UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

RAFAEL SORATTO ORTOLAN

APLICATIVO PARA ELABORAÇÃO E COMPARTILHAMENTO DE QUESTÕES AVALIATIVAS

CRICIÚMA

2014

RAFAEL SORATTO ORTOLAN

APLICATIVO PARA ELABORAÇÃO E COMPARTILHAMENTO DE QUESTÕES AVALIATIVAS

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. Esp. Fabricio Giordani

CRICIÚMA

2014

RAFAEL SORATTO ORTOLAN

APLICATIVO PARA ELABORAÇÃO E COMPARTILHAMENTO DE QUESTÕES AVALIATIVAS

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Desenvolvimento Web.

Criciúma, 25 de junho de 2014.

BANCA EXAMINADORA

Prof. Esp. Fabrício Giordani - UNESC - Orientador

Prof. MSc. Christine Vieira - UNESC

Prof. MSc. Leila Laís Gonçalves - UNESC

Aos meus amigos e familiares.

AGRADECIMENTOS

Agradeço a minha família, aos meus amigos e todos os demais que me ajudaram a desenvolver e conseguir conquistar este objetivo.

“O conhecimento nos faz responsáveis.”

Che Guevara

RESUMO

Os professores necessitam de uma forma de avaliar o conteúdo que o aluno aprendeu sobre um determinado assunto. O trabalho consiste em estudar a prática avaliativa utilizada por professores de instituições de ensino, levando em consideração a importância dela na aprendizagem e de como ela deve ser aplicada. Professor é uma profissão importante para o desenvolvimento de um país e eles estão cada vez mais sendo cobrados e requisitos, devido à escassez de obra qualificada, dessa forma qualquer ideia para melhorar a rotina desta profissão surtirá um grande efeito. Baseado nisso o trabalho demonstra o desenvolvimento de um protótipo de um aplicativo *web*, o qual permite ajudar os professores elaborarem suas avaliações de forma cooperativa e com mais facilidade, enfatizando como um sistema pode ajudá-los nesta área. O desenvolvimento do protótipo foi realizado com a arquitetura *Representational State Transfer* (REST) para o lado *server* e com o *framework* AngularJS para o desenvolvimento do lado *client*, tendo como as duas principais linguagens de programação o JAVA e o JavaScript. O sistema desenvolvido possibilitou conhecer novas tecnologias e como estas trabalham em conjunto na arquitetura REST. Foi realizado um questionário para verificar como os professores costumam aplicar suas avaliações e quais as características que estas têm. Este estudo serviu para entender como os professores avaliam seus alunos e de que forma é feita a avaliação, assim compreendeu-se os principais problemas existentes tanto na elaboração quanto em sua aplicação. Por fim compreende-se que a prática avaliativa é muito importante para que um professor consiga mensurar o conteúdo absorvido pelo aluno, no entanto sua elaboração é maçante e sem otimização na maioria das vezes, onde o professor utiliza técnicas não evoluídas. A criação de um sistema pode mudar este paradigma, mostrando que a tecnologia pode evoluir essa área da educação e em outras existentes.

Palavras-chave: Professores. Avaliação. JAVA. REST. AngularJS.

ABSTRACT

Teachers need a way to assess the content that students learned about a particular subject. The work consists of studying the evaluation practices used by teachers of educational institutions, taking into account its importance in learning and how it should be applied. Teaching is important profession for the development of a country and they are increasingly being charged and required, due to shortage of skilled labor, thus any idea to improve this profession routine causes a big effect. Based on this, the work demonstrates the development of a web application prototype, which allows you to help teachers develop their assessments cooperatively and more easily, highlighting how a system can help them in this area. The development of the prototype was done with the Representational State Transfer (REST) architecture for the server side and the AngularJS framework for developing the client side, and as the two main programming languages JAVA and JavaScript. The system developed has helped understand new technologies and how they work together in the REST architecture. A survey was conducted to investigate how teachers often apply their assessments and what characteristics they have. This study served to understand how teachers assess their students and how the assessment is done, in this way is understood major problems both in the preparation and in their application. Conclude it is acknowledged that the evaluation practice is very important for a teacher to be able to measure the content absorbed by the student, however its development is dull and not optimized in most cases where the teacher uses not evolved techniques. The creation of a system can change this paradigm, showing that technology can evolve this area of education and other related ones.

Keywords: Teachers. Review. JAVA. REST. AngularJS.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AJAX	*Asynchronous Javascript and XML*
API	*Application Programming Interface*
CDI	*Contexts and Dependency Injection*
CSS	*Cascading Style Sheets*
DAO	*Data Acess Object*
DOM	*Document Object Model*
EE	*Enterprise Edition*
EJB	*Enterprise Java Bean*
EL	*Expression Linguage*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
IE	*Internet Explorer*
JAR	*Java ARchives*
JAX-WS	*Java API for eXtensible Markup Language Web Services*
JAX-RS	*Java API for Representational State Transfer Web Services*
JDK	Java *Development Kit*
JPA	*Java Persistence API*
JS	*JavaScript*
JSF	*Java Server Faces*
JSON	*JavaScript Object Notation*
JVM	Java *virtual machine*
MVC	*Model View Controller*
MVP	*Model View Presenter*
PHP	*Hypertext PreProcessor*
REST	*Representational State Transfer*
SO	*Sistema Operacional*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
VO	*Value Object*

XML	*eXtensible Markup Language*
WSDL	*Web Services Description Language*
WWW	*World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

As instituições de ensino contêm professores que precisam aplicar avaliações, estas podem ser provas, testes, exames ou qualquer outro método para testar o conhecimento do aluno sobre um determinado conteúdo. Estas provas ou avaliações podem ser aplicadas de forma escrita, o qual é entregue ao examinando uma folha com perguntas, e o mesmo devolverá ela preenchida (MORETTO, 2005).

Para que um professor consiga elaborar uma prova para seus alunos, ele terá que seguir várias etapas além de ter um bom domínio do conteúdo que deseja avaliar deles. Para conseguir questões, os professores pegam-nas prontas da internet, outras já colocadas em avaliações anteriores ou criam questões a partir de sua criatividade e conhecimento. Não bastando a isso, tem que montar a ordem das questões e geralmente fazer mais de um tipo de avaliação para dificultar fraudes durante a sua aplicação (MORETTO, 2005; PERRENOUD, 1999).

Depois de passarem este esforço elaborando uma avaliação que aparenta ser consistente e íntegra, as questões perdem sua validade, desde o momento que uma questão é usada tem então sua resposta publicada. Caso em uma próxima oportunidade for aplicada uma avaliação do mesmo conteúdo para uma outra turma, pode ser usada a mesma, contudo o efeito provavelmente não será o igual, pois assim que uma questão é usada, ela pode se torna pública, podendo ser consultada a qualquer momento. Para tentar garantir uma avaliação mais rígida, são criadas novas outras questões refazendo o mesmo processo de elaboração citado no inicio.

Para não passarem um árduo trabalho, pode-se reaproveitar questões de outros professores, estas são geralmente usadas de duas formas, uma delas e reescrever a questão apenas reaproveitando o mesmo sentido, outra é pegar um modelo de prova e cortar as questões para poder montar uma nova avaliação. Algumas destas técnicas facilitam a elaboração das provas, porém ainda são todas manuais. Este processo pode dificultar e atrasar as atividades de um professor.

Em relação às instituições de ensino, não há uma padronização nas provas, na grande maioria não se tem noção da consistência da avaliação aplicada pelos professores.

O trabalho consiste em desenvolver um protótipo de aplicação que permita aos professores montar suas avaliações de uma única forma, além de

possibilitar o compartilhamento de questões entre os mesmos, com o principal objetivo de ajudar a reaproveitar avaliações e questões já aplicadas.

1.1 OBJETIVO GERAL

Desenvolver um protótipo de uma aplicação para a elaboração e aplicação de avaliações compartilhando questões entre professores de diferentes instituições de ensino.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos são:

a) estudar a estruturação de questões e avaliações;
b) estudar e utilizar o *framework* AngularJS para o desenvolvimento *front-end*;
c) estudar e utilizar conceitos de RESTFul para o desenvolvimento *server-side*;
d) desenvolver um protótipo de aplicação utilizando as tecnologias estudadas que proporcione a elaboração colaborativa de avaliações para instituições de ensino.

1.3 JUSTIFICATIVA

O professor é extremamente importante para qualquer nação, são estes os responsáveis por disseminar o conhecimento adiante. Estes profissionais passam muito tempo em contato com as pessoas e é neles que muitos se espelham para seguir em frente em uma determinada profissão. Ao longo do tempo percebeu-se que os profissionais de ensino foram sendo mais cobrados em relação à eficácia do seu trabalho e formação acadêmica. Por ser uma profissão que exige um alto nível de conhecimento específico, há certa dificuldade de encontrar profissionais bem qualificados psicologicamente e didaticamente para executar esta função (CAIADO, 2012).

Poucos recursos foram desenvolvidos para facilitar as funções de um professor, eles ainda se deparam com praticamente só *“papel e caneta”*, não sendo

disponíveis muitos recursos tecnológicos. Um professor tem contato com pessoas, seja diretamente ou indiretamente, necessitando de formação pedagógica e paciência para aguentar todas as situações, mas nem todos conseguem, causando então uma grande perturbação e descontentamento com a profissão (FREITAS, 2011).

De acordo com Freitas (2011) são estas situações que diminuem o rendimento de um professor, desempolgam e afastam pessoas qualificadas deste posto de trabalho.

Qualquer contribuição que se possa fazer para desenvolver recursos para esta profissão irá se tornar muito valiosa, visto que com o aumento da população, a procura e vagas de professores crescem relativamente. As pessoas que assumem o perfil de um professor, no mínimo devem ter em mente que estão fazendo isso para passar o conhecimento para seus semelhantes. Este é um belo pensamento, porém para ser mantido precisa de retorno psicológico e colaboração social (MANDELLI, 2012).

1.4 ESTRUTURA DO TRABALHO

O capitulo dois demonstra a metodologia de ensino pesquisada, enfatizando sobre a utilização das avaliações no processo de aprendizagem. O capitulo três contêm a estrutura das avaliações, sua possibilidade de aplicações, abordando também os tipos de questões que podem ser utilizadas e qual parte do conhecimento cada uma explora. O capítulo quatro aborda as tecnologias e utilizadas para o desenvolvimento do trabalho. No capítulo cinco encontram-se alguns trabalhos correlatos pesquisados. Os próximos dois capítulos tratam da parte do desenvolvimento do protótipo, o capítulo seis demonstra como foi a análise e o desenvolvimento do protótipo e quais etapas foram necessárias para atingir os objetivos. O capítulo sete, o último, contém a conclusão e trabalhos futuros.

2 ABORDAGEM DA AVALIAÇÃO NO PROCESSO DE APRENDIZAGEM

A pedagogia relacionada a prática de avaliação não teve muitas mudanças em relação aos seus conceitos, desde de os anos sessenta praticamente as principais ideias são mantidas, definindo a avaliação como uma forma de alinhamento ou prova da qualidade a cerca de um objetivo. A importância da avaliação do ensino ganhou um espaço exclusivo no ambiente de aprendizagem, a ponto do ensino perder o foco na aprendizagem, tornando-se então praticamente voltada ao exame, a principal preocupação do professor depois de lecionar um conteúdo é saber se o aluno está apto a fazer uma prova e não se ele realmente aprendeu (LUCKESI, 2001; MORETTO, 2005; PERRENOUD, 1999).

Na sociedade a parte mais visível desta aplicação é no ensino médio, neste caso o objetivo é ensinar como resolver provas, e não o conteúdo em si, isso acontece por uma força maior da sociedade, a avaliação aplica-se como forma de nivelamento em todos os tipos de conceitos, deste a prática com crianças em casa, até a obtenção de um emprego em concursos públicos, frisando então a importância de terem avaliações bem feitas nos ambientes de ensino (LUCKESI, 2001).

Em sala de aula os professores realizam testes para analisar se o aluno está apto a seguir para o próximo conteúdo, esta verificação se faz na aferição do resultado das provas aplicadas, basicamente são feito três passos para medição deste resultado, o primeiro é a aplicação da prova, o segundo é a transformação do resultado da avaliação em um modo de medir, tendo como exemplo a transformação em um pontuação, nota ou conceito, e por terceiro, é fazer algo a respeito do resultado obtido (LUCKESI, 2001; MORETTO, 2005; PERRENOUD, 1999).

A essência da avaliação na educação, partindo de um propósito em que se tem um esforço demasiado para fornecer um conteúdo a um aluno, ela entra para mensurar qual foi a quantidade de conteúdo absorvido, também para poder corrigir possíveis desvios realizados durante o trabalho de ensino, ajudando a estabelecer critérios de desempenho e realizar observações nas medições dos resultados avaliados. Para obter diagnósticos em relação a aprendizagem do ensinando, é necessário fazer processos de avaliação, para servirem como *feedbacks*, mas não só para avaliar o aluno, como ainda para mensurar todo um contexto escolar, como esta sendo desenvolvida e realizadas as técnicas de ensino passadas pelo próprio

avaliador, validando periodicamente a pedagogia aplicada, servindo como taxa de amostragem da evolução do trabalho proposto (RABELO, 1998; VIANNA, 1976).

O significado de uma avaliação para um aluno, baseia-se na oportunidade de conhecer os resultados do esforço empenhado nos estudos, não só isso, mas também ajuda em adquirir satisfação ao refletir sua capacidade de aprendizado. As atividades avaliativas contribuem para o desenvolvimento social, intelectual e moral do avaliado. Essas características vão ser influenciadas dependendo do seu sucesso ou fracasso (MELCHIOR, 2002).

Ainda de acordo com Melchior (2002), o professor tem como proveito na aplicação das avaliações, uma análise reflexiva, conseguindo mensurar a eficiência de seu desempenho como lecionando. É possível ainda construir métodos e novas formas de avaliar que se adaptem ao ritmo de desenvolvimento de cada aluno.

As avaliações podem ser aplicadas de diversas maneiras, porém qualquer avaliação educacional se enquadra em uma série de classificações em relação ao modo de como ela é feita. Conhecendo algumas classificações e categorias pode-se definir qual é objetivo prévio da avaliação, essas categorias irão definir quais serão as técnicas usadas para realizar a prova.

2.1 FUNDAMENTAÇÕES DA ELABORAÇÃO DE AVALIAÇÕES

Um professor introduzido no ambiente pedagógico, tem como objetivo principal ensinar. Esse objetivo desencadeia uma série de metodologias para atingir esta meta, além de competência e grande conhecimento sobre o conteúdo que ele vai ensinar, é necessário ter técnicas e seguir procedimentos de como transmitir esse conhecimento. Para conseguir elaborar todo um contexto levando em consideração do inicio, o aluno geralmente não faz ideia do que vai aprender nem para que serve, até o final onde ele deve ter conseguido passar ao aprendiz uma síntese do seu ensinamento (LUCKESI, 2001).

A tarefa de construir uma avaliação, para um professor, envolve grande complexidade, muitos estudos já foram realizados para tentar encontrar uma metodologia padrão de elaboração de provas, contudo existe um conceito onde grande parte das metodologias de ensino podem se espelhar, conhecida como "Taxonomia de Bloom" ou "Taxonomia de Objetivos Educacionais". O foco abordado por este trabalho será apenas os objetivos em torno das avaliações, sem considerar

outros detalhes como feições teóricas. A taxonomia não dará um perfeito retorno de como deve ser ensinado ou explorado a avaliação, mas sim um grande auxílio, mostrando as diferentes operações mentais dos alunos envolvendo o processo cognitivo, que devem ser levadas em consideração no momento da aplicação da avaliação, provendo então um conhecimento necessário perante aos educandos, abrindo a possibilidade de replanejamentos pedagógicos (MORETTO, 2005).

A avaliação faz parte do processo de aprendizagem, não só para cobrar, mas sim para acrescentar e contribuir. O estudo de uma elaboração de testes é complexo e leva em consideração muitos fatores. Vários estudiosos já tentaram levar a forma de avaliar em sua excelência. As representações da taxonomia de Bloom ajudam um professor a decidir quais são as técnicas de elaboração de avaliações que ele pode usar para atingir seus objetivos (VIANNA, 1976).

2.1.1 Taxonomia de Bloom

A taxonomia é uma subdisciplina de biologia, pratica por *taxonomistas*, e tem como objetivo dividir, classificar e nomear grupos e seus elementos. Cada grupo recebe uma pontuação, um grupo pode ser agregado por outro, tendo então uma pontuação maior. Não existe regra definida de como estes grupos vão ser classificados, nomeados ou publicados (MORETTO, 2005).

Pesquisadores já usaram a taxonomia para estruturar, nomear e dividir várias teorias conceituais. A aplicação da taxonomia no domínio da aprendizagem, foi dividir em grupos relacionados a gama abrangida por cada domínio específico relacionado a capacidade de aprendizagem o aluno. Criando então as divisões de grupo de desenvolvimento cognitivo, afetivo e psicomotor (BLOOM, 1956; BLOOM, 1974).

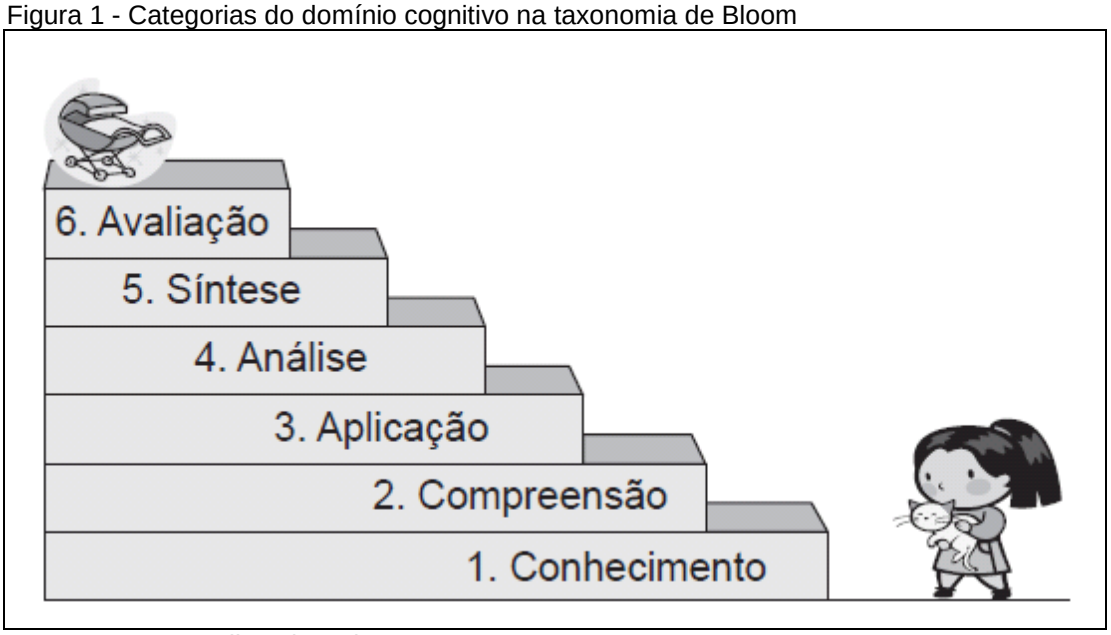
O cognitivo está relacionado a capacidade de aprender, do amadurecimento da intelectualidade do aluno, proporcionando um desenvolvimento continuo de aprendizagem, desde a etapa inicial onde se procura adquirir o conhecimento até o momento de sua aplicação na prática.

Ainda segundo Bloom et al (1956), o cognitivo ficou dividido em seis categorias, estabelecidas por uma ordem hierárquica, para conseguir passar para o próximo nível necessita-se ter um considerável desempenho na camada anterior, pois o conhecimento adquirido anteriormente será usado na medida que o processo

evolui. As categorias ficaram dividas na seguinte forma em ordem hierárquica: conhecimento, compreensão, aplicação, análise, síntese e avaliação.

Todos os três domínios definidos na taxonomia, foram muito discutidos, porém o psicomotor e o afetivo são mais difíceis de ser aplicados e mensurados. O que tem mais aceitação e aplicação é o cognitivo, este domínio é amplamente usado por educadores como base de estruturação de suas metodologias de ensino e de elaboração de avaliações, desta forma este trabalho se enfatizará apenas no domínio cognitivo (FERRAZ; BELHOT, 2010; VIANNA, 1976). A figura 1 ilustra os passos no domínio cognitivo para o desenvolvimento na taxonomia de Bloom.

Figura 1 - Categorias do domínio cognitivo na taxonomia de Bloom



Fonte: Ferraz e Belhot (2010).

