CENTRO DE INFORMÁTICA - UFPB
MÉTODOS DE PROJETOS DE SOFTWARE

API de Banco de Provas da UFPB

DOCUMENTO DE REQUISITOS

VERSÃO 0.2

João Pessoa, Fevereiro de 2017

EQUIPE DO PROJETO

Diogo Ventura Dantas
11403718
dioogoven@gmail.com
Higor Araújo dos Anjos
11403767
higor.araujo@cc.ci.ufpb.br
Marcos Henrique Alves da Silva
11409511
marcos.alves@cc.ci.ufpb.br

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Propósito do documento

Este documento tem por objetivo apresentar a especificação dos requisitos que a *API* de banco de provas da UFPB deve atender em diferentes níveis de detalhamento. Tendo a finalidade de prover serviços de gerenciamento de provas das disciplinas ofertadas na Universidade Federal da Paraíba para o desenvolvimento de aplicações que os utilizem.

1.2 Acrônimos e abreviações

Acrônimo/abreviação	Descrição
RNF	Requisito Não Funcional
RF	Requisito Funcional
UC	Caso de Uso
UFPB	Universidade Federal da Paraíba
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator

Tabela 1: Acrônimo e abreviação

1.3 Visão geral do documento

O presente documento de requisitos é organizado como segue:

- **Introdução** - esta seção fornece informações gerais sobre o documento de requisitos, apresentado seu propósito, escopo e também os acrônimos e abreviações necessários para interpretar o documento descrito.
- **Descrição geral** - fornece uma explanação breve sobre o produto de software a ser produzido e suas restrições gerais.
- **Elicitação dos Requisitos** - lista as técnicas utilizadas e resultados obtidos das elicitações de requisitos realizadas.
- **Requisitos** - apresenta os requisitos funcionais e não funcionais mapeados para serem atendidos no desenvolvimento do sistema
- **Descrição de Casos de Uso** - determina os principais casos de uso do sistema.

2 DESCRIÇÃO GERAL

O sistema descrito neste documento tem como objetivo auxiliar desenvolvedores a criar aplicações utilizando dados de provas aplicadas na UFPB. Através da disponibilidade de serviços de gerenciamento de provas, os usuários podem utilizar da base de dados para consultar provas, adicioná-las e criar estatísticas com esses dados dentro de suas aplicações. Com a utilização desta interface, inúmeras aplicações podem ser criadas para facilitar o estudo dos graduandos da UFPB. O acesso a provas anteriores de um disciplina pode ser considerado um bom exemplo.

2.1 Perspectiva do produto

A UFPB é uma das maiores instituições de ensino do Brasil, com mais de 30 mil alunos ativos e mais de 100 cursos disponíveis[5]. Completar um curso superior em uma universidade pública de qualidade não é uma tarefa simples, o grande número de evasão comprova essa realidade. Fatores como a dificuldade de acesso a informação contribuem de forma incisiva para essa realidade.

De maneira geral, a *API* proposta neste documento tem como principal objetivo disponibilizar diversas provas aplicadas na UFPB de uma forma simples e prática. Através da *API* diversos desenvolvedores poderão consumi-la para criação de aplicativos e ferramentas que auxiliem os alunos durante a sua vida acadêmica.

2.2 Usuários do software

- **Desenvolvedor** - papel de quem utiliza a *API* para a criação de suas aplicações.
- **Administrador** - papel de quem administra a *API*, podendo gerar relatórios e estatísticas a respeito dos dados quem a compõem e das aplicações que a autilizam.

2.3 Suposições e Restrições gerais

Para as funcionalidades de todos os usuários é necessário que estes possuam conhecimento dos serviços disponibilizados pelo sistema e como utilizá-los de forma correta. Não há distinção de funcionalidades entre usuários.

O sistema deve ser implantado em um servidor para que possa ser utilizado publicamente na Internet. Devido a isso, a equipe optou por desenvolver o sistema em um ambiente de testes de forma local e posteriormente, em um serviço de nuvem ou em servidores da UFPB. Ainda, por se tratar de um sistema web e por várias aplicações semelhantes funcionarem do mesmo modo, a equipe optou por utilizar a linguagem de programação Javascript e alguns de seus *frameworks*.

3 ELICITAÇÃO DOS REQUISITOS

Durante a fase de elicitação dos requisitos, foram utilizadas primordialmente as técnicas de *brainstorming* e análise de concorrência para a identificação das características desejadas no sistema. Estas técnicas foram utilizadas principalmente a fim de atender o prazo de desenvolvimento do projeto e pelo público alvo a ser alcançado.

