

Pedro Henrique Araújo Sobral

Desenvolvimento de um sistema de resposta para uso em sala de aula

Juazeiro, Bahia, Brasil

2017

Pedro Henrique Araújo Sobral

Desenvolvimento de um sistema de resposta para uso em sala de aula

Trabalho de conclusão de curso apresentado à Universidade Federal do Vale do São Francisco, Campus Juazeiro – BA, como requisito para a obtenção do título de Engenheiro de Computação.

Orientador: Dr. Max Santana Rolemberg Farias

Juazeiro, Bahia, Brasil

2017

Pedro Henrique Araújo Sobral

Desenvolvimento de um sistema de resposta para uso em sala de aula

Trabalho de conclusão de curso apresentado à Universidade Federal do Vale do São Francisco, Campus Juazeiro – BA, como requisito para a obtenção do título de Engenheiro de Computação.

Trabalho _____. Juazeiro, Bahia, Brasil, 25 de Janeiro de 2017:

Orientador (CCOMP/UNIVASF)
Dr. Max Santana Rolemberg Farias

Professor (CCOMP/UNIVASF)
Dr. Brauliro Gonçalves Leal

Professor (CCIVIL/UNIVASF)
Me. João Carlos Sedraz Silva

Juazeiro, Bahia, Brasil
2017

*“Diga-me eu esquecerei,
ensina-me e eu poderei lembrar,
envolva-me e eu aprenderei.”
(Benjamin Franklin)*

Resumo

Sistemas de resposta em sala de aula permitem ao professor um retorno em tempo real sobre o entendimento de toda uma classe sobre um determinado tópico de estudo. Essa informação é valiosa porque permite ao educador, por exemplo, realizar uma avaliação formativa, orientando-o na prática pedagógica. No entanto, o custo associado à aquisição dos sistemas de resposta podem ser proibitivos ao uso. É importante destacar essa tecnologia apenas como um meio para colaborar no processo de ensino e aprendizagem, e só faz sentido quando associada com práticas pedagógicas de ensino como o aprendizado ativo. Dessa forma, nesse trabalho será desenvolvido um sistema de resposta em sala de aula, que possibilite aos professores e aos estudantes uma ferramenta de software livre, que permita usar *smartphones* como sistema de resposta.

Palavras-chave: Aprendizado Ativo. Instrução pelos Colegas. Sistemas de Resposta em Sala de Aula. Aplicativos Híbridos para Celular.

Abstract

Classroom response systems (CRS) or just clickers allow the teacher a real-time feedback on the understanding of a whole class on a topic of study. This information is valuable because it allows the educator, for example, conduct a formative assessment, guiding it in pedagogical practice. However, it is important to highlight this technology only as a means to assist in the process of teaching and learning, and only makes sense when combined with pedagogical practices of teaching and active learning. Moreover, the cost associated with clickers can be prohibitive to use. Thus, this work will develop a CRS that allows teachers and students a free software that will give them the opportunity to use smartphones as clickers so that it can be mainly used for educational practices of active learning as Peer Instruction.

Keywords: Active Learning. Peer Instruction. Classroom Response Systems. Hybrid Mobile Applications.

Lista de ilustrações

Figura 1 – Preço do <i>i>clicker 2</i>	15
Figura 2 – Preço de algumas sistemas de resposta	17
Figura 3 – Diagrama do processo de implementação do método IpC	22
Figura 4 – Exemplo de uma questão conceitual	23
Figura 5 – Exemplo de <i>clickers</i>	23
Figura 6 – Engenharia de Software - uma tecnologia em camadas	24
Figura 7 – Requisitos iniciais para o sistema em forma de histórias de usuário	30
Figura 8 – Arquitetura do sistema	31
Figura 9 – HelloIonicPage: classe responsável por exibir e controlar a página	32
Figura 10 – HelloIonicPage: elementos visuais da página	33
Figura 11 – Página em <i>Ionic</i> resultado das Figuras 9 e 10	33
Figura 12 – Exemplo de componentes no <i>Ionic</i>	34
Figura 13 – Interface de um serviço	35
Figura 14 – Exemplo registro de <i>hooks</i> no <i>FeathersJS</i>	36
Figura 15 – Exemplo eventos no <i>FeathersJS</i>	37
Figura 16 – Como o ciclo de uma requisição funciona no <i>FeathersJS</i>	37
Figura 17 – Exemplo de um documento em MongoDB	40
Figura 18 – Diagrama do banco de dados orientado a documentos	41
Figura 19 – Interface de linha de comando da ferramenta feathers	42
Figura 20 – <i>PollService</i> - serviço para prover informações do serviço de votações	42
Figura 21 – Parte de página de votação da aplicação aluno - <i>poll.ts</i>	43
Figura 22 – Parte de página de votação da aplicação aluno - <i>poll.html</i>	44
Figura 23 – Exemplo de uma suíte de testes no Jasmine	45
Figura 24 – Exemplo de uma suíte de testes no Jasmine com o Protractor	46
Figura 25 – Exemplo de um teste de aceitação automatizado	48
Figura 26 – Relatório de execução dos testes no Jasmine	49
Figura 27 – Resposta em Sala de Aula (RSA)	50
Figura 28 – Aba <i>Questões</i>	51
Figura 29 – Página para a realizar a frequência dos estudantes	51
Figura 30 – Página listando as frequências realizadas de uma turma	52
Figura 31 – Detalhes de uma frequência realizada	52
Figura 32 – Formulário <i>Nova Questão</i>	53
Figura 33 – Detalhes da aba <i>Turmas</i>	53
Figura 34 – Adicionar estudantes em uma turma	54
Figura 35 – Exemplo de arquivo CSV válido para importar uma lista de estudantes	54
Figura 36 – Aba <i>Questões</i>	55

Figura 37 – Tela exibindo uma questão ainda indisponível para os estudantes . . .	56
Figura 38 – Tela com todas as opções de controle ativadas	56
Figura 39 – Tela listando as sessões encerradas	57
Figura 40 – Detalhes de uma sessão encerrada: resposta por questão e por estudantes	57
Figura 41 – Tela inicial da aplicação móvel do estudante	58
Figura 42 – Tela de identificação dos estudantes	59
Figura 43 – Telas para os estudantes realizarem a frequência	60
Figura 44 – Diferentes tipos de questões no aplicativo	60
Figura 45 – Exemplo de uma tarefa do teste de usabilidade	63
Figura 46 – Resultado do questionário SUS	66
Figura 47 – Comparação das classificações adjetivas, pontuações de aceitabilidade e escalas de classificação escolar, em relação ao escore médio do SUS . .	66

Lista de abreviaturas e siglas

API	Application Programming Interface
BDD	Behavior-Driven Development
BJSON	Binary JavaScript Object Notation
CRS	Classroom Response System
CSS	Cascading Style Sheets
CSV	Comma-separated values
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IpC	Instrução pelos Colegas
JSON	JavaScript Object Notation
LMS	Learning Management System
MIT	Massachusetts Institute of Technology
REST	Representational State Transfer
SMS	Short Message Service
SUS	System Usability Scale
TCP	Transmission Control Protocol
URL	Uniform Resource Locator

Sumário

1	INTRODUÇÃO	11
1.1	Objetivo Geral	12
1.2	Objetivos Específicos	12
1.3	Organização do texto	13
2	REVISÃO DE LITERATURA	14
2.1	Sistemas de Resposta em Sala de Aula	14
2.1.1	Alternativas Disponíveis	15
2.2	Benefícios dos Sistemas de Resposta em Sala de Aula	16
2.2.1	Benefícios para a sala de aula	16
2.2.2	Benefícios para a aprendizagem	18
2.2.3	Benefícios para avaliação	19
2.2.4	Desafios para usar CRSs	20
2.3	Aprendizado Ativo	20
2.3.1	Instrução pelos Colegas (IpC)	21
3	ENGENHARIA DE SOFTWARE	24
3.1	Processos de Software	24
3.1.1	Métodos Ágeis	25
3.2	Metodologias e Ferramentas	25
3.2.1	Objetivos específicos	25
3.3	Especificação	26
3.3.1	Análise de competidores	26
3.3.2	Requisitos gerados a partir da análise de competidores	28
3.3.3	Histórias de Usuário	29
3.4	Projeto e Implementação	29
3.4.1	Plataforma	29
3.4.2	Arquitetura	31
3.4.3	<i>WebSocket</i> para aplicações em tempo real	31
3.4.4	Estrutura do Framework: <i>Ionic</i>	31
3.4.5	Estrutura do Framework: <i>FeathersJS</i>	35
3.4.6	Implementação	38
3.4.7	Servidor: <i>FeathersJS</i>	40
3.4.8	Frontend: <i>Ionic</i>	41
3.5	Verificação do Software	45
3.5.1	Testes Automatizados	45

4	RESPOSTA EM SALA DE AULA	50
4.1	Aplicação web para o Professor	50
4.1.1	Frequência dos alunos	50
4.1.2	Cadastro de Questões	52
4.1.3	Turmas	52
4.1.4	Sessão de questões: Aba Aula	55
4.2	Aplicação para dispositivos móveis, que será utilizado como CRSs	58
4.2.1	Identificação dos estudantes	58
4.2.2	Responder frequência	59
4.2.3	Responder questões	59
5	TESTES DE USABILIDADE	61
5.1	Teste de Usabilidade	61
5.1.1	Métodos, tarefas e usuários	61
5.1.2	Resultado e discussões	64
5.2	Teste em Sala de Aula	68
5.2.1	Participantes	69
5.2.2	Preparação antes do teste	69
5.2.3	Aula	70
5.2.4	Questionário	70
6	CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS	71
6.1	Considerações Finais	71
6.2	Trabalhos Futuros	71
	REFERÊNCIAS	72
	APÊNDICES	78
	APÊNDICE A – QUESTIONÁRIO SUS - SYSTEM USABILITY SCALE	79
	APÊNDICE B – TAREFAS - TESTES DE USABILIDADE	80

1 Introdução

Por que ainda se justifica o modelo de ensino em que se baseia na disseminação de informações, que nunca foi tão fácil achar (internet, livros, etc)? O ensino de hoje deveria estar focado para uma nova visão em que o papel do professor seja de intermediar a aprendizagem (ARAUJO; MAZUR, 2013).

Nesse modelo de ensino, o construtivismo, os estudantes são agentes ativos do próprio processo de ensino e construção do conhecimento. Uma aplicação prática desse modelo teórico de ensino é o aprendizado ativo. O aprendizado ativo é um conjunto de práticas pedagógicas que além de envolver os estudantes no fazer, os faça pensar no que estão fazendo, ou seja, além de ouvir, eles precisam ler, escrever, discutir, ou estarem envolvidos na resolução de problemas (CHARLES; JAMES, 1991).

Uma metodologia de aprendizado ativo que tem alcançado sucesso internacionalmente é a Instrução pelos Colegas (IpC) ou *peer instruction* (ARAUJO; MAZUR, 2013). Resumidamente, o IpC promove o aprendizado ativo com questionamentos regulares na aula, fazendo com que os estudantes passem mais tempo discutindo e pensando sobre um conteúdo, do que assistindo passivamente a aula.

Assim como no modelo de ensino tradicional, a tecnologia também pode ser usada para promover de forma mais eficiente o aprendizado ativo. No caso do IpC, a etapa fundamental de votação de uma questão (apresentado em detalhes no [Capítulo 2, subseção 2.3.1](#)), o uso de sistemas de resposta permitem ao professor uma análise posterior dos resultados, além de dinamizar o processo de votação em sala de aula.

Sistemas de respostas são sistemas que possibilitam que todos os alunos respondam a questões apresentadas pelo professor. Geralmente um gráfico de barras é apresentado logo em seguida que os estudantes submetem as suas soluções utilizando algum dispositivo remoto também conhecido como *clickers*. As respostas são anônimas para os seus colegas, no entanto o professor pode identificar cada estudante individualmente pela identificação única do dispositivo, permitindo assim uma análise individual (KAY; LESAGE, 2009).

Não existe na literatura um consenso sobre a nomenclatura para referenciar sistemas de resposta para uso em sala de aula. Pode-se encontrar termos como sistema de resposta do estudante “*student response system*”, sistema de resposta a audiência “*audience response system*”, sistema de resposta pessoal “*personal response systems*”, sistemas de resposta em sala de aula “*classroom response systems*” (CRS), ou apenas como “*clickers*” (HUNSU; ADESOPE; BAYLY, 2016). Nesse trabalho, para simplificação foi utilizado apenas a sigla CRS “*Classroom Response System*” para referenciar esse tipo de sistema.

O uso de CRS associado com um modelo de aprendizado ativo tem demonstrado trazer vários benefícios para a sala de aula, como aumento na frequência escolar (FOTARIS et al., 2016), melhora na atenção dos estudantes (TERRION; ACETI, 2012) e maior engajamento da turma (KAYA; BALTA, 2016). Também promove benefícios para a aprendizagem com um aumento na interação e discussão na sala de aula (MATTOS, 2015; BARRAGUÉS; MORAIS; GUIASOLA, 2011), e com melhora no aprendizado (SUN, 2014; HUNSU; ADESOPE; BAYLY, 2016). Além dos benefícios para avaliação pelo *feedback* imediato do entendimento dos estudantes sobre um determinado conteúdo (RANA; DWIVEDI, 2016; BLOOD; GULCHAK, 2013).

Apesar dos benefícios significativos do uso associado com CRS na sala de aula, o preço de tais tecnologias podem representar um custo econômico considerável para instituições de ensino, que pode se tornar uma barreira para adoção de CRS e adotá-las no processo de aprendizagem (BLASCO-ARCAS et al., 2013).

Nesse sentido, considerando que os *smartphones* podem ser usados como parte integrante dos CRSs como o dispositivo em que os estudantes vão submeter as respostas (ou *clickers*), e da quase ubiquidade de tais dispositivos entre os jovens brasileiros - já em 2014 mais de 93% dos estudantes da rede privada de ensino e quase 65% dos da rede pública possuem *smartphones* (IBGE, 2016, p. 55) -, este trabalho apresenta o desenvolvimento de um CRS de software livre que permita usar *smartphones* como *clickers*.

1.1 Objetivo Geral

Desenvolver um sistema de resposta para uso sala de aula (CRS), que possibilite aos professores e aos estudantes uma ferramenta de software livre, que permita usar *smartphones* como *clickers*.

1.2 Objetivos Específicos

- Realizar levantamento de requisitos sobre os sistemas de resposta em sala de aula;
- Especificar e implementar uma aplicação para dispositivos móveis, que será utilizado como *clickers*;
- Especificar e implementar uma aplicação web para o professor administrar as questões e gerar relatórios;
- Especificar e implementar um sistema servidor, para receber e enviar dados para os os clientes: dispositivos móveis dos alunos e navegador web do professor.

1.3 Organização do texto

Além desta introdução, este trabalho está dividido em mais quatro capítulos.

Revisão de literatura: Esse capítulo discute um contexto pedagógico para o uso de sistemas de resposta em sala de aula. Inicialmente, é apresentado o conceito de aprendizado ativo e um método de implementação: Instrução pelos Colegas. Em seguida, são apontados os sistemas de resposta em sala de aula como uma ferramenta para ajudar o professor a mediar um aprendizado significativo em sala de aula. O capítulo é encerrado apresentando os benefícios e desafios de uso dessa tecnologia.

Engenharia de Software: Nesse capítulo são descritas as fases de especificação, projeto, implementação e testes do sistema. Os procedimentos, material e métodos, resultados e discussões e cada etapa também é apresentado nesse capítulo.

Resposta em Sala de Aula: Nesse capítulo é apresentado o software desenvolvido, definições, as principais telas e característica do sistema desenvolvido.

Considerações finais e trabalhos futuros: São realizadas as considerações finais do trabalho e apontadas diretrizes para trabalhos futuros.

2 Revisão de Literatura

Neste capítulo, inicialmente, é apresentado o conceito de aprendizado ativo e uma alternativa para a implementação dessa metodologia de ensino. Em seguida, é apontado o CRS como uma ferramenta para ajudar o professor na promoção de um aprendizado significativo em sala de aula. O capítulo é encerrado com informações sobre os benefícios e desafios de uso de dessa tecnologia.

2.1 Sistemas de Resposta em Sala de Aula

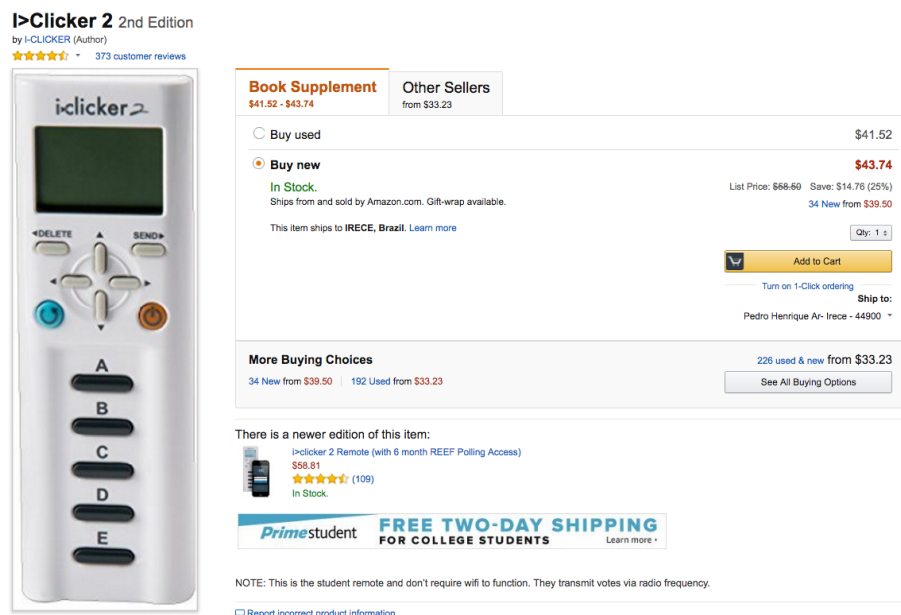
Sistemas de respostas em sala de aula são tecnologias que permitem ao professor realizar questionamentos a toda classe e, assim, obter o resultados das respostas em tempo real (KAY; LESAGE, 2009). Geralmente, as questões são no formato de verdadeiro ou falso e questões de múltipla escolha. Tais sistemas também podem ser usados para saber a impressão dos alunos sobre um determinado tema (FIES; MARSHALL, 2006).

Inicialmente introduzido em 1966 na Universidade de Stanford nos Estados Unidos, os sistemas de resposta não funcionavam direito, eram difíceis de usar e caros (KAY; LESAGE, 2009). Dispositivos de coleta de respostas como os da Figura 5 que usam infravermelho, com preços mais atrativos, começaram a ser extensivamente usados a partir de 2003 e hoje existe pelo menos uma disciplina em cada universidade dos Estados Unidos em que se faz uso de tais sistemas no processo de ensino e aprendizagem (ABRAHAMSON; BRADY, 2014).

Entretanto, ainda que o custo individual de um aparelho seja atrativo para um estudante, a realidade da universidade pública brasileira é diferente em relação a dos Estados Unidos. Assim como não existe obrigatoriedade do aluno comprar um livro-texto no Brasil, também não existe mecanismo que os faça comprar *clickers* (KORTEMAYER; DIAS; CRUZ, 2011). Dessa forma, os *clickers* teriam que ser comprados pela universidade, e o custo total não seria tão atrativo.

Por exemplo, a Universidade de São Paulo, em uma tentativa pioneira de Kortemeyer, Dias e Cruz (2011) de oferecer oportunidades para a avaliação formativa usando *clickers* como os da Figura 5a em duas turmas de física, com 80 alunos cada. Com cada *i>clicker 2* custando \$43,74 (Figura 1) o valor total apenas dos 160 *clickers* seria de R\$22.116,34¹, necessitando para compor o sistema, ainda, o aparelho que recebe as respostas, software que é instalado no computador do professor, treinamento e suporte, ou seja, o valor final pode ser ainda maior. Esse elevado valor de aquisição dos CRSs, é uma das principais barreiras na adoção dessa ferramenta no processo de ensino, que ainda tem a

¹ Considerando o valor do dólar comercial em 12. Ago. 2016 de R\$3,1602. Disponível em: <http://www4.bcb.gov.br/pec/taxas/port/ptaxnpsq.asp?id=txcotacao>

Figura 1 – Preço do *i>clicker 2*

Fonte – [amazon.com](https://www.amazon.com) pesquisando *i>clicker 2*. Acesso em 12. Ago. 2016

resistência de alguns professores em usar novas tecnologias como ferramentas de ensino (MORATELLI; DEJARNETTE, 2014; BLASCO-ARCAS et al., 2013; STRASSER, 2010; KORTEMEYER; DIAS; CRUZ, 2011; KAY; LESAGE, 2009).

Todavia, uma maneira de tornar tal tecnologia mais acessível é usar os próprios celulares dos estudantes como CRSs (STOWELL, 2015; MORRELL; JOYCE, 2015; ARAUJO; MAZUR, 2013). Além de mais acessível, usando os próprios celulares, os estudantes podem ver as questões e os resultados da classe nos próprios aparelhos. Para o professor, uma variedade maior de questões podem ser exploradas, como uma questão aberta (STOWELL, 2015).

Saliente-se ainda que, embora possa haver um pequeno aumento no número de questões sem respostas, as respostas dos estudantes são comparáveis quando se usa *clickers* (MORRELL; JOYCE, 2015; STOWELL, 2015).

Ainda que o uso do celular como CRS possa aumentar o nível de distrações, são muitos os benefícios de usar essa abordagem, no entanto, é preciso estar atento aos pequenos casos de estudantes que não têm a tecnologia adequada, ou que não queiram usar os seus dispositivos (MORRELL; JOYCE, 2015; STOWELL, 2015).

