GABRIELA YUMI MUGIUDA MARQUES
GUILHERME PIMENTA SORREGOTTI
LEONARDO AUGUSTO MIZOGUTI

Gabaritei:
Framework para o Desenvolvimento de Ambientes Virtuais de Ensino e Aprendizagem

São Paulo
2015

GABRIELA YUMI MUGIUDA MARQUES
GUILHERME PIMENTA SORREGOTTI
LEONARDO AUGUSTO MIZOGUTI

Gabaritei:
Framework para o Desenvolvimento de Ambientes Virtuais de Ensino e Aprendizagem

Trabalho de Conclusão de Curso de Engenharia de Computação apresentado à Escola Politécnica da Universidade de São Paulo para a Graduação em Engenharia de Computação.

São Paulo
2015

GABRIELA YUMI MUGIUDA MARQUES
GUILHERME PIMENTA SORREGOTTI
LEONARDO AUGUSTO MIZOGUTI

Gabaritei:
Framework para o Desenvolvimento de Ambientes Virtuais de Ensino e Aprendizagem

Trabalho de Conclusão de Curso de Engenharia de Computação apresentado à Escola Politécnica da Universidade de São Paulo para a Graduação em Engenharia de Computação.

Área de concentração:
Engenharia de Computação

Orientador:
Prof.ª Dr.ª Selma Shin Shimizu Melnikoff

São Paulo
2015

Dedicamos o presente trabalho a todos os nossos colegas e professores.

AGRADECIMENTOS

À Professora Selma Shin Shimizu Melnikoff, pela dedicação, conselhos, paciência e por ter nos orientado e auxiliado em todos os momentos deste projeto de formatura.

Agradecemos a participação do nosso colega Eduardo Almeida, que nos auxiliou e apoiou o projeto durante todo seu desenvolvimento.

Agradecemos aos professores e funcionários da Escola Politécnica da USP que fizeram parte dessa nossa jornada.

A todos os nossos colegas que nos apoiaram durante a execução deste projeto de formatura.

Não eduques as crianças nas várias disciplinas recorrendo à força, mas como se fosse um jogo, para que também possas observar melhor qual a disposição natural de cada um.

(Platão)

RESUMO

O presente trabalho tem como objetivo explorar as novas tendências, métodos, processos e técnicas de Engenharia de Software no desenvolvimento de aplicações. O produto final, denominado Gabaritei, é o resultado da aplicação de tais métodos e técnicas em um projeto prático e útil, o qual permite verificar seus pontos fortes e fracos observados durante o seu desenvolvimento.

Gabaritei é um *framework* para a criação de plataformas educacionais virtuais, sendo seu público alvo escolas de ensino fundamental, médio e cursinhos. Com o advento da tecnologia e da informatização nas diversas atividades humanas, nota-se que a educação virtual é ainda um domínio com grande potencial de exploração no Brasil e no mundo, dado que não são muitas as ferramentas que são consideradas referência.

Vale ressaltar por fim que a criação de uma ferramenta adaptável, livre e de fácil uso, como a que se pretende criar com este trabalho, possui também uma função social importante no que diz respeito ao desenvolvimento educacional e tecnológico no país, muitas vezes carente de recursos. O aprimoramento do ensino e da aprendizagem com o auxílio da tecnologia certamente tem o poder de incitar as mentes dos alunos de hoje a construírem um mundo melhor no amanhã.

Palavras-chave: Educação. Ambiente virtual de ensino e aprendizagem. Plataforma educacional. Educação à distância.

ABSTRACT

This project aims at exploring new trends, methods, processes and techniques of software engineering in the development of applications. The final product, named Gabaritei, is the result of the application of such methods and techniques in a practical and useful project, which enables the verification of the pros and cons observed during its development.

Gabaritei is a framework for the creation of virtual learning platforms, and its target market are the elementary schools, high schools and preparatory schools. Considering the advent of technology and information in several human activities, it is noticeable that virtual education is still a domain with a great potential in Brazil and worldwide, as not many tools are considered a reference.

Finally, it is worth mentioning that the creation of an adaptable, easy-to-use and open source application, like the one proposed by this project, has an important social relevance regarding the development of education and technology in Brazil, whose resources are frequently insufficient. The enhancement of teaching and learning aided by technology certainly has the power to kindle the minds of today's students for the construction of a better world in the future.

Keywords: Education. Virtual teaching and learning environment. Learning platform. Distance education.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AJAX	*Asynchronous Javascript and* XML
API	*Application Programming Interface*
CRUD	*Create, Read, Update and Delete*
CSS	*Cascading Style Sheets*
ESPM	Escola Superior de Propaganda e Marketing
FAAP	Fundação Armando Alvares Penteado
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
Insper	Instituto de Ensino e Pesquisa
JS	Javascript
JSON	JavaScript *Object Notation*
LMS	Learning Managament System
MIT	Massachusetts *Institute of Technology*
MVC	Model-View-Controller
PUC	Pontifícia Universidade Católica
REST	*Representational State Transfer*
SASS	*Syntactically Awesome Style Sheets*
SQL	*Structured Query Language*
TDD	*Test-Driven Development*
URL	*Uniform Resource Locator*
XLS	Excel Binary File Format
XLSX	Office *Open* XML *Spreadsheet*
YAML	YAML *Ain't Markup Language*
YARD	Yay! A Ruby *Documentation Tool*

SUMÁRIO

1. INTRODUÇÃO

Este capítulo descreve, de maneira breve, as motivações para o projeto e seu objetivo. Além disso, analisa-se as soluções atuais e, através de suas falhas junto com a opinião colhida de possíveis usuários, descreve-se as funcionalidades básicas esperadas do nosso sistema. Por fim, busca-se criar o ciclo de desenvolvimento de software e a metodologia de trabalho adequados ao objetivo desse projeto de formatura; e que sejam possíveis para o período de tempo limitado deste trabalho.

1.1. Motivação

A principal motivação do trabalho é a construção de um sistema de código livre que seja útil em uma das áreas mais importantes no desenvolvimento do país: educação. Além disso, observa-se que diversos projetos educacionais foram desenvolvidos e nenhum produto pode ser considerado como uma referência no ramo da educação na Internet.

Logo, tem-se como foco liberar um *framework*, uma plataforma de desenvolvimento que seja livre, bem estruturada, eficiente e adaptável tanto para escolas e colégios, quanto para desenvolvedores. Cria-se, dessa forma, uma estrutura de trabalho sólida para que futuros desenvolvedores possam basear-se e criar novas soluções criativas e de grande importância social.

Para atingir tal objetivo, é utilizado o *Test-Driven Development* (TDD), ou seja, Desenvolvimento Orientado a Testes (Beck, 2003). Dessa forma, o *framework* resultante pode ser modificado e adaptado seguindo um processo que permita controlar sistematicamente a qualidade dos produtos intermediários. Além disso, seguindo-se as recomendações do TDD, novas funções podem ser adicionadas sem prejudicar a estrutura sólida criada inicialmente por meio da aplicação da Engenharia de Software. Essa garantia é de fundamental importância para o próximo desenvolvedor a reutilizar o *framework* Gabaritei.

O potencial de aprendizado para os alunos participantes do projeto é abundante, permitindo a evolução não apenas em técnicas de desenvolvimento de software como o Scrum, bem como na metodologia do TDD. Esses conceitos são imprescindíveis para um futuro profissional na área de Engenharia de Software.

Além disso, por ser um projeto público e de código livre, a documentação desenvolvida deve ser clara, concisa e de fácil entendimento mesmo para desenvolvedores que nunca tiveram contato com o projeto. Ainda mais, o próprio código do projeto deve ser desenvolvido utilizando boas práticas de programação, além de organizado, bem comentado e estruturado.

1.2. Objetivo

O *framework* Gabaritei tem como objetivo, como mencionado anteriormente, uma plataforma aberta, adaptável e flexível, para ser usada como um padrão para a criação de salas de aula virtuais. Em outras palavras, o Gabaritei proporciona uma estrutura que permite ao professor disponibilizar conteúdo *online* aos alunos, desde materiais a exercícios recomendados. Não tem como objetivo de ser um produto comercial.

Em resumo, escolas e cursinhos terão a possibilidade de utilizar o *framework* para a criação de ambientes virtuais para seus alunos, o que auxiliaria na eficiência da educação no país no âmbito do ensino fundamental e médio.

1.3. Justificativa

O cursinho popular da Faculdade de Economia e Administração da Universidade de São Paulo (FEA-USP), em um exemplo mais específico, registrou que muitos dos seus alunos desistem do curso por não conseguir ir às aulas. Sendo assim, uma ferramenta, que crie um canal de comunicação educacional especifico entre esses alunos, professores e coordenadores, é importante para permitir que eles continuem o curso, mesmo que não possam ir a todas as aulas por dificuldades econômicas ou logísticas.

Observa-se que o Gabaritei, sendo um projeto de código livre, atrai, também, os desenvolvedores de software por disponibilizar uma ferramenta poderosa e bem projetada como base para a realização de projetos educacionais. Mais do que poderosa, ela é flexível, pelo apoio do desenvolvimento orientado a testes, o que irá permitir o surgimento de novos produtos e soluções na área de educação.

Além disso, as plataformas educacionais disponíveis no mercado focam em universidades, e, portanto, foram projetadas para um cenário com disciplinas independentes entre si. O Gabaritei, tendo o foco em escolas e cursinhos, permite a criação de salas com várias disciplinas agremiadas, e consequentemente, possibilita também a criação de exames multidisciplinares.

Outro fato observado, é que essas plataformas apresentam interfaces antiquadas e poluídas, de forma que chegam a confundir e atrapalhar o usuário. O Gabaritei, com uma interface mais *flat* e simples, tem como objetivo ser muito intuitivo e fácil de usar.

Inclusive, o projeto já tem dois potenciais interessados. Além do cursinho popular da FEA-USP, onde a ideia nasceu, o Núcleo de Ensino e Aprendizado de São Paulo também busca uma ferramenta semelhante ao *framework* Gabaritei.

1.4. Metodologia de trabalho

A metodologia de trabalho, utilizada neste trabalho para o desenvolvimento do *framework* Gabaritei, pode ser dividida nas seguintes etapas: estudo do estado da arte, análise de requisitos e escopo do sistema, escolha das tecnologias, definição da arquitetura, testes e desenvolvimento do sistema; apresentadas em detalhes a seguir.

1.4.1. Estudo sobre o estado da arte

Após a escolha do objetivo do tema, estudaram-se as plataformas de educação que já existiam, tanto no âmbito nacional, quanto no âmbito internacional. Chegou-se a

duas ferramentas que se destacam em ambos os cenários: Moodle e Blackboard (edutechnica, 2014) e em uma terceira, por ser base da plataforma utilizada pelo Departamento de Engenharia de Computação e Sistemas Digitais: o Sakai. As três referências destacam-se por sua alta adesão em um ambiente de ensino superior, ou seja, universidades. Dessa forma, nota-se que seus recursos são direcionados para o cenário de disciplinas mais independentes. O Gabaritei foca sua atividade em ensino fundamental e cursinhos. Dessa forma, as disciplinas não são independentes e sim agremiadas em uma sala (ou, em inglês *course*).

Outro ponto comum em todas as ferramentas é o *design* que, em geral atrapalha e é antiquado. Em sua maioria, as ferramentas utilizam como interface uma metáfora de mesa de trabalho, semelhante às aplicações desenvolvidas para Windows. O ponto fraco dessa abordagem é que aplicações *web* se caracterizam e são utilizadas de uma maneira diferente. Elas são mais dinâmicas, devem se adaptar em diversos dispositivos e essas ferramentas não provêm uma interface adequada a esses requisitos. Dessa forma, desenvolveu-se um sistema focado em uma interface moderna, utilizando recursos que são estado da arte, como arrastar e soltar, autocompletar, *typeahead*, mecanismos de busca, paginação, organização de conteúdo, entre muitos outros recursos que foram desenvolvidos focando em usabilidade.

Outra característica em comum notada é que as ferramentas avaliadas tentam implementar muitos recursos já existentes. Por exemplo, calendários próprios, sistemas de *chat*, mensagens e vídeo chamadas. O enfoque do Gabaritei também diverge deste ponto, uma vez que foram focados e aprimorados apenas os pontos principais da sua finalidade. Para complementá-lo, sistemas auxiliares podem ser utilizados, por exemplo, através de maior integração com o Google Docs ou Google Calendar.

1.4.2. Análise de requisitos e definição do escopo do sistema

Depois do estudo sobre os trabalhos relacionados, juntamente com os potenciais clientes, foi criado o *Product Backlog*, anexado no Apêndice A, que contém os

requisitos do sistema. Baseando-se nele, foram criados os casos de uso e, posteriormente, os diagramas de classe e o modelo de dados.

1.4.3. Estudo e escolha das tecnologias

A seguir, foi realizado um estudo sobre as atuais ferramentas e tecnologias utilizadas no desenvolvimento de *software*: e optou-se por utilizar o *framework* Ruby on Rails para a implementação do servidor e o *framework* AngularJS para a lógica do cliente.

A escolha do Ruby on Rails baseou-se na tendência de crescimento do uso e da adoção da tecnologia, com uma comunidade altamente ativa, e existência de uma vasta variedade de recursos e extensões *open source* disponíveis para desenvolvimento. Além disso, a ferramenta é de fácil uso e configuração, o que permite uma concentração maior no desenvolvimento. Outra vantagem desse *framework* é a abstração da camada de acesso aos dados, o que permite a livre escolha da tecnologia de banco de dados. Dessa forma, os potenciais clientes poderão escolher a tecnologia que melhor lhe convier, sendo o servidor facilmente configurável.

Para a implementação da parte do cliente, a escolha da adoção do AngularJS baseou-se, na mesma forma que para a escolha do Ruby on Rails, na crescente popularidade da ferramenta. Oferece a integração com uma infinidade de componentes *open source* que aceleram de forma significativa o desenvolvimento, sem contar a simplicidade e facilidade de uso e manutenção, uma vez que a ferramenta exige a estruturação dos *scripts* de execução das páginas HTML.

À medida que o projeto foi avançando, foi necessário pesquisar outras tecnologias que respondessem às necessidades do sistema. Por meio de pesquisas em sites e fóruns sobre as últimas tecnologias web, foi possível identificar outras tecnologias para acelerar o desenvolvimento e resolver problemas.

Mais detalhes e informações sobre essas e outras tecnologias escolhidas são fornecidas na seção 3.2.

1.4.4. Definição da arquitetura

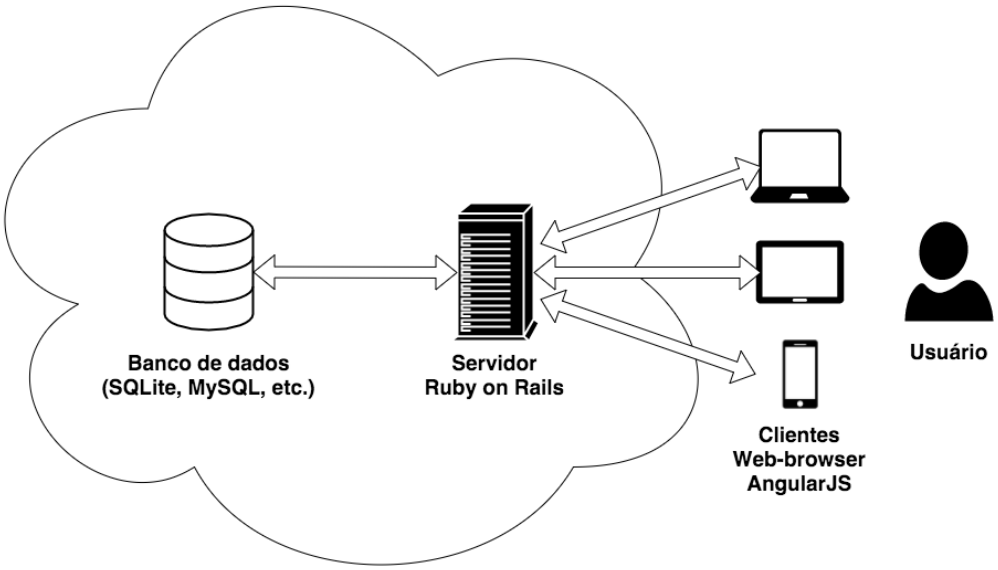
Dado o intuito do projeto de formatura para se construir um *framework* para ambientes de ensino e aprendizado na *web*, a arquitetura do sistema foi definida como um servidor que responde às requisições dos clientes (página *web*) realizadas de forma síncrona e/ou assíncrona por meio do protocolo HTTP.

Tanto o servidor como o cliente aplica a arquitetura MVC internamente [referência]. Com a adoção do AngularJS, as responsabilidades de cada um deles foram alteradas em relação ao padrão de modelos que não utilizam o AngularJS. O cliente passa a implementar parte da lógica de negócio em seu processamento, cabendo ao servidor fornecer dados estruturados e serviços do tipo REST, processar tarefas em segundo plano (como enviar e-mails e importar dados) e manipular o banco de dados. A comunicação e a integração são detalhadas na seção 4.2.

Com o uso de outras ferramentas (como o Bootstrap, ver item 3.2.3), foi possível realizar um único desenvolvimento para todas as plataformas com acesso à *web* (computadores pessoais, *smartphones* e *tablets*).

A Figura 1 ilustra de forma simplificada a arquitetura adotada.

Figura 1 – Arquitetura simplificada do sistema

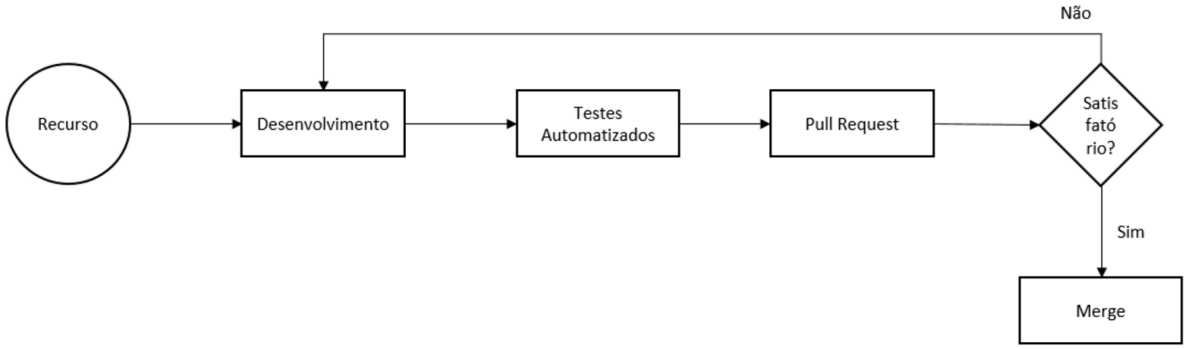


1.4.5. Testes e desenvolvimento do sistema

O repositório central do sistema foi criado no sistema gratuito GitHub. Trabalhou-se em um sistema de *Fork* para impedir que acidentes de desenvolvimento ocorressem, por exemplo, caso um desenvolvedor do sistema fizesse uma alteração que impedisse de funcionar o sistema no repositório padrão. Dessa forma, no repositório principal tem-se sempre uma versão funcional do sistema Gabaritei.