2.1.2 Categorias da taxonomia de Bloom

Conforme Bloom (1956), Ferraz e Belhot (2010) e Gronlund (1968), as categorias do domínio cognitivo podem ser descritas a seguir:

a) conhecimento: o conhecimento está ligado a lembrança, a possibilidade do aprendiz poder resgatar na sua memória algo que ele aprendeu, como fatos, regras, classificações, teorias, procedimentos, palavras. Exemplos de inícios de questões que reproduzem a prática do conhecimento: A principal diferença entre; O melhor exemplo para o princípio; As principais características; O efeito do fenômeno;

b) compreensão: engloba a capacidade do aluno explicar algo que entendeu, conseguindo reproduzir uma opinião com sua própria palavra, versão e interpretação sem perder o sentido do conteúdo principal abordado. Enunciados de questões que abrangem o conhecimento podem ser exemplificados por: O que significa a formula abaixo; Que teoria está implícita na afirmação abaixo;
c) aplicação: é a possibilidade do aluno conseguir aplicar o que aprendeu em novas situações, a grande diferença entre aplicação e compreensão, é de que o aluno irá fazer uma estruturação diferente do conteúdo, de forma que sinta-se familiarizado com a nova hipótese. Uma introdução a uma questão que induz uma resposta usando a aplicação seria: Qual o procedimento experimental mais aconselhável para a pesquisa da seguinte situação... ?;
d) análise: consiste em separar um determinado assunto em partes, conseguindo estabelecer uma relação entre elas, podendo definir qual a influencia e impacto de cada parte sobre as outras, nesta fase não basta o aluno ter compreendido o conteúdo, e assim a estruturação e hierarquia do contexto em estudo. A introdução de uma questão que induz a análise pode ser exemplificada por: No texto abaixo qual é o parágrafo que demonstra a conclusão?;
e) síntese: nesta fase o aluno a cria uma estrutura nova e diferente, baseado em partes ou fragmentos de aprendizados anteriores, nessa fase é onde está mais relacionado a criatividade do aprendiz. Podendo citar como exemplo para uma questão envolvendo síntese a seguinte introdução: Qual formula matemática que melhor contextualiza as seguintes informações;
f) avaliação: permite o aluno poder avaliar algo que está em questão, necessitando compreender e analisar, para depois fazer uma avaliação consciente com base em certos critérios, como o valor de algo para um determinado propósito. Uma introdução de questão que se deseja extrair a capacidade avaliativa do aluno pode ser: Qual das afirmações abaixo está de acordo com os seguintes critérios.

As avaliações podem ser aplicadas de diversas formas, porém qualquer uma se enquadra em uma série de categorias em relação ao modo de como foi feita,

conhecendo algumas classificações e categorias pode-se definir qual é objetivo prévio de sua aplicação, essas categorias irão definir quais serão as técnicas usadas para realizar a prova.

2.1.3 Classificação da avaliação em relação a sua aplicação

As avaliações, de acordo com a maioria dos autores, estão dentro de uma quantidade de parâmetros que classificam elas em tipos distintos, dividindo-as em categorias de acordo com suas características. Algumas categorias geralmente usadas são, quando a regularidade, ao avaliador, explicidade, à comparação e a formação (RABELO, 1998). Na tabela 1 pode-se observar a divisão em relação à aplicação.

Tabela 1 - Classificação de avaliações em relação à aplicação

QUANTO A	**TIPOS**	
regularidade	contínua	pontual
avaliador	Interna	externa
explicidade	implícita	explícita
comparação	normativa	criteral
conhecimentos	diagnóstica	somativa
	formativa	

Fonte: Rabelo (1998).

De acordo com Rabelo (1998), a avaliação quando a formação se divide em três classificações, diagnóstica, somativa e formativa, baseado nas características e objetivos de cada avaliação formulada.

Uma avaliação diagnóstica tem interpretações um pouco diferentes dependendo dos autores, mas o conceito principal é que ela é feita com intuito te retirar um prognóstico inicial de um aluno, geralmente é feita no inicio do processo de aprendizagem, por isso pode receber o nome de avaliação inicial (LOURENÇO, 1970; RABELO, 1998).

Uma avaliação somativa é usada no final do processo de educação, ou final de semestre, bimestre, curso, entre outros. Geralmente é aplicada de forma pontual, seu conteúdo abrange um domínio dos assuntos passados durante a fase de aprendizagem. O objetivo desta avaliação é extrair o resultado final do período de ensinamento. Com o resultado dessa avaliação conseguem-se informações

processadas do que a aluno assimilou, ou seja, é como uma prova real, usada para graduar, classificar, verificar e certificar (MORALES, 2003; RABELO, 1998).

A última classificação em relação ao conhecimento é a avaliação formativa. Esta tem a finalidade de gerar informações em torno de um processo, não tendo como característica provar nada, mas sim, auxiliar e confortar, fazendo parte do processo de aprendizagem. O professor pode acompanhar o desenvolvimento da aprendizagem, ajudando então aos seus alunos com as principais barreiras encontradas. Ela contribui para o desenvolvimento e melhoria da aprendizagem, fornecendo um *feedback* com informações úteis, possibilitando um acompanhamento rápido para verificar se realmente o aluno está conseguindo ter uma evolução, se está com dificuldades, ou até já concluiu está etapa (PERRENOUD, 1999; RABELO, 1998).

A frequência em que a avaliação é aplicada é chamada de classificação pela regularidade, podendo então ser pontual ou contínua. A contínua tem como característica o fato de ser regular aplicada durante a fase de desenvolvimento do aluno, não esperando chegar uma determinada hora ou final para sua utilização. Já a avaliação pontual é utilizada apenas no final da etapa de aprendizagem, esperando todo o conteúdo ser aplicado ao aluno, como um provão final ou exame de recuperação (RABELO, 1998).

Em relação ao avaliador, tem como classificações de avaliação dois tipos, interna e externa. A interna é quando é o mesmo professor que ensina e aplica a prova, esta pode ser pontual ou contínua. As populares avaliações regulares aplicadas durante o processo de aprendizagem, são exemplos típicos de interna e contínua, como exemplo de interna e pontuais, são as aplicadas em determinada fase de tempo, como as provas bimestrais, semestrais e finais. Já a avaliação externa é quando a pessoa que aplica não é a mesma que participou da fase de aprendizagem. A avaliação externa também pode ser pontual ou continua. Quando um coordenador ou orientador frequenta um determinado grupo de alunos para ver como está indo o resultado do aprendizado, este está aplicando um exemplo de avaliação externa e contínua. Já para as avaliações externa e contínua são arduamente encontradas na sociedade, como vestibulares, provas de concursos públicos e exames aplicados pelo governo para avaliar o ensino público (RABELO, 1998).

A questão de explicidade também é considerada um critério de classificação para as avaliações. Quando os avaliados não se dão conta que estão fazendo parte de um processo avaliativo, tem-se então uma avaliação implícita, e pelo contrário, existe a explícita, que é quando todos os avaliados têm noção bem clara que estão fazendo um exame para medir ou testar algum critério sobre eles (RABELO, 1998).

A avaliação quanto à comparação, tem duas classificações, normativa e criterial. A normativa é quando se deseja avaliar um aluno perante a um grupo em que ele pertence, para medir até que ponto ele está apto em relação aos demais, quais são os seus prós e contras, este tipo de avaliação é aplicada praticamente em todos os esportes, tendo como exemplo as competições entre atletas. O objetivo da avaliação criterial é extrair um posicionamento do avaliado em relação a um objetivo pré-estabelecido, fazendo o levantamento do que ele sabe ou pode fazer, assim como o que não pode (RABELO, 1998). A classificação quanto à formação pode ser observada na tabela 2.

Tabela 2 - Classificação de avaliações quanto à formação

PERÍODOS	TIPOS	OBJETIVOS	INTERESSES	BUSCAS
início	diagnóstica	Orientar explorar identificar adaptar predizer	aluno enquanto produtor	a avaliação busca conhecer, principalmente as aptidões, os interesses e as capacidades e competências enquanto pré-requisitos para futuros trabalhos
durante	formativa	Regular situar compreender harmonizar tranquilizar apoiar reforçar corrigir facilitar dialogar	aluno enquanto atividades, processos de produção	a avaliação busca informações sobre estratégias de solução dos problemas e das dificuldades surgidas
depois	somativa	Verificar classificar situar informar certificar pôr à prova	aluno enquanto produto final	a avaliação busca observar comportamentos globais, socialmente significativos, determinar conhecimentos adquiridos e, se possível, dar um certificado

Fonte: Rabelo (1998).

2.2 FORMAS DE APLICAÇÃO

2.2.1 Testes orais

Os testes orais são aqueles que o professor pergunta ao aluno, e o mesmo responde oralmente, a grande importância desse é saber o que realmente o aluno pensa, pois ele irá responder exatamente do jeito que a resposta está montada na cabeça dele, exibindo bem a sua opinião e entendimento do assunto abordado. Um ponto negativo é que a demanda de tempo em sua aplicação é grande, levando em consideração que geralmente são muitos alunos para apenas um professor, em determinadas condições ele não consegue ser aplicado com eficiência. Desta forma os testes orais foram perdendo espaço, até chegar o momento de serem raramente ou nem aplicados mais no ensino, perdendo espaço para os testes objetivos e práticos (MEDEIROS, 1983; MELCHIOR, 2002).

2.2.2 Testes práticos

Testes práticos são os que colocam o aprendiz em situação de tarefa real, simulando realmente alguma atividade que ele poderá exercer na sua vida, como andar de bicicleta, nadar, instalar um cabo de rede, entre outros. Geralmente esses testes são utilizados em aulas práticas ou laboratoriais, com equipes e grupos ajudando a interação entre pessoas do cotidiano (MELCHIOR, 2002).

Os testes práticos para atingirem eficácia devem fornecer ao aluno oportunidade de mostrar sua capacidade, testar suas técnicas de atividades práticas, estas serão avaliadas na integra pelo professor em forma de relatórios (BORDENAVE, 2006; MELCHIOR, 2002).

A avaliação dos testes práticos tem como base algumas características, com a quantidade do trabalho feito, analisar o esforço que o aluno demandou executando uma determinada tarefa, a qualidade do trabalho realizado, a cooperação entre os integrantes do grupo caso exista, a assiduidade, se o aluno é pontual, se realmente interage e acompanha o objetivo da aula e a atitude, levando em consideração a seriedade e importância que o aluno esta dando a atividade (BORDENAVE, 2006; MELCHIOR, 2002).

2.2.3 Testes Objetivos

Os testes objetivos são compostos por questões que possuem alternativas de respostas. Estes testes servem para analisar a extensão do conhecimento do aluno. Existem dois grupos de questões que entram nesse tipo de teste, as que exigem relacionamento da resposta, neste caso incluem-se questões de lacuna ou complemento, que exige do aluno a associação. O outro grupo incluem questões que exigem o reconhecimento do aluno, nestas as respostas estão dispostas para o examinando, basta ele identificar qual é a correta, como questões de verdadeiro ou falso, questões de combinação, ordenação e múltipla escolha. A escolha de quais questões usar vai depender de inúmeros fatores, e quem determinará será o professor, baseando-se nos objetivos em que deseja atingir (MEDEIROS, 1983; MELCHIOR, 2002).

2.2.4 Testes dissertativos

Este teste é constituído por questões em que o avaliado elabora sua própria resposta, tendo que esquematizar, descrever, explicar, comparar e apresentar argumentos, as questão postas nessa prova devem ser elaboradas com enunciados claros, enfatizando realmente o que o aluno tem que dizer na resposta. As questões deste teste podem ser classificadas de acordo com o tamanho de sua resposta, longas, médias ou curtas, esse fator é importante, pois muda o tempo de resolução da questão. Um teste só é considerado dissertativo quando todas as questões incluídas são dissertativas (MELCHIOR, 2002).

Ainda de acordo com Melchior (2002), as vantagens deste tipo de teste são atraentes, como, a facilidade de elaboração da questão devido que não há necessidade de disponibilizar as respostas na prova, a grande redução do acerto casual, a possibilidade de extrair respostas elaboradas aumentando a eficiência do teste. Como desvantagens, exige maior tempo para correção, possibilita margens de interpretações diferentes e considerações do professor na hora da correção.

2.3 TIPOS DE QUESTÕES PARA CADA AVALIAÇÃO

2.3.1 Tipos de questões objetivas

As questões objetivas levam em consideração como principal característica, fornecer opções de resposta ao avaliado. Entre os mais comuns, estão relacionados as de lacuna ou complemento, verdadeiro ou falso, associação, ordenação e múltipla escolha. Estes tipos de questões são geralmente usados desde a escola primária (MEDEIROS, 1983).

2.3.2 Questão de múltipla escolha

Consideram-se com o maior número de benefícios, pois o avaliado terá que optar por uma ou mais respostas, isso eleva o grau de dificuldade do acerto integro da questão (MEDEIROS, 1983).

Segundo Medeiros (1983) e Melchior (2002) as principais vantagens desse tipo de questão são:

a) facilidade e rapidez em responder e corrigir esta questão;
b) estimula a atitude crítica, devido a quantidade de possíveis respostas;
c) reduz o acerto por sorte, dependendo da quantidade de alternativas, sendo como padrão e medida cinco alternativas, elevando a proporção de possibilidade em uma prova de vinte questões por exemplo;
d) sua correção pode ser feita por uma terceira pessoa, desde que ela tenha as respostas em mãos.

Ainda conforme estes autores as desvantagens são:

a) exige conhecimento avançado e grande tempo para elaborar alternativas de respostas;
b) geralmente ocupam maior espaço na prova, consequentemente gastando mais papel;
c) possibilita uma facilidade considerável de cola entre os avaliados.

Existem algumas normas que devem ser seguidas para elaboração deste tipo de questão, tais como (MEDEIROS, 1983; MELCHIOR, 2002):

a) todas as alternativas usadas devem realmente induzir que estão corretas;

b) usar na maioria frases positivas, porém no uso de negativas, estas devem ser destacadas;
c) fazer enunciados curtos e mais objetivos possíveis, para facilitar uma a compreensão;
d) colocar no enunciado o máximo de palavras que ajudarão a resumir as opções;
e) o uso de respostas de questões dissertativas anterior mente aplicadas podem ajudar na elaboração de possíveis respostas.

Um exemplo de questão construída usando as normas e passos citados anteriormente pode ser observado na figura 2.

Figura 2 - Questão de múltipla escolha

Na região de Aimorés, Minas Gerais, está sendo construída uma grande hidrelétrica para obtenção de energia. A localidade de Itueta será totalmente inundada para a formação da represa. Essa prática pode trazer alguns problemas ambientais como:

I. Alteração na diversidade das espécies de peixes.
II. Diminuição das áreas de terras para agricultura.
III. Empobrecimento geral do solo da região.
IV. Expansão de habitats de vetores de doenças.

Os problemas que realmente podem ocorrer são:

A) I, II e III.
B) I, II e IV.
C) I, III e IV.
D) II, III e IV.

Fonte: Minas Gerais (2011).

2.3.3 Questões de associação

Consiste geralmente em duas colunas com itens alinhados, na qual o avaliado deve relacionar os itens de uma coluna com os da outra. Cada elemento de uma coluna pode ou não corresponder à apenas uma alternativa da outra coluna, podendo ser uma relação de muitos para um (MELCHIOR, 2002).

De acordo com Vianna (1976), como principais dentre as vantagens desse tipo de questão pode-se citar:

a) simples elaboração, pois o enunciado resume-se em instruções;
b) sua resolução é fácil e rápida;

c) abrangem um conteúdo sucinto.

Ainda conforme o autor as desvantagens são:

a) grande chance de acerto por sorte.

b) caso uma alternativa seja relacionada errada, isso provoca uma reação em cadeia, fazendo com que outras relações também fiquem incorretas.

Segundo Medeiros (1983) e Melchior (2002) alguns critérios devem ser considerados na elaboração deste tipo de questão, tais como:

a) nos enunciados, sempre que possível elaborar frases positivas;

b) no enunciado deve estar descrito, qual é a forma de associação das alternativas, como por exemplo, ligando ou enumerando;

c) descrever no enunciado qual é a relação das colunas, se é de muitos para um, ou um para um;

d) dar preferência para relações de muitos para um;

e) não construir alternativas que irão abranger o conteúdo das outras.

Um exemplo de questão construída usando as normas e passos citados anteriormente pode ser observado na figura 3.

Figura 3 - Questão de associação

Associe as duas colunas, relacionando os elementos musicais à sua definição.

1. Escala.

2. Harmonia.

3. Melodia. 4. Ritmo.

( ) Conjunto de sons dispostos em ordem simultânea.

( ) Conjunto de sons dispostos em ordem sucessiva.

( ) Disposição complexa de notas numa sequencia de durações curtas e longas dentro de um ou vários compassos.

( ) Progressão de notas em ordem ascendente ou descendente.

A sequencia correta dessa associação é

A) (1), (2), (3), (4).

B) (2), (3), (4), (1).

C) (3), (2), (4), (1).

D) (4), (2), (1), (3).

Fonte: Minas Gerais (2011).

2.3.4 Questões de ordenação

Esta questão impõe ao aluno numerar uma sequencia correta, estabelecendo uma ordem para as alternativas. As vantagens dessas questões são (MEDEIROS, 1983; MELCHIOR, 2002):

a) exige pouco tempo para sua elaboração e resolução;
b) possibilita extrair do aluno a capacidade de fazer relações sobre um contexto.

Ainda conforme estes autores as desvantagens são:

a) todas as alternativas são dependentes, então caso uma esteja errado, a grande chance de todas estarem;
b) sua correção demora mais em relação as outras objetivas, devido que pode ocorrer acerto de apenas uma determinada sequencia;
c) a aplicação se restringe a um conteúdo que tenha uma ordem ou estrutura.

As normas para construção das questões (MEDEIROS, 1983; MELCHIOR, 2002):

a) tentar colocar os itens em uma ordem lógica, como em ordem alfabética ou grau de importância, para evitar evidências de repostas;
b) deixar expresso onde o avaliado deve por os números da ordem;
c) estabelecer uma ordem clara de associação entre os itens.

Um exemplo de questão construída usando as normas e passos citados anteriormente cima pode ser observado na figura 4.

Figura 4 - Questão de ordenação

O processo de poluição global é desencadeado por etapas. Com base na indicação dos termos a seguir, preencha os quadros na ordem sequencial em que ocorrem as etapas.

1. Degradação ambiental.
2. Pressão demográfica.
3. Industrialização / expansão urbana.
4. Emissão de poluentes

A sequencia correta em que ocorre o processo é

A) 1, 2, 3, 4.

B) 3, 2, 4, 1.

C) 1, 3, 4, 2.

D) 4, 3, 2, 1.

Fonte: Minas Gerais (2011).

2.3.5 Questões de certo ou errado

Este tipo de questão contém sentenças a serem julgadas e assinaladas como certas ou erradas.

Segundo Medeiros (1983) e Melchior (2002) existem algumas vantagens sobre este tipo de questão:

a) facilitam a elaboração da avaliação, quando o prazo para sua produção é curto;
b) avaliações com este tipo de questão não precisam necessariamente serem examinadas pelo elaborador, elas podem ser entregues a auxiliares com o seu gabarito, para que possam ser corrigidas;
c) cada alternativa, ao ser julgada como certa ou errada, demonstra de forma direta o domínio do conteúdo pelo examinando.

E ainda com base nos autores as desvantagens são:

a) estas questões só podem ser elaboradas sob pontos obviamente certos ou errados, a fim de evitar ambiguidades;
b) podem fazer o examinando se confundir, tendo noções erradas daquilo que é certo ou errado.

A elaboração deste tipo de questão requer alguns cuidados, que irão torna-la válida (MEDEIROS, 1983; MELCHIOR, 2002):

a) elaborar nas alternativas frases curtas e objetivas;

b) certificar-se de que a questão não se baseia em um erro de interesse secundário, ou seja, não tornar a questão certa ou errada, fundamentada em detalhes;

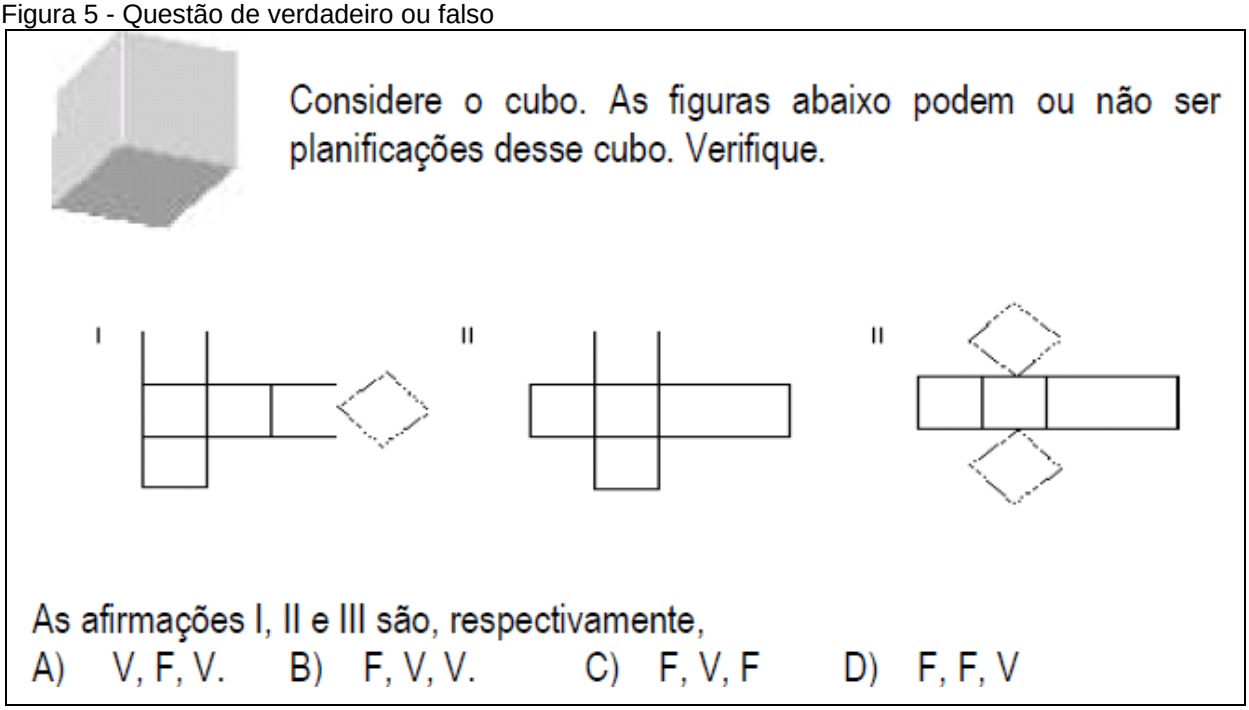
c) ao incluir frases negativas, estas devem ser destacadas;

d) elaborar cada questão com um único assunto. É recomendado fazer duas questões pequenas e específicas do assunto, do que apenas uma;

e) dentro da avaliação, balancear o número de afirmativas falsas e verdadeiras, porém não as dispondo em ordem metódica.