Todas as reuniões de *brainstorm* e dados coletados durante essa fase foram registrados em fontes externas à esse documento.

3.1 Reuniões de *Brainstorming*

Foram realizadas reuniões de *brainstorming* a fim de definir o escopo do projeto. Algumas se deram com possíveis usuários do sistema e outros apenas com a equipe do projeto. As reuniões aconteceram tanto virtualmente, através de videoconferência, quanto presencialmente, no Centro de Informática da UFPB. Os documentos produzidos durante as sessões de *brainstorm* podem ser vistos no apêndice desse documento.

3.2 Análise de Concorrência

Não foram encontrados sistemas que exercem o mesmo objetivo do presente projeto. Porém, durante o processo de análise de concorrência algumas aplicações que podem utilizar *APIs* semelhantes foram encontradas, como o *website* Qconcursos[4].

Diferente do escopo deste projeto, mas ainda se tratando de serviços do mesmo segmento, foram avaliados os produtos de empresas como Dropbox[2] e Google Drive [3].

O mais próximo do objetivo de aplicações que podem ser desenvolvidas com uso da API descrita neste documento foi um ambiente criado no serviço de armazenamento em nuvem Google Drive criado pela Secretaria da Área II [**?**] da Universidade Federal de Pernambuco. Este ambiente é chamado de Mural Digital e é utilizado para ter acesso a listas de exercícios, horários de aulas e outras informações.

Além disso, foram encontrados sites de professores da UFPB onde provas são disponibilizadas, porém não são atualizadas e são informações espalhadas por vários endereços totalmente descentralizados.

Não foi possível realizar comparações entre softwares já existentes pois não foram encontrados APIs que realizassem o proposto neste documento.

4 REQUISITOS

Para estabelecer a prioridade dos requisitos, foram adotadas as denominações *"essencial"*, *"importante"* e *"desejável"*.

- **Essencial** - é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.
- **Importante** - é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.
- **Desejável** - é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

4.1 Requisitos funcionais (RF)

4.1.1 [RF01] Permitir a inserção de prova

Descrição: O Sistema deve permitir a inserção de uma prova informando o arquivo (em formato .pdf) e informações de departamento, curso, disciplina e período em suas requisições. Caso o usuário informe campos erroneamente ou inexistentes, o sistema deve informar o erro na resposta da requisição.

Prioridade: Essencial.

4.1.2 [RF02] Permitir a edição de prova

Descrição: O sistema deve permitir a edição de uma prova já cadastrada, informando o identificador da prova e os campos alterados na requisição. O sistema deve informar o sucesso ou insucesso da operação.

Prioridade: Essencial.

4.1.3 [RF03] Permitir a deleção de prova

Descrição: O sistema deve permitir a deleção de uma prova já cadastrada através de um identificador da prova. O sistema deve informar o sucesso ou insucesso da operação.

Prioridade: Essencial.

4.1.4 [RF04] Permitir a consulta de prova

Descrição: O sistema deve permitir a consulta de uma prova presente no sistema retornando todas as informações sobre ela. O sistema deve informar o sucesso ou insucesso da operação.

Prioridade: Essencial.

4.1.5 [RF05] Permitir a consulta de provas por departamento

Descrição: O usuário do sistema pode requisitar uma lista de provas de um departamento específico da Universidade Federal da Paraíba.

Prioridade: Importante.

4.1.6 [RF06] Permitir a consulta de provas por disciplina

Descrição: O sistema deve responder uma lista de provas de uma disciplina específica de acordo com o desejo do usuário.

Prioridade: Importante.

4.1.7 [RF07] Permitir a consulta de provas por período

Descrição: É permitido ao usuário obter uma lista de provas de um período desejado quando especificado.

Prioridade: Importante.

4.1.8 [RF08] Permitir a consulta de provas por curso

Descrição: É permitido ao usuário obter uma lista de provas de um determinado curso quando especificado.

Prioridade: Importante.

4.1.9 [RF09] Permitir a consulta das últimas provas adicionadas

Descrição: O sistema deve fornecer uma lista das 10 últimas provas adicionadas no serviço quando requisitado.

Prioridade: Desejável.

4.1.10 [RF10] Permitir a consulta das últimas provas adicionadas de uma disciplina específica

Descrição: O sistema deve fornecer uma lista das 10 últimas provas de uma disciplina específica quando requisitado.

