isto.

Para realizar e apresentar tal adaptação, é necessário obter todas as informações possíveis sobre o aluno. De fato, é importante que cada vez que o aprendiz interaja com o STI, este forneça uma representação do estado atual do conhecimento, como também o gosto do aluno, a fim de selecionar o material a um nível adequado de detalhes para o aprendiz, propondo uma interação mais apropriada naquele momento e tornando a experiência mais agradável e produtiva. Este monitoramento frequente deve ser realizado

o máximo possível em background, para não ocupar o aluno com questionários excessivos.

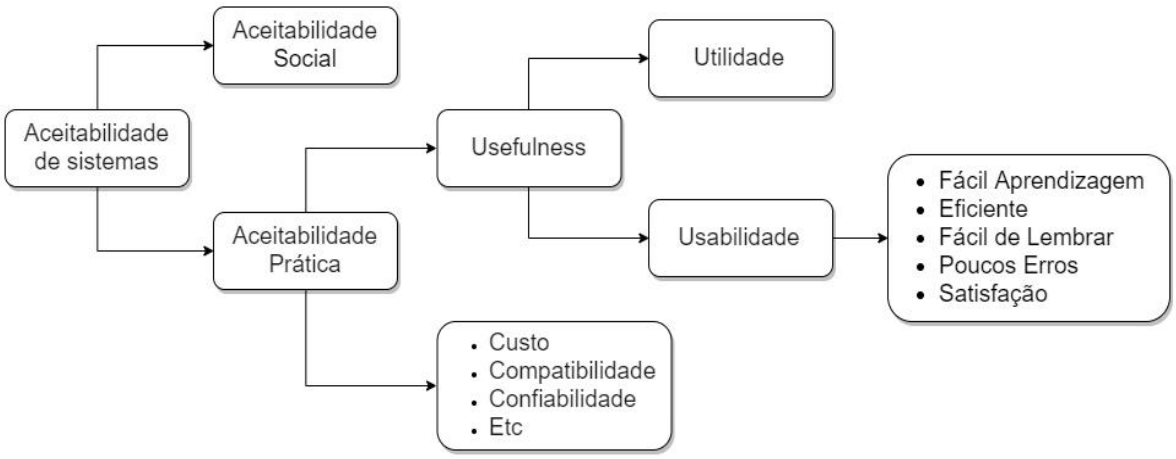
Segundo NIELSEN; MACK et al. (1994), a experiência com a interface deve possuir características de usabilidade satisfatória, como facilidade de aprendizagem, facilidade de utilização e satisfação subjetiva. Com a experiência adaptada ao aluno e apresentação do conteúdo com riqueza de recursos, espera-se que o módulo consiga evitar que o aluno fique entediado ou desmotivado, diminuindo as taxas de evasão do curso.

2.2 Usabilidade de Software

O termo usabilidade faz parte do vocabulário técnico da Ciência da Computação, dentro da área de Interação Humano Computador (IHC), se refere à qualidade da interação entre sistemas e usuários FREITAS; DUTRA (2009). Essa qualidade da interação depende de alguns aspectos, como a facilidade em aprender, a eficiência e a satisfação do usuário.

Usabilidade é um conceito dentro de um grupo chamado de aceitabilidade do sistema NIELSEN; MACK et al. (1994), que combina a aceitabilidade social com a aceitabilidade prática, que está relacionada diretamente à IHC. A aceitabilidade social diz respeito, principalmente, aos impactos sociais que o software pode provocar aos usuários, por exemplo, o sistema de controle de entrada para embarque aéreo, o qual, apesar de benéfico, não é totalmente aceito socialmente, pois dificulta às pessoas adentrarem à área de embarque. Já a aceitabilidade prática está relacionada diretamente a quais técnicas de software são utilizadas, como confiabilidade, custo, compatibilidade, utilidade e usabilidade (Figura 2).

Figura 2 – Atributos de Aceitabilidade de Sistemas



Fonte: Adaptado de NIELSEN; MACK et al. (1994)

NIELSEN; MACK et al. (1994) apontam o que conceito de usabilidade possui características que contribuem para a qualidade da interação entre o sistema e o usuário, sendo elas:

- **Aprendizagem**: quão fácil é para os usuários atingirem um determinado nível de familiaridade com o sistema.
- **Eficiência**: após o aprendizado, quão ágil e correto a modelagem do sistema possibilita a realização das tarefas.
- **Reminiscência**: após um período sem o usuário interagir com o sistema, o quão facilmente se consegue restabelecer proficiência.
- **Erros**: o quão minimizado é a ocorrência de erros. Quão graves são esses erros e quão facilmente se conseguem recuperar dos erros.
- **Satisfação**: quão agradável é utilizar o sistema.

Tais conceitos possibilitam uma validação e qualificação da usabilidade de um sistema. FREITAS; DUTRA (2009) argumentam que nesse processo é necessária uma avaliação da usabilidade, para isso os principais métodos utilizados são:

- **Métodos analíticos**: análises baseadas em regras, recomendações e princípios. Não dependem de participação de usuários;
- **Participação Direta**: testes realizados com usuários, seja através de questionários e entrevistas;
- **Observação Controlada**: análise baseada na observação do uso do sistema.

2.3 Heurísticas de Nielsen

Fortemente conhecidas ao se tratar da usabilidade de interfaces, as Heurísticas de Nielsen são dez regras gerais de usabilidade propostas pelo próprio NIELSEN (1994), as quais são amplamente utilizadas em avaliações de interface de software. Tais regras são usadas por arquitetos de informação e designers de interação em um método de inspeção de interfaces chamado avaliação heurística. De modo rápido, barato e fácil, como também podendo ser aplicado por qualquer equipe de desenvolvimento, tem como objetivo encontrar problemas de utilização na concepção, de modo que estes possam ser atendidos como parte do processo interativo de modelagem de software (SOUZA; GIGLIO, 2015).

Independentes de tecnologia específica e resultantes de uma base de problemas comuns entre muitos sistemas, as dez Heurísticas de Usabilidade, por NIELSEN (1994) são:

2.3.1 Visibilidade do status do sistema

É responsabilidade do sistema sempre informar os usuários sobre o que está acontecendo, por meio de *feedback* apropriado dentro de um prazo razoável.

Como mostrado na Figura 3, quando estamos assistindo/ouvindo uma playlist do Youtube, do lado direito fica claramente exposto qual vídeo estamos assistindo, qual é próximo e quais assistimos ou não.

Figura 3 – Heurística visibilidade do status do sistema

Fonte: Youtube. Vídeo "Qual a diferença entre Inteligência Artificial, Machine Learning, Data Science, Deep Learning, etc?".[1]

2.3.2 Correspondência entre o sistema e o mundo real

O sistema deve falar o idioma do usuário, com palavras, frases e conceitos familiares ao usuário, ao invés de termos orientados ao sistema. É necessário que ele siga as convenções do mundo real, fazendo as informações aparecerem em uma ordem natural e lógica.

Figura 4 – Heurística correspondência entre o sistema e o mundo real

Fonte: Adobe Photoshop Cs6 (2021)

[1] Disponível no canal Filípe Deschamps. Acesso em: 22 de maio de 2020

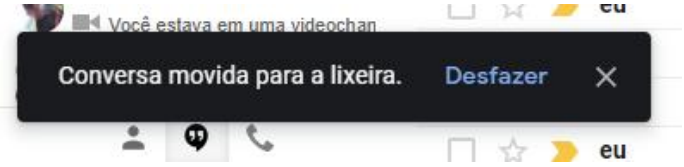
Como ilustrado na Figura 4, é notório esta heurística em vários sistemas, por exemplo, ao utilizar uma seta, ícones (tesoura para função cortar), cor vermelha para elementos negativos, entre outros.

2.3.3 Liberdade de controle fácil pro usuário

Os usuários geralmente escolhem as funções do sistema por engano e precisam de uma "saída de emergência"claramente marcada para deixar o estado indesejado sem ter que passar por um diálogo prolongado, ou seja, um suporte para desfazer e refazer.

Essa liberdade, em algumas aplicações, é nitidamente limitada pela regra de negócio, por exemplo, não possuir a função editar Tweets na plataforma Twitter, pelo fato de alguém ter compartilhado e assim ficar refém a edição do proprietário.

Figura 5 – Heurística liberdade de controle fácil pro usuário



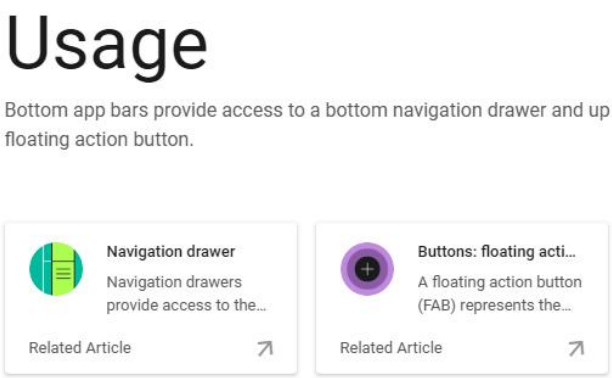
Fonte: Gmail Google (2021), imagem do componente utilizado para desfazer uma ação no Gmail.

A figura 5 apresenta outro exemplo que ocorre quando deletamos um e-mail na plataforma Gmail, a qual apresenta uma opção instantânea para desfazer o processo, e recuperar o e-mail que pode ter sido deletado por engano.

2.3.4 Consistência e padrões

Os usuários não devem se perguntar se palavras, situações ou ações diferentes têm o mesmo significado. Dessa forma é importante manter a consistência e padrão visual (texto, cor, desenho do elemento, som etc).

Figura 6 – Heurística consistência e padrões



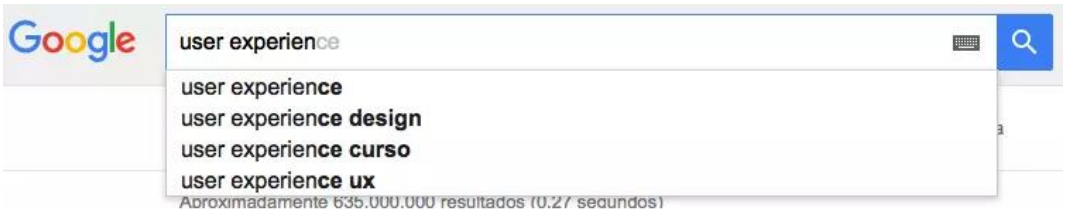
Fonte: Material.io Google (2021)

Por exemplo, o site Material Design de apresentação de componentes, apresenta padronização nos botões, fontes e estrutura de navegação como mostra a figura 6.

2.3.5 Prevenções de erros

Melhor do que boas mensagens de erro, é um design cuidadoso que evita a ocorrência de um problema. Elimine condições propensas a erros ou as verifique e apresente aos usuários uma opção de confirmação antes que eles se comprometam com a ação.

Figura 7 – Heurística Prevenções de erros pré pesquisa



Fonte: Barra de pesquisa do Google (2021)

Por exemplo, a busca do Google sobre alguma palavra, além de sugerir possíveis palavras no momento de digitação (figura 7), também sugere correções ortográficas após a realização da busca (figura 8).

Figura 8 – Heurística Prevenções de erros após pesquisa

Fonte: sugestão de resultados parecidos do Google (2021)

2.3.6 Reconhecimento em vez de memorização

O usuário não tem obrigação de decorar qual foi o caminho que ele fez pra chegar até a página. Dessa forma, é necessário minimizar a carga de memória do usuário, tornando objetos, ações e opções visíveis. Sendo assim, as instruções de uso do sistema devem ser visíveis ou facilmente recuperáveis sempre que necessário.

Por exemplo, quando você entra em um produto do site da Locaweb é disponibilizado o caminho que você fez pra chegar até ele, ilustrado na figura 9, essa funcionalidade é popularmente chamada de *breadcrumb*.

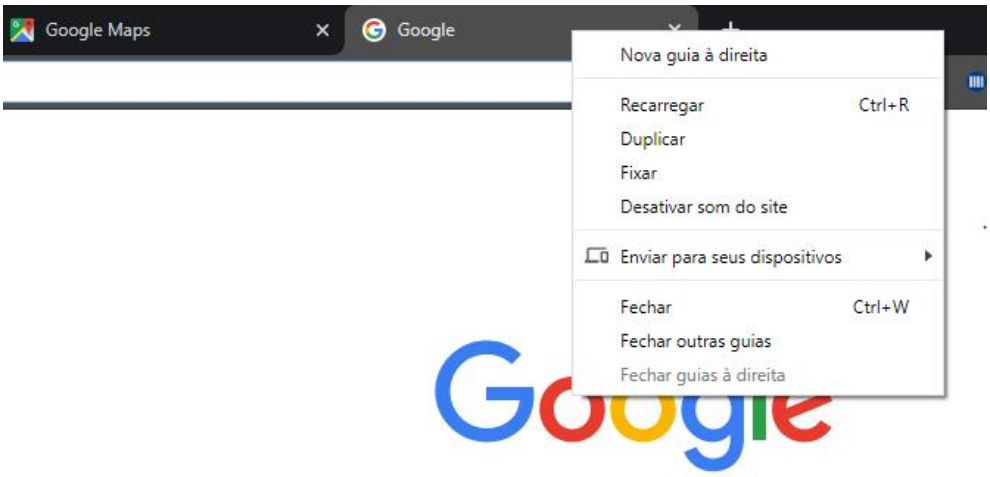
Figura 9 – Heurística Reconhecimento em vez de memorização

Fonte: LocaWeb (2021)

2.3.7 Flexibilidade e eficiência de uso

Aceleradores, mesmo que invisíveis para o usuário iniciante, geralmente aceleram a interação do usuário experiente, de modo que o sistema atenda usuários inexperientes e experientes. Permitindo que os usuários adaptem ações frequentes.

Figura 10 – Flexibilidade e eficiência de uso



Fonte: funções da aba do navegador Google Chrome (2021)

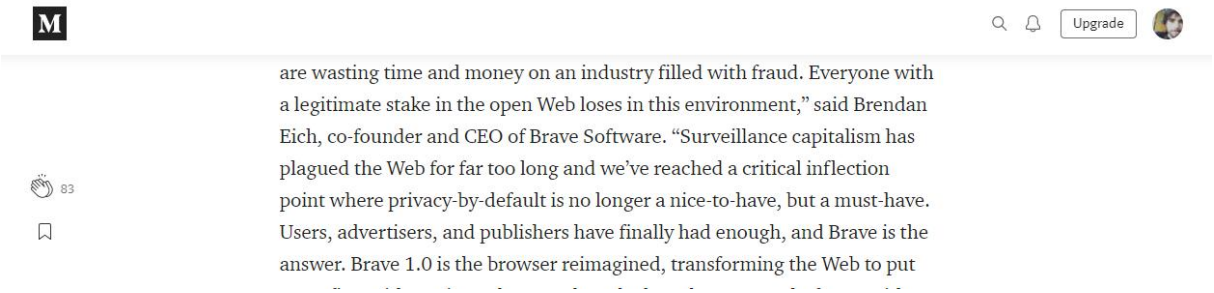
Um exemplo muito utilizado que atende esta heurística, é a utilização e adaptação de teclas de atalho nos sistemas. Como apresentado na figura 10 a função de fechar uma aba no navegador Google Chrome, onde pode ser utilizado click com o scroll do mouse ou botão direito e selecionar a opção.

2.3.8 Estética e design minimalista

Quanto maior for a quantidade de informações dispostas, maior será o tempo gasto de análise e isso pode dificultar a tomada de decisão do usuário. Portanto, é importante fazer uma classificação das informações entre aqueles que são cruciais (primeiro plano) e

as que são necessárias (segundo plano) tornando assim, a aplicação eficiente no objetivo de transmitir uma informação.

Figura 11 – Estética e design minimalista



Fonte: Medium (2021)

Um exemplo é o Medium, o qual possui uma estética minimalista, oferecendo o essencial ao usuário em um momento de leitura de um artigo publicado. Como representado na figura 11, as opções oferecidas são: "curtir"o artigo, salvar, acessar suas notificações e pesquisa.

2.3.9 Ajude os usuários a reconhecerem, diagnosticarem e recuperarem-se de erros

As mensagens de erro devem ser expressas em linguagem simples (sem códigos), indicar com precisão o problema e sugerir construtivamente uma solução.

Figura 12 – Ajude os usuários a reconhecerem, diagnosticarem e recuperarem-se de erros

Fonte: Gmail (2021) área de login

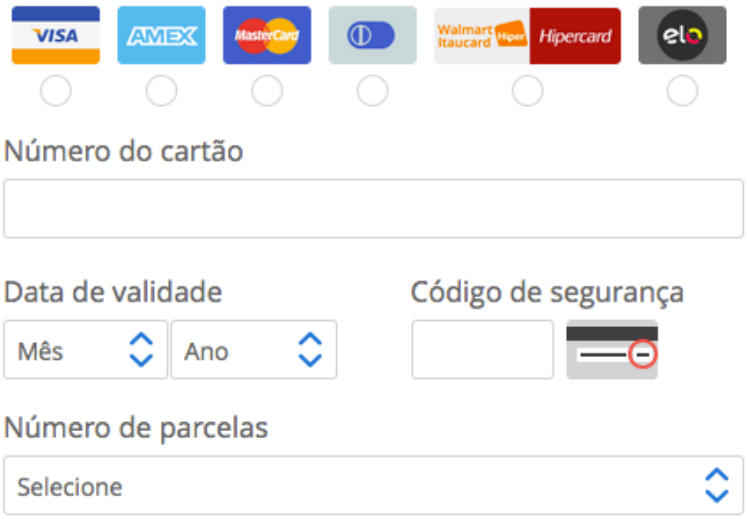
Abordado normalmente no processo de login, o Google atende essa heurística como representado na figura 12, ao informar um email ou número de telefone inválido na tentativa de realizar login.

2.3.10 Ajuda e documentação

Mesmo que seja melhor se o sistema puder ser usado sem documentação, pode ser necessário fornecer ajuda e documentação. Essas informações devem ser fáceis de pesquisar, focadas na tarefa do usuário, listando etapas concretas a serem executadas, sem serem muito grandes.

Por exemplo, no formulário de pagamento do Walmart, na figura 13, tem um campo pra preencher o código de segurança do cartão. Como não é algo muito óbvio, tem uma imagem próxima ao campo mostrando onde fica o código de segurança do cartão. Essa é uma boa forma de fazer uma documentação com facilidade.

Figura 13 – Ajuda e documentação



Fonte: Walmart (2021)

2.4 Modelagem de software

Esta seção apresenta a relevância da modelagem para o desenvolvimento de software. São destacados conceitos que descrevem como um modelo de software influencia diretamente no produto final desenvolvido.

2.4.1 Modelagem

Modelos são a simplificação da realidade. Um modelo de software é um produto de uma metodologia de modelagem de software, que representa de forma simplificada e detalhada o software em sua totalidade. CASARIN (2014) argumenta que com os modelos pode-se realizar planos detalhados, como planos mais gerais, que apresentem uma visão panorâmica do sistema. Dentro deste contexto, pode-se salientar que por meio de modelos, auxiliados pela diagramação, facilitase-se a visão do sistema como um todo.

Sobre o mesmo assunto, ESPÍNDOLA (2016) defende que um bom modelo pode incluir detalhes e componentes de grande importância, omitindo componentes menores que não necessitam de representação em determinado nível de abstração.

A modelagem de software é a parte importante de construir modelos de software na fase de análise de requisitos. Segundo CASARIN (2014), a modelagem de software permite um melhor entendimento das questões arquiteturais e comportamentais do problema a ser resolvido. Ainda afirma que uma boa modelagem de software deve permitir a representação da informação a ser transformada pelo software, das funções (ou sub-funções) responsáveis pelas transformações e do comportamento do sistema durante a ocorrência destas transformações.

A modelagem visual (com o auxílio de diagramas) ajuda a manter a consistência entre os artefatos (produtos) ligados ao desenvolvimento de um sistema: requisitos, projeto e implementação. TACLA (2007) resume que a modelagem visual pode melhorar a capacidade de uma equipe de gerenciar a complexidade do software.

O desenvolvimento de um sistema, inicia-se através de um projeto, dessa forma, como parte deste projeto, ele deve ser modelado como um conjunto de componentes que se relacionam entre si. CASARIN (2014) argumenta que em muitas vezes não é utilizado nenhum método de modelagem formal. Em alguns casos, analistas utilizam desenhos em papel para exemplificar partes do sistema ou mesmo o sistema com um todo. Porém, essa modelagem informal não oferece um padrão de linguagem que pode ser compreendido facilmente por outras pessoas.

Na construção de qualquer tipo de sistema há uma gradação de complexidade. BEZERRA (2015) diz que para a construção de sistemas de software mais complexos, é necessário um planejamento inicial anterior, equivalente ao processo de planejamento de projeto de uma planta de uma casa. O autor ainda apresenta algumas situações e razões pelas quais deve-se utilizar modelos na construção de sistemas, sendo elas:

- **Gerenciamento de complexidade**: principal razão pela utilização de modelos, esta supri as limitações humanas em lidar com complexidade. Por meio desses modelos, os indivíduos envolvidos no desenvolvimento do sistema podem fazer estudos e predizer seus comportamentos. Cada modelo representa uma perspectiva do sistema, dessa forma, características relevantes à resolução de um problema devem ser consideradas, pois modelos revelam as características essenciais de um sistema. Detalhes irrelevantes que podem dificultar o entendimento podem ser ignorados por um momento, estudando-se separadamente cada um dos modelos.

- **Comunicação entre pessoas envolvidas**: o desenvolvimento de um sistema envolve a execução de atividades, as quais se traduzem em informações sobre ele. Essas informações correspondem aos modelos criados para representar tal sistema. Então,

os modelos servem para promover a difusão de informações relativas ao sistema entre os indivíduos envolvidos em sua construção já que estes servem com um ponto de referência comum.

- **Redução nos custos de desenvolvimento**: no desenvolvimento de sistemas, podem ocorrer erros, tanto individuais como de comunicação entre a equipe. A correção desses erros é menos custosa quando detectada e realizada ainda no(s) modelo(s) do sistema. Modelos são mais baratos de construir do que sistemas. Consequentemente, erros identificados sobre modelos têm um impacto menos desastroso.
- **Predição de comportamento futuro do sistema**: o comportamento do sistema pode ser discutido mediante análise dos seus modelos. Uma vez que estes servem de laboratório, no qual diferentes soluções para um problema relacionado à construção do sistema podem ser experimentadas.

Dessa forma, percebe-se que a modelagem de sistemas é de grande importância para o entendimento do problema a ser resolvido. Muitas técnicas e ferramentas já foram desenvolvidas, entretanto, com o uso de sistemas web complexos, a modelagem de software careceu de uma notação que abrangesse este nicho de mercado. Neste contexto, a próxima seção apresenta uma abordagem sobre linguagens de modelagem para aplicações web.

2.4.2 Linguagens de modelagem Web

Com o objetivo de melhorar o desenvolvimento de aplicações web na última década, de acordo com WIMMER et al. (2007), diversas abordagens de modelagem foram propostas no campo de engenharia de software. Ainda segundo os autores, tal situação se assemelha a dos anos 90 com a unificação da linguagem UML, que posteriormente se popularizou na engenharia de software.

Para engenheiros de software, enfrentar mudanças constantes na aplicação é um tormento, por mais que isto seja comum, surgem frequentemente novas possibilidades que desencadeiam novas exigências. ROSSI (2013) argumenta que a forma mais inteligente de lidar com os diferentes aspectos da evolução do software web é utilizar abordagens baseadas em modelos com uma linguagem formal, elevando assim o nível de abstração em que se pensa sobre aplicações web. Nesse cenário, pode-se dizer que o uso de modelos para criação das aplicações oferece vantagens, pelo fato de ser possível descrever funcionalidades complexas, antes mesmo de iniciar a implementação. A utilização da linguagem formal, facilita a compreensão do modelo do projeto para os envolvidos, assim, padronizando a interpretação, deixando-a mais clara e objetiva.

Comportamentos de navegação, interativos e composição de página, assim como funcionalidades características de aplicações web, requerem algumas especificações dife-

rentes, às quais as linguagens utilizadas até certo momento não atendiam muito bem. Há alguns anos, os engenheiros e analistas de software reconheceram a necessidade de avaliar qual linguagem utilizar, como UML ou outra linguagem de modelagem, para construir modelos que possuíssem estes comportamentos.

Nessa linha de pensamento, ROSSI (2013) conclui que o surgimento de um novo padrão, como a IFML mostra que o campo de desenvolvimento e modelagem de software amadureceu. Sendo assim, a próxima seção apresenta detalhes do funcionamento do novo padrão IFML.

2.5 Interaction Flow Modeling Language (IFML)

Na última década, com o acréscimo na utilização de sistemas Web e com o surgimento de novas plataformas, desenvolvedores, engenheiros, analistas e projetistas de software sentiram a necessidade de criar um novo padrão de modelagem que fosse mais adequado. Tal necessidade surgiu porque o padrão proposto pela WebML já não era suficiente para a modelagem de projetos nos quais os usuários interagem com o sistema. Outra opção seria utilizar o padrão UML, porém, este não é específico para modelagem web, tornando sua utilização limitada a certos casos.

Mediante tal situação, foi criada, em 2011, a IFML (Interaction Flow Modeling Language), uma linguagem de modelagem projetada para expressar o conteúdo, a interação do usuário e o comportamento de controle do *front-end* de aplicações de software (CASARIN, 2014). Inspirada nos padrões WebML e WebRatio, a nova linguagem foi adotada como padrão pela Object Management Group (OMG), em março de 2013. O projeto foi iniciado em 2011 e, contou com a participação dos seguintes pesquisadores: Stefano Butti, Marcos Brambilla, Piero Fraternali, Emanuele Molteni, Matteo Sassi e Matteo Silva.

O IFML surgiu de um grupo de pesquisa que teve a ideia de criar uma linguagem baseada em WebML, e para o desenvolvimento do projeto, foram convidados alguns pesquisadores, como o Stefano Butti, Marcos Brambilla, Piero Fraternali, Emanuele Molteni, Matteo Sassi e Matteo Silva, segundo o próprio OMG (Object Management Group). Conforme as especificações, o objetivo da IFML é fornecer aos integrantes de um projeto de software ferramentas para definição de modelos de fluxo de interação que descrevem as principais dimensões de uma aplicação *front-end*:

- Exibição do aplicativo, composta de contêineres e componentes de exibição;
- Objetos que incorporam o estado do aplicativo;
- Referências às ações de lógica de negócio que podem ser executadas;

- A ligação de componentes de visualização a objetos e eventos de dados;
- A lógica de controle que determina as ações a serem executadas após uma ocorrência de evento;
- A distribuição de controle, dados e lógica de negócios nas diferentes camadas da arquitetura.

A IFML traz vários benefícios para o processo de desenvolvimento de aplicações *front-ends*, pois está particularmente atenta para a modelagem de usabilidade e inteligibilidade (CASARIN, 2014). De acordo com (BRAMBILLA; FRATERNALI, 2014), os seguintes aspectos são cobertos pela IFML:

- Múltiplas visualizações para um mesmo aplicativo;
- Aplicativos móveis e multiplataforma;
- Visualização e entrada de dados, e produção de eventos;
- Componentes independentes, widgets e apresentações;
- Fluxo de interação;
- Contexto do usuário;
- Modularização do modelo;
- Validação de entrada do usuário

A notação IFML também é uma opção para a construção de diagramas, facilitando os desenvolvedores no entendimento do fluxo de navegação e interação com os usuários do sistema. Segundo o OMG (2014), um diagrama IFML é realizado em um ou mais View Containers de nível superior e cada contêiner de visão pode ser estruturado internamente em sub-contêineres.

Um contêiner de visão possui componentes de visualização, que apresentam a publicação de conteúdo e interface com elementos para entrada de dados. Esses componentes podem apresentar parâmetros de entrada e saída, utilizando como parâmetro de entrada o identificador dos objetos para exibir e, com isso, ter como saída os parâmetros de entrada ou valores do item selecionado. Tanto o contêiner, quanto o componente de visualização, podem estar associados a eventos os quais, podem causar um disparo de uma ação (OMG, 2014), demonstrando a linearidade e complexidade facilitada do sistema.

Além de tudo isso, a IFML utiliza mecanismos de extensibilidade, o que permite uma extensão dos conceitos do pacote núcleo, podendo ser realizada a introdução de novas construções para uma plataforma específica ou independente (OMG, 2014).

2.5.1 Composição da Linguagem

Segundo o OMG (2014), um diagrama IFML consiste em um ou mais View Containers (contêiner de visão ou visualização) de nível superior. Como exemplo, pode-se citar que um aplicativo desktop ou um aplicativo, pode ser modelado como tendo um contêiner de nível superior, que seria a janela principal. Já um aplicativo web pode ser modelado tendo vários top contêineres (contêineres de topo), um para cada modelo de página dinâmica.

O OMG (2014), diz que cada contêiner de visão pode ser estruturado internamente em uma hierarquia de sub-contêineres. Por exemplo, em uma área de trabalho, a janela principal pode conter vários quadros com guias, que por sua vez podem conter vários painéis aninhados. Na visualização de contêineres filhos, aninhados dentro de um contêiner de exibição pai, eles podem ser exibidos simultaneamente (um painel de objetos e um painel de propriedade, por exemplo), ou em exclusão mútua, (duas guias alternativas, por exemplo). Em casos de (XOR (exclusive OR)), ou seja, contêineres mutuamente excludentes, um poderia ser o contêiner padrão exibido quando o contêiner pai é acessado.

Conforme relata o OMG (2014), um contêiner de visão pode conter componentes de visualização que denotam a publicação de conteúdo e de interface com elementos para entrada de dados, como um formulário de entrada, por exemplo. Um componente de visualização pode ter parâmetros de entrada e saída. Um componente de visualização, para mostrar as propriedades de um objeto, pode ter como parâmetro de entrada o identificador do objeto para exibir. Um formulário de entrada de dados ou uma lista de itens podem ter como saída os parâmetros de entrada ou valores do item selecionado pelo usuário.

2.5.2 Notações IFML

Nesta seção, são apresentadas de forma detalhada as notações de componentes utilizadas pela IFML, as quais foram utilizadas na modelagem do *front-end* desse trabalho, e são apresentadas na seção 5.1.4, do Capítulo 5.

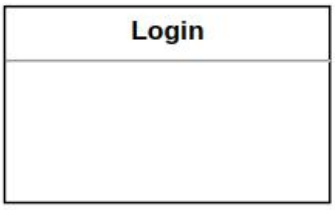
Para a construção dos diagramas IFML voltados para sistemas Web, podem ser utilizadas as ferramentas IFMLEdit, ou Draw.io. A ferramenta IFMLEdit é disponibilizada pela empresa WebRatio, responsável pela IFML. Enquanto a Draw.io é atualmente uma ferramenta incorporada pela Google para criação de diagramas de forma geral, como para IFML.

Conforme descrito em ORG (2018), as notações gráficas essenciais, de acordo com as padrões da IFML, são denominadas nos grupos:

- **Container**: representa um elemento da interface que contém elementos de exibição de conteúdo e interações entre estes e outros Containers. Exemplos de representação

desta notação são: Páginas Web, Janelas etc. Tal notação é ilustrada na Figura 14.

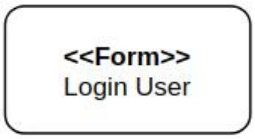
Figura 14 – View Container



Fonte: Adaptado de WebRatio e ORG (2018)

- **View Component**: representa um elemento da interface que apresenta alguma informação ou entrada de dados. Alguns exemplos de representação são: uma lista HTML, uma galeria de imagens em JavaScript ou um formulário de entrada de dados. Esta anotação é ilustrada na Figura 15.

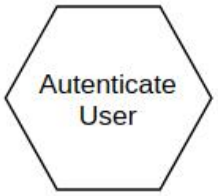
Figura 15 – View Component



Fonte: Adaptado de WebRatio e ORG (2018)

- **Action**: representa parte lógica de uma regra de negócio ou processamento iniciado por algum evento. Este pode ser executado tanto na parte do servidor quanto do cliente. Esta anotação é ilustrada na Figura 16. Alguns exemplos comuns são: atualização de dados na base de dados e envio de um e-mail.

Figura 16 – Action



Fonte: Adaptado de WebRatio e ORG (2018)

- **Event**: representa um acontecimento que altera o estado da aplicação ou parte dela. Alguns exemplos comuns são: cliques de botões e rolagem de barras (forçando a atualização de informações da tela). Esta anotação é ilustrada na Figura 17.

Figura 17 – Event

Fonte: Adaptado de WebRatio e ORG (2018)

- **Mavigation Flow**: representa uma navegação a partir de uma interação com dependência de entrada/saída de dados na qual a origem é de onde saiu o evento e o destino (para onde a seta aponta) é onde o dado irá interferir, como em uma tela ou componente. Esta anotação é ilustrada na Figura 18. Exemplos são os envios e recebimentos de parâmetros via *request* HTTP.

Figura 18 – Navigation Flow

Fonte: Adaptado de WebRatio e ORG (2018)

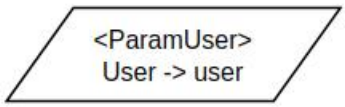
- **Data Flow**: representa a transição de dados entre componentes ou ações como consequência de uma interação anterior de um usuário. Esta anotação é ilustrada na Figura 19. Um exemplo de utilização é ao representar uma informação ser inserida ou deletada da base de dados.

Figura 19 – Data Flow

Fonte: Adaptado de WebRatio e ORG (2018)

- **Parameters Biding Group**: representa a construção dos parâmetros com os dados a serem encaminhados em um fluxo de iteração ou de dados. Esta anotação é ilustrada na Figura 20.

Figura 20 – Parameter Binding Group

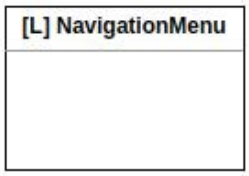


Fonte: Adaptado de WebRatio e ORG (2018)

Dentro do grupo de *Container* existem algumas especificações para quando necessário, como, Landmark View Container, Default View Container, e XOR View Container. É apresentado a seguir com mais detalhes:

- **Landmark View Container**: representa um contêiner de visualização acessível de qualquer outro elemento do usuário interface sem ter Fluxos de interação de entrada. Por exemplo, menus e rodapés. Esta anotação é ilustrada na Figura 21.

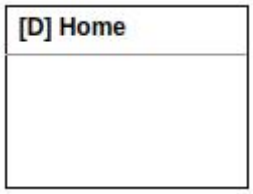
Figura 21 – Landmark View Container



Fonte: Adaptado de WebRatio e ORG (2018)

- **Default View Container**: contêiner de visualização será apresentado por padrão ao usuário, quando seu contêiner for acessado. Como exemplo, seria um tela princial ou *home-page*. Esta anotação é ilustrada na Figura 22.

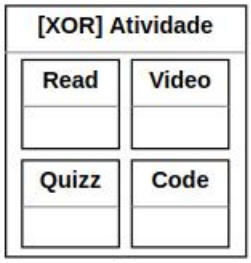
Figura 22 – Default View Container



Fonte: Adaptado de WebRatio e ORG (2018)

- **XOR View Container**: representa uma situação na qual um componente apresenta exclusivamente um dos componentes internos. Por exemplo, no caso do conteúdo da atividade pode ser apresentado exclusivamente um conteúdo do tipo de leitura, vídeo, questionário e código. Esta anotação é ilustrada na Figura 23.

Figura 23 – XOR View Container

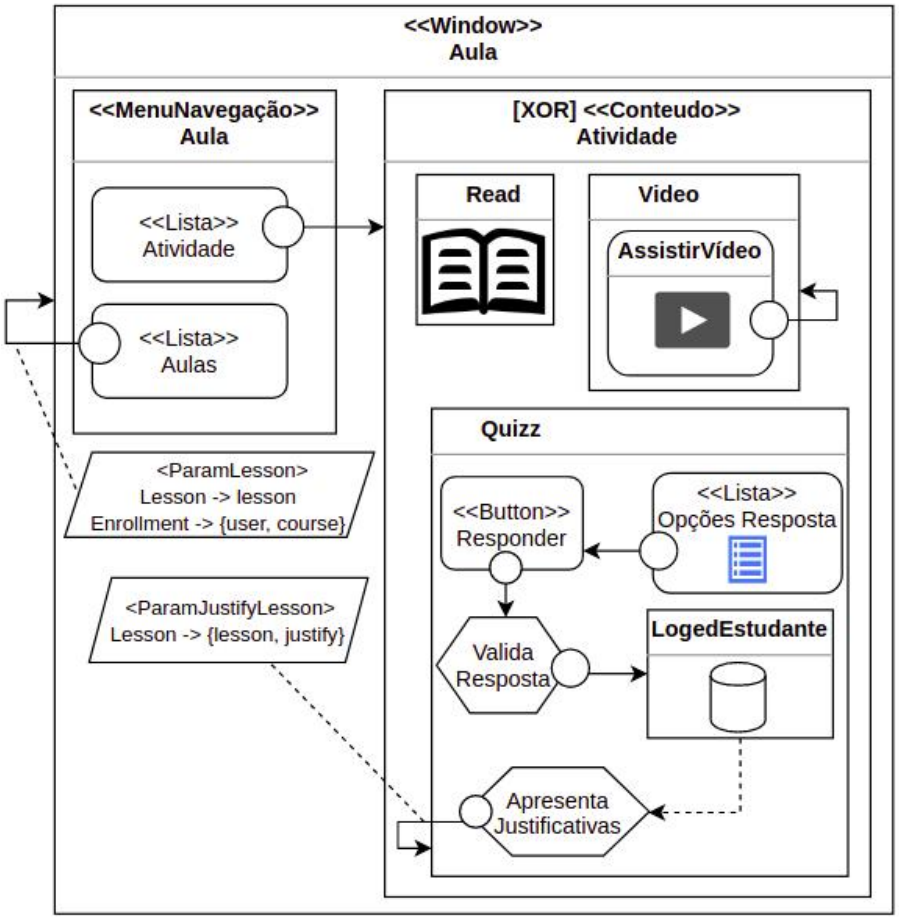


Fonte: Adaptado de WebRatio e ORG (2018)

Em seguida, na figura 24, é apresentado um exemplo de um IFML abordando todas as notações apresentadas, utilizando um cenário real da aplicação educa. O exemplo

mostra a tela de aula, a qual onde expõe o fluxo de apresentação do conteúdo de uma atividade. Este fluxo deixa claro a forma dinâmica que a tela de Atividade apresenta um conteúdo. Assim, de acordo com a atividade selecionada, pela lista de atividades no contêiner pai, é apresentado um contêiner filho de atividade específico, ou seja, caso o usuário tenha selecionado uma atividade do tipo leitura, será apresentada na área de atividade a tela de contudo de leitura.

Figura 24 – IFML tela de conteúdo de Atividade

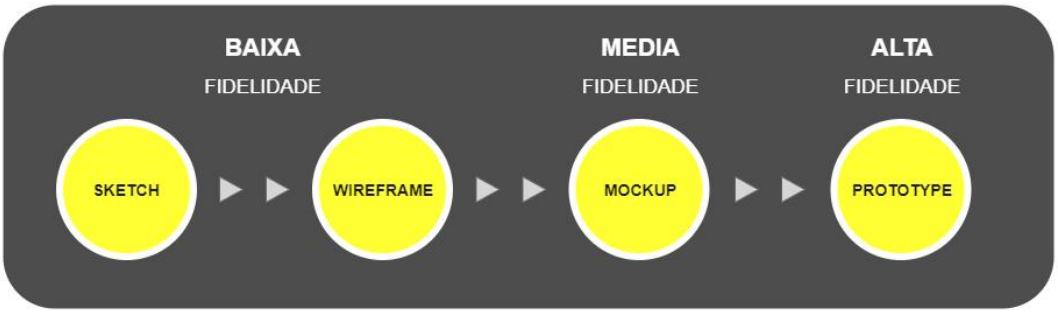


Fonte: Elaborado pelo autor.

2.6 Processo de Modelagem de Design de uma aplicação

O processo de modelagem de uma aplicação se define desde o momento da ideação até a implementação da solução. Este processo é de suma importância pelo fato de definir expectativas e de preceder a grande maioria das discussões e ajustes antes da implementação da aplicação (WARCHOLINSKI, 2019). Dentro deste processo, é executado uma jornada típica de 4 etapas: Sketch, Wireframe, Mockup e Prototype. Tais etapas são apresentadas na figura 25 de forma esquemática.

Figura 25 – Processo de modelagem de design design

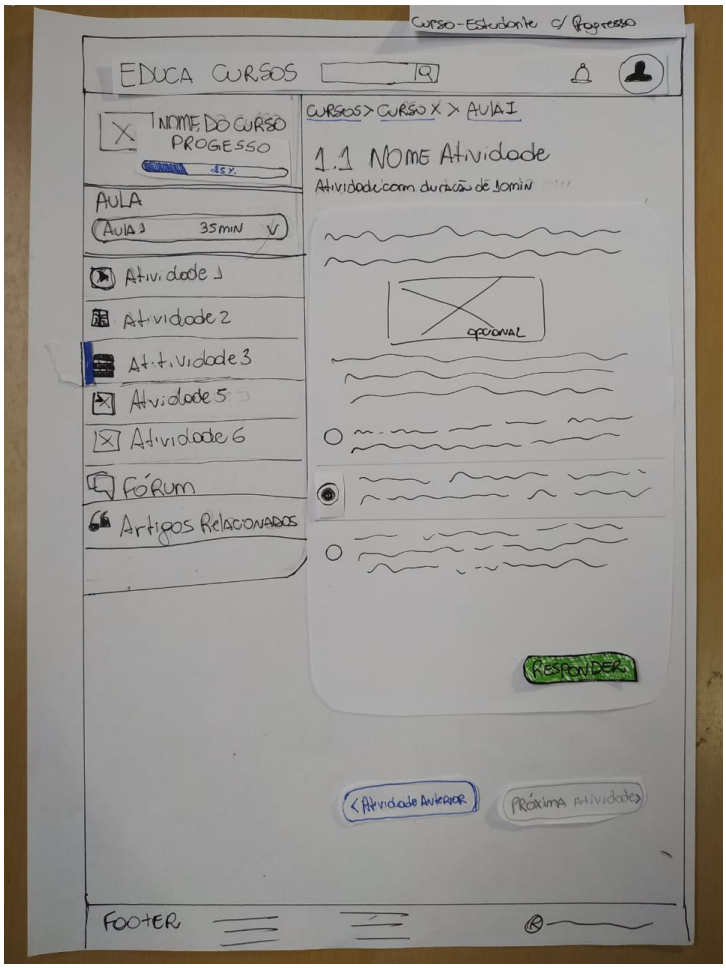


Fonte: Adaptado de (WARCHOLINSKI, 2019)

2.6.1 Sketch

Esta etapa é definida por uma representação em papel de baixa fidelidade da aplicação chamada *Sketch*. O caminho mais rápido para preparar sua ideia é realizar uma discussão breve. Esta etapa possui uma representação de baixa fidelidade pela abertura e facilidade para alterar algo sem muito custo de tempo e esforço. Na figura 26 é apresentado um exemplo de um *sketch* utilizado para representar uma atividade do tipo questionário.

Figura 26 – Sketch da página de uma atividade do tipo aula vídeo do Educa

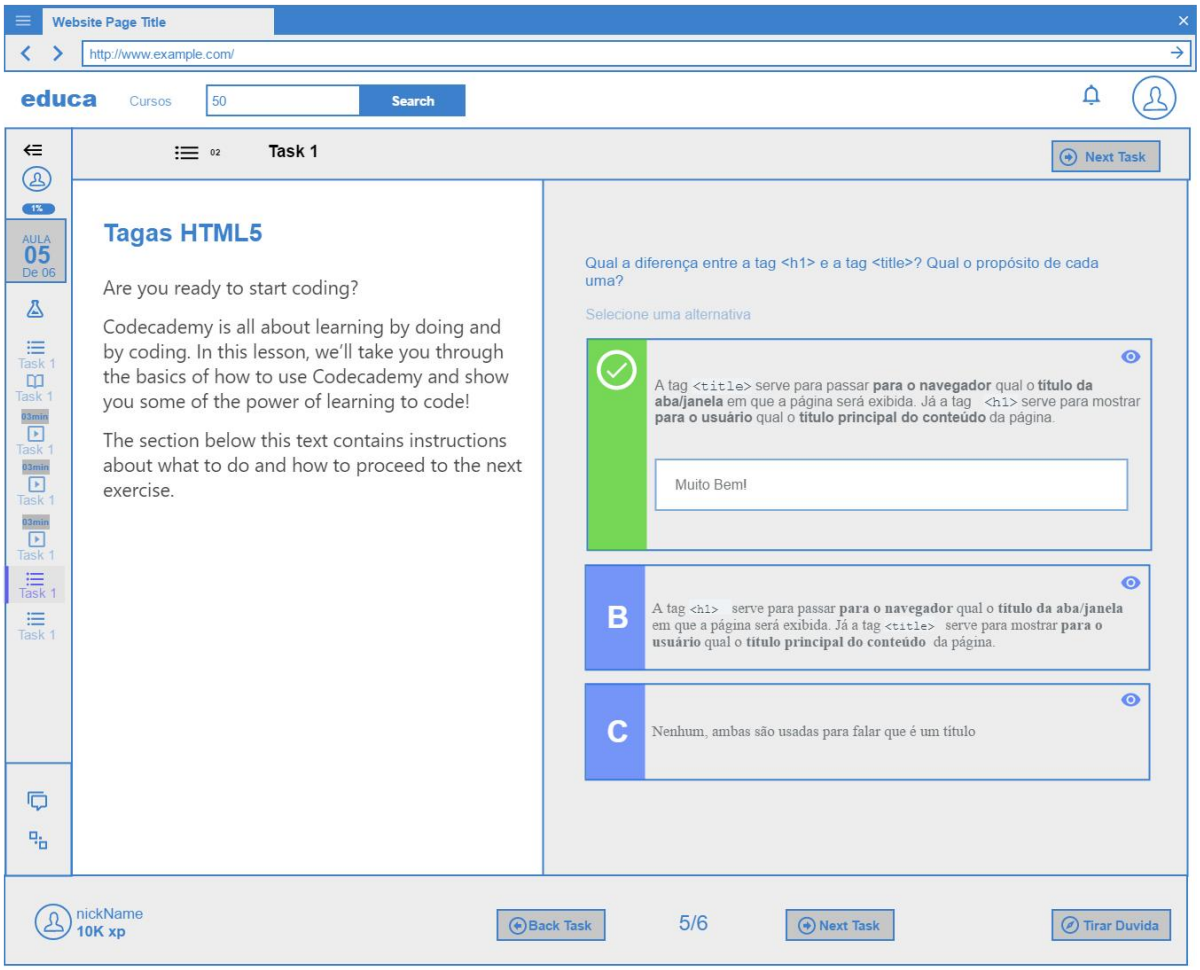


Fonte: Elaborado pelo autor.

2.6.2 Wireframe

Wireframes são representações visuais de baixa fidelidade que representam apenas elementos essenciais da IU (*wireframes* parecem ter sido projetados com fios e é daí que vem o nome). Estes atuam como esqueletos para o design do produto. São comumente construídos de forma simples e chamados dessa forma por parecer ter sido projetados com fios (BABICH, 2020). Os artefatos produzidos nesta etapa têm importante papel no planejamento de critérios de usabilidade e experiência do usuário. Nos critérios de usabilidade são considerados a facilidade de aprendizado do usuário, *feedback*, eficiência da aplicação, entre outros. Já nos critérios de experiência de usuários são considerados as opções de escolhas, sensações de controle, balanceamento entre dificuldades e habilidades. Para a construção do modelo *wireframe* pode se utilizar ferramentas como *Pencil Project*. Na figura 27 é apresentado um exemplo do *wireframe* utilizado para representar uma atividade do tipo questionário a partir do *sketch* descrito anteriormente.

Figura 27 – Wireframe da página de uma atividade do tipo aula questionário do Educa

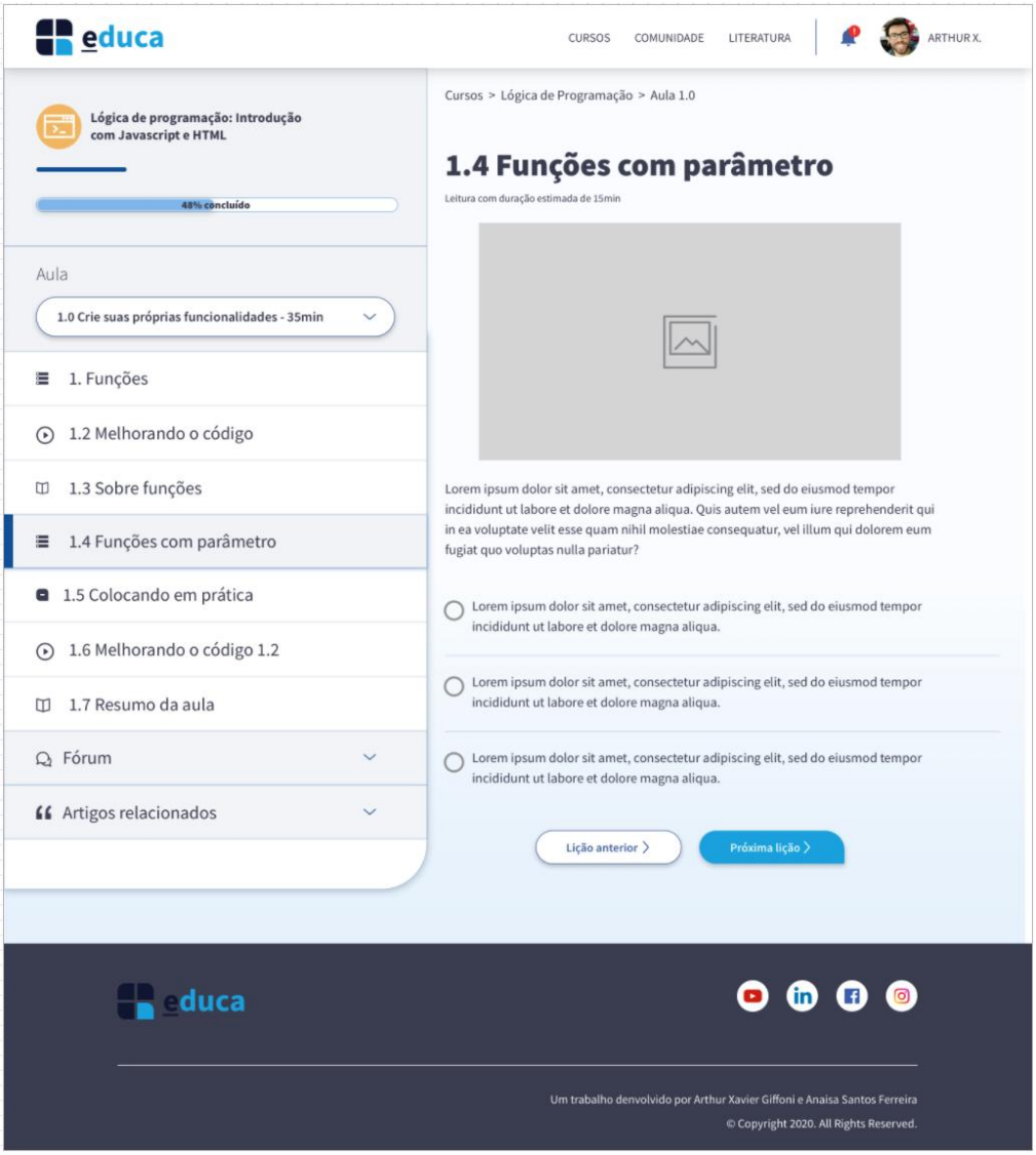


Fonte: Elaborado pelo autor.

2.6.3 Mockups

Nesta etapa, começa a preocupação com o nível de fidelidade da representação criada até o momento. Nesse sentido, os *Mockups* são representações de uma aplicação com uma média fidelidade de como esta será apresentada. O *mockup* dá abertura para uma análise melhor do visual do produto, como decisões, a relação de esquema de cores, estilo visual e tipografia do produto. Nesse momento, é interessante solicitar *feedback* dos usuários em potencial e fazer as alterações necessárias, poupando um custo maior no desenvolvimento. Para a construção do modelo *wireframe* pode se utilizar ferramentas como Sketch. Na figura 28, é apresentado um exemplo de *mockup* que foi utilizado para representar uma atividade do tipo questionário, a partir do *wireframe* descrito anteriormente.

Figura 28 – Mockup da página de uma atividade do tipo questionário do Educa

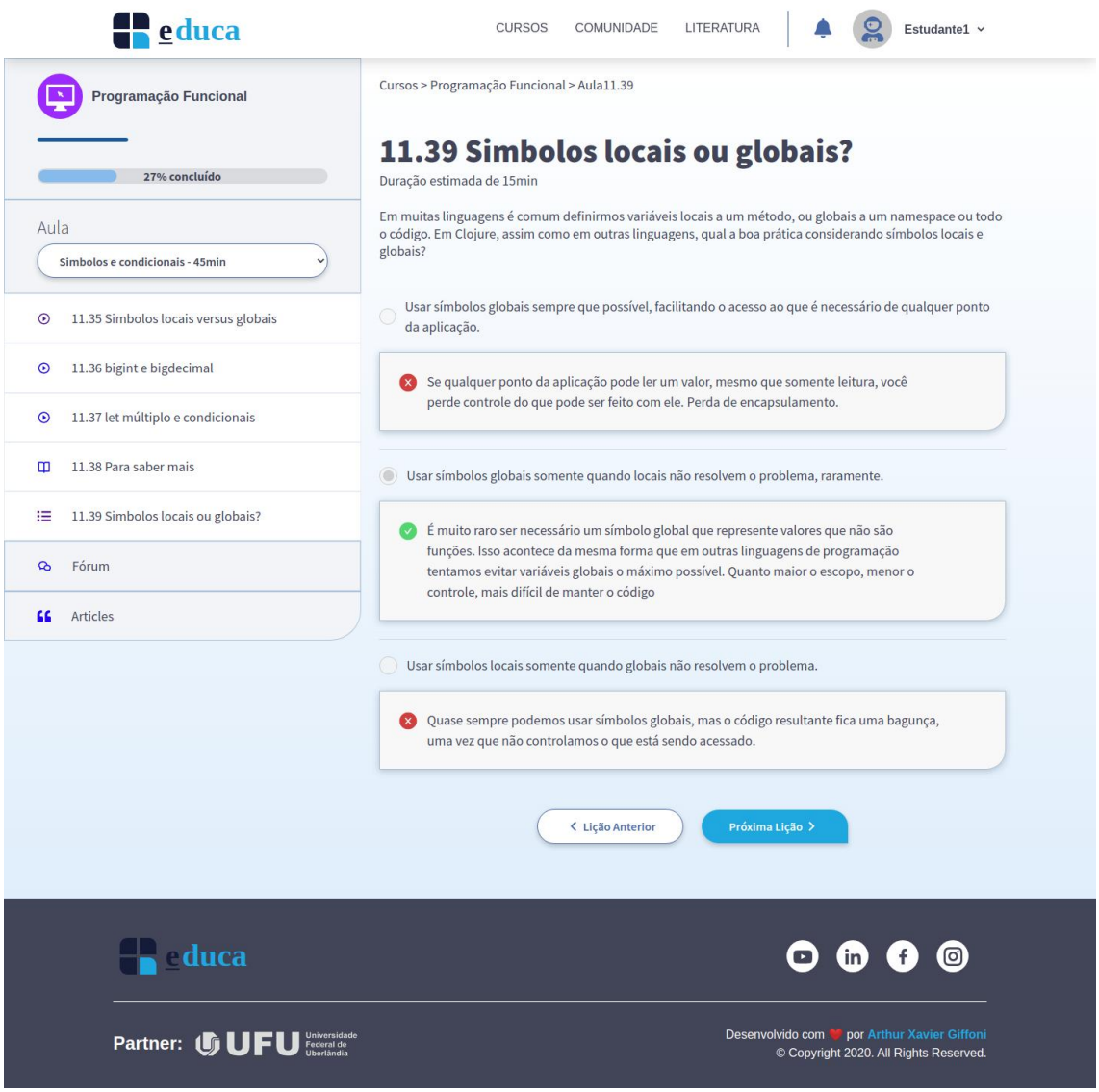


Fonte: Elaborado pelo autor.

2.6.4 Protótipo

Um protótipo é uma representação de alta fidelidade do produto final, destinada a simular a interação do usuário. Ao contrário dos anteriores, um protótipo é clicável e, portanto, permite ao usuário experimentar o conteúdo e as interações na interface. De fato, um protótipo é muito parecido com o próprio produto final. Porém, a diferença entre o produto final e o protótipo é, principalmente, que a interface e o *back-end* não costumam ser interligados no caso de um protótipo. Isso é feito para reduzir os custos de desenvolvimento, até que a interface do usuário seja aprovada. Depois que o protótipo é testado, a equipe pode continuar com a codificação. Para a construção do modelo *wireframe*, pode se utilizar ferramentas como *AdobeXD* ou um desenvolvimento breve. Na Figura 29, é apresentado um exemplo de protótipo que foi utilizado para representar uma atividade do tipo questionário, a partir do *mockup* descrito anteriormente.

Figura 29 – Protótipo da página de uma atividade do tipo questionário do Educa



Fonte: Elaborado pelo autor.

2.7 Arquitetura REST

Nessa seção, será abordado com mais detalhes sobre a arquitetura REST, a qual foi utilizada no *back-end* do sistema. Neste padrão se tem definido a estrutura de uma rota, como:

- Rota: é a descrição do que será requisitado ao backend. Esta é composta pela Base URL + Recurso.
- Base URL: parte da rota que não é alterada com a chamada de recursos diferentes;
- Recurso: é a descrição adicionada à Rota após a Base URL;

Para realizar a chamada de uma rota, no padrão REST, é necessário utilizar "verbos", os quais são os métodos HTTP que indicam quais são as intenções do requisitor ou cliente quando é feito uma requisição ao servidor. Segue abaixo a descrição dos métodos https:

- POST: criar dados no servidor.
- GET: leitura de dados no host.
- PATCH: atualização parcial de um registro.
- PUT: atualização total de um registro.
- DELETE: exclusão de um registro.

Ainda dentro desta estrutura, são definidas as formas de passar o conteúdo a ser manipulado, como também um conteúdo a ser usado como referência para encontrar um registro. Para realizar esse tipo de ação, existem alguns tipos de parâmetros, como:

- Query Params: nomeados e enviados na rota após um carácter “?”. Normalmente utilizados para filtros, paginações, etc;
- Route Params: não nomeados e enviados de acordo com a estrutura da rota esperada. Normalmente utilizados para referenciar algum registro.
- Request Body: corpo da requisição onde são enviados dados de uma estrutura ou parte de um registro. Normalmente utilizado para criação ou atualização de um registro

A Tabela 1 exemplifica uma rota definida no projeto para cadastro de curso. No apêndice C são apresentadas todas as rotas de requisições definidas no projeto.

Tabela 1 – Requisição para cadastro de curso

Requisição para Cadastro de Curso	
Rota	https://educaapi.com/admin/:admin_id/course

Body (json)	{
	"name": "Curso Teste 1",
	"identifier": "curso-teste-1",
	"description": "Descrição curso teste 1",
	"level_id": 1,
	"duration": 45,
	"instructor_ids": [2, 4],
	"active": true
	}

Fonte: Elaborado pelo autor.

2.8 Modelagem de Dados

Segundo REIS (2020a), a Modelagem de Dados é um modelo conceitual utilizado na Engenharia de Software, no qual modelos de dados são utilizados para descrever os objetos (entidades) envolvidos em um domínio de negócios, com suas características (atributos) e como elas se relacionam entre si (relacionamentos). Um modelo de dados normalmente assume a forma de um diagrama, apoiado por descrições textuais. Nestes diagramas são representados visualmente tais entidades, atributos e seus relacionamentos, como tipos de pessoas, lugares, objetos e conceitos que são importantes para o negócio.

Os dois tipos mais usados de modelos de dados são o Diagrama de Entidade-Relacionamento (DER) e o diagrama de classes, embora outras notações de modelagem também possam ser usadas. A notação a ser usada normalmente é determinada pela plataforma de tecnologia da organização. Os diagramas de entidade-relacionamento geralmente são preferidos quando o modelo será usado como base para um banco de dados relacional; enquanto os diagramas de classe são mais usados para apoiar o desenvolvimento orientado a objetos. Neste trabalho foi utilizado a notação de entidade-relacionamento para a modelagem. Esta decisão partiu da opção por utilizar um banco de dados relacional, tornando a modelagem a par do paradigma da linguagem utilizado para implementar a aplicação.

2.8.1 Modelo Entidades de Relacionamento (MER)

Modelagem de Entidades de Relacionamento, ou MER, é um modelo conceitual para o design de bancos de dados criado na década de 1970 por Peter Chen, atualmente membro do corpo docente da Carnegie-Mellon University. Enquanto professor assistente na Sloan School of Management, do MIT, Peter publicou um artigo inovador em 1976, intitulado O Modelo entidade relacionamento: uma visão unificada de dados (em tradução livre).

De acordo com LUCIDCHART (2020b), na década de 1960 e 70, Charles Bachman e A.P.G. Brown trabalhavam a abordagem de Peter Chen. Bachman desenvolveu um tipo de diagrama de estrutura de dados, batizado de diagrama de Bachman. Brown publicou trabalhos sobre modelagens de sistemas do mundo real. James Martin acrescentou refinamentos aos DERs, como a cardinalidade "Pé-de-Galinha". Os trabalhos de Chen, Bachman, Brown, Martin e outros também contribuíram para o desenvolvimento da Linguagem de modelagem unificada (UML, Unified Modeling Languagem), amplamente utilizada em design de software.

2.8.2 Diagrama Entidades de Relacionamento (DER)

O Diagrama de Entidade-Relacionamento (DER) é uma representação gráfica do MER, o qual tem como objetivo facilitar a comunicação entre os integrantes da equipe, pois oferece uma linguagem visual comum, que acaba simplificando, uma vez que, sem uma forma de visualizar as informações, o modelo pode ficar abstrato demais para auxiliar no desenvolvimento do sistema (REIS, 2020b). A seguir é descrito as representações e suas regras.

Entidades

Algo que pode ser definido e que pode ter dados armazenados sobre ele como uma pessoa, um objeto, conceito ou evento. Cabe pensar em entidades enquanto substantivos, por exemplo: um cliente, estudante ou curso. Normalmente eles são representados como um retângulo como na figura 30.

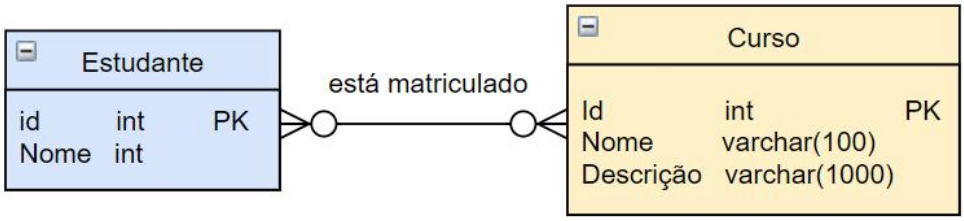
Relacionamentos

Como entidades atuam umas sobre as outras ou estão associadas uma com a outra, pense em relacionamentos como verbos, por exemplo: o estudante pode se inscrever em um curso. As duas entidades seriam o aluno e o curso, enquanto o relacionamento descrito é o ato de matricular-se, assim, conectando as duas entidades. Relacionamentos são tipicamente representados por losangos ou descrição em cima da ligação (este último normalmente utilizado para cardinalidades pé-de-galinha). Neste projeto, é utilizado a notação de relacionamentos descrito na ligação entre as entidades como na figura 30.

Atributos

A propriedade ou característica de uma entidade é muitas vezes representada por um oval ou círculo. Neste trabalho, vamos utilizar a descrição dos atributos dentro da entidade como o exemplo na figura 30.

Figura 30 – Modelo entidade de relacionamento entre as entidades estudante e curso



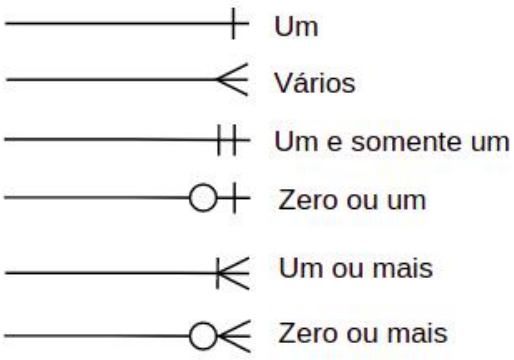
Fonte: Elaborado pelo autor.

Cardinalidade

A cardinalidade define a relação numérica entre duas entidades ou conjuntos de entidades. Os três principais relacionamentos cardinais são: um-para-um, um-para-muitos e muitos-para-muitos. Um exemplo de um-para-um seria um estudante associado a um endereço de correspondência. Um exemplo de um-para-muitos (ou muitos-para-um, dependendo do sentido da relação) é um estudante se inscrever para vários cursos, mas todos esses cursos têm uma única linha que leva de volta ao aluno. Um exemplo de muitos-para-muitos, por outro lado, consiste em estudantes, como um grupo sendo associados a vários membros do corpo docente, e membros do corpo docente, por sua vez, associados a vários alunos.

Existem duas notações comumente utilizadas para representar a cardinalidade, a representação numérica do Peter Chen, como (1, 1), ou a notação de James Martin, chamada popularmente de pé-de-galinha. Neste trabalho, é utilizado a representação pé-de-galinha como apresentado na figura 31.

Figura 31 – Notação da cardinalidade pé-de-galinha criado por de James Martin



Fonte: Adaptado de LUCIDCHART (2020b)

2.9 Dicionário de Dados

Um dicionário de dados, também conhecido como Repositório de Metadados, é um documento ou repositório utilizado para armazenar informações sobre o conteúdo, formato

e estrutura de um banco de dados, bem como os relacionamentos entre os seus elementos. Embora a ferramenta, infelizmente, não seja muito utilizada, é importante para limitar erros ao definir os domínios e a estrutura física do banco de dados. Sendo assim, neste trabalho, com objetivo de simplificar a descrição das tabelas, o dicionário de dados foi separado em duas partes, sendo elas: **entidades** e **atributos**.

Entidades

Parte do dicionário de dados que descreve nome da tabela, relações e descrição, como representado na tabela 2:

Tabela 2 – Exemplo de entidade de um dicionário de dados da tabela estudante

Tabela	**Nome do Relacionamento**	**Relacionamento**	**Descrição**
Estudante	está matriculado	Curso	Tabela para cadastro de estudante
Curso	possui matricula	Estudante	Tabela para cadastro de curso

Fonte: Elaborado pelo autor.

Atributos

Parte do dicionário de dados que descreve os atributos de uma tabela ou entidade. Neste são descritos o nome da coluna, tipo do dado a ser salvo (inteiro, caracteres, data), tamanho, restrições, valor padrão e descrição do campo. Um exemplo é apresentado na tabela 3:

Tabela 3 – Exemplo de atributos de um dicionário de dados da tabela curso

Entidade User			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
Id	Inteiro	PK	Numero identificador do usuário
Nome	Caracter	NOT NULL, tamanho máximo: 30	Nome do usuário

Fonte: Elaborado pelo autor.

2.10 UML

UML, do inglês *Unified Modeling Language*, é uma linguagem visual utilizada para modelar sistemas orientados a objetos. Foi aprovada como padrão em 1997 pela OMG, Object Management Group, e, desde então, tem sido utilizada amplamente pela comunidade de desenvolvedores de sistemas (BEZERRA, 2015).

Durante o desenvolvimento de um sistema de software, que utiliza a UML como linguagem de suporte à modelagem, são criados diversos tipos de documentos, sendo eles textuais ou gráficos. Esses documentos são denominados artefatos de software, os quais são produzidos mediante utilização dos diagramas da UML.

De acordo com organização oficial da UML, na UML 2.0 são definidos treze tipos de diagramas, divididos em três categorias: **estrutura**, **comportamento** e **integração**.

- **Diagramas de Estrutura**: incluem o diagrama de classes, o diagrama de objetos, o diagrama de componentes, o diagrama de estrutura composta, o diagrama de pacotes e o diagrama de implantação.
- **Diagramas de Comportamento**: incluem o Diagrama de Caso de Uso (usado por algumas metodologias durante a coleta de requisitos); diagrama de atividades e diagrama de máquina de estados.
- **Diagramas de Integração**: incluem o diagrama de sequência, o diagrama de comunicação, o diagrama de tempo e o diagrama de visão geral da interação.

Neste trabalho, foram utilizados o diagrama de classes, diagrama de sequência, diagrama de caso de uso e diagrama de atividade. Além destes, foram utilizadas a modelagem de dados e dicionário de dados. Os diagramas escolhidos são mais detalhados a seguir.

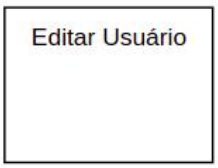
2.10.1 Diagrama de Casos de Uso

Segundo BEZERRA (2015), o diagrama de casos de uso é voltado à apresentação das funcionalidades e características do sistema, assim como para a interação dos usuários com estas funcionalidades e com entidades externas envolvidas num determinado processo.

Os diagramas de caso de uso são compostos por quatro partes principais: **cenário**, **ator**, **caso de uso** e **relacionamento**. Estes são descritos com mais detalhes abaixo.

- **Cenário**: sequência de eventos que acontecem quando um usuário interage com o sistema, como apresentado na figura 32:

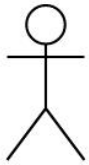
Figura 32 – Cenário



Fonte: Adaptado de BEZERRA (2015)

- **Ator**: usuário do sistema, ou melhor, um tipo de usuário. A representação é como apresentada na figura 33:

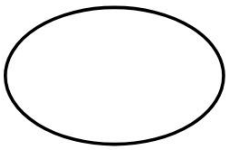
Figura 33 – Ator



Fonte: Adaptado de BEZERRA (2015)

- **Caso de Uso**: tarefa ou uma funcionalidade realizada pelo ator (usuário). A representação é como apresentado na figura 34:

Figura 34 – Caso de Uso



Fonte: Adaptado de BEZERRA (2015)

- **Relacionamento**: descreve a ligação entre ator e caso de uso, entre atores e entre casos de uso, como na na figura 35:

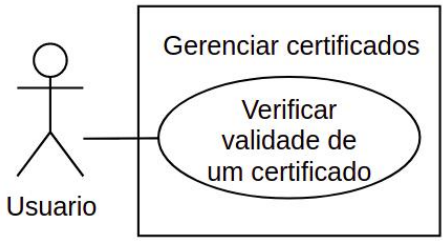
Figura 35 – Relacionamento

Fonte: Adaptado de BEZERRA (2015)

Nesse sentido, um relacionamento pode ser especificado de três maneiras diferentes: relacionamento de comunicação ou associação; relacionamento de inclusão; relacionamento de extensão e relacionamento de herança.

- **Relacionamento de Comunicação ou Associação**: representa a integração, por meio de mensagens, entre o ator e um caso de uso. É representado por uma linha sólida, tal qual apresentada na figura 36.

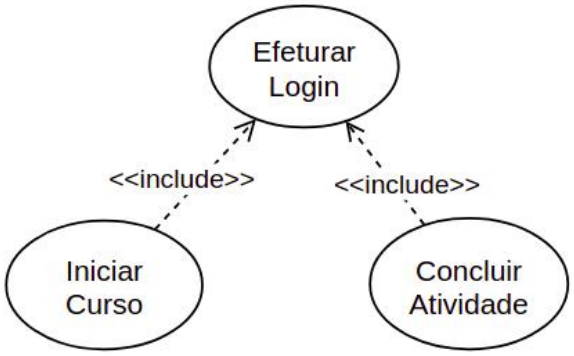
Figura 36 – Relacionamento de Comunicação



Fonte: Adaptado de BEZERRA (2015)

- **Relacionamento de Inclusão**: utilizado quando um comportamento se repete em mais de um caso de uso, por exemplo, no educa para um estudante iniciar um curso ou concluir uma atividade, precisa estar logado. A anotação para esse caso é como apresentada na figura 37.

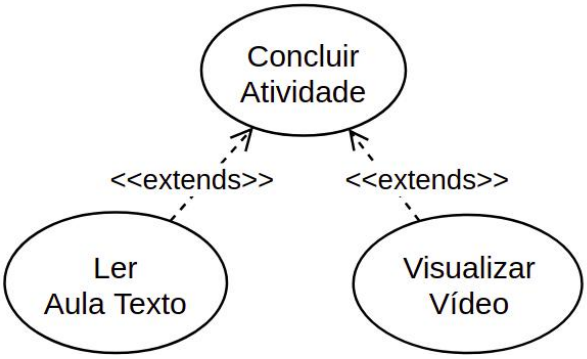
Figura 37 – Relacionamento de Inclusão



Fonte: Adaptado de BEZERRA (2015)

- **Relacionamento de Extensão**: utilizado quando é necessário modelar um relacionamento alternativo, anotação para esse caso é como apresentado na figura 38. Por exemplo, ao "concluir atividade"pode ser por meio de "Ler Atividade Texto"ou "Visualizar Vídeo".

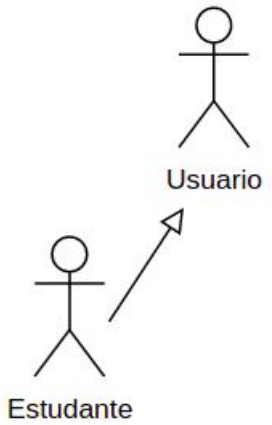
Figura 38 – Relacionamento de Extensão



Fonte: Adaptado de BEZERRA (2015)

- **Relacionamento de Herança**: relacionamento entre atores, utilizado quando é necessário representar uma especialização/generalização, como apresentado na figura 39. Por exemplo na figura a seguir, estudante é especialização de usuário (ou usuário é generalização de estudante), representado por uma linha com um triângulo.

Figura 39 – Relacionamento de Herança



Fonte: Adaptado de BEZERRA (2015)

2.10.2 Diagrama de Sequência

De acordo com LUCIDCHART (2020a), o diagrama de sequência demonstra as interações do objeto em diferentes camadas do processo de uma operação, destacando ainda a ordem em que tais ações acontecem num intervalo de tempo. A sequência em que as diversas operações são executadas ocorre na vertical, de cima para baixo, e os níveis de profundidade são apresentados da esquerda para direita, onde mais à esquerda apresenta a camada mais externa (usuário) e a camada mais à direita apresenta a camada mais interna (banco de dados).

Os diagramas de sequência têm as principais notações:

- **Atores**: representam as entidades externas que interagem com o sistema e que solicitam serviços, como representado na figura 40. Normalmente, o ator primário é o responsável por enviar a mensagem inicial à interação entre os objetos.

Figura 40 – Notação de ator para diagramas de sequência

Fonte: Adaptado de LUCIDCHART (2020a)

- **Objetos**: simbolizam as instâncias das classes apresentadas no processo, não representadas normalmente por um retângulo, como representado na figura 41.

Figura 41 – Notação de objetos para diagramas de sequência

Fonte: Adaptado de LUCIDCHART (2020a)

- **Linha de Vida (lifeline)**: linhas verticais que representam o tempo de vida de um objeto, como representado na figura 42.

Figura 42 – Notação de linha de vida para diagramas de sequência

Fonte: Adaptado de LUCIDCHART (2020a)

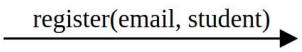
- **Barra de Atividade (Activation bar)**: barras verticais que representam exatamente quando uma instância do objeto passou a existir. Essas barras verticais preenchem a *Linha de Vida*, como representado na figura 43.

Figura 43 – Notação de barra de atividade para diagramas de sequência

Fonte: Adaptado de LUCIDCHART (2020a)

- **Mensagem**: são linhas horizontais sólidas que representam mensagens trocadas entre objetos como representado na figura 44. Estas linhas são acompanhadas de um rótulo que contém o nome da mensagem e, opcionalmente, os seus parâmetros. Pode existir mensagens enviadas para um mesmo objeto, representando uma iteração.

Figura 44 – Notação de barra de mensagens para diagramas de sequência



Fonte: Adaptado de LUCIDCHART (2020a)

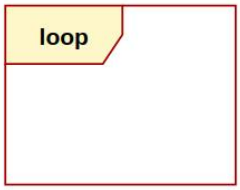
- **Mensagem de Retorno**: linhas horizontais tracejadas que representam mensagens de retornos entre objetos como representado na figura 45. Estas linhas são acompanhadas de um rótulo que contém o nome da mensagem e, opcionalmente, os parâmetros da mesma.;

Figura 45 – Notação de mensagens de retorno para diagramas de sequência

Fonte: Adaptado de LUCIDCHART (2020a)

- **Loop**: usados para modelar cenários em que aquela transição de mensagem será executada repetidas vezes até atingir uma condição, como representado na figura 46.

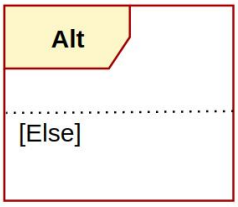
Figura 46 – Notação de loop para diagramas de sequência



Fonte: Adaptado de LUCIDCHART (2020a)

- **Alternativa**: representa um conjunto de possíveis ações no qual uma pode ser executada de acordo com a condição contemplada. Representada por uma forma de retângulo rotulada com uma linha tracejada em seu interior, como na figura 47.

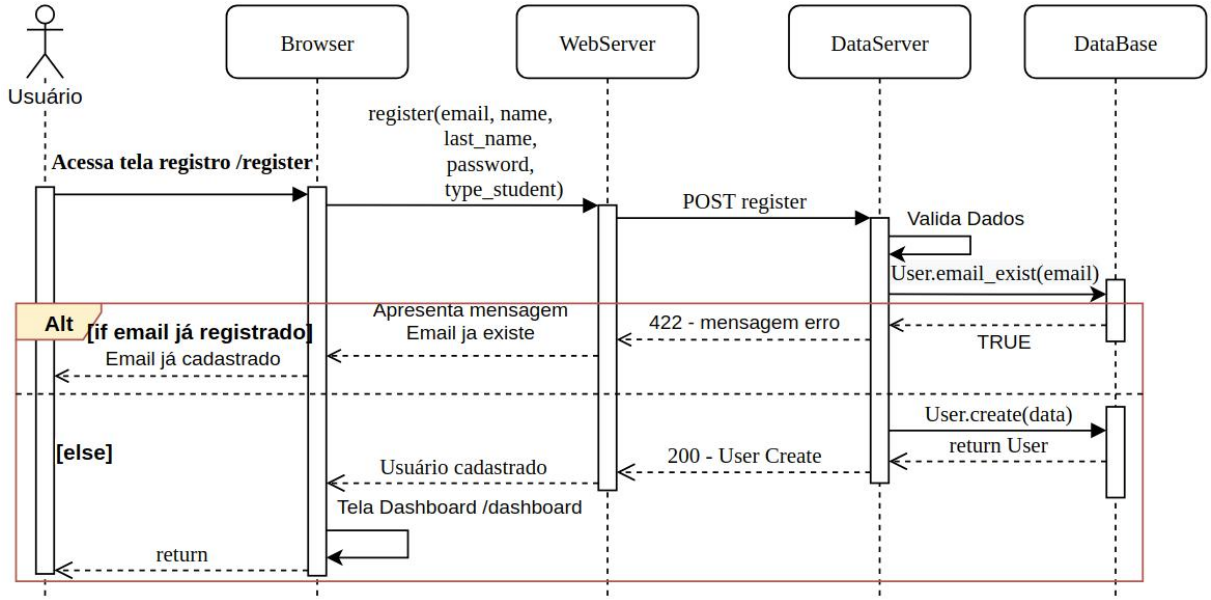
Figura 47 – Notação de alternativa para diagramas de sequência



Fonte: Adaptado de LUCIDCHART (2020a)

O Diagrama de Sequência, exibido na Figura 48, representa um exemplo que determinada a sequência de eventos para o cadastro de um usuário no sistema Educa. Identificando quais mensagens devem ser disparadas entre os elementos envolvidos e em que ordem, o diagrama de sequência, tem o objetivo principal de determinar a ordem em que os eventos ocorrem, as mensagens que são enviadas, os métodos que são chamados e como os objetos interagem dentro de um determinado processo.

Figura 48 – Diagrama de sequência dos eventos para cadastro de um usuário



Fonte: Elaborado pelo autor.

2.10.3 Diagrama de Atividade

O diagrama de atividade utiliza a UML (Linguagem de Modelagem Unificado) para representar a ordem dos passos de um processo e o fluxo de controle. De acordo com UML, diagramas de atividade, junto com diagramas de caso de uso, são considerados diagramas de comportamento porque descrevem o que é necessário acontecer no sistema sendo modelado. O diagrama tem como objetivo apresentar o fluxo de um único processo com clareza e concisão, mostrando como a atividade depende uma da outra. Diagramas de atividade ajudam a unir a área de negócios com a de desenvolvimento em uma organização, ou grupo, para entender o mesmo processo e comportamento.

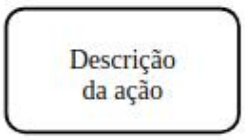
A criação deste diagrama no trabalho foi considerada necessária por este auxiliar em alguns processos relevantes:

- Ilustrar um processo de negócio ou fluxo de trabalho entre usuários e o sistema;
- Descrever as etapas realizadas em um caso de uso UML;
- Simplificar e melhorar qualquer processo ao esclarecer casos de uso complicados;
- Demonstrar a lógica de um algoritmo.

Para a construção de um diagrama de atividade são utilizando alguns componentes básicos, que são:

- **Atividades**: uma etapa que compõe um processo em que o usuário ou software realiza uma determinada tarefa. São representadas por retângulos como na figura 49.

Figura 49 – Símbolo para atividades



Fonte: Adaptado de UML (2021)

- **Nó de decisão**: um ramo condicional no fluxo representado por um diamante. Inclui uma única entrada e duas ou mais saídas, representados por um losango como na figura 50.

Figura 50 – Símbolo para nós de decisão

Fonte: Adaptado de UML (2021)

- **Fluxos de controle**: outro nome dado aos conectores que mostram o fluxo entre as etapas no diagrama. Uma seta de entrada inicia um passo de uma atividade. Uma vez concluído o passo, o fluxo continua com a seta de saída como na figura 51.

Figura 51 – Símbolo para fluxos de Controle

Fonte: Adaptado de UML (2021)

- **Nó inicial**: simboliza o início da atividade. É representado por um círculo preto como na figura 52.

Figura 52 – Símbolo para nó inicial

Fonte: Adaptado de UML (2021)

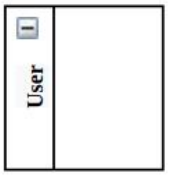
- **Nó final**: representa a etapa final da atividade. É representado por um círculo preto delineado como na figura 53.

Figura 53 – Símbolo para nó final

Fonte: Adaptado de UML (2021)

- **Partição de atividade**: representada por uma notação de raia - com duas linhas, geralmente paralelas, tanto horizontais quanto verticais, e um nome rotulando a partição em uma caixa em uma das extremidades como na figura 54. Quaisquer nós de atividade, por exemplo, ações e arestas colocadas entre essas linhas, são considerados como contidos na partição.

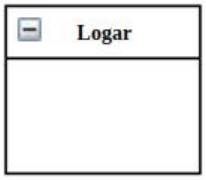
Figura 54 – Símbolo para partições de atividades



Fonte: Adaptado de UML (2021)

- **Processo**: representada por um retângulo externo que contém todo o fluxo no seu interior, e um nome rotulado da atividade na extremidade superior, como representado na figura 55.

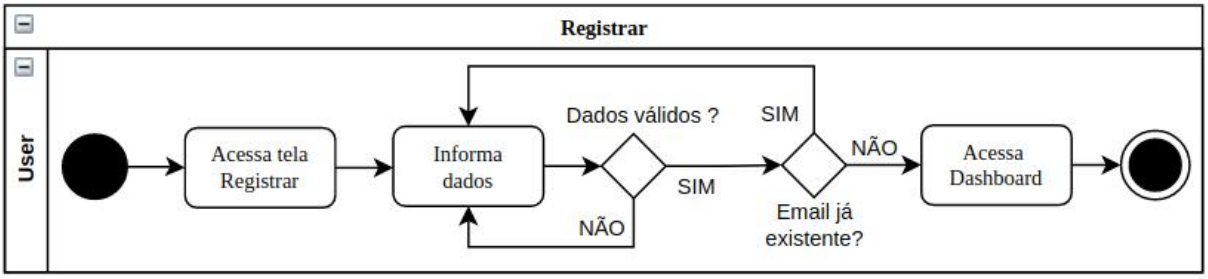
Figura 55 – Símbolo para processos



Fonte: Adaptado de UML (2021)

O diagrama de atividade, exibido na Figura 56, representa um exemplo que determinada o fluxo para um usuário se registrar na plataforma Educa. Neste cenário, o usuário ainda não registrado, acessa a tela R̈egistrar; informa seus dados. Após informar os dados, a plataforma valida tais dados, caso não sejam válidos ou o email já esteja em uso, a mesma informa o usuário ainda pela tela de R̈egistro; assim o usuário repetindo o processo. Caso os dados estejam válidos, o usuário já logado, é redirecionado para a tela de *dashboard*.

Figura 56 – Diagrama de atividade do fluxo para um usuário se cadastrar na plataforma Educa



Fonte: Elaborado pelo autor.

2.11 MVC

MVC (Model-View-Controller) é uma arquitetura amplamente utilizada em aplicativos web modernos, essa relaciona com sucesso e eficiência a interface do usuário aos modelos de dados subjacentes. Discutida pela primeira vez em 1979, por *Trygve Reenskaug*, e introduzida pela primeira vez em 1987, na linguagem de programação *SmalTalk*, foi aceita pela primeira vez como conceito geral em 1988, de acordo com GURU99 (2021)

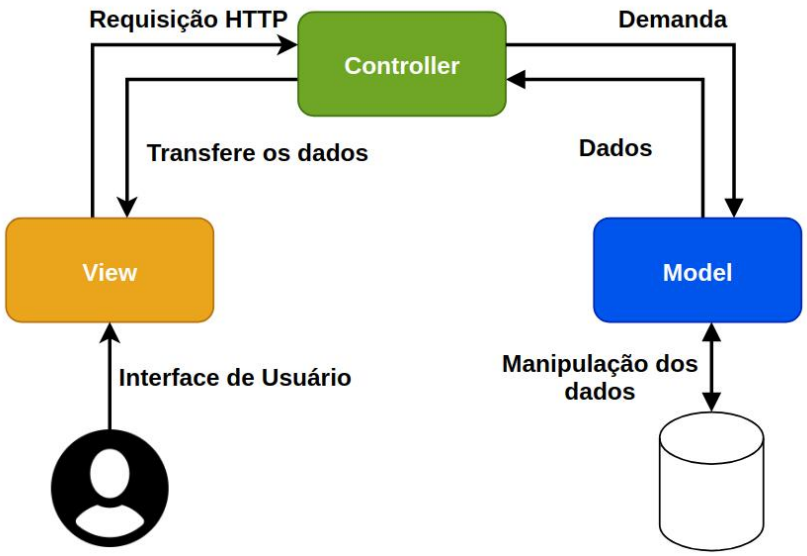
A MVC apresenta vantagens que interferem positivamente no desenvolvimento de uma aplicação, sendo elas:

- Testabilidade fácil e sem atrito;
- Estrutura altamente testável, extensível e plugável;
- Oferece controle total sobre o HTML de forma desacoplada;
- Separação clara de lógica: Modelo, Visão, Controlador e consequentemente a separação das tarefas da aplicação como, lógica de negócios, lógica Ul e lógica de entrada;
- Fácil gerenciamento e mapeamento de URLs tornando-as facilmente compreensíveis e pesquisáveis

De acordo com GOOGLE (2021), existem algumas variações do padrão de design MVC, como MVP (Model View Presenter) e MVVP (Model View ViewModel). Mesmo com o chamado padrão de projeto MVC em si, há alguma variação entre o padrão MVC tradicional e a interpretação moderna em várias linguagens de programação. Por exemplo, alguns frameworks baseados em MVC farão com que a visualização observe as mudanças nos modelos, enquanto outros permitirão que o controlador domine a atualização da visualização.

Este trabalho se concentra na utilização da arquitetura MVC, onde o controlador domina a atualização da visualização. Na Figura 57 é apresentado a arquitetura que sera utilizada no trabalho, e assim suas camadas: **Model**, **View**, **Controller**.

Figura 57 – Arquitetura MVC



Fonte: GOOGLE (2021)

MVC tem sido amplamente utilizado por muitos desenvolvedores de software. Tradicionalmente usado para interfaces gráficas com o usuário, esse padrão se tornou popular para projetar aplicações de software web. Linguagens de programação populares como JavaScript, Python, Ruby, PHP, Java e C# utilizam diretamente na estruturas MVC.

Em um exemplo geral, num cenário de um restaurante o usuário seria o cliente na mesa que quer pedir um prato. O cardápio (View) com a informação dos pratos é utilizado pelo cliente para escolher qual deseja pedir. Após escolher, o cliente requisita para o garçom (Controller) pedir na cozinha para o chefe de cozinha (Model) preparar o prato com os ingredientes (dados).

Model

O modelo ou *model* é a camada da aplicação que representa os dados e possui a lógica relacionada. Os modelos constituem a camada de nível mais baixo do padrão responsável por manter os dados, fazendo contato direto com o banco de dados. Esta, por sua vez, realiza as ações interpretadas pelo controlador, manipulando os dados diretamente no banco de dados.

As principais ações realizadas nos dados são: **cadastro**, **leitura**, **edição** e **remoção**. Além dessas ações, o modelo pode possuir validações de regras de negócios, como, cadastrar um novo usuário, se este tem um nome de no máximo 60 caracteres.

View

O view ou visão, por sua vez, é a camada mais próxima do usuário, a qual é responsável por apresentar visualmente os dados, e também, por onde os usuários interagem com a aplicação. Elas são criadas a partir dos dados fornecidos pelo modelo repassado pelo controlador. Esta apresenta qualquer visualização do cliente, que incluirá todos os componentes da UI e seu design, como, caixas de texto, menus suspensos, etc.

É responsáveis pela entrada de dados e por apresentar visualmente os dados do banco de dados solicitados pelo Modelo. É possível ter várias visões do mesmo dado, como um gráfico de barras para gerenciamento e uma visão tabular para contadores.

A Visão também provoca interações com o usuário, que interage com o Controle (*Controller*). O exemplo básico disso é um botão gerado por uma Visão com a função de cadastrar um Curso, ou seja, quando o usuário clica no botão é acionado a ação de cadastrar curso para o Controle, que é repassado para o modelo registrar as informações do novo curso no banco de dados.

Controller

O controlador ou *controller*, é a parte do aplicativo que controla a interação do usuário. O controlador interpreta as entradas do mouse e do teclado do usuário, informando o modelo e a visualização a serem alterados conforme apropriado. Também é responsável por receber a requisição do usuário, a interpretar consultando os modelos, e enviar como resposta o conjunto de dados consultados esperado ou uma resposta planejada, caso não encontrado.

3 Revisão Sistemática

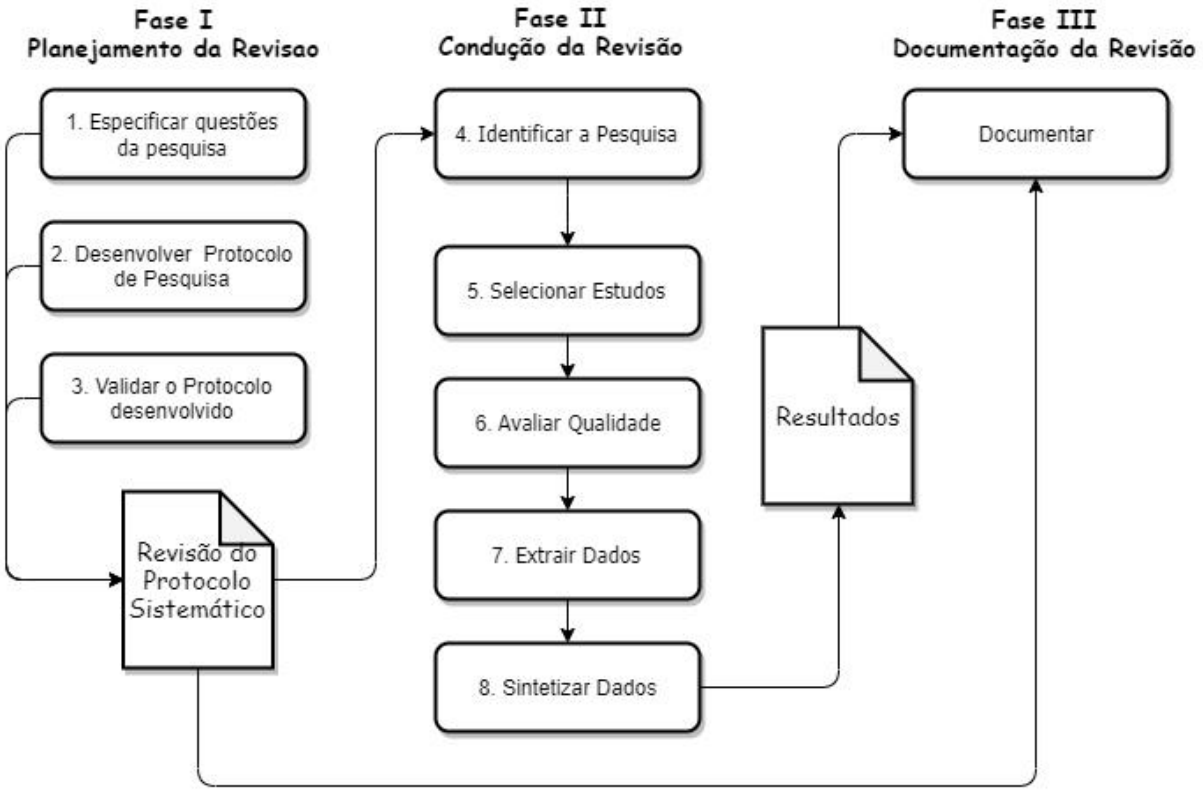
A Revisão Sistemática da Literatura (RSL) é o método científico que visa identificar, analisar e interpretar de forma profunda todos os estudos disponíveis e relevantes para uma questão de pesquisa específica, área temática ou fenômeno de interesse (KITCHENHAM; BRERETON, 2015). Uma RSL constitui-se de procedimentos que são utilizados na realização da revisão, os quais devem ser tão objetivos, analíticos e repetíveis quanto possível, de forma que esse processo, no ideal, seja totalmente replicável, assim, se a revisão fosse repetida por outros, seriam selecionados os mesmos estudos de entrada, portanto, chegando às mesmas conclusões. Consequentemente, o uso deste método amplifica a confiabilidade da pesquisa por utilizar critérios de avaliação críticos, rigorosos, claros e de fácil reprodução, tornando os resultados e a conclusão da pesquisa claros e precisos.

Uma RSL provê aos pesquisadores uma visão geral do conhecimento científico de uma determinada área ou tema, assim, proporciona um planejamento para avanço da pesquisa na área sem realizar redundâncias e/ou repetições de erros. Entretanto, mesmo que uma revisão seja realizada, devem existir protocolos para sustentar sua característica exploratória, pois, sem um protocolo pré-estabelecido e de possível replicabilidade, a pesquisa pode ser dirigida por interesse pessoal dos pesquisadores, dessa maneira, podendo ser caracterizada com uma pesquisa de pouco valor científico, por possivelmente possuir resultados não confiáveis ou replicáveis.

Sendo assim, este capítulo dará enfoque na realização de uma revisão sistemática das publicações em estado de arte relacionadas à modelagem de um modelo de comunicação de STI. Para alcançar tal objetivo, foi desenvolvido um protocolo de revisão utilizando as diretrizes propostas por KITCHENHAM; BRERETON (2015). Tais diretrizes são compostas por três fases principais: planejamento da revisão, condução da revisão e documentação da revisão.

A primeira fase tem como objetivo toda a parte de desenvolvimento do protocolo e sua validação. Com a primeira fase realizada e definido o protocolo, então, é iniciado a segunda fase, a qual tem como objetivo a condução da pesquisa utilizando fielmente os protocolos pré-estabelecidos. Ainda nesta fase são selecionados os estudos e documentado os resultados finais de acordo com os critérios de seleção. Posteriormente, com os resultados da fase dois, é iniciada a fase três, realizando a documentação de todo o processo e resultados estabelecidos. Tal fluxo está representado na Figura 58:

Figura 58 – Representação do fluxo da revisão sistemática



Fonte: KITCHENHAM; BRERETON (2015)

3.1 Planejamento da Revisão

Esta seção expõe a fase I da RSL. Lembrando que esta fase tem como objetivo descrever as diretrizes que conduzem o desenvolvimento do protocolo e sua validação, descrevendo o planejamento da pesquisa com todos os detalhes, a fim de garantir sua integridade e replicabilidade.

3.1.1 Especificação de Questões da Pesquisa

A especificação das questões de pesquisa é um componente crítico do protocolo porque são as questões de pesquisa que conduzem os estágios posteriores da revisão. Seguindo o objetivo deste trabalho, apresentado na seção 1.3, a revisão da literatura tem como objetivo responder as seguintes perguntas:

- Existe um modelo detalhado do módulo de comunicação (ou interface) de um sistema tutor inteligente?
- Se existe, qual modelo formal de modelagem web é utilizado para representar sua arquitetura?

3.1.2 Protocolo de pesquisa

O protocolo desempenha um papel fundamental no planejamento de uma revisão, pois fornece uma estrutura para fazer e documentar as decisões necessárias durante o estudo. O objetivo é minimizar o viés do pesquisador, definindo as etapas que serão seguidas e os critérios com nos quais as decisões serão feitas durante a realização de uma revisão.

Ainda segundo KITCHENHAM; BRERETON (2015), o protocolo tem grande importância em todo o processo por:

- Ajudar a reduzir a probabilidade de viés do pesquisador, limitando a influência das expectativas do pesquisador, por exemplo, na seleção de estudos (primários) individuais ou na síntese de resultados;
- Poder ser avaliado por outros pesquisadores que possam fornecer feedback sobre o projeto de uma revisão antes da sua conduta;
- Formar a base das seções de introdução e método de um relatório de uma revisão.

3.1.3 Base de Busca

Uma base de busca representa um site ou repositório de artigos, monografias, dissertações e teses onde é realizado a pesquisa avançada.Para realizar uma RSL é necessária uma base de busca que apresenta o número total de artigos e estudos encontrados a partir de uma pesquisa. As bases abaixo foram utilizadas por indicar o número total de artigos e estudos encontrados:

- BDTD - https://bdtd.ibict.br
- Springer - www.springerlink.com
- ACM - https://dlnext.acm.org
- IEEE Xplore - www.ieeexplore.ieee.org
- ScienceDirect - www.sciencedirect.com
- Google Scholar - https://scholar.google.com.br
- Microsoft Academic - https://academic.microsoft.com
- Repositório UFU - https://repositorio.ufu.br

3.1.4 Palavras chave

Para realizar uma pesquisa direcionada ao assunto ou tema escolhido, é importante definir as palavras chave e suas variantes. Nesta seção, são apresentadas as palavras chave escolhidas, sendo elas:

- Intelligent Tutoring System;
- Interface;
- Communication;
- IFML;
- Interaction Flow Model

Foram utilizadas as palavras "Communication" e "Interface" pelo fato de alguns autores utilizarem as duas nomenclaturas, "Communication Model", "Interface Model" e "User Interface Model", para se referir ao modelo de STI. Por não existirem resultados com a combinação das palavras chave "IFML" e "Intelligent Tutoring System", em nenhuma das bases utilizadas, foi realizado uma separação de dois grupos: Grupo1 e Grupo2, na qual o primeiro terá como objetivo buscas por "Intelligent Tutoring System" com enfoque no modelo de comunicação/interface, e o segundo na área de Modelagem de Software com enfoque na utilização do padrão "IFML". Foram escolhidos os mais impactantes de cada grupo e, produzido uma modelagem formal do modelo de comunicação utilizando IFML para nosso caso de estudo com Sistemas Tutores Inteligentes.

3.1.5 String de Busca

A *string* de busca representa a maneira com que as palavras chave serão condicionadas e organizadas com intuito de se realizar a pesquisa em uma base de busca. A *string* apresentada a seguir é de forma genérica, ou seja, não considerando ainda as regras de cada base de busca. De acordo com cada grupo, as strings são:

- (intelligent tutoring system) E ((interface) OU (Communication))
- (ifml) OU (interaction flow model)

3.1.6 Refinamento da String de Busca

Cada base de dados possui uma formatação ou especificação própria para string de busca, dessa forma, é apresentado abaixo como foi utilizado as *strings* em cada base para conseguir a pesquisa adequada.

3.1.6.1 IEEE Xplore

- (((“All Metadata”:Communication) OR “All Metadata”:Interface) AND “All Metadata”:Intelligent Tutoring System)
- ((“All Metadata”:IFML) OR “All Metadata”:Interaction Flow Model)

3.1.6.2 Springer, ScienceDirect, BDTD

- “Intelligent Tutoring System” AND (“Interface” OR “Communication”)
- (“IFML” OR “Interaction Flow Model”)

3.1.6.3 ACM

- acmdlTitle:(+Intelligent +Tutoring +System Interface Communication) OR recordAbstract:(+Intelligent +Tutoring +System Interface Communication)
- acmdlTitle:(“ifml” or “interaction flow model”) OR recordAbstract:(“ifml” or “interaction flow model”)

3.1.6.4 Microsoft Academic, Google Scholar

- “intelligent tutoring system” interface communication
- “ifml” “interaction flow model”

3.1.6.5 Repositório UFU

Nesta base de busca não possui uma opção de intervalo de data, sendo assim, foi necessário realizar uma busca para cada ano do intervalo 2014 a 2019.

- intelligent tutoring system
- ifml Interaction Flow Model

3.1.7 Filtros de Busca

O filtro serve para delimitar e direcionar a busca de acordo com as restrições especificadas, com a finalidade da pesquisa não ser genérica.

- Intervalo de publicação do trabalho de 2014 a 2019;
- Buscar as palavras chave somente no título e no resumo;
- Idiomas: inglês, português e espanhol.

3.1.8 Dados da Busca

Os dados de busca de pesquisa serão apresentados em uma estrutura organizada para que futuras pesquisas possam repetir o experimento e avaliar os resultados deste trabalho de maneira fiel. A estrutura se constitui da seguinte forma:

- Nome da base de busca;
- Data de execução;
- String de busca utilizada de acordo com o grupo;
- Quantos artigos encontrados;
- Observações (caso necessário).

3.1.9 Ordenamento das Publicações

BARBOSA (2016) argumenta que um fenômeno ainda comum no sistema de produção científica é o efeito Matthew, no qual os pesquisadores mais célebres tendem a receber mais reconhecimento e recursos financeiros por suas atividades acadêmicas, enquanto os pesquisadores pouco conhecidos tendem a receber pouco ou nenhum reconhecimento por suas atividades.

Ciente disso, as publicações foram ranqueadas pela relevância da base. Esta estratégia foi utilizada a fim de driblar o efeito Matthew, ou seja, a indicação de um artigo somente por ser mais citado, virando um ciclo vicioso. As bases referem a relevância de um trabalho publicado por número de citações, pela recência e pelo fator de impacto da revista, jornal ou congresso em que o trabalho foi publicado.

3.1.10 Questões de Avaliação de Qualidade

As questões de avaliação têm como objetivo determinar a qualidade dos trabalhos encontrados, a fim de extrair informações relevantes de cada um para que seja realizada uma classificação. As perguntas devem ser respondidas realizando uma leitura superficial no trabalho.

Q1. O trabalho aborda uma modelagem de um Sistema Tutor Inteligente?

Q2. O autor utiliza a revisão sistemática da literatura?

Q3. O trabalho aborda sobre modelo de interface ou de comunicação?

Q4. O trabalho aborda o assunto experiência de usuário?

Q5. É utilizado um método formal para modelagem do software e/ou fluxo de interface?

Q6. O trabalho utilizou o padrão IFML para modelagem?

Q7. O processo de feedback ao usuário é claramente apresentado?

Q8. É apresentado um protótipo do sistema?

3.1.11 Critérios de Inclusão e Exclusão de Trabalhos

Nesta seção, são apresentados os critérios de inclusão e exclusão dos trabalhos para continuação de extração de dados. Como critério de inclusão, foram considerados apenas artigos científicos publicados em periódicos, congressos, teses e dissertações. Sendo assim, foram excluídos artigos repetidos e trabalhos que tratam somente de revisão. Para a seleção dos demais trabalhos, as questões de avaliação levantadas na seção 3.1.10 tiveram um fator peso determinante.

Tabela 4 – Pesos de questões de Avaliação

Pesos	**Questões**
3	Q3 Q6
2	Q1 Q4 Q5
1	Q2 Q7 Q8

Fonte: Elaborado pelo autor.

Cada questão foi respondida com uma das seguintes opções: 10 (satisfatório), 5 (parcialmente satisfatório) e 0 (insatisfatório). Após atribuído os valores de cada questão para cada um dos trabalhos, foi realizado um somatório:

$$Trabalho(i) = \sum_{n=1}^{8} Pn.Qn \text{ onde, P=peso e Q=questão}$$

Dessa forma, cada trabalho poderia ter um valor máximo de 150 pontos, caso fosse satisfatório em todos os pontos. Os trabalhos iguais e acima de 80 foram considerados com alta prioridade e incluídos na lista para leitura detalhada. Os trabalhos que obtiverem entre 60 e 79, foram considerados como de média prioridade e parcialmente incluídos na lista dos trabalhos que poderiam ser lidos ou não, dependendo da satisfação da pesquisa após a leitura dos de alta prioridade. Os trabalhos com notas abaixo de 60, por sua vez, foram considerados totalmente excluídos, não sendo levados a leitura profunda em qualquer hipótese. Na Tabela 5, estão representados os critérios de inclusão e exclusão de um artefato neste trabalho, onde n é sua nota.

