
Luiz Felipe Ferreira Mai

# Khoeus: uma plataforma de aprendizado focada no ensino da programação

Vitória, ES

2017

Luiz Felipe Ferreira Mai

# Khoeus: uma plataforma de aprendizado focada no ensino da programação

Monografia apresentada ao Curso de Engenharia de Computação do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Departamento de Informática

Orientador: Prof. Dr. Vítor E. Silva Souza

Vitória, ES

2017

Luiz Felipe Ferreira Mai

# Khoeus: uma plataforma de aprendizado focada no ensino da programação

Monografia apresentada ao Curso de Engenharia de Computação do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Trabalho aprovado. Vitória, ES, 07 de dezembro de 2017:

---

**Prof. Dr. Vítor E. Silva Souza**
Orientador

---

**Monalessa Perini Barcellos**
Universidade Federal do Espírito Santo

---

**Rodolfo Costa do Prado**
Universidade Federal do Espírito Santo

Vitória, ES
2017

# Agradecimentos

Primeiramente agradeço às três pessoas que estiveram ao meu lado durante toda a minha jornada: minha mãe, meu pai e meu irmão. Ao longo desses 23 anos aprendi com vocês que jamais deveria desistir e que eu deveria seguir meus sonhos não importasse quais fossem. Ouvi seus ensinamentos e hoje posso dizer que minha graduação se deve principalmente a vocês que me apoiaram ao longo de todo esse tempo e me ajudaram a superar tantas barreiras encontradas pelo caminho.

Também tive ao meu lado grandes amigos que me acompanharam em meio a tanta coisa que passei. Hannah, Bellinha, Caio, Leo, Daniel, Luana e Felipe, conhecer vocês lá no Ensino Médio foi uma das melhores coisas que poderia ter acontecido. Dizem que as amizades de ensino médio ficam no ensino médio, mas fico feliz por termos quebrado essa regra. Sandor, Mateus, Zé, Joyce e Arlindo, obrigado por estarem tão presentes ao longo desses cinco anos e por me fazerem rir praticamente todos os dias ao longo desse tempo. Gabriel, Rodolfo, Leonardo, Janaína, Raíssa, Renan, Igor, Guilherme e Davi, conhecer vocês no PET teve um papel fundamental na graduação e não consigo imaginar como teriam sido minhas tardes sem vocês. João, Cesar e Well, vocês apareceram na minha vida um pouco depois mas logo ganharam um espaço enorme no meu coração e hoje não me vejo sem vocês. A todos esses e a todos os demais que estiveram ao meu lado nessa longa jornada, o meu muito obrigado! Vocês são fundamentais pra mim.

Gostaria de agradecer também a todos os mestres que tive durante esses cinco anos de faculdade por terem me ensinado aquilo que sabiam e por terem me incentivado a buscar por cada vez mais conhecimento. Agradeço em especial ao meu orientador Vítor e à professora Roberta, tutora do PET na época em que fui bolsista. Obrigado por terem se mostrado sempre tão solícitos e disponíveis para me ajudar quando precisei.

Por fim, agradeço imensamente a Deus por ter guiado meus passos até aqui e permitido que eu chegasse até onde cheguei rodeado de todas essas pessoas pelas quais tenho tanto carinho.

*"Você pode encarar um erro como uma besteira a ser esquecida, ou como um resultado que aponta uma nova direção."*

(Steve Jobs)

# Resumo

Hoje, podemos encontrar diversas plataformas de aprendizado espalhadas pela Web que permitem que alunos e professores desenvolvam atividades relacionadas a um curso de maneira mais simplificada. No entanto, pode-se verificar uma série de barreiras quando o intuito é utilizá-las para cursos relacionados à Computação, uma vez que não dão suporte a códigos de programação e, por serem genéricos demais, estes sistemas possuem funcionalidades demais, apresentando uma interface complexa e não sendo bem aceito por parte dos alunos. Com o intuito de solucionar estes problemas apresentados acima, desenvolveu-se o sistema Khoeus que, por meio de funcionalidades *code-oriented*, busca apoiar diretamente o ensino da programação.

Para a construção do sistema, seguiu-se um processo de Engenharia de Software realizando as etapas de levantamento de requisitos, especificação de requisitos, definição da arquitetura do sistema, implementação e testes. Foram colocadas em prática as disciplinas aprendidas no decorrer do curso, tais como Engenharia de Software, Engenharia de Requisitos, Projeto de Sistema de Software, Programação Orientada a Objetos e Desenvolvimento Web e Web Semântica. Também foram utilizados métodos e técnicas de modelagem e desenvolvimento como a técnica de *Test Driven Development* e o método FrameWeb para projeto de aplicações Web baseadas em frameworks.

**Palavras-chaves**: Sistema Web, Educação, Engenharia de Software, Ruby on Rails, *Test Driven Development*, FrameWeb

# Lista de ilustrações

# Lista de abreviaturas e siglas

| | |
|---|---|
| CRUD | *Create, Read, Update and Retrieve* |
| HTML | Linguagem de Marcação de Hipertexto, do inglês *HyperText Markup Language* |
| HTTP | Protocolo de Transferência de Hipertexto, do inglês *HyperText Transfer Protocol* |
| IDE | Ambiente de Desenvolvimento Integrado, do inglês *Integrated Development Environment* |
| PET | Programa de Educação Tutorial |
| RoR | Ruby on Rails |
| SGBD | Sistema Gerenciador de Banco de Dados |
| UML | Linguagem de Modelagem Unificada, do inglês *Unified Modeling Language* |
| URL | Localizador Uniforme de Recursos, do inglês *Uniform Resource Locator* |

# Sumário

# 1 Introdução

Vivemos em uma época em que a computação tem atuado cada vez mais nas diversas áreas do conhecimento e, obviamente, também passou a atuar na Educação. Manter planilhas de notas, listas de presenças e tirar dúvidas de alunos via e-mail são, sem dúvidas, atividades exaustivas e, muitas vezes, burocráticas. Com o advento de sistemas computacionais, os ambientes de apoio ao aprendizado se popularizaram entre as instituições de ensino superior por remover parte do esforço exigido de um professor e por facilitar o acesso do aluno às informações que deseja.

O PET (Programa de Educação Tutorial) da Engenharia de Computação da Universidade Federal do Espírito Santo (PET EngComp) desenvolve uma série de atividades de ensino, pesquisa e extensão e, dentre elas, está o Introcomp: Introdução à Computação. O projeto consiste no ensino de programação básica de computadores a alunos de ensino médio das escolas públicas (federais e estaduais) da Grande Vitória, no estado do Espírito Santo. Para a realização do projeto, uma série de ferramentas são utilizadas a fim de auxiliar o processo de aprendizado do aluno, bem como facilitar a metodologia de ensino dos professores. Nomeando algumas delas, temos o site do curso, o grupo do Facebook, o fórum de discussões (hoje não mais utilizado) e o Moodle, uma plataforma de aprendizado que permite que alunos e professores desenvolvam atividades relacionadas ao curso de maneira mais simplificada.

Em relação ao Moodle, no entanto, esbarrou-se em dois grandes problemas: (i) a plataforma não dá suporte a códigos de programação e isso dificulta a submissão de tarefas por parte dos alunos e sua correção por parte dos professores; (ii) com o intuito de ser genérico e atender as massas, o sistema possui funcionalidades demais, apresentando uma interface complexa e não sendo bem aceito por parte dos alunos.

Dessa forma, este trabalho vem com o intuito de construir uma plataforma de aprendizado chamada Khoeus, para solucionar os dois problemas apresentados acima por meio de funcionalidades *code-oriented* bem como uma interface simples e de fácil manuseio. Dessa forma, espera-se criar um ambiente em que tanto alunos como professores sintam-se à vontade para utilizá-lo e que apoie diretamente o aprendizado nos mais diversos cursos, sejam eles relacionados à programação ou não.

## 1.1 Objetivos

O objetivo geral deste trabalho é desenvolver um sistema Web que será utilizado por professores e alunos de diversos cursos a fim de auxiliá-los em atividades rotineiras

como publicação de materiais, submissão de tarefas e realização de chamada. Para isso, serão utilizados os conceitos aprendidos ao longo do curso de Engenharia de Computação. São objetivos específicos deste projeto:

- Projetar o sistema Khoeus com base no método FrameWeb (SOUZA, 2007), estendendo seus modelos para o framework *Ruby on Rails*.
- Desenvolver o sistema a fim de que seja utilizado pelo Introcomp em suas edições futuras.

## 1.2 Método de Desenvolvimento do Trabalho

A metodologia utilizada para desenvolver este trabalho foi composta pelas seguintes atividades:

1. *Revisão Bibliográfica*: estudo de boas práticas de Engenharia de Software e de Requisitos, Padrões de Projeto de Sistemas, Programação Orientada a Objetos, uso e projeto de Banco de Dados Relacional, entre outros;
2. *Elaboração da Documentação do Sistema:* definição dos documentos do sistema. Em primeiro lugar, foi elaborado o Documento de Especificação de Requisitos, apresentando uma descrição geral do minimundo do sistema, definição dos requisitos funcionais e não funcionais, além das regras de negócio. Também estão neste documento a apresentação dos subsistemas, casos de uso, modelo estrutural e glossário do projeto. Por fim, foi elaborado o Documento do Projeto, contendo a arquitetura do software e projeto detalhado de cada um dos seus componentes, seguindo a abordagem FrameWeb;
3. *Estudo das Tecnologias:* estudo das tecnologias utilizadas para o desenvolvimento do sistema, tais como a linguagem de programação Ruby, o framework Ruby on Rails, o banco de dados PostgreSQL, dentre outras;
4. *Implementação e Testes:* implementação do sistema e bateria de testes. Para facilitar este processo, foi utilizado o padrão *Test Driven Development* por meio da biblioteca *Rspec* a fim de automatizar os testes no sistema.
5. *Redação da Monografia:* escrita desta monografia. Vale ressaltar que a mesma foi escrita em *LaTeX*[1] utilizando o editor *TexStudio*[2] e o template *abnTeX*[3] que atende os requisitos das normas da ABNT (Associação Brasileira de Normas Técnicas) para elaboração de documentos técnicos e científicos brasileiros.

[1] <http://www.latex-project.org/>.
[2] <http://www.texstudio.org/>.
[3] <http://www.abntex.net.br>.

## 1.3 Organização do Texto

Esta monografia é estruturada em cinco partes e contém, além da presente introdução, os seguintes capítulos:

- **Capítulo 2** – Referencial Teórico: apresenta uma revisão da literatura acerca de temas relevantes ao contexto deste trabalho, a saber: Engenharia de Software, desenvolvimento Web e o método FrameWeb;
- **Capítulo 3** – Especificação de Requisitos: apresenta a especificação de requisitos do sistema, descrevendo o minimundo e exibindo os seus diagramas de classes e casos de uso;
- **Capítulo 4** – Projeto Arquitetural do sistema: apresenta a arquitetura estabelecida para o desenvolvimento do sistema, bem como os diagramas das diferentes camadas do sistema, construídos conforme a abordagem FrameWeb;
- **Capítulo 5** – Apresentação do sistema: apresenta as partes principais da implementação do sistema, ilustradas por capturas de telas;
- **Capítulo 6** – Considerações Finais: apresenta as conclusões do trabalho, dificuldades encontradas, limitações e propostas de trabalhos futuros.
- **Apêndices** – Apresenta os documentos gerados nos processos de Engenharia de Software (Documento de Requisitos e Documento de Projeto)

# 2 Referencial Teórico

Este capítulo apresenta os principais conceitos teóricos que fundamentaram o desenvolvimento do sistema *Khoeus* e está organizado em 4 seções. A Seção 2.1 aborda a Engenharia de Software, destacando os principais conceitos e processos utilizados. A Seção 2.2 apresenta os principais conceitos de desenvolvimento Web. A Seção 2.3 fala sobre o método *Test Driven Development* para automatizar os testes do sistema. A Seção 2.4 apresenta o método FrameWeb.

## 2.1 Engenharia de Software

O desenvolvimento de software é, sem dúvida, uma atividade que vem ganhando cada vez mais importância nos dias atuais pelo simples fato dos computadores estarem sendo utilizados cada vez mais nas mais diversas áreas do conhecimento humano. Visando melhorar a qualidade dos produtos de software e aumentar a produtividade no processo de desenvolvimento, surgiu a Engenharia de Software (FALBO, 2014).

Podemos dizer, de forma simplista, que um software consiste em:

1. instruções que, quando executadas, fornecem características, funções e resultados desejados;
2. estruturas de dados que possibilitam aos programas manipular informações adequadas;
3. informação descritiva, tanto na forma impressa como na virtual, descrevendo a operação e o uso dos programas.

Paralelamente a isso, pode-se dizer que a Engenharia de Software abrange um conjunto de práticas e um leque de ferramentas que possibilitam aos profissionais desenvolverem software de qualidade (PRESSMAN, 2011) por meio da especificação, desenvolvimento e manutenção destes sistemas de software. Tudo isso é feito por meio de tecnologias e práticas de gerência de projetos e outras disciplinas, visando organização, produtividade e qualidade nesse processo de desenvolvimento. A Engenharia de Software trata de aspectos relacionados ao estabelecimento de processos, métodos, técnicas, ferramentas e ambientes de suporte ao desenvolvimento de software (FALBO, 2014).

Embora um processo seja um conjunto de atividades, ações e tarefas realizadas na criação de algum produto, essa definição acaba se diferenciando no contexto da Engenharia de Software: aqui, um processo é uma abordagem adaptável que possibilita à equipe de soft-

ware realizar o trabalho de escolher o conjunto apropriado de ações e tarefas (PRESSMAN, 2011).

Como atividades de um processo de desenvolvimento de software, podemos citar a etapa de especificação e análise de requisitos e a etapa de projeto e implementação. Porém, vale ressaltar que paralelo a estas etapas principais, são realizados diversos processos de apoio e gerência, tais como elaboração de cronogramas, análise dos riscos etc.

Nas próximas seções, serão levantados alguns pontos referentes a este processo de desenvolvimento de software. Na Seção 2.1.1, será abordada a etapa de especificação e análise de requisitos. Na Seção 2.1.2, a etapa retratada será a de projeto e implementação. Em ambas, serão levantados alguns conceitos importantes para a uma melhor compreensão de sua importância para o processo de desenvolvimento de software como um todo.

### 2.1.1 Especificação e Análise de Requisitos

A Engenharia de Requisitos é considerada uma das etapas mais cruciais no planejamento e no desenvolvimento de software, uma vez que lida com os problemas que surgem no planejamento do software correto para o cliente. Dessa forma, tem se tornado cada vez mais um processo que opera em diferentes níveis, incluindo aspectos relativos ao produto, ao projeto e à sua organização. O desenvolvimento da especificação de requisitos de um software é reconhecido como uma das bases das funcionalidades de um sistema. Estes requisitos são determinantes da qualidade do software, dado que estudos empíricos mostraram que erros nos requisitos são as falhas mais comuns no ciclo de vida de um software, bem como as mais caras e custosas a corrigir (AURUM et al., 2013).

Por definição (ASSOCIATION et al., 1990) um **requisito** é:

1. Uma capacidade necessária do ponto de vista de um usuário para resolver um problema ou alcançar um objetivo;

2. Uma condição que deve ser alcançada por um sistema ou componentes de um sistema a fim de satisfazer um contrato, norma, especificação ou outros documentos de formalização;

3. Uma representação documentada de uma condição ou capacidade conforme (1) ou (2).

Dito isto, podemos dizer que as atividade de especificação e análise de requisito não diz respeito apenas a perguntar às pessoas o que elas desejam, mas sim analisar cuidadosamente a organização, o domínio da aplicação e os processos de negócio no qual o sistema será utilizado (KOTONYA; SOMMERVILLE, 1998). Uma parte essencial dessa fase é a elaboração de modelos descrevendo o quê o software tem de fazer (e não como

fazê-lo), dita Modelagem Conceitual. Até este momento, a ênfase está sobre o domínio do problema e não se deve pensar na solução computacional a ser adotada. Com os requisitos pelo menos parcialmente capturados e especificados na forma de modelos, pode-se começar a trabalhar no domínio da solução (FALBO, 2017).

Pode-se dizer que os requisitos de um sistema incluem especificações dos serviços que o sistema deve prover, restrições sob as quais ele deve operar, propriedades gerais do sistema e restrições que devem ser satisfeitas no seu processo de desenvolvimento (FALBO, 2017), sendo estes requisitos divididos em **requisitos funcionais** e **requisitos não funcionais**, definidos segundo (AURUM et al., 2013) da seguinte forma:

- **Requisitos Funcionais:** aquilo que o sistema fará em termos de tarefas e serviços;
- **Requisitos Não-funcionais:** restrições às soluções utilizadas para os requisitos funcionais, como desempenho e segurança.

Embora seja complexo, é possível dividir o processo de especificação e análise de requisitos em algumas etapas, conforme proposto por (KOTONYA; SOMMERVILLE, 1998) e mostrado na Figura 1.

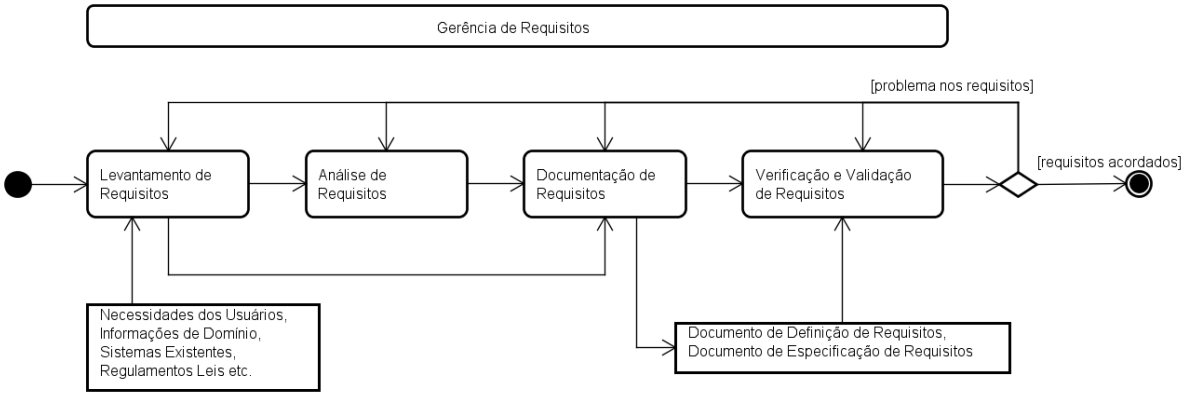


Figura 1 – Processo de Engenharia de Requisitos adaptado de (KOTONYA; SOMMERVILLE, 1998).

O processo começa pelo levantamento de requisitos, que deve levar em conta necessidades dos usuários e clientes, informações de domínio, sistemas existentes, regulamentos, leis etc. Uma vez identificados requisitos, é possível iniciar a atividade de análise, quando os requisitos levantados são usados como base para a modelagem do sistema. Tanto no levantamento quanto na análise de requisitos, é importante documentar requisitos e modelos. Para realizar essa documentação, dois documentos são normalmente utilizados: o Documento de Definição de Requisitos, contendo uma lista dos requisitos de cliente identificados, e o Documento de Especificação de Requisitos, que registra os requisitos de sistema e os vários diagramas resultantes do trabalho de análise. Os documentos produzidos

são, então, verificados e validados. Adicionalmente, um esforço de garantia da qualidade deve ser realizado, visando garantir conformidade em relação a padrões e ao processo estabelecidos pela organização. Caso clientes, usuários e desenvolvedores estejam de acordo com os requisitos, o processo de desenvolvimento pode avançar; caso contrário, deve-se retornar à atividade correspondente para resolver os problemas identificados. Em paralelo a todas as atividades anteriormente mencionadas, há a gerência de requisitos, que se ocupa em gerenciar mudanças nos requisitos (FALBO, 2017).

Dentre os modelos mais utilizados durante a etapa de especificação e análise de requisito, pode-se citar o modelo de casos de uso e o modelo conceitual estrutural. Enquanto o propósito do modelo de casos de uso é capturar e descrever a funcionalidade que um sistema deve prover, o modelo conceitual estrutural de um sistema tem por objetivo descrever as informações que devem ser representadas e gerenciadas (FALBO, 2017).

### 2.1.2 Projeto e Implementação

A fase de projeto tem por objetivo definir e especificar uma solução a ser implementada. É uma fase de tomada de decisão, tendo em vista que muitas soluções são possíveis. Além disso, o projeto é um processo de refinamento. Inicia-se com o projeto da arquitetura do sistema, que visa descrever a estrutura de nível mais alto da aplicação, identificando seus principais elementos ou componentes e como eles se relacionam uns com os outros. Uma vez definida a arquitetura, o projeto passa a se concentrar no detalhamento de cada um desses elementos, até atingir o nível de unidades de implementação (FALBO, 2016), correspondendo à primeira atividade que leva em conta aspectos tecnológicos, sendo, portanto, a fase do processo de software na qual os requisitos, as necessidades do negócio e as considerações técnicas se juntam na formulação de um produto ou sistema de software (PRESSMAN, 2011).

Inicialmente, o projeto é representado em um nível alto de abstração, enfocando a estrutura geral do sistema. Definida a arquitetura, o projeto passa a tratar do detalhamento de seus elementos. Esses refinamentos conduzem a representações de menores níveis de abstração, até se chegar ao projeto de algoritmos e estruturas de dados. Assim, independentemente do paradigma adotado, o processo de projeto envolve as seguintes atividades (FALBO, 2016):

- **Projeto da Arquitetura do Software:** visa definir os elementos estruturais do software e seus relacionamentos.
- **Projeto dos Elementos da Arquitetura:** visa projetar em um maior nível de detalhes cada um dos elementos estruturais definidos na arquitetura, o que envolve a decomposição de módulos em outros módulos menores.

- **Projeto Detalhado:** tem por objetivo refinar e detalhar os elementos mais básicos da arquitetura do software, i.e., as interfaces, os procedimentos e as estruturas de dados. Deve-se descrever como se dará a comunicação entre os elementos da arquitetura (interfaces internas), a comunicação do sistema em desenvolvimento com outros sistemas (interfaces externas) e com as pessoas que vão utilizá-lo (interface com o usuário), bem como se devem projetar detalhes de algoritmos e estruturas de dados.

Ao final da fase de Projeto e Implementação, espera-se que a arquitetura do sistema esteja definida, bem como o projeto de seus componentes, divididos geralmente em camadas a fim de assegurar a separação de funcionalidades pertencentes a nichos diferentes. Embora não seja a única, uma das estruturas mais conhecida para o projeto de software é a que o divide em camada de apresentação, camada de domínio e camada de persistência (FOWLER, 2002). No caso da Web, na camada de **apresentação**, espera-se lidar com requisições HTTP, exibição de informação e iterações do usuário (como cliques de mouse, etc). Na camada de **domínio**, como o próprio nome já diz, o foco está nas variáveis do domínio do sistema, implementando todas as regras de negócio e requisitos do sistema. Já na camada de **persistência**, espera-se realizar todo o relacionamento da aplicação com o banco de dados, permitindo armazenar as informações do sistema e recuperá-las quando necessário.

## 2.2 Desenvolvimento Web

O desenvolvimento de aplicações Web pode ser mais complexo e desafiador do que pode-se imaginar. Dois atributos-chave distinguem o desenvolvimento de softwares baseados na Web do desenvolvimento de softwares tradicionais: um rápido crescimento dos requisitos de sistemas Web e a mudança contínua de seus conteúdos. Dessa forma, aplicações Web precisam ser planejadas para serem escaláveis e de fácil manutenção, uma vez que tais recursos não podem ser adicionados posteriormente. O sucesso na construção, desenvolvimento, implementação e manutenção de um sistema Web depende fortemente de quão bem essas questões foram tratadas.

Com o avanço da Web, surgiu a chamada Engenharia Web, que lida com o processo de desenvolvimento de aplicações e sistemas Web. Sua essência é de gerenciar a diversidade e a complexidade dessas aplicações, evitando potenciais falhas que possam ocasionar implicações sérias. Pode-se dizer que a Engenharia Web é uma abordagem proativa de desenvolver aplicações Web (GINIGE; MURUGESAN, 2001).

### 2.2.1 O Ambiente Web

Para que as aplicações Web sejam executadas corretamente no computador do usuário, existem diversas tecnologias envolvidas neste processo e que mostram-se de suma importância para a total compreensão do problema. Desde o surgimento da Web, o projeto evoluiu e são várias as razões para o seu atual sucesso, mas duas merecem destaque: sua arquitetura simples (mas eficiente) e uma interface igualmente simples, originalmente baseada no paradigma de hipertextos.

A arquitetura por trás da Web é, basicamente, um cliente/servidor instalado sobre uma rede de computadores heterogênea. Do lado do cliente está um programa, chamado *browser* ou navegador, que intermedeia a solicitação de informações ao servidor e as apresenta para o usuário. Do outro lado, o servidor atende os diferentes clientes bem como outros servidores indistintamente. Esta arquitetura tem 3 componentes principais: o protocolo de comunicação *HyperText Transfer Protocol* (HTTP), o sistema de endereçamento *Uniform Resource Locator* (URL) e a linguagem *HyperText Markup Language* (HTML).

O **protocolo HTTP** é um meio de transporte de arquivos na Web que é executado sobre a camada TCP/IP da Internet. O protocolo consiste basicamente da transição de 4 estados: conexão (o cliente estabelece uma conexão com o servidor); requisição (o cliente envia um pedido ao servidor); resposta (o servidor devolve uma resposta ao cliente); e encerramento (a conexão é desfeita por ambas as partes). Quando um documento ou um objeto (como uma imagem, por exemplo) é enviado para o cliente, é anexado um cabeçalho com a informação necessária para que o *browser* possa interpretá-lo e apresentá-lo adequadamente. Isto torna o protocolo independente do *browser*, que ignora o conteúdo de objetos cujo cabeçalho não compreende.

**URL** é a forma conhecida de endereçamentos de objetos na Web. Consiste na identificação do esquema utilizado (HTTP, FTP etc.) seguido do caminho até o objeto ou documento como, por exemplo: <http://www.ufes.br/index.html>.

O terceiro componente é a linguagem **HTML**, que especifica a estrutura e a formatação para documentos do tipo hipertexto por meio de *tags* que indicam a forma como este deve ser visualizado. Pode-se destacar duas grandes vantagens do HTML: (1) a facilidade para associar informações por meio de links permitindo criar grandes redes de dados; e (2) o mecanismo de navegação uniforme com a simples seleção de objetos associados a links (WINCKLER; PIMENTA, 2002).

### 2.2.2 O padrão MVC de arquitetura de software

Com o advento da Internet e da procura cada vez maior por sistemas Web, fez-se necessário criar formas de agilizar o desenvolvimento destes sem comprometer a qualidade do produto final. Com isso, surgiram vários padrões de arquitetura de software em busca

de atributos de qualidade como: (1) decentralização do desenvolvimento por dividir o código da aplicação em vários componentes, possibilitando que os desenvolvedores possam trabalhar paralelamente em diferentes componentes sem que isso cause impactos no que foi desenvolvido por outra pessoa; e (2) fraco acoplamento, uma vez que um dos princípios destes padrões de arquitetura é que suas camadas sejam o mais isoladas possíveis a fim de que não interfiram nas funcionalidades de uma outra.

Dentre os diversos padrões propostos, um dos mais utilizados é o chamado MVC: *Model-View-Controller*. Como o seu próprio nome já descreve, esse padrão busca a divisão da aplicação em três camadas, conforme mostra a Figura 2. Neste modelo, cada camada possui uma funcionalidade bem específica:

- **Model:** consiste na camada que contém a lógica da aplicação, sendo responsável pelas regras de negócio e pela interação da aplicação com o banco de dados.
- **View:** basicamente, é a camada que interage diretamente com o usuário, exibindo dados por meio de arquivos HTML, XHTML, PDF, entre outros. No caso do *Ruby on Rails*, a camada de visão utiliza arquivos HTML contendo códigos Ruby embutidos.
- **Controller:** pode-se dizer que essa camada é a intermediadora do sistema, comunicando-se com as outras duas camadas. As requisições provenientes do *browser* são processadas pelo *controller*, que processa as informações dos *models* e as retorna para as *views* para que sejam exibidas ao usuário.

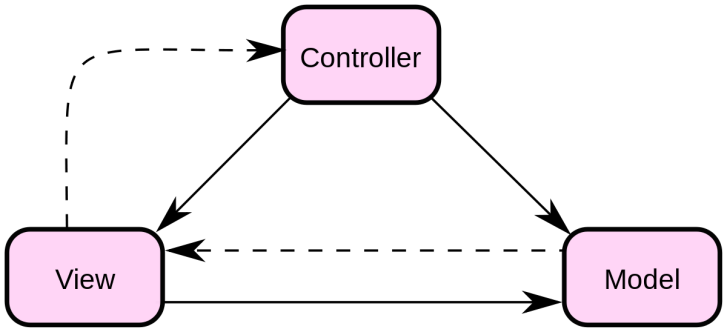


Figura 2 – Representação do modelo de arquitetura de software *Model-View-Controller* (WIKIPEDIA, 2017)

### 2.2.3 Ruby e seu framework *Ruby on Rails*

Ruby é uma linguagem moderna, *open source* e orientada a objetos: tudo que pode ser manipulado é um objeto, assim como o resultado dessas manipulações. Além disso, Ruby consegue ser concisa sem ser ininteligível: é possível expressar ideias naturalmente e de forma clara com códigos desenvolvidos em Ruby. Isso acarreta em programas que são fáceis de escrever e fáceis de interpretar meses após sua escrita.

Para o desenvolvimento Web, o Ruby oferece o *framework Ruby on Rails* (RoR) que facilita o processo de codificação, implementação e manutenção das aplicações. Pode-se dizer que existem duas filosofias por trás do RoR: "*Don't Repeat Yourself (DRY)*" e "*Convention over Configuration*". Na primeira, a ideia é que um pedaço de código deve ser utilizado apenas em um lugar, evitando repetição desnecessária. Na segunda, utiliza-se uma série de convenções adotadas pelo *framework* a fim de reduzir a quantidade de código utilizada (RUBY; THOMAS; HANSSON, 2013).

A arquitetura do *Ruby on Rails* é dividida em módulos que servem a funcionalidades distintas dentro do contexto da aplicação. Para o padrão MVC, são utilizados três módulos que servem a cada uma das camadas definidas acima, como mostra a Figura 3: para as *views*, utiliza-se o módulo ***ActionView***, que provê funcionalidades para a exibição das páginas, construção de formulários, etc. Para os *models*, o módulo ***ActiveRecord*** garante a associação entre as classes do sistema, bem como o relacionamento da aplicação com o banco de dados. Por fim, mas não menos importante, os *controllers* pertencem ao módulo ***ActionController*** (ou simplesmente *Controller*), o qual provê um ambiente intermediário entre os *models* e as *views*, garantindo que a informação correta seja exibida ao usuário.

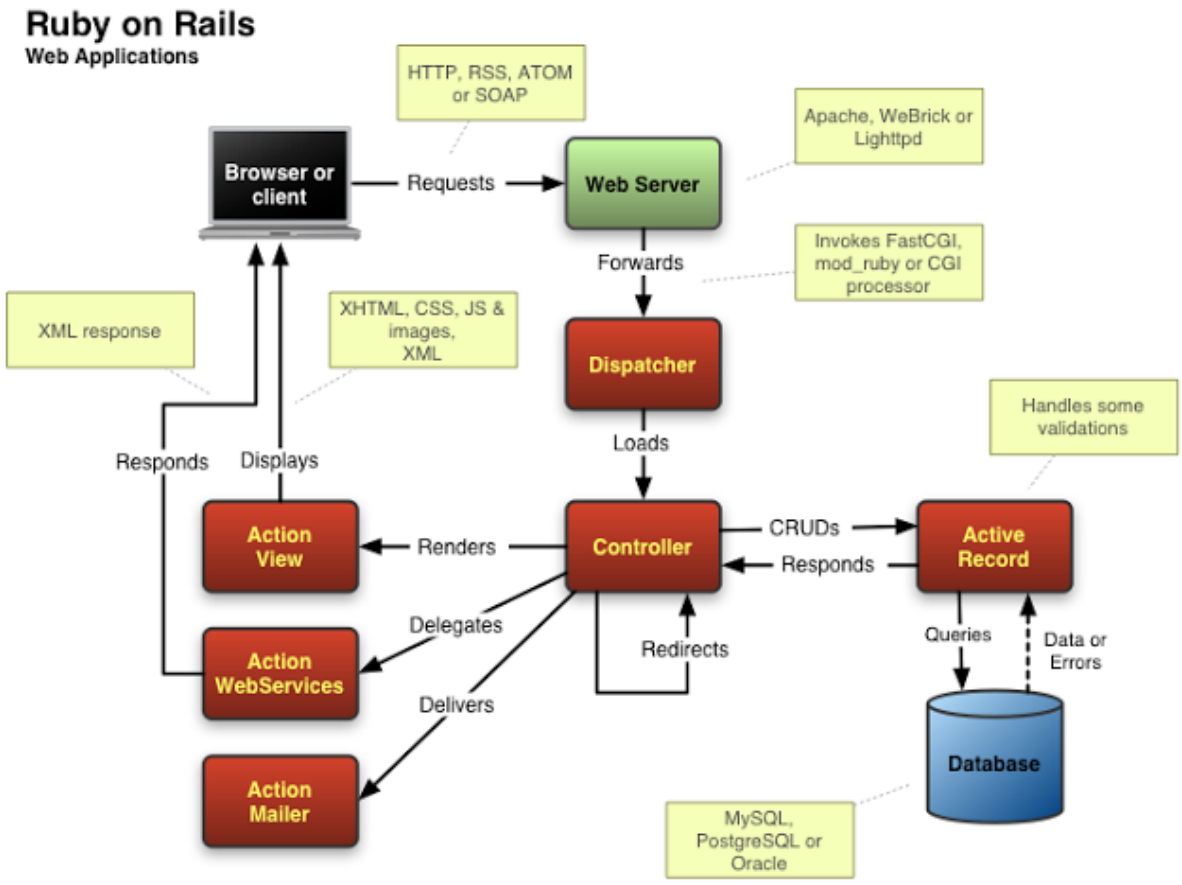


Figura 3 – Arquitetura do Ruby on Rails (MEJIA, 2011).

## 2.3 Test Driven Development (TDD)

A utilização de *Test Driven Development* se tornou uma prática comum na comunidade de desenvolvimento de software. A técnica consiste em escrever testes antes de desenvolver o produto propriamente dito, agregando muitos benefícios ao processo de desenvolvimento. O primeiro deles, e mais claro, são os benefícios na qualidade externa do produto, dando mais segurança ao desenvolvedor na hora de mudanças.

Os testes automatizados, que rodam em questão de segundos, são executados o tempo todo pelo desenvolvedor. Ao contrário dos testes manuais, caso algo pare de funcionar, o desenvolvedor é rapidamente notificado e consegue corrigir o problema de imediato. Além disso, pode-se dizer que um desenvolvedor, ao praticar TDD, divide seu trabalho em pequenas etapas: escreve-se um pequeno teste, implementa-se um pedaço da funcionalidade e esse ciclo se repete até o fim do desenvolvimento. A cada teste escrito (e executado), o desenvolvedor ganha *feedback* dos erros do sistema e, além disso, ganha um ponto de partida para a análise do erro e sua possível solução, como pode ser visualizado na Figura 4.

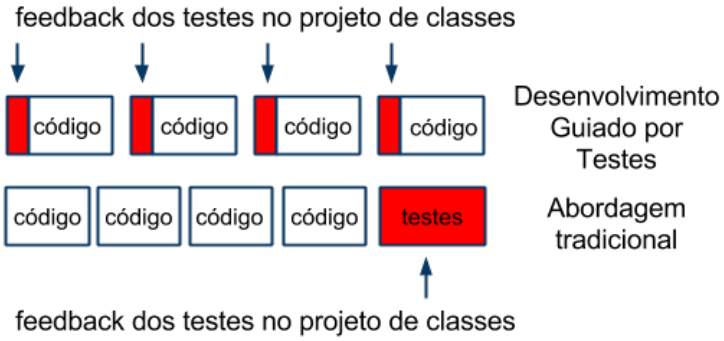


Figura 4 – Comparação da abordagem tradicional de desenvolvimento de software com o desenvolvimento guiado por testes (ANICHE, 2014)

Sempre que um desenvolvedor pega uma funcionalidade para fazer, ele a quebra em pequenas tarefas. Tais tarefas exigem a escrita de código: classes são criadas enquanto outras são modificadas. Ao praticar TDD, o desenvolvedor antes de começar a fazer essas modificações explicita esses objetivos por meio de testes automatizados. O teste em código nada mais é do que um trecho de código que deixa claro o que determinado trecho de código deve fazer. Ao formalizar esse objetivo na forma de um teste automatizado, esse teste falha, claro; afinal, a funcionalidade ainda não foi implementada. O desenvolvedor então trabalha para fazer esse teste passar implementando a funcionalidade da forma que julgar necessária (ANICHE, 2014).

## 2.4 O método FrameWeb

FrameWeb é um método de projeto para construção de sistemas de informação Web (Web Information Systems – WISs) baseado em *frameworks*. O método assume que determinados tipos de frameworks serão utilizados durante a construção da aplicação, define uma arquitetura básica para o WIS e propõe modelos de projeto que se aproximam da implementação do sistema usando esses *frameworks* (SOUZA, 2007).

O FrameWeb define uma arquitetura lógica padrão para WISs baseada no padrão arquitetônico *Service Layer* (Camada de Serviço) (FOWLER, 2002). O sistema é dividido em três grandes camadas (SALVATORE, 2016):

- **Camada de Negócio *(Business Tier)*:** responsável pelas funcionalidades relacionadas às regras de negócio da aplicação. Esta camada é particionada em duas: Lógica de Domínio (*Domain*) e Lógica de Aplicação (*Application*);
- **Camada de Apresentação *(Presentation Tier)*:** responsável por funcionalidades de interface com o usuário (incluindo as telas que serão apresentadas a ele). Esta camada, segundo o padrão proposto, é também particionada em duas outras: Visão (*View*) e Controle (*Control*);
- **Camada de Acesso a Dados *(Data Access Tier)*:** responsável por funcionalidades relacionadas à persistência de dados.

A fase de Projeto concentra as propostas principais do método: (i) definição de uma arquitetura padrão que divide o sistema em camadas, de modo a se integrar bem com os *frameworks* utilizados; (ii) proposta de um conjunto de modelos de projeto que trazem conceitos utilizados pelos *frameworks* para esta fase do processo por meio da definição de uma linguagem específica de domínio que faz com que os diagramas fiquem mais próximos da implementação (SOUZA, 2007; MARTINS, 2016).

Para representar componentes típicos da plataforma Web e dos *frameworks* utilizados, o FrameWeb estende o meta-modelo da UML, especificando, assim, uma sintaxe própria. Com isso, é possível utilizá-lo para a construção de diagramas de quatro tipos:

- **Modelo de Entidades:** é um diagrama de classes da UML que representa os objetos de domínio do problema e seu mapeamento para a persistência em banco de dados relacional;
- **Modelo de Persistência:** é um diagrama de classes da UML que representa as classes DAO existentes, responsáveis pela persistência das instâncias das classes de domínio.

- **Modelo de Navegação:** é um diagrama de classe da UML que representa os diferentes componentes que formam a camada de Lógica de Apresentação, como páginas Web, formulários HTML e *controllers*.
- **Modelo de Aplicação:** é um diagrama de classes da UML que representa as classes de serviço, que são responsáveis pela codificação dos casos de uso, e suas dependências.

# 3 Especificação de Requisitos

Este capítulo aborda alguns resultados da Engenharia de Requisitos para a construção do sistema Khoeus. Na seção 3.1, é apresentado o minimundo do projeto; na seção 3.2, são apresentados os diagramas de pacotes, na seção 3.3, são apresentados diagramas de casos de uso e na seção 3.4, são apresentados os diagramas de classes. Os requisitos funcionais, requisitos não funcionais e regras de negócio podem ser encontrados no **Documento de Especificação de Requisitos** que está disponível no Apêndice ao final dessa monografia.

## 3.1 Descrição do Escopo

Em um ambiente de aprendizagem como uma escola ou universidade, espera-se que os alunos tenham acesso fácil e direto às informações relativas às aulas, como materiais de aula, notas das atividades realizadas e lista de presença. Para isso, espera-se centralizar essas funcionalidade no formato de turmas, de forma que cada uma delas possua seus alunos, professores, atividades, dentre outros elementos envolvidos no sistema.

### 3.1.1 Organização de turmas

A uma turma, estarão vinculados uma série de usuários que poderão exercer papel (*role*) de professor ou de aluno. Para participar de uma turma, um usuário deverá visitá-la e, em seu primeiro acesso, inserir a chave de acesso (que foi definida no momento em que a turma foi criada). Caso a chave esteja correta, o usuário é associado àquela turma e pode acessar seu conteúdo. Além disso, cada turma contém um *board*, que corresponde a todos os itens que serão exibidos aos alunos, como tarefas, arquivos, URLs, dentre outros (descritos na seção 3.1.3). Para melhor organizá-lo, um *board* é subdividido em seções definidas pelo professor e todos os itens deverão estar associados a uma seção. Um *board* contém obrigatoriamente:

- Uma **lista de notícias** que serão criadas exclusivamente pelos professores da turma. As notícias não possuem réplica;
- Um **fórum de discussão** com tópicos que podem ser tanto criados quanto respondidos por professores ou alunos da turma;
- Uma **lista de *LiveQuestions*** com perguntas que os alunos podem realizar durante as aulas;
- A **lista de presença** dos alunos da turma;

- O **livro de notas** contendo a avaliação de cada aluno em cada uma das atividades realizadas.

Além disso, o *board* também poderá conter:

- **Arquivos:** permite realizar o download de um arquivo que foi enviado pelo professor;
- **Links:** permite acessar páginas externas;
- **Questionários:** permite responder a perguntas objetivas sem que existam opções corretas, simulando uma enquete;
- **Provas:** permite responder a perguntas objetivas e discursivas que serão avaliadas;
- **Tarefas:** permite enviar arquivos, textos ou códigos de programação de acordo com a tarefa passada pelo professor;
- **Atividades Externas:** permite avaliar uma atividade que tenha sido realizada fora da plataforma, como uma atividade feita em sala de aula, por exemplo.

### 3.1.2 Controle de usuários

Todas as funcionalidades do sistema dependem do usuário estar logado em sua conta. Para isso, o Khoeus conta com uma página de login, uma de cadastro e uma de recuperação de senha, permitindo que os usuários possam acessar o sistema. Na página de cadastro, o usuário deverá informar seus dados pessoais (como nome, endereço e telefone), seu e-mail (que será utilizado para realizar o login) e uma senha. Feito isso, um e-mail de confirmação será enviado para o e-mail cadastrado a fim de que o usuário confirme seu cadastro. Na página de login, basta inserir o e-mail e senha para que seja verificada sua autenticidade. Caso estejam corretos, o usuário é redirecionado para a lista de turmas. Caso contrário, deverá inserir novamente seus dados. Na página de recuperação de senha, basta que o usuário digite o e-mail cadastrado em sua conta para que receba um link em sua caixa de entrada permitindo a mudança de senha.

Um usuário do sistema pode ter permissão de **Administrador** ou de **Membro**. Enquanto os membros podem apenas alterar o próprio perfil, ingressar em uma turma ou acessar uma turma já ingressada anteriormente, administradores têm acesso completo ao sistema e podem alterar as configurações gerais do sistema, das turmas e editar o perfil de todos os usuários. Para o sistema, espera-se que seja possível configurar o fuso horário utilizado pelo sistema, definir as informações de SMTP para disparo de e-mails, colocar o sistema em modo de manutenção (acessível apenas para administradores). Para as turmas, espera-se poder definir seu título, sua senha de acesso, definir se a turma possui livro de notas e lista de presença além de ser possível definir as categorias de nota que

serão utilizadas para a avaliação dos alunos (mais detalhes na Seção 3.1.4). Já para os alunos, espera-se ser possível alterar seu perfil (nome, senha, endereço, etc.). Além disso, os administradores têm acesso a um registro com logs de todo o sistema, incluindo todas as ações realizadas pelos usuários, como cadastro, login, criação de itens de board e até mesmo submissão de novas tarefas.