Prioridade: Desejável.

4.1.11 [RF11] Permitir Atribuição de pontuação a uma prova

Descrição: O sistema deve permitir que o usuário possa atribuir uma pontuação, em uma escala numérica, a uma prova com o objetivo de realizar a validação da mesma.

Prioridade: Desejável.

4.1.12 [RF12] Permitir geração de provas relacionadas a determinada disciplina

Descrição: É permitido ao usuário gerar uma prova de uma determinada disciplina composta por questões aleatórias de outras provas cadastradas no banco de dados da mesma disciplina.

Prioridade: Desejável.

4.1.13 [RF13] Permitir acesso com privilégios ao sistema

Descrição: O sistema deve fornecer um formulário de autenticação de superusuários.

Prioridade: Essencial.

4.1.14 [RF14] Permitir a inserção de superusuários

Descrição: É permitido a superusuários inserir novos superusuários.

Prioridade: Essencial.

4.1.15 [RF15] Permitir a deleção de superusuários

Descrição: É permitido a superusuários deletar outros superusuários.

Prioridade: Essencial.

4.1.16 [RF16] Permitir a atualização de superusuários

Descrição: É permitido a um superusuário fazer apenas a atualização do seu próprio cadastro.

Prioridade: Essencial.

4.1.17 [RF17] Permitir a consulta de superusuários

Descrição: É permitido a superusuários consultar outros superusuários.

Prioridade: Essencial.

4.1.18 [RF18] Permitir a geração de relatórios para superusuários

Descrição: O sistema deve fornecer a superusuários a opção de gerar relatórios.

Prioridade: Essencial.

4.1.19 [RF19] Permitir a inserção de usuários

Descrição: O sistema deve fornecer um formulário de cadastro de usuários.

Prioridade: Essencial.

4.1.20 [RF20] Permitir a deleção de usuários

Descrição: É permitido a superusuários deletar usuários.

Prioridade: Essencial.

4.1.21 [RF21] Permitir a consulta de usuários

Descrição: É permitido a superusuários consultar usuários.

Prioridade: Essencial.

4.1.22 [RF22] Permitir acesso como usuário

Descrição: O sistema deve fornecer um formulário de autenticação de usuários.

Prioridade: Essencial.

4.1.23 [RF23] Permitir cadastro de projeto

Descrição: O sistema deve fornecer, para usuários cadastrados, o formulário de inserção de projetos.

Prioridade: Essencial.

4.2 Requisitos não funcionais (RNF)

4.2.1 [RNF01] Armazenamento de Provas

Descrição: O sistema deve persistir as provas submetidas pelos usuários de maneira não volátil.

Prioridade: Essencial.

4.2.2 [RNF02] Protocolo de Comunicação

Descrição: O sistema deve se comunicar com os usuários através do protocolo de comunicação web HTTP1.1. Utilizando requisições do tipo GET, POST, PUT e DELETE.

Prioridade: Essencial.

4.2.3 [RNF03] Ferramentas de programação

Descrição: O sistema deve ser desenvolvido utilizando um *framework* de código aberto.

Prioridade: Essencial.

4.2.4 [RNF04] Formato de submissão de provas

Descrição: As provas submetidas pelos usuários devem ser no formato pdf sem rasuras, respostas ou qualquer outro tipo de escrita manual. Devem seguir o fluxo lógico com perguntas sequenciais e legíveis.

Prioridade: Importante.

4.2.5 [RNF05] Disponibilidade de Acesso

Descrição: O sistema deverá estar disponível para acesso pelo menos em 80% do tempo.

Prioridade: Essencial.

4.2.6 [RNF06] Tempo de resposta

Descrição: Nenhuma resposta de consulta a informações deve demorar mais que 3 segundos para as consultas de listas e 1 segundo para a consulta de uma prova específica.

Prioridade: Essencial.

4.2.7 [RNF07] Número de Usuários

Descrição: O sistema deve suportar até 100 usuários simultâneos.

Prioridade: Essencial.

4.2.8 [RNF08] Tempo de Manutenção

Descrição: Um programador de manutenção com pelo menos 6 meses de experiência no suporte ao produto deverá ser capaz de dar suporte em não mais do que 1 hora de trabalho.

Prioridade: Importante.

4.2.9 [RNF09] Persistência de dados

Descrição: O sistema deverá persistir os dados em um banco de dados de modelo não-relacional, de código aberto e com suporte e documentação amplamente consolidado na comunidade.