3.1.12 Validação do Protocolo desenvolvido

Com a finalidade de validar a qualidade do protocolo desenvolvido para a seleção dos trabalhos, foram realizadas consultas experimentais nas bases de buscas levantadas. Durante as buscas experimentais, foram analisados os seguintes fatores:

Tabela 5 – Limites de inclusão e exclusão dos trabalhos

Situação	**Nota**
Incluído	n >= 80
Parcialmente Incluído	60 <= n < 80
Excluído	n < 60

Fonte: Elaborado pelo autor.

- **Trabalhos encontrados**: se buscando esporadicamente pelo tema em estudo, os trabalhos encontrados são os mesmos ou fazem parte do conjunto de trabalhos encontrados seguindo o protocolo;
- **Palavras chave escolhidas**: se faz sentido as palavras escolhidas e se realmente não existia outro termo mais significativo para os temas, segundo os autores;
- **Combinação de busca dos temas (modelagem de software e STIs)**: nesse item foi analisado se realmente não é uma escolha de filtro equivocado o motivo da falta de trabalhos encontrados que abordavam a modelagem IFML em casos de estudo para STIs, principalmente para os modelos de comunicação;
- **Intervalo de tempo**: analisou a possibilidade de equívoco na escolha do intervalo de 2014 a 2019, sabendo que o assunto STIs são relativamente antigos, sendo tratados em meados dos anos 2000, porém observou que a utilização da modelagem IFML é muito recente, por ser tão nova.

Analisando os demais pontos levantados no protocolo, realmente os termos de palavras chave são satisfatórios e, a combinação dos assuntos IFML e STIs não possui até o momento nenhum trabalho realizado. Sobre o intervalo de tempo, o protocolo apresenta trabalhos satisfatórios atuais dos dois temas, lembrando que o assunto IFML ainda é, de certa forma, recente. Como exemplo, a execução trouxe os seguintes trabalhos - previamente identificados como relevantes para a RSL - referentes a cada assunto:

- ITS
  - Improving User Interface and User Experience of MathSpring Intelligent Tutoring System for Students (NGO; TRAN, 2017)
  - Improving User Interface and User Experience of MathSpring Intelligent Tutoring System for Teachers (MENON, 2018)
  - An Intelligent Tutoring System for Learning Android Applications UI Development (REKHAWI; NASER, 2018)
  - The think aloud method for qualitative evaluation of an intelligent tutoring system interface (MORAIS; SCHAAB; JAQUES, 2017)

- IFML
  - Analyzing Interaction Flow Modeling Language in Web Development Lifecycle (WAKIL; JAWAWI, 2017)
  - Sistema de Informação para controle de materiais e doações aos bombeiros voluntários (OLIVEIRA, 2018)

Ainda assim, para o assunto STIs, utilizando o mesmo protocolo com intervalo de datas diferentes da proposta, foram encontrados trabalhos muito relevantes, como o de GRANIC; STANKOV; GLAVINIC (2000), assim, certificando sua qualidade independente de uma data especifica. Dessa forma, conclui-se que o protocolo é válido.

3.2 Condução da Revisão

Nesta seção são apresentados os resultados da pesquisa de acordo com o protocolo estabelecido na seção 3.1

3.2.1 Identificação de Pesquisas

O foco desta seção foi a identificação de estudos primários relevantes. Esse processo constitui o primeiro passo da fase de condução da revisão sistemática.

A pesquisa foi realizada em cada base descrita na subseção 3.1.3, de acordo com as diretrizes estabelecidas, a fim de encontrar os trabalhos para leitura e, se satisfatório, realizar a extração de dados. Foi estipulado a seleção dos 50 primeiros artigos em cada busca. No total, foram encontrados 321516 artigos, destes, 470 foram considerados para uma leitura panorâmica. Os dados de cada pesquisa serão representados a seguir:

3.2.1.1 IEEE Xplore

- Data da busca: 28/09/2019
- Grupo 1
  - Quantidade de artigos encontrados: 131
  - Observação: foram considerados apenas os 50 primeiros artigos.
- Grupo 2
  - Quantidade de artigos encontrados: 1567
  - Observação: foram considerados apenas os 50 primeiros artigos.

3.2.1.2 Springer

- Data da busca: 29/09/2019
- Grupo 1
  - Quantidade de artigos encontrados: 3887
  - Observação: foram considerados apenas os 50 primeiros artigos.
- Grupo 2
  - Quantidade de artigos encontrados: 264765
  - Observação: foram considerados apenas os 50 primeiros artigos.

3.2.1.3 ScienceDirect

- Data da busca: 28/09/2019
- Grupo 1
  - Quantidade de artigos encontrados: 18
- Grupo 2
  - Quantidade de artigos encontrados: 20

3.2.1.4 BDTD

- Data da busca: 28/09/2019
- Grupo 1
  - Quantidade de artigos encontrados: 3
- Grupo 2
  - Quantidade de artigos encontrados: 0

3.2.1.5 ACM

- Data da busca: 29/08/2019
- Grupo 1
  - Quantidade de artigos encontrados: 145
  - Observação: foram considerados apenas os 50 primeiros artigos.

- Grupo 2
  - Quantidade de artigos encontrados: 46861
  - Observação: foram considerados apenas os 50 primeiros artigos.

3.2.1.6 Microsoft Academic

- Data da busca: 28/09/2019
- Grupo 1
  - Quantidade de artigos encontrados: 10
- Grupo 2
  - Quantidade de artigos encontrados: 47

3.2.1.7 Google Scholar

- Data da busca: 29/09/2019
- Grupo 1
  - Quantidade de artigos encontrados: 4040
  - Observação: foram considerados apenas os 50 primeiros artigos.
- Grupo 2
  - Quantidade de artigos encontrados: 20

3.2.1.8 Repositorio UFU

- Data da busca: 29/09/2019
- Grupo 1
  - Quantidade de artigos encontrados: 2
- Grupo 2
  - Quantidade de artigos encontrados: 0

3.2.2 Seleção de Estudos

Os documentos selecionados por meio do processo de busca tiveram sua relevância verificada em relação às questões de pesquisa abordadas por uma revisão, assim como propõe KITCHENHAM; BRERETON (2015). Sendo assim, o foco desta seção foi esse processo de seleção que forma a segunda etapa da fase de condução do processo de revisão sistemática.

Ainda de acordo com KITCHENHAM; BRERETON (2015), a seleção de estudos geralmente é realizada em várias etapas. Inicialmente, uma vez que o conjunto de documentos candidatos foi identificado, aqueles que são claramente irrelevantes podem ser excluídos com base em seu título ou/e resumo. Depois desta triagem inicial, os documentos devem ser analisados com mais detalhes.

Nesse sentido, esta seção documenta a primeira seleção dos estudos com uma leitura panorâmica de acordo com as informações do título, resumo e conclusão de cada artigo. A partir desta leitura, nos 470 trabalhos, detectou-se relevantes 17 trabalhos do Grupo 1 (STIs e ao modelo de comunicação/interface) e 11 trabalhos do Grupo 2 (modelagem de software utilizando IFML). Nas Tabelas 6 e 7 estão listados os trabalhos selecionados e as respectivas bases de buscas.

Tabela 6 – Trabalhos selecionados após leitura panorâmica referentes ao Grupo 1

GRUPO 1	
BDTD	GALAFASSI (2019)
Google Scholar	AL-BASTAMI; NASER (2017) LAUREANO-CRUCES et al. (2014) REKHAWI; NASER (2018)
IEEE Xplore	VERDÚ et al. (2014) BANERES; SAÍZ (2016) MORAIS; SCHAAB; JAQUES (2017) EDUARDO; HUGO (2014) AREVALILLO-HERRÁEZ et al. (2017)
Microsoft Academic	NGO; TRAN (2017) MENON (2018) VANLEHN (2016)
ScienceDirect	MAHMOUD; EL-HAMAYED (2016)
Springer	HOOSHYAR et al. (2016) WERAGAMA; REYE (2014) KHACHATRYAN et al. (2014)
UFU	JUNIOR; FREITAS et al. (2018)

Fonte: Elaborado pelo autor.

Tabela 7 – Trabalhos selecionados após leitura panorâmica referentes ao Grupo 2

GRUPO 2	
ACM	QUEIROZ et al. (2018) BRAMBILLA et al. (2016)
Google Scholar	WAKIL; JAWAWI (2017) BARCELONA et al. (2017) OLIVEIRA (2018)
IEEE Xplore	LAAZ; MBARKI (2016) ROUBI; ERRAMDANI; MBARKI (2016)
Microsoft Academic	BLÁZQUEZ (2018)
ScienceDirect	BERNASCHINA; COMAI; FRATERNALI (2018)
Springer	FALZONE; BERNASCHINA (2018) ACERBIS et al. (2015)

Fonte: Elaborado pelo autor.

3.2.3 Avaliação de Qualidade dos Estudos Selecionados

Nesta seção, foi documentado uma segunda seleção realizada por uma leitura intermediária, que teve como objetivo responder às questões de avaliação de qualidade dos estudos e, assim, apresentar as respostas de cada questão definida na subseção 3.1 para cada trabalho. As quais estão apresentadas na Tabela 8.

É importante observar que em nenhum trabalho foi realizada a revisão sistemática da literatura, e que nenhum trabalho relacionado à STI apresentou uma modelagem formal de interface. Estes são aspectos que dificultam a análise, replicabilidade e o avanço científico na área modelo de interface de um STIs.

3.2.3.1 Publicações levantadas

GALAFASSI (2019) apresenta o desenvolvimento de um agente Modelo de Aluno e sua aplicação em um sistema de tutoria inteligente voltada ao ensino de lógica. Para modelar este agente, utilizou-se um mecanismo de inferência capaz de calcular estatisticamente a probabilidade de o aluno conhecer um dado conceito das regras de dedução natural, na lógica proposicional. É apresentado uma interface simples sem utilização de uma modelagem formal.

Já AL-BASTAMI; NASER (2017) demonstram um Sistema Tutor Inteligente para ajudar os alunos a aprenderem a linguagem de programação C#. O STI foi desenvolvido usando a ferramenta de autoria chamada ITSB (Intelligent Tutoring System Builder), que teve como objetivo específico ajudar alunos a aprenderem a programar com eficiência e tornar o procedimento de aprendizado agradável. Uma base de conhecimento usando o estilo da ITSB foi usada para representar o trabalho do aluno e fornecer feedback e suporte personalizados.

Tabela 8 – Notas atribuídas as publicações escolhidas para a leitura intermediária a partir das respostas das questões de avaliação de qualidade.

Publicação	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Total
Galafassi(2019)	10	0	5	5	0	0	5	5	55
Al-Bastami; Naser(2017)	5	0	5	5	0	0	5	5	45
Laureano-Cruces et al.(2014)	5	0	10	10	0	0	5	5	70
Rekhawi; Naser(2018)	5	0	10	0	0	0	5	5	50
Verdú et al.(2014)	5	0	10	5	0	0	5	5	60
Baneres; Saíz(2016)	5	0	5	5	0	0	5	10	50
Morais; Schaab; Jaques (2017)	5	0	10	10	0	0	10	10	80
Eduardo; Hugo(2014)	0	0	5	5	0	0	5	5	35
Arevalillo-Herráez et al.(2017)	0	0	5	5	0	0	5	5	35
Ngo; Tran(2017)	10	0	10	10	0	0	10	10	90
Menon(2018)	5	0	10	5	0	0	5	10	80
Vanlehn(2016)	0	0	10	0	0	0	10	10	50
Mahmoud; El-Hamayed(2016)	5	0	0	0	0	0	0	0	10
Hooshyar et al.(2016)	10	0	10	0	0	0	5	0	55
Weragama; Reye(2014)	10	0	5	0	0	0	0	0	35
Khachatryan et al.(2014)	10	0	5	5	0	0	5	0	50
Junior; Freitas et al.(2018)	10	0	0	0	0	0	0	0	20
Queiroz et al.(2018)	0	0	0	10	10	10	5	0	75
Brambilla et al.(2016)	0	0	0	10	10	10	5	0	75
Wakil; Jawawi(2017)	0	0	0	10	5	10	5	10	80
Barcelona et al.(2017)	0	0	0	5	10	10	5	0	65
Oliveira (2018)	0	0	0	5	10	10	10	10	80
Laaz; Mbarki(2016)	0	0	0	5	0	0	5	0	15
Roubi; Erramdani; Mbarki(2016)	0	0	0	0	10	10	5	0	55
Blázquez(2018)	0	0	0	5	10	10	5	0	65
Bernaschina; Comai; Fraternali(2018)	0	0	0	0	10	10	5	0	55
Falzone; Bernaschina (2018)	0	0	0	5	10	10	5	10	75
Acerbis et al.(2015)	0	0	0	0	10	10	0	0	50

Fonte: Elaborado pelo autor.

No trabalho de LAUREANO-CRUCES et al. (2014), são discutidos os aspectos recomendáveis ao projetar uma interface que inclua um agente pedagógico colaborativo dentro de um contexto em que o processo de aprendizado colaborativo seja reforçado pelo processo de distribuição de tarefas que o acompanha.

REKHAWI; NASER (2018) descrevem o design de um sistema de tutoria inteligente baseado na Web para ensinar o desenvolvimento de aplicativos Android aos alunos. A ideia do sistema é uma introdução sistemática ao conceito de desenvolvimento de aplicativos Android. O sistema apresenta o tópico "Desenvolvimento de aplicativos Android" e administra problemas gerados automaticamente para que os alunos resolvam, quantificando o tempo que eles utilizaram para resolver e relacionando com o crescimento individual de cada aluno. O trabalho ainda realiza uma avaliação inicial do efeito do uso do STI no desempenho dos alunos matriculados no curso de Desenvolvimento de Aplicativos para Smartphone, na Faculdade de Ciências Aplicadas da Universidade de Gaza.

Já o trabalho de VERDÚ et al. (2014), descreve uma interface de usuário de um STI que permite a personalização de sistemas de e-Learning para alunos que necessitam de recomendação ao longo de caminhos pedagógicos. A interface foi implementada para a plataforma de e-Learning do Moodle, usando um mecanismo de extensão padrão fornecido pelo Sistema de Gerenciamento de Aprendizagem (LMS). A interface do e-Learning Moodle permite que o STI interaja tanto com o LMS, quanto com o aluno, cujo aprendizado é continuamente monitorado e suportado por meio de recomendações e mensagens personalizadas. Os resultados mostram que a plataforma projetada foi bem integrada ao Moodle, gerando uma interface de usuário não intrusiva. A usabilidade do sistema foi avaliada no contexto de um curso sobre Design de Redes de Computadores.

Um STI adaptado para cursos online abertos e massivos (MOOC), que ensina na prática o design de sistemas digitais, é apresentado por BANERES; SAÍZ (2016). O aprendizado dessa habilidade tem várias dificuldades, principalmente para iniciantes na área de especialidades de Ciência da Computação ou Engenharia Eletrônica. O problema agrava em ambientes virtuais sem interação cara a cara com o instrutor. Os STIs são adequados para ambientes MOOC, pois promovem o processo de autoaprendizagem e melhoram o suporte ao feedback personalizado.

EDUARDO; HUGO (2014) apresentam um Tutor de exemplo de rastreamento, um novo tipo de sistema de tutoria inteligente (ITS), que apoia a experiência de aprendizado dos estudantes de engenharia mecatrônica. Um sistema de pontuação baseado em diretrizes de design de ITS, levando à seleção e priorização de um conjunto relevante e geral de problemas mais bem classificados, retirados de um grande espaço inicial de problemas criado por uma revisão do material divulgado nas classes anteriores. São fornecidas experiências sobre o uso avançado da ferramenta de criação e discussões sobre questões de desenvolvimento.

AREVALILLO-HERRÁEZ et al. (2017) descreve uma interface gráfica de usuário de um STI que permite aprender e praticar os aspectos procedimentais que estão envolvidos na tradução das informações de um problema descrito por extenso para uma representação simbólica. O processo de tradução de um problema de palavras em uma forma algébrica foi tratado isoladamente e claramente separado da manipulação algébrica. A interface do usuário foi desenvolvida para forçar uma abordagem sistemática à solução de problemas e também evitar o uso de um raciocínio não algébrico. O suporte afetivo sem sensor foi adicionado usando uma abordagem de aprendizado de máquina que se baseia em dados capturados de uma série de sessões experimentais envolvendo 48 indivíduos. A avaliação da aplicação resultante revelou um impacto positivo e significativo nos ganhos de aprendizagem.

MENON (2018) envolveu o design e o desenvolvimento interativo do Teacher Tools, criado com a contribuição de professores e outros especialistas. O Teacher Tools é um aplicativo da Web desenvolvido como parte do sistema MathSpring.org Intelligent Tutoring - o componente com o qual os professores interagem, para organizar as aulas e analisar os dados resultantes de seus alunos. No estudo, foi redesenhado a versão existente do modulo do professor do MathSpring com base no feedback coletado. Em resumo, foi criado um produto de software para professores que complementa o sistema de tutoria MathSpring, que resume informações valiosas dos registros de dados em visualizações e outras representações. Essas “Ferramentas do Professor” se mostraram úteis para os professores das escolas de ensino fundamental de Massachusetts, que afirmam estarem prontos para usar essas informações para alterar seus planos de ensino.

VANLEHN (2016) concentra em contribuições para entender como uma interface de usuário sem objetivos afeta o design e o desempenho gerais de um sistema de tutoria baseado em etapas. Enquanto uma interface de usuário alinhada a metas exibe aquelas relevantes, como, caixas em branco ou locais vazios que o aluno precisa preencher com conteúdo específico. A análise de interface é feita sobre o STI Andes Physics Tutor.

MAHMOUD; EL-HAMAYED (2016) apresentam um STI chamado “Tutor da gramática árabe” conhecido como “AG_TUTOR”. No trabalho é mostrado uma interface derivada do ITSB (Intelligent Tutoring System Builder) e discutido os módulos: o módulo do tutor, o módulo seletor de perguntas e o módulo especialista. No currículo da gramática árabe da quarta série, em escolas primárias no Egito, é adotado como um conhecimento de domínio.

HOOSHYAR et al. (2016) discutem sobre a eficácia dos STIs em relação a motivação do aprendiz, quando falta interação e orientação oportuna. Para resolver esse problema, um STI baseado em solução é integrado a um jogo de avaliação formativa. A solução baseada em jogos on-line, chamada questionário tic-tac-toe para single-player (TRIS-Q-SP), ensina programação de computadores. Após um estudo experimental com a nova

ferramenta, realizações significativas são observadas e os resultados indicam vantagens consideráveis. As descobertas revelam que o feedback elaborado imediato ao responder a cada pergunta no TRIS-Q-SP faz parte de um design ideal.

Já WERAGAMA; REYE (2014) destacam a abordagem adotada por um STI, chamado PHP ITS, em analisar os trabalhos dos alunos que incluem várias construções de programas semanticamente equivalentes no desenvolvimento web. O PHP ITS tem como objetivo facilitar para os iniciantes o aprendizado da linguagem PHP, a fim de desenvolver páginas web dinâmicas. O PHP ITS consegue fazer tal análise utilizando teorias de Inteligência Artificial (IA), incluindo lógica de predicados de primeira ordem e planejamento clássico e hierárquico para modelar o assunto ensinado pelo sistema. O STI em assunto foi construído usando esse modelo e avaliado em uma unidade da Universidade de Tecnologia de Queensland. Os resultados mostraram que foi capaz de analisar corretamente mais de 96% das soluções para exercícios fornecidos pelos alunos.

KHACHATRYAN et al. (2014) analisa abordagens baseadas na engenharia do conhecimento. Para tal, foi projetado um sistema de software entrevistando especialistas humanos para extrair seu conhecimento e heurística. O sistema Genie 2, da Reasoning Mind, demonstra a viabilidade de trazer elementos da educação matemática russa (conhecida por sua eficácia) para os Estados Unidos. Este foi adotado em larga escala, com cerca de 67.000 estudantes dos Estados Unidos participando no ano letivo de 2012-2013. Além disso, relata-se uma alta aceitação de alunos e professores, aumentos nas pontuações dos testes, menor tempo de tarefa e um perfil afetivo positivo. Nesse trabalho, é descrito o design, função e uso do sistema Genie 2, como também é sugerido uma metodologia para o design de STIs destinados à transferência intercultural de currículos e métodos instrucionais.

No artigo de JUNIOR; FREITAS et al. (2018) é apresentado uma proposta de implementação de um módulo instrucional para um sistema de apoio ao ensino baseado em Processos de Decisão de Markov parcialmente observáveis. A escolha do modelo justificas-se por sua capacidade de tomada de decisões diante das incertezas, quanto ao conhecimento que o aprendiz apresenta frente aos conceitos a serem assimilados, e pelos efeitos não determinísticos das ações propostas. O problema do sequenciamento de ações produzidos por este módulo pode ser representado como um problema de planejamento probabilístico e é, portanto, resolvido por algoritmos de planejamento. Foi aplicado uma metodologia baseada em aprendizagem de máquina para a seleção de um planejador probabilístico que proporcione a escolha das melhores ações instrucionais. A escolha do planejador é realizada considerando as características de um domínio de especificação que ensina conceitos e os avalia conforme a proficiência do estudante. O desempenho obtido pela execução foi comparado com a de outros planejadores candidatos, o que comprovou, dessa forma, a eficiência na aplicação do método proposto.

QUEIROZ et al. (2018) investigam a percepção e propagação da usabilidade em modelos IFML para a interface final. Para desenvolver a pesquisa, realizou-se um estudo empírico para avaliar como a usabilidade é percebida pelos participantes, por meio de modelos IFML, e se a usabilidade é propagada para um protótipo de interface. O estudo mostrou que mesmo IFML sendo uma proposta que suporta a modelagem da interface, nem todos os aspectos da usabilidade são facilmente percebidos e propagados na interface por meio de modelos.

BRAMBILLA et al. (2016) propõem uma nova abordagem que suporta o design de aplicativos da Web e móveis de uma só vez. Começando com um requisito exclusivo e uma especificação comercial, em que aspectos específicos da web e móveis são capturados por meio de marcação, derivaram em um design independentemente da plataforma do sistema especificado em IFML. Esse modelo é posteriormente refinado e detalhado, para as duas plataformas, e usado para gerar automaticamente as versões web e móvel.

WAKIL; JAWAWI (2017) analisam o modelo IFML no ciclo de vida do desenvolvimento do processo, para mostrar a capacidade do método usado no desenvolvimento. Em seguida, fazem uma comparação entre o IFML e outros métodos nas fases do ciclo de vida. Por fim, é adicionado o IFML ao mapa do ciclo de vida da engenharia da web. O resultado deste artigo foi a proposta de um guia para os desenvolvedores usarem IFML no desenvolvimento de novos aplicativos.

BARCELONA et al. (2017) expõe uma metodologia *Model Driven Web Engineering* (MDWE) para incluir a interação baseada na Web no desenvolvimento de Sistemas de Sistemas (SoS). É composto por dez modelos e sete transformações de modelo e é totalmente implementado em uma ferramenta de suporte para seu uso na prática. São expostos os aspectos de qualidade cobertos pela rastreabilidade, desde os requisitos até o código final. A viabilidade da abordagem é validada por sua aplicação em um projeto real. Uma análise preliminar dos benefícios em potencial (redução de esforço, tempo, custo; melhoria da qualidade; relação entre design e código, etc.) é feita em comparação com outro projeto como uma hipótese inicial para uma pesquisa de experimentação planejada futura.

LAAZ; MBARKI (2016) apresentam uma nova abordagem de desenvolvimento orientada a modelo com base no *Model Driven Engineering* (MDE), que mescla os componentes das *user interfaces* (UIs) da descrição lógica e suas interações capturadas com o IFML, a fim de aprimorar a apresentação dos aplicativos da Web da interface do usuário. A abordagem é ilustrada com um exemplo de uma interface de usuário real simples, o processo começa com modelos abstratos, para produzir um modelo html5 como modelo de destino.

ROUBI; ERRAMDANI; MBARKI (2016) demonstram uma abordagem para a geração orientada a modelo de RIAs (Rich Internet Application) utilizando IFML. A

abordagem explora a nova linguagem IFML, estendendo primeiro a parte gráfica do Meta Model para atender às necessidades dos RIAs. Usaram estruturas e tecnologias conhecidas pela engenharia orientada a modelos, como Eclipse Modeling Framework (EMF) para Meta Modelos, Query View Transformation (QVT) para transformações de modelo e Acceleo para geração de código.

BLÁZQUEZ (2018), por sua vez, aborda sobre automação das tarefas de desenvolvimento, adicionando novas funcionalidades ao plug-in AutoCRUD, que multiplicam as possibilidades de geração e customização de operações CRUD (*create*, *read*, *update* e *delete*). Após as melhorias, o usuário pode definir o conjunto de elementos IFML que compõem cada operação CRUD e gerar automaticamente esse padrão de elementos no modelo da web. Isso elimina a necessidade de retocar manualmente o modelo da web após a geração automática. O processo de geração permite ao usuário controlar a velocidade de arranjo dos elementos no modelo. Além disso, tornou possível analisar as informações dos padrões e elementos gerados em um projeto da Web específico, juntamente com vários dados estatísticos de interesse.

BERNASCHINA; COMAI; FRATERNALI (2018) discutem sobre a geração de código a partir de modelos, o qual requer a interpretação inequívoca da semântica das linguagens de modelagem usadas para especificar o aplicativo. Dessa forma, apresenta a formalização da semântica do IFML e suas construções mapeadas para estruturas equivalentes de redes de gráficos de locais (PCN), o que permite sua interpretação precisa. O mapeamento semântico definido é implementado em um ambiente de desenvolvimento on-line orientado a modelos, que permite a criação e inspeção de PCNs a partir do IFML, a análise do comportamento das especificações IFML via simulação PCN e a geração de código para arquiteturas móveis e baseadas na Web.

FALZONE; BERNASCHINA (2018) demostram um fluxo de trabalho de desenvolvimento para a co-evolução de modelo e código, com base no IFMLEdit.org , uma ferramenta on-line para a prototipagem rápida de aplicativos da Web e em sistemas de controle de versão comuns. Na demonstração, os participantes poderão editar as especificações IFML com o IFMLEdit.org, gerando a primeira versão do código de um aplicativo Web/móvel a partir do modelo. Depois do processo, foi melhorado o código gerando detalhes adicionados manualmente (por exemplo, estilo), evoluindo o modelo IFML original, ao introduzindo novos requisitos, e novamente geraram o código da versão atualizada, de uma maneira que preserve totalmente os detalhes codificados manualmente. A abordagem demonstrou resolver o problema conhecido da engenharia avançada orientada a modelos para interromper o ciclo de desenvolvimento automatizado quando os recursos que não podem ser modelados são adicionados manualmente ao código gerado.

ACERBIS et al. (2015) propuseram um conjunto abrangente de ferramentas chamado WebRatio Platform, que seria utilizado para o desenvolvimento orientado a modelos

de aplicativos da *Web* e móveis. A ferramenta suporta desenvolvedores na especificação do modelo de domínio e do modelo de interação do usuário para aplicativos de acordo com duas versões estendidas da linguagem padrão OMG, denominada IFML. De acordo com o autor, as extensões apresentaram primitivas para o desenvolvimento de aplicativos da web e para sistemas móveis. A ferramenta apresentou verificação de modelo e geração completa de código que produz aplicativos móveis da Web e de plataforma cruzada prontos para publicação.

3.2.4 Extração de Dados

De acordo com a avaliação de qualidade dos estudos selecionados, realizada na subseção 3.2.3, quatro publicações foram escolhidas para uma leitura mais profunda com a finalidade de extrair os dados e analisar com mais detalhes a abordagem sobre o modelo de interface no STI e a modelagem IFML, com boas práticas de usabilidade, de acordo com heurísticas de Nielsen.

A partir da leitura proposta nesta subseção as publicações MORAIS; SCHAAB; JAQUES (2017), NGO; TRAN (2017), WAKIL; JAWAWI (2017) e OLIVEIRA (2018) foram escolhidos para serem base na modelagem formal de um protótipo utilizando IFM. Esses trabalhos são de suma importância para embasar a estrutura e usabilidade do protótipo do modelo de comunicação do STI, saciando as necessidades do processo de ensinamento de programação. Dessa forma, eles serão analisados com mais profundidade. Outro fator impactante na escolha destes trabalhos, foi a apresentação de interface com mais detalhes em relação aos demais, mesmo ainda não apresentando um modelo de protótipo completo com a utilização de IFML.

MORAIS; SCHAAB; JAQUES (2017) demonstraram um estudo de caso que utilizou o protocolo *Think Aloud* para a avaliação da interface e usabilidade do STI PAT2Math, mais especificamente, sua ferramenta PATequation. A avaliação foi baseada nas heurísticas de usabilidade, propostas por Nielsen, e teve como objetivo identificar os pontos positivos e problemas de usabilidade na interface gráfica do STI, assim como observar alguns estados emotivos e reações dos alunos ao se deparar com algum problema ou empecilho na interface. O estudo de caso mostrou que o protocolo permite obter informações previamente não previstas em relação às questões funcionais da interface gráfica e às emoções dos usuários. É ressaltado que, embora como estudo de caso tenha sido realizado com um tutor algébrico específico, a metodologia empregada pode ser utilizada para avaliação de usabilidade de interfaces gráficas de sistemas tutores em geral. O STI utilizado como base de estudo, se encontra disponível no endereço: http://pat2math.unisinos.br.

A Figura 59 ilustra a interface gráfica antiga do PATequation. Nesta, é exibida a resolução de uma equação de primeiro grau. A cada passo da resolução do estudante, o sistema fornece um feedback mínimo imediato, através do ícone com o símbolo “Ok”

Figura 59 – Antiga interface gráfica da ferramenta PATequation

Fonte: MORAIS; SCHAAB; JAQUES (2017)

(ícone verde ao lado de cada passo).

Na Figura 60 é ilustrado a nova interface gráfica do PATequation, visando uma maior simplicidade na sua utilização, permitindo uma maior liberdade ao aluno nas resoluções de equações. A nova interface levou em consideração todos os problemas de usabilidade identificados na avaliação Think Aloud e na avaliação heurística.

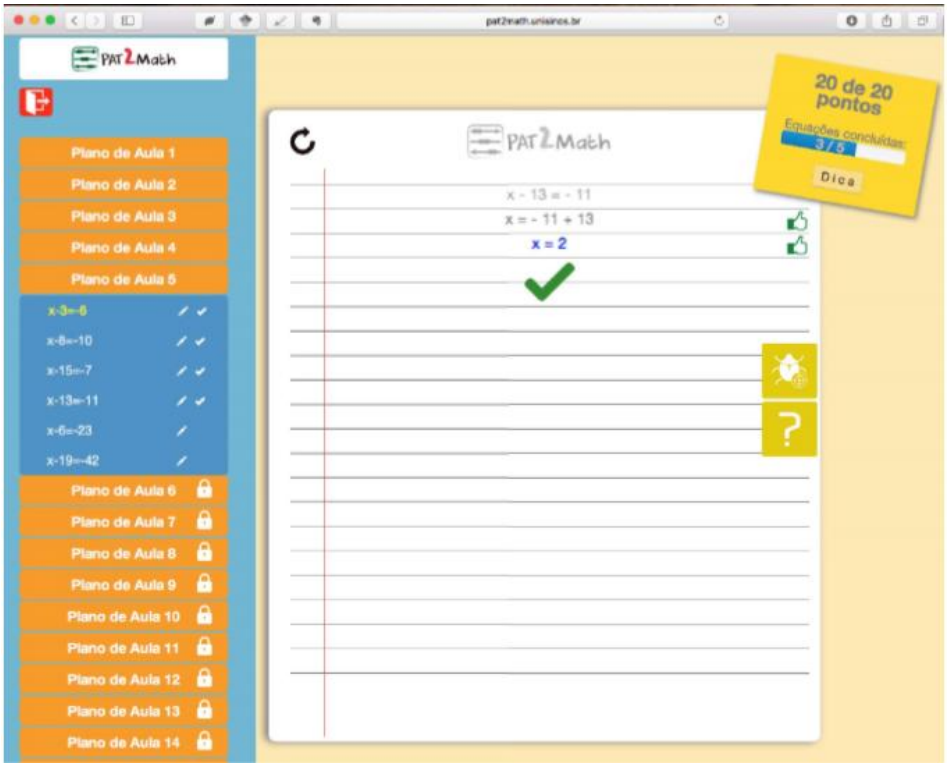
NGO; TRAN (2017) desenvolveram um novo design para um STI chamado *MathSpring*, que tem como objetivo ajudar no aprendizado em matemática. O desenvolvimento do novo design aborda melhorias, as quais a equipe levantou analisando quantitativamente as sugestões dos alunos e professores que utilizaram o STI em questão. Os resultados apresentados mostram melhora na experiência do usuário.

Vale ressaltar que este trabalho foi o único da seleção do grupo de STIs que apresentou um maior detalhamento da sua modelagem de interface, mostrando o processo de modelagem a partir de uma interface de baixa fidelidade (figura 61) até o protótipo final apresentado nas Figuras 62, 63 e 64. Porém, mesmo uma apresentação com mais detalhes, ainda não foi utilizado alguma modelagem formal para tal.

WAKIL; JAWAWI (2017) analisam o modelo IFML no ciclo de vida do desenvolvimento do processo, para mostrar a capacidade do método usado no desenvolvimento do processo. Em seguida, fazem uma comparação entre o IFML e outros métodos nas fases do ciclo de vida de um projeto. Por fim, é adicionado o IFML ao mapa do ciclo de vida em um caso estudo de engenharia da web.

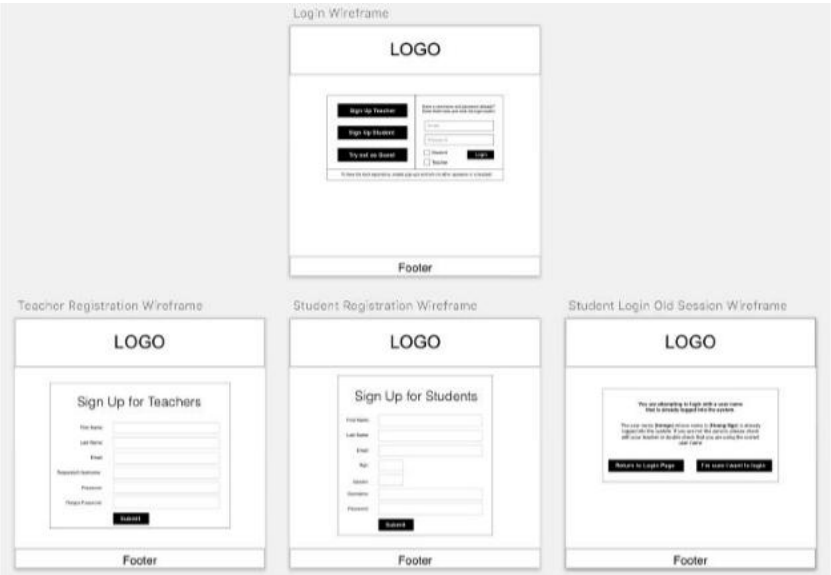
No mesmo foi realizado como caso de estudo a engenharia reversa do site Amazon

Figura 60 – Nova interface gráfica da ferramenta PATequation.



Fonte: MORAIS; SCHAAB; JAQUES (2017)

Figura 61 – Wireframe do processo de registro e login de usuários do STI MathSpring.



Fonte: NGO; TRAN (2017)

Prime, produzindo seu modelo IFML. As Figuras 65, 66 e 67 representam o processo reverso da produção de um IFML a partir de uma aplicação já desenvolvida, em específico o fluxo de selecionar um filme na lista da aplicação. Dessa forma, o resultado deste artigo se denomina um guia para os desenvolvedores usarem IFML no desenvolvimento de novos aplicativos e defende os benefícios do processo.

Figura 62 – Página de registro de novos estudantes do STI MathSpring.

Fonte: NGO; TRAN (2017)

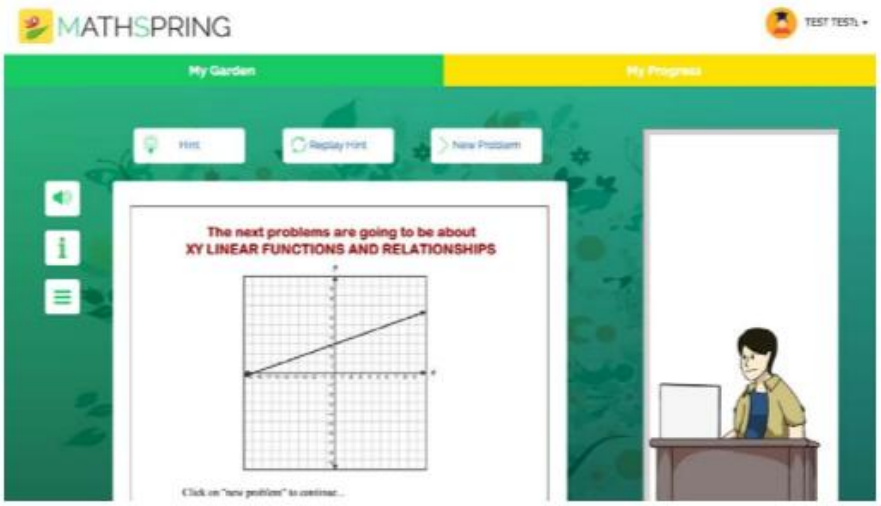
Figura 63 – Página de boas-vindas e dashboard do STI MathSpring.

Fonte: NGO; TRAN (2017)

OLIVEIRA (2018) propõe o desenvolvimento de novas funcionalidades para o projeto SisGera, o qual foi pensado com o propósito de apoiar as atividades administrativas das corporações, realizando um estudo em abordagens utilizadas no desenvolvimento de aplicações Web, bem como a utilização de ferramentas voltadas ao desenvolvimento Web para implementação e validação das novas funcionalidades como a modelagem IFML. No desenvolvimento do sistema, foi apresentado um estudo de abordagens e ferramentas utilizadas na atualidade, como a utilização de técnicas para o projeto do tal software. Uma das ferramentas abordadas foi a modelagem de fluxo com IFML.

As Figuras 68 e 69 apresentam o fluxo detalhado de controle de bens patrimoniais e a tela final a partir do modelo IFML. No trabalho, são listados os vários benefícios que o IFML traz no processo de desenvolvimento de aplicações, pelo fato de estar particular-

Figura 64 – Página principal do STI MathSpring.



Fonte: NGO; TRAN (2017)

Figura 65 – Página principal do Amazon Prime.

Fonte: WAKIL; JAWAWI (2017)

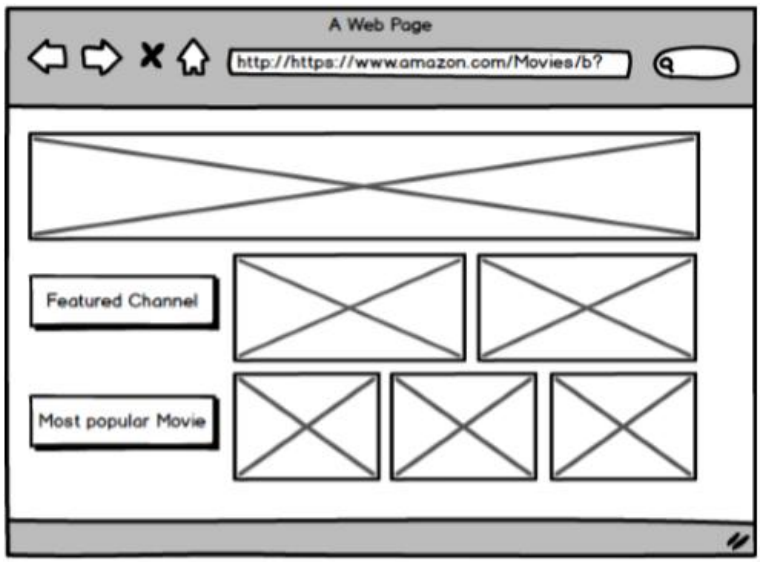
mente dedicado à modelagem de usabilidade e inteligibilidade.

3.2.5 Sintetização de Dados

Nesta seção, será apresentada uma sintetização de dados quantitativa dos trabalhos selecionados na subseção 3.2.3. Na tabela 9, são expostos quais trabalhos apresentam um modelo de interface de um STI e qual modelagem formal utilizada. Na mesma relação quantitativa, foi apresentado também, na tabela 10, o endereço de acesso de cada uma das plataformas desenvolvidas.

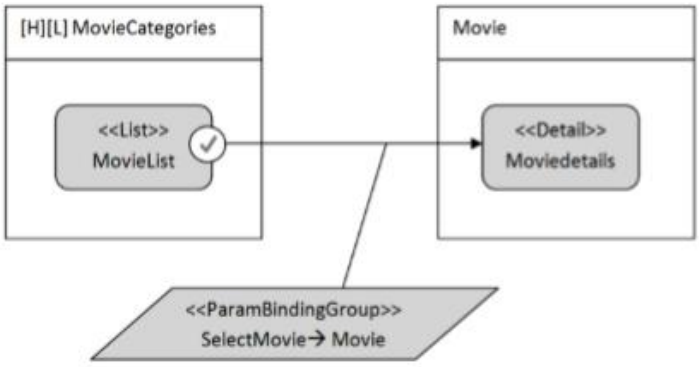
Vale ressaltar que nem todos os trabalhos apresentam um modelo formal para a modelagem do software. Da mesma forma, nem todos disponibilizam o acesso a plataforma desenvolvida.

Figura 66 – Wireframe da página principal do Amazon Prime.



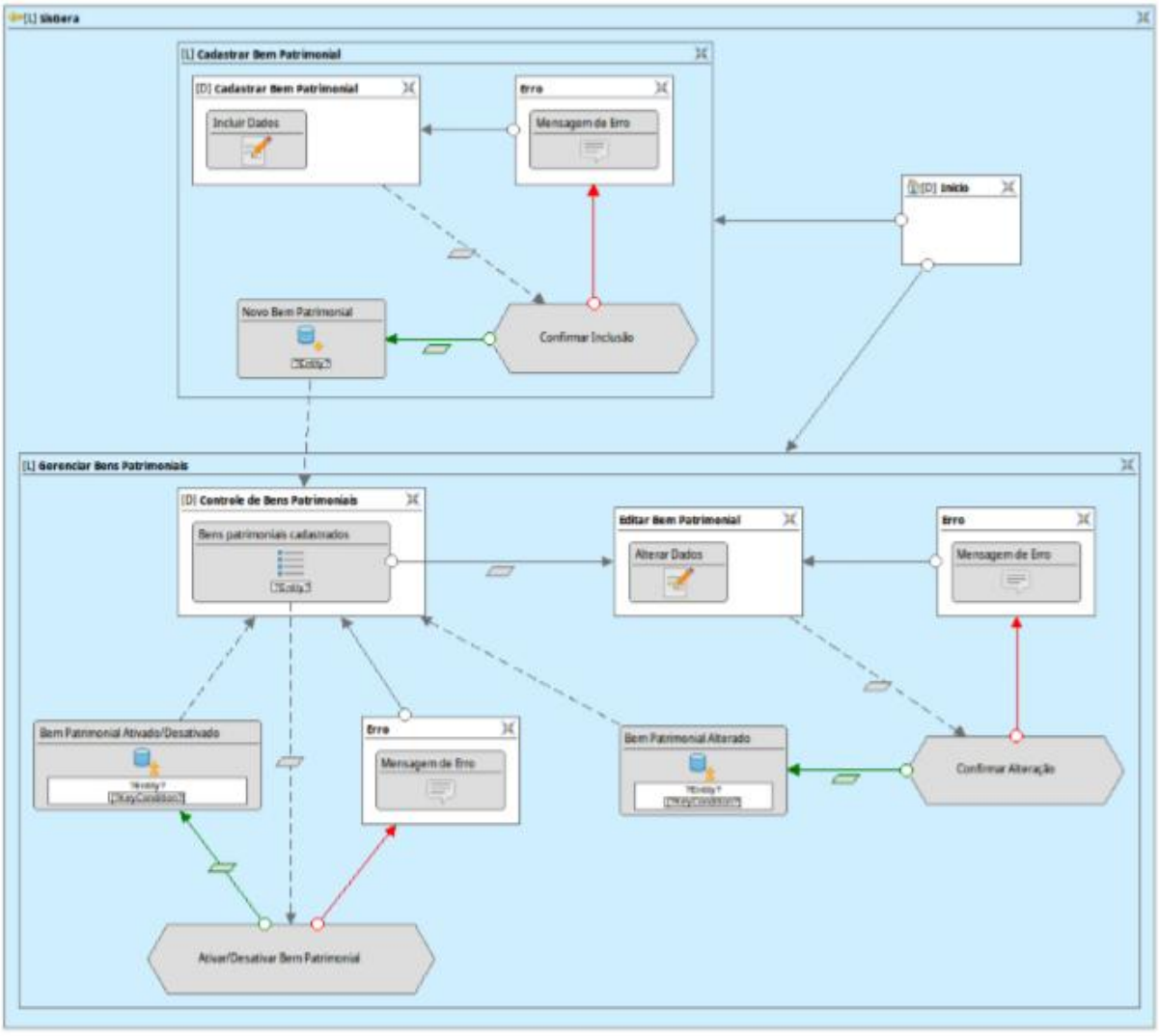
Fonte: WAKIL; JAWAWI (2017)

Figura 67 – IFML do fluxo de selecionar filme na plataforma Amazon Prime.



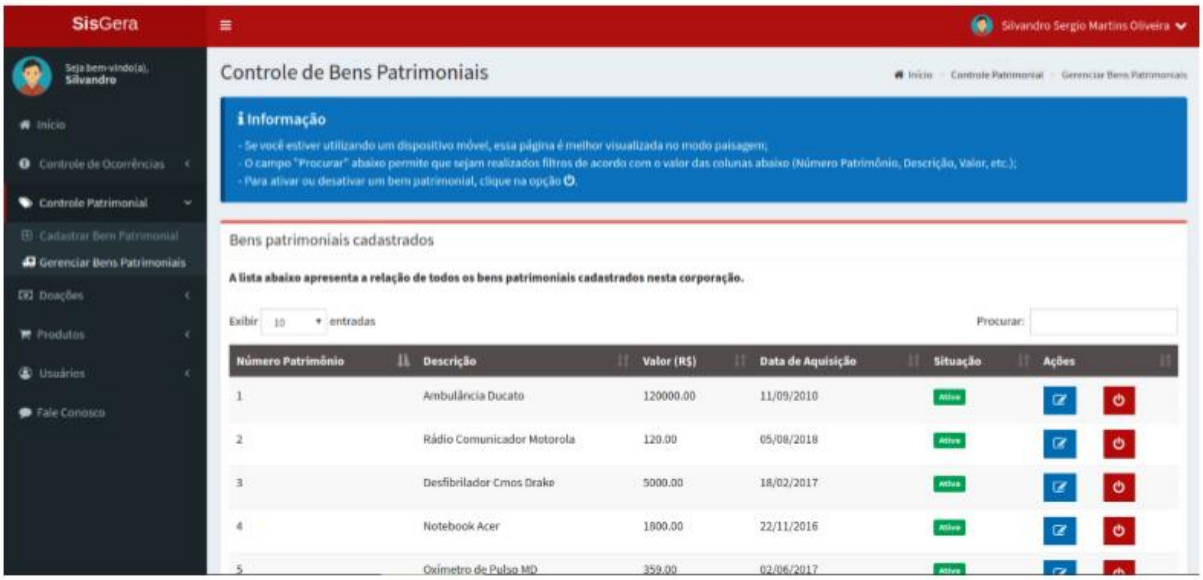
Fonte: WAKIL; JAWAWI (2017)

Figura 68 – IFML do fluxo de controle de bens patrimoniais do SisGera.



Fonte: OLIVEIRA (2018)

Figura 69 – Tela Gerenciador de Bens Patrimoniais da plataforma SisGera.



Fonte: OLIVEIRA (2018)

Tabela 9 – Sintetização de dados de método de modelagem e interface.

Base	**Artigo**	**Modelo Interface STI**	**Método de Modelagem**
ACM	Queiroz et al.(2018)	NÃO	IFML
	Brambilla et al.(2016)	NÃO	IFML MobML
BDTD	Galafassi(2019)	SIM	-
Google Scholar	Al-Bastami; Naser(2017)	SIM	-
	Laureano-Cruces et al.(2014)	SIM	-
	Rekhawi; Naser(2018)	SIM	-
	Wakil; Jawawi(2017)	NÃO	IFML
	Barcelona et al.(2017)	NÃO	IFML UML
	Oliveira (2018)	NÃO	IFML
IEEE Xplorer	Verdú et al.(2014)	SIM	-
	Baneres; Saíz(2016)	SIM	-
	Morais; Schaab; Jaques (2017)	SIM	-
	Eduardo; Hugo(2014)	NÃO	-
	Arevalillo-Herráez et al.(2017)	NÃO	-
	Laaz; Mbarki(2016)	NÃO	-
	Roubi; Erramdani; Mbarki(2016)	NÃO	IFML UML
Micrososft Academic	Ngo; Tran(2017)	SIM	wireframe
	Menon(2018)	SIM	-
	Vanlehn(2016)	NÃO	-
	Blázquez(2018)	NÃO	IFML UML
ScienceDirect	Mahmoud; El-Hamayed(2016)	SIM	-
	Bernaschina; Comai; Fraternali(2018)	NÃO	IFML UML
Springer	Hooshyar et al.(2016)	SIM	Fluxograma
	Weragama; Reye(2014)	SIM	-
	Khachatryan et al.(2014)	SIM	-
	Falzone; Bernaschina (2018)	NÃO	IFML
	Acerbis et al.(2015)	NÃO	IFML
UFU	Junior; Freitas et al.(2018)	SIM	-

Fonte: Elaborado pelo autor.

Tabela 10 – Sintetização de dados de acesso às plataformas.

Base	**Artigo**	**Plataforma Acessível**
ACM	Queiroz et al.(2018)	-
	Brambilla et al.(2016)	-
BDTD	Galafassi(2019)	-
Google Scholar	Al-Bastami; Naser(2017)	-
	Laureano-Cruces et al.(2014)	-
	Rekhawi; Naser(2018)	-
	Wakil; Jawawi(2017)	-
	Barcelona et al.(2017)	primevideo.com
	Oliveira (2018)	-
IEEE Xplorer	Verdú et al.(2014)	sisgera.com.br
	Baneres; Saíz(2016)	github.com/juacas/
	Morais; Schaab; Jaques (2017)	moodle-block_intuitel
	Eduardo; Hugo(2014)	wyoming.uoc.es/pubver
	Arevalillo-Herráez et al.(2017)	pat2math.unisinos.br
	Laaz; Mbarki(2016)	-
	Roubi; Erramdani; Mbarki(2016)	-
Micrososft Academic	Ngo; Tran(2017)	-
	Menon(2018)	-
	Vanlehn(2016)	mathspring.org
	Blázquez(2018)	mathspring.org
ScienceDirect	Mahmoud; El-Hamayed(2016)	andestutor.org
	Bernaschina; Comai; Fraternali(2018)	-
Springer	Hooshyar et al.(2016)	-
	Weragama; Reye(2014)	-
	Khachatryan et al.(2014)	-
	Falzone; Bernaschina (2018)	-
	Acerbis et al.(2015)	my.reasoningmind.org
UFU	Junior; Freitas et al.(2018)	IFMLEdit.org

Fonte: Elaborado pelo autor.

Parte I

Desenvolvimento

4 Levantamento dos Requisitos

Este capítulo apresenta os requisitos identificados, bem como alguns modelos para representar e auxiliar no desenvolvimento, como casos de usos e diagramas de atividades.

4.1 Análise de Requisitos

A partir da necessidade identificada de modelar um modelo de comunicação de um STI, foram levantadas algumas das principais características que este deve possuir. Estes pontos levantados influenciam na escolha do modelo arquitetural para a construção da plataforma. Para o levantamento das especificações dos requisitos das funcionalidades do projeto foi utilizado a técnica de entrevista ou *Storytelling*, descrita em SEGUNDO (2015), e para apresentação das histórias foi utilizada a mesma estrutura citada em SOTTILARE (2015), sendo a seguinte:

- **Como um...** (Quem) (Papel, ator)
- **eu quero...** (O que) (Funcionalidade a ser desenvolvida)
- **de modo que...** (Por que) (Benefício a ser obtido)

Sendo assim, a Tabela 11 exemplifica uma história de um usuário administrativo com o objetivo de cadastrar um curso. O importante de levantar tais dados é deixar o mais claro possível a necessidade do sistema para se pensar futuramente no fluxo, sem precisar repensar muito nisso.

Para não poluir muito a leitura, todas as histórias geradas para as funcionalidades do Educa são apresentadas no apêndice A.

Tabela 11 – Histórias de usuário do Educa.

Cadastrar Curso
Como um usuário admin, **eu quero** cadastrar um novo curso, **de modo que** seja possível vincular um instrutor possibilitando este de adicionar aulas e atividades.

Fonte: Elaborada pelo autor.

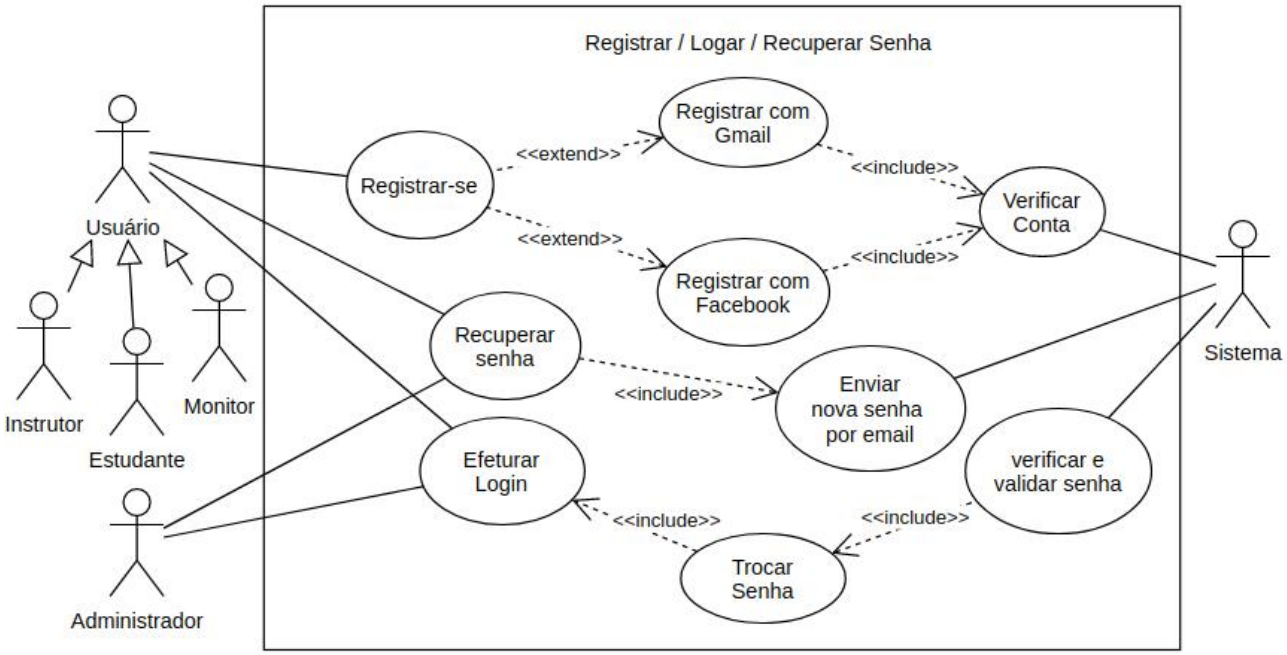
4.2 Casos de Uso

Nessa seção, é apresentanda de forma detalhada todos os casos de uso do sistema a partir dos requisitos levantados na seção anterior.

4.2.1 Registrar e logar

Nesta subseção é apresentado com mais detalhes as funcionalidades comuns de login e registro de um usuário. Tais funcionalidades são apresentadas no modelo de caso de uso na figura 70.

Figura 70 – Caso de uso das funcionalidades registrar, logar e recuperar senha.



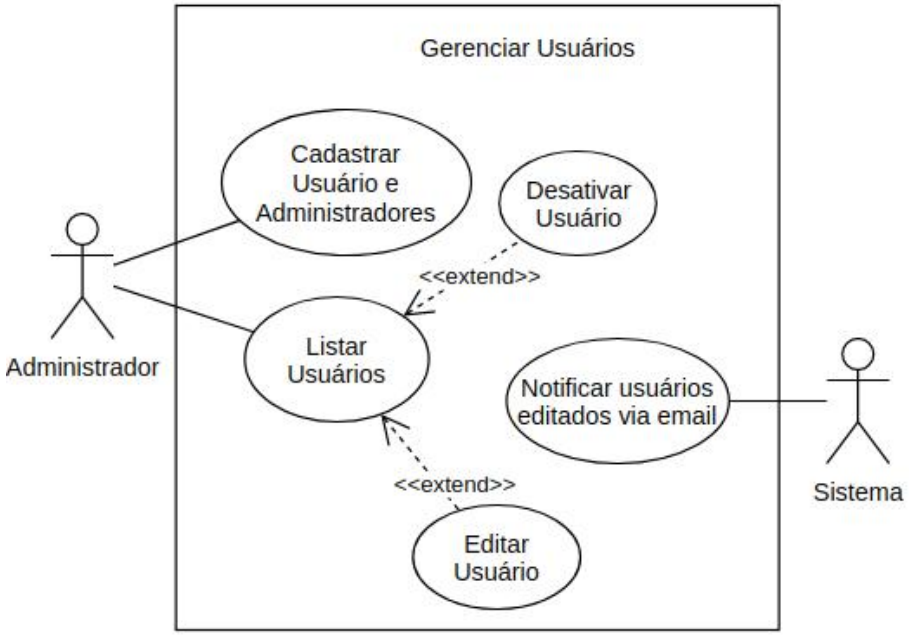
Fonte: Elaborada pelo autor

Qualquer usuário dos tipos instrutor, estudante e monitor podem registrar-se no sistema, recuperar senha e efetuar login. Administradores podem ser registrados no sistema somente por um outro administrador. Quando um usuário for se registrar, além de escolher a opção de informar um e-mail e senha de sua preferência, também pode fazer o cadastro com a conta Gmail ou Facebook, a qual será executada com sucesso após o sistema verificar que a conta é válida. Qualquer usuário pode recuperar a senha, assim, o sistema enviará uma nova senha gerada para o e-mail cadastrado na conta usuário. Qualquer usuário pode, após o sistema validar as credenciais informadas, logar-se no sistema informando o e-mail e senha. Qualquer usuário também pode trocar a senha, sendo está de acordo com as exigências de no mínimo seis caracteres e validada pelo sistema.

4.2.2 Gerenciar usuários

O caso de uso de gerenciar usuários possui funcionalidades exclusivas para usuários Administradores. As funcionalidades descritas nessa seção são apresentadas no modelo de caso de uso na figura 71. De forma que um administrador pode cadastrar novos usuários (sendo administrador ou não) como também listrar, editar e desativar outros usuários. Após ser realizada uma edição, desativação ou cadastro de um novo e-mail, o sistema notificará, de forma específica para cada caso, pelo e-mail cadastrado na conta deste usuário.

Figura 71 – Caso de uso das funcionalidades relacionadas ao gerenciamento de usuários.



Fonte: Elaborado pelo autor.

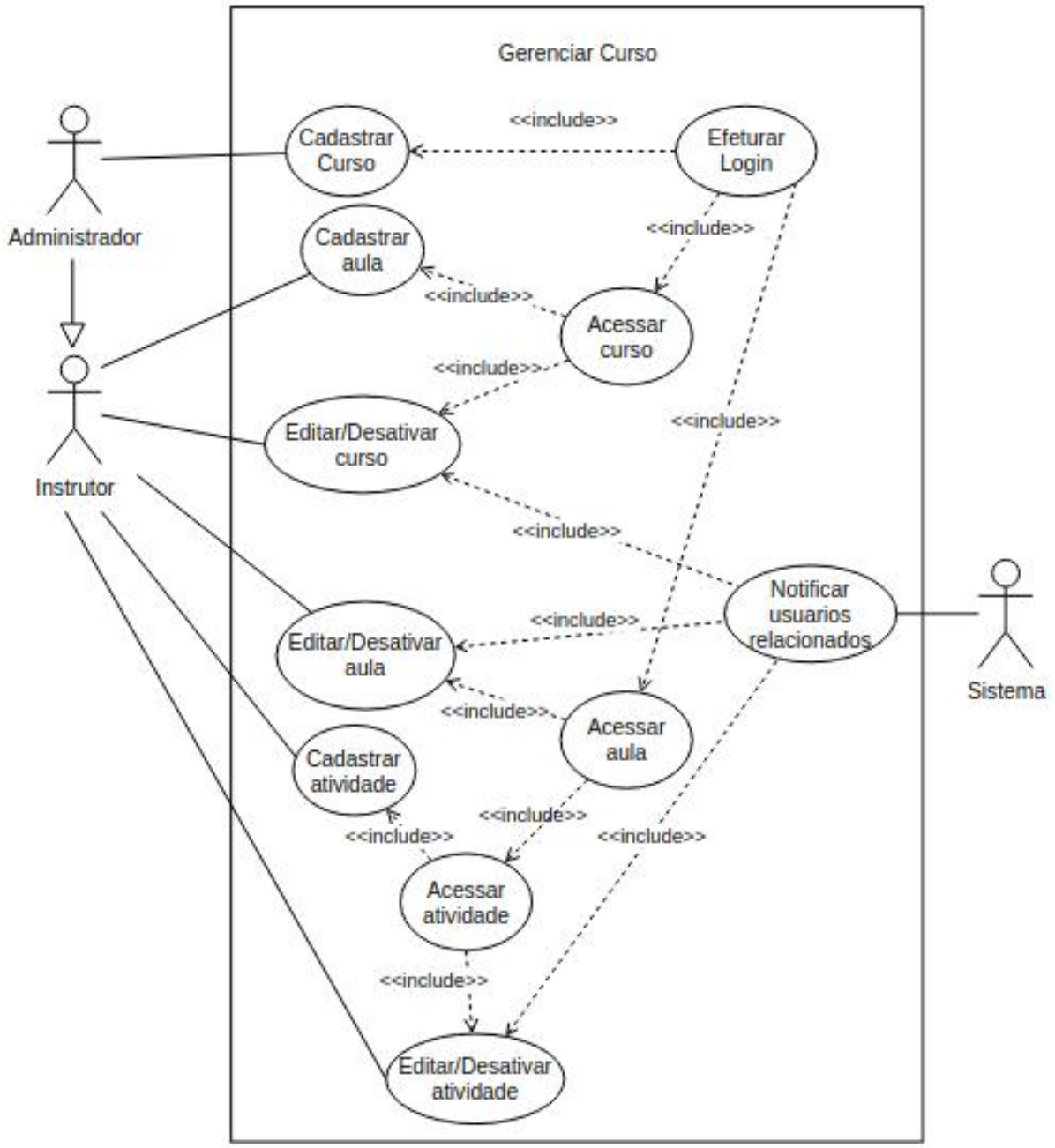
4.2.3 Gerenciar curso

Nesta subseção será detalhado o caso de uso de gerência de curso. As funcionalidades aqui descritas são apresentadas no modelo de caso de uso da figura 72. Este caso de uso abrange as principais funcionalidades:

- cadastrar, editar e desativar cursos;
- cadastrar, editar e desativar aulas;
- cadastrar, editar e desativar atividades.

Somente administradores podem cadastrar um novo e, nesse momento, é obrigatório vincular um instrutor. Administradores ou Instrutores podem editar e desativar cursos. Administrador ou Instrutores podem cadastrar, editar e desativar aulas e atividades. É

Figura 72 – Caso de uso das funcionalidades relacionadas ao gerenciamento de cursos.



Fonte: Elaborado pelo autor.

necessário estar logado no sistema para executar quaisquer das funcionalidades descritas nessa subseção.

Para cadastrar uma aula ou alterar o curso é preciso acessar previamente o curso em questão. Para cadastrar uma atividade ou alterar a aula é preciso previamente acessar a aula em questão. Após cursos, aulas ou atividades serem alteradas ou desativados o sistema notificará todos os usuários relacionados a estes. Após aulas ou atividades cadastradas em um curso em andamento, os usuários relacionados a estas também serão notificados via e-mail pelo sistema.

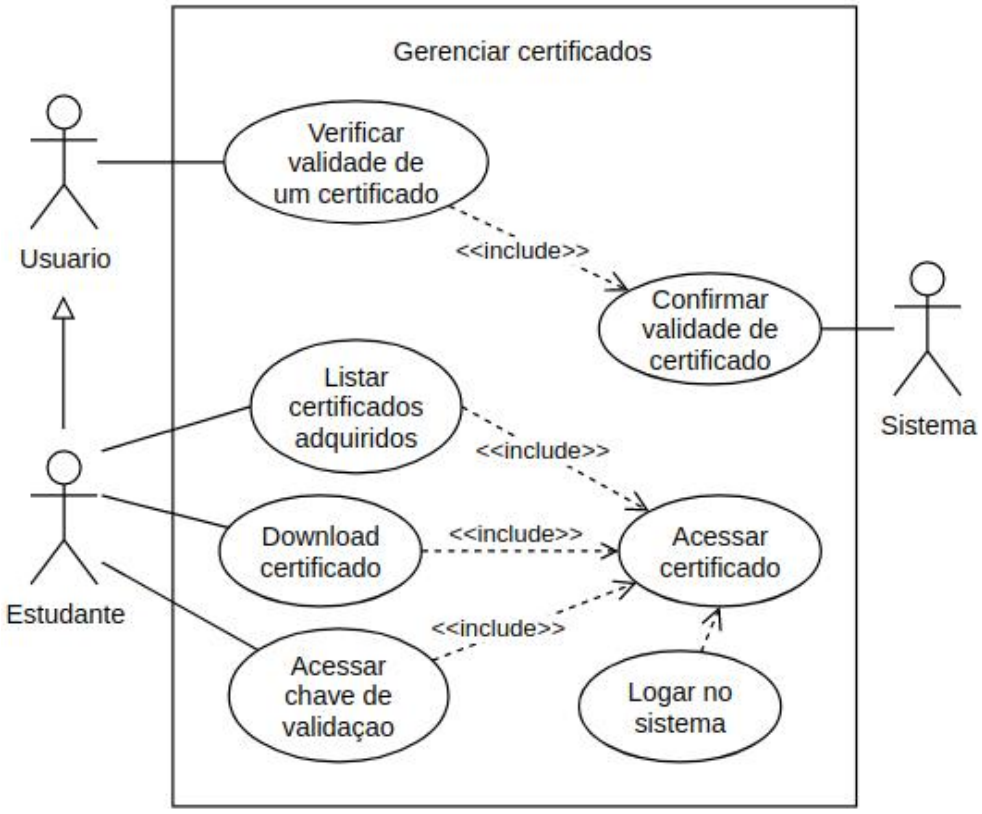
4.2.4 Gerenciar certificados

Nesta subseção será detalhado o caso de uso de gerência de certificados. As funcionalidades descritas nessa seção são apresentadas no modelo de caso de uso da figura 73. Este caso de uso abrange as principais funcionalidades:

- Verificar validade de um certificado;
- Listar certificados adquiridos;
- Realizar download;
- Acessar chave de validação.

Qualquer usuário pode validar e, assim, consultar um certificado por uma página pública, dessa forma, o usuário não precisa estar logado no sistema. Essa funcionalidade é interessante para caso seja necessário confirmar a veracidade de um certificado. Um usuário do tipo estudante pode listar certificados, realizar *download* de um certificado e acessar a chave de validação deste. Para essas funcionalidades, é necessário que o estudante esteja logado no sistema e acesse o certificado.

Figura 73 – Caso de uso das funcionalidades relacionadas ao gerenciamento de certificados.

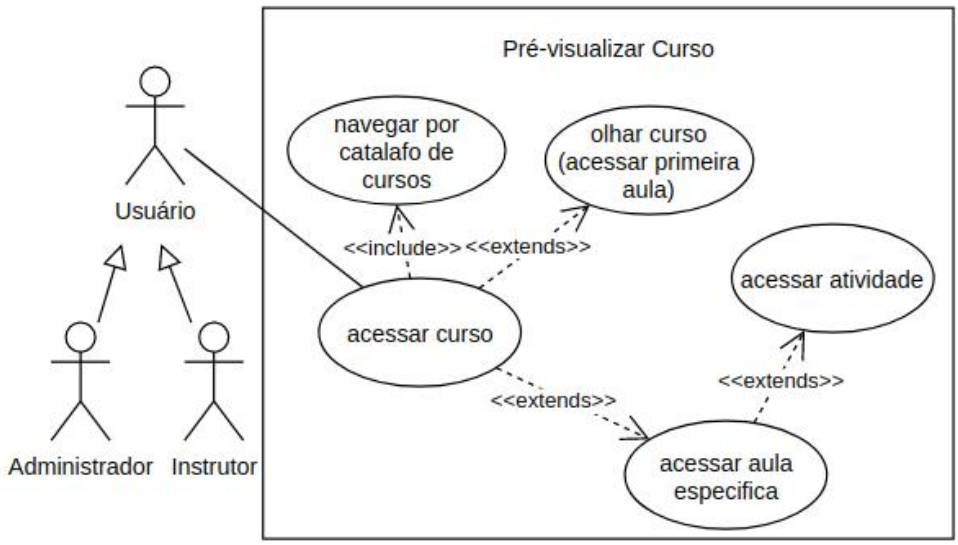


Fonte: Elaborado pelo autor.

4.2.5 Pré-visualizar curso

Qualquer usuário tem a funcionalidade de acessar um curso, assim conseguindo fazer uma pré-visualização dele. Para isso, o usuário não precisa estar logado no sistema, podendo, dessa forma, acessar o catalogo de cursos. Quando acessado um curso, esse usuário tem a opção de acessar a primeira aula clicando no botão "Olhar curso" ou acessar uma aula específica pela listagem de aulas na página do curso.

Figura 74 – Caso de uso das funcionalidades relacionadas à pré-visualização de cursos.



Fonte: Elaborado pelo autor

Após acessada a aula, o estudante pode pré-visualizar uma atividade da aula previamente acessada. A diferença dessa pré-visualização, com cursar de fato, é que a pré-visualização não salva o progresso deste usuário, como as respostas, navegação, entre outros recursos. As funcionalidades descritas nessa seção são apresentadas no modelo de caso de uso da figura 74.

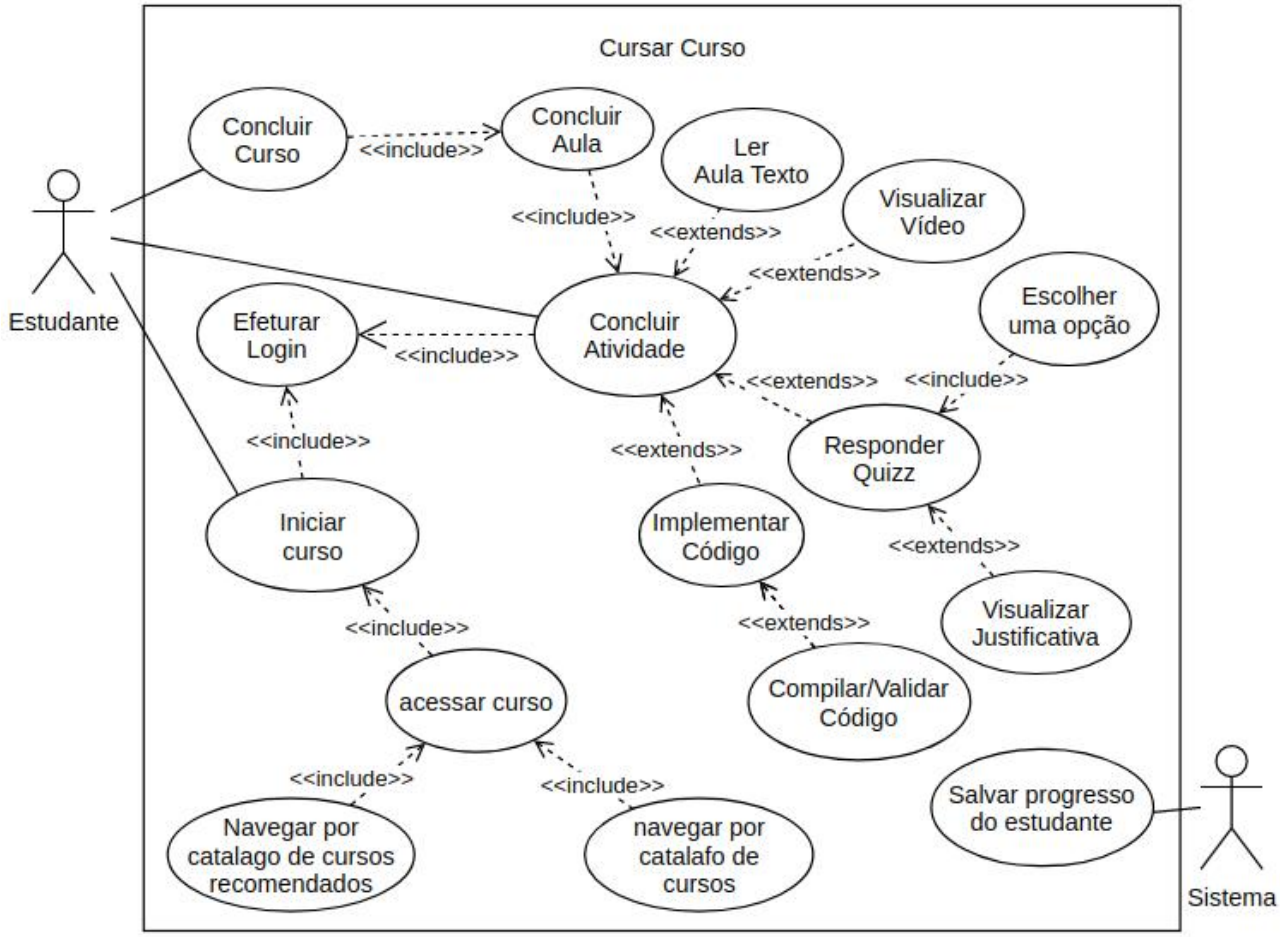
4.2.6 Cursar curso

Nesta subseção é apresentado o principal caso de uso do sistema, o qual é responsável por todo o processo de um usuário cursar um curso. Neste caso de uso, são apresentadas as principais funcionalidades:

- Iniciar Curso;
- Concluir curso;
- Concluir aula;
- Concluir atividade.

As funcionalidades descritas nessa seção são apresentadas no modelo de caso de uso da figura 75. Para quaisquer das funcionalidades descritas, o estudante precisa estar previamente logado no sistema. Um estudante pode iniciar um curso que ainda não tenha iniciado clicando no botão "iniciar curso" na tela do curso. Para acessar este curso ele pode encontrar pelo catálogo geral dos cursos ou pelo catálogo de cursos recomendados. Tal recomendação é processada pelo sistema a partir do histórico do estudante, como cursos relacionados aos já iniciados ou concluídos pelo estudante.

Figura 75 – Caso de uso das funcionalidades em cursar um curso.



Fonte: Elaborado pelo autor

Para um usuário concluir um curso, é necessário que todas as aulas sejam concluídas e, para concluir uma aula, é necessário que todas as atividades desta aula sejam concluídas. Por outro lado, para um usuário concluir uma atividade é necessário que o estudante execute o que foi proposto nesta. Nesse sentido, uma atividade pode propor 4 formas diferentes:

- Leitura de um texto;
- Assistir completamente um vídeo;

- Responder um questionário de múltipla escolha;
- Resolver um problema com codificação certa.

Para os casos de atividades de múltipla escolha, vídeo e codificação, são concluídas automaticamente após a execução da atividade. No caso de atividades de vídeo, elas são consideradas concluídas quando o vídeo é finalizado. já as atividades de codificação são consideradas pelo sistema como concluídas quando a complicação e o testes são realizados com sucesso, confirmando que o código faz o que o exercício propôs.

As atividades de questionário são consideradas concluídas quando o usuário submete uma resposta, certa ou não. Para esse caso específico de questionário, o usuário tem a opção de visualizar as justificativas de cada opção, assim, o sistema informa porquê tais opções são certas ou erradas. Por fim, as atividades de leitura de texto são consideradas concluídas quando o usuário avança para a próxima ou outra atividade.

5 Desenvolvimento e Arquitetura

Este capítulo apresenta detalhadamente a arquitetura proposta e o processo de desenvolvimento do projeto. As subseções apresentam os modelos UML e IFML com os objetivos de representar a arquitetura do projeto e auxiliar no desenvolvimento. Além disso, são apresentadas as tecnologias escolhidas e a justificativa de escolha para o desenvolvimento do projeto.

5.1 Arquitetura de Projeto escolhida

Nesta seção, serão apresentadas as decisões efetuadas para compor a arquitetura do projeto Educa. O qual integra em um sistema com o padrão de arquitetura MVC, consistindo em uma estrutura de *Models*, *Views* e *Controllers*. O projeto foi subdividido em uma API *back-end* e uma aplicação *frot-end* WEB. A separação do *front-web* em relação ao *back-end* foi uma decisão para dar maior abertura de escalabilidade e manuseio futuro à aplicação. Por exemplo, no futuro, pode ser interessante a remoção do modulo de progresso dos estudantes para um micro serviço, onde é esperado que tenha a maior carga de dados, assim, possibilitando uma abertura para uma estratégia de replicabilidade e redundância.

5.1.1 Diagrama de Atividades

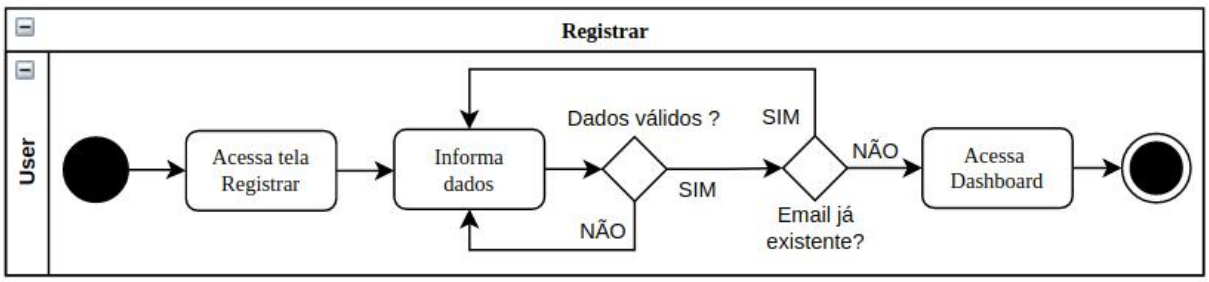
A partir dos requisitos levantados, são definidos, então, os diagramas de atividades para representar graficamente a ordem dos passos de um processo e o fluxo de controle do sistema. Sendo assim, nesta seção serão apresentados os modelos definidos.

5.1.1.1 Processo Registro, Login, Logout

Nesta subseção serão apresentados os processos de Registro, Login e Logout. Como apresentado na figura 76, o processo Registrar consiste em registrar o usuário após informados os seus dados a partir da tela de registro. O registro é realizado com sucesso, após a conferência destes dados, ou seja, se estes são válidos e se o e-mail utilizado já não está cadastrado. Após o registro, o usuário é redirecionado para a tela de Dashboard.

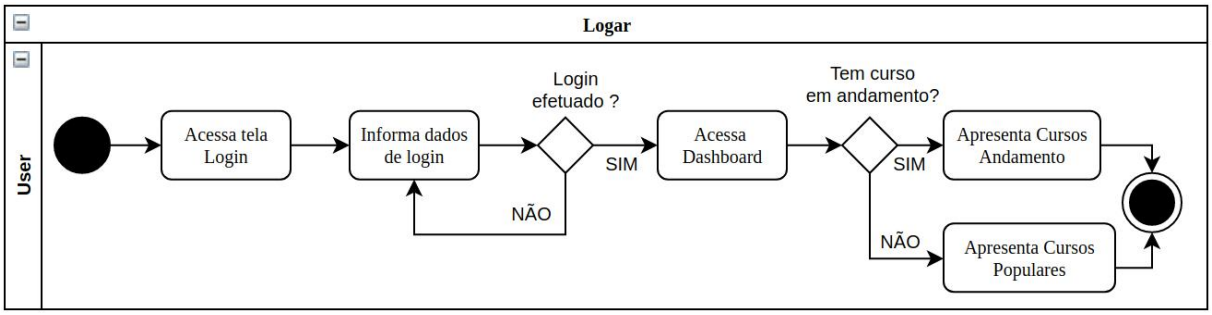
O Processo Logar, como representado na figura 77, consiste em, após informado os dados de login do usuário na tela de login, esse seja redirecionado para a tela de *dashboard*, caso os dados informados estejam corretos. Após direcionado à tela de *dashboard*, será apresentado ao usuário cursos populares ou cursos em andamento caso este usuário tenha algum curso em andamento.

Figura 76 – Diagrama de atividade do processo de registrar de um usuário novo no sistema.



Fonte: Elaborado pelo autor.

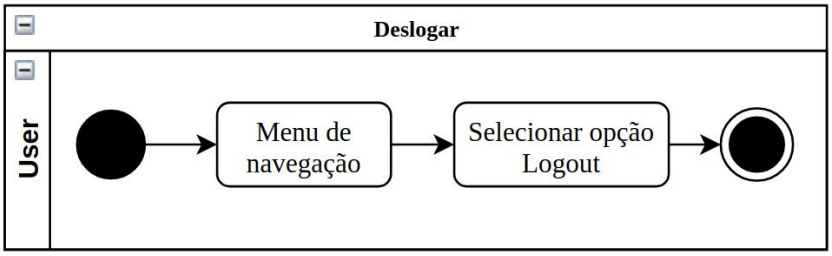
Figura 77 – Diagrama de atividade do processo de um usuário se logar no sistema.



Fonte: Elaborado pelo autor.

O Processo Deslogar, como representado na figura 78, consiste em o usuário deslogar-se após selecionar a opção *Logout* no menu de navegação.

Figura 78 – Diagrama de atividade do processo de um usuário deslogar-se do sistema.



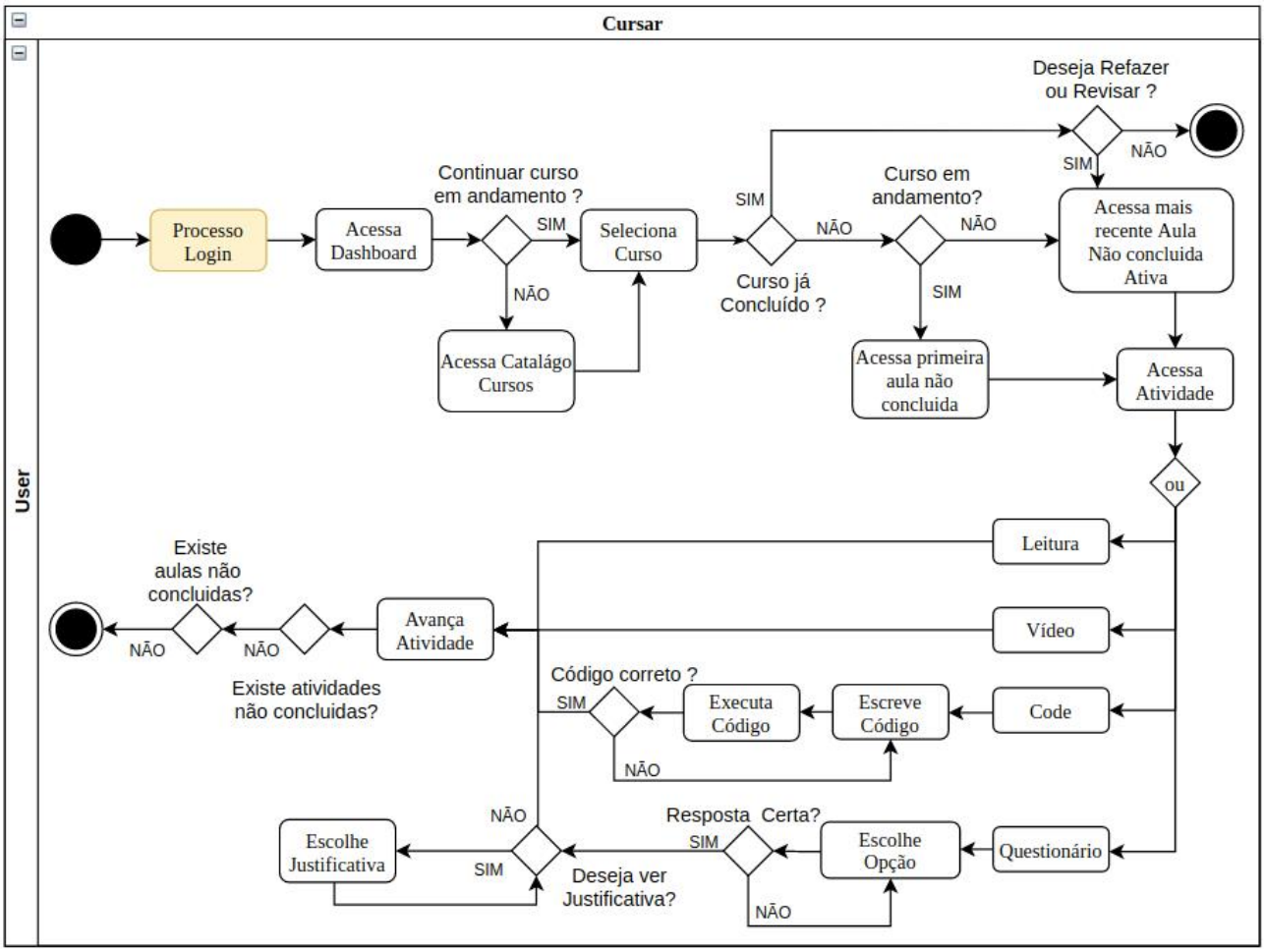
Fonte: Elaborado pelo autor.

5.1.1.2 Processo Cursar

Nesta subseção será apresentado o processo de Cursar um curso. Sendo assim, como apresentado na figura 79, o processo Cursar consiste em, após o usuário do tipo estudante estar logado, ao selecionar um curso novo ou em andamento, acessar a aula mais recente não concluída. Ao selecionar a aula, esta pode ser uma dentre os 4 tipos: leitura, vídeo, questionário ou código. Caso a atividade seja de leitura ou vídeo, basta o usuário, após terminar de assistir o vídeo ou de ler o texto, avançar para a próxima atividade. Se a atividade for do tipo código, após o usuário resolver o problema proposto

com um código válido pelo compilador, poderá avançar para a próxima atividade. Por último, se a atividade for do tipo questionário, o usuário seleciona uma opção que acha que é a certa e envia a resposta ao apertar o botão "Responder". Após o sistema confirmar se está certo ou errado, o usuário tem a opção de verificar a justificativa de cada opção ou avançar para a próxima atividade. O aluno faz esse ciclo de atividades enquanto existirem aulas e atividades não concluídas.

Figura 79 – Diagrama de atividade do processo cursar um curso no sistema.



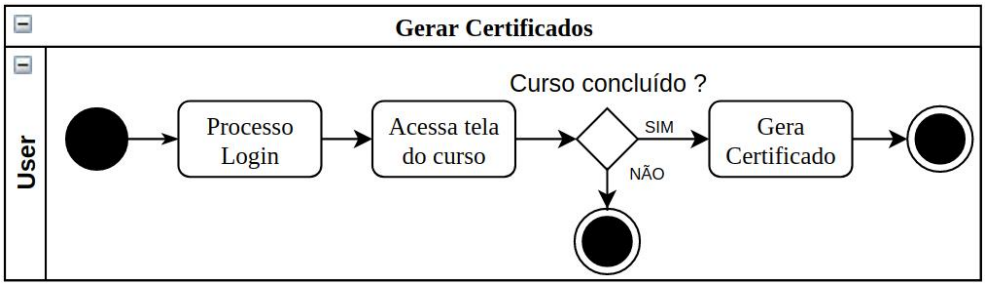
Fonte: Elaborado pelo autor.

5.1.1.3 Processos Gerar, Validar e Baixar Certificados

Nesta subseção serão apresentados os processos de Gerar, Validar e Baixar Certificados. Como apresentado na figura 80, o Processo Gerar Certificados consiste em, após o usuário do tipo estudante se encontrar logado, ao acessar um Curso 100% concluído, poderá gerar um certificado pelo botão "Gerar Certificado", na tela do curso. Este certificado gerado estará presente na área de certificados dentro do perfil do usuário.

Como apresentado na figura 81, o processo de validar certificado consiste em uma tela pública na qual o usuário, ao informar o código de validação de um certificado, terá

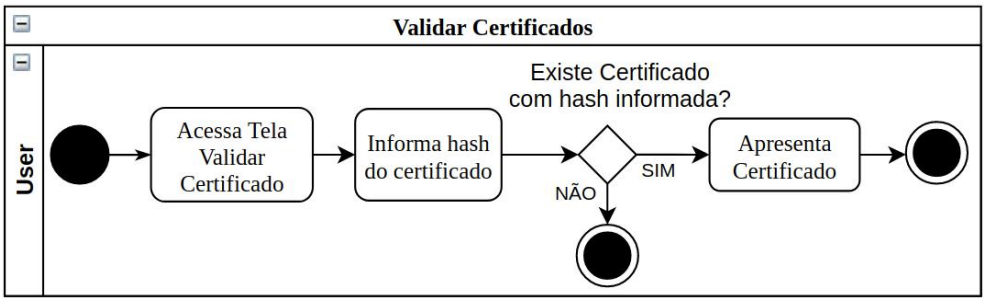
Figura 80 – Diagrama de atividade do processo de gerar certificado de um curso concluído por um estudante.



Fonte: Elaborado pelo autor.

acesso à validade e aos dados referentes ao certificado para confirmação. Caso não seja válido, apresentar-se-á que não existe certificado com o código de validação informado.

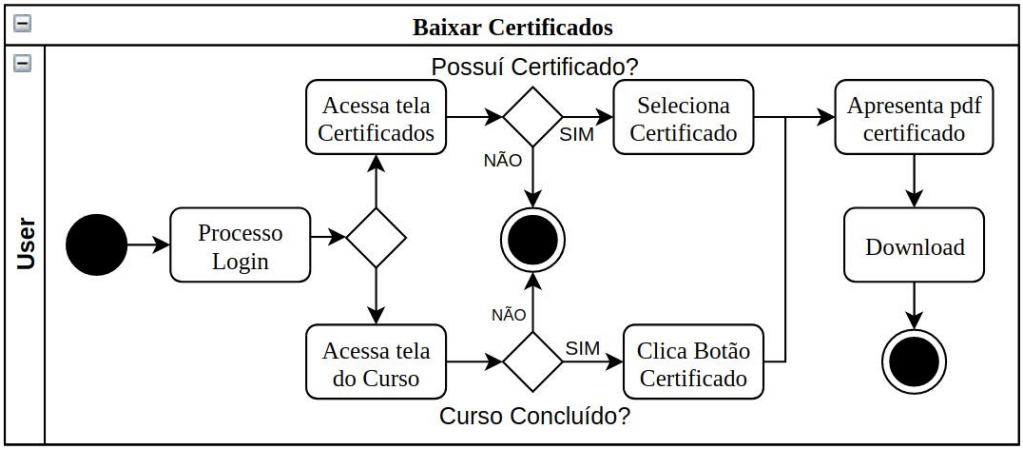
Figura 81 – Diagrama de atividade do processo de validar certificado .



Fonte: Elaborado pelo autor.

O processo de baixar certificado, como apresentado na figura 82, consiste em o usuário estudante, após logado, conseguir baixar o certificado via tela Certificados, pelo perfil de Usuário ou pela tela do Curso relacionado ao Certificado, considerando que o curso tenha sido concluído pelo estudante previamente.

Figura 82 – Diagrama de atividade do processo de fazer download de um certificado.

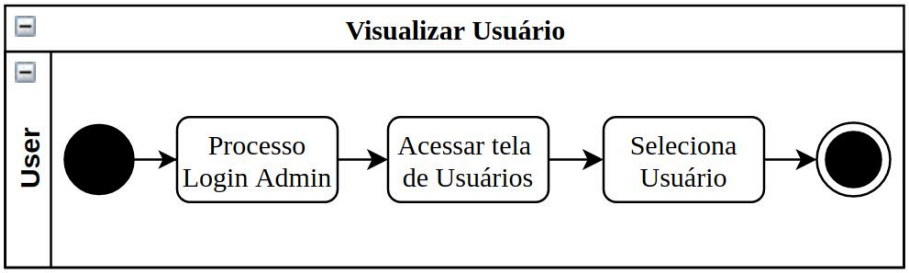


Fonte: Elaborado pelo autor.

5.1.1.4 Gerênciar Usuários

Nesta subseção serão apresentados os processos de Listar, Visualizar, Editar e Deletar usuários não administradores. O processo de listar usuários consiste em, após um usuário administrador estar logado no sistema, acessar a tela de usuários. Nessa tela estará todos os usuários do sistema listados, com as informações em uma tabela, segmentando os seguintes dados: id, tipo, nome, e-mail, *user-url*, cidade, uf, data de criação e data da última atualização. O processo de visualizar os dados de um usuário estende do processo de listar usuários. Após acessado a tela de listagem, acessando um dos usuários, clicando no nome, irá abrir a tela de informações completa do usuário, finalizando o processo de visualizar usuário. Tal processo de visualizar usuários é apresentado na figura 83.

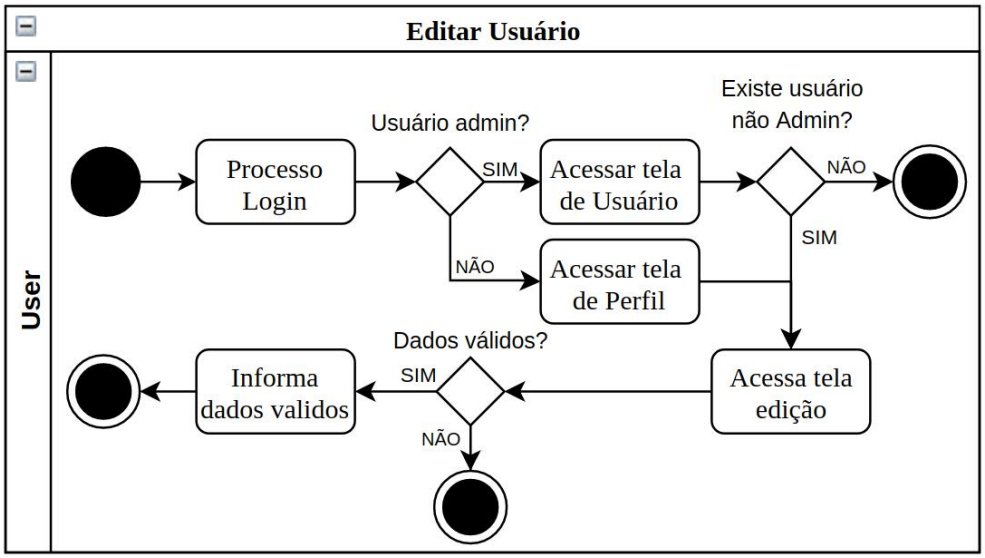
Figura 83 – Diagrama de atividade do processo de visualização de usuários.



Fonte: Elaborado pelo autor.

Como apresentado na figura 84, o processo Editar usuários consiste em após o usuário logar-se no sistema, caso ele seja um Administrador, ao acessar a tela Usuários, poderá selecionar qual usuário deseja editar, informar dados válidos, como e-mail não repetido e, posteriormente salvar a alteração. Caso o usuário seja de outro tipo, diferente de administrador, irá acessar a tela de perfil e editar o perfil com dados válidos.

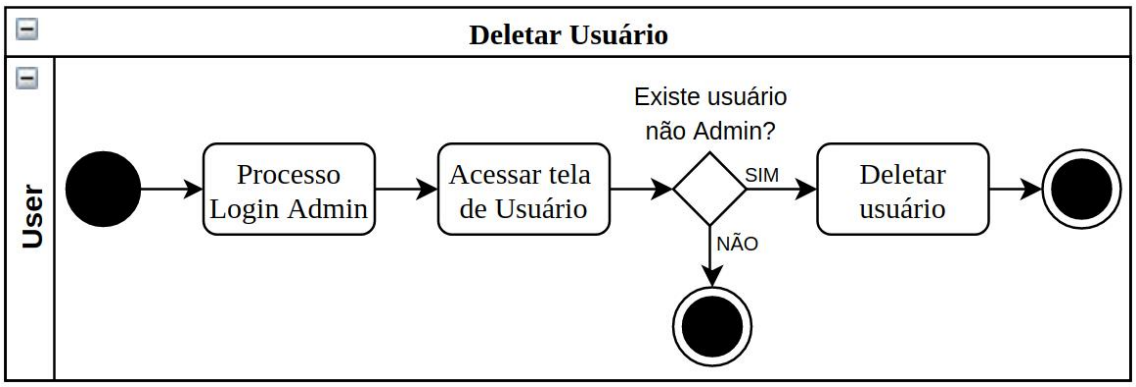
Figura 84 – Diagrama de atividade do processo de edição de usuários.



Fonte: Elaborado pelo autor.

Como apresentado na figura 85, o processo Deletar usuários consiste em, após o usuário do tipo administrador logar, acessar a tela Usuários, selecionar e confirmar qual usuário deseja deletar.

Figura 85 – Diagrama de atividade do processo de deleção de usuários.



Fonte: Elaborado pelo autor.

5.1.1.5 Cadastrar, Editar e Remover Cursos

Nesta subseção, são apresentados os processos de Cadastrar, Editar e Deletar Cursos. Os processos Deletar e Cadastrar cursos são exclusivos de usuários administradores. O processo de Editar curso é exclusivo para administradores e instrutores.

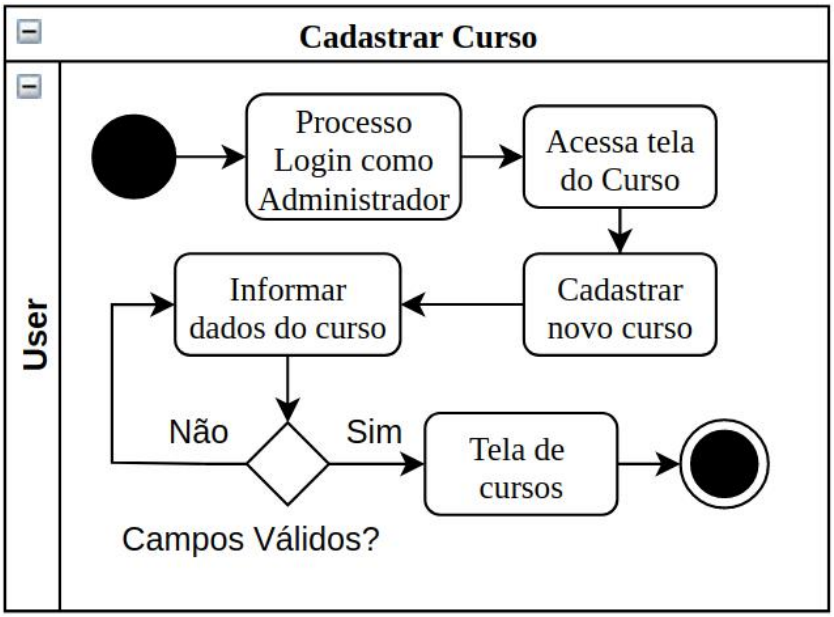
Cadastrar Curso

O processo de cadastrar um curso, como apresentado na Figura 86, consiste em, após um administrador logar no sistema e, acessar a tela de catálogo de cursos, acessar tela de cadastro de curso via botão “Novo Curso”. Após isso são informados os seguintes dados: nome, identificador, descrição, duração em minutos, nível, se já estará ativo para ser acessado por estudantes, e quais instrutores estão relacionados. Após informar estes dados e clicar no botão “Cadastrar”, o sistema irá validar se o instrutor existe e se já não existe o identificador informado. Passando nas validações, o usuário é enviado para a tela de cursos, finalizando o processo.

Editar Curso

O processo de editar um curso existente, como ilustrado na Figura 87, consiste em após um administrador ou instrutor responsável pelo curso logar no sistema e, ao acessar o curso via tela de cursos, acessar tela de edição de dados do curso via botão “Editar Curso”, ilustrado por um ícone de lápis. Posteriormente, os seguintes dados são informados: nome, identificador, descrição, duração em minutos, nível, se este já estará ativo para ser acessado por estudantes, e quais instrutores estão relacionados. Após informar estes dados e clicar no botão “Atualizar”, o sistema irá validar se o instrutor existe e se caso tenha

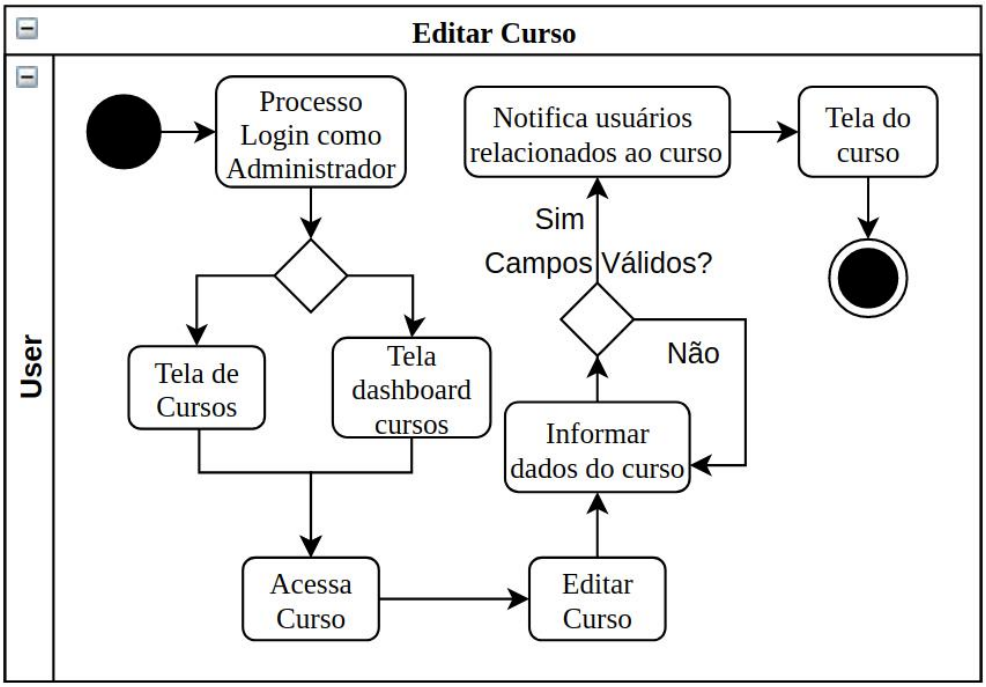
Figura 86 – Diagrama de atividade do processo de cadastrar cursos.



Fonte: Elaborado pelo autor.

sido informado um novo identificado, este novo já não exista no sistema. Passando nas validações, o usuário é enviado para a tela do curso, finalizando o processo.

Figura 87 – Diagrama de atividade do processo de editar cursos.

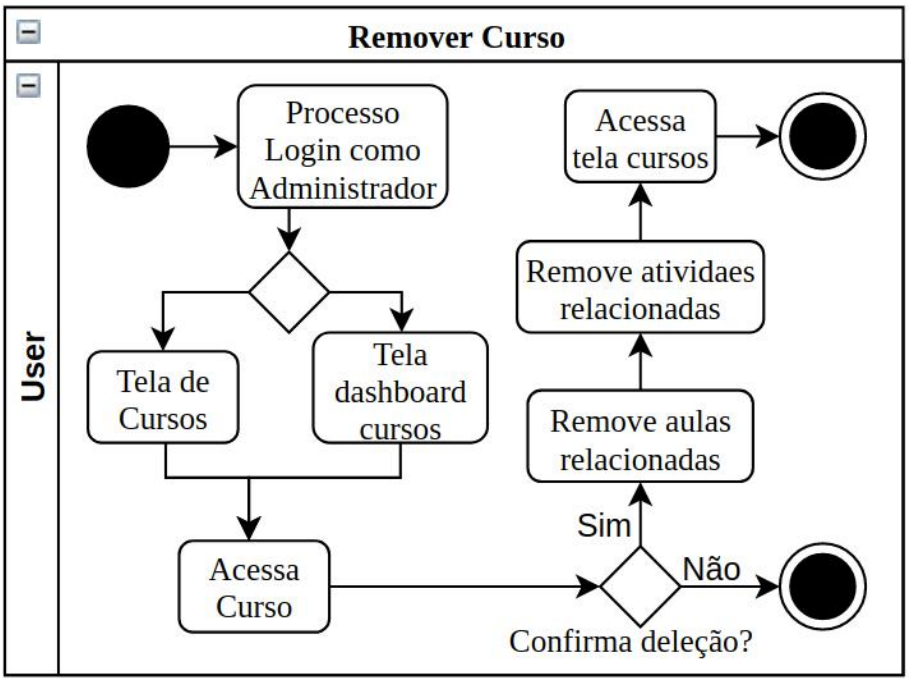


Fonte: Elaborado pelo autor.

Remover Curso

O processo de remover um curso existente, como apresentado na Figura 88, consiste, em após um administrador logar no sistema e, ao acessar o curso via tela de cursos, clicar no botão "Deletar Curso", ilustrado por um ícone de lixo. Após isso, será apresentado, por uma janela flutuante, uma mensagem de confirmação informando que o curso como as aulas e atividades relacionadas serão deletados. Caso seja confirmado, o curso, as aulas e suas atividades são deletados e, então, o usuário é enviado para a tela de cursos, onde será listado todos os cursos menos o curso removido, finalizando o processo. Caso não seja confirmado, o usuário permanece na tela de curso e o processo é finalizado.

Figura 88 – Diagrama de atividade do processo de remover cursos.