Um exemplo de questão construída usando as normas e passos citados anteriormente pode ser observado na figura 5.

Figura 5 - Questão de verdadeiro ou falso



Fonte: Minas Gerais (2011).

2.3.6 Questões de lacuna

O intuito deste tipo de questão é a de completar uma ou mais frases, preenchendo os espaços nela existentes. Cada questão deve conter apenas uma resposta correta (MELCHIOR, 2002).

Segundo Medeiros (1983) e Melchior (2002) podem-se citar algumas vantagens:

a) por serem questões que representam de forma natural um questionamento, são de fácil compreensão pelos alunos;
b) há a probabilidade reduzida de acerto por sorte, pois podem não oferecer opções de escolha;
c) se comparadas com as questões de dissertação, as questões de lacuna são mais fáceis de serem julgadas, por possuírem respostas limitadas.

Ainda de acordo com os autores as desvantagens são:

a) aparentemente são questões fáceis de serem preparadas, por isso costumam possuir falhas na elaboração;
b) esta questão não deve ser aplicada onde se quer extrair um alto grau de conhecimento do aluno;
c) o fato de apenas medir do aluno a capacidade de compreensão e memorização.

Segundo Medeiros (1983) e Melchior (2002) as questões de lacuna são de fácil construção, porém devem ser consideradas algumas normas, para que as questões sejam de qualidade. Entre elas:

a) evitar que mais de uma resposta possa estar correta;
b) a elaboração deste tipo de questão deve ser objetiva e precisa, não sendo necessária a adição de informações supérfluas;
c) uma forma de enriquecer a questão, é colocar mais de um item como resposta para a lacuna.

Um exemplo de questão construída usando as normas e passos citados anteriormente pode ser observado na figura 6.

Figura 6 - Questão de lacuna

Na atmosfera primitiva da Terra, predominavam os gases metano, hidrogênio, amoníaco e vapor de água. Admitindo-se a ausência do gás oxigênio nessa época, supõe-se que os primeiros seres vivos eram _____. Após o surgimento dos organismos _____ no ambiente, a atmosfera passou a ter gás oxigênio livre em sua composição química, permitindo o aparecimento dos seres _____.

Em sequencia as palavras que completam corretamente essas lacunas são:

A) aeróbicos, fotossintetizantes, anaeróbicos.

B) aeróbicos, heterótrofos, anaeróbicos.

C) anaeróbicos, fotossintetizantes, aeróbicos.

D) anaeróbicos, fermentadores, aeróbicos.

Fonte: Minas Gerais (2011).

2.3.7 Questão dissertativa ou de resposta livre

Esta questão tem como característica, deixar o aluno fazer a sua própria resposta, não escolhendo dentre algumas alternativas dispostas para ele. Esse tipo de questão pode ser facilmente formulado para que tenha um nível de dificuldade elevado e também para que tenha uma respostas simples, sendo muito flexível. As simples possuem geralmente enunciados com as palavras, *quem*, *quando*, *onde*. Estas palavras induzem respostas geralmente de uma palavra. As mais complexas já abrangem as palavras, *compare*, *explique*, *como*, contudo quando uma questão dissertativa é formulada, independe do seu grau de dificuldade existe algumas etapas que desejavelmente deve ser levadas em consideração (MELCHIOR, 2002; VIANNA, 1976).

As questões dissertativas são consideradas fáceis de serem elaboradas em relação as objetivas, as dissertativas não requerem um alto nível técnico para sua elaboração, essa facilidade pode se tornar aparente caso a questão não esteja bem formulada, dando subjetividade a quem for responder. Praticamente, o único fator que o professor tem que levar em consideração quando for elaborar um questão dissertativa, é a simplicidade do enunciado, deixando muito claro o que ele quer, como o item é de resposta aberta, o enunciado deve limitar as possibilidade de repostas indicando até quando a resolução deve abranger (MEDEIROS, 1983).

As questões dissertativas geralmente dão muito trabalho em sua correção, as repostas podem ser muito variadas, e cada resposta tem que ser

analisada individualmente, podendo então está estar totalmente correta, parcialmente correta, ou incorreta. Ela pode ser usada para extrair do aluno conhecimentos como, explicar, expor opinião, analisar e fazer relações. Outro fator importante é a previsão de tempo por item dissertativo, visto que ele é muito mais demorado para de ser resolvido pelo aluno (VIANNA, 1976).

Uma avaliação quando bem elabora, tende a dar o retorno esperado, as avaliações vão demonstrar quais são as aptidões dos alunos em relação ao conteúdo que ela aborda. Para conseguir enxergar os resultados dessa parte do processo de ensino, é necessário fazer uma análise dos resultados disponibilizados pelo exame, a partir dessa reflexão o professor pode mensurar se seu objetivo foi cumprido ou não, e também quais são as medidas que ele pode e deve levar em consideração nas próximas etapas do ensino.

2.4 REFLEXÕES DOS RESULTADOS DA AVALIAÇÃO

Depois de um demasiado trabalho na elaboração e aplicação da prova, chegou o momento de analisar os resultados obtidos por ela. Essa parte do período da avaliação é considerada a prova real para o professor e para os alunos, é onde todos vão saber qual foi sua eficiência perante aos seus objetivos. O que ajuda a garantir a qualidade de resultado positivos, é a elaboração de um bom teste, e também a prática de uma boa técnica de correção (VIANNA, 1976).

Um professor pode aplicar uma prova para muitos alunos, então terá que corrigir muitas provas, se ele colocar muitas questão dissertativas, seu trabalho será multiplicado entre o números de alunos e o número de questões dissertativas, levando ainda em consideração que há a possibilidade de ser feitos testes do mesmo conteúdo mas com questão diferentes ou com ordens trocadas, tudo isso faz com que a demanda de tempo para correção do teste seja elevadíssima, tendo que ser levada em consideração (VIANNA, 1976).

Ainda de acordo com Vianna (1976), para facilitar a correção de testes existem algumas características da avaliação que podem ajudar, como a organização das questões, por área, por ordem de peso, ou qualquer outro critério que julgar-se necessário para facilitar na correção. O uso de gabaritos são práticos para as respostas de questões objetivas.

É aconselhável que as provas sejam corrigidas anonimamente, apurando a mesma questão de cada avaliação de uma só vez, e se possível fazer que estas correções sejam feitas por mais de uma pessoa, estas técnicas ajudam e aumentam a velocidade e precisão do resultado (MELCHIOR, 2002; VIANNA, 1976).

Depois da pontuação feita, chamada também de *scores*, acabou-se a primeira parte da apuração dos resultados, os escores sozinhos não representam muita coisa, mas a análise e síntese deles vão demonstrar como os alunos se saíram (MEDEIROS, 1983).

É aconselhável que cada prova seja realmente um instrumento de ensino, para que o professor consiga destacar e extrair de cada prova as dificuldades encontradas pelo aluno, assim pode-se elaborar uma medida para reforçar ou realçar os conteúdos que ele acredita serem indispensáveis, fazendo uma revisão com a turma ou individualmente de pontos que ele julga importantes. A nota acaba sendo a única visão de um teste, só que sozinha a ela não representa nada, porém a interpretação da avaliação vai decidir o que o professor deixou a desejar e onde ele deve se impor para ajustar e nivelar o aprendizado (LINDEMANN, 1987; MEDEIROS, 1983).

3 TECNOLOGIAS PARA O DESENVOLVIMENTO

Para o desenvolvimento de uma aplicação *web*, é necessário avaliar e integrar as ferramentas que serão utilizadas para a produção do sistema. Existem muitas linguagens de programação, *Application Programming Interface* (API) e *frameworks*, alguns voltadas para a *web* outros não. A escolha das tecnologias é o primeiro passo na construção de uma aplicação seguido da definição de sua arquitetura. Dependendo das escolhas, podem surgir dificuldades ou facilidades em programar recursos no sistema, então a grande questão é analisar as ferramentas que serão utilizadas para que o aplicativo atinja o seu objetivo (CADENHEAD, 2013, tradução nossa).

3.1 COMO APLICAÇÕES WEB FUNCIONAM

As aplicações *web* seguem uma estrutura quando estão em execução na internet, toda aplicação *web* está em um *web server***,** que é um computador ligado em tempo integral, disposto para atender pedidos de um *web browser***.** Conhecidos popularmente como navegadores, eles tem como função pedir ao *web server*, solicitações de páginas *HyperText Markup Language* (HTML) (DUCKETT, 2010; HENICK, 2010; WILLARD, 2009, tradução nossa).

Os *web browsers* sabem para qual servidor eles tem que pedir a página à partir de uma *Uniform Resource Locator* (URL), URL é um endereço de um site na internet, todo site tem um endereço, por exemplo o endereço do *Google* é *www.google.com*. Assim que recuperada estas páginas são exibidas para o usuário do navegador. Alguns modelos de navegadores conhecidos são: *Google Chrome*, *Mozilla FireFox* e *Internet Explore* (IE) (DUCKETT, 2010; HENICK, 2010; WILLARD, 2009, tradução nossa).

A definição citada anteriormente separa as aplicações *web* em duas partes, uma parte de código que é executa no próprio *web browser*, ou seja, *client side*, e a outra parte são os códigos que são executados no servidor, *server side*. Nos próximos capítulos, abordaremos quais são os tipos e nomes dessas tecnologias que são executados nos dois lados, tanto no *client* como no *server*.

3.2 CLIENT SIDE

As linguagens de programação usadas no lado do *client* são normalmente interpretadas pelos *web browsers*, elas tem como principal vantagem a velocidade de execução, pois rodam diretamente no *client*, evitando um pedido para o servidor. As linguagens geralmente usadas em aplicações *web* são HTML, *Cascading Style Sheets* (CSS) e *JavaScript* (JS) (DUCKETT, 2010, tradução nossa).

3.2.1 HTML

O HTML é uma linguagem de marcação interpretada pelos *browsers*, ela é usada na criação de documentos, resultando em páginas *web*. Os documentos HTML são baseados em uma estrutura de *tags*, possibilitando a adição de campos nas páginas, também como *links*, tabelas, imagens, vídeos entre outros componentes, todos estes são usados para montar a estrutura da página *web* (GOLDSTEIN; LAZARIS; WEYL, 2011; HENICK, 2010, tradução nossa).

O HTML está em desenvolvimento desde 1992, onde já passou por muitas versões. A versão atual é a 5, a qual ainda não é aceita por todos os *browsers*. Nesta versão os formulários ganharam alguns recursos importantes, como a validação de campos sem a utilização de JS, a possibilidade de colocar *Expression Linguage* (EL) direto no código HTML para fazer validações, entre outras atualizações (SILVEIRA, 2013, tradução nossa).

O HTML fornece formulários completos, com recursos para títulos, cabeçalhos, rodapés, corpo de textos e campos de preenchimento, porém estes formulários quando construídos apenas com HTML, tem uma aparência nada agradável para o usuário, necessitando da intervenção de outra linguagem para melhorar o visual, sendo acionado então o CSS (SCHAFER, 2010, tradução nossa).

3.2.2 CSS

O CSS é uma linguagem de programação interpretada pelos *browsers*, ela prove estilos de apresentações para formulários escritos em uma linguagem de marcação, como HTML ou *eXtensible Markup Language* (XML), o principal benefício

é que ela fornece a separação entre os estilos da página e o formato do documento (DUCKETT, 2010; SCHAFER, 2010, tradução nossa).

O funcionamento do CCS é em torno de rubricas, cada rubrica é equivalente uma série de estilos, então cada elemento de um documento que for anotado com esta rubrica terá os estilos da mesma. Os navegadores podem variar em qual versão do CSS eles suportam, os mais antigos trabalham com CSS1, já os mais novos como *Google Chrome* e *Mozilla Firefox* aceitam a versão CSS3 (SCHAFER, 2010, tradução nossa).

Com CSS e HTML juntos, temos em mãos um conjunto de ferramentas para criação de páginas *web* bem estruturadas e com bons visuais, faltando apenas fazer com que esses formulários fiquem com uma estrutura dinâmica. Para tornar as páginas dinâmicas existe o JS (DUCKETT, 2010; WRIGHT, 2013, tradução nossa).

3.2.3 JavaScript

JavaScript é uma linguagem interpretada pelo *web browser*, sendo a principal linguagem de programação na parte do *client*. Ela foi criada com o propósito de fazer com que trechos de código sejam executados direto no lado do *client*, assim eliminando a necessidade de fazer um pedido para o servidor. Ela é uma linguagem orientada a objetos fundamentada em protótipos, não tem uma tipagem forte, contudo é dinâmica e usa funções de primeira classe (MACCAW 2011, tradução nossa).

Este tipo de linguagem tem um uso interessante quando se tenta tornar páginas *web* mais dinâmicas, porém em certos casos há necessidade de escrever grandes códigos para fazer funções relativamente simples, principalmente quando se tenta utilizar requisições *Asynchronous Javascript and* XML (AJAX), que são requisições assíncronas, estas são submetidas para o servidor por diversas ações diferentes da página, e não necessariamente precisam levar todos os campos do formulário (RUEBBELKE, 2012, tradução nossa).

Pensando na dificuldade em programar recursos complexos com JS, começaram a surgir diferentes *frameworks* que controlam funções mais onerosas para os programadores, como por exemplo, o AngularJS, este *framework* foi desenvolvido com objetivo principal de facilitar formulários com requisições AJAX (GREEN; SESHADRI, 2013, tradução nossa).

3.2.3.1 AngularJS

AngularJS é um *framework* JS mantido pela Google e seu principal objetivo é auxiliar na criação de páginas *web*. Ele foi construído sobre o principio de programação declarativa, isso quer dizer que sua programação declara o que ela faz, não declarando então como seus procedimentos irão funcionar. A linguagem declarativa é muito eficaz na criação de interfaces que irão interagir com o usuário, sendo que, esse é o propósito do AngularJS (GREEN; RUEBBELKE, 2012; SESHADRI, 2013, tradução nossa).

O funcionamento do AngularJS é baseado em HTML, o qual conterá *tags* especiais, que podem ser lidas e identificadas pelo *framework*. Ele traz um recurso chamado *Data-Binding*, que cuidará de todo o gerenciamento de transição e manipulação do *Document Object Model* (DOM), o DOM é a especificação que permite alterações dinâmicas nas estruturas das páginas, então o AngularJS trata todo esse gerenciamento, automatizando uma parte de código que normalmente é feita pelo desenvolvedor (DARWIN; GREEN; KOZLOWSKI; SESHADRI, 2013, tradução nossa).

O AngularJS trata o documento HTML sempre como DOM, dessa forma ele permite trabalhar com *templates* de páginas, a vantagem deste recurso é a possibilidade de estreitar a relação entre os desenvolvedores de páginas e os *web* designers, dando liberdade para o designer fazer o seu trabalho, colocando estilos no HTML sem ter conflito com código do desenvolvedor (DARWIN; GREEN; KOZLOWSKI; SESHADRI, 2013, tradução nossa).

A arquitetura *Model View Controller* (MVC) é um padrão para estruturação de projetos *web*, também como *frameworks*. Este padrão é separado por três camadas *Model,* sendo o modelo, essa camada é a que faz referencia a estrutura do banco de dado, a *View*, ou seja, visão, nesta camada é apresentada os componentes que são visíveis pelos usuários, e *controller,* controlador, o qual é a camada que controla toda a lógica de negócios da aplicação, também fazendo a ligação entre o modelo e a visão (DUCKETT, 2010, tradução nossa).

A arquitetura do AngularJS é baseada no padrão MVC, porém esta pode ter diversas interpretações em relação a autores e desenvolvedores diferentes, no caso do angular sua estrutura se aproxima de algo como *Model View Presenter* (MVP), muito parecido com o MVC, porém o *Presenter* faz o papel do controlador do

MVC com algumas distinções, porém está camada faz a ligação do modelo com visão. Esse padrão proporciona vantagens como, injeção de dependência, inversão de controle, desacoplamento, entre outros (DARWIN ; KOZLOWSKI, 2013, tradução nossa).

Para o desenvolvimento da parte *client side* foram apresentadas ferramentas que contém suas funções separadas e distintas, o HTML cuidará da estrutura na página, o CSS do estilo da mesma e o JS fará a pagina interagir amigavelmente com o usuário. Nos próximos capítulos serão abordados os recursos e ferramentas que estão do outro lado, ou seja, no servidor.

3.3 *SERVER* SIDE

O outro lado de uma aplicação *we*b é o do servidor, sendo responsável por fazer o processamento pesado e acesso a banco de dados da aplicação. A vantagem de utilizar o servidor de aplicação é abstrair o usuário de quais softwares estão sendo utilizados para processar o conteúdo. Outra vantagem é que considerando a necessidade de alto poder de processamento concentrada no servidor, a aplicação fica independente da configuração da máquina do usuário (BURKE, 2010, tradução nossa).

Existem conceitos e várias linguagens de programação do lado do servidor, porém nos próximos capítulos iremos abordar a linguagem de programação Java e um conceito de desenvolvimento chamado REST, os dois juntos irão fornecer serviços que integrarão com o lado do *client*.

3.3.1 Java

A linguagem Java existe desde 1990 quando começou a ser desenvolvida pela *Sun Microsystems*, a popularidade e aceitação do Java faz-se devido o fato que ela é executada em cima de uma *Java Virtual Machine* (JVM), então qualquer Sistema Operacional (SO) ou aparelho eletrônico que consiga rodar a JVM, consegue ler e executar programas feitos em Java, consagrando ela como uma linguagem multiplataforma (LUCKOW; MELO, 2010, tradução nossa).

A linguagem tem como características ser *open source*, ou seja, de código aberto, ser fortemente tipada e orientada a objetos. Java é compilado pela JVM que

gera os bytecodes, e depois a mesma interpreta para que o SO compreenda, sendo assim Java é uma linguagem interpretada e compilada (EVANS, 2013; HUGHES, 2002; VERBURG, 2013, tradução nossa).

Dentre as principais linguagens de programação, Java foi a primeira a ser moldada para *World Wide Web* (WWW). Ela possibilita fazer programas tradicionais na nuvem, além de ter muitas características especiais e bibliotecas que provem recursos e facilidades em desenvolver programas voltados a *web*. Estes incluem suporte extensivo para interfaces gráficas, a capacidade de incorporar um programa Java em um documento *Web*, fácil comunicação com outros computadores, e a capacidade de escrever programas que são executados em paralelo ou em vários computadores ao mesmo tempo (HUGHES, 2002; MCDOWELL; POHL, 2006, tradução nossa).

A versão mais recente do Java é a 7, atualmente desenvolvida pela *Oracle*. Para que se possa desenvolver aplicativos em Java, é necessário ter instalado a *Java Development Kit* (JDK) compatível com a versão do Java que deseja-se usar no computador, também é aconselhável o uso de ferramentas como *Integrated Development Environment* (IDE), as mais populares e comercialmente usadas são grátis, são elas *NetBeans* e *Eclipse* (CADENHEAD, 2013, tradução nossa).

A distribuição do Java é feita em três diferentes plataformas, o Java *Standard Edition* (SE) que dá suporte para construção de aplicações desktop, o Java *Micro Edition* (ME) destinado a suprir as necessidades de desenvolvimento de aplicativos para dispositivos móveis e por fim o Java *Enterprise Edition* (EE) que é voltado para aplicações que rodam em ambientes distribuídos, como aplicações *web*. No próximo capitulo iremos abordar a plataforma EE (EVANS; VERBURG, 2013, tradução nossa).