Prioridade: Essencial.

4.2.10 [RNF10] Versionamento de Código

Descrição: O sistema deve ser versionado utilizando um sistema de controle descentralizado em repositório de acesso público.

Prioridade: Essencial.

4.2.11 [RNF11] Requisições com autorização

Descrição: As operações de deleção, alteração e inserção devem garantir autorização prévia do sistema através do protocolo padrão para autorização *OAuth 2.0*.

Prioridade: Desejável.

4.2.12 [RNF12] Documentação necessária

Descrição: O sistema deve possuir documentação completa de utilização do sistema. Na documentação, é necessário especificar o formato das requisições, os *endpoints* de cada serviço e exemplos de utilização. Para a geração de documentação automática com os padrões atuais, será utilizada a ferramenta *APIDOC* [1].

Prioridade: Importante

4.2.13 [RNF13] Extração de informações do SIGAA

Descrição: O sistema deverá extrair as informações sobre disciplinas e cursos do SIGAA(Sistema Integrado de Gestão de Atividade Acadêmicas) através da API *SIGAA UFPB* [6].

Prioridade: Essencial

4.2.14 [RNF14] Compatibilidade

Descrição: O sistema deverá se comunicar com os navegadores web mais modernos: Internet Explorer 10+, Google Chrome 38+, Mozilla Firefox 34+, Apple Safari 7+ e Opera 26+.

Prioridade: Essencial

4.2.15 [RNF15] Gerador automático de provas

Descrição: O gerador automático de provas do sistema deve retornar uma prova no formato PDF contendo no máximo 10 questões.

