UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

LUCIANO HELENO DA ROSA

SISTEMA PARA ELABORAÇÃO E DISPONIBILIZAÇÃO DE QUESTIONÁRIOS DE AVALIAÇÃO

MONOGRAFIA DE ESPECIALIZAÇÃO

PATO BRANCO
2015

LUCIANO HELENO DA ROSA

SISTEMA PARA ELABORAÇÃO E DISPONIBILIZAÇÃO DE QUESTIONÁRIOS DE AVALIAÇÃO

Trabalho de Conclusão de Curso, apresentado ao III Curso de Especialização em Tecnologia Java, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientadora: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2015

SISTEMA PARA ELABORAÇÃO E DISPONIBILIZAÇÃO DE QUESTIONÁRIOS DE AVALIAÇÃO

Por

Luciano Heleno da Rosa

Esta monografia foi apresentada às 17h30 do dia 7 de outubro de 2015 como requisito parcial para a obtenção do título de ESPECIALISTA, no III curso de Especialização em Tecnologia Java, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. O acadêmico foi arguido pela Banca Examinadora composto pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Profa. Dr. Beatriz Brsoi
Orientadora
UTFPR – Campus Pato Branco

Prof. Msc. Rúbia Eliza de Oliveira Schultz Ascari
Banca
UTFPR – Campus Pato Branco

Prof. Msc. Vinicius Pegorini
Banca
UTFPR – Campus Pato Branco

Prof. Msc. Robison Cris Brito
Coordenador do curso de Especialização
UTFPR – Campus Pato Branco

RESUMO

ROSA, Luciano Heleno da. Sistema para elaboração e disponibilização de questionários de avaliação. 51 f. Monografia (Trabalho de especialização) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2015.

O uso de questionários para avaliação de atividades é bastante comum nos mais diversos setores. Na academia (escolas, universidades, faculdades, centros de ensino e outros), esse tipo de questionário têm o objetivo de permitir, por exemplo, avaliar atividades realizadas ou em realização em termos didáticos e de conteúdo. A existência de um sistema para elaboração de questionários para essas avaliações e disponibilização dos mesmos para que possam ser respondidos traz facilidades para os usuários que os respondem e para os gestores que obtém os resultados. Para que sejam disponibilizados apenas os questionários aos usuários que possuem permissão para respondê-los é necessário que haja um controle adequado de acesso dos grupos de usuários ao sistema. O sistema desenvolvido como resultado deste trabalho permite a elaboração desse tipo de questionário, a disponibilização para grupos específicos e a visualização dos dados obtidos a partir das respostas. O sistema é um aplicativo para *web* desenvolvido em Java para *web* com banco de dados PostgreSQL.

Palavras-chave: Aplicativo *web*. Java para *web*. Aplicação Internet rica.

ABSTRACT

ROSA, Luciano Heleno da. Software to compose and to provide assessment questionnaires. 51 f. Monografia (Trabalho de especialização) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2015.

The use of questionnaires for evaluation activities is quite common in various sectors. In academia (schools, universities, colleges, training centers, etc.), this type of questionnaire are intended to allow, for example, to evaluate activities carried out or being carried out in educational terms and content. The existence of software for the preparation of questionnaires for these assessments brings facilities to the users who answer them and for managers to get the results. So that it is available only questionnaires to users who have permission to respond them there must be an adequate control of access to the system user groups. The software developed as a result of this work allows the development of this type of questionnaire, the provision for specific groups and data visualization obtained from responses. The software is a web application developed in Java web with PostgreSQL as database.

Palavras-chave: Web application. Java *web*. Rich Internet Application.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGOS

LISTA DE SIGLAS

Ajax	*Asynchronous Javascript and XML*
JDK	*Java Development Kit*
HTTP	*Hypertext Transfer Protocol*
HTML	*HyperText Markup Language*
MVC	*Model-View-Controller*
RIA	*Rich Internet Applications*
URL	*Uniform Resource Locator*
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais com o contexto e uma visão geral do aplicativo desenvolvido, que está inserido nesse contexto. São, ainda, apresentados o objetivo e a justificativa de realização do trabalho. No final do capítulo está a organização do texto por meio da apresentação dos seus capítulos subsequentes.

1.1 CONSIDERAÇÕES INICIAIS

A Universidade Tecnológica Federal do Paraná (UTFPR), como muito provavelmente ocorre em outras instituições de ensino, faz avaliações das atividades que realiza, como aulas regulares e cursos de extensão e da atuação dos departamentos e chefias. Essas avaliações são realizadas por meio de questionários respondidos por alunos, professores, técnicos administrativos e demais funcionários. Atualmente, muitas dessas avaliações, especialmente para as aulas dos cursos de especialização e de atividades de extensão, têm sido realizadas por meio de formulários impressos. O sistema acadêmico da UTFPR permite a avaliação do docente pelo discente e das chefias pelos docentes. Contudo, avaliações que não essas têm sido realizadas por formulários impressos ou por questionários desenvolvidos em ferramentas como o Google Docs.

Visando permitir uma maneira padrão e facilitada para a elaboração desse tipo de questionário e a disponibilização dos mesmos aos respectivos respondentes, verificou-se a oportunidade de desenvolver um aplicativo *web* que pudesse atender essa necessidade.

O aplicativo desenvolvido permite a elaboração dos mais diversos tipos de questionários compostos por perguntas abertas e fechadas. O aplicativo realiza o controle de quem já respondeu um determinado questionário para que não haja duplicidade de respondentes, mas não será identificado o autor das respostas.

O aplicativo gerará a compilação das respostas visando facilitar a análise. Em uma primeira versão, serão gerados apenas somatórios das respostas. Os dados

poderão ser facilmente exportados para planilhas de cálculos e, assim, manipulados pelo usuário de acordo com interesses e necessidades.

1.2 OBJETIVOS

A seguir são apresentados o objetivo geral e os objetivos específicos da realização deste trabalho.

1.2.1 Objetivo Geral

Desenvolver um aplicativo *web* para a composição e a disponibilização de questionários de avaliação de atividades.

1.2.2 Objetivos Específicos

- Implementar um editor de questionários que permita compor perguntas descritivas e de múltipla escolha..
- Possibilitar agrupamentos de usuários para acesso aos questionários como respondentes.
- Realizar controle dos respondentes dos questionários, mas sem identificar o autor das respostas.
- Permitir o acesso aos dados obtidos pelas respostas dos questionários a usuários específicos do sistema.

1.3 JUSTIFICATIVA

O desenvolvimento de um aplicativo *web* para a realização de questionários de avaliação de atividades é bastante útil para instituições de ensino, como a UTFPR, que realizam sistematicamente esse tipo de avaliação.

O sistema desenvolvido permitirá compor questionários com perguntas abertas e fechadas, disponibilizá-los para o público alvo específico e determinar o período no qual o questionário estará disponível.

O sistema fará o gerenciamento dos respondentes, não permitindo que um mesmo questionário seja respondido mais de uma vez pela mesma pessoa. Contudo, será mantido o completo anonimato do respondente.

1.4 ESTRUTURA DO TRABALHO

A sequência deste texto está organizada em capítulos. O Capítulo 2 apresenta o referencial teórico do trabalho. Como o aplicativo desenvolvido é para ambiente Internet e caracterizado como de interface rica, o referencial teórico versa sobre esse tipo de aplicação.

No Capítulo 3 estão apresentados os materiais e o método utilizados na modelagem e no desenvolvimento do aplicativo.

Os resultados são apresentados no Capítulo 4 por meio do escopo do sistema, da modelagem realizada, da apresentação do sistema que ocorre pelas suas telas e exemplos de códigos desenvolvidos.

Por fim estão as considerações finais e as referências bibliográficas.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico do trabalho relacionado às aplicações *web* por ser esse o tipo de aplicação desenvolvida.

2.1 APLICATIVOS PARA AMBIENTE WEB

O uso da *World Wide Web* a partir do início de 1990 teve como objetivo tornar possível o acesso à informação de qualquer origem de maneira consistente e simples (JAZAYERI, 2007). Hipertexto, composto por páginas desenvolvidas em *HyperText Markup Language* (HTML), foi adotado como um padrão de uso simples para acesso aos documentos e para vinculá-los entre si. O protocolo *Hypertext Transfer Protocol* (HTTP) foi projetado para permitir a um computador (a máquina cliente) requisitar dados e documentos para outro computador (a máquina servidor). Desta forma, a *web* foi vista como um vasto repositório de informações que provê acesso a um amplo número de usuários (JAZAYERI, 2007).

Páginas baseadas em hipertexto e HTTP definem a *web* 1.0 caracterizada por ser estática (como é a forma padrão de apresentação de conteúdo dos *websites*), não prover conteúdo interativo e prover diretórios de conteúdos para serem vinculados por meio de *links* (VICENTIM, 2013). A Web 1.0 é definida por Almeida e Lourenço (2011) como somente leitura, pela publicação de informação em *web* sites estáticos, por meio dos quais os usuários acessam informações diretamente pelo *browser* ou por máquinas de busca. Em seguida surgiram as trocas de mensagens (a era do *e-mail*) e dos motores de busca, ainda que bastante simplistas (VICENTIM, 2013).

De páginas baseadas em HTML e HTTP, um endereço *web* pode, atualmente, referir-se a uma aplicação, sofisticada em termos de recursos, sendo invocada. A *web* é uma plataforma que oferece um vasto conjunto de ferramentas e componentes para os desenvolvedores de aplicações. Esses elementos permitem agregar valor no sentido de melhorar a interatividade das aplicações e, assim, torná-las mais próximas das aplicações *desktop*, que são consideradas como modelo em termos de recursos de interface.

Fraternali (1999) descreve uma aplicação *web* como um híbrido entre hipermídia e sistemas de informação e, em consequência, apresenta os seguintes como requisitos para essas aplicações:

a) a necessidade de manipular dados estruturados (por exemplo, registros em bancos de dados) e não estruturados (por exemplo, itens multimídia);
b) o suporte para acesso exploratório por meio de interfaces navegacionais;
c) o alto nível de qualidade gráfica;
d) a customização e a adaptação dinâmica da estrutura do conteúdo, primitivas de navegação e estilos de apresentação;
e) o suporte ao comportamento pró-ativo, isto é, para recomendação e filtragem.

Esses requisitos caracterizam um incremento ao fornecido pelas aplicações *web* baseadas em HTML, caracterizada como *web* 1.0 ou tradicional. A ideia de *web* 2.0 foi formulada de maneira compreensível por Tim O'Reilley (2005) e é caracterizada como aplicação dinâmica no sentido da interação do usuário e são responsivas às requisições do usuário como ocorre com as aplicações *desktop* (JAZAYERI, 2007).

A Web 2.0, também chamada de *web* participativa ou social ou mesmo de leitura e escrita (ALMEIDA; LOURENÇO, 2011), foi a revolução dos *blogs* e *chats*, das redes e mídias sociais colaborativas e do conteúdo produzido pelos próprios usuários (VICENTIM, 2013). Além do uso pessoal, a Internet se popularizou nas empresas tornando-se quase que obrigatória para sucesso no mercado. Para Vicentim (2013), essa nova forma de utilizar a Internet passa a caracterizar a *web* como plataforma. E a navegação por meio de dispositivos móveis e uso de aplicativos nesses dispositivos passa a ter presença no cotidiano das pessoas.

