UNIVERSIDADE ESTADUAL DE MONTES CLAROS

Centro de Ciências Exatas e Tecnológicas

Curso de Bacharelado Sistemas de Informação

Ryan Lucas Silva Lemos

APLICAÇÃO DO EXTREME PROGRAMMING NO DESENVOLVIMENTO DE UM AMBIENTE WEB DE ENSINO E APRENDIZAGEM DA LÍNGUA INGLESA

Montes Claros/MG

mês/2019

**Ryan Lucas Silva Lemos**

**APLICAÇÃO DO EXTREME PROGRAMMING NO DESENVOLVIMENTO DE UM AMBIENTE WEB DE ENSINO E APRENDIZAGEM DA LÍNGUA INGLESA**

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Estadual de Montes Claros como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientadora: **Prof.ª. Christine Martins de Matos, MESTRA**.

**Montes Claros/MG**

**mês/2019**

**APLICAÇÃO DO EXTREME PROGRAMMING NO DESENVOLVIMENTO DE UM AMBIENTE WEB DE ENSINO E APRENDIZAGEM DA LÍNGUA INGLESA**

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Estadual de Montes Claros como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Montes Claros, data de 2019.

Membros:

Orientadora: PROFESSORA CHRISTINE MARTINS DE MATOS, MESTRA.

Universidade Estadual de Montes Claros

PROFESSOR 1

Universidade Estadual de Montes Claros

PROFESSOR 2

Universidade Estadual de Montes Claros

**Montes Claros/MG mês de 2019**

***Dedico este trabalho a Deus que nos momentos mais difíceis da vida me deu forças para que eu continuasse a caminhada. Também dedico àqueles que estiveram ao meu lado desde o início, e que motivava a continuar: familiares, orientadora e amigos.***

**AGRADECIMENTOS**

Agradeço a Deus pela sabedoria, força e paciência que me deu durante toda a minha vida, e especialmente no tempo em que estive realizando esse trabalho.

Agradeço a Christine Martins de Matos, minha orientadora, que me deu suporte, me ensinou e me auxiliou de todas as formas possíveis, com toda paciência e dedicação.

“A integração das tecnologias digitais na educação precisa ser feita de modo criativo e crítico, buscando desenvolver a autonomia e a reflexão dos seus envolvidos, para que não sejam apenas receptores de informações”. (Lilian Bacich).

**RESUMO**

Aqui vem o resumo

Palavras-chave: desenvolvimento, sistema *web*.

**ABSTRACT**

Here comes the abstract.

Keywords: development, web system.

LISTA DE ABREVIATURAS E SIGLAS

BPMN *Business Process Model and Notation*

CALL *Computer Assisted Language Learning*

CCAA Centro de Cultura Anglo Americana

CSS *Cascading Style Sheets*

EAD Educação a distância

HTML *Hyper Text Markup Language*

IDE *Integrated Development Environment*

IEEE *Institute of Eletrical and Eletronics Engineers*

IHC Interação Humano-Computador

ILC *International Language Center*

JS JavaScript

MVC *Model*, *View* e *Controller*

PHP *Hypertext PreProcessor*

SGBD Sistema de Gerenciamento de Banco de Dados

SPA *Single Page Application*

SQL *Structured Query Language*

TS TypeScript

Unimontes Universidade Estadual de Montes Claros

URL *Uniform Resource Locator*

XP *eXtreme Programming*

Sumário

1 INTRODUÇÃO 12

2 Referencial teórico 14

2.1 Educação a distância – ambiente virtual 14

2.1.1 Metodologias/sistemas de apoio de ensino de idiomas 14

2.2 Desenvolvimento e tecnologias de sistemas Web 17

2.2.1 Criptografia e controle de acessos 18

2.2.2 Interação humano computador (IHC) 19

2.2.3 Engenharia de Software 20

2.2.3.1 Modelagem de processos com o *Business Process Model and Notation* (BPMN) 22

2.2.3.2 Metodologia Ágil 26

2.2.3.3 *Extreme Programming* (XP) 27

2.2.4 Tecnologias para desenvolvimento WEB 30

2.2.4.1 *Hyper Text Markup Language* (HTML) 30

2.2.4.2 *Cascading Style Sheets* (CSS) 31

2.2.4.3 JavaScript (JS) 34

2.2.4.4 TypeScript 35

2.2.4.5 Angular 36

2.2.4.6 *Hypertext PreProcessor* (PHP) 36

2.2.4.7 *Framework* Laravel 37

2.2.4.8 *Representational State Transfer Application Programming Interfaces* (API REST) 38

2.2.5 Sistema de Gerenciamento de Banco de Dados (MySQL) 39

3 desenvolvimento do ambiente proposto 41

3.1 Ferramentas de desenvolvimento utilizadas 41

3.2 Release 1 – Cadastros Básicos 42

3.2.1 Estruturação do sistema 42

3.2.1.1 Diagrama de banco de dados 43

3.2.1.2 Diagrama de processos 44

3.2.2 Sistema desenvolvido 47

3.2.2.1 Gestor 49

3.2.2.2 Administrador 54

3.2.2.3 Professor 57

3.2.2.4 Estórias dos alunos 68

3.2.2.5 Testes 73

3.3 75

4 Cronograma 76

BIBLIOGRAFIA 77

Apendice A - carta de pedido de permissão para uso de informações da escola International language center 80

# INTRODUÇÃO

A tecnologia tem influenciado e modificado a maneira em que se ensina. Com o advento de uma geração conectada, em que crianças e jovens tem acesso cada vez mais cedo a tecnologias, faz com que se estabeleça novas possibilidades de se adquirir conhecimento, por um outro meio diferente ao convencional. Porém aliar tecnologia ao ensino pode não ser algo tão fácil, como afirma Bacich *et al.* (2015, p. 29), em dizer que “A integração das tecnologias digitais na educação precisa ser feita de modo criativo e crítico, buscando desenvolver a autonomia e a reflexão dos seus envolvidos, para que não sejam apenas receptores de informações”. Portanto a tecnologia pode ser utilizada em prol da educação, mas para que trabalhem juntos deve-se ter um cuidado para que os utilizadores da tecnologia passem a interagir de maneira a não somente receber informações, mas também colaborar e aprender através do uso (BACICH *et al.*, 2015).

Já o estudo de línguas estrangeiras é disseminado globalmente, sendo requisitado em vários aspectos profissionais, educacionais, de pesquisa e de interação das pessoas nos diferentes países. No estudo de idiomas o atrelamento a tecnologias não foi diferente. Surgiram sistemas (de escolas físicas ou não) com o intuito de auxiliar nos processos de aprendizagem de uma língua. Buscou-se contemplar todas as etapas de aprendizado de um idioma, desde a escrita, leitura, até o entendimento e fala de uma língua. Diversas escolas se prontificam a ensinar os idiomas com diferentes métodos de ensino, material e apoio informatizados, porém, há escolas que não contam com apoio computacional para auxiliar no processo de ensino e aprendizagem.

Pensando nisso foi identificada uma escola que não possuía esse apoio informatizado. A escola identificada é a *International Language Center* (ILC) que é situada em Montes Claros, Minas Gerais. A ILC foi escolhida pelo fato de não dispor de sistema de auxílio no ensino/aprendizagem, detinha somente de um *site* desenvolvido pela Empresa Júnior do curso de Sistemas de Informação da Universidade Estadual de Montes Claros (Unimontes), a INFOBITS, contendo informações básicas da escola, sendo uma página estática, não tendo recursos de auxílio ao ensino de idiomas. E, portanto, através deste trabalho, buscou-se desenvolver um ambiente *web* para sanar a falta desse apoio informatizado e apoiar professores e alunos. Nesse ambiente é possível ao professor auxiliar seus alunos, alunos esses que serão divididos por suas turmas. Seja no compartilhamento de materiais ou buscando sanar dúvidas sobre o conteúdo. Para os alunos espera-se que haja envolvimento e aprendizado quanto aos conteúdos disponibilizados no ambiente virtual. Acredita-se que com o ambiente, a interação aluno-professor e aluno-turma transpassará da sala de aula, possibilitando interação de aprendizagem de maneira informatizada.

Ainda para o desenvolvimento deste ambiente usou-se de uma metodologia de desenvolvimento ágil, a metodologia *eXtremme Programming* (XP) para apoiar e agilizar o processo de desenvolvimento do sistema.

Portanto o tema deste trabalho pode ser descrito como o desenvolvimento web com técnicas ágeis, por aliar o desenvolvimento de um ambiente web utilizando-se da metodologia ágil XP. Então busca-se resolver o seguinte problema, de entender como o XP pode apoiar no processo de desenvolvimento de um ambiente web para ensino e aprendizagem de inglês. O objetivo geral deste trabalho é então desenvolver ambiente *web* com técnicas ágeis para auxiliar no processo de ensino e aprendizagem de língua inglesa para a escola *International Language Center* (ILC). E como objetivos específicos disponibilizar materiais e exercícios para os alunos no ambiente desenvolvido; apresentar um calendário com datas de exercícios, provas e eventos para os alunos e identificar conteúdos de maior deficiência a partir de gráficos de desempenho.

# Referencial teórico

“O referencial teórico é o alicerce para o desenvolvimento de uma monografia.” (DIAS; SILVA, 2010, p. 31). Assim pode-se entender o quão importante é descrever aquilo que servirá de base para a concepção de um trabalho científico. Severino (2002) define essa etapa como sendo a busca por documentos que apresentem conteúdo relevante e que fazem sentido ao tema discutido no trabalho científico. Essa busca e escrita de conteúdo serve para que o leitor tenha uma visão inicial do que será abordado e tenha base para compreender os conceitos que são abordados no decorrer do trabalho científico.

## Educação a distância – ambiente virtual

A maneira de se ensinar mudou adequando-se as necessidades e mudanças (sociais, tecnológicas, etc.). Um exemplo disso é o da Educação a Distância (EAD) que surgiu no Brasil por volta de 1900, por meio de cursos oferecidos por correspondência, a fim de possibilitar mais conveniência no estudo (ALVES *et al.*, 2009).

Com o alcance da internet e a maior acessibilidade a computadores e *smartphones* a EAD se propagou pelo território nacional (ALVES *et al.*, 2009). E tem sido de grande importância tendo em vista que pessoas em localidades sem acesso direto ao ensino presencial, podem estudar e alcançar uma formação (MELO NETO *et al.*, 2012).

Nas seções subsequentes são descritos alguns exemplos de sistemas que utilizam da EAD e de processos informatizados para possibilitar e automatizar o processo de ensino.

### Metodologias/sistemas de apoio de ensino de idiomas

A tecnologia influencia e molda a maneira em que o ensino é feito (ALVES *et al.*, 2009). Isso não é diferente para o ensino de idiomas, uma vez que as escolas têm buscado se adequar e inserir tecnologias nos seus processos de ensino, e soluções independentes tem surgido para auxiliar no aprendizado de línguas. Essas escolas e tecnologias são descritas nos próximos parágrafos, apresentando suas principais características.

Como exemplo de escola que tem se moldado a se adequar a novas tecnologias toma-se a *Wizard by Pearson*, que é uma escola de idiomas internacional (possui sedes em outros países além do Brasil) (WIZARD, 2017b). A instituição conta com uma metodologia de ensino que alia a clássica abordagem de sala de aula (chamada pela *Wizard* de *Connections*) ao uso de tecnologias de apoio informatizado (chamada de *Interactive*). Na *connections* os alunos vão a sala de aula e interagem com outros alunos e professores, e são guiados pelo professor. Já no *interactive* o aluno recebe apoio informatizado por meio de um *tablet* denominado Wiz.tab. Para auxiliar a pronúncia dos alunos dispõe-se de uma caneta (denominada Wiz.pen), que faz a leitura de palavras, frases e expressões contidas no material do aluno (WIZARD, 2017a). A Figura 1 demonstra o modelo de aprendizagem da Wizard com as características anteriormente descritas.

Figura 1 - Modelo de aprendizagem da Wizard



Fonte: WIZARD, 2017a, p.3.

A escola ainda conta com uma aplicação *mobile* e um ambiente *Web* denominados Wiz.me, que servem de apoio aos alunos quando estão fora da sala, auxiliando-os na pronúncia e escrita (WIZARD, 2017a). A Figura 2 descreve algumas funcionalidades da versão móvel dessa aplicação.

Figura 2 - Funcionalidades do Wiz.me



Fonte: WIZARD, 2017a, p.4.

Outra escola que interliga tecnologia e ensino de idiomas é a escola Centro de Cultura Anglo Americana (CCAA). A CCAA detém um espaço virtual denominado espaço CCAA aluno (CCAA, sd.). Nele o aluno da escola terá acesso a conteúdo para *tablet*, como textos, áudios e vídeos (CCAA, sd.). Outro recurso disponível é o *Computer Assisted Language Learning* (CALL) que se dá por um “Software educacional que permite que o aluno realize seus exercícios escritos utilizando o computador. É uma forma rápida, fácil, interativa e agradável de fixar o conteúdo aprendido em sala de aula” (CCAA, sd., p. 1). O espaço virtual ainda conta com atividades complementares para auxílio de escrita, pronúncia e leitura.

Já em questão de aplicações independentes para auxílio de idiomas tem-se o exemplo do Babbel, que é uma aplicação disponível para dispositivos móveis e *Web*. Seu objetivo é o auxílio da compreensão, escrita e fala de diversos idiomas como inglês, português, espanhol, alemão, holandês, entre outros. Contempla diversos níveis de conhecimento, indo do básico ao avançado. Apesar da aplicação ser gratuita, somente alguns níveis estão disponíveis de forma gratuita, sendo necessário assim o pagamento de planos para a liberação de todo o conteúdo disponível (BABBEL, 2018) conforme descrito na Figura 3.

Figura 3 - Preços do Babel



Fonte: BABBEL, 2018, p.1.

Outra aplicação semelhante ao Babbel é o Duolingo que possibilita o aprendizado de idiomas de maneira gratuita. É possível o estudo de línguas como inglês, espanhol, francês, alemão, dentre outras. Apresenta-se alguns conceitos de gamificação, como o avanço de níveis e as vidas (corações). Como nos *games*, ao fazer algo de errado ocorre penalização ou eliminação de uma vida, no Duolingo ao errar uma resposta de uma questão um coração é perdido (DUOLINGO, sd.). A Figura 4 ilustra algumas características da aplicação Duolingo.

Figura 4 - Características do Duolingo



Fonte: DUOLINGO, p.1.

O ambiente proposto deseja unir algumas características dos sistemas citados, como a possibilidade de encontrar conteúdos de auxílio (como no espaço do CCAA) e acesso a testes de escrita e escuta como no Duolingo. Acredita-se que ao aliar essas funcionalidades já utilizadas em outros sistemas à novas funcionalidades, possa-se criar um ambiente mais adequado para apoiar o ensino de língua inglesa na ILC.

## Desenvolvimento e tecnologias de sistemas Web

Para entender-se o que é desenvolvimento pode-se tomar o contexto geral da palavra, que tem por significado ser uma “Série de etapas, acontecimentos, ações, etc. que levam ao surgimento de algo, ou à manifestação em todos os seus aspectos[...]” (FERREIRA, 2001, p. 240). Tomando isso para o meio computacional diz-se então que o desenvolvimento de um sistema se dá pela passagem por todas as etapas da sua concepção até o momento em que o sistema apresente tudo aquilo que lhe foi previsto (PRESSMAN, 2011).

Hirama (2011, p. 3) lista os tipos de atividades presentes no processo de desenvolvimento de sistemas, que são as “[...] atividades técnicas de Engenharia de Sistemas, análise, projeto, codificação e testes”. Essas atividades devem estar presentes e são responsáveis por contemplar o processo de desenvolvimento.

Já para o quesito *Web*, os sistemas desenvolvidos são conhecidos como *WebApps* (PRESSMAN, 2011). Esses sistemas têm por característica ser acessados e utilizados por meio da rede. Pressman (2011) cita alguns atributos que estão presentes na grande maioria dos *WebApps* que são: o uso intensivo de redes, simultaneidade, carga não previsível, desempenho, disponibilidade, orientadas a dados, sensibilidade no conteúdo, evolução contínua, imediatismo, segurança e estética. Portanto, para se desenvolver soluções *web* deve-se levar em conta os atributos apresentados por Pressman (2011).

### Criptografia e controle de acessos

Criptografia pode ser entendido como o conjunto de métodos e técnicas capazes de modificar um texto legível transformando-o em um texto não legível. Isso é possível por meio de um algoritmo, que codifica a mensagem baseado em algum padrão específico. A recuperação das informações originais se dá pelo processo inverso ao da criptografia (MORENO; PEREIRA; CHIARAMONTE, 2005). A Figura 5demonstra esse processo de transformação de uma mensagem e o seu retorno a mensagem original.

Figura 5 - Esquema geral para criptografia de um texto



Fonte: MORENO; PEREIRA; CHIARAMONTE, 2005, p.21.

Moreno, Pereira e Chiaramonte (2005) afirmam haver dois meios de se criptografar uma mensagem, por meio de códigos ou por meio de cifras.

Os códigos protegem as informações trocando partes destas por códigos predefinidos. Todas as pessoas autorizadas a ter acesso a uma determinada informação devem conhecer os códigos utilizados. As cifras são técnicas nas quais a informação é cifrada por meio da transposição e/ou substituição das letras da mensagem original. Assim, as pessoas autorizadas podem ter acesso às informações originais conhecendo o processo de cifragem (MORENO; PEREIRA; CHIARAMONTE, 2005, p. 21).

Existem algoritmos que implementam processos de criptografia, gerando um *hash* contendo uma quantidade predeterminada de caracteres a partir de um determinado conteúdo. Cada algoritmo vem aliado a uma chave, que é um valor alterável dentro do algoritmo e que pode ser modificado para se adequar as necessidades do utilizador. Isso implica que mesmo que se conheça o funcionamento do algoritmo de criptografia uma pessoa só seria capaz de reproduzir o mesmo resultado de outra se ambas usassem a mesma chave no processo de criptografia (HINZ, 2000).

O processo de criptografia é utilizado em auxílio a segurança de senhas de usuário. Os dados sensíveis advindos do usuário, como no caso da senha, somente são salvos na base de dados após serem submetidos a um processo de criptografia, utilizando-se de algum algoritmo em específico (PHP, 2018a).

Já o controle de acessos pode ser entendido como a maneira a qual se controla o que cada usuário tem acesso em um sistema, privando o usuário o que não lhe for autorizado, e permitindo somente o que deve se ter acesso de fato (ZAPATER; SUZUKI, 2005). Comumente se utiliza o modelo de controle de acessos baseado em papeis (*Role-Based Access Control*) (RBAC). Esse modelo utiliza a premissa que cada usuário cumpre um ou mais papeis, como por exemplo, um professor, ou um aluno. E a esses papeis são atribuídas permissões. Essas permissões indicarão as funcionalidades que o usuário pode ter acesso no sistema. Por exemplo, um determinado usuário apresenta o papel de professor, e como professor ele pode lançar notas e faltas aos alunos. O sistema deve ser capaz de identificar o papel do usuário e permitir somente o que aquele papel contiver como permissão (SANDHU, 1998).

O *Framework* Laravel, que é discutido na subseção 5.2.5.5, contém avançados algoritmos de criptografia (OTWELL, 2018). Assim as senhas dos usuários serão submetidas a um processo de segurança. Além disso o Laravel apoia o controle de acessos por meio de papeis como descrito anteriormente, para garantir que cada usuário só acesse o que lhe for permitido. Pensa-se que com a utilização de tais mecanismos de segurança possa se conseguir uma melhora na segurança dos dados sensíveis dos usuários e dos acessos no ambiente.

### Interação humano computador (IHC)

*Softwares* são necessários em praticamente todas as áreas de atuação presentes no mercado. Porém não há uma solução computacional que resolva todos os problemas de todas as áreas. Então para cada problema surge uma maneira de implementar uma solução que seja utilizada por um nicho de usuários (BARANAUSKAS; ROCHA, 2003). Portanto a maneira que os sistemas interagem com os usuários tendem a ser diferentes em cada solução.

Para buscar compreender e melhorar a maneira que usuários se interagem com sistemas surgiu o campo de estudo denominado Interação Humano-Computador (IHC) (BARANAUSKAS; ROCHA, 2003). Baranauskas e Rocha (2003) afirmam que a IHC contempla todo o processo de interação do usuário com o sistema e não somente o processo de *design* das interfaces. Então entende-se como Interação Humano-Computador a: “[...] disciplina preocupada com o design, avaliação e implementação de sistemas computacionais interativos para uso humano e com o estudo dos principais fenômenos ao redor deles.” (BARANAUSKAS; ROCHA, 2003, p. 14), e a Figura 6 apresenta algumas informações sobre IHC.

Figura 6 - Interação humano-computador adaptada da descrição do comitê SIGCHI 1992



Fonte: BARANAUSKAS; ROCHA, 2003, p.15.

É importante que se conheça e se aprimore os processos de interação do usuário para que as pessoas consigam efetuar suas funções corretamente em um sistema, uma vez que se compreende o que cada interface significa e o que deve ser feito em cada uma delas.

### Engenharia de Software

O processo de desenvolvimento de *software* pode ser feito de maneiras diversas. Portanto buscou-se padronizar o processo de desenvolvimento para que se minimizasse os custos e problemas de produção e com uma melhor qualidade final do *software*, surgindo assim a Engenharia de *Software* (PRESSMAN, 2011). Hirama (2011, p. 7) apresenta o contexto histórico em que a definição do conceito de Engenharia de *Software* surgiu:

O conceito “Engenharia de Software” foi cunhado em 1969 por Fritz Bauer em uma conferência patrocinada por um Comitê de Ciência da Organização do Tratado do Atlântico Norte (Otan), no momento em que a chamada crise do software precisava de uma solução para a demanda crescente por software dentro de custo e prazo adequados.

Para entender-se o conceito do que é engenharia de *software* toma-se os significados de engenharia e *software*. A definição de engenharia se dá pela: “Aplicação de conceitos científicos e empíricos, e certas habilitações específicas, à criação de estruturas, dispositivos e processos para converter recursos naturais em formas adequadas ao atendimento das necessidades humanas.” (FERREIRA, 2001, p. 289). Já a palavra *software* tem as seguintes definições segundo Ferreira (2001, p. 682):

1. Em um sistema computacional, o conjunto de componentes informacionais, que não faz parte do equipamento físico e inclui os programas e os dados a eles associados.
2. Qualquer programa ou conjunto de programas de computador.

A partir dos significados pode-se deduzir que engenharia de *software* se trata então da aplicação de conceitos e métodos para apoio ao processo de desenvolvimento de um sistema computacional. Sommerville(2011, p. 5) reforça essa afirmação dizendo que: “Engenharia de software é uma disciplina de engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado”. O *Institute of Eletrical and Eletronics Engineers* (IEEE)(1990, p. 67) porém define como Engenharia de software:

1. A aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, a operação e manutenção do software;
2. O estudo de abordagens assim como descrito na etapa 1.

Pressman (2011) caracteriza a engenharia de *software* como sendo em camadas, conforme descrito na Figura 7, sendo as camadas: Ferramentas, métodos, processo e foco na qualidade.

Figura 7 - Camadas da Engenharia de Software



Fonte: PRESSMAN, 2011, p.39.

Quanto ao foco na qualidade, Pressman (2011), afirma que todas as disciplinas ligadas a engenharias devem ter foco na qualidade. Para isso conta-se com modelos de qualidade que apoiam a qualidade tanto no processo quanto no produto final.

Já em relação ao processo, Pressman (2011) ressalta que existem alguns tipos. Um deles é o de engenharia de *software*, que é a “[...] liga que mantém as camadas de tecnologia coesas e possibilita o desenvolvimento de software de forma racional e dentro do prazo.” (PRESSMAN, 2011, p. 39). O outro é o processo de *software* que:

[...] constitui a base para o controle do gerenciamento de projetos de software e estabelece o contexto no qual são aplicados métodos técnicos, são produzidos produtos derivados (modelos, documentos, dados, relatórios, formulários etc), são estabelecidos marcos, a qualidade é garantida e mudanças são geridas de forma apropriada. (PRESSMAN, 2011, p. 40)

Os métodos por sua vez apresentam informações de caráter técnico para auxiliar no processo de desenvolvimento do *software* (PRESSMAN, 2011). “Os métodos envolvem uma ampla gama de tarefas, que incluem: comunicação, análise de requisitos, modelagem de projeto, construção de programa, testes e suporte.” (PRESSMAN, 2011, p. 40).

Já as ferramentas compreendem como o apoio automatizado aos métodos e ao processo (PRESSMAN, 2011). São por exemplos: *softwares* para modelagem, *Integrated Development Environments* (IDES), etc.

#### Modelagem de processos com o *Business Process Model and Notation* (BPMN)

Um processo pode ser entendido como uma sequência ordenada de atividades, praticadas por um ou mais atores, com propósito de se alcançar um objetivo em específico. Então, todo processo oferece algo como resultado da completitude de todas as atividades (CAMPOS, 2014). Um exemplo de processo pode ser visto na Figura 8 que representa um processo cujo objetivo seja a conclusão de um determinado curso (CAMPOS, 2014).

Figura 8 - Exemplo de processo



Fonte: CAMPOS, 2014, p.18

No mundo dos negócios as empresas são compostas por uma série de processos, sejam eles internos ou externos. Então se faz necessário a compreensão quanto ao funcionamento desses processos (CAMPOS, 2014). Porém um processo de negócio poderia ser representado de várias maneiras, e uma delas, por exemplo, poderia ser a maneira descrita na Figura 8.

