UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
BACHARELADO EM ENGENHARIA DE SOFTWARE

StudentHistory: Uma Ferramenta de Análise Visual do Desempenho de Alunos de Graduação da UFRN

Jonathan Martins Barros Costa

Natal-RN
junho de 2018

Jonathan Martins Barros Costa

StudentHistory: Uma Ferramenta de Análise Visual do Desempenho de Alunos de Graduação da UFRN

Monografia de Graduação apresentada ao Departamento de Informática e Matemática Aplicada do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte como requisito parcial para a obtenção do grau de bacharel em Engenharia de Software.

Orientador

Prof. Dr. Bruno Santana Da Silva

Universidade Federal do Rio Grande do Norte – UFRN

Departamento de Informática e Matemática Aplicada – DIMAp

Natal-RN

junho de 2018

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Setorial Prof. Ronaldo Xavier de Arruda - CCET

Costa, Jonathan Martins Barros.
StudentHistory: uma ferramenta de análise visual do desempenho de alunos de graduação da UFRN / Jonathan Martins Barros Costa. - 2018.
55f.: il.

Monografia (graduação) - Universidade Federal do Rio Grande do Norte, Centro de Ciências Exatas e da Terra, Departamento de Informática e Matemática Aplicada, Bacharelado em Engenharia de Software. Natal, 2018.
Orientador: Bruno Santana da Silva.

1. Engenharia de software - Monografia. 2. Análise visual - Monografia. 3. Análise de desempenho educacional - Monografia. 4. Desenvolvimento de software - Monografia. I. Silva, Bruno Santana da. II. Título.

RN/UF/CCET CDU 004.41

Elaborado por Joseneide Ferreira Dantas - CRB-15/324

Monografia de Graduação sob o título *StudentHistory: Uma Ferramenta de Análise Visual do Desempenho de Alunos de Graduação da UFRN* apresentada por Jonathan Martins Barros Costa e aceita pelo Departamento de Informática e Matemática Aplicada do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte, sendo aprovada por todos os membros da banca examinadora abaixo especificada:

Prof. Dr. Bruno Santana Da Silva

Orientador

Instituto Metrópole Digital

Universidade Federal do Rio Grande do Norte

Profa. Dra. Adja Ferreira de Andrade

Instituto Metrópole Digital

Universidade Federal do Rio Grande do Norte

Profa. Dra. Marcia Jacyntha Nunes Rodrigues Lucena

Departamento de Informática e Matemática Aplicada

Universidade Federal do Rio Grande do Norte

Natal-RN, vinte e um de junho de dois mil e dezoito.

Dedico este trabalho aos meus pais, **Cylene** e **Antônio**.

Agradecimentos

Foram anos de uma longa caminhada até a chegada desse momento que parecia sem fim até pouco tempo atrás. Aqui deixo meu muito obrigado especial a todos que permitiram isso acontecer.

Agradeço muito aos meus pais, Cylene e Antônio, por todo o sacrifício e determinação na minha criação, pelas oportunidades que tive, graças a vocês, pois me fizeram ir mais longe que eu mesmo acreditava poder ir. Obrigado a minha irmã, Jéssyca, por todo o amor e confiança depositada em mim para chegar onde cheguei. Essa conquista também é de vocês.

Agradeço à UFRN por ter aberto tantas portas na minha vida, por ter me permitido uma experiência tão engrandecedora, tanto pessoal quanto profissional. Obrigado por me permitir conhecer pessoas e profissionais que contribuíram para meu aprendizado na área de engenharia de *software*.

Agradeço ao meu orientador Bruno Santana, sem o qual essa etapa final de conclusão de curso não teria sido possível, obrigado por toda a orientação e interesse na concretização deste trabalho.

Agradeço aos meus amigos e minha companheira, obrigado por fazerem parte da minha vida.

Agradeço aos amigos que fiz durante a graduação, em especial aos que estiveram comigo na 4Soft, vocês fizeram o tempo na universidade ser mais acolhedor. Ter a oportunidade de trabalhar com alguns de vocês de novo fora da universidade é muito gratificante. Vocês são responsáveis por eu nunca desistir da profissão e saibam que me inspiro em cada um de vocês para continuar fazendo o meu melhor e contribuir mundo afora como já o fazem. Obrigado Rafael, Thiago, Iago, Andreza, Waldyr, Bernardo. Obrigado também aos que não estão listados aqui, não se sintam excluídos, por favor.

Obrigado a todos que ajudaram na minha motivação para chegar aqui.

Obrigado.

StudentHistory: Uma Ferramenta de Análise Visual do Desempenho de Alunos de Graduação da UFRN

Autor: Jonathan Martins Barros Costa
Orientador: Prof. Dr. Bruno Santana Da Silva

RESUMO

Analisar o desempenho de estudantes em uma universidade nem sempre é uma atividade fácil de ser realizada, principalmente quando aplicada a um conjunto grande de indivíduos e uma diversidade de cursos. Este trabalho apresenta o StudentHistory, uma ferramenta de análise visual do desempenho de alunos de graduação da UFRN, integrada ao seu Sistema Integrado de Gestão de Atividades Acadêmicas, o SIGAA. Esta é uma solução web que permite docentes da universidade visualizem gráficos do tipo Sankey com o fluxo de alunos conforme suas notas entre disciplinas escolhidas. A geração destes gráficos ocorre de forma automática, sem necessidade de conhecimento aprofundado em TI, Estatística ou Design. O projeto, desenvolvimento e avaliação preliminar desta ferramenta são descritos neste documento. Essa ferramenta pode facilitar a identificação de oportunidades de melhoria ou de demandas de vagas em disciplinas de graduação da UFRN.

Palavras-chave: Análise visual, análise de desempenho educacional, desenvolvimento de software.

StudentHistory: A Tool for Performance Visual Analysis of Undergraduates at UFRN

Author: Jonathan Martins Barros Costa
Advisor: Prof. Dr. Bruno Santana Da Silva

ABSTRACT

Analysing the performance of students in a university is not always an easy activity to perform, especially when applied to a large group of individuals and a diversity of courses. This work presents the StudentHistory, a tool for academic performance visual analysis of undergraduates at UFRN, integrated to its integrated academic activities management system (SIGAA). This is a web-based solution that allows university professors to view Sankey graphs with the flow of students according to their grades between chosen subjects. The generation of these graphs occurs automatically, without need of in-depth knowledge in IT, Statistics or Design. The design, development and preliminary evaluation of this tool are described in this document. This tool can facilitate identification of opportunities for improvement or demand for vacancies in undergraduate courses at UFRN.

Keywords: Visual analysis, educational performance analysis, software development.

Lista de figuras

Lista de abreviaturas e siglas

NDE - Núcleo Docente Estruturante, p. 12

UFRN - Universidade Federal do Rio Grande do Norte, p. 13

SINFO - Superintendência de Informática, p. 16

API - Application Programming Interface, p. 17

MVC - Model-View-Controller, p. 24

BTI - Bacharelado em Tecnologia da Informação, p. 32

CSV - Comma-Separated Values, p. 32

SIGAA - Sistema Integrado de Gestão de Atividades Acadêmicas, p. 32

SCM - Source Configuration Management, p. 35

IMD - Instituto Metrópole Digital, p. 35

HTML - HyperText Markup Language, p. 37

SVG - Scalable Vector Graphics, p. 37

CSS - Cascading Style Sheets, p. 37

DOM - Document Object Model, p. 37

JSON – JavaScript Object Notation, p. 38

SQL - Standard Query Language, p. 38

TI - Tecnologia da Informação, p. 48

Sumário

1 Introdução

Existe uma expectativa na sociedade que indivíduos que se propõe às atividades acadêmicas desenvolvam habilidades e adquiram conhecimento das suas mais variadas formas. Alguns destes indivíduos apresentam dificuldades em alcançar este objetivo. Quando isto ocorre, geralmente é despertado um interesse em entender os problemas no desempenho destes indivíduos.

O acompanhamento do desempenho acadêmico de alunos é uma atividade de suma importância para professores e instituições de ensino. Além de apontar situações em que os alunos demonstram dificuldades, ele permite definir metas de melhoria no ensino e identificar demanda de disciplinas e professores para os próximos períodos. Entretanto, acompanhar o desempenho acadêmico nem sempre é uma tarefa fácil, principalmente quando aplicada a um conjunto grande de indivíduos e uma diversidade de cursos presentes nas universidades.

A representação visual de dados atua como papel fundamental na cognição humana a partir da percepção do mundo através da visão (WARE, 2004). Quando feita de forma adequada, bem estruturada, é possível facilitar interpretação de uma grande quantidade de dados em pouco tempo. Nesse sentido, a criação de gráficos e diagramas se torna ferramenta importante para a interpretação e análise de dados sobre o desempenho dos alunos.

Aliado a visualização gráfica de dados, temos a evolução e utilização dos sistemas de informação que contribuem principalmente com o poder de processamento computacional. Aproveitar a capacidade de processamento dos computadores faz com que análises antes feitas manualmente sejam realizadas cada vez mais rápido e com a menor quantidade de retrabalho humano possível.

1.1 Problema

Manipular uma quantidade significativa de dados, dificuldade em recuperar, organizar e interagir com grande volume de informações e retrabalho ao construir gráficos semelhantes são exemplos de obstáculos encontrados por docentes e instituições de ensino ao analisar o

desempenho acadêmico de alunos universitários. Outro problema importante no que se diz respeito à análise de desempenho dos estudantes é a falta de acompanhamento contínuo dos resultados obtidos pelos alunos. Análises feitas esporadicamente ou com dados muito antigos não dão o respaldo que os agentes educacionais necessitam para propor e executar intervenções adequadas. Muitas dessas análises acabam sendo pontuais e não são continuadas devido à dificuldade em manipular e analisar tantos dados com uma frequência semestral ou anual.

1.2 Objetivos

O objetivo principal deste trabalho é projetar e desenvolver o StudentHistory, um sistema web para apoiar, através da visualização gráfica de dados, o acompanhamento do desempenho de alunos de graduação ao longo das disciplinas do curso feito por agentes educacionais integrantes da coordenação, colegiado, Núcleo Docente Estruturante (NDE) ou setor pedagógico dos centros das universidades.

A princípio, o sistema deve restringir as análises feitas aos dados da Universidade Federal do Rio Grande do Norte e o acesso a docentes da mesma. Além da criação, o sistema deve permitir a visualização, edição e remoção de análises.

1.3 Organização do trabalho

O presente trabalho está organizado da seguinte forma: O capítulo 2 aborda o projeto da ferramenta StudentHistory, com visão geral, casos de uso, domínio e arquitetura. O capítulo 3 relata seu desenvolvimento, incluindo as etapas de implementação, tecnologias utilizadas e resultado obtido. O capítulo 4 descreve uma avaliação da ferramenta. O capítulo 5 apresenta as considerações finais do trabalho.

2 Projeto

Neste capítulo será apresentada a fase do projeto da ferramenta StudentHistory, descrita neste trabalho.

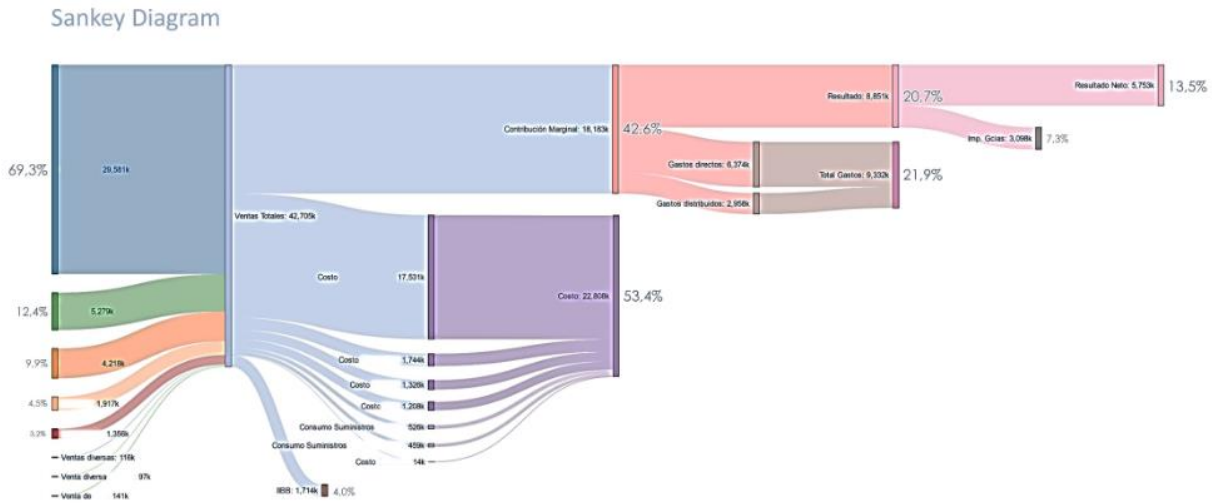
2.1 Visão Geral

Docentes universitários enfrentam uma dificuldade de acompanhar o rendimento dos estudantes nas disciplinas acadêmicas ao longo do curso. A partir desse problema, iniciou-se o planejamento de uma solução que fosse capaz de facilitar a visualização do desempenho de estudantes nos cursos de graduação da UFRN.

Para isso, uma série de requisitos foram elicitados, a fim de moldar o que a ferramenta precisaria dispor. Esses requisitos então, definiram a funcionalidade principal do sistema, StudentHistory, que seria permitir a criação de gráficos sobre o desempenho dos alunos em determinadas disciplinas para melhor compreensão dos desempenhos acadêmicos. Juntamente a isto temos a constante preocupação com a sigilosidade dos dados oferecidos aos usuários, sendo este o principal requisito não-funcional da ferramenta.