A Web 2.0 se tornou parte integrante da vida das pessoas e dos negócios. Empresas, governos e outras organizações conduziram à criação de novas aplicações e modelos de negócio para necessidades internas e externas às organizações (HOGG *et al*, 2006). A informação contida na *web* é pouco estruturada, limitando seu potencial de uso (ALMEIDA; LOURENÇO, 2011). Hendler (2009) ressalta que a Web 2.0 é também restringida pelo seu extraordinário volume de informação disponível, a alta escala com que o conteúdo é publicado e a falta de habilidade dos sistemas existentes em integrar dados originados de diferentes fontes ou diferentes formatos.

O termo Web 3.0 recebe denominações como “Web Semântica” ou “Web Inteligente” (VICENTIM, 2013). A inteligência associada à Web 3.0 visa tratar a falta de estrutura e organização da Web 2.0 em termos do conteúdo que disponibiliza, fazendo, assim, que a *web* seja mais fácil de usar, mais eficiente e mais valorosa para os usuários. E a semântica está associada à interpretação do conteúdo pesquisável e assim entregar conteúdo mais apropriado e relevante para o usuário final (ALMEIDA; LOURENÇO, 2011).

A Web 3.0 oferece a informação de forma organizada para que as máquinas possam entender. Assim, elas podem ajudar as pessoas respondendo pesquisas e perguntas com mais efetividade. A Web 3.0 representa o uso mais inteligente do conhecimento e conteúdo disponibilizado *online*, com sites e aplicações, experiência e publicidade baseada nos interesses e no comportamento de cada indivíduo (VICENTIM, 2013). A Web 3.0 oferece ferramentas para melhor gerenciar fluxos de informação e entregar uma experiência de uso da web mais rápida e rica (HENDLER, 2009).

Um próximo passo em termos de evolução da Internet é a “*mobile web*”, ou Web 4.0. A Web 4.0 visa promover conexão entre todos os dispositivos no mundo real e virtual em tempo real (FLAT EDUCATION, 2015).

Web 5.0 será sobre interação emocional entre humanos e computadores. É uma *web* conectada que permite comunicação ampla entre as pessoas e das pessoas com os dispositivos (como um assistente pessoal). Web 5.0 é chamada de *web* simbiótica (FLAT EDUCATION, 2015). A interação ocorrerá como um hábito diário para muitas pessoas baseadas na neurotecnologia. Um exemplo da Web 5.0, está em www.wefeelfine.org com um aplicativo que mapeia a emoção das pessoas (FLAT EDUCATION, 2015).

A Figura 1 apresenta, de forma gráfica, a expansão da *web*: da *web* baseada em HTML com disponibilização de conteúdo estático (Web 1.0), passando pela *web* da comunicação (Web 2.0: troca de mensagens, *emails*, *blogs*), *web* do contexto (Web 3.0: semântica), das coisas (Web 4.0) e chegando a *web* do pensamento (Web 5.0) e tratando das áreas: entretenimento, comunicação e negócio.

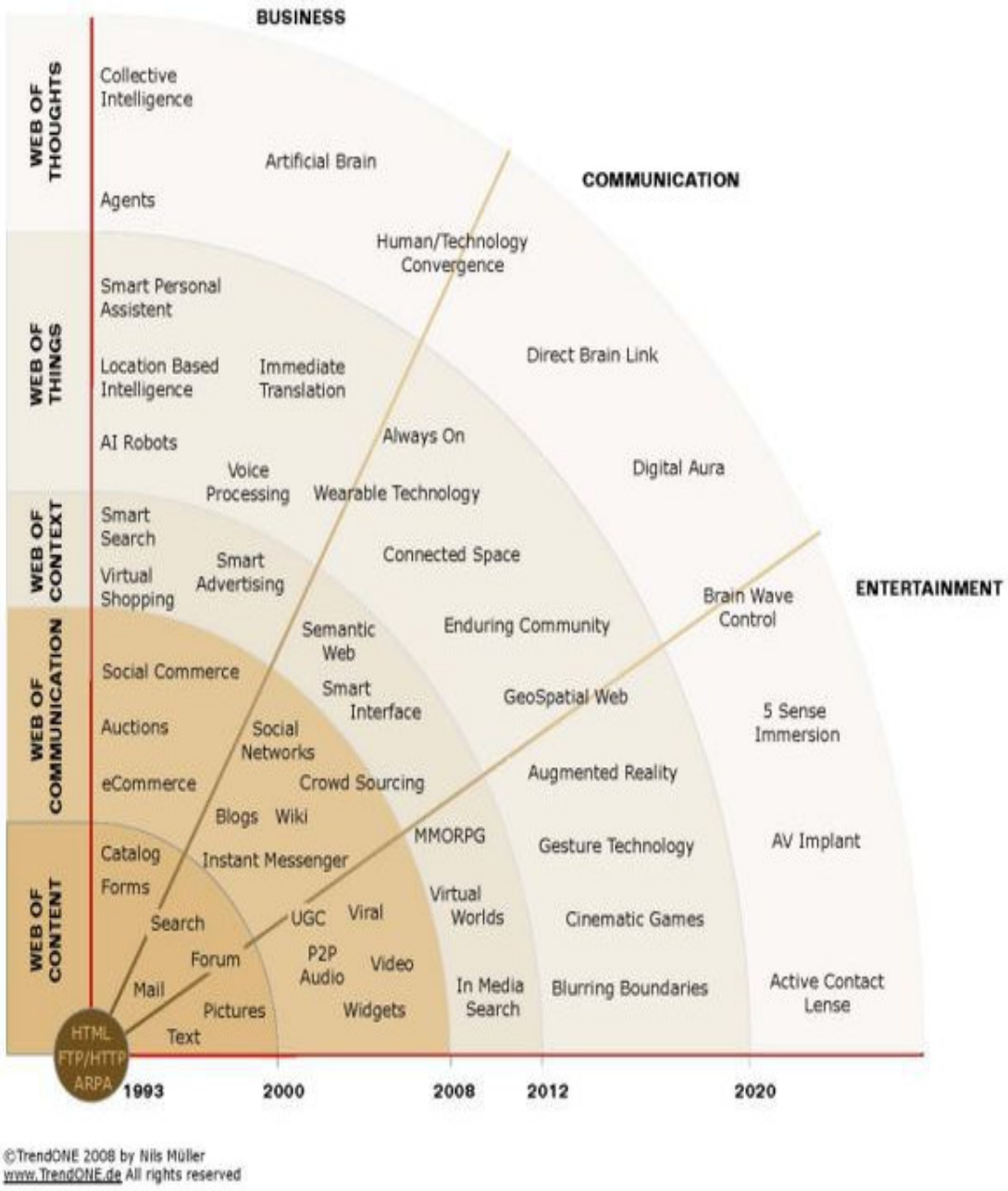

Figura 1 – Evolução da web
Fonte: Flat Education (2015, p. 1).

Em uma outra forma de definição, Aghaei, Nematbakhsh e Farsani (2012) caracterizam a Web 1.0 como a *web* da cognição, a Web 2.0 como a Web da comunicação, a Web 3.0 como a *web* da cooperação e a Web 4.0 como a Web da integração.

Além das mudanças inerentes ao acesso, produção e uso do conteúdo, o desenvolvimento de aplicações para a *web* tem passado por mudanças que em

partes estão relacionadas às exigências dos usuários por uma interface das aplicações *web* com mais recursos e facilidade. Iniciando em 2005, essas são as aplicações *web* denominadas *Rich Internet Applications* (RIA).

Nas RIA uma parte relevante da lógica de negócio é interposta para o computador que faz o papel de cliente em um ambiente cliente-servidor como são caracterizadas as aplicações *web*. Essa evolução causou mudança de objetivos, técnicas e modelos de engenharia envolvido no desenvolvimento de sistemas *web* (TRAMONTANA; AMALFITANO; FASOLINO, 2013). Amalfitano et al. (2010) ressaltam que, durante os últimos anos, processos, tecnologias e ferramentas para desenvolvimento *web* têm evoluído consideravelmente. Em decorrência disso as aplicações *web* recentes são caracterizadas por usabilidade melhorada da sua interface e por proverem uma experiência do usuário mais interativa, dinâmica e satisfatória. Aplicações que atendem a esses requisitos são geralmente chamadas de RIA e desenvolvidas utilizando tecnologias como *Asynchronous Javascript and* XML (AJAX) (GARRETT, 2005), Silverlight (MICROSOFT SILVERLIGHT, 2010) ou Flex (ADOBE FLEX, 2015).

O surgimento das RIAs em relação às tecnologias utilizadas para implementá-las representa uma significativa evolução ao padrão de desenvolvimento baseado em HTML da Web 1.0. Contudo, embora HTML tenha sido fundamental para o desenvolvimento da *web*, ela definiu uma Internet baseada em hipertexto com navegadores interpretando e compondo páginas *web* estáticas. HTML não permite atualização parcial da tela (PAVLIĆ; PAVLIĆ; JOVANOVIĆ, 2012). A atualização parcial é possível, por exemplo, com o uso de JavaScript permitindo a manipulação de elementos específicos da página *web*.

3 MATERIAIS E MÉTODO

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

3.1 MATERIAIS

O Quadro 1 apresenta as ferramentas e as tecnologias que foram utilizadas para modelar e implementar o sistema.

Ferramenta / Tecnologia	**Versão**	**Referência**	**Finalidade**
Java Platform (*Java Development Kit (*JDK))	1.7.0_75	http://www.oracle.com/	Linguagem de programação
PostgreSQL	9.4.1	http://www.postgresql.org/	Banco de dados
pgAdmin	1.20.0	http://www.pgadmin.org/	Sistema de gestão de banco de dados
WildFly	8.1.0. Final	http://www.apache.org/	Servidor web para a aplicação
Hibernate	4.2.17	http://hibernate.org/orm/downloads/	*Framework* de mapeamento de banco de dados
Vraptor	4.1.4	http://www.vraptor.org/pt/download/	*Framework* MVC de desenvolvimento *web*
AngularJs	1.4.3	https://angularjs.org/	*Framework* JavaScript open-source
Bootstrap	3.3.5	http://getbootstrap.com/	*Framework* HTML, CSS, para desenvolvimento web responsivo
Apache Maven	3.0.5	https://maven.apache.org/index.html	Ferramenta de automação de compilação de projetos Java
Jasypt 1.9.2		http://www.jasypt.org/	Biblioteca de criptografia

Quadro 1 – Ferramentas e tecnologias utilizadas

3.2 MÉTODO

As atividades para a modelagem e a implementação do sistema foram organizadas de acordo com o modelo sequencial linear proposto por Pressman (2006). Ajustes nos requisitos foram realizados à medida que a implementação ocorria, mas uma primeira versão com a descrição e a modelagem dos requisitos foi elaborada no início do desenvolvimento. Os requisitos foram fornecidos por futuros usuários do sistema, professores do Departamento Acadêmico de Informática, da UTFPR, Câmpus Pato Branco. Esses usuários acompanharam o desenvolvimento das atividades de modelagem e de implementação do sistema.

As principais atividades realizadas foram:

a) Levantamento de requisitos

O levantamento de requisitos iniciou com futuros usuários do sistema apresentando a descrição geral dos requisitos funcionais e não funcionais pretendidos. Esses requisitos foram baseados nas avaliações de acompanhamento que são realizadas pelos alunos e professores, pelo interesse dos professores em promover alterações na metodologia atual de realização dessas avaliações e pela necessidade de elaborar as próprias avaliações no escopo de departamentos e cursos.