Para garantir o correto desenvolvimento, bem como sua consistência, adotou-se o ciclo de desenvolvimento descrito no diagrama da Figura 2:

Figura 2 – Ciclo de desenvolvimento



Dado um recurso a ser desenvolvido no sistema, o desenvolvedor planeja e cria a solução (desenvolvimento), escreve testes automatizados e monta um *Pull Request*, que é um *code review* e testes da alteração que ele irá provocar no sistema. Quando todos os desenvolvedores do sistema aceitarem a modificação, é feito o *merge* com o repositório principal (chamado de *master*).

Com os testes automatizados (conforme será descrito nas tecnologias da próxima sessão do documento), garantiu-se que alterações não gerem impactos nas partes que estão funcionando do sistema.

1.5. Estrutura do documento

Os demais capítulos do trabalho estão estruturados do seguinte modo:

- O **Capítulo 2** contém os trabalhos relacionados e contextualiza o que já foi feito na área. Para isso, são apresentadas as plataformas mais utilizadas atualmente na área da educação.

- O **Capítulo 3** apresenta o contexto e explica os conceitos e as tecnologias utilizados nesse trabalho.

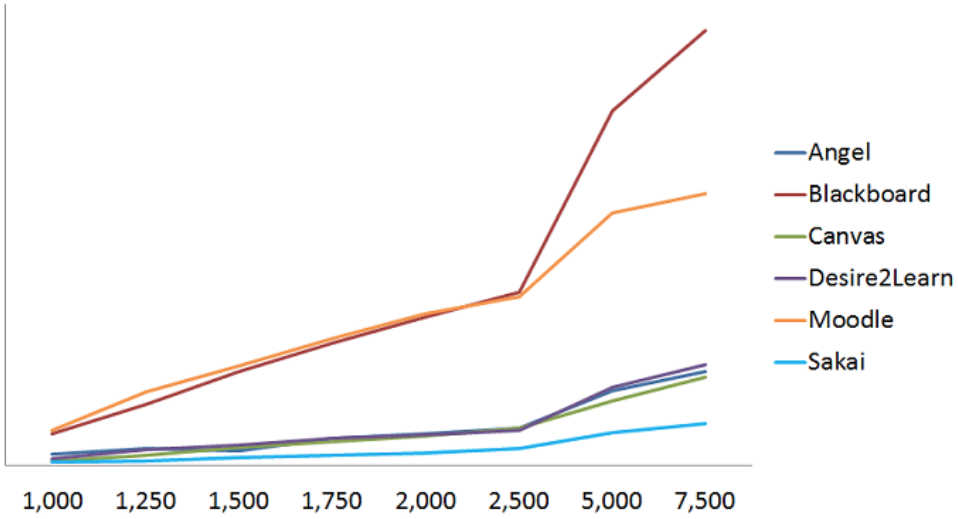
- No **Capítulo 4** é descrito como foi realizado o desenvolvimento do *framework* Gabaritei, desde da especificação de requisitos até a obtenção do produto final e a sua respectiva avaliação.

- Finalmente, o **Capítulo 5** contém as considerações finais, ou seja, as conclusões sobre o resultado final, as contribuições desse trabalho e os trabalhos futuros.

2. ESTADO DA ARTE

Foi realizado um estudo sobre trabalhos relacionados para que fosse possível uma compreensão mais profunda do que já foi realizado na área. Como para escolas e cursinhos quase não existem plataformas especializadas, também foram pesquisadas Sakai, Moodle e o Blackboard. O Moodle e a Blackboard são as mais usadas segundo uma pesquisa da edutechnica (edutechnica, 2014), como mostra a Figura 3. O Sakai foi escolhido por ser a base da plataforma educational utilizada pelo Departamento de Engenharia de Computação e Sistemas Digitais, o Tidia-Ae[1].

Figura 3 – Número de instituições usando cada LMS



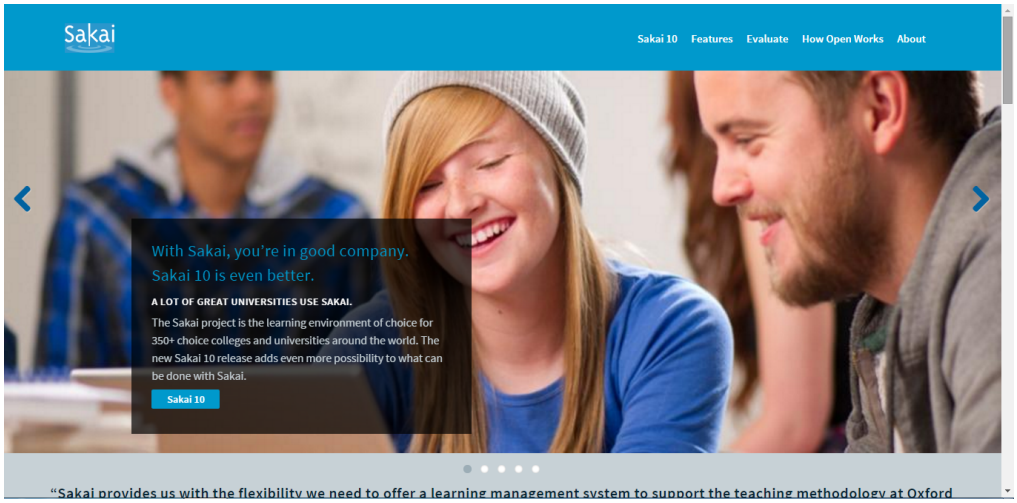
Fonte: (edutechnica, 2014)

2.1. Sakai

O Sakai, como ilustrado pela Figura 4, é um projeto de código aberto que atualmente encontra-se na sua versão dez. Construído por um conjunto de grandes universidades dos Estados Unidos (como Standford e MIT) em 2004, de acordo com o site oficial, é usado por mais de 350 instituição de ensino em todo o mundo (Sakai.org, 2015).

[1] Página do Tidia-Ae - http://tidia-ae.usp.br/portal

Figura 4 – Página inicial do Sakai

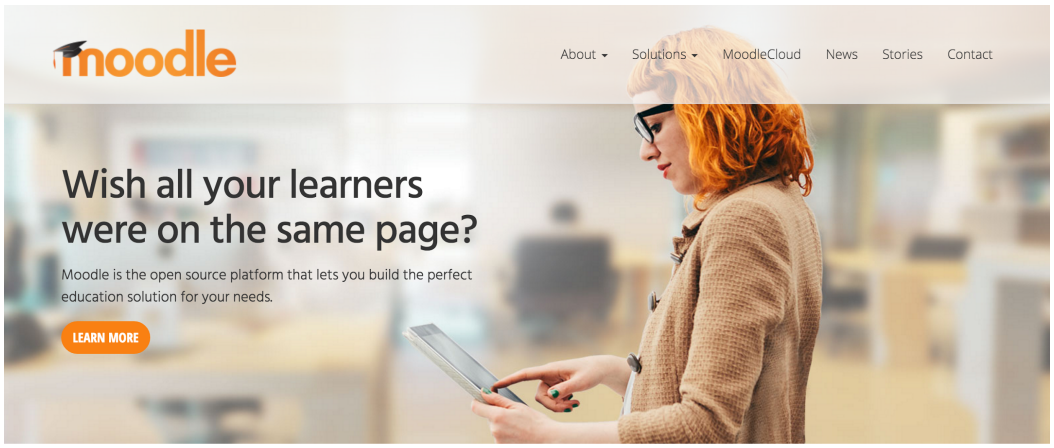


O projeto roda sobre uma plataforma Java e utilizando o servidor Apache Tomcat e possui já cerca de 11 anos desenvolvimento. Permite personalização por meio de módulos que podem ser adicionados ao núcleo do sistema.

2.2. Moodle

(Alves, et al., 2009) acreditam que o sucesso do Moodle, como mostra a Figura 5, é devido à alta possibilidade de customização na criação de plataformas de educação. Desse modo, cada universidade pode escolher e personalizar seus respectivos sites de acordo com o que achar mais relevante.

Figura 5 – Página inicial do Moodle

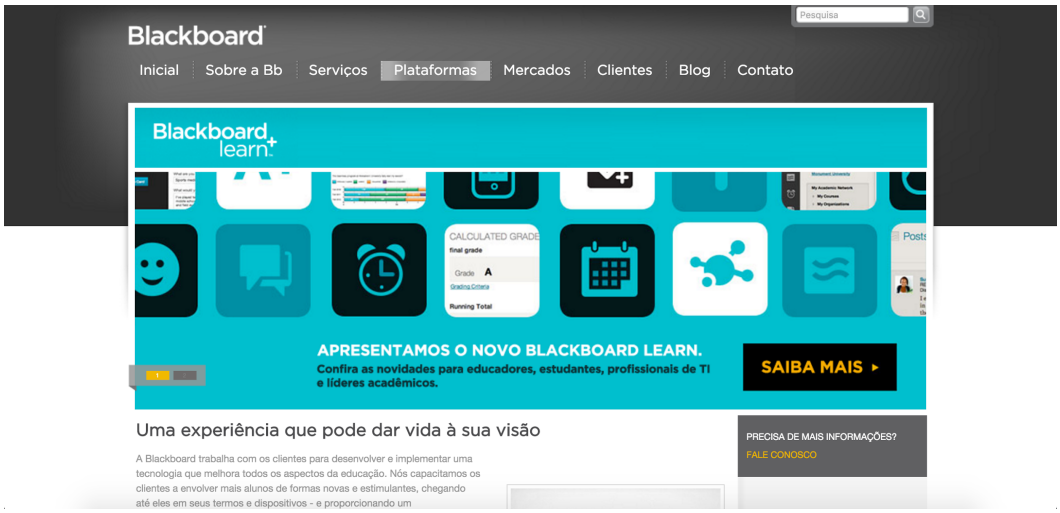


O Moodle é amplamente utilizado no mundo inteiro: existem mais de 60 mil sites em 223 países[2]. Alguns exemplos internacionais são: Universidade do Porto em Portugal, Technische Universität Darmstadt na Alemanha e Monash University na Austrália. Já no Brasil alguns grandes exemplos são faculdades públicas como a Escola Politécnica da Universidade de São Paulo (EP-USP), Universidade Federal do Rio de Janeiro (UFRJ) e Universidade Federal da Bahia (UFBA), e um exemplo de faculdade particular é a Pontifícia Universidade Católica de São Paulo (PUC-SP).

2.3. Blackboard Learn

A Blackboard, cuja a página inicial é ilustrada pela Figura 6, foi fundada em 1997 e tem sua sede em Washington DC, mas tem escritórios no mundo inteiro. Além da plataforma educacional, a Blackboard Learn, a Blackboard ainda oferece outras plataformas como a Analytics, Collaborate, Connect e Mobile. Dessa forma, atua em diversos mercados como governos, corporativo e de ensino (Blackboard, 2015).

Figura 6 – Página inicial da Blackboard Learn



Para o desenvolvimento da Blackboard Learn procurou-se levar em conta a experiência dos professores e estudantes, os principais beneficiados da plataforma, para promover ao máximo o aprendizado online. Além disso, existem mais de 5 mil instituições de ensino superior utilizando a Blackboard Learn no mundo todo. Segundo

[2] Moodle – http://moodle.org

o site oficial, 72% das 200 melhores universidades usam a Blackboard Learn. Para ilustrar, alguns exemplos americanos são a renomada Princeton e Washington State University. Já entre as universidades nacionais, podemos citar Insper, FAAP e ESPM (Blackboard, 2015).

3. CONCEITOS E TECNOLOGIAS

Este capítulo discute as tecnologias utilizadas no projeto, aprofundando e motivando o por quê do uso de cada tecnologia e como elas se alinham com as diretrizes traçadas para o projeto. Com tecnologia entende-se não somente ferramentas de software utilizadas no projeto, mas também técnicas e métodos de desenvolvimento de software, que se encaixam em Engenharia de Software.

3.1. Técnicas e métodos de Engenharia de Software

O desenvolvimento do *framework* Gabaritei é, na verdade, uma aplicação de técnicas de Engenharia de Software modernas e bem consolidadas. Foram três pontos bases para o desenvolvimento do projeto, os quais são discutidos neste capítulo: o Scrum, o Desenvolvimento Orientado a Testes e desenvolvimento à distância por meio de controle de versões.

O Scrum é um *framework* de processo consolidado e amplamente utilizado pelas empresas de desenvolvimento de software (Scrum.org, 2015) (Scrum.org, 2014). Essa técnica de desenvolvimento ágil foi selecionada para estruturar o projeto de maneira rápida. Além disso, como dois integrantes encontravam no exterior (Alemanha e França) durante certo período, teve-se a oportunidade de desenvolver técnicas de desenvolvimento à distância. O Scrum, nesse contexto, se encaixou perfeitamente às necessidades do projeto, uma vez que permitiu a divisão do projeto em tarefas menores e possibilitou o desenvolvimento independente do projeto por membros em países diferentes.

O Scrum, portanto, foi utilizado para estruturar o projeto de maneira segmentada, permitindo o desenvolvimento por membros separados fisicamente (no caso, entre países). Além disso, a produção de entregáveis regulares e em prazos curtos, permitiu um maior direcionamento do projeto, o alinhando com as diretivas e objetivos iniciais propostos na documentação, bem como ficando de acordo com a perspectiva do cliente, que teve participação ativa no desenvolvimento do projeto.

O Desenvolvimento Orientado a Testes foi uma técnica fundamental ao desenvolvimento do projeto. Um bom planejamento de testes e validação das funcionalidades desenhadas na etapa de desenvolvimento da estrutura do projeto possibilitou garantir a flexibilidade da aplicação. Os testes desenvolvidos foram, principalmente, de dois tipos: os testes de unidade e de integração. Os testes de unidades focaram nos módulos de maneira individual, enquanto os testes de integração focaram na interação dos módulos desenvolvidos e, portanto, no funcionamento do sistema como um todo.

Por módulos, entende-se uma unidade do sistema que realiza uma tarefa específica. Por exemplo, tem-se o módulo de questões, responsável por criar, editar, visualizar e editar questões. Este módulo pode ser testado internamente através dos testes de unidade e, depois, integrado com outros módulos do sistema. Nesse exemplo, o módulo de questões interage com o módulo de testes (um conjunto de questões). Dessa forma, deve-se utilizar testes de integração para verificar se juntos esses módulos (ou unidades) realizam as funções esperadas.

Outro ponto fundamental do Desenvolvimento Orientado a Testes é que criar os testes antes de criar o código garante que o código desenvolvido se adapte às funcionalidades planejadas e não o contrário.

O conjunto de testes constitui uma ferramenta única e importante para garantir o funcionamento do sistema. Além disso, a flexibilidade de alterar o código e conhecer o impacto das suas alterações é de essencial importante para um *framework* como o projeto Gabaritei está sendo construído. Como o intuito do projeto é que ele seja adaptado e utilizado em diversos ambientes, com diversas configurações e modificações, é importante que as funcionalidades inicialmente planejadas continuem funcionando e, além disso, que os desenvolvedores possam ter controle sobre suas modificações sobre o sistema. Para isso, a construção de uma base sólida de testes foi fundamental.

Por fim, o controle de versões escolhido foi o Git, pela sua extensa documentação e base de usuários. Um projeto com desenvolvimento à distância irá necessitou de um

controle rigoroso sobre o código fonte do projeto, o que foi proporcionado e organizado pela ferramenta Git e pelo serviço GitHub.

3.2. Tecnologias, *frameworks* e softwares auxiliares

3.2.1. Ruby on Rails

Ruby é uma linguagem de programação moderna e bastante flexível. Sua filosofia de ser simples, com uma sintaxe muito fluída e fácil de ler, permite que possamos atingir nosso objetivo de disponibilizar um *framework* que seja fácil de ser entendido e modificado (Chaffee, 2012) e (Grosenbach, 2010).

Ruby é uma linguagem consolidada e com muitos anos de desenvolvimento. A comunidade de utilizadores é grande, a quantidade de *gems* (RubyGems.org, 2009), módulos desenvolvidos por terceiros que podem ser integrados a sua aplicação é grande. Poucas vezes o desenvolvedor vai ter que implementar alguma funcionalidade complexa não relacionada a aplicação de forma direta, muitas vezes irá se utilizar desse completo sistema de *gems* para que possa construir uma aplicação sólida e depreender esforços sentido da aplicação e não da sua infraestrutura.

O projeto tem como base principal o *framework* para desenvolvimento Web em Ruby on Rails[3] Este *framework*, além de obrigar o desenvolvimento do *design pattern* MVC (Reenskaug, 2007) e, portanto, o desenvolvimento de um código bem organizado, nos permite desenvolver aplicações Web de forma rápida, com pouca preocupação com a infraestrutura técnica do projeto. Além disso, o ambiente propicia ferramentas para focar nas funcionalidades e nas técnicas descritas no capítulo anterior e menos na parte técnica do desenvolvimento para *Web*.

Além disso, pelo processo Scrum, é recomendável produzir entregáveis regularmente e, portanto, o tempo de desenvolvimento do projeto é curto. O Ruby on Rails permite um rápido desenvolvimento de aplicações, o que nos auxiliaria nesta parte.

[3] Página oficial Ruby on Rails - http://rubyonrails.org/

3.2.2. Ferramentas de testes Ruby on Rails

Deve-se considerar o fato de que o Ruby on Rails já possui *scripts* e suítes de teste padrão, incluído na sua instalação base e disponível em todos os projetos por ela criada. Rake é uma ferramenta que permite estruturar os testes em arquivos de forma lógica e, também, fornece uma ferramenta para realizar todos os testes de maneira rápida.

Além disso, as extensões da instalação padrão do Ruby on Rails, as chamadas *gems*, nos permite estender as capacidades de testes da plataforma padrão. Foram utilizadas duas extensões do Ruby on Rails:

- **Capybara:** um auxiliador para os testes de integração, que permite simular a interação dos usuários com a aplicação desenvolvida. Fornece ricas funcionalidades para interagir com o navegador, (por exemplo, enviar comandos, mudar de URL, obter o HTML, *drivers* para diversos navegadores), bem como uma sintaxe bem simples e prática para simulação de interações do usuário com a página.

- **Selenium:** é um *driver* para o Capybara que permite o envio de comandos diretamente para um *browser* verdadeiro. Com este *driver*, é possível manipular os tradicionais navegadores Google Chrome e Mozilla Firefox, enviando comandos por meio da suíte de testes do Gabaritei e resgatando os resultados esperados.

O ambiente de desenvolvimento utilizado será o Mac OS X ou Linux, uma vez que as ferramentas Ruby on Rails utilizadas possuem maior integração e compatibilidade com estas plataformas do que com o Microsoft Windows. Ainda assim, durante o projeto, os testes foram realizados nos três sistemas operacionais, de forma que não exista imprevistos caso o usuário ou o desenvolvedor necessitar rodar o projeto na plataforma Windows.

3.2.3. Ferramentas *web*

Entre as tecnologias utilizadas no lado do cliente, foi utilizado o *framework* de estilos Twitter Bootstrap. Amplamente documentado e utilizado, este *framework* permite o desenvolvimento rápido, fácil e confiável de visuais e estilos para páginas *web*. O *framework* possui tecnologias para gerar menus, modais, botões e formulários visualmente bonitos e muito práticos. Além disso, ainda permite a utilização do site através de plataforma móveis, como celulares e *tablets,* pois possui suporte a *designs* responsivos, ou seja, que se adaptam, sem interação do usuário, a diferentes resoluções (Eis, 2011) (Otto & Thornton, 2015).

