UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

TEACHUB: APLICATIVO PARA GESTÃO DE AULAS PARTICULARES

LUCAS AMARAL

BLUMENAU
2017

LUCAS AMARAL

TEACHUB: APLICATIVO PARA GESTÃO DE AULAS PARTICULARES

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Profª Luciana Pereira de Araújo, Mestra - Orientadora

BLUMENAU
2017

TEACHUB: APLICATIVO PARA GESTÃO DE AULAS PARTICULARES

Por

LUCAS AMARAL

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Profª. Luciana Pereira de Araújo, Mestra – Orientadora, FURB
Membro:	Prof. Gilvan Justino, Mestre – FURB
Membro:	Profª. Gabriele Jennrich Bambineti, Especialista – FURB

Blumenau, 30 de Junho de 2017

Dedico este trabalho aos meus pais que sempre me inspiraram e motivaram para fazer a diferença na vida das pessoas, a minha irmã querida e companheira, aos meus amigos da vida e a minha namorada motivadora e carinhosa.

AGRADECIMENTOS

Aos meus pais por todas as oportunidades, carinho e amor que recebi e continuo recebendo.

A minha namorada por toda motivação, paciência, carinho e amor.

A minha irmã pelo carinho e companheirismo.

A minha orientadora, Prof.ª Luciana Pereira de Araújo, pela motivação, ajuda e reconhecimento na realização do trabalho.

Aos meus amigos, que são poucos mas verdadeiros e importantes na minha vida.

Aos meus colegas de trabalho, que me apoiaram muito durante o desenvolvimento deste trabalho.

Tudo o que temos que decidir é o que fazer com o tempo que nos é dado.

J. R. R. Tolkien

RESUMO

Este trabalho apresenta o desenvolvimento de um aplicativo *web* responsivo e móvel voltado para gestão de aulas particulares. O aplicativo, denominado TEACHub, tem como objetivo o engajamento de professor e alunos na colaboração e interação para melhorar a aprendizagem. O TEACHub foi criado para ser um sistema colaborativo, incorporando funcionalidades como mural de publicações, *chat*, compartilhamento de arquivos, acompanhamento de desempenho do aluno e avaliação do serviço prestado pelo professor. Para o desenvolvimento do trabalho foi utilizada a plataforma NodeJS no módulo servidor, o Quasar *framework* no *front-end* do aplicativo e o banco de dados MySQL. O aplicativo foi colocado em teste em duas turmas da professora orientadora, somando 37 participantes testando as funcionalidades, interface e operacionalidade.

Palavras-chave: Aulas particulares. Sistemas colaborativos. Aplicativo híbrido.

ABSTRACT

This essay presents the development of a responsive and mobile web application for private lessons management. The application, known as TEACHub, proposes the engagement of tutors and students on collaboration and interaction for the betterment of learning. TEACHub was created to be a collaborative system, incorporating functionalities such as publications board, chat, file sharing, student performance monitoring and tutor service evaluation. In the development of the application was used NodeJS platform for the server module, Quasar framework in the application front-end and MySQL for the database. The application was put to test in two classes teached by this essay's adviser professor, a total of 37 students were testing the application's functionalities, interface and operationality.

Key-words: Private lessons. Collaborative systems. Hybrid applications.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface

AVA – Ambiente Virtual de Aprendizagem

CSCW – Computer Supported Cooperative Work

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

JS – JavaScript

MOODLE – Object Oriented Dynamic Learning Environment

ORM – Object Relactional Modeling

REST – Representational State Transfer

RF – Requisitos Funcionais

RNF – Requisitos Não Funcionais

SPA – Single Page Applications

UC – Diagrama de Casos de Uso

UML – Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

A utilização da tecnologia de informação como recurso para o ensino é indispensável, sendo capaz de identificar falhas de aprendizado e de fornecer meios para superar problemas na aprendizagem (SOUZA; SOUZA, 2010). Segundo Almeida (2001, p. 2), “com o uso da tecnologia de informação e comunicação, professores e alunos têm a possibilidade de utilizar a escrita para descrever/reescrever suas ideias, comunicar-se, trocar experiências e produzir histórias.”.

Um exemplo de uso de tecnologia no âmbito educacional é o uso de sistemas colaborativos no apoio à interação, colaboração e troca de experiências entre alunos e professores. Segundo Scheidemantel (2013, p. 27), os sistemas colaborativos “[...] surgem diante do novo perfil da sociedade informacional, que pede colaboração, interação, inteligência e organização acoplados ao ambiente virtual”.

Embora a tecnologia de informação tenha proporcionado à educação novas oportunidades de melhoria, alguns alunos possuem dificuldade de compreender assuntos discutidos em sala de aula ou simplesmente querem buscar algum conhecimento extra (SOUZA; SOUZA, 2010). Sendo assim, uma das possibilidades é a contratação de aulas particulares. Com a procura dessas aulas particulares, surgem vários professores que podem ajudar esses alunos a alcançar seus objetivos (COSTA, 2007). Dessa forma, é necessário que haja uma gestão eficiente na qualidade do serviço prestado para que o professor obtenha o destaque necessário afim de alcançar seus alunos (PROFES, 2016).

Existem algumas ferramentas que se propõem realizar a gestão de aulas particulares, mas na maioria são portais que possuem acesso somente via navegador web e são pouco adaptáveis à dispositivos móveis. As soluções existentes para aplicativos móveis são menos robustas que suas concorrentes web, não possuindo interação e colaboração entre aluno e professor, sendo prejudicial no desempenho do aluno e no seu acompanhamento pelo professor.

Segundo Castro e Menezes (2012, 135), “A colaboração tem impacto determinante na construção do conhecimento, pois envolve níveis de cognição mais elaborados do que os envolvidos na ação individual de aprendizagem”. Ou seja, uma pessoa pode obter muito conhecimento através do estudo individual, mas nas atividades em grupo, o indivíduo põe em prova toda sua capacidade de defender uma teoria, de argumentar e debater sobre determinado assunto.

Fedoce e Squirra (2011) observam que novos paradigmas de ensino e aprendizagem estão se formando a partir das inovações tecnológicas. Com o aumento de meios de comunicação em tempo real e de redes sociais, a demanda de uma revisão na forma de ensino é necessária, de forma que contemple novas tecnologias.

Considerando esta demanda, o trabalho apresenta o desenvolvimento de um aplicativo *web* responsivo e móvel voltado para gestão de aulas particulares. O aplicativo tem como propósito auxiliar os professores particulares a gerir suas aulas e manter viva a interação e comunicação com seus alunos. O aplicativo foi concebido para ser um sistema colaborativo, incorporando funcionalidades como mural de publicações, *chat*, compartilhamento de arquivos, acompanhamento de desempenho do aluno e avaliação do serviço prestado pelo professor.

1.1 OBJETIVOS

O objetivo geral deste trabalho é disponibilizar um aplicativo para gestão de aulas particulares.

Os objetivos específicos são:

a) fornecer suporte à interação e colaboração entre alunos e professor;
b) disponibilizar um Web Service para as regras de negócio do aplicativo, de modo que possa ser integrado com qualquer aplicativo de ensino, desde que siga a arquitetura de comunicação definida.

1.2 ESTRUTURA

Este trabalho está dividido em quatro capítulos. O primeiro capítulo é composto pela introdução do trabalho, que relata os problemas encontrados e justifica o motivo do desenvolvimento deste, dos objetivos definidos para o mesmo e a apresentação de sua estrutura.

O segundo capítulo apresenta a fundamentação teórica, explorando conceitos sobre sistemas colaborativos, tecnologias empregadas no ensino, desenvolvimento híbrido de aplicativos móveis utilizando tecnologias web e trabalhos correlatos. No terceiro capítulo é apresentado o desenvolvimento da aplicação, onde são listados os principais requisitos, bem como é realizada a especificação do sistema através de diagramas da Unified Modeling Language (UML). Além disso, são exploradas as técnicas e ferramentas utilizadas para o desenvolvimento deste trabalho, são apresentados trechos de códigos fontes mais relevantes,

bem como é relatada uma avaliação em campo realizada para testar a usabilidade do aplicativo.

O quarto capítulo apresenta as conclusões e limitações deste trabalho. Por fim, são sugeridas extensões como trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os principais conceitos utilizados para o desenvolvimento deste trabalho, sendo dividido em quatro seções. A primeira seção aborda sobre sistemas colaborativos, indicando seus conceitos e características necessárias para o desenvolvimento de um sistema colaborativo. A segunda seção descreve sobre tecnologias aplicadas ao ensino, apresentando formas diferentes de aprendizagem. A terceira seção expõe desenvolvimento híbrido de aplicativos móveis utilizando tecnologias web. Por fim, a quarta seção apresenta os trabalhos correlatos a este, destacando os pontos chaves de cada um deles.

2.1 SISTEMAS COLABORATIVOS

Segundo Gutwein e Greenberg (2001), o termo Sistemas Colaborativos representa a tradução dos termos *groupware* e Computer Supported Cooperative Work (CSCW). Alguns autores consideram *groupware* e CSCW como sinônimos. Já outros utilizam *groupware* para referenciar sistemas computacionais responsáveis por auxiliar o trabalho em grupo e CSCW para referenciar uma perspectiva holística, do próprio sistema e os efeitos psicológicos, sociais e organizacionais do trabalho em grupo (GREENBERG; GUTWEIN, 2001; NICOLACI-DA-COSTA; PIMENTEL, 2012).

Para o desenvolvimento de sistemas colaborativos não é suficiente apenas conhecer de tecnologia e linguagens de programação, é necessário entender o comportamento humano e as novas formas de trabalho e organização social. Contudo, os sistemas colaborativos possuem o poder de construir novas formas de trabalho e interação social (NICOLACI-DA-COSTA; PIMENTEL, 2012).

Na projeção e seleção de sistemas colaborativos é comum o uso de teorias e modelos. Estes recursos fornecem uma visão de como as pessoas colaboram e como elas trabalham em grupo (DAMIANI, 2006; FUKS et al, 2012; RIBEIRO, 2013). Segundo Fuks et al. (2012) destacam-se a teoria dos jogos, a teoria da atividade, o modelo 3C de colaboração, padrões de colaboração e o modelo de Tuckman sobre o desenvolvimento de grupo. Dentre esses o modelo 3C, que será utilizado para o desenvolvimento deste trabalho, é um modelo que observa e analisa a colaboração em três dimensões: comunicação, coordenação e cooperação (FUKS et al., 2012). A Figura 1 representa estas três dimensões.

Figura 1 - Três dimensões do modelo 3C

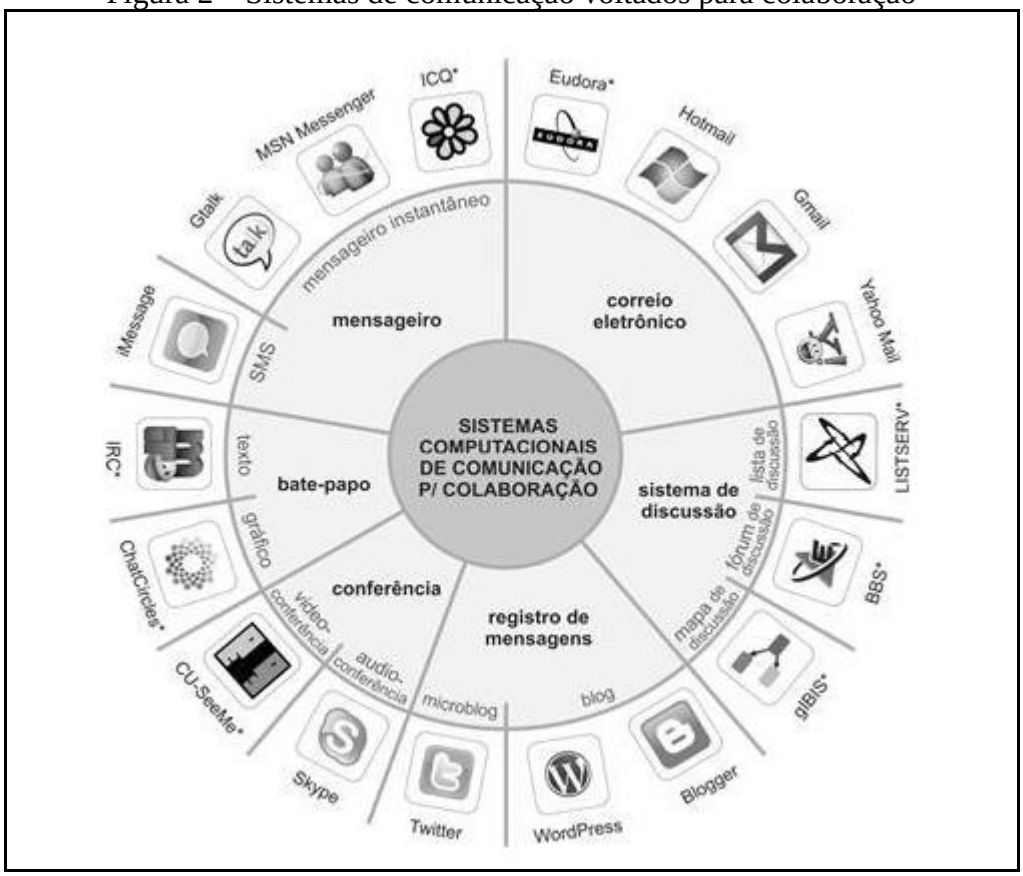
Fonte: Fuks et al. (2012).

Segundo Fuks et al. (2012) a comunicação é a dimensão responsável pela ação, pois através desta dimensão, as pessoas trocam informações, negociam e tomam decisões. A dimensão de coordenação é responsável por organizar atividades que evitem desperdício de recursos e gerir conflitos que surgem na comunicação (FUKS et al., 2012). Por sua vez, a cooperação é a dimensão responsável por propiciar um ambiente favorável ao trabalho em grupo (FUKS et al., 2012).

Os sistemas colaborativos ganham maior destaque com a computação móvel, uma vez em que os dispositivos móveis expandem os limites físicos e aumentam o tempo disponível de colaboração. Por exemplo, um grupo de pessoas pode utilizar serviços de localização para compartilhar informações associadas ao local onde estão. Outro exemplo da presença da colaboração é a comunicação e interação entre usuários conectados em uma rede (FILIPPO et al., 2012).

A comunicação, sendo um dos pilares da colaboração, é fundamental como estrutura básica para qualquer sistema colaborativo, uma vez que eles são utilizados na construção de redes sociais, correios eletrônicos, chats, videoconferências e outros (PIMENTEL; GEROSA; FUKS, 2012). Segundo Pimentel, Gerosa e Fuks (2012), existem vários tipos de sistemas de comunicação voltados para a colaboração, sendo ilustrados na Figura 2.

Figura 2 – Sistemas de comunicação voltados para colaboração



Fonte: Pimentel, Gerosa e Fuks (2012).

Na Figura 2 pode-se visualizar os tipos de sistemas computacionais de comunicação para colaboração classificados por Pimentel, Gerosa e Fuks (2012). Essa classificação é baseada no tipo de sistema de mensagem e ainda são apresentados exemplos de aplicações que utilizam este modelo de comunicação. Dentre eles, destacam-se os sistemas de bate-papo, mensageiro, registro de mensagens e sistema de discussão, utilizados na composição do aplicativo desenvolvido.

Os sistemas mensageiros e de bate-papo servem para uma ou mais pessoas se comunicarem, trocando mensagens de forma que os destinatários das mensagens possam receber notificações de novas mensagens (PIMENTEL; GEROSA; FUKS, 2012). Além de trocar mensagens, uma grande característica dos sistemas mensageiros e de bate-papo é visualizar os usuários conectados ao sistema em tempo real, ou seja, verificar se o usuário está *online* (PIMENTEL; GEROSA; FUKS; 2012).

Os sistemas de registro de mensagens e de listas de discussão estão presentes frequentemente em redes sociais, listas de discussão e fóruns. As listas de discussão surgiram para aumentar o poder dos correios eletrônicos no que se refere a comunicação entre várias pessoas. Funcionam através de um e-mail principal, responsável por enviar mensagens para uma lista de contatos e assim os membros da lista podem discutir os assuntos entre si. Já o

registro de mensagens frequentemente relacionado aos *blogs* e *microblogs*, possibilita aos usuários postarem conteúdos de forma pública com potencial para serem lidos por muitos. Geralmente possuem imagens, quadros e outras mídias vinculadas à postagem. No caso de *microblogs* as mensagens são limitadas em caracteres, tornando as mensagens mais informais e rápidas (PIMENTEL; GEROSA; FUKS, 2012).

Os fóruns surgiram para organizar essas listas de forma mais sistêmica, sendo organizadas em tópicos, nas quais as mensagens estão dispostas encadeada conforme suas respostas. Os fóruns de forma geral possuem controle de acesso e perfil de usuário, compartilhamento de arquivo e sistemas de bate-papo (PIMENTEL; GEROSA; FUKS, 2012).

Segundo Souza et al. (2012), até pouco tempo atrás as informações eram mantidas em sigilo visando vantagens competitivas. O autor observa que atualmente essas vantagens não são mais tão evidentes, uma vez que através de sistemas de comunicação e colaboração é gerado uma massa de informações gigantesca, disponível e acessível a outras pessoas. Porém, com essa grande disponibilidade de informações, surge a dificuldade de assimilar e gerar conhecimento.

O conhecimento é difícil de gerar pois ele é o produto da relação entre as informações e as experiências, sendo muito subjetivo e individual (SOUZA et al., 2012). Esse conhecimento por ser tão individual é difícil de ser compartilhado ou ensinado a outras pessoas. Segundo Souza et al. (2012), o conhecimento só se torna explícito para as outras pessoas quando o detentor primário consegue expor em diagramas, desenhos ou esquemas este conhecimento, de forma que outras pessoas consigam assimilar as suas experiências e também absorvê-lo.

Agrupando esses documentos gerados pelos detentores do conhecimento é possível gerar uma base de conhecimento, com o objetivo de disponibiliza-la para outras pessoas de forma mais colaborativa. Este repositório de conhecimento é chamado de memória de grupo e é fundamental pois expõe o conhecimento necessário para um grupo funcionar de forma harmoniosa. Além disso, cria uma cultura de não dependência nos grupos, ou seja, se algum membro sair do grupo, o conhecimento não sai com ele (SOUZA et al., 2012). Uma das formas de suportar essa memória em grupo é através dos sistemas de comunicação mencionados anteriormente.

Baseado nos fatores apresentados, pode-se afirmar que os sistemas colaborativos auxiliam na gestão do conhecimento e construção de memória de grupo. São responsáveis por sistematizar e disponibilizar a base de conhecimento de um grupo, incentivando o engajamento dos membros (SOUZA et al., 2012).

2.2 TECNOLOGIAS EMPREGADAS NO ENSINO

As novas tecnologias estão transformando os métodos de organização, comunicação, produção, comercialização, diversão, ensino e aprendizagem da sociedade contemporânea (MORAN, 2000; SILVA, 2011). Com o avanço da tecnologia da informação e comunicação, as informações estão mais acessíveis, a comunicação está mais rápida e as culturas estão mais interligadas (AGUIAR, 2004).

Segundo Almeida (2001, p. 6), o uso da tecnologia "[..] permite a quem o utiliza percorrer distintos caminhos, criar múltiplas conexões entre informações, textos e imagens; ligar contextos, mídias e recursos.". Esse benefício do uso da tecnologia é muito importante, uma vez que democratiza as informações e permite com que as pessoas busquem fatos para fundamentar suas opiniões e críticas mais facilmente.

Quando empregada no ensino, as tecnologias de informação e comunicação estimulam o aluno a buscar e socializar conhecimento, além de funcionar como um reforço no aprendizado (SOUZA, I.; SOUZA, L., 2010). É papel do professor utilizar esses recursos a favor do aprimoramento e transformação da sua prática (ALMEIDA, 2001). O professor então passa a diminuir sua dependência do livro didático e começa a usá-lo mais como uma das fontes de conhecimento.

Um exemplo de tecnologia de informação utilizada na aprendizagem é o Ambiente Virtual de Aprendizagem (AVA). O AVA é utilizado em cursos a distância e no apoio de cursos presenciais, servindo como suporte para transmissão de conteúdos e para comunicação (SEBASTIÃO, 2015). Um dos AVAs mais utilizados é o Modular Object-Oriented Dynamic Learning Environment (MOODLE) (TARCIA; COSTA, 2010). O MOODLE é uma plataforma de código aberto, baseado na web, que disponibiliza recursos para o ensino e aprendizagem a distância (SEBASTIÃO, 2015; MARQUES; CAETANO, 2014). O MOODLE é um software de gestão de aprendizagem, que permite gerir as salas de aula, participantes, cursos on-line, atividades escolares, entre outras necessidades de aprendizagem (SEBASTIÃO, 2015; MARQUES; CAETANO, 2014).

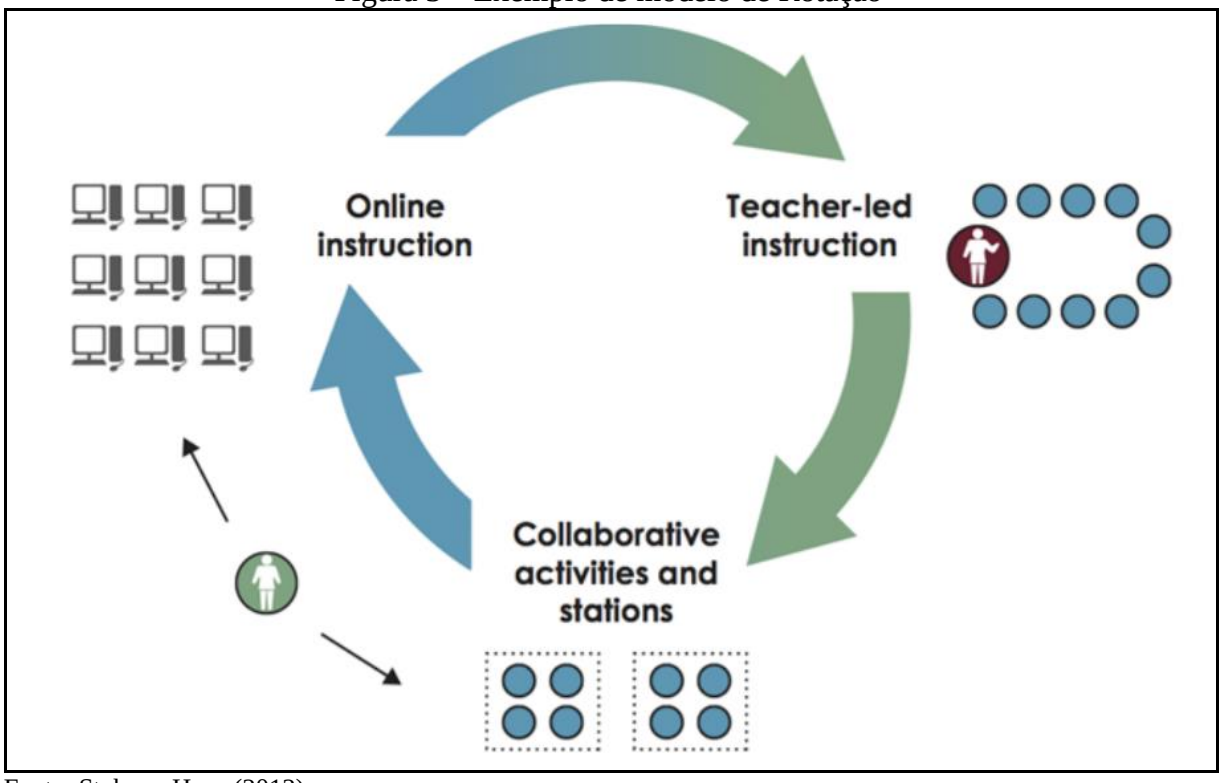
Com a união das novas tecnologias de educação e os novos meios de aprendizagem, obtém-se o ensino híbrido, mais suscetível a mudanças e engajamento de alunos, professores e pais (CHRISTENSEN; HORN; STAKER, 2013). O ensino híbrido promove um melhor aprendizado utilizando sistemas colaborativos e ambientes online, assim como otimiza o tempo do professor, individualiza o acompanhamento do aluno e maximiza a participação dos pais na vida escolar do aluno (FUNDAÇÃO LEMANN, 2017).

O modelo híbrido de ensino surge como uma inovação sustentada, ou seja, não tem como objetivo uma quebra total de paradigma de como funciona a aprendizagem atual. Ela busca melhorar os processos e evoluir organicamente para um modelo de ensino ideal para as gerações futuras (CHRISTENSEN; HORN; STAKER, 2013).

Segundo Staker e Horn (2012), os principais modelos de ensino híbrido emergentes são o modelo de Rotação, Flex, A La Carte e o modelo Virtual Enriquecido. O modelo de Rotação ainda possui algumas subdivisões, divididas em Rotação por Estações de Trabalho, Laboratório Rotacional, Sala de Aula Invertida e Rotação Individual (SOUZA; ANDRADE, 2016).

O modelo de rotação tem como objetivo definir passos de aprendizagem que funcionam como estações (STAKER; HORN, 2012). Assim o aluno poderá aprender através de diversas dinâmicas diferentes. Esse método pode ser aplicado em uma disciplina, como Matemática ou até mesmo em um conteúdo específico da matéria (SOUZA; ANDRADE, 2016). Staker e Horn (2012) explicam esse processo através do exemplo representado na Figura 3.

Figura 3 – Exemplo de modelo de Rotação



Fonte: Staker e Horn (2012).

Na Figura 3 pode-se ver a divisão de três estações diferentes. Elas funcionam como um ciclo contínuo, com o objetivo de criar diferentes experiências de ensino aos alunos. Na primeira estação, rotulada de *Teacher-led instruction*, o professor orienta os alunos acerca dos

conteúdos. A segunda estação possui diversos grupos de alunos, onde cada grupo deve buscar as informações e realizar atividades colaborativas (*Collaborative activities and stations*). Na última estação do exemplo, os alunos utilizam os computadores para realizar as tarefas e se comunicarem. Nesse modelo, é possível que existam grupos de alunos em estações distintas em um mesmo momento.

O modelo Flex possui como base o ambiente virtual e quase todas as atividades são feitas *online*. Os alunos seguem uma ementa flexível e adaptada para atender singularmente as necessidades dos estudantes (CHRISTENSEN; HORN; STAKER, 2013).

O modelo A La Carte é aplicado quando os alunos participam de um curso online e podem realizar outros cursos presencialmente. Este modelo flexibiliza os horários dos estudantes e permite aos alunos estudarem assuntos específicos que muitas vezes não estão disponíveis em todas as instituições de ensino (CLAYTON CHRISTENSEN INSTITUTE, 2017).

O último modelo apontado por Staker e Horn (2012) é o modelo Virtual Enriquecido. Este modelo tem como objetivo disponibilizar em instituições que possuem a maioria das disciplinas presenciais, acesso a materiais, conteúdos e lições *online*.

A combinação do uso de tecnologias e os métodos tradicionais trazem aos alunos uma nova forma de engajamento e incentivam os estudantes a buscarem mais conteúdo online. Quanto mais os alunos se aprofundam nos assuntos, mais eles trazem novidades e dúvidas para as salas de aula (ESPÍNDOLA, 2016).

2.3 DESENVOLVIMENTO HÍBRIDO DE APLICATIVOS MÓVEIS UTILIZANDO TECNOLOGIAS *WEB*

Existem diversas abordagens de desenvolvimento para aplicativos móveis, sendo as mais populares a nativa e a híbrido (LOPES, 2016). O termo híbrido no contexto deste trabalho, se refere a um aplicativo desenvolvido a partir de tecnologias web, como Hypertext Markup Languagem 5 (HTML) e JavaScript (JS), mas disponibilizado em forma de aplicativo nativo para diversas plataformas como Android, iOS e Windows Phone (MENDES; GARBAZZA; TERRA, 2014).

Quando o objetivo é desenvolver um aplicativo multiplataforma é comum o uso da abordagem híbrida, uma vez que esta utilize padrões já estabelecidos pelo desenvolvimento web (LOPES, 2016). Esta abordagem possui um custo de desenvolvimento muito menor a um nativo, pois não necessita de equipes especialistas em Java para Android ou Objective-

C/Swift para iOS (CAMPAGNOLI, 2015; MENDES; GARBAZZA; TERRA, 2014). A curva de aprendizado para o híbrido fica menor ainda quando a equipe responsável por desenvolver o aplicativo possui conhecimentos em HTML, JS e Cascading Style Sheets (CSS) (LOPES, 2016; SILVA et al., 2013).

Segundo Silva, Pires e Neto (2015, p. 27) “Esses aplicativos tipicamente envolvem o conteúdo HTML dentro de um controle de navegador em modo de tela cheia, sem a barra de endereços visível ou outros controles embutidos do navegador”. Os autores explicam que é através de um WebView que as aplicações são processadas pelas páginas web.

Segundo Lopes (2016, p. 2), “A solução mais comum atualmente para construção de aplicativos multiplataforma é o Cordova”. O *framework* Apache Cordova cria uma camada nativa para o aplicativo, responsável por instanciar um navegador onde o aplicativo será executado e realizar a comunicação do aplicativo com as camadas nativas do dispositivo móvel (LOPES, 2016; MATTJE, 2014).

