TEACHING LOGIC – UMA FERRAMENTA PARA AUXILIAR O ENSINO DE LÓGICA

Trabalho de Conclusão de Curso
Engenharia da Computação

Hartur Barreto Brito
Orientador: Prof. Dr. Gustavo Henrique Porto de Carvalho

Hartur Barreto Brito

TEACHING LOGIC – UMA FERRAMENTA PARA AUXILIAR O ENSINO DE LÓGICA

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco - Universidade de Pernambuco

Universidade de Pernambuco

Escola Politécnica de Pernambuco

Graduação em Engenharia de Computação

Orientador: Prof. Dr. Gustavo Henrique Porto de Carvalho

Recife - PE, Brasil

29 de novembro de 2016

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 21 de 12 de 2016, às 8:00 horas, reuniu-se para deliberar a defesa da monografia de conclusão de curso do discente **HARTUR BARRETO BRITO**, orientado pelo professor **Gustavo Henrique Porto de Carvalho**, sob título **TEACHING LOGIC – UMA FERRAMENTA PARA AUXILIAR O ENSINO DE LÓGICA**, a banca composta pelos professores:

Mêuser Jorge Silva Valença
Gustavo Henrique Porto de Carvalho

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

☒ Aprovada ☐ Aprovada com Restrições* ☐ Reprovada

e foi-lhe atribuída nota: 10,0 (DEZ)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O discente terá 03 dias para entrega da versão final da monografia a contar da data deste documento.

Mêuser Jorge Silva Valença
MÊUSER JORGE SILVA VALENÇA

Gustavo Henrique
GUSTAVO HENRIQUE PORTO DE CARVALHO

* Este documento deverá ser encadernado juntamente com a monografia em versão final.

Dedico este trabalho aos meus pais, Patricia Karla Barreto Brito e José Wilson de Souza Brito, ao meu irmão Eduardo Barreto Brito, à minha namorada Camila Gabriela de Oliveira Victor e a todos os meus familiares e amigos. Em especial, dedico à minha avó Zelda Azoubel Barreto (*in memoriam*).

Agradecimentos

A realização desse trabalho só foi possível graças à colaboração de muitas pessoas. Manifesto então meu agradecimento aos meus pais, Patricia Karla Barreto Brito e José Wilson de Souza Brito, por sempre batalharem para me dar suporte enquanto não posso "andar com minhas próprias pernas" e por ter me dado a oportunidade de estudar em uma das maiores universidades desse país, fazendo aquilo que sempre almejei. Ao meu irmão, Eduardo Barreto Brito, por estar sempre do meu lado, principalmente nas noites em claro, estudando para que eu não ficasse só. À minha namorada, Camila Gabriela de Oliveira Victor, por estar sempre me apoiando e ajudando a superar as fases difíceis. A todos os meus familiares, por compreenderem os meus momentos de ausência dedicados ao estudo durante a graduação, incentivando-me e ensinando-me a fazer tudo com amor e dedicação. Em especial, agradeço à minha avó Zelda Azoubel Barreto (*in memoriam*), por ter feito de mim quem eu sou, sempre me ensinando e incentivando a estudar e dar o máximo de mim para garantir uma vida melhor no futuro, além de todo o carinho, apoio e suporte que me deu. Aos meus amigos que contribuíram direta ou indiretamente para a realização desse sonho, ajudando-me em diversos momentos de dúvida durante a minha formação acadêmica. Agradeço também ao meu Orientador, o professor Gustavo Carvalho, por toda paciência e por ter acreditado em mim e no meu potencial, impulsionando-me nos momentos que eu mais precisei de estímulo. Aos professores que durante a graduação me proporcionaram o conhecimento não apenas racional, mas o da esfera da formação profissional e humana. Estes são professores cujos, sem nominar, terão os meus eternos agradecimentos. Por fim à Universidade de Pernambuco e a todos os seus funcionários e colaboradores por terem me proporcionado uma educação de qualidade, que proporcionaram a feitura desse trabalho.

Resumo

Esse trabalho tem como objetivo a criação de um software que auxilie os professores e os alunos durante o ensino e aprendizado de lógica em cursos superiores. Este programa busca ultrapassar as barreiras na transformação de linguagem natural em expressões lógicas, partindo do princípio da contextualização. Desenvolvido como uma aplicação web em C# na plataforma ASP.NET, utilizando o padrão MVVM (*Model View View Model*), o software permite que o professor crie, de maneira livre, vários domínios, para que o conteúdo de lógica seja contextualizado de diferentes formas. O software viabiliza também a criação de listas de exercícios, e a avaliação destas. Proporciona ainda um melhor planejamento do lançamento de listas de exercício, pois o programa permite que o professor selecione uma data específica para a lista ser liberada e fechada para os alunos. Tais fatores caracterizam o software como uma ferramenta de apoio ao ensino, com possibilidade de contribuição na formação dos profissionais de computação.

Palavras-chave: lógica, ensino, software.

Abstract

This work aims to develop a software to assist both students and lecturers with the process of teaching and learning Logic in higher education courses. The software intends to surpass the barriers in the transformation from the natural language into logical expressions, based on the principle of contextualization. Developed as a C web application, it makes use of the ASP.NET platform and the MVVM (Model View View Model) design pattern. The software allows the teacher to create domains freely, enabling the contextualization of logic in different ways. It also supports developing and marking assignments. Moreover, it aids the development of a schedule for these assignments, since it allows the lecturer to define specific dates for publishing and handing the assignments. These features make the software a teaching tool, which helps the way of learning and tutoring, with interesting possibilities of contributing to the education of computer professionals.

Keywords: logic, education, tool.

Lista de figuras

Lista de abreviaturas e siglas

HOL	High Order Logic
HTML	HyperText Markup
IDE	Integrated Development Environment
MVC	Model View Controller
MVVM	Model View View Model
PB	Principle of Bivalence
URL	Uniform Resource Locator

Sumário

1 Introdução

Dispositivos digitais estão cada vez mais presentes no dia-a-dia das pessoas e, com esse maior contato, elas estão ficando cada vez mais exigentes com a qualidade dos produtos que usam. Os softwares atuais exigem um alto nível de conhecimento dos profissionais da área para o seu desenvolvimento. Tal fato impulsiona a necessidade de uma base sólida e de qualidade nos cursos de computação.

Um dos conceitos básicos da área de computação, é o de lógica matemática. O estudo de lógica consiste em entender como conclusões podem ser alcançadas a partir de argumentos lógicos, estes últimos baseados em premissas (NOLT JOHN; ROHATYN, 2011). Esse conceito base de lógica é aplicado em diversas áreas da computação (MYERS JR., 1990), o que a torna importante para os demais conteúdos abordados por um curso de computação. Em sua tese, KIM (1995) demonstra que os alunos com mais familiaridade com lógica, tendem a ter mais sucesso no curso de ciência da computação.

Alguns exemplos de disciplinas, no curso de computação, que utilizam o conhecimento de lógica são: inteligência artificial, com sistemas de dedução, sistemas especialistas e formalismos; linguagens de programação, com linguagem funcional, processamento paralelo e projeto de linguagens; banco de dados, com representações de modelos de dados e consultas; engenharia de software, com testes de programas e especificações formais; e hardware, com design e otimização de circuitos e linguagens de hardware (MICHAUD, 2012; MYERS JR., 1990).

Apesar de ser uma matéria base para várias outras, existe uma grande dificuldade por parte dos alunos no entendimento do conteúdo visto na disciplina de lógica (MICHAUD, 2012; HERMAN et al., 2012). A partir de testes realizados, Herman et al. (2012) observou que uma das principais dificuldades encontradas pelos alunos é a de transformar linguagem natural em expressões lógicas, devido às abstrações utilizadas pelas representações formais. Ele também observou que, quando utilizados conectores que remetem diretamente às expressões (ex.: e, ou), os alunos conseguiam realizar a tradução “frase-expressão lógica”, mas, quando são utilizados conectores que necessitam de mais atenção, por não serem usuais na fala (ex.: ou exclusivo, se somente se), os alunos apresentaram maior dificuldade na tradução.

O programa desenvolvido nesse estudo busca minimizar os problemas supracitados e aperfeiçoar o ensino de lógica com a promoção da interação turma-aluno-professor, e também com possibilidade de criação de diferentes domínios. Caracterizando-o como um software adaptável, capaz de representar diferentes situações. Por exemplo, com o software desenvolvido neste trabalho, o professor pode ilustrar os conceitos de lógica em diferentes

domínios, tais como, formas geométricas, animais marinhos, sinalização de trânsito, entre outros.

1.1 Objetivos

O objetivo geral desse trabalho é o desenvolvimento de uma ferramenta que apoie o ensino e o aprendizado de lógica, partindo do princípio da definição dinâmica pelo professor de exemplos que concretizem os conceitos abstratos normalmente ensinados.

O objetivo geral supracitado desdobra-se nos seguintes objetivos específicos:

- Levantar as principais dificuldades no aprendizado de lógica;
- Analisar soluções propostas para minimizar estas dificuldades;
- Desenvolver uma ferramenta de apoio ao ensino de lógica.

1.2 Resultados e impactos esperados

Esse trabalho analisou as dificuldades encontradas pelos alunos ao estudar lógica, além das soluções sugeridas via software pela literatura, desenvolvendo, a partir disso, uma aplicação que serve de apoio ferramental para o ensino de lógica, com o objetivo de minimizar os obstáculos encontrados.

Espera-se que a ferramenta possua um impacto positivo no que se refere ao ensino de lógica por possibilitar diferentes contextualizações do conteúdo e que seja possível acessá-la a partir de qualquer dispositivo, facilitando e aumentando a interação entre discentes e docentes.

A aplicação permite que o professor atribua listas de exercícios para os alunos, além de permitir a avaliação das respostas enviadas por eles, tornando possível o acompanhamento do desenvolvimento pessoal de cada aluno. Portanto, espera-se que o principal resultado deste trabalho, a ferramenta *Teaching Logic*, contribua para uma melhor formação em lógica.

1.3 Estrutura do documento

Este documento encontra-se estruturado conforme descrito a seguir. No Capítulo 2, são explanadas as informações encontradas na revisão bibliográfica, dentre elas, problemas enfrentados pelos alunos no aprendizado de lógica, metodologias mostradas como mais adequadas para o ensino e softwares que foram desenvolvidos visando a melhoria do aprendizado do conteúdo de lógica. Capítulo 3 descreve como foi realizada a pesquisa

bibliográfica e como a ferramenta proposta nesse trabalho foi desenvolvida. No Capítulo 4 são mostrados os resultados obtidos com o desenvolvimento da ferramenta *Teaching Logic*, descrevendo seu comportamento e apresentando suas telas, além da comparação da ferramenta desenvolvida com as encontradas na referência bibliográfica descritas no Capítulo 2. Por fim, no Capítulo 5 são mostradas as conclusões tomadas a partir da análise da ferramenta desenvolvida nesse trabalho e as pretensões de trabalhos futuros.

2 Revisão bibliográfica

Com o intuito de embasar o desenvolvimento da ferramenta *Teaching Logic*, realizou-se uma revisão bibliográfica para levantar metodologias (Seção 2.1) e ferramentas (Seção 2.2) já desenvolvidas para auxiliar o ensino de lógica.

2.1 Metodologias para ensino de lógica

Como dito anteriormente, Herman et al. (2012) realizou uma pesquisa que descreve as dificuldades encontradas pelos alunos ao estudar lógica. A partir da análise destes resultados, foi observado que o principal problema no entendimento do conteúdo de lógica consiste na contextualização dos assuntos. Na tentativa de mensurar a dificuldade das atividades que envolvem lógica, este estudo definiu ainda alguns parâmetros a serem avaliados: familiaridade da especificação, ambiguidade e interferência, tipos de operadores boleanos, representação do problema e o contexto do problema.

Os reflexos desse panorama mostram a importância da compreensão da lógica, desde sua base, sendo a passagem de conhecimento uma etapa que merece atenção especial. Bruner R. R. Oliver (1966) categorizou o processamento de informação em três formas de representar experiências sensoriais e o pensamento, sendo elas:

- Representação simbólica: Consiste em adquirir conhecimento através de símbolos (por exemplo, texto ou sinais). As representações simbólicas são concisas e são especialmente adequadas se alguém já tiver uma percepção intuitiva apropriada. É responsável pela capacidade intelectual.
- Representação por ícone (imagem): os fatos são representados como imagens. Objetos concretos, eventos ou procedimentos são compreensíveis como visualizações. Responsável pela ampliação dos sentidos. O folheto de um hotel ou um mapa da cidade, muitas vezes é o suficiente para obter a imagem e orientar-se.
- Representação enativa (músculos): é a aprendizagem através do seu próprio fazer. Este é o caso das crianças, por exemplo, que aprendem a andar de triciclo sem precisar ler um manual de instruções. Elas aprendem através de suas próprias ações, agrupamento de objetos e através da observação.

Arnold, Langheinrich e Hartmann (2007), no seu estudo, acrescentam uma quarta forma de representação, levando em conta o contexto tecnológico que a sociedade está inserida. Sendo ela:

- Representação enativa (virtual): através da manipulação em um ambiente de software, processos enativos são simulados. Um exemplo são os ambientes de aprendizagem, no quais os alunos podem controlar robôs virtuais em suas telas.

Inserindo essas informações no contexto virtual, Arnold, Langheinrich e Hartmann (2007) (apud R. Schulmeister 2003), fundamentaram seis níveis de interação com um software; são eles:

- Nível 1: há apenas a exposição à informação, sem que haja nenhuma interação
- Nível 2: os usuários navegam através da representação de informações.
- Nível 3: há representações múltiplas do conteúdo.
- Nível 4: o usuário pode modificar os parâmetros da representação.
- Nível 5: o usuário pode manipular o conteúdo em si.
- Nível 6: o usuário pode criar e manipular objetos e ver o sistema reagir, ou seja, recebe feedback.

Ainda no trabalho destes autores, é citado um modelo de ensino chamado de "*rule e.g. rule*", que é usado tradicionalmente no ensino de várias matérias e consiste em primeiro apresentar uma regra aos alunos, para depois mostrar um exemplo de aplicação dessa regra, e então mostrar novamente a regra para que ela possa ser fixada pelos alunos. Os autores observaram que essa metodologia não é interessante para apresentar o conteúdo de lógica, tendo em vista que os alunos têm dificuldade de conectar as representações formais com contextos do seu dia-a-dia. Para resolver esse problema, os autores criaram o modelo "*e.g. rule e.g. rule*", que consiste em primeiro mostrar um exemplo da aplicação do conteúdo, para só então mostrar a regra, utilizar um outro exemplo para fixar a contextualização da regra e, por fim, mostrar a regra novamente para que os alunos fixem.

Levando em consideração esses parâmetros, softwares já foram desenvolvidos com o intuito de facilitar o aprendizado e reduzir o nível de incompreensão da lógica pelos alunos. A seguir, alguns destes softwares são apresentados na Seção 2.2.

2.2 Softwares utilizados para auxiliar o ensino de lógica

As próximas seções descrevem softwares utilizados para auxiliar o ensino de lógica: WinKE (Seção 2.2.1), InfoTraffic (Seção 2.2.2), ensinando lógica a partir da modificação de regras de jogos virtuais (Seção 2.2.3) e Tarski's World (Seção 2.2.4).

2.2.1 WinKE

O WinKE é um software que auxilia a prova de expressões lógicas e tem a intenção de auxiliar o processo de ensino de lógica e razão. Ele se baseia no sistema KE desenvolvido pelos próprios autores em (D'AGOSTINO, 1994), utilizando principalmente o princípio da bivalência (PB – do Inglês *Principle of Bivalence*), que define que toda sentença deve ter apenas um valor, podendo ser ele verdadeiro ou falso. Para que seja possível utilizar o PB em sua resolução, utiliza de um tableau analítico enfraquecido para representar o processo de prova de uma expressão (D'AGOSTINO et al., 1998).

Sua interface é baseada no design do Windows 95, sistema operacional para o qual ele foi destinado a ser usado. Portanto, mantém um visual simples, com 4 janelas principais: a principal, a gráfica, a que permite uma visualização de um mapa de todo o quadro de desenho e uma janela com uma caixa de ferramentas. Desta forma, o entendimento do software é intuitivo e simples, já que cada janela detém um objetivo claro (D'AGOSTINO et al., 1998). Ele oferece também uma representação em forma de grafos da expressão, onde cada aresta do grafo direcionado representa uma premissa, mostrando, de outra maneira, as condições necessárias para que a fórmula seja falsa, provando que ela não é uma tautologia (ENDRISS, 2000). Um exemplo de tela do WinKE pode ser visto na Figura 1.

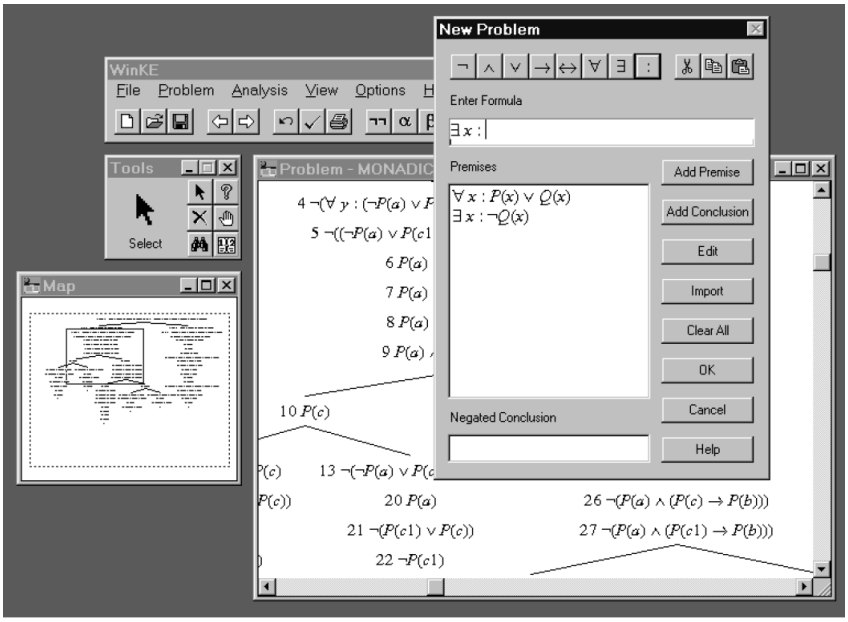

Figura 1 – Tela do WinKE

[Fonte: reproduzido de D'Agostino et al. (1998)]

Retratando sua aplicação no ensino, o WinKE é dividido em 3 modos de ensino: o modo pedagógico, modo supervisionado e o modo assistido. O modo mais apropriado para iniciantes é o pedagógico, no qual é realizada uma análise sintática e semântica de

todas as regras que são escritas. Por outro lado, para um usuário mais familiarizado com o assunto de lógica, o WinKE oferece o modo supervisionado, onde apenas a sintaxe das regras é checada, visando prevenir erros de escrita. Por fim, para um usuário ainda mais avançado, existe o modo assistido, onde o WinKE fornece um sistema de prova de teoremas de primeira ordem, além de fórmulas pré-selecionadas para que o usuário escolha usar a mais apropriada. (ENDRISS, 2000)

Ainda sobre sua ajuda no ensino, uma característica interessante no WinKE é que o software permite que sejam salvos 3 diferentes tipos de arquivos: prova completa, prova em andamento ou um problema, permitindo que diferentes tipos de exercícios sejam criados pelo professor. Como complemento, ele também permite que os erros sejam corrigidos e retratados em diferentes níveis, possibilitando que o professor exija diferentes classes de conhecimento de seus alunos ao longo do período letivo.

No auxílio a provas, o WinKE permite que sejam mostrados, passo-a-passo, cada galho gerado no tableau analítico, ou que todos sejam gerados de uma só vez, permitindo a visualização direta do contra-modelo, fechando os que não deram resultados com um X e marcando os galhos que explicitam um contra modelo com um círculo. Com isso, ele mantém uma exibição semelhante à uma prova por tableau feita a mão no papel, facilitando o entendimento do usuário por poder visualizar as provas da mesma forma que são representadas em seu dia-a-dia nas aulas (D'AGOSTINO et al., 1998).

Aparentemente, a interação entre aluno e professor é feita de forma manual, trocando arquivos de perguntas e respostas.

2.2.2 InfoTraffic

O InfoTraffic é um software que visa facilitar o entendimento de conceitos fundamentais de lógica proposicional a partir do princípio de que o ensino se torna muito mais eficiente quando o objeto de estudo está relacionado com situações do dia-a-dia do aluno. Para isso, o programa faz uso de uma metáfora, relacionando as fórmulas da lógica proposicional com o tráfego automobilístico, que é uma situação corriqueira para a maioria, senão todos os estudantes. Desta forma, torna-se mais fácil o aprendizado dos conceitos básicos de lógica, o que o leva a ser usado até no ensino médio (ARNOLD; LANGHEINRICH; HARTMANN, 2007).

