UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LUCAS BATISTELLA DE CÓL

SISTEMA PARA ACOMPANHAMENTO DE REALIZAÇÃO DE ESTÁGIOS CURRICULARES

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2016

LUCAS BATISTELLA DE CÓL

SISTEMA PARA ACOMPANHAMENTO DE REALIZAÇÃO DE ESTÁGIOS CURRICULARES

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2016

ATA Nº: 282

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO LUCAS BATISTELLA DE COL

Às 13:30 hrs do dia 24 de junho de 2016, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Andréia Scariot Beulke (Convidada) e Vinicius Pegorini (Convidado), para avaliar o Trabalho de Diplomação do aluno Lucas Batistella de Col, matrícula 1270524, sob o título Sistema para Acompanhamento de Realização de Estágios Curriculares; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho APROVADO. Às 14:00 hrs foi encerrada a sessão.

_____
Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

_____
Profa. Andréia Scariot Beulke, Esp.
Convidada

_____
Prof. Vinicius Pegorini, M.Sc.
Convidado

_____
Profa. Eliane Maria de Bortoli Fávero, M.Sc
Coordenadora do Trabalho de Diplomação

_____
Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

DE COL, Lucas Battistela. Sistema para acompanhamento de realização de estágios curriculares. 2016. 51f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2016.

A atividade de estágio curricular ocorre na modalidade não obrigatória e obrigatória. Essa última é validada como parte da carga horária do curso e para tanto o aluno apresenta um relatório que é defendido perante uma banca. A atividade de estágio é coordenada por um professor que é denominado como responsável. Esse professor faz o gerenciamento dos aspectos relacionados ao curso, ou seja, controle dos documentos de acompanhamento e da apresentação do relatório final. A parte documental é de responsabilidade de um Departamento específico da Universidade. Como forma de auxiliar no gerenciamento dessas atividades foi implementado um aplicativo *web* para ser utilizado pelo professor responsável pelo estágio. Para o desenvolvimento do aplicativo foram utilizadas tecnologias *web* consideradas recentes pelo tempo de disponibilização de mercado. Essas tecnologias visam fornecer agilidade na implementação, responsividade e maior facilidade de interação para o usuário, além de reduzir o tráfego entre cliente e servidor, pela possibilidade de implementação de processamento no cliente. Assim, o desenvolvimento desse trabalho foi, também, uma grande oportunidade de aprendizado de tecnologias.

Palavras-chave: AngularJS. Rich Internet Aplication. Controle de estágios curriculares.

ABSTRACT

DE COL, Lucas Battistela. System for monitoring internships. 2016. 51f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2016.

The traineeship activity occurs in not mandatory and binding mode. The latter is validated as part of the workload of the course and for both the student presents a report that is defended before a panel. The stage activity is coordinated by a teacher who is named as responsible. This teacher makes the management of aspects related to the course, ie control of accompanying documents and the presentation of the final report. The documentary part is the responsibility of a specific department of the University. a web application to be used by the teacher responsible for the stage as a way to help manage these activities was implemented. For application development web technologies were used considered recent by market availability of time. These technologies are intended to provide flexibility in implementation, responsiveness and ease of interaction to the users and reduce the traffic between client and server, by the possibility of processing on the client implementation. Thus, the development of this work was also a great opportunity of learning technologies.

Palavras-chave: AngularJS. Rich Internet Aplication. Control traineeships.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGEM DE CÓDIGO

LISTA DE SIGLAS

HTML	*HyperText Markup Language*
MVC	*Model-View-Controller*
RIA	*Rich Internet Applications*
RA	Registro Acadêmico
UTFPPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, o objetivo e a justificativa deste trabalho. O capítulo é finalizado com a apresentação dos capítulos subsequentes que compõem o texto.

1.1 CONSIDERAÇÕES INICIAIS

O estágio curricular, como previsto na Lei nº 11.788, de 25 de setembro de 2008, é ato educativo escolar supervisionado, que visa à preparação para o trabalho de educandos que estejam frequentando o ensino regular em instituições de educação superior, profissional, especial, de ensino médio e dos anos finais do ensino fundamental quando na modalidade profissional da educação de jovens e adultos (UNIVERSIDADE…, 2014, p. 1). Ainda de acordo com o regulamente de estágios da Universidade Tecnológica Federal do Paraná (UTFPR) (UNIVERSIDADE…, 2014, p. 1), "o estágio visa ao aprendizado de competências próprias da atividade profissional e à contextualização curricular, objetivando o desenvolvimento do educando para a vida cidadã e para o trabalho".

O estágio curricular, de acordo com as especificidades de cada curso e como estabelecido no projeto pedagógico do curso, pode ocorrer nas modalidades de estágio obrigatório e não obrigatório. O estágio obrigatório possui carga horária prevista no projeto do curso e essa quantidade de horas é incorporada no cômputo da carga horária do curso, como previsto em sua matriz curricular.

Cada curso da UTFPR possui um professor responsável pela atividade de estágio que, seja na modalidade obrigatório ou não obrigatório, precisa ser documentado para que possa ser realizado. Além disso, são gerados relatórios de acompanhamento e de avaliação final. Quando o estágio é realizado em uma empresa, os relatórios são gerados pelo supervisor (a pessoa da empresa na qual o estágio está sendo realizado que é responsável pelo estagiário), pelo professor orientador e pelo próprio aluno que está realizando o estágio.

Há os documentos vinculados aos estágios há os que são periódicos, como os relatórios semestrais e que são dependentes de eventos específicos, como o relatório final de estágio e os documentos para o início de realização do estágio.

Um sistema computacional que auxilie o professor responsável pela atividade de estágio no controle desses documentos e relatórios facilita as tarefas de acompanhamento dos

alunos em estágio e dos documentos necessários. Considerando esse contexto e necessidade verificou-se a oportunidade de implementar um sistema para *web* que pudesse auxiliar no controle dos documentos e relatórios vinculados às atividades de estágio realizadas pelos alunos da UTFPR.

1.2 OBJETIVOS

A seguir estão o objetivo geral e os objetivos específicos deste trabalho.

1.2.1 Objetivo Geral

Desenvolver um sistema para controle dos documentos e relatórios relacionados à atividade de estágio curricular.

1.2.2 Objetivos Específicos

Facilitar o controle dos relatórios necessários na atividade de estágio curricular obrigatório e não obrigatório.

Fornecer ao professor responsável pela atividade de estágio um mecanismo para gerenciamento dos dados relacionados ao estágio curricular.

1.3 JUSTIFICATIVA

O sistema desenvolvido como resultado deste trabalho se justificativa por ser uma alternativa de controle dos documentos envolvidos e relatórios necessários para a realização do estágio curricular. Por meio do sistema o professor responsável por esse tipo de atividade terá acesso facilitado aos dados de relatórios faltantes, prazos para os relatórios e envio de comunicação ao supervisor, orientador e aluno de prazos de relatórios.

O desenvolvimento de um sistema *web* facilita o acesso pela rede e o envio de mensagens por *email*, embora essas não sejam características exclusivas desse tipo de sistema. As tecnologias utilizadas no desenvolvimento visam fornecer recursos de interface que facilitar a interação do usuário e agilizar a realização das tarefas implementadas no sistema.

1.4 ESTRUTURA DO TRABALHO

A sequência deste texto está organizada em capítulos. No capítulo 2 é apresentado o referencial teórico que fundamenta o tipo de sistema desenvolvido: um aplicativo para web. As tecnologias e as ferramentas utilizadas para a modelagem e a implementação do sistema são apresentadas no Capítulo 3. O Capítulo 4 contém o resultado da realização do trabalho que é um sistema para o controle dos relatórios e documentos de estágio supervisionado. Por fim, estão as considerações finais seguidas das referências utilizadas no texto.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico que fundamenta a proposta deste trabalho de pesquisa.

2.1 APLICAÇÕES WEB E DE INTERFACE RICA

A *HyperText Markup Language* (HTML) foi originalmente criada como uma linguagem de marcação de hipertexto baseada na Internet que os navegadores web podem usar para interpretar e compor páginas *web* estáticas (PAVLIĆ; PAVLIĆ; JOVANOVIĆ, 2012). HTML não permite a atualização parcial de página, isto é, somente a atualização de áreas ou componentes específicos. Uma maneira de tornar isso possível é pelo uso de programação com JavaScript que permite a manipulação de elementos específicos de uma página *web*.

A organização das aplicações *web* em camadas define uma arquitetura em três camadas (PAVLIĆ; PAVLIĆ; JOVANOVIĆ, 2012):

a) camada de banco de dados – servidor de banco de dados;

b) camada de aplicação – servidor de aplicação;

c) camada de apresentação – computador do usuário final, ou mais especificamente, o software sendo usado para apresentar a aplicação.

Linguagens de programação como PHP, ASP e Java (agregada por *frameworks* e bibliotecas) podem ser utilizadas para gerar páginas HTML dinâmicas. A junção de Ajax (JavaScript API) com HTML torna as aplicações *web* tradicionais mais dinâmicas.

Nos últimos anos a *web* tem ser tornado a plataforma de referência para o desenvolvimento de todos os tipos de solução integrada de negócio (BOZZON et al., 2006). Esses autores ressaltam que devido a crescente complexidade dessas aplicações, as tecnologias *web* estão começando a apresentar limites de interação e usabilidade. Isso em decorrência que a experiência do usuário com aplicações *web* não é comparável com a interface das aplicações *desktop*, a responsividade é baixa devido a sobrecarga de rede, a quantidade de acessos ao servidor é elevada e o uso *offline* (desconectado da Internet) não é possível (DRIVER; VALDES; PHIFER, 2005). Para Duhl (2003), as *Rich Internet Applications* (RIAs) tem sido propostas como uma resposta para esses inconvenientes.

A necessidade de atender a crescente demanda por melhores recursos de interação com aplicações permitiu o desenvolvimento de tecnologias - o que inclui linguagens de programação do lado servidor e cliente, *frameworks* – que levaram o desenvolvimento das aplicações denominadas Internet Ricas, as *Rich Internet Applications (*RIA).

Os usuários de aplicações categorizadas como RIA utilizam uma tecnologia de interação na interface do cliente que provê aos usuários uma melhor experiência com a Internet e a rede de computadores (BI-FENG, 2011).

A Figura 1 apresenta o modelo básico de comunicação de uma RIA. De acordo com esse modelo, há um servidor que permite executar a aplicação na Internet e com múltiplas plataformas. No lado cliente há um programa especial que a partir da descrição da interface gráfica do usuário e eventos da janela obtidos do servidor compõem a interface no cliente. Esse programa é normalmente chamado de *thin client* (PAVLIĆ; PAVLIĆ; JOVANOVIĆ, 2012).

Figura 1 – Modelo de comunicação de uma RIA
Fonte: Traduzido de Pavlić, Pavlić e Jovanović (2012, p. 1367).

As tecnologias RIA suportam armazenamento no lado cliente em uma forma que depende de tecnologias e dispositivos específicos. Por exemplo, a máquina pode localmente armazenar os dados do carrinho de compras em uma aplicação de comércio eletrônico enquanto o usuário está realizando o processo de compra. Nas aplicações *web* tradicionais, o servidor é responsável pela lógica de negócio da aplicação. As tecnologias RIA permitem mover parte da computação para o cliente (FRATERNALI; ROSSI; SÁNCHEZ-FIGUEROA, 2010). Esses autores ressaltam que as aplicações *web* tradicionais são máquinas de requisição e resposta, na qual o servidor envia informação somente em resposta a uma requisição do cliente. E nas RIAs, tanto o cliente quanto o servidor podem iniciar uma comunicação, elementos de programa no cliente permanecem prontos para receber e executar comandos

assíncronos do servidor. Essa bidirecionalidade elimina muitas idas e vindas de comunicação de dados desnecessárias que são típicas de aplicações *thin-client*.

As RIAs combinam os benefícios do modelo de distribuição da *web* com multimídia de alta interatividade disponível nas aplicações *desktop* (LINAJE; PRECIADO; SÁNCHEZ-FIGUEROA, 2007).