Existem outros *frameworks* que auxiliam na criação de aplicativos móveis. Um deles é o Quasar Framework. O Quasar é um *framework* que auxilia no desenvolvimento de aplicativos híbridos, focado na aparência do aplicativo e na interação do usuário com o mesmo (QUASAR, 2017). Ele não substitui o Cordova, pelo contrário, ele utiliza o Cordova e foca apenas no *front-end* do aplicativo. Além de gerar aplicações para dispositivos móveis, o Quasar Framework pode gerar aplicações *web* e *desktop (*QUASAR, 2017).

O Quasar utiliza uma biblioteca para desenvolver os componentes de tela chamada VueJS (QUASAR, 2017). VueJS é uma biblioteca para aplicações *web* que permite estender a sintaxe padrão do HTML, criando componentes reutilizáveis para cada parte da aplicação (VUE, 2017). Em conjunto, as bibliotecas Vuex e VueRouter auxiliam no desenvolvimento de Single Page Applications (SPA), pois tratam a manutenção do estado dos dados no *front-end* da aplicação e toda a parte de rotas e mudança de telas (VUEJS, 2017).

2.4 TRABALHOS CORRELATOS

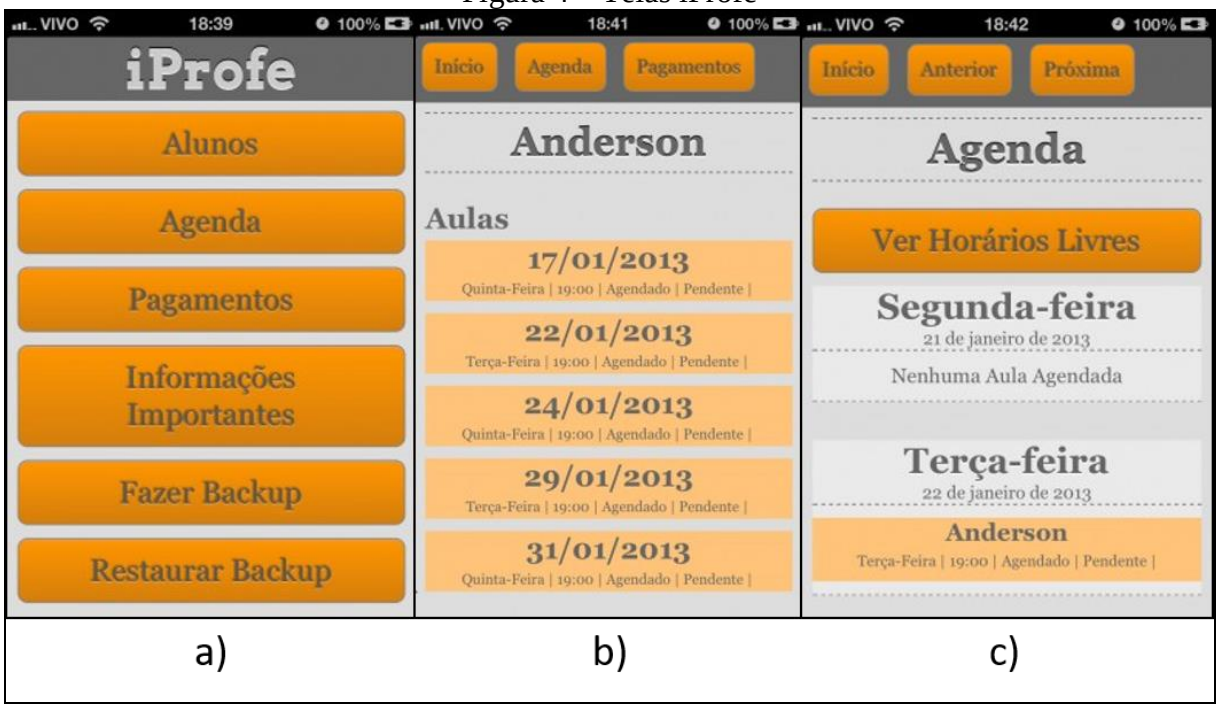
Pode-se citar como trabalhos correlatos o aplicativo móvel iProfe (SCARTEZINI, 2013), o portal Profes (PROFES, 2012) e a plataforma Superprof (SUPERPROF, 2011). O aplicativo iProfe possui entre 500 e 1000 downloads em dispositivos Android, mas sua última atualização foi em agosto de 2014 (PLAY STORE, 2017). O portal Profes possui 100 mil alunos e aproximadamente 7 mil professores cadastrados (PROFES, 2012). A plataforma Superprof possui aproximadamente 1,5 milhões de professores cadastrados (SUPERPROF,

2011). Nas próximas seções são apresentados com detalhes cada trabalho correlato e ao final é apresentada uma tabela comparativa entre eles.

2.4.1 Aplicativo iProfe

A ferramenta iProfe é um aplicativo móvel feito especificamente para professores particulares. Através do mesmo é possível gerenciar os alunos, aulas, pagamentos, agenda semanal e grade horária (SCARTEZINI, 2013). A Figura 4 apresenta algumas telas do aplicativo.

Figura 4 – Telas iProfe



Fonte: AppsZoom (2016).

As telas apresentadas na Figura 4 representam respectivamente, a tela inicial (Figura 4 a), a tela referente as aulas com um aluno (Figura 4 b) e a tela da agenda do professor (Figura 4 c). Na tela inicial pode-se ver botões para acessar as telas de alunos, agenda, pagamentos, informações importantes, fazer *backup* e restaurar *backup*. Na tela representada na Figura 4 (b), pode-se verificar a agenda do professor com um determinado aluno. Na tela apresentada na Figura 4 (c), pode-se ver a agenda do professor, com as respectivas aulas e alunos da semana.

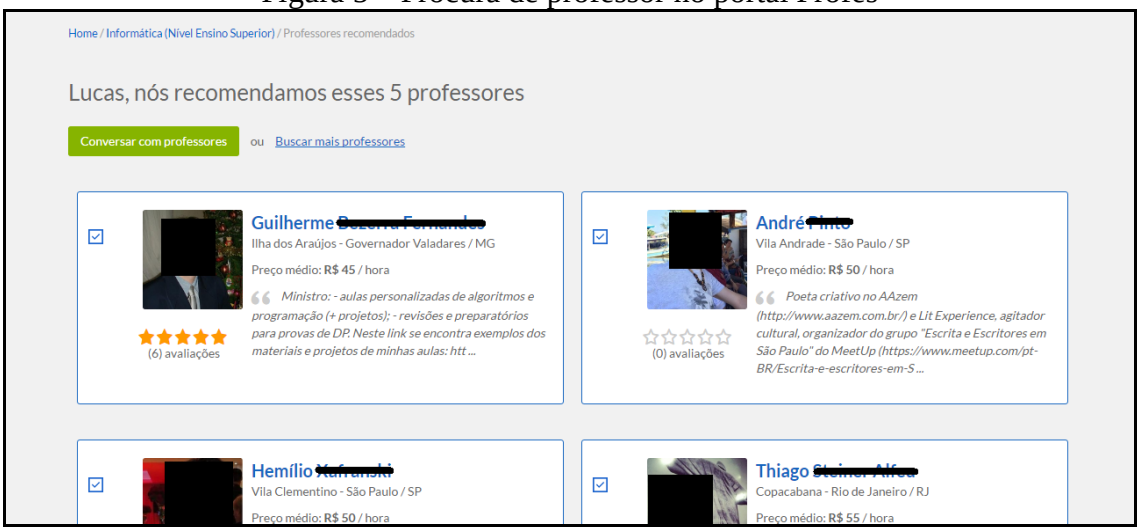
O iProfe é um aplicativo que não possibilita a comunicação do professor com seus alunos. O professor utiliza-o basicamente como agenda de aulas e como meio de controlar o pagamento do aluno (SCARTEZINI, 2013). A ferramenta iProfe não pode ser considerado um sistema de colaboração, por trabalhar sem comunicação entre os usuários do aplicativo.

2.4.2 Portal Profes

O portal Profes é considerado o maior portal de aulas particulares do Brasil. Ele foi fundado em 2012 e é um *marketplace* com aproximadamente 7672 professores, distribuídos em 72 matérias e 5644 especialidades (PROFES, 2012).

No Profes, o aluno busca os melhores professores e solicita orçamentos. De acordo com o custo-benefício, o aluno escolhe a melhor opção e pode realizar o pagamento das aulas através do portal que é integrado com várias formas de pagamento. Quando o pagamento é feito através de meios digitais, o Profes apenas libera o pagamento após o feedback positivo do usuário (PROFES, 2012). Na Figura 5 é representada a procura de um professor no portal Profes.

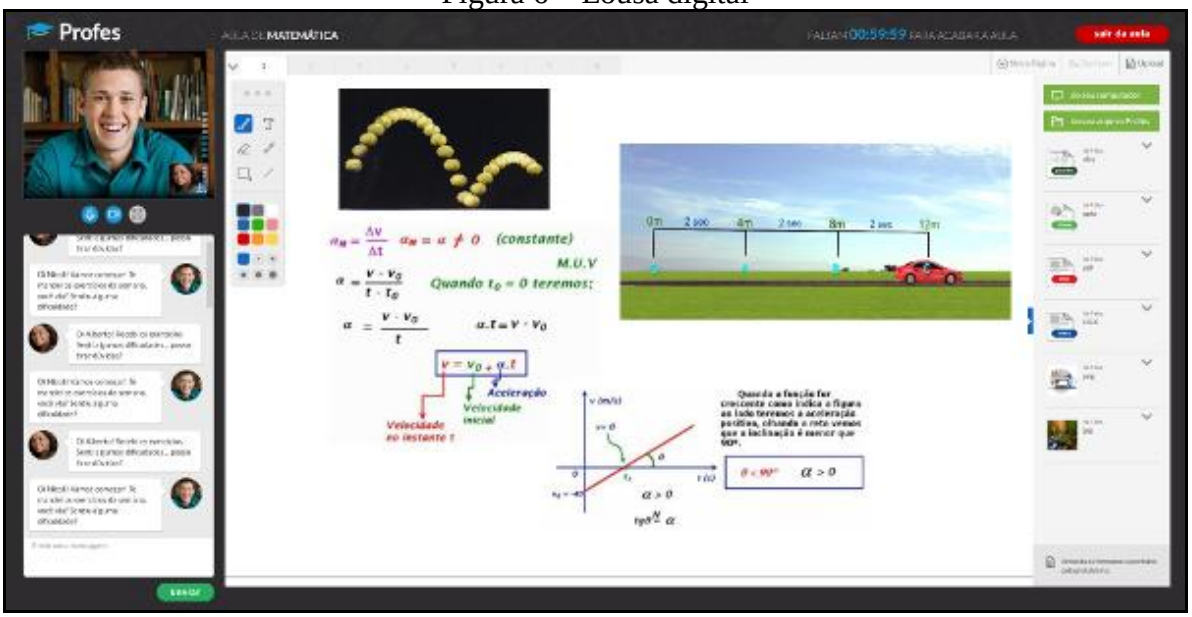
Figura 5 – Procura de professor no portal Profes



Fonte: Profes (2012).

O aluno e professor utilizam o portal para gerenciar as aulas e até mesmo realizar conferências e aulas *on-line*. O portal oferece total apoio através de *chat* e fórum para discursões, além de uma lousa compartilhada para o professor e aluno (PROFES, 2012). A Figura 6 representa essa interação que o Profes proporciona ao professor e aluno.

Figura 6 – Lousa digital



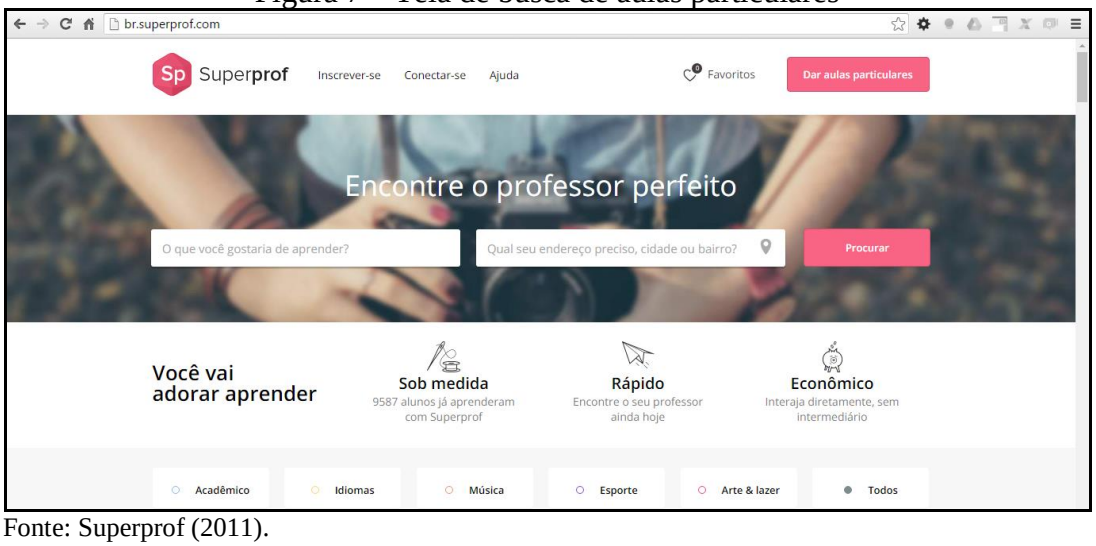
Fonte: Profes (2016).

Através dos recursos disponíveis no portal, pode-se afirmar que o portal Profes possui suporte a colaboração entre os professores e alunos. A lousa digital auxilia o ensino e aumenta a interação no ensino online.

2.4.3 Plataforma Superprof

A plataforma Superprof fornece apoio para a divulgação de forma *online* de cursos, sendo que gerencia todo o ciclo de vida de anúncios, solicitação de orçamento, agendamento das aulas e feedback do aluno (SUPERPROF, 2011). Ela é utilizada em diversos países, como Brasil, Portugal, Canadá, Estados Unidos da América, Reino Unido, França, Itália, Alemanha, entre outros (SUPERPROF, 2011). Na Figura 7 é representada a tela de busca de uma aula particular no Superprof.

Figura 7 – Tela de busca de aulas particulares



Fonte: Superprof (2011).

O Superprof disponibiliza a pesquisa de aulas particulares, com professores verificados por especialistas e avaliados por antigos alunos (SUPERPROF, 2011). O aluno pode entrar em contato com um professor para explicar suas dúvidas e necessidades sobre um determinado curso (SUPERPROF, 2011). A Figura 8 representa o formulário de solicitação de aula.

Figura 8 - Formulário de solicitação de aula

Fonte: Superprof (2011).

A partir do momento que um aluno envia uma solicitação de aula, o professor responsável pelo curso recebe uma notificação. A Figura 9 representa a tela de solicitações de aulas recebidas.

Figura 9 - Tela de mensagens do professor

Fonte: Superprof (2011).

Após contratada a aula, o aluno pode avaliar a qualidade do serviço prestado de uma aula contratada (SUPERPROF, 2011). A Figura 10 representa a tela de avaliação de uma aula particular.

Figura 10 – Tela de avaliação de aula particular do Superprof

Fonte: Superprof (2011)

2.4.4 Comparação entre os trabalhos relacionados

As ferramentas apontadas como trabalhos correlatos apresentam diferentes formas de suprir as necessidades do mercado. A ferramenta iProfe atende somente o mercado de aplicativos móveis, enquanto as demais atingem o nicho *web*. O Quadro 1 representa uma comparação entre as características das três.

Quadro 1 – Comparativo de características dos trabalhos correlatos

Característica	iProfe	Profes	Superprof
Aplicativo Móvel	Sim	Não	Não
Aplicativo *Web*	Não	Sim	Sim
Anúncio de Cursos	Não	Sim	Sim
Compartilhamento de Recursos/Arquivos	Não	Sim	Não
Chat professor/alunos	Não	Sim	Parcial
Agenda/Calendário	Parcial	Sim	Parcial
Controle de pagamentos	Parcial	Sim	Sim
Suporte aula *on-line*	Não	Sim	Não
Gestão de Alunos	Sim	Sim	Não
Gestão de Aulas	Não	Sim	Não
Notificações/Avisos do Professor	Não	Sim	Sim
Mural de mensagens / notificações	Não	Sim	Sim

Fonte: Elaborado pelo autor.

O portal Profes e a plataforma Superprof possuem um viés maior de colaboração e interação, enquanto o iProfe apenas oferece aos professores uma melhor forma de organizar

suas aulas. As ferramentas Profes e Superprof possuem anúncios de cursos, mural de notificações e mensagens. Já a ferramenta Profes suporta aulas *online,* diferentemente das outras. A gestão das aulas é outro ponto forte no Profes. A ferramenta suporta compartilhamento de arquivos, mural de mensagens e *chat* entre alunos e professores. Em todos os trabalhos correlatos, tem-se uma forma de acompanhar o calendário das aulas e controle de pagamento. Apenas as ferramentas Profes e Superprof suportam pagamento *online*.

3 DESENVOLVIMENTO

Neste capítulo estão descritas as particularidades técnicas do aplicativo desenvolvido, sendo dividido em quatro seções. A primeira seção realiza um levantamento das informações. A segunda seção especifica o projeto desenvolvido, apresentando os Requisitos Funcionais (RF), Requisitos Não Funcionais (RNF), diagrama de Casos de Uso (UC), diagrama de classes, diagrama de arquitetura e diagrama de atividades. A terceira seção aborda sobre as metodologias e técnicas empregadas no desenvolvimento do trabalho, bem como apresenta os principais trechos de código e relata sobre a operacionalidade do sistema. Por fim, a quarta seção aborda sobre o teste de usabilidade realizado, faz um comparativo entre os trabalhos correlatos e este e apresenta os resultados obtidos com o desenvolvimento.

3.1 LEVANTAMENTOS DE INFORMAÇÕES

Este trabalho apresenta a construção de um aplicativo *web* e móvel, desenvolvido utilizando conceitos de desenvolvimento híbrido de aplicativos móveis. Este trabalho é voltado para gestão de aulas particulares e aplica conceitos de sistemas colaborativos para permitir a interação entre professor e alunos. O aplicativo permite ao professor particular gerir suas aulas, turmas, horários, locais, entre outras informações de forma que seus alunos possam visualizar essas informações. O professor pode criar avaliações para seus alunos, informando como está o desempenho de cada aluno em particular. Do mesmo modo, os alunos podem avaliar o serviço prestado pelo professor. O aplicativo disponibiliza um mural de publicações e compartilhamento de arquivos e um *chat* para comunicação entre alunos e professor.

O trabalho desenvolvido possui um módulo servidor, responsável por tratar as requisições do aplicativo através de um Web Service Representational State Transfer (REST) Ful. Este módulo servidor é responsável por disponibilizar as informações necessárias ao aplicativo e processar as regras de negócio. Os dados são armazenados em banco de dados presente no servidor. Como o aplicativo precisa requisitar informações para este servidor, é necessário que o dispositivo no qual o aplicativo for executado esteja conectado à internet.

3.2 ESPECIFICAÇÃO

Esta seção especifica o sistema desenvolvido através dos diagramas da UML, contemplando os requisitos funcionais, não funcionais e diagramas desenvolvidos. Foi

utilizado a ferramenta Astah Community (ASTAH, 2009) para o desenvolvimento dos diagramas.

O Quadro 2 apresenta os Requisitos Funcionais (RF) previstos para o sistema e sua rastreabilidade, ou seja, vinculação com o UC associado.

Quadro 2 – Requisitos funcionais

Requisitos Funcionais	**Caso de Uso**
RF01: O aplicativo deverá permitir a manutenção de conta do usuário, com nome, data de nascimento, e-mail, telefone, qualificações, descrição do usuário e redes sociais para contato.	UC01
RF02: O aplicativo deverá permitir ao usuário professor manter seus cursos oferecidos. Cada curso possui título, descrição, assunto, horários disponíveis, valor, individual/em grupo, avaliação de satisfação.	UC02
RF03: O aplicativo deverá permitir ao usuário professor manter turmas de um curso. Cada turma possui quais são os participantes e os horários.	UC03
RF04: O aplicativo deverá permitir ao usuário aluno cancelar a turma a qualquer momento.	UC04
RF05: O aplicativo deverá permitir ao usuário aluno acompanhar seu desempenho.	UC05
RF06: O aplicativo deverá permitir ao usuário aluno avaliar a qualidade do serviço prestado pelo professor.	UC06
RF07: O aplicativo deverá permitir aos participantes de uma turma interagir através de um mural de publicações.	UC07, UC11
RF08: O aplicativo deverá permitir a troca de mensagens entre usuários.	UC08, UC10
RF09: O aplicativo deverá permitir aos participantes de uma turma compartilhar arquivos.	UC09
RF10: O aplicativo deverá notificar os usuários sobre novas mensagens, novas publicações, novos comentários em publicações e novas avaliações.	UC12
RF11: O aplicativo deverá permitir aos professores avaliar e acompanhar o desempenho dos alunos.	UC13
RF12: O aplicativo deverá permitir aos usuários visualizar se os outros usuários estão *online* no aplicativo.	UC14
RF13: O aplicativo deverá permitir aos alunos visualizar suas turmas e horários.	UC15

Fonte: Elaborado pelo Autor.

O Quadro 3 lista os requisitos não funcionais previstos para o sistema.

Quadro 3 – Requisitos não funcionais

Requisitos Não Funcionais
RNF01: O módulo servidor deverá ser implementado utilizando a plataforma NodeJS.
RNF02: O aplicativo deverá ser implementado utilizando o Quasar Framework.
RNF03: O módulo servidor deverá utilizar o MySQL para armazenar os dados.
RNF04: O aplicativo deverá ser responsivo quando utilizado sua versão em navegadores.
RNF05: O aplicativo deverá ser suportado em dispositivos Android.
RNF06: O aplicativo deverá permitir troca de mensagem de forma síncrona e assíncrona.
RNF07: O aplicativo deverá controlar as funcionalidades por perfil de acesso.
RNF08: O aplicativo deverá representar os usuários *onlines* com um sinal verde.

Fonte: Elaborado pelo autor.

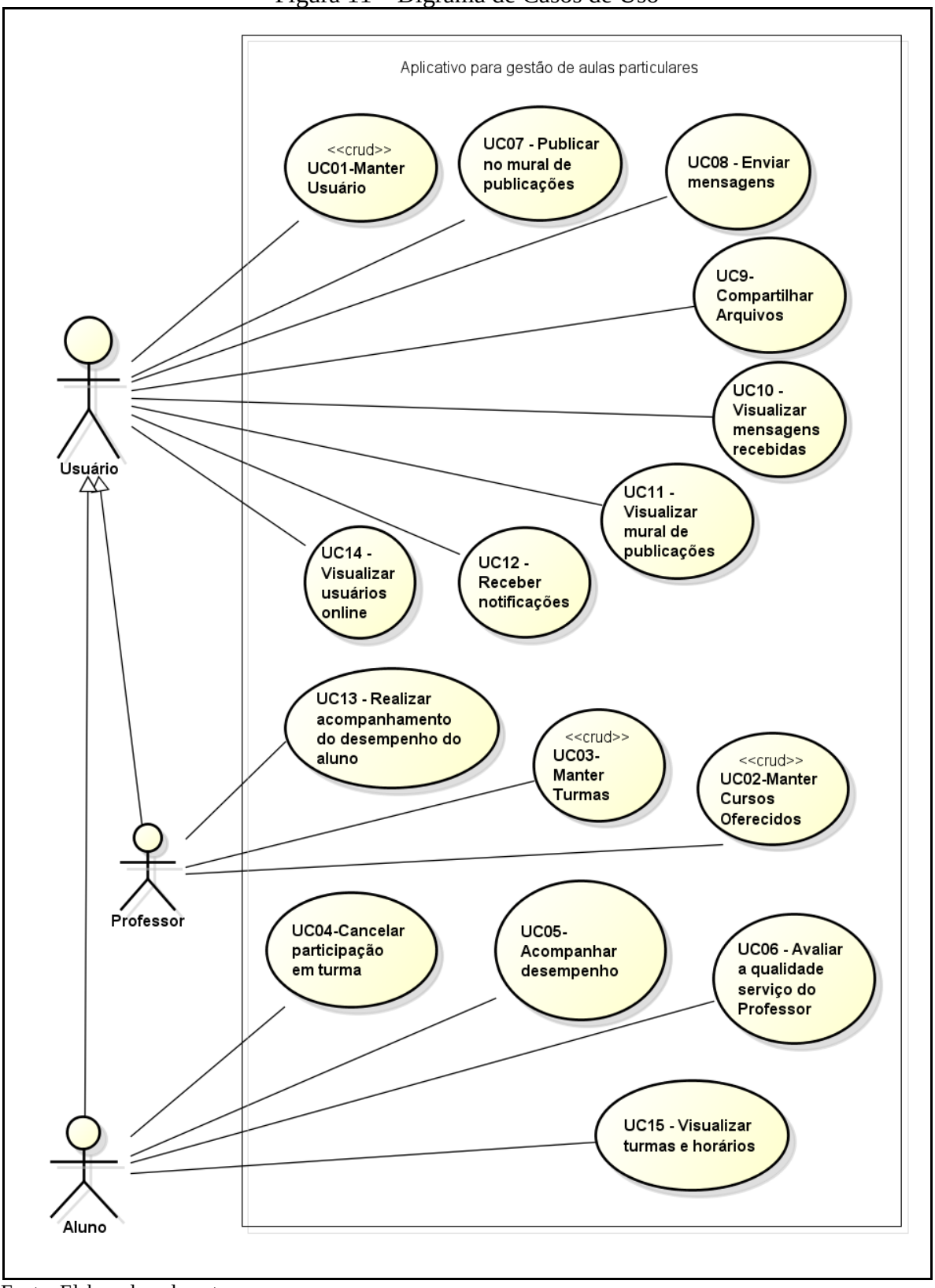
3.2.1 Diagrama de casos de uso

A Figura 11 apresenta o diagrama de casos de uso do aplicativo desenvolvido. Os atores apresentados no diagrama e presentes no sistema são: `usuário`, `professor` e `aluno`.

Os atores `professor` e `aluno` são especializações do ator `usuário` e ambos os atores especialistas compartilham os casos de uso do `usuário`. O ator `usuário` tem permissão para manter sua conta, enviar e receber mensagens privadas e/ou do grupo da turma, publicar no mural da turma, adicionar anexos nas publicações do mural e receber notificações.

Além dos casos de uso herdados do ator `usuário`, o ator `professor` realiza a manutenção dos seus cursos e turmas e participa do acompanhamento do desempenho dos `alunos`. O ator `aluno` pode cancelar sua participação na turma, acompanhar seu desempenho, avaliar o serviço prestado pelo `professor` e visualizar suas turmas e horários.

Figura 11 – Digrama de Casos de Uso



Fonte: Elaborado pelo autor.

Nas próximas subseções são apresentadas as narrativas de casos de uso dos principais casos de uso do sistema, identificando os atores envolvidos, pré-condições, fluxo principal, fluxos alternativos e a pós-condição.

3.2.1.1 UC06 - Avaliar a qualidade do serviço do professor

O caso de uso relatado nesta subseção é relacionado com a qualidade das aulas oferecidas pelo professor. O aluno por meio desse caso de uso pode dar uma nota e fazer um comentário sobre o curso que o mesmo está participando. O Quadro 4 explica mais detalhadamente o caso de uso.

Quadro 4 – Descrição do caso de uso Avaliar a Qualidade do Serviço do Professor (UC06)

Caso de Uso	**Avaliar a qualidade do serviço do Professor**
Descrição	O aplicativo deverá permitir o aluno avaliar o serviço prestador por um professor
Ator	Aluno
Pré-condição	O aluno deverá estar autenticado
Fluxo principal	1. O aluno informa a sua avaliação do serviço prestado pelo professor; 2. O aplicativo envia as informações ao módulo servidor; 3. O módulo servidor valida as informações, realiza a operação e retorna ao aplicativo; 4. O aplicativo informa a situação ao usuário.
Fluxo alternativo	3. O módulo servidor valida e constata que o aluno não preencheu a nota do professor; 4. O módulo servidor retorna o erro ao aplicativo; 5. O aplicativo informa a situação ao aluno.
Pós-condição	O aluno avaliou o serviço prestado pelo professor. O professor recebeu uma notificação da avaliação realizada pelo aluno.

Fonte: Elaborado pelo autor.

Como descrito no Quadro 4, o aluno deve estar autenticado para realizar o fluxo principal deste cenário. Estando autenticado, o aluno navega até as suas turmas e seleciona a turma que deseja avaliar. O aluno informa a nota e adiciona um comentário à sua avaliação. O aplicativo então envia estas informações ao módulo servidor. O módulo servidor em posse destas informações, valida e realiza a operação de salvar a avaliação. O resultado da operação é retornado ao aplicativo que por sua vez, informa ao aluno a situação.

Após finalizar a operação, o professor recebe uma notificação da avaliação realizada pelo aluno na sua tela inicial. O professor então pode visualizar a avaliação recebida clicando na notificação.

3.2.1.2 UC07 - Publicar no mural de publicações

O caso de uso relatado nesta seção descreve como publicar e comentar no mural de publicações de uma turma. O professor e os alunos da turma podem realizar estas ações acessando o mural da turma. O Quadro 5 explica em detalhes o caso de uso.