3.3.2 Java EE

Com um mundo empresarial movimentado, os aplicativos precisam acessar dados, aplicar a lógica de negócios, adicionar camadas de apresentação, ser móvel, utilizar geolocalização e se comunicar com sistemas externos e serviços online. As empresas estão procurando usar tecnologias robustas que podem suportar o seu crescimento. Estas aplicações devem ser ágeis e capazes de escalar

para acomodar a crescente demanda de usuários (GONÇALVES, 2013; GUPTA, 2013).

Java EE surgiu no final da década de 1990 e trouxe para a linguagem Java uma alta gama de ferramentas para o desenvolvimento empresarial (GUPTA, 2013, tradução nossa).

Java EE é uma plataforma baseada em padrões para o desenvolvimento de aplicações corporativas e *web*. Estas aplicações são normalmente compostas em múltiplas camadas, com uma camada de interface consistindo em um *framework web*, uma camada intermediária proporcionando segurança e transações, e uma camada de conectividade proporcionando níveis de acesso à um banco de dados (GUPTA, 2013, tradução nossa).

Os padrões abertos são coletivamente um dos principais pontos fortes do Java EE. Uma aplicação escrita com tecnologias padrões do JAVA EE é facilmente portável através da servidor de aplicação. Open source é outro ponto forte do Java EE, contendo suporte e licenciamento com fontes abertas como *GlassFish*, *EclipseLink*, *Hibernate Validator*, *Mojarra* e *Jersey* (GONÇALVES, 2013, tradução nossa).

Foi lançado em junho de 2013 a versão 7 o qual fornece uma maneira mais fácil de usar e construir aplicações *web* corporativas em relação as versões anteriores 5 e 6. A plataforma Java EE define API's de componentes diferentes em cada camada, e também fornece alguns serviços adicionais, tais como, injeção, e gestão de recursos que se estendem através da plataforma. Ela é composta por cerca de trinta especificações, valendo citar os nomes de algumas importantes (GONÇALVES, 2013; GUPTA, 2013, tradução nossa).

Na camada de negócios:

a) *Contexts and Dependency Injection* 1.1 (CDI);

b) *Bean Validation* 1.1;

c) *Enterprise Java Bean* 3.2 (EJB);

d) *Java Persistence* API 2.1 (JPA).

Camada *web*:

a) *Servlet* 3.1;

b) *Java Server Faces* 2.2 (JSF);

c) EL 3.0.

E na camada de interoperabilidade:

a) Java API for *eXtensible Markup Language Web Services* 2.3 (JAX-WS);
b) Java API for *Representational State Transfer* (REST) *Web Services* 2.0 (JAX-RS).

A linguagem Java é largamente usada no desenvolvimento de aplicativos *web*, possuindo extensões para a plataforma Java EE, as ferramentas dispostas abrangem todas as camadas que uma aplicação necessita, na camada de interoperabilidade existe a JAX-RS que será obordada a seguir (GUPTA, 2013, tradução nossa).

3.3.3 REST

REST é um estilo arquitetural baseado em como a *web* funciona. Aplicado aos serviços, ele tenta traduzir os verbos da *web* em seu sentido literal. Para criar um serviço *web* REST, precisa-se conhecer o *Hypertext Transfer Protocol* (HTTP) e *Uniform Resource Identifier* (URI) e observar alguns princípios de design (BURKE, 2010; GONÇALVES, 2013, tradução nossa).

A arquitetura REST rapidamente se tornou popular porque ela conta com um robusto protocolo de transporte HTTP. Serviços *web* REST reduzem o acoplamento entre a parte do *client* e do server, tornando-a muito mais fácil de desenvolver uma interface REST ao longo do tempo, sem precisar alterar os clientes existentes quando há necessidade de fazer alguma manutenção. Além disso, os serviços são fáceis de construir, pois não necessitam de nenhum kit de ferramentas *Web Services Description Language* (WSDL) (BURKE, 2010; GUPTA, 2013, tradução nossa).

Podemos definir que REST é uma arquitetura baseado em serviços, quando um serviço é criado ele pode ser consumido por alguém. Esse serviço fica esperando ser requisitado, as requisições podem ser *get*, *post*, *delete*, *put*, entre outros. Dependendo de como os serviços são criado eles podem retornar vários formatos, como por exemplo, XML ou *JavaScript Object Notation* (JSON), além disso no cabeçalho de retorno sempre há um *status* indicando como a requisição foi tratada (NEWMARCH, 2010, tradução nossa).

Para conseguir localizar os serviços REST deve se obervar as URI, ou seja, o endereço do serviço, mas a mesma URI pode conter muitos tipos de serviços, sendo distinguidos pelo tipo de requisição que é feita, por exemplo, quando

se solicitada um requisição HTTP/GET para um serviço, apenas a parte *get* do serviço será acionada. G*et*, *post*, *put*, *delete* são os mais comuns verbos HTTP, cada um desses tem uma função específica, o *get* irá retornar algo, servindo então para buscas, o *put* é usado para inserção, o *post* para atualização e *delete* para deletar (BURKE, 2010; GUPTA, 2013, tradução nossa).

Nas respostas das requisições feitas para os serviços, contém representações em códigos do protocolo HTTP, como, 200, 201, 404, 500 entre outros. Esses códigos servem para dizer à página como foi o sucesso ou insucesso da requisição feita. Quando é retornado o código 201, significa que aconteceu um cadastro bem sucedido no servidor, quando o retorno for o código 404 significa que o servidor não conseguiu encontrar o que pediram para ele achar. Existem códigos para cada determinada função, sendo que é a partir destes códigos que as páginas vão conseguir se orientar para exibir o retorno da requisição (SILVA, 2008, tradução nossa).

4 TRABALHOS CORRELATOS

4.1 PROTÓTIPO DE UM SISTEMA PARA ACOMPANHAMENTO DE TRABALHOS DE CONCLUSÃO DE CURSO

Este trabalho foi defendido na Universidade Regional de Blumenau pela acadêmica Elisângela Cristina Lombardi Klitzke em 2009, com propósito de desenvolver uma aplicação *web*, que forneça um auxilio para estudantes e orientadores que estão envolvidos com a disciplina de TCC. Dentre as tecnologias usadas no desenvolvimento da aplicação destaca-se JavaScript com auxílio do *framework* jQuery*,* o uso do banco de dados de modelo relacional MySQL e Hypertext PreProcessor (PHP) e CSS no desenvolvimento das páginas. A aplicação final forneceu as principais funcionalidades para que atenda os requisitos mínimos propostos, mas não ficou compatível com todos os navegadores *web*, contendo quebras de *layout* no IE 7 por exemplo (KLITZKE, 2009).

4.2 FERRAMENTO WEB PARA MODELAGEM LÓGICA EM PROJETOS DE BANCOS DE DADOS RELACIONAIS

Trabalho defendido por Paulo Alberto Bugmann em 2012 na Universidade Regional de Blumenau. Este trabalho tem como objetivo fazer um aplicativo que auxilie na construção da modelagem de bancos de dados relacionais. O contexto do aplicativo é *web*, usando a plataforma Java e arquitetura MVC no desenvolvimento dos padrões de projeto. O uso do AJAX se fez necessário para tornar em alguns momentos a aplicação mais dinâmica para o usuário. O trabalho foi desenvolvido baseando-se no trabalho WebModeler de Bachmann desenvolvido em 2007, visando aperfeiçoar, incluir recursos e funcionalidades. As principais funções fornecidas pelo sistema são a criação de diagramas para modelagem do banco, geração de scripts para construção do banco e elaborar diagramas *Unified Modeling Language* (UML) de casos de uso graficamente (BUGMANN, 2012).

4.3 REDUZINDO ACOPLAMENTO E AUMENTANDO A ESCALABILIDADE DE SISTEMAS CORPORATIVOS ATRAVÉS DE REST

Trabalho defendido por Jackson Dos Santos Oliveira em 2011 na universidade Feevale. O objetivo é mostrar o uso de conceitos REST para desenvolvimento de aplicações de médio e grande porte. Realçando vantagens do uso de REST como escalabilidade e desacoplamento. O trabalho demonstra o desenvolvimento de uma aplicação usando a teoria estudada. Para melhor demonstração das vantagens na aplicação do conceito foi escrito e ilustrado um passo a passo do desenvolvimento da aplicação. Enfim são apresentadas detalhes, vantagens e conclusões sobre o uso do conceito REST para ajudar o desacoplamento na integração de sistemas (OLIVEIRA, 2011).

4.4 AVALIAÇÃO: SENTENÇA OU DIAGNÓSTICO

Trabalho de conclusão de curso desenvolvido por Fernanda Müller Heuser em 2008, apresentado na Universidade Federal do Rio Grande do Sul. O trabalho é sobre a avaliação da educação básica, com objetivo de investigar e saber qual a percepção de coordenadores, professores, pedagogos e alunos de ensino fundamental e médio sobre a avaliação educacional. Os pontos levantados em entrevistas com os elementos citados foram sobre a importância da avaliação, qual seria a finalidade da avaliação e como devem ser levados em considerações às notas obtidas. Com os estudos realizados neste trabalho concluiu-se que a avaliação tem um importante papel no ensino, na visão dos professores, eles possuem um bom conhecimento sobre a prática avaliativa, mas ela ainda deve ser estudada para ser aplicada com consciência. Na visão dos alunos a avaliação que trás resultados mais satisfatórios são as bem elaboradas, na maioria das vezes as avaliações são mal feitas, não tendo uma boa formulação, explicação e comentários importantes, causando até dupla interpretação. O trabalho relata que outros estudos podem ser feitos e aprofundados nesses requisitos, pois ainda é bem carente. (HEUSER, 2008).

5 DESENVOLVIMENTO DO PROTÓTIPO

O trabalho consiste no desenvolvimento de um protótipo para auxiliar os professores a elaborarem suas avaliações de forma cooperativa. O software permite que eles cadastrem e compartilhem questões entre si, podendo elaborar e gerar suas avaliações com questões feitas por outros professores. Um mecanismo de busca possibilita encontrar e selecionar as questões que irão compor as avaliações e gera-las facilmente.

A metodologia se divide em quatro etapas bem definidas, a primeira delas é o levantamento de requisitos, o qual foi definido quais funcionalidades o sistema deveria ter para atender seus objetivos. A segunda é o desenvolvimento do protótipo baseado nos critérios definidos no levantamento de requisitos, a terceira é uma fase testes funcionais e a última é a análise dos resultados gerados.

5.1 LEVANTAMENTOS DOS REQUISITOS

Os requisitos necessários que o sistema deveria ter foram baseados no levantamento bibliográfico, mas o principal critério foi a pesquisa realizada com os professores da UNESC. A pesquisa visou mostrar o interesse dos professores no projeto, sua aceitação e alguns pontos que o mesmo deve conter para atender a necessidade a ser resolvida.

5.1.1 Pesquisa com os professores da UNESC

A pesquisa foi elaborada com oito questões sobre as aplicações de avaliações dos professores. Esse questionário foi aplicado através do Google Docs com professores de diversos cursos do ensino superior da UNESC, o questionário obteve noventa e seis respostas e ele pode ser observado conforme apêndice A.

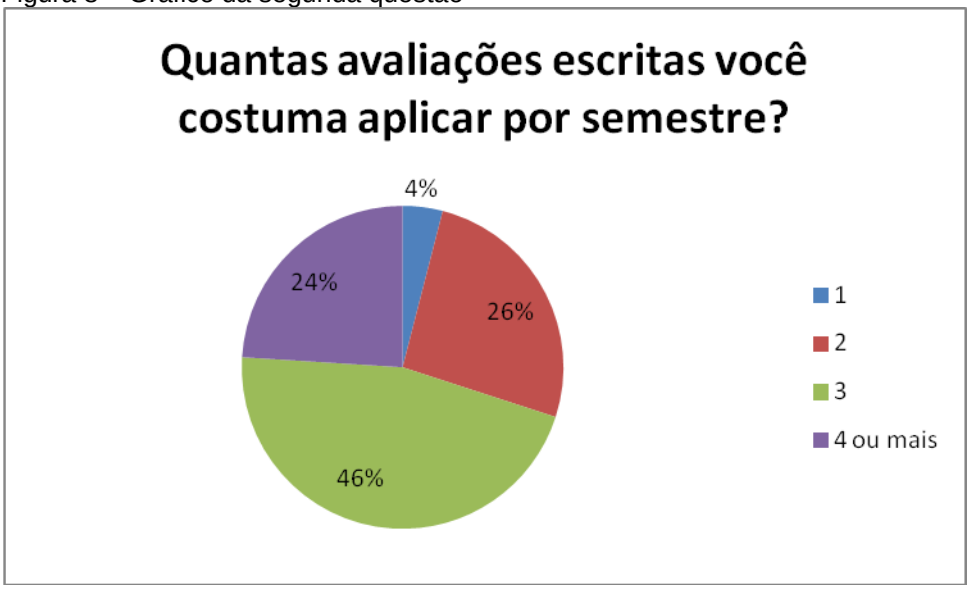
A primeira questão perguntava qual era o tempo, em anos, de experiência de cada professor em atuação, 6% responderam menos de um ano, 22% responderam de um a três anos, 11% de quatro a seis anos e 61% responderam seis ou mais anos. A figura 7 demonstra os resultados obtidos nesta questão.

Figura 7 – Gráfico da primeira questão

Fonte: Do autor.

A segunda questão perguntava quantas avaliações costumam aplicar por semestre, 4% responderam menos de uma avaliação, 26% responderam duas, 46% três e 24% responderam quatro ou mais avaliações. A figura 8 demonstra os resultados obtidos nesta questão.

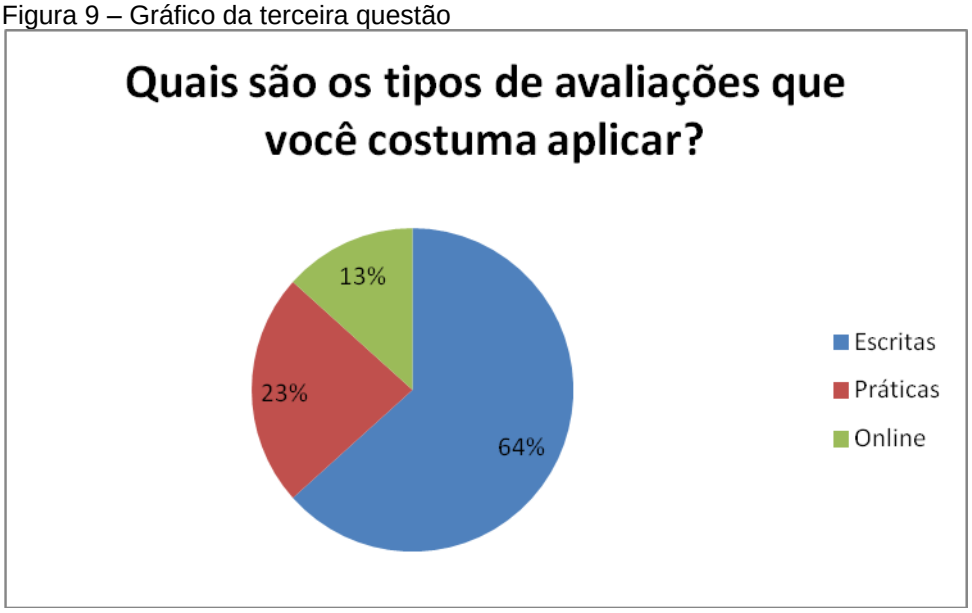
Figura 8 – Gráfico da segunda questão



Fonte: Do autor.

A terceira questão perguntava quais são os tipos de avaliações costumam aplicar por semestre, 57% responderam que aplicam avaliação escrita, 31% avaliação prática e 12% responderam avaliação online. A figura 9 demonstra os resultados obtidos nesta questão.

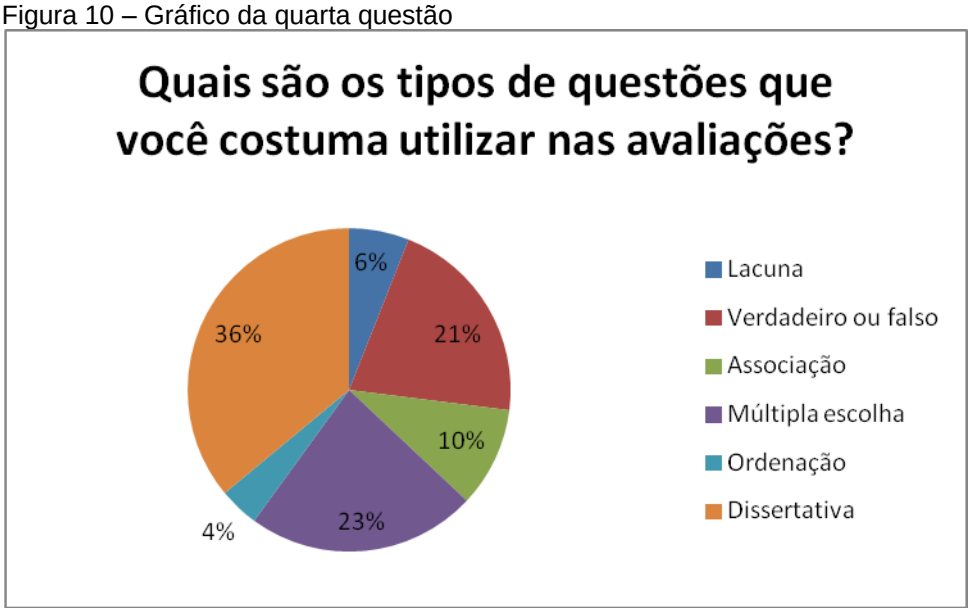
Figura 9 – Gráfico da terceira questão



Fonte: Do autor.

A quarta questão perguntava quais são os tipos de questão que geralmente são utilizadas nas avaliações, 13% responderam lacuna, 21% verdadeiro ou falso, 10% associação, 23% múltipla escolha, 4% ordenação e 36% responderam dissertativa. A figura 10 demonstra os resultados obtidos nesta questão.

Figura 10 – Gráfico da quarta questão

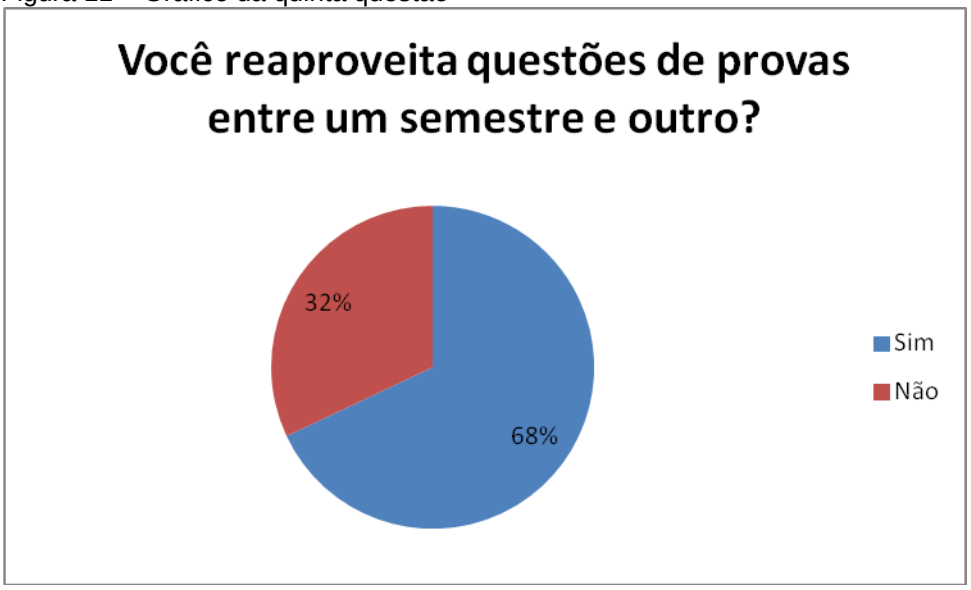


Fonte: Do autor.

A quinta questão perguntava se o professor reaproveita questão de avaliações entre um semestre e outro, 68% responderam que aproveitam questões e

32% responderam que não. A figura 11 demonstra os resultados obtidos nesta questão.

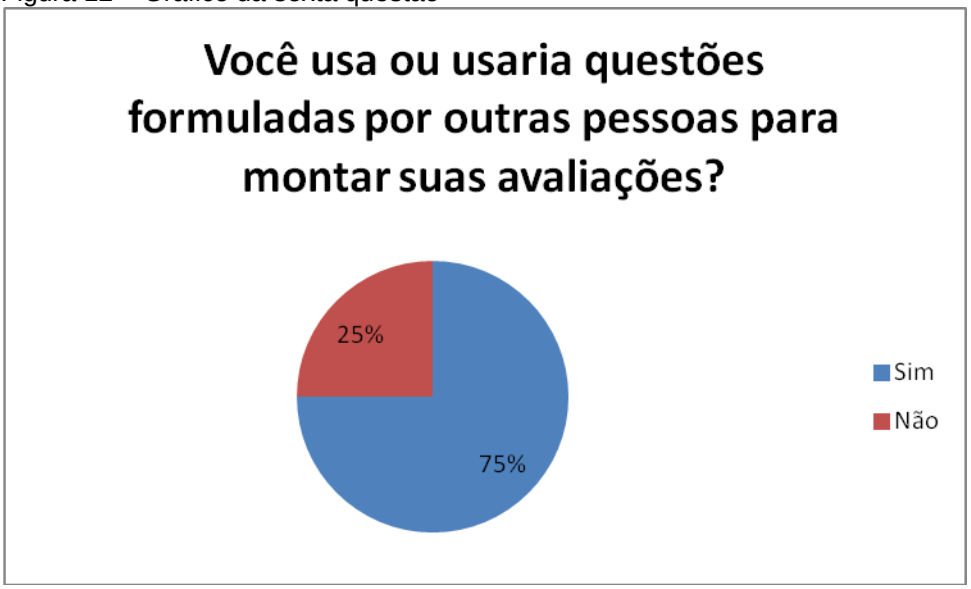
Figura 11 – Gráfico da quinta questão



Fonte: Do autor.