2.2 ESTÁGIO CURRICULAR

A Lei No. 11.788 de 25/09/2008 (BRASIL, 2008) em seu Art. 1º caracteriza o estágio como um ato educativo escolar supervisionado, desenvolvido no ambiente de trabalho, que visa à preparação para o trabalho dos alunos que estejam frequentando o ensino regular em instituições de educação superior, profissional, de ensino médio, da educação especial e dos anos finais do ensino fundamental, na modalidade profissional da educação de jovens e adultos. De acordo com o § 2° desse mesmo artigo, o estágio visa ao aprendizado de competências próprias da atividade profissional e à contextualização curricular, objetivando o desenvolvimento do educando para a vida cidadã e para o trabalho.

O estágio é categorizado como obrigatório ou não-obrigatório (BRASIL, 2008):

a) O estágio obrigatório é definido no projeto pedagógico do curso e a sua carga horária é requisito para aprovação e obtenção de diploma.

b) O estágio não-obrigatório é desenvolvido como atividade opcional que é acrescentada à carga horária regular e obrigatória do curso.

O estágio deverá ter a sua realização acompanhada de um professor orientador da instituição de ensino. O aluno que está realizando estágio deve apresentar periodicamente, em prazo não superior a 6 (seis) meses, relatório das atividades que realiza como estágio.

O regulamento dos estágios curriculares supervisionados da UTFPR (UNIVERSIDADE…, 2014) define que essa é uma atividade prevista nos projetos pedagógicos dos cursos e tem como objetivos:

a) Facilitar a inserção do estudante no mercado do trabalho;

b) Promover a articulação da Universidade com o mundo do trabalho;

c) Facilitar a adaptação social e psicológica do estudante à sua futura atividade profissional.

d) Complementar as competências e habilidades previstas no perfil do egresso de cada curso.

Esse mesmo regulamento em seu Art. 32 determina que compete à unidade cedente de estágio, a instituição que fornece o estágio, por exemplo:

a) Indicar funcionário de seu quadro de pessoal, com formação ou experiência profissional na área de conhecimento desenvolvida no curso do estagiário, para atuar como supervisor de estágio;

b) Verificar e acompanhar a assiduidade do estagiário, inclusive o controle do horário por meio do registro de frequência;

c) Receber visita do professor orientador de estágio;

São obrigações do professor responsável pela atividade de estágio, definidas no Art. 34 do Regulamento dos estágios curriculares supervisionados da UTFPR (UNIVERSIDADE…, 2014).

a) Aprovar o plano de estágio apresentado pelo estudante e estabelecer cronograma para entrega dos relatórios;

b) Efetuar a matrícula do estudante na disciplina/unidade curricular de estágio curricular obrigatório imediatamente após a assinatura completa dos termos de concessão de estágio.

c) Orientar quanto à sistemática adotada pela respectiva coordenação para a escolha do professor orientador de estágio;

d) organizar o evento de avaliação de estágio curricular obrigatório, quando houver;

e) Fixar e divulgar datas e horários dos eventos de avaliação de estágio curricular obrigatório para avaliação das atividades desenvolvidas pelos estudantes concluintes do estágio, quando houver;

f) Realizar a avaliação final do estagiário e efetuar o lançamento da nota junto ao Departamento de Registros Acadêmicos do Câmpus da UTFPR;

g) Receber, analisar e aprovar (ou não aprovar) pedidos de validação da disciplina/unidade curricular Estágio Curricular Obrigatório;

h) Supervisionar as atividades de acompanhamento dos estágios não obrigatórios;

i) Divulgar este regulamento junto aos estudantes.

As incumbências do professor orientador de estágio são, de acordo com o Art. 35 do Regulamento dos estágios curriculares supervisionados da UTFPR (UNIVERSIDADE…, 2014):

a) Orientar o estagiário, durante o período de realização do estágio, por meio de acompanhamento direto (com visitas ao local de realização da atividade de estágio) ou indireto (por meio de recursos tecnológicos);

b) Exigir do educando a apresentação dos relatórios de estágio, conforme cronograma estabelecido no plano de estágio;

c) acompanhar a elaboração do relatório de estágio, quando de estágio curricular obrigatório;

d) Receber, avaliar e arquivar no sistema integrado de estágio, os relatórios parciais de estágio e o relatório final;

e) Postar, no sistema integrado de estágio, os relatórios de estágio;

f) No caso do estágio curricular obrigatório, acompanhar o estagiário no evento de avaliação de estágio, quando houver, ou participar da sistemática de avaliação, definida pelo curso;

g) Elaborar relatório circunstanciado da situação encontrada e discorrer sobre as atividades do estagiário na unidade concedente de estágio.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados para a modelagem e a implementação do sistema.

3.1 MATERIAIS

Os materiais, constantes no Quadro 1, estão relacionados às tecnologias e às ferramentas para a documentação da modelagem e a implementação do sistema. Na implementação, além, da linguagem de programação e o ambiente para utilizá-la, estão os recursos para desenvolver a interface e o banco de dados, com a ferramenta que permite a administração desse banco.

Ferramenta / Tecnologia	**Versão**	**Referência**	**Finalidade**
Astah Community	7.0	http://astah.net/editions/community	Modelagem do sistema.
Toad Data Modeler	5.4	http://software.dell.com/products/toad-data-modeler/	Modelagem do diagrama de entidades e relacionamentos do banco de dados.
Java	7	http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html	Linguagem de programação.
Visual Studio Code	1.2	https://www.jetbrains.com/webstorm/download/	Ambiente de Desenvolvimento (*front end*).
JavaScript	-	https://www.javascript.com/	Linguagem para interface.
Html5	5	http://www.w3.org/TR/html5/	Linguagem para interface.
Eclipse Luna	4.4.2	https://eclipse.org/downloads/packages/release/luna/sr2	Ambiente de desenvolvimento (*back end*).
Postgre	9.1	https://www.postgresql.org/download/	Banco de dados.
Tomcat	7	http://tomcat.apache.org/download-70.cgi	Servidor de aplicação.
Hibernate	5.0.2.Final	http://hibernate.org/orm/downloads/	Efetuar o mapeamento objeto-relacional e a persistência dos dados.
AngularJS Framework	1.4.7	https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js	*Framewok* estrutural para desenvolvimento de aplicações *web*.
Bootstrap	-	http://getbootstrap.com/	*Framework front end*
Yeoman	-	http://yeoman.io/	Gerador de código para *front end.*

JHipster	3.4.0	https://jhipster.github.io/installation/	Gerador Yeoman usado para criar projetos Spring Boot + AngularJS
JDLStudio	1	http://jhipster.github.io/jdl-studio/	Ferramenta para desenhar diagramas UML
LiquiBase	3.5.1	http://www.liquibase.org/download/	Gerenciador de alterações de banco.
Spring Bot	2.4.0	http://projects.spring.io/spring-boot/	Reaproveita Tecnologias existentes para facilitar o desenvolvimento *back end*
Spring MVC	2.4.0	http://projects.spring.io/spring-boot/	*Front end controller*, solução completa para REST.
Spring Secutiry	4.1.0	http://projects.spring.io/spring-boot/	Framework de segurança para aplicações Java
OAuth2	2.0.9	http://oauth.net/2/	Framework de segurança para a proteção de aplicações e API's REST
Gulp	3.9.1	https://www.npmjs.com/package/gulp-download	Automatizador de tarefas *front end*
JUnit		http://junit.org/junit4/	Automatizador de testes unitários
NPM	2.15.1	https://nodejs.org/en/download/	Gerenciador de pacotes usado para a instalação de módulos node
Bower	1.7.9	https://bower.io/	Gerenciador de pacotes *front end*
Stimulsoft JS	1	https://www.stimulsoft.com/en/downloads/reports-js	Ferramenta de Relatórios para JavaScript

Quadro 1 – Tecnologias e ferramentas utilizadas na modelagem e na implementação do aplicativo

A seguir é apresentada uma pequena descrição da tecnologia AngularJS por ser a tecnologia enfatizada no desenvolvimento deste trabalho e relativamente nova, se comparada às demais.

O AngularJS é um framework JavaScript para a criação de páginas *web* front-end. Entre os seus diversos benefícios, destaca-se o uso de Data Binding e do padrão *Model-View-Controller* (MVC) (DEVMEDIA, 2016). O Angular JS permite ampliar a lógica de negócio (da aplicação) no cliente, evitando que execuções desnecessárias sejam enviadas para o servidor que fica encarregado de funcionalidades como persistir dados e fazer integrações ou validações que dependam de outros componentes (SAMPAIO, 2016).

AngularJS permite ampliar a HTML convencional e adicionar *views* dinâmicas de modo a criar um vocabulário próprio. Desenvolvido e mantido pelo Google, o projeto segue o padrão MVC, com as *views* sendo especificadas usando HTML + AngularJS da própria

linguagem de *template*; os modelos e controladores são especificados por meio de objetos e funções do JavaScript (SAMPAIO, 2016).

Para usar o AngularJS é necessário criar uma página HTML contendo a *tag* ng-app. Esse atributo, pertence ao AngularJS e se encarrega de configurar a aplicação na página. No arquivo Index.html, além da *tag* do angular, devem ser importados todos os arquivos CSS e JS necessários para o projeto funcionar ex: <script src="app/app.module.js"></script>.

Os arquivos com extensão .js da aplicação quando usando a expressão de função IFFE remove variáveis de escopo global. A remoção desse tipo de variável ajuda a evitar que variáveis e declarações de função fiquem ativas na memória mais tempo do que o esperado no escopo global, evitando, ainda colisões entre variáveis.

3.2 MÉTODO

As etapas para desenvolver o sistema foram sequenciais. O sistema é simples, do ponto de vista de complexidade e quantidade de requisitos, e está bem definido do ponto de vista das regras de negócio envolvidas. Assim, foi possível desenvolver as atividades tendo como base o modelo sequencial linear de Pressman (2011), embora não sejam utilizadas as mesmas denominações que esse autor. A seguir essas fases são apresentadas.

a) Levantamento de Requisitos

Os requisitos do ponto de vista de negócio foram definidos pela docente responsável pela atividade de estágio de um dos cursos do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Esses requisitos estiveram voltados para os controles necessários para gerenciar os relatórios que devem ser apresentados pelo aluno, professor orientador e supervisor na empresa, quando for o caso.

Dos requisitos foram identificadas as funcionalidades para o sistema e os atores: aluno, professor orientador e professor responsável pelas atividades. Nessa versão do sistema os atores aluno e professor orientador possuem permissões apenas de consulta. Verificou-se que futuramente os relatórios poderiam ser preenchidos por meio do próprio sistema.

b) Modelagem dos requisitos

A atividade de análise e projeto do sistema foi realizada com o objetivo de documentar os requisitos pretendidos para o sistema e fornecer o suporte para a implementação deles. A modelagem esteve baseada na definição do diagrama de entidades e relacionamento do banco de dados e na análise do plano de estágio que é elaborado. Desse modelo de plano foram

identificados requisitos para os dados para o formulário de cadastro do plano e as respectivas tabelas de banco de dados.

c) Implementação

A implementação foi realizada utilizando as tecnologias constantes no Quadro 1. Durante a implementação foram realizados testes informais com o objetivo de identificar erros de codificação e atendimentos aos requisitos.

4 RESULTADOS

Este capítulo apresenta o resultado da realização deste trabalho que é a modelagem e o desenvolvimento de um sistema para acompanhamento de realização de estágios curriculares por parte do professor responsável por essa atividade.

4.1 ESCOPO DO SISTEMA

O estágio curricular é realizado por alunos dos cursos de graduação da UTFPR. O estágio pode nas modalidades: obrigatório e não-obrigatório. Ambos devem atender a regulamento da Universidade e disposições legais. E para o estágio obrigatório pode haver complementação regulamentar em cada curso. Para realizar o estágio o aluno deve estar matriculado pelo menos no segundo período ou ano do curso, na dependência do regime do curso e atender outras disposições regulamentares.

No sistema desenvolvido como resultado deste trabalho e que visa auxiliar no controle de realização de estágios, serão registrado alguns dados, como, o tipo de estágio que o aluno está realizado ou realizará, a empresa na qual a atividade está sendo realizada, o supervisor de estágio na empresa, o professor orientador, a data de início e a data final prevista. O sistema também permite o controle dos relatórios de acompanhamento e finais necessários para cada tipo de modalidade. Esses relatórios são elaborados pelo aluno, supervisor e orientador, de acordo com a modalidade de estágio e podem envolver a necessidade de visita na unidade cedente de estágio.

O sistema enviará *email* ao professor supervisor de estágio alertando-o de relatórios e vistas que devem ser realizadas. No *email* constará os dados do aluno, da empresa e dos documentos que devem ser preenchidos.