Quadro 5 – Descrição do caso de uso Publicar no mural de publicações (UC07)

Caso de Uso	**Publicar no mural de publicações**
Descrição	O aplicativo deverá permitir ao usuário publicar no mural de publicações
Ator	Usuário
Pré-condição	O usuário deve estar autenticado.
Fluxo principal	1. O usuário escolhe a opção: 1.1 Nova publicação; 1.2 Novo comentário. Passos para 1.1 Nova Publicação: 1. O usuário informa ao aplicativo o título, o conteúdo e opcionalmente anexos para uma nova publicação; 2. O aplicativo envia as informações da nova publicação ao módulo servidor; 3. O módulo servidor valida a requisição e realiza a operação; 4. O módulo servidor responde a requisição do aplicativo; 5. O aplicativo informa ao usuário a situação da nova publicação; 6. O aplicativo notifica os demais participantes da turma sobre a nova publicação no mural. Passos para 1.2 Novo comentário: 1. O usuário informa ao aplicativo o novo comentário para publicação; 2. O aplicativo envia as informações do novo comentário ao módulo servidor; 3. O módulo servidor valida a requisição e realiza a operação; 4. O módulo servidor responde a requisição do aplicativo; 5. O aplicativo informa ao usuário a situação do novo comentário; 6. O aplicativo notifica os demais participantes da turma sobre o novo comentário.
Fluxo alternativo	3. O módulo servidor valida a requisição e encontra um erro ao cadastrar; 4. O módulo servidor responde a requisição do aplicativo com o erro que aconteceu; 5. O aplicativo informa o usuário da situação.
Pós-condição	O usuário publicou no mural de publicações da turma

Fonte: Elaborado pelo autor.

No Quadro 5 o fluxo principal começa com o usuário escolhendo uma das opções: nova publicação ou novo comentário. Escolhendo uma nova publicação, o usuário informa um título para a publicação, um conteúdo e opcionalmente arquivos para anexar. Com essas informações o aplicativo envia uma requisição ao módulo servidor. O módulo servidor valida a requisição e realiza a operação, persistindo a nova publicação. O módulo servidor retorna à situação da requisição ao aplicativo e o aplicativo informa ao usuário.

No caso de novo comentário, o usuário informa o novo comentário para uma publicação. O aplicativo envia a requisição ao módulo servidor. Este por sua vez, faz a validação, persiste o novo comentário da publicação e responde a requisição do aplicativo. O aplicativo então informa o usuário da situação.

Ambos os fluxos realizam uma etapa de notificação. Após terminar, o aplicativo notifica os participantes da turma na qual foi adicionado a nova publicação ou comentário. Os participantes da turma recebem a notificação na tela inicial do aplicativo.

3.2.1.3 UC08 – Enviar mensagens

O caso de uso relatado nesta seção contempla a comunicação entre usuários do aplicativo via mensagem direta e/ou mensagem em grupo. O Quadro 6 apresenta maiores detalhes sobre o caso de uso.

Quadro 6 – Descrição caso de uso Enviar mensagens (UC08)

Caso de Uso	**Enviar mensagens**
Descrição	O aplicativo deverá permitir aos usuários enviarem mensagens entre si.
Ator	Usuário
Pré-condição	O usuário deve estar autenticado.
Fluxo principal	1. O usuário escolhe a opção: 1.1 Nova mensagem direta para outro usuário; 1.2 Nova mensagem para grupo da turma. Passos para 1.1 Nova mensagem direta para outro usuário: 1. O usuário acessa a conversa com o usuário destino; 2. O usuário informa ao aplicativo a nova mensagem; 3. O aplicativo envia as informações da nova mensagem ao módulo servidor; 4. O módulo servidor valida a requisição e realiza a operação; 5. O módulo servidor responde a requisição do aplicativo; 6. O aplicativo informa o usuário da situação da nova mensagem; 7. O aplicativo notifica o usuário destino da nova mensagem. Passos para 1.2 Nova mensagem para grupo de uma turma: 1. O usuário acessa a conversa do grupo de uma turma; 2. O usuário informa ao aplicativo a nova mensagem; 3. O aplicativo envia as informações da nova mensagem ao módulo servidor; 4. O módulo servidor valida a requisição e realiza a operação; 5. O módulo servidor responde a requisição do aplicativo; 6. O aplicativo informa o usuário da situação da nova mensagem; 7. O aplicativo notifica os usuários da turma sobre a nova mensagem
Fluxo alternativo	4. O módulo servidor valida a requisição e encontra um erro ao cadastrar a mensagem; 5. O módulo servidor responde a requisição informando o erro que aconteceu; 6. O aplicativo informa o usuário da situação;
Pós-condição	Uma nova mensagem é enviada

Fonte: Elaborado pelo autor.

O Quadro 6 descreve duas opções para fluxo principal. Na primeira opção, nova mensagem direta para outro usuário, o usuário acessa a tela da conversa com outro usuário e informa a nova mensagem ao aplicativo. O aplicativo envia a requisição ao módulo servidor. Este por sua vez valida a requisição e realiza persistência da nova mensagem. O módulo servidor responde a requisição do aplicativo e o aplicativo informa o usuário da situação. O aplicativo também notifica o usuário destino da mensagem.

Na segunda opção, nova mensagem para grupo da turma, o usuário acessa a tela da conversa em grupo de uma turma e informa a nova mensagem ao aplicativo. O aplicativo envia a requisição ao módulo servidor, este por sua vez valida a requisição e realiza persistência da nova mensagem. O módulo servidor responde a requisição do aplicativo e o aplicativo informa o usuário da situação. O aplicativo também notifica os usuários do grupo da turma.

3.2.1.4 UC13 - `Realizar acompanhamento do desempenho do aluno`

O caso de uso relatado nesta seção pertence somente ao professor, pois apenas ele é autorizado a criar uma atividade de avaliação dos alunos da turma. Cada atividade possui uma descrição, as notas e comentários dos respectivos alunos da turma. O Quadro 7 descreve melhor o caso de uso.

Quadro 7 – Descrição do caso de uso Realizar acompanhamento do desempenho do aluno (UC13)

Caso de Uso	**Realizar acompanhamento do desempenho do aluno**
Descrição	O aplicativo deverá permitir aos professores realizar o acompanhamento do desempenho do aluno
Ator	Professor
Pré-condição	O professor deve estar autenticado.
Fluxo principal	1. O usuário escolhe a opção: 1.1 Nova avaliação; 1.2 Tornar uma avaliação pública aos alunos; 1.3 Acompanhar o desempenho dos alunos. Passos para 1.1 Nova avaliação: 1. O professor informa ao aplicativo uma descrição para a avaliação, as notas e comentários dos respectivos alunos da turma; 2. O aplicativo envia as informações para o módulo servidor; 3. O módulo servidor valida e realiza a operação; 4. O módulo servidor retorna à situação ao aplicativo; 5. O aplicativo recebe a situação e informa ao professor. Passos para 1.2 Tornar uma avaliação pública aos alunos: 1. O professor acessa as avaliações e seleciona uma avaliação; 2. O professor informa ao aplicativo que a avaliação deve estar pública aos alunos; 3. O aplicativo envia as informações para o módulo servidor; 4. O módulo servidor valida e realiza a operação; 5. O módulo servidor retorna à situação ao aplicativo; 6. O aplicativo recebe a situação e informa ao professor; 7. O aplicativo notifica os alunos da turma sobre a avaliação. Passos para 1.3 Acompanhar o desempenho dos alunos: 1. O professor acessa a tela de turmas; 2. O professor requisita ao aplicativo a tela de avaliações de uma turma específica; 3. O aplicativo requisita as avaliações de uma turma para o módulo

	servidor; 4. O módulo servidor busca essas informações e retorna para o aplicativo; 5. O aplicativo abre a tela de avaliações com as informações retornadas pelo módulo servidor
Fluxo alternativo	3. O módulo servidor valida e constata que o professor não preencheu a nota de um ou mais alunos; 4. O módulo servidor retorna uma mensagem de erro; 5. O aplicativo informa ao professor que nem todos os alunos foram avaliados.
Pós-condição	O usuário professor realiza o acompanhamento do desempenho do aluno. Os alunos recebem notificação sobre a avaliação.

Fonte: Elaborado pelo autor.

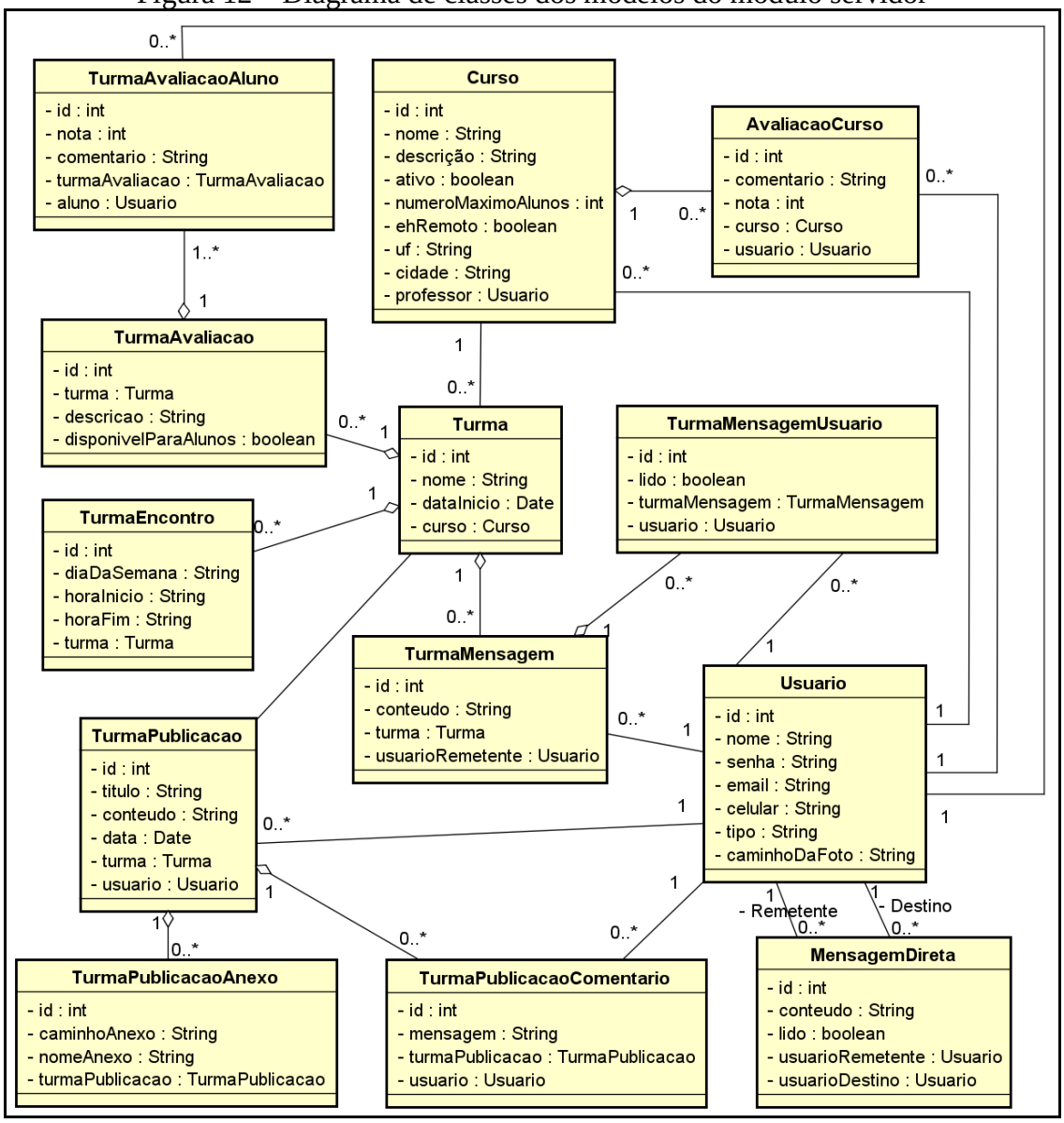
Como descrito no Quadro 7, o professor autenticado acessa a tela de avaliações de alunos. Nesta tela ele pode acompanhar as avaliações feitas para uma turma e também pode criar novas avaliações. Para criar uma nova avaliação, basta o professor informar uma descrição para a avaliação, as notas e comentários dos respectivos alunos. Se o professor não preenche a nota de um ou mais alunos, o aplicativo informa ao professor que ele deve avaliar todos os alunos da turma. O professor pode manter a avaliação em privado, sem que os alunos possam ver a avaliação criada.

Após criar ou editar uma avaliação, os alunos avaliados recebem uma notificação na tela inicial do aplicativo. Clicando na notificação, os alunos podem verificar as suas avaliações na turma em que foram avaliados.

3.2.2 Diagrama de classes

Esta seção contempla os diagramas de classes do sistema desenvolvido. As classes apresentadas no diagrama de classes representado na Figura 12 estão presentes no módulo servidor, que encapsula as regras de negócio do trabalho desenvolvido.

Figura 12 – Diagrama de classes dos modelos do módulo servidor



Fonte: Elaborado pelo autor.

A Figura 12 apresenta a composição entre as classes do sistema. A classe `Usuario` é uma das classes chaves do sistema, possuindo as informações sobre os usuários da aplicação e se relacionando com as classes `Curso`, `AvaliacaoCurso`, `TurmaAluno`, `TurmaMensagem`, `TurmaMensagemAluno`, `TurmaPublicacao`, `TurmaPublicacaoComentario` e `MensagemDireta`.

A classe `Curso` possui informações básicas dos cursos oferecidos na aplicação e possui ligação com as classes `Turma` e `AvaliacaoCurso`. A classe `AvaliacaoCurso` apenas possui as informações das avaliações que um determinado curso recebeu.

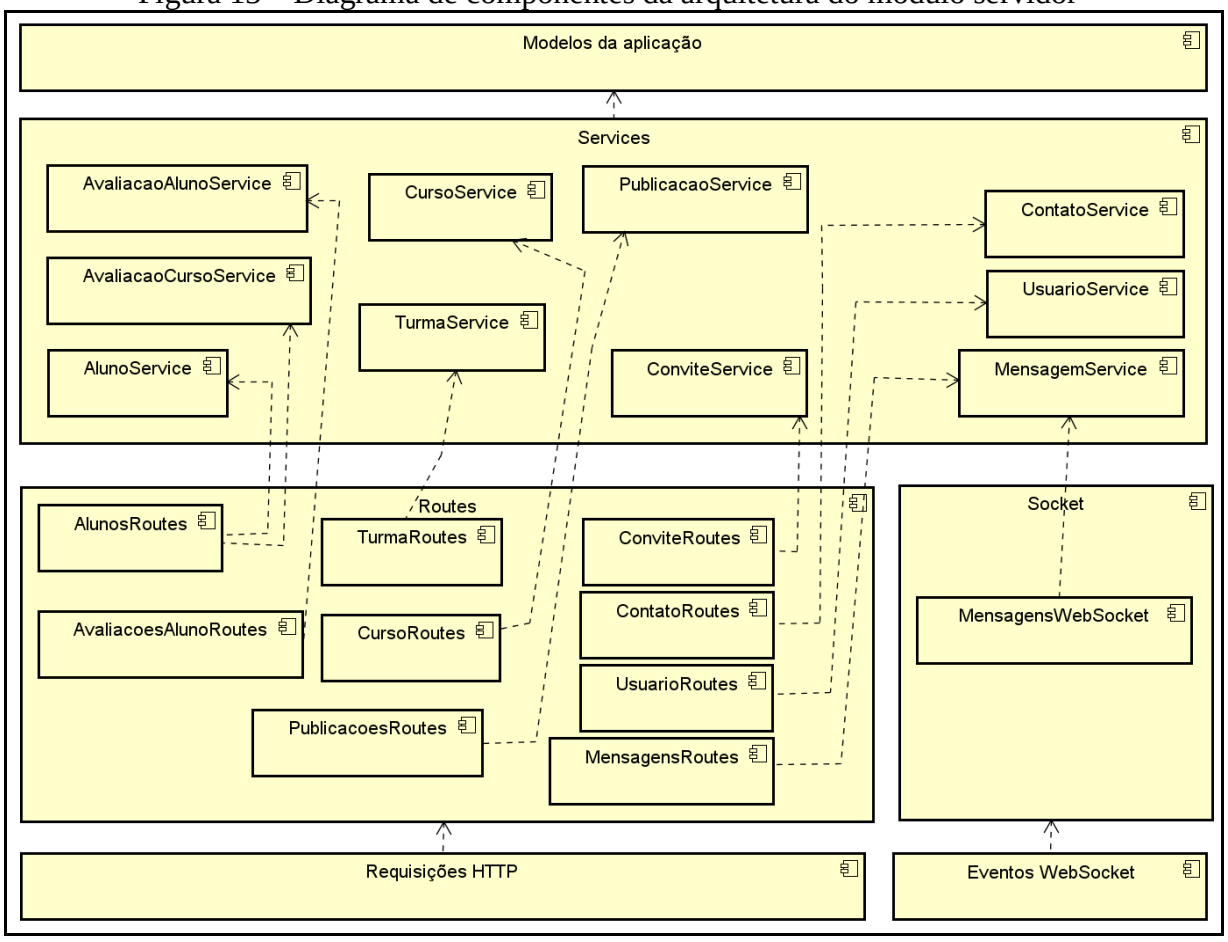
A classe `Turma` é responsável por armazenar as informações básicas das turmas e compor as demais informações referentes a turmas representadas nas classes `TurmaAluno`, `TurmaAvaliacao`, `TurmaEncontro`, `TurmaMensagem` e `TurmaPublicacao`.

As demais classes presentes na Figura 12 representam as funcionalidades restantes da aplicação, como mural de publicações, *chat*, avaliações das atividades dos alunos e informações referentes aos encontros das turmas. O diagrama de classe contém apenas os modelos do sistema, sendo que a ligação entre os modelos do sistema e o restante dos componentes são mostrados na próxima seção.

3.2.3 Diagrama de arquitetura do sistema

Nesta seção é descrita a arquitetura do sistema utilizando dois diagramas de componentes. Foram criados dois diagramas para especificar melhor a arquitetura do módulo servidor e do *front-end* do aplicativo. A Figura 13 representa a arquitetura do módulo servidor.

Figura 13 – Diagrama de componentes da arquitetura do módulo servidor



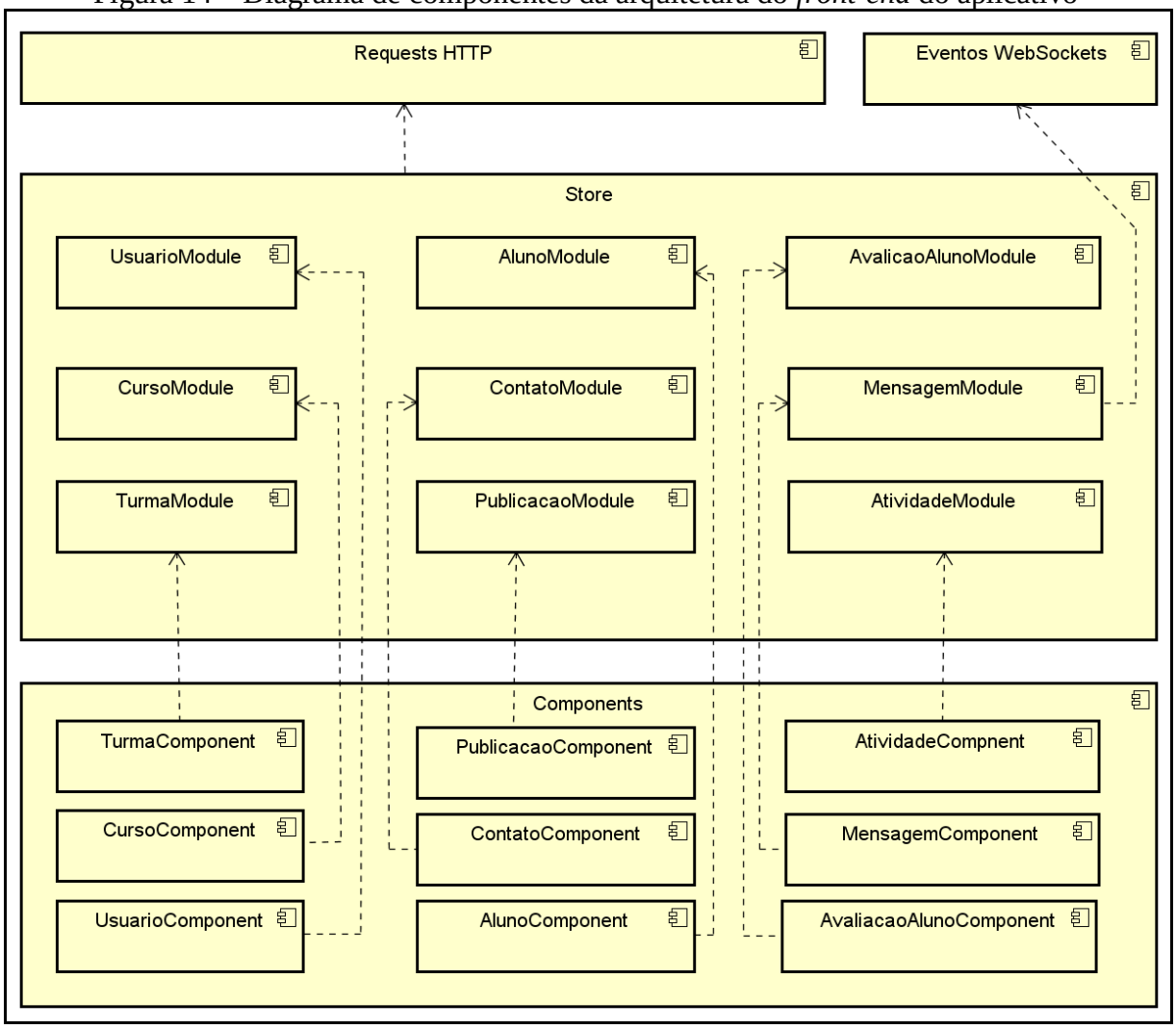
Fonte: Elaborado pelo autor.

O módulo servidor é dividido em quatro camadas principais e duas externas. A primeira camada é referente as rotas da Application Programming Interface (API) RESTFul expostas pelo módulo servidor e é chamada de `Routes`. As rotas se comunicam com a segunda camada chamada de `Services`. Os serviços são responsáveis por realizar as lógicas

de negócio e retornar os dados às rotas. Para persistir ou consultar os dados, a camada de serviços se comunica com a última camada representada pelos modelos da aplicação, através de um Framework Object Relactional Modeling (ORM).

Existe ainda a camada chamada `Socket` responsável por lidar com eventos WebSocket presentes na troca de mensagens entre usuários. Ela também se comunica com a camada `Services`. Além das quatros camadas principais, existem as duas camadas externas. Elas são responsáveis por tratar as requisições HTTP e eventos WebSocket.

Figura 14 – Diagrama de componentes da arquitetura do *front-end* do aplicativo



Fonte: Elaborado pelo autor.

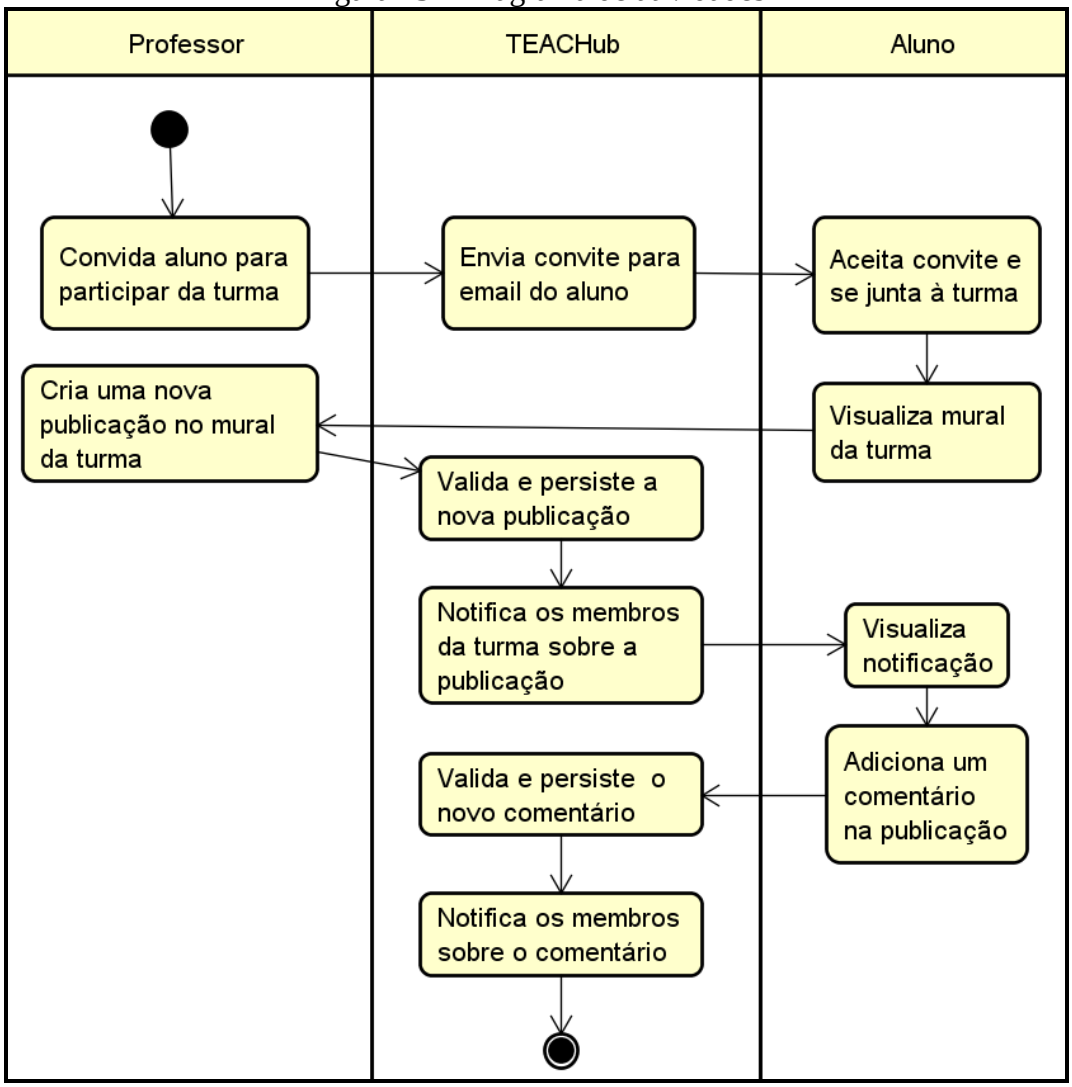
A Figura 14 apresenta o *front-end* do aplicativo que é formado por duas camadas. A camada `Components` representa as telas do aplicativo. Acima dos componentes está a camada `Store`. A `Store` é dividida em módulos. Cada módulo contém as informações necessárias para renderizar as componentes de tela. Além disso, cada módulo possui regras de negócio responsáveis por lidar com interações do usuário com o aplicativo. Por exemplo, o módulo `PublicacaoModule` possui as informações das publicações de uma determinada turma e as

regras de negócios referentes ao mural de publicações da turma. Neste caso, as regras de negócio seriam por exemplo, enviar ao módulo servidor uma nova publicação, novo comentário, entre outros.

3.2.4 Diagrama de atividade

Os diagramas de atividade desenvolvidos para o sistema são facilitadores no que se refere ao entendimento do funcionamento do aplicativo. A Figura 15 representa o diagrama de atividades referente a interação e colaboração entre professor e alunos no mural de publicações da turma.

Figura 15 – Diagrama de atividades



Fonte: Elaborado pelo autor.

O diagrama de atividade apresentado na Figura 15 possui os três principais atores do sistema: `Professor`, `Aluno` e `TEACHub`. No diagrama apresentado é dado início com o `Professor` convidando um aluno para participar da turma. O `TEACHub` envia o convite para o e-mail do aluno. Após aceitar o convite e se juntar a turma, o `Aluno` visualiza o mural de

publicações da turma. O `Professor` cria uma nova publicação no mural da turma e o `TEACHub` valida e persiste. O `TEACHub` notifica os membros da turma sobre a nova publicação. O `Aluno` visualiza a notificação e acessa a nova publicação. O `Aluno` então adiciona um novo comentário na publicação e o `TEACHub` valida e persiste. O `TEACHub` então notifica os membros da turma sobre o novo comentário.

3.3 IMPLEMENTAÇÃO

Essa seção apresenta as técnicas e ferramentas utilizadas para o desenvolvimento do sistema. Além disso, é relatado sobre o desenvolvimento do sistema, no qual são apresentados trechos de códigos-fontes das principais rotinas desenvolvidas. Por fim, é apresentada a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Nesta seção são apresentadas as técnicas e ferramentas utilizadas para a construção do sistema. Para o desenvolvimento desde trabalho foram utilizadas as ferramentas:

a) Atom Editor de texto;
b) MySQL como banco de dados;
c) Astah Community para criação de diagramas;
d) Amazon AWS para hospedar aplicação para testes com usuários;
e) Bitbucket como repositório para projetos Git;
f) NodeJS como plataforma de desenvolvimento;
g) Quasar Framework como *framework* para desenvolvimento da aplicação web e do aplicativo móvel;
h) VueJS como biblioteca para manipulação de telas;
i) SequelizeJS como *framework* ORM;
j) ExpressJS como *framework* para tratar requisições HTTP;
k) SocketIO como *framework* para tratar eventos WebSocket.