A sexta questão perguntava se o professor usa ou utilizaria questões formuladas por outras pessoas em suas avaliações, 75% responderam que sim e 25% responderam que não utilizariam. A figura 12 demonstra os resultados obtidos nesta questão.

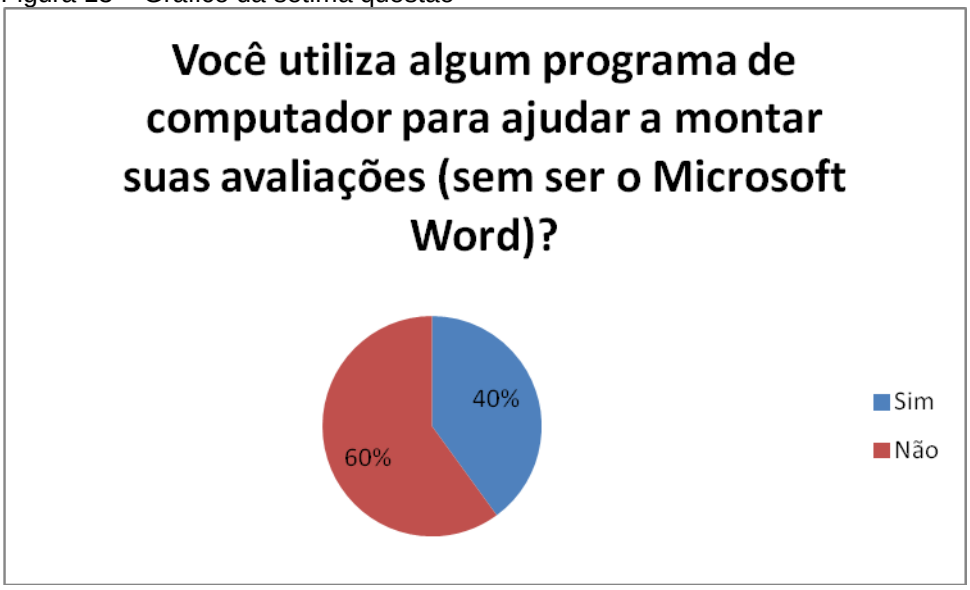
Figura 12 – Gráfico da sexta questão



Fonte: Do autor.

A sétima questão perguntava se o professor utiliza algum programa de computador, sem ser o Microsoft Word, para elaboram suas avaliações, 40% responderam que sim e 60% responderam que não utilizam. A figura 13 demonstra os resultados obtidos nesta questão.

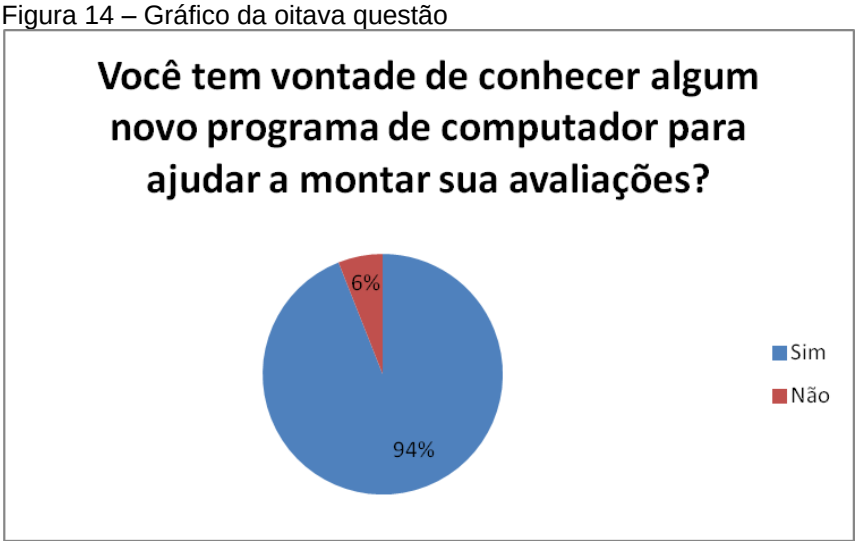
Figura 13 – Gráfico da sétima questão



Fonte: Do autor.

A oitava e última questão perguntava se o professor tem vontade de conhecer um programa de computador que ajudaria e montar as avaliações, 94% responderam que sim e 6% responderam que não utilizariam. A figura 14 demonstra os resultados obtidos nesta questão.

Figura 14 – Gráfico da oitava questão



Fonte: Do autor.

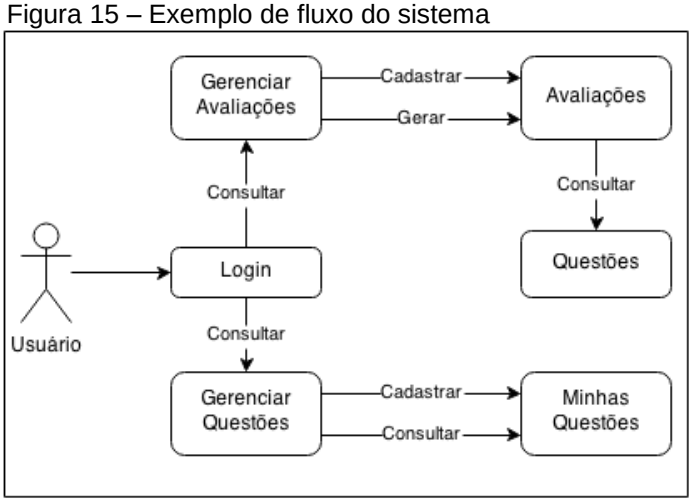
Com o resultado do questionário aplicado, também baseado nos levantamento bibliográficos, definiram-se quais seriam os requisitos para que o sistema atende-se a proposta a ser cumprida. Então na próxima etapa serão abordadas quais foram as estratégias utilizadas para que o sistema atendesse a necessidade.

5.1.2 Requisitos

Nos requisitos do protótipo foi definido que ele deve possuir uma tela de cadastro de questões, onde a mesma deve possibilitar gravar todos os tipos de questões mencionados neste trabalho. Uma tela de elaboração de avaliações, contendo um mecanismo de pesquisa que facilite encontrar as questões desejadas e uma tela para organizar a estrutura das avaliações e depois gera-las para impressão.

O fluxo de interação é de que os usuários possam entrar na aplicação através de um *login*, desta forma as questões ficam vinculadas no usuário. Na tela de questões é possível visualizar apenas as que foram cadastradas pelo próprio usuário, também podendo editar e excluir as mesmas. Na tela de avaliações as funções possíveis são localizar as avaliações geradas e também edita-las, excluí-las ou gera-las para impressão. Na figura 15 pode-se observar um possível fluxo de navegação no sistema.

Figura 15 – Exemplo de fluxo do sistema



Fonte: Do autor.

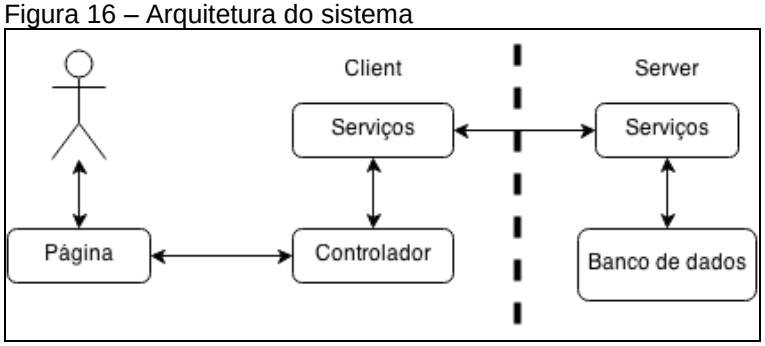
Determinados os requisitos mínimos, a estrutura do protótipo, o fluxo de interação do usuário e as tecnologias a serem utilizadas, a próxima etapa é o inicio do desenvolvimento do protótipo utilizando os conceitos definidos até o momento.

5.2 CONFIGURAÇÕES DO SERVIDOR DE APLICAÇÃO E CRIAÇÃO DO PROJETO

Para o desenvolvimento do projeto foi utilizado a IDE Eclipse, esta ferramenta é gratuita e disponibilizada no próprio site do fabricante, a IDE permite criação de projetos JAVA para WEB, além de integração com servidores de aplicação como Tomcat, JBoss e Glassfish. A escolha do servidor de aplicação foi baseada na facilidade e simplicidade de configuração do mesmo, por isso a escolha foi o Apache Tomcat na versão 7.

Para criação da aplicação foi utilizado uma arquitetura client server, onde o lado *server* tem serviços que executam operações com o banco de dados, o lado client utiliza controladores em cada página, estes controladores utilizam serviços do client que se conectam com os serviços do lado do *server*, trocando objetos do tipo JSON. A arquitetura mencionada pode ser observada na figura 16.

Figura 16 – Arquitetura do sistema



Fonte: Do autor.

A criação do projeto foi realizada pela IDE, realizando também as configurações básicas de uma aplicação online. A definição da arquitetura aplicada serviu para determinar quais seriam os pontos que o protótipo conseguiria atender. Desta forma com o projeto criado, a arquitetura definida e o servidor de aplicação pronto para publica-la, pode-se iniciar a codificação.

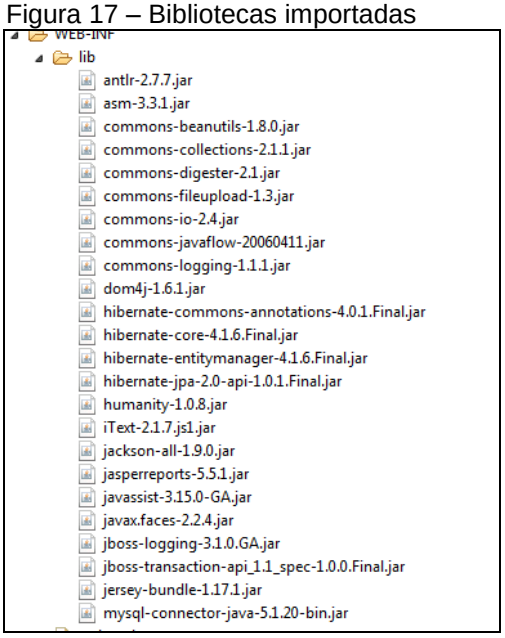
5.2.1 Desenvolvimento em JAVA

A primeira parte do desenvolvimento foi os serviços REST, provendo todos os dados necessários que a aplicação precisava, esta etapa do desenvolvimento foca apenas no trecho de código que irá rodar no lado do servidor. Para desenvolver os serviços foi necessário a utilização de algumas bibliotecas de apoio para facilitar o desenvolvimento.

Foi escolhido Hibernate, para facilitar a comunicação com o banco de dados, que é a uma implementação de JPA. JPA é um padrão JAVA que define como os frameworks irão trabalhar se relacionando com o banco, desta forma facilita manipulação de objetos, consultas e persistência. Com a utilização de JPA o banco de dados pode ser qualquer suportado por ele, mas para o desenvolvimento deste protótipo foi utilizado o MySQL.

Foram adicionados os Java ARchives (JAR) do Jersey que é o framework que implementa o padrão REST utilizado para criar os serviços, também os JAR's do Japer*Report*s para possibilitar a geração do relatório. Foram citadas as bibliotecas mais relevantes, porém existem outras que são necessárias porque são dependências, todas estão disponíveis nos *sites* dos fabricantes para *download*. O diretório "lib" é o local onde estas bibliotecas foram reunidas dentro do projeto podendo ser observado na figura 17.

Figura 17 – Bibliotecas importadas



Fonte: Do autor.

A partir deste momento a aplicação possuía os recursos necessários para o inicio da codificação. A primeira parte foi à modelagem do banco de dados, onde foram usados os recursos do JPA, sendo necessário criar e mapear os objetos que utilizam o padrão Value Object (VO). Cada um destes objetos representa uma tabela do banco de dados, assim o JPA consegue criar o banco com base nos VO's. Esta configuração foi utilizada visando uma modelagem orientada a objetos.

Na configuração do JPA, foi necessário criar um arquivo chamado "persistence.xml" e coloca-lo dentro do diretório "META-INF" do projeto. Nesse arquivo foram configuradas as classes que representam o modelo de dados do protótipo, usuário, senha do banco e os demais parâmetros de configuração para que o JPA gere o modelo do banco de dados automaticamente. O arquivo "persistence.xml" pode ser observado na figura 18.

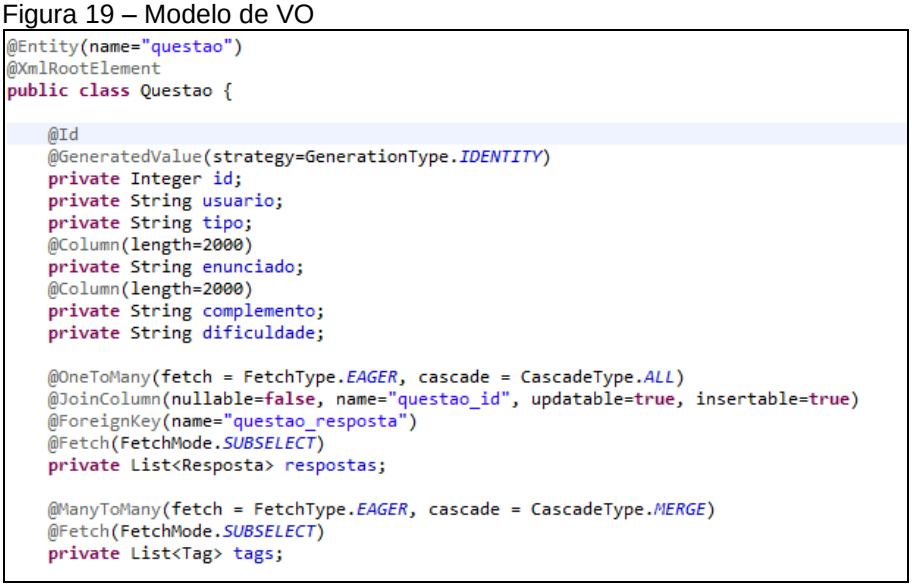
Figura 18 – Arquivo persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
    <persistence-unit name="AvaliacoesPS" transaction-type="RESOURCE_LOCAL">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>
        <class>br.avaliacoes.model.Usuario</class>
        <class>br.avaliacoes.model.Questao</class>
        <class>br.avaliacoes.model.Resposta</class>
        <class>br.avaliacoes.model.Tag</class>
        <class>br.avaliacoes.model.Avaliacao</class>
        <properties>
            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/avaliacoes"/>
            <property name="javax.persistence.jdbc.user" value="root"/>
            <property name="javax.persistence.jdbc.password" value="admin"/>
            <!--
                <property name="hibernate.hbm2ddl.auto" value="create" />
            -->
            <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
        </properties>
    </persistence-unit>
</persistence>
```

Fonte: Do autor.

A criação dos VO's foi feita com as anotações do JPA, e é baseado nessas anotações que o JPA cria o banco. Nas anotações podem-se definir propriedades de colunas, relacionamentos entre tabelas, valores padrões entre outros atributos de tabelas e chaves. Um modelo de VO anotado pode ser observado na figura 19 que é o modelo da tabela de questões, com definições de suas colunas e relacionamentos.

Figura 19 – Modelo de VO

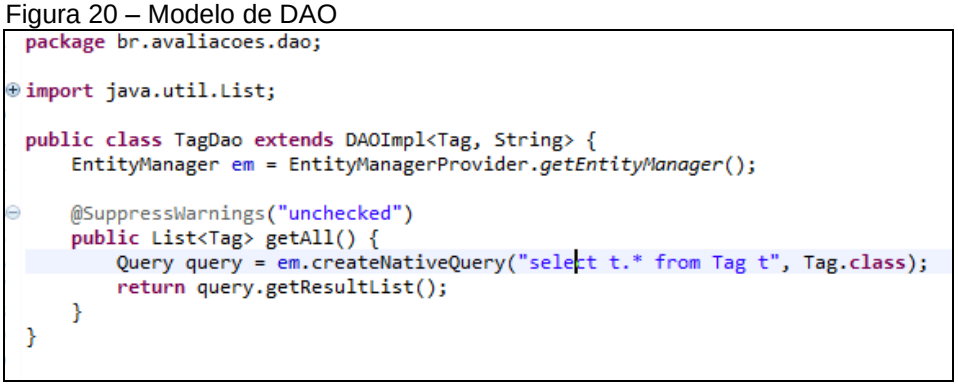


```
@Entity(name="questao")
@XmlRootElement
public class Questao {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer id;
    private String usuario;
    private String tipo;
    @Column(length=2000)
    private String enunciado;
    @Column(length=2000)
    private String complemento;
    private String dificuldade;

    @OneToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinColumn(nullable=false, name="questao_id", updatable=true, insertable=true)
    @ForeignKey(name="questao_resposta")
    @Fetch(FetchMode.SUBSELECT)
    private List<Resposta> respostas;

    @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.MERGE)
    @Fetch(FetchMode.SUBSELECT)
    private List<Tag> tags;
```

Fonte: Do autor.

Agora que o banco de dados está criado é preciso criar as classes que irão conter os métodos de operação com o banco, estas classes são os Data Access Object (DAO), estes foram criado estendendo uma classe genérica que tem a implementação de uma interface. Dessa forma cada DAO criado contêm os métodos genéricos necessários para comunicação do VO com o banco. As particularidades de acesso ao banco como consultas personalizadas foram feitas no próprio DAO do objeto. A estrutura de um DAO com métodos personalizados pode ser observada na figura 20, neste caso é a classe de comunicação do objeto Tag e existe uma consulta personalizada que retorna todas as tags chamada *getAll*.

Figura 20 – Modelo de DAO



```
package br.avaliacoes.dao;

import java.util.List;

public class TagDao extends DAOImpl<Tag, String> {
    EntityManager em = EntityManagerProvider.getEntityManager();

    @SuppressWarnings("unchecked")
    public List<Tag> getAll() {
        Query query = em.createNativeQuery("select t.* from Tag t", Tag.class);
        return query.getResultList();
    }
}
```

Fonte: Do autor.

Para cada tabela do banco de dados foi criado um VO, para cada VO criado um DAO para acesso ao banco. Ao todo sendo criados os VO's para as tabelas de avaliações, questões, respostas, *tags* e usuários.

Para utilizar toda essa estrutura criada com o banco de dados, em todos os locais que foram feitas interações com o banco era necessário criar uma conexão, utilizar ela e depois fecha-la, então para não precisar fazer isso em todos os momentos que a aplicação iria usar o banco de dados, foi criada uma classe que filtrará todas as requisições, desta forma esta classe fará o gerenciamento da conexão de banco de dados automaticamente. A classe de gerenciamento de conexões com o banco de dados pode ser observada na figura 21, ela contêm no corpo do método doFilter a lógica para abrir e fechar conexões com o banco de dados a cada requisição feita.

Figura 21 – Classe de gerenciamento de conexões com o banco de dados

```
public class EntityManagerInterceptor implements Filter {

    @Override
    public void destroy() {}

    @Override
    public void init(FilterConfig fc) throws ServletException {}

    @Override
    public void doFilter(ServletRequest req, ServletResponse res,
            FilterChain chain) throws IOException, ServletException {

            try {
                EntityManagerProvider.beginTransaction();
                chain.doFilter(req, res);
                EntityManagerProvider.commit();
            } catch (RuntimeException e) {

                if ( EntityManagerProvider.getEntityManager() != null && EntityManagerProvider.getEntityManager().isOpen())
                    EntityManagerProvider.rollback();
                throw e;

            } finally {
                EntityManagerProvider.closeEntityManager();
            }
    }
}
```

Fonte: Do autor.

Com a criação dos VO's, dos DAO's e com o "persistence.xml" configurado para criar o banco de dados completou-se a estrutura de acesso ao banco. A próxima etapa foi a utilização dessa implementação com o banco para a criação dos serviços que serão consumidos pelo lado *client* da aplicação.

5.2.1.1 Desenvolvimento dos serviços

Para criar os serviços foi utilizado à biblioteca Jersey que é uma implementação do padrão REST. O download dos JAR's já foi demonstrado no momento da configuração do projeto, então a próxima etapa é criar e configurar os serviços.

No primeiro momento foi configurado no arquivo "web.xml" da aplicação o Servlet do Jersey para que o mesmo consiga gerenciar os serviços, após foi parametrizado o local de onde o Jersey vai encontrar os serviços criados. A configuração do "web.xml" com os dois passos descritos anteriormente pode ser observado na figura 22, no parâmetro com.sun.jersey.config.property.packages foi especificado o pacote de onde estariam os serviços criados e na configuração do Servlet foi definida a URL de como estes serviços poderiam ser acessados, nesse caso */rest/**.

Figura 22 – Configurações do Jersey

```
<servlet>
  <servlet-name>Jersey REST Service</servlet-name>
  <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>com.sun.jersey.config.property.packages</param-name>
    <param-value>br.avaliacoes.rest</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>Jersey REST Service</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

Fonte: Do autor.

