UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS PATO BRANCO
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JOHNNY ROCKEMBACH KRAEMER

SISTEMA DE APOIO À DECISÃO PARA PREVENÇÃO DA EVASÃO NAS INSTITUIÇÕES DE ENSINO SUPERIOR

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2018

JOHNNY ROCKEMBACH KRAEMER

SISTEMA DE APOIO À DECISÃO PARA PREVENÇÃO DA EVASÃO NAS INSTITUIÇÕES DE ENSINO SUPERIOR

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Dr. Edilson Pontarolo

PATO BRANCO
2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

SISTEMA DE APOIO À DECISÃO PARA PREVENÇÃO DA EVASÃO NAS INSTITUIÇÕES DE ENSINO SUPERIOR

POR

JOHNNY ROCKEMBACH KRAEMER

Este trabalho de conclusão de curso foi apresentado no dia 20 de junho de 2018, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Universidade Tecnológica Federal do Paraná. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Banca examinadora:

Prof. Dr. Edilson Pontarolo
Orientador

Prof. Dr. Dalcimar Casanova

Prof. Dr. Pablo Gauterio Cavalcanti

Prof. Dr. Edilson Pontarolo
Coordenador do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Profª Drª Beatriz Terezinha Borsoi
Responsável pela Atividade de Trabalho de Conclusão de Curso

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

KRAEMER, Johnny Rockembach. **Sistema de apoio à decisão para prevenção da evasão nas instituições de ensino superior**. 2018. 92 f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2018.

O atual cenário das instituições de ensino superior no Brasil se caracteriza como um ambiente altamente competitivo, causado pelo crescimento do número de instituições, cursos disponíveis, facilidade de acesso, troca e reingresso de acadêmicos entre estes cursos e instituições. Ao mesmo tempo, verificam-se elevados índices de evasão, que em 2014 chegaram a 25,4% nos cursos presenciais, fenômeno este que gera prejuízos ao aluno, à instituição de ensino e a toda sociedade. Com o intuito de auxiliar a inferir os alunos com maior tendência a evadirem dos cursos, este trabalho apresenta o desenvolvimento de um sistema de estimativa da probabilidade de evasão, utilizando-se de algoritmos de classificação já existentes e um sistema web de *Business Intelligence*, tornando assim possível a tomada de decisão a partir de informações quantitativas mais precisas. Este sistema foi desenvolvido e testado sobre uma base de 10.371 registros de alunos de dez cursos de graduação e obteve uma margem de sucesso na previsão da tendência de evasão do aluno variando de 92,34% no mínimo e 99,32% no máximo, com uma média de 95,81% entre todos os cursos analisados sobre a base de testes. Entre os potenciais benefícios da aplicação da mineração de dados neste contexto podemos citar o auxílio na identificação dos estudantes com maior probabilidade de evadirem dos respectivos cursos, possibilitando assim uma melhor tomada de decisão no combate à evasão.

Palavras-chave: Evasão Escolar. Ensino Superior. Algoritmos de Classificação. *Business Intelligence*. Sistemas web.

ABSTRACT

KRAEMER, Johnny Rockembach. **Decision support system to prevent dropout in higher education institutions**. 2018. 92 f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2018.

The current scenario of higher education institutions in Brazil is characterized as a highly competitive environment, caused by the increase in the number of institutions, available programs, ease of access, exchange and re-entry of students among these programs and institutions. At the same time, there been high dropout rates, which in 2014 reached 25.4% concerning on-site undergraduate programs, a phenomenon that causes losses to students, educational institutions, and society. In order to help to infer the students who are more likely to drop out of the programs, this work presents the development of a system designed to estimate a student's probability of dropping out of a program. The system employs some of the well-known classification algorithms, integrated into a Business Intelligence web system, thus making possible a more accurate decision-making based on quantitative information. We tested the system on 10,371 student's records from 10 on-site undergraduate programs and obtained, on the experimental setup, a margin of success in predicting student's dropout tendency ranging from 92.34% minimum and 99.32% maximum, with an average of 95.81%. Among the potential benefits of the application of data mining in this context, we can cite the aid in the identification of the students who are more likely to drop out, thus enabling a better decision-making in the fight against dropout and its undesired consequences.

Keywords: Education dropout. Undergraduate programs. Classification algorithms. Business Intelligence. Web systems.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGO

LISTA DE SIGLAS

API	*Application Programming Interface* (Interface de Programação de Aplicativos)
CASE	*Computer-Aided Software Engineering* (Engenharia de Software Assistida por Computador)
EAD	Educação a Distância
IDE	*Integrated Development Environment* (Ambiente Integrado de Desenvolvimento)
IES	Instituição de Ensino Superior
INEP	Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira
JVM	Máquina Virtual Java
NUAPE	Núcleo de Acompanhamento Psicopedagógico e Assistência Estudantil
SAD	Sistema de Apoio à Decisão
UML	*Unified Modeling Language* (Linguagem de Modelagem Unificada)
UTFPR	Universidade Tecnológica Federal do Paraná
JSON	*JavaScript Object Notation* (Notação de Objeto JavaScript)
LWL	*Locally Weighted Learning* (Aprendizagem Localmente Ponderada)
AAPE	Alunos com Alta Probabilidade de Evasão

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, os objetivos e a justificativa da realização deste trabalho. O texto é finalizado com a apresentação dos capítulos subsequentes.

1.1 CONSIDERAÇÕES INICIAIS

O atual cenário da educação superior se caracteriza como um ambiente altamente competitivo, causado pelo crescimento do número de instituições de ensino, cursos disponíveis, facilidade de acesso, troca e reingresso de acadêmicos entre estes cursos e instituições (SILVA FILHO et al., 2007). Ao mesmo tempo, em 2014 segundo dados do INEP (2016), a taxa de evasão dos cursos presenciais no país atingiu o índice de 27,9% na rede privada e 18,3% na pública, já nos cursos EAD, no mesmo ano, o índice chegou a 32,5% na rede privada e 26,8% na pública, gerando assim grandes, caros e longos prejuízos ao aluno, à instituição de ensino e a toda sociedade (JUNGHANS, 2014). As instituições de ensino superior têm despendido esforços na tentativa de diminuir a quantidade de alunos evadidos, no entanto, tal esforço carece de informações precisas e uma quantidade considerável de conhecimento do sistema social no âmbito em que tal problema é gerado. Nesse sentido as ferramentas computacionais voltadas para predição de dados e análise matemática podem ser de grande auxílio, uma vez que tendem a descobrir correlações, padrões e comportamentos que não são facilmente percebidos pelo ser humano, especialmente quando se tratam de relações não lineares ou pouco evidentes, em domínios envolvendo grandes volumes de dados numéricos. Desta forma este trabalho visa a obtenção de um algoritmo capaz de identificar tendências deste fenômeno, através da utilização de algoritmos de classificação e desenvolver um sistema web de *Business Intelligence*, para realizar a coleta, organização e análise dos dados, criando assim um Sistema de Apoio à Decisão (SAD) capaz de auxiliar as coordenações de curso e demais gestores a identificar e agir (tomar decisões) com base em informações mais precisas no processo de combate à evasão.

Silva (2011, p.32), afirma que *Business Intelligence*:

> consiste na transformação metódica e consciente dos dados provenientes de quaisquer fontes de dados (estruturados e não estruturados) em novas formas de proporcionar informação e conhecimento dirigidos aos negócios e orientados aos resultados.

1.2 OBJETIVOS

A seguir são apresentados o objetivo geral e os objetivos específicos definidos para este trabalho.

1.2.1 Objetivo Geral

Desenvolver um sistema de estimativa da probabilidade de evasão, utilizando-se de algoritmos de classificação já existentes, integrando-os a um sistema web de *Business Intelligence* que servirá como apoio à tomada de decisão, destinado a coordenações de curso e demais gestores de uma IES (Instituição de Ensino Superior).

1.2.2 Objetivos Específicos

Por meio do sistema desenvolvido é possível:

Possibilitar a análise do cenário relativo à evasão de acadêmicos do curso e da instituição, através de algoritmos de mineração e de técnicas de *Business Intelligence*, provendo uma melhor compreensão dos dados por meio de *dashboards*[1] interativos.

Com base em informação gerada por meio de algoritmos de classificação e técnicas de mineração de dados, auxiliar a identificação dos acadêmicos com maior

[1] Apresentação visual das informações mais importantes e necessárias para alcançar um ou mais objetivos de negócio, consolidadas e ajustadas em uma tela para fácil acompanhamento do negócio.

probabilidade de evadirem dos respectivos cursos, possibilitando assim uma melhor tomada de decisão no combate da evasão.

Possibilitar a redução de perdas com alunos evadidos e evitar a má destinação de recursos financeiros.

1.3 JUSTIFICATIVA

Este trabalho baseia-se na percepção de que o atual cenário da educação superior brasileira se caracteriza como um ambiente altamente competitivo, sendo um dos motivos o crescimento do número de instituições de ensino, que se manteve em constante ascensão nos últimos 14 anos, com um índice de crescimento de acumulado de 101%. Em 2014 havia 2.368 IES no país, sendo 2.070 instituições privadas e 298 instituições públicas (INEP, 2016). Em 2016 foram ofertados 34.366 cursos de graduação em 2.407 IES, sendo 2.111 privadas e 296 públicas (INEP, 2018).

Da mesma maneira o número de matrículas em cursos presenciais nas IES públicas e privadas no Brasil aumentou de forma expressiva nas últimas três décadas. De 2000 a 2014 chegou a crescer 141%. Em 2014 havia cerca 6,5 milhões de matrículas, já no ano anterior, esse total era de 6,2 milhões de matrículas, representando um crescimento total de 5,3% entre 2013 e 2014.

No entanto, apesar do crescimento no número de instituições de ensino superior e de alunos matriculados, a evasão apresenta índices preocupantes. Os índices de evasão se caracterizam como extremamente alarmantes, segundo dados do INEP (2016), a taxa de evasão dos cursos presenciais no país atingiu o índice de 27,9% na rede privada e 18,3% na pública, já nos cursos EAD, no mesmo ano, o índice chegou a 32,5% na rede privada e 26,8% na pública. Os níveis gerais de abandono são em média ainda mais elevados nos cursos de formação de professores (licenciaturas). Em 2014, a evasão nos cursos de Pedagogia alcançou 39%, em Física 57,2%, em Química 52,3%, e em Matemática 52,6% de desistências (INEP, 2016).

Desta forma este trabalho visa o desenvolvimento de um SAD que possa auxiliar os coordenadores e demais gestores destas intuições a identificar e agir com base em informações mais precisas no processo de combate à evasão.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado em capítulos. Este é o primeiro e apresenta a introdução do trabalho. O Capítulo 2 contém o referencial teórico que está centrado no fenômeno da evasão e em sistemas de apoio à decisão. No Capítulo 3 estão descritos os materiais que foram utilizados para modelagem e implementação do sistema e o método que contém as principais atividades realizadas para a modelagem e a implementação do sistema. O Capítulo 4 apresenta o sistema e exemplos dos códigos desenvolvidos. Por fim, no Capítulo 5 são apresentadas as considerações finais.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico referente à elaboração deste trabalho. A Seção 2.1 aborda a evasão no ensino superior. A Seção 2.2 trata de trabalhos similares. A Seção 2.3 levanta informações sobre os sistemas de apoio à decisão. A Seção 2.4 aborda mineração de dados e a Seção 2.5 apresenta os trabalhos anteriores.

2.1 EVASÃO NO ENSINO SUPERIOR

Atualmente o ensino superior se encontra em uma fase de desaceleração da demanda, além de enfrentar diversos desafios, tais como: capacidade instalada x número de vagas x demanda; grande número de IES numa mesma região oferecendo os mesmos cursos; vagas ociosas; inadimplência e evasão (COBRA; BRAGA, 2004).

Da mesma forma um dos principais desafios de uma IES é compreender que os alunos não compram cursos, e sim, uma carreira profissional de sucesso. E que estas instituições necessitam agregar valor a seus cursos para que seus alunos tenham mais condições de disputar espaço no mercado de trabalho (COBRA; BRAGA, 2004).

Segundo Garcia (2006), o setor atingiu um ponto de estagnação, não existindo mais folga de alunos e a competição tornou-se ainda mais acirrada e que durante o período de crescimento acelerado, ocorreram algumas mudanças no setor: alunos mais exigentes e mais informados; mudança de hábitos, preferências e gostos; queda no conceito de fidelidade, tornando mais fácil mudar de IES, causando baixa fidelidade e consequente evasão; a curva de desenvolvimento atingiu o ponto de maturação e já começa a dar sinais de declínio; grande incremento de competição, com novos *players* no mercado e redução geral dos valores de mensalidades.

Devido à grande procura por vagas ofertadas que havia, as instituições de ensino pouco se preocuparam com questões futuras como retenção de aluno, sua prioridade era atender essa demanda inicial por vagas. Com o passar do tempo a

preocupação passou a ser como reter os alunos já conquistados e atrair novos candidatos para o processo seletivo (MARTINS, 2007).

Nesse sentido, Lopes (2006, p. 112) afirma:

> Muito se faz para conquistar novos alunos, mas muito pouco esforço tem sido feito no sentido de reter ou aumentar o nível de satisfação de seus atuais [....] A manutenção dos seus alunos é, cada vez mais, uma preocupação compartilhada. As taxas de evasão crescem na medida em que crescem as ofertas de novos cursos e novas instituições.

Desta forma a evasão pode ser considerada uma ameaça e, ao mesmo tempo, uma oportunidade no sentido de que, com a queda da demanda, as IES estão percebendo que a manutenção do aluno é tão importante quanto a sua captação (MARTINS, 2007).

2.2 TRABALHOS SIMILARES

Alguns estudos já foram pautados no contexto de tentar entender os motivos e buscar formas de prevenir a evasão, entre eles o realizado por Walter e Abbad (2008). Esse estudo objetivou analisar o relacionamento entre características da clientela, curso e comportamentos do aluno em relação a cursos a distância e o impacto destes em relação à evasão. Através da aplicação de questionais e análise de regressão logística, identificou-se que indivíduos que já haviam participado de cursos a distância anteriormente foram os que menos se evadiram dos cursos; indivíduos que responderam mais favoravelmente aos itens relativos aos fatores positivos intrínsecos e extrínsecos aos cursos foram os que menos evadiram e indivíduos que mais indicaram fatores desfavoráveis relacionados aos cursos foram os que mais se evadiram (WALTER; ABBAD, 2008).

Werlang (2014) realizou um estudo no curso de Ciências Contábeis da Universidade Federal do Rio Grande do Sul (UFRGS) com o objetivo de identificar os fatores extrínsecos e intrínsecos que motivam a escolha e a permanência dos discentes no curso. Através de questionários analisando características qualitativas e quantitativas, foi possível identificar que maioria dos discentes da amostra se considera com boa ou excelente motivação para permanecer e ter um bom

desempenho no curso. Os fatores mais relevantes no sentido da permanência são “possibilidade de ser aluno da UFRGS”, “ter bons professores”, “família” e “amigos”.

Barbosa (2007) propôs um estudo na Universidade Federal de Uberlândia (UFU), com o intuito de verificar a evolução do percentual de alunos evadidos nos anos anteriores e posteriores da adoção do Sistema de Seleção Unificada (SISU). Através de uma pesquisa documental e posterior teste estatístico binominal foi possível identificar que as áreas que apresentam os maiores percentuais de alunos evadidos são a de Ciências Sociais Aplicadas, Ciências Humanas e Ciências Exatas e da Terra. As áreas Ciências Agrárias, Ciências Biológicas, Engenharias, Ciências Exatas e da Terra e Ciências da Saúde apresentam aumento significativos nos percentuais de alunos evadidos após o SISU, enquanto as de Ciências Humanas, Linguística, Letras e Artes e Ciências Sociais Aplicadas apresentaram reduções significativas nesses percentuais.

Manhães (2011) com o intuito de identificar precocemente alunos em risco de evasão nos cursos de graduação, realizou um estudo meio da utilização de técnicas de mineração de dados, no qual obteve uma precisão de 80% ao identificar a situação final do aluno do curso, neste caso foram utilizadas as primeiras notas semestrais dos calouros da graduação da Universidade Federal do Rio de Janeiro.

Souza (2008) realizou um estudo com o objetivo de gerar conhecimento a partir da investigação da evolução dos alunos compostos em uma base de dados da graduação de engenharia. Através da utilização de técnicas de mineração de dados, foram criadas regras de classificação, através de árvores de decisão, e regras de associação. Os resultados obtidos mostraram uma série de disciplinas que conduzem à reprovação, indicando a necessidade de se rever o currículo dos cursos ou a abordagem das disciplinas (SOUZA, 2008).

2.3 SISTEMA DE APOIO À DECISÃO

O surgimento na década de 70 do conceito de SAD se deu em virtude de fatores como: competição entre as organizações, necessidade de informações rápidas para auxiliar no processo de tomada de decisão, disponibilidade de tecnologias de hardware e software que possibilitam buscar rapidamente as informações,

possibilidade de armazenar o conhecimento e as experiências de especialistas em bases de conhecimentos, necessidade de a informática apoiar o processo de planejamento estratégico empresarial (SPRAGUE; WATSON, 1991).

Segundo Sprague e Watson (1991), qualquer Sistema de Informação que forneça informações para auxílio à decisão é considerado um SAD. Sendo questionada por Falsarella e Chaves (1995), que defendem que SAD são sistemas que não só fornecem informações para apoio à tomada de decisão, mas também contribuem para essa atitude, uma vez que obtenção da informação é apenas parte do processo.

Sendo assim, os SAD, são sistemas interativos que fornecem suporte à tomada de decisões em ambientes complexos e dinâmicos, e que se adaptam conforme as necessidades do problema, utilizando ferramentas de análise e modelagem sofisticada (NUNES, 2007).

2.4 MINERAÇÃO DE DADOS

A mineração de dados é o processo de exploração e análise, por meio automático ou semiautomático, de grandes quantidades de dados, a fim de descobrir padrões e regras (BERRY; LINOFF, 1997). Consiste na habilidade de identificar, nos dados, os padrões válidos, novos, potencialmente úteis e compreensíveis, envolvendo métodos estatísticos, ferramentas de visualização e técnicas de inteligência artificial (FAYYAD et al., 1996).

O processo da mineração de dados pode ser dividido em cinco etapas, sendo elas: seleção dos dados; pré-processamento; formatação; mineração de dados; interpretação e avaliação, resultando assim em conhecimento.

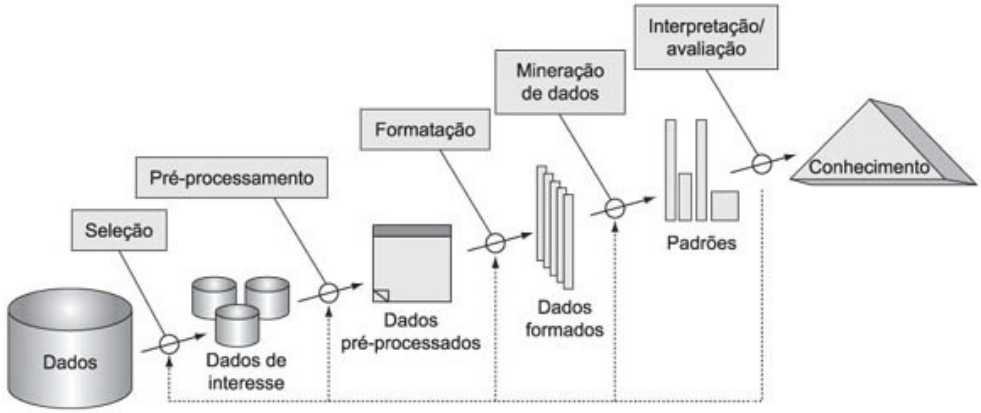

Figura 1 - Etapas do processo de mineração de dados
Fonte: Adaptado de Fayyad et al. (1996).

Este processo de descoberta de conhecimento deve apresentar algumas características, tais como: ser eficiente, genérico (aplicável a vários tipos de dados) e flexível (facilmente modificável) (STEINER et al., 2006). Este conhecimento pode ser empregado de diferentes maneiras, sendo algumas delas, como no gerenciamento de informações, tomada de decisão, controle de processos entre outras diversas aplicações.

Segundo Berson (1997) e Ramos (1999), a mineração de dados pode ser realizada de acordo com algumas tarefas, tais como: associação, agrupamento e classificação. A seguir será apresentado uma breve descrição sobre classificação assim como de alguns de seus algoritmos que são foco deste trabalho.

2.4.1 Classificação

A classificação é uma das tarefas mais comuns na mineração de dados, consiste em avaliar os dados processados, classificando-os de acordo com as suas características, aonde as classes são previamente definidas e os dados são separados em conjuntos de teste e treinamento. Nesta tarefa, o modelo analisa o conjunto de registros fornecidos, com cada registro já contendo a indicação à qual classe pertence, a fim de ‘aprender’ como classificar um novo registro (aprendizado supervisionado).

São exemplos de tarefas de classificação (GOEBEL; GRUENWALD, 1999): classificar pedidos de créditos como de baixo, médio e alto risco; esclarecer pedidos

de seguros fraudulentos; identificar se um aluno tem maiores chances de evadir ou não do curso, tema foco deste trabalho.

Neste trabalho foram utilizados 3 (três) algoritmos distintos para realizar a classificação dos dados, sendo eles o LWL, J48 e Naive Bayes. A seguir será apresentado uma breve explicação sobre cada um deles.

2.4.1.1 LWL

O LWL (*Locally Weighted Learning*) assim como os demais algoritmos da família *Lazy*, baseiam-se no paradigma computacional de *lazy learning* que se refere a qualquer processo de aprendizado de máquinas que priorize a fase de consulta, a generalização da requisição ocorre quando esta é enviada, em contraste ao *eager learning*, processo que prioriza a fase de treinamento, onde a generalização das requisições é previa (BROWNLEE, 2007; WEBB; SAMMUT, 2011).

As características de *lazy learning* fazem com que os algoritmos baseados em seus conceitos tenham bons desempenhos com grupos de testes menores ou dados que sofram atualizações constantes, que é o caso dos dados provenientes da web. Outra característica inerte a esses algoritmos é o menor tempo na fase de treinamento compensada pela demora na predição dos dados novos (BROWNLEE, 2007; WEBB; SAMMUT, 2011).

2.4.1.2 Algoritmo J48

O algoritmo J48 permite a criação de modelos de decisão em árvore. O modelo de árvore de decisão é construído pela análise dos dados de treino e o modelo utilizado para classificar dados ainda não classificados. O J48 gera árvores de decisão, em que cada nó da árvore avalia a existência ou significância de cada atributo individual. As árvores de decisão são construídas do topo para a base, através da escolha do atributo mais apropriado para cada situação (HAN; KAMBER, 2006). Uma vez escolhido o atributo, os dados de treino são divididos em subgrupos,

correspondendo aos diferentes valores dos atributos e o processo é repetido para cada subgrupo até que uma grande parte dos atributos em cada subgrupo pertençam a uma única classe. A indução por árvore de decisão é um algoritmo que habitualmente aprende um conjunto de regras com elevada acuidade (HAN; KAMBER, 2006).

2.4.1.3 Naive Bayes

O algoritmo Naive Bayes é um dos mais simples classificadores probabilísticos. O modelo construído por este algoritmo é um conjunto de probabilidades. As probabilidades são estimadas pela contagem da frequência de cada valor de característica para as instâncias dos dados de treino (MONTGOMERY; RUNGER, 2006). Dada uma nova instância, o classificador estima a probabilidade de essa instância pertencer a uma classe específica, baseada no produto das probabilidades condicionais individuas para os valores característicos da instância. O algoritmo é também denominado de Naive, uma vez que todos os atributos são independentes dado o valor da variável da classe (MONTGOMERY; RUNGER, 2006). Apesar deste pressuposto, o algoritmo apresenta um bom desempenho em muitos dos cenários de predição de classes.

2.4.2 Discretização

A discretização é um dos processos frequentemente realizados na etapa de pré-processamento dos dados (LIU et al, 2002). O seu objetivo é transformar atributos contínuos em atributos categóricos. Essa transformação é feita associando intervalos de valores contínuos à novos valores categóricos. Assim, os métodos de discretização reduzem e simplificam os dados, tornando o aprendizado mais rápido e os resultados mais compactos (GARCIA et al., 2013). Para este trabalho foram utilizados dois métodos de discretização, sendo eles *Discretize* e *NumericToNominal* ambos fornecidos pela API Weka. O método *Discretize* usa largura igual ou mesma

frequência armazenada para discretizar um intervalo de atributos numéricos, aonde o número de caixas é escolhido automaticamente, já o método *NumericToNominal* converte atributos numéricos em nominais simplesmente adicionando cada valor numérico distinto à uma lista de valores nominais.

2.4.3 Validação Cruzada

Validação cruzada é um método estatístico utilizado para se estimar as probabilidades de classificações incorretas, que consiste em dividir a base de dados em x partes (folds) (DUDA et al., 2001). Destas, x-1 partes são utilizadas para o treinamento e uma serve como base de testes. Este processo é repetido x vezes, de forma que cada parte seja usada uma vez como conjunto de testes. Ao final, é calculada a média dos resultados obtidos em cada etapa, obtendo-se assim uma estimativa da qualidade e precisão do classificador.

2.4.3.1 Leave-one-Out

O método leave-one-out é uma simplificação da validação cruzada. Onde o número de subconjuntos k é igual a N (o número de elementos do conjunto original), os N padrões são divididos em dois conjuntos, um com 1 elemento e o outro com N-1 elementos (DUDA et al., 2001). Este processo é repetido até que todas os elementos sejam utilizados uma vez como teste. Ao final é calculado a média dos erros de cada uma das iterações assim gerando a taxa de erro final.