Apoiado nisso, foi pensado o tipo de gráfico que seria interessante apresentar para o usuário. Chegando, então, a decisão de uso do diagrama Sankey, ilustrado na Figura 1. Este tipo de diagrama representa um fluxo de informações que vão de um determinado ponto a outro. Neste contexto, essas informações seriam as notas dos estudantes em cada disciplina analisada.

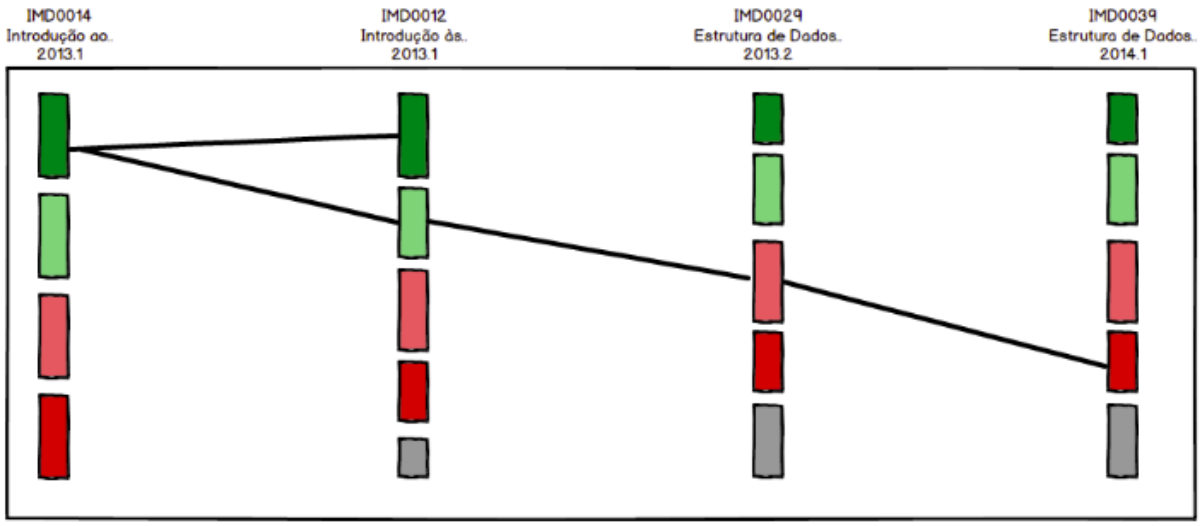
Figura 1 - Diagrama Sankey



Fonte: Adrianchiogna - Own work, CC BY-SA 4.0, goo.gl/3Rraow

A Figura 2 mostra um protótipo do diagrama proposto para o StudentHistory. Cada coluna representa as turmas de uma disciplina em determinado semestre, estratificada pelas notas dos alunos nessas turmas: maiores ou iguais a sete, maiores ou iguais a cinco e menores que sete, maiores ou iguais a três e menores que cinco, menores que três. Aqueles alunos que não cursaram a disciplina no semestre também seriam indicados em um grupo separado, totalizando cinco grupos por disciplina.

Figura 2 - Protótipo do diagrama Sankey proposto para o StudentHistory



Fonte: Bruno Santana da Silva (2017)

As ligações entre os cinco grupos de cada coluna com suas colunas adjacentes à direita formam um fluxo unidirecional que mostra por quais nós (grupos) do diagrama cada estudante da análise passou. Uma ligação entre dois nós possui largura proporcional à quantidade de estudantes que estão partindo de um grupo de notas à outro em outra disciplina, comparada ao total de alunos analisados.

O projeto do StudentHistory também levantou um questionamento importante sobre o esforço manual que seria necessário ao docente para a criação de um gráfico como o da Figura 2. O docente teria que recuperar, organizar e calcular sistematicamente um grande volume de informações. Esse esforço deveria ser melhor empregado na atividade de análise do gráfico propriamente dito, não no seu desenho. Assim, o sistema deveria facilitar a criação de gráficos que possibilitem realizar análises específicas, possuindo funcionalidades como criação, edição, visualização e remoção das análises.

O detalhamento dos casos de uso da ferramenta apresentada neste trabalho, está descrito a seguir.

2.2 Casos de Uso

Um caso de uso ilustra uma unidade de funcionalidade fornecida pelo sistema. O objetivo principal do diagrama de casos de uso é ajudar as equipes de desenvolvimento visualizarem os requisitos funcionais de um sistema, incluindo a relação entre os atores com processos essenciais, bem como as relações entre diferentes casos de uso (LARMAN, 2007;

BLAHA, 2006). Os atores nos casos de uso representam usuários e sistemas externos que interagem com o sistema.

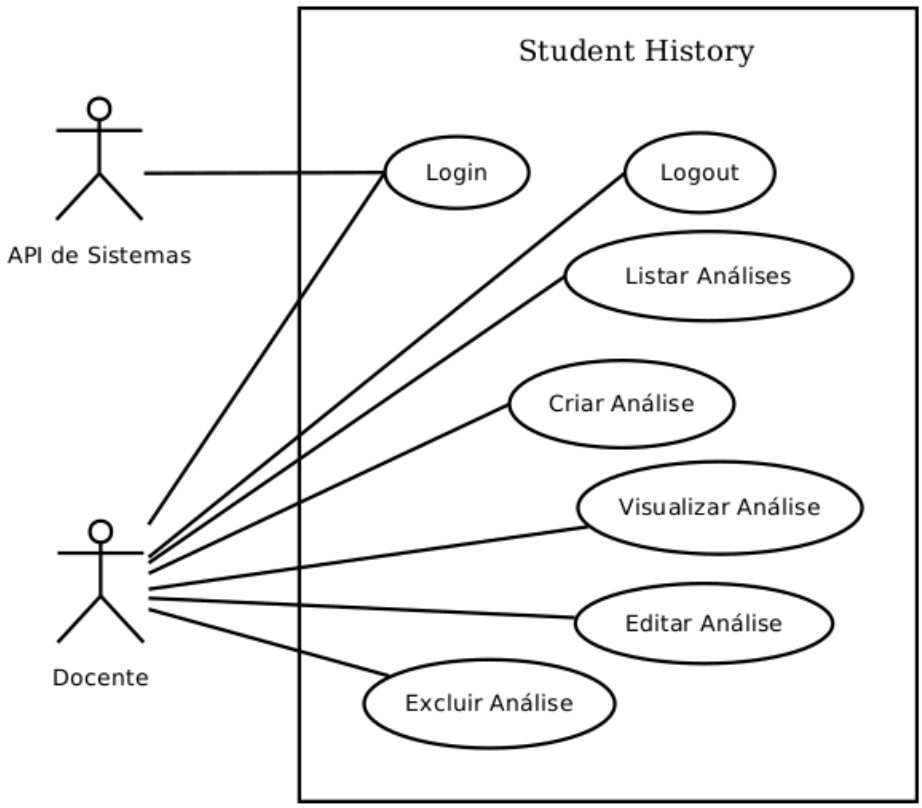
Um bom exemplo de caso de uso são os *login* em sistemas, onde temos a interação necessária por parte do usuário (ator), que seria digitar o *login* e senha e a partir daí aguardar o retorno do sistema. A importância de conhecer esses passos intercaláveis entre sistema e usuário é conseguir definir o que será preciso implementar no que se diz respeito a codificação para que esta funcionalidade se torne concreta.

A Figura 3 apresenta o diagrama de casos de uso do sistema descrito neste trabalho. O diagrama de casos de uso é representado pelos seguintes elementos: os bonecos simbolizam os atores que interagem com o sistema; cada balão simboliza um caso de uso; o retângulo que

envolve o conjunto de casos de uso simboliza o sistema apresentado pelo texto no topo da imagem.

O StudentHistory possui dois atores principais, o usuário final, que terá ação em todas funcionalidades do sistema e o sistema externo da Superintendência de Informática (SINFO) da UFRN que possui papel fundamental na funcionalidade de *login*, contribuindo na autenticação do docente. Inicialmente os usuários finais previstos serão docentes envolvidos com a coordenação, núcleo docente estruturante ou colegiado de curso da UFRN. Futuramente estes usuários também vão incluir pedagogos e demais profissionais que atuam na assistência pedagógica dos centros acadêmicos. Além destes dois atores, a Figura 3 apresenta também as sete principais funcionalidades do sistema, que são: *login*, *logout*, listar análises, criar análise, visualizar análise, editar análise e excluir análise.

Figura 3 - Diagrama de casos de uso do StudentHistory



Fonte: Própria (2018)

A seguir serão detalhados os casos de uso definidos para o sistema StudentHistory a fim de atingir seu objetivo principal de permitir um gerenciamento de análises de desempenho de estudantes nos cursos de graduação da UFRN.

2.2.1 Login SINFO

A definição da funcionalidade de login partiu principalmente da necessidade de identificar os autores das análises feitas na ferramenta. Além disso, é importante que o acesso seja restrito por envolver dados sigilosos. Essa restrição de acesso apenas para docentes da UFRN, foi um ponto decisivo para a inclusão da integração do login com as credenciais já existentes na SINFO, além de não forçar o usuário a lidar com um novo login específico para esta plataforma.

Este caso de uso é realizado em duas etapas. Na tela de login do sistema, existe uma opção para efetuar o login integrado com a SINFO. Após o usuário selecionar a opção, o sistema redireciona-o para um ambiente externo onde será feita a primeira verificação de suas credenciais, através dos campos usuário e senha, pela própria SINFO. Com a confirmação do login, a SINFO redireciona o usuário novamente para o sistema, que por sua vez, realiza a segunda etapa de autorização, permitindo acesso apenas a usuário docente e redirecionando-o para a página principal do sistema.

Fluxo principal:

1. Usuário seleciona a opção de login com a SINFO.
2. Sistema redireciona o usuário para a página de login da SINFO (API de Sistemas).
3. Usuário fornece login e senha.
4. SINFO (API de Sistemas) redireciona o usuário de volta para o sistema.
5. Sistema autoriza o usuário e redireciona para a visualização das análises.

Fluxo alternativo 1:

4a. Credenciais fornecidas não são válidas.

5a. SINFO (API de Sistemas) informa os erros de credenciais.

Fluxo alternativo 2:

5b. Usuário não é docente e a autorização falha.

6b. Sistema redireciona o usuário para a seleção de login.

2.2.2 Logout

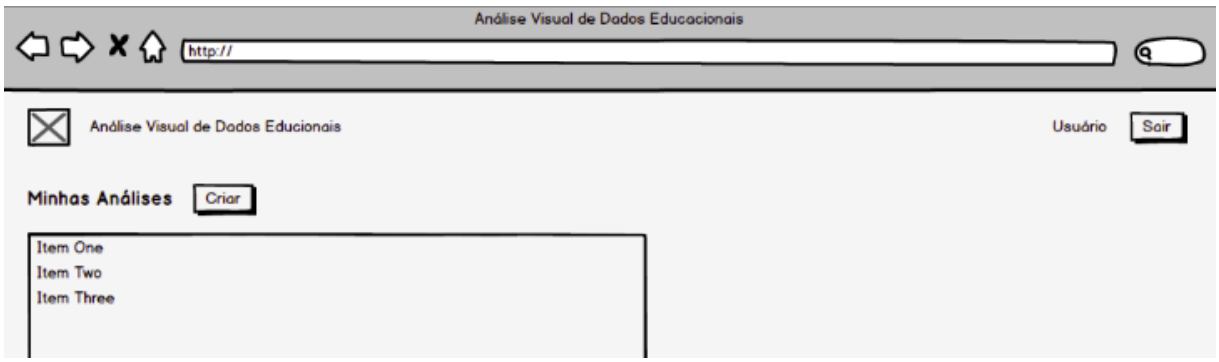
O sistema disponibiliza uma opção no menu principal com a função de encerrar a sessão do usuário.

Fluxo principal:

1. Usuário seleciona a opção de logout no menu principal do sistema.
2. Sistema encerra a sessão e redireciona o usuário para a tela de login.

2.2.3 Listar Análises

Figura 4 - Protótipo de tela da listagem de análises do StudentHistory



Fonte: Bruno Santana da Silva (2017)

A Figura 4 mostra um protótipo de tela da página principal do sistema, com a presença da funcionalidade de listagem das análises do usuário logado. Além disto, para cada análise existem as opções de visualizar, editar e excluir análise. Cada análise é apresentada com nome, curso analisado e ano e semestre de ingresso dos estudantes envolvidos na análise.

Fluxo principal:

1. Usuário seleciona a opção de análises no menu principal.

2. Sistema redireciona o usuário para a listagem de suas análises.

2.2.4 Criar Análise

Permite o usuário criar uma análise, contendo informações como: nome da análise, curso a ser analisado, ano e semestre de ingresso dos estudantes considerados na análise, disciplinas, ano e semestre das turmas de cada disciplina escolhida.

Fluxo principal:

1. Usuário ativa o botão de criar análise.

2. Sistema redireciona o usuário para uma tela com um formulário contendo campos para o nome da disciplina, curso, ano e semestre de ingresso dos estudantes que serão considerados na análise, disciplinas e ano e semestre em que cada disciplina foi ministrada.