2.1.1 Alternativas Disponíveis

Em um estudo para desenvolver uma estratégia de ensino utilizando CRSs como recurso didático, Mattos (2015), utilizou o *Poll Everywhere*, que foi escolhido principalmente por ser compatível com as principais plataformas móveis (Android, iOS e Windows Phone).

O *Poll Everywhere* oferece uma versão gratuita com no máximo 40 respostas por votação com algumas limitações. Na versão paga, o estudante pode pagar \$14/ano ou o professor pode pagar \$349/semestre (vide Figura 2a).

Outra solução é o *Socrative*, que também é compatível com todas as plataformas, e na sua versão gratuita permite até 50 alunos por votação, questões de múltipla escolha, de verdadeira e falso, resposta curta, *quizzes*, etc (SOCRATIVE, 2016). Alguns relatos do uso do *Socrative* em sala de aula para promover a interatividade são encontrados em (KAYA; BALTA, 2016; TRINDADE, 2014). Uma opção interessante do *Socrative* é o *Exit Ticket*, em que ao final da aula os estudantes respondem a três perguntas. A primeira funciona como uma autoavaliação, perguntando o quanto o estudante entendeu da aula, já as outras duas como identificação, que são o nome do aluno e uma pergunta que o professor coloca no quadro, ou seja, teoricamente, apenas os alunos presentes vão ser capazes de responder corretamente, servindo então como um controle de frequência. Na versão paga o professor pode pagar \$49,99/ano (vide Figura 2b).

Por último, o aplicativo *TopHat*, usado diariamente por mais 500 instituições de ensino pelo mundo (TOPHAT, 2016). Para o professor, *TopHat* permite incorporar uma variedade de questões, uma revisão interativa da aula e para provas (NEILSON et al., 2016; LANTZ; STAWISKI, 2014). A inscrição mais popular no *TopHat* são de \$36/ano (vide Figura 2c).

2.2 Benefícios dos Sistemas de Resposta em Sala de Aula

Os benefícios do uso de sistemas de resposta em sala de aula podem ser divididos em benefícios para a sala de aula, para a aprendizagem e para avaliação (KAY; LESAGE, 2009). As subseções a seguir discutem cada tópico.

2.2.1 Benefícios para a sala de aula

2.2.1.1 Frequência Escolar

Não apenas uma ferramenta que também pode facilitar o controle de frequência (STRASSER, 2010), os sistemas de resposta em sala de aula têm sido utilizados com sucesso para aumentar a frequência escolar (FOTARIS et al., 2016; VELASCO; ÇAVDAR, 2013; PUENTE; SWAGTEN, 2012; MAYER et al., 2009; CALDWELL, 2007). O fato dos estudantes terem tido um retorno imediato das atividades que faziam interativamente na sala de aula, pode ter contribuído para motivar os alunos para irem para as aulas (PUENTE; SWAGTEN, 2012). Outro fator para o aumento da frequência é quando as questões dos CRSs contribuem na nota final, em que o peso pode variar de 5% a 15% que o resultado parece ser o mesmo (CALDWELL, 2007).

Figura 2 – Preço de algumas sistemas de resposta(a) Disponível em: polleverywhere.com/plans/higher-ed: em 16 ago. 2016**Higher education plans**

Over 100,000 Educators use Poll Everywhere in their classrooms and lecture halls

	Higher-Ed Free	Per Student	Per Instructor	Institution-wide
	Free	\$14 year per student	\$349 semester	Custom
	Sign Up	Sign Up	Sign Up	Contact Us
Responses per poll <input type="checkbox"/> Explain	40	Varies	400	Custom
Choose your username	-	✓	✓	✓

(b) Disponível em: socrative.com/pricing: em 16 ago. 2016

K-12 EDUCATORS

HIGHER ED/CORPORATE

SOCRATIVE

Free

GET FREE

★ SOCRATIVE PRO

\$49⁹⁹

ANNUAL PRICING

GET PRO

(c) Disponível em: tophat.com/pricing em: 16 ago. 2016



Top Hat Lecture

Students purchase a subscription to access the real-time platform
[Learn more about Top Hat Lecture](#)

1 Term

\$24

4 Month access to your courses on Top Hat

Get Started

Annual

\$36

Most Popular!

12 Month access to your courses on Top Hat

Get Started

Lifetime

\$72

Lifetime access* to your courses on Top Hat

Get Started

*Access to Top Hat for as long as you're in school

2.2.1.2 Atenção

Sabe-se hoje que a maioria das pessoas não conseguem se concentrar ininterruptamente por mais de 20 min em uma única atividade (CALDWELL, 2007; D'INVERNO; DAVIS; WHITE, 2003). Com aulas que duram entre 2h-3h, os professores podem quebrar as aulas em miniaulas com questões ao final delas usando os CRSs (HUNSU; ADESOPE; BAYLY, 2016). Estudo de ondas cerebrais mostraram que a atenção dos estudantes aumenta durante as atividades de votação, além de reduzir a ansiedade (??). Acrescenta-se também que quando os CRSs são adotados, os estudantes relatam a necessidade de estudar antes da aula para participarem e prestarem atenção, dessa forma participam mais da aula (TERRION; ACETI, 2012).

2.2.1.3 Privacidade e Participação

Os estudantes podem responder as questões sem se preocupar com o julgamento de seus colegas de classe ou do professor. Nesse ambiente seguro o estudante não tem nenhuma razão para sentir medo de responder errado (SCHMIDT, 2011). A falta de privacidade pode coibir a completa honestidade na votação (CALDWELL, 2007), e a presença da mesma da a oportunidade para os estudantes responderem e tomarem posições polêmicas em questões sensíveis (RANA; DWIVEDI, 2016).

2.2.1.4 Engajamento

Inúmeros estudos indicam estudantes mais interessados ou engajados no material apresentado quando sistemas de resposta são usados (KAYA; BALTA, 2016; RANA; DWIVEDI, 2016; HORNE, 2015; MATTOS, 2015; MORATELLI; DEJARNETTE, 2014; KULATUNGA; RAMEEZDEEN, 2014; BLOOD; GULCHAK, 2013; TERRION; ACETI, 2012; CALDWELL, 2007).

2.2.2 Benefícios para a aprendizagem

2.2.2.1 Interação e Discussão

A interação na sala de aula é importante por três motivos: primeiro pelo que foi apresentado na [subseção 2.2.1.2](#), segundo que permite clarificar pontos obscuros e terceiro que permite uma monitoração do entendimento da classe e velocidade do que é apresentado (D'INVERNO; DAVIS; WHITE, 2003). Nesse sentido, estudos indicam aumento da interação na sala de aula (MATTOS, 2015; BARRAGUÉS; MORAIS; GUIASOLA, 2011; TITMAN; LANCASTER, 2011; MAYER et al., 2009; CALDWELL, 2007).

2.2.2.2 *Contingent Teaching*

Contingent Teaching nada mais é do que a possibilidade de modificar o curso do que está sendo ensinado com base no *feedback* dos estudantes (ARNESEN et al., 2013; CALDWELL, 2007). Por exemplo, se pouco mais da metade da classe responde corretamente a uma questão, conclui-se uma falta de compreensão geral do tópico apresentado, e assim o professor pode tentar explicar o conceito de maneira diferente (TERRION; ACETI, 2012; STRASSER, 2010).

2.2.2.3 Melhora no aprendizado

Lantz e Stawiski (2014) verificaram um aumento significativo (15%) no grupo de participantes que tiveram aulas com CRSs em relação ao sem o uso em testes que ocorriam dois dias depois as aulas, e também permitiu aos estudantes corrigir eventuais equívocos sobre o material de aula. Outros estudos apontam melhora na performance dos estudantes (COLDWELL, 2007). Entretanto, em um recente estudo de meta-análise, em que se comparou classes que usaram e não usaram sistemas de resposta em sala de aula, indicou um significativo impacto positivo nas variáveis cognitivas (retenção, transferência de conhecimento e sucesso), e nas variáveis não-cognitivas (engajamento e participação, frequência, interesse, percepção de qualidade) (HUNSU; ADESOPE; BAYLY, 2016).

2.2.3 Benefícios para avaliação

2.2.3.1 *Feedback*

Como destacado na subseção 2.3.1, dentre as quatro formas disponíveis para o professor obter um *feedback* ou retorno do entendimento dos estudantes sobre determinado tópico, se disponível, os CRSs são os mais eficazes (CROUCH et al., 2007). Os CRSs permitem um retorno imediato, preciso, privacidade (vide subseção 2.2.1.3) e são divertidos de usar (RANA; DWIVEDI, 2016; BLOOD; GULCHAK, 2013; CALDWELL, 2007).

2.2.3.2 Avaliação Formativa

Avaliar o estudante permanentemente, e não apenas utilizando avaliações pontuais, não apenas classificar os alunos em os que sabem e os que não sabem, não apenas verificar e sim avaliar, dedicar mais tempo nos erros (UNIVESPTV, 2013). Esses são alguns pontos da avaliação formativa, que para o professor orienta a prática pedagógica, e para o aluno permite uma autoavaliação sobre a aprendizagem (KAY; LESAGE, 2009). Os artigos a seguir indicam que o uso de sistemas de resposta em sala de aula ajudam a fornecer avaliações formativas eficazes (KORTEMAYER, 2016; THAMPY; AHMAD, 2014; KAY; LESAGE, 2009; FIES; MARSHALL, 2006).

2.2.4 Desafios para usar CRSs

O uso dos CRSs também trazem alguns desafios, principalmente tecnológicos ou estruturais e desafios centrado nos estudantes (CUBRIC; JEFFERIES, 2015; KAY; LESAGE, 2009).

2.2.4.1 Tecnológicos/Estruturais

Stowell (2015) recomenda que as salas de aula tenham acesso confiável a rede Wi-Fi, por isso o professor deve estar preparado quando a internet não estiver disponível (STRASSER, 2010). Como já abordado nesse trabalho (vide seção 2.1), nem todos os estudantes vão ter a tecnologia adequada ou vontade de usar (MORRELL; JOYCE, 2015; STOWELL, 2015). Dessa forma, Velasco e Çavdar (2013) aconselham os CRSs como uma ferramenta opcional para os estudantes.

2.2.4.2 Desafio para os estudantes

Alguns estudantes podem ter resistência ao novo, principalmente se já acomodados a passividade em sala de aula (TERRION; ACETI, 2012; KAY; LESAGE, 2009). Além disso, quando usados apenas para controle de frequência, os estudantes podem ter uma percepção negativa da tecnologia (TERRION; ACETI, 2012; CALDWELL, 2007).

2.3 Aprendizado Ativo

Apesar das seções anteriores serem animadoras quanto ao uso dos CRSs é importante destacar a tecnologia apenas como um meio para colaborar no processo de ensino e aprendizagem, e só faz sentido quando associada com práticas pedagógicas de ensino como o aprendizado ativo (TERRION; ACETI, 2012; MORAN; MASETO; BEHRENS, 2006). O leitor interessado sobre o uso das novas tecnologias em sala de aula, poderá buscar nas obras indicadas em (BRASIL, 2014).

Em resumo, o aprendizado ativo é um conjunto de práticas pedagógicas que além de envolver os estudantes no fazer, os faça pensar no que estão fazendo (CHARLES; JAMES, 1991, p. 19).

Os princípios do aprendizado ativo são dois: introduzir atividades nas salas de aula tradicionais e promover o envolvimento dos estudantes. O primeiro é a forma mais simples do aprendizado ativo. Um exemplo seria fazer pequenas pausas na aula e colocar os estudantes para revisar as notas de aula com um colega. No entanto, é preciso que tais atividades promovam o engajamento dos estudantes no processo de ensino e aprendizagem (PRINCE, 2004, p. 3).

Em salas de aula que utilizam o aprendizado ativo pode-se ter um aumento de até 6% na média final nas provas dos alunos e que aquelas salas de aula puramente expositivas têm 55% mais chance de reprovarem os alunos do que aquelas com aprendizado ativo. Esse aumento que pode parecer pouco (0,3 na média final) colocaria a média daqueles estudantes que desistem do curso bem próximo daqueles que permanecem, podendo-se dessa forma aumentar a taxa de retenção dos estudantes (FREEMAN et al., 2014, p. 4).

O estudo de meta-análise de Freeman et al. (2014) envolveu mais de 200 artigos, que comparavam as performances dos alunos em salas de aula com pelo menos algum elemento de aprendizado ativo com as tradicionais aulas expositivas. Além de mostrarem evidências de que o aprendizado ativo pode melhorar o aprendizado dos estudantes de graduação, principalmente nas áreas de ciência, tecnologia, engenharia e matemática, Freeman et al. propõem aos futuros pesquisadores testarem não mais a eficiência dos métodos de aprendizado ativo frente as tradicionais aulas expositivas (“primeira geração de pesquisas”), mas sim qual o tipo de aprendizado ativo é mais apropriado para cada área do conhecimento (“segunda geração de pesquisas”).

Apesar dos indícios dos benefícios da Aprendizagem Ativa, no contexto brasileiro, tornar o aluno um agente ativo no processo de ensino e aprendizagem não é uma tarefa fácil. São muitas as adversidades de infraestrutura e institucionais encontradas de modo a propiciar o desenvolvimento dessa metodologia. Seja por salas com numero excessivo de alunos, estes desinteressados, professores mal pagos, ou com a pressão de produzir cientificamente (ARAUJO; MAZUR, 2013).

Por outro lado, muitas são as iniciativas encontradas na literatura mostrando resultados satisfatórios que podem ajudar o professor nesse processo (CROUCH; MAZUR, 2001; GOK, 2013; BARROS et al., 2004). Na próxima seção será apresentado o método ativo de ensino *Peer Instruction* ou *Instrução pelos Colegas* (IpC).

2.3.1 Instrução pelos Colegas (IpC)

O IpC foi desenvolvido pelo Professor Eriz Mazur da Universidade de Harvard nos Estados Unidos, na década de 1990. O objetivo do IpC é fazer com que todos os estudantes se engajem em discussões com o vizinho de opinião diferente sobre um determinado conceito e fazer com que cada estudante tente explicar o conceito um para o outro (MAZUR, 2009).

No método IpC, geralmente, o professor começa fazendo uma breve exposição dialogada do conteúdo (15min). Depois é colocado para os estudantes uma questão conceitual, que é desenvolvida de modo a avaliar o entendimento dos estudantes sobre um tópico. A Figura 3 resume o processo de implementação do método IpC na sala de aula.

Um exemplo de uma questão conceitual de introdução a física é mostrada na Figura 4. Os estudantes respondem individualmente a questão (1-2min), geralmente,

utilizando *clickers*, que são pequenos dispositivos transmissores como os da Figura 5. Em seguida, dependendo do percentual de alunos que acertem a questão, o professor pode revisar o assunto (acerto $< 30\%$), fazer uma breve explanação da questão e ir para um próximo tópico ou nova questão (acerto $> 70\%$), ou o que se deseja do método, um percentual de acerto entre 30% e 70% em que, nessa situação, o professor estimula que os alunos encontrem um parceiro que respondeu de forma diferente e que tentem explicar um para o outro o porquê de estar correto. Com esse processo, os alunos se auto instruem, o que justifica o nome do método “Instrução pelos Colegas” (MAZUR, 2009; CROUCH; MAZUR, 2001).

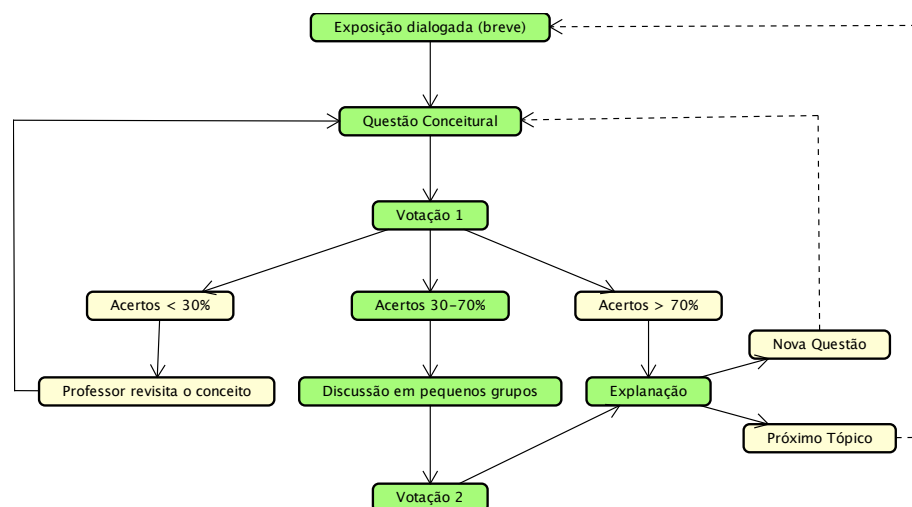
Uma das etapas do IpC na sala de aula é a votação, em que os estudantes indicam as suas respostas. Existem pelo menos quatro maneiras (CROUCH et al., 2007) do professor obter um *feedback* da votação dos estudantes:

Levantar as mãos: a forma mais simples é pedir para os alunos levantarem as mãos para cada alternativa, mas, dentre as várias limitações desse método, os estudantes podem se influenciar pela resposta dos outros.

Cartões coloridos: uma segunda alternativa seria o uso de cartões coloridos (*flashcards*) que dificultaria os alunos ver a resposta dos outros e facilitaria a contagem pelo professor, porém uma limitação desse método, assim como o anterior é a dificuldade de alguma forma guardar os resultados.

Folha de respostas: outra alternativa, no entanto não seria possível ter um resultado imediato das respostas.

Figura 3 – Diagrama do processo de implementação do método IpC

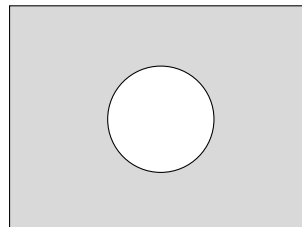


Fonte – Adaptado de (ARAUJO; MAZUR, 2013)

Figura 4 – Exemplo de uma questão conceitual

Considere uma placa de metal retangular com um furo circular no centro. Se a placa for uniformemente aquecida, o diâmetro do buraco:

1. aumenta;
2. permanece o mesmo;
3. diminui;



Fonte – Adaptado de (WATKINS; MAZUR, 2013)

Figura 5 – Exemplo de *clickers***(a)** *i>clicker 2***(b)** *ResponseCard RF LCD*

Fonte – (a) iclicker.com (b) turningtechnologies.com

Sistemas de resposta em sala de aula: exemplo os *clickers* da Figura 5, ou *smartphones*, e sistemas web possibilitam aos estudantes enviarem imediatamente as respostas ao computador do professor, de forma anônima aos colegas de sala e visualizar graficamente os resultados.

3 Engenharia de Software

3.1 Processos de Software

Engenharia de Software pode ser definida como:

1. the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software [...] 2. the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software (SYSTEMS..., 2010).

A engenharia de software deve ter foco na qualidade, que apoia as outras camadas dessa tecnologia, que são as camadas de processo, métodos e ferramentas [Figura 6](#). A camada de processo define um conjunto de atividades ou um arcabouço que tem como finalidade garantir a efetiva utilização da tecnologia engenharia de software, que dessa forma leva à produção de um software. Os detalhes de como fazer o software pertencem a camada de métodos. Os métodos da engenharia de software incluem tarefas de planejamento e estimativa de software, análise de requisitos, modelagem de projeto, codificação, testes e manutenção. As ferramentas de engenharia de software auxiliam as camadas de processo e métodos, com ferramentas automatizadas, que por sua vez, quando integradas, é estabelecido um suporte ao desenvolvimento de software chamado CASE - *Computer Aided Software Engineering* ([PRESSMAN, 2009](#); [SOMMERVILLE, 2011](#)).

Entre o conjunto de atividades definidas pela camada de processo, quatro são fundamentais, a saber, especificação de software, projeto e implementação de software, validação de software e evolução de software. Especificação de software ou engenharia de requisitos é uma fase importante e crítica do processo de engenharia de software. Importante porque é uma análise de requisitos bem feita que possibilitará atender as demandas dos usuários. Crítica porque um sistema mal especificado, pode até ser bem

Figura 6 – Engenharia de Software - uma tecnologia em camadas



Fonte – ([PRESSMAN, 2009](#))

projetado e construído, mas não vai atender as necessidades dos usuários. Em seguida, na fase de projeto e implementação os requisitos são projetados e programados, tendo como resultado um sistema executável. Depois, o software deve ser verificado para mostrar que atende às demandas dos usuários (validação do software). Finalmente, na fase de evolução de software, o mesmo é modificado devido às mudanças de requisitos e às necessidades dos usuários.

3.1.1 Métodos Ágeis

Contrapondo-se aos modelos prescritivos em que propoem especificar por completo os requisitos do sistema e só então projetar, construir e testar o sistema, surgiu os métodos ágeis, que têm como filosofia o manifesto ágil (SOMMERVILLE, 2011). Esse manifesto afirma (BECK et al., 2001):

Estamos descobrindo melhores maneiras de desenvolver softwares, fazendo-o e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais:

- Indivíduos e interações do que processos e ferramentas
- Software em funcionamento do que documentação abrangente
- Colaboração do cliente do que negociação de contrato
- Resposta a mudanças do que seguir um plano

Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda.

Abordagens ágeis incluem *Extreme Programming* e o *Scrum*. Eles propõem diferentes processos para que tenha-se um desenvolvimento e entrega incremental do sistema, tendo em comum princípios baseados no manifesto ágil.

