

FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS
FATEC PROFESSOR JESSEN VIDAL

HELEN DE CASSIA DOS REIS

EVASYSTEM – SISTEMA PARA APLICAÇÃO DO
EXAME DE VERIFICAÇÃO DE APRENDIZAGEM
(E.V.A)

São José dos Campos

2017

HELEN DE CASSIA DOS REIS

**EVASYSTEM – SISTEMA PARA APLICAÇÃO DO
EXAME DE VERIFICAÇÃO DE APRENDIZAGEM
(E.V.A)**

Trabalho de Graduação
apresentado à Faculdade de
Tecnologia São José dos Campos,
como parte dos requisitos
necessários para a obtenção do
título de Tecnólogo em Banco de
Dados.

Orientador: Lucas Gonçalves Nadalete

São José dos Campos

2017

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

Reis, Helen Cassia

EVASYSTEM – SISTEMA PARA APLICAÇÃO DO EXAME DE VERIFICAÇÃO DE APRENDIZAGEM (E.V.A).

São José dos Campos, 2017.

69f.

Trabalho de Graduação – Curso de Tecnologia em Banco de Dados,

FATEC de São José dos Campos: Professor Jessen Vidal, 2017.

Orientador: Lucas Gonçalves Nadalete.

Áreas de conhecimento. I. Faculdade de Tecnologia. FATEC de São José dos Campos:

Professor Jessen Vidal. Divisão de Informação e Documentação. II. Título

REFERÊNCIA BIBLIOGRÁFICA –

Reis, Helen C. EVASYSTEM – SISTEMA PARA APLICAÇÃO DO EXAME DE VERIFICAÇÃO DE APRENDIZAGEM (E.V.A). 2017. 69f.

Trabalho de Graduação - FATEC de São José dos Campos: Professor Jessen Vidal.

CESSÃO DE DIREITOS –

NOME DO AUTOR: Helen de Cassia dos Reis

TÍTULO DO TRABALHO: EVASYSTEM – SISTEMA PARA APLICAÇÃO DO EXAME DE VERIFICAÇÃO DE APRENDIZAGEM (E.V.A).

TIPO DO TRABALHO/ANO: Trabalho de Graduação / 2017.

É concedida à FATEC de São José dos Campos: Professor Jessen Vidal permissão para reproduzir cópias deste Trabalho e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte deste Trabalho pode ser reproduzida sem a autorização do autor.



Helen de Cassia dos Reis

Rua Doutor Antônio Mazzuco, nº 190, Residencial Dom Bosco

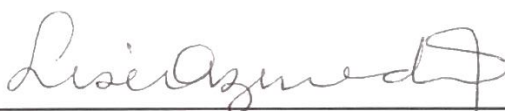
CEP: 12.225-864 – São José dos Campos

Helen de Cassia dos Reis

**EVASYSTEM – SISTEMA PARA APLICAÇÃO DO
EXAME DE VERIFICAÇÃO DE APRENDIZAGEM
(E.V.A)**

Trabalho de Graduação
apresentado à Faculdade de
Tecnologia São José dos Campos,
como parte dos requisitos
necessários para a obtenção do
título de Tecnólogo em Banco de
Dados.

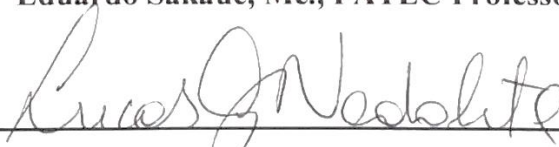
Composição da Banca



Lise Virginia Vieira de Azevedo, Me., FATEC Professor Jessen Vidal



Eduardo Sakaue, Me., FATEC Professor Jessen Vidal



Lucas Gonçalves Nadalete, Me., FATEC Professor Jessen Vidal

14/12/2017

DATA DA APROVAÇÃO

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado força e sabedoria durante toda essa caminhada rumo à conclusão de mais uma graduação. Agradeço aos meus pais, Benedita e Edis, pelo apoio nos momentos difíceis. Durante todo o desenvolvimento deste trabalho tive o apoio e a pronta orientação dos professores Lucas Gonçalves Nadalete que em momento algum duvidou que este trabalho pudesse ser concluído e Lise V. V. Azevedo pela disponibilização das informações para que o trabalho pudesse ser desenvolvido. Agradeço aos meus grandes amigos Mariana Mello, Jerry Gilbert e Miqueias Hidai pelo apoio.

RESUMO

A Faculdade de Tecnologia (FATEC) de São José dos Campos recebe semestralmente Alunos de todas as partes em busca de cursos superiores e com ênfase na preparação para o mercado de trabalho. Como todos sabemos a concorrência por vagas de emprego cresce muito a cada ano e quanto mais preparado o Aluno estiver maiores serão suas chances para conquistar uma vaga. Para preparar os Alunos, além dos cursos de graduação, a FATEC de São José dos Campos aplica o Exame de Verificação de Aprendizagem (E.V.A) aos Alunos que estejam cursando o segundo, quarto e sexto semestres de Inglês de todos os cursos. Desde que foi criada a prova E.V.A é aplicada de forma impressa e sua correção é feita manualmente. Analisando as necessidades tecnológicas nesse contexto, notou-se a possibilidade da criação e implementação de uma ferramenta informatizada, com a proposta de permitir ao Professor criar novas questões e ver os resultados dos alunos a qualquer momento, este mesmo software permite ao Aluno realizar a prova eletronicamente. Utilizando o padrão Model, View and Controller, foi desenvolvido uma ferramenta composta por 2 módulos sendo eles um módulo voltado para o Professor e um módulo responsável por prover ao Aluno a experiência de realizar a prova informatizada. No módulo do Professor é possível gerenciar provas existentes e visualizar em tempo real o desempenho do Aluno. Assim que o Professor iniciar a prova o Aluno poderá acessar o módulo desenvolvido para a responder a prova, os dados de todos os módulos são gravados em um banco de dados. Como resultado espera-se que o Professor ganhe tempo já que a correção era manual, e com o EvaSystem o Professor conseguirá obter os resultados finais por curso e/ou Aluno, podendo assim gerir as provas existentes e novas através do sistema.

Palavras-Chave: e.v.a, eva, java, angularjs, exame, spring mvc

ABSTRACT

Faculdade de Tecnologia (FATEC) placed in São José dos Campos receives students from a lot of cities. These students search for good graduation courses, with an emphasis on preparing for the job market, as we know the competition for job vacancies grows a lot per year and the more prepared the student is, the greater their chances of winning the job. In order to prepare the students, in addition to the undergraduate courses, the FATEC applies the Learning Verification Exam, known as well as Exame de Verificação de Aprendizagem (E.V.A), to its students who are attending the second, fourth and sixth semesters of English to all courses. Since the E.V.A test was created, it is applied in printed form and its corrections are done manually. It was then proposed to create a system where teachers would be able to create new tests, see all results and also where the student could perform the test. Based on the requirements, the software contains two web modules. The first one which is aimed for the teachers' use and the second one is, responsible for managing the online version of the test, available for all the students. In the teachers' module it is possible to manage existing tests and creating new ones. The student will have access to perform the test as soon as it was started by the teacher. The data generate by the modules are recorded in the database. As the result states, the teacher will gain time when using the system, since the test corrections are made by EvaSystem.

Keywords: e.v.a, eva, java, angularjs, exam, spring mvc

LISTA DE ABREVIATURAS E SIGLAS

AWS	<i>Amazon Web Service</i>
CPU	<i>Central Processing Unit</i>
DDL	<i>Data Definition Language</i>
DML	<i>Data Manipulation Language</i>
EVA	Exame de Verificação de Aprendizagem
FATEC	Faculdade de Tecnologia
HTML	<i>HyperText Markup Language</i>
JPA	<i>Java Persistence API</i>
JSON	<i>JavaScript Object Notation</i>
JWT	<i>JSON Web Token</i>
MVC	<i>Model, View, Controller</i>
MER	Modelo Entidade Relacionamento
ORM	<i>Object Relational Mapping</i>
PDF	<i>Portable Document Format</i>
REST	<i>Representational State Transfer</i>
SQL	<i>Structured Query Language</i>
TOEFL	<i>Test of English as a Foreign Language</i>
UC	<i>Use Case</i>

LISTA DE FIGURAS

Figura 1 - Diagrama de Casos de Uso Professor	17
Figura 2 - Diagrama de Casos de Uso Aluno	18
Figura 3 - Protótipo da Tela de Cadastro de Prova.....	24
Figura 4 - Protótipo da Tela de Cadastro Nova Seção	25
Figura 5 - Protótipo da Tela de Cadastro de Parte.....	26
Figura 6 - Protótipo da Tela de Cadastro de Questões	27
Figura 7 - Protótipo da Tela de Questão com Resposta em Áudio.....	28
Figura 8 - Arquitetura Spring Framework.....	30
Figura 9 - Funcionamento JWT.....	31
Figura 10 - Arquitetura JSON	32
Figura 11 - Arquitetura de Solução	33
Figura 12 - Modelo Entidade Relacionamento.....	34
Figura 13 - Representação da Classe Course e da Tabela COURSE Mapeada.....	35
Figura 14 - Persistence.xml	35
Figura 15 - Exemplo Repository Spring Data	36
Figura 16 - Diagrama de Classe das Entidades do Sistema.....	37
Figura 17 - Anotação N:N no Hibernate	37
Figura 18 - Exemplo Rota	38
Figura 19 - Arquitetura dos Módulos Professor e Aluno	39
Figura 20 - Obtenção dos Dados da Visão	39
Figura 21 - Requisição REST.....	40
Figura 22 - Retorno Requisição.....	41
Figura 23 - Spring @RequestBody	42
Figura 24 - Anotação Segurança	43
Figura 25 - Tabelas de Segurança.....	43
Figura 26 - Senha Criptografada com BCrypt.....	44
Figura 27 - Método de Criptografia.....	44
Figura 28 - Arquitetura Geral	47
Figura 29 - Tela <i>Login</i> Professor.....	50
Figura 30 - Cadastro de Prova	51
Figura 31 - Lista de Estudantes	51
Figura 32 - Acessos Pendentes	52

Figura 33 - Tela <i>Login</i> Aluno	52
Figura 34 - Aviso Prova Disponível	53
Figura 35 - Tela Prova	54
Figura 36 - Ajuda com HelpJS	55
Figura 37 - Menu Report	56
Figura 38 - Relatório por Aluno	56
Figura 39 - Relatório PDF Aluno	57
Figura 40 – Lista Alunos Realizando Prova.....	57
Figura 41 - Detalhe Progresso Aluno	58
Figura 42 - Gráfico Resultados Por Questão	59
Figura 43 - Botão Play.....	61
Figura 44 - Estatísticas Java VisualVM	62
Figura 45 - Comparativo Métodos de Tratamento de Áudio.....	63

LISTA DE TABELAS

Tabela 1 - Caso de Uso: Logar no Sistema	18
Tabela 2 - Caso de Uso: Cadastrar Prova	19
Tabela 3 - Caso de Uso: Cadastrar Questões	19
Tabela 4 - Caso de Uso: Iniciar Prova	20
Tabela 5 - Caso de Uso: Cadastrar Nova Parte	20
Tabela 6 - Caso de Uso: Cadastrar Nova Seção	21
Tabela 7 - Caso de Uso: Gerar Relatório por Curso	21
Tabela 8 - Caso de Uso: Gerar Relatório por Aluno	22
Tabela 9 - Caso de Uso: Visualizar Resultados por Questão	22
Tabela 10 - Caso de Uso: Realizar Prova	23
Tabela 11 - Mapeamento Serviços POST	45
Tabela 12 - Mapeamento Serviços GET	46
Tabela 13 - Configurações do Servidor do Módulo Backend	47
Tabela 14 - Configurações do Servidor de Banco de Dados	48
Tabela 15 – Configurações do Servidor dos Módulos Frontend	49
Tabela 16 - Resultados Experimento 1 Professor	59
Tabela 17 - Resultados Experimento 1 Aluno	60
Tabela 18 - Resultados Experimento 2 Aluno	61
Tabela 19 - Ganhos Identificados com o EvaSystem	65

SUMÁRIO

1. INTRODUÇÃO	14
1.1. Problema.....	15
1.2. Objetivo Geral	15
1.2.1. Objetivos Específicos	15
1.2.2. Detalhes das Funcionalidades	16
2. LEVANTAMENTO DE REQUISITOS E TECNOLOGIAS	17
2.1. Casos de Uso	17
2.2. Especificação de Casos de Uso	18
2.2.1. Caso de Uso Compartilhado	18
2.2.2. Casos de Uso Professor	19
2.2.3. Caso de Uso Aluno	23
2.3. Protótipos de Telas	23
2.3.1. Protótipo Tela de Cadastro de Prova	24
2.3.2. Protótipo Tela de Cadastro de Seção	25
2.3.3. Protótipo Tela de Cadastro de Parte	26
2.3.4. Protótipo Tela de Cadastro de Questões	27
2.4. Tecnologias Utilizadas	29
2.4.1. Liquibase e MySQL	29
2.4.2. Apache Maven	29
2.4.3. Java	29
2.4.4. Hibernate	30
2.4.5. Spring Framework	30
2.4.6. JSON Web Token	31
2.4.7. Angular JS	31
2.4.8. JSON	32
3. DESENVOLVIMENTO	33
3.1. Arquitetura do Sistema	33
3.1.1. Banco de Dados	33
3.1.2. Persistência	34
3.1.3. <i>Web Service</i>	37
3.1.4. Visão	38

3.2. Serviços	40
3.1.1. Serviços Tipo GET	41
3.1.2. Serviços Tipo POST	41
3.3. Segurança	42
3.3.1. JSON Web Token.....	42
3.3.2. Spring Security	43
3.3.3. Segurança Aplicada em Senhas.....	44
3.4. Rotas Disponibilizadas pelo <i>Web Service</i>	44
3.4.1. Rotas POST	45
3.4.2. Rotas GET	46
3.5. Infraestrutura para Implantação.....	47
3.5.1. Módulo Backend	47
3.5.2. Servidor de Banco de Dados	48
3.5.3. Módulos Frontend	49
4. RESULTADOS	50
4.1. Módulo Professor	50
4.2. Módulo Aluno	52
4.3. Experimentos.....	55
4.3.1. AdminTeacher – Experimento 1	55
4.3.2. AdminTeacher – Experimento 2	56
4.3.3. AdminStudent - Experimento 1	60
4.3.4. AdminStudent - Experimento 2.....	60
4.3.5. AdminStudent – Experimento 3	62
5. CONCLUSÃO	64
5.1. Lições Aprendidas.....	64
5.2. Considerações Finais	64
5.3. Trabalhos Futuros.....	66
REFERÊNCIAS	68

1. INTRODUÇÃO

Semestralmente o Exame de Verificação de Aprendizagem (E.V.A) é aplicado para todo os Alunos que estão matriculados na disciplina de língua de inglesa nos semestres II, IV e VI.

O exame possui os mesmos moldes do inglês *Test Of English as a Foreign Language* (TOEFL), onde o Aluno responde questões de *Reading* e *Listening* permitindo assim que duas importantes habilidades da língua sejam avaliadas.

O E.V.A é dividido em três níveis e cada nível é constituído por 2 (duas) seções (*Listening e Reading*), 7 (sete) Partes (*Photographs, Question-Response, Conversation, Talks, Incomplete Sentences, Text Completion e Reading Comprehension*) onde cada parte possui um número variado de questões.

A primeira seção da prova possui 4 (quatro) partes onde o Aluno responde questões com foco em audição; já a segunda seção da prova possui questões com foco em leitura, interpretação de texto e gramática.

Com aplicação de forma impressa do E.V.A o Aluno não consegue seguir efetivamente seu ritmo e o Professor precisa dedicar seu tempo corrigindo as provas.

As respostas não devem ser assinaladas na folha de questões, porém muitos Alunos assinalam fazendo com que aquela prova impressa não possa ser reaproveitada e aplicada a outros Alunos posteriormente.