Fonte: Elaborado pelo autor.

5.1.1.6 Cadastrar, Editar e Remover Aulas

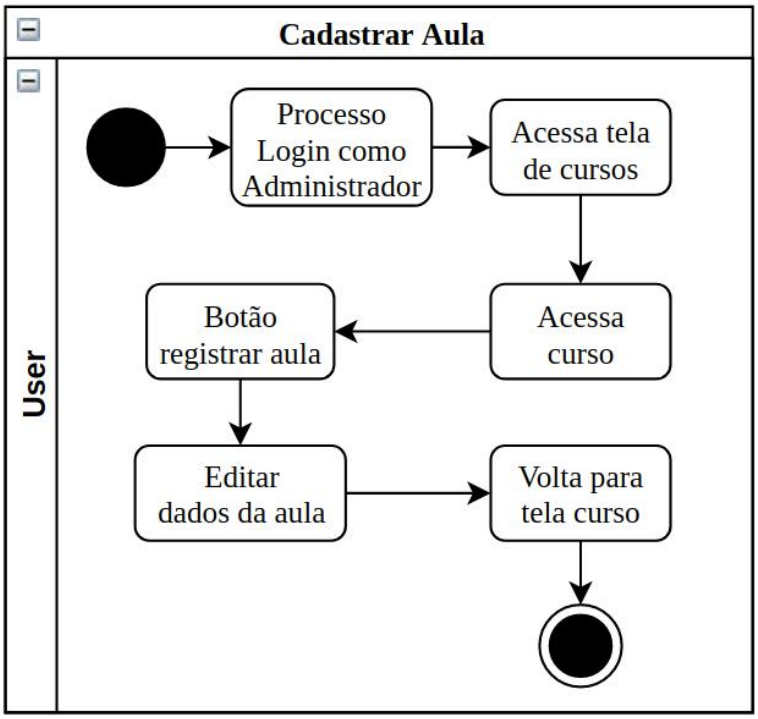
Nesta subseção, são apresentados os processos de Cadastrar, Editar e Deletar Aulas. Todos os processos são exclusivos de usuários administradores e instrutores.

Cadastrar Aulas

O processo de cadastrar uma aula, apresentado na Figura 89, consiste em, após um administrador ou instrutor responsável pelo curso logar no sistema, e acessar um curso, via tela de catálogo de cursos, acessar tela de cadastro, nova aula via botão "Registrar Aula" ilustrado por um ícone com sinal de mais (+). Após isso, informa os dados: nome, duração em minutos, e se este já estará ativo para ser acessado por estudantes. Após informar estes dados e clicar no botão "Cadastrar", o sistema irá validar o tamanho dos dados informados

referentes às descrições do dicionário de dados no Apêndice B. Passando nas validações, o usuário é direcionado para a tela da aula cadastrado, finalizando o processo.

Figura 89 – Diagrama de atividade do processo de cadastrar aulas.



Fonte: Elaborado pelo autor.

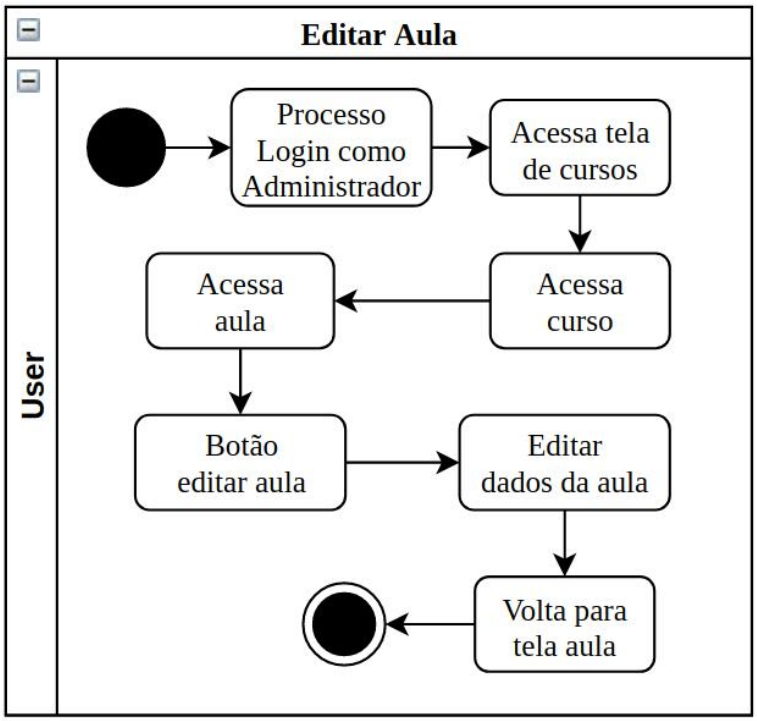
Editar Aulas

O processo de editar um curso existente, apresentado na Figura 90, consiste em, após um administrador ou instrutor responsável pelo curso logar no sistema, ao acessar a aula pela tela do curso, acessar tela de edição de dados da aula via botão “Editar Aula”, ilustrado por um ícone de lápis. Após isso, informa os dados: nome, duração em minutos, se este já estará ativo para ser acessado por estudantes. Após informar estes dados e clicar no botão “Atualizar”, o sistema irá validar o tamanho dos dados informados referentes às descrições do dicionário de dados no Apêndice B. Passando nas validações, o usuário é redirecionado para a tela da aula editada, finalizando o processo.

Remover Aulas

O processo de remover um curso, como ilustrado na Figura 91, existente consiste em, após um administrador ou instrutor responsável pelo curso logar no sistema, ao acessar a aula pela tela do curso, clicar no botão “Deletar Aula”, ilustrado por um ícone de lixo. Após isso, será apresentado, por uma janela flutuante, uma mensagem de confirmação informando que a aula com as atividades relacionadas serão deletadas. Caso seja confirmado, a aula é deletada e então o usuário e enviado para a tela do curso já descon-

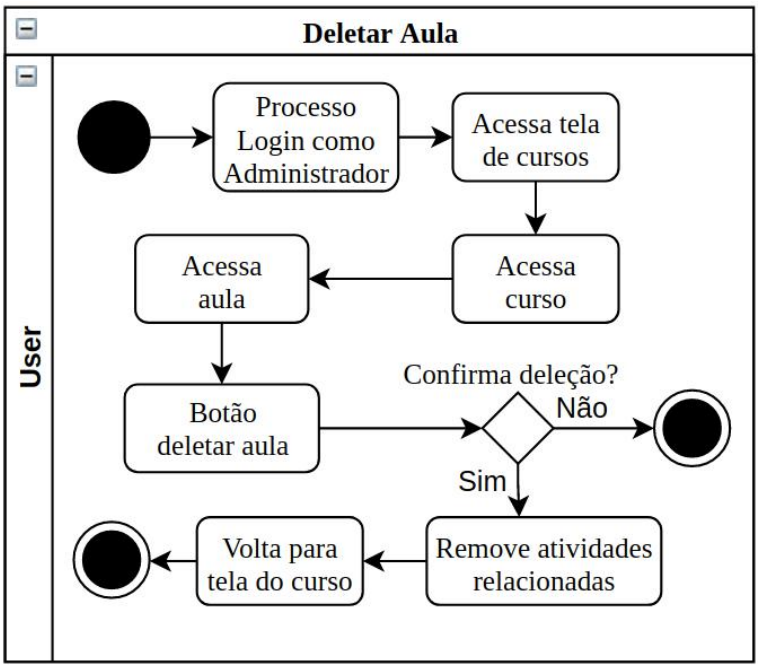
Figura 90 – Diagrama de atividade do processo de editar aulas.



Fonte: Elaborado pelo autor.

siderando a aula removida, finalizando o processo. Caso não seja confirmado, o usuário permanece na tela da aula e o processo é finalizado.

Figura 91 – Diagrama de atividade do processo de remover aulas.



Fonte: Elaborado pelo autor.

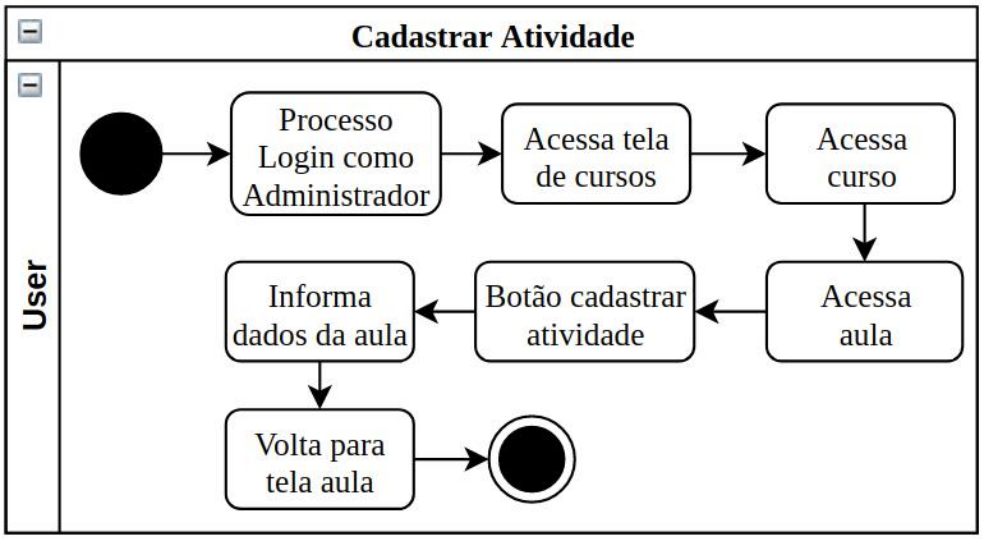
5.1.1.7 Cadastrar, Editar e Remover Atividades

Nesta subseção são apresentados os processos de Cadastrar, Editar e Deletar Atividades. Todos os processos são exclusivos de usuários administradores.

Cadastrar Atividades

O processo de cadastrar uma atividade, apresentado na Figura 92, consiste em, após um administrador ou instrutor responsável pelo curso logar no sistema, acessar uma aula pela tela do curso. Após acessar a tela da aula, acessar a tela de cadastro de nova atividade, via botão "Registrar Atividade", ilustrado por um ícone do sinal de mais (+). Em seguida, informa os dados: nome, se este já estará ativo para ser acessado por estudantes, e qual o tipo da atividade. De acordo com o tipo da atividade, será modificado os campos de inserção de dados. Para leitura, deve informar somente o texto. Para vídeo, deve informar o texto e o url do vídeo. Para múltipla escolha, deverá informar o texto da questão, quais as opções com suas justificativas e qual opção é a correta. Após informar estes dados e clicar no botão "Cadastrar", o sistema irá validar o tamanho dos dados informados referentes às descrições do dicionário de dados no Apêndice B. Para os casos de múltipla escolha, é validado se somente existe uma opção certa. Passando nas validações, o usuário é enviado para a tela da atividade cadastrada, finalizando o processo.

Figura 92 – Diagrama de atividade do processo de cadastrar atividades.



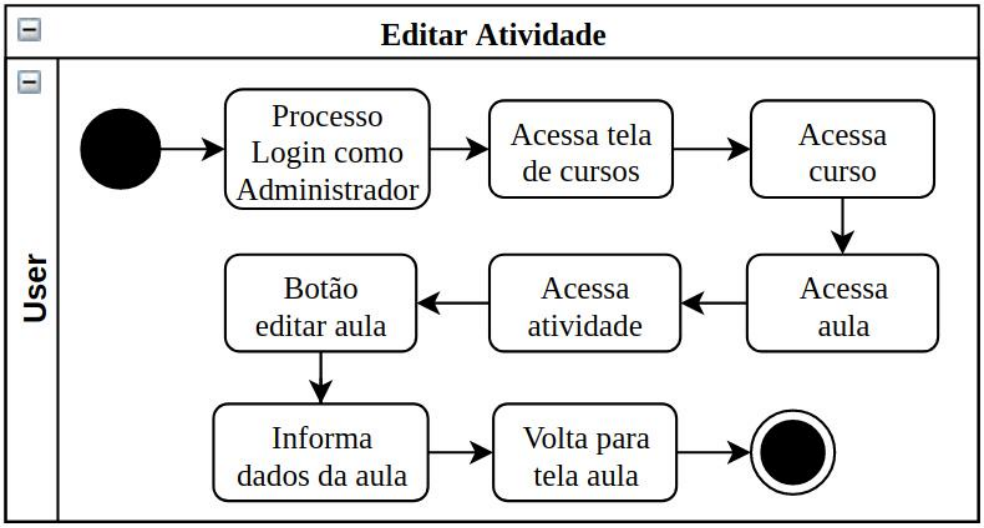
Fonte: Elaborado pelo autor.

Editar Atividades

O processo de editar uma atividade existente, apresentado na Figura 93, consiste em, após um administrador ou instrutor responsável pelo curso logar no sistema, ao acessar a atividade pela tela da aula, acessar tela de edição de dados da atividade via botão "Editar Atividade" ilustrado por um ícone de lápis. Após isso informa os dados: nome, se este já

estará ativo para ser acessado por estudantes, qual o tipo da atividade. De acordo com o tipo da atividade, serão modificado os campos de inserção de dados. Para leitura, deve informar somente o texto. Para vídeo, deve informar o texto e o url do vídeo. Para múltipla escolha, deverá informar o texto da questão, quais as opções com suas justificativas, e é a opção de resposta correta. Após informar estes dados e clicar no botão "Atualizar", o sistema irá validar o tamanho dos dados informados referentes às descrições do dicionário de dados no Apêndice B, além de validar para casos de múltipla escolha, se somente existe uma opção certa. Passando nas validações, o usuário é redirecionado para a tela da atividade editada, finalizando o processo.

Figura 93 – Diagrama de atividade do processo de editar atividades.



Fonte: Elaborado pelo autor.

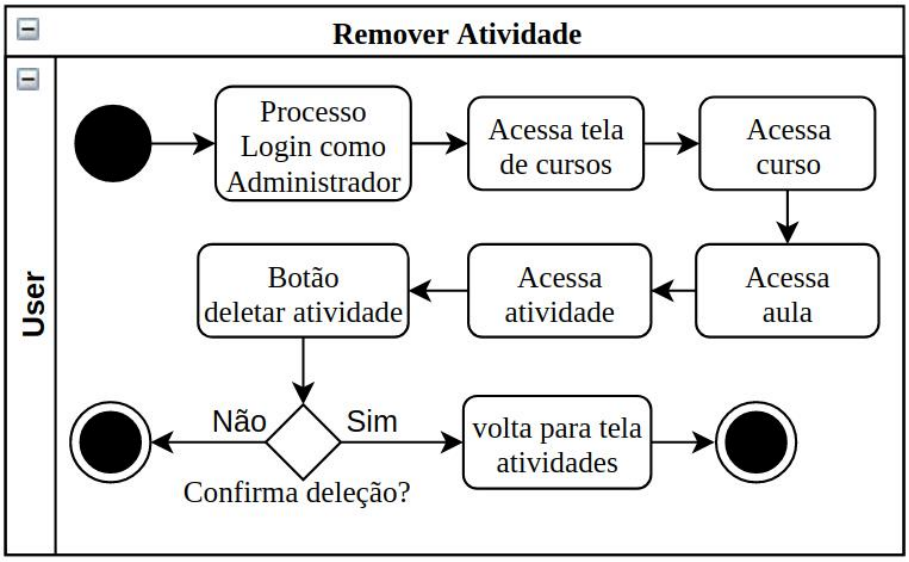
Remover Atividades

O fluxo de remover uma atividade existente consiste em após um administrador ou instrutor responsável pelo curso logar no sistema, ao acessar a atividade pela tela da aula, clicar no botão "Deletar Atividade", ilustrado por um ícone de lixo. Posteriormente, será apresentado, por uma janela flutuante, uma mensagem de confirmação, informando que a atividade será deletada. Caso seja confirmado, a atividade é deletada e então o usuário e enviado para a tela da aula já desconsiderando a atividade removida, finalizando o processo. Caso não seja confirmado, o usuário permanece na tela da atividade e o processo é finalizado. Tal fluxo é apresentado na Figura 94.

5.1.2 Modelo de Dados

Para modelagem de dados, foi utilizado o padrão DDD, sendo sua modelagem feita para facilitar a implementação de complexas regras/processos de negócio nomeadas como domínios. Os domínios levantados foram: Usuário, Curso, Matricula, Licenciatura,

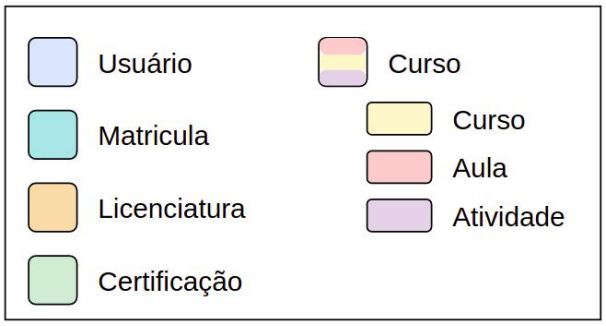
Figura 94 – Diagrama de atividade do processo de remover atividades.



Fonte: Elaborado pelo autor.

Certificação. Para auxiliar na representação dos domínios foram utilizadas cores distintas, como apresentado na figura 95.

Figura 95 – Legenda da definição dos domínios e subdomínios.

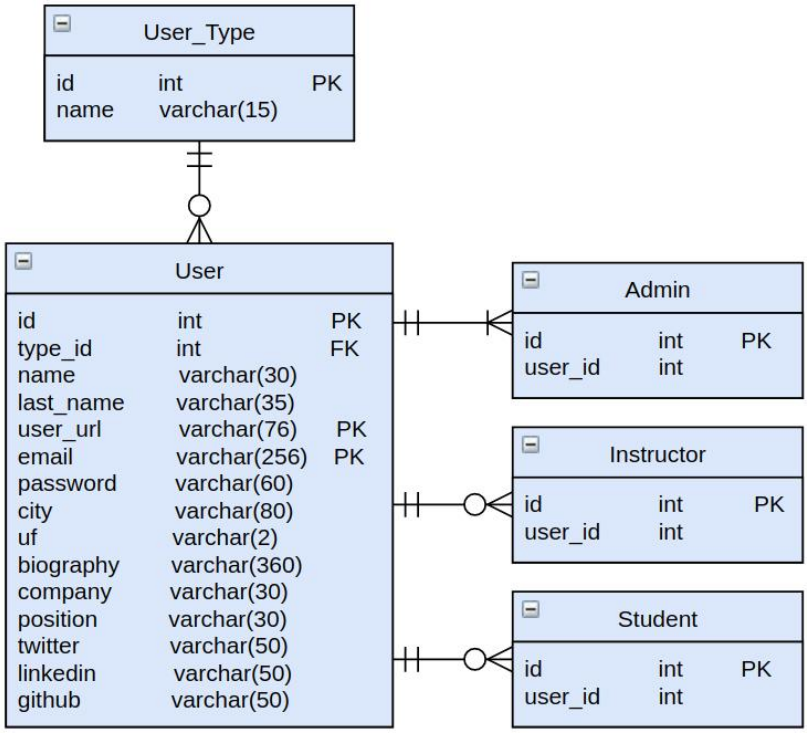


Fonte: Elaborado pelo autor.

5.1.2.1 Domínio Usuário

Com apresentado na figura 96, o domínio de **Usuário** é caracterizado por uma tabela *User*, e sua especificação é definida por meio da relação com a tabela *User_type*. Um usuário pode ser Estudante, Professor, Monitor ou Administrador. Dentro desse domínio um *User* deve estar relacionado com um *User_type* e um mesmo *User_Type*, pode estar relacionado a nenhum ou vários Users. Um Instrutor, ou *Student* ou *Admin* deve estar relacionado a um, e somente um *User*, como *User* pode estar relacionado a zero ou vários *Instructor* e *Student* em exceção a *Admin*, onde *User* pode estar relacionado a um ou vários *Admin*.

Figura 96 – Modelo de dados do domínio Usuário.

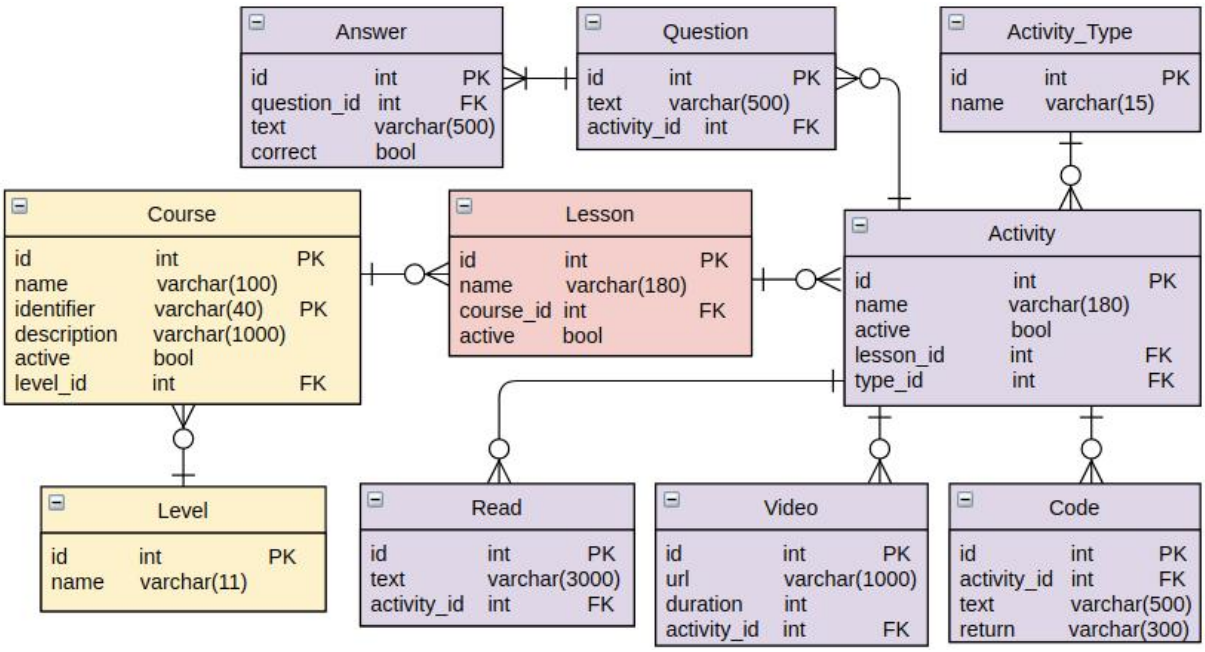


Fonte: Elaborado pelo autor.

5.1.2.2 Domínio Curso

Como ilustrado na figura 97, o domínio de Curso, é definido em 3 subdomínios: *Course*, *Lesson* e *Activity*.

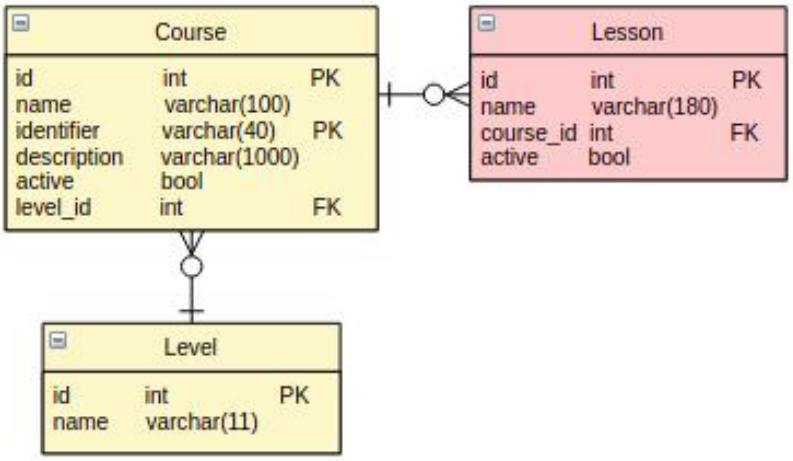
Figura 97 – Modelo de dados do domínio Curso.



Fonte: Elaborado pelo autor.

O subdomínio de **Course** exibido na figura 98, é caracterizado por uma tabela *Course* onde no domínio de Curso tem relação com o módulo *Level* e *Lesson*. A tabela *Level* especifica a dificuldade do curso entre Iniciante, Moderado e Avançado. A tabela *Lesson* representa a relação do curso possuir aulas. Sobre a relação entre curso e *level*, um curso pode possuir somente um *level* (dificuldade) e um *level* pode estar relacionado a nenhum ou vários cursos. Um Curso pode possuir zero ou várias *Lesson*.

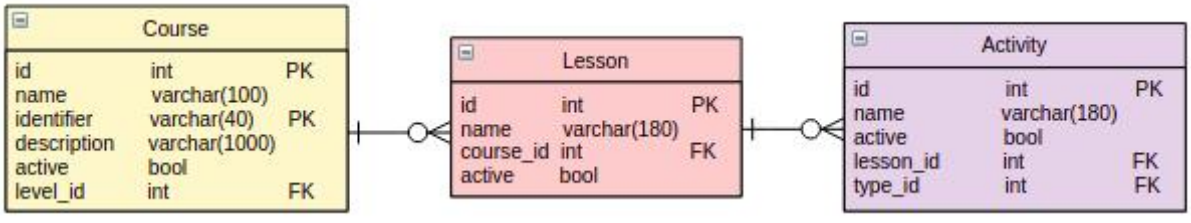
Figura 98 – Modelo de dados do subdomínio Curso.



Fonte: Elaborado pelo autor.

O subdomínio de **Lesson**, apresentado na figura 99 é caracterizado por uma tabela *Lesson*, a qual representa a aula, que tem como objetivo representar um conjunto de atividades. Este ainda tem relações com os módulos *Course* (relacionamento de pertencimento a um curso) e *Activity* (representar a relação de uma aula possuir atividades). Uma *Lesson* deve estar relacionada a um curso e pode possuir zero ou maior atividades.

Figura 99 – Modelo de dados para subdomínio Aula.

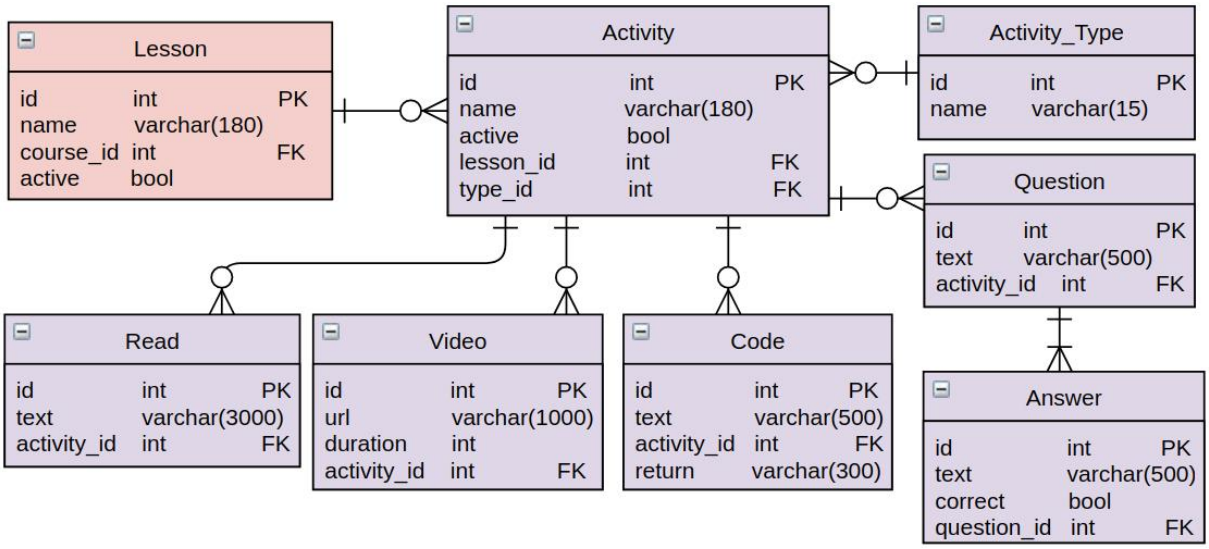


Fonte: Elaborado pelo autor.

O subdomínio de **Activity**, apresentado na figura 100, é composto pelas tabelas *Activity*, *Activity_Type*, *Read*, *Video*, *Question*, *Answer* e *Code*. *Activity* representa a própria atividade, *Activity_type* representa qual o tipo de atividade dentre (*Read*, *Video*, *Question* e *Code*), *Read* representa ao conteúdo de atividades do tipo Leitura, *Video* representa ao conteúdo de atividades do tipo vídeo, *Question* representa ao conteúdo da atividade do tipo Questionário e *Code* representa ao conteúdo da atividade do tipo de

Código. *Answer* representa as opções de escolhas de resposta para atividades do tipo questionário. Uma *Activity* deve estar relacionada a somente uma Aula.

Figura 100 – Modelo de dados para subdomínio Atividade.

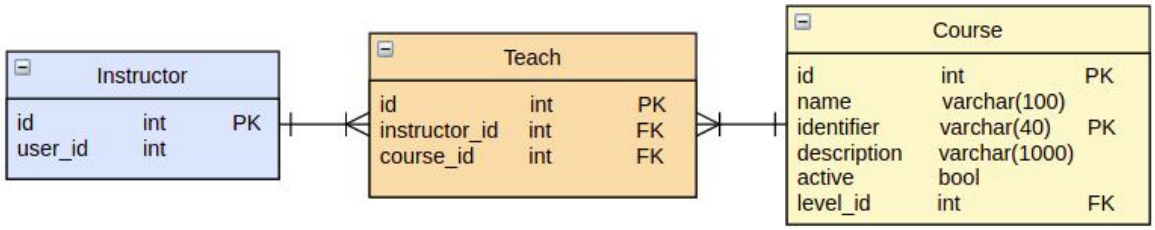


Fonte: Elaborado pelo autor.

5.1.2.3 Domínio Licenciatura

Como apresentado na figura 101, o domínio licenciatura é caracterizado por uma tabela *Teach*, a qual representa a relação de um instrutor ao lecionar um curso. Por esse domínio, será feito o controle de qual curso um instrutor poderá: editar, adicionar conteúdo (aulas e atividades) e ter acessos aos dados de respostas dos estudantes cursantes deste curso.

Figura 101 – Modelo de dados para Domínio Licenciatura.



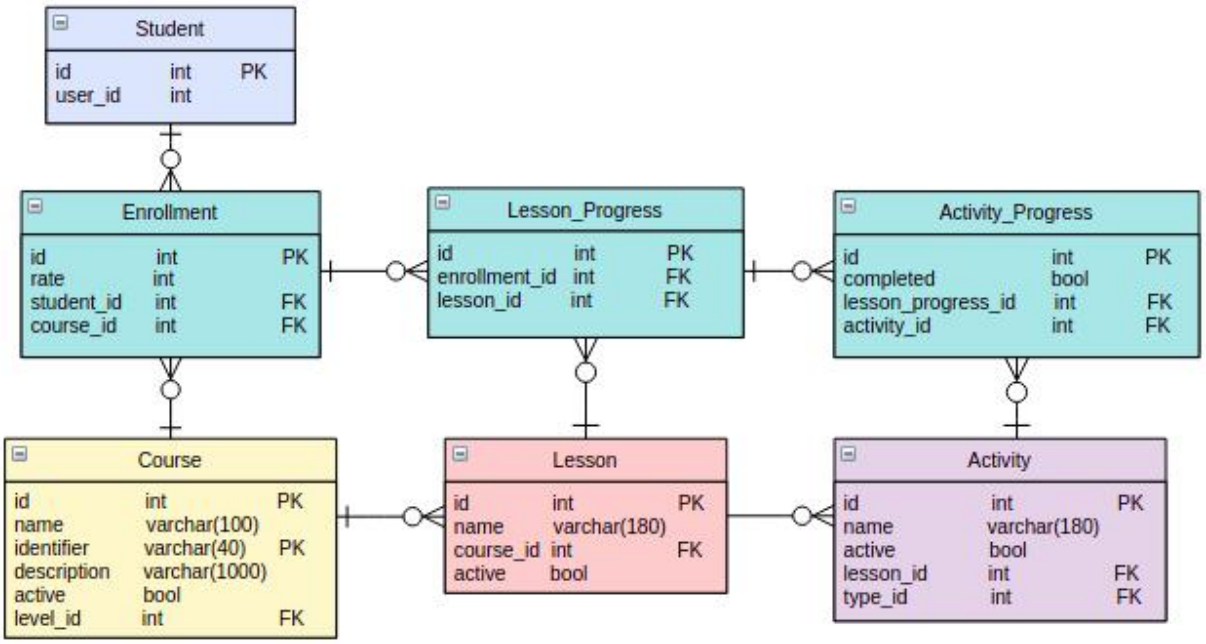
Fonte: Elaborado pelo autor.

5.1.2.4 Domínio Matrícula

Como apresentado na figura 102, o domínio Matrícula é caracterizado pelas tabelas *Enrollment*, *Lesson_Progress* e *Activity_Progress*. Este Domínio representa a relação

entre estudante e curso. Por esse domínio, são constituídas as matriculas e progressos de estudantes com um curso. Essa subdivisão foi pensada para possibilitar controle de quais atividades foram feitas pelo estudante e, dessa forma, o progresso da aula, assim, apresentando o progresso em porcentagem do curso. O controle dessa porcentagem de progresso do curso é utilizado para realizar futuramente a certificação do estudante com o curso.

Figura 102 – Modelo de dados para Domínio Matrícula.



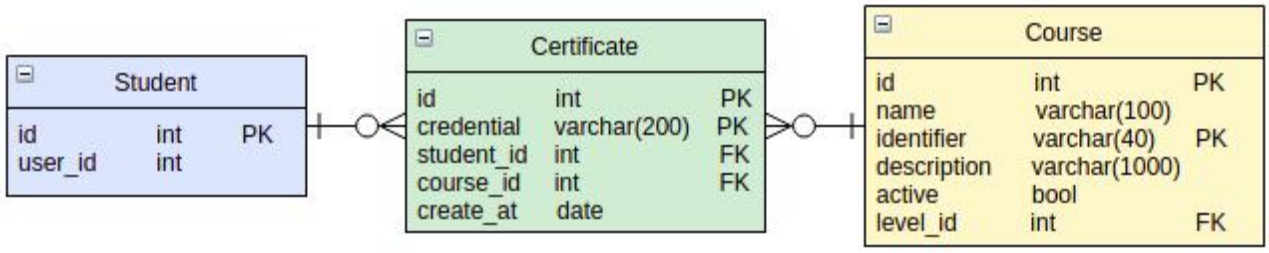
Fonte: Elaborado pelo autor.

Um *Enrollment* deve estar relacionado a somente um *Student* e um *Course*, enquanto *Student* e *Course* podem possuir zero ou mais matrículas. *Lesson_Progress* deve estar relacionado a somente um *Enrollment* e uma *Lesson*, enquanto uma *Lesson* pode estar relacionada a zero ou vários *Lesson_Progress*. *Activity_Progress* deve estar relacionada a uma *Activity*, enquanto uma *Activity* pode estar relacionada a zero ou mais *Activity_Progress*.

5.1.2.5 Dominio Certificação

Como apresentado na figura 103, o domínio Certificação é caracterizado pela tabela *Certificate* e tem como objetivo representar os certificados conquistados pelos estudantes ao atingir um progresso de 100% de um curso. Este domínio tem uma relação entre *Student* e o *Course* onde *Certificate* deve estar relacionado a um *Studente* e um *Couse*, enquanto *Student* e *Course* podem estar relacionados a zero ou mais *certificate*.

Figura 103 – Modelo de dados para Domínio Certificação.



Fonte: Elaborado pelo autor.

5.1.3 Dicionário de Dados

Para padronizar a nomenclatura e glossário dentro do projeto, foi levantado um Dicionário de Dados. Dessa forma, facilitando a integração e manutenção do projeto. Esse levantamento do dicionário de dados foi separado em duas partes:

- **Entidades** - descreve sobre a entidade e as relações que a entidade tem no banco de dados;
- **Atributos** - descreve em detalhes sobre cada atributo de uma entidade.

Entidades

Para definição das relações das entidades, foi construída uma tabela como a 12, referente a tabela do bando de dados *Course*, onde tem relações com as tabelas de dados *Level*, *Tech*, *Enrollment*, *Lesson* e *Certificate*. Estas descrevem com mais detalhes, a tabela, nome de relacionamento, relacionamento e descrição. Para não poluir a leitura do texto o restante encontra-se no Apêndice B.

Tabela 12 – Dicionário de dados da tabela Curso.

Tabela Course - Entidade			
Tabela	**Nome de Relacionamento**	**Relacionamento**	**Descrição**
Course	Has	Level	Tabela para cadastro de Cursos
	Taught	Tech	
	Has	Enrollment	
	Has	Lesson	
	Issue	Certificate	

Fonte: Elaborado pelo autor.

Atributos

Para definição mais específica dos atributos definidos no dicionário de dados, foram detalhados na tabela 13, a qual descreve com mais detalhes, nome da tabela no banco de

dados, nome da coluna, tipo, validações e descrição. O restante se encontra no Apêndice B.

Tabela 13 – Atributos tabela de dados Curso.

Tabela Course - Atributos			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
ID	Inteiro	PK, not null	Numero identificador do curso
name	Caracter	maxlenght: 120, not null	Nome do curso
identifier	Caracter	maxlenght: 40, unique, not null	Nome identificador do curso
description	Caracter	maxlenght: 1000, not null	Descrição do curso
active	Boleano	default: false, not null	Campo de identificação se o curso esta ativo ou não
level_id	Inteiro	not null	Código de identificação da dificuldade do curso (Básico, Moderado, Avançado)
create_at	Data	not null	Data de criação do curso
updated_at	Data	not null	Data da última atualização do curso

Fonte: Elaborado pelo autor.

5.1.4 Modelos IFML

Nesta seção, são apresentados os modelos IFML, descrevendo o fluxo de telas de acordo com as funcionalidades levantadas. A subseção 2.5.2 apresenta as notações do IFML, assim, podendo auxiliar na leitura dos modelos a seguir.

5.1.4.1 Home

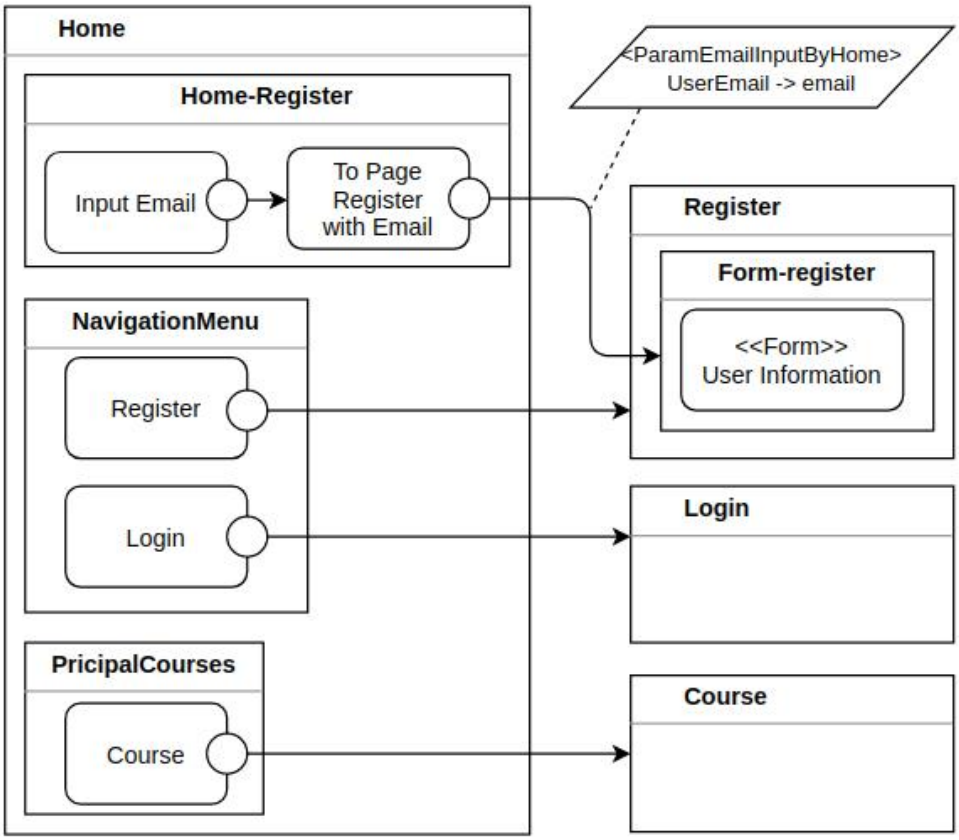
Nesta subseção, é apresentado o fluxo da tela *Home*. Tal fluxo é apresentado de forma modular pela Figura 104. Dentro da tela *home*, se encontram-se 3 áreas principais:

- pré-registro;
- menu de navegação;
- principais cursos.

Na área de pré-registro, o usuário informa um e-mail. Com o e-mail informado, após o clique no botão “Registrar”, o usuário é então redirecionado para a tela de formulário de registro, já com o e-mail preenchido, levando como parâmetro o e-mail informado no *input* da área de pré-registro da tela *home*.

Na área do Menu de Navegação, o usuário pode clicar no botão de registro ou de *login*. Caso opte por clicar em registro, será é redirecionado para a tela de registro,

Figura 104 – Modelo IFML da tela *home*.



Fonte: Elaborado pelo autor.

com os todos os *inputs* vazios para serem preenchidos. Caso opte por clicar em *login*, será redirecionado para a tela de *Login*.

Na área de cursos principais, o usuário terá uma lista de cartões de cursos. Após clicar em um desses cursos, o usuário é redirecionado para a tela do curso.

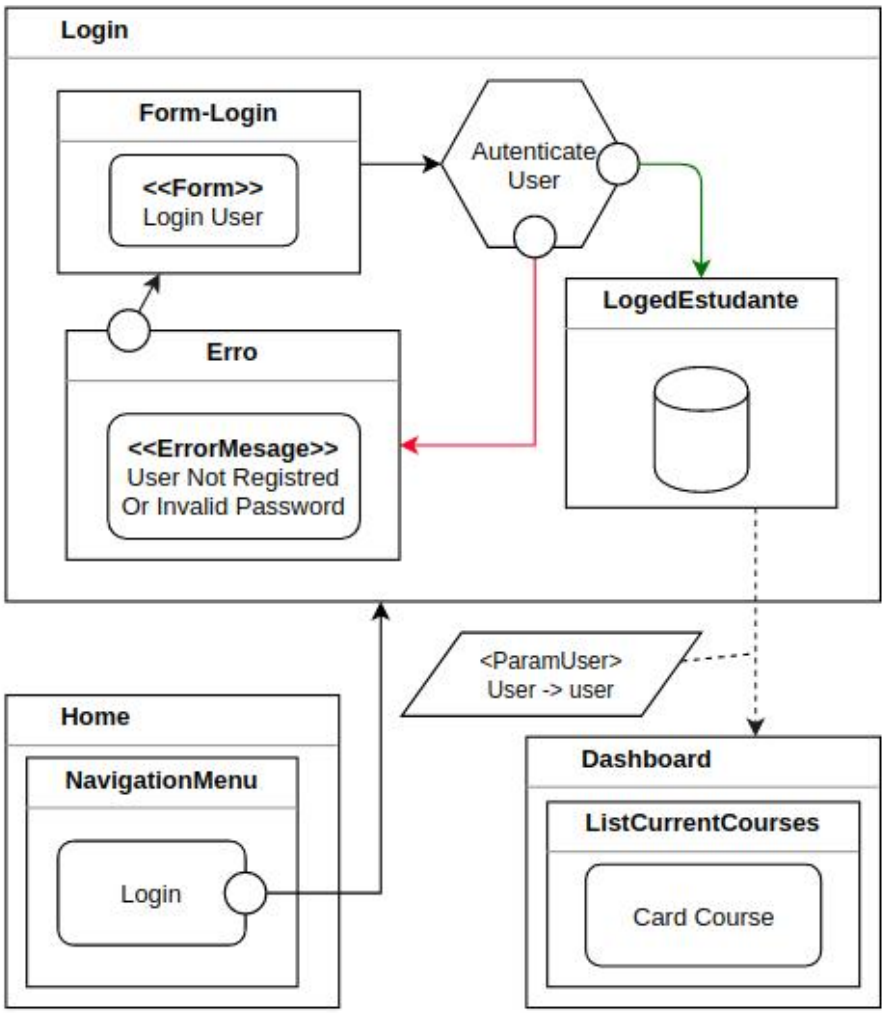
5.1.4.2 Login

Nesta subseção, é apresentado o fluxo da tela de *Login*. Tal fluxo é apresentado de forma modular pela Figura 105. Dentro da tela, o usuário terá um formulário no qual informará o e-mail e senha. Após informados esses dados, o sistema valida a autenticação.

Caso aconteça algum erro na autenticação, é informado um *pop-up* ou tela flutuante para o usuário e, após clicar no botão de confirmação, o usuário é direcionado novamente para a tela com o formulário de *login*. Caso a autenticação aconteça com sucesso, é registrado um evento de que o usuário logou no sistema e este é direcionado o usuário para a tela de *Dashboard*.

Ao ser redirecionado da tela de login para a tela de *dashboard*, os dados do usuário são enviados como parâmetro. Isso é necessário para informar os dados do usuário na tela, como, nome, foto e cursos em andamento caso tenha.

Figura 105 – Modelo IFML da tela de login.



Fonte: Elaborado pelo autor.

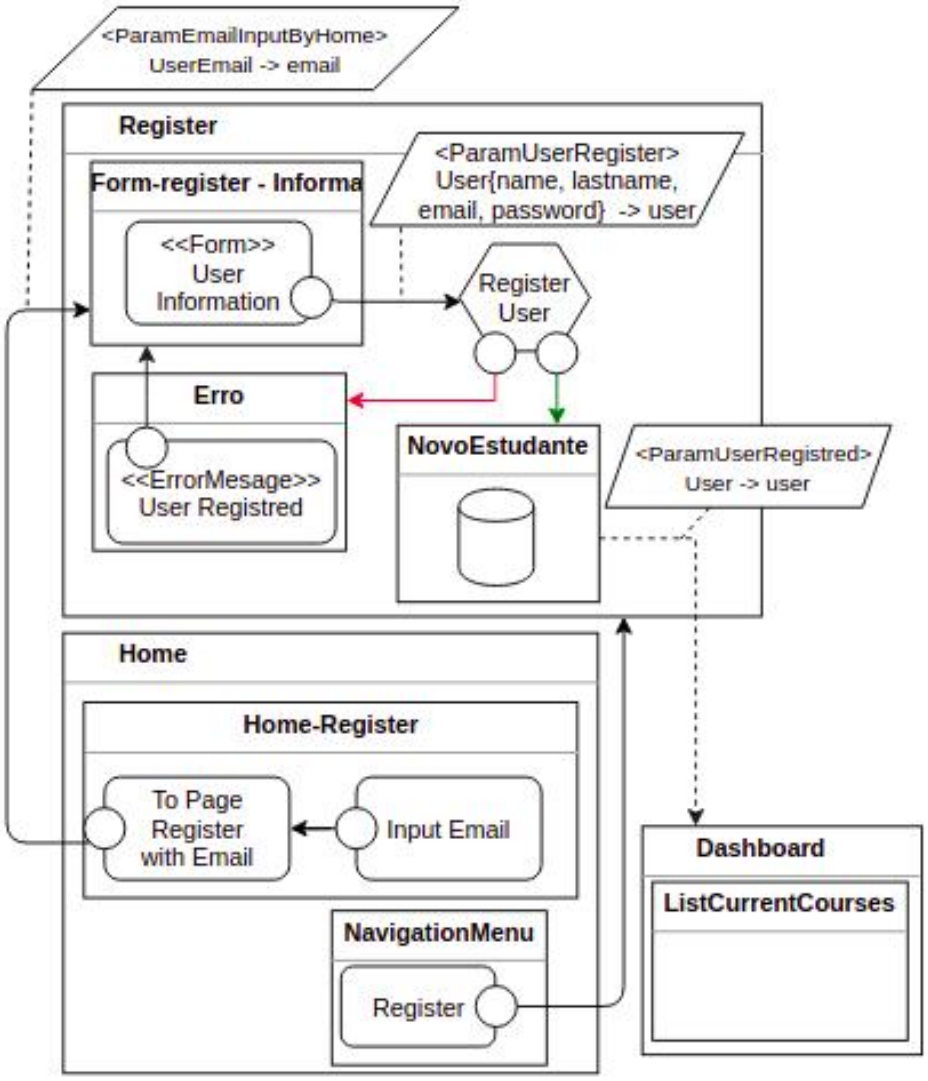
5.1.4.3 Register

Nesta subseção, é apresentado o fluxo da tela de registro. Tal fluxo é apresentado de forma modular pela Figura 106. Na tela, o usuário terá um formulário para se registrar, no qual é informado o e-mail, senha, nome, sobrenome e tipo de usuário. Após informados os dados, o sistema realiza algumas validações, como se já existe algum e-mail cadastrado igual ao informado.

Após realizar as validações, caso seja barrado em alguma validação, é apresentado um *pop-up* ou tela flutuante para o usuário, informando qual o problema na tentativa de registro. Após confirmar a mensagem, clicando no botão de confirmação, o usuário é direcionado novamente para a tela com o formulário de registro.

Caso o cadastro do usuário passe nas validações, é registrado no banco de dados o novo usuário e, então, redirecionado para a tela de *dashboard*, onde será apresentada uma lista de cursos.

Figura 106 – Modelo IFML da tela de registro.



Fonte: Elaborado pelo autor.

5.1.4.4 Dashboard-Geral

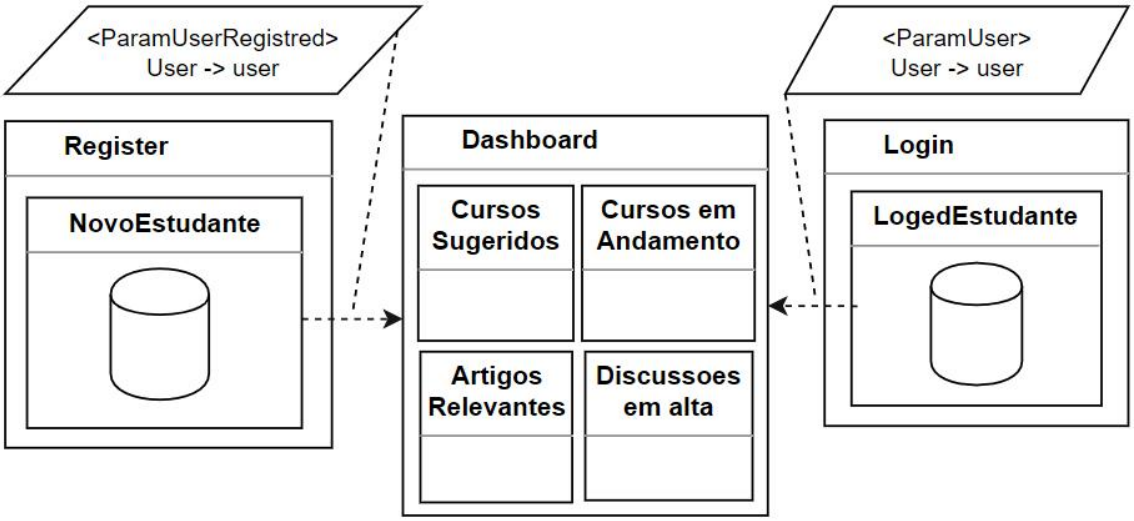
Nesta subseção, é apresentado o fluxo geral da tela dashboard. Tal fluxo é apresentado de forma modular pela Figura 107. Para o usuário chegar à tela de *dashboard*, precisa primeiro finalizar o fluxo da tela Registrar ou *Login*.

Após o usuário realizar o registro ou login, é redirecionado para a tela de *dashboard*, levando como parâmetro os dados do usuário. Estes dados do usuário serão utilizados para apresentar algumas informações na tela, como:

- Nome do perfil na tela;
- Progresso dos cursos em andamento;
- Cursos recomendados em relação aos já matriculados;

- Discussões dos cursos que está vinculado;
- Artigos dos cursos matriculados.

Figura 107 – Modelo IFML para chegar á tela de dashboard.



Fonte: Elaborado pelo autor.

O fluxo da apresentação das seções especificas, as quais utilizam os dados do usuário, é apresentado abaixo na próxima seção.

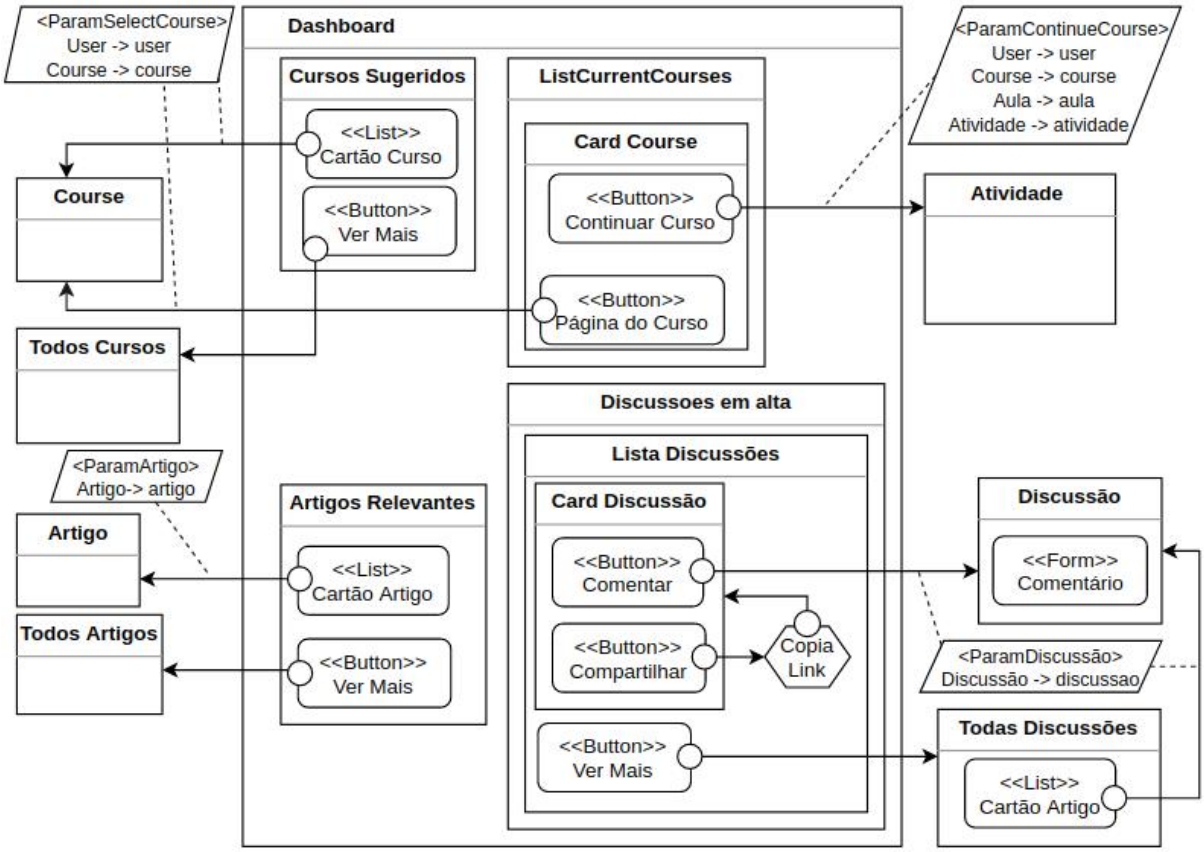
5.1.4.5 Dashboard-Conteúdo

Nesta subseção, é apresentado o fluxo da tela *dashboard*. Tal fluxo é apresentado de forma modular pela Figura 108. Continuando o fluxo apresentado na subseção anterior 5.1.4.4, o usuário, ao chegar na tela de *dashboard*, encontra quatro áreas principais:

- Cursos em andamento;
- Cursos sugeridos;
- Discussões em alta;
- Artigos relevantes.

Na área de **Cursos em andamento**, é apresentada uma lista de cartões. Nesses cartões, são apresentados o progresso do usuário referente ao curso e dois botões, **Continuar Curso** e **Página do Curso**. O botão "Continuar Curso" tem a funcionalidade de redirecionar o usuário estudante para a próxima atividade, posterior à última finalizada, levando os parâmetros: usuário, curso, aula e atividade. Tais parâmetros são importantes para encontrar a matrícula referente ao estudante com o curso, e auxiliar a plataforma expor qual atividade deve ser apresentada, sabendo também a aula e o curso referente. O

Figura 108 – Modelo IFML da tela dashboard.



Fonte: Elaborado pelo autor.

botão **Página do Curso**, por sua vez, é responsável por encaminhar o usuário estudante para a tela principal do curso, levando como parâmetros os dados do usuário e do curso referente. Os dados do usuário são, assim, encaminhados para apresentar os progressos já adquiridos, auxiliando na apresentação dinâmica da tela. Tais detalhes da tela do curso serão apresentados na seção 5.1.4.9.

A área de **Cursos sugeridos** apresenta uma lista de cartões de cursos sem progresso, que o sistema irá sugerir, de acordo com os cursos em andamento, e um botão de **Ver mais**. Essa sugestão de cursos é realizada apresentando cursos relacionados com os já em andamento, por exemplo, o curso *Programação 2* é apresentado, ao considerar um estudante cursando *Programação 1*. No momento em que o usuário seleciona um curso desta lista, é encaminhado para a tela do curso levando consigo como parâmetros os dados identificadores do curso e usuário. No botão **Ver mais** o usuário é encaminhado para a tela *Todos Cursos*.

Na área **Artigos Relevantes**, é apresentada uma lista de cartões de artigos e um botão de **Ver mais**. Essa sugestão de artigos é realizada apresentando artigos relacionados com os cursos em andamento e concluídos pelo usuário, como, um artigo sobre a JVM para usuários que estão realizando ou tenham concluído o curso de *Programação Java* (ou outra

linguagem que utilize a JVM). No momento em que o usuário seleciona um artigo desta lista, clicando no cartão, é encaminhado para a tela de apresentação do artigo referente, levando consigo os identificadores do artigo referente. No botão **Ver mais** o usuário é encaminhado para a tela *Todos Artigos*.

A área **Discussões em Alta**, por sua vez, apresenta uma lista de cartões de discussões e um botão **Ver mais**. Tais discussões também são sugeridas de acordo com o cursos em andamento ou já concluídos, bem como discussões nas quais o usuário tenha comentado. Nesse cartão de uma discussão, é apresentado um botão para comentar e outro **Compartilhar**. Caso o usuário estudante clique no botão **Comentar**, será direcionado para a tela da discussão, levando consigo os dados identificadores da discussão aberta. Ainda dentro da área do cartão da discussão, caso o usuário clique no botão **Compartilhar**, será copiado o link para acesso da tela específica da discussão na área de compartilhamento do dispositivo. Fora da área do cartão de discussão, dentro da área geral das discussões em alta, quando o usuário clicar no botão **Ver mais**, será direcionado para a tela de todas as discussões, as quais serão apresentadas em forma de lista de cartões, de forma cronológica, considerando a última criada ou comentada. Nesta área, caso o usuário clique em um cartão de discussão, será direcionado para a tela da discussão em aberto.

5.1.4.6 Catálogo de Todos os Cursos

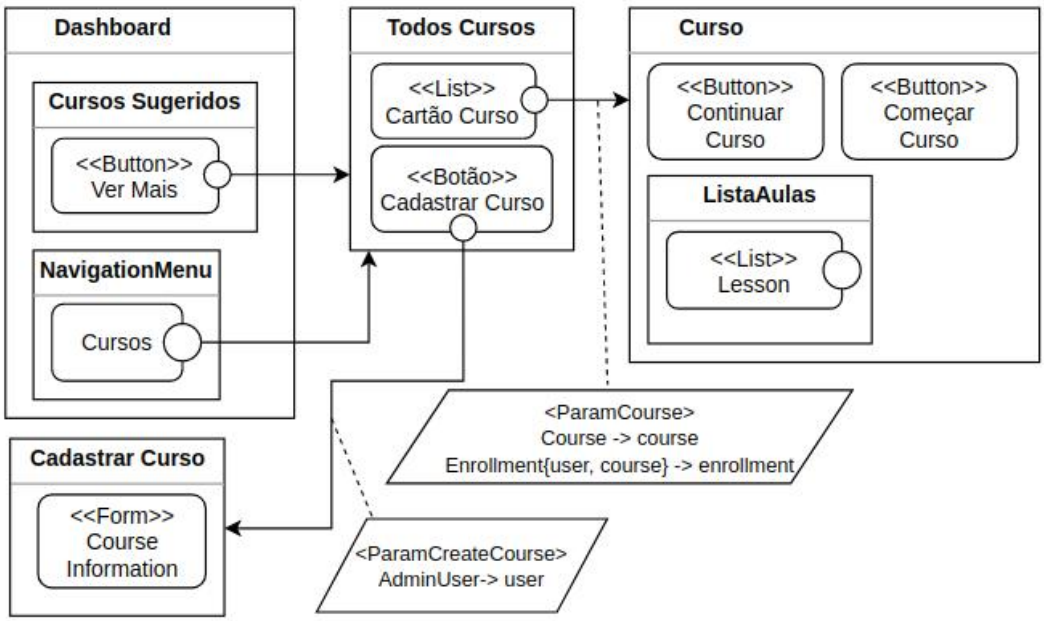
Nesta subseção, é apresentado o fluxo da tela *Catálogo de todos os cursos*. Tal fluxo é apresentado de forma modular pela Figura 109. Nesta tela, são apresentados um botão **Cadastrar Curso** e uma lista de cartões de todos os cursos. O botão é acessível somente para o usuário administrador.

No momento em que o usuário administrador clica no botão **Cadastrar Curso**, ele é direcionado para a tela de Registrar Curso junto com os parâmetros de identificação do usuário administrador. Na tela de Cadastro de Curso, é apresentado um formulário para a realização do cadastro. Esta é mais detalhada na próxima seção.

Caso o usuário estudante clique em algum destes cartões de cursos apresentados, é então direcionado para a tela do Curso levando como parâmetro os dados de identificação do curso, estudante e matrícula dele com o curso, caso exista. O fluxo da tela de Curso é apresentado com mais detalhes na seção 5.1.4.9.

O usuário pode chegar de duas maneiras na tela *Todos os Cursos*, pelo botão **Ver Mais**, da área *Cursos sugeridos*, ou pelo botão **Cursos**, no menu de navegação. Ambos os pontos de partida são encontrados na tela *Dashboard*.

Figura 109 – Modelo IFML da tela de catálogo de cursos.

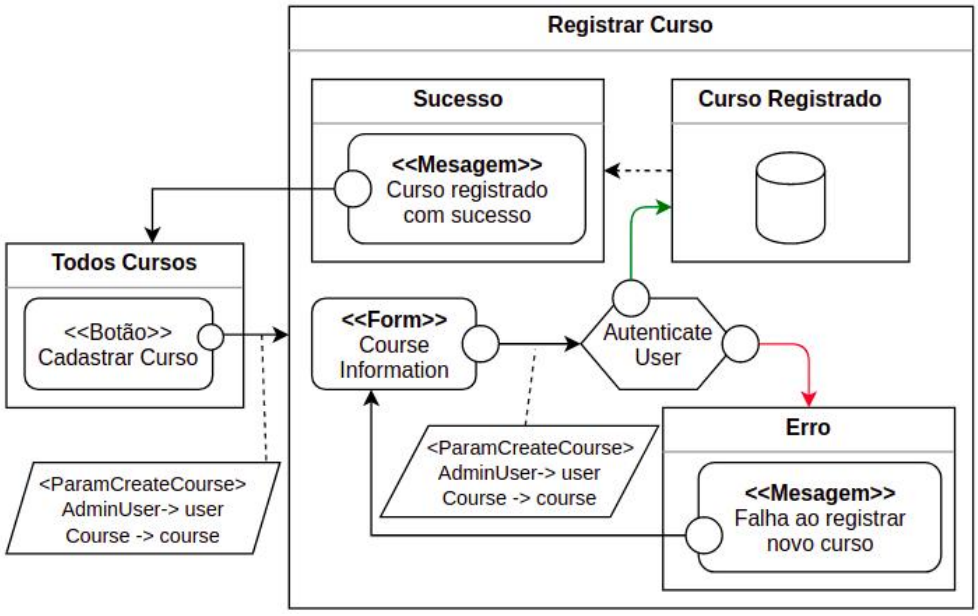


Fonte: Elaborado pelo autor.

5.1.4.7 Curso - Registrar

Nesta subseção, é apresentado o fluxo da tela *Registrar Curso*. Tal fluxo é apresentado de forma modular detalhada pela Figura 110.

Figura 110 – Modelo IFML da tela de registro de cursos.



Fonte: Elaborado pelo autor.

Esta tela é accessível somente para usuários administradores. Nela é apresentado um formulário para cadastro de um novo curso. Para o usuário administrador chegar a essa tela, é preciso clicar no botão **Cadastrar Curso**, localizado na tela *Todos os Cursos*.

A página de registrar curso consiste em um formulário de cadastro de um novo curso, o qual, quando preenchido e clicado no botão cadastrar, é enviado como parâmetro os dados preenchidos do novo curso. Após isso, é realizado a verificação de validação dos dados informados e a autenticação do usuário administrador. Nesta validação de dados, é verificado se a identificação do novo curso é igual a uma já existente, se o tamanho máximo de informações ou informações obrigatórias foram respeitados.

Após o processo de validação, caso os dados informados não estejam corretos ou validáveis, é apresentada uma tela modal, informando uma mensagem de erro ("Falha ao registrar o novo curso") e, logo em seguida, na mesma apresentação, qual o motivo encontrado na validação. Logo, o usuário é direcionado novamente para a tela de registrar curso. Caso os dados informados sejam validáveis, o curso é registrado na base de dados e, então, apresentado uma tela modal informando a mensagem de sucesso ("Curso registrado com sucesso"). Após isso, o usuário é direcionado para a tela *Todos os Cursos* onde serão apresentados todos os cursos, inclusive o novo cadastrado.

5.1.4.8 Curso - Administrador

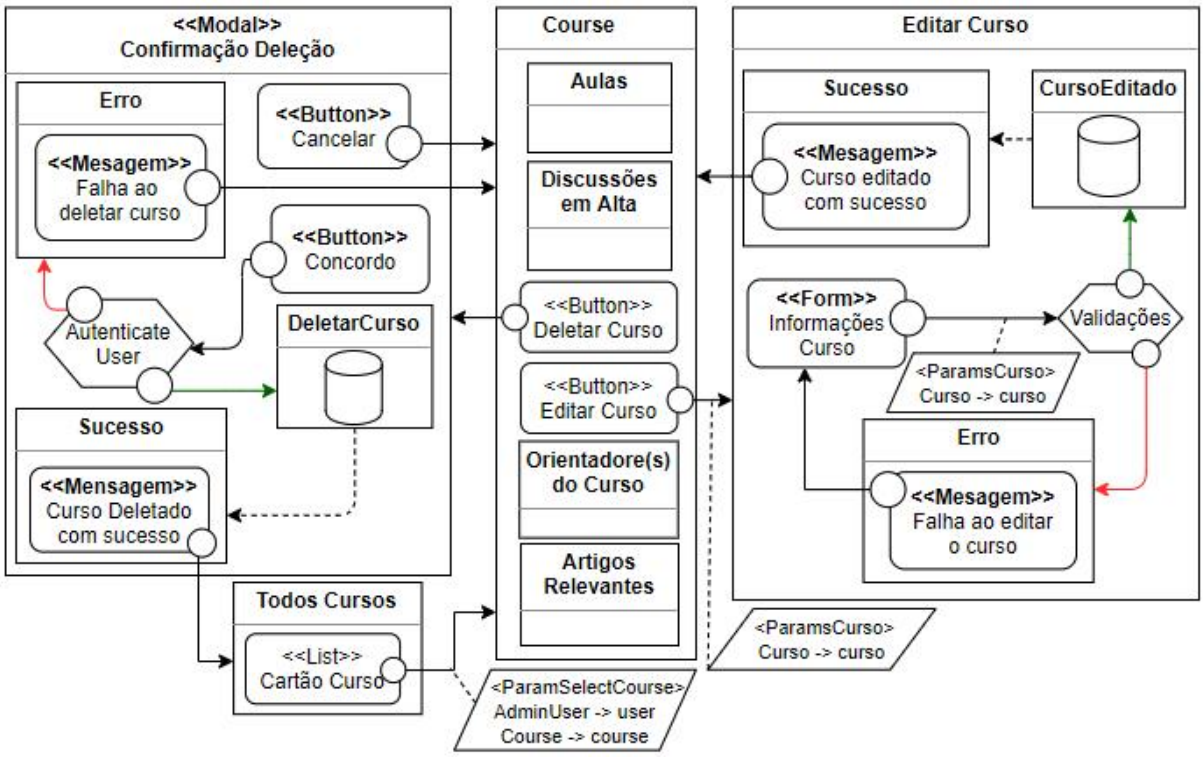
Nesta subseção, é apresentado o fluxo de edição e deleção de *Curso*. Tal fluxo é apresentado de forma modular pela Figura 111. Nesta tela, existem dois botões principais para administradores: "Editar Curso" e "Deletar Curso", é apresentado o fluxo destes de forma detalhada a seguir.

A tela de Curso também possui outros três botões para usuário não administradores ("pré-visualizar curso", "favoritar curso" e "começar curso") e 4 áreas principais (**Aulas**, **Orientador(es) do Curso**, **Discussões em Alta** e **Artigos Relevantes**), as quais são apresentadas de forma mais detalhada nas seções seguintes.

Esta tela é acessada por meio da tela *Todos os Cursos* ou da tela *Dashboard* pela listagem de *cursos em andamento* ou *cursos sugeridos*. Em ambos os acessos são encaminhados os parâmetros do usuário, para reconhecer se este é um usuário do tipo administrador, e do curso, para apresentar as informações do curso selecionado. Os botões "Editar Curso" e "Deletar Curso" são de uso exclusivo do Administrador.

Quando o usuário administrador clicar no botão **Deletar Curso**, é apresentada na tela uma janela do tipo modal, informando a mensagem: "Concorda com a deleção do curso? Todos os registros, aulas e atividades serão deletados também". Posteriormente, o administrador tem duas escolhas, concordar ou não. Caso o usuário não concorde, este clica no botão **Cancelar** e então é redirecionado para a tela do Curso. Caso concorde é verificado a autenticação do administrador, confirmando o perfil do usuário, o curso é deletado e é informado uma mensagem na tela dizendo: "Curso deletado com sucesso", e, posteriormente, este usuário é redirecionado para a tela *Todos Cursos*. Caso a autenticação do usuário falhe, é informada uma mensagem na tela dizendo: "Falha ao deletar o curso".

Figura 111 – Modelo IFML das telas de edição e deleção de cursos.



Fonte: Elaborado pelo autor.

Dessa forma, o usuário é redirecionado para a tela do Curso.

O botão **Editar Curso**, ao ser clicado pelo administrador, o encaminhará para a tela de edição de curso, na qual é apresentado um formulário com os dados atuais do curso, os quais são encaminhados via parâmetro. Após o usuário informar os novos dados, é realizado a validação destes, como identificação única do curso na base de dados, tamanhos máximo e mínimo de dados e campos obrigatórios. Após validação com sucesso, a alteração do curso é registrada na base de dados. Após isso, é apresentado em uma tela modal a mensagem de sucesso, informando: "Curso editado com sucesso", e, depois de alguns segundos, o usuário é redirecionado para a tela do curso, já com os dados alterados. Caso a validação não seja executada com sucesso, é informado na tela por uma mensagem em janela modal a mensagem: "Falha ao editar o curso", apresentando, em seguida, o motivo da falha. Logo o usuário é redirecionado para a tela de edição do curso, para informar novamente os dados no formulário.

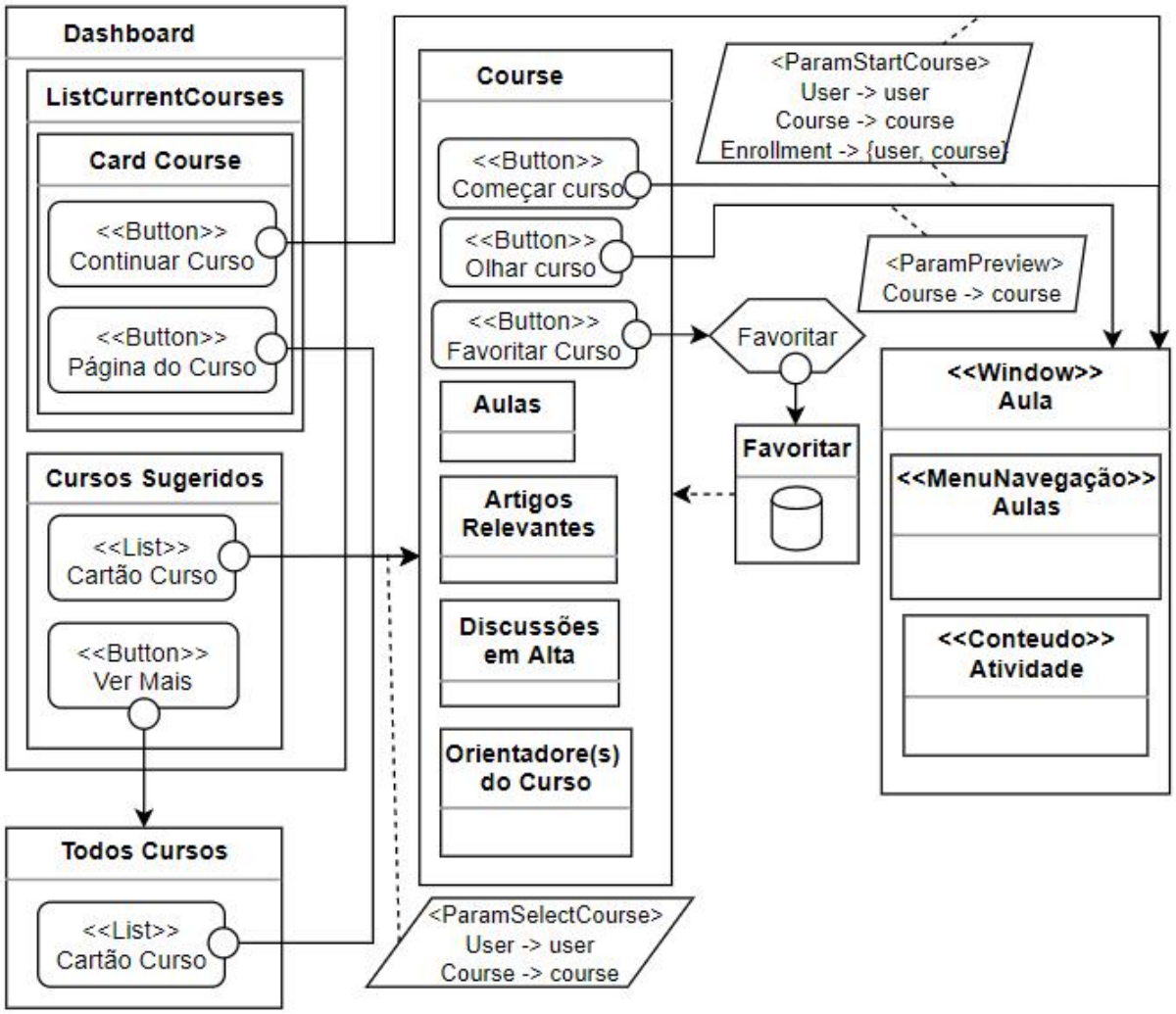
5.1.4.9 Curso - Botões Estudantes

Nesta subseção, é apresentado o fluxo dos botões: "pré-visualizar curso", "favoritar curso" e "começar curso". Tal fluxo é apresentado de forma modular pela Figura 112.

Como mencionado na seção anterior, esta tela é acessada por meio da tela *Todos os Cursos* ou na tela *Dashboard*, pela listagem de cursos em andamento ou cursos sugeridos.

Por ambos os acessos, são encaminhados os parâmetros do usuário, para saber quais dos botões serão apresentados, e do curso, para apresentar os dados detalhados do mesmo na página.

Figura 112 – Modelo IFML da tela Curso com foco nos botões pré-visualizar, favoritar e começar curso



Fonte: Elaborado pelo autor.

Já na tela do curso, caso seja um usuário não registrado na plataforma, ou não tenha efetuado uma matricula no curso, este pode optar pela opção de pré-visualizar o curso pelo botão **Olhar Curso**. O usuário será direcionado para a primeira atividade da primeira aula do curso. Sendo assim, o usuário não terá registro nenhum de progresso.

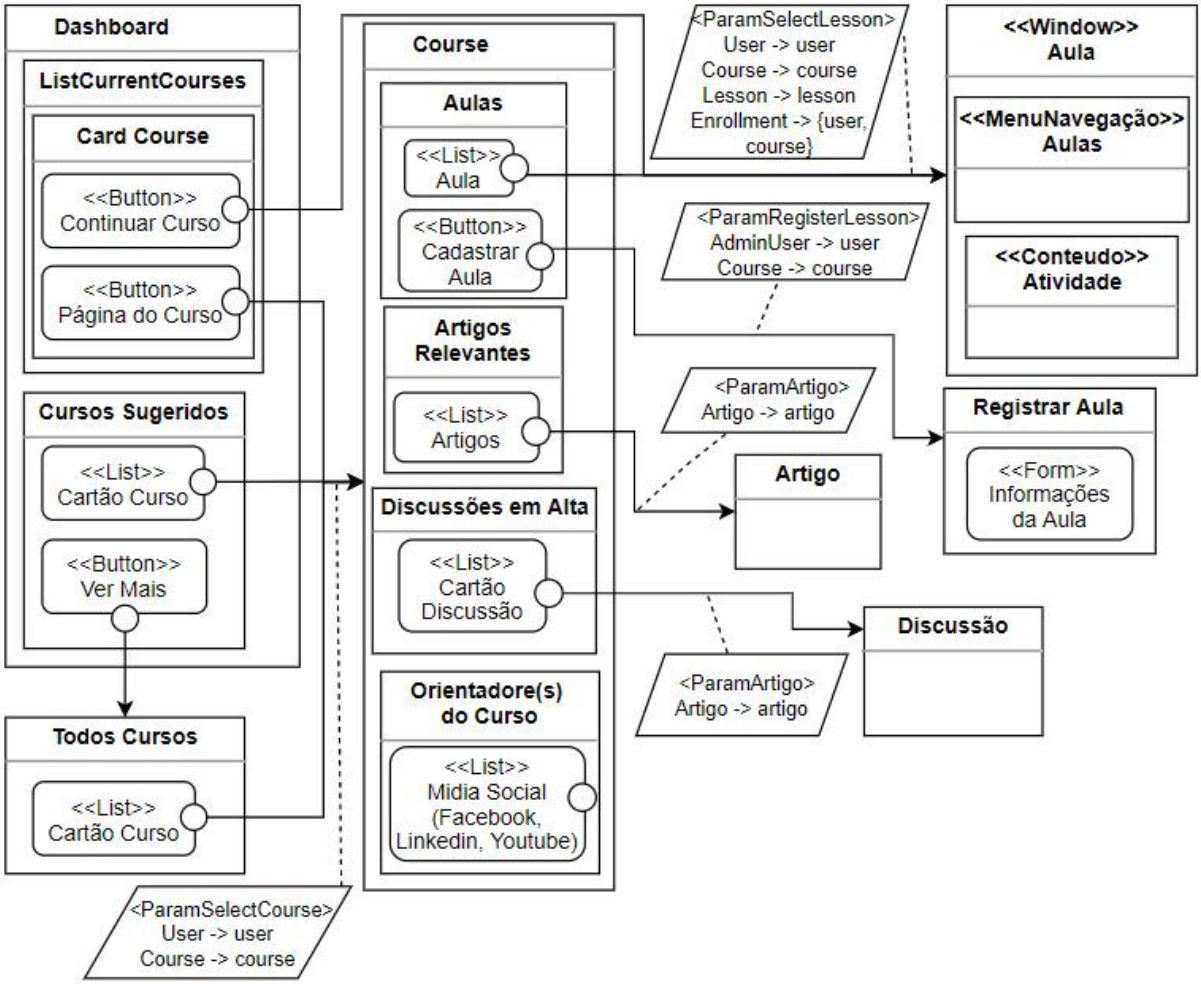
Caso o estudante esteja logado, mas ainda não matriculado no curso, ele tem duas opções em botões: **Favoritar Curso** e **Começar Curso**. No primeiro, o estudante apenas adiciona o curso à sua lista de favoritos, assim, este será apresentado na lista de cursos sugeridos na tela de *dashboard*. Neste fluxo, o usuário se mantém na tela do curso, sendo apresentado apenas uma mensagem, informando: “Curso adicionado na lista

de favoritos". No segundo botão, *começar curso*, será registrada a matrícula do estudante no curso, assim, a partir desse momento, será registrado todo o seu progresso dentro do curso. Nesse momento, o estudante é direcionado para a primeira atividade da primeira aula.

5.1.4.10 Curso - Áreas

Nesta subseção, é apresentado o fluxo das áreas principais da tela *Curso*. As áreas que serão detalhadas são: **Aulas**, **Orientador(es) do Curso**, **Discussões em Alta** e **Artigos Relevantes**. Tal fluxo é apresentado de forma modular pela Figura 113.

Figura 113 – Modelo IFML da tela de curso



Fonte: Elaborado pelo autor.

Como mencionado na seção anterior, esta tela é acessada por meio da tela *Todos os Cursos* ou na tela *Dashboard*, pela listagem de cursos em andamento ou cursos sugeridos. Em ambos acessos são encaminhados os parâmetros do usuário e do curso.

Na área de **Aulas**, é apresentada uma lista de aulas e o botão **Cadastrar Aula**, este último é exclusivo para usuários administradores. Na listagem de aulas, é apresentado o progresso resumido de aulas feitas o total de aulas, caso a tela de curso seja acessada por

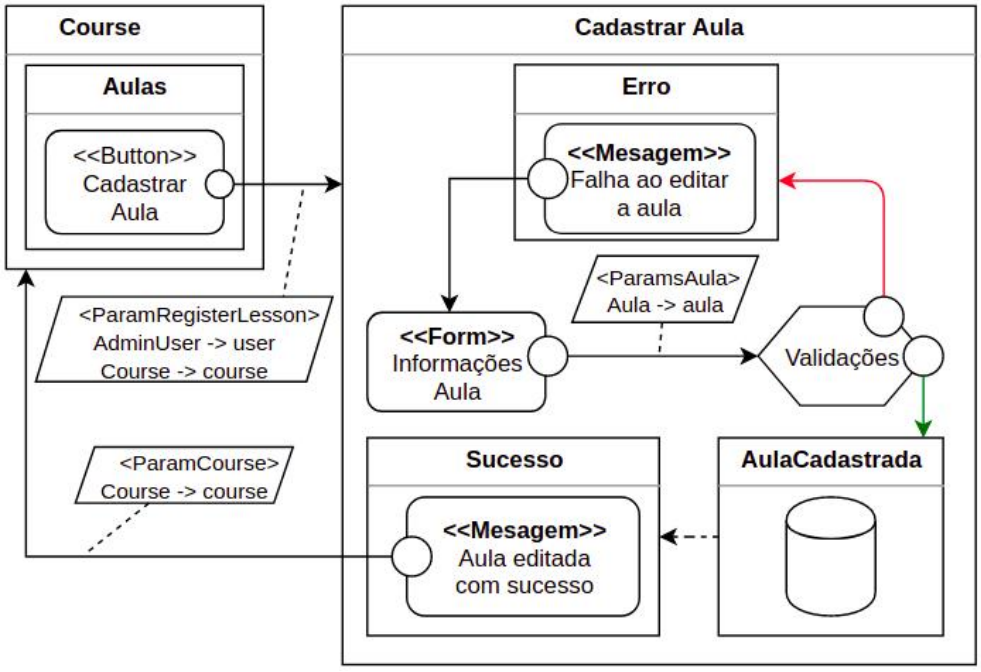
um estudante já matriculado no curso. Nesta listagem de aulas, caso o usuário selecione uma delas, ele é redirecionado para a página de *Aula*, contendo o menu de navegação da aula (apresentando as atividades desta) e o conteúdo da primeira atividade. O usuário administrador, ao clicar no botão **Cadastrar Aula**, é direcionado para a tela de registro de aula, na qual é apresentado um formulário para o cadastro.

Na área de **Artigos Relevantes**, o usuário, ao clicar em algum dos cartões listados, será redirecionado para a tela do *Artigo*, trafegando junto os parâmetros do artigo selecionado. Na área de **Discussões em Alta**, o usuário é redirecionado junto com os parâmetros da discussão selecionada para a tela de Discussão. Na área de **Orientador(es) do Curso**, são apresentados os dados do orientador, juntamente com os botões das redes sociais (Linkedin, Youtube, Facebook) cadastradas previamente pelo instrutor, ao clicar, o usuário é redirecionado para a tela externa do perfil do instrutor na rede social referente ao botão.

5.1.4.11 Aula - Registrar

Nesta subseção é apresentado o fluxo da tela *Registrar Aula*. Tal fluxo é mostrado de forma modular detalhada pela Figura 114.

Figura 114 – Modelo IFML da tela de registro de aulas



Fonte: Elaborado pelo autor.

Esta tela é accessível somente para usuários administradores pela tela do curso que será registrada a aula. O administrador, ao clicar no botão de **Registrar Aula**, acima da listagem das aulas, será direcionado para a tela de registro de aula, na qual é apresentado

um formulário para cadastro de nova aula. Nesse redirecionamento de tela, são trafegados juntos os dados de identificação do usuário administrador e do curso.

A página de registrar curso consiste em um formulário de cadastro de uma nova aula, no qual, quando preenchido e clicado no botão **Cadastrar**, são enviados como parâmetro os dados preenchidos da nova aula. Após isso, é realizado a verificação de validação dos dados informados e a autenticação do usuário administrador. Nessa validação de dados, é verificado se o tamanho máximo de informações e obrigatoriedades foram respeitados.

Após o processo de validação, caso os dados informados não estejam corretos ou validáveis, é apresentada uma tela modal informando uma mensagem de erro ("Falha ao registrar aula"), informando, logo em seguida, qual o motivo encontrado na validação. Então, o usuário é direcionado novamente para a tela de registrar aula. Caso os dados informados sejam validáveis, a aula é registrada na base de dados e, assim, é apresentada uma tela modal informando a mensagem de sucesso ("Aula registrada com sucesso"). Posteriormente, o usuário é direcionado para a tela do curso, onde serão listadas todas as aulas na área de aulas, considerando esta última cadastrada.

5.1.4.12 Aula

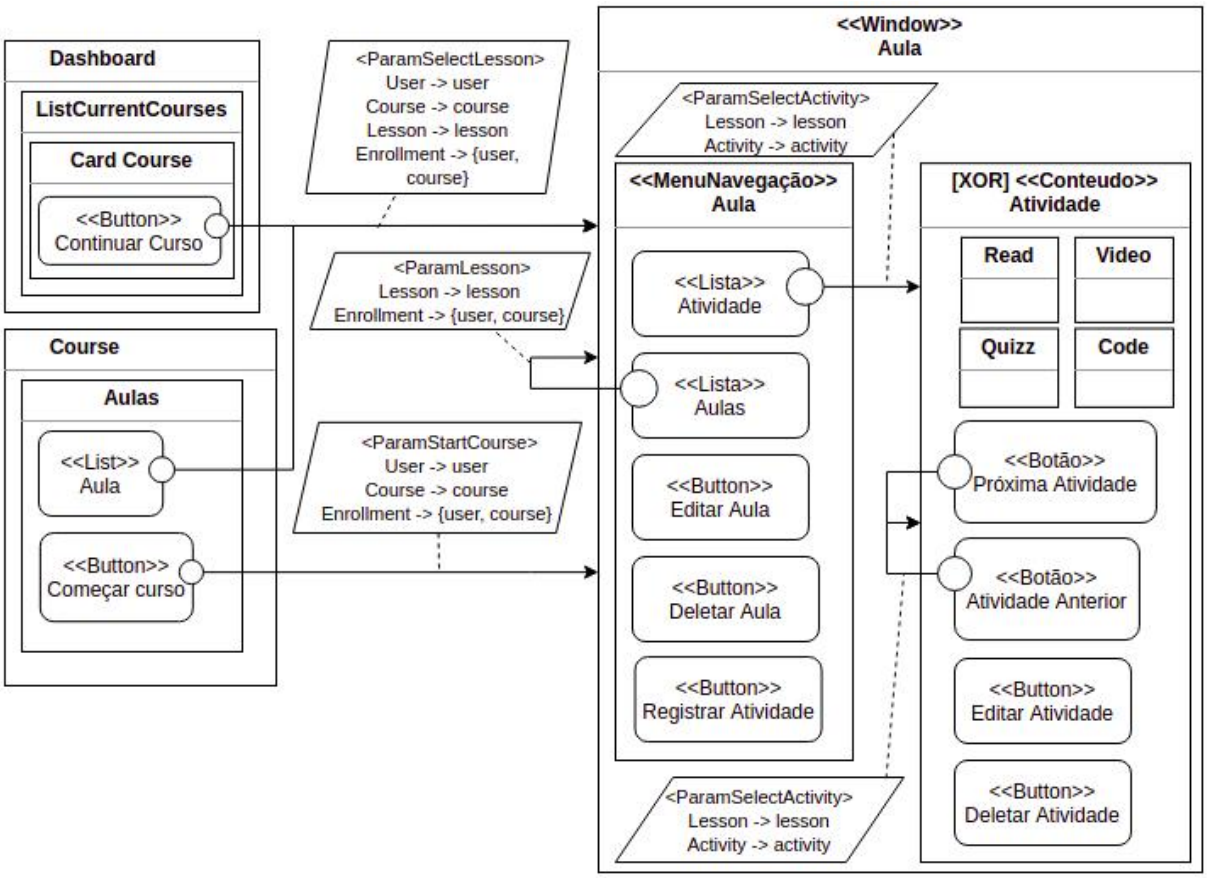
Nesta subseção, é exposto o fluxo da tela de Aula dividida em duas áreas: **Menu de Navegação da Aula** e **Conteúdo da Atividade**. Tal fluxo é apresentando com mais detalhes de forma modular na Figura 115

O usuário é encaminhado para a tela de Aula por alguns fluxos diferentes. Na tela do curso, o usuário pode chegar à tela de *aula* pelos botões **Olhar curso** ou **Começar Curso** na área do **Curso**. Outras formas são: selecionando uma das aulas listadas na área de aulas ou por um dos botões abaixo da listagem de aulas, **Começar curso** ou **Olhar curso**. Outra forma é pelo botão **Continuar Curso** na listagem de cursos em andamento na tela de *Dashboard*. Em ambos os fluxos, são trafegados como parâmetro: os identificadores do usuário, curso e aula. Este trafego de parâmetros difere apenas nos fluxos de *Continuar Curso* ou seleção de uma aula com progresso, nos quais também é informada a matrícula do estudante.

Já dentro da tela de aula, na área de **Negação de Aula**, é apresentado: botão de editar a aula, botão para deletar a aula, botão para registrar atividade, uma lista de aulas e uma lista de atividades. Os botões de **Editar Aula**, **Deletar Aula** e **Registrar Atividade** são exclusivos de usuário administradores. Os fluxos de administradores da tela de Aula serão apresentados com mais detalhes na próxima seção.

Na lista de aulas, o usuário pode selecionar outra aula, assim trafegando a matrícula do usuário (caso existir) e os dados da aula selecionada. Após isso, a listagem de atividades é atualizada para mostrar atividades desta aula selecionada e na área de ati-

Figura 115 – Modelo IFML da tela de aula



Fonte: Elaborado pelo autor.

vidade é apresentado o conteúdo da primeira atividade selecionada. Sempre que acessar esta tela ou trocar a aula, apresentar-se-á a primeira aula não concluída.

Ainda na área de navegação de aula, o usuário pode selecionar uma das atividades listadas referentes à aula previamente selecionada. Ao selecionar uma das atividades, é informado, por meio de parâmetros, o identificador da atividade selecionada, para a área de **Conteúdo da Atividade**, apresentando, assim, o conteúdo da atividade.

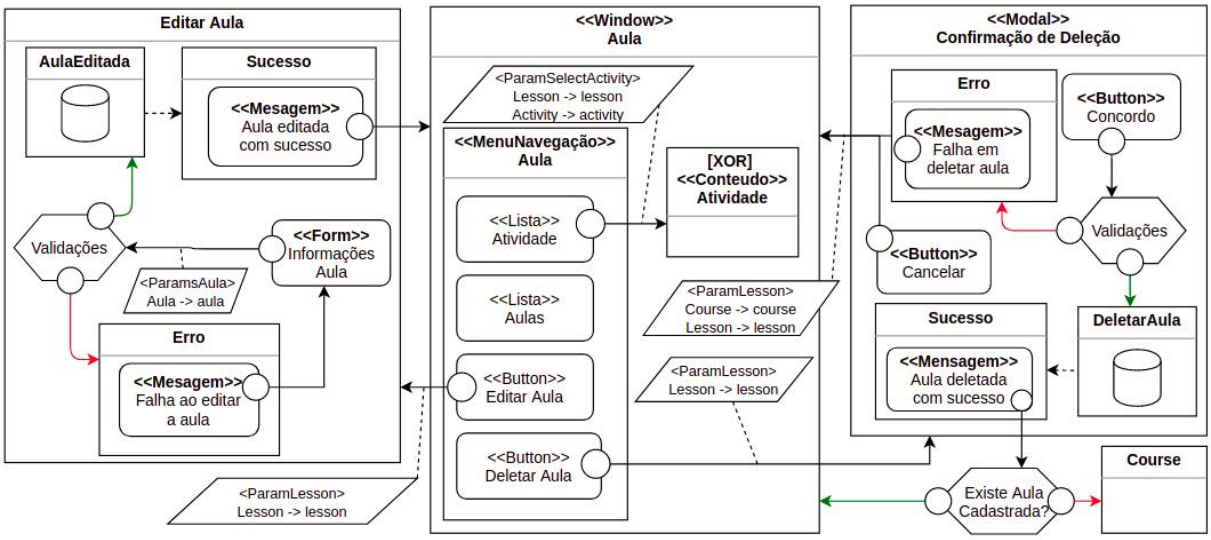
Na área de Atividade, são apresentados quatro tipos de conteúdo diferentes, como: **Read**, **Vídeo**, **Quizz**, **Code**. Estas áreas serão detalhadas na seção: 5.1.4.14.

Na mesma área também existem os botões de navegação: “Atividade Anterior” e “Próxima Atividade”. Os quais direcionam o usuário para a atividade anterior ou posterior, respectivamente. Esse direcionamento muda somente as informações da área de conteúdo de atividade, permanecendo assim na tela Aula. Ao realizar esse direcionamento, são informados os parâmetros da aula e matrícula do estudante.

5.1.4.13 Editar e Deletar Aula

Nesta subseção, é apresentado o fluxo de edição e deleção de aulas na tela de Aula. Tal fluxo é apresentando com mais detalhes de forma modular na Figura 116.

Figura 116 – Modelo IFML das telas de edição e deleção de aulas



Fonte: Elaborado pelo autor.

Na área de Atividade é apresentado, somente para usuário administradores, os botões: **Editar Atividade** e **Deletar Atividade**.

No fluxo de edição de aula, o usuário administrador, após já estar em uma tela de aula, clica no botão **Editar Aula** e é direcionado para a tela de edição da aula, realizando o fluxo de edição. Este fluxo se assemelha com os de edição anteriormente descritos, como o de edição do curso. O fluxo de deleção de aula se difere um pouco dos demais, pois, após o usuário administrador clicar no botão **Deletar Aula**, é apresentado a mensagem de confirmação, como nos demais fluxos de deleção. Porém, o que difere nesse é o fato de que se o usuário optar realmente por deletar a aula, e ela for a única existente no curso, o usuário é redirecionado para a tela do curso, caso contrário, é redirecionado para a tela de aula, apresentando a aula anteriormente a esta deletada.

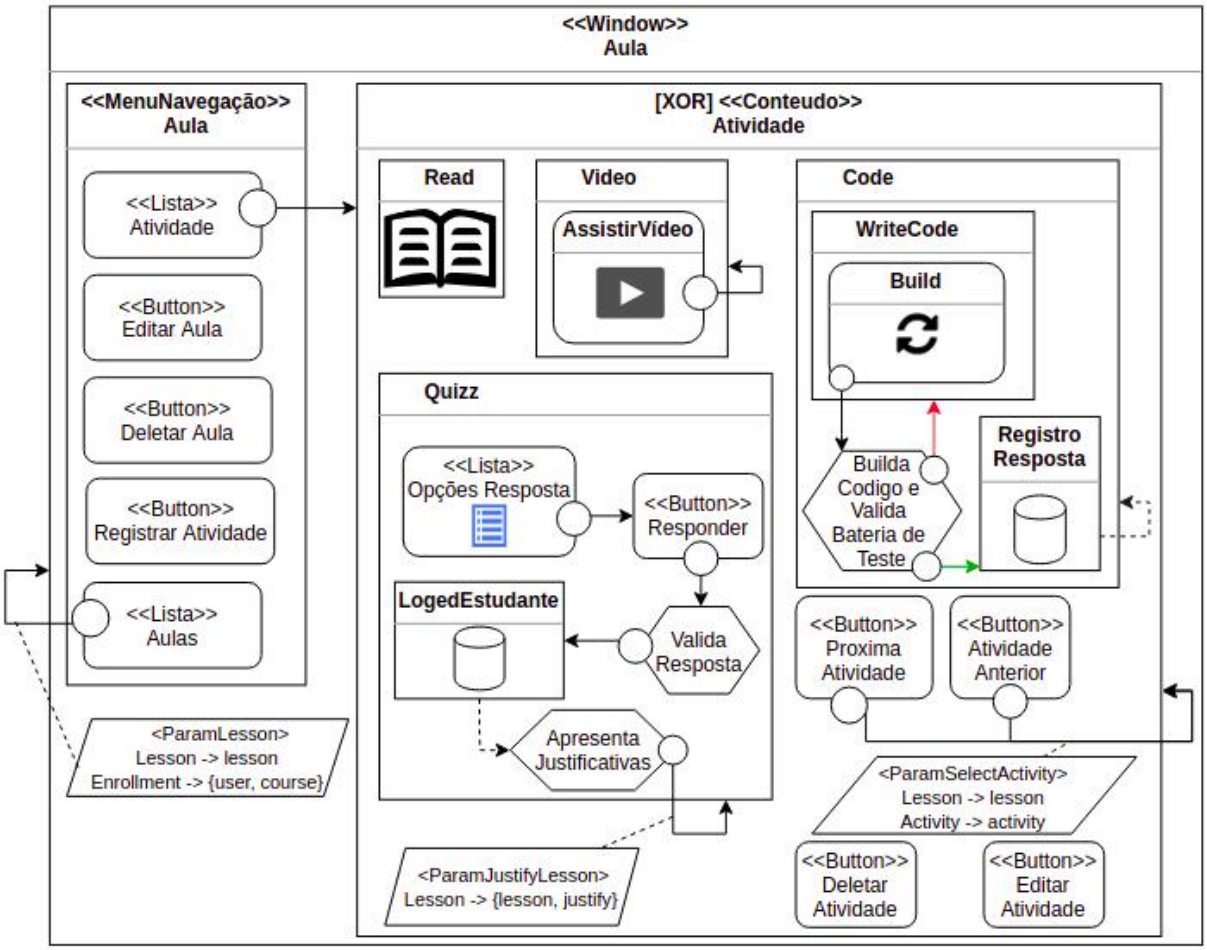
5.1.4.14 Atividade

Nesta subseção, é apresentado o fluxo de cada atividade específica a seu conteúdo como: **Leitura**, **Vídeos**, **Questionário** e **Codificação**. Tais fluxos são apresentados com mais detalhes de forma modular na imagem 117.

Após um usuário chegar à tela de aula, será apresentado, no menu lateral de navegação da área de aula, o conteúdo referente ao tipo da atividade selecionada: para atividades do tipo leitura, será apresentado um texto; para as atividades de vídeo será apresentado um vídeo junto com um texto descritivo, nesse tipo de conteúdo, o usuário, ao

clicar no *play* ou pausar do vídeo, sempre irá alterar o estado do conteúdo; para atividades do tipo questionário, será apresentado um texto descritivo da questão e, abaixo, uma lista de opções de respostas, sendo uma destas a correta. Após o estudante selecionar uma destas, e clicar no botão responder para submeter a resposta, será validado a resposta e registrada qual opção o usuário escolheu. Após isso é apresentado se o usuário acertou e as justificativas em baixo de cada opção de resposta, dizendo o porquê desta ser a correta ou não.

Figura 117 – Modelo IFML da tela de atividade



Fonte: Elaborado pelo autor.

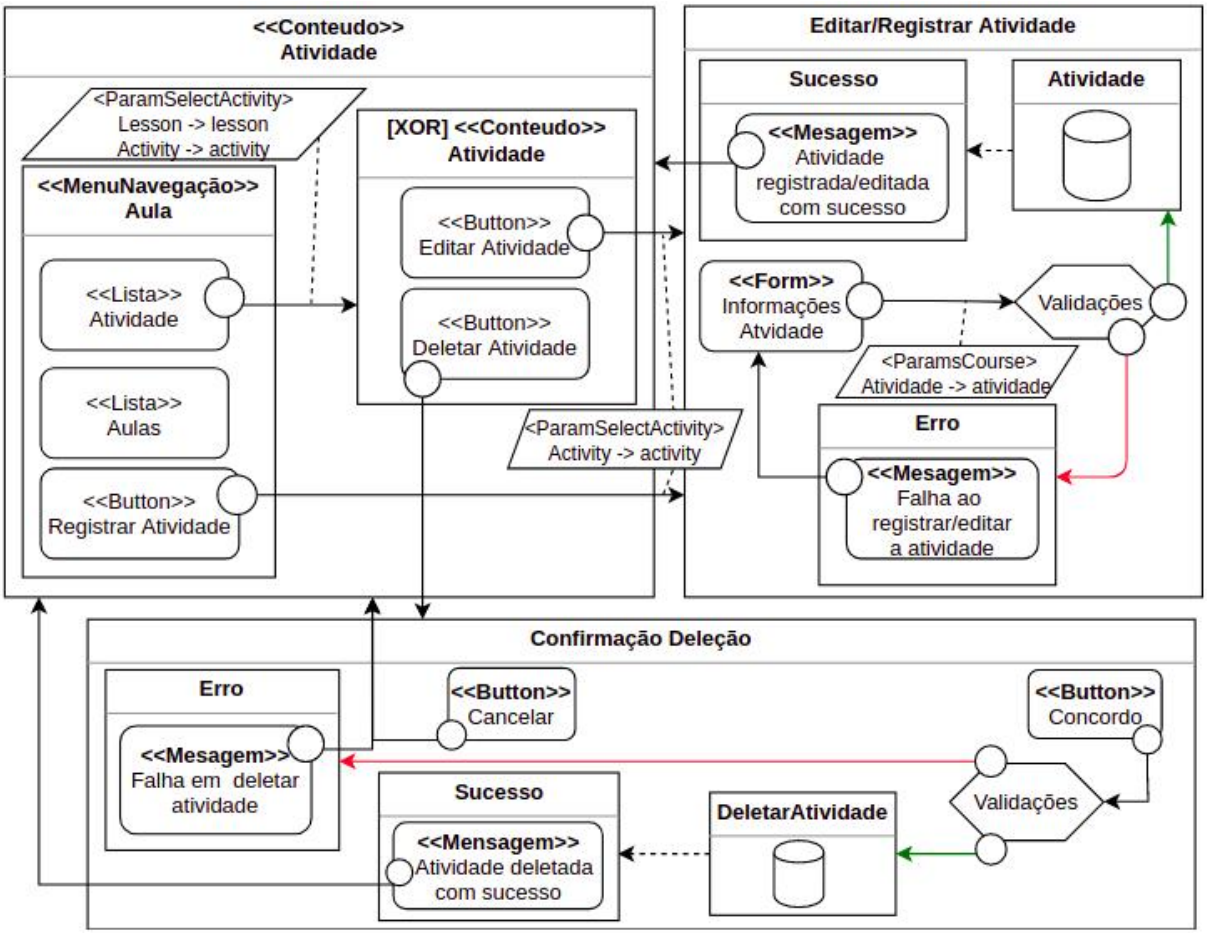
Para atividades do tipo codificação, será apresentado um texto descritivo do problema a ser resolvido e, então, apresentado a área de **Escrita de código**, a qual será especifica para o usuário codificar a solução. Após o usuário codificar a solução que acredita ser a adequada, ele pode clicar no botão de "Build" o qual executará o *build* do código compilado e testará com uma bateria de testes para validar a solução implementada. Caso não seja eficiente em questão de solução, tempo e espaço, é então apresentada em forma de tela modal a mensagem "Solução Inválida", com uma descrição em seguida do porquê, podendo ser por: problema de solução, tempo ou memória. Caso a solução seja válida, é apresentada em forma de tela modal a mensagem de sucesso "Solução Válida".

O conteúdo do estudante é registrado de diferentes maneiras de acordo com o conteúdo da atividade em questão. Para conteúdo do tipo leitura, ao avançar para a próxima atividade, é registrado o progresso e a conclusão da atividade. Para conteúdo do tipo de vídeo, é registrado o progresso e a conclusão da atividade, assim que se avança para uma próxima atividade e o vídeo foi completamente assistido, caso contrário, não é registrado como concluída a atividade. Para os conteúdos do tipo de questionário, é registrado o progresso da atividade, ao submeter uma resposta, sendo o botão de avançar para a próxima atividade apenas um auxiliador de navegação. Para atividades do tipo de codificação, por sua vez, é registrado o progresso assim que o código é validado com sucesso, sendo assim, o botão de avançar para a próxima atividade é, também, apenas um auxiliador de navegação.