Buscando a padronização da notação de modelagem dos processos de negócios criou-se o *Business Process Model and Notation* (BPMN) que se trata de um padrão de notação para modelagem de processos. O BPMN auxilia nessa padronização de notação pois fornece elementos gráficos para a modelagem de fluxo de um determinado processo. O BPMN indica por meio desses elementos gráficos quais são as atividades que compõem um processo e quem são os atores e executores de cada atividade (SILVER, 2017).

Campos (2014, p. 51) define a notação BPMN como “[...] rica em elementos de representação”. Dentre esses elementos de representação, os considerados básicos a toda modelagem são os conectores, *gateways*, eventos, *data objects*, *pools* e *lanes* (CAMPOS, 2014)*.* Cada um deles será descrito nos parágrafos subsequentes, identificando os seus significados e representações visuais.

Os conectores representam a conexão de atividades dentro de um processo. Representam a ligação e por conseguinte sequência de atividades do processo. A representação visual dos conectores se dá por linhas indicativas (setas) conforme visto na Figura 9, que relata um processo de compra de um determinado item (CAMPOS, 2014).

Figura 9 – Exemplo de conectores em um processo de compra



Fonte: CAMPOS, 2014, p. 53.

Já o *gateway* se trata de uma validação necessária que irá determinar o curso do fluxo do processo, alterando ou não o seu fluxo natural. A notação utilizada para representar o *gateway* é a figura de um losango, conforme visto na Figura 10. A Figura 10 relata um processo de compra, porém na atividade de receber, ocorre uma validação de constatação de conformidade com o que foi solicitado, havendo conformidade, paga-se, não havendo retorna-se a atividade anterior (CAMPOS, 2014).

Figura 10 - Exemplo de *gateway* em um processo de compra



Fonte: CAMPOS, 2014, p. 54.

Os eventos para o BPMN servem de indicação de ações externas ao processo como por exemplo indicar o início e o término de um processo (CAMPOS, 2014). A representação gráfica dos eventos é feita por meio de círculos. Os círculos com a borda espessa indicam o fim de um processo. Enquanto os círculos que tem bordas duplas significam eventos intermediários de um processo. Já os círculos com borda menos espessa indicam o início do processo. Um exemplo de utilização dos eventos descritos é o visto na Figura 11.

Figura 11 - Exemplo de utilização de eventos em um processo de compra



Fonte: CAMPOS, 2014, p. 56.

*Data objects* por sua vez, representam dados e informações que são usadas e criadas ao longo de um processo (CAMPOS, 2014). Esses dados e informações servem para utilização na atividade subsequente a atividade que criou a informação. É representado graficamente por uma folha de papel com a ponta superior dobrada, ligada por linhas seccionadas em formato de seta, que ligam a atividade que gerou a informação a atividade que irá consumir a informação (CAMPOS, 2014). Exemplos de *data objects* são vistos na Figura 12, como o pedido de compra que é gerado ao comprar um produto.

Figura 12 - Exemplo da utilização de *data objects* em um processo de compra



Fonte: CAMPOS, 2014, p. 58.

As *Pools*, ou piscinas em português, representam um processo como um todo, englobando todas as atividades de um processo (CAMPOS, 2014). São representados graficamente por um retângulo contendo o nome do processo.

*Lanes*, ou raias em português, representam os atores participantes de um processo, sendo esses atores pessoas, departamentos, setores, cargos, etc. (CAMPOS, 2014). As raias são representadas por um retângulo contendo o nome do ator. As raias e as demais atividades do processo devem estar englobadas dentro de uma piscina, conforme visto na Figura 13.

Figura 13 - Exemplo de utilização de piscinas e raias em um processo de compra



Fonte: CAMPOS, 2014, p. 59.

Acredita-se que a utilização de modelagem de processos neste trabalho servirá de documentação ao desenvolvedor e ao cliente, ajudando-os a compreender um determinado processo.

#### Metodologia Ágil

Metodologia ágil se trata de um movimento que buscou mudar a maneira como os projetos de *software* eram executados. Apresentou-se como premissa o aumento na velocidade de desenvolvimento de sistemas sem que houvesse uma eventual perda de qualidade. E buscou-se diferenciar das metodologias tradicionais como o modelo em cascata. Uma das diferenças se dá pela filosofia iterativa (ou espiral) e não mais linear como no modelo em cascata (HIRAMA, 2011). Esse modo iterativo prega o desenvolvimento de *software* como algo constante. A cada iteração o cliente recebe uma parte do sistema podendo opinar sobre melhorias ou eventuais erros de modelagem (TELES, 2014). A cada iteração então recebe-se do cliente o seu *feedback* acerca do que foi implementado, além do cliente ter a possibilidade de utilizar o sistema sem que esse esteja completamente terminado. Ao final de todas as iterações tem-se o sistema completo e operante, uma vez que o sistema foi validado ao fim de cada iteração (HIRAMA, 2011). A Figura 15 demonstra um exemplo de representação do modelo iterativo ou espiral.

Figura 14 - Modelo em espiral



Fonte: HIRAMA, 2011, p39.

O XP, que será descrito suas características e peculiaridades na subseção 5.2.3.5, e utilizado no decorrer deste trabalho, é um exemplo de uma metodologia ágil.

#### *Extreme Programming* (XP)

O *Extreme Programming* (XP), se trata de uma metodologia de desenvolvimento ágil, que busca aliar agilidade no desenvolvimento, com qualidade no produto final. O XP pode então, ser descrito como “[...] um processo de desenvolvimento que busca garantir que o cliente receba o máximo de valor de cada dia de trabalho da equipe de desenvolvimento” (TELES, 2014, p. 24). Mas ao contrário do modelo em cascata, o XP é uma metodologia que apoia as mudanças. O XP entende que o cliente pode não ter certeza sobre o que quer logo de início, mas que vai descobrindo assim que tem contato com um protótipo do *software*. Assim, ao ter contato direto com o *software*, o cliente é capaz de verificar o que deve ser melhorado ou mudado. Além disso os desenvolvedores podem identificar ainda o que o cliente considera de mais importante. Portanto é de suma importância ao XP o *feedback* do cliente acerca do que foi implementado (TELES, 2014).

O *feedback* é um dos quatro valores fundamentais do XP, junto com a comunicação, a simplicidade e a coragem. Portanto o *feedback* diz respeito a capacidade de se receber informação de volta do cliente. E através dessas informações ser capaz de entender as especificações e requisitos necessários para o desenvolvimento, conhecendo assim as necessidades e anseios do cliente, em vista que o cliente estará em contato direto com o *software* (TELES, 2014).

A comunicação diz respeito a equipe constantemente estar em contato direto com o cliente, pois estabelecendo contato frente a frente, a equipe pode retirar potenciais dúvidas ou conhecer certos comportamentos requeridos no *software* (TELES, 2014). A diferença de se ter um contato frente a frente, para uma documentação escrita, é que a documentação não exprime emoções ou sentimentos, já na comunicação direta a linguagem corporal toma forma, e o sentimento é expressado de maneira mais eficaz como afirma Teles (2014).

A simplicidade consiste em desenvolver código o mais simples possível, somente resolvendo o necessário para o cliente (TELES, 2014). Teles (2014) explica que isso se dá, pois, um código mais simples tende a ser mais fácil de ser alterado em caso de necessidade de alteração.

A coragem diz respeito a equipe ter coragem de enfrentar algo novo. Para desenvolver usando o XP alguns padrões devem ser deixados de lado (TELES, 2014). Por exemplo, o XP prega que o desenvolvimento deve ser feito em par, ou seja, dois desenvolvedores numa mesma máquina, desenvolvendo o mesmo código. Porém pode-se acreditar que isso seja algo custoso ou dispendioso, mas Teles (2014) afirma que se deve ter coragem para esquecer-se de premissas tradicionais para acreditar e assumir o uso dessa metodologia.

Quanto a programação em par citada anteriormente, ela é uma das práticas do XP, junto com o jogo do planejamento, o cliente presente, o *stand up meeting*, desenvolvimento guiado a testes, *refactoring*, código coletivo, código padronizado, *design* simples, metáfora, ritmo sustentável, integração contínua e *releases* curtos. A programação em par é característica que levanta mais dúvida quanto ao uso do XP. Essa dúvida ocorre pelo fato de que se pensa que com a utilização de dois programadores fazendo um único trabalho, está se desperdiçando um programador que poderia estar desenvolvendo em outra máquina. Porém ao se utilizar desse desenvolvimento em par espera-se que a qualidade final do código seja melhor, pois enquanto um programa o outro verifica e valida o código. Além disso pode-se aliar duas soluções distintas para um problema, cada um com uma experiência e maneira de solucionar algo, com o intuito de encontrar uma solução considerada melhor (TELES, 2014).

Além da programação em par, destaca-se o desenvolvimento orientado a testes, que se trata de se testar primeiro antes de codificar de fato (HIRAMA, 2011). Pensa-se que ao se produzir o teste primeiro o desenvolvedor consiga compreender melhor o que será desenvolvido (TELES, 2014). Teles (2014) defende que o ato de testar deve ser uma prática constante no processo de desenvolvimento, pois assim assume-se que os erros encontrados através dos testes serão corrigidos imediatamente ou que pelos testes possa também haver a validação do que foi requisitado.

Outro fator que se destaca no XP são as estórias de usuários que são pequenos trechos escritos em cartões ao qual o cliente descreve como quer que seja uma determinada funcionalidade do sistema (TELES, 2014). Toda estória de usuário se transforma em um requisito e, por conseguinte uma funcionalidade da aplicação desenvolvida. Um exemplo de estória de usuário é o presente na Figura 16.

Figura 15 - Exemplo de uma estória de usuário



Fonte: TELES, 2014, p.78.

Como o XP se trata de uma metodologia iterativa, em certos períodos de tempo, tem-se entregas de porções do *software* para que que o cliente possa utilizar o sistema sem que o mesmo esteja totalmente pronto. Essas entregas são conhecidas como *releases* (TELES, 2014). Segundo Teles (2014) os *releases* no XP duram cerca de 2 meses, e por isso são considerados *releases* curtos, já que o intervalo entre as entregas é pequeno. Após o término de um *release* faz-se uma nova entrevista com o cliente e coleta as estórias que serão desenvolvidas no novo *release* (TELES, 2014). Cada *release* é composto por iterações, uma iteração pode ser entendida como “[...] um pequeno espaço de tempo dedicado para a implementação de um conjunto de estórias” (TELES, 2014, p. 87). Segundo Teles (2014) o tempo de uma iteração pode variar cerca de uma a três semanas, mas que geralmente o tempo de duas semanas é o mais utilizado em empresas com o XP.

Pretende-se neste trabalho fazer utilização da metodologia de desenvolvimento denominada XP. Hirama (2011) afirma que o XP pode não ser recomendado a grandes projetos de *software*. Porém pretende-se fazer utilização desta metodologia a ponto de se chegar a uma avaliação pessoal acerca da experiência de uso do XP e assim verificar como a metodologia conseguiu apoiar o desenvolvimento do ambiente. Deve-se ressaltar que não será possível a aplicação da programação em par, que é uma das práticas do XP, pois a execução do projeto será feita por uma pessoa.

### Tecnologias para desenvolvimento WEB

O desenvolvimento de soluções *web* envolve uma série de tecnologias. Cada uma com uma determinada função e utilidade. As tecnologias Web podem ser divididas em *Front-End* e *Back-End*. Tecnologias *Front-End* são aquelas que estão em contato direto com o usuário, como por exemplo, o *Hypertext Markup Language* (HTML) (ROBBINS, 2013). Já as tecnologias *Back-End* são as que estão em contato direto com o servidor e não mantém vínculo direto com o usuário final, um exemplo é a linguagem de *scripting* PHP: *Hypertext Preprocessor* (PHP) (PHP, 2018b). Todas essas tecnologias trabalham em conjunto para oferecer uma melhor experiência de uso para o usuário (ROBBINS, 2013). Algumas dessas tecnologias são abordadas nas seções seguintes deste trabalho.

#### *Hyper Text Markup Language* (HTML)

*Hyper Text Markup Language* (HTML) se trata de uma linguagem de marcação capaz de transformar documentos de texto em páginas *web*. Seu objetivo é prover uma descrição semântica do conteúdo e formar uma estrutura para o documento. A delimitação de conteúdo no HTML é feita por meio de *tags* que vão indicar onde um determinado conteúdo começa e termina (ROBBINS, 2013). A Figura 19 consiste na estrutura de *tags* básica de um documento HTML.

A *tag* ‘html’ indica onde se inicia e onde se termina o documento HTML. A *tag* ‘*head*’ representa o cabeçalho do documento, dentro dela há uma outra *tag* chamada ‘*title*’ que indica o título da página. Já a ‘*body’* representa o corpo do documento HTML (ROBBINS, 2013).

Figura 16 - Estrutura básica do HTML



Fonte: PRÓPRIA, utilizando o SublimeText 4.

O HTML está na sua versão 5, e essa versão trouxe uma série de mudanças acerca das *tags* a fim de deixar o documento HTML mais semântico. Com essas mudanças deixou-se a leitura de documentos e páginas mais fácil por caracterizar melhor os elementos da página (ROBBINS, 2013).

Por acreditar que o HTML seja uma maneira simples e concisa de marcação, além do Laravel reconhecer HTML, optou-se então pelo seu uso no desenvolvimento deste trabalho.

#### *Cascading Style Sheets* (CSS)

*Cascading Style Sheets,* com acrônimo (CSS), é uma linguagem criada com o objetivo de cuidar do *layout* e do estilo de uma página (CAELUM, 2018). Além de estilização dos elementos, com o CSS também é possível acrescentar efeitos nos elementos como desaparecer da tela por exemplo (MCFARLAND, 2013). E apesar de o CSS colaborar com o HTML, ele não se trata de HTML. Enquanto o HTML provê uma organização estrutural de um documento, o CSS trabalha em colaboração com o *browser* com a finalidade de fazer com que uma página HTML contenha estilo próprio (MCFARLAND, 2013). Um exemplo da sintaxe do CSS é o descrito na Figura 20.

Figura 17 - Sintaxe CSS



Fonte: CAELUM, 2018, p.21.

Quanto a inclusão do CSS em elementos HTML, ela pode ser feita de três maneiras. A primeira delas é inserindo o código CSS diretamente na *tag* HTML do elemento, utilizando-se do atributo ‘*style*’ da *tag* em questão conforme descrito na Figura 21.

Figura 18 - CSS inserido diretamente na tag HTML



Fonte: CAELUM, 2018, p.21.

Outra maneira de se inserir o CSS é por meio de uma *tag* HTML denominada ‘*style*’, onde o seletor do elemento deve ser referenciado como na Figura 22. O seletor pode ser entendido como uma referência aos elementos de uma página. As principais maneiras de se referenciar elementos são pelo atributo id (identificado com o símbolo #), pelo atributo *class* (identificado com o ponto final), ou pela *tag* de um elemento. Na sintaxe CSS dentro da *tag* ‘*style’* deve-se indicar o seletor, e no caso da Figura 22 o seletor é uma *tag* de parágrafo (p), e o que vier dentro das chaves são seus estilos (CAELUM, 2018).

Figura 19 - CSS inserido através da *tag style*



Fonte: CAELUM, 2018, p.22.

A terceira maneira de se estilizar os elementos HTML se dá por meio de um arquivo separado como visto na Figura 23, de maneira que o código CSS fique separado da codificação feita em HTML (CAELUM, 2018).

Figura 20 - CSS contido no arquivo estilos.css



Fonte: CAELUM, 2018, p.23.

A sintaxe CSS segue o modelo descrito pela Figura 22. A única diferença é que para utilizar a estilização contida no arquivo CSS no documento HTML deve-se utilizar no código HTML uma *tag* ‘*link*’. A *tag* ‘*link’* é responsável por carregar e possibilitar o uso dos estilos a partir de um arquivo externo. Para indicar a localização do arquivo CSS, deve-se utilizar um atributo da *tag* ‘*link’* chamado ‘href’, e nesse atributo indicar o caminho até o arquivo de estilos, conforme demonstrado na Figura 24. O estilo criado na Figura 23 é utilizado na *tag* <p> do documento HTML da Figura 24 (CAELUM, 2018).

Figura 21 - CSS inserido através de um arquivo externo



Fonte: CAELUM, 2018, p.22.

Com o uso do CSS no ambiente proposto espera-se contemplar os usuários com páginas mais bonitas e agradáveis para que a experiência de uso seja a melhor possível.

#### JavaScript (JS)

O JavaScript (JS) se trata de uma linguagem de *scripting*,que tem por característica ser uma linguagem interpretada, ou seja, não depende de compilação para sua execução (CAELUM, 2018). Além disso se trata de uma linguagem de tipagem dinâmica, ou seja, as variáveis não estão diretamente ligadas a somente um tipo, como por exemplo int, *double*, mas variam de acordo com o dado inserido na variável (CROCKFORD, 2008).

O JavaScript é comumente utilizado em *web* browsers, e os *web* browsers que têm o papel de interpretar o código em JavaScript e gerar uma saída. O intuito da linguagem é prover dinamicidade aos conteúdos, uma vez que o HTML e o CSS provêm conteúdo estático a um documento *web* (CAELUM, 2018). Para se utilizar o JavaScript em páginas HTML é necessário que se faça uso de uma *tag* específica chamada *script* conforme descrito na Figura 25 e na Figura 26. É possível ainda escrever os *scripts* de duas maneiras. A primeira delas é inserindo o *script* diretamente na página HTML (Figura 25).

Figura 22 - Exemplo de uso do JavaScript diretamente no HTML



Fonte: PRÓPRIA, utilizando o SublimeText 4.

A outra maneira de se utilizar o JavaScript em páginas HTML é escrevendo o *script* em um arquivo separado e anexá-lo ao documento HTML como visto na Figura 26. Acredita-se que assim possa separar melhor HTML de JavaScript e deixando mais coesa a leitura de um documento HTML (CAELUM, 2018). Porém em algumas ocasiões é necessário inserir o *script* diretamente no documento HTML.

Figura 23 - Exemplo de uso do JavaScript por meio de um arquivo externo



Fonte: PRÓPRIA, utilizando o SublimeText 4.

Portanto, visando alcançar dinamicidade nos documentos HTML, optou-se pela utilização do JavaScript. Pensa-se que assim a experiência final do usuário com o sistema possa ser mais dinâmica.

#### TypeScript

O TypeScript, ou TS, é um “superconjunto” ou “*superset*” em inglês da linguagem JavaScript. Isso significa que o TS serve para agregar funcionalidades e melhorias a linguagem JavaScript como o exemplo a criação de classes, objetos e atributos das classes como herança e polimorfismo. Porém o que se destaca mais no TypeScript em diferença ao JavaScript é a tipagem dos dados, onde em um Script TypeScript os dados devem ser tipados para facilitar a leitura e compreensão do código, além de evitar que uma variável receba um tipo de dado não esperado (ABREU, 2017).

O trecho de código da figura 27 se trata de um exemplo de Script TS. Nota-se a tipagem da variável modelo definindo seu tipo como *string*. Além disso como dito anteriormente há também o acréscimo de funcionalidades, na figura demonstra-se o exemplo das classes, algo que não existe no JS comum.

Figura 24 - Classe em TypeScript



Fonte: PRÓPRIA, utilizando o Visual Studio Code.

Para que o TS seja reconhecido nos navegadores é necessário um processo de compilação que transforma o código TypeScript para JavaScript que é entendido pelos navegadores. Além disso esse processo de compilação, também conhecido como “transpilação”, converte o EcmaScript 6 que é a versão mais atual do JavaScript em uma versão ao qual a maioria dos navegadores interpreta que é a EcmaScript 5 (ABREU, 2017).

#### Angular

O Angular é uma tecnologia criada pela Google afim de prover dinamicidade no processo de utilização de sistemas web. Segundo a Google (2019), o Angular apresenta como diferenciais: velocidade e desempenho, através de renderização no lado do servidor; a possibilidade de desenvolvimento para diversas plataformas, como dispositivos móveis de maneira nativa ou não, desktop e web com um único código; grande quantidade de ferramentas disponíveis, um exemplo se dá neste trabalho com a utilização de um plugin Angular para gerar calendários dinâmicos; e por último, o fato de ser utilizado amplamente, oferecendo as características produtivas dos aplicativos Google.

Então pode se dizer que o Angular se trata de um framework *frontend* que gera como resultado aplicações chamadas de *Single Page Application* (SPA), ou em português, aplicações de página única. Ou seja, não há o carregamento da página web a cada interação com o usuário. Entendendo melhor, o Angular age em um contexto assíncrono, diferente do PHP por exemplo que é um contexto síncrono. No Angular o usuário realiza uma ação que gera uma requisição no servidor, porém a página não é recarregada como no PHP. O utilizador pode fazer outras interações com a página ou até mesmo outras requisições ao servidor, sem que a primeira requisição tenha terminado. Isto gera um ambiente dinâmico de acesso e utilização.

Uma aplicação Angular é toda particionada em componentes, módulos, serviços, modelos, dentre outros. Um componente se trata de um pedação da aplicação que sofre interação com o usuário, como por exemplo um botão, ou um campo de pesquisa. Esse componente é formado por um *template* que é a parte HTML do componente. Por um Script TS que é responsável pela lógica que será empregada naquele componente. E por último, um arquivo de estilos CSS que é responsável pela estilização do componente. Os módulos são responsáveis por agrupar componentes, serviços etc. (GUEDES 2017).

Tendo em vista que este seja um a ambiente que tem por característica a interação contínua com o usuário, optou-se pela utilização do Angular para auxiliar nesse processo e deixar a utilização mais fluída e dinâmica.

#### *Hypertext PreProcessor* (PHP)

O *PHP Hypertext* *PreProcessor* (PHP) é uma linguagem de *script* usada principalmente no contexto *web* para fazer a manipulação de páginas HTML (PHP, 2018b). Para seu uso é necessário a abertura e fechamento de uma *tag* PHP conforme descrito na Figura 27.

Figura 25 - Exemplo de código PHP em página HTML



Fonte: PHP, 2018a, p.1.

O PHP tem por característica ser uma linguagem interpretada *Server-Side*, ou seja,executada no lado do servidor (SKLAR, 2016). Assim, uma aplicação utilizando o PHP pode ser disponibilizada para uma série de pessoas através de uma *Uniform Resource Locator* (URL) (SKLAR, 2016). Sklar (2016) explica que se detendo da URL, o usuário faz uma solicitação ao servidor que interpreta o documento PHP e retorna o resultado HTML, que é entendível pelo *Browser*. Os servidores *web* em que geralmente o PHP é utilizado são os servidores Apache ou Ngix (LOCKHART, 2015).

A linguagem apresenta uma série de recursos e aditivos, como por exemplo manipulação de imagens, recursos de abstração de banco de dados, além de pacotes dos mais variados fins criados pela comunidade e que auxiliam na popularização da linguagem e seu uso para diversos fins (SKLAR, 2016).

A utilização do PHP neste trabalho se dará por meio do *Framework* Laravel que será discutido na subseção 5.2.5.5.

#### *Framework* Laravel

O Laravel é um *Framework* baseado na estratégia de desenvolvimento *Model*, *View* e *Controller* (MVC) e feito sobre linguagem PHP, que tem como objetivo facilitar e melhorar o processo de desenvolvimento (STAUFFER, 2017). Stauffer (2017, p. 22) apresenta o porquê da utilização de *frameworks* para desenvolvimento dizendo:

Frameworks como o Laravel – e o Symfony, Sliex, Lumen e Slim – pré-empacotam um conjunto de componentes de terceiros com características personalizadas do framework, como arquivos de configuração, provedores de serviço, estruturas de diretório prescritas e carregadores de aplicativos. Logo, o benefício de usar um framework em geral é o de que alguém tomou decisões por você não só sobre componentes individuas, mas também sobre como eles devem ser integrados.

Portanto pode-se afirmar que um *framework* como o Laravel simplifica alguns processos comuns a sistemas (como por exemplo o processo de autenticação) por meio de pacotes prontos. Isso faz com que o desenvolvedor se concentre em processos específicos da lógica de negócio ao qual seu sistema pertence, agilizando o desenvolvimento (STAUFFER, 2017).

O Laravel demonstra como valores o aumento da velocidade de desenvolvimento e a satisfação do desenvolvedor. Para isso dispõe de uma série de ferramentas que auxiliam em diversos processos no desenvolvimento, como por exemplo o processo de autenticação, o envio de *e-mails*, processos de banco de dados (criação de tabelas, transações e consultas), entre outros (STAUFFER, 2017).

Tendo em vista a quantidade de recursos disponíveis pelo Laravel, uma comunidade que auxilia em momentos de dúvida, além do conhecimento prévio, decidiu-se pela sua utilização neste trabalho de conclusão de curso.

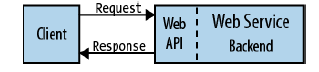
#### *Representational State Transfer Application Programming Interfaces* (API REST)

Segundo Massé (2012) o termo *Representional State Transfer* (REST) surgiu devido a necessidade de se ter outros verbos *Hyper Text Transfer Protocol* (HTTP) que representassem as ações de fato. Então surgiu-se em uma tese de doutorado o termo REST que se trata de acrescer verbos HTTP para as ações de atualização de dados (PUT ou PATCH) e para exclusão de dados (DELETE). Esses verbos vieram em acréscimo aos verbos GET (busca de dados) e POST (envio de dados), que antes eram utilizados para as ações de atualizar e deletar (MASSÉ, 2012).