A relação entre lógica e trânsito funciona da seguinte forma: as variáveis se transformam em faixas de trânsito e os valores verdade das variáveis são os semáforos de cada faixa, sendo verde o valor verdade verdadeiro e vermelho o valor verdade falso. O objetivo do usuário é definir se, de acordo com a combinação de valores verdade, o trânsito será seguro (valor verdade da expressão lógica verdadeiro) ou não (valor verdade da expressão lógica falso). Visualmente, no tráfego das faixas, é definido como trânsito seguro se nenhum

par de faixas que se cruzam têm semáforos de cor verde ao mesmo tempo, o que resultaria em um acidente na vida real. Caso exista um par de faixas que se cruzam e ambos seus semáforos são verdes, o trânsito deixa de ser seguro e a expressão lógica terá um valor verdade falso.

A partir das combinações de valores verdades e sinais de trânsito, os alunos constroem uma tabela verdade da expressão lógica, baseando-se no auxílio visual. Após completarem a tabela verdade, os alunos podem conferir se os resultados obtidos por eles correspondem ao resultado real ao fazer uma simulação da situação, checando se aconteceria ou não um acidente, através do botão de simulação.

Sua interface é dividida em 5 partes principais de uma única janela (Figura 2): a situação do tráfego, a tabela verdade que está sendo construída, que também separa entre situações seguras e não seguras, a parte onde é explicitada a expressão em questão, a parte onde possibilita a alteração da expressão e, por fim, controles que permitem reiniciar a execução, obter dicas e realizar simulações.

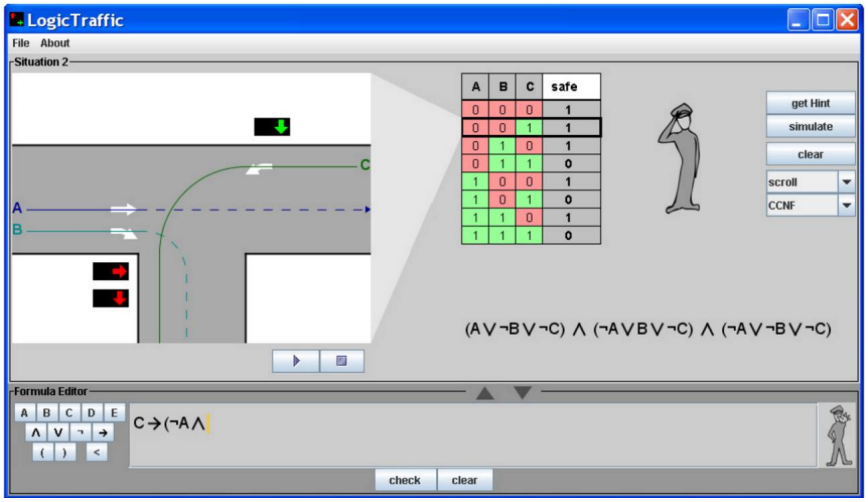

Figura 2 – Tela do InfoTraffic

[Fonte: reproduzido de Arnold, Langheinrich e Hartmann (2007)]

Com o intuito de tornar o entendimento ainda mais simples, é possível que o usuário altere o funcionamento das faixas e dos semáforos tanto a partir da expressão, quanto a partir da tabela ou simplesmente clicando nos semáforos, alterando seus valores verdade. Desta forma, o programa possibilita que o usuário perceba, de várias formas, qual o impacto de cada alteração, facilitando o entendimento.

Através de um modo criativo de representar o universo da lógica proposicional, o InfoTraffic aborda conceitos de variáveis, valor verdade, operadores lógicos, fórmulas, equivalência de fórmulas e formas normais. Entretanto, nada foi citado no artigo que descreve a ferramenta (ARNOLD; LANGHEINRICH; HARTMANN, 2007) sobre a intera-

ção entre professor e alunos, levando a entender que o objetivo da ferramenta é apenas a representação do universo da lógica proposicional afim de facilitar o aprendizado, não apresentando um módulo no qual o professor possa designar listas de exercício para os alunos com o intuito de avaliá-los posteriormente.

2.2.3 Ensinando lógica a partir da modificação de regras de jogos virtuais

A abordagem explorada por Weng, Tseng e Lee (2010) consiste na utilização de jogos virtuais para ensinar lógica. Os autores citam que, apesar de existirem outros exemplos de jogos que utilizam da lógica booleana, eles foram desenvolvidos apenas para fins de diversão. Em seu artigo, Weng, Tseng e Lee (2010) focam no jogo clássico do Pac-Man, por ser um jogo difundido e de características e regras simples, com regras e personagens bem definidos.

Os autores descrevem que a decisão de utilizar jogos para ensinar lógica foi tomada com o objetivo de incentivar os alunos a estudarem por gostarem de passar o tempo jogando. Além disso, por meio do jogo, os alunos terão como visualizar a consequência das alterações realizadas nas expressões pelas animações e pelas suas interações, facilitando assim o entendimento da matéria, além de tornar interessante a aprendizagem, estimulando os discentes a procurarem as modificações que cada alteração provoca.

Entretanto, de acordo com os autores, fundir o estudo de lógica com jogos não é uma tarefa simples. A primeira barreira descrita para possibilitar essa junção é a de avaliar como seriam modeladas e manipuladas as regras dos jogos. Para resolver tal problema, foi definida uma gramática livre de contexto, que foi utilizada durante o desenvolvimento da ferramenta. Dessa forma, foram criadas funções baseadas em ações do jogo em questão (Pac-Man) e, com essas funções simples, o usuário pode combiná-las e criar e/ou modificar as regras do jogo, utilizando lógica booleana simples. Por exemplo, uma das regras no jogo clássico de Pac-Man é que se o personagem comer uma *pill* e tocar em algum dos fantasmas, será adicionado mais um ponto na pontuação do jogador. Traduzindo para a gramática do jogo, utilizando a lógica booleana, tem-se: $Ate\_Pill \wedge Touched\_Ghost \longrightarrow Score$. Perceba que Ate_Pill, $Touched\_Ghost$ e $Score$ são ações referentes ao jogo e todas são tratadas como funções básicas.

Contudo, como as regras são modificadas e/ou criadas baseadas em funções que se referem a ações básicas do jogo, esse tipo de abordagem para o ensino da lógica só pode ser utilizado com um jogo específico, uma vez que cada jogo apresenta ações próprias.

Essa abordagem possibilita que o usuário observe o comportamento das expressões booleanas, explore o jogo modificado por ele mesmo e, assim, entenda melhor como funciona cada operador lógico. É importante notar que, quanto maior a dificuldade do jogo, maior será a dificuldade de expressar suas regras através de expressões booleanas, gerando um

desafio para o usuário que deseja ter uma experiência mais divertida e duradoura.

Um dos programas que utiliza essa abordagem para o entendimento de lógica é o The Scratch, desenvolvido por Resnick no MIT Media Lab, em 2007. A interface do The Scratch é bastante intuitiva, separando claramente, em uma janela principal, o jogo gerado pelas regras escolhidas pelo usuário, a aparência dos personagens (*sprites*), as regras do jogo (*script*), as fórmulas que podem ser usadas para criar ou alterar regras do jogo em questão e, por fim, a aba de controle. Assim, por ser auto-explicativo, o The Scratch exige apenas um conhecimento mínimo das expressões booleanas para a criação das regras e permite que o jogo seja alterado de inúmeras maneiras, limitadas apenas pelas ações básicas do jogo.

Para a criação da ferramenta na qual são alteradas as regras do jogo Pac-Man, Weng, Tseng e Lee (2010) utilizou o The Scratch. Exemplos de funções lógicas utilizadas para definir as regras do jogo e da tela do The Scratch podem ser vistos na Figura 3.

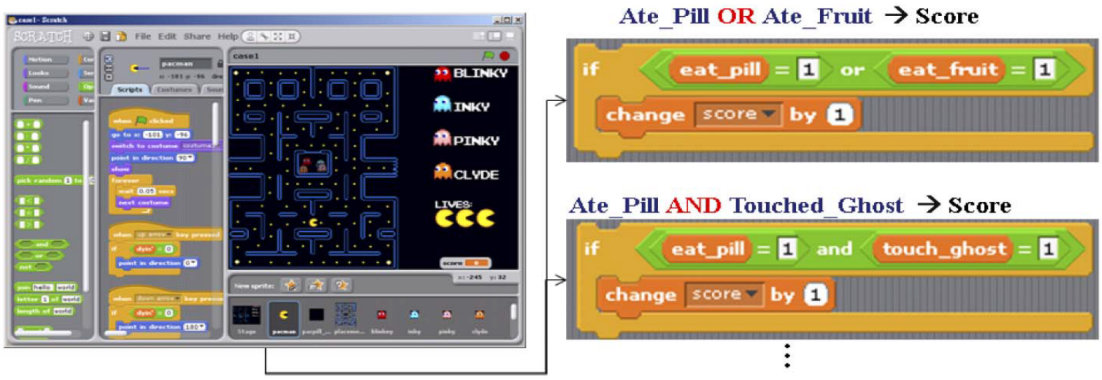

Figura 3 – Exemplo de regras do jogo expressas em expressões lógicas

[Fonte: reproduzido de Weng, Tseng e Lee (2010)]

2.2.4 Tarski's World

Outro software que foi criado com a intenção de facilitar o ensino de lógica é o Tarski World. Homenageando Tarski, o lógico, matemático e filósofo polonês, o software combina representação visual e expressões da lógica proposicional, checando tanto a sintaxe da expressão lógica, quanto sua veracidade baseada no mundo que está aberto (BARKER-PLUMMER JON BARWISE; LIU, 2014).

Utilizando de figuras geométricas, o Tarski permite que seja criado um mundo fictício, inserindo quantos objetos o usuário desejar e permitindo atribuir a estes objetos características de tamanho, nome (A, B, C, D, E etc.), forma (tetraedro, cubo ou dodecaedro) e posição relativa.

O mundo é representado em cima de uma malha quadriculada, que se assemelha a

um tabuleiro de xadrez. Essa estratégia foi adotada para facilitar a visualização da posição relativa entre os objetos, que é representada em 3D. Além disso, o Tarski também permite alterar o ângulo de visualização do mundo, mudando a posição relativa entre os objetos. Por exemplo, caso exista um objeto A que está na frente do objeto B, ao girar 180° no sentido horizontal, o objeto B passará a estar na frente do objeto A, logo, a sentença "A está na frente de B" (utilizando a linguagem do Tarski: $FrontOf(A, B)$) deixará de ser verdadeira.

Além de criar mundos, é possível criar sentenças envolvendo os nomes dos objetos ou variáveis, utilizando as características dos objetos. Vale ressaltar que as sentenças também podem ser criadas sem um mundo associado, entretanto, nessas condições elas são checadas apenas sintaticamente.

Ao criar sentenças com um mundo associado, é possível validá-las como verdadeiras ou falsas para aquele mundo. Por exemplo, se é criado um mundo onde só existem cubos e uma das sentenças diz que, para todo x, x é dodecaedro, essa sentença será falsa. Por outro lado, se existir outra sentença que diz que para todo x, x é cubo, esta será verdadeira.

Também há a possibilidade de alterar mundos já criados e verificar se as sentenças permanecem válidas ou não.

Com essas funcionalidades, o Tarski permite que o entendimento das expressões se dê tanto pela visualização do ambiente gráfico quanto pela visualização das sentenças, permitindo que o usuário avalie os impactos gerados pela modificação do mundo e da expressão lógica.

Seu ambiente é composto por 5 partes principais: a visualização do mundo selecionado; as sentenças que estão sendo avaliadas; uma região para a criação ou modificação das sentenças, que enumera todas as funções e operadores; a região de criação e modificação dos objetos, que apresenta todas as características dos objetos e, por último, a região dos controles, que apresenta opções importantes, como a de verificar uma sentença para o mundo ou de verificar todas as sentenças. Um exemplo de tela do Tarski's World pode ser visualizado na Figura 4.

Outra funcionalidade adicional do Tarski é permitir que o usuário questione a validade que o Tarski atribuiu a uma sentença, onde o software interage com o usuário, fazendo perguntas que mostram ao usuário o porquê de a sentença ter sido validada daquela forma.

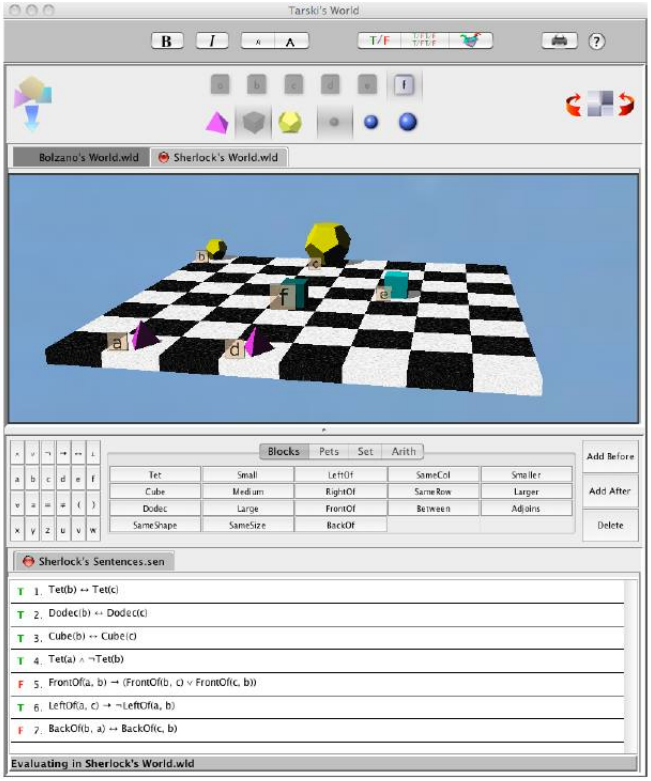

Figura 4 – Tela do Tarski's World

[**Fonte: reproduzido de Barker-Plummer Jon Barwise e Liu (2014)**]

O programa também permite criar, salvar, abrir e modificar mundos e sentenças, demonstrando uma grande flexibilidade para ser usado como forma de exercício para discentes da matéria de lógica, uma vez que o docente pode preparar qualquer situação e pedir para o aluno chegar em alguma solução alterando o mundo e/ou as sentenças.

Além disso, o software serve de complemento para um livro que contém diversos exercícios não apenas de lógica proposicional, mas de outros tipos de lógica. Para esses exercícios específicos, que são alterados todos os anos, o software oferece um sistema onde o docente pode seguir o passo-a-passo do exercício e enviar as respostas, para que as mesmas sejam corrigidas automaticamente e o resultado seja enviado para o professor responsável.

A avaliação da resposta do aluno pelo software vai além de *certo* ou *errado*. Também são apontados onde o aluno errou em cada exercício, sendo uma alternativa interessante para que os alunos melhorem o seu conhecimento.

Apesar de ter uma ferramenta de avaliação poderosa, ela é limitada aos exercícios propostos pelo grupo. Caso o professor tenha o interesse de criar novas questões, não será possível utilizar o sistema de análise para identificar os erros dos alunos, sendo necessário o acompanhamento individual do aluno nesta situação. Por fim, uma possível barreira na adoção da ferramenta Tarski's World é o fato dela ser paga.

2.3 Análise de softwares de apoio ao ensino de lógica

A principal funcionalidade que não pôde ser encontrada nos softwares explanados nas seções anteriores é a dinamicidade na representação e contextualização do problema. Apesar de alguns deles terem a representação do conteúdo em mais de uma forma (sentença e representações visuais), não é possível a criação de ambientes e contextos diferentes para o desenvolvimento de atividades.

Além disso, também pôde ser identificado a falta de apoio das ferramentas no que se diz respeito à interação entre os alunos e o professor, sendo presente apenas na ferramenta Tarski's World. Esta, apesar de dar a opção de enviar exercícios e avaliar as respostas, não dá a opção de definir uma data para que o exercício seja liberado para os alunos e uma data limite para o envio das respostas.

Por fim, uma outra limitação identificada diz respeito à compatibilidade das ferramentas com mais de um sistema operacional: algumas estão disponíveis apenas para Windows. Desta forma, justifica-se o objeto deste trabalho que consiste em desenvolver uma ferramenta multiplataforma de apoio ao ensino de lógica, que permita o professor elaborar listas de exercícios considerando diferentes domínios de aplicação.

3 Metodologia

Este capítulo descreve a metodologia empregada durante o desenvolvimento deste trabalho; em particular, no processo de criação da ferramenta *Teaching Logic*. A Seção 3.1 descreve os parâmetros que guiaram a revisão bibliográfica, e a Seção 3.2 descreve as etapas de desenvolvimento da ferramenta.

3.1 Realização da revisão bibliográfica

A revisão bibliográfica foi realizada nas bibliotecas Portal ACM[1], IEEE Xplore[2] e SpringerLink[3], considerando os seguintes termos de busca:

- *Teaching logic*
- *Teaching propositional logic*
- *Tool to teach logic*
- *Logic difficulties*
- *Difficulties of teaching propositional logic*
- *Model View View Model*

Para a seleção dos artigos, foram considerados todos os encontrados em Inglês, que contemplassem o ensino de lógica como tema principal. Com isso, foram selecionados inicialmente 21 artigos.

Em seguida. foi realizada a leitura do resumo, introdução e conclusão de cada artigo, escrevendo um resumo com as principais informações encontradas, além de descrever possíveis relações com o trabalho a ser desenvolvido. Após essa etapa, 12 dos 19 artigos foram selecionados. Estes 12 artigos foram lidos por completo e destes 7 foram considerados como diretamente relacionados ao escopo deste trabalho.

Além dos artigos, também foram selecionados um livro e uma tese de doutorado que têm como tema principal lógica (NOLT JOHN; ROHATYN, 2011; KIM, 1995), um livro que tem como tema principal metodologias de ensino (BRUNER R. R. OLIVER, 1966) e um livro que descreve uma arquitetura de software proposta pela Microsoft (CONERY SCOTT HANSELMAN, 2009).

[1] <http://dl.acm.org/>
[2] <http://ieeexplore.ieee.org/>
[3] <http://link.springer.com/>

3.2 Desenvolvimento do software

O software *Teaching Logic* foi desenvolvido a partir de um processo composto por seis etapas: levantamento de funcionalidades (Seção 3.2.1), desenvolvimento de um protótipo de telas (Seção 3.2.2), elaboração do diagrama de casos de uso (Seção 3.2.3), definição de padrões arquiteturais (Seção 3.2.4), desenvolvimento do diagrama de classes (Seção 3.2.5) e escolha de tecnologias (Seção 3.2.6).

3.2.1 Levantamento de funcionalidades

Inicialmente, foi realizado um *brainstorm* para discutir as principais funcionalidades da ferramenta e, com isso, definir o escopo que seria abordado durante esse trabalho e qual seria o escopo abordado em trabalhos futuros. Nesta etapa, decidiu-se que a ferramenta a ser desenvolvida deveria contemplar as seguintes funcionalidades:

- Professor:
  - Criação e manutenção livre de domínios;
  - Criação e manutenção de perguntas e listas de exercícios;
  - Avaliação das respostas dos alunos;
  - Criação e manutenção de turmas.
- Aluno:
  - Resolução das listas de exercícios propostas pelo professor;
  - Visualização de notas dos exercícios finalizados.

Outro ponto abordado a partir do *brainstorm*, foi o de desenvolver o software como uma aplicação web que tivesse um layout responsivo, o que viabiliza o seu acesso, tanto pelos os alunos quanto pelos os professores, de qualquer dispositivo que tenha conexão com a internet.

3.2.2 Protótipo de telas

A partir da organização das ideias, foi montado um protótipo de telas do software, utilizando o Balsamiq[4], para que fosse possível realizar uma análise crítica inicial. Por prover uma visão mais específica do que seria necessário para desenvolver a nova ferramenta, o protótipo de telas possibilitou o levantamento de questionamentos não percebidos anteriormente; por exemplo, informações necessárias para realizar a interação entre alunos e

[4] <https://balsamiq.com/>

professores. Além disso, ele proporcionou a reflexão a respeito da localização das informações no software. As principais telas do protótipo serão discutidas a seguir.

As páginas iniciais, tanto do professor (Figura 5) quanto do aluno (Figura 6), foram projetadas inicialmente para mostrar todo o conteúdo disponível para o usuário. A partir da análise dos protótipos das telas iniciais, concluiu-se que algumas informações deveriam ser redistribuídas em outras telas.