Ficou estabelecido como requisitos principais do sistema: um editor para a composição dos questionários de avaliação, a disponibilização dos questionários de avaliação por período e somente para os que os responderiam, a não identificação do respondente, mas a identificação de quem já respondeu ou ainda não respondeu para que não seja possível mais de uma resposta a um mesmo questionário pela mesma pessoa. E, ainda, a forma de acesso aos resultados das avaliações.

Também foram identificados os atores ou papéis: aluno, professor, coordenador, chefe de departamento e administrador. Um professor pode exercer o papel de professor e também de coordenador e/ou chefe de departamento.

b) Análise e projeto

Dos requisitos levantados foi modelado um diagrama de casos de uso, de classes e de entidades e relacionamentos do banco de dados. As tabelas do banco de dados foram descritas para identificação de campos de chaves primária e estrangeira, tipo de dado armazenado e se o campo pode ser nulo.

c) Implementação

O desenvolvimento do aplicativo foi realizado com as ferramentas e as tecnologias indicadas na Seção 3.1. Os testes foram realizados com o objetivo de identificar erros de codificação e para verificação do atendimento aos requisitos.

d) Implantação

O sistema foi implantado em um servidor disponibilizado para que testes de usuários pudessem mais facilmente ser realizados e para uso posterior do sistema.

4 RESULTADO

Este capítulo apresenta o resultado da realização deste trabalho. E está organizado em seções. Em 4.1 é apresentado o escopo com uma visão geral do sistema. A seção 4.2 apresenta a modelagem realizada. O sistema desenvolvido, apresentado por meio de suas telas, está na Seção 4.3 e exemplos dos códigos gerados para implementar as funcionalidades expostas por meio das telas são apresentados na Seção 4.4.

4.1 ESCOPO DO SISTEMA

O aplicativo tem o objetivo de ser utilizado para a elaboração e aplicação de avaliações de disciplinas, turmas, docentes e discentes. A avaliação é baseada em questionários que são elaborados no próprio sistema e disponibilizados para os respondentes. Somente terá acesso aos respectivos questionários o usuário que possuir permissão para respondê-los.

O aplicativo não permitirá identificar o aluno, professor ou servidor que respondeu o questionário. Apenas identificar se o respectivo usuário já respondeu uma avaliação atribuída a ele. E em tendo respondido, não poderá fazê-lo novamente.

Os questionários possuem período especificado para ficarem disponíveis para avaliação. Eles são disponibilizados em uma determinada data inicial e ficam disponíveis até uma data definida como final. Essas datas são pré-configuradas no sistema. Dessa forma, tornar os questionários disponíveis para resposta pode ser realizado de maneira automática, a partir das datas pré-definidas.

Os alunos estarão vinculados a disciplinas e poderão responder somente os questionários de avaliação das disciplinas que eles estiverem matriculados.

O professor realizará a avaliação somente das turmas para as quais ministra aulas e do coordenador do curso ao qual está vinculado e do chefe dos departamentos para os quais ele pertence. O professor terá acesso somente ao resultado das avaliações das turmas para as quais ele ministra aula.

O coordenador tem acesso aos resultados das avaliações realizadas pelos alunos do seu curso e dos respectivos professores. O chefe de departamento tem acesso aos resultados das avaliações realizadas pelos alunos e professores dos cursos pertencentes ao respectivo departamento.

O usuário administrador não terá acesso aos resultados das avaliações, mas poderá realizar todos os cadastros e a composição dos questionários. O coordenador do curso também poderá realizar o cadastramento e o gerenciamento dos questionários do seu curso. O vínculo de alunos e professores ao curso, as turmas e disciplinas é realizado pelo coordenador do curso.

Professores, alunos, chefes de departamento, coordenadores de curso realizam auto-avaliação.

4.2 MODELAGEM DO SISTEMA

A visão geral do sistema está centrada na funcionalidade essencial do sistema que é a composição dos questionários de avaliação e a realização das avaliações. Essa visão geral é apresentada por meio da Figura 1.

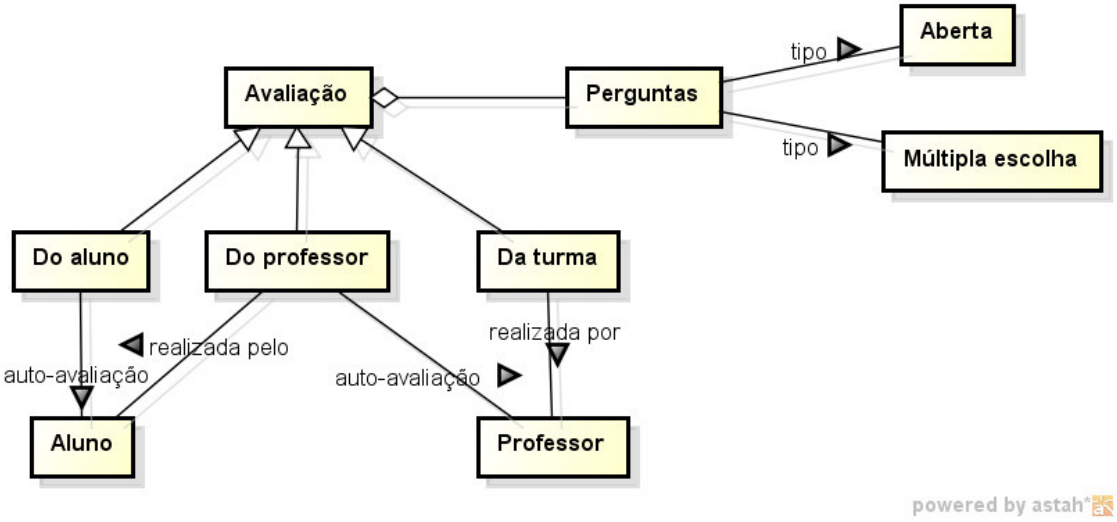

Figura 1 – Visão geral do sistema

De acordo com a representação da Figura 1, a avaliação é realizada pelos alunos e professores e são avaliados professores, turmas, coordenadores de curso e chefes de departamento. Esses dois últimos papéis são realizados por professores

e, portanto, são auto-avaliações quando eles próprios se avaliam e avaliações de funções quando são avaliados.

A listagem a seguir (Quadro 2) apresenta os requisitos funcionais identificados para o sistema.

Identificação	Nome	Descrição
RF01	Cadastro de alunos	Os alunos que podem responder questionários.
RF02	Cadastro de professores	Os professores que podem responder questionários.
RF03	Cadastro de perfis	São os tipos de usuários que utilizarão o sistema: administrador, coordenador, chefe de departamento, professor e aluno.
RF04	Cadastro de categorias de questionários	São os tipos de questionários, como auto-avaliação do aluno, auto-avaliação do professor, avaliação da turma, avaliação do professor, avaliação do coordenador e do chefe de departamento.
RF05	Cadastro de departamentos	Os departamentos aos quais pertencem os cursos e respectivos professores e alunos sendo avaliados.
RF06	Cadastro de curso	Cadastro de cursos aos quais pertencem os usuários do sistema. Um curso pertence a um departamento. Um curso possui um coordenador que é professor.
RF07	Cadastro de disciplinas	Disciplinas que pertencem a um curso.
RF08	Cadastro de turmas	As turmas que pertencem a um curso.
RF09	Relacionamento entre alunos e disciplinas	As disciplinas que o aluno cursa e avalia os professores das respectivas disciplinas.
RF10	Relacionamento entre professores e turmas	As turmas que o professor pode avaliar. São as turmas para as quais o professor ministra aula.
RF11	Cadastro de perguntas	As perguntas que compõem cada um dos questionários. Cada pergunta pode ter “n” alternativas e/ou uma resposta aberta.
RF12	Cadastro de questionário	Cadastro dos questionários.
RF13	Composição de questionário	Um questionário é composto por perguntas previamente cadastradas, que podem ser abertas ou de múltipla escolha.

Quadro 2 – Requisitos funcionais

O Quadro 2 apresenta os requisitos não-funcionais identificados para o sistema. Os requisitos não funcionais explicitam regras de negócio, restrições ao

sistema de acesso, por exemplo, requisitos de qualidade, desempenho, segurança e outros.

Identificação	Nome	Descrição
RNF01	Sistema *web*	A solução implementada deverá ser para ambiente *web*, independentemente da tecnologia adotada.
RNF02	Usuário que respondeu o questionário	Deverá haver controle dos usuários que responderam os questionários.
RNF03	Respostas a questionários	Usuários aluno e professor respondem o seu questionário de auto-avaliação. Usuários aluno respondem um mesmo tipo de questionário para professores distintos. Usuários professor respondem o mesmo tipo de questionário para turmas distintas. Usuários professor avaliam coordenador de curso e chefe de departamento.
RNF04	Sigilo de respostas a questionários	Não poderá, de forma alguma, ser identificado o usuário que respondeu o questionário. Apenas identificados os usuários que já responderam um determinado tipo de questionário, visando evitar que um mesmo usuário responda um mesmo questionário mais de uma vez.
RNF05	Usuários que respondem questionários	Somente usuários identificados (cadastrados no sistema) podem responder aos questionários. Os usuários identificados são os relacionados ao curso ou cursos que está respondendo ao questionário.
RNF06	Usuários	Um usuário poderá pertencer a mais de um curso ou departamento, mas no momento da resposta do questionário ele deverá ser identificado como pertencente ao departamento/curso do qual responderá o questionário.
RNF07	Usuário chefe departamento	O usuário chefe de departamento possui permissão de visualizar todos os resultados dos cursos do seu departamento.
RNF08	Visualizar respostas	As respostas do questionário de auto-avaliação podem ser visualizadas de forma identificada apenas por quem respondeu o questionário, ou seja, o aluno ou professor que se auto-avaliou. Cada professor visualiza as respostas da turma para a sua disciplina. Cada coordenador visualiza as respostas de todos os tipos de avaliação do seu curso. As respostas não podem ser identificadas. O chefe do departamento visualiza as

		respostas dos cursos do seu departamento, de todos os tipos de avaliação.
RNF09	Controle de avaliações disponíveis e realizadas	Informar o usuário, ao logar-se, quais as avaliações ele tem para realizar e quais ele já realizou. A tabela QuestionarioDisponivel permite identificar para quais disciplinas o aluno pode avaliar o respectivo professor. Assim é possível saber quais avaliações o aluno pode fazer. Para saber quais avaliações o usuário já respondeu existe a tabela UsuarioRespondido. A tabela Disciplinas permite identificar para quais disciplinas o professor ministra aulas e por meio do período daquela disciplina identificar quais turmas ele pode avaliar. A tabela QuestionarioResposta armazena a identificação do usuário que respondeu e de qual avaliação ele respondeu. Assim, para aluno e professor é possível identificar as avaliações pendentes e as que já respondeu.
RNF10	Formulário de cadastro de questionários e perguntas	O formulário para cadastro dos questionários e perguntas deverá conter uma espécie de editor de texto para permitir ao usuário cadastrar uma pergunta e as suas alternativas, se a pergunta é de alternativas. A pergunta pode ser de resposta aberta. Nesse caso, o usuário terá um campo de texto para inserir a resposta. As alternativas serão apresentadas para o usuário como botões de rádio, caixa de combinação ou outro elemento que permita ao usuário selecionar uma ou mais dentre as alternativas informadas.