Além disso, foi usada a biblioteca jQuery que, além de ser necessária para o correto funcionamento do Bootstrap, permite maior flexibilidade na manipulação de elementos da página.

3.2.4. AngularJS e componentes associados

Angular JS é uma tecnologia recente, porém vem ganhando muitos adeptos e vem sendo utilizada para diferentes tipos de projetos. Por exemplo, a implementação do Youtube para Playstation 3 utiliza o AngularJS (Google, 2014). Além disso, o *framework* é mantido sob licença do MIT pela empresa Google*,* o que reflete na sua manutenção ao longo termo e assegura sua qualidade do código (Palmer, 2011).

AngularJS foi escolhido por três motivos principais. Primeiramente, nos permite a organização do código Javascript de forma que ele fique estruturado e fácil de ser entendido por terceiros. Isso é o contrário do que acontece quando se utiliza uma implementação pura utilizando apenas uma biblioteca. Isso aconteceria com a biblioteca JQuery que, apesar de ser muito poderosa e utilizada pela aplicação desse trabalho, não é um *framework,* ou seja, não obriga a utilizar uma estrutura fixa (Lamb, 2014).

O segundo motivo se refere à sua integração ao ecossistema da ferramenta NodeJS. Assim, pode-se utilizar ferramentas poderosas, dando ênfase ao conjunto de tecnologias Jasmine e Karma[4].

O Jasmine é uma ferramenta que nos permite criar testes automatizados para Javascript independente de qualquer tipo de *framework* utilizado no programa a ser testado (Jasmine, 2015). O grande atrativo da ferramenta *Jasmine* para o uso no desenvolvimento foi sua simplicidade sintática e sua completa integração com o Karma (AngularJS, 2015).

O Karma permite executar os testes através do Jasmine de forma mais personalizada, permitindo adicionar relatórios de *code coverage* (Karma, 2015), injeção de HTML diretamente nos testes permitindo, assim, o teste de diretivas; permite também depurar o código por meio do *browser* (Karma, 2015) (Karma-Runner, 2015).

Por último, o AngularJS vai de encontro com o nosso objetivo desse trabalho de tornar o site moderno e dinâmico. Os inúmeros módulos disponíveis para AngularJS permitem realizar tarefas que tornam a página mais dinâmica e fácil de utilizar. Ações comuns nas páginas mais interativas hoje, como arrastar e soltar e internacionalização, são muito fáceis de serem utilizadas. A estrutura modular e a habilidade de estender vocabulário ao HTML nos permite criar uma página extremamente dinâmica mas que, ainda assim, possua um HTML limpo e de fácil entendimento tanto por desenvolvedores quanto por profissionais de *design*. Essa característica é muito importante para ambientes profissionais, em que muitas vezes o *design* e a codificação do seu site são realizados por pessoas diferentes.

3.2.5. *ActiveRecord* e banco de dados

Não foi necessário lidar diretamente com o banco de dados neste projeto. A ferramenta Ruby on Rails disponibiliza total integração com um componente

[4] GTAC 2013: Karma - Test Runner for JavaScript-https://www.youtube.com/watch?v=YG5DEzaQBIc

denominado *ActiveRecord*, responsável por abstrair boa parte da complexidade da interface entre a aplicação e o banco de dados[5].

Com o *ActiveRecord*, tem-se a total flexibilidade no que diz respeito à tecnologia de banco de dados, pois ele se integra facilmente com qualquer banco de dados relacional, como MySQL, SQLite, PostgreSQL, etc. Ele pode ainda integrar-se com bancos de dados não relacionais, mas nesse caso algumas configurações extras devem ser feitas.

Neste projeto, durante a etapa de desenvolvimento, utilizou-se o banco de dados SQLite, pois ele necessita de muita pouca configuração, e não é necessário possuir um ambiente de execução dedicado para manipulá-lo, tratando-se portanto de uma solução flexível para o desenvolvimento.

Por outro lado, para o ambiente de produção, optou-se pelo MySQL, que é um banco de dados mais poderoso que o SQLite, mas ao mesmo tempo de fácil configuração e aberto. Além disso, em geral os serviços de plataforma na nuvem não dão suporte ao SQLite, um outro motivo pelo qual optou-se pelo MySQL para o ambiente de produção.

3.2.6. Ferramentas de documentação

Adotaram-se duas ferramentas de documentação automática, uma para o código do servidor (Ruby) e outra para o código do cliente (Javascript).

Para a parte do servidor, utilizou-se a ferramenta YARD[6], que permite a geração automática das páginas HTML da documentação com base em comentários estruturados inseridos no código. Essa parte é importante em especial para a documentação do modelo de dados, dado que podem-se fazer descrições mais detalhadas nas classes de *model* e da parte da lógica de negócios.

[5] RailsGuides – Active Record Basics - http://guides.rubyonrails.org/active_record_basics.html
[6] A Ruby Documentation Tool - http://yardoc.org

Para a parte do cliente, utilizou-se o componente *grunt-ng-docs*[7] por meio da ferramenta Grunt Task Runner, o qual gera automaticamente páginas HTML da mesma forma que o YARD. Essa parte é importante em especial para a documentação da interface e de parte da lógica de negócios.

3.2.7. Ferramentas de versionamento

O software de controle de versão a ser utilizado é o Git que, conforme explicitado na sessão 3.1, é importante para manter o código organizado durante todo o ciclo de vida do sistema. Seu nível de confiança é visto pelo número de projetos importantes que atualmente utilizam o sistema de versionamento Git: Ruby[8], Rails[9], AngularJS[10], Microsoft[11], NodeJS[12], entre muitos outros.

O sistema online GitHub será utilizado para disponibilizar o código do Gabaritei. Suas ferramentas avançadas aumentam em muito o alcance do *framework* Gabaritei: sistema de *Fork* (GitHub, 2015), *Pull Requests* (GitHub, 2015), além de ferramentas que tornam o repositório como uma rede social, onde outros desenvolvedores podem marcar o repositório como favorito, seguir as novidades e modificações do projeto. Ainda mais, é possível criar uma documentação online fácil e gratuita utilizando o sistemas de páginas do GitHub.

3.3. Considerações finais do capítulo

Este capítulo teve por objetivo descrever de forma sintética as tecnologias utilizadas em nosso projeto e, também, as relacionar com as principais diretrizes do projeto Gabaritei. As tecnologias escolhidas de *front-end*, como o AngularJS, visaram o encontro com a diretriz de uma interface moderna e dinâmica, ao mesmo tempo que

[7] Grunt plugin to create a documentation - https://www.npmjs.com/package/grunt-ngdocs

[8] Repositório Ruby - https://github.com/ruby/ruby

[9] Repositório Rails - https://github.com/rails/rails

[10] Repositório AngularJS - https://github.com/angular

[11] Repositório Microsoft - https://github.com/Microsoft

[12] Repositório NodeJS - https://github.com/nodejs/node

permitiu o desenvolvimento testes automatizados, um padrão de escrita de código utilizando o *AngularJS Style Guide.*

Do ponto de vista *server-side*, utilizamos o Ruby on Rails para garantir uma base em uma tecnologia sólida e facilmente modificada. E, assim como o *front-end*, que também permite o desenvolvimento de testes automatizados, indo de encontro com a diretriz de criar um sistema testável e modificável.

Por fim, foram apresentadas as tecnologias de controle de versão, que auxiliam no desenvolvimento de tecnologias a distância bem como um sólido controle sobre as alterações do projeto. E, seguindo a diretriz de tornar o sistema livre e acessível, apresentou-se o GitHub, onde o código foi disponibilizado livremente na internet.

4. DESENVOLVIMENTO DO *FRAMEWORK*

Este capítulo tem como objetivo descrever os requisitos do sistema (funcionais e não funcionais) e verificar como foi planejada e executada a implementação das funcionalidades discutidas. São apresentados os documentos construídos para desenvolver o sistema, entre eles Diagramas de Caso de Uso, Descrição de Casos de Uso, *Backlog* do produto, entre outros a serem discutidos. Por fim, será discutida a implementação de requisito à requisito.

4.1. Especificação de requisitos

4.1.1. Requisitos funcionais

Os requisitos funcionais apresentados nessa seção foram baseados no documento de *Backlog* e casos de uso que se encontram em anexo neste documento nos Apêndices A, B e C. Tem-se como requisitos, portanto:

- **Administração de usuários:** correspondem à criação, importação e manutenção de base de usuários no sistema. Inclui a separação em diversos tipos de usuários (Papéis), para que as funcionalidades do sistema possam ser acessadas de acordo com o papel do usuário dentro do sistema.
- **Administração de cursos:** é chamado de curso um agregado de disciplinas e professores. Ele representa o que, em um colégio, é chamado de sala de aula. Uma sala contém alunos, uma grade com diversas disciplinas e um professor.
- **Administração de disciplinas e áreas:** o sistema permite a subdivisão de disciplinas em áreas. Assim, uma disciplinas, como Física, pode ser subdividida em áreas como Mecânica, Ótica, Eletricidade, etc.
- **Administração de perguntas:** os professores podem criar e editar perguntas. Os alunos podem responder perguntas e recomendá-las a outros alunos. As perguntas podem ser da forma discursiva ou múltipla escolha e o sistema deve estar apto a tratar de forma diferente esses dois estilos de pergunta.
- **Administração de conteúdo:** os usuários devem ser capazes de carregar o site com conteúdo: documentos de texto, arquivos PDF, imagens, etc. O

sistema deve ser capaz de armazenar e permitir que os usuários façam download desses arquivos.

- **Administração de testes:** é chamado de teste um agregado de perguntas. Os professores devem ser capazes de agregar perguntas em um Teste; os alunos devem ser capazes de responder estas perguntas através do sistema; e os professores devem ser capazes de observar o desempenho dos alunos através do sistema.

4.1.2. Requisitos não funcionais

Os requisitos não funcionais foram obtidos através da experiência pessoal dos participantes do projeto de formatura, em conjunto com possíveis usuários do sistema e, também, com a opinião dos próprios estudantes da Universidade. Os principais requisitos não funcionais do sistema Gabaritei são:

- **Interface responsiva:** responsiva se refere à capacidade do sistema ser executado e adaptável, sem prejuízo visual, a diversos tipos de dispositivos: computadores, *notebooks*, celulares e *tablets*.
- **Tecnologias:** utilizar tecnologias modernas e no estado da arte para desenvolvimento web. Utilizando tecnologias mais novas, tanto no *front-end* quanto no *back-end,* o sistema foi preparado para ter sua vida estendida nos próximos anos.
- **Simplicidade:** o sistema deve ser simples para os usuários, ou seja, as funções devem ser evidentes e fáceis de achar, com o mínimo de informação na tela.
- **Testes automatizados:** o sistema deve possuir uma infraestrutura pronta para realizar seus testes. Com a utilização de relatórios de cobertura e testes de unidades, o sistema fica mais fácil para ser desenvolvido por terceiros.
- **Código livre e aberto:** o código do sistema deve ser livre e aberto. Escolas e cursinhos, bem como desenvolvedores independentes devem poder utilizar o sistema gratuitamente, bem como modificá-lo livremente para adequá-lo ao seu uso.

4.1.3. Modelo de casos de uso

Os casos de uso foram levantados junto a potenciais clientes e foram pontuados em um documento denominado *Product Backlog* (consultar Apêndice A). Nele foram incluídas as funcionalidades esperadas do sistema, bem como a pontuação calculada com base no peso da funcionalidade para o desenvolvedor, em termos de dificuldades de implementação, e na importância para o cliente. Destes dois fatores, foi criada uma métrica que dá uma direção para onde seguir, ou seja, qual funcionalidade é mais importante de ser desenvolvida primeiro.

Depois, os requisitos foram filtrados e foi construída uma lista de casos de uso a serem considerados no nosso projeto. Os casos de uso são importantes para a criação dos testes de validação de cada funcionalidade. Este documento é constituído por três partes básicas: um diagrama indicando casos de uso por recurso e quais atores estão envolvidos (ver Apêndice B), uma listagem dos casos de uso (esta foi classificada por ator e por funcionalidade) e uma descrição detalhada dos casos de uso (consultar Apêndice C).

A listagem de casos de uso é muito extensa, portanto a descrição detalhada dos casos de uso análogos do Apêndice C foi generalizada. Um exemplo que pode ser citado são os casos de uso de CRUD de Exercícios, Exames, Usuários, etc..

4.1.4. Modelo de dados e diagrama de classes

Analisando os casos de uso, foram identificadas as entidades envolvidas no sistema, seus atributos e suas associações (ver Apêndice E). Em seguida, com base nessa análise, foram definidas as tabelas no banco de dados e as classes que acessam os dados (os *models* na arquitetura MVC).

Para facilitar a leitura e o entendimento do modelo de classes, o diagrama de classes foi dividido em quatro sub-diagramas que giram em torno das entidades fundamentais do sistema. O Apêndice D apresenta esses quatro sub-diagramas:

- Diagrama de classes - Cursos;
- Diagrama de classes - Questões;
- Diagrama de classes - Materiais, recomendações e *ratings*;
- Diagrama de classes - Papéis, permissões e usuários.

4.2. Arquitetura e integração cliente-servidor

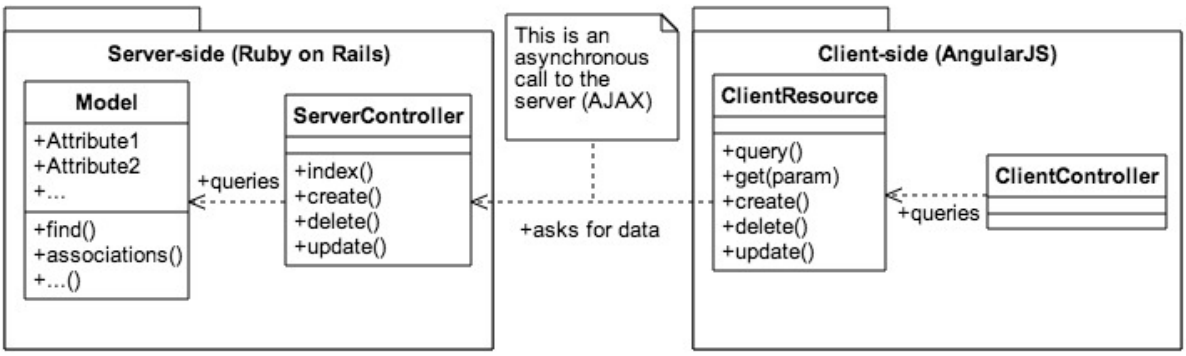
A questão da integração cliente-servidor é resolvida por meio da integração entre os componentes desenvolvidos por meio do AngularJS no lado cliente, e os componentes desenvolvidos por meio do Ruby on Rails no lado servidor (Green & Pai, 2015).

Ambas as ferramentas implementam o *design-pattern* MVC. No lado do cliente, os *controllers* gerenciados pelo AngularJS manipulam diretamente a árvore de elementos das páginas HTML (*views*) com base nos dados que os denominados *resources* (o equivalente aos *models*) fornecem. Os *resources*, por sua vez, são componentes que realizam requisições assíncronas ao servidor (por meio da ferramenta AJAX), para poder obter os dados exigidos pelos *controllers*.

No lado do servidor, os *controllers* recebem as requisições do cliente para diversas ações. Para responder a essas requisições, eles manipulam os *models*, ou seja, as classes de interface com o banco de dados para obter os dados necessários. Uma vez que os dados são obtidos, os controllers os transformam em respostas na forma de JSON.

O diagrama de classes da Figura 7 esquematiza a integração cliente servidor:

Figura 7 – Integração cliente-servidor



Além disso, o Apêndice F ilustra a integração cliente-servidor por meio de um diagrama de sequência genérico, pois a visualização, adição, atualização ou remoção de dados funcionam de forma análoga.

4.3. Banco de dados e acesso aos dados

A modelagem do banco de dados fez-se por meio do *ActiveRecord*. Como descrito na seção 3.2.5, a ferramenta cria uma camada de abstração entre a aplicação e o banco de dados: logo, não foi necessário utilizar a SQL para acessar o banco de dados. Ao invés disso, foram definida as *migrations*, que nada mais são que classes manipuladas pelo *ActiveRecord* na criação do banco de dados. Declaram-se nas *migrations* as tabelas que se quer construir (campos e atributos), além das relações entre elas (chaves estrangeiras).

Migrations são muito versáteis, pois permitem a modificação do banco de dados em ambiente de produção, fazendo a migração de dados de forma coerente e automática. Entretanto, não foi feito uso dessa funcionalidade no projeto, dado que o banco pode facilmente ser apagado e recriado pelas *migrations* quando necessário. Durante o desenvolvimento, não trabalhou-se com dados de produção, somente com dados de teste. Dessa forma, torna-se mais eficiente fazer modificações diretamente nas *migrations* de base e recriar o banco do que fazer diversas migrações com diversas versões de *migrations*, ou seja, todo o modelo de dados é descrito nos poucos arquivos das *migrations* de base, e não estaria distribuído por diversos arquivos, facilitando a manutenção do modelo de dados.

Basicamente desenvolveram-se duas *migrations* principais: *CreateModelRelations* e *AddModelFields,* cujos extratos são ilustrados pelas Figura 8 e Figura 9, respectivamente.

Figura 8 – Extrato da *migration CreateModelRelations* com a declaração das tabelas de matérias e áreas

```
class CreateModelRelations < ActiveRecord::Migration
 def change
  create_table :subjects do |t|
   t.timestamps null: false
  end
  create_table :fields do |t|
   t.belongs_to :subject, index: true, required: true
   t.timestamps null: false
  end
[...]
```

Figura 9 – Extrato da migration *AddModelFields* com os campos da tabela de requerimentos de inscrição

```
class AddModelFields < ActiveRecord::Migration
 def change
  # Registration requests
  add_column :registration_requests, :first_name, :string
  add_column :registration_requests, :last_name, :string
  add_column :registration_requests, :email, :string
  add_column :registration_requests, :birthdate, :datetime
  add_column :registration_requests, :text, :text
  add_column :registration_requests, :response, :text
  add_column :registration_requests, :accepted, :boolean
[...]
```

A primeira declara todas as tabelas do banco e as relações entre elas, sem declarar os campos. A segunda declara os campos e os atributos dos campos para cada tabela. Estruturou-se dessa forma, pois a manutenção se torna mais fácil, separando as relações (chaves estrangeiras) dos campos das tabelas.

Há ainda outras *migrations* que são criadas automaticamente por componentes para o seu funcionamento. Por exemplo, o componente que implementa a execução de tarefas em *background* (*delayed jobs*) cria uma *migration* para gerar as tabelas necessárias para lidar com essas tarefas. Outro exemplo é o componente Devise, responsável pelo sistema de autenticação do *framework*. Ele gera campos específicos de seu uso na tabela de usuários, e para tanto cria uma *migration*.

O *ActiveRecord* interpreta todas as *migrations* e gera uma única *migration*, denominada Schema. Essa *migration* então é utilizada para criar o banco de dados. Vale ressaltar que a sintaxe das *migrations* é muito mais simples que a sintaxe de um documento SQL convencional, tornando a manutenção do banco de dados muito mais fácil.