Na seção de *Listening* o Aluno precisa escutar o áudio para responder às questões, o áudio é gerenciado pelo Professor e reproduzido uma única vez para que o Aluno responda as questões relacionadas. Esse áudio é tocado em caixas de som para que todos possam ouvir ao mesmo tempo, portando o Aluno que sentar-se distante das caixas terá maior dificuldade de compreender o que foi dito.

Com esse processo do Professor gerenciar o áudio o Aluno não consegue seguir seu ritmo, com isso ele precisa aguardar com que todos os Alunos respondam para que ele possa ir para próxima questão. Na seção *Reading* o Aluno precisa escolher uma alternativa para a pergunta e assinalar na folha de resposta.

Caso o Aluno, por motivo de falta atenção ou por não ter analisado a questão corretamente, assinale a alternativa incorreta, ele não terá a chance de alterar sua resposta pois o gabarito não permite rasuras.

Após todo o processo de aplicação da prova ter passado, o Professor então precisa realizar a correção das mesmas. Esse processo é feito na folha de resposta do aluno utilizando o gabarito do professor.

Atualmente todo o processo de gerar estatísticas de como os Alunos se saíram em determinada parte da prova é feita manualmente, todos os levantamentos são feitos com base nas provas corrigidas, e esse processo leva dias para ser concluído, com isso não é possível ser aplicada de forma imediata uma correção em sala de aula para o problema identificado.

1.1. Problema

Problemas identificados sob a perspectiva do Professor:

- Perda de tempo na execução da prova;
- Perda de tempo no processo de correção das provas e consolidação dos resultados finais;
- Alteração de provas já existentes gera custos;

Problemas identificados sob a perspectiva do Aluno:

- Dificil compreensão do áudio nas seções de *listening*;
- Controle do áudio feito pelo professor;

1.2. Objetivo Geral

Prover um sistema para gerenciamento e realização do Exame de Verificação de Aprendizagem.

1.2.1. Objetivos Específicos

- Estudo para entender e compreender como é todo o processo do Exame de Verificação de Aprendizagem, desde a sua criação até sua aplicação;
- Levantamento de requisitos funcionais e não funcionais para o desenvolvimento da ferramenta;
- Entendimento as regras que compõe o E.V.A;
- Desenvolvimento um módulo específico para o Professor gerir as provas e consultar resultados dos Alunos;

- Desenvolvimento um módulo que permita ao Aluno realizar a prova;
- Armazenamento das informações no banco de dados para posteriores consultas;
- Propiciar ao Professor uma funcionalidade no sistema capaz fornecer os resultados dos Alunos a qualquer momento;
- Permitir ao Professor extrair relatórios do desempenho dos alunos;

1.2.2. Detalhes das Funcionalidades

O módulo do Professor deve permitir cadastro e gestão de provas, obtenção de resultados por Aluno, por curso e resultados estatísticos com base nos componentes da prova. Além disso o módulo deverá fornecer ao Professor a funcionalidade de iniciar a prova somente para os Alunos que desejar.

O módulo do Aluno deve possibilitar a realização da prova no tempo estimado pelo Professor. As questões devem ser apresentadas de forma clara sem gerar dúvidas, os áudios e imagens devem ser claros e concisos com a questão.

2. LEVANTAMENTO DE REQUISITOS E TECNOLOGIAS

Após reuniões com a coordenadora do E.V.A na Fatec de São José dos Campos, foi possível identificar o processo realizado desde a criação da prova até a obtenção dos resultados.

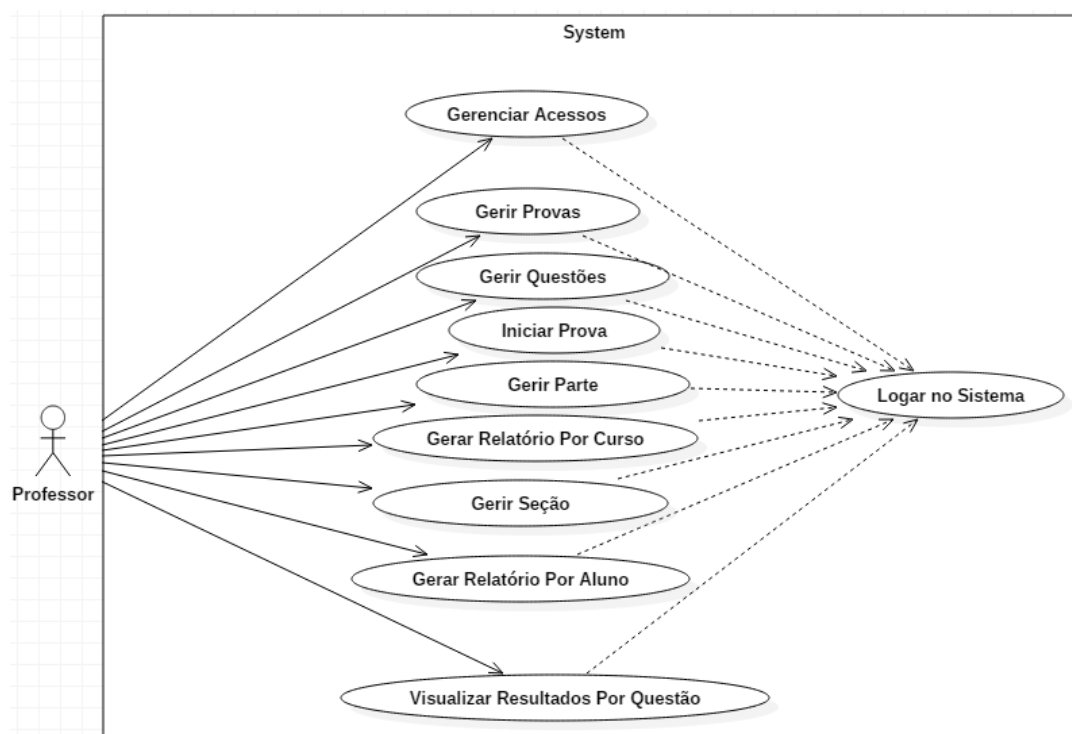
2.1.Casos de Uso

Foi identificado dois atores que utilizarão o sistema, Professor e Aluno.

O Professor terá todo o acesso para gestão das provas e resultados, o Aluno terá acesso apenas para realização da prova.

A Figura 1 nos mostra os casos de uso relacionados com o ator Professor.

Figura 1 - Diagrama de Casos de Uso Professor

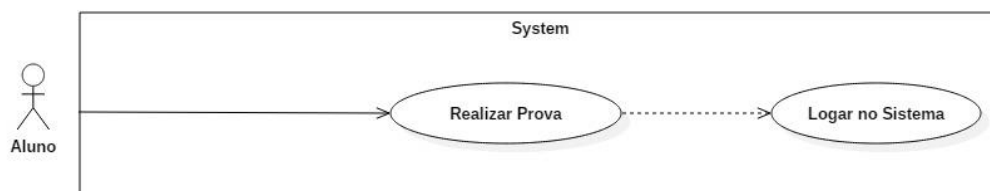


Fonte: Autor, 2017.

Todos os casos de uso do Professor possuem como dependência o Caso de Uso Logar no Sistema.

Outro ator identificado foi o Aluno, este por sua vez terá apenas a ação de realizar a prova. A Figura 2 ilustra o Caso de Uso do Aluno.

Figura 2 - Diagrama de Casos de Uso Aluno



Fonte: Autor, 2017.

O Aluno possui dois casos de uso sendo que Realizar Prova só poderá ser alcançado após a ação de Logar no Sistema ter sido efetuada com sucesso.

2.2. Especificação de Casos de Uso

Conforme mencionado foram identificados dois perfis de acesso, Professor e Aluno, foi observado que Aluno e Professor compartilham apenas o Caso de Uso Logar no Sistema.

2.2.1. Caso de Uso Compartilhado

Tabela 1 - Caso de Uso: Logar no Sistema

ID do Caso de Uso	UC1
Nome do Caso de Uso	Logar no Sistema
Ator Principal	Professor/Aluno
Resumo	Esse Caso de Uso descreve as etapas percorridas pelo Professor ou Aluno conseguir acessar a página principal do sistema
Fluxo Principal	
Ações dos atores Aluno e Professor	Ações do Sistema
1 – Ator Acessa página de autenticação.	Sistema carrega formulário para inserir usuário e senha
2 - Informar usuário e senha.	
3 - Logar	Sistema carrega página principal
Restrições/Validações	1 - É necessário que o usuário e senha informados tenham sido previamente cadastrados e estejam corretos.
Fluxo de Exceção - Informações de dados para autenticação incorretos	
Ações dos atores Aluno e Professor	Ações do Sistema AdminStudent e AdminTeacher
Ator informa usuário ou senha incorretos	1 - Apresentar mensagem "Usuário ou senha incorretos".

Fonte: Autor, 2017.

O Caso de Uso UC1 presente na Tabela 1 representa ações de Professor e Aluno.

2.2.2. Casos de Uso Professor

Tabela 2 - Caso de Uso: Cadastrar Prova

ID do Caso de Uso	UC2
Nome do Caso de Uso	Cadastrar Prova
Ator Principal	Professor
Resumo	Esse Caso de Uso descreve as etapas percorridas pelo Professor para cadastrar nova prova.
Pré-condições	Executar o Caso de Uso UC1
Fluxo Principal	
Ações do Ator Professor	Ações do Sistema AdminTeacher
1 – Ator aguarda página principal carregar	Sistema exibe tela principal.
2 – Ator seleciona “New Test”	Sistema abre a tela com o formulário para cadastrar prova.
3 – Ator preenche todos os campos do formulário	
4 – Ator escolhe salvar	Sistema salva os dados no banco de dados
Restrições/Validações	1 – Todos os campos devem estar preenchidos.
Fluxo de Exceção - Informações de campos não preenchidos	
Ações do Ator Professor	Ações do Sistema AdminTeacher
Tentar salvar sem que todas as informações estejam preenchidas	1 – Exibir uma mensagem que todas as informações devem ser fornecidas

Fonte: Autor, 2017.

O Caso de Uso UC2 representa as ações necessárias para cadastrar uma nova prova.

Tabela 3 - Caso de Uso: Cadastrar Questões

ID do Caso de Uso	UC3
Nome do Caso de Uso	Cadastrar Questões
Ator Principal	Professor
Resumo	Esse Caso de Uso descreve as etapas percorridas pelo Professor para cadastrar nova questão.
Pré-condições	Executar o Caso de Uso UC1
Fluxo Principal	
Ações do Ator Professor	Ações do Sistema AdminTeacher
1 – Ator aguarda página principal carregar	Sistema redireciona usuário para tela principal de realização da prova.
2 – Ator seleciona “New Question”	Sistema carrega a tela com o formulário para cadastrar nova questão.
3 – Ator preenche todos os campos do formulário	
4 – Salvar	Sistema salva os dados no banco de dados
Restrições/Validações	1 – Todos os campos devem estar preenchidos.
Fluxo de Exceção - Informações de campos não preenchidos	
Ações do Ator Professor	Ações do Sistema AdminTeacher
Tentar salvar sem que todas as informações estejam preenchidas	1 – Exibir uma mensagem que todas as informações devem ser fornecidas

Fonte: Autor, 2017.

O Caso de Uso UC3 representa as ações necessárias para cadastrar uma nova questão.

Tabela 4 - Caso de Uso: Iniciar Prova

ID do Caso de Uso	UC4
Nome do Caso de Uso	Iniciar Prova
Ator Principal	Professor
Resumo	Esse Caso de Uso descreve o passo-a-passo para iniciar uma nova prova.
Pré-condições	Executar o Caso de Uso UC1
Fluxo Principal	
Ações do Ator Professor	Ações do Sistema AdminTeacher
1 – Ator aguarda página principal carregar	Sistema exibe tela principal.
2 – Ator seleciona “ <i>Start New Test</i> ”	
3 – Ator aguarda o carregamento da página com o botão de importar lista	Sistema carrega a página onde será possível preencher as informações para iniciar a prova.
4 – Ator escolhe a lista de alunos	Sistema exibe lista escolhida
5 – Ator inicia prova	Sistema carrega todas as questões e mídias da prova iniciada
Restrições/Validações	Todos os campos deverão estar preenchidos
Fluxo de Exceção – Erro ao Iniciar Prova	
Ações do Ator Professor	Ações do Sistema AdminTeacher
Importar lista de estudantes	Sistema exibe a mensagem de que houve um erro ao importar a lista de estudante informada.
Iniciar Prova	Sistema exibe a mensagem de que houve um erro inicializar a prova escolhida

Fonte: Autor, 2017.

O Caso de Uso UC4 representa as ações necessárias para iniciar uma nova prova e os alunos que irão realizar.

Tabela 5 - Caso de Uso: Cadastrar Nova Parte

ID do Caso de Uso	UC5
Nome do Caso de Uso	Cadastrar Nova Parte
Ator Principal	Professor
Resumo	Esse Caso de Uso descreve o passo-a-passo para cadastrar nova parte
Pré-condições	Executar o Caso de Uso UC1
Fluxo Principal	
Ações do Ator Professor	Ações do Sistema AdminTeacher
1 – Ator aguarda tela principal carregar	Sistema exibe tela principal.
2 – Ator seleciona “ <i>New Part</i> ”	Sistema carrega a tela com o formulário para cadastro de nova parte.
3 – Ator preenche todos os campos do formulário	
4 – Ator salva	Sistema salva os dados no banco de dados.
Restrições/Validações	1 – Todos os campos deverão estar preenchidos
Fluxo de Exceção – Erro ao Salvar Parte	
Ações do Ator Professor	Ações do Sistema
Salvar	1 – Sistema exibe a mensagem de que houve um erro ao salvar a nova parte

Fonte: Autor, 2017.

O Caso de Uso UC5 representa as ações necessárias para cadastrar uma nova Parte da prova.

Tabela 6 - Caso de Uso: Cadastrar Nova Seção

ID do Caso de Uso	UC6
Nome do Caso de Uso	Cadastrar Nova Parte
Ator Principal	Professor
Resumo	Esse Caso de Uso descreve o passo-a-passo para cadastrar nova seção
Pré-condições	Executar o Caso de Uso UC1
Fluxo Principal	
Ações do Ator Professor	Ações do Sistema AdminTeacher
1 – Ator aguarda tela principal carregar	Sistema exibe tela principal.
2 – Ator escolhe “New Section”	Sistema carrega a tela com o formulário para cadastro de nova seção.
3 – Ator preenche todos os campos do formulário.	
7 - Salvar	Sistema salva os dados no banco de dados.
Restrições/Validações	1 – Todos os campos deverão estar preenchidos
Fluxo de Exceção – Erro ao Salvar Seção	
Ações do Ator Professor	Ações do Sistema AdminTeacher
Salvar	1 – Sistema exibe a mensagem de que houve um erro ao salvar a nova seção

Fonte: Autor, 2017.

O Caso de Uso UC6 representa as ações necessárias para cadastrar uma nova Seção da prova.

Tabela 7 - Caso de Uso: Gerar Relatório por Curso

ID do Caso de Uso	UC7
Nome do Caso de Uso	Gerar Relatório por Curso
Pré-condições	Executar o Caso de Uso UC1
Ator Principal	Professor
Resumo	Esse Caso de Uso descreve o passo-a-passo para obter o relatório de desempenho de todos os Alunos de um determinado curso
Pré-condições	Executar o Caso de Uso UC1
Fluxo Principal	
Ações do Ator Professor	Ações do Sistema AdminTeacher
1 – Ator aguarda tela principal carregar	Sistema exibe tela principal.
2 – Ator seleciona “Report by Course”	Sistema carrega a tela com o formulário para gerar o relatório.
3 – Ator preenche todos os campos necessários para gerar o relatório.	
4 – Gerar Relatório	Sistema consulta o banco de dados e gerar o relatório em PDF com todos os Alunos do curso escolhido que fizeram a prova.
Restrições/Validações	1 – Todos os campos deverão estar preenchidos.
Fluxo de Exceção – Erro ao Gerar Relatório do Curso	
Ações do Ator Professor	Ações do Sistema AdminTeacher
Escolher Gerar Relatório	1 – Sistema exibe a mensagem de que houve um erro ao gerar o relatório do curso

Fonte: Autor, 2017.