O módulo servidor do aplicativo foi desenvolvido utilizando NodeJS como plataforma para desenvolvimento. NodeJS é uma plataforma de desenvolvimento que possui o Javascript como linguagem de programação e utiliza o interpretador Javascript V8, o mesmo utilizado pelo navegador Google Chrome. A plataforma é utilizada na criação de servidores *web*, aplicações que lidam com recursos do sistema operacional e até ferramentas para auxiliar o

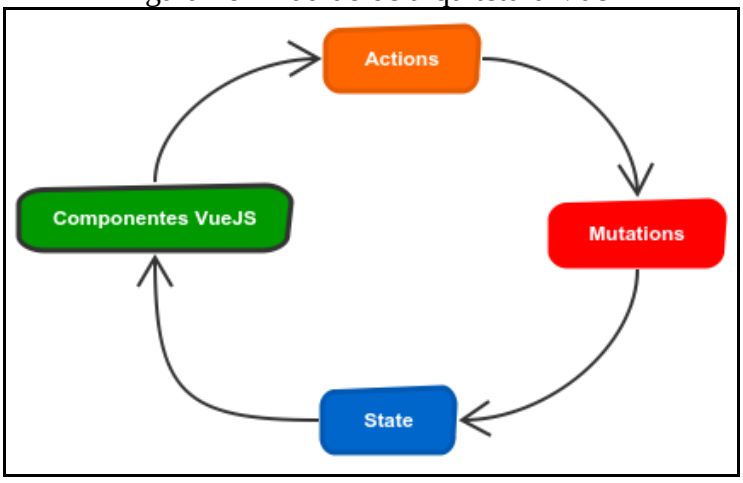
desenvolvimento de software, como editores de texto ou ferramentas de interface de linha de comando.

A parte *front-end* do aplicativo foi desenvolvida para ser embarcada em um aplicativo nativo e também para ser executado em um navegador de forma responsiva. Para o desenvolvimento foi utilizado o Quasar Framework. Este *framework* contém vários componentes de tela já prontos, como barras de navegações, botões, menus laterais, listas, entre outros seguindo padrões de *design* de aplicativo Android e iOS. No momento de gerar um aplicativo para dispositivos móveis com o Quasar Framework é selecionado em qual dispositivo o aplicativo irá rodar. Assim o *framework* consegue construir o aplicativo seguindo os padrões de *design* do respectivo dispositivo.

Os componentes de tela feitos no Quasar Framework são todos destinados para usar no aplicativo com a biblioteca VueJS. O VueJS é uma biblioteca de manipulação de interface escrita em Javascript que facilita o desenvolvimento de Single Page Applications (SPA). No trabalho desenvolvido foi utilizado também as ferramentas VueRouter e Vuex para roteamento das páginas e controle compartilhado das informações entre os componentes de tela, respectivamente.

O Vuex é inspirado pela arquitetura Flux do Facebook, sendo responsável por manter o controle das informações entre os componentes de tela. Na Figura 16 é apresentado o padrão da arquitetura presente no Vuex.

Figura 16 – Padrão de arquitetura Vuex



Fonte: Elaborado pelo autor.

Como representando na Figura 16, o Vuex possui três camadas que se comunicam de forma unidirecional. A camada `state` é responsável por manter as informações a serem renderizadas pelos componentes Vue, por exemplo uma lista com os cursos de um professor. As `actions` são eventos disparados pelos componentes Vue para realizar uma determinada

regra de negócio. É na camada das actions que são feitas as requisições HTTP para buscar ou persistir dados no módulo servidor. As `actions` não podem alterar o estado presente no `state`, então elas disparam as `mutations`. As `mutations` são responsáveis apenas por realizar as mutações no `state`.

Para lidar com os eventos WebSocket foi utilizado o *framework* SocketIO. Este *framework* permite a comunicação bidirecional entre o módulo servidor e o aplicativo por meio de eventos. É utilizado o protocolo WebSocket para o envio dos eventos. Este protocolo deixa aberto um canal entre o módulo servidor e o *front-end* do aplicativo ativo, para que ambas as partes possam se comunicar.

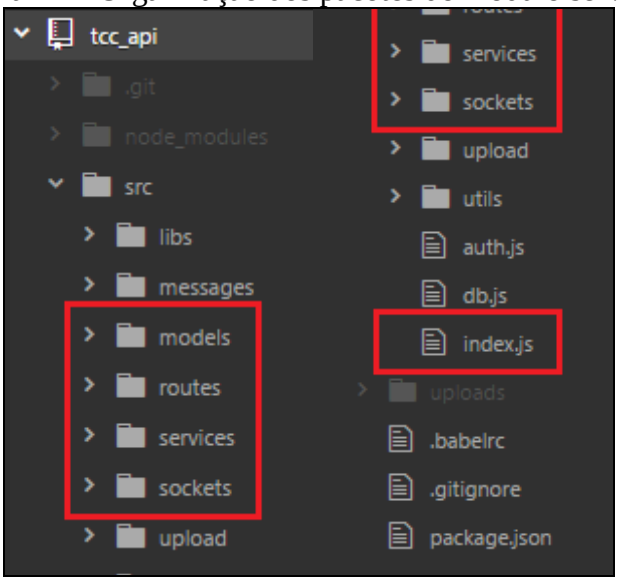
3.3.2 Desenvolvimento

Nesta seção são apresentados trechos relevantes do desenvolvimento da aplicação e do servidor, através de códigos-fontes e exemplos utilizados no trabalho desenvolvido.

3.3.2.1 Módulo servidor

O módulo servidor está organizado em pacotes, como visto é ilustrado na Figura 17. De acordo com a figura, dentro da pasta `src` que é a pasta raiz, destacam-se os pacotes `models`, `services`, `routes` e `sockets`. O pacote `models` contém os modelos de classes. O pacote `services` contém os serviços e regras de negócio. O pacote `routes` possui os pontos de entrada do módulo servidor, ou seja, é responsável por lidar com as requisições HTTP. Por fim, o pacote `sockets` é responsável pelos eventos WebSockets enviados pelo cliente e por manter uma estrutura de dados dos usuários *online*.

Figura 17 – Organização dos pacotes do módulo servidor



Fonte: Elaborado pelo autor.

O arquivo `index.js` presente na raiz do pacote `src` é responsável por carregar os pacotes do módulo servidor, fazer a sincronização dos modelos presentes no pacote `models` com a base de dados e inicializar o servidor. Após inicializado, o módulo servidor recebe requisições HTTP através das rotas definidas no pacote `routes`. As rotas foram desenhadas a partir dos princípios de arquitetura REST. No Quadro 8 pode-se ver um exemplo de rotas para consulta, criação, atualização e remoção dos cursos.

Quadro 8 – Rotas para consulta, criação, atualização e remoção dos cursos

```
1  import * as ErrorMessages from "../messages/errorMessages.js"
2
3  module.exports = app => {
4      const CoursesService = app.services.courses
5
6      app.route("/api/courses")
7          .all(app.auth.authenticate())
8          .get((req, res) =>
9              CoursesService.find(req.user.id)
10             .then(courses => res.json(courses))
11         )
12         .post((req, res) =>
13             CoursesService.save(req.user.id, req.body)
14             .then(course => res.json(course))
15             .catch(error => res.status(412).json(error))
16         )
17
18     app.route("/api/courses/:courseId")
19         .all(app.auth.authenticate())
20         .get((req, res) =>
21             CoursesService.findOne(req.params.courseId)
22             .then(course => res.json(course))
23             .catch(error => res.status(412).json(error))
24         )
25         .delete((req, res) =>
26             CoursesService.remove(req.params.courseId, req.user.id)
27             .then(result => res.json(result))
28             .catch(error => res.status(412).json(error))
29         )
30 }
```

Fonte: Elaborado pelo autor.

O Quadro 8 representa o código-fonte que realiza as rotas das ações do curso. Pode-se visualizar quatro rotas distintas, sendo: listagem de cursos (linha 8), salvar um curso (linha 12), consulta de um curso (linha 20) e remover um curso (linha 25). As rotas de listagem de cursos e de salvar um curso possuem o mesmo ponto de entrada, sendo ele a URL "/api/courses". A diferença entre elas é o método da requisição utilizado. A primeira rota é feita esperando uma requisição HTTP com método GET, já a segunda com método POST.

As rotas de consulta de um curso e remoção de um curso possuem um ponto de entrada que recebe um parâmetro, sendo a identificação do curso. A diferença também está apenas

nos métodos HTTP utilizados, enquanto a primeira espera uma requisição com método GET a outra espera uma requisição com método DELETE.

Todas as rotas representadas no Quadro 8 realizam autenticação de usuário como exemplificado nas linhas 7 e 19. Após autenticado o usuário, as rotas chamam as respectivas funções do serviço de cursos, definido na linha 4. O serviço de cursos possui algumas funções em sua assinatura. O Quadro 9 apresenta a função `save`, responsável por salvar um curso de um professor e suas funções auxiliares de criação (`create`) e atualização (`update`).

Quadro 9 – Função salvar do serviço de cursos

```
55  let save = function(teacherId, course) {
56      if (course.id) {
57          return update(course.id, teacherId, course)
58      }
59      return create(teacherId, course)
60  }
61
62  let create = function(teacherId, course) {
63      course.teacherId = teacherId
64      return Courses.create(course)
65  }
66
67  let update = function(id, teacherId, course) {
68      return findOneThrowingException(id, teacherId)
69          .then(result => {
70              return Courses.update(course, {
71                  where: {
72                      id,
73                      teacherId
74                  }
75              })
76          })
77  }
```

Fonte: Elaborado pelo autor.

A função `save` descrita no Quadro 9 recebe as variáveis `teacherId` e `course`. Se o curso possuir uma identificação então é executado a função `update`, senão é executado a função `create`. Na função `create` é apenas adicionado a identificação do professor e executado a função `create` da constante `Courses`. Esta função cria um novo curso na base de dados.

Na função `update` é verificado se existe um curso com aquela identificação e se pertence aquele professor. A função `findOneThrowingException` (linha 68) busca o curso

identificado pelos parâmetros `id` e `teacherId`. Se o curso existir a função `update` da constante `Courses` é executada passando o objeto do curso atualizado e uma cláusula para atualizar apenas o curso recebido. Se o curso não existir, é lançada uma exceção.

No Quadro 9 tem-se o uso de uma constante `Courses` na manutenção dos cursos nas linhas 64 e 70. A constante `Courses` está presente no pacote `models` e é definida conforme o Quadro 10.

Quadro 10 – Modelo Curso

```
1  module.exports = (sequelize, DataType) => {
2    const Courses = sequelize.define("Courses", {
3      id: {
4        type: DataType.INTEGER,
5        primaryKey: true,
6        autoIncrement: true
7      },
8      name: {
9        type: DataType.STRING,
10       allowNull: false,
11       validate: {
12         notEmpty: true
13       }
14     },
15   › description: {…
19   › active: {…
23   › maxNumberOfStudents: {…
27   › remote: {…
31   › uf: {…
35   › city: {…
39   }, {
40     classMethods: {
41   ›   associate: (models) => {…
55     }
56   });
57   return Courses;
58 };
```

Fonte: Elaborado pelo autor.

O Quadro 10 apresenta a utilização da variável `sequelize` para definir o modelo cursos. A partir do momento em que é executada a função `sequelize.define` (linha 2), o *framework* SequelizeJS cria a tabela de cursos na base de dados se ela não existir. Em destaque estão as colunas da tabela de cursos: `id` (linha 4) e `name` (linha 8). As colunas possuem alguns atributos, como por exemplo: `type` (linha 4), `primaryKey` (linha 5), `autoIncrement` (linha 6), `allowNull` (linha 10) e `validate` (linha 11).

Todo modelo possui algumas funções definidas do *framework* SequelizeJS, como por exemplo: `find`, `create`, `update` e `destroy`. Todas as funções específicas a um modelo são

definidas no atributo `classMethods` (linha 40). A função `associate` (linha 41) está em destaque no Quadro 11.

Quadro 11 – Método `associate` do modelo Courses

```
41      associate: (models) => {
42        Courses.belongsTo(models.Users, {
43          foreignKey: 'teacherId'
44        })
45        Courses.hasMany(models.Classes, {
46          foreignKey: 'courseId',
47          onDelete: 'cascade'
48        })
49        Courses.hasMany(models.CourseRatings, {
50          foreignKey: 'courseId',
51          as: "ratings",
52          onDelete: 'cascade'
53        })
54      }
```

Fonte: Elaborado pelo autor.

No Quadro 11 tem-se a definição de relacionamento da tabela de cursos com outras tabelas presentes no banco de dados. É possível definir a cardinalidade com as funções `belongsTo` (linha 42), `hasMany` (linhas 45 e 49) e outras definidas na documentação do SequelizeJS. A função `belongsTo` (linha 42) define que o modelo `Courses` possui um atributo `teacherId` (linha 43) que é uma *foreign-key* da tabela `Users` (linha 42). A função `hasMany` (linha 45) define que o modelo `Courses` é referenciado através da foreign-key `courseId` (linha 46) na tabela `Classes` (linha 45).

A próxima seção trata a estrutura de pacotes e padrões de desenvolvimento aplicados no *front-end* do aplicativo.

3.3.2.2 *Front-end* do aplicativo

O *front-end* do aplicativo foi desenvolvido utilizando o *framework* Quasar, que por sua vez foi construído para ser utilizado com a biblioteca VueJS. A biblioteca permite ao desenvolvedor escrever códigos com a extensão vue e esses arquivos possuem o HTML, CSS e Javascript do componente de tela. No Quadro 12 é apresentada a estrutura de um componente VueJS.

Quadro 12 – Estrutura de um componente VueJS

```
1  <style>
2  </style>
3
4  <template>
5  </template>
6
7  <script>
8  export default {}
9  </script>
```

Fonte: Elaborado pelo autor.

Um componente VueJS é divido nas camadas `style`, `template` e `script`. No `style` vão todos os estilos pertinentes ao componente. O `template` contém a marcação HTML referente ao componente. Na *tag* `script` fica o Javascript responsável por lidar com a interação do usuário com o componente. No Quadro 13 pode-se visualizar parte do `template` do componente de manutenção de cursos.

Quadro 13 – Parte do `template` do componente de manutenção de cursos

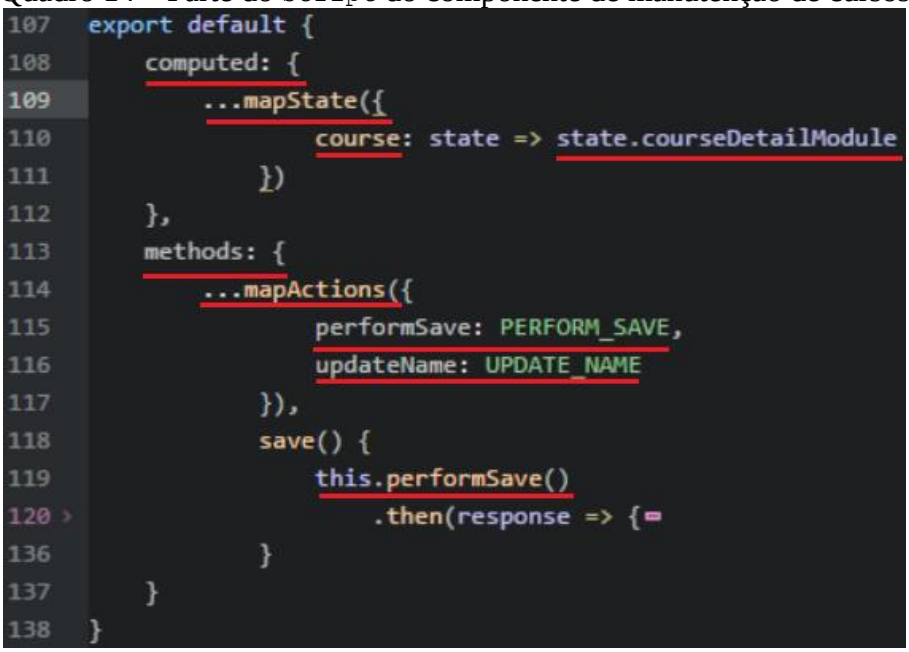
```
26 <q-layout>
27     <div slot="header" class="toolbar">
28         <button @click="save">
29             <i>done</i>
30         </button>
31     </div>
32     <div id="course-wrapper">
33         <div class="layout-padding">
34             <div class="row">
35                 <div class="floating-label">
36                     <input required class="full-width" :value="course.name" @change="updateName">
37                     <label>Nome</label>
38                 </div>
39             </div>
40         </div>
41     </div>
42 </q-layout>
```

Fonte: Elaborado pelo autor.

O Quadro 13 demonstra o uso da *tag* `<q-layout>` na linha 26 disponibilizada pelo Quasar Framework para definir o leiaute do aplicativo. Logo em seguida, na linha 28 é criado um botão de salvar, responsável por executar a função `save` definida na camada `script` do componente. Na linha 36 é feito o mapeamento do valor do campo de texto referente ao nome do curso. Também é de responsabilidade do campo de texto executar a função `updateName` quando o valor dele é alterado.

Ambas funções `save` e `updateName` e a variável `course` estão disponíveis no `template` graças à definição na camada `script`. O Quadro 14 demonstra parte da camada `script` do componente de manutenção de cursos.

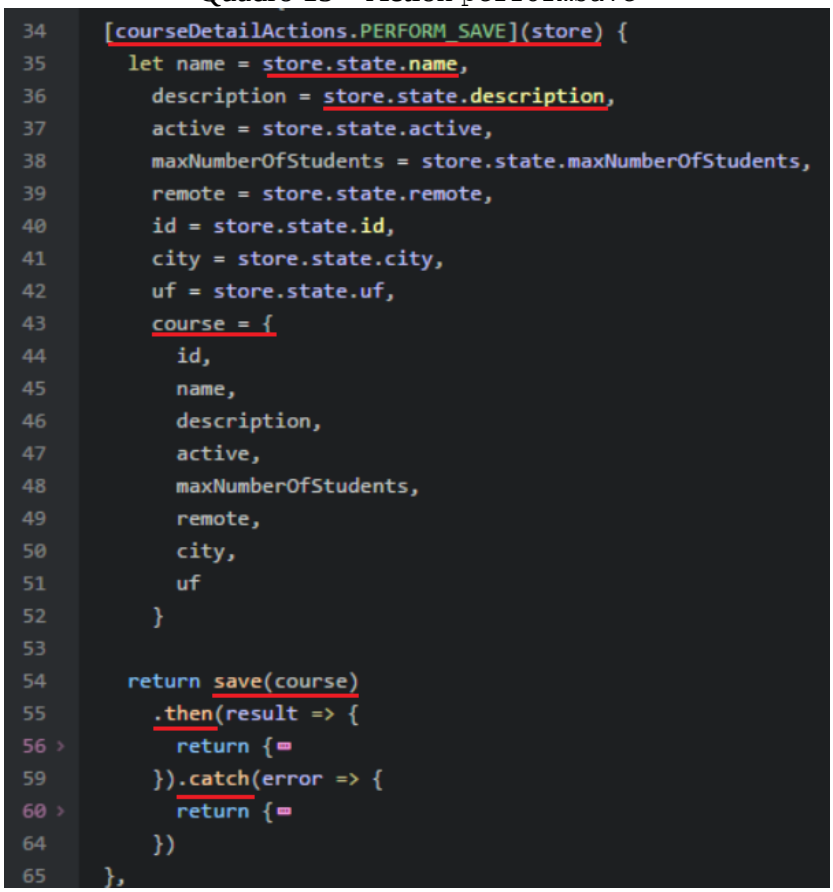
Quadro 14 – Parte do `script` do componente de manutenção de cursos



Fonte: Elaborado pelo autor.

No Quadro 14 tem-se a definição do objeto `computed` utilizando o método `mapState` para mapear valores presentes no `state` e disponibiliza-los no `template` do componente. No objeto `methods` acontece algo parecido, mas neste caso ele está mapeando `actions` e definindo um método `save`. A partir do momento que o componente é criado, essas variáveis e funções estão disponíveis para apresentar informações e lidar com a interação do usuário. A função `performSave` faz parte das `actions` do módulo de cursos e é descrita no Quadro 15.

Quadro 15 – Action `performSave`



```
34  [courseDetailActions.PERFORM_SAVE](store) {
35    let name = store.state.name,
36      description = store.state.description,
37      active = store.state.active,
38      maxNumberOfStudents = store.state.maxNumberOfStudents,
39      remote = store.state.remote,
40      id = store.state.id,
41      city = store.state.city,
42      uf = store.state.uf,
43      course = {
44        id,
45        name,
46        description,
47        active,
48        maxNumberOfStudents,
49        remote,
50        city,
51        uf
52      }
53
54    return save(course)
55      .then(result => {
56 ›      return {…
59      }).catch(error => {
60 ›      return {…
64      })
65  },
```

Fonte: Elaborado pelo autor.

De acordo com o Quadro 15, todas as `actions` recebem o parâmetro `store` como primeiro argumento (linha 34) e os demais argumentos são optativos (linha 34), variando de `action` para `action`. Através do atributo `state` presente no parâmetro `store` é possível recuperar os dados atualizados e executar a função `save`, responsável por fazer uma requisição HTTP ao módulo servidor. Após o módulo servidor retornar à requisição, a função passada no método `then` é chamada se o retorno é positivo. Se o retorno for negativo, ou seja, aconteceu algum erro, a função passada no método `catch` é chamada.

Todas as telas do aplicativo foram desenvolvidas seguindo esse padrão de arquitetura. Na próxima seção é apresentado como foi aplicado esse padrão no *chat* do aplicativo e também como o módulo servidor suporta os eventos WebSockets.

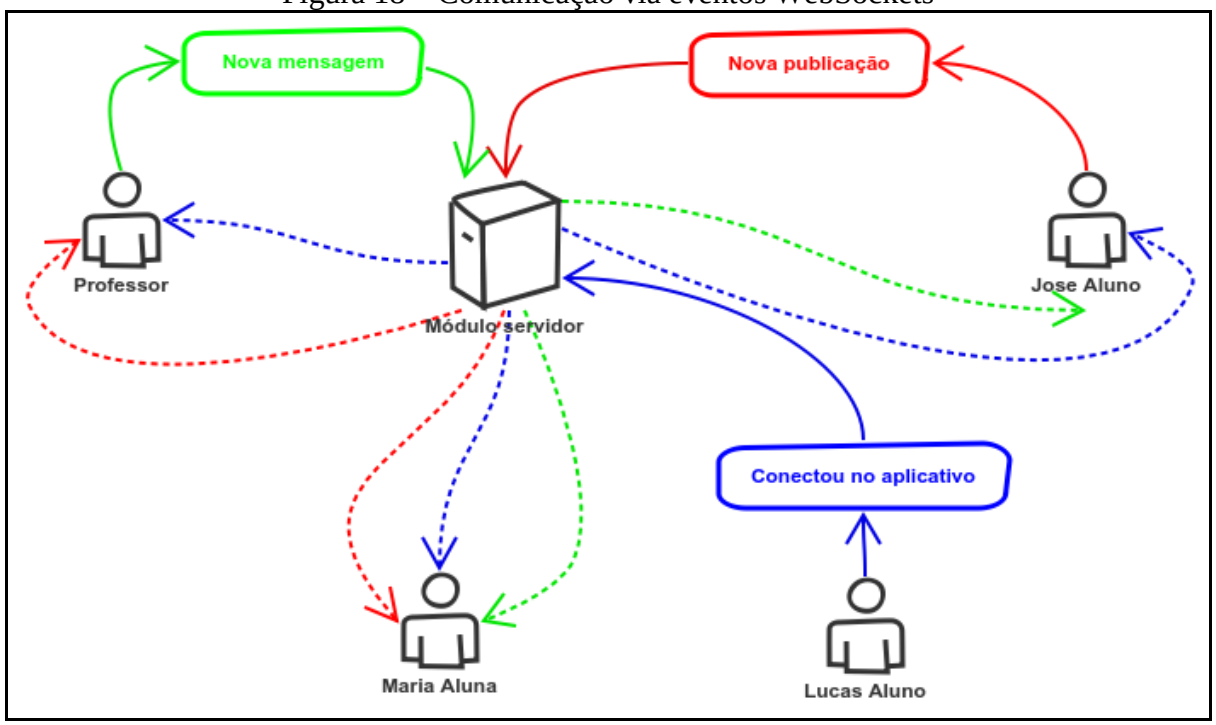
3.3.2.3 WebSockets com SocketIO

Uma parte importante do aplicativo desenvolvido é o *chat*. Esta funcionalidade foi construída utilizando o *framework* SocketIO, responsável por lidar com os eventos WebSockets. Através destes eventos foi possível desenvolver o *chat* para ele ser em tempo

real, ou seja, a cada mensagem, o usuário destino automaticamente recebe a mensagem sem precisar atualizar os dados do aplicativo.

Além do *chat*, as funcionalidades de receber notificações e visualizar usuários *online* também foram desenvolvidas com o auxílio do *framework* SocketIO. A Figura 18 demonstra a comunicação entre os usuários conectados no aplicativo e como funciona os eventos WebSockets.

Figura 18 – Comunicação via eventos WebSockets



Fonte: Elaborado pelo autor.

Na Figura 18 tem-se um exemplo de comunicação via eventos WebSockets que ocorre no aplicativo. As linhas tracejadas são os eventos emitidos pelo módulo servidor para os usuários conectados após determinadas ações. A cada nova mensagem no *chat* da turma, por exemplo, o módulo servidor salva as mensagens no banco de dados e depois dispara eventos para os usuários da turma que estão conectados, informando sobre a nova mensagem. O mesmo acontece para novas publicações e novos comentários. A partir do momento que um usuário acessa o aplicativo, o módulo servidor atualiza a lista de usuários conectados e dispara um evento para todos os usuários informando que um novo usuário está conectado.

Para exemplificar o uso do SocketIO no aplicativo é analisada a funcionalidade de *chat*. O componente de tela responsável pelo *chat* também foi desenvolvido utilizando o VueJS e o seu *template* é mostrado no Quadro 16.

Quadro 16 – Template do componente de tela referente ao *chat*

```
9   <q-layout>
10
11    <div slot="header" class="toolbar">
12      <button @click='$router.push("/conversations")'>
13              <i>arrow_back</i>
14          </button>
15      <q-toolbar-title>
16        {{contact.name}}
17      </q-toolbar-title>
18    </div>
19
20    <sidebar :show-menu-floating-button="false"></sidebar>
21
22    <div id="room-wrapper">
23      <messages-list :messages="messages" :show-from="false"></messages-list>
24    </div>
25    <message-textarea :send="sendMessage"></message-textarea>
26
27  </q-layout>
```

Fonte: Elaborado pelo autor.

Em destaque no Quadro 16 estão o componente de tela responsável por renderizar as mensagens já enviadas, o componente responsável pelo campo de texto e o botão de enviar, respectivamente. Observa-se a função `sendMessage` (linha 26) destacada no componente `message-textarea`, responsável por emitir um evento WebSocket para o módulo servidor quando o usuário envia uma nova mensagem. No Quadro 17 tem-se a função `sendMessage`.

Quadro 17 – Função `sendMessage`

```
65    sendMessage(content) {
66      this.$socket.emit("direct-message", {
67        to: this.contactId,
68        content
69      })
70    }
```

Fonte: Elaborado pelo autor.

No Quadro 17, na linha 65 tem-se a função `sendMessage` que recebe o parâmetro `content` e emite o evento `direct-message` contendo as informações de usuário destino e conteúdo da mensagem, representadas respectivamente por `to` e `content`. No Quadro 18 tem-se a função que lida com esse evento no módulo servidor.

Quadro 18 – Função responsável por receber evento de nova mensagem direta

```
19  socket.on("direct-message", app.sockets.authentication(io, socket, (error, user, message) => {
20      if (error) return;
21      DirectMessagesService.save(user.id, message.to, message.content)
22          .then(directMessage => {
23              emitToUsers('direct_message', [user.id, message.to], directMessage)
24          })
25  }))
```

Fonte: Elaborado pelo autor.

Como demonstrado no Quadro 18, é registrada uma função responsável por receber os eventos `direct-message` e processá-los. Esta função utiliza um facilitador para autenticar o usuário que enviou o evento, representado pela função `app.sockets.authentication`. O serviço `DirectMessagesService` é responsável por salvar a mensagem e após a mensagem ser salva, é emitido o evento `direct_message` ao usuário remete e ao usuário destinatário.

A parte *front-end* do aplicativo possui um módulo responsável por escutar esse evento e atualizar a listagem de mensagens. Inicialmente o módulo busca todas as mensagens a cada mensagem nova recebida. Esse comportamento foi definido para facilitar o desenvolvimento do protótipo, mas sabe-se que não é performático. O módulo é representado no Quadro 19.

Quadro 19 – Módulo responsável atualizar as mensagens a cada mensagem nova recebida

```
10  loadConversations(store) {
11    return loadConversations()
12      .then(response => {
13        let all = response.data
14        store.commit('load_conversations', {
15          all
16        })
17      })
18  },
19  socket_directMessage(store) {
20    store.dispatch('loadConversations')
21  },
```

Fonte: Elaborado pelo autor.