3.2 Metodologias e Ferramentas

A metodologia escolhida nesse projeto levou em consideração as necessidades de um trabalho de conclusão de curso no curto prazo e os recursos limitados. Dessa forma, uma abordagem baseada em metodologias ágeis foi utilizada para especificação, projeto e implementação do software. Nesse sentido, por exemplo, na fase elicitação de requisitos não procurou-se a completa definição dos requisitos do software e ela nem foi uma fase, a elaboração contínua dos requisitos fez parte do projeto de desenvolvimento como um todo.

3.2.1 Objetivos específicos

Para cada objetivo específico deste trabalho, as seguintes técnicas e ferramentas foram utilizadas:

3.2.1.1 Realizar levantamento de requisitos sobre os sistemas de resposta em sala de aula

A elicitação inicial de alto nível dos requisitos do sistema utilizou a análise de competidores que consistiu basicamente em buscar em alguns sistemas de resposta existentes, referências positivas e negativas para definição do modelo a ser proposto.

3.2.1.2 Especificar e implementar uma aplicação web para o professor administrar as questões e gerar relatórios

Os requisitos iniciais gerados na análise de competidores, foram então transformados em tarefas, descritas como histórias de usuário, com critérios de aceitação. A linguagem de programação usada na fase de implementação do aplicativo foi JavaScript, tendo como auxílio o *framework Ionic*.

3.2.1.3 Especificar e implementar uma aplicação para dispositivos móveis, que será utilizado como CRSs

Os requisitos iniciais gerados na análise de competidores, foram então transformados em tarefas, descritas como histórias de usuário, com critérios de aceitação. A linguagem de programação usada na fase de implementação do aplicativo foi JavaScript, tendo como auxílio o *framework Ionic*.

3.2.1.4 Especificar e implementar um sistema servidor, para receber e enviar dados para os clientes: dispositivos móveis dos alunos e navegador web do professor

O sistema servidor foi desenvolvido utilizando tecnologias como *Node.js*, *MongoDB*, e o *framework FeathersJS*. Tais tecnologias foram utilizadas por permitir o fácil desenvolvimento de aplicações web de tempo real entre o servidor e os seus clientes.

A teoria mais aprofundada sobre os métodos e ferramentas citadas serão descritas nas próximas seções.

3.3 Especificação

A fase inicial do projeto foi a de planejamento para a definição inicial de alto nível dos requisitos do sistema. Nessa etapa, utilizou-se a análise de competidores para elucidar os requisitos que posteriormente foram descritos como histórias de usuário.

3.3.1 Análise de competidores

A análise de competidores é uma técnica oriunda engenharia da usabilidade que consiste em avaliar produtos concorrentes em busca de pontos positivos e negativos. Tal

técnica é útil no levantamento de requisitos de um novo sistema, identificação de pontos fortes e fracos os produtos, reutilização de design, dentre outros (SCHADE, 2013).

Avaliar produtos concorrentes é valioso, porque oferece a oportunidade de novos produtos evitarem problemas existentes dos competidores, explorar os pontos fracos, além da reutilização dos pontos positivos (SCHADE, 2013).

Nesse sentido, a análise de competidores foi utilizada neste trabalho para elicitar requisitos e boas práticas de design de interfaces. Os resultados obtidos foram utilizados no desenvolvimento do software.

3.3.1.1 *Socrative*

Socrative é um sistema de resposta específico para usar em salas de aula. O sistema pode ser acessado pelo site ou nos aplicativos para *iOS* e *Android*. No *Socrative*, apenas o professor precisa fazer um cadastro no site (questões demográficas são solicitadas). Existe uma versão gratuita e paga do aplicativo.

Na conta do professor, é possível criar questionários de múltipla escolha, verdadeira e falso e de questões abertas. Quando o professor cria uma conta, é gerado um código de identificação para que os alunos possam entrar na sala virtual. Na interface do estudante, é necessário colocar o código de identificação do professor.

3.3.1.2 *PollEverywhere*

O *PollEverywhere* é um sistema de resposta mais genérico, possibilitando fazer votações em shows e apresentações diversas. Possibilita integração com ferramentas de apresentação como o *PowerPoint*. Outra característica é a possibilidade dos usuários votarem por SMS. Além dos tipos de questões básicas, o *PollEverywhere* permite criar nuvem de palavras e questões com imagens clicáveis.

A conta do usuário é associado com uma URL, em que é usada para os participantes da votação entrarem e votarem.

3.3.1.3 *TopHat*

TopHat é outra solução voltada para a educação, contando com seis tipos de questões. Adicionalmente o *TopHat* permite ao professor fazer a chamada dos estudantes, isso porque o professor pode gerar um código aleatório no *TopHat* para que os estudantes presentes possam enviar o código e marcar presença. O produto também disponibiliza uma sala de discussão e a possibilidade de criar slides dentro do aplicativo.

Tabela 1 – Análise de Competidores

Caraterística	<i>PollEverywhere</i>	<i>TopHat</i>	<i>Socrative</i>
Open-Source	Não	Não	Não
Integração com LMS	Blackboard	Excel	MasteryConnect
Formatos	JSON, RSS, CSV	Não	Não
Read-only API	Sim	Não	Não
Integração com PowerPoint	Possibilita	Não	Não
Métodos de votação	SMS, web	SMS, Web	Internet
Tempo-real	Sim	Sim	Sim
Acesso ao sistema	URL	Código de Acesso	Código de Acesso
Tipos de questões	5	7	3
Mínimo de passos para votação	2	3	4
Anonimato	Possibilita	Possibilita	Possibilita
Contagem regressiva	Possibilita	Possibilita	Não
Download CSV	Possibilita	Não	Não
Relatórios por estudante	Possibilita	Possibilita	Não

Fonte – do autor (2017)

3.3.2 Requisitos gerados a partir da análise de competidores

A partir das informações coletadas na análise de competidores, foram extraídos um conjunto de requisitos iniciais para o sistema. Os requisitos funcionais gerados pela análise dos competidores foram então descritos como histórias de usuário. A seguir descreve-se os requisitos gerados:

Integração com sistemas LMS : O sistema deve permitir integração com sistemas LMS (Amadeus, Moodle);

Todas as plataformas: é muito importante que o sistema seja capaz de funcionar em smartphones, tables e computadores independentemente do sistema operacional.

Questões abertas, verdadeiro/falso e de múltipla-escolha: O sistema deve fornecer pelo menos esses três tipos básico de questões;

Modo de votação: O sistema deve permitir votação anônima ou requisitar a identificação;

Customização das questões: O sistema deve permitir inserção de equações matemáticas (\LaTeX), imagens e texto como opção das questões;

Controle da votação: Opções básicas como ativar ou desativar a votação e limpar uma votação em andamento;

Controle de frequência: o sistema gera um código aleatório ou uma questão trivial, em que o professor pode solicitar que os estudantes respondam, contando como controle de frequência. Os dados devem ser facilmente exportados para CSV.

Tempo-real: No momento da votação, o professor pode escolher entre apresentar o resultado em tempo-real, quando todos votarem, ou quando determinado;

Banco de questões: As questões elaboradas pelo professor podem ser armazenadas em um banco de questões que o sistema deve manter;

Facilidade do uso e de criação de votação: O sistema não deve oferecer dificuldades de uso e de criação de questões;

Código de acesso: O sistema deve gerar um código de acesso único para identificar o ambiente do professor, usado para que os alunos respondam.

3.3.3 Histórias de Usuário

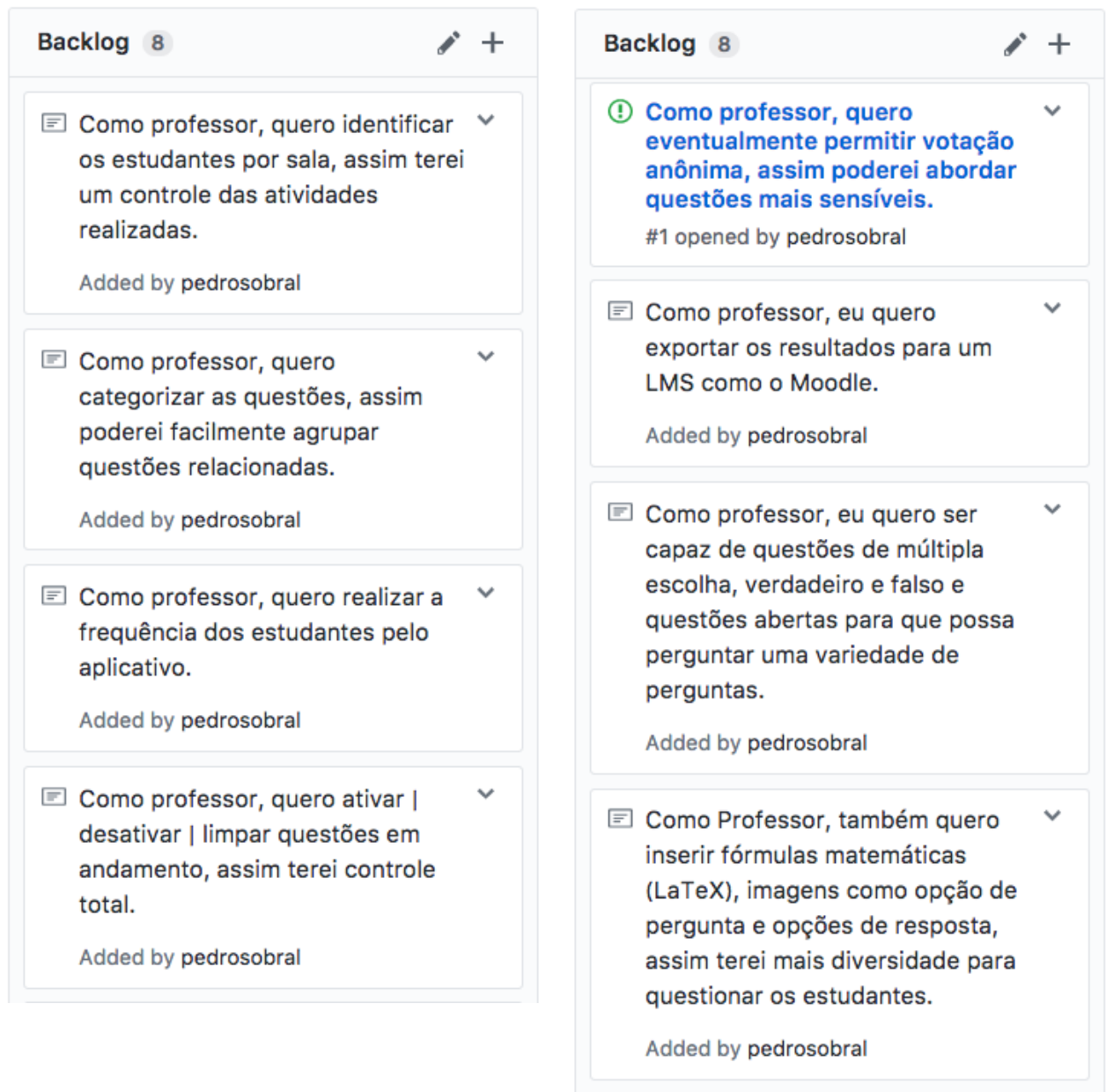
A metodologia ágil de software, *eXtreme Programming (XP)*, introduziu a prática de expressar os requisitos de software na forma de *histórias de usuário*, que são descrições informais do que o sistema deve fazer, evitando qualquer terminologia técnica (SOMMERVILLE, 2011).

As histórias de usuário (Figura 7), formaram a lista de tarefas do projeto. Essa lista de tarefas foi então priorizada, de forma que, por exemplo, desenvolver a arquitetura que possibilita-se ao professor apresentar uma questão no quadro e habilitar para os alunos responderem foi a primeira tarefa a ser desenvolvida. Por outro lado, a tarefa de permitir categorizar as questões para permitir um agrupamento de questões teve uma prioridade baixa.

3.4 Projeto e Implementação

3.4.1 Plataforma

JavaScript: é uma linguagem de programação leve, interpretada e orientada a objetos com funções de primeira classe (funções no JavaScript podem ser passadas como argumento para outras funções, pode ser o valor retornado por outras funções e ainda podem ser atribuídas para variáveis). Ela é uma linguagem de scripting baseada em protótipos, multi-paradigma e dinâmica, suportando os estilos orientado a objetos, imperativo e funcional. Uma das implementações ou *engine* mais populares de JavaScript é o V8 da Google que é utilizada pelo navegador Google Chrome e também pelo Opera (NETWORK, 2017).

Figura 7 – Requisitos iniciais para o sistema em forma de histórias de usuário

Fonte – do autor (2017)

Node.js: é uma plataforma construída sobre o *engine* V8 da Google para construir aplicações de rede rápidas e escaláveis. Node.js usa um modelo de I/O direcionada a evento não bloqueante que o torna leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos (FOUNDATION, 2017).

MongoDB: é um sistema de gerenciamento de banco de dados orientado à documentos. Ele é classificado como um banco de dados *NoSQL*, ou seja, o mecanismo de armazenamento e recuperação é modelado de outras formas além da forma relacional.

O MongoDB usa o modelo de dados JSON para mapear as aplicações de forma simples e rápida (MONGODB, 2017).

3.4.2 Arquitetura

A Figura 8 exibe a arquitetura do sistema. Ela consiste dos clientes (aplicação professor e aplicativo dos alunos), e do servidor desenvolvido na plataforma Node.js com o *framework* *FeathersJS* que recebe as requisições dos clientes. A comunicação entre o nó cliente e o nó servidor é por meio do protocolo *WebSocket*. O servidor faz a interface com o banco de dados MongoDB.

Figura 8 – Arquitetura do sistema



Fonte – do autor (2017)

3.4.3 WebSocket para aplicações em tempo real

WebSocket é um protocolo que possibilita abrir um canal interativo de comunicação entre o navegador e o servidor. Na verdade, esse canal é bidirecional (*full-duplex*) que utiliza apenas um soquete TCP (WEBSOCKET, 2016). A tecnologia *WebSocket* foi usada para permitir votação e controle de frequência em tempo-real.

3.4.4 Estrutura do Framework: *Ionic*

Ionic é um *framework open-source* para o desenvolvimento de aplicativos híbridos utilizando tecnologias web como HTML, CSS e JavaScript otimizadas para dispositivos móveis, com código fonte sobre a licença MIT (DRIFTY, 2016). Uma das principais vantagens do desenvolvimento de aplicativos híbridos é que com apenas um código base é possível criar aplicativos para várias plataformas como *iOS*, *Android* e *Windows Phone*, Desktop, que aliás foi uma das razões que fez o Moodle usar o *Ionic* como *framework* para o desenvolvimento do *Moodle Mobile 2* (MOODLE, 2015).

3.4.4.1 Pages

Um aplicativo desenvolvido no *Ionic* é composto por um conjunto de pages ou páginas. Cada página é composta por alguns arquivos. Um arquivo é responsável pelo elemento visual da página, desenvolvido em HTML. Existe o arquivo de estilos da página, desenvolvido em CSS. O arquivo principal é o responsável por controlar a página, desenvolvido em TypeScript.

As Figuras 9 e 10 são um exemplo básico de uma página de um aplicativo desenvolvido em *Ionic*. O resultado dessa página é mostrado na Figura 11. Observe que com apenas um código base, os elementos visuais da página são diferentes dependendo da plataforma (*iOS*, *Android* e *Windows*).

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'page-hello-ionic',
5    templateUrl: 'hello-ionic.html'
6  })
7  export class HelloIonicPage {
8    constructor() {}
9  }
```

Figura 9 – HelloIonicPage: classe responsável por exibir e controlar a página

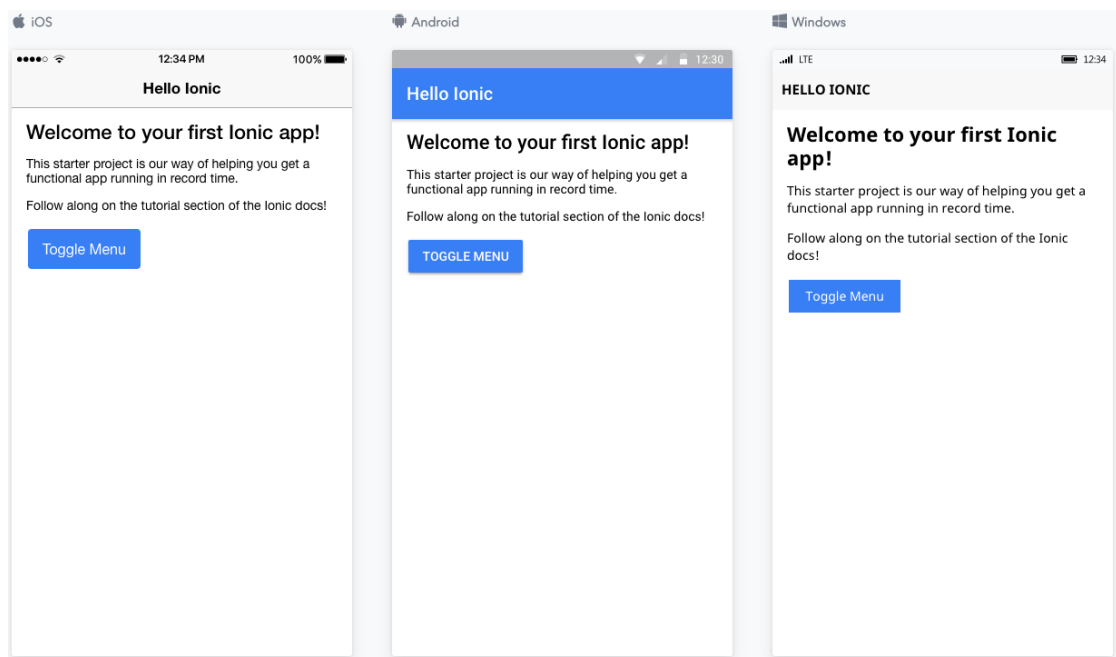
3.4.4.2 Components

Os elementos visuais e também o comportamento desses elementos em uma página são normalmente construídos por meio dos *components* ou componentes. Os componentes permitem criar facilmente a interface do aplicativo. Exemplo de componentes são botões, *modals*, *popup* e *cards*. Um aspecto interessante é que os componentes se adaptam visualmente a cada plataforma, como já mostramos na Figura 11. Além do aspecto visual eles também se comportam de maneira diferente dependendo da plataforma. Por comportamento, entende-se, por exemplo, os efeitos visuais de cada componente e também efeitos de transição entre as páginas. A Figura 12 ilustra alguns componentes disponíveis no *Ionic*.

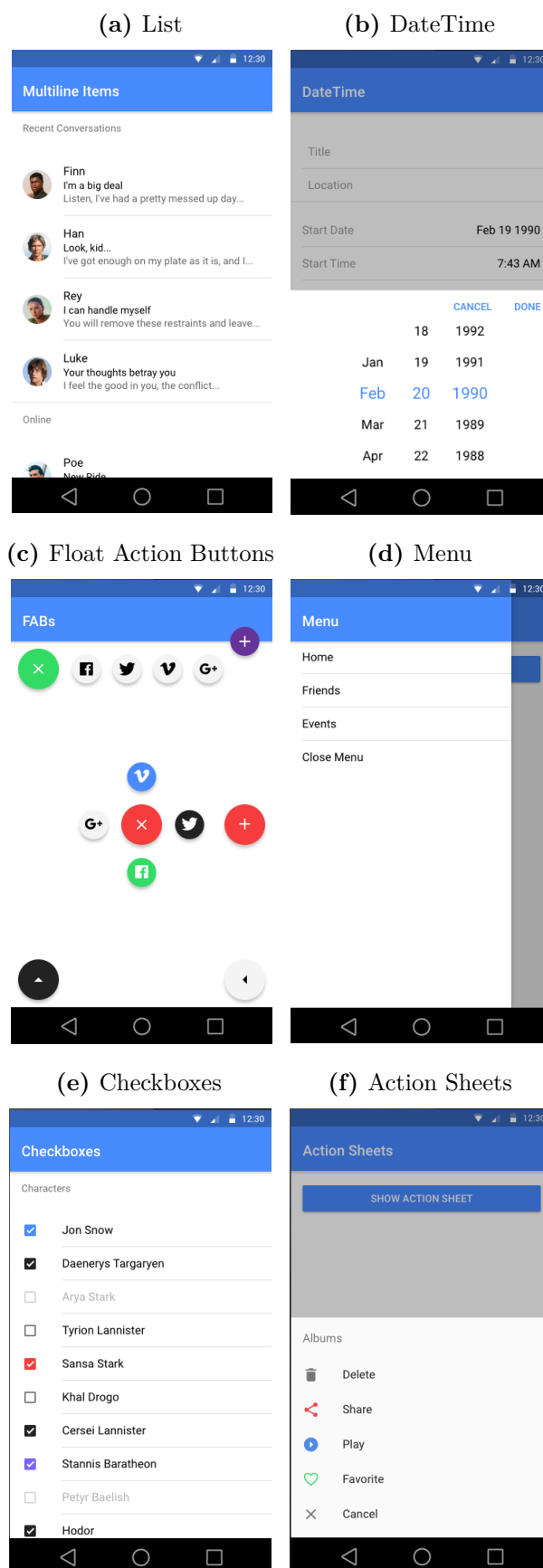
Figura 10 – HelloIonicPage: elementos visuais da página