2.5 TRABALHO ANTERIOR

O autor deste trabalho nos anos de 2014 e 2015, participou como bolsista[2] de um projeto do curso de Tecnologia em Análise e Desenvolvimento de Sistemas da UTFPR (Universidade Tecnológica Federal do Paraná) Câmpus Pato Branco, que teve por objetivo caracterizar a evasão do respectivo curso e elaborar um modelo probabilístico preditivo com base nos fatores de risco mais relevantes para o curso.

O desenvolvimento do projeto seguiu as cinco etapas do processo de mineração de dados apresentada na seção 2.3, que seriam: seleção dos dados; pré-processamento; formatação; mineração de dados; interpretação e avaliação.

1ª Etapa - Seleção dos dados: A seleção dos dados, deu-se através do sistema acadêmico da UTFPR e de planilhas contendo a relação de alunos comtemplados no sistema de bolsas permanência da UTFPR campus Pato Branco fornecidas pelo NUAPE (Núcleo de Acompanhamento Psicopedagógico e Assistência Estudantil).

A base de dados ficou então composta por 434 registro do curso de Tecnologia em Análise e Desenvolvimento de Sistemas da UTFPR campus Pato Branco, entre os anos de 2007 a 2014, contendo as seguintes variáveis: sexo, período, matriz curricular, ano da situação, semestre da situação, ano de ingresso, semestre de ingresso, cidade, escola, cota de ingresso, número de bolsas permanência, semestres cursados e situação.

1ª Etapa – Pré-processamento: O pré-processamento dos dados foi realizado com o intuito unir todos os dados obtidos através do sistema acadêmico e das planilhas fornecidas pelo NUAPE, formando assim uma única base de dados.

2ª Etapa – Formatação: A formatação dos dados, teve por objetivo transformar os dados qualitativos em dados numéricos, aonde se fosse capaz de gerar um vetor numérico que representasse cada um dos acadêmicos.

3ª Etapa – Mineração dos dados: O processo de mineração dos dados foi implementado utilizando da linguajem de programação Java e a API do Weka, para realizar a classificação dos dados. O algoritmo de classificação escolhido foi o Logistic, fornecido pela API do Weka.

---

[2] Edital 036/2014 – PROGRAD/UTFPR - Apoio a projetos de melhoria dos cursos de graduação e técnicos de nível médio da UTFPR.

4ª Etapa - Interpretação e avaliação: Na etapa de interpretação e avaliação, constatou-se que o sistema se mostrou satisfatório na base de dados estuda, constou-se também a necessidade de expandir a quantidade de dados analisados e o desenvolvimento de uma ferramenta de *Business Intelligence* para uma melhor apresentação dos dados gerados.

O resultado decorrente deste trabalho foi o desenvolvimento de um software de apoio à decisão, destinado a prever a probabilidade de evasão dos acadêmicos, onde o mesmo obteve uma margem de acerto global de 94,47% (410 acertos) nos 434 casos estudados.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados para a realização deste trabalho. Os materiais estão relacionados às tecnologias e ferramentas utilizadas e o método apresenta a sequência das principais atividades realizadas.

3.1 MATERIAIS

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

Ferramenta / Tecnologia	Versão	Finalidade
Astah Community	7.0	Modelagem do diagrama de casos de uso e diagrama de classes.
DB Designer	4.0	Modelagem entidade-relacionamento.
NetBeans IDE	8.1	Ambiente para desenvolvimento da aplicação.
Java	8.0	Linguagem de programação utilizada para desenvolver o servidor de mineração.
PHP	7.2	Linguagem de programação utilizada para desenvolver o servidor de apresentação dos dados.
Laravel	5.5	Framework PHP utilizado para desenvolver o servidor de apresentação de dados.
MySQL	8.0	Banco de dados para armazenar os dados gerados.
Weka	3.8	Ferramenta de mineração de dados que será utilizada para criação e teste dos algoritmos de classificação.
Xampp	7.2	Pacote de serviços utilizada para hospedar o banco de dados MySQL e o servidor Apache.
Javascript	1.8	Linguagem de programação interpretada, utilizada no desenvolvimento da interface do sistema web.
DataTables	1.1	Biblioteca Javascript utilizada para criação das tabelas do sistema web.

AmCharts	3.2	Biblioteca Javascript utilizada para criação dos gráficos do sistema web.

Quadro 1 - Lista de ferramentas e tecnologias
Fonte: Autoria própria.

3.1.1 Astah Community

De acordo com Brondani et al. (2015), Astah Community é uma ferramenta gratuita voltada para a modelagem de diagramas UML (Unified Modeling Language). Além do Astah Community, existem outras três versões: Astah UML, Astah Professional e Astah Share que disponibilizam outras funcionalidades além da modelagem UML, porém, sua licença é comercial.

A ferramenta Astah Community é conhecida por sua praticidade e simplicidade em elaborar diagramas, como por exemplo: diagramas de classe, caso de uso, sequência, atividade, comunicação, máquina de estado, componentes, implantação, estrutura de composição, objetos e pacote.

3.1.2 DB Designer

DB Designer é uma ferramenta CASE (Computer-Aided Software Engineering) desenvolvida pela empresa Fabulous Force Database Tools. Um editor visual utilizado para a criação, modelagem, desenvolvimento e manutenção de banco de dados, que está disponível sob a licença GNU General Public License (GLP). O software é bastante utilizado com o banco de dados MySQL, porém também suporta outros bancos de dados, por exemplo: Oracle, MS SQL Server, SQLite.

3.1.3 NetBeans IDE

NetBeans é um ambiente integrado de desenvolvimento (IDE) multiplataforma, uma ferramenta que auxilia programadores a escrever, compilar,

debugar e instalar aplicações. Foi arquitetada em forma de uma estrutura reutilizável para simplificar o desenvolvimento e aumentar a produtividade, pois reúne em uma única aplicação todas estas funcionalidades. (NETBEANS, 2011).

Esta ferramenta é completamente escrita em Java, porém suporta, outras linguagens de programação, tais elas como o C, C++, Ruby, PHP, XML e HTML. Como ele é multiplataforma, ele funciona em qualquer sistema operacional, desde que o mesmo possua suporte à Máquina Virtual Java (JVM).

3.1.4 Java

Java é uma linguagem de programação orientada a objetos que foi lançada pela Sun Microsystems (atualmente de propriedade da Oracle) em 1995. É uma linguagem orientada a objetos, ou Programação Orientada a Objetos (POO). É um tipo de paradigma de análise, para a programação de sistemas no qual todos os elementos inseridos são objetos (MASTERTECH; 2018). Foi uma das tentativas de trazer a programação para um nível mais semelhante ao cotidiano. Java é considera o mais simples pois não é preciso se preocupar tanto com detalhes de baixo nível, como memória, processamento, ponteiros e lixo, por exemplo, pois possui um gerenciamento automático que facilita a vida do desenvolvedor, mas consome mais processamento. Java não foi feito para criar sistemas pequenos, mas sim para aplicações de médio e grande porte e com um time maior de desenvolvedores, o principal foco é para sistemas que cresçam com o tempo, pois permite alterações mais fáceis e rápidas no sistema (MASTERTECH; 2018).

3.1.5 PHP

O PHP é uma linguagem de programação principalmente utilizada para a programação de aplicativos Web. Diferentemente de linguagens de programação como o Javascript, que possui uma função similar, o PHP roda do lado do servidor, e não na máquina do cliente. Isso significa que além de ser enviado HTML puro para o

cliente, o PHP ainda consegue interagir com as aplicações existentes no servidor, como o banco de dados, sem ser necessário expor o código-fonte para o cliente. Com isso, ele é ideal para desenvolver programas que necessitam lidar com informações confidenciais, já que senhas e outras informações de caráter pessoal e intransferível não serão mostradas.

3.1.6 Laravel

Laravel é um Framework PHP utilizado para o desenvolvimento web, que utiliza a arquitetura MVC e tem como principal característica ajudar a desenvolver aplicações seguras e performáticas de forma rápida, com código limpo e simples, já que ele incentiva o uso de boas práticas de programação e utiliza o padrão PSR-2 como guia para estilo de escrita do código (DEVMEDIA; 2018). Para a criação de interface gráfica, o Laravel utiliza uma Engine de *template* chamada *Blade*, que traz uma gama de ferramentas que ajudam a criar interfaces bonitas e funcionais de forma rápida e evitar a duplicação de código.

3.1.7 MySQL

O MySQL é um Sistema Gerenciador de Banco de Dados relacional de código aberto usado na maioria das aplicações gratuitas para gerir suas bases de dados, o MySQL é um dos componentes centrais da maioria das aplicações públicas da Internet. O sistema foi desenvolvido pela empresa sueca MySQL AB e publicado, originalmente, em maio de 1995 (TECTUDO, 2012). Após, a empresa foi comprada pela Sun Microsystems e, em janeiro de 2010, integrou a transação bilionária da compra da Sun pela Oracle Corporation. O MySQL é de Código Aberto (Open-Source), desenvolvido e distribuído sob as licenças GNU/GLP (General Public Licence, ou traduzindo, Licença Pública Geral), a qual determina o que se pode ou não fazer à ferramenta e demais recursos.

3.1.8 Weka

A suíte Weka (Waikato Enviroment for Knowledge Analysis) é formado por um conjunto de implementações de algoritmos de diversas técnicas de Mineração de Dados (UNIVERSITY OF WAIKATO, 2010). Este software foi implementado na linguagem Java, contendo uma GUI para interagir com arquivos de dados e produzir resultados visuais, além de uma API geral, sendo assim possível incorporá-lo, como qualquer outra biblioteca, a seus próprios aplicativos para realizar tarefas de mineração de dados.

3.1.9 Xampp

O XAMPP é formado por um pacote que incluí os principais servidores de código aberto existentes, incluindo FTP, banco de dados MySQL e Apache com suporte às linguagens PHP e Perl (DATATABLES, 2018). Através dele é possível rodar os famosos CMS tais como: WordPress e Drupal na máquina local (no seu próprio computador), agilizando o desenvolvimento.

XAMPP é uma compilação de softwares livres (comparável a uma distribuição Linux), é gratuito e é livre para ser copiado sob os termos da GNU General Public Licence. Mas é apenas a compilação do XAMPP que é publicada sob a licença GPL (DATATABLES, 2018).

3.1.10 DataTables

DataTables é um plug-in para a biblioteca Javascript do jQuery. É uma ferramenta altamente flexível, construída sobre os fundamentos do aprimoramento progressivo, que adiciona recursos avançados a qualquer tabela HTML. Essa ferramenta permite que o usuário final consiga facilmente extrair informações úteis da

tabela o mais rápido possível e, para isso, o DataTables incorporou recursos como ordenação, pesquisa e paginação.

3.1.11 AmCharts

AmCharts é uma biblioteca de recursos de gráficos, para sites e aplicativos, e também, compatível com todos os navegadores modernos (AMCHATS, 2018). Possui a capacidade de exportar gráficos dinamicamente para vários formatos, incluindo imagens estáticas, SVG, PDF, Excel e CSV (AMCHATS, 2018). Os dados de fontes externas podem ser carregados e configurados nos formatos JSON e CSV.

3.2 MÉTODO

O método utilizado para a realização do trabalho está baseado nas fases de análise, projeto, codificação e testes do modelo sequencial linear exposto em Pressman (2002). Essas fases foram utilizadas como base, mas sofreram inclusões e alterações para adaptar-se aos objetivos deste projeto. Ressalta-se que as atividades realizadas não foram e não serão estritamente sequenciais. As principais fases ou etapas definidas para o ciclo de vida do sistema são:

a) Aquisição da base de dados – Os dados utilizados neste trabalho foram fornecidos pela UTFPR, contemplando os câmpus e cursos apresentados no Quadro 2, totalizando 10.371 (dez mil trezentos e setenta e um) registros, com as seguintes variáveis: câmpus, sede, nível de ensino, grau, periodicidade, funcionamento, curso, ano, semestre, nome, código, ano de ingresso, categoria stricto sensu, coeficiente de rendimento, código do curso, código INEP do curso, data de colação de grau, data de ingresso, data de nascimento, disciplinas aprovadas, disciplinas consignadas, disciplinas matriculadas, disciplinas reprovadas por frequência, disciplinas reprovadas por nota, e-mail, forma de ingresso, gênero, idade, matriz curricular, mudou de curso - mesmo câmpus, mudou de curso - outro câmpus, município, município SISU, nota ENEM humanas, nota ENEM linguagem, nota ENEM matemática, nota ENEM

natureza, nota ENEM redação, nota final SISU, número de entradas em outros cursos, número de entradas no curso, ordem de ingresso no curso, país de nascimento, período, provável jubilamento, regime de ensino, retenção parcial, retenção total, semestre de ingresso no curso, situação, tipo de cota, total de períodos do curso, total de semestres cursados, UF e UF SISU. Vale ressaltar que os dados utilizados no processo de classificação foram os registros do último semestre disponível relativo à situação de cada aluno. Os quantitativos encontram-se sintetizados no Quadro 2.

b) Levantamento de requisitos – Para definir o problema foram utilizadas entrevistas com usuários e pesquisa em fontes bibliográficas. O levantamento dos requisitos iniciou com um documento elaborado pelo autor contendo os requisitos que ele considera necessários para o sistema, definindo a primeira ideia do que é objetivado com o sistema, enfatizando funcionalidades e aspectos de qualidade relevantes.

c) Análise – Para realizar a análise foram utilizados conceitos e a metodologia do paradigma da orientação a objetos. Em seguida foi elaborado o diagrama de casos de uso com a descrição formalizada dos requisitos.

d) Projeto - Diagramas de classes foram elaborados para a composição do projeto do sistema. Os componentes e as classes que compõem o sistema serão distribuídos em cada uma das camadas da arquitetura. Também serão definidos os recursos mais relevantes da linguagem para implementar aspectos importantes do sistema.

e) Implementação do sistema – Foram utilizados os recursos descritos na Seção 3.1. O sistema de previsão da probabilidade de evasão dos alunos foi desenvolvido na linguagem Java, utilizando como apoio a API de mineração de dados Weka.

O banco de dados para armazenar os dados gerados pelo algoritmo de previsão da probabilidade de evasão, que também disponibilizará os mesmos dados para o software web de *Business Intelligence* foi desenvolvido utilizando a ferramenta MySQL, com a linguagem SQL.

O software de *Business Intelligence* foi desenvolvido em uma plataforma web, utilizando do framework de desenvolvimento Laravel, que é baseado na linguagem de programação PHP, linguagem de estruturação HTML e de estilização CSS, além das bibliotecas javascript DataTables e AmCharts.

f) Realização dos testes - Na implementação foram realizados os testes relacionados ao código. Esses testes foram informais e realizados pelo programador visando identificar erros de codificação e o atendimento aos requisitos definidos para o sistema.

Curso	**Câmpus**	**Número de Alunos**
Campo Mourão	Engenharia Ambiental	867
Campo Mourão	Engenharia Civil	839
Campo Mourão	Curso Superior De Tecnologia Em Alimentos	811
Curitiba	Engenharia Elétrica	1351
Curitiba	Engenharia Mecânica	1381
Curitiba	Curso Superior De Tecnologia Em Design Gráfico	940
Pato Branco	Ciências Contábeis	1216
Pato Branco	Administração	1223
Pato Branco	Agronomia	1209
Pato Branco	Tecnologia Em Análise E Desenvolvimento De Sistemas	534

Quadro 2 - Lista de câmpus, cursos e número de alunos
Fonte: Autoria própria.

4 RESULTADOS

Este capítulo apresenta o sistema que foi desenvolvido como resultado deste trabalho. Inicialmente é apresentada a descrição do mesmo. Em seguida é apresentada a sua modelagem, telas e explicação do seu funcionamento assim como a implementação é exemplificada por meio da listagem de trechos do código-fonte.

4.1 ESCOPO DO SISTEMA

O sistema tem por objetivo oferecer informações mais precisas para análises das coordenações de curso e demais gestores, possibilitando a redução de perdas com alunos evadidos e evitar a má destinação de recursos financeiros.

O sistema deve permitir o cadastro, alteração e exclusão de usuários e seu acesso deve ser feito através de um site web, realizado através de uma autenticação de usuário por meio de um *login* e senha.

O sistema deve permitir a geração da probabilidade de evasão dos acadêmicos, através de vários algoritmos, escolhendo o melhor com base em seu sucesso em prever corretamente a probabilidade do mesmo evadir levando em conta sua situação atual, isso será realizado semestralmente, através de um processo automatizado.

Deve ser possível a análise da probabilidade de evasão de um determinado grupo (por exemplo: curso, campus e instituição), e de cada aluno individualmente. Permitindo também a emissão de relatórios gerenciais, com relações de alunos com maior e menor probabilidade de evadir, e cursos com maiores índices de probabilidade de evasão e entre outros.

4.2 MODELAGEM DO SISTEMA

A seguir são apresentados os requisitos e os diagramas desenvolvidos a fim de apresentar os processos e a estrutura da aplicação. Os requisitos estão dispostos como funcionais e não funcionais fazendo uso do identificador RF para os funcionais e RNF para os não funcionais.

RF1 - Cadastrar usuários do sistema.				**Oculto ( )**
Descrição: Cadastrar usuários do sistema. Um usuário possui os campos nome, e-mail, cargo, câmpus, curso e senha. Permitir exclusão do usuário. Editar os campos, nome, e-mail, cargo, campus, curso e senha. Consultar listagem de usuários.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF 1.1 – Acesso.	O sistema deve permitir que apenas usuários autenticados e com cargos de Desenvolvedor, Administrador da Instituição ou Administrador do Campus possam acessar esta funcionalidade.	Regra de Segurança.	( )	(X)
RNF 1.2 – E-mail único.	O sistema não deve permitir o cadastro de dois e-mails iguais.	Regra de negócio.	( )	(X)
RNF 1.3 – Confirmação de senha.	O sistema deve fazer uma confirmação através de dois campos para garantir que a senha foi digitada corretamente.	Regra de negócio.	( )	(X)
RNF 1.4 – Obrigatoriedade do campo curso.	O sistema deve obrigar o usuário informar um ou mais cursos se o usuário a ser cadastrado for um coordenador de curso ou professor.	Regra de negócio.	( )	(X)

Quadro 3 - Detalhamento do requisito cadastrar usuários do sistema
Fonte: Autoria própria.

RF2 - Cadastrar Situação dos Alunos.				**Oculto ( )**
Descrição: Cadastrar Situação dos Alunos. As situações dos alunos são criadas automaticamente no processo de carregamento de dados, mas através deste cadastro o Desenvolvedor pode alterar a situação resumida e a descrição da situação dos alunos.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF 2.1 – Acesso.	O sistema deve permitir que apenas usuários autenticados e com cargo de Desenvolvedor possam acessar esta funcionalidade.	Regra de Segurança.	( )	(X)
RNF 2.2 – Alteração da situação completa.	O sistema não deve permitir a alteração do campo situação completa.	Regra de integridade.	( )	(X)

Quadro 4 - Detalhamento do requisito cadastrar situação dos alunos
Fonte: Autoria própria.

RF3 - Carregar Dados.				**Oculto ( )**
Descrição: Carregamento de Dados. Através desta funcionalidade, o Desenvolvedor pode fazer o carregamento dos dados de uma planilha contida em um arquivo em formato ".xls" ou ".xlsx" de modo que cada registro desta planilha será inserido em sua respectiva tabela no banco de dados.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF 3.1 – Acesso.	O sistema deve permitir que apenas usuários autenticados e com cargo de Desenvolvedor possam acessar esta funcionalidade.	Regra de Segurança.	( )	(X)
RNF 3.2 – Verificar se o arquivo está vazio.	O sistema deve verificar se o arquivo está vazio, se o mesmo estiver deve informar ao usuário.	Regra de integridade.	( )	(X)
RNF 3.3 – Verificar se o arquivo está no padrão.	O sistema deve verificar se o arquivo está no padrão, se o mesmo não estiver deve informar ao usuário.	Regra de negócio.	( )	(X)
RNF 3.4 – Verificar duplicidade de registros.	O sistema deve verificar se o registro já existe no banco de dados, caso exista não deve realizar a inclusão novamente.	Regra de negócio.	( )	(X)

Quadro 5 - Detalhamento do requisito carregar dados
Fonte: Autoria própria.

RF4 - Classificar Dados.				**Oculto ( )**
Descrição: Classificar Dados. Através desta funcionalidade, o Desenvolvedor pode executar o processo de classificação que calcula as probabilidades de evasão de cada um dos alunos.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF 4.1 – Acesso.	O sistema deve permitir que apenas usuários autenticados e com cargo de Desenvolvedor possam acessar esta funcionalidade.	Regra de Segurança.	( )	(X)
RNF 4.2 – Classificar somente classificadores selecionados.	O sistema deve realizar a classificação somente dos classificadores selecionados na página de “Classificadores”.	Regra de negócio.	( )	(X)
RNF 4.3 – Classificar somente as variáveis selecionadas.	O sistema deve realizar a classificação somente das variáveis selecionadas na página de “Variáveis”.	Regra de negócio.	( )	(X)
RNF 4.4 – Classificar somente cursos selecionados.	O sistema deve realizar a classificação somente dos cursos selecionados na página de “Cursos”.	Regra de negócio.	( )	(X)

Quadro 6 - Detalhamento do requisito classificar dados
Fonte: Autoria própria.

RF5 - Editar Variáveis.				**Oculto ( )**
Descrição: Editar Variáveis. Permite visualizar as variáveis disponíveis e indicar se cada uma deve, ou não, ser utilizada no processo de classificação, ou passar pelo processo de discretização.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF 5.1 – Acesso.	O sistema deve permitir que apenas usuários autenticados e com cargo de Desenvolvedor possam acessar esta funcionalidade.	Regra de Segurança.	( )	(X)

Quadro 7 - Detalhamento do requisito editar variáveis
Fonte: Autoria própria.

RF6 - Editar Classificadores.				**Oculto ( )**
Descrição: Editar Classificadores. Visualizar os classificadores disponíveis e alterar se o mesmo deve ser utilizado no processo de classificação.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF 6.1 – Acesso.	O sistema deve permitir que apenas usuários autenticados e com cargo de Desenvolvedor possam acessar esta funcionalidade.	Regra de Segurança.	( )	(X)

Quadro 8 - Detalhamento do requisito editar classificadores
Fonte: Autoria própria.

RF7 - Editar Cursos.				**Oculto ( )**
Descrição: Editar Cursos. Visualizar os cursos disponíveis e alterar se o mesmo deve ser utilizado no processo de classificação.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF 7.1 – Acesso.	O sistema deve permitir que apenas usuários autenticados e com cargo de Desenvolvedor possam acessar esta funcionalidade.	Regra de Segurança.	( )	(X)

Quadro 9 - Detalhamento do requisito editar cursos
Fonte: Autoria própria.

RF8 - *Dashboard* Instituição.				**Oculto ( )**
Descrição: Visualizar e exportar gráficos e tabelas com informações da instituição tais como: alunos por situação por campus idade, gênero, quantidade de semestres cursados, período, cursos com maior índice de evasão, entre outras.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF 8.1 – Acesso.	O sistema deve permitir que apenas usuários autenticados e com cargo de Desenvolvedor ou Administrador da Instituição possam acessar esta funcionalidade.	Regra de Segurança.	( )	(X)

Quadro 10 - Detalhamento do requisito *dashboard* instituição
Fonte: Autoria própria.

RF9 - *Dashboard* Campus.				**Oculto ( )**
Descrição: Visualizar e exportar gráficos e tabelas com informações do campus tais como: alunos por situação por curso, idade, gênero, quantidade de semestres cursados, período, cursos com maior índice de evasão e entre outras.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF 9.1 – Acesso.	O sistema deve permitir que apenas usuários autenticados e com cargo de Desenvolvedor, Administrador da Instituição ou Administrador do Câmpus possam acessar esta funcionalidade.	Regra de Segurança.	( )	(X)

Quadro 11 - Detalhamento do requisito *dashboard* campus
Fonte: Autoria própria.

RF10 - *Dashboard* Curso.				**Oculto ( )**
Descrição: Visualizar e exportar gráficos e tabelas com informações do curso tais como: alunos por situação, idade, gênero, quantidade de semestres cursados, período, alunos com maior índice de evasão e entre outras.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF 10.1 – Acesso.	O sistema deve permitir que apenas usuários autenticados possam acessar esta funcionalidade.	Regra de Segurança.	( )	(X)

Quadro 12 - Detalhamento do requisito *dashboard* curso
Fonte: Autoria própria.

RF11 - *Dashboard* Aluno.				**Oculto ( )**
Descrição: Visualizar e exportar dados do aluno, tais como: data de nascimento, idade de ingresso, gênero, cidade, semestre e ano de ingresso, forma de ingresso, tipo de cota de ingresso e entre outras. Além das informações semestrais, como: coeficiente de rendimento, disciplinas aprovadas, consignadas, matriculas, reprovadas e entre outras. Fornece também qual a probabilidade de o mesmo evadir do curso.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF 11.1 – Acesso.	O sistema deve permitir que apenas usuários autenticados possam acessar esta funcionalidade.	Regra de Segurança.	( )	(X)

Quadro 13 - Detalhamento do requisito *dashboard* aluno
Fonte: Autoria própria.

O diagrama de casos de uso apresentado na Figura 2, exibe as principais funcionalidades do sistema e seus cinco atores: Desenvolvedor, Administrador da Instituição, Administrador do Campus, Coordenador de Curso e Professor.