3. Usuário preenche o formulário, seleciona as opções disponíveis e submete.

4. Sistema salva a análise e redireciona o usuário para a visualização da análise.

Fluxo alternativo 1:

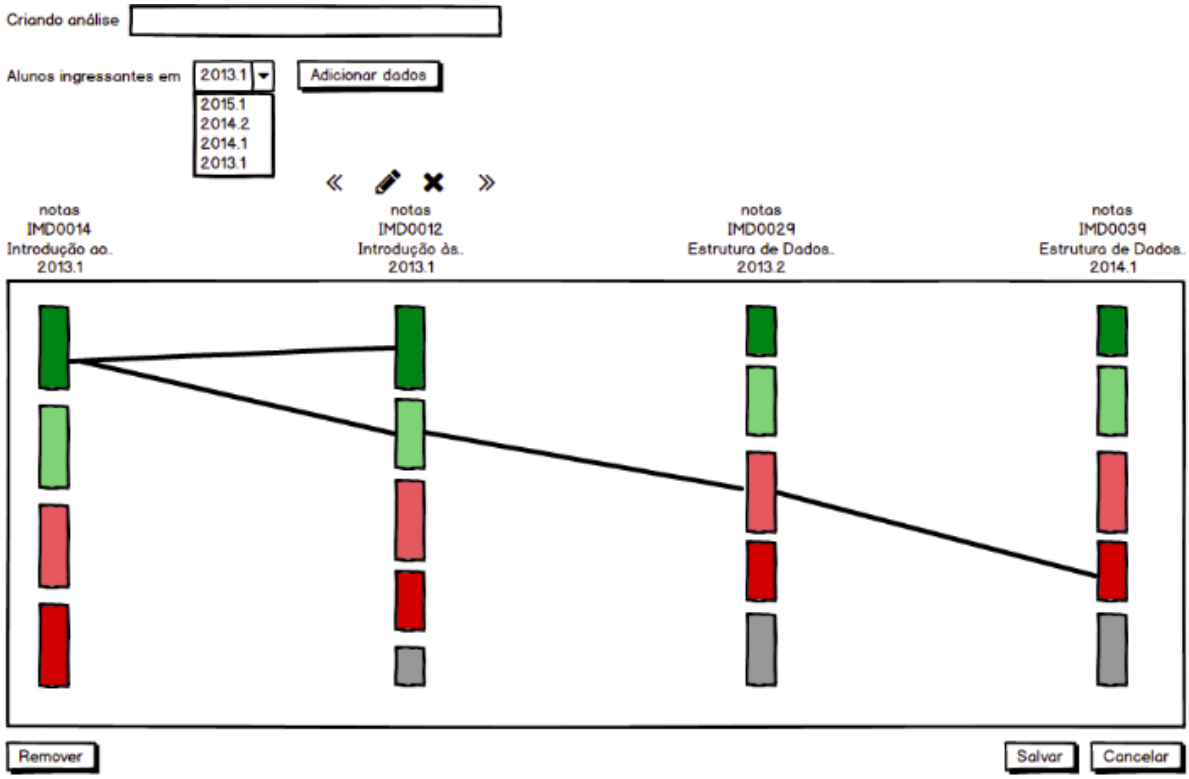
4a. Informações obrigatórias não são preenchidas pelo usuário.

5a. Sistema retorna mensagem informando os erros.

2.2.5 Visualizar Análise

Na Figura 5, temos um protótipo de tela que orientou a definição da funcionalidade de visualização de análises, contendo suas principais informações e um gráfico gerado pelo sistema contendo ligações entre as notas dos estudantes nas turmas de cada disciplina analisada.

Figura 5 - Protótipo de tela da criação/edição/visualização de análise no StudentHistory



Fonte: Bruno Santana da Silva (2017)

Fluxo principal:

1. Usuário seleciona o botão de visualizar análise presente na listagem.
2. Sistema redireciona o usuário para a tela de visualização da análise.

2.2.6 Editar Análise

Caso de uso que permite o usuário editar a definição de uma análise escolhida. Caso informações relativas a geração do gráfico sejam alteradas, um novo gráfico será mostrado.

Fluxo principal:

1. Usuário ativa o botão de editar análise presente na listagem.

2. Sistema redireciona o usuário para uma tela com um formulário já preenchido pelos dados da análise, contendo campos para o nome da disciplina, curso, ano e semestre de ingresso dos estudantes considerados, disciplinas e ano e semestre das turmas de cada disciplina escolhida anteriormente.

3. Usuário edita no formulário as informações desejadas e confirma edição.

4. Sistema salva a análise e redireciona o usuário para a visualização da análise.

2.2.7 Excluir Análise

Permite ao usuário a exclusão de uma análise escolhida. O sistema deleta do banco de dados a análise.

Fluxo principal:

1. Usuário ativa o botão de remoção a partir da listagem de análises.

2. Sistema exibe mensagem perguntando se o usuário deseja realmente excluir a análise.

3. Usuário confirma a remoção.

4. Sistema remove a análise do banco de dados e redireciona o usuário para a listagem atualizada de análises.

Fluxo alternativo 1:

3a. Usuário não confirma a remoção.

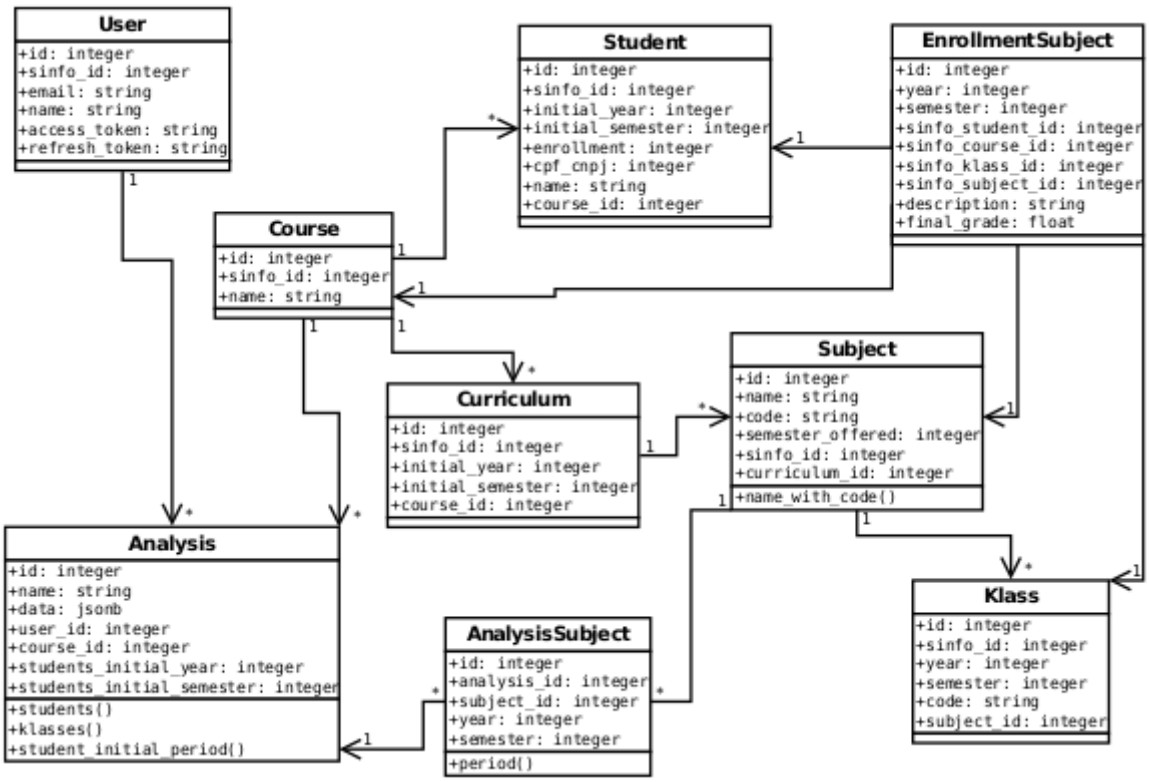
4a. Sistema cancela a ação e o usuário permanece na tela de listagem.

2.3 Diagrama de classes

Esta seção descreve as entidades e relacionamentos definidos para o domínio do projeto. A apresentação do diagrama de classes inclui também atributos e métodos importantes para a comunicação entre as entidades e a realização de suas responsabilidades. O

diagrama de classes é definido principalmente pelos seguintes elementos gráficos: caixas com três seções representam as classes, uma seção para o nome da classe, uma seção para os atributos e uma seção para os métodos; linhas ligando as caixas representam as relações entre elas, as ligações podem incluir a cardinalidade indicando se uma classe se relaciona com um ou muitos elementos representados por outra classe (LARMAN, 2007; BLAHA, 2006).

Figura 6 - Diagrama de classes do StudentHistory



Fonte: Própria (2018)

O diagrama de classes da Figura 6 representa o modelo de dados do StudentHistory. Temos que:

1. Um usuário possui zero ou mais análises.
2. Cada análise pertence a apenas um usuário.
3. Uma análise está relacionada a apenas um curso de graduação da UFRN.
4. Um curso possui zero ou muitos estudantes.
5. Um curso possui zero ou muitas matrizes curriculares.
6. Uma matriz curricular possui zero ou muitas disciplinas.
7. Cada disciplina possui zero ou muitas turmas.

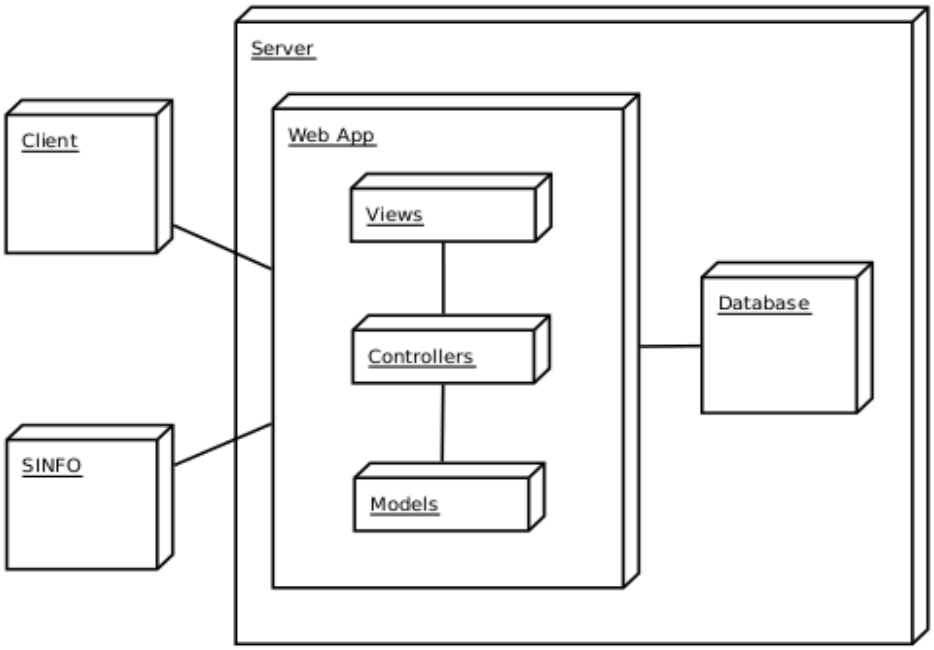
8. A partir do período de ingresso dos alunos escolhido na criação da análise é possível recuperar os alunos envolvidos na mesma, sendo eles subconjunto do total de estudantes do curso analisado.
9. A relação entre análise e disciplina é de muitas para muitas, ou seja, uma análise pode ter várias disciplinas que, por sua vez, também podem se relacionar a outras análises.
10. A partir de cada disciplina, ano e semestre escolhido para a mesma, é possível obter as turmas relevantes para a análise.
11. A classe *EnrollmentSubject* representa um registro da matrícula de um estudante em uma turma de determinado curso. Nela ficam salvos dados como descrição da situação atual, como, por exemplo, se um aluno foi aprovado ou não, e as médias finais. Média final será utilizada para dividir os estudantes em grupos, apresentados futuramente no gráfico Sankey.

2.4 Arquitetura do sistema

Esta seção apresenta a arquitetura projetada para o StudentHistory, com os principais componentes e suas responsabilidades na ferramenta.

Durante a fase de projeto do sistema, foi definido que a interação do usuário seria feita através de uma interface web. A arquitetura então, como mostrada na Figura 7, foi dividida entre: lado cliente, lado servidor, banco de dados e comunicação com o sistema externo da SINFO/UFRN (API de Sistemas da SINFO).

Figura 7 - Visão geral da arquitetura do StudentHistory



Fonte: Própria (2018)

A parte cliente é responsável por fazer as requisições http via browser para o sistema. É o ponto de partida para qualquer interação com a aplicação. Para toda requisição, uma resposta é aguardada contendo como resultado as telas do sistema e suas informações. Também é no lado cliente que é feita a renderização dos gráficos das análises presentes no StudentHistory, pois no futuro isso facilitará a evolução da ferramenta para exibir os mesmos dados de outras formas para promover uma compreensão mais rica do contexto educacional.

A parte servidor é responsável por receber as requisições do usuário, tratá-las e respondê-las. O tratamento dessas requisições inclui a comunicação com o banco de dados, bem como a comunicação com sistemas externos, que no caso do StudentHistory é feita com o sistema da SINFO/UFRN (API de Sistemas da SINFO). Após a recuperação de informações, a parte servidor responde a parte cliente com as páginas web e dados para exibição nos gráficos que permitirão ao usuário fazer uma análise visual adequada do desempenho dos alunos.