```
1  <ion-header>
2    <ion-navbar>
3      <button ion-button menuToggle>
4        <ion-icon name='menu'></ion-icon>
5      </button>
6      <ion-title>Hello Ionic</ion-title>
7    </ion-navbar>
8  </ion-header>
9
10 <ion-content padding>
11
12   <h3>Welcome to your first Ionic app!</h3>
13
14   <p>
15     This starter project is our way of helping you get a functional
16       app running in record time.
17   </p>
18   <p>
19     Follow along on the tutorial section of the Ionic docs!
20   </p>
21   <button ion-button color='primary' menuToggle>Toggle Menu</button>
22 </p>
23
24 </ion-content>
```

Fonte – do autor (2017)

Figura 11 – Página em *Ionic* resultado das Figuras 9 e 10

Fonte – do autor (2017)

Figura 12 – Exemplo de componentes no *Ionic*

3.4.5 Estrutura do Framework: *FeathersJS*

FeathersJS é um *framework open-source* de desenvolvimento rápido para aplicações web em tempo-real escritas em JavaScript. Disponibiliza uma arquitetura simples mas poderosa para a construção de aplicações utilizando padrões de programação orientada a aspectos e serviços (FEATHERS, 2017). Os principais componentes do *FeathersJS* são os *services*, *hooks* e *events* que são detalhados nas próximas seções.

3.4.5.1 Services

Services ou serviços são a camada principal do *FeathersJS*. Um serviço é simplesmente uma instância de uma classe JavaScript que implementa métodos básicos para criação, consulta, atualização e destruição de dados. Esse conjunto de operações ou funcionalidades é conhecido como CRUD (*Create, Read, Update, Delete*).

Os serviços no *FeathersJS* expõem uma interface uniforme de acesso, permitindo assim fornecer uma única API tanto para chamadas HTTP REST e *websockets*. Os verbos HTTP (GET, POST, PUT, PATCH e DELETE) têm a correspondência com os métodos de um serviço no *FeathersJS* listados na Figura 13.

Figura 13 – Interface de um serviço

```
1 const meuServico = {
2   // GET /path
3   find(params, callback) {},
4   // GET /path/<id>
5   get(id, params, callback) {},
6   // POST /path
7   create(data, params, callback) {},
8   // PUT /path/<id>
9   update(id, data, params, callback) {},
10  // PATCH /path/<id>
11  patch(id, data, params, callback) {},
12  // DELETE /path/<id>
13  remove(id, params, callback) {}
14 }
```

Fonte – do autor (2017)

3.4.5.2 Hooks

Hooks são tecnicamente *middleware* ou funções que têm acesso aos objetos de solicitação (**req**) e resposta (**res**). Dessa forma os *hooks* podem fazer mudanças nos objetos de solicitação e resposta. O *FeathersJS* permite registrar *hooks* antes (*before*), depois (*after*) ou em caso de erro (*error*) dos métodos de um serviço, como mostrado na Figura 13.

Como eles têm acesso ao objetos de uma requisição (**req** e **res**), eles são usados para política de controle de acesso da aplicação, registro de eventos, enviar notificações, adicionar propriedades e muito mais.

Essa abordagem é conhecida como Programação Orientada a Aspectos, que permite a separação de propriedades ortogonais (ou que não fazem parte da funcionalidade principal) dos componentes funcionais de uma forma natural e concisa.

Na [Figura 14](#) três *hooks* foram registrados para um serviço de questões (*questions*) (ℓ. 1), em que é adicionado a propriedade **createdAt** antes (*before*) da criação (*create*) de um objeto questão (ℓ. 3 – 5), e a propriedade **updatedAt** quando uma questão é modificada (*update* e *patch*), (ℓ. 7 – 13).

Figura 14 – Exemplo registro de *hooks* no *FeathersJS*

```
1  app.service('questions').hooks({
2    before: {
3      create(hook) {
4        hook.data.createdAt = new Date();
5      },
6
7      update(hook) {
8        hook.data.updatedAt = new Date();
9      },
10
11     patch(hook) {
12       hook.data.updatedAt = new Date();
13     }
14   }
15 });
```

Fonte – do autor (2017)

3.4.5.3 Events

São os *events* ou eventos no *FeathersJS* permitem a criação de aplicações de tempo-real usando *WebSockets*.

No *FeathersJS*, os serviços enviam automaticamente eventos ou notificações **created**, **updated**, **patched**, **removed** quando algum dos respectivos métodos listados na [Figura 13](#) finalizam com sucesso. Os clientes da aplicação podem então ouvir a esses eventos e reagirem de acordo.

Na [Figura 15](#) o cliente obtém uma referência do serviço de votação (ℓ. 2) e então passa a ouvir quando uma nova votação é criada (**created**) (ℓ. 5 – 7). Nesse caso, os clientes de uma aplicação de votação, por exemplo, poderiam receber as questões da votação publicada por outro cliente e então responder.

Figura 15 – Exemplo eventos no *FeathersJS*

```

1 // Retrieve the wrapped service object which will be an event emitter
2 const poll = app.service('poll');
3
4 // Listen 'created' event
5 poll.on('created', (poll) => {
6   console.log('New poll created', poll);
7 });

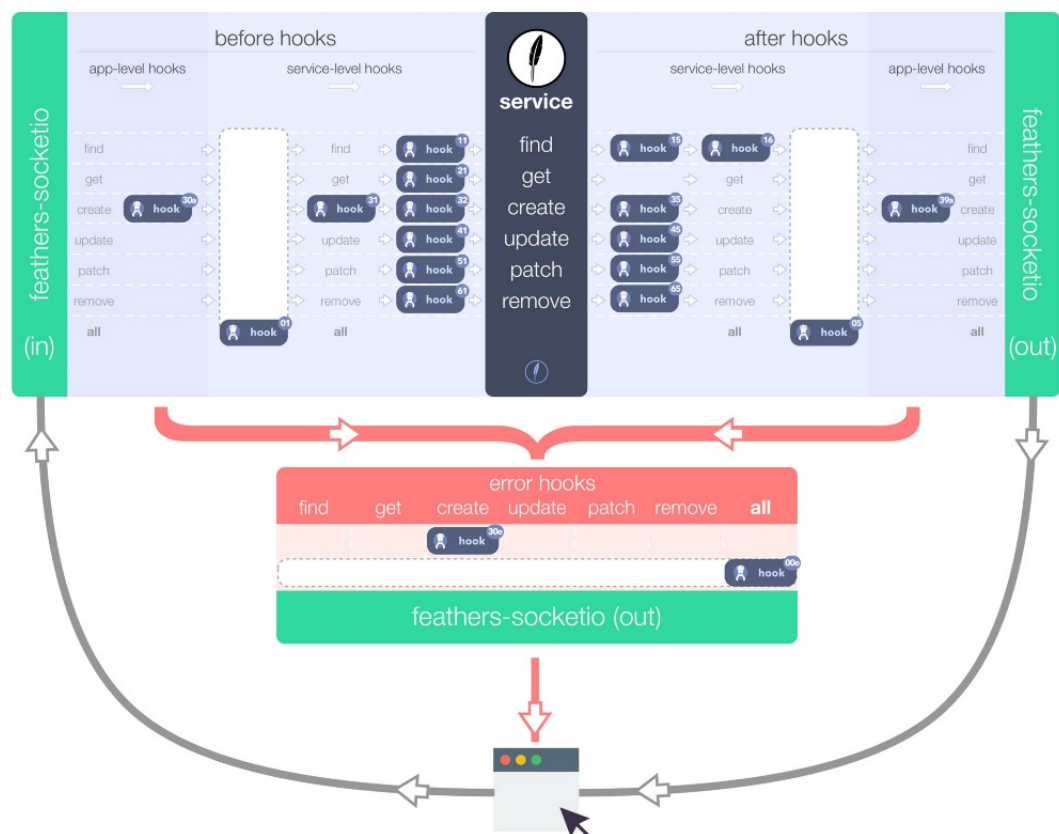
```

Fonte – do autor (2017)

3.4.5.4 Visão geral

A Figura 16 ilustra como funciona o ciclo de uma requisição entre clientes e uma aplicação baseada no *FeathersJS*.

O cliente faz uma requisição para um serviço, que antes de chegar no serviço passa pela camada de *before hooks*, o método requisitado pode completar com sucesso indo para a *after hooks* e enviando um evento para os clientes conectados. Qualquer erro no processo é enviado para a camada *error hooks* que também pode notificar os clientes da aplicação.

Figura 16 – Como o ciclo de uma requisição funciona no *FeathersJS*

Fonte – Disponível em: docs.feathersjs.com/guides/step-by-step/basic-feathers/all-about-hooks.html. Acesso em 29 Mai. 2017

A utilização do *FeathersJS* simplificou muito o processo de construção do software por disponibilizar facilidades para o programador e implementar uma API de tempo-real via serviços de forma nativa. Além disso, a interface dos serviços [Figura 13](#) torna fácil a integração com qualquer banco de dados. Nesse sentido, o *FeathersJS* suporta alguns ORM que permitem uma integração com uma variedade de banco de dados por meio de uma interface única.

3.4.6 Implementação

Os itens da lista de tarefas da [Figura 7](#) foram desenvolvidos de forma incremental. Para cada história de usuário desenvolvida, procurava-se desenvolver os critérios de aceitação da mesma. Os critérios de aceitação geralmente definem o comportamento esperado da funcionalidade desenvolvida pela história. Dessa forma, eles são úteis para definir quando uma história foi finalizada e implementada de forma correta.

Os critérios de aceitação foram escritos usando o formato utilizado no desenvolvimento dirigido por comportamento (*Behavior-Driven Development (BDD)*). No BDD descreve-se o comportamento no seguinte formato:

Dado que (*Given*): determinadas pré-condições são atendidas;

Quando (*When*): um determinado evento ocorre;

Então (*Then*): isso deve acontecer.

Considere por exemplo, a história de usuário:

Como professor

Gostaria ser capaz de criar (questões de múltipla escolha | verdadeiro e falso | questões abertas)

para que eu tenha variedade de perguntas para explorar.

Tem-se os seguintes critérios de aceitação:

1. **Dado que** o professor pode criar uma questão
 - Quando** ele escolher criar uma questão do tipo múltipla escolha
 - E** preencher o campo *questão*
 - E** adicionar pelo menos duas alternativas
 - E** escolher uma alternativa como a correta
 - E** clicar no botão *CONCLUÍDO*

Então o sistema deve permitir salvar a questão

E indicar que a questão foi salva com sucesso

E eu devo ser redirecionado para a aba *Questões*.

2. **Dado que** o professor pode criar uma questão

Quando ele escolher criar uma questão do tipo múltipla escolha

E preencher o campo *questão*

E não adicionar pelo menos duas alternativas

E não escolher uma alternativa como a correta

Então o sistema não deve permitir salvar a questão

E indicar que é necessário adicionar pelo menos duas alternativas

E indicar que é necessário escolher uma alternativa como a correta.

3.4.6.1 Banco de Dados: MongoDB

O banco de dados utilizado foi o MongoDB que é um banco de dados NoSQL orientado a documentos. Em tais bancos os dados são *semiestruturados*. Dados semiestruturados são dados em que o esquema de representação está presente em conjunto com o dado, ou seja, eles são *auto-descritivos*. As nomenclaturas do MongoDB diferem dos bancos relacionais. A [Tabela 2](#) apresenta como eles se relacionam.

Tabela 2 – Nomenclaturas dos banco de dados relacionais *versus* MongoDB

Banco Relacional	MongoDB
Base de dados	Base de dados
Tabela	Coleção
Registro	Documento
Coluna	Campo
Índice	Índice
Join	Documento Embarcado
Chave estrangeira	Referência

Fonte – do autor (2017)

O MongoDB armazena os dados no formato chamado BSON (Binary JSON). O BSON estende o JSON (JavaScript Object Notation) incluindo suporte para tipos de dados `int`, `long`, `date`, `floating point` e `decimal128`. Um documento BSON contém um ou mais campos, e cada campo um valor de um tipo específico, incluindo vetores, dados binários, e outros sub-documentos ([MONGODB, 2017](#)). A [Figura 17](#) apresenta um exemplo de um documento em MongoDB.

Documentos (ou *documents*) que tendem a compartilhar a mesma estrutura são organizados como *coleções* (ou *collections*). Coleções são análogas a uma tabela em um

banco de dados relacional, documentos são similares a registros ou linhas, e campos são parecidos com as colunas.

Figura 17 – Exemplo de um documento em MongoDB

```
1  {
2    "_id" : ObjectId("59526b5801f55103c054779c"),
3    "name" : "Engenharia de Software",
4    "code" : "ENGSOFT123",
5    "user" : ObjectId("59526b2001f55103c054779b"),
6    "updatedAt" : ISODate("2017-06-27T14:27:36.113Z"),
7    "createdAt" : ISODate("2017-06-27T14:27:36.113Z"),
8    "peopleOnline" : -2,
9    "private" : true,
10   "online" : true,
11   "students" : [
12     {
13       "_id" : ObjectId("59526b7601f55103c054779e"),
14       "online" : false,
15       "id" : "102",
16       "name" : "Paulo"
17     },
18     {
19       "_id" : ObjectId("59526b7601f55103c054779d"),
20       "online" : false,
21       "id" : "101",
22       "name" : "Pedro"
23     }
24   ],
25   "--v" : 0
26 }
```

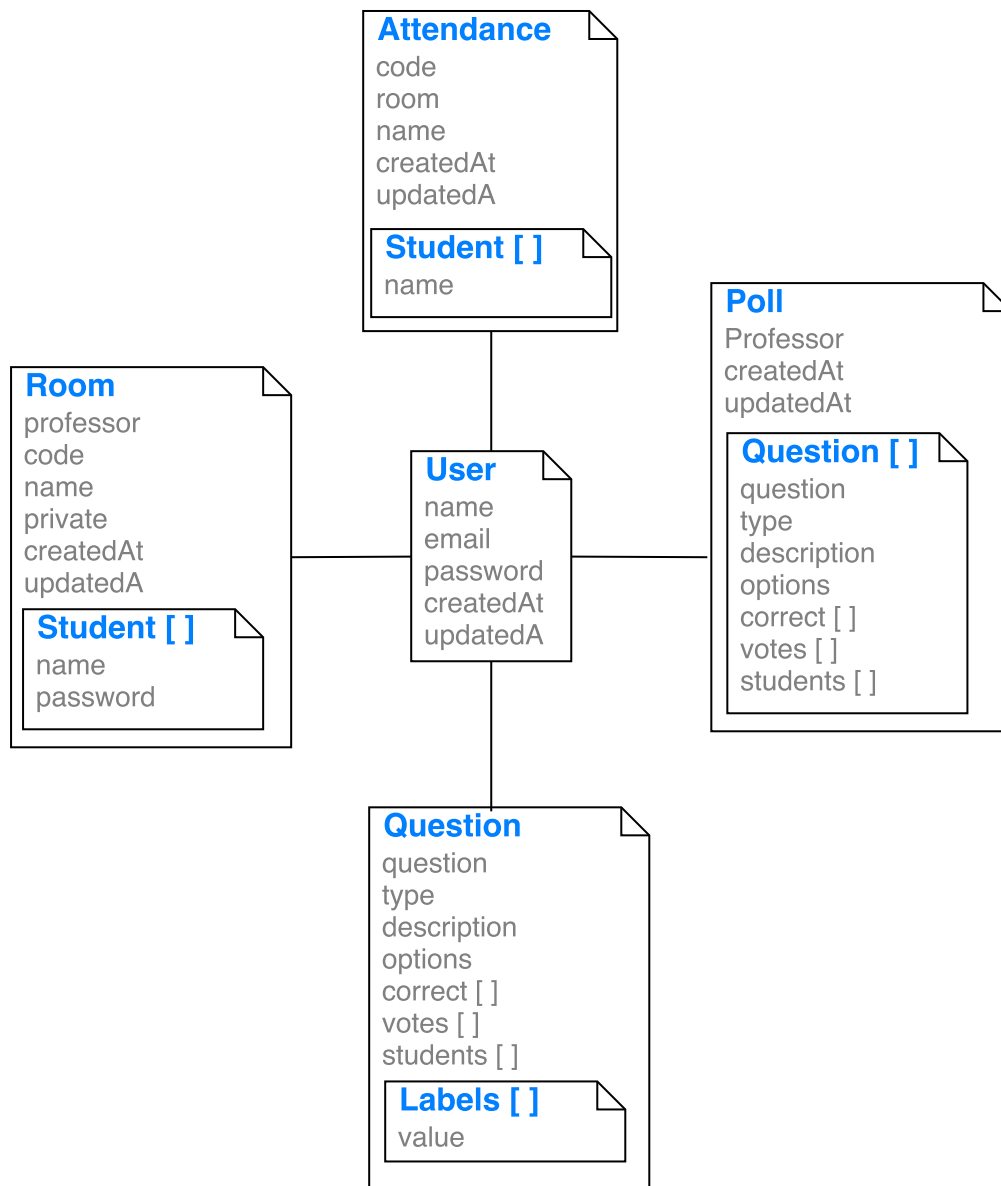
Fonte – do autor (2017)

A [Figura 18](#) exibe uma versão simplificada de como os dados foram estruturados. O sistema tem basicamente quatro coleções: *User* (*Usuário*), *Poll* (*votação*), *Room* (*Sala*), *Attendance* (*Frequência*), *Question* (*Questão*). Nesse esquema, as questões (*question*) são sub-documentos dentro de uma votação (*poll*), dessa forma, com apenas uma leitura no banco de dados é possível obter toda a informação de uma votação.

3.4.7 Servidor: FeathersJS

O principal componente de uma aplicação desenvolvida com o *FeathersJS* são os serviços. Os serviços foram explicados em detalhes na [subseção 3.4.5.1](#).

Para cada coleção do banco de dados mostrado na [Figura 18](#) foi desenvolvido um serviço correspondente no *FeathersJS*. Para auxiliar nesse processo, o *FeathersJS* conta com uma ferramenta para tornar ainda mais rápido o processo de desenvolvimento de aplicações. Um deles é o utilitário de linha de comando **feathers** capaz de gerar os principais componentes do *framework* como os *services* e *hooks*.

Figura 18 – Diagrama do banco de dados orientado a documentos

Fonte – do autor (2017)

A [Figura 19](#) ilustra a utilização da ferramenta para a geração do serviço *question* do tipo *MongoDB*. Nesse exemplo, como o serviço é do tipo *MongoDB*, ao final do processo tem-se uma REST API totalmente funcional, por não ser preciso definir um modelo do serviço. Se um banco de dados relacional fosse usado como o tipo do serviço, teria-se que desenvolver o modelo do esquema da tabela primeiro.

3.4.8 Frontend: Ionic

Uma das características interessantes do FeathersJS é que ele também funciona como cliente no navegador exatamente da mesma forma como no servidor. A [Figura 20](#), mostra um serviço que faz parte da aplicação web desenvolvida em Ionic, que é responsável

Figura 19 – Interface de linha de comando da ferramenta feathers

```

→ rsa-backend git:(master) X feathers generate service
? What kind of service is it? MongoDB
? What is the name of the service? question
? Which path should the service be registered on? /question
? Does the service require authentication? No
create src/services/question/question.service.js
force src/services/index.js
create src/services/question/question.hooks.js
create src/services/question/question.filters.js
create test/services/question.test.js
force src/mongodb.js

```

Fonte – do autor (2017)

por obter as informações nesse caso específico *Poll* (votação).

Na (ℓ. 2) importa-se a classe responsável por fazer a conexão com o servidor e que também expõem alguns métodos como por exemplo para obter uma referência de um serviço do servidor (ℓ. 6). Na (ℓ. 11 – 13) tem-se o método para criar uma votação. Na (ℓ. 16 – 23), o método `poll(room)` retorna uma votação que tenha um código específico de uma sala (ℓ. 19) e que não ainda não tenha sido encerrada (ℓ. 20).

Figura 20 – *PollService* - serviço para prover informações do serviço de votações

```

1  import { Injectable } from '@angular/core';
2  import { FeathersProvider } from '../feathers';
3
4  @Injectable()
5  export class PollService {
6      polls = this.app.service('polls');
7
8      constructor(public app: FeathersProvider) { }
9
10     // create a poll
11     create(poll) {
12         return this.polls.create(poll);
13     }
14
15     // get poll info
16     poll(room: any) {
17         return this.polls.find({
18             query: {
19                 'room.code': room.code,
20                 isOver: false,
21             }
22         });
23     }
24 }

```

Fonte – do autor (2017)

Agora que o acesso ao serviço foi definido, desenvolve-se a página que vai fazer uso do serviço criado. No exemplo da Figura 21, tem-se a página de votação no aplicativo

do aluno, representada pelo arquivo *poll.ts*. Inicia-se importando os serviços usado na página (ℓ. 1 – 6), como por exemplo o *PollService* apresentado anteriormente na Figura 20. Na (ℓ. 20), exibe o uso do método *poll* que obtém uma votação disponível de uma turma. Como última parte da página, falta definir os componentes de visualização. No exemplo da Figura 22 segue a definição visual da página de votação do aplicativo do estudante representada pelo arquivo *poll.html*. Esse arquivo é responsável por representar a interface gráfica e interagir com os estudantes. Nessa página, quando não existe uma questão disponível para votação, ela exibe as informações do estudante, da sala e do professor (ℓ. 10 – 23). Quando o professor habilita uma questão para votação, a página passa a exibir a questão, as alternativas e o botão para responder (ℓ. 25 – 47).

Figura 21 – Parte de página de votação da aplicação aluno - *poll.ts*