Quadro 3 – Requisitos não funcionais

Os casos de uso estão representados na Figura 2. O ator aluno tem permissão para responder o seu questionário de auto-avaliação e de avaliar os professores das disciplinas que ele cursa. O ator professor tem permissão para realizar a sua auto-avaliação e avaliar as turmas para as quais ele ministra aulas. O ator coordenador tem acesso aos relatórios de resultados das disciplinas do seu curso. O ator chefe de departamento tem acesso aos relatórios dos cursos que compõem o departamento. O ator administrador realiza os cadastros necessários e compõe os questionários. Ele não tem acesso aos resultados das avaliações.

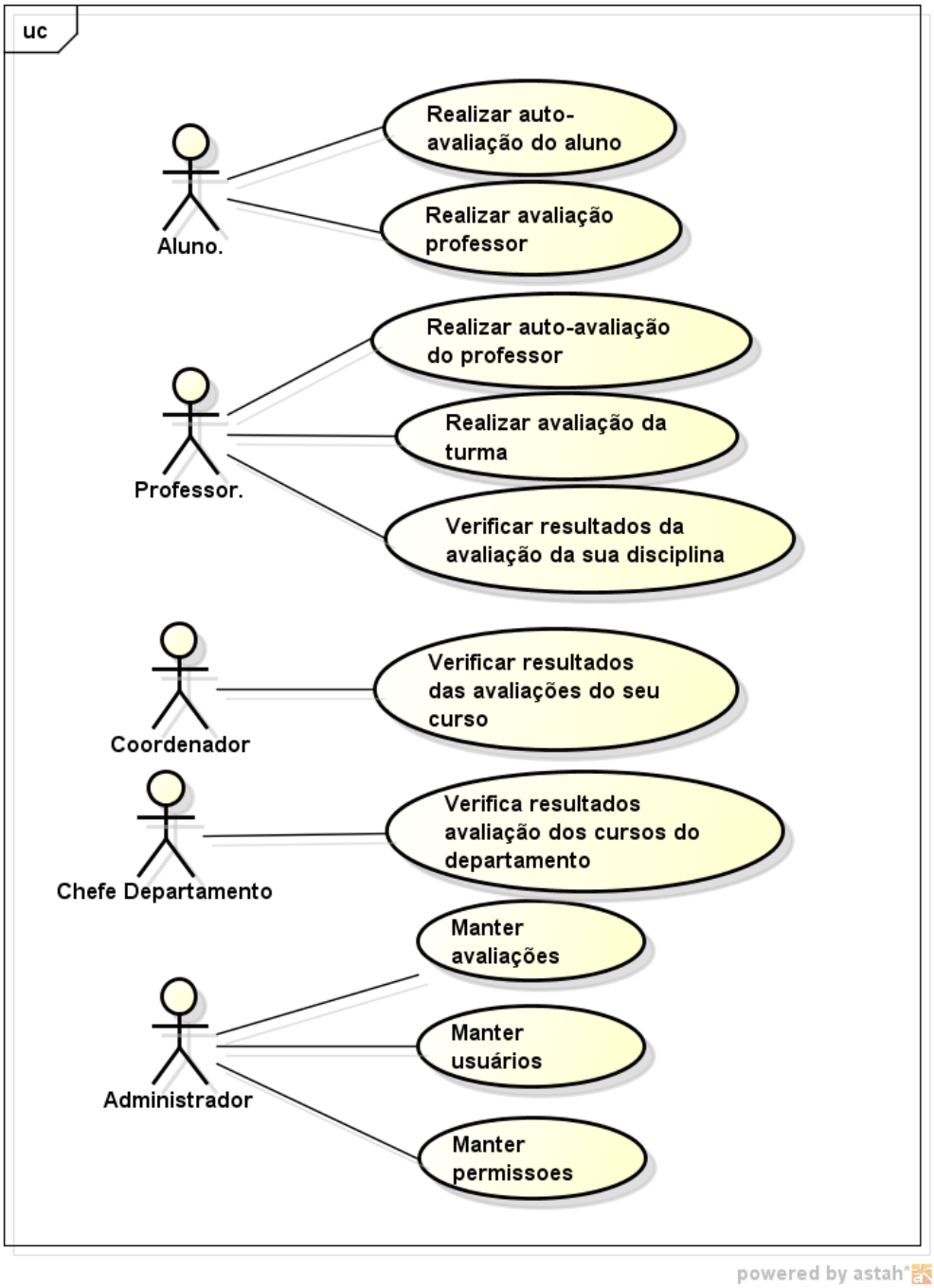

Figura 2 – Diagrama de casos de uso

A Figura 3 apresenta o diagrama de classes do sistema. A classe usuário é herdada pelas classes que representam os tipos de usuários ou atores do sistema. A classe questionário está relacionada à categoria do questionário, à composição do questionário por meio de perguntas definidas e das respostas a essas perguntas.

Além disso, há o conjunto de classes relacionado ao departamento, curso, turma e disciplina que permitem definir as avaliações permitidas a cada tipo de usuário.

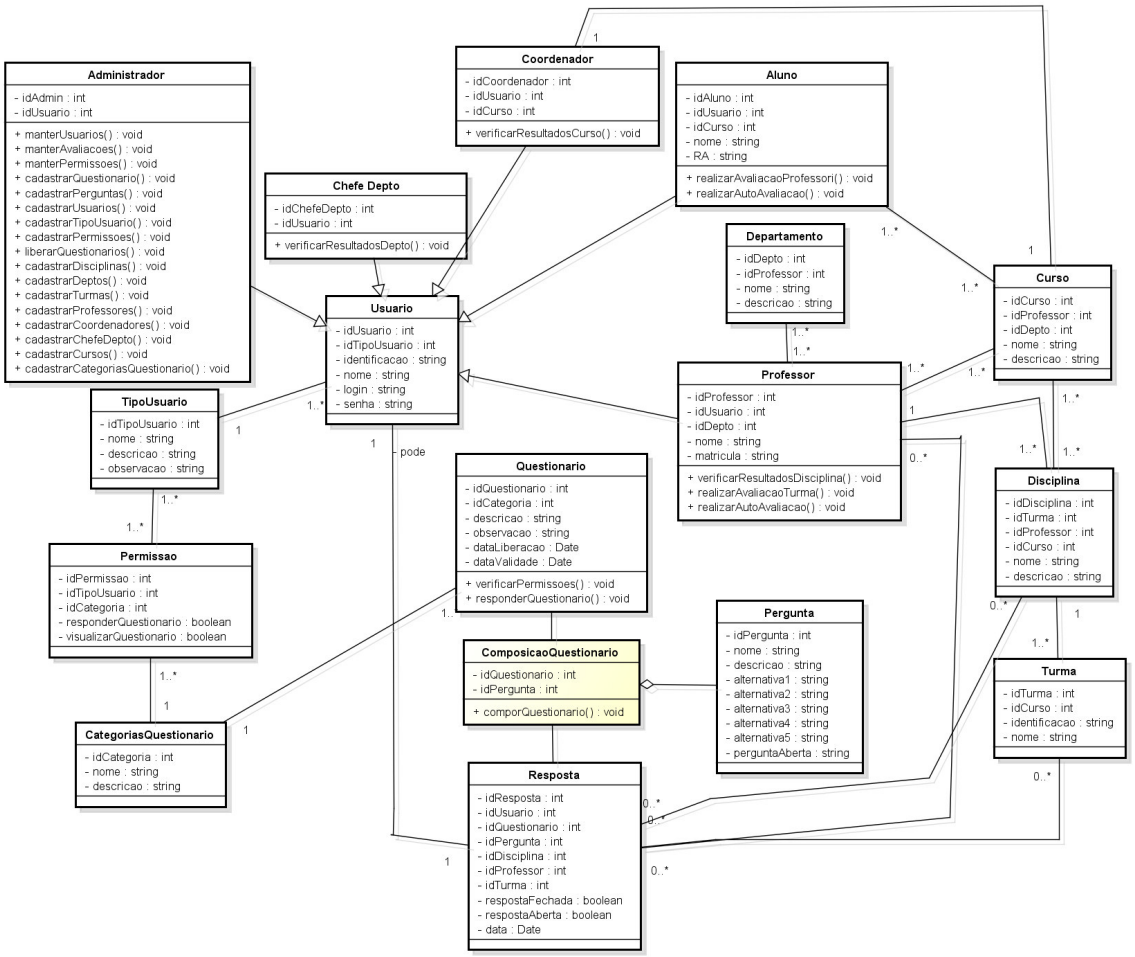

Figura 3 – Diagrama de classes

A Figura 4 apresenta o diagrama de entidades e relacionamentos do banco de dados.

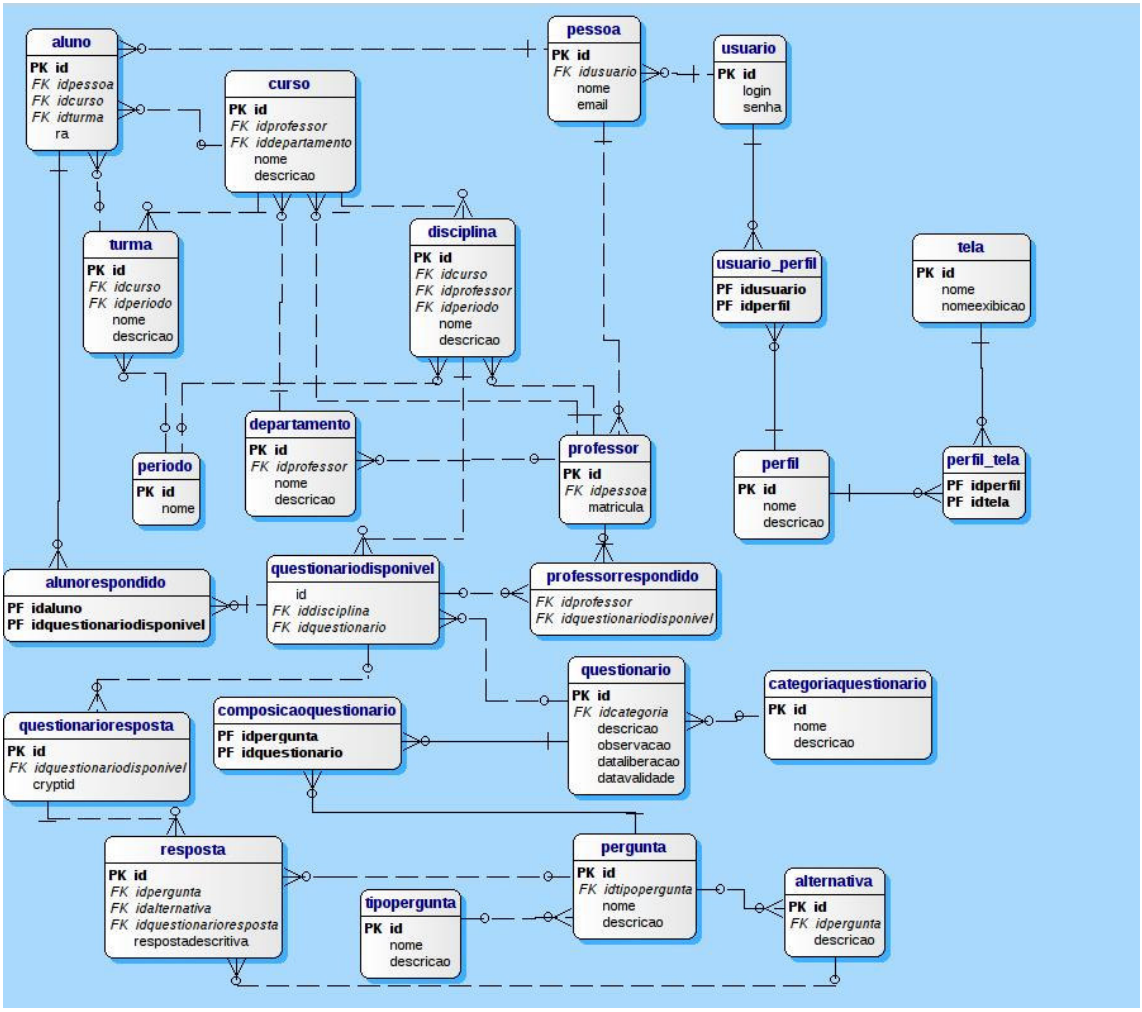

Figura 4 – Diagrama de entidades e relacionamentos do banco de dados

Os quadros a seguir contêm a descrição das tabelas apresentadas no diagrama da Figura 4. O Quadro 4 apresenta a tabela de cadastro de usuários.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
login	Texto	Não	Não	Não	
senha	Texto	Não	Não	Não	Deve ser criptografada