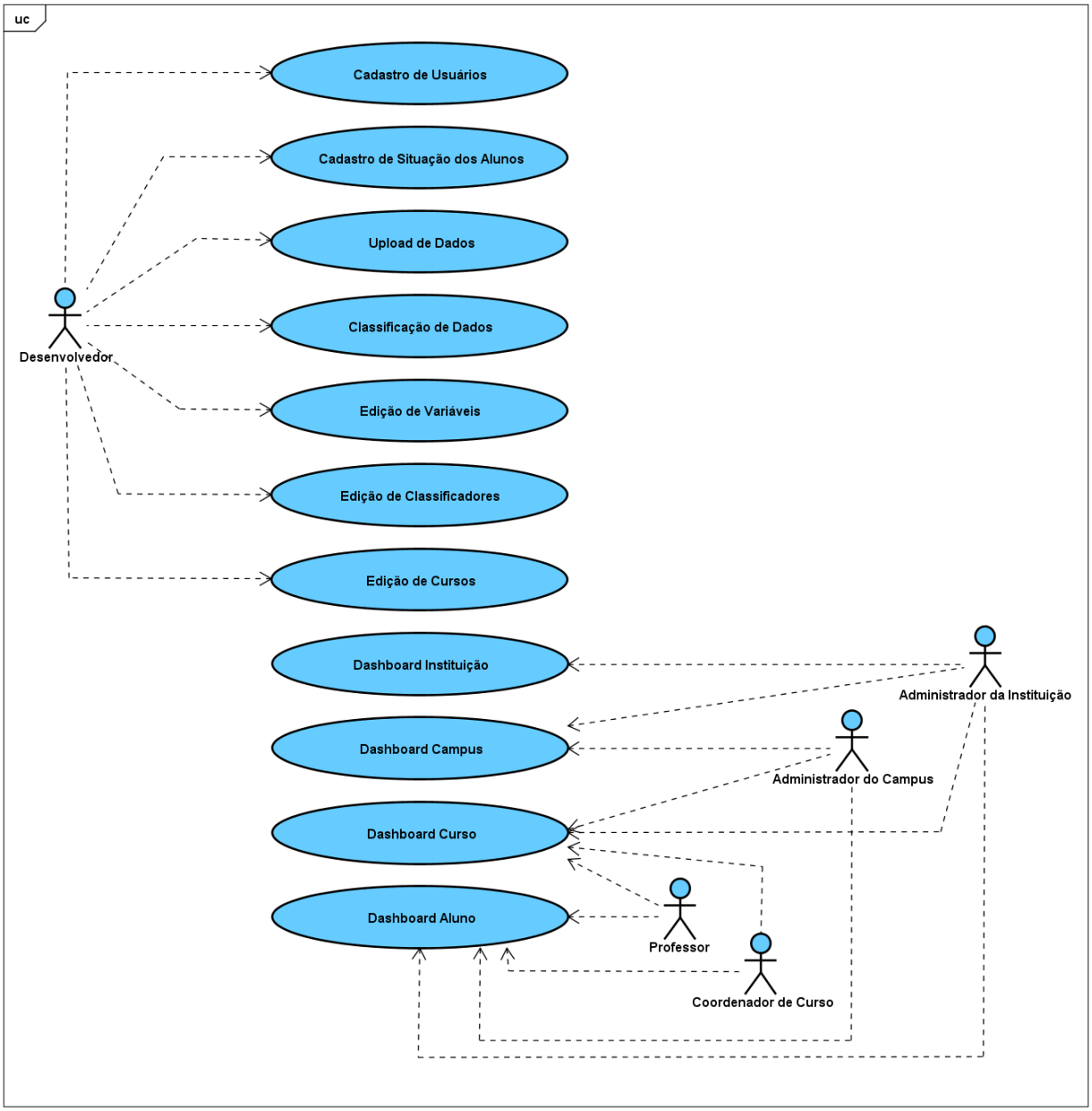

Figura 2 - Diagrama de casos de uso
Fonte: Autoria própria.

O diagrama de classes descreve a visão estática do sistema em termos de classes e relacionamento entre elas. A Figura 3 representa o diagrama de classes do sistema com seus respectivos relacionamentos.

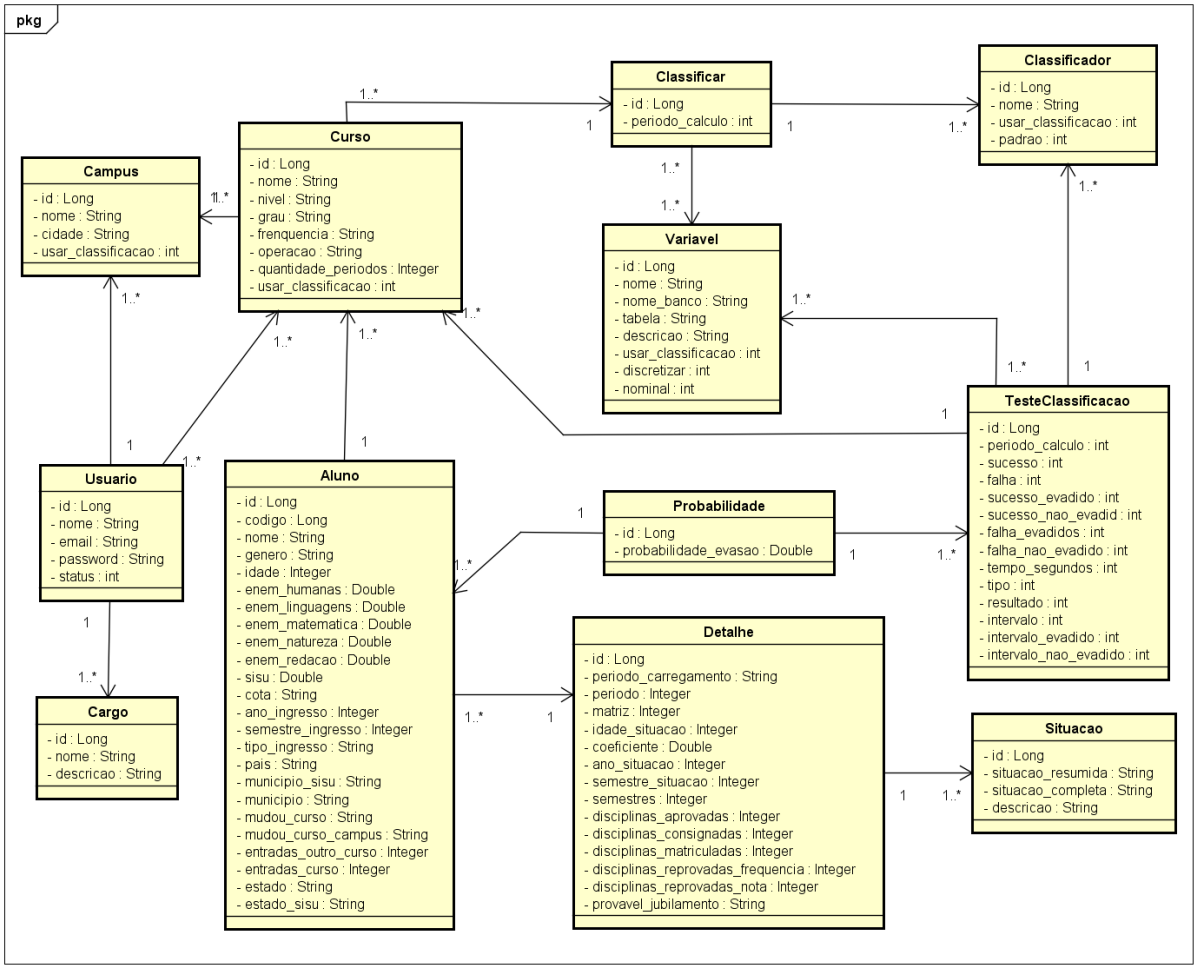

Figura 3 - Diagrama de classes
Fonte: Autoria própria.

O banco de dados é composto das tabelas *campus*, *classificador*, *classificar*, *classificar_variavel*, *curso*, *curso_classificar*, *detalhe*, *cargo*, *probabilidade*, *situacao*, *aluno*, *teste_classificacao*, *teste_classificacao_variavel*, *usuario*, *usuario_curso* e *variavel*. A Figura 4 apresenta o Diagrama Entidade-Relacionamento definido para o banco de dados.

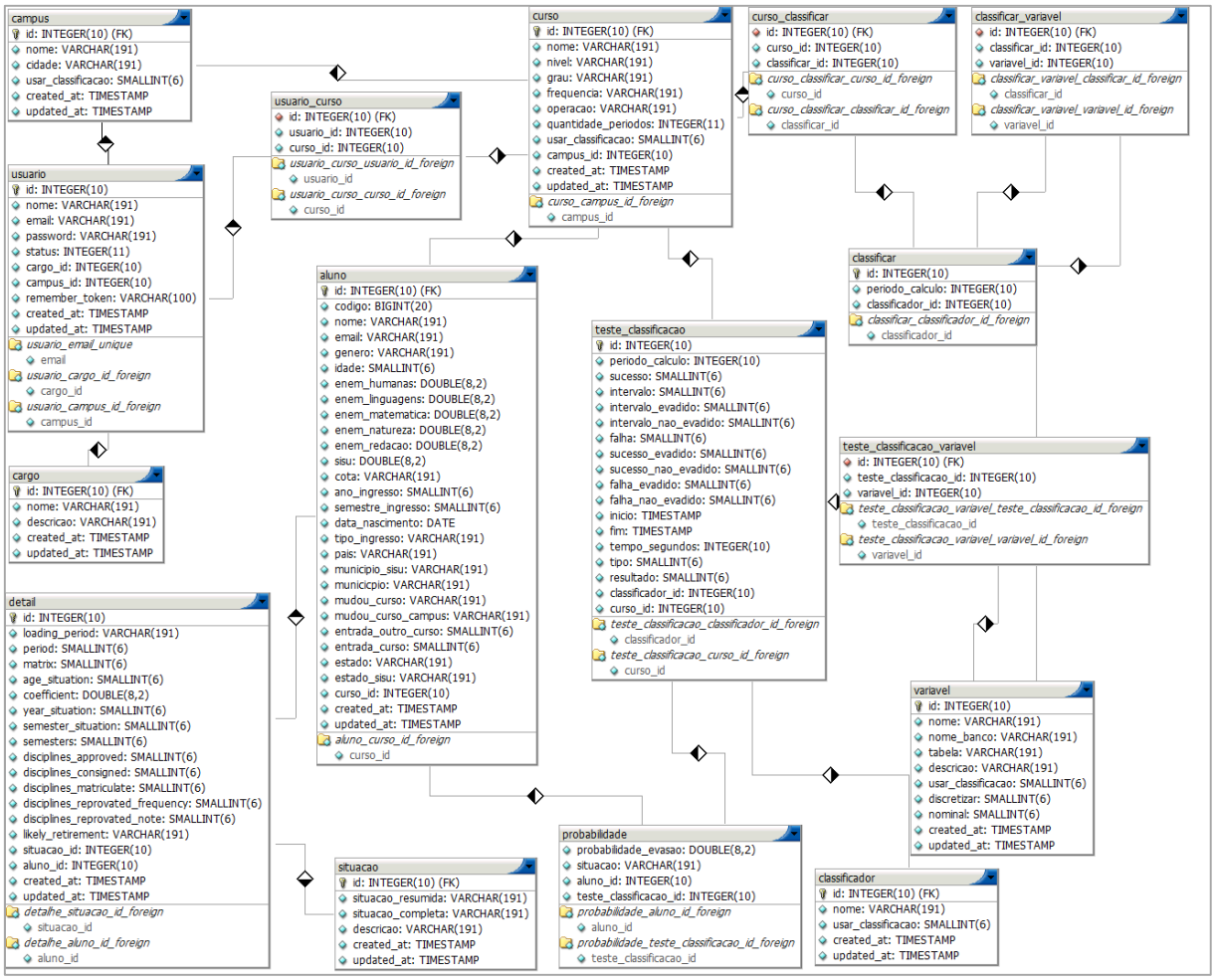

Figura 4 - Diagrama Entidade-Relacionamento
Fonte: Autoria própria.

A seguir será apresentado uma breve explicação de cada uma das tabelas que fazem parte do banco de dados representado na Figura 4.

O Quadro 14 apresenta os campos da tabela *campus*. Nesta tabela são armazenadas as informações dos câmpus da IES.

Coluna	Tipo	Nulo	Predefinido	Ligações para
id *(Primária)*	int(10)	Não		
nome	varchar(191)	Não		
cidade	varchar(191)	Sim	*NULL*	

Quadro 14 - Campos da tabela *campus*
Fonte: Autoria própria.

O Quadro 15 apresenta os campos da tabela *curso*. Nesta tabela são armazenadas as informações referente aos cursos, assim como se o mesmo deve ser utilizado no processo de classificação através do campo *usar_classificacao*.

Coluna	Tipo	Nulo	Predefinido	Ligações para
id *(Primária)*	int(10)	Não		
nome	varchar(191)	Não		
nivel	varchar(191)	Sim	*NULL*	
grau	varchar(191)	Sim	*NULL*	
frenquencia	varchar(191)	Sim	*NULL*	
operacao	varchar(191)	Sim	*NULL*	
quantidade_periodos	int(11)	Sim	*NULL*	
usar_classificacao	smallint(6)	Não	*0*	
campus_id	int(10)	Não		campus -> id

Quadro 15 - Campos da tabela *curso*
Fonte: Autoria própria.

O Quadro 16 apresenta os campos da tabela *usuario*. Nesta tabela são armazenadas as informações dos usuários, assim como seu cargo e campus, pelos respectivos campos *cargo_id* e *campus_id*.

Coluna	Tipo	Nulo	Predefinido	Ligações para
id *(Primária)*	int(10)	Não		
nome	varchar(191)	Não		
email	varchar(191)	Não		
password	varchar(191)	Não		
status	int(11)	Não		
cargo_id	int(10)	Não		cargo -> id
campus_id	int(10)	Sim	*NULL*	campus -> id

Quadro 16 - Campos da tabela *usuario*
Fonte: Autoria própria.

O Quadro 17 apresenta os campos da tabela *usuario_curso*. Nesta tabela são armazenadas as informações de quais cursos cada usuário tem acesso. Sendo assim, cada usuário pode ter acesso a 1 (um) ou mais cursos.

Coluna	Tipo	Nulo	Predefinido	Ligações para
usuario_id	int(10)	Não		usuario -> id
curso_id	int(10)	Não		curso -> id

Quadro 17 - Campos da tabela *usuario_curso*
Fonte: Autoria própria.

O Quadro 18 apresenta os campos da tabela *cargo*. Nesta tabela são armazenadas as informações referentes aos cargos dos usuários. Os cargos servem para definir os níveis de acesso dos usuários no sistema.

Coluna	Tipo	Nulo	Predefinido	Ligações para
id *(Primária)*	int(10)	Não		
nome	varchar(191)	Não		
descricao	varchar(191)	Sim	*NULL*	

Quadro 18 - Campos da tabela *cargo*
Fonte: Autoria própria.

O Quadro 19 apresenta os campos da tabela *classificador*. Nesta tabela são armazenadas as informações referentes aos classificadores, assim como se o mesmo deve ser utilizado no processo de classificação através do campo *usar_classificacao*.

Coluna	Tipo	Nulo	Predefinido	Ligações para
id *(Primária)*	int(10)	Não		
nome	varchar(191)	Não		
usar_classificacao	smallint(6)	Não	*0*	

Quadro 19 - Campos da tabela *classificador*
Fonte: Autoria própria.

O Quadro 20 apresenta os campos da tabela *variavel*. Nesta tabela são armazenadas as informações referentes as variáveis disponíveis, assim como se a mesma deve ser utilizada no processo de classificação através do campo *usar_classificacao*, também se deve ser realizado o processo de discretização, através dos campos *discretizar* e *nominal*.

Coluna	Tipo	Nulo	Predefinido	Ligações para
id *(Primária)*	int(10)	Não		
nome	varchar(191)	Sim	*NULL*	
nome_banco	varchar(191)	Sim	*NULL*	
tabela	varchar(191)	Sim	*NULL*	
descricao	varchar(191)	Sim	*NULL*	
usar_classificacao	smallint(6)	Não	0	
discretizar	smallint(6)	Não	0	
nominal	smallint(6)	Não	0	

Quadro 20 - Campos da tabela *variavel*
Fonte: Autoria própria.

O Quadro 21 apresenta qual *classificar* deve ser utilizado no processo de classificação de cada curso. O Quadro 22 apresenta os campos da tabela *classificar* em que são armazenadas as informações referentes ao classificador que deve ser utilizado no processo de classificação, e o Quadro 23 armazena quais *variaveis* devem ser utilizadas com os *classificadores* para cada curso, no processo de classificação.

Coluna	Tipo	Nulo	Predefinido	Ligações para
curso_id	int(10)	Não		curso -> id
classificar_id	int(10)	Não		classificar -> id

Quadro 21 - Campos da tabela *curso_classificar*
Fonte: Autoria própria.

Coluna	Tipo	Nulo	Predefinido	Ligações para
id *(Primária)*	int(10)	Não		
periodo_calculo	int(10)	Não		
classificador_id	int(10)	Não		classificador -> id

Quadro 22 - Campos da tabela *classificar*
Fonte: Autoria própria.

Coluna	Tipo	Nulo	Predefinido	Ligações para
classificar_id	int(10)	Não		classificar -> id
variavel_id	int(10)	Não		variavel -> id

Quadro 23 - Campos da tabela *classificar_variavel*
Fonte: Autoria própria.

O Quadro 24 apresenta os campos da tabela *aluno*. Nesta tabela são armazenadas as informações dos alunos. Destacasse que estas informações geralmente não costumam sofrer alterações com grande frequência. Já as informações que se alterar semestralmente foram separadas na tabela *detalhes* que será apresentado no Quadro 25.

Coluna	Tipo	Nulo	Predefinido	Ligações para
id *(Primária)*	int(10)	Não		
codigo	bigint(20)	Não		
nome	varchar(191)	Não		
email	varchar(191)	Sim	NULL	
genero	varchar(191)	Não		
idade	smallint(6)	Sim	NULL	
enem_humanas	double(8,2)	Sim	*NULL*	
enem_linguagens	double(8,2)	Sim	NULL	
enem_matematica	double(8,2)	Sim	*NULL*	
enem_natureza	double(8,2)	Sim	NULL	
enem_redacao	double(8,2)	Sim	*NULL*	
sisu	double(8,2)	Sim	NULL	
cota	varchar(191)	Sim	*NULL*	
ano_ingresso	smallint(6)	Não		
semestre_ingresso	smallint(6)	Não		
data_nascimento	date	Sim	NULL	
tipo_ingresso	varchar(191)	Sim	*NULL*	
pais	varchar(191)	Sim	NULL	
municio_sisu	varchar(191)	Sim	*NULL*	
municipio	varchar(191)	Sim	NULL	
mudou_curso	varchar(191)	Sim	*NULL*	
mudou_curso_outro_campus	varchar(191)	Sim	NULL	
entradas_outro_curso	smallint(6)	Sim	*NULL*	
entradas_curso	smallint(6)	Sim	NULL	
estado	varchar(191)	Sim	*NULL*	
estado_sisu	varchar(191)	Sim	NULL	
curso_id	int(10)	Não		*curso -> id*

Quadro 24 - Campos da tabela *aluno*
Fonte: Autoria própria.

O Quadro 25 apresenta os campos da tabela *detalhe*. Nesta tabela são armazenadas as informações dos alunos que sofrem alterações com grande

frequência, sendo assim existe apenas 1 (um) registro de *aluno* e 1 (um) ou mais registros de *detalhes* para cada aluno.

Coluna	Tipo	Nulo	Predefinido	Ligações para
id *(Primária)*	int(10)	Não		
periodo_carregamento	varchar(191)	Sim	NULL	
periodo	smallint(6)	Sim	*NULL*	
matriz	smallint(6)	Sim	NULL	
ano_situacao	smallint(6)	Sim	*NULL*	
coeficiente	double(8,2)	Sim	NULL	
idade_situacao	smallint(6)	Sim	*NULL*	
semestre_situacao	smallint(6)	Sim	NULL	
semestres	smallint(6)	Sim	*NULL*	
disciplinas_aprovadas	smallint(6)	Sim	NULL	
disciplinas_consignadas	smallint(6)	Sim	*NULL*	
disciplinas_matriculadas	smallint(6)	Sim	NULL	
disciplinas_reprovadas_frequencia	smallint(6)	Sim	*NULL*	
disciplinas_reprovadas_nota	smallint(6)	Sim	NULL	
provavel_jubilamento	varchar(191)	Sim	*NULL*	
situacao_id	int(10)	Não		situacao -> id
aluno_id	int(10)	Não		aluno -> *id*

Quadro 25 - Campos da tabela *detalhe*
Fonte: Autoria própria.

O Quadro 26 apresenta os campos da tabela *situacao*. Nesta tabela são armazenadas quais as possíveis situações de alunos, assim como a situação completa e resumida pelos respectivos campos *situacao_completa* e *situacao_resumida*.

Coluna	Tipo	Nulo	Predefinido	Ligações para
id *(Primária)*	int(10)	Não		
situacao_completa	varchar(191)	Sim	NULL	
situacao_resumida	varchar(191)	Não		
descricao	varchar(191)	Sim	NULL	

Quadro 26 - Campos da tabela *situacao*
Fonte: Autoria própria.

O Quadro 27 apresenta os campos da tabela *teste_classificacao*. Nesta tabela são armazenadas as informações referentes aos testes realizados na classificação, tais como qual classificador utilizado (*classificador_id*), qual curso analisado (*curso_id*), tempo em segundos (*tempo_segundos*) e quais foram os resultados obtidos: sucesso geral (*sucesso*), falha geral (*falha*) e entre outros.

Coluna	Tipo	Nulo	Predefinido	Ligações para
id *(Primária)*	int(10)	Não		
periodo_calculo	int(10)	Não		
sucesso	smallint(6)	Não	0	
intervalo	smallint(6)	Não	0	
intervalo_evadido	smallint(6)	Não	0	
intervalo_nao_evadido	smallint(6)	Não	0	
falha	smallint(6)	Não	0	
sucesso_evadido	smallint(6)	Não	0	
sucesso_nao_evadido	smallint(6)	Não	0	
falha_evadido	smallint(6)	Não	0	
falha_nao_evadido	smallint(6)	Não	0	
inicio	timestamp	Sim	*NULL*	
fim	timestamp	Sim	*NULL*	
tempo_segundos	int(10)	Sim	*NULL*	
type	smallint(6)	Não		
result	smallint(6)	Não	0	
classificador_id	int(10)	Não		classificador -> id
curso_id	int(10)	Não		curso -> id

Quadro 27 - Campos da tabela *teste_classificacao*
Fonte: Autoria própria.

O Quadro 28 apresenta os campos da tabela *teste_classificacao_variavel*. Nesta tabela são armazenadas quais variáveis foram utilizadas no processo de classificação.

Coluna	Tipo	Nulo	Predefinido	Ligações para
teste_classificacao_id	int(10)	Não		teste_classificacao -> id
variavel_id	int(10)	Não		variavel -> id

Quadro 28 - Campos da tabela *teste_classificacao_variavel*
Fonte: Autoria própria.

O Quadro 29 apresenta os campos da tabela *probabilidade*. Nesta tabela são armazenadas quais as probabilidades de evasão de cada um dos alunos e qual o teste que gerou este resultado, através do campo *teste_classificacao_id*.

Coluna	Tipo	Nulo	Predefinido	Ligações para
id *(Primária)*	int(10)	Não		
probabilidade_evasao	double(8,2)	Não		
situacao	varchar(191)	Sim	*NULL*	
aluno_id	int(10)	Não		aluno -> id
teste_classificacao_id	int(10)	Não		teste_classificacao -> id

Quadro 29 - Campos da tabela *probabilidade*
Fonte: Autoria própria.

4.3 APRESENTAÇÃO DO SISTEMA

O leiaute base (Figura 5) é composto pela logo do sistema na parte superior esquerda e na direita o menu usuário, onde este poderá realizar o *logout* e acessar seu perfil, e pelo menu de acesso as demais páginas que se encontra na lateral esquerda, sendo este dividido em três seções: *Dashboards*, Administrativo e Desenvolvedores.

Figura 5 - Leiaute base do sistema
Fonte: Autoria própria.

Após o usuário realizar o *login* no sistema, o mesmo é redirecionado para a página principal. A seguir serão apresentados os diferentes cenários para cada um dos tipos de usuários e detalhado cada uma das páginas do sistema.

Se o usuário for um Administrador da Instituição o mesmo será redirecionado para o *dashboard* da Instituição, como é demostrado no Figura 6, esta página é composta por gráficos, indicadores e tabelas que demostram o atual cenário da instituição como um todo. Somente os usuários Administrador da Instituição e Desenvolvedor tem acesso a esta página.

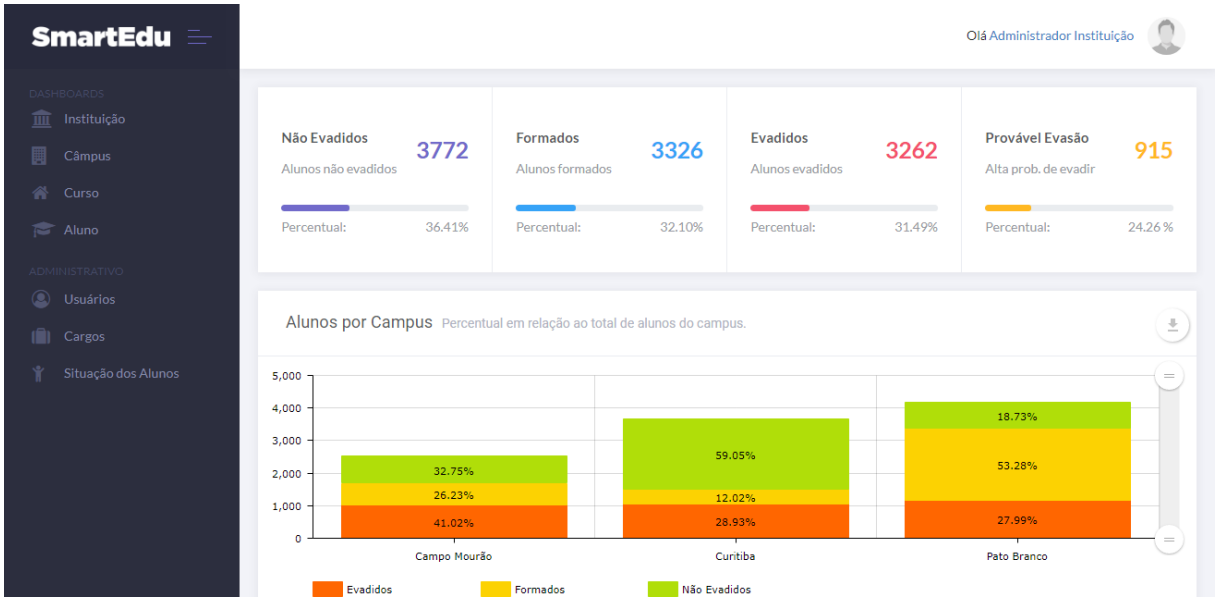

Figura 6 - *Dashboard* Instituição (Usuário: Administrador da Instituição)
Fonte: Autoria própria.