Uma vez criado o banco de dados, o *ActiveRecord* fornece uma interface para a realização de *queries* de forma simplificada. Quando um elemento de uma tabela é selecionado, seus campos são mapeados nos atributos de uma classe de *model*, que também se encarrega de atualizar o banco de dados quando necessário. Logo, o banco de dados se torna transparente à aplicação.

4.4. Projeto de interface do usuário

Nesta seção está descrita a interface homem-computador (IHC) do Gabaritei.

4.4.1. Características da interface

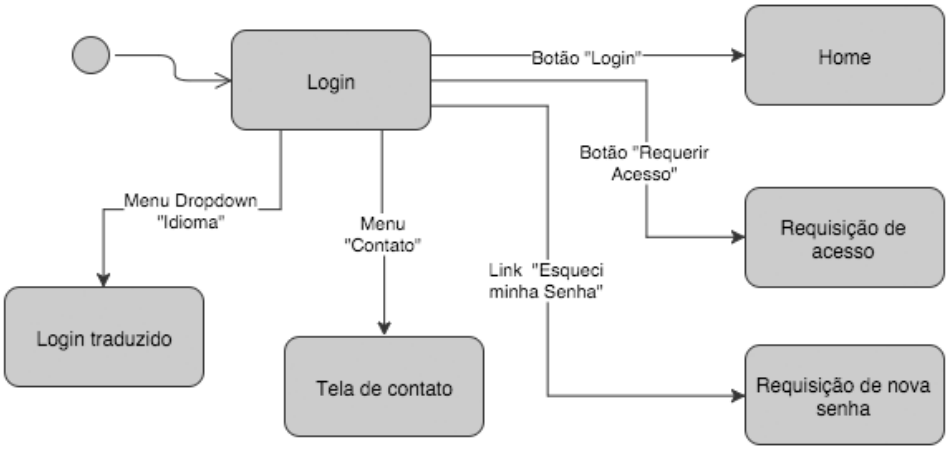
Como mencionado no Capítulo 1, um dos principais fatores motivacionais deste projeto de formatura era a implementação de um *framework* moderno, intuitivo e fácil de usar. Dessa forma, a interface resultante do sistema é *responsive*, de modo que

ela se adapta tanto a computadores quanto a *smartphones* e *tablets*. Além disso, ela é baseada no Twitter Bootstrap que é customizada com o uso de SASS e CSS, e tem uma característica *flat*, uma tendência das interfaces atuais.

4.4.2. Protótipo de Navegação de Telas

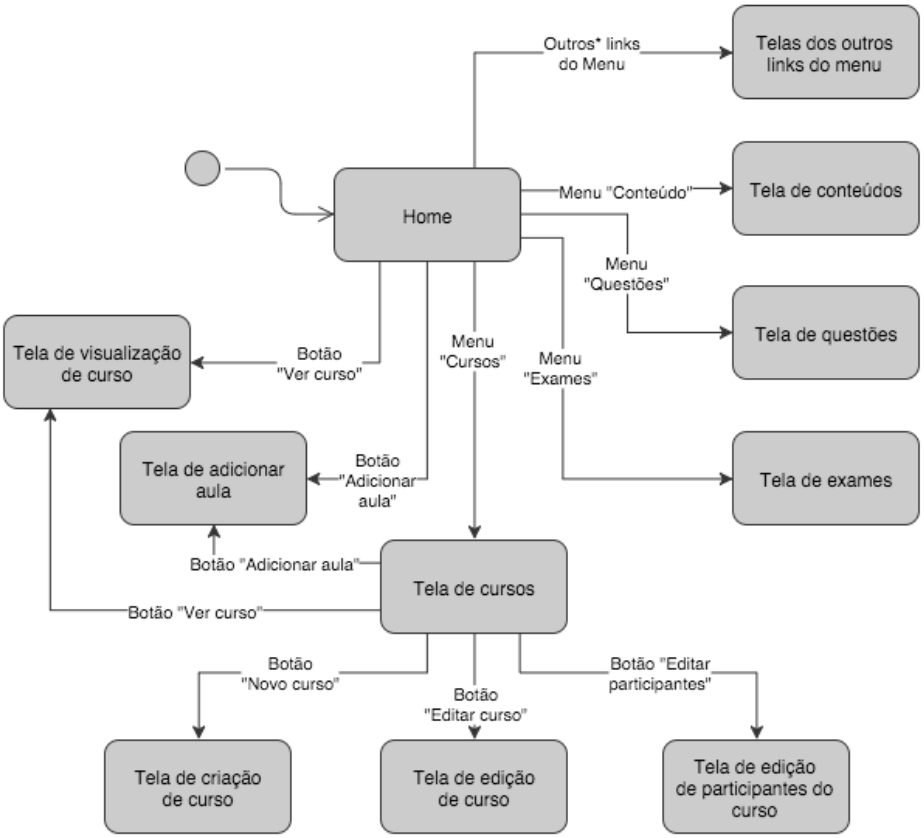
A Figura 10 mostra o fluxo de telas a partir da tela de *login.* Pode-se perceber que o visitante tem acesso apenas ao contato da escola ou cursinho. Caso ele seja um aluno ou professor, que por algum motivo ainda não esteja cadastrado no sistema, ele pode fazer um requisição de acesso, através do botão "Requerir acesso".

Figura 10 – Fluxo da tela de *login*



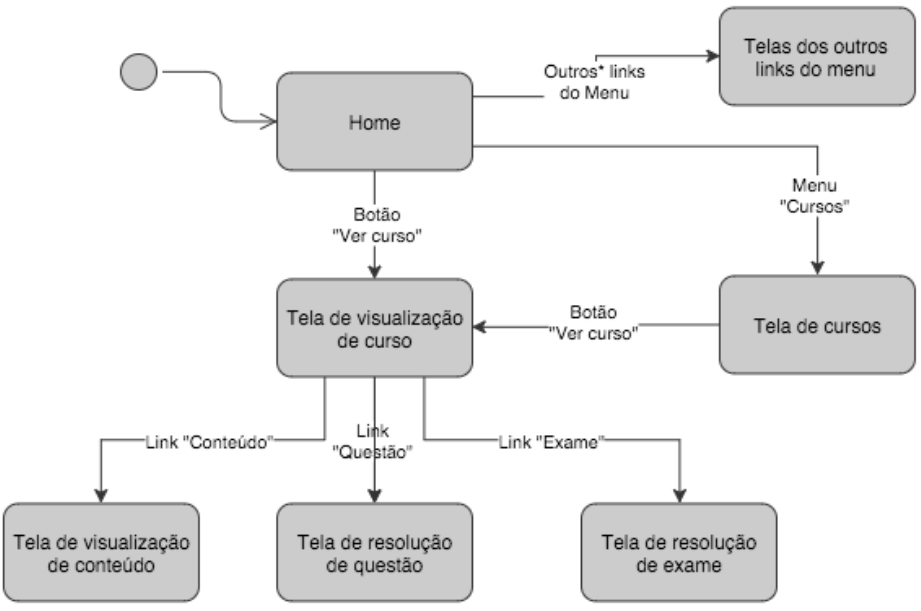
Após o *login* como aluno ou professor, ambos são direcionados para a tela inicial, conhecida como *Home*. A Figura 11 e a Figura 12 apresentam os fluxos de telas a partir da *Home* segundo a visão do professor e do aluno, respectivamente. Com esses fluxos, buscou-se facilitar o uso pelo professor ou aluno, de modo que as funcionalidades mais usadas por cada um tenham acesso fácil, se não, direto da tela inicial.

Figura 11 – Fluxo de telas na visão do professor



* Links que todo o usuário autenticado tem, tais como Perfil, Logout, Contato, etc.

Figura 12 – Fluxo de telas na visão do aluno



* Links que todo o usuário autenticado tem, tais como Perfil, Logout, Contato, etc.

4.5. Implementação

4.5.1. AngularJS *style guide*

Grande parte da lógica do Gabaritei está desenvolvida em Javascript e AngularJS; por isso, procurou-se padronizar o estilo de escrita de código, de forma a facilitar a leitura do código por parte dos desenvolvedores. Foi adotado um estilo de codificação criado por John Papa[13], que foi premiado e foi recomendado por grande parte dos desenvolvedores.

4.5.2. Autenticação e usuários

Para implementar o mecanismo de autenticação, utilizou-se o componente *Devise* integrável ao ambiente do servidor (Plataforma Tec, 2015). Este componente se encarrega de gerenciar senhas e o processo de autenticação, disponibilizando métodos para acessar o usuário que está atualmente autenticado e de controlar as respostas do servidor caso algum usuário não esteja autenticado (emitindo respostas 401: *Unauthorized access*).

Na criação dos usuários, o componente também é capaz de gerar senhas aleatórias. A ideia é que cada usuário, ao ser cadastrado no sistema, receba uma senha gerada aleatoriamente, e seja forçado a fornecer uma nova senha no primeiro acesso. O Devise encripta as senhas antes de salvá-las no banco de dados, garantindo maior proteção ao usuário, que será o único a saber sua senha.

Para o cliente, utilizou-se uma biblioteca denominada *angular-devise*[14] que implementa o código necessário para acessar as informações do Devise a partir do Javascript. Dessa forma, pode-se controlar o acesso dos usuários tanto no cliente como no servidor, garantindo que usuários não possam acessar conteúdos que ele não estão autorizados a ver.

---

[13] AngularJS style guide por John Papa - https://github.com/johnpapa/angular-styleguide
[14] Repositório angular-devise - https://github.com/cloudspace/angular_devise

4.5.3. Importação de dados

Para fazer a importação de dados por meio de arquivos, foi necessário resolver inicialmente dois problemas. O primeiro foi fazer o *upload* de arquivos. O AngularJS não dá suporte para *upload* de arquivos, sendo necessário portanto, utilizar uma biblioteca auxiliar chamada *ng-file-upload* [x26] que permite o upload de arquivos mantendo-se a estrutura do AngularJS. O segundo problema foi conseguir realizar a leitura de diversos tipos de arquivos. O ambiente Ruby possui nativamente uma biblioteca para leitura de arquivos CSV, entretanto a ideia era fornecer suporte para outros arquivos, sobretudo planilhas do Excel.

Foi possível implementar uma lógica de leitura de arquivos por meio de uma biblioteca externa chamada *roo*[15]. Essa biblioteca nos permitiu ler os arquivos de forma estruturada, abstraindo o formato, isto é, podem-se ler arquivos CSV, XLS, XLSX e ODT com o mesmo código.

Por fim, dado que a importação de dados pode ser um processo demorado, optou-se por fazê-la em um processo em segundo plano. Para tanto, utilizou-se um componente denominado *Delayed jobs*, que permite a execução de tarefas estruturadas em um pilha em segundo plano. Essa mesma biblioteca pode ser usada para enviar e-mails[16].

4.5.4. Internacionalização

Visando gerar uma aplicação que pode ser facilmente traduzida para qualquer linguagem, passamos a utilizar uma biblioteca angular chamada *angular-translate*[17]. Esta ferramenta funciona da seguinte maneira: ao invés de colocar texto diretamente no código HTML da página, passamos a colocar códigos de fácil identificação. Estes códigos são colocados em um arquivo YAML, que permite a fácil tradução em um único arquivo centralizado de todo o texto usado na aplicação. Utilizar o código desse

[15] Repositório roo - https://github.com/roo-rb/roo
[16] Repositório delayed jobs - https://github.com/collectiveidea/delayed_job
[17] Repositório angular-translate - https://github.com/angular-translate/angular-translate

jeito também nos permite isolar os componentes de visualização dos da lógica de negócios.

4.5.5. Modais, mensagens e paginação

Nenhuma ferramenta de modal se adaptou aos fins desse projeto e, por isso, criou-se uma pequena diretiva angular que pode ser utilizada de maneira genérica em toda a aplicação. Dessa forma, além de tornar genérica a utilização, centraliza-se toda a codificação específica dessa parte em um único lugar. Utilizando em forma de diretiva, cada modal recebe três parâmetros importantes:

- Título: código da tradução referente ao título
- Corpo: código de tradução referente ao corpo do modal
- *Callback*: função acionada caso o usuário confirme alguma ação da modal

O serviço de mensagens também foi pensando e desenvolvido pelos membros do projeto de formatura, pois era necessário um serviço simples o suficiente e universal. Dessa forma, foi criada uma API própria que permite que os desenvolvedores invoquem uma mensagem para o usuário utilizando comandos simples.

Com relação à paginação, foi necessário encontrar um componente que pudesse integrar bem o funcionamento do AngularJS com o servidor Rails. Para isso, utilizamos um componente chamado *paginate-anything*[18], presente tanto no servidor como o cliente, provendo métodos para implementarmos facilmente a paginação.

4.5.6. Papéis, permissões, requerimentos e contextos

As funções disponíveis para cada usuário dependem do seu papel no sistema. Três papéis são disponíveis inicialmente, sendo que outros papéis podem ser adicionados posteriormente: administradores, professores e alunos. Além disso, para cada um desses papéis já existe uma série de permissões que lhe são associadas por padrão, a qual também pode ser alterada posteriormente.

[18] Repositório paginate-anything - https://github.com/begriffs/angular-paginate-anything

Cada permissão representa uma funcionalidade diferente do sistema. Elas definem contexto do usuário, ou seja, os recursos aos quais ele terá acesso. Por exemplo, um administrador possui acesso a todas as áreas do sistema, logo todas as opções podem ser acessadas por ele. Um aluno possui menos permissões, enxergando portanto menos opções de ações no sistema.

Da mesma forma que a autenticação, o controle de permissões é realizado tanto no cliente quanto no servidor, garantindo que um usuário não executará uma ação cuja permissão ele não possui.

Implementou-se também o mecanismo de requerimentos, no qual um visitante do sistema poderá pedir sua inscrição, e um aluno poderá pedir sua matrícula em um curso. Em uma única página, o administrador consegue ver todos os requerimentos a serem tratados. O sistema informa por e-mail os requerentes da resposta dada pelo administrador.

4.5.7. Matérias e áreas

As matérias foram definidas e implementadas para serem correlacionadas com áreas e cursos. Focando em matérias e áreas, uma matéria é uma grande área de conhecimento. Por exemplo, pode-se dizer que Física, História e Geografia são matérias. Além disso, é possível quebrar essas disciplinas em áreas. Por exemplo, História pode ser subdividida em História Geral e História do Brasil.

4.5.8. Cursos, aulas e materiais

O administrador tem acesso a uma área para criar cursos, escolhendo seus participantes e outra para tratar requerimentos de matrícula. O professor pode criar materiais para seus cursos (conteúdos multimídia, pelo *upload* de arquivos ou indicação de URL), além de publicar novidades que são notificadas a todos os participantes. Pode também organizar um curso em aulas, dedicando materiais para

cada aula. Os alunos podem acessar os materiais e exercícios criados pelo professor, organizado por aulas.

4.5.9. Questões, exames, recomendações e *ratings*

As questões foram implementadas em dois tipos diferentes: questões de múltipla escolha e questões dissertativas. Cada tipo ativa um comportamento diferente do sistema. As questões de múltiplas escolhas permitem que o usuário que está criando adicione quantas alternativas quiser, através de um controle por *checkboxes* do estilo *slider*. Já as questões dissertativas permitem que o usuário, que responde às questões, digite uma resposta por meio de um editor de texto rico (ou *Rich Text*), ou seja, permite que figuras, *links* e vídeos sejam colocados dentro da própria reposta. O texto da questão também pode ser em formato de texto rico nos dois casos.

Os alunos podem gerar *ratings* de questões, isto é, classificar os exercícios como difíceis, médias ou fáceis, de forma que tanto o professor quanto os demais alunos possam ver a classificação geral da dificuldade do exercício.

Os exames constituem uma agregação de questões. O professor pode escolher, dentro do banco de dados de questões do Gabaritei, algumas questões, agrupá-las e disponibilizar para seus alunos. Este tipo de agregação (chamado de exame) tem o objetivo de ser um mecanismo de estudo que auxilie na consolidação do conhecimento e não uma prova, ou seja, não tem valor de nota para os alunos.

Um aluno ou um professor pode fazer uma recomendação para outro aluno, indicando um material ou um exercício. Assim, cada aluno pode ver todos os materiais e exercícios que lhe foram recomendados.

4.6. Avaliação do sistema

Estando finalizada a implementação do sistema, chegou-se a uma base sólida, pronta para ser expandida por outros desenvolvedores. Por base sólida entendemos que as principais funcionalidades básicas do sistema: *server-side* com implementação

robusta e flexível de banco de dados, sistema de autenticação e gerenciamento de conteúdo mídia, *client-side* com uma infraestrutura robusta e flexível, podendo ser personalizada com temas, traduzida facilmente para outros idiomas, tecnologias de testes implementadas e funcionais.

Por outro lado, analisou-se que a tarefa de levantamento de requisitos feita à distância, bem como a dificuldade inicial dos participantes com as tecnologias utilizadas atrasou o cronograma de desenvolvimento, de forma que não conseguiu-se entregar todas as funcionalidades descritas nos casos de uso. Apesar dessa deficiência, reforça-se que o trabalho mais árduo e demorado, que é a modelagem da arquitetura do sistema, o aprendizado de novas tecnologias e a implementação de uma base genérica e flexível foi feita, de forma que conseguiu-se alinhas as expectativas iniciais do projeto com o que foi entregue.

Criou-se um conjunto de dados fictícios para a realização da avaliação do sistema. Estes dados são apresentados no Apêndice G. Com a inserção desses dados no banco de dados, pudemos verificar que as funcionalidades implementadas correspondem ao que se era esperado. A interface mostrou-se simples, intuitiva e esteticamente agradável, os dados e informações puderam ser processados com rapidez e a implementação permitiu as interações básicas previstas entre usuários e sistema (ex. cadastro de usuários, cadastro de perguntas, criação de resposta a exercícios, manipulação de matérias, aulas e salas, entre outros).

5. CONSIDERAÇÕES FINAIS

5.1. Conclusões

Concluiu-se que a área escolhida para o projeto de formatura é bastante aberta e que ainda enfrenta grandes desafios. O desenvolvimento de sistemas educacionais ainda esbarra em grandes dificuldades, como fragmentação (diversos sistemas que possuem recursos semelhantes) e dificuldade na interface homem-computador.

Os sistemas existentes padecem de uma interface mais robusta e amigável. A equipe de projeto de formatura julga essencial que uma interface que facilita o uso por pessoas, muitas vezes leigas, no que se trata de ferramentas computacionais. Dessa forma, buscou-se, ao máximo, estudar a melhor maneira de construir uma interface simples de ser utilizada mas que, ao mesmo tempo, fosse rápida e flexível.

Além disso, buscou-se utilizar tecnologias mais recentes, por dois motivos principais. Primeiro porque elas permite construir interfaces mais ricas e responsíveis, compatível com ferramentas de ponta da Internet. Além disso, essas novas tecnologias, em geral, são mais simples de serem utilizadas e mantidas, e garantem que o projeto tenha um horizonte de vida longa.

Uma diretriz adotada foi que não desenvolver componentes e módulos não essenciais ao sistema e que já têm uma solução ótima já desenvolvida. Ou seja, não buscamos desenvolver ferramentas como calendários ou clientes de e-mail e *chat*. Dessa forma, deixou-se de implementar componentes internos que dificilmente seriam utilizados, pois essas ferramentas já possuem ótimas soluções gratuitas na Internet; e pode-se focar em otimizar o principal uso do sistema, que é o mecanismo de perguntas e respostas, além de melhorar a usabilidade com um *design* mais limpo.

