

UNIVERSIDADE DE CAXIAS DO SUL  
ÁREA DE CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS  
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ADRIANO MARGARIN

**Evolução da Ferramenta de  
Gerenciamento do Portal de  
Algoritmos da UCS**

Alexandre Erasmo Krohn Nascimento  
Orientador

Ricardo Vargas Dorneles  
Coorientador

Caxias do Sul, Julho de 2018

# **Evolução da Ferramenta de Gerenciamento do Portal de Algoritmos da Universidade de Caxias do Sul**

por

Adriano Margarin

Trabalho de Conclusão de Curso submetido ao curso de Bacharelado em Sistemas de Informação da Área de Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul, como requisito obrigatório para graduação.

## **Trabalho de Conclusão de Curso**

Orientador: Alexandre Erasmo Krohn Nascimento

Coorientador: Ricardo Vargas Dorneles

Banca examinadora:

Ricardo Vargas Dorneles

ACCEE/UCS

André Luis Martinotto

ACCEE/UCS

Trabalho de Conclusão de Curso apresentado em  
26 de Junho de 2018

André Gustavo Adami  
Coordenador

*"A vida não é fácil e ninguém disse que seria."*  
HENRIQUE BASTOS

## **AGRADECIMENTOS**

Agradeço a minha esposa, Marciele Luis, meus pais, Neusa Maria Margarin e Mauri Augusto Margarin, meus irmãos, André Augusto Margarin e Letícia Margarin pelo apoio no caminho trilhado até aqui.

A todos vocês, minha sincera gratidão.

Adriano Margarin

# SUMÁRIO

<b>LISTA DE ACRÔNIMOS . . . . .</b>	<b>7</b>
<b>LISTA DE FIGURAS . . . . .</b>	<b>8</b>
<b>LISTA DE TABELAS . . . . .</b>	<b>11</b>
<b>RESUMO . . . . .</b>	<b>12</b>
<b>1 INTRODUÇÃO . . . . .</b>	<b>13</b>
<b>2 EVOLUÇÃO DE SOFTWARE . . . . .</b>	<b>16</b>
2.1 Reengenharia de Software . . . . .	17
2.2 Engenharia Reversa . . . . .	18
2.3 Engenharia de Software . . . . .	19
2.4 Processo de Software . . . . .	19
2.5 Engenharia de Requisitos . . . . .	20
2.6 Metodologia ICONIX . . . . .	21
2.6.1 Diagramas de Classe . . . . .	21
2.6.2 Diagrama de Sequência . . . . .	22
2.6.3 Diagrama de Robustez . . . . .	23
2.6.4 Casos de Usos . . . . .	23
2.7 Modelos de Domínio . . . . .	25
2.8 Projeto de Arquitetura . . . . .	25
2.9 Usabilidade . . . . .	25
2.10 Tecnologias . . . . .	26
2.10.1 Java EE . . . . .	26
2.10.2 WildFly . . . . .	27
2.10.3 DAO . . . . .	27
2.10.4 REST . . . . .	27
2.10.5 AngularJS . . . . .	27

<b>2.11 Ambiente Virtual de Aprendizagem . . . . .</b>	<b>29</b>
<b>3 AVALIAÇÃO DO SOFTWARE DO PORTAL DE ALGORITMOS DA UCS . . . . .</b>	<b>31</b>
<b>3.1 Diagrama de Classe de Domínio . . . . .</b>	<b>31</b>
<b>3.2 Interfaces Gráficas . . . . .</b>	<b>34</b>
<b>3.2.1 Cadastro de Aluno . . . . .</b>	<b>34</b>
<b>3.2.2 Criação de Solução de Problemas . . . . .</b>	<b>34</b>
<b>3.2.3 Gerenciamento de Alunos . . . . .</b>	<b>36</b>
<b>3.2.4 Gerenciamento de Problemas . . . . .</b>	<b>37</b>
<b>3.2.5 Edição de Palavra-Chave . . . . .</b>	<b>38</b>
<b>3.2.6 Solução de Problemas . . . . .</b>	<b>39</b>
<b>4 PROPOSTA DE SOLUÇÃO . . . . .</b>	<b>41</b>
<b>4.1 Diagrama de Classe de Domínio . . . . .</b>	<b>42</b>
<b>4.2 Requisitos de projeto . . . . .</b>	<b>44</b>
<b>4.2.1 Requisitos Funcionais . . . . .</b>	<b>44</b>
<b>4.2.2 Requisitos Não-Funcionais . . . . .</b>	<b>45</b>
<b>4.3 Casos de Uso . . . . .</b>	<b>45</b>
<b>4.3.1 Descrição dos Casos de Uso . . . . .</b>	<b>48</b>
<b>4.4 Interfaces Gráficas . . . . .</b>	<b>66</b>
<b>4.4.1 Cadastro de Aluno . . . . .</b>	<b>67</b>
<b>4.4.2 Autenticação . . . . .</b>	<b>68</b>
<b>4.4.3 Boas Vindas ao Administrador e Professor . . . . .</b>	<b>69</b>
<b>4.4.4 Problemas . . . . .</b>	<b>70</b>
<b>4.4.5 Tipo de Problema . . . . .</b>	<b>71</b>
<b>4.4.6 Cadastro e Edição de Problema . . . . .</b>	<b>72</b>
<b>4.4.7 Palavras-chave . . . . .</b>	<b>73</b>
<b>4.4.8 Dados de Testes . . . . .</b>	<b>74</b>
<b>4.4.9 Visualização de Solução de Problema . . . . .</b>	<b>75</b>
<b>4.4.10 Lista de Grupos . . . . .</b>	<b>76</b>
<b>4.4.11 Cadastro e Edição de Grupo . . . . .</b>	<b>77</b>
<b>4.4.12 Adicionar Participantes . . . . .</b>	<b>78</b>
<b>4.4.13 Adicionar Lista de Problemas . . . . .</b>	<b>79</b>
<b>4.4.14 Lista de Alunos . . . . .</b>	<b>80</b>
<b>4.4.15 Cadastro de Alunos . . . . .</b>	<b>81</b>
<b>4.4.16 Listas de Problemas . . . . .</b>	<b>82</b>
<b>4.4.17 Cadastro e Edição de Lista de Problemas . . . . .</b>	<b>83</b>
<b>4.4.18 Lista de Usuários . . . . .</b>	<b>84</b>

4.4.19	Cadastro e Edição de Usuários . . . . .	85
4.4.20	Lista de Professores . . . . .	86
4.4.21	Cadastro e Edição de Professores . . . . .	87
4.4.22	Lista de Instituições . . . . .	88
4.4.23	Cadastro e Edição de Instituições . . . . .	89
4.4.24	Lista de Permissões . . . . .	90
4.4.25	Cadastro e Edição de Permissões . . . . .	91
4.4.26	Lista de Grupos de Administradores . . . . .	92
4.4.27	Cadastro e Edição de Grupos de Administradores . . . . .	93
4.4.28	Lista de Países, Estados e Cidades . . . . .	94
4.4.29	Cadastro e Edição de País . . . . .	95
4.4.30	Cadastro e Edição de Estado . . . . .	96
4.4.31	Cadastro e Edição de Cidade . . . . .	97
<b>4.5</b>	<b>Diagramas de Robustez . . . . .</b>	<b>98</b>
4.5.1	Cadastro de Aluno . . . . .	98
4.5.2	Autenticação . . . . .	99
4.5.3	Boas Vindas do Administrador e do Professor . . . . .	100
4.5.4	Problemas . . . . .	100
4.5.5	Tipo de Problema . . . . .	101
4.5.6	Cadastro e Edição de Problema . . . . .	101
4.5.7	Palavras-chave . . . . .	102
4.5.8	Dados de Testes . . . . .	102
4.5.9	Visualização de Solução partindo da listagem de Problemas . . . . .	103
4.5.10	Lista de Grupos . . . . .	103
4.5.11	Cadastro e Edição de Grupo . . . . .	104
4.5.12	Adicionar Participantes . . . . .	104
4.5.13	Adicionar Lista de Problemas . . . . .	105
4.5.14	Lista de Alunos . . . . .	106
4.5.15	Cadastro de Alunos pelo Administrador . . . . .	107
4.5.16	Lista de Problemas . . . . .	108
4.5.17	Cadastro e Edição de Lista de Problemas . . . . .	109
4.5.18	Lista de Usuários . . . . .	110
4.5.19	Cadastro e Edição de Usuários . . . . .	110
4.5.20	Lista de Professores . . . . .	111
4.5.21	Cadastro e Edição de Professores . . . . .	111
4.5.22	Lista de Instituições . . . . .	112
4.5.23	Cadastro e Edição de Instituições . . . . .	112
4.5.24	Lista de Permissões . . . . .	113
4.5.25	Cadastro e Edição de Permissões . . . . .	113

4.5.26	Lista de Grupos de Administradores . . . . .	114
4.5.27	Cadastro e Edição de Grupos de Administradores . . . . .	114
4.5.28	Lista de Países, Estados e Cidades . . . . .	115
4.5.29	Cadastro e Edição de País . . . . .	115
4.5.30	Cadastro e Edição de Estado . . . . .	116
4.5.31	Cadastro e Edição de Cidade . . . . .	116
<b>4.6</b>	<b>Diagramas de Sequência</b> . . . . .	117
<b>4.7</b>	<b>Arquitetura do Software</b> . . . . .	118
<b>5</b>	<b>CONSIDERAÇÕES FINAIS</b> . . . . .	120
<b>5.1</b>	<b>Trabalhos Futuros</b> . . . . .	121
<b>REFERÊNCIAS</b> . . . . .		122
<b>APÊNDICE A - SEGURANÇA</b> . . . . .		124
<b>APÊNDICE B - DOCUMENTAÇÃO</b> . . . . .		126
<b>APÊNDICE C - SQL DE CRIAÇÃO DO BANCO DE DADOS</b> . . . . .		176
<b>APÊNDICE D - MIGRAÇÃO BANCO DE DADOS</b> . . . . .		211
<b>APÊNDICE E - PROJETOS</b> . . . . .		214

## LISTA DE ACRÔNIMOS

<b>API</b>	<i>Application Programming Interface</i>
<b>AVA</b>	<i>Ambiente Virtual de Aprendizagem</i>
<b>DAO</b>	<i>Data Access Object</i>
<b>HTML</b>	<i>Hyper Text Markup Language</i>
<b>HTTP</b>	<i>Hyper Text Transfer Protocol</i>
<b>HTTPS</b>	<i>Hyper Text Transfer Protocol Secure</i>
<b>SSL</b>	<i>Secure Socket Layer</i>
<b>IHC</b>	Interação humano-computador
<b>JAVA EE</b>	<i>Java Enterprise Edition</i>
<b>MVC</b>	<i>Model-View-Controller</i>
<b>NPAPI</b>	<i>Netscape Plugin Application Programming Interface</i>
<b>ORM</b>	<i>Object-relational mapping</i>
<b>REST</b>	<i>Representational State Transfer</i>
<b>SQL</b>	<i>Structured Query Language</i>
<b>UCS</b>	Universidade de Caxias do Sul
<b>UML</b>	<i>Unified Modeling Language</i>
<b>TCP</b>	<i>Transmission Control Protocol</i>
<b>CSS</b>	<i>Cascading Style Sheets</i>
<b>JDBC</b>	<i>Java Database Connectivity</i>
<b>IDE</b>	<i>Integrated Development Environment</i>

## **LISTA DE FIGURAS**

Figura 2.1: Processo de Reengenharia . . . . .	18
Figura 2.2: Camadas da Engenharia de Software . . . . .	19
Figura 2.3: Diagramas de Classe . . . . .	21
Figura 2.4: Diagrama de Sequência . . . . .	22
Figura 2.5: Diagrama e Robustez . . . . .	23
Figura 2.6: Casos de uso . . . . .	24
Figura 2.7: Casos de uso de transferência de dados . . . . .	25
Figura 2.8: Interação entre AngularJS e Arquitetura . . . . .	29
Figura 3.1: Diagrama de Domínio do Portal de Algoritmos Atual . . . . .	32
Figura 3.2: Cadastro de Aluno . . . . .	34
Figura 3.3: Criação de Solução de Problemas . . . . .	35
Figura 3.4: Gerência de Alunos . . . . .	36
Figura 3.5: Gerência de Problemas . . . . .	37
Figura 3.6: Edição de palavras-chave . . . . .	38
Figura 3.7: Visualizando solução de um aluno . . . . .	39
Figura 4.1: Diagrama de Domínio do Portal de Algoritmos Novo . . . . .	42
Figura 4.2: Casos de Uso que envolvem o ator Administrador . . . . .	46
Figura 4.3: Casos de Uso que envolvem o ator Professor . . . . .	46
Figura 4.4: Casos de Uso que envolvem o ator Aluno . . . . .	47
Figura 4.5: Interface Gráfica de Cadastro de Aluno . . . . .	67
Figura 4.6: Interface Gráfica de Autenticação . . . . .	68
Figura 4.7: Interface Gráfica de Boas Vindas do Administrador e Professor .	69
Figura 4.8: Interface Gráfica de Problemas . . . . .	70
Figura 4.9: Interface Gráfica de Listagem Tipos de Problemas . . . . .	71
Figura 4.10: Interface Gráfica de Novo Problema . . . . .	72
Figura 4.11: Interface Gráfica de Palavras Chaves . . . . .	73
Figura 4.12: Interface Gráfica de Dados de Testes . . . . .	74
Figura 4.13: Interface Gráfica de Visualização de Solução do Problema . . . . .	75

Figura 4.14: Interface Gráfica de Grupos . . . . .	76
Figura 4.15: Interface Gráfica de Cadastro de Grupo . . . . .	77
Figura 4.16: Interface Gráfica de Cadastro de Participantes no Grupo . . . . .	78
Figura 4.17: Interface Gráfica de Cadastro de Lista de Problemas no Grupo . .	79
Figura 4.18: Interface Gráfica de Listagem de Alunos . . . . .	80
Figura 4.19: Interface Gráfica de Cadastro de Alunos . . . . .	81
Figura 4.20: Interface Gráfica de Lista de Problemas . . . . .	82
Figura 4.21: Interface Gráfica de Cadastro e Edição de Lista de Problemas . .	83
Figura 4.22: Interface Gráfica de Lista de Usuários . . . . .	84
Figura 4.23: Interface Gráfica de Cadastro e Edição de Usuários . . . . .	85
Figura 4.24: Interface Gráfica de Lista de Professores . . . . .	86
Figura 4.25: Interface Gráfica de Cadastro e Edição de Professores . . . . .	87
Figura 4.26: Interface Gráfica de Lista de Instituições . . . . .	88
Figura 4.27: Interface Gráfica de Cadastro e Edição de Instituições . . . . .	89
Figura 4.28: Interface Gráfica de Lista de Permissões . . . . .	90
Figura 4.29: Interface Gráfica de Cadastro e Edição de Permissões . . . . .	91
Figura 4.30: Interface Gráfica de Lista de Grupos de Administradores . . . . .	92
Figura 4.31: Interface Gráfica de Cadastro e Edição de Grupos de Adminis- tradores . . . . .	93
Figura 4.32: Interface Gráfica de Listagem de Países, Estados e Cidades . . . .	94
Figura 4.33: Interface Gráfica de Cadastro e Edição de País . . . . .	95
Figura 4.34: Interface Gráfica de Cadastro e Edição de Estado . . . . .	96
Figura 4.35: Interface Gráfica de Cadastro e Edição de Cidade . . . . .	97
Figura 4.36: Diagrama de Robustez de Cadastro de Aluno . . . . .	98
Figura 4.37: Diagrama de Robustez de Autenticação . . . . .	99
Figura 4.38: Diagrama de Robustez de Boas Vindas Administrador . . . . .	100
Figura 4.39: Diagrama de Robustez de Problemas . . . . .	100
Figura 4.40: Diagrama de Robustez de Tipo de Problema . . . . .	101
Figura 4.41: Diagrama de Robustez de Cadastro e Edição de Problema . . . .	101
Figura 4.42: Diagrama de Robustez de Palavras-chave . . . . .	102
Figura 4.43: Diagrama de Robustez de Dados de Testes . . . . .	102
Figura 4.44: Diagrama de Robustez de Visualização de Solução partindo da listagem de Problemas . . . . .	103
Figura 4.45: Diagrama de Robustez de Lista de Grupos . . . . .	103
Figura 4.46: Diagrama de Robustez de Cadastro e Edição de Grupo . . . . .	104
Figura 4.47: Diagrama de Robustez de Adicionar Participantes . . . . .	104
Figura 4.48: Diagrama de Robustez de Adicionar Lista de Problemas . . . . .	105
Figura 4.49: Diagrama de Robustez de Lista de Alunos . . . . .	106
Figura 4.50: Diagrama de Robustez de Cadastro de Alunos pelo Administrador	107

Figura 4.51: Diagrama de Robustez de Lista de Problemas . . . . .	108
Figura 4.52: Diagrama de Robustez de Cadastro e Edição de Lista de Problemas	109
Figura 4.53: Diagrama de Robustez de Lista de Usuários . . . . .	110
Figura 4.54: Diagrama de Robustez de Cadastro e Edição de Usuários . . . . .	110
Figura 4.55: Diagrama de Robustez de Lista de Professores . . . . .	111
Figura 4.56: Diagrama de Robustez de Cadastro e Edição de Professores . . .	111
Figura 4.57: Diagrama de Robustez de Lista de Instituições . . . . .	112
Figura 4.58: Diagrama de Robustez de Cadastro e Edição de Instituições . . .	112
Figura 4.59: Diagrama de Robustez de Lista de Permissões . . . . .	113
Figura 4.60: Diagrama de Robustez de Cadastro e Edição de Permissões . . .	113
Figura 4.61: Diagrama de Robustez de Lista de Grupos de Administradores .	114
Figura 4.62: Diagrama de Robustez de Cadastro e Edição de Grupos de Ad-	
ministradores . . . . .	114
Figura 4.63: Diagrama de Robustez de Lista de Países, Estados e Cidades . .	115
Figura 4.64: Diagrama de Robustez de Cadastro e Edição de País . . . . .	115
Figura 4.65: Diagrama de Robustez de Cadastro e Edição de Estado . . . . .	116
Figura 4.66: Diagrama de Robustez de Cadastro e Edição de Cidade . . . . .	116
Figura 4.67: Diagrama de Sequência de Cadastro e Edição de Problemas . .	117
Figura 4.68: Arquitetura Lógica do Portal de Algoritmos . . . . .	118
Figura 4.69: Arquitetura do Software do Portal de Algoritmos . . . . .	119

## **LISTA DE TABELAS**

Tabela 3.1: Tabelas do Banco de Dados do Portal de Algoritmos Atual . . . . .	33
Tabela 4.1: Tabelas do Banco de Dados do Portal de Algoritmos Novo . . . . .	43
Tabela 4.2: Requisitos funcionais . . . . .	44
Tabela 4.3: Requisitos não-funcionais . . . . .	45
Tabela 4.4: Caso de Uso Manter Usuários . . . . .	48
Tabela 4.5: Caso de Uso Manter Alunos . . . . .	49
Tabela 4.6: Caso de Uso Manter Professores . . . . .	50
Tabela 4.7: Caso de Uso Manter Administradores . . . . .	51
Tabela 4.8: Caso de Uso Manter Tipo de Problema . . . . .	52
Tabela 4.9: Caso de Uso Manter Problemas . . . . .	53
Tabela 4.10: Caso de Uso Manter Palavras-chave . . . . .	54
Tabela 4.11: Caso de Uso Manter Entrada e Saídas . . . . .	55
Tabela 4.12: Caso de Uso Manter Soluções de Problemas . . . . .	56
Tabela 4.13: Caso de Uso Manter Listas de Problemas . . . . .	57
Tabela 4.14: Caso de Uso Manter Instituições . . . . .	58
Tabela 4.15: Caso de Uso Manter Grupos . . . . .	59
Tabela 4.16: Caso de Uso Manter Permissões . . . . .	60
Tabela 4.17: Caso de Uso Manter Grupos de Administradores . . . . .	61
Tabela 4.18: Caso de Uso Manter Países . . . . .	62
Tabela 4.19: Caso de Uso Manter Estados . . . . .	63
Tabela 4.20: Caso de Uso Manter Cidades . . . . .	64
Tabela 4.21: Caso de Uso Aluno Manter suas Informações Pessoais . . . . .	65

## **RESUMO**

Este trabalho apresenta uma breve pesquisa bibliográfica sobre evolução de *software*, reengenharia e engenharia de *software*, bem como a metodologia ICONIX utilizada no decorrer do trabalho. O trabalho traz uma avaliação do Portal de Algoritmos antigo para se ter uma base para que se torne possível a evolução. Além da avaliação e do estudo teórico é proposto a evolução do mesmo, através de diagramas e interface gráficas, e por fim a escrita de um novo *software*.

**Palavras-chave:** Portal de Algoritmos, Algoritmos, Evolução, Software.

# 1 INTRODUÇÃO

No presente trabalho foi realizada a evolução do módulo de gerenciamento do portal de algoritmos da Universidade de Caxias do Sul. No ano de 2016 foi concluído pelo aluno Gabriel Weber um trabalho que evoluiu o analisador algorítmico dos problemas apresentados em sala de aula (SANTOS WEBER, 2015).

O portal de algoritmos, desenvolvido no ano de 2009 pelos professores Ricardo Vargas Dorneles e Delcino Picinin Junior, da Universidade de Caxias do Sul, tem por objetivo auxiliar no ensino da lógica de programação através da linguagem do português estruturado, também conhecida como portugol e é utilizado pelos alunos da Área de Conhecimento de Ciências Exatas e Engenharias e público em geral. Esse portal oferece ao aluno a possibilidade de exercitar sua lógica através de exercícios cadastrados pelos professores, utilizando a linguagem portugol em um editor específico. O aluno submete soluções de problemas a fim de validá-las e pode acompanhar seu desempenho através de um ranking de submissões de soluções corretas. O portal possui uma seção de gerenciamento que somente administradores podem acessar. Nessa administração há a possibilidade de acompanhar a evolução dos alunos, suas submissões de soluções, visualização e edição de usuário, de problemas, de dados de testes e palavras-chave (DORNELES; JUNIOR; ADAMI, 2010).

No ano de 2016 o *software* cliente do portal de algoritmos foi refeito, pois naquele ano foi preciso migrar o programa escrito em *Java Applet* para um aplicativo *Desktop*, esse *software Desktop* é instalado localmente e consumindo o *Web Services* do portal de algoritmos antigo. No entanto, o *software* servidor não foi migrado, e é neste contexto que este trabalho está inserido.

Na criação de um problema, o administrador informa um nome e uma descrição do problema a ser solucionado, dicas e palavras-chave, sendo que as últimas informações não são obrigatórias. Também são informadas entradas de dados para testes e as saídas esperadas para as entradas informadas. Já na edição do problema, as mesmas informações citadas acima podem ser alteradas.

No módulo de administração é possível manter todas as informações dos usuários, problemas, palavras-chave, dados de testes, dentre outras informações.

Devido à constante evolução tecnológica, o portal de algoritmos ficou defasado, com problemas de compatibilidade com os navegadores atuais e pouca ou nenhuma segurança. Funcionalidades limitadas no seu gerenciamento também foram apontadas pelos usuários como alvo de melhorias.

As tecnologias utilizadas no desenvolvimento do portal atual, *Python* versão 2.6, *Django* versão 1.2, *Plugins NPAPI* (*Netscape Plugin Application Programming Interface*) (*Java Applet*), estão desatualizadas e/ou foram descontinuadas. No caso da tecnologia *NPAPI*, esta foi totalmente desativada (PYTHON, 2018; DJANGO, 2018; GOOGLE, 2018).

Um dos problemas citados acima são as versões do *Python* e *Django*, que estão em versões desatualizadas e sem suporte técnico por seus desenvolvedores. Outro problema é o editor de soluções do portal atual foi programado como sendo um *Applet Java*. *Applets* executam nos navegadores, utilizando a tecnologia de plugins *NPAPI* (GOOGLE, 2018). *Plugins NPAPI* deixaram de ser suportados pelos navegadores atuais, pelo fato de causarem riscos de segurança para quem esteja utilizando. Desde o dia 1º de Setembro de 2015, o navegador *Google Chrome* deixou de suportar todas as tecnologias que utilizam *NPAPI*, como *Flash*, *Java*, entre outros. Mesmo eles não sendo mais suportados nativamente, é possível ativar isso no navegador, mas isso pode causar vulnerabilidades para quem deseja fazer isto (GOOGLE, 2018).

Para resolver os problems citados acima, este trabalho visa realizar a engenharia reversa do aplicativo, da modelagem de banco de dados atual, reengenharia de *software* e evolução de *software* através de técnicas de engenharia de *software* e desenvolvimento.

Através da engenharia reversa, o programa é analisado e são extraídas as informações, facilitando a documentação de sua organização e funcionalidades (SOMMERRVILLE, 2011).

Para realizar a engenharia reversa é preciso fazer a tradução de seu código fonte, sendo que o atual *software* foi desenvolvido utilizando a linguagem de programação *Python* e o *Django* (DJANGO, 2018).

Com *Python* e *Django* foram desenvolvidas todos os modelos de classes de domínio, que serão detalhadas no Capítulo 3, usando o *ORM* (*Object-relational mapping*) do *Django* foram criadas as tabelas de banco de dados e são realizadas as consultas.

Será realizada a diagramação das classes atuais utilizando-se da notação *UML* (*Unified Modeling Language*). Serão utilizados diagramas de classes de domínio, que representarão essas classes, interfaces e suas associações, para que depois sejam usadas no desenvolvimento de um modelo de sistema orientado a objetos (PRESSMAN, 2011; SOMMERRVILLE, 2011).

Através da engenharia de *software* será produzido um novo portal de algorit-

mos, desde os estágios iniciais da especificação do sistema até sua implementação e instalação. Com o uso de engenharia de *software* espera-se obter resultados de qualidade e requeridos dentro do cronograma (SOMMERVILLE, 2011).

No Capítulo 2 são apresentados todos os conceitos metodológicos da engenharia de *software*, quais tecnologias que serão utilizadas e suas funções no contexto do trabalho.

No Capítulo 3 é apresentada a reengenharia de *software* realizada no portal de algoritmos atual.

No Capítulo 4 é apresentada a modelagem para a evolução do *software*, suas interfaces e diagramas relacionados.

No Capítulo 5.1 é apresentado uma proposta de segurança para o servidor, essa segurança deve ser configurada na infra-estrutura.

No Capítulo 5 são apresentadas as considerações finais do trabalho.

## 2 EVOLUÇÃO DE SOFTWARE

Para manter um *software* útil para seu objetivo ele deve evoluir continuamente. Essa mudança pode ser a partir de uma pressão constante de mudanças que os usuários impõem, para facilitar e/ou automatizar algumas tarefas do dia-a-dia.

Todos os *softwares* passarão pelo processo de envelhecimento, isso é inevitável. Algumas causas de problemas podem ser previstas, minimizando os impactos dos danos causados. A continuidade de uso do *software* implica que ocorram mudanças, que podem ocorrer em regras de negócio ou nas expectativas dos usuários (SOMMERVILLE, 2011).

REZENDE (2005), define que um *software* tem um ciclo de vida de no máximo 10 anos, quando ele não sofre novas implementações. O ciclo de vida natural de um *software* abrange as seguintes fases: concepção, construção, implementações, implantação, maturidade e utilização plena, declínio, manutenção e morte.

Devido a esse ciclo de vida, uma evolução de *software* pode ser desencadeada por necessidades de novos componentes, por defeitos relatados ou devido a mudanças de outros sistemas (SOMMERVILLE, 2011).

A evolução de *software* compreende as mudanças que irão ocorrer a fim de deixá-lo completo e, se possível, livre de erros (SOMMERVILLE, 2011). Mas para essa evolução acontecer é necessário considerar diversos fatores que servirão de base para que um novo software seja construído, com base nos requisitos do atual.

O processo de evolução varia conforme o tipo de *software* que esteja sendo mantido, dos processos de desenvolvimento e as habilidades das pessoas envolvidas. Em alguns casos a evolução pode ser um processo informal, em que na maioria das vezes as mudanças resultam de conversas com usuários. Já em outros casos é um processo formal, envolvendo documentação estruturada que é produzida em cada estágio do processo (SOMMERVILLE, 2011).

O processo de evolução de *software* envolve a compreensão do *software* que tem que ser alterado. Para tornar-se possível a evolução uma das técnicas que podem ser usada é a reengenharia no *software* atual, visando melhorar sua estrutura e inteligibilidade (SOMMERVILLE, 2011).

Para tornar possível a evolução de *software* é preciso seguir alguns processos. Nas próximas seções serão apresentadas as metodologias e tecnologias que serão utilizadas neste trabalho.

- Reengenharia de *Software*
- Engenharia Reversa
- Engenharia de *Software*
- Processo de *Software*
- Engenharia de Requisitos
- Casos de Uso
- Modelagem de Domínio
- Metodologia ICONIX
- Projeto de Arquitetura
- Usabilidade
- Tecnologias
  - *Python* e *Django*
  - *Java EE*
  - *Wildfly*
  - Padrões de Projeto: *DAO* (*Data Access Object*)
  - *REST* (*Representational State Transfer*)
  - *AngularJS*

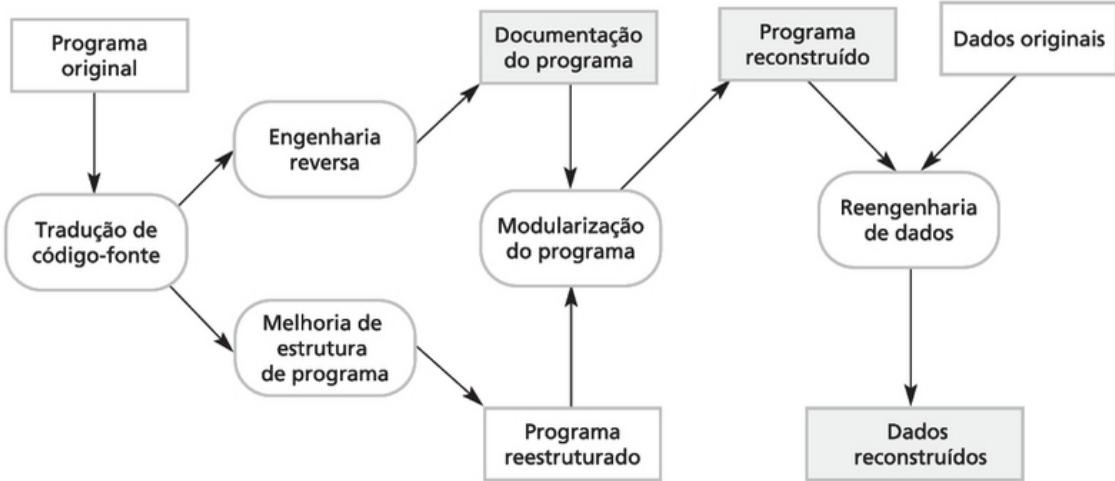
## 2.1 Reengenharia de Software

A reengenharia de *software* pode envolver a redocumentação do sistema, a refatoração da arquitetura, a mudança de linguagem de programação para uma linguagem mais moderna e modificações e atualização de estrutura e dos dados de sistema. A funcionalidade não é alterada, e geralmente deve evitar grandes mudanças na arquitetura (SOMMERVILLE, 2011).

Alguns benefícios importantes na reengenharia é o risco reduzido quando trata-se de um *software* crítico de negócio, onde podem haver erros nas especificações e atrasos no início do novo, e o custo reduzido, onde o custo da reengenharia se torna significamente menor do que o desenvolvimento de um novo.

A Figura 2.1 demonstra o processo geral da reengenharia, onde a entrada é um sistema legado e a saída é uma versão melhorada do mesmo.

Figura 2.1: Processo de Reengenharia



Fonte: (SOMMERVILLE, 2011)

1. Tradução de código-fonte: através de alguma ferramenta de tradução, o programa é convertido para uma versão mais atual da linguagem ou para outra diferente.
2. Engenharia reversa: o programa é analisado e as informações são extraídas a partir dele.
3. Melhoria na estrutura de programa: a estrutura de controle é analisada e modificada para que se torne mais fácil de ler e entender.
4. Modularização de programa: partes relacionadas do programa são agrupadas, e onde houver redundância, se apropriado, esta é removida. Em alguns casos, esse estágio pode envolver refatoração de arquitetura.
5. Reengenharia dos dados: os dados processados pelo programa são alterados para refletir as mudanças de programa.

Nem sempre é necessário seguir todas as etapas da Figura 2.1. Pode haver casos em que se utiliza o mesmo ambiente de desenvolvimento da linguagem de programação. Nesse caso não é necessário a tradução do código (SOMMERVILLE, 2011).