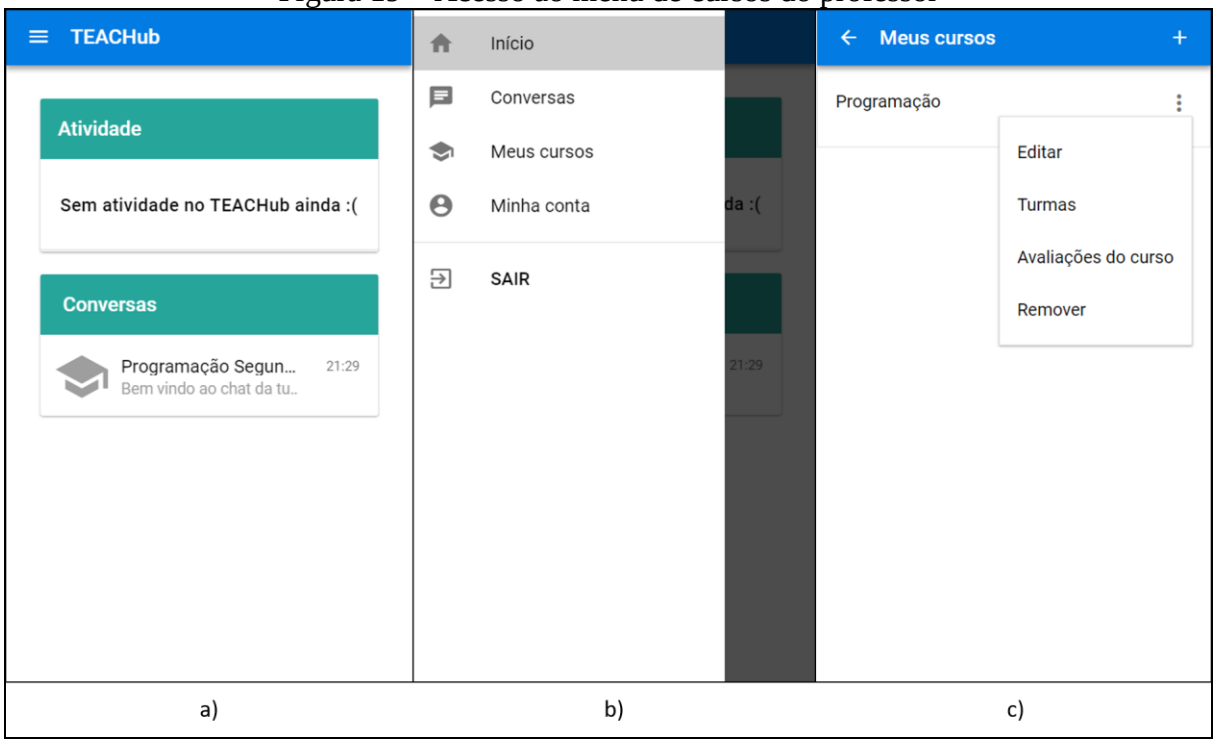
A função `socket_directMessage` (linha 19) é chamada após o módulo servidor emitir o evento `direct_message`. Esta função apenas chama a função `loadConversations` definida na linha 10, responsável por buscar as conversas e atualizar as mensagens do módulo. Na linha 14 do Quadro 19, é atualizada a lista de mensagens e todos os componentes de tela que dependem dessa informação se atualizam automaticamente.

3.3.3 Operacionalidade da implementação

Essa seção demonstra o fluxo apresentado no diagrama de atividades, com o propósito de visualizar a operacionalidade da implementação. O diagrama de atividades apresentado na

seção 3.2.4 engloba o professor fazer um convite a um novo aluno para participar da sua turma, assim como o uso de funcionalidades relacionadas ao mural de publicações da turma. O primeiro passo é o professor entrar no aplicativo e acessar o menu de cursos, como demonstrado nas etapas da Figura 19.

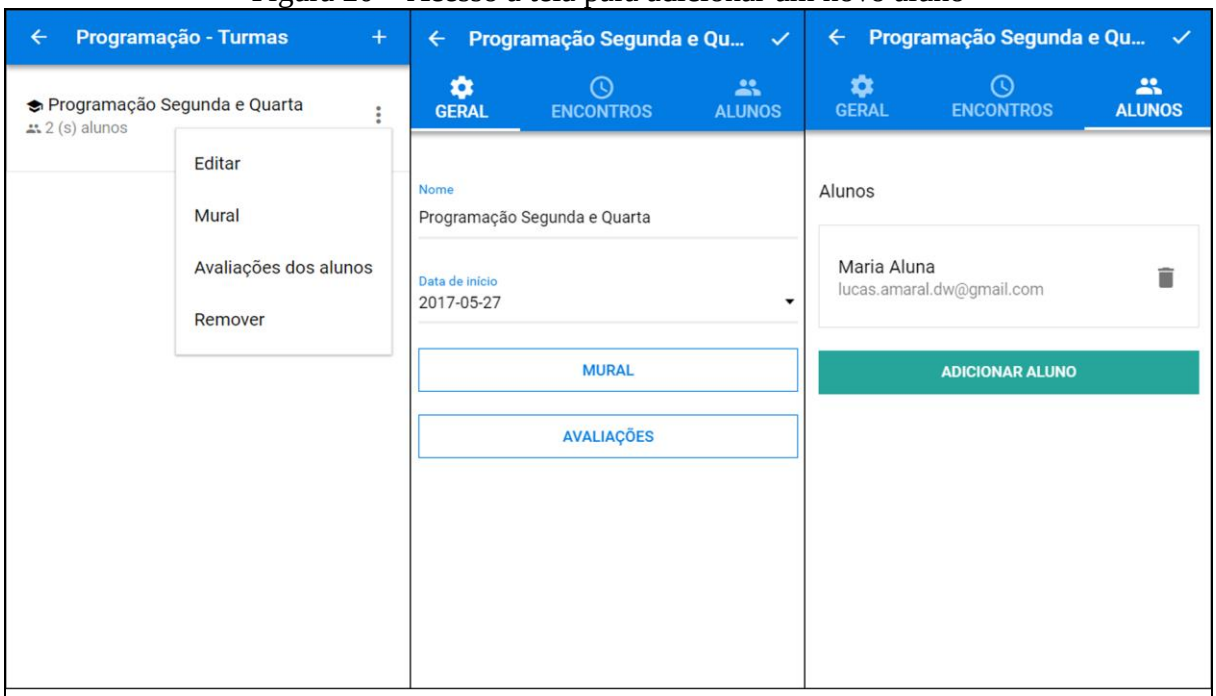
Figura 19 – Acesso ao menu de cursos do professor



Fonte: Elaborado pelo autor.

As etapas descritas na Figura 19 representam, o acesso a tela inicial do aplicativo (Figura 19 a), o acesso ao menu principal do aplicativo (Figura 19 b) e o acesso a listagem de cursos (Figura 19 c). Após selecionar o item “turmas” do menu de contexto do curso, o professor acessa o menu das turmas do curso e seleciona uma turma para editar (Figura 20 a). Posteriormente o professor é redirecionado para tela de edição de uma turma, onde ele seleciona o menu alunos (Figura 20 b). Então ele visualiza os alunos da turma e seleciona a opção de adicionar aluno (Figura 20 c).

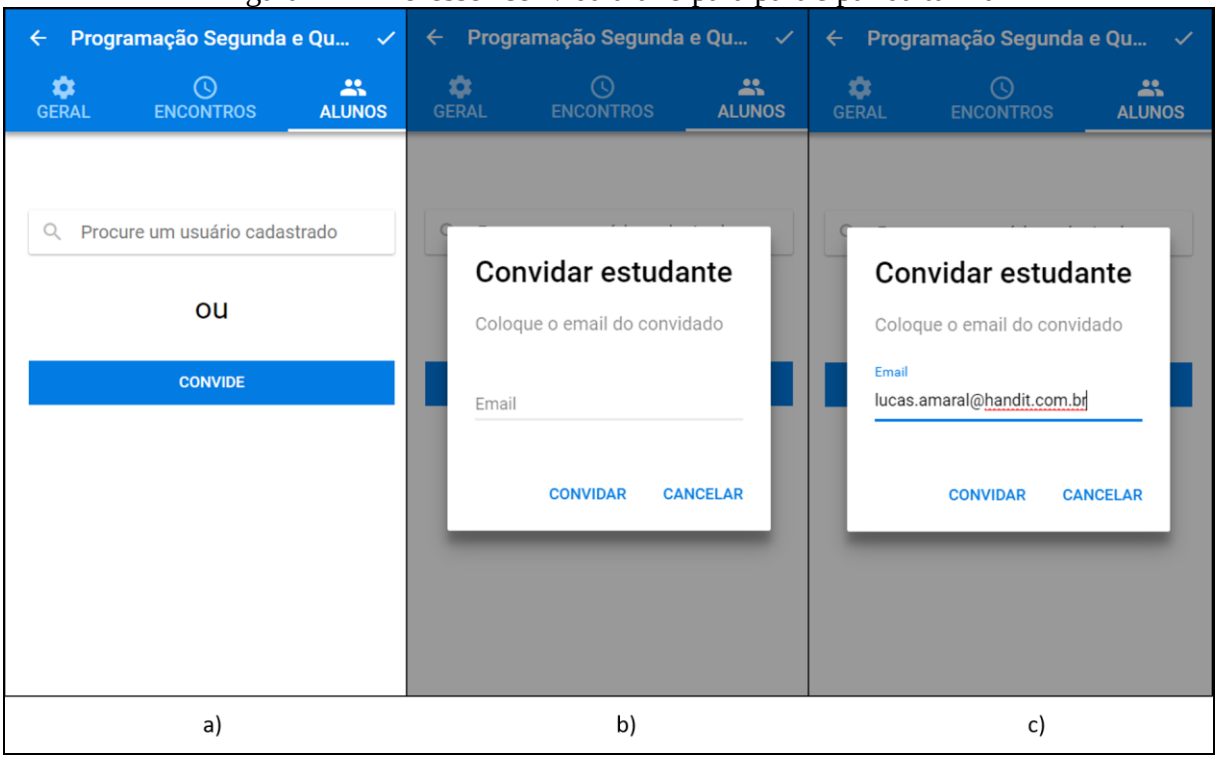
Figura 20 – Acesso a tela para adicionar um novo aluno



Fonte: Elaborado pelo autor.

Após realizar os passos descritos na Figura 20, o professor é redirecionado para tela de adição de alunos e ele seleciona a opção de convidar aluno (Figura 21 a). O aplicativo mostra um campo de texto para o professor preencher o e-mail do aluno convidado (Figura 21 b). O professor preenche o e-mail e o aplicativo envia um convite ao novo aluno (Figura 21 c).

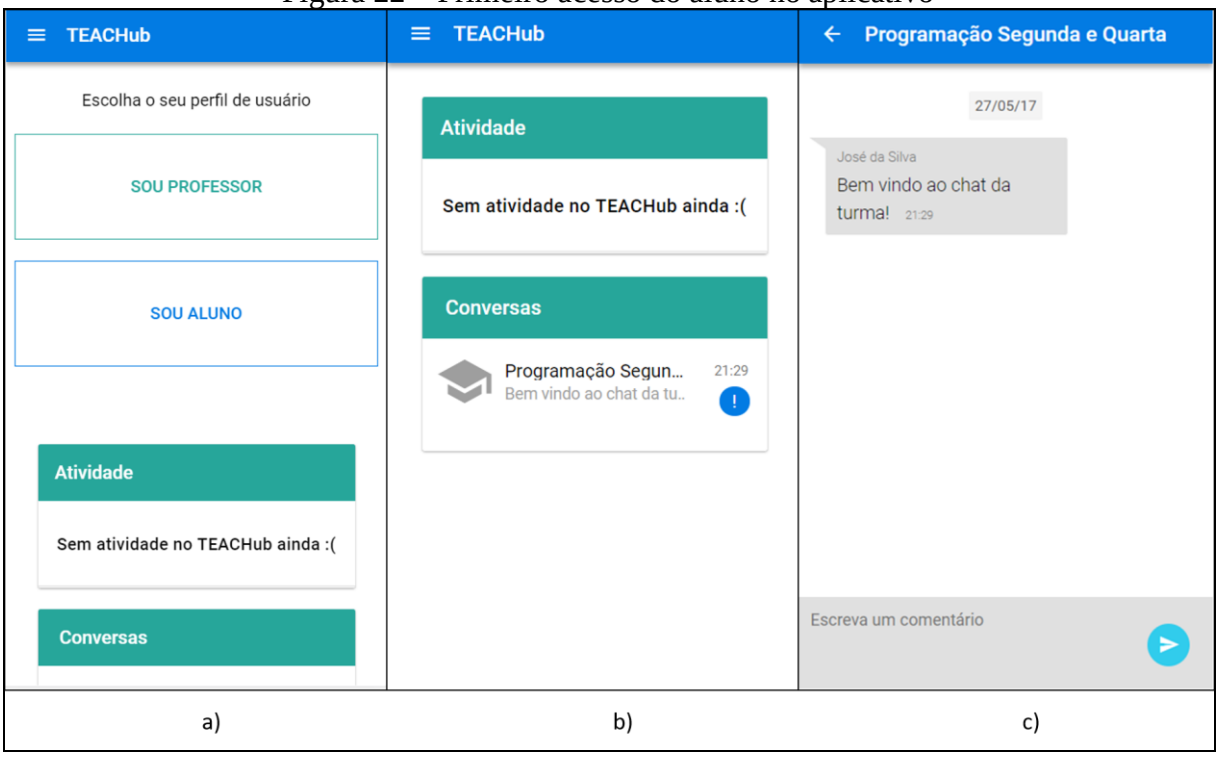
Figura 21 – Professor convida aluno para participar da turma



Fonte: Elaborado pelo autor.

O aluno então recebe um e-mail com um endereço para acessar o aplicativo e uma chave de acesso para finalizar o cadastro do seu usuário. Após terminado a validação do convite, o aluno acessa o aplicativo pela primeira vez, como demonstrado na Figura 22.

Figura 22 – Primeiro acesso do aluno no aplicativo

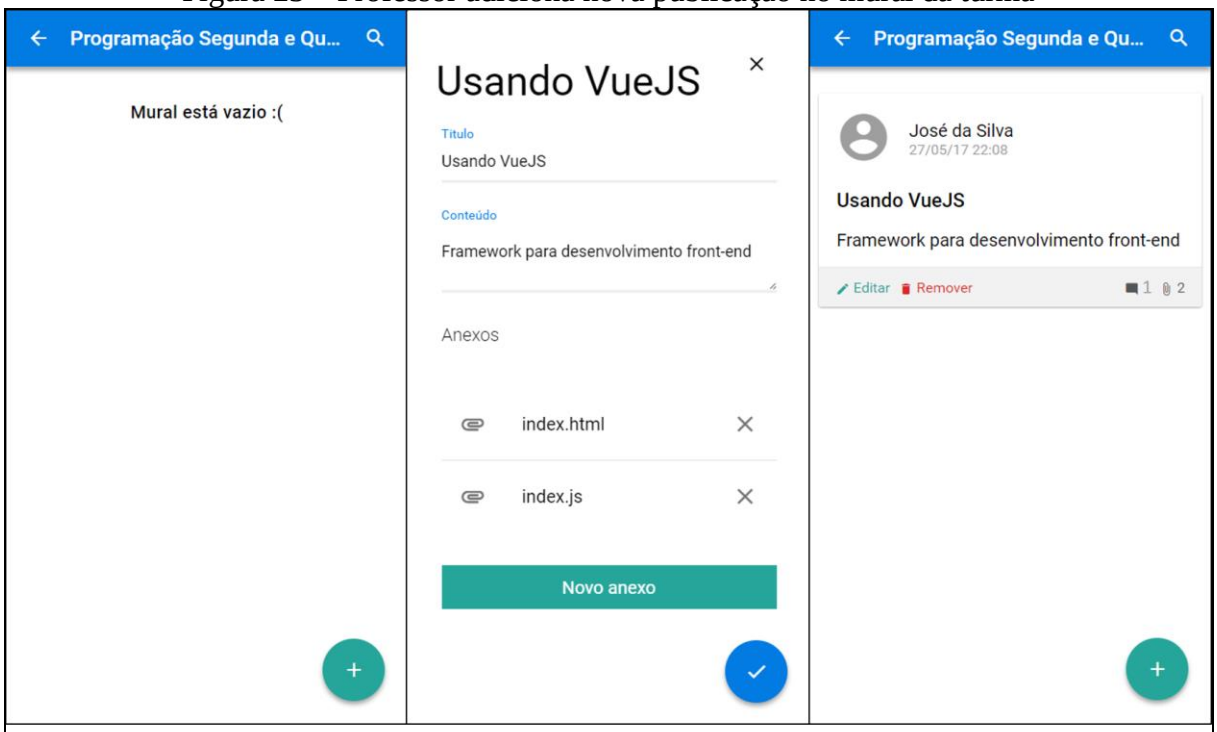


Fonte: Elaborado pelo autor.

Na Figura 22 pode-se visualizar três etapas que o aluno pode fazer ao acessar o aplicativo pela primeira vez. Na primeira etapa ele seleciona o perfil de usuário que ele deseja (Figura 22 a), no caso o perfil de aluno. Após selecionado o perfil, ele verifica a nova mensagem no grupo da turma em que ele participa (Figura 22 b). Dessa forma, ele pode acessar o *chat* do grupo e visualizar as mensagens enviadas (Figura 22 c).

Tendo o aluno adicionado, o professor pode interagir com ele. A comunicação pode ser feita através de *chat* e de mural de publicações. Na Figura 23, o professor acessa o mural de publicações da turma (Figura 23 a), cria uma nova publicação (Figura 23 b) e visualiza o mural, agora com uma publicação criada (Figura 23 c).

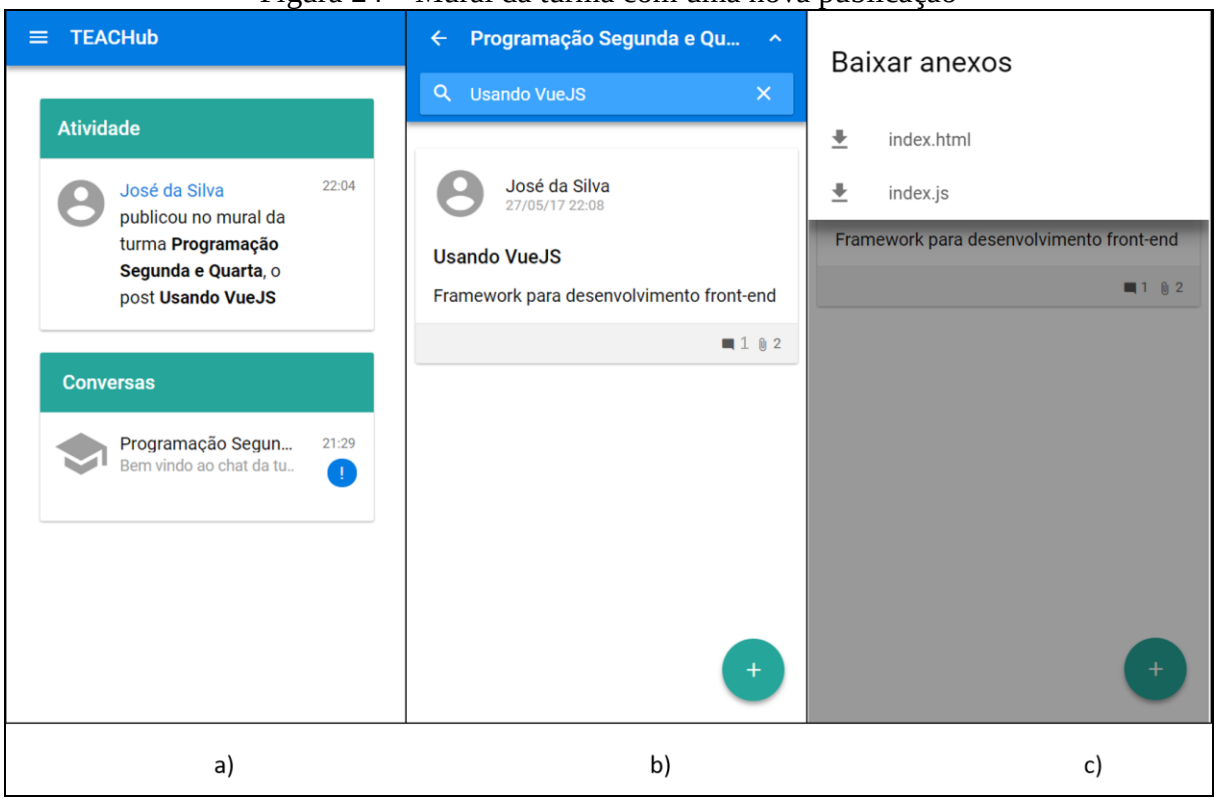
Figura 23 – Professor adiciona nova publicação no mural da turma



Fonte: Elaborado pelo autor.

Após seguir os passos descritos na Figura 23 e criar a publicação, o aluno recebe uma notificação que uma publicação foi criada no mural, como demonstra a Figura 24 (a).

Figura 24 – Mural da turma com uma nova publicação

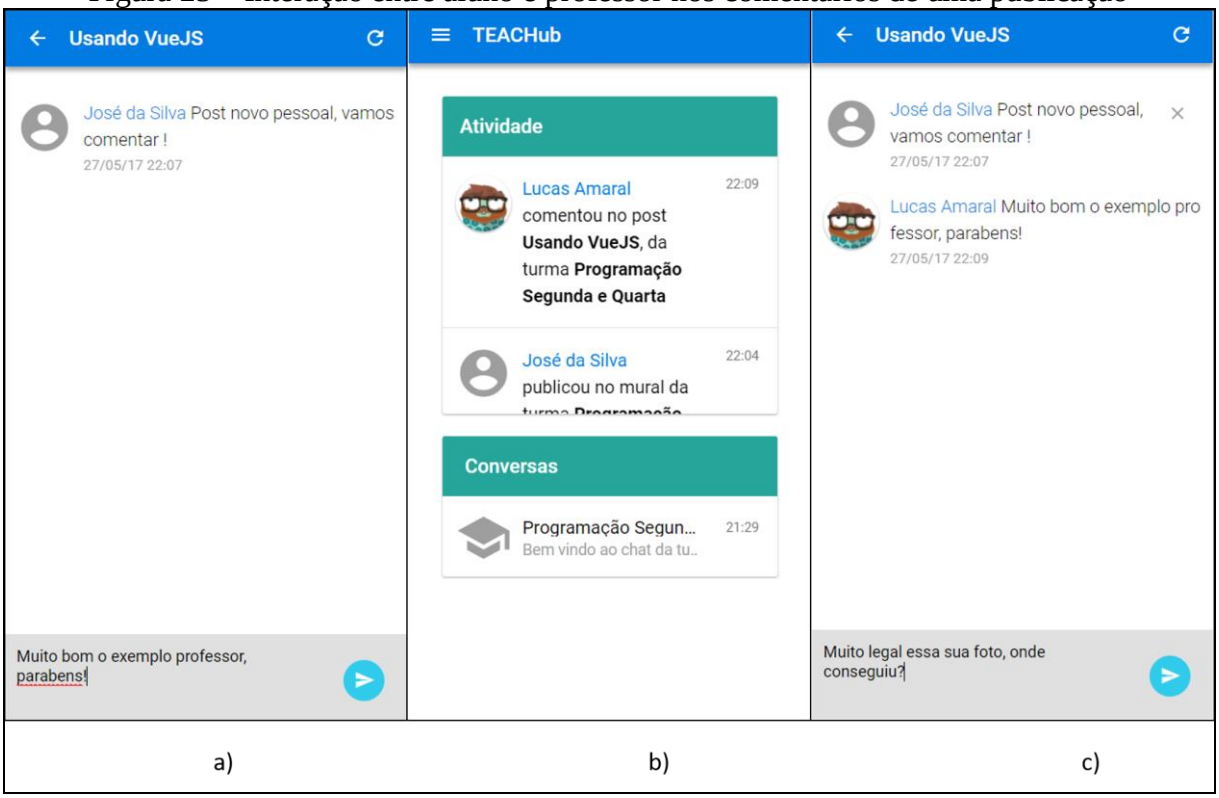


Fonte: Elaborado pelo autor.

Na Figura 24 o aluno recebe a notificação da nova publicação recebida (Figura 24 a) e ele então pode acessar o mural de publicações (Figura 24 b). No mural, cada publicação está representada com um título, conteúdo, autor, data da última atualização, quantidade de comentários e quantidade de anexos. Clicando no ícone de anexos, representado por um clipes, o usuário pode salvar os anexos presentes na publicação (Figura 24 c).

A Figura 25 demonstra a interação entre o aluno e professor através dos comentários de uma publicação. Na primeira etapa (Figura 25 a) o aluno acessa os comentários e verifica que já existe um comentário do professor. O aluno então adiciona um novo comentário. Na segunda etapa (Figura 25 b), o professor recebe uma notificação referente ao novo comentário na publicação. Na terceira etapa (Figura 25 c), o professor visualiza o novo comentário e adiciona um novo comentário.

Figura 25 – Interação entre aluno e professor nos comentários de uma publicação



Fonte: Elaborado pelo autor.

Além das funcionalidades apresentadas, o aplicativo ainda permite que o aluno avalie o professor e vice-versa. Ao realizar uma avaliação, o usuário avaliado recebe uma notificação, podendo assim visualizar seu desempenho.

A operacionalidade demonstrada foi utilizada como base para uma pesquisa de interação do usuário com o aplicativo e os resultados são demonstrados na próxima seção.

3.4 RESULTADOS E DISCUSSÕES

Os trabalhos correlatos ajudaram a identificar qual a melhor forma de formatar a experiência do usuário no aplicativo e quais funcionalidades são mais importantes para manter a interação e comunicação do professor com os alunos. As ferramentas Superprof e Profes serviram de base para construção das regras de negócio, sendo estes mais voltados para a colaboração. Estes dois trabalhos possuem suas interfaces apenas para navegadores *web*, então o aplicativo desenvolvido teve que adaptar algumas ideias retiradas desses trabalhos para o uso responsivo e móvel. O Quadro 20 apresenta a relação de características presentes nos trabalhos correlatos e no aplicativo desenvolvido.

Quadro 20 – Relação de características presentes nos trabalhos correlatos e no aplicativo desenvolvido

Característica	**iProfe**	**Profes**	**Superprof**	**TEACHub**
Aplicativo Móvel	Sim	Não	Não	Sim
Aplicativo *Web*	Não	Sim	Sim	Sim
Anúncio de Cursos	Não	Sim	Sim	Não
Compartilhamento de Recursos/Arquivos	Não	Sim	Não	Sim
Chat professor/alunos	Não	Sim	Parcial	Sim
Agenda/Calendário	Parcial	Sim	Parcial	Parcial
Controle de pagamentos	Parcial	Sim	Sim	Não
Suporte aula *on-line*	Não	Sim	Não	Não
Gestão de Alunos	Sim	Sim	Não	Sim
Gestão de Aulas	Não	Sim	Não	Sim
Notificações/Avisos do Professor	Não	Sim	Sim	Sim
Mural de mensagens / notificações	Não	Sim	Sim	Sim

Fonte: Elaborado pelo autor.

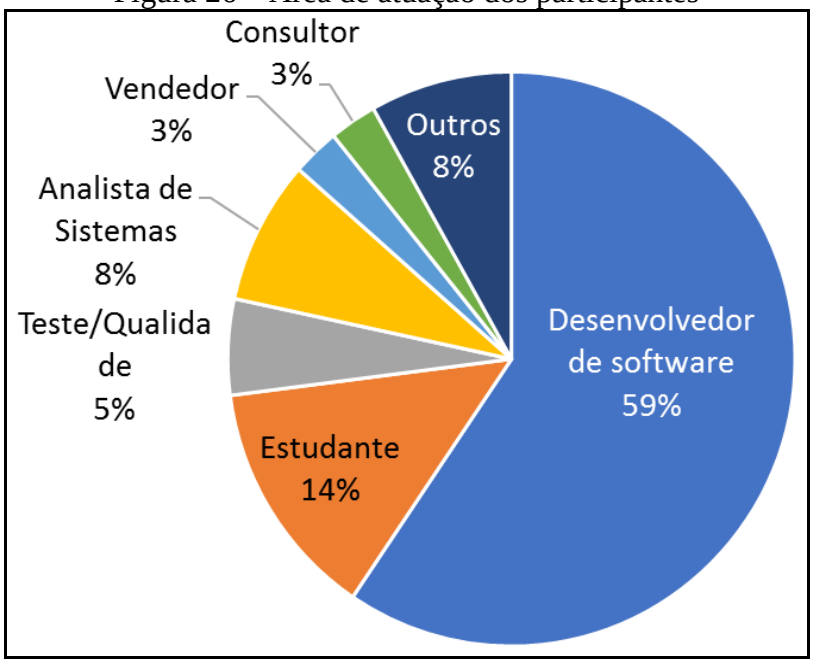
Percebe-se através do Quadro 20 a carência de funcionalidades de divulgação, venda e suporte a aulas *online* no aplicativo desenvolvido. Mesmo não contemplado no escopo do projeto, estas funcionalidades são passíveis de implementação em trabalhos futuros. O aplicativo conseguiu reunir as funcionalidades pertinentes a gestão das aulas particulares e a colaboração entre o professor e os alunos. Além de ser acessível por aplicativo móvel e *web*.

A característica “Agenda/Calendário” apresentada no Quadro 20 foi definida como atende parcialmente no aplicativo desenvolvido, pois apenas mostra os dias que possuem aula, assim como seus horários. Mas não é possível analisar de forma mensal, cancelar e remarcar aulas.

Para testar a usabilidade e operacionalidade do aplicativo foi realizado um questionário com 37 alunos de duas turmas de programação *web* da orientadora deste trabalho. Os participantes com idade entre 20 e 24 anos representam aproximadamente 65% do total de participantes, seguidos por 22% com idade entre 15 e 19 anos. Os 13% restantes são

representados por participantes com mais de 25 anos. A relação das áreas de atuação dos participantes da pesquisa está disposta na Figura 26.