Prioridade: Desejável

5 CASOS DE USO

5.1 Diagrama Geral de Casos de Uso

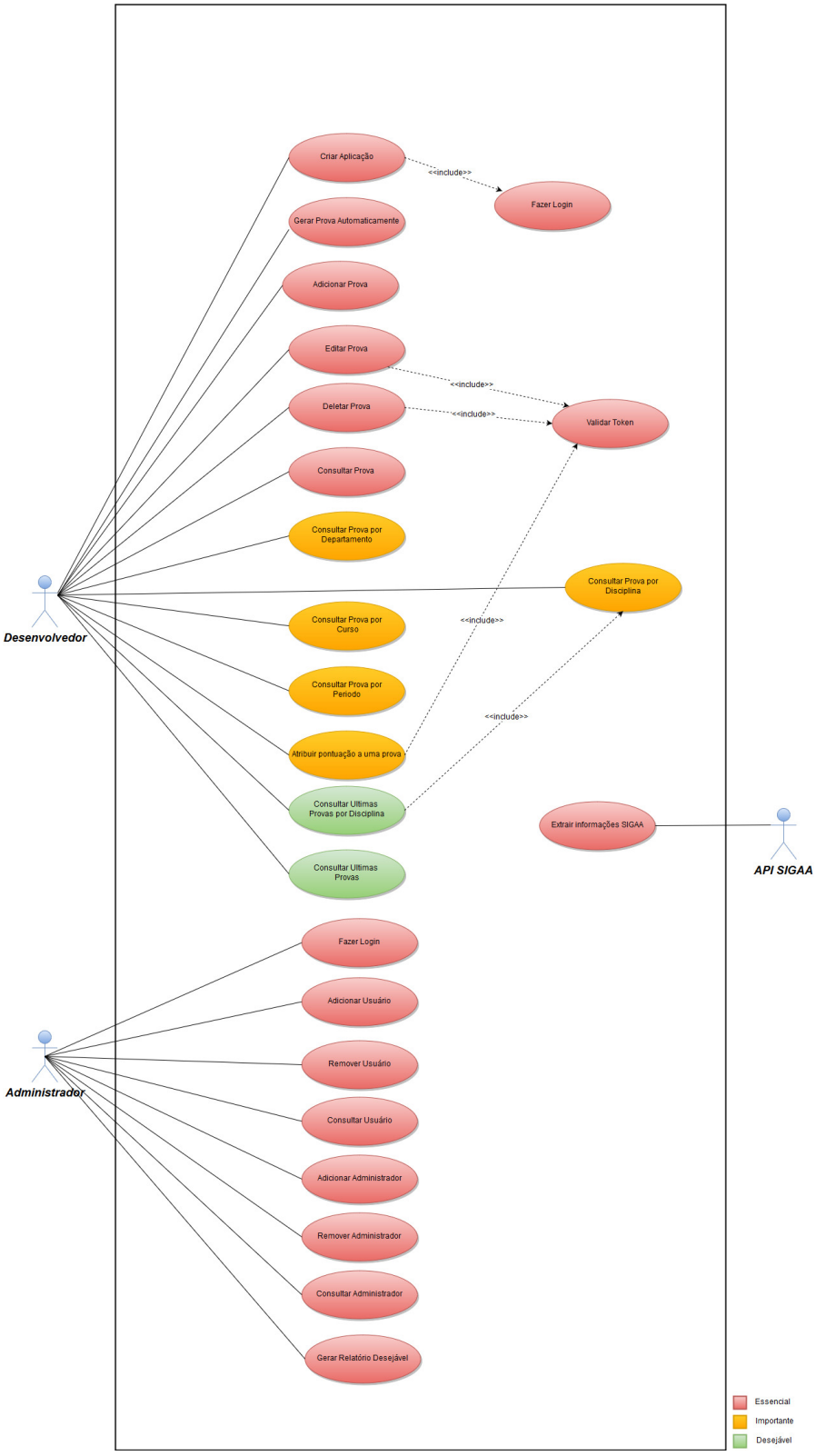

Figura 1: Diagrama de Casos de Uso

5.2 Descrição de Casos de Uso

5.2.1 [UC01] Submeter prova

Identificador:[UC01]
Nome: Submeter Prova
Descrição: Esse caso de uso se dá quando o usuário envia um requisição para a submissão de uma prova através do protocolo HTTP.
Prioridade: Essencial
Atores: Desenvolvedor
Pré-condições: Não há
Pós-condições: A prova é inserida no banco de dados.
Fluxo Principal:

1. O usuário envia a requisição do tipo POST informando as informações da prova
2. O sistema verifica as condições para a inserção e informa o sucesso ou insucesso

Fluxo Secundário - Informações fora do padrão

1. Caso as informações da prova não correspondam aos padrões especificados uma messagem de erro deve ser retornada

Fluxo Secundário - Expressão mal-formada

1. Caso a requisição não esteja no padrão utilizado pelo sistema, uma mensagem de erro deve ser retornada

5.2.2 [UC02] Deletar prova

Identificador:[UC02]
Nome: Deletar prova
Descrição: O ator envia uma requisição de deleção para o servidor (DELETE).
Prioridade: Essencial
Atores: Desenvolvedor
Pré-condições: Não há
Pós-condições: A prova é apagada do banco de dados.
Fluxo Principal:

1. O usuário envia a requisição do tipo DELETE informando o identificador da prova
2. O sistema verifica a existência da prova no banco de dados e retorna o sucesso ou insucesso da operação

Fluxo Secundário - Prova Inexistente

1. Caso não exista prova com esse identificador, o sistema responde a requisição com uma mensagem de erro

Fluxo Secundário - Expressão mal-formada

1. Caso a requisição não esteja no padrão utilizado pelo sistema, uma mensagem de erro deve ser retornada

5.2.3 [UC03] Editar prova

Identificador: [UC03]
Nome: Editar prova
Descrição: Esse caso de uso se dá quando o usuário envia uma requisição de alteração de uma prova específica através do protocolo HTTP informando o identificador único da prova
Prioridade: Essencial
Atores: Desenvolvedor
Pré-condições: Não há
Pós-condições: As informações da prova são alteradas no banco de dados
Fluxo Principal:

1. O usuário envia a requisição do tipo PUT informando o identificador da prova pela *URL* e as informações atualizadas no corpo da requisição
2. O sistema consulta o identificador da prova e altera as informações da prova em questão
3. O sistema informa o sucesso ou insucesso da operação

Fluxo Secundário - Prova Inexistente

1. Caso não exista prova com esse identificador, o sistema responde a requisição com uma mensagem de erro

Fluxo Secundário - Expressão mal-formada

1. Caso a requisição não esteja no padrão utilizado pelo sistema, uma mensagem de erro deve ser retornada

5.2.4 [UC04] Consultar prova

Identificador:[UC04]
Nome: Consultar prova
Descrição: Esse caso de uso se dá quando o usuário envia uma requisição para a consulta de uma prova específica através do protocolo HTTP.
Prioridade: Essencial
Atores: Desenvolvedor
Pré-condições: Não há
Pós-condições: As informações específicas dessa provas são retornadas
Fluxo Principal:

1. O usuário envia a requisição do tipo GET informando o identificador da prova
2. O sistema consulta no banco de dados e retorna as informações da prova pesquisada

Fluxo Secundário - Prova Inexistente

1. Caso não exista prova com esse identificador, o sistema responde a requisição com uma mensagem de erro

Fluxo Secundário - Expressão mal-formada

1. Caso a requisição não esteja no padrão utilizado pelo sistema, uma mensagem de erro deve ser retornada

6 DIAGRAMA DE CLASSES

6.1 DIAGRAMA DE CLASSES DE ANÁLISE

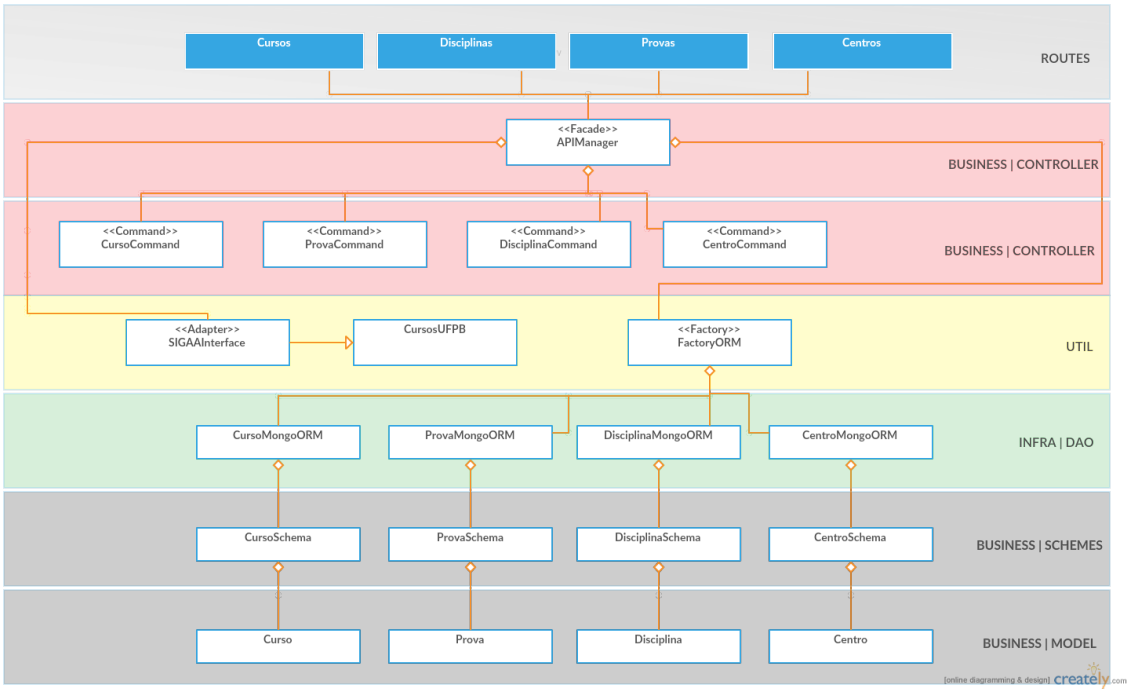

Figura 2: Diagrama de classes de análise servidor

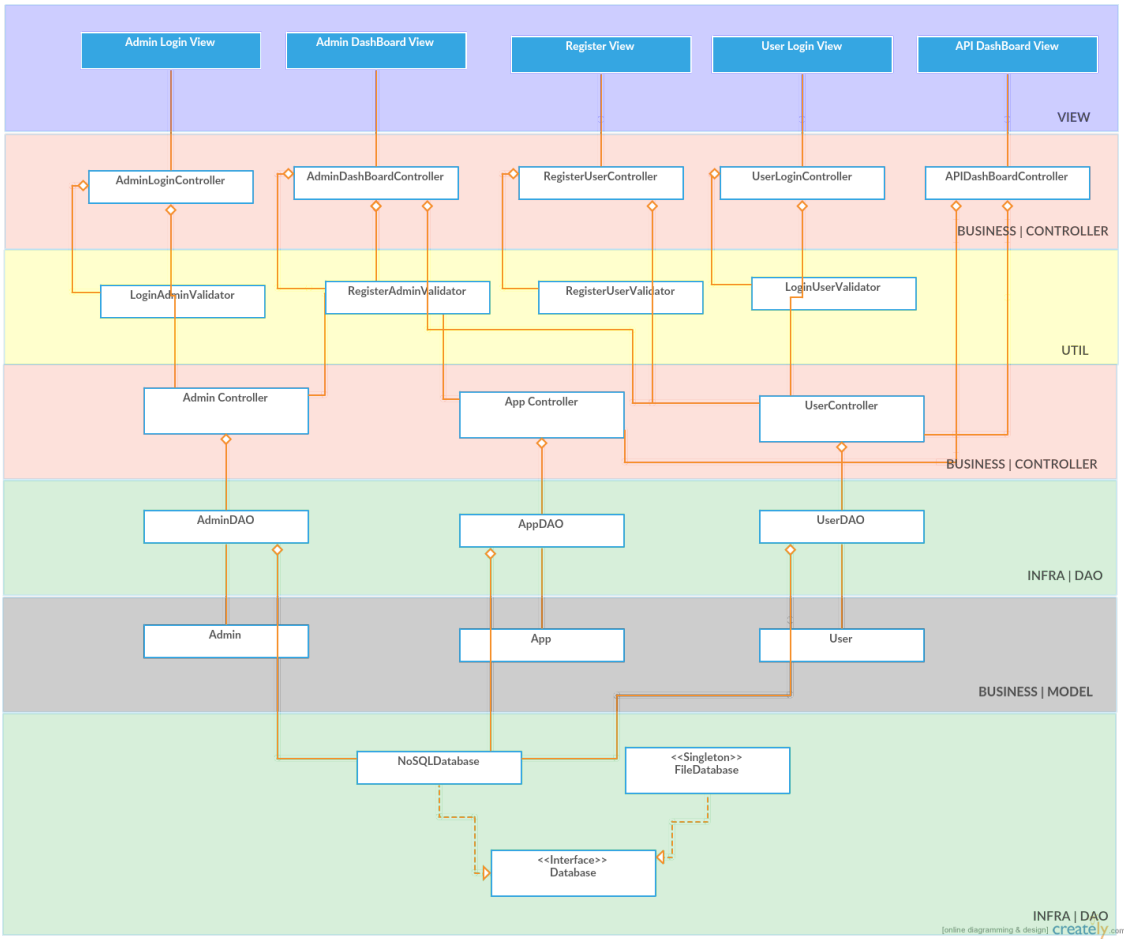

Figura 3: Diagrama de classes de análise cliente

7 GLOSSÁRIO

prova - Uma prova, nesse documento, é considerado o teste avaliativo aplicado por professores da Universidade Federal da Paraíba em uma disciplina específica.

disciplina - Um curso possui várias disciplinas em seu currículo. No escopo desse documento, uma disciplina designa um determinado ramo do conhecimento ou um conjunto de aulas aos quais os alunos assistem e sobre as quais eles serão examinados através de provas, podendo ser aprovados ou não.

curso - Um curso é o programa completo de estudos necessários para se obter um diploma na universidade. No presente documento, este termo se refere apenas aos programas oferecidos pela UFPB.