Além das permissões vinculadas à conta, os usuários também possuem um papel (*role*) em cada turma da qual fazem parte, podendo este ser de **Professor** ou **Aluno**. Assim, um usuário com a permissão de membro pode assumir a posição de professor ou de aluno dentro de uma turma. Em uma turma, um aluno poderá consultar os itens do *board*, como enviar tarefas e baixar arquivos, consultar as próprias presenças na lista de chamada, consultar suas notas no livro de notas, verificar o calendário de aulas planejadas, criar novas *Live Questions* e enviar mensagens para alunos e professores. Paralelamente a isso, um professor pode inserir novos itens no *board*, lançar presenças e notas dos alunos cadastrados na turma, inserir uma nova aula no calendário, verificar as *Live Questions* vigentes, enviar mensagem para alunos e professores, realizar download em lote dos envios de uma tarefa, prova ou questionário, visualizar o log de atividades dos alunos e definir as configurações da turma, já descritas anteriormente.

Vale ressaltar que um usuário pode ter sua *role* alterada de acordo com a vontade de um administrador. Dessa forma, é possível que um Membro torne-se Administrador (ou vice-versa) e que um Aluno vire Professor (ou vice-versa).

### 3.1.3 Itens do Board

Itens do *board* estão sempre associados a uma seção com exceção do fórum de discussões, das notícias, das *Live Questions*, da lista de presença e do livro de notas, que são fixados no topo do *board*. Cada item do *board* possui uma ação diferente e possui campos diferentes no momento da inclusão.

#### 3.1.3.1 Arquivo

Ao acessar um arquivo, o aluno realiza o download do mesmo. Para criar um novo arquivo, o professor deverá informar seu nome, a descrição e deverá fazer o upload a partir dos arquivos de seu computador.

#### 3.1.3.2 Link

Ao acessar um link, o aluno é redirecionado para a URL correspondente. Para inserir um novo link no board, o professor deverá informar o título do link, sua descrição e a sua respectiva URL.

#### 3.1.3.3 Notícias

No topo do board, o aluno pode acessar a lista de notícias relacionada àquela turma e onde estão todas as notícias já criadas pelos professores da turma. Para inserir uma nova noticia, o professor deverá informar apenas o título e o conteúdo desejado.

#### 3.1.3.4 Fórum de Discussão

No fórum de discussão, tanto alunos quanto professores podem criar novos tópicos e, da mesma forma, responder tópicos já criados anteriormente. Para criar um novo tópico, espera-se que o usuário insira um título e seu conteúdo. No caso da réplica de um tópico, basta seu conteúdo.

#### 3.1.3.5 *Live Questions*

Na lista de *Live Questions*, alunos poderão criar novas dúvidas e professores poderão visualizá-las por até 24h após sua criação. A ideia é que as dúvidas sejam criadas e sanadas no momento da aula. Na sua criação, o aluno deve inserir sua dúvida e informar se deseja ser tratado como Anônimo ou não.

#### 3.1.3.6 Questionário

Os questionários simulam, basicamente, uma enquete ou uma pesquisa de opinião: no momento da criação, o professor define uma série de perguntas (todas de múltipla escolha) e as respostas possíveis para cada uma delas, sendo possível marcar apenas uma das alternativas. Nesse item, não existe resposta correta e, por isso, os questionários devem ser utilizados para recolher a opinião dos alunos sobre algum assunto determinado. Ao acessar um questionário, os alunos visualizarão as perguntas que foram criadas pelo professor e deverão respondê-las, não sendo possível responder a um questionário mais de uma vez. Questionários possuem uma data para iniciar e um tempo-limite para que possam ser respondidos. Após isso, é possível coletar os resultados do questionário.

#### 3.1.3.7 Prova

Uma prova, como o próprio nome já diz, é elegível para ser uma forma de avaliação do aluno. Dessa forma, no momento da criação, o professor deve associá-la a uma categoria de nota e informar um título e uma descrição. Provas possuem uma data para iniciar e uma data de término que também deverão ser informadas. Além disso, o professor deve inserir quais perguntas haverá na prova, o tipo da questão (discursiva ou objetiva) e qual a nota máxima para cada uma das questões (totalizando 100 pontos). No caso de perguntas objetivas, deve-se informar qual das alternativas é a correta. Após encerrado o prazo da prova, os professores poderão visualizar as respostas dos alunos para cada uma das

questões e atribuir a elas uma nota variando de 0 à nota máxima para aquela questão, sendo possível também informar um feedback para cada questão.

#### 3.1.3.8 Tarefa

Tarefas representam a principal forma de avaliar um aluno, uma vez que permitem o envio de textos, arquivos ou até mesmo códigos de programas. No momento da criação da tarefa, o professor deverá informar o tipo de tarefa, a que categoria de nota esta pertence, título, descrição, data de início e data limite de envio. Uma vez finalizada, o professor poderá avaliar a tarefa de cada aluno concedendo-o uma nota de 0 a 100 e sendo possível ainda dar um feedback em forma de texto ou comentar linhas do código enviado pelo aluno.

A forma de submissão da tarefa varia de acordo com o seu tipo: caso seja do tipo texto, o aluno deverá preencher um campo de texto com o que foi pedido na descrição da tarefa. Caso seja do tipo arquivo, o aluno deverá fazer o upload do arquivo que foi requisitado. No caso da tarefa ser do tipo código, ele deverá inseri-lo no campo de texto, podendo realizar a compilação em tempo real a fim de verificar a corretude de seu código.

#### 3.1.3.9 Atividade Externa

Atividades externas representam exercícios ou provas que foram feitas em sala e que, por isso, não são avaliados diretamente no Khoeus. Dessa forma, alunos não precisarão submeter quaisquer informações neste tipo de item (uma vez que esta foi feita presencialmente), mas o professor poderá lançar as notas de cada um dos alunos e incluir um pequeno feedback a respeito da atividade.

Atividades externas também devem estar vinculadas a uma categoria de nota.

### 3.1.4 Avaliação

Dentro das configurações de uma turma, é possível definir a forma com que a média final é calculada e quais as condições de aprovação dos alunos. Dessa forma, professores podem criar categorias de nota e associar a cada uma delas um peso no cálculo da média final. É possível criar, por exemplo, as categorias de nota “Prova” e “Trabalho” e definir a média final como 60% da nota de prova somado a 40% da nota de trabalho.

Para que a média seja calculada corretamente, o professor deverá definir no momento da criação de uma **Tarefa**, **Prova** ou **Atividade Externa** a que categoria de nota ela pertence.

## 3.2 Diagrama de Pacotes

Dada a descrição do minimundo, foi possível estruturar o sistema em sete pacotes, como pode ser visto na Figura 5. A descrição de cada um dos subsistemas está contida na Tabela 1.

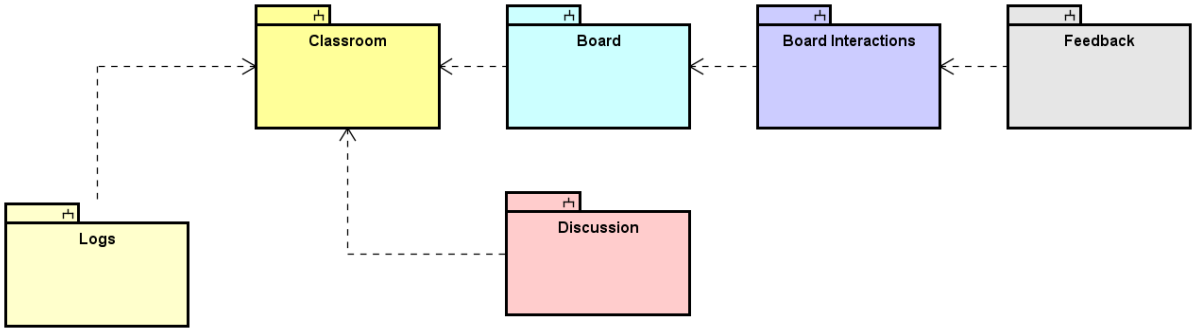


Figura 5 – Diagrama de pacotes para o sistema Khoeus

Tabela 1 – Descrição dos subsistemas

| Subsistema | Descrição |
|---|---|
| Classroom | Subsistema contendo as funcionalidades relacionadas diretamente às turmas. |
| Board | Envolve todas as funcionalidades relativas ao gerenciamento dos itens do board. |
| Board Interactions | Corresponde a todas as interações que os usuários podem realizar com os itens do board, como submeter tarefas ou realizar provas. |
| Feedback | Permite a avaliação de atividades e a visualização das notas do aluno em uma turma. |
| Discussions | Subsistema contendo as funcionalidades relativas ao fórum de discussão, às notícias e às *Live Questions*. |
| Logs | Este sistema consiste na visualização de logs gerados a partir das atividades dos usuários do sistema. |

## 3.3 Diagrama de Casos de Uso

Como visto na Seção 3.2, este projeto foi divido em sete subsistemas. Na Subseção 3.3.2 os casos de uso do subsistema `Classroom`, na Subseção 3.3.3 os casos de uso do subsistema `Board`, na Subseção 3.3.4 os casos de uso do subsistema `Board Interactions`, na Subseção 3.3.5 os casos de uso do subsistema `Feedback`, na Subseção 3.3.6 os casos de uso do subsistema `Discussion` e na Subseção 3.3.7 os casos de uso do subsistema `Log`.

### 3.3.1 Atores x Casos de Uso

O modelo de casos de uso visa capturar e descrever as funcionalidades que um sistema deve prover para os atores que interagem com o mesmo. No contexto deste projeto, temos basicamente dois tipos de atores: aqueles relacionados ao papel do usuário no sistema (Visitante, Administrador ou Membro) e aqueles relacionados ao papel do usuário dentro de uma turma (Professor ou Aluno).

Um Membro pode ser um professor ou aluno e, por isso, utilizou-se um mecanismo de controle de acesso baseado em *roles* (papéis) a fim de evitar que alunos possam, por exemplo, configurar turmas ou criar itens do board. Dessa forma, cada membro do sistema possui um papel associado a ele dependendo da turma em que está inscrito.

Maiores informações e detalhes sobre os casos de uso poderão ser consultados no **Documento de Análise de Requisitos** que está disponível no Apêndice ao final dessa monografia.

### 3.3.2 Subsistema Classroom

A Figura 6 mostra os casos de uso do subsistema `Classroom` que serão descritos a seguir. Esta divisão do sistema foi criada para prover funcionalidades relacionadas diretamente às turmas, como é o caso dos casos de uso **Configurar turma**, **Ingressar em uma turma**, **Definir roles de usuários na turma** e **Gerenciar turmas**, sendo esse último um caso de uso cadastral, assegurando as operações básicas de CRUD.

Praticamente todas as funcionalidades do sistema giram em torno da inscrição dos usuários em determinadas turmas e, por isso, o caso de uso para **Ingressar em uma turma** espera que um usuário tenha acesso ao conteúdo da turma desejada após inserida a senha estipulada no momento de sua criação. Criação esta que está contida no caso de uso cadastral **Gerenciar turmas** que fornece as operações básicas de CRUD para essa classe do sistema. Criada uma turma, professores também podem alterar suas configurações por meio do caso de uso **Configurar turma**. Uma vez inscrito em uma turma, um usuário pode assumir os papéis de aluno ou professor. Esse controle do papel de um usuário é gerenciado pelo caso de uso **Definir roles de usuários na turma**.

Além disso, esse subsistema também provê as funcionalidades básicas de uma turma e que não dependem de seu board. De um lado, professores podem **Gerenciar eventos no calendário**, adicionando-os na data prevista e lançar a presença dos alunos nas aulas definidas no calendário. Do outro lado, alunos podem consultar suas presenças ao longo das aulas e os eventos registrados no calendário. Para facilitar a comunicação entre membros de uma mesma turma, o caso de uso **Enviar mensagens** permite que alunos e professores troquem mensagens entre eles.

Por fim, o caso de uso **Definir configurações do sistema** permite que adminis-

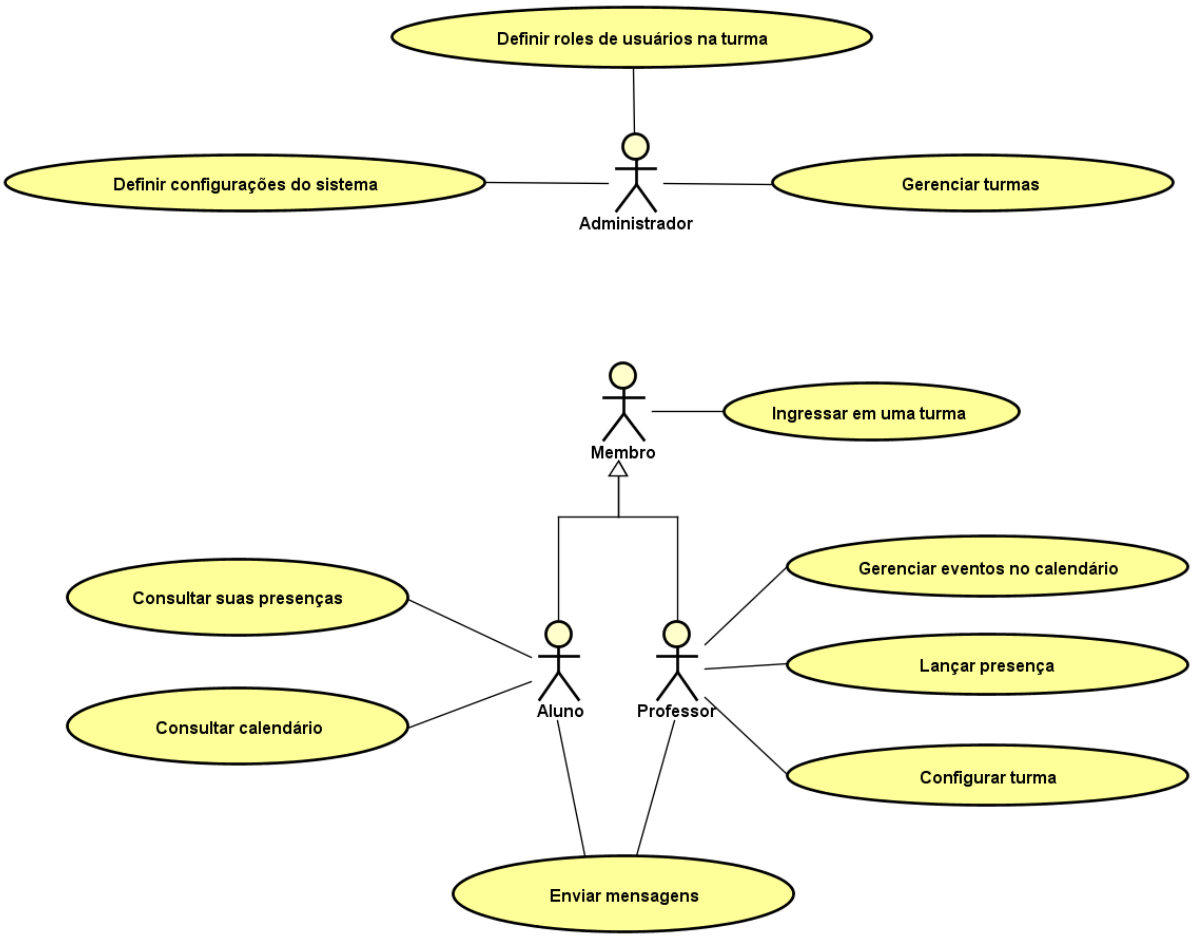


Figura 6 – Diagrama de Casos de Uso para o subsistema Classroom.

tradores definam diversas variáveis que irão reger o sistema, como definições de fuso-horário, definições SMTP e modo de manutenção.

### 3.3.3 Subsistema Board

A Figura 7 mostra os casos de uso do subsistema `Board` que serão descritos a seguir. Esta divisão do sistema foi criada para prover o gerenciamento dos itens do board de uma turma por meio dos casos de uso **Gerenciar seções do board**, **Gerenciar arquivos do board**, **Gerenciar links do board**, **Gerenciar questionários do board**, **Gerenciar provas do board**, **Gerenciar tarefas do board** e **Gerenciar atividades externas do board**.

Para cada caso de uso listado, espera-se que um professor possa criar, alterar, consultar e excluir a classe em questão, passando as informações relativas a ela no momento de sua criação ou alteração. Além disso, com exceção do caso de uso **Gerenciar seções do board**, deve-se informar a seção a que cada item do board pertence no momento de sua criação.

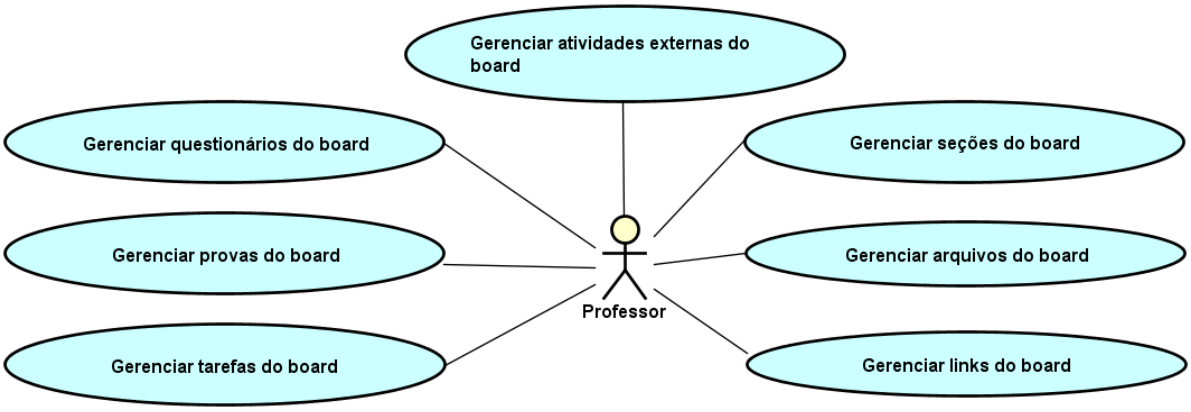


Figura 7 – Diagrama de Casos de Uso para o subsistema Board.

### 3.3.4 Subsistema Board Interactions

A Figura 8 mostra os casos de uso do subsistema **`Board Interactions`** que serão descritos a seguir. Esta divisão do sistema foi criada para prover as funcionalidades em que alunos possam interagir com os itens de board criados pelo subsistema **`Board`**.

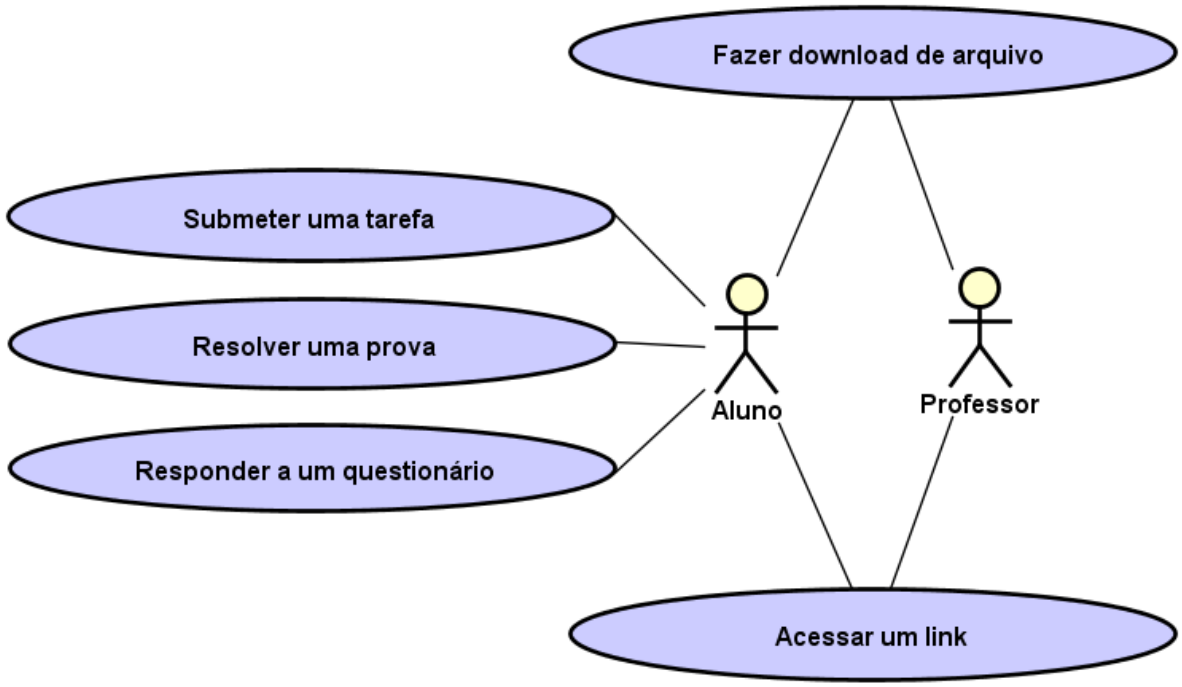


Figura 8 – Diagrama de Casos de Uso para o subsistema Board Interactions.

Nesse subsistema, cada tipo de item de board possui uma interação diferente com o usuário. Para o caso de uso **Acessar um link**, o usuário simplesmente é redirecionado para a página em questão e para o caso de uso **Fazer download de arquivo**, inicia-se o download do arquivo que foi inserido no sistema.

Para os casos de uso **Responder a um questionário** e **Resolver uma prova**, o usuário deverá responder a uma série de questões (objetivas no caso de questionários e

objetivas ou discursivas no caso de provas), submetendo suas respostas ao final do processo. Para **Submeter uma tarefa**, o aluno poderá submeter tarefas que sejam no formato texto, arquivo ou código.

Professores poderão **Visualizar resultado de questionários** uma vez que tenha se passado a data limite para respondê-lo, sendo possível visualizar seus resultados de forma resumida por meio de porcentagens, gráficos, taxas de resposta etc.

### 3.3.5 Subsistema Feedback

A Figura 9 mostra os casos de uso do subsistema **`Feedback`** que serão descritos a seguir. Esta divisão do sistema foi criada para permitir que professores avaliem as provas, tarefas e atividades externas dos alunos e para que estes possam consultar suas notas e o feedback atribuído às suas atividades.

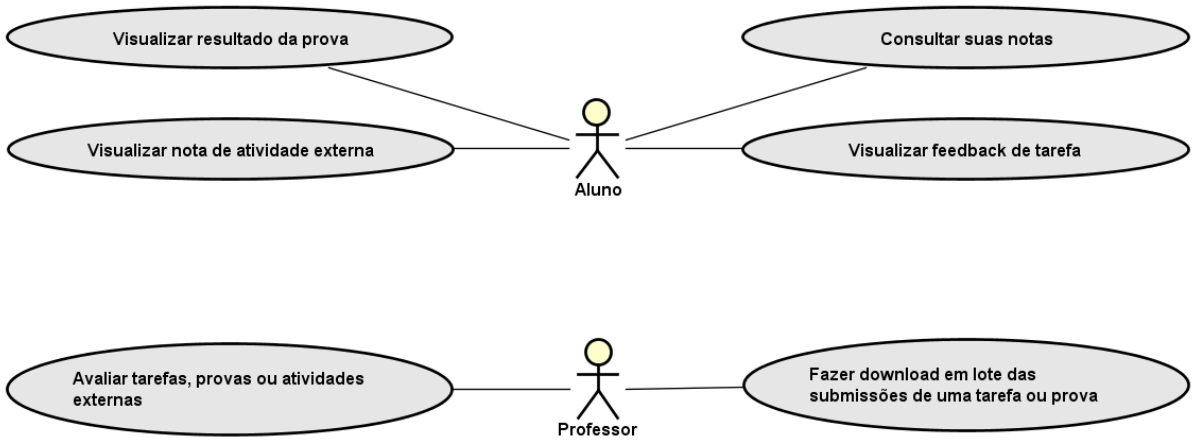


Figura 9 – Diagrama de Casos de Uso para o subsistema Feedback.

Nesse subsistema, tem-se dois tipos diferentes de casos de usos: (i) casos de uso de professores para avaliação de atividades. Para facilitar o processo de avaliação, é possível **Fazer download em lote das submissões de uma tarefa ou prova** por meio de um arquivo compactado. Além disso, é possível **Avaliar tarefas, provas ou atividades externas individualmente**, concedendo uma nota e opcionalmente um feedback para cada um dos alunos ou **Avaliar tarefas, provas ou atividades externas em lote** por meio do *upload* de uma planilha contendo as notas e feedbacks de todos os alunos daquela turma; (ii) casos de uso de alunos para consulta de notas de forma que eles possam **Visualizar resultado de provas**, **Visualizar nota de atividade externa** e **Visualizar feedback de tarefa**. Além disso, o aluno tem a sua disposição um livro de notas onde pode **Consultas suas notas** ao longo do curso.

### 3.3.6 Subsistema Discussion

A Figura 10 mostra os casos de uso do subsistema **`Discussion`** que serão descritos a seguir. Esta divisão do sistema foi criada para permitir que professores e alunos troquem informações por meio do fórum, de notícias e de *Live Questions*.

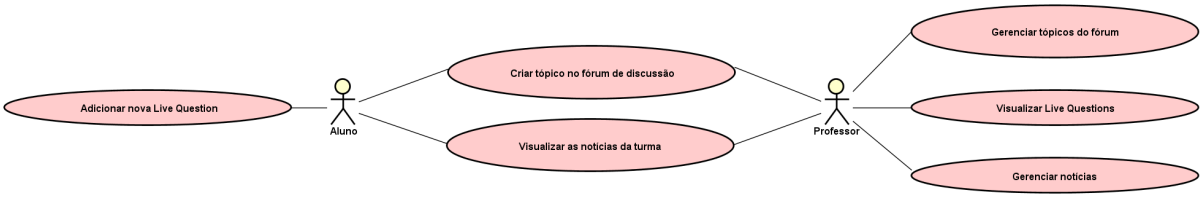


Figura 10 – Diagrama de Casos de Uso para o subsistema Discussion.

O fórum de discussão tem o intuito de ser um ambiente livre para que alunos e professores possam discutir temas de seu interesse e, por isso, ambos podem **Criar tópico no fórum de discussão**, **Responder a um tópico no fórum de discussão** e **Visualizar os tópicos do fórum de discussão**. No caso dos dois primeiros, basta que o usuário digite o conteúdo da mensagem e, no caso da criação de tópicos, fornecer um título. Para manter o controle das publicações, professores podem **Gerenciar tópico do fórum** de forma que podem realizar as operações básicas de CRUD sobre eles quando julgarem necessário.

As notícias de uma turma dizem respeito exclusivamente a informações que os professores desejam passar para seus alunos e, por isso, apenas professores podem **Gerenciar notícias**, podendo criar, alterar, consultar e excluí-las quando julgar necessário. Embora não possam criá-las, alunos podem **Visualizar as notícias da turma**.

As *Live Questions* devem ser usadas para realizar perguntas para os professores no momento da aula. O fluxo consiste no fato do aluno **Adicionar nova Live Question** enquanto o professor poderá **Visualizar Live Questions**. Como o intuito é de que essas perguntas sejam feitas em momento de aula, elas serão excluídas passadas 24 horas de sua criação, conforme Figura 11.

### 3.3.7 Subsistema Log

A Figura 12 mostra os casos de uso do subsistema **`Log`** que serão descritos a seguir. Esta divisão do sistema foi criada para garantir um controle do que foi feito dentro do sistema e quem o fez.

Basicamente, o sistema possui duas categorias de log: como o papel de um professor está restrito à sua turma, só é possível **Visualiar logs dos alunos** daquela turma em que estão inseridos. Paralelamente a isso, administradores podem **Visualiar todos os logs do sistema** a fim de garantir uma espécie de "auditoria", podendo tomar atitudes caso sejam necessárias.

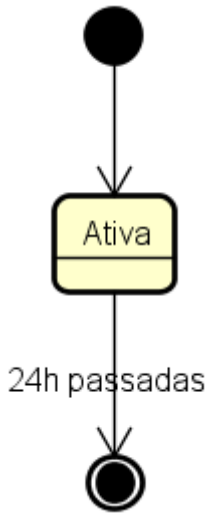


Figura 11 – Diagrama de Estados para uma LiveQuestion.

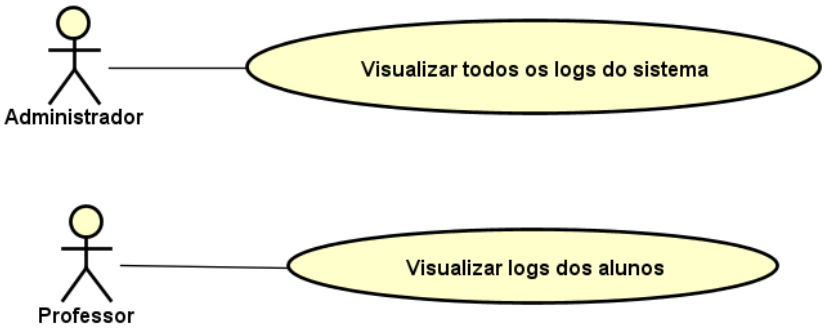


Figura 12 – Diagrama de Casos de Uso para o subsistema Log.

## 3.4 Diagrama de Classes

Assim como os casos de uso na Seção 3.3, os diagramas de classes estão divididos de acordo com a divisão dos subsistemas. Na Subseção 3.4.1 estão as classes pertecentes ao subsistema `Classroom`, na Subseção 3.4.2 estão as classes pertecentes ao subsistema `Board`, na Subseção 3.4.3 estão as classes pertecentes ao subsistema `Board Interactions`, na Subseção 3.4.4 estão as classes pertecentes ao subsistema `Feedback`, na Subseção 3.4.5 estão as classes pertecentes ao subsistema `Discussion` e na Subseção 3.4.6 estão as classes pertecentes ao subsistema `Log`.

Vale ressaltar que algumas associações são consideradas obrigatórias nos dois sentidos pois considera-se que o tempo decorrido entre a criação de ambas é extremamente pequeno. Além disso, por estarmos considerando **entidades**, uma só faria sentido quando estivesse associado à outra.

### 3.4.1 Subsistema Classroom

A Figura 13 exibe o diagrama de classes do subsistema **`Classroom`**.

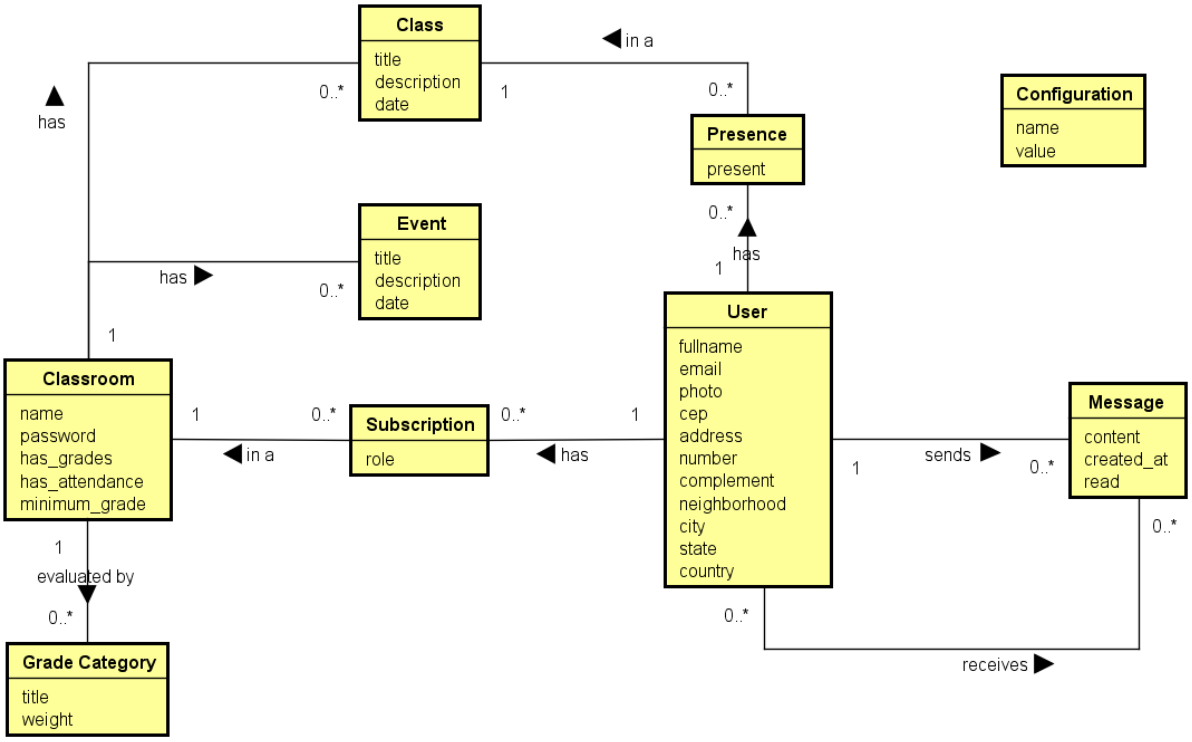


Figura 13 – Diagrama de Classes do subsistema Classroom.

Aqui, fica evidente a relação de `User` com `Classroom` por meio da classe `Subscription`: cada usuário poderá estar inscrito em uma turma, assumindo um determinado papel (*role*). Dentro de uma turma, existe um calendário com os eventos (`Events`) previstos e as aulas (`Class`) que foram ou serão lecionadas, sendo que pra cada delas o professor poderá lançar a presença ou falta de cada aluno por meio da classe `Presence`.

Aqui, vale notar que as turmas podem não estar associadas a nenhuma presença (por meio das aulas) ou categorias de nota, já que o administrador terá a opção de definir sua existência por meio dos atributos `has_grades` e `has_attendance`.

Para que seja possível verificar se alunos foram aprovados ou reprovados, cada turma terá um conjunto de `Grade Category` que especificará a forma com que a média final será calculada. Além disso, a classe `Message` permite que alunos possam trocar mensagens entre si ou com os professores daquela turma.

Por fim, a fim de garantir a consistência do sistema, a classe `Configuration` permite que administradores realizem as configurações necessárias para seu bom funcionamento, como configurações para envio de e-mail e idioma de preferência.

### 3.4.2 Subsistema Board

A Figura 14 exibe o diagrama de classes do subsistema **`Board`**.

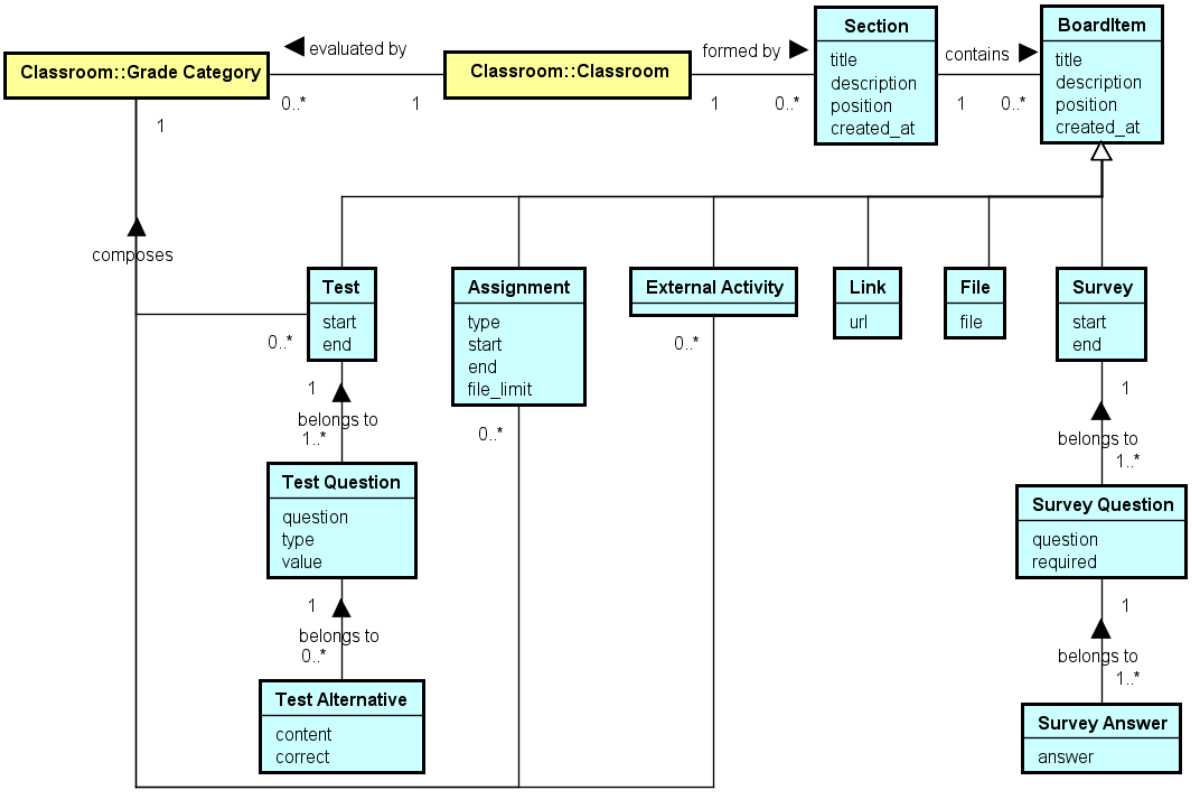


Figura 14 – Diagrama de Classes do subsistema Board.

Nesse subsistema estão contidas as classes relacionadas aos diversos itens do *board* da turma. Para melhor organização, todos os `BoardItens` estão organizados em `Sections`, agrupando itens relacionados entre si. `BoardItens` podem ser de 6 tipos diferentes: `Link`, `File` (Arquivo), `Survey` (Questionário), `Test` (Prova), `Assignment` (Tarefa) e `External Activity` (Atividade Externa). Aqui, fica evidente que os itens do board precisam estar, necessariamente, associados a uma seção da turma.

`Surveys` conterão exclusivamente perguntas objetivas e, por isso, possuem associadas a ela, estão as `Survey Questions` que correspondem a uma pergunta de determinado questionário. Associada a estas últimas estão as `Survey Answers`, que representam as possíveis escolhas para aquela questão.

Da mesma forma, `Tests` possuem associados a eles uma série de `Test Questions` que podem ser discursivas ou objetivas (determinadas pelo atributo `type`). No caso das questões objetivas, a questão possui um conjunto de `Test Alternatives` associadas a ela. A alternativa correta da questão é definida pelo atributo `correct`.

### 3.4.3 Subsistema Board Interactions

A Figura 15 exibe o diagrama de classes do subsistema **`Board Interactions`**.

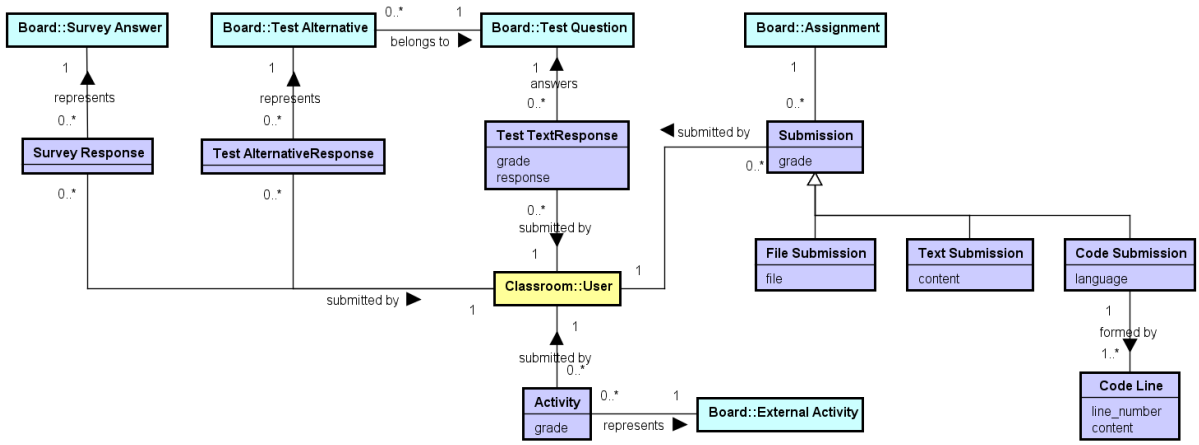


Figura 15 – Diagrama de Classes do subsistema Board Interactions.

De forma complementar ao **Board**, esse subsistema representa as interações dos alunos com os itens do board. As classes `Survey Response` e `Test AlternativeResponse` armazenam a escolha de uma alternativa das questões objetivas de questionários e provas. Além disso, a classe `Test TextResponse` armazena a resposta de um aluno para questões discursivas, bem como a nota atribuída pelo professor.

Tarefas podem ser submetidas por meio de textos, arquivos ou códigos. `Text Submission`, `File Submission` e `Code Submission` representam as submissões do aluno para cada um desses tipos de tarefa. No caso das submissões de código, este é armazenado linha a linha por meio da classe `Code Line`.

Por fim, a classe `Activity` representa a atividade externa desenvolvida por um aluno e a nota atribuída a ela.

### 3.4.4 Subsistema Feedback

A Figura 16 exibe o diagrama de classes do subsistema **`Feedback`**.

A fim de permitir que professores avaliem as atividades de um aluno e deem um feedback a eles, esse subsistema reúne as classes `Text Feedback` e `Code Line Feedback`. A primeira delas permite que professores atribuam um texto avaliando a prova, tarefa ou atividade externa do aluno enquanto a última permite que o professor atribua comentários para linhas de código da tarefa de um aluno.

### 3.4.5 Subsistema Discussion

A Figura 17 exibe o diagrama de classes do subsistema **`Discussion`**.

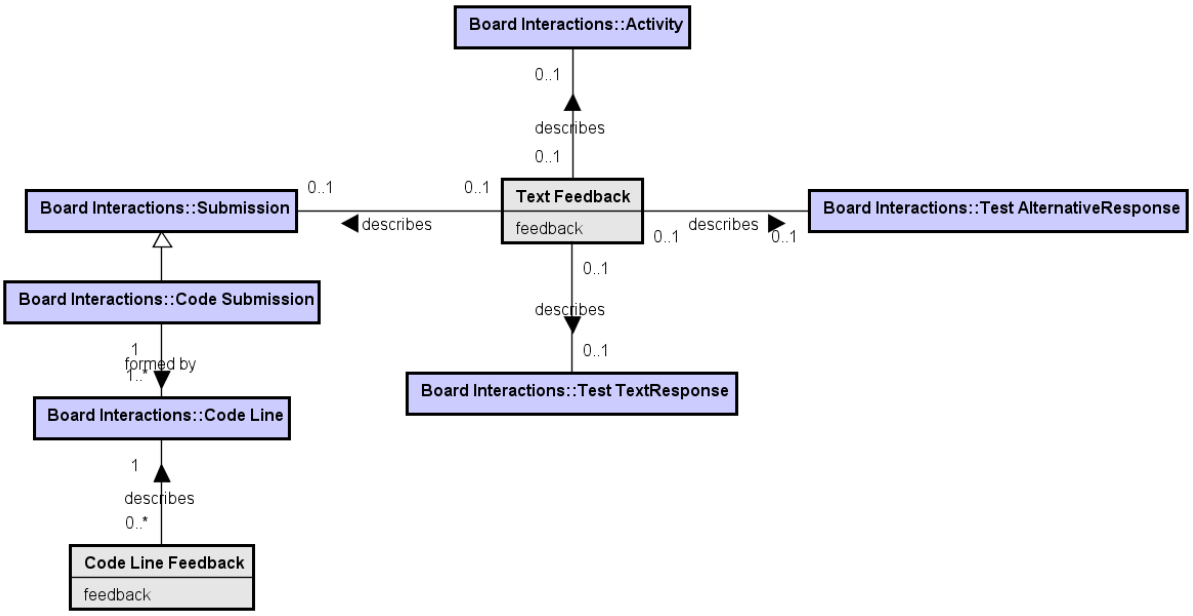


Figura 16 – Diagrama de Classes do subsistema Feedback.