A Figura 7 apresenta os indicadores gerais da instituição, como a soma de alunos não evadidos, formados e evadidos, e de seu percentual em relação a todos os alunos da instituição, apresenta também o total de alunos com alta probabilidade de evasão (mais de 60,00% de probabilidade de evasão) e seu percentual em relação aos alunos não evadidos.

Figura 7 - Indicadores gerais da instituição (*Dashboard* Instituição)
Fonte: Autoria própria.

A Figura 8 apresenta o gráfico de “Alunos por Campus” que fornece o percentual de alunos por situação resumida (Evadidos, Formados e Não Evadidos) e uma escala da quantidade de alunos por campus.

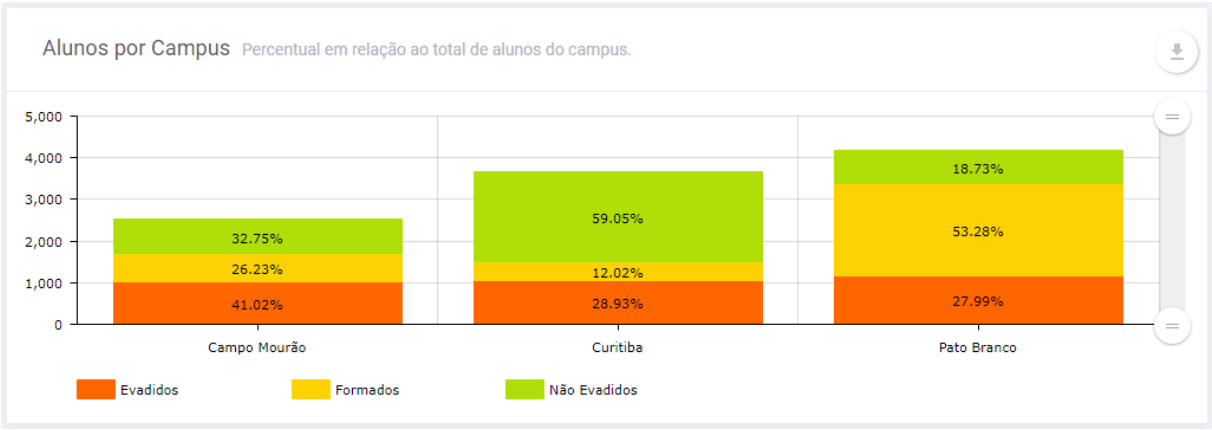

Figura 8 - Gráfico de alunos por campus e situação resumida (*Dashboard* Instituição)
Fonte: Autoria própria.

A Figura 9 apresenta o gráfico de "Alunos por Gênero" que fornece o percentual de alunos por gênero (Feminino e Masculino) e situação resumida (Evadidos, Formados e Não Evadidos) da instituição.

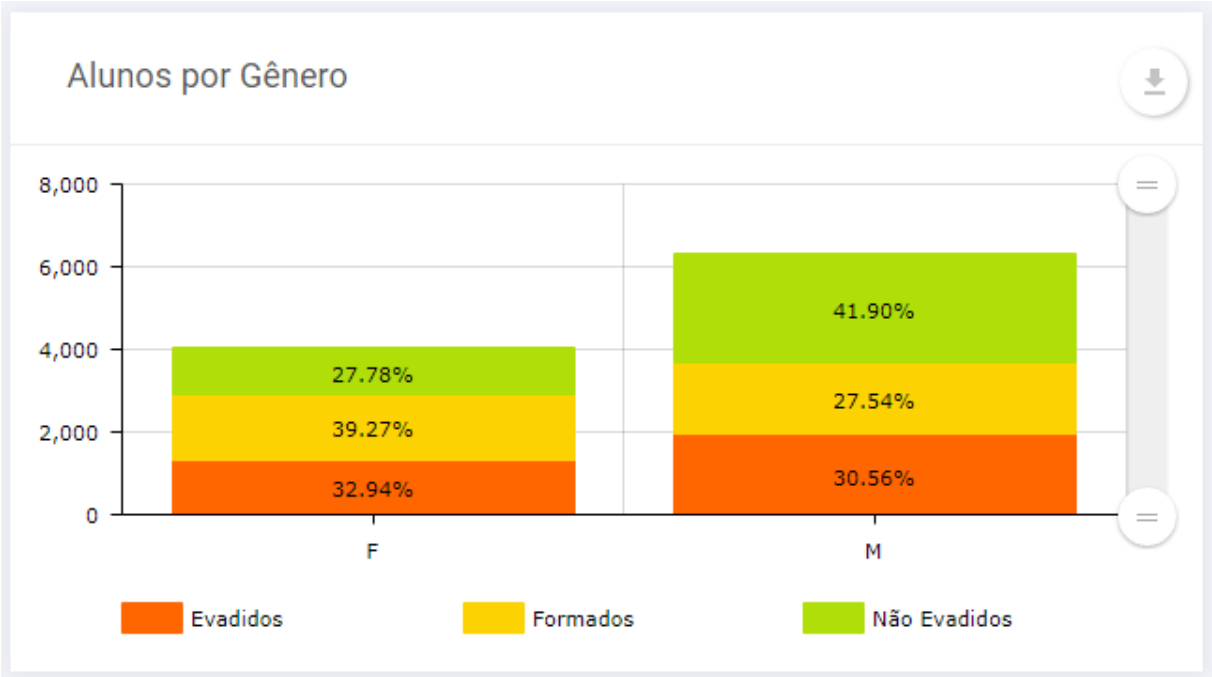

Figura 9 - Gráfico de alunos por gênero e situação resumida (*Dashboard* Instituição)
Fonte: Autoria própria.

A Figura 10 apresenta a tabela de "Campus da Instituição" nela temos a relação de campus cadastrados, assim como o total de cursos, alunos evadidos, não evadidos, formados e Alunos com Alta Probabilidade de Evasão (AAPE) que é a soma de alunos com mais de 60,00%[3] de probabilidade de evadir de cada um destes campus.

[3] O limiar *p(evasão)>60%* foi definido ad hoc para classificar os alunos mais "tendentes" à evasão.

Figura 10 - Tabela com detalhes de cada Campus (*Dashboard* Instituição)
Fonte: Autoria própria.

Além dos indicadores, gráficos e tabela apresentados anteriormente, são fornecidos os seguintes gráficos no *Dashboard* Instituição: Alunos por Idade de Ingresso, Alunos por Idade de Situação, Alunos por Ano/Semestre, Alunos por Quantidade de Semestres Cursados e Alunos por Período.

No caso de o usuário for um Administrador do Campus o mesmo será redirecionado para a página “Campus”, como é apresentado na Figura 11. Esta página é composta por indicadores, gráficos e tabelas que demostram o atual cenário do campus e de seus cursos, estando acessível aos usuários: Desenvolvedor, Administrador da Instituição e Administrador do Campus.

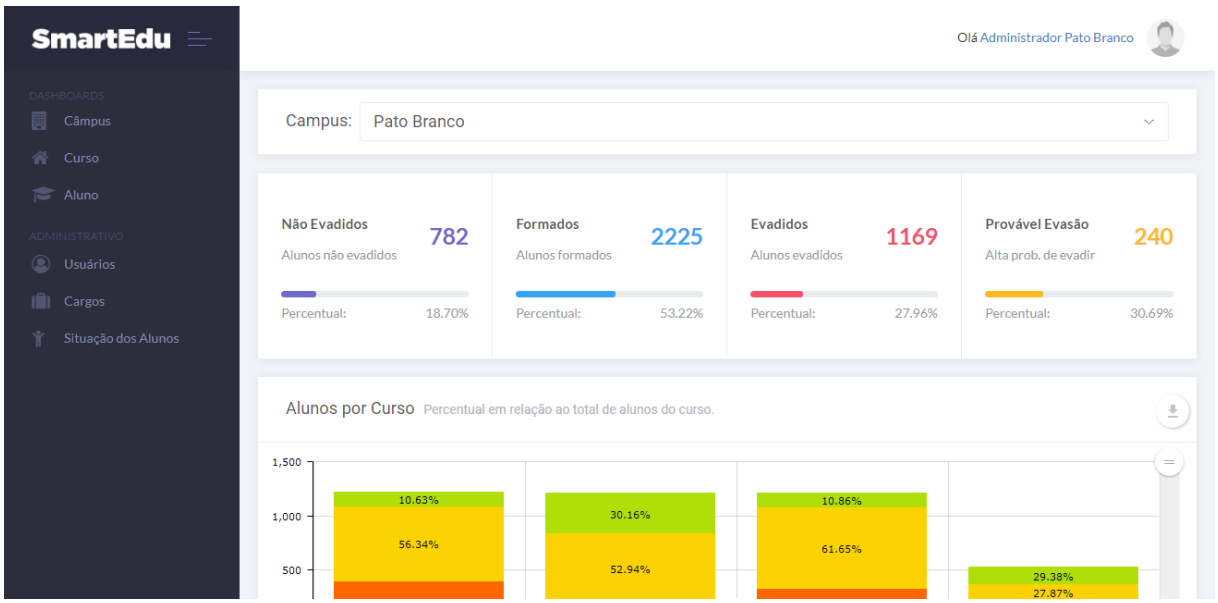

Figura 11 - *Dashboard* Campus (Usuário: Administrador do Campus)
Fonte: Autoria própria.

A Figura 12 apresenta os indicadores gerais do campus, como a soma de alunos não evadidos, formados e evadidos, e de seu percentual em relação a todos os alunos do campus, apresenta também o total de alunos com alta probabilidade de evasão e seu percentual em relação aos alunos não evadidos.

Figura 12 - Indicadores gerais do campus (*Dashboard* Campus)
Fonte: Autoria própria.

A Figura 13 apresenta o gráfico de “Alunos por Curso” que fornece o percentual de alunos por situação resumida (Evadidos, Formados e Não Evadidos) de cada curso do campus.

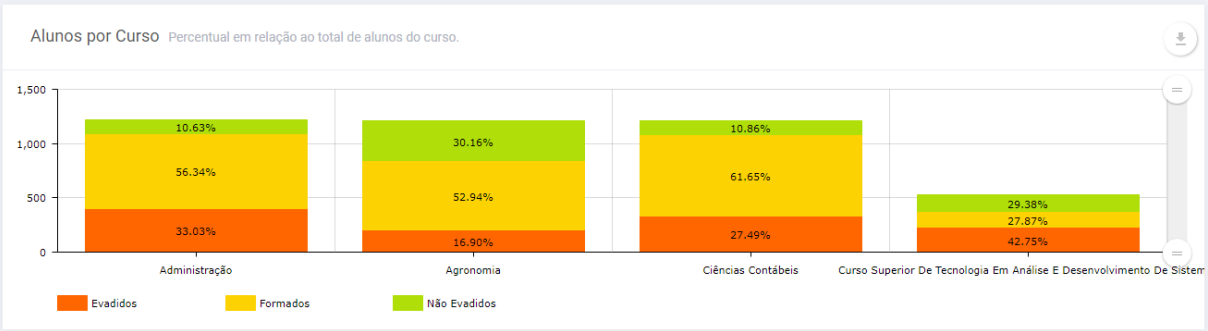

Figura 13 - Gráfico de alunos por curso e situação resumida (*Dashboard* Campus)
Fonte: Autoria própria.

A Figura 14 apresenta o gráfico de “Alunos por Quantidade de Semestres Cursados” que fornece o percentual de alunos por quantidade de semestres cursados e por situação resumida (Evadidos, Formados e Não Evadidos) do campus.

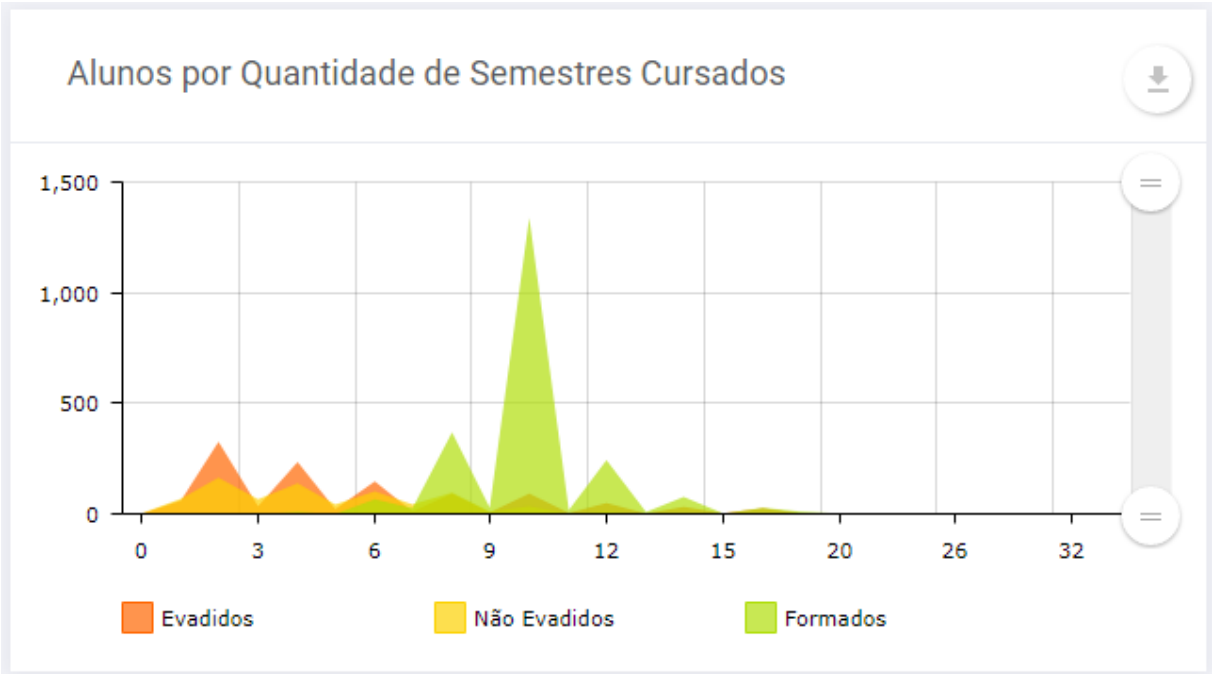

Figura 14 - Gráfico de alunos por quantidade de semestres cursados e situação resumida (*Dashboard* Campus)
Fonte: Autoria própria.

A Figura 15 apresenta a tabela de “Cursos do Campus” nela temos a relação de cursos cadastrados, assim como o total de alunos evadidos, não evadidos, formados e os AAPE de cada um destes cursos.

Figura 15 - Tabela com detalhes de cada curso do campus (*Dashboard* Campus)
Fonte: Autoria própria.

Além dos indicadores, gráficos e tabela apresentados anteriormente, são fornecidos os seguintes gráficos no *Dashboard* Campus: Alunos por Idade de Ingresso, Alunos por Idade de Situação, Alunos por Ano/Semestre, Alunos por Gênero e Alunos por Período.

Caso o usuário seja um Administrador da Instituição ou Desenvolvedor, o mesmo terá acesso a todos os campos (Figura 16).

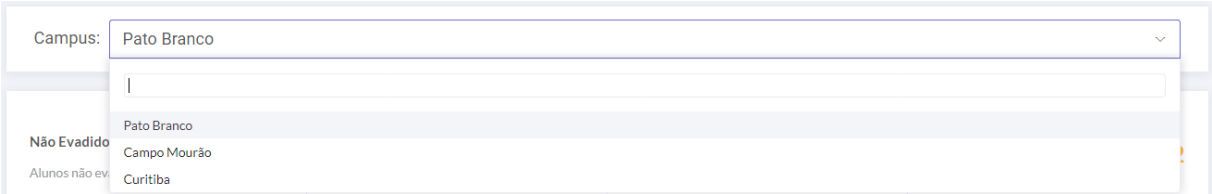

Figura 16 - Listas de campus (Usuário: Administrador da Instituição ou Desenvolvedor)
Fonte: Autoria própria.

Caso o usuário seja um Administrador do Campus, terá acesso apenas ao campus selecionado a ele na hora do cadastro (Figura 17).

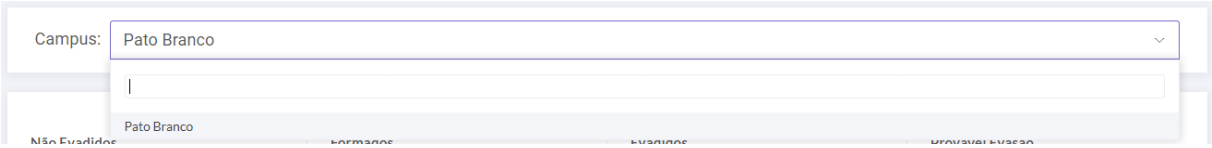

Figura 17 - Listas de campus (Usuário: Administrador do Campus)
Fonte: Autoria própria.

Na hipótese de que o usuário seja um Coordenador de Curso ou Professor o mesmo será redirecionado para a página “Curso”, como é ilustrado na Figura 18, esta página é composta por gráficos e tabela que demostram o atual cenário do curso e de seus alunos. Esta página é acessível a todos os usuários do sistema.

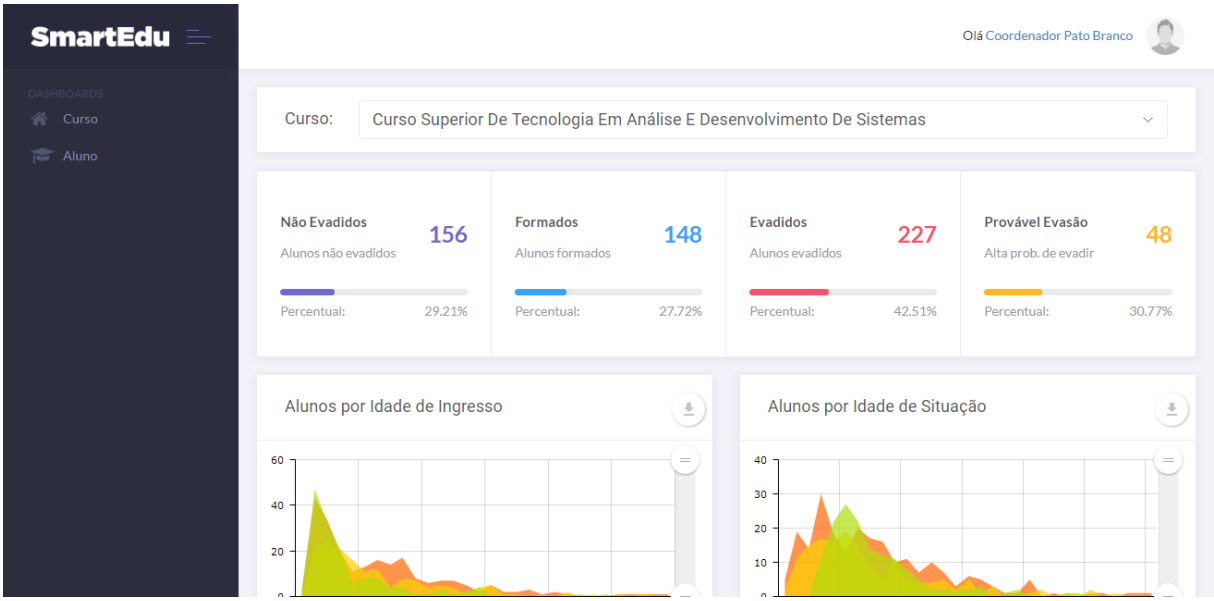

Figura 18 - *Dashboard* Curso (Usuário: Coordenador de Curso)
Fonte: Autoria própria.

A Figura 19 apresenta os indicadores gerais do curso, como a soma de alunos não evadidos, formados e evadidos, e de seu percentual em relação a todos os alunos

do curso, apresenta também o total de alunos com alta probabilidade de evasão e seu percentual em relação aos alunos não evadidos.

Figura 19 - Indicadores gerais do curso (*Dashboard* Curso)
Fonte: Autoria própria.

A Figura 20 apresenta o gráfico de “Alunos por Período” que fornece o percentual de alunos por período e por situação resumida (Evadidos e Não Evadidos) do curso.

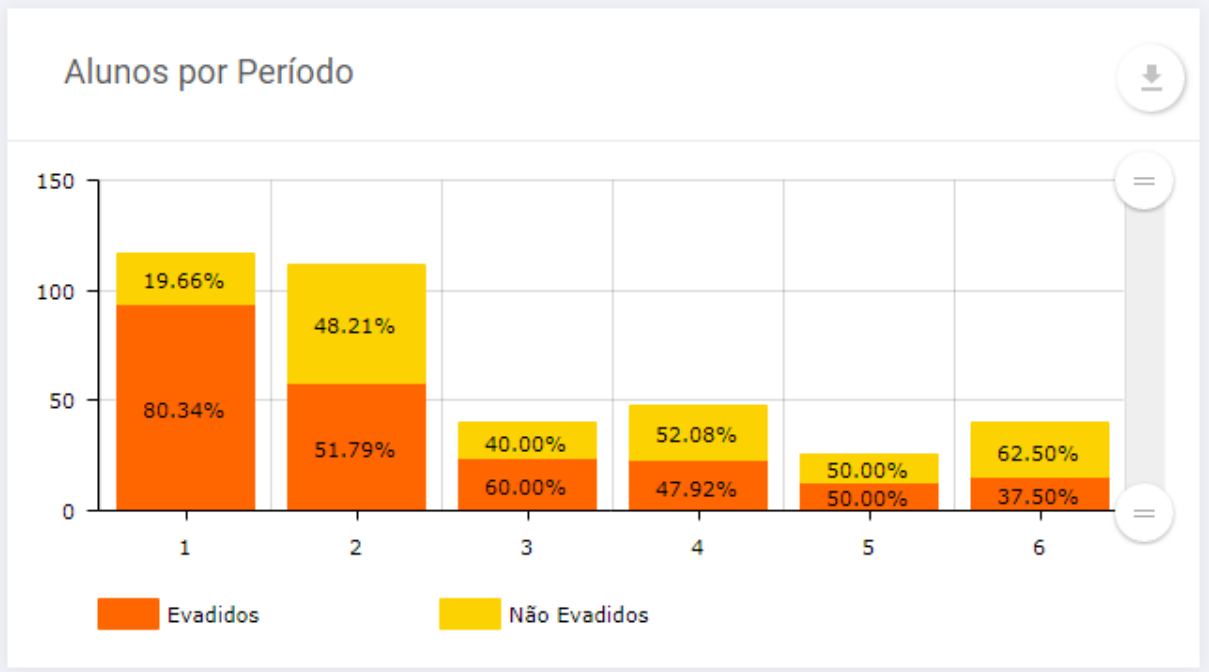

Figura 20 - Gráfico de alunos por período e situação resumida (*Dashboard* Curso)
Fonte: Autoria própria.

A Figura 21 apresenta o gráfico de “Alunos por Ano/Semestre” que fornece o percentual de alunos por ano/semestre e situação resumida (Evadidos e Formados) do curso.

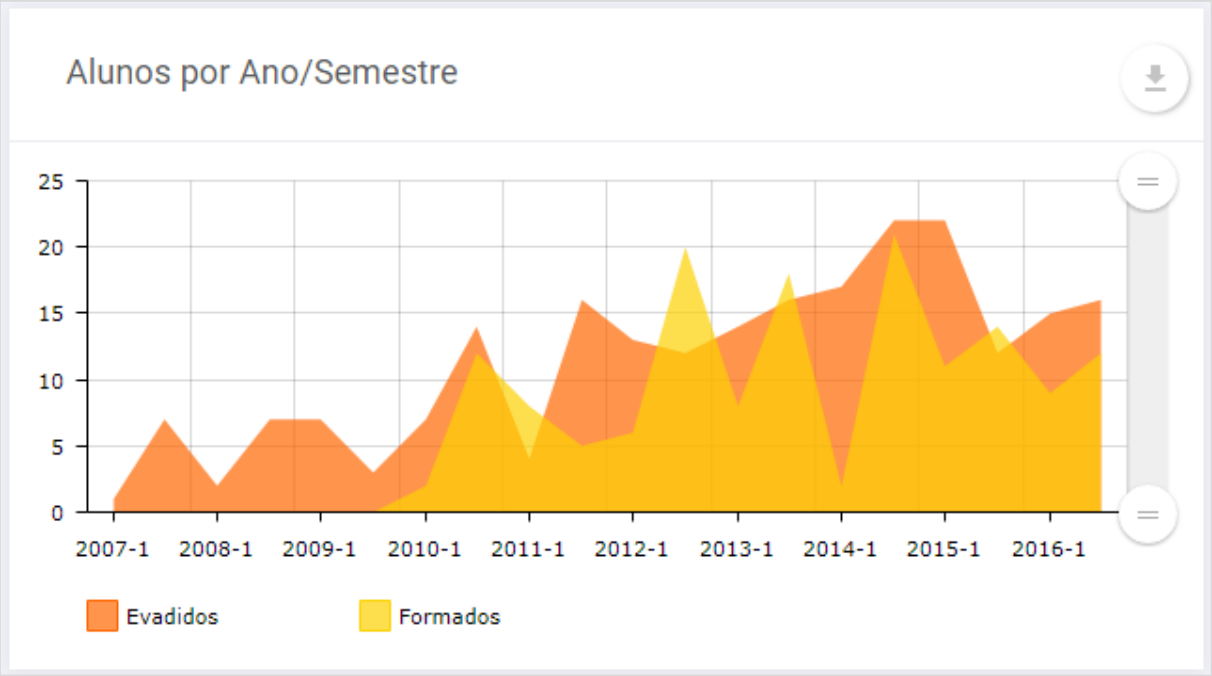

Figura 21 - Gráfico de alunos por ano/semestre e situação resumida (*Dashboard* Curso)
Fonte: Autoria própria.