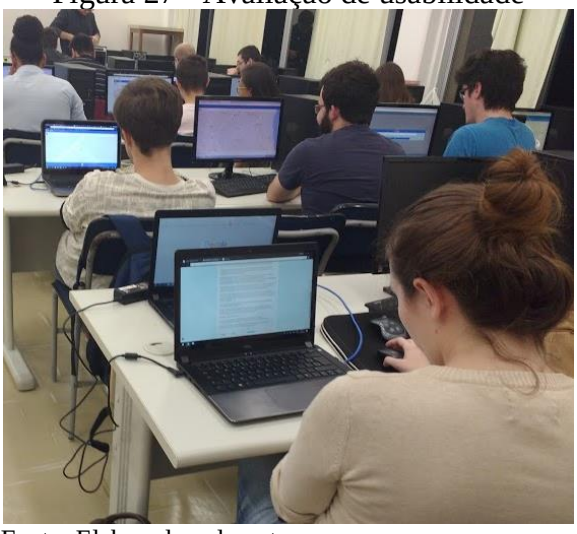
Figura 26 – Área de atuação dos participantes



Fonte: Elaborado pelo autor.

Antes de começar a pesquisa foi explicado os objetivos do trabalho desenvolvido e da pesquisa. A pesquisa elaborada contém um roteiro de testes para os participantes executarem e posteriormente um questionário para avaliarem o uso do aplicativo. Essa pesquisa, com o roteiro de testes e o questionário correspondente, podem ser visualizados no Apêndice A. O aplicativo foi disponibilizado em sua versão *web* hospedada na Amazon.

Figura 27 - Avaliação de usabilidade



Fonte: Elaborado pelo autor.

A primeira turma com 30 alunos foi dividida em pequenos grupos, no qual um dos membros de cada grupo desempenhou o papel de professor e o restante o papel de aluno. O

participante com perfil de professor criou uma nova turma e enviou convites por e-mail para os demais participantes do grupo. Após o cadastro dos demais participantes, foi dado início ao roteiro de testes presente no questionário. Nesta primeira turma os participantes utilizaram por aproximadamente 45 minutos o aplicativo e durante este tempo o autor e a orientadora tiraram dúvidas e acompanharam os testes. A avaliação pode ser confirmada na Figura 27.

Na segunda turma com 7 alunos o autor desempenhou o papel de professor, criou uma nova turma e enviou convites para os participantes. Todos os participantes desta segunda turma utilizaram o perfil de aluno e da mesma forma da turma anterior, testaram o aplicativo conforme o roteiro de testes do questionário por aproximadamente 25 minutos. O autor e a orientadora também acompanharam os testes e tiraram dúvidas sobre o aplicativo.

Os participantes receberam um questionário elaborado com o auxílio da ferramenta Google Forms. O questionário aplicado contém 13 afirmações a serem avaliadas através da escala *likert* de um (1) a cinco (5), onde um (1) significa discordo completamente e cinco (5) significa concordo completamente. As afirmações são apresentadas no Quadro 21.

Quadro 21 – Afirmações a serem avaliadas na escala *likert*

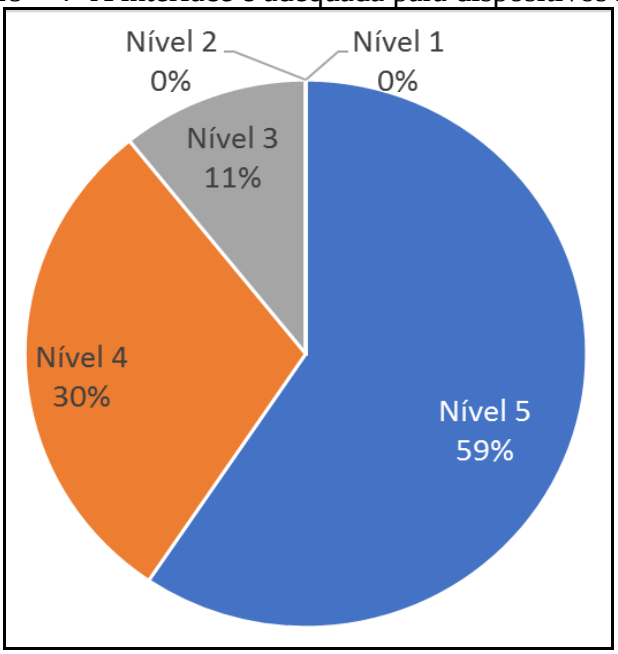
Afirmação
1- A interface é intuitiva e clara.
2- A interface, as cores e os textos são agradáveis e legíveis.
3- As funcionalidades são facilmente identificadas e acessadas.
4- A interface é adequada para dispositivos móveis.
5- A apresentação das publicações no mural da turma é feita de forma clara e bem estruturada.
6- É intuitivo criar um novo post no mural.
7- Foi fácil de explorar o mural da turma.
8- Foi simples de achar a opção para avaliar o curso.
9 - A forma de avaliar o curso através de estrelas é clara e objetiva.
10 - É uma boa forma de avaliar o serviço prestado pelo professor.
11 - Foi fácil de achar o menu de conversas.
12 - O chat do aplicativo é simples mas bem funcional.
13 - O chat do aplicativo é uma boa forma de manter uma boa comunicação entre o professor e os alunos.

Fonte: Elaborado pelo autor.

As afirmações apresentadas no Quadro 21 em sua grande maioria obtiveram média 4 na escala *likert*, ou seja, as respostas foram mais próximas do nível 5 (concordo completamente). A seguir serão apresentados em detalhes 3 afirmativas do questionário. As demais respostas podem ser vistas no Apêndice B.

A afirmativa número 4, em um total de 37 respostas, obteve 22 respostas no nível 5, 11 respostas no nível 4 e 4 respostas no nível 3. A Figura 28 apresenta estas informações de forma gráfica.

Figura 28 – 4- A interface é adequada para dispositivos móveis

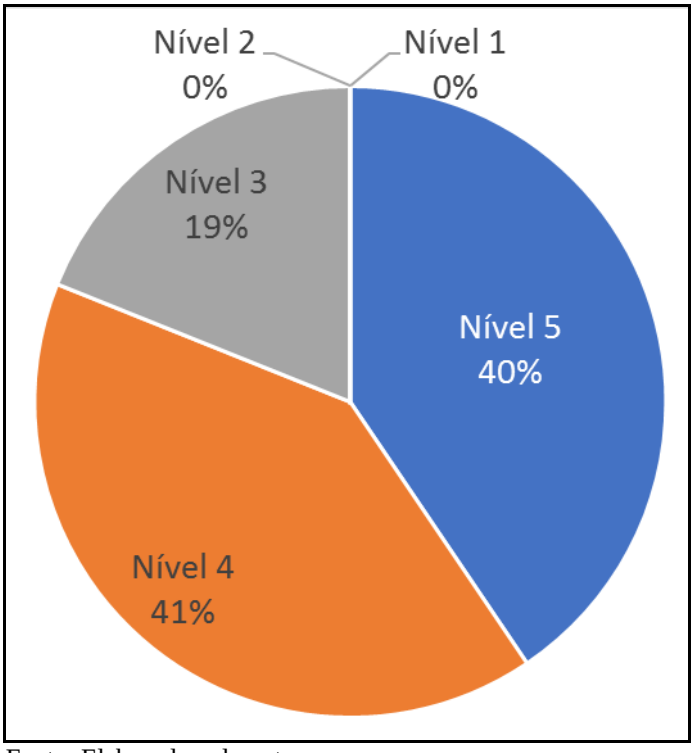


Fonte: Elaborado pelo autor.

As respostas apresentadas na Figura 28 indicam que a maioria dos participantes concordam que a interface do aplicativo é adequada para dispositivos móveis.

A afirmativa 5, em um total de 37 respostas, obteve 15 respostas no nível 5, 15 respostas no nível 4 e 7 respostas no nível 3. A Figura 29 apresenta estas informações de forma gráfica.

Figura 29 – 5- A apresentação das publicações no mural da turma é feita de forma clara e bem estruturada

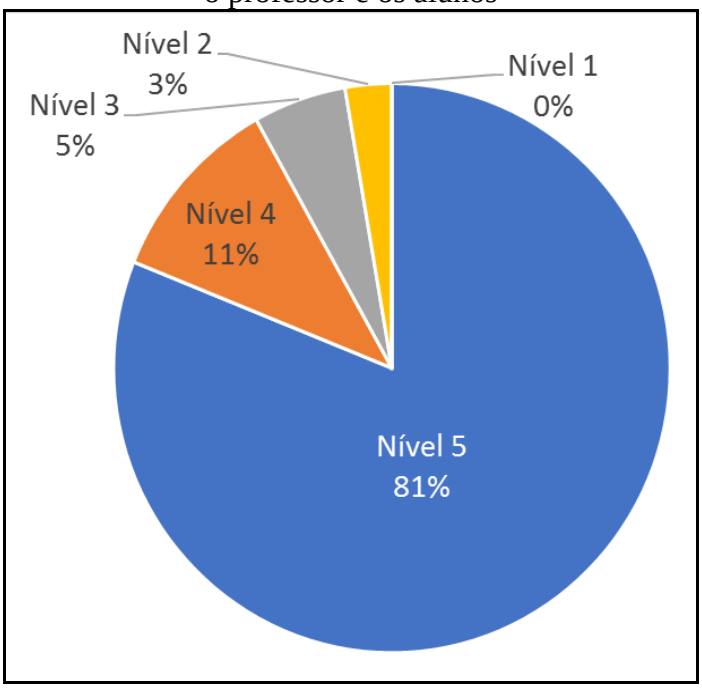


Fonte: Elaborado pelo autor.

As respostas apresentadas na Figura 29 indicam que a maioria dos participantes concordam que a apresentação dos posts no mural da turma é feita de forma clara e bem estruturada.

A afirmativa 13, em um total de 37 respostas, obteve 30 respostas no nível 5, 4 respostas no nível 4, 2 respostas no nível 3 e 1 resposta no nível 2. A Figura 30 apresenta estas informações de forma gráfica.

Figura 30 – 13- O *chat* do aplicativo é uma boa forma de manter uma boa comunicação entre o professor e os alunos



Fonte: Elaborado pelo autor.

As respostas apresentadas na Figura 30 indicam que a maioria dos participantes concordam que o *chat* do aplicativo é uma boa forma de manter uma boa comunicação entre o professor e os alunos.

Além destas afirmativas, foram adicionadas perguntas qualitativas e descritivas, com o propósito de obter pontos fortes e fracos do aplicativo, observações, sugestões de melhoria e feedback de possíveis usuários do aplicativo. O Quadro 22 apresenta alguns pontos fortes, fracos, observações e sugestões destacados pelos participantes do questionário.

Quadro 22 – Pontos fortes, fracos e observações/sugestões destacados pelos participantes do questionário

Pontos Fortes	Pontos Fracos	Observações/Sugestões
Responsividade, fácil adaptação a zoom in/out, tela mobile, clareza, interface limpa	Para um navegador em um computador a interface não ficou muito intuitiva.	Acredito que um pouco mais de texto para descrever onde/o que cada botão faz e onde o usuário deveria clicar.

Se adapta muito bem ao mobile	Em alguns pontos há uma demora para carregar as informações, por exemplo para carregar a foto de perfil	Ao navegar entre as páginas é interessante um feedback avisando que está aguardando, da atual forma parece que clico e não acontece nada.
Uma ideia muito boa para ser aplicada, com uma interface interessante pelo fato de ser responsiva.	Pesado, demorando para mandar mensagens ou carregar imagem	Arrumaria uns bugs que acontecem de interface, botão salvar está muito oculto (na tela de perfil), colocaria uma introdução de como usar pela primeira vez (no primeiro acesso)
Ideia interessante, Responsividade, Simplicidade	As notificações somem muito rápido, não dá tempo para ler o que está escrito nelas.	O botão de criação de publicação quebra todos os padrões que o site tinha (que era no caso que o botão ficava no canto superior direito)
Gostei muito do layout e da ideia	Alguns botões não estão muito bem destacados, tornando difícil percebe-los, como por exemplo o botão de salvar informações do perfil	A interface ficou boa para mobile mais estranho para desktop.

Fonte: Elaborado pelo autor.

Nas respostas das perguntas descritivas ficou bem claro que deve ser feito uma interface e design diferente para telas maiores. Algumas ideias de design aplicadas funcionam muito bem apenas para dispositivos móveis e deixam os usuários de desktop um pouco perdidos. Segundo os alunos, o aplicativo móvel ficou com uma interface simples e limpa, mas peca em pouca padronização de botões e posicionamento de elementos em algumas telas. Além de comentários e sugestões relacionados a usabilidade do aplicativo, os alunos de forma geral elogiaram a ideia e visualizam uma aplicabilidade no mercado.

4 CONCLUSÕES

O aplicativo desenvolvido neste trabalho foi projetado para atender professores e alunos no âmbito de melhorar e otimizar a gestão de aulas particulares e engajar a comunicação e interação entre eles. Conforme os objetivos do trabalho, pode-se afirmar que o primeiro objetivo específico fornecer suporte à interação e colaboração entre alunos e professor foi alcançado. Isso pode ser afirmado pois através de testes com alunos de duas turmas da professora orientadora foi concluído que o aplicativo consegue gerir aulas particulares de forma simples e satisfatória, além de conseguir estimular a colaboração entre alunos e professores.

A arquitetura adotada para o trabalho desenvolvido foi importante pois alcançou o segundo objetivo específico referente a disponibilizar um Web Service responsável pelas regras de negócio do aplicativo, de modo que possa ser integrado com qualquer aplicativo de ensino. Mesmo não sendo feita nenhuma integração com outros sistemas, pode-se dizer que como o módulo servidor e o *front-end* do aplicativo são aplicações totalmente separadas e se comunicam através do Web Service é possível realizar uma integração de forma simples.

As ferramentas utilizadas no desenvolvimento provaram ser úteis e ágeis para criação de um aplicativo *web* responsivo e móvel com uma mesma base de código. Além disso, foi possível utilizar uma mesma linguagem tanto no módulo servidor, quanto no *front-end* do aplicativo, aumentando a produtividade e diminuindo a curva de aprendizagem. Os *frameworks* VueJS e Quasar auxiliaram no desenvolvimento do *front-end* do aplicativo, uma vez que o autor já possuía um conhecimento intermediário sobre o *framework* VueJS.

O aplicativo também se provou um forte aliado na busca de colaboração e engajamento entre os professores e alunos. A pesquisa realizada no trabalho torna evidente a afirmação anterior, uma vez que a grande maioria dos participantes achou as funcionalidades intuitivas e claras. Além de se referirem ao aplicativo com um bom potencial para ser uma ferramenta complementar ao ensino. Com relação aos pilares da colaboração, o aplicativo demonstra coordenação através da gestão das aulas, alunos, horários e avaliações; comunicação através dos *chats* privados e em grupo; e a cooperação através do mural de publicações e comentários.

Através da pesquisa, pode-se perceber que o aplicativo possui algumas funcionalidades faltando. Essas funcionalidades foram identificadas através das sugestões de melhoria e evolução capturados pelo questionário. A mais nítida entre os comentários é a construção de interfaces diferentes dependendo do dispositivo utilizado pelo usuário. Em alguns casos, as

telas do aplicativo funcionam muito bem para dispositivos móveis, mas deixam a desejar quando são utilizadas em navegadores *desktop*. Alguns detalhes sobre *design* e posicionamento de componentes de telas também foram observados pelos participantes da pesquisa, além de pequenas falhas na padronização das telas. Na grande maioria das telas o botão de salvar/adicionar, por exemplo, estava no canto superior direito e em uma minoria estava no canto inferior direito.

4.1 EXTENSÕES

Foram identificadas algumas limitações no trabalho, e com base nessas foram destacadas algumas possíveis extensões. Os pontos observados durante a construção e validação do aplicativo que servem como extensões para o trabalho são:

a) criar um perfil de usuário destinado aos pais dos alunos;
b) criar um *bot* para o chat da turma, para ajudar o professor a responder perguntas frequentes dos alunos;
c) versão de interface diferente para dispositivos com tamanhos de telas diferentes;
d) aplicar outro protocolo de comunicação ou técnica para o chat;
e) criar um guia de introdução para primeiro acesso;
f) criar área destinada a anúncios de cursos;
g) refinar padrão de design das telas;
h) gerar e testar versão em dispositivos IOS;
i) suporte para aulas *online*;
j) melhorar performance do envio e recebimento de mensagens do *chat* de forma que não seja feito uma busca de mensagens a cada nova mensagem recebida;
k) suporte para pagamento das aulas particulares;
l) visualizar as turmas que o aluno faz parte, caso o aluno esteja em mais de uma turma do professor.

REFERÊNCIAS

AGUIAR, Paulo Henrique. **Sistema de Informação para Gestão Educacional:** sistematização de uma proposta de modelo e avaliação do processo de sua construção. 2004. 181 f. Dissertação (Mestrado) - Curso de Profissional em Computação, Centro de Ciências e Tecnologia, Universidade Estadual do Ceará, Ceará, 2004.

ALCANTARA, Carlos Augusto Almeida; VIEIRA, Anderson Luiz Nogueira. **Tecnologia móvel:** uma tendência, uma realidade. 2009. Disponível em: <https://arxiv.org/ftp/arxiv/papers/1105/1105.3715.pdf>. Acesso em: 11 set. 2016.

ALMEIDA, Maria Elizabeth Bianconcini de**. Tecnologia de informação e comunicação na escola**: aprendizagem e produção da escrita. 2001. Disponível em: <http://www.eadconsultoria.com.br/matapoio/biblioteca/textos_pdf/texto24.pdf>. Acesso em: 17 jun. 2017.

APPSZOOM. **Iprofe:** Gerenciador de Alunos – Grátis. 2013. Disponível em: <http://pt.appszoom.com/iphone_applications/productivity/iprofe-gerenciador-de-alunos-gratis_fnqme.html>. Acesso em: 30 ago. 2016.

ASTAH. Astah Community. 2009. Disponível em: <http://astah.net/editions/community>. Acesso em: 06 jul. 2017.

CAMPAGNOLI, Julian de Alessandro. **Como criar aplicativos mobile híbridos e offline.** 2015. Disponível em: <http://www.devmedia.com.br/como-criar-aplicativos-mobile-hibridos-e-offline/32361>. Acesso em: 7 set. 2016.

CASTRO, Alberto; MENEZES, Crediné. Aprendizagem colaborativa com suporte computacional. In: FUKS, Hugo; PIMENTEL, Mariano. **Sistemas Colaborativos.** São Paulo: Elsevier, 2012. p. 135-156.

CHRISTENSEN, Clayton M.; HORN, Michael B.; STAKER, Heather. **Ensino Híbrido**: uma Inovação Disruptiva?: Uma introdução à teoria dos híbridos. 2013. Disponível em: <https://s3.amazonaws.com/porvir/wp-content/uploads/2014/08/PT_Is-K-12-blended-learning-disruptive-Final.pdf>. Acesso em: 16 jun. 2017.

CLAYTON CHRISTENSEN INSTITUTE. **Modelos de Ensino Híbrido**. 2017. Disponível em: <http://www.blendedlearning.org/modelos/?lang=pt-br>. Acesso em: 16 jun. 2017.

COSTA, Jorge Adelino. As explicações (aulas particulares) enquanto vantagem competitiva no mercado educativo: os novos herdeiros e as estratégias privadas de sucesso público. In: CONGRESSO LUSO-BRASILEIRO, 5., 2007, Porto Alegre. **Por uma escola de qualidade para todos**. Porto Alegre: Anpae, 2007. v. 4, p. 1 - 15.

DAMIANI, Magda Floriana. **A Teoria da Atividade como ferramenta para entender o desempenho de duas escolas de ensino fundamental**. 2006. 17 f. Monografia (Especialização) - Curso de Pós-graduação em Educação, Universidade Federal de Pelotas, Pelotas, 2006.

ESPÍNDOLA, Rafaela. **O que é e como funciona o blended learning?** 2016. Disponível em: <http://www.edools.com/blended-learning/>. Acesso em: 16 jun. 2017.

FEDOCE, Rosângela Spagnol; SQUIRRA, Sebastião Carlos. A tecnologia móvel e os potenciais da comunicação na educação. **Logos,** Rio de Janeiro, v. 18, n. 2, p.267-278, dez. 2011.

FILIPPO, Denise et al. Mobilidade e ubiquidade para colaboração. In: FUKS, Hugo; PIMENTEL, Mariano. **Sistemas Colaborativos.** São Paulo: Elsevier, 2012. p. 294-316.

FUKS, Hugo et al. Teorias e Modelos de colaboração. In: FUKS, Hugo; PIMENTEL, Mariano. **Sistemas Colaborativos.** São Paulo: Elsevier, 2012. p. 16-33.

FUNDAÇÃO LEMANN. **Ensino Híbrido**: Personalização e tecnologia na educação. 2017. Disponível em: <http://www.fundacaolemann.org.br/ensino-hibrido/>. Acesso em: 16 jun. 2017.

GUTWEIN, Carl; GREENBERG, Saul. **A Descriptive Framework of Workspace Awareness for Real-Time Groupware.** Saskatchewan: University of Saskatchewan, 2001.

LOPES, Sergio. **Aplicações mobile híbridas com Cordova e PhoneGap**. São Paulo: Casa do Código, 2016.

MARQUES, Isabela Quaglia; CAETANO, Fabiana Sesmilo de Camargo. A Utilização do Moodle em Cursos Presenciais em uma Instituição de Ensino Superior. **Revista Científica em Educação A Distância,** Rio de Janeiro, v. 4, n. 2, p.107-123, dez. 2014.

MATTJE, Guilherme Lopes. **Integrando serviços corporativos em aplicativo para smartphones sem uso de linguagens nativas:** Estudo de caso na empresa Softplan Planejamento e Sistemas Ltda. 2014. 43 f. Monografia (Especialização) - Curso de Curso de Análise e Desenvolvimento de Sistemas, Universidade Estácio de SÁ, São José, 2014.

MENDES, Mariana Ribeiro; GARBAZZA, Itagildo Edmar; TERRA, Daniela Costa. **Desenvolvimento híbrido versus desenvolvimento nativo de aplicativos móveis.** 2014. Disponível em: <http://www.cefetbambui.edu.br/portal/files/j7_ifmg_bambui_in3.pdf>. Acesso em: 7 set. 2016.

MORAN, José Manuel. Ensino e aprendizagem inovadores com tecnologias. **Informática na Educação: Teoria & Prática**, Porto Alegre, v. 3, n. 1, p.137-144, set. 2000.

NICOLACI-DA-COSTA, Ana Maria; PIMENTEL, Mariano. Sistemas Colaborativos para uma nova sociedade e um novo ser humano. In: PIMENTEL, Mariano; FUKS, Hugo. **Sistemas Colaborativos.** São Paulo: Elsevier, 2012. p. 3-15.

PEREIRA, Maria da Conceição; SILVA, Tânia Maria da. O uso da tecnologia na educação na era digital. **Revistas - Saberes em Rede**, Cuiabá, v. 2, n. 2, p.85-94, dez. 2013.

PIMENTEL, Mariano; GEROSA, Marco Aurélio; FUKS, Hugo. Sistemas de comunicação para colaboração. In: PIMENTEL, Mariano; FUKS, Hugo (Org.). **Sistemas Colaborativos**. São Paulo: Elsevier, 2012. p. 65-93.

PLAY STORE. **iProfe - Ger. de Alunos Free**. 2017. Disponível em: <https://play.google.com/store/apps/details?id=br.com.iprofe.lite>. Acesso em: 21 jun. 2017.

PREECE, J.; ROGERS, Y. e SHARP, H. (2002). **What is Interaction Design. In: Interaction Design**: Beyond Human Computer Interaction, USA: John Wiley & Sons, Inc. p. 1-33.

PROFES. **Blog do profes**: 4 ótimas dicas de organização para professore particular. 2016. Disponível em: <https://br.portalprofes.com/blog/4-otimas-dicas-de-organizacao-para-professor-particular>. Acesso em: 30 ago. 2016.

PROFES. **Profes**: Quem somos. 2012. Disponível em: <https://br.portalprofes.com/info/sobre/>. Acesso em: 30 ago. 2016.

QUASAR. **Quasar Framework**: SPA front-end on steroids. 2017. Disponível em: <http://quasar-framework.org/>. Acesso em: 16 jun. 2017.

RIBEIRO, Vitor Filincowsky. **Decisão colaborativa com utilização de Teoria dos Jogos para o sequenciamento de partidas em aeroportos.** 2013. 122 f. Monografia (Especialização) - Curso de Pós-graduação em Informática, Universidade de Brasília, Brasília, 2013.

SCARTEZINI, Raphael. **Iprofe**. 2013. Disponível em: <http://www.iprofe.com.br/>. Acesso em: 30 ago. 2016.

SCHEIDEMANTEL, Andressa Abreu. **O impacto dos sistemas colaborativos nas organizações**: o caso Fiat Mio. 2013. 82 f., il. Monografia (Bacharelado em Comunicação Organizacional)—Universidade de Brasília, Brasília, 2013.

SEBASTIÃO, Ana Paula Ferreira. A utilização do ambiente virtual de aprendizagem moodle em uma instituição de ensino superior pública. **Revista Profissão Docente,** Uberaba, v. 15, n. 32, p.131-139, jul. 2015.

SILVA, Angela Carrancho da. Educação e tecnologia: entre o discurso e a prática. **Revista Ensaio: Avaliação e Políticas Públicas em Educação**, Rio de Janeiro, v. 19, n. 72, p.527-554, set. 2011.

SILVA, Leandro L.B.; PIRES, Daniel F.; NETO, Silvio C. Desenvolvimento de Aplicações para Dispositivos Móveis: Tipos e Exemplo de Aplicação na plataforma iOS. In: II WORKSHOP DE INICIAÇÃO CIENTÍFICA EM SISTEMAS DE INFORMAÇÃO, 2., 2015, Goiânia. **Anais eletrônicos...** Goiânia: Uni-FACEF, 2015. 4 p.

SILVA, Maicon Anderson Mattos da et al. Desenvolvimento de um Caderno de Campo para Plataformas Móveis utilizando PhoneGap. In: CONGRESSO BRASILEIRO DE AGROINFORMÁTICA, 9., 2013, Cuiabá. **Agroinformática: Inovação para a Sustentabilidade do Agronegócio Brasileiro.** Cuiabá: Sbiagro, 2013. p. 1 - 6.

SOUZA, Isabel Maria Amorim de; SOUZA, Luciana Virgília Amorim de. **O uso da tecnologia como facilitadora da aprendizagem do aluno na escola.** 2010. Disponível em: <http://www.seer.ufs.br/index.php/forumidentidades/article/view/1784/1573>. Acesso em: 7 set. 2016.

SOUZA, Jano Moreira de et al. Gestão do conhecimento e memória de grupo. In: FUKS, Hugo; PIMENTEL, Mariano. **Sistemas Colaborativos**. São Paulo: Elsevier, 2012. Cap. 13. p. 206-220.

SOUZA, Pricila Rodrigues de; ANDRADE, Maria do Carmo Ferreira de. Modelos de rotação do ensino híbrido: estações de trabalho e sala de aula invertida. **E-tech**: Tecnologias para competitividade industrial, Florianópolis, v. 1, n. 9, p.3-16, jan. 2016. Disponível em: <http://revista.ctai.senai.br/index.php/edicao01/article/view/773>. Acesso em: 16 jun. 2017.

STAKER, Heather; HORN, Michael B.. **Classifying K–12 Blended Learning**. 2012. Disponível em: <https://www.christenseninstitute.org/wp-content/uploads/2013/04/Classifying-K-12-blended-learning.pdf>. Acesso em: 16 jun. 2017.

SUPERPROF. **Superprof**. 2011. Disponível em: <http://br.superprof.com/>. Acesso em: 30 ago. 2016.

TARCIA, Rita Maria Lino; COSTA, Silvia Maria Coelho. Contexto da educação a distância. In: CARLINI, Alda Luiza; TARCIA, Rita Maria Lino(org.). **20% a distância e agora?** Orientações práticas para o uso de tecnologias de educação a distância no ensino presencial. São Paulo: Pearson Education do Brasil, 2010.

VUE. **The Progressive Javascript Framework**. 2017. Disponível em: <https://vuejs.org/>. Acesso em: 16 jun. 2017.

APÊNDICE A – FORMULÁRIO DE AVALIAÇÃO

Este apêndice contém o formulário de avaliação disponibilizado na pesquisa realizada com os alunos de duas turmas da professora orientadora. Este conteúdo foi gerado com o auxílio da ferramenta Google Forms.