Dentro de cada turma existe um ambiente de discussão formado por um fórum de discussões, uma lista de notícia e o grupo de *Live Questions*. Um fórum de discussões é formado por `Forum Topics` que podem ser criados e respondidos tanto por professores quanto alunos. As réplicas de um tópico são representadas por `Forum Reply`. Já a lista de notícias é exclusividade dos professores e cada uma é representada pela classe `News`. Por fim, as `Live Questions` podem ser criadas por alunos e visualizadas por professores.

### 3.4.6 Subsistema Log

A Figura 18 exibe o diagrama de classes do subsistema **`Log`**.

Nesse subsistema, vale notar que os `Logs` estão associados a uma conta a fim de indicar as ações que um determinado usuário realizou dentro do sistema.

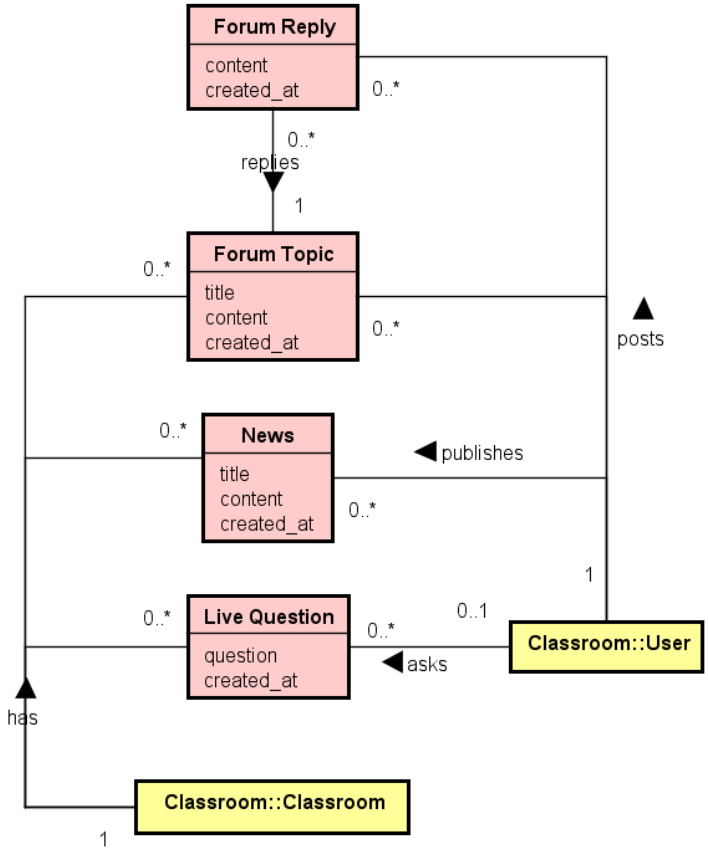


Figura 17 – Diagrama de Classes do subsistema Discussion.

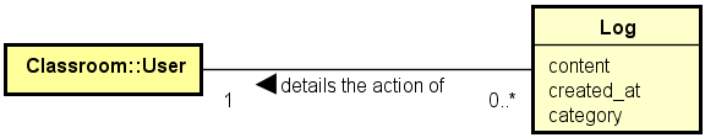


Figura 18 – Diagrama de Classes do subsistema Log.

# 4 Projeto Arquitetural

Como proposto na Seção 2.1, após as fases de especificação e análise de requisitos, ocorre a fase de projeto do sistema. Esta etapa é responsável pela modelagem de como será a implementação do sistema, incorporando, aos requisitos, as tecnologias a serem utilizadas.

Neste capítulo iremos mostrar a arquitetura do sistema. Na Seção 4.1, a arquitetura do sistema é descrita, enquanto na Seção 4.2 os modelos FrameWeb são apresentados.

## 4.1 Arquitetura do Sistema

A arquitetura de software do sistema Khoeus baseia-se no padrão MVCS (*Model, View, Controller, Services*) em uma tentativa de separar a representação de suas informações da interação realizada pelo usuário. A Figura 19 retrata o relacionamento entre as camadas deste padrão. Vale notar que as linhas sólidas representam relacionamento direto, enquanto as linhas tracejadas representam relacionamento indireto. Dessa forma, temos que:

- **Model:** consiste na camada responsável pelas regras de negócio e pela interação da aplicação com o banco de dados.
- **View:** basicamente, é a camada que interage diretamente com o usuário, exibindo dados por meio de arquivos HTML, XHTML, PDF, entre outros. No caso do *Ruby on Rails*, a camada de visão utiliza arquivos HTML contendo código Ruby embutidos.
- **Controller:** pode-se dizer que essa camada é a intermediadora do sistema, comunicando-se com as outras três camadas. As requisições provenientes do *browser* são processadas pelo *controller*, que processa as informações dos *models*, chama os serviços necessários e as retorna para as *views* para que sejam exibidas ao usuário.
- **Services:** nesta camada, espera-se agrupar os casos de uso desejados de forma a desacoplar as funcionalidades do sistema dos *controllers* ou dos *models*. Além disso, espera-se obter o máximo de reusabilidade dos serviços.

Falando mais especificamente do *Ruby on Rails*, temos que sua arquitetura é dividida em módulos que servem a funcionalidades distintas dentro do contexto da aplicação. Para o padrão MVCS, são utilizados três módulos que servem a cada uma das camadas definidas acima, como já mostrado na Figura 3: para as *views*, utiliza-se o módulo ***ActionView***, que provê funcionalidades para a renderização das páginas, construção de formulários, etc. Para os *models*, o módulo ***ActiveRecord*** garante a associação entre as classes do sistema,

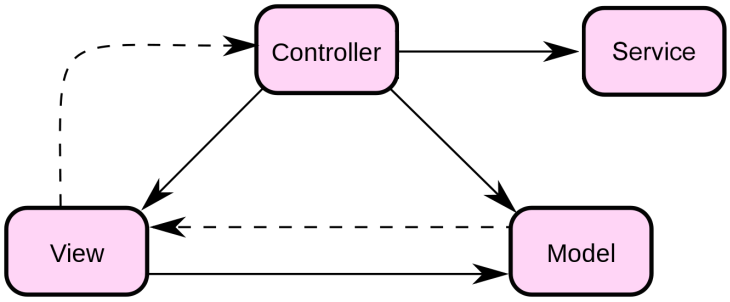


Figura 19 – Representação do padrão de design MVCS.

bem como o relacionamento da aplicação com o banco de dados. Por fim, mas não menos importante, os *controllers* pertencem ao módulo ***ActionController*** (ou simplesmente *Controller*), o qual provê um ambiente intermediário entre os *models* e as *views*, garantindo que a informação correta seja exibida ao usuário. Os *Services* do sistema são chamados **POROs** - *Plain Old Ruby Objects*, que não passam de classes comuns do Ruby com os métodos providos por aquele serviço.

## 4.2 Modelos FrameWeb

Nesta seção, são exibidos os modelos FrameWeb, citados anteriormente na Seção 2.4. Esses modelos também estão divididos nas camadas da arquitetura do sistema, conforme citado na Seção 4.1.

Apresentamos primeiramente o **Modelo de Entidades**. Os mapeamentos de persistência são meta-dados das classes de domínio que permitem que os *frameworks* ORM (*Object/Relational Mapping*), *frameworks* que realizam o mapeamento objeto/relacional automaticamente, transformem objetos que estão na memória em linhas de tabelas no banco de dados relacional. Por meio de mecanismos leves de extensão da UML, como estereótipos e restrições, adicionamos tais mapeamentos ao diagrama de classes de domínio, guiando os desenvolvedores na configuração do *framework* ORM. Apesar de tais configurações serem relacionadas mais à persistência do que ao domínio, elas são representadas no Modelo de Entidades porque as classes que são mapeadas e seus atributos são exibidas neste diagrama. (SALVATORE, 2016)

As Figuras 20, 21, 22, 23, 24, 25 e 26 mostram o modelo de entidades para os subsistemas `Auth`, `Classroom`, `Board`, `BoardInteractions`, `Feedback`, `Discussion` e `Log`, respectivamente. A fim de se aproximar da abordagem adotada por bancos de dados, todos os atributos do tipo `String` tiveram um tamanho (`size`) atribuído a eles. Da mesma forma, atributos do tipo Data (`Date`) especificam a granularidade do valor armazenado (*time*, *date* ou *timestamp*), sendo que no último caso, a restrição pode ser omitida. Além disso, diferentemente da abordagem original do FrameWeb proposto em 2007, todos os atributos que são não nulos tiveram a *tag not null* omitida, bem como todas as associações foram

consideradas do tipo *lazy* devido à forma com que o *Ruby on Rails* carrega os objetos na memória.

Vale ressaltar que todas as classes de domínio estendem de `ActiveRecord` do *Ruby on Rails*, que é a classe relacionada à persistência de dados do *framework*, sendo que essa herança não é mostrada no diagrama com o intuito de não poluí-lo com várias associações. Além disso, por padrão do framework, todas as classes possuem navegabilidade para os dois lados.

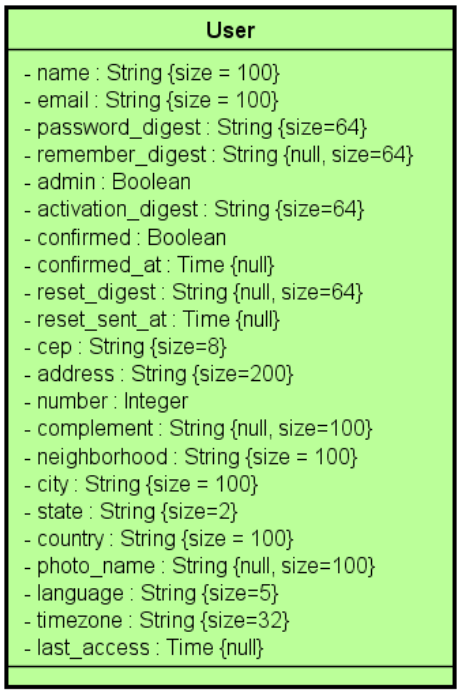


Figura 20 – Modelo de Entidades do subsistema Auth

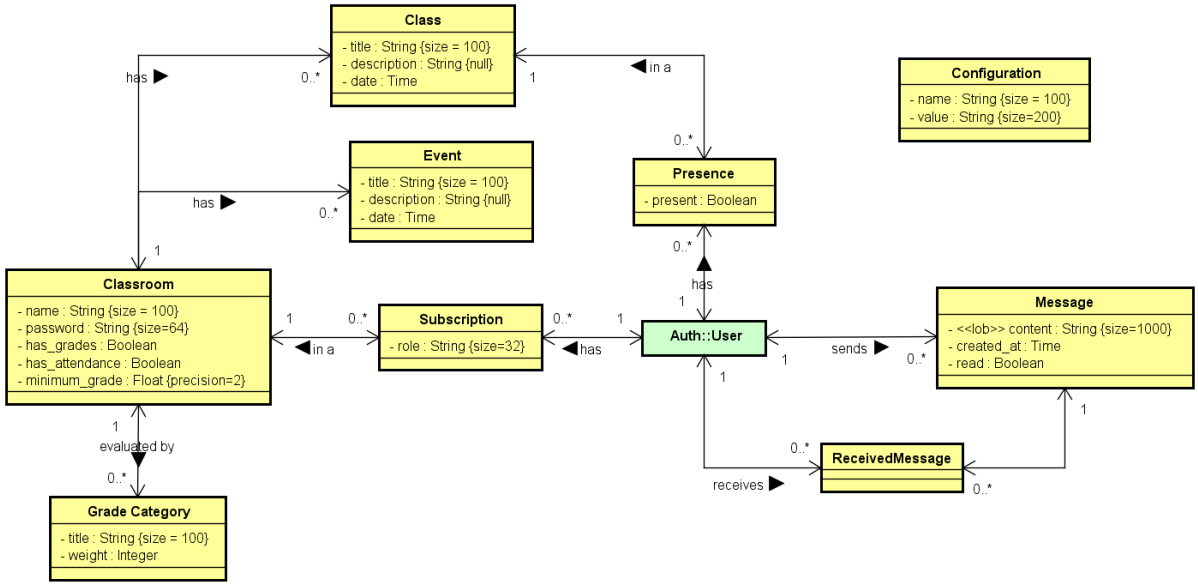


Figura 21 – Modelo de Entidades do subsistema Classroom

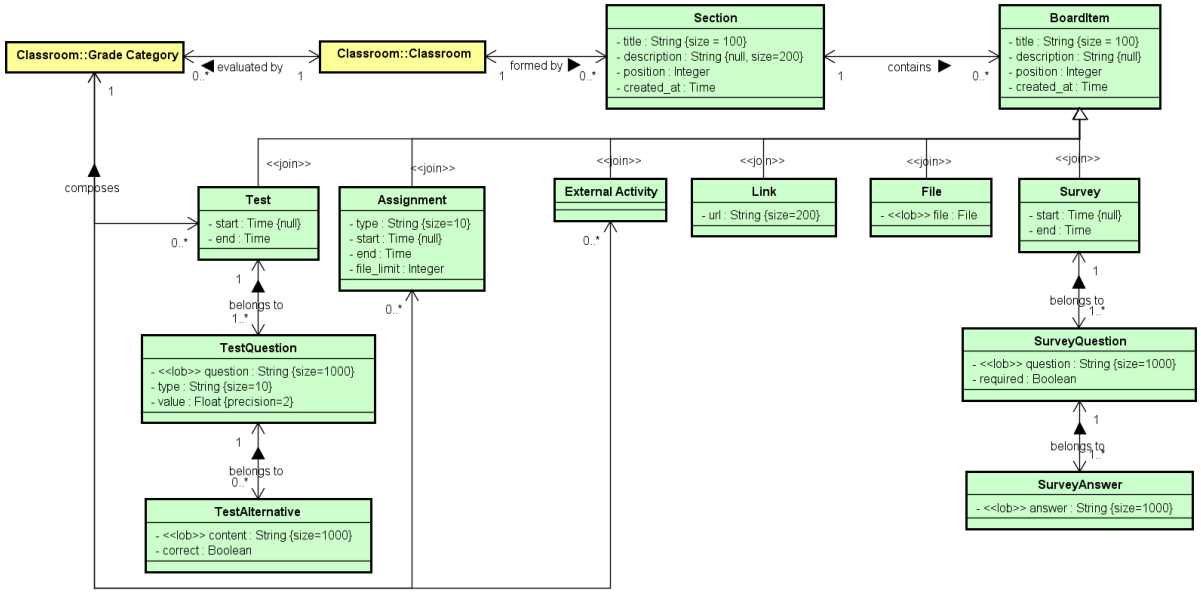


Figura 22 – Modelo de Entidades do subsistema Board

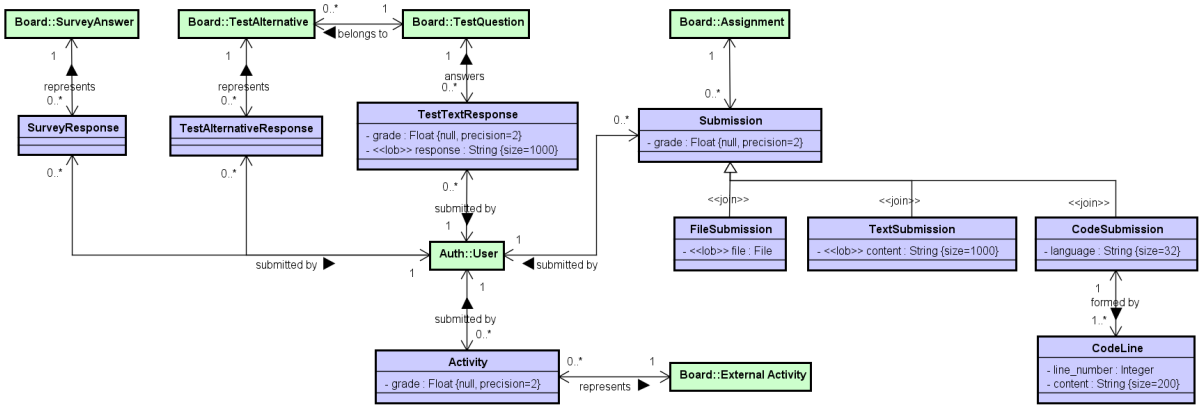


Figura 23 – Modelo de Entidades do subsistema BoardInteractions

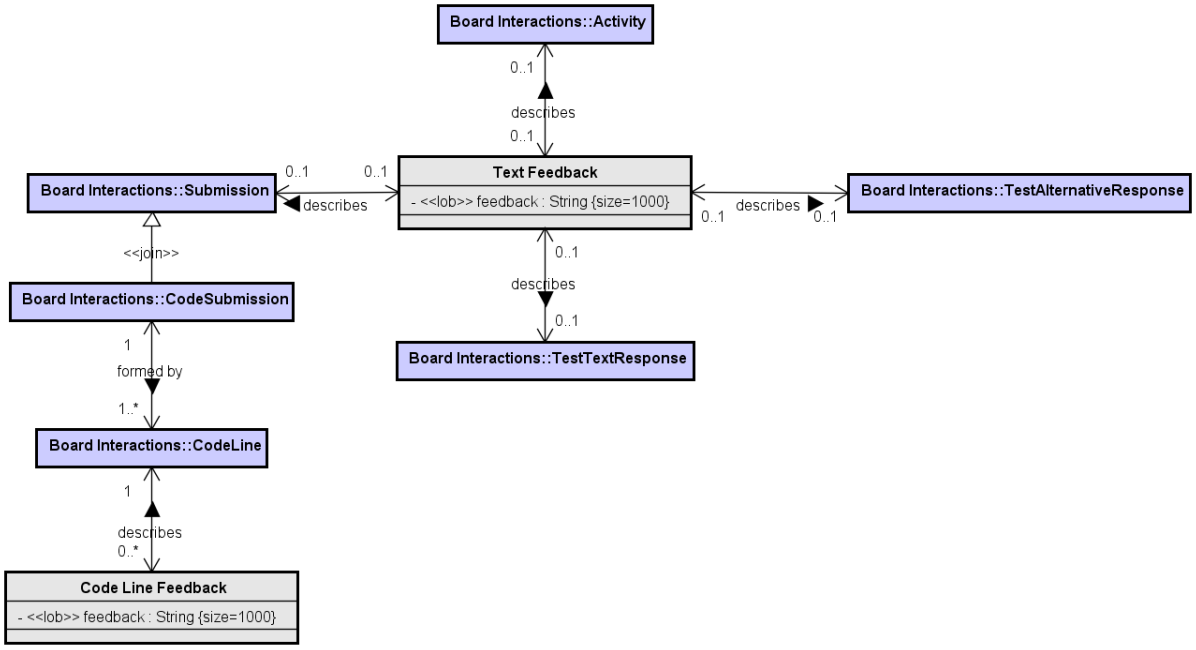


Figura 24 – Modelo de Entidades do subsistema Feedback

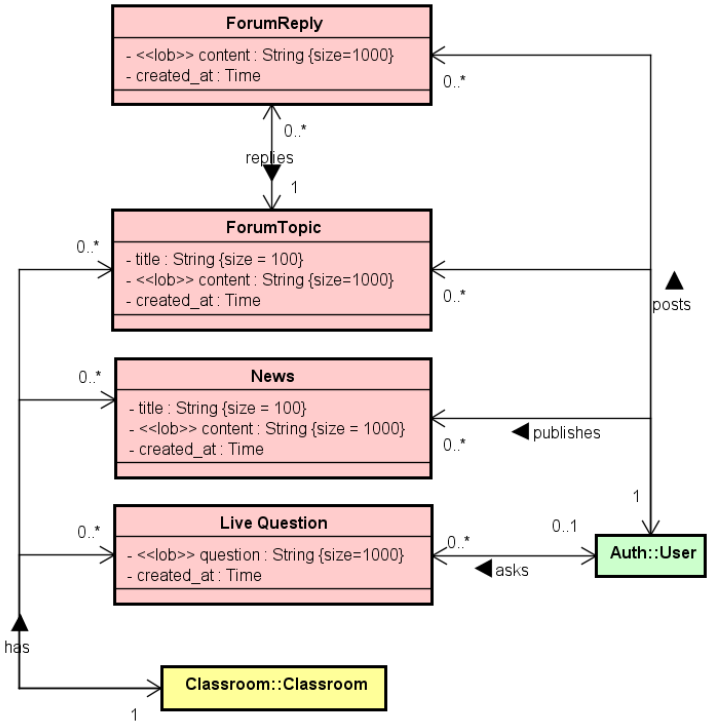


Figura 25 – Modelo de Entidades do subsistema Discussion

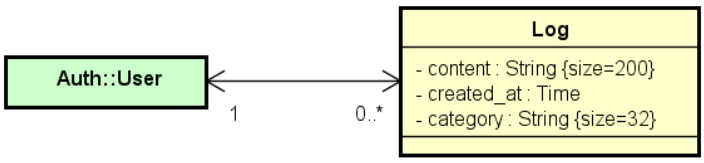


Figura 26 – Modelo de Entidades do subsistema Log

Já o **Modelo de Navegação** é um diagrama de classe da UML que representa os diferentes componentes que formam a camada de Apresentação, como páginas Web, formulários HTML e classes de ação. Esse modelo é utilizado pelos desenvolvedores para guiar a codificação das classes e componentes das camadas de ***View*** e ***Controller***. A classe de ação é o principal componente deste modelo: suas associações de dependência ditam o controle de fluxo quando uma ação é executada.

As funcionalidades criar, editar, excluir e visualizar (abreviadas de CRUD, do inglês *create, read, update e delete*), seguem um mesmo fluxo de execução e de interação com o usuário. Tais funcionalidades são similares para todos os casos de uso cadastrais devido à utilização de *Ruby on Rails*. Esse fluxo de execução similar é representado pela Figura 27 que é um modelo de apresentação genérico. Neste caso, utilizou-se o caso de uso cadastral de `ForumTopic` a título de exemplo.

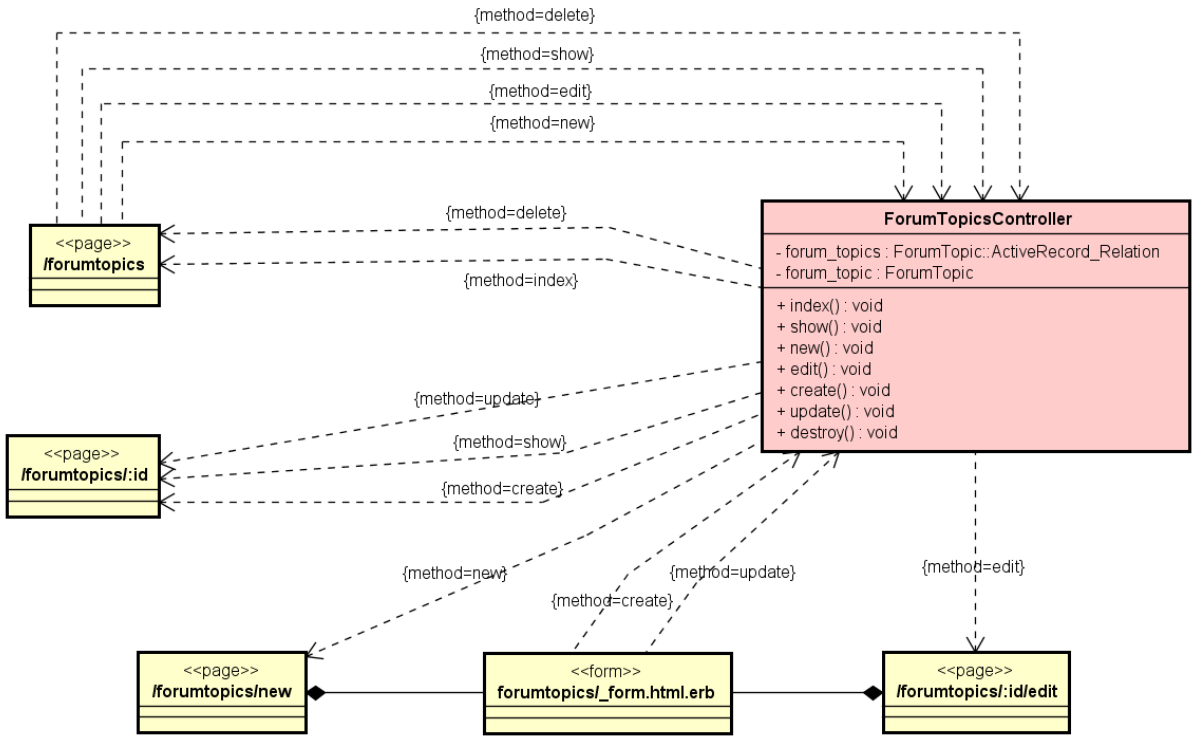


Figura 27 – Modelo de Navegação para casos de usos cadastrais.

Este é um modelo genérico para todas as entidades que possuem funcionalidades do tipo CRUD. Deve existir, para cada entidade, um *controller* central que fará o controle das

requisições executadas pelo usuário. Para interfacear com o usuário, o sistema provê uma tela de listagem de entidades (*/forumtopics*), uma para detalhamento (*/forumtopics/:id*), uma para criação (*/forumtopics/new*) e outra para edição (*/forumtopics/:id/edit*). A partir delas, determinados métodos são invocados no *controller* a fim de realizar a ação desejada e, após isso, uma resposta é retornada ao usuário.

Para os casos de uso que apresentam funções diferentes das CRUDs, o modelo anterior não pode ser aplicado. A Figura 28 apresenta o modelo de navegação para o fluxo de autenticação do sistema, incluindo os casos de uso **Criar conta**, **Efetuar login** e **Obter senha esquecida**.

Nesse diagrama, vale notar a presença da restrição **result** indicando o tipo de retorno da requisição. Em todos os casos, caso algum erro tenha ocorrido durante o processamento das informações, o usuário retorna para a página anterior a fim de que possa corrigir seu erro.

Além disso, fica evidente o fluxo de navegação caso as informações estejam corretas: tanto para o login quanto para o reset de senha propriamente dito, o usuário é redirecionado para a lista de turmas. Já no caso do cadastro e da solicitação de uma nova senha, o usuário é redirecionado para a tela de login, onde um aviso deverá informá-lo para que acesse seu e-mail para prosseguir com o processo.

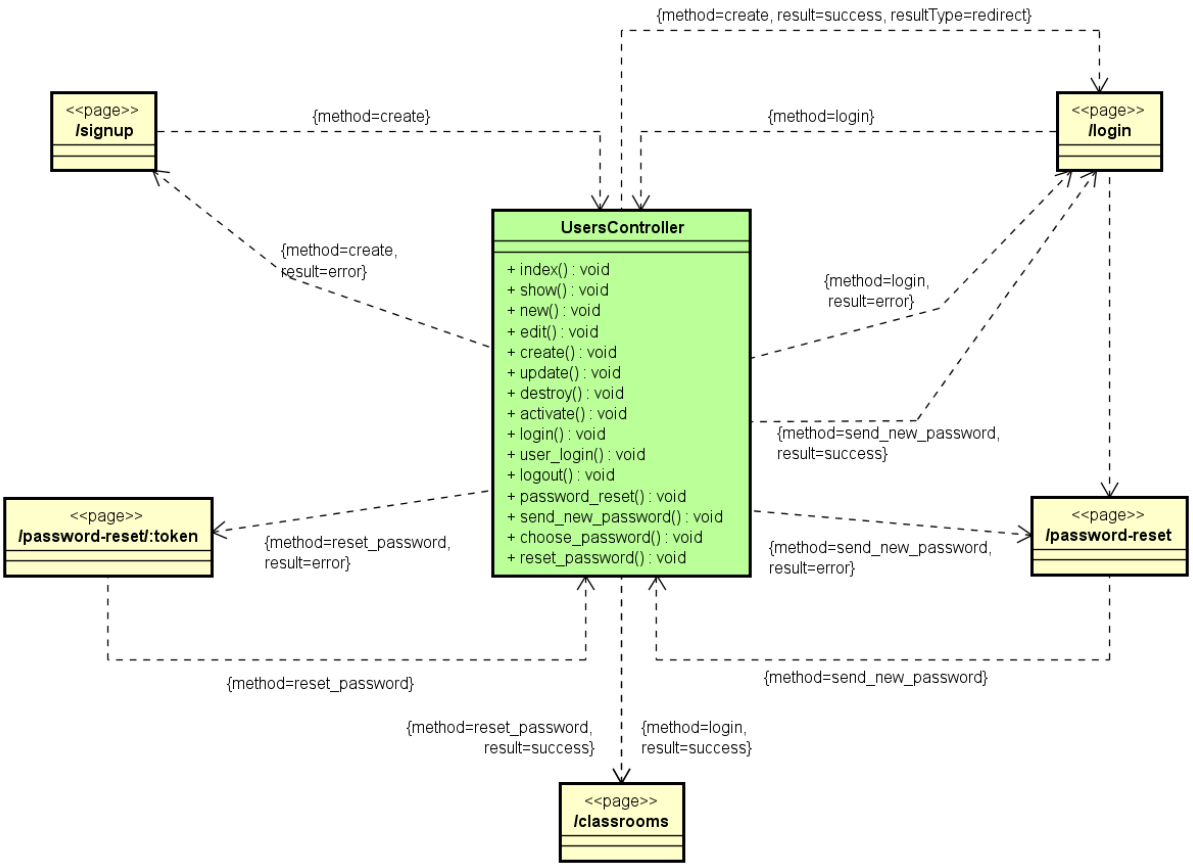


Figura 28 – Modelo de Navegação para o fluxo de autenticação do sistema

A Figura 29 exibe o modelo de navegação para o caso de uso *Fazer download em lote das submissões de uma tarefa ou prova*. Neste modelo, levou-se em consideração o download em lote das submissões de uma tarefa. Para isso, o professor poderá acessar a página de detalhamento de uma determinada tarefa e, lá, solicitar o download das submissões. Com isso, o método `download_submissions` é invocado, retornando para o usuário um arquivo compactado **.zip** contendo todas as submissões dos alunos daquela turma para aquela determinada tarefa.

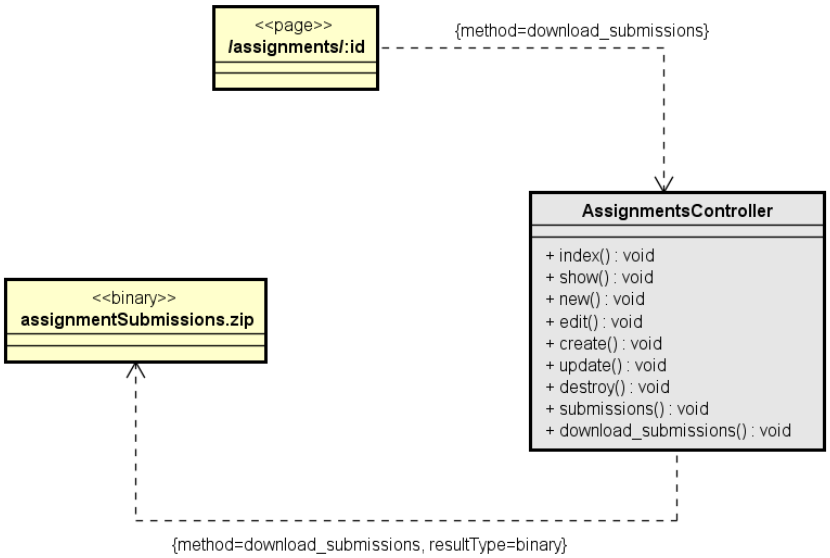


Figura 29 – Modelo de Navegação para o caso de uso 'Fazer download em lote das submissões de uma tarefa ou prova'.

O próximo modelo FrameWeb a ser mostrado é o **Modelo de Aplicação**, que consiste em um diagrama de classes da UML que representa as classes de serviço, responsáveis pela codificação dos casos de uso, e suas dependências. Esse diagrama é utilizado para guiar a implementação das classes do pacote **Aplicação** e a configuração das dependências entre os pacotes **Controle e Aplicação**, ou seja, quais *controllers* dependem de quais classes de serviço (SOUZA, 2007).

Todas as classes de serviço estendem de *CrudService*, representada na Figura 30 de forma genérica e essa herança não é mostrada no diagrama com o intuito de não polui-lo com várias associações.

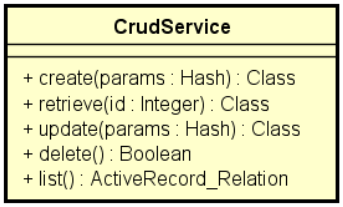


Figura 30 – Representação da classe `CrudService` contendo os métodos de CRUD.

Embora a arquitetura do *Ruby on Rails* não incorpore injeção de dependências, assume-se que existe uma dependência entre o *controller* e a classe de serviço uma vez que, sem ela, o primeiro deixaria de funcionar.

A seguir, a Figura 31 representa o modelo de aplicação para o subsistema *Auth*, a Figura 32 representa o modelo de aplicação para o subsistema *Classroom*, a Figura 33 representa o modelo de aplicação para o subsistema *Discussion*, a Figura 34 representa o modelo de aplicação para o subsistema *Board*, a Figura 35 representa o modelo de aplicação para o subsistema *Board Interactions*, a Figura 36 representa o modelo de aplicação para o subsistema *Feedback* e a Figura 37 representa o modelo de aplicação para o subsistema *Logs*.

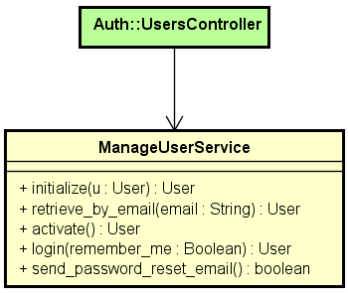


Figura 31 – Modelo de Aplicação para o subsistema Auth.

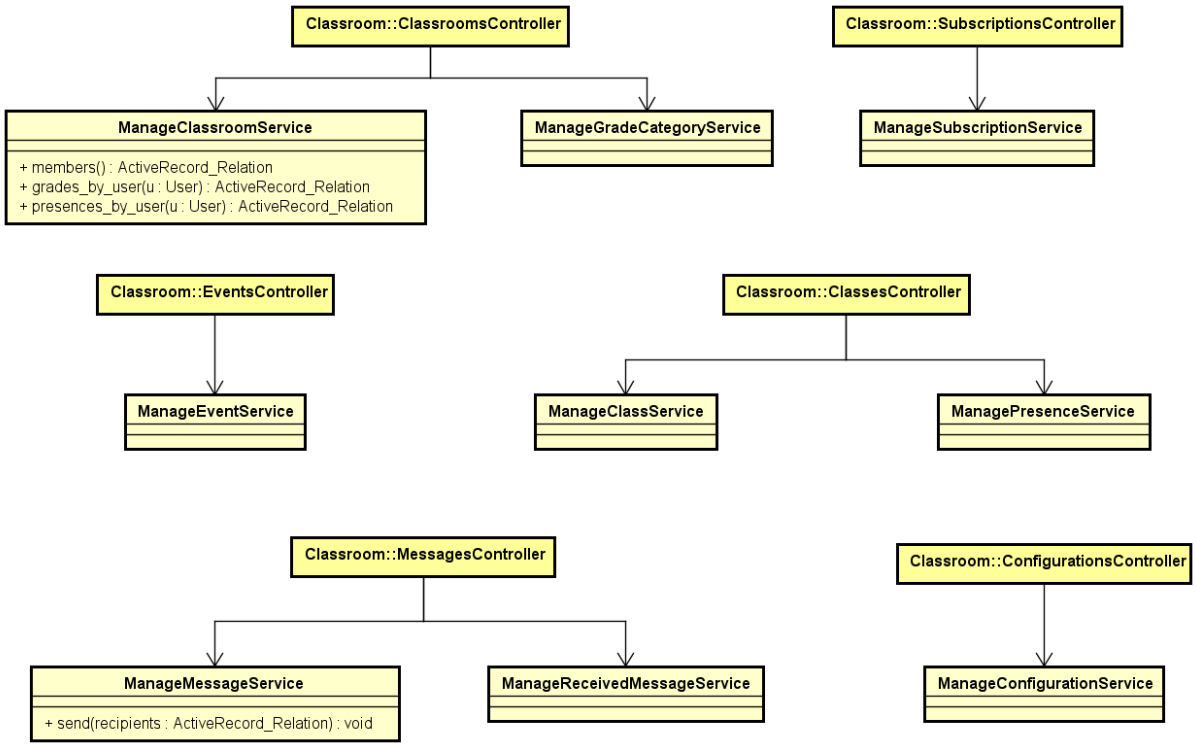


Figura 32 – Modelo de Aplicação para o subsistema Classroom.

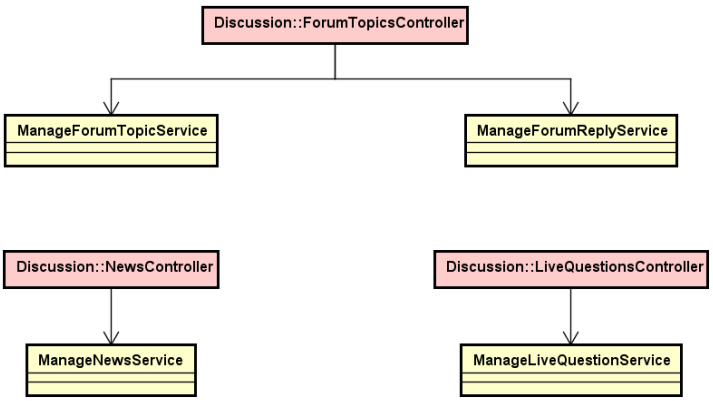


Figura 33 – Modelo de Aplicação para o subsistema Discussion.

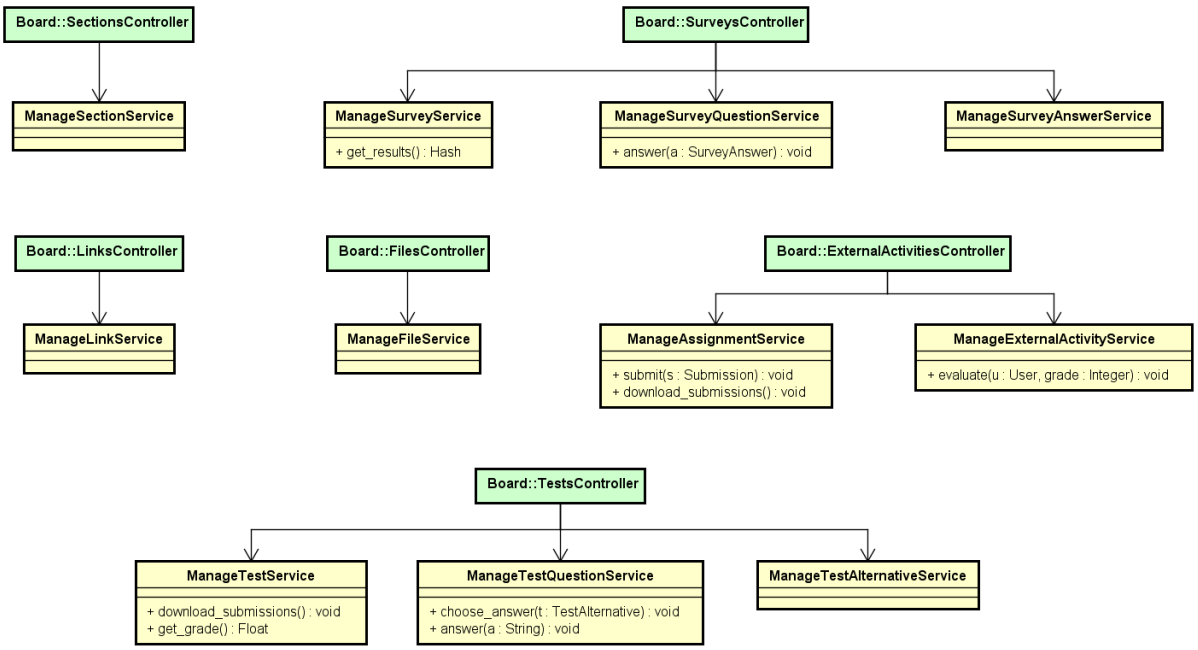


Figura 34 – Modelo de Aplicação para o subsistema Board.

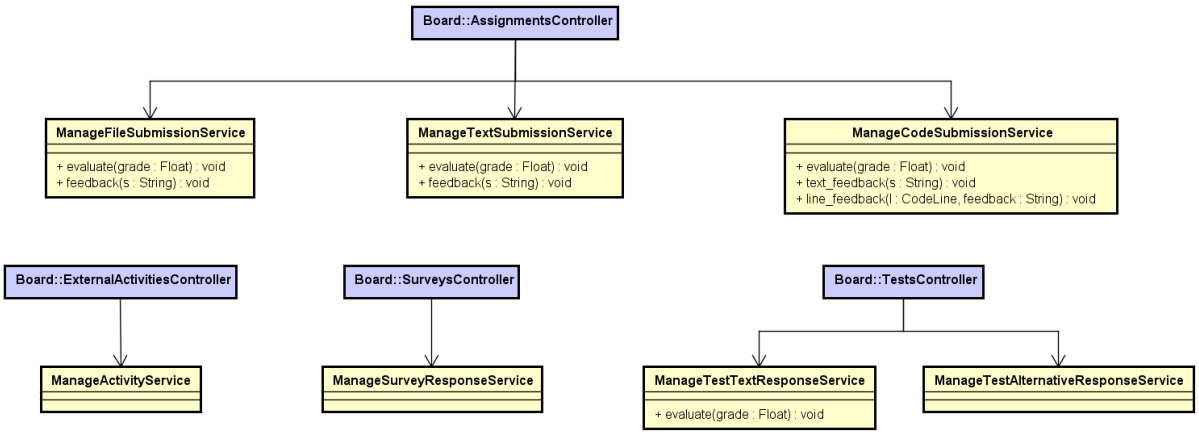


Figura 35 – Modelo de Aplicação para o subsistema Board Interactions.

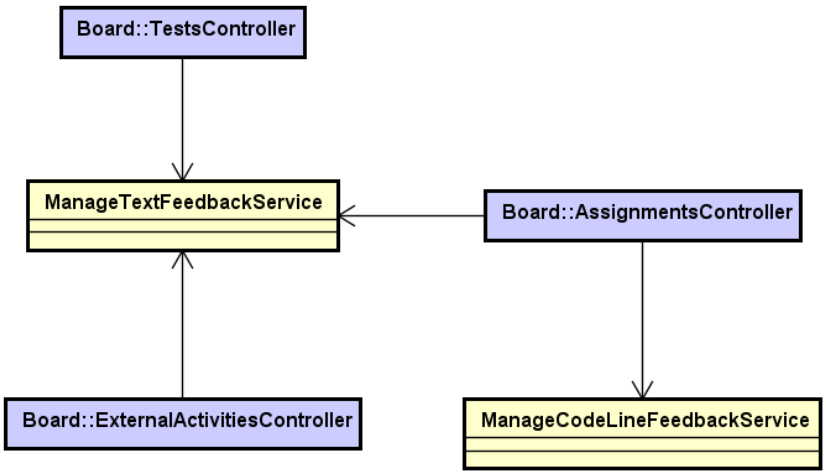


Figura 36 – Modelo de Aplicação para o subsistema Feedback.

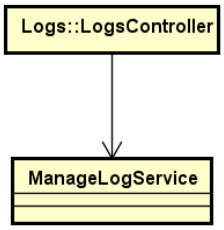


Figura 37 – Modelo de Aplicação para o subsistema Logs.