Com relação à tela do professor (Figura 5), a região com as listas de exercícios atribuídas às turmas foi passada para a tela de configuração de turma. Já com relação à tela inicial do aluno (Figura 6), as informações das listas de exercícios passaram a ficar ocultas, só sendo mostradas quando o usuário solicita a visualização das listas para uma determinada turma.

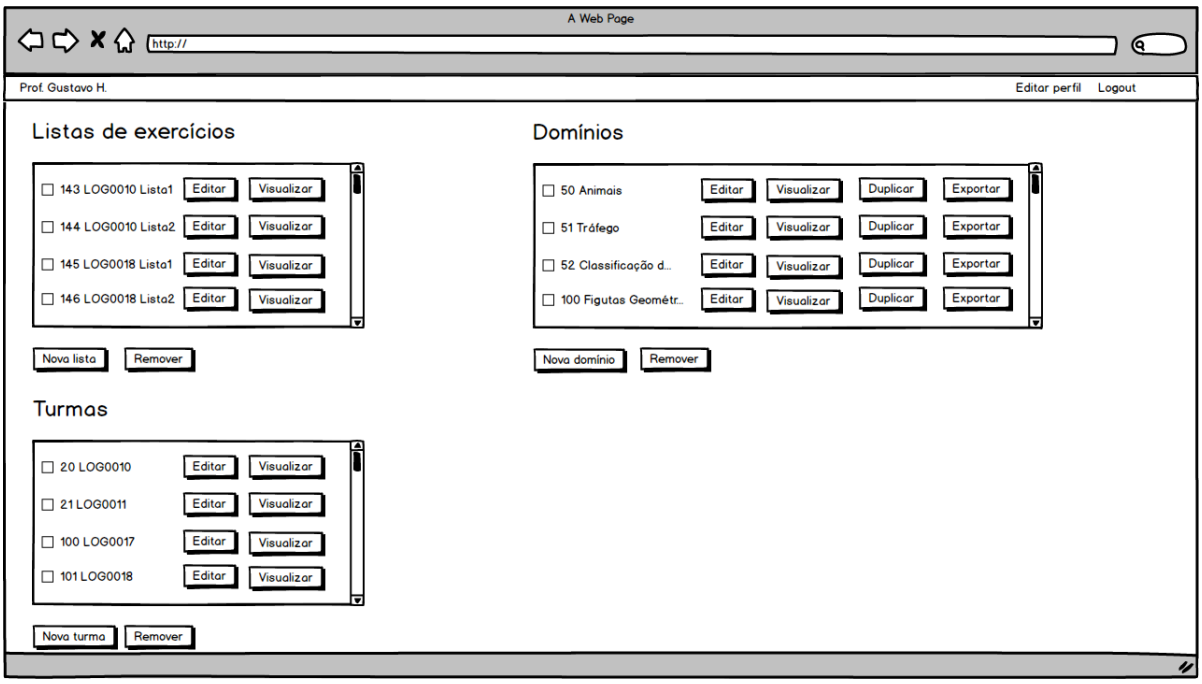

Figura 5 – Protótipo de tela de inicial do professor

Com relação ao protótipo de tela de configuração do domínio (Figura 7), inicialmente foi pensado em mostrar todas as informações em tabelas com barras de rolagem, para que fosse possível visualizar quantos itens o domínio possuísse para aquele atributo. Essa abordagem iria dificultar a adaptação do *layout* para dispositivos móveis, além de poluir visualmente a tela. Com o intuito de minimizar esses problemas, a região que define partições de um tipo abstrato foi movida para dentro da região que lista os tipos abstratos. Além disso, foi utilizada uma estrutura de tabela simples, para que o *layout* se adaptasse mais facilmente a diversos dispositivos.

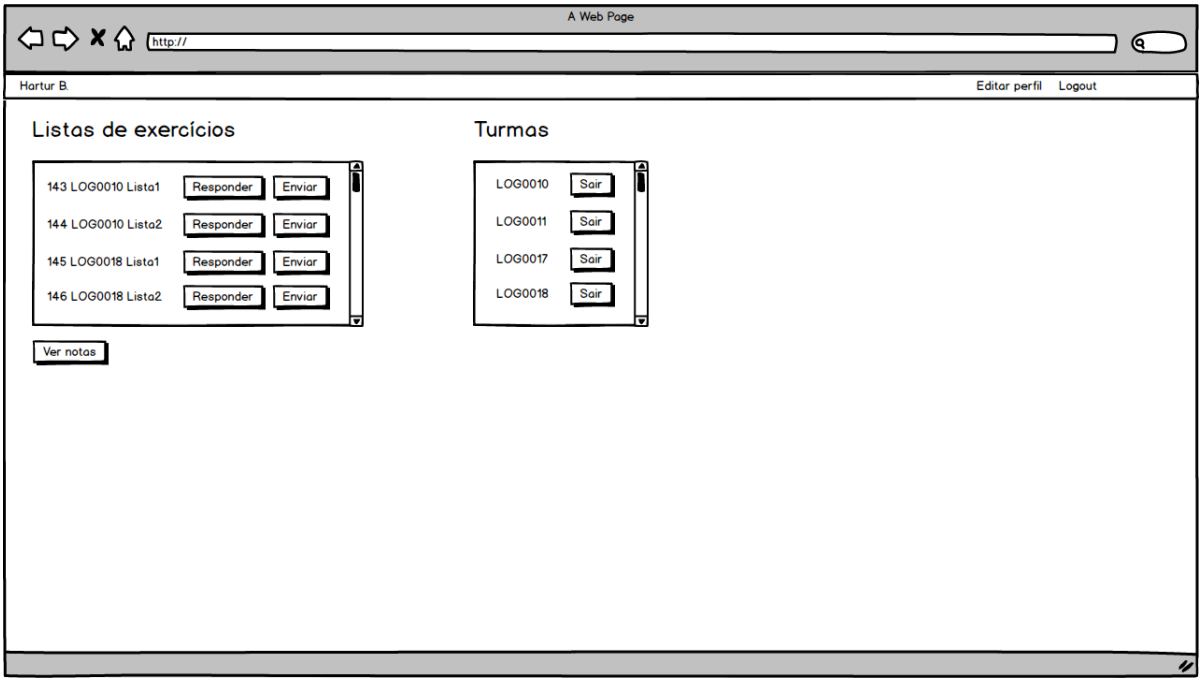

Figura 6 – Protótipo de tela inicial do aluno

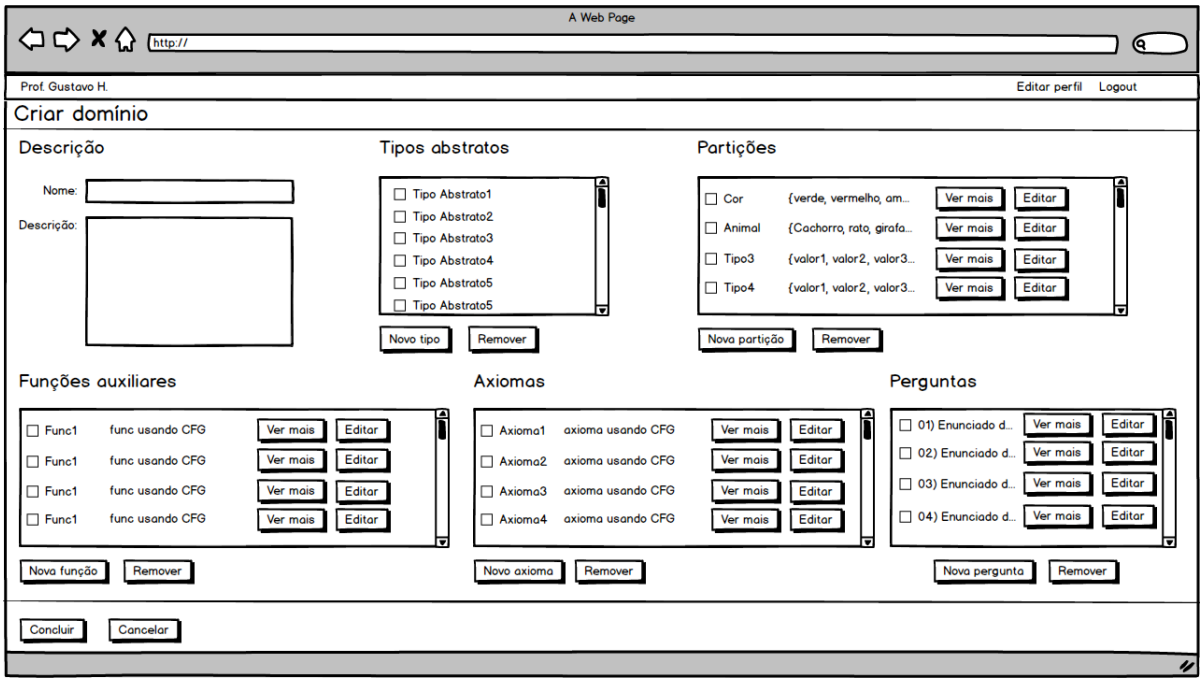

Figura 7 – Protótipo de tela de novo domínio

O desenvolvimento do protótipo de tela em que o aluno responde uma pergunta (Figura 8) possibilitou a reflexão de como seriam detalhadas as informações do domínio

(Figura 9), permitindo que o aluno as consulte quando necessário. A abordagem utilizada no protótipo não permite a visualização das informações do domínio enquanto a questão é respondida. Para resolver esse problema, foi adicionado um link à tela para que fosse possível expandir ou comprimir as informações do domínio.

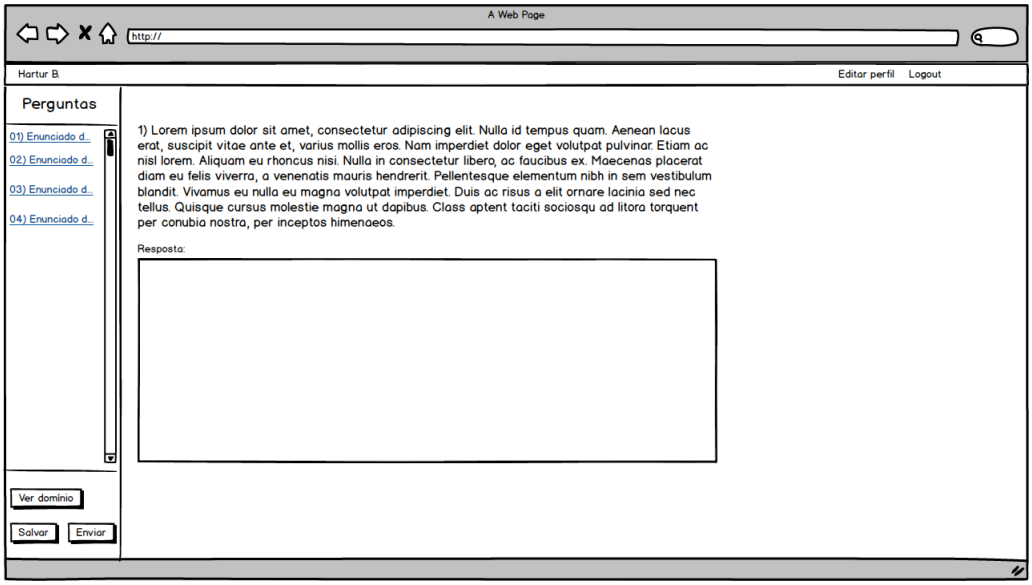

Figura 8 – Protótipo de tela de resposta de pergunta

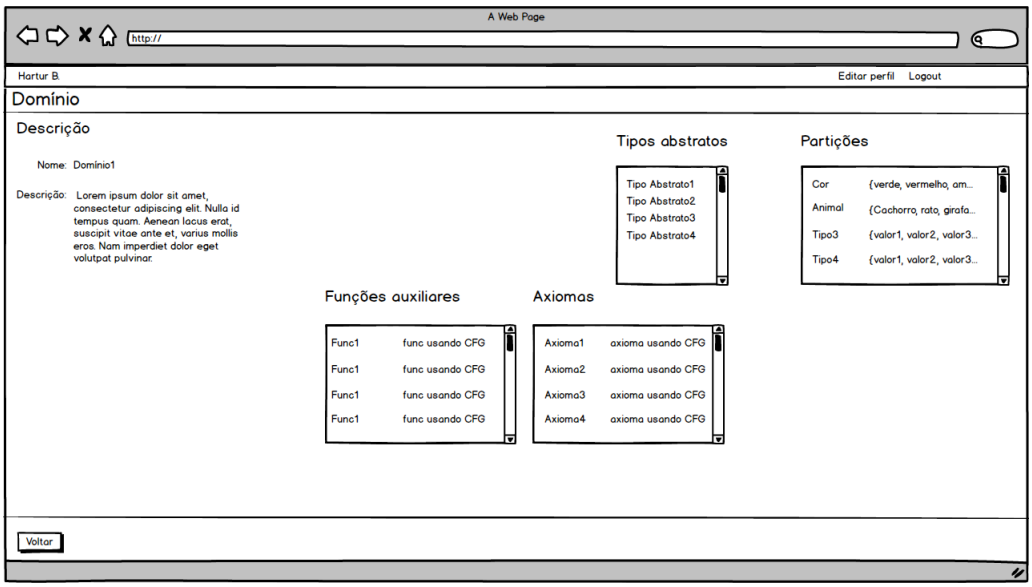

Figura 9 – Protótipo de tela de visualização de detalhes do domínio

Outras telas desenvolvidas durante a fase de prototipagem podem ser encontradas no Apêndice B. Mais detalhes de como ficaram as telas da aplicação serão discutidos no Capítulo 4.

3.2.3 Diagrama de casos de uso

Após o desenvolvimento dos protótipos de tela, criou-se um diagrama de casos de uso (Figura 10) para capturar as principais funcionalidades do sistema. Nele, pode ser observado a funcionalidade de cada um dos atores (professor e estudante).

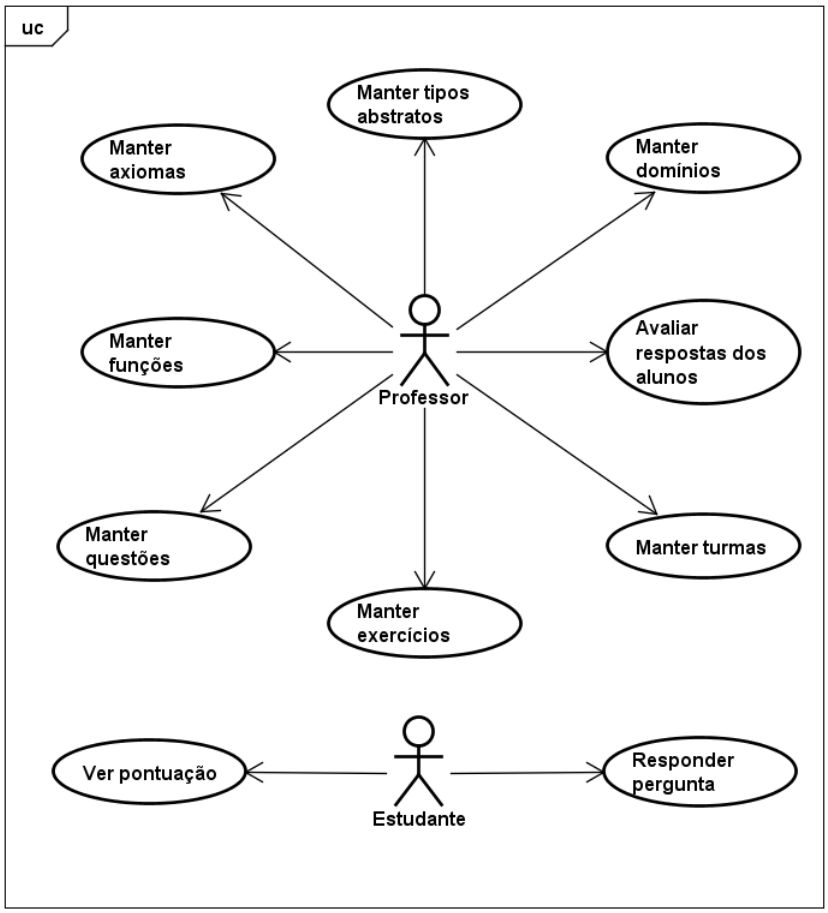

Figura 10 – Diagrama de casos de uso

O professor pode manter as informações dos domínios criados pelo mesmo. Um domínio, por sua vez, possui como características uma coleção de tipos abstratos, axiomas, funções, questões e exercícios. Ou seja, para manter um domínio é preciso manter todas essas características. Uma outra relação que pode ser destacada é a entre exercícios e questões. Um exercício deve conter uma coleção de questões, as quais são limitadas às questões definidas para o domínio em foco.

O professor também deve manter turmas, que são representadas por uma coleção de estudantes e de listas de exercícios designados à turma. Além disso, a partir do momento em que a data limite do exercício é atingida, o professor pode avaliar as respostas enviadas pelos alunos. Além de poder responder às perguntas dos exercícios designados às turmas que participa, o estudante também pode ver sua pontuação final após a avaliação da

resposta.

O diagrama de casos de uso facilitou a visualização de quais funcionalidades deveriam ser implementadas, além de proporcionar um entendimento melhor da separação dos papéis do professor e do aluno.

3.2.4 Definição de padrões arquiteturais

Após o desenvolvimento do diagrama de casos de uso, foram definidos que padrões arquiteturais seriam observados durante o desenvolvimento da aplicação. Por se tratar de uma aplicação web com possíveis expansões futuras, optou-se por utilizar um padrão arquitetural semelhante ao sugerido pela Microsoft em (CONERY SCOTT HANSELMAN, 2009) (Figura 11). Este padrão foi desenvolvido baseado em arquiteturas como MVVM (*Model View View Model*) e MVC (*Model View Controller*) (SYROMIATNIKOV; WEYNS, 2014). As camadas consideradas no desenvolvimento da ferramenta, podem ser vistas na Figura 12.

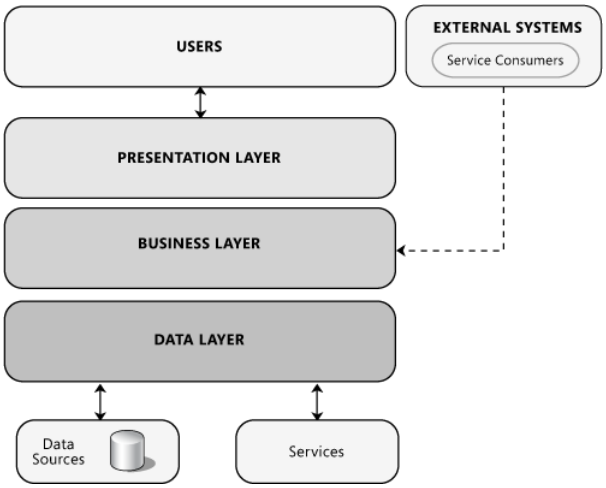

Figura 11 – Padrão arquitetural sugerido pela Microsoft

[Fonte: reproduzido de Conery Scott Hanselman (2009)]

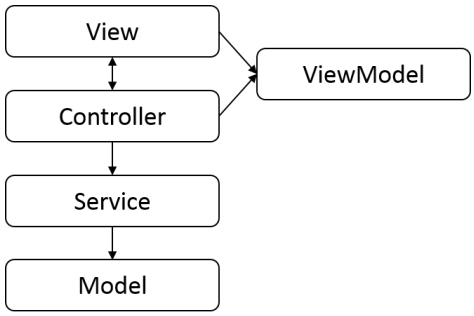

Figura 12 – Padrão arquitetural utilizado na aplicação

As responsabilidades atribuídas para cada camada, de acordo com a implementação, são descritas a seguir.

- *Model:* responsável por definir os modelos de domínio do software, que representa o estado real dos dados. Essa camada é desenvolvida utilizado orientação à objetos. Exemplos de modelos de domínio são: *Student*, *Professor*, *Exercise*, *Question* entre outras.

- *Service:* a camada de serviços é responsável por intermediar a comunicação entre as camadas *Model* e a *Controller*. Sua função é de realizar as operações necessárias ao correto funcionamento das operações do *Controller*, aplicando as regras de negócio. Um exemplo de operação seria o fato de que, ao atualizar uma questão de uma lista de exercício que já foi atribuída aos alunos, será necessário apagar a resposta dada anteriormente pelo aluno, para que ele possa preencher novamente com sua resposta de acordo com as novas condições da questão. Para realizar tal operação, o *Controller* só precisará solicitar que o serviço atualize os dados adequadamente.