5.1.4.15 Atividade - CRUD

Nesta subseção, é apresentado o fluxo de registro, edição e deleção de atividades. Tais fluxos são exclusivos de usuários administradores e são apresentados com detalhes de forma modular na imagem 118.

Figura 118 – Modelo IFML das telas de registrar, editar e deletar aula



Fonte: Elaborado pelo autor.

O fluxo de registrar ou editar uma atividade é iniciado ao clicar no botão **Registrar Atividade** na área do menu de navegação da aula ou no botão **Editar Atividade** na área do conteúdo da atividade. Após o administrador clicar em um destes botões, é direcionado para a tela de registro ou edição de atividade. Nessa página, é apresentado um formulário de cadastro de uma nova atividade, o qual, quando preenchido e clicado no botão **Cadastrar** ou **Editar**, é enviado como parâmetro dos dados preenchidos da nova atividade. Após isso, é realizado a verificação de validação dos dados informados e a autenticação do usuário administrador. Nesta validação de dados, é verificado se o tamanho máximo de informações e informações obrigatórias foram respeitadas.

Após o processo de validação, caso os dados informados não estejam corretos ou validáveis, é apresentado uma tela modal informando uma mensagem de erro "Falha ao registrar a nova atividade", informando, logo em seguida, na mesma apresentação, qual o motivo encontrado na validação. Então, o usuário é redirecionado para a tela de registrar atividade. Caso os dados informados sejam validáveis, é registrada a atividade na base de dados e, então, apresentada uma tela modal, informando a mensagem de sucesso "Curso registrado com sucesso", para registro de atividade, ou "Curso editado com sucesso", para caso de edição. Posteriormente, o usuário é direcionado para a tela de *Aula* com a atividade cadastrada selecionada.

O fluxo de deleção de uma atividade, é iniciado quando o administrador clica no botão **Deletar Atividade**, na área do conteúdo da atividade. Após o clique, é apresentada na tela uma janela do tipo modal, informando a mensagem "Concorda com a deleção da atividade? Todo o progresso desta atividade será deletado também". Após isso, o administrador tem duas escolhas, concordar ou não. Caso o usuário não concorde, basta clicar no botão *Cancelar* e então será redirecionado para a tela da atividade. Caso concorde, é verificado a autenticação do administrador, confirmando o perfil do usuário, o curso é deletado e é informado uma mensagem na tela dizendo "Atividade deletada com sucesso" e então este usuário é redirecionado para a tela *Atividade*, apresentando a atividade anterior caso exista, caso não, é redirecionado para a tela do *Curso*. Se a autenticação do usuário falhar, é informado a mensagem na tela dizendo "Falha ao deletar a atividade", assim o usuário é redirecionado para a tela da atividade já selecionada.

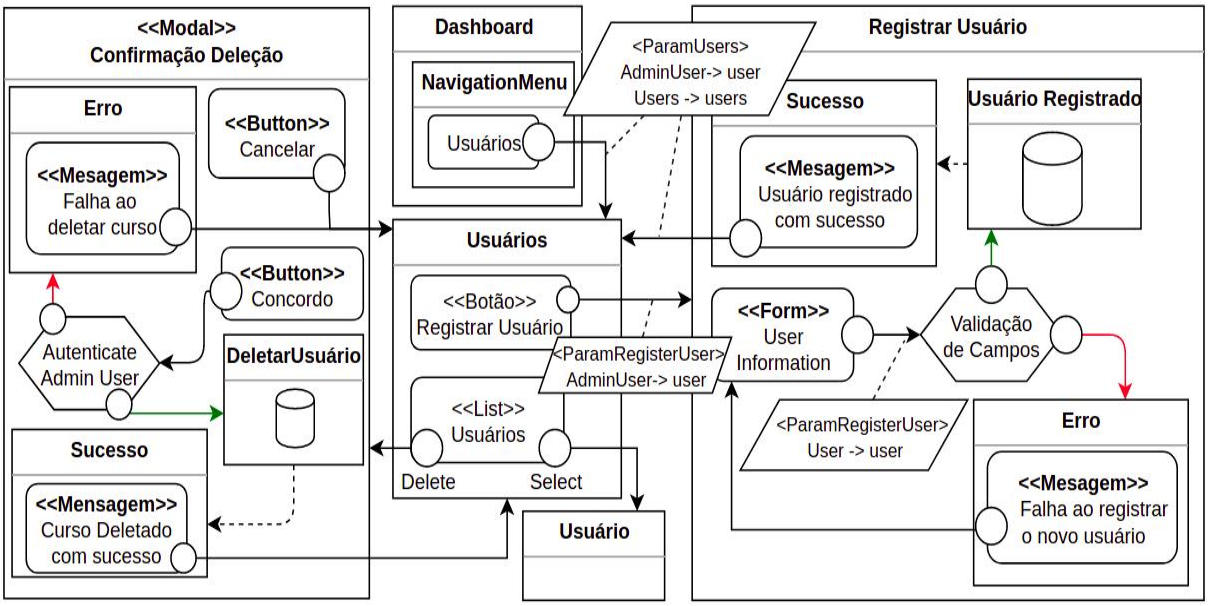
5.1.4.16 Usuários

Nesta subseção, é apresentado o fluxo de **Listar**, **Registrar** e **Deletar** usuários pela tela de listagem de usuários. Tais fluxos são exclusivos de usuários administradores e são apresentados com mais detalhes de forma modular na imagem 119.

Após o usuário administrador estar logado e na tela de *dashboard*, poderá acessar a tela de *Usuários* pelo item Usuários, no menu de navegação. Após clicar neste, será redirecionado para a tela de Usuários, assim, listando em forma de tabela todos os usuários

cadastrados no sistema, com a ordenação padrão de recência de cadastro. Nesta listagem, é apresentado em cada item o botão **Deletar** responsável em deletar o usuário da mesma linha na tabela do botão. Acima desta tabela de listagem de usuário é apresentado o botão de **Registrar novo Usuário**.

Figura 119 – Modelo IFML da tela de listagem de usuários e o fluxo de deleção de usuários



Fonte: Elaborado pelo autor.

O fluxo de deleção de um usuário é iniciado quando o administrador clica no botão **Deletar**. Após o clique, é apresentada na tela uma janela do tipo modal, informando a mensagem "Concorda com a deleção do usuário? Todos os dados de progressos serão apagados". Após isso, o administrador tem duas escolhas, concordar ou não. Caso não concorde, basta clicar no botão **Cancelar** e será redirecionado para a tela de usuários. Caso concorde, é verificado a autenticação do administrador, confirmando o perfil do usuário, ele é deletado e é informada uma mensagem na tela dizendo "Usuário deletado com sucesso", dessa forma, o administrador é redirecionado para a tela *Usuários*. Se a autenticação do usuário falhar, é informado a mensagem na tela dizendo "Falha ao deletar o usuário" e o administrador é redirecionado para a tela *Usuários*.

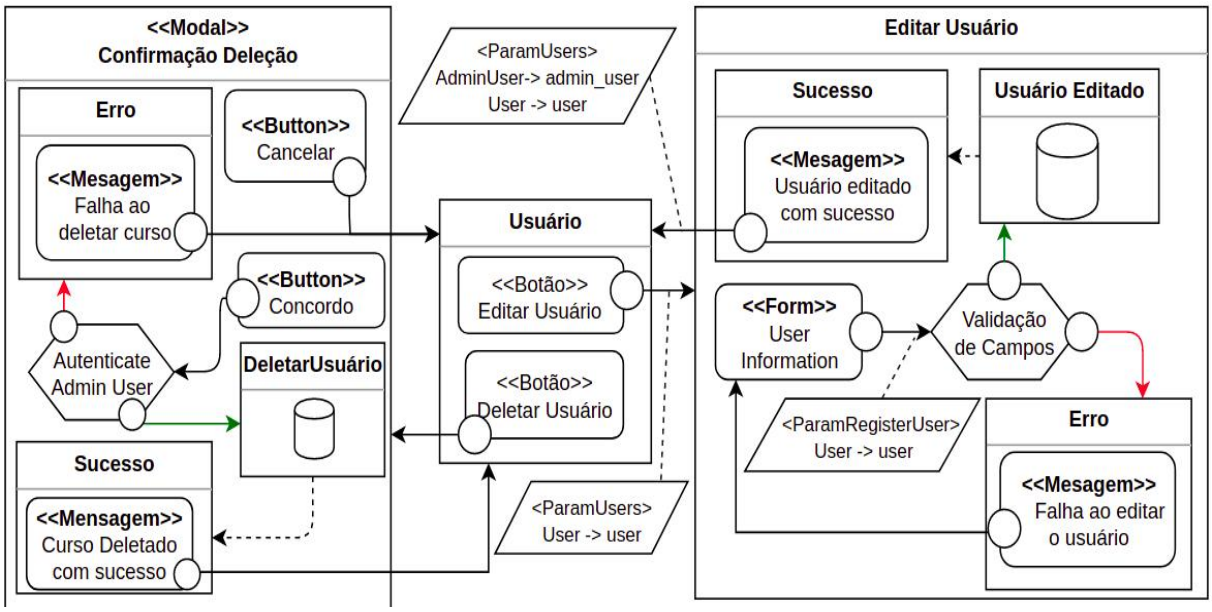
O fluxo de acessar os dados de um usuário é, iniciado quando o administrador clica no item da lista de usuários, sendo redirecionado para a tela de informação de usuários, apresentando os dados referentes ao usuário selecionado.

5.1.4.17 Usuário

Nesta subseção, é apresentado o fluxo de **Deletar** e **Editar** pela tela de detalhes de um Usuário. Tais fluxos são exclusivos de usuários administradores e são apresentados com mais detalhes de forma modular na imagem 120.

Após o usuário administrador ter selecionado um usuário na listagem de usuários, pela tela *Usuários*, são apresentados os dados do usuário selecionado e os botões *Deletar* e *Editar*. No fluxo de deleção de um usuário é, então, iniciado quando o administrador clica no botão **Deletar**. Após o clique, é realizado o mesmo fluxo explicado na seção anterior. O ponto aqui é somente para demonstrar que o fluxo também pode ser iniciado pela tela de detalhes de um usuário.

Figura 120 – Modelo IFML da tela de usuário e fluxo das telas de deleção e edição de usuários



Fonte: Elaborado pelo autor.

O fluxo de edição de um usuário é iniciado quando o administrador clica no botão **Editar**. Após o clique, o administrador será direcionado para a tela de edição dos dados do usuário selecionado, na qual é apresentado um formulário com os dados atuais do usuário, os quais são encaminhados via parâmetro. Após o administrador informar os novos dados, é realizada a validação dos dados, como identificação única do usuário na base de dados, tamanhos máximo e mínimo de dados e campos obrigatórios.

Após validação com sucesso, é registrado a alteração do curso na base de dados. Após isso, é apresentado em uma tela modal a mensagem de sucesso, informando "Usuário editado com sucesso" e, após alguns segundos, o usuário é redirecionado para a tela de detalhes do Usuário, já com os dados alterados. Caso a validação não seja executada com sucesso, é informada na tela, em janela modal, a mensagem "Falha ao editar o usuário", apresentando em seguida o motivo da falha e, em seguida, o usuário é redirecionado para a tela de edição do usuário para informar novamente os dados no formulário.

5.1.5 Modelos Design

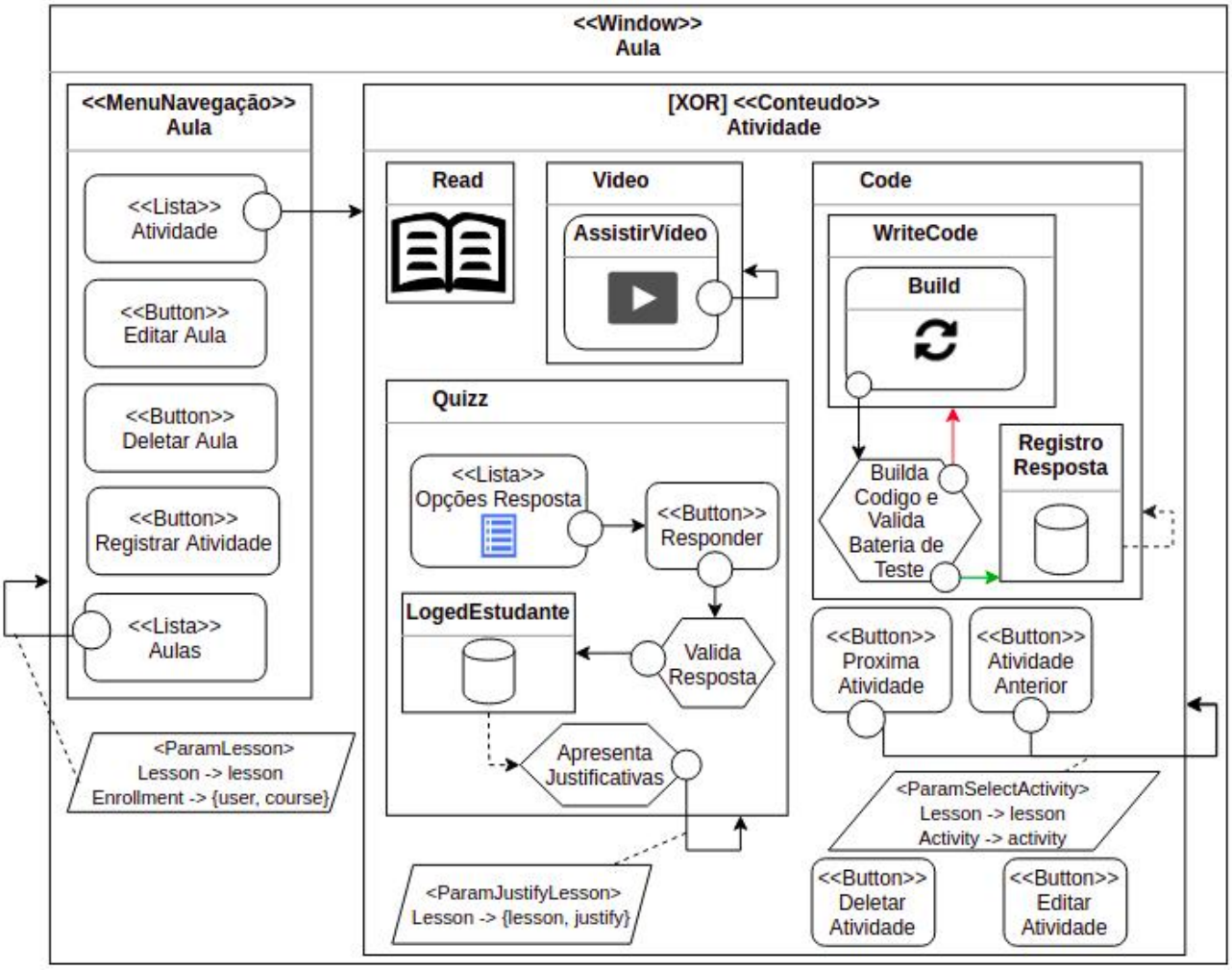
Nesta seção, será apresentado o modelo de design da tela de atividade do tipo questionário. Os demais modelos estão disponíveis no Apêndice D; Tais modelos foram construídos a partir do IFML, seguindo as quatro etapas: *Sketch*, *Wireframe*, *Mockup* e *Prototype*.

Para a construção dos *Wireframes*, foi utilizado a ferramenta *open-source* chamada *Pencil Project*. Para a construção dos *Mockups*, foram as ferramentas *Ilustrator* para as ilustrações e *Sketch* para a parte estrutural. Os *Mockups* foram disponibilizados via *Zeplin*. Já na parte visual dos protótipos, foram utilizados *React*, *HTML* e *CSS*

Modelo IFML

Nesta subseção, é apresentado o modelo IFML do fluxo de execução de atividade, Figura 121. Os detalhes deste fluxo foram apresentados na subseção 5.1.4.14.

Figura 121 – Modelo IFML do fluxo de login.

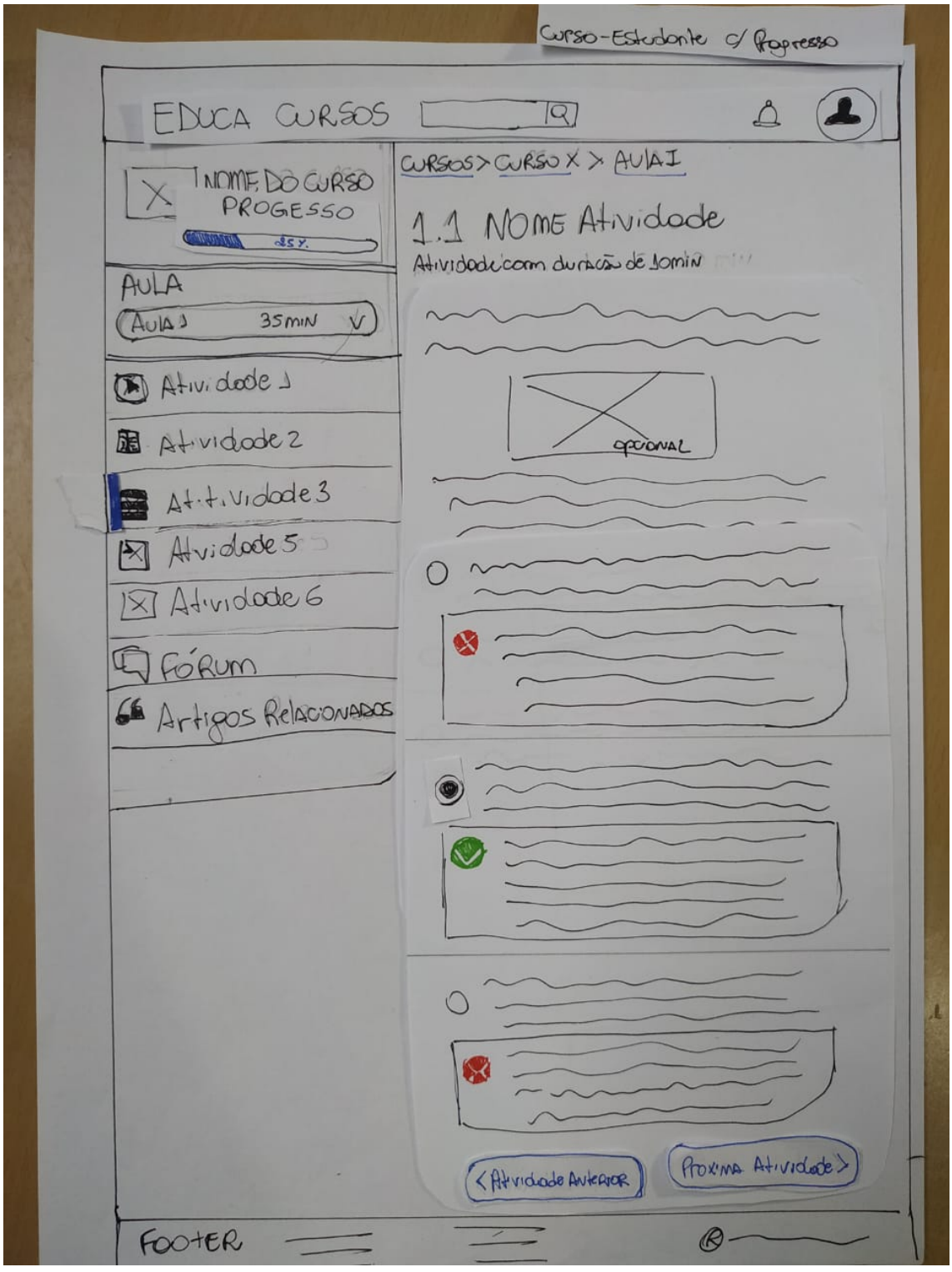


Fonte: Elaborado pelo autor.

Sketch

Neste, é apresentado o *sketch* ou esboço da tela de atividades do tipo questionário, em acesso de estudantes na situação de submissão de respostas, Figura 122.

Figura 122 – Esboços tela de atividades do tipo questionário.

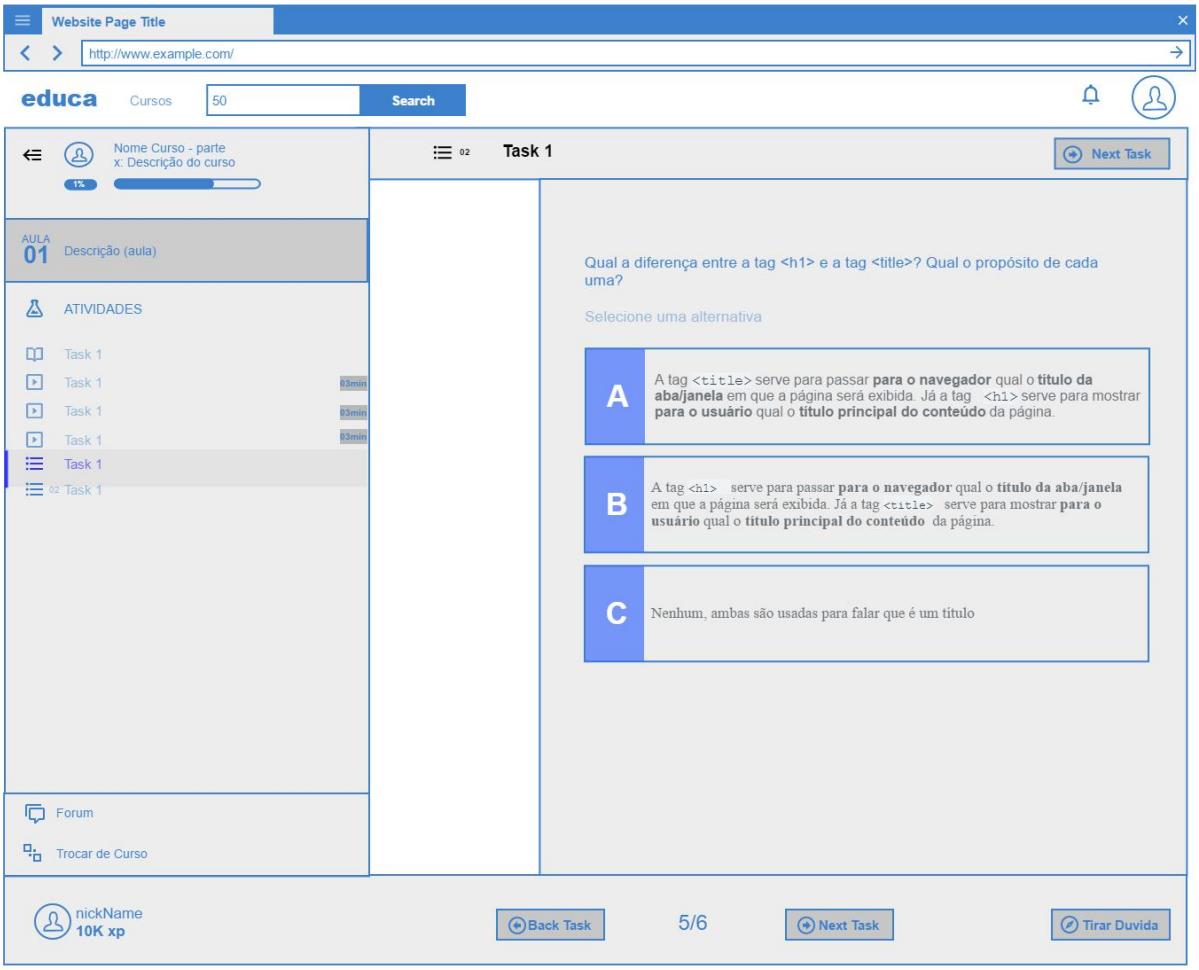


Fonte: Elaborado pelo autor.

Wireframe

Aqui é apresentado o *wireframe* construído a partir do *sketch* anterior 5.1.5, Figura 123. Para a construção dos *wireframes*, foi utilizado a ferramenta *Pencil Project*.

Figura 123 – Wireframe tela de atividades do tipo questionário.

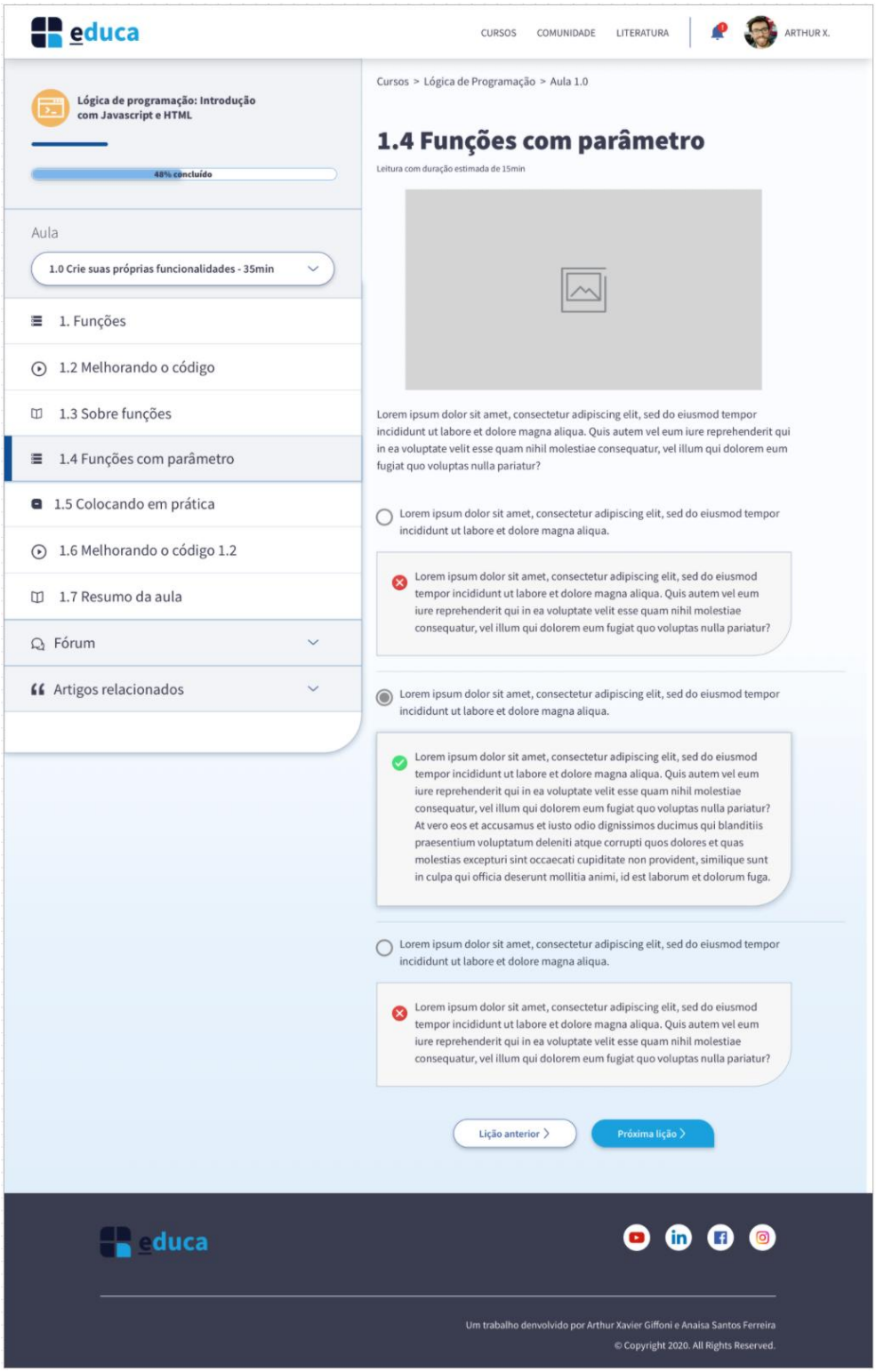


Fonte: Elaborado pelo autor.

Mockup

Nesta subseção, é presentado o *mockup* construído a partir do *wireframe* 5.1.5, Figura 124. Para esses *mockups*, foi utilizada a ferramenta *Ilustrator* para criação e disponibilizado para desenvolvimento via *Zepelin*.

Figura 124 – Mockup tela de atividades do tipo questionário.

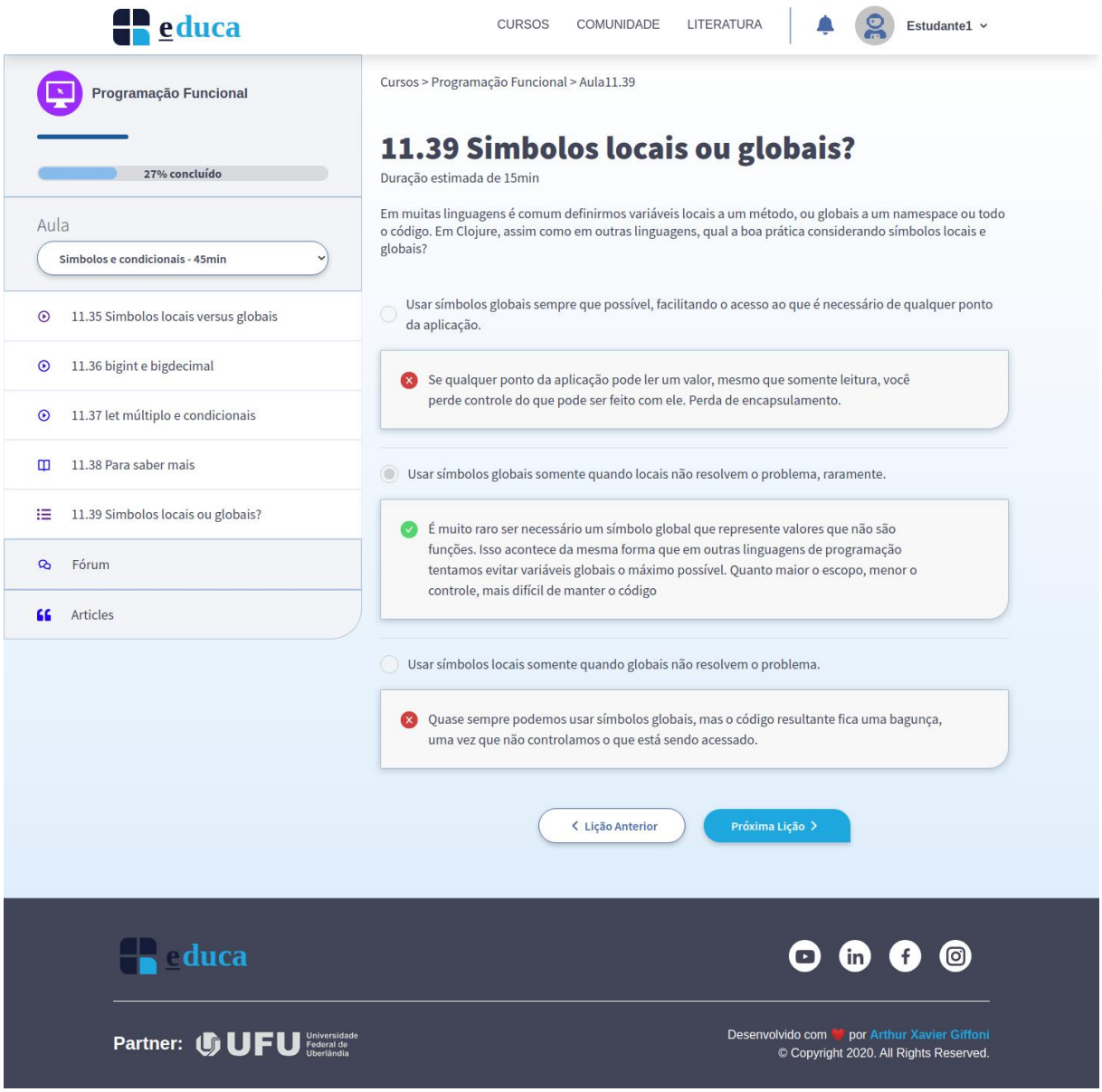


Fonte: Elaborado pelo autor.

Protótipo

Nesta subseção é presentado o protótipo da tela de atividades do tipo questionário, ilustrado na figura 125. Este modelo é o de fidelidade mais próxima em relação ao sistema final.

Figura 125 – Protótipo tela de atividades do tipo questionário



Fonte: Elaborado pelo autor.

5.1.6 Arquitetura do Backend

Como descrito no capítulo 5.1, a arquitetura escolhida para este projeto foi o padrão MVC. Neste caso no área de *back-end* do projeto, é constituída pelas camada de modelos ou da lógica da aplicação (*Model*) e camada de controles ou controladores (*Controllers*).

5.1.6.1 Controladores e Modelos

Nesta seção, serão descritas as características e quais foram os controladores e modelos construídos na API do projeto.

Controladores

Responsáveis por receber a requisição enviada à API, os controladores a interpretam, consultando os modelos, e enviam como resposta o conjunto de dados esperado ou uma resposta planejada, caso não encontrado. Dessa forma, para a regra de negócio da aplicação foram definidos os controladores e suas funções:

- UserController
  - **register**: registra um novo usuário informando obrigatoriamente: nome, sobrenome, e-mail, senha e tipo de usuário. É registrado com sucesso caso não exista outro usuário de mesmo e-mail, nome e sobrenome;
  - **index**: lista todos os usuários;
  - **login**: loga ao sistema informando um conjunto de e-mails e senhas válidas;
  - **show**: consulta todos os dados de um usuário específico, informando o seu identificador;
  - **update**: atualiza o usuário equivalente ao identificado informado, respeitando a mesma validação do cadastro (register);
  - **delete**: deleta o usuário após informado o identificador correspondente.
- UserTypeController
  - **create**: registra um novo tipo de usuário, informando obrigatoriamente o nome do tipo de usuário. É registrado com sucesso, caso não exista outro de mesmo nome;
  - **index**: lista todos os tipos de usuários (Estudante, Instrutor, Adminsitrador e Monitor);
  - **show**: consulta os dados de um tipo de usuário especifico;
  - **update**: atualiza um tipo de usuário existente, respeitando a mesma validação do cadastro (create);
  - **delete**: deleta um tipo de usuário, após informado o identificador do correspondente.
- LevelController

  - **create**: registra um novo *level* de curso, informando obrigatoriamente o nome do *level*. É registrado com sucesso, caso não exista outro de mesmo nome;
  - **index**: lista todos os *levels* de cursos cadastrados (Básico, Moderado, Avançado);
  - **update**: atualiza um *level* existente, respeitando a mesma validação do cadastro (create);
  - **delete**: deleta um *level*, após informado o identificador do correspondente.

- CourseController
  - **index**: lista todos os cursos cadastrados;
  - **show**: consulta os dados de um curso especifico, informando o identificador (tag) do curso;
  - **create**: registra um novo, curso informando obrigatoriamente o nome, identificador(*tag* do curso), descrição, o ID do administrador que está criando, o level correspondente, a duração em minutos e os instrutores correspondentes. É registrado com sucesso, caso não exista outro de mesmo identificador (*tag*);
  - **update**: atualiza um curso existente, respeitando a mesma validação do cadastro (create);
  - **delete**: deleta um curso após informado o identificador (tag) do curso correspondente. Neste caso, também são deletadas todas as aulas e atividades relacionadas a este curso.

- LessonController
  - **create**: registra uma nova aula, informando obrigatoriamente o nome, identificador (*tag*) do curso relacionado, duração em minutos e o ID do instrutor responsável pelo curso. É registrado com sucesso, caso exista o curso e instrutores relacionados e também não exista outra aula de mesmo nome relacionado ao mesmo curso;
  - **index**: lista todos as aulas cadastradas de um curso específico, informando a tag do curso;
  - **show**: consulta os dados de uma aula específica, informando o identificador (tag) do curso e o ID da aula;
  - **update**: atualiza uma aula existente, respeitando as mesmas validações do cadastro (create);
  - **delete**: deleta uma aula existente após informado o ID da aula, identificador (tag) do curso e o ID do instrutor responsável correspondente. Neste caso, também são deletadas todas as atividades relacionadas a este curso.

- ActivityTypeController
  - **create**: registra um novo tipo de atividade, informando obrigatoriamente o nome. É registrado com sucesso, caso não exista outra atividade de mesmo nome;
  - **index**: lista todos os tipos de atividades cadastrados (Texto, Vídeo, Questionário, Código);
  - **show**: consulta os dados do tipo de um tipo de atividade, informando o id;
  - **update**: atualiza um tipo de atividade existente, respeitando a mesma validação do cadastro (create);
  - **delete**: deleta um tipo de atividade, após informado o identificador do tipo de atividade correspondente.
- ActivityController
  - **create**: registra uma nova atividade, informando obrigatoriamente o nome, ID do tipo da aula, identificador (tag) do curso relacionado, ID da aula, e o ID do instrutor responsável pelo curso e o conteúdo da aula. É registrado com sucesso, caso exista o curso, aula e instrutores relacionados e também não exista outra atividade de mesmo nome relacionado ao mesmo curso;
  - **index**: lista todos as atividades cadastradas de uma aula especifica informada pelo o ID da aula;
  - **show**: consulta os dados de uma atividade especifica informando os identificadores das atividade e aula da aula;
  - **update**: atualiza uma atividade existente respeitando as mesmas validações do cadastro (create);
  - **delete**: deleta uma atividade existente após informado o ID da atividade, o ID da aula, identificador (tag) do curso e o ID do instrutor responsável correspondente.
- EnrollmentController
  - **register**: registra a matrícula do estudante com um curso. É registrado com sucesso caso o estudante e o curso existam e não exista já uma matricula entre este estudante com este curso.
  - **show**: apresenta os dados da matricula do estudante com o curso, seu progresso e a nota de avaliação que o estudante deu ao curso.
  - **list**: lista todas matriculas, caso exista, de um estudante especifico. É listado com sucesso, caso o estudante informado exista no sistema.

  - **rate**: atribui a nota que o estudante avaliou o curso. É executado com sucesso caso o estudante e o curso existam. Caso o estudante mude o valor da nota, é substituído a anterior.
  - **continue**: informa qual a última atividade realizada pelo estudante daquele curso, auxiliando-a saber qual a próxima deve fazer. Caso o estudante não tenha realizado ainda nenhuma atividade, é informado que a continuação deve ser a partir da primeira atividade da primeira aula do curso. Estudante, curso e matrícula devem existir no sistema.

- LessonProgressController
  - **register**: registra um vínculo do estudante com a aula iniciada informando o ID do estudante, o identificador do curso e o ID da aula. O registro é realizado com sucesso contando que o curso, a aula e o estudante existam como também a matrícula do estudante com o curso relacionado a aula;
  - **list**: lista todos os progressos de atividades relacionados ao progresso da aula e ao estudante informado. A listagem é realizada com sucesso contando que o estudante e a aula existam, como também a matrícula do estudante com o curso.

- ActivityProgressController
  - **register**: registra um vínculo do estudante com a atividade iniciada, informando o ID do estudante, o identificador do curso, o ID da aula e o ID da atividade. O registro é realizado com sucesso, considerando que o estudante e a atividade existam e que também exista uma matrícula do estudante com o curso;
  - **complete**: atualiza o status do progresso do estudante com uma atividade para completado. A atualização é realizada com sucesso, considerando que ela já não esteja completada e que o estudante, curso, aula e atividade existam.

- CertificateController
  - **create**: registra uma novo certificado, informando o ID do usuário e o identificador (*tag*) do curso;
  - **index**: lista todos os certificados registrados no sistema;
  - **show**: consulta os dados de um certificado, informando o ID ou código de validação do certificado;
  - **student**: lista todos os certificados que um estudante conquistou informando o ID do estudante;
  - **course**: lista todos os certificados vinculados a um curso informando o identificador (*tag*) do curso.

Modelos

Um detalhe sobre esse ponto do MVC é que nem sempre as entidades levantadas de uma regra de negócio serão uma tabela no banco de dados, podendo ser um modelo. Por exemplo, neste projeto, no caso dos usuários, não se tem uma tabela específica para administradores, porém, é interessante ter um modelo que represente um administrador no *Back-end* para a implementação das funcionalidades e validações necessárias. Toda essa análise, sobre ser construído um modelo de uma entidade que não é uma tabela, dependente muito da sua regra de negócio. Se a entidade terá validações e funcionalidades específicas, então, faz sentido ser criado um modelo.

Na área de desenvolvimento e arquitetura, é comum as pessoas escolherem colocar as validações nos controladores ou nos modelos, dadas suas justificativas particulares. Neste projeto, foram colocadas somente validações de requisição nos controladores, enquanto o detalhamento de validações e funcionalidades foram projetos para ficarem nos modelos. Esta decisão foi tomada de modo a tornar o controlador mais específico em organizar e interpretar as requisições recebidas pelo *Front-end*. Dessa forma, acreditando facilitar a manutenção e organização do código, ao centralizar as validações e ferramentas no modelo. Sendo assim, para a regra de negócio da aplicação, foram definidos os seguintes modelos e suas funções:

- UserType
  - Associações:

    hasMany User: um mesmo tipo de usuário pode estar relacionado a nenhum ou vários usuários;
  - Funcionalidades:

    type_exist: dizer se um tipo de usuário existe pelo ID ou nome;
- User
  - Associações:

    belongsTo UserType: um usuário deve estar relacionado a um tipo de usuário;

    hasOne Student: um usuário deve ser somente um usuário, caso ele seja do tipo usuário;

    hasOne Instructor: um usuário deve ser somente um instrutor, caso ele seja do tipo instrutor.
  - Funcionalidades:

    authentication: dizer se a senha com o e-mail informados pelo usuário estão corretos;

e-mail_exist: dizer se existe um usuário pelo e-mail;

user_url_exist: dizer se existe um usuário pelo user_url;

is_admin: dizer se um usuário é do tipo admin;

generate_url_user: gerar um url_user a partir do nome e sobrenome do usuário, sendo a tag do usuário.

- Student
  - Associações:

    belongsTo User: um estudante deve estar relacionado a um usuário;

    hasMany Enrollment: um usuário pode estar relacionado a zero ou muitas matrículas.

- Instructor
  - Associações:

    belongsTo User: um instrutor deve estar relacionado a um usuário;

    hasMany Teach: um instrutor pode estar relacionado a zero ou muitos lecionados .

- Level
  - Associações:

    hasMany Course: um mesmo level pode estar relacionado a nenhum ou vários cursos.

  - Funcionalidades:

    level_exist: dizer se um level existe pelo ID ou nome.

- Course
  - Associações:

    belongsTo Level: um curso está relacionado a um *level*;

    hasMany Teach: um curso é lecionado por um ou mais instrutores;

    hasMany Enrollment: um curso pode ter muitos estudantes matriculados;

    hasMany Lesson: um curso pode ter muitas aulas cadastradas;

    hasMany Certificate: um curso pode estar relacionado a vários certificados.

  - Funcionalidades:

    instructors: lista os instrutores que lecionam este curso;

    course_exist: diz se um curso existe, informando a tag identificadora ou id;

formated_identifier_course: formata a tag identificadora do curso, quando gerado;

firstActivity: informa qual a primeira aula do curso;

average_rates: retorna a média das notas avaliadas pelos alunos sobre o curso.

- Lesson
  - Associações:

    belongsTo Course: uma aula está relacionada a um curso;

    hasMany Activity: uma aula pode ter várias atividades;

    hasMany LessonProgress: uma aula pode ter vários progressos de estudantes diferentes.

  - Funcionalidades:

    exist_by_name: diz se uma aula existe, informando o nome e o curso relacionado;

    exist_by_lesson_id: diz se uma aula existe, informando o ID e o curso relacionados.

- Activity
  - Associações:

    belongsTo ActivityType: uma atividade está relacionada a um tipo de atividade;

    belongsTo Lesson: uma atividade está relacionada a uma aula;

    hasOne Read: uma atividade pode estar relacionada a somente um conteúdo do tipo texto;

    hasOne Video: uma atividade pode estar relacionada a somente um conteúdo do tipo vídeo;

    hasOne Question: uma atividade pode estar relacionada a somente um conteúdo do tipo questionário;

    hasOne Code: uma atividade pode estar relacionada a somente um conteúdo do tipo código;

    hasMany ActivityProgress: uma atividade pode estar relacionada a vários progressos de estudantes diferentes.

  - Funcionalidades:

    exist_by_id: diz se uma atividade existe, informando o ID e a aula relacionado;

exist_by_name: diz se uma atividade existe, informando o nome e a aula relacionados;

content: retorna o conteúdo da aula de acordo com o tipo da atividade;

create_activity_content: registra o conteúdo, informando o tipo da atividade;

update_activity_content: atualiza o conteúdo, informando o tipo da atividade.

- Read
  - Associações:

    belongsTo Activity: um conteúdo de leitura está relacionado a uma atividade.

- Video
  - Associações:

    belongsTo Activity: um conteúdo de vídeo está relacionado a uma atividade.

- Question
  - Associações:

    hasMany Answer: um questionário pode possuir várias opções de respostas.

    belongsTo Activity: um conteúdo de questionário está relacionado a uma atividade.
  - Funcionalidades:

    random_answers: Randomiza quatro opções de resposta, com posições diferentes, sendo apenas uma delas correta;

    correct_answer: Diz se a resposta informada é correta ou não.

- Answer
  - Associações:

    belongsTo Question: uma resposta está relacionada a uma questão.

- Code
  - Associações:

    belongsTo Activity: um conteúdo de desafio de código está relacionado a uma atividade.
  - Funcionalidades:

correct_compile: diz se o código foi compilável e, caso não, o porquê;

correct_answer: diz se o código informado é suficiente para a resolução do desafio, rodando a bateria de testes.

- Teach
  - Associações:

    belongsTo Course: um lecionamento está vinculado a um curso;

    belongsTo Instructor: um lecionamento está vinculado a um instrutor.

- Enrollment
  - Associações:

    hasMany LessonProgress: uma matrícula pode estar relacionada a vários progressos de aulas;

    belongsTo Course: uma matrícula está relacionada a um curso;

    belongsTo Student: uma matrícula está relacionada a um estudante.
  - Funcionalidades:

    last_unfinished_activity: diz qual é a última atividade iniciada e não finalizada pelo estudante;

    percentage_progress: retorna qual o percentual de progresso de um estudante ao curso;

    activities_completed: diz quais atividades foram completadas neste curso pelo estudante;

    total_activities: retorna o número total de atividades ativas.

- LessonProgress
  - Associações:

    hasMany ActivityProgress: um progresso de aula pode estar relacionado a vários progressos de atividades;

    belongsTo Enrollment: um progresso de aula está relacionado a uma matrícula;

    belongsTo Lesson: um progresso de aula está relacionado a uma aula.
  - Funcionalidades:

    register_lesson_progress: registra um progresso de aula.

- ActivityProgress
  - Associações:

belongsTo LessonProgress: um progresso de atividade está relacionado a um progresso de aula;

belongsTo Activity: um progresso de atividade está relacionado a uma atividade.

– Funcionalidades:

activities_completed: diz quais as atividades já concluídas de uma matrícula.

5.1.6.2 Arquitetura REST API

Nessa seção, será abordada com mais detalhes a arquitetura e as decisões específicas sobre o *Back-end* do sistema. Dessa forma, foi construída uma API seguindo o padrão REST.

A Tabela 14 detalha as rotas de recursos definidos do projeto, que atendem os cenários para o modelo de *User*. Já no Apêndice C, são apresentadas todas as rotas de requisições definidas no projeto.

Tabela 14 – Rotas do recurso de User da API do sistema.

Verbo	Rota	Controlador	Descrição
POST	/register	User.register	registra um novo usuário informando obrigatoriamente: nome, sobrenome, e-mail, senha e tipo de usuário. É registrado com sucesso, caso não exista outro usuário de mesmo e-mail, nome e sobrenome;
GET	/users	User.index	lista todos os usuários;
POST	/login	User.login	loga ao sistema informando um conjunto válido de e-mails e senhas;
GET	/users/:user_url	User.show	consulta todos os dados de um usuário específico, informando o se identificador;
PUT	/users/:user_url	User.update	atualiza o usuário equivalente ao identificado informado, respeitando a mesma validação do cadastro (register);
DELETE	/users/:user_url	User.delete	deleta o usuário após informado o identificador do usuário correspondente.

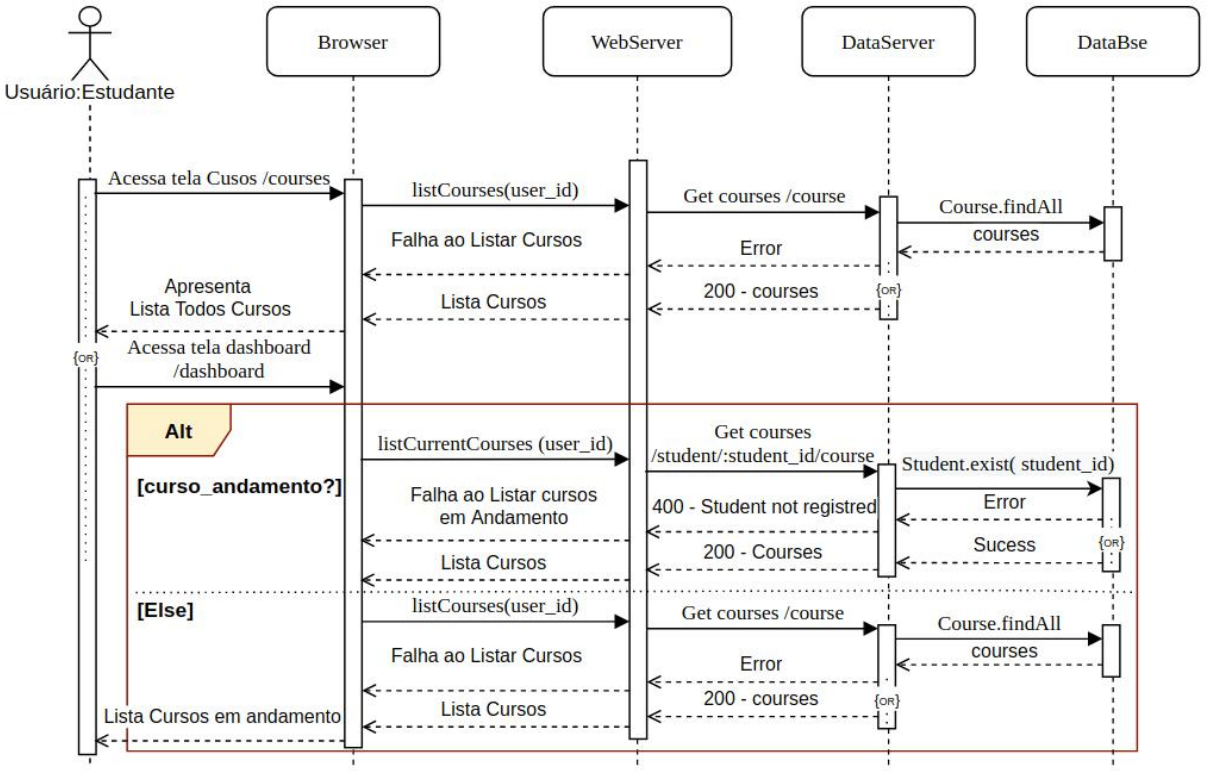
Fonte: Elaborado pelo autor.

5.1.7 Diagramas de Sequência

Nessa seção, será apresentado com mais detalhes a arquitetura das iterações entre os objetos diferentes em cada camada do processo. Dessa forma, foram construídos os modelos de sequência.

O modelo apresentado na Figura 126 detalha a sequência do processo de após o usuário estiver logado, acessar á área cursos da tela de *dashboard*. Lembrando que caso o usuário ainda não tenha nenhum curso em andamento, o sistema apresentará o catálogo de todos os cursos. No Apêndice E, são apresentados todos diagramas de sequência criados.

Figura 126 – Diagrama de sequência do processo de um usuário acessar cursos



Fonte: Elaborado pelo autor.

5.1.8 Arquitetura do Frontend

Como descrito no cap 5.1, a arquitetura escolhida para este projeto foi o padrão MVC. Neste caso, a área de *front-end* do projeto é constituída pela camada de apresentação ou visualização (*Views*). Nessa arquitetura, o conjunto de *views* foi dividido na construção das páginas e componentes.

5.1.8.1 Visões

Nesta seção, são descritas as características e quais foram os *Views* construídos, assim, estruturando a camada de apresentação do projeto. Vale ressaltar que, nesse projeto, a área de *views* foi separada em dois grupos, **pages** e **components**. A página é composta por componentes, formando a apresentação de uma *view*. Além disso, é importante ressaltar que cada componente, como as páginas, possui arquivos de estilização próprios, além da estilização global.

Nesse projeto a área de *views* foi separada em dois grupos, **Componentes Web** (*components*) e **Páginas Web** (*pages*). A página é composta por componentes, formando a apresentação de uma *view*. Outro fator que vale levantar é que cada componente ou página possuem arquivos de estilização próprio além da estilização global.

Componentes Web

Nesta subseção são apresentados os compoentes, os quais foram utilizados para evitar a recodificação de itens mais comuns presentes nas *pages*, como listagem de *cards*, *menus*, *footer* e área de atividade. Estes estão descritos com detalhes em seguida:

- **Menus**: componentes com objetivo de representarem menus com apresentação frequente na plataforma;
  - **TopBar**: responsável por representar menu do topo, o qual apresenta as opções de acessar *dashboard*, cursos, artigos, discussões e área de perfil;
  - **DropDown**: menu de lista vertical, responsável por apresentar as funções relacionadas ao perfil e logout, como dados de perfil, certificados, meus cursos e sair.
- **Footer**: responsável por apresentar área fixa no final da página, com algumas informações, como Logo da Plataforma (*link* para acessar as páginas de *Home* ou *Dashboard* caso esteja logado), cursos, redes sociais etc.
- **Cards**: responsável por apresentar uma lista de cartões, na qual pode ser apresentado cursos, artigos etc. O card a ser clicado navega o usuário até a página do item selecionado, por exemplo, um curso ou artigo.
- **Activities**: componentes com objetivo de representar a área de conteúdo para um tipo de atividades específica, sendo ela, de leitura, vídeo, questionário ou codificação.
  - **Read**: responsável por apresentar conteúdo de atividades do tipo de leitura, pode apresentar tanto somente um texto, quanto um texto com imagem.
  - **Video**: responsável por apresentar conteúdo de atividades do tipo de vídeo, apresenta tanto somente um vídeo como também pode apresentar um texto a partir do vídeo.
  - **Question**: responsável por apresentar conteúdo de atividades do tipo de questionário, apresenta um questionário e as opções de respostas.
  - **Code**: responsável por apresentar conteúdo de atividades do tipo de codificação, apresenta um texto e uma área de escrita de código, junto a um terminal que apresenta o retorno da execução do código escrito.

Páginas Web

Nesta subseção são apresentadas as páginas da aplicação e seus objetivos particulares. São estas:

- **Home**: página principal, com objetivo de apresentar de forma geral a plataforma;
- **Register**: responsável por viabilizar o registro de novos usuários por tela;
- **Logon**: responsável por viabilizar o login de usuários por tela;
- **Dashboard**: página que o usuário acessa após se logar na plataforma. Nesta são apresentas sugestões de cursos e, caso tenha cursos em andamento, apresenta área de acesso rápido ao curso, artigos relacionados e discussões dos cursos em andamento;
- **Courses**: páginas relacionadas à área de cursos:
  - **Todos os Cursos**: página responsável por listar todos os cursos da plataforma;
  - **Meus Cursos**: página responsável por listagem dos cursos que o usuário já tenha iniciado um progresso;
  - **Show**: página responsável por apresentar o conteúdo de um curso selecionado previamente por um usuário;
  - **Register**: página para realizar registro de um novo curso, esta é acessível somente para usuários administradores;
  - **Edit**: página para realizar edição de cursos, está é acessível para usuários administradores e instrutores relacionados ao curso.
- **Lesson**: páginas relacionadas à área de aulas:
  - **Show**: apresentação do conjunto de atividades de uma aula, selecionada prevamente por um usuário;
  - **Register**: página para realizar registro de uma nova aula dentro de um curso, acessível para usuários administradores e instrutores relacionados ao curso;
  - **Edit**: página para realizar edição da aulas, acessível para usuários administradores e instrutores relacionados.
- **Activity**: páginas relacionadas à área de atividades:
  - **Show** :apresentação do conteúdo de uma atividade, selecionada previamente por um usuário na aula;
  - **Register**: página para realizar registro de uma nova atividade dentro de uma aula, acessível para usuários administradores e instrutores relacionados ao curso;

  - **Edit**: página para realizar edição de atividades, acessível para usuários administradores e instrutores relacionados ao curso.

- **Articles**: páginas relacionadas à área de artigos:
  - **List**: página responsável por listar todos os artigos da plataforma;
  - **Show**: apresentação do conteúdo de um artigo, selecionado previamente por um usuário na aula;
  - **Register**: página para realizar registro de um novo artigo, acessível para usuários instrutores;
  - **Edit**: página para editar um artigo registrado, acessível para usuários instrutores.

- **Discussion**: páginas relacionadas à área de discussões:
  - **List**: listagem de discussões relacionadas a algum curso;
  - **Show**: apresentação de uma discussão relacionada a algum curso, listando todos os comentários criados por usuários.

- **Users**: páginas relacionadas à área de gerência de usuários:
  - **List**: listagem de usuários cadastrados no sistema, acessível para usuários administradores;
  - **Register**: registro de usuários no sistema pela área de gerência, acessível para usuários administradores;
  - **Show**: apresentação dos dados de um usuário, acessível para o próprio usuário ou por usuários administradores;
  - **Edit**: edição dos dados de um usuário, acessível para o próprio usuário ou usuários administradores.

5.2 Implementação

Nesta seção são apresentadas as tecnologias e quais versões utilizadas, como linguagem, frameworks e bibliotecas. Vale ressaltar que, ambas as implementações, tanto *back-end* quanto *front-end*, se encontram no meu perfil do *GitHub*. Os repositórios são:

- **Back-end**: https://github.com/arthurxavier/educa-api
- **Front-end**: https://github.com/arthurxavier/educa-web

5.2.1 Back-end

Nesta seção, será descrito como foi implementado a arquitetura definida do *back-end*. Para a implementação da API, descrita em 5.1.6.2, foram utilizadas as tecnologias seguintes.

5.2.1.1 Tecnologias Utilizadas

- Node.js (versão 14.16) - uma plataforma para construir aplicações web escaláveis de alta performance, utilizando a linguagem *JavaScript*. Construído em cima da *engine V8* que interpreta *JavaScript*, criado pela *Google* e usado em seu navegador, o *Chrome*. Possibilitando utilizar a linguagem tanto para o servidor quanto para o *browser*. Utiliza uma arquitetura voltada a eventos, integrando muito bem com *JavaScript* (*callbacks!*). Usando um *loop* de eventos, o Node interpreta, em uma única *thread*, as requisições de forma assíncrona em vez de sequenciais, e não permitindo bloqueios. Isso o torna rápido, e ideal para lidar com um número muito alto de requisições;
- Sequelize (versão 5.21) - um ORM Node.js baseado em *promisses* para Postgres, MySQL, MariaDB, SQLite e Microsoft SQL Server. Ele oferece suporte para transações sólidas, relações, carregamento rápido e lento, replicação de leitura;
- Nodemon (versão 2.0.3) - uma ferramenta que ajuda a desenvolver aplicativos baseados em node.js reiniciando automaticamente o aplicativo quando mudanças, de arquivo no diretório, são detectadas;
- PostgreSQL (versão 9.5.23) - um sistema de banco de dados relacional de objeto de código aberto com uma forte reputação de confiabilidade, robustez de recursos e desempenho;
- CORS (versão 2.8.5) - um pacote node.js para fornecer um *middleware Connect/Express* que pode ser usado para habilitar *CORS* com várias opções;
- Express (versão 4.17.1) - um pacote node.js para fornecer uma camada de recursos fundamentais para aplicativos da web, como o gerenciamento e manipulação de rotas.

5.2.2 Front-end

Nessa seção, será descrito como foi implementado a arquitetura definida do *front-web*. Para a implementação foi utilizado como tecnologia a linguagem *javascript* com o *framework reactjs* e o uso de algumas bibliotecas públicas que serão escritas nas próximas subseções.

5.2.2.1 Tecnologias Utilizadas

- React (versão 16.13.1) - uma biblioteca JavaScript criada pelo Facebook para o desenvolvimento de aplicações *front-end*. Baseada em componentes, que permitem o reaproveitamento de código e facilita a manutenção. No padrão de arquitetura MVC Model View Control ou Modelo Visão Controle, em português, ela é comparado ao desenvolvimento da camada *View*, que é a interface com o usuário (UI);
- React-dom (versão 16.13.1) - pacote ou biblioteca que fornece métodos específicos do DOM que podem ser usados no nível superior da aplicação como renderizar as páginas e componentes;
- React-icons (versão 3.9.0) - biblioteca que possibilita o uso de ícones em projetos React. Utiliza importações do tipo ES6, permitindo incluir apenas os ícones que serão usados;
- React-router-dom (versão 5.1.2) - biblioteca padrão de gerenciamento de rotas do React.js, mantendo a interface do usuário em sincronia com o valor atual da URL acessada;
- React-scripts (versão 3.4.1) - biblioteca com conjunto de *scripts* que ajudam a iniciar projetos sem a necessidade de uma configuração robusta;
- Axios - (versão 0.19.12) - biblioteca para gerenciamento de *promises* possibilitando requisições assíncronas.

Parte II

Conclusão

6 Conclusão

Neste capítulo, são apresentadas as conclusões e contribuições que foram alcançadas por meio deste trabalho e, ainda, algumas propostas de trabalhos futuros com o intuito de dar continuidade a esta pesquisa.

6.1 Conclusões

Foi apresentado neste trabalho um levantamento de arquiteturas utilizadas para a elaboração do módulo de comunicação para STIs, bem como as ferramentas utilizadas. Este foi realizado recentemente por meio de uma Revisão Sistemática da Literatura (RSL), Capitulo 3. Na RSL, foram mostrados trabalhos de modelagem de software, os quais utilizaram a linguagem IFML como ferramenta de modelagem. Constatou-se que, na maioria dos trabalhos encontrados, não há destaque para o módulo de comunicação de um STI. Além disso, dos trabalhos encontrados, nenhum apresentou uma modelagem e arquitetura descrita. Sendo assim, ficou claro a carência na atuação de modelagem e estudo de arquiteturas, considerando o módulo de comunicação na área de STIs.

Posteriormente à RSL, foram elaborados requisitos funcionais e os casos de uso para um módulo de comunicação, no Capítulo 4. Em seguida, os fluxos de processos do sistema, utilizando diagramas de atividade, na subseção 5.1.1 e, depois, a modelagem da interface e fluxo de usabilidade com a ferramenta IFML, na subseção 5.1.4. A partir da modelagem IFML, foram construídos os modelos visuais das telas do sistema, apresentados na subseção 5.1.5. A construção desses modelos visuais ocorreu de forma gradual, seguindo a sequência: *sketchs*, *wireframes*, *mockups* e, por fim, a implementação do protótipo.

Foi definido, então, uma arquitetura para um módulo de comunicação de um STI com padrão MVC. Nesta arquitetura, foi estruturado um banco de dados relacional e seu Dicionário de Dados, de acordo com os domínios citados, subseção 5.1.2. Na implementação, foi arquitetado o sistema particionando uma REST API na parte do *back-end* utilizando node.js e um *front-end web* utilizando React como framework.

O objetivo específico 1 foi atingido no capítulo 3, mediante a leitura dos artigos encontrados, utilizando o protocolo de pesquisa, apresentado na subseção 3.1.2. Concluiu-se que nenhum trabalho de STI, que apresenta uma modelagem detalhada completa foi encontrado, muito menos utilizando IFML como método de modelagem. Deixando claro, ser o módulo de comunicação/interface o menos trabalhado.

Os objetivos específicos 2 e 3, foram alcançados e apresentados no capitulo 5, com o desenvolvimento da plataforma Educa, disponibilizada atualmente no site de domínio

educaweb.herokuapp.com. Nota-se, assim que é possível modelar um fluxo de navegação utilizando uma técnica formal, como o IFML. Com o Educa desenvolvido, conclui-se também que é válido a utilização de uma arquitetura MVC com API REST e Web Clients.

Diante da resolução dos objetivos específicos, pode-se dizer que o objetivo geral de modelar uma arquitetura para o módulo de comunicação e interface do STI utilizando uma técnica formal, foi alcançado, apresentado a definição de uma arquitetura MVC na subseção 5.1 e utilizada como uma das modelagens formais, o IFML, apresentada na 5.1.4, no qual por meio da implementação, foi concluído que é possível implementar uma interface gráfica de um módulo de comunicação de um STI a partir da utilização de métodos formais de modelagens, utilizando IFML, assim, confirmando a hipótese apresentada neste trabalho.

6.2 Contribuições

Este trabalho contribuiu para a área acadêmica, ao se realizar uma revisão sistemática da literatura sobre STIs e IFML, bem como, para pesquisadores e indústrias da área educação e engenharia/modelagem de software.

Contribuiu também por apresentar um modelo de arquitetura de interface de um STI. De acordo com a Revisão Sistemática da Literatura realizada, o destaque principal é a própria modelagem de um STI utilizando IFML como ferramenta de modelagem, visto que é, senão o único, um dos poucos a unificar os dois assuntos. Além disso, o cuidado da pesquisa ser replicável foi considerando ao decorrer desse trabalho.

6.3 Trabalhos futuros

Como proposta para trabalhos futuros foram levantados os seguintes itens:

- Retirar as regras de negócios das camadas de modelos e controladores, colocando em uma camada específica;
- Implementar um controle de sugestões, o qual, a partir do histórico de navegação do usuário, irá sugerir qual tipo de atividade se adequaria melhor ao usuário, como videoaula ou leitura;
- Colocar uma camada do módulo de dados específica para registrar o histórico de navegação do usuário, está será utilizada para controle de sugestão e revisões;
- Implementar atividades do tipo de codificação;
- Realizar estudos de usabilidade com grupo de estudantes, recolhendo, assim, dados para melhorias, principalmente na área de *feedback* e reajuste do sistema.

Referências

ACERBIS, R. et al. Model-driven development based on omgs ifml with webratio web and mobile platform. In: SPRINGER. *International Conference on Web Engineering*. [S.l.], 2015. p. 605–608. Citado 2 vezes nas páginas 80 e 86.

AL-BASTAMI, B. G.; NASER, S. S. A. Design and development of an intelligent tutoring system for c# language. 2017. Citado 2 vezes nas páginas 79 e 80.

AREVALILLO-HERRÁEZ, M. et al. Gui-driven intelligent tutoring system with affective support to help learning the algebraic method. In: IEEE. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.], 2017. p. 2867–2872. Citado 2 vezes nas páginas 79 e 83.

BABICH, N. *Sketch, Wireframe, Mockup, and Prototype: Why, When and How*. 2020. Disponível em: <https://uxplanet.org/sketch-wireframe-mockup-and-prototype-why-when-and-how-29a25b3157c4>. Citado na página 48.

BANERES, D.; SAÍZ, J. Intelligent tutoring system for learning digital systems on mooc environments. In: IEEE. *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*. [S.l.], 2016. p. 95–102. Citado 2 vezes nas páginas 79 e 82.

BARBOSA. Implicações éticas do efeito mateus na ciência. *Revista de Ciências Sociais*, 2016. Citado na página 73.

BARCELONA, M. et al. Applying a model-based methodology to develop web-based systems of systems. *J. Web Eng.*, v. 16, n. 3&4, p. 212–227, 2017. Citado 2 vezes nas páginas 80 e 85.

BERNASCHINA, C.; COMAI, S.; FRATERNALI, P. Formal semantics of omgs interaction flow modeling language (ifml) for mobile and rich-client application model driven development. *Journal of Systems and Software*, Elsevier, v. 137, p. 239–260, 2018. Citado 2 vezes nas páginas 80 e 86.

BEZERRA, E. *Princıpios de Analise e Projeto de Sistemas com UML*. 3. ed. Elsevier, 2015. ISBN 978-85-352-2626-3. Disponível em: <http://eic.cefet-rj.br/papsuml3ed/index.html>. Citado 6 vezes nas páginas 38, 55, 56, 57, 58 e 59.

BLÁZQUEZ, J. S. *Una propuesta de automatización de operaciones CRUD en IFML*. Dissertação (B.S. thesis), 2018. Citado 2 vezes nas páginas 80 e 86.

BRAMBILLA, M.; FRATERNALI, P. *Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML*. [S.l.]: Morgan Kaufmann, 2014. Citado na página 41.

BRAMBILLA, M. et al. A model-based method for seamless web and mobile experience. In: ACM. *Proceedings of the 1st International Workshop on Mobile Development*. [S.l.], 2016. p. 33–40. Citado 2 vezes nas páginas 80 e 85.

BUTZ, C. J.; HUA, S.; MAGUIRE, R. B. A web-based bayesian intelligent tutoring system for computer programming. *Web Intelligence and Agent Systems: An International Journal*, IOS Press, v. 4, n. 1, p. 77–97, 2006. Citado na página 29.

CASARIN, N. V. Um panorama da modelagem de software web por empresas e instituições de ensino do sudoeste do paraná. Universidade Tecnológica Federal do Paraná, 2014. Citado 4 vezes nas páginas 37, 38, 40 e 41.

CRUZ, V. M. et al. *Diseño e Implementación de planificadores instruccionales en sistemas tutoriales inteligentes mediante el uso combinado de metodologías borrosa y multiagente*. [S.l.]: Universidad de La Laguna, Servicio de Publicaciones, 2009. Citado na página 29.

DORÇA, F. A. *Um sistema inteligente multiagente para educação a distância apoiado em web*. Tese (Doutorado) — Dissertação (mestrado em ciência da computação), Uberlândia, Brasil, 2004. Citado 2 vezes nas páginas 28 e 29.

EDUARDO, C. W. O.; HUGO, G. P. V. Development of example-tracing tutors for teaching control systems performance fundamentals. In: IEEE. *2014 4th World Congress on Information and Communication Technologies (WICT 2014)*. [S.l.], 2014. p. 290–295. Citado 2 vezes nas páginas 79 e 82.

ESPÍNDOLA, E. C. A importância do modelagem de objetos no desenvolvimento de sistemas. *Site:< http://www. linhadecodigo. com. br/artigo/1293/a-importancia-do-modelagemde-objetos-no-desenvolvimento-de-sistemas. aspx# ixzz4EnBeOMDJ>. Acessado em: julho de*, 2016. Citado na página 38.

FALZONE, E.; BERNASCHINA, C. Model based rapid prototyping and evolution of web application. In: SPRINGER. *International Conference on Web Engineering*. [S.l.], 2018. p. 496–500. Citado 2 vezes nas páginas 80 e 86.

FOWLER, D. G. A model for designing intelligent tutoring systems. *Journal of medical systems*, Springer, v. 15, n. 1, p. 47–63, 1991. Citado na página 27.

FREITAS, R. C.; DUTRA, M. de A. Usabilidade e interatividade em sistemas web para cursos online. *Brazilian Journal of Computers in Education*, v. 17, n. 02, p. 48, 2009. Citado 2 vezes nas páginas 30 e 31.

GALAFASSI, F. F. P. *Identificando conhecimentos a partir da aplicação das regras de dedução natural na lógica proposicional, um estudo prático*. [S.l.]: UFRGS, 2019. Citado 2 vezes nas páginas 79 e 80.

GOOGLE. *MVC Architecture*. 2021. Disponível em: <https://developer.chrome.com/docs/apps/app_frameworks/>. Citado 2 vezes nas páginas 65 e 66.

GRANIC, A.; STANKOV, S.; GLAVINIC, V. User interface aspects of an intelligent tutoring system. In: IEEE. *ITI 2000. Proceedings of the 22nd International Conference on Information Technology Interfaces (Cat. No. 00EX411)*. [S.l.], 2000. p. 157–164. Citado na página 76.

GURU99. *MVC Tutorial for Beginners: What is, Architecture Example*. 2021. Disponível em: <https://www.guru99.com/mvc-tutorial.html>. Citado na página 65.

HOOSHYAR, D. et al. A solution-based intelligent tutoring system integrated with an online game-based formative assessment: development and evaluation. *Educational Technology Research and Development*, Springer, v. 64, n. 4, p. 787–808, 2016. Citado 2 vezes nas páginas 79 e 83.

JUNIOR, G.; FREITAS, E. et al. Uma abordagem para planejamento instrucional apoiada em processos de decisão markovianos parcialmente observáveis. Universidade Federal de Uberlândia, 2018. Citado 3 vezes nas páginas 29, 79 e 84.

KHACHATRYAN, G. A. et al. Reasoning mind genie 2: An intelligent tutoring system as a vehicle for international transfer of instructional methods in mathematics. *International Journal of Artificial Intelligence in Education*, Springer, v. 24, n. 3, p. 333–382, 2014. Citado 2 vezes nas páginas 79 e 84.

KITCHENHAM, B.; BRERETON. *Evidence-based software engineering and systematic reviews*. [S.l.]: Chapman Hall/CRC, 2015. Citado 4 vezes nas páginas 68, 69, 70 e 79.

LAAZ, N.; MBARKI, S. Integrating ifml models and owl ontologies to derive uis web-apps. In: IEEE. *2016 International Conference on Information Technology for Organizations Development (IT4OD)*. [S.l.], 2016. p. 1–6. Citado 2 vezes nas páginas 80 e 85.

LAUREANO-CRUCES, A. L. et al. A pedagogical agent as an interface of an intelligent tutoring system to assist collaborative learning. *Creative Education*, Citeseer, v. 5, n. 08, p. 619, 2014. Citado 2 vezes nas páginas 79 e 82.

LUCIDCHART. *O que é um diagrama de sequência UML?* 2020. Disponível em: <https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-sequencia-uml>. Citado 3 vezes nas páginas 59, 60 e 61.

LUCIDCHART. *O que é um diagrama entidade relacionamento?* 2020. Disponível em: <https://www.lucidchart.com/pages/pt/o-que-e-diagrama-entidade-relacionamento>. Citado 2 vezes nas páginas 53 e 54.

MAHMOUD, M. H.; EL-HAMAYED, S. H. A. An intelligent tutoring system for teaching the grammar of the arabic language. *Journal of Electrical Systems and Information Technology*, Elsevier, v. 3, n. 2, p. 282–294, 2016. Citado 2 vezes nas páginas 79 e 83.

MENON, N. *Improving User Interface and User Experience of MathSpring Intelligent Tutoring System for Teachers*. Tese (Doutorado) — Worcester Polytechnic Institute, 2018. Citado 3 vezes nas páginas 75, 79 e 83.

MORAIS, F. de; SCHAAB, B.; JAQUES, P. The think aloud method for qualitative evaluation of an intelligent tutoring system interface. In: IEEE. *2017 Twelfth Latin American Conference on Learning Technologies (LACLO)*. [S.l.], 2017. p. 1–8. Citado 6 vezes nas páginas 28, 75, 79, 87, 88 e 89.

NGO, H. M.; TRAN, H. Q. Improving user interface and user experience of mathspring intelligent tutoring system for students. Worcester Polytechnic Institute, 2017. Citado 7 vezes nas páginas 75, 79, 87, 88, 89, 90 e 91.

NIELSEN, J. *Heuristic evaluation*. 1994. Disponível em: <https://www.nngroup.com/articles/ten-usability-heuristics>. Citado na página 31.

NIELSEN, J.; MACK, R. L. et al. *Usability inspection methods*. [S.l.]: Wiley New York, 1994. v. 1. Citado 2 vezes nas páginas 30 e 31.

OLIVEIRA, S. S. M. Sistema de informação para controle de materiais e doações aos bombeiros voluntários. 2018. Citado 6 vezes nas páginas 76, 80, 87, 90, 93 e 94.

OMG. *Objetc Management Group*. 2014. Disponível em: <https://www.omg.org/ifml>. Citado 3 vezes nas páginas 40, 41 e 42.

ORG. *Objetc Management Group*. 2018. Disponível em: <https://www.ifml.org>. Citado 4 vezes nas páginas 42, 43, 44 e 45.

QUEIROZ, R. et al. Evaluating usability of ifml models: How usability is perceived and propagated. In: ACM. *Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems*. [S.l.], 2018. p. 21. Citado 2 vezes nas páginas 80 e 85.

REIS, T. B. Fábio dos. *Modelagem de Dados - Conceitos de Bancos de Dados*. 2020. Disponível em: <https://www.youtube.com/watch?v=Q_KTYFgvu1s>. Citado na página 52.

REIS, T. B. Fábio dos. *Modelagem de Dados - Modelo Entidade-Relacionamento e Diagrama ER*. 2020. Disponível em: <https://www.youtube.com/watch?v=W2Z1STnjNJo>. Citado na página 53.

REKHAWI, H. A. A.; NASER, S. S. A. An intelligent tutoring system for learning android applications ui development. *International Journal of Engineering and Information Systems (IJEAIS)*, v. 2, n. 1, p. 1–14, 2018. Citado 3 vezes nas páginas 75, 79 e 82.

ROSSI, G. Web modeling languages strike back. *IEEE Internet Computing*, IEEE, v. 17, n. 4, p. 4–6, 2013. Citado 2 vezes nas páginas 39 e 40.

ROUBI, S.; ERRAMDANI, M.; MBARKI, S. Extending graphical part of the interaction flow modeling language to generate rich internet graphical user interfaces. In: IEEE. *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*. [S.l.], 2016. p. 161–167. Citado 2 vezes nas páginas 80 e 85.

SEFFRIN, H. M. Avaliando o conhecimento algébrico do estudante através de redes bayesianas dinâmicas: um estudo de caso com o sistema tutor inteligente pat2math. Universidade do Vale do Rio dos Sinos, 2015. Citado na página 28.

SEGUNDO, C. a. a. Pesquisa exploratória em elicitação de requisitos: teoria e prática. 2015. Citado na página 98.

SOTTILARE, R. A. *Fundamentals of adaptive intelligent tutoring systems for self-regulated learning*. [S.l.], 2015. Citado 4 vezes nas páginas 27, 28, 29 e 98.

SOUZA, M. V. de; GIGLIO, K. *Mídias Digitais, Redes Sociais e Educação em Rede: Experiências na Pesquisa e Extensão Universitária*. [S.l.]: Editora Blucher, 2015. Citado na página 31.

TACLA, C. A. *Análise e projeto OO & UML 2.0*. [S.l.]: UTFPR, 2007. Citado na página 38.

UML. *UML Activity Diagrams*. 2021. Disponível em: <https://www.uml-diagrams.org/activity-diagrams.html>. Citado 3 vezes nas páginas 62, 63 e 64.

VANLEHN, K. Reflections on andes goal-free user interface. *International Journal of Artificial Intelligence in Education*, Springer, v. 26, n. 1, p. 82–90, 2016. Citado 2 vezes nas páginas 79 e 83.

VERDÚ, E. et al. Intelligent tutoring interface for technology enhanced learning in a course of computer network design. In: IEEE. *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. [S.l.], 2014. p. 1–7. Citado 2 vezes nas páginas 79 e 82.

WAKIL, K.; JAWAWI, D. Analyzing interaction flow modeling language in web development lifecycle. *International Journal of Advanced Computer Science and Applications*, SCIENCE & INFORMATION SAI ORGANIZATION LTD 19 BOLLING RD, BRADFORD, WEST , v. 8, n. 1, p. 286–293, 2017. Citado 7 vezes nas páginas 76, 80, 85, 87, 88, 91 e 92.

WARCHOLINSKI, M. *What Is the Difference Between Wireframe, Mockup and Prototype?* 2019. Disponível em: <https://brainhub.eu/blog/difference-between-wireframe-mockup-prototype/>. Citado 2 vezes nas páginas 46 e 47.

WERAGAMA, D.; REYE, J. Analysing student programs in the php intelligent tutoring system. *International Journal of Artificial Intelligence in Education*, Springer, v. 24, n. 2, p. 162–188, 2014. Citado 2 vezes nas páginas 79 e 84.

WIMMER, M. et al. On the integration of web modeling languages: Preliminary results and future challenges. In: *Workshop on Model-driven Web Engineering (MDWE), held in conjunction with ICWE, Como, Italy*. [S.l.: s.n.], 2007. Citado na página 39.

WOOLF, B. P. *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. [S.l.]: Morgan Kaufmann, 2010. Citado 3 vezes nas páginas 24, 28 e 29.

Apêndices

APÊNDICE A – Levantamento de Requisitos

Este capítulo apresenta as histórias de usuários geradas no levantamento de requisitos das funcionalidades do Educa.

A.1 Registro e Login

Registrar no Sistema com Email
Como um usuário não cadastrado,
eu quero me cadastrar no sistema como Estudante, Instrutor ou Monitor com email e senha,
de modo que seja possível logar e utilizar o sistema de acordo com o tipo de usuário que me cadastrei.

Registrar no Sistema com a conta FaceBook
Como um usuário não cadastrado,
eu quero me cadastrar no sistema como Estudante, Instrutor ou Monitor com a conta Facebook,
de modo que seja possível logar e utilizar o sistema logando pela conta do FaceBook.

Registrar no Sistema com a conta Gmail
Como um usuário não cadastrado,
eu quero me cadastrar no sistema como Estudante, Instrutor ou Monitor com a conta Gmail,
de modo que seja possível logar e utilizar o sistema de acordo com a conta do Gmail.

Logar no Sistema
Como um usuário qualquer,
eu quero me logar no sistema informando email e senha,
de modo que seja possível eu usufruir do sistema como um usuário logado.

Trocar Senha
Como um usuário qualquer,
eu quero mudar minha senha,
de modo que seja possível eu logar no sistema somente com a nova senha registrada.

A.2 Curso

Cadastrar Curso
Como um usuário administrador,
eu quero cadastrar um novo curso,
de modo que seja possível vincular um instrutor possibilitando este de adicionar aulas e atividades.

Editar Curso
Como um usuário instrutor ou usuário administrador,
eu quero editar um curso,
de modo que seja possível editar os dados do curso e sequencia de aulas.

Deletar Curso
Como um usuário administrador,
eu quero remover ou indisponibilizar um curso,
de modo que nenhum usuário consiga acessar o curso que não seja administrador.

Prévisualizar Curso
Como um usuário não logado,
eu quero acessar o curso mesmo sem cadastro e sem fazer matrícula,
de modo que quando acessar um curso eu consiga visualizar o curso e visualizar o conteúdo das aulas e atividades.

Concluir Curso
Como um usuário estudante,
eu quero concluir um curso,
de modo que após concluir seja possivel acessar o certificado.

A.3 Aulas

Cadastrar Aula
Como um usuário administrador ou instrutor,
eu quero cadastrar aulas,
de modo que eu consiga registrar novas aulas para adicionar atividades futuramente.

Editar Aulas
Como um usuário administrador ou instrutor,
eu quero editar uma aula,

de modo que eu consiga editar corrigir alguma informação registrada erroneamente.

Remover Aulas
Como um usuário administrador ou instrutor,
eu quero remover uma aula,
de modo que nenhum usuário consiga acessar a aula que não seja administrador ou instrutor.

Concluir Aula
Como um usuario estudante,
eu quero concluir uma aula,
de modo que após concluir todas as atividades de uma aula, essa aula apreça como cluída, aumentando o percentual do progresso de um curso.

A.4 Atividades

Cadastrar Tipo de Atividade
Como um usuário administrador,
eu quero cadastrar tipos de atividades,
de modo que eu consiga registrar novos tipos de atividades.

Cadastrar Atividade
Como um usuário instrutor ou administrador,
eu quero cadastrar atividades,
de modo que o estudante possa realizar as atividades do curso.

Editar Atividade
Como um usuário instrutor ou administrador,
eu quero editar uma atividade,
de modo que eu consiga editarcorrigir alguma informação registrada erroneamente.

Deletar Atividade
Como um usuário instrutor ou administrador,
eu quero deletar ua atividade,
de modo que eu consiga editarcorrigir alguma informação registrada erroneamente.

Executar Atividade - Texto
Como um usuário estudante,
eu quero ler o texto de uma atividade texto,

de modo que após ler o texto e avançar para próxima atividade, e esta seja concluída.

Executar Atividade - Vídeo
Como um usuário estudante,
eu quero assistir a video aula de uma atividade do tipo vídeo,
de modo que após assistir o vídeo completo a atividade seja concluída automaticamente.

Executar Atividade - Questionário
Como um usuário estudante,
eu quero selecionar uma opção acreditando ser a certa e ser avaliado por isso,
de modo que quando submetar a resposta a atividade seja concluída e seja apresentado um feedback dizendo se a resposta selecionada é a correta, e se não, o porquê.

Visualizar justificativas do Questionário
Como um usuário estudante,
eu quero visualizar as justificativas de cada opção de após submeter a resposta,
de modo que seja apresentado a justificativa do porquê de estar certo ou errado de cada opção de resposta.

Compilar Código dá atividade
Como um usuário estudante,
eu quero compilar meu código,
de modo que caso esse seja compilado com sucesso e passado pela bateria de testes.

Executar Atividade - Código
Como um usuário estudante,
eu quero executar meu código,
de modo que caso esse seja compilado com sucesso e passado pela bateria de testes a atividade se torne concluida.

Concluir Atividade
Como um usuário estudante,
eu quero concluir uma atividade,
de modo que ao conclui-la selve nos meus progressos.

Salvar Progresso
Como um usuário estudante,
eu quero salvar o progresso que quando concluir uma atividade, aula ou curso,
de modo que quando eu queria continuar um curso, continue de onde parou.

A.5 Certificação

Certificar em um Curso
Como um usuário estudante,
eu quero me certificar em um curso,
de modo que após concluir totalente o curso, eu possa ter acesso a um certificado validável.

Acessar meus Certificados
Como um usuário estudante,
eu quero acessar meus certificados,
de modo que eu consiga listar os certificados que possuo.

Validar Certificados
Como um usuário qualquer,
eu quero verificar a validade de um certificado,
de modo que informando o numero de registro do certificado e o sistema apresentar o certificado confirmando a validade do mesmo.

Baixar Certificado
Como um usuário qualquer logado,
eu quero fazer o download de um certificado,
de modo que eu tenha um arquivo baixado no meu aparelho local

A.6 Usuários

Cadastrar um Administrador
Como um usuário administrador,
eu quero cadastrar outro usuário administrador,
de modo que outro usuário consiga usufruir do sistema como um administrador.

Listar Usuários
Como um administrador,
eu quero listar todos os usuários,
de modo que eu consiga posteriormente acessar dados de qualquer usuário do sistema.

Editar Usuário
Como um administrador,
eu quero editar um usuário,

de modo que o usuário tenha seus dados alterados após inserido e salvado a nova informação.

Desativar Usuário
Como um administrador,
eu quero desativar um usuário,
de modo que este usuário não consiga entrar no sistema ou seja considerado nas métricas estatísticas do sistema.

Notificar os usuários
Como um sistema,
eu quero notificar a todos os usuários sobre a mudança,
de modo que todos os usuários sejam notificados pelo email após alguma mudança na conta respectiva.

APÊNDICE B – Dicionário de Dados

Este capítulo apresenta os dicionário dos dados de acordo com cada domínio do sistema Educa.

B.1 Entidades

Apresenta o relacionamento entre as entidades do banco e a descrição destas.

B.1.1 Domínio Usuário

Tabela 15 – Dicionário de dados das tabelas do dominio Usuário.

Tabela	**Nome de Relacionamento**	**Relacionamento**	**Descrição**
User	Has	User_Type	Tabela para cadastro de usuário
	Is	Instructor	
	Is	Student	
	Is	Admin	
User_Type	Has	User	Tabela para cadastro de tipos de pessoa
Admin	is	User	Tabela para cadastro de Admin
Student	Is	User	Tabela para cadastro de estudante
	Enroll	Enrollment	
	Obtain	Certificate	
Instructor	Is	User	Tabela para cadastro de estudante
	Has	Teach	

Fonte: Elaborado pelo autor.

B.1.2 Domínios Curso, Aula e Ativividade

Tabela 16 – Dicionário de dados das tabelas dos domínios Curso, Aula e Ativividade.

Tabela	**Nome de Relacionamento**	**Relacionamento**	**Descrição**
Course	Has	Level	Tabela para cadastro de Cursos
	Taught	Teach	
	Has	Enrollment	
	Has	Lesson	
	Issue	Certificate	
Lesson	Has	Course	Tabela para cadastro de Aulas
	Has	Lesson_Progress	
	Has	Activity	
Activity	Has	Lesson	Tabela para cadastro de Atividades
	Has	Activity_progress	
	Has	Activity_Type	
	Is	Video	
	Is	Read	
	Is	Quiz	
Activity_Type	Has	Activity	Tabela para cadastro de tipos de atividades
Video	Is	Activity	Tabela para cadastro de video
Read	Is	Activity	Tabela para cadastro de texto
Code	Is	Activity	Tabela para cadastro de problema
Question	Is	Activity	Tabela para cadasdtro de perguntas
	Have	Answer	
Answer	Have	Question	Tabela para cadastro de respostas

Fonte: Elaborado pelo autor.

B.1.3 Domínio Matrícula

Tabela 17 – Dicionário de dados das tabelas do domínio matrícula.

Tabela	**Nome de Relacionamento**	**Relacionamento**	**Descrição**
Enrollment	Enroll	Student	Tabela para cadastro de inscrições de alunos em cursos
	Has	Course	
	Has	Lesson_Progress	
Lesson_Progress	Has	Lesson	Tabela para cadastro de progresso do estudante na aula
	Progress	Enrollment	
	Has	Activity_Progress	
Activity_Progress	Progress	Lesson_Progress	Tabela para cadastro de Atividades concluídas pelo Estudante
	Has	Activity	

Fonte: Elaborado pelo autor.

B.1.4 Domínios Certificação e Licenciatura

Tabela 18 – Dicionário de dados das tabelas dos domínios Certificação e Licenciatura.

Tabela	**Nome de Relacionamento**	**Relacionamento**	**Descrição**
Certificate	Obtain	Student	Tabela para cadastros de certificados de cursos obtidos por alunos
	Issue	Course	
Teach	Has	Instructor	Tabela para cadastros de certificados de cursos obtidos por alunos
	Taught	Course	

Fonte: Elaborado pelo autor.

B.2 Atributos

Descreve com detalhes cada atributos de cada tabela do banco.

B.2.1 User

Tabela 19 – Atributos da tabela user do domínio usuário.

Tabela User			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
	Inteiro	PK	
	Inteiro	FK	
	Caracter	NOT NULL, Maxlenght: 30	
	Caracter	NOT NULL, Maxlenght: 35	
user_url	Caracter	PK, NOT NULL, Maxlenght: 76	Código Identificador do usuário combinado o nome com o subrenome
	Caracter	PK, NOT NULL, Maxlenght: 256	
	Caracter	NOT NULL, Maxlenght: 60	
city	Caracter	Maxlenght: 80	Cidade do usuário
	Caracter	Maxlenght: 80	
	Caracter	Maxlenght: 360	
	Caracter	Maxlenght: 30	
	Caracter	Maxlenght: 30	
	Caracter	Maxlenght: 50	
linkedin	Caracter	Maxlenght: 50	Linkedin do usuário
github	Caracter	Maxlenght: 50	Github do usuário
create_at	Data	NOT NULL	Data de criação do usuário
updated_at	Data	NOT NULL	Data da última atualização do usuário
	Have	Answer	
Answer	Have	Question	Tabela para cadastro de respostas

Fonte: Elaborado pelo autor.

B.2.2 User Type

Tabela 20 – Atributos da tabela user_type do domínio usuário.

Tabela User_Type			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador do tipo de usuário
name	Caracter	NOT NULL, Maxlenght: 25	Nome do usuário
create_at	Data	NOT NULL	Data de criação do tipo do usuário
updated_at	Data	NOT NULL	Data da última atualização do tipo do usuário

Fonte: Elaborado pelo autor.

B.2.3 Instrutor

Tabela 21 – Atributos da tabela instrutor do domínio usuário.

Tabela Instructor			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador do Instrutor
user_id	Inteiro	NOT NULL	Identificador de usuário do Instrutor
create_at	Data	NOT NULL	Data de criação do tipo do Instrutor
updated_at	Data	NOT NULL	Data da última atualização do tipo do Instrutor

Fonte: Elaborado pelo autor.

B.2.4 Student

Tabela 22 – Atributos da tabela student do domínio usuário.

Tabela Student			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador do estudante
user_id	Inteiro	NOT NULL	Identificador de usuário do estudante
create_at	Data	NOT NULL	Data de criação do tipo do estudante
updated_at	Data	NOT NULL	Data da última atualização do tipo do estudante

Fonte: Elaborado pelo autor.

B.2.5 Lecionar

Tabela 23 – Atributos da tabela teach do domínio lecionar.

Tabela Teach			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador do ensino
instructor_id	Inteiro	NOT NULL	Identificador de usuário do estudante
course_id	Inteiro	NOT NULL	Identificador do curso a ser lecionado
create_at	Data	NOT NULL	Data de criação do ensino
updated_at	Data	NOT NULL	Data da última atualização do ensino

Fonte: Elaborado pelo autor.

B.2.6 Course

Tabela 24 – Atributos da tabela course do domínio curso.

Tabela Course			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador do curso
name	Caracter	NOT NULL, Maxlenght: 120	Nome do curso
identifier	Caracter	NOT NULL, UNIQUE, Maxlenght: 40	Nome identificador do curso
description	Caracter	NOT NULL, Maxlenght: 1000	Descrição do curso
active	Boleano	NOT NULL, DEFAULT: false	Campo de identificação se o curso está ativo ou não
level_id	Inteiro	NOT NULL	Código de identificação da dificuldade do curso (Básico, Moderado, Avançado)
create_at	Data	NOT NULL	Data de criação do curso
updated_at	Data	NOT NULL	Data da última atualização do curso

Fonte: Elaborado pelo autor.

B.2.7 Level

Tabela 25 – Atributos da tabela level do domínio curso.

Tabela Level			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador da dificuldade de curso
name	Caracter	NOT NULL, UNIQUE, Maxlenght: 11	Nome da dificuldade de cursos
create_at	Data	NOT NULL	Data de criação do level de dificuldade de cursos
updated_at	Data	NOT NULL	Data da última atualização do level da dificuldade de cursos

Fonte: Elaborado pelo autor.

B.2.8 Lesson

Tabela 26 – Atributos da tabela lesson do domínio curso.

Tabela Lesson			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador da aula
name	Caracter	NOT NULL, Maxlenght: 120	Nome da aula
active	Boleano	NOT NULL, DEFAULT: false	Idetificador se a aula esta ativa
duration	Inteiro	NOT NULL, DEFAULT: 0	Duração da aula em minutos
course_id	Integer	NOT NULL	Campo de identificação do curso esta aula pertence
create_at	Data	NOT NULL	Data de criação da aula
updated_at	Data	NOT NULL	Data da última atualização da aula

Fonte: Elaborado pelo autor.

B.2.9 Activity Type

Tabela 27 – Atributos da tabela activity type do domínio curso.

Tabela Activity Type			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador do tipo de atividade
name	Caracter	NOT NULL, Maxlenght: 15	Nome do tipo de atividade
create_at	Data	NOT NULL	Data de criação do tipo de atividade
updated_at	Data	NOT NULL	Data da última atualização do tipo de atividade

Fonte: Elaborado pelo autor.

B.2.10 Activity

Tabela 28 – Atributos da tabela activity do domínio curso.

Tabela Activity			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador da atividade
name	Caracter	NOT NULL, Maxlenght: 120	Nome da atividade
active	Boleano	NOT NULL, DEFAULT: false	Identificador se a atividade está ativa
type_id	Inteiro	NOT NULL, DEFAULT: 0	Identificador de qual tipo a aula é (Read, Video, Question)
lesson_id	Integer	NOT NULL	Campo de identificação de qual aula esta atividade pertence
create_at	Data	NOT NULL	Data de criação da atividade
updated_at	Data	NOT NULL	Data da última atualização da aula

Fonte: Elaborado pelo autor.

B.2.11 Read

Tabela 29 – Atributos da tabela read do domínio curso.

Tabela Read			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador da atividade de leitura
text	Caracter	NOT NULL, Maxlenght: 3000	Texto da atividade de leitura
activity_id	Inteiro	NOT NULL	Identificador da atividade relacionada
create_at	Data	NOT NULL	Data de criação do conteudo de leitura
updated_at	Data	NOT NULL	Data da última atualização do conteudo de leitura

Fonte: Elaborado pelo autor.

B.2.12 Video

Tabela 30 – Atributos da tabela vídeo do domínio curso.

Tabela Video			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador da atividade de video
text	Caracter	NOT NULL, Maxlenght: 3000	Texto da atividade de video
url	Caracter	NOT NULL, Maxlenght: 500	Url do vídeo da atividade
duration	Inteiro	NOT NULL	Tempo de duração do vídeo da atividade
activity_id	Inteiro	NOT NULL	Identificador da atividade relacionada
create_at	Data	NOT NULL	Data de criação do conteudo de video
updated_at	Data	NOT NULL	Data da última atualização do conteudo de video

Fonte: Elaborado pelo autor.

B.2.13 Code

Tabela 31 – Atributos da tabela code do domínio curso.

Tabela Code			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador da atividade de codificação
text	Caracter	NOT NULL, Maxlenght: 3000	Texto da descrição do problema a ser resolvido
return	Data	NOT NULL, Maxlenght: 4000	Data de criação do conteudo de leitura
activity_id	Inteiro	NOT NULL	Identificador da atividade relacionada
create_at	Data	NOT NULL	Data de criação do conteúdo
updated_at	Data	NOT NULL	Data da última atualização do conteudo

Fonte: Elaborado pelo autor.

B.2.14 Question

Tabela 32 – Atributos da tabela question do domínio curso.

Tabela Question			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador da atividade de questionário
text	Caracter	NOT NULL, Maxlenght: 2000	Texto da questionário
activity_id	Inteiro	NOT NULL	Identificador da atividade relacionada
create_at	Data	NOT NULL	Data de criação do conteudo de questionário
updated_at	Data	NOT NULL	Data da última atualização do questionário

Fonte: Elaborado pelo autor.

B.2.15 Answer

Tabela 33 – Atributos da tabela Answer do domínio curso.

Tabela Answer			
Atributo	**Tipo de Dados**	**Restrições**	**Descrição**
id	Inteiro	PK, NOT NULL	Numero identificador da resposta
text	Caracter	NOT NULL, Maxlenght: 500	Texto da resposta
correct	Boleano	NOT NULL DEFAULT: false	Identificador se a resposta é correta ou não
justification	Caracter	NOT NULL, Maxlenght: 1000	Justificativa da resposta, explicando caso a resposta seja correta ou não
question_id	Inteiro	NOT NULL	Identificador do questionário relacionado
create_at	Data	NOT NULL	Data de criação da resposta
updated_at	Data	NOT NULL	Data da última atualização da resposta

Fonte: Elaborado pelo autor.

APÊNDICE C – Rotas e Recursos da API

Este capítulo apresenta todos as rotas e recursos da API do sistema Educa.

C.1 UserTypeController

Tabela 34 – Rotas do recurso de User da API do sistema.

Verbo	Rota	Controlador	Descrição
POST	"/user_type"	UserType.create	registra um novo tipo de usuárioinformando obrigatóriamenteo nome do tipo de usuário.
GET	"/user_type"	UserType.index	lista todos os tipos de usuários (Estudante, Instrutor, Adminsitrador e Monitor);
GET	"/user_type/:id"	UserType.show	consulta os dados de um tipo de usuário especifico;
PUT	"/user_type/:id"	UserType.update	atualiza um tipo de usuário existente respeitando a mesma validação do cadastro (create);
DELETE	"/user_type/:id"	UserType.delete	deleta um tipo de usuário após informado o identificador do tipo de usuário correspondente.

Tabela 35 – Rotas do recurso de User Type da API do sistema.

Fonte: Elaborado pelo autor.

C.2 UserController

Verbo	**Rota**	**Controlador**	**Descrição**
POST	"/register"	User.register	registra um novo usuário informando obrigatiamente: nome, sobrenome, email, senha, e tipo de usuário. É registrado com sucesso caso não exista outro usuário de mesmo email, nome e sobrenome;
GET	"/users"	User.index	lista todos os usuários;
POST	"/login"	User.login	loga ao sistema informando um conjunto de email e senha válido;
GET	"/users/:user_url"	User.show	consulta todos os dados de um usuário especifico informando o identificador deste;
PUT	"/users/:user_url"	User.update	atualiza o usuário equivalente ao identificado informado respeitando a mesma validação do cadastro (register);
DELETE	"/users/:user_url"	User.delete	deleta o usuário após informado o identificador do usuário correspondente.

Fonte: Elaborado pelo autor.

C.3 LevelController

Tabela 36 – Rotas do recurso de Level da API do sistema.

Verbo	**Rota**	**Controlador**	**Descrição**
POST	"/level"	Level.create	registra um novo level de curso informando obrigatóriamente o nome do level. É registrado com sucesso caso não exista outro de mesmo nome;
GET	"/level"	Level.index	lista todos os leveis de cursos cadastrados (Básico, Moderado, Avançado);
PUT	"/level/:id"	Level.update	atualiza um level existente respeitando a mesma validação do cadastro (create);
DELETE	"/level/:id"	Level.delete	deleta um level após informado o identificador do level correspondente.
PUT	"/users/:user_url"	User.update	atualiza o usuário equivalente ao identificado informado respeitando a mesma validação do cadastro (register);
DELETE	"/users/:user_url"	User.delete	deleta o usuário após informado o identificador do usuário correspondente.

Fonte: Elaborado pelo autor.

C.4 CourseController

Tabela 37 – Rotas do recurso de Course da API do sistema.

Verbo	Rota	Controlador	Descrição
GET	"/course"	Course.index	lista todos os cursos cadastrados;
GET	"/course/:identifier"	Course.show	consulta os dados de um curso especifico informando o identificador (tag) do curso;
POST	"/admin/:admin_id/ course"	Course.create	registra um novo curso informando obrigatoriamente o nome, identificador(tag do curso), descrição, o id do administrador que esta criando, o level correspondente, a duração em minutos e os instrutores correspondentes. É registrado com sucesso caso não exista outro de mesmo identificador (tag);
PUT	"/update_course/ :user_id/course/ :identifier"	Course.update	atualiza um curso existente respeitando a mesma validação do cadastro (create);
DELETE	"/admin/:admin_id/ course/:identifier"	Course.delete	deleta um curso após informado o identificador (tag) do curso correspondente. Neste caso também é deletado todas as aulas e atividades relacionadas a este curso.

Fonte: Elaborado pelo autor.

C.5 LessonController

Tabela 38 – Rotas do recurso de Lesson da API do sistema.

Verbo	Rota	Controlador	Descrição
POST	"/instructor/ :instructor_id/course/ :identifier/lesson"	Lesson.create	registra uma nova aula informando obrigatoriamente o nome, identificador (tag) do curso relacionado, duração em minutos, e o id do instrutor responsável pelo curso. É registrado com sucesso caso, exista o curso e instrutores relacionados e também não exista outra aula de mesmo nome relacionado ao mesmo curso;
GET	"/course/:identifier/lesson"	Lesson.index	lista todos as aulas cadastradas de um curso especifico informando a tag do curso;
GET	"/course/:identifier/ lesson/:lesson_id"	Lesson.show	consulta os dados de uma aula especifica informando o identificador (tag) do curso e o id da aula;
PUT	"/instructor/ :instructor_id/course/ :identifier/lesson/:id"	Lesson.update	atualiza uma aula existente respeitando as mesmas validações do cadastro (create);
DELETE	"/instructor/ :instructor_id/course/ :identifier/lesson/:id"	Lesson.delete	deleta uma aula existente após informado o id da aula, identificador (tag) do curso e o id do instrutor responsável correspondente. Neste caso também é deletado todas as atividades relacionadas a este curso.

Fonte: Elaborado pelo autor.

C.6 ActivityTypeController

Tabela 39 – Rotas do recurso de Activity Type da API do sistema.

Verbo	**Rota**	**Controlador**	**Descrição**
POST	"/activity_type"	ActivityType.create	registra um novo tipo de atividade informando obrigatoriamente o nome. É registrado com sucesso caso não exista outra atividade de mesmo nome;
GET	"/activity_type"	ActivityType.index	lista todos os tipos de atividades cadastrados (Texto, Vídeo, Questionário, Código);
GET	"/activity_type/:id"	ActivityType.show	consulta os dados do tipo de um tipo de atividade informando o id;
PUT	"/activity_type/:id"	ActivityType.update	atualiza um tipo de atividade existente respeitando a mesma validação do cadastro (create);
DELETE	"/activity_type/:id"	ActivityType.delete	deleta um tipo de atividade após informado o identificador do tipo de atividade correspondente.

Fonte: Elaborado pelo autor.

C.7 ActivityController

Tabela 40 – Rotas do recurso de Activity da API do sistema.

Verbo	**Rota**	**Controlador**	**Descrição**
POST	"/course/:identifier/ lesson/:lesson_id/ activity"	Activity.create	registra uma nova atividade informando obrigatoriamente o nome, id do tipo da aula, identificador (tag) do curso relacionado, id da aula, e o id do instrutor responsável pelo curso e o conteúdo desta aula. É registrado com sucesso caso, exista o curso, aula e instrutores relacionados e também não exista outra atividade de mesmo nome relacionado ao mesmo curso;
GET	"/course/:identifier/ lesson/:lesson_id/ activity"	Activity.index	lista todos as atividades cadastradas de uma aula especifica informada pelo o id da aula;
GET	"/course/:identifier/ lesson/:lesson_id/ activity/:id"	Activity.show	consulta os dados de uma atividade especifica informando os identificadores das atividade e aula da aula;
PUT	"/course/:identifier/ lesson/:lesson_id/ activity/:id"	Activity.update	atualiza uma atividade existente respeitando as mesmas validações do cadastro (create);
DELETE	"/course/:identifier/ lesson/:lesson_id/ activity/:id"	Activity.delete	deleta uma atividade existente após informado o id da atividade, o id da aula, identificador (tag) do curso e o id do instrutor responsável correspondente.

Fonte: Elaborado pelo autor.

C.8 CertificateController

Tabela 41 – Rotas do recurso de Certificate da API do sistema.