# 5 Implementação e Apresentação

Neste capítulo serão apresentados aspectos da implementação do sistema, bem como seu resultado final por meio de diversas capturas de tela.

## 5.1 Implementação do sistema

Conforme mencionado na Seção 4.1, o Khoeus foi desenvolvido utilizando o padrão MVCS e o framework *Ruby on Rails*. Dessa forma, os diversos diretórios adotados foram organizados a fim de favorecer a navegação entre os vários arquivos conforme a Figura 38. Dentre os diversos diretórios listados, destacam-se:

- **/app:** inclui a maioria dos arquivos específicos da aplicação
  - **/controllers:** contém todos os arquivos da camada *Controller* do sistema.
  - **/models:** contém todos os arquivos da camada *Model* do sistema.
  - **/services:** contém todos os arquivos da camada *Service* do sistema.
  - **/views:** contém todos os arquivos da camada *View* do sistema.
- **/config:** contém arquivos de configuração de servidor, bancos de dados, email e rotas.
- **/db:** contém os arquivos chamados *migrations* que são responsáveis pela criação/edição de tabelas do banco de dados.
- **/spec:** inclui os arquivos de testes utilizados pelo padrão TDD, exposto na Seção 2.3.

Nessa seção, serão levados em consideração apenas os diretórios relacionados diretamente ao padrão MVCS, ou seja: os diretórios **/controllers**, **/models**, **/services** e **/views**.

### 5.1.1 Model

Neste diretório, o Rails armazena as informações de cada classe específica do sistema, como **Classroom** ou **User**, sendo que cada uma delas possui um arquivo exclusivo contendo informações relativas a validação de dados, associações e herança. Vale ressaltar que os atributos de cada classe são definidos pelas *migrations*, e não pelos *models*.

A título de exemplo, o Código 5.1 mostra algumas das propriedades possíveis para um *Model* do sistema. Para melhor exemplificar a validação de atributos, as associações entre classes e até mesmo o mecanismo de herança, escolheu-se a classe `BoardItem`.

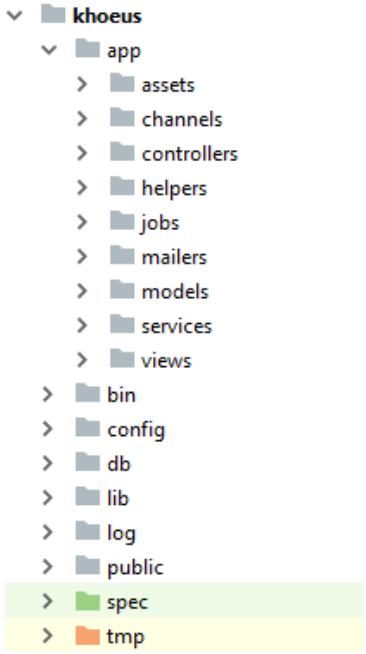


Figura 38 – Estrutura de diretórios utilizada no sistema

Código 5.1 – *Model* da classe BoardItem

```
1  class BoardItem < ApplicationRecord
2    belongs_to :section
3
4    has_attached_file :document_file
5
6    self.inheritance_column = :type
7
8    validates :title,
9      presence: true
10   validates :position,
11     presence: true,
12     numericality: {only_integer: true}
13 end
```

Para as associações, o *Ruby on Rails* provê os métodos **belongs_to**, **has_one** e **has_many** de forma a especificar quais são as outras classes com que aquela se relaciona, bem com especificando a cardinalidade dessa associação.

Para o caso de herança, o *Ruby on Rails* implementa por padrão o modelo de herança baseado em *Single Table Inheritance*, ou seja, o banco de dados possui apenas uma tabela contendo os atributos de todas as classes especializadas. Para permitir isso, deve-se informar qual será a coluna que irá diferenciar a classe em questão por meio do

atributo **`inheritance_column`**.

A fim de permitir que certos parâmetros sejam validados no momento de sua criação ou edição, é possível especificar quais serão as validações para cada um dos atributos da classe. No caso da classe **`BoardItem`**, tanto o atributo `title` quanto o atributo `position` devem estar obrigatoriamente presentes. No caso deste último, deve-se garantir que ele é um inteiro.

### 5.1.2 Controller

A fim de controlar as requisições realizadas pelos usuários e seus respectivos retornos, o sistema utiliza de classes controladoras de forma a modularizar o tratamento dado a cada uma delas. Por meio do arquivo de rotas (não detalhado neste capítulo a fim de simplificação), o *Ruby on Rails* garante que, ao realizar uma requisição para uma determinada URL, um determinado método do controlador responsável será executado.

Sempre que um determinado usuário realiza uma requisição a uma determinada rota do sistema, o arquivo de rotas (Código 5.2) encontra o método do controlador responsável por aquela requisição e o executa. Com relação aos casos de uso CRUD para as turmas do sistema, todas as requisições são gerenciadas pela classe `ClassroomsController`, conforme Código 5.3. Como é possível observar, o *Controller* realiza o processo de autenticação e autorização das rotas por meio do método `load_and_authorize_resource` e carrega a turma (`Classroom`) em questão no caso dos métodos `show`, `edit`, `update` e `destroy`, uma vez que estes se tratam de uma turma já criada anteriormente.

Código 5.2 – Gerenciamento das rotas do sistema

```
1 Rails.application.routes.draw do
2
3 resources :users
4 resources :classrooms
5 # Other routes and resources
6
7 end
```

Código 5.3 – *Controller* responsável pela inscrição de usuários em turmas (versão simplificada)

```
1 class ClassroomsController < ApplicationController
2   load_and_authorize_resource
3
4   before_action :set_classroom, only: [:show, :edit, :update,
        :destroy]
```

```
5
6   # GET /classrooms
7   def index
8     @classrooms = Classroom.all
9   end
10
11  # GET /classrooms/1
12  def show
13    @sections = ManageSectionService.new.
        retrieve_from_classroom(@classroom)
14    @items = ManageBoardItemService.new.
        retrieve_from_classroom(@classroom)
15  end
16
17  # GET /classrooms/new
18  def new
19    @classroom = Classroom.new
20  end
21
22  # GET /classrooms/1/edit
23  def edit
24  end
25
26  # POST /classrooms
27  def create
28    if (@classroom = ManageClassroomService.new.create(
        classroom_params))
29      redirect_to @classroom, notice: 'Classroom was
          successfully created.'
30    else
31      render :new
32    end
33  end
34
35  # PATCH/PUT /classrooms/1
36  def update
37    if ManageClassroomService.new(@classroom).update(
        classroom_params)
38      redirect_to @classroom, notice: 'Classroom was
```

```
          successfully updated.'
39      else
40        render :edit
41      end
42    end
43
44    # DELETE /classrooms/1
45    def destroy
46      ManageClassroomService.new(@classroom).delete
47      redirect_to classrooms_url, notice: 'Classroom was
          successfully destroyed.'
48    end
49
50    private
51    def set_classroom
52      @classroom = ManageClassroomService.new.retrieve(params[:
          id]) || ManageClassroomService.new.retrieve(params[:
          classroom_id])
53    end
54
55    def classroom_params
56      params.require(:classroom).permit(:name, :password, :
          has_grades, :ha_attendance, :minimum_grade )
57    end
58
59  end
```

A partir disso, cada um dos métodos públicos do controlador irá cuidar da requisição pela qual ele é necessário. No caso do método `create`, por exemplo, o serviço `ManageClassroomService` (explicado mais detalhadamente na Seção 5.1.4) é invocado numa tentativa de criar uma nova instância de turma. Caso seja bem sucedido, o usuário é redirecionado para a página da turma e notificado de que a turma foi criada com sucesso. Caso contrário, ele retorna ao formulário de criação para que possa corrigir seus erros.

### 5.1.3 View

Na camada *View* do sistema são gerenciados todos os arquivos relacionados à exibição de informações aos usuários como páginas HTML, formulários e *partials* (conteúdo HTML reaproveitável em múltiplas páginas). Para permitir a exibição de conteúdo dinâmico, é possível embutir código Ruby em todos estes arquivos HTML. Para isso, o *framework*

utilizado permite que todas as variáveis com um `@` no começo definidas no controlador possam ser acessadas nos arquivos da camada de visão relacionados àquele *controller*. Como exemplo, pode-se citar a possibilidade de iterar sobre uma lista de turmas e acessar o título de cada uma delas, conforme mostrado no Código 5.4.

Código 5.4 – Exibição das turmas do sistema

```
<div class="row">
  <% @classrooms.each do |classroom| %>
    <!-- exibição de informações da turma -->
    <p class="number text-center"><%= students_count(
       classroom) %></p>
    <h2 class="classroom-title"><%= classroom.name %></h2>
  <% end %>
</div>
```

### 5.1.4 Services

Por padrão, o *Ruby on Rails* não fornece uma camada de serviço e, portanto, espera-se que todas as funcionalidades do sistema sejam tratadas diretamente na camada *Model*. No entanto, conforme proposto por (LEWIS, 2017), a utilização de uma Camada de Serviço dentro do *framework* torna-se um adicional de grande valor uma vez que facilita a organização dos casos de uso do sistema: ao invés de cada classe do sistema tratar um pedaço de diversos casos de uso, estes são concentrados em classes específicas.

A fim de facilitar os métodos de CRUD para as diversas classes do sistema, foi criada uma classe de serviço genérica chamada `CrudService` contendo os cinco métodos principais: `list`, `retrieve`, `create`, `update` e `delete` conforme Código 5.5. A fim de permitir que todas as outras classes de serviço possam utilizar esses métodos, a classe `CrudService` inicializa em seu construtor as variáveis que definirão a classe e a instância em questão (caso exista). Com isso, todas as classes de serviço herdam de `CrudService` os métodos básicos de CRUD e podem implementar outros caso seja necessário.

Código 5.5 – Classe genérica de serviço contendo os principais métodos de CRUD

```
class CrudService

  def initialize(model, entity = nil)
    @model = model.classify.constantize
    @entity = entity if entity
  end

```

```
8   def list
9     @model.all
10  end
11
12  def create(params)
13    @entity = @model.create(params)
14    @entity.save ? @entity : false
15  end
16
17  def retrieve(id)
18    @entity = @model.find_by(id: id)
19  end
20
21  def update(params)
22    @entity.update_attributes(params)
23  end
24
25  def delete
26    @entity.destroy
27  end
28
29 end
```

### 5.1.5 Test-Driven Development

Como já abordado na Seção 2.3, usou-se do método de programação orientada a testes a fim de garantir a consistência da plataforma. Para isso, utilizou-se o framework *Rspec* que provê uma série de classes que permitem realizar os testes automatizados no sistema.

Conforme Código 5.6, pode-se observar que os testes relacionados aos *controllers* do sistema simulam acessos a certas páginas assumindo determinadas *roles* e, com isso, espera-se uma certa resposta (que pode ser um redirecionamento, a exibição de um erro ou de uma determinada página). No caso das ações de *create* para um CRUD, fica nítido que espera-se a criação de uma nova entidade do sistema.

Código 5.6 – Trecho do código responsável por testar a classe *ClassroomController*

```
1 RSpec.describe ClassroomsController, type: :controller do
2   let(:user) {create :user, confirmed: true}
3   let(:other_user) {create :user, email: 'aaa@gmail.com',
```

```
    confirmed: true}
let(:admin_user) {create :admin_user}
let(:classroom) {create :classroom}
let(:subscription) {create :subscription}
let(:teacher_subscription) {create :teacher_subscription}

describe 'GET #new' do
  context 'when admin' do
    it 'returns a success response' do
      log_in admin_user
      get :new
      expect(response).to have_http_status 200
    end
  end
  context 'when not admin' do
    it 'returns status 403' do
      log_in user
      get :new
      expect(response).to have_http_status 403
    end
  end
  context 'when logged out' do
    it 'redirect to login page' do
      get :new
      expect(response).to redirect_to login_path
    end
  end
end


describe 'POST #create' do
  context 'when not admin' do
    it 'returns status 403' do
      log_in user
      post :create, params: {classroom: attributes_for(:
          classroom)}
      expect(response).to have_http_status 403
    end
  end
```

```
41      context 'when admin with invalid params' do
42        it 'redirects to create page' do
43          log_in admin_user
44          post :create, params: {classroom: attributes_for(:
                classroom, name: nil)}
45          expect(response).to render_template :new
46        end
47      end
48      context 'when admin with valid params' do
49        before(:each) {log_in admin_user}
50        it 'creates a new Classroom' do
51          expect {
52          post :create, params: {classroom: attributes_for(:
                classroom)}
53          }.to change(Classroom, :count).by(1)
54        end
55
56        it 'redirects to the created classroom' do
57          post :create, params: {classroom: attributes_for(:
                classroom)}
58          expect(response).to redirect_to(Classroom.last)
59        end
60      end
61    end
62
63    #Other tests
64  end
```

Além dos testes orientados aos *controller*, também foram criados testes voltados à validação dos atributos das entidades e alguns testes focados na funcionalidade de algumas classes de serviço, como a resolução de uma prova.

## 5.2 Apresentação do sistema

Nesta seção, apresentar-se-á o sistema por meio de uma série de capturas de tela. A Figura 39 mostra a tela inicial de login no sistema onde o usuário poderá escolher entre a opção de se cadastrar ou, caso já possua um cadastro, efetuar login.

Caso o usuário queira se cadastrar, ele será levado à tela de cadastro (Figura 40) onde deverá preencher suas informações como nome, e-mail e senha desejada. Aqui, vale


Figura 39 – Tela inicial

ressaltar que uma série de validações é feita, como por exemplo a validação do formato do e-mail, da presença do nome e do tipo de dado inserido no campo Número (que deve aceitar apenas inteiros). Uma vez realizado o cadastro, o usuário receberá em sua caixa de entrada um e-mail contendo o link de ativação para que possa acessar o sistema.

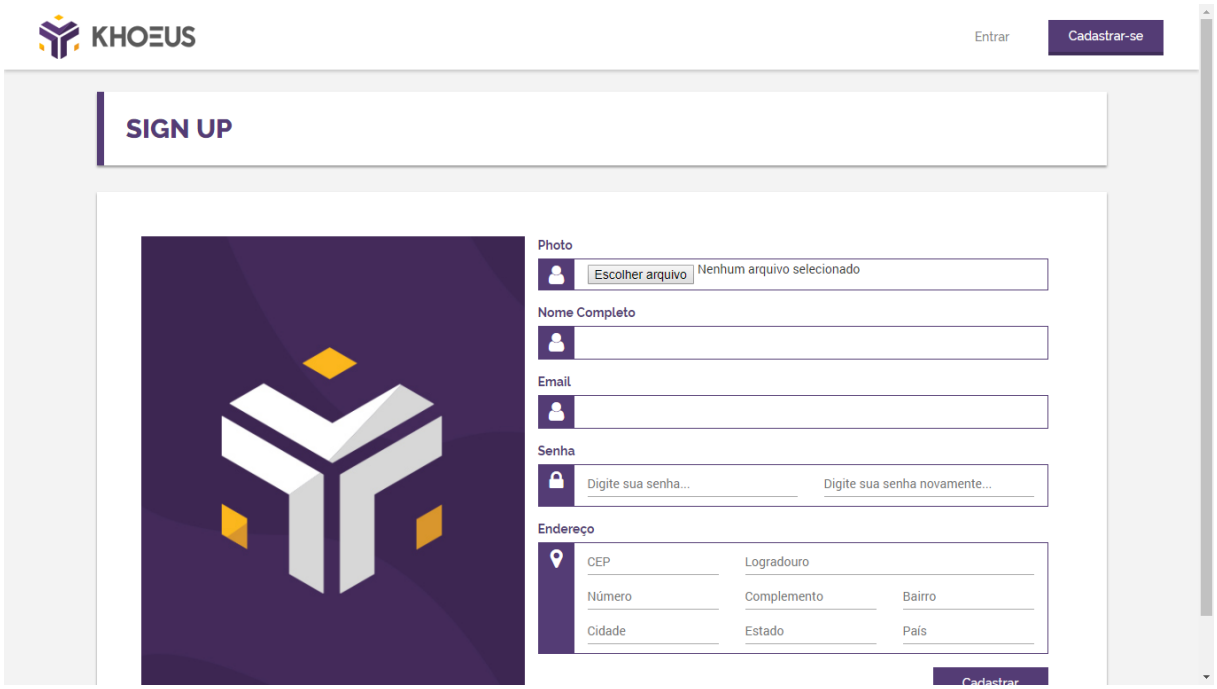


Figura 40 – Tela de cadastro

Assim que a conta do usuário estiver ativada, ele poderá acessar a tela de login (Figura 41) para acessar o sistema por meio de seu e-mail e senha escolhidos durante

o cadastro. Caso deseje permanecer logado mesmo após fechar o navegador, o usuário poderá marcar a opção *Lembrar de mim*. Caso tenha esquecido sua senha, o usuário poderá solicitá-la por meio do link *Forgot your password?* ainda na tela de login. Uma vez solicitada, o usuário receberá em sua caixa de entrada um e-mail com um link que dá direito a realizar o *reset* de sua senha.

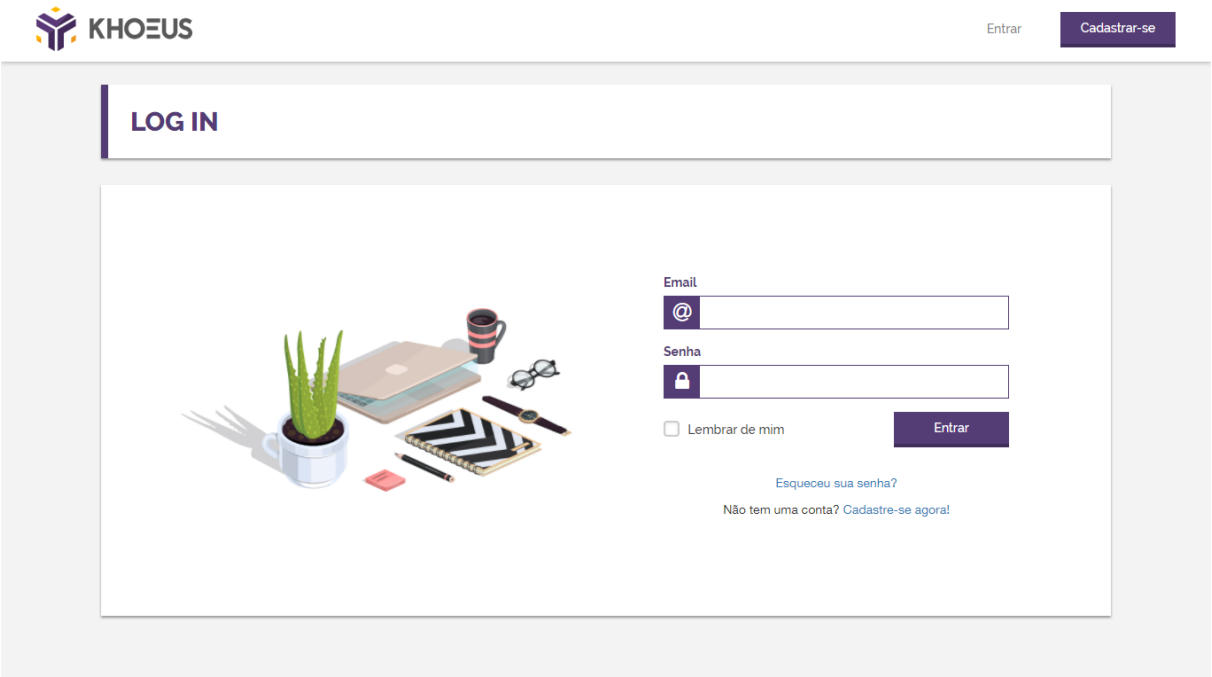


Figura 41 – Tela de login

Após finalmente logar no sistema, o usuário terá acesso a uma lista de turmas, conforme mostra a figura 42. Caso o usuário seja um administrador do sistema, ele poderá criar novas turmas (Figura 43) ou acessar qualquer um das turmas existentes no sistema. Vale ressaltar que no momento da criação da turma é importante especificar a forma com que a média será calculada por meio da definição das categorias de nota e seus respectivos pesos. Caso o usuário não seja um administrador, ele deverá se inscrever na turma desejada por meio de uma senha que deverá ser fornecida a ele previamente.

Uma vez inscrito em uma turma, o usuário terá acesso ao *board* daquela turma, bem como ao seu livro de notas e à lista de presença.

No *board*, o aluno terá acesso a todo o conteúdo publicado pelos seus professores divididos em seções, conforme a Figura 44. Cada seção pode conter links, documentos, questionários, provas, tarefas ou atividades externas. Caso o usuário seja um professor da turma, ele poderá adicionar novos itens ao *board*, cada um deles com seu próprio formulário e seus próprios atributos. No caso de provas, tarefas e atividades externas, vale ressaltar que o professor deverá escolher a qual categoria de nota aquele item pertence. Isso será essencial no momento do cálculo da média dos alunos. Caso o usuário seja apenas um aluno, ele poderá interagir de forma diferente com cada um dos tipos de item disponíveis.

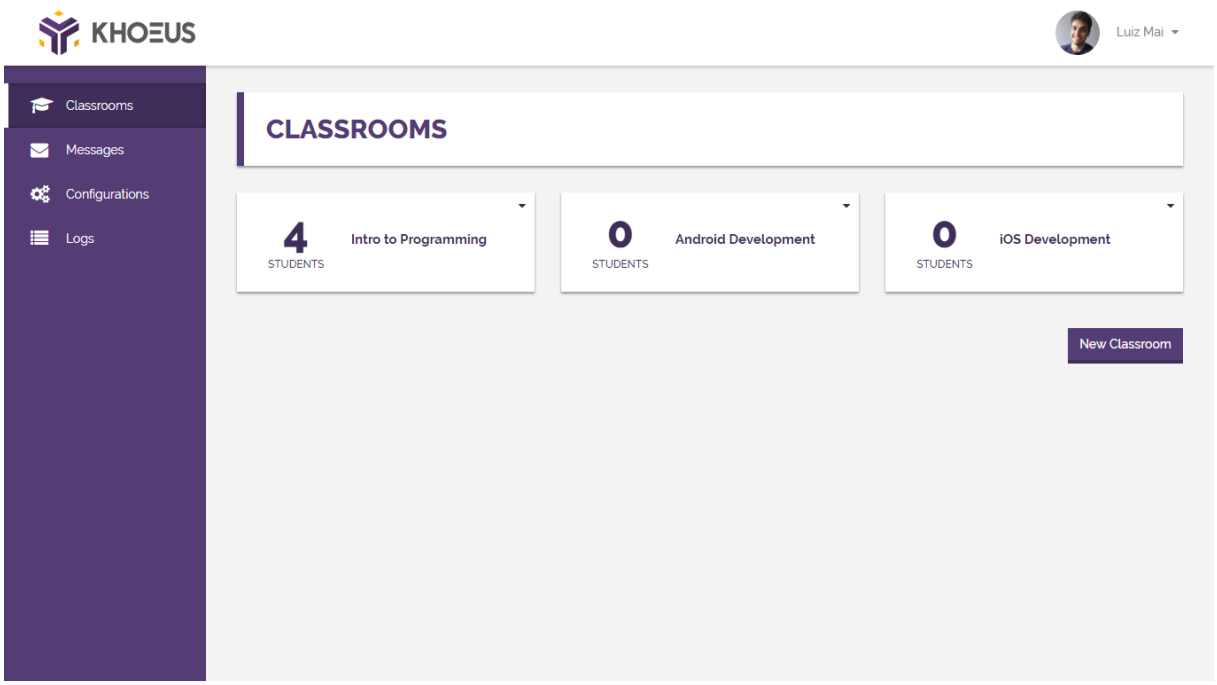


Figura 42 – Lista de turmas cadastradas no sistema

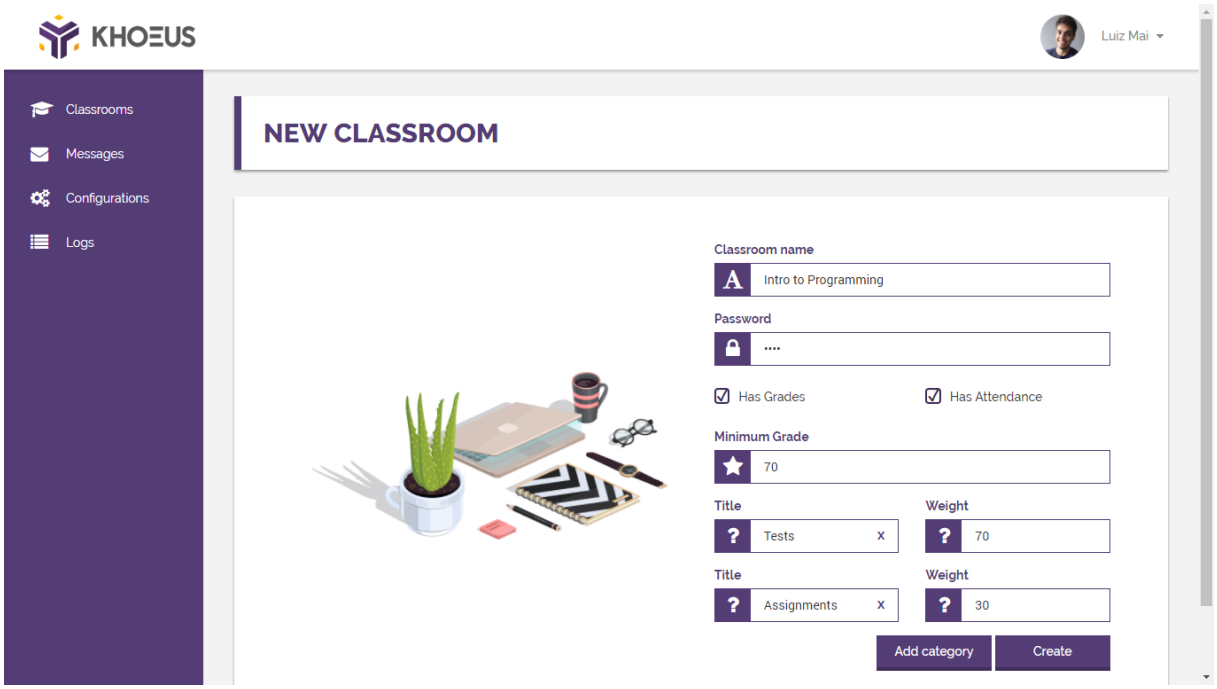


Figura 43 – Formulário de criação de turma

Caso o aluno clique sobre um **Link**, uma nova aba abrirá no navegador levando o usuário para a página em questão. Caso clique sobre um **Arquivo**, será dado início ao *download* daquele documento em questão. No caso de **Questionários**, o aluno poderá responder a uma série de perguntas objetivas mas que não possuem caráter avaliativo, conforme mostra a Figura 45. Caso o aluno já tenha respondido ao questionário, não poderá fazê-lo novamente. Caso o usuário seja um professor da turma, ao clicar em um questionário

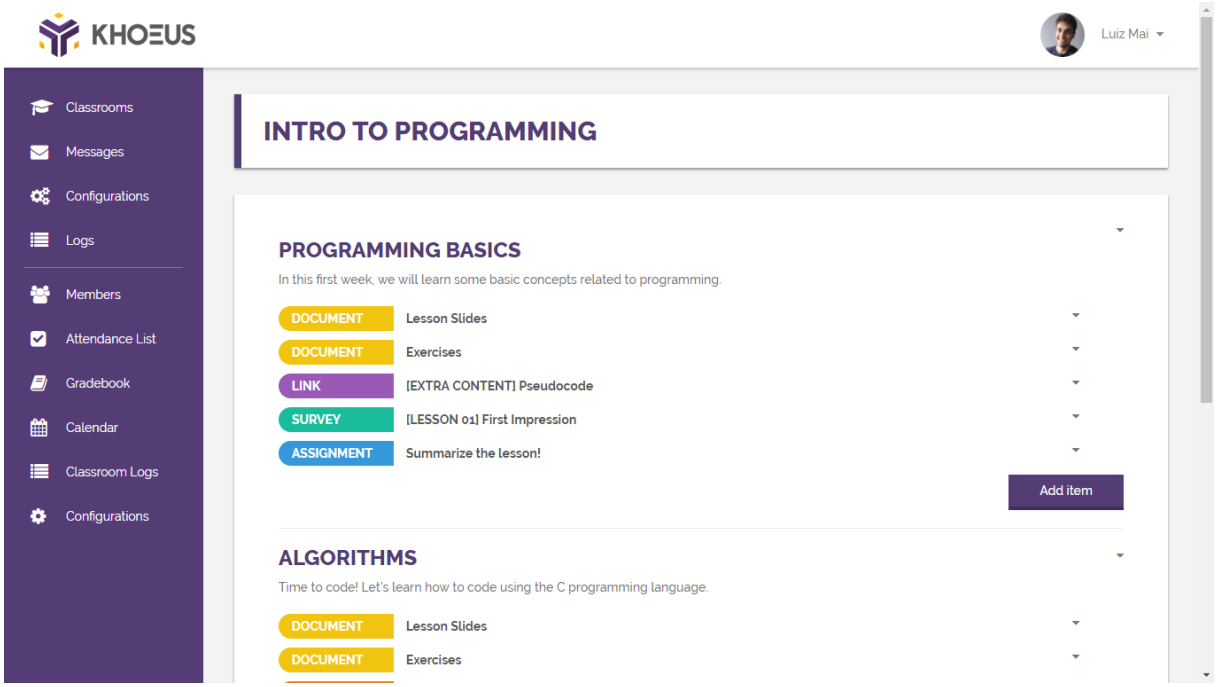


Figura 44 – Board de uma determinada turma

ele poderá conferir as estatísticas das respostas que foram dadas até o momento, conforme Figura 46.

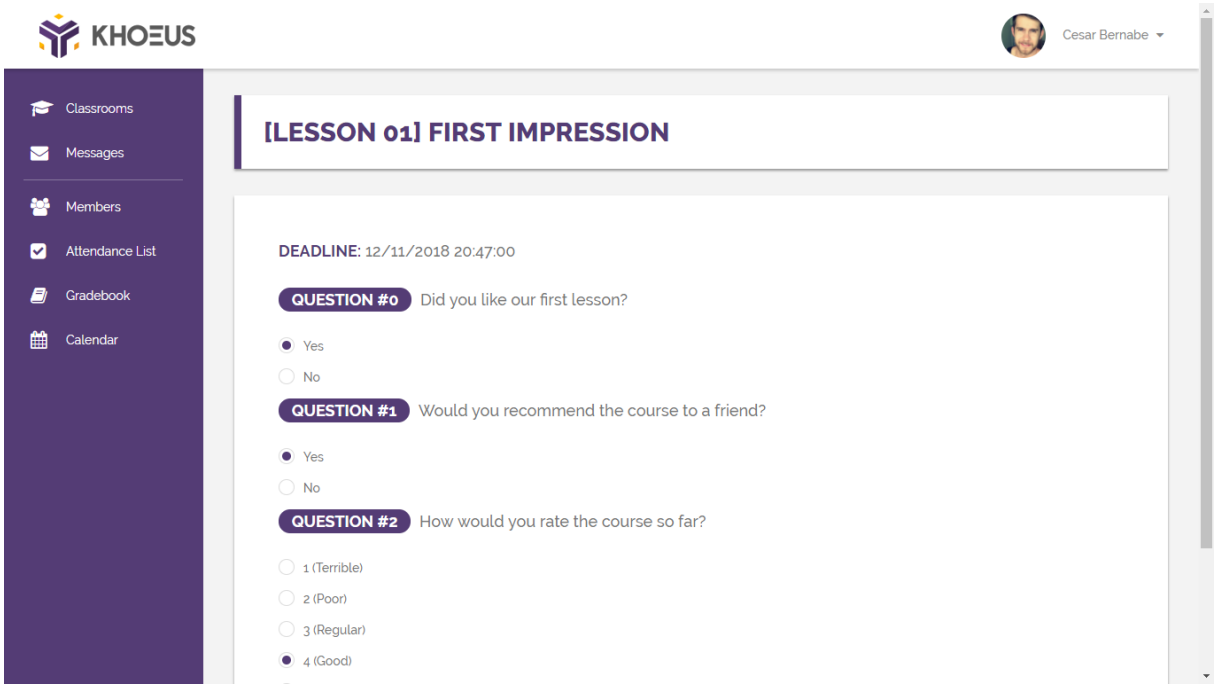


Figura 45 – Tela para que o aluno responda a um questionário

Ao clicar sobre uma **Prova**, o aluno será levado a uma tela contendo as diversas questões de uma prova, as quais deverá resolvê-la, conforme. Cada prova é composta por questão objetivas e/ou discursivas, sendo que cada uma delas possui um valor definido pelo professor no momento da criação. Caso o aluno já tenha resolvido a prova, ao clicar sobre o

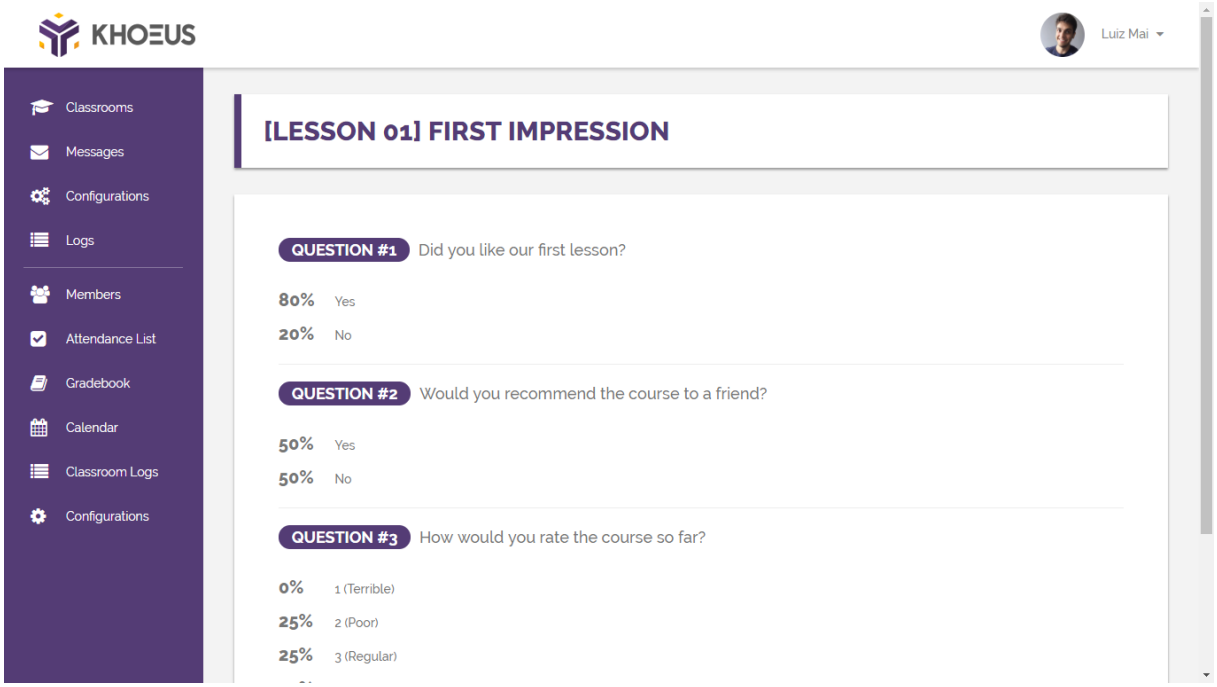


Figura 46 – Tela com as estatísticas de um questionário

item do *board* ele será levado para a tela contendo o *feedback* daquela prova (caso já tenha sido avaliada), conforme mostra a Figura 48. Caso o usuário seja um professor da turma, ele será levado a uma tela com a listagem de todos os alunos onde poderá escolher qual aluno deseja avaliar (Figura 49), sendo levado a uma tela onde poderá corrigir cada uma das respostas do aluno (no caso de questões discursivas), conforme mostra a Figura 50.

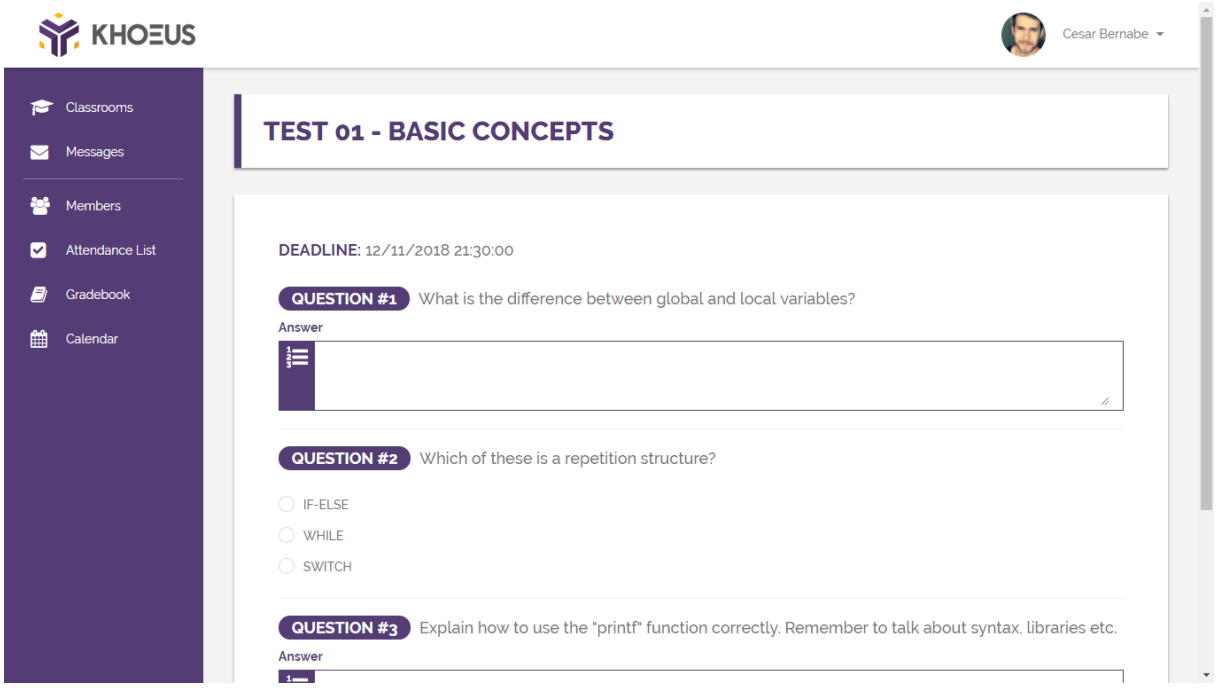


Figura 47 – Tela onde um aluno poderá resolver uma prova

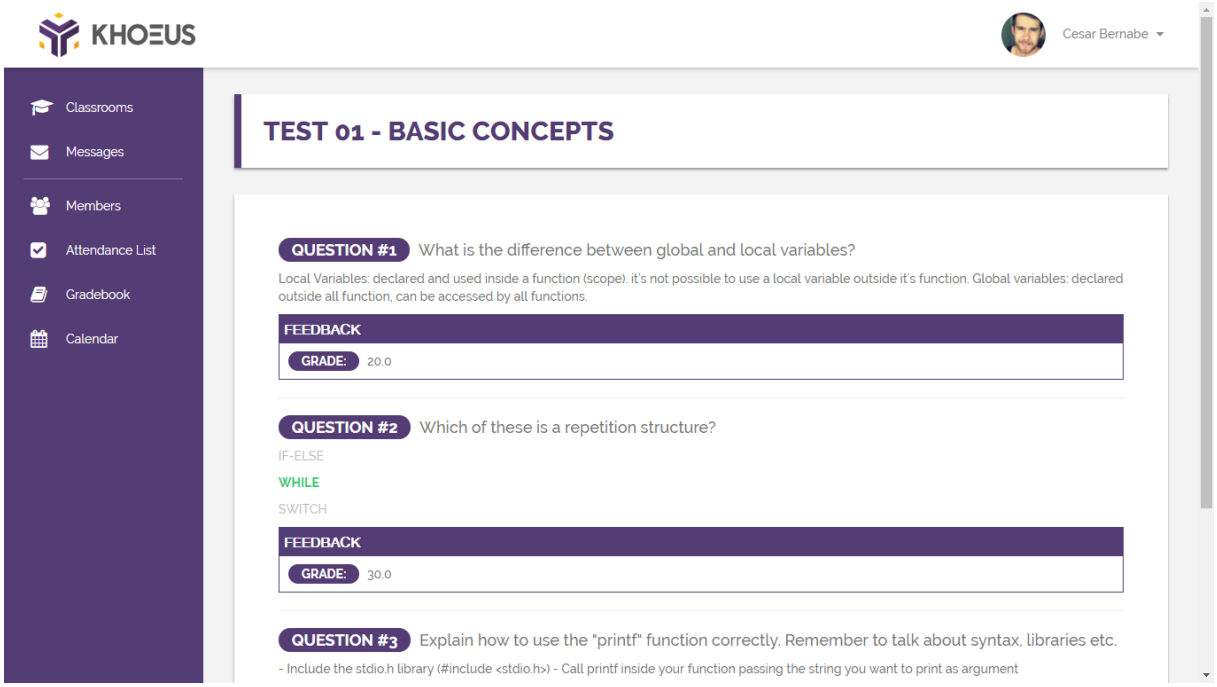


Figura 48 – Tela onde o aluno poderá consultar sua nota e comentários do professor

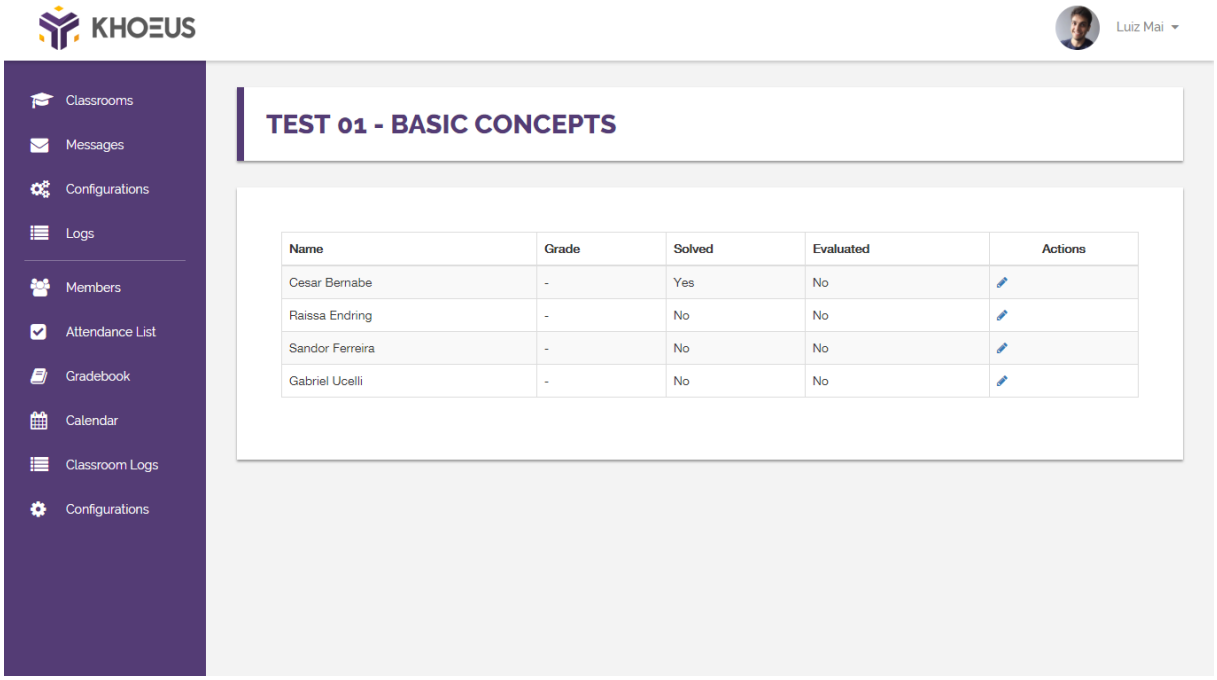


Figura 49 – Tela onde o professor poderá ver a listagem de todos os alunos

No caso de **Tarefas**, o aluno poderá se deparar com três tipos diferentes de submissão:

- **Texto:** o aluno deverá escrever um texto que responda o que foi solicitado na tarefa, conforme Figura 51.
- **Arquivo:** o aluno deverá enviar um arquivo contendo o conteúdo solicitado na tarefa,

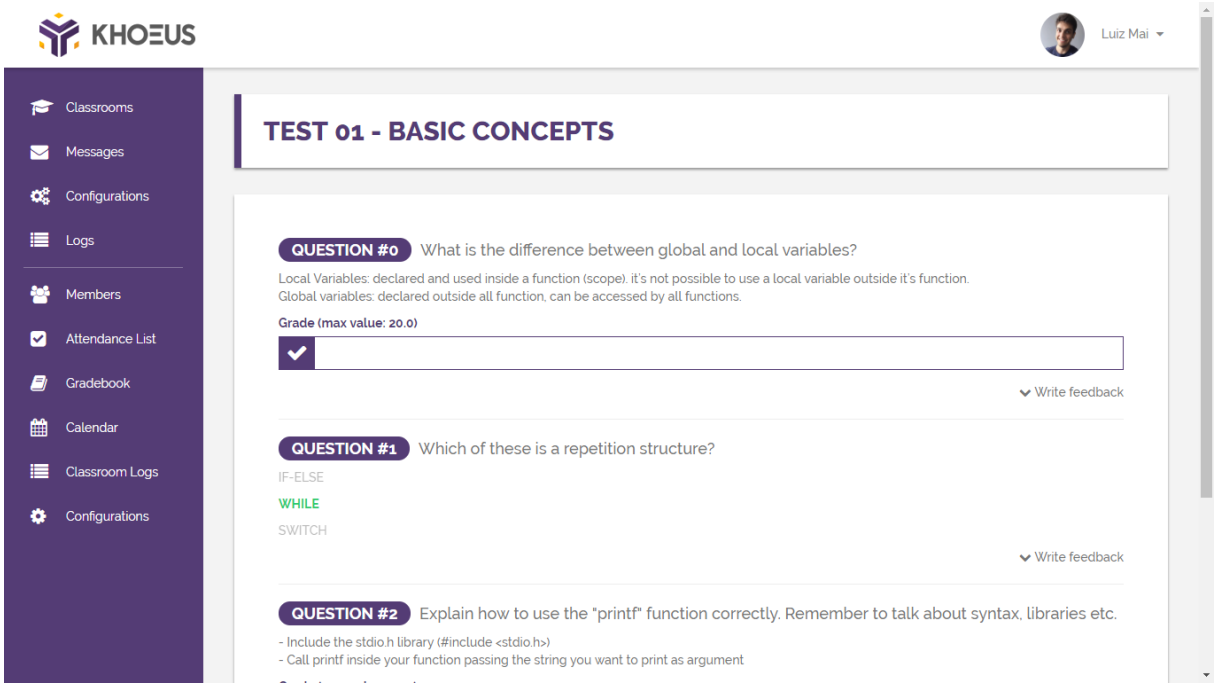


Figura 50 – Tela onde o professor poderá avaliar a prova de um aluno

conforme Figura 52.