- *View Model:* essa camada é responsável por realizar uma abstração das telas, ou seja, os objetos só terão as informações que são realmente necessárias para a construção da *View*. Essa camada facilita o manuseamento das entidades pelo controlador. Um exemplo de aplicação dessa camada é: para uma tela onde serão listados todas as turmas pertencentes ao professor, não é necessário ter a informação de quais alunos estão contidos nessa turma. Então, deverá ser criado um objeto na camada *View Model* para representar apenas as informações que seriam mostradas na tela, ou seja, uma lista de nomes de turmas associadas ao professor.

- *Controller:* realiza a integração entre as camadas *Service* e *View*, transformando os objetos contidos na camada *Model* em objetos da camada *View Model*, para que sejam repassados apenas os dados necessários para a construção da tela. Um exemplo de utilização dessa camada é: ao selecionar o link de listagem de turmas, o controlador deverá solicitar à camada de serviços a lista de turmas associadas ao professor, para então transformar essa lista de turmas, representada por objetos da camada *Model*, em objetos da camada *View Model*. . Em seguida, esta informação transformada é passada para a tela de listagem de turmas, localizada na camada *View*.

- *View:* responsável por armazenar todas as telas que serão mostradas para os usuários.

3.2.5 Diagrama de classes

Definidos os padrões arquiteturais que seriam adotados, diagramas de classes foram desenvolvidos. Aqui, apresenta-se uma visão geral do agrupamento destas classes a partir

de um diagrama de pacotes (Figura 13). Informações mais detalhadas das classes contidas em cada pacote podem ser encontradas nos diagramas de classes detalhados no Apêndice A.

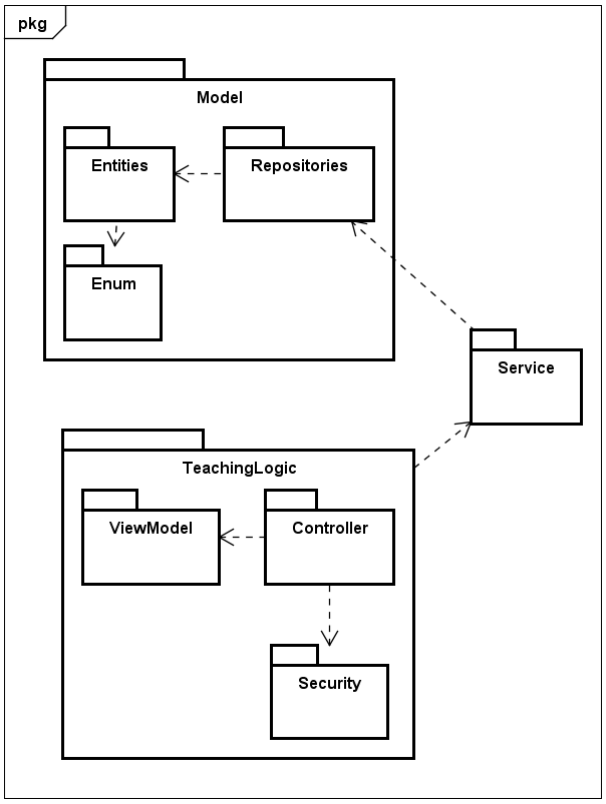

Figura 13 – Diagrama de pacotes

A partir da análise do diagrama da Figura 13, podem ser observadas as camadas bem definidas conforme descritas na Seção 3.2.4. A camada de serviço, definida pelo pacote *Service*, é responsável por manusear as entidades contidas na camada de modelo (*Model*) e repassá-las para os controladores (contidos no pacote *Controller*). Na camada de modelo, que está contida no pacote *Model*, podem ser encontrados os pacotes responsáveis por representar as entidades (*Entities*), realizar as operações com o banco de dados (*Repositories*) e representar valores fixos utilizados na aplicação (*Enum*). Por fim, no pacote *TeachingLogic*, podem ser encontrados os pacotes responsáveis por representar os objetos que serão mostrados nas telas (*ViewModel*), o pacote de segurança (*Security*), que é responsável por gerir o acesso dos usuários, e o pacote dos controladores (*Controller*), que manipulam as entidades entregues pelo serviço para transformá-las em objetos do pacote *ViewModel*, podendo assim, serem repassados para as telas.

3.2.6 Escolha de tecnologias

Para o desenvolvimento da ferramenta, foi utilizada a linguagem de programação C#[5] com a plataforma ASP.NET[6]. A plataforma ASP.NET foi desenvolvida pela Microsoft para possibilitar o desenvolvimento de páginas web dinâmicas utilizando qualquer linguagem integrada com o .NET Framework, a exemplo da linguagem C#.

Para a criação do projeto e escrita do código, foi utilizada a IDE Visual Studio 2013 Community[7], que é uma ferramenta fornecida pela Microsoft para desenvolvimento de aplicações não comerciais que utilizam suas tecnologias. Além disso, por ser uma ferramenta utilizada para desenvolver códigos, ela também tem integração direta com git[8], facilitando o processo de controle de versão.

Foi utilizado o controle de versão do código via git para garantir que todas as alterações realizadas no código fossem rastreadas e que fosse possível voltar a um estado anterior, caso acontecesse algum problema durante o desenvolvimento da aplicação.

O armazenamento das informações geradas pelo sistema é feito em um banco de dados SQL Server[9], desenvolvido também pela Microsoft. A comunicação do software com o banco se deu por meio da utilização do Entity Framework[10], que é um framework de mapeamento objeto-relacional utilizado para facilitar a criação de consultas ao banco de dados.

Outra tecnologia que foi utilizada no desenvolvimento da ferramenta foi o Razor[11], que é uma sintaxe de programação definida para criação de páginas dinâmicas pela plataforma ASP.NET. Além do Razor, também foram utilizados HTML e Bootstrap[12] para desenvolver o layout das páginas, e Javascript e Ajax para implementar comportamentos dinâmicos.

Para auxiliar o manuseio dos objetos, foi utilizado o Auto Mapper[13], que é um conjunto de convenções para mapeamento objeto-objeto. Ele foi útil para configurar o mapeamento de objetos contidos na camada *Model* para objetos da camada *View Model*, conforme descrito na Seção 3.2.4.

[5] <https://msdn.microsoft.com/library/67ef8sbd.aspx>
[6] <https://www.asp.net/>
[7] <https://www.visualstudio.com/>
[8] <https://git-scm.com/>
[9] <http://www.microsoft.com/SQL>
[10] <https://www.asp.net/entity-framework>
[11] <https://msdn.microsoft.com/library/gg675215.aspx>
[12] <http://getbootstrap.com/>
[13] <http://automapper.org>

4 Resultados

Este capítulo detalha as funcionalidades oferecidas pela ferramenta *Teaching Logic*: um ambiente web para auxiliar o ensino de lógica. A Seção 4.1 apresenta uma visão geral da ferramenta, enquanto que as Seções 4.2 e 4.3 descrevem as funcionalidades associadas ao perfil de professor e aluno, respectivamente. Por fim, a Seção 4.4 compara a ferramenta desenvolvida com as soluções previamente discutidas no Capítulo 2.

4.1 Uma visão geral da ferramenta Teaching Logic

As informações obtidas com a revisão bibliográfica proporcionaram um melhor entendimento das dificuldades encontradas pelos alunos ao estudar lógica, e o conhecimento dos trabalhos já desenvolvidos com o objetivo de superar essas dificuldades. A partir da reflexão desses pontos, foi pensado no desenvolvimento de uma nova ferramenta para auxiliar o ensino de lógica, o *Teaching Logic*.

Essa tarefa é realizada promovendo uma interação entre o professor e o aluno, tornando possível a criação de domínios, que são representações formais de lógica contextualizada em situações cotidianas, além da criação de listas de exercício para serem designadas a um grupo de alunos com o intuito de avaliar o aprendizado.

A ferramenta permite que o professor faça um melhor planejamento das atividades, tendo em vista que é possível escolher a data na qual a lista ficará disponível para os alunos. Além disso, o professor também pode avaliar as repostas dadas assim que a data limite do exercício for alcançada.

Por meio da criações de diversos domínios, o professor terá a liberdade de criar ambientes com vários níveis de dificuldades, podendo assim, analisar em quais contextualizações a turma apresenta maior dificuldade. Além disso, para um mesmo domínio, podem ser criadas várias questões e listas de exercício. Cada lista de exercício tem uma coleção de questões com a sua respectiva pontuação máxima, e cada questão tem um tipo de pergunta e um tipo de resposta. Com as informações dos tipos de pergunta e resposta, o professor pode realizar uma análise crítica a respeito das dificuldades que os alunos estão enfrentando de acordo com a pontuação que eles receberam nas respostas.

A partir da análise do desenvolvimento da turma, cabe ao professor decidir como variar o nível de abstração do domínio e de dificuldade das perguntas por lista de exercício, e promover um aumento na quantidade de questões de um tipo determinado para que os alunos treinem e se tornem hábeis a resolver esse tipo de questão.

A ferramenta define dois tipos de usuários, um que representa o professor e outro

que representa o aluno. Para ter acesso às funcionalidades da aplicação, é necessário ter feito o cadastro previamente para que se torne possível realizar o login (Figura 14). As funcionalidades de cada um desses tipos de usuário são exploradas nas Seções 4.2 e 4.3.

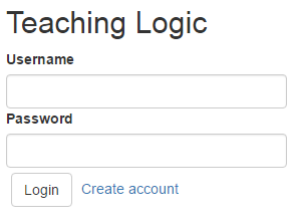

Figura 14 – Tela de login

4.2 Funcionalidades do perfil de professor

O professor tem acesso a maior parte das funcionalidades do software, tendo como objetivo criar e configurar domínios, criando listas de exercício para que sejam atribuídas às turmas. Ao efetuar o login com esse tipo de usuário, a primeira tela apresentada é a do painel do professor (Figura 15).

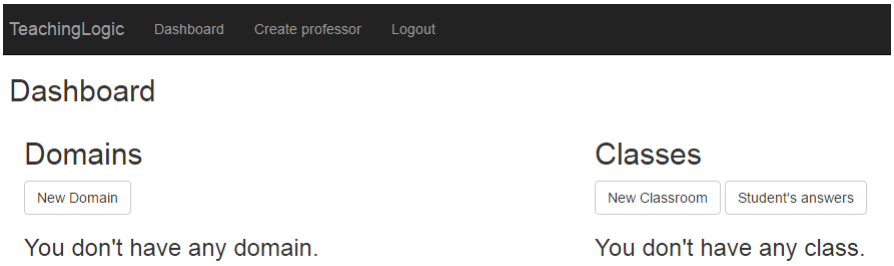

Figura 15 – Página inicial da conta do tipo professor

O professor tem total liberdade para criar, configurar, editar e deletar domínios e turmas. Com isso, é possível que o professor se organize da maneira que considerar mais adequada para a realização das suas atividades de ensino.

O ideal é que o domínio represente uma situação comum no dia-a-dia dos alunos para que facilite o entendimento da matéria. Entretanto, isso não impede que o professor crie domínios mais complexos, com representações abstratas, para que ele possa avaliar a capacidade dos seus alunos.

As turmas são compostas por um conjunto de alunos, o que permite o professor delegar listas de exercícios adequadas para o grupo em questão. Para criar uma nova turma, basta que o professor clique no botão "*New Classroom*", que pode ser encontrado

na Figura 15. Então, é apresentada uma tela para que seja preenchido o nome-código da nova turma, conforme a Figura 16. Para finalizar o processo, basta clicar no botão "Create" da tela.

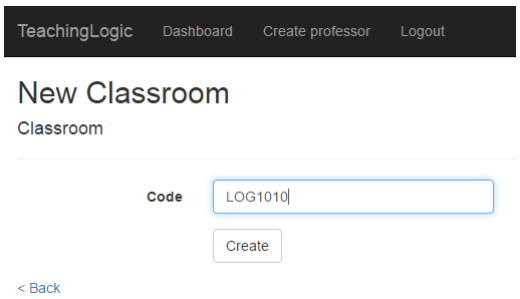

Figura 16 – Tela de criação de classe.

Para criar um novo domínio, o professor deve clicar no botão "*New Domain*", que pode ser encontrado na Figura 15. Então, é apresentada uma tela para que seja preenchido o nome e a descrição desse novo domínio, conforme a Figura 17.

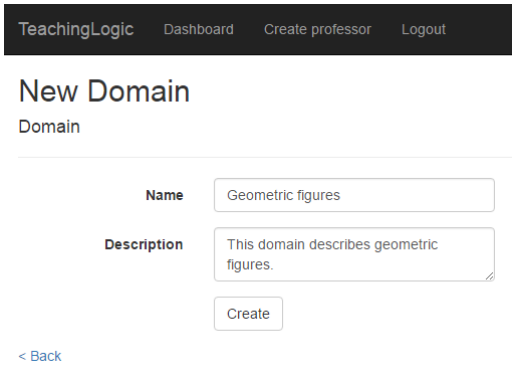

Figura 17 – Tela de criação de domínio.

Ao finalizar a criação de turma e domínio, eles serão listados na tela inicial da conta do tipo professor, com as opções para configurar (link "*Configure*"), editar (link "*Edit*") ou deletar (link "*Delete*") para cada uma dessas estruturas, conforme a Figura 18.

Caso seja selecionada a opção de editar, são exibidas as telas de edição de acordo com o tipo de dado que foi selecionado.

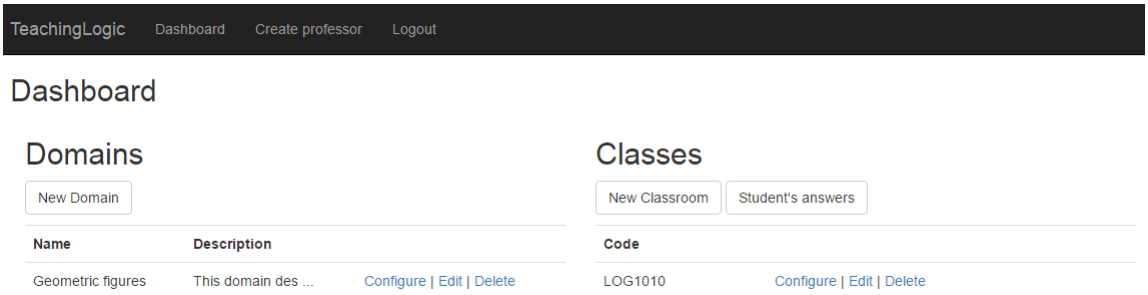

Figura 18 – Página inicial da conta do tipo professor

Para iniciar a configuração do domínio, deve-se selecionar a opção "*Configure*" da linha do domínio que se deseja configurar. Essa opção pode ser visualizada na Figura 15. Ao selecionar a opção de configuração de domínio, uma tela semelhante à da Figura 19 é apresentada, mostrando o nome e a descrição do domínio que será configurado, e as opções de configurações.

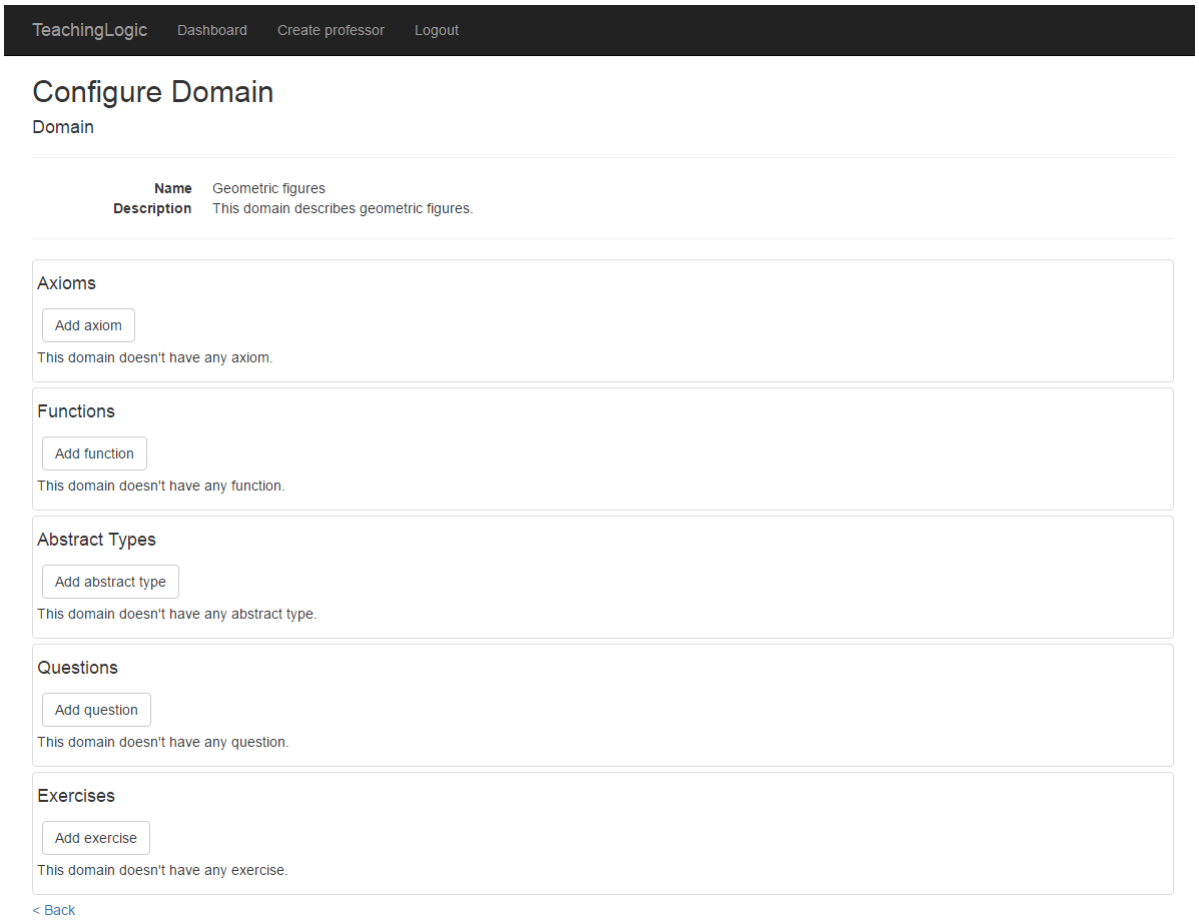

Figura 19 – Página de configuração de domínio

Para incluir um novo tipo abstrato ao domínio, clica-se no botão "*Add abstract*

type", que pode ser visto na Figura 19. Ao clicar, é aberta uma tela conforme a Figura 20. Nessa tela, são preenchidos os campos de nome do tipo abstrato e quais são seus possíveis valores. Após o preenchimento, deve-se clicar no botão "*Create*" para finalizar a adição de um novo tipo abstrato ao domínio.

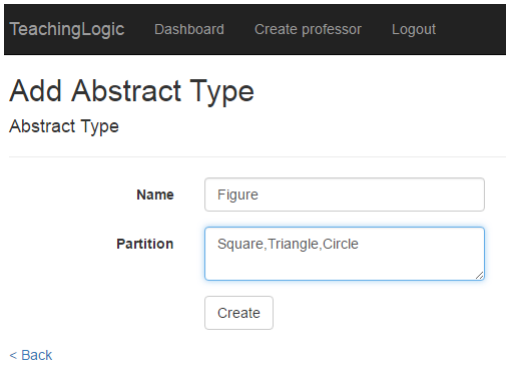

Figura 20 – Tela de criação de tipo abstrato.

Para incluir um novo axioma ao domínio, clica-se no botão "*Add axiom*", que pode ser visto na Figura 19. Ao clicar, é aberta uma tela conforme Figura 21. Nessa tela, são preenchidos os campos de nome do axioma e predicado que o descreve.

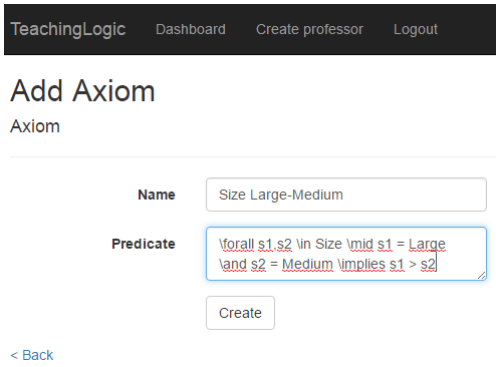

Figura 21 – Tela de criação de axioma.

Para incluir uma nova função ao domínio, deve-se clicar no botão "*Add function*", que pode ser visto na Figura 19. Ao clicar, é aberta uma tela conforme a Figura 22. Nessa tela, são preenchidos os campos de nome da função, seus parâmetros de entrada e seu retorno.