Soma-se a isso o fato de ter-se superado o grande desafio de coordenar o desenvolvimento de software à distância, ou seja, mesmo durante o período em que os integrantes do grupo encontravam-se em países diferentes, foi possível organizar-se e seguir o desenvolvimento. Entre os desafios, pode-se citar o alinhamento entre os membros do projeto de formatura, evidentemente devido à distância e à diferença de fuso horário, que foi superado por meio de reuniões semanais via Skype. Foi constatado que o desenvolvimento e coordenação de software à distância é viável com o uso das ferramentas disponíveis gratuitamente na internet.

5.2. Contribuições

As contribuições do projeto de formatura Gabaritei foi o desenvolvimento de um *Framework* para ensino fundamental, médio e cursinhos. Ele é flexível o suficiente para pode ser utilizado em outras áreas além das citadas, como por exemplo, uma escola de inglês.

A moderna interface o destaca dos principais sistemas existentes. O foco na construção da interface do usuário foi a tornar mais dinâmica e simples possível, de forma que qualquer usuário consiga navegar através do site. Utilizando tecnologias que permitem o desenho de uma interface dinâmica, o Gabaritei permite a interação com o sistema através de arrastar e soltar, quadros dinâmicos que suportam conteúdo do externos como Google Docs, YouTube, pré-visualização de documentos PDF, internacionalização (suporte a múltiplos idiomas).

O *design flat* adapta-se aos padrões modernos de desenho de interface, utilizado pelos mais importantes sistemas atualmente, como o iOS, Windows 10 e o Material Design do Android. O último destaque é dado ao *design responsive* que permite com que o gabaritei se adapte a telas de celulares e *tablets.*

O Gabaritei pode ser instalado através do instalador Gabaritei que não exige nenhum tipo de configuração adicional. Basta-se executar o arquivo e esperar que a instalação seja concluída, de maneira muito mais rápida e fácil que os demais sistemas, como o Sakai e o Moodle.

Por fim, é completamente personalizável, e através do CSS compilável SASS é possível trocar cores e imagens com facilidade. O sistema deve se adaptar às cores da instituição que está utilizando sem obrigar o usuário a editar os arquivos de CSS padrão do Gabaritei.

5.3. Trabalhos futuros

Como trabalhos futuros tem-se o desenvolvimento de um maior número de testes automatizados de unidade, para conseguir chegar a uma cobertura de código maior que a atual. Além disso, preparou-se a base para o desenvolvimento de testes de integração, utilizando ferramentas como o Selenium, para garantir a integração e funcionamento dos módulos de maneira conjunta.

Espera-se que seja desenvolvido o módulo de B.I., para que os professores consigam monitorar o progresso e o desempenho de seus alunos. Além disso, pode comparar o desempenho de turmas diferentes, de forma que consiga ele próprio melhorar o seu desempenho como professor.

Além disso, deseja-se que, futuramente, o sistema seja integrado com outros recursos, como Google Calendar, para que suas funcionalidades sejam expandidas sem aumento da complexidade da solução.

6. REFERÊNCIAS

Alves, \L., Barros, D. & Okada, A., 2009. *Moode: Estratégias Pedagógicas e Estudos de Casos.* s.l., EDUNEB.

AngularJS, 2015. *Developer Guide - Unit Testing.* [Online] Available at: https://docs.angularjs.org/guide/unit-testing [Acesso em 25 11 2015].

Beck, K., 2003. *Test Driven Development: By Example.* s.l.:Addison-Wesley.

Blackboard, 2015. *Blackboard - Clientes.* [Online] Available at: http://blackboard.grupoa.com.br/clientes/ [Acesso em 20 11 2015].

Blackboard, 2015. *Sobre a Bb.* [Online] Available at: http://blackboard.grupoa.com.br/sobre/sobre-a-bb/ [Acesso em 5 11 2015].

Chaffee, A., 2012. *Repositório com notas sobre Ruby.* [Online] Available at: https://github.com/alexch/ruby_notes/blob/master/ruby-intro.md [Acesso em 18 9 2015].

edutechnica, 2014. *Number of institutions running each Learning Management Systems (LMS) by FTE.* [Online] Available at: http://edutechnica.com/2014/02/10/lmss-of-smaller-colleges/ [Acesso em 25 9 2015].

Eis, D., 2011. *Introdução ao Responsive Web Design.* [Online] Available at: http://tableless.com.br/introducao-ao-responsive-web-design/ [Acesso em 18 9 2015].

GitHub, 2015. *GitHub Help Page - Fork a Repo.* [Online] Available at: https://help.github.com/articles/fork-a-repo/ [Acesso em 20 9 2015].

GitHub, 2015. *GitHub Help Page - Using pull requests.* [Online] Available at: https://help.github.com/articles/using-pull-requests/ [Acesso em 20 9 2015].

Google, 2014. *Built with AngularJS.* [Online] Available at: https://builtwith.angularjs.org/ [Acesso em 19 9 2015].

Green, M. & Pai, A., 2015. *AngularJS Tutorial: Learn to Build Modern Web Apps with Angular and Rails.* [Online]
Available at: https://thinkster.io/angular-rails#introduction
[Acesso em 20 11 2015].

Grosenbach, G., 2010. *What pythonistas think of ruby.* [Online]
Available at: http://blog.pluralsight.com/what-pythonistas-think-of-ruby
[Acesso em 18 9 2015].

Jasmine, 2015. *Jasmine.* [Online]
Available at: http://jasmine.github.io/edge/introduction.html
[Acesso em 18 9 2015].

Karma, 2015. *Karma - Config.* [Online]
Available at: http://karma-runner.github.io/0.8/config/
[Acesso em 18 9 2015].

Karma-Runner, 2015. *Repositório do Plugin karma-ng-html2js-preprocessor.* [Online]
Available at: https://github.com/karma-runner/karma-ng-html2js-preprocessor
[Acesso em 18 9 2015].

Lamb, D., 2014. *jQuery vs. AngularJS: A Comparison and Migration Walkthrough.* [Online]
Available at: https://www.airpair.com/angularjs/posts/jquery-angularjs-comparison-migration-walkthrough
[Acesso em 25 11 2015].

Mathieson, S., 2015. *Repositório Grunt Task para JSLint.* [Online]
Available at: https://github.com/stephenmathieson/grunt-jslint
[Acesso em 18 9 2015].

moodle, 2015. *Registered Moodle sites.* [Online]
Available at: https://moodle.net/sites/
[Acesso em 18 09 2015].

mr7, 2015. *Repositório do Grunt Task.* [Online]
Available at: https://github.com/m7r/grunt-ngdocs
[Acesso em 18 9 2015].

Otto, M. & Thornton, J., 2015. *Bootstrap - CSS - Responsive Utilities.* [Online]
Available at: http://getbootstrap.com/css/#responsive-utilities
[Acesso em 18 9 2015].

Palmer, J., 2011. *Repositório AngularJS - License.* [Online] Available at: https://github.com/angular/angular.js/blob/master/LICENSE [Acesso em 18 9 2015].

Plataforma Tec, 2015. *Repositório do devise.* [Online] Available at: https://github.com/plataformatec/devise [Acesso em 20 11 2015].

Reenskaug, T., 2007. *The original MVC reports.* [Online] Available at: http://heim.ifi.uio.no/~trygver/2007/MVC_Originals.pdf. [Acesso em 18 9 2015].

RubyGems.org, 2009. *RubyGems.* [Online] Available at: https://rubygems.org/ [Acesso em 18 9 2015].

Scrum.org, 2014. *Scrum Guides.* [Online] Available at: http://scrumguides.org/ [Acesso em 18 09 2015].

Scrum.org, 2015. *Scrum.* [Online] Available at: https://www.scrum.org/About [Acesso em 18 09 2015].

APÊNDICE A: *Product Backlog*

Os requisitos classificados, enumerados e pontuados abaixo foram levantados com base em discussões com potenciais clientes do *framework* desenvolvido. Cada requisito recebe um valor agregado (*business value*) e um valor de complexidade (*user point*). A razão desses dois valores fornece o ROI (*return on investment*).

1. Exercícios

1.1. Sou um professor e desejo fazer upload de um novo exercício, segmentado por salas

- Business value: 100
- User point: 40
- ROI: 2,5

1.2. Sou um professor e desejo ter a opção de disponibilizar ou não a resolução de um exercício

- Business value: 50
- User point: 2
- ROI: 25

1.3. Sou um professor e desejo marcar um exercício como “quente”, ou seja, altamente indicado

- Business value: 100
- User point: 2
- ROI: 50

1.4. Sou um aluno e desejo resolver exercícios extras

- Business value: 100
- User point: 20
- ROI: 50

1.5. Sou um aluno e desejo verificar a solução de um exercício na internet

- Business value: 100
- User point: 8
- ROI: 12,5

1.6. Sou um aluno e desejo procurar exercícios nos exercícios disponíveis da minha sala

- Business value: 100

- User point: 13
- ROI: 7,69

1.7. Sou um aluno e desejo ver o material disponível separado por área de conhecimento

- Business value: 50
- User point: 8
- ROI: 6,25

1.8. Sou um aluno e desejo marcar um exercício como fácil, intermediário ou difícil

- Business value: 20
- User point: 8
- ROI: 2,5

2. Material

2.1. Sou um professor e desejo disponibilizar o material na forma de download para os alunos

- Business value: 100
- User point: 5
- ROI: 20

2.2. Sou um professor e desejo disponibilizar o material na forma de visualização para os alunos, impedindo o download de material protegido

- Business value: 20
- User point: 100
- ROI: 0,2

2.3. Sou um aluno e desejo ver o material disponível para mim separado por área de conhecimento

- Business value: 100
- User point: 8
- ROI: 12,5

2.4. Sou um aluno e desejo ver o material disponibilizado por um professor

- Business value: 100
- User point: 8
- ROI: 12,5

2.5. Sou um aluno e desejo assistir a um vídeo disponibilizado por um professor

- Business value: 100
- User point: 20
- ROI: 5

3. Controle de conteúdo

3.1. Sou um administrador e desejo manter o conteúdo de cada sala restrito aos respectivos alunos

- Business value: 100
- User point: 20
- ROI: 5

4. Salas

4.1. Como administrador, desejo poder criar uma sala de aula virtual

- Business value: 100
- User point: 8
- ROI: 12,5

4.2. Como administrador, desejo poder adicionar alunos e professores a uma sala da aula virtual

- Business value: 100
- User point: 8
- ROI: 12,5

4.3. Como aluno, desejo poder pedir para me cadastrar em uma sala de aula, caso não tenha sido cadastrado previamente

- Business value: 100
- User point: 8
- ROI: 12,5

5. Novidades

5.1. Como professor, desejo enviar um e-mail para todos os alunos avisando sobre novo material disponível

- Business value: 20
- User point: 8
- ROI: 2,5

5.2. Como aluno, desejo poder visualizar as novidades em cada sala de aula de forma rápida

- Business value: 50
- User point: 8

- ROI: 6,25

6. Home page

6.1. Como professor, desejo acesso rápido ao meu conteúdo

- Business value: 20
- User point: 8
- ROI: 2,5

6.2. Como professor, desejo ver qual é o meu material mais acessado (se possível implementar gráfico)

- Business value: 50
- User point: 8
- ROI: 6,25

6.3. Como aluno, desejo saber quais os exercícios mais indicados para mim

- Business value: 20
- User point: 100
- ROI: 0,2

6.4. Como aluno, desejo ver de forma rápida os materiais mais indicados para mim

- Business value: 20
- User point: 100
- ROI: 0,2

6.5. Como administrador, desejo facilidade no cadastro de salas e envio de permissão de cadastro

- Business value: 100
- User point: 13
- ROI: 7,69

7. Perfil

7.1. Como professor, desejo adicionar minha imagem, minha formação e aonde dou aula

- Business value: 50
- User point: 5
- ROI: 10

7.2. Como aluno, desejo adicionar minha imagem e quais as áreas que eu tenho mais dificuldade

- Business value: 50
- User point: 5
- ROI: 10

8. Administração

8.1. Como administrador, desejo poder cadastrar alunos e professores no sistema

- Business value: 100
- User point: 5
- ROI: 20

8.2. Como administrador, desejo poder excluir usuários do sistema

- Business value: 100
- User point: 5
- ROI: 20

9. Visitantes

9.1. Como visitante, desejo poder visualizar informações sobre a escola / o cursinho no site

- Business value: 100
- User point: 1
- ROI: 100

9.2. Como visitante, desejo poder entrar em contato com o cursinho

- Business value: 100
- User point: 2
- ROI: 50

10. Relatório de desempenho

10.1. Como professor, desejo poder visualizar um relatório sobre o desempenho geral dos alunos em um determinado exercício

- Business value: 50
- User point: 40
- ROI: 1,25

10.2. Como professor, desejo poder visualizar um relatório sobre o desempenho dos alunos em determinado exame

- Business value: 50
- User point: 40
- ROI: 1,25

10.3. Como aluno, desejo visualizar a evolução do meu desempenho em exercícios em cada área

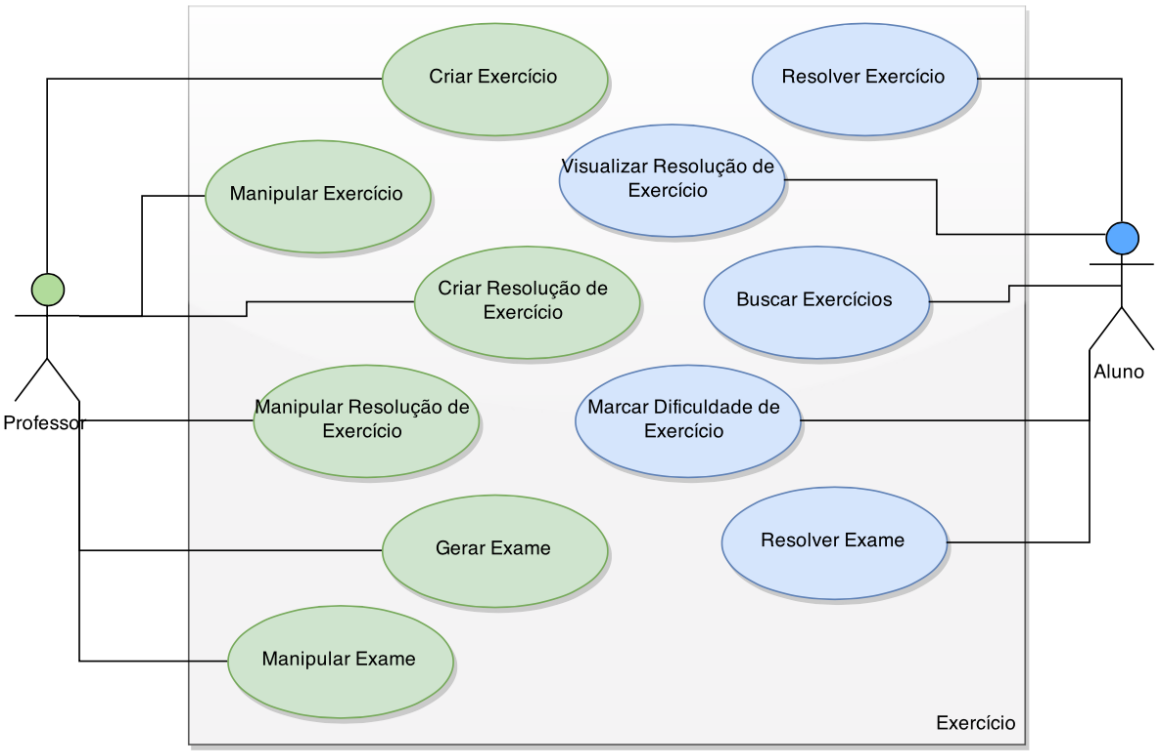
- Business value: 50
- User point: 40
- ROI: 1,25

10.4. Como aluno, desejo visualizar meu histórico de desempenho em simulados

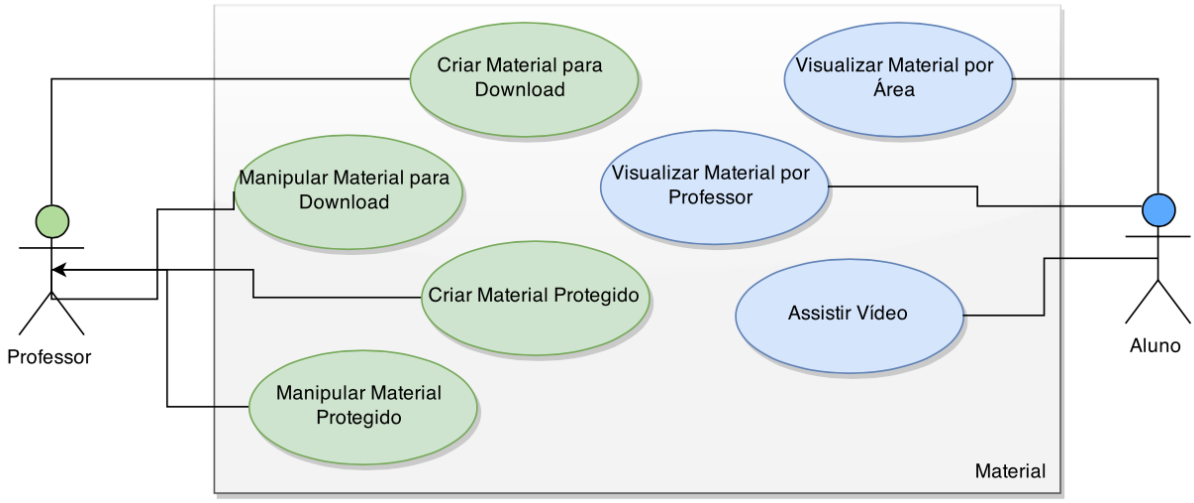
- Business value: 50
- User point: 40
- ROI: 1,25