Verbo	**Rota**	**Controlador**	**Descrição**
GET	"/certificates/course/ :course_id/student/ :student_id/"	Certificate.index	registra uma novo certificado informando o id do usuário e o identificador (tag) do curso;
GET	"/certificates"	Certificate.index	lista todos os certificados registrados no sistema;
GET	"/certificate"	Certificate.show	consulta os dados de um certificado informando o id ou hash do certificado;
GET	"/student/:student_id/ certificate"	Certificate.student	lista todos os certificados que um estudante conquistou informando o id do estudante;
GET	"/course/:identifier/ certificate"	Certificate.course	lista todos os certificados vinculados a um curso informando o identificador (tag) do curso.

Fonte: Elaborado pelo autor.

C.9 Enrollment

Tabela 42 – Rotas do recurso de Enrollment da API do sistema.

Verbo	**Rota**	**Controlador**	**Descrição**
POST	"/student/:student_id/ course/:identifier/ register"	Enrollment.register	registra a matrícula do estudante com um curso. É registrado com sucesso caso o estudante e o curso existam e não exista ja uma matricula entre este estudante com este curso.
GET	"/student/:student_id/ course/:identifier/ enrollment"	Enrollment.show	apresenta os dados da matricula do estudante com o curso, seu progresso e a nota de avaliação que o estudante deu ao curso.
GET	"/student/:student_id/ course"	Enrollment.list	lista todas matriculas, caso exista, de um estudante especifico. É listado com sucesso caso o estudante informado exista no sistema.
POST	"/student/:student_id/ course/:identifier/rate"	Enrollment.rate	atribui a nota que o estudante avaliou o curso. É executado com sucesso caso o estudante e o curso existam. Caso o estudante mude o valor da nota, é substituído a anterior.
POST	"/student/:student_id/ course/:identifier/ continue"	Enrollment.continue	informa qual a última atividade realizada pelo estudante daquele curso, auxiliando saber qual a próxima esse estudante deve fazer. Caso o estudante não tenha realizado ainda nenhuma atividade, é informado que a continuação deve ser a partir da primeira atividade da primeira aula do curso. Estudante, curso e matricula devem existir no sistema.

Fonte: Elaborado pelo autor.

C.10 LessonProgress

Tabela 43 – Rotas do recurso de Lesson Progress da API do sistema.

Verbo	**Rota**	**Controlador**	**Descrição**
POST	"/student/:student_id/ course/:identifier/ lesson/:lesson_id/ progress"	LessonProgress.register	registra um vinculo do estudante com a aula iniciada informando o id do estudante, o identificador do curso, o id da aula. O registro é realizado com sucesso considerando que o curso, a aula e o estudante existam como também a matricula do estudante com o curso relacionado a aula;
GET	"/student/:student_id/ course/:identifier/ lesson/:lesson_id/ progress"	LessonProgress.list	lista todos os progressos de atividades relacionados ao progresso da aula e ao estudante informado. A listagem é realizada com sucesso quando o estudante e a aula existam como também a matrícula do estudante com o curso.

Fonte: Elaborado pelo autor.

C.11 ActicityProgress

Tabela 44 – Rotas do recurso de Acticity Progress da API do sistema.

Verbo	**Rota**	**Controlador**	**Descrição**
POST	"/student/:student_id/ course/:identifier /lesson/:lesson_id/ activity/:id/progress"	ActicityProgress.register	registra um vinculo do estudante com a atividade iniciada informando o id do estudante, o identificador do curso, o id da aula e o id da atividade. O registro é realizado com sucesso considerando o estudante e a atividade existam e que também exista uma matrícula do estudante com o curso;
PUT	"/student/:student_id/ course/:identifier/ lesson/:lesson_id/ activity/:id/progress"	ActicityProgress.complete	atualiza o status do progresso do estudante com uma atividade para completado. A atualização é realizada com sucesso considerando que a atividade já não esteja completada e que o estudante, curso, aula e atividade existam.

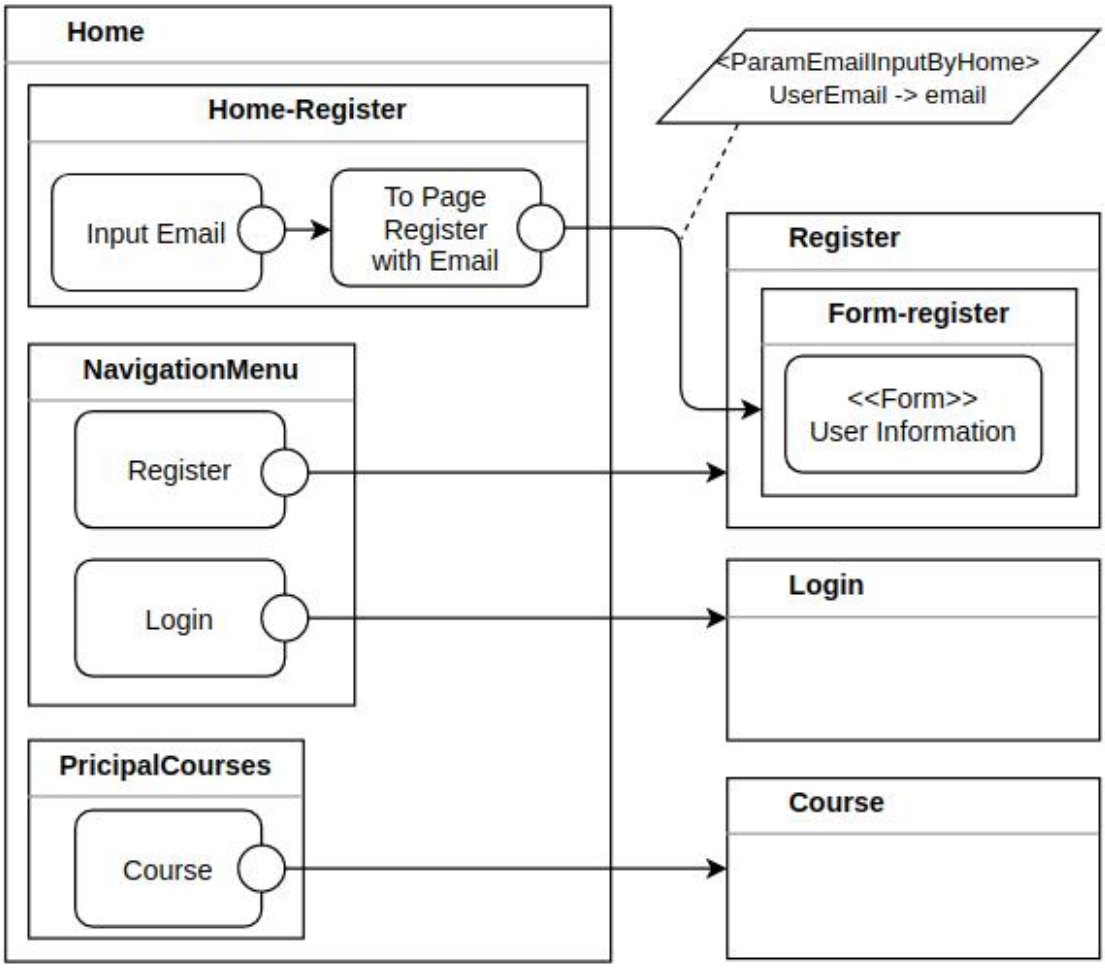
Fonte: Elaborado pelo autor.

APÊNDICE D – Modelos Design

Este capítulo apresenta todos os modelos criados para a esquematização visual do sistema Educa. Estes estão organizados separados em conjuntos, sendo eles: IFML, Sketch, Wireframe, Mockup e Protótipo.

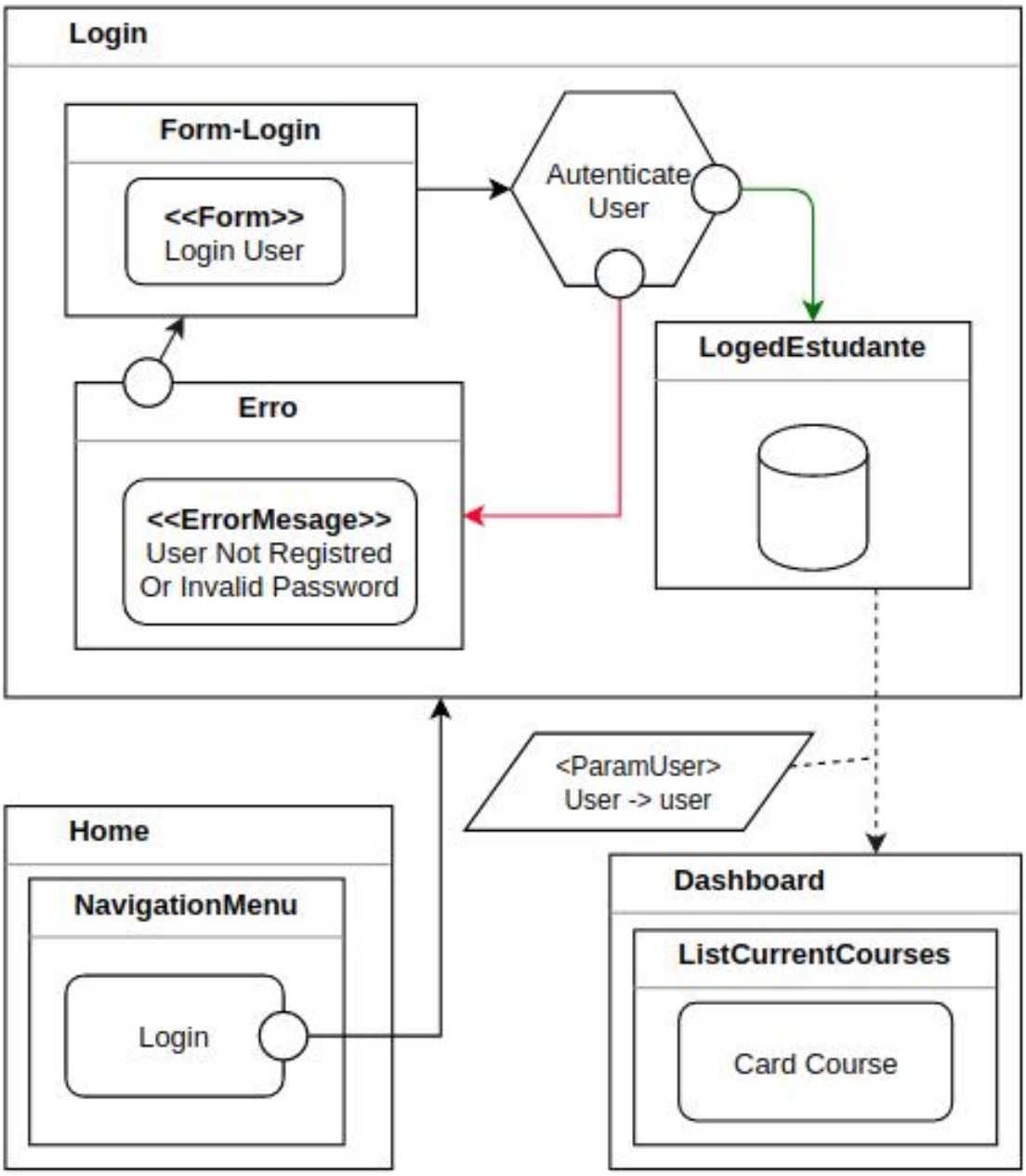
D.1 IFML

Figura 127 – Modelo IFML fluxo da tela Home



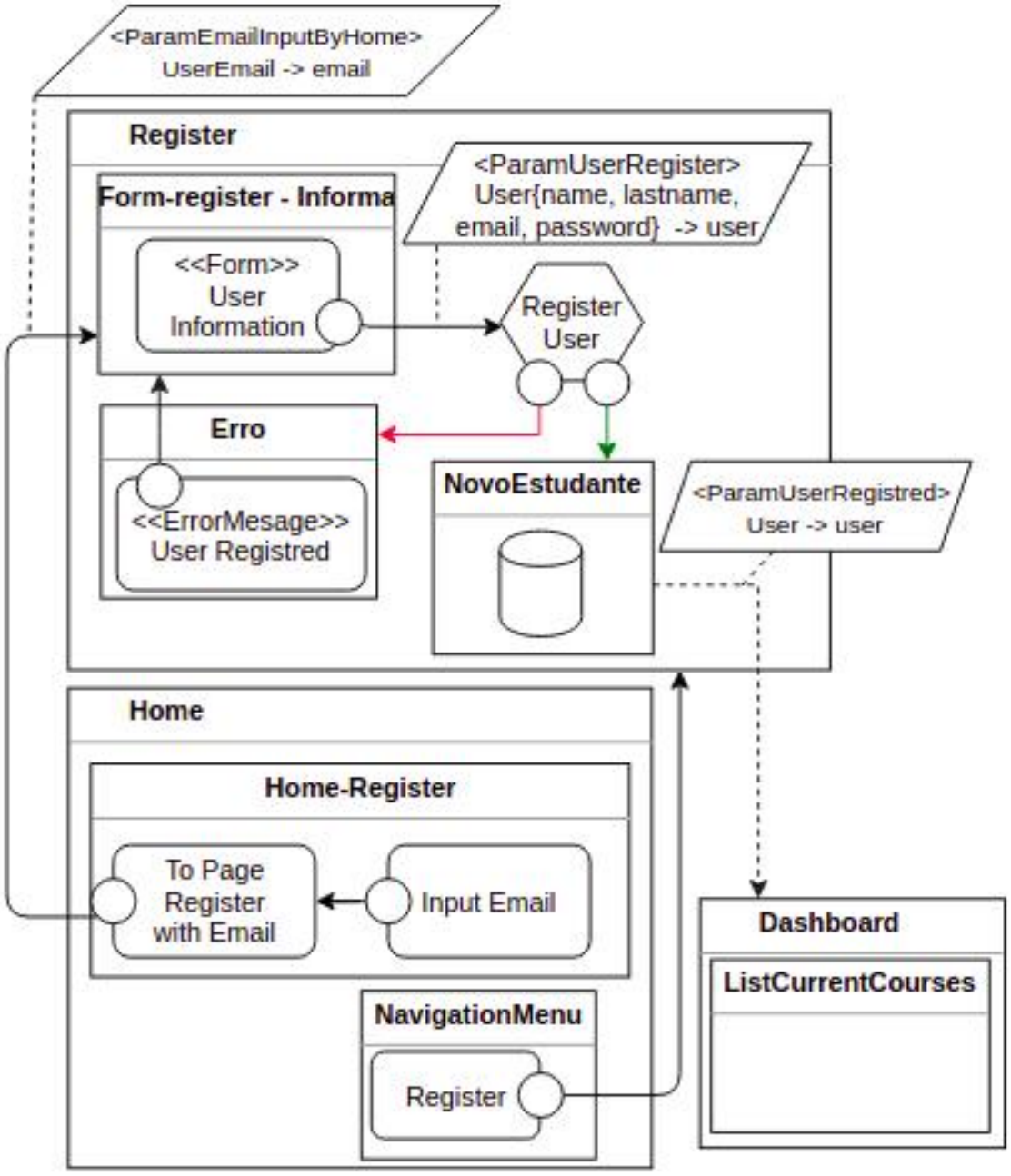
Fonte: Elaborado pelo autor.

Figura 128 – Modelo IFML fluxo de login



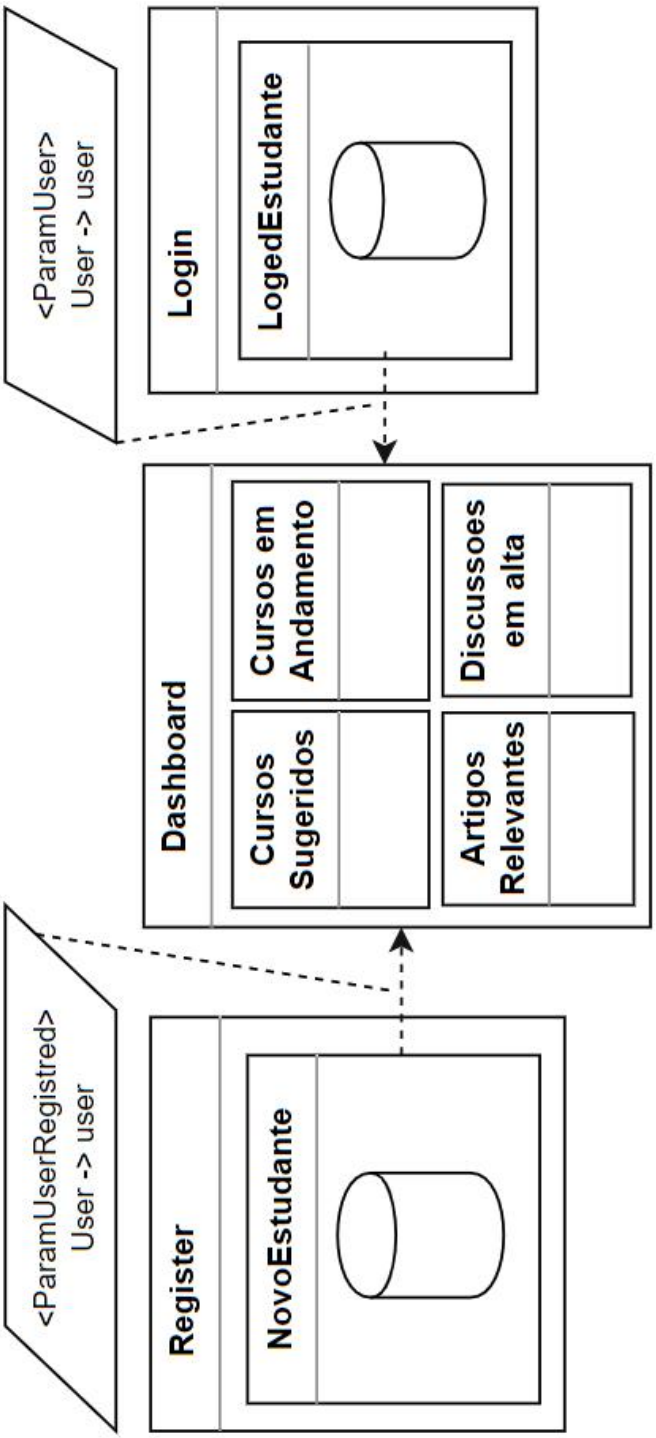
Fonte: Elaborado pelo autor.

Figura 129 – Modelo IFML fluxo da tela Registro



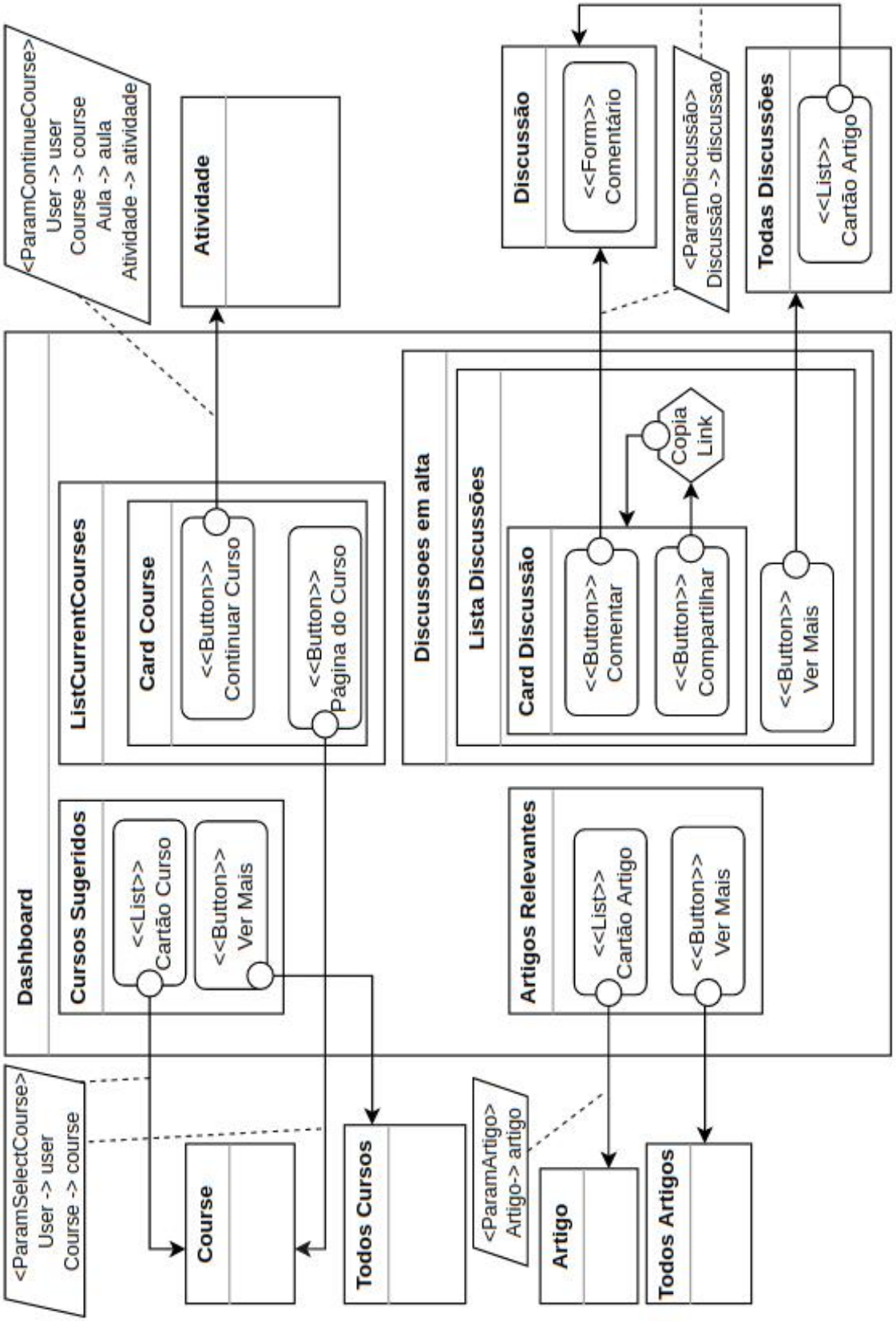
Fonte: Elaborado pelo autor.

Figura 130 – Modelo IFML fluxo para chegar na tela de Dashboard



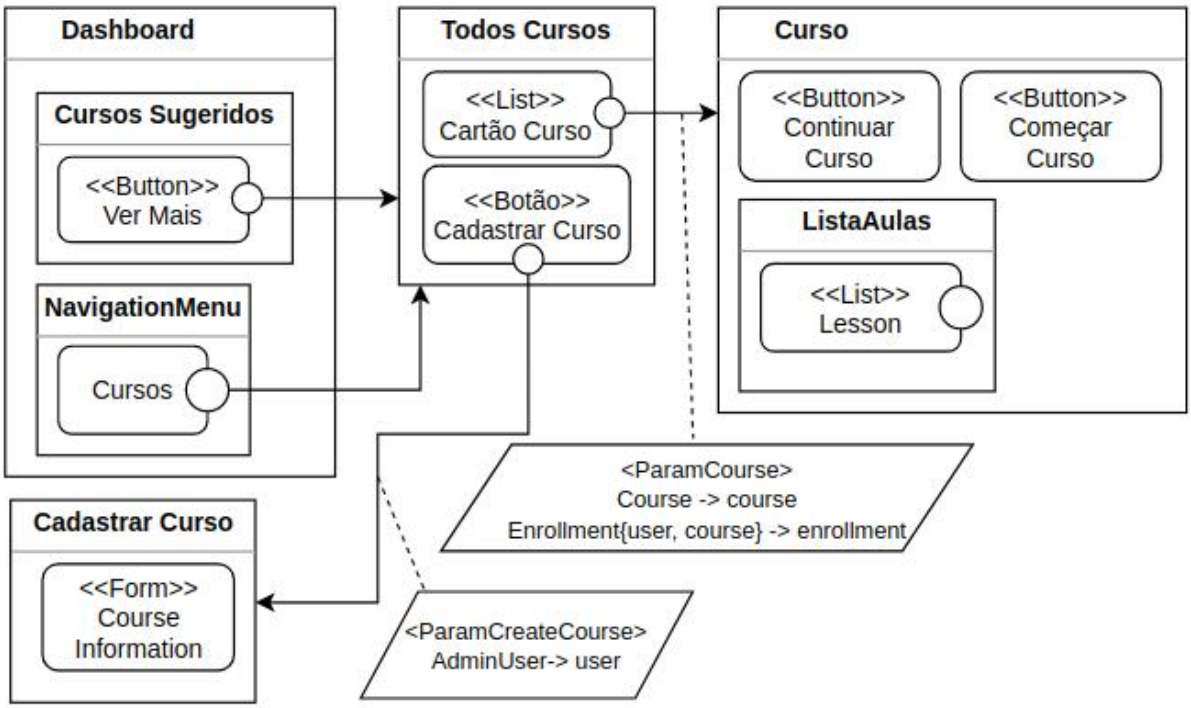
Fonte: Elaborado pelo autor.

Figura 131 – Modelo IFML fluxo da tela Dashboard



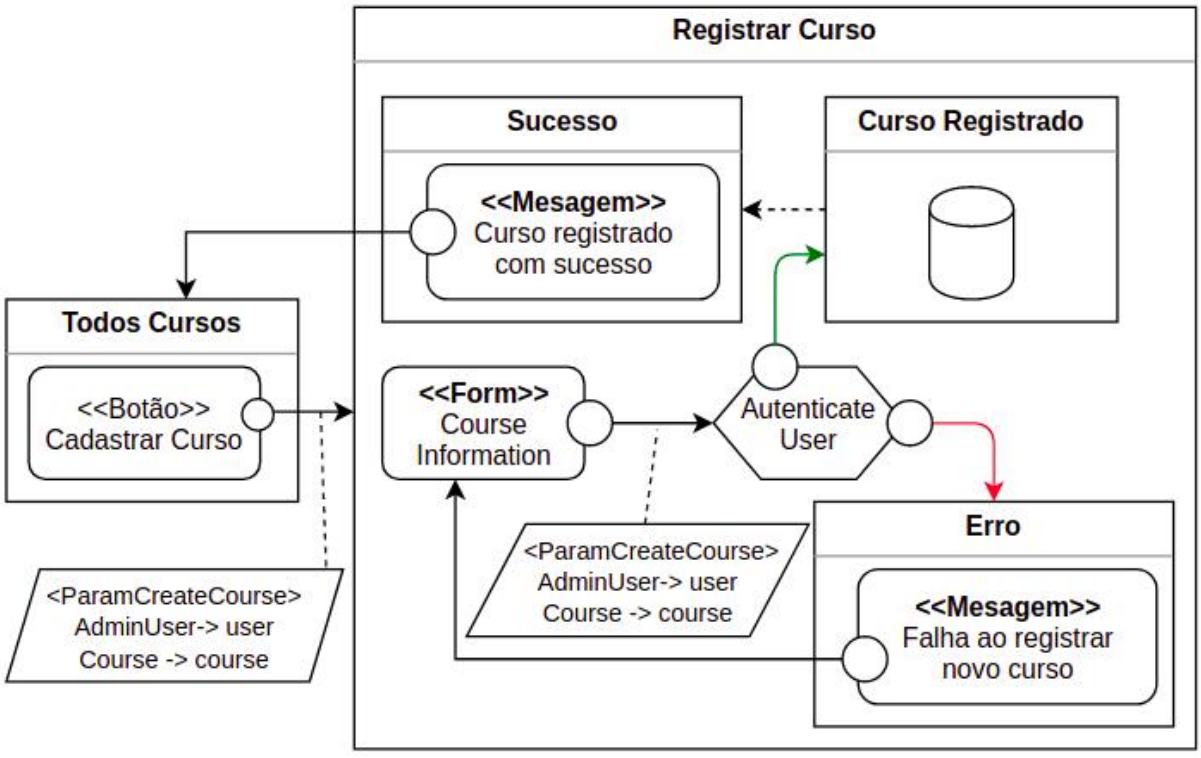
Fonte: Elaborado pelo autor.

Figura 132 – Modelo IFML do fluxo tela catalago de cursos



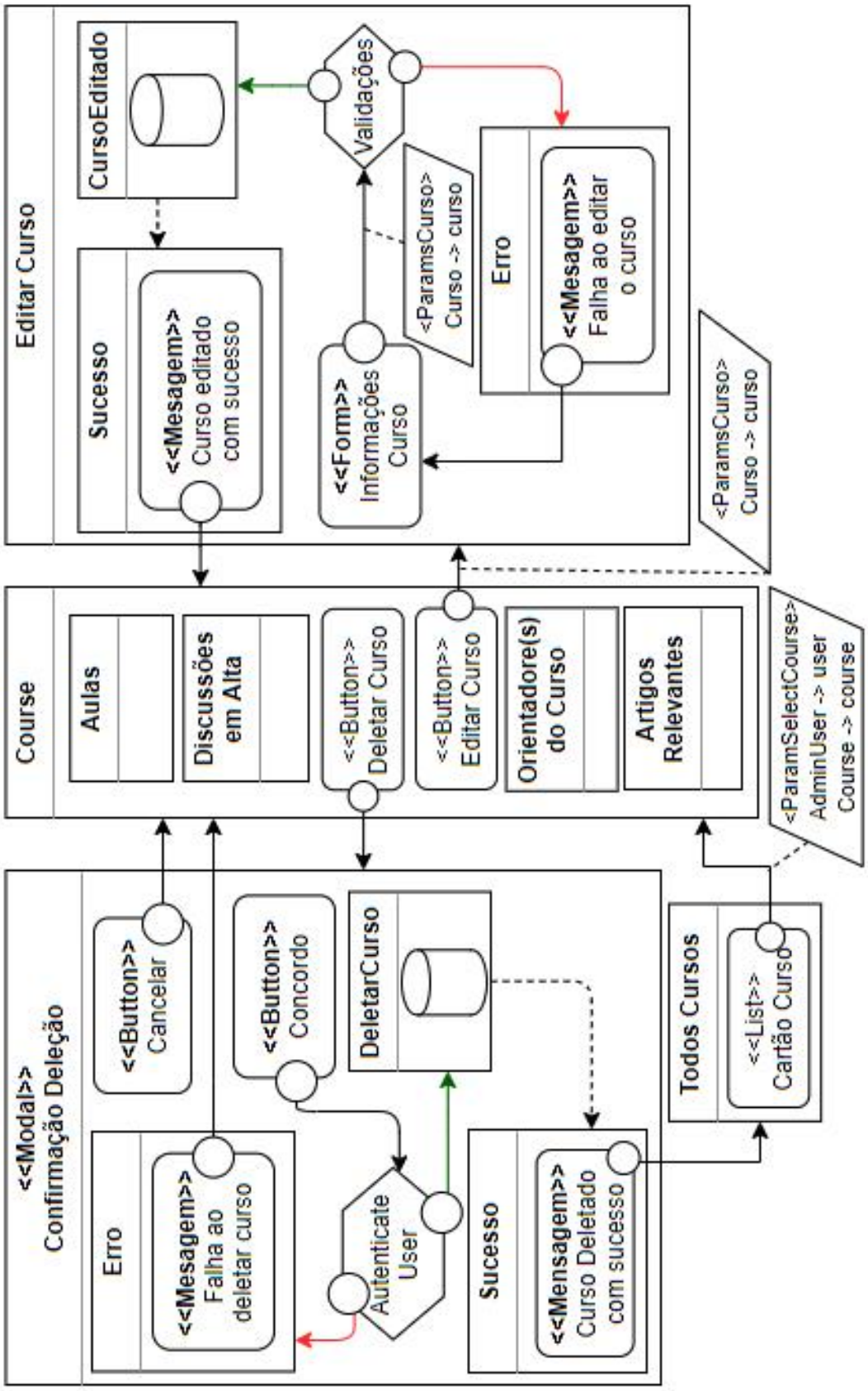
Fonte: Elaborado pelo autor.

Figura 133 – Modelo IFML do fluxo de registrar cursos



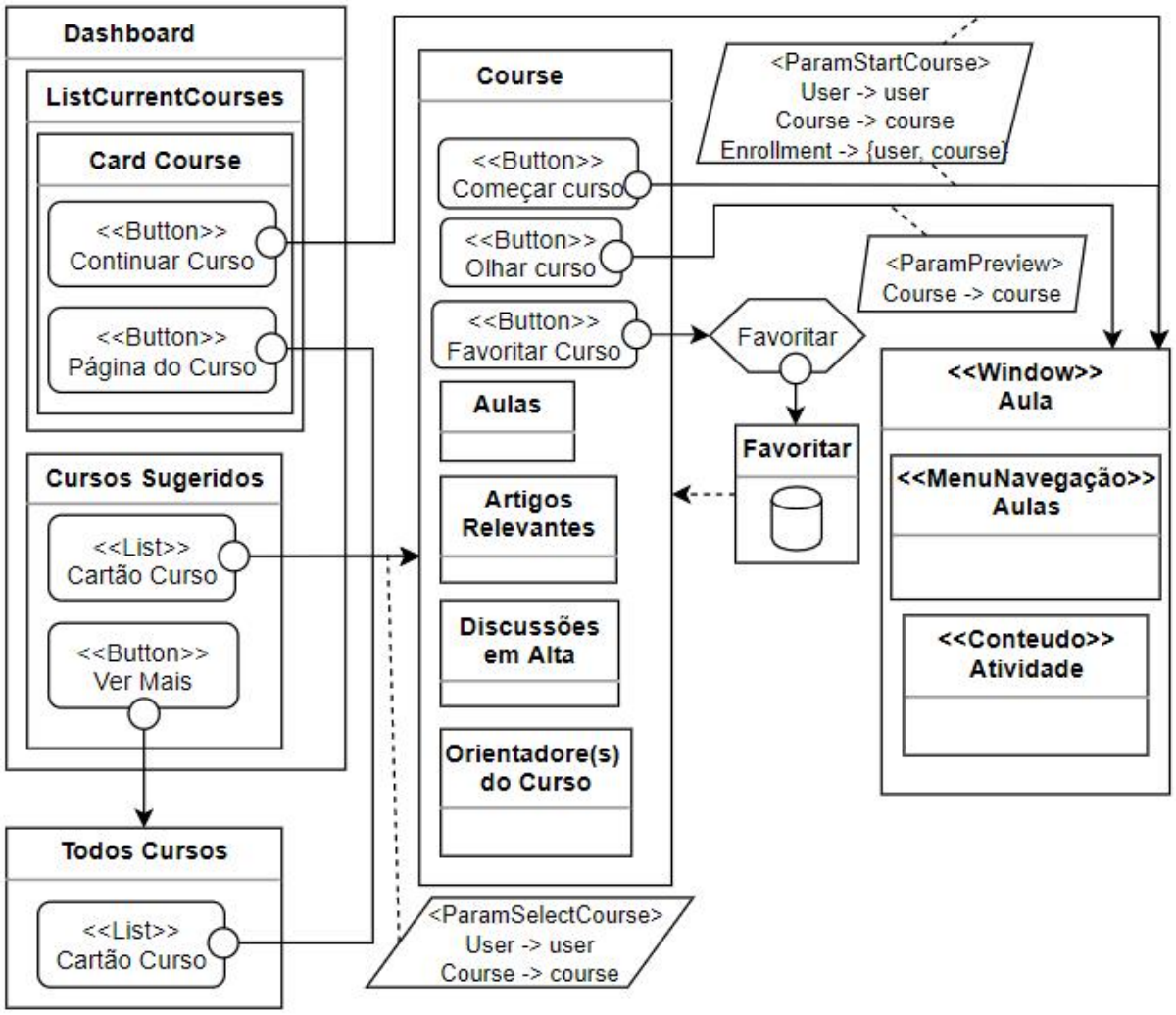
Fonte: Elaborado pelo autor.

Figura 134 – Modelo IFML do fluxo de editar e deletar cursos



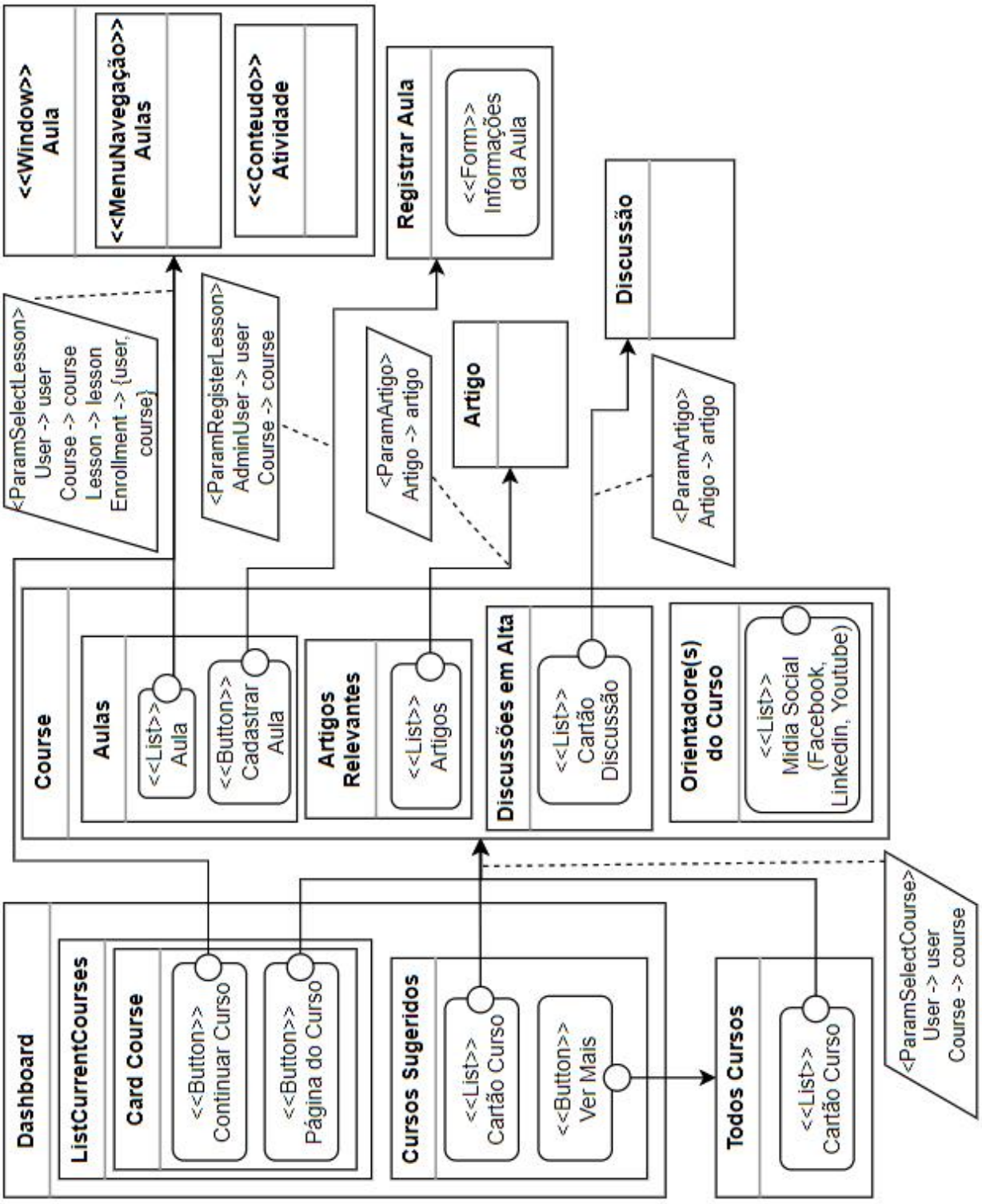
Fonte: Elaborado pelo autor.

Figura 135 – Modelo IFML do fluxo dos botões da tela do curso



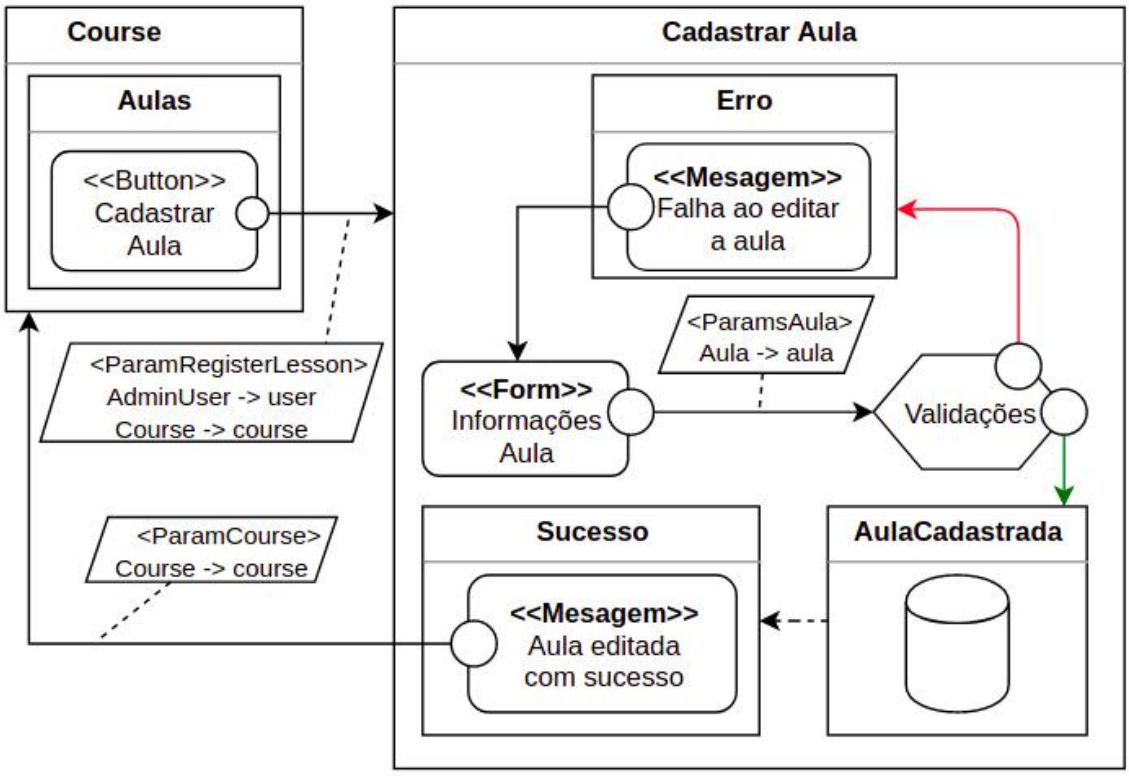
Fonte: Elaborado pelo autor.

Figura 136 – Modelo IFML do fluxo da tela de curso



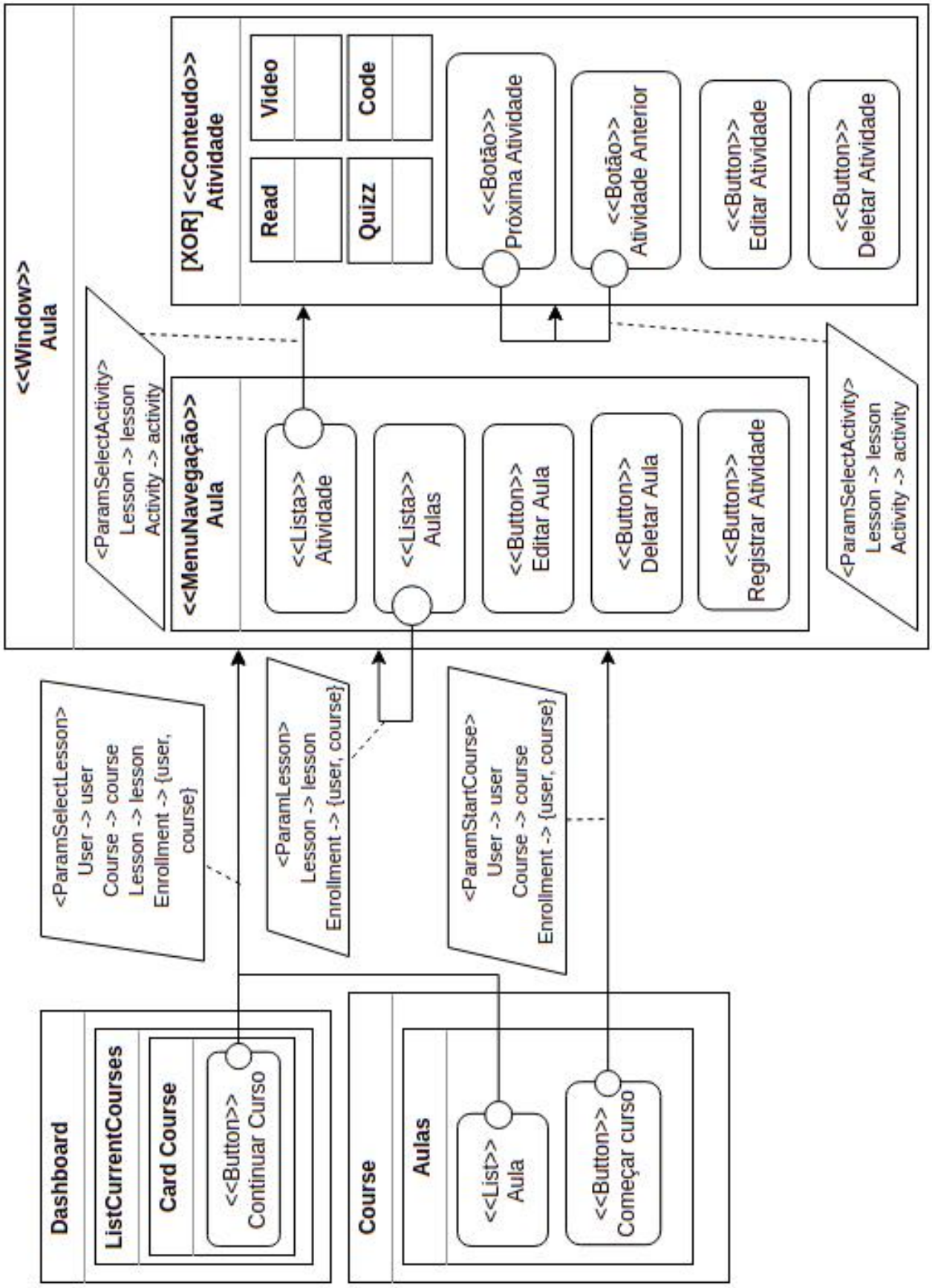
Fonte: Elaborado pelo autor.

Figura 137 – Modelo IFML do fluxo de registro de aulas



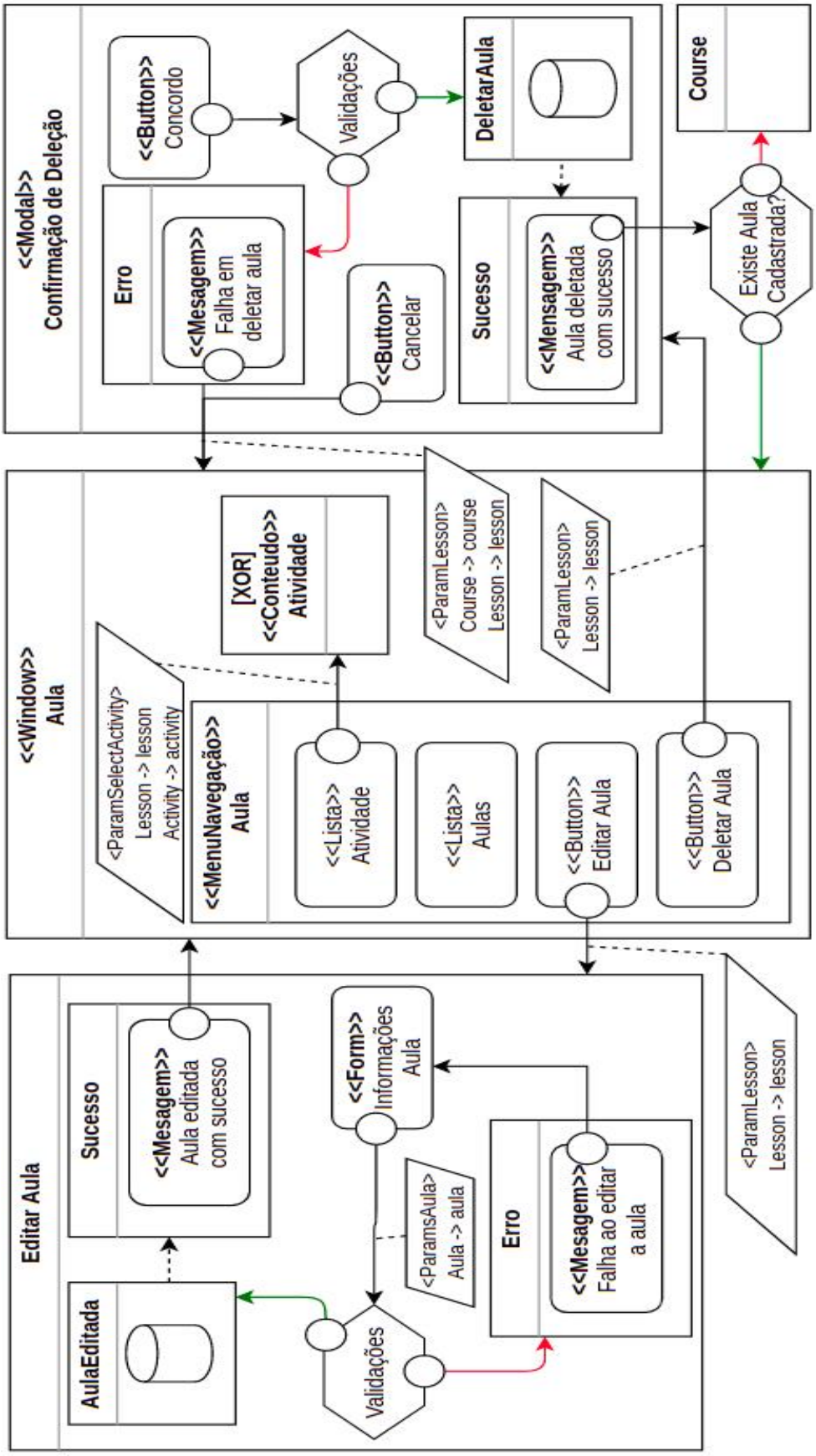
Fonte: Elaborado pelo autor.

Figura 138 – Modelo IFML do fluxo da tela de aula



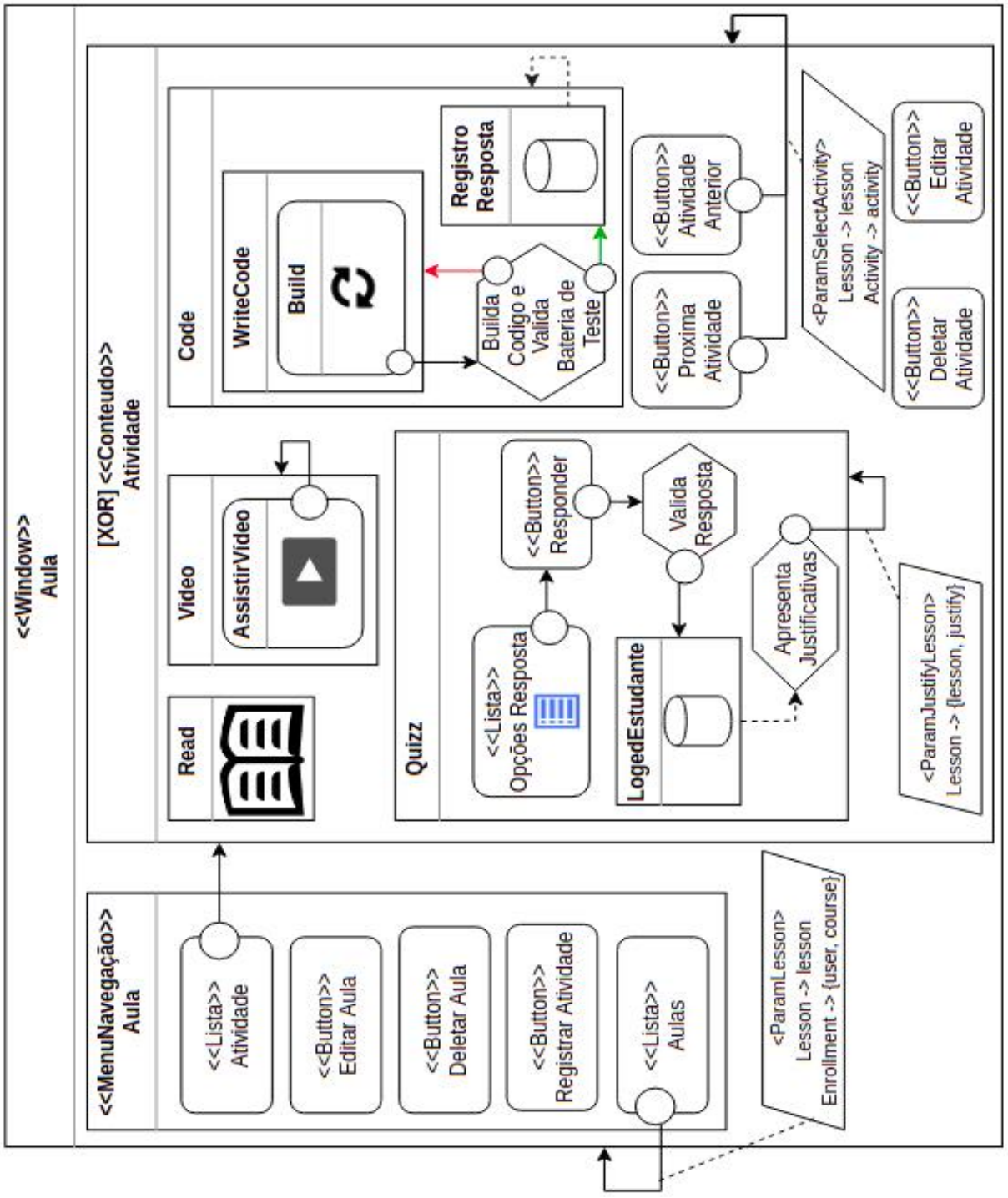
Fonte: Elaborado pelo autor.

Figura 139 – Modelo IFML do fluxo de edição e deleção de aulas



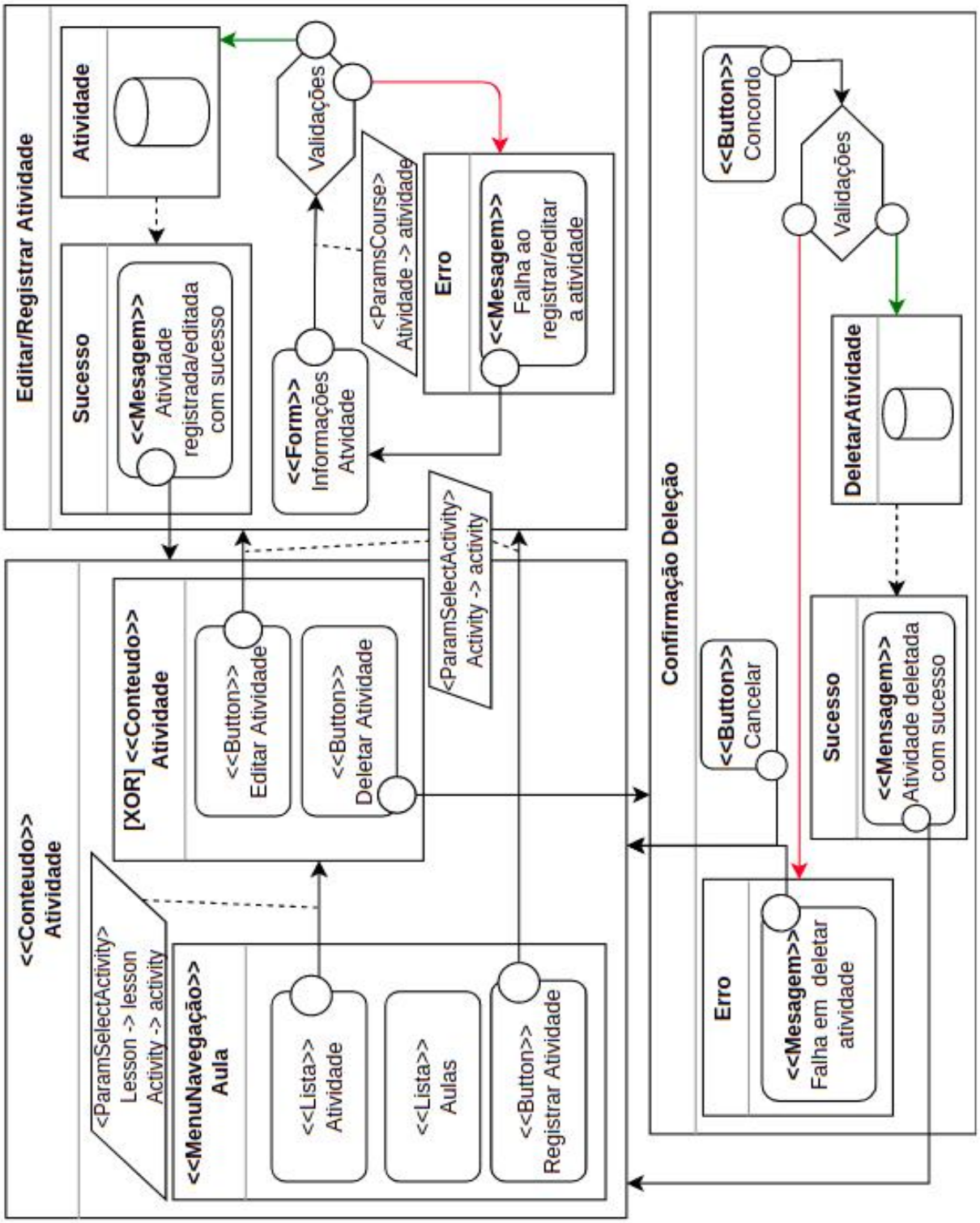
Fonte: Elaborado pelo autor.

Figura 140 – Modelo IFML do fluxo da tela de atividade



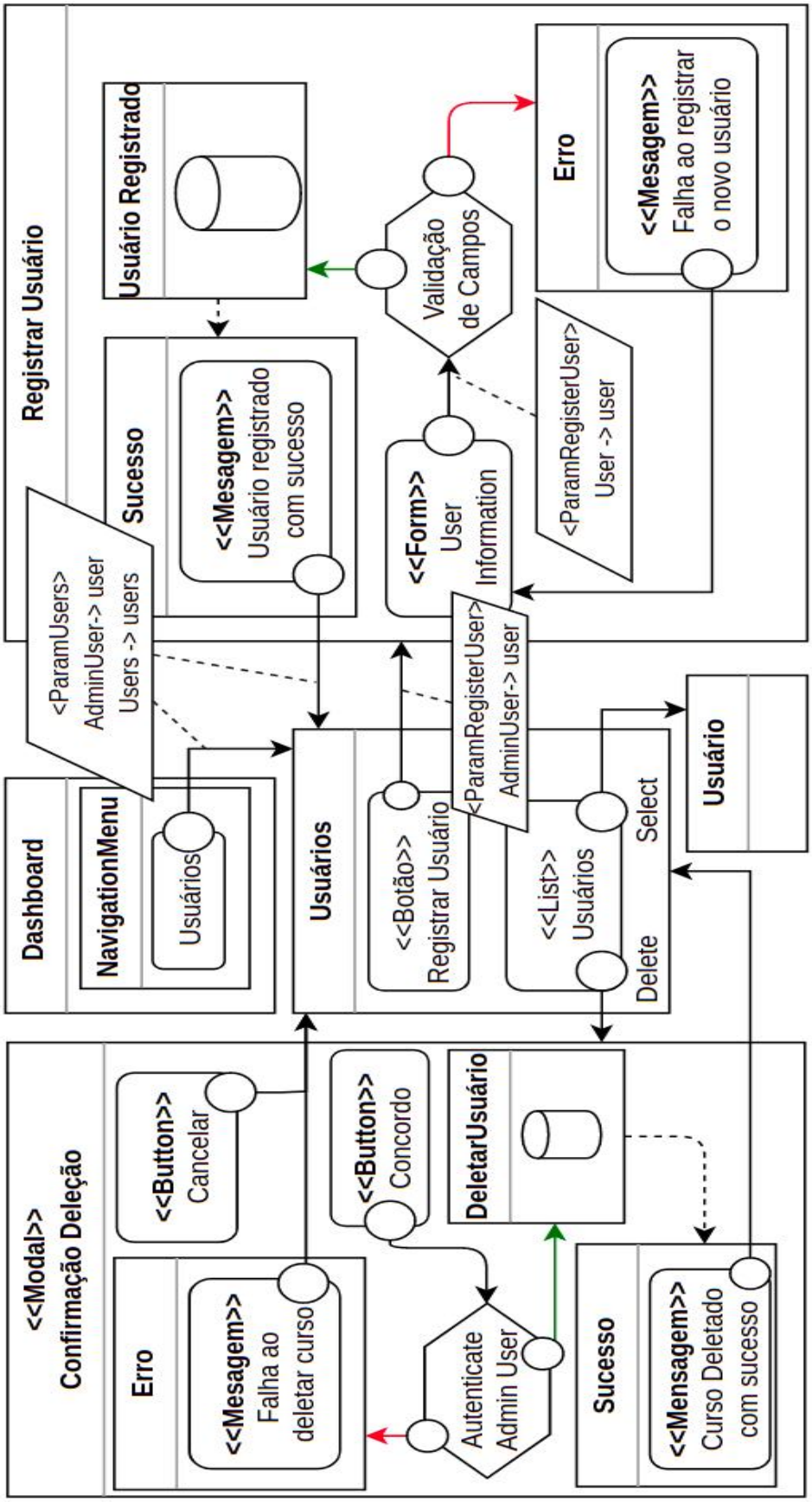
Fonte: Elaborado pelo autor.

Figura 141 – Modelo IFML do fluxo tela registrar atividades



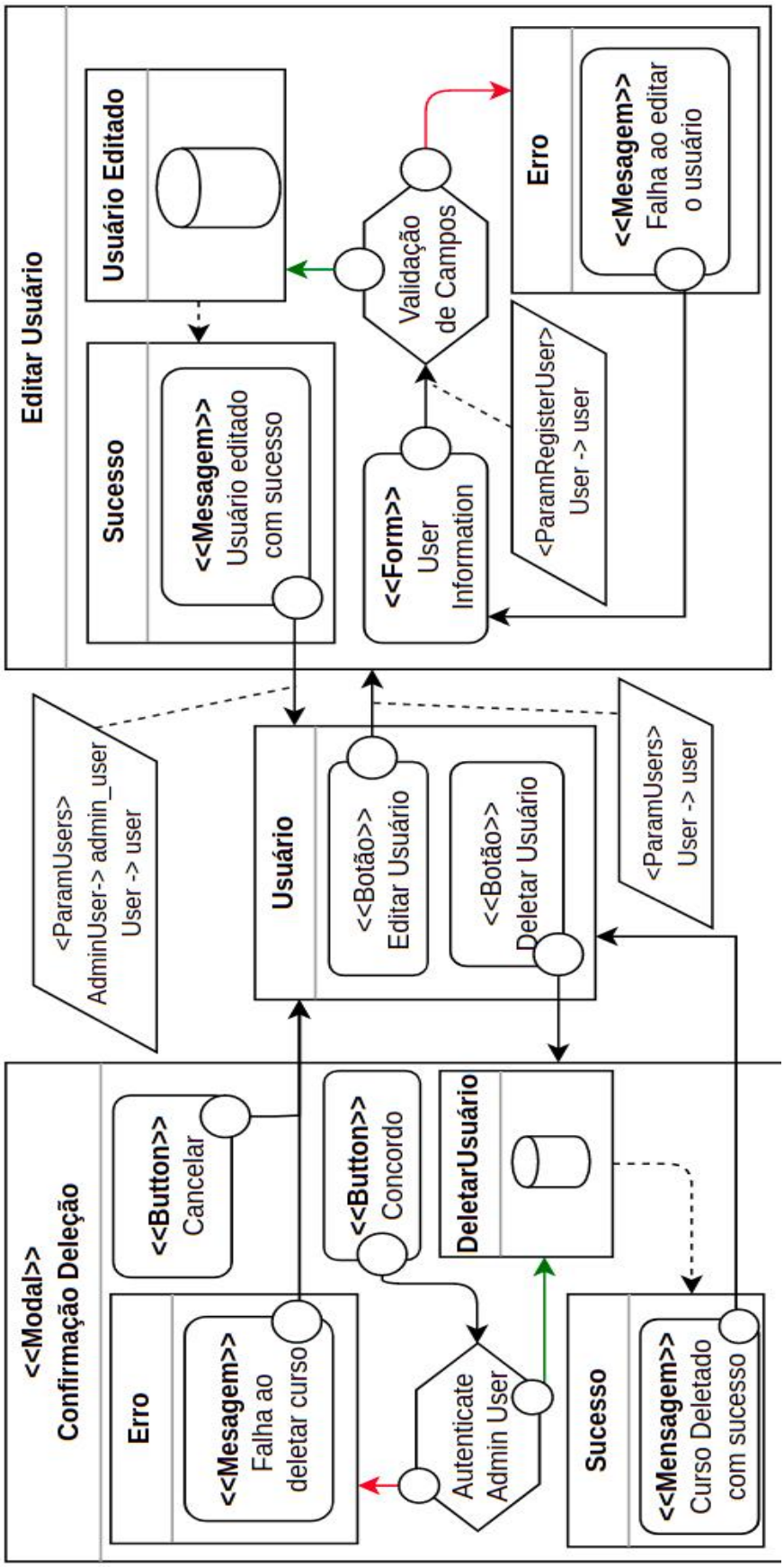
Fonte: Elaborado pelo autor.

Figura 142 – Modelo IFML do fluxo tela de gerência de usuários



Fonte: Elaborado pelo autor.

Figura 143 – Modelo IFML do fluxo tela de edição e deleção de usuários

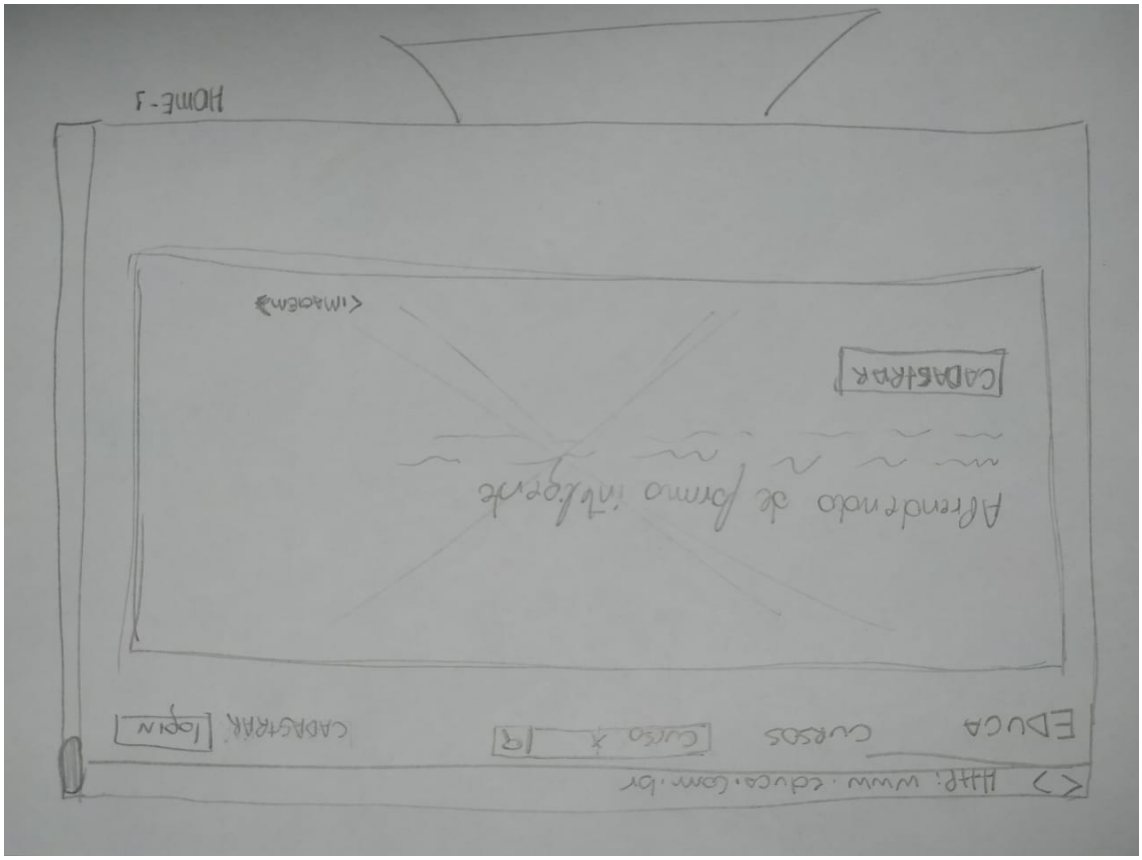


Fonte: Elaborado pelo autor.

D.2 Sketchs

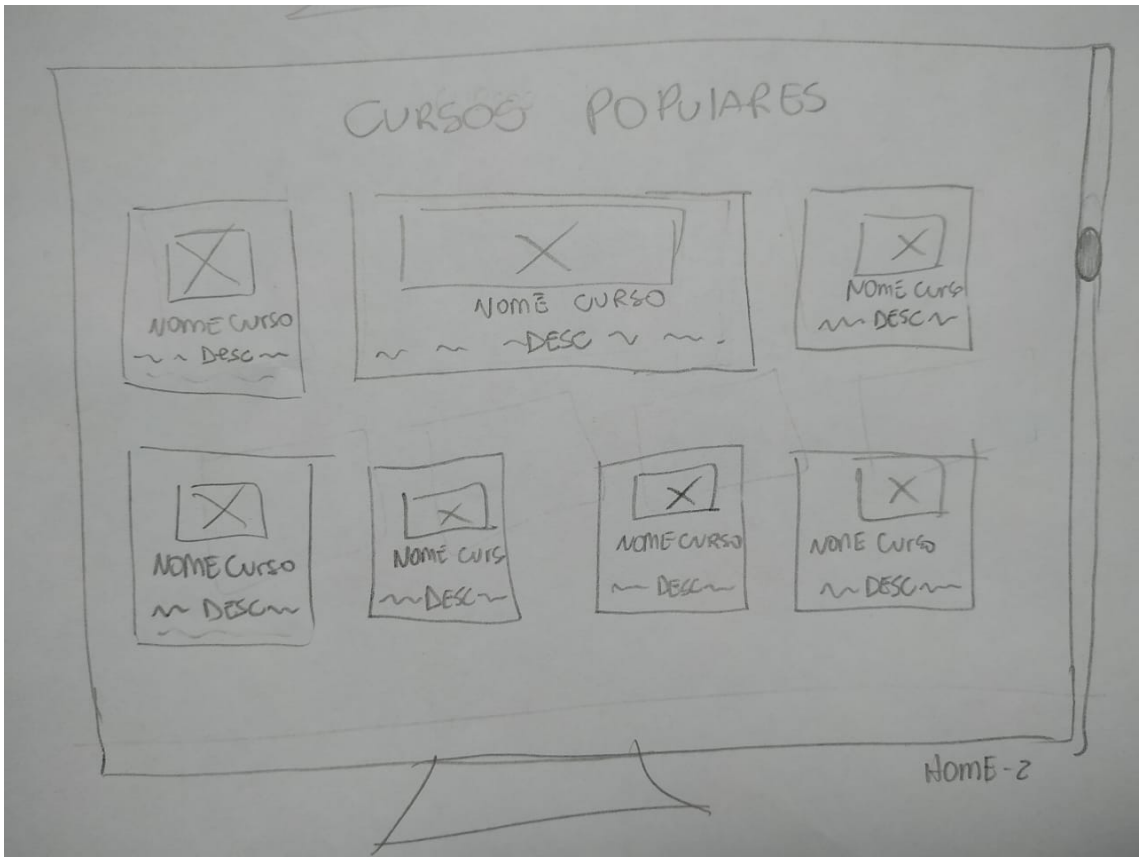
D.2.1 Home

Figura 144 – Modelo sketch da tela home parte 1



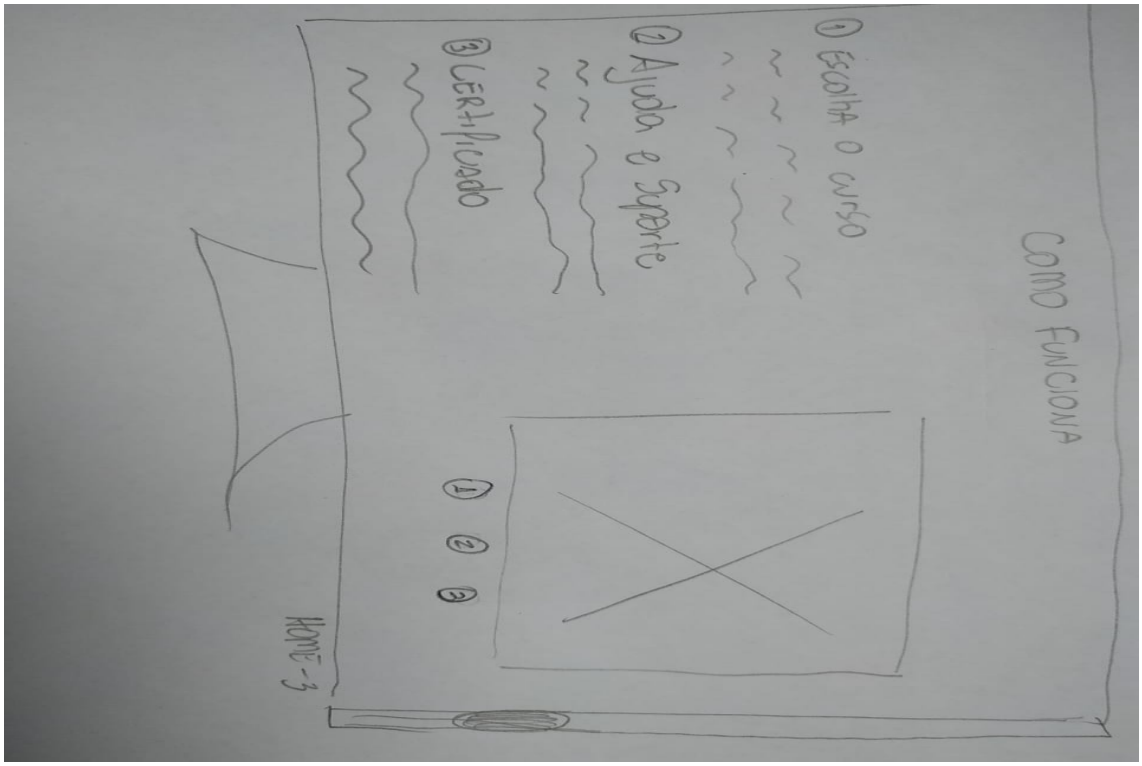
Fonte: Elaborado pelo autor.

Figura 145 – Modelo sketch da tela home parte 2



Fonte: Elaborado pelo autor.

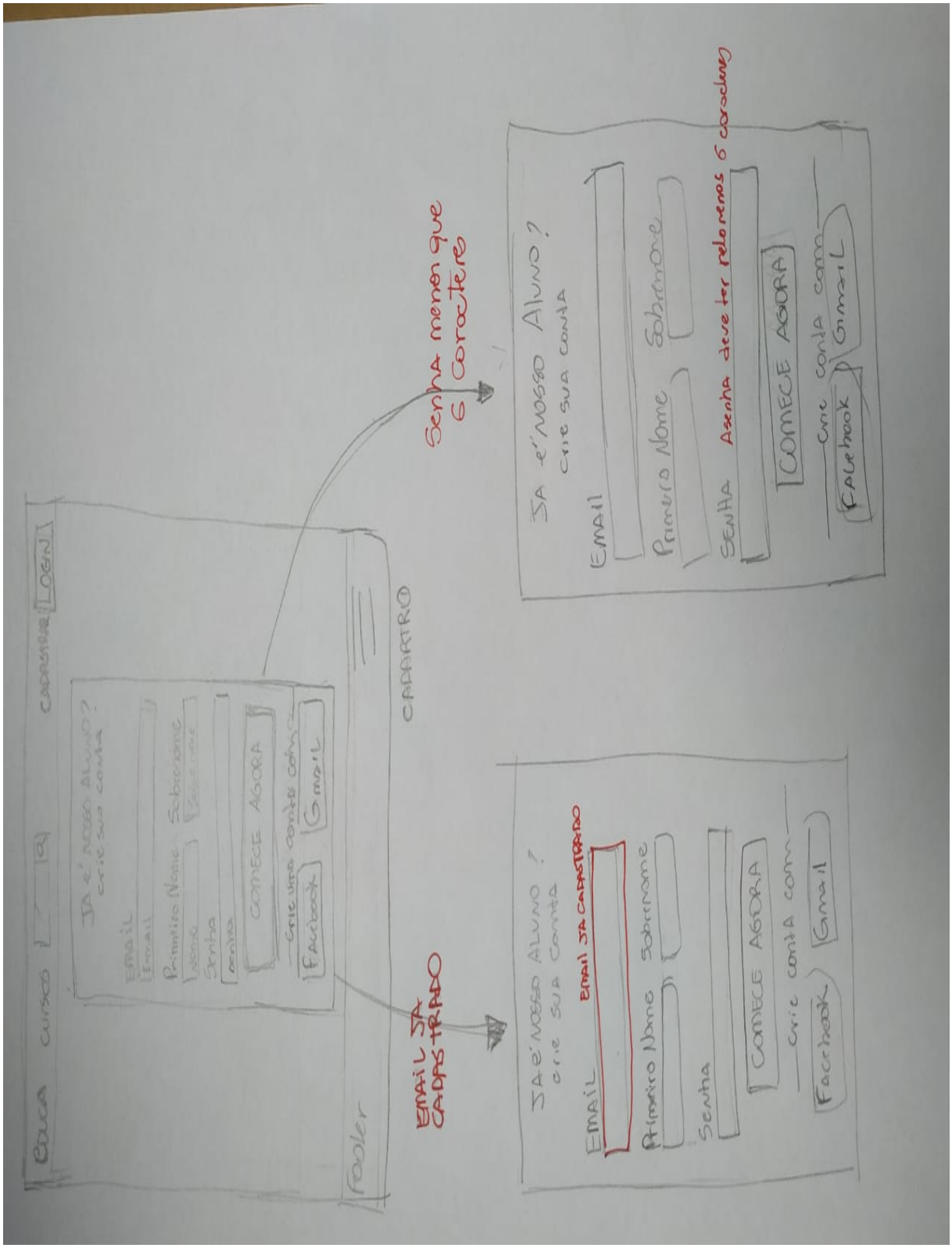
Figura 146 – Modelo sketch da tela home parte 3



Fonte: Elaborado pelo autor.

D.2.2 Cadastro

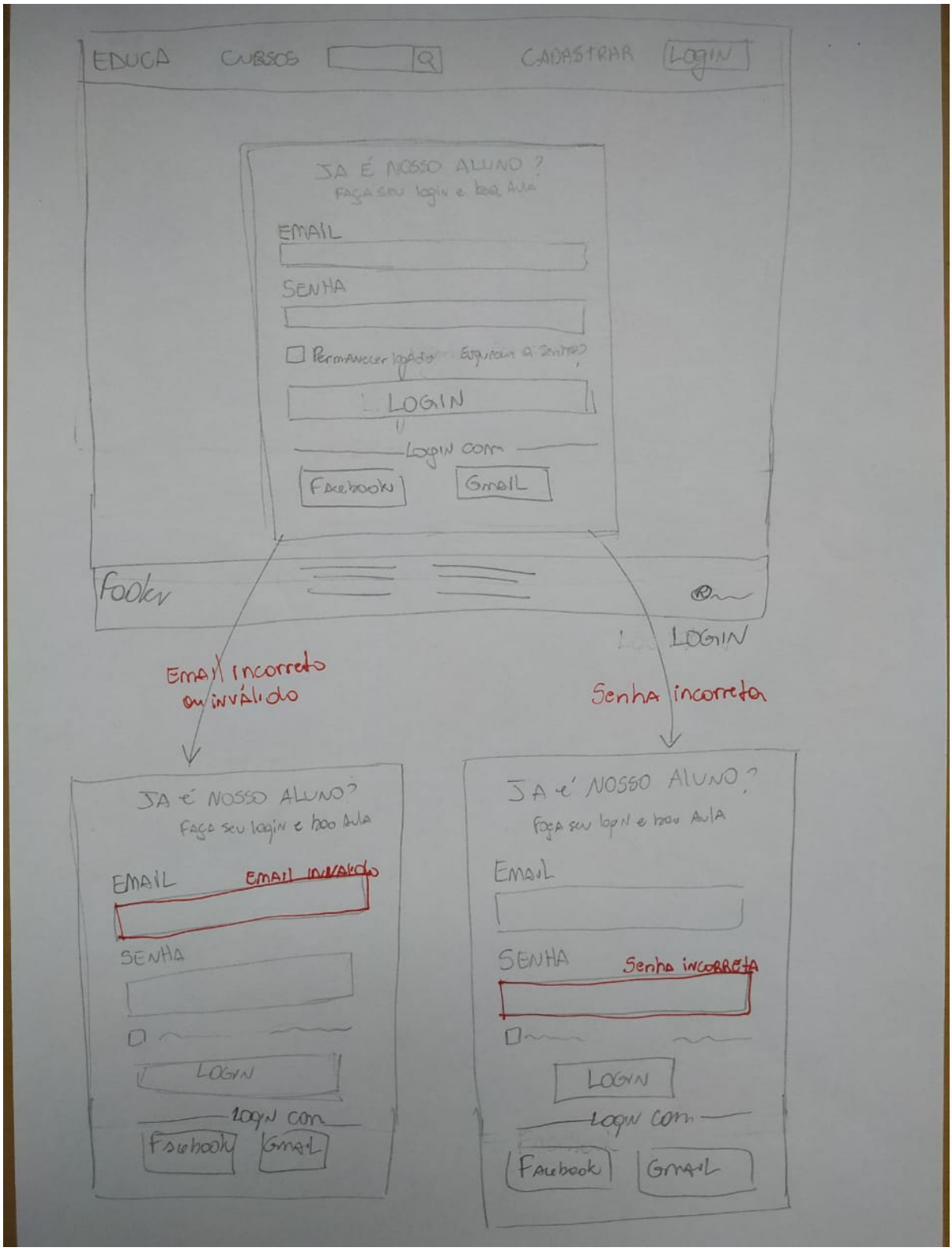
Figura 147 – Modelo sketch da tela cadastro



Fonte: Elaborado pelo autor.

D.2.3 Login

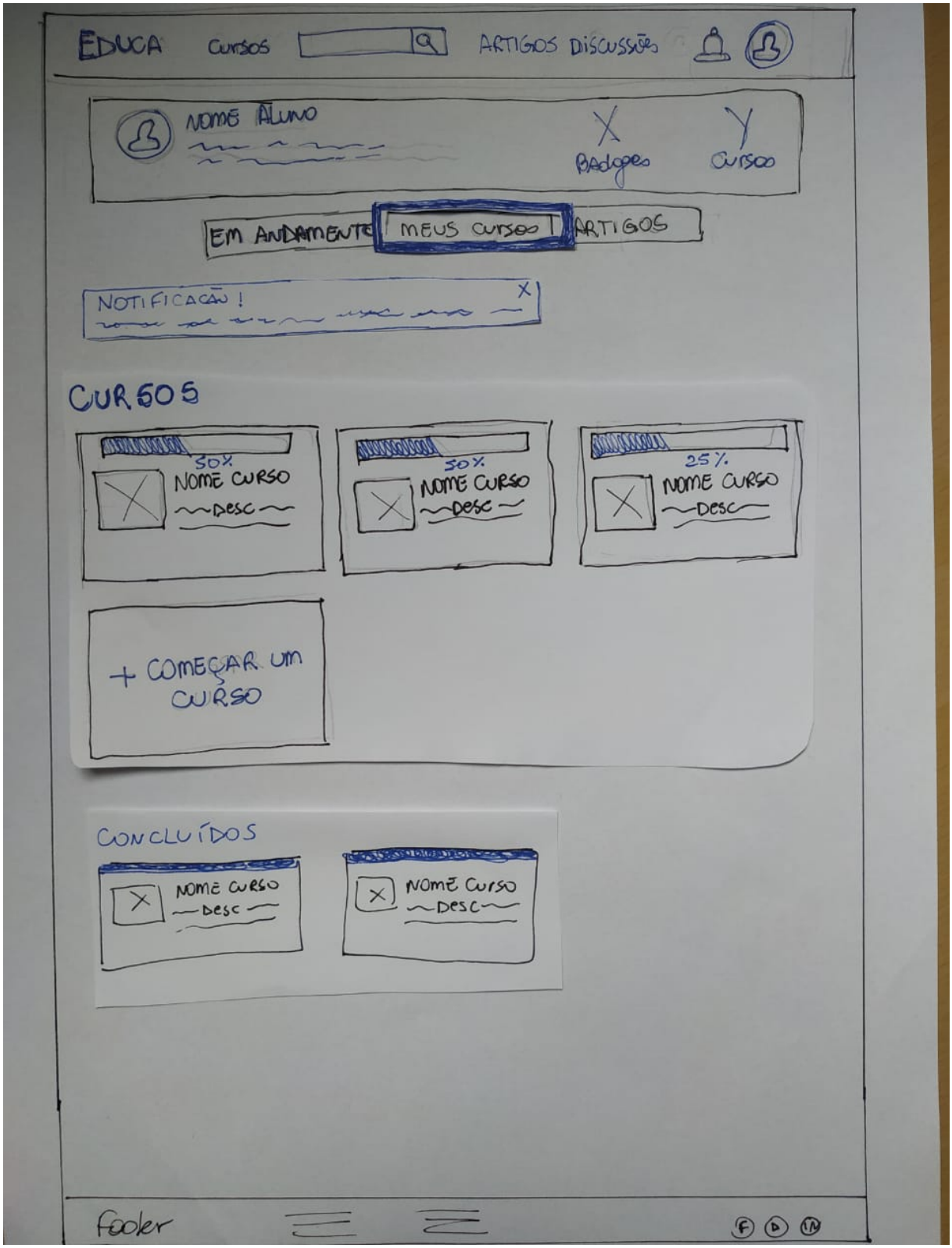
Figura 148 – Modelo sketch da tela login



Fonte: Elaborado pelo autor.

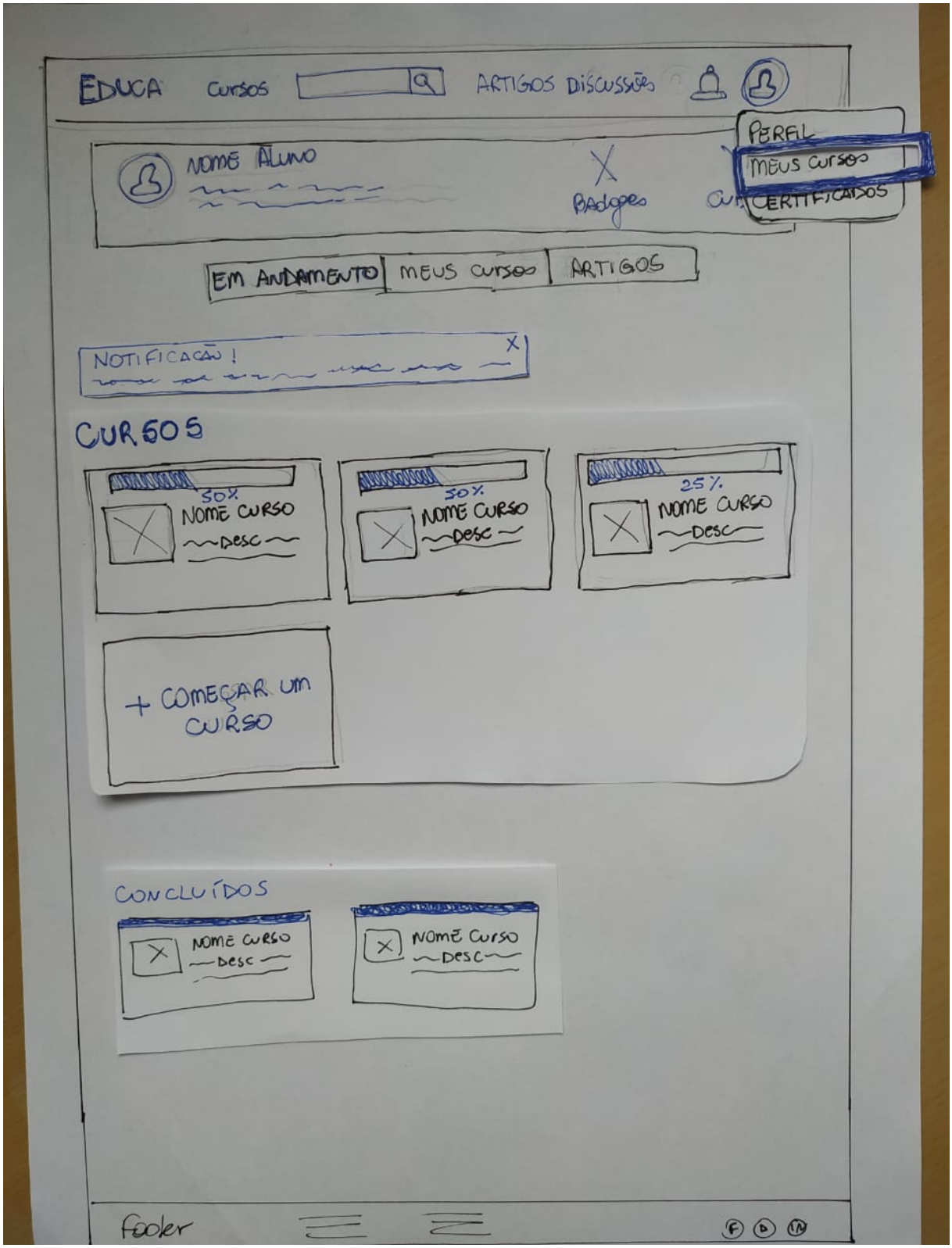
D.2.4 Dashboard

Figura 149 – Modelo sketch da tela dashboard 1



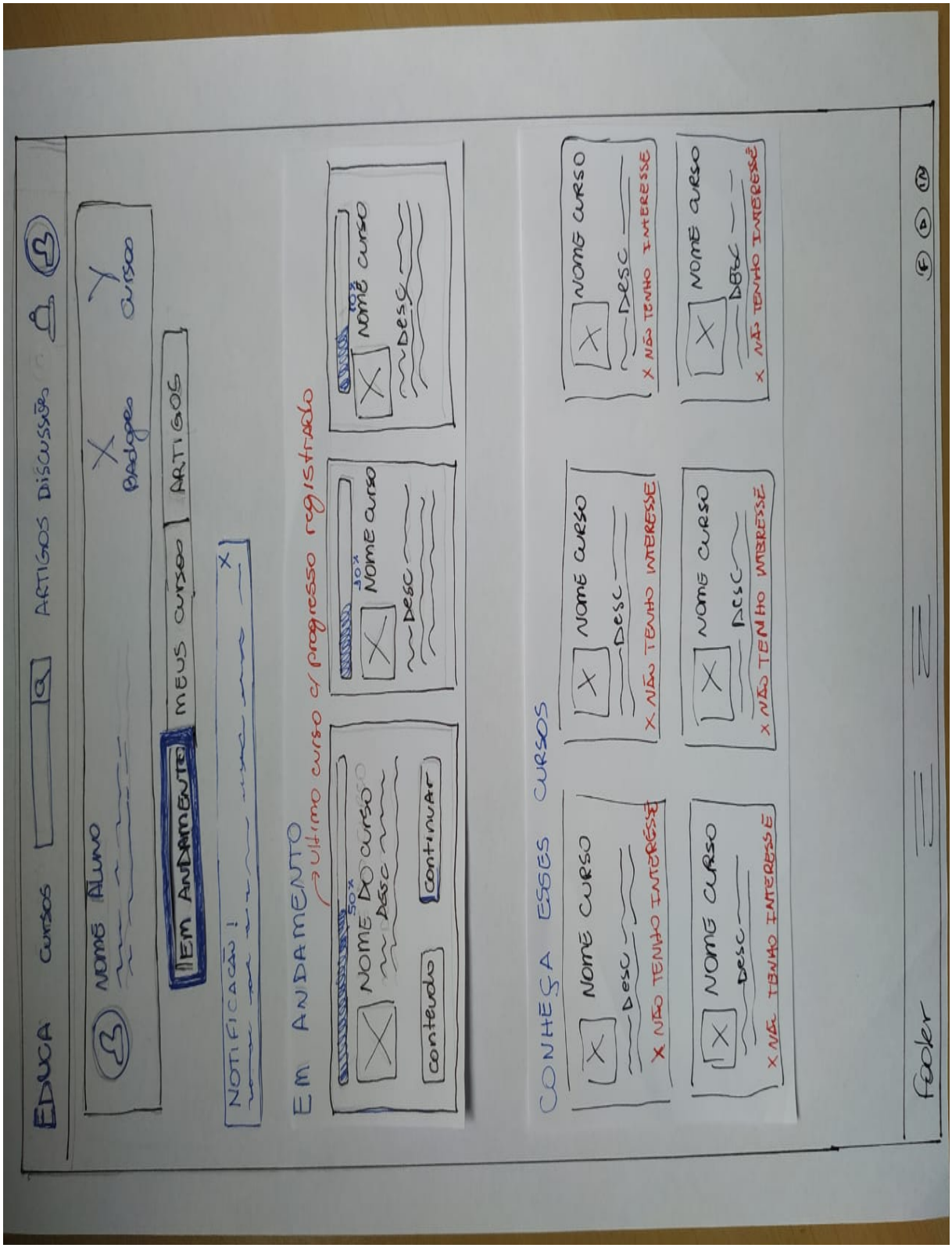
Fonte: Elaborado pelo autor.

Figura 150 – Modelo sketch da tela dashboard 2



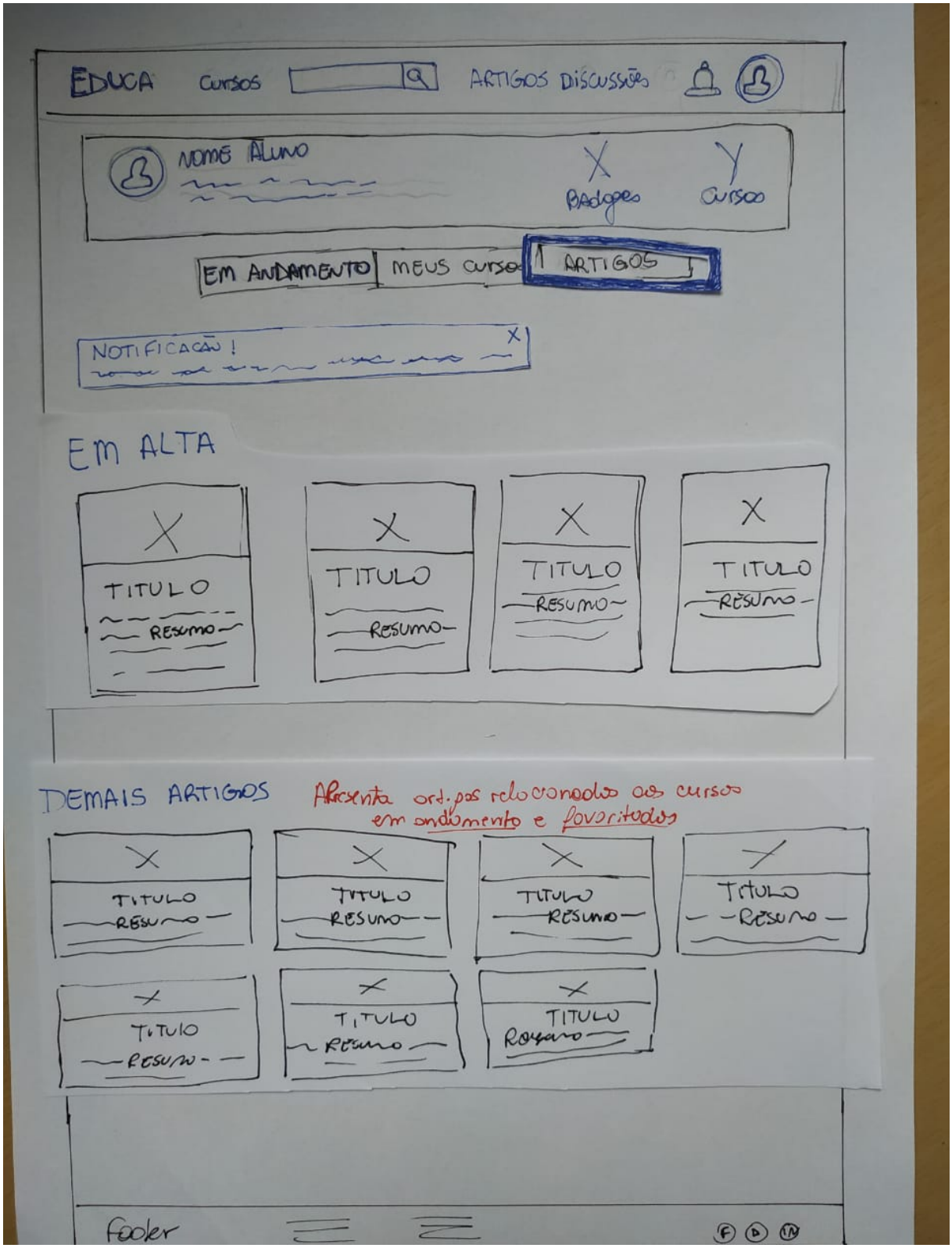
Fonte: Elaborado pelo autor.

Figura 151 – Modelo sketch da tela dashboard em estato de já com cursos em andamento



Fonte: Elaborado pelo autor.

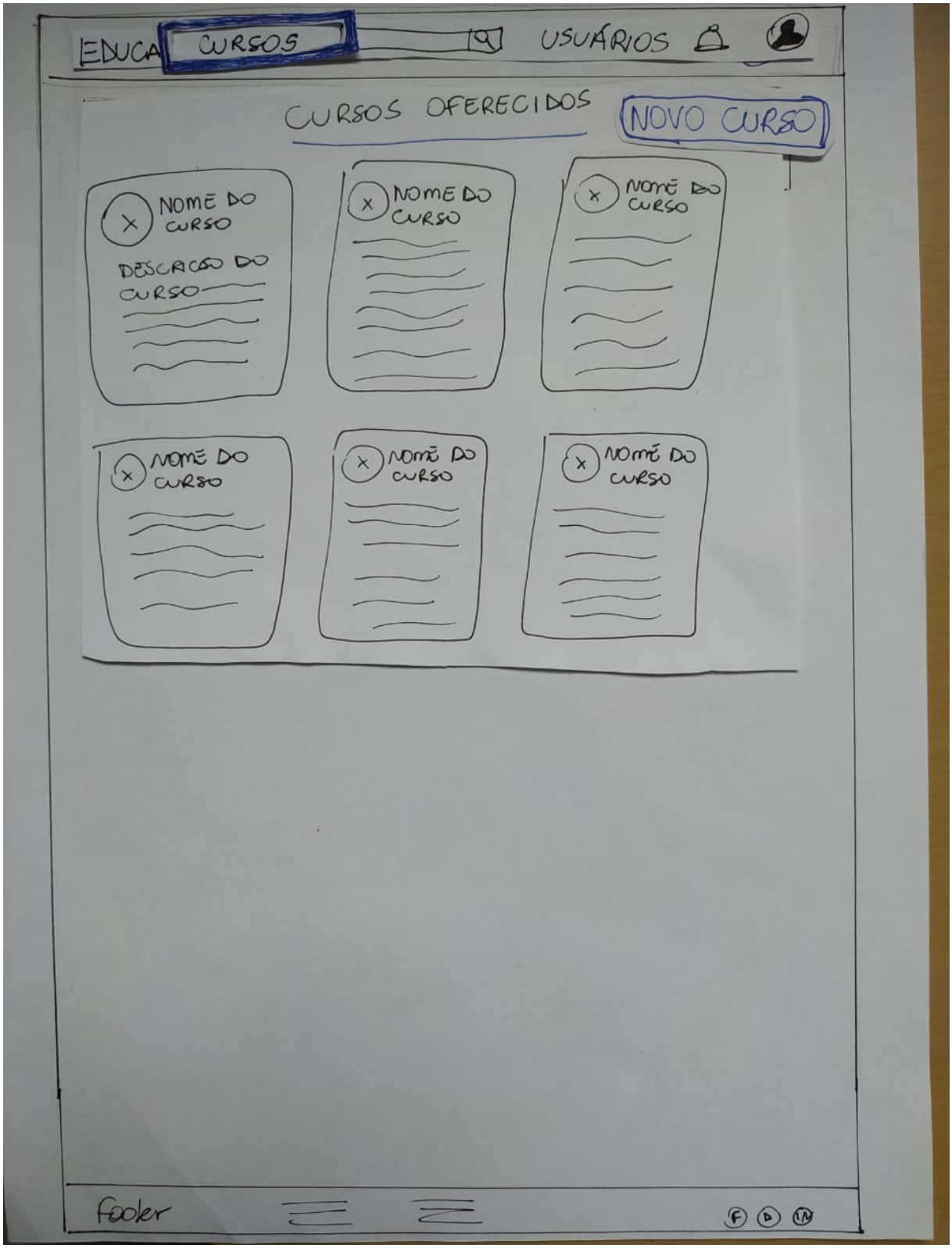
Figura 152 – Modelo sketch da tela dashboard área de artigos



Fonte: Elaborado pelo autor.

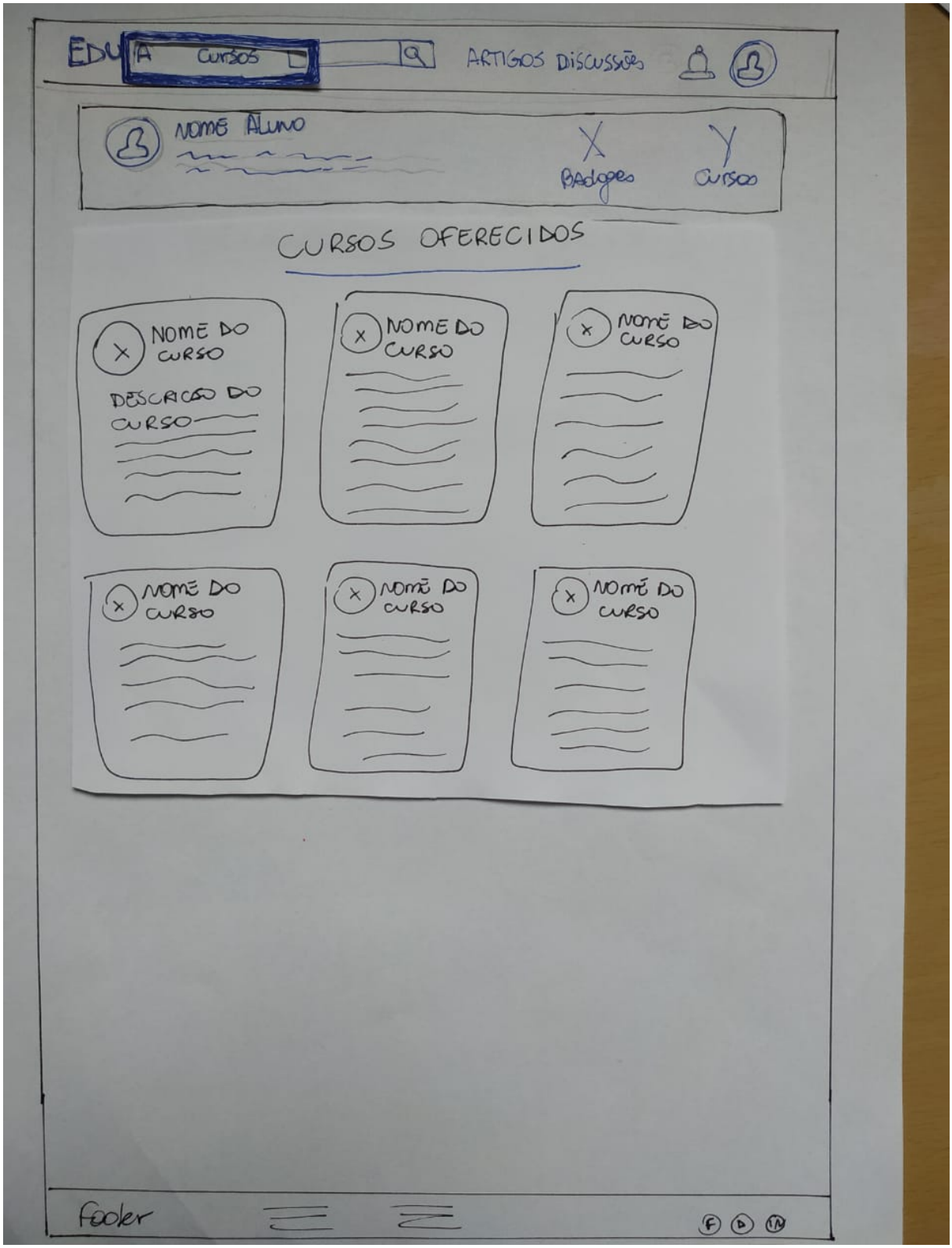
D.2.5 Cursos

Figura 153 – Modelo sketch da tela de cursos para usuários administradores



Fonte: Elaborado pelo autor.