departamento - É uma divisão administrativa de uma universidade que lida com o processo de ensino e aprendizagem de docentes e discentes. Um departamento pode oferecer vários cursos e, no documento, os departamentos e cursos são os pertencentes à UFPB.

período - Um período é o espaço no calendário acadêmico onde as aulas das universidades são ministradas. Muitas vezes um período é composto por 6 meses e existem pausas para férias entre períodos. Um período, neste documento, é representado pelo ano e número do semestre separado por ponto. Pro exemplo, o primeiro período do ano de 2017 é representado por *2017.1*.

usuário - Um usuário, nesse documento, são pessoas ou organizações que utilizarão nosso serviço.

8 APÊNDICES

8.1 Elicitação de Requisitos

8.1.1 Brainstorm

[BS01] Reunião 1
Participantes: Diogo Ventura; Higor Anjos; Marcos Alves.
Data: 10 de Fevereiro de 2017.
Local: Centro de Informática - UFPB.
Ferramentas Utilizadas: Google Docs.
Focos de Discussão:

1. Ideias inicias sobre o escopo da proposta.
2. Técnicas de elicitações a ser utilizadas.

Resumo:

Ficou decidido que a proposta se basearia apenas na implementação de uma API sem o *front-end*.

Foi decidido por realizar *brainstormings* como uma das técnicas de elicitações, e seriam realizadas de acordo com a disponibilidade do grupo.