Já o termo *Application Programming Interfaces* (API), surgem como o intermédio do usuário com serviços web. Servindo então de ponte entre o usuário e um serviço. Então quando se diz que uma aplicação funciona como uma API REST, quer dizer que essa aplicação possibilitará ao usuário as ações conforme descritas no modelo REST, além de servir de ponte para os serviços web, como por exemplo o serviço de banco de dados (MASSÉ, 2012). O usuário fará requisições a API que então será responsável por processar essa requisição e entregar o serviço requisitado pelo usuário.

Neste ambiente será utilizado dois frameworks que serão utilizados um no *frontend* (Angular), e outro no *backend* (Laravel). Então a aplicação Angular conforme descrita rodará diretamente no browser do usuário. Para que essa aplicação consiga comunicar-se com a base de dados será utilizado o Laravel como API ou intermédio. Ou seja, a aplicação Angular sempre que precisar de informações da base de dados irá requisitar a API Laravel que será responsável por processar a requisição e retornar os dados a aplicação Angular. Então a aplicação Angular demonstra os dados ao usuário, uma representação visual deste processo é descrita na figura X.

Figura 26 - Funcionamento de uma API



Fonte: MASSÉ, 2012, p6.

### Sistema de Gerenciamento de Banco de Dados (MySQL)

Bancos de dados estão presentes em todos os lugares em que se tem uso de computadores, desde a gestão de dados de uma multinacional, até numa farmácia de esquina. Mas o que de fato são bancos de dados? Elmasri e Navathe (2011, p. 3) definem como banco de dados “[...] uma coleção de dados relacionados” que possuem as seguintes características (ELMASRI; NAVATHE, 2011):

* Apresentar algum aspecto do mundo real;
* Ser um conjunto de dados que apresentem algum sentindo inerente;
* Projetado, construído e populado com dados com um fim em específico.

Conhecendo sobre o que se trata um banco de dados toma-se o conceito de Sistemas de Gerenciamento de Banco de Dados (SGBD) como “um conjunto de dados associados a um conjunto de programas para acesso a esses dados” (SILBERCHATZ; KORTH; SUDARSHAN, 1999). Portanto, para se gerir os dados que estão presentes em um banco de dados é necessário a atuação de um SGBD. O SGBD que será responsável então pela “[...] definição das estruturas de armazenamento das informações e a definição dos mecanismos para manipulação dessas informações.” (SILBERCHATZ; KORTH; SUDARSHAN, 1999, p. 1).

Dentre os SGBS, um que se destaca é o MySQL (CARVALHO, 2015). O MySQL é um SGBD gratuito e com base de licença *open source*, com ferramentas de desenvolvimento robustas e de qualidade (CARVALHO, 2015). A Figura 28 representa um mapa mental que contém as principais características do MySQL.

Figura 27 - Características do MySQL



Fonte: CARVALHO, 2015, p.3.

Com o MySQL é possível criar, editar e excluir dados através de sentenças na linguagem *Structured Query Language* (SQL). Além disso, como visto na Figura 28, apresenta características como possibilidade de uso de mais de um usuário, estabilidade, etc. Diversas linguagens de programação oferecem integração com o MySQL, e uma delas é o PHP e, por conseguinte o Laravel (STAUFFER, 2017). Um dos motivos da escolha do MySQL como SGBD deste trabalho foi o fato de ser gratuito e a fácil integração com o Laravel, sem se esquecer das características que auxiliam em qualidade e que foram descritas na Figura 28.

# desenvolvimento do ambiente proposto

Este capítulo vem demonstrar o processo de desenvolvimento do ambiente proposto, cuja a finalidade é auxiliar os processos de ensino e aprendizagem da língua inglesa na ILC.

Como o XP é um modelo de desenvolvimento incremental e dividido em entregas (*releases*), como discutido na seção numeroseção. Sabendo disso a estrutura deste trabalho foi dividida a contemplar cada release descrevendo as modelagens de cada release, as funcionalidades implementadas para cada perfil de usuário, e os testes utilizados para validar as funcionalidades.

Porém há um meio em específico que independe dos releases. Se trata das ferramentas utilizadas no processo de desenvolvimento. Para tal será destinado um tópico e após a finalização desse tópico, os *releases* serão abordados e destrinchados.

## Ferramentas de desenvolvimento utilizadas

Para o desenvolvimento da aplicação descrita foram utilizadas tecnologias que compreendem o *frontend* e também o *backend* conforme descrito na seção x. Além disso tem-se também ferramentas que apoiam o desenvolvimento, como as modelagens. E ferramentas de manipulação em bases de dados.

Quanto o *frontend* foi-se utilizado o framework Angular na versão 7.1.4 a versão mais atual na data em que se iniciou o desenvolvimento, tal como o TypeScript que se utilizou a versão 3.1.6. Utilizando-se *tags* e diretivas próprias do Angular, juntamente com HTML na versão 5 e CSS na versão 3. Isso se deu para buscar uma melhor qualidade visual. Juntamente utilizou-se o Framework CSS chamado Materialize CSS que traz componentes baseados no Material Design da Google.

Para o backend foi-se utilizado o framework Laravel na versão 5.5.44 com o PHP na versão 7.2.13. Foi-se utilizado o XAMPP que é a junção do Servidor Apache, o PHP e o MySQL. Como dito na seção x. O Laravel foi utilizado como API sendo o intermédio entre o *frontend* e a base de dados. Isso se deu pelo fato de se utilizar uma ferramenta específica para a parte visual da aplicação, deixando de lado os componentes de auxilio visual do Laravel.

Para a modelagem de processos, foi-se utilizado o Bizagi Modeler, que oferece todos os componentes necessários para se modelar um processo, além de oferecer uma funcionalidade de validação da modelagem. Ainda é possível exportar as modelagens para diversos tipos de extensão, como png, além de oferecer todos esses recursos de maneira gratuita. Essa validação ajuda a encontrar erros de modelagem, bem como erros de conexão entre as atividades do processo. Já para a modelagem de banco de dados relacional foi-se utilizado o MySQL Workbench na versão 8.0. O Workbench oferece uma gama de opções de modelagens, como a parte de relacionamento de tabelas, o que agiliza o processo de desenvolvimento.

Quanto as tecnologias de codificação utilizadas, foi-se utilizado duas distintas. Uma para o *frontend* e outra para o *backend*. Para a primeira foi-se utilizada o Visual Studio Code (VSCODE) da Microsoft, pois apoia o desenvolvimento em TypeScript auxiliando em complementação de nomes de funções e pacotes. É uma solução gratuita e completa, pois conta com uma comunidade que desenvolve uma série de plugins que auxiliam vários processos de desenvolvimento. Já para o *backend* utilizou-se uma ferramenta paga chamada PHP Storm. Porém há distribuição gratuita para estudantes até que concluam seus estudos. Ela oferece uma série de recursos que auxiliam o desenvolvimento, aumentando a produtividade e velocidade de desenvolvimento.

## Release 1 – Cadastros Básicos

O primeiro release é marcado por ser a inicialização e estruturação do projeto. Pode compreender as funcionalidades mais básicas, mas como pregado no XP, deve-se desenvolver o que for de maior necessidade para o usuário no momento. Então serão apresentadas funcionalidades que são básicas e necessárias no sistema para sua utilização, como também as funcionalidades que foram requisitadas com maior necessidade para o cliente.

### Estruturação do sistema

Alguns modelos de dados e documentos foram utilizados para suportar o desenvolvimento do ambiente. Como modelos têm-se a modelagem de banco de dados e a modelagem de processos com o BPMN. Essas modelagens servem de auxílio ao desenvolvedor já que dá uma visão acerca do problema a ser resolvido. Como documentação para o ambiente, seguindo o que é pregado pela metodologia XP, foram utilizados além das modelagens, as estórias de usuário e os testes. Esses documentos serão descritos e apresentados nas seções seguintes.

#### Diagrama de banco de dados

Através de entrevistas e estudo dos requisitos gerou-se um modelo de banco de dados do ambiente para o primeiro release. Este modelo, por se tratar de um banco de dados relacional, vem explicitar as entidades e os seus relacionamentos. Assim os próximos parágrafos explicam o significado de cada tabela e o seu motivo de relacionar com outras tabelas.

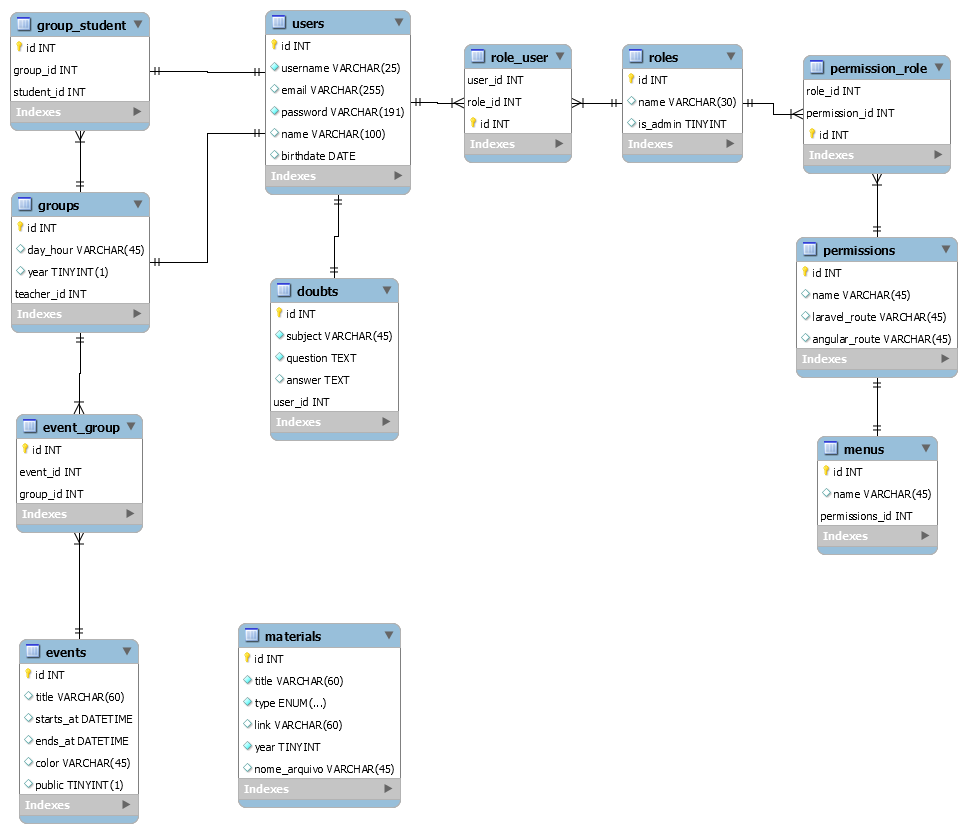
A figura x representa o modelo de banco de dados relacional do primeiro release. Nota-se primeiramente ao observar a figura que as tabelas têm seus nomes no idioma inglês. Isso se dá para uma melhor adequação ao Laravel que reconhece os nomes das tabelas em seu *Models* e acrescenta a pluralização através do idioma inglês. Se o *model* se chama *User*, o Laravel automaticamente entende que deve procurar na base de dados uma tabela com nome *users*. Porém isso pode ser mudado na configuração do Laravel, mas a escolha do idioma inglês poupa esse tempo de trocar as configurações de cada *model*. Outro motivo pela escolha do idioma inglês se dá pelo pensamento de expandir esse projeto no futuro, então para padronizar deixou-se os nomes em inglês e seguindo o padrão do Laravel.

Como dito nos parágrafos anteriores, a confecção das tabelas foi feita através de entrevistas aos professores e gestores da escola. Tem-se a tabela base de usuários (*users*) que se relaciona com diversas tabelas, um desses relacionamentos é com a tabela de perfis (*roles*) que dita qual perfil o usuário tem. Além disso a tabela de usuários também se relaciona com a tabela de turmas (*groups*), de duas maneiras uma sendo aluno e outro um usuário professor. Por último a tabela de usuário se relaciona com a tabela de dúvidas (*doubts*). Esse relacionamento se trata de uma dúvida de um aluno.

Quanto as turmas, podem-se relacionar com os eventos (*events*), já que um evento pode ou não pertencer a uma turma. O atributo *public* da tabela de eventos, indica se o evento foi cadastrado para uma turma em específico ou para toda a escola.

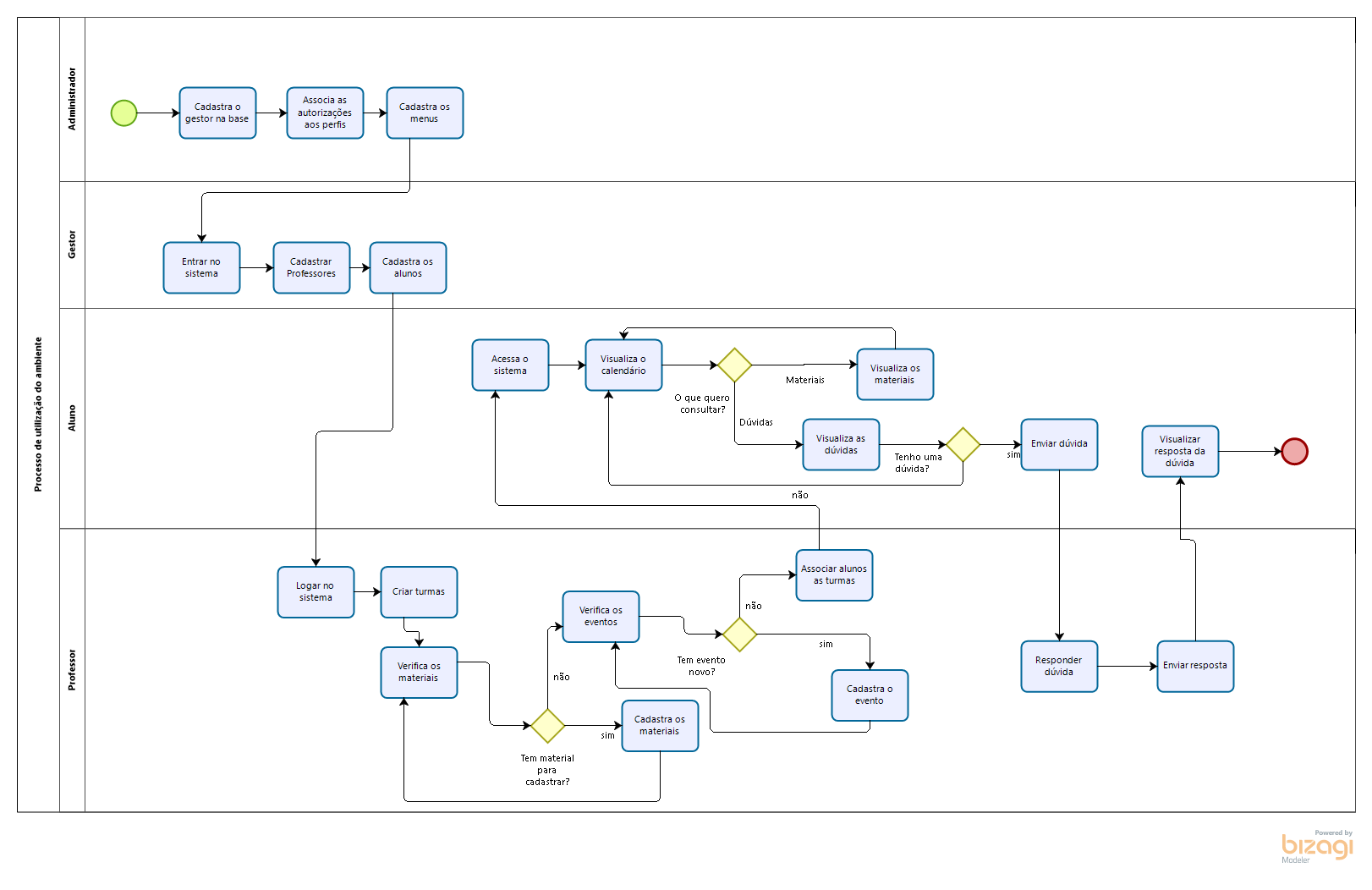
Tem-se ainda a tabela de permissões (*permissions*), que se relaciona com duas outras tabelas, a de perfis, indicando que cada perfil pode ter n permissões. E se relaciona com um menu indicando que o menu deve conter uma permissão, permissão essa que vai indicar qual o caminho o usuário deve seguir ao clicar no menu.

Tem-se a tabela de materiais (*materials*) que não se relaciona aparentemente com ninguém, porém os materiais que um aluno recebe podem ser filtrados por meio do ano em que o aluno se encontra. Então há um relacionamento indireto da tabela de materiais com as tabelas de aluno e turma (já que é na turma que se encontra o ano em que o aluno está).



#### Diagrama de processos

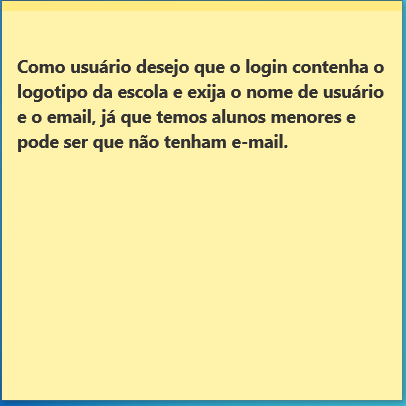
Para o primeiro release, como foi dito anteriormente, focou-se em funcionalidades iniciais de cadastro juntamente com as dúvidas dos alunos e as turmas. O processo foi modelado a contemplar esse processo de cadastros. Então há a figura inicial do administrador que é responsável por cadastrar o gestor na base. Feito isso o administrador é responsável por cadastrar os menus da aplicação e autorizar o que cada perfil pode fazer dentro da aplicação. Ao término dessa etapa, o usuário com perfil de gestor entra no sistema e cadastra os professores da escola, posteriormente cadastra todos os alunos também. Assim os professores podem criar suas turmas e posteriormente associar esses alunos que foram cadastrados pelo gestor. Porém antes disso o professor deve inserir materiais caso haja algum material, ou inserir eventos da turma para que o aluno possa receber um calendário específico da sua turma. Feito isso o aluno entra e verifica seu calendário que é apresentado logo na tela inicial do ambiente. O aluno ainda pode visualizar seus materiais ou as suas dúvidas. Caso o aluno tenha uma nova dúvida, deve envia-la ao professor que a responderá. Todo esse processo pode ser visto na figura X.



### Sistema desenvolvido

As estórias de usuários, conforme descrito na seção X, são um modelo de se recolher os requisitos e documentação considerado pelo XP. Então para apoio do ambiente proposto foram colhidas as estórias de usuários para cada requisito do ambiente. As estórias estão dividas de modo a compreender as necessidades de cada perfil de usuário do ambiente. Os perfis de usuário são o aluno, professor, o gestor e o administrador (ou desenvolvedor). Todas as interfaces foram implementadas utilizando o *material design* através do Framework CSS, MaterializeCss.

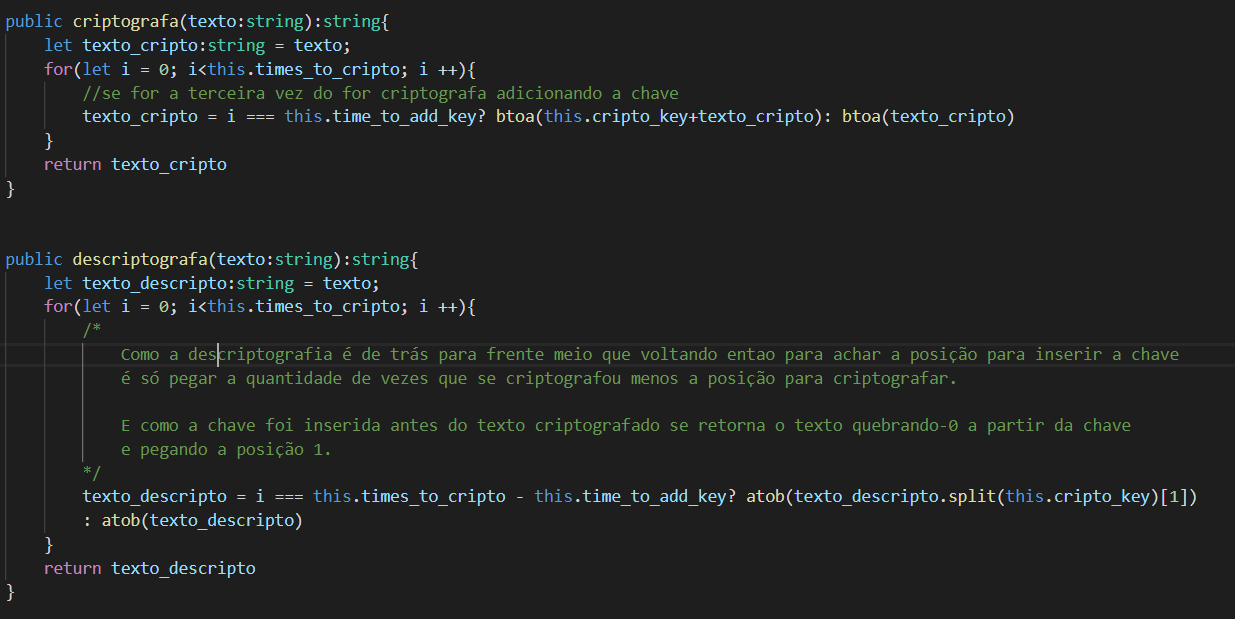
O release foi dividido por cada perfil de usuário, sendo apresentado as funcionalidades que serão utilizadas por estes perfis. Porém há três estórias que são válidas para todos os perfis de usuários. Se trata da funcionalidade de login descrita na figura X, a funcionalidade de notificação descrita pela figura x e a troca de senhas.



Essa estória define como será a interface de login que pode ser vista na figura X. Além disso as estórias descritas neste trabalho seguem o modelo ideal de estória definido por Santos (2017), que define como estrutura: O nome do perfil de usuário que utilizará a funcionalidade, acompanhado do que o usuário gostaria de ser feito, e o porquê.



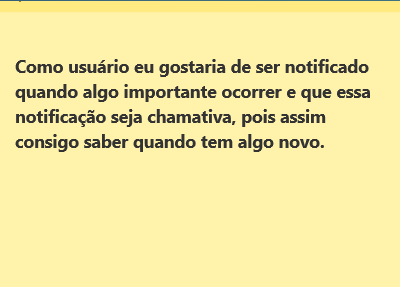
Deve-se ressaltar, como discutido na seção X os dados sensíveis, como a senha do usuário passaram por um processo de criptografia utilizado pelo Laravel, possibilitando assim a segurança dos dados. Porém por se tratar de uma aplicação que une um *backend* e *frontend* gerenciado por frameworks diferentes se fez necessário em alguns momentos criptografar os dados em que as duas aplicações conversam. Porém se faz necessário recuperar a informação no estado anterior a criptografia, então surge o processo de criptografia como sugerido na seção x. Um desses momentos se dá no retorno das informações do usuário no momento de login feito pela API. Alguns dados como informações dos usuários, menus, permissões, são salvas no *LocalStorage* que é uma espécie de memória local do navegador, que pode ser acessada e recuperada a partir do navegador. Então o usuário poderia facilmente descobrir o processo de criptografia, já que há como descriptografar. Pensando nesses problemas descritos desenvolveu-se uma função de criptografia e outra de descriptografia, como visto nas figuras x e x. Essas funções foram implementadas tanto no *backend* quanto no *frontend* para que assim as duas faces da aplicação consigam comunicar os dados sensíveis de maneira mais segura.



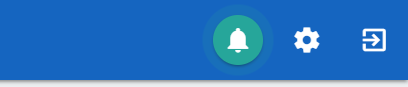
A figura x se trata das funções de criptografia e descriptografia, implementadas no *frontend*. O que a função de criptografia faz é rodar a função ‘btoa’ que codifica o texto passado para o método de codificação base-64. Essa técnica utiliza os caracteres como letras maiúsculas e minúsculas, números, ‘+’, ‘/’ e ‘=’. Além disso pode ser criptografada e descriptografada, a descriptografia é feita no JavaScript pela função ‘atob’. O que o algoritmo de criptografia faz é repetir a função ‘btoa’ um número de vezes definidas na codificação. Desse jeito ainda seria possível a um intruso pensar a mesma coisa. Aí entra o que foi discutido na seção x o acréscimo de uma chave de segurança. Então em algum momento definido, como visto pela variável ‘this.time\_to\_add\_key” o algoritmo acrescenta uma chave ao texto já criptografado. Isso dificulta já que para descobrir o texto original, deve-se conhecer a chave e saber em qual momento ela é utilizada. No processo de recuperação da mensagem original o que acontece é que simplesmente pegar essa posição de adição não resolveria já que o processo de recuperação é o inverso da criptografia, para não perder a referencia e saber em qual momento se deve adicionar a chave, pega-se quantas vezes são utilizadas para criptografar subtraído ao momento de se adicionar a chave. Para-se entender usamos a metodologia utilizada no XP chamada de ‘metáfora’, conforme explicitado na seção x.