Quadro 4 – Tabela Usuários

No Quadro 5 está a descrição da tabela para armazenar os dados dos perfis de usuários. Os perfis de usuários são os atores do sistema.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	

Quadro 5 – Tabela Perfil

No Quadro 6 está a descrição da tabela para armazenar o vínculo entre os usuários e os perfis de sistema.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idusuario	Numérico	Não	Sim	Sim	
idperfil	Numérico	Não	Sim	Sim	Da tabela TiposUsuarios

Quadro 6 – Tabela usuario_perfil

No Quadro 7 está a descrição da tabela para armazenar as categorias de questionários.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	

Quadro 7 – Tabela CategoriasQuestionarios

No Quadro 8 está a descrição da tabela para armazenar os dados dos professores. Um curso pertence a um departamento.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
idpessoa	Numérico	Não	Sim	Sim	Da tabela Departamentos
Matricula	Texto	Não	Não	Não	Identificação do professor

Quadro 8 – Tabela Professores

No Quadro 9 está a descrição da tabela para armazenar os dados de departamentos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
Id	Numérico	Não	Sim	Não	
idprofessor	Numérico	Não	Não	Sim	Chefe de Departamento
nome	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	

Quadro 9 – Tabela Departamentos

No Quadro 10 está a descrição da tabela para armazenar os dados de turmas.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
idcurso	Numérico	Não	Não	Sim	Da tabela Cursos
Idperiodo	Numérico	Não	Não	Sim	
nome	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	

Quadro 10 – Tabela Turmas

No Quadro 11 está a descrição da tabela para armazenar os dados de disciplinas.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
idperiodo	Numérico	Não	Sim	Sim	Da tabela Periodo
idProfessor	Numérico	Não	Não	Sim	Da tabela Professores
idCurso	Numérico	Não	Sim	Sim	Da tabela Cursos
nome	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	

Quadro 11 – Tabela Disciplinas

No Quadro 12 é mostrada a descrição da tabela Periodo, que armazena os períodos a que a disciplina pertence.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
Id	Numérico	Não	Sim	Não	
Nome	Texto	Não	Não	Não	

Quadro 12 – Tabela Períodos

No Quadro 13 está a descrição da tabela para armazenar os dados dos cursos. Um curso pertence a um departamento.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
Id	Numérico	Não	Sim	Não	
idProfessor	Numérico	Não	Não	Sim	Da tabela Professores
idDepartamento	Numérico	Não	Não	Sim	Da tabela Departamentos
nome	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	

Quadro 13 – Tabela Cursos

No Quadro 14 está a descrição da tabela para armazenar os dados dos alunos. Essa tabela permite identificar que aluno pode responder quais questionários.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
Id	Numérico	Não	Sim	Não	
Idpessoa	Numérico	Não	Não	Sim	
idCurso	Numérico	Não	Sim	Sim	Da tabela Cursos
RA	Numérico	Não	Não	Não	Registro acadêmico

Quadro 14 – Tabela Alunos

No Quadro 15 está a tabela para vincular as disciplinas aos questionários. Essa tabela permite identificar qual questionário ficará disponível para qual disciplina.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
Id	Numérico	Não	Sim	Sim	
IdDisciplina	Numérico	Não	Não	Sim	Da tabela Disciplinas
Idquestionario	Numérico	Não	Não	Sim	Da tabela Questionários

Quadro 15 – Tabela questionariodisponivel

No Quadro 16 está a descrição da tabela para armazenar os dados dos questionários.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
idCategoria	Numérico	Não	Não	Sim	Da tabela Categorias
Status	Texto	Não	Não	Não	Status: Salvo, Em curso ou Finalizado
descricao	Texto	Sim	Não	Não	
observacao	Texto	Não	Não	Não	
Dataliberacao	Data	Sim	Não	Não	
Datavalidade	Data	Não	Não	Não	

Quadro 16 – Tabela Questionarios

No Quadro 17 está a descrição da tabela para armazenar as perguntas elaboradas e que serão utilizadas para compor um questionário.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
Idtipopergunta	Numérico	Não	Não	Sim	Da tabela TipoPergunta
nome	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	

Quadro 17 – Tabela Perguntas

No Quadro 18 está a descrição da tabela TipoPergunta, responsável por armazenar os tipos de perguntas possíveis no sistema.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
nome	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	

Quadro 18 – Tabela TipoPergunta

No Quadro 19 está a descrição da tabela para armazenar a composição de questionários. Um questionário é composto por um conjunto de perguntas.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idQuestionario	Numérico	Não	Sim	Sim	Da tabela Questionários
idPergunta	Numérico	Não	Sim	Sim	Da tabela Perguntas

Quadro 19 – Tabela ComposicaoQuestionarios

No Quadro 20 está a descrição da tabela Alternativa, que armazena os dados das alternativas das perguntas.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
idpergunta	Numérico	Não	Não	Sim	
descricao	Texto	Não	Não	Não	

Quadro 20 – Tabela de Alternativa

No Quadro 21, fica a descrição da tabela QuestionarioRespota, responsável por armazenar a referência à tabela QuestionarioDisponivel, para que depois seja possível identificar qual respostas são para qual disciplina.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
idquestionariodisponivel	Numérico	Não	Sim	Sim	Da tabela Questionariodisponivel

Quadro 21 – Tabela QuestionarioRespota

No Quadro 22, fica a tabela que armazena as respostas das perguntas.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Numérico	Não	Sim	Não	
idpergunta	Numérico	Não	Não	Sim	Da Tabela Pergunta
idalternativa	Numérico	Sim	Não	Sim	Da Tabela Alternativa
idquestionarioresposta	Numérico	Não	Não	Sim	Da Tabela Questionario Resposta
respostadescritiva	Texto	Sim	Não	Não	Campo criptografado

Quadro 22 – Tabela de Respostas

No Quadro 23, fica a descrição da tabela AlunosRespondidos, que armazena somente a referência do aluno que respondeu um questionário, e o questionário respondidos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idusuario	Numérico	Não	Sim	Sim	Da Tabela de Usuários
idquestionario disponivel	Numérico	Não	Sim	Sim	Da Tabela de Questionários disponíveis

Quadro 23 – Tabela de UsuariosRespondido

4.3 APRESENTAÇÃO DO SISTEMA

O leiaute básico do sistema é composto por duas partes principais: o menu de navegação, que fica localizado na parte superior da tela, e o setor de conteúdos, que é responsável por apresentar a tela requisitada pelo usuário por meio de itens de menu ou *link* no sistema.

A Figura 5 apresenta essas duas partes, com o setor de conteúdo mostrando somente a tela inicial, após ser efetuado o *login.*

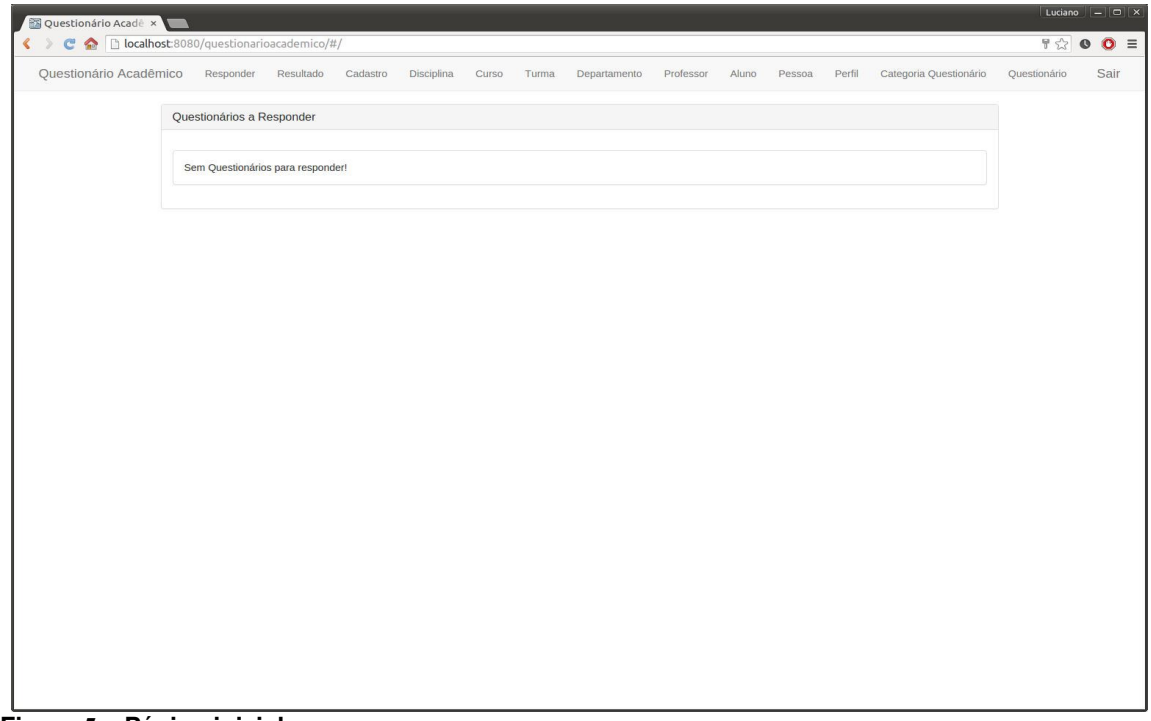

Figura 5 – Página inicial

O menu do sistema é filtrado de acordo com o perfil do usuário logado. Na Figura 5 está um exemplo para um usuário do tipo “Administrador”. A Figura 6 mostra a tela que é apresentada para um usuário do tipo “Aluno”.

Figura 6 – Menu filtrado para usuário aluno

Ao ser clicado em uma das opções do menu, o setor de conteúdo mostrará a lista dos registros requisitados. Na Figura 7, é apresentada a tela de questionário com a listagem dos questionários cadastrados.