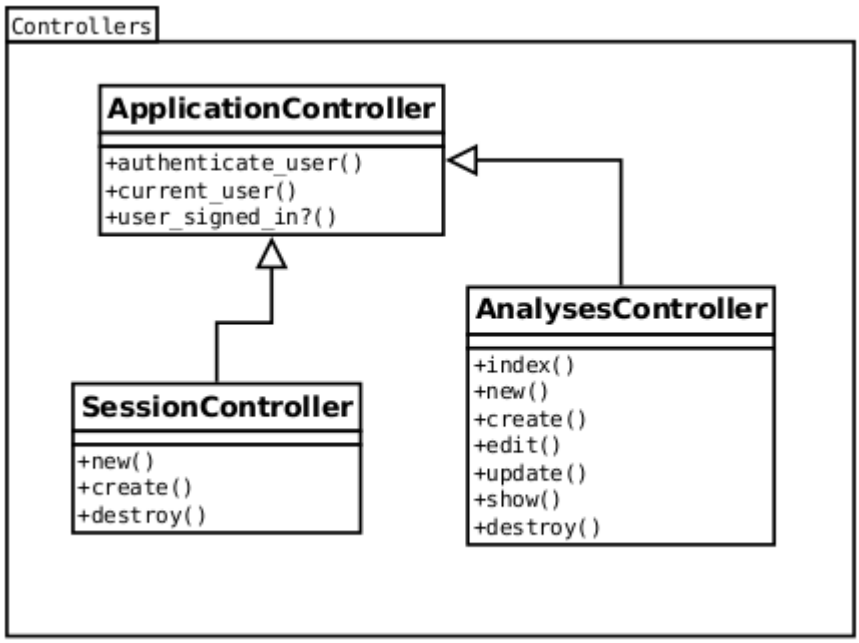
Para uma melhor divisão de responsabilidades da aplicação web, optou-se pela utilização do padrão arquitetural MVC (Model-View-Controller) que consiste em separar em camadas os componentes responsáveis pela manipulação dos dados, lógica de negócio e apresentação dos dados do sistema (BURBECK, 1987). A seguir expõe-se detalhes de cada

camada da aplicação web StudentHistory e como cada uma interage com as outras partes do sistema.

2.4.1 Controllers

O *controller* interpreta as entradas do mouse ou teclado do usuário e comanda o *model* e/ou a *view* à efetuar as mudanças apropriadas (BURBECK, 1987). A Figura 8 representa a camada de *controllers* da aplicação no StudentHistory, contendo cada *controller* e suas principais ações. A figura também mostra a relação de herança entre os *controllers*, representada pelas setas direcionadas. Três *controllers* lidam com as requisições do sistema: *ApplicationController*, *SessionController* e *AnalysesController*.

Figura 8 - Camada de controllers do StudentHistory



Fonte: Própria (2018)

O *ApplicationController* é o pai de todos os outros *controllers*, ou seja, todos os outros *controllers* derivam dele e compartilham suas características e métodos. Nele temos alguns métodos importantes para controle de acesso dos usuários. Sendo eles:

1. ***authenticate_user***

Responsável por verificar se um usuário está logado ou não e executar o devido redirecionamento para a tela de *login* em caso negativo. Isso impede que usuários não logados tenham suas requisições respondidas.

2. current_user

Recupera o usuário atual da sessão caso haja algum usuário logado.

3. user_signed_in?

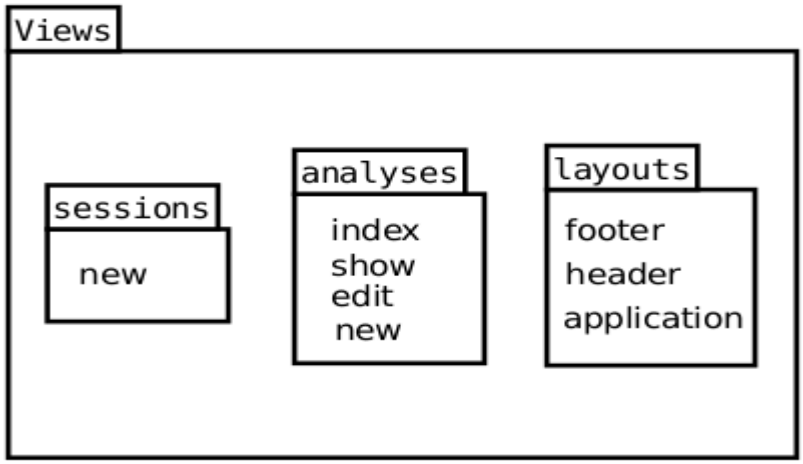
Retorna um *boolean* para verificar se existe um usuário logado.

SessionController por sua vez, é responsável por receber as requisições para novas sessões (*login*) e a destruição das mesmas (*logout*). Já o *AnalysesController* lida com as requisições feitas pelo usuário para criação, visualização, edição e destruição de análises.

2.4.2 Views

As views do padrão MVC gerenciam a saída gráfica e/ou textual na interface com usuário da aplicação (Burbeck, 1987). A camada de visualização é responsável por captar as interações do usuário e por apresentar as telas do sistema, desde listagens, formulários, menus, layouts, páginas de visualização.

Figura 9 - Camada de visualização da aplicação StudentHistory



Fonte: Própria (2018)

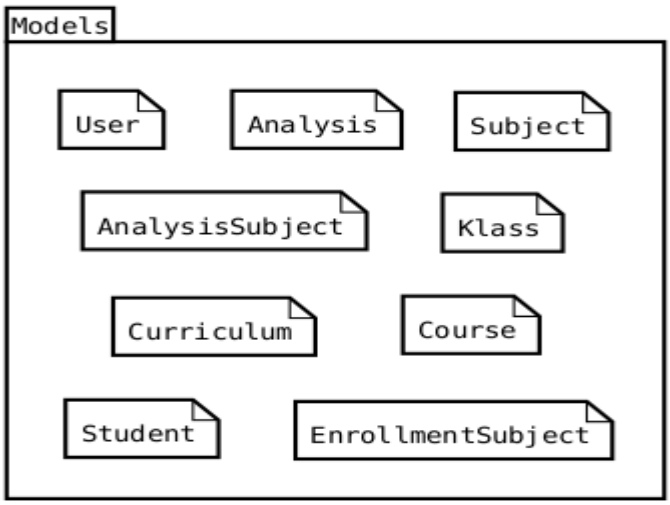
A Figura 9 expõe a camada de visualização do StudentHistory, dividida da seguinte forma: arquivos de *layout* contendo os elementos visuais compartilhados entre as telas do sistema, como é o caso do menu de navegação e o *footer* das páginas; arquivos responsáveis por apresentar a tela para nova sessão (*login*); arquivos responsáveis por apresentar a listagem das análises do sistema, formulários de criação e edição das análises, bem como a visualização de uma análise específica (geração dos gráficos a partir dos dados recebidos do servidor).

2.4.3 Models

Os *models* gerenciam o comportamento e os dados do domínio da aplicação, respondem às requisições por informações sobre seu estado, e respondem às instruções de alterações de estado (Burbeck, 1987).

Os *models* separam o resto da aplicação da manipulação dos dados que se encontram no banco de dados. São nesses arquivos de modelo que se encontram as validações para os campos e seus tipos e métodos responsáveis por alterar ou recuperar informações. Cada *model* é uma classe que representa uma tabela da base de dados do sistema, garantido a definição de responsabilidade única.

Figura 10 - Camada de modelos da aplicação StudentHistory



Fonte: Própria (2018)

A Figura 10 mostra os modelos do sistema descrito no presente trabalho. Cada *model* deve ser capaz de recuperar, alterar, inserir, remover dados da sua tabela correspondente no banco de dados. Além disso, deve garantir que regras de negócio sejam atendidas, como por exemplo, o limite máximo de caracteres de um campo de texto.

3 Desenvolvimento

Este capítulo descreve o desenvolvimento do StudentHistory. Está organizado da seguinte forma: etapas do desenvolvimento da ferramenta, que inclui decisões tomadas frente a obstáculos que surgiram durante a implementação; descrição das tecnologias escolhidas para implementar o sistema; apresentação do resultado obtido após a implementação.

3.1 Etapas do desenvolvimento

Com base nos requisitos definidos para o sistema e nas decisões tomadas durante o projeto iniciou-se o processo de desenvolvimento da ferramenta de modo incremental e iterativo.

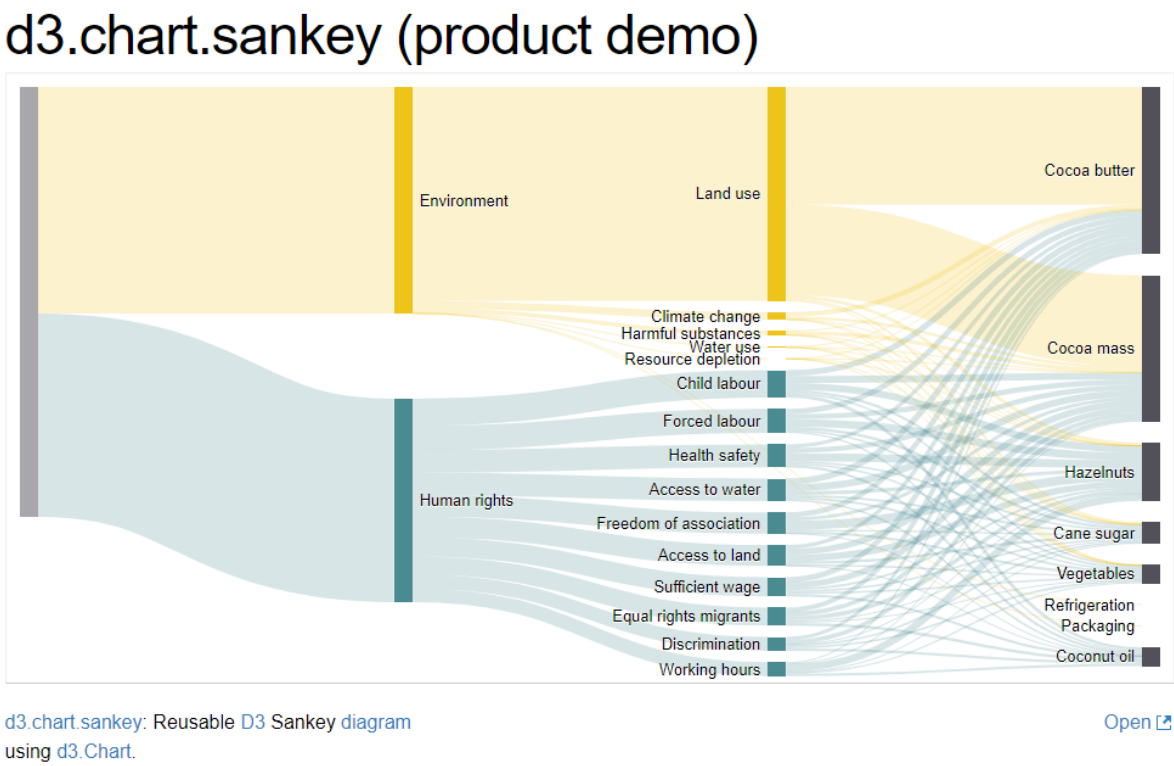
O primeiro passo do desenvolvimento foi selecionar algumas ferramentas e tecnologias que permitiram a concretização do projeto. Por se tratar de uma aplicação web dividida em camadas e seguir o padrão arquitetural MVC, a linguagem Ruby e o framework MVC Ruby on Rails foram escolhidos para a implementação. Uma primeira versão da ferramenta foi desenvolvida conforme a arquitetura projetada. Era possível realizar *login*, *logout*, listar e criar análises. As análises criadas nesta primeira versão só continham nome e ainda não contavam com a geração do gráfico esperado para sua versão final.

O próximo passo foi buscar opções de bibliotecas que fossem capazes de gerar o gráfico projetado. Com alguma pesquisa na web, duas bibliotecas javascript foram cogitadas por disponibilizarem plugins para exibição de diagramas Sankey: Google Charts e D3js. Apesar do resultado com diagramas sankey ser bastante parecido, D3js foi escolhido, principalmente pela necessidade de customização do que era oferecido como padrão nessas bibliotecas e por oferecer maior controle sobre dados sigilosos.

A definição de um diagrama Sankey é feito pela especificação de nós e relações entre eles, como em um grafo. O posicionamento dos nós no diagrama Sankey dessas bibliotecas em javascript é feito automaticamente e leva em conta as origens e destinos das relações entre os nós. Tomou-se como base o exemplo de d3.chart.sankey desenvolvido por wvengen (Figura 11). Nesta implementação do diagrama Sankey usando D3js, os nós que possuem

apenas ligações partindo deles, ou seja, apenas originam relações, são remanejados totalmente à esquerda, enquanto os nós que são apenas destino de relações são posicionados mais à direita possível no diagrama. Esse comportamento não condiz com o resultado que o StudentHistory gostaria de apresentar por não manter os nós em posições fixas para as disciplinas. Portanto, foi necessário alterar a implementação da biblioteca javascript original para chegar no gráfico sankey projetado para determinar a posição dos nós (notas nas disciplinas) e rótulos das colunas (informações sobre a respectiva disciplina).

Figura 11 - Exemplo de diagrama Sankey renderizado pela biblioteca D3js.



Fonte: https://bl.ocks.org/wvengen/cab9b01816490edb7083

Em outro momento, sabendo-se que os envolvidos no projeto fazem parte da UFRN, foi decidido que os dados que permitem a criação de análises partiriam da base de dados da própria UFRN. Para isso, iniciou-se um estudo das informações que precisavam ser recuperadas e como fazer essa recuperação. A consulta dos dados necessários ao StudentHistory foi feita a partir da API de Sistemas da SINFO que disponibiliza acesso a

dados abertos da universidade e possui uma documentação sobre todos os serviços disponíveis e instruções de utilização.