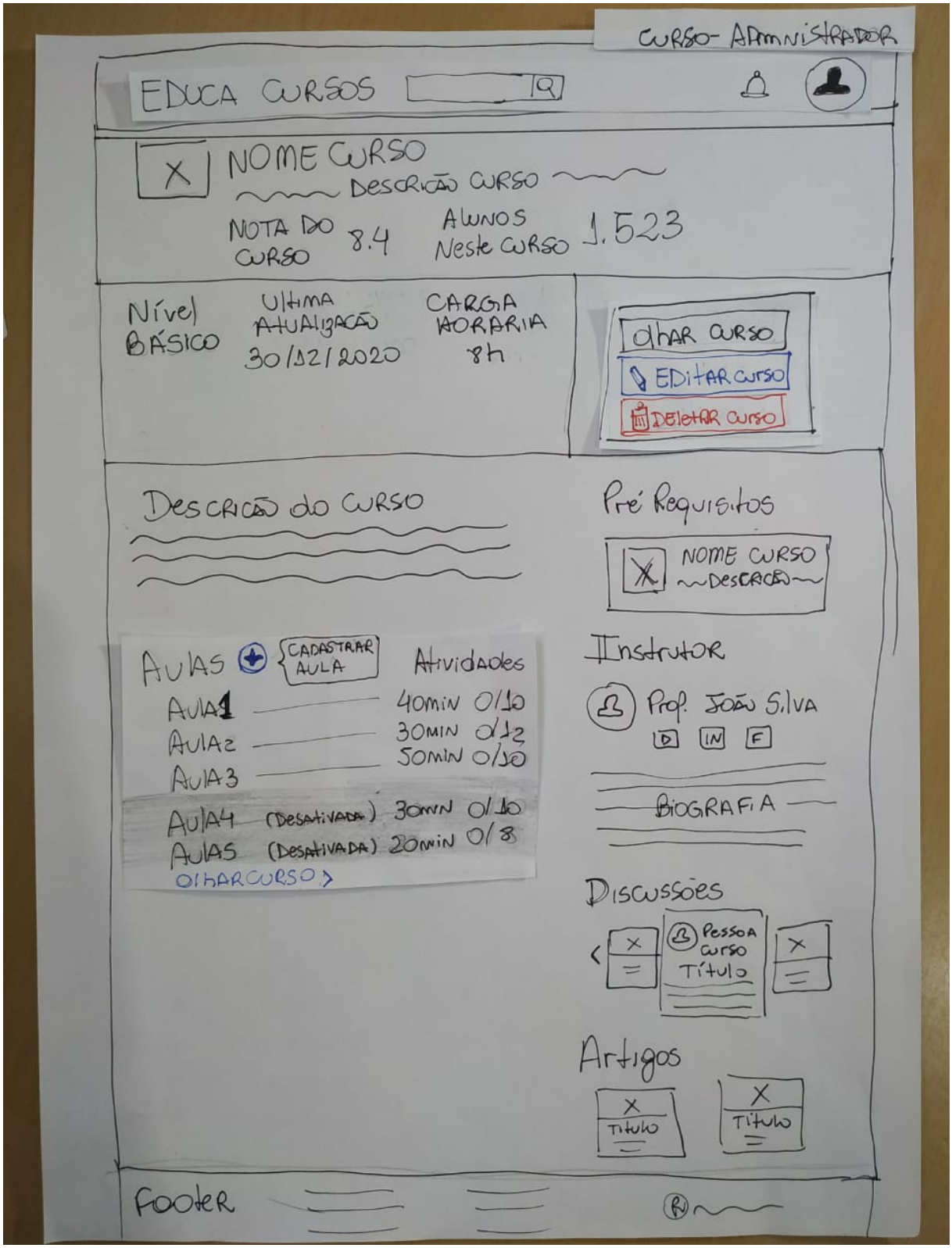
Figura 154 – Modelo sketch da tela de cursos para usuários estudantes ou instrutores



Fonte: Elaborado pelo autor.

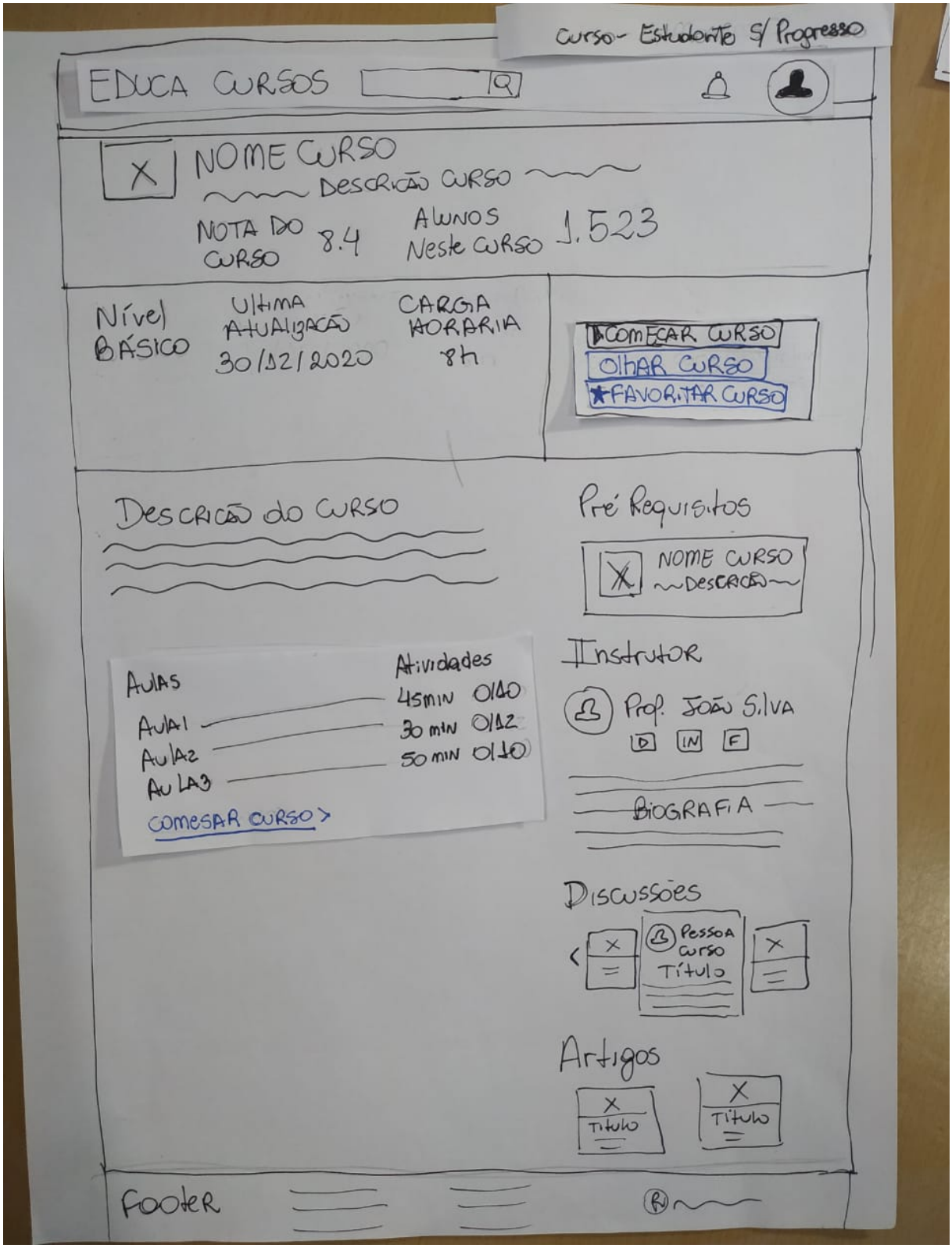
D.2.6 Curso

Figura 155 – Modelo sketch da tela de curso para administradores



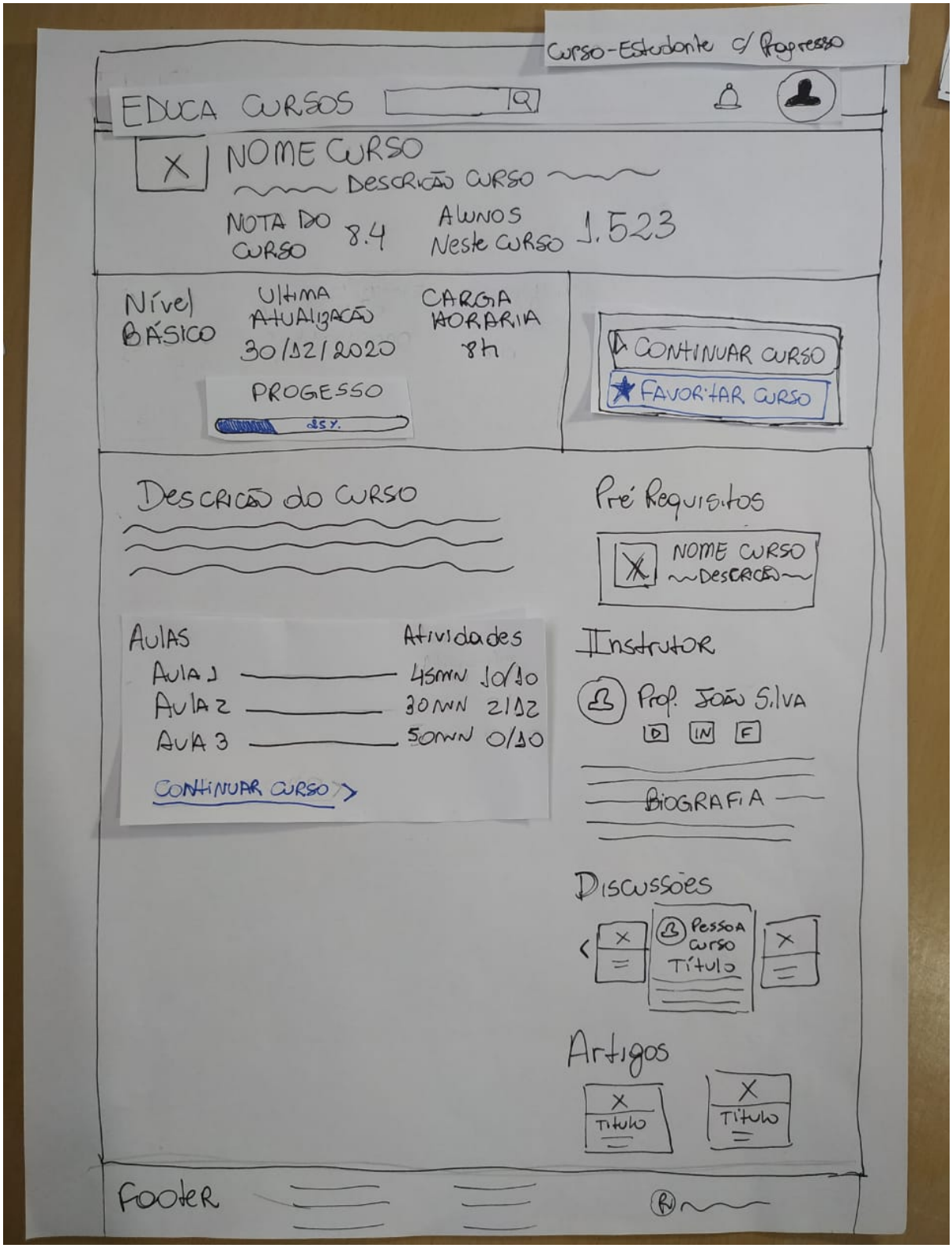
Fonte: Elaborado pelo autor.

Figura 156 – Modelo sketch da tela de curso para estudantes sem progresso registrado



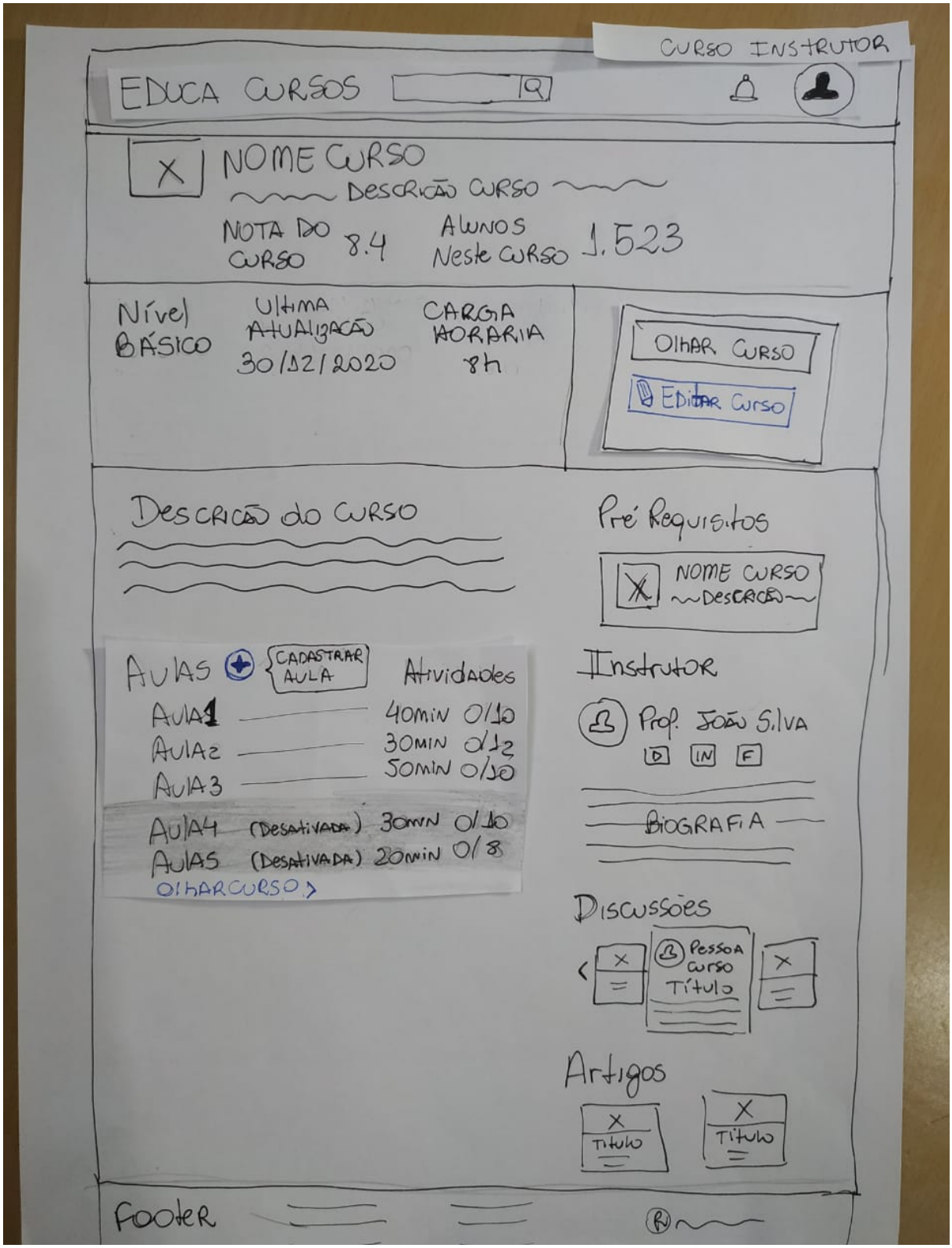
Fonte: Elaborado pelo autor.

Figura 157 – Modelo sketch da tela de curso para estudantes com progresso registrado



Fonte: Elaborado pelo autor.

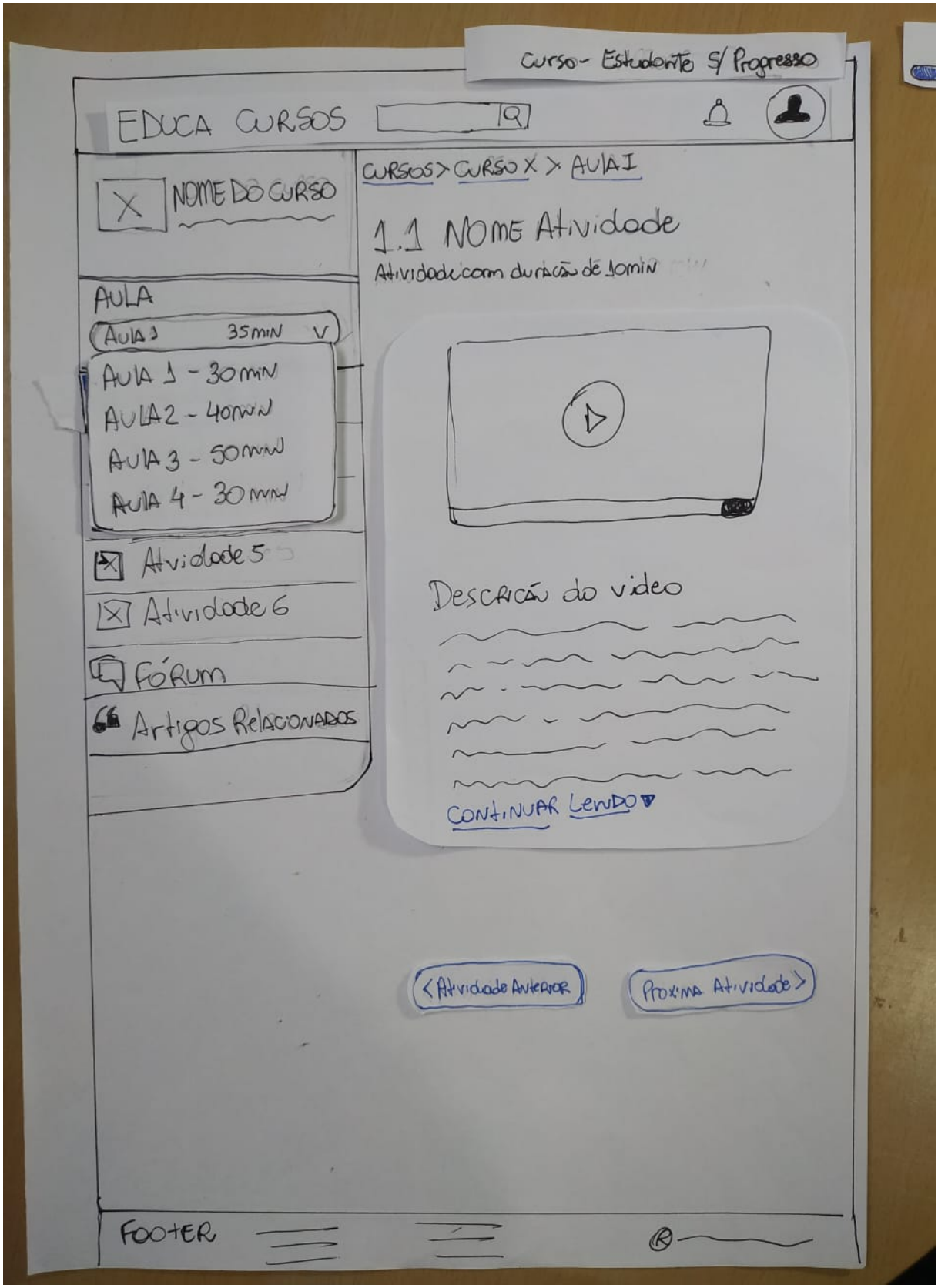
Figura 158 – Modelo sketch da tela de curso para instrutores



Fonte: Elaborado pelo autor.

D.2.7 Aula

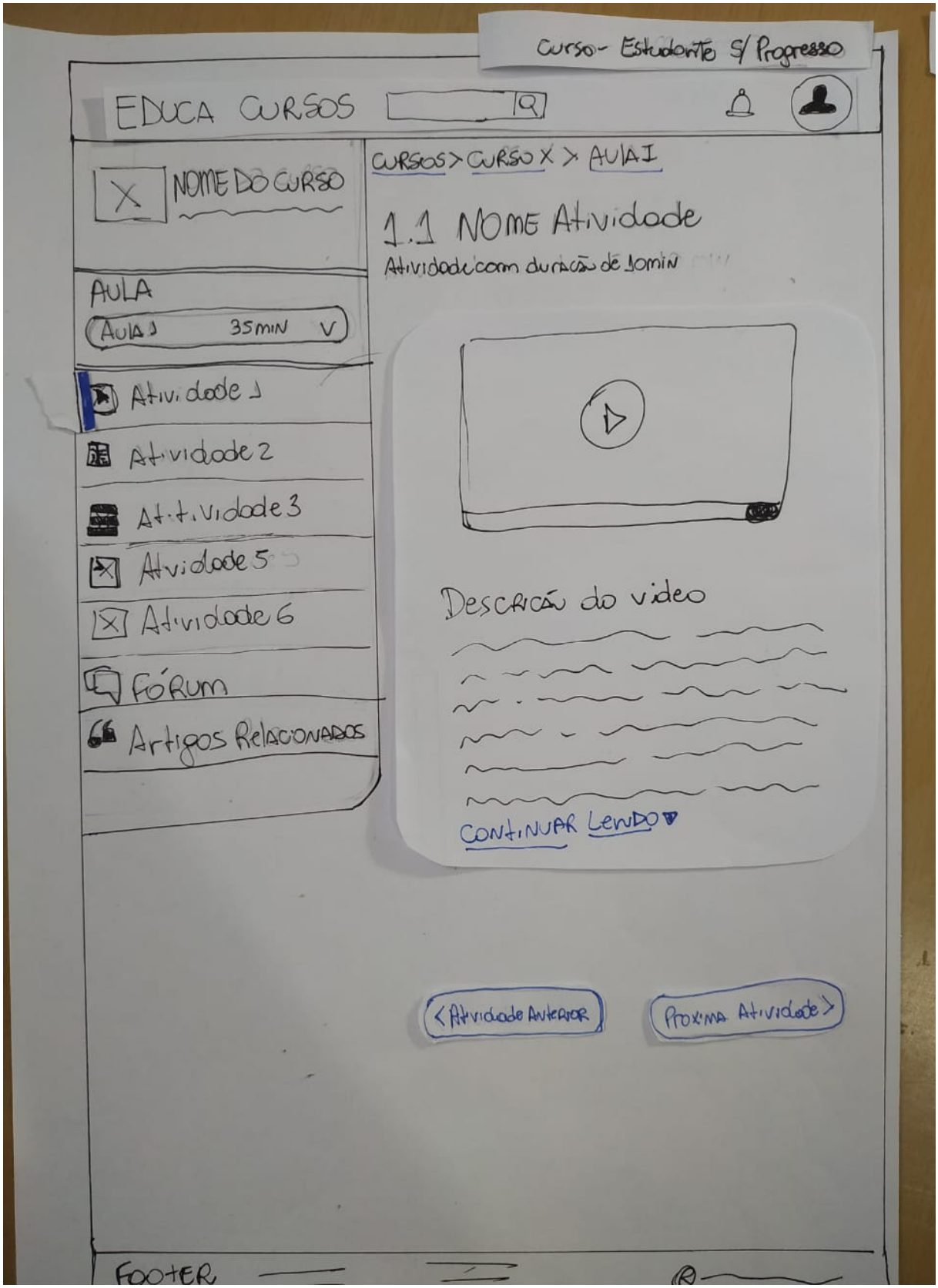
Figura 159 – Modelo sketch da tela de curso para administradores



Fonte: Elaborado pelo autor.

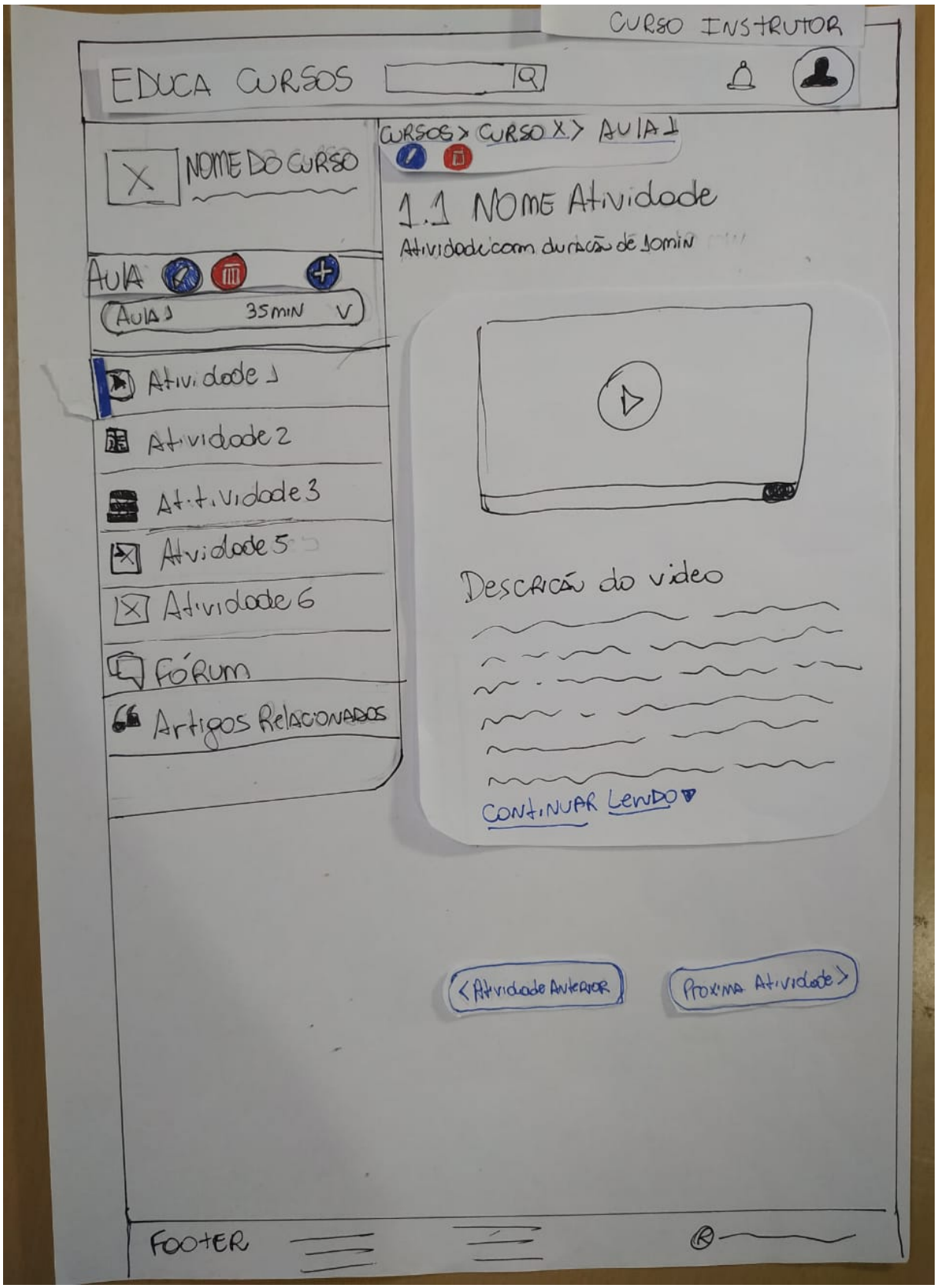
D.2.8 Atividade

Figura 160 – Modelo sketch da tela de atividades para estudantes



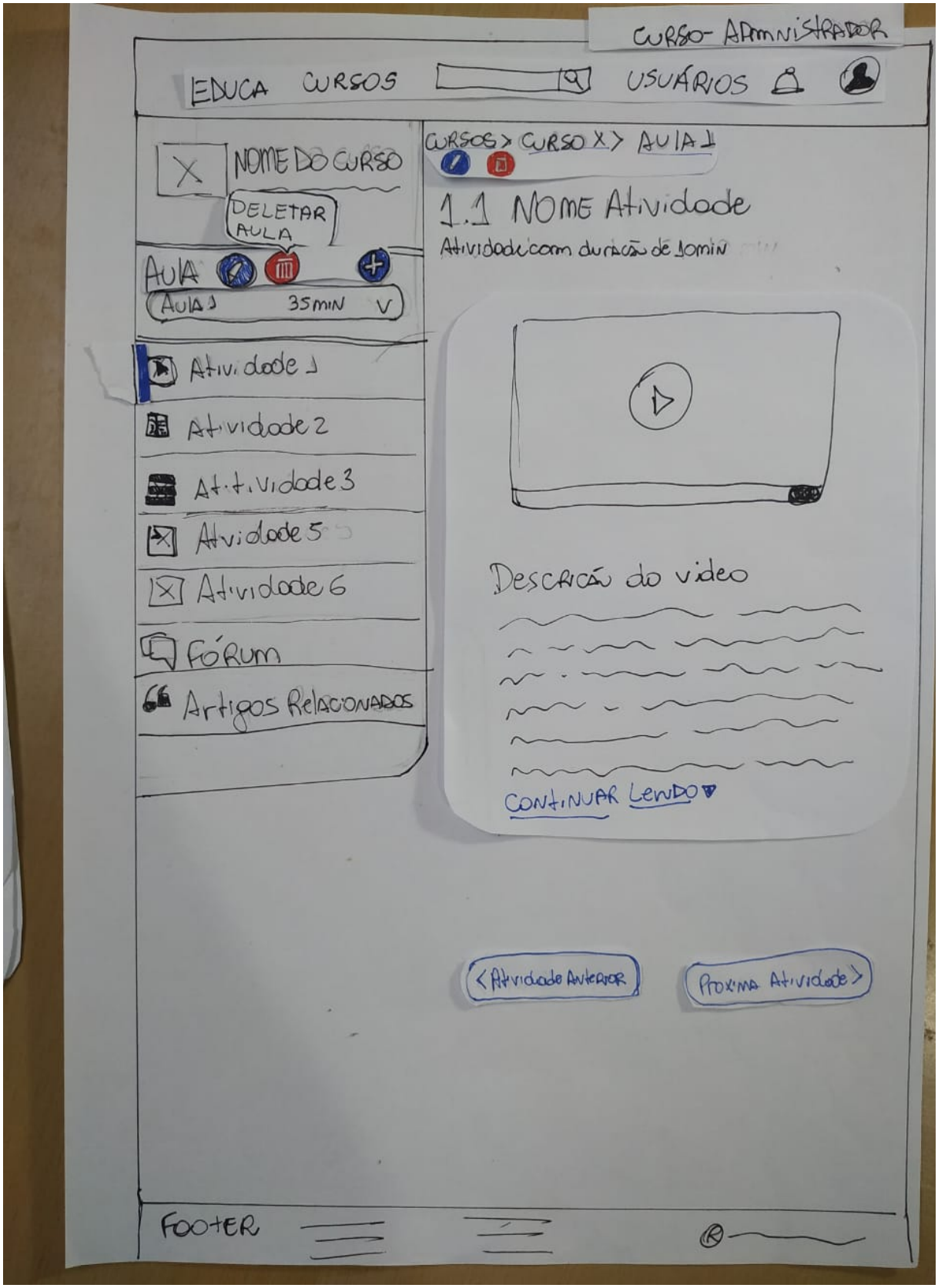
Fonte: Elaborado pelo autor.

Figura 161 – Modelo sketch da tela de atividades para instrutores



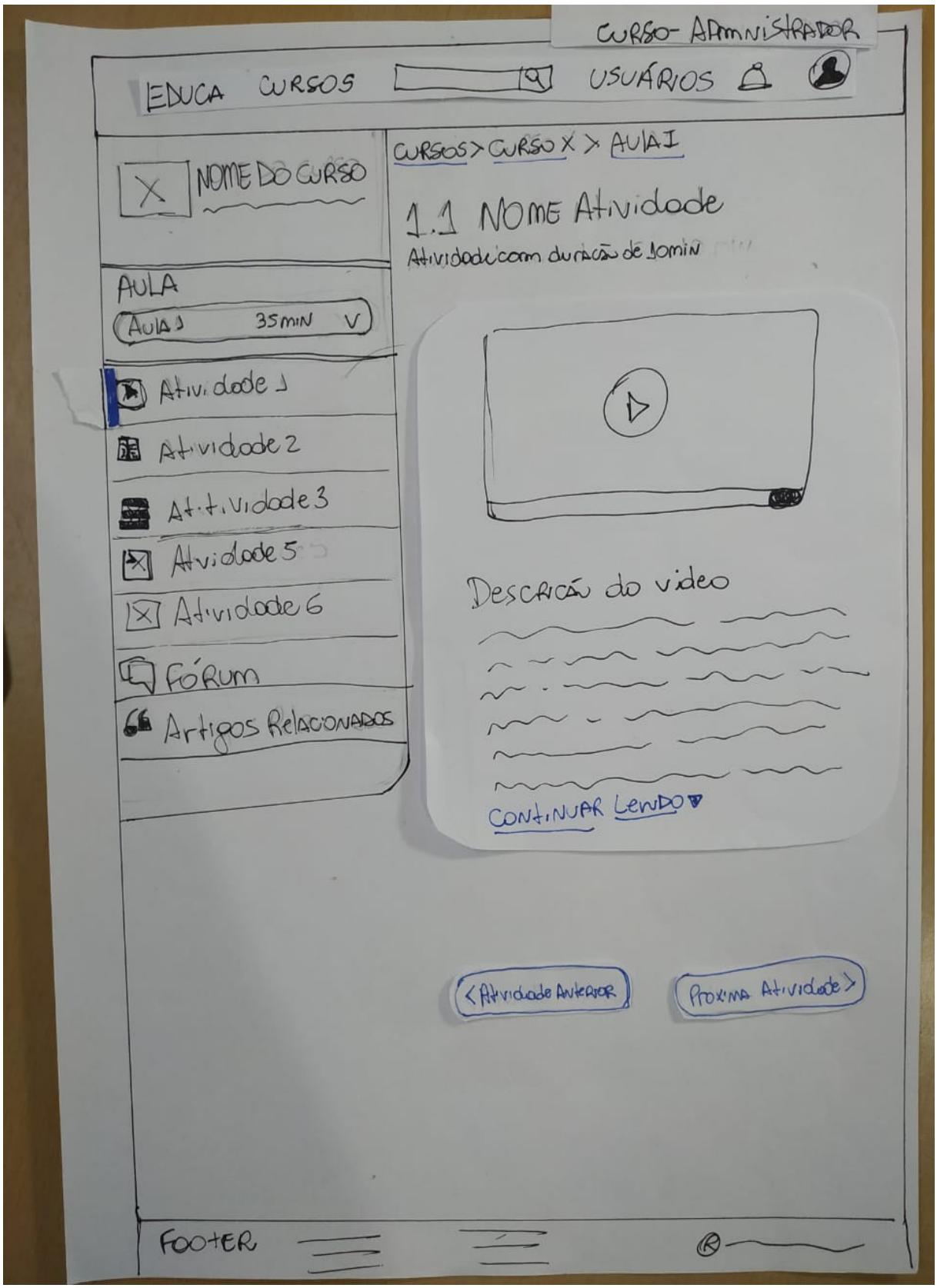
Fonte: Elaborado pelo autor.

Figura 162 – Modelo sketch da tela de atividades para administrador - parte 1



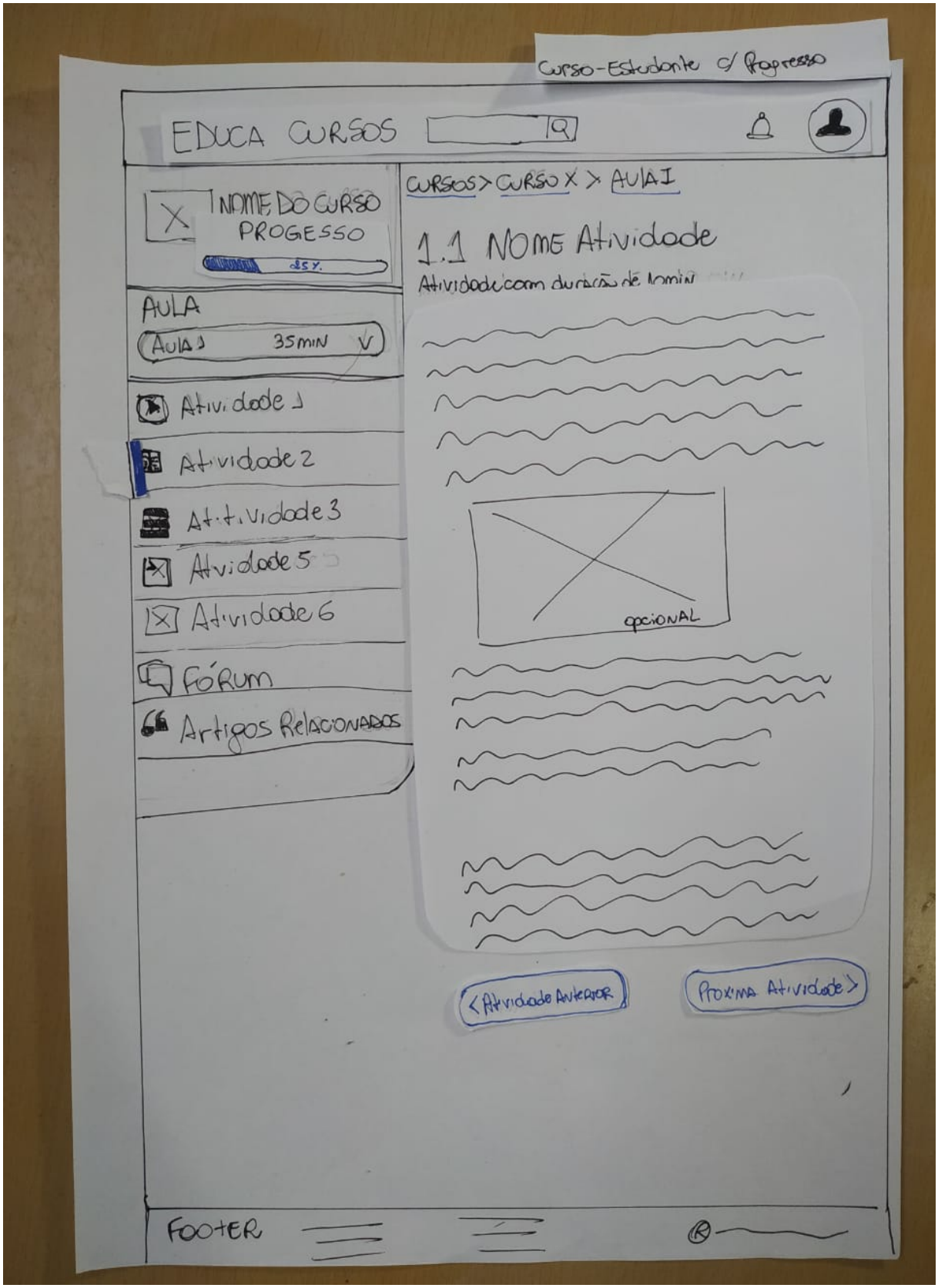
Fonte: Elaborado pelo autor.

Figura 163 – Modelo sketch da tela de atividades para administrador - parte 2



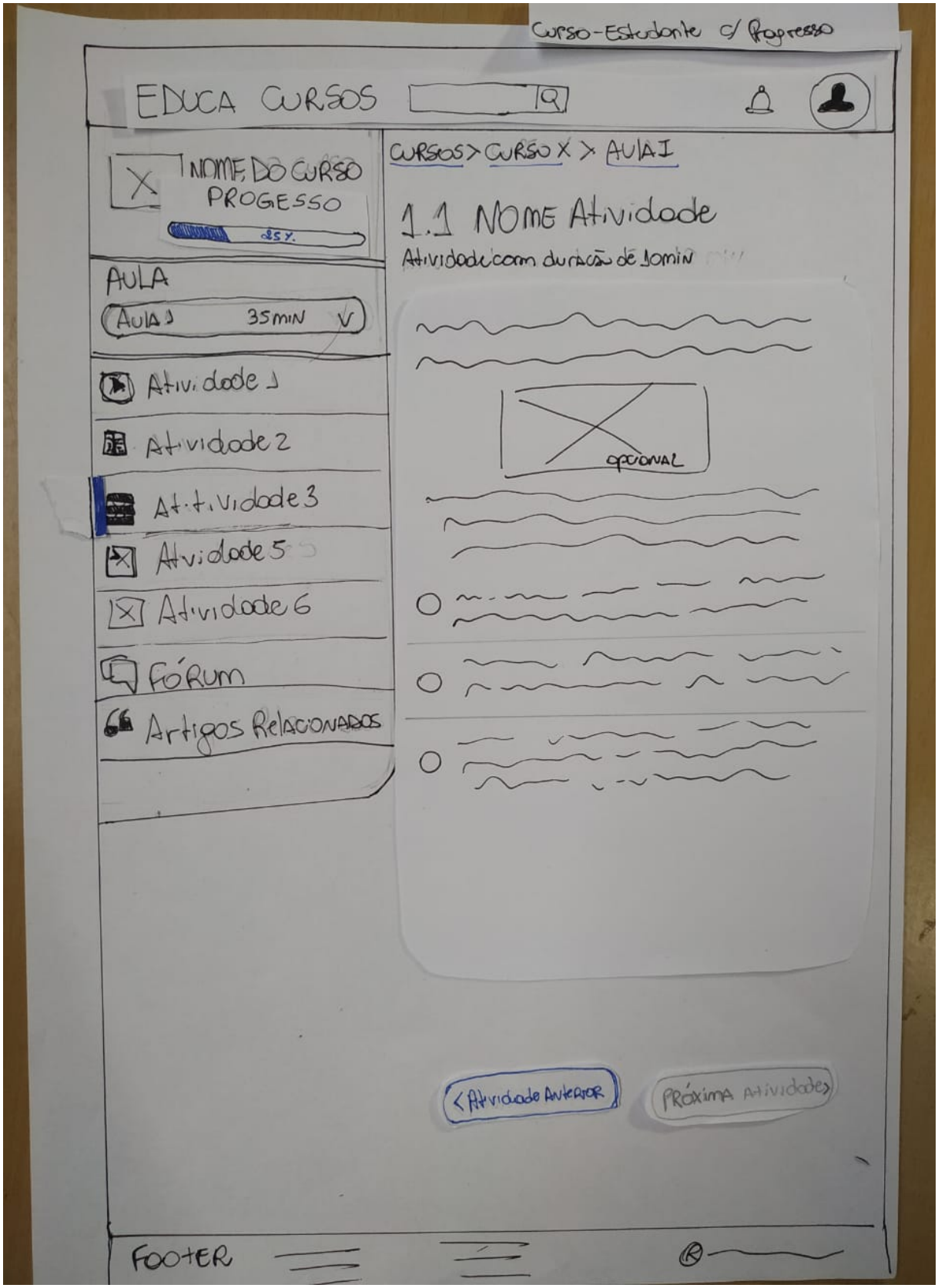
Fonte: Elaborado pelo autor.

Figura 164 – Modelo sketch da tela de atividades do tipo leitura para estudantes



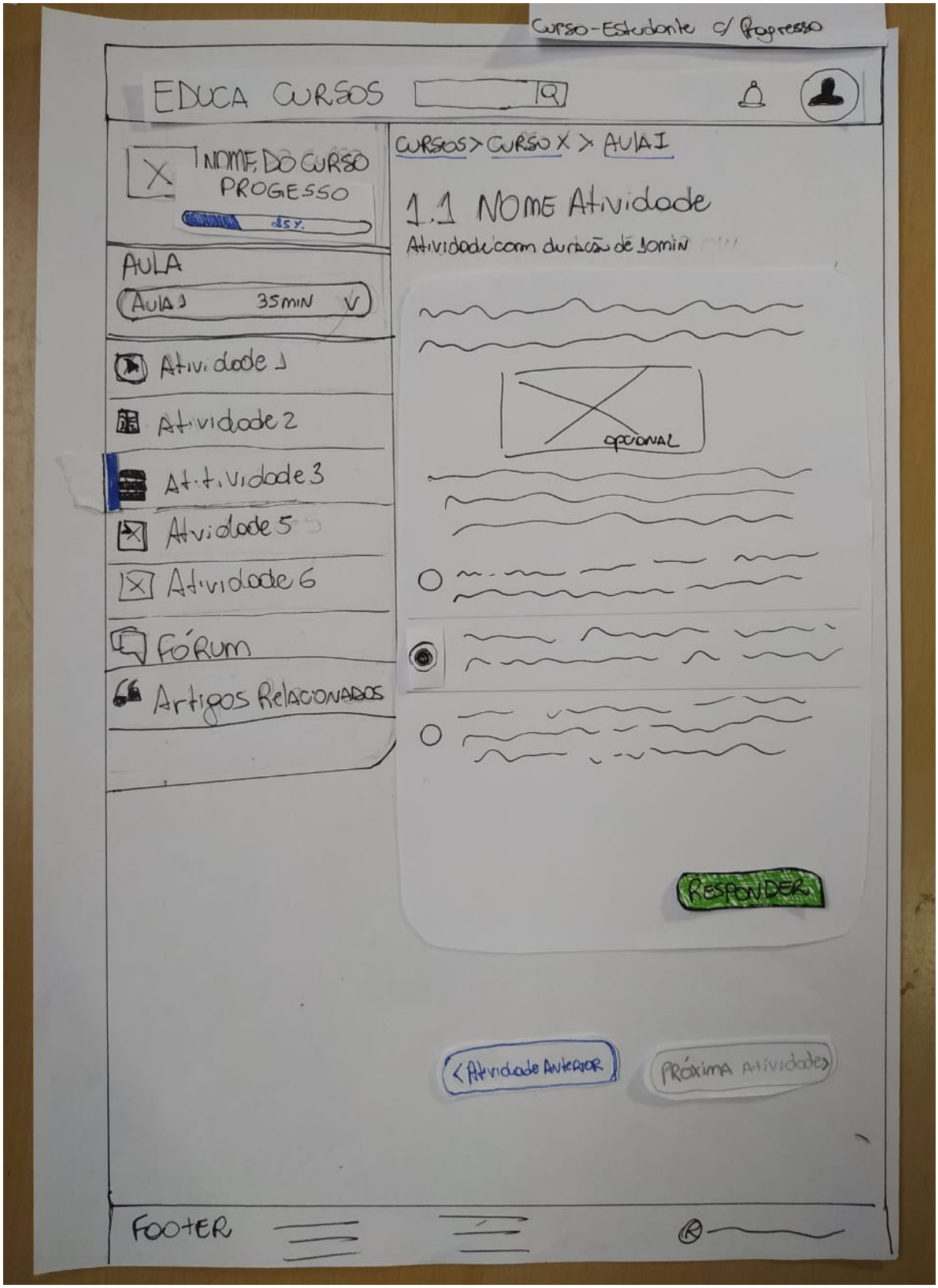
Fonte: Elaborado pelo autor.

Figura 165 – Modelo sketch da tela de atividades do tipo questionário para estudantes - parte 1



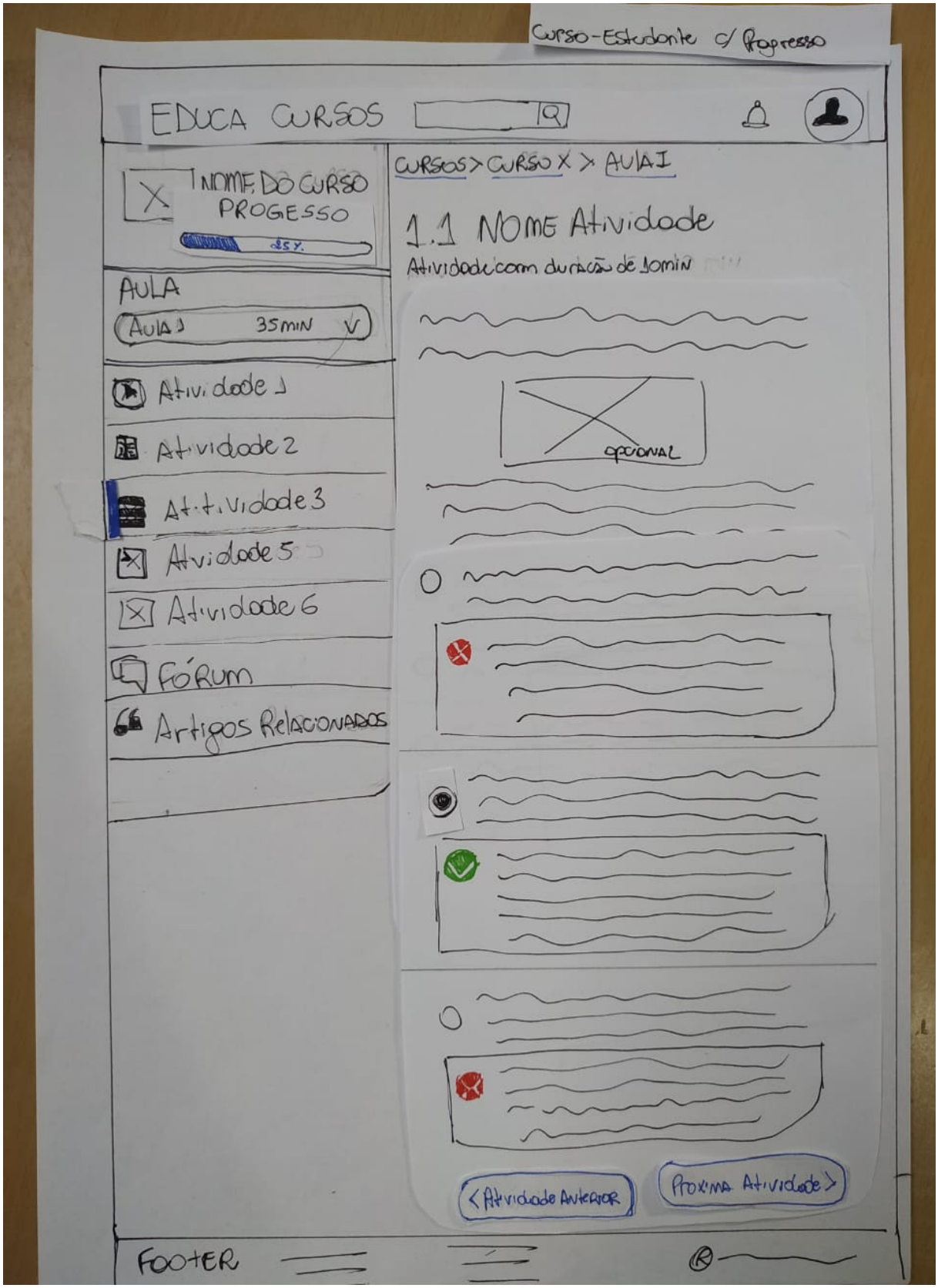
Fonte: Elaborado pelo autor.

Figura 166 – Modelo sketch da tela de atividades do tipo questionário para estudantes - parte 2



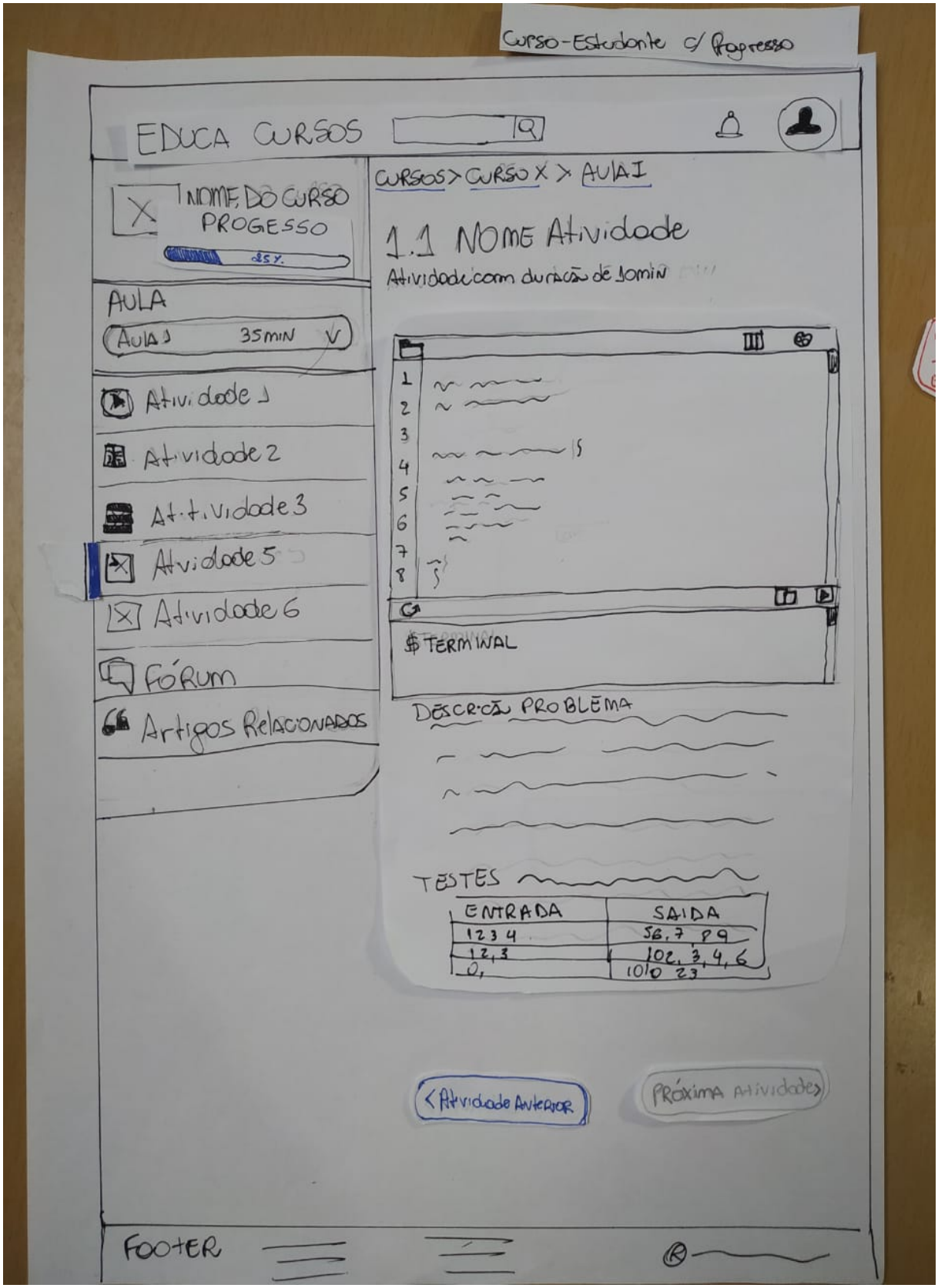
Fonte: Elaborado pelo autor.

Figura 167 – Modelo sketch da tela de atividades do tipo questionário para estudantes - parte 3



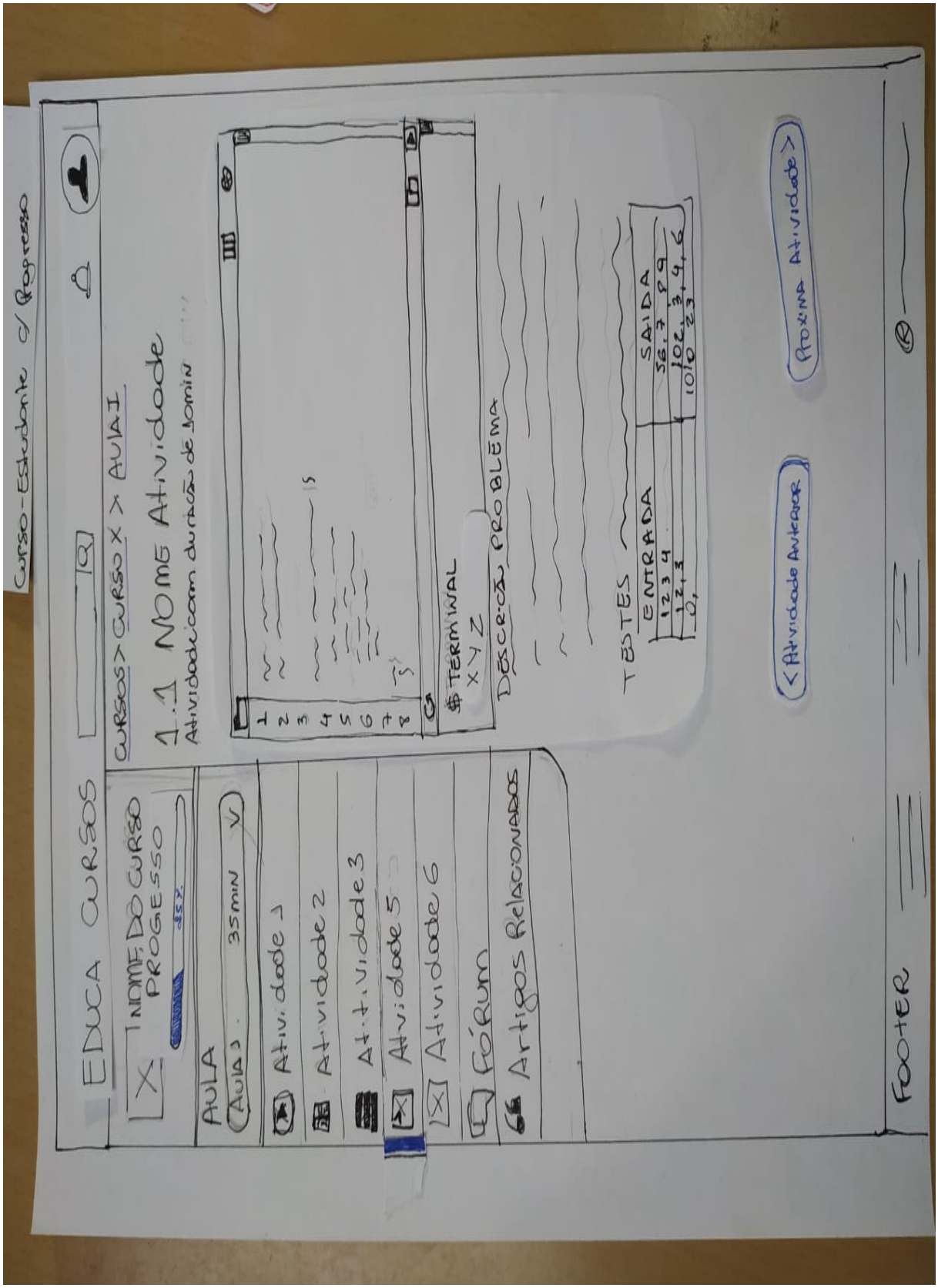
Fonte: Elaborado pelo autor.

Figura 168 – Modelo sketch da tela de atividades do tipo código para estudantes em modo de visualização horizontal - parte 1



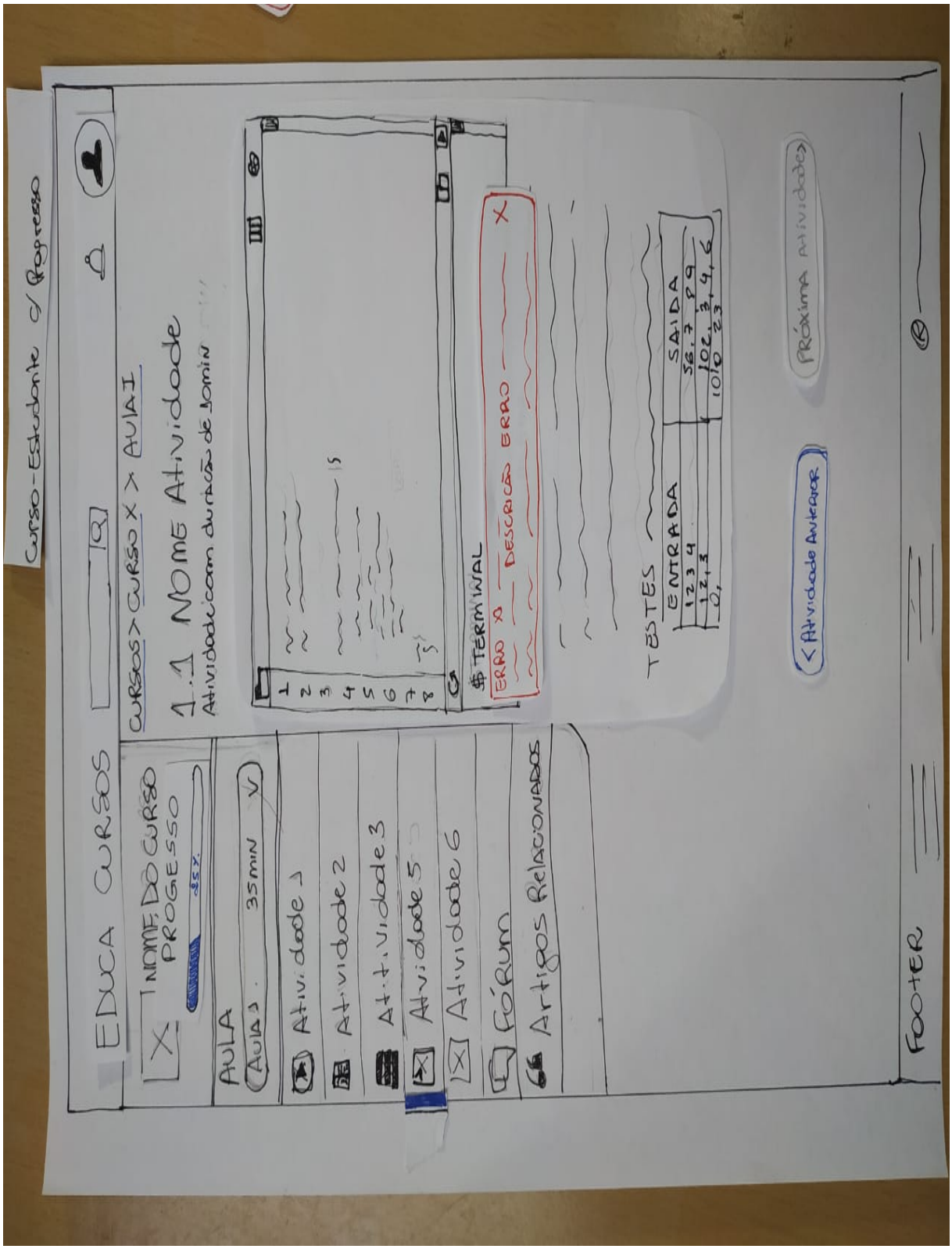
Fonte: Elaborado pelo autor.

Figura 169 – Modelo sketch da tela de atividades do tipo código para estudantes em modo de visualização horizontal - parte 2



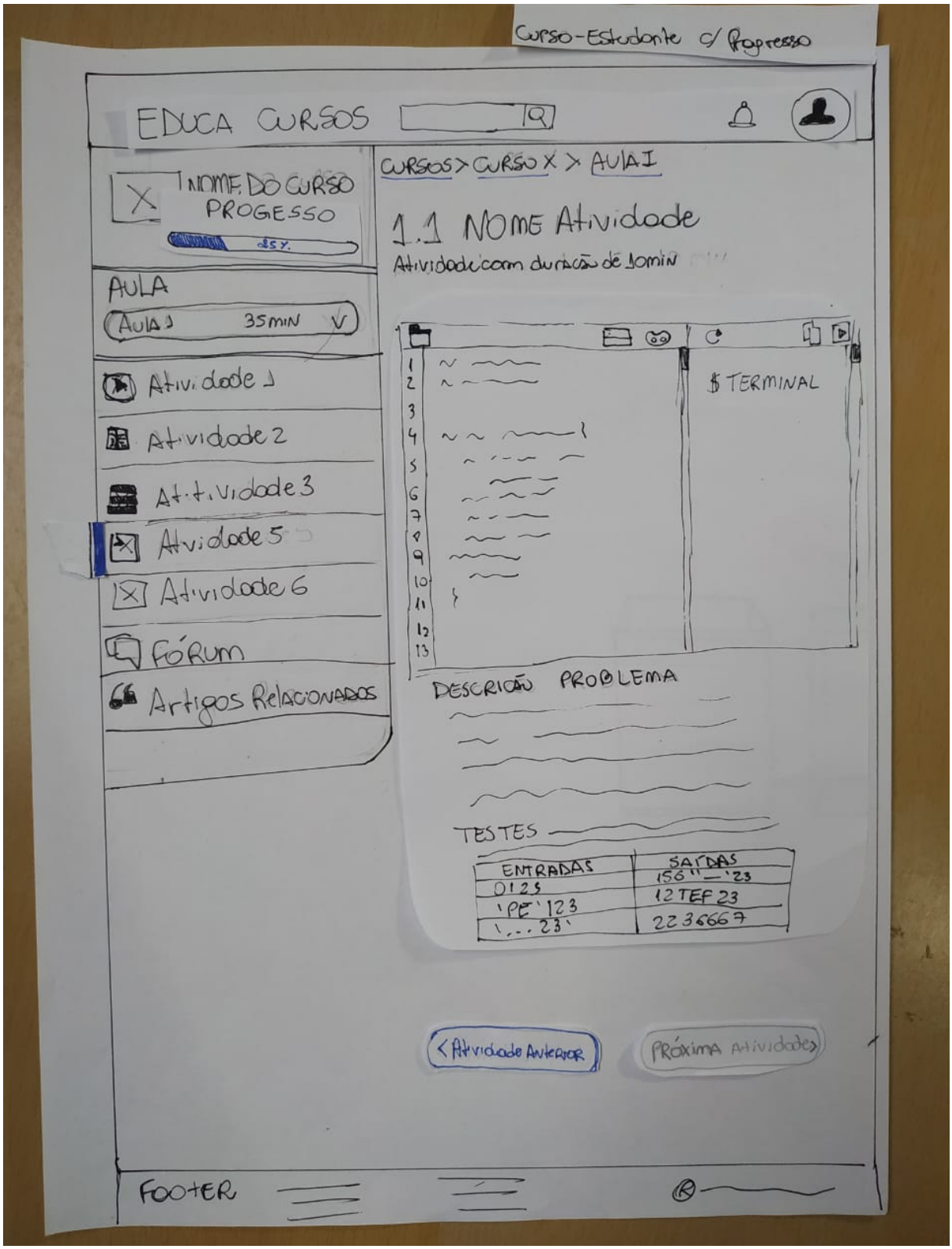
Fonte: Elaborado pelo autor.

Figura 170 – Modelo sketch da tela de atividades do tipo código para estudantes em situação de erro em modo de visualização horizontal - parte 2



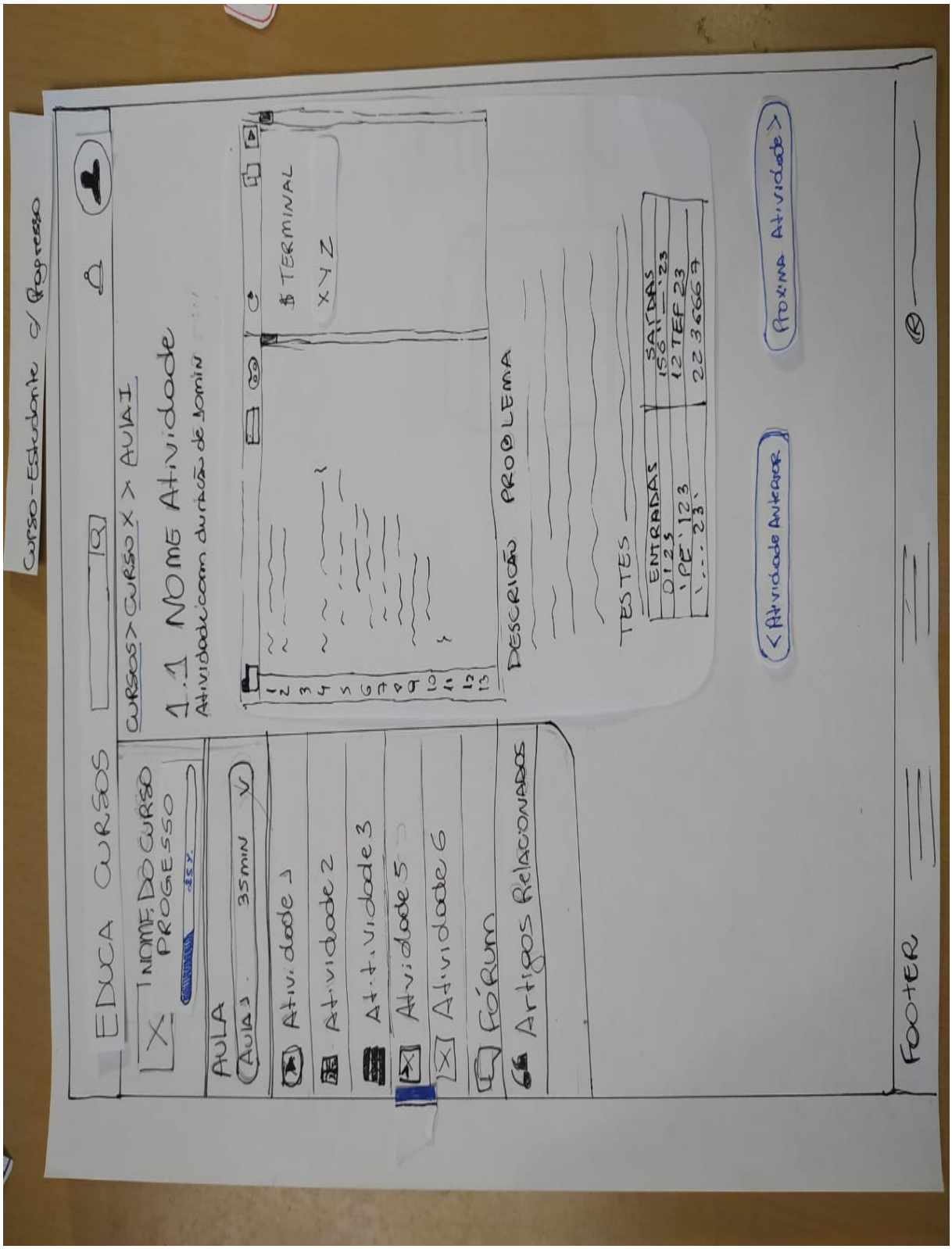
Fonte: Elaborado pelo autor.

Figura 171 – Modelo sketch da tela de atividades do tipo código para estudantes em modo de visualização vertical - parte 1



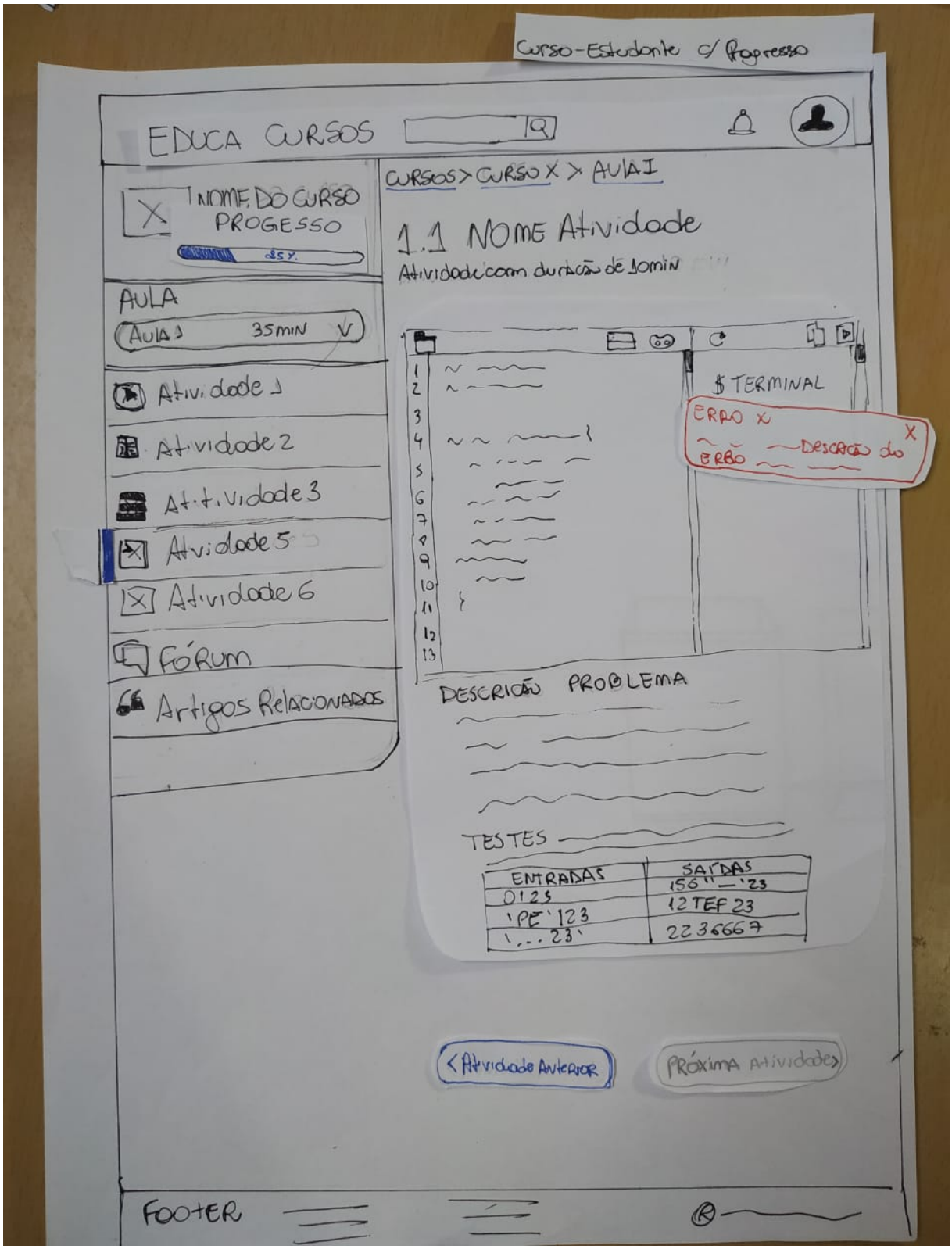
Fonte: Elaborado pelo autor.

Figura 172 – Modelo sketch da tela de atividades do tipo código para estudantes em modo de visualização vertical - parte 2



Fonte: Elaborado pelo autor.

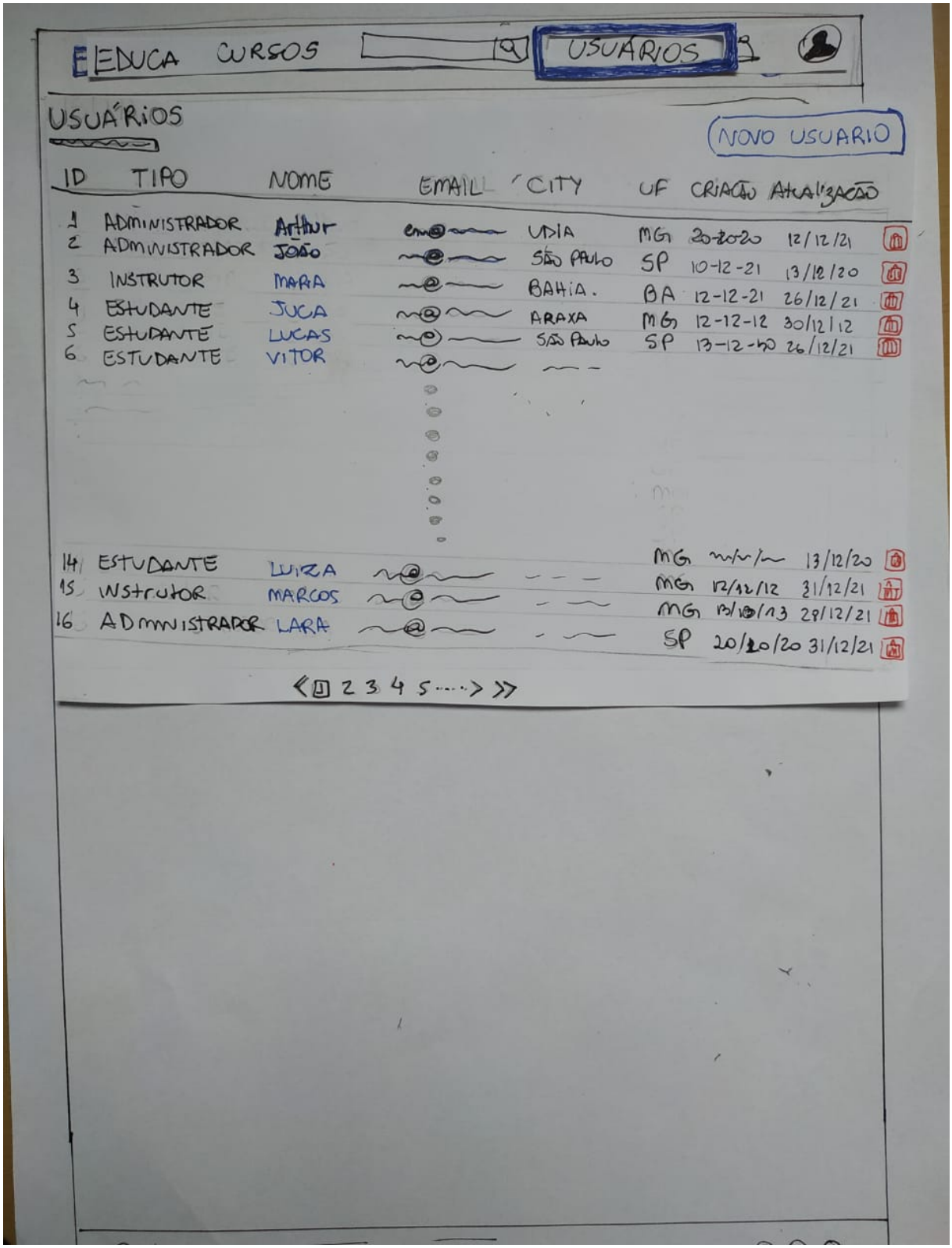
Figura 173 – Modelo sketch da tela de atividades do tipo código para estudantes em situação de erro em modo de visualização vertical - parte 2



Fonte: Elaborado pelo autor.

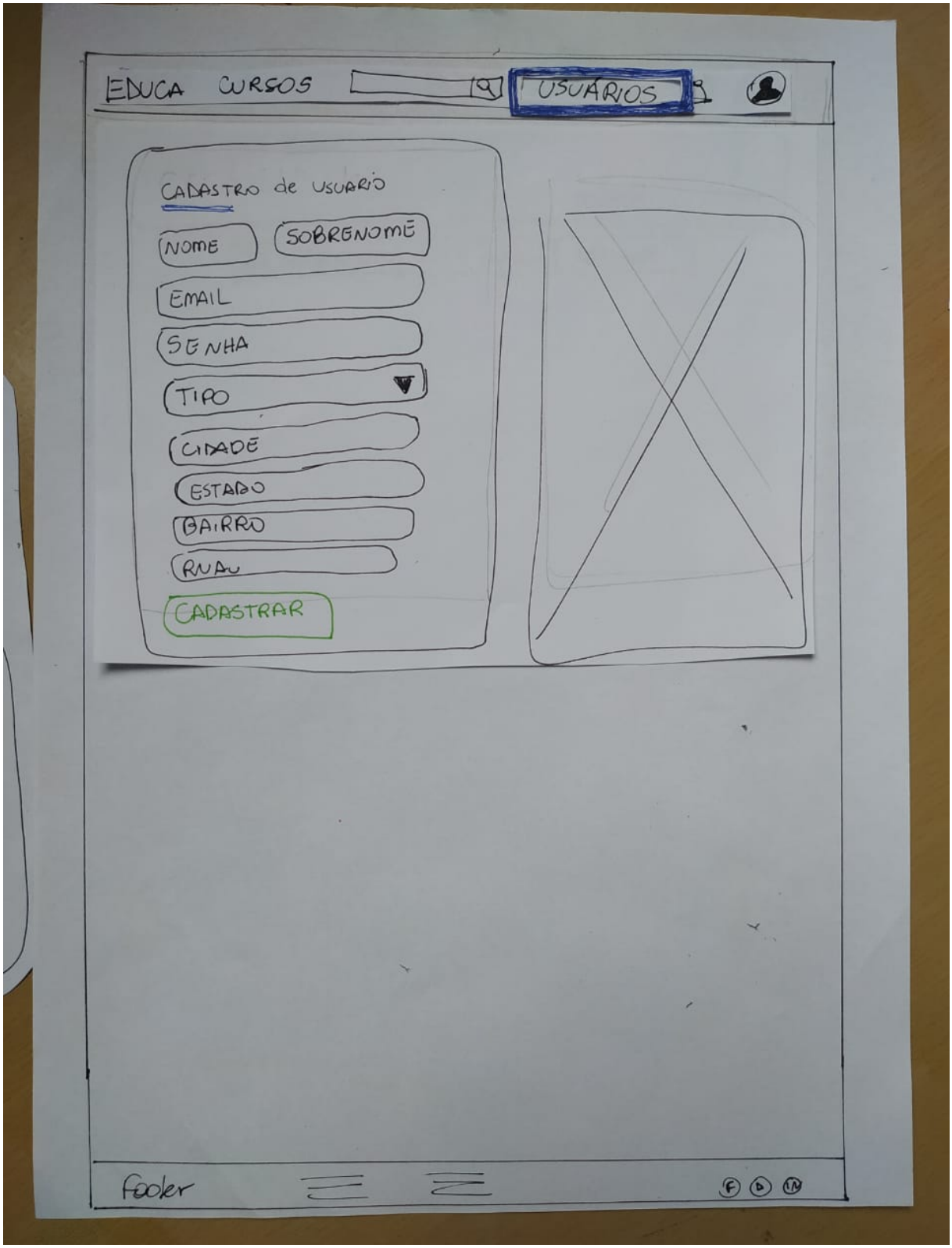
D.2.9 Administrativo

Figura 174 – Modelo sketch da tela de usuários para administrador



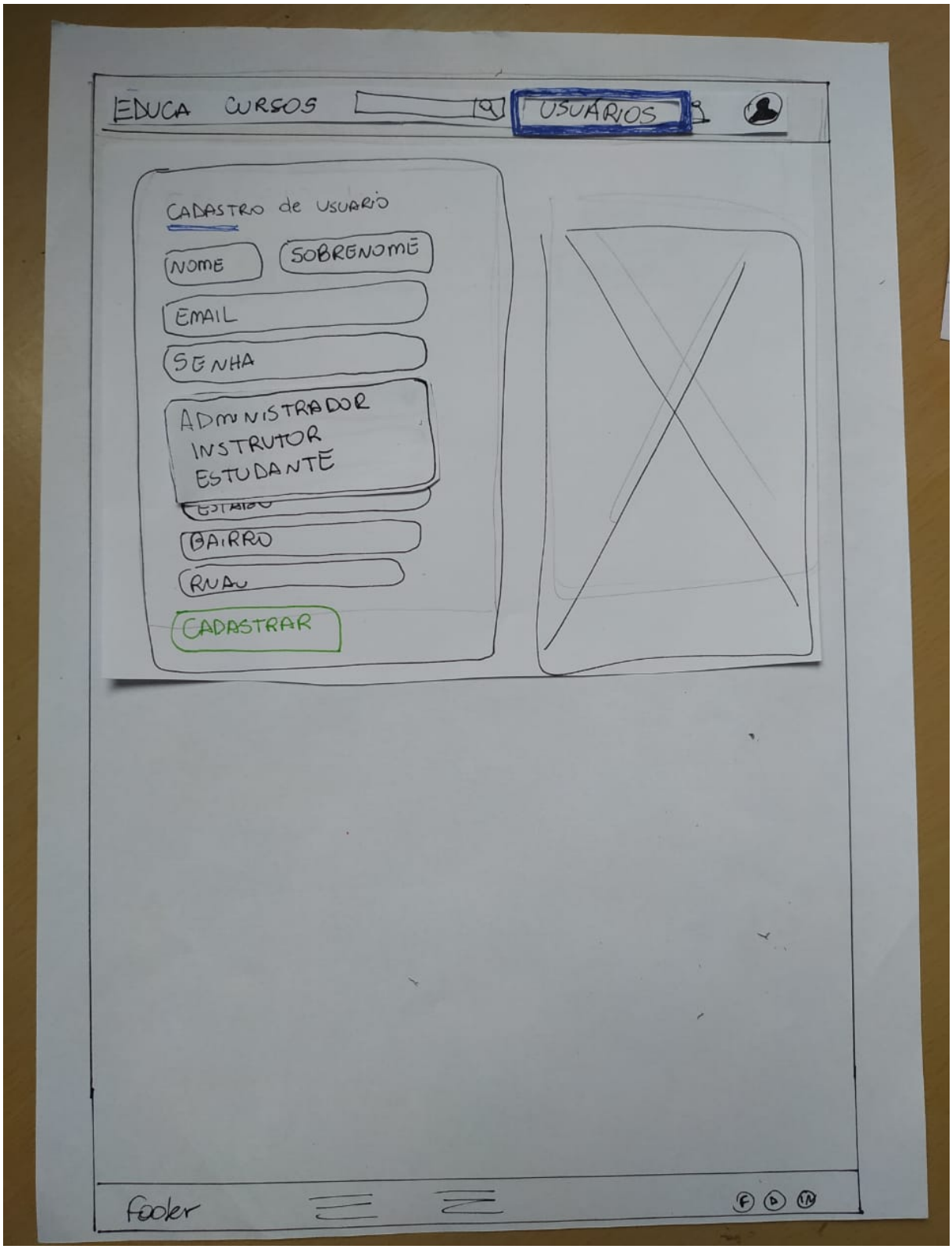
Fonte: Elaborado pelo autor.

Figura 175 – Modelo sketch da tela de cadastro de usuários para administrador



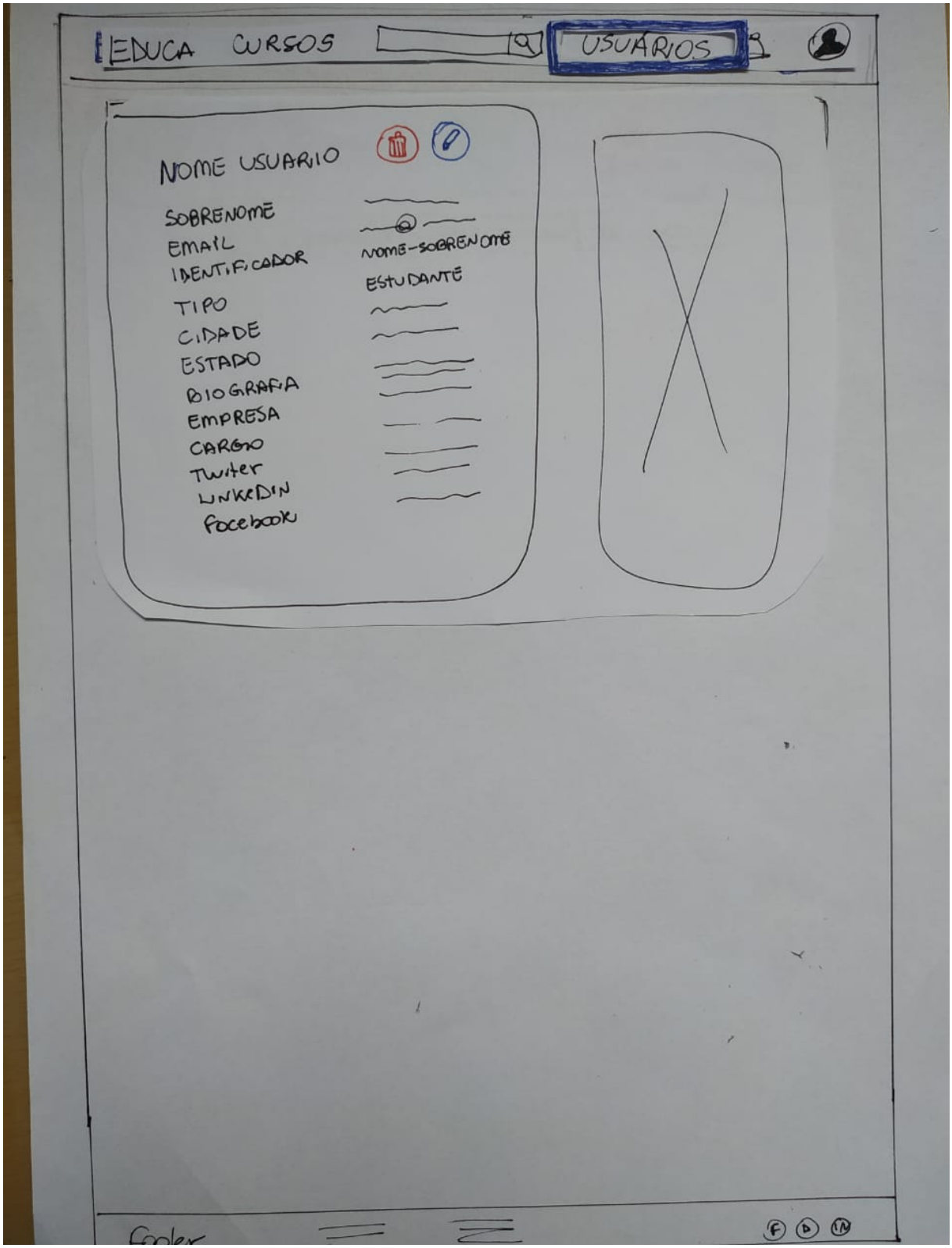
Fonte: Elaborado pelo autor.

Figura 176 – Modelo sketch da tela de cadastro de usuários para administrador ao selecionar tipo



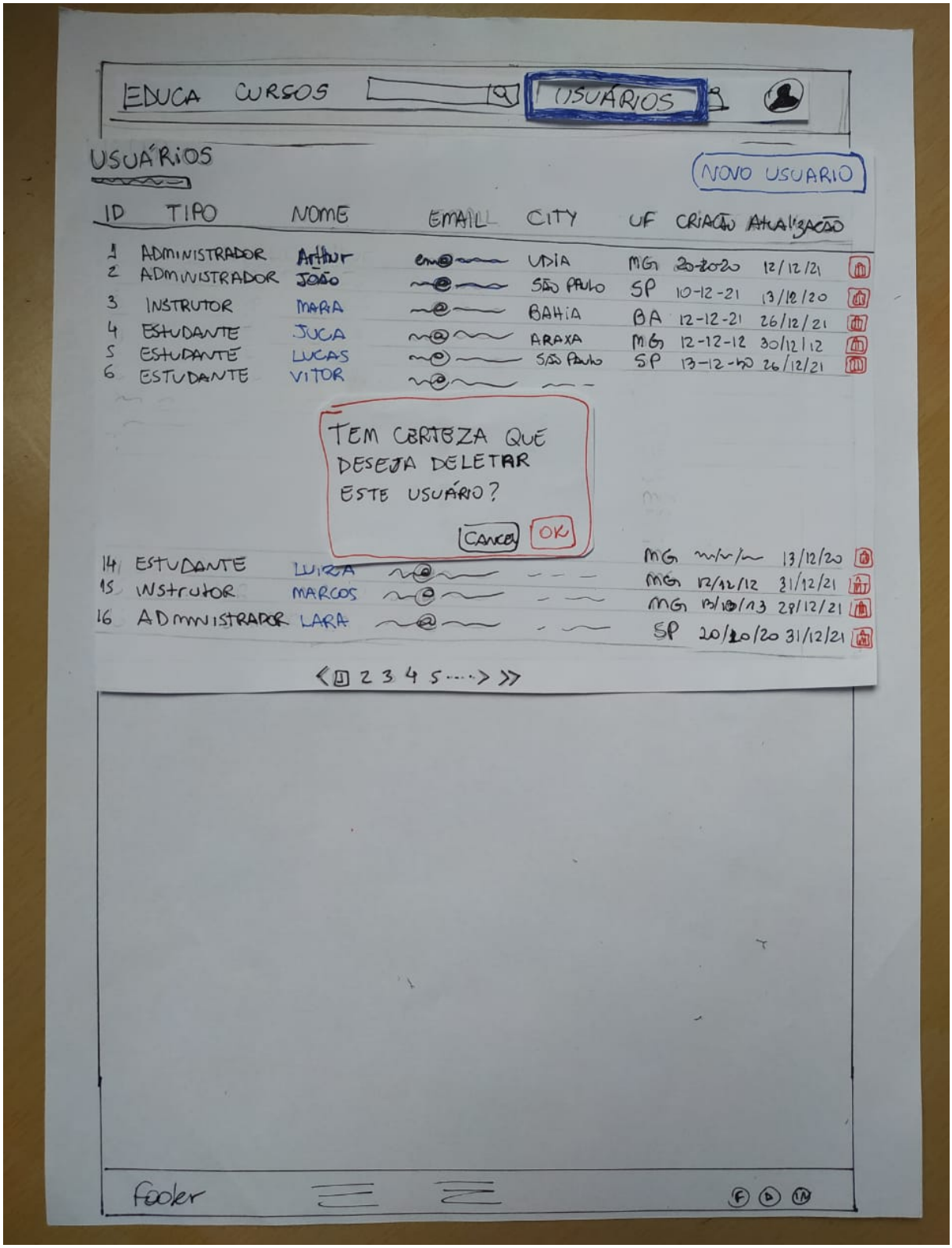
Fonte: Elaborado pelo autor.

Figura 177 – Modelo sketch da tela de detalhes de usuários para administrador



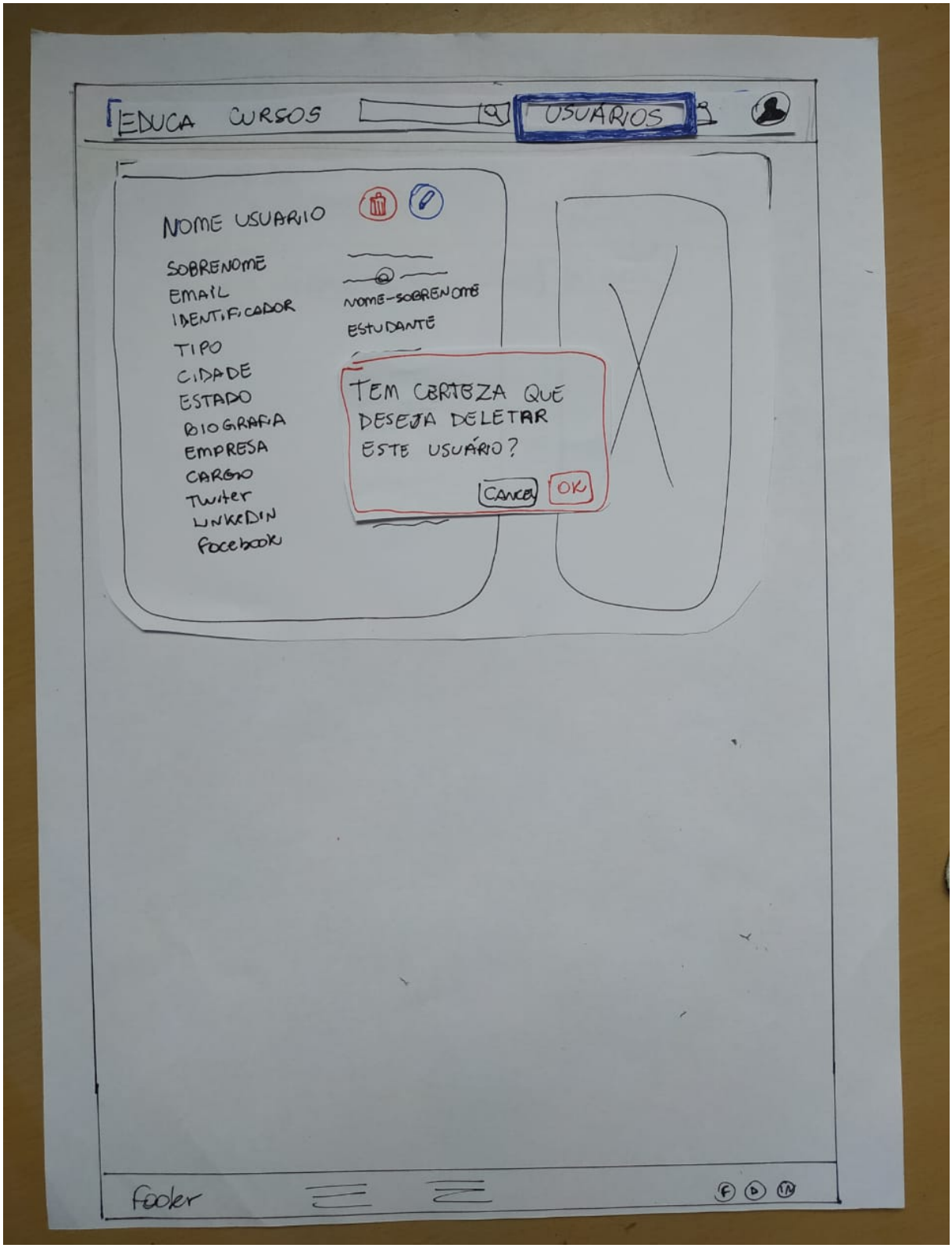
Fonte: Elaborado pelo autor.

Figura 178 – Modelo sketch da confirmação de deleção usuários por listagem para administrador



Fonte: Elaborado pelo autor.

Figura 179 – Modelo sketch da confirmação de deleção de usuários para administrador via tela de detalhes do usuário

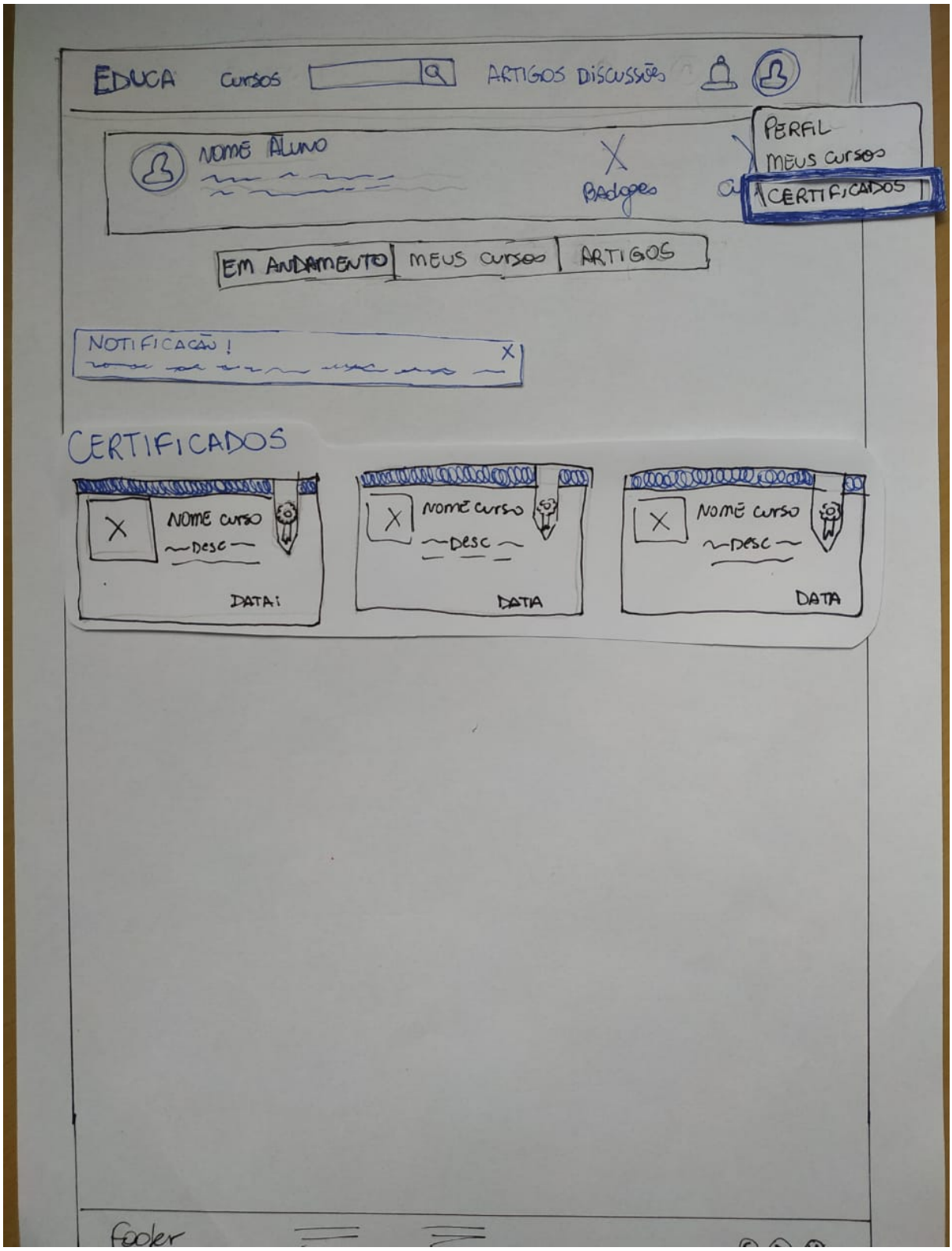


Fonte: Elaborado pelo autor.

D.2.10 Certificado

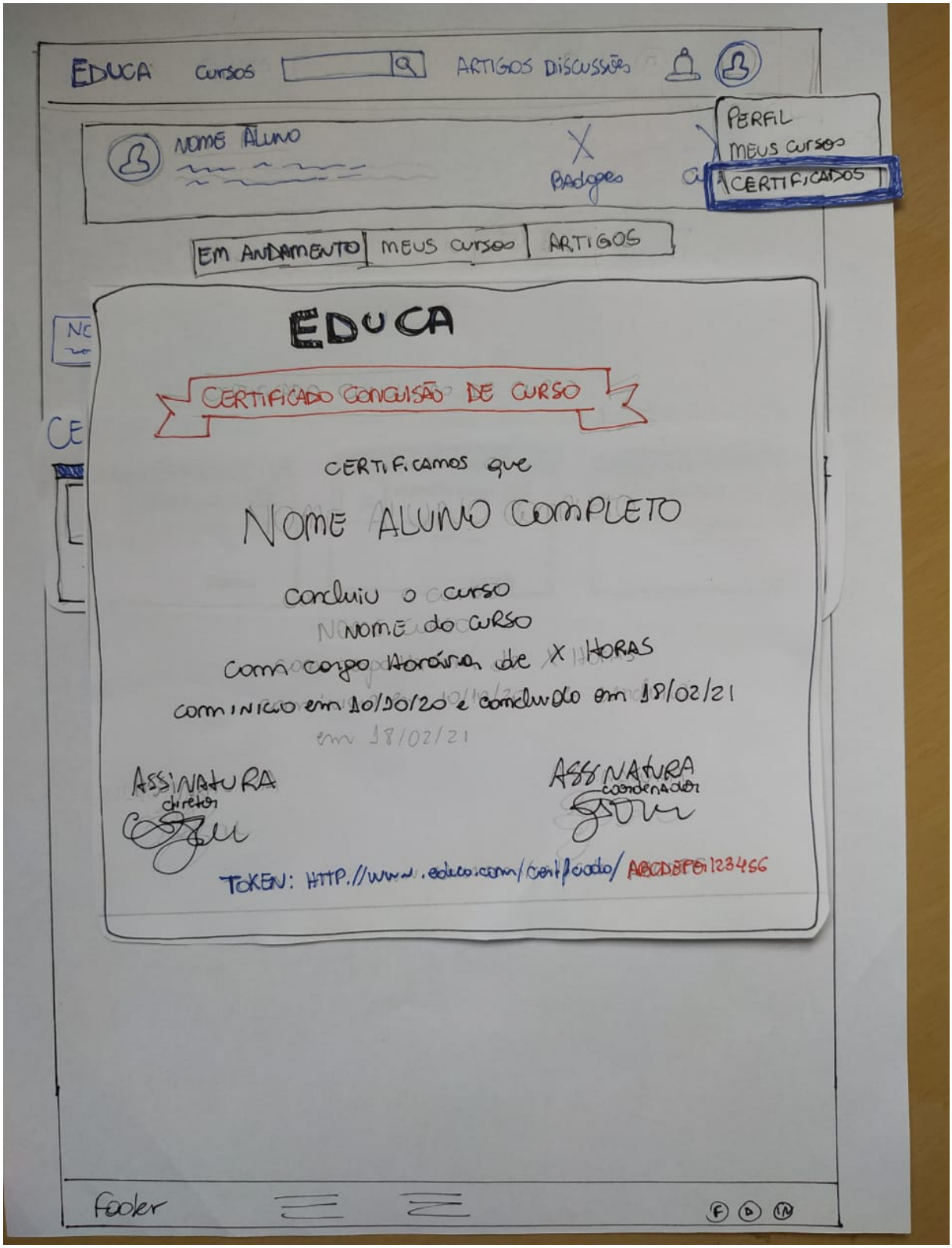
D.2.10.1 Estudante

Figura 180 – Modelo sketch da tela de listagem de certificados do estudante



Fonte: Elaborado pelo autor.

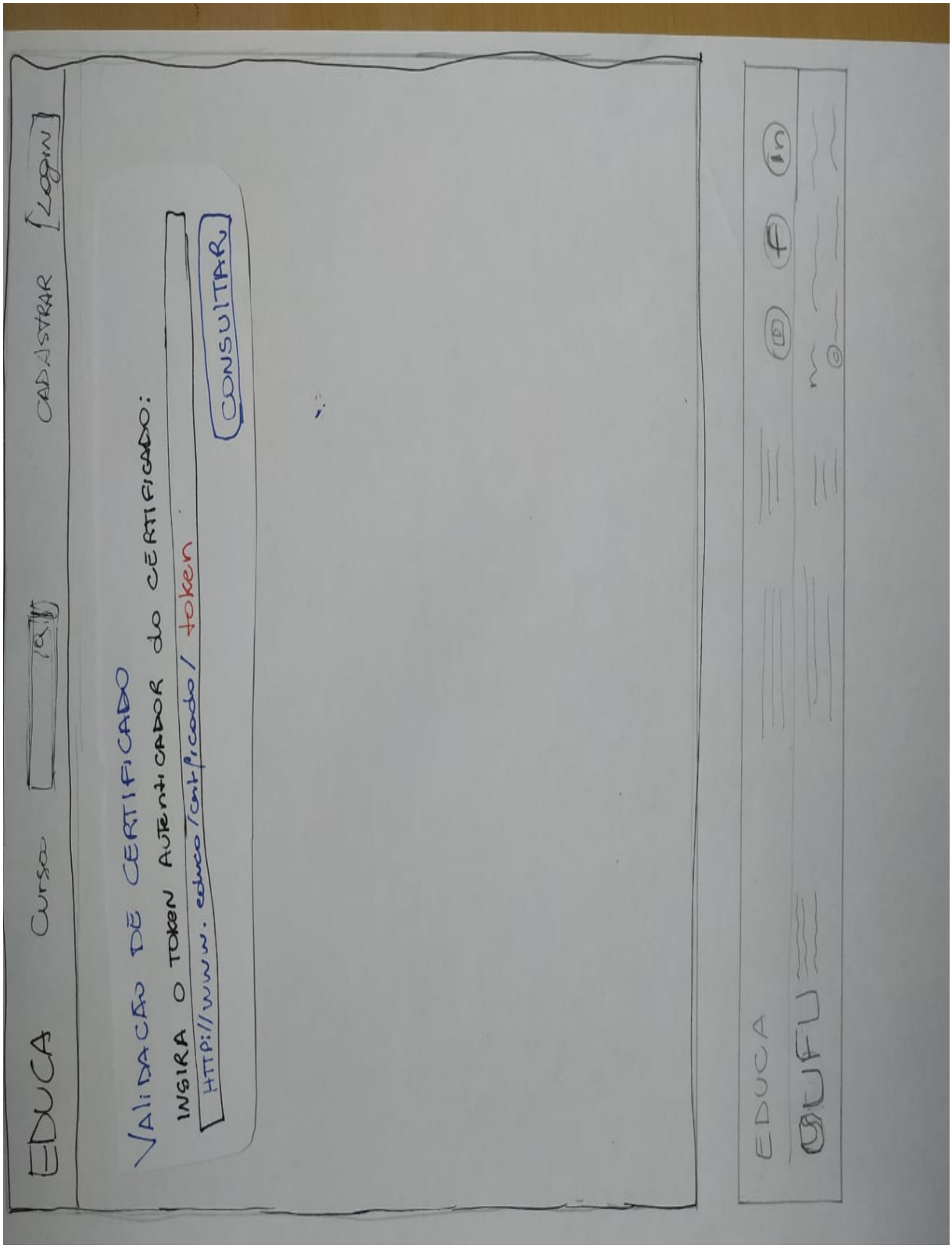
Figura 181 – Modelo sketch da tela do certificado



Fonte: Elaborado pelo autor.