```

1      import {
2          AttendanceProvider,
3          FeathersProvider,
4          PollService,
5          RoomsProvider,
6      } from '../providers/providers';
7
8      @IonicPage({
9          segment: 'poll/:id',
10         defaultHistory: ['HomePage']
11     })
12     @Component({
13         selector: 'page-poll',
14         templateUrl: 'poll.html',
15     })
16     export class PollPage {
17         constructor(public pollService: PollService) { }
18
19         initPoll() {
20             this.pollService.poll({ code: this.code })
21                 .subscribe((poll) => {
22                 if (poll.data.length > 0 && poll.data[0].available !== -1) {
23                     this.poll = poll.data[0];
24                     this.question = this.poll.questions[this.poll.available];
25
26                     if (this.checkAlreadyAnswered(this.question)) {
27                         this.poll = null;
28                     }
29                 } else {
30                     this.poll = null;
31                 }
32             });
33     }
34 }
```

Fonte – do autor (2017)

Figura 22 – Parte de página de votação da aplicação aluno - *poll.html*

```

1      <ion-header>
2
3      <ion-navbar>
4          <ion-title>{{room?.name}} #{{code}}</ion-title>
5      </ion-navbar>
6
7  </ion-header>
8
9  <ion-content padding>
10     <div *ngIf='!poll'>
11
12         <div class='welcome'>
13             <p>
14                 <span *ngIf='!student?.name'>Bem</span>
15                 <span *ngIf='student?.name'>
16                     <span ion-text color='primary'>{{student?.name}}</span>, bem
17                 </div>
18             </div>
19
20         <div *ngIf='poll'>
21             <div class='question'>
22                 <span>{{question?.question}}</span>
23             </div>
24
25             <div class='description'>
26                 <span>{{question?.description}}</span>
27             </div>
28
29             <ion-list *ngIf 'question?.type !== 'free ' radio-group [(
30                 ngModel)]'answer '>
31                 <ion-item *ngFor='let option of question.options; let i=index
32                     '>
33                     <ion-label>{{option.text}}</ion-label>
34                     <ion-radio [value]'option '></ion-radio>
35                 </ion-item>
36             </ion-list>
37
38             <ion-item *ngIf 'question?.type=== 'free ' >
39                 <ion-label stacked>Resposta</ion-label>
40                 <ion-textarea placeholder 'Digite aqui a sua resposta ' [(
41                     ngModel)] 'answer '></ion-textarea>
42             </ion-item>
43
44             <button (click) 'submit(answer) ' ion-button color 'primary '
45                 full>ENVIAR</button>
46         </div>
47     </ion-content>

```

Fonte – do autor (2017)

3.5 Verificação do Software

3.5.1 Testes Automatizados

Para garantir a qualidade do software desenvolvido, desenvolveu-se testes automatizados para avaliar se o software estava sendo desenvolvido como esperado. A base para desenvolver os testes automatizados foram os critérios de aceitação de cada história de usuário. Alguns exemplos de critérios de aceitação foram descritos na [subseção 3.4.6](#).

As ferramentas utilizadas nesse processo foram os *frameworks* de teste *Jasmine* e o *Protractor* que são descritas nas seções seguintes.

3.5.1.1 Jasmine

Jasmine é um *framework open-source* BDD para testar código JavaScript, que não depende de nenhum outro *framework* (LABS, 2017).

Um conjunto de testes ou suíte de testes no Jasmine começa com a chamada da função global **describe** com dois parâmetros: uma *string* e uma função. A *string* é o nome ou título para um conjunto de *especificações* e geralmente referem-se ao que está sendo testado.

Uma *especificação* ou *spec* contém uma ou mais *expectativas* que verificam o estado do código. Uma especificação é definida com a chamada da função global *it*, com dois parâmetros: uma *string* (título da especificação) e uma função (código do teste). Uma *expectativa* é uma asserção que tem geralmente como valor verdadeiro ou falso.

O exemplo da [Figura 23](#) exhibe uma suíte de testes no Jasmine. A suíte de testes começa com a chamada da função **describe** (ℓ. 1), e uma especificação. A especificação da (ℓ. 7 – 9) espera que seja a chamada da função **soma** definida anteriormente com os parâmetros 2 e 4 seja igual a 6 (ℓ. 8).

Figura 23 – Exemplo de uma suíte de testes no Jasmine

```
1 describe("Suite de testes", function () {  
2  
3     function soma(a, b) {  
4         return a + b;  
5     }  
6  
7     it("outra especulacao", function () {  
8         expect(soma(2, 4)).toEqual(6);  
9     });  
10 });
```

Fonte – do autor (2017)

3.5.1.2 Protractor

O Protractor é um *framework open-source*, desenvolvido pela Google, baseado em Node.js, para a criação de testes *e2e* (*end to end*) principalmente para aplicações desenvolvidas em AngularJS (GOOGLE, 2017). O Protractor permite simular usuários interagindo com a aplicação, em navegadores reais, como o Chrome e o Firefox. Para isso, ele disponibiliza algumas classes que permitem interagir com os elementos da página.

O exemplo da suíte de testes da Figura 24 desenvolvido em Jasmine utilizando o Protractor ilustra algumas dessas classes que permitem interagir com a aplicação. Por exemplo, na (ℓ.3) a classe **browser** permite acessar a URL especificada. Assim que a página carregar, simula-se uma pesquisa. Na (ℓ.5) é digitado no campo localizado por CSS a palavra *inicio*. Continuando, é localizado o botão para fazer a pesquisa (também localizado por CSS) e então a ação de clique é executada (ℓ.6). Por fim, é feita a verificação que o elemento localizado contém o texto especificado (ℓ.8).

Figura 24 – Exemplo de uma suíte de testes no Jasmine com o Protractor

```
1 describe('Homepage', () => {  
2   it('deve fazer uma pesquisa na pagina de api', () => {  
3     browser.get('http://localhost:8080/#/api');  
4  
5     element(by.css('.searchTerm')).sendKeys('inicio');  
6     element(by.css('.searchButton')).click();  
7  
8     expect(element(by.css('.api-title')).getText()).toContain('browser .  
        inicio');  
9   });  
10 });
```

Fonte – do autor (2017)

3.5.1.3 Teste de aceitação: criar questão

A suíte de testes do exemplo da Figura 25 foi desenvolvido seguindo os critérios de aceitação da seguinte história de usuário:

Como professor

Gostaria ser capaz de criar (questões de múltipla escolha | verdadeiro e falso | questões abertas)

para que eu tenha variedade de perguntas para explorar.

O seguinte exemplo descreve um critério de aceitação:

Dado que o professor pode criar uma questão

Quando ele escolher criar uma questão do tipo múltipla escolha

E preencher o campo *questão*

E adicionar pelo menos duas alternativas

E escolher uma alternativa como a correta

E clicar no botão *CONCLUÍDO*

Então o sistema deve permitir salvar a questão

E indicar que a questão foi salva com sucesso

E eu devo ser redirecionado para a aba *Questões*.

Alguns arquivos de configuração do exemplo da [Figura 25](#) foram omitidos para simplificação do exemplo. Antes de chegar nesse teste, um usuário é autenticado no sistema, satisfazendo a condição “*como professor*”, em seguida, para atender que “*dado que o professor pode criar uma questão*”, o sistema deve permitir que o professor consiga abrir o formulário para adicionar a questão (ℓ. 8 – 20), na (ℓ. 18) o é verificado se o formulário foi aberto. Continuando, na (ℓ. 22 – 40), o sistema deve permitir inserir uma questão de verdadeiro e falso, quando o professor preenche todos os requisitos. Na (ℓ. 38) é esperado que o número de questões seja igual ao que tinha antes mais um, já que uma questão foi adicionada. Avançando no teste, outro formulário para adicionar questão é aberto para verificar que o sistema não deve permitir adicionar uma questão de múltipla escolha sem que o professor escolha uma alternativa como correta (ℓ. 42 – 58).

A [Figura 26](#) exibe o relatório da execução dos teste pelo Jasmine. Outros testes que foram desenvolvidos envolveu a utilização dois navegadores abertos ao mesmo tempo, um simulando o professor disponibilizando questões, e no outro navegado, um aluno respondendo a questão disponibilizada, dessa forma, foi possível testar o software de uma maneira abrangente ao envolver todos os usuários do sistema.

Figura 25 – Exemplo de um teste de aceitação automatizado

```
1 describe('Questions Tab', () => {
2   let questionsTab: QuestionsTab;
3
4   beforeEach(() => {
5     questionsTab = new QuestionsTab();
6   });
7
8   describe('add new questions', () => {
9
10    it('should open new question form', () => {
11      questionsTab.navigateToPage();
12      browser.refresh().then(() => {
13        questionsTab.sleep(1000);
14
15        questionsTab.getAddQuestionButton().click();
16        questionsTab.sleep();
17
18        expect(questionsTab.getModalAddQuestion().isDisplayed()).
19          toBeTruthy();
20      });
21    });
22
23    it('should add true or false question', () => {
24      let form = new NewQuestionPage();
25      form.setQuestionInput('To be or not to be');
26
27      form.setTrueOrFalseQuestionType();
28
29      browser.driver.sleep(500);
30
31      form.selectCorrectAlternative();
32
33      form.save().click();
34
35      browser.driver.sleep(2000);
36
37      questionsTab.getNumQuestions()
38        .then(questions => {
39          expect(questions).toEqual(questionsTab.questions + 1);
40        });
41    });
42
43    it('should not allow mc question without correct option', () => {
44      questionsTab.getAddQuestionButton().click();
45      questionsTab.sleep();
46
47      const form = new NewQuestionPage();
48      form.setQuestionInput('To be or not to be');
49
50      form.addAlternative();
51      form.addAlternative();
52
53      form.setAlternativeText(0, 'TO BE');
54      form.setAlternativeText(1, 'NOT TO BE');
55
56      expect(form.save().isEnabled()).toBeFalsy();
57
58      form.cancel();
59    });
60  });
```

Figura 26 – Relatório de execução dos testes no Jasmine

```
Jasmine started

Questions Tab

  add new questions
    ✓ should open new question form
    ✓ should add true or false question
    ✓ should not allow mc question without correct option
    ✓ should not allow mc question with one option

  Poll Page
    ✓ should change tab when starts a new session
[15:00:26] W/element - more than one element found for locator By(css s
    ✓ student should be able to answer

  Home Page

    Student Login
      ✓ join button should be disabled
      ✓ should go to votes page with title that contains #ENGSOFT123
      ✓ should come back to home page when student cancels
      ✓ should come back to home page with inexisted room code

Executed 10 of 10 specs SUCCESS in 42 secs.
[15:00:41] I/launcher - 0 instance(s) of WebDriver still running
[15:00:41] I/launcher - chrome #01 passed
→ rsa-app git:(master) █
```

Fonte – do autor (2017)

4 Resposta em Sala de Aula

As principais tecnologias, ferramentas e métodos foram descritos nos capítulos anteriores. Como resultado final foi desenvolvido um sistema de resposta em sala de aula denominado *Resposta em Sala de Aula* (RSA) [Figura 27](#).

Figura 27 – Resposta em Sala de Aula (RSA)



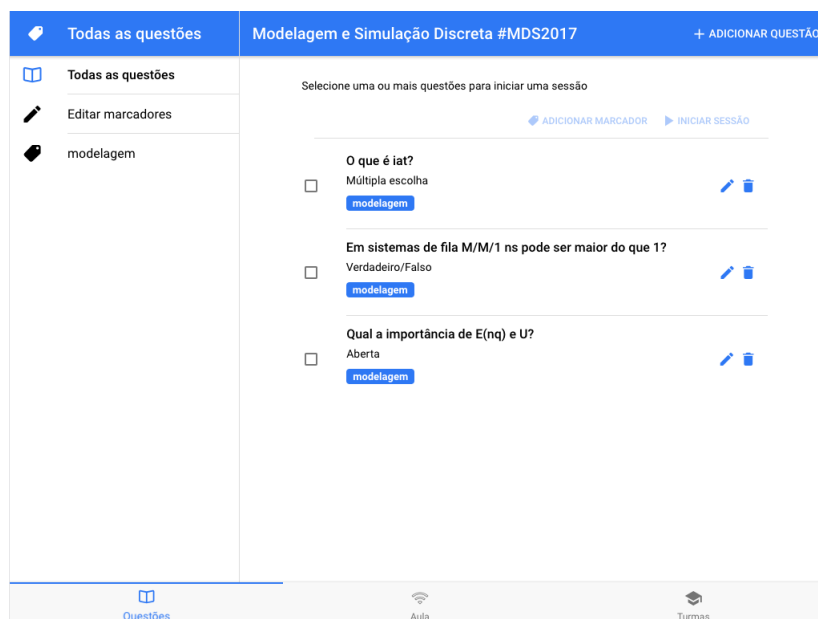
Fonte – do autor (2017)

4.1 Aplicação web para o Professor

A [Figura 28](#) ilustra a tela inicial da aplicação web para o professor. Nela é possível perceber as três abas principais do sistema *Questões*, *Aula* e *Turmas*. A primeira *Questões* que está selecionada, permite ao professor gerir um banco de questões criadas, iniciar uma sessão com questões selecionadas e categorizar as questões para uma melhor organização das mesmas.

4.1.1 Frequência dos alunos

Um dos requisitos do sistema era permitir ao professor realizar a frequência dos alunos pelo aplicativo. O sistema gera um código aleatório de quatro dígitos e permite aos estudantes submeterem esse código pelo aplicativo. O professor pode ativar a frequência clicando no botão *Frequência* no canto superior direito da aba *Questões* ([Figura 28](#)). Em seguida, o sistema muda para a aba *Aula*, tendo como resultado a [Figura 29](#). Nela é

Figura 28 – Aba *Questões*

Fonte – do autor (2017)

possível perceber o código de aleatório de quatro dígitos que foi gerado e alguns botões de ação no canto inferior direito, permitindo ao professor encerrar a frequência por exemplo.

Figura 29 – Página para a realizar a frequência dos estudantes



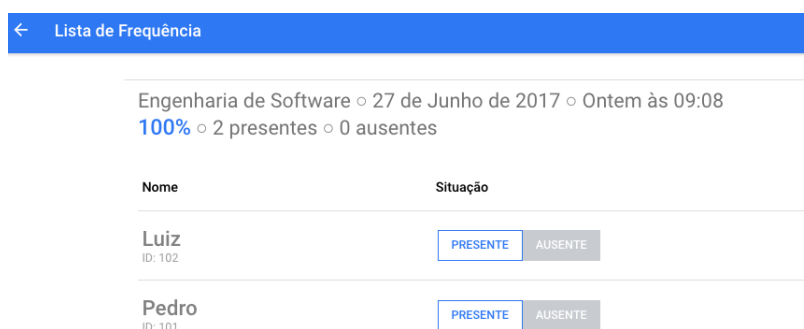
Fonte – do autor (2017)

Posteriormente, o professor pode acessar a lista de frequência por meio do aba *Turmas*, clicando para cada turma a opção de *Frequências*, Figura 30. A Figura 31 exibe a lista de frequência em detalhes. Nela é possível editar a lista de presentes.

Figura 30 – Página listando as frequências realizadas de uma turma


Data	Frequência	Presentes	Ausentes		
27 de Junho de 2017 <small>Ontem às 20:38</small>	0%	0	3		
27 de Junho de 2017 <small>Ontem às 09:08</small>	100%	2	0		
27 de Junho de 2017 <small>Ontem às 08:58</small>	50%	1	1		

Fonte – do autor (2017)

Figura 31 – Detalhes de uma frequência realizada


Nome	Situação
Luiz <small>ID: 102</small>	<input checked="" type="button" value="PRESENTE"/> <input type="button" value="AUSENTE"/>
Pedro <small>ID: 101</small>	<input checked="" type="button" value="PRESENTE"/> <input type="button" value="AUSENTE"/>

Fonte – do autor (2017)

4.1.2 Cadastro de Questões

A [Figura 32](#) exibe o formulário para o cadastro de uma nova questão. A aplicação permite a criação de questões de múltipla escolha, verdadeiro e falso e questão aberta. Em questões do tipo múltipla escolha o professor deve indicar uma alternativa como correta. Também a possibilidade de adicionar a URL de uma imagem, que vai ser visível apenas no aplicativo dos estudantes.

4.1.3 Turmas

Na aba de *Turmas* ([Figura 33](#)), o professor faz o cadastro das turmas para permitir acesso aos estudantes. Foi criado o conceito de *turmas públicas* e *turmas privadas*. Uma turma pública é aquele em que não existe uma lista de alunos cadastrados, ou seja, qualquer aluno com o código de acesso da turma vai poder acessar e participar das atividades apenas digitando o código de acesso.

Quando uma lista de estudantes é adicionada em uma turma (a [Figura 34](#) ilustre esse processo) ela se torna uma turma privada. Quando os estudantes acessarem a turma

Figura 32 – Formulário *Nova Questão*

Nova Questão CANCELAR CONCLUÍDO

- Digite a questão
- Escolha uma opção como correta
- ~~Adicione pelo menos duas alternativas~~

Questão
 Digite a sua questão aqui

Imagem (opcional)
 Cole aqui a URL da imagem

Descrição (opcional)
 Você pode usar este espaço para adicionar detalhes

Múltipla escolha ☐

Verdadeiro/Falso ☒

Aberta ☐

ADICIONAR ALTERNATIVA

Correta	Verdadeiro	<input type="radio"/>	X
Correta	Falso	<input type="radio"/>	X

Fonte – do autor (2017)

eles terão que digitar além do código da turma, o código de identificação de cada um que foi cadastrado no sistema.

Figura 33 – Detalhes da aba *Turmas*

Turmas + ADICIONAR TURMA

Modelagem e Simulação Discreta
 MDS2017 (Acesso privado)

13 ESTUDANTES FREQUÊNCIAS

Fonte – do autor (2017)

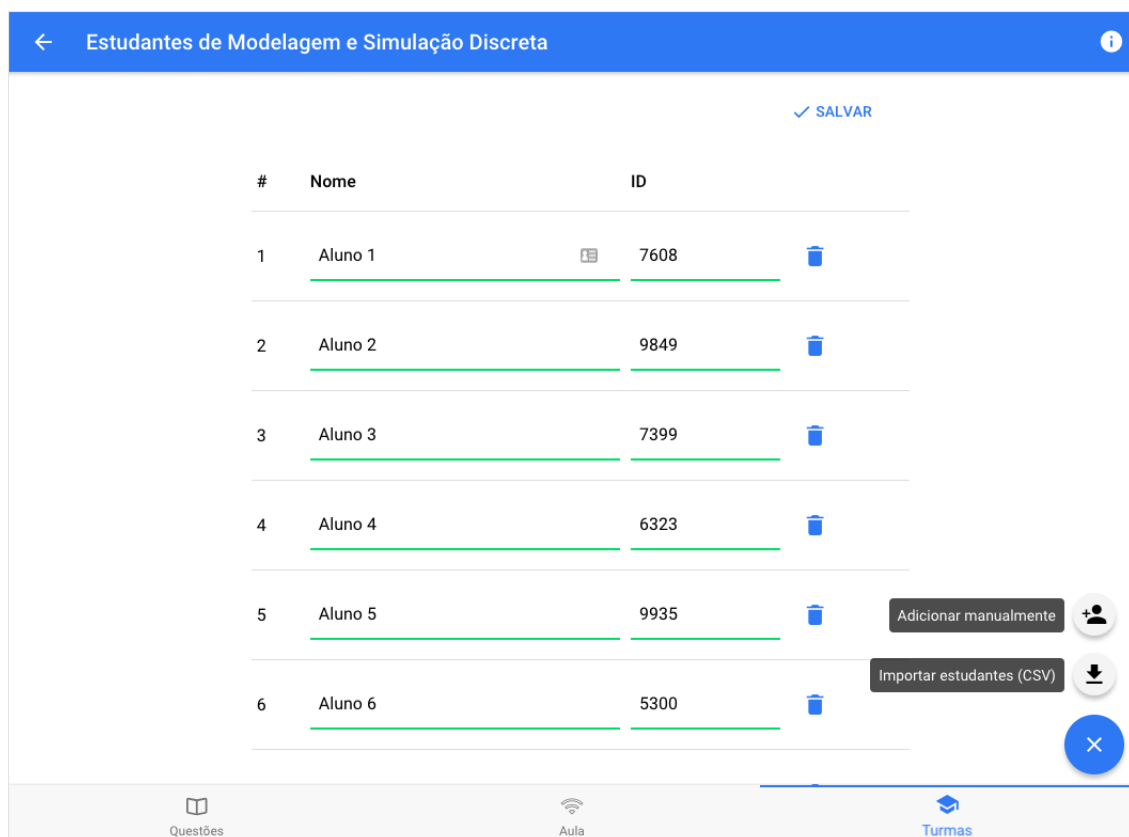
4.1.3.1 Lista de estudantes em uma turma

A Figura 34 ilustra o cadastro de estudantes de uma turma. O professor tem a opção de adicionar manualmente cada estudante, clicando na opção *Adicionar manualmente* no

canto inferior direito, digitando o nome do estudante e o código de identificação (esse é o código que os estudantes terão que digitar além do código da sala para ter acesso).

A outra opção para adicionar a lista de estudantes é importar um arquivo CSV contendo o nome dos estudantes e código de identificação, clicando na opção *Importar Estudantes (CSV)* no canto inferior direito. Um exemplo de um arquivo CSV válido para importação é ilustrado na Figura 35. A importação do arquivo CSV trata a primeira linha como o cabeçalho de coluna, e deve conter exclusivamente `name,id`. Nas outras linhas deve incluir primeiro o nome do estudante e depois o código de identificação.

Figura 34 – Adicionar estudantes em uma turma



Fonte – do autor (2017)

Figura 35 – Exemplo de arquivo CSV válido para importar uma lista de estudantes