A princípio, a ideia era recuperar os dados relacionados aos cursos sob demanda durante o processo de criação da análise. Entretanto, foram descobertas algumas limitações técnicas. Será necessária uma grande quantidade de requisições à API de Sistemas da SINFO para recuperar dados dos cursos, alunos, disciplinas, turmas e notas dos alunos para cada análise. Durante o estudo do desempenho dos alunos, é muito provável que o usuário faça várias análises similares em seguida para aprofundar e ampliar sua compreensão sobre a realidade dos alunos. Essas análises consecutivas tendem a ser similares, compartilhando curso, disciplinas, turmas e alunos. Se os dados sempre fossem buscados diretamente da API de Sistemas da SINFO ocorreriam várias requisições repetidas, afetando a eficiência do StudentHistory (por ser mais demorado recuperar da API do que de um repositório local da aplicação) e aumentando significativamente a carga de trabalho do servidor que oferece os serviços da API. Além disso, existem limitações importantes da API de Sistemas da SINFO que provavelmente foram criadas para evitar sobrecarga no servidor. Cada requisição feita à API retorna no máximo 100 registros por vez. Se a aplicação precisar recuperar mais registros terá que realizar outras requisições utilizando um *offset* (indicação da posição do primeiro registro retornado). A API de Sistemas da SINFO também impõe um limite de 5000 requisições por hora.

Por conta dessas limitações da API de Sistemas da SINFO, uma camada de serviços na arquitetura foi introduzida, para comportar primeiramente o serviço de importação dos dados (*RetrieveCourseInformation*) utilizados na criação das análises, o qual precisa ser executado antes do uso do sistema pelos usuários. O *RetrieveCourseInformation* executa uma série de requisições para a API de Sistemas da SINFO e salva os resultados em um banco de dados localizado na parte servidor do StudentHistory. São recuperadas quase todas as informações necessárias para análise de desempenho de alunos de graduação da UFRN: informações sobre curso, alunos, matrizes curriculares, disciplinas e turmas. Depois de importadas, essas informações são disponibilizadas para criação das análises e seus respectivos gráficos pelo usuário.

Durante o estudo da API de Sistemas da SINFO, não foi possível identificar serviço da API que recuperasse as notas dos alunos, ponto determinante para análise do desempenho dos alunos nas disciplinas da UFRN. A SINFO informou que as notas dos alunos são dados

sigilosos e não são disponibilizados através da API. Sempre que alguém precisar analisar as notas dos alunos deve fazer uma solicitação formal específica para a SINFO determinando quais dados necessita. O acesso a estes dados sigilosos envolve assinatura de um termo de compromisso por todos os envolvidos com o sistema. Deste modo, para a realização deste trabalho foi solicitado à SINFO as notas de todos os ingressantes do Bacharelado em Tecnologia da Informação (BTI) entre 2013 e 2017 para todas as turmas oferecidas neste período. Os envolvidos com o sistema assinaram um termo de compromisso para garantir o anonimato e privacidade dos alunos durante o desenvolvimento do StudentHistory.

A SINFO, então, respondeu à solicitação entregando um arquivo CSV (*comma-separated values*) com as notas e *status* da matrícula (aprovado ou reprovado). Por isso, outro serviço (*ImportGrades*) foi criado no sistema para importar os dados do arquivo CSV para o banco de dados do StudentHistory. Assim como o primeiro serviço mencionado, o *ImportGrades* também precisa ser executado antes do usuário utilizar o sistema. A versão pós implementação do StudentHistory permite análises referentes ao curso de Bacharelado em Tecnologia da Informação da UFRN. Para que o sistema permita análises feitas sobre dados de outros cursos da UFRN é necessário realizar a importação de dados e a solicitação do arquivo CSV referentes a esses cursos.

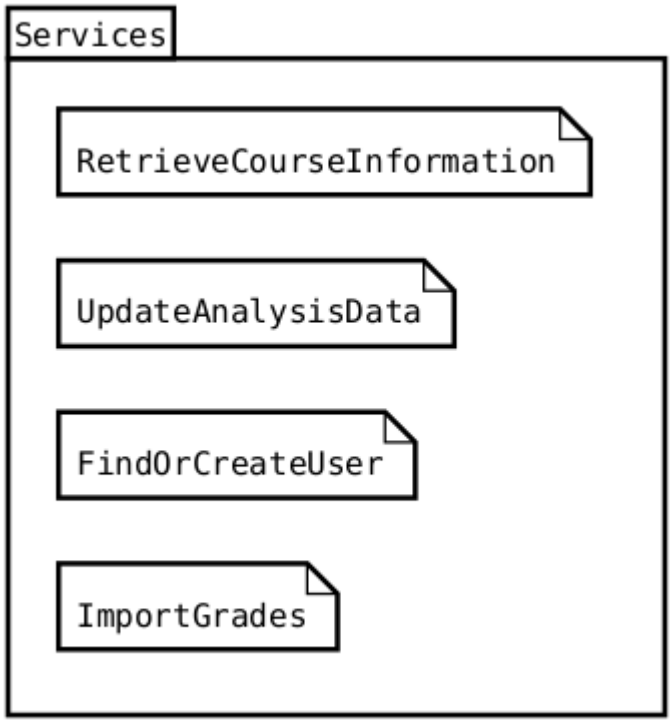
Devido a existência de dados sigilosos no StudentHistory e a restrição de acesso à docentes da UFRN, tornou-se ainda mais necessária a integração da funcionalidade de *login* do sistema com o *login* do SIGAA e a verificação dos vínculos do usuário. As funcionalidades de *login* e *logout* que estavam presentes na primeira versão do sistema foram modificadas para satisfazer esse requisito. Parte do processo de autenticação e autorização do usuário foi movida para o serviço *FindOrCreateUser* que é responsável por identificar o usuário que efetuou o *login* na SINFO, verificar seus vínculos na UFRN, registrar ou recuperar o usuário no banco de dados (para saber quem criou as análises) e retornar para o controlador de sessão do sistema.

Após a importação dos dados, quando o banco de dados já possuía as informações necessárias para a criação de uma análise, houve um novo ciclo no processo de desenvolvimento para implementar as funcionalidades de edição, remoção e visualização das análises (dos gráficos). Também foi alterado o formulário de criação da análise para que o usuário escolha quais informações existentes no banco de dados são relevantes para seu interesse no momento: curso, alunos, turmas, notas, etc.

Por fim, como último ciclo na programação do StudentHistory, os esforços foram direcionados para a integração da biblioteca javascript D3js com o sistema para gerar o diagrama Sankey na parte cliente da ferramenta durante a visualização das análises. Para isso, foi implementado o serviço *UpdateAnalysisData* responsável por filtrar o conjunto de dados relevantes para a análise, a partir dos parâmetros selecionados pelo usuário no formulário, e agrupar estes dados para que possam ser enviados ao lado cliente da aplicação e utilizados pelo D3js para criar o diagrama Sankey.

A Figura 12 mostra a camada de serviços presente na arquitetura do sistema com os quatro serviços explicitados anteriormente: *RetrieveCourseInformation*, *ImportGrades*, *FindOrCreateUser* e *UpdateAnalysisData*.

Figura 12 - Classes de serviços do sistema StudentHistory



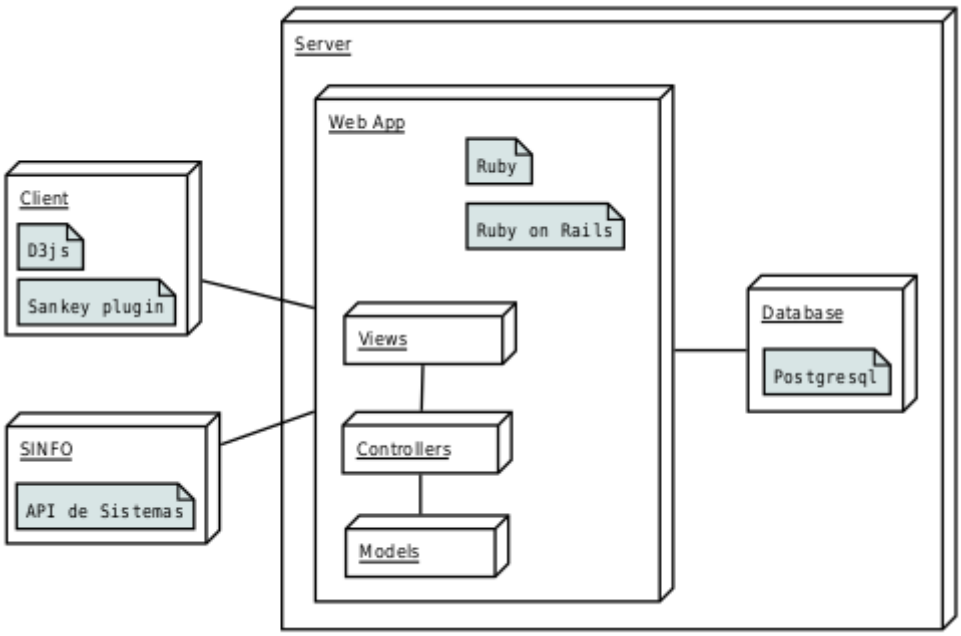
Fonte: Própria (2018)

3.2 Tecnologias Utilizadas

Nesta seção serão apresentadas as tecnologias utilizadas durante o desenvolvimento do StudentHistory.

A Figura 13 apresenta tecnologias utilizadas de acordo com a divisão arquitetural do StudentHistory. No lado cliente, destacam-se a biblioteca D3js e o *plugin* Sankey utilizados na criação dos gráficos mostrados aos usuários. No lado servidor, a linguagem de programação Ruby e o *framework* Ruby on Rails foram utilizados na implementação da aplicação *web*. Além disso utilizou-se a API de Sistemas da SINFO, sistema externo responsável por atuar na autenticação dos usuários, bem como alimentar a base de dados construída e gerenciada com Postgresql.

Figura 13 - Tecnologias utilizadas em cada camada da arquitetura do StudentHistory



Fonte: Própria (2018)

Apesar do desenvolvimento do StudentHistory ser individual, o uso de um sistema de controle de versão teve como objetivo facilitar a manipulação dos arquivos e permitir o acompanhamento do histórico de mudanças no código fonte em diferentes máquinas.

As tecnologias utilizadas serão descritas, individualmente, a seguir.

3.2.1 Sistema de Controle de Versão

Como sugere o nome, os conhecidos SCM (*Source Configuration Management*), são utilizados amplamente na engenharia de software para versionar o código fonte. Isso garante uma maior segurança no controle das mudanças realizadas durante o processo de desenvolvimento como um todo.

O processo de criação de versões para cada ponto que o projeto se encontra pode ser facilmente comparado a uma foto de um instante em que se encontram todos os arquivos presentes no código.

Além de contar com um histórico e seus responsáveis, as alterações podem ser revertidas ou rearranjadas de maneira mais consciente pelos desenvolvedores. Em um ambiente colaborativo, onde várias pessoas trabalham paralelamente no mesmo projeto, a importância de um sistema de controle se torna muito mais evidente, pois a resolução de conflitos para alterações em um mesmo arquivo e a documentação das mesmas, permite um melhor controle do que está sendo realizado no código.

Este trabalho utilizou um dos mais populares sistemas de controle de versão existentes atualmente. O GIT possui um modelo distribuído, que consiste na não-centralização do código fonte em apenas uma máquina, apesar disso ser possível através de convenções determinadas dentro da equipe de desenvolvimento.

Os locais onde são armazenados os códigos fonte são chamados de repositório. No caso do GIT, cada máquina se torna um repositório, onde o processo de trabalho independe de outras pessoas. Existem também os chamados repositórios remotos, que podem ser utilizados como centralizadores, para uma maior organização do trabalho, como por exemplo, servir apenas para o envio de mudanças definitivas. Repositórios remotos podem ser públicos ou privados, gratuitos ou pagos e são amplamente utilizados nos diversos fluxos de trabalho que cada time de desenvolvimento define. O código fonte do StudentHistory está hospedado no GitLab do IMD de forma privada.

3.2.2 Ruby

Ruby é uma linguagem de programação, criada por Yukihiro “Matz” Matsumoto, que une algumas características de linguagens como Perl, Smalltalk, Eiffel, Ada e Lisp. Foi apresentada ao público em 1995 e se tornou bastante popular em 2005/2006 em grande parte pela popularidade dos softwares escritos na linguagem, principalmente o meta-framework Ruby on Rails.

É uma linguagem *open-source*, interpretada, de tipagem dinâmica e forte. Tudo em Ruby é um objeto, que herdam da classe Object, até mesmo os tipos primitivos. Suporta apenas herança simples, mas possui o conceito de *Mixins*, ou seja, permite que módulos e seus métodos sejam incluídos (“herdados”) por outras classes.

Como na maioria das linguagens de programação, Ruby utiliza um grande conjunto de bibliotecas de terceiros. A maioria delas são distribuídas em forma de uma “gem”. O Ruby possui um gerenciador de pacotes similar ao “apt-get” de sistemas Linux, chamado RubyGems, voltado inteiramente para software em Ruby.

3.2.3 Ruby on Rails

Rails é um framework MVC escrito na linguagem de programação Ruby, focado no desenvolvimento ágil de aplicações web orientadas a banco de dados. Dentro da comunidade Rails é comum priorizar convenção ao invés de configuração.

O fluxo básico das requisições no Rails acontece da seguinte forma:

1. Usuário faz uma requisição através do *browser* para uma URL.
2. O *router* do rails identifica e solicita para o *controller* e *action* responsáveis por atender aquela requisição.
3. O *controller* por sua vez acessa o *model* e manipula os dados necessários para passar para a *view* adequada.
4. A *view* estrutura estes dados recuperados e o resultado disso é enviado de volta para o browser do usuário.