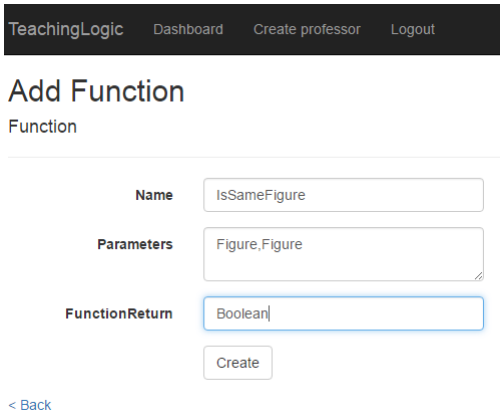

Figura 22 – Tela de criação de função.

Para incluir uma nova questão associada ao domínio selecionado, deve-se clicar no botão "*Add question*", que pode ser visto na Figura 19. Ao clicar, é aberta uma tela semelhante à da Figura 23. Nessa tela, descreve-se o enunciado da pergunta, seleciona-se o tipo de pergunta, e o tipo de resposta. As opções de tipo de resposta são: múltipla escolha, aberta e verdadeiro e falso. Dependendo do tipo de resposta, deve-se ainda preencher as opções de resposta que os alunos terão (Figuras 23 e 24).

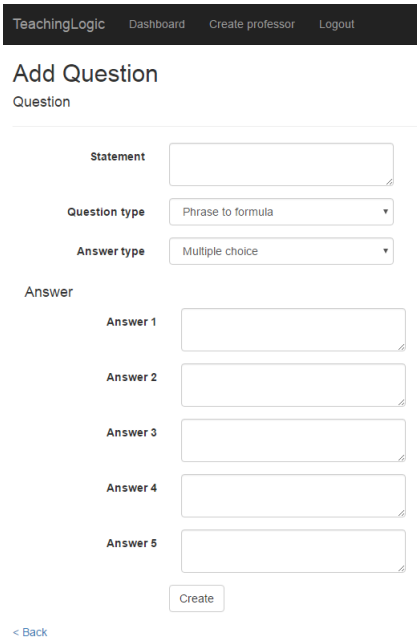

Figura 23 – Tela de criação de pergunta múltipla escolha

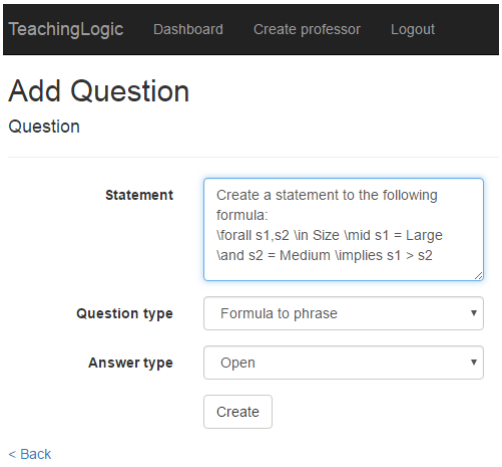

Figura 24 – Tela de criação de pergunta aberta

Após a criação de questões, pode-se elaborar listas de exercícios, considerando as questões previamente criadas. Para incluir uma nova lista de exercícios ao domínio, deve-se

clicar no botão "*Add exercise*", que pode ser visto na Figura 19. Ao clicar, é aberta uma tela conforme a Figura 25. Nessa tela, são preenchidos os campos de nome do exercício, data de início (data que a lista será disponibilizada para os alunos), data de fim (data final para os alunos submeterem as respostas), além de selecionar as questões que farão parte da lista. Para cada questão incluída na lista, o professor deve informar a sua pontuação máxima. Ao clicar em "*Details*", o enunciado da questão seleciona é exibido por completo (Figura 26).

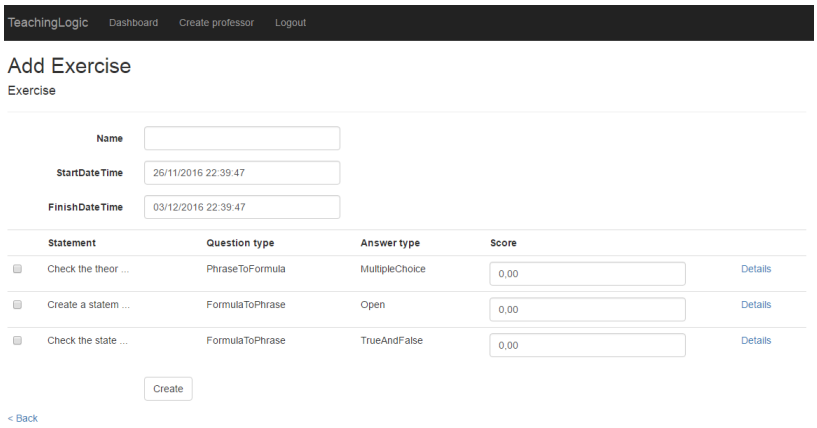

Figura 25 – Tela de criação de lista de exercício

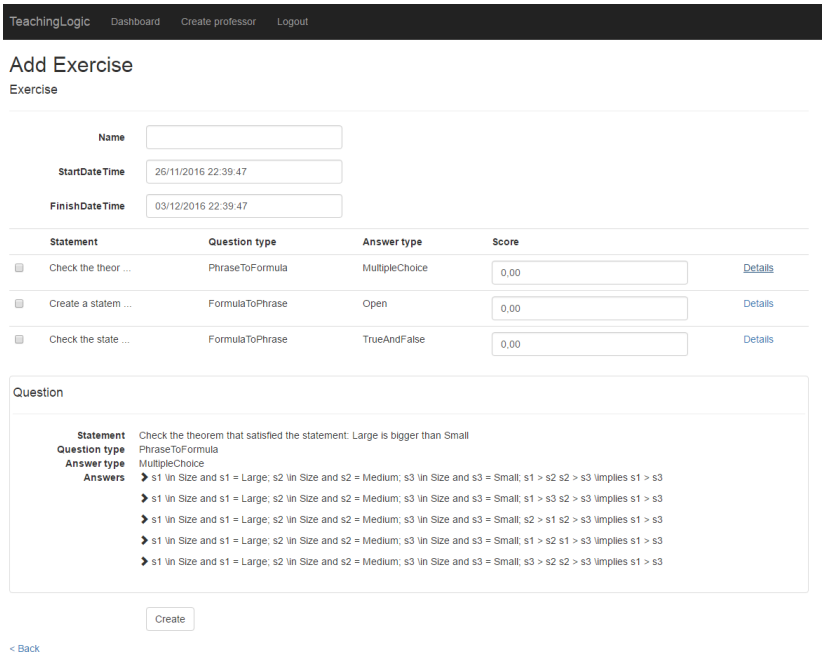

Figura 26 – Tela de criação de lista de exercício com detalhamento de questão

Cada um dos itens citados anteriormente (tipos abstratos, axiomas, funções, questões e listas de exercícios) pode ser alterado clicando na opção "*Edit*", que fica do lado

direito de cada uma das estruturas listadas. Essa opção pode ser visualizada na Figura 27, que mostra a tela de domínio preenchida.

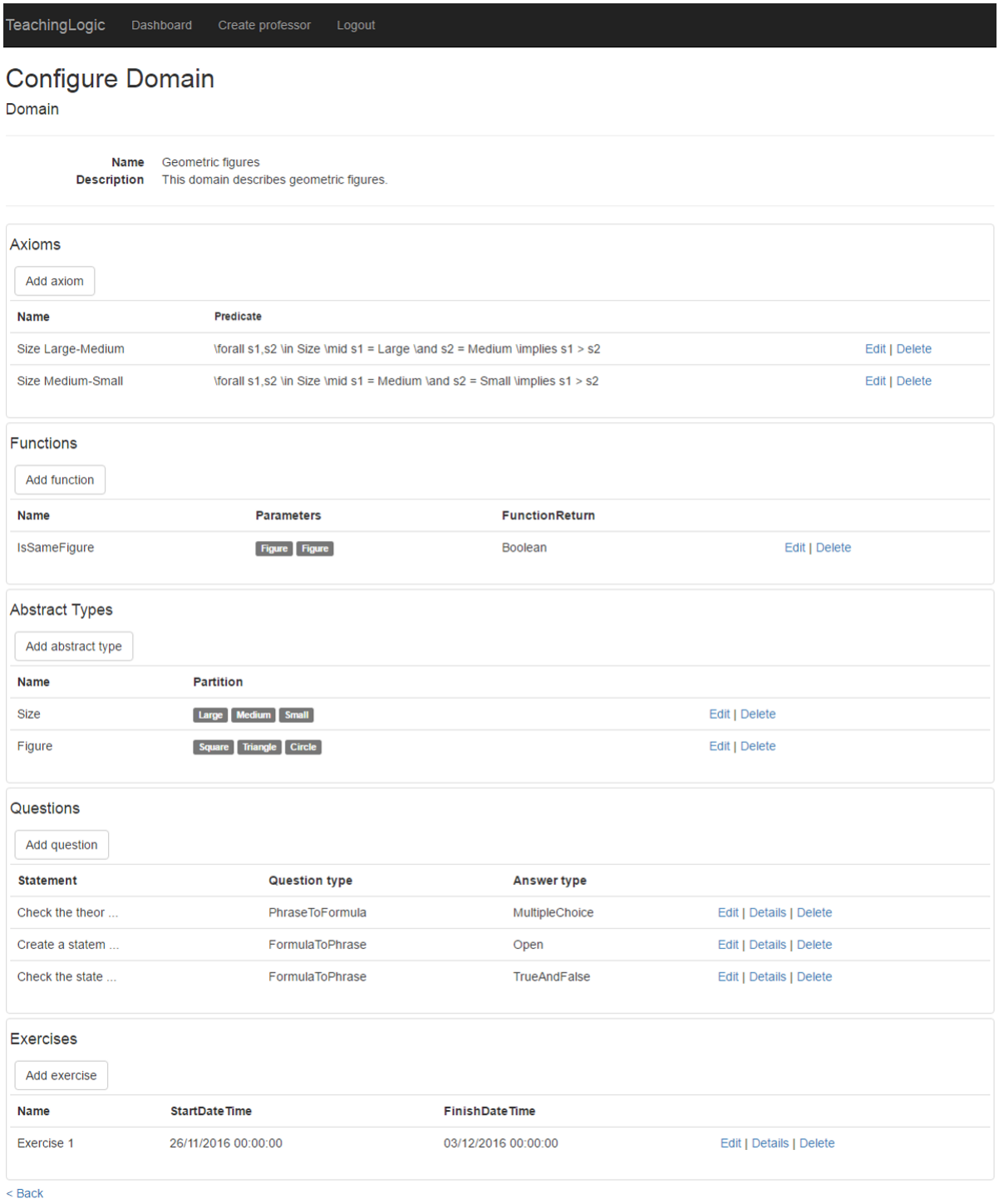

Figura 27 – Tela de configuração de domínio com dados configurados

Após criar listas de exercícios, o professor precisa associar estas às turmas desejadas. Para realizar a configuração de uma turma, deve-se clicar no link "*Configure*" localizado à

direita do nome da turma, conforme a Figura 18. Ao clicar nessa opção, é apresentada uma tela semelhante à da Figura 28.

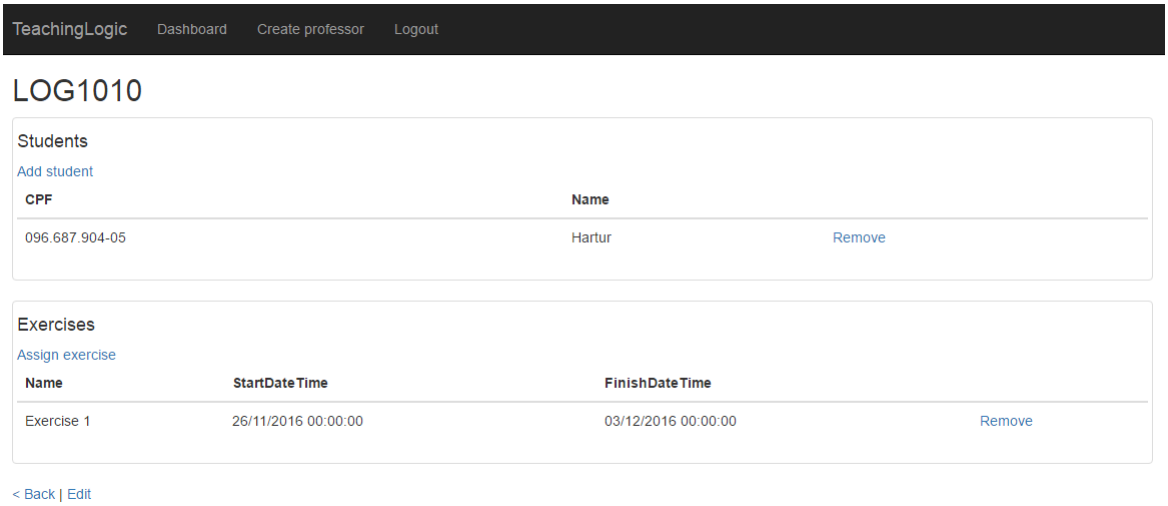

Figura 28 – Tela de configuração de turma

Para adicionar um aluno à turma, deve-se clicar no link "*Add student*", que pode ser visto na tela exibida na Figura 28. Com isso, é mostrada uma tela (Figura 29) onde é possível escolher e adicionar os alunos previamente cadastrados.

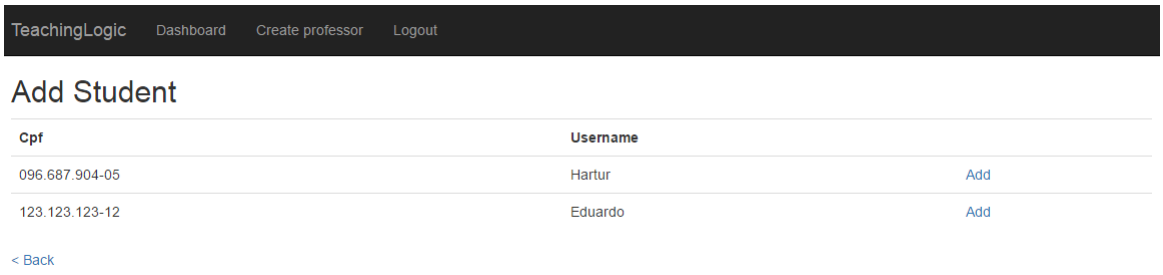

Figura 29 – Tela de alocação de aluno na turma

Para adicionar uma lista de exercício à turma, deve-se clicar no link "*Add exercise*", que pode ser encontrado na tela exibida na Figura 28. Com isso, é mostrada a tela da Figura 30. Nela, são listados os domínios que o professor possui, para que ele possa expandir as listas de exercícios contidas em cada um dos domínios, como pode ser visto na Figura 31, e selecione uma das listas. Ao escolher a lista, o professor deverá clicar no link "*Add*" para adicioná-la à turma.

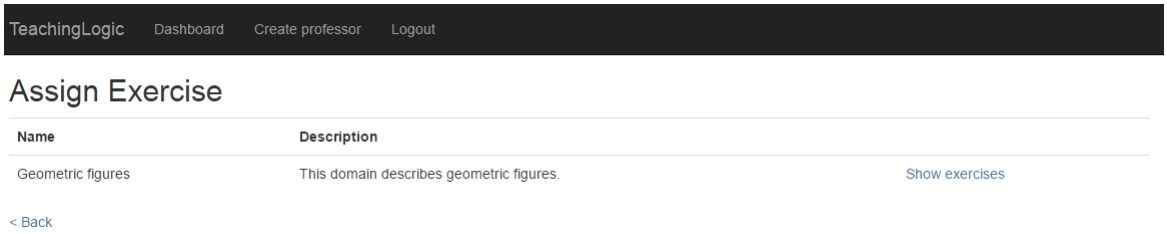

Figura 30 – Tela inicial de alocação de lista de exercício na turma

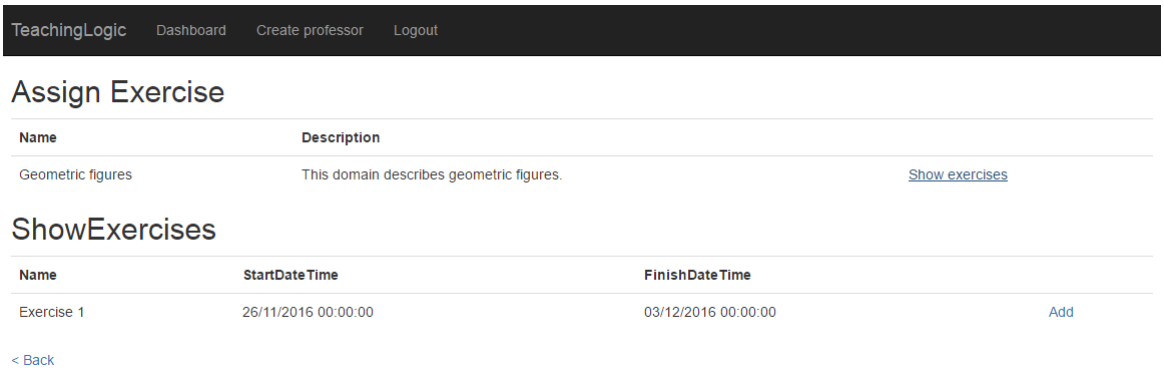

Figura 31 – Tela de alocação de lista de exercício

Para acessar a tela de visualização das respostas enviadas pelos alunos, o professor deve selecionar a opção "*Student's answers*", que pode ser encontrada na tela inicial para contas do tipo professor (como o exemplo da Figura 15). Ao acessar essa opção, é exibida uma tela (Figura 32) com a lista das respostas dadas pelos alunos, com a informação da turma, nome do exercício, 15 primeiros caracteres do enunciado, nome do aluno, pontuação máxima da questão e um campo para que seja preenchido com a pontuação que o aluno irá receber. Para visualizar a resposta dada pelo aluno e os destalhes da questão, o professor deve clicar na opção "*Show answer*", que irá expandir uma região mostrando o enunciado completo, as possíveis respostas para questões do tipo múltipla escolha e verdadeiro e falso, bem como a resposta dada pelo aluno.

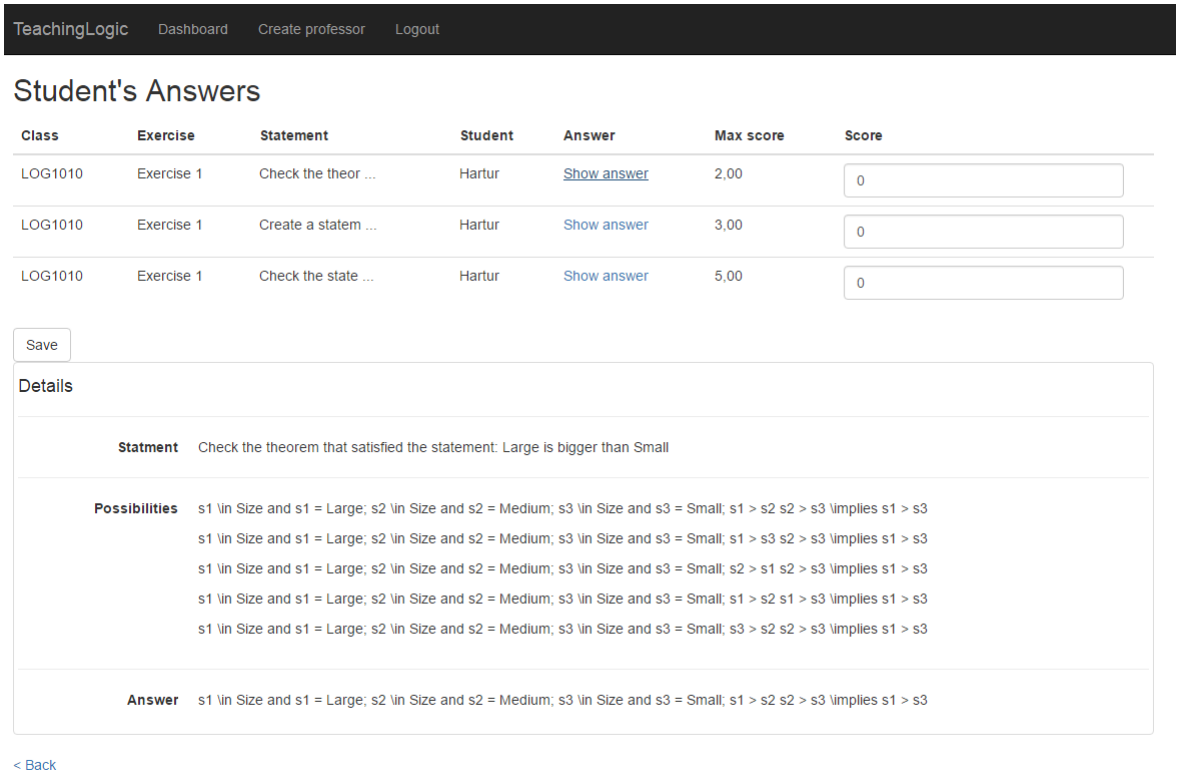

Figura 32 – Tela de visualização de resposta do aluno

4.3 Funcionalidades do perfil de aluno

Ao realizar o login com uma conta do tipo aluno, é exibida uma tela semelhante à da Figura 33. A partir dessa tela que o aluno pode visualizar em quais turmas ele está inscrito.