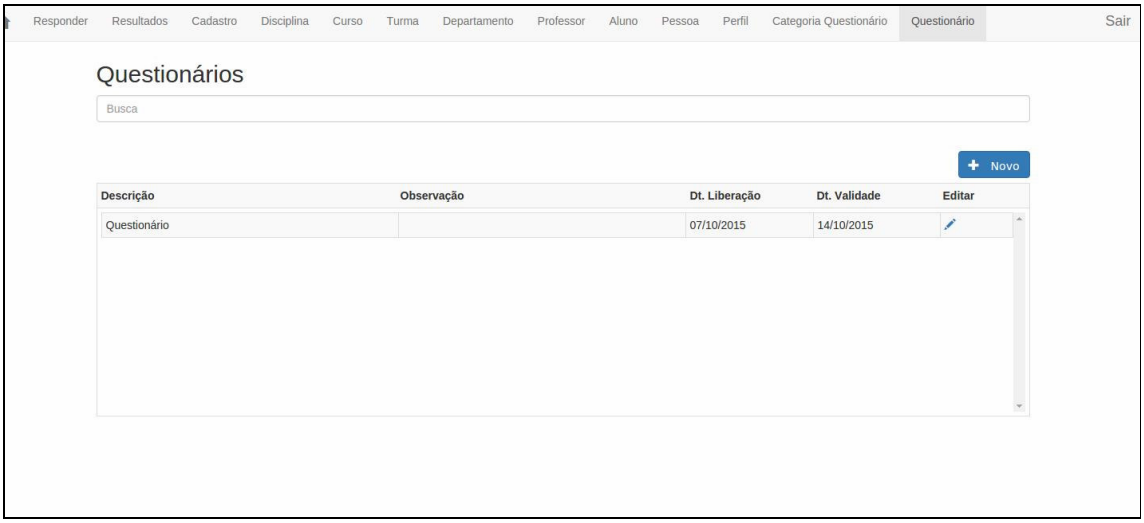

Figura 7 – Lista de questionários

A partir dessa lista é possível inserir um novo registro clicando em “Novo +”. O formulário para cadastro de um novo questionário é apresentado na Figura 8.

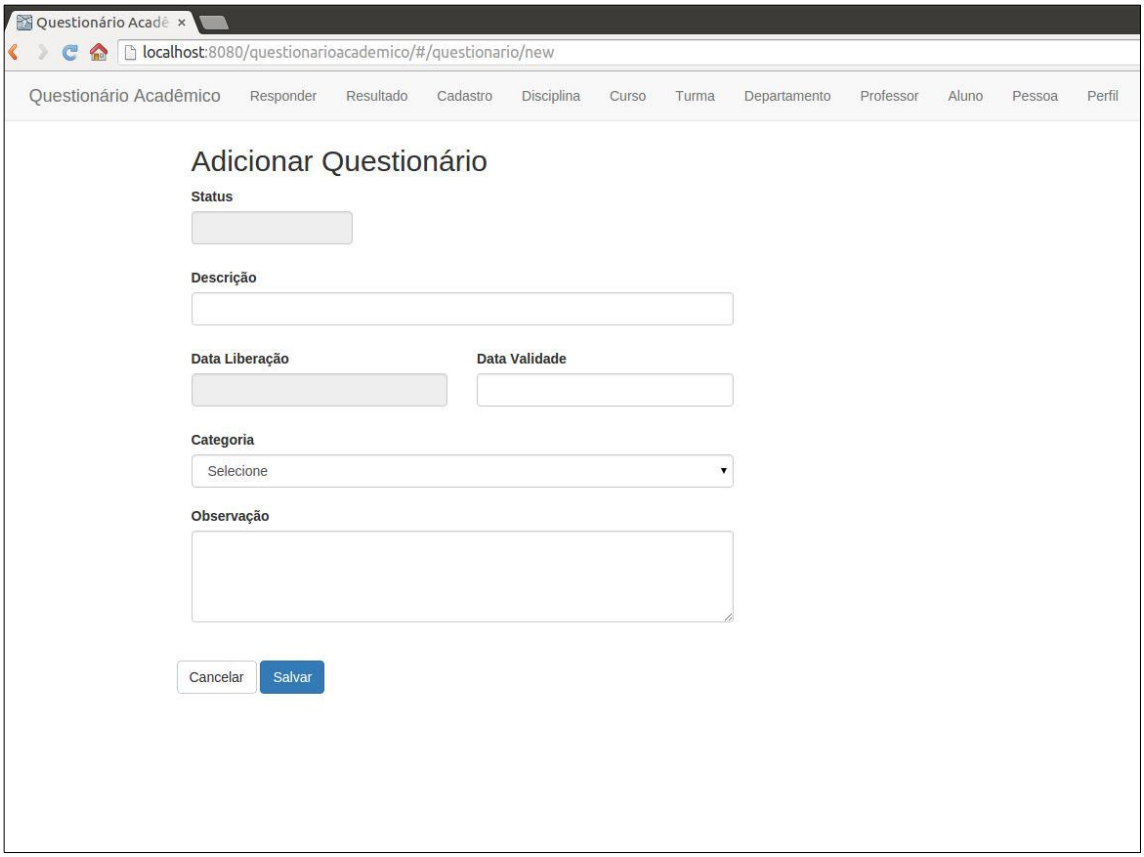

Figura 8 – Adicionar questionário

A validação dos dados nos campos é realizada por meio das validações disponíveis pela especificação HTML5. Por exemplo, ao ser clicado no botão “Salvar” a mensagem “Preencha este campo” é apresentada como mostra a área destacada na Figura 9.

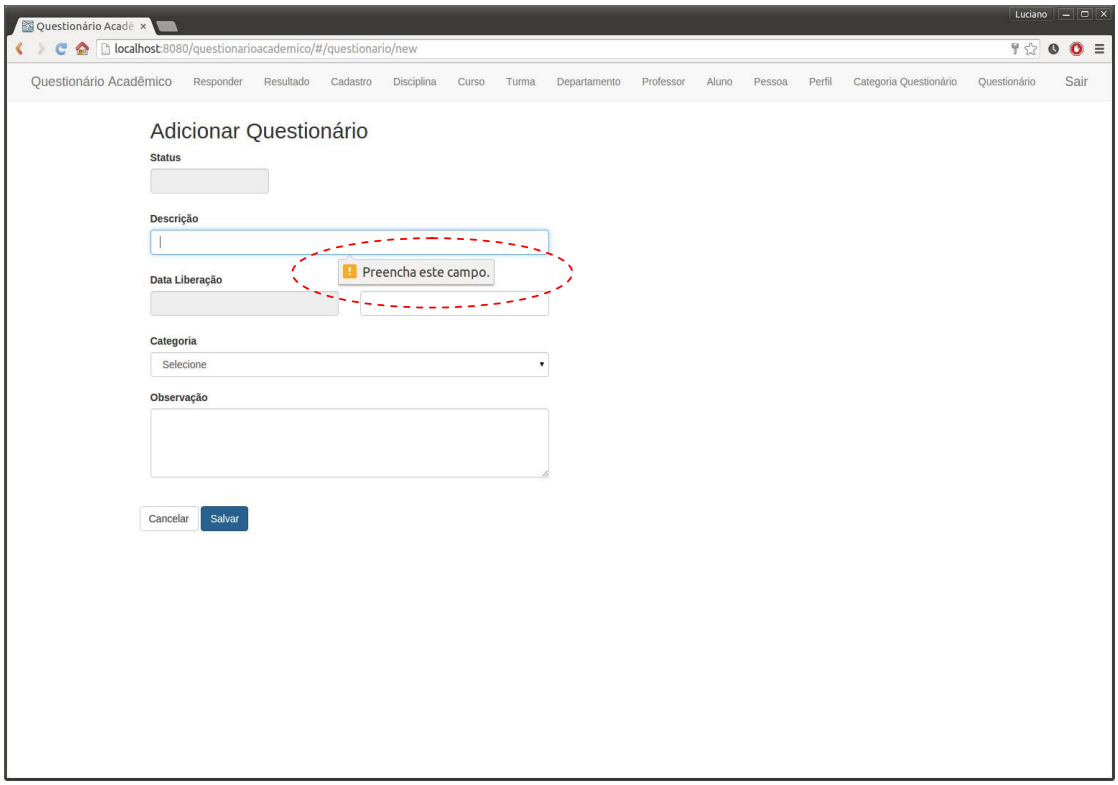

Figura 9 – Validação dos campos

Após preenchidos todos os campos necessários e depois de clicado no botão "Salvar", o processo termina com o redirecionamento do setor de conteúdo para a listagem de questionário novamente. A edição de registros já salvos é feita clicando no ícone de edição na lista (Figura 10).

Figura 10 – Ícone de edição de itens

Ao ser clicado no ícone que abre o formulário para edição, o sistema direciona para a tela de edição do respectivo registro (Figura 11).

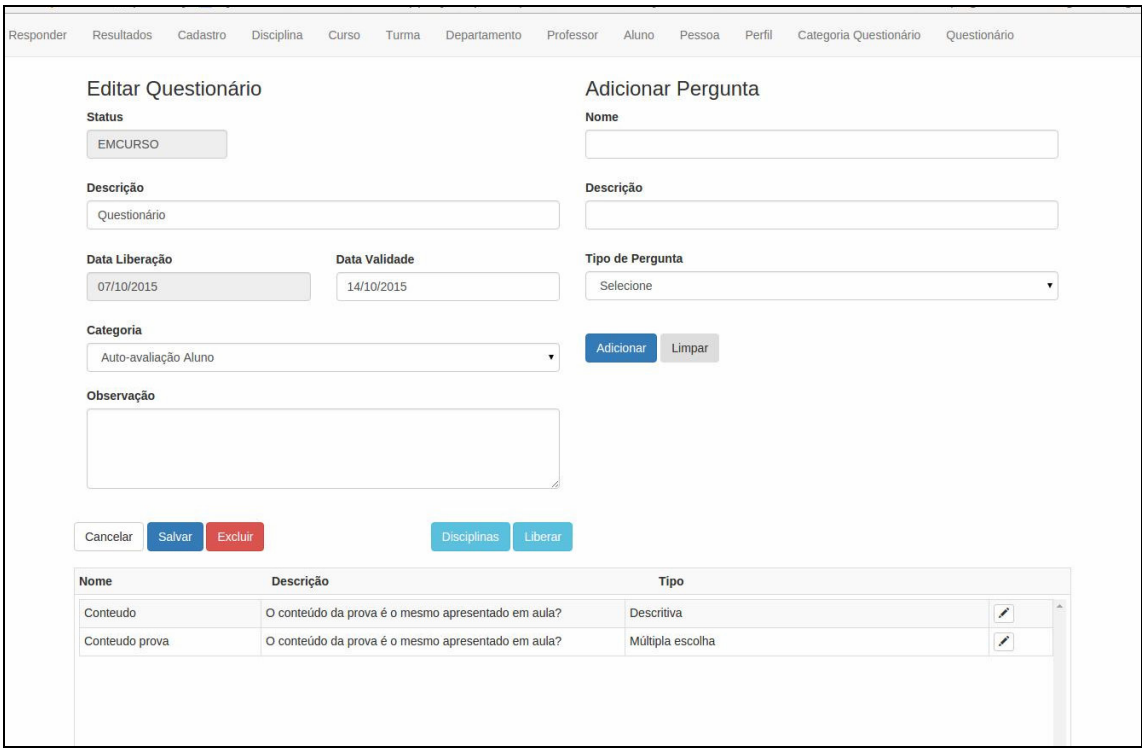

Figura 11 – Tela de edição de questionário

A edição permite alterar dados do registro e também de excluir o respectivo registro. Essas duas ações direcionarão para a tela de listagem inicial. Todos os botões “Excluir” do sistema possuem a chamada de uma confirmação de exclusão (Figura 12).