Na reengenharia um dos processos é a engenharia reversa. Na próxima seção é descrito como ela é utilizada no processo de evolução.

## 2.2 Engenharia Reversa

A engenharia reversa, segundo SOMMERVILLE (2011), consiste em uma técnica de análise de software com o objetivo de recuperar o projeto e suas especificações técnicas.

É possível fazer a engenharia reversa através de diversas formas, na maioria das

vezes utilizando os códigos fontes, além dos conhecimentos técnicos e experiências dos próprios desenvolvedores.

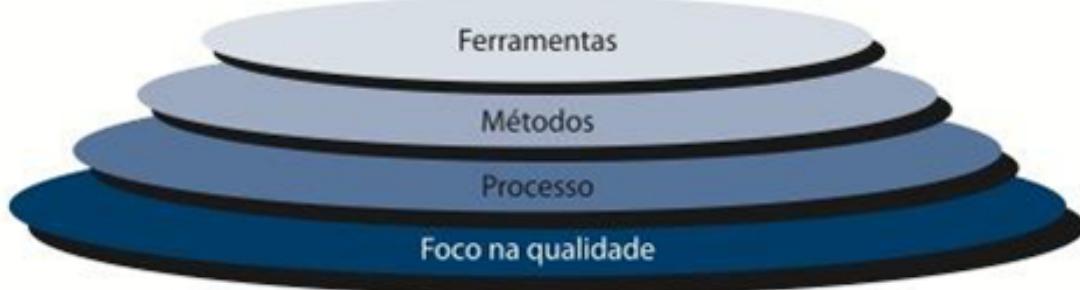
Na seção seguinte é descrita a Engenharia de *software* e suas respectivas camadas.

## 2.3 Engenharia de Software

Engenharia de *software* é uma disciplina cujo foco está em todos os aspectos da produção de software, partindo dos estágios iniciais da especificação do *software* até sua manutenção, quando o *software* já está em funcionamento (SOMMERVILLE, 2011). De acordo com REZENDE (2005), “é a metodologia de desenvolvimento e manutenção de sistemas modulares, com as seguintes características: processo dinâmico, integrado e inteligente de soluções tecnológicas; adequação aos requisitos funcionais do negócio do cliente e seus respectivos procedimentos pertinentes; efetivação de padrões de qualidade, produtividade e efetividade em suas atividades e produtos; fundamentação da Tecnologia da Informação disponível, viável, oportuna e personalizada; planejamento e gestão de atividades, recursos, custos e datas”.

Conforme podemos ver na Figura 2.2, a engenharia de *software* é uma tecnologia em camadas. A base para a engenharia de *software* é a camada de processos. O processo de engenharia de *software* é o método que permite manter as camadas de tecnologia coesa e possibilita o desenvolvimento do *software* (PRESSMAN, 2011).

Figura 2.2: Camadas da Engenharia de Software



Fonte: (PRESSMAN, 2011)

A engenharia de *software* é realizada através de processos de *software*, que serão descritos a seguir.

## 2.4 Processo de Software

Um processo de *software* é um conjunto de atividades, ações e tarefas relacionadas que levam à produção de um produto de *software* (SOMMERVILLE, 2011; PRESSMAN, 2011). No contexto da engenharia de *software*, um processo não é uma prescrição rígida de como desenvolver, ele é adaptável, que possibilita às pes-

soas realizar o trabalho de selecionar e escolher o conjunto apropriado de ações e tarefas (PRESSMAN, 2011).

Dentre muitos processos de *software* existentes todos devem incluir quatro atividades fundamentais (SOMMERVILLE, 2011).

- Especificação de *software*
- Projeto e implementação de *software*
- Evolução de *software*

De acordo com SOMMERVILLE (2011), essas atividades fazem parte do processo de *software*. Na prática eles são complexos, possuem subatividades, entre elas levantamento de requisitos, projeto de arquitetura, testes etc.

Para melhor entendimento desses processos, nas próximas seções serão descritas com mais detalhes algumas dessas atividades.

## 2.5 Engenharia de Requisitos

Engenharia de requisitos de sistemas basicamente é o conjunto das descrições do que o sistema deve fazer, o que ele oferece de serviço e restrições a seu funcionamento SOMMERVILLE (2011). A engenharia de requisitos abrange sete tarefas distintas: concepção, levantamento, elaboração, negociação, especificação, validação e gestão, onde geralmente algumas ocorrem em paralelo e todas podem ser adaptadas à necessidade de cada projeto (PRESSMAN, 2011)

Somente descrever os requisitos não é suficiente, é preciso entender o que está descrito, e essa é uma das tarefas mais difíceis enfrentadas por um engenheiro de *software*.

Os requisitos de *software* frequentemente são classificados em funcionais e não-funcionais.

Os Requisitos funcionais são declarações de serviço que o sistema deve fornecer, de como fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem explicitar o que o sistema não deve fazer (SOMMERVILLE, 2011).

Os Requisitos não-funcionais são restrições aos serviços ou funções oferecidas pelo sistema. Incluem restrições de *timing*, restrições no processo de desenvolvimento e restrições impostas pelas normas. Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais muitas vezes aplicam-se ao sistema como um todo (SOMMERVILLE, 2011).

## 2.6 Metodologia ICONIX

Para desenvolver um projeto, é necessário uma metodologia. Nesse trabalho, será utilizada a metodologia ICONIX.

A metodologia ICONIX foi elaborada por Doug Rosenberg e Kendal Scott, a partir de um processo simples e unificado dos pesquisadores Booch, Rumbaugh e Jacobson (ROSENBERG et al., 2005).

As vantagens de se utilizar a metodologia ICONIX são: metodologia prática, simples, específica de forma objetiva e possui rastreabilidade dos requisitos (ROSENBERG et al., 2005).

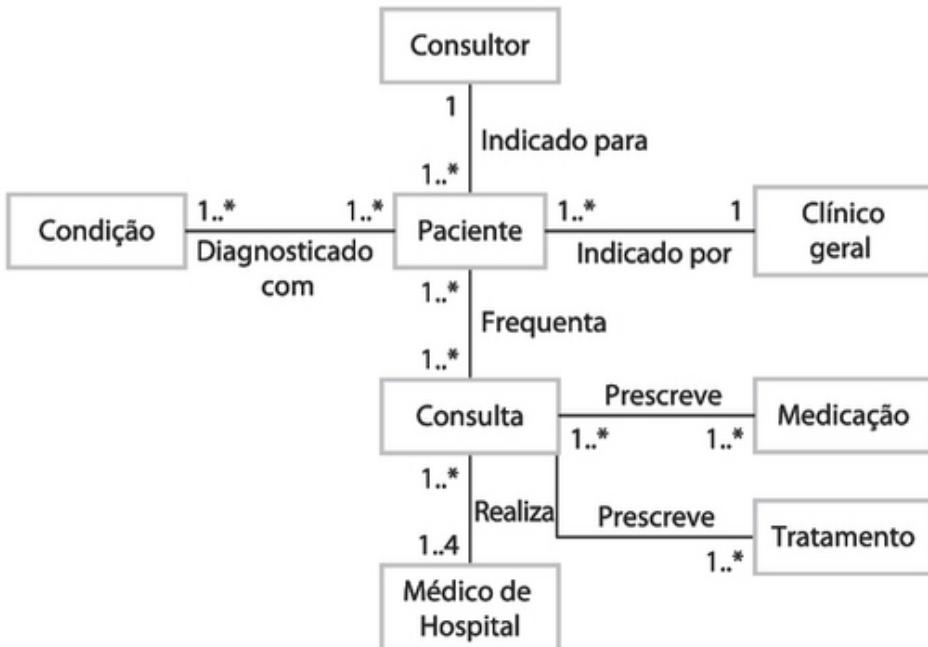
A metodologia ICONIX utiliza-se de um subconjunto da *UML* no qual apenas 4 diagramas são utilizados: diagramas de classe, diagrama de sequência, diagrama de robustez e caso de usos (ROSENBERG et al., 2005).

### 2.6.1 Diagramas de Classe

Os diagramas de classes são usados no desenvolvimento de um modelo de sistema orientado a objetos para mostrar as classes de um sistema e as associações entre essas classes (SOMMERVILLE, 2011).

A Figura 2.3 indica as relações entre os objetos da classe Paciente e objetos de outras classes (SOMMERVILLE, 2011).

Figura 2.3: Diagramas de Classe

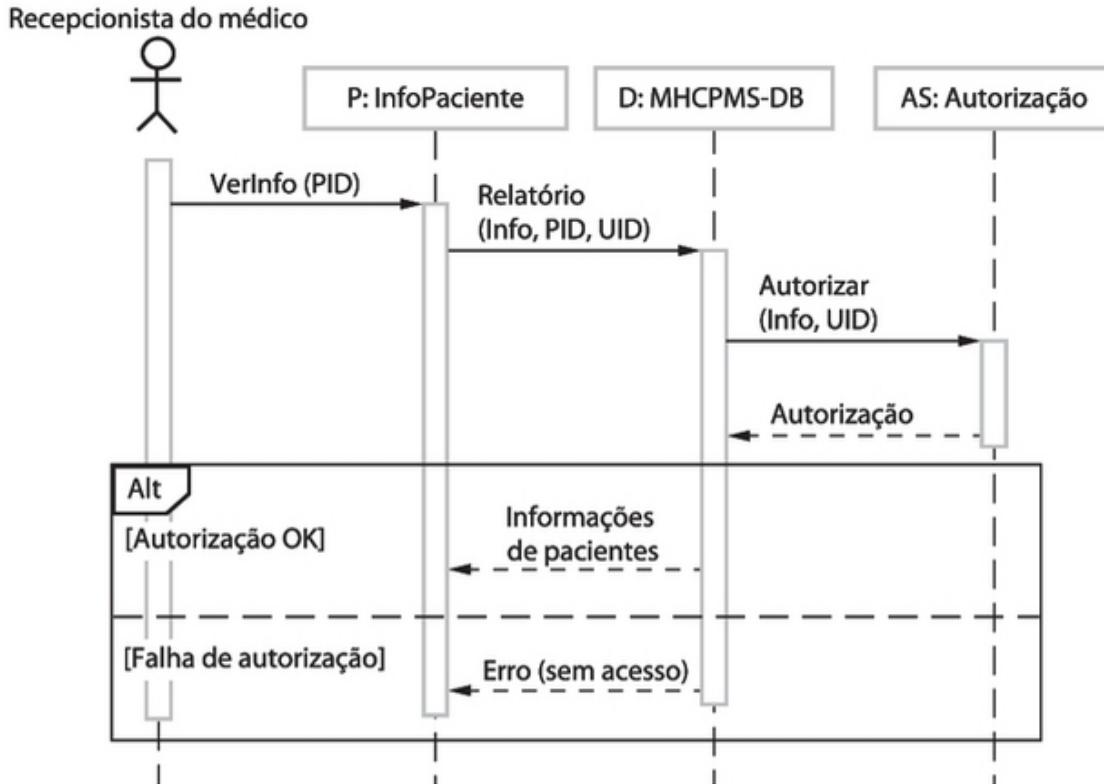


Fonte: (SOMMERVILLE, 2011)

### 2.6.2 Diagrama de Sequência

Os diagramas de sequência geralmente são utilizados para modelar as interações entre os atores e os objetos em um sistema (SOMMERVILLE, 2011).

Figura 2.4: Diagrama de Sequência



Fonte: (SOMMERVILLE, 2011)

A Figura 2.4 pode ser lida da seguinte maneira:

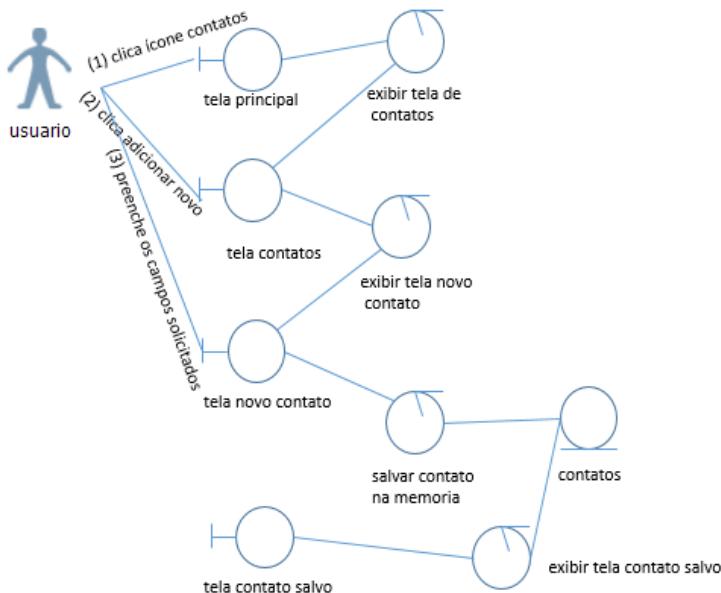
1. A recepcionista do médico aciona o método VerInfo em uma instância P da classe de objeto InfoPaciente, fornecendo o identificador do paciente (PID, do inglês *patient's identifier*). A instância P é um objeto de interface do usuário, exibido como um formulário que mostra os dados do paciente.
2. A instância P chama o banco de dados para retornar as informações necessárias, fornecendo o identificador da recepcionista, que permite a verificação de proteção (nessa fase, não importa de onde vem o esse UID - do inglês, *user's identifier*).
3. O banco de dados verifica, com o sistema de autorização, que o usuário está autorizado a essa ação.
4. Se autorizado, as informações de pacientes são retornadas, e um formulário é preenchido na tela do usuário. Se falhar a autorização, aparece uma mensagem de erro.

### 2.6.3 Diagrama de Robustez

Este é um diagrama que não existe na *UML* e é geralmente um diagrama de colaboração adaptado e que faz uso dos estereótipos *entity*, *boundary* e *control*. Ele é utilizado em processos como o *ICONIX* para passar da análise (o que) para o desenho (como). Esse é um diagrama que não é necessário ser mantido atualizado, uma vez que é utilizado apenas para a transição entre os *softwares*. A análise de robustez consiste então em ler o texto do caso de uso e identificar de forma preliminar, o conjunto de objetos que irão participar do caso de uso.

A Figura 2.5 representa a interação entre o usuário e as interface de um sistema, bem como todas as interações entre as interfaces. Como podemos observar, o ator usuário clica no “ícone de contatos” na tela principal, após o clique é exibido a tela de contatos, na sequência o ator clica em “adicionar novo” no qual resulta na exibição da tela de novo contato, continuando a ação o ator preenche os campos selecionados e faz a ação de salvar contato na memória retornando assim para a tela de exibição de contatos.

Figura 2.5: Diagrama e Robustez



Fonte: (GALEOTE, 2015)

### 2.6.4 Casos de Usos

Casos de uso tem por objetivo descrever os requisitos funcionais, delimitação do contexto do sistema documentado e entendimento dos requisitos, onde cada caso de uso deve descrever somente uma funcionalidade ou objetivo do sistema (SOMMERVILLE, 2011) e (PRESSMAN, 2011).

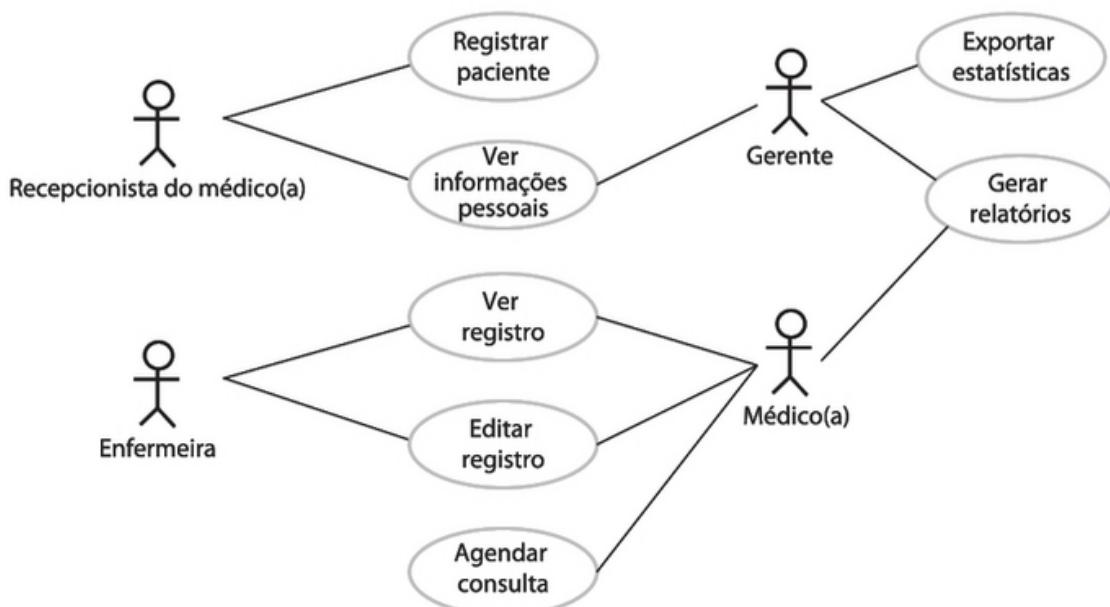
Um conjunto de casos de uso representa todas as possíveis interações que são

descritas nos requisitos de sistema. Os atores podem ser pessoas ou outros sistemas e são representados como figuras “palitos” e cada classe de interação é representada por uma elipse (SOMMERVILLE, 2011).

Casos de uso possuem atores e cenários, onde os atores podem ser pessoas ou outros sistemas que interagem entre si, e os cenários são sequências específicas de ações. Em outros termos casos de uso é uma coleção de cenários relacionados ao sucesso ou fracasso (LARMAN, 2007).

A Figura 2.6 apresenta um exemplo de caso de uso de um consultório médico, onde podemos observar todos os atores envolvidos e suas respectivas ações.

Figura 2.6: Casos de uso



Fonte: (SOMMERVILLE, 2011)

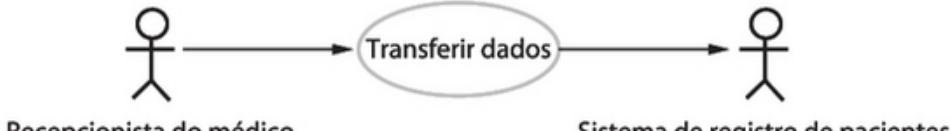
A modelagem de caso de uso é um apoiador para a elicitação de requisitos, geralmente descreve o que o usuário espera do sistema. Cada caso de uso representa uma tarefa que envolve a interação externa com o sistema (SOMMERVILLE, 2011).

Diagrama de casos de usos descrevem funcionalidades propostas para o novo sistema, fornecendo uma descrição clara e consistente do que o sistema deve fazer.

A Figura 2.7 representa o caso de uso de transferência de dados que envolve os atores Recepção do médico e Sistema de registro de pacientes (SOMMERVILLE, 2011).

Como podemos observar na Figura 2.7 a recepcionista do médico realiza a transferência de dados para o sistema de registro de pacientes.

Figura 2.7: Casos de uso de transferência de dados



Fonte: (SOMMERVILLE, 2011)

## 2.7 Modelos de Domínio

Um modelo de domínio exibe como está organizado o sistema em termos de seus componentes e seus relacionamentos. Podem ser estáticos ou dinâmicos, onde os modelos estáticos mostram a estrutura do sistema e os dinâmicos, onde é exibido quando ele está em execução (SOMMERVILLE, 2011).

De acordo com SOMMERVILLE (2011), “os diagramas de classe são utilizados no desenvolvimento de um modelo de sistema orientado a objetos para mostrar as classes de um sistema e as associações entre essas classes”.

Um modelo de domínio é uma representação visual de classes conceituais, ou objetos do mundo real, em um domínio (LARMAN, 2007). Também são conhecidos como modelos conceituais.

## 2.8 Projeto de Arquitetura

O projeto de arquitetura é a representação da estrutura de dados e seus componentes. Ele compreende como o sistema deve ser organizado a fim de atender as necessidades levantadas na engenharia de requisitos (SOMMERVILLE, 2011; PRESSMAN, 2011).

Na arquitetura em camadas o *software* é dividido em subconjuntos funcionais denominadas camadas, onde cada parte possui um propósito bem definido e cada parte conhece apenas a parte imediatamente inferior (SOMMERVILLE, 2011).

Na arquitetura em camadas encontram-se todas as partes do *software*, e define-se a responsabilidade de cada uma. Esse padrão de arquitetura é uma das maneiras de se conseguir independência entre elas, como por exemplo o padrão *MVC* (*Model-View-Controller*), em que são separadas as camadas de apresentação da interação dos dados do sistema (SOMMERVILLE, 2011; PRESSMAN, 2011).

## 2.9 Usabilidade

Sistemas devem ser flexíveis, simples e agradáveis de usar. A usabilidade é a principal ciência da *IHC* (Interação humano-computador), *IHC* tem por objetivo produzir sistemas usáveis, seguros e funcionais.

Na *IHC*, a usabilidade se refere a simplicidade e facilidade com que uma interface de um sistema pode ser utilizado. A importância do *IHC* no desenvolvimento de *software* é de ter uma definição de padrão visual, padrão de mensagens e prototipação e validação de telas com usuário, medindo a usabilidade e garantindo a padronização e consistência.

De acordo com BENYON (2011), um sistema com usabilidade terá as seguintes características:

- Será eficiente no sentido de que as pessoas poderão fazer as coisas mediante uma quantidade adequada de esforço.
- Será eficaz no sentido de que conterá as funções e o conteúdo de informações adequadas e organizadas de forma apropriada.
- Será fácil aprender como fazer as coisas e será fácil de lembrar como fazê-las após algum tempo.
- Será seguro de operar na variedade de contextos em que será usado.
- Terá um alto grau de utilidade no sentido de que fará as coisas que as pessoas querem que sejam feitas.

Até aqui foram apresentadas as metodologias que serão utilizadas na evolução do aplicativo. Na seção seguinte serão apresentadas as tecnologias escolhidas para a evolução do gerenciamento do portal de algoritmos.

## 2.10 Tecnologias

Nesta seção serão descritas as tecnologias que vão ser utilizadas na evolução do portal de algoritmos, tais como a linguagem de programação *Java*, *REST* e *AngularJS*. São tecnologias bem consolidadas no mercado, com *upgrade* garantido por tempo indeterminado, mantidas por empresas conhecidas e de grande porte.

### 2.10.1 Java EE

A linguagem *Java* é uma linguagem de programação orientada a objetos, com portabilidade, independência de plataforma, extensas bibliotecas de rotinas que facilitam recursos de rede e segurança, podendo executar programas via rede com restrições de execuções (JAVA, 2018).

Além disso, ela se destaca com a similaridade de sintaxe da linguagem C/C++, facilidade de internacionalização, simplicidade nas especificações, entre outras (JAVA, 2018).

O *JAVA EE* (*Java Enterprise Edition*) é uma série de especificações que descrevem como deve ser implementado um *software* que faz uso de serviços de infraestrutura. Também é considerado uma maneira de desenvolver aplicativos com suporte

a escabilidade, flexibilidade e segurança (JAVA, 2018).

### 2.10.2 WildFly

O servidor de aplicação *WildFly* implementa a mais recente versão do *JAVA EE*, sendo mantido pela *Red Hat* (WILDFLY, 2018).

Os *frameworks* que compõem o *JAVA EE* são fortemente testados em diversas combinações. De acordo com padrões com os quais o servidor foi desenvolvido o desenvolvedor pode focar nas regras de negócio e utilizar-se dos recursos de infraestrutura fornecidas pelo *framework* (WILDFLY, 2018).

### 2.10.3 DAO

*DAO* é um padrão de projeto para trabalhar com fontes de dados, que podem ser um banco de dados relacional, banco de dados orientado a objetos, entre outros. (ALUR, 2004).

Com *DAO* é possível adaptar a diferentes esquemas de armazenamento sem afetar outros componentes de negócio, basicamente o *DAO* atua como um adaptador entre o componente de apresentação de dados e a fonte de dados (ALUR, 2004).

### 2.10.4 REST

A *REST* é um estilo de arquitetura que define um conjunto de restrições e propriedades baseado no *HTTP* (*Hyper Text Transfer Protocol*), utilizando-se dos verbos desse protocolo. As princípios fundamentais do *REST* são: dê a todas as coisas um identificador, vincule as coisas, utilize métodos padronizados, recursos com múltiplas representações e comunique sem estado (FIELDING, 2000).

O *REST* possui um conjunto de operações bem definidas, os mais importantes são *GET*, *POST*, *PUT* e *DELETE* (RICHARDSON; AMUNDSEN, 2013). Conforme (FIELDING, 2000), *REST* é um modelo de arquitetura bem definido para servir aplicações *WEB*.

### 2.10.5 AngularJS

*AngularJS* foi criado por Misko Hevery e Adam Abrons em 2009, sendo seu código fonte aberto (*Open Source*). Ele é um *framework JavaScript* que é executado no navegador de internet do usuário, através do qual é possível aumentar sua produtividade no desenvolvimento *WEB* (BRANAS, 2014).

*AngularJs* foi construído com a crença de que a programação declarativa é a melhor escolha para a construção de interfaces de usuários. Para isso, o *AngularJs* aumenta o vocabulário do *HTML* (*Hyper Text Markup Language*) padrão, tornando mais versátil o desenvolvimento de sistemas *WEB* (BRANAS, 2014).

O resultado é o desenvolvimento reutilizável e aplicação sustentável de compo-

nentes, deixando para trás códigos desnecessários e mantendo a equipe focada no que é importante (BRANAS, 2014).

O padrão *MVC* ganhou muita popularidade nas fábricas de *software*, tornando-se um dos projetos de arquitetura empresarial mais utilizados. Basicamente o modelo (*Model*) consiste nos dados da aplicação, regras de negócios, lógicas e funções. A visão (*View*) é a saída de representação dos dados e o controle (*Controller*) faz a intermediação da entrada ou saída para o modelo ou visão.

Uma aplicação em *AngularJS* trabalha com *HTML* e *MVC*, mas também possui serviços, diretivas e filtros (BRANAS, 2014).

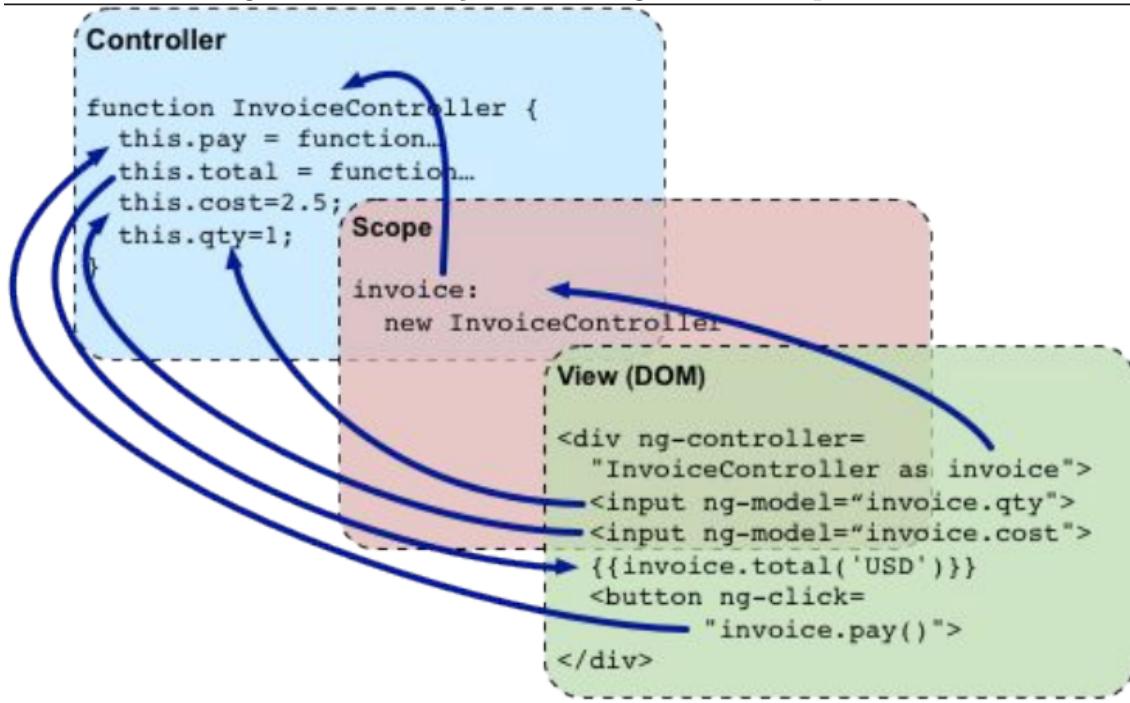
A *View* é escrita em *HTML*, que faz com que *web designers* e programadores trabalhem lado a lado, com a ajuda das diretivas, que são um tipo de extensão do vocabulário *HTML*, que traz a capacidade de executar tarefas de linguagem de programação (BRANAS, 2014).

Atrás da *View* existe um *Controller*, que contém toda a lógica do negócio usado pela *View*.

A conexão entre a visão e o controlador é feita por um objeto compartilhado chamado *scope*. Ele está localizado entre eles e é usado para trocar informações relacionados com o *Model*.

A Figura 2.8 representa a interação entre os componentes do *AngularJS*

Figura 2.8: Interação entre AngularJS e Arquitetura



Fonte: (BRANAS, 2014)

## 2.11 Ambiente Virtual de Aprendizagem

AVA (*Ambiente Virtual de Aprendizagem*) é um aplicativo *WEB* que possui um conjunto de elementos tecnológicos, onde são disponibilizadas ferramentas que permitem o acesso a um ou mais cursos ou disciplinas de uma instituição de ensino. De modo geral, um AVA refere-se ao uso de recursos digitais de comunicação, principalmente, através de softwares educacionais via internet que reúnem diversas ferramentas de interação (OLIVEIRA; COSTA; MOREIRA, 2004; VALENTINI; SOARES, 2005).

O objetivo de um ambiente virtual de aprendizagem é de facilitar o acesso de alunos ao ensino, práticas de exercícios e livros online para consulta. Na Universidade de Caxias do Sul o AVA já é utilizado desde meados de 2005, onde é possível acessar os materiais disponibilizados pelos professores em suas respectivas disciplinas, podendo também acompanhar o cronograma, entre outras funcionalidades (OLIVEIRA; COSTA; MOREIRA, 2004; VALENTINI; SOARES, 2005).

O portal de algoritmos é um ambiente virtual de aprendizagem utilizado pelos alunos da UCS (Universidade de Caxias do Sul) nas disciplinas de ciências exatas. O portal tem por objetivo auxiliar no ensino da lógica de programação através da linguagem do português estruturado (DORNELES; JUNIOR; ADAMI, 2010).

Vimos até aqui todos os conceitos necessários para o desenvolvimento do trabalho, no capítulo a seguir veremos a reengenharia do portal de algoritmos atual, onde são descritos os problemas do *software* e modelagem de uma nova aplicação.

## 3 AVALIAÇÃO DO SOFTWARE DO PORTAL DE ALGORITMOS DA UCS

Atualmente existe um aplicativo *Desktop* do Portal de Algoritmos que serve de interface gráfica para os alunos da UCS. Esse aplicativo foi desenvolvido para suprir a necessidade de uma atualização rápida para os alunos, pois *Java Applets* não está sendo suportado pelos navegadores atuais. Ele faz a comunicação com o *software* antigo via *Web Service*.