O Caso de Uso UC7 representa as ações necessárias para extrair relatório por curso.

Tabela 8 - Caso de Uso: Gerar Relatório por Aluno

ID do Caso de Uso	UC8
Nome do Caso de Uso	Gerar Relatório por Curso
Ator Principal	Professor
Resumo	Esse Caso de Uso descreve o passo-a-passo para obter o relatório de desempenho de um Aluno.
Pré-condições	Executar o Caso de Uso UC1
Fluxo Principal	
Ações do Ator Professor	Ações do Sistema AdminTeacher
1 – Ator aguarda tela principal carregar	Sistema redireciona usuário para tela principal.
2 – Ator seleciona “ <i>Report By Student</i> ”	Sistema carrega a tela com o formulário para gerar o relatório
3 -Ator preencher todos os campos necessários para gerar o relatório.	
4 – Ator escolhe Gerar Relatório	Sistema consulta o banco de dados e gerar o relatório em PDF com o resultado do Aluno escolhido.
Restrições/Validações	1 – Todos os campos deverão estar preenchidos.
Fluxo de Exceção – Erro ao Gerar Relatório do Curso	
Ações do Ator Professor	Ações do Sistema AdminTeacher
Escolher Gerar Relatório	1 – Sistema exibe a mensagem de que houve um erro ao gerar o relatório o Aluno

Fonte: Autor, 2017.

O Caso de Uso UC8 representa as ações necessárias para extrair relatório por Aluno.

Tabela 9 - Caso de Uso: Visualizar Resultados por Questão

ID do Caso de Uso	UC9
Nome do Caso de Uso	Gerar Relatório por Curso
Ator Principal	Professor
Resumo	Esse Caso de Uso descreve o passo-a-passo para obter o relatório de desempenho de um Aluno.
Pré-condições	Executar o Caso de Uso UC1
Fluxo Principal	
Ações do Ator Professor	Ações do Sistema AdminTeacher
1 – Ator aguarda tela principal carregar	Sistema redireciona usuário para tela principal.
2 – Ator seleciona “ <i>Overall Results</i> ”	Sistema carrega a tela com o formulário para gerar o relatório
3 – Ator preenche todos os campos necessários para gerar o gráfico das questões.	
4 – Ator seleciona Gerar Gráfico	Sistema consulta o banco de dados para gerar o gráfico com a quantidade de Alunos que acertaram cada questão.
Restrições/Validações	1 – Todos os campos deverão estar preenchidos.
Fluxo de Exceção – Erro ao Gerar Gráfico com as Questões	

Ações do Ator Professor	Ações do Sistema AdminTeacher
Escolher Gerar Gráfico	1 – Sistema exibe a mensagem de que houve um erro ao gerar o gráfico

Fonte: Autor, 2017.

O Caso de Uso UC9 representa as ações necessárias para extrair relatório por curso.

2.2.3. Caso de Uso Aluno

Tabela 10 - Caso de Uso: Realizar Prova

ID do Caso de Uso	UC10
Nome do Caso de Uso	Cadastrar Aluno
Ator Principal	Aluno
Resumo	Esse Caso de Uso descreve o passo-a-passo para o Aluno realizar a prova.
Pré-condições	Executar o Caso de Uso UC1
Fluxo Principal	
Ações do Ator Aluno	Ações do Sistema AdminStudent
1 – Ator aguarda tela principal carregar	Sistema redireciona usuário para tela principal de realização da prova.
2 – Ator escolhe iniciar prova	
3 – Ator aguarda o carregamento da primeira questão	Sistema carrega primeira questão
4 – Ator escolhe uma alternativa	
5 – Ator escolhe ir para próxima questão	Sistema carrega próxima questão
6 – Ator Realiza os passos 4, 5 até o final da prova	
7 – Ator seleciona finalizar prova	Sistema carregar página inicial.
Restrições/Validações	1 – Todas as questões deverão estar respondidas
Fluxo de Exceção - Informações de questão não respondida	
Ações do Ator Aluno	Ações do Sistema AdminStudent
Ir para próxima questão sem escolher alternativa	1 – Sistema exibe a mensagem de que a questão deve ser respondida para continuar

O Caso de Uso UC10 representa as ações necessárias feitas pelo aluno para que seja possível realizar a prova.

2.3. Protótipos de Telas

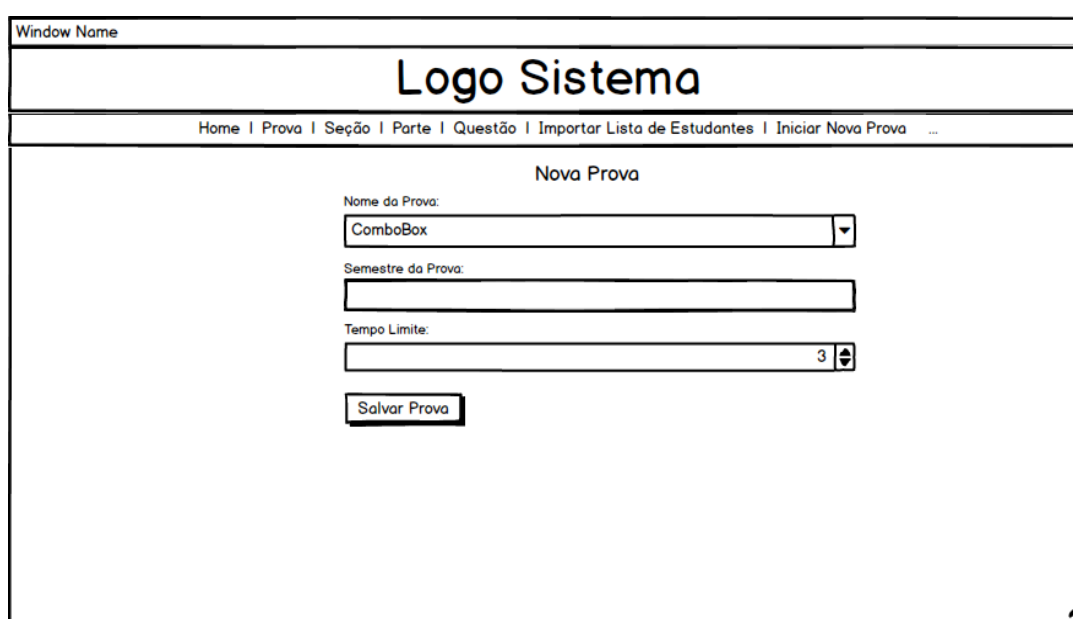
Nesta seção serão apresentados os protótipos das telas. As telas que compõem serão acessadas somente pelo Professor, previamente cadastrado, que efetuou o acesso ao sistema.

A partir do momento que o Professor acessa o sistema ele terá disponível algumas opções das ações que podem ser realizadas por ele. Pela barra de menu o Professor poderá acessar opções de cadastro e edição de Provas, Seção, Partes e Questões.

2.3.1. Protótipo Tela de Cadastro de Prova

A Figura 3 exibe o protótipo da tela de cadastro de nova prova. A tela possui três campos principais e obrigatórios.

Figura 3 - Protótipo da Tela de Cadastro de Prova



O protótipo da tela de cadastro de prova apresenta uma interface com uma barra de menu no topo contendo links como 'Home', 'Prova', 'Seção', 'Parte', 'Questão', 'Importar Lista de Estudantes' e 'Iniciar Nova Prova'. Abaixo da barra, o título 'Logo Sistema' é exibido. O formulário principal, intitulado 'Nova Prova', contém três campos obrigatórios: 'Nome da Prova' (um campo de texto com uma seta para baixo), 'Semestre da Prova' (um campo de texto) e 'Tempo Limite' (um campo de texto com o valor '3' e uma seta para cima). Um botão 'Salvar Prova' está localizado abaixo dos campos.

Fonte: Autor, 2017.

Como podemos ver na Figura 3 a tela de cadastro de provas possui três campos e um botão para salvar os dados preenchidos nos campos.

- **Nome da Prova:** Campo do tipo texto onde o Professor irá inserir o nome que deseja dar para aquela prova que será criada, esse nome será referência para criação de todos os outros elementos que compõe a prova.
- **Semestre da Prova:** Campo do tipo *combobox* onde o Professor irá selecionar para qual semestre aquela prova foi desenvolvida.

- **Tempo Limite:** Campo onde o Professor indicará o tempo máximo para realização da prova, nesses campos são aceitos somente números que variam entre 30 e 80 minutos.

2.3.2. Protótipo Tela de Cadastro de Seção

A Figura 4 exibe o formulário de cadastro de nova Seção, semelhante à tela de cadastro de provas aqui também temos 3 (três) campos que deverão ser preenchidos pelo Professor.

Figura 4 - Protótipo da Tela de Cadastro Nova Seção



O protótipo da tela de cadastro de nova seção apresenta a seguinte estrutura:

- Barra superior: "Window Name" e "Logo Sistema".
- Barra de navegação: "Home | Prova | Seção | Parte | Questão | Importar Lista de Estudantes | Iniciar Nova Prova ...".
- Título central: "Nova Seção".
- Formulário de entrada:
 - "Nome da Prova:" seguido de um campo de tipo ComboBox.
 - "Nome da Seção:" seguido de um campo de texto.
 - "Descrição:" seguido de um campo de texto maior.
- Botão: "Salvar Seção".

Fonte: Autor, 2017.

Ao observarmos a Figura 4 veremos que temos 2 (dois) campos do tipo texto e 1 (um) campo do tipo *combobox*.

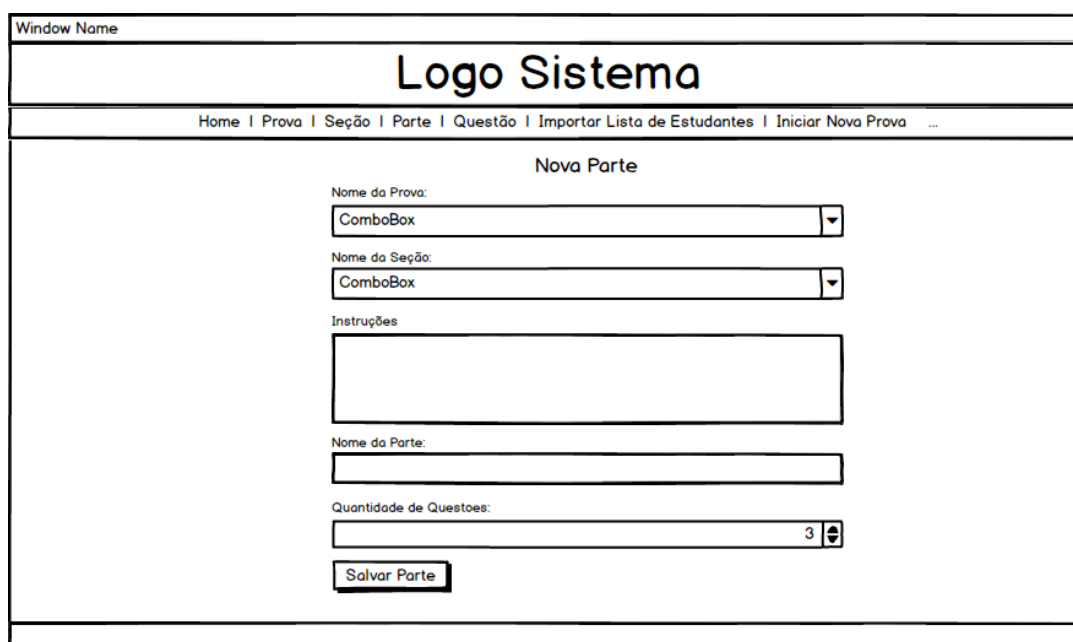
- **Nome da Prova:** Será apresentado ao Professor todos os nomes de provas previamente cadastrados.
- **Nome da Seção:** O Professor poderá indicar o nome que deseja dar àquela seção.
- **Descrição:** explicação clara e sucinta da seção que está sendo criada.

2.3.3. Protótipo Tela de Cadastro de Parte

A Figura 5 exibe o formulário para cadastro de uma nova parte, como vimos na introdução uma parte pode conter várias questões, devido à essa característica o cadastro da parte possui um campo onde o Professor irá indicar quantas questões poderá ter naquela parte.

Como podemos ver na Figura 5 a tela de cadastro de parte possui 5 (cinco) campos e um botão para salvar os dados preenchidos.

Figura 5 - Protótipo da Tela de Cadastro de Parte



Window Name

Logo Sistema

Home | Prova | Seção | Parte | Questão | Importar Lista de Estudantes | Iniciar Nova Prova ...

Nova Parte

Nome da Prova:
ComboBox

Nome da Seção:
ComboBox

Instruções

Nome da Parte:

Quantidade de Questoes:

Salvar Parte

Fonte: Autor, 2017.

- **Nome da Prova:** Campo do tipo texto onde o Professor irá inserir o nome que deseja dar para aquela prova que será criada, esse nome será referência para criação de todos os outros elementos que compõe a prova.
- **Nome da Seção:** Campo do tipo *combobox* onde o Professor irá selecionar qual seção deseja incluir a nova parte.
- **Instruções:** Campo onde o Professor irá inserir informações a respeito da nova parte.

- **Nome da Parte:** Campos do tipo texto onde o Professor irá indicar o nome.
- **Quantidade de Questões:** Campo do tipo número que receberá a quantidade de questões que aquela parte possui para posteriormente ser utilizado no módulo do Aluno.

2.3.4. Protótipo Tela de Cadastro de Questões

A Figura 6 exibe o formulário para cadastro de novas questões. As questões podem conter arquivos de áudio e vídeo, cada questão pode conter os dois tipos de mídia.

Figura 6 - Protótipo da Tela de Cadastro de Questões

Window Name

Logo Sistema

Home | Prova | Seção | Parte | Questão | Importar Lista de Estudantes | Iniciar Nova Prova ...

Nova Questão

Nome da Prova:

Nome da Seção:

Nome da Parte:

Enunciado da Questão:

☒ Complementar Enunciado com Imagem:

Seleção de Imagem:

Imagem:

Seleção de Áudio:

☒ Complementar Enunciado com Áudio

Áudio:

Respostas:

☐ Resposta em Áudio

Resposta A:

☐ Resposta Correta

Resposta B:

☐ Resposta Correta

Resposta C:

☐ Resposta Correta

Resposta D:

☐ Resposta Correta

Fonte: Autor, 2017.

Como podemos observar a tela de cadastro de questões possui uma série de campos a serem preenchidos, porém alguns campos não são obrigatórios.

O grupo Seleção de Imagem é exibido somente quando o *checkbox* “Complementar Enunciado com Imagem” é selecionado pois assim o Professor conseguirá inserir como parte da pergunta uma imagem.

Temos também um outro grupo que é exibido quando o Professor deseja complementar o enunciado com áudio.

As respostas podem ser inseridas pelo Professor de duas maneiras distintas, por meio de áudio ou então o próprio Professor digita as alternativas para questão. Caso seja por meio de áudio o formulário acima sofre modificações em sua estrutura de resposta, essa modificação pode ser observada na Figura 7.

Figura 7 - Protótipo da Tela de Questão com Resposta em Áudio

Window Name

Logo Sistema

Home | Prova | Seção | Parte | Questão | Importar Lista de Estudantes | Iniciar Nova Prova ...

Nova Questão

Nome da Seção:

Nome da Parte:

Enunciado da Questão:

☒ Complementar Enunciado com Imagem:

Seleção de Imagem:

Imagem:

Seleção de Áudio:

☒ Complementar Enunciado com Áudio

Áudio:

Respostas:

☒ Resposta em Áudio

Áudio:

☐ A: Correct Answer ☐ B: Correct Answer ☐ C: Correct Answer ☐ D: Correct Answer

Fonte: Autor, 2017.

Quando a resposta está em um áudio o Professor não irá preencher os campos de alternativas, ele apenas irá marcar qual a alternativa é correta.

2.4. Tecnologias Utilizadas

Nesta seção serão apresentadas as tecnologias utilizadas para o desenvolvimento do sistema.