A Figura 22 apresenta a tabela de “Alunos” nela são apresentadas as informações dos alunos do curso, tais como: nome, ingresso (semestre/ano de ingresso), período, probabilidade de evasão, situação e status e entre outras.

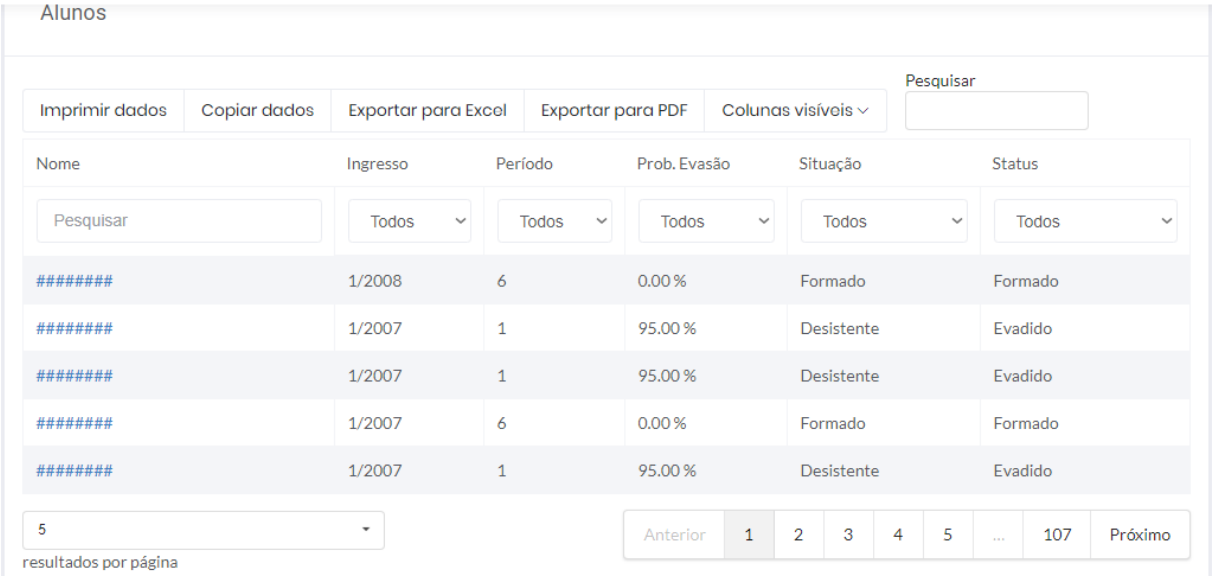

Figura 22 - Tabela com detalhes dos alunos do curso (*Dashboard* Curso)
Fonte: Autoria própria.

Além dos indicadores, gráficos e tabela apresentados anteriormente, são fornecidos os seguintes gráficos no *Dashboard* Curso: Alunos por Idade de Ingresso,

Alunos por Idade de Situação, Alunos por Gênero e Alunos por Quantidade de Semestres Cursados.

Caso o usuário seja um Administrador da Instituição ou Desenvolvedor, o mesmo terá acesso a todos os cursos de todos os câmpus (Figura 23), caso seja um Administrador do Campus terá acesso aos cursos do campus selecionado a ele (Figura 24), e caso o mesmo seja um Coordenador de Curso ou Professor, terá acesso aos cursos atrelados a ele no cadastro (Figura 25), podendo ser 1 (um) ou mais, desde que sejam do mesmo campus.

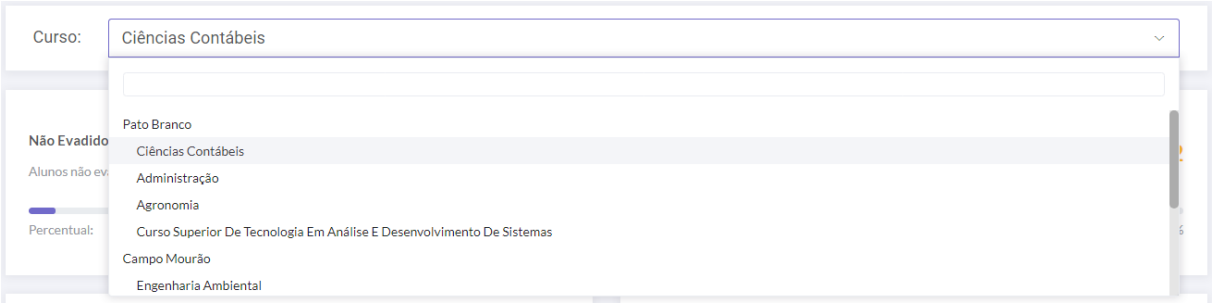

Figura 23 - Lista de cursos (Usuário: Administrador da Instituição ou Desenvolvedor)
Fonte: Autoria própria.

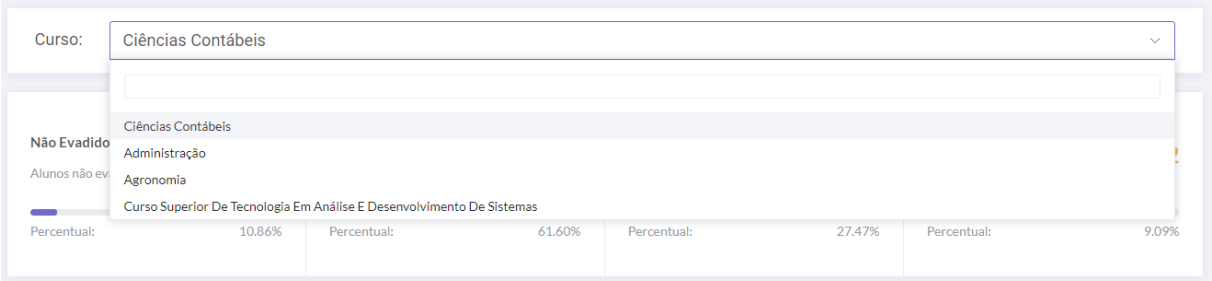

Figura 24 - Lista de cursos (Usuário: Administrador do Campus)
Fonte: Autoria própria.

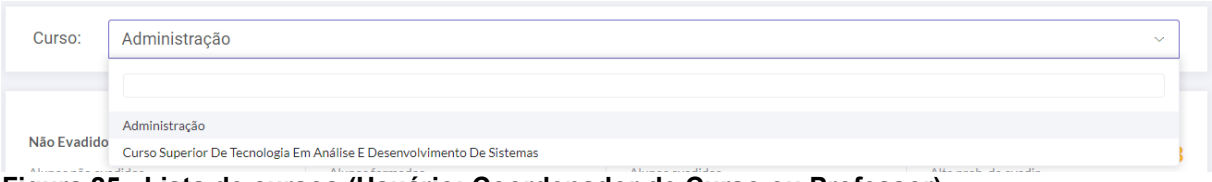

Figura 25 - Lista de cursos (Usuário: Coordenador de Curso ou Professor)
Fonte: Autoria própria.

A Figura 26 apresenta a página de “Aluno”, no qual são apresentadas as informações de gerais de cada aluno, seu histórico semestre à semestre e sua probabilidade de evasão que é gerada pelo sistema. No exemplo, com o intuito de preservar os dados pessoais dos alunos assim como sua imagem, os dados foram substituídos por “#########”. Esta página é acessível a todos os usuários do sistema.

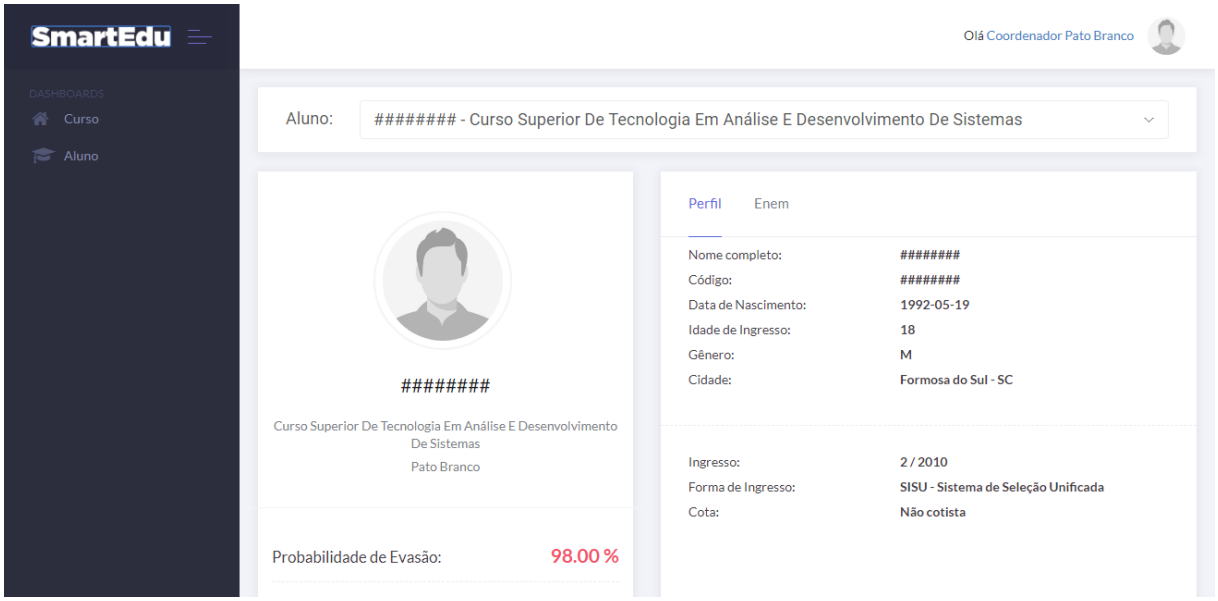

Figura 26 - Página Aluno (Usuário: Coordenador de Curso ou Professor)
Fonte: Autoria própria.

A Figura 27 apresenta a tabela de “Detalhes por Semestre Cursado”. Nessa tabela são apresentadas as informações semestre a semestre que o aluno cursou, as informações são as seguintes: Sem/Ano Situação (Semestre/Ano da Situação), CR (Coeficiente de Rendimento), Disciplinas Consignadas, Disciplinas Matriculadas, Disciplinas Reprovadas por Frequência, Disciplinas Reprovadas por Nota, Idade, Período e Situação.

Figura 27 - Tabela de detalhes por semestre cursado do aluno (Página Aluno)
Fonte: Autoria própria.

A Figura 28 apresenta a página de Usuários, no qual é apresentada uma tabela com os usuários cadastrados no sistema, assim como as opções de editar e

deletar registros. Esta página é acessível aos usuários Desenvolvedor e Administrador da Instituição.

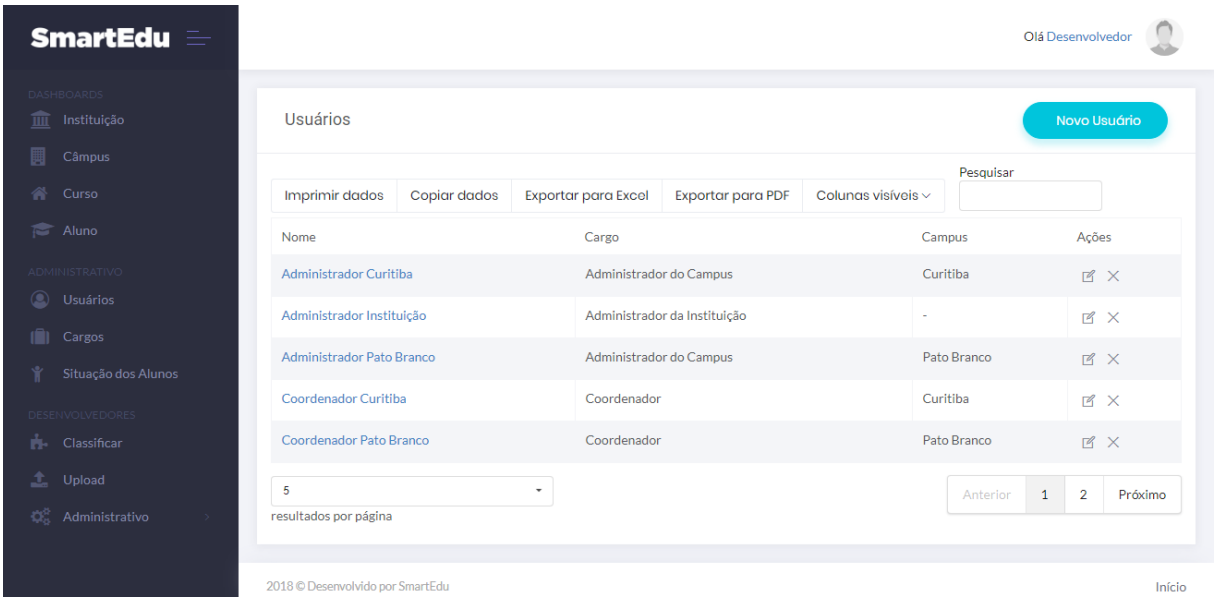

Figura 28 - Página Usuário
Fonte: Autoria própria.

Ao clicar no botão de “Novo Usuário”, o usuário será redirecionado para a página de criação de um novo usuário, aonde deve-se preencher os dados que são obrigatórios sendo estes representados pelo asterisco ao lado do nome de cada um dos campos, como representado na Figura 29, caso o mesmo não seja feito, o sistema apresentará uma mensagem requisitando que os campos que são obrigatórios sejam preenchidos, assim representado pela Figura 29.

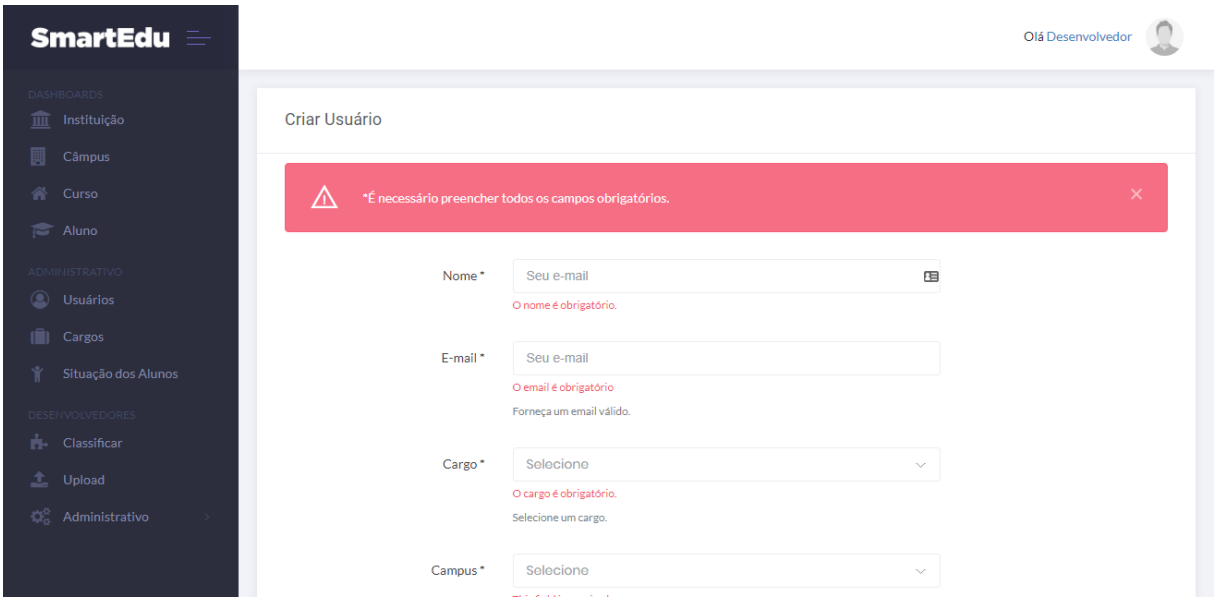

Figura 29 - Página Novo Usuário
Fonte: Autoria própria.

Ao clicar no botão de “Excluir” da tabela de “Usuários” apresentada na Figura 29, é aberto um *modal* (Figura 30) com as opções “Cancelar”, que cancela a ação, e a opção “Deletar”, que executa a ação de apagar o registro do usuário selecionado.

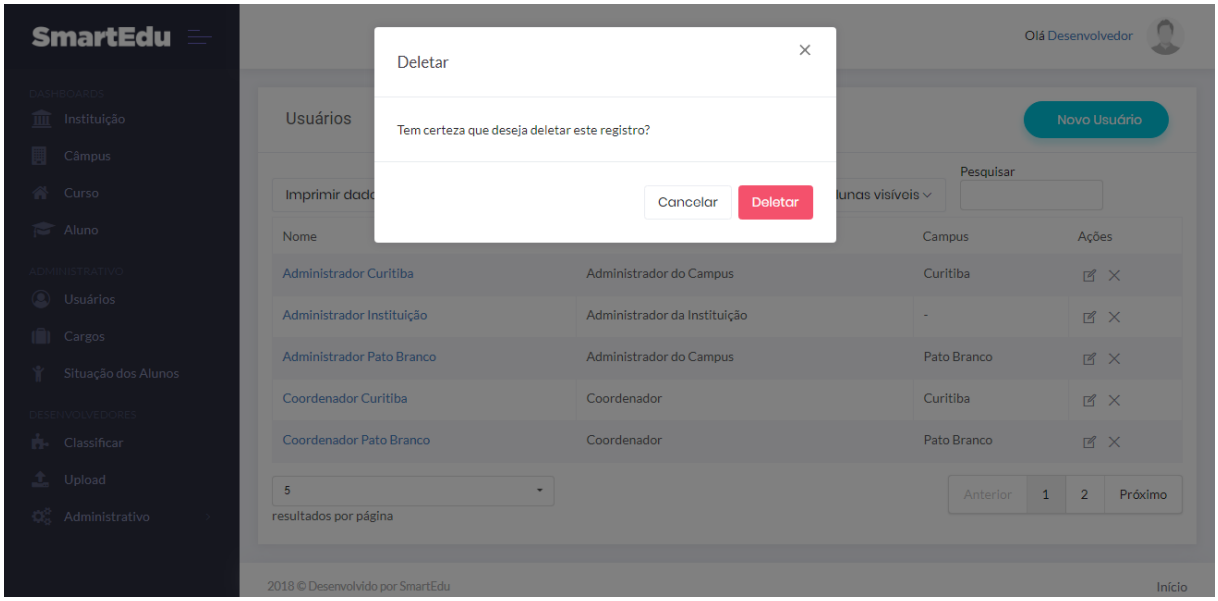

Figura 30 - Página usuário função deletar
Fonte: Autoria própria.

Ao clicar no botão “Editar”, o usuário será redirecionado a página de edição Figura 31, aonde será possível alterar o registro do usuário selecionado.

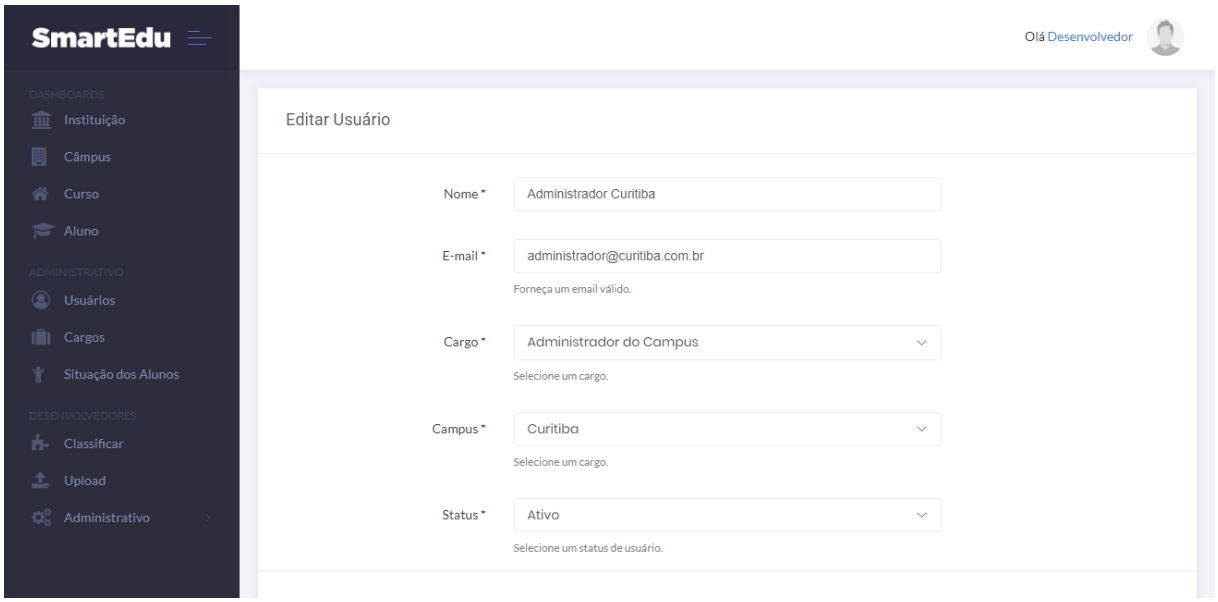

Figura 31 - Página Editar Usuário
Fonte: Autoria própria.

A Figura 32 apresenta a página de “Cargos”, no qual é apresentada uma tabela com os cargos cadastrados no sistema, os cargos não podem ser alterados ou

deletados. Esta página é acessível aos usuários Desenvolvedor e Administrador da Instituição.

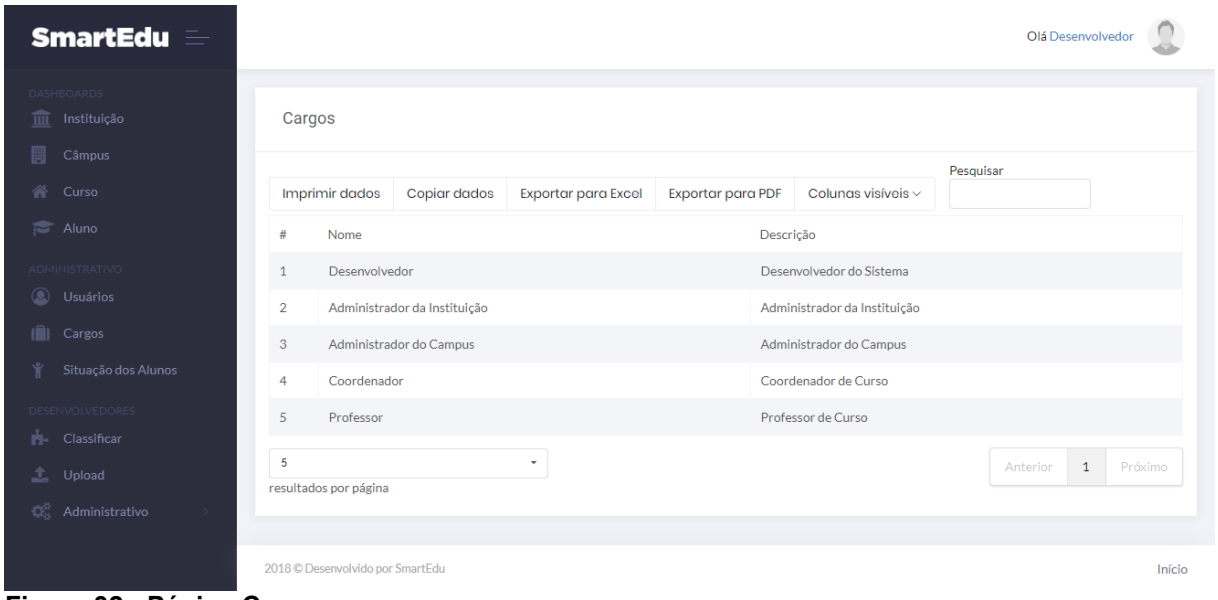

Figura 32 - Página Cargos
Fonte: Autoria própria.

A Figura 33 apresenta a página de “Situação dos Alunos”, no qual é apresentada uma tabela com as possíveis situações dos alunos da instituição, esta tabela é povoada dinamicamente no processo de upload dos dados, aonde o mesmo identifica todas as possíveis situações de aluno. As situações não podem ser deletadas, apenas é possível realizar a edição dos campos “Descrição” e “Situação Resumida”, através do botão “Editar”. Esta página é acessível aos usuários Desenvolvedor e Administrador da Instituição.