Imagine-se que se está empacotando caixas, colocando umas dentro das outras. São um total de 5 caixas (que seria quantas vezes o dado será criptografado). Digamos que antes de empacotar a segunda caixa, colocou-se um selo. E assim, continua-se o processo de empacotamento até que as 5 caixas estejam uma dento da outra, algo como uma camada. Para que eu possa saber qual caixa eu devo retirar o selo, devo desempacotar a primeira caixa (que seria a última no processo de empacotamento), restando 4 caixas, fazendo o mesmo processo novamente, tira-se mais uma caixa (que seria a número 4 no processo de empacotamento), tira-se mais uma (que seria a número 3) e chegamos na caixa que inserimos o selo. Podemos assim removê-lo e ter a caixa em seu estado natural. Ou seja, adicionamos na segunda caixa, e removemos na terceira (5-2). Assim se dá essa função de criptografia criada, algo como empacotando o dado e um determinado momento inserindo uma chave, que ao desempacotar deve ser removida para que se consiga acesso ao dado original.

As notificações são um recurso responsável por avisar o usuário a respeito de algo novo que ocorreu. Serve para facilitar a utilização e identificação de recursos a serem utilizados no ambiente. A estória da figura x define como o usuário imaginou o recurso. A figura X é a demonstração de como ele foi implementado.

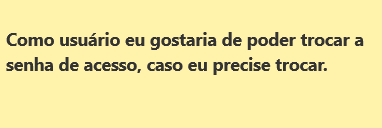


Assim como foi solicitado pelo usuário foram-se utilizadas cores chamativas, para dar um destaque ao elemento. Além disso foi adicionado um efeito de pulsação sobre o elemento que da uma visão de que o elemento está chamando o foco para si. Assim chama-se mais a atenção do usuário para o elemento.

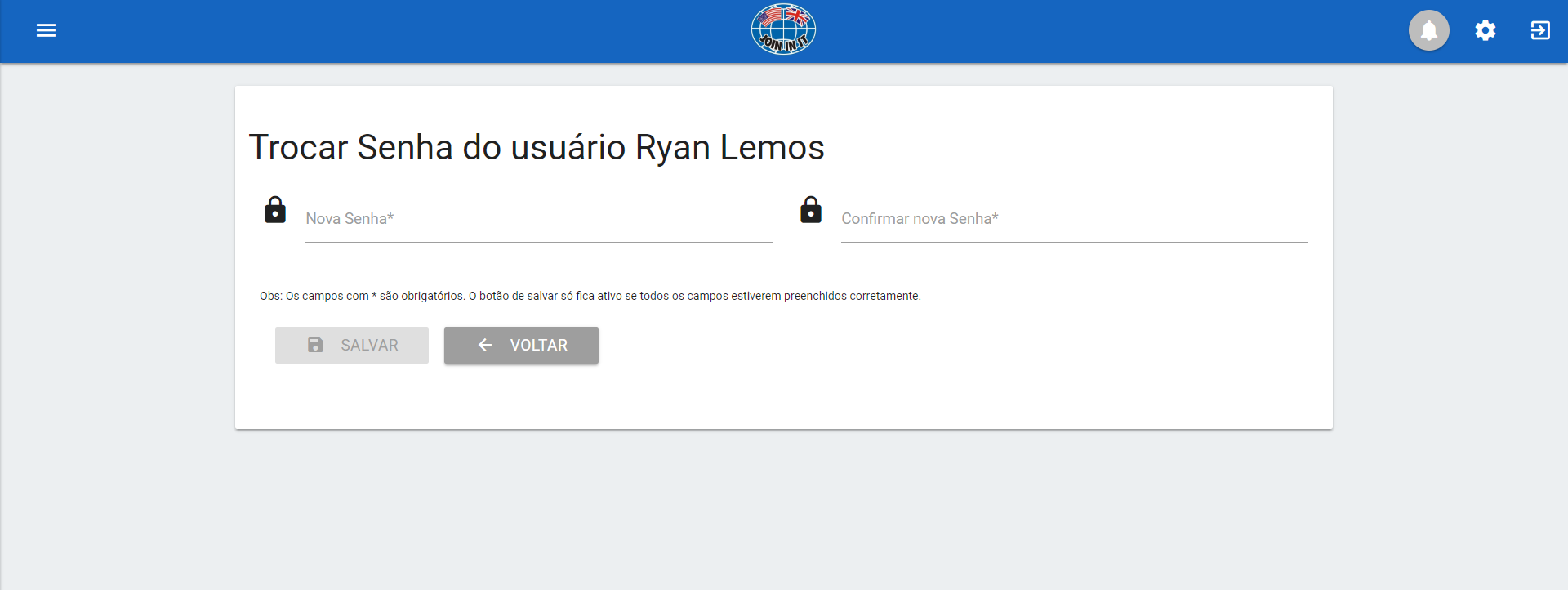


As notificações foram criadas utilizando uma funcionalidade própria do Laravel. Através dela pode-se mandar notificações de e-mail ou até salvar na base de dados, como é o caso da aplicação. A notificação é associada a um usuário, ao qual deve receber a notificação, e outros dados podem ser passados, como um texto ou dados do usuário que enviou a notificação. Esses dados adicionais são salvos por meio de um campo JSON. Sendo assim é possível criar vários tipos de notificação, cada uma com suas especificidades e utilizar uma mesma tabela de dados. Cada tipo de notificação criada no Laravel é composto por uma classe que deve ser criada pelo usuário, e pode ser criada utilizando o artisan que é a ferramenta de linha de comandos do Laravel.

Quanto a troca de senha, a estória representada pela figura X representa o que foi requisitado pelo cliente. É uma função simples, e a sua interface pode ser vista na figura X.

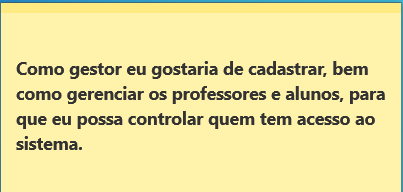


O usuário é capaz de trocar sua senha, digitando e confirmando a senha digitada, lembrando que a senha deve ser de no mínimo 6 caracteres.

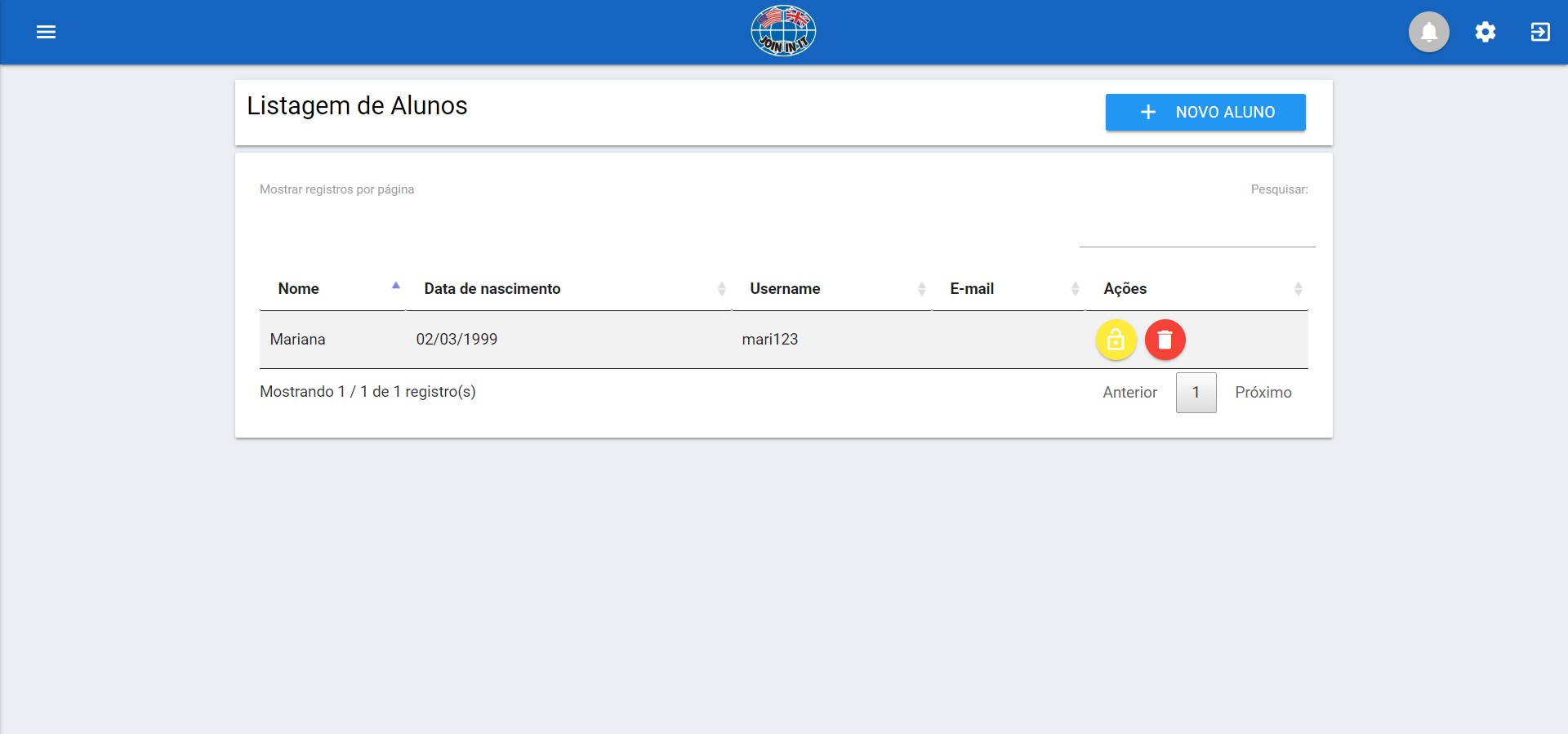


#### Gestor

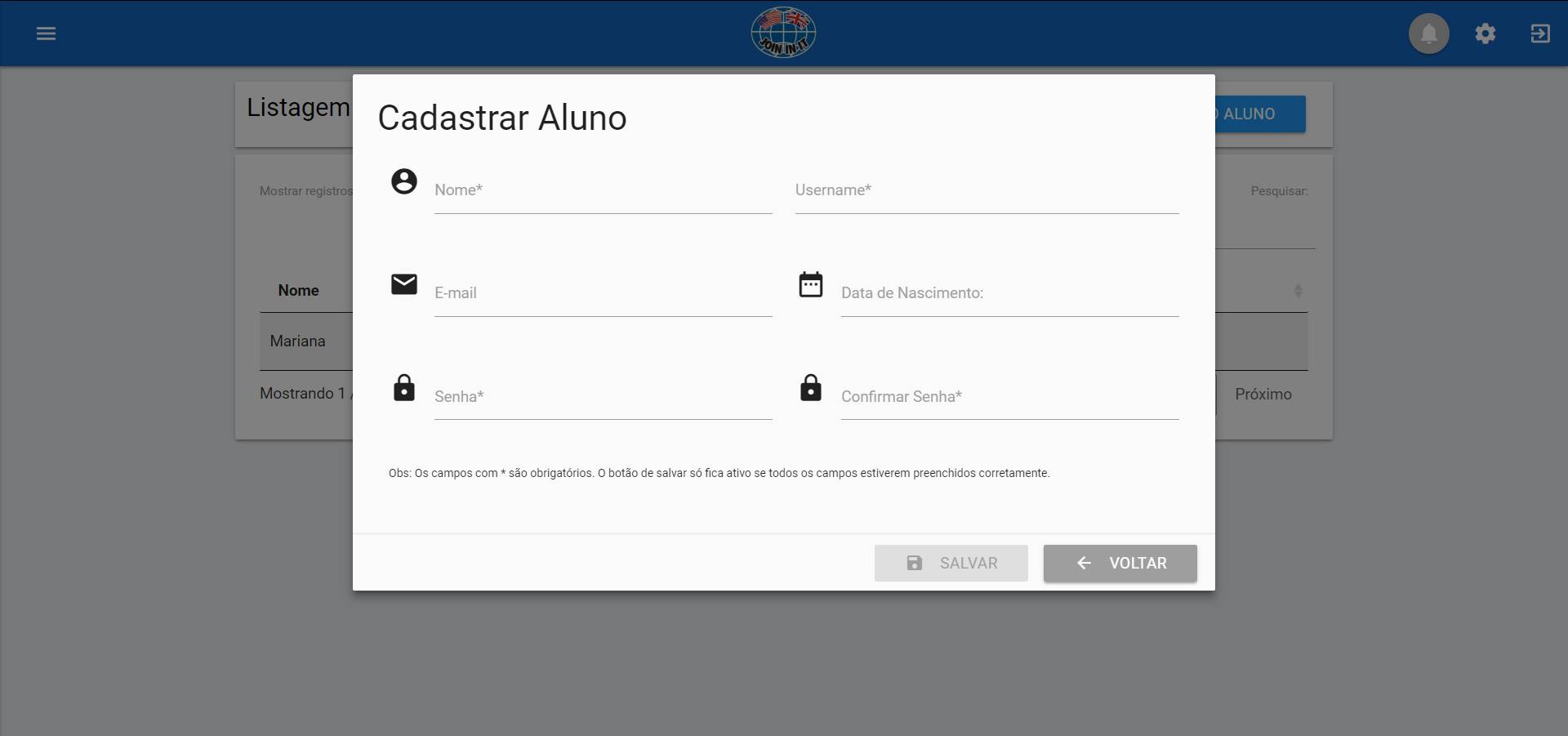
Os papeis do gestor nesse primeiro release compreendem em ações de cadastros de usuários (mais especificamente alunos e professores) e a gestão dos eventos da escola. Portanto a primeira estória compreende no cadastro e gestão de alunos e professores e pode ser descrita pela figura X.



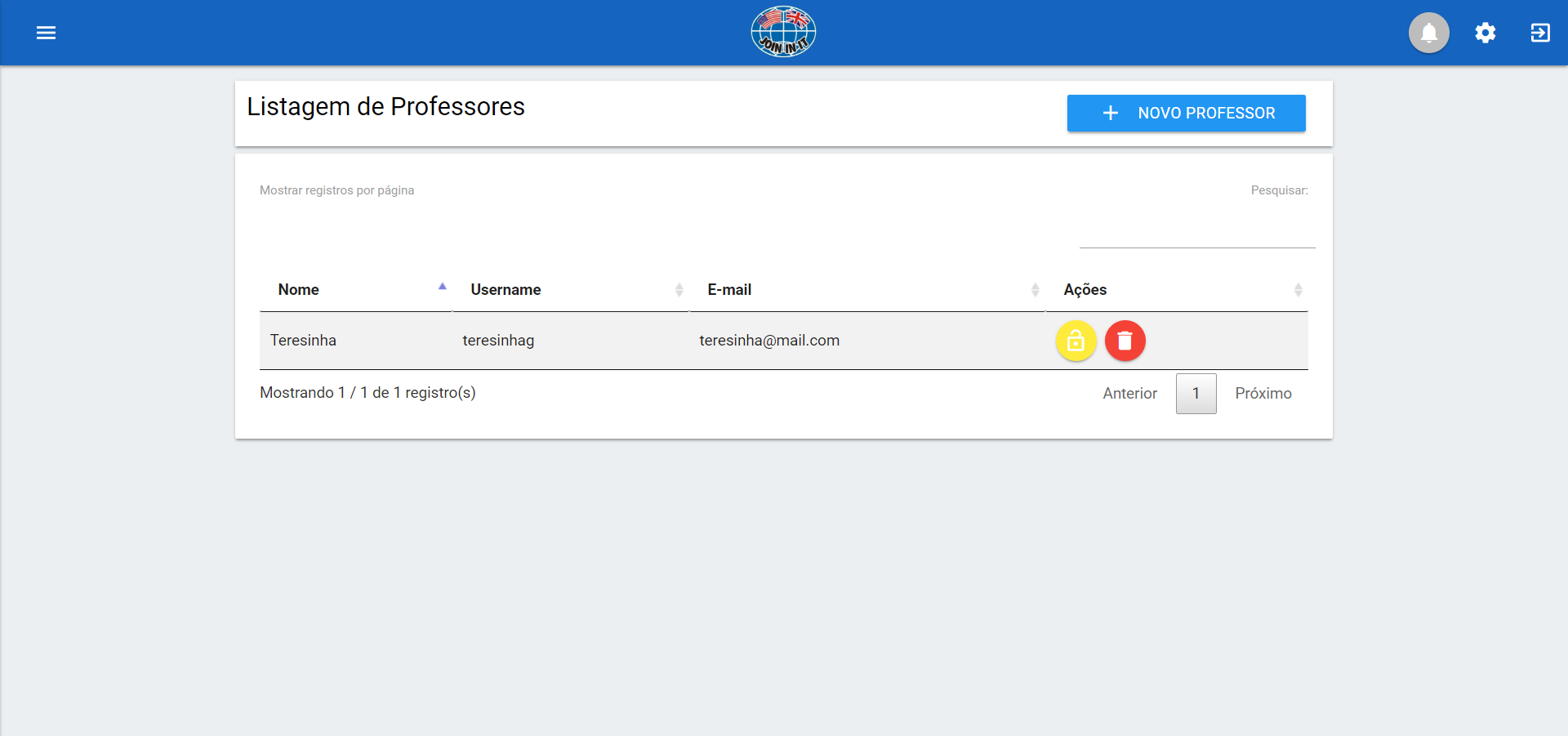
Na gestão dos alunos é possível que os gestores apaguem algum aluno ou troquem a senha do aluno. A troca de senhas é a mesma interação descrita pela figura X e permite trocar as senhas dos alunos em caso de perda ou esquecimento. Foi-se utilizado um recurso chamado *Datatables* que se trata de um plugin Jquery que monta uma tabela dinâmica. O próprio plugin adiciona os elementos de paginação, busca e filtragem. O que agiliza o processo de desenvolvimento.



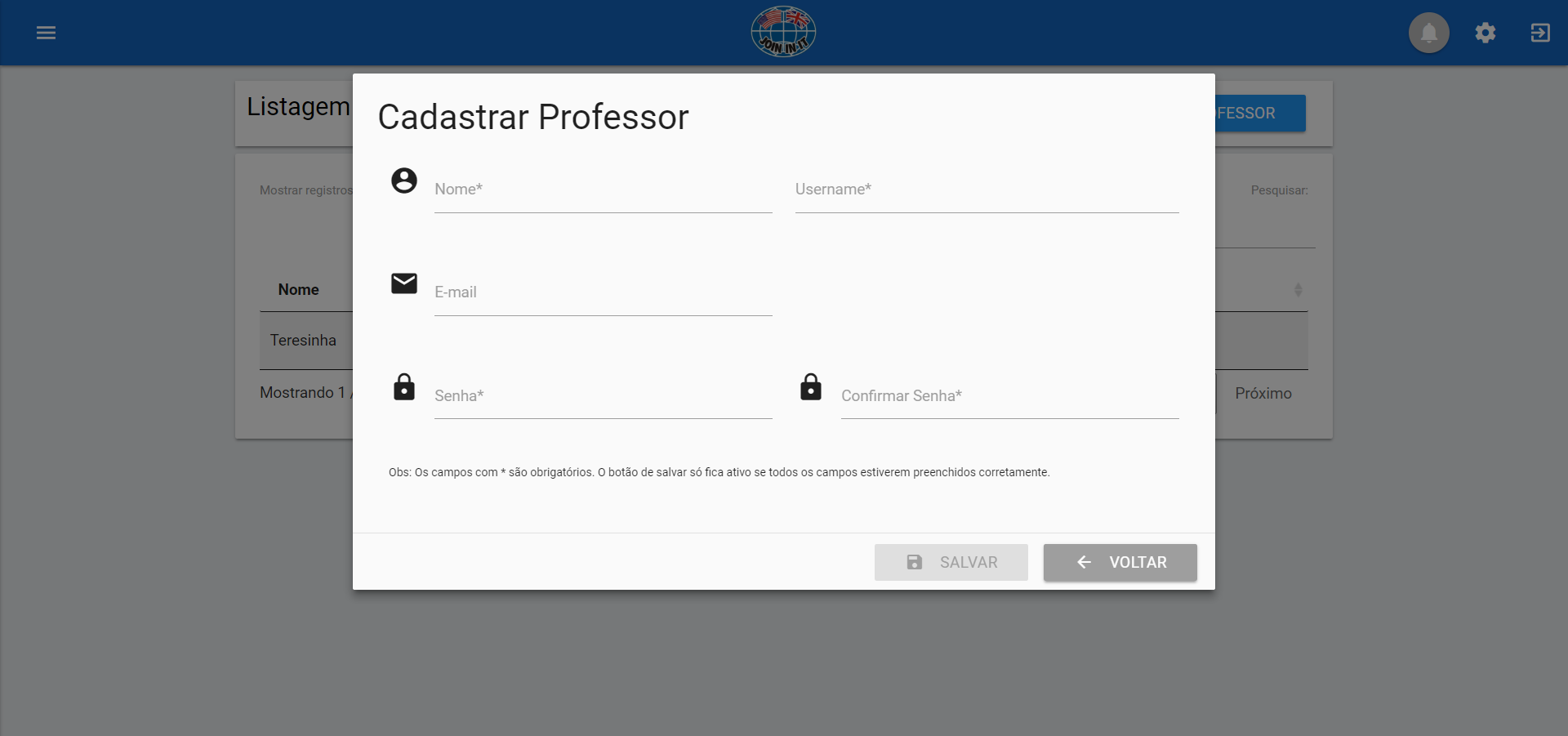
Quanto ao cadastro, foram identificados através de entrevistas juntamente com os gestores quais os dados constariam no cadastro. Foi definido então que teria o nome do aluno, juntamente com seu nome de usuário, e-mail caso o aluno tenha, a data de nascimento caso o aluno queira passar e a senha. Vale ressaltar que o *username* e o e-mail são identificações únicas. Portanto ao sair dos campos citados em caso de um *username* ou *email* já estiverem cadastrados na base, uma mensagem de erro surge dizendo que o usuário deve escolher outro *username* ou *e-mail*. Como visto pela figura x, cada campo tem um ícone relacionando, os ícones utilizados são disponibilizados pelo Google e podem ser utilizados não somente no quesito web quanto *mobile* também. Os ícones servem para dar um melhor entendimento da interação que o campo ou botão propõe. O MaterializeCSS contém elementos que se integram aos ícones disponibilizados pela Google, o que deixa a interface mais harmoniosa já que a integração é nativa.



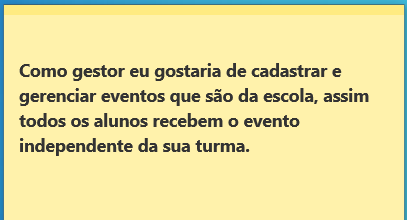
A listagem dos professores segue o mesmo princípio da de alunos. Pode-se pensar que poderia se utilizar somente uma interação para isso. Porém como os gestores e utilizadores do ambiente não tem um contato prévio com tecnologias, buscou-se então deixar o processo o mais simples possível dividindo em duas gestões.



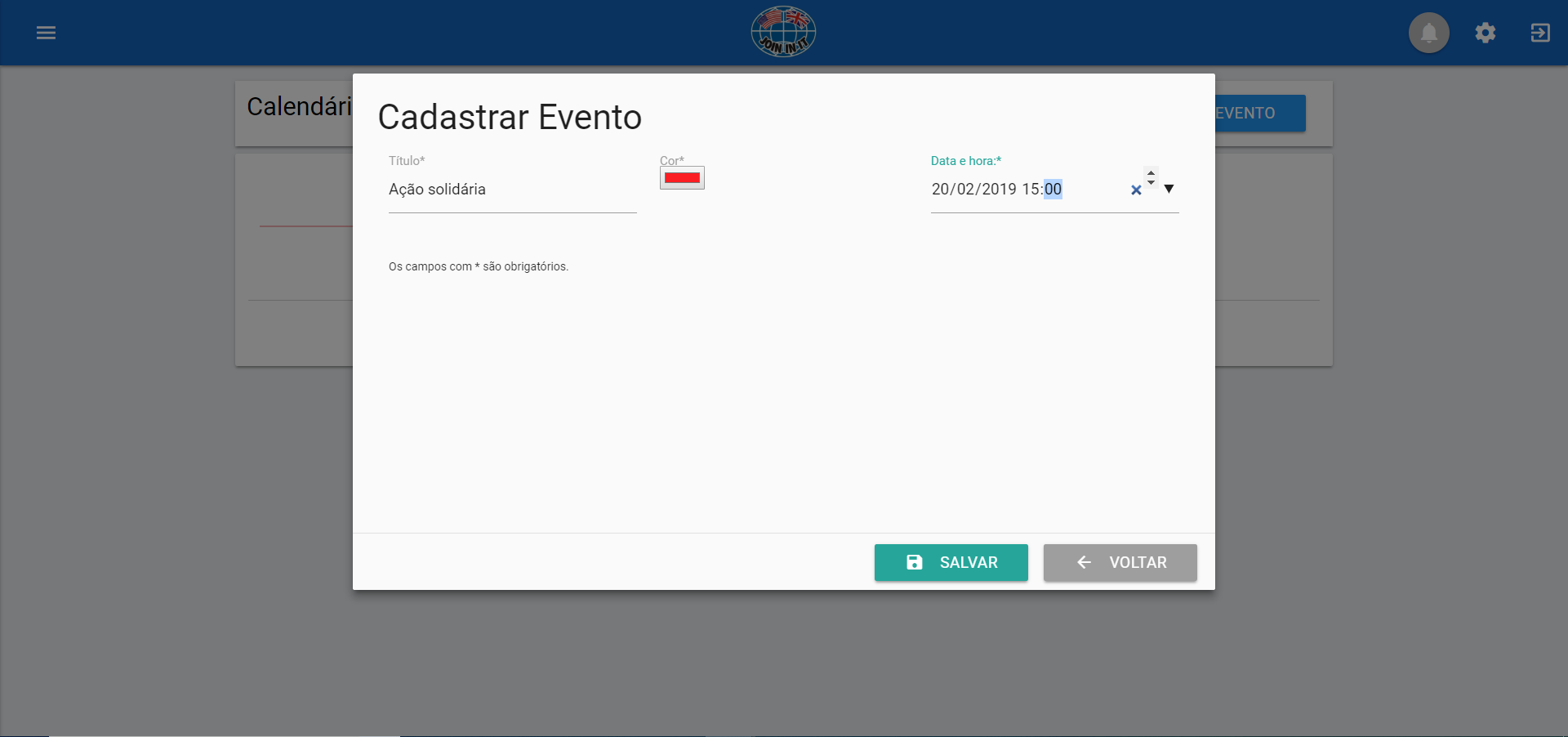
O cadastro dos professores também segue a linha do de alunos. A única diferença é a não existência do campo de data de nascimento. Como foi dito, essa divisão foi feita afim de deixar o processo mais simples e direto.