Figura 31 - Introdução ao questionário

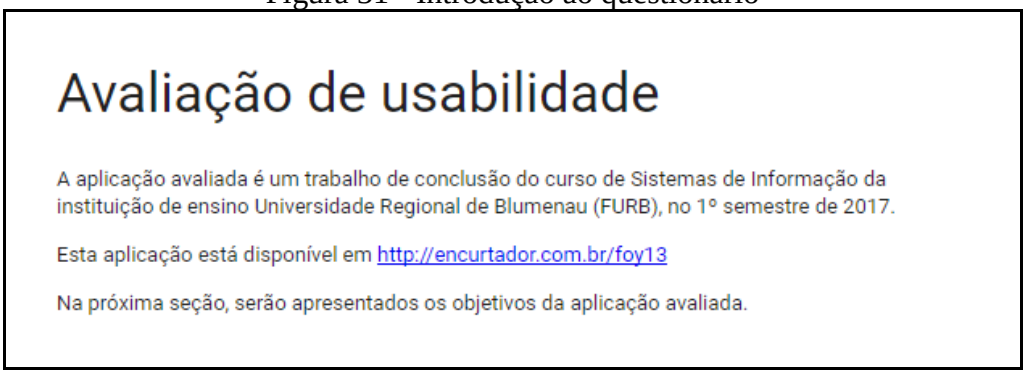


Figura 32 - Objetivos do questionário

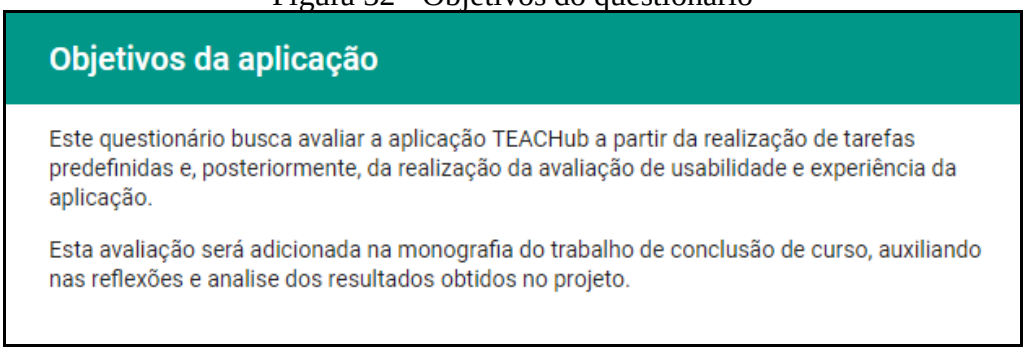


Figura 33 - O aplicativo

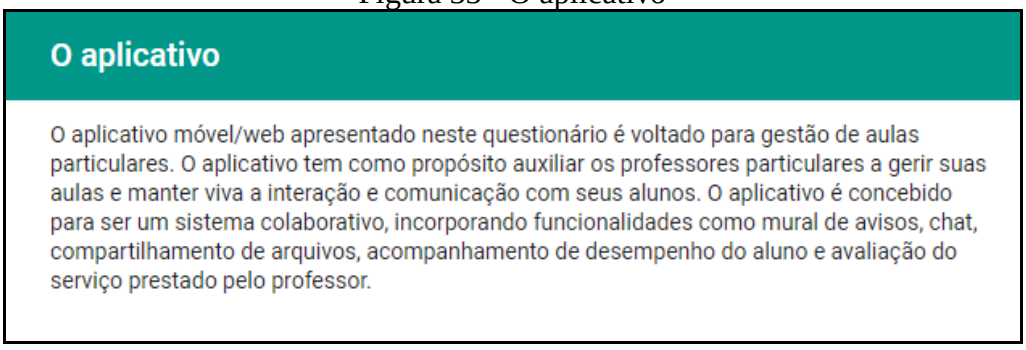
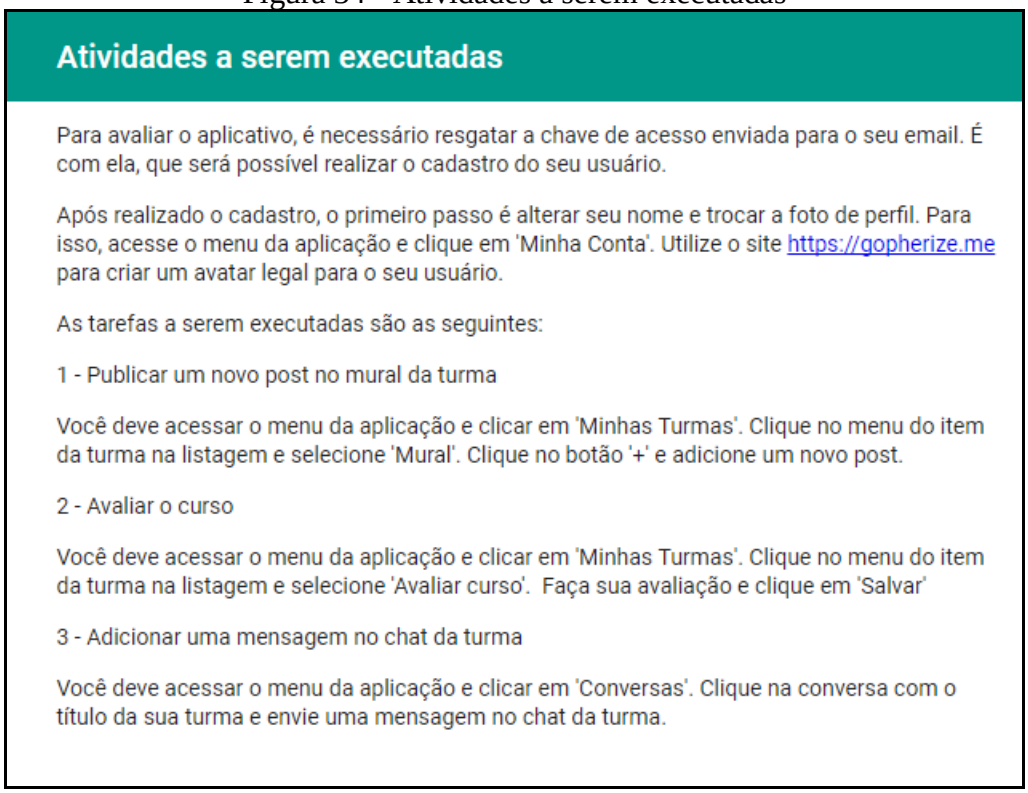


Figura 34 - Atividades a serem executadas



Atividades a serem executadas

Para avaliar o aplicativo, é necessário resgatar a chave de acesso enviada para o seu email. É com ela, que será possível realizar o cadastro do seu usuário.

Após realizado o cadastro, o primeiro passo é alterar seu nome e trocar a foto de perfil. Para isso, acesse o menu da aplicação e clique em 'Minha Conta'. Utilize o site https://gopherize.me para criar um avatar legal para o seu usuário.

As tarefas a serem executadas são as seguintes:

1 - Publicar um novo post no mural da turma

Você deve acessar o menu da aplicação e clicar em 'Minhas Turmas'. Clique no menu do item da turma na listagem e selecione 'Mural'. Clique no botão '+' e adicione um novo post.

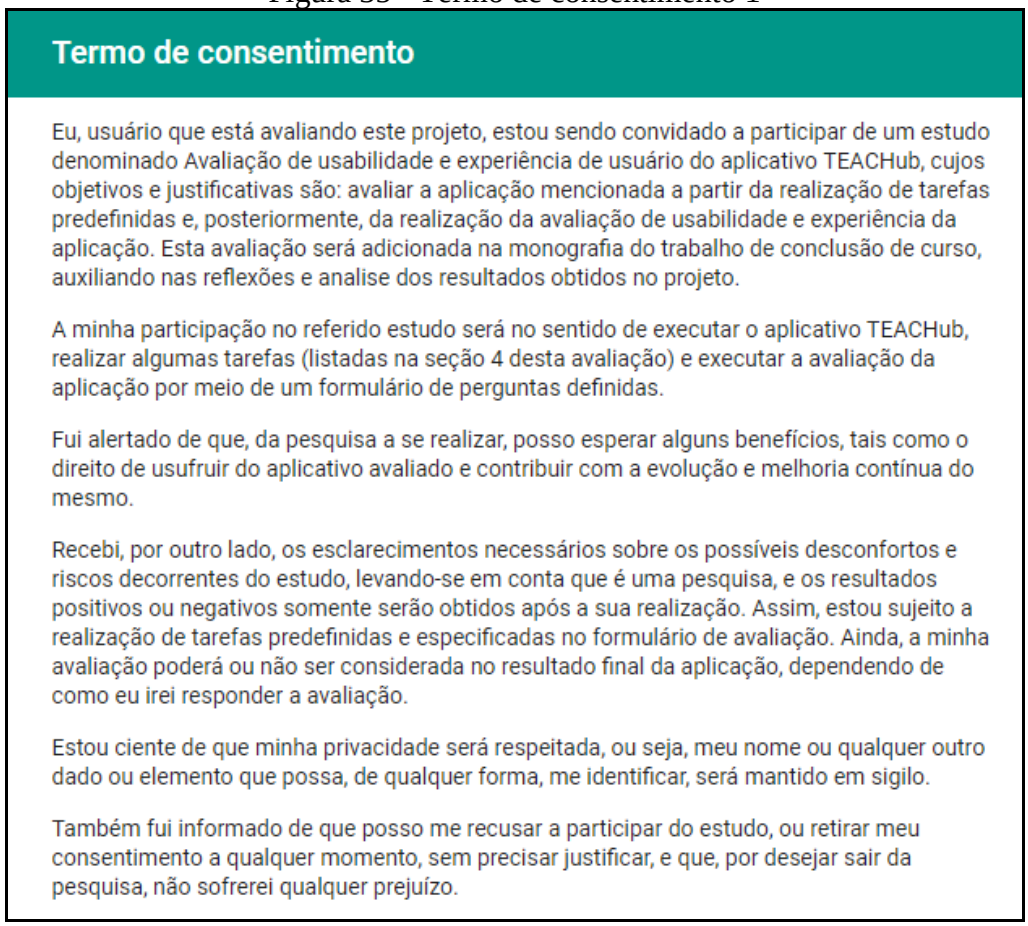
2 - Avaliar o curso

Você deve acessar o menu da aplicação e clicar em 'Minhas Turmas'. Clique no menu do item da turma na listagem e selecione 'Avaliar curso'. Faça sua avaliação e clique em 'Salvar'

3 - Adicionar uma mensagem no chat da turma

Você deve acessar o menu da aplicação e clicar em 'Conversas'. Clique na conversa com o título da sua turma e envie uma mensagem no chat da turma.

Figura 35 - Termo de consentimento 1



Termo de consentimento

Eu, usuário que está avaliando este projeto, estou sendo convidado a participar de um estudo denominado Avaliação de usabilidade e experiência de usuário do aplicativo TEACHub, cujos objetivos e justificativas são: avaliar a aplicação mencionada a partir da realização de tarefas predefinidas e, posteriormente, da realização da avaliação de usabilidade e experiência da aplicação. Esta avaliação será adicionada na monografia do trabalho de conclusão de curso, auxiliando nas reflexões e analise dos resultados obtidos no projeto.

A minha participação no referido estudo será no sentido de executar o aplicativo TEACHub, realizar algumas tarefas (listadas na seção 4 desta avaliação) e executar a avaliação da aplicação por meio de um formulário de perguntas definidas.

Fui alertado de que, da pesquisa a se realizar, posso esperar alguns benefícios, tais como o direito de usufruir do aplicativo avaliado e contribuir com a evolução e melhoria contínua do mesmo.

Recebi, por outro lado, os esclarecimentos necessários sobre os possíveis desconfortos e riscos decorrentes do estudo, levando-se em conta que é uma pesquisa, e os resultados positivos ou negativos somente serão obtidos após a sua realização. Assim, estou sujeito a realização de tarefas predefinidas e especificadas no formulário de avaliação. Ainda, a minha avaliação poderá ou não ser considerada no resultado final da aplicação, dependendo de como eu irei responder a avaliação.

Estou ciente de que minha privacidade será respeitada, ou seja, meu nome ou qualquer outro dado ou elemento que possa, de qualquer forma, me identificar, será mantido em sigilo.

Também fui informado de que posso me recusar a participar do estudo, ou retirar meu consentimento a qualquer momento, sem precisar justificar, e que, por desejar sair da pesquisa, não sofrerei qualquer prejuízo.

Figura 36 - Termo de consentimento 2

Os pesquisadores envolvidos com o referido projeto são: Lucas Amaral, da Universidade Regional de Blumenau (FURB), onde posso entrar em contato pelo e-mail lucas.amaral.dw@gmail.com e Professora Luciana Pereira de Araújo, da Universidade Regional de Blumenau (FURB), onde posso entrar em contato pelo e-mail lpa@furb.br.

É assegurada a assistência durante toda pesquisa, bem como me é garantido o livre acesso a todas as informações e esclarecimentos adicionais sobre o estudo e suas consequências, enfim, tudo o que eu queira saber antes, durante e depois da minha participação.

Enfim, tendo sido orientado quanto ao teor de todo o aqui mencionado e compreendido a natureza e o objetivo do já referido estudo, manifesto meu livre consentimento em participar, estando totalmente ciente de que não há nenhum valor econômico, a receber ou a pagar, por minha participação.

Caso ocorra algum dano decorrente da minha participação no estudo, serei devidamente indenizado, conforme determina a lei.

Em caso de reclamação ou qualquer tipo de denúncia sobre este estudo devo entrar em contato com Professora Luciana Pereira de Araújo, da Universidade Regional de Blumenau (FURB), onde posso entrar em contato pelo e-mail lpa@furb.br.

Blumenau, 18 de maio de 2017.

Lucas Amaral, Acadêmico - Universidade Regional de Blumenau (FURB)
Luciana Pereira de Araújo, Professora - Universidade Regional de Blumenau (FURB)

AO PROSSEGUIR PARA A PRÓXIMA SEÇÃO DESTE FORMULÁRIO DE AVALIAÇÃO, DECLARO QUE ESTOU DE ACORDO COM OS TERMOS EXPLÍCITOS ACIMA.

Figura 37 - Informações do participante

Informações do participante

Informe o seu e-mail para futuras notícias sobre esta pesquisa
O seu e-mail não será compartilhado, divulgado e nem será utilizado para envio de spam. A única finalidade desta informação é para notificações futuras sobre o resultado desta pesquisa.

Sua resposta

Qual é a sua área de atuação/profissão/estudo? *

Sua resposta

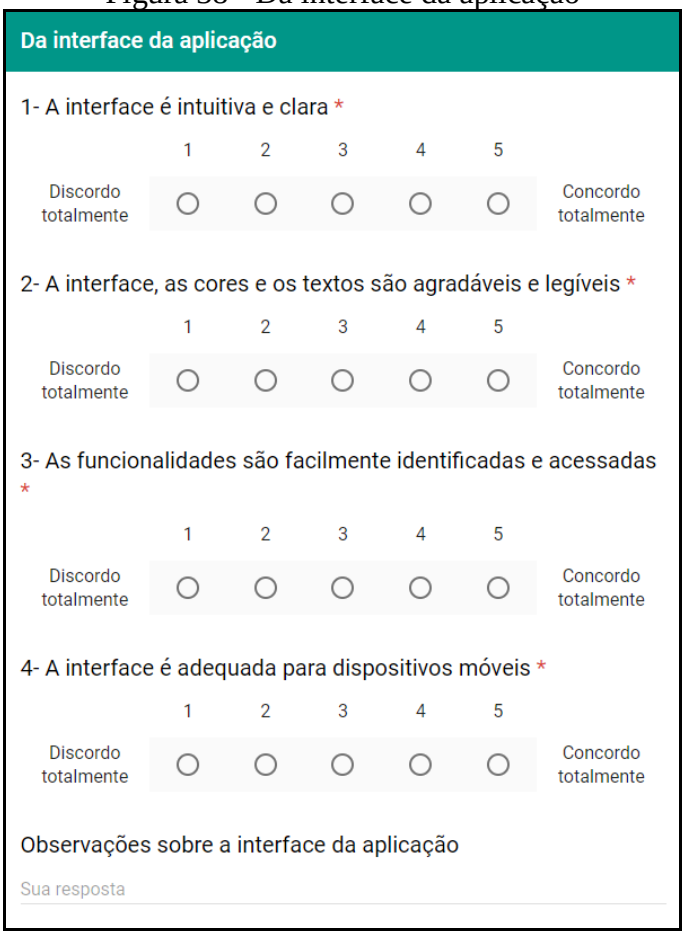
Qual é a sua idade? *

- ○ 15 - 19
- ○ 20 - 24
- ○ 25 - 29
- ○ 30 - 34
- ○ 35 - 39
- ○ 40+

Você utiliza frequentemente o navegador de dispositivos móveis? *

- ○ Sim
- ○ Não

Figura 38 - Da interface da aplicação



Da interface da aplicação

1- A interface é intuitiva e clara *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

2- A interface, as cores e os textos são agradáveis e legíveis *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

3- As funcionalidades são facilmente identificadas e acessadas *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

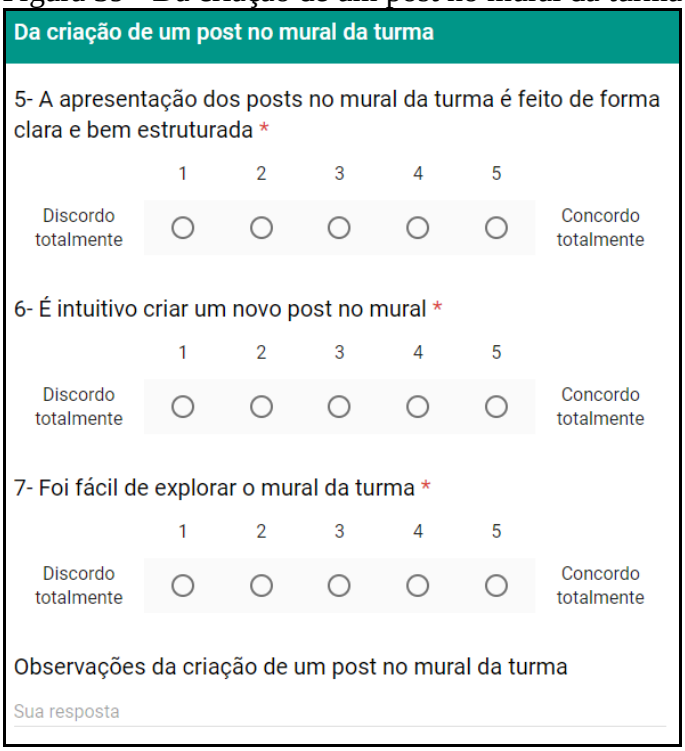
4- A interface é adequada para dispositivos móveis *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Observações sobre a interface da aplicação

Sua resposta

Figura 39 - Da criação de um post no mural da turma



Da criação de um post no mural da turma

5- A apresentação dos posts no mural da turma é feito de forma clara e bem estruturada *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

6- É intuitivo criar um novo post no mural *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

7- Foi fácil de explorar o mural da turma *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Observações da criação de um post no mural da turma

Sua resposta

Figura 40 - Da avaliação do curso

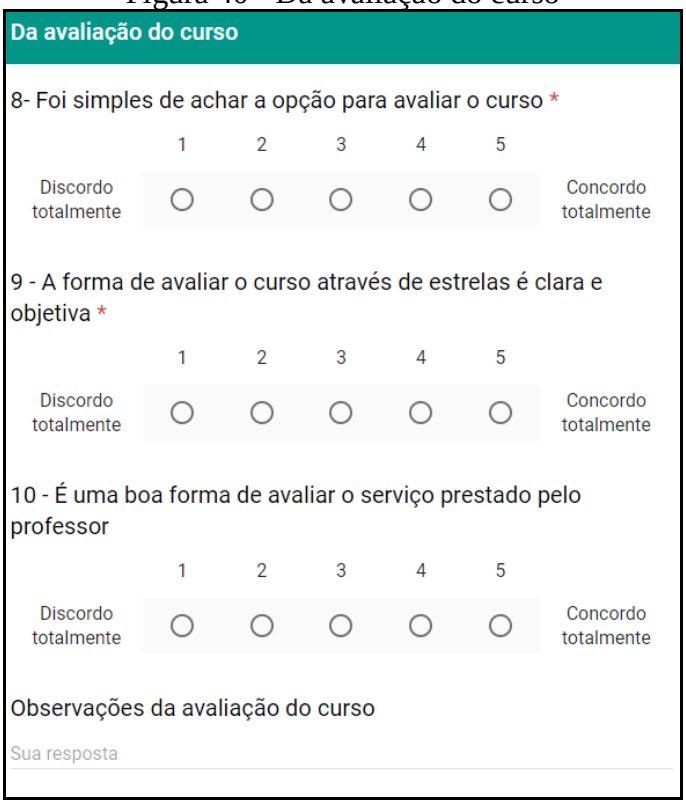


Figura 41 - Do chat da turma

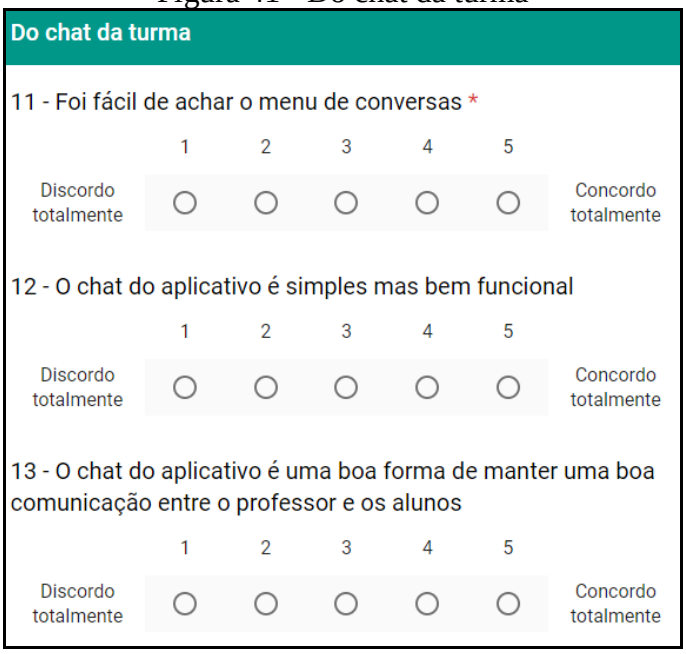
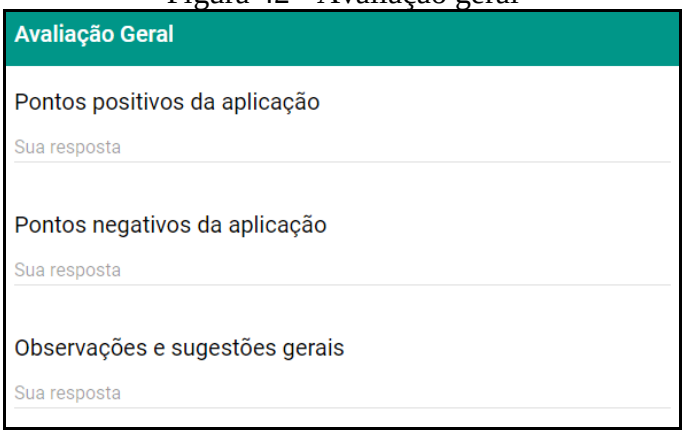


Figura 42 - Avaliação geral



APÊNDICE B – RESULTADOS DA AVALIAÇÃO

Este apêndice contém os resultados da pesquisa realizada. Os gráficos foram gerados com o auxílio do Google Forms.