A Figura 14 apresenta a estrutura de arquivos em uma aplicação Rails. O diretório *app* é a parte central para o código da aplicação. É nesse diretório que se encontram as pastas específicas para *models*, *views* e *controllers* da arquitetura MVC. O diretório “*app/assets*” é

responsável por manter os arquivos estáticos utilizados na parte cliente da aplicação, ou seja, imagens, arquivos de estilo e *javascripts*. Os *assets* externos à aplicação, por exemplo, javascripts importados de outros locais, ficam localizados no diretório *vendor*.

Figura 14 - Estrutura do diretório de aplicação Rails chamada *rails_folders*.

Fonte: Própria (2018)

3.2.4 D3js

D3js é uma biblioteca JavaScript para manipulação de documentos baseados em dados desenvolvidos com HTML, SVG e CSS. D3js permite associar dados arbitrários a um DOM (*Document Object Model*) e aplicar transformações orientadas a dados neste documento. Por exemplo, D3js é capaz de gerar uma tabela HTML a partir de um array de números, ou usar esse mesmo conjunto de dados para criar um gráfico em barra interativo com SVG.

A manipulação de elementos DOM usando esta biblioteca é sintaticamente parecida com JQuery e empregada de maneira declarativa, operando em seleções dos elementos de uma página. Apesar da semelhança com JQuery, estilos, atributos e outras propriedades podem ser especificadas como funções de dados, não apenas simples constantes.

3.2.4.1 Sankey

O diagrama Sankey da biblioteca javascript D3js é formado por *nodes* (nós) e *links* (ligações). Cada *node* pode ser origem e destino de *links*, formando assim, um fluxo com as ligações entre os nós do diagrama.

Para utilização do diagrama Sankey pelo D3js, um arquivo *javascript* precisa ser importado no sistema, em forma de *plugin*, contendo as instruções necessárias para a criação e manipulação do diagrama. O código *javascript* recebe dados em formato JSON recuperado da parte servidor do StudentHistory para desenhar o diagrama.

A criação do diagrama Sankey é feita da seguinte forma: define-se a largura e altura da porção de tela responsável por exibir o diagrama; proporcionalmente aos dados, desenha-se cada nó com determinada largura e altura, os títulos de cada nó e as devidas configurações como, por exemplo, cores dos nós; desenha-se também as ligações entre os nós e seus devidos rótulos.

No StudentHistory, foi realizada uma pequena alteração no *plugin* d3.chart.sankey, a fim de permitir o posicionamento fixo de nós. Também foi inserida uma verificação relacionada a presença de ligações de origem e destino em cada nó, para escondê-los caso não possuam ligações com outros nós.

3.2.5 Postgresql

Postgresql é um sistema gerenciador de banco de dados objeto-relacional, de código aberto, que usa e estende a linguagem SQL (Standard Query Language). O Postgresql funciona em grandes sistemas operacionais como Linux, Windows e Mac e sua origem data de 1986. Suporta diferentes tipos de dados desde primitivos à documentos e tipos de dados customizados como, por exemplo, inteiros, *string*, booleano, *jsonb*, *hstore* (chave-valor), etc; transações, concorrência, estratégias para melhora de performance. Além das muitas funcionalidades que o postgresql oferece, possui uma comunidade de desenvolvedores espalhada pelo mundo todo que contribuem para sua manutenção.

3.2.6 API de Sistemas da SINFO

A API de Sistemas da SINFO é uma interface de programação que permite operar sob os dados gerados pelos sistemas da UFRN. Através da API da SINFO é possível recuperar diversos tipos de informações da UFRN como, por exemplo, cursos, estudantes, disciplinas, professores, e mais uma variedade de dados que podem ser encontrados na documentação dos serviços da mesma.

A utilização da API é dividida em ambientes de teste e produção. É necessário preencher previamente um formulário para cadastrar a aplicação que utilizará a API e como resposta ao cadastro, a SINFO disponibiliza algumas chaves e identificadores da aplicação que serão usados no processo de autenticação e na geração de *token* de acesso através do qual é possível consumir os serviços da API.

A API de Sistemas possui dois métodos de autenticação: *client credentials* e *authorization code*. O método *client credentials* foi implementado para ser usado por aplicações que querem acessar dados públicos dos sistemas da SINFO, tais como eventos, notícias, telefones, entre outros e seu fluxo é realizado em duas etapas, primeiramente é feita uma requisição ao servidor de autorização da SINFO que retorna, dentre outras informações, o *token* de acesso. O método de autenticação *authorization code* é utilizado quando as aplicações querem acessar dados privados das contas de usuários dos sistemas SINFO, por isso durante o fluxo desse método é exigido o preenchimento das credenciais do usuário antes da geração do *token* de acesso.

O StudentHistory faz uso do método de autenticação *authorization code* e suas principais requisições à API são relacionadas a recuperação dos seguintes dados: *token* de acesso, dados pessoais e vínculos do usuário; cursos da universidade, estudantes, matrizes curriculares, disciplinas e turmas. Todos os registros do banco de dados da SINFO possuem identificador único. Esses identificadores são salvos na base de dados do StudentHistory para o devido relacionamento entre as entidades quando necessário.

Os principais *endpoints* e serviços da API de Sistemas da SINFO, bem como suas responsabilidades, utilizados pelo StudentHistory foram:

1. **GET /usuarios/info**. Recupera nome, email e identificador único do usuário logado.
2. **GET /vinculos**. Recupera os vínculos de um usuário.

3. **GET /cursos**. Recupera nome e identificador único de um curso.
4. **GET /matrizes-curriculares**. Recupera ano inicial, semestre inicial e identificador único das matrizes curriculares de um curso.
5. **GET /componentes-curriculares**. Recupera nome, código, semestre ofertado e identificador único dos componentes curriculares de uma matriz curricular.
6. **GET /turmas**. Recupera ano, semestre, código e identificador único das turmas de um componente curricular.

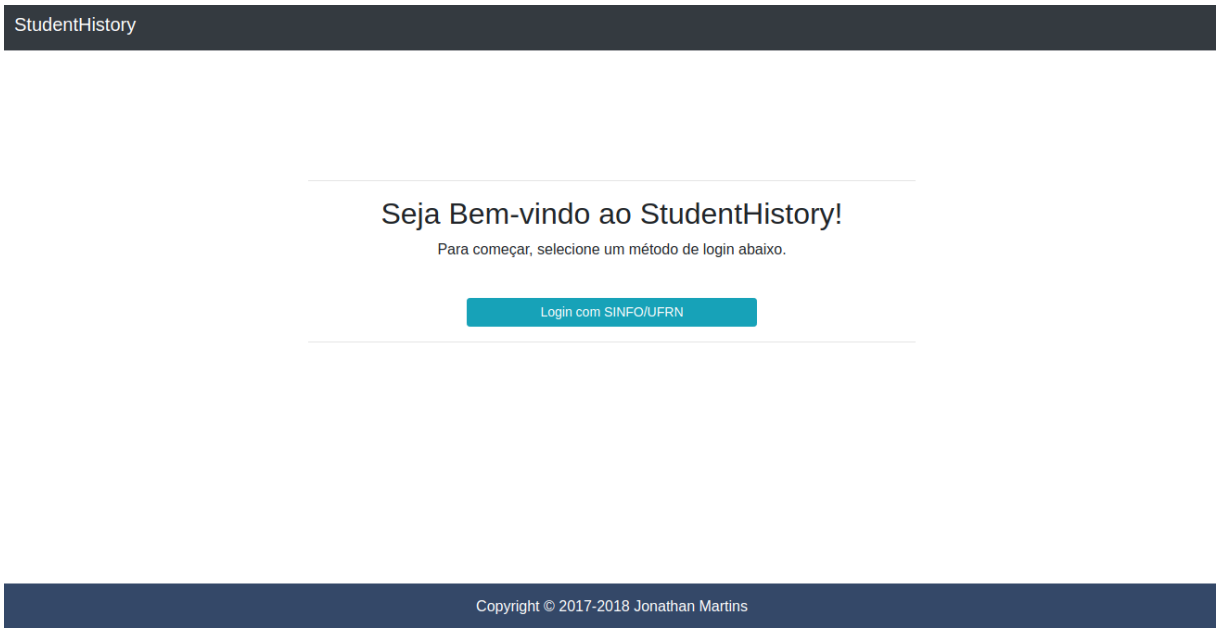
3.3 StudentHistory

Esta seção apresenta as telas da última versão da ferramenta StudentHistory como resultado da implementação dos casos de uso.

3.3.1 Login

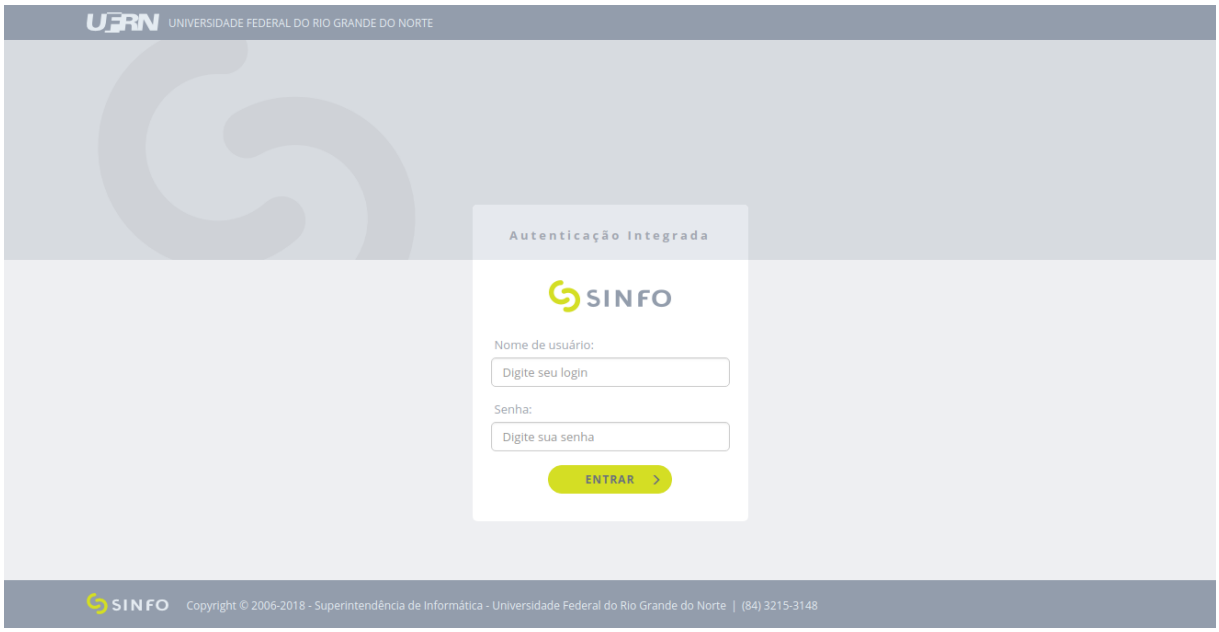
As Figuras 15 e 16 mostram o fluxo de telas que o usuário passa para realizar o login do sistema. A tela de *login* do StudentHistory, mostrada na Figura 15, possui um botão que permite o acesso do usuário ao sistema utilizando as credenciais do SIGAA. Ao clicar neste botão, o usuário é redirecionado para a tela mostrada na Figura 16, onde lhe é exigido a inserção de nome de usuário e senha do SIGAA. O fluxo segue então para o próximo caso de uso, Listagem de Análises.

Figura 15 - Tela de login do StudentHistory



Fonte: Captura de tela feita pelo autor (2018)

Figura 16 - Tela de login da SINFO



Fonte: Captura de tela feita pelo autor (2018)

3.3.2 Listagem de Análises

A Figura 17 expõe a tela de listagem das análises do usuário, página principal do StudentHistory. O usuário é redirecionado para esta página após um *login* bem-sucedido. A partir desta tela é possível acessar todas as outras funcionalidades do sistema. O menu superior apresenta opção para retornar para a listagem das análises (menu “Minhas Análises”), caso seja necessário. Também no menu superior é possível identificar o usuário logado e opção para sair do sistema, através da funcionalidade de *logout*.