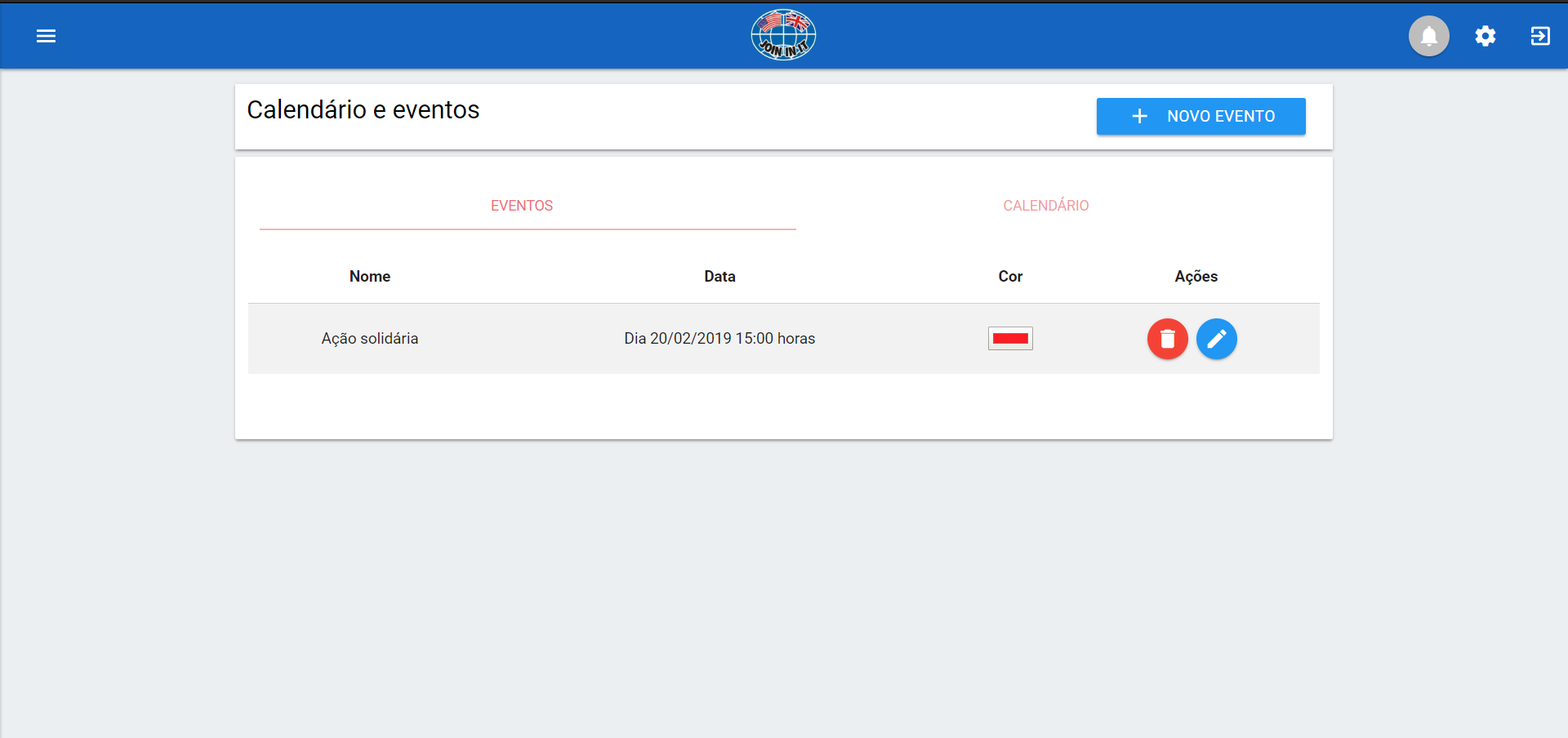
A escola como um todo pode oferecer eventos aos alunos, como uma gincana ou uma viagem por exemplo. Então surgiu-se a necessidade de que o gestor possa gerenciar esses eventos através do ambiente. Assim os alunos ficam sabendo do que está ocorrendo na escola. A estória definida pela figura x descreve esse processo pela visão do gestor.



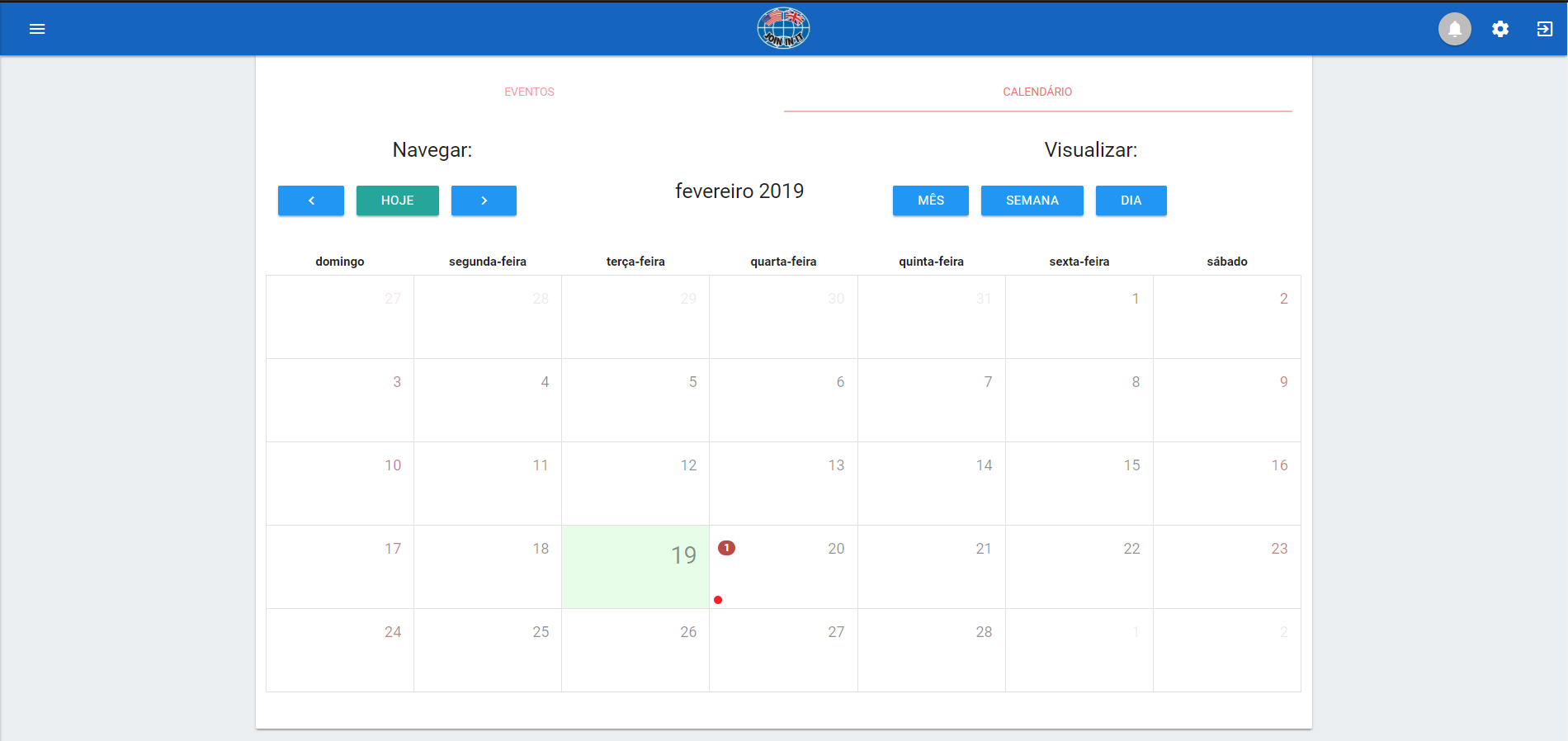
A figura x demonstra como se deu o processo de cadastro de um evento da escola. O gestor indica o nome do evento, data e hora, juntamente com uma cor, que serve para que o aluno possa identificar o evento no seu calendário. Vale ressaltar que não foram definidos padrões de cores, o gestor fica livre para escolher a cor que mais lhe agrada.



Após o cadastro o gestor fica disposto a uma tela que lista todos os eventos que ele cadastrou em uma aba e na outra ele pode ver os eventos no calendário conforme demonstrada pela figura x. Com a gestão dos eventos o gestor pode excluir um evento ou edita-lo. A tela de edição é semelhante a de cadastros que é descrita pela figura x.

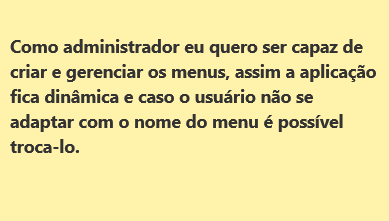


Ao clicar na aba de calendário o gestor tem um calendário interativo contendo os eventos cadastrados. Foi-se utilizado um plugin Angular que é responsável por gerar o calendário interativo, o que facilita a implementação já que se tem uma reutilização de algo já criado. Neste calendário o gestor pode interagir, mudando sua visão para dia, semana ou mês. Além de se locomover pelos dias, semanas ou meses no calendário. Os eventos aparecem marcados no calendário com a cor escolhida no momento do cadastro. Ao clicar em uma data com o evento, uma descrição do evento surge. Ainda há outra funcionalidade, em caso de mais de um evento para o mesmo dia o calendário mostra um contador de eventos naquela data juntamente com as cores daqueles eventos.



#### Administrador

O administrador é o perfil de usuário com acesso total ao sistema. Porém, há algumas funcionalidades, para ser mais exato duas, que somente o administrador pode desempenhar. Vale ressaltar que o administrador deve ter conhecimento em desenvolvimento para cumprir essas tarefas, já que as funcionalidades abordam aspectos específicos do desenvolvimento. A primeira função do administrador citada pela estória da figura x se trata do gerenciamento dos menus. Isso se dá, pois, os menus da aplicação são gerados de maneira dinâmica, não sendo assim fixos. O usuário tem a liberdade de trocar os nomes dos menus caso não se adapte ao nome.



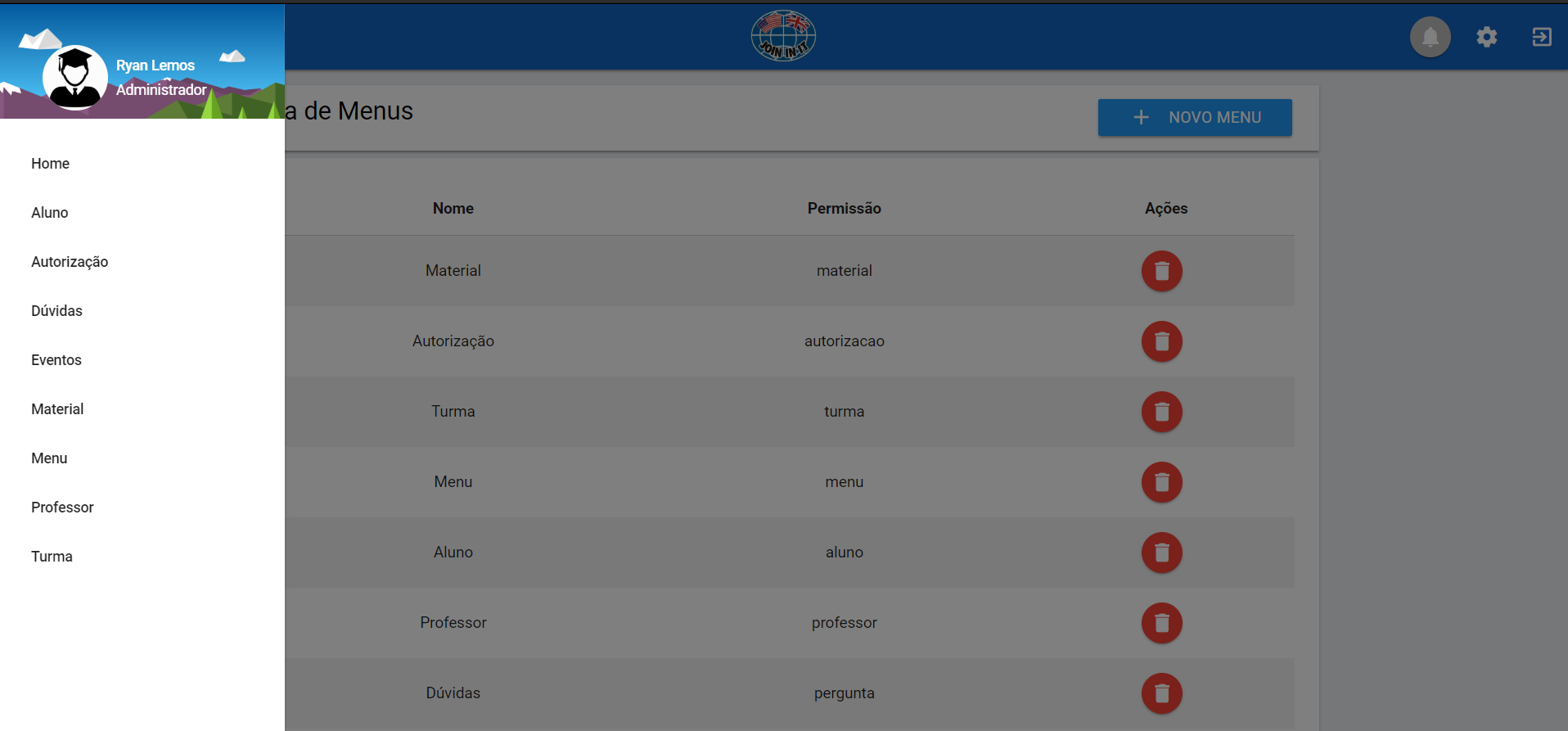
Como papel de gestão de menus, o administrador pode criar um menu ou excluir um menu já criado, conforme visto na figura x. Cada menu está ligado a uma permissão do sistema. Na verdade, essa permissão nada mais é do que a rota em que o usuário será direcionado ao clicar no menu.



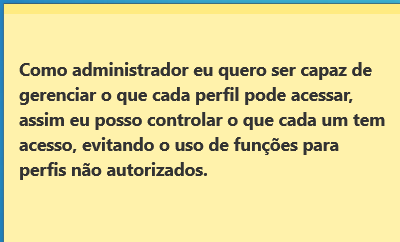
Ao clicar em cadastrar surge uma tela onde o administrador pode indicar o nome do menu a ser cadastrado, juntamente com a permissão associada ao menu como visto pela figura x.



A figura x se trata de todos os menus da aplicação no release 1. A listagem dos menus é feita com base no perfil do usuário e suas permissões. Ou seja, cada perfil tem um conjunto de menus associados. Contanto, há um menu padrão para todos os usuários e que não fica salvo na base. Se trata do menu *home*, que redireciona o usuário para a página inicial da aplicação.



Assim como os menus, as permissões dos usuários são dinâmicas. O administrador tem a função de delegar o que cada um pode acessar no ambiente. Portanto a próxima estória de usuário, representada pela figura x, descreve essa necessidade do ambiente.

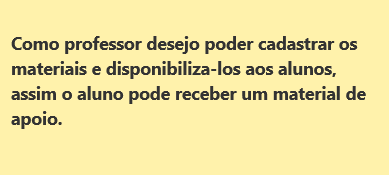


A interação descrita pela estória da figura x foi implementada conforme visto na figura x. O administrador escolhe qual perfil quer autorizar e as permissões surgem em seguida. O administrador marca quais permissões deseja ao usuário e clica no botão salvar. Assim surge uma mensagem de confirmação de autorização para o perfil de usuário. E o usuário com aquele perfil autorizado consegue acessar o que lhe foi permitido. Como descrito, as permissões nada mais são do que as rotas da aplicação. Mas as rotas foram divididas a contemplar os dois âmbitos da aplicação, o *frontend* e o *backend*. Ou seja, há rotas específicas do Laravel (que tem seu sistema de rotas), e as rotas do Angular que também tem um módulo de roteamento. Então para que o usuário acesse determinado recurso tem que lhe ser permitido as autorizações no Angular e no Laravel. Caso somente seja permitido em um âmbito, o perfil de usuário não conseguirá acesso por completo do recurso. Caso seja permitido acesso somente a rota do Angular o perfil só conseguirá visualizar a tela, porém não conseguirá interagir com a base de dados. Caso só permitir no Laravel o usuário não terá uma tela de interação, somente a possibilidade de requisição na API. Por isso se faz necessário que o usuário administrador tenha os conhecimentos necessários no desenvolvimento para permitir o acesso.

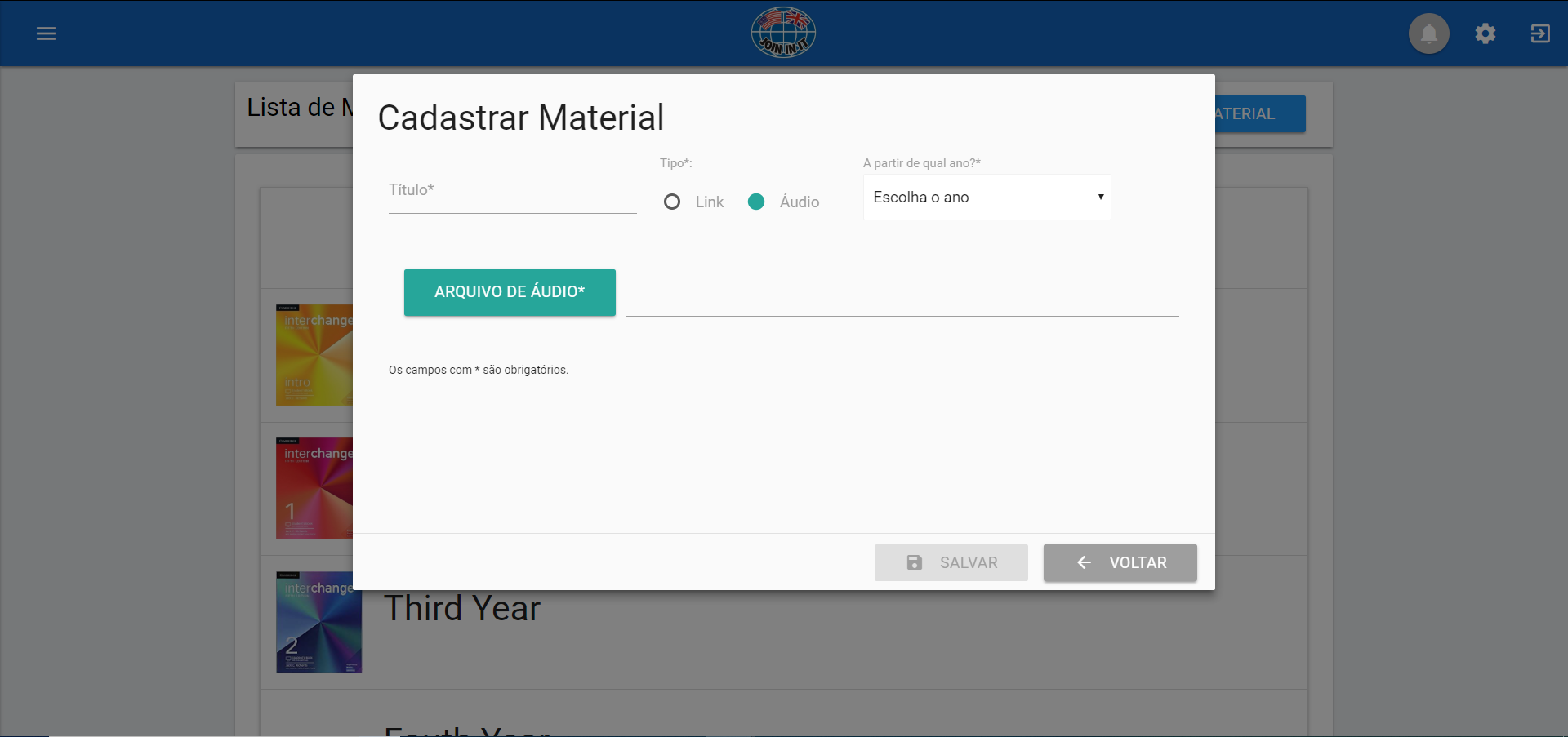


#### Professor

Esta seção se trata das funcionalidades implementadas no primeiro release que são direcionadas ao professor. Uma das responsabilidades do professor no ambiente é a disponibilização de matérias de apoio aos alunos. Com isso surge a necessidade do professor de inserir esses materiais no ambiente. A estória retratada na figura X representa esse desejo do professor.



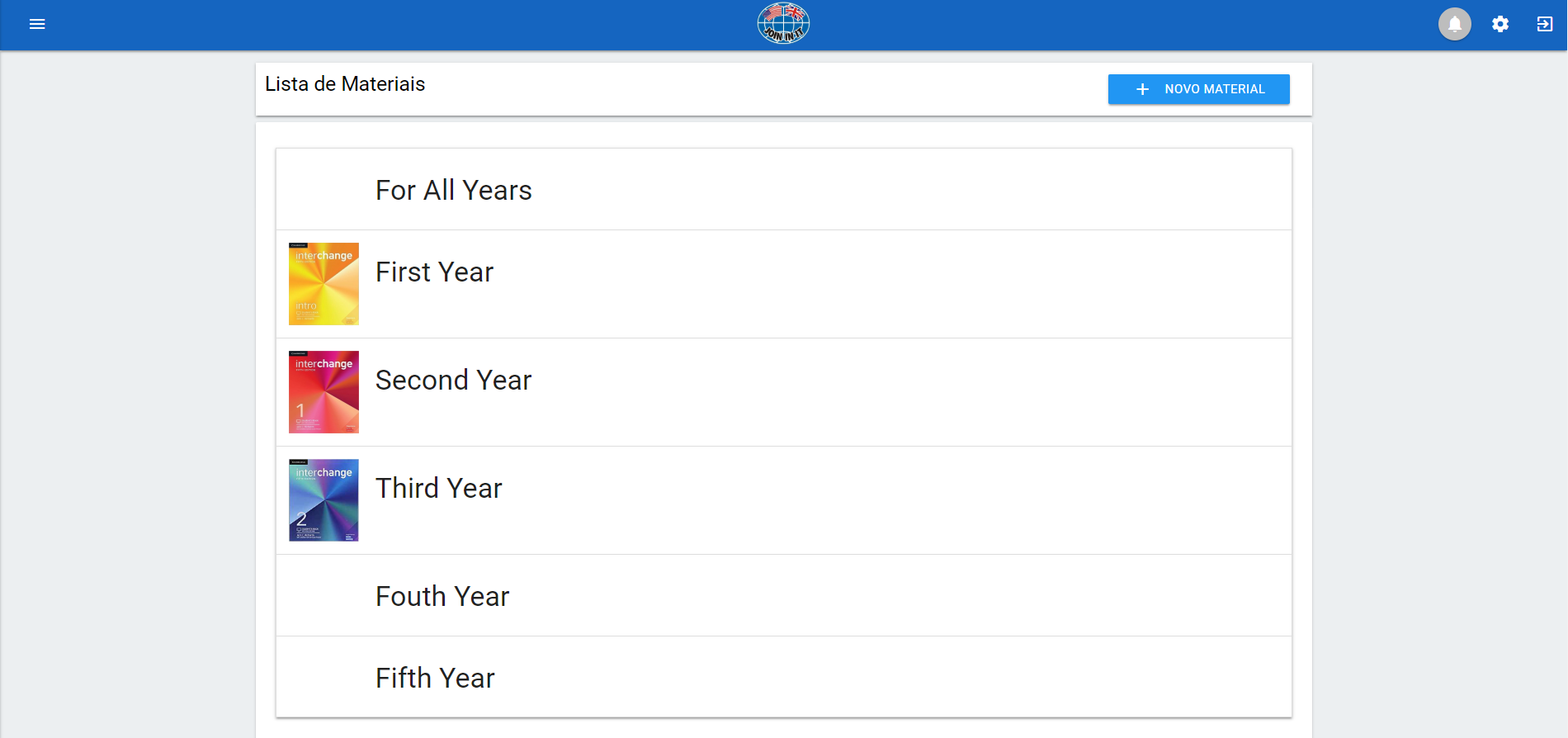
A implementação desta estória pode ser vista na figura X. Vale ressaltar que em entrevistas feitas aos professores, foi identificado que os tipos de materiais usados por eles são links ou áudios. Então o professor pode escolher se quer cadastrar um link ou um áudio. Em caso de escolha de link, surge-se um campo de digitação para indicar o endereço do link. Caso contrário surge um botão ao qual o professor pode enviar um arquivo de aúdio.