```

1  name , id
2  Aluno 1, 4216
3  Aluno 2, 8356
4  Aluno 3, 2435
5  Aluno 4, 8752
6  Aluno 5, 6952

```

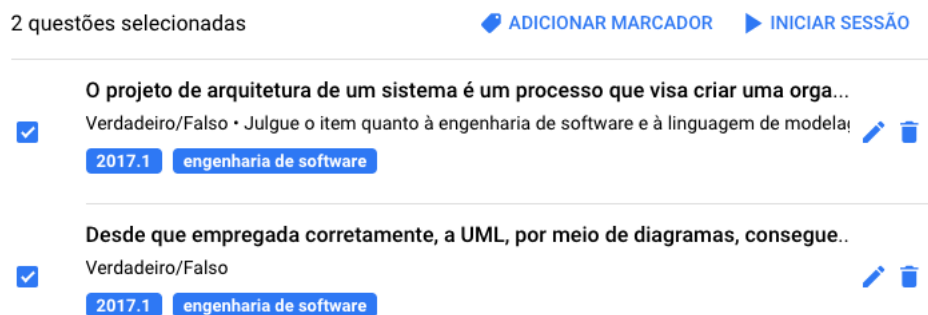
Fonte – do autor (2017)

4.1.4 Sessão de questões: Aba Aula

No sistema tem-se o conceito de *Sessão de questões*, que é simplesmente um conjunto de questões selecionadas pelo professor para apresentar para os estudantes.

O professor pode iniciar uma sessão na aba *Questões*, selecionando uma ou mais questões que deseja apresentar para os estudantes e em seguida clicar no botão *INICIAR SESSÃO*. A [Figura 36](#) destaca da aba *Questões* um exemplo de duas questões selecionadas e o botão *INICIAR SESSÃO* no canto superior direito. Quando o professor inicia uma sessão, o sistema redireciona para a aba *Aula* projetando a primeira questão selecionada, como exibido na [Figura 37](#).

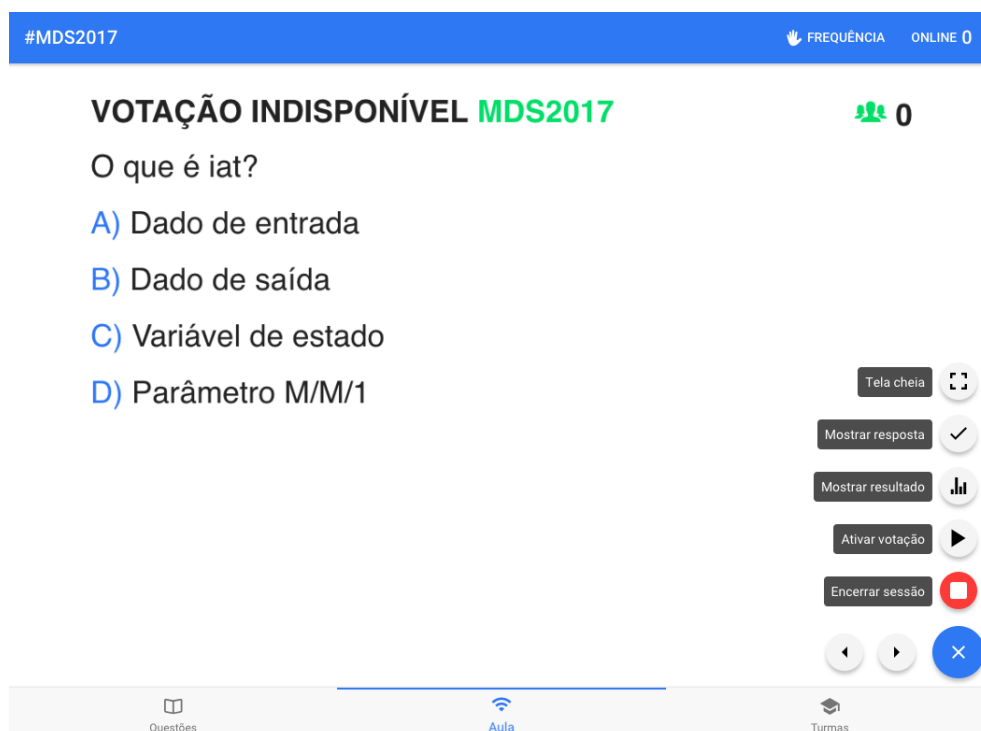
Figura 36 – Aba *Questões*



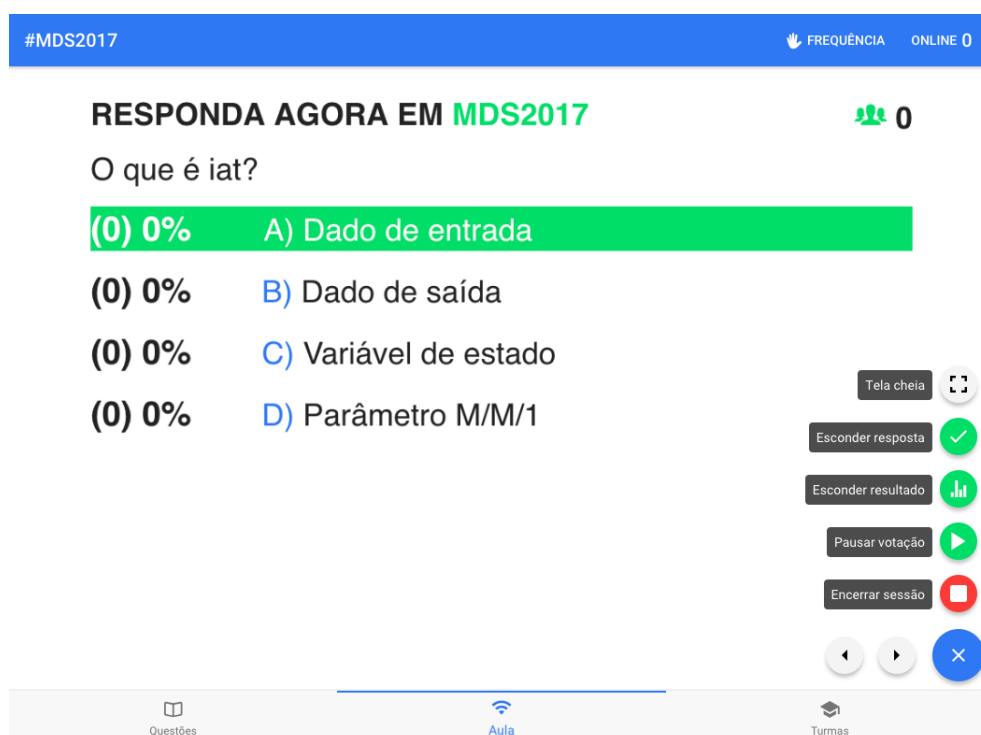
Fonte – do autor (2017)

A [Figura 37](#) ilustra uma sessão iniciada de uma questão. Uma questão quando iniciada ainda não fica disponível para os estudantes responderem, que está indicado na parte superior da tela com o texto *VOTAÇÃO INDISPONÍVEL*. Nessa mesma parte, o código para acesso da sala fica visível para os estudantes, no caso *MDS2017*.

No canto inferior direito da [Figura 37](#), o professor tem acesso a um conjunto de funcionalidades para controlar a sessão. A opção *Ativar votação*, quando selecionada, permite ao professor disponibilizar para os alunos a questão que está sendo exibida para que eles possam responder no aplicativo (a [subseção 4.2.3](#) detalha esse processo). As opções *Mostrar resposta* e *Mostrar resultado* permitem respectivamente indicar na tela a alternativa correta e exibir na tela como os estudantes responderam. No canto superior direito da tela um número é incrementado indicando a quantidade de alunos que já responderam a questão. A [Figura 38](#) ilustra quando todas essas opções estão selecionadas.

Figura 37 – Tela exibindo uma questão ainda indisponível para os estudantes

Fonte – do autor (2017)

Figura 38 – Tela com todas as opções de controle ativadas

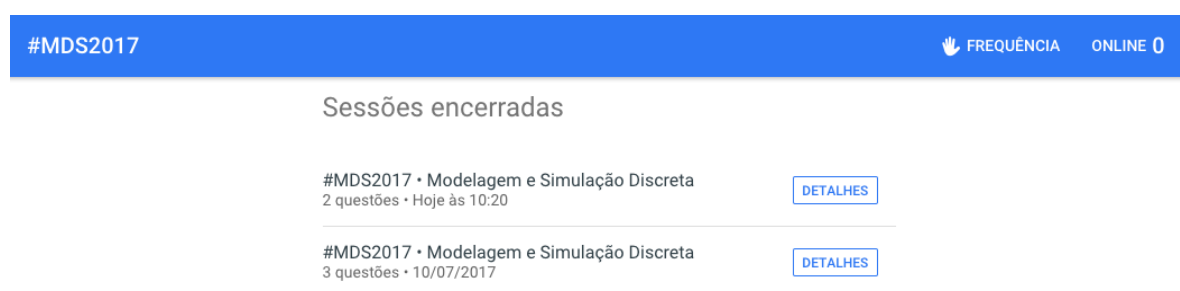
Fonte – do autor (2017)

4.1.4.1 Sessões encerradas

Quando uma sessão é encerrada, clicando no botão *Encerrar sessão* no canto inferior direito exibido na Figura 37, uma lista das sessões encerradas é mostrada, indicando o nome da turma, a quantidade de questões, o número de alunos participantes e a data da sessão, como exibido na Figura 39.

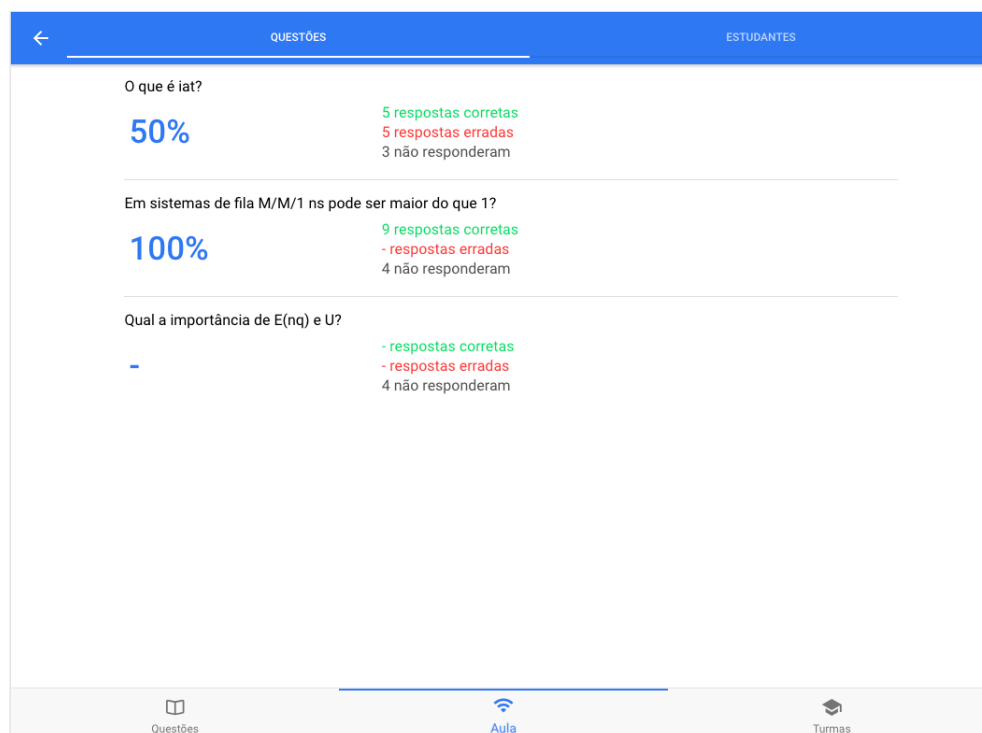
Clicando em um item dessa lista, é possível ver como cada estudante respondeu as questões e um desempenho geral da turma por questão. A Figura 40 exibe o desempenho de cada estudante por questão e a percentagem de acerto na sessão.

Figura 39 – Tela listando as sessões encerradas



Fonte – do autor (2017)

Figura 40 – Detalhes de uma sessão encerrada: resposta por questão e por estudantes

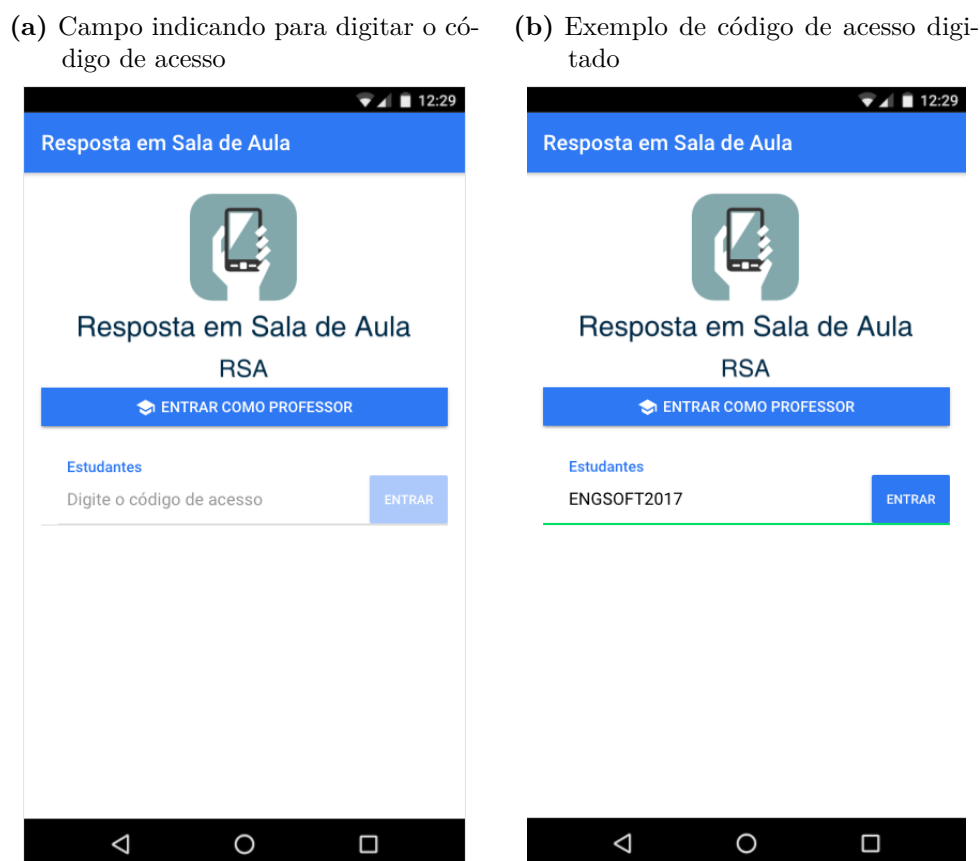


Fonte – do autor (2017)

4.2 Aplicação para dispositivos móveis, que será utilizado como CRSs

A [Figura 41](#) exibe a tela inicial do aplicativo em que é possível perceber um campo para que os estudantes possam digitar o código de acesso da turma, conforme detalhou-se na [subseção 4.1.3](#).

Figura 41 – Tela inicial da aplicação móvel do estudante



Fonte – do autor (2017)

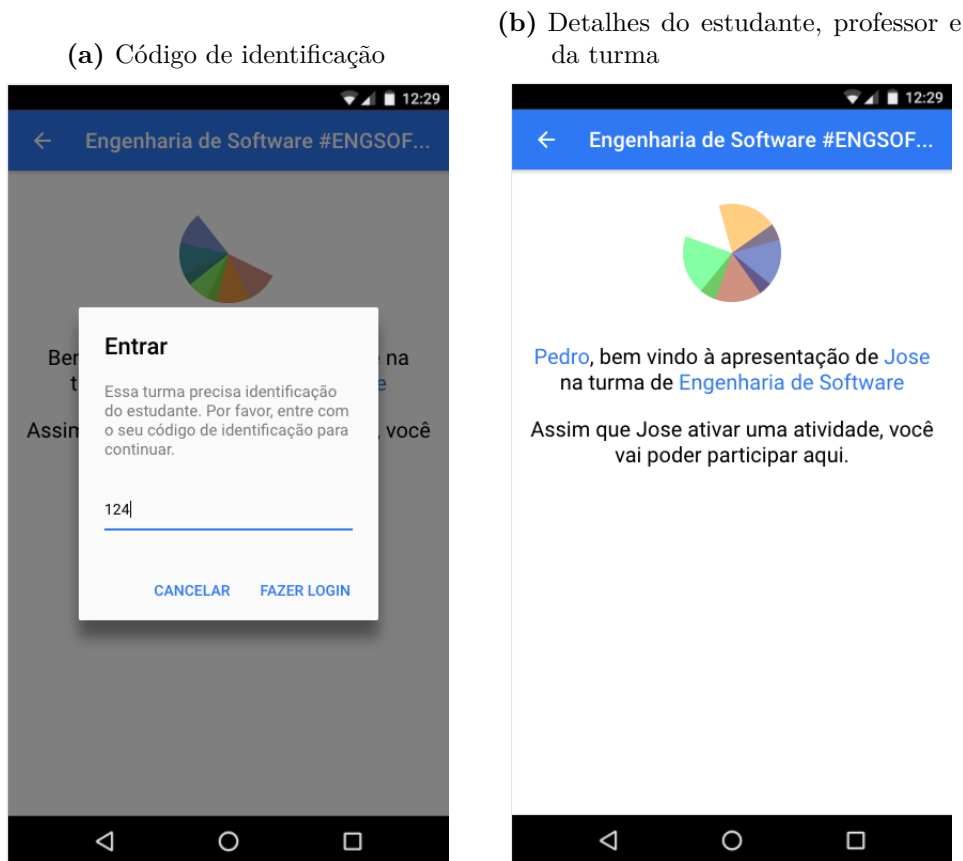
4.2.1 Identificação dos estudantes

Conforme apresentado na [subseção 4.1.3](#) tem-se o conceito de turmas públicas (que permitem acesso a turma apenas com o código da turma) e as turmas privadas que além do código da turma é necessário o código de identificação do estudante cadastrado previamente pelo professor.

A [Figura 42](#) ilustra o processo de identificação do estudante considerando que o acesso a turma é privado. Na [Figura 42a](#) o aplicativo exibe uma mensagem e um campo para que o estudante digite o código de identificação único do estudante. A [Figura 42b](#),

exibe a tela quando o estudante digita o código correto, detalhando o nome do estudante, do professor e da turma.

Figura 42 – Tela de identificação dos estudantes



Fonte – do autor (2017)

4.2.2 Responder frequência

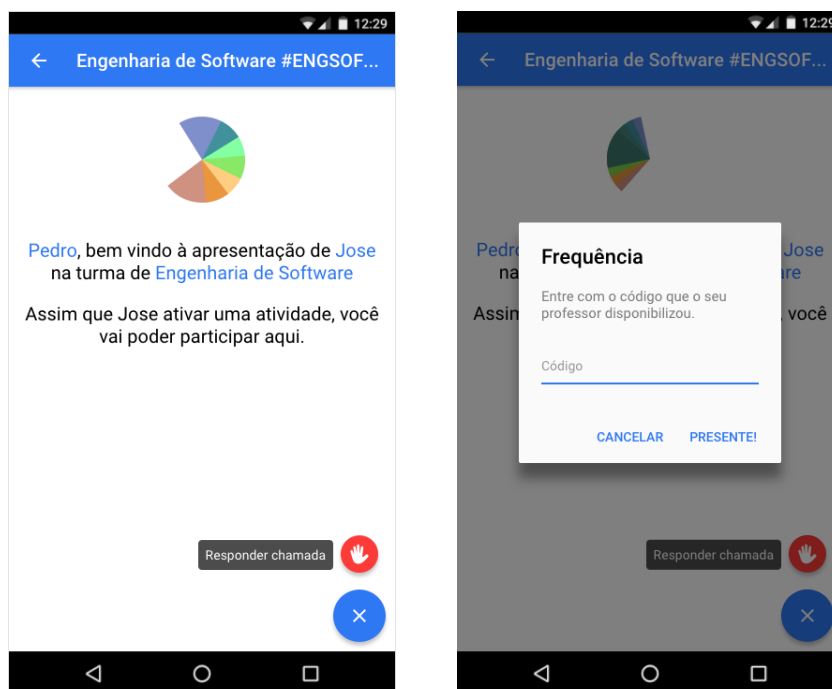
Na [subseção 4.1.1](#) tem-se o processo para o professor realizar a frequência dos estudantes. Quando o professor habilita para os estudantes enviarem o código de quatro dígitos gerado, um botão aparece no aplicativo dos estudantes com a opção *Responder chamada* no canto inferior direito, como mostra na [Figura 43a](#). Em seguida, quando o estudante clica nesse botão, o aplicativo exibe uma mensagem e um campo para que os estudantes digitem o código disponibilizado pelo professor ([Figura 43b](#)).

4.2.3 Responder questões

O sistema permite ao professor criar questões de múltipla escolha e de resposta livre. A [Figura 44](#) ilustra como fica apresentado no aplicativo as questões. A [Figura 44a](#) exibe uma questão de verdadeiro e falso e a [Figura 44b](#) ilustra uma questão aberta.

Figura 43 – Telas para os estudantes realizarem a frequência

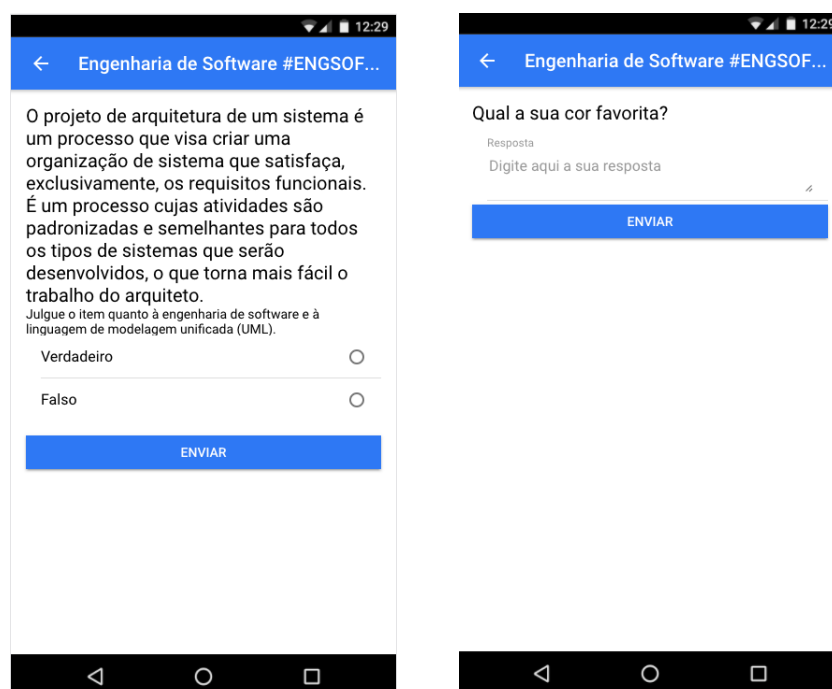
- (a) Botão para responder a chamada (b) Tela com campo para digitar código



Fonte – do autor (2017)

Figura 44 – Diferentes tipos de questões no aplicativo

- (a) Questão do tipo múltipla escolha (b) Questão do tipo resposta curta



Fonte – do autor (2017)

5 Testes de Usabilidade

Neste capítulo são apresentados e discutidos os testes de usabilidade utilizados para validar o software desenvolvido, apresentado no [Capítulo 4](#).

5.1 Teste de Usabilidade

Um dos requisitos do software apresentado no [Capítulo 3](#), subseção 3.3.2 era que:

Facilidade do uso e de criação de votação: O sistema não deve oferecer dificuldades de uso e de criação de questões;

Ou seja, trata-se um de um requisito não funcional do tipo produto (requisitos que especificam o comportamento do produto), mais especificamente um requisito de usabilidade. A usabilidade propõe-se a garantir que o produto seja fácil de aprender a usar, eficaz e agradável do ponto de vista do usuário ([ROGERS; SHARP; PREECE, 2013](#)). Para [Nielsen \(2012\)](#), a usabilidade é definida por cinco componentes de qualidade:

Apreensibilidade (*Learnability*): facilidade para os usuários realizarem tarefas básicas pela primeira vez.

Eficiência (*Efficiency*): rapidez com que os usuários realizam uma tarefa, uma vez aprendida.

Recordação (*Memorability*): facilidade em lembrar como fazer algo novamente, depois de um período sem uso.

Erros (*Errors*): o sistema deve ter uma baixa taxa de erros e ter capacidade de recuperação dos erros.

Satisfação (*Satisfaction*): quão agradável é utilizar o sistema.

5.1.1 Métodos, tarefas e usuários

Para verificar a usabilidade do software desenvolvido, realizou-se testes de usabilidade, com o objetivo de testar o quão usável é o software. Para isso, utilizou-se o seguinte método: teste de usabilidade que incluía a observação dos usuário utilizando o software e também a gravação em vídeo da tela do software. Além disso, um questionário de satisfação SUS (*System Usability Scale*) do usuário foi utilizado para descobrir como os usuários se sentiram usando o software.

5.1.1.1 Questionário de satisfação: SUS

O SUS é um método normalizado desenvolvido em 1986 por John Brooke. Ele é um questionário constituído por apenas 10 itens, gratuito e simples de usar. Utiliza-se a escala de Likert (valores 1 - *discordo totalmente* e 5 *concordo totalmente*) O critérios que o SUS ajuda a avaliar são (BROOKE et al., 1996):

Efetividade (*effectiveness*) a capacidade dos usuários de completar tarefas.

Eficiência (*efficiency*) o nível de esforço e recursos necessários completar as tarefas.

Satisfação (*satisfaction*) reações subjetivas dos usuários (a experiência foi satisfatória?).

O questionário SUS foi originalmente escrito na língua inglesa. Dessa forma, a versão para língua portuguesa utilizada nesse trabalho (disponível no [Apêndice A](#)) foi a desenvolvida por [Tenório et al. \(2011\)](#) que passou por um processo de tradução.

O resultado do SUS é a soma individual de cada item, resultando em um número entre 0 e 100. Para os itens 1, 3, 5, 7 e 9 deve-se subtrair 1 de cada resposta. Para os itens 2, 4, 6, 8 e 10 deve-se subtrair de 5 o valor de cada resposta. O escore final SUS é soma de todos os itens multiplicada por 2,5 (BROOKE et al., 1996).

5.1.1.2 Tarefas

As tarefas foram desenvolvidas de modo a permitir que os participantes percorressem todo o processo para permitir disponibilizar as questões para os alunos, consistindo basicamente nas seguintes tarefas: cadastrar uma turma e alunos, criar questões, iniciar uma sessão de questões, e controlar a sessão de questões. As tarefas foram escritas no seguinte formato:

Título: em poucas palavras o objetivo da tarefa.

Descrição: contexto da tarefa (situação para realizar a tarefa).

Tarefas: o que o participante deveria fazer.

O exemplo da [Figura 45](#) exhibe uma tarefa apresentada para os participantes seguindo o formato apresentado. A lista completa e detalhada das tarefas utilizadas no teste está disponível no [Apêndice B](#).

A seguir, apresenta-se quais ações eram pretendidas dos participantes em cada tarefa:

Figura 45 – Exemplo de uma tarefa do teste de usabilidade

Tarefa #3 Criar uma nova questão

- Com a turma criada e estudantes adicionados, você como professor quer adicionar questões para perguntar aos alunos na sala de aula.
 - Você deve tentar adicionar uma nova questão de **verdadeiro e falso** e indicar que ela é **FALSA**. Salve.
 - Tente criar outra questão de **múltipla escolha** com **4 alternativas**. Salve.

Fonte – do autor (2017)

Tarefa #1 Criar uma nova turma

- Espera-se que o professor identifique facilmente a opção para adicionar uma turma e habilite a opção para ativar a turma que ele criou.

Tarefa #2 Adicionar estudantes em uma turma

- Espera-se que o professor identifique o botão para adicionar estudantes. Na tela de adicionar estudantes, que ele identifique a opção *Adicionar manualmente*, adicione os estudantes e em seguida salve as alterações.

Tarefa #3 Criar uma nova questão

- Espera-se que o professor localize a aba *Questões* e identifique facilmente o botão para adicionar uma questão. No formulário para adicionar uma questão, que ele adicione a pergunta, alternativas e marque uma opção como correta.

Tarefa #4 Realizar frequência dos estudantes

- Nessa tarefa, o botão de frequência está na aba *Aula*. Seguindo as tarefas de forma sequencial, o professor estará na aba *Questões*. Nesse sentido, espera-se que o professor explore a aba *Aula* e identifique o botão *Frequência*.

Tarefa #5 Acessar a sala como estudante

- Agora no papel de aluno, espera-se que o professor relacione o código da turma que ele criou na tarefa 1 para ter acesso como estudante no aplicativo. Na tela de identificação do estudante, espera-se que ao ler a mensagem ele coloque o código de identificação de um estudante adicionado na tarefa 2.

Tarefa #6 Responder a frequência

- Já identificado no aplicativo do estudante, espera-se que o professor identifique o botão *Responder chamada* e entre com o código gerado para a frequência na tarefa 4. Por último, espera-se que ele verifique que no software do professor, o sistema identificou que um aluno realizou a frequência.

Tarefa #7 Iniciar sessão de perguntas

- Espera-se que o professor navegue para a aba *Questões* e identifique que para iniciar uma sessão de perguntas, ele deve selecionar pelo menos uma questão que ele criou na tarefa 3. Ele também deve explorar as opções de controle da sessão e ativar uma questão para votação.

Tarefa #8 Responder questão

- Com a questão já ativada, o professor deve facilmente escolher uma alternativa e enviar a resposta utilizando o aplicativo do estudante.

Tarefa #9 Finalizar sessão de perguntas

- Espera-se que o professor explore as opções de pausar, mostrar a resposta correta e mostrar o resultado da votação.

Tarefa #10 Encerrar sessão de perguntas

- Espera-se que o professor identifique a opção de encerrar a sessão de perguntas.

Tarefa #11 Verificar resposta dos estudantes

- Encerrando a sessão de perguntas, a sessão encerrada deve ser listada no software. Espera-se que o professor clique na opção detalhes da sessão listada e explore as respostas dos estudantes.

5.1.1.3 Usuários

Os participantes foram considerados usuários típicos, 4 professores da UNIVASF e um estudante de graduação de que já trabalhou como professor do ensino médio. Todos os participantes eram do sexo masculino.

5.1.2 Resultado e discussões

Com a utilização do questionário SUS foi possível mensurar o grau de satisfação quanto a usabilidade do software desenvolvido. Além disso, ele permite avaliar os componentes da qualidade indicadas por Nielsen (2012) distribuídas nas afirmativas do questionário:

- **Apreensibilidade:** afirmações 3, 4, 7 e 10:
 - AF3: Eu achei o sistema fácil de usar.
 - AF4: Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema
 - AF7: Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.
 - AF10: Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.
- **Eficiência:** afirmações 5, 6 e 8:
 - AF5: Eu acho que as várias funções do sistema estão muito bem integradas.
 - AF6: Eu acho que o sistema apresenta muita inconsistência.
 - AF8: Eu achei o sistema atrapalhado de usar.
- **Recordação:** afirmação 2:
 - AF2: Eu acho o sistema desnecessariamente complexo.
- **Erros:** afirmação 6:
 - AF6: Eu acho que o sistema apresenta muita inconsistência.
- **Satisfação:** afirmações 1, 4 e 9:
 - AF1: Eu acho que gostaria de usar esse sistema frequentemente.
 - AF4: Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.
 - AF9: Eu me senti confiante ao usar o sistema.

A [Figura 46](#) apresenta as respostas dos 5 participantes no teste de usabilidade. A coluna *Afirmações* contém as afirmações do questionário SUS, seguida pelas colunas $P1, \dots, P5$, que representam cada participante. Os valores da escala de Likert (1 - discordo totalmente e 5 - concordo plenamente) escolhidas pelos participantes estão mostradas na tabela.

Abaixo das afirmações apresenta-se a *Pontuação SUS* de cada questionário. O cálculo para obter tais resultados foi apresentado na [subseção 5.1.1.1](#). Obteve-se uma média de 85,50 com desvio padrão de 9,42.

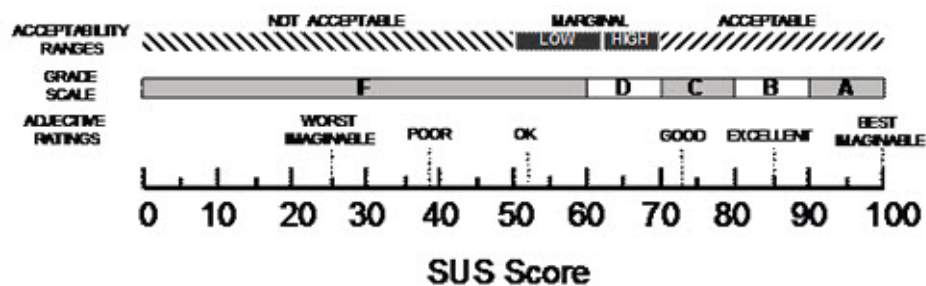