Figura 17 - Tela de listagem de análises do StudentHistory

Fonte: Captura de tela feita pelo autor (2018)

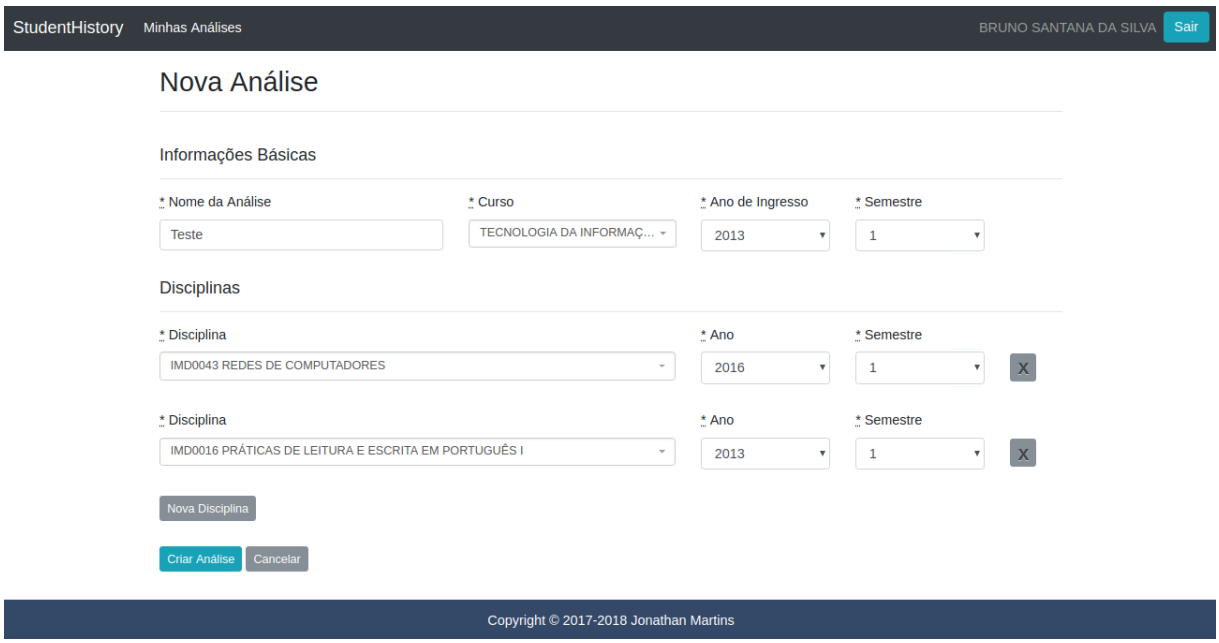
No corpo da tela é mostrada uma tabela com as análises do usuário e os seus respectivos identificadores, nomes, cursos analisados, ano e semestre de ingresso dos grupos de estudantes relevantes para as análises, bem como botões de ações para visualizar, editar ou excluir uma determinada análise. Para o caso de exclusão de uma análise, uma mensagem de confirmação é mostrada ao usuário antes da finalização da operação. Além das análises e suas ações também é possível encontrar na tela o botão que permite a criação de uma nova análise.

3.3.3 Criação e Edição de Análise

As Figuras 18 e 19 mostram, respectivamente, os formulários de criação e edição de análises. Essas duas telas são visualmente parecidas. Suas principais diferenças são: os títulos das páginas sendo uma para nova análise e uma para editar análise; e os botões de confirmação para criar ou atualizar a análise. Também existem diferenças no comportamento. Na edição o formulário inicia preenchido com a definição anterior da análise e atualiza os registros no banco de dados no final, enquanto que na criação o formulário inicia vazio e insere novos registros no banco de dados no final. Com relação aos formulários em si, para ambas as funcionalidades, as telas apresentam uma seção de informações básicas da análise, contendo nome da análise, curso a ser analisado, ano e semestre de ingresso dos alunos que serão levados em consideração na análise.

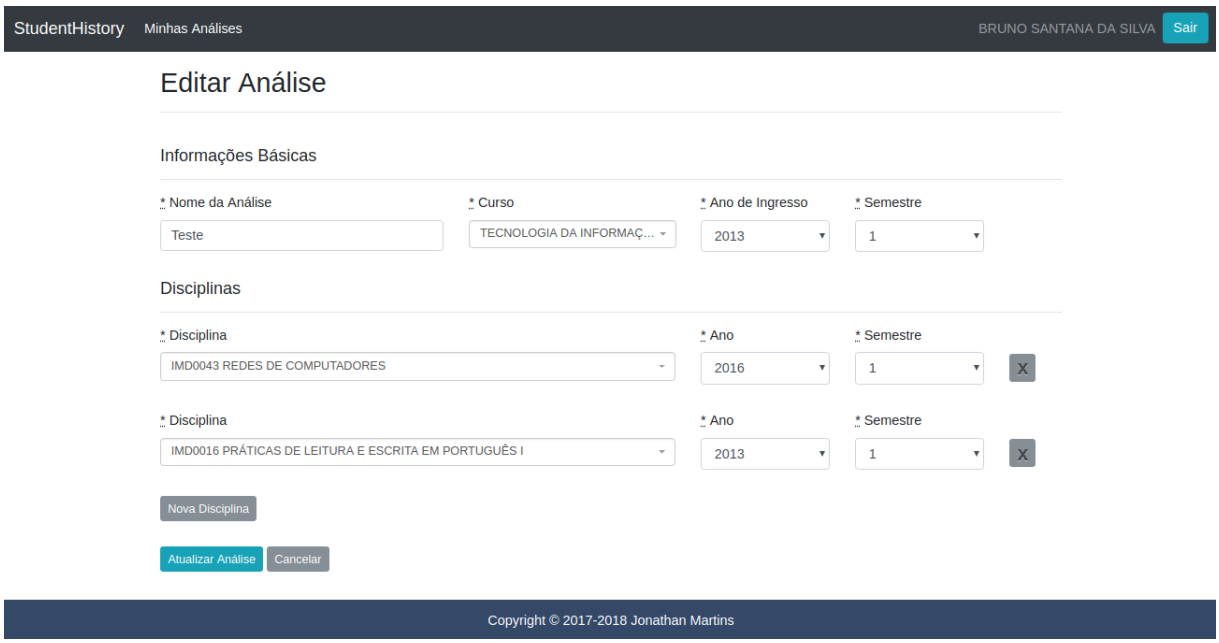
Além das informações básicas, na seção de disciplinas da análise, o usuário pode inserir ou remover disciplinas, com a restrição de pelo menos duas ou no máximo cinco disciplinas por análise, para uma melhor organização visual do gráfico gerado. Em cada disciplina no formulário, o usuário pode buscar por nome ou código da disciplina e selecionar o ano e semestre das turmas ministradas para a mesma.

Figura 18 - Tela de criação de análise do StudentHistory



Fonte: Captura de tela feita pelo autor (2018)

Figura 19 - Tela de edição de análise do StudentHistory

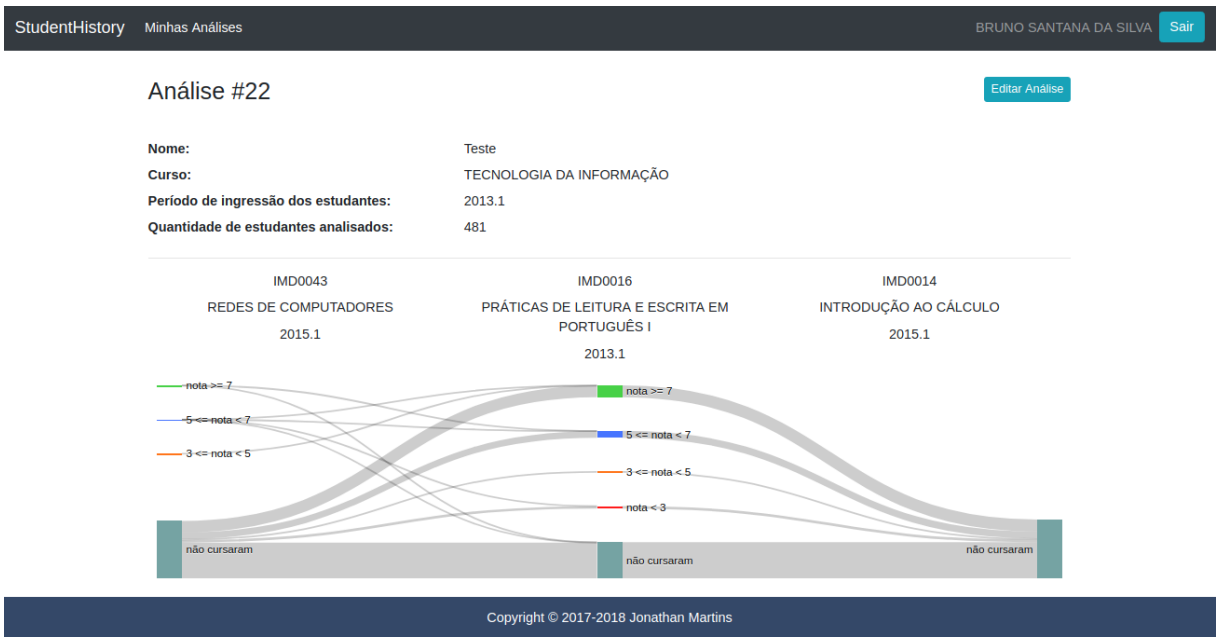


Fonte: Captura de tela feita pelo autor (2018)

3.3.4 Visualização de Análise

A Figura 20 revela a tela da funcionalidade principal do StudentHistory, a visualização de análise. Nesta tela é possível identificar as informações básicas da análise, como nome, curso analisado, período de ingresso dos alunos, quantidade total de alunos considerados. Também é possível a partir desta tela ir para o formulário de edição da análise.

Figura 20 - Tela de visualização de análise do StudentHistory



Fonte: Captura de tela feita pelo autor (2018)

A porção de tela que apresenta o diagrama Sankey possui a indicação das disciplinas, seus códigos, e ano e semestre das turmas analisadas. Para cada disciplina e semestre, uma coluna no diagrama Sankey é criada com a presença de cinco nós, ou seja, cinco grupos de alunos ingressantes conforme suas médias finais (maior que sete, entre cinco e sete, entre três e cinco, entre zero e três) naquela disciplina naquele semestre. Os alunos que não se matricularam naquela disciplina no referido semestre formam um grupo à parte.

Cada ligação entre as colunas indica que um grupo de alunos que obteve determinada média final numa disciplina à esquerda obteve determinada média final em outra disciplina à direita, ou não se matriculou nelas. A largura das linhas entre colunas indica a quantidade de alunos naquela situação. Deste modo, é possível visualizar o fluxo de alunos pelas disciplinas conforme seus desempenhos.

4 Avaliação do StudentHistory

Após a fase de desenvolvimento, avaliou-se o StudentHistory com a participação de potenciais usuários, considerando os seguintes objetivos:

- Verificar se os participantes na avaliação conseguem criar análises e o tempo necessário para a execução das tarefas propostas.
- Comparar a compreensão dos diagramas pelos participantes e pelos desenvolvedores do sistema.
- Consultar a opinião dos participantes sobre o uso da ferramenta no seu trabalho e sugestões de melhorias.

4.1 Metodologia

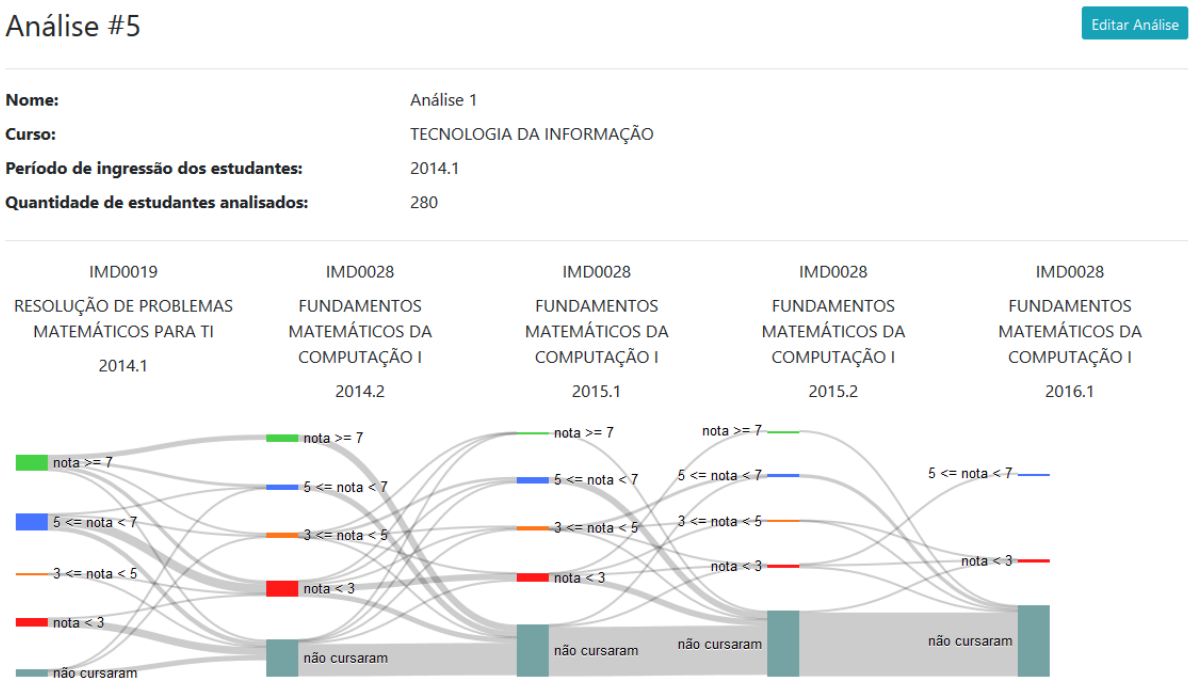
As metodologias utilizadas nesta avaliação foram Observação de Uso e Grupo Focal (Barbosa e Silva, 2010). Três pedagogos do Setor Pedagógico do IMD foram convidados e aceitaram participar desta pesquisa após esclarecimentos, conforme assinatura do termo de consentimento no Anexo A. Eles realizaram individualmente duas tarefas usando o StudentHistory. Na Tarefa 1, os participantes criaram uma análise contendo cinco disciplinas de matemática ou programação do curso de BTI da UFRN. Na Tarefa 2, eles interpretaram o gráfico de uma análise já existente no sistema (Figura 21) sobre a disciplina de Fundamentos Matemáticos para Computação. A execução destas tarefas foi observada e gravada em vídeo e os participantes escreveram observações sobre o desempenho dos alunos analisados, conforme sua compreensão, expectativas e sugestões de melhoria do sistema.

Após a realização das tarefas, os participantes realizaram uma discussão em grupo a respeito da sua opinião sobre o StudentHistory, o potencial da ferramenta na prática profissional e sugestões de melhoria. Essa discussão foi registrada em áudio. O roteiro de acompanhamento dessas atividades encontra-se no Anexo B.