### 3.1 Diagrama de Classe de Domínio

Durante a tradução do código fonte dos modelos de classes do *Django*, foi constatada nenhuma padronização em nomes de variáveis e classes. Abaixo algumas considerações:

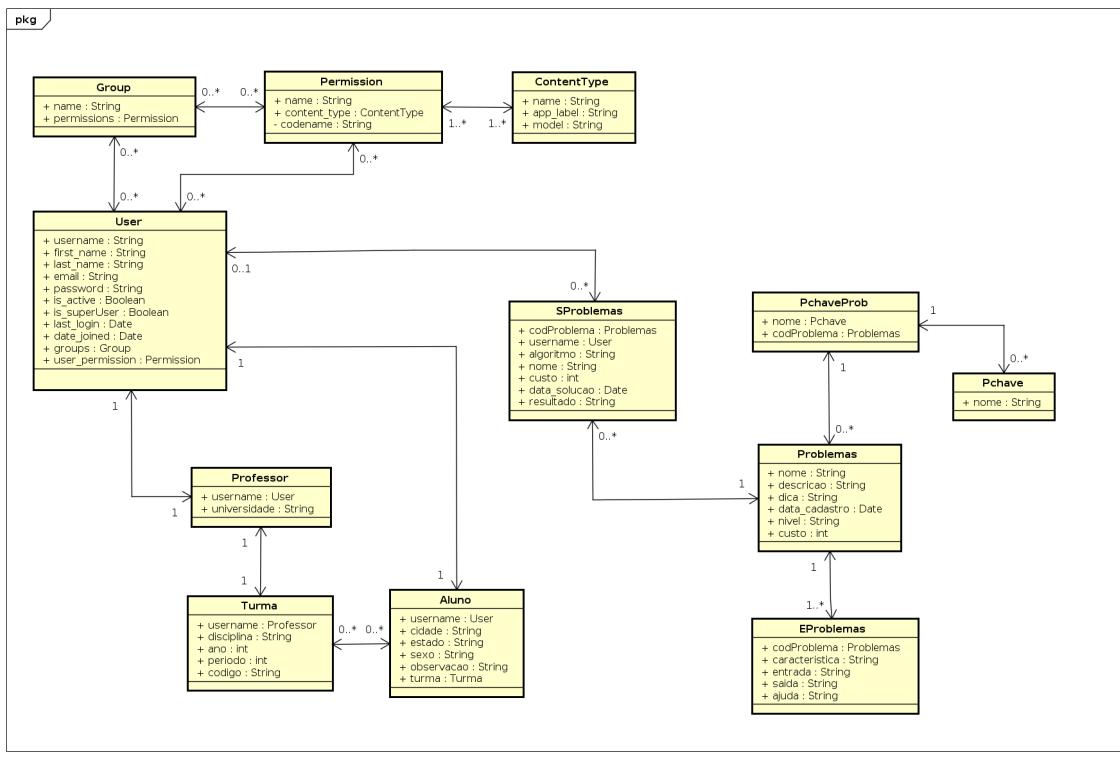
- Os campos não possuem nomes padronizados utilizando *CamelCase*<sup>1</sup>;
- Nomes de campos escritos em inglês e em português;
- Nomes de classes escritos em inglês e em português;
- Nomes de classes não são intuitivos quanto ao seu objetivo;

---

<sup>1</sup> *CamelCase* é a denominação em inglês para a prática de escrever palavras compostas, onde cada palavra é iniciada com minúscula ou maiúscula e unidas sem espaço

A Figura 3.1 representa as classes do portal que é utilizado atualmente.

Figura 3.1: Diagrama de Domínio do Portal de Algoritmos Atual



Fonte: (AUTOR, 2018)

O modelo acima foi obtido através da engenharia reversa do banco de dados do Portal de Algoritmos atual.

A Tabela 3.1 mostra as tabelas do banco de dados e suas descrições referentes ao portal de algoritmos. Visa-se manter todas as funcionalidades atuais após a evolução do mesmo, e adicionar algumas novas.

Tabela 3.1: Tabelas do Banco de Dados do Portal de Algoritmos Atual

	<b>Nome da Tabela</b>	<b>Descrição</b>
1	User	Representa os usuários. O usuário tem algumas funções importantes no sistema. De fato, somente com um usuário é possível ter acesso a áreas restritas, bem como resolver os exercícios do portal.
2	Groups	Representa os grupos do ORM do framework Django. Essa classe tem por objetivo agrupar usuários com determinadas permissões, mas para o sistema atual ela não é utilizada.
3	Permission	Corresponde às permissões utilizadas no sistema. Tem por objetivo definir permissões de acesso ao sistema, como por exemplo se um usuário possui acesso ao gerenciamento do portal, pode-se cadastrar um aluno ou um novo problema.
4	ContentType	Corresponde aos tipos de conteúdo. É uma classe padrão do Framework Django, que representa informações dos modelos utilizados no projeto. Sempre que é criado um modelo novo é criado um tipo de conteúdo automaticamente.
5	Aluno	Corresponde aos alunos cadastrados no sistema. No portal essa classe é um complemento aos dados do usuário, representando um aluno com suas respectivas informações.
6	Professor	Corresponde aos professores que podem cadastrar problemas no sistema. No portal é um complemento aos dados do usuário, representando um professor com suas respectivas informações.
7	Turma	Corresponde a turmas cadastradas no sistema. As turmas são usadas para agrupar alunos. Essa classe não é utilizada no sistema atual.
8	Problema	Corresponde aos problemas. Essa classe representa todas as informações correspondentes a um problema. Somente administradores do portal tem permissão para gravar informações.
9	Pchave	Corresponde às palavras-chave. Representa uma palavra-chave que é um facilitador para a pesquisa de problemas.
10	PchaveProb	Corresponde à relação entre a palavra-chave e um problema. Essa classe é utilizada para poder referenciar mais de uma palavra-chave para o mesmo problema.
11	EProblema	Corresponde às entradas e saídas esperadas de um determinado problema. Nela é possível informar alguma característica que ajude o usuário a resolver um determinado exercício.
12	SProblema	Corresponde às soluções submetidas pelos usuários. Possui relação direta com o usuário e um problema.

Fonte: (AUTOR, 2018)

Nesta seção foram descritas as classes de domínios do portal de algoritmos, a seguir são exibidas as interfaces gráficas do mesmo.

## 3.2 Interfaces Gráficas

Nessa seção são descritas as interfaces gráficas do software antigo, bem como os problemas encontrados nelas.

### 3.2.1 Cadastro de Aluno

A Figura 3.2 é a interface de Cadastro de Aluno, que é realizado pelo próprio aluno. As informações do aluno serão mantidas nas intefaces novas, mantendo assim consistência nas informações dos mesmos.

Figura 3.2: Cadastro de Aluno

Fonte: (AUTOR, 2018)

Problemas encontrados:

- *Java Applet* não funciona mais em nenhum navegador de internet atual.

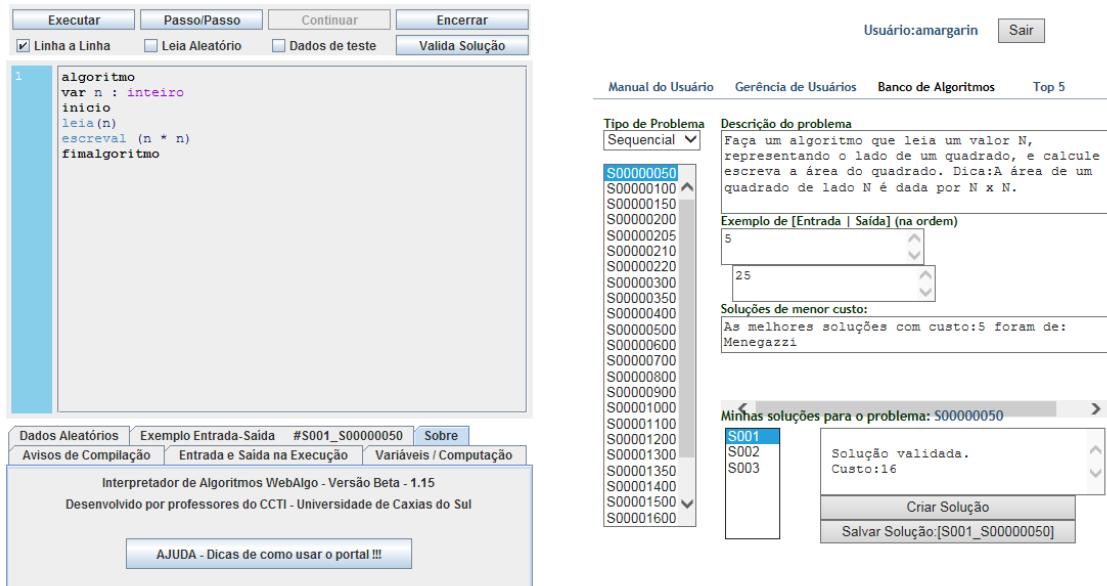
### 3.2.2 Criação de Solução de Problemas

A Figura 3.3 exibe um aluno já autenticado no portal e um problema e sua solução já selecionados. Nessa interface gráfica encontramos as seguintes funcionalidades:

- Criar nova solução.
- Salvar solução.
- Executar solução.
- Validar solução.

Problemas encontrados:

Figura 3.3: Criação de Solução de Problemas



Fonte: (AUTOR, 2018)

- *Java Applet* não funciona mais em nenhum navegador de internet atual.
- Pouca Usabilidade, uma vez que para selecionar outro problema são necessárias diversas confirmações antes de executar a ação.

### 3.2.3 Gerenciamento de Alunos

A Figura 3.4 é a interface de gerenciamento de alunos. Nessa interface encontramos as seguintes funcionalidades:

- Pesquisa de problemas: é possível realizar uma pesquisa livre, entendendo-se por livre qualquer palavra digitada no campos de pesquisa.
- Pesquisa de alunos: é possível realizar uma pesquisa livre, entendendo-se por livre qualquer palavra digitada no campos de pesquisa.
- Selecionando um problema, automaticamente o sistema faz uma pesquisa dos alunos que já solucionaram o problema, e selecionando o aluno é realizada uma pesquisa para encontrar as soluções desse aluno.
- Selecionando um aluno, automaticamente o sistema faz uma pesquisa dos problemas que esse aluno já resolveu, e selecionando o problema é realizada uma pesquisa para encontrar as soluções desse aluno.

Figura 3.4: Gerênciade Alunos

Fonte: (AUTOR, 2018)

Problemas encontrados:

- Pouca Usabilidade, uma vez que os campos dispostos na interface de maneira

pouco intuitiva ao usuário, bem como muito pequenos.

### 3.2.4 Gerenciamento de Problemas

A Figura 3.5 é a interface de gerenciamento de problemas. Nessa interface encontramos as seguintes funcionalidades:

- Cadastrar novo problema
- Editar um problema:
  - Alterar descrição e dicas
  - Alterar palavras-chave
  - Alterar entradas e saídas

Figura 3.5: Gerência de Problemas

The screenshot shows a web-based application for managing problems. At the top, there's a header with the title "PortAlgo - Gerencia de Problemas" and the administrator name "Administrador: amargarin". Below the header, there are two main sections: "Novo Problema" (New Problem) and "Cadastrar Problema" (Create Problem). On the left, a sidebar lists existing problems with IDs like S00000050, S00000100, S00000150, etc. The main section displays a specific problem with ID [S00000050]. It includes fields for "Descrição do problema" (Description of the problem) containing the text "Faça um algoritmo que leia um valor N, representando o lado de um quadrado, e calcule e escreva a área do quadrado.", "Dica para o problema" (Hint for the problem) containing "A área de um quadrado de lado N é dada por N x N.", "Palavras chave o problema" (Keywords for the problem), "Exemplo de entrada (na ordem)" (Example of input (in order)), and "Saída apóis e/ou durante processamento da entrada" (Output after and/or during processing of the input). At the bottom, there are four buttons: "Gravar Problema" (Save Problem), "Alterar Descrição/Dicas" (Change Description/Hints), "Alterar Palavras Chave" (Change Keywords), and "Alterar Entradas/Saídas" (Change Inputs/Outputs).

Fonte: (AUTOR, 2018)

Problemas encontrados:

- Pouca Usabilidade.
  - Campos dispostos na interface de maneira pouco intuitiva ao usuário.
  - Não possui campo de pesquisa.
  - Ações descentralizadas que confundem o usuário.

### 3.2.5 Edição de Palavra-Chave

A Figura 3.6 mostra a mensagem de sucesso após a edição de alguma informação do problema.

Figura 3.6: Edição de palavras-chave

**PortAlgo - Gerencia de Problemas      Administrador: amargarin**

The screenshot shows the PortAlgo application interface for managing problems. On the left, a sidebar lists existing problems with S00000050 selected. The main area displays problem details for S00000050, including its description: "Faca um algoritmo que leia um valor N, representando o lado de um quadrado, e calcule e escreva a área do quadrado." Below this is a hint: "A área de um quadrado de lado N é dada por N x N." The keyword field contains "Palavras chave o problema". The input example field shows "Exemplo de entrada (na ordem)" with "5|3|4|6". The output example field shows "Saída após e/ou durante processamento" with "25|9|16|36". At the bottom, buttons include "Gravar Problema", "Alterar Descrição/Dicas", "Alterar Palavras Chave" (which is highlighted in blue), and "Alterar Entradas/Saídas". A modal dialog box titled "Mensagem da página da web" with a warning icon and an "OK" button is overlaid on the interface.

Fonte: (AUTOR, 2018)

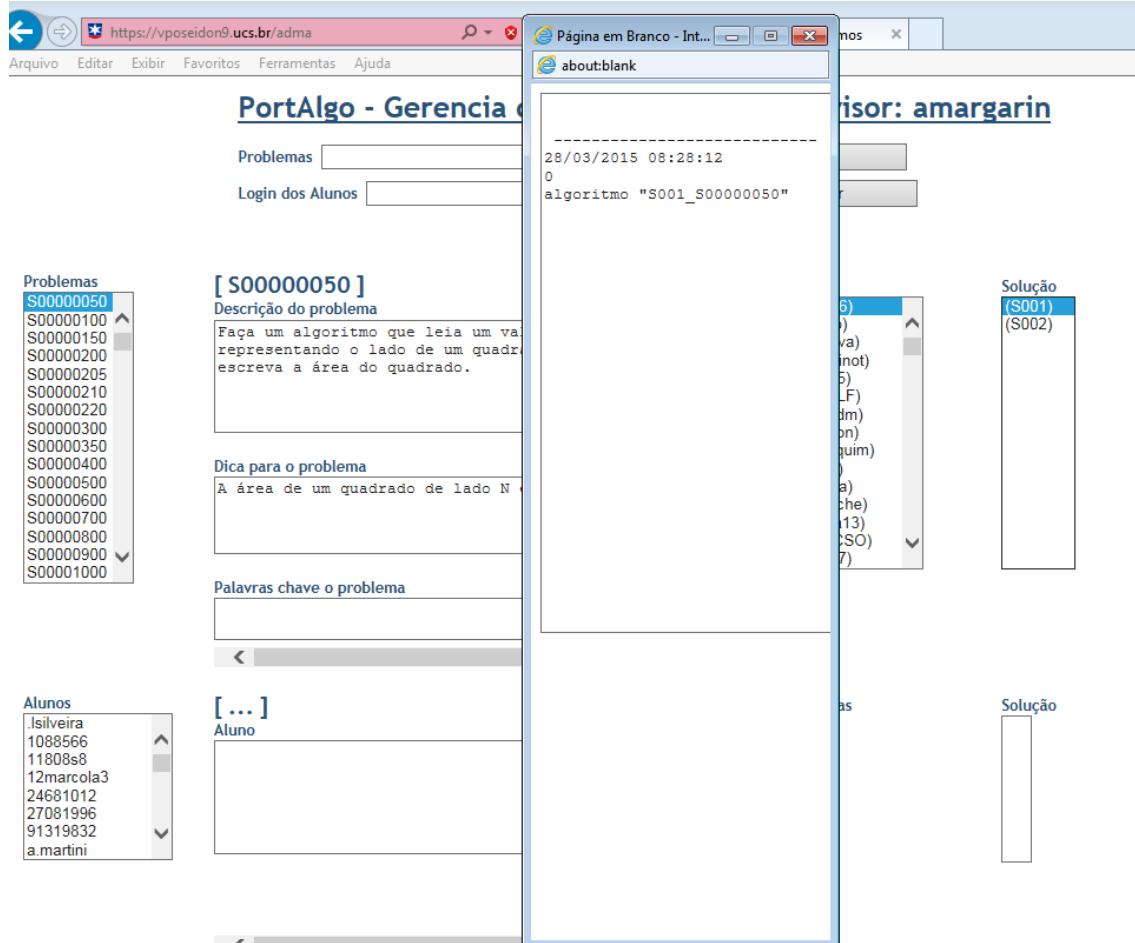
Problemas encontrados:

- Pouca Usabilidade: a mensagem não possui uma informação clara do que foi editado, ou o que foi realizado.

### 3.2.6 Solução de Problemas

A Figura 3.7 exibe a solução de um problema em uma nova janela.

Figura 3.7: Visualizando solução de um aluno



Fonte: (AUTOR, 2018)

Problemas encontrados:

- Pouca Usabilidade.
  - A nova janela que é aberta é muito pequena e sem a possibilidade de aumentar.
  - Não é possível editar a solução do problema.

Para solucionar e deixá-lo mais usável, fazendo com que mais alunos sejam beneficiados pelo portal, bem como facilitando o uso por parte dos professores, no próximo capítulo será descrita toda a modelagem, novas interfaces e novas funcionalidades, utilizando-se de tecnologias mais atuais.

Foram apresentados nesse capítulo problemas atuais do serviço do portal de algoritmos. No próximo capítulo serão apresentados alternativas de soluções para

estes problemas.

## 4 PROPOSTA DE SOLUÇÃO

O presente trabalho tem por objetivo realizar a evolução do gerenciamento do portal de algoritmos. Para tanto, foi realizada a engenharia reversa do *software* atual, através da análise do código fonte, suas funcionalidades, sua arquitetura e seu banco de dados.

Para descrever a evolução de *software*, nas seções seguintes serão apresentados conceitos e artefatos da engenharia de *software*. O *software* será modelado utilizando-se de artefatos da metodologia ICONIX será modelado o *software*.

A solução proposta será desenvolvida na linguagem de programação Java, a fim de unificar as tecnologias do gerenciamento do portal de algoritmos com seu analisador algorítmico.

O *software* será construído com uma arquitetura orientada a serviços, cujo objetivo é que ele seja disponibilizado em interfaces, ou seja, seja acessível através de *REST*.

Na evolução do *software* serão mantidas as funcionalidades atuais e adicionadas novas funcionalidades.

Funcionalidades que serão mantidas são:

- Manter Usuários.
- Manter Permissões.
- Manter Tipos de Problemas.
- Manter Problemas.
- Manter Entradas e Saídas
- Manter Palavras-chave
- Manter Soluções de Problemas

Novas funcionalidades:

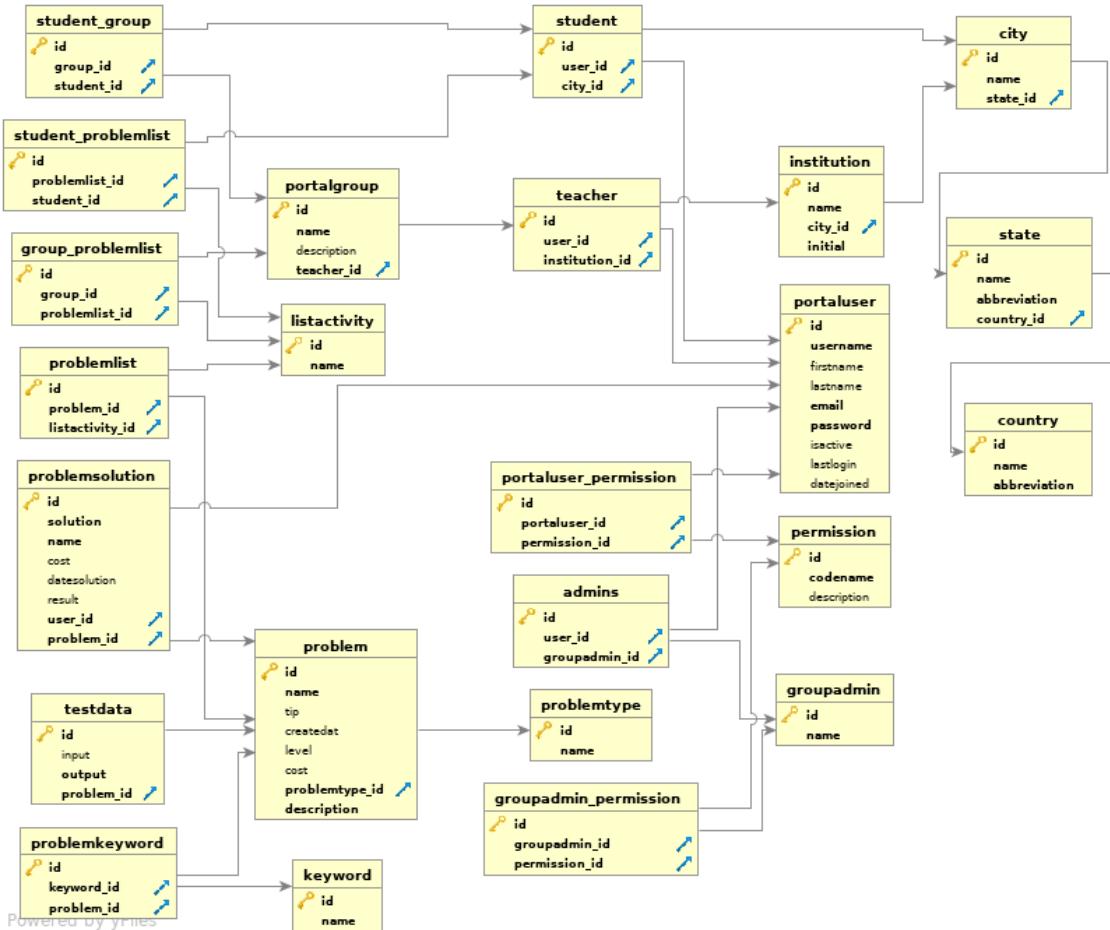
- Manter Grupos de Administradores.
- Manter Alunos e Professores.
- Manter Grupos de Alunos.

- Manter Instituições.
- Manter Países, Estados e Cidades.

## 4.1 Diagrama de Classe de Domínio

A Figura 4.1 representa o diagrama de classe de domínio proposto para a evolução do portal de algoritmos. Nesse novo diagrama foram melhoradas as descrições dos campos, descrevendo-os em inglês.

Figura 4.1: Diagrama de Domínio do Portal de Algoritmos Novo



Fonte: (AUTOR, 2018)

A Tabela 4.1 mostra as tabelas do banco de dados e suas descrições referente ao portal de algoritmos.

Tabela 4.1: Tabelas do Banco de Dados do Portal de Algoritmos Novo

	<b>Nome da Tabela</b>	<b>Descrição</b>
1	admins	Representa a classe de administradores do portal de algoritmos. Essa classe foi criada para fazer a associação entre um portaluser e um groupadmin.
2	city	Representa as cidades dos usuários e instituições, classe nova que surgiu da necessidade de padronizar nomes de cidades.
3	country	Representa os países dos usuários e instituições, classe nova que surgiu da necessidade de padronizar nomes de países.
4	group_problemlist	Representa a associação múltipla entre portalgroup e problemlist.
5	groupadmin	Representa a classe de Grupo de Administradores. Essa classe é equivalente à classe Group do portal de algoritmos antigo.
6	groupadmin_permission	Representa a associação múltipla entre groupadmin e permission.
7	institution	Representa as instituições que utilizam o portal de algoritmos e serve para identificar a localização de um professor. Essa classe surgiu da necessidade de padronização, pois antes era um campo de texto livre na classe Professor.
8	keyword	Representa as palavras-chave. É equivalente à classe Pchave do portal de algoritmos antigo.
9	listactivity	Representa uma lista de problemas.
10	permission	Representa as permissões. Essa classe é equivalente à classe Permission do portal de algoritmos antigo.
11	portalgroup	Representa os grupos de alunos. É equivalente à classe Turma no portal de algoritmos antigo, que não estava sendo utilizada.
12	portaluser	Representa os usuários. Possui a mesma função da classe User do portal de algoritmos antigo.
13	portaluser_permission	Representa a associação múltipla entre portaluser e permission.
14	problem	Representa os problemas do portal de algoritmos. É equivalente à classe Problema do portal de algoritmos antigo.
15	problemkeyword	Representa a relação entre palavras-chave com o problema. É equivalente à classe PchaveProb do portal de algoritmos antigo.
16	problemlist	Representa a associação múltipla entre problem e listactivity.
17	problemsolution	Representa as soluções submetidas pelos usuários. É equivalente à classe SProblema.
18	problemtypes	Representa os tipos de problemas. É uma classe nova que surgiu da necessidade de agrupar os problemas por tipos, não mais pela nomenclatura utilizada atualmente.
19	state	Representa os estados dos usuários e instituições, classe nova que surgiu da necessidade de padronizar nomes de estados.
20	student	Representa os estudantes. Essa classe é equivalente à classe Aluno do portal de algoritmos antigo.
21	student_group	Representa a associação múltipla entre student e portalgroup.
22	student_problemlist	Representa a associação múltipla entre student e problemlist.
23	teacher	Representa os professores. Essa classe é equivalente à classe Professor do portal de algoritmos antigo.
24	testdata	Representa os dados de teste dos problemas. É equivalente à classe EProblema.

Fonte: (AUTOR, 2018)

Ao realizar a modelagem do diagrama de classes de domínios do novo portal de algoritmos, foram padronizados nomes das tabelas e colunas, de uma forma a ser mais descriptivos. Na seção seguinte são descritos os requisitos funcionais e não-funcionais do projeto.

## 4.2 Requisitos de projeto

Antes do desenvolvimento do *software* é preciso realizar o levantamento de requisitos funcionais e não-funcionais. Esses requisitos descrevem o que o sistema deve fazer e suas restrições.

### 4.2.1 Requisitos Funcionais

A Tabela 4.2 mostra os requisitos funcionais do portal de algoritmos. Nesta Tabela entende-se quando se refere em “manter”, listar, cadastrar, editar e deletar um objeto.

Tabela 4.2: Requisitos funcionais

Código	Descrição
RF-001	O software deve manter Usuários.
RF-002	O software deve manter Alunos.
RF-003	O software deve manter Professores.
RF-004	O software deve manter Administradores.
RF-005	O software deve manter Tipo de Problemas.
RF-006	O software deve manter Problemas.
RF-007	O software deve manter Palavras-chave.
RF-008	O software deve manter Entradas e Saídas para os Problemas.
RF-009	O software deve manter Soluções de Problemas.
RF-010	O software deve manter Lista de Problemas.
RF-011	O software deve manter Instituições.
RF-012	O software deve manter Grupos.
RF-013	O software deve manter Grupos de Administradores.
RF-014	O software deve manter Permissões.
RF-015	O software deve manter Países.
RF-016	O software deve manter Estados.
RF-017	O software deve manter Cidades.
RF-018	O usuário deve manter suas informações pessoais

Fonte: (AUTOR, 2018)

#### 4.2.2 Requisitos Não-Funcionais

A Tabela 4.3 são os requisitos não-funcionais do portal de algoritmos.

Tabela 4.3: Requisitos não-fucionais

Código	Descrição
RFN-001	O software deve ser uma aplicação para internet.
RFN-002	O software deve possuir uma API REST.
RFN-003	O software deve executar nos principais navegadores de internet.

Fonte: (AUTOR, 2018)

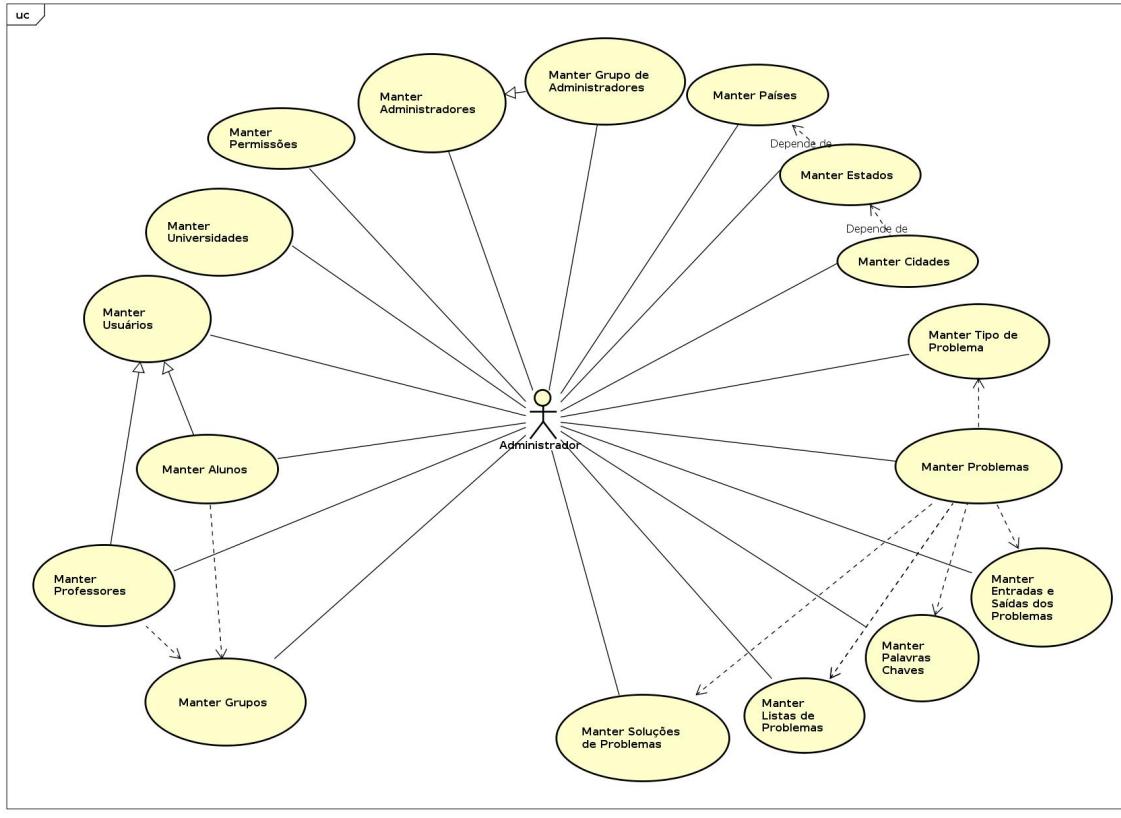
Cada requisito funcional elicitado pode se tornar um caso de uso. Esses casos de uso serão descritos na seção seguinte.

### 4.3 Casos de Uso

Após o levantamento e descrição dos requisitos do *software*, pode ser criado o diagrama macro de interação entre o usuário e os casos de uso do sistema, ressaltando que um *software* dessa natureza deve ter um ambiente de execução com acesso a *internet*.

Na Figura 4.2 estão representados os casos de uso em que o ator “Administrador” está envolvido para o gerenciamento do portal.

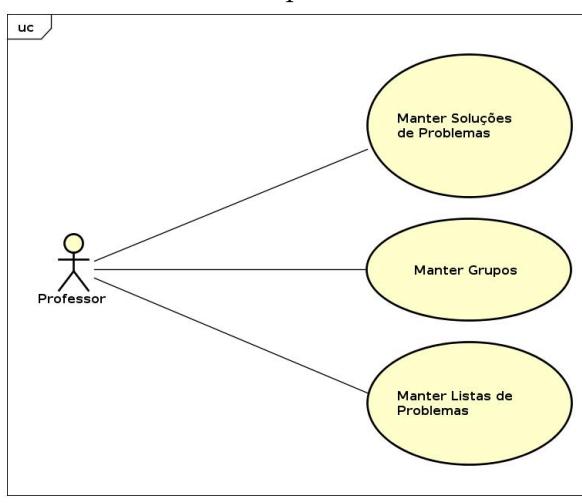
Figura 4.2: Casos de Uso que envolvem o ator Administrador



Fonte: (AUTOR, 2018)

Na Figura 4.3 estão representados os casos de uso em que o ator “Professor” está envolvido para o gerenciamento do portal.

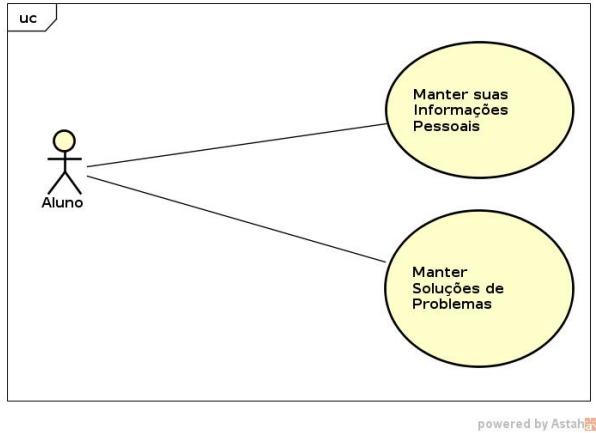
Figura 4.3: Casos de Uso que envolvem o ator Professor



Fonte: (AUTOR, 2018)

Na Figura 4.4 estão representados os casos de uso em que o ator “Aluno” está envolvido para o gerenciamento do portal.

Figura 4.4: Casos de Uso que envolvem o ator Aluno



Fonte: (AUTOR, 2018)

A seguir são apresentados detalhadamente cada um dos casos de uso apresentados nas Figuras 4.2, 4.3 e 4.4. Serão apresentados os fluxos de execução principais e alternativos de cada caso de uso.

### 4.3.1 Descrição dos Casos de Uso

Nesta seção são descritos os casos de uso levantados a partir dos requisitos funcionais.