APÊNDICE B: Diagramas de casos de uso

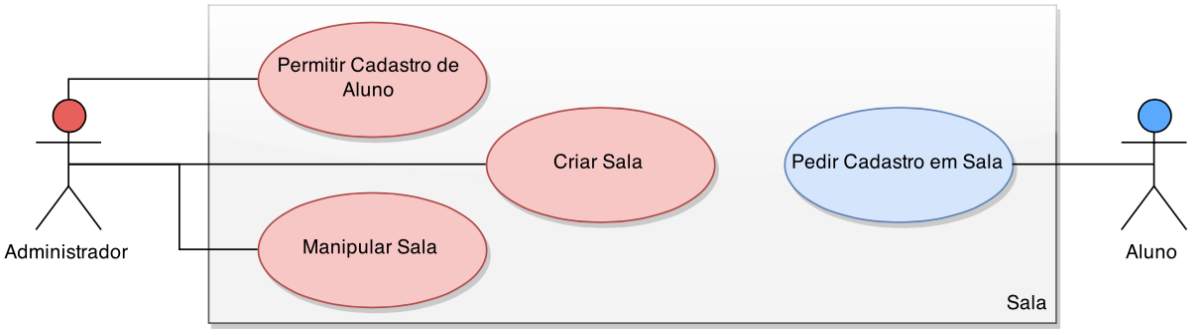
1. Exercícios



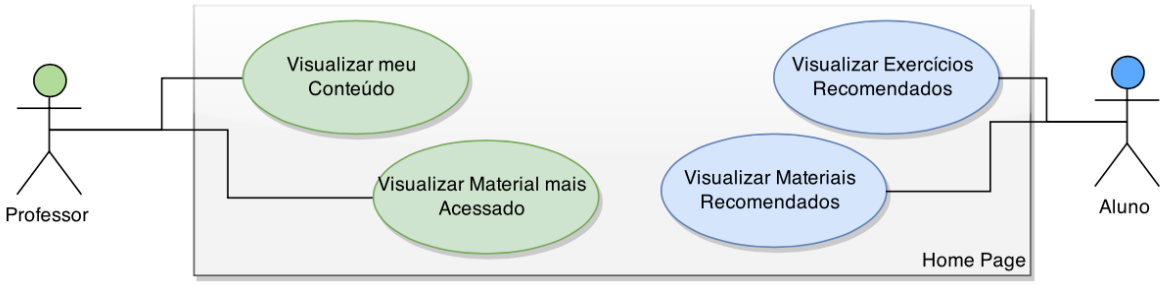
2. Material



3. Sala

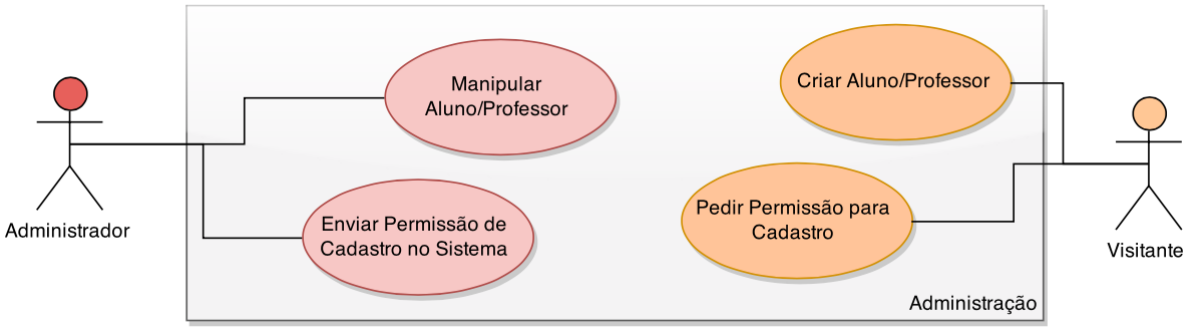


4. Home Page

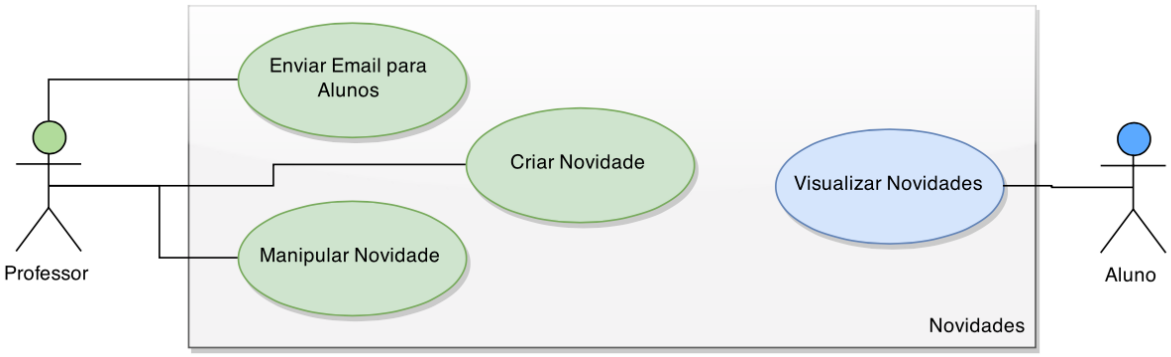


5. Perfil

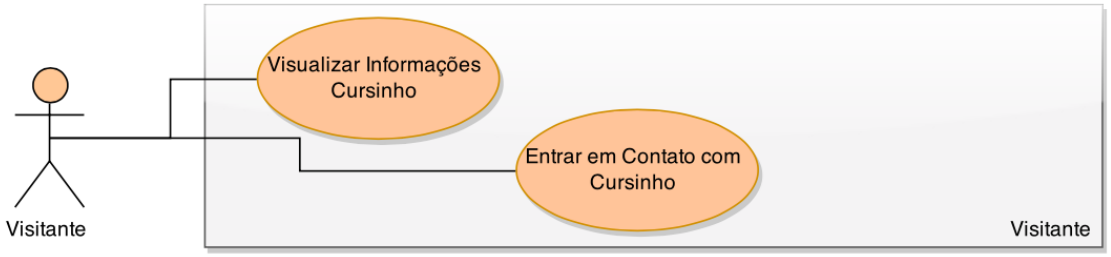
6. Administração



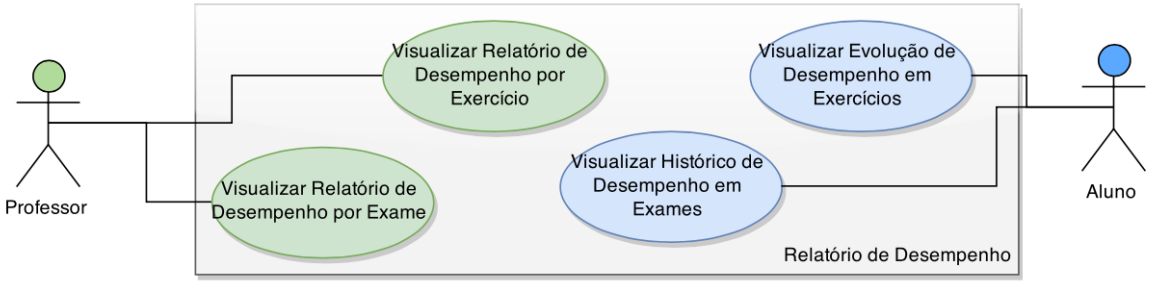
7. Novidades



8. Visitante



9. Relatório de Desempenho



APÊNDICE C: Detalhamento dos casos de uso

Casos de uso genéricos 1: CRUD (Create, Read, Update, Delete)

Nesse sistema, o CRUD é composto por dois casos de uso: Criar e Manipular. A manipulação inclui editar, atualizar e excluir.

O CRUD para a Resolução de Exercícios, Material para Download, Material Protegido, Novidades, Sala, Perfil e Aluno/Professor é análogo ao de Exercícios, detalhamento explicitado a seguir, porém com as mudanças necessárias relacionadas aos campos de preenchimento (especificamente no passo 3) e também nos seus respectivos atores.

Criar e Manipular Sala também incluem adicionar/remover Alunos e Professores. Importante mencionar que Criar Aluno/Professor tem como ator um Visitante que recebeu a chave de acesso do Administrador (Permissão de Cadastro), e por outro lado, Manipular Aluno/Professor é um caso de uso do ator Administrador.

O próprio Aluno/Professor altera seu respectivo perfil apenas, com o caso de uso Manipular Perfil.

Caso de uso 1.1: Criar Exercício

Descrição: permite a um professor, anteriormente autenticado no sistema, adicionar um exercício ao sistema.

Atores: Professor.

Evento iniciador: ator seleciona “Adicionar Exercício” no menu do sistema.

Pré--condição: ator previamente autenticado no sistema.

Sequência de eventos:

1. O ator clica em “Adicionar exercício”;
2. O sistema carrega a tela de criação de um novo exercício;
3. O ator preenche o formulário com um identificador da questão (“nome”), identificando da onde foi extraída a questão e o ano da questão (“Fuvest -96”). Seleciona, através de um menu *dropdown*, se a questão é discursiva ou testes. Disponibiliza as alternativas no caso de questão de testes. Seleciona a área de conhecimento (Física, Matemática, Geografia, Português, etc), podendo selecionar mais de uma opção (questões interdisciplinares). Seleciona se a questão é quente ou não.

4. O ator clica no botão "Prévia";
5. O sistema verifica os campos;
6. O sistema gera uma prévia da página do exercício e exibe ao usuário.
7. O ator confirma as informações contidas e clica no botão “Criar”;
8. O sistema exibe uma mensagem de criação do exercício, o armazena e redireciona o usuário a página do exercício.

Pós--condições: Exercício cadastrado no sistema.

Extensões:

1. O ator clica em “Cancelar” para anular a criação do exercício;
    1. O sistema exibe uma caixa de confirmação da operação;
    2. Confirmado, o sistema cancela a criação do exercício e leva o professor é levado até a tela de listagem de exercícios.
2. Caso os campos não tenham sido preenchidos corretamente (passo 5):
    1. O sistema exibe uma mensagem de erro;
    2. Fluxo segue normal do passo 2.

Inclusões:

1. Autenticação dos dados, ou seja, a verificação se os campos preenchidos estão válidos e se todos os campos obrigatórios foram preenchidos (passo 5).

Caso de uso 1.2: Manipular Exercício

Descrição: a manipulação de exercícios inclui editar, atualizar e apagar o exercício cadastrado.

Atores: Professor.

Evento iniciador: ator seleciona “Editar Exercício” no menu do sistema.

Pré-condição: ator previamente autenticado no sistema e exercício cadastrado no sistema.

Sequência de eventos:

1. O ator clica em “Editar exercício”;
2. O sistema carrega a tela de edição de um exercício cadastrado no sistema. Essa tela contém os campos: “nome”, “tipo”, “área de conhecimento”, “quente”.
3. O ator altera os campos desejados;
4. O ator clica no botão "Prévia";
5. O sistema verifica os campos;

6. O sistema gera uma prévia da página do exercício e exibe ao usuário.
7. O ator confirma as informações contidas e clica no botão “Editar”;
8. O sistema exibe uma mensagem de edição do exercício, o armazena e redireciona o usuário a página do exercício.

Pós-condições: Exercício atualizado no sistema.

Extensões:

1. Caso os campos não tenham sido preenchidos corretamente (passo 5):
    1. O sistema exibe uma mensagem de erro;
    2. Fluxo segue normal do passo 2.
2. O tipo do exercício é alterado de alternativas para discursivo:
    1. O sistema remove o campo de alternativas;
    2. Fluxo segue normal do passo 4.
3. O tipo de exercício é alterado de discursivo para alternativas:
    1. O sistema adiciona o campo de alternativas;
    2. Fluxo segue normal do passo 4.
4. O ator clica em “Cancelar” para anular a criação do exercício:
    1. O sistema exibe uma caixa de confirmação da operação;
    2. Confirmado, o sistema cancela a edição do exercício e leva o ator até a tela de listagem de exercícios.
5. Caso desejado, o ator pode excluir o cadastro do exercício:
    1. O ator aperta o botão “Excluir Cadastro” ao invés de apertar o botão “Prévia” (passo 4);
    2. Confirmado, o sistema exclui o exercício e leva o ator até a tela de listagem de exercícios.

Inclusões:

1. Autenticação dos dados, ou seja, a verificação se os campos preenchidos estão válidos e se todos os campos obrigatórios foram preenchidos (passo 5).

Caso de uso genérico 2: Visualização

A visualização de Material mais Acessado, Informações do Cursinho, Relatório de Desempenho por Exercício, Relatório de Desempenho por Exame, Evolução de Desempenho em Exercícios e Histórico de Desempenho em Exames é análoga ao de Resolução de Exercícios, com as mudanças relacionadas a quais informações sobre

o objeto desejado aparecem na tela. A visualização de Material por Área, Material por Professor, Novidades, meu Conteúdo, Exercícios Recomendados e Materiais Recomendados também são análogas, mas ao invés de aparecerem informações, listagens são mostradas na tela. Assistir Vídeo é um caso de uso também análogo ao Visualizar Resolução de Exercício, porém um vídeo *player* com o vídeo desejado é aberto na tela.

Caso de uso 2.1: Visualizar Resolução de Exercício
Descrição: permite a um aluno visualizar a resolução de um exercício.
Atores: Aluno
Evento iniciador: ator seleciona o exercício que queira visualizar a resolução na listagem de exercícios.
Pré-condição: ator previamente cadastrado no sistema e exercício e respectiva resolução disponíveis nas salas em que está cadastrado.
Sequência de eventos:

1. Ator clica no exercício que deseja visualizar resolução;
2. O sistema abre uma tela com a resolução do exercício.

Pós-condições: Resolução de Exercício aparecendo na tela.

Casos de uso 3: Exercícios

Caso de uso 3.1: Resolver Exercício
Descrição: permite a um aluno resolver um exercício
Atores: Aluno
Evento iniciador: ator seleciona o exercício que queira resolver na listagem de exercícios.
Pré-condição: ator previamente cadastrado no sistema e exercício disponível nas salas em que está cadastrado.
Sequência de eventos:

1. Ator clica no exercício que deseja resolver;
2. O sistema abre uma tela de resolução do exercício;
3. O ator entra com a resposta (no caso de um exercício dissertativo) ou a seleciona (no caso de alternativas);

4. O ator clica no botão "Enviar resposta”;
5. O sistema exibe, caso disponível, a resposta do exercício;
6. O sistema registra a resposta do aluno na base dados.

Pós-condições: Exercício resolvido armazenado no sistema.

Extensões:

1. O ator, ao invés de enviar a resposta, clica em “Voltar” (passo 4):
    1. O sistema ignora a resposta do exercício da base de dados;
    2. O ator é direcionado novamente à tela de exercícios.

Caso de uso 3.2: Buscar Exercício

Descrição: permite a um aluno buscar um exercício dentro da lista exercícios disponíveis para as salas cadastradas.

Atores: Aluno

Evento iniciador: ator digita o termo que deseja buscar na lista de exercícios disponíveis e aperta botão "Buscar".

Pré-condição: ator previamente cadastrado no sistema.

Sequência de eventos:

1. Ator pressiona o botão “Buscar" com o termo desejado inserido no campo de busca;
2. O sistema processa a busca desejada;
3. O sistema abre uma tela com a listagem buscada de exercícios;

Pós-condições: Listagem de Exercícios aparecendo na tela.

Extensões:

1. Caso não haja resultado para o termo buscado, mostrar mensagem: “Não há exercícios com esse termo” (passo 3).

Caso de uso 3.3: Gerar exame

Descrição: permite a um professor, anteriormente autenticado no sistema, criar um coleção de exercícios, estes previamente cadastrados no sistema. Esta coleção de exercícios chamamos de exame.

Atores: Professor

Evento iniciador: ator seleciona “Criar Exame” no menu do sistema.

Pré-condição: professor previamente autenticado no sistema.

Sequência de eventos:

1. Ator clica em “Criar exames”;
2. O sistema carrega um formulário do sistema para criação do exame
3. O professor deverá selecionar a sala para qual será disponibilizado o exame.
4. O sistema deverá exibir os exercícios correspondentes à área do professor e correspondentes à sala selecionada
5. O professor deverá selecionar todos os exercícios que quiser incluir no exame.
6. O professor deve clicar em “Criar” e será redirecionado para uma tela de confirmação da criação
7. O professor clica em “Confirmar”
8. O exercício deverá ser criado no sistema

Pós-condições: exame cadastrado no sistema.

Extensões:

1. O professor, ao invés de confirmar, clica em “cancelar”:
    1. O sistema não deve incluir o exame na base de dados
    2. O professor deverá ser levado a tela de listagem de exames

Caso de uso 3.4: Manipular Exame

Descrição: a manipulação de exames inclui editar, atualizar e apagar uma coleção de exercícios previamente cadastrados no sistema, chamada Exame.

Atores: Professor

Evento iniciador: ator seleciona “Editar Exame” no menu do sistema.

Pré-condição: professor previamente autenticado no sistema.

Sequência de eventos:

1. Ator clica em “Editar exame”;
2. O sistema carrega um formulário do sistema para edição do exame;
3. O professor pode modificar somente os exercícios selecionados e não pode mudar a sala para qual foi disponibilizado (esse campo aparece bloqueado);
4. Ao finalizar a modificação, o professor clica em “Modificar Exame”;
5. O sistema exibe uma mensagem de confirmação da modificação;
6. O sistema salva as alterações no sistema.
7. O sistema redireciona o ator para a página de listagem de exames.

Pós-condições: Exame atualizado no sistema.

Extensões:

1. O professor, ao invés de confirmar, clica em “cancelar”:
    1. O sistema não deve incluir as modificações do exame na base de dados;
    2. O ator é levado a tela de listagem de exames.
2. Caso desejado, o ator pode excluir o cadastro do exercício:
    1. O ator aperta o botão “Excluir Cadastro” ao invés de apertar o botão “Modificar Exame” (passo 4);
    2. Confirmado, o sistema exclui o exame e leva o ator até a tela de listagem de exames.

Caso de uso 3.5: Resolver Exame

Descrição: permite a um aluno resolver um exame

Atores: Aluno

Evento iniciador: ator seleciona o exame que queira resolver na listagem de exames.

Pré-condição: ator previamente cadastrado no sistema e exame disponível nas salas em que está cadastrado.

Sequência de eventos:

1. Ator clica no exame que deseja resolver;
2. O sistema abre uma tela de resolução do exame com todos os exercícios do exame listados;
3. O ator entra com as respectivas respostas (no caso de um exercício dissertativo) ou as seleciona (no caso de alternativas);
4. O ator clica no botão "Enviar resposta”;
5. O sistema registra a resposta do aluno na base dados.

Pós-condições: Exame resolvido armazenado no sistema.

Extensões:

1. O ator, ao invés de enviar a resposta, clica em “Voltar” (passo 4):
    1. O sistema ignora a resposta do exame da base de dados;
    2. O ator é direcionado novamente à tela de exame.

Caso de uso 3.6: Marcar Dificuldade de Exercício

Descrição: permite a um aluno marcar a dificuldade de um exercício. Não depende de ter resolvido o exercício.

Atores: Aluno

Evento iniciador: ator previamente autenticado no sistema.

Sequência de eventos:

1. Ator seleciona o exercício que deseja visualizar;
2. Na tela de visualização, o usuário clica em um dos três botões disponíveis no topo direito da tela, correspondentes a fácil, médio ou difícil;
3. O sistema registra a dificuldade escolhida pelo usuário para o exercício selecionado

Pós-condições: Dificuldade do exercício segundo o ator salva no sistema.

Extensões:

1. O sistema permite que o aluno altere a dificuldade do exercício, seguindo os mesmos passos a partir do item 1.

Caso de uso 4: Sala

Caso de uso 4.1: Pedir Cadastro em Sala

Descrição: permite a um aluno pedir para ser adicionado em certa sala.

Atores: Aluno

Evento iniciador: aluno clica em solicitar cadastro na sua página inicial

Pré-condição: ator previamente autenticado no sistema e sala cadastrada.

Sequência de eventos:

1. Ator clica em "Solicitar cadastro" na tela inicial;
2. Sistema lista todas as salas disponíveis;
3. O aluno seleciona a sala que deseja pedir cadastro;
4. O sistema mostra uma modal confirmando a sala;
5. O aluno deve preencher, nesta modal, os motivos pelo quais quer fazer o cadastro na sala. Ator clicar em "Confirmar";
6. O sistema registra o pedido de cadastro.

Pós-condições: Pedido de cadastro na sala enviado ao Administrador.