Depois de configurado o uso do Jersey, foi criado cada serviço exatamente no pacote onde foi mapeado no "web.xml". Os serviços possuem um caminho, *path*, por onde é acessado pelo lado *client*, também contêm o tipo de requisição que é feito para acessar ele, podendo ser POST, GET, PUT, DELETE entre outros e ainda qual é o tipo de objeto que ele vai receber e retornar. Um serviço criado pode ser observado na figura 23, este serviço possui o caminho de acesso */avaliacao* e nele existe dois serviços em GET e outro POST, ambos os serviços irão receber e retornar objetos do tipo JSON.

Figura 23 – Modelo de serviço REST

```
@Path("/avaliacao")
public class AvaliacaoResources {
    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Response save(@Context HttpServletRequest req, Avaliacao avaliacao){
        AvaliacaoDao avaliacaoDao = new AvaliacaoDao();
        avaliacao.setUsuario(SessionUtils.getSession(req).getAttribute(SessionUtils.USER_LOGIN).toString());
        avaliacao = avaliacaoDao.save(avaliacao);
        return Response.status(200).entity(avaliacao).build();
    }

    @GET
    @Path("{id}")
    @Produces(MediaType.APPLICATION_JSON)
    public Avaliacao getById(@PathParam("id") Integer id){
        AvaliacaoDao avaliacaoDao = new AvaliacaoDao();
        Avaliacao avaliacao = avaliacaoDao.getById(Avaliacao.class, id);
        return avaliacao;
    }
```

Fonte: Do autor.

Para cada rotina da aplicação foi criado um serviço que contêm métodos que salva, recupera, apaga e modifica objetos do banco de dados, ou então apenas executar procedimentos e retornar valores. Os serviços criados foram os de avaliações, *tags*, usuários e questões.

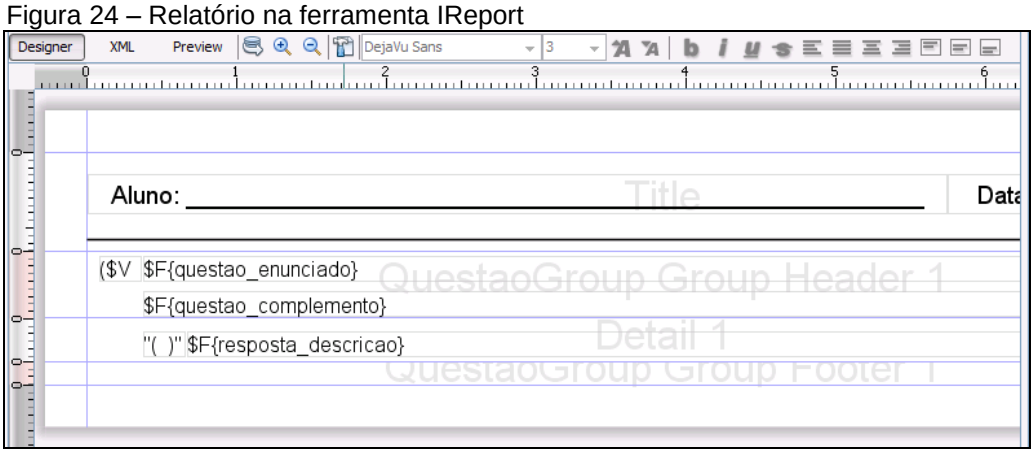
5.2.1.1 Desenvolvimento do serviço de relatório

Na criação do relatório foram usados dois recursos, um deles é o iReport, ele é uma ferramenta gratuita disponibilizada no próprio site do fabricante. Outro é um *framework* chamado JasperReport que consegue trabalhar junto com o iReport, os dois fornecem o necessário para gerar relatórios JAVA.

O iReport provê uma interface gráfica onde pode-se construir o *layout* de um relatório apenas arrastando componentes. A ferramenta foi construída na linguagem JAVA e permite que se faça programação nesta linguagem dentro do *layout* do relatório, facilitando condicionamentos e criação de componentes dinâmicos no relatório.

O *layout* do relatório foi criado usando a ferramenta iReport, já que ela disponibiliza facilmente uma gama de componentes prontos para uso. A interface do iReport pode ser observada na figura 24, o relatório disposto é o da avaliação, sendo o único que protótipo possui.

Figura 24 – Relatório na ferramenta IReport



Fonte: Do autor.

Para exibir o relatório para o usuário, foi criado um serviço, onde este chama uma classe utilitária que lê o arquivo JasperReport, passa todos os parâmetros necessários para ele e devolve o resultado para o navegador. A imagem

do serviço pode ser observada na figura 25, o serviço recebe quatro parâmetros e acessa um método utilitário que escreve o relatório para o usuário.

Figura 25 – Serviço de geração do relatório

```
@GET
@Path("/gerar")
public void gerar(@Context HttpServletRequest req, @Context HttpServletResponse res,
        @QueryParam("id") Integer id,
        @QueryParam("sortQuestao") boolean sortQuestao,
        @QueryParam("sortResposta") boolean sortResposta,
        @QueryParam("order") String order){

    String[] ordem = order.split(",");
    if (sortQuestao){
        Collections.shuffle(Arrays.asList(ordem));
    }
    order = Arrays.toString(ordem).replace("[", "").replace("]", "");
    ReportGenerator.reportAsPdf(req, res, id, sortQuestao, sortResposta, order);
}
```

Fonte: Do autor.

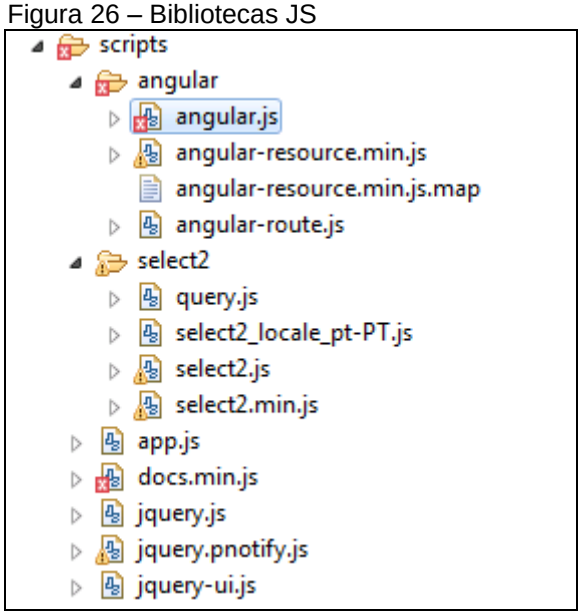
Nesta etapa concluí se o desenvolvimento JAVA, sendo que o último passo foi o desenvolvimento do relatório, desta forma o desenvolvimento no lado *server* terminou, a partir deste fase foi iniciado o desenvolvimento das páginas do sistema e a programação do lado *client* da aplicação.

5.2.2 Desenvolvimento do lado *client*

O lado *client* é o que executa no computador do usuário, diretamente no navegador, compondo a parte das telas e apresentação dos dados. Foram utilizadas as linguagens de programação JS, HTML e CSS, com auxílio de *frameworks* dessas linguagens, exceto de HTML.

O AngularJS, foi o *framework* mais utilizado de JS para o desenvolvimento *client*, também foi usado o *framework* Twitter Bootstrap para componente CSS, este possui um *layout* responsivo com componentes padrões e de agradável visualização. Foi utilizado o Select2 para fazer componentes de consultas e o Pnotify para exibir mensagens, ambos também JS. Todos os *frameworks* utilizados são encontrados nos próprios sites dos fabricantes e podem ser adquiridos gratuitamente. A figura 26 demonstra alguns dos arquivos JS baixados, na imagem alguns arquivos possuem uma marcação vermelha pois a IDE não consegui interpretar corretamente os códigos dos arquivos.

Figura 26 – Bibliotecas JS



Fonte: Do autor.

Para utilizar o AngularJS foi criado um arquivo de configuração chamado "app.js", esse arquivo contêm rotas, definição de controladores e o nome de todos os módulos que o angular vai precisar para fazer o controle da aplicação. O arquivo "app.js" pode ser observado na figura 27, em "angular.module" foram declarados todos os módulos que aplicação utilizou, e em "config" configurou-se as rotas, as páginas e o controlador das mesmas.

Figura 27 – Arquivo de configuração do AngularJS

```
'use strict';

angular.module('avaliacao', [
  'ngRoute',
  'avaliacao.questoescontroller',
  'avaliacao.minhasquestoescontroller',
  'avaliacao.minhasavaliacoescontroller',
  'avaliacao.avaliacoescontroller',
  'avaliacao.questoesservice',
  'avaliacao.avaliacaoservice',
  'avaliacao.tagservice',
  'avaliacao.geraravaliacoescontroller'
]).
config(['$routeProvider', function($routeProvider) {
    $routeProvider.when('/questoes', {templateUrl: 'app/questoes/minhas/minhas_questoes.html',
        controller:'MinhasQuestoesCtrl'});
    $routeProvider.when('/questoes/cadastro', {templateUrl: 'app/questoes/cadastro/cad_questoes.html',
        controller:'QuestaoCtrl'});
    $routeProvider.when('/questoes/cadastro/:id', {templateUrl: 'app/questoes/cadastro/cad_questoes.html',
        controller:'QuestaoCtrl'});
    $routeProvider.when('/avaliacoes', {templateUrl: 'app/avaliacoes/minhas/minhas_avaliacoes.html',
        controller:'MinhasAvaliacoesCtrl'});
    $routeProvider.when('/avaliacoes/cadastro', {templateUrl: 'app/avaliacoes/cadastro/cad_avaliacoes.html',
        controller:'AvaliacoesCtrl'});
    $routeProvider.when('/avaliacoes/cadastro/:id', {templateUrl: 'app/avaliacoes/cadastro/cad_avaliacoes.html',
        controller:'AvaliacoesCtrl'});
    $routeProvider.when('/avaliacoes/gerar/:id', {templateUrl: 'app/avaliacoes/gerar/ger_avaliacoes.html',
        controller:'GerarAvaliacoesCtrl'});
}]);
```

Fonte: Do autor.

A introdução ao desenvolvimento foi a página de acesso ao sistema. Criado um controlador para ela que é responsável por fazer o gerenciamento da dinâmica da página. Este controlador acessa um serviço que faz a conexão com serviços no lado *Server*. Uma imagem do controlador da página *login* pode ser observada na figura 28, o controlador possui um nome chamado *UserCtrl*, também consegue-se observar o método de *save* e os módulos que o controlador pode usufruir.

Figura 28 – Controlador AngulasJS

```
'use strict';

/* Controllers */
angular.module('login',['login.controller','login.services']);

angular.module('login.controller', [])
    .controller('UserCtrl', ['$scope','UserService', function($scope, UserService) {

          $scope.initialCreate = function() {
              $scope.mensagemCad = undefined;
              $scope.cadUser = null;
          };

          $scope.save = function() {
              if ($scope.formCad.$valid){
                  if ($scope.cadUser.senha != $scope.cadUser.senhaReply){
                      sendErrorMessage('A senha e a confirmação de senha devem ser iguais.');
                      return;
                  }
                  UserService.create($scope.cadUser,
                      function success(data, status) {
                      location.href = '/Avaliacoes/#/questoes';
                   },
                   function error(data) {
                       sendErrorMessage(data.data);
                       return;
                   });
               }
          };
```

Fonte: Do autor.

O controlador também tem funções como colocar valores em campos, resgatar e formatar valores vindos do servidor, controle de tabelas, listas e mensagens de erro. Ele precisa de serviços para se conectar com os serviços criados no lado *server*, estes serviços são acessados apenas para recuperar as informações do banco de dados, sendo que a manipulação dos dados recuperados e como eles vão ser demonstrados é função do controlador. O serviço de *login* pode ser observado na figura 29, este possui dois métodos de acesso, ambos POST e duas URL's definidas que servem para encontrar os serviços do lado *server*.

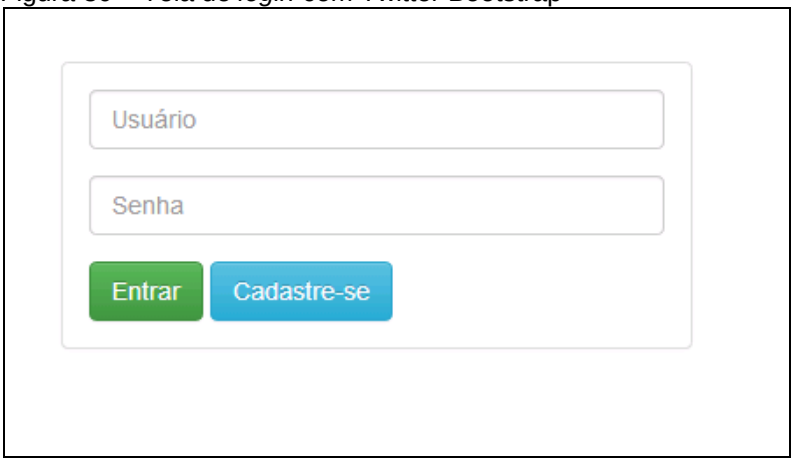
Figura 29 – Serviço AngulasJS

```
'use strict';

/* Service */

angular.module('login.services',['ngResource'])
    .factory('UserService', function ($resource) {
        return $resource('/Avaliacoes/rest/usuario/cadastro', {}, {
            create: { method: 'POST' },
            login: { method: 'POST', url:'/Avaliacoes/rest/usuario/login' }
        });
});
```

Fonte: Do autor.

A próxima etapa foi criar o *layout* da página, com o auxilio do Twitter Bootstrap foi feito uma tela onde o usuário pode entrar e se cadastrar caso ainda não tenha usuário. A linguagem usada para definir o *layout* foi apenas CSS. O esboço do *login* pode ser observado na figura 30, o painel está centralizado e disposto embaixo do nome do sistema.

Figura 30 – Tela de *login* com Twitter Bootstrap



Fonte: Do autor.

Para que não precise importar todos os *frameworks* em cada página, escrever o menu e outros trechos de códigos repetidos, foi criado uma página chamada index.html, essa página serviu como uma casca. Todo o conteúdo comum das páginas foram adicionados no "index.html", com exceção do *login* que é uma página externa e sem menu, todas as outras páginas utilizam ela para serem exibidas.

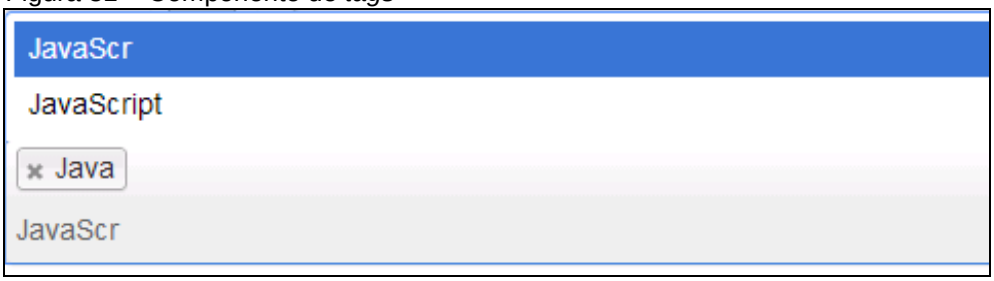
A estrutura básica de todas as outras páginas criadas foi a mesma da página de *login*, um arquivo HTML que contêm o *layout* da página, um controlador JS que utiliza o AngularJS para gerenciar as informações e um arquivo de serviços

que faz a conexão com lado *server* da aplicação. Existem outros componentes que necessitaram considerável nível de pesquisa e desenvolvimento, os quais serão descritos a seguir.

5.2.2.1 Componentes de consulta e cadastro de *tags*

Para o desenvolvimento do componente de *tags*, foi utilizado os *framework* Select2 e AngularJS. O AngularJS ficou responsável pela função de recuperar e gravar as informações no banco de dados via serviços REST. O Select2 tratou do comportamento do componente, como consulta e disponibilização dos dados. Este componente foi utilizado para fazer um sistema de marcações nas questões, ficando mais fácil de consulta-las. Uma imagem do componente criado pode ser observada na figura 31, nela o usuário está pesquisando pela palavra "JavaScr" e o componente está sugerindo para ele "JavaScript".

Figura 31 – Componente de *tags*



Fonte: Do autor.

5.2.2.2 Componentes de mensagens

Para informar o usuário sobre o que está acorrendo no sistema foi utilizado as mensagem do PNotify, este *framework* JS permiti que sejam adicionadas na página mensagens de alerta, erro ou apenas uma informação para o usuário. Na implementação foi criado uma arquivo genérico de acesso a qualquer página e depois métodos para que a página consiga mostrar as mensagens para o usuário. Uma imagem da implementação do Pnotify pode ser observada na figura 32, nela pode-se observar que existe três funções, uma para cada tipo de mensagem que é exibida para o usuário.

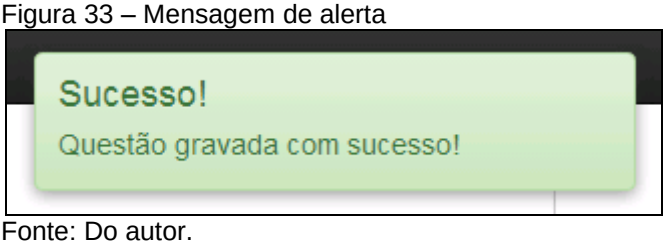
Figura 32 – Componente de mensagens

```
function sendErrorMessage(msg){
    $.pnotify({
        title: 'Erro!',
        text: msg,
        type: 'error'
    });
};

function sendSuccessMessage(msg){
    $.pnotify({
        title: 'Sucesso!',
        text: msg,
        type: 'success'
    });
};

function sendInfoMessage(msg){
    $.pnotify({
        title: 'Info!',
        text: msg,
        type: 'info'
    });
};
```

Fonte: Do autor.

Com essa implementação de código, todas as páginas podem facilmente chamar as mensagens no Pnotify, a mensagem exibida tem uma cor diferenciada, de acordo com sua gravidade. Elas permanecem um tempo para o usuário e depois são ocultadas automaticamente. O resultado gerado por estas mensagens pode ser observado na figura 33, está mensagem é exibida toda vez que o usuário concluí uma ação com êxito.

Figura 33 – Mensagem de alerta



Fonte: Do autor.

A mesma lógica de desenvolvimento demonstrada até aqui foi aplicada em todas as outras telas e funcionalidades, concluindo então a fase de implementação do protótipo, iniciando a última etapa no processo de construção do protótipo, os testes, esta fase prioriza a correção de erros gerados na fase de implementação e será detalhada a seguir.

5.3 TESTES

O protótipo foi testado apenas pelo próprio desenvolvedor, não conteve analises aprofundados nem relevantes à usabilidade e desempenho. Os testes realizados foram focados nas funcionalidades do sistema, para garantir que todos os componentes e telas tenham o comportamento esperado, desta forma não contendo erros que impeçam o usuário de utilizar o sistema com quebras de página e de fluxo de navegação.

Os testes foram feitos partindo do principio de que o usuário não sabe mexer no sistema, desta forma foi realizados cliques em objetos de ordem aleatória, tentativas de cadastros incorretos, uso de componentes indevidamente, entre outros testes do gênero. Nesta etapa foram encontrados muitos erros, principalmente no componente de *tags* e na tela de cadastrar avaliações, pois esta tela trabalha com duas tabelas e contêm JS monitorando e manipulando ambas. Os erros encontrados foram ajustados e testados novamente para garantir que não ocorreriam mais.

Outro teste realizado foi o de requisitos, nesta etapa foi considerada a aplicação como um todo, desta forma foi avaliado no sistema quais eram os requisitos que ela deveria atender e quais requisitos ela atendia. Foram criados vários casos levando em consideração os tipos de questões, a facilidade de encontrar a que se deseja e a possibilidade de construir uma avaliação, editar e armazenar a mesma. Nesta fase não foi identificado nenhum problema, isso demonstra que o levantamento de requisitos e o projeto do protótipo obtiveram sucesso.

No decorrer do desenvolvimento foram encontrados muitos erros em diversos locais do sistema, sejam problemas de origem na codificação, problema de incompatibilidade, problemas de concorrência, problemas de interação de objetos entre outros. Estes tipos de erros foram sendo resolvidos na medida em que foram sendo encontrados. Esta fase foi feita por etapas de desenvolvimento do protótipo, minimizando os testes finais.