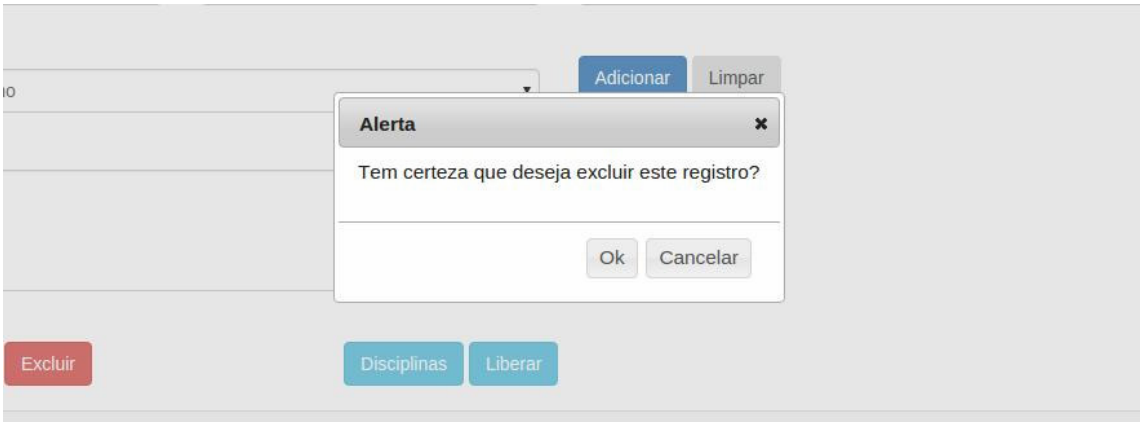

Figura 12 – Confirmação de exclusão de registro

4.4 IMPLEMENTAÇÃO DO SISTEMA

Como apresentado na Seção 4.3, o leiaute do sistema é composto por duas partes principais. O leiaute é elaborado pelo AngularJs, mais especificamente o “Ui-router” que organiza as partes da interface por meio de estados (“state”), que podem ou não ter um endereço ou *Uniform Resource Locator* (*URL).* Pelo fato de o AngularJs ser um *framework* que segue o padrão *Model-View-Controller* (MVC*),* os estados do Ui-router também o fazem. Dessa forma, cada *state* conterá uma *view* ou página HTML que será chamada de *template,* conterá também um *controller,* que fica responsável pelo comportamento da página e do modelo (*model*). Um exemplo de estado do *ui-router* é apresentado na Listagem 1.

```
$stateProvider
        .state('app', {
                'abstract': true,
                views: {
                        'menu@': {
                                templateUrl:'view/menu.html',
                                controller: "MenuController",
                        }
                },
                resolve: {
         authorize: function(authorization) {
           return authorization.authorize();
                },
         identity: function(authService){
           return authService.identity();
         }
                }
        })
        .state("home", {
                parent: 'app',
                url: "/",
                views:{
                        "content@" : {
                                templateUrl:'view/home.html',
                                controller: "HomeController",
                        }
                },
                data: {
                        roles: ["ALUNO", "ADMINISTRADOR", "COORDENADOR", "CHEFE", "PROFESSOR"]
                }
        })
        .state("login", {
                url: "/login",
                views: {
                        'content@': {
                                templateUrl:'view/login.html',
                                controller: "MainController"
                        }
                },
                data: {
                roles: []
                }
        })
```

Listagem 1 – Exemplo de state do ui-router

Na Listagem 1 é possível verificar a declaração do estado “app”, que não possui uma URL declarada e possui o atributo “abstract” como “true”. Logo, esse *state* não serve como um estado de navegação. Ele serve apenas para armazenar uma referência ao *template* de menu (*@menu*) juntamente com *controller* “MenuController”, que ficará responsável pelo comportamento do menu de navegação. Logo abaixo, fica o estado “home”, nele é possível perceber que o estado “app” foi declarado como “parent”. Ao fazer isso está sendo informado ao *ui-router* que o estado “home” herdará as características de “app”, que entre elas está o menu.

É dessa forma, que o AngularJs junto com Ui-Router fazem a junção dos dois *templates* em uma única página. Esses *templates* são salvos em arquivos *html* diferentes. Na Listagem 2 está o *template* de menu.

```
<nav id="main-menu" class="navbar navbar-default navbar-fixed-top">
  <div class="navbar-header">
    <a class="navbar-brand" ui-sref="home">Questionário Acadêmico</a>
      <ul class="nav navbar-nav">
            <li ng-repeat="tela in telas"  ui-sref-active="active">
          <a ui-sref="{{tela.nome}}">{{tela.nomeExibicao}}</a>
        </li>
           </ul>
  </div>
  <a class="navbar-brand" ng-click="logout()" href="#/">Sair</a>
</nav>
```

Listagem 2 – Template de menu

A Listagem 3 apresenta o *template* home.

```
<div class="home">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h3 class="panel-title">Questionários a Responder</h3>
    </div>
    <div class="panel-body">
      <div class="panel panel-default">
        <div class="panel-body">
          <div class="list-group">
            <div class="list-group-item" ng-repeat="questionariodisponivel in
questionariodisponivels">
                        {{questionariodisponivel.questionario.descricao}} |
{{questionariodisponivel.disciplina.nome}}
                        <a class="btn btn-default btn-xs pull-right" ui-sref="responder">Responder</a>
                    </div>
          </div>
        </div>
      <div class="panel-body" ng-if="questionariodisponivels.length===0 || !questionariodisponivels">Sem
Questionários para responder!</div>
      </div>
    </div>
  </div>
```

```
</div>
```

Listagem 3 – Template de home

Na declaração dos estados na Listagem 1, percebe-se que na parte das *views* é usada a notação “@menu” para o *template* de menu e “@content” para o *template* de *home*. Essa notação é usada pelo Ui-Router para renderizar os dois estados em uma única página, que é chamada de “index.jsp”. Essa página é responsável por carregar todas as partes constituintes do sistema, no que diz respeito à interface.

```
<!doctype html>
<html ng-app="QuestionarioAcademico" id="ng-app">
<head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <meta name="description" content="QuestionarioAcademico">
        <meta name="author" content="Luciano Heleno da Rosa">

        <!-- scripts e links -->

        <title>Questionário Acadêmico</title>

</head>
<body ng-controller="MainController">

        <!-- Aqui é rederizada as views menu e o conteudo requisitado-->
        <div ui-view="menu"></div>
        <div ui-view="content" class="container"></div>

</body>
</html>
```

Listagem 4 – Index.jsp

Na Listagem 4 é possível verificar que existem duas *divs* que possuem uma *tag* especial chamada *ui-view*. É nessas *divs* que o Ui-Router renderizará o conteúdo de cada *template*, respectivamente. O menu e o *template* da tela *home* ou qualquer *template* que for requisitado.

Os *controllers* são responsáveis pelo comportamento das páginas. Toda ação que representa uma funcionalidade no sistema é declarada nesses *controllers*. Como por exemplo, salvar, excluir, editar, etc. Essas ações são feitas com o auxílio de um outro conceito do AngulasJs, o Two-way-databind. O *template* apresentado na Listagem 5 apresenta o código que utiliza esses conceitos.

```
<h2 ng-show="model.curso.id">Alterar Curso</h2>
<h2 ng-hide="model.curso.id">Adicionar Curso</h2>
<form name="mainForm" ng-submit="save()">
        <div class="form-group">
                <div class="row">
                        <div class="col-xs-6">
                                <label>Nome</label><input type="text" class="form-control"
name="nome" ng-model="model.curso.nome" required>
                                <validate-msg-for field="mainForm.nome"></validate-msg-for>
                        </div>
                </div>
                <div class="row">
                        <div class="col-xs-6">
                                <label>Descrição</label>
                                <input type="text" class="form-control"  name="descricao" class="span4"
ng-model="model.curso.descricao" ng-required="true">
                                <validate-msg-for field="mainForm.descricao"></validate-msg-for>
                        </div>
                </div>
                <div class="row">
                        <div class="col-xs-6">
                                <label>Coordenador do Curso</label>
                                <select id="professores" class="form-control"  ng-
model="model.curso.professor" name="professor" ng-options="t.pessoa.nome for t in professores track by
t.id" ng-required="true">
                                        <option value="">Selecione um professor</option>
                                </select>
                                <validate-msg-for field="mainForm.professor"></validate-msg-for>
                        </div>
                </div>
                <div class="row">
                        <div class="col-xs-6">
                                <label>Departamento</label>
                                <select id="departamentos" class="form-control"  ng-
model="model.curso.departamento" name="departamento" ng-options="d.nome for d in departamentos track
by d.id" ng-required="true">
                                        <option value="">Selecione um Departamento</option>
                                </select>
                                <validate-msg-for field="mainForm.departamento"></validate-msg-for>
                        </div>
                </div>
                <br>
                <a ui-sref="curso" class="btn btn-default">Cancelar</a>
                <button type="submit" class="btn btn-primary">Salvar</button>
                <button type="button" ng-click="destroy()" ng-show="model.curso.id" class="btn btn-
danger">Deletar</button>
        </div>
</form>
```

Listagem 5 – Template de curso

Na Listagem 5 é possível verificar que em cada campo ou *input*, há uma *tag* chamada ng-model, ela é responsável por mapear esse campo para um objeto e/ou atributo dependendo da necessidade. No caso da listagem, ao digitar um valor no campo “nome”, por exemplo, o AngularJs criará um objeto de acordo com a notação presente dentro da *tag* ng-model (*model*.curso.nome), criando um objeto *model* com

um outro objeto dentro dele chamado curso e por sua vez conterá um atributo chamado nome, com o valor digitado no campo. Em seguida, esse novo objeto *model* poderá ser usado no *controller*, como pode ser visto a seguir na Listagem 6.

```
App.controller("CursoControllerEdit", ['$scope', '$location', 'CursoResource', 'ProfessorResource',
                        'DepartamentoResource', '$window', '$stateParams',

    function($scope, $location, CursoResource, ProfessorResource, DepartamentoResource, $window,
$stateParams){

                var root = '/curso/';

                ProfessorResource.query(function (res) { $scope.professores = res; });
                DepartamentoResource.query(function (res) { $scope.departamentos = res; });

                masterUpdate($scope, $stateParams, $window, $location, CursoResource, root);
                masterDelete($scope,$stateParams,$window, $location, CursoResource, root);

        }
]).controller("CursoControllerNew", ['$scope', '$location', 'CursoResource', 'ProfessorResource',
'DepartamentoResource',
    function($scope, $location, CursoResource, ProfessorResource, DepartamentoResource){

                var root = '/curso/';
                var emptyObj = {curso: {
                        "id":0,
                        "nome":"",
                        "descricao":""
                }};

                ProfessorResource.query(function (res) { $scope.professores = res; });
                DepartamentoResource.query(function (res) { $scope.departamentos = res; });

                masterCreate($scope, $location, CursoResource, root, emptyObj);
        }

]).controller("CursoControllerList", ['$scope', '$location', 'CursoResource',
   function($scope, $location, CursoResource){
                masterRead($scope, $location, CursoResource);
        }
]);
```

Listagem 6 – Controllers de curso

O objeto criado pelo ng-model fica automaticamente dentro da variável especial do AngularJs chamada “$scope”, que por sua vez poderá ser usada para qualquer ação presente no *controller*. Essas ações são métodos simples em JavaScript, que podem ser referenciados nas páginas, como por exemplo, no evento *click* de um botão ou no evento *submit* de um formulário. Tomando como base o *template* da Listagem 5, vê-se que existe um método sendo chamado no evento *submit* do formulário (<form name="mainForm" ng-submit="save()">). Esse método está declarado dentro do método “masterCreate()” no *controller*

"CursoControllerNew" da Listagem 6, que para efeito de reutilização de código está em um outro arquivo. A Listagem 7 apresenta esse método.

```
masterCreate = function($scope, $location, Resource, root, emptyObj) {
        $scope.model = new Resource(emptyObj);
        $scope.save = function() {
                $scope.model.$save(function(res) {
                        $location.path(root);
                });
        }
}
```

Listagem 7 – Método masterCreate

O método "save" da Listagem 7 precisa ser declarado dentro da variável "$scope" para poder ser referenciada na página ou no *controller*. Ao ser chamado, ele usará os dados digitados na página para enviar ao servidor que ficará responsável por salvar esses dados no banco de dados.