2.4.1. Liquibase e MySQL

Como banco de dados foi utilizado MySql na versão 5.0. O MySql é um Sistema Gerenciados de Banco de Dados de código aberto com grande capacidade de armazenamento de informações, confiável e de simples utilização, essa ferramenta é utilizada por grandes empresas ao redor do mundo como, Facebook, Google e Adobe (MYSQL, 2017)

Para criação e gerenciamento do banco de dados foi utilizada a ferramenta de controle de scripts DDL (Data Definition Language) e DML (Data Manipulation Language) Liquibase. Esta ferramenta também de código aberto faz todo o versionamento do banco de dados, evitando assim inconsistências e até mesmo erros durante as atualizações do banco (LIQUIBASE, 2017).

2.4.2. Apache Maven

Para a gestão das bibliotecas no sistema foi utilizada a ferramenta Maven.

Apache Maven é uma ferramenta de código aberto desenvolvida pela Apache Foundation muito utilizada pelo mercado para controle de bibliotecas e versionamento de pacotes em projetos Java, ela possui um arquivo principal chamado *pom.xml* onde são inseridas todas as bibliotecas que serão utilizadas pelo projeto e após serem inseridas no arquivo o Maven faz o download de todas e automaticamente associa-as ao projeto (APACHE MAVEN, 2017).

2.4.3. Java

Para desenvolvimento do *Web Service* foi escolhida a linguagem de programação Java, o *Web Service* foi projetado utilizando a arquitetura REST.

Foi escolhida a linguagem de programação Java devido à seu vasto portfólio de bibliotecas e sua característica de portabilidade. Java é uma linguagem de fácil entendimento, orientada à objetos (BOYARSKY J. SELIKOFF S, 2017);

2.4.4. Hibernate

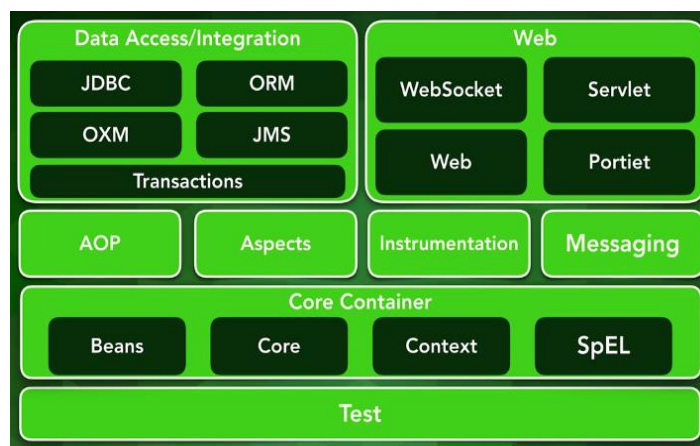
Hibernate é um *framework* utilizado para implementar a camada de persistência *Java Persistence API* (JPA), o *Hibernate* fornece também a facilidade de mapeamento entre o banco de dados e a camada de serviço por meio do *Object Relational Mapping* (ORM). (HIBERNATE, 2017).

2.4.5. Spring Framework

O *Framework Spring*, desenvolvido em 2003, é baseado na tecnologia Java e possui a característica de injeção de dependência e gestão de objetos Java, dentre outras funções (SPRING, 2017).

O Spring é dividido em vários módulos onde cada módulo possui responsabilidades diferentes, a Figura 8 nos mostra como os módulos são dispostos.

Figura 8 - Arquitetura Spring Framework



Fonte: Adaptado Udemy, 2017

A arquitetura Spring é dividida em 4 áreas onde cada módulo possui comportamentos específicos.

- *Test*
- *Core Container*
- *Data Access/Integration*

- *Web*

Para o desenvolvimento do software serão utilizados os módulos *Servlet*, *Beans* e *Core*, localizados em *Web* e *Core Container* respectivamente, ambos os módulos serão empregados na camada de serviço (SPRING, 2017)

2.4.6. JSON Web Token

JSON Web Token (JWT) é definido de acordo com as regras da RFC 7519 e é considerado um modo seguro de representar informações que serão transferidas entre duas camadas. A Figura 9 nos mostra como o JWT funciona.

Figura 9 - Funcionamento JWT



Fonte: Adaptado de Mikey Stecky, 2017

Como podemos observar na Figura 9 quando o usuário realiza a requisição de *login* na aplicação o servidor de autenticação é o responsável por validar os dados do *login* e em seguida gerar chave da autenticação (*token*), posteriormente toda e qualquer requisição que este usuário venha à fazer ele deverá informar o token à requisição (VADIUM SOFTWARE, 2017).

2.4.7. Angular JS

AngularJS é um framework baseado na tecnologia JavaScript voltada para desenvolvimento da camada visão da aplicação. Seu funcionamento baseia-se em tags chamadas diretivas inseridas diretamente no código HTML permitindo assim o fácil manuseio dos dados a serem exibidos para o usuário.

O Framework AngularJS possui funções que facilitam e agilizam o manuseio de dados de uma tela permitindo assim fácil manutenção do código (ANGULARJS, 2017).

2.4.8. JSON

JSON (*JavaScript Object Notation*) é um formato de dados utilizado para comunicação entre camadas dos sistemas, sua arquitetura principal baseia-se em chaves e valores, conforme Figura 10.

Figura 10 - Arquitetura JSON

```
{  
  "testId": 2,  
  "testSemester": 2,  
  "testTimeLimit": 65,  
  "testName": "ds"  
}
```

Fonte: Autor, 2017.

Os campos do JSON são dispostos na forma de chave e valor, utilizando nosso exemplo veremos que “testId” é a chave e o valor para este campo é 2, o nome da chave do JSON é exatamente o mesmo nome dado a esse atributo na respectiva classe Java (JSON, 2017).

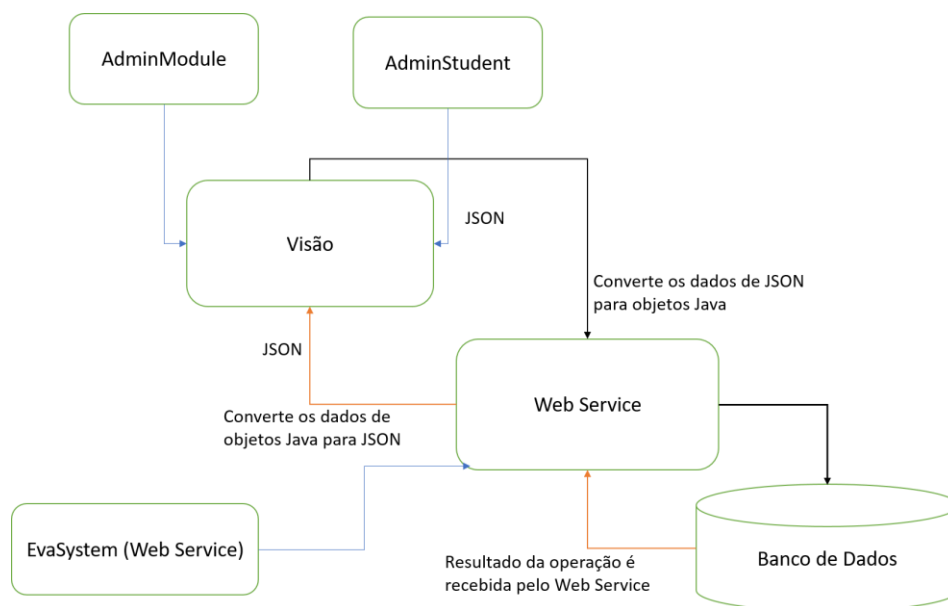
3. DESENVOLVIMENTO

Este capítulo apresenta a arquitetura de solução proposta, os detalhes das camadas do sistema desenvolvido.

3.1. Arquitetura do Sistema

A Figura 11 exibe as camadas da Arquitetura de Solução do Sistema.

Figura 11 - Arquitetura de Solução



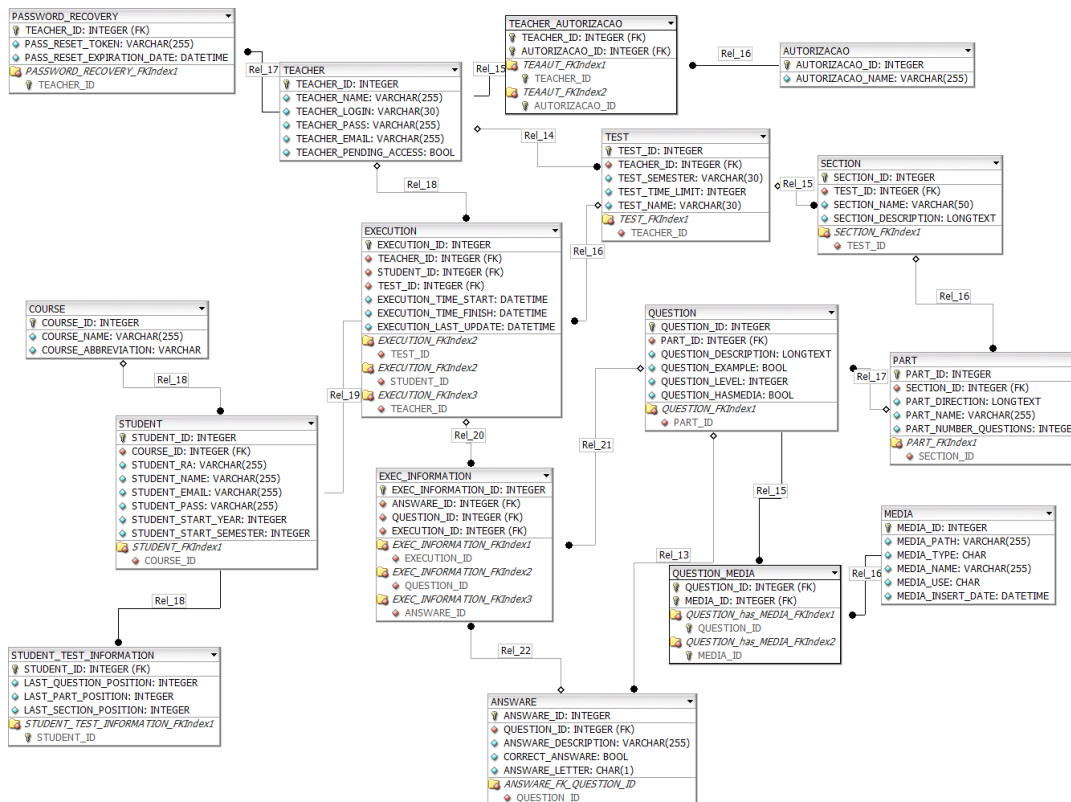
Fonte: Autor, 2017.

Ao observarmos a Figura 11 veremos que todas as comunicações entre a *Visão* e o *Web Service* são feitas via rotas REST onde a troca de informações é feita com os dados no formato de JSON.

3.1.1. Banco de Dados

A Figura 12 exibe o projeto de banco de dados desenvolvido para o EvaSystem, conforme necessidade apresentada durante o levantamento de requisitos.

Figura 12 - Modelo Entidade Relacionamento



Fonte: Autor, 2017.

O modelo é composto por 16(dezesseis) tabelas sendo 7(sete) tabelas diretamente relacionadas com as provas e as demais tabelas são utilizadas para registro dos dados das provas realizadas e informações do Professor e Aluno.

O banco de dados foi projetado de acordo com as características que a prova E.V.A possui atualmente, com isso existe relacionamentos uma para muitos em todas as tabelas direcionadas para o armazenamento da prova, sendo elas, *Test*, *Section*, *Part*, *Question* e *Answer*, onde.

3.1.2. Persistência

Para evitar qualquer inconsistência e ainda garantir agilidade no momento de persistir os dados no banco de dados, foram utilizados dois *frameworks* nas classes responsáveis pela persistência, Hibernate e Spring Data.

A biblioteca Hibernate foi aplicada para realizar o mapeamento das colunas do banco de dados com as classes que representam as Entidades do Modelo no padrão MVC, esta biblioteca é responsável também pela conexão com o banco de dados e pela

validação dos dados mapeados, garantindo que o tipo de dados criado nas classes Java é o mesmo esperado pelo banco de dados. A Figura 13 nos mostra como foi realizado o mapeamento comparando a tabela e a Entidade da classe *Course*.

Figura 13 - Representação da Classe Course e da Tabela COURSE Mapeada

```
CREATE TABLE COURSE (
  COURSE_ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  COURSE_NAME VARCHAR(255) NOT NULL,
  COURSE_ABBREVIATION VARCHAR(10) NULL,
  PRIMARY KEY(COURSE_ID)
);
```

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name = "course_id", unique = true, nullable = false)
private Integer courseId;

@Column(name = "course_name", unique = true, nullable = false)
private String courseName;

@Column(name = "course_abbreviation", unique = true, nullable = false)
private String courseAbbreviation;

@JsonIgnore
@OneToMany(fetch = FetchType.EAGER, mappedBy = "course")
private List<Student> student;
```

Fonte: Autor, 2017.

Os atributos em Java foram fielmente mapeados quanto ao tipo de dados do banco, como podemos observar na Figura 13 os atributos com a anotação *@Column* representam a respectiva coluna do banco de dados, esta anotação também possui seus próprios atributos como, *name*, *unique* e *nullable*, que indicam o nome da coluna, valores repetidos e nulos respectivamente (HIBERNATE ORM, 2017).

Após o *Web Service* (EvaSystem) ser iniciado o Hibernate faz a conexão com o banco de dados e então valida se o tipo de dado está corretamente mapeado. A Figura 14 exibe como a conexão é configurada para que o Hibernate a faça com sucesso.

Figura 14 - Persistence.xml

```
<properties>
  <property name="hibernate.hbm2ddl.auto" value="validate" />
  <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect" />
  <property name="hibernate.show_sql" value="true" />
  <property name="hibernate.format_sql" value="true" />
  <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
  <property name="javax.persistence.jdbc.user" value="root" />
  <property name="javax.persistence.jdbc.password" value="root" />
  <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/evaschema" />
</properties>
```

Fonte: Autor, 2017.

A configuração contida na Figura 14 é utilizada pelo Hibernate para conexão com o banco de dados, nessas configurações que definimos o tipo de linguagem do banco de dados, o usuário e senha do banco bem como o servidor em que ele se encontra, para o funcionamento correto do Hibernate o arquivo *persistence.xml* precisa estar disponível na pasta WEB-INF do projeto.

Em conjunto com o funcionamento do Hibernate foi utilizada biblioteca *Spring Data*, esta biblioteca foi utilizada para realizar as consultas e persistência dos dados.

Nas classes do *Web Service* responsáveis por acesso ao banco de dados foi utilizado JPQL, recurso do *Spring Data* que nos fornece a possibilidade de realizar operações de consulta ou atualização dos dados utilizando *query* como *string* e que utiliza atributos do objeto a ser atualizado ou consultado. A Figura 15 exibe como foi utilizado os recursos do *Spring Data*.

Figura 15 - Exemplo Repository Spring Data