Extensões:

1. O ator aperta botão "Cancelar":

1. O sistema envia mensagem cancelando o cadastro do aluno na sala;
2. O sistema redireciona ator a tela inicial.

Caso de uso 5: Novidades

Caso de uso 5.1: Enviar Email para Alunos

Descrição: permite ao professor enviar Email a todos os participantes cadastrados em certa sala.

Atores: Professor

Evento iniciador: Professor clica em enviar *e-mail* para sala.

Pré-condição: ator previamente autenticado no sistema e sala cadastrada.

Sequência de eventos:

1. Professor clica em enviar *e-mail*;
2. O sistema mostra uma página com formulário;
3. O professor preenche o título e o corpo do *e-mail*;
4. O professor clica em enviar;
5. O sistema envia o *e-mail* a todos os participantes da sala.

Pós-condições: *e-mail* enviado aos alunos cadastrados na sala.

Extensões:

1. O professor clica em cancelar (passo 4):
    1. O sistema não envia o email.

Caso de uso 6: Administração

Caso de uso 6.1: Enviar Permissão de Cadastro no Sistema

Descrição: permite ao aluno se cadastrar no sistema, utilizando uma chave enviada pelo administrador ao aluno

Atores: Administrador

Evento iniciador: O administrador clica em “Ver requisições de cadastro” na sua tela inicial.

Pré-condição: ator previamente autenticado no sistema e pedido de cadastro realizado anteriormente por um Visitante.

Sequência de eventos:

1. O administrador verifica qual dos pedidos quer aceitar e clica para selecioná-lo;
2. O sistema exibe uma tela, informando o nome e *e-mail* do aluno;
3. O administrador escreve uma mensagem para ser enviada ao aluno e clica enviar;
4. O sistema gera uma chave, e envia por *e-mail*, junto com a mensagem do administrador.

Pós-condições: Permissão de cadastro no sistema enviada ao Visitante que fez o pedido.

Extensões:

1. O administrador clica em cancelar;
2. O sistema não envia a permissão de cadastro.

Caso de uso 6.2: Pedir Permissão de Cadastro no Sistema

Descrição: permite a um Visitante realizar um pedido de cadastro no sistema para alunos

Atores: Visitante

Evento iniciador: Visitante clica em "Solicitar cadastro".

Pré-condição: não há

Sequência de eventos:

1. Visitante preenche o formulário de cadastro com: nome, *e-mail*, série, sala, professor, data de nascimento e clica em "Prévia";
2. O sistema exibe os dados para confirmação;
3. Ao confirmar, o sistema salva os dados e o sistema exibe uma mensagem de sucesso.

Pós-condições: Pedido de permissão de cadastro enviado ao Administrador.

Extensões:

1. O visitante clica em cancelar e não conclui a operação.

Caso de uso 7: Visitante

Caso de uso 7.1: Entrar em Contato com Cursinho

Descrição: permite a um visitante entrar em contato com o cursinho através de um formulário.

Atores: Visitante

Evento iniciador: ator pressiona botão "Entrar em contato com Cursinho".

Pré-condição: nenhuma.

Sequência de eventos:

1. Ator clica no botão "Entrar em contato com Cursinho";
2. O sistema abre uma tela de formulário de contato;
3. O ator preenche os campos obrigatórios e pressiona botão "Enviar Mensagem";
4. O sistema verifica os campos;
5. O sistema redireciona o ator a uma página com mensagem de envio de mensagem.

Pós-condições: Resolução de Exercício aparecendo na tela.

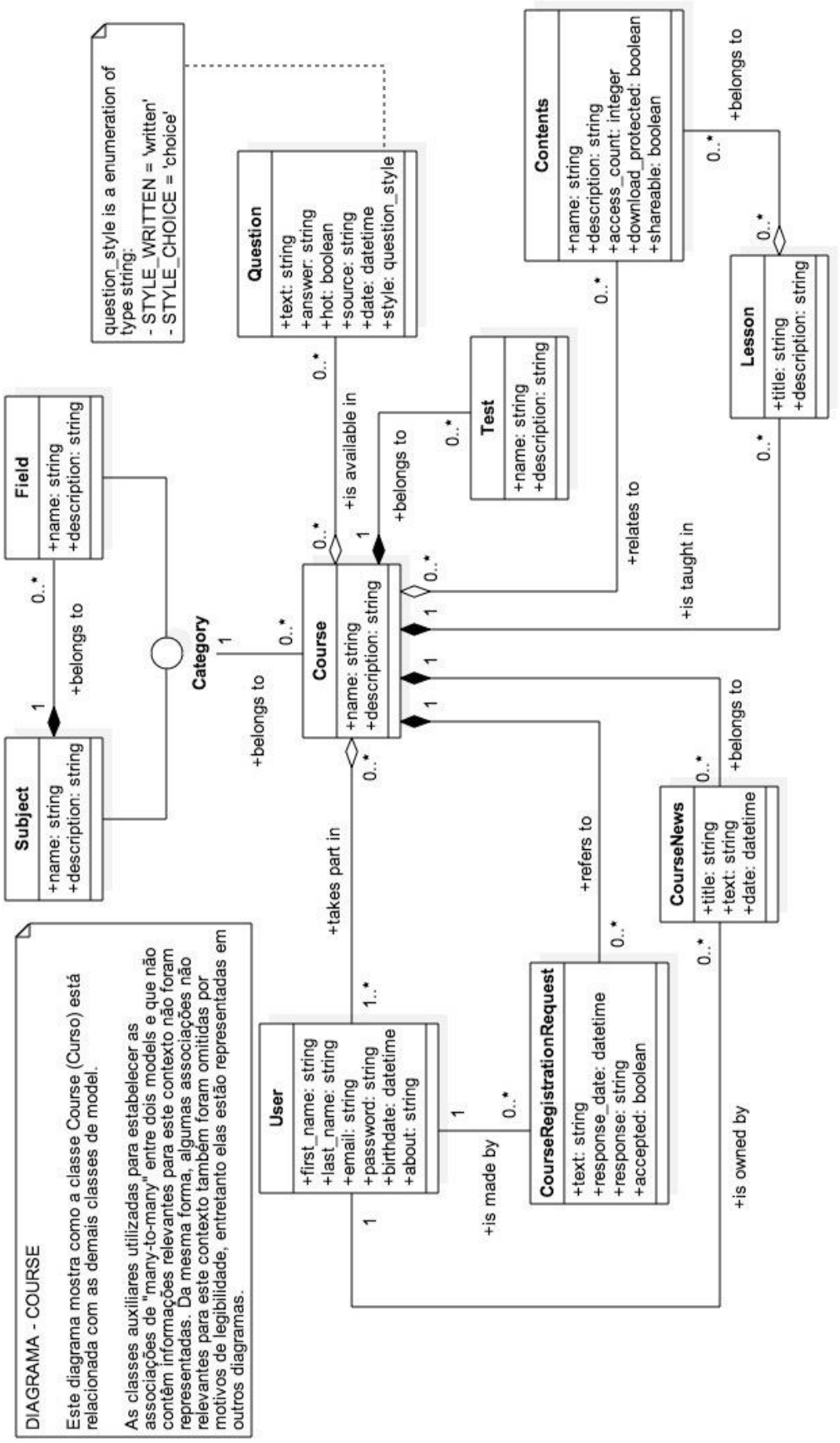
Extensões:

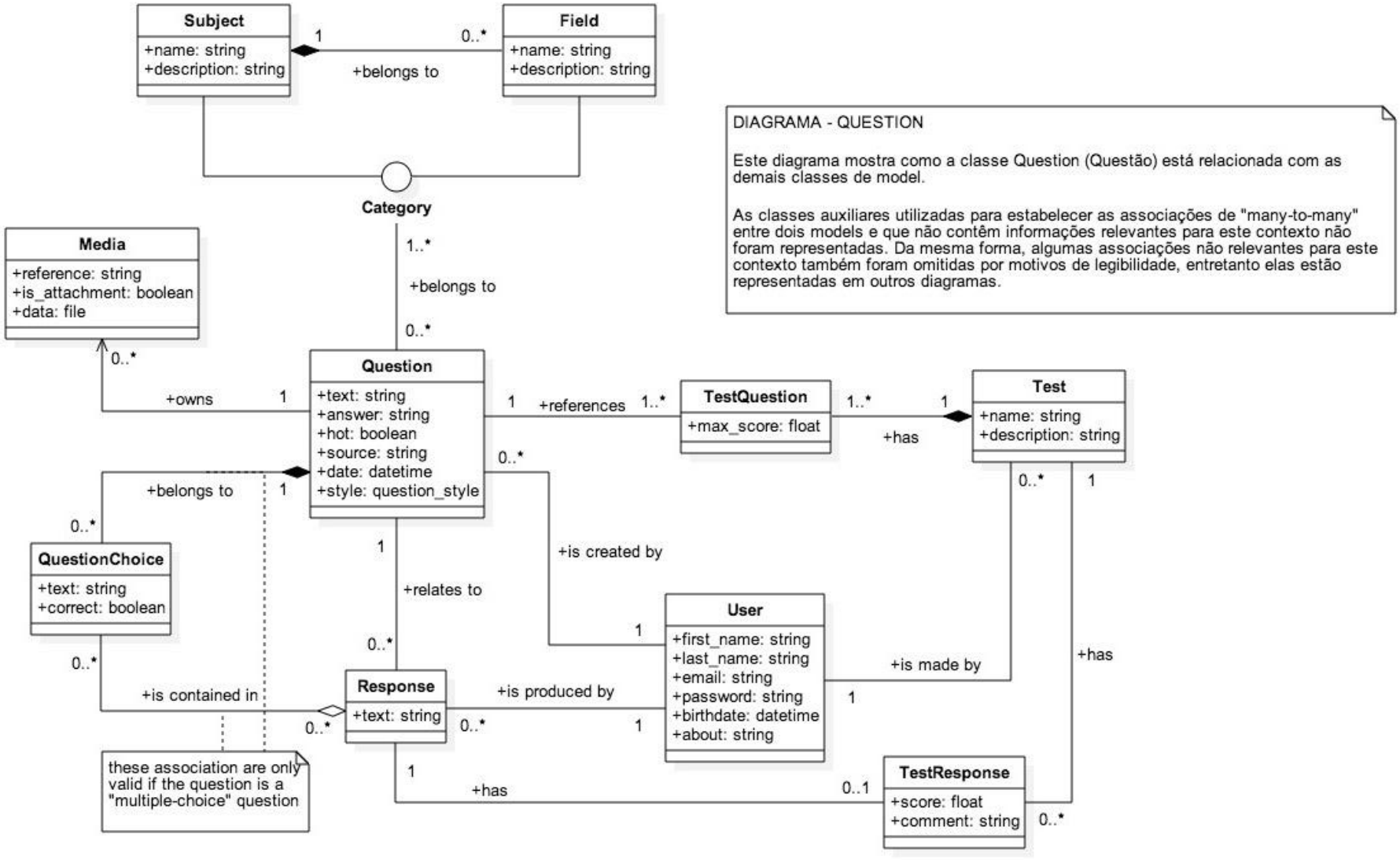
1. Caso os campos não tenham sido preenchidos corretamente (passo 4):
    1. O sistema exibe uma mensagem de erro;
    2. Fluxo segue normal do passo 2.
2. O ator clica em “Cancelar” para anular o envio de mensagem:
    1. O sistema exibe uma caixa de confirmação da operação;
    2. Confirmado, o sistema leva o ator até a tela inicial.

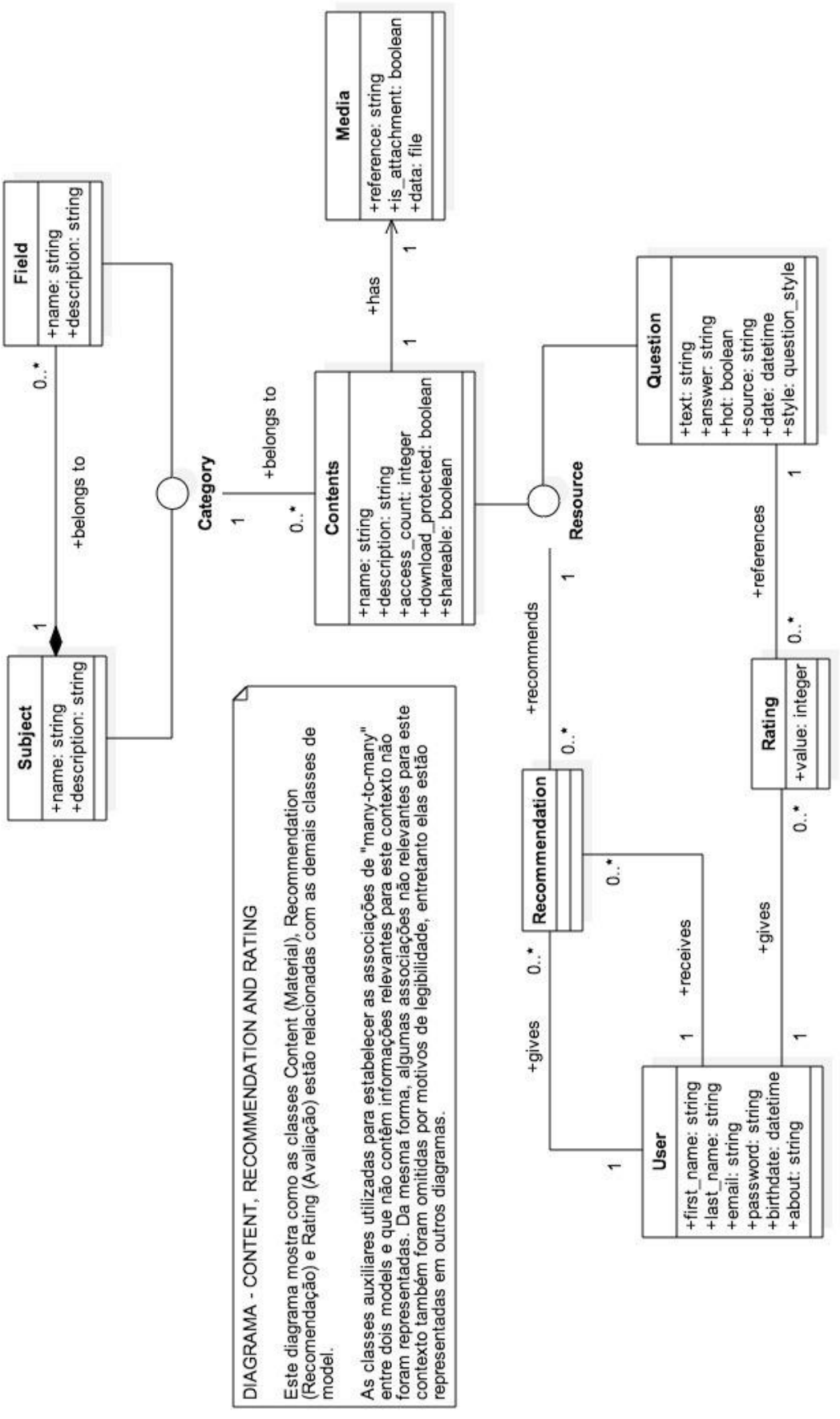
Inclusões:

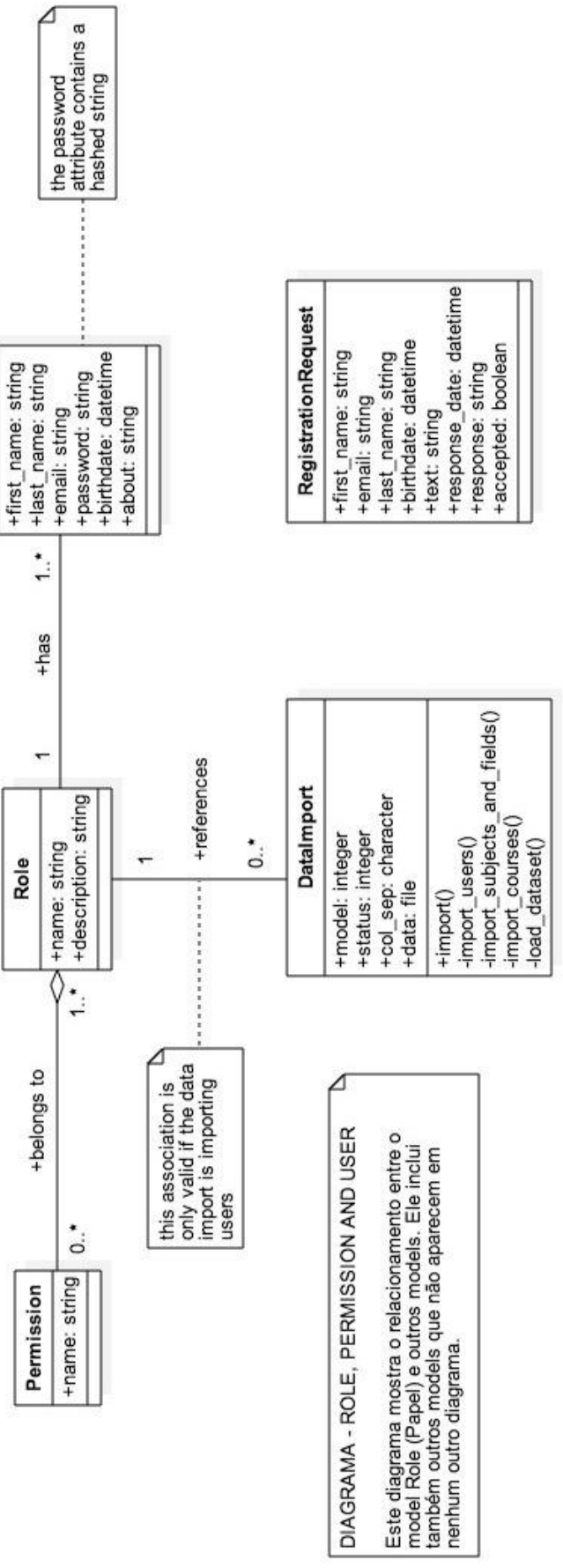
1. Autenticação dos dados, ou seja, a verificação se os campos preenchidos estão válidos e se todos os campos obrigatórios foram preenchidos (passo 4).