O sistema permitirá ao professor responsável pela atividade de estágio acompanhar se os relatórios estão sendo apresentados nos prazos definidos e organizar as visitas às empresas quando necessário. Também é possível registrar os dados da banca de apresentação do estágio, como o docente que foi banca e data de apresentação.

O aluno poderá castrar-se no sistema para acompanhar as datas de entrega de relatórios e também para cadastrar o plano de estágio. O professor orientador de estágio também poderá acompanhar os dados dos relatórios dos seus orientados. E, assim, ter acesso às datas de entrega dos relatórios, por exemplo.

4.2 MODELAGEM DO SISTEMA

O sistema possuirá três tipos de usuários: professor responsável, professor orientador e aluno. O professor responsável é o professor que é responsável pela atividade de estágio e quem realiza os cadastros no sistema, além do registro dos dados relacionados ao estágio realizado e dos relatórios apresentados. O professor orientador tem acesso para consulta aos dados do plano de estágio e de relatórios dos seus orientados. O usuário aluno pode realizar o seu cadastro no sistema, consultar dados referentes às datas dos relatórios que devem ser apresentados e preencher o seu plano de estágio.

O Quadro 2 apresenta a listagem dos requisitos funcionais identificados para o sistema.

Identificação	Nome	Descrição
RF01	Cadastrar alunos	Os alunos realizam as atividades de estágio.
RF02	Cadastrar plano de estágio	O estágio realizado por ser do tipo obrigatório ou não obrigatório. Também pode ser validada como atividade de estágio a experiência profissional. Quando o aluno já atua como profissional e isso ocorreu em um período de um ano ou mais, esse período pode ser validado como estágio. O estágio pode ser realizado na forma de estudo dirigido, sem vínculo com a empresa e, nesse caso, não há supervisor. A atividade de estágio quando realizada na empresa possui um supervisor na empresa. Um professor orientador é obrigatório.
RF03	Registrar acompanhamento (cronograma) de estágio	As datas dos relatórios são cadastradas no sistema para controle de realização dos mesmos.
RF04	Cadastrar departamentos	Os departamentos aos quais estão vinculados os cursos.
RF05	Cadastrar cursos	Os cursos aos quais os alunos pertencem. Um curso está vinculado a um departamento.
RF06	Compor banca	Compor a banca para a apresentação do estágio.
RF07	Cadastrar membros para as bancas	Membros de banca podem ser professores ou pessoas das empresas nas quais as atividades de estágio são realizadas.

Quadro 2 – Requisitos funcionais

A Figura 2 apresenta os casos de uso do sistema e os dois atores vinculados aos respectivos casos de uso.

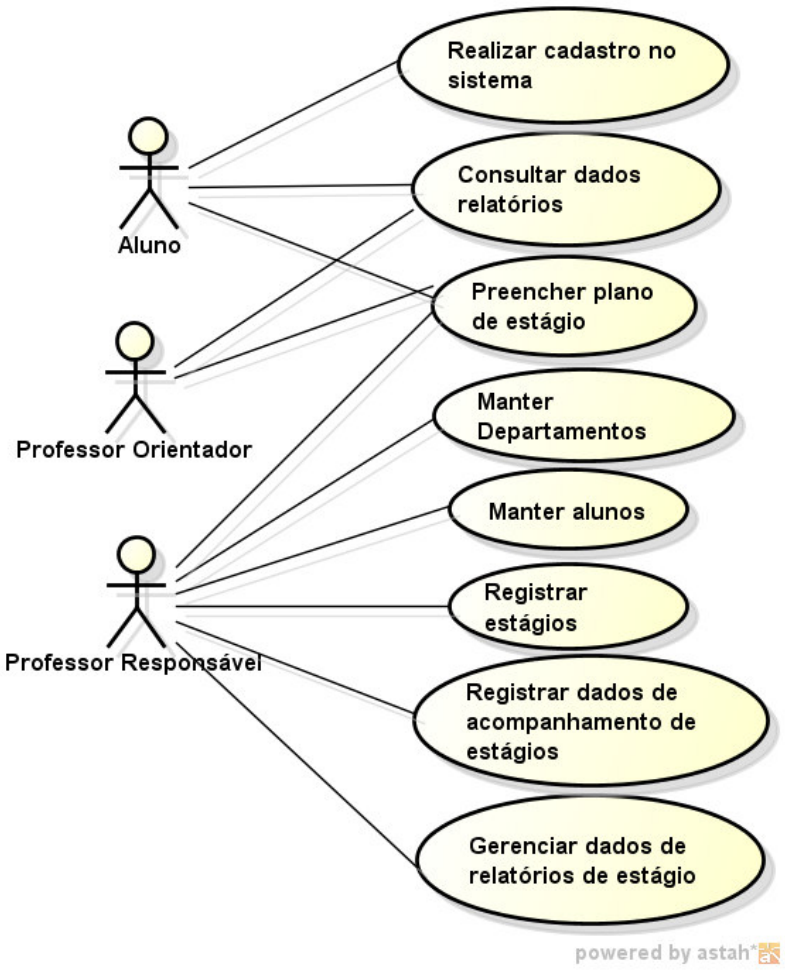
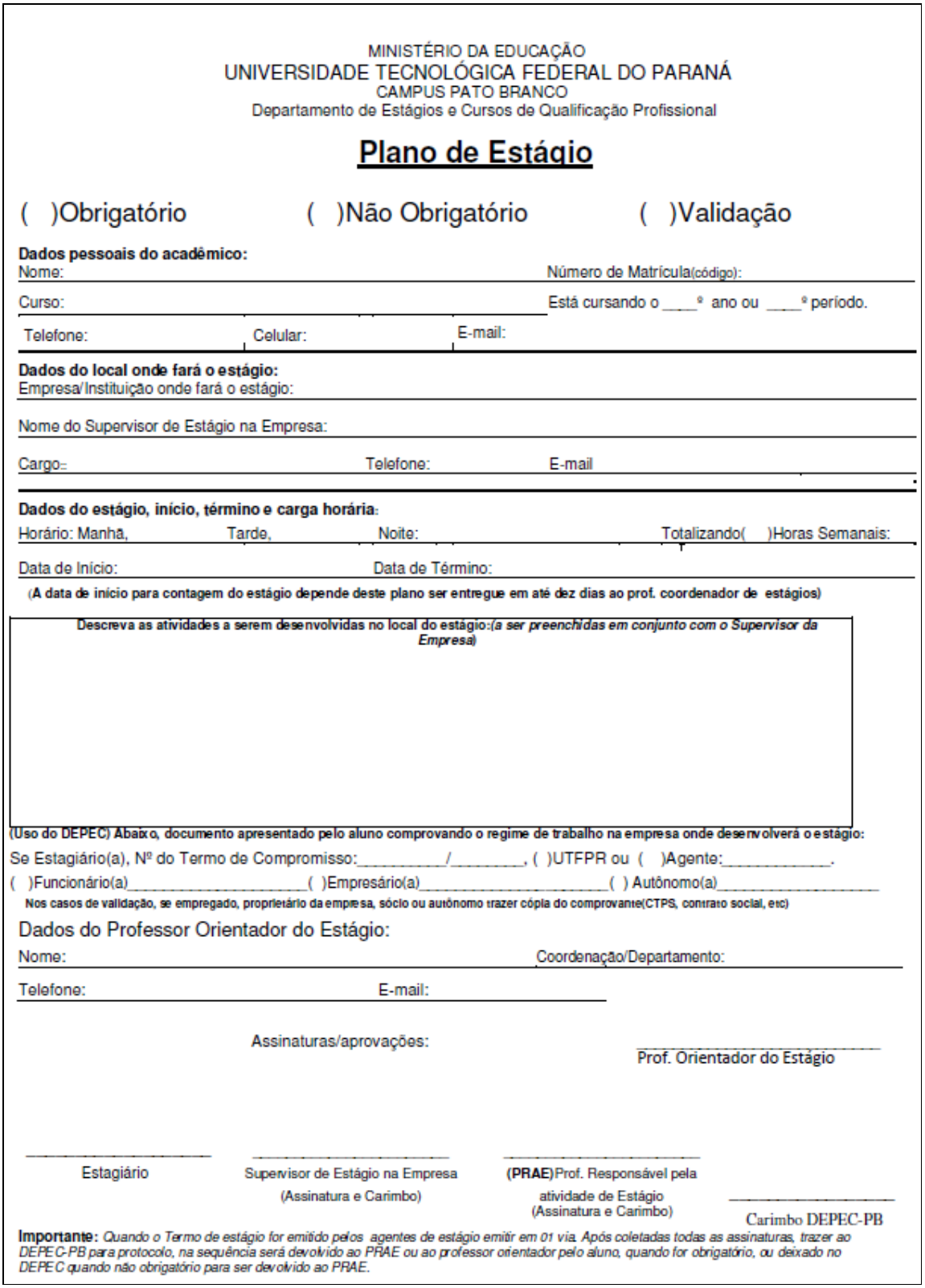


Figura 2 – Diagrama de casos de uso

Ao se cadastrar no sistema, o aluno cria um usuário. Nesse cadastro o número de Registro Acadêmico (RA) é obrigatório. E esse número será o vínculo entre o usuário aluno e o respectivo plano de estágios e relatórios relacionados. O professor responsável pela atividade, se necessário, complementa os dados do plano de estágio para que possa ser realizada a impressão do mesmo e sejam apanhadas as assinaturas necessárias. O professor orientador tem acesso, sem edição ao plano de estágio dos seus orientados. O aluno e seu orientador têm acesso aos dados de datas para apresentar os relatórios de estágio. O sistema envia, automática e antecipadamente, *email* para o aluno e seu respectivo orientador informando-os de datas para apresentação de relatório.

A Figura 3 apresenta o plano de estágio cujos dados são preenchidos pelo aluno e do qual é gerado um relatório que é o plano para assinatura e arquivamento. Esse documento contém os dados relacionados à orientação, supervisão, empresa, atividades realizadas e datas e horários de início e fim.



MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS PATO BRANCO
Departamento de Estágios e Cursos de Qualificação Profissional

Plano de Estágio

( )Obrigatório ( )Não Obrigatório ( )Validação

Dados pessoais do acadêmico:
Nome: Número de Matrícula(código):
Curso: Está cursando o ____º ano ou ____º período.
Telefone: Celular: E-mail:

Dados do local onde fará o estágio:
Empresa/Instituição onde fará o estágio:
Nome do Supervisor de Estágio na Empresa:
Cargo: Telefone: E-mail

Dados do estágio, início, término e carga horária:
Horário: Manhã, Tarde, Noite: Totalizando( )Horas Semanais:
Data de Início: Data de Término:
(**A data de início para contagem do estágio depende deste plano ser entregue em até dez dias ao prof. coordenador de estágios**)

Descreva as atividades a serem desenvolvidas no local do estágio:*(a ser preenchidas em conjunto com o Supervisor da Empresa)*

(Uso do DEPEC) Abaixo, documento apresentado pelo aluno comprovando o regime de trabalho na empresa onde desenvolverá o estágio:
Se Estagiário(a), Nº do Termo de Compromisso:_____/_____, ( )UTFPR ou ( )Agente:_____.
( )Funcionário(a)_____( )Empresário(a)_____( ) Autônomo(a)_____
Nos casos de validação, se empregado, proprietário da empresa, sócio ou autônomo trazer cópia do comprovante(CTPS, contrato social, etc)

Dados do Professor Orientador do Estágio:
Nome: Coordenação/Departamento:
Telefone: E-mail:

Assinaturas/aprovações:

_____
Prof. Orientador do Estágio

_____ Estagiário

_____ Supervisor de Estágio na Empresa (Assinatura e Carimbo)

_____ **(PRAE)**Prof. Responsável pela atividade de Estágio (Assinatura e Carimbo)

_____ Carimbo DEPEC-PB

Importante: *Quando o Termo de estágio for emitido pelos agentes de estágio emitir em 01 via. Após coletadas todas as assinaturas, trazer ao DEPEC-PB para protocolo, na sequência será devolvido ao PRAE ou ao professor orientador pelo aluno, quando for obrigatório, ou deixado no DEPEC quando não obrigatório para ser devolvido ao PRAE.*

Figura 3 – Modelo de plano de estágio

A Figura 4 apresenta o modelo de entidades e relacionamento do banco de dados.