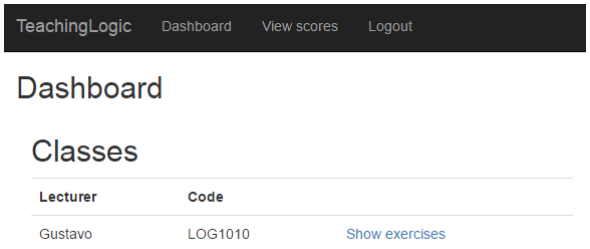

Figura 33 – Tela inicial da conta do tipo aluno

Para cada turma que o aluno está inscrito, ele pode visualizar quais as listas de exercícios disponíveis para serem respondidas. Para realizar tal ação, ele deve clicar no link "*Show exercises*", que irá expandir uma área com as informações. Tal área pode ser visualizada na Figura 34.

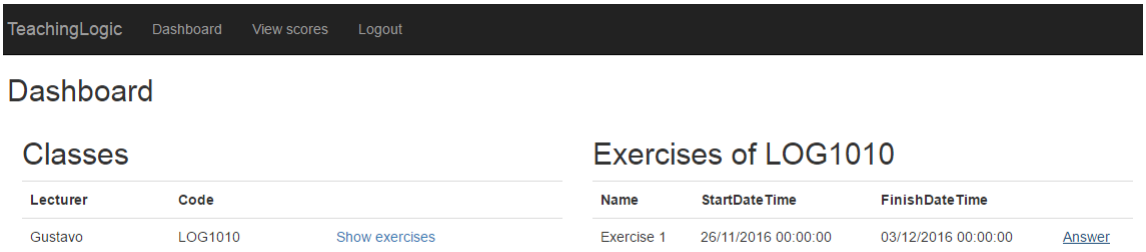

Figura 34 – Tela com exercícios da turma

Após expandir a turma para visualizar seus exercícios, o aluno pode selecionar um dos exercícios para visualizar suas perguntas clicando no link "*Answer*", que aparece ao lado direito de cada um dos exercícios disponíveis. A tela aberta após clicar no link pode ser visualizada na Figura 35.

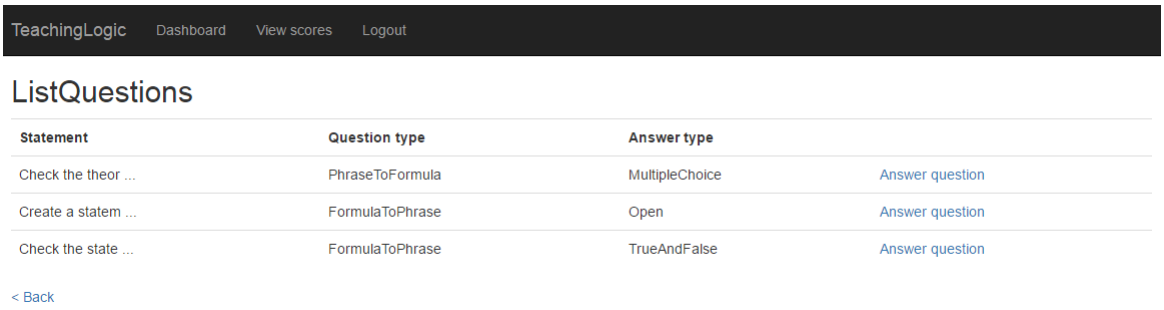

Figura 35 – Tela de listagem de questões da lista de exercícios

As questões são listadas mostrando as 15 primeiras letras do enunciado, assim como o tipo de pergunta e de resposta. Para responder às perguntas, o aluno precisará visualizar as informações a respeito do domínio definidas pelo professor. Para visualizar tais informações, o aluno deve selecionar a opção "*Show domain details*". O resultado dessa ação pode ser visualizado na Figura 36.

Todas as respostas salvas pelo aluno podem sofrer alterações durante o prazo definido para o exercício. Ao finalizar o prazo, as respostas ficarão disponíveis para serem avaliadas pelo professor conforme descrito anteriormente.

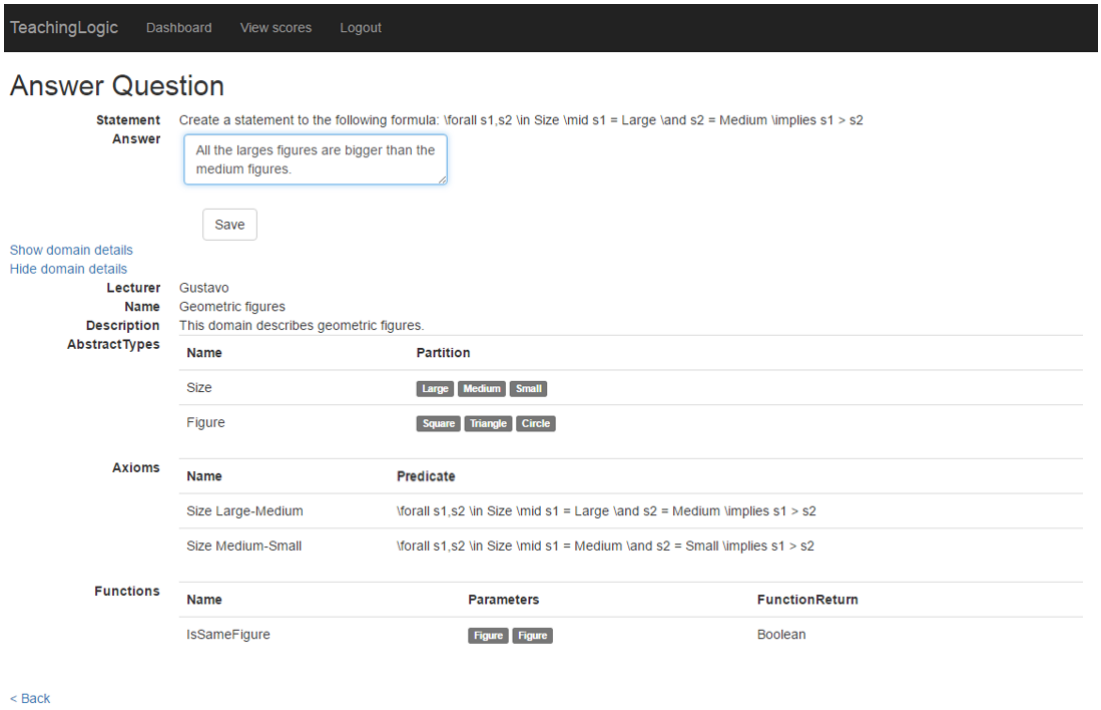

Figura 36 – Tela de detalhamento de pergunta

Para visualizar as notas definidas pelo professor, o aluno acessa a opção "*View scores*", que pode ser encontrada na barra de menu superior em todas as telas após a realização de login com um usuário do tipo aluno. Um exemplo de tela de visualização de notas pode ser observado na Figura 37.

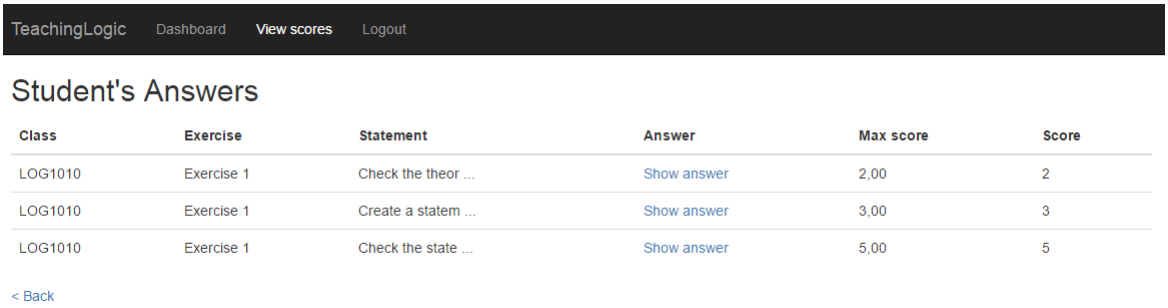

Figura 37 – Tela de visualização de notas

4.4 Comparação do Teaching Logic com outras ferramentas

Considerando a análise realizada na Seção 2.3, e descrição de funcionalidades descritas nas Seções 4.1, 4.2 e 4.3, pode-se observar que o software proposto por esse trabalho visa solucionar as principais limitações identificadas na revisão bibliográfica.

Para permitir a contextualização do conteúdo de lógica, o *Teaching Logic* permite a criação de domínios com liberdade de descrição. Apesar do trabalho desempenhado pelo professor para configurar o domínio, isso torna possível a criação de diversos exemplos, dos mais simples aos mais complexos, fazendo com que o professor possa avaliar a capacidade de entendimento da turma de acordo com a dificuldade do domínio criado.

Ainda no ambiente de avaliação, o professor tem a possibilidade de criar diversas questões para um mesmo domínio, podendo estas serem utilizadas para definir listas de exercícios. Cada lista de exercício possui uma data de início, que é a data que a lista será liberada para os alunos, e uma data de fim, que é a data limite para que os alunos submetam suas respostas. Após a data limite, o professor pode visualizar e avaliar as respostas enviadas pelos alunos.

Com o intuito de facilitar o acesso pelo usuário, *Teaching Logic* foi desenvolvido como uma aplicação web com um layout responsivo, o que possibilita seu acesso a partir de qualquer dispositivo conectado à internet, independentemente do seu sistema operacional. Outra vantagem do Teaching Logic é o fato dele ser disponibilizado como um código aberto, permitindo que outras pessoas possam baixar seu código e realizar as personalizações desejadas e contribuir com o crescimento do projeto.

Em contrapartida, o *Teaching Logic* não realiza verificação de sintaxe das expressões escritas, não permite criar visualizações gráficas do domínio, nem verificar se sentenças são verdadeiras ou falsas nestes, além de não permitir exercícios de prova. Estas funcionalidades podem ser encontradas nas ferramentas mencionadas no Capítulo 2.

5 Conclusões e trabalhos futuros

Este documento apresentou o software *Teaching Logic* para apoiar o ensino de lógica. Espera-se que este software desenvolvido possa contribuir positivamente na formação base dos cursos que contêm lógica na sua estrutura curricular, como Engenharia da Computação, Engenharia Eletrônica, Ciência da Computação, entre outros, pois ele supera boa parte das limitações encontradas no levantamento bibliográfico realizado. Em particular, destaca-se: a necessidade da contextualização do conteúdo de lógica, pois o *Teaching Logic* permite que o professor seja livre durante a criação de um domínio; a falta de um canal para que o professor disponibilize listas de exercícios; e a ausência de um local para que o professor visualize um histórico de notas com o objetivo de identificar quais são as maiores dificuldades da turma, preenchendo essa lacuna; além disso, a ferramenta desenvolvida neste trabalho, ainda permite o acesso remoto a sua plataforma via internet.

A arquitetura utilizada no desenvolvimento da ferramenta foi escolhida para tornar possível sua expansão, de forma que ela divide o código da aplicação em camadas bem definidas, fato que facilita o processo de manutenção e evolução. Ou seja, embora hoje já atenda às expectativas esperadas por este trabalho, chegando ao nível 3 de acordo com a escala de níveis de interação com software educacional, descrita na Seção 2, a ferramenta *Teaching Logic* tem uma alta capacidade de expansão, podendo alcançar o nível 6 a partir do desenvolvimento de um módulo que represente o domínio graficamente, que também permita realizar interações com as representações.

Tais fatos caracterizam a aplicação como uma ferramenta inovadora e de grande potencial na esfera acadêmica, pois ela possui funcionalidades que atingem tanto alunos quanto professores, sendo capaz ainda de evoluir com o aumento de funcionalidades. Como trabalhos futuros, destacam-se as seguintes iniciativas:

- Verificação sintática e semântica de predicados para evitar que sejam escritos erroneamente;
- Integração da ferramenta com um provador de teoremas para que se torne possível definir e responder questões que avaliem a capacidade do aluno em realizar provas algébricas;
- Viabilizar a representação gráfica de domínios por meio de animações e imagens, o que facilitaria o entendimento dos mesmos;
- Permitir interação do usuário com a representação gráfica do domínio, tornando mais simples o entendimento das informações passadas por ele.

Referências

ARNOLD, R.; LANGHEINRICH, M.; HARTMANN, W. Infotraffic: Teaching important concepts of computer science and math through real-world examples. *SIGCSE Bull.*, ACM, New York, NY, USA, v. 39, n. 1, p. 105–109, mar. 2007. ISSN 0097-8418. Disponível em: <http://doi.acm.org/10.1145/1227504.1227349>.

BARKER-PLUMMER JON BARWISE, J. E. D.; LIU, A. *Tarski's World: Revised and Expanded Edition*. [S.l.]: CSLI Publications, 2014.

BRUNER R. R. OLIVER, P. M. G. J. S. *Studies in Cognitive Growth*. [S.l.]: John Wiley and Sons, 1966.

CONERY SCOTT HANSELMAN, P. H. S. G. R. *Microsoft Application Architecture Guide*. [S.l.]: Microsoft Press, 2009.

D'AGOSTINO, M. The tamming of the cut. classical refutations with analytic cut. *Journal of Logic Computation*, 1994.

D'AGOSTINO, M. et al. Winke: A pedagogic tool for teaching logic and reasoning. In: _____. *Intelligent Tutoring Systems: 4th International Conference, ITS' 98 San Antonio, Texas, USA, August 16–19, 1998 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. cap. WinKE: A Pedagogic Tool for Teaching Logic and Reasoning, p. 605–605. ISBN 978-3-540-68716-0. Disponível em: <http://dx.doi.org/10.1007/3-540-68716-5_69>.

ENDRISS, U. The interactive learning environment winke for teaching deductive reasoning. *First International Congress on Tools for Teaching Logic*, 2000.

HERMAN, G. L. et al. Describing the what and why of students’ difficulties in boolean logic. *Trans. Comput. Educ.*, ACM, New York, NY, USA, v. 12, n. 1, p. 3:1–3:28, mar. 2012. ISSN 1946-6226. Disponível em: <http://doi.acm.org/10.1145/2133797.2133800>.

KIM, Y. *The reasoning ability and achievement of college level students enrolled in a logic class in computer science*. Tese (Doutorado) — University of Texas at Austin, 1995.

MICHAUD, L. N. And, or, not: Teaching logic in cs0. *J. Comput. Sci. Coll.*, Consortium for Computing Sciences in Colleges, USA, v. 27, n. 6, p. 99–104, jun. 2012. ISSN 1937-4771. Disponível em: <http://dl.acm.org/citation.cfm?id=2184451.2184476>.

MYERS JR., J. P. The central role of mathematical logic in computer science. *SIGCSE Bull.*, ACM, New York, NY, USA, v. 22, n. 1, p. 22–26, fev. 1990. ISSN 0097-8418. Disponível em: <http://doi.acm.org/10.1145/319059.319071>.

NOLT JOHN; ROHATYN, D. V. A. *Schaum's Outline of Logic*. [S.l.]: McGraw-Hill Education, 2011.

SYROMIATNIKOV, A.; WEYNS, D. A journey through the land of model-view design patterns. In: *2014 IEEE/IFIP Conference on Software Architecture*. [S.l.: s.n.], 2014. p. 21–30.

WENG, J. F.; TSENG, S. S.; LEE, T. J. Teaching boolean logic through game rule tuning. *IEEE Transactions on Learning Technologies*, v. 3, n. 4, p. 319–328, Oct 2010. ISSN 1939-1382.