- **Código:** o aluno poderá executar seu código antes de enviá-lo, conforme Figura 53.

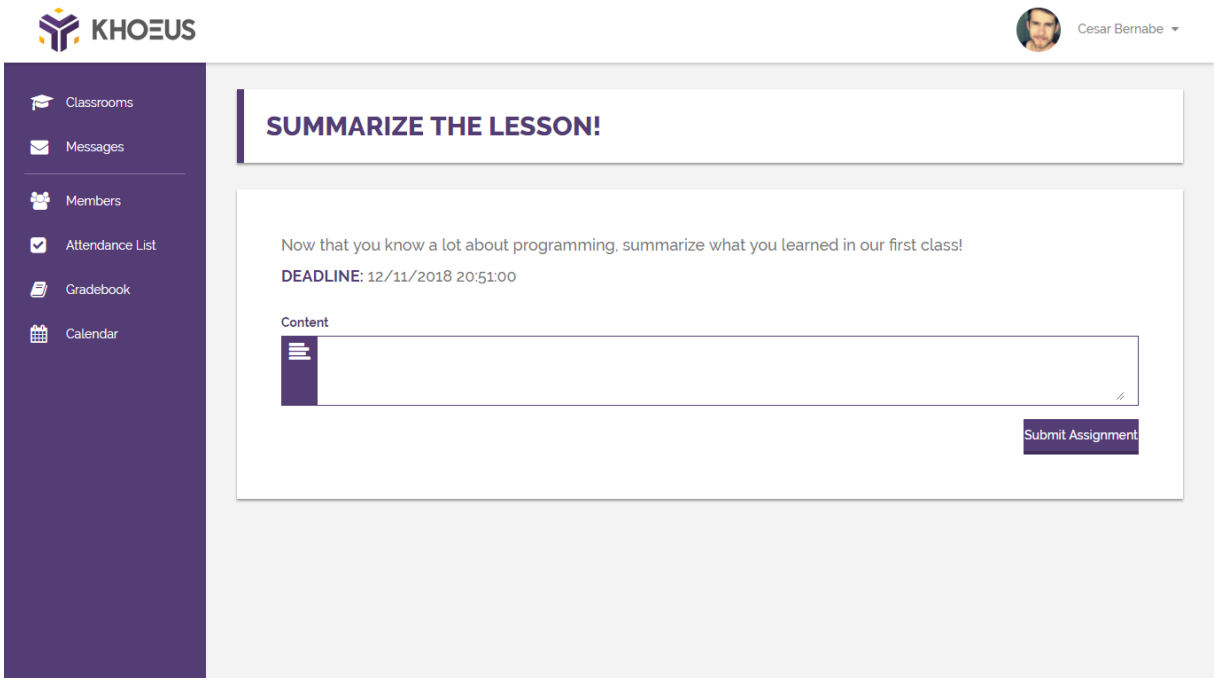


Figura 51 – Submissão de tarefas de texto

Caso o usuário seja um professor da turma, ele será levado a uma tela com a listagem de todos os alunos similar à mostrada na Figura 49 onde poderá escolher qual aluno deseja avaliar, sendo levado a uma tela onde poderá atribuir uma nota e um *feedback* à submissão

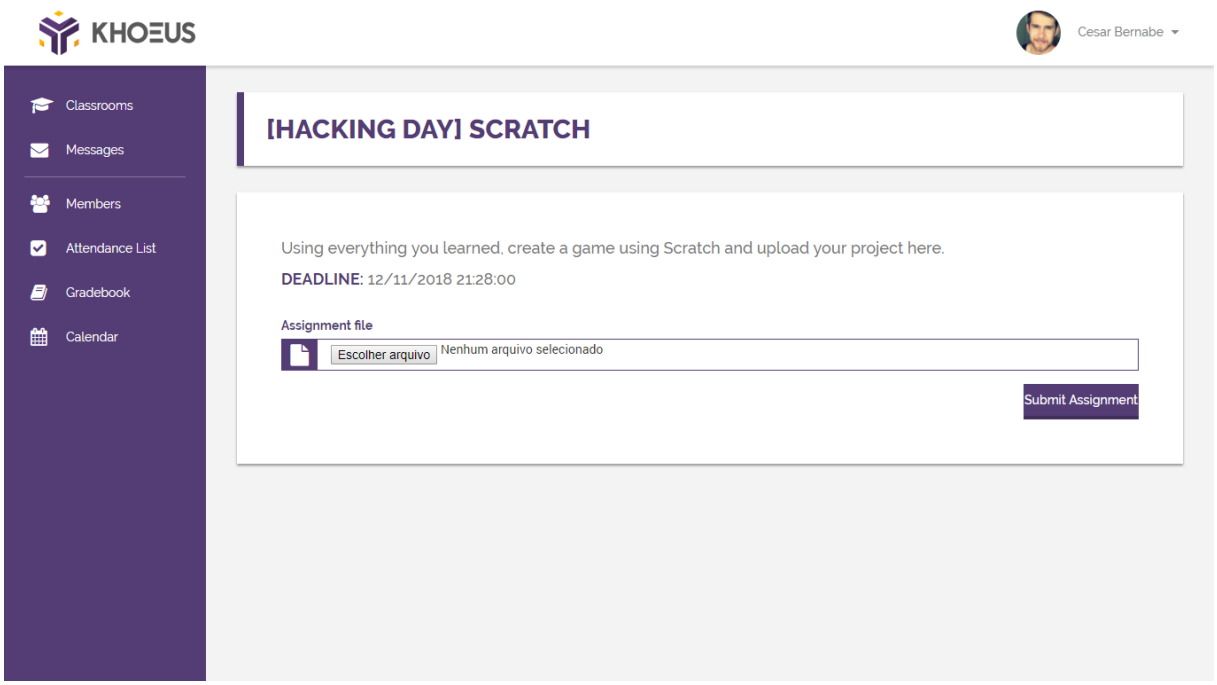


Figura 52 – Submissão de tarefas de arquivo

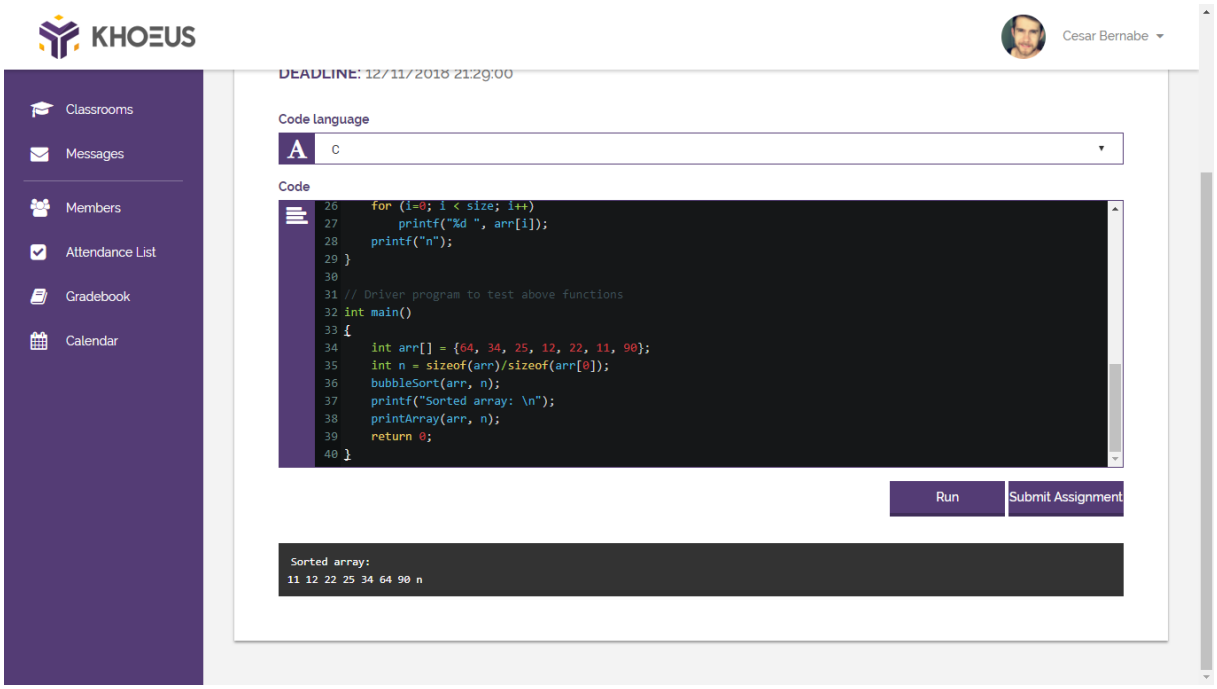


Figura 53 – Execução do código antes de submetê-lo

do aluno. No caso de tarefas de código, o professor poderá deixar um comentário em uma linha de código específica, caso julgue necessário (Figura 54). Uma vez avaliada a submissão de um aluno, ele poderá consultá-la pela página da tarefa, onde verá a sua nota e o comentário deixado pelo professor. No caso de tarefas de código, o aluno também poderá consultar os comentários feitos em linhas de códigos específicas, conforme a Figura 55.

Por fim, a interação com as atividades externas é muito mais simples uma vez

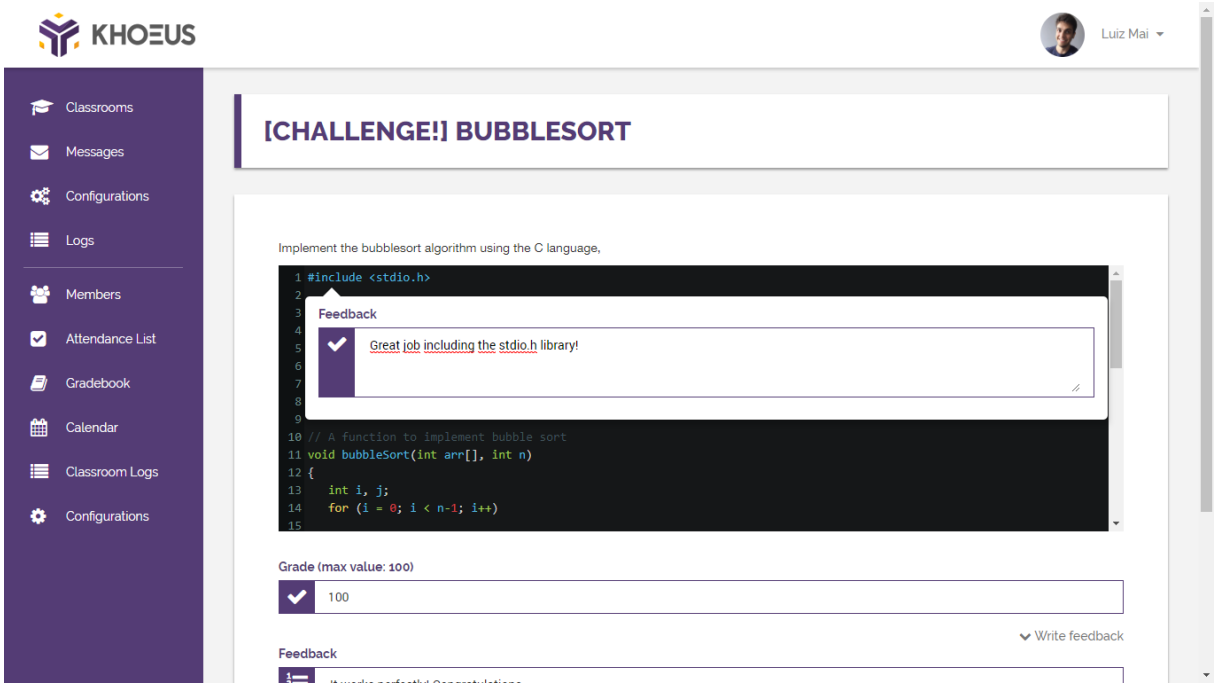


Figura 54 – Avaliação de tarefas de código

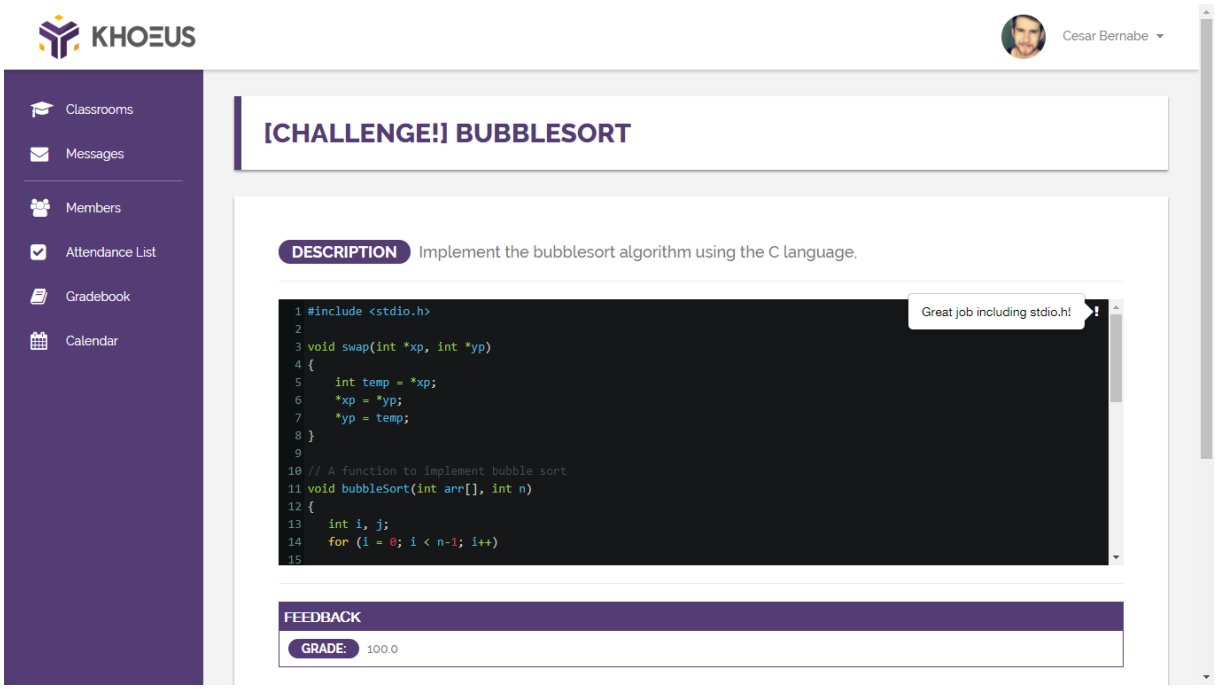


Figura 55 – Feedback atribuído a linhas de código individualmente

que assume-se que ela já foi realizada externamente ao sistema. Por isso, professores podem apenas atribuir uma nota aos alunos da turma (Figura 56) e alunos podem apenas consultar sua nota (Figura 57).

Ao longo do curso, os alunos poderão consultar seu Livro de Notas (Figura 58) onde são exibidas as notas em cada uma das atividades avaliativas que foram realizadas ao longo do curso.

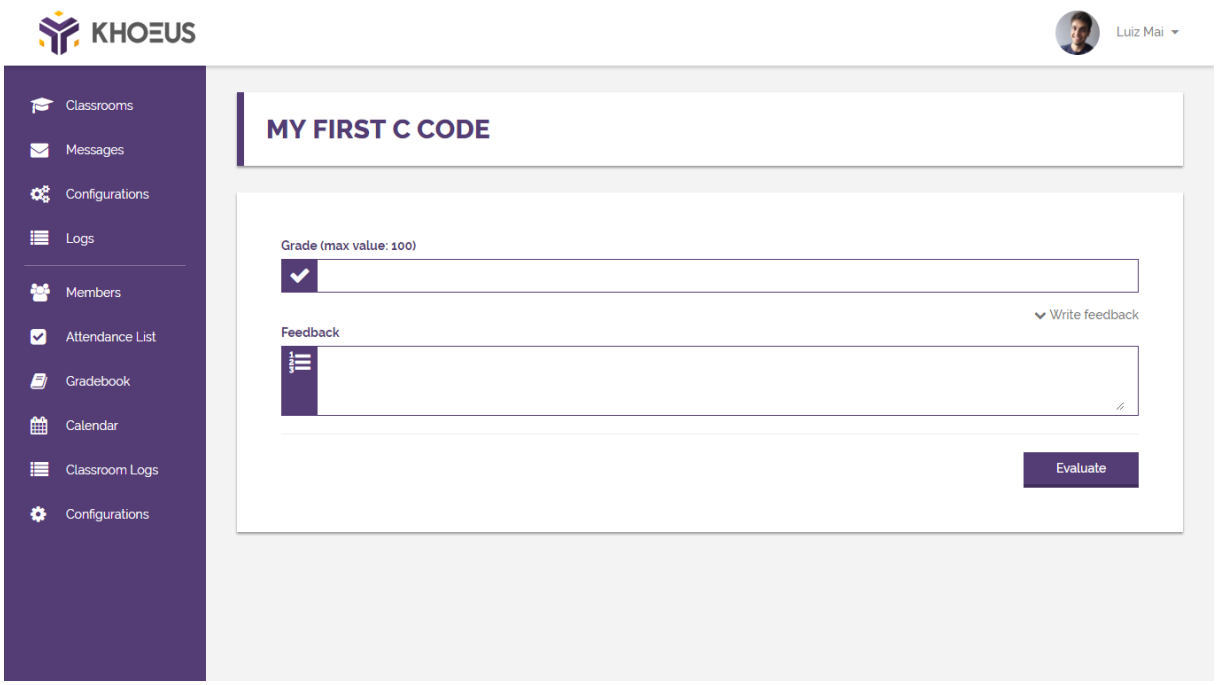


Figura 56 – Tela para avaliar atividades externas de um aluno

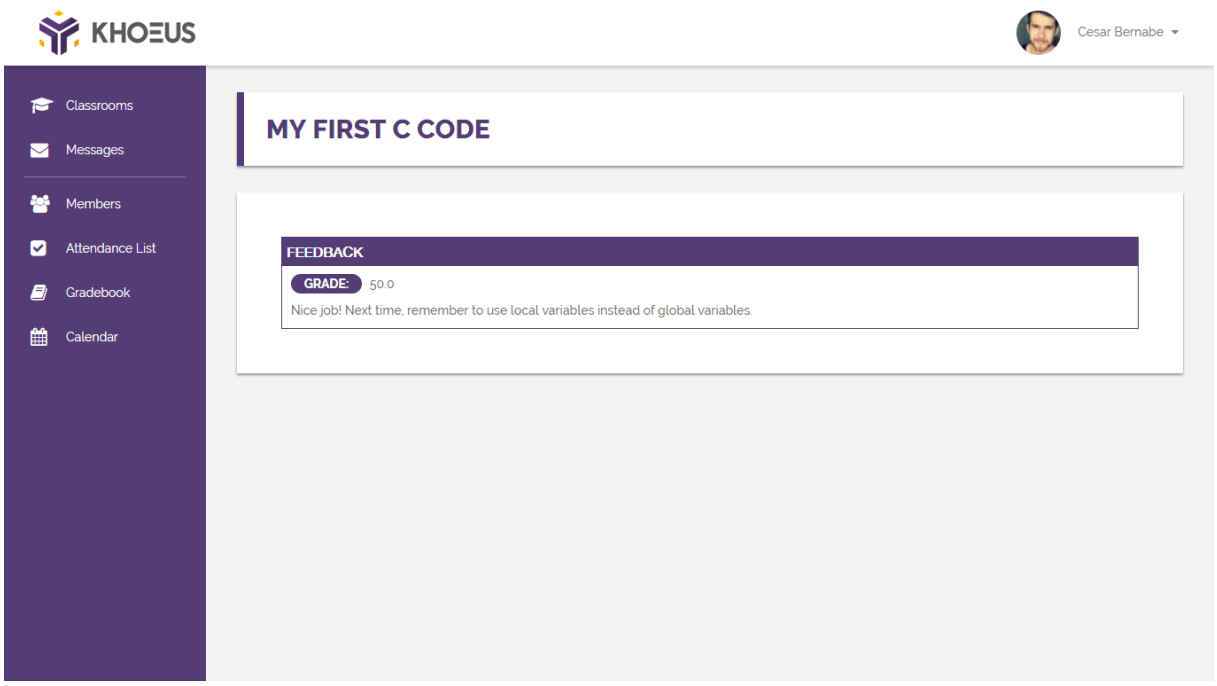


Figura 57 – Tela com a nota e o feedback de uma atividade externa

Para permitir que os alunos acompanhem melhor o curso, um professor poderá adicionar novas aulas ao calendário de aulas (Figura 59) por meio de um formulário onde deverá atribuir uma data para aquela aula.

Com base nas aulas definidas, um professor poderá atribuir presença ou ausência aos alunos de uma turma em cada uma das aulas por meio de um formulário conforme mostra a Figura 60. Ao longo do curso, os alunos poderão consultar a lista de presença

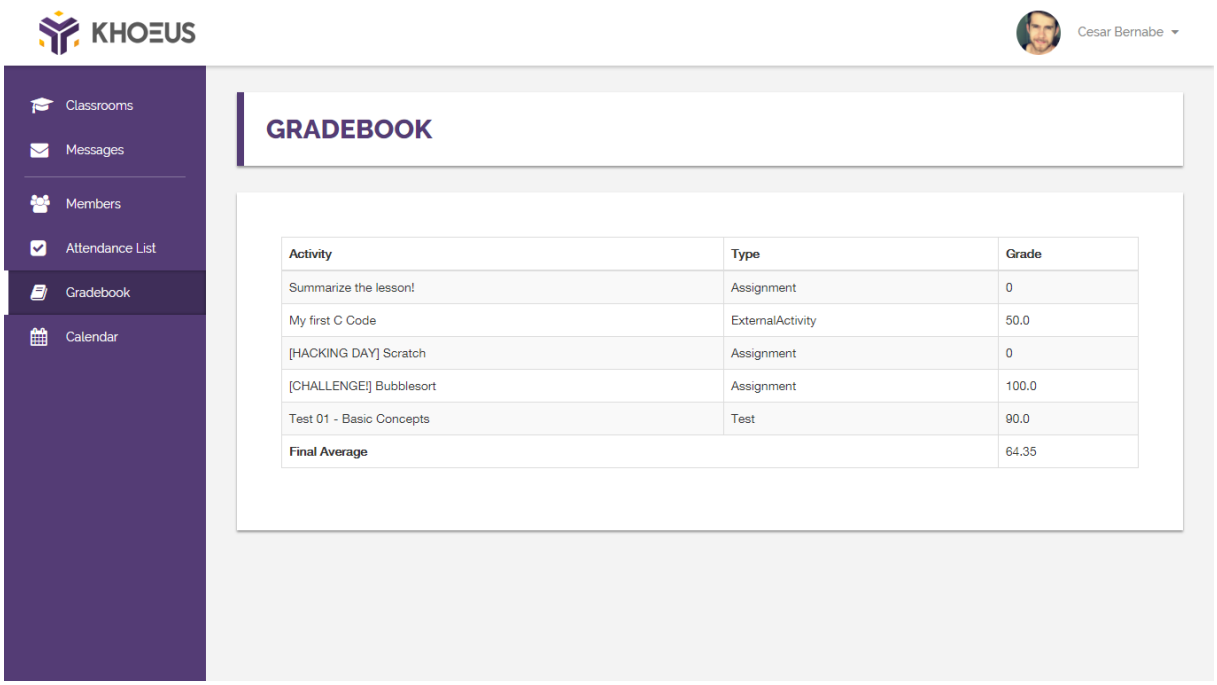


Figura 58 – Livro de notas

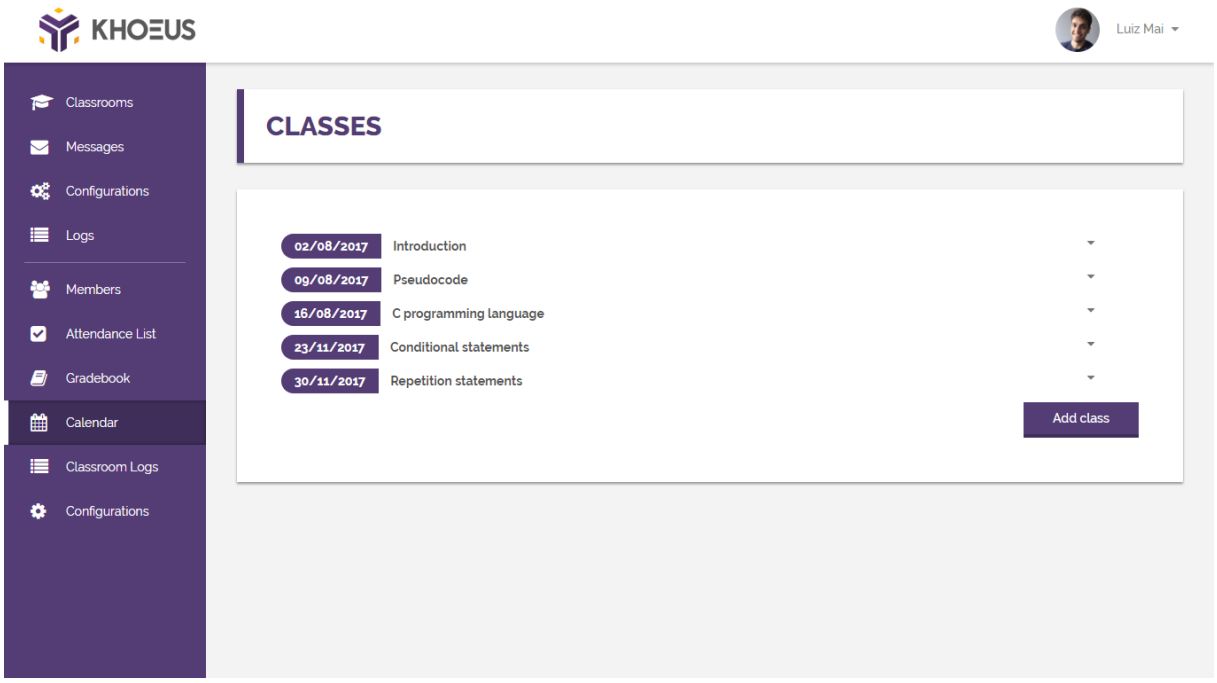


Figura 59 – Calendário de aulas

(incluindo a porcentagem de aulas em que esteve presente), conforme a Figura 61.

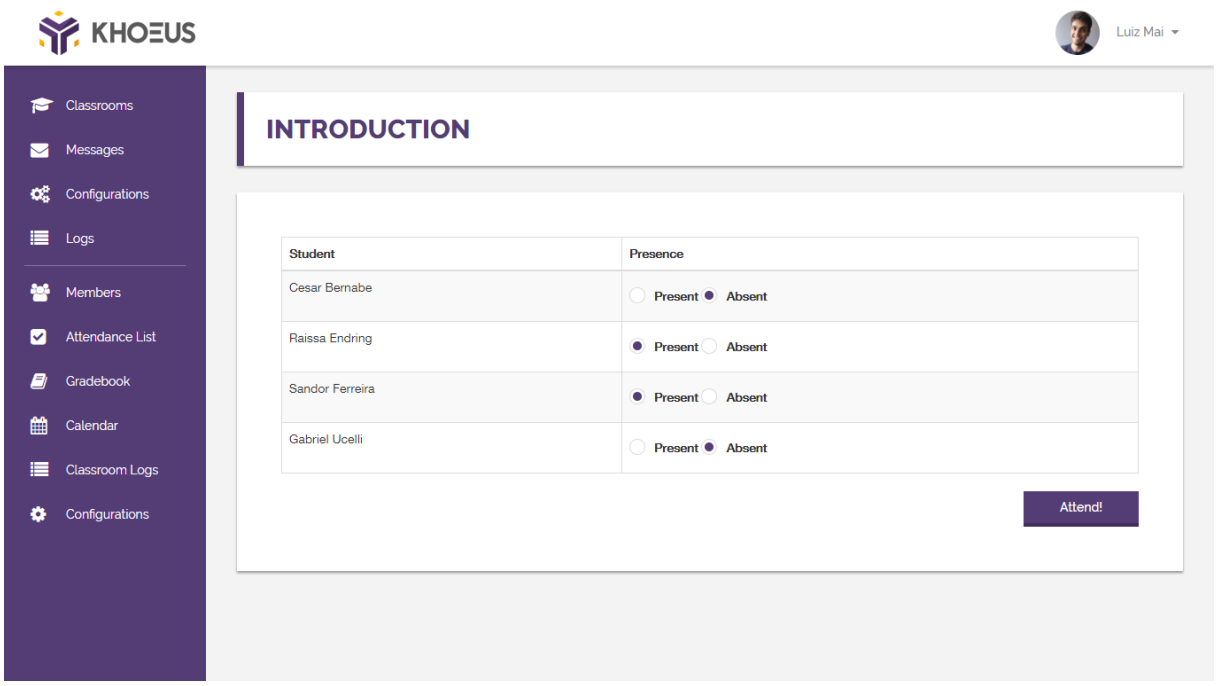


Figura 60 – Formulário para a atribuição de presença/ausência em uma determinada aula

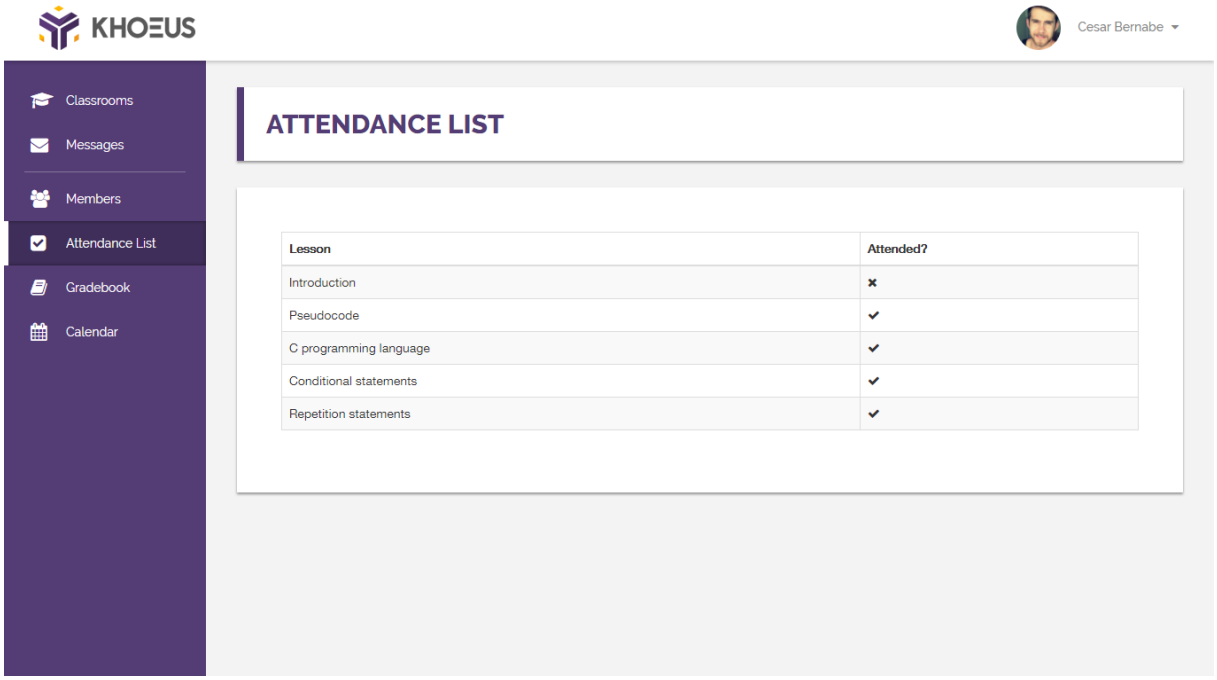


Figura 61 – Lista de presença de uma turma

# 6 Considerações Finais

Este capítulo apresenta as conclusões do trabalho realizado, mostrando suas contribuições. Por fim, são apresentadas suas limitações e perspectivas de trabalhos futuros.

## 6.1 Conclusões

Com a necessidade de um ambiente de apoio ao aprendizado simples e voltado ao ensino da programação por parte do Introcomp, viu-se a necessidade de implantar um sistema dessa natureza que, mais tarde, foi concebido como o Khoeus. Embora sua ideia inicial tenha sido a de ser utilizado no ensino da programação, notou-se a possibilidade e utilizá-lo para cursos das mais diversas áreas, mesmo que não relacionadas à programação. Como resultado, espera-se que o sistema seja utilizado pelo Introcomp em 2018 a fim de permitir um maior amadurecimento do sistema e a correção de quaisquer falhas que venham a ser encontradas nesse período.

Todos os objetivos levantados no Capítulo 1 foram atingidos, sendo que toda a documentação proposta foi produzida alinhado às boas práticas de Engenharia de Software. Foram levantados os requisitos, em seguida foi feita a análise de tais requisitos, produzindo, então, o Documento de Especificação de Requisitos. Este documento contém informações sobre requisitos funcionais, não funcionais, regras de negócio, descrição do sistema, definição de atores, casos de uso e diagramas de classe. Com o documento de requisitos finalizado, foi criado o Documento de Projeto, contendo todas as informações relacionadas à arquitetura do sistema. Nessa etapa foram criados os modelos propostos na Seção 2.4, seguindo a abordagem FrameWeb.

Ao longo de todo o desenvolvimento (desde a fase de documentação até a implementação propriamente dita), diversas dificuldades foram encontradas: o aprendizado de uma linguagem e de um *framework* desconhecidos até então foi, sem dúvida, a maior barreira encontrada, mas que pôde ser superada por meio da leitura de diversos livros e tutoriais na Internet; a utilização do padrão TDD para desenvolvimento de software também se mostrou de grande complexidade embora a teoria por trás dele fosse simples, o que levou à necessidade de praticar constantemente até atingir um grau de maturidade suficiente para utilizá-lo neste projeto; além disso, o *FrameWeb* se mostrou um método diferente dos quais são utilizados normalmente no desenvolvimento de software, mas com alguma leitura e prática, foi possível elaborar os modelos corretamente e em tempo hábil.

Com a utilização do FrameWeb, foi notável a facilidade agregada tanto na documentação como na implementação do sistema, principalmente com os modelos de entidade

e aplicação. Como o Ruby on Rails segue algumas convenções diferentes das esperadas pelo método, foi necessário realizar algumas pequenas adaptações como considerar as navegabilidades bidirecionais para todas as classes do Modelo de Entidades e criar uma camada de serviços (não fornecida por *default* pelo framework) para que o Modelo de Aplicação fosse melhor utilizado. No entanto, tais adaptações se mostraram simples de serem realizadas, mostrando a flexibilidade do método com os diversos *frameworks* existentes atualmente.

Por fim, mas não menos importante, faz-se necessário destacar a fundamentação adquirida ao longo de cinco anos de curso e que auxiliaram no desenvolvimento deste trabalho: disciplinas como Engenharia de Software, Projeto de Sistemas de Software e Programação se mostraram essenciais no desenvolvimento do raciocínio lógico e no processo de aprendizado. No final, foi possível agregar o conhecimento adquirido ao longo dessa longa jornada e produzir um artefato palpável.

## 6.2 Limitações e Perspectivas Futuras

O desenvolvimento de software é tido como um processo constante e que exige mudanças e correções ao longo de todo o seu ciclo de vida, implementando novas funcionalidades, descartando aquelas não mais viáveis, refatorando código não tão eficiente e, obviamente, corrigindo possíveis falhas que venham a ser encontradas. A partir dos resultados alcançados, foi possível verificar que algumas funcionalidades do sistema demandam mais atenção e que abrem as portas para a realização de trabalhos futuros. Essas limitações são apresentadas abaixo.

- Incluir as funcionalidades de fórum de discussões, lista de notícias e *Live Questions*, permitindo a discussão entre alunos e entre aluno-professor;
- Permitir a troca de mensagens entre usuários de uma mesma turma;
- Aplicar o Modelo de Persistência ao padrão utilizado pelo *Ruby on Rails* para a interação com o banco de dados;
- Aprimorar a funcionalidade de compilação/execução de código para outras linguagens.

# Referências

ANICHE, M. *TDD*. [s.n.], 2014. Disponível em: <http://tdd.caelum.com.br/>. Citado 2 vezes nas páginas 6 e 22.

ASSOCIATION, I. S. et al. Standard glossary of software engineering terminology. *lEEE Std*, p. 610–12, 1990. Citado na página 15.

AURUM, A. et al. *Managing software engineering knowledge*. [S.l.]: Springer Science & Business Media, 2013. Citado 2 vezes nas páginas 15 e 16.

FALBO, R. d. A. *Engenharia de Software*. [s.n.], 2014. 144 p. Disponível em: <https://inf.ufes.br/~falbo/files/ES/Notas_Aula_Engenharia_Software.pdf>. Citado na página 14.

FALBO, R. d. A. *Projeto de Sistemas*. [s.n.], 2016. 138 p. Disponível em: <https://inf.ufes.br/~falbo/files/PSS/Notas_Aula_Projeto_Sistemas_2016.pdf>. Citado na página 17.

FALBO, R. d. A. *Engenharia de Requisitos*. [s.n.], 2017. 178 p. Disponível em: <https://inf.ufes.br/~falbo/files/ER/Notas_Aula_Engenharia_Requisitos.pdf>. Citado 2 vezes nas páginas 16 e 17.

FOWLER, M. *Patterns of enterprise application architecture*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 2002. Citado 2 vezes nas páginas 18 e 23.

GINIGE, A.; MURUGESAN, S. Web engineering: An introduction. *IEEE multimedia*, IEEE, v. 8, n. 1, p. 14–18, 2001. Citado na página 18.

KOTONYA, G.; SOMMERVILLE, I. *Requirements engineering: processes and techniques*. [S.l.]: Wiley Publishing, 1998. Citado 3 vezes nas páginas 6, 15 e 16.

LEWIS, B. *Using Services to Keep Your Rails Controllers Clean and DRY*. 2017. Disponível em: <https://www.engineyard.com/blog/keeping-your-rails-controllers-dry-with-services>. Citado na página 58.

MARTINS, B. F. S. *Uma abordagem dirigida a modelos para o projeto de Sistemas de Informação Web com base no método FrameWeb*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2016. Citado na página 23.

MEJIA, A. *Ruby on Rails Architectural Design*. [s.n.], 2011. Disponível em: <http://adrianmejia.com/blog/2011/08/11/ruby-on-rails-architectural-design/>. Citado 2 vezes nas páginas 6 e 21.

PRESSMAN, R. S. *Engenharia de software*. [S.l.]: Makron books Sao Paulo, 2011. v. 7. Citado 3 vezes nas páginas 14, 15 e 17.

RUBY, S.; THOMAS, D.; HANSSON, D. H. *Agile Web Development with Rails 4*. [S.l.]: Pragmatic Programmers, 2013. Citado na página 21.

SALVATORE, T. R. *AlocaWeb e BibLattes - Módulos do sistema Marvin*. Monografia

(Projeto de Graduação) — Universidade Federal do Espírito Santo, 2016. Citado 2 vezes nas páginas 23 e 43.

SOUZA, V. E. S. *FrameWeb – A Framework-based Design Method for Web Engineering*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2007. Citado 3 vezes nas páginas 12, 23 e 49.

WIKIPEDIA. *MVC*. 2017. Disponível em: <http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm>. Citado 2 vezes nas páginas 6 e 20.

WINCKLER, M.; PIMENTA, M. S. Avaliação de usabilidade de sites web. *Escola de Informática da SBC SUL (ERI 2002) ed. Porto Alegre: Sociedade Brasileira de Computação (SBC)*, v. 1, p. 85–137, 2002. Citado na página 19.

# Apêndices


Documento de Especificação de Requisitos

# Khoeus - Plataforma de Aprendizado

Registro de Alterações:

| Versão | Responsável | Data | Alterações |
|---|---|---|---|
| 1.0 | Luiz Felipe F. Mai | 13/03/2017 | Versão inicial da documentação |
| 1.1 | Vítor E. Silva Souza | 17/03/2017 | *Revisado até Seção 4.* |
| 1.2 | Luiz Felipe F. Mai | 22/03/2017 | *Correções realizadas na versão 1.1* |
| 1.3 | Vítor E. Silva Souza | 17/04/2017 | *Revisado até Seção 6.* |
| 1.4 | Luiz Felipe F. Mai | 25/04/2017 | *Correções realizadas na versão 1.3* |
| 1.5 | Vítor E. Silva Souza | 15/05/2017 | *Revisado até o final.* |
| 1.6 | Luiz Felipe F. Mai | 30/05/2017 | *Correções realizadas na versão 1.4* |
| 1.7 | Vítor E. Silva Souza | 14/06/2017 | *Nova revisão completa.* |
| 1.8 | Luiz Felipe F. Mai | 27/06/2017 | *Redivisão dos subsistemas e correções realizadas na versão 1.7* |
| 2.0 | Luiz Felipe F. Mai | 16/12/2017 | *Correções após apresentação* |

Vitória, ES

2017

# 1 Introdução

Este documento apresenta os requisitos de usuário e a análise dos requisitos do sistema Khoeus. A atividade de análise de requisitos foi conduzida aplicando-se técnicas de modelagem de casos de uso, modelagem de classes e levando em consideração as boas práticas de programação relacionadas ao desenvolvimento de aplicações web. Os modelos apresentados foram elaborados usando a UML.