A [Figura 47](#) apresenta como alguns estudos classificam a nota média do SUS. Dessa forma, o software pode ser classificado como *80-90*, *aceitável*, *B* e *Excelente*. Assim, pode-se dizer que o software apresenta os cinco componentes de qualidade de usabilidade definidos por [Nielsen \(2012\)](#), ou seja, apreensibilidade, eficiência, recordação, erros e satisfação de forma excelente.

Figura 46 – Resultado do questionário SUS

Resultado do questionário SUS					
Afirmações	Participantes				
	P1	P2	P3	P4	P5
AF1 - Eu acho que gostaria de usar esse sistema frequentemente.	5	4	4	4	5
AF2 - Eu acho o sistema desnecessariamente complexo.	4	1	2	2	1
AF3 - Eu achei o sistema fácil de usar.	4	5	3	4	5
AF4 - Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.	1	1	4	4	2
AF5 - Eu acho que as várias funções do sistema estão muito bem integradas.	4	5	5	4	4
AF6 - Eu acho que o sistema apresenta muita inconsistência.	2	1	1	1	1
AF7 - Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.	5	5	4	4	5
AF8 - Eu achei o sistema atrapalhado de usar.	1	1	1	2	1
AF9 - Eu me senti confiante ao usar o sistema.	5	4	4	5	5
AF10 - Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.	1	2	2	1	1
Pontuação SUS	85	95	75	77,5	95

Fonte – do autor (2017)

Figura 47 – Comparação das classificações adjetivas, pontuações de aceitabilidade e escalas de classificação escolar, em relação ao escore médio do SUS



Fonte – (BANGOR; KORTUM; MILLER, 2009)

5.1.2.1 Análise das gravações por tarefa

Para cada uma das tarefas propostas, as principais observações de cada tarefa estão descritas a seguir:

Tarefa #1 Criar uma nova turma

Expectativa: Espera-se que o professor identifique facilmente a opção para adicionar uma turma e habilite a opção para ativar a turma que ele criou.

Observações: Todos os participantes conseguiram concluir essa tarefa sem ajuda. O botão para adicionar a turma foi identificado com facilidade. 1 participante não ativou a turma.

Melhorias: Quando nenhuma turma estiver ativada, exibir uma mensagem indicando: *Você ainda não tem uma turma ativada. Ative uma turma para iniciar uma sessão de perguntas.* Outra solução seria ativar a primeira turma criada automaticamente e exibir uma mensagem para o usuário.

Tarefa #2 Adicionar estudantes em uma turma

Esperado: Espera-se que o professor identifique o botão para adicionar estudantes. Na tela de adicionar estudantes, que ele identifique a opção *Adicionar manualmente*, adicione os estudantes e em seguida salve as alterações.

Observações: 2 participantes tiveram dificuldades para encontrar o botão para adicionar estudantes no canto inferior direito da tela.

Solução: Quando a lista de estudantes estiver vazia, exibir uma mensagem indicando: *Você ainda não tem nenhum estudante cadastrado na turma. Comece adicionando estudantes nas opções ao lado.*

Tarefa #3 Criar uma nova questão

Esperado: Espera-se que o professor localize a aba *Questões* e identifique facilmente o botão para adicionar uma questão. No formulário para adicionar uma questão, que ele adicione a pergunta, alternativas e marque uma opção como correta.

Observações: Todos os participantes concluíram essa tarefa sem ajuda. 2 participantes tiveram um pouco de dificuldade para marcar uma alternativa como correta.

Melhorias: Colocar a descrição *Correta* ao lado do botão que seleciona a questão como correta. Outra melhoria seria que quando o tipo de questão for de múltipla escolha, já adicionar duas alternativas como padrão, diminuindo assim dois cliques no processo.

Tarefa #5 Acessar a sala como estudante

Esperado: Agora no papel de aluno, espera-se que o professor relacione o código da turma que ele criou na tarefa 1 para ter acesso como estudante no aplicativo. Na tela de identificação do estudante, espera-se que ao ler a mensagem ele coloque o código de identificação de um estudante adicionado na tarefa 2.

Observações: 1 participante precisou de ajuda para completar a tarefa. Uma observação de um participante foi que ele estava precisando digitar o código da turma a todo instante.

Melhorias: Quando um estudante entrar em uma turma com um código de acesso, o sistema deve guardar o código em uma lista de acessos recentes para facilitar os acessos futuros.

Tarefa #7 Iniciar sessão de perguntas

Esperado: Espera-se que o professor navegue para a aba *Questões* e identifique que para iniciar uma sessão de perguntas, ele deve selecionar pelo menos uma questão que ele criou na tarefa 3. Ele também deve explorar as opções de controle da sessão e ativar uma questão para votação.

Observações: 3 participantes pediram ajuda para completar essa tarefa. Percebeu-se que na aba *Aula*, era exibido o aviso - *Nenhuma sessão encerrada*, fazendo com que o usuário entenda-se que uma sessão deveria ser iniciada em algum lugar na aba *Aula*, já que a mesma fazia referência as sessões encerradas.

Melhorias: Na aba *Aula* quando um aviso indicando que: *Para iniciar uma sessão, na aba QUESTÕES selecione uma ou mais questões e clique em INICIAR SESSÃO*. Na aba *Questões*, quando o professor clicar no botão *INICIAR SESSÃO* sem nenhuma questão selecionada, deve aparecer uma mensagem indicando: *Selecione uma ou mais questões para iniciar uma sessão*.

Tarefas #4, 6, 8, 9, 10 e 11

Observação: Todos os participantes concluíram essas tarefas com sucesso e sem ajuda.

5.2 Teste em Sala de Aula

O teste de usabilidade descrito nas seções anteriores, pode-se classificá-lo além de um teste de usabilidade, também como um teste alpha. Pressman (2009) define o teste alpha como um teste conduzido pelo usuário ou cliente no ambiente do desenvolvedor, este observando o usuário, registrando erros e problemas de uso.

O teste de usabilidade foi conduzido em um ambiente definido pelo desenvolvedor, este controlando todas as variáveis do ambiente (internet, computador, celular, etc), capturando as ações do usuário e fazendo anotações sobre o comportamento, vozes, etc.

Quando um software é testado em um ambiente não controlado pelo desenvolvedor, chama-se esse teste de *teste beta*. Para Pressman (2009), o teste beta é aquele em que uma versão do software é disponibilizado aos usuários para que possam experimentar e eventualmente levantar os problemas encontrados com os desenvolvedores.

Dessa forma, realizou-se um teste *beta* para testar funcionamento do aplicativo do estudante em diferentes *smartphones* e/ou computadores e o uso do software do professor na sala de aula. Para isso, uma versão do sistema operacional Android do aplicativo foi gerada e disponibilizada na loja na loja online da Google para distribuição de aplicações, Google Play¹. Dessa forma, tanto o professor quanto os estudantes puderam instalar o aplicativo nos seus aparelhos.

5.2.1 Participantes

O teste foi realizado na disciplina *Modelagem e Simulação* do curso de graduação em Engenharia de Computação da Universidade Federal do Vale do São Francisco (UNIVASF). O objetivo geral dessa disciplina é “tornar os alunos capazes de construir sistemas de simulação, de validar e verificar os modelos, e de validar os resultados da simulação” (CCOMP, 2011). O teste foi realizado no dia 19 de julho de 2017, no campus Juazeiro - BA.

A turma era composta por um professor e 13 alunos, com faixa etária entre 20 e 27 anos, 3 (23%) alunos do sexo feminino e 10 (74%) alunos do sexo masculino. Todos os estudantes tinham acesso a um *smartphone*, 12 (92%) do sistema operacional Android e um 1 (8%) do sistema operacional Windows Phone. Nenhum deles havia utilizado um aplicativo de sistema de resposta na sala de aula anteriormente.

O professor da disciplina anteriormente ao teste na sala de aula, participou do teste de usabilidade, dessa forma ele já tinha os conhecimentos básicos do sistema.

5.2.2 Preparação antes do teste

Na semana anterior ao teste, foi feita uma visita no horário da aula para identificar as principais dificuldades que poderiam dificultar a realização do teste. Primeiramente, foi verificado que o sinal de acesso à internet da universidade não era satisfatório na sala de aula, sendo assim, foi necessário a utilização de um roteador para melhorar o sinal Wi-Fi na sala de aula. Além disso, considerando que apenas um aluno utilizava um *smartphone* com o sistema operacional Windows Phone, identificou-se que era necessário um aparelho adicional com o sistema operacional Android no dia do teste.

Cada estudante participante foi orientado a instalar o aplicativo em seu respectivo aparelho. Enquanto isso, foi apresentado para os alunos o objetivo do sistema que estava

¹ Endereço do aplicativo no Google Play: play.google.com/store/apps/details?id=apps.pedrosobral.rsa

sendo testado, bem como informado que o teste era para avaliar o funcionamento do sistema, e não os usuários. Em seguida, três questões (duas do tipo múltipla escolha e uma do tipo aberta) foram apresentadas para verificar o funcionamento do sistema.

Com a preparação bem sucedida, o professor voluntário ficou então responsável por preparar uma aula com questões para serem respondidas pelos alunos com o aplicativo.

5.2.3 Aula

No dia do teste formal, novamente foi reforçado para os alunos o objetivo do sistema que estava sendo testado, bem como comunicado que o teste era para avaliar o funcionamento do sistema, e não os usuários.

5.2.4 Questionário

No final da aula, os estudantes foram convidados a responderem um questionário online de avaliação.

6 Considerações Finais e Trabalhos Futuros

6.1 Considerações Finais

Este trabalho teve como objetivo principal o desenvolvimento de um sistema de resposta para uso em sala de aula, que possibilita-se aos professores e aos estudantes uma ferramenta de software livre, que permitesse usar *smartphones* como *clickers*.

Nesse sentido, os resultados obtidos são animadores ao apontar que a solução testada alcançou bons índices de usabilidade tanto pelos professores, quanto pelo estudantes. Além disso, a percepção otimista do professor voluntário no experimento do uso do sistema na sala de aula e o interesse dele em continuar a usar o sistema em suas aulas reforçam que o objetivo foi alcançado, contribuindo assim com a comunidade acadêmica com um software livre - ainda que no início do seu desenvolvimento - de qualidade.

6.2 Trabalhos Futuros

Acredita-se que a melhor forma de continuar esse trabalho é utilizando o sistema desenvolvido em um turma por um longo período para assim descobrir e desenvolver outros requisitos que não foram considerados nesse trabalho. Além disso, pode-se fazer um estudo de caso para verificar os benefícios do sistema na sala de aula, ou seja, aqueles que foram descritos no [Capítulo 2, seção 2.2](#), como aumento na frequência escolar, melhora na atenção, engajamento da turma, etc.

Referências

- ABRAHAMSON, L.; BRADY, C. A Brief History of Networked Classrooms to 2013:. **International Journal of Quality Assurance in Engineering and Technology Education**, v. 3, n. 3, p. 1–54, 2014. ISSN 2155-496X. Citado na página 14.
- ARAÚJO, I. S.; MAZUR, E. Instrução pelos colegas e ensino sob medida: uma proposta para o engajamento dos alunos no processo de ensino-aprendizagem de Física. **Caderno Brasileiro de Ensino de Física**, v. 30, n. 2, p. 362–384, 2013. ISSN 2175-7941. Citado 4 vezes nas páginas 11, 15, 21 e 22.
- ARNESEN, K. et al. Experiences with use of various pedagogical methods utilizing a student response system - Motivation and learning outcome. **Electronic Journal of e-Learning**, v. 11, n. 3, p. 169–181, 2013. ISSN 14794403. Citado na página 19.
- BANGOR, A.; KORTUM, P.; MILLER, J. Determining what individual sus scores mean: Adding an adjective rating scale. **Journal of usability studies**, Usability Professionals' Association, v. 4, n. 3, p. 114–123, 2009. Citado na página 66.
- BARRAGUÉS, J.; MORAIS, A.; GUIASOLA, J. Use of a classroom response system (CRS) for teaching mathematics in Engineering with large groups. **Education in a technological world: communicating current and emerging research and technological efforts**, p. 572–580, 2011. Citado 2 vezes nas páginas 12 e 18.
- BARROS, J. A. D. et al. Engajamento interativo no curso de Física I da UFJF. **Revista Brasileira de Ensino de Física**, v. 26, n. 1, p. 63–69, 2004. Citado na página 21.
- BECK, K. et al. **Manifesto para Desenvolvimento Ágil de Software**. 2001. Disponível em: <<http://agilemanifesto.org/iso/ptbr/manifesto.html>>. Acesso em: 18 Mai. 2017. Citado na página 25.
- BLASCO-ARCAS, L. et al. Using clickers in class. the role of interactivity, active collaborative learning and engagement in learning performance. **Computers and Education**, Elsevier Ltd, v. 62, p. 102–110, 2013. ISSN 03601315. Citado 2 vezes nas páginas 12 e 15.
- BLOOD, E.; GULCHAK, D. Embedding “Clickers” Into Classroom Instruction: Benefits and Strategies. **Intervention in School & Clinic**, v. 48, n. 4, p. 246–253, 2013. ISSN 1053-4512. Citado 3 vezes nas páginas 12, 18 e 19.
- BRASIL, P. **Portal do Professor disponibiliza lista de livros sobre novas tecnologias**. 2014. Disponível em: <<http://www.brasil.gov.br/educacao/2014/07/portal-do-professor-disponibiliza-lista-de-livros-sobre-novas-tecnologias#wrapper>>. Acesso em: 18 Ago. 2016. Citado na página 20.
- BROOKE, J. et al. Sus-a quick and dirty usability scale. **Usability evaluation in industry**, London, United kingdom, v. 189, n. 194, p. 4–7, 1996. Citado na página 62.
- CALDWELL, J. E. Clickers in the Large Classroom: Current Research and Best-Practice Tips. **CBE - Life Sciences Education**, v. 6, p. 1–15, 2007. ISSN 0004-069X. Citado 4 vezes nas páginas 16, 18, 19 e 20.

CCOMP. **Projeto Pedagógico do Curso de Engenharia de Computação**. 2011. 242 p. Disponível em: <http://www.cecomp.univasf.edu.br/wp-content/uploads/2016/06/PPC_CECOMP.pdf>. Acesso em: 15 Jul. 2017. Citado na página 69.

CHARLES, C.; JAMES, A. **Active Learning: Creating Excitement in the Classroom**. Ashe-eric. Washington, D.C: The George Washington University, School of Education and Human Development, 1991. 121 p. ISBN 1878380087. Citado 2 vezes nas páginas 11 e 20.

CROUCH, C. H.; MAZUR, E. Peer instruction: Ten years of experience and results. **American Journal of Physics**, v. 69, n. 9, p. 970–977, 2001. ISSN 00029505. Citado 2 vezes nas páginas 21 e 22.

CROUCH, C. H. et al. Peer Instruction: Engaging Students One-on-One, All At Once. **Research-Based Reform of University Physics**, p. 1–55, 2007. ISSN 10476938. Citado 2 vezes nas páginas 19 e 22.