```

10 @Repository
11 public interface CourseRepository extends CrudRepository<Course, Integer> {
12
13     Course findByCourseName(String courseName);
14
15     Course findByCourseAbbreviation(String abbreviation);
16
17     @Modifying
18     @Query("UPDATE Course c SET c.courseName=?1 WHERE c.courseId=?2")
19     int update(String courseName, Integer courseId);
20 }

```

Fonte: Autor, 2017.

A Figura 15 possui duas implementações diferentes dos recursos oferecidos pela biblioteca.

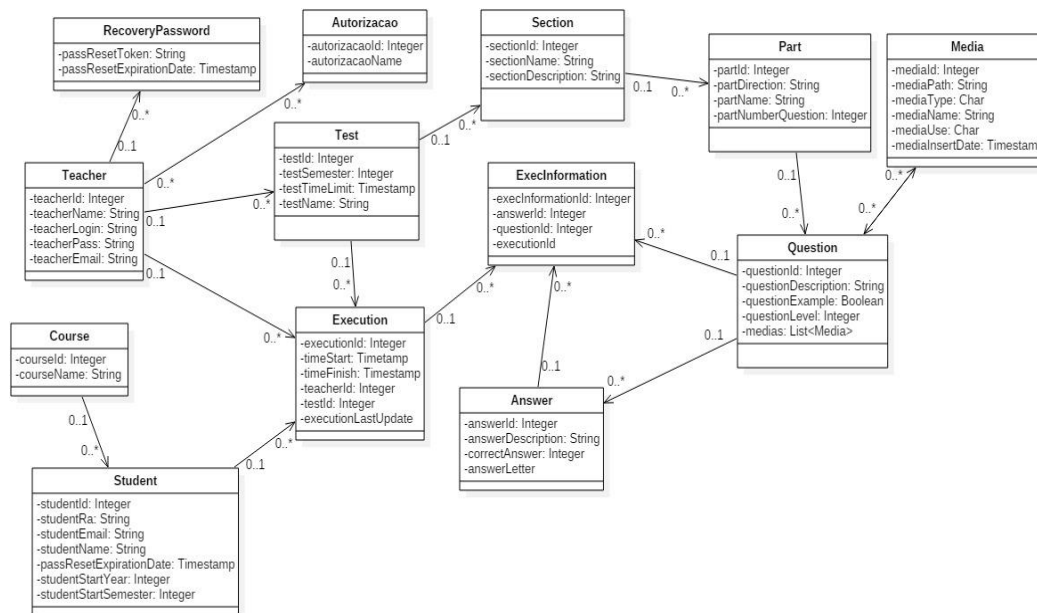
As linhas 13 e 15 são utilizadas para a consulta no banco de dados com base nos atributos *CourseName* e *CourseAbbreviation* respectivamente, ao utilizarmos dessa forma o *Spring Data* se encarregará de realizar a consulta no banco de dados e trará todos os dados que tiverem valores idênticos aos solicitados.

Nesse mesmo trecho de código temos uma implementação diferente, na linha 18 temos a anotação *@Query* localizada logo acima do método *update*, essa anotação possui ainda uma *String* dizendo como a ação deve ser realizada, ao contrário da implementação feita nas linhas 13 e 15 que realiza ação de forma padrão.

3.1.3. Web Service

Na Figura 16 veremos o Diagrama de Classe das Entidades do Sistema que foi desenvolvido.

Figura 16 - Diagrama de Classe das Entidades do Sistema



Fonte: Autor, 2017.

A Figura 16 nos permite identificar as classes que compõe as entidades persistentes do sistema.

Para os casos de relacionamento muitos para muitos o *framework* Hibernate possui anotações específicas que devemos inserir nos atributos Java que irão compor o relacionamento, a Figura 17 nos mostra como foi feito o relacionamento usando as anotações do Hibernate.

Figura 17 - Anotação N:N no Hibernate

```

@ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
@JoinTable(name = "question_media", catalog = "evaschema", joinColumns = {
    @JoinColumn(name = "QUESTION_ID", nullable = false, updatable = false)},
    inverseJoinColumns = { @JoinColumn(name = "MEDIA_ID",
        nullable = false, updatable = false) })
List<Media> medias;
  
```

Fonte: Autor, 2017.

As anotações *@JoinTable* e *@JoinColumn* indicam ao Hibernate que as tabelas *Question* e *Media* irão se relacionar por meio das colunas *QUESTION_ID* e *MEDIA_ID*.

As duas classes envolvidas no relacionamento possuem as anotações *@Entity* que indica a tabela que aquela classe representa com isso o framework consegue mapear os relacionamentos corretamente.

Na camada *Web Service* é onde estará disponível os serviços REST à serem consumidos pela camada de visão.

A Figura 18 nos mostra como um método responsável pela a rota é configurado.

Figura 18 - Exemplo Rota

```
@PreAuthorize("hasRole('ROLE_TEACHER')")
@RequestMapping(value = "/addnewtest",
    method = RequestMethod.POST,
    consumes = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Test> addNewTest(@RequestBody Test test) {

    test = testService.save(test);
    return new ResponseEntity<Test>(test, HttpStatus.OK);
}
```

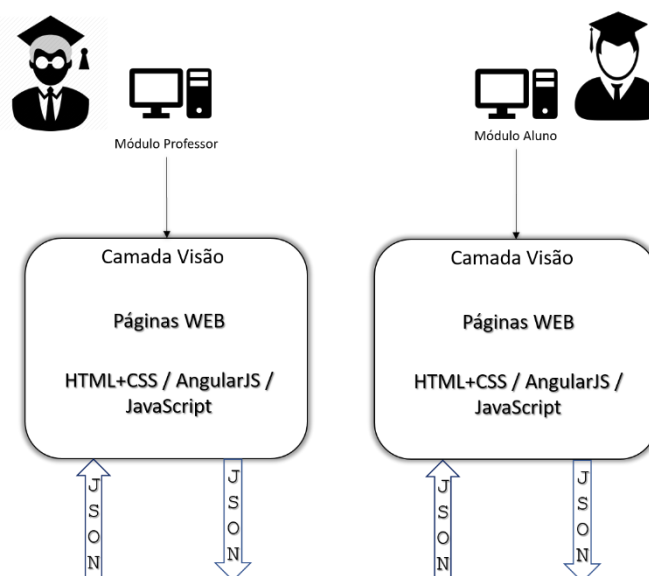
Fonte: Autor, 2017.

Cada rota que desejamos acessar possui seu respectivo método na classe Java, o tipo de dados a ser consumido e/ou produzido é anotado utilizando as anotações *consumes* e *produces* respectivamente.

3.1.4. Visão

Como descrito no capítulo 2 o software é composto por dois módulos web que serão utilizados pelo Aluno e pelo Professor, conforme mostra a Figura 19.

Figura 19 - Arquitetura dos Módulos Professor e Aluno



Fonte: Autor, 2017.

A Camada Visão é a camada responsável por apresentar, de forma amigável, ao usuário todo o processamento realizado após uma requisição feita.

Os módulos do Professor e do Aluno foram desenvolvidos utilizando as linguagens de programação HTML, CSS, *AngularJS* e *JavaScript* e a comunicação com a camada *Web Service* é feita utilizando o padrão de dados JSON que por sua vez é tratado pela camada de *Web Service*,

Após devolver o formato JSON para a visão, esse objeto é então tratado nesta camada e exibido de forma resumida em uma tabela, para que o Professor confira se todos os estudantes foram corretamente importados.

Os dados em JSON são gerados através dos valores dos campos preenchidos na tela, foram desenvolvidas funções utilizando o *framework AngularJS* para obtenção desses dados e então a geração dos JSON para posteriormente serem enviados via requisições para o *Web Service*, a Figura 20 exibe a função que obtém todos os dados preenchidos na tela e enviá-los para a camada de serviço (JSON, 2017).

Figura 20 - Obtenção dos Dados da Visão

```
var test = $scope.test;
```

Fonte: Autor, 2017.

A palavra reservada `$scope` do angular é utilizada para obtenção de dados da visão e esses valores são atribuídos, em forma de JSON, à variável `test`. Após receber os dados da tela os dados são então enviados via requisição REST para o *Web Service*.

A Figura 21 exibe como as requisições são preparadas para o envio.

Figura 21 - Requisição REST

```
url: "http://" + getServerAndPort() + "/evasystem/test/addnewtest",  
method: "POST",  
data: test,  
contentType: "application/json",
```

Fonte: Autor, 2017.

Ao observarmos a Figura 21 vemos que a requisição possui 4(quatro) atributos principais, *url*, *method*, *data* e *contentType*.

No campo *url* indicamos qual a rota REST será utilizada para o envio dos dados.

O campo *method* indica qual é a forma de comunicação com aquela rota, podemos ter rotas do tipo *POST*, *GET*, *PUT* e *DELETE*.

Os campos *data* e *contentType* indicam os dados que serão enviados no nosso caso o objeto *test* e o formato que estes dados estão respectivamente.

Os dados no formato JSON são então recebidos e convertidos em objeto java que posteriormente será utilizado para uma operação no banco de dados.

3.2. Serviços

Conforme descrito na seção 3.1.2 a camada responsável por exibir os dados faz a comunicação com o *Web Service* utilizando JSON.

Para que essa comunicação seja possível a camada de *Web Service* provê serviços, conhecidos também como rotas REST, por onde os dados são enviados e recebidos.

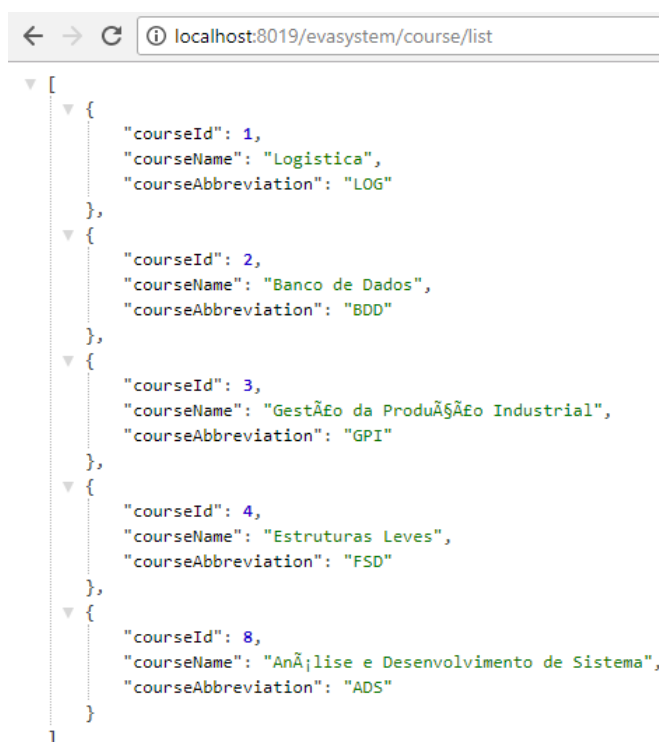
Na seção 3.1.4 citamos os quatro tipos de rotas REST existentes, no *Web Service* (EvaSystem) foram utilizadas as rotas POST e GET.

3.1.1. Serviços Tipo GET

Rotas do tipo GET são constantemente utilizadas pelos navegadores quando o mesmo precisa obter dados vindo do servidor.

Esse tipo de rota é utilizado apenas para obter dados do servidor através da URL fornecida pelo *Web Service*. A Figura 22 nos mostra o retorno de uma requisição GET.

Figura 22 - Retorno Requisição



Fonte: Autor, 2017.

A rota contida na Figura 22 retornou a lista de cursos cadastrado no sistema, conforme já explicado os dados retornaram no formato JSON de dados, diferentemente da rota POST na rota GET não é obrigatório parâmetros para obter os dados desejados (LECHETA, 2017)

3.1.2. Serviços Tipo POST

Requisições POST são frequentemente utilizadas para inserir novos registros no banco de dados e/ou para atualizar dados de algum registro.

Quando criamos requisições POST obrigatoriamente precisamos passar dados como parâmetro da rota, no caso do EvaSystem foi utilizado a anotação `@RequestBody` do Spring Framework para conversão dos objetos JSON para os respectivos objetos Java, conforme Figura 23.

Figura 23 - Spring @RequestBody

```

52 @PreAuthorize("hasRole('ROLE_TEACHER')")
53 @RequestMapping(value = "/add", method = RequestMethod.POST, consumes = MediaType.APPLICATION_JSON_VALUE)
54 public ResponseEntity<Course> add(@RequestBody Course course) {
55
56     course = courseService.save(course);
57     if (course == null) {
58         return new ResponseEntity<Course>(HttpStatus.FOUND);
59     }
60     return new ResponseEntity<Course>(course, HttpStatus.OK);
61
62 }

```

Fonte: Autor, 2017.

Conforme explicado no capítulo 3.1.4 as requisições são feitas por meio de funções do AngularJS onde são passados os dados JSON por meio do parâmetro `'data'` do método POST, no exemplo da Figura 23 o dado vindo da requisição é convertido para o objeto `course` que então é salvo no banco de dados.

Na linha 56 temos a função responsável por inserir o novo curso no banco de dados, após o processo ser concluído esse curso salvo é retornado para camada visão em formato JSON que por sua vez trata os dados para serem exibidos na tela se necessário.

3.3. Segurança

Para garantir a segurança das informações foi utilizado a biblioteca Spring Security contida no Spring Framework, junto com ela foi utilizada a biblioteca JWT responsável por gerar os *tokens* de acesso após a autenticação ter sido feita com sucesso pela biblioteca do Spring.

3.3.1. JSON Web Token

Ao realizar o *login* no software a camada de *Web Service* realiza a autenticação do usuário utilizando a biblioteca JWT (*Java Web Token*) e em seguida

atribui à esse usuário um token válido por 120 minutos indicando que a autenticação foi feita com sucesso.

3.3.2. Spring Security

A biblioteca JWT cuida da autenticação no sistema, a segurança implementada nas rotas contidas no *Web Service* foi inserida utilizando uma anotação do Spring como podemos ver na Figura 24.

Figura 24 - Anotação Segurança

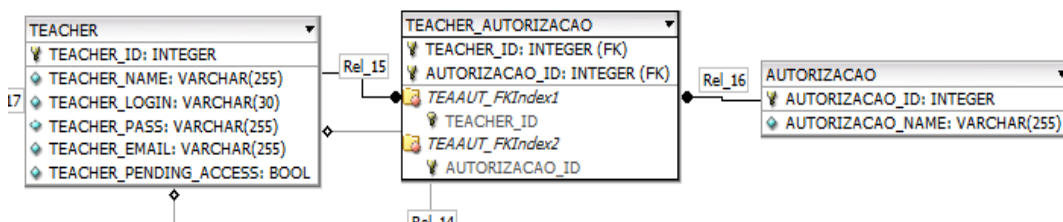
```
@PreAuthorize("hasRole('ROLE_TEACHER')")
```

Fonte: Autor, 2017.

A anotação `@PreAuthorize` indica que o acesso só poderá ser realizado pelo Professor que possuir aquela permissão, no nosso caso, 'ROLE_TEACHER'.

Para a associação dos papéis aos usuários foram criadas duas tabelas conforme podemos observar na Figura 25.

Figura 25 - Tabelas de Segurança



Fonte: Autor, 2017.

O relacionamento muitos para muitos entre as tabelas **TEACHER** e **AUTORIZACAO** gerou a tabela **TEACHER_AUTORIZACAO**, quando um Professor solicita acesso ao software ele automaticamente recebe a Role de Professor.

Após todo o processo de verificação de permissões é terminado o software então verifica se o *token* utilizado ainda está dentro do prazo de validade

3.3.3. Segurança Aplicada em Senhas

Para garantir maior segurança ao usuário, além da segurança das rotas e o uso de token após a autenticação, as senhas dos Alunos e dos Professores serão inseridas no banco de dados utilizando a criptografia BCrypt.

Criada por Niels Provos e David Mazieres, BCrypt é uma criptografia implementada com base na criptografia de Bruce Blowfish, a senha é criptografada utilizando um algoritmo onde a senha que se deseja criptografar é quebrada em diversas partes pelo algoritmo (DOCS SPRING, 2017).

A Figura 26 nos mostra a senha após o processamento da criptografia.

Figura 26 - Senha Criptografada com BCrypt

teacher_id	teacher_pass
1	\$2a\$10\$olSw015zffGvmnQel8FP9e7kvBlug15d2AWEyiHOtknnXN7EpWTj2
2	\$2a\$10\$PSTekhRnHIdus6JbcFFZO0xtD6QwjJYWGe9pbeupLbHsQLFC5cx2
3	\$2a\$10\$B0Bl.x81r0P3wTSsxAScdeqIS3kmxROXA1PzdxlSxkrms.9B0pv6C
4	\$2a\$10\$Y0c79gVfhqM8hYOyyIXVTujgIIA/cW7mby9105Hr8.JvmDqqsD7Zy
5	\$2a\$10\$2n7Otyr5apT0fNlxaZosne2AtP56Ob/TzRYGdhGEHidHPZtoCvMoq

Fonte:

Autor, 2017.

Após o JSON com os dados do Professor ou do Aluno chegar à camada de *Web Service* os dados são convertidos para os respectivos objetos e então a senha é criptografada como podemos ver na Figura 27.

Figura 27 - Método de Criptografia