Na Seção 2 deste documento descreve-se de forma geral o sistema, apresentando superficialmente suas principais características. Já na Seção 3 estão apresentadas as funcionalidades do Khoeus, bem como suas dependências. A Seção 4 lista os requisitos de usuário do sistema (funcionais, não funcionais e suas regras de negócio). Além disso, a Seção 5 exibe os subsistemas considerados para este projeto enquanto a Seção 6 apresenta o modelo de casos de uso, incluindo descrições de atores, os diagramas de casos de uso e suas respectivas descrições. Na Seção 7 estão apresentados os modelos conceituais estruturais do sistema na forma de diagramas de classes. Por fim, a Seção 8 apresenta o dicionário do projeto, contendo as definições das classes identificadas.

# 2 Descrição do Propósito do Sistema

A possibilidade de um ambiente em que professores possam disponibilizar materiais de aula e requisitar tarefas dos alunos é bastante atraente para ambas as partes e, por isso, a utilização de plataformas de aprendizado tem se tornado cada vez maior em ambientes de ensino como escolas, faculdades ou cursos externos. No entanto, numa tentativa de atender a todos, as plataformas mais utilizadas atualmente acabam se tornando confusas e fornecendo ferramentas que, muitas vezes, não são utilizadas.

Não obstante a vasta gama de funcionalidades fornecidas por essas plataformas, nenhuma delas lida com códigos de programação de forma eficiente: em sua maioria, códigos são gerenciados como arquivos ou textos comuns, impedindo que seja possível explorar ao máximo o aprendizado do aluno na área de programação.

Dessa forma, o Khoeus vem com o intuito de facilitar o processo de aprendizado de alunos e, ao mesmo, tentar corrigir as falhas e os excessos das plataformas já existentes de forma a ser o mais simples e prático possível, agregando a possibilidade de trabalhar com códigos de programação.

# 3 Descrição do Minimundo

Em um ambiente de aprendizagem como uma escola ou universidade, espera-se que os alunos tenham acesso fácil e direto às informações relativas às aulas, como materiais de aula, notas das atividades realizadas e lista de presença. Para isso, espera-se centralizar essas funcionalidade no formato de turmas, de forma que cada uma delas possua seus alunos, professores, atividades, dentre outros elementos envolvidos no sistema.

## 3.1 Organização de turmas

A uma turma, estarão vinculados uma série de usuários que poderão exercer papel (*role*) de professor ou de aluno. Para participar de uma turma, um usuário deverá visitá-la e, em seu primeiro acesso, inserir a chave de acesso (que foi definida no momento em que a turma foi criada). Caso a chave esteja correta, o usuário é associado àquela turma e pode acessar seu conteúdo. Além disso, cada turma contém um *board*, que corresponde a todos os itens que serão exibidos aos alunos, como tarefas, arquivos, URLs, dentre outros (descritos na seção 3.3). Para melhor organizá-lo, um *board* é subdividido em seções definidas pelo professor e todos os itens deverão estar associados a uma seção. Um *board* contém obrigatoriamente:

- Uma **lista de notícias** que serão criadas exclusivamente pelos professores da turma. As notícias não possuem réplica.
- Um **fórum de discussão** com tópicos que podem ser tanto criados quanto respondidos por professores ou alunos da turma.
- Uma **lista de *LiveQuestions*** com perguntas que os alunos podem realizar durante as aulas
- A **lista de presença** dos alunos da turma
- O **livro de notas** contendo a avaliação de cada aluno em cada uma das atividades realizadas.

Além disso, o *board* também poderá conter:

- **Arquivos:** permite realizar o download de um arquivo que foi enviado pelo professor;
- **Links:** permite acessar páginas externas;

- **Questionários:** permite responder a perguntas objetivas sem que existam opções corretas, simulando uma enquete.
- **Provas:** permite responder a perguntas objetivas e discursivas que serão avaliadas.
- **Tarefas:** permite enviar arquivos, textos ou códigos de programação de acordo com a tarefa passada pelo professor.
- **Atividades Externas:** permite avaliar uma atividade que tenha sido realizada fora da plataforma, como uma atividade feita em sala de aula, por exemplo.

## 3.2 Controle de usuários

Todas as funcionalidades do sistema dependem do usuário estar logado em sua conta. Para isso, o Khoeus conta com uma página de login, uma de cadastro e uma de recuperação de senha, permitindo que os usuários possam acessar o sistema. Na página de cadastro, o usuário deverá informar seus dados pessoais (como nome, endereço e telefone), seu e-mail (que será utilizado para realizar o login) e uma senha. Feito isso, um e-mail de confirmação será enviado para o e-mail cadastrado a fim de que o usuário confirme seu cadastro. Na página de login, basta inserir o email e senha para que seja verificada sua autenticidade. Caso estejam corretos, o usuário é redirecionado para a lista de turmas. Caso contrário, deverá inserir novamente seus dados. Na página de recuperação de senha, basta que o usuário digite o e-mail cadastrado em sua conta para que receba um link em sua caixa de entrada permitindo a mudança de senha.

Um usuário do sistema pode ter permissão de **Administrador** ou de **Membro**. Enquanto os membros podem apenas alterar o próprio perfil, ingressar em uma turma ou acessar uma turma já ingressada anteriormente, administradores têm acesso completo ao sistema e podem alterar as configurações gerais do sistema, das turmas e editar o perfil de todos os usuários. Para o sistema, espera-se que seja possível configurar o fuso horário utilizado pelo sistema, definir as informações de SMTP para disparo de e-mails, colocar o sistema em modo de manutenção (acessível apenas para administradores). Para as turmas, espera-se poder definir seu título, sua senha de acesso, definir se a turma possui livro de notas e lista de presença além de ser possível definir as categorias de nota que serão utilizadas para a avaliação dos alunos (mais detalhes na Seção 3.4). Já para os alunos, espera-se ser possível alterar seu perfil (nome, senha, endereço, etc.). Além disso, os administradores têm acesso a um registro com logs de todo o sistema, incluindo todas as ações realizadas pelos usuários, como cadastro, login, criação de itens de board e até mesmo submissão de novas tarefas.

Além das permissões vinculadas à conta, os usuários também possuem um papel (*role*) em cada turma da qual fazem parte, podendo este ser de **Professor** ou **Aluno**.

Assim, um usuário com a permissão de membro pode assumir a posição de professor ou de aluno dentro de uma turma. Em uma turma, um aluno poderá consultar os itens do *board*, como enviar tarefas e baixar arquivos, consultar as próprias presenças na lista de chamada, consultar suas notas no livro de notas, verificar o calendário de aulas planejadas, criar novas *Live Questions* e enviar mensagens para alunos e professores. Paralelamente a isso, um professor pode inserir novos itens no *board*, lançar presenças e notas dos alunos cadastrados na turma, inserir uma nova aula no calendário, verificar as *Live Questions* vigentes, enviar mensagem para alunos e professores, realizar download em lote dos envios de uma tarefa, prova ou questionário, visualizar o log de atividades dos alunos e definir as configurações da turma, já descritas anteriormente.

Vale ressaltar que um usuário pode ter sua *role* alterada de acordo com a vontade de um administrador. Dessa forma, é possível que um Membro torne-se Administrador (ou vice-versa) e que um Aluno vire Professor (ou vice-versa).

## 3.3 Itens do Board

Itens do *board* estão sempre associados a uma seção com exceção do fórum de discussões, das notícias, das *Live Questions*, da lista de presença e do livro de notas, que são fixados no topo do *board*. Cada item do *board* possui uma ação diferente e possui campos diferentes no momento da inclusão.

### 3.3.1 Arquivo

Ao acessar um arquivo, o aluno realiza o download do mesmo. Para criar um novo arquivo, o professor deverá informar seu nome, a descrição e deverá fazer o upload a partir dos arquivos de seu computador.

### 3.3.2 Link

Ao acessar um link, o aluno é redirecionado para a URL correspondente. Para inserir um novo link no board, o professor deverá informar o título do link, sua descrição e a sua respectiva URL.

### 3.3.3 Notícias

No topo do board, o aluno pode acessar a lista de notícias relacionada àquela turma e onde estão todas as notícias já criadas pelos professores da turma. Para inserir uma nova noticia, o professor deverá informar apenas o título e o conteúdo desejado.

### 3.3.4 Fórum de Discussão

No fórum de discussão, tanto alunos quanto professores podem criar novos tópicos e, da mesma forma, responder tópicos já criados anteriormente. Para criar um novo tópico, espera-se que o usuário insira um título e seu conteúdo. No caso da réplica de um tópico, basta seu conteúdo.

### 3.3.5 *Live Questions*

Na lista de *Live Questions*, alunos poderão criar novas dúvidas e professores poderão visualizá-las por até 24h após sua criação. A ideia é que as dúvidas sejam criadas e sanadas no momento da aula. Na sua criação, o aluno deve inserir sua dúvida e informar se deseja ser tratado como Anônimo ou não.

### 3.3.6 Questionário

Os questionários simulam, basicamente, uma enquete ou uma pesquisa de opinião: no momento da criação, o professor define uma série de perguntas (todas de múltipla escolha) e as respostas possíveis para cada uma delas, sendo possível marcar apenas uma das alternativas. Nesse item, não existe resposta correta e, por isso, os questionários devem ser utilizados para recolher a opinião dos alunos sobre algum assunto determinado. Ao acessar um questionário, os alunos visualizarão as perguntas que foram criadas pelo professor e deverão respondê-las, não sendo possível responder a um questionário mais de uma vez. Questionários possuem uma data para iniciar e um tempo-limite para que possam ser respondidos. Após isso, é possível coletar os resultados do questionário.

### 3.3.7 Prova

Uma prova, como o próprio nome já diz, é elegível para ser uma forma de avaliação do aluno. Dessa forma, no momento da criação, o professor deve associá-la a uma categoria de nota e informar um título e uma descrição. Provas possuem uma data para iniciar e uma data de término que também deverão ser informadas. Além disso, o professor deve inserir quais perguntas haverão na prova, o tipo da questão (discursiva ou objetiva) e qual a nota máxima para cada uma das questões (totalizando 100 pontos). No caso de perguntas objetivas, deve-se informar qual das alternativas é a correta. Após encerrado o prazo da prova, os professores poderão visualizar as respostas dos alunos para cada uma das questões e atribuir a elas uma nota variando de 0 à nota máxima para aquela questão, sendo possível também informar um feedback para cada questão.

### 3.3.8 Tarefa

Tarefas representam a principal forma de avaliar um aluno, uma vez que permitem o envio de textos, arquivos ou até mesmo códigos de programas. No momento da criação da tarefa, o professor deverá informar o tipo de tarefa, a que categoria de nota esta pertence, título, descrição, data de início e data limite de envio. Uma vez finalizada, o professor poderá avaliar a tarefa de cada aluno concedendo-o uma nota de 0 a 100 e sendo possível ainda dar um feedback em forma de texto ou comentar linhas do código enviado pelo aluno.

A forma de submissão da tarefa varia de acordo com o seu tipo: caso seja do tipo texto, o aluno deverá preencher um campo de texto com o que foi pedido na descrição da tarefa. Caso seja do tipo arquivo, o aluno deverá fazer o upload do arquivo que foi requisitado. No caso da tarefa ser do tipo código, ele deverá inserí-lo no campo de texto, podendo realizar a compilação em tempo real a fim de verificar a corretude de seu código.

### 3.3.9 Atividade Externa

Atividades externas representam exercícios ou provas que foram feitas em sala e que, por isso, não são avaliados diretamente no Khoeus. Dessa forma, alunos não precisarão submeter quaisquer informações neste tipo de item (uma vez que esta foi feita presencialmente), mas o professor poderá lançar as notas de cada um dos alunos e incluir um pequeno feedback a respeito da atividade.

Atividades externas também devem estar vinculadas a uma categoria de nota.

## 3.4 Avaliação

Dentro das configurações de uma turma, é possível definir a forma com que a média final é calculada e quais as condições de aprovação dos alunos. Dessa forma, professores podem criar categorias de nota e associar a cada uma delas um peso no cálculo da média final. É possível criar, por exemplo, as categorias de nota “Prova” e “Trabalho” e definir a média final como 60% da nota de prova somado a 40% da nota de trabalho.

Para que a média seja calculada corretamente, o professor deverá definir no momento da criação de uma **Tarefa**, **Prova** ou **Atividade Externa** a que categoria de nota ela pertence.

# 4 Requisitos de Usuário

Tomando por base o contexto do sistema acima e considerando como principais *stakeholders* os professores e alunos do Introcomp e de outros cursos relacionados ao ensino de programação, foram identificados os seguintes requisitos de usuário e regras de negócio:

Tabela 1 – Requisitos Funcionais

| ID | Descrição | Prioridade | Depende |
|---|---|---|---|
| RF-1 | O sistema deve permitir que administradores possam definir as configurações de sistema (fuso horário, definições SMTP e modo de manutenção). Essas configurações devem ser armazenadas de forma genérica, de forma que seja guardada o nome da configuração e seu respectivo valor. | Alta | |
| RF-2 | Administradores devem poder criar novas turmas informando seu título, sua senha de acesso e se possui livro de notas e lista de presença. | Alta | |
| RF-3 | O sistema deve permitir que professores possam definir as configurações de turmas (título, senha de acesso, categorias de nota e existência de livro de notas e lista de presença). Deve-se também ser definida a forma com que a média final será calculada (baseada nas categorias de nota, informando o título e o peso de cada uma delas) e qual a condição para a aprovação dos alunos inscritos. | Alta | RF-6 |
| RF-4 | O sistema deve permitir que alunos se inscrevam em uma turma por meio de uma senha associada a ela. | Alta | RF-7 |
| RF-5 | O sistema deve permitir que administradores possam gerenciar os usuários dentro das turmas, sendo possível alterar sua *role* de aluno para professor ou vice-versa. | Alta | RF-8 |
| RF-6 | Professores devem ser capazes de criar novas seções no board de uma turma. | Alta | RF-9 |
| RF-7 | Professores devem ser capazes de adicionar novos arquivos ao *board* de sua turma e configurar corretamente seu título, descrição e realizar o *upload* do arquivo em questão. | Alta | RF-10 |

| | | | |
|---|---|---|---|
| RF-8 | Professores devem ser capazes de adicionar novos links ao *board* de sua turma e configurar corretamente seu título, descrição e informar a URL do link em questão. | Alta | RF-10 |
| RF-9 | Professores devem ser capazes de adicionar novos questionários ao *board* de sua turma e configurar corretamente seu título, descrição, data de início e data de término. Além disso, o professor deve informar todas as perguntas e suas respectivas respostas. | Alta | RF-10 |
| RF-10 | Professores devem ser capazes de adicionar novas provas ao *board* de sua turma e configurar corretamente seu título, descrição, data de início e data de término. Além disso, o professor deve informar todas as questões da prova, seu tipo (discursiva ou objetivo) e sua pontuação (a soma da pontuação de todas as questões deve totalizar 100 pontos). No caso de perguntas objetivas, o professor deve informar todas as alternativas e qual delas é a correta. | Alta | RF-10 |
| RF-11 | Professores devem ser capazes de adicionar novas tarefas ao *board* de sua turma e configurar corretamente seu título, descrição, data de início, data de término e tipo (texto, arquivo ou código). Caso seja do tipo Arquivo, o professor também deverá informar o número máximo de arquivos que podem ser enviados. | Alta | RF-10 |
| RF-12 | Professores devem ser capazes de adicionar novas atividades externas ao *board* de sua turma e configurar corretamente seu título e descrição. | Alta | RF-10 |
| RF-13 | Professores devem ser capazes de publicar notícias no *board* de uma turma. | Alta | RF-10 |
| RF-14 | Professores e alunos devem ser capazes de visualizar, publicar e responder tópicos no *board* de uma turma. | Alta | RF-10 |
| RF-15 | Professores devem ser capazes de visualizar *live questions* no *board* de uma turma. | Alta | RF-30 |
| RF-16 | As *Live Questions* deverão ser deletadas automaticamente decorrido um dia de sua criação. | Alta | RF-30 |
| RF-17 | Professores devem ser capazes de gerenciar as aulas e os eventos de uma turma por meio do calendário, inserindo, removendo e alterando-os. | Alta | RF-9 |

| | | | |
|---|---|---|---|
| RF-18 | Alunos devem ser capazes de fazer download dos arquivos de sua turma. | Alta | RF-11 |
| RF-19 | Alunos devem ser capazes de acessar os links de sua turma. | Alta | RF-12 |
| RF-20 | Alunos devem ser capazes de responder aos questionários de sua turma. | Alta | RF-13 |
| RF-21 | Alunos devem ser capazes de resolver as provas de sua turma. | Alta | RF-14 |
| RF-22 | Alunos devem ser capazes de submeter as tarefas de sua turma. No caso de tarefas do tipo Código, os alunos devem ter acesso a um compilador acoplado ao sistema para que possam testar seu código antes de submetê-lo. | Alta | RF-15 |
| RF-23 | Alunos devem ser capazes de consultar as aulas e eventos no calendário de sua turma. | Alta | RF-21 |
| RF-24 | Após encerrado um questionário, professores devem poder consultar seu resultado. | Alta | RF-24 |
| RF-25 | Alunos devem ser capazes de visualizar as notícias no *board* de sua turma. | Alta | RF-17 |
| RF-26 | Alunos devem ser capazes de criar novas *Live Questions* no *board* de sua turma. | Alta | RF-10 |
| RF-27 | O sistema deve permitir que professores realizem download em lote das submissões de uma prova ou tarefa. | Alta | RF-25 e RF-26 |
| RF-28 | Professores devem ser capazes de avaliar provas, tarefas e atividades externas, atribuindo-lhes uma nota e, opcionalmente, um feedback. Essas avaliações poderão ser feitas individualmente (aluno por aluno) ou em lote (via planilha externa). | Alta | RF-26 |
| RF-29 | Deve ser possível que alunos consultem sua nota em uma prova (assim como a pontuação em cada questão) a qualquer momento após sua data limite. | Alta | RF-32 |
| RF-30 | Deve ser possível que alunos consultem sua nota e seu respectivo feedback em uma tarefa a qualquer momento após sua data limite. | Alta | RF-32 |
| RF-31 | Deve ser possível que alunos consultem sua nota e seu respectivo feedback em uma atividade externa a qualquer momento após sua data limite. | Alta | RF-32 |

| | | | |
|---|---|---|---|
| RF-32 | Deve ser possível que alunos consultem suas notas por meio do livro de notas, sendo possível ver todas as suas notas ao longo do curso e sua média parcial até o momento. | Alta | RF-32 |
| RF-33 | Professores devem ser capazes de lançar as presenças de seus alunos em cada aula no sistema. | Alta | RF-21 |
| RF-34 | Deve ser possível que alunos consultem suas presenças em aula a qualquer momento. | Alta | RF-37 |
| RF-35 | O sistema deve permitir que membros de uma turma enviem mensagens entre si. | Média | RF-9 |
| RF-36 | O sistema deve permitir que professores de uma turma gerem logs com os acessos, visualizações e submissões de seus alunos. | Baixa | RF-9 |
| RF-37 | O sistema deve permitir que administradores gerem relatórios com os acessos, visualizações e interações dos professores e alunos. | Baixa | RF-9 |

Tabela 2 – Requisitos Não Funcionais

| ID | Descrição | Categoria | Escopo | Prioridade |
|---|---|---|---|---|
| RNF-1 | A plataforma deve estar disponível como uma aplicação Web, acessível a partir dos principais navegadores disponíveis no mercado. | Portabilidade | Sistema | Alta |
| RNF-2 | O sistema deve controlar o acesso às suas funcionalidades por meio de autenticação e autorização, utilizando o papel de cada usuário para definir a que funcionalidades ele terá acesso. | Segurança | Sistema | Alta |
| RNF-3 | Todas as informações sensíveis (como senhas e tokens de acesso) devem ser criptografados. | Confidenciabilidade | Sistema | Alta |
| RNF-4 | A plataforma dever ser de aprendizado fácil, não sendo necessário nenhum treinamento especial para seu uso. | Facilidade de Aprendizado | Sistema | Alta |
| RNF-5 | O sistema deve garantir a consistência de dados por meio da obrigatoriedade no momento da inserção de certos dados. | Confiabilidade | Sistema | Alta |

| | | | | |
|---|---|---|---|---|
| RNF-6 | A plataforma deve ser de fácil operação, não sendo necessário uso contínuo para uma boa operação do sistema. | Usabilidade | Sistema | Média |
| RNF-7 | O desenvolvimento do sistema deve facilitar manutenções futuras, utilizando padrões de design de software. | Redigibilidade | Sistema | Média |

Tabela 3 – Regras de Negócio

| ID | Descrição | Prioridade | Depende |
|---|---|---|---|
| RN-1 | Um usuário não pode estar associado às *roles* de aluno e professor simultaneamente em uma mesma turma. | Alta | RF-9 |
| RN-2 | Um item de *board* precisa estar, necessariamente, associado a uma seção do *board*. | Alta | RF-11, RF-12, RF-13, RF-14, RF-15 e RF-16 |
| RN-3 | Alunos não poderão submeter questionários, tarefas e provas após a data limite. | Alta | RF-24, RF-26 e RF-25 |
| RN-4 | Professores não poderão avaliar tarefas e provas antes da data limite. | Alta | RF-32 |
| RN-5 | Não é possível enviar um número de arquivos superior ao limite estipulado em uma tarefa. | Alta | RF-26 |
| RN-6 | A nota de uma prova, tarefa ou atividade externa não poderá ser superior a 100. | Alta | RF-32 |
| RN-7 | A somatória do peso das categorias de notas deve ser igual a 100. | Alta | RF-7 |
| RN-8 | Usuários não podem trocar mensagens com alguém que não esteja em uma mesma turma que ele. | Alta | RF-39 |

# 5 Identificação de Subsistemas

Tal projeto foi dividido em 4 subsistemas a fim de facilitar o desenvolvimento de funcionalidades que são, de certa forma, isoladas, originando assim os subsistemas Auth, Log, Classroom e Board. A Figura 1 mostra os subsistemas identificados no contexto do presente projeto e suas interdependências, enquanto a Tabela 4 apresenta breve descrição de cada um deles.

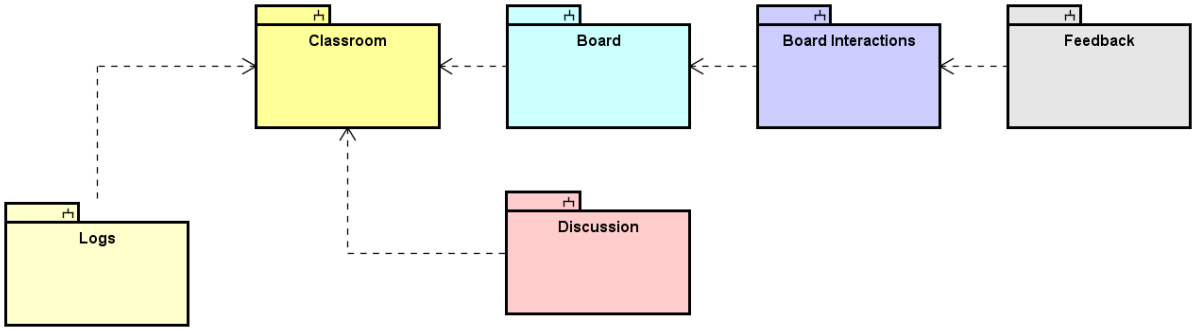


Figura 1 – Diagrama de Pacotes e os Subsistemas Identificados.

Tabela 4 – Subsistemas

| Subsistema | Descrição |
|---|---|
| Classroom | Subsistema contendo as funcionalidades relacionadas diretamente às turmas. |
| Board | Envolve todas as funcionalidades relativas ao gerenciamento dos itens do board. |
| Board Interactions | Corresponde a todas as interações que os usuários podem realizar com os itens do board, como submeter tarefas ou realizar provas. |
| Feedback | Permite a avaliação de atividades e a visualização das notas do aluno em uma turma. |
| Discussions | Subsistema contendo as funcionalidades relativas ao fórum de discussão, às notícias e às *Live Questions*. |
| Logs | Este sistema consiste na visualização de logs gerados a partir das atividades dos usuários do sistema. |

# 6 Modelo de Casos de Uso

O modelo de casos de uso corresponde a uma tentativa de descrever a relação das funcionalidades do sistema com cada um de seus atores. Os atores identificados no contexto deste projeto estão descritos na Tabela 5.

Tabela 5 – Atores

| Ator | Descrição |
|---|---|
| Visitante | Qualquer pessoa que acesse o sistema e não tenha realizado o *login*. |
| Administrador | Usuário que realizou o *login* e pode gerenciar todo o sistema. |
| Membro | Usuário que realizou o *login* no sistema mas que não possui permissões para configurá-lo. |
| Aluno | Corresponde a um Membro vinculado a alguma turma com permissões de aluno, como enviar tarefas e consultas notas. |
| Professor | Corresponde a um Membro vinculado a alguma turma com permissões de professor, como criar itens do *board* e configurar as informações da turma. |

A seguir, são apresentados os diagramas de casos de uso e descrições associadas, organizados por subsistema.

## 6.1 Subsistema Classroom

A Figura 3 apresenta o diagrama de casos de uso do subsistema Classroom.

Com exceção do UC-10, todos os outros casos de uso abaixo mencionados geram um *Log* indicando o usuário que realizou a ação, qual ação foi tomada e o momento em que isso aconteceu. Além disso, cada *Log* fica associado a uma determinada categoria. Por exemplo, o UC-15 pertenceria à categoria "Envio de mensagens".

A seguir, são apresentadas as descrições de cada um dos casos de uso identificados. Os casos de uso cadastrais de baixa complexidade, envolvendo inclusão, alteração, consulta e exclusão, são descritos na Tabela 14.

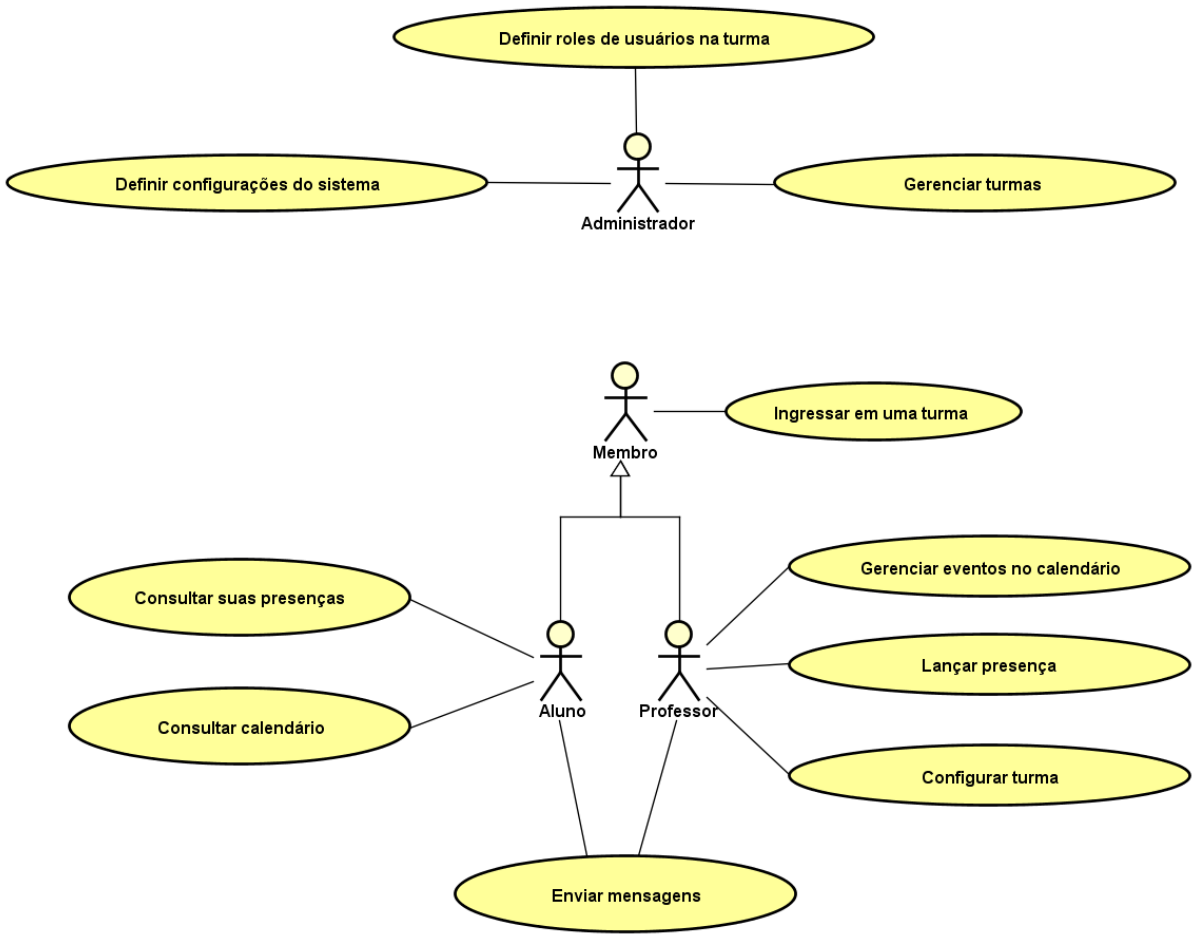


Figura 2 – Diagrama de Casos de Uso do subsistema Classroom.

Tabela 6 – Casos de Uso Cadastrais

| Id | Nome | Ações | Observações | Requisitos | Classes |
|---|---|---|---|---|---|
| UC-1 | Gerenciar turmas | I | Informar: título e senha de acesso. Opcionalmente, também é possível escolher professores da turma caso estes já estejam cadastrados no sistema. | RF-6 | Classroom |
| | | A | | | |
| | | C | | | |
| | | E | A exclusão de uma turma deve acarretar na exclusão de todos os itens de *board* e suas interações (como submissões de tarefa e respostas de provas). | | |
| UC-2 | Gerenciar eventos no calendário | I | Informar: título, descrição, data e tipo (Evento ou Aula). | RF-21 | Classroom, Event, Class |
| | | A | | | |
| | | C | | | |
| | | E | | | |

Os casos de uso de consulta mais abrangente que as consultas a um único objeto, mas ainda de baixa complexidade, tais como consultas que combinam informações de vários objetos envolvendo filtros, estão descritos na Tabela 15.

Tabela 7 – Casos de Uso de Consulta

| Id | Nome | Observações | Requisitos | Classes |
|---|---|---|---|---|
| UC-3 | Consultar suas presenças | Os alunos de uma turma terão acesso ao seu quadro de presenças contendo a lista de aulas do curso em que estão inscritos e se estiveram presentes (ou não) nas aulas que já ocorreram. | RF-38 | User, Classroom, Class e Presence |
| UC-4 | Consultar calendário | Os alunos de uma turma terão acesso ao calendário contendo as aulas e eventos registrados para aquela turma. Para cada um, exibe-se o nome, a descrição (caso tenha) e a data em que acontecerá. | RF-27 | User, Classroom, Class e Event |

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-5
**Caso de Uso:** Definir configurações do sistema
**Descrição Sucinta:** Este caso de uso permite que o usuário altere as principais configurações do sistema, acessíveis apenas para administradores.

Tabela 8 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Definir configurações | O usuário deverá estar logado e ser um administrador | 1. O administrador poderá alterar todas as configurações do sistema listadas no RF-5<br>2. Ao finalizar, o sistema salva todas essas alterações de forma genérica, armazenando o nome da configuração bem como seu respectivo valor. |

**Requisitos Relacionados:** RF-5
**Classes Relacionadas:** User, Configuration.

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-6
**Caso de Uso:** Ingressar em uma turma
**Descrição Sucinta:** Este caso de uso permite que um usuário se vincule a uma turma por meio de uma chave de inscrição.

Tabela 9 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Ingressar em uma turma | O usuário deverá estar logado | 1. O usuário deverá escolher a turma que deseja ingressar e informar a senha de acesso fornecida por um dos professores.<br>2. O sistema vincula aquele usuário àquela turma em uma *role* de aluno, garantindo que este possa acessá-la a qualquer momento. |

Tabela 10 – Fluxos de Eventos Variantes

| Nome do Fluxo | Variante | Descrição |
|---|---|---|
| Chave incorreta | O usuário digitou a chave de inscrição incorreta | 1. O sistema informa que a chave de inscrição está errada e solicita que o usuário tente novamente. |

**Requisitos Relacionados:** RF-8
**Classes Relacionadas:** User, Classroom, Subscription.

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-7
**Caso de Uso:** Lançar presença
**Descrição Sucinta:** Este caso de uso permite que o professor lance a presença dos alunos de sua turma para cada uma das aulas cadastradas no calendário.

Tabela 11 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Lançar presença | O usuário deverá estar logado e ser um professor da turma correspondente | 1. O usuário deverá informar em uma tabela aluno x aula se o aluno esteve presente (ou não) na aula em questão.<br>2. O sistema salva uma presença para cada um dos alunos que compareceram à(s) aula(s). |

**Requisitos Relacionados:** RF-37
**Classes Relacionadas:** User, Classroom, Class, Presence.

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-8
**Caso de Uso:** Definir *roles* de usuários nas turmas
**Descrição Sucinta:** Este caso de uso permite que um administrador possa conceder a *role* de professor ou de usuário a um determinado aluno.

Tabela 12 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Definir *roles* | O usuário deverá estar logado e ser um administrador do sistema | 1. Em uma lista de todos os usuários inscritos naquela turma, o administrador poderá escolher qual será a *role* associada àquele usuário naquela turma.<br>2. O sistema salva a nova *role* do(s) usuário(s), garantindo novas permissões para aqueles que foram alterados. |

**Requisitos Relacionados:** RF-9, RN-4
**Classes Relacionadas:** User, Classroom, Subscription.

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-9
**Caso de Uso:** Configurar turma
**Descrição Sucinta:** Este caso de uso permite que o professor altere as configurações de uma turma.

Tabela 13 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Configurar turma | O usuário deverá estar logado e ser um professor da turma correspondente | 1. O usuário poderá alterar todas as configurações de turma listadas no RF-7<br>2. Ao finalizar, o sistema salva todas essas alterações. |

**Requisitos Relacionados:** RF-7, RN-10
**Classes Relacionadas:** User, Classroom.

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-10
**Caso de Uso:** Enviar mensagens
**Descrição Sucinta:** Este caso de uso permite que usuários de uma mesma turma troquem mensagens entre si.

Tabela 14 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Enviar mensagens | O usuário deverá estar logado e estar inscrito na turma correspondente | 1. O usuário deverá escolher para quem deseja enviar a mensagem e preencher o campo com o texto desejado.<br>2. O sistema armazena a mensagem enviada e envia um e-mail para o(s) destinatário(s) informando-o(s) sobre a nova mensagem. |

**Requisitos Relacionados:** RF-39, RN-11
**Classes Relacionadas:** User, Message.

## 6.2 Subsistema Board

A Figura 4 apresenta o diagrama de casos de uso do subsistema Board.

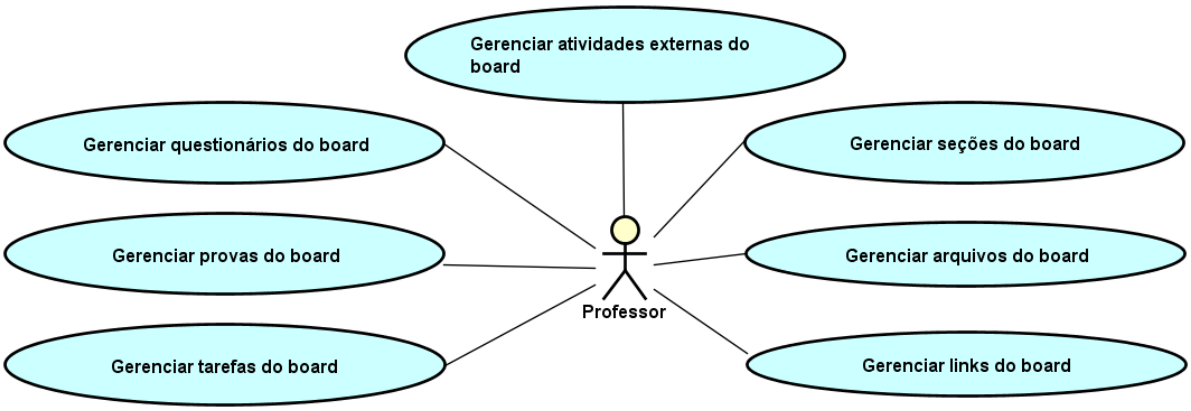


Figura 3 – Diagrama de Casos de Uso do Subsistema Board.

Todos os casos de uso mencionados geram um *Log* indicando o usuário que realizou a ação, qual ação foi tomada e o momento em que isso aconteceu. Além disso, cada *Log* fica associado a uma determinada categoria. Por exemplo, o UC-17 pertenceria à categoria "Gerenciamento de arquivos".

A seguir, são apresentadas as descrições de cada um dos casos de uso identificados. Os casos de uso cadastrais de baixa complexidade, envolvendo inclusão, alteração, consulta e exclusão, são descritos na Tabela 23.

Tabela 15 – Casos de Uso Cadastrais

| Id | Nome | Ações | Observações | Requisit | Classes |
|---|---|---|---|---|---|
| UC-11 | Gerenciar seções do *board* | I | Informar: título e, opcionalmente, descrição. No momento da criação da seção, deve-se escolher a posição do *board* em que esta será inserida. | RF-10 | User, Classroom e Section |
| | | A | | | |
| | | C | | | |
| | | E | A exclusão de uma seção deverá excluir todos os itens de *board* associados a ela. | | |

Tabela 15 – Casos de Uso Cadastrais

| Id | Nome | Ações | Observações | Requisit | Classes |
|---|---|---|---|---|---|
| UC-12 | Gerenciar arquivos do *board* | I | A seção à qual o arquivo estará vinculado será escolhida antes de sua criação. Informar título, descrição e realizar o upload do arquivo desejado. No momento da criação do arquivo, deve-se escolher a posição da seção em que esta será inserido. | RF-11 RN-5 | User, Classroom, Section e File |
| | | A | | | |
| | | C | | | |
| | | E | | | |
| UC-13 | Gerenciar links do *board* | I | A seção à qual o link estará vinculado será escolhida antes de sua criação. Informar título, descrição e o link desejado. No momento da criação do link, deve-se escolher a posição da seção em que esta será inserido. | RF-12, RN-5 | User, Classroom, Section e Link |
| | | A | | | |
| | | C | | | |
| | | E | | | |
| UC-14 | Gerenciar questionários do *board* | I | A seção à qual o questionário estará vinculado será escolhida antes de sua criação. Informar título, descrição, data de início, data de encerramento, as perguntas do questionário e suas respectivas alternativas. No momento da criação do questionário, deve-se escolher a posição da seção em que esta será inserido. | RF-13, RN-5 | User, Classroom, Section, Survey, Survey-Question e SurveyAnswer |
| | | A | | | |
| | | C | | | |
| | | E | A exclusão de um questionário deverá resultar na exclusão de todas as respostas que já foram enviadas. | | |

Tabela 15 – Casos de Uso Cadastrais

| Id | Nome | Ações | Observações | Requisit | Classes |
|---|---|---|---|---|---|
| UC-15 | Gerenciar provas do *board* | I | A seção à qual a prova estará vinculada será escolhida antes de sua criação. Informar título, descrição, data de início, data de encerramento, as questões da prova, seus valores (nota máxima) e seus respectivos tipos (discursiva ou objetiva). No caso de questões objetivas, o professor deverá informar suas alternativas e qual delas é a correta em cada questão. No momento da criação da prova, deve-se escolher a posição da seção em que esta será inserida. | RF-14, RN-5 | User, Classroom, Section, Test, Test-Question e Text Alternative |
| | | A | | | |
| | | C | | | |
| | | E | A exclusão de uma prova deverá resultar na exclusão de todas as resoluções que já foram enviadas. | | |
| UC-16 | Gerenciar tarefas do *board* | I | A seção à qual a tarefa estará vinculada será escolhida antes de sua criação. Informar título, descrição, data de início, data de encerramento e o tipo de tarefa. No caso de tarefas do tipo arquivo, deve-se informar o número máximo de arquivos que poderão ser submetidos. No momento da criação da tarefa, deve-se escolher a posição da seção em que esta será inserida. | RF-15, RN-5 | User, Classroom, Section e Assignment |
| | | A | | | |
| | | C | | | |
| | | E | A exclusão de uma tarefa deverá resultar na exclusão de todas as submissões que já foram feitas e de seus respectivos feedbacks. | | |

Tabela 15 – Casos de Uso Cadastrais

| Id | Nome | Ações | Observações | Requisit | Classes |
|---|---|---|---|---|---|
| UC-17 | Gerenciar atividades externas do *board* | I | A seção à qual a atividade externa estará vinculada será escolhida antes de sua criação. Informar título e descrição. No momento da criação da atividade externa, deve-se escolher a posição da seção em que esta será inserida. | RF-16, RN-5 | User, Classroom, Section, External Activity |
| | | A | | | |
| | | C | | | |
| | | E | A exclusão de uma atividade externa deverá resultar na exclusão de todas as atividades associadas a ela. | | |

## 6.3 Subsistema Board Interactions

A Figura 5 apresenta o diagrama de casos de uso do subsistema Board Interactions.

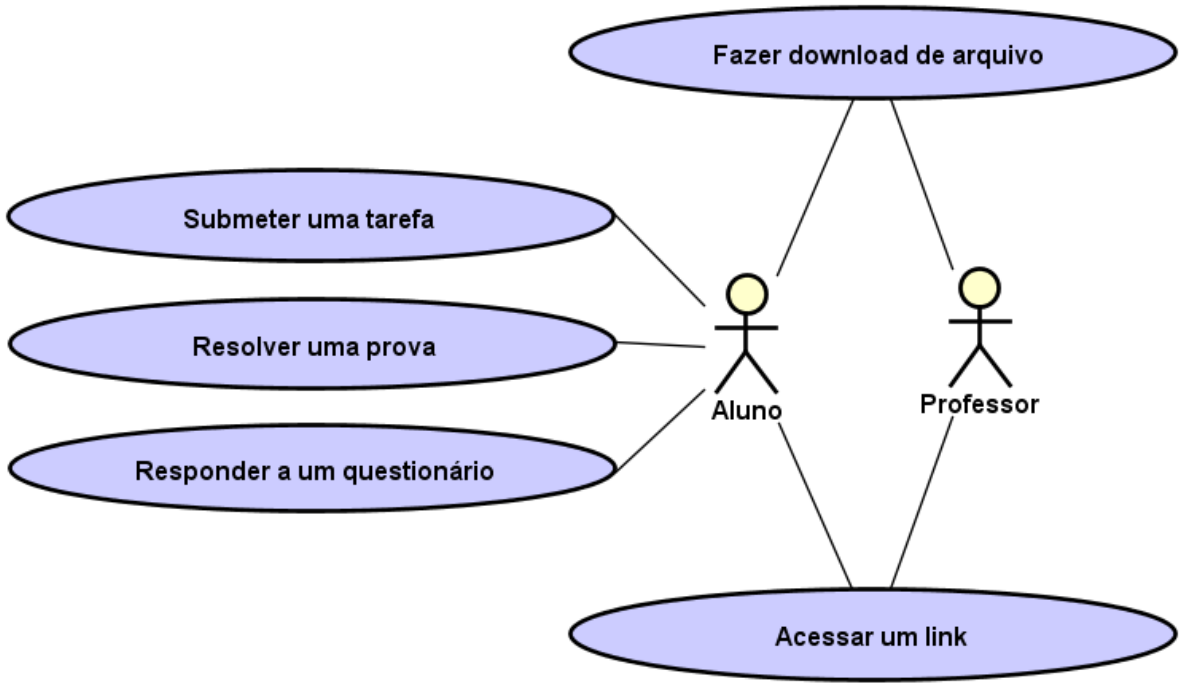


Figura 4 – Diagrama de Casos de Uso do Subsistema Board Interactions.