#### 4.3.1.1 *Manter Usuários*

A Tabela 4.4 descreve o caso de uso de manter o cadastro, edição e remoção das informações dos usuários. Esse caso de uso possui um único ator, o "Administrador" e a pré-condição é ser um administrador do sistema.

Tabela 4.4: Caso de Uso Manter Usuários

<b>CASO DE USO 001 - Manter Usuários (RF-001)</b>	
<b>Descrição</b>	O administrador cadastra usuários para o software, podendo editar e remover.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de usuários.</li> <li>2. Sistema lista todos usuários já cadastrados.</li> <li>3. Administrador acessa cadastro de usuários.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador preenche campos obrigatórios.</li> <li>6. Administrador submete formulário.</li> <li>7. Sistema retorna para listagem de usuários.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de usuários.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos campos obrigatórias.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Usuário cadastrado

Fonte: (AUTOR, 2018)

#### 4.3.1.2 Manter Alunos

A Tabela 4.5 descreve o caso de uso de manter o cadastro, edição e remoção das informações dos alunos. Esse caso de uso possui um único ator, o “Administrador” e possui duas pré-condições, ser administrador do sistema e o aluno precisa estar associado a um usuário.

Tabela 4.5: Caso de Uso Manter Alunos

<b>CASO DE USO 002 - Manter Alunos (RF-002)</b>	
<b>Descrição</b>	O administrador cadastra alunos para o software, podendo editar e remover. O usuário é cadastrado automaticamente.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	CASO DE USO 001. Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de alunos.</li> <li>2. Sistema lista todos alunos já cadastrados.</li> <li>3. Administrador acessa cadastro de alunos.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador preenche campos obrigatórios.</li> <li>6. Administrador submete formulário.</li> <li>7. Sistema retorna para listagem de alunos.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de alunos.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Aluno cadastrado.

Fonte: (AUTOR, 2018)

#### 4.3.1.3 Manter Professores

A Tabela 4.6 descreve o caso de uso de manter o cadastro, edição e remoção das informações dos professores. Esse caso de uso possui um único ator, o “Administrador” e possui duas pré-condições, ser administrador do sistema e o professor precisa estar associado a um usuário.

Tabela 4.6: Caso de Uso Manter Professores

<b>CASO DE USO 003 - Manter Professores (RF-003)</b>	
<b>Descrição</b>	O administrador cadastra professores para o software, podendo editar e remover. O usuário é cadastrado automaticamente.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	CASO DE USO 001. Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de professores.</li> <li>2. Sistema lista todos professores já cadastrados.</li> <li>3. Administrador acessa cadastro de professores.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador preenche campos obrigatórios.</li> <li>6. Administrador submete formulário.</li> <li>7. Sistema retorna para listagem de professores.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de professores.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Aluno cadastrado.

Fonte: (AUTOR, 2018)

#### 4.3.1.4 Manter Administradores

A Tabela 4.7 descreve o caso de uso de manter o cadastro, edição e remoção das informações dos administradores do sistema. Esse caso de uso possui um único ator, o “Administrador”, possui duas pré-condições, ser um administrador do sistema e possuir um usuário para adicionar como administrador do sistema.

Tabela 4.7: Caso de Uso Manter Administradores

<b>CASO DE USO 004 - Manter Administradores (RF-004)</b>	
<b>Descrição</b>	O administrador cadastra outros administradores para o software, podendo editar e remover. O usuário é cadastrado automaticamente.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	CASO DE USO 001. Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de administradores.</li> <li>2. Sistema lista todos administradores já cadastrados.</li> <li>3. Administrador acessa cadastro de administradores.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador preenche campos obrigatórios.</li> <li>6. Administrador submete formulário.</li> <li>7. Sistema retorna para listagem de administradores.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de administradores.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Administrador cadastrado.

Fonte: (AUTOR, 2018)

#### 4.3.1.5 Manter Tipo de Problema

A Tabela 4.8 descreve o caso de uso de manter o cadastro, edição e remoção das informações de um tipo de problema. Um tipo de problema serve para agrupar problemas que possuem relação.

Tabela 4.8: Caso de Uso Manter Tipo de Problema

<b>CASO DE USO 005 - Manter Tipo de Problema (RF-005)</b>	
<b>Descrição</b>	O administrador cadastra um tipo de problema, podendo editar e remover. O tipo de problema agrupa problemas relacionados.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de tipos de problemas.</li> <li>2. Sistema lista todos tipos de problemas já cadastrados.</li> <li>3. Administrador acessa cadastro de tipos de problemas.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador preenche campos obrigatórios</li> <li>6. Administrador submete formulário.</li> <li>7. Sistema retorna para listagem de tipos de problemas.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de tipos de problemas.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> <li>2. Sistema exibe página de login para usuário.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Tipo de Problema cadastrado.

Fonte: (AUTOR, 2018)

#### 4.3.1.6 Manter Problemas

A Tabela 4.9 descreve o caso de uso de manter o cadastro, edição e remoção das informações de problemas.

Tabela 4.9: Caso de Uso Manter Problemas

<b>CASO DE USO 006 – Manter Problemas (RF-006)</b>	
<b>Descrição</b>	O administrador cadastra problemas, podendo editar e remover. Para o cadastro de um problema é necessário possuir um tipo de problema já cadastrado.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	CASO DE USO 005. Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de problemas.</li> <li>2. Sistema lista todos problemas já cadastrados.</li> <li>3. Administrador acessa cadastro de problemas.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador preenche campos obrigatórios.</li> <li>6. Administrador submete formulário.</li> <li>7. Sistema retorna para listagem de problemas.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de problemas.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Problema cadastrado.

Fonte: (AUTOR, 2018)

#### 4.3.1.7 Manter Palavras-chave

A Tabela 4.10 descreve o caso de uso de manter o cadastro, edição e remoção das informações das palavras-chave de um ou mais problemas.

Tabela 4.10: Caso de Uso Manter Palavras-chave

<b>CASO DE USO 007 – Manter Palavra-chave (RF-007)</b>	
<b>Descrição</b>	O administrador cadastra palavras-chave para um problema, podendo editar e remover.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	CASO DE USO 006. Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de problemas.</li> <li>2. Sistema lista todos problemas já cadastrados.</li> <li>3. Administrador acessa cadastro de problemas.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador seleciona o cadastro de palavras-chave.</li> <li>6. Sistema exibe formulário para cadastro.</li> <li>7. Administrador preenche campos obrigatórios.</li> <li>8. Administrador submete formulário.</li> <li>9. Sistema retorna para listagem de problemas.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de problemas.</li> <li>2. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 7:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Palavra chave cadastrada.

Fonte: (AUTOR, 2018)

#### 4.3.1.8 Manter Entradas e Saídas

A Tabela 4.11 descreve o caso de uso de manter o cadastro, edição e remoção das informações das entradas e saídas de um determinado problema.

Tabela 4.11: Caso de Uso Manter Entrada e Saídas

<b>CASO DE USO 008 – Manter Entradas e Saídas (RF-008)</b>	
<b>Descrição</b>	O administrador cadastra entradas e saídas para um problema, podendo editar e remover.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	CASO DE USO 006. Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de problemas.</li> <li>2. Sistema lista todos problemas já cadastrados.</li> <li>3. Administrador acessa cadastro de problemas.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador seleciona o cadastro de entradas e saídas.</li> <li>6. Sistema exibe formulário para cadastro.</li> <li>7. Administrador preenche campos obrigatórios.</li> <li>8. Administrador submete formulário.</li> <li>9. Sistema retorna para listagem de problemas.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de problemas.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 7:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Entrada e Saída Cadastrada.

Fonte: (AUTOR, 2018)

#### 4.3.1.9 Manter Soluções de Problemas

A Tabela 4.12 descreve o caso de uso de manter o cadastro, edição e remoção das soluções de um determinado problema. As soluções podem ser cadastradas por administradores, professores e/ou alunos.

Tabela 4.12: Caso de Uso Manter Soluções de Problemas

<b>CASO DE USO 009 - Manter Soluções de Problemas (RF-009)</b>	
<b>Descrição</b>	O administrador, aluno e/ou professor pode cadastrar uma solução para um problema, podendo editar e remover.
<b>Ator</b>	Administrador, Aluno, Professor.
<b>Pré-condição</b>	CASO DE USO 006.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Usuário autentica-se no sistema</li> <li>2. Usuário acessa área de soluções de problemas</li> <li>3. Sistema exibe listagens de problemas e editor de soluções</li> <li>4. Usuário cria uma nova solução</li> <li>5. Sistema valida solução</li> </ol>
<b>Fluxo alternativo e exceções</b>	<b>Item 5:</b> <ol style="list-style-type: none"> <li>1. Sistema não valida solução</li> <li>1. Usuário refaz solução</li> </ol>
<b>Pós-condição</b>	Solução cadastrada.

Fonte: (AUTOR, 2018)

#### 4.3.1.10 Manter Listas de Problemas

A Tabela 4.13 descreve o caso de uso de manter o cadastro, edição e remoção das listas de problemas. Essas listas agrupam problemas para serem aplicados a um grupo e/ou aluno.

Tabela 4.13: Caso de Uso Manter Listas de Problemas

<b>CASO DE USO 010 - Manter Listas de Problemas (RF-010)</b>	
<b>Descrição</b>	O administrador e o professor podem cadastrar listas de problemas, estas listas ficam relacionadas a um grupo de alunos, ou apenas a um aluno, podendo editar ou remover.
<b>Autor</b>	Administrador, Professor.
<b>Pré-condição</b>	CASO DE USO 006. Ser um administrador do portal de algoritmos. Ser um professor.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador/Professor acessa listagem de lista de problemas.</li> <li>2. Sistema lista todas as listas de problemas             <ol style="list-style-type: none"> <li>1. Administrador consegue acessar todas as listas de problemas já criadas.</li> <li>2. Professor consegue acessar somente as listas de problemas criadas por ele.</li> </ol> </li> <li>3. Administrador/Professor acessa o cadastro de lista de problemas.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador/Professor preenche campos obrigatórios.</li> <li>6. Administrador/Professor submete formulário.</li> <li>7. Sistema retorna para listagem de lista de problemas.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador/Professor não tem acesso a listagem de administradores.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Lista de Problemas cadastrada.

Fonte: (AUTOR, 2018)

#### 4.3.1.11 Manter Instituições

A Tabela 4.14 descreve o caso de uso de manter o cadastro, edição e remoção das instituições que podem ser utilizadas no cadastro do professor.

Tabela 4.14: Caso de Uso Manter Instituições

<b>CASO DE USO 011 - Manter Instituições (RF-011)</b>	
<b>Descrição</b>	O administrador cobra instituições, podendo editar ou remover. A instituição pode ser selecionada no cadastro do aluno e/ou professor.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de instituições.</li> <li>2. Sistema lista todas as instituições já cadastrados.</li> <li>3. Administrador acessa cadastro de instituições.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador preenche campos obrigatórios.</li> <li>6. Administrador submete formulário.</li> <li>7. Sistema retorna para listagem de instituições</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de instituições.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Instituição cadastrada.

Fonte: (AUTOR, 2018)

#### 4.3.1.12 Manter Grupos

A Tabela 4.15 descreve o caso de uso de manter o cadastro, edição e remoção de grupos. Os grupos servem para agrupar alunos e professores, onde os professores e administradores podem criar listas de problemas específicos para os grupos ou alunos.

Tabela 4.15: Caso de Uso Manter Grupos

<b>CASO DE USO 012 – Manter Grupos (RF-012)</b>	
<b>Descrição</b>	O administrador ou professor cadasstra grupos, associando alunos para o mesmo, podendo editar ou remover.
<b>Autor</b>	Administrador, Professor.
<b>Pré-condição</b>	CASO DE USO 001. CASO DE USO 006. Ser um administrador do portal de algoritmos. Ser um professor.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador/Professor acessa listagem de grupos.</li> <li>2. Sistema lista todos grupos já cadastrados.</li> <li>3. Administrador/Professor acessa cadastro de grupos.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador/Professor preenche campos obrigatórios.</li> <li>6. Administrador/Professor submete formulário.</li> <li>7. Sistema retorna para listagem de grupos.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador/Professor não tem acesso a listagem de grupos.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador/Professor não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Grupo cadastrado.

Fonte: (AUTOR, 2018)

#### 4.3.1.13 Manter Permissões

A Tabela 4.16 descreve o caso de uso de manter o cadastro, edição e remoção das permissões do portal de algoritmos. Essas permissões servem para permitir acesso a determinados cadastros do sistema.

Tabela 4.16: Caso de Uso Manter Permissões

<b>CASO DE USO 013 – Manter Permissões (RF-013)</b>	
<b>Descrição</b>	O administrador cadastra permissões para o software, podendo editar ou remover.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de permissões.</li> <li>2. Sistema lista todas permissões já cadastrados.</li> <li>3. Administrador acessa cadastro de permissões .</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador preenche campos obrigatórios.</li> <li>6. Administrador submete formulário.</li> <li>7. Sistema retorna para permissões.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a permissões.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Permissão cadastrada.

Fonte: (AUTOR, 2018)

#### 4.3.1.14 Manter Grupos de Administradores

A Tabela 4.17 descreve o caso de uso de manter o cadastro, edição e remoção de grupos de administradores. É um agrupador de usuários que tem a permissão de acessar o gerenciamento do portal de algoritmos.

Tabela 4.17: Caso de Uso Manter Grupos de Administradores

<b>CASO DE USO 014 – Manter Grupos de Administradores (RF-014)</b>	
<b>Descrição</b>	O administrador cadastra grupos de administradores, podendo editar ou remover. No cadastro pode ser associado às permissões já cadastradas.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	CASO DE USO 013. Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de grupos de administradores.</li> <li>2. Sistema lista todos os grupos de administradores já cadastrados.</li> <li>3. Administrador acessa cadastro de grupos de administradores.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador preenche campos obrigatórios.</li> <li>6. Administrador submete formulário.</li> <li>7. Sistema retorna para listagem de grupos de administradores.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de grupos de administradores.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Grupos de Administradores cadastrado.

Fonte: (AUTOR, 2018)

#### 4.3.1.15 Manter Países

A Tabela 4.18 descreve o caso de uso de manter o cadastro, edição e remoção de países.

Tabela 4.18: Caso de Uso Manter Países

<b>CASO DE USO 015 – Manter Países (RF-015)</b>	
<b>Descrição</b>	O administrador cadastra países, podendo editar ou remover.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de países.</li> <li>2. Sistema lista todos países já cadastrados.</li> <li>3. Administrador acessa cadastro de países.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador preenche campos obrigatórios.</li> <li>6. Administrador submete formulário.</li> <li>7. Sistema retorna para listagem de países.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de países.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	País cadastrado.

Fonte: (AUTOR, 2018)

#### 4.3.1.16 Manter Estados

A Tabela 4.19 descreve o caso de uso de manter o cadastro, edição e remoção de estados.

Tabela 4.19: Caso de Uso Manter Estados

<b>CASO DE USO 016 - Manter Estados (RF-016)</b>	
<b>Descrição</b>	O administrador cadastra estados, podendo editar ou remover. É necessário ter um país cadastrado para associar ao estado.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	CASO DE USO 015. Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de estados.</li> <li>2. Sistema lista todos estados já cadastrados.</li> <li>3. Administrador acessa cadastro de estados.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador preenche campos obrigatórios.</li> <li>6. Administrador submete formulário.</li> <li>7. Sistema retorna para listagem de estados.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de estados.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Estados cadastrado.

Fonte: (AUTOR, 2018)

#### 4.3.1.17 Manter Cidades

A Tabela 4.20 descreve o caso de uso de manter o cadastro, edição e remoção de cidades.

Tabela 4.20: Caso de Uso Manter Cidades

<b>CASO DE USO 017 - Manter Cidades (RF-017)</b>	
<b>Descrição</b>	O administrador cadastra cidades, podendo editar ou remover. É necessário ter um estado já cadastrado.
<b>Autor</b>	Administrador.
<b>Pré-condição</b>	CASO DE USO 016. Ser um administrador do portal de algoritmos.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Administrador acessa listagem de cidades.</li> <li>2. Sistema lista todos cidades já cadastrados.</li> <li>3. Administrador acessa cadastro de cidades.</li> <li>4. Sistema exibe formulário para cadastro.</li> <li>5. Administrador preenche campos obrigatórios.</li> <li>6. Administrador submete formulário.</li> <li>7. Sistema retorna para listagem de cidades.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<p><b>Item 1:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não tem acesso a listagem de cidades.</li> <li>1. Sistema exibe mensagem de bloqueio.</li> </ol> <p><b>Item 5:</b></p> <ol style="list-style-type: none"> <li>1. Administrador não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Estados cadastrado.

Fonte: (AUTOR, 2018)

#### 4.3.1.18 Manter suas Informações Pessoais

A Tabela 4.21 descreve o caso de uso de cadastro, edição e remoção das informações pessoais do aluno.

Tabela 4.21: Caso de Uso Aluno Manter suas Informações Pessoais

<b>CASO DE USO 018 – Manter suas informações pessoais (RF-018)</b>	
<b>Descrição</b>	O aluno pode se cadastrar, editar e remover suas informações pessoais.
<b>Autor</b>	Aluno.
<b>Pré-condição</b>	Não possui.
<b>Fluxo principal</b>	<ol style="list-style-type: none"> <li>1. Usuário acessa cadastro público do portal.</li> <li>2. Sistema exibe formulário para cadastro.</li> <li>3. Usuário preenche todos os campos obrigatórios.</li> <li>4. Sistema retorna para formulário de login.</li> </ol>
<b>Fluxo alternativo e exceções</b>	<b>Item 3:</b> <ol style="list-style-type: none"> <li>1. Usuário não preenche todos os campos obrigatórios.</li> <li>1. Sistema exibe mensagens de erros.</li> </ol>
<b>Pós-condição</b>	Aluno cadastrado.

Fonte: (AUTOR, 2018)

Foram apresentados os casos de uso que serão utilizados para a evolução do portal de algoritmos. Na próxima Seção serão apresentados os protótipos de interface do portal de algoritmos.

## 4.4 Interfaces Gráficas

Os protótipos exibidos nessa seção procuram seguir os seguintes princípios de usabilidade conforme BENYON (2011):

1. Visibilidade: garante que os elementos sejam visíveis, de forma que as pessoas possam ver quais funções estão disponíveis e o que o sistema está fazendo atualmente.
2. Consistência: mantém consistência no uso de características de *design* e com sistema semelhantes e método-padrão de trabalho.
3. Familiaridade: utiliza linguagem e símbolos com os quais os futuros usuários estão familiarizados.
4. *Affordance*: criar *design* de forma que fique claro para o quê elas servem.
5. Navegação: proporcione suporte para que as pessoas possam se movimentar pelo sistema.
6. Retorno: retorna as informações rapidamente a informação do sistema para os usuários para que elas saibam que efeito suas ações causaram.
7. Estilo: *designs* devem ser elegantes e atraentes.

#### 4.4.1 Cadastro de Aluno

A Figura 4.5 é a interface gráfica de cadastro de um novo aluno.

Figura 4.5: Interface Gráfica de Cadastro de Aluno

The screenshot shows a web browser window with a blue header bar. Below it is a white form titled "Quero me cadastrar". The form contains several input fields: "Nome" and "Sobrenome" (Name and Surname), "E-mail" and "Usuário" (Email and Username), "Senha" and "Repita a senha" (Password and Repeat password). There are dropdown menus for "Instituição" (Institution) and "Cidade" (City). At the bottom of the form are two buttons: "Limpar" (Clear) and "Salvar" (Save).

Fonte: (AUTOR, 2018)

Nessa interface gráfica o aluno preenche todos os campos abaixo:

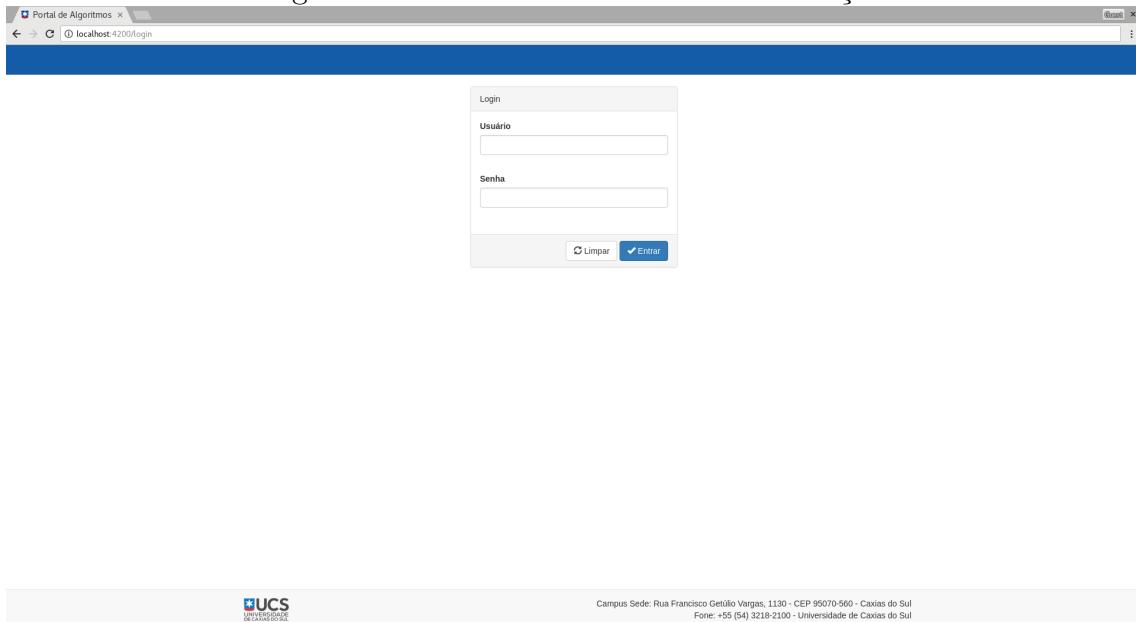
- Nome.
- Sobrenome.
- E-mail.
- Login: o login é seu *username* que será utilizado para acessar o portal de algoritmos.
- Senha.
- Repita a senha: nesse campo faz-se a verificação se a senha digitada anteriormente corresponde a essa.
- Instituição: exibe todas instituições cadastradas pelo administrador.
- Cidade: exibe todas cidades cadastradas pelo administrador.

Após o preenchimento de todos os campos, o alunos clica em “Salvar” e seu cadastro estará realizado.

#### 4.4.2 Autenticação

A Figura 4.6 é a interface gráfica de autenticação.

Figura 4.6: Interface Gráfica de Autenticação



Fonte: (AUTOR, 2018)

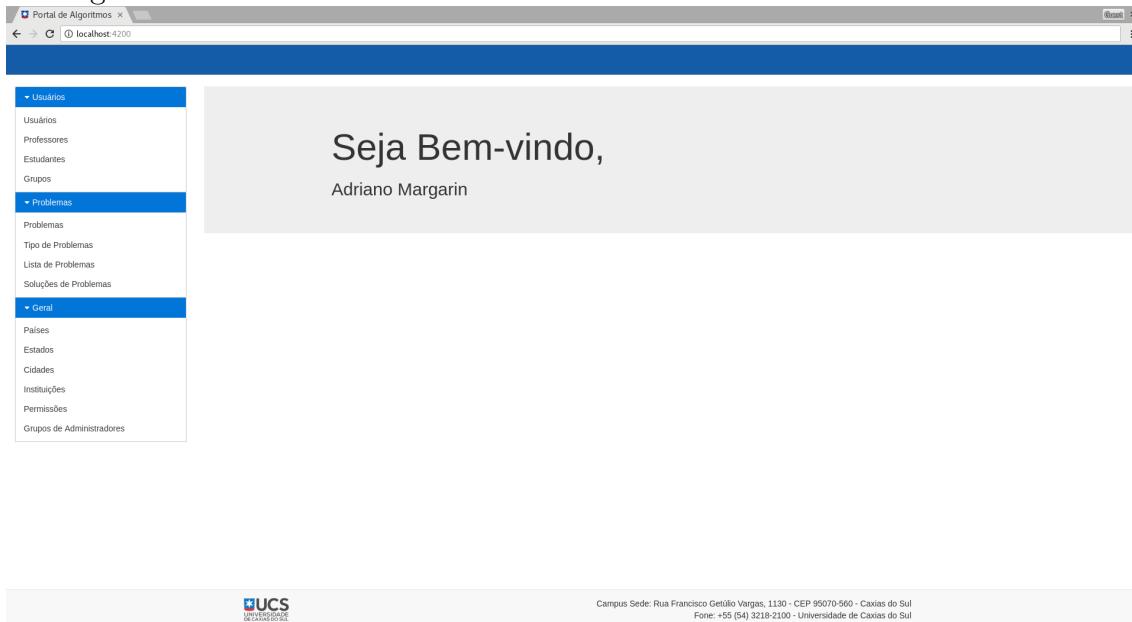
Nessa interface gráfica o aluno, professor ou administrador realiza a autenticação no portal de algoritmos. A autenticação somente é autorizada para usuários ainda ativos no sistema.

O seu funcionamento é simples, o usuário preenche os campos de “Usuário ou E-mail” e “Senha” com as seguintes informações, e-mail ou usuário e senha, o usuário estando ativo no sistema ele será redirecionado para a página inicial do sistema.

#### 4.4.3 Boas Vindas ao Administrador e Professor

A Figura 4.7 é a interface gráfica de Boas Vindas do Administrador e Professor.

Figura 4.7: Interface Gráfica de Boas Vindas do Administrador e Professor



Fonte: (AUTOR, 2018)

Essa interface gráfica é exibida após a autenticação com sucesso do Administrador ou Professor. Nessa interface é exibida uma árvore com links para as demais interfaces gráficas, agrupadas por contextos.

#### 4.4.4 Problemas

A Figura 4.8 é a interface gráfica da listagem de problemas.

Figura 4.8: Interface Gráfica de Problemas

#	Nome	Tipo de Problema	Criado em	Ações
7	Problema 2	Tipo 1		
8	Problema 3	Tipo 2		
9	Problema 4	Tipo 1		
3	Problema 5	Tipo 1		
10	Problema 6	Tipo 2		
6	Problema 1	Tipo 1		

Fonte: (AUTOR, 2018)

Essa interface gráfica possui as seguintes ações:

- Pesquisa livre.
  - O administrador pode realizar a filtragem por qualquer palavra.
- Novo Tipo de Problema.
  - Selezionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.9.
- Novo Problema.
  - Selezionando essa ação, o adminsitrador é direcionado para a interface gráfica da Figura 4.10.
- Editar Problema.
  - Selezionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.10.
- Remover Problema.
  - Selezionando essa ação, o administrador remove o problema.

#### 4.4.5 Tipo de Problema

A Figura 4.9 é a interface gráfica de listagem de tipos de problemas.

Figura 4.9: Interface Gráfica de Listagem Tipos de Problemas

#	Nome
2	Tipo 1
6	Tipo 2

**Ações**

Fonte: (AUTOR, 2018)

Nessa interface gráfica o administrador visualiza todas os Tipos de Problemas já cadastrados, podendo adicionar, editar e remover.

#### 4.4.6 Cadastro e Edição de Problema

A Figura 4.10 é a interface gráfica de cadastro e/ou edição de um novo problema.

Figura 4.10: Interface Gráfica de Novo Problema

A interface gráfica de Novo Problema, intitulada "Editando Problema #6", está dividida em três abas: "Problema", "Palavras Chaves" e "Dados de Testes". A aba "Problema" é a ativa, mostrando campos para preencher: "Tipo de Problema" (selecionado "Tipo 1"), "Nome" (campo com placeholder "Problema 1"), "Nível" (selecionado "Médio") e "Custo" (campo com placeholder "2"). Abaixo desses campos, há uma seção "Dica" com um campo de texto vazio. A seção "Descrição" contém uma barra de ferramentas com opções de formatação (Normal, Sans Serif, B, I, U, A, etc.) e uma área de texto com placeholder "Descrição do problema. Esse texto pode ser formatado. Adicionar imagens. Adicionar trechos de códigos.". No rodapé da interface, há botões "× Fechar" e "Salvar".

Fonte: (AUTOR, 2018)

A interface está dividida em três abas, Problema, Palavras Chave e Dados de Testes, cada uma dessas abas estão descritas abaixo:

- Problema.
  - O cadastro ou edição do problema consiste em preencher os campos, “Tipo de Problema”, “Nome do Problema”, “Dica” e “Nível”, o “Custo” é correspondente ao menor custo de alguma solução daquele problema.
- Palavras Chaves.
  - O cadastro de palavras-chave é descrito na Seção 4.4.7
- Dados de Testes.
  - O cadastro de dados de testes é descrito na Seção 4.4.8

Toda a ação nessa interface deve ser gravada ao clicar no botão “Salvar”.

#### 4.4.7 Palavras-chave

A Figura 4.11 é a interface gráfica de cadastro e/ou edição de palavras-chave de um problema.

Figura 4.11: Interface Gráfica de Palavras Chaves

A interface gráfica de Palavras Chaves, intitulada "Editando Problema #6", exibe três guias: "Problema" (ativo), "Palavras Chaves" (selecionado) e "Dados de Testes". A seção "Palavras Chaves" contém uma barra de pesquisa e uma lista de palavras disponíveis ("Palavras Chaves Disponíveis") com "Palavra 2" e "Palavra 4". À direita, uma lista de palavras usadas ("Palavras Chaves Usadas") mostra "Palavra 1" e "Palavra 3". Botões de navegação (setas) permitem mover palavras entre as listas. Um botão "Adicionar" está posicionado ao lado da lista de palavras disponíveis. No rodapé, há botões "Fechar" e "Salvar".

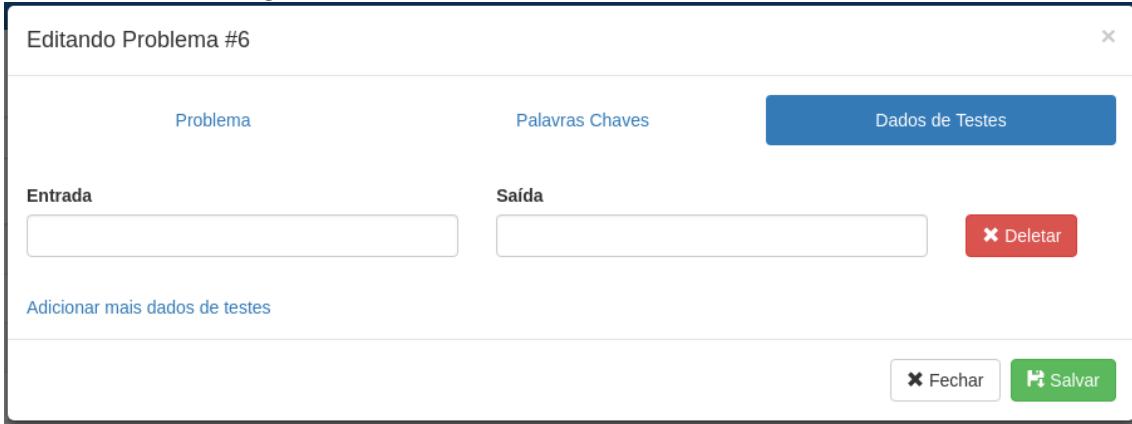
Fonte: (AUTOR, 2018)

Nessa interface o administrador preenche o campo “Palavra-chave” e clica em “Adicionar”. Fazendo essa ação ele associa a nova palavra-chave ao problema que ele está cadastrando ou editando. O administrador pode também editar ou remover uma palavra-chave já cadastrada, para isso é preciso selecionar um das ações, “Editar” ou “Remover”.

#### 4.4.8 Dados de Testes

A Figura 4.12 é a interface gráfica de cadastro e/ou edição de dados de testes de um problema.

Figura 4.12: Interface Gráfica de Dados de Testes



A interface gráfica de dados de testes é intitulada "Editando Problema #6". Ela contém três guias: "Problema", "Palavras Chaves" e "Dados de Testes", com o último selecionado. No topo, há botões para "Entrada" e "Saída", cada um com um campo de texto vazio. À direita do campo de "Saída" está um botão vermelho com a opção "Deletar". Abaixo dos campos, há uma barra com o link "Adicionar mais dados de testes". No lado direito da interface, estão os botões "Fechar" (branco com ícone de fechamento) e "Salvar" (verde com ícone de disco).

Fonte: (AUTOR, 2018)

Nessa interface o administrador preenche os campos “Entrada” e “Saída” e clica em “Adicionar”, fazendo essa ação ele associa o novo dado de teste ao problema que ele está cadastrando ou editando. O administrador pode também editar ou remover um dado de teste já cadastrado, para isso é preciso selecionar um das ações, “Editar” ou “Remover”. Um problema pode ter apenas dados de testes de “Entrada” ou somente de “Saída”.