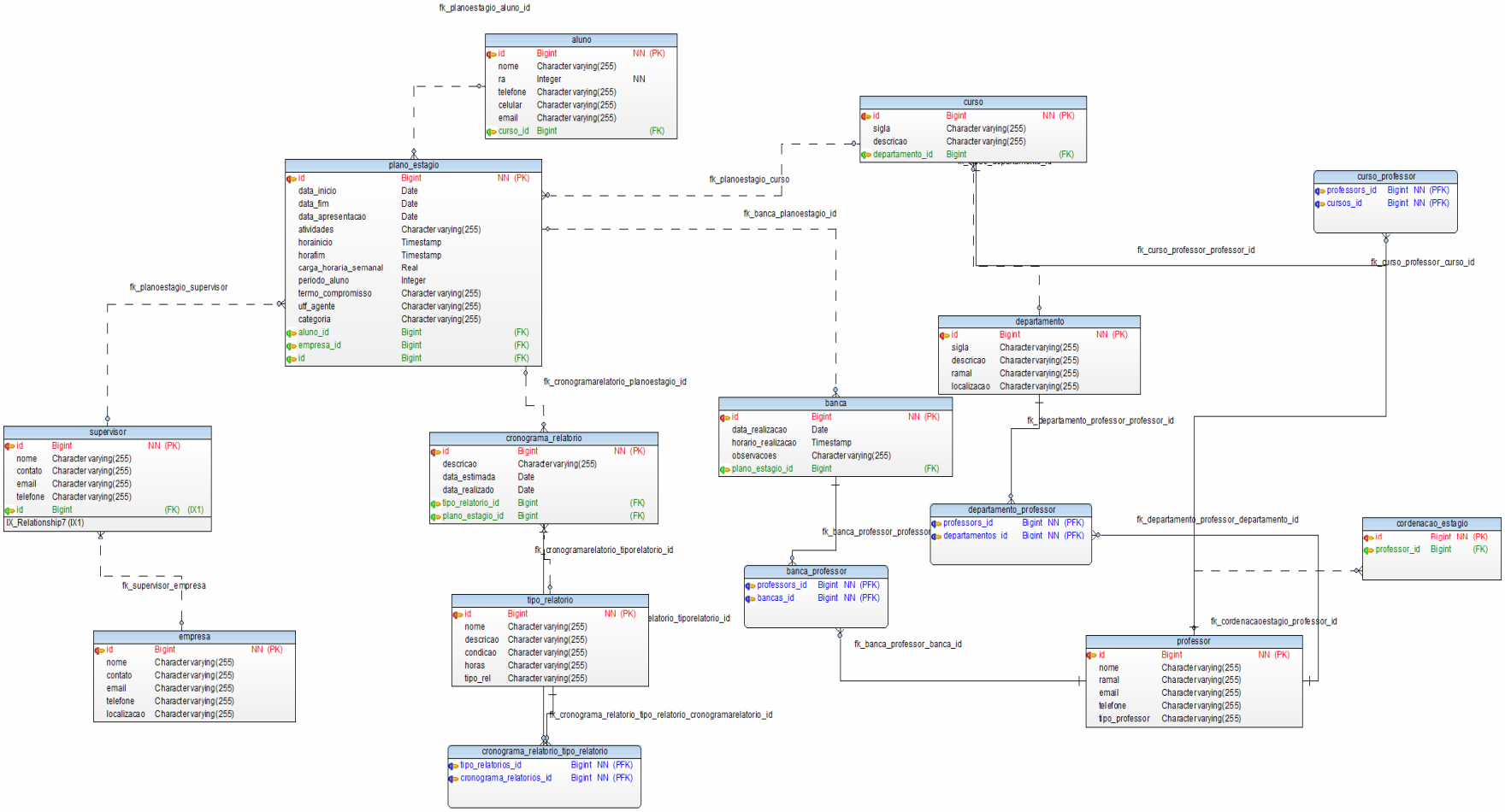


Figura 4 – Entidades do banco de dados

De acordo com a Figura 4, a tabela Professores armazena os dados de professores que exercem o papel de orientador em estágios e que são membros de bancas de avaliação de relatórios de estágio curricular supervisionado.

A tabela “PlanoEstagio” contém os dados relacionados ao estágio sendo realizado como aluno, empresa, supervisor, orientador, data de início e fim, as atividades realizadas no período de estágio e outros dados necessários ao preenchimento do plano de estágio. Essa tabela é utilizada para gerar o plano de estágio.

Os dados de apresentação do relatório de estágio são armazenados na tabela Bancas e as datas para apresentação dos relatórios ficam na tabela “CronogramaRelatorios”. Essa tabela é consultada pelo aluno e pelo orientador e os dados da mesma são utilizados para o envio de *emails* pelo sistema.

4.3 APRESENTAÇÃO DO SISTEMA

O leiaute do sistema é composto por duas áreas: a área superior contém a logo da UTFPR e os menus de navegação e a área inferior contém o conteúdo referente ao item que foi selecionado no menu de navegação. A Figura 5 apresenta as duas áreas do sistema. Na área superior estão sendo exibidas as opções de navegabilidade para o usuário no sistema. Na área inferior está sendo exibido o conteúdo referente ao plano de estágio, neste caso estão sendo listados todos os planos de estágios cadastrados.

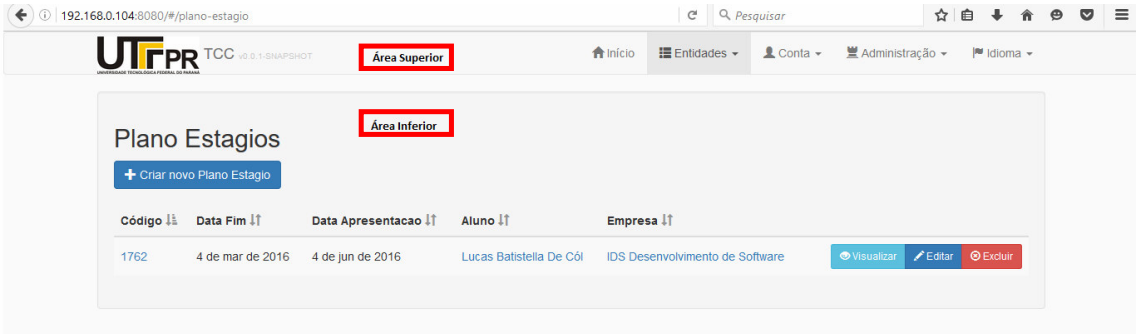


Figura 5 – Página inicial

O sistema conta com um leiaute responsivo se adaptar em qualquer dispositivo (computador, *tablet* e outros) utilizado para acessar o aplicativo, sem causar prejuízos na acessibilidade e usabilidade do sistema, como mostra a Figura 6.

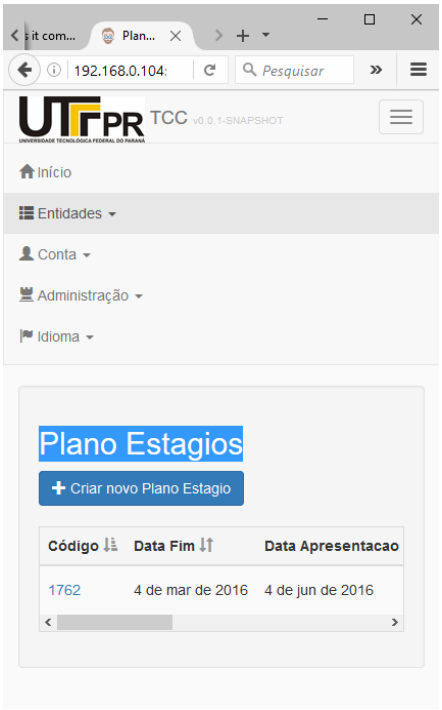


Figura 6 – Sistema sendo exibido no navegador com tamanho reduzido

A Figura 7 mostra a área inferior, com o conteúdo da página dos Planos de Estágio. Nessa área o usuário visualiza todos os planos cadastrados podendo ordenar as informações por qualquer das colunas. Além disso, ele pode realizar as operações de cadastro clicando nos botões.

Figura 7 – Lista de Planos de Estágio

Ao clicar no *link* "Novo Plano de Estágio" (área marcada como inclusão na Figura 7) o sistema abrirá uma nova janela modal mostrando todas as informações necessárias para o usuário cadastrar um novo plano de estágio e as ações que ele pode executar nessa tela, como mostra a Figura 8.

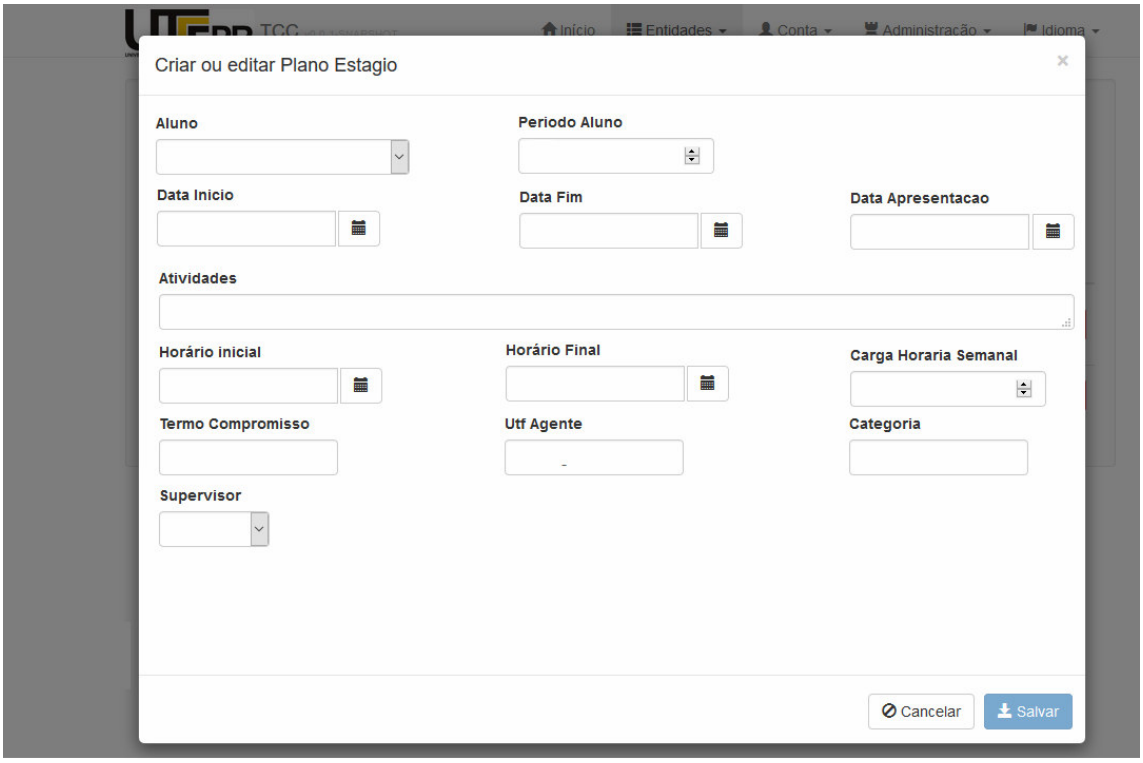


Figura 8 – Formulário de Inclusão de Plano de Estágio

Se o usuário excluir a informação de um campo obrigatório ou não preenchê-lo, será apresentada uma mensagem informando que o campo deve ser preenchido adequadamente. Enquanto os campos não informados atendendo as validações definidas, o botão para salvar o registro ficará desabilitado, como mostra a Figura 9..