APÊNDICE A – Diagrama de classes

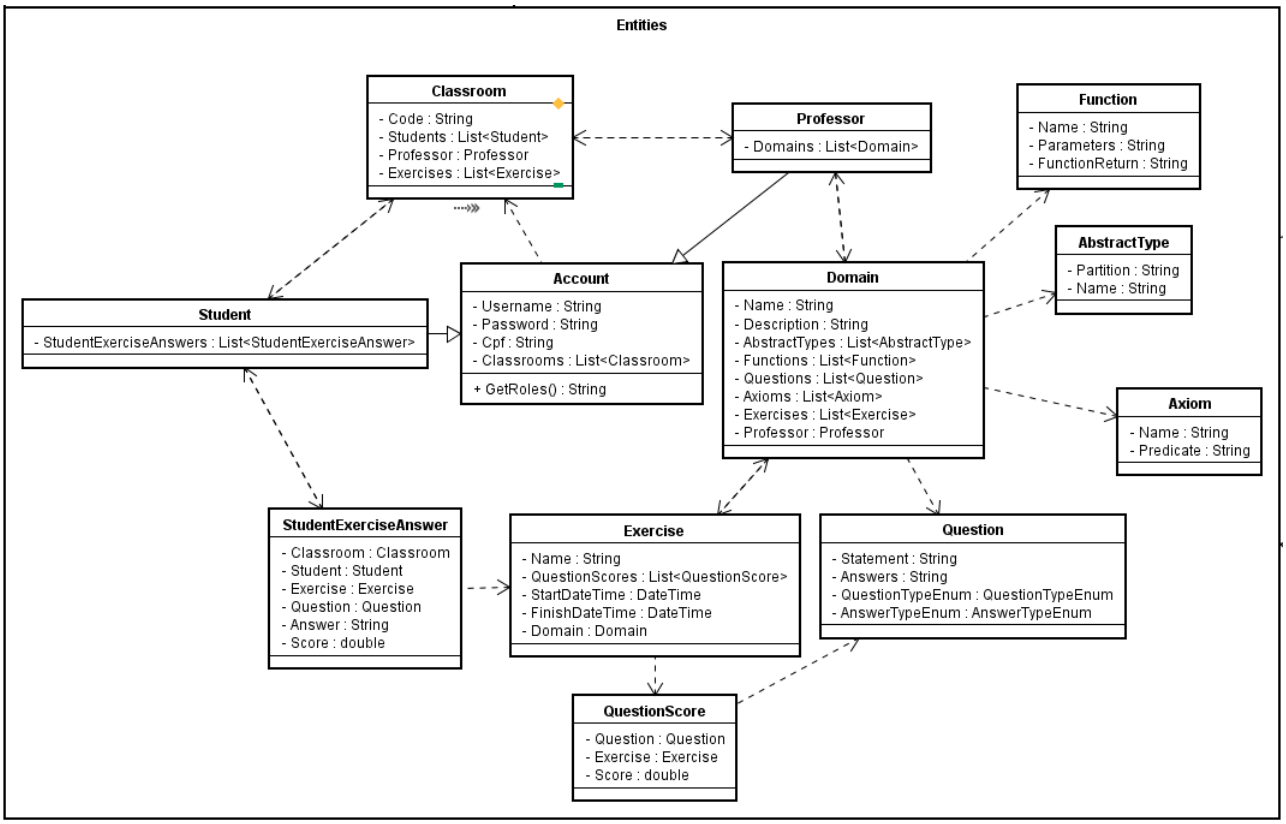

Figura 38 – Diagrama de classes do pacote *Entities*

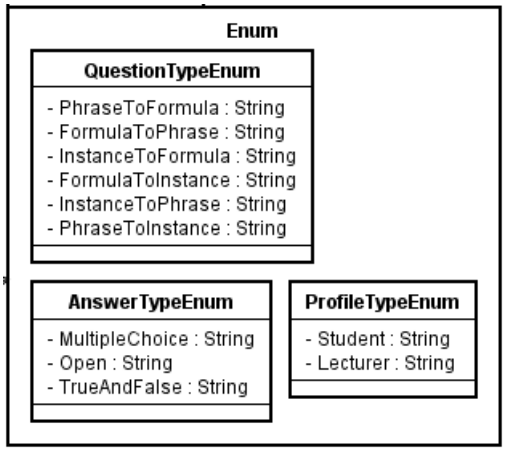

Figura 39 – Diagrama de classes do pacote *Enum*

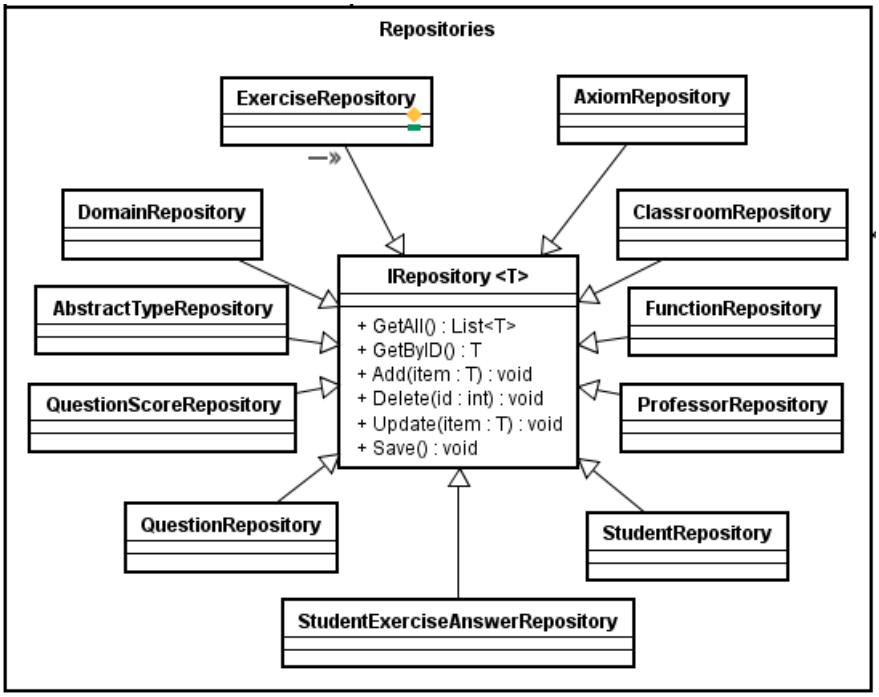

Figura 40 – Diagrama de classes do pacote *Repositories*

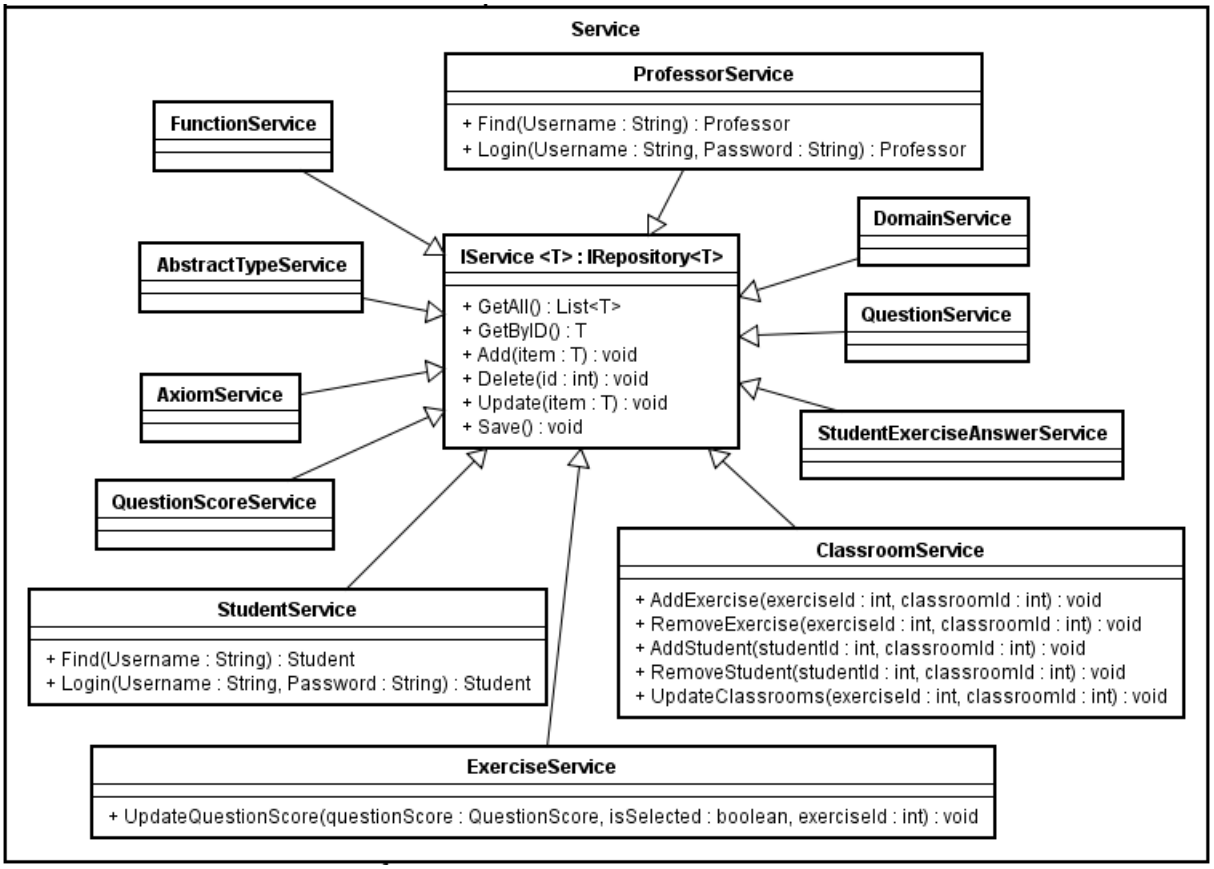

Figura 41 – Diagrama de classes do pacote *Service*

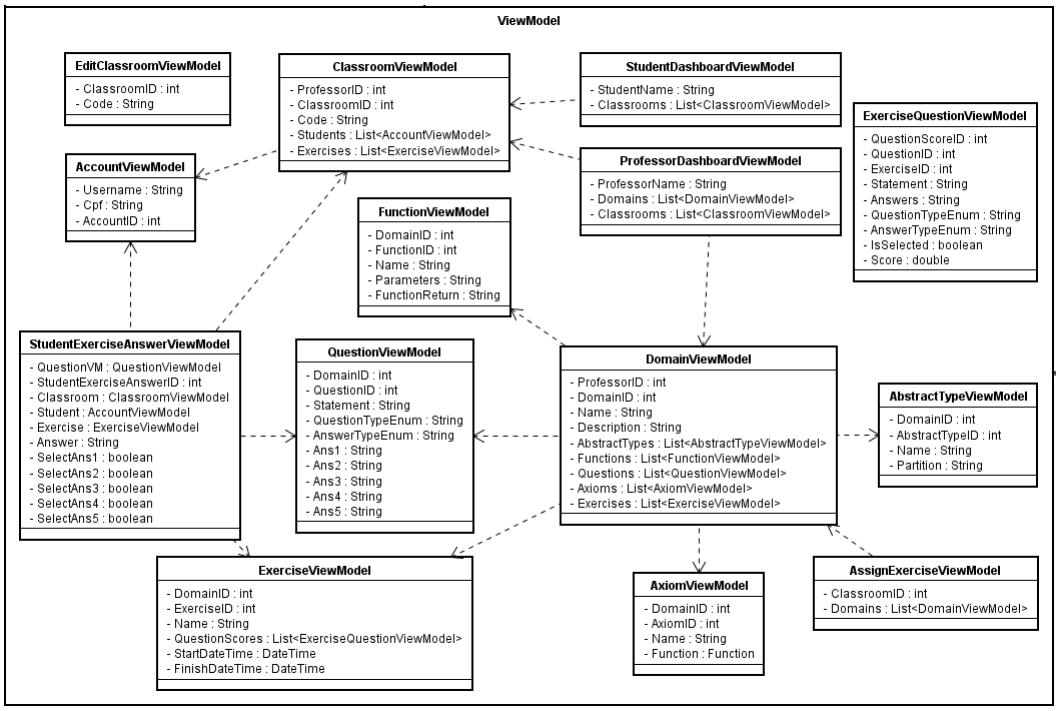

Figura 42 – Diagrama de classes do pacote *ViewModel*

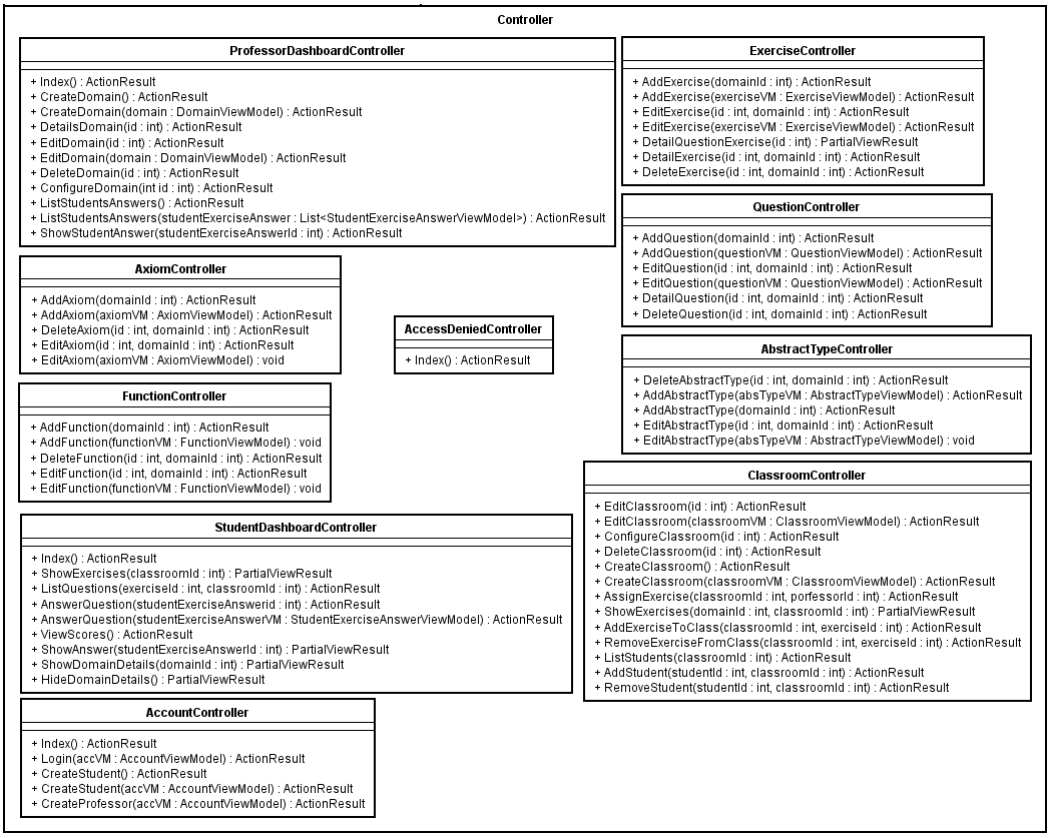

Figura 43 – Diagrama de classes do pacote *Controller*

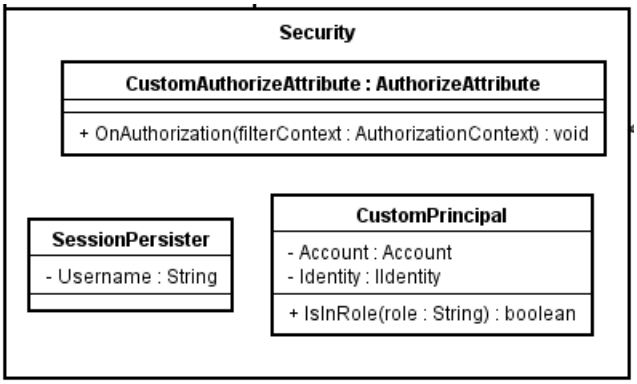

Figura 44 – Diagrama de classes do pacote *Security*

APÊNDICE B – Protótipo de telas

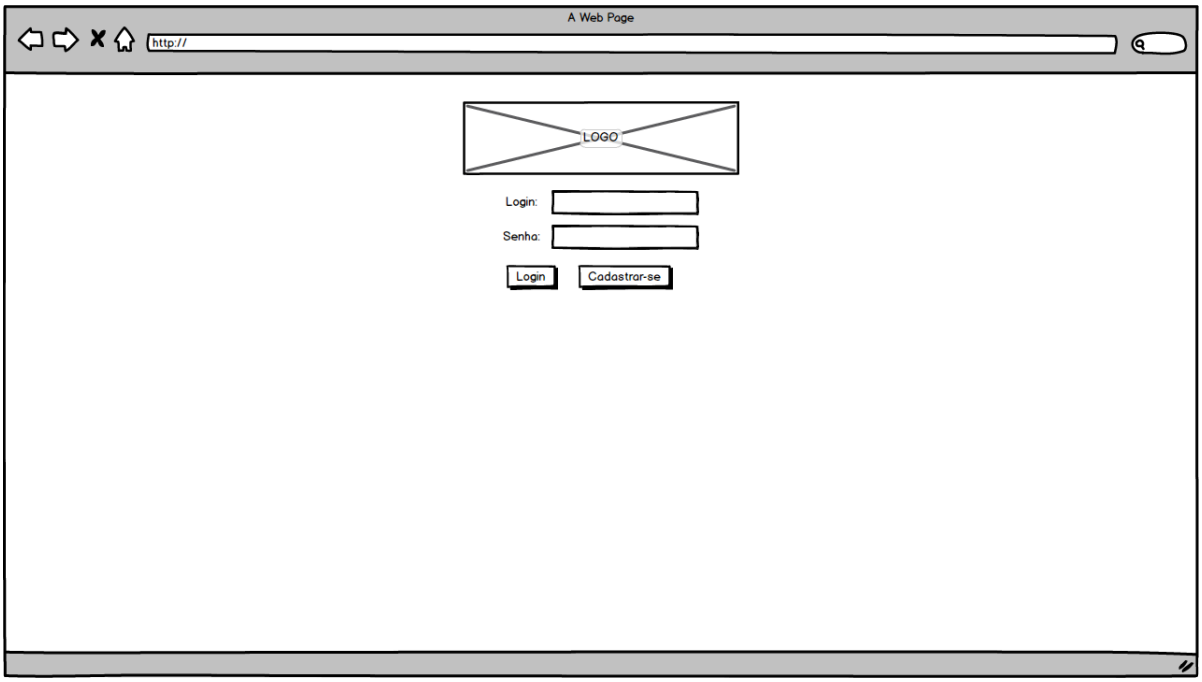

Figura 45 – Protótipo de tela de login

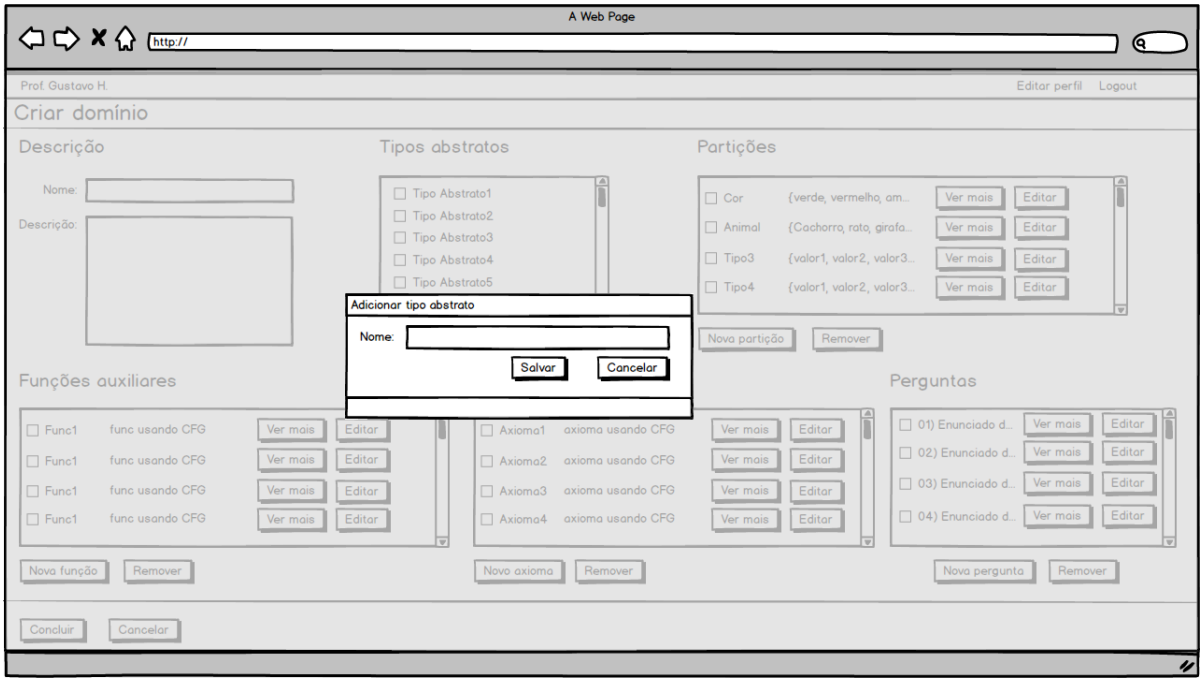