#### 4.4.9 Visualização de Solução de Problema

A Figura 4.13 é a interface gráfica de visualização de uma solução de um problema.

Figura 4.13: Interface Gráfica de Visualização de Solução do Problema

The screenshot shows a window titled "Editando Solução de Problema #1". The window contains several input fields and dropdown menus:

- Nome:** A text input field containing "solução 1".
- Usuário:** A dropdown menu showing "pedro".
- Problema:** A dropdown menu showing "Problema 1".
- Custo:** An empty text input field.
- Resultado:** An empty text input field.
- Solução:** A large text area containing the text "aaaaaaaaaa".

At the bottom right of the window are two buttons: "× Fechar" (Close) and a green "Salvar" (Save) button with a disk icon.

Fonte: (AUTOR, 2018)

#### 4.4.10 Lista de Grupos

A Figura 4.14 é a interface gráfica de listagem dos grupos já cadastrados. Essa listagem corresponde a todos os grupos, caso seja um administrador com total acesso, ou apenas aos grupos cadastrados por um professor, sendo que o professor somente terá na listagem os grupos que ele cadastrou.

Figura 4.14: Interface Gráfica de Grupos

Fonte: (AUTOR, 2018)

Nessa interface gráfica possuímos as seguintes ações:

- Pesquisa Livre.
  - O administrador pode realizar a pesquisa por qualquer palavra.
- Novo Grupo.
  - Selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.15
- Editar Grupo.
  - Selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.15
- Remover Grupo.
  - Selecionando essa ação, o administrador remove somente o grupo, sendo assim, não remove problemas, soluções e/ou usuários.

#### 4.4.11 Cadastro e Edição de Grupo

A Figura 4.15 é a interface gráfica de cadastro e edição de um grupo.

Figura 4.15: Interface Gráfica de Cadastro de Grupo

Grupo	Paticipantes	Lista de Problemas
<div><p>Lista de Problemas Disponíveis</p><div><input type="text"/> <span>Q</span></div><div><span>^</span><span>^</span><span>v</span><span>v</span></div><div><p>Listas:</p><ul style="list-style-type: none"><li>Listas 4</li><li>Listas 2</li><li>Listas 6</li><li>Listas 7</li><li>Listas 3</li><li>Listas 9</li></ul><div style="margin-top: 10px;">▼</div></div></div> <div><p>Lista de Problemas Usados</p><div><input type="text"/> <span>Q</span></div><div><span>^</span><span>^</span><span>v</span><span>v</span></div></div>		

Fonte: (AUTOR, 2018)

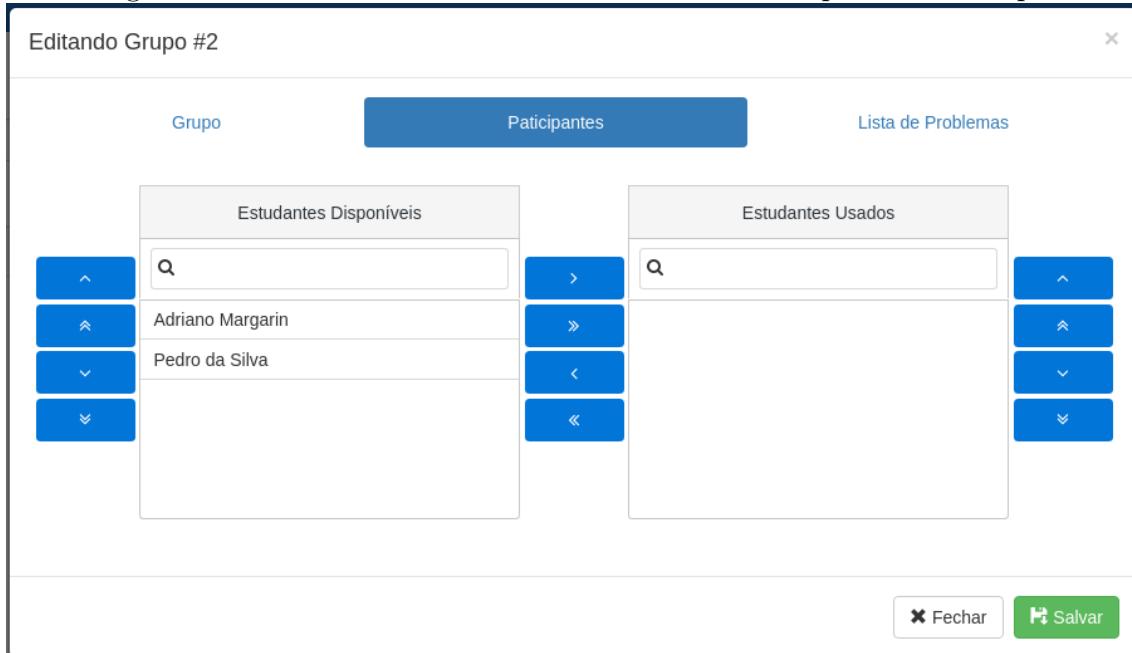
A interface gráfica está dividida em três abas, Grupo, Participantes, Lista de Problemas sendo que cada uma dessas abas estão descritas abaixo:

- Grupo.
    - Para o cadastro de um novo grupo, basta preencher os campos, Nome e Descrição.

#### 4.4.12 Adicionar Participantes

A Figura 4.16 é a interface gráfica de associação de participantes em um grupo.

Figura 4.16: Interface Gráfica de Cadastro de Participantes no Grupo



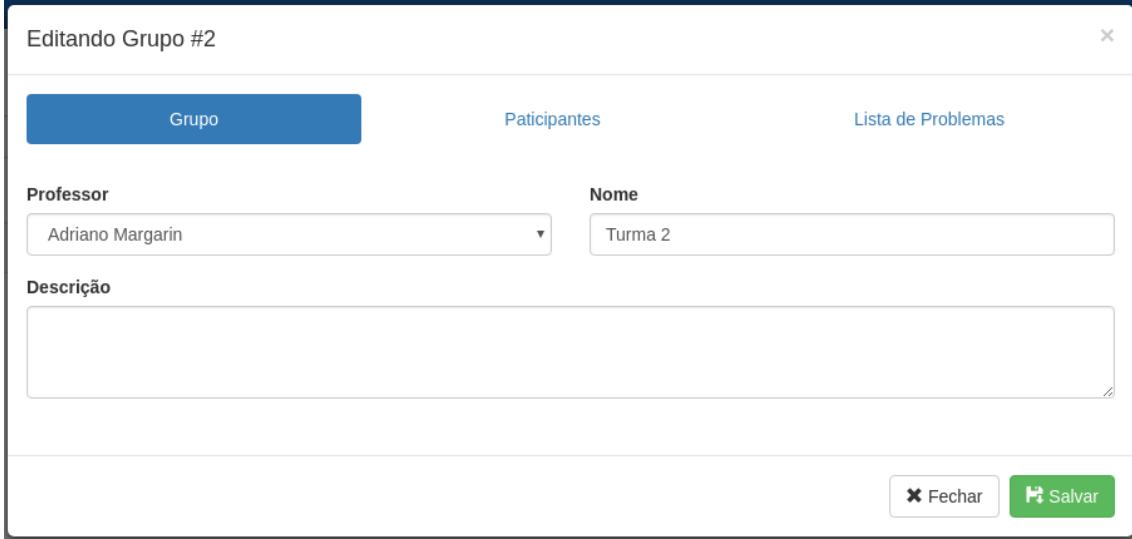
Fonte: (AUTOR, 2018)

Utilizando os botões centrais, é possível adicionar ou remover participantes, podendo-se adicionar ou remover todos ou um e/ou mais por vez.

#### 4.4.13 Adicionar Lista de Problemas

A Figura 4.17 é a interface gráfica de associação de lista de problemas em um grupo.

Figura 4.17: Interface Gráfica de Cadastro de Lista de Problemas no Grupo



A interface gráfica de cadastro de lista de problemas no grupo, intitulada "Editando Grupo #2". Ela contém três seções principais: "Grupo" (destacada com um fundo azul), "Paticipantes" e "Lista de Problemas". A seção "Grupo" inclui campos para "Professor" (Adriano Margarin) e "Nome" (Turma 2). A seção "Descrição" é um campo de texto vazio. No lado direito da interface, há botões para "Fechar" e "Salvar".

Fonte: (AUTOR, 2018)

Utilizando os botões centrais, é possível adicionar ou remover listas de problemas, podendo-se adicionar ou remover todos ou um e/ou mais por vez.

#### 4.4.14 Lista de Alunos

A Figura 4.18 é a interface gráfica da listagem de alunos.

Figura 4.18: Interface Gráfica de Listagem de Alunos

#	Usuário	Ações
1	adrianomargarin	
2	pedro	

Fonte: (AUTOR, 2018)

Nessa interface gráfica encontramos as seguintes ações:

- Pesquisa Livre: o administrador pode realizar a pesquisa por qualquer palavra.
- Novo Aluno: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.19
- Editar Aluno: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.19
- Remover Aluno: selecionando essa ação, o administrador remove o aluno.

#### 4.4.15 Cadastro de Alunos

A Figura 4.19 é a interface gráfica do cadastro e edição de alunos.

Figura 4.19: Interface Gráfica de Cadastro de Alunos



A interface gráfica para o cadastro e edição de alunos, intitulada "Editando Estudante #1". A interface contém campos para Nome, Sobrenome, E-mail, Usuário, Senha, Repita a senha, Instituição e Cidade. No lado direito da interface, há botões para Fechar e Salvar.

Nome	Sobrenome
<input type="text"/>	<input type="text"/>

E-mail	Usuário
<input type="text"/>	<input type="text"/>

Senha	Repita a senha
<input type="password"/>	<input type="password"/>

Instituição	Cidade
<input type="text"/>	<input type="text"/>

Fonte: (AUTOR, 2018)

A única informação que é necessário preencher é selecionar um Usuário sendo que esse usuário ficará associado ao cadastro do aluno.

#### 4.4.16 Listas de Problemas

A Figura 4.20 é a interface da listagem de listas de problemas. Essa interface apresenta todas as listas de problemas já cadastradas no portal de algoritmos.

Figura 4.20: Interface Gráfica de Lista de Problemas

#	Nome	Ações
4	Lista 4	
2	Lista 2	
6	Lista 6	
7	Lista 7	
3	Lista 3	
8	Lista 8	
5	Lista 5	
9	Lista 9	
1	Lista 1	
10	Lista 10	

Fonte: (AUTOR, 2018)

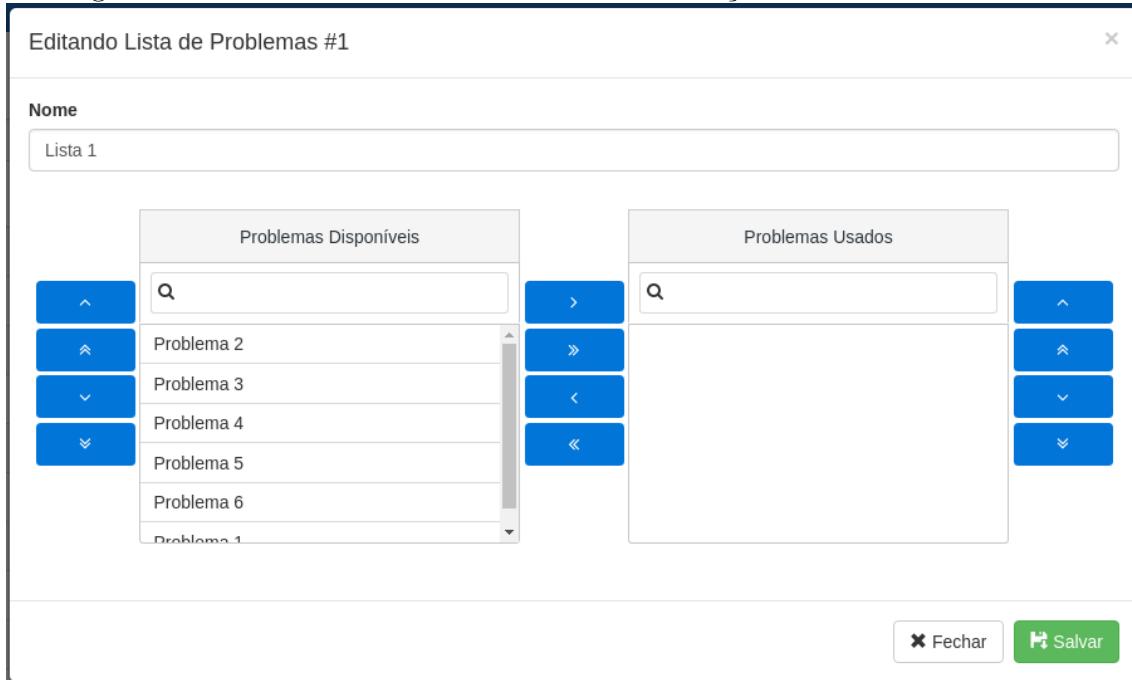
Nessa interface gráfica temos as seguintes ações:

- Pesquisa Livre: o administrador pode realizar a pesquisa por qualquer palavra.
- Nova Lista: selecionado essa ação, o administrador é direcionado para a interface gráfica da Figura 4.21
- Editar Lista: selecionado essa ação, o administrador é direcionado para a interface gráfica da Figura 4.21
- Remover Lista: selecionando essa ação, o administrador remove uma lista.

#### 4.4.17 Cadastro e Edição de Lista de Problemas

A Figura 4.21 é a interface de cadastro e edição de lista de problemas.

Figura 4.21: Interface Gráfica de Cadastro e Edição de Lista de Problemas



Fonte: (AUTOR, 2018)

Para cadastrar ou editar, o administrador deve preencher um nome para lista e selecionar os problemas conforme as ações disponíveis na interface.

#### 4.4.18 Lista de Usuários

A Figura 4.22 é a interface gráfica da listagem dos usuários. Essa interface lista todos os usuários cadastrados no portal de algoritmos, incluindo os inativos.

Figura 4.22: Interface Gráfica de Lista de Usuários

#	Usuário	E-mail	Nome	Sobrenome	Criado em	Último login em	Está ativo?	Ações
8	pedro	pedro@ucs.br	Pedro	da Silva			Sim	
5	adrianomargarin	amargarin@ucs.br	Adriano	Margarin			Sim	

Fonte: (AUTOR, 2018)

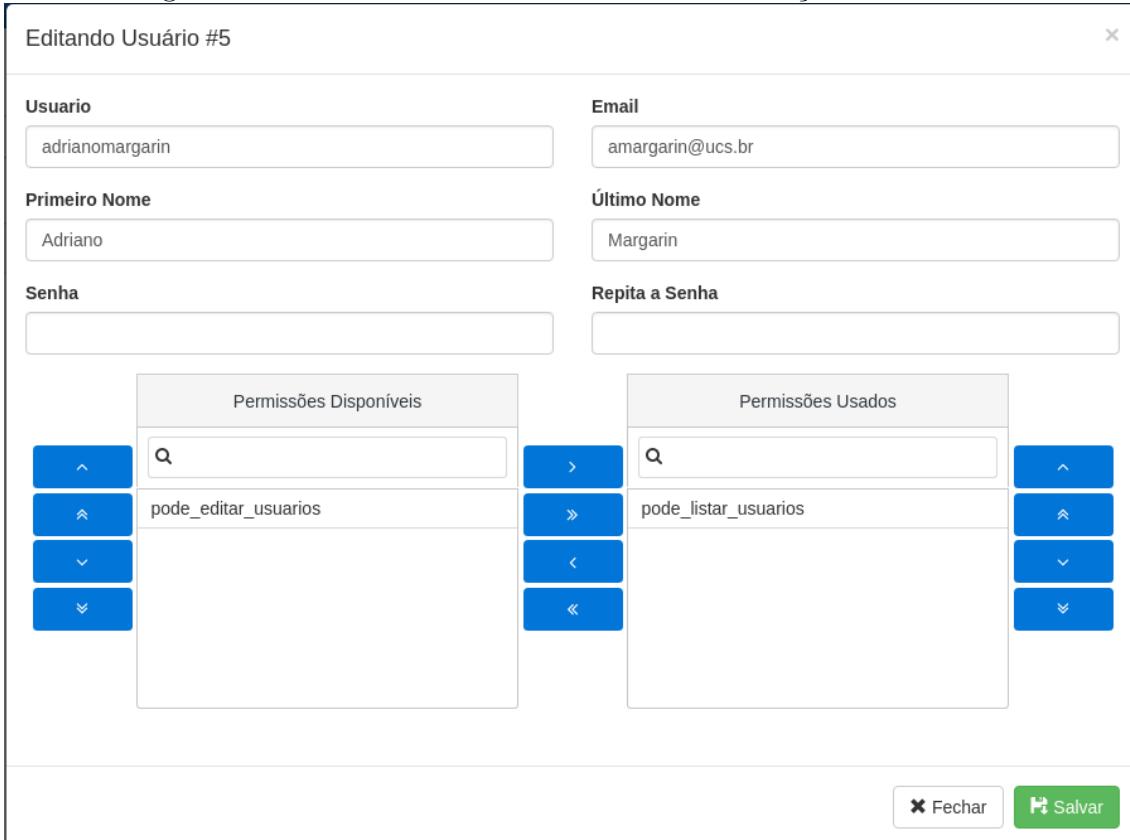
Nessa interface gráfica temos as seguintes ações:

- Pesquisa Livre: o administrador pode realizar a pesquisa por qualquer palavra.
- Novo Usuário: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.23.
- Editar Usuário: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.23.
- Remover Usuário: selecionando essa ação, o administrador remove um usuário.

#### 4.4.19 Cadastro e Edição de Usuários

A Figura 4.23 é a interface gráfica do cadastro e edição dos usuários.

Figura 4.23: Interface Gráfica de Cadastro e Edição de Usuários



A interface gráfica para o cadastro e edição de usuários, intitulada "Editando Usuário #5". A interface é dividida em campos de texto e uma seção central com listas suspensas para permissões.

- Campos de Texto:**
  - Usuario:** adrianomargarin
  - Email:** amargarin@ucs.br
  - Primeiro Nome:** Adriano
  - Último Nome:** Margarin
  - Senha:** (campo vazio)
  - Repita a Senha:** (campo vazio)
- Permissões:** A interface mostra duas listas suspensas para gerenciar permissões:
  - Permissões Disponíveis:** Contém o item "pode\_editar\_usuarios".
  - Permissões Usados:** Contém o item "pode\_listar\_usuarios".
 As listas são controladas por botões de navegação (setas para cima, para baixo, para esquerda, para direita) e uma barra de pesquisa.
- Botões de Ação:** No lado direito da interface, há dois botões: "Fechar" (branco com ícone de fechamento) e "Salvar" (verde com ícone de disco).

Fonte: (AUTOR, 2018)

O administrador preenche todos os campos do cadastro e clica em “Salvar”. Após o administrador cadastrar ou editar o aluno o sistema retorna para listagem de usuários.

#### 4.4.20 Lista de Professores

A Figura 4.24 é a interface gráfica da listagem dos professores.

Figura 4.24: Interface Gráfica de Lista de Professores

#	Usuário	Instituição	Ações
1	adrianomargarin	Universidade de Caxias do Sul	

Fonte: (AUTOR, 2018)

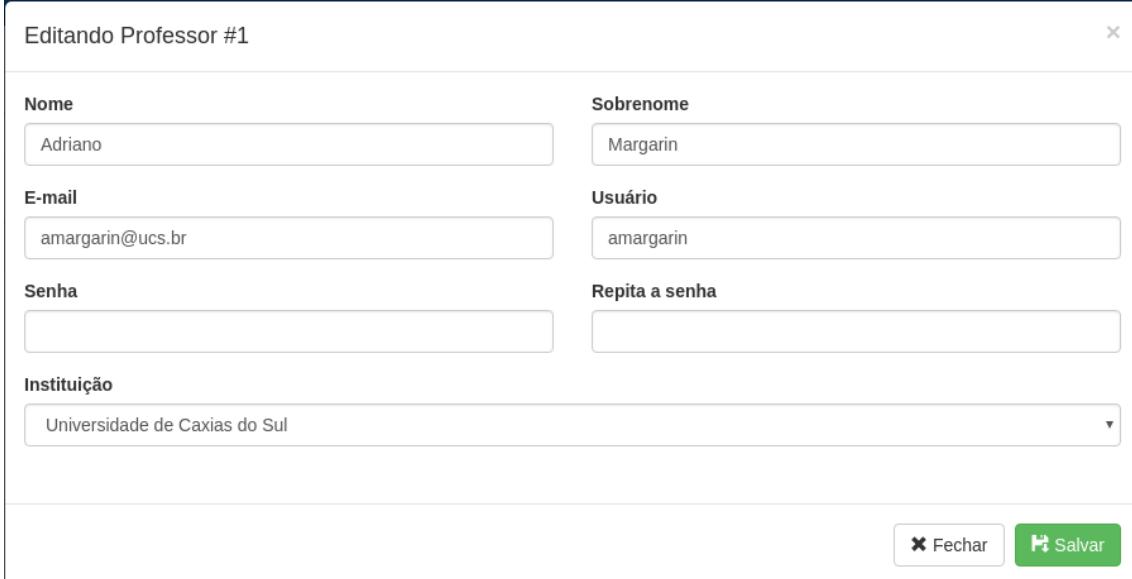
Nessa interface gráfica temos as seguintes ações:

- Pesquisa Livre: o administrador pode realizar a pesquisa por qualquer palavra.
- Novo Professor: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.25.
- Editar Professor: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.25.
- Remover Professor: selecionando essa ação, o administrador remove um Professor.

#### 4.4.21 Cadastro e Edição de Professores

A Figura 4.25 é a interface gráfica do cadastro e edição dos professores.

Figura 4.25: Interface Gráfica de Cadastro e Edição de Professores



A interface gráfica para o cadastro e edição de professores, intitulada "Editando Professor #1". A interface contém campos para Nome (Adriano), Sobrenome (Margarin), E-mail (amargarin@ucs.br), Usuário (amargarin), Senha (campo vazio), Repita a senha (campo vazio), e Instituição (Universidade de Caxias do Sul). No lado direito da interface, há botões para Fechar (com ícone de fechamento) e Salvar (em destaque com ícone de disco).

Fonte: (AUTOR, 2018)

O administrador seleciona um usuário para cadastrar como professor, podendo também associar a instituição que ele pertence. Após preencher todo o cadastro, o administrador clica em “Salvar” e o sistema direciona para listagem de professores.

#### 4.4.22 Lista de Instituições

A Figura 4.26 é a interface gráfica da listagem das Instituições.

Figura 4.26: Interface Gráfica de Lista de Instituições

#	Nome	Sigla	Cidade	Estado	País	Ações
11	Universidade de Caxias do Sul	UCS	Caxias do Sul	Rio Grande do Sul/RS	Brasil/BR	
17	Universidade de Caxias do Sul	UCS	Farroupilha	Rio Grande do Sul/RS	Brasil/BR	
18	Universidade Federal de Santa Catarina	UFSC	Florianópolis	Santa Catarina/SC	Brasil/BR	

Fonte: (AUTOR, 2018)

Nessa interface gráfica temos as seguintes ações:

- Pesquisa Livre: o administrador pode realizar a pesquisa por qualquer palavra.
- Novo Instituição: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.27.
- Editar Instituição: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.27.
- Remover Instituição: selecionando essa ação, o administrador remove uma Instituição.

#### 4.4.23 Cadastro e Edição de Instituições

A Figura 4.27 é a interface gráfica do cadastro e edição de instituições.

Figura 4.27: Interface Gráfica de Cadastro e Edição de Instituições



A interface gráfica para o cadastro e edição de instituições, intitulada "Editando Instituição #11". Ela contém campos para "Cidade" (selecionado como "Caxias do Sul"), "Nome" ("Universidade de Caxias do Sul") e "Sigla" ("UCS"). No lado direito, há botões para "Fechar" e "Salvar".

Cidade	Nome	Sigla
Caxias do Sul	Universidade de Caxias do Sul	UCS

**Botões:**  
Fechar (branco com ícone de fechamento) | Salvar (verde com ícone de salvar)

Fonte: (AUTOR, 2018)

O administrador preenche um nome e seleciona uma cidade para cadastrar ou editar uma instituição. Após preencher as informações, o administrador clica em “Salvar” e o sistema direciona para listagem das instituições.

#### 4.4.24 Lista de Permissões

A Figura 4.26 é a interface gráfica da listagem das Permissões.

Figura 4.28: Interface Gráfica de Lista de Permissões

#	Nome	Descrição	Ações
1	pode_listar_usuarios	Permite listar todos os usuários.	
4	pode_editar_usuarios	Permite editar todos usuários.	

Fonte: (AUTOR, 2018)

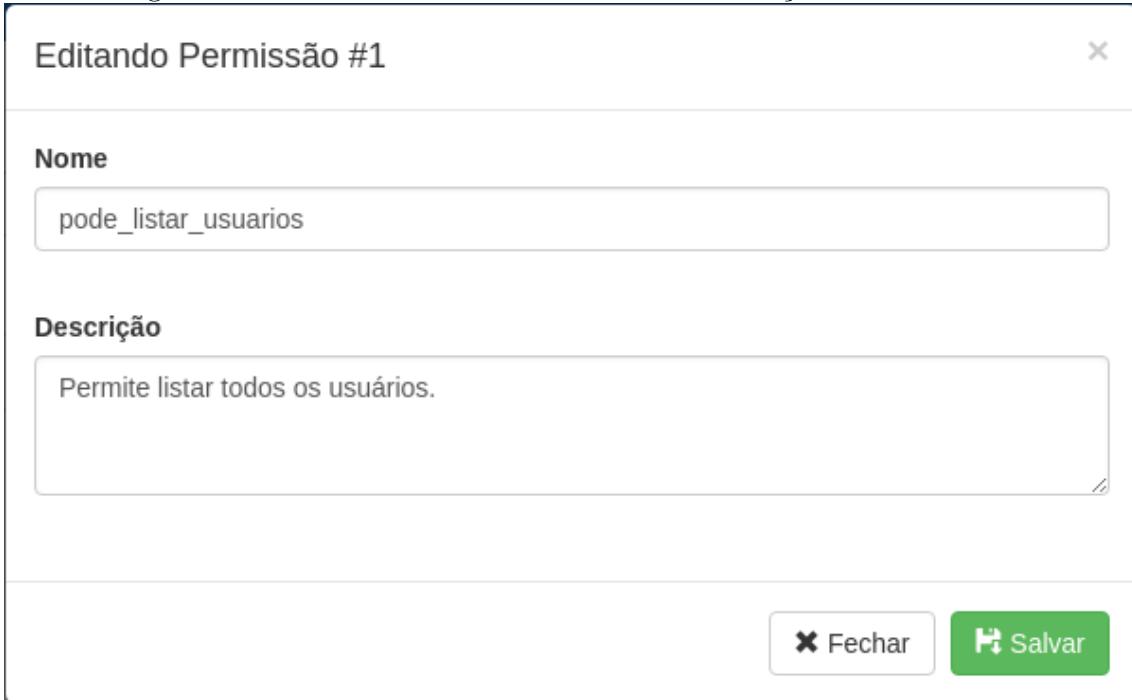
Nessa interface gráfica temos as seguintes ações:

- Pesquisa Livre.
  - O administrador pode realizar a pesquisa por qualquer palavra.
- Nova Permissões.
  - Selezionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.29.
- Editar Permissões.
  - Selezionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.29.
- Remover Permissões.
  - Selezionando essa ação, o administrador remove uma Permissão.

#### 4.4.25 Cadastro e Edição de Permissões

A Figura 4.29 é a interface gráfica do cadastro e edição de permissões.

Figura 4.29: Interface Gráfica de Cadastro e Edição de Permissões



A interface gráfica para o cadastro e edição de permissões, intitulada "Editando Permissão #1". Ela contém campos para "Nome" (pode\_listar\_usuarios) e "Descrição" (Permite listar todos os usuários). No lado direito, há botões para "Fechar" e "Salvar".

Editando Permissão #1	
<b>Nome</b>	pode_listar_usuarios
<b>Descrição</b>	Permite listar todos os usuários.
<b>Salvar</b>	

Fonte: (AUTOR, 2018)

O administrador preenche um nome e uma descrição para cadastrar uma permissão. Após o preenchimento o administrador clica em “Salvar” e o sistema direciona para a listagem das permissões.

#### 4.4.26 Lista de Grupos de Administradores

A Figura 4.30 é a interface gráfica da listagem das Grupos de Administradores.

Figura 4.30: Interface Gráfica de Lista de Grupos de Administradores

#	Nome	Ações
1	Professores Bloco 71	
2	Professores Bloco D	

Fonte: (AUTOR, 2018)

Nessa interface gráfica temos as seguintes ações:

- Pesquisa Livre: o administrador pode realizar a pesquisa por qualquer palavra.
- Novo Grupo: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.31.
- Editar Grupo: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.31.
- Remover Grupo: selecionando essa ação, o administrador remove um Grupo de Administrador.

#### 4.4.27 Cadastro e Edição de Grupos de Administradores

A Figura 4.31 é a interface gráfica do cadastro e edição de permissões.

Figura 4.31: Interface Gráfica de Cadastro e Edição de Grupos de Administradores

Nome  
Professores Bloco 71

Permissões Disponíveis	Permissões Usados
<input type="text"/>	<input type="text"/> pode_listar_usuarios pode_editar_usuarios

Usuários Disponíveis	Usuários Usados
<input type="text"/>	<input type="text"/> pedro adrianomargarin

**Fechar** **Salvar**

Fonte: (AUTOR, 2018)

O administrador preenche um nome, uma descrição e associa permissões a esse grupo. Após preencher as informações o administrador clica em “Salvar” e o sistema direciona para a listagem dos grupos de administradores.

#### 4.4.28 Lista de Países, Estados e Cidades

A Figura 4.32 é a interface gráfica da listagem de países, estados e cidades cadastradas no portal de algoritmos.

Figura 4.32: Interface Gráfica de Listagem de Países, Estados e Cidades

#	Nome	Sigla	Ações
36	Argentina	AR	
88	Uruguai	UR	
1	Brasil	BR	

Fonte: (AUTOR, 2018)

Nessa interface encontramos as seguintes ações:

- Pesquisa Livre: o administrador pode realizar a pesquisa por qualquer palavra.
- Novo País: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.33.
- Editar País: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.33.
- Remover País: selecionando essa ação, o administrador remove um país e todos os estados e cidades associados a ele.
- Novo Estado: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.34.
- Editar Estado: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.34.
- Remover Estado: selecionando essa ação, o administrador remove um estado e todas as cidades associadas a ele.
- Nova Cidade: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.35.
- Editar Cidade: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.35.
- Remover Cidade: selecionado essa ação, o administrador remove uma cidade.

#### 4.4.29 Cadastro e Edição de País

A Figura 4.33 é a interface gráfica do cadastro e edição de um País.

Figura 4.33: Interface Gráfica de Cadastro e Edição de País

A interface gráfica para edição de uma cidade, intitulada "Editando Cidade #4". Ela contém campos para "Estado" (selecionado como "Rio Grande do Sul") e "Nome" (digitar "Caxias do Sul"). Abaixo dos campos, há botões para "Fechar" (branco com ícone de fechamento) e "Salvar" (verde com ícone de salvar).

Fonte: (AUTOR, 2018)

O administrador preenche o nome do País e sua Abreviação e clica em “Salvar”, após retorna para listagem de países, estados e cidades.

#### 4.4.30 Cadastro e Edição de Estado

A Figura 4.34 é a interface gráfica do cadastro e edição de um Estado.