Em geral a fase de testes não foi profundamente explorada, não sendo realizado um total mapeamento de todos os cenários possíveis de testes, eles foram feitos apenas para garantir a funcionalidade básica do sistema e eliminar os erros perceptíveis. Desta maneira finalizou-se o desenvolvimento da aplicação, tendo então protótipo um funcional para analisar os resultados.

5.4 RESULTADOS OBTIDOS

Ao longo do desenvolvimento do trabalho foram sendo cumpridos os objetivos propostos na introdução do mesmo, gradativamente cada etapa foi feita em uma ordem hierárquica para que cada fase seguinte pudesse ser realizada, por fim gerando um trabalho conforme o estudo proposto. A taxonomia de Bloom e outros estudos sobre o ensino e a avaliação serviram para demonstrar quais possíveis influências estes poderiam interferir no momento da construção do sistema, mas nessa caso nenhuma característica foi aproveitada.

Com o estudo sobre a educação, os métodos de ensino e como a prática avaliativa se encaixa nesse processo, foi possível entender e priorizar quais eram as principais características pertinentes nessa área e quais o protótipo deveria conter.

A pesquisa realizada com os professores demonstrou as principais rotinas que os mesmo utilizam na prática das avaliações, sendo então possível compreender as necessidades deles. Desta forma o sistema foi projetado para que realmente ajudasse a otimizar o tempo e aumentar os recursos disponíveis na elaboração de avaliações, levando em consideração que todas as questões são utilizadas e devem ter no sistema.

O desenvolvimento do protótipo foi realizado com o *framework* AngularJS, este colaborou em grande escala para a construção da estrutura feita do lado *client*, foram também utilizados outros recursos de apoio, porém todas as outras tecnologias utilizadas foram apenas complementos para facilitar o desenvolvimento.

O conceito usado para o desenvolvimento do lado *server* foi baseado todo no conceito REST. Focando apenas em serviços e não guardando estado no servidor, desta forma essa implementação fica totalmente desacoplada e sem dependência com a parte desenvolvida no *client*.

O protótipo final possibilitou aos professores cadastrarem suas questões e avaliações, desta forma criou-se um ambiente de gerenciamento destas informações para os professores. O portal criado permite aos usuários reaproveitarem questões de outras pessoas e também compartilhem as suas, além de consultarem as suas avaliações criadas entre semestres e anos.

6 CONCLUSÃO

A avaliação dentro da prática de ensino, apesar de antiga, ainda é uma boa forma de extrair o conhecimento que o ensinando absorveu. Quando uma avaliação é elaborada por um professor, se este tem a noção de como fazê-la para atingir seus objetivos, ela pode sim refletir o resultado absorvido pelo aluno.

As avaliações feitas em sala de aula podem ser aplicadas de várias maneiras e ter diferentes tipos de questões, existindo então uma diversidade de possíveis testes. Para criar uma avaliação diferente do convencional é investido um tempo maior em sua elaboração, só que ela acaba sendo utilizada poucas vezes, pois perde sua validade assim que publicada. Este conteúdo talvez não tenha mais valia para quem já fez uma vez, porém ela pode ser compartilhada para o crescimento de outras avaliações do mesmo tema que poderão ser criadas.

A minoria dos professores usam recursos que ajudam na elaboração das avaliações, outros ainda contam com técnicas não tão atualizadas utilizando questões repetidas, perdendo muita eficiência e eficácia na aplicação da mesma. Esta falta de recursos pode ser por causa da falta de conhecimento dos professores em informática, da aversão a novas tecnologias ou porque não conhecem ferramentas disponíveis.

Conforme a pesquisa realizada durante o trabalho, os professores não se importam em utilizar questões de outras pessoas, além de aproveitarem questões entre semestres. A maioria já tem um tempo considerável trabalhando como professor e estes realmente têm interesse de conhecer uma ferramenta que os ajudem com esta função.

O protótipo desenvolvido disponibilizou recursos que ajudam os professores a construírem suas avaliações. A possibilidade de criar elas apenas consultando questões que já estão prontas agilizando o processo, além de disponibilizar outras visões, pois as questões podem ter sido elaboradas por pessoas diferentes. O sistema guarda todas as informações das avaliações aplicadas, desta forma torna mais viável a sua reutilização, visto que é só resgatar uma já aplicada, fazer algumas adaptações e utilizar ela novamente. Com os recursos de embaralhar questões e alternativas destas a reaplicação ficou ainda mais viável.

O desenvolvimento ajudará os professores a elaborar suas avaliações com mais facilidade, porém existe a necessidade de pesquisar quais seriam outras dificuldades deles e também quais funcionalidades deixariam o sistema mais interessante para os usuários. Como sugestão de trabalhos futuros para implementação de recursos no software pode citar:

a) Possibilitar um mecanismo de pontuação, onde o usuário pode dar uma nota para as questões criada.
b) Possibilitar a aplicação da avaliação criada de forma *online*.
c) Disponibilizar recursos para copiar questões ou avaliações feitas por outras pessoas.
d) Criar a opção gerar prova gabarito.
e) Pesquisar outras funcionalidades não previstas neste trabalho para que o sistema se torne mais eficiente.

REFERÊNCIAS

BLOOM, B. S. **Taxonomy of educational objectives:** the classification of educational goals: handbook I: cognitive domain. New York: Longman, 1956.

BLOOM, B., et al. **Taxonomia de objetivos educacionais:** domínio cognitivo. Porto Alegre: Globo, 1974.

BORDENAVE, Juan Díaz; PEREIRA. Adair Martins. **Estratégias de Ensino-Aprendizagem**. Petrópolis: Vozes, 2006. 312 p.

BUGMANN, Paulo Alberto. **Ferramenta web para modelagem lógica em projetos de bancos de dados relacionais.** 2012. 84 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade Regional de Blumenau, Blumenau, 2012.

BURKE, Bill. **RESTful Java with JAX-RS**. Sebastopol: O'Reilly Media, 2010. 312 p.

CADENHEAD, Rogers. **Sams Teach Yourself Java™ in 24 Hours**. 6. United States of America: Sams, 2013. 429 p.

CAIADO. Elen Campos. **O professor e a educação de valores**. fev. 2012, Disponível em: <http://educador.brasilescola.com/orientacoes/o-professor-educacao-valores.htm>. Acessado em 04 mai. 2013.

DUCKETT, Jon. **Beginning HTML, XHTML, CSS, and JavaScript**. Indianapolis: Wiley Publishing, 2010. 866 p.

EVANS, Benjamin J.; VERBURG. Martijn. **The Well-Grounded Java Developer**. Nova York: MANNING, 2013. 462 p.

FERRAZ, Ana Paula do Carmo Marcheti; BEHOLT, Renato Vairo. **Taxonomia de Bloom:** Revisão teórica e apresentação das adequações do instrumento para definição de objetivos instrucionais. 2010. Disponível em: <http://issuu.com/luiz/docs/name3f9174>. Acesso em: 25 abr. 2014.

FREITAS. Eduardo. **A situação do professor brasileiro**. dez. 2011, Disponível em: <http://educador.brasilescola.com/trabalho-docente/a-situacao-professor-brasileiro.htm>. Acessado em 08 nov. 2012.

GOLDSTEIN, Alexis; LAZARIS, Louis; WEYL, Estelle. **HTML5 & CSS3 for the Real World.** Collingwood: Sitepoint Pty., 2011. 377 p.

GONÇALVES, António. **Beginning Java EE 7**. United States of America: Apress, 2013. 608 p.

GRONLUND, Norman Edward. **Constructing achievement tests**. Englewood Cliffs, N.J: Prentice-Hall, 1968. 118 p.

GREEN, Brad; SESHADRI, Shyam. **AngularJS.** Sebastopol,: O'Reilly Media, 2013. 196 p.

GUPTA, Arun. **Java EE 7 Essentials**. Sebastopol: O'Reilly Media, 2013. 362 p.

HENICK, Ben. **HTML & CSS**: The Good Parts. Sebastopol: O'Reilly Media, Inc, 2010. 352 p.

HEUSER, Fernanda Müller. **Avaliação: sentença ou diagnóstico.** 2008. 39 f. TCC (Graduação) - Curso de Pedagogia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2008.

HUGHES, David. **Fundamentals of Computer Science Using Java**. Mississauga: Jones and Bartlett, 2002. 528 p.

KLITZKE, ElisÂngela Cristina Lombardi. **Protótipo de um sistema para acompanhamento de trabalhos de conclusão de curso.** 2009. 56 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade Regional de Blumenau, Blumenau, 2009.

KOZLOWSKI, Pawel; DARWIN, Peter Bacon. **Mastering Web Application Development with AngularJS:** Build single-page web applications using the power of AngularJS. Birmingham: Packt Publishing, 2013. 372 p.

LINDEMAN, Richard H.. **Medidas Educacionais.** 6. ed. Rio de Janeiro: Globo, 1987. 176 p.

LOURENÇO, Prefácio. **Teste e medidas na Educação**. Rio de Janeiro: Fundação Getulio Vargas, 1970. 115 p.

LUCKESI, Cipriano Calos. **Avaliação da aprendizagem escolar**. 20. ed. São Paulo: Cortez, 2001. 180 p.

LUCKOW, Décio Heinzelmann; MELO. Alexandre Altair de. **Programação Java para a Web**. São Paulo: Rubens Prates, 2010. 71 p.

MACCAW, Alex. **JavaScript Web Applications.** Sebastopol: O'Reilly Media, 2011. 280 p.

MANDELLI. Mariana. **Proporção de professores com nível superior cresce 7,6% entre 2010 e 2011**. abr. 2012, Disponível em: <http://www.todospelaeducacao.org.br/comunicacao-e-midia/noticias/22421/proporcao-de-professores-com-nivel-superior-cresce-76-entre-2010-e-2011/> Acessado em 04 mai. 2013.

MEDEIROS, Ethel Bauzer. **Provas Objetivas, Discursivas, Orais e Práticas**: Técnicas de Construção. 7. Rio de Janeiro: Fundação Getulio Vargas, 1983. 178 p.

MELCHIOR, Maria Celina. **Avaliação pedagógica**: função e necessidade. Porto Alegre: Mercado aberto Ltda, 2002. 150 p.

MINAS GERAIS. **Guia de Elaboração e Revisão de Questões e Itens de Múltipla Escolha**, 2011. 34 p.

MORALES, Pedro. Avaliação escolar: o que é, como se faz. São Paulo: Loyola, 2003.

MORETTO, Vasco Pedro. (1942). **Prova**: um momento privilegiado de estudo não um acerto de contas. 6. Rio de Janeiro: DP&A, 2005. 150 p.

NEWMARCH, Jan; A RESTful Approach: Clean UPnP without SOAP: Using Java in Service-Oriented Architectures. *Consumer Communications and Networking Conference,* 2005. CCNC. 2005 Second IEEE: 134-138, Jan. 2005.

OLIVEIRA, Jackson dos Santos. **Reduzindo acoplamento e aumentando a escalabilidade de sistemas corporativos através de rest.** 2011. 61 f. TCC (Graduação) - Curso de Sistemas de Informação, Universidade Feevale, Novo Hamburgo, 2011.

PERRENOUD, Philippe. **Avaliação**: da excelência à regulação das aprendizagens. Porto Alegre: ARTMED, 1999. 183 p.

POHL, Ira; MCDOWELL. Charlie. **Java by Dissection**. 2. Santa Cruz: ISBN, 2006. 454 p.

RABELO, Edmar Henrique. **Avaliação**: Novos Tempos Novas Práticas. Petrópolis: Editora Vozes, 1998. 144 p.

RUEBBELKE, Lukas. **Awesome AngularJS Features.** Disponível em: <http://net.tutsplus.com/tutorials/javascript-ajax/5-awesome-angularjs-features/>. Acesso em: 07 nov. 2013.

SCHAFER, Steven. **HTML, XHTML, AND CSS BIBLE**. 5. Indianapolis: Wiley Publishing, 2010. 759 p.

SILVA, Pereira Bruno Luiz. REST vs WS-*: Uma Visão Pragmática. Java Magazine, São Paulo, Edição 54, pág. 38 à 47, 2008.

SILVEIRA, Rodrigo. **Learn HTML5 by Creating Fun Games.** Birmingham: Packt Publishing, 2013. 374 p.

VIANNA, Heraldo Marelim. **Testes em educação**. 2. São Paulo: IBRASA, 1976. 220 p.

WILLARD, Wendy. **HTML**: A Beginner's Guide. 4. New York: MC Graw Hill, 2009. 539 p.

WRIGHT, Tim. **Learning JavaScript:** a hands-on guide to the fundamentals of modern JavaScript. Michigan: Addison-wesley, 2013. 350 p.

APÊNDICE(S)

APENDICE A - Questionário aplicado

Questionário sobre avaliações

Este questionário será usado no TCC de ciência da computação, sobre um desenvolvimento de um aplicativo que irá ajudar os professores a montar sua avaliações.

* Required

Quantos anos de experiência você tem como professor? *

De 1 a 3

Quantas avaliações escritas você costuma aplicar por semestre? *

2

Quais são os tipos de avaliações que você costuma aplicar? *

- [x] Escritas
- [x] Práticas
- [ ] Online

Quais são os tipos de questões que você costuma utilizar nas avaliações? *

- [x] Lacuna
- [x] Verdadeiro ou falso
- [ ] Associação
- [ ] Múltipla escolha
- [ ] Ordenação
- [ ] Dissertativa

Você reaproveita questões de provas entre um semestre e outro? *

Sim

Você usa ou usaria questões formuladas por outras pessoas para montar suas avaliações? *

Sim

Você utiliza algum programa de computador para ajudar a montar suas avaliações (sem ser o Microsoft Word)? *

Não

Você tem vontade de conhecer algum novo programa de computador para ajudar a montar sua avaliações? *

Sim

APENDICE B – Artigo

Aplicativo para elaboração e compartilhamento de questões avaliativas

Rafael Soratto Ortolan, Fabricio Giordani

Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC)
Av. Universitária, 1105 – Caixa Postal 3167 – CEP 88806-000 – Santa Catarina – SC – Brasil

rafasoratto@hotmail.com, fgiordani@gmail.com

Abstract. *Teachers need a way to assess the content that students learned about a particular subject. Based on this, the work demonstrates the development of a web application prototype, which allows you to help teachers develop their assessments cooperatively and more easily, highlighting how a system can help them in this area. The development of the prototype was done with the Representational State Transfer (REST) architecture for the server side and the AngularJS framework for developing the client side, and as the two main programming languages JAVA and JavaScript.*

Resumo. *Os professores necessitam de uma forma de avaliar o conteúdo que o aluno aprendeu sobre um determinado assunto. Baseado nisso o trabalho demonstra o desenvolvimento de um protótipo de um aplicativo web, o qual permite ajudar os professores elaborarem suas avaliações de forma cooperativa e com mais facilidade, enfatizando como um sistema pode ajudá-los nesta área. O desenvolvimento do protótipo foi realizado com a arquitetura Representational State Transfer (REST) para o lado server e com o framework AngularJS para o desenvolvimento do lado client, tendo como as duas principais linguagens de programação o JAVA e o JavaScript.*

1. INTRODUÇÃO

Para que um professor consiga elaborar uma prova para seus alunos, ele terá que seguir várias etapas além de ter um bom domínio do conteúdo que deseja avaliar deles. Para conseguir questões, os professores pegam-nas prontas da internet, outras já colocadas em avaliações anteriores ou criam questões a partir de sua criatividade e conhecimento. Não bastando a isso, tem que montar a ordem das questões e geralmente fazer mais de um tipo de avaliação para dificultar fraudes durante a sua aplicação (MORETTO, 2005; PERRENOUD, 1999).
O trabalho consiste em desenvolver um protótipo de aplicação que permita aos professores montar suas avaliações de uma única forma, além de possibilitar o compartilhamento de questões entre os mesmos, com o principal objetivo de ajudar a reaproveitar avaliações e questões já aplicadas.

2. DESENVOLVIMENTO

O trabalho demonstra a metodologia de ensino pesquisada, enfatizando sobre a utilização das avaliações no processo de aprendizagem, assim como a estrutura das avaliações, sua possibilidade de aplicações, abordando também os tipos de questões que podem ser utilizadas. As tecnologias utilizadas para o desenvolvimento do protótipo

2.1. ABORDAGEM DA AVALIAÇÃO NO PROCESSO DE APRENDIZAGEM

Na sociedade a parte mais visível desta aplicação é no ensino médio, neste caso o objetivo é ensinar como resolver provas, e não o conteúdo em si, isso acontece por uma força maior da sociedade, a avaliação aplica-se como forma de nivelamento em todos os tipos de conceitos,

deste a prática com crianças em casa, até a obtenção de um emprego em concursos públicos, frisando então a importância de terem avaliações bem feitas nos ambientes de ensino (LUCKESI, 2001).
O significado de uma avaliação para um aluno, baseia-se na oportunidade de conhecer os resultados do esforço empenhado nos estudos, não só isso, mas também ajuda em adquirir satisfação ao refletir sua capacidade de aprendizado. As atividades avaliativas contribuem para o desenvolvimento social, intelectual e moral do avaliado. Essas características vão ser influenciadas dependendo do seu sucesso ou fracasso (MELCHIOR, 2002).
Ainda de acordo com Melchior (2002), o professor tem como proveito na aplicação das avaliações, uma análise reflexiva, conseguindo mensurar a eficiência de seu desempenho como lecionando. É possível ainda construir métodos e novas formas de avaliar que se adaptem ao ritmo de desenvolvimento de cada aluno.

2.1.1. FUNDAMENTAÇÕES DA ELABORAÇÃO DE AVALIAÇÕES

Um professor introduzido no ambiente pedagógico, tem como objetivo principal ensinar. Esse objetivo desencadeia uma série de metodologias para atingir esta meta, além de competência e grande conhecimento sobre o conteúdo que ele vai ensinar, é necessário ter técnicas e seguir procedimentos de como transmitir esse conhecimento. Para conseguir elaborar todo um contexto levando em consideração do inicio, o aluno geralmente não faz ideia do que vai aprender nem para que serve, até o final onde ele deve ter conseguido passar ao aprendiz uma síntese do seu ensinamento (LUCKESI, 2001).
A avaliação faz parte do processo de aprendizagem, não só para cobrar, mas sim para acrescentar e contribuir. O estudo de uma elaboração de testes é complexo e leva em consideração muitos fatores. Vários estudiosos já tentaram levar a forma de avaliar em sua excelência. As representações da taxonomia de Bloom ajudam um professor a decidir quais são as técnicas de elaboração de avaliações que ele pode usar para atingir seus objetivos (VIANNA, 1976).

2.1.2. FORMAS DE APLICAÇÃO

2.1.2.1 Testes orais

Os testes orais são aqueles que o professor pergunta ao aluno, e o mesmo responde oralmente, a grande importância desse é saber o que realmente o aluno pensa, pois ele irá responder exatamente do jeito que a resposta está montada na cabeça dele, exibindo bem a sua opinião e entendimento do assunto abordado. Um ponto negativo é que a demanda de tempo em sua aplicação é grande, levando em consideração que geralmente são muitos alunos para apenas um professor, em determinadas condições ele não consegue ser aplicado com eficiência. Desta forma os testes orais foram perdendo espaço, até chegar o momento de serem raramente ou nem aplicados mais no ensino, perdendo espaço para os testes objetivos e práticos (MEDEIROS, 1983; MELCHIOR, 2002).

2.1.2.2 Testes práticos

Testes práticos são os que colocam o aprendiz em situação de tarefa real, simulando realmente alguma atividade que ele poderá exercer na sua vida, como andar de bicicleta, nadar, instalar um cabo de rede, entre outros. Geralmente esses testes são utilizados em aulas práticas ou laboratoriais, com equipes e grupos ajudando a interação entre pessoas do cotidiano (MELCHIOR, 2002).

Os testes práticos para atingirem eficácia devem fornecer ao aluno oportunidade de mostrar sua capacidade, testar suas técnicas de atividades práticas, estas serão avaliadas na integra pelo professor em forma de relatórios (BORDENAVE, 2006; MELCHIOR, 2002).
A avaliação dos testes práticos tem como base algumas características, com a quantidade do trabalho feito, analisar o esforço que o aluno demandou executando uma determinada tarefa, a qualidade do trabalho realizado, a cooperação entre os integrantes do grupo caso exista, a assiduidade, se o aluno é pontual, se realmente interage e acompanha o objetivo da aula e a atitude, levando em consideração a seriedade e importância que o aluno esta dando a atividade (BORDENAVE, 2006; MELCHIOR, 2002).

2.1.2.3 Testes Objetivos

Os testes objetivos são compostos por questões que possuem alternativas de respostas. Estes testes servem para analisar a extensão do conhecimento do aluno. Existem dois grupos de questões que entram nesse tipo de teste, as que exigem relacionamento da resposta, neste caso incluem-se questões de lacuna ou complemento, que exige do aluno a associação. O outro grupo incluem questões que exigem o reconhecimento do aluno, nestas as respostas estão dispostas para o examinando, basta ele identificar qual é a correta, como questões de verdadeiro ou falso, questões de combinação, ordenação e múltipla escolha. A escolha de quais questões usar vai depender de inúmeros fatores, e quem determinará será o professor, baseando-se nos objetivos em que deseja atingir (MEDEIROS, 1983; MELCHIOR, 2002).