Figura 46 – Protótipo de tela de novo tipo abstrato

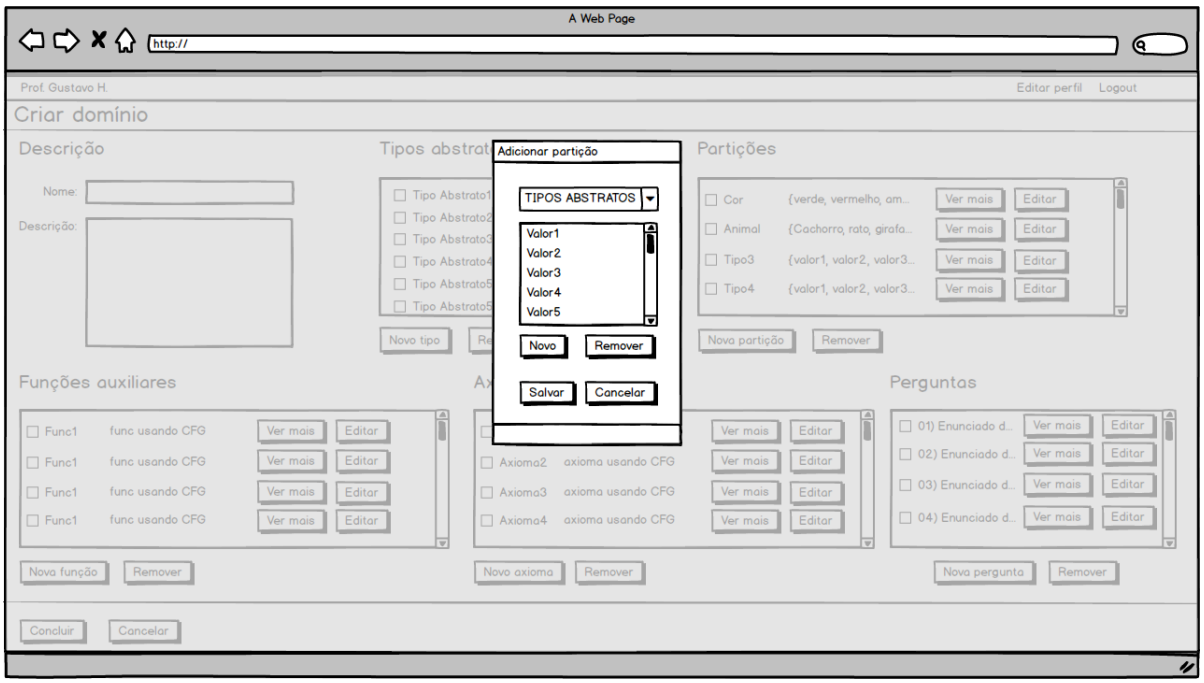

Figura 47 – Protótipo de tela de nova partição

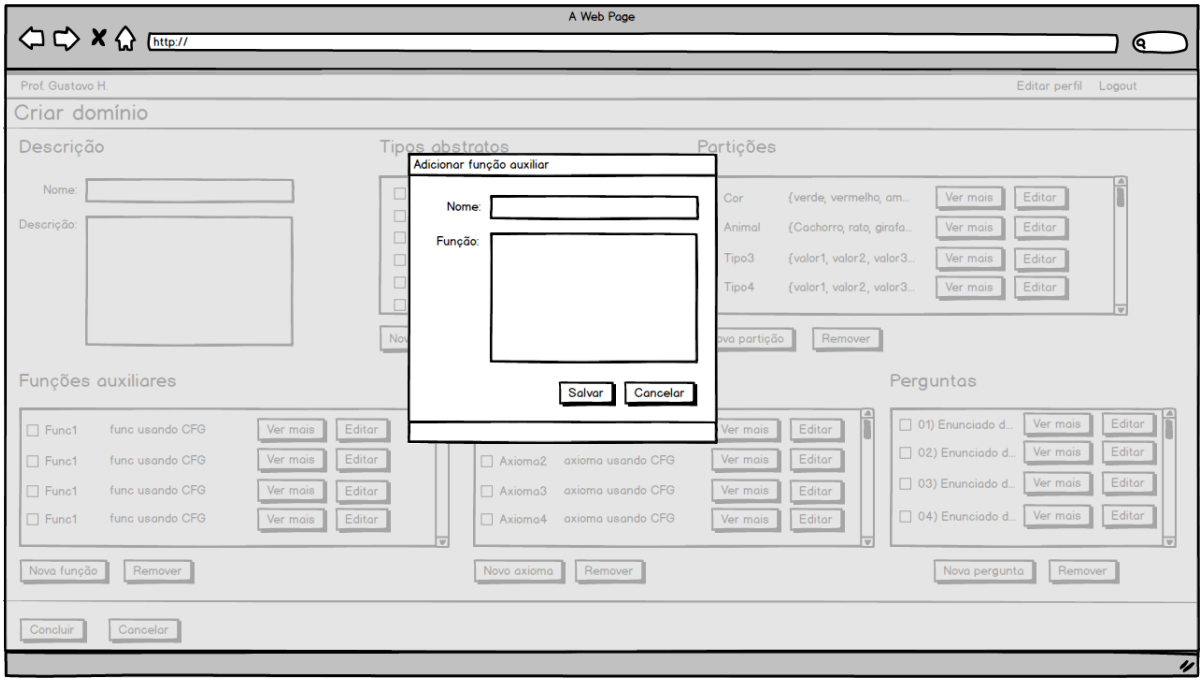

Figura 48 – Protótipo de tela de nova função

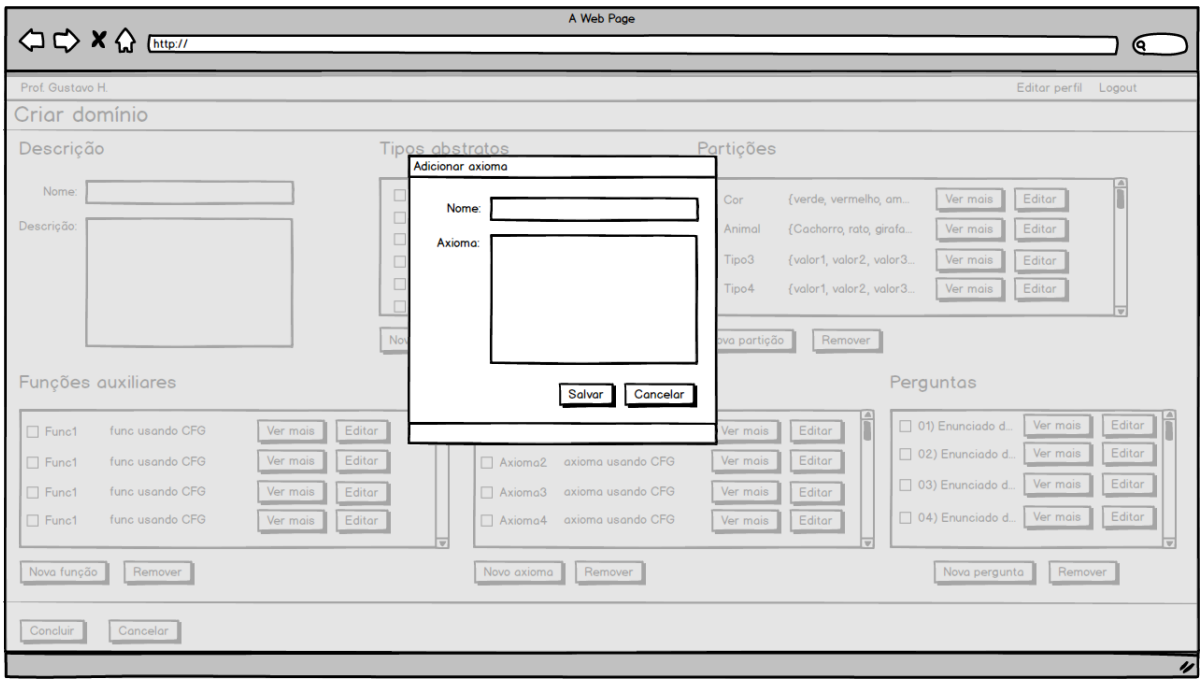

Figura 49 – Protótipo de tela de novo axioma

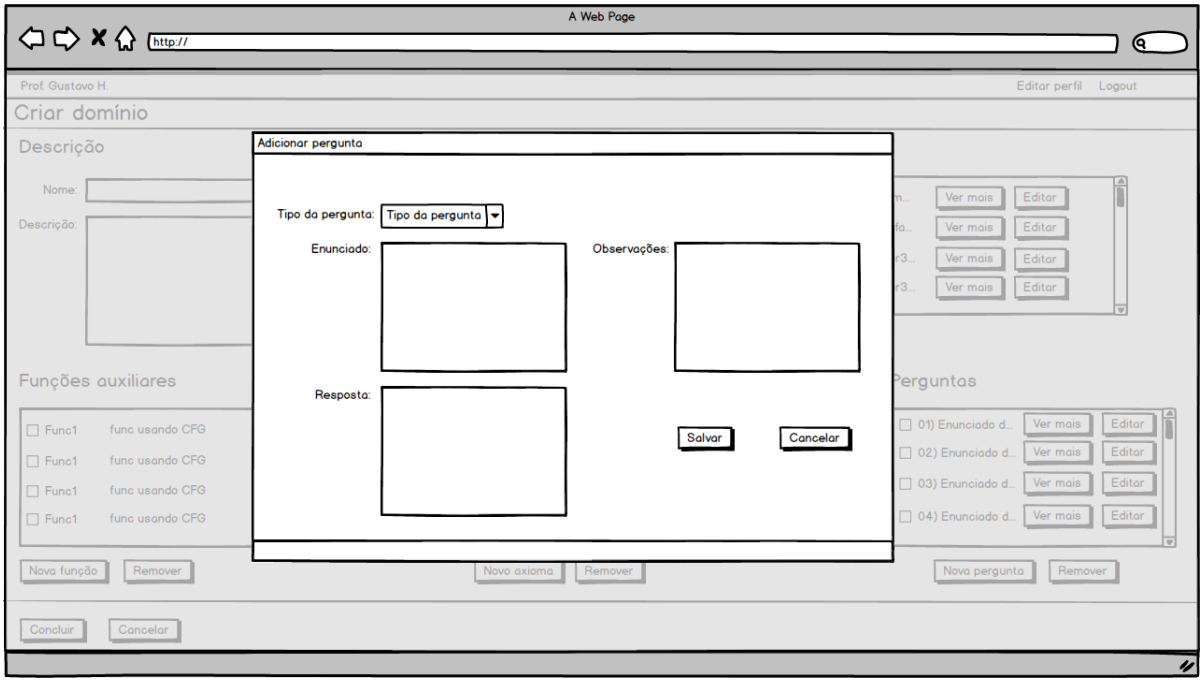

Figura 50 – Protótipo de tela de nova pergunta

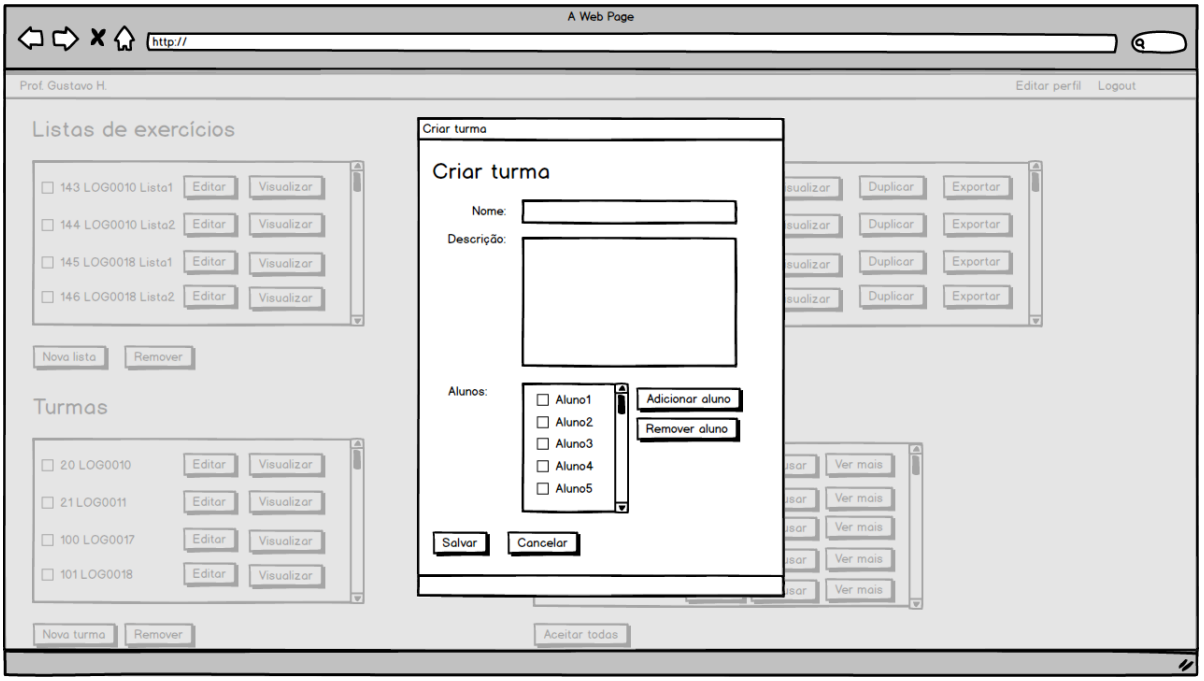

Figura 51 – Protótipo de tela de nova turma

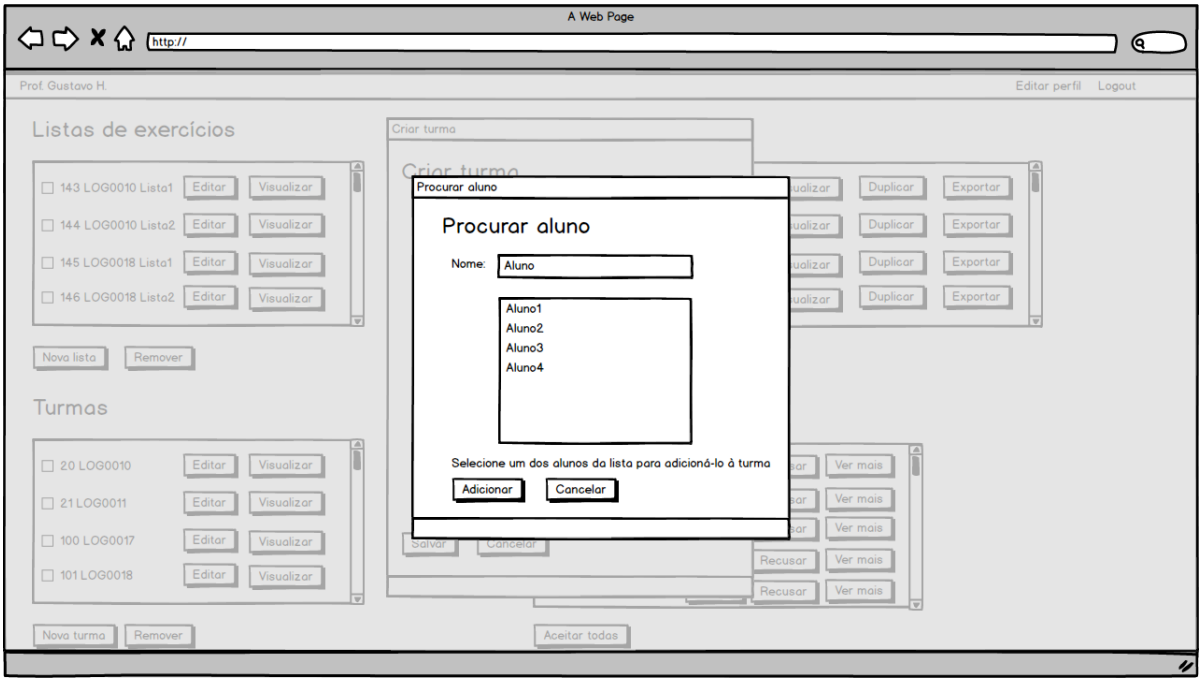

Figura 52 – Protótipo de tela de adição de aluno à turma

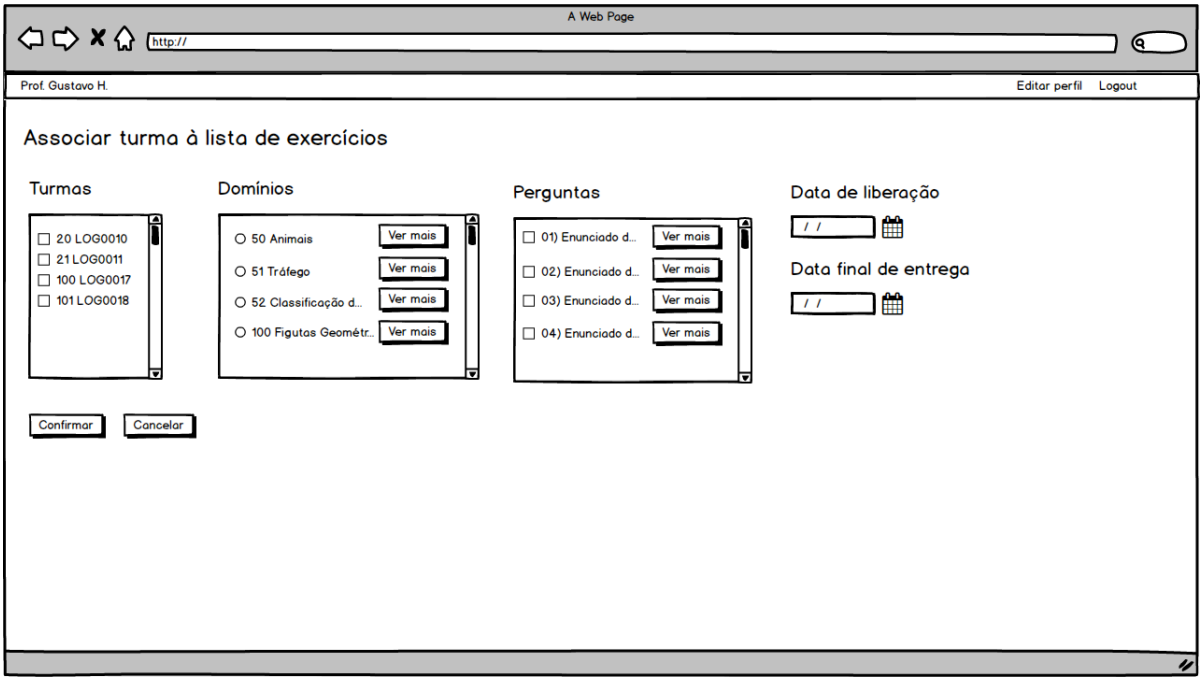

Figura 53 – Protótipo de tela de associação de lista à turma

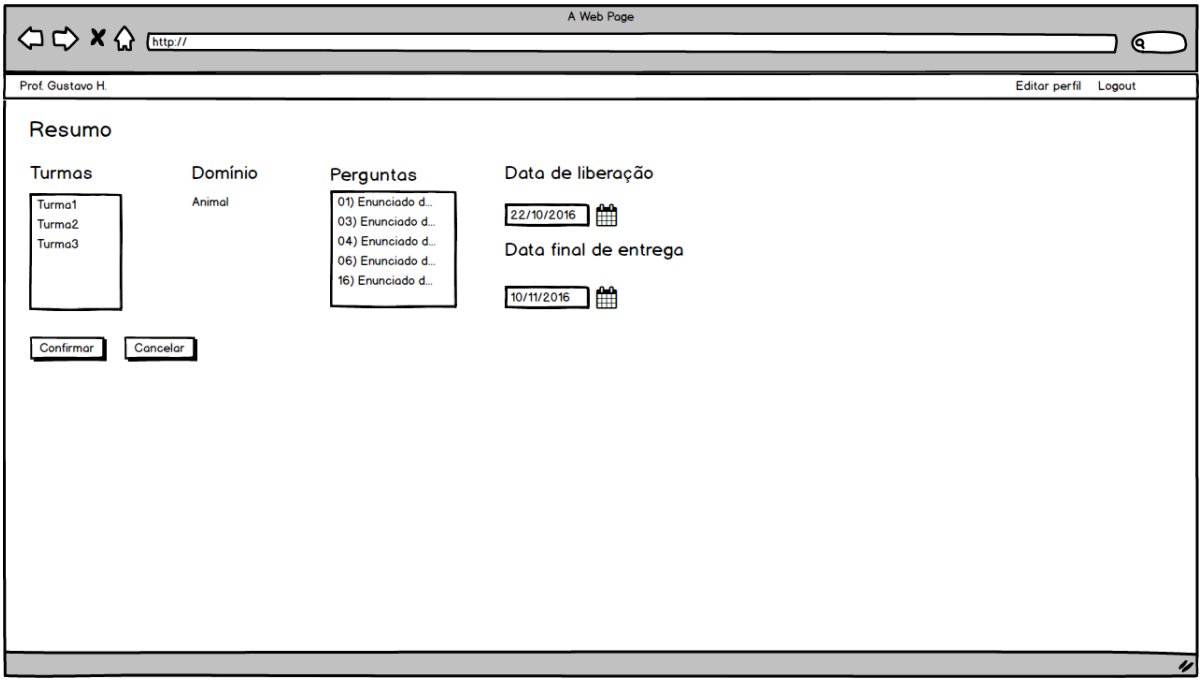

Figura 54 – Protótipo de tela de resumo de associação

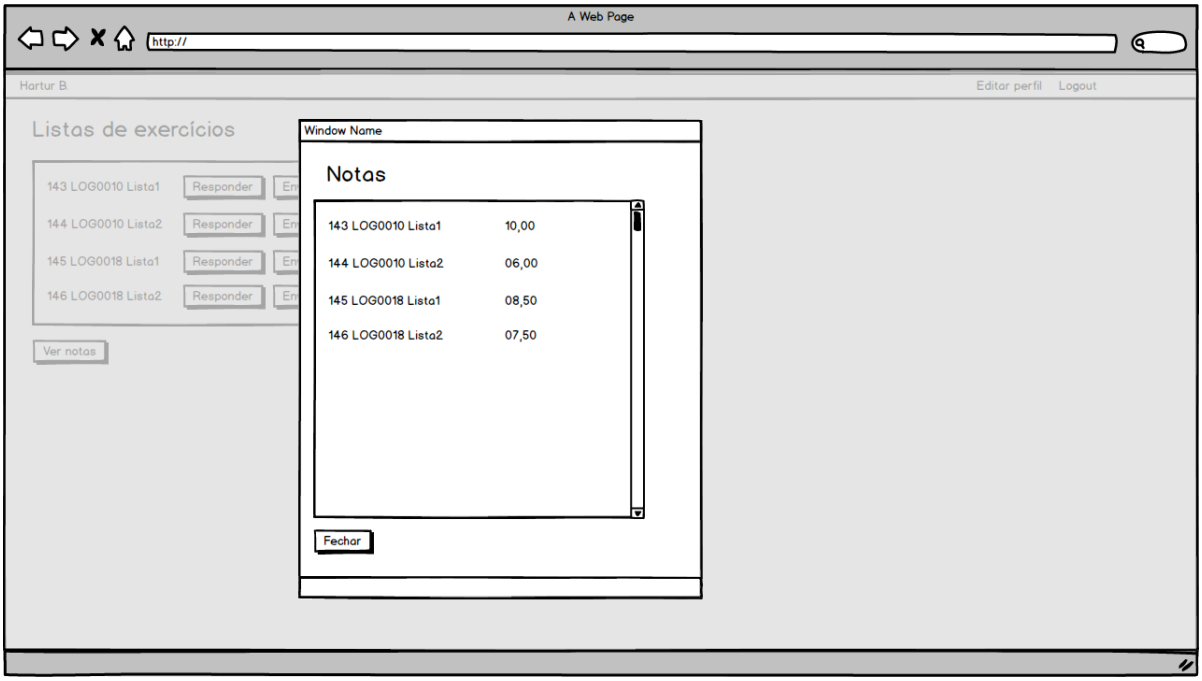

Figura 55 – Protótipo de tela de visualização de notas dos alunos