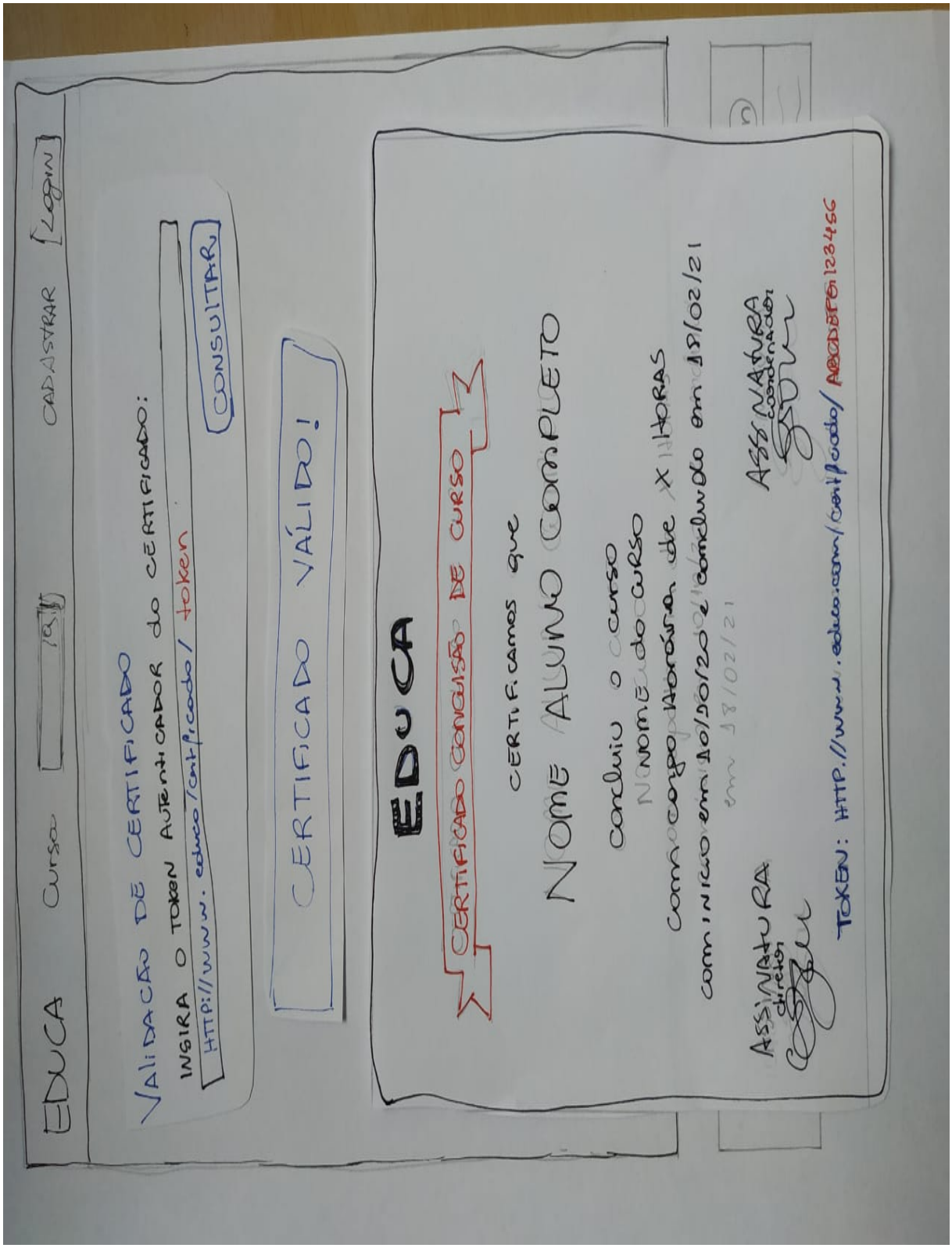
D.2.10.2 Validação

Figura 182 – Modelo sketch da tela de validação de certificados



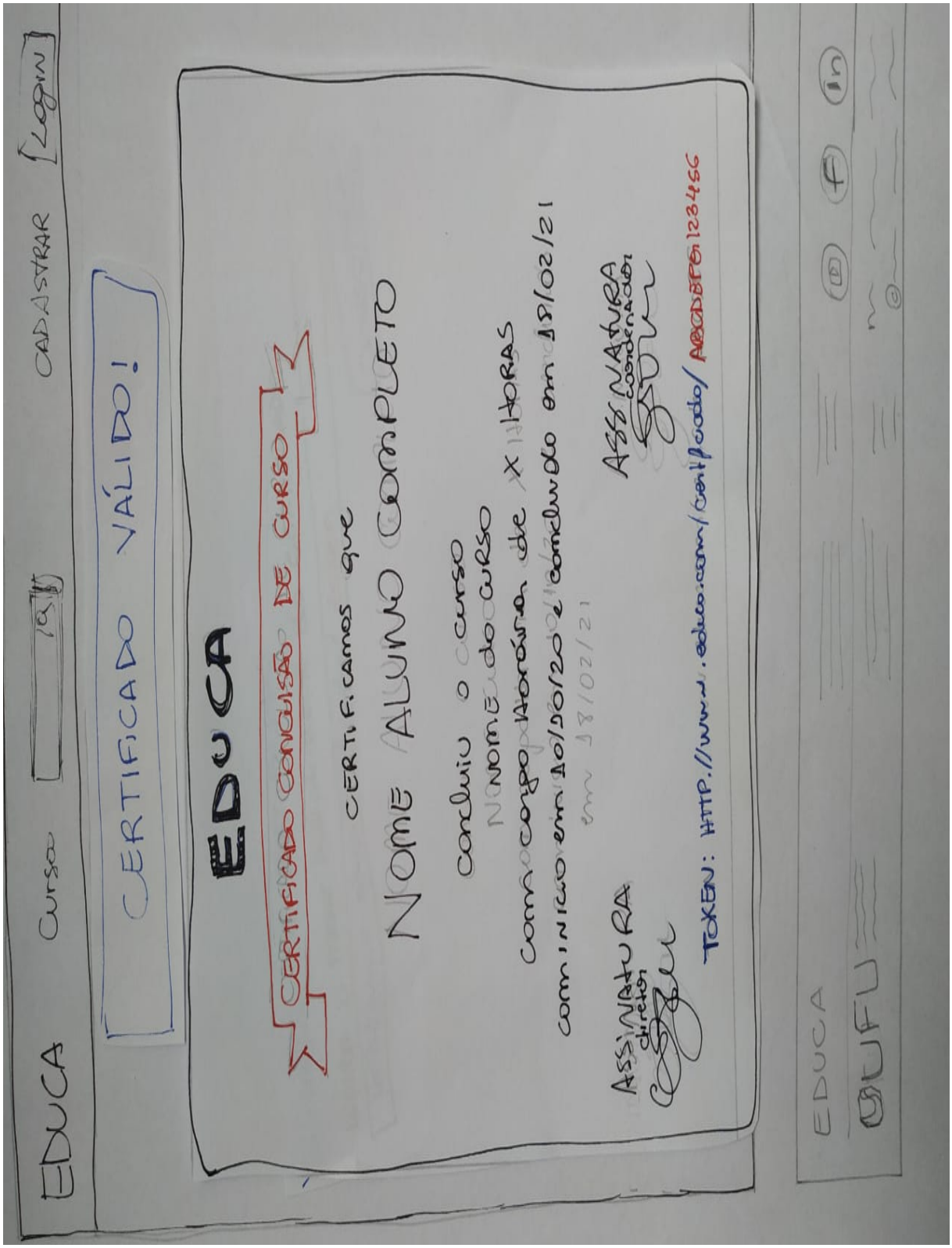
Fonte: Elaborado pelo autor.

Figura 183 – Modelo sketch da tela de validação de certificados - parte 1



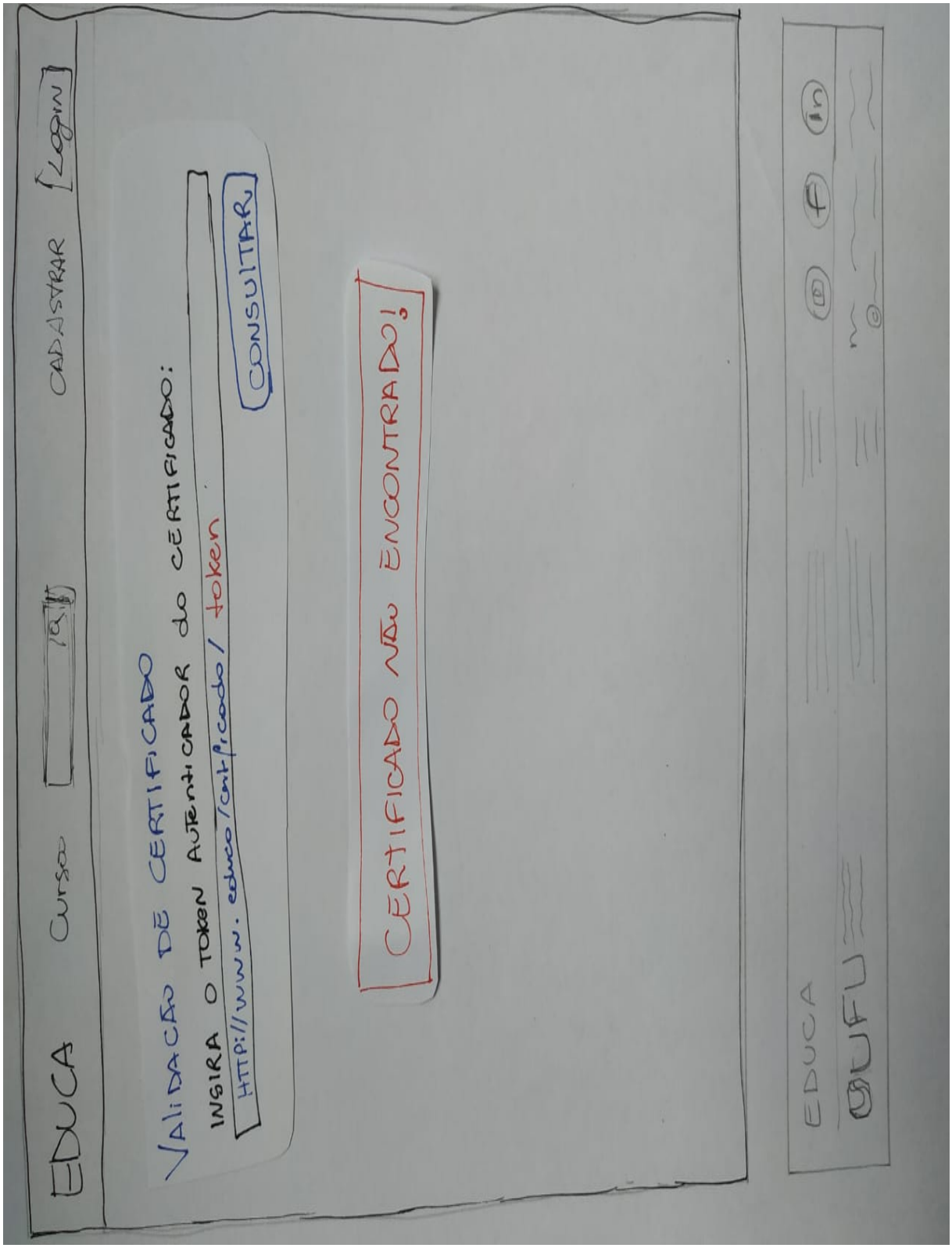
Fonte: Elaborado pelo autor.

Figura 184 – Modelo sketch da tela de validação de certificados - parte 2



Fonte: Elaborado pelo autor.

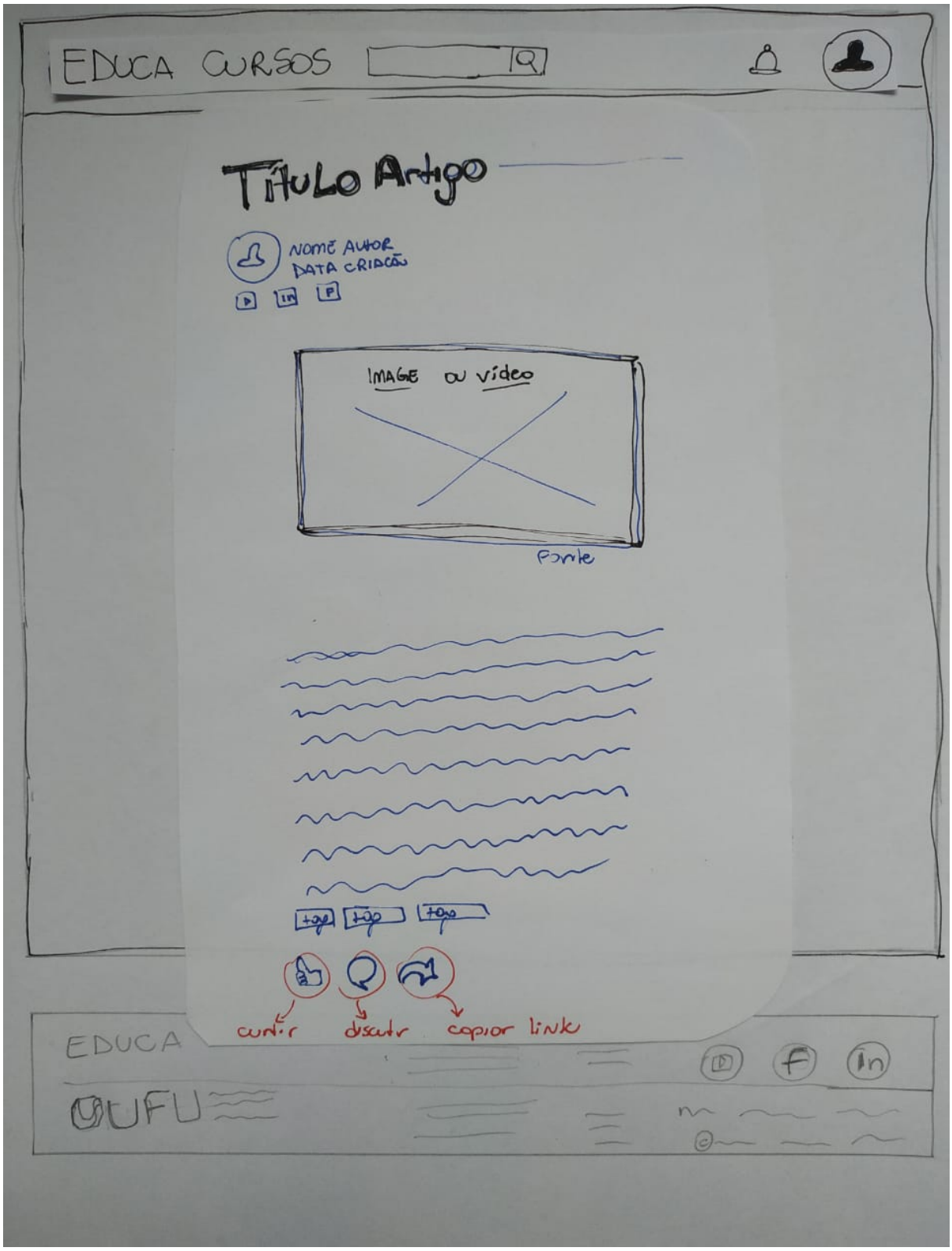
Figura 185 – Modelo sketch da tela de validação de certificados situacao invalida



Fonte: Elaborado pelo autor.

D.2.11 Artigo

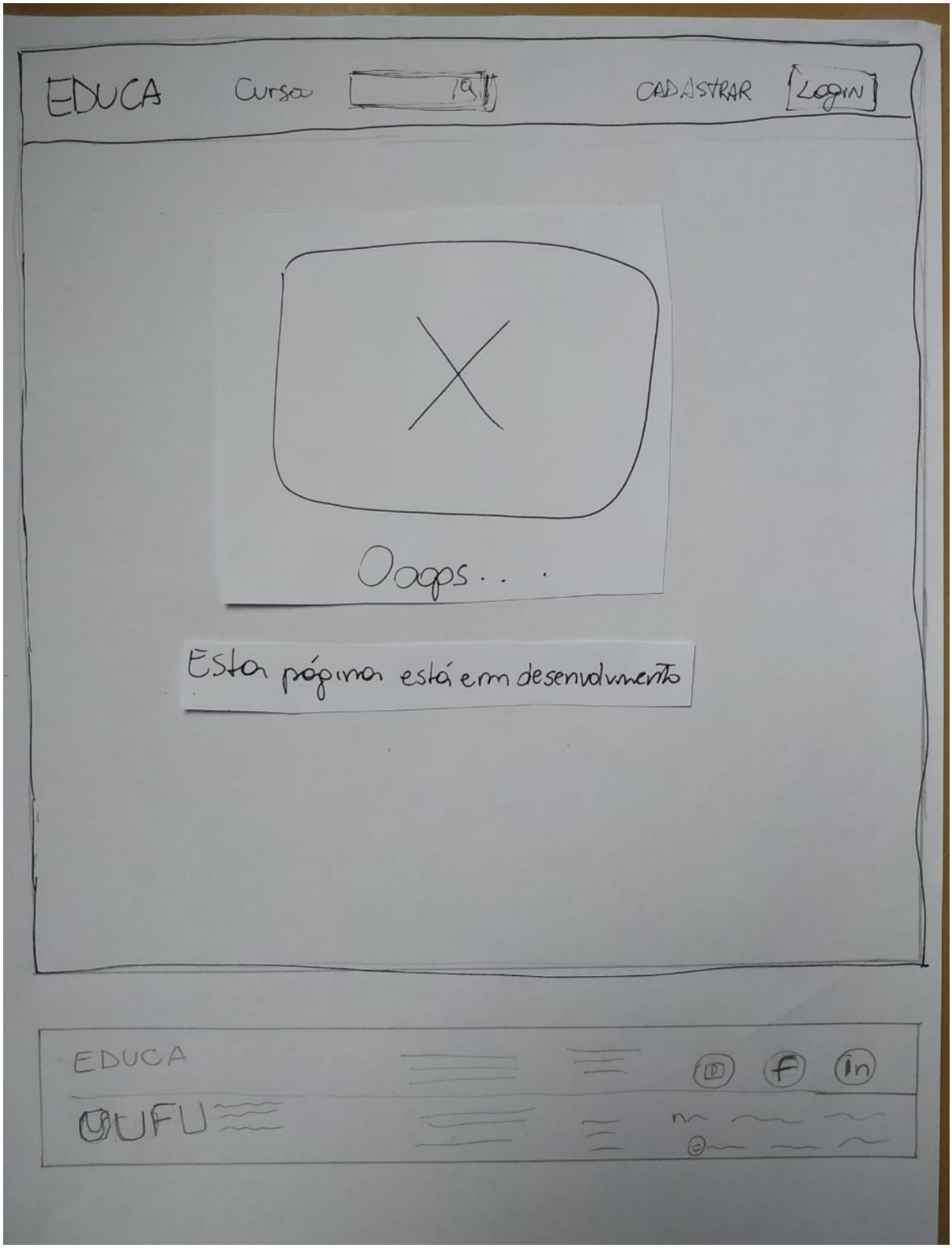
Figura 186 – Modelo sketch de artigos



Fonte: Elaborado pelo autor.

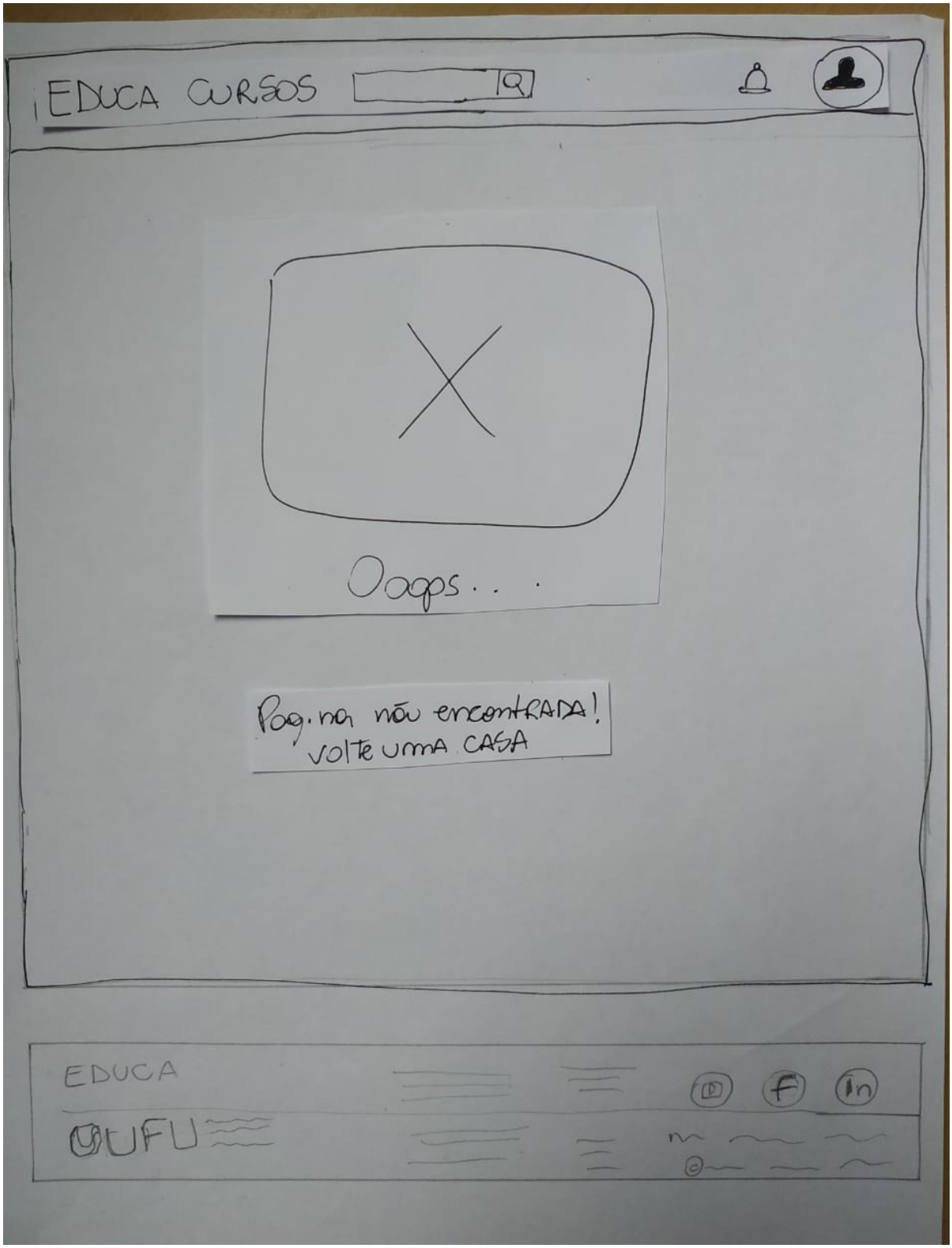
D.2.12 Tratativas

Figura 187 – Modelo sketch de página em desenvolvimento



Fonte: Elaborado pelo autor.

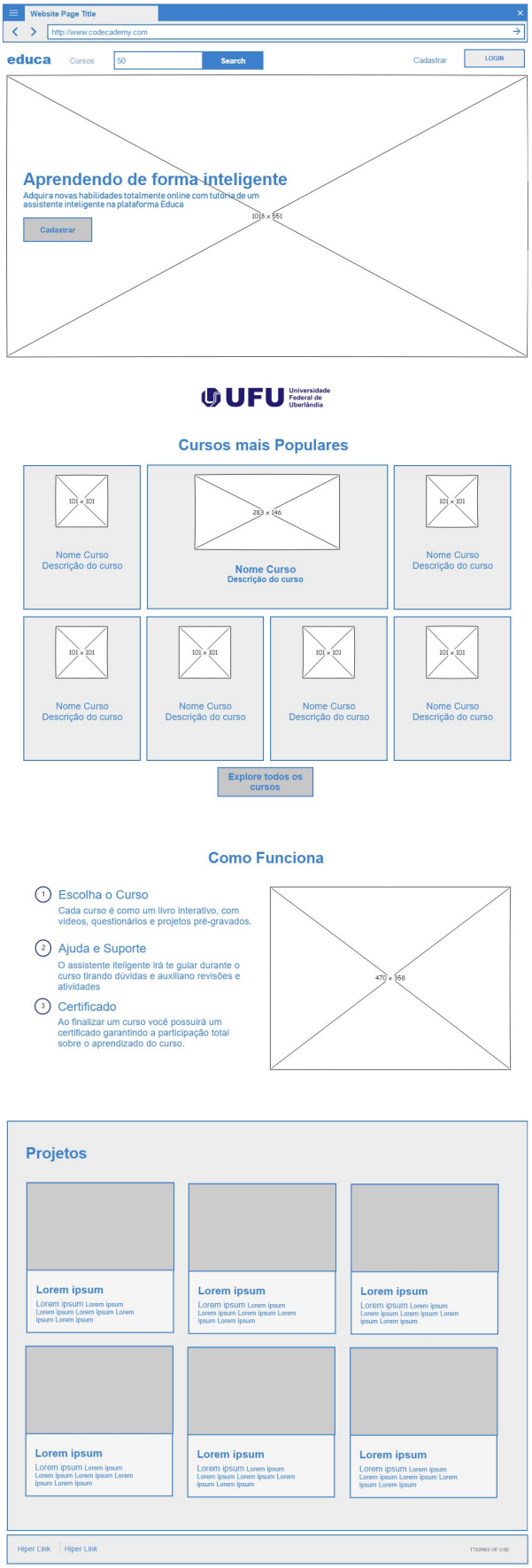
Figura 188 – Modelo sketch de página não encontrada



Fonte: Elaborado pelo autor.

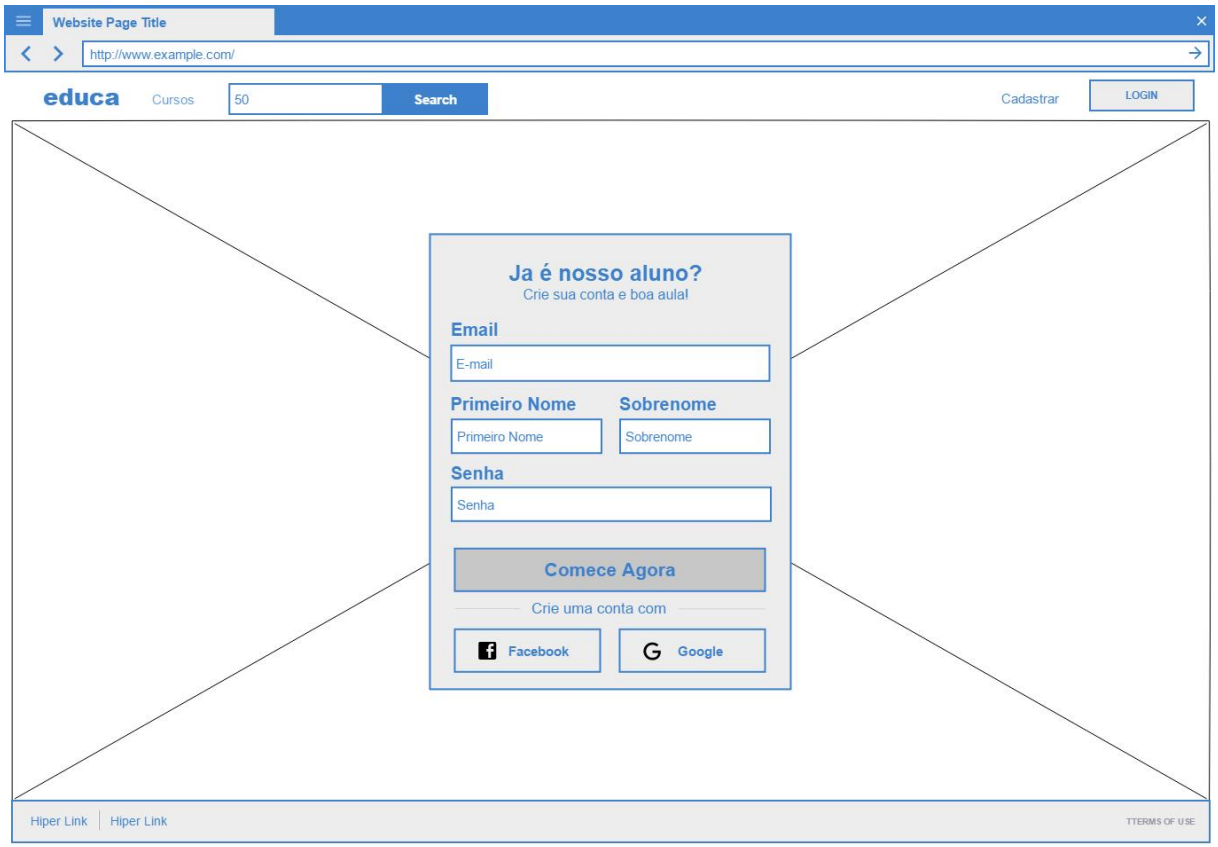
D.3 Wireframes

Figura 189 – Wireframe da tela home



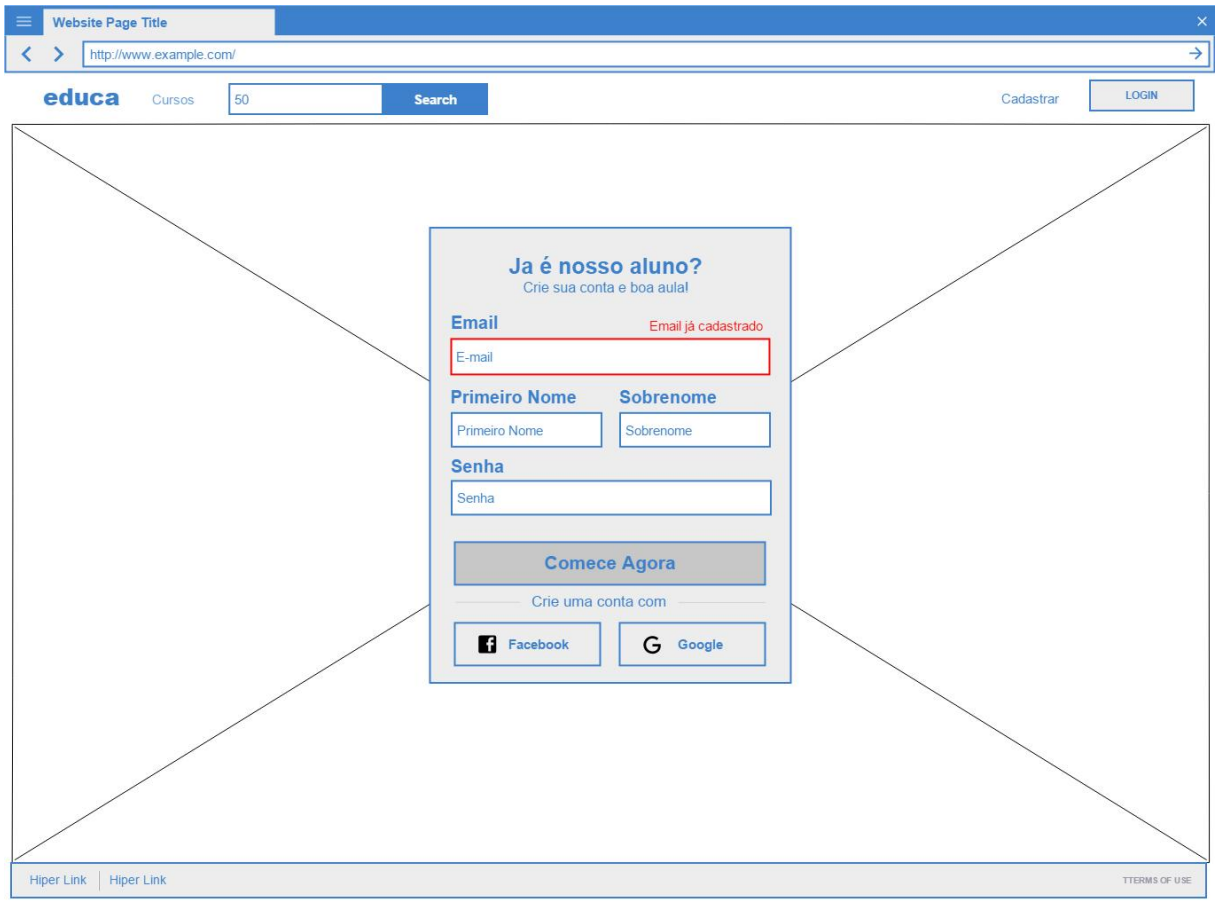
Fonte: Elaborado pelo autor.

Figura 190 – Wireframe da tela de cadastro



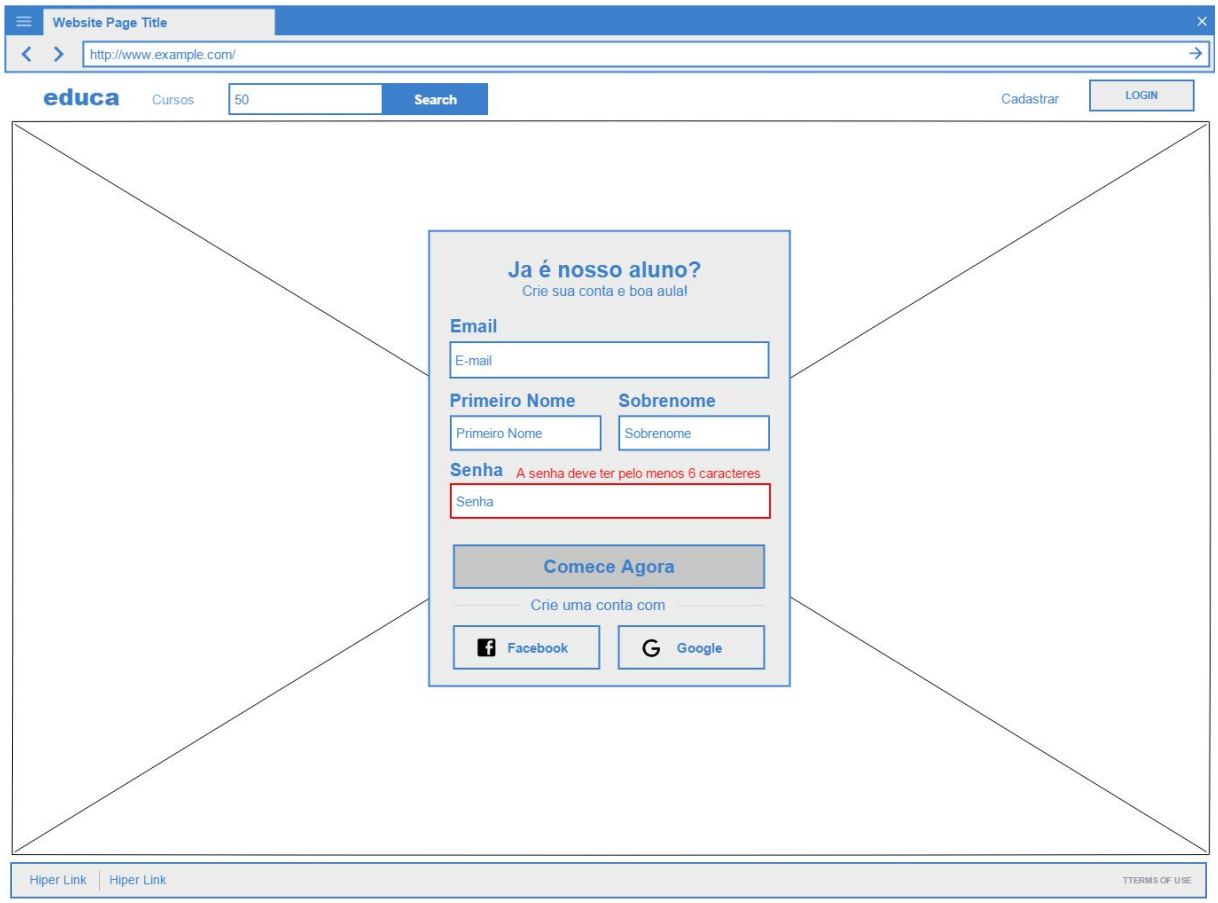
Fonte: Elaborado pelo autor.

Figura 191 – Wireframe da tela de cadastro em situação de email ja cadastrado



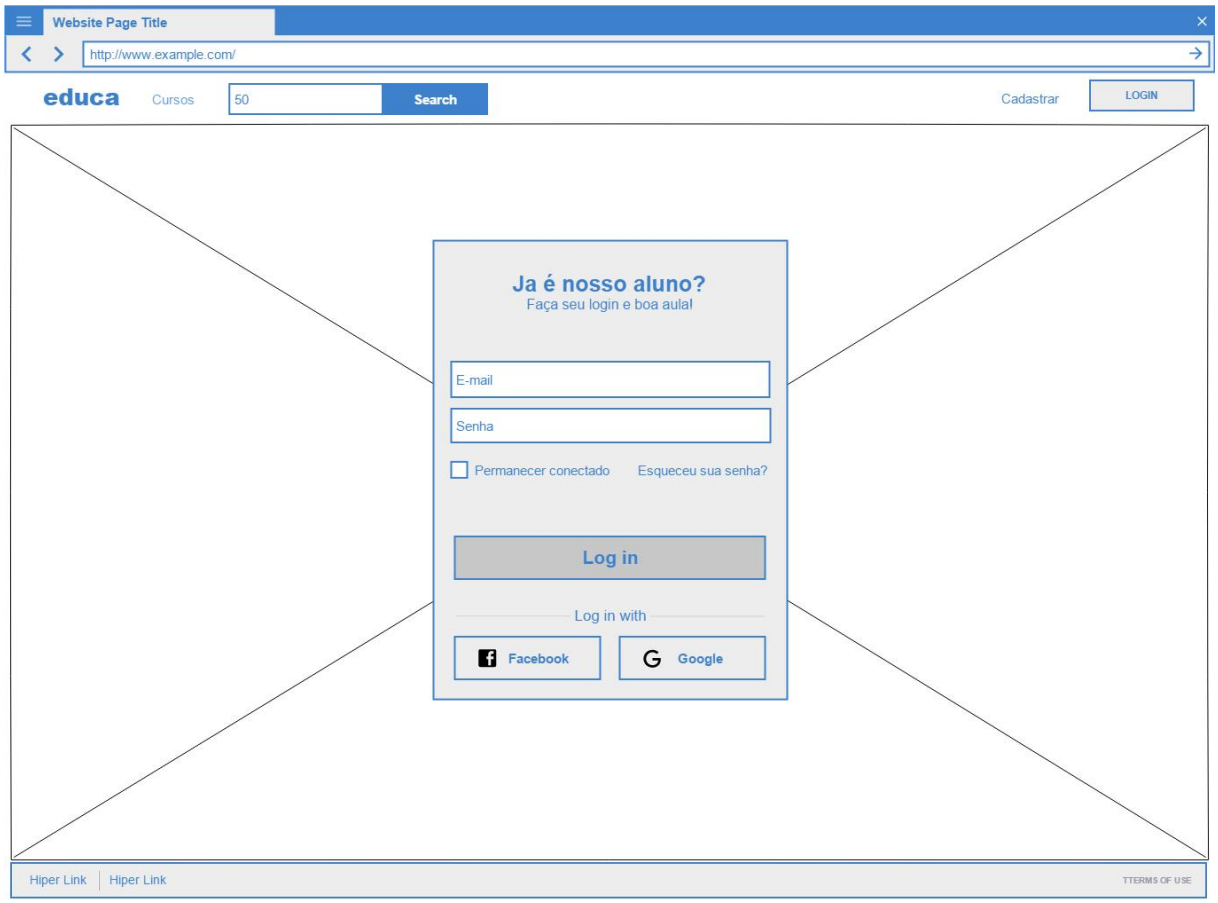
Fonte: Elaborado pelo autor.

Figura 192 – Wireframe da tela de cadastro em situação de senha não atender os requisitos



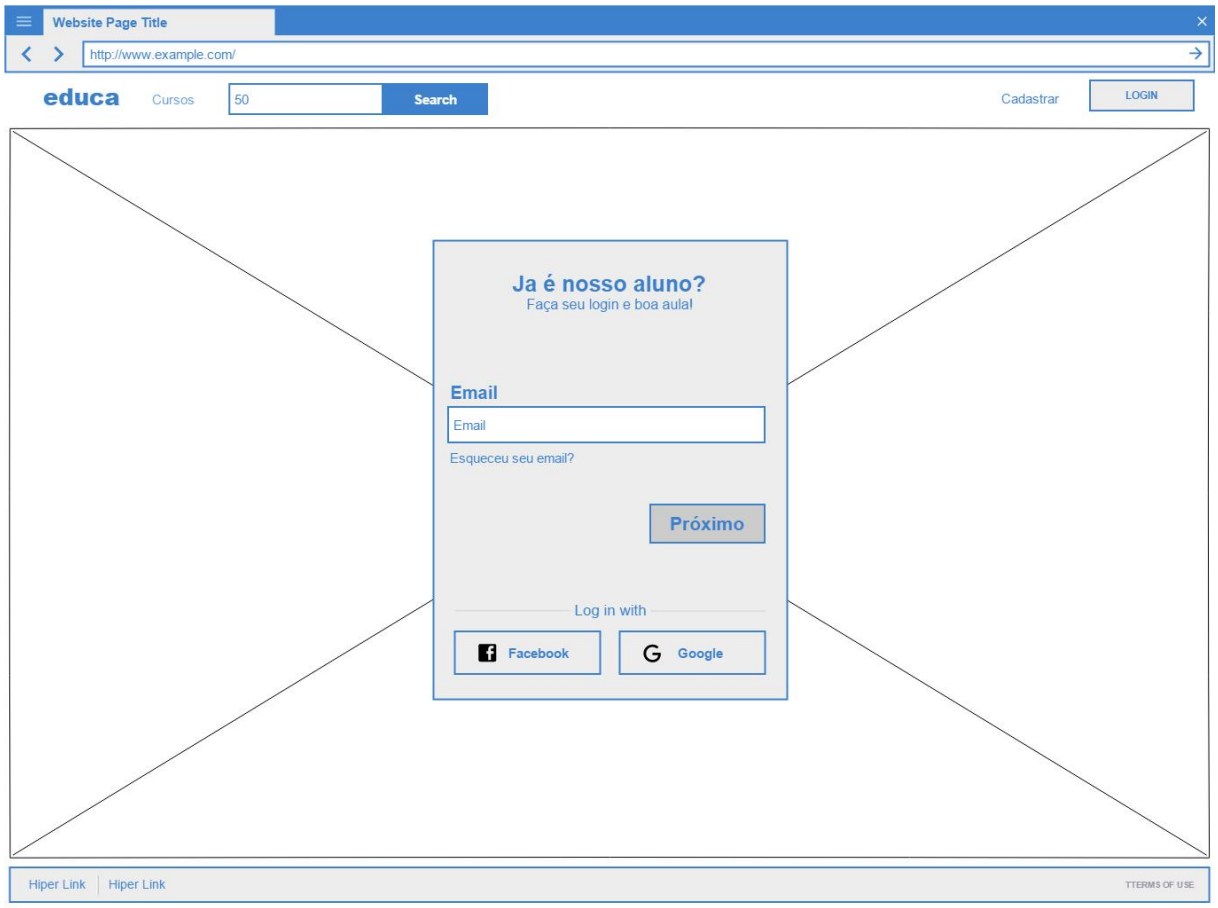
Fonte: Elaborado pelo autor.

Figura 193 – Wireframe da tela de login



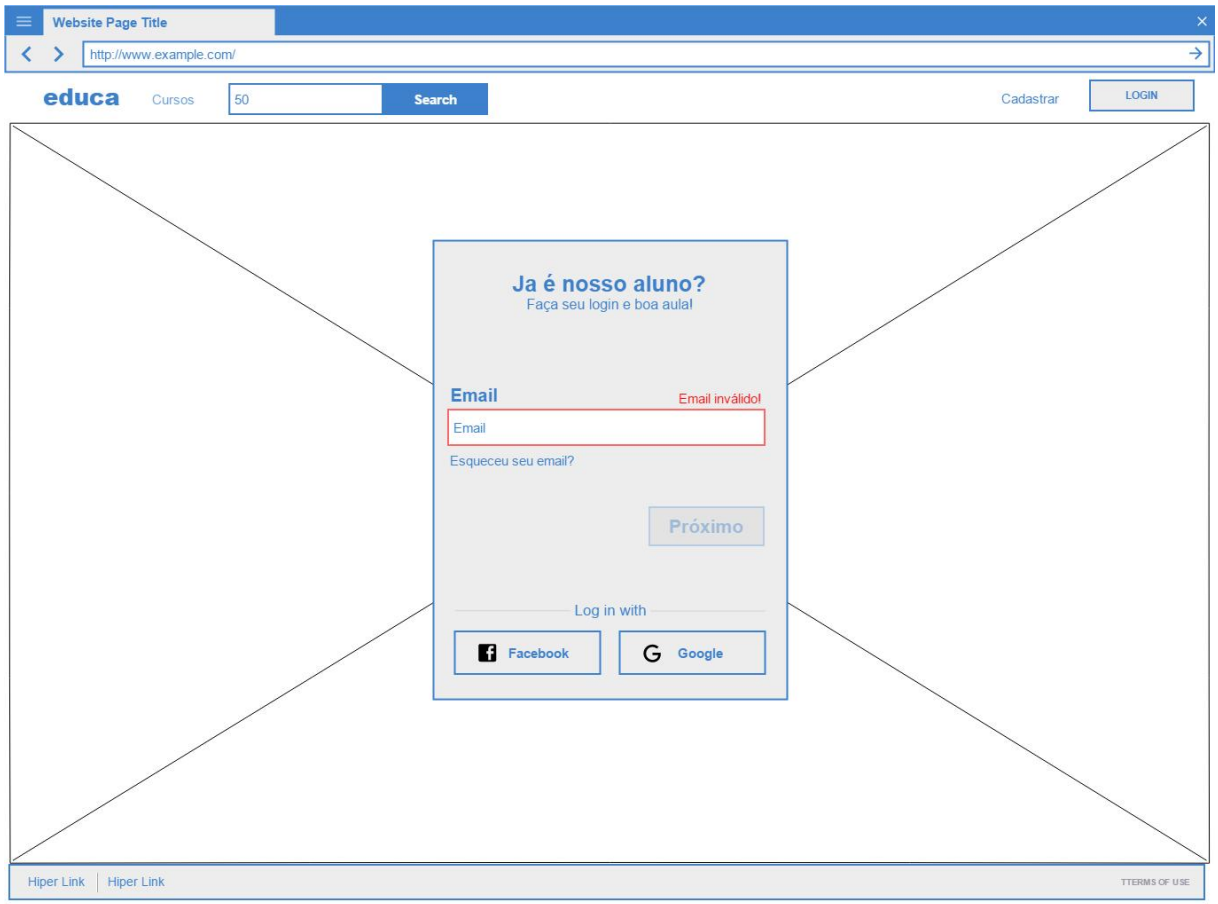
Fonte: Elaborado pelo autor.

Figura 194 – Wireframe da tela de login fase de inserir email



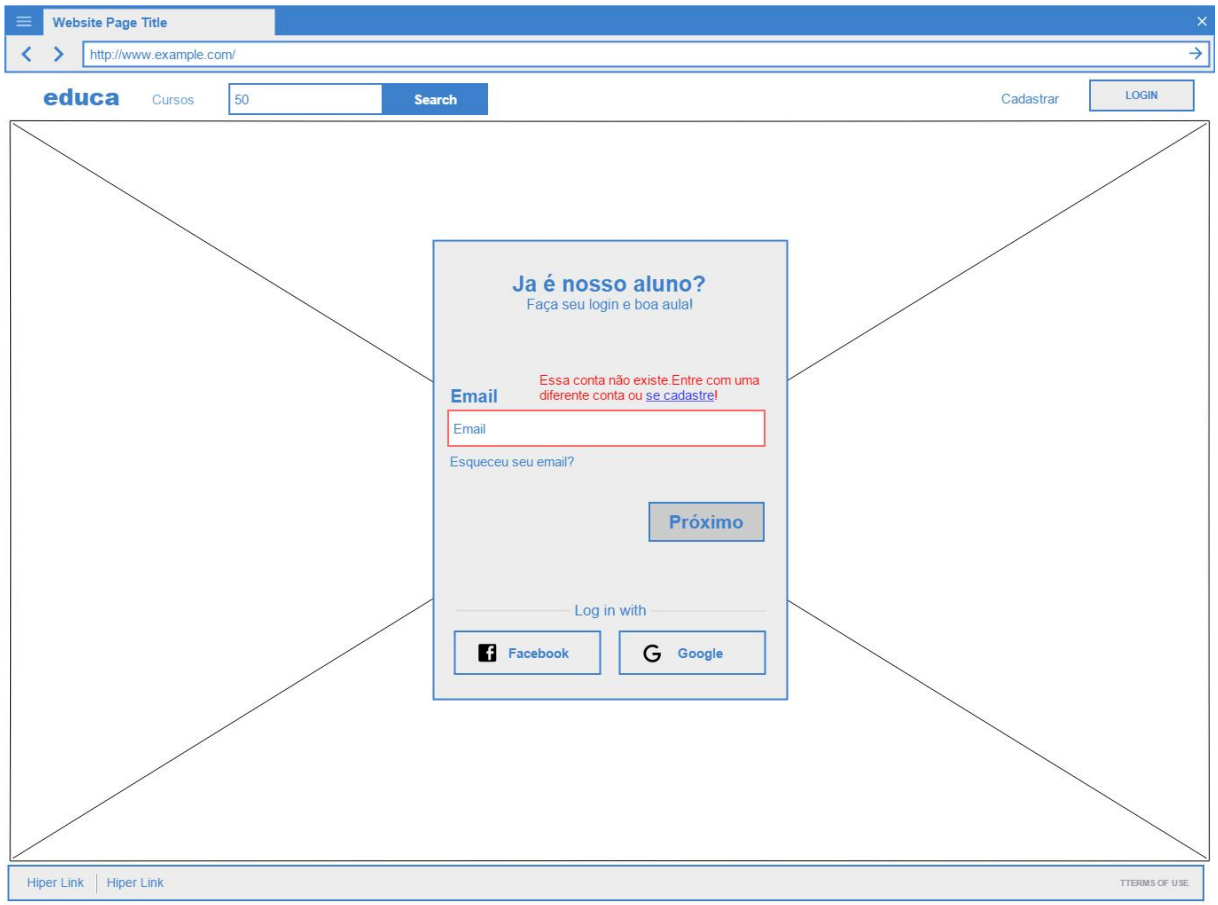
Fonte: Elaborado pelo autor.

Figura 195 – Wireframe da tela de login fase de inserir email



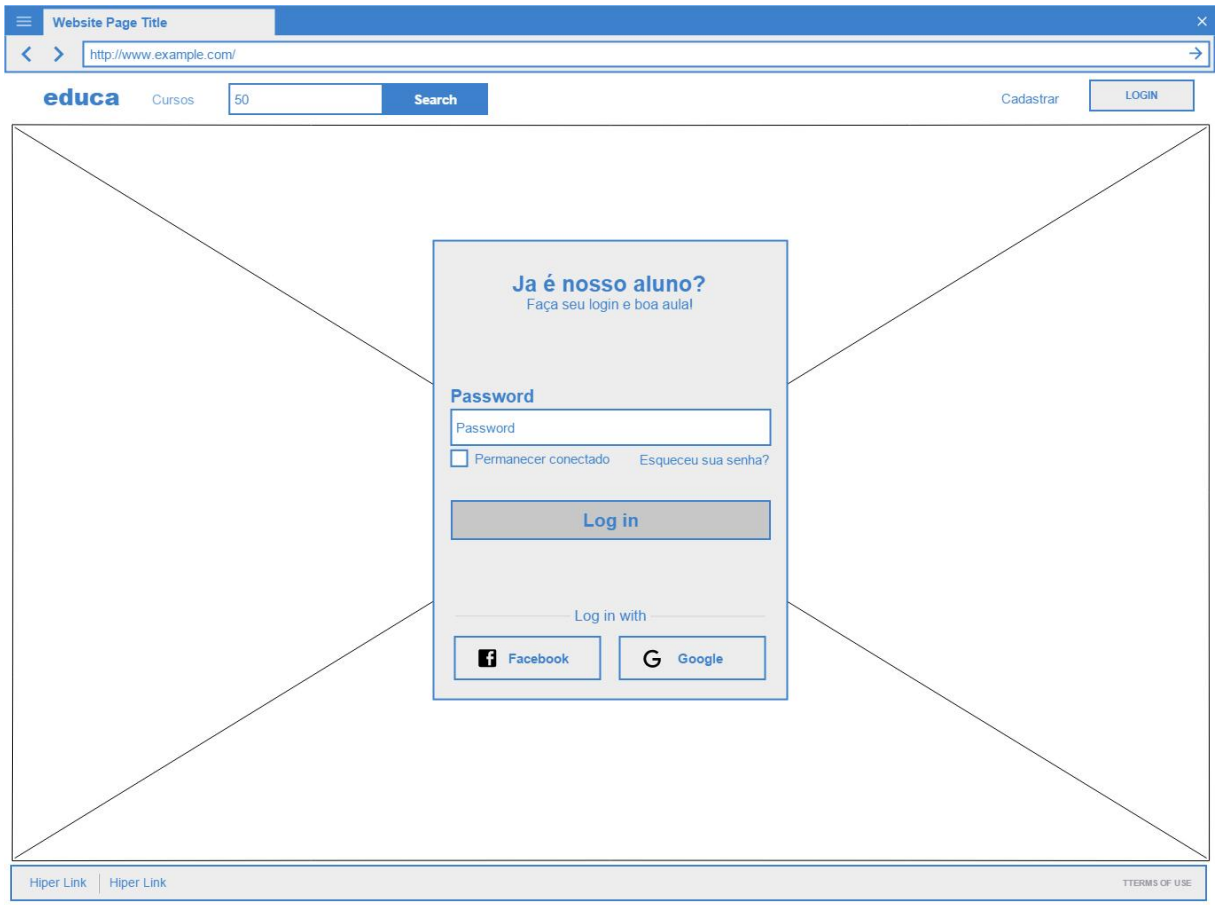
Fonte: Elaborado pelo autor.

Figura 196 – Wireframe da tela de login em situação de email não existente



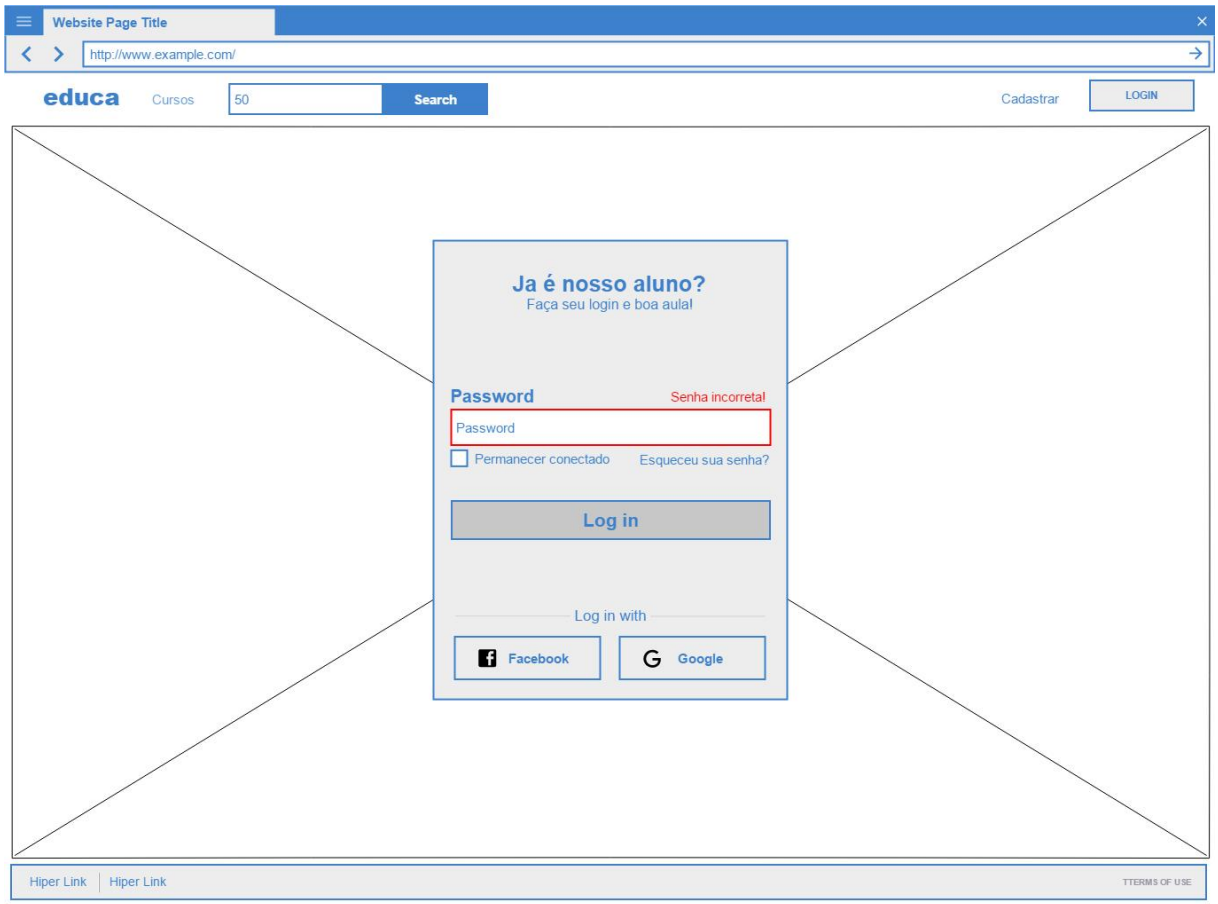
Fonte: Elaborado pelo autor.

Figura 197 – Wireframe da tela de login fase de inserir a senha



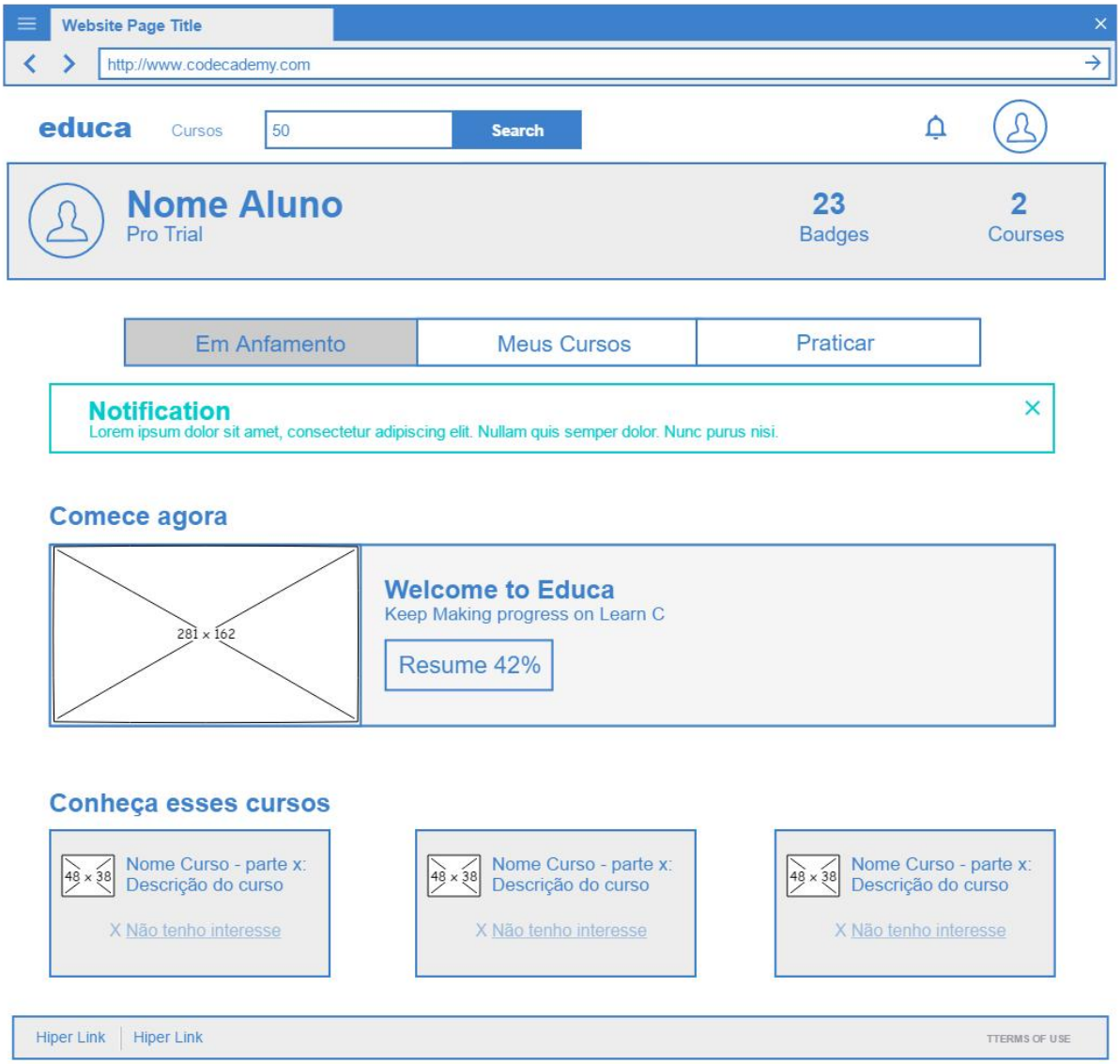
Fonte: Elaborado pelo autor.

Figura 198 – Wireframe da tela de login em situação de senha inválida



Fonte: Elaborado pelo autor.

Figura 199 – Wireframe da tela de dashboard em situação de começando o curso

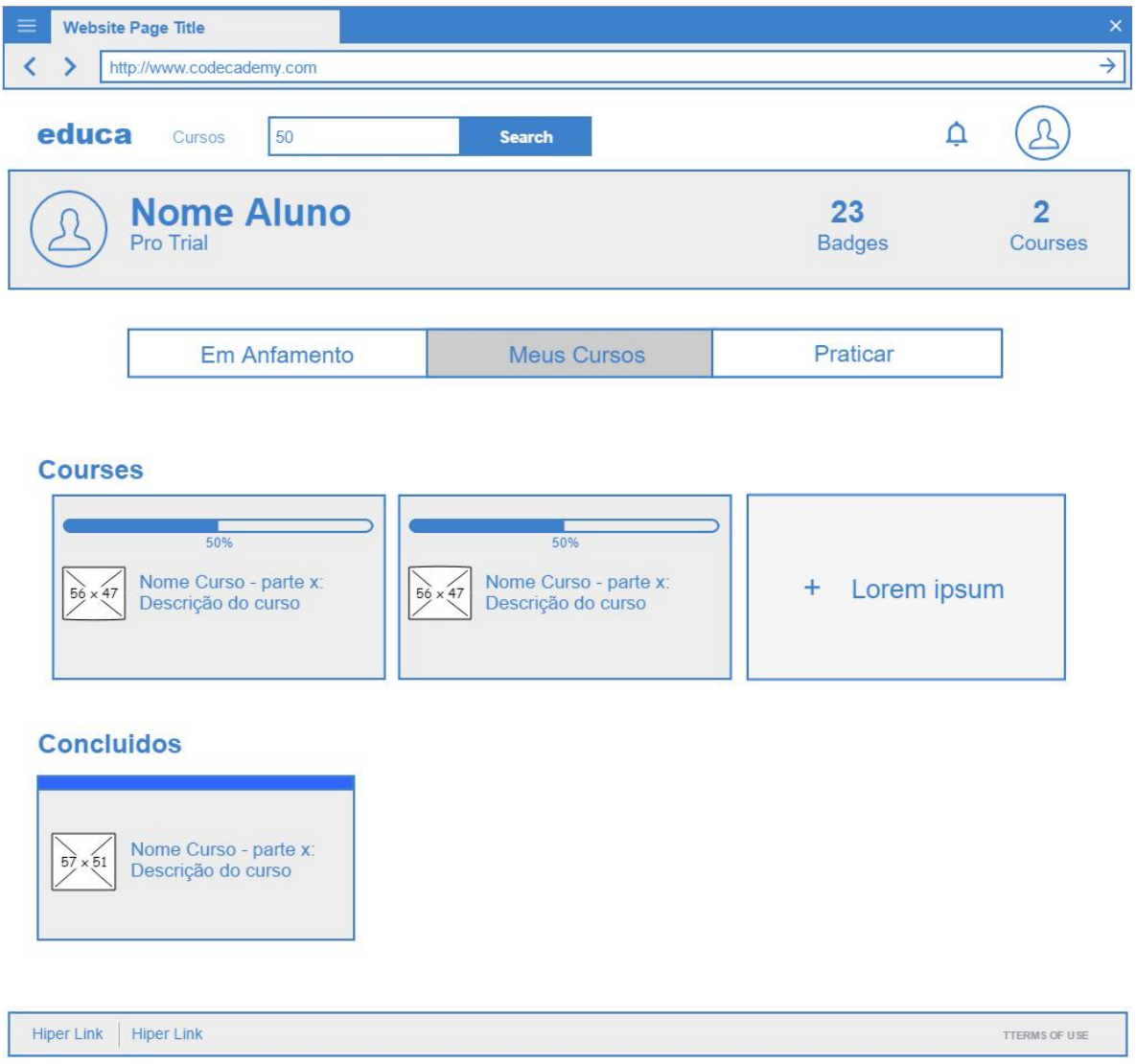


Fonte: Elaborado pelo autor.

Figura 200 – Wireframe da tela de dashboard em situação cursos em andamento

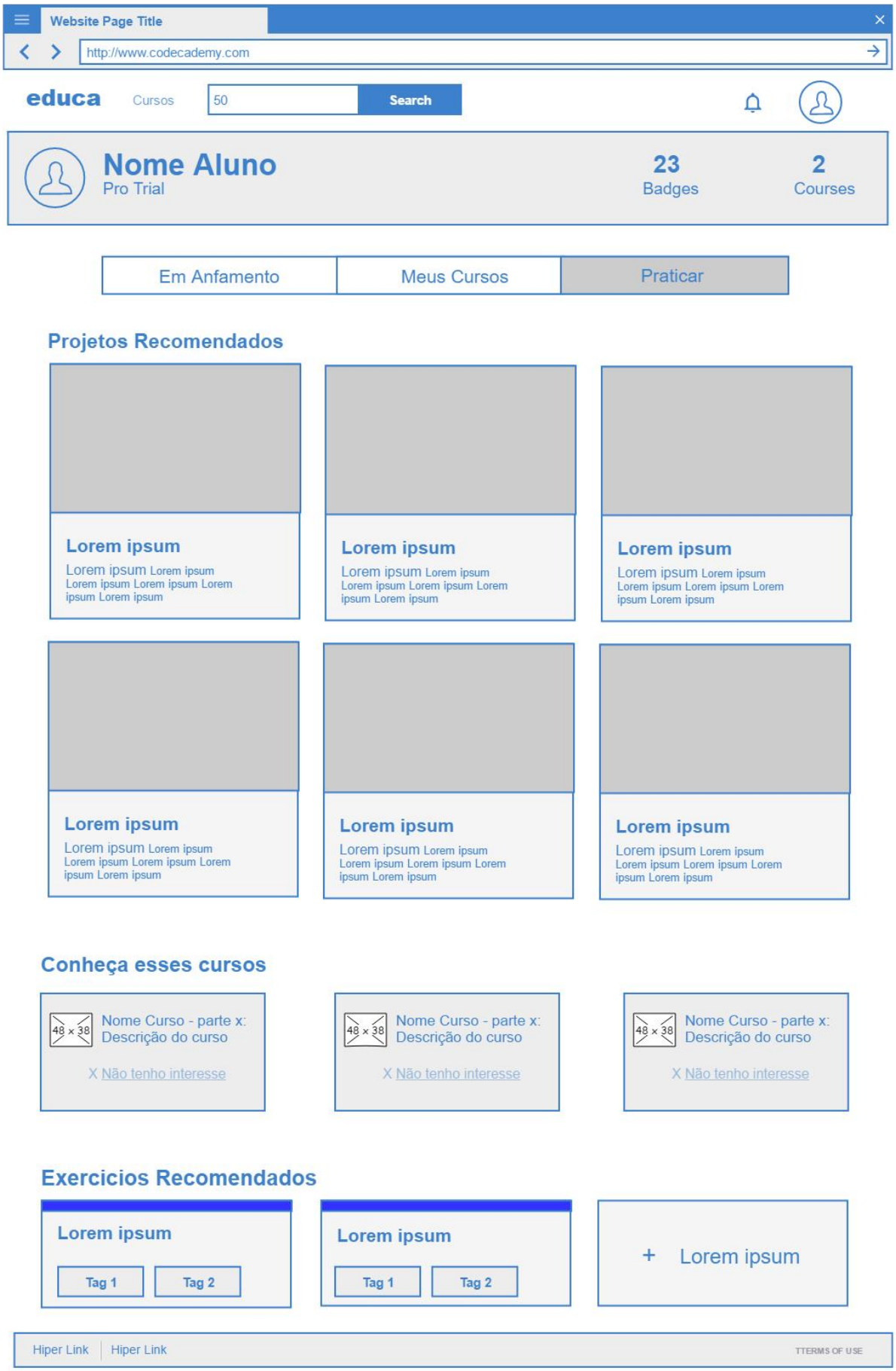
Fonte: Elaborado pelo autor.

Figura 201 – Wireframe da tela de meus cursos



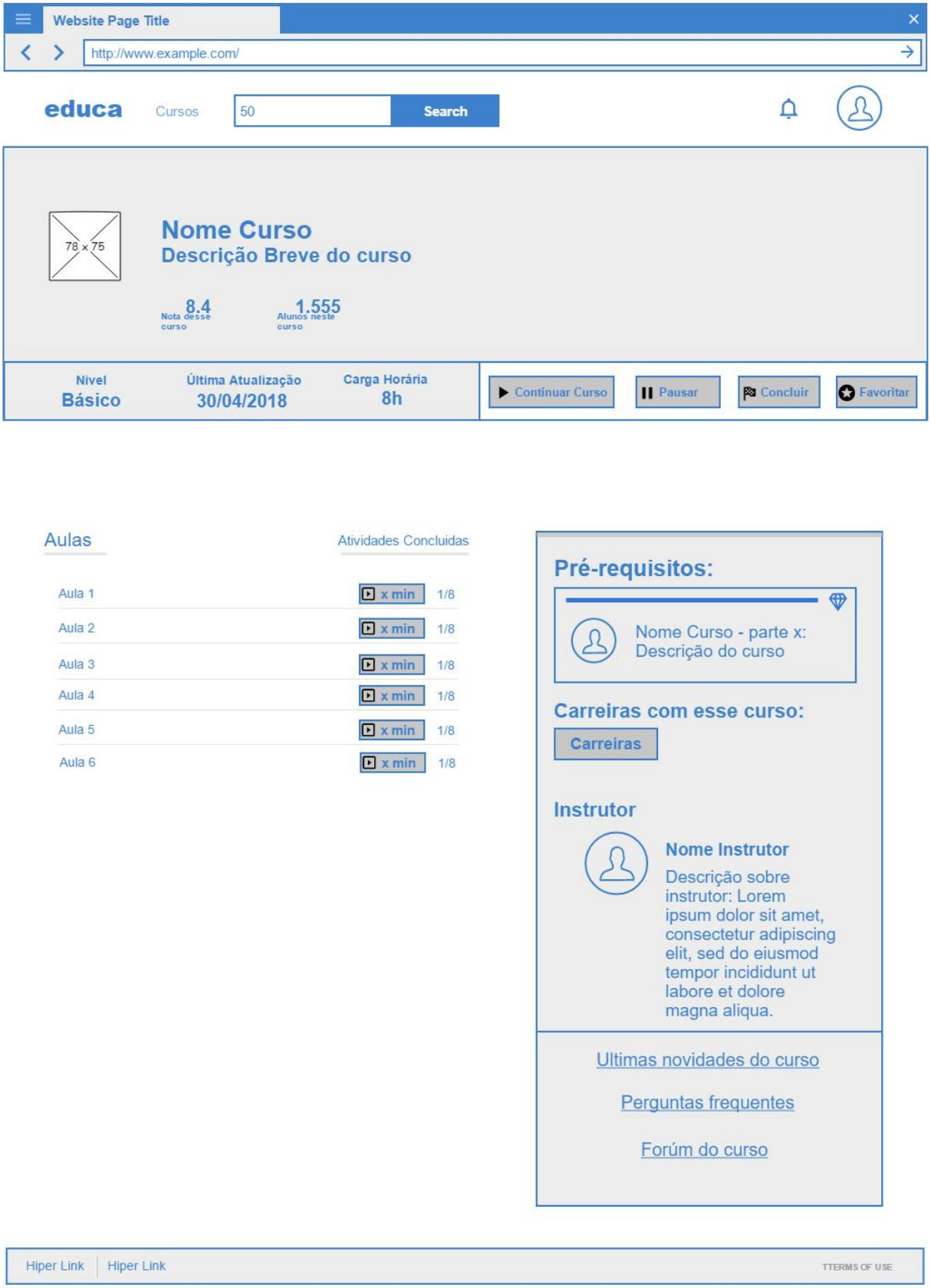
Fonte: Elaborado pelo autor.

Figura 202 – Wireframe da tela de dashboard em praticar



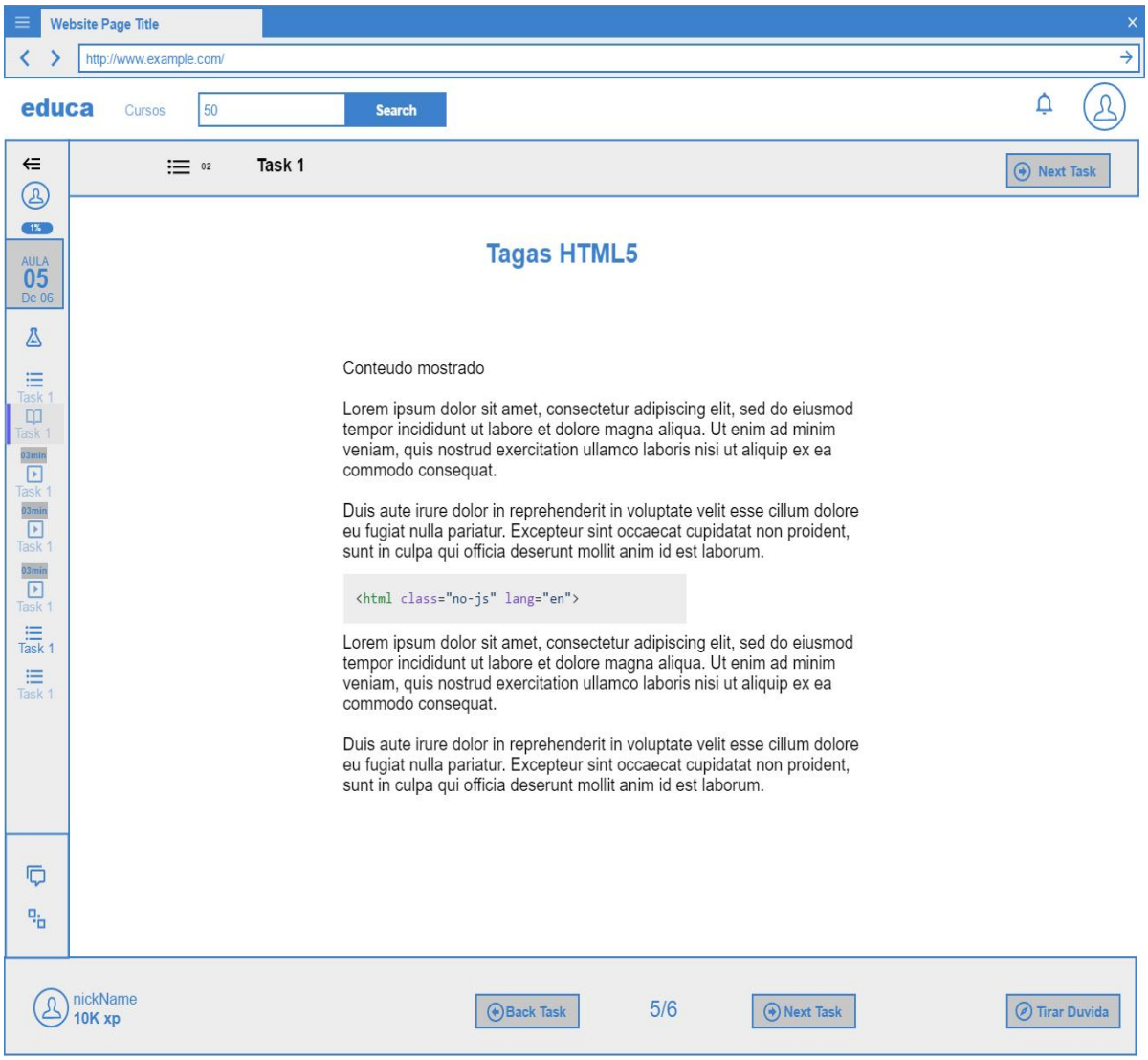
Fonte: Elaborado pelo autor.

Figura 203 – Wireframe da tela do curso



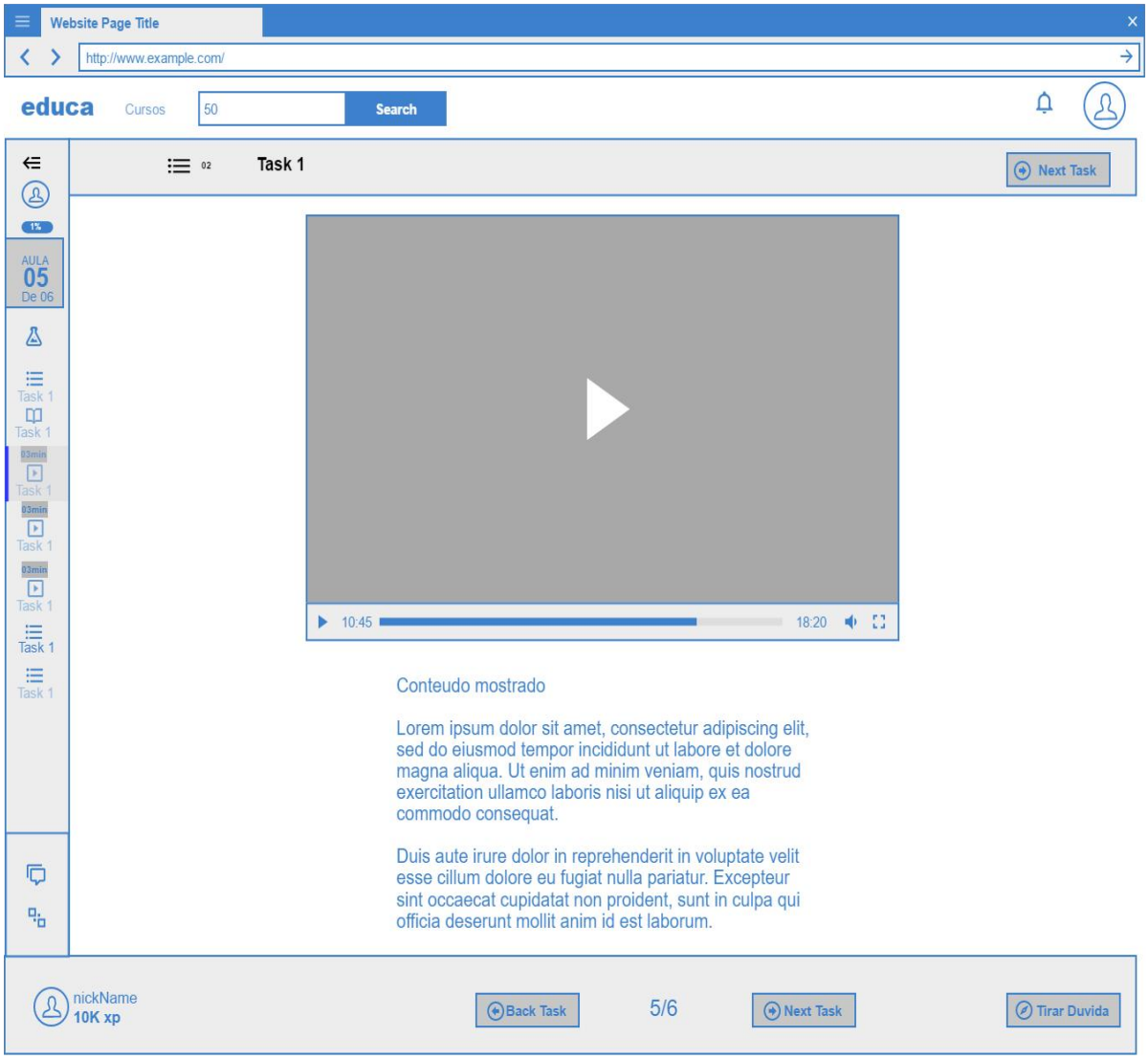
Fonte: Elaborado pelo autor.

Figura 204 – Wireframe da tela de atividade tipo de leitura



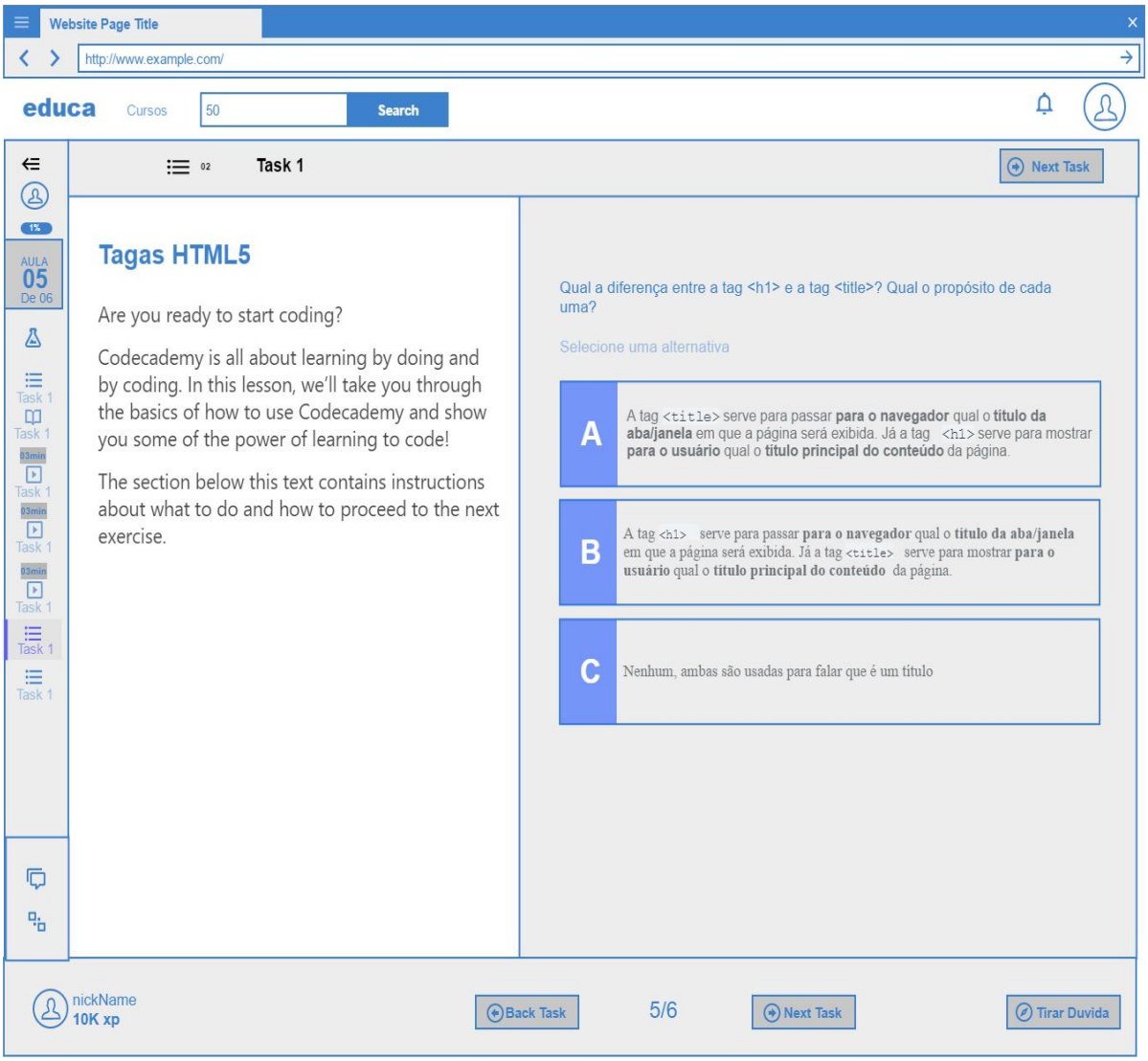
Fonte: Elaborado pelo autor.

Figura 205 – Wireframe da tela de atividade tipo de video



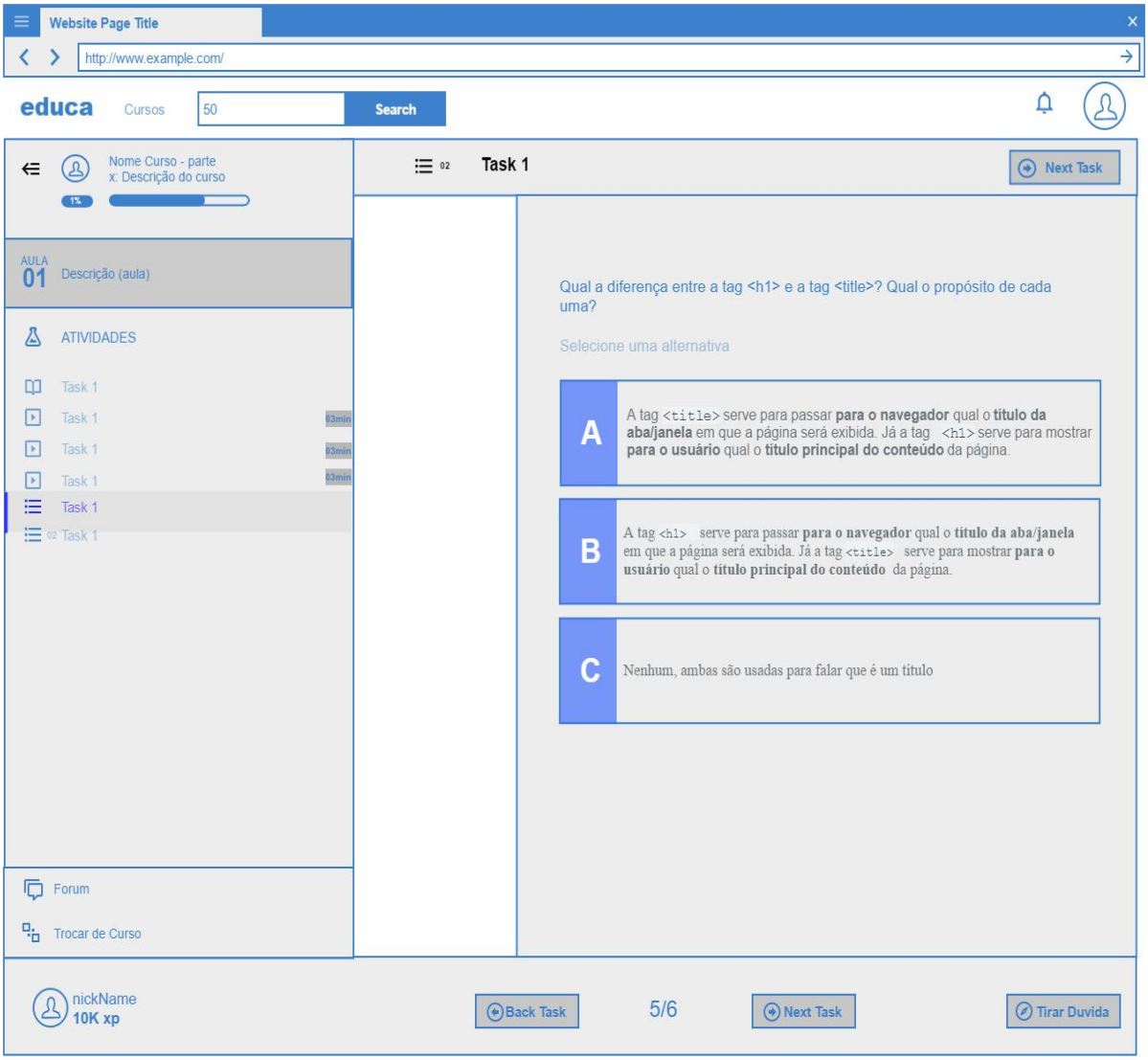
Fonte: Elaborado pelo autor.

Figura 206 – Wireframe da tela de atividade tipo de questionário menu desabilitado



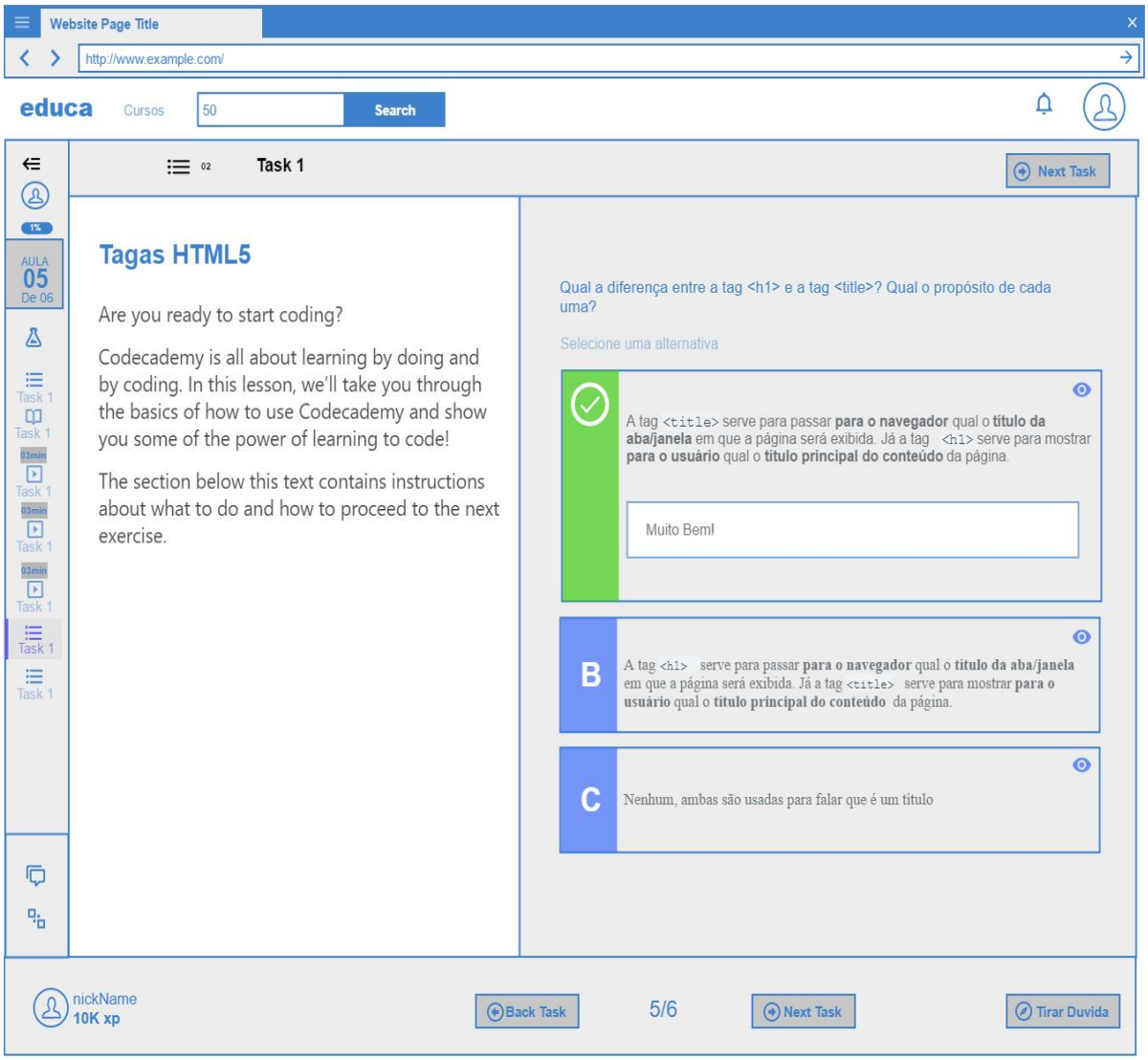
Fonte: Elaborado pelo autor.

Figura 207 – Wireframe da tela de atividade tipo de questionário menu habilitado



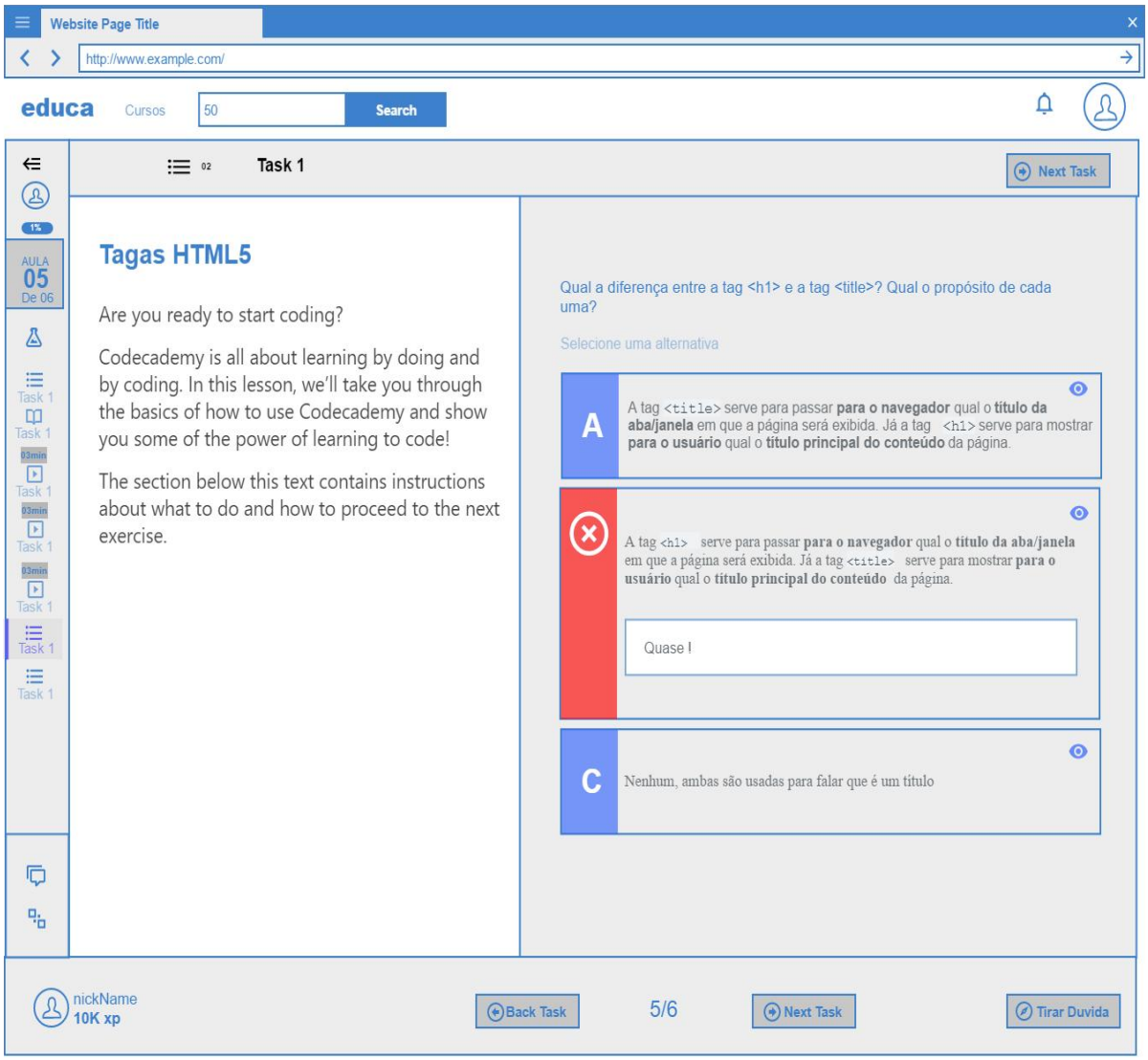
Fonte: Elaborado pelo autor.

Figura 208 – Wireframe da tela de atividade tipo de questionário respondido corretamente



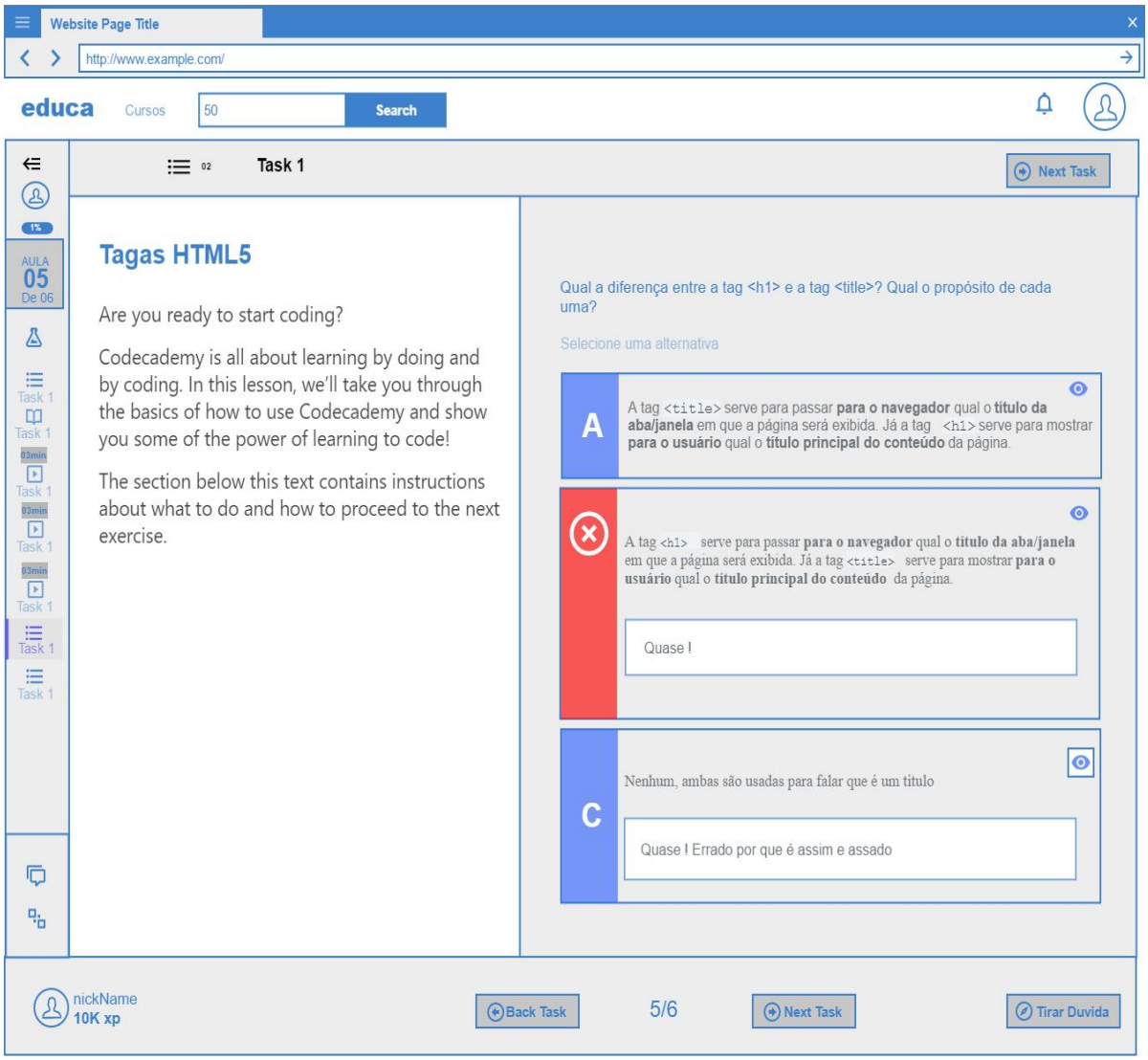
Fonte: Elaborado pelo autor.

Figura 209 – Wireframe da tela de atividade tipo de questionário respondido erroneamente



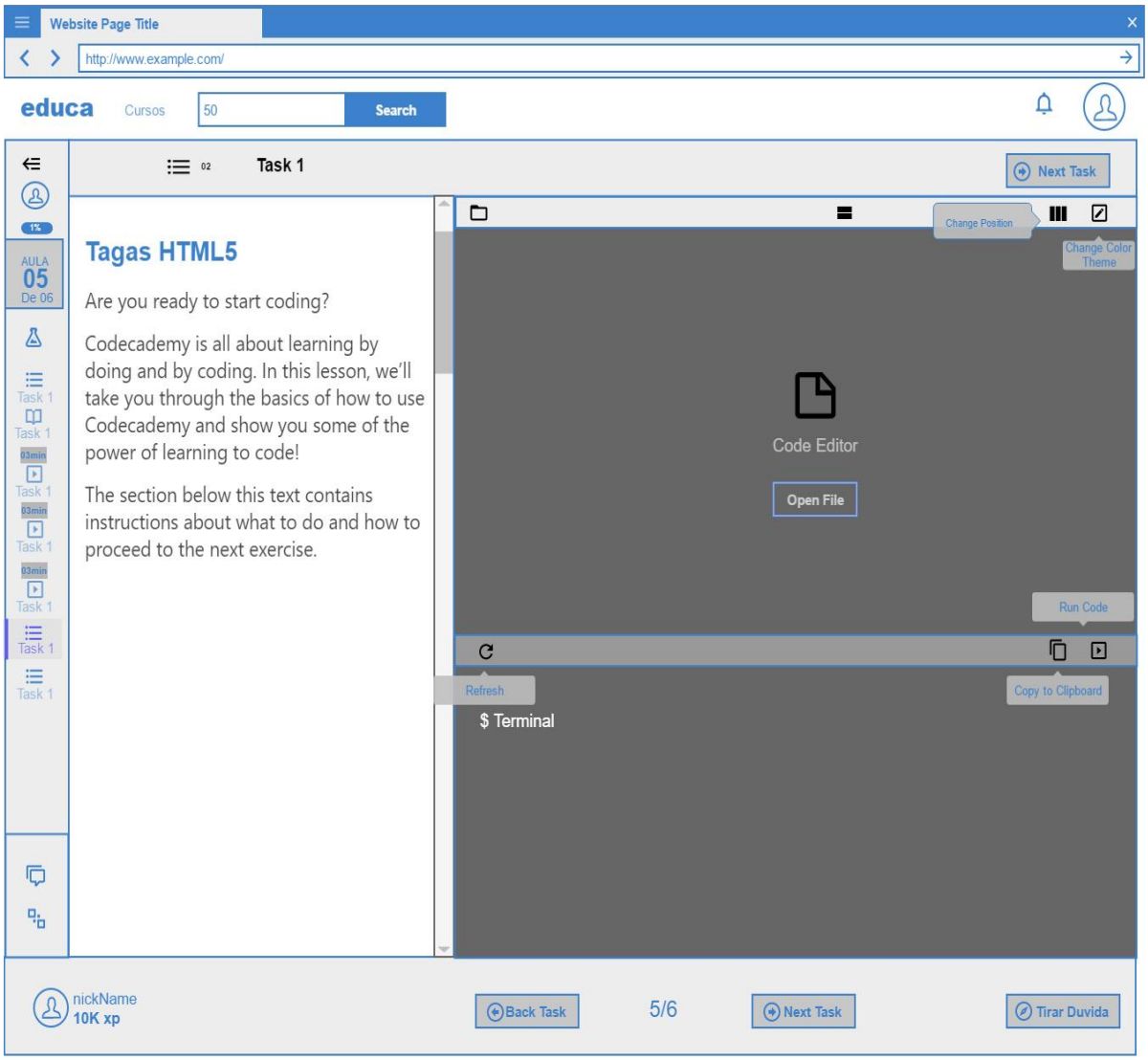
Fonte: Elaborado pelo autor.