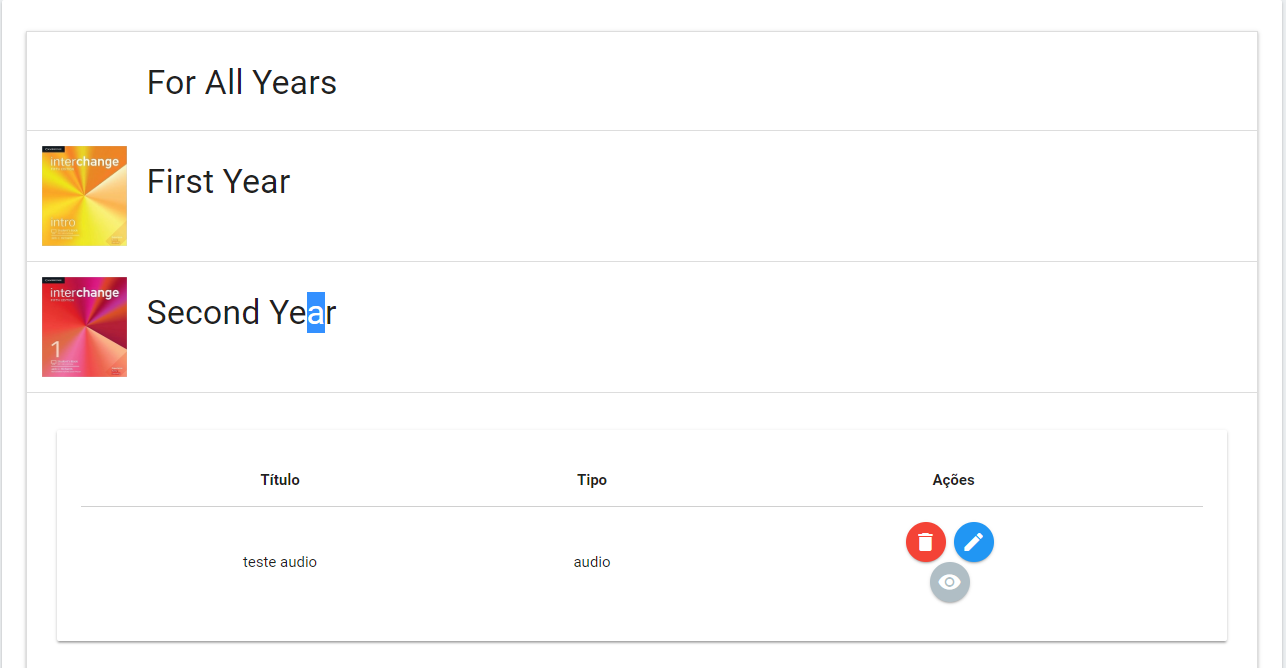
A estória retratada na figura x se trata de como seria a listagem desses materiais cadastrados. Essa estória, diferente das outras, contém restrições ao qual a funcionalidade deve possuir para ser válida ao usuário.



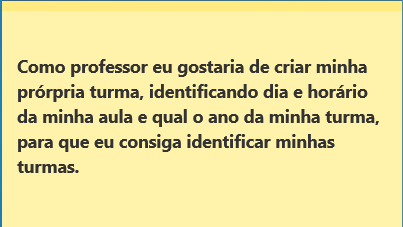
A implementação da funcionalidade pode ser constatada pela figura x. Os materiais são descritos pelos anos e ainda há a possibilidade de se ter um material que seja disponível para todos os alunos independente do seu ano. A primeira restrição descrita pela estória da figura X pode também ser vista na figura X. Uma vez que os livros utilizados pelo primeiro, segundo e terceiro ano acompanham o nome do ano para ajudar na identificação.



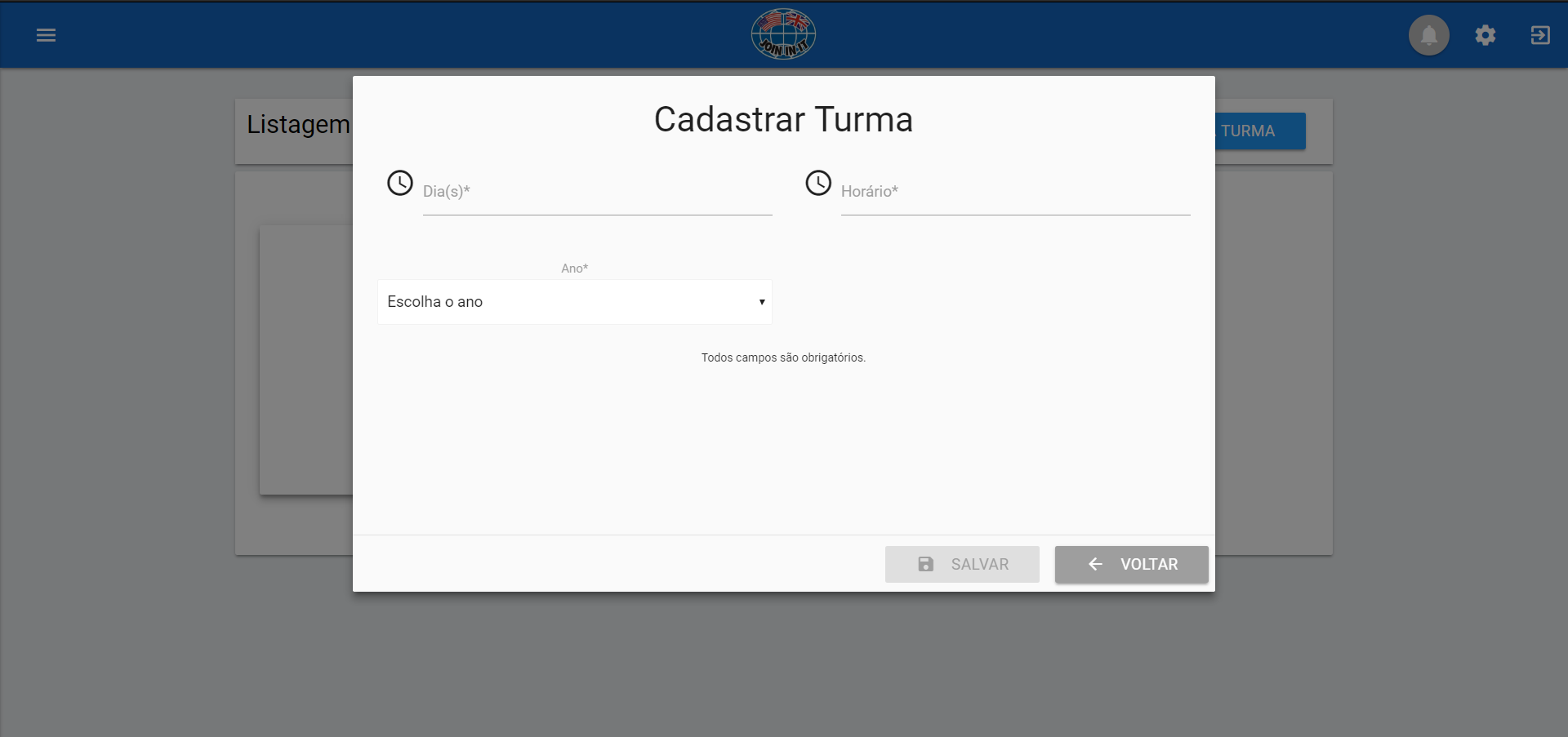
A segunda restrição descrita pela figura X implementada pode ser vista na figura X. Ao clicar sobre o ano executa-se um efeito de sanfona abrindo e então os materiais daquele ano surgem. Para essa interface foi-se utilizado um componente do MaterializeCSS que é responsável por gerar esse efeito sanfona descrito.



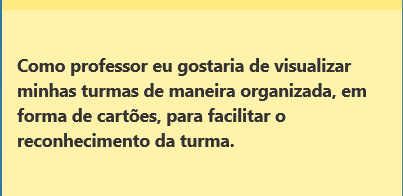
A estória seguinte se trata de como será o cadastro das turmas pelo professor. A figura X representa essa estória. Nela o professor explica que cada turma é identificada pelo ano de graduação (no caso primeiro, segundo, até o quinto ano), o dia e horários em que a aula é realizada.



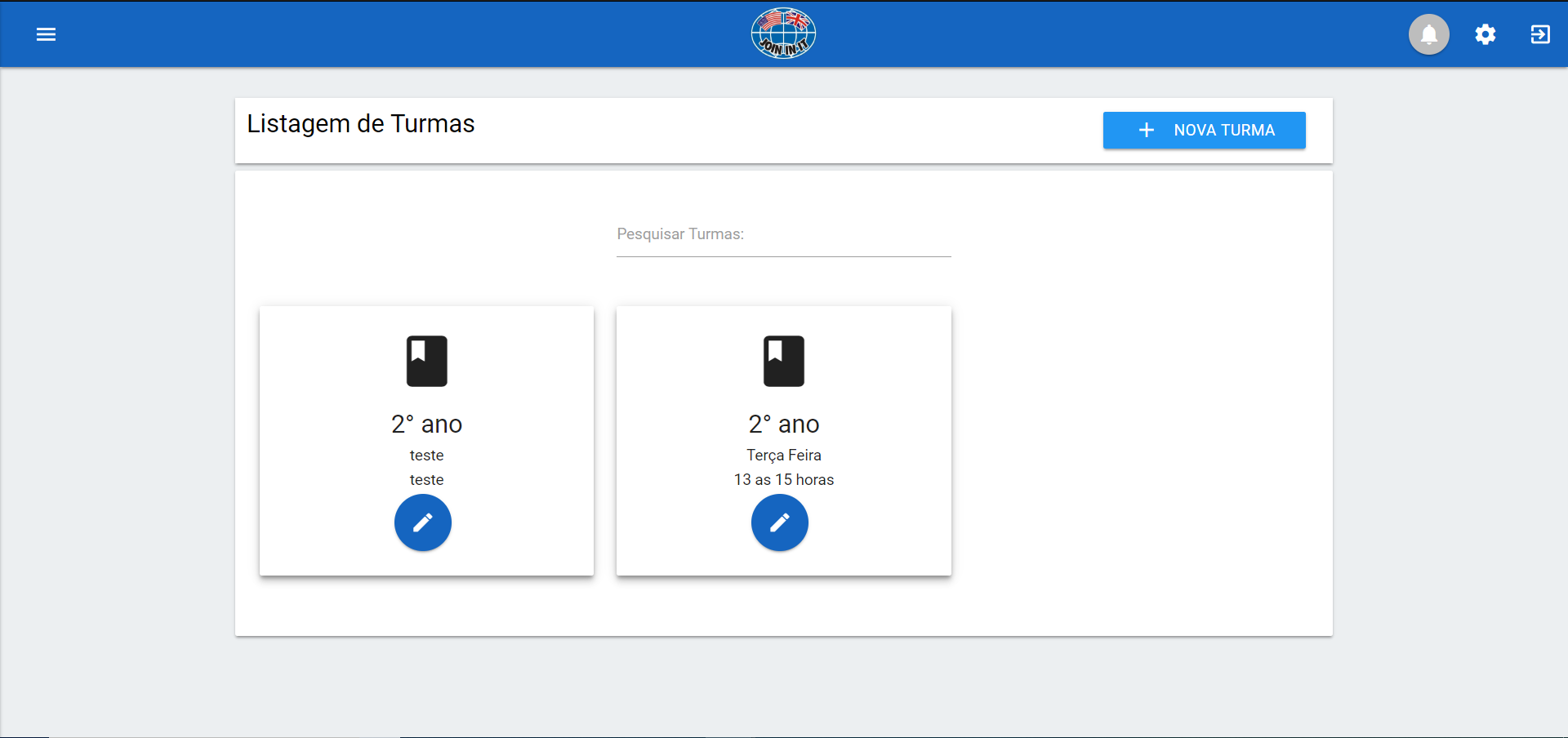
A implementação desta funcionalidade é descrita pela figura X que explicita os campos indicados pelo professor que são o dia, horário e ano.



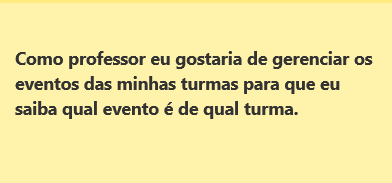
Ao professor também é possível visualizar suas turmas. A figura X se trata da estória que explicita como o professor imaginou a listagem das turmas. Um dos desejos para essa funcionalidade é que as turmas sejam dispostas em forma de cartão, para que fique mais fácil de identificar a turma.



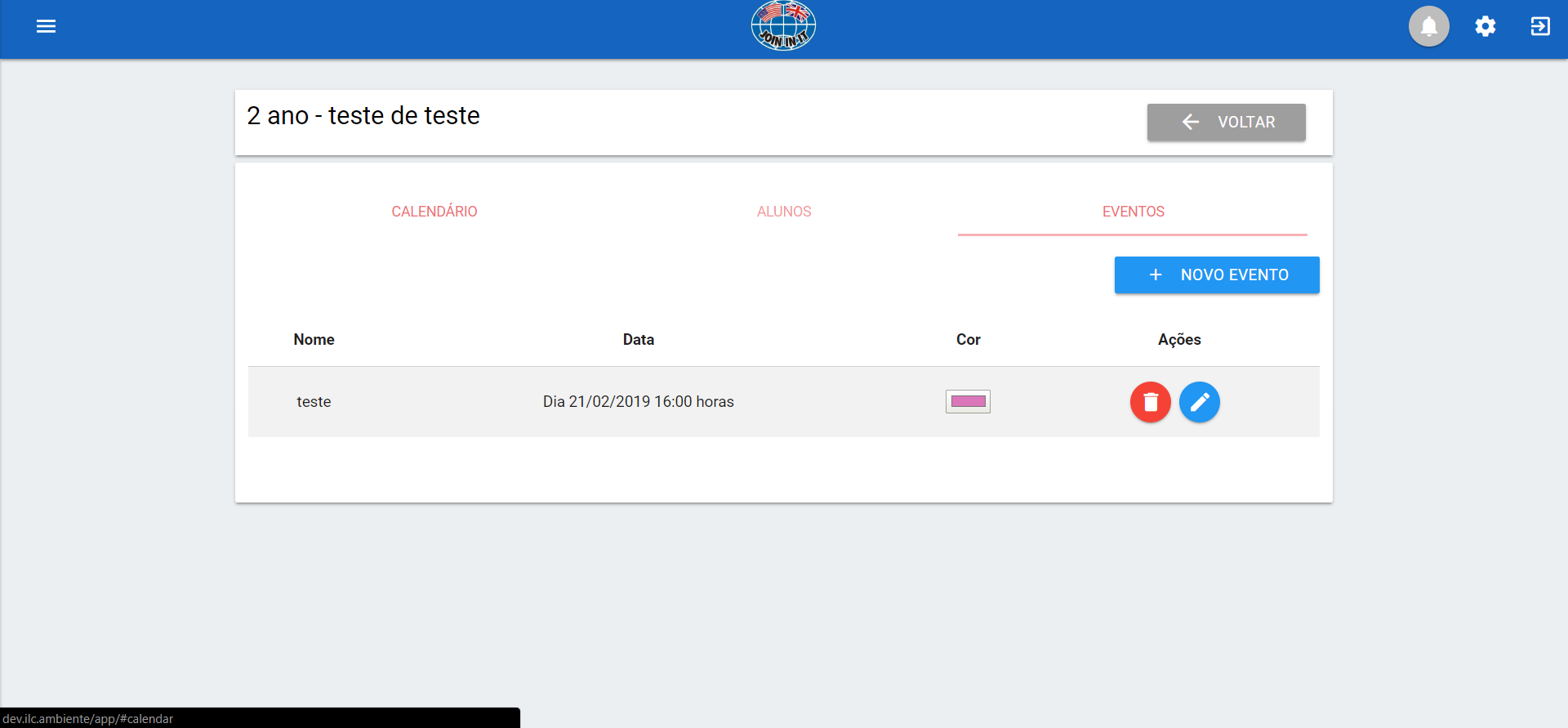
A figura X explicita como foi feita a implementação dessa funcionalidade. As turmas são listadas em forma de cartão conforme requisitado. Ainda é possível ao professor gerenciar uma turma em específico clicando no botão com a figura de um lápis. Além disso o professor pode pesquisar por uma turma, filtrando os resultados, e listando somente os cartões conforme a busca. Essa visão em cartões é característica do Material Design, então é um componente que já vem nativo no MaterializeCSS, o que facilita a implementação do requisito desejado.



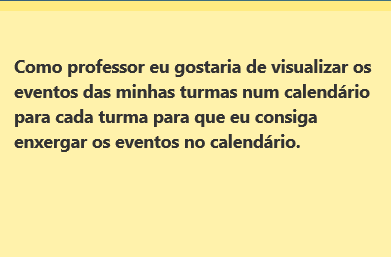
Como professor é possível dentro de uma turma, gerenciar os eventos da determinada turma. A estória descrita pela figura x representa esses anseios em se gerenciar os eventos.



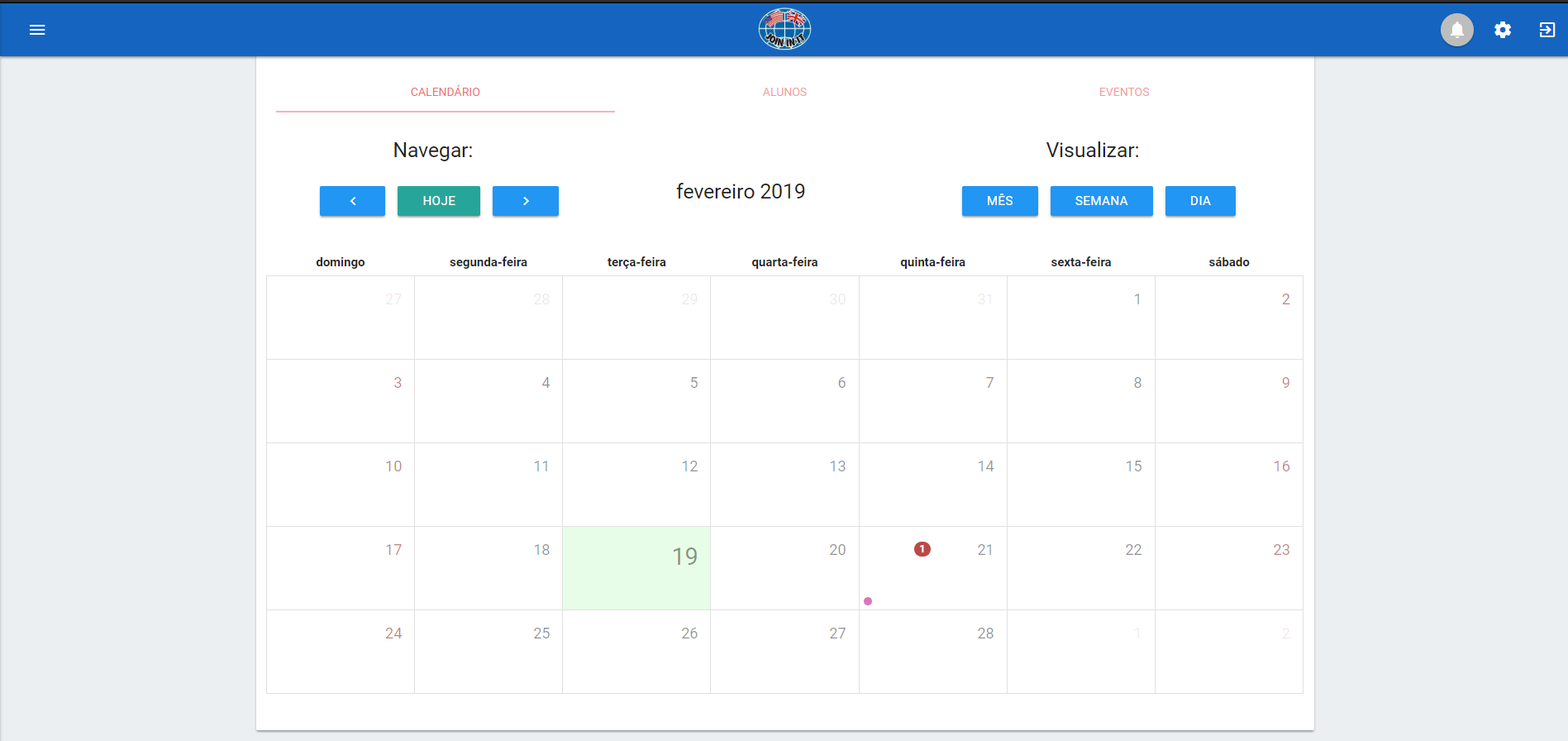
A figura x demonstra a implementação da estória da figura x. Dentro da turma o professor escolhe a aba eventos e então os eventos cadastrados surgem. É possível ao professor cadastrar, excluir e editar um evento de uma turma. O funcionamento dessa estória, juntamente com as interfaces e interações, é semelhante a estória da figura x.



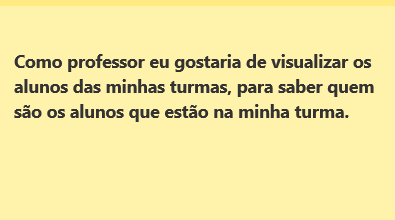
Ainda é possível ao professor utilizar o calendário para se situar conforme descrito pela estória da figura x.



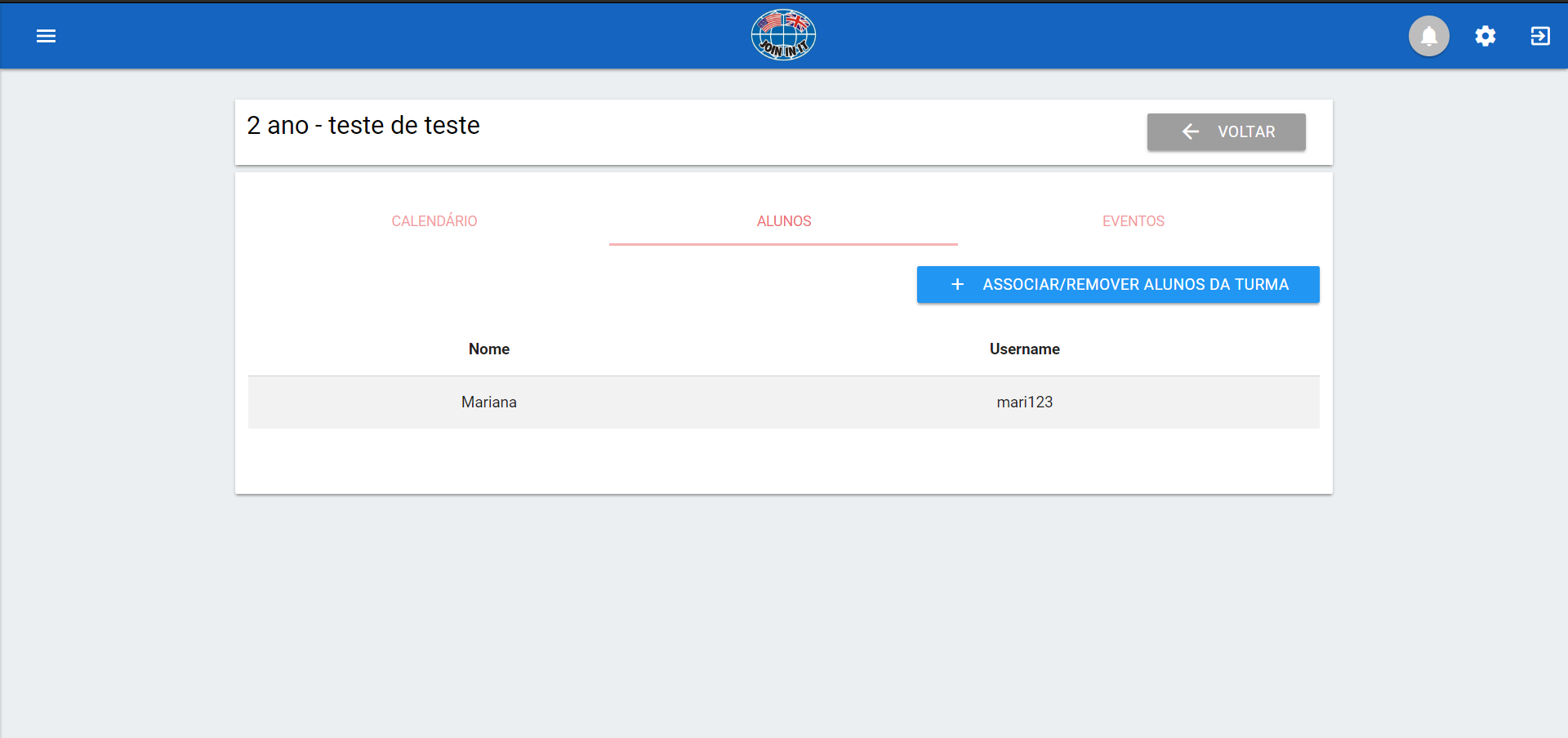
Dentro da gestão da turma, o professor pode acessar o calendário clicando na aba ‘calendário’. O formato do calendário é igual para o professor, aluno e gestor conforme visto na figura x. O que vai mudar são os eventos que cada um pode ver. O aluno pode ver os eventos da escola (cadastrados pelo gestor) e os eventos da turma (cadastrados pelo seu professor). O professor só pode ver os eventos relacionados a sua turma em específico. Quanto ao gestor, só pode ver os eventos cadastrados para a escola, o gestor não tem acesso aos eventos das turmas. O motivo dessa limitação foi para manter organização e evitar que professores possam gerenciar os eventos da escola que impactariam todos os alunos. Por outro lado o gestor ao gerir o evento de uma turma também impactaria nos alunos daquela turma.