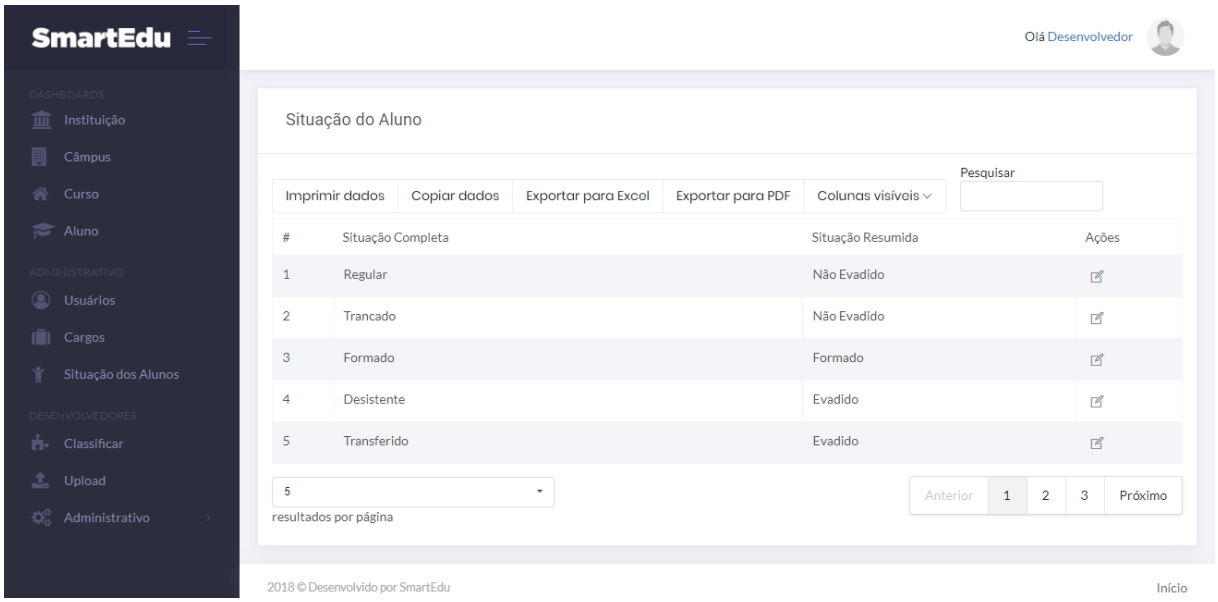

Figura 33 - Página Situação do Aluno
Fonte: Autoria própria.

O Quadro 30 apresenta a relação completa de todas as possíveis situações dos alunos identificadas nessa base de dados, elas foram agrupadas desta forma com a intenção de diminuir as possíveis classes.

Situação Completa	**Situação Resumida**
Desistente	Evadido
Jubilado	Evadido
Mudou de Curso	Evadido
Transferido	Evadido
Formado	Formado
Afastado para estudos no exterior	Não Evadido
Enade pendente	Não Evadido
Regular	Não Evadido
Trancado	Não Evadido
Em mobilidade (intercâmpus)	Outro
Expulso	Outro
Falecido	Outro
Matrícula sub judice	Outro

Quadro 30 - Relação de situações dos alunos
Fonte: Autoria própria.

A Figura 34 apresenta a página “Classificar”, na qual é apresentada gráficos e tabelas demonstrando os resultados obtidos no processo de classificação de forma geral e individual de cada um dos cursos, assim como o botão “Gerar Classificação”, responsável por executar o processo de classificação.

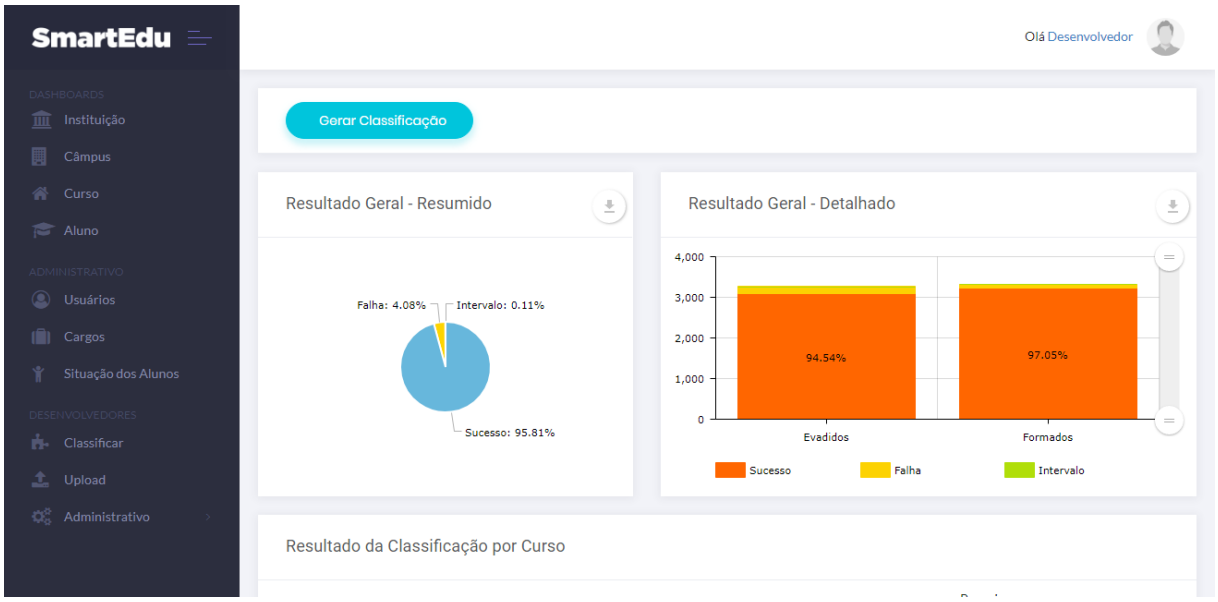

Figura 34 - Página Classificar
Fonte: Autoria própria.

A Figura 35 apresenta o gráfico de “Resultado Geral - Resumido” que fornece a média percentual de sucesso, intervalo e falha geral entre todos os cursos analisados.

Figura 35 - Gráfico de resultado geral entre todos os cursos analisados (Página de Classificar)
Fonte: Autoria própria.

A Figura 36 apresenta o gráfico de “Resultado Geral - Detalhado” que fornece a média percentual de sucesso, intervalo e falha dividido entre alunos Evadidos e Formados, entre todos os cursos analisados.

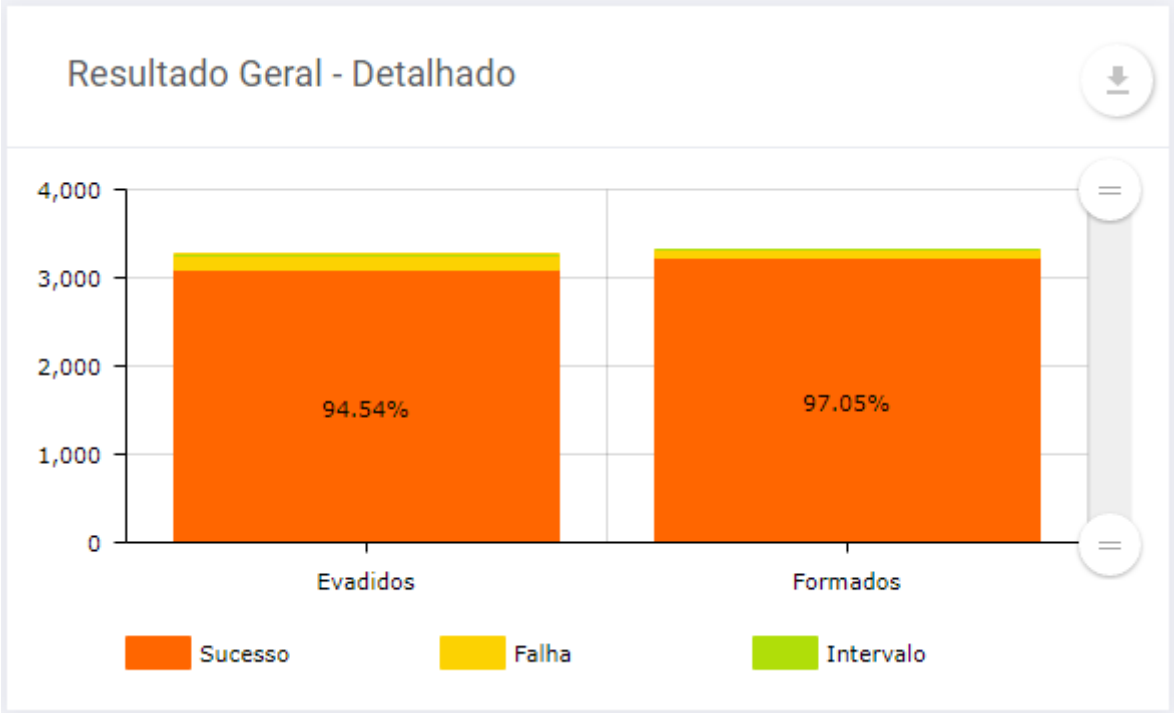

Figura 36 - Gráfico de resultado geral detalhado entre todos os cursos analisados (Página de Classificar)
Fonte: Autoria própria.

A Figura 37 apresenta a tabela de “Resultado da Classificação por Curso”. Nessa tabela são apresentados os resultados (sucesso, intervalo e falha) da classificação para cada um dos cursos, assim como qual o classificador e variáveis foram utilizadas.

Resultado da Classificação por Curso

Imprimir dados | Copiar dados | Exportar para Excel | Exportar para PDF | Colunas visíveis

Pesquisar

Classificador	Variaveis	Campus	Curso	Sucesso(%)	Intervalo(%)	Falha(%)
J48	Disciplinas aprovadas - Idade	Pato Branco	Administração	92.50 %	0.27 %	7.23 %
Lwl	Disciplinas aprovadas - Disciplinas reprovadas por frequência - Disciplinas reprovadas por nota	Pato Branco	Ciências Contábeis	92.34 %	0.00 %	7.66 %
Lwl	Disciplinas aprovadas - Idade	Campo Mourão	Engenharia Ambiental	98.74 %	0.00 %	1.26 %
Lwl	Disciplinas aprovadas	Curitiba	Engenharia Elétrica	99.27 %	0.00 %	0.73 %
Lwl	Disciplinas aprovadas	Curitiba	Curso Superior De Tecnologia Em Design Gráfico	90.73 %	0.00 %	9.27 %

5 resultados por página

Anterior | 1 | 2 | Próximo

Figura 37 - Tabela de resultados da classificação por curso (Página de Classificar)
Fonte: Autoria própria

O Quadro 31 apresenta os resultados obtidos no processo de classificação. Nele são apresentados os resultados (sucesso, intervalo e falha) da classificação para cada um dos cursos, assim como qual o classificador e variáveis foram utilizadas.

Classificador	Variáveis	Campus	Curso	Sucesso (%)	Intervalo (%)	Falha (%)
Naivebayes	Coeficiente de rendimento - Disciplinas aprovadas - Disciplinas consignadas - Disciplinas reprovadas por frequência - Semestre de ingresso no curso	Campo Mourão	Curso Superior De Tecnologia Em Alimentos	99,32 %	0,14 %	0,54 %
Lwl	Disciplinas aprovadas - Idade	Campo Mourão	Engenharia Ambiental	98,74 %	0,00 %	1,26 %
Lwl	Disciplinas aprovadas	Campo Mourão	Engenharia Civil	99,00 %	0,00 %	1,00 %
Lwl	Disciplinas aprovadas	Curitiba	Curso Superior De Tecnologia Em Design Gráfico	90,73 %	0,00 %	9,27 %
Lwl	Disciplinas aprovadas	Curitiba	Engenharia Elétrica	99,27 %	0,00 %	0,73 %
Naivebayes	Coeficiente de rendimento - Disciplinas aprovadas - Forma de ingresso	Curitiba	Engenharia Mecânica	97,97 %	0,25 %	1,78 %
J48	Disciplinas aprovadas - Idade	Pato Branco	Administração	92,50 %	0,27 %	7,23 %
Naivebayes	Coeficiente de rendimento - Disciplinas aprovadas - Disciplinas reprovadas por nota - Idade	Pato Branco	Agronomia	97,39 %	0,24 %	2,38 %
Lwl	Disciplinas aprovadas - Disciplinas reprovadas por frequência - Disciplinas reprovadas por nota	Pato Branco	Ciências Contábeis	92,34 %	0,00 %	7,66 %
Lwl	Coeficiente de rendimento - Disciplinas aprovadas - Disciplinas consignadas - Disciplinas matriculadas - Forma de ingresso - Nota ENEM Liguagem - Nota final SISU	Pato Branco	Tecnologia Em Análise E Desenvolvimento De Sistemas	97,60 %	0,00 %	2,40 %

Quadro 31 - Resultado do processo de classificação
Fonte: Autoria própria.

As taxas de sucesso, intervalo e falha foram definidas sobre o conjunto de registros empregado para treinamento dos classificadores e teste do sistema. Cada classificação foi executada com um determinado classificador e um conjunto de variáveis, gerando uma estimativa de probabilidade de evasão atribuída a cada estudante do conjunto de testes. Cada classificador foi treinado e validado com base nos dados de alunos de um curso do conjunto de testes, empregando-se a estratégia *leave-one-out* (ver Seção 2.4.3.1) e considerando apenas dos registros de alunos que tinham uma situação final definitiva na base de testes, seja “evadido”, seja “formado”.

Em seguida, o registro deixado de fora do treinamento era classificado. O classificador era aplicado a este registro gerando uma probabilidade de evasão estimada. Caso a probabilidade estimada de evasão fosse menor ou igual a 40,00% seria considerada baixa probabilidade de evadir, classificando o aluno no grupo que “não irá evadir”; se fosse maior que 40,00% e menor ou igual a 60,00% seria considerado como neutro e ficaria no grupo “intervalo”, classificando o aluno como inconclusivo; e se fosse maior que 60,00% seria considerado com alta probabilidade de evadir, classificando o aluno no grupo que “irá evadir”.

Em seguida, a classificação gerada para o aluno (não evasão, inconclusivo, evasão) era comparada à situação real do aluno (evadido, formado). A classificação do registro individual de um aluno foi considerada “sucesso” quando indicou evasão para um aluno realmente evadido, ou quando indicou não evasão para um aluno realmente formado. De maneira similar, a classificação do registro individual de um aluno foi considerada “falha” quando indicou evasão para um aluno que na realidade era formado, ou quando não indicou evasão para um aluno que na realidade era evadido. Os demais casos (inconclusivos) foram considerados “intervalo”.

O processo de treinamento seguido da classificação era repetido para todos os alunos de um curso, gerando as taxas percentuais de classificações corretas (“sucesso”), neutras (“intervalo”) e incorretas (“falha”) apresentadas no Quadro 31.

No Quadro 31, o percentual de “Sucesso” na classificação para um dado curso é a quantidade relativa de classificações bem-sucedidas e o percentual de “Fracasso” é a quantidade relativa de classificações malsucedidas, geradas em ambos os casos por um mesmo classificador e conjunto de variáveis em relação ao número total de registros do curso em questão. O percentual de “Intervalo” é a quantidade relativa de classificações não conclusivas geradas pelo classificador para o curso.

A Figura 38 apresenta a página “Upload”, através desta página podemos fazer o carregamento dos dados dos alunos da instituição. Ao clicar no botão “Escolher arquivos” é possível selecionar 1 (um) ou mais arquivos no formato “.xlsx”, os arquivos devem conter um cabeçalho em cada coluna identificando de qual dado a coluna se refere, como apresentado no Quadro 32.

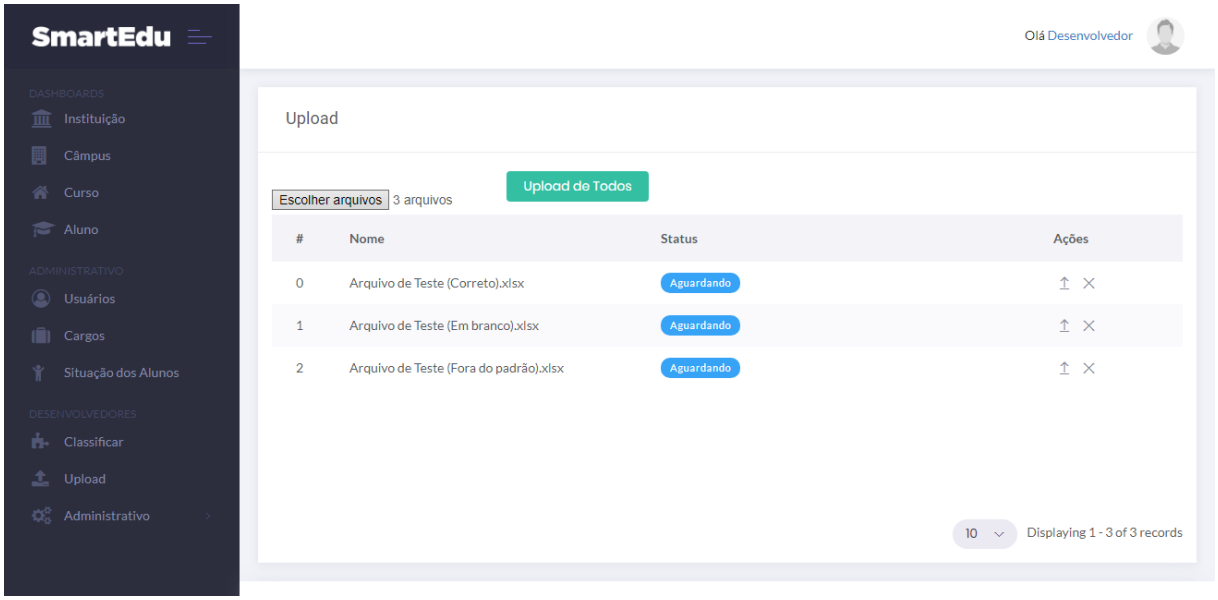

Figura 38 - Página Upload
Fonte: Autoria própria.

Ano	Semestre	Disciplinas aprovadas	Disciplinas consignadas	Disciplinas matriculadas	Disciplinas reprovadas por frequência	Disciplinas reprovadas por nota
2008	1	8	0	8	0	0
2007	1	7	0	0	0	1
2007	2	0	0	0	0	0
2007	1	0	0	8	5	0
2007	2	0	0	0	0	0
2007	1	7	0	8	0	1
2007	2	2	0	5	1	2
2007	1	7	0	8	0	1
2007	2	0	0	0	0	0
2007	1	7	0	8	0	1
2007	2	0	0	5	5	0

Quadro 32 - Exemplo de arquivo “.xlsx” com cabeçalho em cada coluna para carregamento
Fonte: Autoria própria.

Caso o arquivo esteja fora dos padrões ou em branco o sistema não realizara o upload destes dados e informara que existem erros no arquivo, conforme apresentado pela Figura 39.

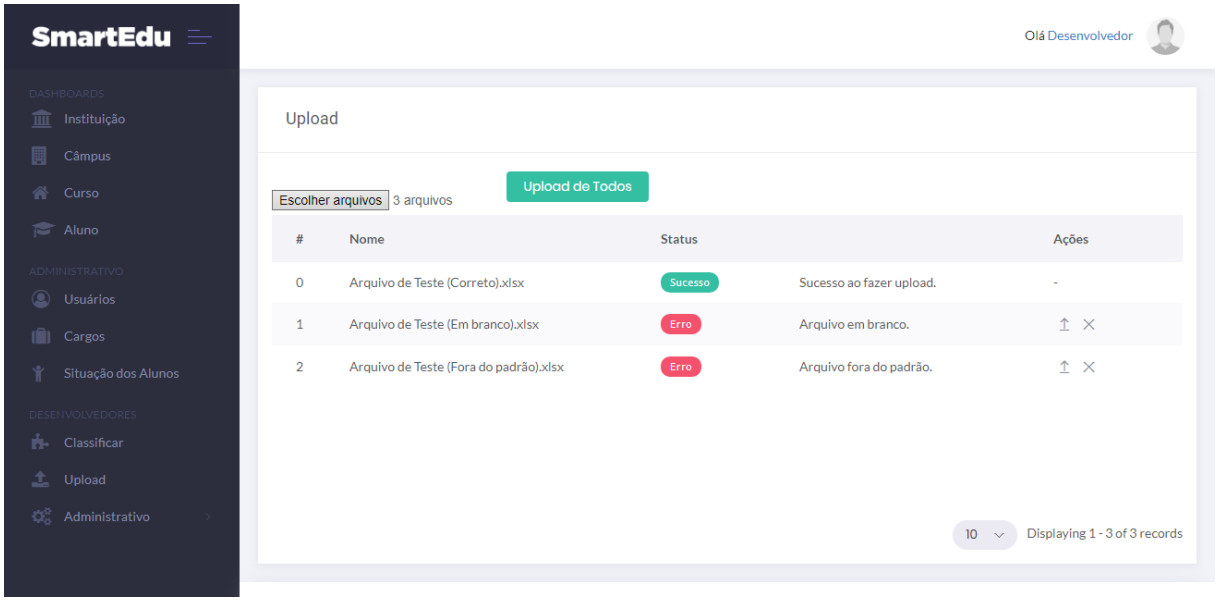

Figura 39 - Página *Upload* (Após click "Upload de Todos")
Fonte: Autoria própria.

Caso o arquivo esteja nos padrões adequados o sistema irá realizar o carregamento dos dados da planilha, povoando as tabelas do sistema, sendo elas: *Campus*, *Curso*, *Aluno*, *Detalhe* e *Situacao*. Vale ressaltar que o sistema valida se a informação já existe no banco de dados, evitando assim que existam registros duplicados.

A Figura 40 apresenta a página "Classificadores", na qual é apresentada quais os classificadores disponíveis no sistema, assim como é possível selecionar quais serão utilizados no processo de classificação, através da caixa de seleção "classificar".

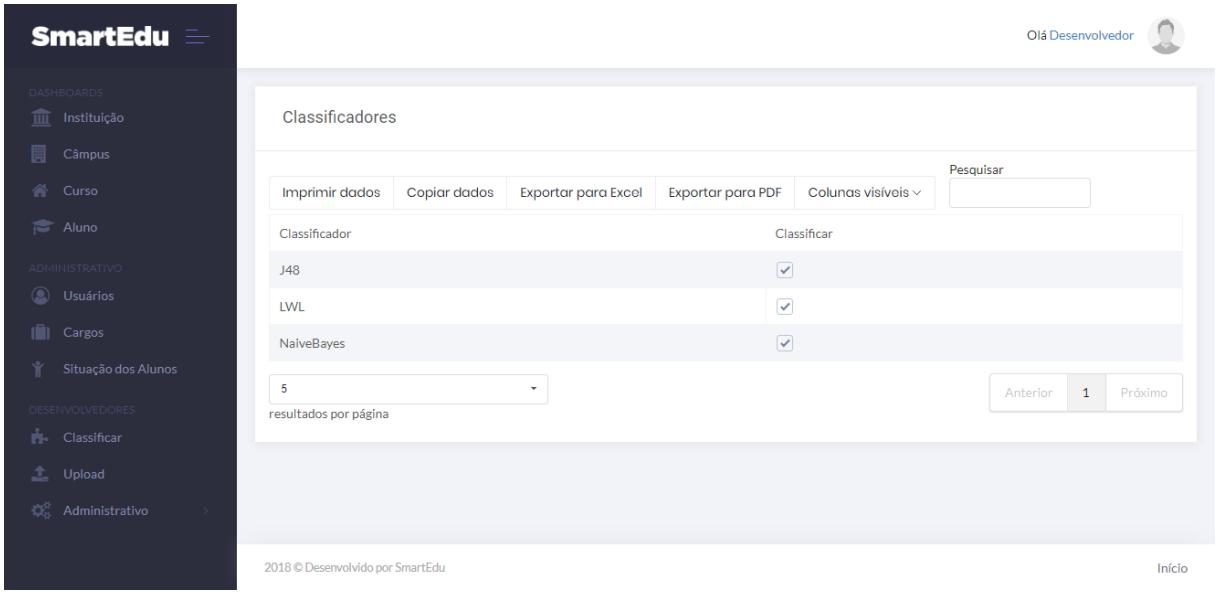

Figura 40 - Página Classificadores
Fonte: Autoria própria.

A Figura 41 apresenta a página “Variáveis”, na qual é apresentada quais as variáveis disponíveis no sistema, assim como é possível selecionar quais serão utilizados no processo de classificação, e quais passaram pelo procedimento de discretização.

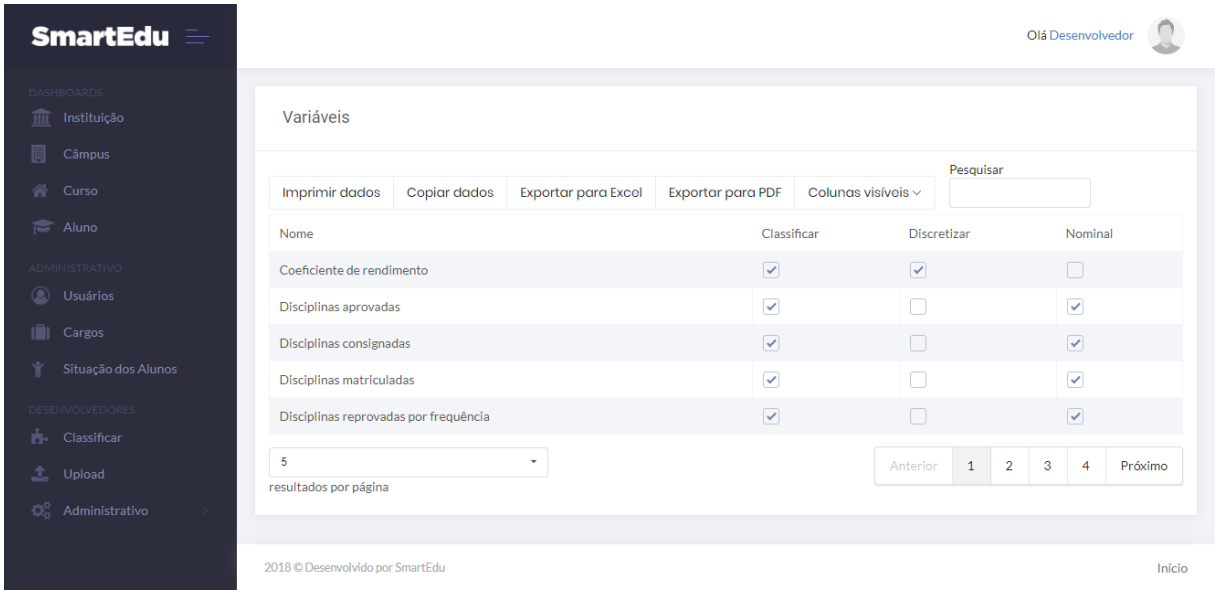

Figura 41 - Página Variáveis
Fonte: Autoria própria.