```
public String encode(String password) {
    return new BCryptPasswordEncoder().encode(password);
}
```

Fonte: Autor, 2017.

O método acima é utilizado para criptografar senhas dos Alunos e dos Professores, ele recebe apenas a senha como parâmetro e retorna a mesma criptografada.

3.4. Rotas Disponibilizadas pelo *Web Service*

Para ser possível a comunicação entre a camada visão e o *Web Service* é necessário que haja, do lado do *Web Service*, rotas que permitem essa comunicação e

consequentemente a transferência de informações. Conforme explicado na seção 3.2, no *Web Service* EvaSystem foram utilizadas rotas dos tipos POST e GET.

3.4.1. Rotas POST

A Tabela 11 exibe todos os serviços do tipo POST que a camada visão dos módulos do Professor e Aluno utilizam.

Tabela 11 - Mapeamento Serviços POST

Base URL	Complemento	Parâmetro	Retorno	Segurança
/course	/add	Course	Course	Sim
/execinformat ion	/getAllQuestionByStudent	Student	List<Question>	Sim
/execinformat ion	/save	ExecInformatio n		Sim
/execinformat ion	/getExecInformationBySt udent	Student	List<ExecInformat ion>	Sim
/execution	/test/getGeneratedTest	Student	Test	Não
/execution	/finish	Execution	Execution	Não
/execution	/getExecutionListByTest	Test	List<Execution>	Sim
/execution	/start	Execution	Execution	Não
/login	/student	Student	Student	Não
/login	/teacher	Teacher	Teacher	Não
/media	/addnewmedia	Media	Media	Sim
/media	/question	Question	List<Media>	Não
/part	/addnewpart	Part	Part	Sim
/part	/saveeditedpart	Part	Part	Sim
/part	/getPartStatistics	Part	List<Question>	Sim
/part	/section	Section	List<Part>	Sim
/part	/getPartByQuestion	Question	Part	Sim
/question	/addnewquestion	Question	Question	Sim
/question	/partquestion	Part	List<Question>	Sim
/question	/part	Part	List<Question>	Sim
/password	/recovery	RecoveryPassw ord	RecoveryPassword	Não
/section	/addnewsection	Section	Section	Sim
/section	/getSectionByPart	Part	Section	Sim
/section	/test	Test	List<Section>	Sim
/student	/addstudentlist	MediaDTO	List<Student>	Sim
/teacher	/recovery	Teacher	Teacher	Não
/teacher	/changepassword	Teacher	Teacher	Sim
/teacher	/list	Teacher	List< Teacher>	Sim
/teacher	/addnewteacher	Teacher	Teacher	Não
/teacher	/approvependingteacher	Teacher	Teacher	Sim

/test	/addstudentaslist	List<Execution>	List<Execution>	Sim
/test	/addnewtest	Test	Test	Sim
/test	/student	Student	Test	Não
/test	/getTestByExecution	Execution	Test	Sim

Fonte: Autor, 2017.

Como podemos observar foram criadas 34 rotas POST, e 24 rotas só podem ser acessadas pelo Professor.

3.4.2. Rotas GET

Na Tabela 12 estão mapeadas as rotas tipo GET.

Tabela 12 - Mapeamento Serviços GET

Base URL	Complemento	Parâmetro	Retorno	Segurança
/answare	/list		List<Answare>	Sim
/answare	/ {id}	Integer	Answare	Sim
/answare	/question	Question	List<Answare>	Sim
/course	/list		List<Course>	Sim
/course	/ {id}		Course	Sim
/execinform ation	/list		List<ExecInformat ion>	Sim
/execution	/list		List<Execution>	Sim
/media	/list		List<Media>	Sim
/part	/list		List<Part>	Sim
/question	/list		List<Question>	Sim
/report	/getReportStudent/{idStu dent}/{idTest}/{datestart}/{d ateend}/{testName}	Integer, Integer, String, String ,String	List<Student>	Não
/report	/getReportCourse/{testId}/ {courseId}/{datestart}/{dat eend}/{testName}	Integer, Integer, String, String, String	List<Student>	Não
/section	/list		List<Section>	Sim
/student	/list		List<Student>	Sim
/teacher	/pendingteacher		List<Teacher>	Sim
/test	/list		List<Test>	Sim

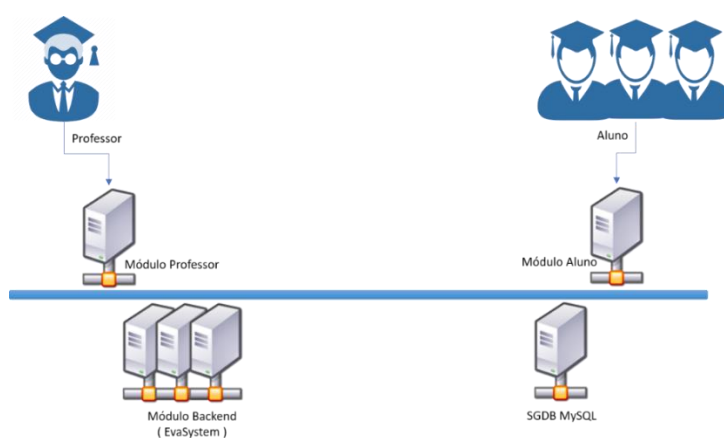
Fonte: Autor, 2017.

Alguns serviços GET possuem parâmetros para customização dos dados a serem buscados, outros serviços retornam todos os registros que estão no banco de dados.

3.5. Infraestrutura para Implantação

Para implantação do sistema é necessário que tenha disponível um servidor para hospedar todos os módulos e um servidor onde está o banco de dados, a Figura 28 exibe como a arquitetura ficou distribuída.

Figura 28 - Arquitetura Geral



Fonte: Autor, 2017.

3.5.1. Módulo Backend

Como já descrito no Capítulo 3, o *Web Service* foi totalmente desenvolvido em Java e para que o mesmo disponibilize os serviços faz-se necessário que o servidor possua o software Apache Tomcat instalado.

O Apache Tomcat será o servidor de aplicação onde o EvaSystem será executado. A Tabela 13 exibe as configurações necessárias para o servidor onde será hospedado o *Web Service*.

Tabela 13 - Configurações do Servidor do Módulo Backend

Especificação de <i>Hardware</i>	
Componente	Especificação
Memória	2GB
Disco Rígido	2 GB disponíveis

Processador	2.5 <i>Giga Hertz</i> ou superior
Especificação de <i>Software</i>	
Software	Versão
Java	8
Aplicação Servlet	Qualquer versão compatível com Java 8

Fonte: Autor, 2017.

Os componentes citados acima fazem parte dos requisitos mínimos para a execução do *Web Service*.

3.5.2. Servidor de Banco de Dados

O servidor onde o EvaSystem irá ser executado deverá possuir o MySQL instalado. O banco de dados a ser utilizado pelo *Web Service* deverá estar criado e os dados de acesso a esse banco deverão estar configurados no arquivo *persistence.xml* conforme descrito na seção 3.1.2. A Tabela 14 exibe as configurações necessárias para o servidor onde será hospedado Banco de Dados

Tabela 14 - Configurações do Servidor de Banco de Dados

Especificação de <i>Hardware</i>	
Componente	Especificação
Memória	2 GB
Disco Rígido	5 GB disponíveis
Processador	2.5 <i>Giga Hertz</i> ou superior
Especificação de <i>Software</i>	
Software	Versão
Sistema de Gerenciamento de Banco de Dados MySQL	5.0.8

Fonte: Autor, 2017.

Os componentes citados acima fazem parte dos requisitos mínimos para a execução do Banco de Dados.

3.5.3. Módulos Frontend

Os módulos *AdminTeacher* e *AdminStudent* também precisam ser implantados no Apache Tomcat e inicializados.

Após o processo de inicialização ser concluída o acesso aos módulos deverá, para melhor funcionamento, ser feito pelo navegador web Google Chrome. A Tabela 15 exibe as configurações necessárias para o servidor onde será hospedado Banco de Dados

Tabela 15 – Configurações do Servidor dos Módulos Frontend

Especificação de <i>Hardware</i>	
Componente	Especificação
Memória	2 GB
Processador	2.5 <i>Giga Hertz</i> ou superior
Especificação de <i>Software</i>	
Software	Versão
Google Chrome	62 ou superior

Fonte: Autor, 2017.

Os componentes citados acima fazem parte dos requisitos mínimos para a execução dos módulos do Aluno e do Professor.

4. RESULTADOS

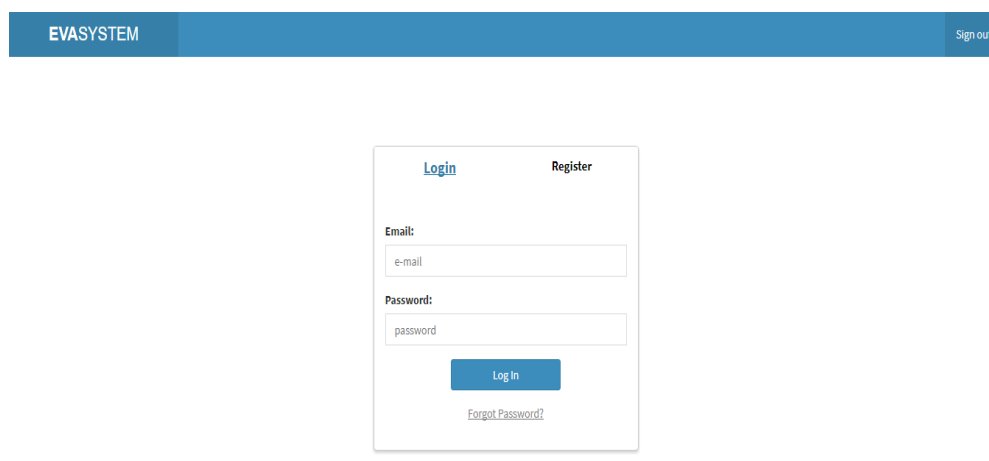
Neste capítulo serão apresentados os resultados dos experimentos realizados bem como as soluções implementadas para correções dos erros e propostas de melhorias.

4.1. Módulo Professor

No módulo do Professor será possível cadastrar novas provas, questões, poderá importar a lista de Alunos que farão a prova, iniciar uma nova prova dentre outras ações.

A Figura 29 exibe a tela de *login* do Professor.

Figura 29 - Tela *Login* Professor



A imagem mostra a interface de login do Professor no sistema EVASystem. No topo, há uma barra azul com o texto "EVASYSTEM" à esquerda e "Sign out" à direita. Centralizado na tela, há um formulário branco com o título "Login" e "Register". O formulário contém campos para "Email:" (com o placeholder "e-mail") e "Password:" (com o placeholder "password"). Abaixo dos campos, há um botão azul "Log In" e um link "Forgot Password?".

Fonte: Autor, 2017.

O Professor terá acesso aos recursos do menu após efetuar o *login* no módulo.

Cada menu possui a função de cadastrar e/ou editar os componentes que compõem a prova, como mostra a Figura 30.

Figura 30 - Cadastro de Prova

Fonte: Autor, 2017.

A Figura 30 nos mostra o formulário de cadastro de nova prova, onde conforme descrito no capítulo 2, o Professor deverá preencher todos os campos antes de salvar a prova.

Na opção ‘*Start New Test*’ o Professor poderá importar um arquivo do tipo xls com a lista de estudantes que farão a prova naquele dia.

Todos os campos deverão ser do tipo texto e ao serem importados serão convertidos pela camada de serviço em objeto Java e retornados à camada de Visão por meio do tipo JSON, conforme exibe Figura 31.

Figura 31 - Lista de Estudantes

#	RA	Name	E-mail
1	1234567890	Helen	helen@fatec.sp.gov.br
2	9485678932	Gabriel	gabriel@fatec.sp.gov.br
3	9485678933	Daniel	daniel@fatec.sp.gov.br
4	9485678934	Pedro	pedro@fatec.sp.gov.br
5	9485678935	Felipe	felipe@fatec.sp.gov.br
6	9485678936	Marcos	Marcos@fatec.sp.gov.br


Save Students Cancel

Fonte: Autor, 2017.

Professores que ainda não possuem acesso ao *EvaSystem* poderão se cadastrar através da aba *Register* conforme exibe a Figura 29.

Após solicitar o cadastro o acesso ficará pendente conforme podemos observar na Figura 32.

Figura 32 - Acessos Pendentes

Name	Login	Email	Approve	Reprove
teste2	teste2	teste2@teste.com		
teste3	teste3	teste3@teste.com		

Fonte: Autor, 2017.

Qualquer Professor conseguirá aprovar ou reprovar as solicitações, caso seja reprovado, o cadastro do Professor(a) pendente será excluído necessitando então de um novo cadastro caso necessário.

4.2. Módulo Aluno

O Aluno terá acesso ao módulo somente em dias que ele irá realizar a prova, seu acesso é liberado automaticamente a partir do momento que o professor inicia um novo teste no módulo do Professor. A Figura 33 exibe a tela de *login* do módulo Aluno.

Figura 33 - Tela Login Aluno

Student Login

Email:

Password:

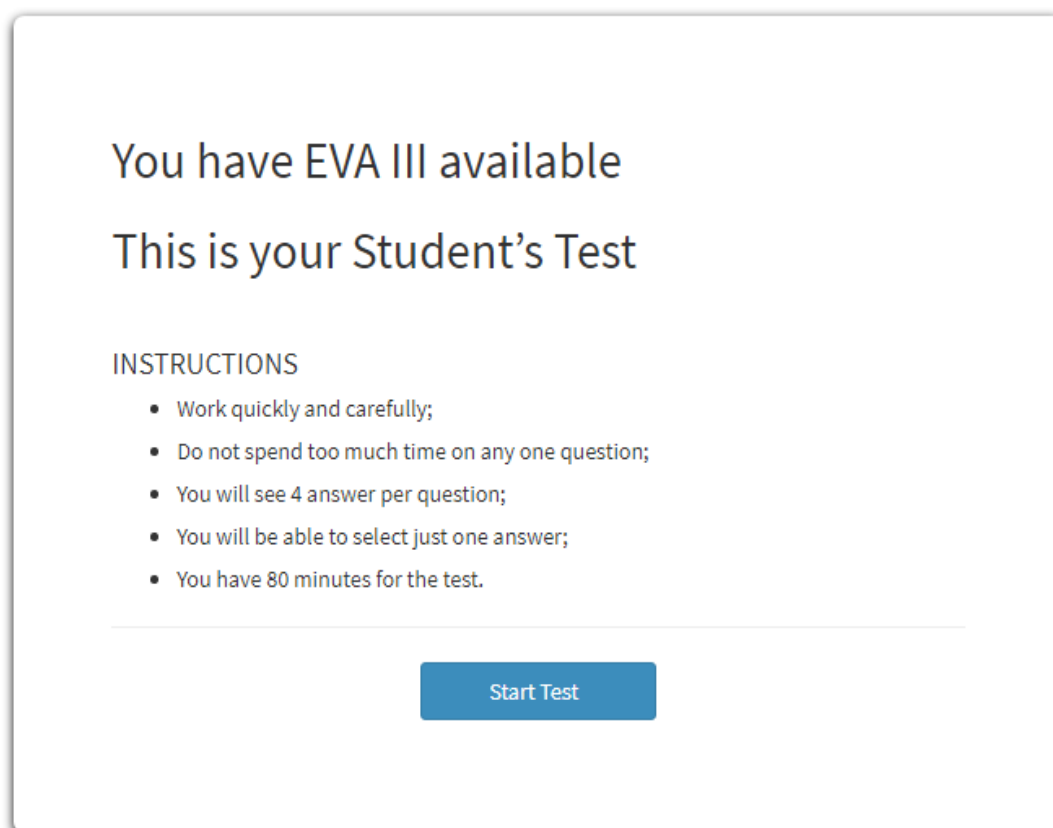
LOG IN

Fonte: Autor, 2017.

Na Figura 33 podemos observar que o Aluno utilizará o seu e-mail institucional e como senha utilizará seu RA. Ambos os dados serão previamente cadastrados pelo Professor.

Após realizar o *login* o estudante verá o aviso conforme mostra a Figura 34 sobre a prova que ele terá que realizar.

Figura 34 - Aviso Prova Disponível

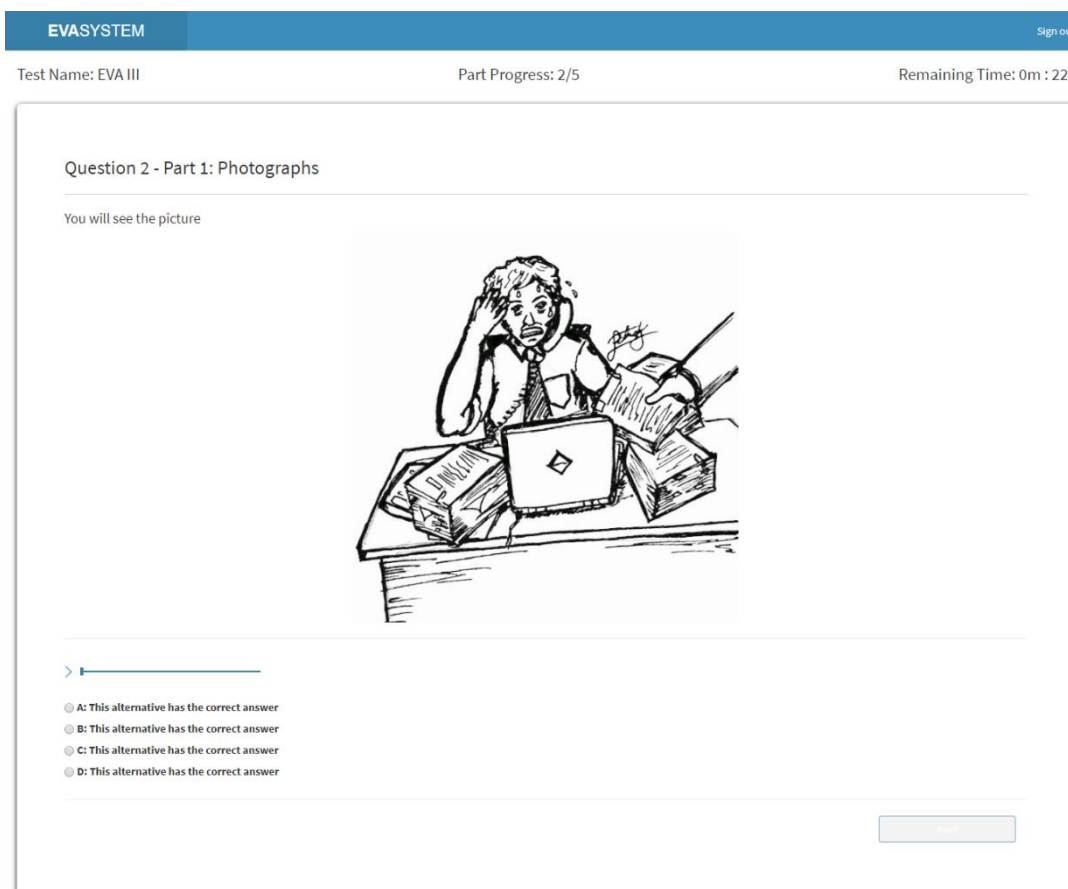


Fonte: Autor, 2017.

O botão '*Start Test*' levará o Aluno para a página onde ele poderá responder as questões da prova, a página possui informações da prova a sendo realizada, tempo de prova e o progresso do estudante segmentado por partes que compõem a prova.

Após o Aluno clicar para iniciar o a prova uma nova página será carregada com os dados da questão de exemplo, após ler e entender o que deve ser feito ele poderá prosseguir clicando em *Next*. A Figura 35 nos exemplifica como é a tela onde o Aluno realizará a prova,

Figura 35 - Tela Prova



Fonte: Autor, 2017.

A Figura 35 nos mostra que a tela de realização da prova possui logo em seu cabeçalho informações da prova sendo realizada, do progresso do Aluno e também o tempo restante para realizar a prova. Logo abaixo podemos observar informações da questão sendo realizada e qual parte aquela questão está associada.

O E.V.A possui questões com imagens e áudios, a questão utilizada acima como exemplo possui os dois tipos de mídia, o Aluno poderá então observar a imagem, escutar o áudio e então escolher a melhor alternativa que se aplica àquela questão.