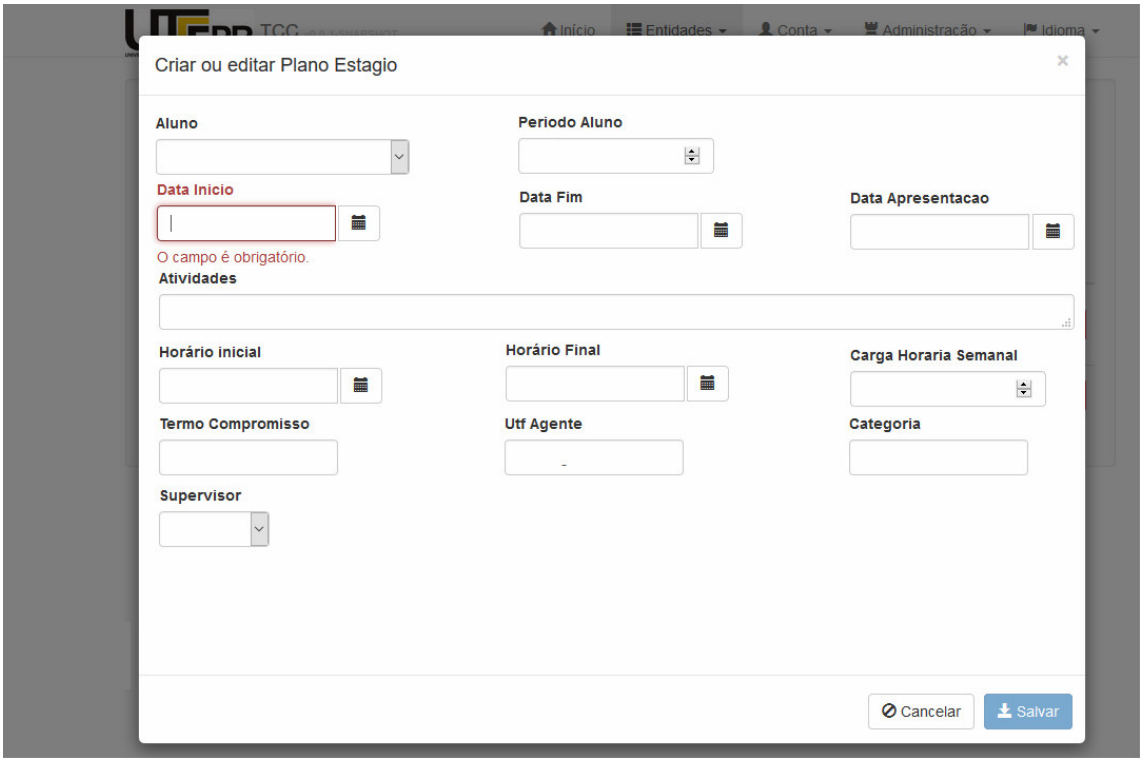


Figura 9 – Validação de campos obrigatórios

Se todos os campos estiverem preenchidos corretamente e o usuário clicar em no botão Salvar, o sistema gravará essas informações na base de dados e automaticamente fechará a tela de manutenção dos registros. Após isso, serão recarregados todos os planos de estágio e uma mensagem é exibida informando ao usuário que um novo plano de estágio foi gravado. Nessa mensagem também é apresentado o campo ID relacionado ao plano de estágio que foi incluído no banco, como mostra a Figura 10.

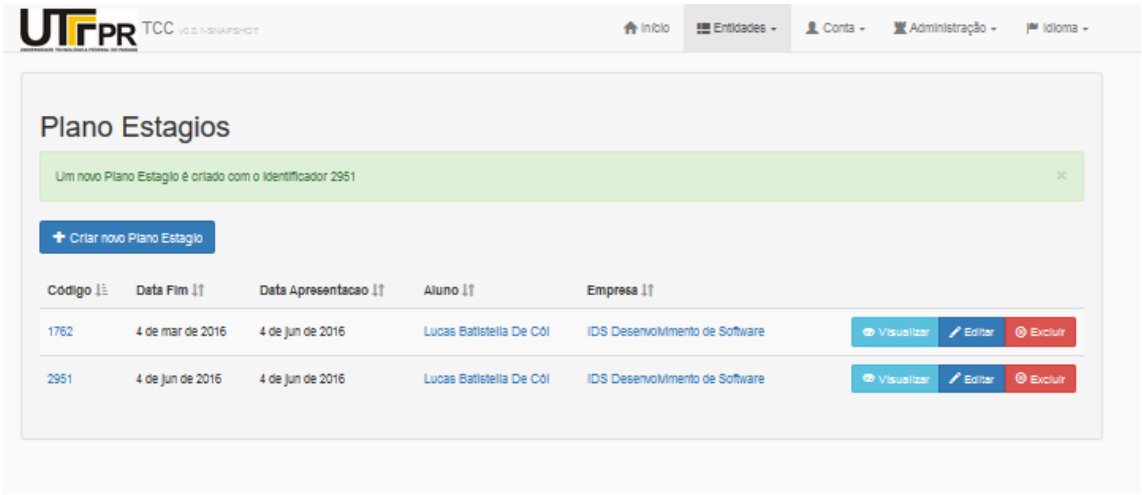


Figura 10 – Formulário para incluir plano de estágio

A edição de registro utiliza o mesmo formulário da inclusão, com algumas diferenças:

a) O campo ID é exibido mostrando ao usuário qual é o identificar do registro no banco.

b) Após abrir a tela é realizada uma operação de GET[1] para buscar todas as informações do registro armazenado na base de dados.

Se todos os campos estiverem preenchidos corretamente e o usuário clicar no botão Salvar, o sistema atualizará o registro no banco com as informações preenchidas em tela. Após esse procedimento a tela de manutenção dos registros será automaticamente. Além disso, serão recarregados todos os planos de estágio com as novas informações gravadas em banco e uma mensagem é exibida informando de qual foi o plano de estágio editado.

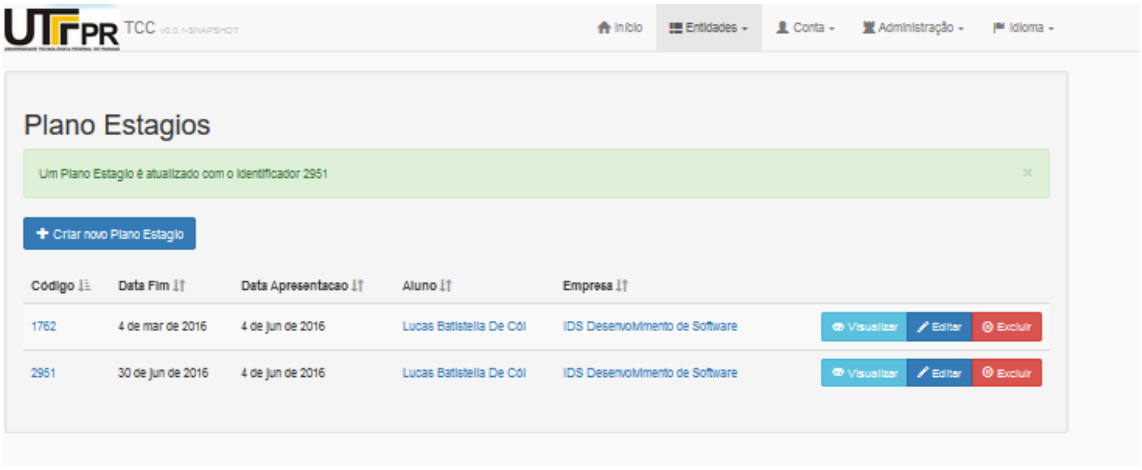


Figura 11 – Formulário para alterar plano de estágio

Para realizar a exclusão de um registro é necessário clicar no botão Excluir referente ao registro que o usuário deseja excluir, após isso será aberta uma nova janela modal mostrando qual é o registro que será excluído, se for confirmada a exclusão (clicando no excluir) a tela de exclusão de registro será fechada. Em seguida serão recarregados todos os planos de estágio com as novas informações gravadas em banco e será exibida uma mensagem de qual foi o plano de estágio excluído, como mostra a Figura 12.

[1] Recupera informações sobre o recurso identificado pela URI. Uma requisição GET não deve modificar nenhum recurso do seu sistema, ou seja, não deve ter efeitos colaterais, apenas recupera informações do sistema.

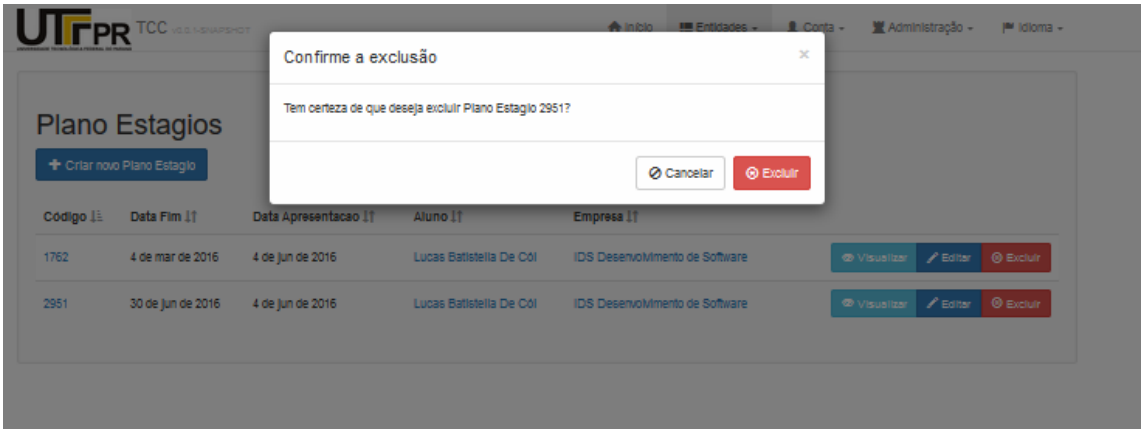


Figura 12 – Formulário para exclusão de plano de estágio

Alguns cadastros terão listas de opções para serem selecionadas. Nesse caso o usuário deve manter a tecla *CTRL* pressionada e clicar nos registros que deseja selecionar, ou manter pressionada a tecla *SHIFT* e selecionar em lote, ou seja, o primeiro e o último registro do bloco que deseja selecionar, como mostra a Figura 13.

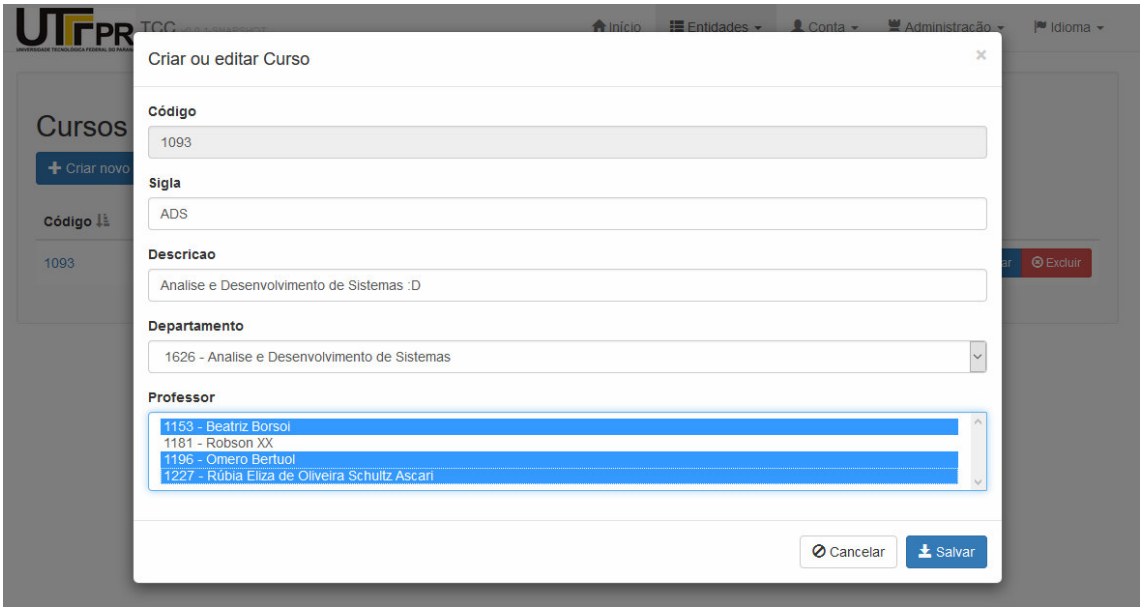


Figura 13 – Formulário para edição de um curso selecionando vários professores

A Figura 14 apresenta o relatório gerado pela ferramenta Stimulsoft que é o plano de estágio.