APÊNDICE D: Diagramas de classes




Subject
+name: string
+description: string
1
0..*
+belongs to
Field
+name: string
+description: string
Category
DIAGRAMA - QUESTION
Este diagrama mostra como a classe Question (Questão) está relacionada com as demais classes de model.
As classes auxiliares utilizadas para estabelecer as associações de "many-to-many" entre dois models e que não contêm informações relevantes para este contexto não foram representadas. Da mesma forma, algumas associações não relevantes para este contexto também foram omitidas por motivos de legibilidade, entretanto elas estão representadas em outros diagramas.
Media
+reference: string
+is_attachment: boolean
+data: file
1..*
+belongs to
0..*
0..*
+owns
1
Question
+text: string
+answer: string
+hot: boolean
+source: string
+date: datetime
+style: question_style
1
+references
1..*
TestQuestion
+max_score: float
1..*
+has
1
Test
+name: string
+description: string
0..*
1
+belongs to
1
0..*
QuestionChoice
+text: string
+correct: boolean
1
+relates to
0..*
0..*
+is contained in
0..*
Response
+text: string
+is created by
1
User
+first_name: string
+last_name: string
+email: string
+password: string
+birthdate: datetime
+about: string
+is made by
1
+has
+is produced by
0..*
1
these association are only valid if the question is a "multiple-choice" question
1
+has
0..1
TestResponse
+score: float
+comment: string
0..*


Field
+name: string
+description: string
0..*
+belongs to
1
Subject
+name: string
+description: string
Category
1
+belongs to
0..*
Contents
+name: string
+description: string
+access_count: integer
+download_protected: boolean
+shareable: boolean
+has
1
1
Media
+reference: string
+is_attachment: boolean
+data: file
Resource
1
+recommends
0..*
Recommendation
0..*
+gives
0..*
1
User
+first_name: string
+last_name: string
+email: string
+password: string
+birthdate: datetime
+about: string
1
+receives
0..*
+gives
1
0..*
Rating
+value: integer
+references
0..*
1
Question
+text: string
+answer: string
+hot: boolean
+source: string
+date: datetime
+style: question_style
DIAGRAMA - CONTENT, RECOMMENDATION AND RATING
Este diagrama mostra como as classes Content (Material), Recommendation (Recomendação) e Rating (Avaliação) estão relacionadas com as demais classes de model.
As classes auxiliares utilizadas para estabelecer as associações de "many-to-many" entre dois models e que não contêm informações relevantes para este contexto não foram representadas. Da mesma forma, algumas associações não relevantes para este contexto também foram omitidas por motivos de legibilidade, entretanto elas estão representadas em outros diagramas.


User
+first_name: string
+last_name: string
+email: string
+password: string
+birthdate: datetime
+about: string
the password attribute contains a hashed string
RegistrationRequest
+first_name: string
+email: string
+last_name: string
+birthdate: datetime
+text: string
+response_date: datetime
+response: string
+accepted: boolean
+has
1..*
1
Role
+name: string
+description: string
1
+references
0..*
DataImport
+model: integer
+status: integer
+col_sep: character
+data: file
+import()
-import_users()
-import_subjects_and_fields()
-import_courses()
-load_dataset()
+belongs to
1..*
0..*
Permission
+name: string
this association is only valid if the data import is importing users
DIAGRAMA - ROLE, PERMISSION AND USER
Este diagrama mostra o relacionamento entre o model Role (Papel) e outros models. Ele inclui também outros models que não aparecem em nenhum outro diagrama.

APÊNDICE E: Descrição do modelo de dados

Foram descritas abaixo todas as entidades necessárias para a implementação do projeto e presentes nos diagramas de classes do apêndice D. Elas são apresentadas em ordem alfabética, com o nome em inglês e a tradução equivalente em português. Cada uma dessas entidades possuem uma tabela equivalente no banco de dados.

1. CategoryDifficulty (Dificuldade em categoria)

A dificuldade em categoria é uma entidade de junção entre um usuário e uma categoria (isto é, uma matéria ou área). Ela estabelece uma relação "n-para-n" entre essas entidades, e representa uma matéria ou área na qual um aluno possui dificuldade. Um aluno pode ter dificuldade em várias matérias e/ou áreas, e vários alunos podem ter dificuldade em uma mesma matéria ou área.

2. Content (Material)

Materiais podem ser criados por usuários (por exemplo, professores) como recursos adicionais à suas salas virtuais. Sendo associado com um objeto de mídia, um material pode possuir uma grande variedade de formatos, como PDF, imagens, documentos como *Word*, *Excel*, *PowerPoint*, além de recursos online, como vídeos do YouTube. Eles devem estar associados a uma matéria ou área, e podem ser recomendados para um usuário por outro. Uma vez que um material é criado, seu autor pode escolher se ele deverá ficar restrito à sala para o qual ele foi criado, ou se ele poderá ser compartilhado com outras salas.

3. Course (Sala)

Uma sala é uma agregação de usuários com diferentes papéis (no caso básico, professores e alunos). Uma sala pertence a uma categoria (matéria ou área). Usuários (essencialmente professores), de acordo com as permissões de seu papel, podem criar e compartilhar materiais na sala, e criar novidades, exercícios e exames para a sala. Outros usuários (alunos), podem fazer exames e os exercícios. Usuários que não pertençam a uma sala poderão requirir sua matrícula com um requerimento de matrícula em curso.

4. CourseContent (Material de sala)

Esta é uma entidade auxiliar de junção que estabelece o relacionamento "n-para-n" entre salas e materiais. Uma sala pode ter vários materiais, e um material pode pertencer a várias salas.

5. CourseNews (Novidade)

Uma novidade é um anúncio que um usuário pode criar e compartilhar para todos os participantes de uma sala.

6. CourseQuestion (exercício de sala)
Esta é uma entidade auxiliar de junção entre sala e exercício. Ela estabelece uma relação "n-para-n" entre essas entidades. Uma sala pode ter vários exercícios, e um exercício pode ser usado em várias salas.

7. CourseRegistrationRequest (requerimento de matrícula em sala)
Um requerimento de matrícula em sala é um requerimento que um usuário pode fazer quando ele deseja ser matriculado em um curso. Uma vez o requerimento aprovado pelo administrador, o usuário é matriculado e pode acessar todo o conteúdo da sala.

8. DataImport (importação de dados)
Um objeto de importação de dados permite a extração de dados de arquivos estruturados. Cada arquivo pode ser importado por um único objeto, que importa um tipo específico de dado (usuário, sala ou matéria/área). Além disso, quando um objeto importa usuários, eles devem ser associados a um papel que corresponderá ao papel dos usuários importados.

9. Field (área)
Uma área é uma sub-divisão de uma matéria. Salas, materiais e questões podem ser relacionados com uma área específica no lugar de uma matéria. "Geometria", por exemplo, seria uma área da matéria "Matemática". O termo "Categoria" é aplicado para se referenciar a uma entidade polimórfica que pode assumir a forma de uma matéria ou de uma área.

10. Lesson (aula)
Uma aula é uma sub-divisão de uma sala, isto é, uma sala é uma composição de aulas. Uma aula pode possuir materiais específicos a ela. Uma aula representa um determinado tópico dentro de uma sala.

11. LessonContent (material de aula)
Esta é uma entidade auxiliar de junção que estabelece o relacionamento "n-para-n" entre aulas e materiais. Uma aula pode ter vários materiais, e um material pode pertencer a várias aulas (de uma mesma sala ou de salas diferentes).

12. Medium (mídia)

Um objeto de mídia mantém uma referência a um arquivo ou um recurso online. Ele pode ser integrado e renderizado de diversas formas, dependendo do contexto.

13. Permission (permissão)
Uma permissão é concedida a um papel de usuário e que permite aos usuários com aquele papel de realizar uma determinada ação no sistema.

14. Question (exercício)
Um exercício representa uma questão que pode ser respondida com a escolha de alternativas (múltipla escolha) ou de forma dissertativa (texto). Um exercício pertence a uma categoria (uma matéria ou uma área). Pode conter objetos de mídia. Um exercício pode ser usado em uma sala, pode ser recomendado a um aluno, pode ser avaliado pelos alunos e respondido por eles no contexto de exercício complementar ou exercício em um exame.

15. QuestionCategory (categoria de exercício)
Esta é uma entidade auxiliar de junção entre um exercício e uma categoria (matéria ou área). Ela estabelece uma relação "n-para-n" entre essas entidades. Um exercício pode ser multidisciplinar e estar associado a várias categorias, e uma categoria pode ter vários exercícios a ela associados.

16. QuestionChoice (alternativa de exercício)
Os exercícios de múltipla escolha possuem objetos que representam as alternativas de respostas possíveis.

17. Rating (avaliação)
Uma avaliação é dada por um aluno para um determinado exercício, que o classifica segundo o grau de complexidade sentido pelo aluno.

18. Recommendation (recomendação)
Um usuário pode recomendar um exercício ou um material para outro usuário por meio de um objeto de recomendação.

19. RegistrationRequest (requerimento de inscrição)
Um visitante do sistema pode requerir sua inscrição por meio de um requerimento de inscrição. Os dados do visitante são registrados neste objeto, usado em seguida para gerar seu cadastro no sistema.

20. Response (resposta)
Uma resposta é dada a um exercício por um aluno. A resposta pode ser dada pelo aluno no contexto de exercício normal ou de exercício em um exame. Neste último caso, a resposta é associado ao respectivo exame. Se o exercício respondido for de múltipla escolha, a resposta também é associada às alternativas escolhidas pelo aluno.

21. ResponseChoice (alternativa de resposta)
Esta é uma entidade auxiliar de junção entre resposta e alternativa de exercício. Ela estabelece uma relação "n-para-n" entre essas entidades. Um aluno, dependendo do tipo do exercício respondido, pode selecionar mais de uma alternativa como resposta, e uma alternativa pode ser selecionada por vários alunos em suas respostas ao exercício.

22. Role (papel)
Um papel é uma agregação de permissões. Um usuário que possua um papel poderá executar somente as ações permitidas ao papel no sistema. Exemplos de papéis são: aluno, professor e administrador.

23. RolePermission (permissão de papel)
Esta é uma entidade auxiliar de junção que estabelece o relacionamento "n-para-n" entre papéis e permissões. Um papel pode possuir várias permissões, e um permissão pode ser aplicada a vários papéis.

24. Subject (matéria)
Uma matéria descreve a natureza de diversas entidades, tais como salas, materiais e exercícios. Uma matéria pode ser sub-dividida em áreas, que também servem para categorizar salas, materiais e exercícios.

25. Test (exame)
Um exame é uma agregação de exercícios que podem ser avaliados, com os seus participantes (dentro de uma sala) recebendo um feedback.

26. TestQuestion (exercício de exame)
Esta é uma entidade auxiliar de junção que estabelece o relacionamento "n-para-n" entre exame e exercício. Um exame pode ter vários exercícios, e um exercício pode pertencer a vários exames. Essa entidade ainda define o tanto que o exercício em questão vale dentro do exame.

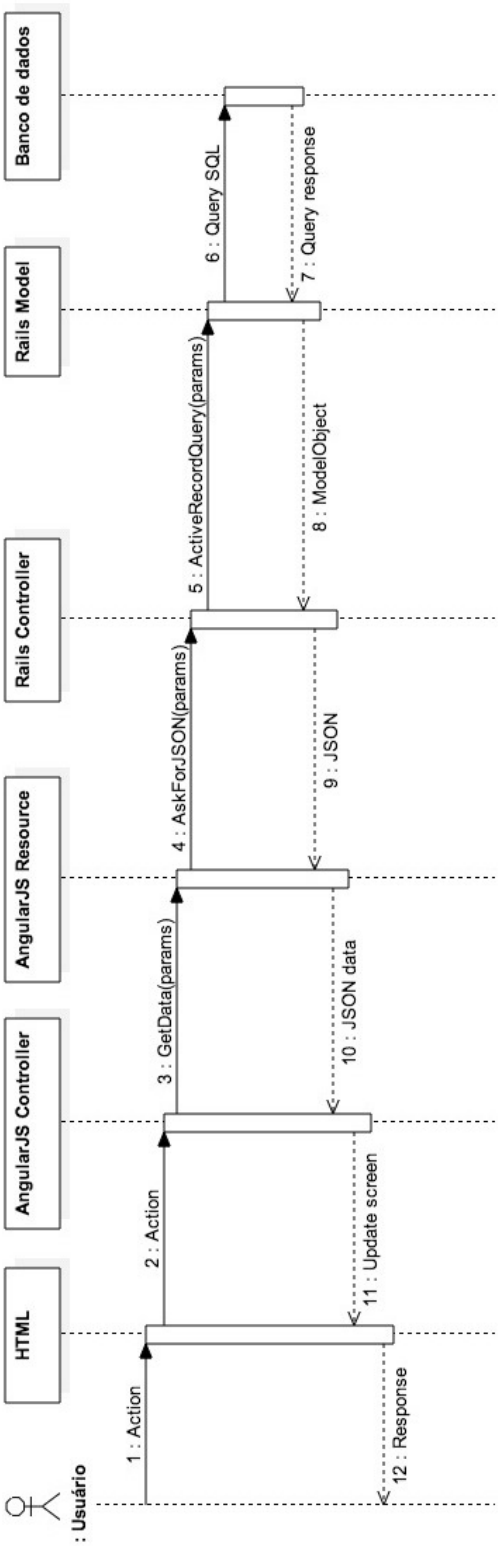
27. TestResponse
Esta é uma entidade auxiliar de junção que estabelece o relacionamento "1-para-n" entre exame e resposta. Um exame pode ter várias respostas de alunos aos seus exercícios, mas uma resposta a um exercício em um exame só pertence àquele exame. Essa entidade pode ainda conter a nota que o aluno tirou ao responder ao exercício em questão.

28. User
Um usuário representa uma pessoa dentro do sistema.

29. UserCourse
Esta é uma entidade auxiliar de junção que estabelece o relacionamento "n-para-n" entre usuário e sala. Uma sala pode ter vários participantes, e um usuário pode pertencer a várias salas.

APÊNDICE F: Integração cliente-servidor

O diagrama de sequência abaixo ilustra uma ação genérica no sistema, seja a visualização, adição, atualização ou remoção de dados, todas funcionam de forma similar.



APÊNDICE G: Dados de teste para avaliação do sistema

Desenvolveu-se um conjunto de dados fictícios para teste de forma que o sistema pudesse ser avaliado. Os dados visam simular uma pequena escola de ensino médio brasileira. Os nomes dos usuários foram gerados por um gerador de dados fictícios[19].

Para tornar a avaliação mais simples, o conjunto de dados gerado não é muito extenso. Ele é composto por 1 administrador, 6 professores, 20 alunos e somente algumas das matérias e áreas previstas pelo Ministério da Educação para o currículo do ensino médio brasileiro[20].

A todos os usuários foi atribuída uma mesma senha “12345678” para facilitar os testes.

As tabelas abaixo mostram os dados pertencentes ao conjunto.

Usuários		
Papel	**Nome**	**E-mail**
Admin.	Sophia Carvalho Fernandes	scf@gabaritei.com
Professor	Eloá Flávia Souza	eloa-flavia79@opera.com
Professor	Arthur João Giovanni Ribeiro	arthur.joao.ribeiro@hp.com
Professor	Clara Campos	clara_campos@elegantthemes.com
Professor	Henrique Leonardo Castro	henrique-leonardo93@hibu.com
Professor	Raul Gomes	raul_enrico@ow.ly
Professor	Alice Marina Fernandes	aefernandes@claro.com.br
Aluno	Luiza Santos Ferreira	luiza_santos@nba.com
Aluno	Augusto Marcelo Gomes	augusto_m_gomes@comdados.com

[19] Gerador de documentos de pessoas - http://www.4devs.com.br/gerador_de_pessoas

[20] Ministério da Educação - Parâmetros Curriculares Nacionais para o Ensino Médio (PCNEM) - http://portal.mec.gov.br/expansao-da-rede-federal/195-secretarias-112877938/seb-educacao-basica-2007048997/12598-publicacoes-sp-265002211

Aluno	Breno Matheus Pereira	breno.matheus.pereira@bplan.com.br
Aluno	Laura Reis	lreis@is.gd
Aluno	Glória Martins	gmartins@uol.com.br
Aluno	João Caio Pereira	joao_i_pereira@drimenezes.com
Aluno	Denis Castro	dcastro@ucoz.ru
Aluno	Lara Costa	lcosta@ucsd.edu
Aluno	Evelyn Gomes	evelyng@creativecommons.org
Aluno	Maria Stella Ribeiro	msribeiro@com.com
Aluno	Maria Das Flores	mfloresl@oaic.gov.au
Aluno	Samuel Bernardo Barros	sbarros@microsoft.com
Aluno	Nicolas Henrique Martins	npmartins@wisc.edu
Aluno	Alexandre Dias	alexdias@shareasale.com
Aluno	Lucas Guilherme Dias	lucas_caua@uol.com.br
Aluno	Giovana Barros	gbarros@thetimes.co.uk
Aluno	Carolina Pereira Azevedo	cpazevedo@eepurl.com
Aluno	Aline Alves Ribeiro	aline_ribeiro@parallels.com
Aluno	Luiz Silva Melo	lsmelo@blogspot.com
Aluno	Tiago Castro Martins	tcastro@acquirethisname.com

Matérias e áreas	
Matéria	**Áreas**
Português	Gramática
	Literatura
	Redação
Matemática	Álgebra
	Geometria

	Estatística
História	História do Brasil
	História Geral
Geografia	Geopolítica
	Geografia física
Física	Mecânica
	Termodinâmica
	Eletromagnetismo
Química	Química inorgânica
	Química orgânica
Biologia	Zoologia
	Botânica
	Genética

Salas	
Nome: Álgebra básica 1 **Matéria:** Matemática **Área:** Álgebra	**Professor:** Eloá Flávia Souza **Alunos:** Luiza Santos Ferreira Augusto Marcelo Gomes Breno Matheus Pereira Laura Reis Glória Martins João Caio Pereira Denis Castro Lara Costa Evelyn Gomes Maria Stella Ribeiro
Nome: Geometria 3 **Matéria:** Matemática **Área:** Geometria	**Professor:** Eloá Flávia Souza **Alunos:** Maria Das Flores

	Samuel Bernardo Barros Nicolas Henrique Martins Alexandre Dias Lucas Guilherme Dias Giovana Barros Carolina Pereira Azevedo Aline Alves Ribeiro Luiz Silva Melo Tiago Castro Martins
Nome: Literatura brasileira pré-modernismo **Matéria:** Português **Área:** Literatura	**Professor:** Arthur João Giovanni Ribeiro **Alunos:** João Caio Pereira Denis Castro Lara Costa Evelyn Gomes Maria Stella Ribeiro Maria Das Flores Samuel Bernardo Barros Nicolas Henrique Martins Alexandre Dias Lucas Guilherme Dias
Nome: Física elementar 1 **Matéria:** Física **Área:** N/A	**Professor:** Clara Campos **Alunos:** Luiza Santos Ferreira Augusto Marcelo Gomes Breno Matheus Pereira Laura Reis Glória Martins João Caio Pereira Denis Castro Lara Costa Evelyn Gomes Maria Stella Ribeiro
Nome: Biologia 2 **Matéria:** Biologia **Área:** N/A	**Professor:** Henrique Leonardo Castro **Alunos:** Evelyn Gomes Maria Stella Ribeiro Maria Das Flores Samuel Bernardo Barros Nicolas Henrique Martins

	Alexandre Dias Lucas Guilherme Dias
Nome: Química orgânica 2 **Matéria:** Química **Área:** Química orgânica	**Professor:** Raul Gomes **Alunos:** Lara Costa Evelyn Gomes Maria Stella Ribeiro Maria Das Flores Samuel Bernardo Barros Nicolas Henrique Martins Alexandre Dias Lucas Guilherme Dias Giovana Barros Carolina Pereira Azevedo
Nome: Geografia Geral e do Brasil **Matéria:** Geografia **Área:** N/A	**Professor:** Alice Marina Fernandes **Alunos:** Luiza Santos Ferreira Augusto Marcelo Gomes Breno Matheus Pereira Laura Reis Glória Martins João Caio Pereira Denis Castro Lara Costa Evelyn Gomes Maria Stella Ribeiro Maria Das Flores Samuel Bernardo Barros Nicolas Henrique Martins Alexandre Dias Lucas Guilherme Dias Giovana Barros Carolina Pereira Azevedo Aline Alves Ribeiro Luiz Silva Melo Tiago Castro Martins