Figura 4.34: Interface Gráfica de Cadastro e Edição de Estado



Figura 4.34: Interface Gráfica de Cadastro e Edição de Estado

Editando Estado #6

**País**

Brasil

**Nome**

Rio Grande do Sul

**Sigla**

RS

**Fechar** **Salvar**

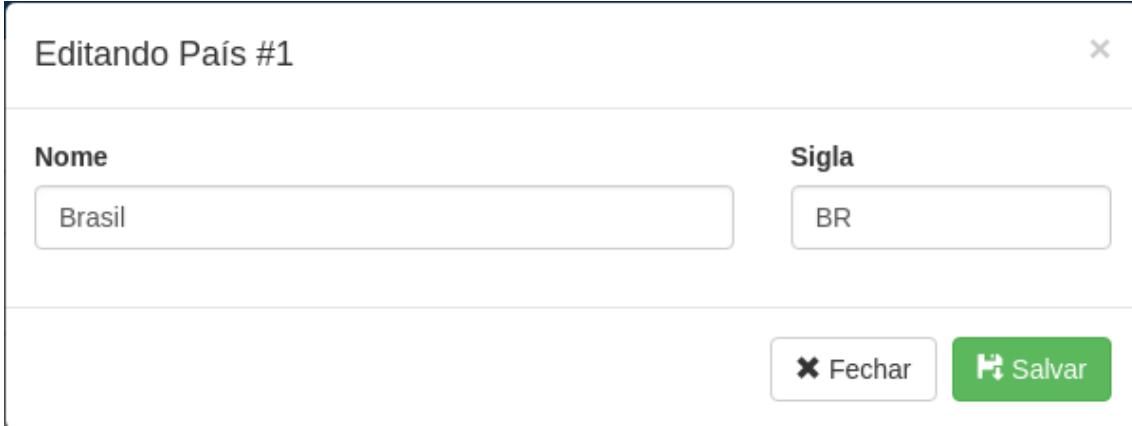
Fonte: (AUTOR, 2018)

O administrador seleciona um País, preenche o nome do Estado e sua Abreviação e clica em “Salvar”, após retorna para listagem de países, estados e cidades.

#### 4.4.31 Cadastro e Edição de Cidade

A Figura 4.34 é a interface gráfica do cadastro e edição de uma Cidade.

Figura 4.35: Interface Gráfica de Cadastro e Edição de Cidade



A interface gráfica para o cadastro e edição de uma Cidade, intitulada "Editando País #1". A interface contém campos para "Nome" (Brasil) e "Sigla" (BR). No lado direito, há botões para "Fechar" e "Salvar".

Nome	Sigla
Brasil	BR

**Fechar** **Salvar**

Fonte: (AUTOR, 2018)

O administrador seleciona um País, seleciona um Estado e preenche o nome da cidade e clica em “Salvar”, após retorna para listagem de países, estados e cidades.

Nesta seção foram apresentados todos os protótipos de interfaces gráficas, a seguir veremos os diagramas de robustez na qual é demonstrada a ligação entre as interfaces e as classes de domínio.

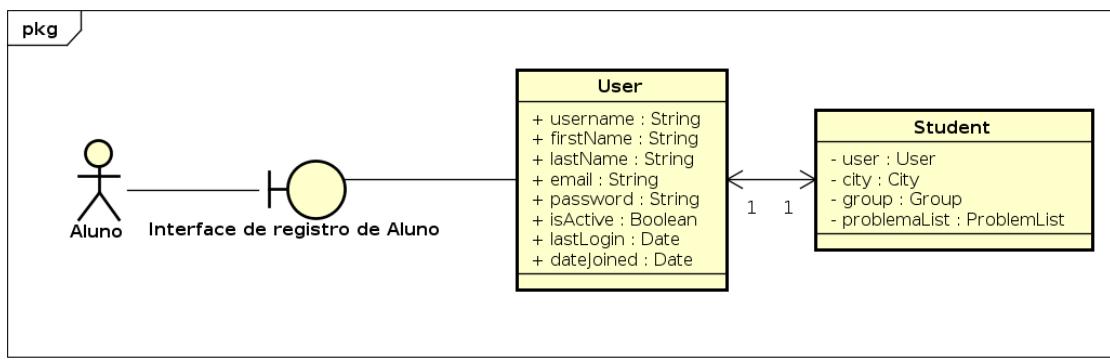
## 4.5 Diagramas de Robustez

Diagramas de Robustez tem por objetivo demonstrar a ligação dos casos de uso com as interfaces gráficas e classes de domínio dentro do contexto da modelagem do *software*.

### 4.5.1 Cadastro de Aluno

A Figura 4.36 demonstra as associações das classes de domínio na interface gráfica de cadastro de alunos.

Figura 4.36: Diagrama de Robustez de Cadastro de Aluno

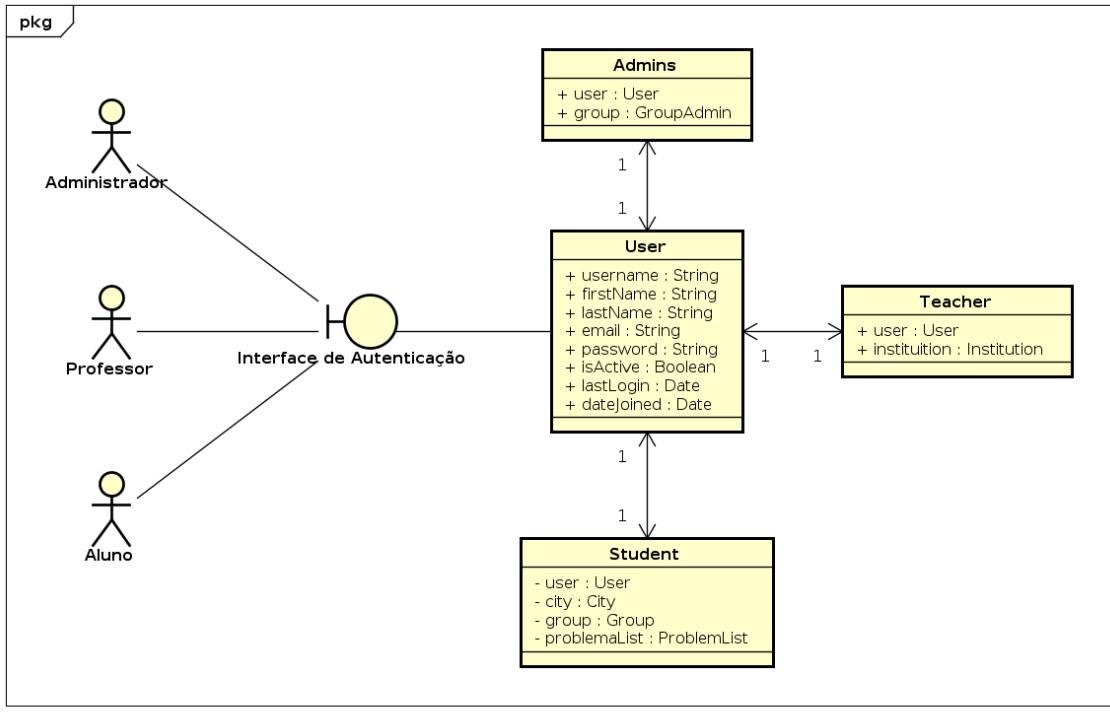


Fonte: (AUTOR, 2018)

#### 4.5.2 Autenticação

A Figura 4.37 demonstra as associações das classes de domínio na interface gráfica de autenticação.

Figura 4.37: Diagrama de Robustez de Autenticação



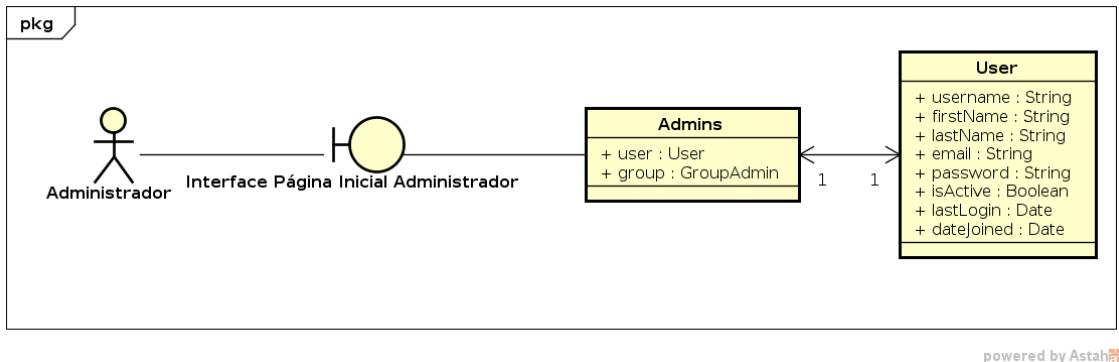
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.3 Boas Vindas do Administrador e do Professor

A Figura 4.38 demonstra as associações das classes de domínio na interface gráfica de boas vindas.

Figura 4.38: Diagrama de Robustez de Boas Vindas Administrador

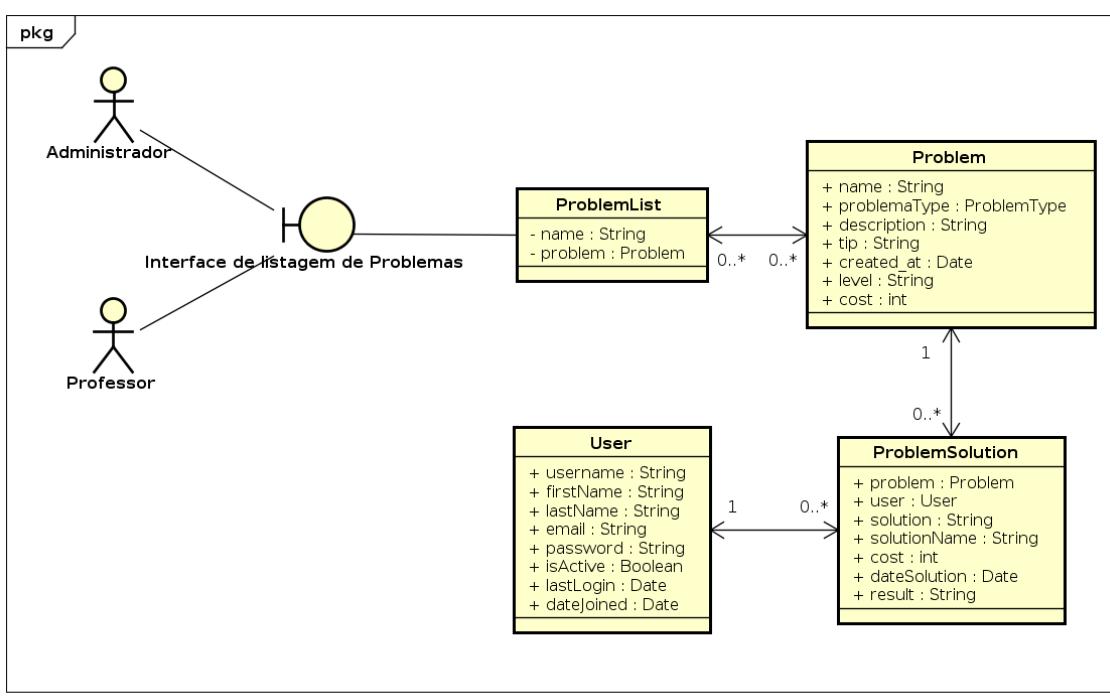


Fonte: (AUTOR, 2018)

#### 4.5.4 Problemas

A Figura 4.39 demonstra as associações das classes de domínio na interface gráfica de listagem de problemas.

Figura 4.39: Diagrama de Robustez de Problemas

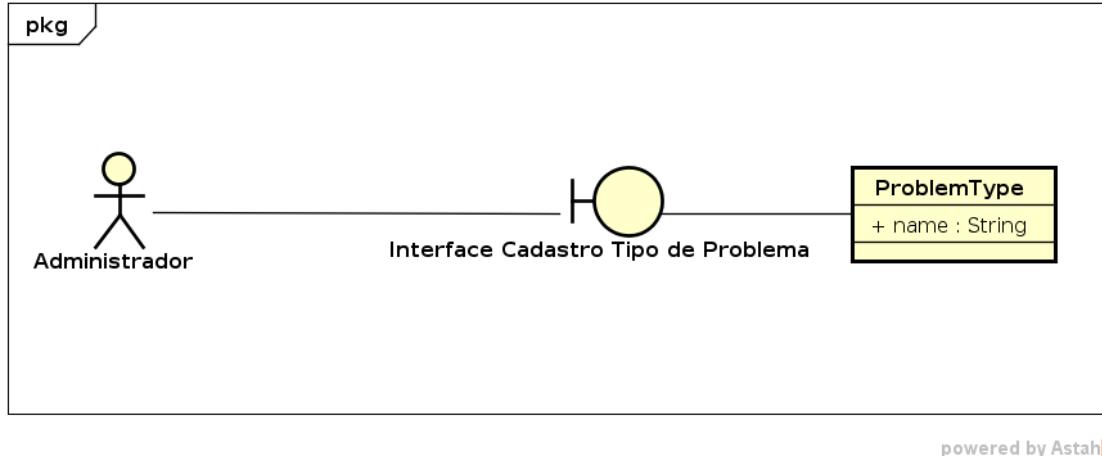


Fonte: (AUTOR, 2018)

#### 4.5.5 Tipo de Problema

A Figura 4.40 demonstra as associações das classes de domínio na interface gráfica de novo tipo de problemas.

Figura 4.40: Diagrama de Robustez de Tipo de Problema



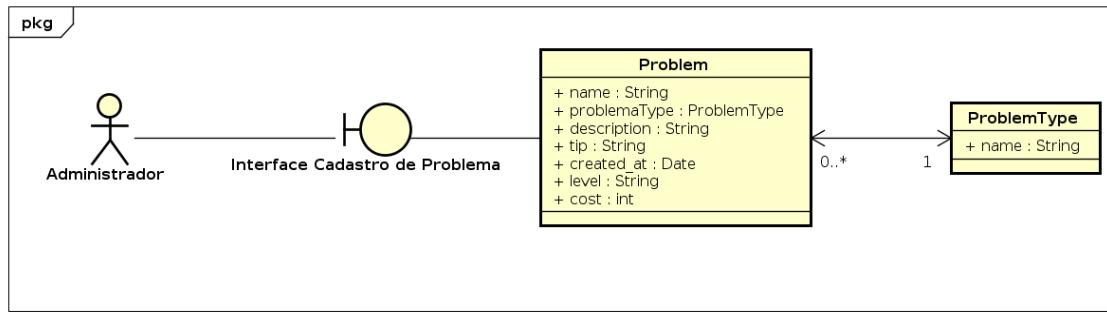
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.6 Cadastro e Edição de Problema

A Figura 4.41 demonstra as associações das classes de domínio na interface gráfica de cadastro de novo problema.

Figura 4.41: Diagrama de Robustez de Cadastro e Edição de Problema



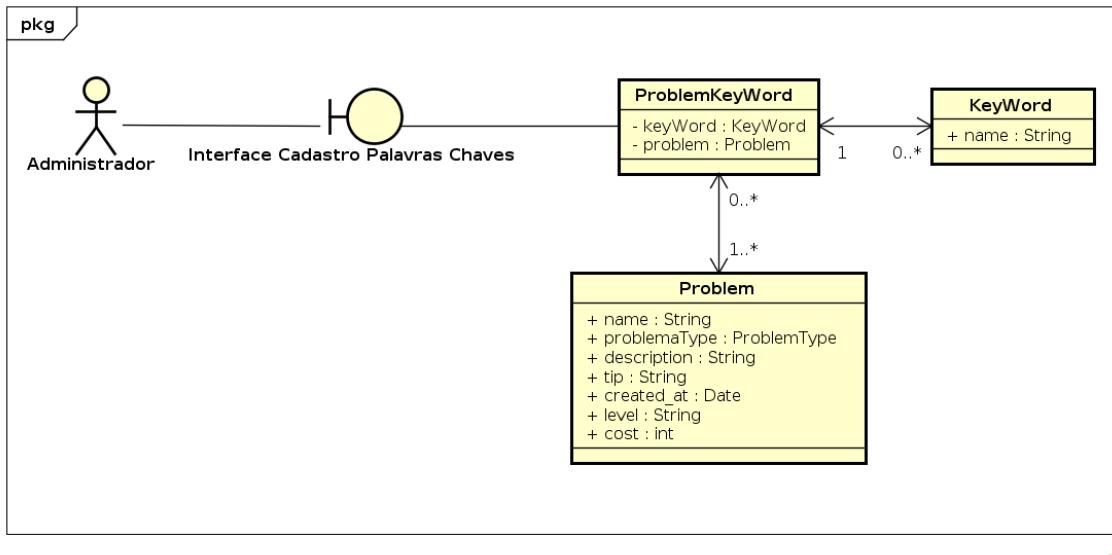
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.7 Palavras-chave

A Figura 4.42 demonstra as associações das classes de domínio na interface gráfica de cadastro de palavras chaves.

Figura 4.42: Diagrama de Robustez de Palavras-chave



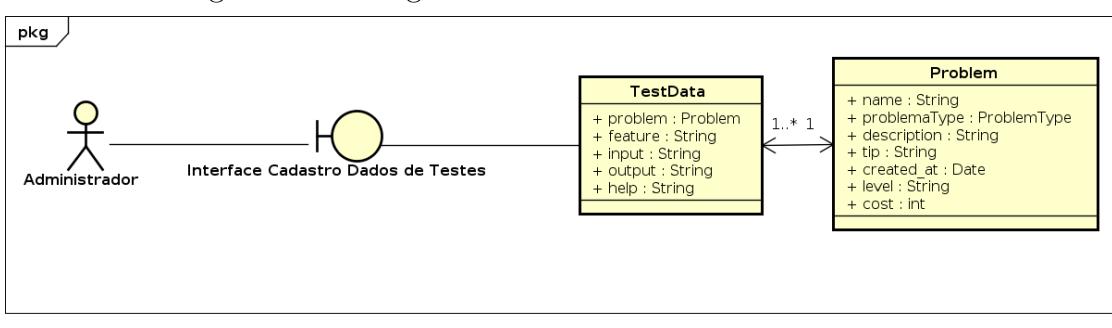
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.8 Dados de Testes

A Figura 4.43 demonstra as associações das classes de domínio na interface gráfica de cadastro de dados de testes.

Figura 4.43: Diagrama de Robustez de Dados de Testes



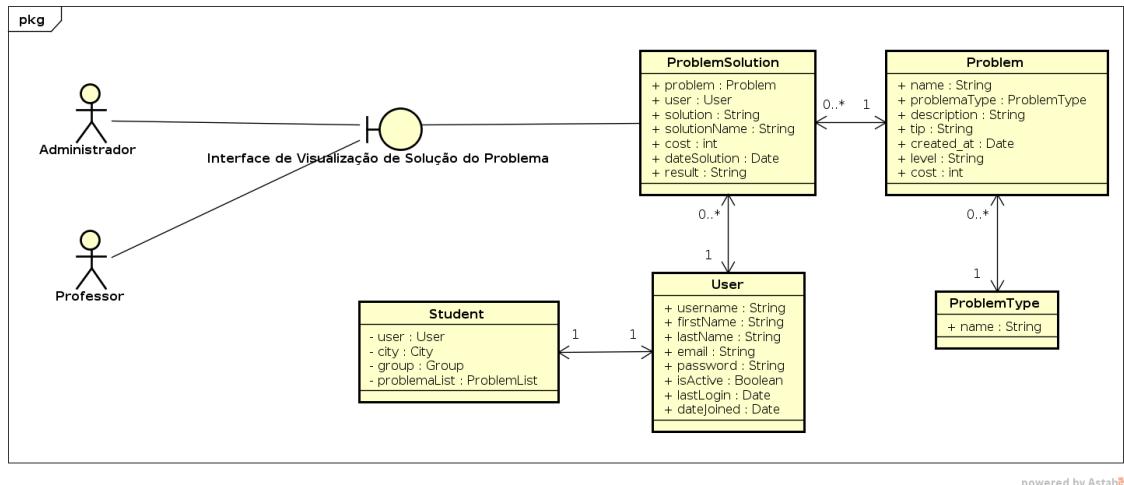
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.9 Visualização de Solução partindo da listagem de Problemas

A Figura 4.44 demonstra as associações das classes de domínio na interface gráfica de visualização da solução de problemas.

Figura 4.44: Diagrama de Robustez de Visualização de Solução partindo da listagem de Problemas



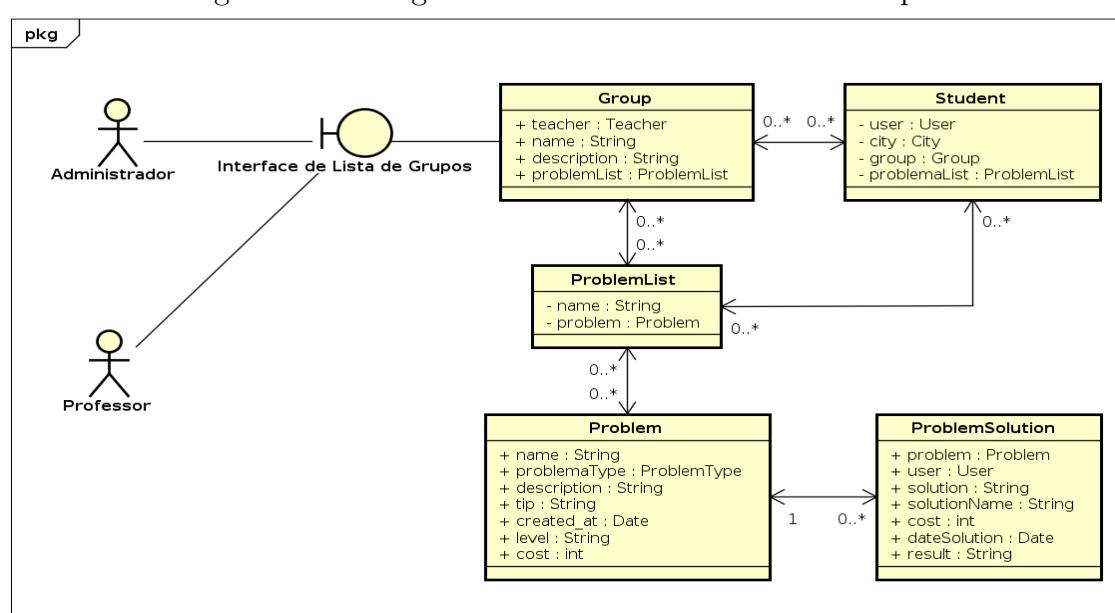
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.10 Lista de Grupos

A Figura 4.45 demonstra as associações das classes de domínio na interface gráfica de listagem de grupos.

Figura 4.45: Diagrama de Robustez de Lista de Grupos



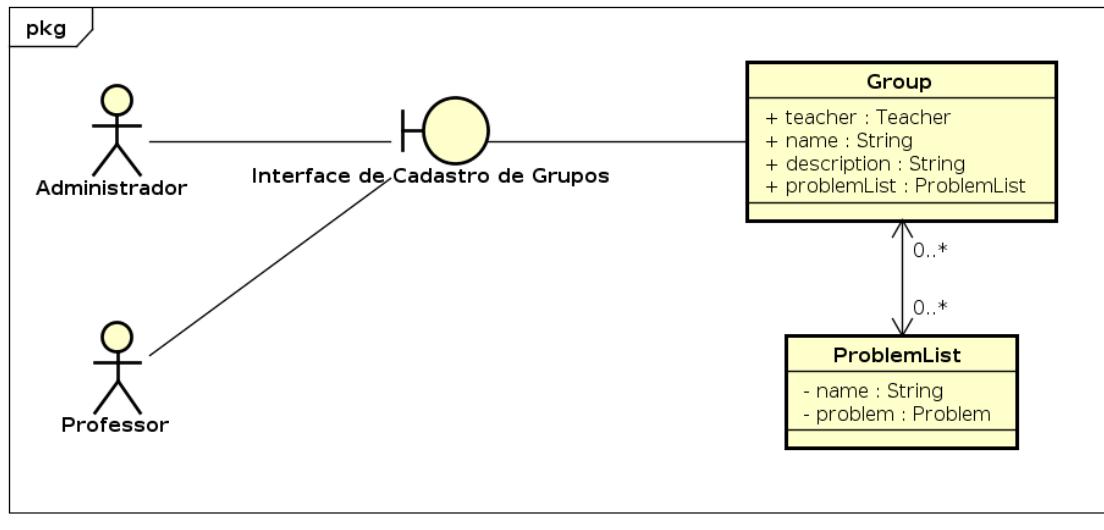
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.11 Cadastro e Edição de Grupo

A Figura 4.46 demonstra as associações das classes de domínio na interface gráfica de cadastro e edição de grupos.

Figura 4.46: Diagrama de Robustez de Cadastro e Edição de Grupo



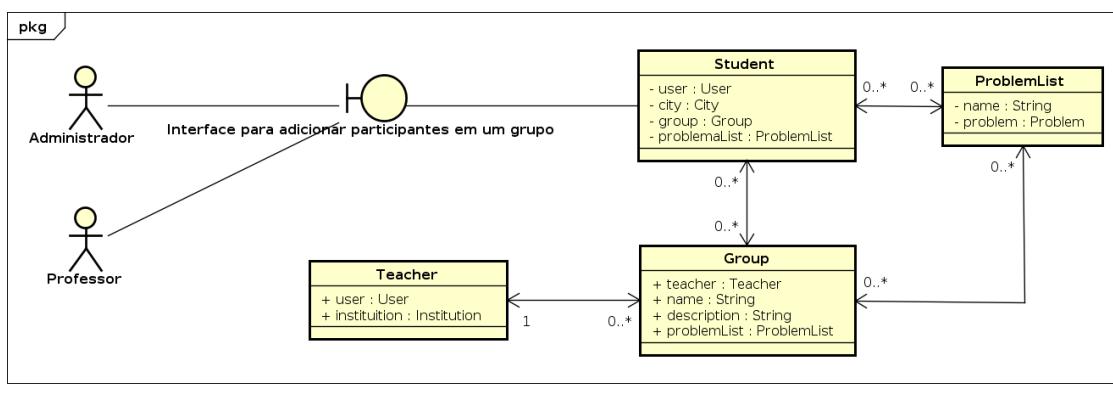
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.12 Adicionar Participantes

A Figura 4.47 demonstra as associações das classes de domínio na interface gráfica de adicionar participantes a um grupo.

Figura 4.47: Diagrama de Robustez de Adicionar Participantes



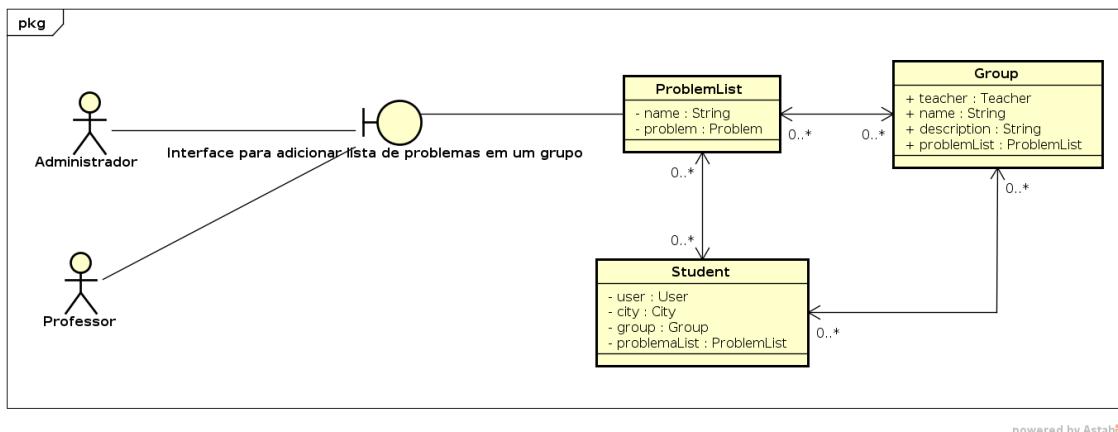
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.13 Adicionar Lista de Problemas

A Figura 4.48 demonstra as associações das classes de domínio na interface gráfica de adicionar listas de problemas a um grupo.

Figura 4.48: Diagrama de Robustez de Adicionar Lista de Problemas



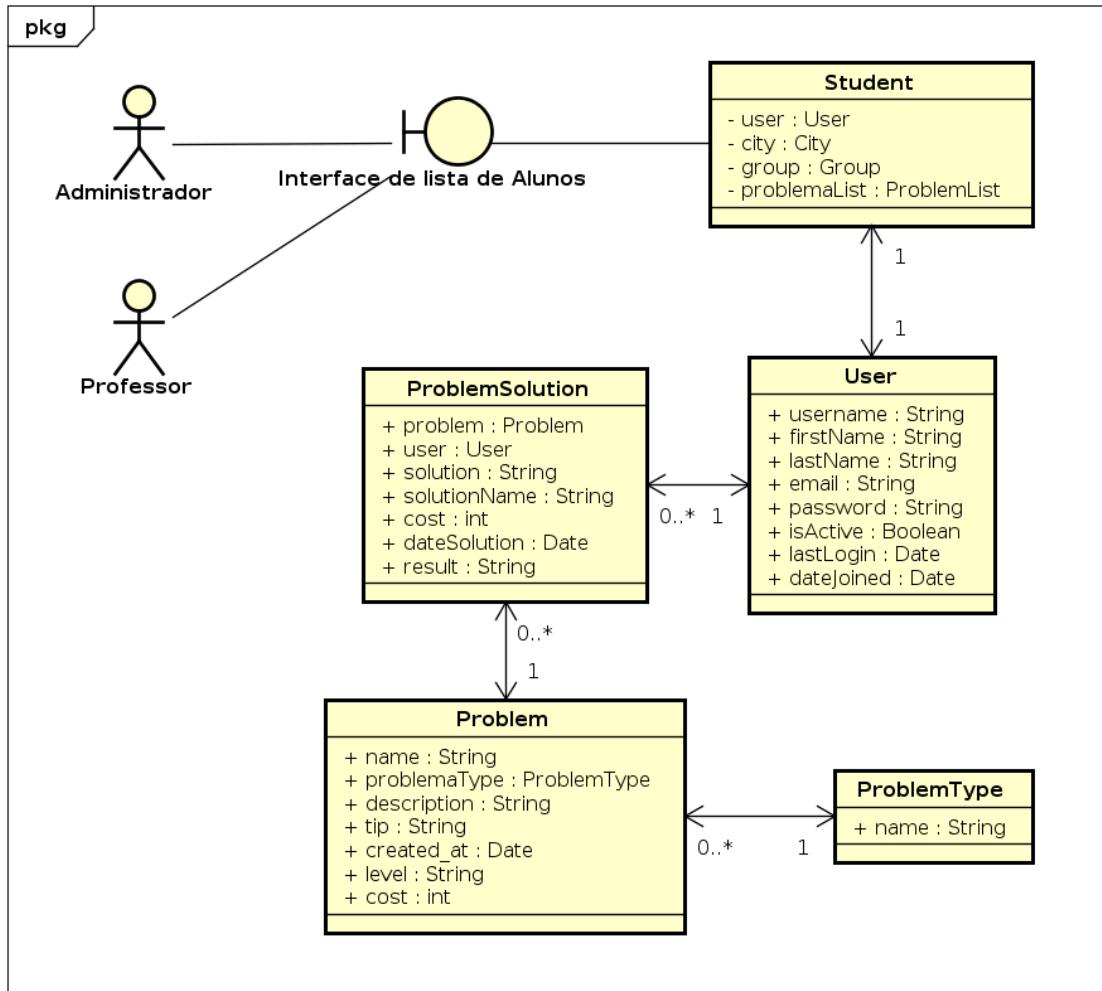
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.14 Lista de Alunos

A Figura 4.49 demonstra as associações das classes de domínio na interface gráfica e listagem de alunos.

Figura 4.49: Diagrama de Robustez de Lista de Alunos



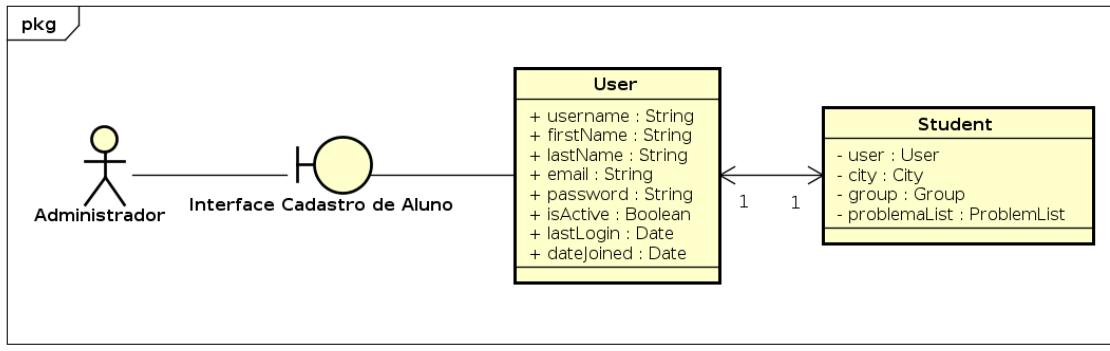
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.15 Cadastro de Alunos pelo Administrador

A Figura 4.50 demonstra as associações das classes de domínio na interface gráfica de cadastro de aluno.