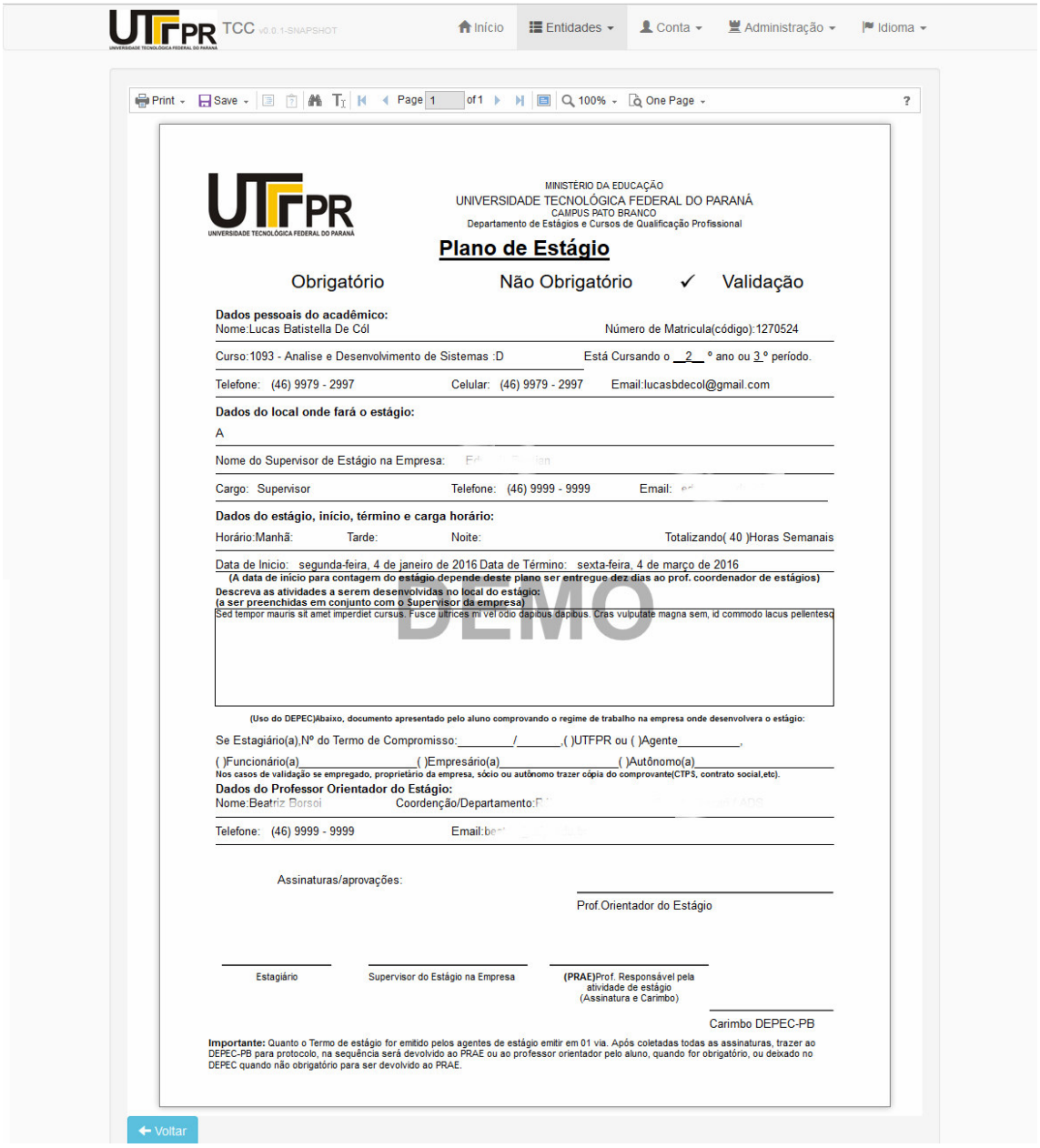


Figura 14 – Relatório gerado pelo sistema para o plano de estágio

4.4 IMPLEMENTAÇÃO DO SISTEMA

Na Listagem 1 está o *bean* (modelo) da classe “Aluno” com todas suas propriedades e anotações do Hibernate.

```java
/**
 * A Aluno.
 */
@Entity
@Table(name = "aluno")
public class Aluno implements Serializable {

private static final long serialVersionUID = 1L;

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Long id;

@Column(name = "nome")
private String nome;

@NotNull
@Column(name = "ra", nullable = false)
private Integer ra;

@Column(name = "telefone")
private String telefone;

@Column(name = "celular")
private String celular;

@Column(name = "email")
private String email;

@ManyToOne
private Curso curso;

@OneToMany(mappedBy = "aluno")
@JsonIgnore
private Set<PlanoEstagio> alunotoplanos = new HashSet<>();
```

Listagem 1 – Aluno.java

Ressalta-se que são utilizadas várias anotações do Hibernate, como:

a) @Entity

A anotação @Entity encontra-se no pacote javax.persistence, assim como as anotações padrão da JPA. Esta anotação *@Entity* marca a classe como sendo um *bean* de entidade *(entity bean*). Portanto, ela deve possuir um construtor sem argumentos que é visível ao menos com um escopo protegido (*protected*).

b) @Table

A anotação @Table identifica que é uma tabela. Por meio da propriedade *name,* pode ser alterado qual seria o nome da tabela em banco no caso.

c) @id

Cada *bean* de entidade necessita ter uma chave-primária, na qual será anotada na classe com a anotação @Id. Tipicamente a chave-primária será um campo único, apesar de que ela, também, pode ser composta por múltiplos campos.

d) @GeneratedValue

Essa anotação serve para o Hibernate saber como vai gerar a chave primária da tabela, se não for especificado o tipo do gerador, o *default* será AUTO. Existem quatro diferentes tipo de geradores de chave-primária no GeneratorType, são eles:

- AUTO - o Hibernate decide o tipo do gerador a usar, baseado no suporte da base de dados para geração de chave-primárias;
- IDENTITY - a base de dados é responsável por determinar a próxima chave-primária;
- SEQUENCE - algumas bases de dados suportam um tipo de coluna SEQUENCE;
- TABLE - mantém uma tabela separada com os valores das chaves primárias.

e) @Column

Indica para mapear uma coluna.

f) @NotNull

Campo não pode ser nulo.

g) @ManyToOne

Ligação do tipo Muitos para Uma. Nesse caso muitos alunos para um único curso.

h) @OneToMany

Ligação do tipo um para muitos.

i) @JsonIgnore

Ignora atributo na hora de retornar o Json.

A seguir é apresentado o código do controlador do REST[2] para o aluno. Na Listagem 2 está o cabeçalho da classe Java do REST.

```
/**
 * REST controller for managing Aluno.
 */
@RestController
@RequestMapping("/api")
      public class AlunoResource {
```

Listagem 2 – Cabeçalho da classe Java do rest

Na Listagem 3 é apresentado o mapeamento do método HTTP POST[3], o qual é utilizado pelo controlador para inclusão de um novo aluno.

```
    /**
     * POST  /alunos : Cria um Novo aluno.
     *
```

[2] Conjunto de restrições que define um padrão arquitetural com características específicas.
[3] Adiciona informações usando o recurso da URI. Pode adicionar informações a um recurso ou criar um novo recurso. No POST a URI significa o lugar que vai tratar a informação.

```
     * @param aluno o aluno que será criado.
     * @return a ResponseEntity com status 201 (Criado) e com o body do
novo aluno, ou com status 400 (Bad Request) se o aluno já existir
     * @throws URISyntaxException se a localização da uri estive errada
     */
    @RequestMapping(value = "/alunos",
        method = RequestMethod.POST,
        produces = MediaType.APPLICATION_JSON_VALUE)
    @Timed
    public ResponseEntity<Aluno> createAluno(@Valid @RequestBody Aluno
aluno) throws URISyntaxException {
        log.debug("REST request to save Aluno : {}", aluno);
        if (aluno.getId() != null) {
            return ResponseEntity.badRequest().headers(HeaderUtil.
createFailureAlert("aluno", "idexists", "A new aluno cannot already have an
ID")).body(null);
}
        Aluno result = alunoRepository.save(aluno);
        return ResponseEntity.created(new URI("/api/alunos/" +
result.getId()))
            .headers(HeaderUtil.createEntityCreationAlert("aluno",
result.getId().toString()))
            .body(result);
    }
```

Listagem 3 – AlunoResource.java POST(criar novo aluno)

Na Listagem 4 é apresentada a atualização de um aluno já existente, que é realizado por meio de uma requisição HTTP PUT[4].

```
/**
     * PUT  /alunos : Atualiza um Aluno existente.
     *
     * @param aluno o aluno a ser atualizado
     * @return a ResponseEntity com status 200 (OK) e com o body do aluno
atualizado,
     * ou com status 400 (Bad Request) se o aluno não for válido,
     * ou com status 500 (Internal Server Error) se o aluno não puder ser
atualizado
     * @throws URISyntaxException se a localização da uri estive errada
     */
    @RequestMapping(value = "/alunos",
        method = RequestMethod.PUT,
        produces = MediaType.APPLICATION_JSON_VALUE)
    @Timed
    public ResponseEntity<Aluno> updateAluno(@Valid @RequestBody Aluno
aluno) throws URISyntaxException {
        log.debug("REST request to update Aluno : {}", aluno);
        if (aluno.getId() == null) {
            return createAluno(aluno); }
        Aluno result = alunoRepository.save(aluno);
        return ResponseEntity.ok()
            .headers(HeaderUtil.createEntityUpdateAlert("aluno",
aluno.getId().toString()))
            .body(result);
     }
```

[4] Adiciona (ou modifica) um recurso na URI. No PUT a URI significa o lugar em que a informação será armazenada.

Listagem 4 – AlunoResource.java PUT(atualizar aluno)

O método Get é utilizado para retornar uma lista no formato JSON, que é gerada fazendo um select ALL na tabela alunos, o código é apresentado na Listagem 5.

```
     * GET  /alunos : pega todos alunos.
     *
     * @param pageable informoação de paginação
     * @return a ResponseEntity com status 200 (OK) e a lista de todos
alunos no body.
     * @throws URISyntaxException se houver algum erro em gerar a
paginação.
     */
    @RequestMapping(value = "/alunos",
        method = RequestMethod.GET,
        produces = MediaType.APPLICATION_JSON_VALUE)
    @Timed
    public ResponseEntity<List<Aluno>> getAllAlunos(Pageable pageable)
        throws URISyntaxException {
        log.debug("REST request to get a page of Alunos");
        Page<Aluno> page = alunoRepository.findAll(pageable);
        HttpHeaders headers =
PaginationUtil.generatePaginationHttpHeaders(page, "/api/alunos");
        return new ResponseEntity<>(page.getContent(), headers,
HttpStatus.OK);
    }
```

Listagem 5 – AlunoResouce.java GET(Pega Todos Alunos)

A listagem 6 apresenta o código do mapeamento do método HTTP GET, que recebe como parâmetro o código de um aluno, e retorna uma os dados desse aluno no formato JSON.

```
/**
     * GET  /alunos/:id : pega aluno pelo id.
     *
     * @param id o identificador do aluno a ser retornado.
     * @return a ResponseEntity com status 200 (OK) e com o aluno no body,
ou status 404 (Not Found)
     */
    @RequestMapping(value = "/alunos/{id}",
        method = RequestMethod.GET,
        produces = MediaType.APPLICATION_JSON_VALUE)
    @Timed
    public ResponseEntity<Aluno> getAluno(@PathVariable Long id) {
        log.debug("REST request to get Aluno : {}", id);
        Aluno aluno = alunoRepository.findOne(id);
        return Optional.ofNullable(aluno)
            .map(result -> new ResponseEntity<>(
                result,
                HttpStatus.OK))
            .orElse(new ResponseEntity<>(HttpStatus.NOT_FOUND));
    }
```

Listagem 6 – AlunoResouce.java GET (buscar aluno por id)

Na Listagem 7 está o código para excluir aluno utilizando o seu id, que é realizado por meio de uma requisição DELETE[5].

```
/**
     * DELETE  /alunos/:id : deleta o aluno pelo id.
     *
     * @param id o id do aluno a ser deletado
     * @return a ResponseEntity com status 200 (OK)
     */
    @RequestMapping(value = "/alunos/{id}",
        method = RequestMethod.DELETE,
        produces = MediaType.APPLICATION_JSON_VALUE)
    @Timed
    public ResponseEntity<Void> deleteAluno(@PathVariable Long id) {
        log.debug("REST request to delete Aluno : {}", id);
        alunoRepository.delete(id);
        return
ResponseEntity.ok().headers(HeaderUtil.createEntityDeletionAlert("aluno",
id.toString())).build();
    }
```

Listagem 7 – AlunoResouce.java DELETE (Excluir aluno por id)

O aplicativo desenvolvido está usando o I8n[6] angularJS language e suporta, atualmente as linguagens português e inglês. A classe que faz esse controle é a language.controller.js. Para isso é necessário, basicamente, dois arquivos no formato Json com as traduções e no aplicativo usar a tag : translate="nomepropriedade".