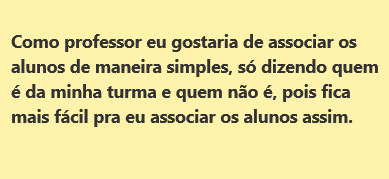
A estória da figura x representa a visualização dos alunos da turma. Assim o professor pode ver quem são os alunos que fazem parte da sua turma.



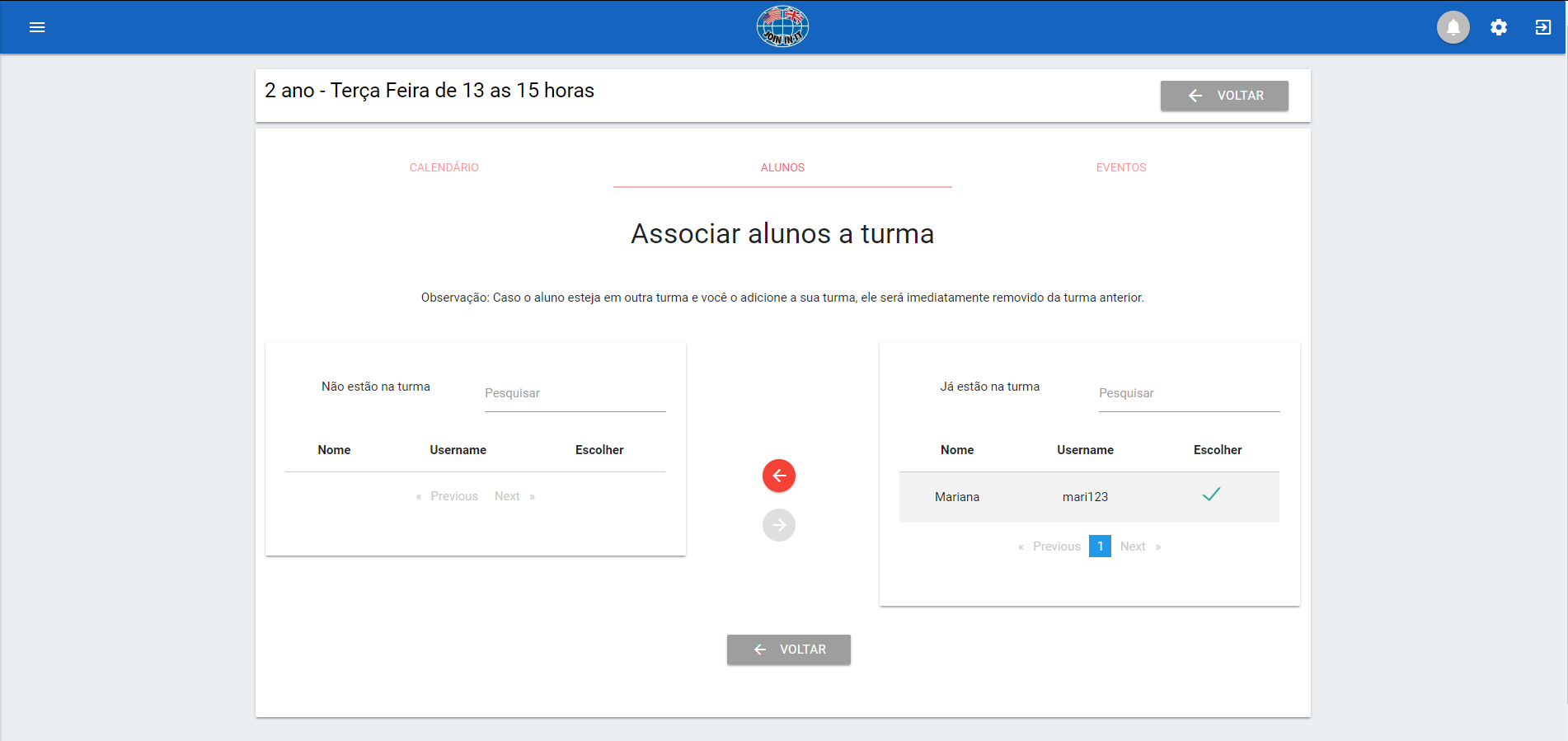
Ao entrar numa turma em específico o professor tem uma lista dos alunos que fazem parte da sua turma conforme descrito pela figura x.



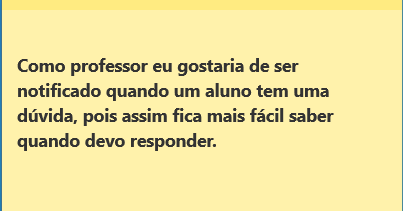
Ainda é possível ao professor, como evidenciado pela estória da figura x, gerenciar os alunos que fazem parte da sua turma. Podendo então remover ou adicionar alunos a sua turma conforme a necessidade do professor.



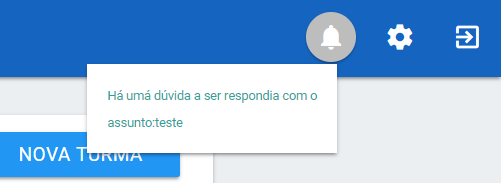
A figura x representa essa maneira de associar descrita pela estória da figura x. Buscou-se deixar o processo de associação de alunos o mais simples possível como requerido pela estória. Ao professor, tem-se duas tabelas, a da esquerda e a da direita. A da esquerda contém os alunos que não fazem parte da turma. A da direita, por conseguinte se trata dos alunos que fazem parte da turma. Cabe ao professor marcar quem ele quer na turma, pode pesquisar em caso de muitos usuários, e ao marcar um aluno o botão com o ícone de seta em direção a direita fica ativa na cor verde indicando que o professor irá adicionar os alunos marcados. Ao clicar os alunos são associados a turma. Na tabela da direita o processo é o mesmo, ao marcar um aluno o botão com o ícone de seta em direção a esquerda é habilitado na cor vermelha, indicando que o professor irá retirar os alunos da turma, conforme visto na figura x. Ao clicar na seta os alunos são removidos. Foi-se utilizado um plugin Angular para fazer a paginação dos alunos. Caso haja uma quantidade enorme de alunos, o plugin de paginação exibe uma quantidade relativa a esses registros, no caso dessa interface exibe apenas 6 alunos, e divide os outros alunos em páginas que podem ser acessadas e, por conseguinte recuperar os alunos restantes. Lembrando, como se trata de uma tecnologia Angular, *frontend*, não há o recarregamento da página na transição das páginas de alunos. Além disso o plugin possibilita a transição das páginas sem o carregamento dos dados a cada página clicada, a navegação é fluida e rápida.



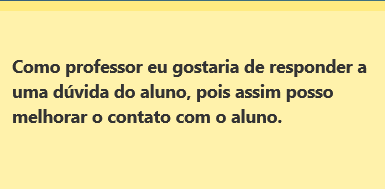
A estória apresentada na figura x representa o desejo do professor ao saber quando um aluno tem dúvida. Surge então a necessidade de avisar o professor de uma dúvida do aluno assim que ela é enviada.



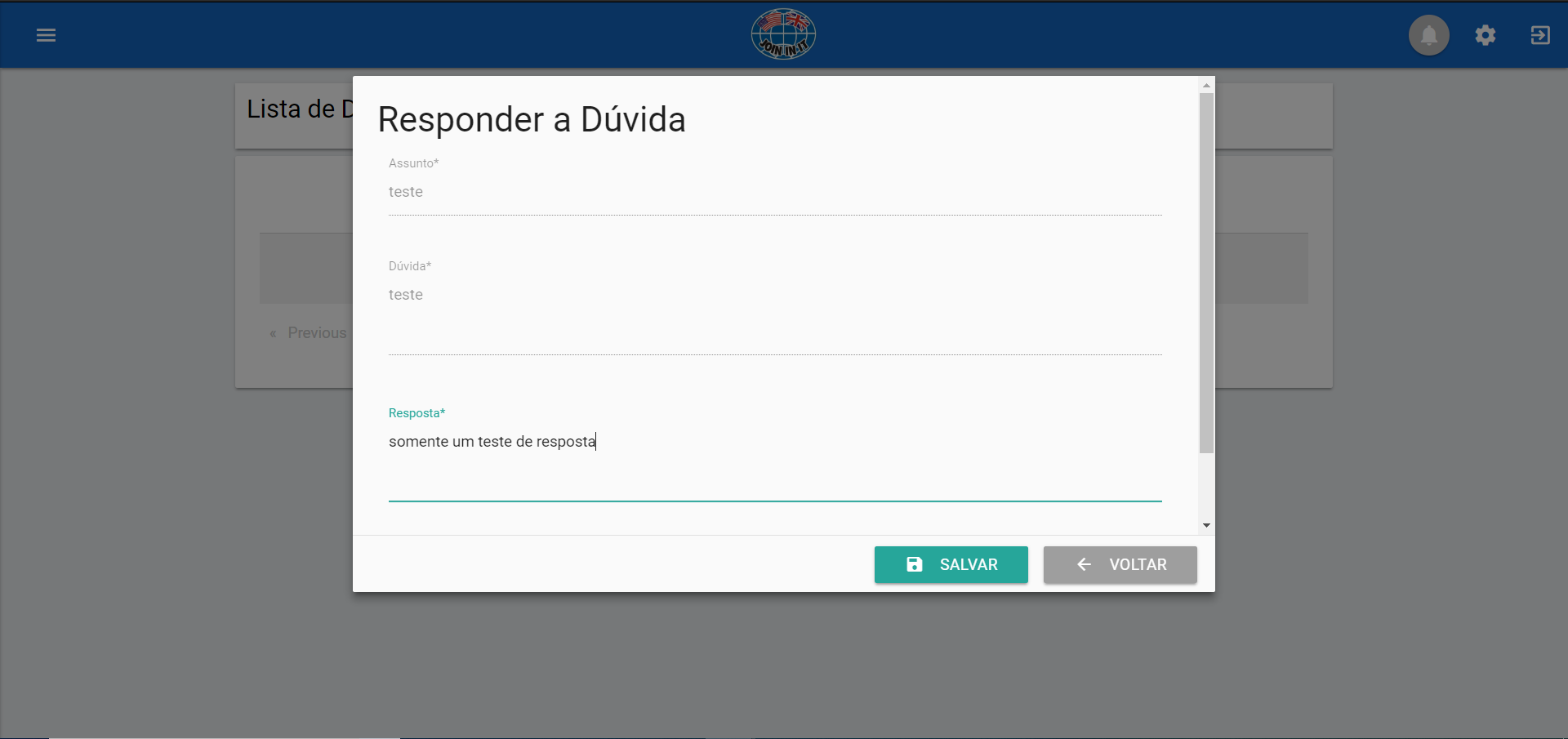
O sistema de notificações do ambiente fica responsável por notificar os professores de uma nova dúvida. Assim que a dúvida da notificação é respondida, todas as notificações são excluídas, evitando aos outros professores de responder a uma dúvida já respondida. A figura x representa essa interação, que utiliza as notificações Laravel para retornar as notificações relacionadas a uma dúvida.



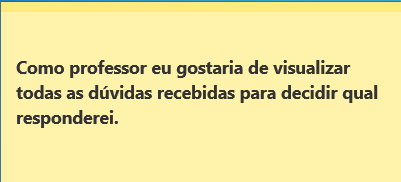
A estória definida pela figura x se trata da necessidade de interação do aluno para com o professor. Ao surgir uma dúvida o professor deve ser capaz de responde-la, e então ao ser notificado, seguir a notificação e verificar se consegue responder à pergunta. Caso contrário outro professor tem a possibilidade de responder. Então as dúvidas são enviadas a todos os professores e não somente ao professor da turma do aluno com dúvida. Isso se deu pelo fato de possibilitar agilidade no processo de resposta, já que os professores da escola detêm conhecimento e capacidade para sanar as dúvidas dos alunos.



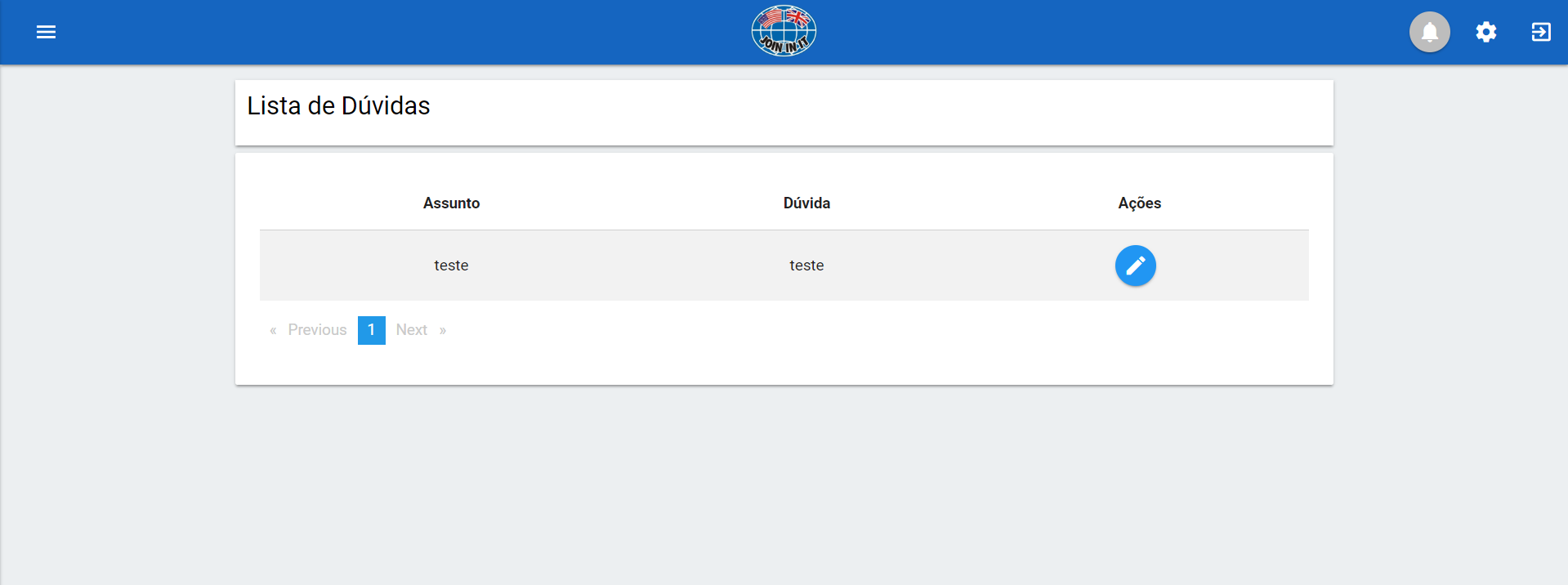
A figura x representa a interface de resposta à dúvida. Nessa interface o professor pode visualizar qual é o assunto da dúvida. E qual a dúvida em si. Assim tecer uma resposta ao questionamento do aluno. É importante ressaltar que o sistema de dúvidas não foi implementado a se comportar como chat. Em caso de uma nova dúvida, o aluno deve envia-la aos professores para retirada de dúvidas.



Por último ao professor, pode surgir a necessidade de não exatamente seguir a notificação de uma dúvida, mas verificar quais são as dúvidas geradas pelos alunos e escolher qual responder. A figura x descreve a estória que representa esse processo, ou seja, a listagem de todas as dúvidas cadastradas.



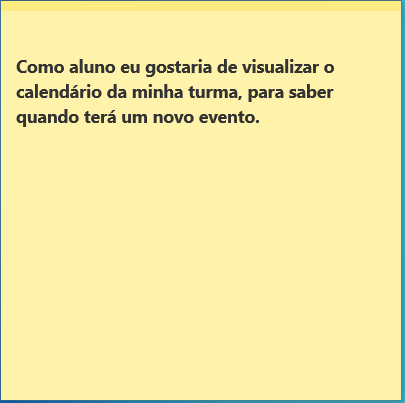
A listagem das dúvidas requisitada pela estória da figura x, pode ser vista na figura x. Nela o professor tem acesso a todas as dúvidas geradas pelos alunos até o determinado momento e pode escolher qual responder, clicando no botão com ícone de lápis. A janela que surge ao clicar no botão citado é a apresentada na figura x.



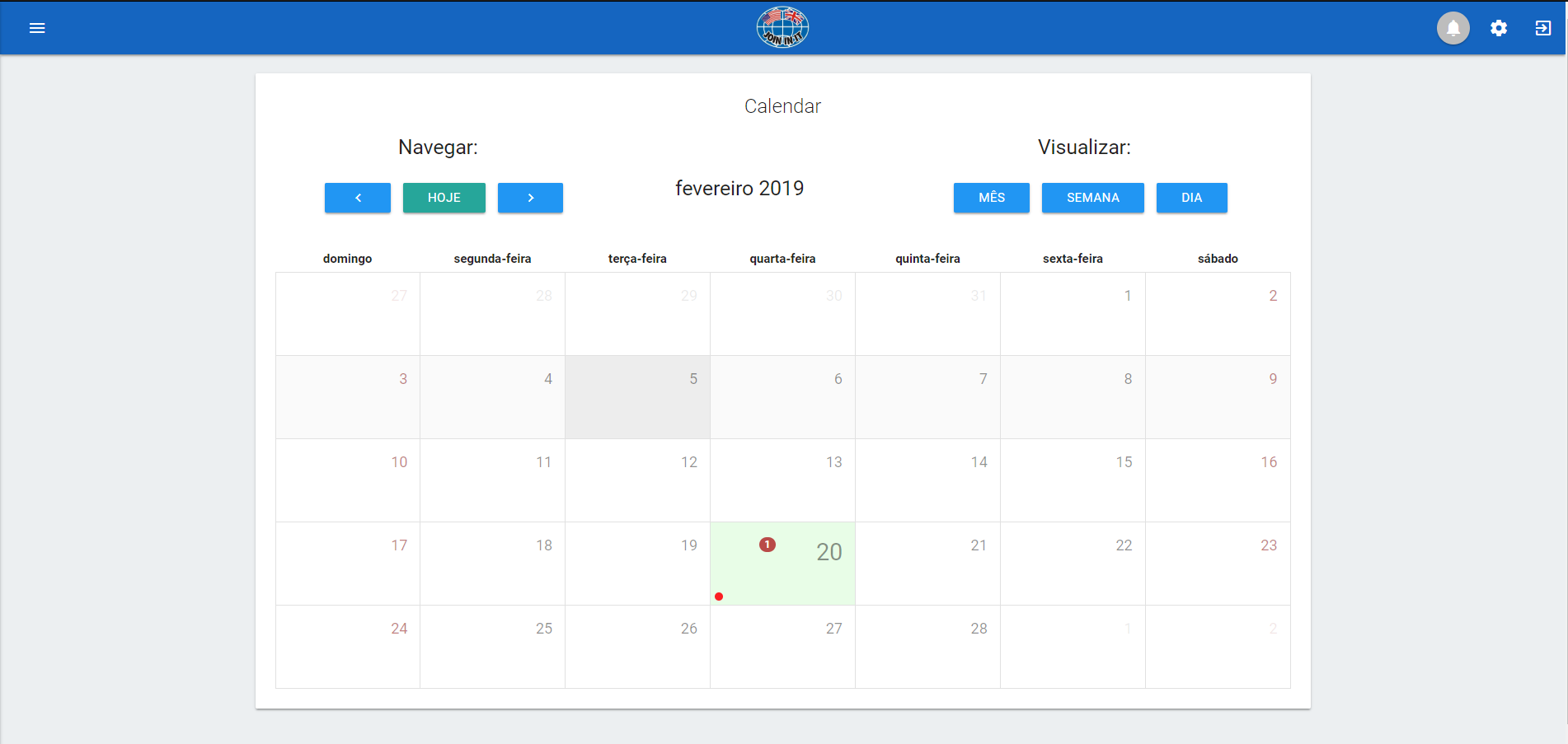
#### Estórias dos alunos

Os alunos no ambiente desempenham algumas funções, tais como visualizar os materiais dispostos pelos professores, como também enviar dúvidas aos professores, acessar o calendário, etc. Com isso foi-se desenvolvido as seguintes estórias para mapear as necessidades de utilização dos alunos.

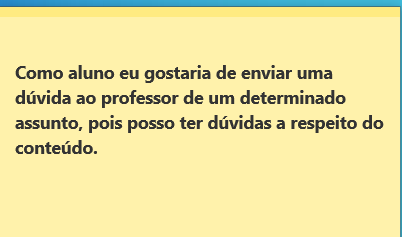
A primeira estória se trata da tela de inicialização do sistema, e a necessidade de exibição do calendário da turma, com eventos como provas ou atividades, para que o aluno fique por dentro do calendário da turma. Essa estória é definida pela figura X, e o design da interface se encontra na figura x.



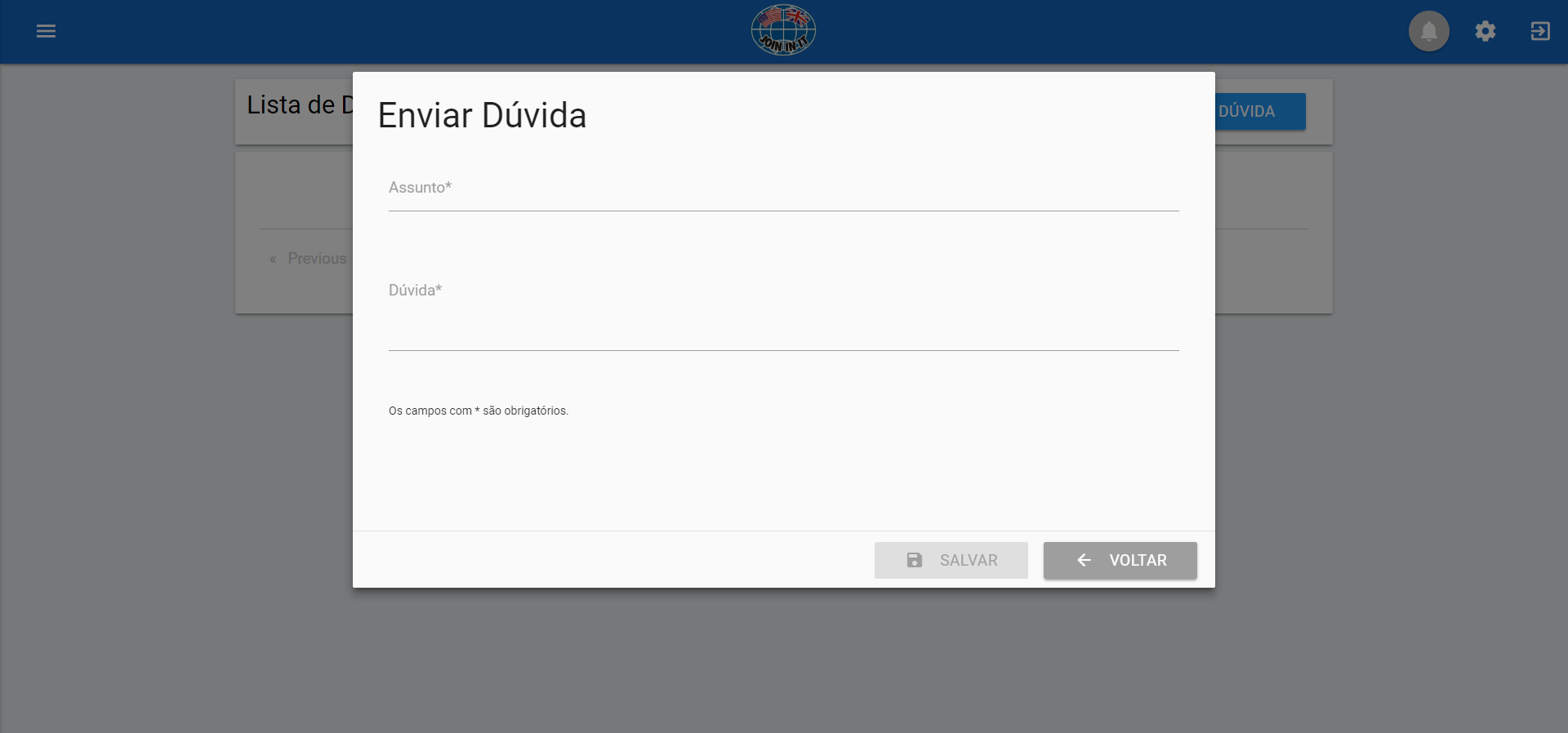
É apresentado ao aluno um calendário interativo, ao qual o aluno pode navegar pelos dias, meses ou semanas, além disso é possível visualiza-lo pelo mês, pela semana ou pelo dia. O Aluno ainda pode conferir os eventos que a escola ou o professor da sua turma cadastrou. Os eventos ficam destacados no calendário conforme a cor escolhida por quem cadastrou o evento. Caso haja mais de um evento na mesma data ou horário o calendário apresenta um contador. Ao clicar no dia em que se há eventos, a descrição dos eventos daquele dia são apresentadas.