Como um segunda técnica de elicitação o grupo optou pela análise de concorrência, ficou acordado pesquisas iniciais sobre APIs e sistemas relacionados a banco de provas.

[BS02] Reunião 2
Participantes: Diogo Ventura; Higor Anjos; Marcos Alves.
Data: 24 de Feveiro de 2017 **Ferramentas Utilizadas:** Quadro branco; Sharelatex.
Focos de Discussão:

1. Análise de concorrência.
2. Funcionalidades que a API proposta iria fornecer.
3. Levantamento dos requisitos (funcionais e não-funcionais).

Resumo:

Algumas funcionalidades foram retiradas para focar no principal objetivo do software. Após pesquisas, decidiu-se por proporcionar uma api simples e organizada para facilitar o uso e possibilitar a criação rápida de aplicações utilizando-a.

A *stack* de desenvolvimento foi escolhida: *Nodejs*, *Expressjs* e o banco de dados não-relacional *MongoDB*.

[BS03] Reunião 3
Participantes: Diogo Ventura; Higor Anjos; Marcos Alves.
Data: 28 de Fevereiro de 2017.
Local: Videoconferência via Google Hangouts.
Ferramentas Utilizadas: Draw.io; Sharelatex.
Focos de Discussão:

1. Última reunião antes da primeira entrega.
2. Criação do diagrama de casos de uso.
3. Criação do diagrama de classes.

Resumo:

Todas as recomendações do professor foram revisadas e incorporadas no documento. Os diagramas foram adicionados na primeira versão.

O grupo decidiu por incorporar a arquitetura MVC no projeto devido ao uso anterior pelos integrantes da equipe e entendimento prévio sobre o assunto.

[BS04] Reunião 4
Participantes: Diogo Ventura; Higor Anjos; Marcos Alves.
Data: 17 de Março de 2017.
Local: Videoconferência via Google Hangouts.
Ferramentas Utilizadas: Sharelatex.
Focos de Discussão:

1. Reunião após a correção da primeira entrega.
2. Modificação e revisão do documento.

Resumo:
O grupo tentou incorporar as sugestões do professor no projeto. Nesse momento ficou

decidido haver uma interface para cadastro de administradores para que este possa gerar relatórios.

Apesar de fugir da ideia inicial da API, o grupo tentou incorporar a criação automática de provas ao projeto cadastrando questões sobre uma disciplina específica. Assim, o usuário que quisesse, poderia solicitar uma prova de uma disciplina específica com a quantidade de questões desejada e obteria uma prova automática.

[BS04] Reunião 5
Participantes: Higor Anjos e Marcos Alves.
Data: 16 de Maio de 2017.
Local: Videoconferência via Google Hangouts.
Ferramentas Utilizadas: Sharelatex.
Focos de Discussão:

1. Revisao do diagrama de casos de uso.
2. Revisão do diagrama de classes

Resumo:
Ficou decidido retirar o caso de validar informações já que essa ação pode ser considerada intrínseca em todos os casos. Os casos de usos de consulta não necessitam de validação de token e é necessário incluir o caso de uso de pontuação de provas como forma de avaliação. Entendemos que o OAuth será utilizado apenas como um framework e não um serviço externo já que irá apenas formecer uma interface na criação do access token para o cliente poder utilizar recursos privados da api. Por estarmos usando o framework Mongoose para criar os esquema como modelo, resolvemos alterar os atributos de uma prova como uma coleção para atributos concretos. No lado da API, alterações foram feitas para melhorar o diagrama de classes. (Anotado em papel de rascunho)

REFERÊNCIAS

[1] apidocjs.com. Disponível em: <http://apidocjs.com/>, Acesso em: 20 de Fev. 2017.

[2] Dropbox. Disponível em: <https://www.dropbox.com/developers/documentation>, Acesso em: 20 de Fev. 2017.

[3] Google drive. Disponível em: <https://developers.google.com/drive/v3/web/about-sdk>, Acesso em: 20 de Fev. 2017.

[4] Qconcursos.com. Disponível em: <https://www.qconcursos.com>, Acesso em: 20 de Fev. 2017.

[5] UFPB. Disponível em: <http://www.ufpb.br/content/hist%C3%B3rico>, Acesso em: 27 de Fev. 2017.

[6] Ygor, CRISPIM David, GUILHERME Fernando, BRITO, Gustavo, SOBRAL. Cursos UFPB API. Disponível em: <https://github.com/fernandobrito/cursos_ufpb>, Acesso em: 27 de Fev. 2017.