O código HTML da barra de navegação de menus é um conjunto de listas de menus com seus respectivos links permitindo a navegabilidade entre os menus. Em cada *tag* <*li*> existe a propriedade ui-sref="exemplo" que se refere a um *link* que pode ser clicado e vai ter uma ação vinculada a ele.

O menu Administração só é exibido se o usuário tiver privilégio, nesse caso "ROLE_ADMIN", isso pode ser customizado para outros menus. Parte do código para composição do menu é exibido na Listagem 8.

```
<nav class="navbar navbar-default" role="navigation">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" ng-
click="vm.toggleNavbar()">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand logo" href="#/" ng-
click="vm.collapseNavbar()">
                <img class="logo-img" src="content/images/logo-utfpr.png"/>
                <span translate="global.title">TCC</span> <span
```

[5] Remove o recurso representado pela URI.
[6] Implementação para que produtos possam ser facilmente adaptados para outras línguas.

```
class="navbar-version">v{{VERSION}}</span>
            </a>
        </div>
        <div class="navbar-collapse" uib-collapse="vm.isNavbarCollapsed"
ng-switch="vm.isAuthenticated()">
            <ul class="nav navbar-nav navbar-right">
                <li ui-sref-active="active">
                    <a ui-sref="home" ng-click="vm.collapseNavbar()">
                        <span class="glyphicon glyphicon-home"></span>
                        <span class="hidden-sm"
translate="global.menu.home">Home</span>
                    </a>
                </li>
                <!-- jhipster-needle-add-element-to-menu - JHipster will
add new menu items here -->
                <li ng-class="{active: vm.$state.includes('entity')}" ng-
switch-when="true" uib-dropdown class="dropdown pointer">
                    <a class="dropdown-toggle" uib-dropdown-toggle href=""
id="entity-menu">
                        <span>
                            <span class="glyphicon glyphicon-th-
list"></span>
                            <span class="hidden-sm"
translate="global.menu.entities.main">
                                Entities
                            </span>
                            <b class="caret"></b>
                        </span>
                    </a>
                    <ul class="dropdown-menu" uib-dropdown-menu>
                        <li ui-sref-active="active" >
                            <a ui-sref="departamento" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
asterisk"></span> 
                                <span
translate="global.menu.entities.departamento">Departamento</span>
                            </a>
                        </li>
                        <li ui-sref-active="active" >
                            <a ui-sref="curso" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
asterisk"></span> 
                                <span
translate="global.menu.entities.curso">Curso</span>
                            </a>
                        </li>
                        <li ui-sref-active="active" >
                            <a ui-sref="aluno" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
asterisk"></span> 
                                <span
translate="global.menu.entities.aluno">Aluno</span>
                            </a>
                        </li>
                        <li ui-sref-active="active" >
                            <a ui-sref="professor" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
```

```
asterisk"></span> 
                                    <span
translate="global.menu.entities.professor">Professor</span>
                                </a>
                            </li>
                            <li ui-sref-active="active" >
                                <a ui-sref="banca" ng-
click="vm.collapseNavbar()">
                                    <span class="glyphicon glyphicon-
asterisk"></span> 
                                    <span
translate="global.menu.entities.banca">Banca</span>
                                </a>
                            </li>
                            <li ui-sref-active="active" >
                                <a ui-sref="empresa" ng-
click="vm.collapseNavbar()">
                                    <span class="glyphicon glyphicon-
asterisk"></span> 
                                    <span
translate="global.menu.entities.empresa">Empresa</span>
                                </a>
                            </li>
                            <li ui-sref-active="active" >
                                <a ui-sref="supervisor" ng-
click="vm.collapseNavbar()">
                                    <span class="glyphicon glyphicon-
asterisk"></span> 
                                    <span
translate="global.menu.entities.supervisor">Supervisor</span>
                                </a>
                            </li>
                            <li ui-sref-active="active" >
                                <a ui-sref="cronograma-relatorio" ng-
click="vm.collapseNavbar()">
                                    <span class="glyphicon glyphicon-
asterisk"></span> 
                                    <span
translate="global.menu.entities.cronogramaRelatorio">Cronograma
Relatorio</span>
                                </a>
                            </li>
                            <li ui-sref-active="active" >
                                <a ui-sref="tipo-relatorio" ng-
click="vm.collapseNavbar()">
                                    <span class="glyphicon glyphicon-
asterisk"></span> 
                                    <span
translate="global.menu.entities.tipoRelatorio">Tipo Relatorio</span>
                                </a>
                            </li>
                            <li ui-sref-active="active" >
                                <a ui-sref="plano-estagio" ng-
click="vm.collapseNavbar()">
                                    <span class="glyphicon glyphicon-
asterisk"></span> 
                                    <span
translate="global.menu.entities.planoEstagio">Plano Estagio</span>
                                </a>
                            </li>
                            <li ui-sref-active="active" >
```

```
                            <a ui-sref="cordenacao-estagio" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
asterisk"></span> 
                                <span
translate="global.menu.entities.cordenacaoEstagio">Cordenacao
Estagio</span>
                            </a>
                        </li>
                        <!-- jhipster-needle-add-entity-to-menu - JHipster
will add entities to the menu here -->
                    </ul>
                </li>
                <li ng-class="{active: vm.$state.includes('account')}" uib-
dropdown class="dropdown pointer">
                    <a class="dropdown-toggle" uib-dropdown-toggle href=""
id="account-menu">
                        <span>
                            <span class="glyphicon glyphicon-user"></span>
                            <span class="hidden-sm"
translate="global.menu.account.main">
                                Account
                            </span>
                            <b class="caret"></b>
                        </span>
                    </a>
                    <ul class="dropdown-menu" uib-dropdown-menu>
                        <li ui-sref-active="active" ng-switch-when="true">
                            <a ui-sref="settings" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
wrench"></span> 
                                <span
translate="global.menu.account.settings">Settings</span>
                            </a>
                        </li>
                        <li ui-sref-active="active" ng-switch-when="true">
                            <a ui-sref="password" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
lock"></span>
                                <span
translate="global.menu.account.password">Password</span>
                            </a>
                        </li>
                        <li ui-sref-active="active" ng-switch-when="true">
                            <a href="" ng-click="vm.logout()" id="logout">
                                <span class="glyphicon glyphicon-log-
out"></span> 
                                <span
translate="global.menu.account.logout">Sign out</span>
                            </a>
                        </li>
                        <li ui-sref-active="active" ng-switch-when="false">
                            <a href="" ng-click="vm.login()" id="login">
                                <span class="glyphicon glyphicon-log-
in"></span> 
                                <span
translate="global.menu.account.login">Sign in</span>
                            </a>
                        </li>
```

```
                        <li ui-sref-active="active" ng-switch-when="false">
                            <a ui-sref="register" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-plus-
sign"></span> 
                                <span
translate="global.menu.account.register">Register</span>
                            </a>
                        </li>
                    </ul>
                </li>
                <li ng-class="{active: vm.$state.includes('admin')}"  ng-
switch-when="true" has-authority="ROLE_ADMIN" uib-dropdown class="dropdown
pointer">
                    <a class="dropdown-toggle" uib-dropdown-toggle href=""
id="admin-menu">
                        <span>
                            <span class="glyphicon glyphicon-tower"></span>
                            <span class="hidden-sm"
translate="global.menu.admin.main">Administration</span>
                            <b class="caret"></b>
                        </span>
                    </a>
                    <ul class="dropdown-menu" uib-dropdown-menu>
                        <li ui-sref-active="active">
                            <a ui-sref="user-management" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
user"></span> 
                                <span
translate="global.menu.admin.userManagement">User management</span>
                            </a>
                        </li>
                        <li ui-sref-active="active">
                            <a ui-sref="jhi-metrics" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
dashboard"></span> 
                                <span
translate="global.menu.admin.metrics">Metrics</span>
                            </a>
                        </li>
                        <li ui-sref-active="active">
                            <a ui-sref="jhi-health" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
heart"></span> 
                                <span
translate="global.menu.admin.health">Health</span>
                            </a>
                        </li>
                        <li ui-sref-active="active">
                            <a ui-sref="jhi-configuration" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-list-
alt"></span> 
                                <span
translate="global.menu.admin.configuration">Configuration</span>
                            </a>
                        </li>
                        <li ui-sref-active="active">
```

```
                            <a ui-sref="audits" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
bell"></span> 
                                <span
translate="global.menu.admin.audits">Audits</span>
                                </a>
                        </li>
                        <li ui-sref-active="active">
                            <a ui-sref="logs" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
tasks"></span> 
                                <span
translate="global.menu.admin.logs">Logs</span>
                            </a>
                        </li>
                        <li ng-hide="vm.inProduction || vm.swaggerDisabled"
ui-sref-active="active">
                            <a ui-sref="docs" ng-
click="vm.collapseNavbar()">
                                <span class="glyphicon glyphicon-
book"></span> 
                                <span
translate="global.menu.admin.apidocs">API</span>
                            </a>
                        </li>
                        <!-- jhipster-needle-add-element-to-admin-menu -
JHipster will add entities to the admin menu here -->
                    </ul>
                </li>
                <li ui-sref-active="active" uib-dropdown class="dropdown
pointer" ng-controller="JhiLanguageController as languageVm">
                    <a class="dropdown-toggle" uib-dropdown-toggle href=""
ng-if="languageVm.languages.length > 1">
                        <span>
                            <span class="glyphicon glyphicon-flag"></span>
                            <span class="hidden-sm"
translate="global.menu.language">Language</span>
                            <b class="caret"></b>
                        </span>
                    </a>
                    <ul class="dropdown-menu" uib-dropdown-menu ng-
if="languageVm.languages.length > 1">
                        <li active-menu="{{language}}" ng-repeat="language
in languageVm.languages">
                            <a href="" ng-
click="languageVm.changeLanguage(language);vm.collapseNavbar();">{{language
| findLanguageFromKey}}</a>
                        </li>
                    </ul>
                </li>
            </ul>
        </div>
    </div>
</nav>
```

Listagem 8 – navbar.html

Praticamente todos os códigos HTML possuem os seus *controllers*. Um *controller* é basicamente um arquivo JavaScript no qual estarão escritas as funções para manipular HTML. Na Listagem 9 está um exemplo com o *controller* da *navbar*. Nesse código está implementada a função de *login* (chama a classe LoginService para abrir a janela do *login*) e a *logout* (que finaliza o *login* do sistema e redireciona a aplicação para o *state home*). No construtor do *controller* sempre deve ser injetado as classes que vão ser utilizadas.

```
(function() {
    'use strict';

    angular
        .module('tccApp')
        .controller('NavbarController', NavbarController);

    NavbarController.$inject = ['$state', 'Auth', 'Principal',
'ProfileService', 'LoginService'];

    function NavbarController ($state, Auth, Principal, ProfileService,
LoginService) {
        var vm = this;

        vm.isNavbarCollapsed = true;
        vm.isAuthenticated = Principal.isAuthenticated;

        ProfileService.getProfileInfo().then(function(response) {
            vm.inProduction = response.inProduction;
            vm.swaggerDisabled = response.swaggerDisabled;
        });

        vm.login = login;
        vm.logout = logout;
        vm.toggleNavbar = toggleNavbar;
        vm.collapseNavbar = collapseNavbar;
        vm.$state = $state;

        function login() {
            collapseNavbar();
            LoginService.open();
        }

        function logout() {
            collapseNavbar();
            Auth.logout();
            $state.go('home');
        }

        function toggleNavbar() {
            vm.isNavbarCollapsed = !vm.isNavbarCollapsed;
        }

        function collapseNavbar() {
            vm.isNavbarCollapsed = true;
        }
    }
})();
```

Listagem 9 – navbar.controller.js

Resouce é uma *factory* que cria um objeto que lhe permite interagir com as fontes de dados do lado do servidor RESTful. A Listagem 10 mostra as chamadas do aluno.service.js para o REST.

```
(function() {
    'use strict';
    angular
        .module('tccApp')
        .factory('Aluno', Aluno);

    Aluno.$inject = ['$resource'];

    function Aluno ($resource) {
        var resourceUrl =  'api/alunos/:id';

        return $resource(resourceUrl, {}, {
            'query': { method: 'GET', isArray: true},
            'get': {
                method: 'GET',
                transformResponse: function (data) {
                    if (data) {
                        data = angular.fromJson(data);
                    }
                    return data;
                }
            },
            'update': { method:'PUT' }
        });
    }
})();
```

Listagem 10 – aluno.service.js

HTML é usado para exibir informações do aluno. Nesse caso estão sendo exibidas todas as informações do aluno e qual curso ele está vinculado. Além disso, a propriedade ui-sref cria o *link* do curso vinculado ao aluno. Se clicado será aberta a tela de detalhe de cursos. Quem realiza esse mapeamento é o curso.state.js.