Os vídeos de interação foram analisados para contabilizar o tempo de realização e o sucesso da conclusão das tarefas. As observações escritas pelos participantes foram analisadas para determinar compatibilidade com a intenção dos desenvolvedores. O áudio das discussões

do grupo focal foi analisado para identificar recorrências nas opiniões dos participantes, bem como outros aspectos relevantes.

Figura 21 - Tela da “Análise 1” apresentada aos participantes



Fonte: Captura de tela feita pelo autor (2018)

4.2 Resultados

Cada participante da avaliação do StudentHistory é referenciado por “P#”, sendo “#” um número identificador único.

4.2.1 Observação de Uso

Na Tarefa 1, P1 levou 8 minutos para criar a análise. Grande parte desse tempo foi gasto com a visualização das disciplinas oferecidas para análise. Apesar da conclusão parcial da tarefa de criar uma análise, P1 não ficou satisfeito com o gráfico resultante pois o conjunto de alunos analisados e o período das disciplinas escolhidas não geraram um gráfico com variações de notas suficientes para a observação do desempenho dos alunos. P1 editou a

análise quatro vezes e mudou ano e semestre de ingresso dos alunos, disciplinas, ano e semestre das disciplinas em busca de um gráfico com mais ligações entre os nós. P1 concluiu que a falta de conhecimento do currículo e oferta das disciplinas do curso dificultou na escolha de disciplinas que pudessem gerar um gráfico com um maior número de turmas cursadas.

O participante P2 demorou aproximadamente 8 minutos para realizar a criação da primeira análise. Ao longo da Tarefa 1, P2 observou uma análise já criada no sistema, tentou salvar a análise com campos obrigatórios em branco e os corrigiu após validação do sistema, e criou outras duas análises extras. Em uma das análises criadas por P2, foi selecionado o período 2017.2 para ingresso dos alunos e nenhum gráfico foi gerado, pois o curso de BTI da UFRN não possui alunos ingressantes para esse semestre. Em suas observações, P2 escreveu algumas sugestões como botões de navegação para voltar entre telas e o sistema realizar busca para o que foi ofertado no semestre.

P3 levou pouco menos de 7 minutos para concluir a criação da primeira análise durante a Tarefa 1. P3 editou quatro vezes a análise criada. Na primeira edição da análise, alterou o ano de ingresso dos estudantes. Na segunda e terceira edição da análise, P3 removeu uma das disciplinas selecionadas e inseriu outra disciplina. Na última edição feita, o participante editou o semestre para as turmas analisadas de uma das disciplinas. P3 relatou que o gráfico fica mais claro ao longo do uso e, assim como P1, constatou a necessidade do conhecimento da grade curricular do curso analisado. P3 sugeriu que o sistema alertasse os usuários caso escolhessem uma disciplina que ainda não foi cursada pelo grupo de alunos analisados. P3 teve dificuldades em compreender o *label* do *link* entre disciplinas que informa quantos alunos saem e chegam em nós “não cursaram”.

Na Tarefa 2, os participantes compreenderam o universo de estudantes ingressantes analisados e conseguiram identificar que uma coluna representava alunos agrupados por nota ou sem matrícula em determinada disciplina. Apesar disso, houve relato sobre a falta de alinhamento entre o gráfico e os elementos que informam as disciplinas presentes na análise.

P1 tomou nota sobre os componentes curriculares analisados e o desempenho de grupos de alunos em determinados semestres e considerou as disciplinas escolhidas nesta tarefa (Resolução de Problemas Matemáticos para TI e Fundamentos Matemáticos da Computação I) como críticas pelo alto número de reprovações e desistências. Relatou que entre uma disciplina e outra existiram comportamentos diferenciados. Em alguns casos alunos aumentaram as notas, enquanto em outros obtiveram rendimento inferior, inclusive nos casos

de repetência em um semestre seguinte para uma mesma disciplina. P2 levantou questionamentos se a espessura dos links está relacionada com a evolução do desempenho entre as disciplinas. Também menciona sobre rótulo informar a quantidade de alunos presentes em cada grupo presente no gráfico. P3 relata sobre a relação das notas entre as disciplinas analisadas e a taxa de aprovação e retenção de alunos em uma mesma disciplina.

4.2.2 Grupo Focal

Para o Grupo Focal realizado, três pontos principais guiaram as discussões, sendo eles a opinião dos participantes sobre o sistema; o potencial de uso da ferramenta; sugestões de melhoria.

P3 achou o sistema interessante por dar uma noção visual do que aconteceu com um grupo de alunos do curso, diferente do histórico individual padrão que apresenta toda a vida acadêmica de apenas um aluno por vez. No histórico em formato de tabela, ela relatou dificuldade em comparar o desempenho de um aluno com um conjunto de outros alunos do curso e em analisar sua evolução ao longo do tempo, principalmente quando se trata de reprovações. P1 falou sobre a possibilidade de analisar a evolução do desempenho dos alunos em um componente curricular ao longo do tempo, o índice de desistência de cursar a disciplina e o índice de sucesso para os casos de repetição de uma disciplina. P1 ficou fascinado com as possibilidades que o sistema lhe oferece. P2 comenta sobre o sistema possibilitar a análise do desempenho de alunos em diferentes disciplinas pré-requisito/co-requisito de outras disciplinas e permitir questionamentos do tipo “...na disciplina A ele (o aluno) foi bem-sucedido e na disciplina B não. Por quê?”. P2 também comenta sobre a possibilidade de avançar nos semestres para uma mesma disciplina a fim de uma análise com visão voltada a outros fatores além dos alunos, como por exemplo, troca de docentes que ministram as turmas da disciplina ou até mesmo tentar entender fatores externos que possam ter levado a uma alteração no desempenho dos alunos, como por exemplo, greves na universidade, protestos e paralisações na cidade.

Sobre o potencial de uso no trabalho dos pedagogos, P2 cita sobre a possibilidade de expansão da ferramenta para outros vínculos da UFRN (técnicos administrativos) e adaptações na utilização do sistema com foco na avaliação de cursos em outros níveis de ensino na universidade, como básico e técnico. Também foi levantamento o questionamento do uso do sistema para avaliar o desempenho acadêmico além de semestres concluídos, com o

detalhamento das unidades ao longo do semestre corrente, na tentativa de tentar identificar e corrigir problemas no início. P1 relata sobre a dificuldade em realizar análises de desempenho manualmente para uma grande quantidade de alunos pelo esforço e tempo necessários. Isso inviabiliza a realização de um acompanhamento contínuo dos alunos durante o andamento dos cursos, o que atrapalha a tomada de providências necessárias, como, por exemplo, encaminhamento para o plantão pedagógico, apoio de monitoria para reforço e esclarecimento de dúvidas. Os participantes consideraram que a visualização e compreensão do fluxo de alunos pelas disciplinas permite avaliar disciplinas com alto índice de reprovação, tentar entender os motivos das desistências (abandono de uma turma em determinado semestre) e o que acontece com os alunos a ponto de não haver uma melhora nas notas ao longo das reprovações em disciplinas.

A respeito das sugestões de melhoria, P1 diz que seria interessante indicação das disciplinas elegíveis conforme o ano e ingresso do grupo de alunos selecionados, a fim de evitar análises que gerem gráficos com poucos estudantes que cursaram as disciplinas. P2 aconselhou sobre a existência de legenda para sinalizar disciplinas que são pré-requisito de outras ou que não foram ofertadas no período escolhido. P1, P2 e P3 também sugerem a possibilidade de analisar as turmas das disciplinas durante o decorrer do semestre, com a separação das notas por unidades, com o objetivo de tentar antecipar reprovações e como ferramenta de autoavaliação dos docentes sobre suas turmas durante o semestre em execução. Também foi apontada a necessidade de analisar as turmas considerando os professores e a indicação de um grupo de alunos aprovados distinto do grupo de alunos que “não cursaram” a disciplina.

Em resumo, as principais sugestões de melhoria feitas pelos participantes foram:

- Expansão da ferramenta para outros vínculos técnico administrativos da UFRN.
- Tratamento do sistema para semestres sem alunos ingressantes.
- Alinhamento entre o diagrama e os rótulos que representam as disciplinas de uma análise.
- Botões de navegação entre algumas telas.
- Permitir avaliar o desempenho dos alunos para o semestre corrente utilizando as notas separadas por unidades.
- Sinalização de disciplinas pré-requisito de outras.

Apesar das dificuldades inicialmente demonstradas, os participantes obtiveram uma compreensão do sistema e dos gráficos próxima da esperada pelos desenvolvedores.

Identificaram os elementos gráficos dos diagramas, o universo de alunos ingressantes, as disciplinas analisadas e relacionaram as ligações entre os grupos de cada disciplina como parte do fluxo de desempenho desses alunos ao longo do curso. Para realização de análises interessantes, é importante ter um bom conhecimento acerca do currículo do curso, bem como ter acesso a um tutorial sobre o uso do gráfico Sankey.

5 Considerações Finais

Ao decorrer deste trabalho foi apresentada a ferramenta StudentHistory, uma aplicação web que busca auxiliar docentes da Universidade Federal do Rio Grande do Norte na análise do desempenho dos estudantes de graduação da universidade. O sistema apresenta aos docentes um diagrama do tipo Sankey com o fluxo de estudantes entre as disciplinas analisadas de acordo com suas respectivas notas.

O sistema também conta com a integração à API de Sistemas da Superintendência de Informática (SINFO) da UFRN, através da qual são recuperados dados que são utilizados nas análises, além de permitir a reutilização das credenciais de acesso já existentes no Sistema Integrado de Gestão de Atividades Acadêmicas (SIGAA). Trabalhos futuros podem ser realizados com a intenção de melhorias na ferramenta, implementação de novas funcionalidades, aplicação de diferentes tipos de visualização sobre os presentes dados analisados ou permitir análises sobre outros relacionados ao desempenho dos alunos.

Referências

BARBOSA, S.D.J.; SILVA, B.S. (2010) **Interação Humano-Computador**. Campus-Elsevier.

BLAHA, Michael; RUMBAUGH, James. (2006) **Modelagem e projetos baseados em objetos com UML 2**. 2. ed. rev. e atual. Rio de Janeiro RJ: Elsevier Campus.

BURBECK, Steve. (1987) **Applications Programming in Smaltalk-80 (TM): How to use Model-View-Controller.** Disponível em http://www.dgp.toronto.edu/~dwigdor/teaching/csc2524/2012_F/papers/mvc.pdf. Acesso em junho de 2018.

LARMAN, Craig. (2007) **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento interativo**. 3.ed. Porto Alegre: Bookman.

WARE, Colin. (2004) **Information Visualization: Perception for Design**. Morgan Kaufmann. 2nd Edition.

ANEXO A - Termo de Consentimento

Você foi convidado(a) para participar de uma pesquisa sobre uma ferramenta educacional, StudentHistory, sendo desenvolvida pelo aluno Jonathan Martins e o professor Bruno Santana, do IMD. O objetivo dessa pesquisa é investigar como o sistema pode apoiar a atividade de visualização do desempenho dos alunos de graduação da UFRN.

É importante destacar que **não** estamos interessados em avaliar seu conhecimento nessa área. Apenas **buscamos investigar como o sistema influencia sua atividade de acompanhamento do desempenho de alunos.** Gostaríamos de ouvir suas opiniões e comentários sobre o sistema, pois seu ponto de vista é muito importante para nós.

Por estas razões, solicitamos seu consentimento para observarmos a realização da visualização das análises feitas com apoio do sistema, bem como para realização de uma breve discussão em grupo sobre sua experiência. Vamos gravar o que for falado, gravar sua interação com o sistema e coletar suas respostas de identificação. Para tanto, é importante que você tenha algumas informações adicionais:

Os dados coletados destinam-se estritamente a atividades de pesquisa.

- Somente os pesquisadores envolvidos terão acesso aos dados brutos.
- A divulgação dos resultados desta pesquisa pauta-se no respeito à sua privacidade e ao seu anonimato em quaisquer documentos elaborados.
- O consentimento para participar desta pesquisa é uma escolha livre, feita mediante a prestação de todos os esclarecimentos necessários sobre a pesquisa.
- Você tem toda liberdade para interromper as atividades e desistir de participar da pesquisa. Neste caso, os pesquisadores se comprometem a descartar os dados coletados com sua contribuição.
- Você pode entrar em contato conosco pelo e-mail bruno@imd.ufrn.br e jonhmbc@gmail.com.

De posse das informações acima, gostaríamos que você se pronunciasse acerca da sua participação nesta pesquisa.

( ) Dou meu consentimento para sua realização.

( ) Não autorizo sua realização.

Natal, _____

Pesquisador: Jonathan Martins Barros Costa _____

Participante: _____ _____

ANEXO B - Roteiro de Observação e Grupo Focal

Ler o termo de consentimento e pedir assinatura.

Abrir o currículo de BTI no SIGAA.

Abrir o sistema no navegador http://10.7.16.4:3000/.

Configurar o programa de gravação.

Iniciar gravação.

Tarefa 1 - Criação de Análise:

1. Analise o desempenho de alunos de BTI em cinco disciplinas de programação ou matemática.
2. Escreva algumas observações sobre o desempenho dos alunos analisados.

Tarefa 2 - Visualização de Análise:

1. Consulte a “Análise 1” e escreva suas observações sobre o desempenho dos alunos analisados.

Parar a gravação e salvar.

Grupo Focal:

Iniciar gravação do áudio.

Questionamentos:

- O que você achou do sistema?
- Qual o potencial da ferramenta para apoiar o seu trabalho?
- Sugestões de melhorias.

Concluir a gravação do áudio.