Após o tempo atribuído para realização da prova terminar o módulo do Aluno enviará uma requisição ao *Web Service* para que a prova do Aluno seja considerada como finalizada, mesmo que o Aluno não tenha terminado de responder todas as questões.

4.3. Experimentos

Após o término do desenvolvimento do *Web Service* e dos módulos AdminTeacher e AdminStudent, foram feitos experimentos em busca de possíveis inconsistências ou pontos de melhoria.

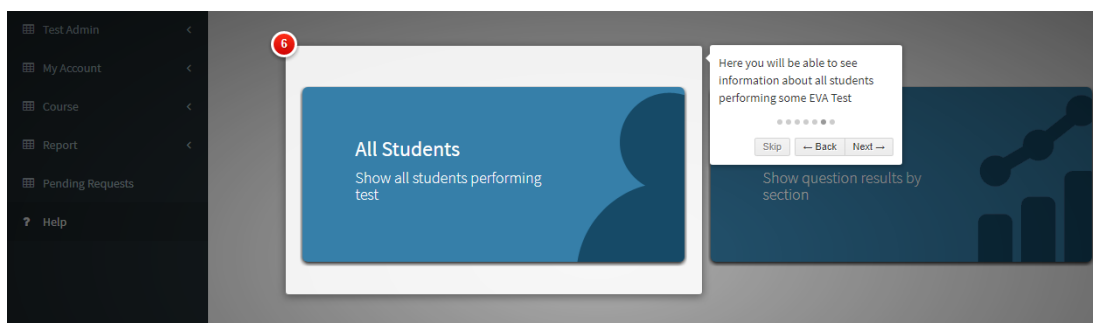
4.3.1. AdminTeacher – Experimento 1

O módulo do Professor foi disponibilizado para testes no dia 08 de outubro em um servidor *Amazon Web Service* (AWS), o servidor foi contratado especialmente para esse experimento e todo o sistema foi utilizado a partir deste servidor.

Os dados para acesso foram encaminhados à Professora para que houvesse um primeiro contato com o módulo, nesse primeiro experimento o objetivo era verificar uma possível dificuldade no uso do módulo, após cerca de 15 dias utilizando apenas para cadastrar dados não houve nenhum erro a ser apontado pela professora, porém surgiram dúvidas de como os menus são divididos e o que pode ser feito em cada opção do menu.

Para solucionar esse problema foi implementado, utilizando a biblioteca *HelpJS*, um roteiro de ajuda que é executado assim que é realizado o primeiro *login* no módulo do professor, conforme exibe a Figura 36.

Figura 36 - Ajuda com HelpJS



Fonte: Autor, 2017.

A biblioteca será executada ao ser realizado o *login* ou poderá ser executada pelo professor utilizando o menu *Help*.

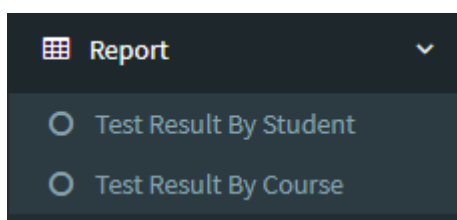
4.3.2. AdminTeacher – Experimento 2

No dia 30 de outubro após a prova E.V.A III ter sido aplicada a dois ex-Alunos da Fatec o módulo do Professor foi testado visando o entendimento dos resultados obtidos pelos Alunos.

O Professor possui 3 formas principais de conferir a performance dos Alunos na prova aplicada. em duas formas é possível indicar exatamente a questão que o Aluno erro ou acertou,

A Figura 37 exibe o menu que possibilita a extração de relatórios por Aluno ou por curso.

Figura 37 - Menu Report



Fonte: Autor, 2017.

Em ambas as opções o Professor deve preencher os campos *Test Name*, *End Date* e *Start Date*, a Figura 38 exibe como é o formulário para extração do relatório por Aluno.

Figura 38 - Relatório por Aluno

Test Result By Student

Test Name:

Student Name:

Select the start and end date:

Start Date:

End Date:

Fonte: Autor, 2017.

Após preencher todos os campos exibidos na Figura 38 o Professor poderá clicar no botão *Show Report* que então será apresentado a ele, em um documento no formato PDF, o resultado daquele estudante na prova selecionada, a Figura 39 exibe um exemplo de como será o relatório.

Figura 39 - Relatório PDF Aluno

RESULTS OF STUDENT					
Test: EVA III	Teacher: Helen Reis	Student: Erick	Course: Banco de Dados		
Section Name	Part Name	Question Description	Student Answer	Correct Answer	Result
Section I: Listening	Part 1: Photographs	You will see the picture	C	D	Failed
Section I: Listening	Part 1: Photographs	You will see the picture	C	C	Success
Section I: Listening	Part 1: Photographs	You will see the picture	B	A	Failed
Section I: Listening	Part 1: Photographs	You will see the picture	A	A	Success
Section I: Listening	Part 2: Question-Response	Listen to the audio and select the correct alternative	C	C	Success
Section I: Listening	Part 2: Question-Response	Listen to the audio and select the correct alternative	B	B	Success
Section I: Listening	Part 2: Question-Response	Listen to the audio and select the correct alternative	D	D	Success
Section I: Listening	Part 2: Question-Response	Listen to the audio and select the correct alternative	B	B	Success
Section I: Listening	Part 2: Question-Response	Listen to the audio and select the correct alternative	A	A	Success

Fonte: Autor, 2017.

Se o Professor desejar acompanhar o desempenho do Aluno ele poderá consultar os resultados em uma segunda opção que fica na página inicial, logo após o acesso na aplicação na opção *All Students*, ao selecionar a opção informada, o Professor será redirecionado para a página onde terá a lista de Aluno escolhido para realizar a prova e o respectivo status, conforme consta na Figura 40.

Figura 40 – Lista Alunos Realizando Prova

All Students				
Choose a test to see the progress of all students:				
EVA III				
#	RA	Name	E-mail	Test Status
1	1234	Bene	benedito.vitorino@fatec.sp.gov.br	Finished
2	1234567890	Helen	helen@fatec.sp.gov.br	In Progress
3	5678	Erick	erickcpv2005@gmail.com	Finished

Fonte: Autor, 2017.

Na Figura 40 temos 3 Alunos que foram escolhidos para realizar a prova, a coluna *Test Status* pode apresentar os status, *Finished*, *In Progress* e *Not Started*.

Dentre os Alunos listados o Professor poderá escolher apenas um por vez para acompanhar o progresso na prova. Após escolher o Aluno a página com os resultados obtidos até o momento será exibida, a Figura 41 exibe o resultado da tela do Aluno selecionado na página anterior.

Figura 41 - Detalhe Progresso Aluno

Student Test Detail

[Filter](#)

Question Description	Student Answer	Correct Answer	Result
What is NOT true about the event?	C	C	Success
If the organizers consider Alex's comments, for the next event:	D	D	Success
Which city has the passenger already been to before?	A	B	Failed

[Export Student Results](#)

Partial Results	
Correct	56
Incorrect	20

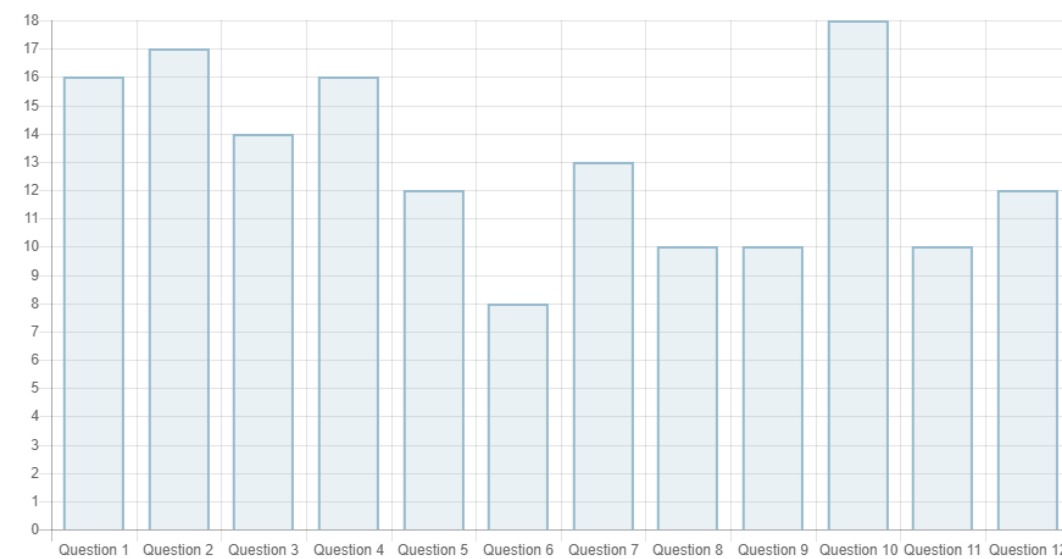
Fonte: Autor, 2017.

Na Figura 41 consta parte do resultado do Aluno, nela podemos observar uma tabela onde possui a descrição da pergunta, a resposta que o Aluno escolheu e a resposta correta daquela pergunta, a última coluna *Result* indica em cores distintas se o Aluno obteve sucesso ou não na questão. Na tabela *Partial Results* é apresentado quantas questões o Aluno acertou e quantas ele errou.

O Professor tem também a opção de exportar o resultado em formato PDF com um arquivo semelhante ao apresentado na Figura 39.

A terceira forma que o Professor tem para consultar o progresso dos Alunos é na forma de gráfico onde será exibido quantos Alunos acertaram as questões que compõem aquela parte.

Após escolher os dados necessário para obter o resultado graficamente o Professor deverá clicar no botão *Show Statistics Information* conforme mostra a Figura 42

Figura 42 - Gráfico Resultados Por Questão

Fonte: Autor, 2017.

O gráfico contido na Figura 42 mostra, no eixo y, a quantidade de Alunos que acertaram determinada questão. A informação apresentada acima consta somente as questões da parte escolhida.

Durante o experimento foram encontradas algumas melhorias a serem trabalhadas, conforme exhibe a Tabela 16.

Tabela 16 - Resultados Experimento 1 Professor

Descrição	Tipo	Correção Realizada
Correções ortográfica da língua inglesa	Erro	Feita a revisão da escrita e corrigido conforme proposto
Gráfico de desempenho disponível mesmo após a prova	Melhoria	Feito todos os ajustes necessários para que o gráfico possa ser visualizado mesmo após o término da prova, foram inseridos os dados de curso e data para a busca realizada seja consistente
Visualização da prova está considerando questões de exemplo.	Erro	Foram feitos os ajustes necessários para que seja exibido somente as questões reais da prova.

Fonte: Autor, 2017.

4.3.3. AdminStudent - Experimento 1

No dia 26 de setembro foi solicitado à 15 Alunos do sexto semestre do curso de banco de dados a realização de uma prova fictícia.

A prova era composta por 10 questões e foi realizada na rede da Fatec de São José dos Campos, na ocasião o *Web Service* e os módulos AdminStudent ficaram hospedados na mesma máquina, com isso o *Web Service* disponibilizou os serviços para serem consumidos.

Foi solicitado aos Alunos que fornecessem os dados citados na seção 4.3.3 para que os mesmos pudessem realizar a prova.

O objetivo principal desse experimento foi achar o maior número de erros/inconsistências possíveis, durante a realização da prova foram encontrados 1 erro e 2 inconsistências no módulo conforme Tabela 17.

Tabela 17 - Resultados Experimento 1 Aluno

Descrição	Tipo	Correção Realizada
Falha ao pausar áudio	Erro	Verificado que ao clicar em <i>Next</i> ocorria uma falha no javascript que não permita que o áudio fosse corretamente parado.
Alternativas não são assinaladas corretamente	Inconsistência	Verificado que o ID do radiobutton estava sendo atribuído de forma inconsistente.
Alteração da rota getGeneratedTest para POST	Inconsistência	Rota alterada de GET para POST e como parâmetro foi adicionado os dados do Aluno.

Fonte: Autor, 2017.

Conforme apresentado na Tabela 17, todos os erros e inconsistências foram corrigidos.

4.3.4. AdminStudent - Experimento 2

No dia 30 de outubro ocorreu o segundo experimento com o EvaSystem, usado não somente para verificar erros na aplicação, mas também para ver como ela

se comporta quando está hospedada em algum servidor externo, diferentemente do Experimento 1, esse experimento foi realizado utilizando uma prova verdadeira do E.V.A, porém com ex-Alunos da Fatec de São José dos Campos que se propuseram a avaliar a aplicação.

Utilizando um servidor externo foi feita implantação do EvaSystem e iniciado todos os módulos e o *Web Service*.

Não ocorreram erros na aplicação, porém notou-se uma certa lentidão para carregamento de perguntas com imagem e áudio, indicando assim que a internet e o servidor de aplicação devem suportar um alto número de requisições ao *Web Service*, consultas ao banco de dados e alto tráfego de dados pela rede.

Apesar de não terem sido identificadas falhas funcionais no sistema, os ex-Alunos identificaram algumas melhorias, conforme registrado na Tabela 18.

Tabela 18 - Resultados Experimento 2 Aluno

Descrição	Tipo	Correção Realizada
Botão play pouco intuitivo	Melhoria	Botão foi alterado para o <i>design</i> conforme a Figura 43 exhibe.
Aumentar tamanho das imagens	Melhoria	As imagens apontadas como pequenas foram alteradas para uma resolução adequada para a leitura do texto contido na imagem

Fonte: Autor, 2017.

Figura 43 - Botão Play



Fonte: Autor, 2017.

O modelo anterior do botão play mostrado na Figura 43 não era preenchido de azul e isso poderia causar dúvidas nos Alunos.

4.3.5. AdminStudent – Experimento 3

Neste experimento o módulo do Aluno foi disponibilizado para 19 Alunos do curso de Tecnologia em Automação Manufatura Digital em um dos laboratórios da Fatec.

Os Alunos foram informados que iriam realizar a prova E.V.A III, durante o experimento foi detectada lentidão em questões que possuem áudio.

Foram feitas análises no código e identificado que o método responsável por tratar o áudio das questões era o motivo da lentidão detectada, essa causa foi detectada após realizar diversos testes de carga no projeto simulando requisições de 19 Alunos ao mesmo tempo, a Figura 44 exibe o tempo de uso em milissegundos da CPU do método responsável pela lentidão.