2.1.2.4 Testes dissertativos

Este teste é constituído por questões em que o avaliado elabora sua própria resposta, tendo que esquematizar, descrever, explicar, comparar e apresentar argumentos, as questão postas nessa prova devem ser elaboradas com enunciados claros, enfatizando realmente o que o aluno tem que dizer na resposta. As questões deste teste podem ser classificadas de acordo com o tamanho de sua resposta, longas, médias ou curtas, esse fator é importante, pois muda o tempo de resolução da questão. Um teste só é considerado dissertativo quando todas as questões incluídas são dissertativas (MELCHIOR, 2002).
Ainda de acordo com Melchior (2002), as vantagens deste tipo de teste são atraentes, como, a facilidade de elaboração da questão devido que não há necessidade de disponibilizar as respostas na prova, a grande redução do acerto casual, a possibilidade de extrair respostas elaboradas aumentando a eficiência do teste. Como desvantagens, exige maior tempo para correção, possibilita margens de interpretações diferentes e considerações do professor na hora da correção.

2.1.3 TIPOS DE QUESTÕES PARA CADA AVALIAÇÃO

As questões objetivas levam em consideração como principal característica, fornecer opções de resposta ao avaliado. Entre os mais comuns, estão relacionados as de lacuna ou complemento, verdadeiro ou falso, associação, ordenação e múltipla escolha. Estes tipos de questões são geralmente usados desde a escola primária (MEDEIROS, 1983).
As questões dissertativas são consideradas fáceis de serem elaboradas em relação as objetivas, as dissertativas não requerem um alto nível técnico para sua elaboração, essa facilidade pode se tornar aparente caso a questão não esteja bem formulada, dando subjetividade a quem for responder. Praticamente, o único fator que o professor tem que levar em consideração quando for elaborar um questão dissertativa, é a simplicidade do enunciado, deixando muito claro o que ele quer, como o item é de resposta aberta, o enunciado deve limitar as possibilidade de repostas indicando até quando a resolução deve abranger (MEDEIROS, 1983).

As questões dissertativas geralmente dão muito trabalho em sua correção, as repostas podem ser muito variadas, e cada resposta tem que ser analisada individualmente, podendo então está estar totalmente correta, parcialmente correta, ou incorreta. Ela pode ser usada para extrair do aluno conhecimentos como, explicar, expor opinião, analisar e fazer relações. Outro fator importante é a previsão de tempo por item dissertativo, visto que ele é muito mais demorado para de ser resolvido pelo aluno (VIANNA, 1976).

2.1.4 REFLEXÕES DOS RESULTADOS DA AVALIAÇÃO

Depois de um demasiado trabalho na elaboração e aplicação da prova, chegou o momento de analisar os resultados obtidos por ela. Essa parte do período da avaliação é considerada a prova real para o professor e para os alunos, é onde todos vão saber qual foi sua eficiência perante aos seus objetivos. O que ajuda a garantir a qualidade de resultado positivos, é a elaboração de um bom teste, e também a prática de uma boa técnica de correção (VIANNA, 1976).
É aconselhável que cada prova seja realmente um instrumento de ensino, para que o professor consiga destacar e extrair de cada prova as dificuldades encontradas pelo aluno, assim pode-se elaborar uma medida para reforçar ou realçar os conteúdos que ele acredita serem indispensáveis, fazendo uma revisão com a turma ou individualmente de pontos que ele julga importantes. A nota acaba sendo a única visão de um teste, só que sozinha a ela não representa nada, porém a interpretação da avaliação vai decidir o que o professor deixou a desejar e onde ele deve se impor para ajustar e nivelar o aprendizado (LINDEMANN, 1987; MEDEIROS, 1983).

2. 2 TECNOLOGIAS PARA O DESENVOLVIMENTO

Para o desenvolvimento de uma aplicação *web*, é necessário avaliar e integrar as ferramentas que serão utilizadas para a produção do sistema. Existem muitas linguagens de programação, *Application Programming Interface* (API) e *frameworks*, alguns voltadas para a *web* outros não. A escolha das tecnologias é o primeiro passo na construção de uma aplicação seguido da definição de sua arquitetura. Dependendo das escolhas, podem surgir dificuldades ou facilidades em programar recursos no sistema, então a grande questão é analisar as ferramentas que serão utilizadas para que o aplicativo atinja o seu objetivo (CADENHEAD, 2013, tradução nossa).

2.2.1 CLIENT SIDE

As linguagens de programação usadas no lado do *client* são normalmente interpretadas pelos *web browsers*, elas tem como principal vantagem a velocidade de execução, pois rodam diretamente no *client*, evitando um pedido para o servidor. As linguagens geralmente usadas em aplicações *web* são HTML, *Cascading Style Sheets* (CSS) e *JavaScript* (JS) (DUCKETT, 2010, tradução nossa).

2.2.1.2 AngularJS

AngularJS é um framework JS mantido pela Google e seu principal objetivo é auxiliar na criação de páginas web. Ele foi construído sobre o principio de programação declarativa, isso quer dizer que sua programação declara o que ela faz, não declarando então como seus procedimentos irão funcionar. A linguagem declarativa é muito eficaz na criação de interfaces que irão interagir com o usuário, sendo que, esse é o propósito do AngularJS (GREEN; RUEBBELKE, 2012; SESHADRI, 2013, tradução nossa).

A arquitetura *Model View Controller* (MVC) é um padrão para estruturação de projetos *web*, também como *frameworks*. Este padrão é separado por três camadas *Model*, sendo o modelo, essa camada é a que faz referencia a estrutura do banco de dado, a *View*, ou seja, visão, nesta camada é apresentada os componentes que são visíveis pelos usuários, e *controller*, controlador, o qual é a camada que controla toda a lógica de negócios da aplicação, também fazendo a ligação entre o modelo e a visão (DUCKETT, 2010, tradução nossa).
A arquitetura do AngularJS é baseada no padrão MVC, porém esta pode ter diversas interpretações em relação a autores e desenvolvedores diferentes, no caso do angular sua estrutura se aproxima de algo como *Model View Presenter* (MVP), muito parecido com o MVC, porém o *Presenter* faz o papel do controlador do MVC com algumas distinções, porém está camada faz a ligação do modelo com visão. Esse padrão proporciona vantagens como, injeção de dependência, inversão de controle, desacoplamento, entre outros (DARWIN ; KOZLOWSKI, 2013, tradução nossa).

2.2.2 SERVER SIDE

O outro lado de uma aplicação *we*b é o do servidor, sendo responsável por fazer o processamento pesado e acesso a banco de dados da aplicação. A vantagem de utilizar o servidor de aplicação é abstrair o usuário de quais softwares estão sendo utilizados para processar o conteúdo. Outra vantagem é que considerando a necessidade de alto poder de processamento concentrada no servidor, a aplicação fica independente da configuração da máquina do usuário (BURKE, 2010, tradução nossa).
Existem conceitos e várias linguagens de programação do lado do servidor, porém nos próximos capítulos iremos abordar a linguagem de programação Java e um conceito de desenvolvimento chamado REST, os dois juntos irão fornecer serviços que integrarão com o lado do *client*.

2.2.2.1 REST

REST é um estilo arquitetural baseado em como a *web* funciona. Aplicado aos serviços, ele tenta traduzir os verbos da *web* em seu sentido literal. Para criar um serviço *web* REST, precisa-se conhecer o *Hypertext Transfer Protocol* (HTTP) e *Uniform Resource Identifier* (URI) e observar alguns princípios de design (BURKE, 2010; GONÇALVES, 2013, tradução nossa).
A arquitetura REST rapidamente se tornou popular porque ela conta com um robusto protocolo de transporte HTTP. Serviços *web* REST reduzem o acoplamento entre a parte do *client* e do server, tornando-a muito mais fácil de desenvolver uma interface REST ao longo do tempo, sem precisar alterar os clientes existentes quando há necessidade de fazer alguma manutenção. Além disso, os serviços são fáceis de construir, pois não necessitam de nenhum kit de ferramentas *Web Services Description Language* (WSDL) (BURKE, 2010; GUPTA, 2013, tradução nossa).
Podemos definir que REST é uma arquitetura baseado em serviços, quando um serviço é criado ele pode ser consumido por alguém. Esse serviço fica esperando ser requisitado, as requisições podem ser *get*, *post*, *delete*, *put*, entre outros. Dependendo de como os serviços são criado eles podem retornar vários formatos, como por exemplo, XML ou JavaScript Object Notation (JSON), além disso no cabeçalho de retorno sempre há um *status* indicando como a requisição foi tratada (NEWMARCH, 2010, tradução nossa).

2.3 O PROTÓTIPO

A metodologia se divide em quatro etapas bem definidas, a primeira delas é o levantamento de requisitos, o qual foi definido quais funcionalidades o sistema deveria ter para atender seus

objetivos. A segunda é o desenvolvimento do protótipo baseado nos critérios definidos no levantamento de requisitos, a terceira é uma fase testes funcionais e a última é a análise dos resultados gerados.

2.3.1 Levantamentos dos requisitos

Nos requisitos do protótipo foi definido que ele deve possuir uma tela de cadastro de questões, onde a mesma deve possibilitar gravar todos os tipos de questões mencionados neste trabalho. Uma tela de elaboração de avaliações, contendo um mecanismo de pesquisa que facilite encontrar as questões desejadas e uma tela para organizar a estrutura das avaliações e depois gera-las para impressão.
O fluxo de interação é de que os usuários possam entrar na aplicação através de um *login*, desta forma as questões ficam vinculadas no usuário. Na tela de questões é possível visualizar apenas as que foram cadastradas pelo próprio usuário, também podendo editar e excluir as mesmas. Na tela de avaliações as funções possíveis são localizar as avaliações geradas e também edita-las, excluí-las ou gera-las para impressão. Pode-se observar um possível fluxo de navegação no sistema (Figura 1).

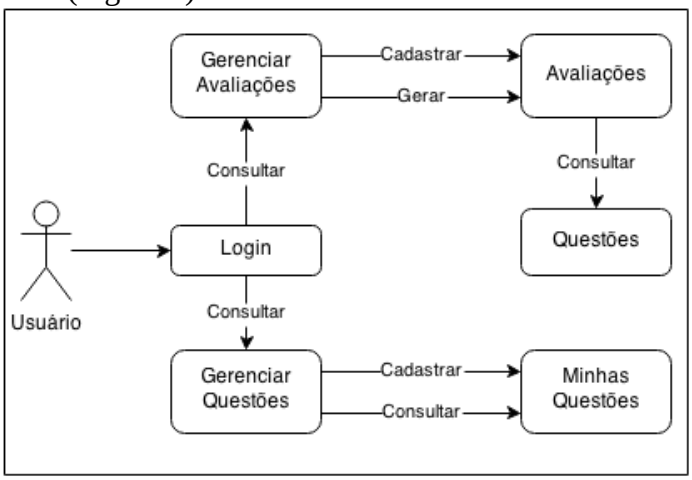


Figura 1. Exemplo de fluxo do sistema

2.3.2 Arquitetura do projeto

Para o desenvolvimento do projeto foi utilizado a IDE Eclipse, esta ferramenta é gratuita e disponibilizada no próprio site do fabricante, a IDE permite criação de projetos JAVA para WEB, além de integração com servidores de aplicação como Tomcat, JBoss e Glassfish. A escolha do servidor de aplicação foi baseada na facilidade e simplicidade de configuração do mesmo, por isso a escolha foi o Apache Tomcat na versão 7.
Para criação da aplicação foi utilizado uma arquitetura client server, onde o lado *server* tem serviços que executam operações com o banco de dados, o lado client utiliza controladores em cada página, estes controladores utilizam serviços do client que se conectam com os serviços do lado do *server*, trocando objetos do tipo JSON (Figura 2).

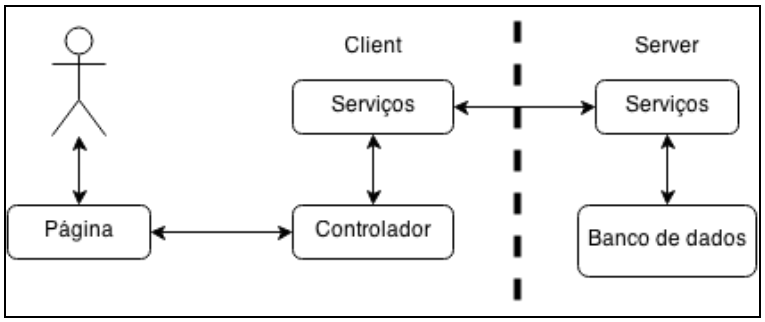


Figura 2. Arquitetura do sistema

2.3.3 Client Side

A primeira parte do desenvolvimento foi os serviços REST, provendo todos os dados necessários que a aplicação precisava, esta etapa do desenvolvimento foca apenas no trecho de código que irá rodar no lado do servidor. Para desenvolver os serviços foi necessário a utilização de algumas bibliotecas de apoio para facilitar o desenvolvimento.
Foi escolhido Hibernate, para facilitar a comunicação com o banco de dados, que é a uma implementação de JPA. JPA é um padrão JAVA que define como os frameworks irão trabalhar se relacionando com o banco, desta forma facilita manipulação de objetos, consultas e persistência. Com a utilização de JPA o banco de dados pode ser qualquer suportado por ele, mas para o desenvolvimento deste protótipo foi utilizado o MySQL.
A partir deste momento a aplicação possuía os recursos necessários para o inicio da codificação. A primeira parte foi à modelagem do banco de dados, onde foram usados os recursos do JPA, sendo necessário criar e mapear os objetos que utilizam o padrão Value Object (VO). Cada um destes objetos representa uma tabela do banco de dados, assim o JPA consegue criar o banco com base nos VO's.
Para criação dos serviços REST no primeiro momento foi configurado no arquivo "web.xml" da aplicação o Servlet do Jersey para que o mesmo consiga gerenciar os serviços, após foi parametrizado o local de onde o Jersey vai encontrar os serviços criados. A configuração do "web.xml" com os dois passos descritos anteriormente (Fgirau 3), no parâmetro com.sun.jersey.config.property.packages foi especificado o pacote de onde estariam os serviços criados e na configuração do Servlet foi definida a URL de como estes serviços poderiam ser acessados, nesse caso */rest/**.

```
<servlet>
  <servlet-name>Jersey REST Service</servlet-name>
  <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>com.sun.jersey.config.property.packages</param-name>
    <param-value>br.avaliacoes.rest</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>Jersey REST Service</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

Figura 3. Configurações do Jersey

Depois de configurado o uso do Jersey, foi criado cada serviço exatamente no pacote onde foi mapeado no "web.xml". Os serviços possuem um caminho, *path*, por onde é acessado pelo lado *client*, também contêm o tipo de requisição que é feito para acessar ele, podendo ser POST, GET, PUT, DELETE entre outros e ainda qual é o tipo de objeto que ele vai receber e retornar.

2.3.4 Client Side

O lado *client* é o que executa no computador do usuário, diretamente no navegador, compondo a parte das telas e apresentação dos dados. Foram utilizadas as linguagens de programação JS, HTML e CSS, com auxílio de *frameworks* dessas linguagens, exceto de HTML.
O AngularJS, foi o *framework* mais utilizado de JS para o desenvolvimento *client*, também foi usado o *framework* Twitter Bootstrap para componente CSS, este possui um *layout* responsivo com componentes padrões e de agradável visualização. Foi utilizado o Select2 para fazer componentes de consultas e o Pnotify para exibir mensagens, ambos também JS. Todos os *frameworks* utilizados são encontrados nos próprios sites dos fabricantes e podem ser adquiridos gratuitamente.
A introdução ao desenvolvimento foi a página de acesso ao sistema. Criado um controlador para ela que é responsável por fazer o gerenciamento da dinâmica da página. Este controlador acessa um serviço que faz a conexão com serviços no lado *Server*.
O controlador também tem funções como colocar valores em campos, resgatar e formatar valores vindos do servidor, controle de tabelas, listas e mensagens de erro. Ele precisa de serviços para se conectar com os serviços criados no lado *server*, estes serviços são acessados apenas para recuperar as informações do banco de dados, sendo que a manipulação dos dados recuperados e como eles vão ser demonstrados é função do controlador. O serviço de *login* pode ser observado na figura 29, este possui dois métodos de acesso, ambos POST e duas URL's definidas que servem para encontrar os serviços do lado *server*.
A estrutura básica de todas as outras páginas criadas foi a mesma da página de *login*, um arquivo HTML que contêm o *layout* da página, um controlador JS que utiliza o AngularJS para gerenciar as informações e um arquivo de serviços que faz a conexão com lado *server* da aplicação. Existem outros componentes que necessitaram considerável nível de pesquisa e desenvolvimento, os quais serão descritos a seguir.

3. CONCLUSÃO

A avaliação dentro da prática de ensino, apesar de antiga, ainda é uma boa forma de extrair o conhecimento que o ensinando absorveu. Quando uma avaliação é elaborada por um professor, se este tem a noção de como fazê-la para atingir seus objetivos, ela pode sim refletir o resultado absorvido pelo aluno.
As avaliações feitas em sala de aula podem ser aplicadas de várias maneiras e ter diferentes tipos de questões, existindo então uma diversidade de possíveis testes. Para criar uma avaliação diferente do convencional é investido um tempo maior em sua elaboração, só que ela acaba sendo utilizada poucas vezes, pois perde sua validade assim que publicada. Este conteúdo talvez não tenha mais valia para quem já fez uma vez, porém ela pode ser compartilhada para o crescimento de outras avaliações do mesmo tema que poderão ser criadas.
A minoria dos professores usam recursos que ajudam na elaboração das avaliações, outros ainda contam com técnicas primitivas utilizando questões repetidas, perdendo muita eficiência e eficácia na aplicação da mesma. Esta falta de recursos pode ser por causa da falta de conhecimento dos professores em informática, da aversão a novas tecnologias ou porque não conhecem ferramentas disponíveis.
O protótipo desenvolvido disponibilizou recursos que ajudam os professores a construírem suas avaliações. A possibilidade de criar elas apenas consultando questões que já estão prontas agilizando o processo, além de disponibilizar outras visões, pois as questões podem ter sido elaboradas por pessoas diferentes. O sistema guarda todas as informações das avaliações aplicadas, desta forma torna mais viável a sua reutilização, visto que é só resgatar

uma já aplicada, fazer algumas adaptações e utilizar ela novamente. Com os recursos de embaralhar questões e alternativas destas a reaplicação ficou ainda mais viável.

REFERÊNCIAS

Bordenave, Juan Díaz; Pereira. Adair Martins. Estratégias de Ensino-Aprendizagem. Petrópolis: Vozes, 2006. 312 p.

Burke, Bill. RESTful Java with JAX-RS. Sebastopol: O'Reilly Media, 2010. 312 p. *Conference*, 2005. CCNC. 2005 Second IEEE: 134-138, Jan. 2005.

Cadenhead, Rogers. Sams Teach Yourself Java™ in 24 Hours. 6. United States of America: Sams, 2013. 429 p.

Duckett, Jon. Beginning HTML, XHTML, CSS, and JavaScript. Indianapolis: Wiley Publishing, 2010. 866 p.

Gonçalves, António. Beginning Java EE 7. United States of America: Apress, 2013. 608 p.

Green, Brad; Seshadri, Shyam. AngularJS. Sebastopol,: O'Reilly Media, 2013. 196 p.

Gupta, Arun. Java EE 7 Essentials. Sebastopol: O'Reilly Media, 2013. 362 p.

Kozlowski, Pawel; Darwin, Peter Bacon. Mastering Web Application Development with AngularJS: Build single-page web applications using the power of AngularJS. Birmingham: Packt Publishing, 2013. 372 p

Lindeman, Richard H.. Medidas Educacionais. 6. ed. Rio de Janeiro: Globo, 1987. 176 p.

Luckesi, Cipriano Calos. Avaliação da aprendizagem escolar. 20. ed. São Paulo: Cortez, 2001. 180 p.

Medeiros, Ethel Bauzer. Provas Objetivas, Discursivas, Orais e Práticas: Técnicas de Construção. 7. Rio de Janeiro: Fundação Getulio Vargas, 1983. 178 p.

Melchior, Maria Celina. Avaliação pedagógica: função e necessidade. Porto Alegre: Mercado aberto Ltda, 2002. 150 p.

Moretto, Vasco Pedro. (1942). Prova: um momento privilegiado de estudo não um acerto de contas. Rio de Janeiro: DP&A, 2005. 150 p.

Newmarch, Jan; A RESTful Approach: Clean UPnP without SOAP: Using Java in Service-Oriented Architectures. *Consumer Communications and Networking*

Perrenoud, Philippe. Avaliação: da excelência à regulação das aprendizagens. Porto Alegre: ARTMED, 1999. 183 p.

Vianna, Heraldo Marelim. Testes em educação. 2. São Paulo: IBRASA, 1976. 220 p.