Figura 4.50: Diagrama de Robustez de Cadastro de Alunos pelo Administrador



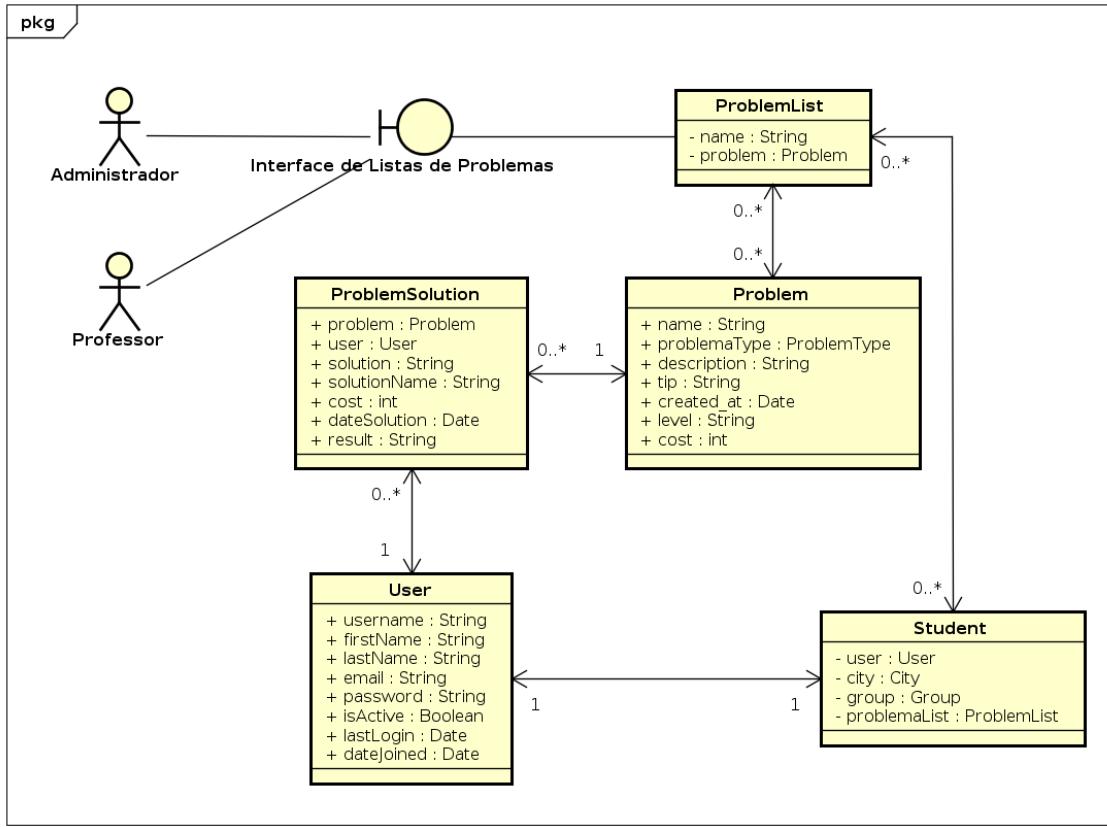
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.16 Lista de Problemas

A Figura 4.51 demonstra as associações das classes de domínio na interface gráfica de listagem de problemas.

Figura 4.51: Diagrama de Robustez de Lista de Problemas



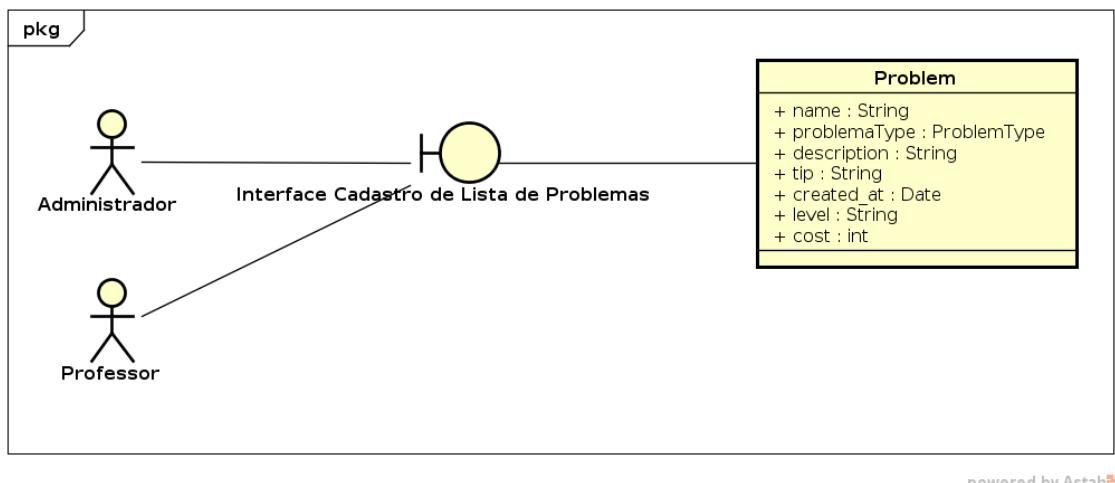
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.17 Cadastro e Edição de Lista de Problemas

A Figura 4.52 demonstra as associações das classes de domínio na interface gráfica de cadastro e edição de listas de problemas.

Figura 4.52: Diagrama de Robustez de Cadastro e Edição de Lista de Problemas



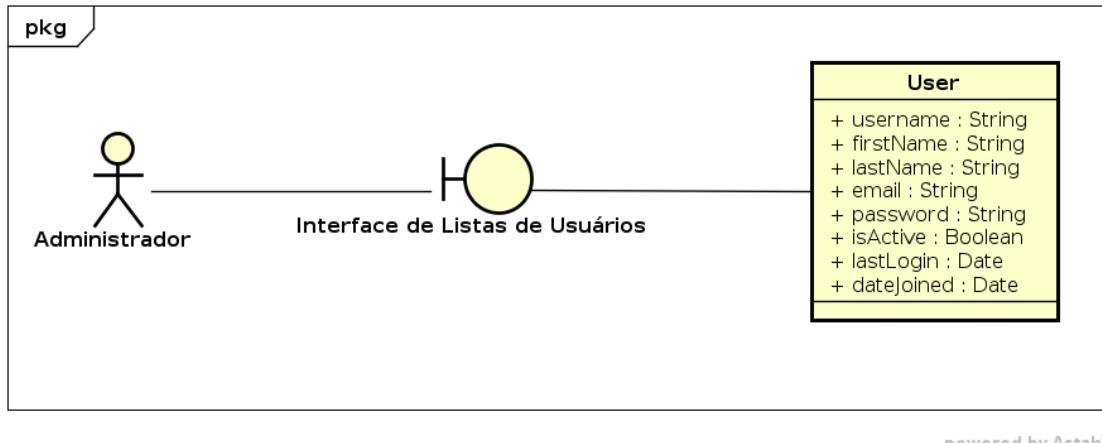
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.18 Lista de Usuários

A Figura 4.53 demonstra as associações das classes de domínio na interface gráfica de listagem de usuários.

Figura 4.53: Diagrama de Robustez de Lista de Usuários



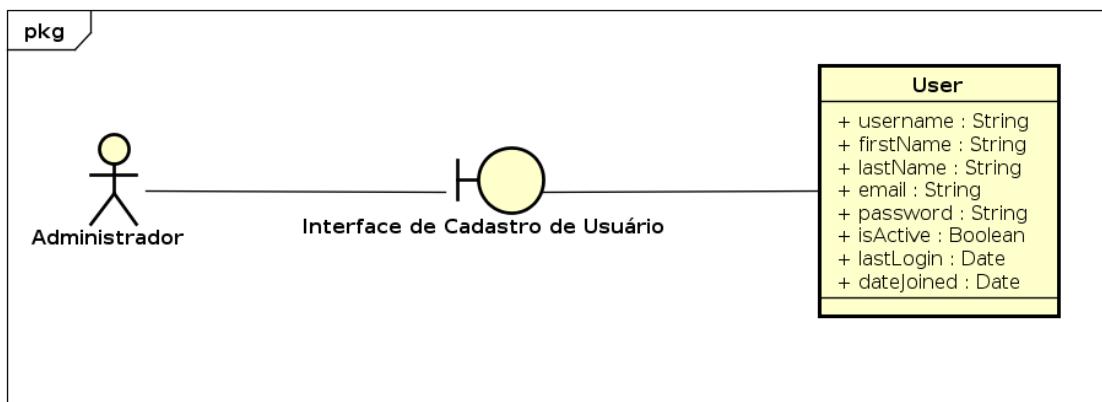
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.19 Cadastro e Edição de Usuários

A Figura 4.54 demonstra as associações das classes de domínio na interface gráfica de cadastro e edição de usuários.

Figura 4.54: Diagrama de Robustez de Cadastro e Edição de Usuários



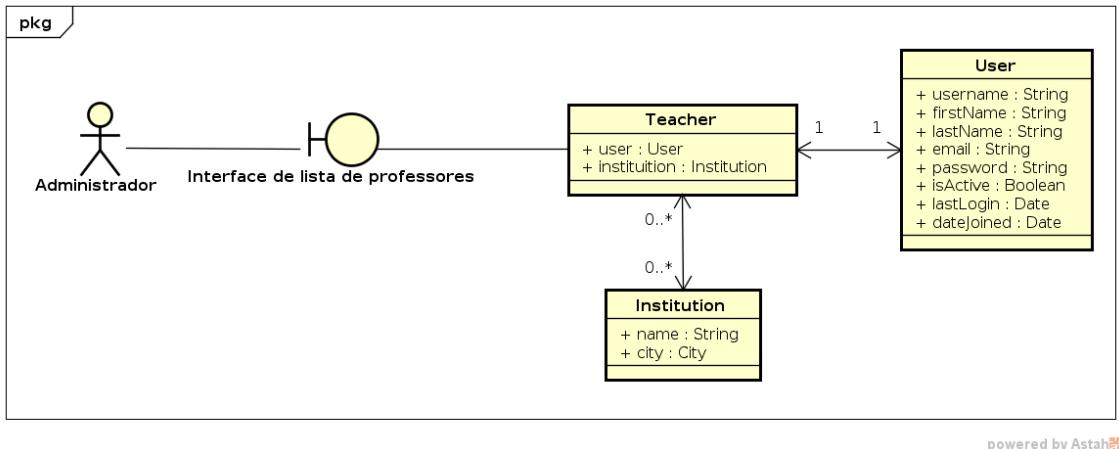
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.20 Lista de Professores

A Figura 4.55 demonstra as associações das classes de domínio na interface gráfica de listagem de professores.

Figura 4.55: Diagrama de Robustez de Lista de Professores



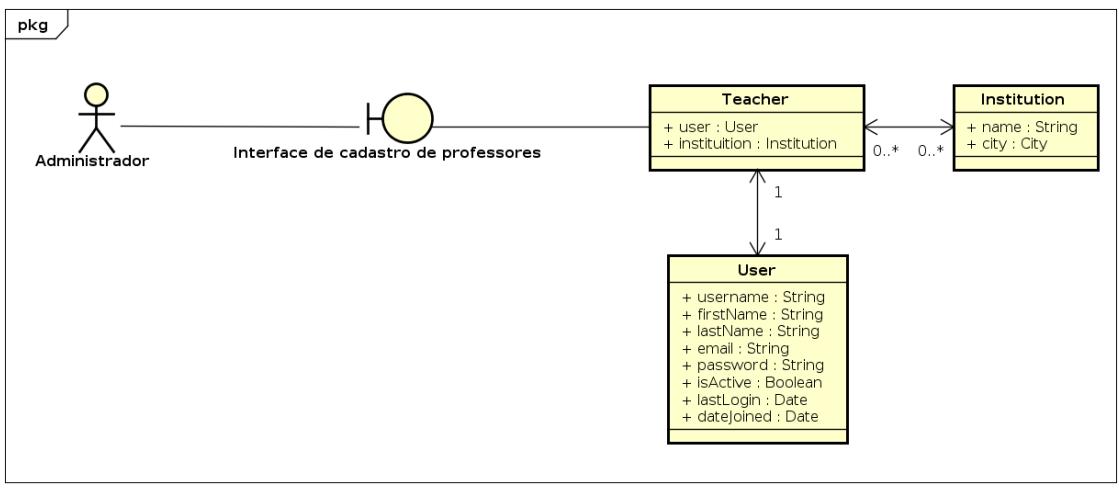
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.21 Cadastro e Edição de Professores

A Figura 4.56 demonstra as associações das classes de domínio na interface gráfica de cadastro e edição de professores.

Figura 4.56: Diagrama de Robustez de Cadastro e Edição de Professores



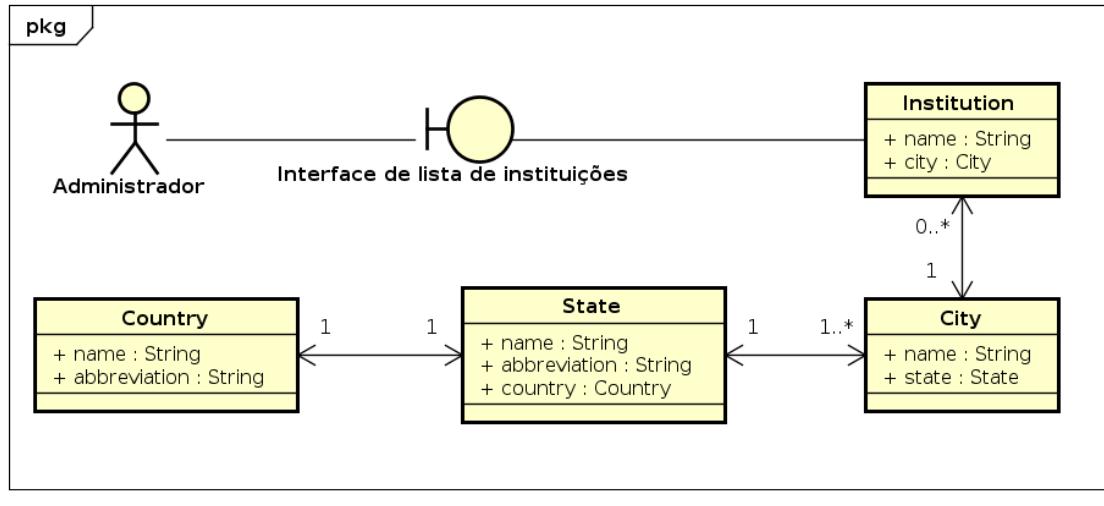
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.22 Lista de Instituições

A Figura 4.57 demonstra as associações das classes de domínio na interface gráfica de listagem de instituições.

Figura 4.57: Diagrama de Robustez de Lista de Instituições



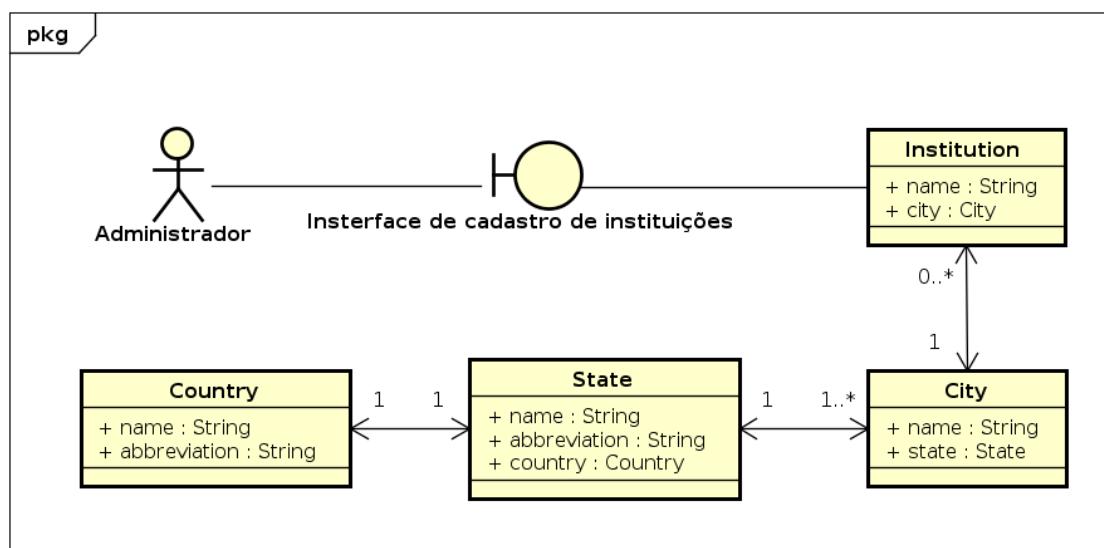
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.23 Cadastro e Edição de Instituições

A Figura 4.58 demonstra as associações das classes de domínio na interface gráfica de cadastro e edição de instituições.

Figura 4.58: Diagrama de Robustez de Cadastro e Edição de Instituições



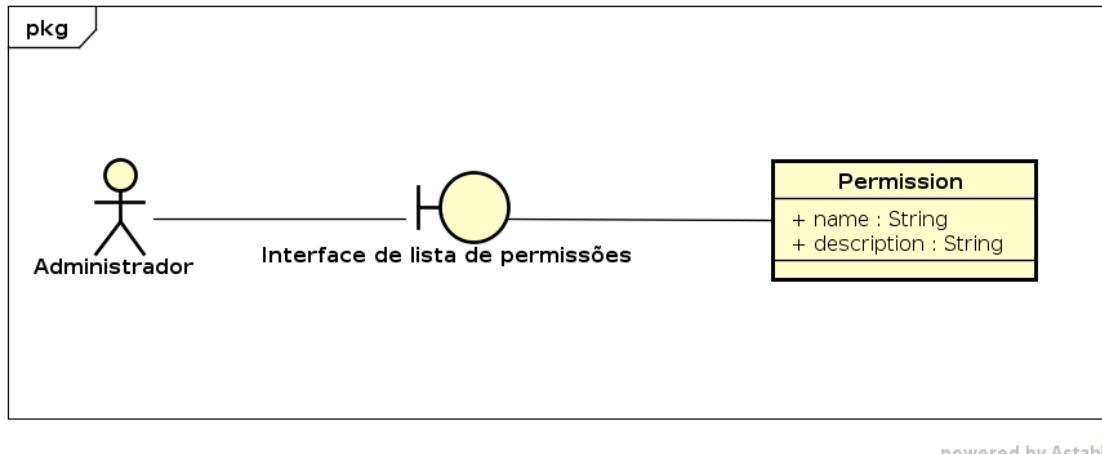
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.24 Lista de Permissões

A Figura 4.59 demonstra as associações das classes de domínio na interface gráfica de listagem de permissões.

Figura 4.59: Diagrama de Robustez de Lista de Permissões



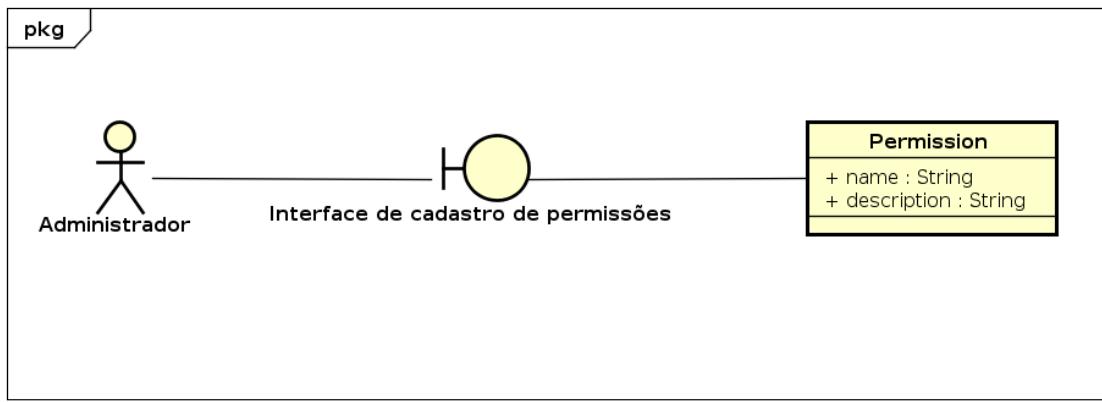
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.25 Cadastro e Edição de Permissões

A Figura 4.60 demonstra as associações das classes de domínio na interface gráfica de cadastro e edição de permissões.

Figura 4.60: Diagrama de Robustez de Cadastro e Edição de Permissões



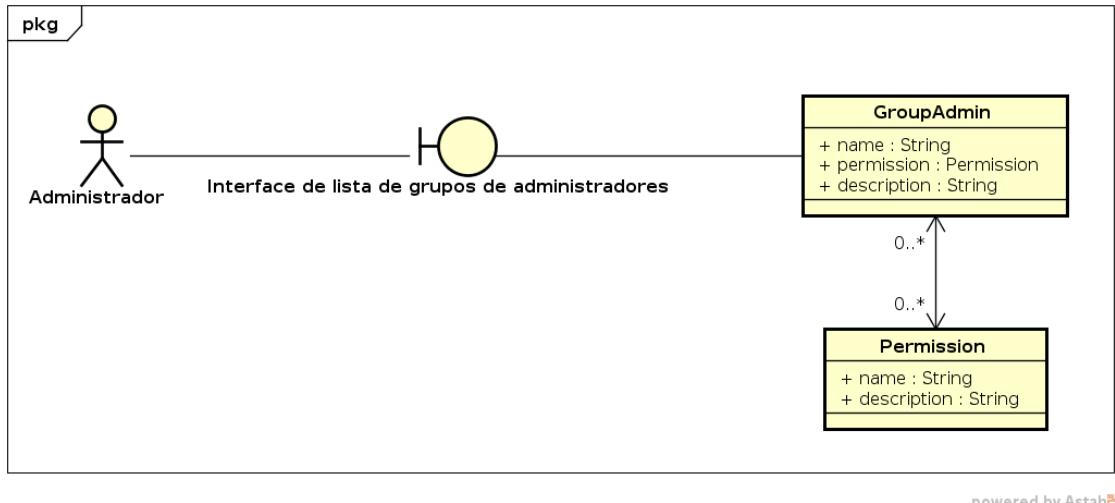
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.26 Lista de Grupos de Administradores

A Figura 4.61 demonstra as associações das classes de domínio na interface gráfica de listagem de grupos de administradores.

Figura 4.61: Diagrama de Robustez de Lista de Grupos de Administradores



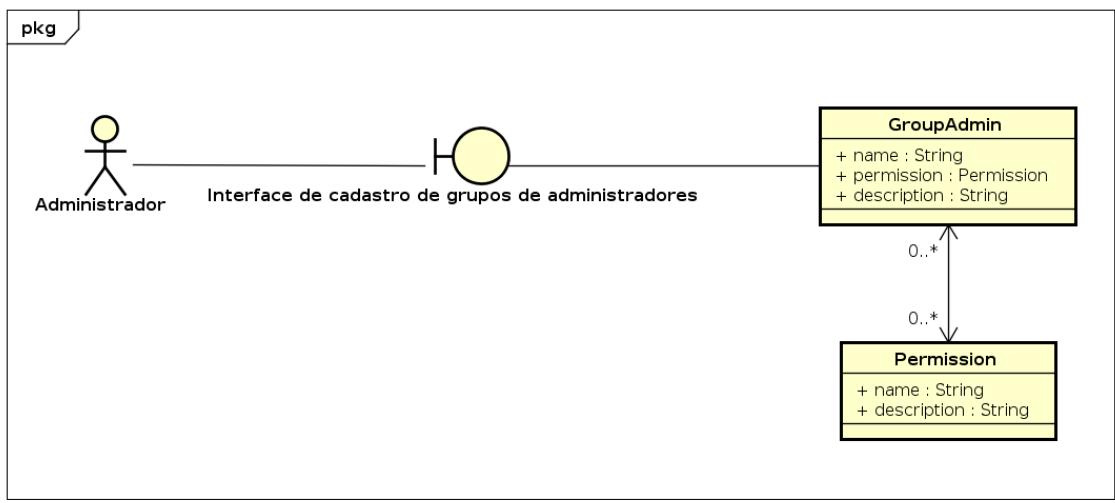
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.27 Cadastro e Edição de Grupos de Administradores

A Figura 4.62 demonstra as associações das classes de domínio na interface gráfica de cadastro e edição de grupos de administradores.

Figura 4.62: Diagrama de Robustez de Cadastro e Edição de Grupos de Administradores



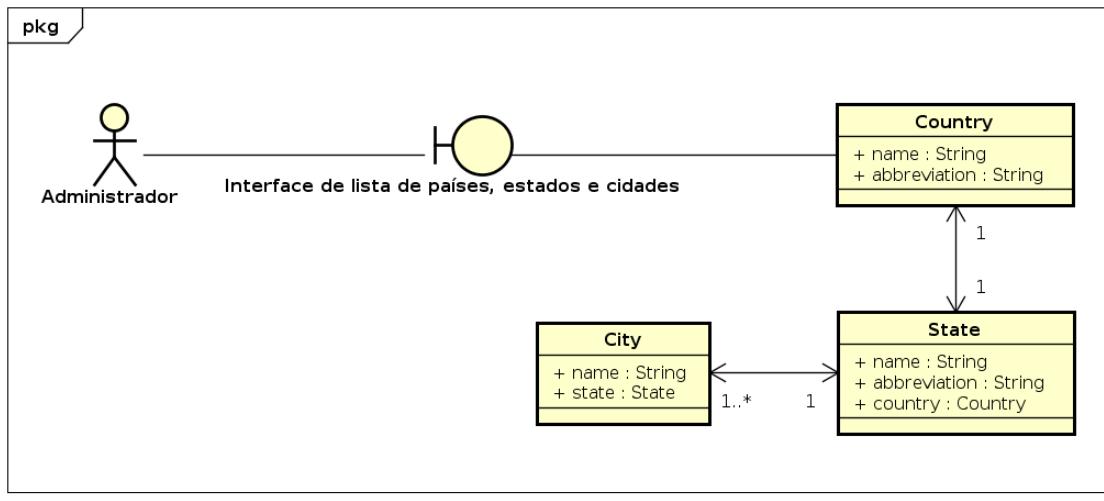
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.28 Lista de Países, Estados e Cidades

A Figura 4.63 demonstra as associações das classes de domínio na interface gráfica de listagem de países, estados e cidades.

Figura 4.63: Diagrama de Robustez de Lista de Países, Estados e Cidades



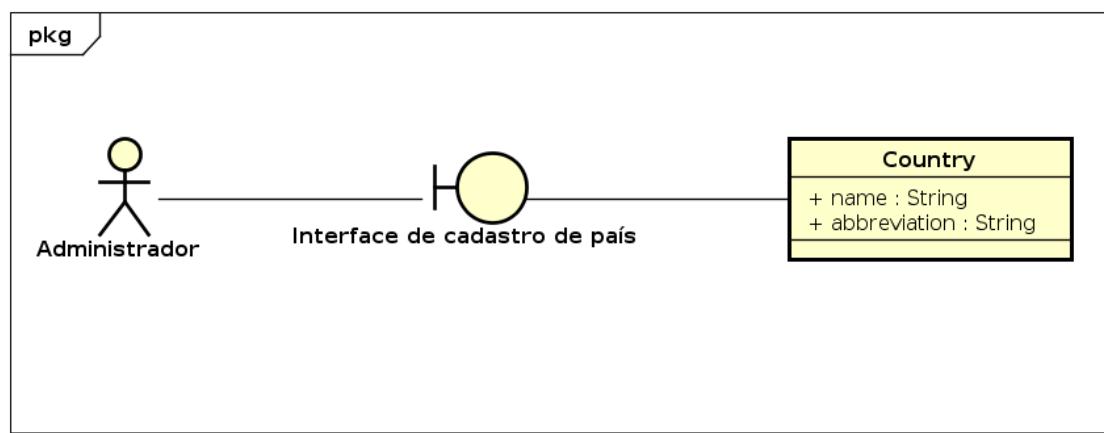
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.29 Cadastro e Edição de País

A Figura 4.64 demonstra as associações das classes de domínio na interface gráfica de cadastro e edição de países.

Figura 4.64: Diagrama de Robustez de Cadastro e Edição de País



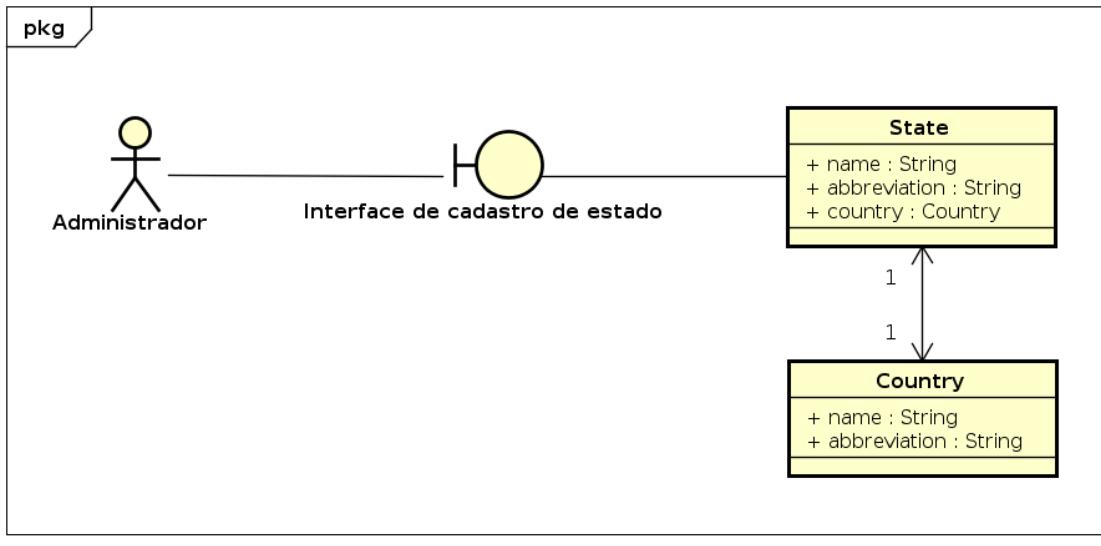
powered by Astah

Fonte: (AUTOR, 2018)

#### 4.5.30 Cadastro e Edição de Estado

A Figura 4.65 demonstra as associações das classes de domínio na interface gráfica de cadastro e edição de estados.

Figura 4.65: Diagrama de Robustez de Cadastro e Edição de Estado

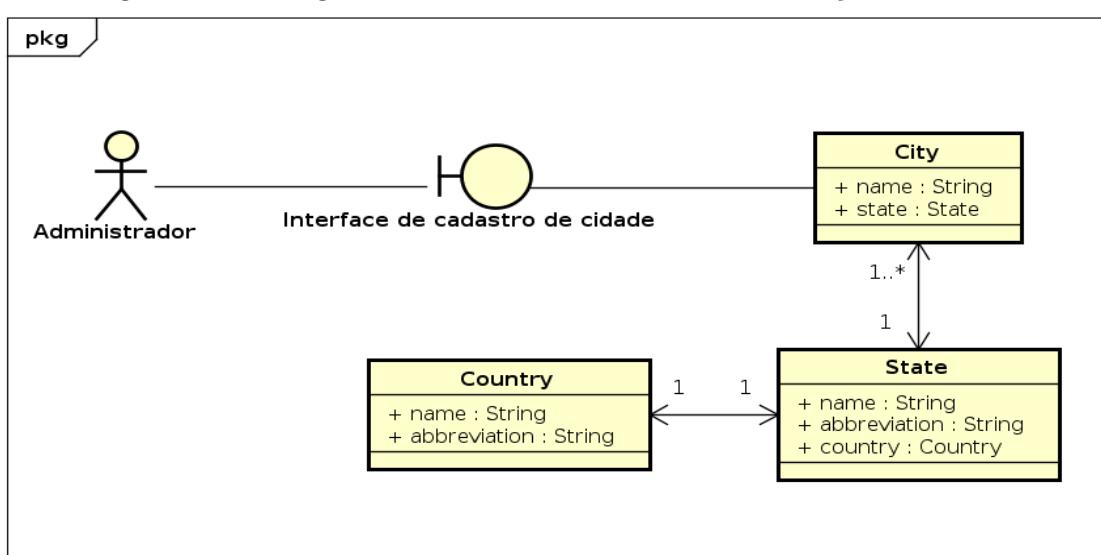


Fonte: (AUTOR, 2018)

#### 4.5.31 Cadastro e Edição de Cidade

A Figura 4.66 demonstra as associações das classes de domínio na interface gráfica de cadastro e edição de cidades.