Figura 43 - Área de atuação dos participantes

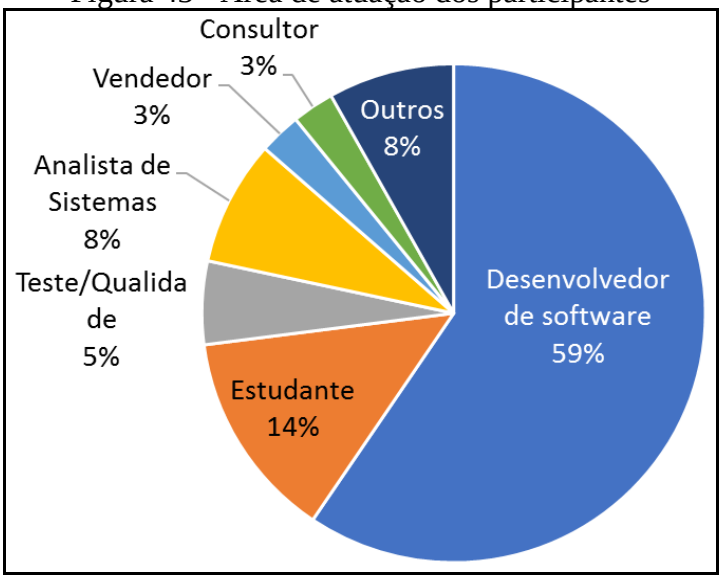


Figura 44 - Idade dos participantes

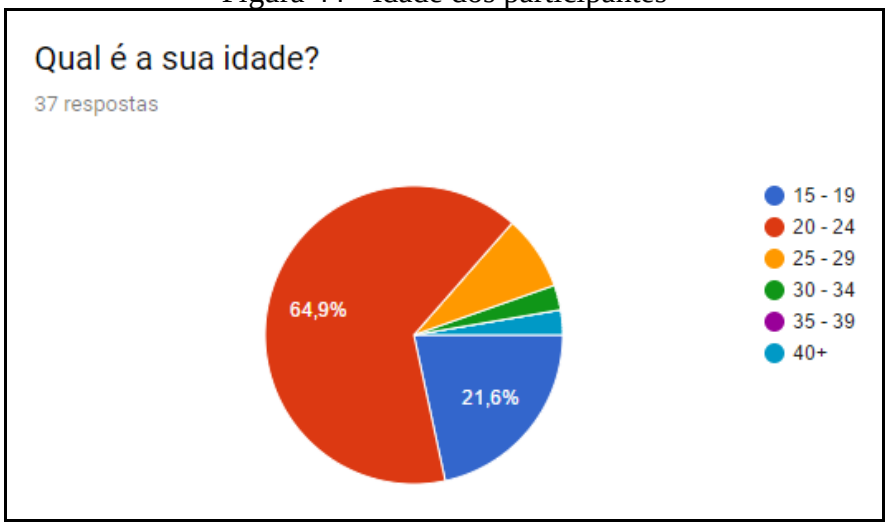


Figura 45 - Uso do navegador em dispositivos móveis

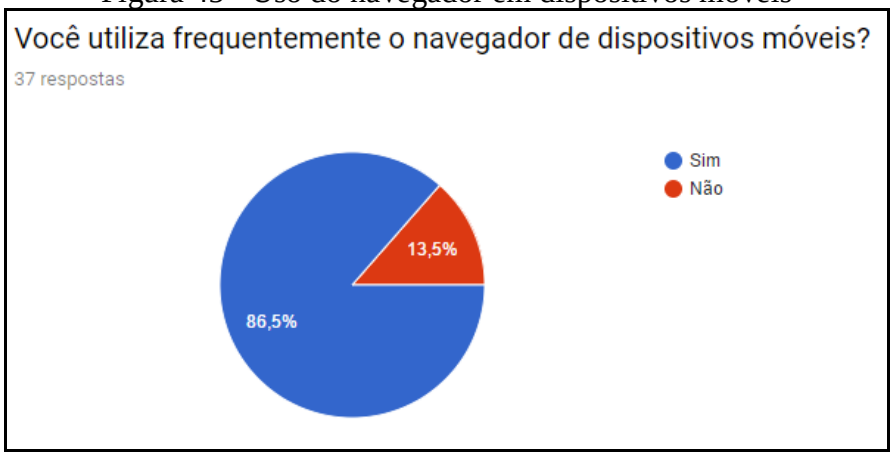


Figura 46 – Resultados da questão 1

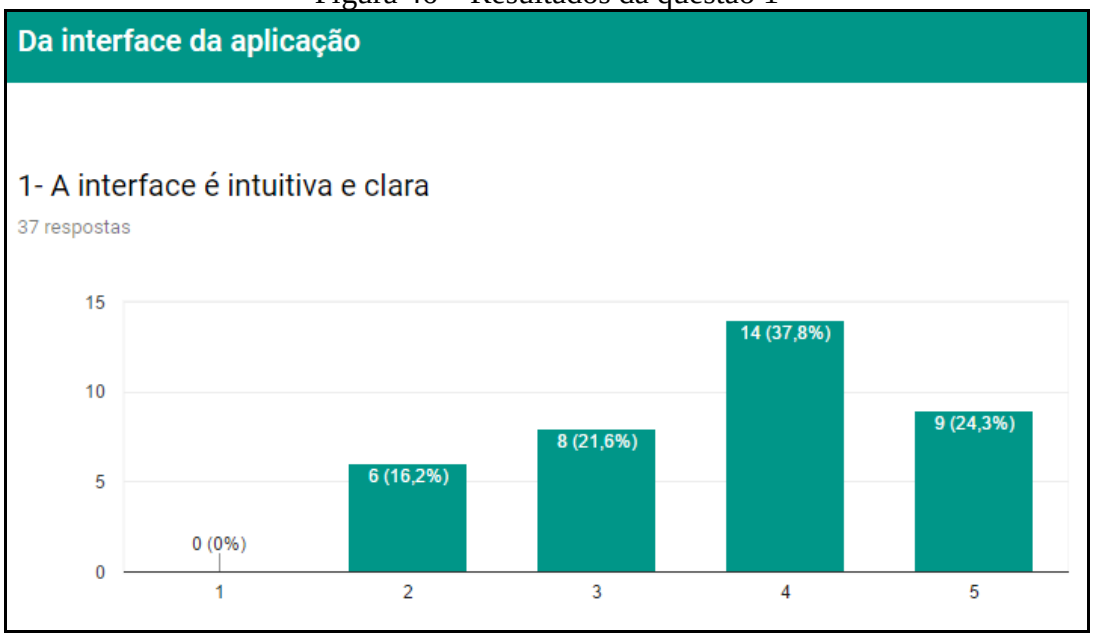


Figura 47 – Resultados da questão 2

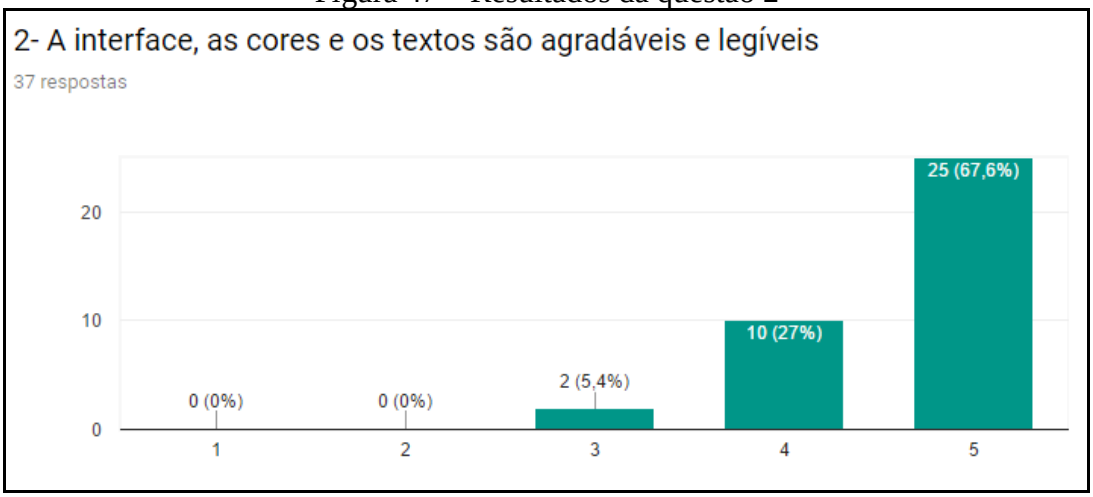


Figura 48 – Resultados da questão 3

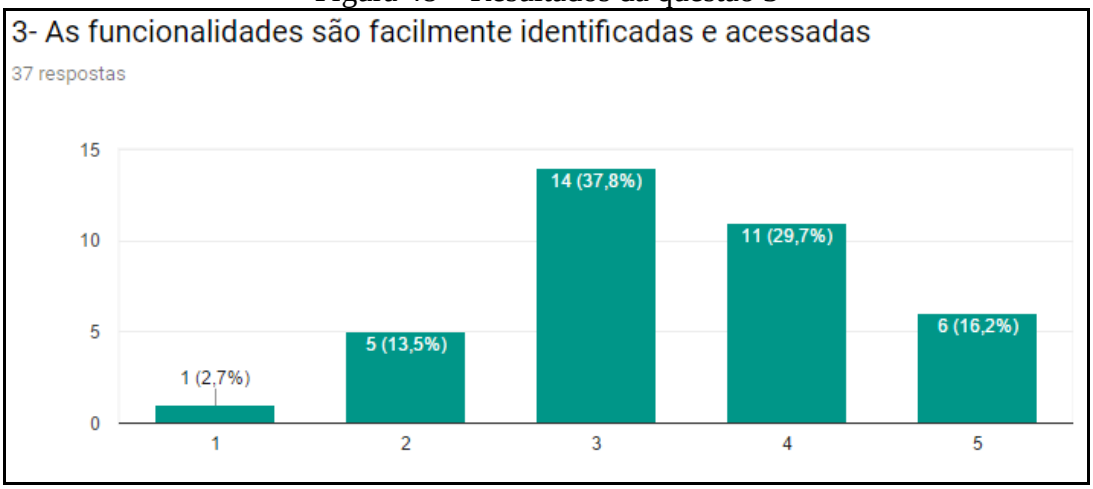


Figura 49 – Resultados da questão 4

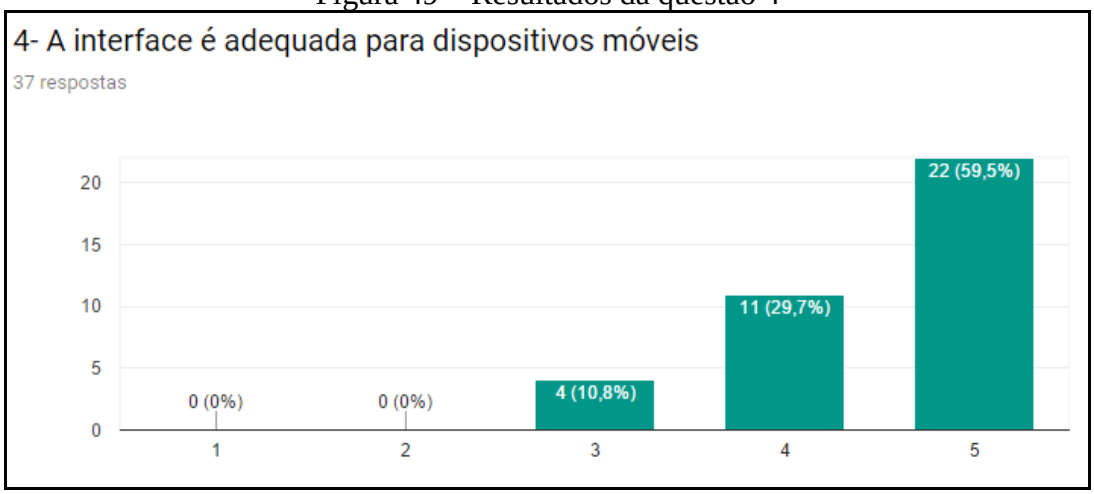


Figura 50 - Observações sobre a interface da aplicação

Em algumas telas, como login, ações esperadas como ENTER para logar não funcionam
após acessar menus de turma ou outros, não aparece mais o menu principal (é necessário voltar todas as páginas até aparecer o menu)
Interface geral é boa.
Utilizei um navegador web e não gostei da interface por ela não ser muito intuitiva e em alguns momentos acaba se perdendo onde é necessário clicar.
A interface ficou boa para mobile mais estranho para desktop.
No primeiro acesso, se a pessoa não recebe orientação, ela fica totalmente perdida.
Deveria aparecer o email da pessoa nas conversas como default, e não "Usuário convidado".
Botões de caixas de diálogo, como Salvar e Cancelar estão invertidos. Falta um feedback sobre laoding das páginas: ao clicar nos nas opções não é apresentado nenhuma mensagem e parece que o aplicativo travou. Muitas funcionalidades estão escondidas, precisando entrar em muitas telas sem muita ligação lógica para acessá-las. Alguns campos, como o de convidar alunos, não deixa claro que tem que clicar em confirmar no canto da tela. Os alunos não conseguem ver as turmas que estão. A tela inicial de professor deveria facilitar o acesso a todas as outras. Alguns ícones, como o de novas mensagens não deixam claro seu significado, pois deveria ter um hint quando passa o mouse. Ao entrar em alguma função pela tela inicial, ao voltar, passa telas que não entrei.
Recebo notificações da tuma mas ao ir em minhas turmas não estão lá.
Ao criar uma publicação no mural, o padrão do botão de criação sai lá de cima pra ficar no canto inferior direito da tela, confundindo o usuario iniciante
O teste foi realizado com base na resolução de m aparelho iPhone 6
Botões "Convidar" e "Cancelar" ao convidar aluno estão invertidos, botão para confirmar o convite do aluno não é muito intuitivo, não aparece as turmas em que o aluno está cadastrado.
O botão 'Salvar' poderia ser mais destacado quando o usuário está utilizando o sistema através de um computador.
Tive dificuldade de achar o ícone de confirmação na hora de alterar o nome da minha conta
Não testei a interface no dispositivo móvel por falta de bateria.

Figura 51 - Resultados da questão 5

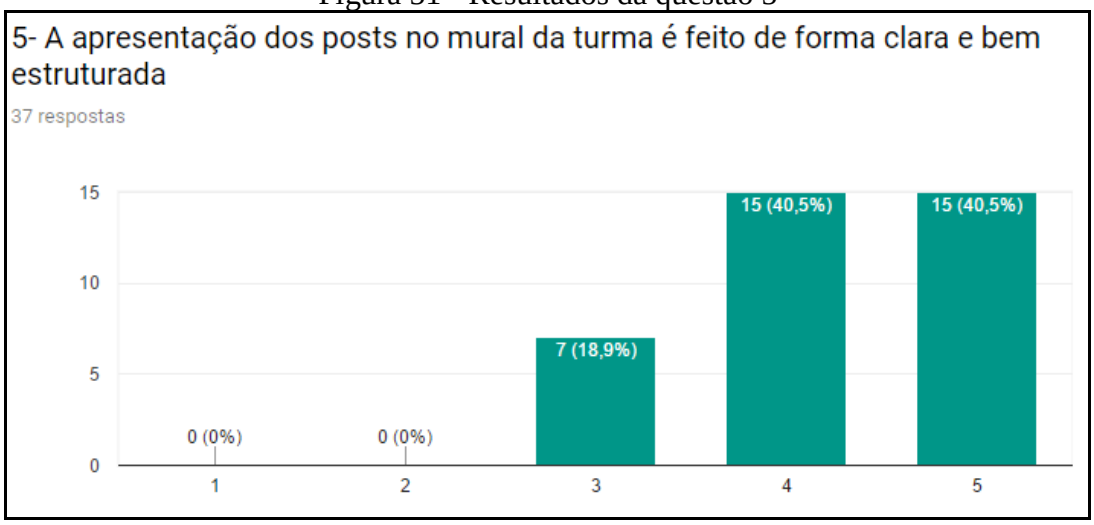


Figura 52 - Resultados da questão 6

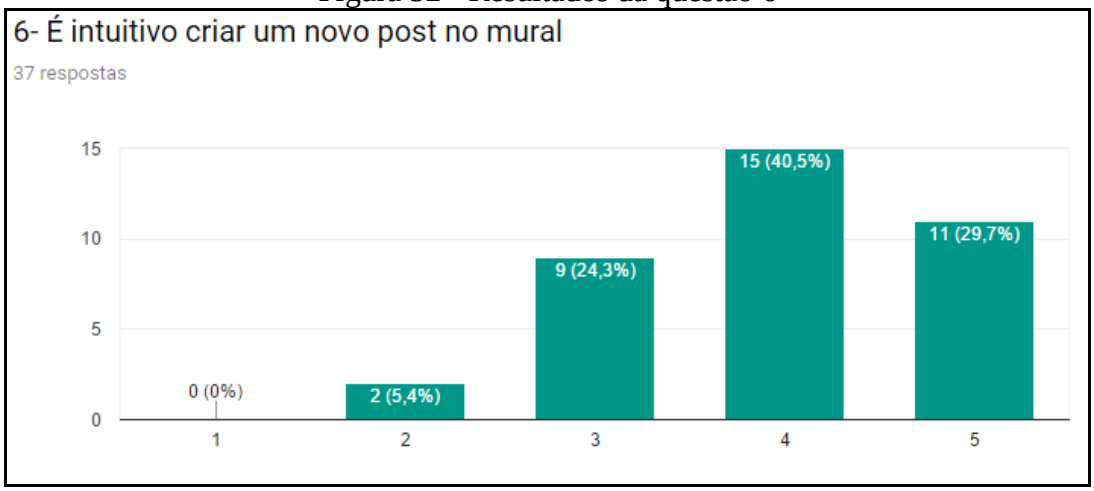


Figura 53 - Resultados da questão 7

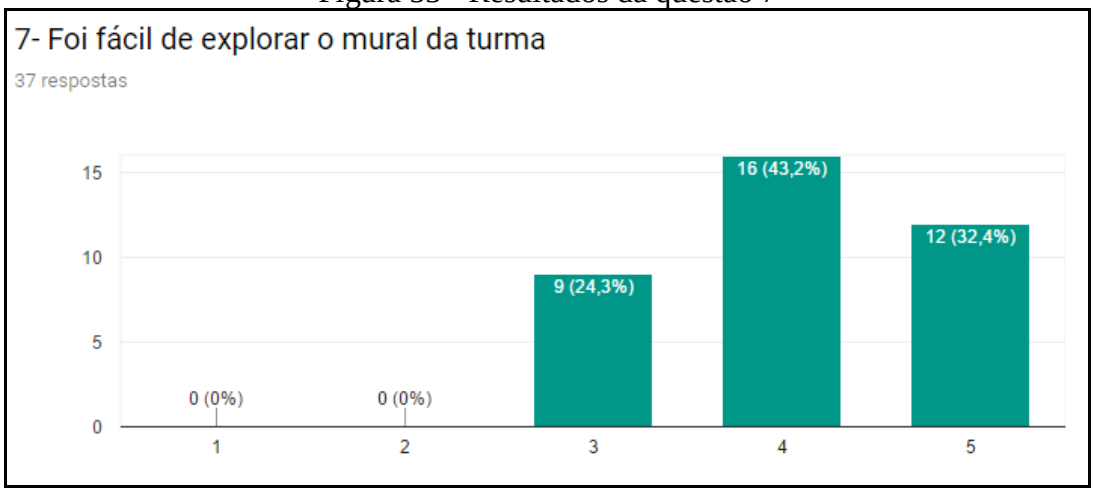


Figura 54 - Observações da criação de um post no mural da turma

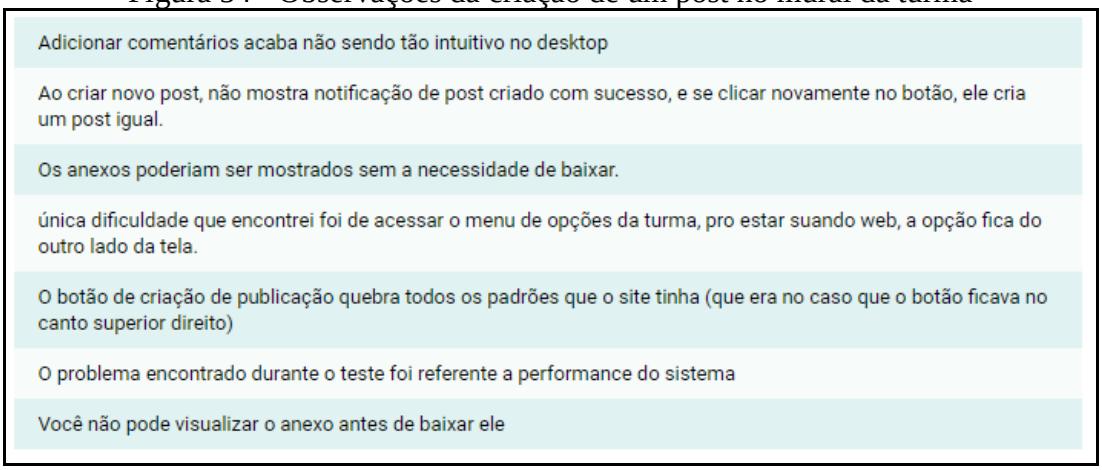


Figura 55 - Resultados da questão 8

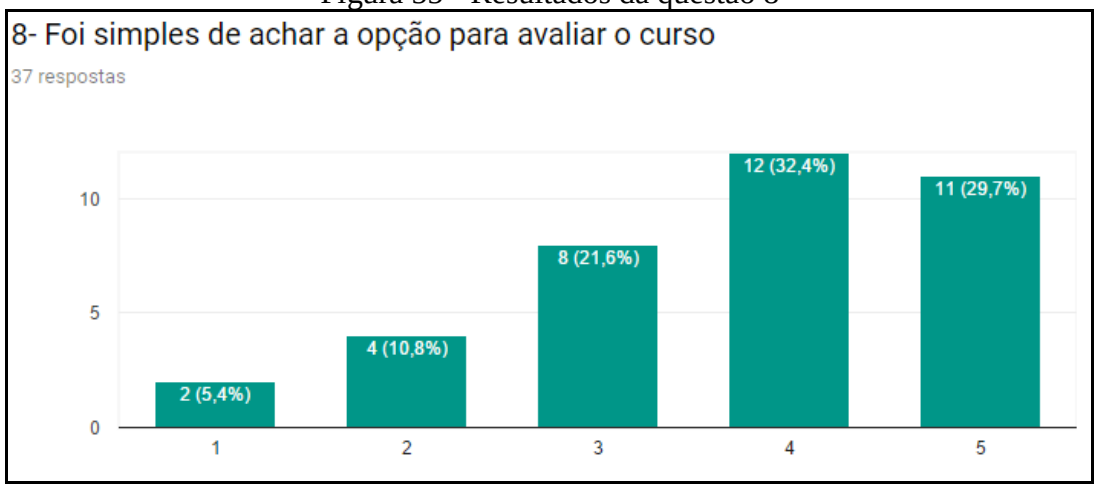


Figura 56 - Resultados da questão 9

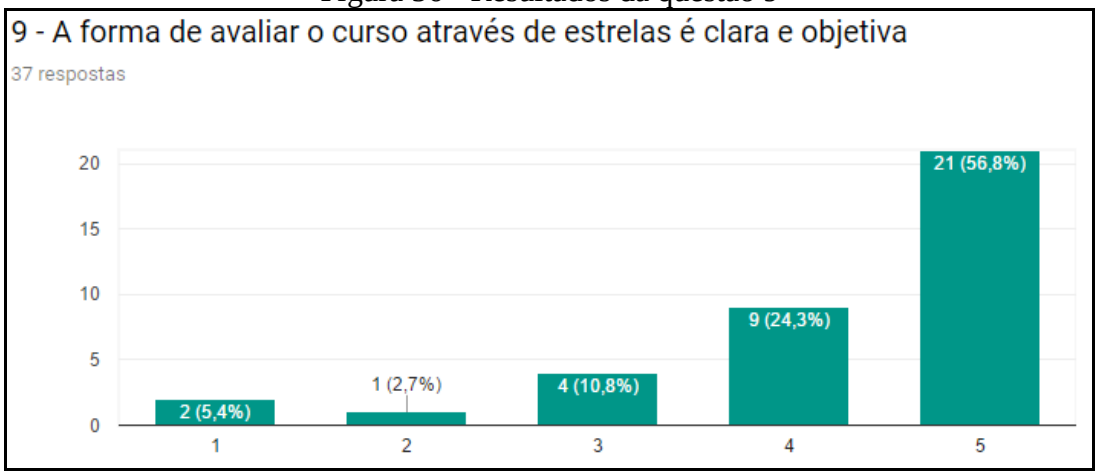


Figura 57 - Resultados da questão 10

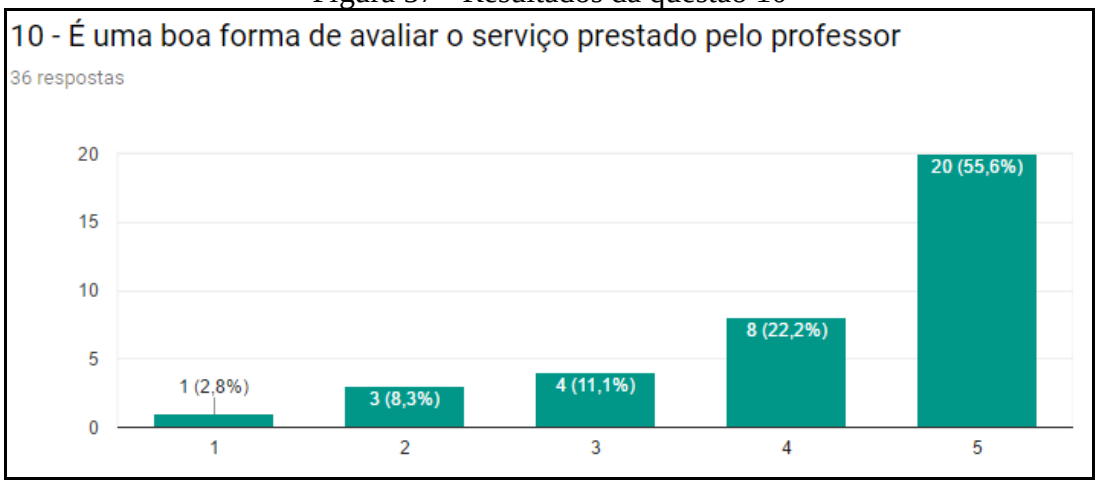


Figura 58 - Observações da avaliação do curso

Podia aparecer as ações direto, sem ter q apertar um botão para isso.
Não apresenta a turma que estou na opção minhas turmas somente nas conversas
Deveria ter mais perguntas(optativas) para esclarecer melhor a avaliação realizada.
Não consegui avaliar
Deveria ter 10 estrelas
Não tem como saber que ao clicar nos 3 pontos aparecerá uma opção de avaliação. O legal seria que além do ícone de 3 pontos, ao entrar na turma aparecesse todas as opções dos 3 pontos... Criar avaliação, consultar avaliações, deixar turma e etc.

Figura 59 - Resultados da questão 11

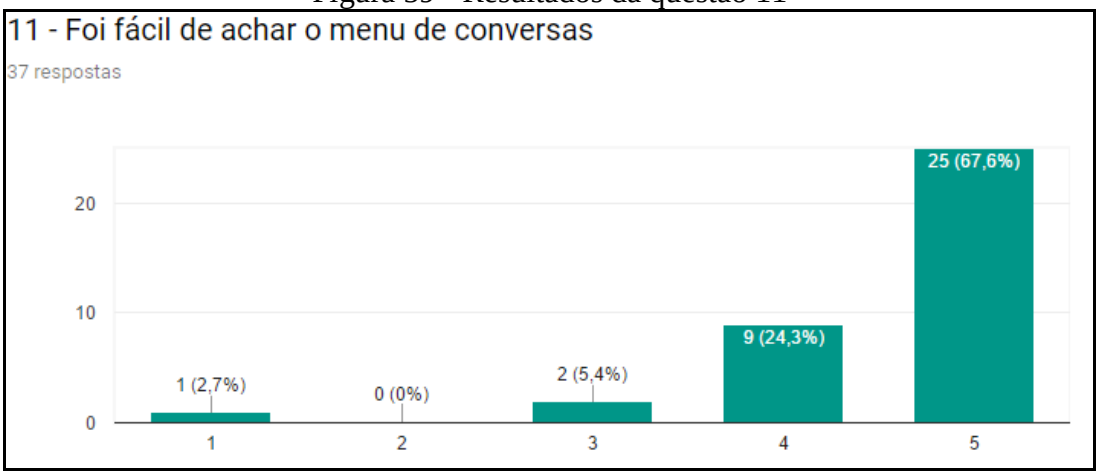


Figura 60 - Respostas da questão 12

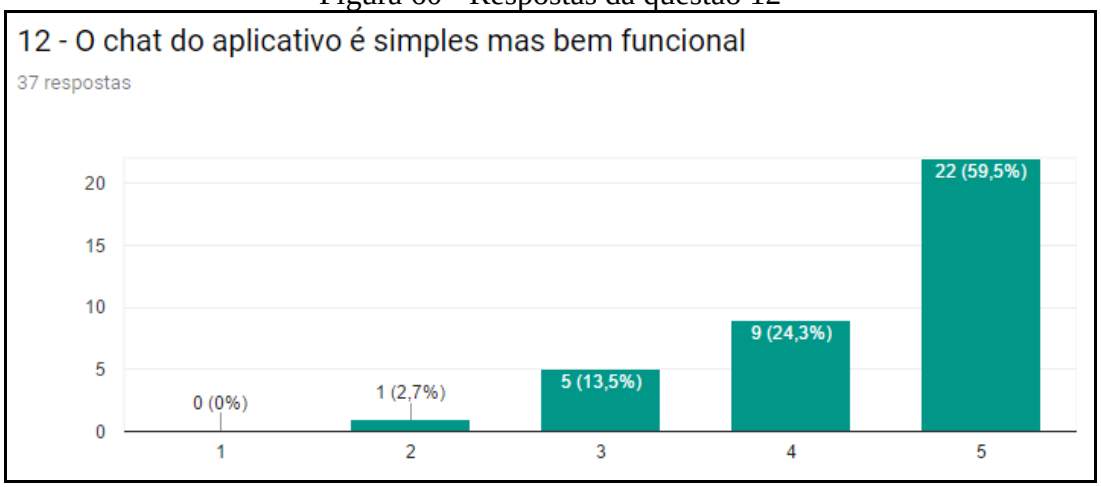


Figura 61 - Resultados da questão 13

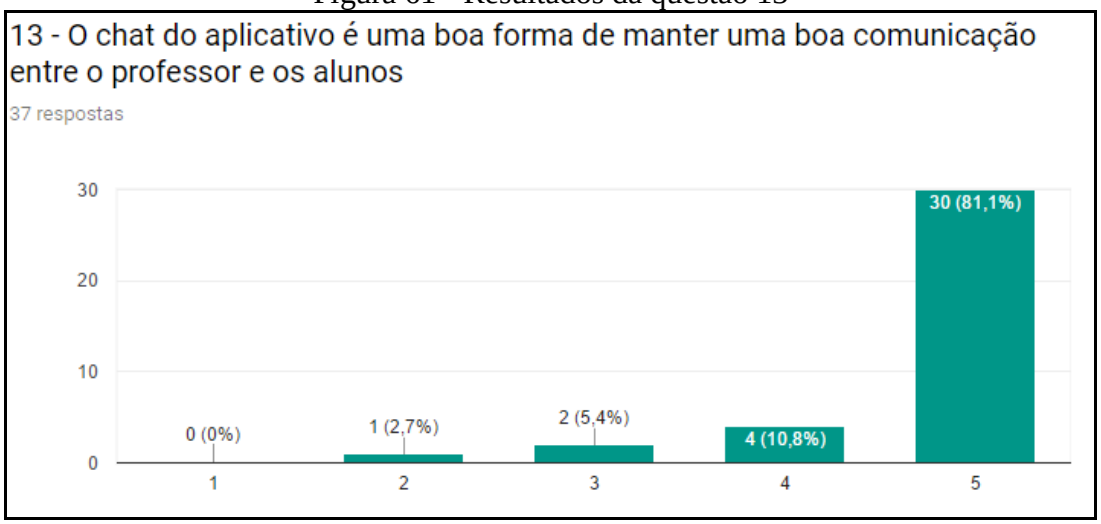


Figura 62 - Pontos positivos da aplicação 1

Fácil de usar (2)
Interface simples
Responsividade, fácil adaptação a zoom in/out, tela mobile, clareza, interface limpa
Se adapta muito bem ao mobile
Usabilidade
Bonito, leve e funciona muito bem em dispositivos móveis.
Para dispositivos mobile ficou 10/10
Aplicativo de interface simples e clara
As interfaces ficam excelentes em mobile
Bem simples
totalmente responsivo
Resposta rápida
Ele é simples e bem prático, fácil acesso as funções.
Fácil de usar; Designer otimizado e clean.

Figura 63 - Pontos positivos da aplicação 2

Fácil de usar; Designer otimizado e clean.
Gostei muito do layout e da ideia
Interface limpa e com cores boas
Simples e limpa
Uma ideia muito boa para ser aplicada, com uma interface interessante pelo fato de ser responsiva.
Aplicação limpa, e cores claras
Simples mas bem utilizável
Design clean, ótimo de ler e bem definido
Simples de usar, intuitivo, não apresentou erros.
Ideia interessante, Responsividade, Simplicidade

Figura 64 - Pontos negativos da aplicação

Alguns botões não estão muito bem destacados, tornando difícil percebe-los, como por exemplo o botão de salvar informações do perfil
Exceções não tratadas no upload de anexos, chat com um pouco de lentidão, menu principal some ao acessar algumas opções
Ao entrar no menu "Minhas Turmas" não apareceu a turma em que fui cadastrado anteriormente.
Alterar dados de usuário tem botão só em cima, as vezes se torna imperceptível.
Para um navegador em um computador a interface não ficou muito intuitiva.
Em alguns pontos há uma demora para carregar as informações, por exemplo para carregar a foto de perfil
O leiaute não fica muito bom para desktop
Tão simples que as vezes você não sabe onde precisa ir, ou desconhece tal função
Notificação na conversa
Bug: ao tentar criar uma turma, e cancelar a criação clicando na seta pra voltar, o grupo é criado mesmo assim
As notificações somem muito rapido, não da tempo para ler o que esta escrito nelas.
Performance
Faltam mensagens de confirmação de ação
Pesado, demorando pra mandar mensagem ou carregar imagem
Performance.
Pouco intuitiva
Pouco intuitiva (versão Google Chrome)
Algumas coisas são um pouco confusas no começo.
Me perdi um pouco
Muitos passos para chegar a um determinado lugar (ex: mural da turma) e se der um voltar não tem salvo que o usuário está logado.
Não chega a ser um ponto negativo, mais por usabilidade, na hora de salvar um post e no cadastro do usuário deixaria a opção 'salvar' abaixo dos quadros preenchidos e as outras opções de menor importância ao lado ou mais discretos. Apenas para ser mais rápido de salvar.

Figura 65 - Observações e sugestões gerais

Acredito que um pouco mais de texto para descrever onde/o que cada botão faz e onde o usuário deveria clicar.
Ao navegar entre as páginas é interessante um feedback avisando que está aguardando, da atual forma parece que clico e não acontece nada.
Ao tentar trocar a senha de usuário está dando mensagem que senha anterior está inválida, mas está correta a senha.
Arrumaria uns bugs que acontecem de interface, botão salvar está muito oculto (na tela de perfil), colocaria uma intro de como usar pela primeira vez (no primeiro acesso)
Em relação as conversas, deveria mostrar que existem mensagens não lidas
Não existe um motivo para aquele menu na pagina inicial onde tem o link para a página inicial se só aparece na pagina inicial
Quando clico em um submenu lateral deveria aparecer uma barra de progresso para que o usuário saiba que esta acontendo algo.
Possibilidade do usuário escolher o que deseja visualizar no mural. Opção na tela de login, para o usuário que esqueceu a senha.
Botões mais visíveis
Talvez um ícone de "home" possa ajudar o usuário a chegar na página inicial quando está no chat. Ficará mais intuitivo que a flecha de voltar.
Enviar a chave por e-mail talvez seria uma ideia interessante