Todos os casos de uso mencionados geram um *Log* indicando o usuário que realizou a ação, qual ação foi tomada e o momento em que isso aconteceu. Além disso, cada *Log*

fica associado a uma determinada categoria. Por exemplo, o UC-28 pertenceria à categoria "Submissão de tarefa".

Para os casos de uso UC-26, UC-27 e UC-28, a classe correspondente deve ser vinculada ao usuário que interagiu com ela. Por exemplo, no caso de uso UC-28, a classe Submission deve estar associada ao usuário que submeter aquela tarefa.

A seguir, são apresentadas as descrições de cada um dos casos de uso identificados.

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-18
**Caso de Uso:** Fazer download de arquivo
**Descrição Sucinta:** Este caso de uso permite que o usuário realize o download de um arquivo adicionado no *board* da turma.

Tabela 16 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Fazer download | O usuário deverá estar logado e inscrito na turma correspondente | 1. O usuário escolhe o item do *board* de sua turma correspondente ao arquivo.<br>2. O download é iniciado pelo sistema. |

**Requisitos Relacionados:** RF-22
**Classes Relacionadas:** User, Classroom e File.

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-19
**Caso de Uso:** Acessar um link
**Descrição Sucinta:** Este caso de uso permite que o usuário acesse um link adicionado no *board* da turma.

Tabela 17 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Acessar link | O usuário deverá estar logado e inscrito na turma correspondente | 1. O usuário escolhe o item do *board* de sua turma correspondente ao link.<br>2. O navegador acessará a URL quando o usuário escolher o item do *board* correspondente ao link. |

**Requisitos Relacionados:** RF-23
**Classes Relacionadas:** User, Classroom e Link.

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-20
**Caso de Uso:** Responder a um questionário
**Descrição Sucinta:** Este caso de uso permite que um aluno responda a um questionário em sua turma.

Tabela 18 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Responder questionário | O usuário deverá estar logado e inscrito na turma correspondente | 1. O usuário escolhe o item do *board* de sua turma correspondente ao questionário que deseja responder.<br>2. O usuário deverá responder a todas as perguntas obrigatórias e submeter o questionário.<br>3. O sistema armazena todas as respostas para aquele questionário. |
| Visualizar resultado de questionário | O usuário deverá estar logado<br><br>e inscrito na turma correspondente | 1. O usuário escolhe o item do *board* de sua turma correspondente ao questionário.<br><br>2. O usuário poderá visualizar o resultado do questionário. |

Tabela 19 – Fluxos de Eventos Variantes

| Nome do Fluxo | Variante | Descrição |
|---|---|---|
| Data limite atingida | O usuário tenta acessar um questionário em uma data posterior à limite | 1. Passada a data limite, o sistema avisa que o período para responder ao questionário já terminou. |

**Requisitos Relacionados:** RF-24, RN-6
**Classes Relacionadas:** User, Classroom, Survey, SurveyQuestion, SurveyAnswer e SurveyResponse.

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-21
**Caso de Uso:** Resolver uma prova
**Descrição Sucinta:** Este caso de uso permite que o usuário resolva as questões de uma prova, sendo elas discursivas ou objetivas.

Tabela 20 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Resolver prova | O usuário deverá estar logado e inscrito na turma correspondente | 1. O usuário escolhe o item do *board* de sua turma correspondente à prova desejada.<br>2. O usuário deverá responder às perguntas objetivas e discursivas e submeter a prova.<br>3. O sistema armazena as respostas do aluno. |

Tabela 21 – Fluxos de Eventos Variantes

| Nome do Fluxo | Variante | Descrição |
|---|---|---|
| Data limite atingida | O usuário tenta resolver uma prova em uma data posterior à limite | 1. Passada a data limite, o sistema exibe apenas a nota de cada questão e seu feedback. |

**Requisitos Relacionados:** RF-25, RN-6
**Classes Relacionadas:** User, Classroom, Test, TestQuestion, TestAlternative, Test TextResponse e Test AlternativeResponse.

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-22
**Caso de Uso:** Submeter uma tarefa
**Descrição Sucinta:** Este caso de uso permite que o usuário submeta uma tarefa do tipo especificado pelo professor no momento de sua criação.

Tabela 22 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Submeter tarefa | O usuário deverá estar logado e inscrito na turma correspondente | 1. O usuário escolhe o item do *board* de sua turma correspondente à tarefa desejada.<br>2. O usuário deverá submeter a tarefa da seguinte forma: caso a tarefa seja do tipo Texto ou Código, ele deverá preencher o campo com o texto/código correspondente. Caso a tarefa seja do tipo Arquivo, o usuário deverá fazer upload do(s) arquivo(s) correspondente(s). Caso a tarefa seja do tipo Código, o usuário deverá selecionar qual a linguagem de programação foi utilizada.<br>3. O sistema armazena as informações da tarefa enviada. Caso esta seja do tipo Código, cada linha é armazenada separadamente. |

Tabela 23 – Fluxos de Eventos Variantes

| Nome do Fluxo | Variante | Descrição |
|---|---|---|
| Data limite atingida | O usuário tenta submeter uma tarefa em uma data posterior à limite | 1. Passada a data limite, o sistema exibe apenas a nota da tarefa e seu feedback. |
| Número de arquivos superior ao permitido | O usuário tenta submeter uma tarefa do tipo Arquivo fazendo upload de mais arquivos que o permitido | 1. O sistema informa o número máximo de arquivos que o usuário pode submeter. |

**Requisitos Relacionados:** RF-26, RN-8, RN-6
**Classes Relacionadas:** User, Classroom, Assignment, TextSubmission, CodeSubmission e FileSubmission.

## 6.4 Subsistema Feedback

A Figura 6 apresenta o diagrama de casos de uso do subsistema Feedback.

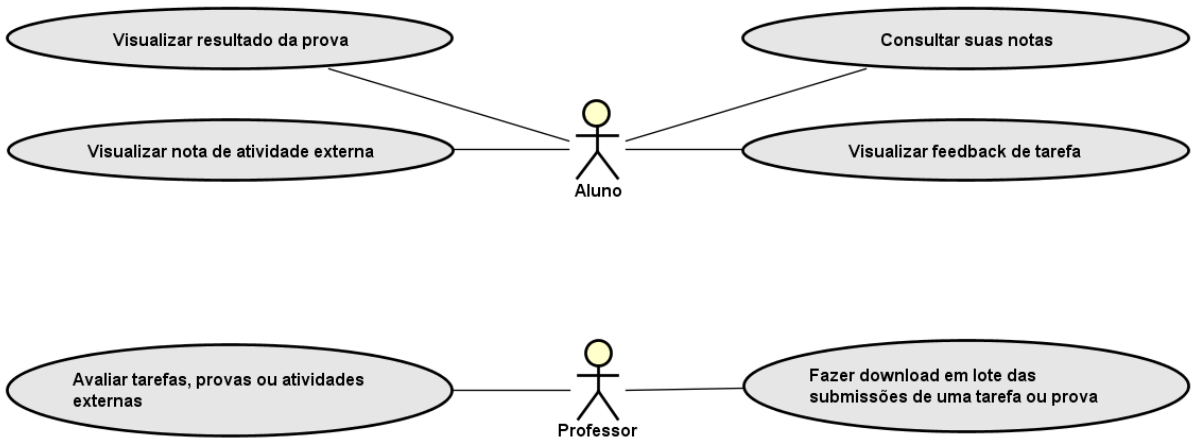


Figura 5 – Diagrama de Casos de Uso do Subsistema Feedback.

Todos os casos de uso mencionados geram um *Log* indicando o usuário que realizou a ação, qual ação foi tomada e o momento em que isso aconteceu. Além disso, cada *Log* fica associado a uma determinada categoria. Por exemplo, o UC-32 pertenceria à categoria "Consulta de notas".

A seguir, são apresentadas as descrições de cada um dos casos de uso identificados. Os casos de uso de consulta mais abrangente que as consultas a um único objeto, mas ainda de baixa complexidade, tais como consultas que combinam informações de vários objetos envolvendo filtros, estão descritos na Tabela 33.

Tabela 24 – Casos de Uso de Consulta

| Id | Nome | Observações | Requisitos | Classes |
|---|---|---|---|---|
| UC-23 | Visualizar resultado da prova | O resultado de uma prova fica disponível para ser visualizado na tela de detalhamento da prova correspondente. Para cada prova, espera-se ver a resposta dada para cada questão, a nota atribuída a cada questão e o feedback dado pelo professor. | RF-33 | User, Classroom, Test, TestQuestion, TestAlternative, Test TextResponse e Test AlternativeResponse |
| UC-24 | Visualizar feedback de tarefa | O resultado de uma tarefa fica disponível para ser visualizado na tela de detalhamento da tarefa correspondente. Para cada tarefa, espera-se ver a submissão feita (seja ela texto, arquivo ou código), a nota atribuída à tarefa e o feedback dado pelo professor. | RF-34 | User, Classroom, Assignment, Submission, Text Submission, File Submission, Code Submission, Code Line, Code Line Feedback e Text Feedback |
| UC-25 | Visualizar nota de atividade externa | A nota de uma atividade externa fica disponível para ser visualizada na tela de detalhamento da atividade correspondente. Para cada atividade externa, espera-se ver a nota atribuída a ela e o feedback dado pelo professor. | RF-35 | User, Classroom, External Activity e Activity |
| UC-26 | Consultar suas notas | Os alunos de uma turma terão acesso ao seu livro de notas. Para cada prova, tarefa e atividade externa realizada ao longo do curso, espera-se ver a nota atribuída a ela. Além disso, deve ser possível também consultar a média parcial no curso. | RF-36 | User, Classroom, Grade Category, Assignment, Submission, Test, Test Question, Test Alternative, Test TextResponse, Test AlternativeResponse, ExternalActivity e Activity |

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-27
**Caso de Uso:** Fazer download em lote das submissões de uma tarefa ou prova
**Descrição Sucinta:** Este caso de uso permite que o professor efetue o download de todas as submissões de uma tarefa ou das respostas de uma prova simultaneamente.

Tabela 25 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Download em lote | O usuário deverá estar logado e ser professor na turma correspondente | 1. O professor escolhe o item do *board* de sua turma correspondente à tarefa/prova desejada.<br>2. Ao clicar sobre o botão correspondente, dar-se-á início ao download de um arquivo compactado contendo todas as provas/tarefas submetidas. |

**Requisitos Relacionados:** RF-31
**Classes Relacionadas:** User, Classroom, Test, TestQuestion, TestAnswer, Assignment, TextSubmission, CodeSubmission e FileSubmission.

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-28
**Caso de Uso:** Avaliar tarefas, provas ou atividades externas
**Descrição Sucinta:** Este caso de uso permite que um professor avalie uma tarefa, prova ou atividade externa de um determinado aluno, sendo possível verificar o que foi submetido.

Tabela 26 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Avaliação individual | O usuário deverá estar logado e ser professor na turma correspondente | 1. O usuário escolhe o item do *board* de sua turma correspondente à tarefa/prova/atividade externa desejada.<br>2. O professor deverá escolher um dos alunos para que possa avaliar. A menos que seja uma atividade externa, ele terá acesso à submissão do aluno: no caso de uma prova, todas as questões e respostas são exibidas. No caso de uma tarefa, o usuário poderá fazer download dos arquivos submetidos (caso a tarefa seja do tipo Arquivo) ou visualizar diretamente na tela o conteúdo enviado (caso seja do tipo Texto ou Código).<br>3. O professor deverá informar a nota correspondente àquela tarefa/prova/atividade e, se desejar, um texto de feedback. No caso de uma tarefa do tipo Código, o professor poderá realizar comentários para cada linha de código. No caso de uma prova, cada questão terá uma nota individual e a soma será feita automaticamente. No caso de uma atividade externa, a avaliação da atividade deverá estar associada ao usuário que realizou a tarefa. |
| Avaliação em lote | O usuário deverá estar logado e ser professor na turma correspondente | 1. O usuário escolhe o item do *board* de sua turma correspondente à tarefa/prova/atividade externa.<br>2. O usuário deverá realizar o download de uma planilha no formato .xls já preenchida com o nome dos alunos e que deverá ser completada com a nota de cada um dos alunos e, opcionalmente, um texto de feedback. No caso de uma prova, a planilha conterá uma coluna para cada questão, sendo possível avaliá-las individualmente. No caso de tarefas de código, o feedback fica restrito a um feedback do tipo texto, não sendo possivel avaliar linhas de código separadamente. No caso de uma atividade externa, a avaliação da atividade deverá estar associada ao usuário que realizou a tarefa.<br>3. O professor deverá efetuar o upload da planilha devidamente preenchida.<br>4. O sistema irá armazenar as notas de todos os alunos para aquela atividade. |

**Requisitos Relacionados:** RF-32, RN-7, RN-9
**Classes Relacionadas:** User, Classroom, Test, TestQuestion, Test TextResponse, Assignment, Submission, TextFeedback, CodeLineFeedback.

## 6.5 Subsistema Discussion

A Figura 7 apresenta o diagrama de casos de uso do subsistema Discussions.

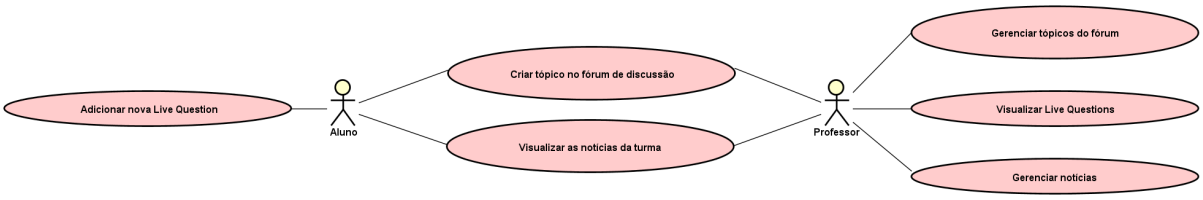


Figura 6 – Diagrama de Casos de Uso do Subsistema Discussion.

Com exceção do caso de uso UC-44, todos os outros casos de uso mencionados geram um *Log* indicando o usuário que realizou a ação, qual ação foi tomada e o momento em que isso aconteceu. Além disso, cada *Log* fica associado a uma determinada categoria. Por exemplo, o UC-41 pertenceria à categoria "Criação de tópico".

No caso das *Live Questions*, o fluxo consiste no fato do aluno **Adicionar nova Live Question** enquanto o professor poderá **Visualizar Live Questions**. Como o intuito é de que essas perguntas sejam feitas em momento de aula, elas serão excluídas passadas 24 horas de sua criação, conforme Figura **??**.

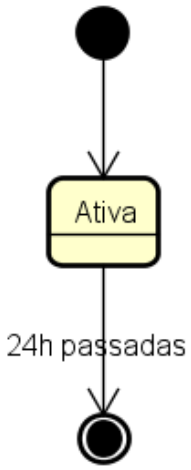


Figura 7 – Diagrama de Estados para uma LiveQuestion.

Para os casos de uso UC-41 e UC-36, a classe correspondente deve ser vinculada ao usuário que interagiu com ela. Por exemplo, no caso de uso UC-41, a classe Forum Topic deve estar associada ao usuário que o criou. No caso do caso de uso UC-43, o usuário só será vinculado à classe caso o usuário não tenha escolhido permanecer no anonimato.

A seguir, são apresentadas as descrições de cada um dos casos de uso identificados. Os casos de uso cadastrais de baixa complexidade, envolvendo inclusão, alteração, consulta e exclusão, são descritos na Tabela 37.

Tabela 27 – Casos de Uso Cadastrais

| Id | Nome | Ações | Observações | Requisit | Classes |
|---|---|---|---|---|---|
| UC-29 | Gerenciar notícias | I | Informar: título e conteúdo. A inclusão de uma nova notícia deve enviar um email para todos os alunos daquela turma contendo seu conteúdo. A notícia criada deve estar associada ao usuário que a criou. | RF-17 | User, Classroom, News |
| | | A | | | |
| | | C | | | |
| | | E | | | |
| UC-30 | Gerenciar tópicos do fórum | I | Descrito em UC-41 | RF-18 | User, Classroom, ForumTopic e ForumReply |
| | | A | | | |
| | | C | | | |
| | | E | A exclusão de um tópico deve acarretar na exclusão de todas as réplicas vinculadas a ele. | | |

Os casos de uso de consulta mais abrangente que as consultas a um único objeto, mas ainda de baixa complexidade, tais como consultas que combinam informações de vários objetos envolvendo filtros, estão descritos na Tabela 38.

Tabela 28 – Casos de Uso de Consulta

| Id | Nome | Observações | Requisitos | Classes |
|---|---|---|---|---|
| UC-31 | Visualizar as notícias da turma | As notícias de uma turma ficam disponíveis para serem visualizadas na lista de notícias no topo do *board*. Para cada notícia, espera-se ver o título, conteúdo, data de publicação, nome do autor e foto do autor. | RF-29 | User, Classroom e News |
| UC-32 | Visualizar *Live Questions* | O professor de uma turma terá acesso a uma lista com todas as *Live Questions* ativas no momento para que possa visualizá-las e responder as dúvidas em sala de aula. | RF-19 | User, Classroom e LiveQuestion |

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-33
**Caso de Uso:** Criar tópico no fórum de discussão
**Descrição Sucinta:** Este caso de uso permite que o usuário crie um novo tópico no fórum de discussões de uma turma.

Tabela 29 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Criar tópico no fórum de discussões | O usuário deverá estar logado e inscrito na turma correspondente | 1. O usuário irá acessar o fórum de discussões da turma.<br>2. O usuário deve inserir o título e o conteúdo do tópico.<br>3. O sistema salva o novo tópico. |
| Responder tópico no fórum de discussões | O usuário deverá estar logado e e inscrito na turma correspondente | 1. O usuário irá acessar o fórum de discussões de sua turma e selecionar o tópico que deseja responder.<br>2. O usuário deverá inserir o conteúdo de sua resposta ao tópico.<br>3. O sistema salva as informações da nova réplica. |
| Visualizar os tópicos do fórum de discussão | O usuário deverá estar logado e e inscrito | 1. O usuário irá acessar o fórum de discussões de sua turma e selecionar o tópico que deseja visualizar.<br>2. O usuário poderá visualizar o tópico e suas respostas. |

**Requisitos Relacionados:** RF-18
**Classes Relacionadas:** User, Classroom e ForumTopic.

**Descrição de Caso de Uso**

**Projeto:** Khoeus - Plataforma de Aprendizado
**Identificador do Caso de Uso:** UC-34
**Caso de Uso:** Adicionar nova Live Question
**Descrição Sucinta:** Este caso de uso permite que o aluno de uma turma crie uma nova *Live Question.*

Tabela 30 – Fluxos de Eventos Normais

| Nome do Fluxo | Precondição | Descrição |
|---|---|---|
| Adiciona nova *LiveQuestion* | O usuário deverá estar logado e inscrito na turma correspondente | 1. O usuário escolhe o item do *board* de sua turma correspondente à lista de *LiveQuestions*<br>2. O usuário deverá informar qual é a sua dúvida e se deseja permanecer no anonimato ou não.<br>3. O sistema armazena as informações da dúvida e o momento de sua criação. Após 24h que a dúvida foi submetida, o sistema deve deletá-la automaticamente. |

**Requisitos Relacionados:** RF-30
**Classes Relacionadas:** User, Classroom, LiveQuestion.

## 6.6 Subsistema Log

A Figura 8 apresenta o diagrama de casos de uso do subsistema Log.

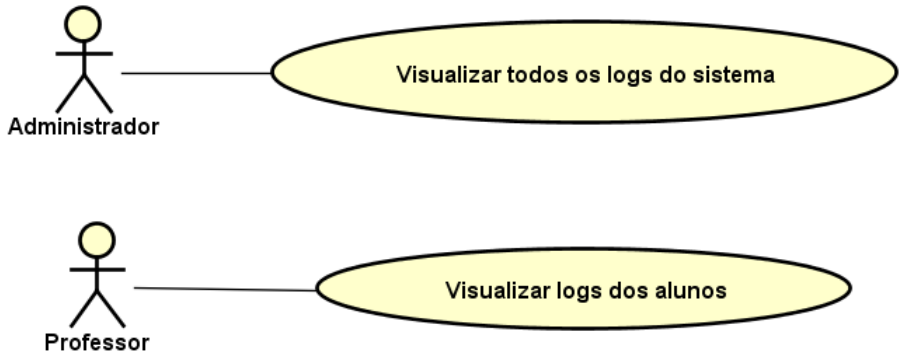


Figura 8 – Diagrama de Casos de Uso do Subsistema Log.

Os casos de uso de consulta mais abrangente que as consultas a um único objeto, mas ainda de baixa complexidade, tais como consultas que combinam informações de vários objetos envolvendo filtros, estão descritos na Tabela 43.

Tabela 31 – Casos de Uso de Consulta

| Id | Nome | Observações | Requisitos | Classes |
|---|---|---|---|---|
| UC-35 | Visualizar logs dos alunos | Dentro de uma turma em que o usuário é professor, ele poderá visualizar os logs de todas as ações realizadas pelos alunos daquela turma, podendo filtrá-los pela data do log, categoria ou pelo usuário que está vinculado a ele. Para cada log, espera-se ver sua data de criação e qual ação foi realizada. | RF-40 | User e Log |
| UC-36 | Visualizar logs do sistema | Administradores do sistema podem visualizar os logs de todas as ações de todos os usuários dentro do sistema, podendo filtrá-los pela data do log, categoria ou pelo usuário que está vinculado a ele. Para cada log, espera-se ver sua data de criação e qual ação foi realizada. | RF-41 | User e Log |

# 7 Modelo Estrutural

O modelo conceitual estrutural visa capturar e descrever as informações (classes, associações e atributos) que o sistema deve representar para prover as funcionalidades descritas na seção anterior. A seguir, são apresentados os diagramas de classes de cada um dos subsistemas identificados no contexto deste projeto. Na Seção 8 – Dicionário de Projeto – são apresentadas as descrições das classes, atributos e operações presentes nos diagramas apresentados nesta seção.

Vale ressaltar que algumas associações são consideradas obrigatórias nos dois sentidos pois considera-se que o tempo decorrido entre a criação de ambas é extremamente pequeno. Além disso, por estarmos considerando **entidades**, uma só faria sentido quando estivesse associado à outra.

## 7.1 Subsistema Classroom

A Figura 10 apresenta o diagrama de classes do subsistema Classroom.

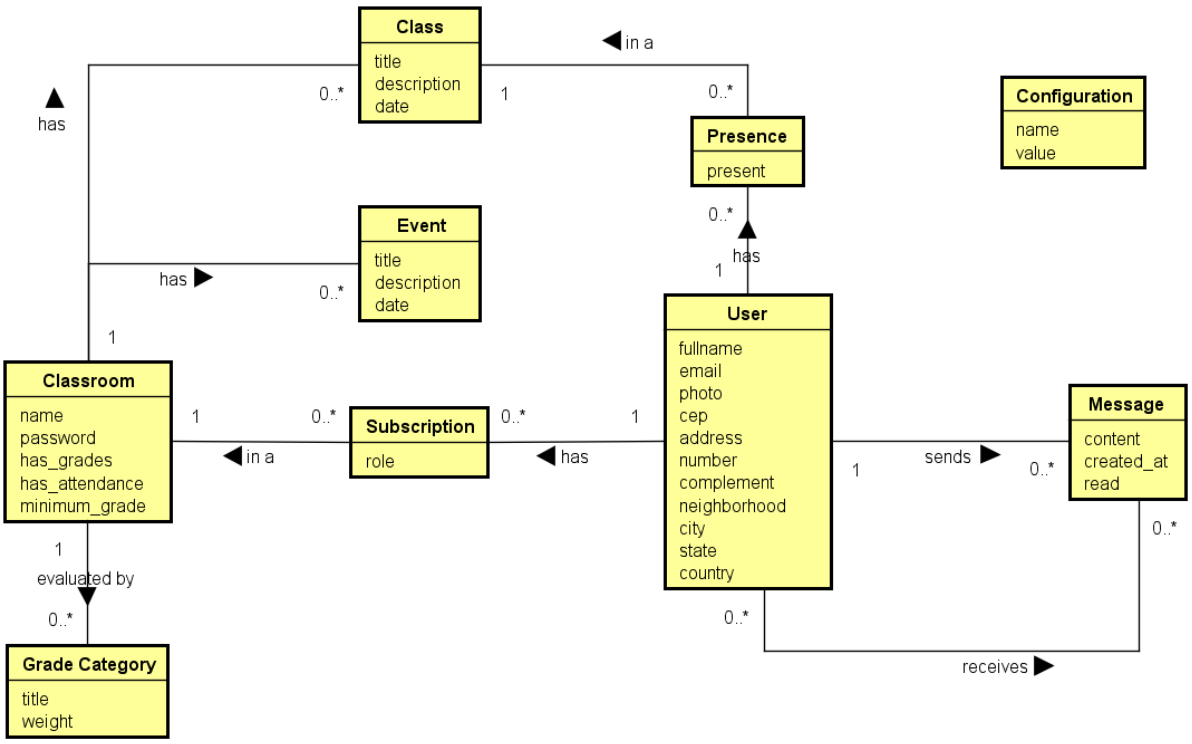


Figura 9 – Diagrama de Classes do subsistema Classroom.

## 7.2 Subsistema Board

A Figura 11 apresenta o diagrama de classes do subsistema Board.

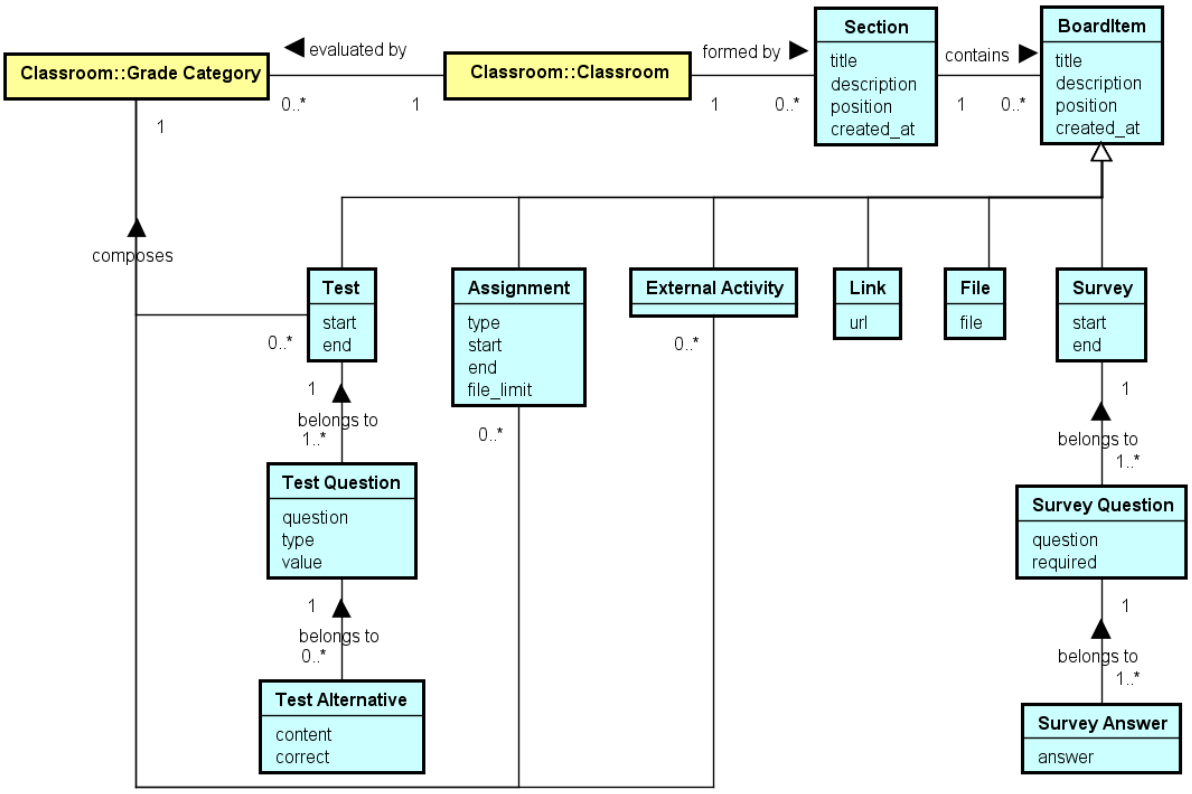


Figura 10 – Diagrama de Classes do Subsistema Board.

## 7.3 Subsistema Board Interactions

A Figura 12 apresenta o diagrama de classes do subsistema Board Interactions.

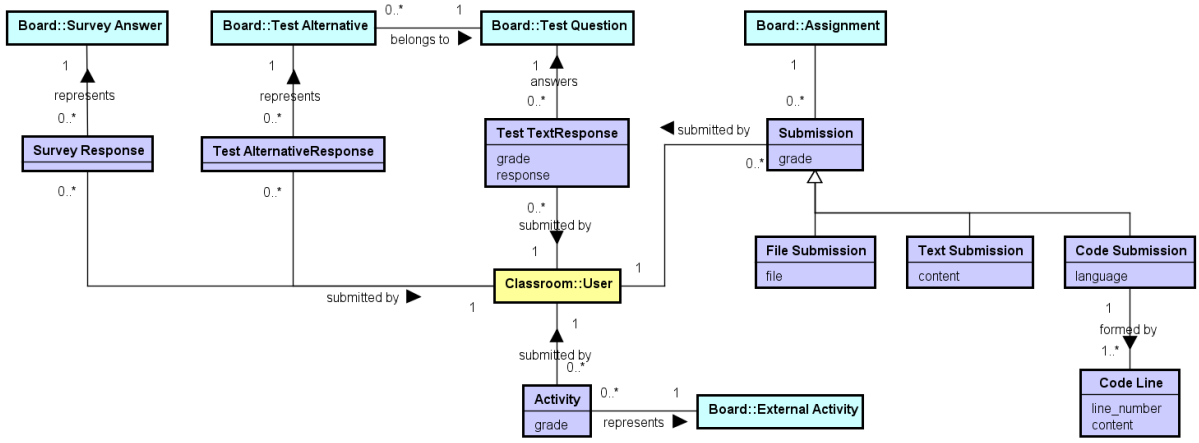


Figura 11 – Diagrama de Classes do Subsistema Board Interactions.

## 7.4 Subsistema Feedback

A Figura 13 apresenta o diagrama de classes do subsistema Feedback.

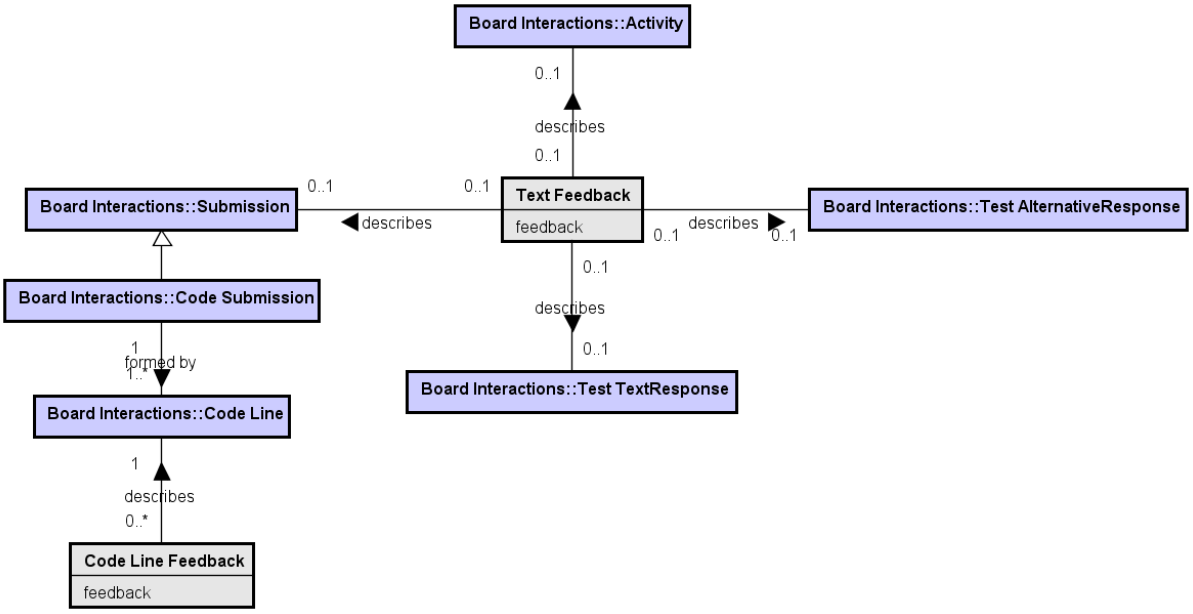


Figura 12 – Diagrama de Classes do Subsistema Feedback.

## 7.5 Subsistema Discussion

A Figura 14 apresenta o diagrama de classes do subsistema Discussion.

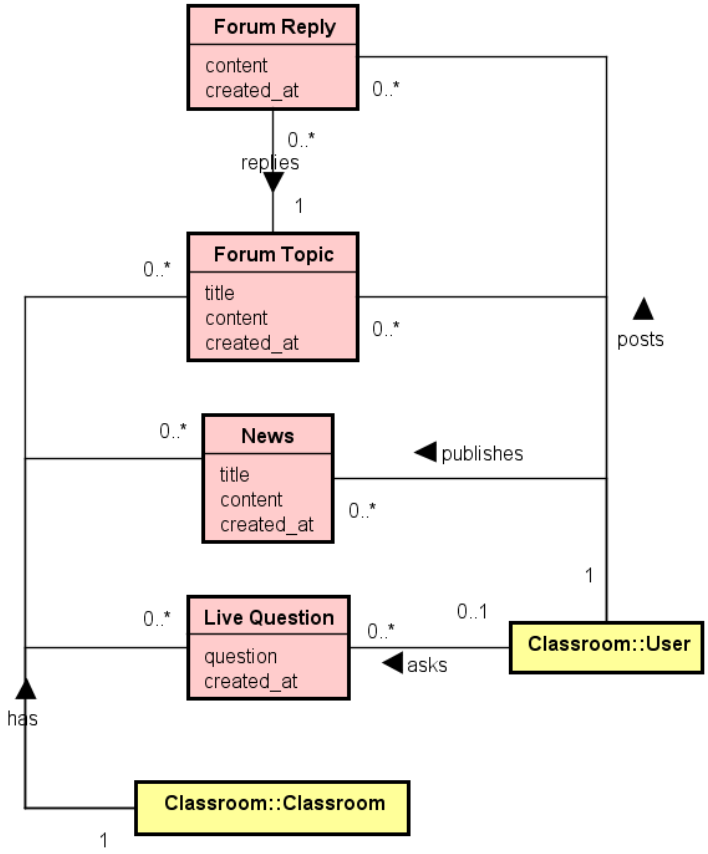


Figura 13 – Diagrama de Classes do Subsistema Discussion.

## 7.6 Subsistema Log

A Figura 15 apresenta o diagrama de classes do subsistema Log.

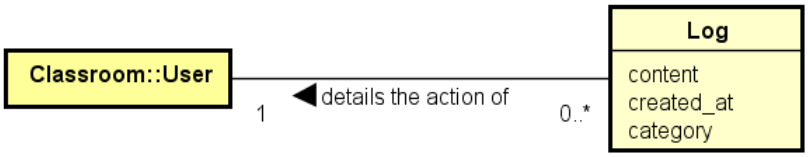


Figura 14 – Diagrama de Classes do Subsistema Log.

# 8 Dicionário de Projeto

Esta seção apresenta as definições das classes (e seus atributos), servindo como um glossário do projeto. As definições são organizadas por subsistema. Vale destacar que eventuais operações que estas classes vierem a ter não são listadas e descritas nesta fase do projeto. Além disse, algumas classes estão envolvidas em mais de um subsistema. Nesses casos, a classe será descrita apenas uma vez.

## 8.1 Subsistema Classroom

### 8.1.1 User

| **Propriedade** | **Tipo** | **Obrigatório?** | **Descrição** |
|---|---|---|---|
| email | Texto | x | Email do usuário utilizado para realizar login. |
| photo | Arquivo | | Arquivo referente à foto do usuário. |
| fullname | Texto | x | Nome completo do usuário. |
| cep | Texto | x | CEP da residência do usuário. |
| address | Texto | x | Logradouro da residência do usuário. |
| number | Número | x | Número da residência do usuário. |
| complement | Texto | | Complemento da residência do usuário. |
| neighborhood | Texto | x | Bairro do usuário. |
| city | Texto | x | Cidade do usuário. |
| state | Texto | x | Estado do usuário. |
| country | Texto | x | País do usuário. |

### 8.1.2 Classroom

| **Propriedade** | **Tipo** | **Obrigatório?** | **Descrição** |
|---|---|---|---|
| name | Texto | x | Nome da turma. |
| password | Texto | | Senha de acesso para a turma. |
| has_grades | Booleano | x | Indica se a turma tem livro de notas. |
| has_attendance | Booleano | x | Indica se a turma tem lista de presença. |
| minimum_grade | Número | x | Nota mínima exigida para a aprovação dos alunos. |
| users | User | | Usuários inscritos na turma. |
| grade_categories | Grade Category | | Categorias de nota associadas à turma para cálculo da média final dos alunos. |
| events | Event | | Eventos adicionados ao calendário da turma |
| classes | Class | | Aulas adicionadas ao calendário da turma |

### 8.1.3 Subscription

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| role | Texto | x | Papel atribuído ao usuário naquela turma (Professor ou Aluno). |
| user | User | x | Usuário que está se inscrevendo na turma. |
| classroom | Classroom | x | Turma em que o usuário está se inscrevendo. |

### 8.1.4 Grade Category

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| title | Texto | x | Nome da categoria. |
| weight | Número | x | Peso da categoria no cálculo da média. |
| classroom | Classroom | x | Turma que utiliza a categoria de nota para cálculo da média. |

### 8.1.5 Message

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| content | Texto | x | Conteúdo da mensagem. |
| created_at | Data | x | Timestamp do momento em que a mensagem foi criado. |
| read | Booleano | x | Indica se a mensagem já foi lida. |
| sent_by | User | x | Usuário que enviou a mensagem. |
| received_by | User | x | Usuários que receberam a mensagem. |

### 8.1.6 Event

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| title | Texto | x | Nome do evento. |
| description | Texto | | Descrição do evento |
| date | Data | x | Timestamp da data em que o evento ocorrerá. |
| classroom | Classroom | x | Turma na qual o evento foi adicionado ao calendário. |

### 8.1.7 Class

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| title | Texto | | Nome associado à aula. |
| description | Texto | | Descrição da aula |
| date | Data | x | Timestamp da data em que a aula ocorrerá. |
| classroom | Classroom | x | Turma na qual a aula foi adicionada ao calendário. |
| students | User | | Alunos que compareceram (ou não) à aula. |

### 8.1.8 Presence

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| present | Booleano | x | Indica se o aluno esteve presente ou não à aula correspondente. |
| class | Class | x | Aula à qual o aluno esteve presente (ou não). |
| user | User | x | Aluno do qual a presença se trata. |

### 8.1.9 Configuration

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| name | Texto | x | Nome da configuração |
| value | Data | x | Valor associado à configuração |

## 8.2 Subsistema Board

### 8.2.1 Section

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| title | Texto | | Título da seção. |
| description | Texto | | Descrição da seção. |
| position | Número | x | Posição da seção no board. |
| created_at | Data | x | Timestamp do momento em que a seção foi criada. |
| classroom | Classroom | x | Turma à qual a seção está vinculada. |
| items | Board Item | | Items do board vinculados à seção. |

### 8.2.2 BoardItem

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| title | Texto | x | Título do item. |
| description | Texto | | Descrição do item. |
| position | Número | x | Posição do item na seção. |
| created_at | Data | x | Timestamp do momento em que o item foi criada. |
| section | Section | x | Seção à qual o item está vinculado. |

### 8.2.3 File

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| file | Arquivo | x | Arquivo a ser baixado. |

### 8.2.4 Link

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| url | Texto | x | URL da página a ser acessada. |

### 8.2.5 Survey

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| start | Data | x | Timestamp do momento em que o questionário deve ser iniciado. |
| end | Data | x | Timestamp do momento em que o questionário deve ser encerrado. |
| questions | Survey Question | x | Questões do questionário |

### 8.2.6 Survey Question

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| question | Texto | x | Pergunta do questionário que deve ser respondida. |
| required | Booleano | x | Indica se a questão é obrigatória ou não. |
| survey | Survey | x | Questionário ao qual a pergunta está vinculada. |
| answers | Survey Answer | x | Alternativas possíveis para a questão. |

### 8.2.7 Survey Answer

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| answer | Texto | x | Resposta para uma pergunta do questionário. |
| question | Survey Question | x | Pergunta do questionário à qual a resposta está vinculada. |
| responses | Survey Response | x | Associação com o usuário que selecionou a opção. |

### 8.2.8 Test

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| start | Data | x | Timestamp do momento em que a prova deve ser iniciada. |
| end | Data | x | Timestamp do momento em que a prova deve ser encerrada. |
| questions | Test Question | x | Questões da prova. |
| category | Grade Category | x | Categoria de nota à qual a prova está associada. |

### 8.2.9 Test Question

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| question | Texto | x | Pergunta da enquete que deve ser respondida. |
| type | Texto | x | Representa se a questão é objetiva ou discursiva. |
| value | Número | x | Nota máxima para a questão. |
| alternatives | Test Alternative | | No caso de questões objetivas, representa as alternativas possíveis para a questão. |
| responses | Test TextResponse | | No caso de questões discursivas, representa a resposta que o aluno deu para a questão. |

### 8.2.10 Test Alternative

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| content | Texto | x | Resposta da alternativa para a questão. |
| correct | Booleano | x | Representa se esta é a alternativa correta para a questão. |
| question | Test Question | x | Representa a questão à qual a alternativa está vinculada. |
| responses | Test AlternativeResponse | | Associação com o usuário que selecionou a alternativa. |

### 8.2.11 Assignment

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| type | Texto | x | Representa se a tarefa é do tipo Texto, Arquivo ou Código |
| start | Data | x | Timestamp do momento em que a tarefa deve ser iniciada. |
| end | Data | x | Timestamp do momento em que a tarefa deve ser encerrada. |
| file_limit | Número | | Número máximo de arquivos que poderão ser enviados para uma tarefa do tipo Arquivo. |
| category | Grade Category | x | Categoria de nota à qual a tarefa está associada. |
| submissions | Submission | | Submissões realizadas naquela tarefa. |

### 8.2.12 External Activity

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| activities | Activity | | Atividades que os alunos realizaram. |
| category | Grade Category | x | Categoria de nota à qual a atividade externa está associada. |

## 8.3 Subsistema Board Interactions

### 8.3.1 Survey Response

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| user | User | x | Usuário que respondeu ao questionário. |
| answer | Survey Answer | x | Alternativa selecionada pelo usuário. |

### 8.3.2 Test AlternativeResponse

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| alternative | Test Alternative | x | Alternativa selecionada pelo usuário. |
| user | User | x | Usuário que respondeu à questão. |
| feedback | Text Feedback | | Feedback concedido à questão no momento da avaliação. |

### 8.3.3 Test TextResponse

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| response | Texto | x | Resposta para a questão. |
| grade | Número | | Nota atribuída à resposta. |
| question | Test Question | x | Questão para a qual a resposta foi dada. |
| user | User | x | Usuário que respondeu à questão. |
| feedback | Text Feedback | | Feedback concedido à questão no momento da avaliação. |

### 8.3.4 Submission

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| grade | Número | | Nota atribuída à submissão. |
| assignment | Assignment | x | Tarefa na qual a submissão foi realizada. |
| user | User | x | Usuário que realizou a submissão da tarefa |
| feedback | Text Feedback | | Feedback dado à tarefa no momento da avaliação. |

### 8.3.5 Text Submission

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| content | Texto | x | Conteúdo (texto) da submissão. |

### 8.3.6 File Submission

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| file | Arquivo | x | Arquivo submetido para a tarefa. |

### 8.3.7 Code Submission

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| language | Texto | x | Linguagem de programação do código submetido |
| lines | Code Line | x | Linhas do código submetido na tarefa. |

### 8.3.8 Code Line

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| line_number | Número | x | Posição da linha no código |
| content | Texto | x | Conteúdo da linha de código |
| submission | Code Submission | x | Código ao qual a linha pertence. |
| feedback | Code Line Feedback | | Feedback dado àquela linha de código no momento da avaliação. |

### 8.3.9 Activity

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| grade | Número | x | Nota atribuída àquela atividade. |
| activity | External Activity | x | Atividade externa que foi realizada. |
| user | User | x | Aluno que realizou a atividade externa. |
| feedback | Text Feedback | | Feedback atribuído à atividade no momento da avaliação. |

## 8.4 Subsistema Feedback

### 8.4.1 Code Line Feedback

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| feedback | Texto | x | Comentário/Feedback dado a uma linha de código específica. |
| line | Code Line | x | Linha de código à qual o feedback foi atribuído. |

### 8.4.2 Text Feedback

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| feedback | Texto | x | Comentário/Feedback dado a uma submissão em formato de texto. |
| alternative | Test Alternative Response | | Questão objetiva de prova à qual o feedback foi atribuído. |
| response | Test TextResponse | | Questão discursiva de prova à qual o feedback foi atribuído. |
| submission | Submission | | Submissão de tarefa à qual o feedback foi atribuído. |
| activity | Activity | | Atividade externa à qual o feedback foi atribuído. |

## 8.5 Subsistema Discussion

### 8.5.1 News

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| title | Texto | x | Título da notícia. |
| content | Texto | x | Conteúdo da notícia |
| created_at | Data | x | Timestamp da data em que a notícia foi criada. |
| author | User | x | Usuário que publicou a notícia. |
| classroom | Classroom | x | Turma na qual a notícia foi publicada. |

### 8.5.2 Forum Topic

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| title | Texto | x | Título do tópico. |
| content | Texto | x | Conteúdo do tópico |
| created_at | Data | x | Timestamp da data em que o tópico foi criado. |
| author | User | x | Usuário que publicou o tópico. |
| classroom | Classroom | x | Turma na qual o tópico foi publicado. |
| replies | Forum Reply | | Réplicas criadas para o tópico. |

### 8.5.3 Forum Reply

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| content | Texto | x | Conteúdo da resposta |
| created_at | Data | x | Timestamp da data em que a resposta foi criada. |
| author | User | x | Usuário que publicou a réplica. |
| classroom | Classroom | x | Turma na qual a réplica foi publicada. |
| topic | Forum Topic | x | Tópico no qual a réplica foi publicada. |

### 8.5.4 Live Question

| Propriedade | Tipo | Obrigatório? | Descrição |
|---|---|---|---|
| question | Texto | x | Dúvida do usuário. |
| created_at | Data | x | Timestamp do momento em que a dúvida foi criada. |
| classroom | Classroom | x | Turma na qual a dúvida foi feita. |
| user | User | | Usuário que criou a dúvida. |

## 8.6 Subsistema Log

### 8.6.1 Log

| **Propriedade** | **Tipo** | **Obrigatório?** | **Descrição** |
|---|---|---|---|
| content | Texto | x | Descrição do log. |
| category | Texto | x | Indica o tipo de ação a que o log se refere. |
| created_at | Data | x | Timestamp do momento em que o log foi criado. |
| user | User | x | Usuário que realizou a ação que deu origem ao log |


Documento de Projeto de Sistema

# Khoeus - Plataforma de Aprendizado

Registro de Alterações:

| Versão | Responsável | Data | Alterações |
|---|---|---|---|
| 1.0 | Luiz Mai | 09/06/2017 | Versão Inicial |
| 1.1 | Vítor E. Silva Souza | 30/06/2017 | Revisão até Seção 3 |
| 1.2 | Luiz Mai | 03/09/2017 | Ajustes da versão 1.1 + Seção 4 |
| 1.3 | Vítor E. Silva Souza | 20/09/2017 | Revisão até Seção 4 |
| 1.4 | Luiz Mai | 25/09/2017 | Ajustes da versão 1.3 |
| 1.5 | Vítor E. Silva Souza | 16/10/2017 | Versão final. |
| 2.0 | Luiz Mai | 16/12/2017 | Ajustes após apresentação. |

Vitória, ES

2017

# 1 Introdução

Este documento apresenta o documento de projeto (*design*) do sistema Khoeus - Plataforma de Aprendizado. Este documento está organizado da seguinte forma: a Seção 2 apresenta a plataforma de software utilizada na implementação da ferramenta; a Seção 3 trata de táticas utilizadas para tratar requisitos não funcionais (atributos de qualidade); por fim, a Seção 4 apresenta o projeto da arquitetura de software e suas subseções explicam cada uma de suas camadas.