CUBRIC, M.; JEFFERIES, A. The benefits and challenges of large-scale deployment of electronic voting systems: University student views from across different subject groups. **Computers and Education**, Elsevier Ltd, v. 87, p. 98–111, 2015. ISSN 03601315. Citado na página 20.

D'INVERNO, R. A.; DAVIS, H. C.; WHITE, S. Using a personal response system for promoting student interaction. **Teaching Mathematics and its Application**, v. 22, n. 4, p. 163–169, 2003. ISSN 02683679. Citado na página 18.

DRIFTY. **Ionic: Advanced HTML5 Hybrid Mobile App Framework**. 2016. Disponível em: <<http://ionicframework.com/>>. Acesso em: 20 Ago. 2016. Citado na página 31.

FEATHERS. **Feathers: A minimalist real-time framework for tomorrow's apps**. 2017. Disponível em: <<https://docs.feathersjs.com/>>. Acesso em: 22 Mai. 2017. Citado na página 35.

FIES, C.; MARSHALL, J. Classroom response systems: A review of the literature. **Journal of Science Education and Technology**, v. 15, n. 1, p. 101–109, 2006. ISSN 10590145. Citado 2 vezes nas páginas 14 e 19.

FOTARIS, P. et al. Climbing Up the Leaderboard : An Empirical Study of Applying Gamification Techniques to a Computer Programming Class. **Electronic Journal of E-Learning**, v. 14, n. 2, 2016. ISSN 14794403. Citado 2 vezes nas páginas 12 e 16.

FOUNDATION, N. **About Node.js**. 2017. Disponível em: <<https://nodejs.org/en/about/>>. Acesso em: 20 Mai. 2017. Citado na página 30.

FREEMAN, S. et al. Active learning increases student performance in science, engineering, and mathematics. **Proceedings of the National Academy of Sciences of the United States of America**, v. 111, n. 23, p. 8410–5, 2014. ISSN 1091-6490. Citado na página 21.

GOK, T. A comparison of students' performance, skill and confidence with peer instruction and formal education. **Journal of Baltic Science Education**, v. 12, n. 6, p. 747–758, 2013. ISSN 16483898. Citado na página 21.

- GOOGLE. **Protractor: end to end testing for AngularJS**. 2017. Disponível em: <<http://www.protractortest.org/#/>>. Acesso em: 24 Mai. 2017. Citado na página 46.
- HORNE, A. **An Evaluation of an Electronic Student Response System in Improving Class-wide Behavior**. 56 p. Tese (Graduate Theses and Dissertations) — University of South Florida, 2015. Citado na página 18.
- HUNSU, N. J.; ADESOPE, O.; BAYLY, D. J. A meta-analysis of the effects of audience response systems (clicker-based technologies) on cognition and affect. **Computers and Education**, Elsevier Ltd, v. 94, p. 102–119, 2016. ISSN 03601315. Citado 4 vezes nas páginas 11, 12, 18 e 19.
- IBGE. **Acesso à internet e à televisão e posse de telefone móvel celular para uso pessoal: 2014**. Rio de Janeiro: IBGE, 2016. 89 p. ISBN 9788524043758. Disponível em: <www.mc.gov.br/publicacoes/doc_download/2799-pnad-tic-2014>. Acesso em: 15 ago 2016. Citado na página 12.
- KAY, R. H.; LESAGE, A. A strategic assessment of audience response systems used in higher education. **Australasian Journal of Educational Technology**, v. 25, n. 2, p. 235–249, 2009. ISSN 14495554. Citado 6 vezes nas páginas 11, 14, 15, 16, 19 e 20.
- KAYA, A.; BALTA, N. Taking Advantages of Technologies: Using the Socrative in English Language Teaching Classes. **International Journal of Social Sciences & Educational Studies**, v. 2, n. 3, p. 4–12, 2016. Citado 3 vezes nas páginas 12, 16 e 18.
- KORTEMAYER, G. The Psychometric Properties of Classroom Response System Data: A Case Study. **Journal of Science Education and Technology**, Springer Netherlands, n. March, p. 1–14, 2016. ISSN 15731839. Citado na página 19.
- KORTEMAYER, G.; DIAS, E.; CRUZ, H. The effect of formative assessment in Brazilian university physics courses. **Revista Brasileira de Ensino de Física**, v. 33, n. n. 4, p. 4501, 2011. ISSN 01024744. Citado 2 vezes nas páginas 14 e 15.
- KULATUNGA, U.; RAMEEZDEEN, R. Use of Clickers to Improve Student Engagement in Learning: Observations from the Built Environment Discipline. **International Journal of Construction Education and Research**, v. 10, n. 1, p. 3–18, 2014. ISSN 1557-8771. Citado na página 18.
- LABS, P. **Jasmine: A JavaScript Testing Framework**. 2017. Disponível em: <<https://github.com/jasmine/jasmine>>. Acesso em: 24 Mai. 2017. Citado na página 45.
- LANTZ, M. E.; STAWISKI, A. Effectiveness of clickers: Effect of feedback and the timing of questions on learning. **Computers in Human Behavior**, Elsevier Ltd, v. 31, n. 1, p. 280–286, 2014. ISSN 07475632. Citado 2 vezes nas páginas 16 e 19.
- MATTOS, E. B. D. A. **Imunologia e Biotecnologia: o clicker como recurso didático estratégico**. 109 p. Tese (Dissertação de Mestrado) — Universidade Federal Fluminense, 2015. Citado 3 vezes nas páginas 12, 15 e 18.
- MAYER, R. E. et al. Clickers in college classrooms: Fostering learning with questioning methods in large lecture classes. **Contemporary Educational Psychology**, Elsevier Inc., v. 34, n. 1, p. 51–57, 2009. ISSN 0361476X. Citado 2 vezes nas páginas 16 e 18.

MAZUR, E. Farewell, Lecture? **Science**, v. 323, p. 50–51, 2009. ISSN 1095-9203. Citado 2 vezes nas páginas 21 e 22.

MONGODB. **NoSQL Databases Explained**. 2017. Disponível em: <<https://www.mongodb.com/nosql-explained>>. Acesso em: 20 Mai. 2017. Citado 2 vezes nas páginas 31 e 39.

MOODLE. **Moodle Mobile 2.0 spec**. 2015. Disponível em: <https://docs.moodle.org/dev/Moodle_Mobile_2.0_spec>. Acesso em: 20 Ago. 2016. Citado na página 31.

MORAN, J. M.; MASETO, M. T.; BEHRENS, M. A. **Novas Tecnologias e mediação pedagógica**. 10. ed. Campinas, SP: Papirus, 2006. 173 p. ISBN 85-308-0594-1. Citado na página 20.

MORATELLI, K.; DEJARNETTE, N. K. Clickers to the rescue. **The Reading Teacher**, v. 67, n. 8, p. 586–593, 2014. ISSN 00340561. Citado 2 vezes nas páginas 15 e 18.

MORRELL, L. J.; JOYCE, D. A. Interactive lectures: Clickers or personal devices? **F1000Research**, v. 64, p. 1–12, 2015. ISSN 2046-1402. Citado 2 vezes nas páginas 15 e 20.

NEILSON, M. et al. Students perceptions regarding the use of tophat as an interactive tool in meat science clall. **Meat Science**, 2016. Citado na página 16.

NETWORK, M. D. **JavaScript**. 2017. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>>. Acesso em: 20 Mai. 2017. Citado na página 29.

NIELSEN, J. **Usability 101: Introduction to Usability**. 2012. Disponível em: <<https://www.nngroup.com/articles/usability-101-introduction-to-usability/>>. Acesso em: 20 Jun. 2017. Citado 3 vezes nas páginas 61, 64 e 65.

PRESSMAN, R. S. **Software Engineering A Practitioner's Approach**. 7. ed. New York, NY 10020: McGraw-Hill, 2009. 930 p. ISSN 1098-6596. ISBN 978-0-07-337597-7. Citado 3 vezes nas páginas 24, 68 e 69.

PRINCE, M. Does active learning work? A review of the research. **Journal of Engineering education- Washington**, v. 93, n. July, p. 223–232, 2004. ISSN 1069-4730. Citado na página 20.

PUENTE, S. G.; SWAGTEN, H. Designing learning environments to teach interactive Quantum Physics. **European Journal of Engineering Education**, v. 37, n. June 2013, p. 37–41, 2012. ISSN 03043797. Citado na página 16.

RANA, N. P.; DWIVEDI, Y. K. Using Clickers in a Large Business Class: Examining Use Behavior and Satisfaction. **Journal of Marketing Education**, v. 38, n. 1, p. 47–64, 2016. ISSN 02734753. Citado 3 vezes nas páginas 12, 18 e 19.

ROGERS, Y.; SHARP, H.; PREECE, J. **Design de Interação - 3ed**. Bookman Editora, 2013. ISBN 9788582600085. Disponível em: <https://books.google.com.br/books?id=d_s4AgAAQBAJ>. Citado na página 61.

SCHADE, A. **Competitive Usability Evaluations: Learning from Your Competition**. 2013. Disponível em: <<https://www.nngroup.com/articles/competitive-usability-evaluations/>>. Acesso em: 10 Mai. 2017. Citado na página 27.

- SCHMIDT, B. Teaching engineering dynamics by use of peer instruction supported by an audience response system. **European Journal of Engineering Education**, v. 36, n. 5, p. 413–423, 2011. ISSN 0304-3797. Citado na página 18.
- SOCRATIVE. 2016. Disponível em: <<http://socrative.com/pricing>>. Acesso em: 16 Ago. 2016. Citado na página 16.
- SOMMERVILLE, I. **Engenharia de software**. [S.l.]: Pearson Prentice Hall, 2011. ISBN 9788579361081. Citado 3 vezes nas páginas 24, 25 e 29.
- STOWELL, J. R. Use of clickers vs. mobile devices for classroom polling. **Computers and Education**, Elsevier Ltd, v. 82, p. 329–334, 2015. ISSN 03601315. Citado 2 vezes nas páginas 15 e 20.
- STRASSER, N. Who Wants to Pass Math? Using Clickers in Calculus. **Journal of College Teaching Learning**, v. 7, n. 3, p. 49–52, 2010. ISSN 15440389. Citado 4 vezes nas páginas 15, 16, 19 e 20.
- SUN, J. C. Y. Influence of polling technologies on student engagement: An analysis of student motivation, academic performance, and brainwave data. **Computers and Education**, Elsevier Ltd, v. 72, p. 80–89, 2014. ISSN 03601315. Citado na página 12.
- SYSTEMS and software engineering – Vocabulary. **ISO/IEC/IEEE 24765:2010(E)**, p. 1–418, Dec 2010. Citado na página 24.
- TENÓRIO, J. M. et al. Desenvolvimento e avaliação de um protocolo eletrônico para atendimento e monitoramento do paciente com doença celíaca. **Revista de Informática Teórica e Aplicada**, v. 17, n. 2, p. 210–220, 2011. Citado 2 vezes nas páginas 62 e 79.
- TERRION, J. L.; ACETI, V. Perceptions of the effects of clicker technology on student learning and engagement: A study of freshmen Chemistry students. **Research in Learning Technology**, v. 20, n. 2, 2012. ISSN 21567069. Citado 4 vezes nas páginas 12, 18, 19 e 20.
- THAMPY, H.; AHMAD, Z. How to. Use audience response systems. **Education for Primary Care**, v. 25, n. 5, p. 294–296, 2014. ISSN 14739879. Citado na página 19.
- TITMAN, A.; LANCASTER, G. Personal response systems for teaching postgraduate statistics to small groups. **Journal of Statistics Education**, v. 19, n. 2, p. 1–20, 2011. ISSN 10691898. Citado na página 18.
- TOPHAT. 2016. Disponível em: <<http://tophat.com/customers/>>. Acesso em: 16 Ago. 2016. Citado na página 16.
- TRINDADE, J. Promoção da interatividade na sala de aula com Socrative: estudo de caso. **Indagatio Didactica**, v. 6, n. 1, 2014. Citado na página 16.
- UNIVESPTV. **D-29 - Avaliação da Aprendizagem: Formativa ou Somativa?** 2013. Disponível em: <<https://youtu.be/G5VEkMf5DRk>>. Acesso em: 16 Ago. 2016. Citado na página 19.
- VELASCO, M.; ÇAVDAR, G. Teaching Large Classes with Clickers: Results from a Teaching Experiment in Comparative Politics. **PS: Political Science & Politics**, v. 46, n. 04, p. 823–829, 2013. ISSN 1049-0965. Citado 2 vezes nas páginas 16 e 20.

WATKINS, B. J.; MAZUR, E. Retaining Students in Science, Technology, Engineering, and Mathematics (STEM) Majors. **Journal of College Science Teaching**, v. 42, n. 5, p. 36–41, 2013. ISSN 0047-231X. Citado na página 23.

WEBSOCKET. **About HTML5 WebSocket**. 2016. Disponível em: <<https://www.websocket.org/aboutwebsocket.html>>. Acesso em: 20 Ago. 2016. Citado na página 31.

Apêndices

APÊNDICE A – Questionário SUS - System Usability Scale

Questionário SUS aplicado aos participantes dos testes de usabilidade. Versão em língua portuguesa desenvolvida por (TENÓRIO et al., 2011) que passou por um processo de tradução.

Cada um dos itens do questionário era composto por alternativas na escala de Likert como a seguir:

1. Discordo totalmente.
2. Discordo parcialmente.
3. Não concordo, nem discordo.
4. Concordo parcialmente.
5. Concordo totalmente.

Questionário:

1. Eu acho que gostaria de usar esse sistema frequentemente.
2. Eu acho o sistema desnecessariamente complexo.
3. Eu achei o sistema fácil de usar.
4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.
5. Eu acho que as várias funções do sistema estão muito bem integradas.
6. Eu acho que o sistema apresenta muita inconsistência.
7. Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.
8. Eu achei o sistema atrapalhado de usar.
9. Eu me senti confiante ao usar o sistema.
10. Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.

APÊNDICE B – Tarefas - Testes de Usabilidade

1. Tarefa #1 Criar uma nova turma

- Suponha que você seja o professor da disciplina Matemática Discreta. Você deseja criar uma nova turma.
 - Tente criar a turma Mat. Discreta com código de acesso DISC17.
 - Ative a turma que você acabou de criar.

2. Tarefa #2 Adicionar estudantes em uma turma

- Agora que você criou a turma DISC17, você deseja inserir estudantes para que tenham acesso a nova turma criada.
 - Tente adicionar o estudante de nome Pedro com ID 101 e o estudante Luiz com ID 102.

3. Tarefa #3 Criar uma nova questão

- Com a turma criada e estudantes adicionados, você como professor quer adicionar questões para perguntar aos alunos na sala de aula.
 - Você deve tentar adicionar uma nova questão de verdadeiro e falso e indicar que ela é FALSA. Salve.
 - Tente criar outra questão de múltipla escolha com 4 alternativas. Salve.

4. Tarefa #4 Realizar frequência dos estudantes

- Agora você está preparado para a sua aula. Na sala de aula você quer realizar a frequência dos estudantes pelo sistema.
 - Inicie o processo para realizar frequência dos estudantes pelo sistema.
 - Siga a próxima tarefa.

5. Tarefa #5 Acessar a sala como estudante

- Suponha que você é aluno da turma de Discreta com código de acesso DISC17 e o seu código de identificação é 101.
 - Você chegou na sala de aula.
 - Tente fazer o login no sistema.

6. Tarefa #6 Responder a frequência

- O professor já iniciou o processo para realizar a frequência da sala pelo aplicativo.
 - Tente realizar a frequência pelo aplicativo.
 - Encerre a frequência no aplicativo do professor.

7. Tarefa #7 Iniciar sessão de perguntas

- Chegou o momento de apresentar algumas questões para os estudantes.
 - Tente iniciar uma sessão com as perguntas que você criou.
 - Ative uma questão para votação e veja o que acontece no aplicativo do estudante.
 - Siga para a próxima tarefa.

8. Tarefa #8 Responder questão

- A questão foi disponibilizada pelo professor para ser respondida.
 - Responda a questão pelo aplicativo
 - Veja o que acontece na tela do professor.

9. Tarefa #9 Finalizar sessão de perguntas

- Chegou o momento de apresentar algumas questões para os estudantes.
 - Pause a votação
 - Mostre qual a resposta correta
 - Mostre o resultado da votação.

10. Tarefa #10 Encerrar sessão de perguntas

- Suponha que a aula acabou.
 - Tente finalizar a sessão de questões.

11. Tarefa #11 Verificar resposta dos estudantes

- Suponha que agora você quer saber como cada estudante respondeu a questões.
 - Veja se você consegue obter essa informação no sistema.