No código apresentado na Listagem 11, HTML está sendo utilizado uma propriedade do angular que é o Binding. O Binding é a sincronização entre o *model* e a *view* EX:{{vm.aluno.ra}}

```
<div>
    <h2><span translate="tccApp.aluno.detail.title">Aluno</span>
{{vm.aluno.id}}</h2>
    <hr>
    <jhi-alert-error></jhi-alert-error>
    <dl class="dl-horizontal jh-entity-details">
        <dt><span translate="tccApp.aluno.nome">Nome</span></dt>
        <dd>
            <span>{{vm.aluno.nome}}</span>
        </dd>
        <dt><span translate="tccApp.aluno.ra">Ra</span></dt>
        <dd>
            <span>{{vm.aluno.ra}}</span>
        </dd>
```

```
        <dt><span translate="tccApp.aluno.telefone">Telefone</span></dt>
        <dd>
            <span>{{vm.aluno.telefone}}</span>
        </dd>
        <dt><span translate="tccApp.aluno.celular">Celular</span></dt>
        <dd>
            <span>{{vm.aluno.celular}}</span>
        </dd>
        <dt><span translate="tccApp.aluno.email">Email</span></dt>
        <dd>
            <span>{{vm.aluno.email}}</span>
        </dd>
        <dt><span translate="tccApp.aluno.curso">Curso</span></dt>
        <dd>
            <a ui-sref="curso-
detail({id:vm.aluno.curso.id})">{{vm.aluno.curso.id + ' - ' +
vm.aluno.curso.descricao}}</a>
        </dd>
    </dl>
    <button type="submit"
            onclick="window.history.back()"
            class="btn btn-info">
        <span class="glyphicon glyphicon-arrow-left"></span> <span
translate="entity.action.back"> Back</span>
    </button>
</div>
```

Listagem 11 – aluno-detail.html

Em curso.state.js estão todos os estados da página de cursos (curso, curso-detail, curso.new, curso.edit, curso.delete), mapeando quem será o *controller*, o HTML e a URL de cada página. Se clicado no curso (na página de aluno) redirecionará para a página curso-detail.html, que por sua vez vai ser carregada exibindo o curso que o aluno estava vinculado (Listagem 12).

```
.state('curso-detail', {
    parent: 'entity',
    url: '/curso/{id}',
    data: {
        authorities: ['ROLE_USER'],
        pageTitle: 'tccApp.curso.detail.title'
    },
    views: {
        'content@': {
            templateUrl: 'app/entities/curso/curso-detail.html',
            controller: 'CursoDetailController',
            controllerAs: 'vm'
        }
    },
    resolve: {
        translatePartialLoader: ['$translate', '$translatePartialLoader',
function ($translate, $translatePartialLoader) {
            $translatePartialLoader.addPart('curso');
            return $translate.refresh();
        }],
        entity: ['$stateParams', 'Curso', function($stateParams, Curso) {
            return Curso.get({id : $stateParams.id}).$promise;
        }]
```

```
    }
})
.state('curso.new', {
    parent: 'curso',
    url: '/new',
    data: {
        authorities: ['ROLE_USER']
    },
```

Listagem 12 – curso.state.js

A Figura 15 apresenta o leiaute do relatório que gera o plano de estágio. Esse leiaute é composto na ferramenta Stimulsoft.

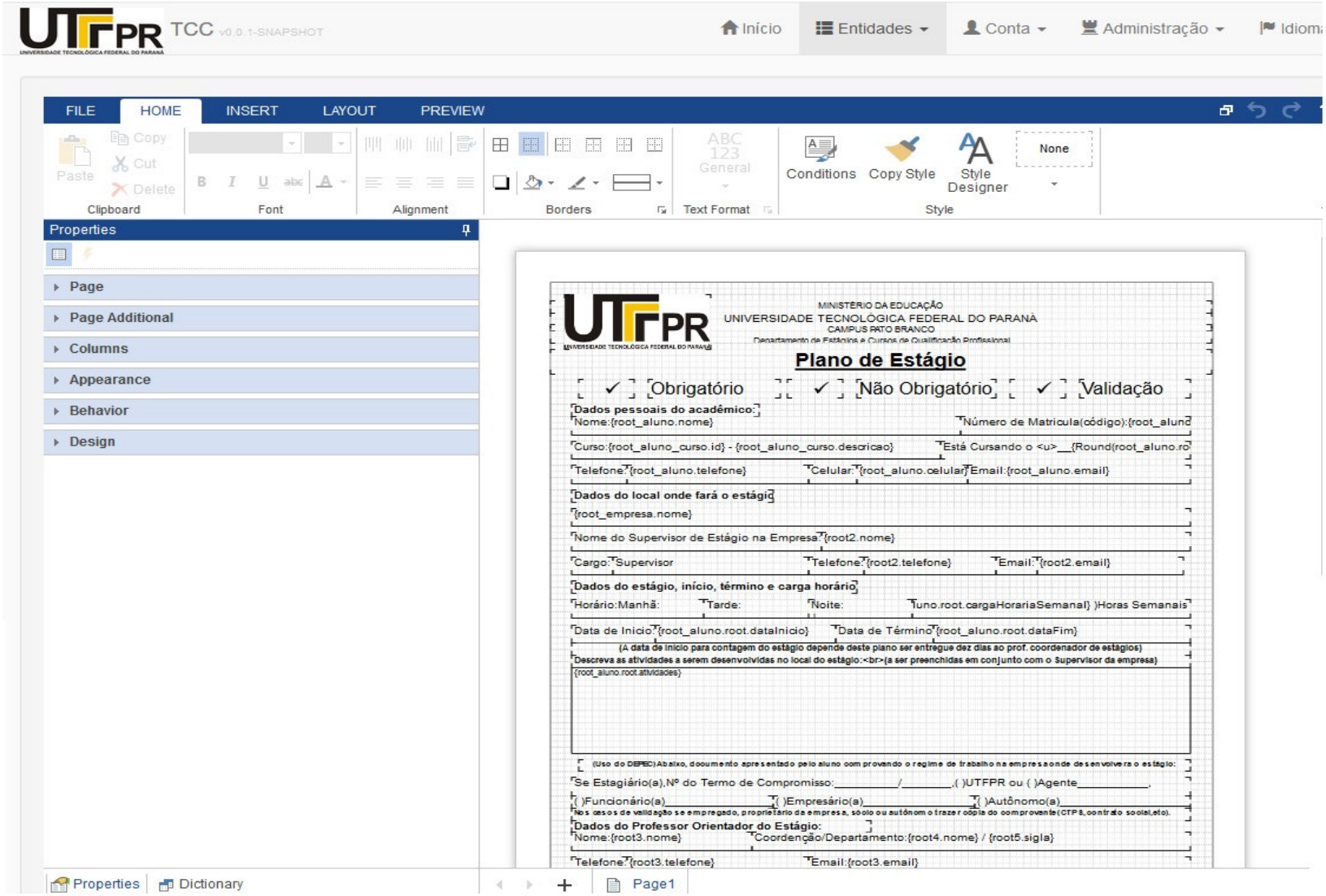


Figura 15 – Composição de relatório na ferramenta Stimulsoft

5 CONCLUSÃO

O objetivo deste trabalho foi desenvolver um aplicativo que auxiliasse no gerenciamento das atividades básicas relacionadas aos estágios curriculares. Esses estágios são realizados por alunos e existem nas modalidades: obrigatório e não obrigatório. A segunda modalidade é validada como parte da carga horária do curso e para ser formalizada um relatório de estágio é apresentado. O aplicativo permite o controle das datas dos relatórios, facilitando o acompanhamento pelo supervisor e pelo professor responsável pela atividade de estágio.

Para o desenvolvimento foram utilizadas tecnologias para desenvolvimento web que visam facilitar a implementação do programador, criar aplicações de leiaute responsivo e que permitem implementar lógica de negócio no cliente, minimizando, assim, o tráfego de dados e a carga de processamento no servidor.

Há uma grande diversidade para o desenvolvimento *web*, o que pode dificultar a escolha do desenvolvedor. Considerando o conhecimento por atuação profissional, foi, inicialmente, pensado usar a tecnologia Flex. Contudo, um dos propósitos deste trabalho de conclusão de curso foi o aprendizado. E, por isso, optou-se pelo *framework* AngularJS. Completamente novas. No final do trabalho, provou ter sido uma ótima escolha. O AngulasJS é de fácil aprendizagem, mas um pouco difícil de acompanhar, pois está em constante evolução

Uma das dificuldades no início do desenvolvimento do projeto foi difícil definir uma metodologia de estudo. A Internet possui uma diversidade de materiais, porém nem todo de qualidade. Com o decorrer do desenvolvimento descobriu-se que a própria documentação e buscas em guia de estilos (boas práticas) de pessoas que são referência na linguagem (Todd Motto/John Papa) foram os melhores tutores de estudo.

REFERÊNCIAS

BI-FENG, Chen. Technology and Application of Rich Client Based on AJAX. **Computer Science**, v. 38, n.10, October 2011.

BOZZON, Alessandro; COMAI, Sara; FRATERNALI, Piero; CARUGHI, Giovanni Toffetti. **Conceptual modeling and code generation for Rich Internet Applications.** International Conference on Web Engineering (ICWE), 2006, p. 1-8.

BRASIL. Lei Nº 11.788, de 25 de setembro de 2008. **Presidência da República. Casa Civil.Subchefia para Assuntos Jurídicos**. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2007-2010/2008/lei/l11788.htm>. Acesso em: 30 out. 2015.

DEVMEDIA. **AngularJS e Node.js**. Disponível em:<http://www.devmedia.com.br/curso/serie-angularjs-e-node-js/434>. Acesso em: 01 jun. 2016.

DRIVER, Mark; VALDES, Ray; PHIFER. Gene. **Rich internet applications are the next evolution of the web**. Technical report, Gartner, May 2005. Disponível em: <https://www.gartner.com/doc/480725/rich-internet-applications-evolution-web>. Acesso em: 27 ago. 2015.

DUHL, Joshua. **White paper: Rich Internet Applications**. Technical report, IDC, November 2003.

FRATERNALI, Piero; ROSSI, Gustavo; SÁNCHEZ-FIGUEROA, Fernando. Rich Internet Applications. **IEEE Computer Society**, may/jun 2010, p. 9-12

LINAJE, Marino; PRECIADO, Juan Carlos; SÁNCHEZ-FIGUEROA, Fernando. Engineering rich internet application user interfaces over legacy web models. **IEEE Computer Society**, November/December 2007, p. 53-59.

PAVLIĆ; Daniel; PAVLIĆ; Mile; JOVANOVIĆ, Vladan. **Future of Internet technologies.** In: International ICT Convention MIPRO on information and communication technology, electronics and microeletronics, 2012. p. 1366-1371

PRESSMAN, Roger. **Engenharia de software**. Rio de Janeiro: McGraw-Hill, 2012.

SAMPAIO, Julio. **Criando aplicação de notícias com AngularJS**. Disponível em: <http://www.devmedia.com.br/criando-aplicacao-de-noticias-com-angularjs/34062>. Acesso em: 01 jun. 2016.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ. **Regulamento dos estágios curriculares supervisionados dos cursos de educação profissional técnica de nível médio, dos cursos superiores de tecnologia e dos cursos de bacharelado da UTFPR**. Alteração

Aprovada – Resolução Nº. 033/14 – COGEP de 16/05/14. Curitiba, março de 2014. Disponível em: <http://www.utfpr.edu.br/estrutura-universitaria/pro-reitorias/prograd/legislacao/RegulamentoEstgioBachareladoseTecnologiaversaopsCOGEP.p>. Acesso em: 31 mar. 2016.