Figura 210 – Wireframe da tela de atividade tipo de questionário resposta verificado



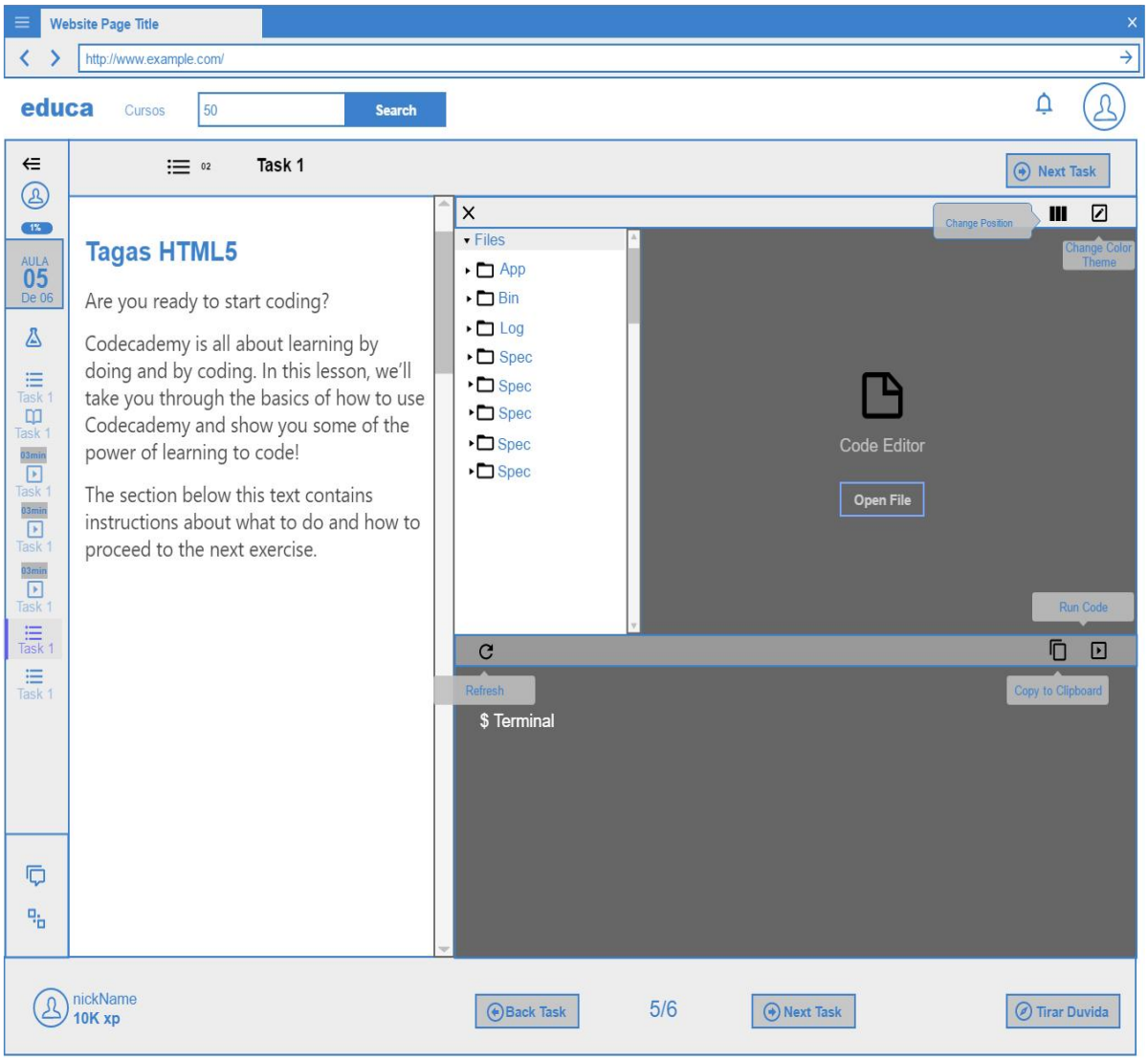
Fonte: Elaborado pelo autor.

Figura 211 – Wireframe da tela de atividade tipo de codificação visualização horizontal



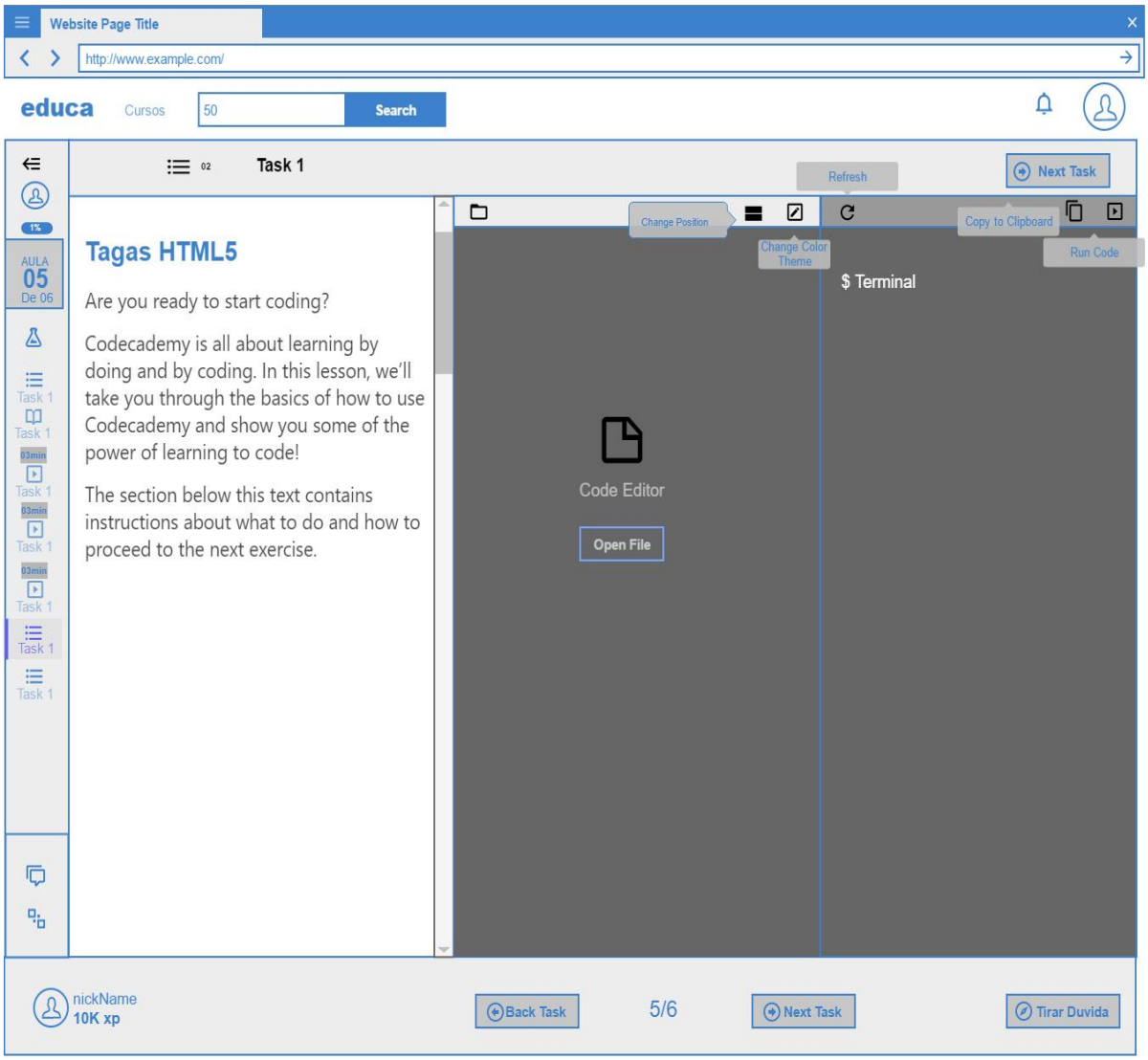
Fonte: Elaborado pelo autor.

Figura 212 – Wireframe da tela de atividade tipo de codificação visualização de pastas



Fonte: Elaborado pelo autor.

Figura 213 – Wireframe da tela de atividade tipo de codificação visualização vertical



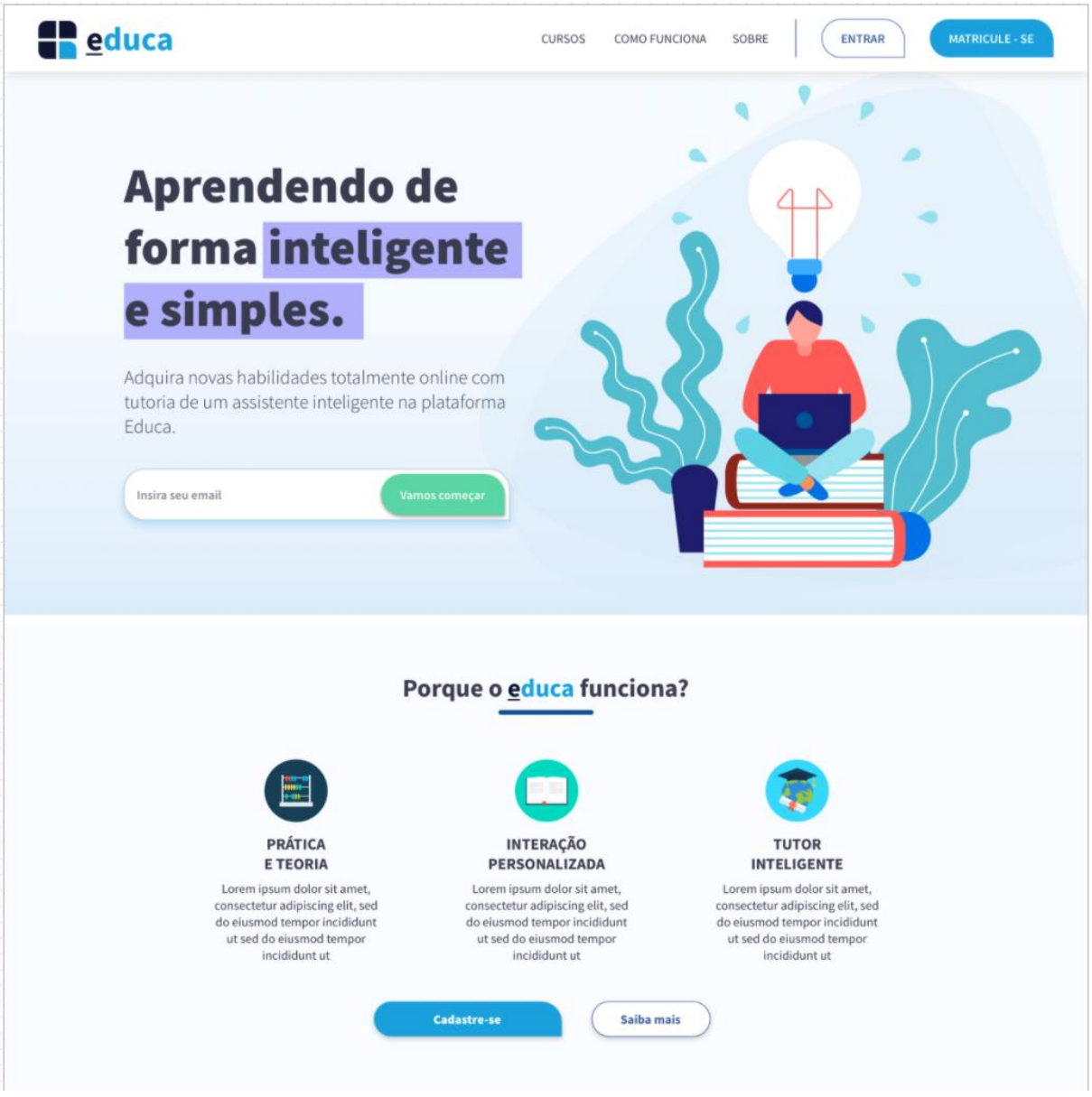
Fonte: Elaborado pelo autor.

D.4 Mockups

Figura 214 – Mockup da tela de homepage

Fonte: Elaborado pelo autor.

Figura 215 – Mockup da tela de homepage - zoom 1

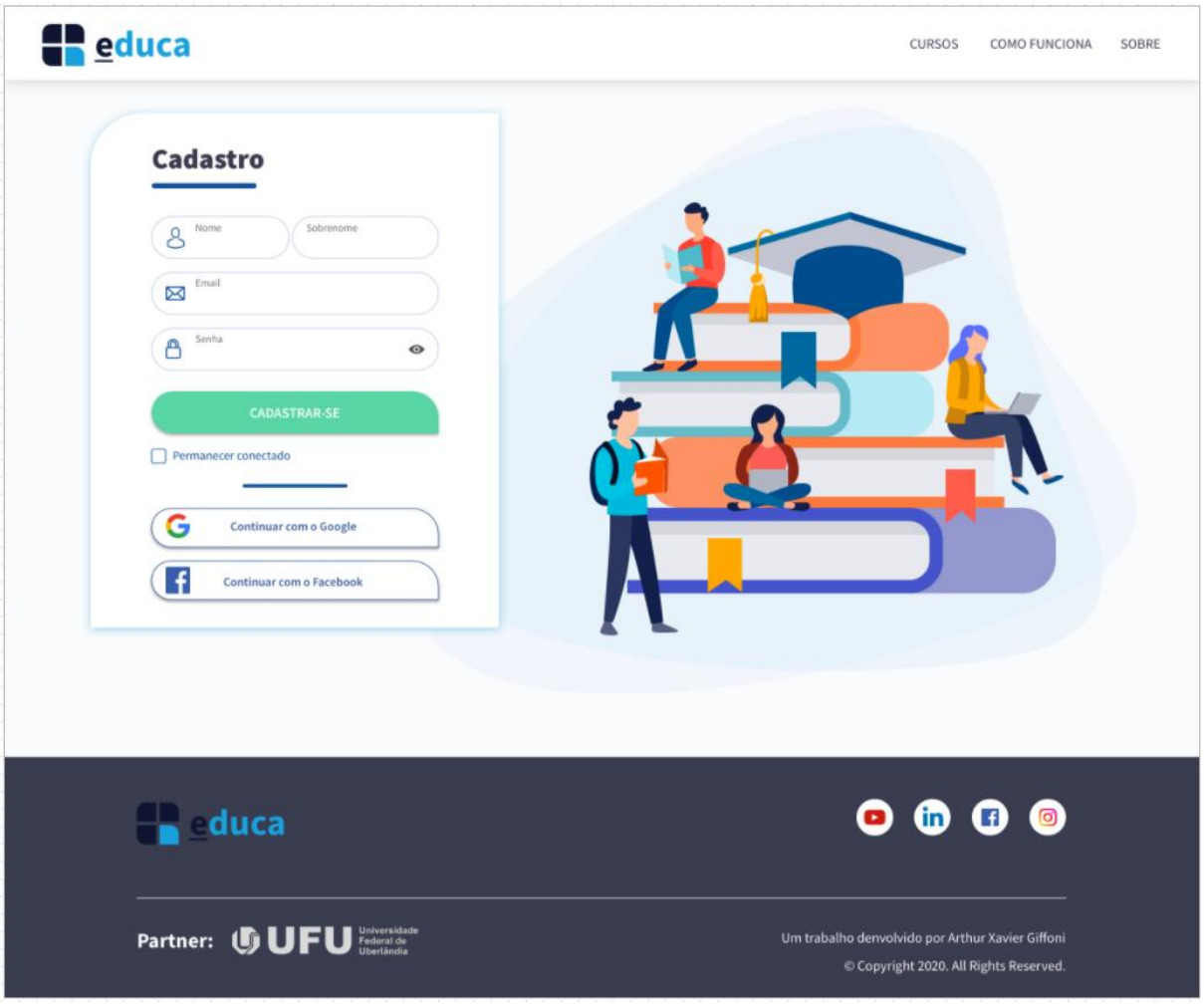


Fonte: Elaborado pelo autor.

Figura 216 – Mockup da tela de homepage - zoom 2

Fonte: Elaborado pelo autor.

Figura 217 – Mockup da tela de cadastro



Fonte: Elaborado pelo autor.

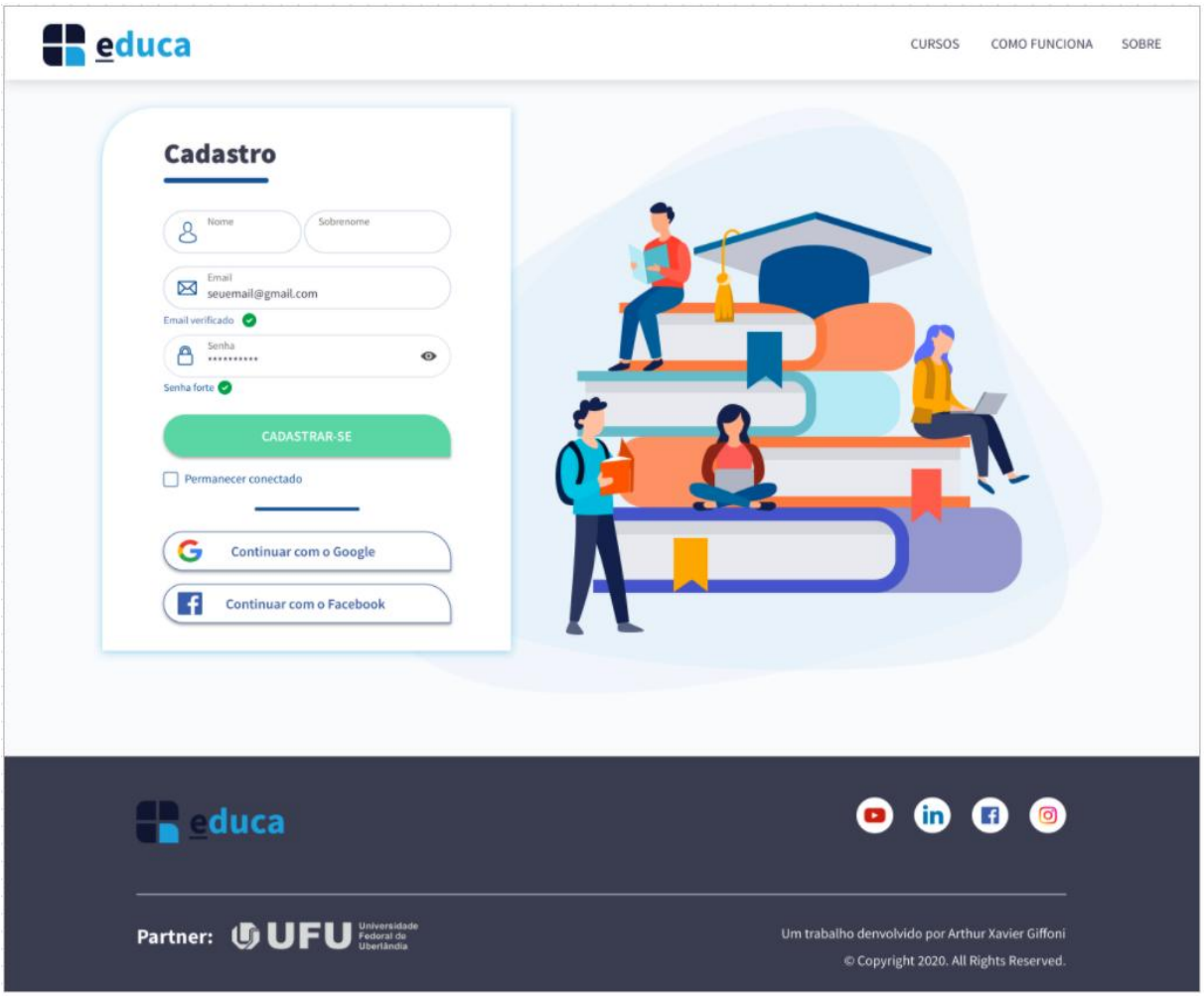
Figura 218 – Mockup da tela de cadastro - 2

Fonte: Elaborado pelo autor.

Figura 219 – Mockup da tela de cadastro - 3

Fonte: Elaborado pelo autor.

Figura 220 – Mockup da tela de cadastro - 4



Fonte: Elaborado pelo autor.

Figura 221 – Mockup da tela de login - 1

Fonte: Elaborado pelo autor.

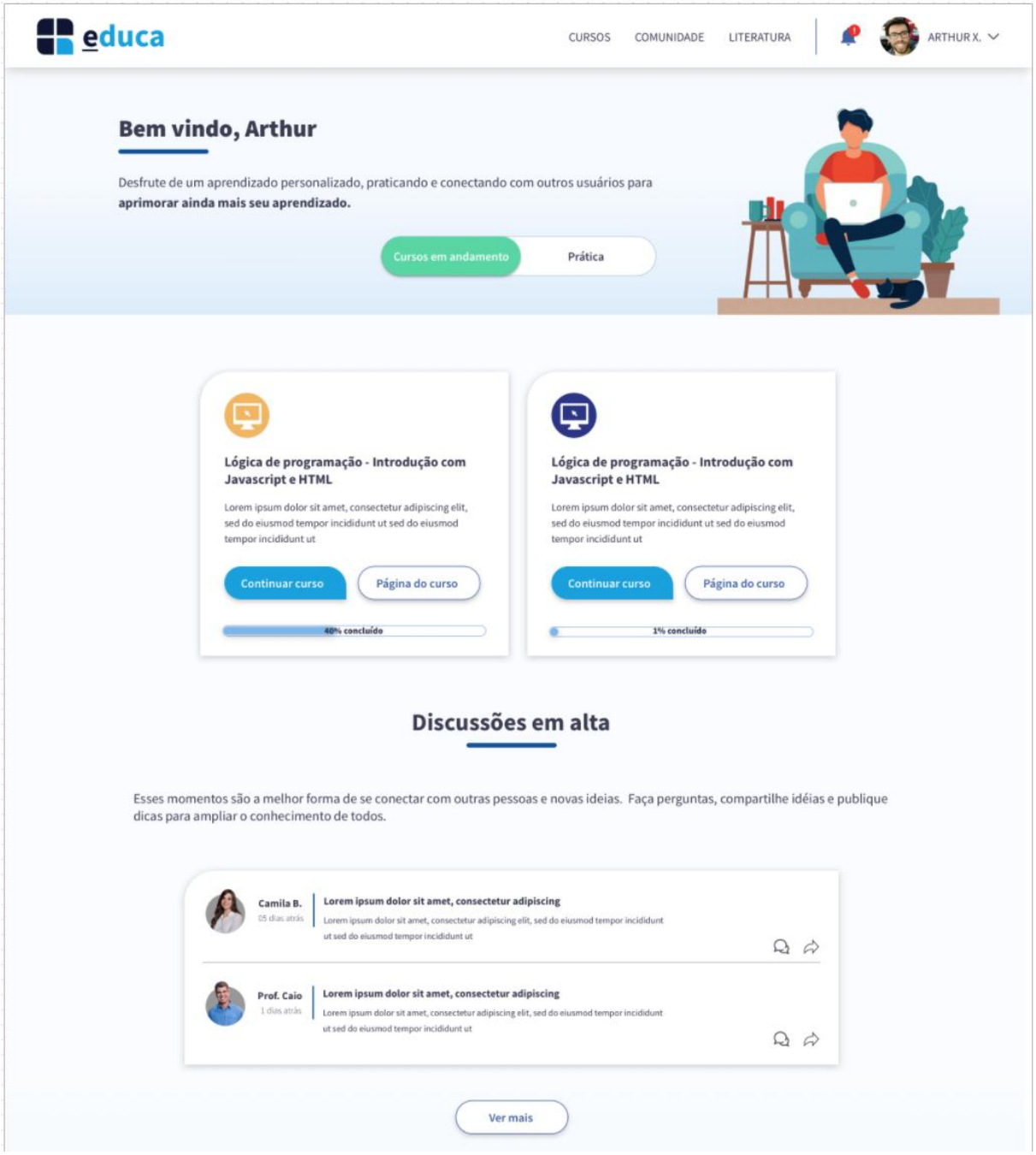
Figura 222 – Mockup da tela de login - 2

Fonte: Elaborado pelo autor.

Figura 223 – Mockup da tela de login - 3

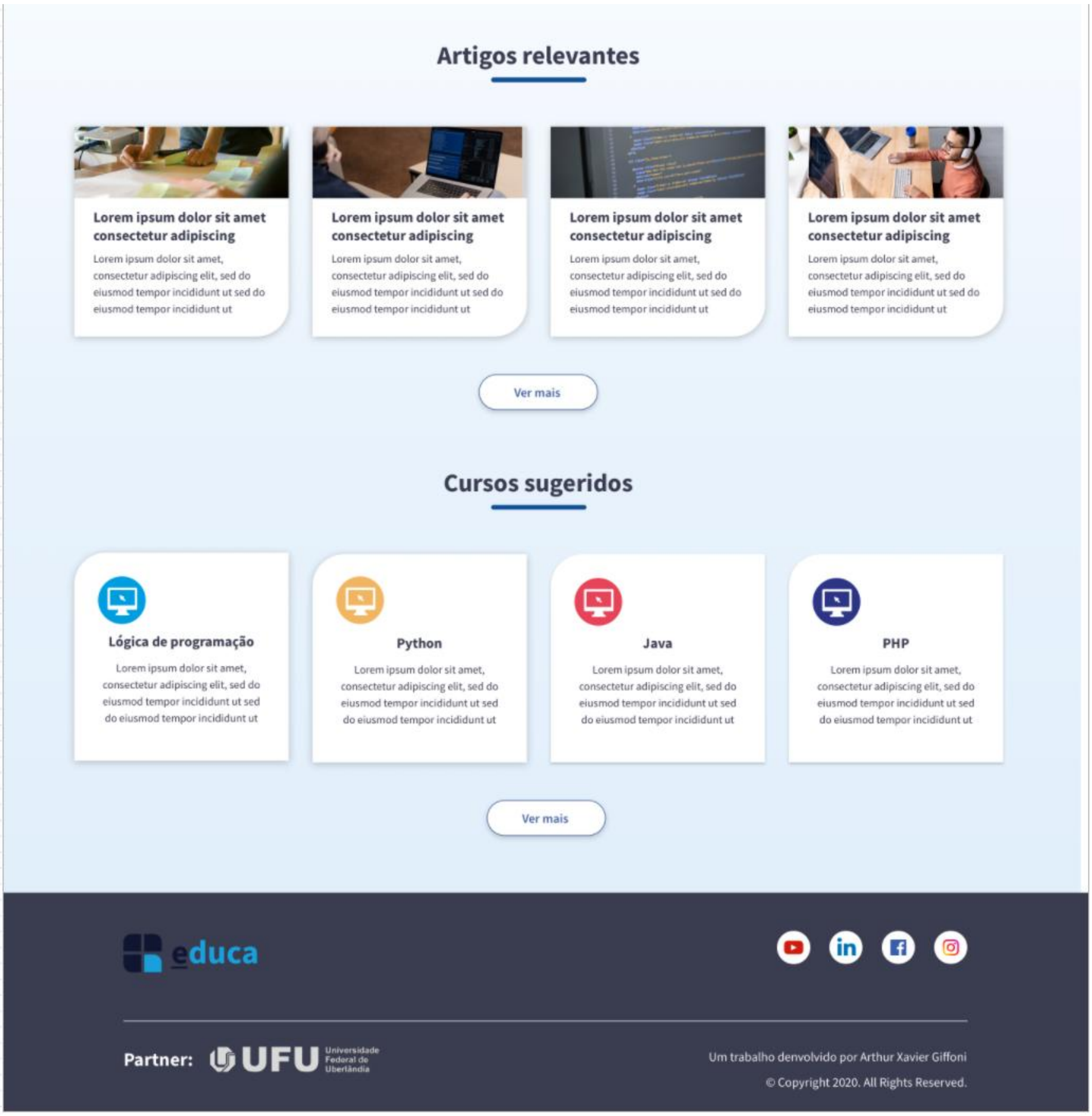
Fonte: Elaborado pelo autor.

Figura 224 – Mockup da tela de dashboard 1



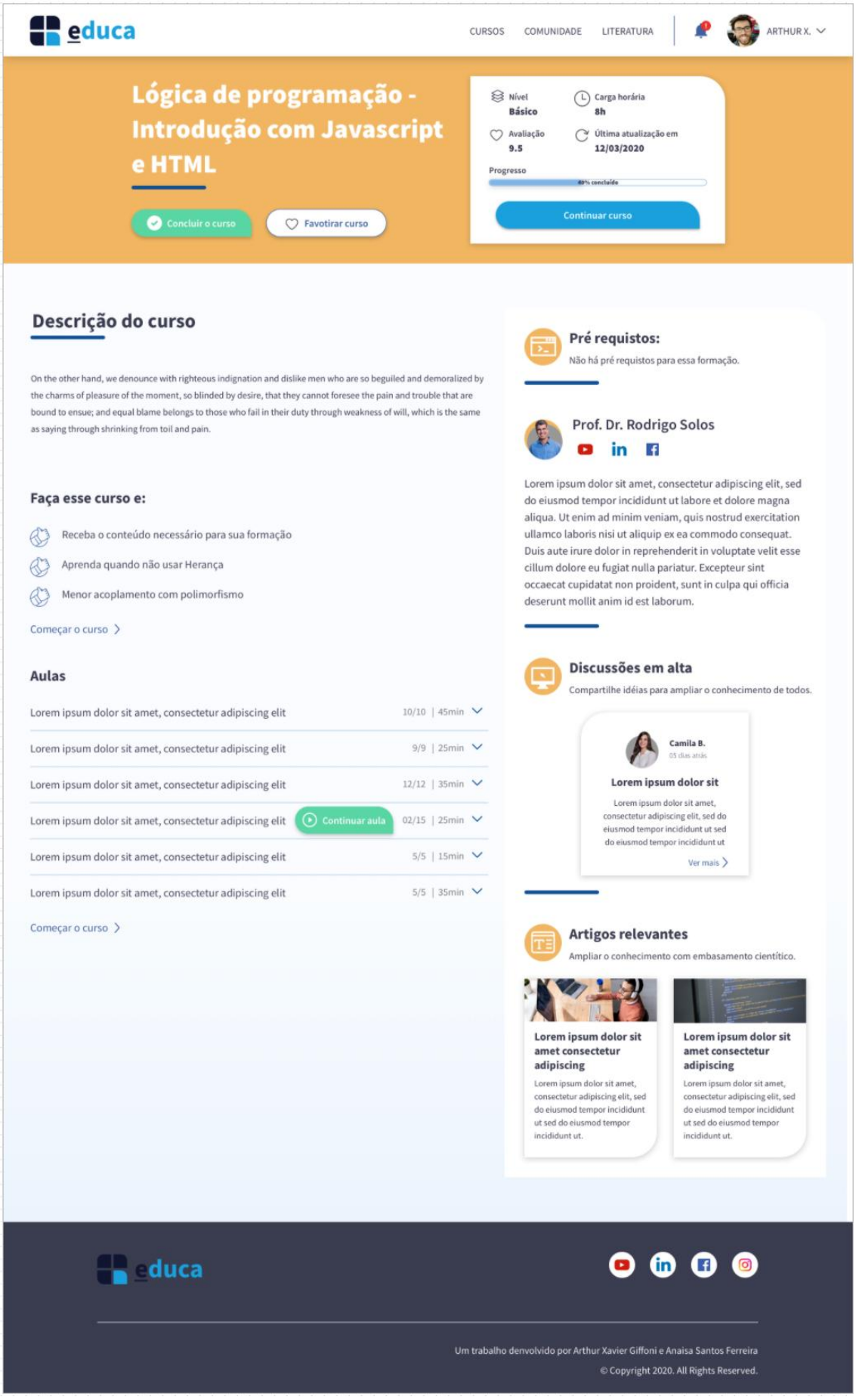
Fonte: Elaborado pelo autor.

Figura 225 – Mockup da tela de dashboard 2



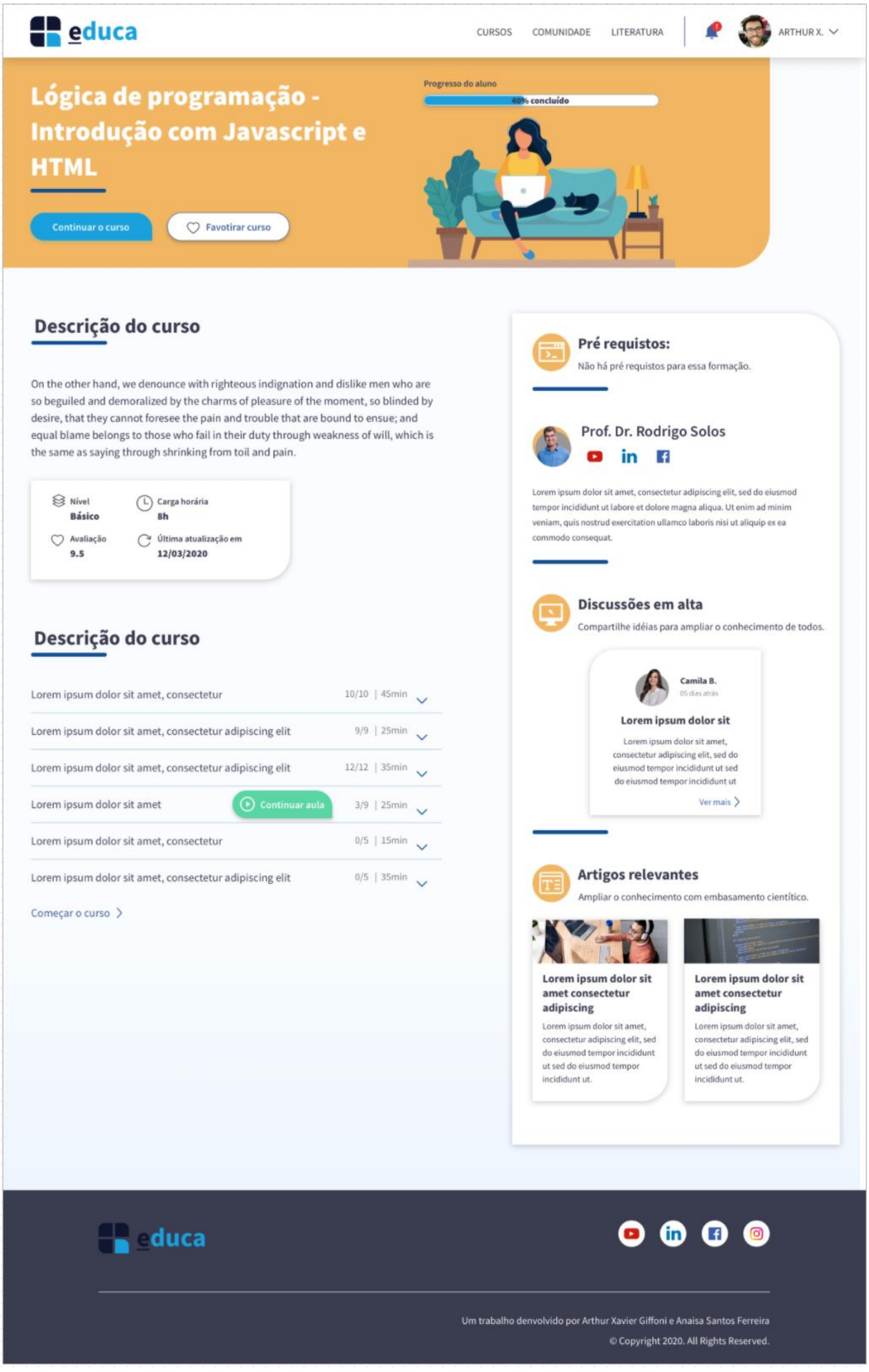
Fonte: Elaborado pelo autor.

Figura 226 – Mockup da tela de curso



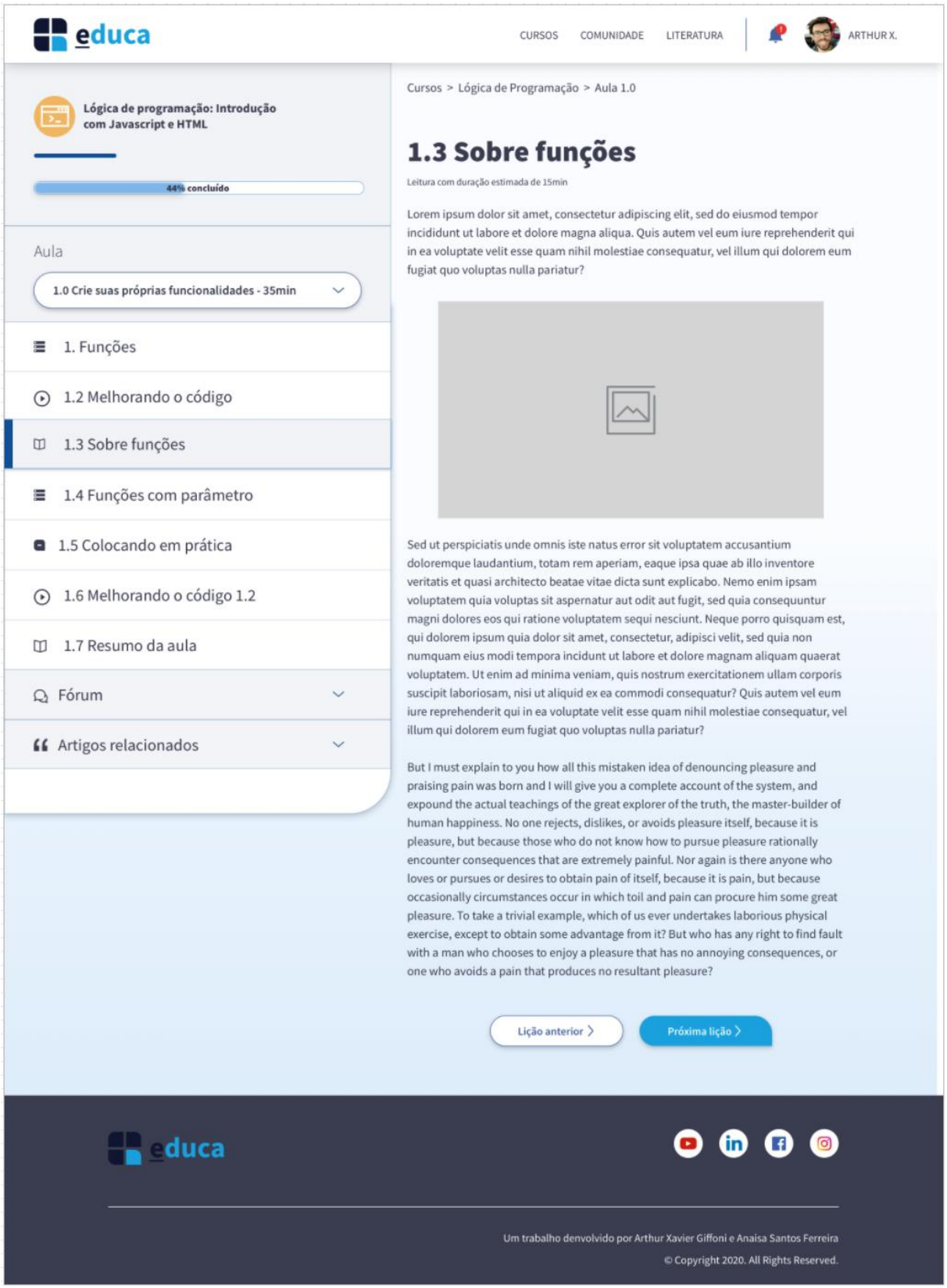
Fonte: Elaborado pelo autor.

Figura 227 – Mockup da tela de curso versão 2



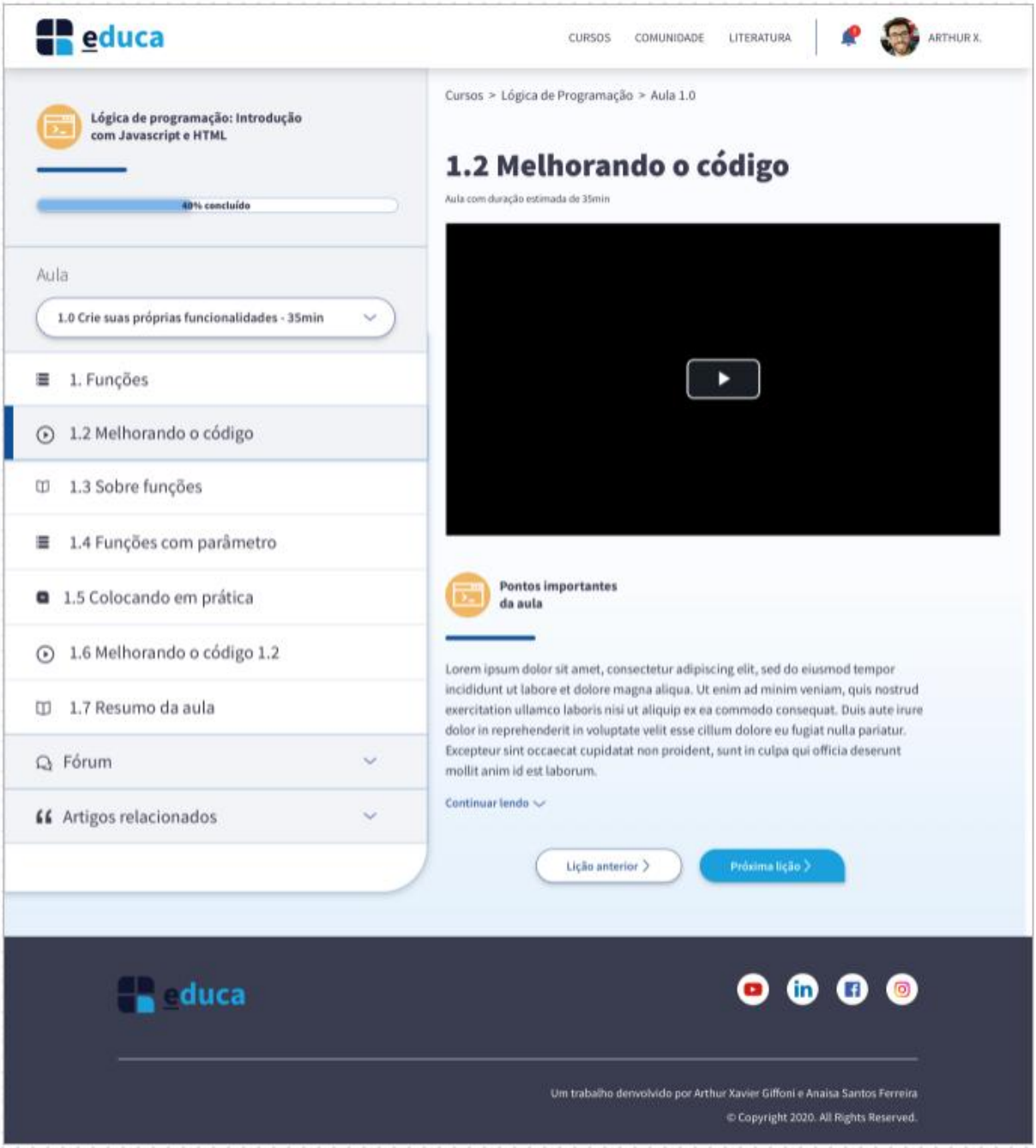
Fonte: Elaborado pelo autor.

Figura 228 – Mockup da tela de atividade modalidade leitura



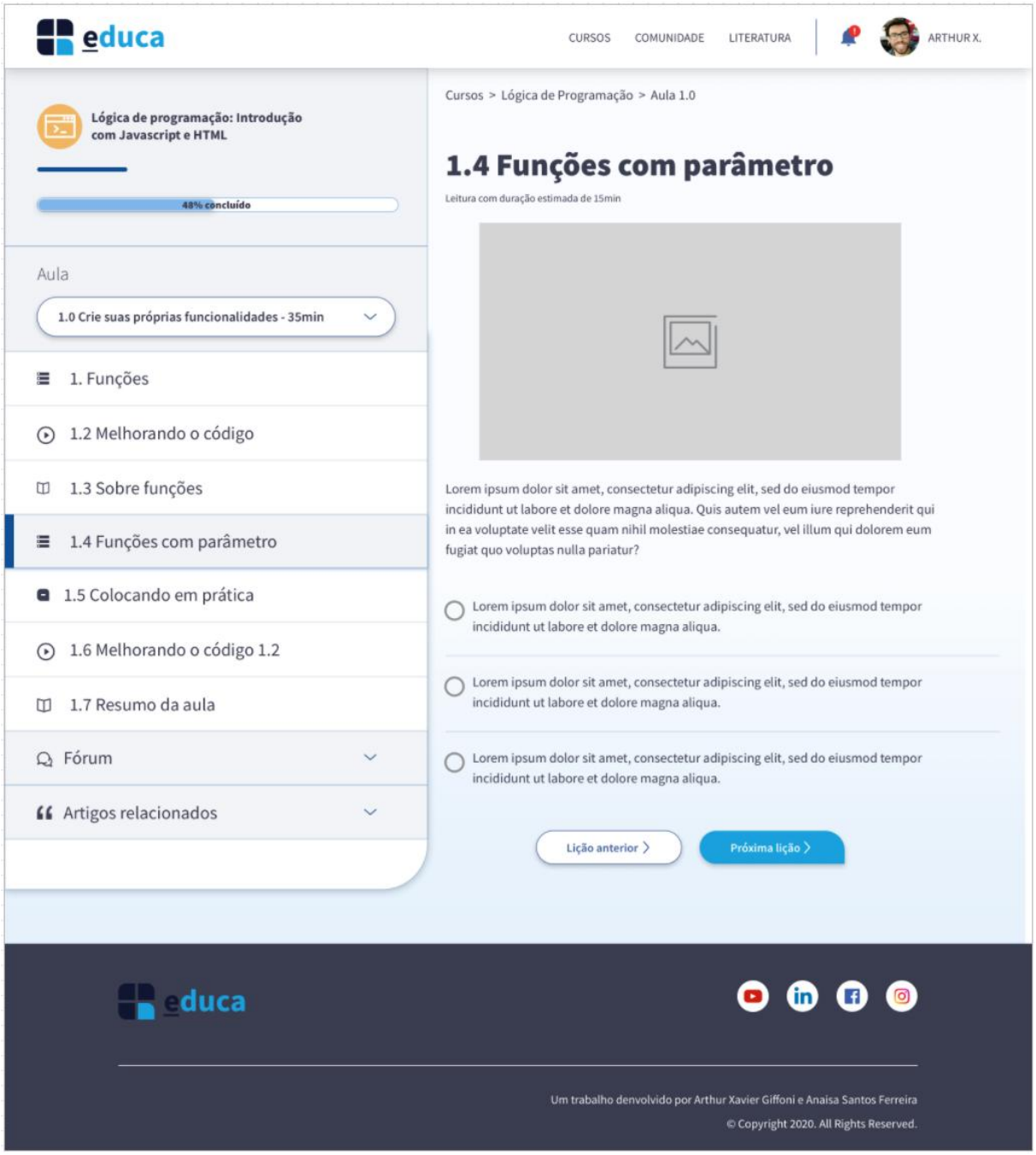
Fonte: Elaborado pelo autor.

Figura 229 – Mockup da tela de atividade modalidade video



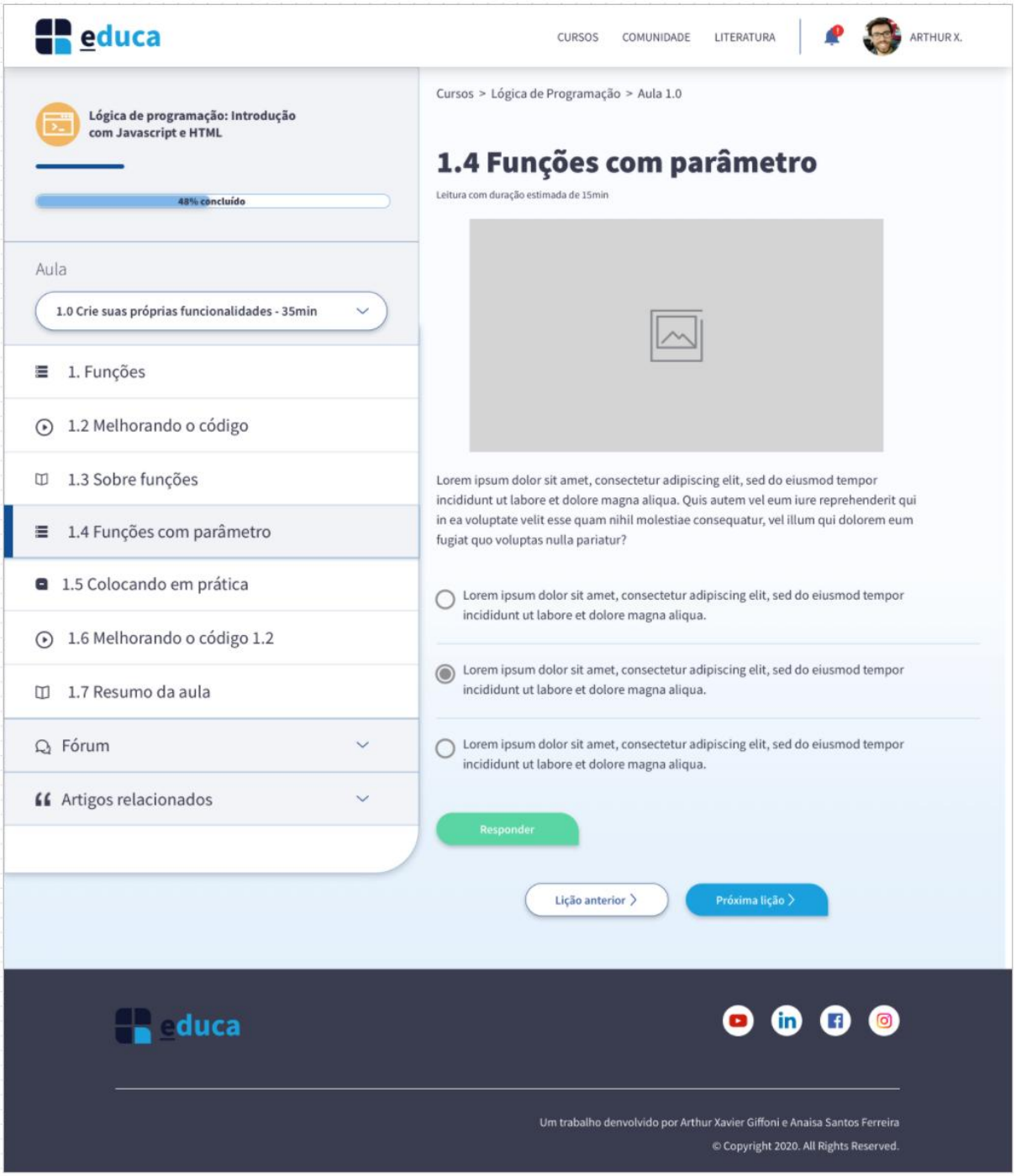
Fonte: Elaborado pelo autor.

Figura 230 – Mockup da tela de atividade modalidade questionário



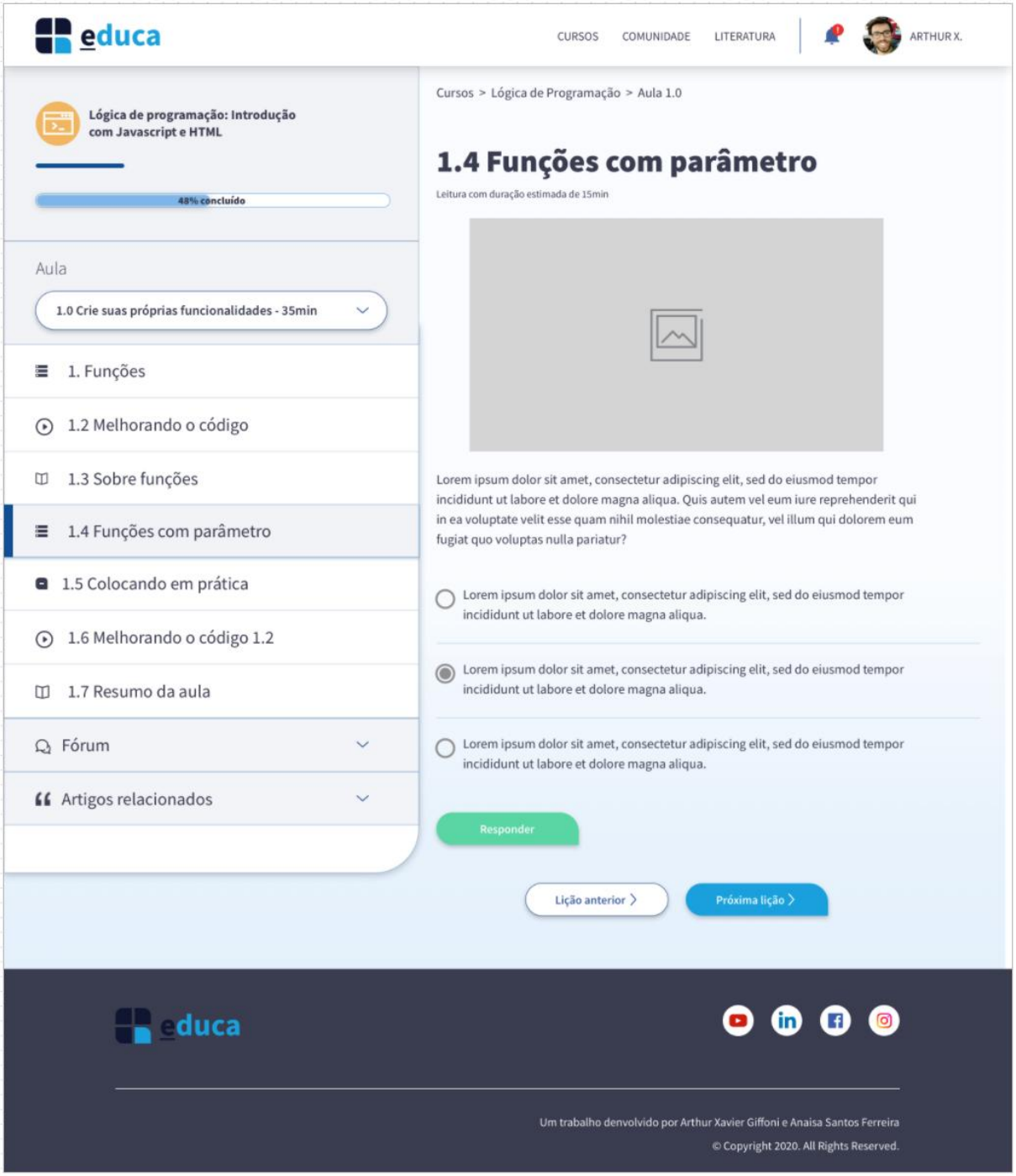
Fonte: Elaborado pelo autor.

Figura 231 – Mockup da tela de atividade modalidade questionário com opção de resposta selecionada



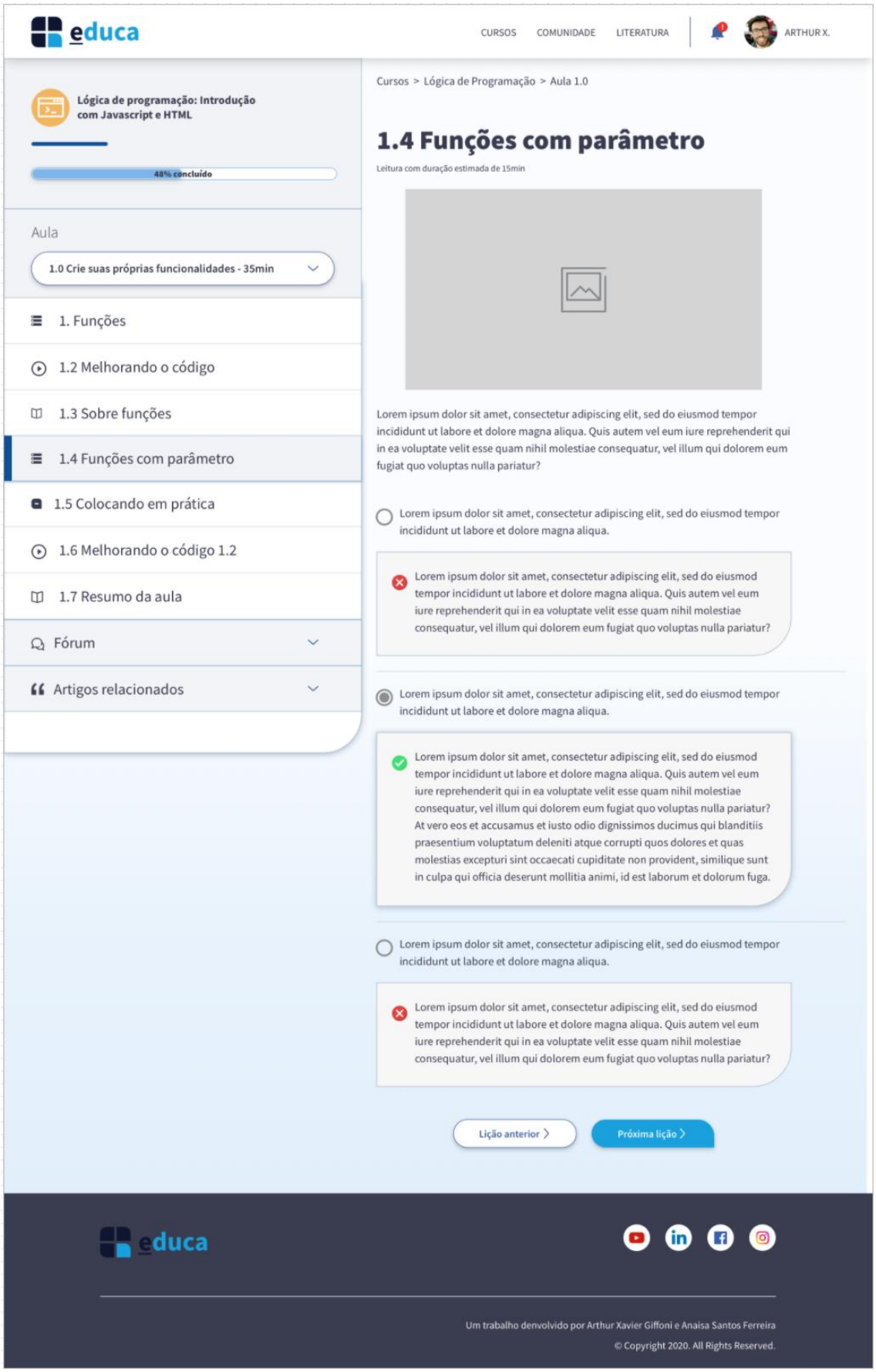
Fonte: Elaborado pelo autor.

Figura 232 – Mockup da tela de atividade modalidade questionário respondido



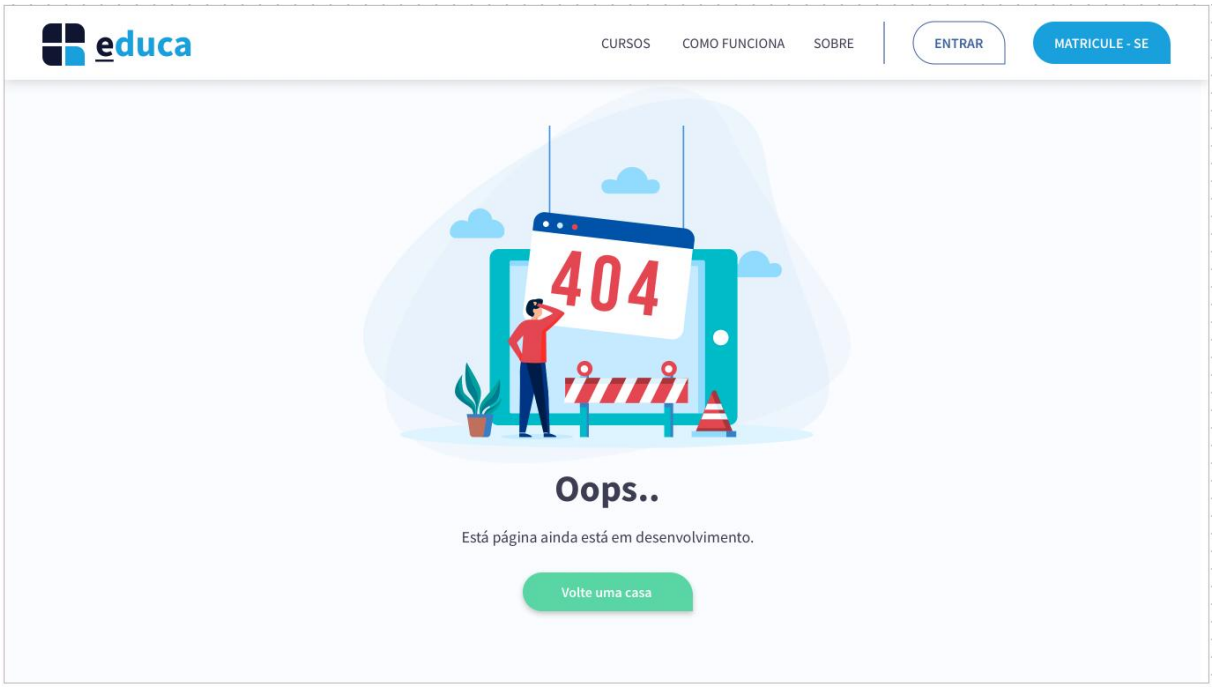
Fonte: Elaborado pelo autor.

Figura 233 – Mockup da tela de atividade modalidade questionário respondido



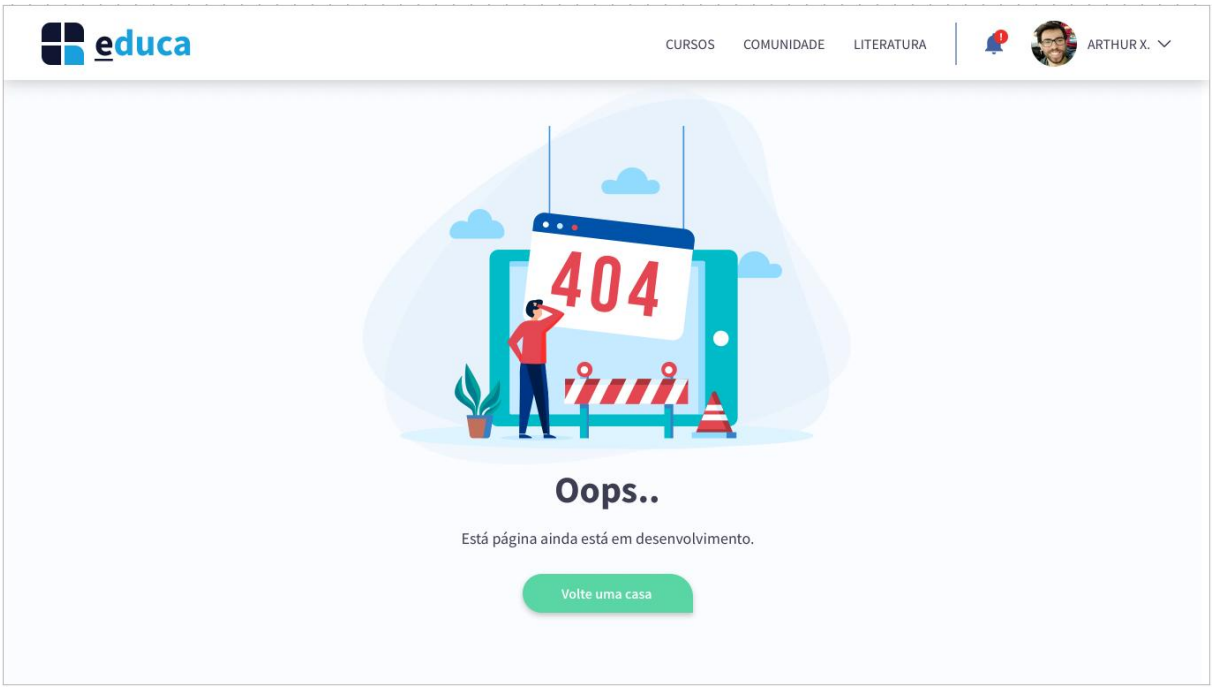
Fonte: Elaborado pelo autor.

Figura 234 – Mockup da tela de página não encontrada



Fonte: Elaborado pelo autor.

Figura 235 – Mockup da tela de página não encontrada com usuário logado



Fonte: Elaborado pelo autor.

D.5 Protótipos

Figura 236 – Protótipo da tela Home

Fonte: Elaborado pelo autor.

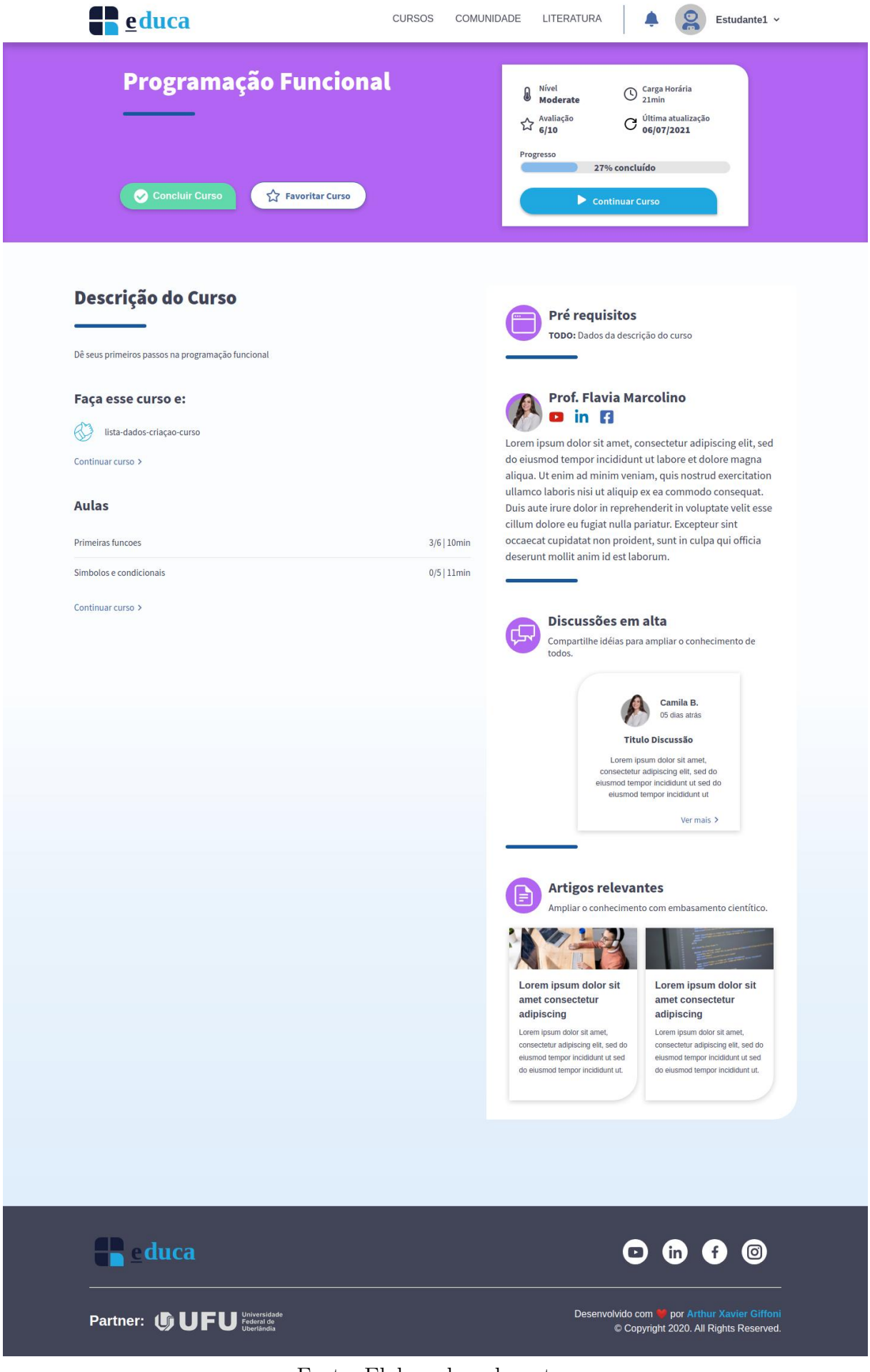
Figura 237 – Protótipo da tela de cadastro

Fonte: Elaborado pelo autor.

Figura 238 – Protótipo da tela de login

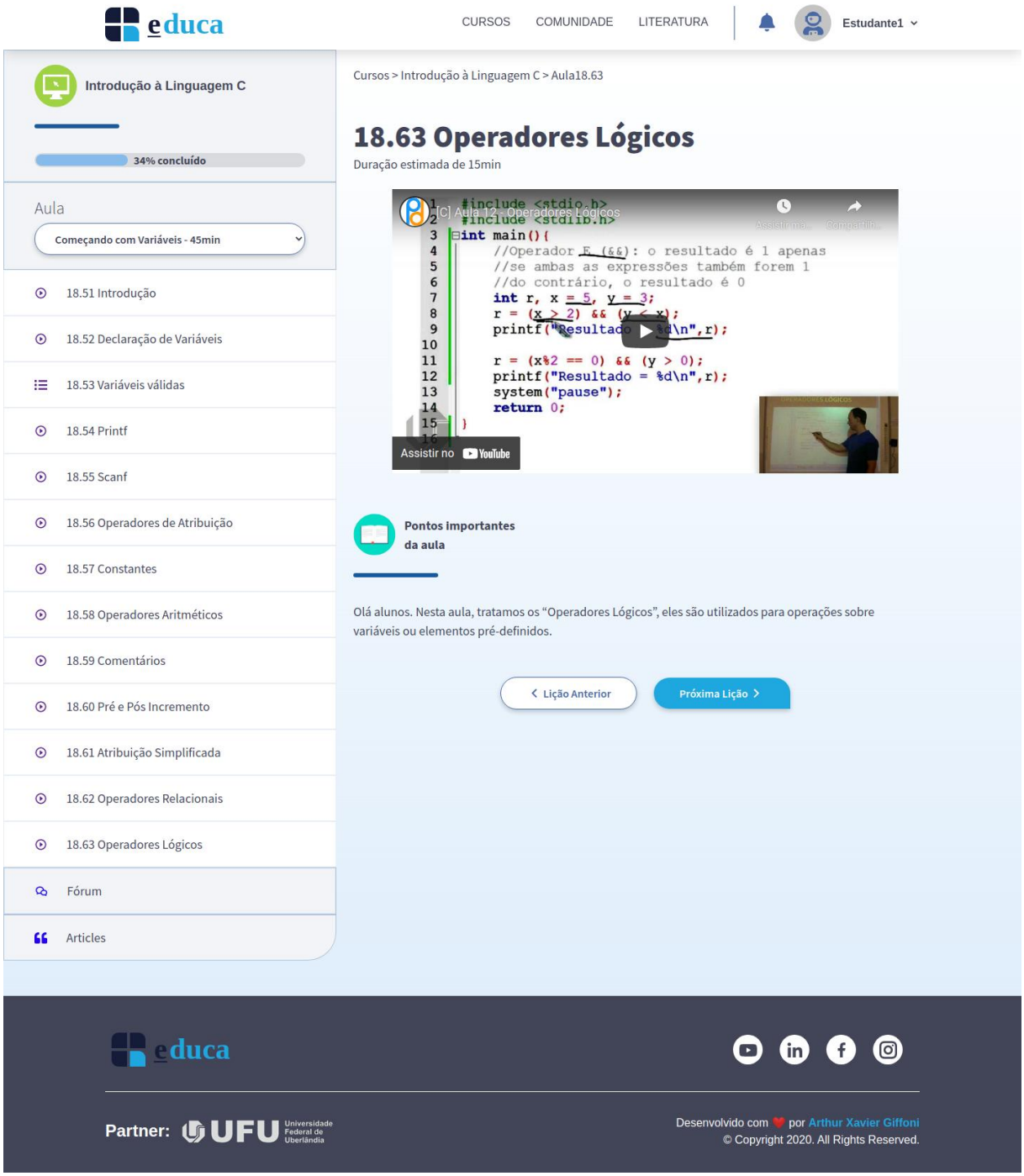
Fonte: Elaborado pelo autor.

Figura 239 – Protótipo da tela de curso



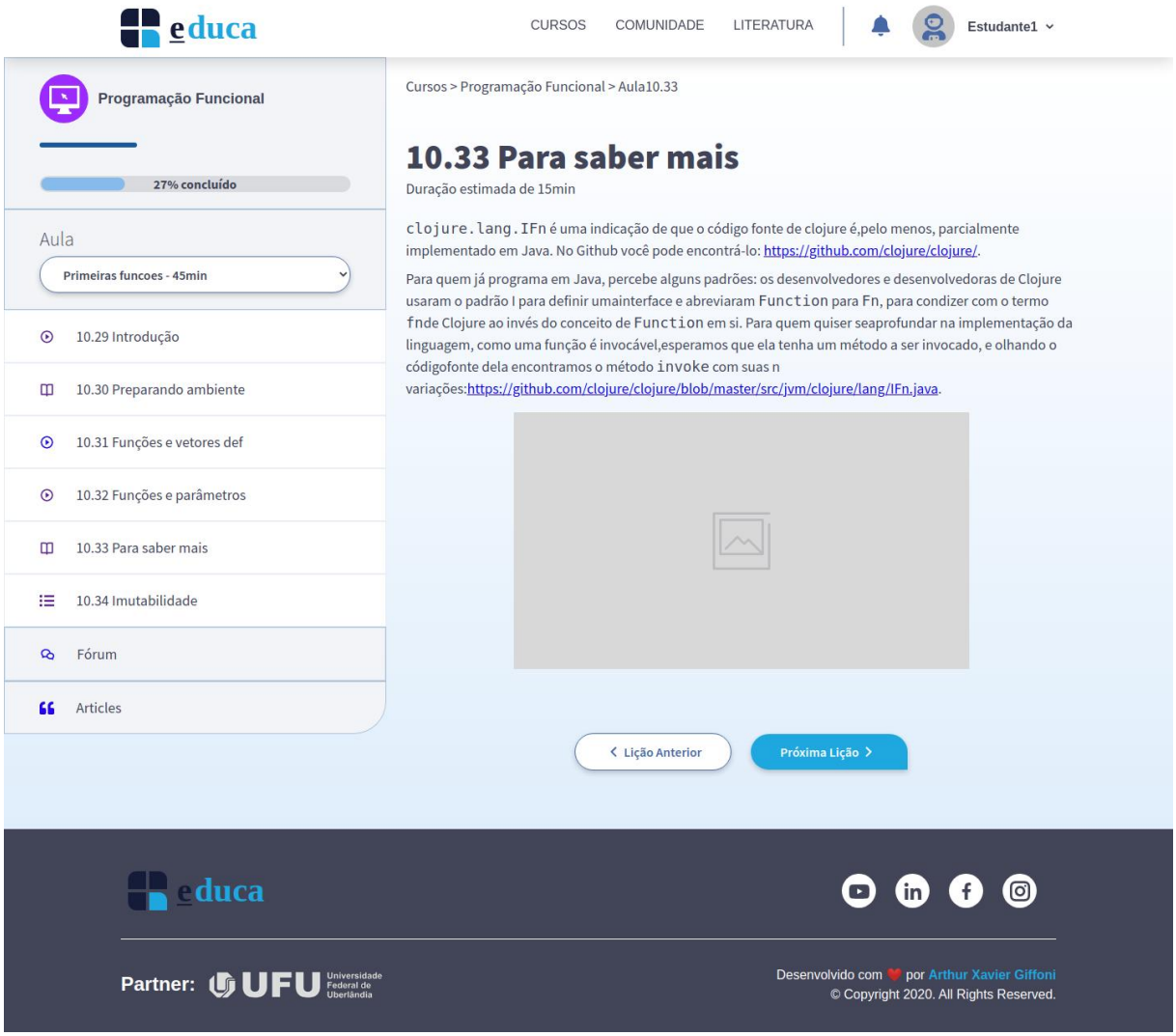
Fonte: Elaborado pelo autor.

Figura 240 – Protótipo da tela de atividade do tipo vídeo



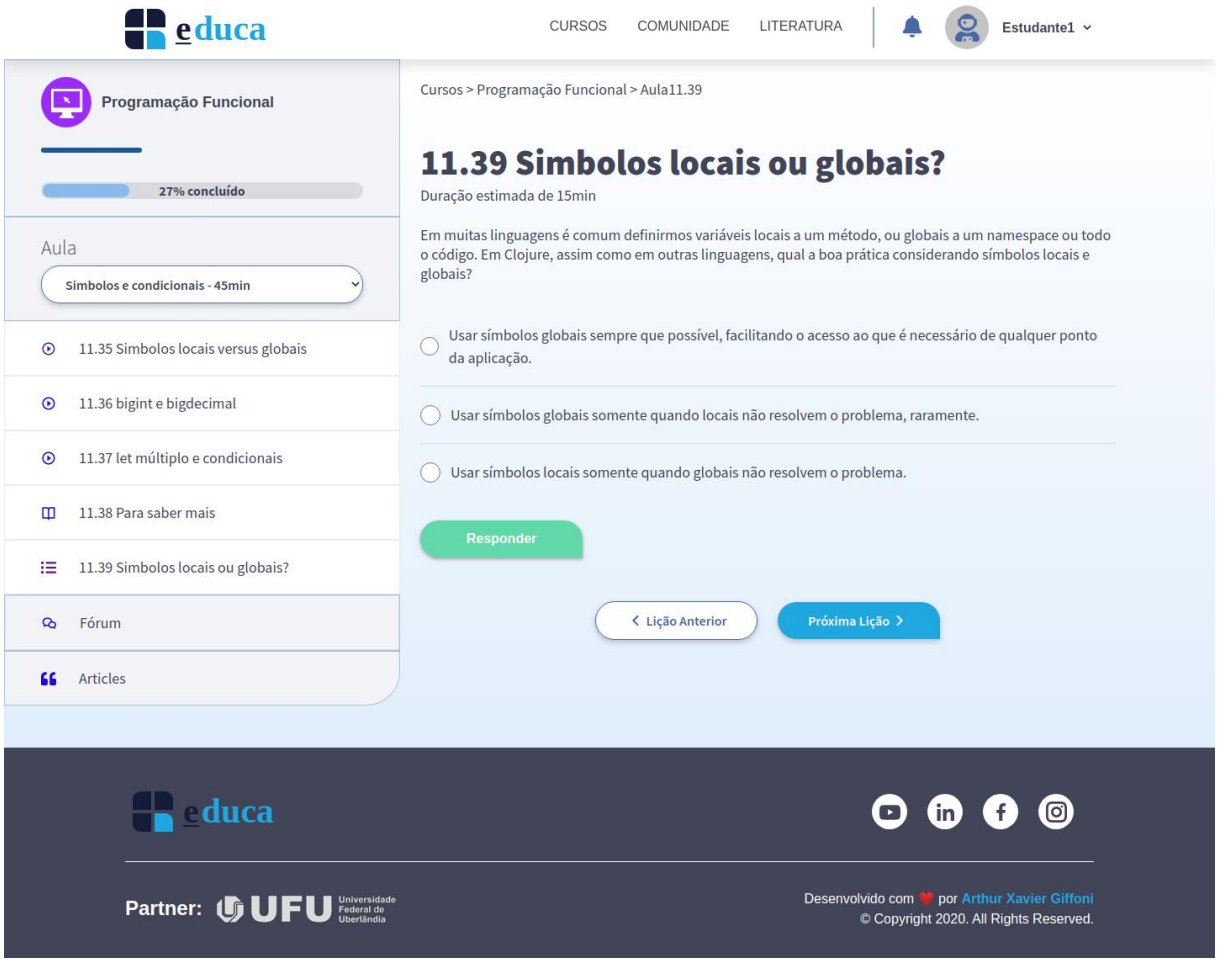
Fonte: Elaborado pelo autor.

Figura 241 – Protótipo da tela de atividade do tipo leitura



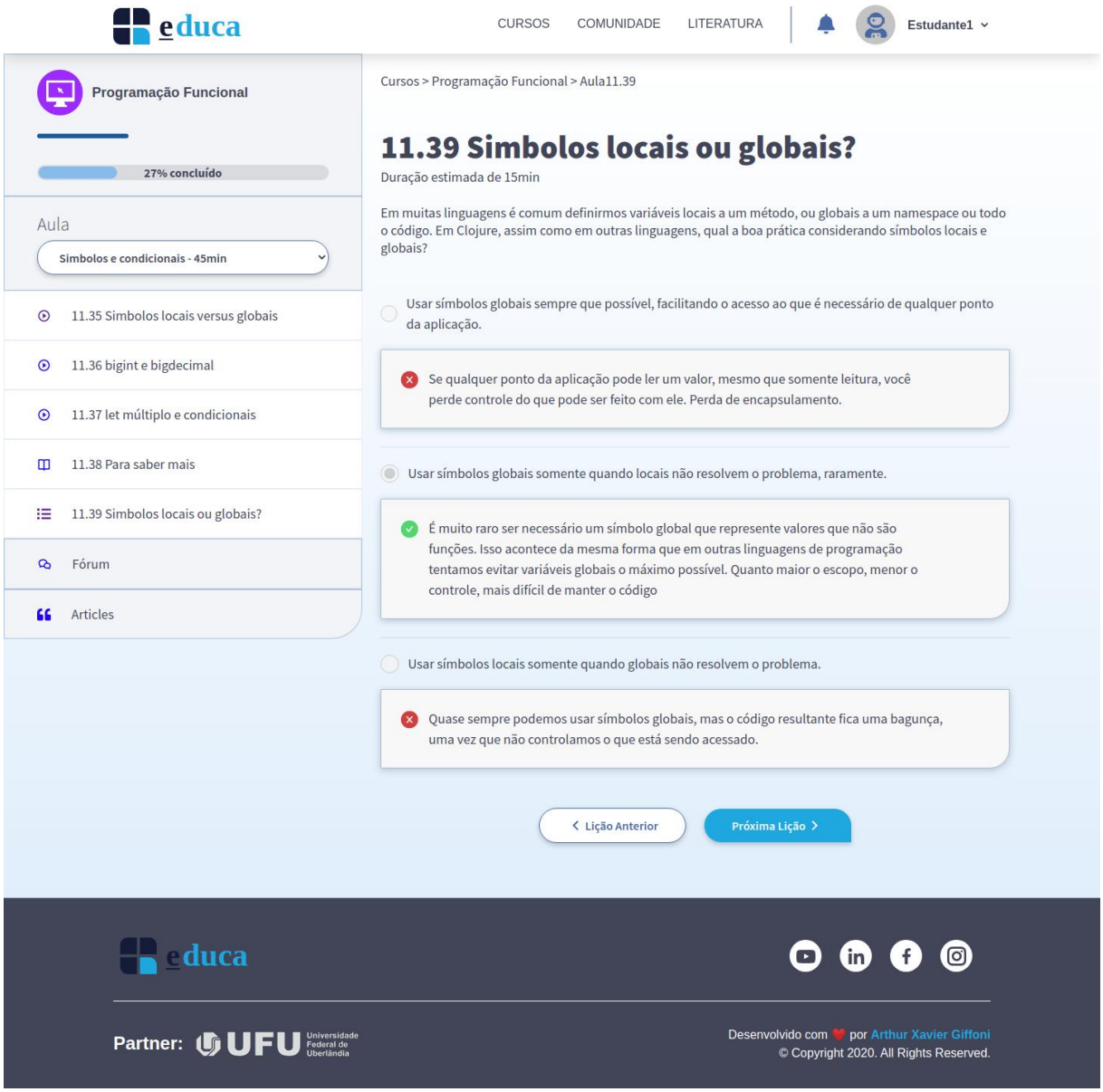
Fonte: Elaborado pelo autor.

Figura 242 – Protótipo da tela de atividade do tipo questionário



Fonte: Elaborado pelo autor.

Figura 243 – Protótipo da tela de atividade do tipo questionário respondido

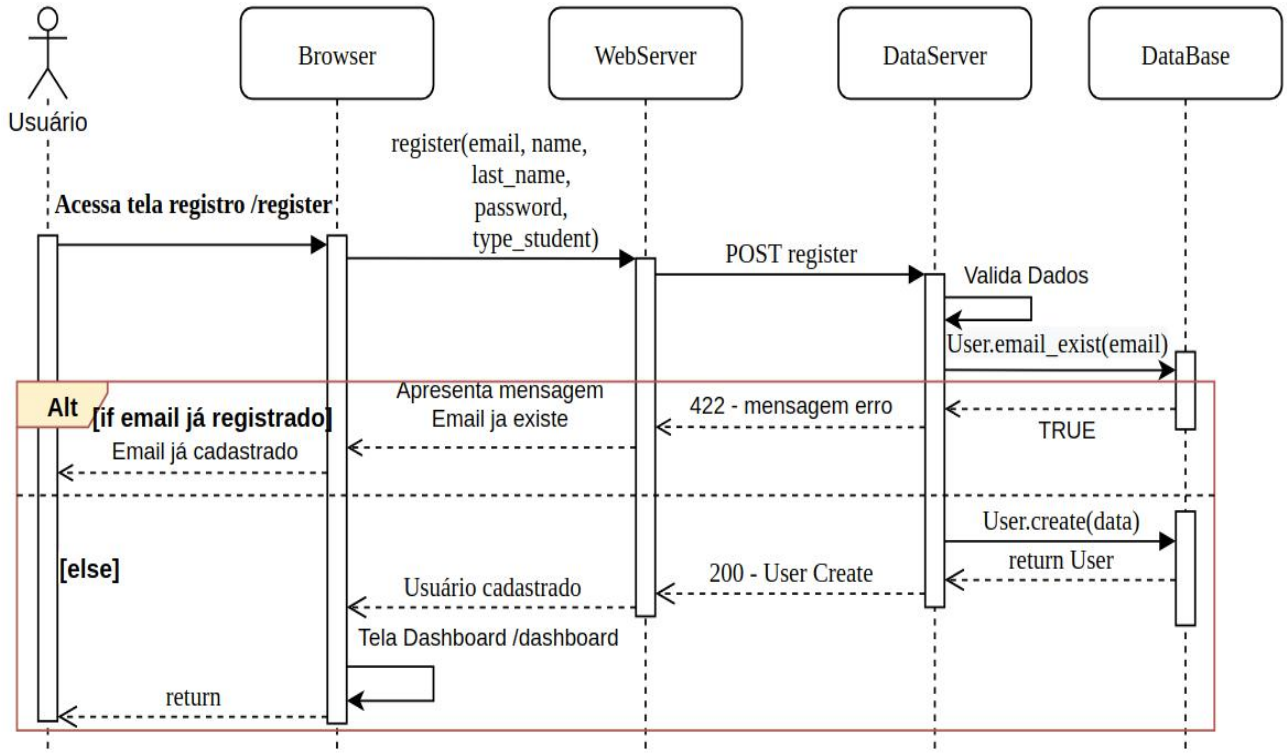


Fonte: Elaborado pelo autor.

APÊNDICE E – Diagramas de Sequência

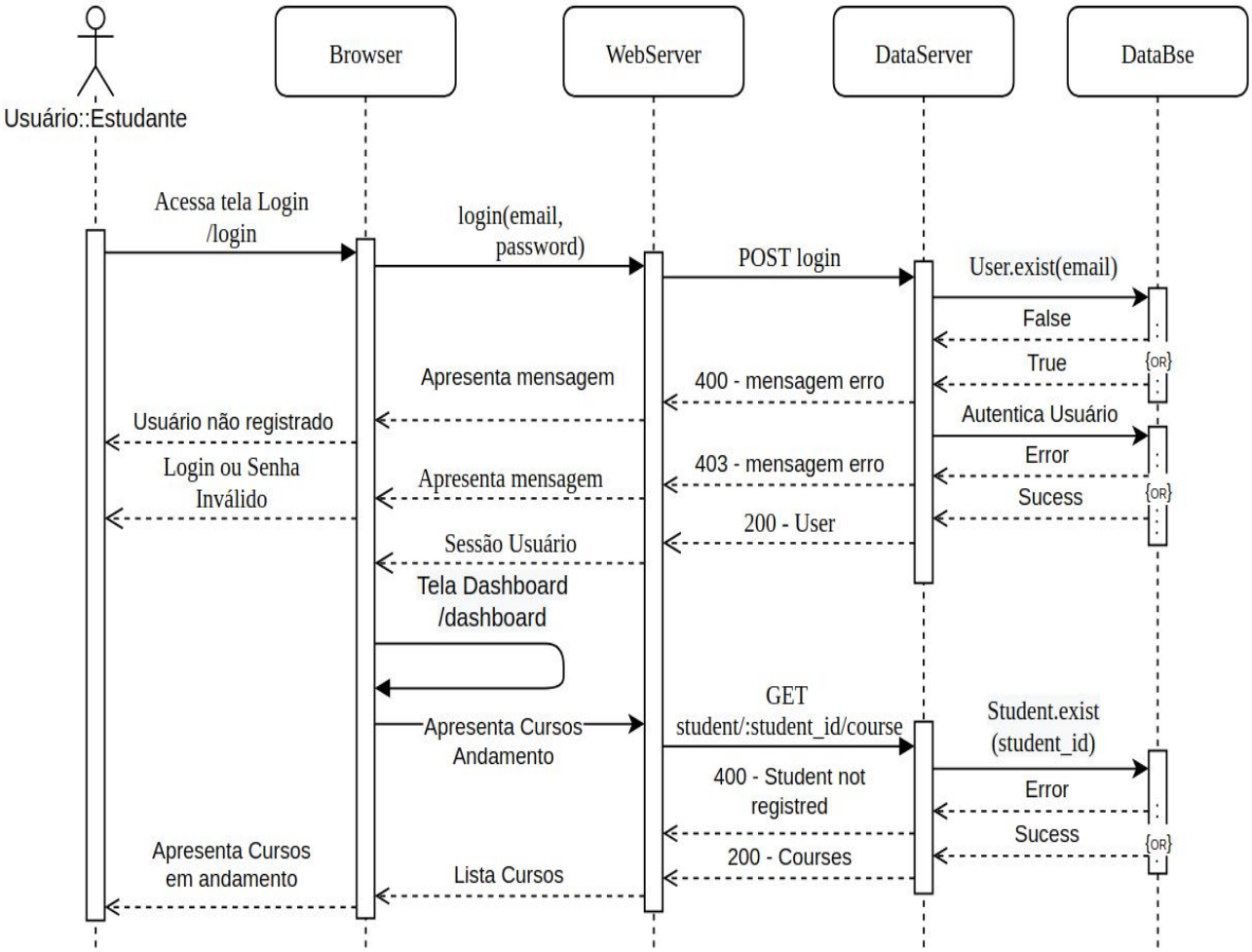
Este capítulo apresenta todos os diagramas de sequência.

Figura 244 – Diagrama de sequência do processo de um usuário registrar-se



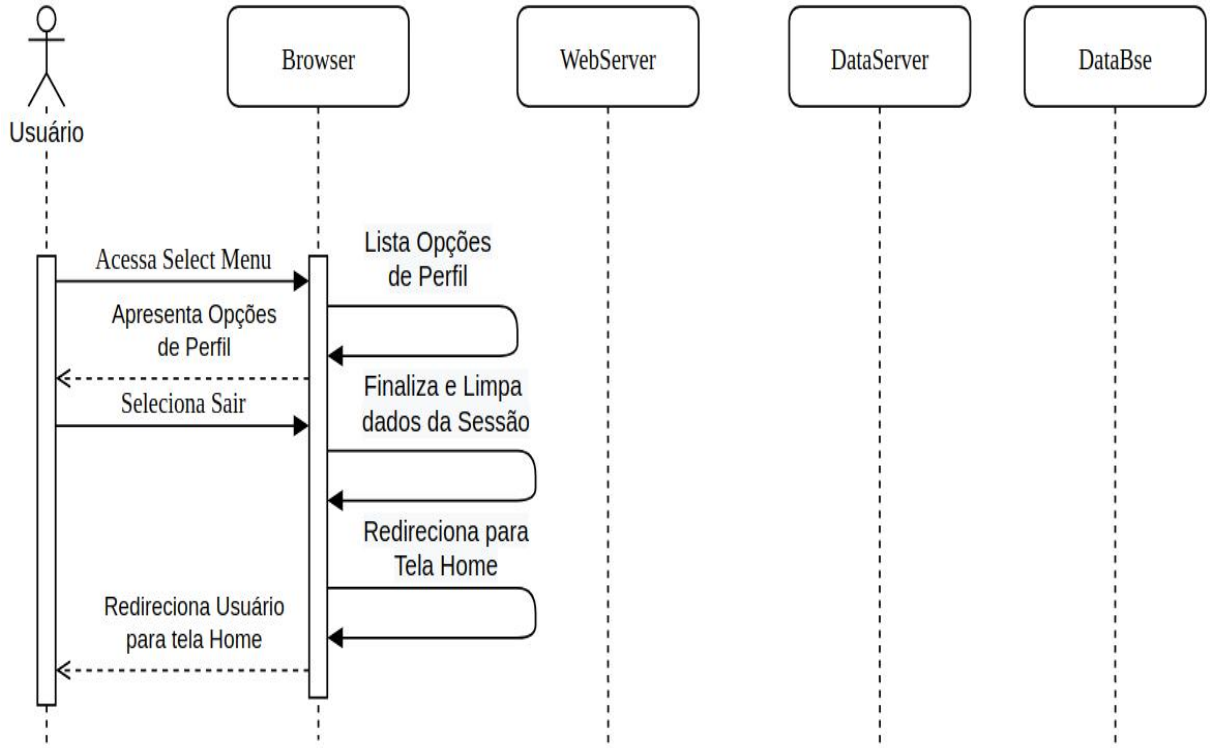
Fonte: Elaborado pelo autor.

Figura 245 – Diagrama de sequência do processo de *logar*



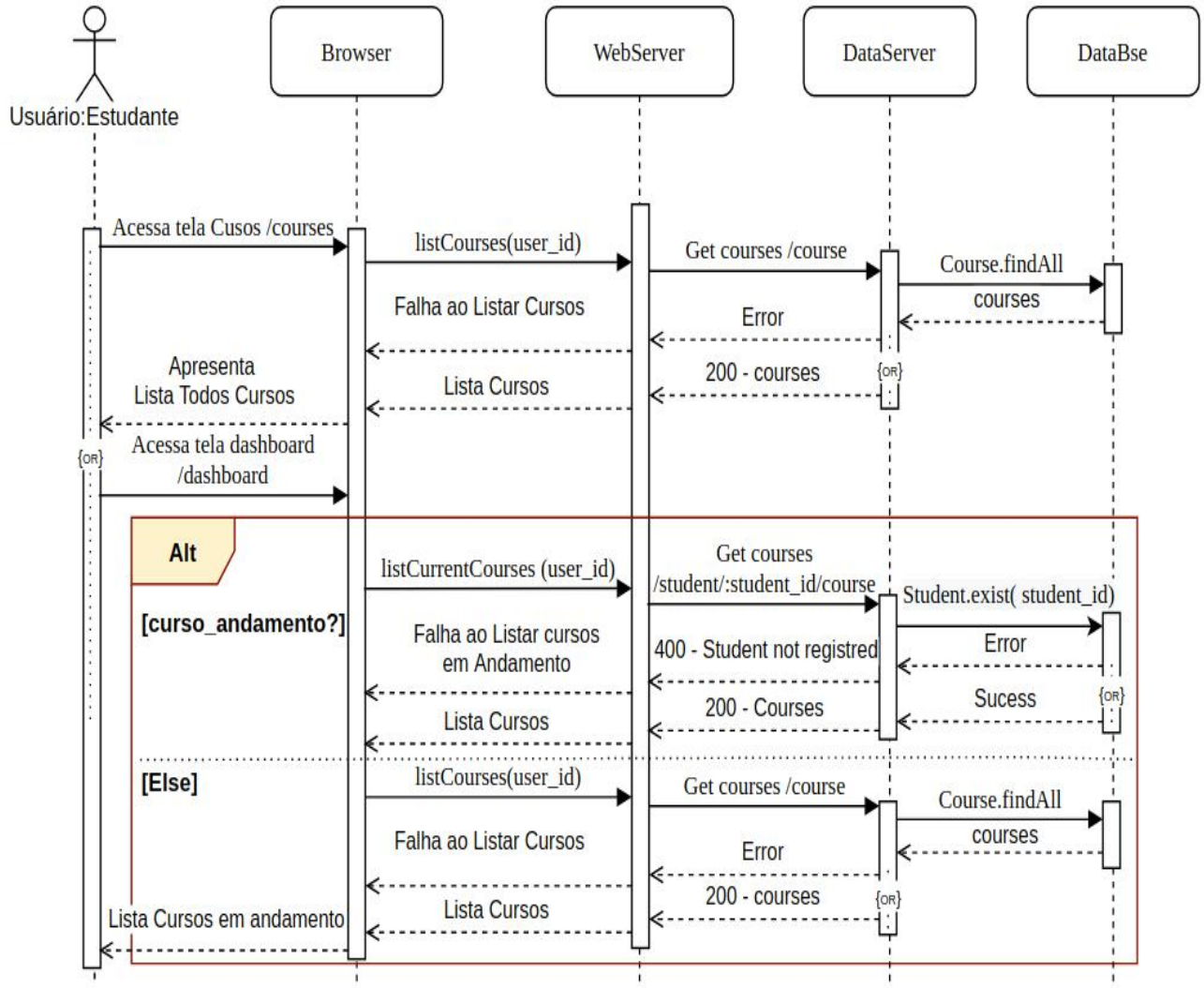
Fonte: Elaborado pelo autor.

Figura 246 – Diagrama de sequência do processo de *deslogar*



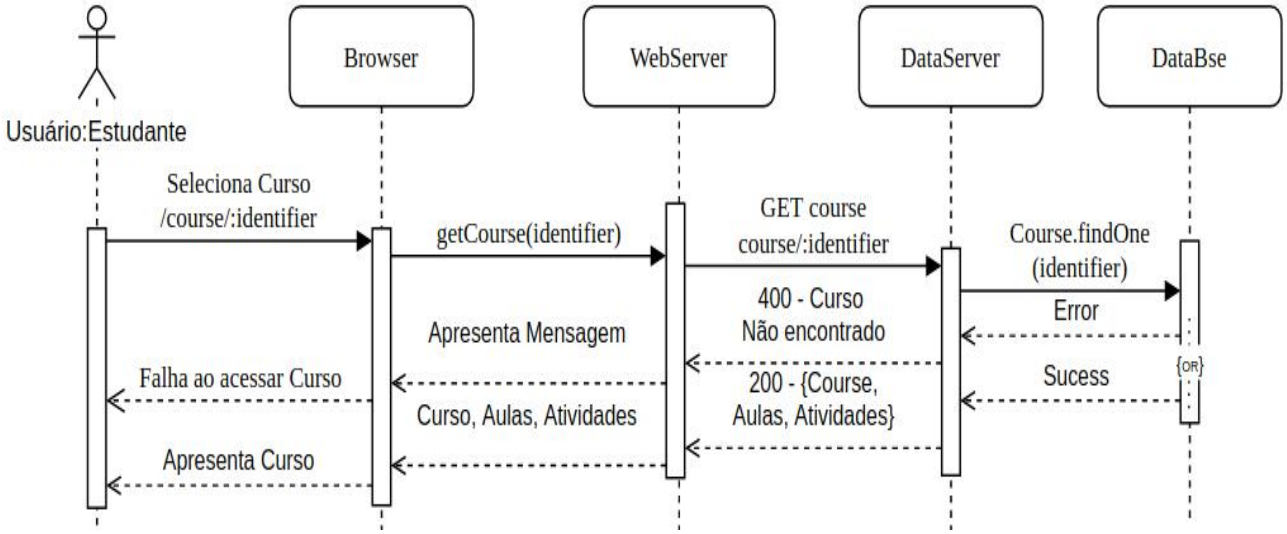
Fonte: Elaborado pelo autor.

Figura 247 – Diagrama de sequência do processo de listar cursos



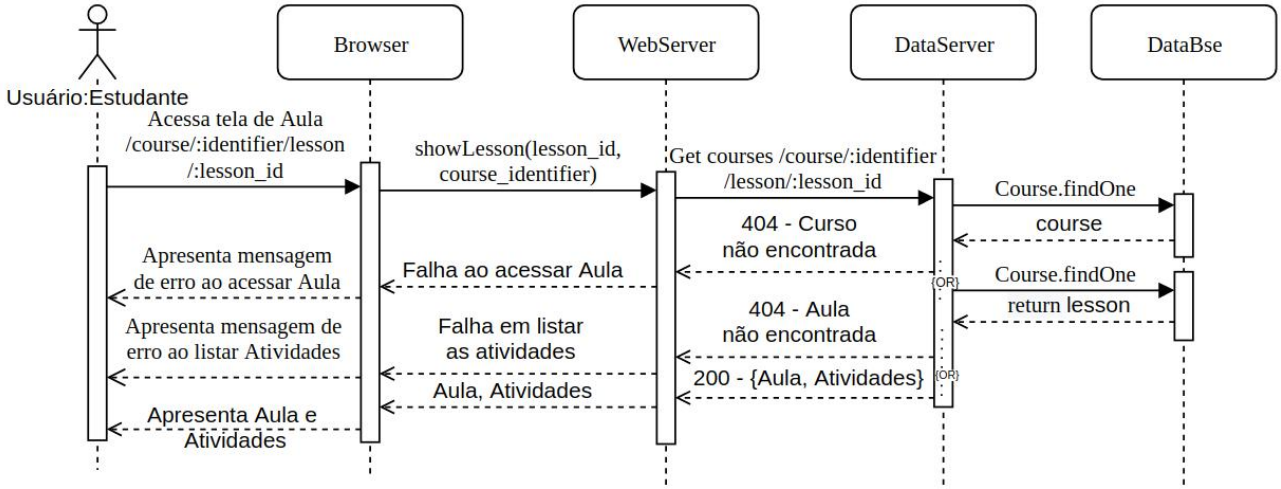
Fonte: Elaborado pelo autor.

Figura 248 – Diagrama de sequência do processo de selecionar curso



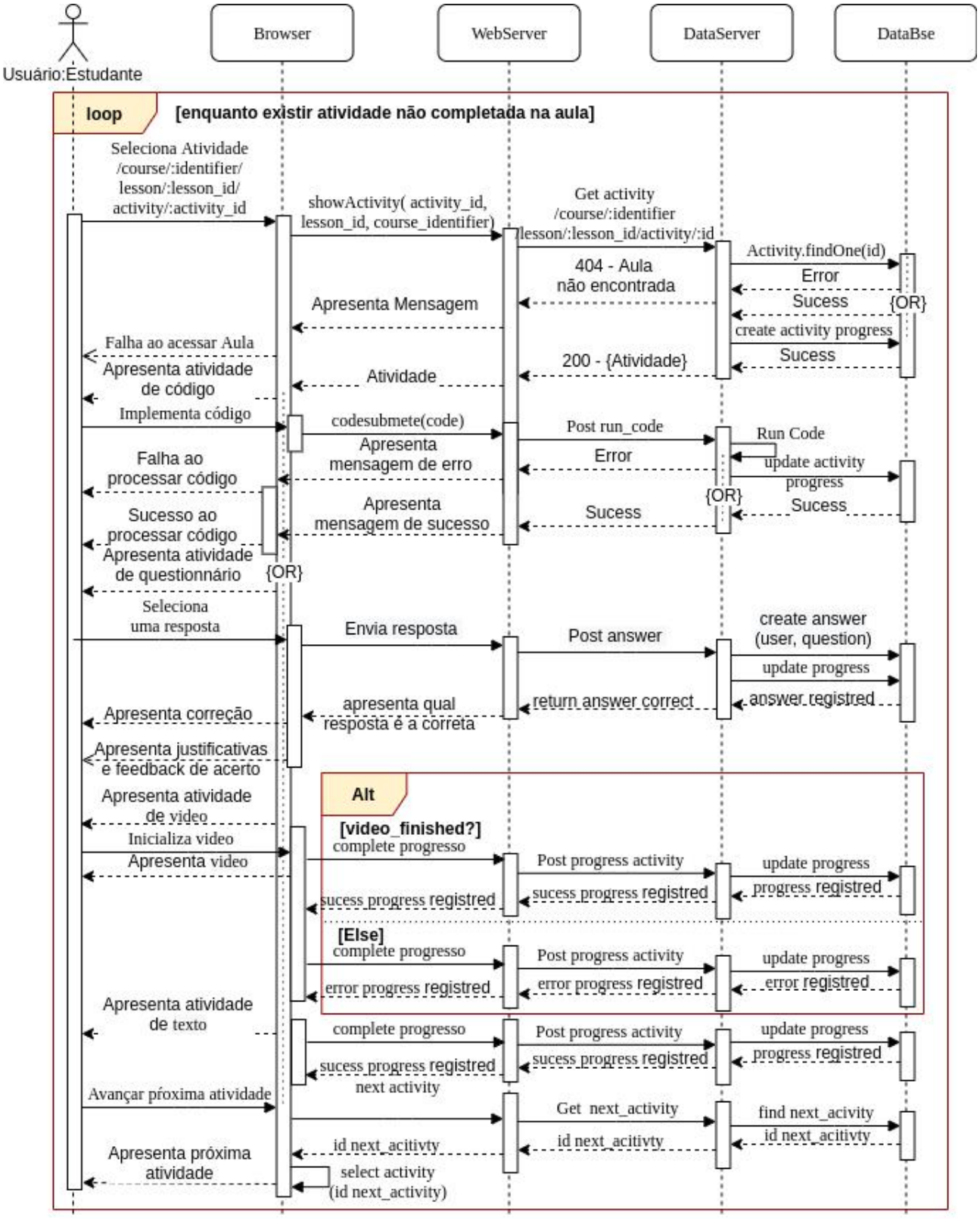
Fonte: Elaborado pelo autor.

Figura 249 – Diagrama de sequência do processo de acessar aulas e atividades



Fonte: Elaborado pelo autor.

Figura 250 – Diagrama de sequência do processo de cursar



Fonte: Elaborado pelo autor.