Na parte servidor da aplicação, também é usado um outro *framework* MVC de desenvolvimento, o Vraptor. Ele também possui um *controller* para receber os dados e devolver uma resposta para a página. Para o caso da tela de Cursos como no exemplo, existe o "CursoController", cujo código é apresentado na Listagem 8.

```
package br.edu.utfpr.pb.questionarioacademico.controller;

import javax.inject.Inject;

import br.com.caelum.vraptor.Consumes;
import br.com.caelum.vraptor.Controller;
import br.com.caelum.vraptor.Delete;
import br.com.caelum.vraptor.Get;
import br.com.caelum.vraptor.Path;
import br.com.caelum.vraptor.Post;
import br.com.caelum.vraptor.Put;
import br.com.caelum.vraptor.Result;
import br.edu.utfpr.pb.questionarioacademico.model.Curso;
import br.edu.utfpr.pb.questionarioacademico.repository.CursoRepository;

@SuppressWarnings("serial")
@Controller
@Path("cursos")
public class CursoController extends br.edu.utfpr.pb.questionarioacademico.controller.commons.Controller{

        private Result result;
        private CursoRepository repository;

        @Inject
        public CursoController(Result result, CursoRepository repository) {
                super(result);
                this.result = result;
                this.repository = repository;
        }
```

```
	/*CDI only*/
	protected CursoController() {
		this(null, null);
	}

	@Get
	@Path({"","/"})
	public void list() {
		serializer(repository.findAll()).serialize();
	}

	@Get
	@Path("/{start}/{limit}")
	public void list(Integer start, Integer limit) {
		serializer(repository.pagination(start, limit, null)).serialize();
	}

	@Get
	@Path("/{id}")
	public void find(Long id) {
		serializer(repository.find(id),true).serialize();
	}

	@Post
	@Path({"","/"})
	@Consumes("application/json")
	public void insert(Curso curso) {
		repository.insert(curso);
		result.nothing();
	}

	@Put
	@Path("/{curso.id}")
	@Consumes("application/json")
	public void update(Curso curso) {
		repository.update(curso);
		result.nothing();
	}

	@Delete
	@Path("/{curso.id}")
	public void delete(Curso curso) {
		repository.delete(curso);
		result.nothing();
	}
}
```

Listagem 8 – CursoController

Ao ser chamado, o método “save” enviará os dados digitados para o método “insert” da Listagem 8, que por sua vez, passará a responsabilidade de salvar esses dados para o “repository” (código na Listagem 9).

```
package br.edu.utfpr.pb.questionarioacademico.business.common;

import java.io.Serializable;
import java.lang.reflect.ParameterizedType;
import java.util.List;
import java.util.Map;

import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;

import br.edu.utfpr.pb.questionarioacademico.model.commons.Entity;

@SuppressWarnings("unchecked")
@Stateless
public class RepositoryImpl<T extends Entity, I extends Serializable> {

	@PersistenceContext(unitName = "entityManager")
	protected EntityManager entityManager;

	protected final Class<T> clazz = retornaTipo();

	public void insert(T entity) {
		entity.setId(null);
		entityManager.persist(entity);
	}

	public T insertReturn(T entity){
		entity.setId(null);
		return (T) entityManager.merge(entity);
	}

	public T update(T entity) {
		return entityManager.merge(entity);
	}

	public void delete(T entity) {
		entityManager.remove(find((I) entity.getId()));
	}

	public T find(I id) {
		return entityManager.find(clazz, id);
	}

	public List<T> findAll() {
		Query query = entityManager.createQuery(new StringBuilder("from
").append(clazz.getName()).toString());
		List<T> resultList = query.getResultList();
		return resultList;
	}

	public List<T> pagination(int first, int size, Map<String,String> filters) {

		StringBuilder queryString = new StringBuilder("select o from
").append(clazz.getSimpleName()).append(" o ");

		if (getDefaultOrderField() != null && getDefaultOrderField().length() > 0) {
			queryString.append(" order by ").append(getDefaultOrderField());
		}
```

```
                Query query = entityManager.createQuery(queryString.toString());

                query.setFirstResult(first);
                query.setMaxResults(size);

                return query.getResultList();
        }

        protected String getDefaultOrderField() {
                return null;
        }

        private Class<T> retornaTipo() {
                Class<?> clazz = this.getClass();

                while (!clazz.getSuperclass().equals(RepositoryImpl.class)) {
                        clazz = clazz.getSuperclass();
                }

                ParameterizedType tipoGenerico = (ParameterizedType) clazz.getGenericSuperclass();
                return (Class<T>) tipoGenerico.getActualTypeArguments()[0];
        }
}
```

Listagem 9 – Repository que implementa as operações básicas de banco de dados

5 CONCLUSÃO

O objetivo principal deste trabalho, representado pelo desenvolvimento de um aplicativo *web* para a composição e a disponibilização de questionários de avaliação de atividades, foi alcançado. Um aplicativo *web*, utilizando conceitos de interface rica, foi desenvolvido por meio do uso da linguagem Java e tecnologias associadas. O aplicativo facilitará a composição, aplicação e análise dos dados de questionários que tem a finalidade de acompanhar a realização de atividades ou avaliar a realização das mesmas.

O aplicativo foi pensado para o ambiente acadêmico visando suprir uma necessidade do Departamento Acadêmico de Informática da UTFPR, Câmpus Pato Branco, mas pode ser aplicado para outras instituições que possuam interesse em avaliar a realização das suas atividades.

No âmbito do desenvolvimento de aplicações *web* existe um grande número de *frameworks* e ferramentas. No caso do Vraptor, que foi utilizado no projeto de desenvolvimento do sistema objeto deste trabalho, o grande foco está na alta produtividade, na boa curva de aprendizagem e na ampla comunidade de desenvolvedores, que possibilita encontrar facilmente o que é necessário para a resolução de um problema. É uma ferramenta muito robusta e extensível, que, para o caso de surgir uma necessidade fora do escopo do próprio Vraptor, possui muitos *plugins* de fácil instalação para muitos dos problemas enfrentados pelos desenvolvedores *web*.

Porém, um dos aspectos nos quais a ferramenta apresenta restrições é em relação às *views* ou páginas do sistema. Diferentes soluções, como JSF, fornecem uma cobertura mais abrangente incluindo do servidor até as telas do sistema. No Vraptor cabe ao desenvolvedor escolher se o sistema usará somente páginas JSP ou outra abordagem. Para o caso do projeto desenvolvido, foi usada a junção entre o *framework* JavaScript AngularJs e o *framework* de desenvolvimento de aplicações *web* responsivas Bootstrap.

No AngularJs, a facilidade de desenvolvimento vem da conveniência que a ferramenta fornece em montar um objeto Json toda vez que algo é digitado em um campo mapeado pelo ng-model. E isso combinado com o recurso chamado *two-way-data* faz com que seja muito rápido escrever métodos que usem recursos do servidor. Outro aspecto relevante é o recurso chamado *ng-resource*, que auxilia

muito aplicações RESTful fornecendo atalhos paras os métodos implementados na parte de servidor da aplicação.

Em relação do Bootstrap, ele fornece uma boa estrutura de classes CSS que com um pouco de aprendizagem permite desenvolver interfaces agradáveis, em termos de recursos de interação, ou seja, que atendem padrões de usabilidade, sem muito esforço por parte do desenvolvedor.

Com base nisso, a junção entre AngularJs e Bootstrap é uma boa resposta para a questão de o quê utilizar como *front-end* em uma aplicação desenvolvida utilizando um *framework* como o Vraptor.

Como trabalhos futuros, complementares ao que foi desenvolvido, está a implementação de funcionalidades relacionadas ao auxílio na análise dos dados. Gerando, assim, agrupamentos de acordo com metodologias de Estatística.

REFERÊNCIAS

ADOBE FLEX. Disponível em: <http://www.adobe.com/products/flex/>. Acesso em: 20 abr. 2015.

AGHAEI, Sareh; NEMATBAKHSH, Mohammad Ali; FARSANI, Hadi Khosravi. **Evolution of the world wide web: from Web 1.0 to Web 4.0**. International Journal of Web & Semantic Technology (IJWesT), v.3, n.1, January 2012. Disponível em: <http://airccse.org/journal/ijwest/papers/3112ijwest01.pdf>. Acesso em: 10 jan. 2015.

ALMEIDA, Fernando L. F.; LOURENÇO, Justino M. R. **eCreation of value with web 3.0 technologies**. In: 6th Iberian Conference on Information Systems and Technologies (CISTI), 2011, p. 1-4.

AMALFITANO Domenico; FASOLINO, Anna Rita; POLCARO, Armando; TRAMONTANA, Porfirio. **Comprehending Ajax web applications by the DynaRIA tool.** 2010 Seventh International Conference on the Quality of Information and Communications Technology, p. 122-131.

FLAT EDUCATION. **Web 1.0 vs Web 2.0 vs Web 3.0 vs Web 4.0 vs Web 5.0 – A bird's eye on the evolution and definition**. Disponível em: <https://flatworldbusiness.wordpress.com/flat-education/previously/web-1-0-vs-web-2-0-vs-web-3-0-a-bird-eye-on-the-definition/>. Acesso em: 10 jan. 2015.

FRATERNALI, Piero. Tools and approaches for developing dataintensive web applications: a survey. **ACM Comput. Surv**. v. 31, n. 3, p. 227–263, 1999.

GARRETT, James. **AJAX: A new approach to web applications**, Adaptive Path, 2005.

HENDLER, James. A. Web 3.0 emerging, **Computer**, v. 42, n. 1, 2009, p. 111-113.

HOGG Roman; MECKEL, Miriam; STANOEVSKA-SLABEVA, Katarina; MARTIGNONI, Robert. **Overview of business models for web 2.0 communities**, Proceedings of GeNeMe, 2006, p. 23-37.

JAZAYERI, Mehdi. **Some trends in web application development**. Future of software engineering (FOSE'07), 2007, p. 1-15.

MICROSOFT SILVERLIGHT. Disponível em: <http://silverlight.net/>. Acesso em: 20 abr. 2015.

O'REILLEY, Tim. What is Web 2.0. **Design patterns and business models for the next generation of software**, 2005. Disponível em:

<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/whatis-web-20.html>. Acesso em: 10 jan. 2015.

PAVLIĆ, Daniel; PAVLIĆ, Mile; JOVANOVIĆ, Vladan. **Future of internet technologies**. 35th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2012, p. 1366- 1371.

PRESSMAN, Roger. **Engenharia de software**. Rio de Janeiro: McGraw-Hill, 2006.

TRAMONTANA, Porfirio; AMALFITANO, Domenico; FASOLINO, Anna Rita. **Reverse engineering techniques: from web applications to rich internet applications**, 2013 IEEE, p.83-86.

VICENTIM, Joice. **Web 10, Web 2.0 e Web 3.0 … enfim, o que é isso.** 2013. Disponível em: <http://www.ex2.com.br/blog/web-1-0-web-2-0-e-web-3-0-enfim-o-que-e-isso/>. Acesso em: 10 jan. 2015.