# 2 Plataforma de Desenvolvimento

Na Tabela 1 são listadas as tecnologias utilizadas no desenvolvimento da ferramenta, bem como o propósito de sua utilização.

| Tecnologia | Versão | Descrição | Propósito |
|---|---|---|---|
| Ruby | 2.4.1 | Linguagem de programação interpretada, orientada a objetos, de tipagem dinâmica e forte. Projetada tanto para a programação em grande escala quanto para codificação rápida. | Desenvolvimento de aplicação em linguagem de programação orientada a objetos e de codificação rápida. |
| Ruby on Rails | 5.1.1 | Framework web escrito em Ruby sob a licença do MIT. Utiliza uma estrutura model-view-controller (MVC), fornecendo estruturas padrão para banco de dados, serviços e páginas web. | Reduzir a complexidade do desenvolvimento, implantação e gerenciamento da aplicação, de modo que o desenvolvedor não se preocupe demasiadamente com segurança, escalabilidade e desempenho. |
| Bootstrap | 3.3.7 | Framework HTML/CSS/JavaScript para desenvolvimento de sites responsivos e *mobile-first*. | Melhorar a produtividade, permitindo a construção de interfaces para web usando um conjunto de componentes pré-construídos e de caráter responsivo. |
| SASS | 3.4.22 | Extensão CSS que ajuda a reduzir problemas de repetição e manutenção de CSS tradicionais. | Melhorar a produtividade por meio da redução de repetição de código. |
| jQuery | 1.11 | Biblioteca JavaScript que permite a manipulação de documentos HTML, manipulação de eventos, animações e requisições Ajax de forma mais simples. | Melhorar a produtividade por meio da redução de esforço em desenvolvimento de funcionalidades recorrentes na aplicação. |
| PostgreSQL Server | 9.6.1 | Sistema Gerenciador de Banco de Dados Relacional. | Persistência dos dados manipulados pela ferramenta. |
| Git | 2.10 | Sistema de controle de versão distribuído projetado para lidar com velocidade e eficiência. | Assegurar o controle de versão da aplicação. |
| Heroku CI | - | Plataforma em nuvem que permite criar, entregar, monitorar e dimensionar aplicações. | Permitir o *deploy* da aplicação na nuvem. |

Tabela 1 – Plataforma de Desenvolvimento e Tecnologias Utilizadas

Na Tabela 2 vemos os softwares que apoiaram o desenvolvimento de documentos e também do código fonte.

| Tecnologia | Versão | Descrição | Propósito |
|---|---|---|---|
| JetBrains Ruby Mine | 2016.3.1 | Ambiente de desenvolvimento (IDE) para a linguagem Java. | Facilitar a atividade de implementação de software. |
| Astah Professional | 7.0.0 | Ferramenta para modelagem em UML | Modelar os diagramas de classes, casos de uso, etc. |
| TeXstudio | 2.11.2 | Editor de LaTeX. | Escrever a documentação do sistema. |

Tabela 2 – Softwares de Apoio ao Desenvolvimento do Projeto

# 3 Atributos de Qualidade e Táticas

Na Tabela 3 são listados os atributos de qualidade considerados neste projeto, com uma indicação se os mesmos são condutores da arquitetura ou não e as táticas a serem utilizadas para tratá-los.

| Categoria | Requisitos Não Funcionais | Condutor da Arquitetura | Tática |
|---|---|---|---|
| Portabilidade, Redigibilidade | RNF-1, RNF-7 | Sim | 1- O sistema deve ser organizado segundo o padrão MVC (*Model, View, Controller*).<br>2- Utilizar uma linguagem que seja compatível com os principais navegadores do mercado. |
| Facilidade de Aprendizado, Usabilidade | RNF-4, RNF-6 | Não | Prover ao usuário a capacidade de entrar com comandos que permitam operar o sistema de modo mais amigáveis. Para tal, as interfaces do sistema devem permitir, sempre que possível, a entrada por meio de seleção ao invés da digitação de campos. |
| Confiabilidade | RNF-5 | Sim | Sempre que o usuário inserir informações em um formulário, o sistema deve garantir que todos os campos obrigatórios tenham sido devidamente preenchidos, evitando qualquer tipo de inconsistência. |
| Segurança | RNF-2 | Não | O sistema deverá alertar ou redirecionar o usuário quando este tentar realizar uma operação que esteja fora de alcance para a sua *role*. |
| Confidenciabilidade | RNF-3 | Sim | O sistema deverá aplicar uma função de dispersão utilizando o método BCrypt em informações sensíveis a fim de garantir o sigilo dos dados. |

Tabela 3 – Atributos de Qualidade e Táticas Utilizadas

# 4 Arquitetura de Software

A arquitetura de software do sistema Khoeus baseia-se no padrão MVC (*Model, View, Controller*) acoplado a uma Camada de Serviço, sendo comumente chamado de padrão MVCS em uma tentativa de separar a representação de suas informações da interação realizada pelo usuário, conforme proposto em (LEWIS, 2017) e representado na Figura 1.

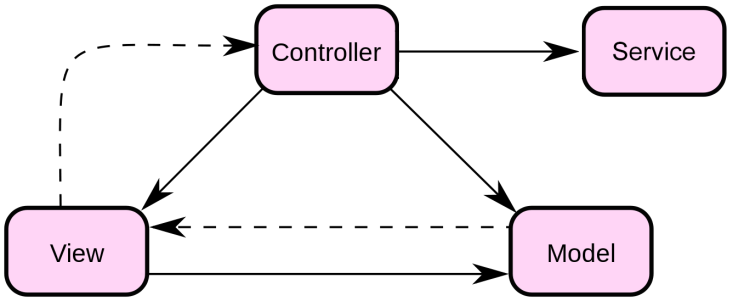


Figura 1 – Representação do padrão de design MVCS.

Nas próximas seções, serão apresentados diagramas no padrão proposto pelo *FrameWeb* (SOUZA, 2007) de forma adaptada ao Ruby on Rails.

## 4.1 Camada *Model*

A seguir, são apresentados os modelos de entidade seguindo o método FrameWeb. Diferentemente da abordagem original proposta em 2007, todos os atributos que não podem ser nulos tiveram a tag `not null` omitida e aqueles que podem tiveram a tag `null` adicionada de forma a reduzir o impacto visual nos diversos diagramas. Pelo mesmo motivo, foi considerada que a estratégia padrão de recuperação de uma associação é do tipo *lazy*, e não *eager* como proposto pelo FrameWeb.

Todas as classes de domínio estendem `ApplicationRecord` que, por sua vez, estende de `ActiveRecord::Base`, uma classe *default* do *Ruby on Rails* responsável pelo mapeamento entre classes e tabelas, por associações, validações, dentre outras funcionalidades. Essa herança não é mostrada nos diagramas com o intuito de não poluí-los com várias associações, mas é possível saber mais sobre o `ActiveRecord` em <https://github.com/rails/rails/tree/master/activerecord>.

A seguir, a Figura 2 representa o modelo de entidades para o subsistema *Auth*, a Figura 3 representa o modelo de entidades para o subsistema *Classroom*, a Figura 4 representa o modelo de entidades para o subsistema *Discussion*, a Figura 5 representa o modelo de entidades para o subsistema *Board*, a Figura 6 representa o modelo de entidades

para o subsistema *Board Interactions*, a Figura 7 representa o modelo de entidades para o subsistema *Feedback* e a Figura 8 representa o modelo de entidades para o subsistema *Logs*.

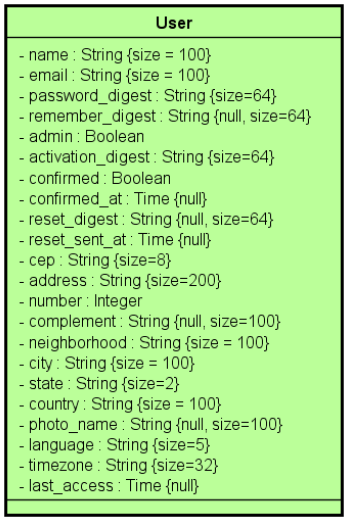


Figura 2 – Modelo de Entidades para o subsistema Auth.

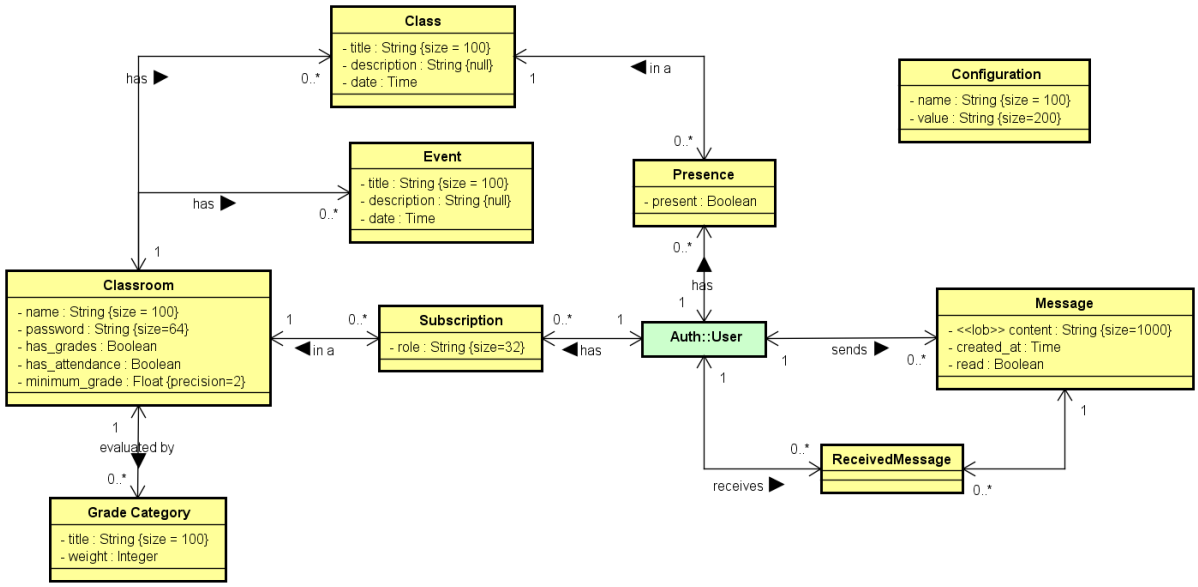


Figura 3 – Modelo de Entidades para o subsistema Classroom.

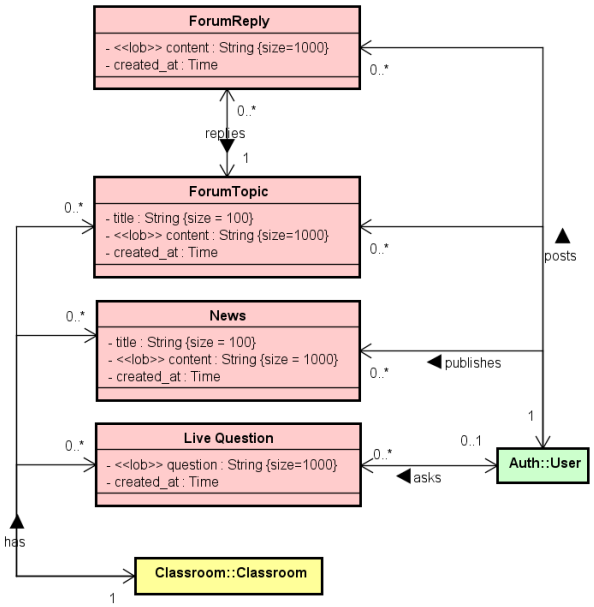


Figura 4 – Modelo de Entidades para o subsistema Discussion.

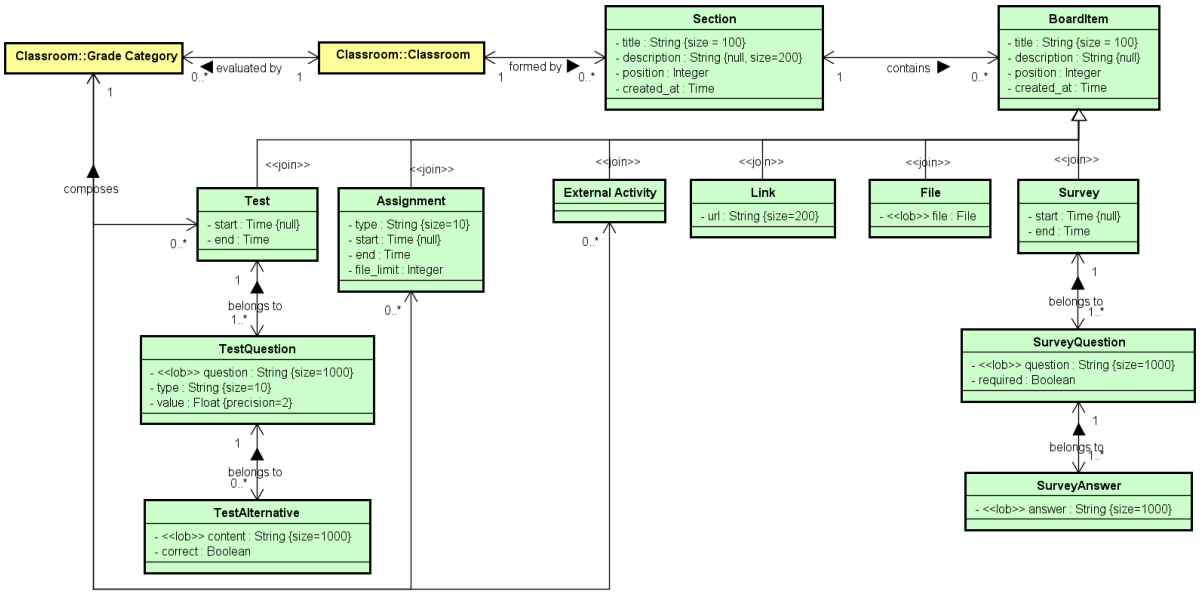


Figura 5 – Modelo de Entidades para o subsistema Board.

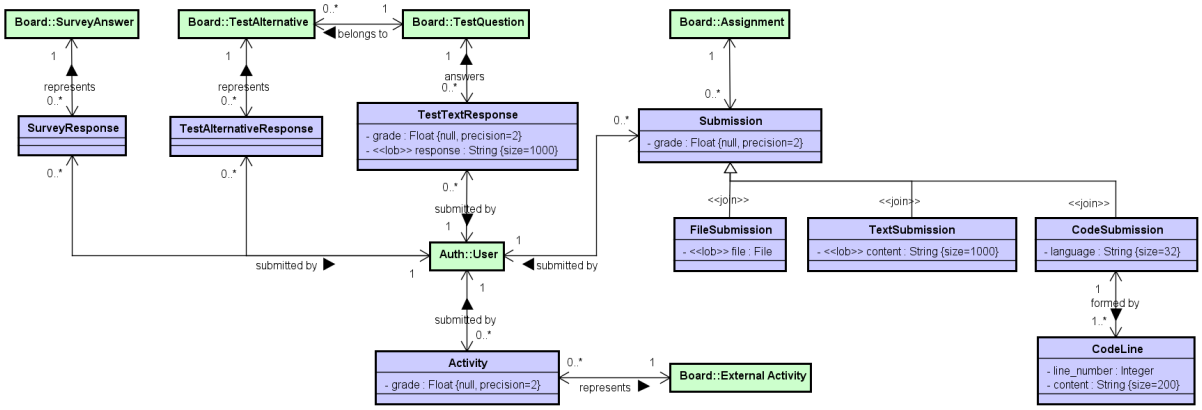


Figura 6 – Modelo de Entidades para o subsistema Board Interactions.

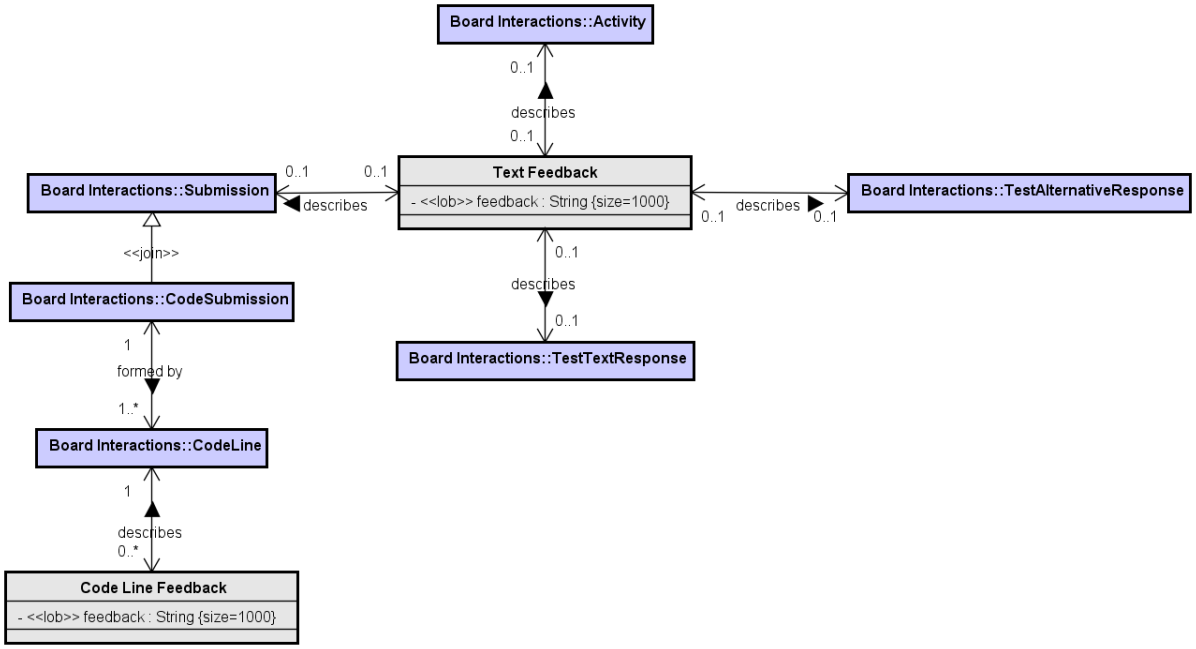


Figura 7 – Modelo de Entidades para o subsistema Feedback.

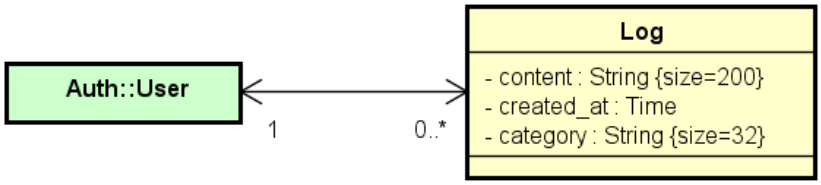


Figura 8 – Modelo de Entidades para o subsistema Logs.

## 4.2 Camada *View* e *Controller*

As funcionalidades criar, visualizar, editar e excluir (abreviadas de CRUD, do inglês *create, read, update and delete*), seguem um mesmo fluxo de execução e de interação

com o usuário. Tais funcionalidades são similares para todos os casos de uso cadastrais, usufruindo do fato de que as classes de domínio todas estendem de `ApplicationRecord`. Esse fluxo de execução similar é representado na Figura 9 por meio de um Modelo de Navegação genérico.

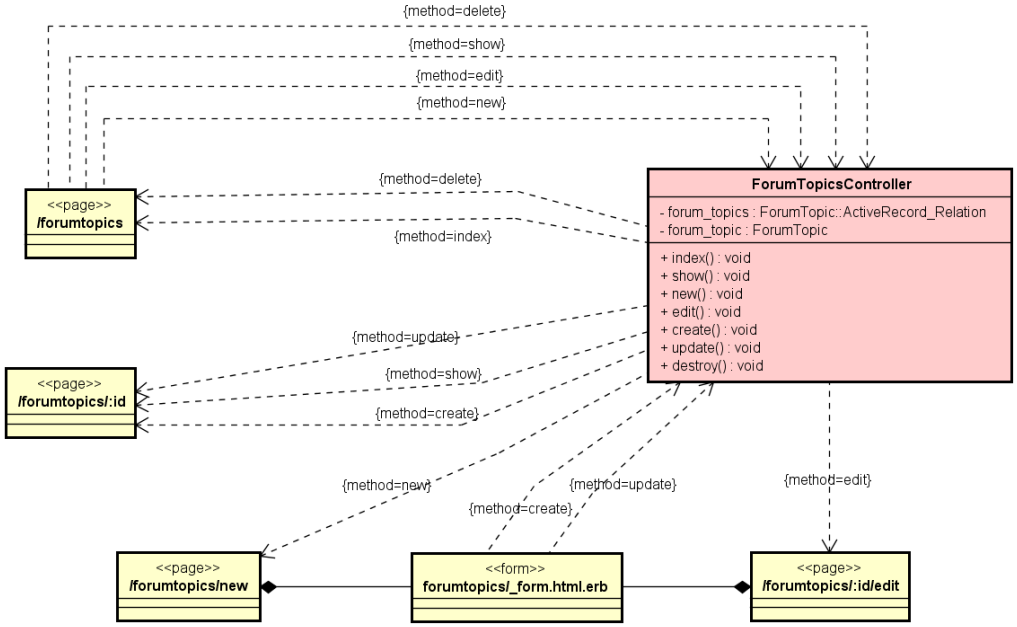


Figura 9 – Modelo de Navegação de um CRUD usando como base os casos de uso cadastrais para tópicos do fórum.

Para os casos de uso que apresentam funções diferentes de apenas as básicas de cadastro, o modelo de navegação mostrado anteriormente não pode ser aplicado. Segue na Figura 10 o modelo de navegação para o caso de uso 'Fazer download em lote das submissões de uma tarefa ou prova', sendo que, neste modelo, foram utilizadas as tarefas como base, e na Figura 11 o modelo de navegação para os casos de uso 'Criar conta', 'Efetuar login' e 'Obter senha esquecida', todos eles relacionados com a autenticação do sistema.

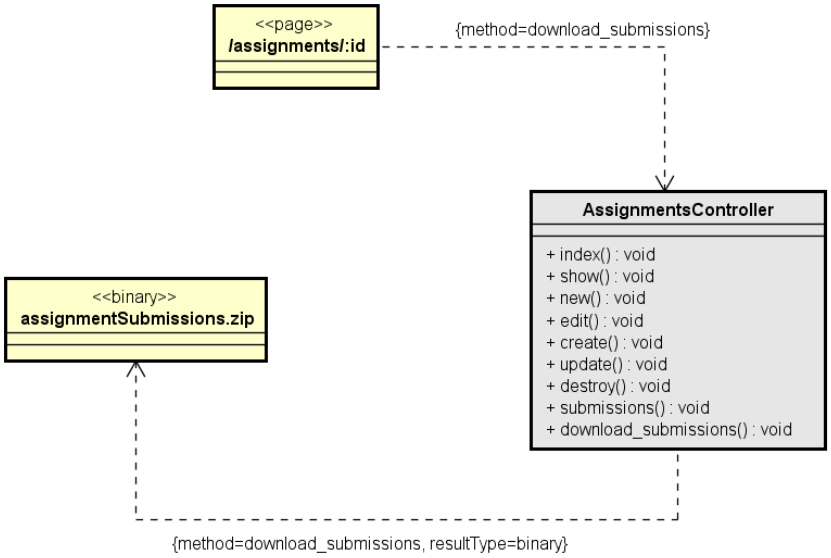


Figura 10 – Modelo de Navegação de 'Fazer download em lote das submissões de uma tarefa'.

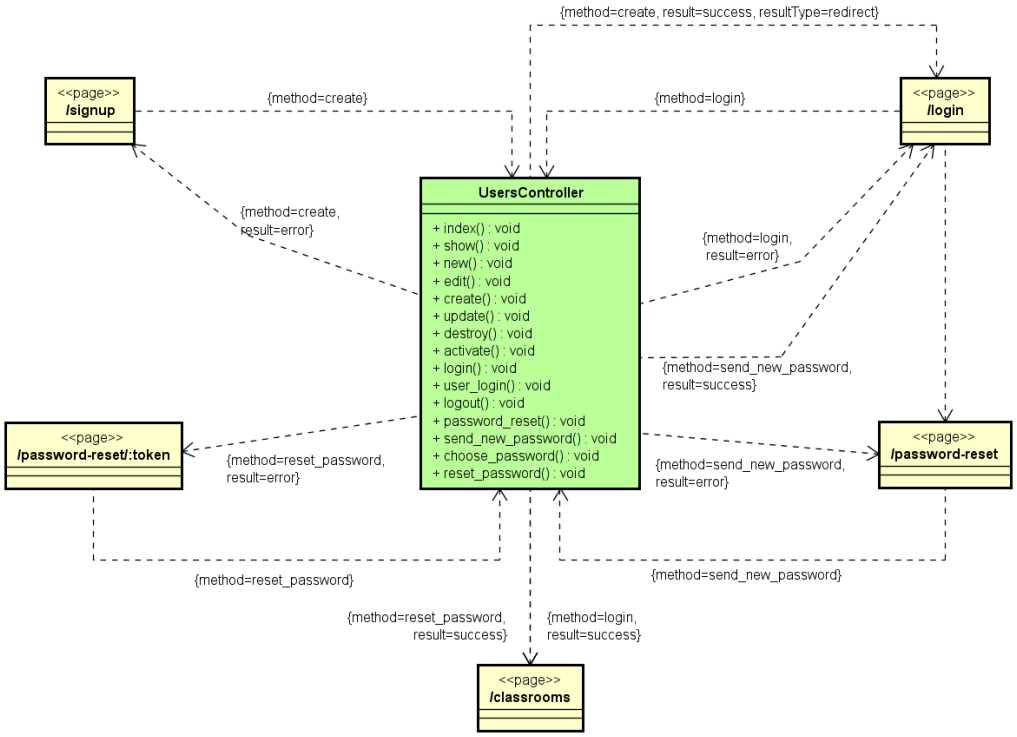


Figura 11 – Modelo de Navegação de 'Criar conta', 'Efetuar login' e 'Obter senha esquecida'.

## 4.3 Camada *Services*

Todas as classes de serviço representadas nos modelos abaixo são **POROs** - *Plain Old Ruby Objects*, ou seja, classes Ruby comuns que implementarão métodos a fim de desacoplar dos *controllers* as funcionalidades que se relacionam com aplicações externas

ou com o banco de dados. Todas as classes de serviço estendem de `CrudService`. Tal classe está representada na Figura 12 de forma genérica e essa herança não é mostrada no diagrama com o intuito de não polui-lo com várias associações

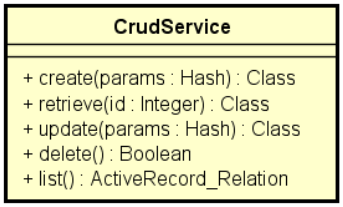


Figura 12 – Representação da classe `CrudService` contendo os métodos de CRUD.

Embora a arquitetura do *Ruby on Rails* não incorpore injeção de dependências, assume-se que existe uma dependência entre o *controller* e a classe de serviço uma vez que, sem ela, o primeiro deixaria de funcionar.

A seguir, a Figura 13 representa o modelo de aplicação para o subsistema *Auth*, a Figura 14 representa o modelo de aplicação para o subsistema *Classroom*, a Figura 15 representa o modelo de aplicação para o subsistema *Discussion*, a Figura 16 representa o modelo de aplicação para o subsistema *Board*, a Figura 17 representa o modelo de aplicação para o subsistema *Board Interactions*, a Figura 18 representa o modelo de aplicação para o subsistema *Feedback* e a Figura 19 representa o modelo de aplicação para o subsistema *Logs*,

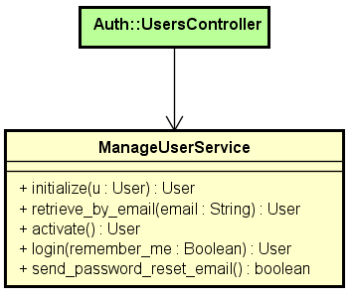


Figura 13 – Modelo de Aplicação para o subsistema Auth.

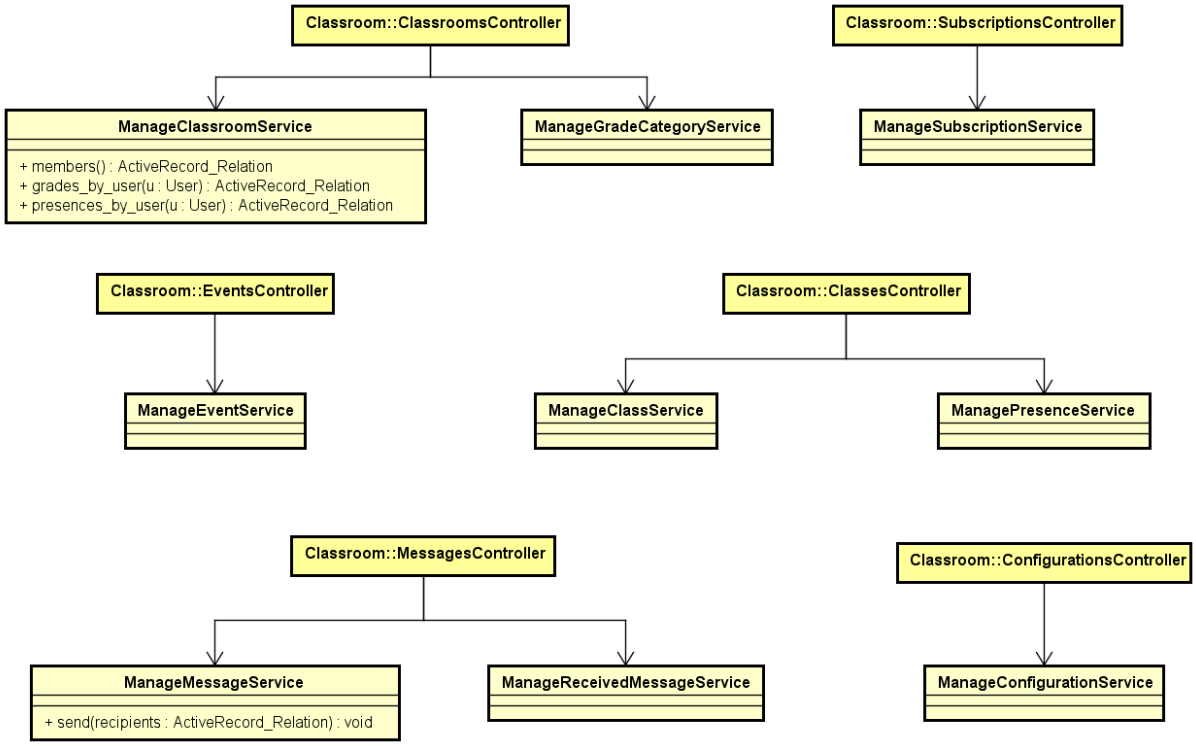


Figura 14 – Modelo de Aplicação para o subsistema Classroom.

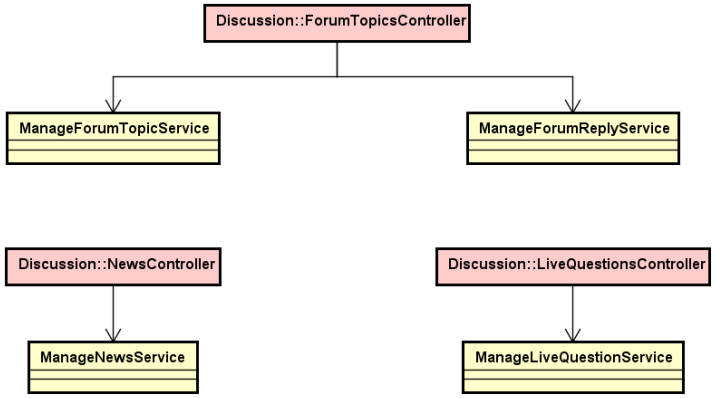


Figura 15 – Modelo de Aplicação para o subsistema Discussion.

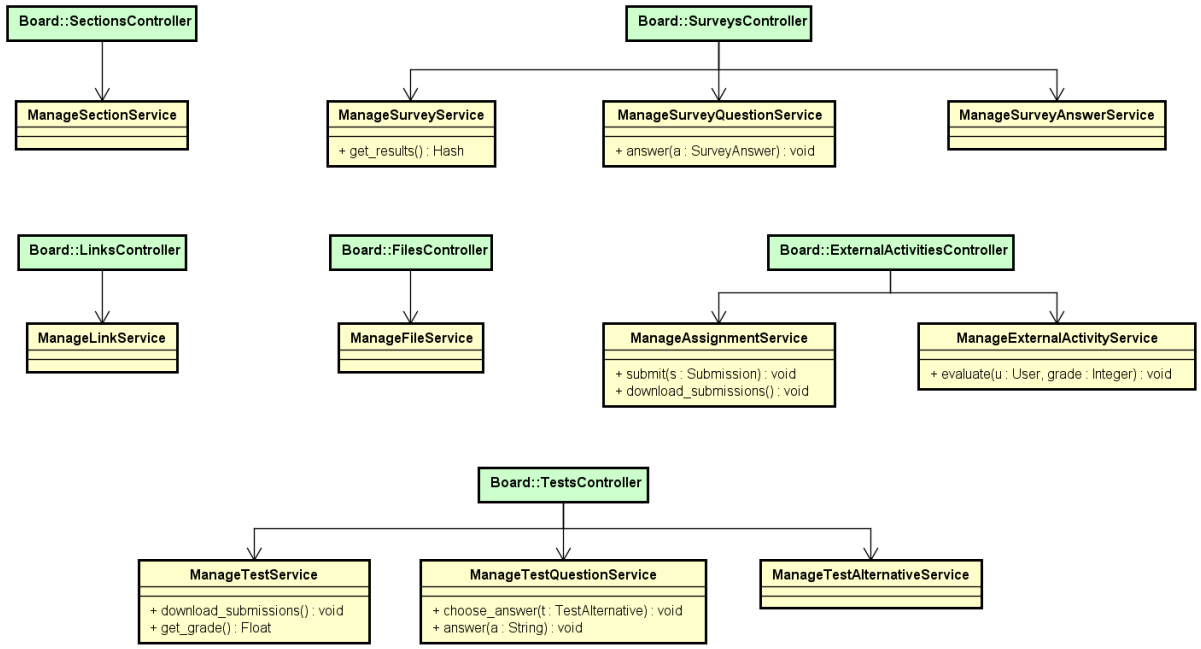


Figura 16 – Modelo de Aplicação para o subsistema Board.

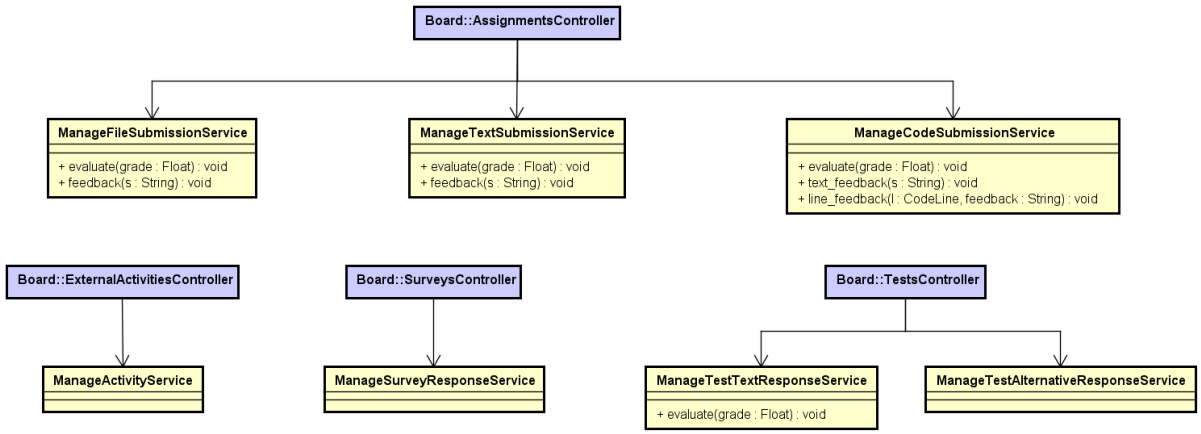


Figura 17 – Modelo de Aplicação para o subsistema Board Interactions.

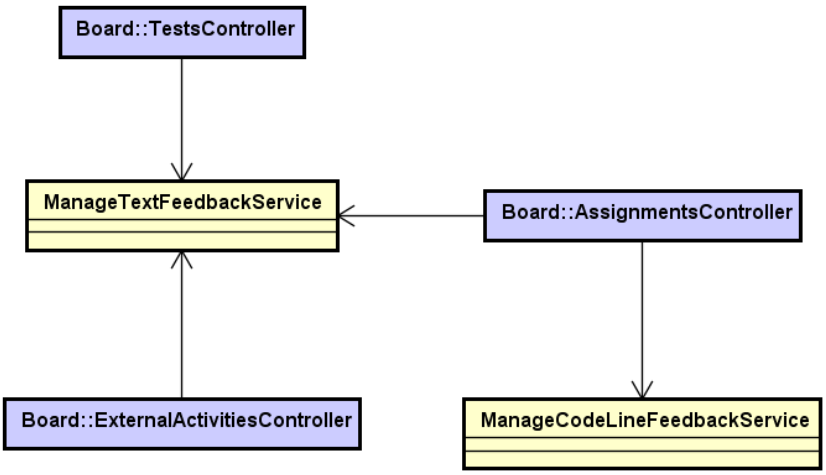


Figura 18 – Modelo de Aplicação para o subsistema Feedback.

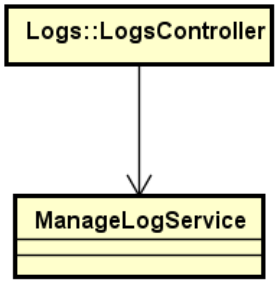


Figura 19 – Modelo de Aplicação para o subsistema Logs.

# Referências

LEWIS, B. *Using Services to Keep Your Rails Controllers Clean and DRY*. [s.n.], 2017. Disponível em: <http://www.engineyard.com/blog/keeping-your-rails-controllers-dry-with-services>. Citado na página 5.

SOUZA, V. E. S. *FrameWeb – A Framework-based Design Method for Web Engineering*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2007. Citado na página 5.