A Figura 42 apresenta a página “Cursos”, na qual é apresentada quais os cursos cadastrados no sistema, assim como é possível selecionar quais serão utilizados no processo de classificação.

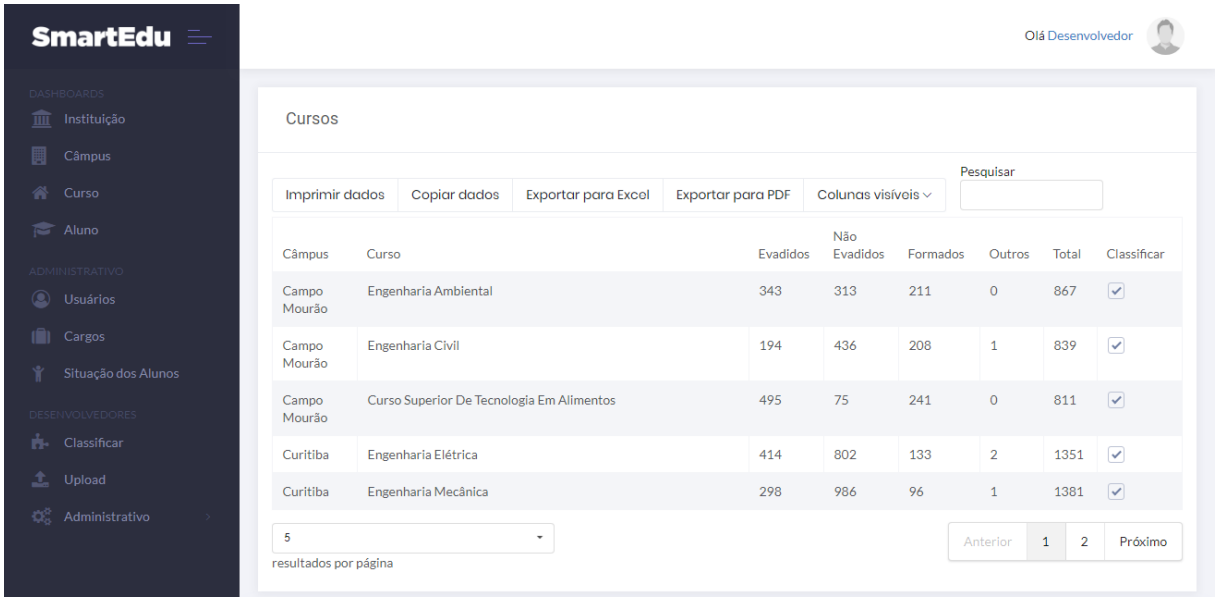

Figura 42 - Página Cursos
Fonte: Autoria própria.

Todas as tabelas do sistema apresentam as seguintes funcionalidades, demonstradas na Figura 43:

a) Imprimir dados: Imprime os dados da tabela;

b) Copiar dados: Copia os dados da tabela para sua área de transferência;

c) Exportar para Excel: Exporta os dados da tabela para um arquivo “.xlsx”;

d) Exportar para PDF: Exporta os dados da tabela para um arquivo “.pdf”;

e) Colunas visíveis: Através desta funcionalidade podemos selecionar o conjunto de colunas a serem exibidas;

f) Pesquisa: Realiza uma pesquisa através do texto digitado entre todos os dados apresentados;

g) Resultados por página: Através desta funcionalidade podemos selecionar quantos registros desejamos mostrar por página, as possibilidades são: 5, 10, 25 ou todos os registros;

h) Navegação entre as páginas: Este componente é responsável por selecionar a página a ser exibida na tabela;

Figura 43 - Funcionalidades das tabelas do sistema
Fonte: Autoria própria.

Todas os gráficos do sistema apresentam as seguintes funcionalidades, apresentadas a seguir e demonstradas na Figura 44:

a) Baixar como: Fazer o download do gráfico nos formatos “PNG”, “JPG”, “SVG” e “PDF”;

b) Salvar como: Permite salvar as informações do gráfico nos formatos “CSV”, “XLSX” e “JSON”;

c) Editar: Permite realizar edições no gráfico, como desenhar e inserir formas entre outras;

d) Imprimir: Realiza a impressão do gráfico.

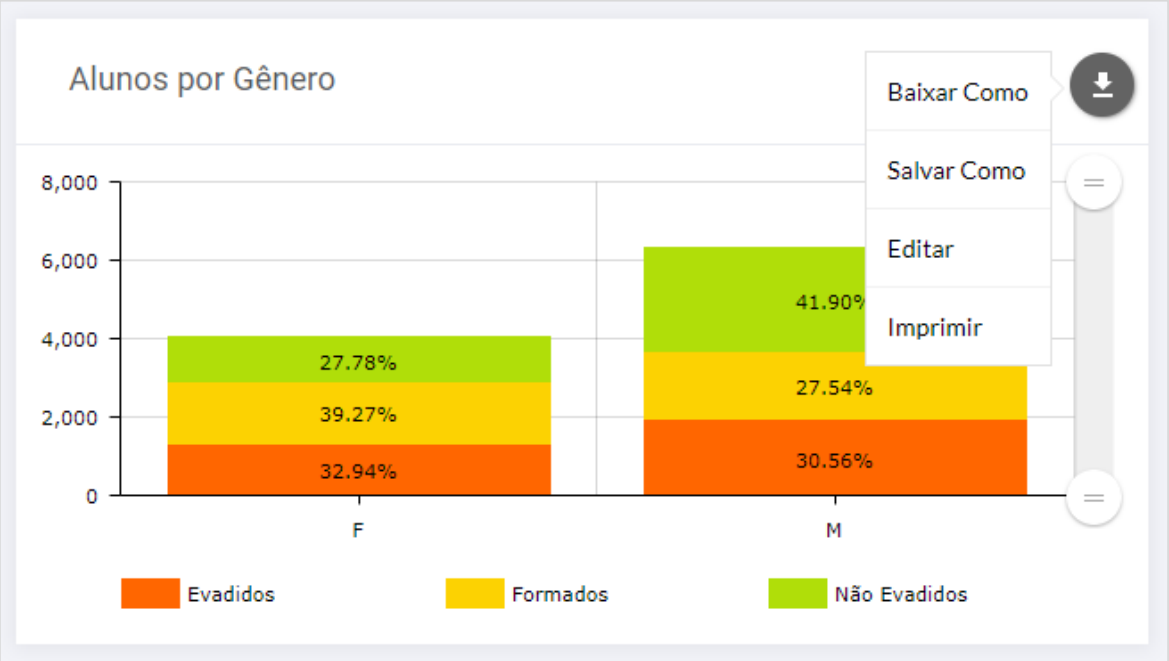

Figura 44 - Funcionalidades dos gráficos do sistema
Fonte: Autoria própria.

4.4 IMPLEMENTAÇÃO DO SISTEMA

A Subseção 4.4.1 apresenta a implementação do servidor de mineração e na Subseção 4.4.2 está descrita a implementação do sistema web.

4.4.1 Servidor de Mineração

A Listagem 1 apresenta o método connectDataBase, responsável realizar a conexão com o banco de dados e fornecer um *InstanceQuery* utilizado para realizar as consultas ao banco e converter estes dados em *Instances*, que é o formato padrão utilizado no processo de classificação da API Weka.

```
public InstanceQuery connectDataBase() throws Exception {
    DatabaseUtils databaseUtils = new DatabaseUtils(new
File("/DatabaseUtils.props"));
    InstanceQuery query = new InstanceQuery();
    query.setUsername("smartedu");
    query.setPassword("smartedu");
```

```
    return query;
}
```

Listagem 1 - DataBaseController – Método connectDataBase
Fonte: Autoria própria.

A Listagem 2 apresenta o método “getDataSet”, responsável por retornar as *Instances* que serão utilizadas no processo de classificação, nele é feito o processo de conexão com o banco de dados, criado o *SQL* e recebido os dados em formato de instancias, logo após é realizado o processo de discretização e por fim retornado os dados já discretizados.

```
public Instances getDataSet(List<Variable> variaveis, long idCurso, int type)
throws Exception {
    InstanceQuery query = connectDataBase();
    query.setQuery(createSQL(variaveis, idCurso));
    Instances dataSet = query.retrieveInstances();

    String col_Discretize = "";
    String col_NumericToNominal = "";
    int position = 2;

    for (Variable variable : variaveis) {
        if (variable.getDiscretize() == 1) {
            if ("".equals(col_Discretize)) {
                col_Discretize += "" + position;
            } else {
                col_Discretize += "," + position;
            }
        }
        if (variable.getNominal() == 1) {
            if ("".equals(col_NumericToNominal)) {
                col_NumericToNominal += "" + position;
            } else {
                col_NumericToNominal += "," + position;
            }
        }
        position++;
    }
    Discretize discretize = new Discretize();
    dataSet = discretize.discretize(dataSet, col_Discretize, col_NumericToNominal);
    return dataSet;
}
```

Listagem 2 - DataBaseController – Método getDataBase
Fonte: Autoria própria.

A Listagem 3 apresenta um exemplo do resultado obtido pelo método “getDataSet”, após receber os dados em formato de *Instances* e discretizar os mesmos.

```
@attribute id numeric
@attribute coefficient {'\'(-inf-0.093]\'','\'(0.093-0.186]\'','\'(0.186-0.279]\'','\'(0.279-0.372]\'','\'(0.372-0.465]\'','\'(0.465-0.558]\'','\'(0.558-0.651]\'','\'(0.651-0.744]\'','\'(0.744-0.837]\'','\'(0.837-inf)\'}
@attribute situation_short { Formado , Evadido }

@data
```

```
2413,\'(0.837-inf)\",Formado
2414,\'(0.744-0.837]\",Formado
2415,\'(0.744-0.837]\",Formado
2416,\'(0.651-0.744]\",Formado
2417,\'(0.651-0.744]\",Formado
2418,\'(0.744-0.837]\",Formado
2419,\'(0.558-0.651]\",Formado
2422,\'(0.651-0.744]\",Formado
2423,\'(0.651-0.744]\",Formado
2424,\'(0.558-0.651]\",Formado
2426,\'(0.744-0.837]\",Formado
2897,\'(0.465-0.558]\",Evadido
2898,\'(0.093-0.186]\",Evadido
2902,\'(0.558-0.651]\",Evadido
2904,\'(0.558-0.651]\",Evadido
2430,\'(0.744-0.837]\",Formado
2431,\'(0.651-0.744]\",Formado
2432,\'(0.651-0.744]\",Formado
```

Listagem 3 - Exemplo de resultado obtido pelo método getDataSet
Fonte: Autoria própria.

A Listagem 4 apresenta o método “createSQL”, responsável montar a consulta com a lista de variáveis fornecida para cada curso.

```
public String createSQL(List<Variable> variaveis, long idCurso) throws Exception {
    String sql = "SELECT student.id,";
    for (Variable variable : variaveis) {
        sql += " \n" + variable.getTable() + "." + variable.getName_database() +
",";
    }

    sql += "\nsituation.situation_short\n";

    sql += "FROM student\n"
            + "LEFT JOIN detail\n"
            + "ON detail.student_id = student.id\n"
            + "LEFT JOIN situation\n"
            + "ON situation.id = detail.situation_id\n"
            + "LEFT JOIN curso\n"
            + "ON curso.id = student.curso_id\n"
            + "LEFT JOIN campus\n"
            + "ON campus.id = curso.campus_id\n"
            + "WHERE detail.loading_period = (SELECT MAX(detail.loading_period)
FROM detail WHERE detail.student_id = student.id)\n"
            + "AND curso.id =  " + idCurso + "\n"
            + "AND (situation.situation_short NOT LIKE 'Outro')\n"
            + "GROUP BY student.id\n"
            + "ORDER BY student.id";
    return sql;
}
```

Listagem 4 - DataBaseController – Método createSQL
Fonte: Autoria própria.

A Listagem 5 apresenta o método “discretize” responsável por discretizar a base de dados.

```
public Instances discretize(Instances dataSet, String col_Discretize, String
col_NumericToNominal) throws Exception {

    if (!col_Discretize.isEmpty()) {
        String[] opDiscretize = new String[2];
        opDiscretize[0] = "-R";
        opDiscretize[1] = col_Discretize;

        weka.filters.unsupervised.attribute.Discretize mDiscretize = new
weka.filters.unsupervised.attribute.Discretize();
        mDiscretize.setOptions(opDiscretize);
        mDiscretize.setInputFormat(dataSet);
        dataSet = Filter.useFilter(dataSet, mDiscretize);
    }

    if (!col_NumericToNominal.isEmpty()) {
        String[] op_NumericToNominal = new String[2];
        op_NumericToNominal[0] = "-R";
        op_NumericToNominal[1] = col_NumericToNominal;
        NumericToNominal m_NumericToNominal = new NumericToNominal();
        m_NumericToNominal.setOptions(op_NumericToNominal);
        m_NumericToNominal.setInputFormat(dataSet);
        dataSet = Filter.useFilter(dataSet, m_NumericToNominal);
    }
    return dataSet;
}
```

Listagem 5 - DiscretizeController – Método discretize
Fonte: Autoria própria.

A Listagem 6 apresenta o método “base”, responsável por gerar os testes base do sistema, nele são obtidos os classificadores, variáveis e cursos a serem classificados, e é realizado o teste de todos os classificadores com todas as variáveis para todos os cursos, garantindo assim que cada variável seja testada com todos os classificadores.

```
@GetMapping("/base")
public ResponseEntity base() throws Exception {
    List<Classifier> classifiersList = classifierRepository.findByUseClassify(1);
    List<Variable> variaveisList = variableRepository.findByUseClassify(1);
    List<Curso> cursosList = cursoRepository.findByUseClassify(1);

    int period_calculation =
testClassifierRepository.findMaxPeriodCalculationByType(TestClassifier.TEST_BASE);
    weka.classifiers.Classifier[] wekaClassifiers =
classifierController.getWekaClassifiers(classifiersList);

    try {
        for (Curso curso : cursosList) {
            int position_classifier = 0;
            for (weka.classifiers.Classifier wekaClassifier : wekaClassifiers) {
                if (wekaClassifier != null) {
                    for (Variable variable : variaveisList) {
                        List<Variable> variaveis = new ArrayList<>();
                        variaveis.add(variable);
                        Instances dataSet =
dataBaseController.getDataSet(variaveis, curso.getId(),
dataBaseController.TRAINING);
```

```
classifyController.ClassifyTraining(classifiersList.get(position_classifier),
wekaClassifier, dataSet, curso, variaveis, TestClassifier.TEST_BASE,
period_calculation + 1);
                    }
                }
                position_classifier++;
            }
        }
        return new ResponseEntity("Success in making the basic classification!",
HttpStatus.OK);
    } catch (Exception ex) {
        return new ResponseEntity(ex.getMessage(),
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Listagem 6 - TestClassifierController – Método base
Fonte: Autoria própria.

A Listagem 7 apresenta o método “setBestBase”, responsável selecionar os 3 (três) classificadores que obtiveram o maior sucesso em classificar corretamente os dados e depois selecionar as 7 (sete) variáveis que obtiveram o maior sucesso para cada um destes classificadores, estas informações são gravadas em um *Classify*, que é a classe responsável por armazenar as informações de quais classificadores e quais variáveis serão utilizadas para cada curso.

```
@GetMapping("/set-best-base")
public ResponseEntity setBestTestsBase() throws Exception {
    List<Curso> cursosList = cursoRepository.findByUseClassify(1);
    int period_calculation = classifyRepository.findMaxPeriodCalculation();
    try {
        for (Curso curso : cursosList) {
            List<Classificar> Classificars = curso.getClassificar();
            List<Classifier> best_classifiers =
classifierRepository.findTopXClassifiersByCurso(curso.getId(), 3);

            for (Classifier classifier : best_classifiers) {
                Classificar Classificar = new Classificar();
                List<Variable> best_variable_by_classifier =
variableRepository.findTopXVariableByCursoAndClassifier(curso.getId(),
classifier.getId(), 7);
                Classificar.setClassifier(classifier);
                Classificar.setVariable(best_variable_by_classifier);
                Classificar.setPeriodCalculation(period_calculation + 1);
                Classificar = ClassificarRepository.save(Classificar);
                Classificars.add(Classificar);
            }
            curso.setClassificar(Classificars);
            cursoRepository.save(curso);
        }
        return new ResponseEntity("Success in set best base classification!",
HttpStatus.OK);
    } catch (Exception ex) {
        return new ResponseEntity(ex.getMessage(),
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Listagem 7 - TestClassifierController – Método setBestTestsBase
Fonte: Autoria própria.

A Listagem 8 apresenta o método “combinations”, que gera todas as combinações possíveis para uma determinada lista de variáveis sem que se repita elementos no mesmo conjunto, neste sistema foi utilizado uma lista com 7 (sete) variáveis, agrupadas de 1 (um) à 7 (sete) elementos, gerando assim 127 possibilidades de combinações.

```
public List combinations(List<Variable> variaveis, int limit) {
    List<SortedSet<Comparable>> allCombList = new
ArrayList<SortedSet<Comparable>>();
    for (Variable nstatus : variaveis) {
        allCombList.add(new TreeSet<Comparable>(Arrays.asList(nstatus.getId())));
    }
    for (int nivel = 1; nivel < variaveis.size(); nivel++) {
        List<SortedSet<Comparable>> statusAntes = new
ArrayList<SortedSet<Comparable>>(allCombList);

        for (Set<Comparable> antes : statusAntes) {
            SortedSet<Comparable> novo = new TreeSet<Comparable>(antes);
            novo.add(variaveis.get(nivel).getId());
            if (!allCombList.contains(novo) && novo.size() <= limit) {
                allCombList.add(novo);
            }
        }
    }
    Collections.sort(allCombList, new Comparator<SortedSet<Comparable>>() {
        @Override
        public int compare(SortedSet<Comparable> o1, SortedSet<Comparable> o2) {
            int sizeComp = o1.size() - o2.size();
            if (sizeComp == 0) {
                Iterator<Comparable> o1iIterator = o1.iterator();
                Iterator<Comparable> o2iIterator = o2.iterator();
                while (sizeComp == 0 && o1iIterator.hasNext()) {
                    sizeComp = o1iIterator.next().compareTo(o2iIterator.next());
                }
            }
            return sizeComp;
        }
    });
    return allCombList;
}
```

Listagem 8 - TestClassifierController – Método combinations
Fonte: Autoria própria.

A Listagem 9 apresenta o método “classifyBestBase”, responsável por classificar cada um dos 3 (três) classificadores selecionados no “setBestBase”, com todas as combinações possíveis de variáveis, para todos os cursos. Formando assim um conjunto de teste de 127 testes para cada um dos 3 (três) classificadores, totalizando 381 testes por curso.

```
@GetMapping("/classify-best-base")
public ResponseEntity classifyBestBase() throws Exception {
    List<Curso> cursosList = cursoRepository.findByUseClassify(1);
    int period_calculation =
testClassifierRepository.findMaxPeriodCalculationByType(TestClassifier.TEST_PATTERN
);
    try {
        for (Curso curso : cursosList) {
            List<Classify> classifys =
```

```
classifyRepository.findByCursoAndMaxPeriodCalcularion(curso.getId());
            for (Classify classify : classifys) {
                weka.classifiers.Classifier classifier =
classifierController.NewClassifier(classify.getClassifier());
                List combinations =
classifyController.Combinations(classify.getVariable(), 3);
                if (classifier != null) {
                    for (int p = 0; p < combinations.size(); p++) {
                        List<Variable> newVariaveis = new ArrayList<>();
                        String[] s_combinations =
combinations.get(p).toString().replace("[", "").replace("]", "").replace(" ",
"").split(",");
                        for (int t = 0; t < s_combinations.length; t++) {

newVariaveis.add(variableRepository.findOne(Long.parseLong(s_combinations[t])));
                        }
                        Instances dataSet =
dataBaseController.getDataSet(newVariaveis, curso.getId(),
dataBaseController.TRAINING);

classifyController.ClassifyTraining(classify.getClassifier(), classifier, dataSet,
curso, newVariaveis, TestClassifier.TEST_PATTERN, period_calculation + 1);
                    }
                }
            }
        }
        return new ResponseEntity("Success in set best base classification!",
HttpStatus.OK);
    } catch (Exception ex) {
        return new ResponseEntity(ex.getMessage(),
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Listagem 9 - TestClassifierController – Método classifyBestBase
Fonte: Autoria própria.

A Listagem 10 apresenta o método “setPattern”, responsável por selecionar o teste com maior percentual de sucesso obtido no método “classifyBestBase”, após este processo, é gerado novamente a classificação com o intuito de armazenar as probabilidades de evasão para cada um dos alunos que se encontram entre as seguintes situações resumidas: Não Evadido, Evadido e Formado.

```
@GetMapping("/set-pattern")
public ResponseEntity setPattern() throws Exception {
    List<Curso> cursosList = cursoRepository.findByUseClassify(1);
    int period_calculation =
testClassifierRepository.findMaxPeriodCalculationByType(TestClassifier.TEST_PATTERN
);

    try {
        for (Curso curso : cursosList) {
            TestClassifier testClassifier =
testClassifierRepository.findTop1ByCursoAndPeriodCalculationOrderBySuccessDesc(curs
o, period_calculation);
            weka.classifiers.Classifier classifier =
classifierController.NewClassifier(testClassifier.getClassifier());

            Instances dataSetTraining =
dataBaseController.getDataSet(testClassifier.getVariable(), curso.getId(),
dataBaseController.TRAINING);
            List<Student> students = studentRepository.findByCurso(curso.getId());
            classifyController.ClassifyTestAll(testClassifier, classifier,
```

```
dataSetTraining, students);

            Instances dataSetTest =
dataBaseController.getDataSet(testClassifier.getVariable(), curso.getId(),
dataBaseController.TEST);
            List<Student> students_not_evaded =
studentRepository.findByCursoTest(curso.getId());
            classifyController.ClassifyTest(testClassifier, classifier,
dataSetTraining, dataSetTest, students_not_evaded);

            testClassifier.setType(TestClassifier.PATTERN_RESULT);
            testClassifierRepository.save(testClassifier);
        }

        return new ResponseEntity("Success in set best base classification!",
HttpStatus.OK);
    } catch (Exception ex) {
        return new ResponseEntity(ex.getMessage(),
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Listagem 10 - TestClassifierController – Método setPattern
Fonte: Autoria própria.

4.4.2 Sistema Web

As *Migrations* são métodos que permitem a criação e manipulação de bancos de dados, nela são declarados os campos de cada tabela, seu tipo e suas particularidades. Um exemplo é apresentado na Listagem 11, a qual fornece os métodos de criação e exclusão da tabela de usuários do sistema (*user*).

```
class CreateUserTable extends Migration
{
    public function up()
    {
        Schema::create('user', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password');
            $table->integer('status');
            $table->integer('position_id')->unsigned();
            $table->foreign('position_id')->references('id')-
>on('position');
            $table->integer('campus_id')->unsigned()->nullable();
            $table->foreign('campus_id')->references('id')->on('campus');
            $table->rememberToken();
            $table->timestamps();
        });
```

```
    }
    public function down()
    {
        Schema::table('user', function (Blueprint $table) {
            $table->dropForeign('user_position_id_foreign');
        });
        Schema::dropIfExists('user');
    }
}
```

Listagem 11 - Migration CreateUserTable
Fonte: Autoria própria.

Os *Models* são as classes responsáveis por interagir diretamente com o banco de dados, cada tabela de banco de dados tem um "Modelo" correspondente. Os modelos permitem consultar dados em suas tabelas, além de inserir e alterar registros. Um exemplo é apresentado na Listagem 12, aonde é declarado qual o nome da tabela do banco de dados correspondente e seus campos.

```
class User extends Authenticatable
{
    use Notifiable;

    protected $table = 'user';

    protected $fillable = [
        'name', 'email', 'password', 'status', 'position_id', 'campus_id',
    ];
}
```

Listagem 12 - Model User
Fonte: Autoria própria.

O controle de acesso as páginas do sistema é realizado através dos *Middleware's*, que é um recurso que permite interceptar a requisição e analisa-la antes de devolver a resposta ao browser, podendo assim fazer a verificação do nível de acesso do usuário.

```
class Admin
{
    public function handle($request, Closure $next)
    {
        if (!\Auth::check()) {
            return redirect('login');
        } else {
            $loggedUser = \Auth::user();
            if ($loggedUser->position_id == 1 || $loggedUser->position_id == 2) {
                return $next($request);
            } else {
                $request->session()->flash('type', 'danger');
                $request->session()->flash('message', 'Você não tem permissão para
acessar esta área!');
```

```
                if ($loggedUser->position_id == 3) {
                    return redirect('admin/campus');
                } else if ($loggedUser->position_id == 4 || $loggedUser-
>position_id == 5) {
                    return redirect('admin/curso');
                }
            }
        }
    }
}
```

Listagem 13 - Middleware Admin
Fonte: Autoria própria.

A Listagem 13 apresenta o “Middleware Admin”, que é responsável por validar se o usuário tem permissão de acesso as páginas administrativas da instituição, primeiramente é realizado uma verificação se o mesmo está *logado*, caso contrário ele é redirecionado a página de *login*, se estiver *logado* é verificado se seu cargo é de “Desenvolvedor” ou “Administrador da Instituição”, se for o caso ele será redirecionado a página requisitada, se for um “Administrador do Campus” será redirecionado para o *dashboard* Campus, e se for um “Coordenador de Curso” ou “Professor” será redirecionado para o *dashboard* Curso.