Já a segunda estória do aluno se trata de quando ele tem uma dúvida a respeito de algum conteúdo e por isso deseja-se enviar essa dúvida para possível resposta de um professor. Essa estória é definida pela figura X, e sua interface pode ser notada na figura x.



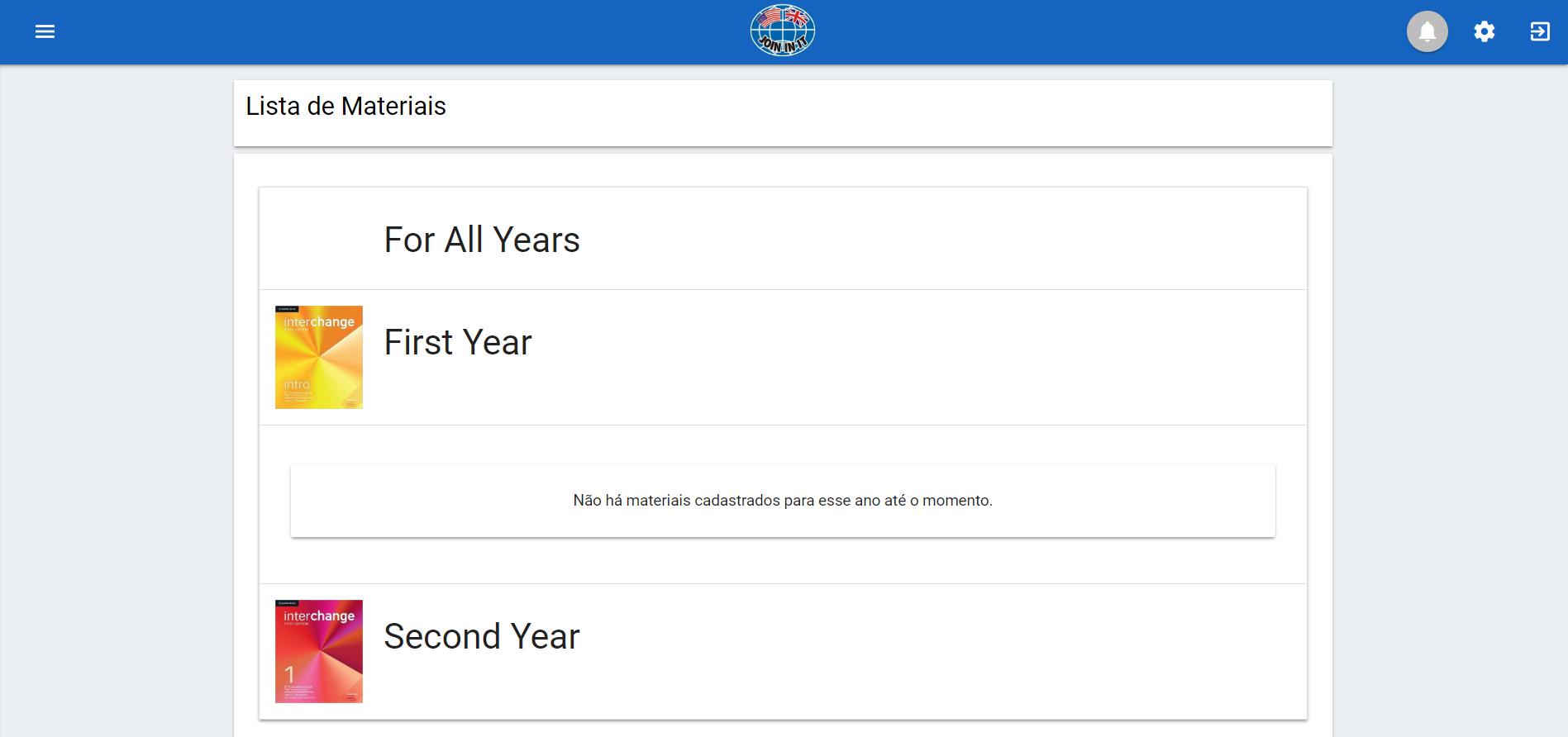
O aluno pode enviar uma dúvida a respeito de um determinado assunto. A inserção do assunto serviu para ajudar o professor a identificar sobre o que se trata a dúvida do aluno. O campo dúvida, refere-se a dúvida em si.



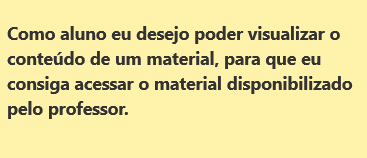
A estória definida pela figura X se trata da limitação na visualização dos materiais, já que os alunos não podem acessar materiais mais avançados de anos superiores ao ano cursado.



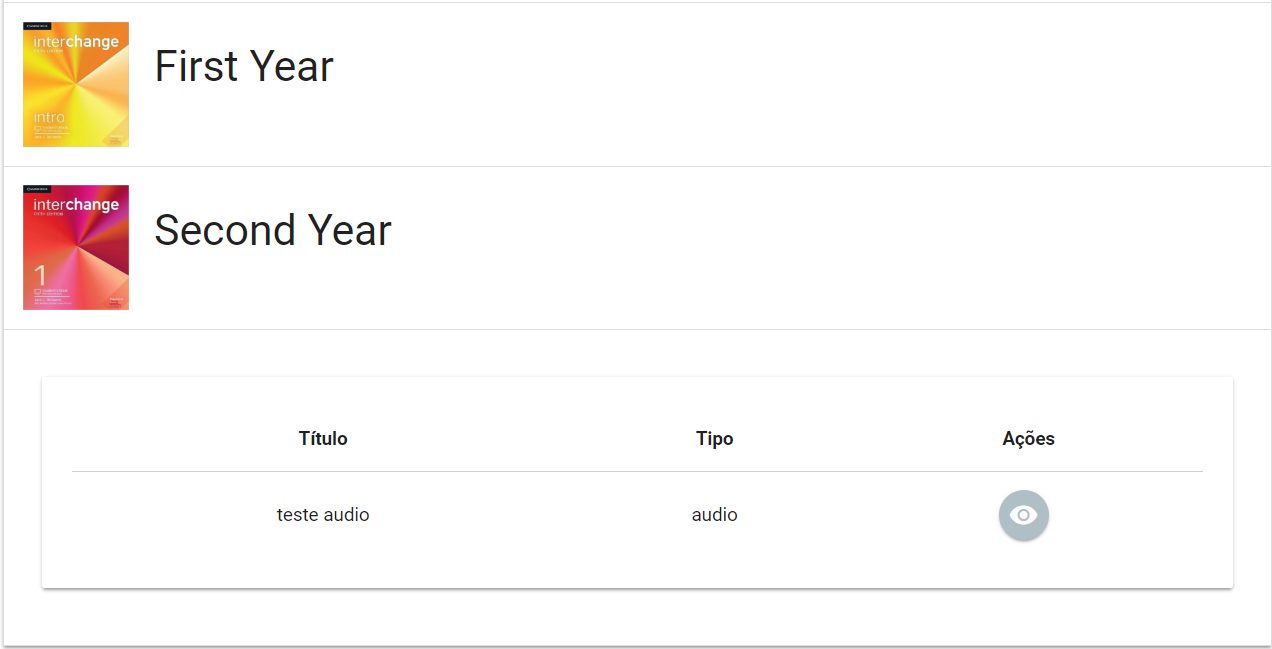
A figura X demonstra como a estória foi implementada, uma vez que o aluno faz parte do segundo ano, então a listagem dos materiais é filtrada para materiais até o ano que o aluno está cursando. Há também a possibilidade de os materiais serem disponíveis a todos, o que pode ser visto pela primeira camada chamada “*For All Years*”.



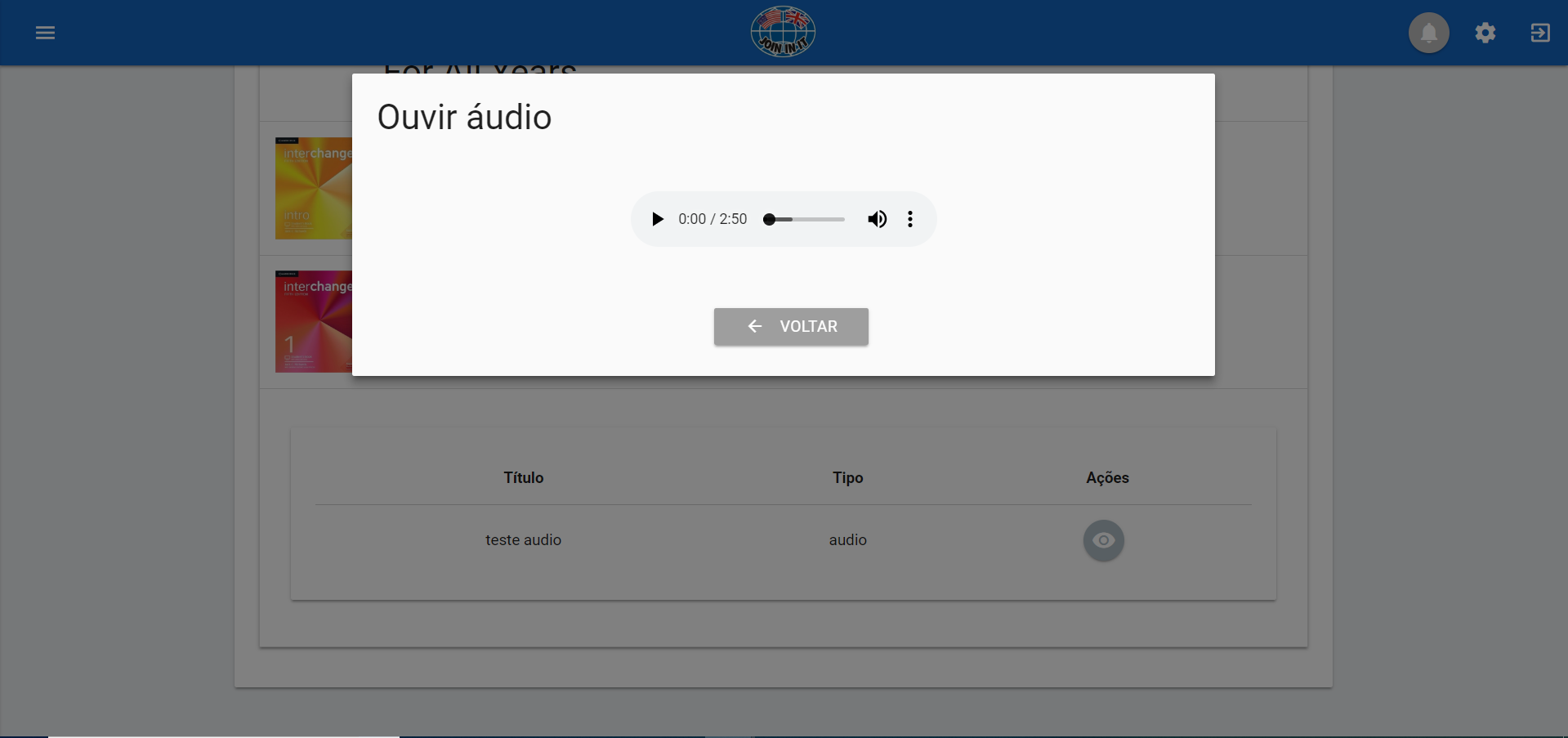
Ainda como aluno é possível que ele acesse o material cadastrado pelo professor. A figura X representa a estória que descreve esse anseio do aluno.



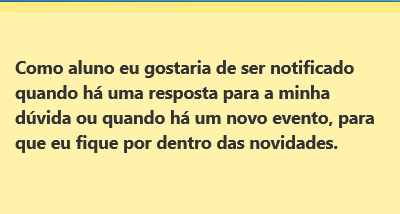
A implementação dessa estória é composta de algumas etapas. Na listagem dos materiais surge um botão com ícone de olho conforme visto na figura X. Porém ao clicar nesse botão, dependendo do tipo do material a interação pode mudar. Em caso de link o usuário será redirecionado a página referente ao link indicado.



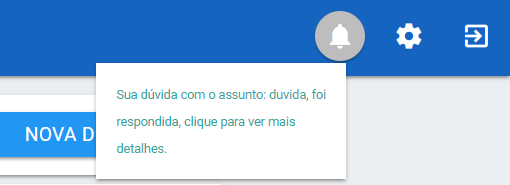
Em caso de áudio, surgirá uma tela em que o aluno pode escutar o áudio. A figura X demonstra como é essa interface de visualização de materiais de áudio pelo aluno. O Laravel conta com um sistema capaz de manipular o sistema de arquivos do servidor. Assim o cadastro de arquivos, bem como a sua recuperação é feita de maneira bem simples pelo *frontend* que somente deve identificar o caminho do arquivo no *backend*.



Já a estória definida pela figura X se trata da função de notificação do aluno a uma possível resposta do professor a uma dúvida, assim ele tem um *feedback* visual de quando a pergunta foi respondida.



Assim que o aluno clica sobre o ícone de notificações, a notificação referente a resposta da dúvida surge. Ele é informado a respeito de qual dúvida foi respondida e ao clicar sobre o texto, o aluno é direcionado para a visualização da dúvida. Assim ele pode ver a resposta dada pelo professor a sua dúvida.



#### 

A biblioteca de testes utilizada foi o PHPUnit, que já vem integrado com o Laravel. O Laravel apoia as funções nativas do PHPUnit e acrescenta algumas funcionalidades que auxiliam nos momentos de teste. Nesta subseção serão abordados alguns testes utilizados no primeiro release.

O primeiro exemplo de teste se trata do trecho de código abaixo, que compreende na classe de Teste de usuário, demonstrando a função de teste de inserção. Tem-se a utilização de dois Traits Laravel, o ‘*WithoutMiddleware’ e ‘DatabaseMigrations’*. O primeiro serve para não utilizar *middlewares* que podem impedir o acesso a determinados conteúdos para determinados tipos de usuários. O segundo acrescenta as migrações, que cria toda a base de dados no ambiente de teste. Há ainda uma função chamada ‘setUp’, que seria uma configuração inicial dos testes, nela é possível configurar o que for necessário para todos os testes antes que o teste ocorra em si. Para isso usou-se o comando artisan ‘db:seed’ que serve para ‘alimentar’ a base de dados com registros. O restante da função serve para autenticar um usuário que será utilizado em outro trecho da classe de testes de usuário.

**class** UserTest **extends** TestCase  
  
{  
 **use** WithoutMiddleware;  
 **use** DatabaseMigrations;  
 **private** $header;  
  
 **public function** setUp()  
 {  
 **parent**::*setUp*();  
 $this->artisan('db:seed');  
 $login = $this->json('POST', route('auth.login\_for\_tests'), ['username'=>'teresinhag', 'password'=>'secret']);  
  
 $this->header = ['Authorization'=>"Bearer {$login->json()['token']}"];  
 }  
  
  
 **public function** testStore()  
 {  
  
   
 $response = $this->postJson(route('user.store'), ['name'=>'Ryan','password'=>'secret', 'username'=>'test', 'role\_id'=>1], [  
  
 'Content-type' => 'application/json'  
 ]);  
 $response->assertStatus(200)  
 ->assertJson([  
 'success',  
 ]);  
  
 $this->assertDatabaseHas('users', ['name'=>"Ryan", 'username'=>'test']);  
 $this->assertDatabaseHas('role\_user', ['role\_id'=>1]);  
  
 }

Para a função ‘testStore’ que verifica a inserção de um novo usuário na base de dados, tem-se a primeira linha que recebe a resposta de uma requisição para a rota de inserção de usuários, passando os dados do usuário por meio de um *array*. Após receber essa resposta começam as asserções, que são validações feitas afim de testar um determinado comportamento esperado. Nesse caso eu espero que o status da requisição HTTP seja 200, que significa que a requisição foi feita com sucesso. Espera-se ainda receber um JavaScript Object Notation (JSON), que se trata de uma notação de objetos entendida de maneira padrão por diversas tecnologias. Espero receber um objeto contendo a palavra ‘*success*’, indicando que tudo ocorreu bem. Ainda há mais duas asserções que verificam se os dados foram salvos na base de dados.

O teste do trecho de código abaixo é responsável por verificar se é possível enviar uma notificação ao professor, salvar na base, ao enviar uma dúvida. As asserções seguem o mesmo exemplo do teste de inserção dos usuários.

**public function** testIfSendNotificationOnCreate(){  
  
 $response = $this->enviaDuvida();  
  
  
 $response->assertStatus(200);  
 $response->assertJson(['success']);  
 $this->assertDatabaseHas('notifications', ['notifiable\_id' => 2]);  
}

O próximo teste se trata da atualização de um evento criado por um professor, utiliza-se a função para criar um evento de um professor. Após criar o evento e salva-lo na base de dados recebe-se a resposta da rota de atualização pela função ‘json’ (passando como parâmetros o verbo HTPP que nesse caso foi o PUT, a rota, os dados e o cabeçalho, que se trata do token que verifica que o usuário se autenticou no ambiente. As funções de asserção são praticamente as mesmas, o que irá mudar é a ultima verificação que verifica se tem algum dado faltando ou não existente na base. Isso se dá pelo fato que o processo de atualizar os dados antigos do evento não devem existir. Somente devem existir na base os dados atuais que foram atualizados. Então faz-se duas verificações, se existe o dado novo e se o dado antigo não existe mais.

**public function** testUpdate(){  
 $this->createGroup();  
 $this->createEventTeacher();  
 $data2 = ['title'=>'teste edição', 'date'=>'2019-02-03 18:32:24', 'color'=>'#e66465', 'public'=>0];  
  
 $response= $this->json('PUT', route('event.update', 1), $data2, $this->header\_teacher);  
 $response->assertStatus(200);  
 $response->assertJson(['success']);  
 $this->assertDatabaseHas('events', $data2);  
 $this->assertDatabaseMissing('events', ['title'=>'teste']);  
  
}

## 



# Referências

ABREU, L. **TypeScript:** O JavasCript Moderno para Criação de Aplicações. Lisboa: FCA – Editora de Informática. 2017.

ALVES, J. R. M. *et al.* **Educação a Distância:** o estado da arte. São Paulo: Pearson Education do Brasil, v. 1, 2009.

BABBEL. **Preços**. 2018. Disponível em: <https://home.babbel.com/prices>. Acesso em: 23 ago. 2018.

BACICH, L. *et al*. **Ensino Híbrido:** Personalização e tecnologia na educação. Porto Alegre: Penso, 2015.

BARANAUSKAS, M. C. C.; ROCHA, H. V. D. **Design e Avaliação de Interfaces Humano-Computador**. Campinas: UNIVERSIDADE ESTADUAL DE CAMPINAS, 2003. Disponível em: <https://www.nied.unicamp.br/biblioteca/design-e-avaliacao-de-interfaces-humano-computador/ > Acesso em: 22 set. 2018.

BENTO, E. J. **Desenvolvimento Web com PHP e MySQL**. São Paulo: Casa do Código, 2013.

CAELUM. **Desenvolvimento Web com HTML, CSS e JavaScript**. São Paulo: Caelum ensino e inovação, 2018. Disponível em: <https://www.caelum.com.br/download/caelum-html-css-javascript.pdf>. Acesso em: 07 out. 2018.

CAMPOS, A. L. N. **Modelagem de Processos com BPMN**. 2. ed. Rio de Janeiro: Brasport, 2014.

CARVALHO, V. **MySQL:** Comece com o principal banco de dados open source do mercado. São Paulo: Casa do Código, 2015.

CCAA. **Espaço CCAA Aluno**. sd. Disponível em: <https://www.ccaa.com.br/espacoccaa/conteudos/>. Acesso em: 23 ago. 2018.

CROCKFORD, D. **JavaScript:** The Good Parts. Sebastopol: O'Reilly, 2008.

DIAS, D. D. S.; SILVA, M. F. D. **Como escrever uma monografia:** Manual de elaboração com exemplos e exercícios. Rio de Janeiro: Atlas, 2010.

DUOLINGO. **Aprenda idiomas de graça. Para sempre**, sd. Disponível em: <https://pt.duolingo.com/>. Acesso em: 23 ago. 2018.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 6. ed. São Paulo: Pearson Education, 2011.

FERREIRA, A. B. D. H. **Mini Aurélio Século XXI:** O minidicionário da língua portuguesa. 5. ed. Rio de Janeiro: Nova Fronteira S.A, 2001.

GIL, A. C. **Como Elaborar Projeos de Pesquisa**. 4. ed. São Paulo: Atlas, 2002.

GOOGLE. **Angular**, 2019. Disponível em: <https://angular.io/>. Acesso em: 08 fev. 2019.

GUEDES, T. **Crie aplicações com Angular**: o novo Framework do Google. São Paulo: Casa do Código, 2017.

HINZ, M. A. M. **Um estudo descritivo de novos algoritmos de criptografia.** 2000. 58f. Monografia (Bacharel em Informática) - Universidade Federal de Pelotas, Pelotas, 2000. Disponível em: < http://www.jabour.com.br/ufjf/apa/Mono-MarcoAntonio.pdf >. Acesso em: 5 out. 2018.

HIRAMA, K. **Engenharia de Software:** Qualidade e Produtividade com Tecnologia. Rio de Janeiro: Elsevier, 2011.

INSTITUTE OF ELETRICAL AND ELETRONICS ENGINEERS. **IEEE Std 610.12-1990:** IEEE Standard Glossary of Software Engineering Terminology. New York: [s.n.], 1990. 84 p. Disponível em: < http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE\_SoftwareEngGlossary.pdf>. Acesso em: 9 set. 2018.

LOCKHART, J. **PHP Moderno**. São Paulo: Novatec, 2015.

MARCONI, M. D. A.; LAKATOS, E. M. **Fundamentos de metodologia científica**. 5. ed. São Paulo: Atlas, 2003.

MASSÉ, M. **REST API:** Design RuleBook. Sebastopol: O'Reilly, 2012.

MCFARLAND, D. S. **CSS3:** the missing manual. 3. ed. Sebastopol: O'Reilly, 2013.

MELO NETO, J. A. D. *et al.* **Educação a distância:** o estado da arte. São Paulo: Pearson Education do Brasil, v. 2, 2012.

MORENO, E. D.; PEREIRA, F. D.; CHIARAMONTE, R. B. **Criptografia em Hardware e Software**. São Paulo: Novatec, 2005.

OTWELL, T. **Encryption.** 2018. Disponível em: <https://laravel.com/docs/5.7/encryption>. Acesso em: 05 out. 2018.

PHP. **Modelo de Armazenamento Criptografado.** 2018a. Disponível em: <https://secure.php.net/manual/pt\_BR/security.database.storage.php>. Acesso em: 05 out. 2018.

PHP. **O que é o PHP?**, 2018b. Disponível em: <https://secure.php.net/manual/pt\_BR/intro-whatis.php>. Acesso em: 30 set. 2018.

PRESSMAN, R. S. **Engenharia de Software:** Uma abordagem Profissional. 7. ed. Porto Alegre: Bookman, 2011.

ROBBINS, J. N. **HTML5:** Pocket Reference. 5. ed. Sebastopol: O'Reilly, 2013.

SANDHU, R. S. Role-based Access Control. In: **Advances in Computers.** Fairfax: Academic Press, v. 46, 1998. p. 237-286. Disponível em: <http://www.profsandhu.com/articles/advcom/adv\_comp\_rbac.pdf>. Acesso em: 5 out. 2018.

SANTOS, L. dos. **Como escrever boas histórias de usuário (User Stories).** 2017. Disponível em: <https://medium.com/vertice/como-escrever-boas-users-stories-hist%C3%B3rias-de-usu%C3%A1rios-b29c75043fac>. Acesso em: 17 fev. 2019.

SEVERINO, A. J. **Metodologia de trabalho científico**. 22. ed. São Paulo: Cortez, 2002.

SILBERCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 3. ed. São Paulo: Pearson Education, 1999.

SILVER, B. **BPMN Method and Style:** with Bpmn Implementer's Guide. 2. ed. Altadena: Cody-Cassidy Press, 2017.

SKLAR, D. **Aprendendo PHP:** Introdução amigável à linguagem mais popular da WEB. São Paulo: Novatec, 2016.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

STAUFFER, M. **Desenvolvendo com Laravel:** Um Framework para construção de aplicativos PHP modernos. São Paulo: Novatec, 2017.

TELES, V. M. **Extreme Programming:** Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. 2. ed. São Paulo: Novatec, 2014.

WAZLAWICK, R. S. **Metodologia de Pesquisa para ciência da Computação**. Rio de Janeiro: Elsevier, 2009.

WIZARD. **Experiências Wizard**, 2017a. Disponível em: <http://www.wizard.com.br/experiencias-wizard/>. Acesso em: 23 ago. 2018.

WIZARD. **Sobre a Wizard**, 2017b. Disponível em: <http://www.wizard.com.br/sobre-wizard/>. Acesso em: 23 ago. 2018.

ZAPATER, M.; SUZUKI, R. **Segurança da Informação:** Um diferencial determinante na competitividade das corporações. Promon Business & Tecnology Review. Rio de Janeiro, p. 28. 2005. Disponível em: <http://www.teleco.com.br/promon/pbtr/Seguranca\_4WEB.pdf>. Acesso em: 12 out. 2018.

# Apendice A - carta de pedido de permissão para uso de informações da escola International language center