Figura 44 - Estatísticas Java VisualVM

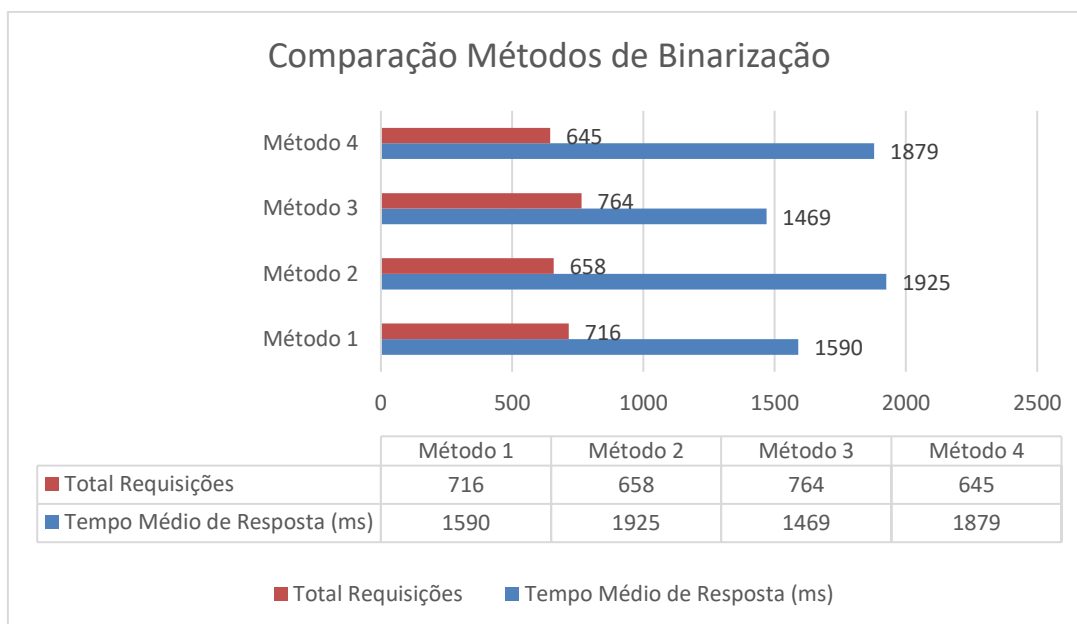
Hot Spots - Method	Self Time ▼	Self Time (CPU)	Total Time	Total Time (CPU)
org.apache.log4j.Category. callAppenders ()	4.008.233 ms (75,1%)	9.874 ms	4.089.920 ms	91.562 ms
org.apache.coyote.http11.InternalInputBuffer. fill ()	901.241 ms (16,9%)	901.241 ms	901.241 ms	901.241 ms
org.apache.tomcat.util.threads.TaskQueue. poll ()	162.139 ms (3%)	0,000 ms	162.139 ms	0,000 ms
org.apache.catalina.core.ContainerBase\$ContainerBackgroundProcessor. run ()	84.433 ms (1,6%)	0,000 ms	87.361 ms	2.927 ms
org.apache.log4j.spi.LocationInfo. <init> ()	65.914 ms (1,2%)	65.914 ms	65.914 ms	65.914 ms
org.apache.commons.io.IOUtils. copyLarge ()	15.690 ms (0,3%)	15.690 ms	15.690 ms	15.690 ms
br.edu.fatec.eva.util.FileFactory. generateBase64FromFile ()	9.741 ms (0,2%)	9.741 ms	39.240 ms	39.240 ms

Fonte: Autor, 2017.

Ao extrair o relatório, conforme consta na Figura 44, foi verificado que o método **generateBase64FromFile** utiliza 9.7ms da CPU em cada vez que é utilizado, esse valor é somando à coluna *Total Time*. Esse valor aumenta exponencialmente conforme aumenta o número de requisições.

Para a solução do problema foram feitos ajustes do método alterando a forma de leitura e tratamento do áudio e então realizados novos testes de carga.

Os novos testes de carga foram realizados com 4 métodos de tratamento de áudio diferentes, foi observado que um dos métodos apresentou melhora de performance em comparação aos demais, conforme exibe a Figura 45.

Figura 45 - Comparativo Métodos de Tratamento de Áudio

Fonte: Autor, 2017.

Ao observarmos a Figura 45 podemos dizer que o tempo médio de resposta ficou em torno de 1,7 segundos. Se isolarmos esse valor pensando em apenas um Aluno realizando as provas não há necessidade de melhora, porém quando temos um cenário com vários Alunos realizando em média 3 requisições por segundo esse valor torna-se alto prejudicando assim o tempo de prova.

Como solução para o problema foi implementado um mapeamento entre as questões escolhidas para a prova e seus respectivos áudios, com isso os áudios serão tratados e então colocados nesse mapeamento já tratado e as requisições do Aluno irão apenas consultar o mapeamento e então devolver para o Aluno o áudio da questão.

Com a nova implementação, o mapeamento será criado assim que uma nova prova for iniciada, a fim de evitar lentidão para iniciar a prova foi utilizado um recurso do *Spring Framework* chamado *DeferredResult* onde processos são executados paralelamente.

5. CONCLUSÃO

Nesse capítulo serão apresentadas as considerações finais do trabalho bem como melhorias propostas a serem realizadas por meio de trabalhos futuros.

5.1. Lições Aprendidas

Após os requisitos terem sido levantados e identificado a necessidade de criação de dois módulos web, foram feitas análises das linguagens de programação voltadas para desenvolvimento web e então foram escolhidas as linguagens que permitissem o desenvolvimento ágil dos módulos. As páginas desenvolvidas deveriam permitir fácil entendimento sem causar dúvidas quanto ao funcionamento, esse desafio foi superado após grande dificuldade em posicionamento dos elementos de tela e com a utilização da biblioteca Bootstrap, aplicada nas cores e formas dos componentes de uma página web.

Para o *Web Service* foram utilizados *frameworks* que propiciassem a fluidez e produtividade no desenvolvimento e, mais uma vez, houveram problemas para adequar os *frameworks* às necessidades levantadas, além dessas adequações foram identificados problemas no desempenho para tratar questões com *listening*. As correções foram aplicadas, porém não houve tempo hábil para a realização de um novo teste com uma prova real do E.V.A.

Durante as reuniões foram identificadas as dificuldades encontradas pelos professores com a utilização do E.V.A no formato impresso. Os professores não possuem a autonomia de alterar a prova para um conteúdo mais atualizado, a alteração de questões já existentes gera custos, pois após uma questão ser alterada todas as provas que contem aquela questão deverão ser impressas novamente.

Com o áudio sendo administrado pelo professor, o aluno não consegue imprimir seu ritmo durante a prova e pode não entender o áudio corretamente devido à ruídos que possam vir a ocorrer.

5.2. Considerações Finais

Atualmente o cumprimento de todas as etapas da prova E.V.A dura cerca de 1 semana, o processo engloba desde a aplicação da prova até o momento em qual o

Aluno recebe sua nota, a correção da prova pelos Professores leva em média 2 dias, considerando que o Professor não terá outras atividades nesse período.

O projeto desenvolvido nesse trabalho veio com o intuito de possibilitar a criação e edição de provas, desonerar processo que atualmente precisa de 1 semana para ser concluído, pois engloba desde a aplicação da prova até a nota ser recebida pelo Aluno. Esse processo foi informatizado e dividido em módulos que visam propiciar ao aluno e ao professor uma melhor experiência. Com o *AdminTeacher* a criação de uma prova poderá ser feita por qualquer Professor e sempre que ele achar necessário, está mesma prova pode ter suas questões editadas em minutos. O *AdminStudent*, módulo onde o Aluno realiza a prova, proporciona aos Alunos um melhor ritmo de prova, pois eles poderão seguir seu próprio ritmo, ouvir os áudios no momento que desejarem, assim não será necessário que o Professor inicie os áudios e ele poderá então dedicar-se em acompanhar os resultados dos Alunos durante a prova.

Com o trabalho desenvolvido podemos analisar que o processo que engloba o Exame de Verificação de Aprendizagem terá seu tempo reduzido, com isso o Professor terá como identificar as dificuldades encontradas pelos Alunos em um curto período de tempo. A Tabela 19 exibe os ganhos observados ao utilizar o EvaSystem.

Tabela 19 - Ganhos Identificados com o EvaSystem

Descrição do Processo	Atual	Com o EvaSystem
Aplicação da Prova e obtenção dos Resultados.	Necessário 1 Semana.	Instantes após o término da prova.
Correção da Prova.	São necessários 2 dias.	Esta etapa é feita pelo EvaSystem durante a extração dos relatórios.
Possibilidade de modificação da Prova em geral.	Existe a possibilidade, porém gerava custos para impressão de novas provas.	Pode ser feita pelo módulo do Professor.

Descrição do Processo	Atual	Com o EvaSystem
Criação de nova prova.	Gera custos para ser feita e não pode ser criada a qualquer momento.	Não há custos financeiros e pode ser feita a qualquer momento.
Resposta em Gabarito	Atualmente os alunos precisam marcar suas respostas em gabaritos e sem rasuras.	Não será necessário utilizar gabaritos e o aluno terá a possibilidade de durante a pergunta, caso ele ache necessário, trocar a alternativa escolhida.
Seção Listening.	Os áudios são executados e controlados pelo professor em uma caixa de som disponibilizada pela FATEC	Os alunos irão utilizar seus próprios fones de ouvido durante a prova.

Fonte: Autor, 2017.

A Tabela 19 exibe o tempo que o processo atualmente precisa para ser concluído e o ganho obtido com o EvaSystem.

5.3. Trabalhos Futuros

Durante os experimentos foram solicitadas novas *features* no módulo do Professor, as modificações servirão para melhor análise dos dados obtidos com a prova e assim possibilitar a geração de dados estatísticos do desempenho dos Alunos.

Como trabalho futuro foram identificadas as necessidades de criação dos itens abaixo:

- Relatório com gráficos das perguntas que os Alunos mais acertam e erram;
- Relatório de Seção e Parte que os Alunos possuem maior dificuldade;

- Gráfico com quantidade de acertos e erros (atualmente são apresentados somente os acertos);
- Implementação da arquitetura *web socket* para obtenção dos resultados dos Alunos.
- Desabilitar o acesso dos Alunos que realizaram a prova (em lote) após sua execução. E caso futuramente haja uma nova importação do mesmo Aluno, o mesmo seja reabilitado e associado à nova prova;
- Relatório que possibilite visualizar todas as provas do E.V.A realizadas pelo Aluno, com questões respondidas e acertos/erros.
- Restrição onde o professor poderá somente acessar resultados de provas que ele aplicou.
- Função que permita ao Professor realizar o versionamento da prova.

REFERÊNCIAS

ANGULARJS. **What Is AngularJS?**. Disponível em: <https://docs.angularjs.org/guide/introduction>. Acesso em: 12/04/2017.

APACHE MAVEN. **Maven - Welcome to Apache Maven**. Disponível em: <https://maven.apache.org/>. Acesso em 22/02/2017.

BOOTSTRAP. **Getting Started - Bootstrap**. Disponível em: <http://getbootstrap.com/gettingstarted/>. Acesso em: 13/04/2017

BOYARSKY J, SELIKOFF S. **OCA: Oracle Certified Associate Java SE 8 Programmer I**. Indianapolis: Sybex Wiley, 2015.

CODECADEMY. **MVC: Model, View, Controller**. disponível em: <https://www.codecademy.com/articles/mvc>. Acesso em: 22/07/2017.

DOCS SPRING. **Class BCrypt**. Disponível em: <https://docs.spring.io/spring-security/site/docs/current/apidocs/org/springframework/security/crypto/bcrypt/BCrypt.html>. Acesso em: 12/07/2017.

ENVATOTUTS+. **Guia de Introdução aos conceitos HTTP e REST**. Disponível em: <https://code.tutsplus.com/pt/tutorials/a-beginners-guide-to-http-and-rest--net-16340>. Acesso em: 09/08/2017.

HIBERNATE ORM. **Hibernate ORM - Hibernate ORM**. Disponível em: <http://hibernate.org/orm/>. Acesso em: 25/02/2017.

JAVA. **The History of Java Technology**. Disponível em: <http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html>. Acesso em: 05/05/2017.

JSON. **Introducing JSON**. Disponível em: <https://www.json.org>. Acesso em: 26/06/2017.

LECHETA, Ricardo. **Web Services RESTful: Aprenda a criar web services RESTful em Java na nuvem do Google**. São Paulo: Novatec Editora Ltda, 2015.

LIQUIBASE. **Liquibase - Database Refactoring**. Disponível em: <http://www.liquibase.org/documentation/index.html>. Acesso em: 20/04/2017

MYSQL. **Why MySQL**. Disponível em: <https://www.mysql.com/why-mysql/>. Acesso em: 10/03/2017.

SPRING. **Understanting REST**. Disponível em: <https://spring.io/understanding/REST>. Acesso em; 15/08/2017.

SPRING. **Spring Data JPA – Reference Documentation**. Disponível em: <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods>. Acesso em: 19/09/2017.

TOEFL. **TOEFL**. Disponível em: <https://www.ets.org/toefl>. Acesso em: 23/09/2017.

UML. **Fundamentos Básicos de UML - O Diagrama de Classes: Uma Introdução aos diagramas de estrutura UML 2**. Disponível em: <https://www.ibm.com/developerworks/br/rational/library/content/RationalEdge/sep04/bell/index.html>. Acesso em: 20/07/2017.

VADIUM SOFTWARE. **5 Easy Steps to Understanding JSON Web Token**. Disponível em: <https://medium.com/vandium-software/5-easy-steps-to-understanding-json-web-tokens-jwt-1164c0adfcec>. Acesso em: 10/06/2017.