O controle de rotas é realizado através das *Route’s*, que são métodos que definem o endereço de acesso as páginas, assim como qual o nível de acesso, e a qual controlador cada rota redireciona.

```
Route::group(['prefix' => '/admin', 'middleware' => 'admin', 'namespace' =>
'admin'], function () {
    Route::get('/position', 'PositionController@index');
    Route::resource('/user', 'UserController');
    Route::resource('/situation', 'SituationController');
});
```

Listagem 14 - Route Admin
Fonte: Autoria própria.

A Listagem 14 apresenta o controle de rotas para as páginas “/admin”, aonde é definido qual controlador é responsável por cada uma das rotas, a rota “/admin/position” pelo tipo *get* chama o método *index* do controlador *PositionController*, já a rota “/admin/user” pelo tipo *resource* identifica cada um dos tipos de acessos no controlador *UserController*, sendo eles: *index*, *create*, *store*, *show*, *edit*, *update* e *destroy*.

O sistema web possui vários métodos padrão para as funcionalidades de cadastros, como o método *index* apresentado na Listagem 15. Este método é utilizado para apresentar a listagem dos usuários.

```
public function index(Request $request)
{
    try {
        $objects = User::all();
        return view('admin.user.index', compact([
            'objects', $objects,
        ]));
    } catch (Exception $e) {
        $request->session()->flash('type', 'danger');
        $request->session()->flash('message', 'Ocorreu um erro no sistema.');
        return redirect('/');
    }
}
```

Listagem 15 - UserController – Método index
Fonte: Autoria própria.

A Listagem 16 apresenta o método *create*, responsável por carregar a lista de cargos (*cargos*), cursos (*cursos*) e câmpus (*campus*), e apresenta a tela de criação de um novo usuário.

```
public function create(Request $request)
{
    try {
        $positions = Position::all();
        $cursos = Curso::all();
        $campus = Campus::all();
        return view('admin.user.create', compact([
            'positions', $positions,
            'cursos', $cursos,
            'campus', $campus,
        ]));
    } catch (Exception $e) {
        $request->session()->flash('type', 'danger');
        $request->session()->flash('message', 'Ocorreu um erro no sistema.');
        return redirect('/');
    }
}
```

Listagem 16 - UserController – Método create
Fonte: Autoria própria.

A Listagem 17 apresenta o método *store*, responsável por salvar os registros do usuário no banco de dados, nele é recebido como parâmetro um *Request* aonde é injetado os dados fornecidos no cadastro, primeiramente é feita uma validação para verificar se os dados estão corretos, caso eles sejam válidos serão salvos no banco de dados.

```
public function store(Request $request)
{
    $validator = Validator::make($request->all(), [
        'name' => 'required|min:6|max:191',
        'email' => 'required|email|unique:user|max:191',
```

```
        'position_id' => 'required',
        'campus_id' => 'required',
        'cursos' => 'required',
    ], $this->messages);

    if ($validator->fails()) {
        return redirect()->back()->withInput()->withErrors($validator);
    } else {
        try {
            $object = User::create([
                'name' => $request->name,
                'email' => $request->email,
                'password' => bcrypt($request->password),
                'status' => 1,
                'position_id' => $request->position_id,
                'campus_id' => $request->campus_id,
            ]);

            foreach ($request->get('cursos') as $id) {
                $curso = Curso::find($id);
                if ($curso->campus_id == $object->campus_id) {
                    $object->curso()->attach($curso);
                }
            }

            $request->session()->flash('type', 'success');
            $request->session()->flash('message', ucfirst('Usuário criado com
sucesso!'));
            return redirect('admin/user');
        } catch (Exception $e) {
            $request->session()->flash('type', 'danger');
            $request->session()->flash('message', ucfirst('Erro ao criar
usuário.'));
            return redirect('admin/user');
        }
    }
}
```

Listagem 17 - UserController – Método store
Fonte: Autoria própria.

A Listagem 18 apresenta o método *show*, nele é recebido como parâmetro um *Request* e um id do usuário, o qual é utilizado para buscar o usuário na base de dados e apresentar as informações do mesmo.

```
public function show(Request $request, $id)
{
    try {
        try {
            $object = User::findOrFail($id);
        } catch (Exception $e) {
            $request->session()->flash('type', 'danger');
            $request->session()->flash('message', 'Este usuário não existe!');
            return redirect('admin/user');
        }
        return view('admin.user.show', compact('object', $object));
    } catch (Exception $e) {
        $request->session()->flash('type', 'danger');
        $request->session()->flash('message', 'Ocorreu um erro no sistema.');
        return redirect('admin/user');
    }
}
```

Listagem 18 - UserController – Método show
Fonte: Autoria própria.

A Listagem 19 apresenta o método *edit*, que recebe como parâmetro um *Request* e um id do usuário, o qual é utilizado para buscar o usuário na base de dados e apresentar as informações deste no formulário, aonde é possível realizar alterações e salvar posteriormente.

```
public function edit(Request $request, $id)
{
    try {
        try {
            $object = User::findOrFail($id);
        } catch (Exception $e) {
            $request->session()->flash('type', 'danger');
            $request->session()->flash('message', 'Este usuário não existe!');
            return redirect('admin/user');
        }
        $positions = Position::all();
        $cursos = Curso::all();
        $campus = Campus::all();

        return view('admin.user.edit', compact([
            'object', $object,
            'positions', $positions,
            'cursos', $cursos,
            'campus', $campus,
        ]));

    } catch (Exception $e) {
        $request->session()->flash('type', 'danger');
        $request->session()->flash('message', 'Ocorreu um erro no sistema!');
        return redirect('admin/user');
    }
}
```

Listagem 19 - UserController – Método show
Fonte: Autoria própria.

A Listagem 20 apresenta o método *update*, nele é recebido como parâmetro um *Request* aonde é injetado os dados fornecidos no cadastro e um id do usuário, o qual é utilizado para buscar o usuário na base de dados e realizar a substituição das informações originais pelas novas informações fornecidas pelo usuário.

```
public function update(Request $request, $id)
{
    $validator = Validator::make($request->all(), [
        'name' => 'required|min:6|max:191',
        'email' => 'required|email|max:191',
        'position_id' => 'required',
        'campus_id' => 'required',
        'cursos' => 'required',
    ], $this->messages);

    if ($validator->fails()) {
        return redirect()->back()->withInput()->withErrors($validator);
    } else {
        try {
            try {
                $object = User::findOrFail($id);
            } catch (Exception $e) {
                $request->session()->flash('type', 'danger');
                $request->session()->flash('message', 'Este usuário  não existe!');
                return redirect('admin/user');
```

```
            }

            $object->name = $request->name;
            $object->email = $request->email;
            $object->position_id = $request->position_id;
            $object->status = $request->status;
            $object->campus_id = $request->campus_id;
            $object->save();
            $object->curso()->detach();

            foreach ($request->get('cursos') as $id) {
                $curso = Curso::find($id);
                if ($curso->campus_id == $object->campus_id) {
                    $object->curso()->attach($curso);
                }
            }

            $request->session()->flash('type', 'success');
            $request->session()->flash('message', ucfirst('Usuário atualizado com
sucesso!'));
            return redirect('admin/user');

        } catch (Exception $e) {
            $request->session()->flash('type', 'danger');
            $request->session()->flash('message', ucfirst('Erro ao atualizar
usuário.'));
            return redirect('admin/user');
        }
    }
}
```

Listagem 20 - UserController – Método update
Fonte: Autoria própria.

A Listagem 21 apresenta o método *destroy*, nele é recebido como parâmetro um *Request* aonde é injetado a variável *var_delete* que é o id do usuário selecionado a ser deletado, inicialmente é feito a busca por este usuário na base de dados, excluído sua ligação com os cursos e por fim sua exclusão do banco.

```
public function destroy(Request $request)
{
    try {
        try {
            $object = User::findOrFail($request->var_delete);
        } catch (Exception $e) {
            $request->session()->flash('type', 'danger');
            $request->session()->flash('message', 'Este usuário não existe!');
            return redirect('admin/user');
        }
        $object->curso()->detach();
        $object->delete();

        $request->session()->flash('type', 'success');
        $request->session()->flash('message', ucfirst('Usuário deletado com
sucesso!'));
        return redirect('admin/user');
    } catch (Exception $e) {
        $request->session()->flash('type', 'danger');
        $request->session()->flash('message', ucfirst('Erro ao deletar usuário.'));
        return redirect('admin/user');
    }
}
```

Listagem 21 - UserController – Método destroy
Fonte: Autoria própria.

A Listagem 22 apresenta o método *upload* que foi desenvolvido em *javascript*, este método é responsável por receber o arquivo ".xlsx" e converter o mesmo para o formato *json*, que será enviado para o controlador *UploadController*.

```
function upload(position){
   try {
      var rABS = true;
      var f = array_files[position];

      var reader = new FileReader();
      reader.onload = function(e) {
         var data = e.target.result;
         if(!rABS) data = new Uint8Array(data);

         var workbook = XLSX.read(data, {
            type: rABS ? 'binary' : 'array',
            cellDates: true,
            dateNF:"yyyy-mm-dd"});

         var worksheets = [];
         for (var i = 0; i < workbook.SheetNames.length; ++i) {
            var worksheet = workbook.Sheets[workbook.SheetNames[i]];
            var cell = worksheet[XLSX.utils.encode_cell({c: 0, r: 0})];

            if(cell == undefined){
               worksheet_is_empty = true;
               console.log("Arquivo em branco!");
            }else if(cell.v != "Câmpus"){
               worksheet_is_not_correct_pattern = true;
               console.log("Planilha fora dos padrões!");
            }else{
               worksheet_sucess = true;
               var dados_json =
XLSX.utils.sheet_to_json(worksheet,{'date_format':'yyyy-mm-dd'});
               worksheets.push(dados_json);
            }
         }

         if(worksheets.length){
            worksheets.forEach(function (item) {
                     send_data(item, position);
            });
         }else{
            if(worksheet_is_empty && !worksheet_is_not_correct_pattern){
               array[position].message = "Arquivo em branco.";
               array[position].state = 1;
               updateTable();
               mApp.unblockPage();
            }else{
               array[position].message = "Arquivo fora do padrão.";
               array[position].state = 1;
               updateTable();
               mApp.unblockPage();
            }
         }
      };
      if(rABS) reader.readAsBinaryString(f); else reader.readAsArrayBuffer(f);
   }catch(err) {
      array[position].state = 1;
      updateTable();
      console.log(err.message);
   }
}
```

Listagem 22 - index.blade.php – Método upload

Fonte: Autoria própria.

A Listagem 23 apresenta o método *upload*, responsável por receber os dados no formato *json* e realizar a inclusão dos mesmos na base de dados, vale ressaltar que antes de realizar a inclusão dos dados é verificado se o mesmo já existe, evitando assim que existam registros duplicados.

```
public function upload(Request $request)
{
            try {
                $student = $this->student_repository->getStudentsBase(
                    array(0 => "student.id"),
                    array(
                        array(0 => "student.name", 1 => " = ", 2 => "'" .
$mydata["Nome"] . "'"),
                        array(0 => "student.code", 1 => " = ", 2 =>
$mydata["Código"]),
                        array(0 => "student.curso_id", 1 => " = ", 2 => $curso),
                        array(0 => "student.year_ingress", 1 => " = ", 2 =>
$mydata["Ano de ingresso"]),
                        array(0 => "student.semester_ingress", 1 => " = ", 2 =>
$mydata["Semestre de ingresso no curso"])
                    ),
                    array(0 => "student.id")
                );

                if ($student == null) {
                    $dataNascimento = $mydata["Data de nascimento"];
                    $dataIngresso = $mydata["Data de ingresso"];
...
                    $student = Student::create([
                        'name' => $mydata["Nome"],
                        'code' => $mydata["Código"],
                        'year_ingress' => $mydata["Ano de ingresso"],
                        'genre' => $mydata["Gênero"],
                        'age' => $idadeIngresso,
                        'birth_date' => $dataNascimento,
                        'type_ingress' => $mydata["Forma de ingresso"],
                        'changed_curso' => $mydata["Mudou de curso - mesmo
câmpus"],
                        'changed_curso_campus' => $mydata["Mudou de curso - outro
câmpus"],
                        'municipality' => $mydata["Município"],
                        'municipality_sisu' => $mydata["Município SISU"],
                        'email' => $mydata["E-mail"],
                        'enem_human' => $enemHumanas,
                        'enem_language' => $enemLinguagem,
                        'enem_math' => $enemMatematica,
                        'enem_nature' => $enemNatureza,
                        'enem_redaction' => $enemRedacao,
                        'sisu' => $notaFinalSISU,
                        'entries_other_curso' => $mydata["Número de entradas em
outros cursos"],
                        'entries_curso' => $mydata["Número de entradas no curso"],
                        'country' => $mydata["País de nascimento"],
                        'semester_ingress' => $mydata["Semestre de ingresso no
curso"],
                        'quota' => $mydata["Tipo de cota"],
                        'state' => $mydata["UF"],
                        'state_sisu' => $mydata["UF SISU"],
                        'curso_id' => $curso,
                    ]);
                    $student = $student->id;
                    $count_students = $count_students + 1;
```

```
                } else {
                    $student = $student[0]->id;
                }
            } catch (Exception $e) {
                return "Erro ao criar/buscar o Aluno!";
            }
...
        $result = [
            "campus" => $count_campus,
            "cursos" => $count_cursos,
            "students" => $count_students,
            "details" => $count_details,
            "amount_data" => $amount_data
        ];

        return $result;
    } catch (Exception $e) {
        return $e;
    }
}
```

Listagem 23 - UploadController – Método upload
Fonte: Autoria própria.

5 CONCLUSÃO

Este trabalho propôs e apresentou a modelagem e implementação de um SAD que inclui características de *Business Intelligence* visando fornecer uma visualização global e sobre uma base numericamente precisa de informações relativas à estimativa da tendência de evasão de estudantes do ensino superior, a ser baseada em cálculo de probabilidades e algoritmos de classificação.

A identificação da estimativa da tendência de evasão através do uso de técnicas de mineração de dados mostrou-se viável. Este trabalho avaliou a técnica através do uso de três algoritmos de classificação distintos, sobre uma base de dados composta por 10.371 (dez mil trezentos e setenta e um) registros de alunos de dez diferentes cursos da UTFPR. Os resultados foram uma margem de sucesso na previsão da tendência de evasão do aluno variando de 92,34% no mínimo e 99,32% no máximo, e com uma média global de 95,81% entre todos os cursos analisados.

O desempenho obtido pelos algoritmos de mineração de dados dos mais simples aos mais sofisticados foram semelhantes. Todos os cursos foram analisados pelos três classificadores, com todas as combinações possíveis das sete variáveis que obtiveram melhor resultado individualmente, estas combinações foram agrupadas de um a sete elementos totalizando 381 (trezentos e oitenta e um) testes por curso. O algoritmo de classificação que se mostrou mais eficiente foi LWL, pois foi o que obteve a maior taxa de acerto em seis dos dez cursos analisados, seguido pelo Naive Bayes que obteve a maior taxa de acerto em três, e por último o J48 que obteve a maior taxa de acerto em apenas um curso.

Além da acurácia, a taxa de erro dos classificadores foi considerada na análise, portanto, a previsão incorreta de alunos com risco de evasão é considerada erro grave do classificador. No entanto, a acurácia dos classificadores e a taxa de erro são fortemente influenciadas pelos vieses da base de dados, isto é, alunos que evadem do curso mesmo com rendimento acadêmico alto e alunos que concluem o curso com rendimento acadêmico abaixo da média, estes casos estão fora do padrão das classes aprendidas pelos classificadores

O presente estudo ainda está em fase inicial, no entanto, já indica que a identificação da estimativa da tendência de evasão pode ser feita a partir de um número reduzido de variáveis, por exemplo, verificou-se que variável mais importante

é o número de disciplinas aprovadas, pois está no melhor resultado de todos os cursos, seguida pelo coeficiente de rendimento, idade, disciplinas reprovadas por nota e entre outras.

Entre os benefícios potenciais da aplicação da mineração de dados neste contexto podemos citar o auxílio na identificação dos acadêmicos com maior probabilidade de evadirem dos respectivos cursos, possibilitando assim uma melhor tomada de decisão no combate da evasão, assim como possibilitar a redução de perdas com alunos evadidos e evitar a má destinação de recursos financeiros.

Como trabalhos futuros, consideramos aplicar procedimentos semelhantes para outros cursos da universidade e de outras instituições de ensino privadas, verificando se os resultados até agora observados se repetem para outros subconjuntos de alunos de graduação. Sugere-se também uma abordagem do problema considerando não apenas as informações oriundas do sistema acadêmico, mas também fatores de caráter socioeconômico, assim como utilizar as informações dos outros semestres de forma individual ou agrupadas, identificando por meio de ferramentas estatísticas alunos que possam estar caminhando para a evasão, pela análise das variáveis colocadas, agindo assim preventivamente.

REFERÊNCIAS

AMCHART. **JavaSrcript AmChats**. Disponível no site da amCharts (2018). Disponível em: < https://www.amcharts.com/javascript-charts/ > Acesso em: 07 de junho de 2018.

BARBOSA, João Paulo Gomes et al. A adoção do SiSU e a evasão na Universidade Federal de Uberlândia. **Revista Ibero-Americana de Estudos em Educação**, v. 12, n. 2, p. 722-738, 2017.

BERRY, Michael J.A.; LINOFF, Gordon S. **Data Mining Tehniques – for marketing, sales, and customer support**. United States: Wiley Computer Publishing, 1997.

BERSON, Alez; SMITH, Stephen J., **Data Warehousing, Data Mining & OLAP**. McGraw-Hill, Estados Unidos, 1997. Disponível em: <http://www.tdan.com/book005.htm>. Acesso em: 10 nov. 2016.

BRONDANI, Camila Hubner; AREND, Cesar Frantz; SOUZA, Darciele Aparecidae Zilio de; PIRES, José Carlos Puiati. **Guia prático de utilização da ferramenta Astah Community 6.1**. Disponível em: <http://docslide.com.br/documents/astah-559c10075fd2f.html>. Acesso em: 10 nov. 2016.

BROWNLEE, Jason. **Lazy and Competitive Learning.** Melbourne, 2007.

CHAVES Eduardo; FALSARELLA, Orandi M. **Sistemas de Informação e Sistemas de Apoio à Decisão**. Revista do Instituto de Informática PUCCAMP. Campinas, SP, v. 3, n. 1, 1995.

COBRA, M.; BRAGA, R. **Marketing educacional:** ferramentas de gestão para Instituições de ensino. São Paulo / Espírito Santo: Cobra / Hoper, 2004. 148p.

DATATABLES. **Manual**. Disponível no site da DataTables (2018). Disponível em: < https://datatables.net/manual/index > Acesso em 07 de junho de 2018.

DEVMEDIA. **Laravel tutorial**. Disponível no site da DevMedia (2015). Disponível em <https://www.devmedia.com.br/laravel-tutorial/33173> Acesso em 03 de Maio de 2018.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern Classification**. 2. ed. [S.l.]: Wiley-Interscience, 2001.

FabFORCE.net. **Fabulous Force Database Tools.** Disponível em: <http://www.fabforce.net/dbdesigner4/index.php>. Acesso em: 10 nov. 2016.

FAYYAD, Usama M. et al. **Advances in knowledge discovery and data mining**. Menlo Park, Calif.: AAAI Press: MIT Press; c1996; 611p.

SILVA FILHO, Roberto Leal Lobo; MOTEJUNAS, Paulo Roberto; HIPÓLITO, Oscar; MELO LOBO, Maria Beatriz de Carvalho. **A evasão no ensino superior brasileiro:**

instituto lobo para o desenvolvimento da educação, da ciência e da tecnologia. **Cadernos de Pesquisa,** v. 37, n. 132, 2007.

GARCIA, Salvador et al. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. **IEEE Trans. on Knowledge and Data Engineering**, v. 25, n. 4, p. 734-750, 2013.

GARCIA, Mauricio. **Gestão profissional em instituições privadas de educação superior** – Um Guia de sobrevivência para mantenedores, acionistas, reitores, pró-reitores, diretores, coordenadores, gerentes e outros gestores institucionais. 1 ed. São Paulo: Hoper, 2006.

GOEBEL, Michael; GRUENWALD, L. **A survey of data mining and knowlodge Discovery software tools**. SIGKDD Explorations, v. 1, p. 20-33, 1999.

HAN, Jiawei; KAMBER, Micheline. **Data Mining:** Concepts and Techniques. 2ª ed. San Francisco: Morgan Kaufmann, 2006.

INEP. **Censo da Educação Superior no Brasil 2016: notas estatísticas**. Disponível em: <http://download.inep.gov.br/educacao_superior/censo_superior/documentos/2016/notas_sobre_o_censo_da_educacao_superior_2016.pdf>. Acesso em: 11 jun. 2018.

INEP**. Sinopses Estatísticas da Educação Superior – Graduação (2014)**. Disponível em: <http://portal.inep.gov.br/superior-censosuperior-sinopse>. Acesso em: 10 nov. 2016.

JUNGHANS, Daniel**. O fracasso escolar no ensino de engenharia na UTFPR -** análise do desempenho escolar em um ambiente de ensino público tecnológico. 2014. 114 f. Dissertação (Mestrado em Educação). Universidad Del Mar, Viña del Mar, 2014.

LIU, Huan et al. Discretization: An enabling technique. **Data mining and knowledge discovery**, v. 6, n. 4, p. 393–423, outubro 2002.

LOPES, Lilá Reis. **O Marketing nas IES privadas da Bahia:** um estudo sobre o nível de conhecimento e potencialidades de uso do marketing, e sobre as aspirações e necessidades dos estudantes candidatos. 2006. 172 f. Dissertação (Mestrado em Administração). Universidade Federal da Bahia, Salvador, 2006.

MANHÃES, Laci Mary Barbosa et al. **Previsão de Estudantes com Risco de Evasão Utilizando Técnicas de Mineração de Dados**. In: XXII Simpósio Brasileiro de Informática na Educação. Anais do XXII SBIE – XVII WIE. Aracaju, 21 a 25 de novembro de 2011.

MASTERTECH. **O que é Java e o que posso fazer com isso?**. Disponível no site da MasterTech (2018). Disponível em: < https://blog.mastertech.tech/tecnologia/o-que-e-java-e-o-que-posso-fazer-com-isso/ > Acesso em: 07 de junho de 2018

MARTINS, Cledis. **Evasão De Alunos Nos Cursos De Graduação Em Uma Instituição De Ensino Superior**. Fundação Pedro Leopoldo, Pedro Leopoldo, 2007.

MONTGOMERY, Douglas C; RUNGER, George C. **Estatística aplicada e probabilidade para engenheiros**. Rio de Janeiro, 2006.

NETBEANS. **O que é o NetBeans**. Disponível em: <http://www.netbeans.org/index_pt_BR.html>. Acesso em: 10 nov. 2016.

NUNES, A. C. O. FADA. **Ferramenta de Apoio à Decisão Acadêmica. In: XIII Congresso Internacional de Educação a Distância**, 2007, Curitiba. Anais do XIII Congresso... Curitiba: ABED, abr. 2007, p. 1-11. Disponível em: <http://www.abed.org.br/congresso2007/tc/542007105921PM.pdf>. Acesso em: 11 jun. 2018.

PRESSMAN, Roger. **Engenharia de software.** 5ª ed., Rio de Janeiro: McGraw-Hill, 2002.

RAMOS, Patrícia Gouveia. **Uma Investigação das Redes NeuroFuzzy aplicadas à Mineração de Dados**. Recife, 1999. Dissertação de Graduação (Graduação em Ciência da Computação) – Universidade Federal de Recife, Recife, 1999. Disponível em: <http://www.di.ufpe.br/~tg/1999-1/pgr.doc>. Acesso em: 10 nov. 2016.

SILVA, Dhiogo Cardoso da. **Uma arquitetura de business intelligence para processamento analítico baseado em tecnologias semânticas e em linguagem natural**. 2011. Dissertação (Mestrado) – Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento, Florianópolis, 2011.

SOUZA, Solange Lima de. **Evasão no ensino superior**: um estudo utilizando a mineração de dados como ferramenta de gestão do conhecimento em um banco de dados referente à graduação de engenharia. 2008. 107 p. Dissertação (Mestrado em Engenharia Civil). Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2008.

SPRAGUE, Ralph H.; WATSON, Hugh J., **Sistemas de Apoio à Decisão**. Rio de Janeiro: Campus, 1991.

STEINER, Maria Teresinha Arns et al. **Abordagem de um problema médico por meio do processo de KDD com ênfase à análise exploratória dos dados**. **Gest. Prod.** [online] v. 13, n. 2, p. 325-337, 2006.

TECHTUDO. **O que é e como usar MySQL?**. Disponível no site do techtudo (2012). Disponível em < http://www.techtudo.com.br/artigos/noticia/2012/04/o-que-e-e-como-usar-o-mysql.html > Acesso em: 11 jun. 2018.

UNIVERSITY OF WAIKATO. **Weka 3 – Machine Learning Software in Java**. Disponível no site da University of Waikato (2010). Disponível em: <http://www.cs.waikato.ac.nz/ml/weka>. Acesso em: 10 nov. 2016.

WALTER, Amanda Moura; ABBAD, Gardênia da Silva. **Variáveis preditoras de evasão em dois cursos a distância**. In: XXXII Encontro da ANPAD. **Anais...** Rio de Janeiro, 6 a 10 de setembro de 2008.

WEBB, Geoffrey I; SAMMUT, Claude. **Encyclopedia of Machine Learning**. Nova York, 2011.

WERLANG, Jorge Daniel. **Fatores extrínsecos e intrínsecos que motivam os discentes na escolha e na permanência no curso de Ciências Contábeis da Universidade Federal do Rio Grande do Sul**. 2014. 30 f. Monografia (Graduação em Ciências Contábeis). Universidade Federal do Rio Grande do Sul, Porto Alegre, 2014.