Figura 4.66: Diagrama de Robustez de Cadastro e Edição de Cidade

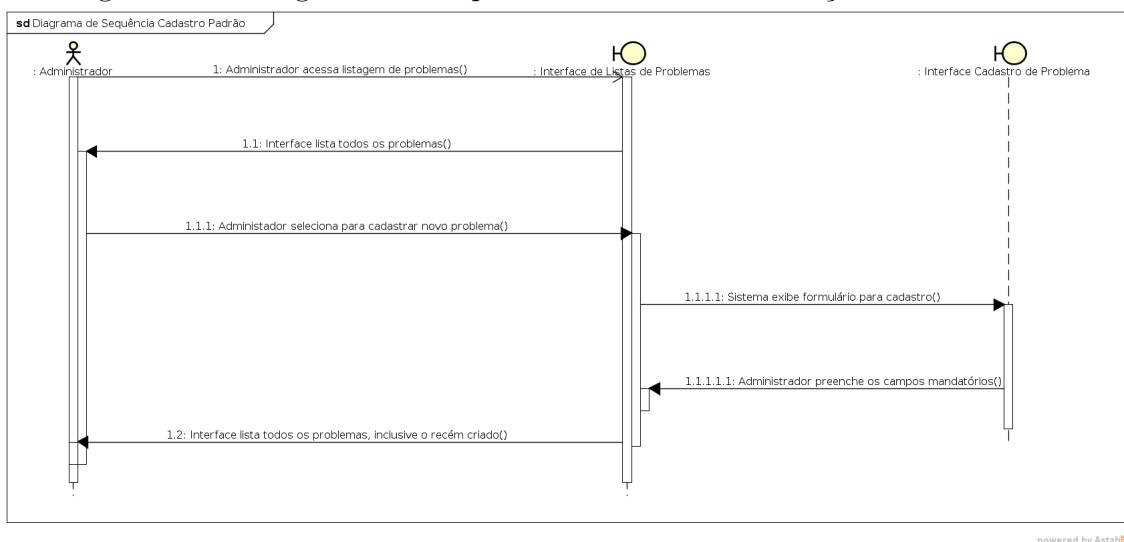


Fonte: (AUTOR, 2018)

## 4.6 Diagramas de Sequência

A Figura 4.67 demonstra o fluxo de cadastro de um novo problema. O administrador acessa a listagem de problemas, selecionando na interface para o cadastro de um novo problema, o sistema exibe o formulário de cadastro e o administrador preenche todos os campos mandatórios e clica em “salvar” fazendo com que o sistema retorne para a listagem de problemas.

Figura 4.67: Diagrama de Sequência de Cadastro e Edição de Problemas



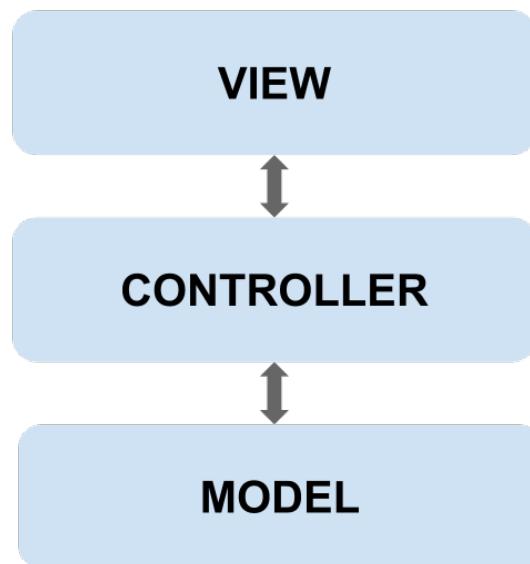
Fonte: (AUTOR, 2018)

Não foram construídos outros diagramas de sequência, porque todos seriam muito semelhantes, trocando-se somente as classes envolvidas.

## 4.7 Arquitetura do Software

Para o desenvolvimento da evolução do portal de algoritmos foi definido utilizar a arquitetura de *software MVC*, estas representadas na Figura 4.68. Onde na camada de *Model* é feita a manipulação de dados, leitura, escrita de dados. Já camada de *Controller* é responsável por receber todas as requisições do usuário, nessa camada possui métodos de ações. Por fim na camada de *View* temos a interação com o usuário, essa camada faz apenas a exibição dos dados.

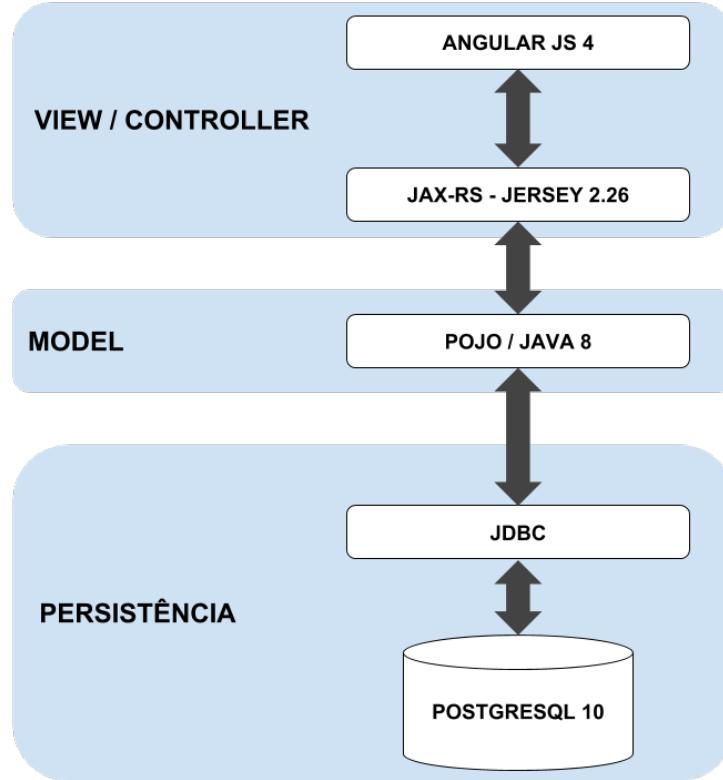
Figura 4.68: Arquitetura Lógica do Portal de Algoritmos



Fonte: (AUTOR, 2018)

Além da arquitetura lógica, temos a arquitetura de sistema, que é apresentado na Figura 4.69.

Figura 4.69: Arquitetura do Software do Portal de Algoritmos



Fonte: (AUTOR, 2018)

Na camada de *View e Controller* utilizamos duas tecnologias, *AngularJS 4* e *JAX-RS* com a implementação *Jersey 2.26*, essas tecnologias são responsáveis pela exibição do dados e comunicação com a camada de modelos.

A camada de *POJO e JAVA 8* é responsável pela manipulação dos dados transformando-os em objetos *Java*.

Na camada de persistência temos o *JDBC* (*Java Database Connectivity*) que é responsável pela comunicação com o banco de dados *Postgresql 10*.

Até o presente momento foi apresentado a modelagem e estrutura do novo *software*, a seguir é sugerido a configuração de segurança para o servidor de aplicação.

## 5 CONSIDERAÇÕES FINAIS

O objetivo do trabalho era realizar a implementação de um *software* novo visando a evolução do antigo. Esse objetivo foi entregue em uma forma mais organizada em questão de arquitetura e codificação. Durante o trabalho foram encontrados diversos problemas no *software* antigo, esses problemas foram sendo resolvidos de diversas formas, como as citadas a seguir.

O portal antigo utilizava tecnologias desatualizadas, como *Java Applet*, *HTML* sem estruturação e sem validações. Esse problema foi resolvido utilizando *AngularJS 4*, um framework javascript para implementar o consumo de *API (Application Programming Interface) REST*. Além do consumo de dados é preciso exibir esses dados, então foi utilizado um *framework* de *CSS (Cascading Style Sheets)*, este conhecido como *Bootstrap*. Com ele foi possível criar interfaces responsivas e intuitivas para o uso do *software*.

O novo *software* foi desenvolvido utilizando tecnologias de ponta, como *Java* na última versão estável, *AngularJS* na versão mais recente estável e um *framework* de *CSS* fácil de utilizar.

Com a nova arquitetura do *software* ficou mais fácil e rápido de implementar novas funcionalidades conforme as necessidades forem surgindo. A arquitetura engloba uma *API REST* fácil de integrar e configurar.

O sistema existente antes da elaboração deste trabalho foi analisado a fim de identificar as possíveis possibilidades de melhoria. Como visto no Capítulo 3, foi apresentada a estrutura de banco de dados e as telas que compõe a interface deste sistema. Em algumas destas telas verificou-se pouca usabilidade conforme visto nas Figuras 3.4, 3.5, 3.6 e 3.7.

Na proposta de melhorias do software apresentada no Capítulo 4, observou-se a elaboração de tabelas com mais elementos e que contemplaram as necessidades do sistema. Os diagramas de requisitos possibilitaram um melhor entendimento do problema para a implementação. Além disso foram apresentados os casos de uso para descrever estes requisitos. Com a proposta de interface gráfica e o diagrama de robustez foi possível identificar as interações do usuário com o sistema contribuindo

também na implementação final e na interação com os domínios. Os diagramas de fluxo permitiram uma visão geral do software possibilitando a verificação e a validação da solução proposta.

## 5.1 Trabalhos Futuros

Com esse novo *software* implementado é possível planejar o desenvolvimento de:

- Aplicativo móvel
- Novas funcionalidades necessárias
- Integração com analisador algorítmico
- Testes unitários
- Testes de integração
- Testes de interface

## REFERÊNCIAS

ALUR, D. **Core J2EE Patterns** : as melhores práticas e estratégias de design. 2.ed. Rio de Janeiro: Elsevier, 2004. 587p. ISBN 8535212728.

BENYON, D. **Interação Humano-Computador**. São Paulo - SP: Person, 2011. 442p. ISBN 978-85-7936-109-8.

BRANAS, R. **AngularJS Essentials. Design and construct reusable, maintainable, and modular web applications with AngularJs**. Birmingham - UK: Packt Publishing, 2014. 164p. ISBN 978-1-78398-008-6.

DJANGO. **Django**. <Disponível em: <https://www.djangoproject.com/>>. Acesso em: Junho de 2018.

DORNELES, R. V.; JUNIOR, D. P.; ADAMI, A. G. AlgoWeb: a web-based environment for learning introductory programming. **ADVANCED LEARNING TECHNOLOGIES (ICALT)**, Sousse, v.10, p.83–85, 2010.

ENCRYPTION, L. **Let's Encrypt**. <Disponível em: <https://letsencrypt.org/about/>>. Acesso em: 3 de Junho de 2018.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. <Disponível em: [https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding\\_dissertation.pdf](https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf)>. Acesso em: Junho de 2018.

GALEOTE, S. **Do requisito ao diagrama de classe de projeto: ágil e descomplicado**. <Disponível em: <http://www.galeote.com.br/blog/2013/05/do-requisito-ao-diagrama-de-classe-de-projeto-agil-e-descomplicado/>>. Acesso em: 2 de Dezembro de 2015.

GOOGLE. **Conteúdo baseado em plug-in não funciona no Google Chrome**. <Disponível em: <https://support.google.com/chrome/answer/6213033?hl=pt-BR>>. Acesso em: Junho de 2018.

JAVA. **Java.** <Disponível em: <https://docs.oracle.com/javaee/7/tutorial/>>. Acesso em: Junho de 2018.

LARMAN, C. **Utilizando UML e Padrões. Uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo.** 3.ed. Porto Alegre - RS: Bookman, 2007. 696p. ISBN 978-85-60031-52-8.

OLIVEIRA, C. C. de; COSTA, J. W.; MOREIRA, M. **Ambientes informatizados de aprendizagem:** produção e avaliação de software educativo. São Paulo - SP: [s.n.], 2004. 144p. ISBN 85-3080-634-4.

OWASP. **Man in the Middle.** <Disponível em: [https://www.owasp.org/index.php/Man-in-the-middle\\_attack](https://www.owasp.org/index.php/Man-in-the-middle_attack)>. Acesso em: 3 de Junho de 2018.

OWASP. **SQL Injection.** <Disponível em: [https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection)>. Acesso em: 3 de Junho de 2018.

PRESSMAN, R. S. **Engenharia de Software, Uma Abordagem Profissional.** 7.ed. São Paulo - SP: AMGH Editora Ltda, 2011. 780p. ISBN 978-85-63308-7.

PYTHON. **Python.** <Disponível em: <https://www.python.org/>>. Acesso em: Junho de 2018.

REZENDE, D. A. **Engenharia de Software e Sistemas de Informação.** Rio de Janeiro - RJ: BRASPORT, 2005. 316p. ISBN 85-7452-215-5.

RICHARDSON, L.; AMUNDSEN, M. **RESTful Web APIs.** USA: O'Reilly, 2013. 372p.

ROSENBERG, D. et al. . New York: Apress, 2005. 261p. ISBN 1-59059-464-9.

SANTOS WEBER, G. dos. **Desenvolvimento de uma Representação Intermediária para o Portal de Algoritmos da UCS.** Caxias do Sul, RS, 2015. 76p.

SOMMERVILLE, I. **Engenharia de Software.** São Paulo - SP: PERSON, 2011. 529p. ISBN 978-85-7936-108-1.

VALENTINI, C. B.; SOARES, E. M. S. **Aprendizagem em Ambientes Virtuais:** compartilhando idéias e construindo cenários. Caxias do Sul - RS: EDUCS, 2005. 209p. ISBN 978-85-7061-600-5.

WILDFLY. **WildFly.** <Disponível em: <http://wildfly.org/>>. Acesso em: Junho de 2018.

## APÊNDICE A - SEGURANÇA

O Portal de Algoritmos antigo estava suscetível a algumas falhas de segurança. Embora não se tenha notícia delas terem sido exploradas, foi preocupação desse trabalho sanar tais possibilidades. Dentre as falhas possíveis, cita-se as seções a seguir:

### Ataque Man in the Middle

O ataque *man-in-the middle* intercepta uma comunicação entre dois sistemas. Por exemplo, em uma transação *HTTP*, o destino é a conexão *TCP* (*Transmission Control Protocol*) entre cliente e servidor. Usando técnicas diferentes, o invasor divide a conexão *TCP* original em duas novas conexões, uma entre o cliente e o invasor e a outra entre o invasor e o servidor. Quando a conexão *TCP* é interceptada, o invasor age como um *proxy*, sendo capaz de ler, inserir e modificar os dados na comunicação interceptada.(OWASP, 2018a)

### Ataque SQL Injection

Um ataque de injeção SQL consiste em inserção ou ”injeção” de uma consulta SQL por meio dos dados de entrada do cliente para o aplicativo. Uma exploração de injeção SQL bem-sucedida pode ler dados confidenciais do banco de dados, modificar dados do banco de dados dentre eles, inserir, atualizar e excluir. Os ataques de injeção de SQL são um tipo de ataque de injeção , no qual os comandos SQL são injetados na entrada do plano de dados para efetuar a execução de comandos SQL predefinidos.(OWASP, 2018b)

### Uso de HTTPS sem Certificado de Segurança Válido

O *HTTPS* (*Hyper Text Transfer Protocol Secure*) tecnicamente falando é *HTTP* sobre *SSL* (*Secure Socket Layer*), ele codifica e decodifica solicitações de páginas de usuários. É importante saber que ele protege contra ataques no site.

Para adicionar essa camada de segurança foi estudado o *Let's Encrypt*, ele é uma autoridade certificadora livre, autorizada e aberta. Eles fornecem certificados válidos gratuitamente.(ENCRYPT, 2018)

## Sugestão de instalação

No mercado existem diversas autoridades certificadoras que fornecem certificados digitais para fornecer segurança e criptografia de dados nos sistemas online.

Bem como existem certificados pagos, existem também certificados gratuitos. E para esse trabalho sugerimos utilizar a *Let's Encrypt*, uma autoridade certificadora que fornece certificados gratuitos com validade de 3 meses, mas que podem ser auto-renovados diretamente no servidor da aplicação.

O ataque do tipo de *SQL (Structured Query Language) Injection* foi mitigado pela reprogramação utilizando serviços *REST* e padrão *DAO*, o que isolou o usuário final da camada de persistência.

## APÊNDICE B - DOCUMENTAÇÃO

Essa documentação tem por objetivo de exemplificar todas as chamadas da API do Portal de Algoritmos.

Código	Descrição
200	OK
201	Recurso criado
204	Sem conteúdo
401	Credenciais inválidas
403	Não permitido
404	Não encontrado
500	Erro interno do servidor

O código 200 é gerado numa requisição de sucesso. Por exemplo na listagem de algum recurso.

O código 201 é gerado na criação de um novo recurso.

O código 204 é gerado na deleção de um recurso.

O código 401 é gerado quando as credenciais são inválidas.

O código 403 é gerado quando as credenciais não tem permissão no recurso solicitado.

O código 404 é gerado quando não é encontrado determinado recurso.

O código 500 é gerado quando o servidor apresentou algum erro interno.

## API Login

Para todas as chamadas que necessitam de autenticação para seu uso é preciso realizar o login para obter o token de autenticação, esse *token* é adicionado no header da requisição.

### *Request*

```
1 POST /rest/login
```

```
1 {  
2     "username": "String",  
3     "password": "String"  
4 }
```

### *Response*

```
1 {  
2     "token": "String"  
3 }
```

## Cabeçalho

Todas as requisições que exigem autenticação será preciso enviar o token recebido no login.

```
1 "Authorization": "Bearer <token>"
```

## API de Países

API de países.

### Buscar todos os países

*Request*

```
1 GET /rest/countries
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "name": "String",
5     "abbreviation": "String"
6   }
7 ]
```

### Detalhes de um país

*Request*

```
1 GET /rest/countries/{id}
```

*Response*

```
1 {
2   "id": "Integer",
3   "name": "String",
4   "abbreviation": "String"
5 }
```

### Criar um novo país

*Request*

```
1 POST /rest/countries
```

```
1 {
2   "name": "String",
3   "abbreviation": "String"
4 }
```

*Response*

```
1 {  
2     "id": "Integer",  
3     "name": "String",  
4     "abbreviation": "String"  
5 }
```

## Atualizar um país

*Request*

```
1 PUT /rest/countries
```

```
1 {  
2     "id": "Integer",  
3     "name": "String",  
4     "abbreviation": "String"  
5 }
```

*Response*

```
1 {  
2     "id": "Integer",  
3     "name": "String",  
4     "abbreviation": "String"  
5 }
```

## Deletar um país

*Request*

```
1 DELETE /rest/countries/{id}
```

## API de Estados

API de estados.

### Buscar todos os estados

*Request*

```
1 GET /rest/states
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "name": "String",
5     "abbreviation": "String",
6     "country": {
7       "id": "Integer",
8       "name": "String",
9       "abbreviation": "String"
10      }
11    }
12 ]
```

### Detalhes de um estado

*Request*

```
1 GET /rest/states/{id}
```

*Response*

```
1 {
2   "id": "Integer",
3   "name": "String",
4   "abbreviation": "String",
5   "country": {
6     "id": "Integer",
7     "name": "String",
8     "abbreviation": "String"
9   }
10 }
```

## Criar um novo estado

*Request*

```
1 POST /rest/states
```

```
1 {
2     "name": "String",
3     "abbreviation": "String",
4     "country": {
5         "id": "Integer"
6     }
7 }
```

*Response*

```
1 {
2     "id": "Integer",
3     "name": "String",
4     "abbreviation": "String",
5     "country": {
6         "id": "Integer",
7         "name": "String",
8         "abbreviation": "String"
9     }
10 }
```

## Atualizar um estado

*Request*

```
1 PUT /rest/states
```

```
1 {
2     "id": "Integer",
3     "abbreviation": "String",
4     "country": {
5         "id": "Integer"
6     }
7 }
```

*Response*

```

1 {
2   "id": "Integer",
3   "name": "String",
4   "abbreviation": "String",
5   "country": {
6     "id": "Integer",
7     "name": "String",
8     "abbreviation": "String"
9   }
10 }
```

**Deletar um estado***Request*

```
1 DELETE /rest/states/{id}
```

**API de Cidades**

API de cidades.

**Buscar todos as cidades***Request*

```
1 GET /rest/cities
```

*Response*

```

1 [
2   {
3     "id": "Integer",
4     "name": "String",
5     "state": {
6       "id": "Integer",
7       "name": "String",
8       "abbreviation": "String",
9       "country": {
10         "id": "Integer",
11         "name": "String",
12         "abbreviation": "String"
13       }
14     }
15   }
16 ]
```

## Detalhes de uma cidade

*Request*

```
1 GET /rest/cities/{id}
```

*Response*

```
1 {
2     "id": "Integer",
3     "name": "String",
4     "state": {
5         "id": "Integer",
6         "name": "String",
7         "abbreviation": "String",
8         "country": {
9             "id": "Integer",
10            "name": "String",
11            "abbreviation": "String"
12        }
13    }
14 }
```

## Criar uma nova cidade

*Request*

```
1 POST /rest/cities
```

```
1 {
2     "name": "String",
3     "state": {
4         "id": "Integer"
5     }
6 }
```

*Response*

```
1 {
2     "id": "Integer",
3     "name": "String",
4     "state": {
5         "id": "Integer",
6         "name": "String",
7         "abbreviation": "String"
8         "country": {
9             "id": "Integer",
10            "name": "String",
11            "abbreviation": "String"
12        }
13    }
14 }
```

## Atualizar uma cidade

*Request*

```
1 PUT /rest/cities  
  
1 {  
2     "id": "Integer",  
3     "name": "String",  
4     "state": {  
5         "id": "Integer",  
6     }  
7 }
```

*Response*

```
1 {  
2     "id": "Integer",  
3     "name": "String",  
4     "state": {  
5         "id": "Integer",  
6         "name": "String",  
7         "abbreviation": "String"  
8         "country": {  
9             "id": "Integer",  
10            "name": "String",  
11            "abbreviation": "String"  
12        }  
13    }  
14 }
```

## Deletar uma cidade

*Request*

```
1 DELETE /rest/cities/{id}
```

## API de Instituições

API de instituições.

### Buscar todas as instituições

*Request*

```
1 GET /rest/institutions
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "name": "String",
5     "initial": "String",
6     "city": {
7       "id": "Integer",
8       "name": "String",
9       "state": {
10         "id": "Integer",
11         "name": "String",
12         "abbreviation": "String",
13         "country": {
14           "id": "Integer",
15           "name": "String",
16           "abbreviation": "String"
17         }
18       }
19     }
20   }
21 ]
```

### Detalhes de uma instituição

*Request*

```
1 GET /rest/institutions/{id}
```

*Response*

```

1 {
2     "id": "Integer",
3     "name": "String",
4     "initial": "String",
5     "city": {
6         "id": "Integer",
7         "name": "String",
8         "state": {
9             "id": "Integer",
10            "name": "String",
11            "abbreviation": "String",
12            "country": {
13                "id": "Integer",
14                "name": "String",
15                "abbreviation": "String"
16            }
17        }
18    }
19 }
```

## Criar uma nova instituição

*Request*

```
1 POST /rest/institutions
```

```

1 {
2     "name": "String",
3     "initial": "String",
4     "city": {
5         "id": "Integer"
6     }
7 }
```

*Response*

```

1 {
2     "id": "Integer",
3     "name": "String",
4     "initial": "String",
5     "city": {
6         "id": "Integer",
7         "name": "String",
8         "state": {
9             "id": "Integer",
10            "name": "String",
11            "abbreviation": "String",
12            "country": {
13                "id": "Integer",
14                "name": "String",
15                "abbreviation": "String"
16            }
17        }
18    }
19 }
```

## Atualizar uma instituição

*Request*

```
1 PUT /rest/institutions
```

```

1 {
2     "id": "Integer",
3     "name": "String",
4     "initial": "String",
5     "city": {
6         "id": "Integer"
7     }
8 }
```

*Response*

```

1 {
2   "id": "Integer",
3   "name": "String",
4   "initial": "String",
5   "city": {
6     "id": "Integer",
7     "name": "String",
8     "state": {
9       "id": "Integer",
10      "name": "String",
11      "abbreviation": "String",
12      "country": {
13        "id": "Integer",
14        "name": "String",
15        "abbreviation": "String"
16      }
17    }
18  }
19 }
```

**Deletar uma instituição***Request*

```
1 DELETE /rest/institutions/{id}
```

**API de Permissões**

API de permissões.

**Buscar todas as permissões***Request*

```
1 GET /rest/permissions
```

*Response*

```

1 [
2   {
3     "id": "Integer",
4     "codename": "String",
5     "description": "String"
6   }
7 ]
```

## Detalhes de uma permissão

*Request*

```
1 GET /rest/permissions/{id}
```

*Response*

```
1 {
2     "id": "Integer",
3     "codename": "String",
4     "description": "String"
5 }
```

## Criar uma nova permissão

*Request*

```
1 POST /rest/permissions
```

```
1 {
2     "codename": "String",
3     "description": "String"
4 }
```

*Response*

```
1 {
2     "id": "Integer",
3     "codename": "String",
4     "description": "String"
5 }
```

## Atualizar uma permissão

*Request*

```
1 PUT /rest/permissions
```

```
1 {
2     "id": "Integer",
3     "codename": "String",
4     "description": "String"
5 }
```

*Response*

```

1 {
2   "id": "Integer",
3   "codename": "String",
4   "description": "String"
5 }
```

## Deletar uma permissão

*Request*

```
1 DELETE /rest/permissions/{id}
```

# API de Usuários

API de usuários.

## Buscar todos os usuários

*Request*

```
1 GET /rest/users
```

*Response*

```

1 [
2   {
3     "id": "Integer",
4     "username": "String",
5     "firstname": "String",
6     "lastname": "String",
7     "email": "String",
8     "password": "*****",
9   }
10 ]
```

## Detalhes de um usuário

*Request*

```
1 GET /rest/users/{id}
```

*Response*

```
1 {  
2   "id": "Integer",  
3   "username": "String",  
4   "firstname": "String",  
5   "lastname": "String",  
6   "email": "String",  
7   "password": "*****",  
8 }
```

## Criar um novo usuário

*Request*

```
1 POST /rest/users
```

```
1 {  
2   "username": "String",  
3   "firstname": "String",  
4   "lastname": "String",  
5   "email": "String",  
6   "password": "*****",  
7 }
```

*Response*

```
1 {  
2   "id": "Integer",  
3   "username": "String",  
4   "firstname": "String",  
5   "lastname": "String",  
6   "email": "String",  
7   "password": "*****",  
8 }
```

## Atualizar um usuário

*Request*

```
1 PUT /rest/users

1 {
2   "id": "Integer",
3   "username": "String",
4   "firstname": "String",
5   "lastname": "String",
6   "email": "String",
7   "password": "*****",
8 }
```

*Response*

```
1 {
2   "id": "Integer",
3   "username": "String",
4   "firstname": "String",
5   "lastname": "String",
6   "email": "String",
7   "password": "*****",
8 }
```

## Deletar um usuário

*Request*

```
1 DELETE /rest/users/{id}
```

## API de Professores

API de professores.

### Buscar todos os professores

*Request*

```
1 GET /rest/teachers
```

*Response*

```

1 [
2   {
3     "id": "Integer",
4     "user": {
5       "id": "Integer",
6       "username": "String",
7       "firstname": "String",
8       "lastname": "String",
9       "email": "String",
10      "password": "String",
11      "isactive": "Boolean",
12      "lastlogin": "Date",
13      "datejoined": "Date"
14    },
15    "institution": {
16      "id": "Integer",
17      "name": "String",
18      "initial": "String",
19      "city": {
20        "id": "Integer",
21        "name": "String",
22        "state": {
23          "id": "Integer",
24          "name": "String",
25          "abbreviation": "String",
26          "country": {
27            "id": "Integer",
28            "name": "String",
29            "abbreviation": "String"
30          }
31        }
32      }
33    }
34  ]
35 ]
```

## Detalhes de um professor

*Request*

```
1 GET /rest/teachers/{id}
```

*Response*

```
1 {
2     "id": "Integer",
3     "user": {
4         "id": "Integer",
5         "username": "String",
6         "firstname": "String",
7         "lastname": "String",
8         "email": "String",
9         "password": "String",
10        "isactive": "Boolean",
11        "lastlogin": "Date",
12        "datejoined": "Date"
13    },
14    "institution": {
15        "id": "Integer",
16        "name": "String",
17        "initial": "String",
18        "city": {
19            "id": "Integer",
20            "name": "String",
21            "state": {
22                "id": "Integer",
23                "name": "String",
24                "abbreviation": "String",
25                "country": {
26                    "id": "Integer",
27                    "name": "String",
28                    "abbreviation": "String"
29                }
30            }
31        }
32    }
33 }
```

## Criar um novo professor

*Request*

```
1 POST /rest/teachers
```

```
1 {
2     "user": {
3         "id": "Integer"
4     },
5     "city": {
6         "id": "Integer"
7     }
8 }
```

*Response*

```
1 {
2     "id": "Integer",
3     "user": {
4         "id": "Integer",
5         "username": "String",
6         "firstname": "String",
7         "lastname": "String",
8         "email": "String",
9         "password": "String",
10        "isactive": "Boolean",
11        "lastlogin": "Date",
12        "datejoined": "Date"
13    },
14    "institution": {
15        "id": "Integer",
16        "name": "String",
17        "initial": "String",
18        "city": {
19            "id": "Integer",
20            "name": "String",
21            "state": {
22                "id": "Integer",
23                "name": "String",
24                "abbreviation": "String",
25                "country": {
26                    "id": "Integer",
27                    "name": "String",
28                    "abbreviation": "String"
29                }
30            }
31        }
32    }
33 }
```

## Atualizar um professor

*Request*

```
1 PUT /rest/teachers
```

```
1 {
2     "user": {
3         "id": "Integer"
4     },
5     "city": {
6         "id": "Integer"
7     }
8 }
```

*Response*

```
1 {
2     "id": "Integer",
3     "user": {
4         "id": "Integer",
5         "username": "String",
6         "firstname": "String",
7         "lastname": "String",
8         "email": "String",
9         "password": "String",
10        "isactive": "Boolean",
11        "lastlogin": "Date",
12        "datejoined": "Date"
13    },
14    "institution": {
15        "id": "Integer",
16        "name": "String",
17        "initial": "String",
18        "city": {
19            "id": "Integer",
20            "name": "String",
21            "state": {
22                "id": "Integer",
23                "name": "String",
24                "abbreviation": "String",
25                "country": {
26                    "id": "Integer",
27                    "name": "String",
28                    "abbreviation": "String"
29                }
30            }
31        }
32    }
33 }
```

## Deletar um Professor

*Request*

```
1 DELETE /rest/teachers/{id}
```

## API de Estudantes

API de estudantes.

### Buscar todos os estudantes

*Request*

```
1 GET /rest/students
```

*Response*

```

1 [
2   {
3     "id": "Integer",
4     "user": {
5       "id": "Integer",
6       "username": "String",
7       "firstname": "String",
8       "lastname": "String",
9       "email": "String",
10      "password": "String",
11      "isactive": "Boolean",
12      "lastlogin": "Date",
13      "datejoined": "Date"
14    },
15    "institution": {
16      "id": "Integer",
17      "name": "String",
18      "initial": "String",
19      "city": {
20        "id": "Integer",
21        "name": "String",
22        "state": {
23          "id": "Integer",
24          "name": "String",
25          "abbreviation": "String",
26          "country": {
27            "id": "Integer",
28            "name": "String",
29            "abbreviation": "String"
30          }
31        }
32      }
33    }
34  ]
35 ]
```

## Detalhes de um estudante

*Request*

```
1 GET /rest/students/{id}
```

*Response*

```
1 {
2     "id": "Integer",
3     "user": {
4         "id": "Integer",
5         "username": "String",
6         "firstname": "String",
7         "lastname": "String",
8         "email": "String",
9         "password": "String",
10        "isactive": "Boolean",
11        "lastlogin": "Date",
12        "datejoined": "Date"
13    },
14    "institution": {
15        "id": "Integer",
16        "name": "String",
17        "initial": "String",
18        "city": {
19            "id": "Integer",
20            "name": "String",
21            "state": {
22                "id": "Integer",
23                "name": "String",
24                "abbreviation": "String",
25                "country": {
26                    "id": "Integer",
27                    "name": "String",
28                    "abbreviation": "String"
29                }
30            }
31        }
32    }
33 }
```

## Criar um novo estudante

*Request*

```
1 POST /rest/students
```

```
1 {
2     "user": {
3         "id": "Integer"
4     },
5     "city": {
6         "id": "Integer"
7     }
8 }
```

*Response*

```
1 {
2     "id": "Integer",
3     "user": {
4         "id": "Integer",
5         "username": "String",
6         "firstname": "String",
7         "lastname": "String",
8         "email": "String",
9         "password": "String",
10        "isactive": "Boolean",
11        "lastlogin": "Date",
12        "datejoined": "Date"
13    },
14    "institution": {
15        "id": "Integer",
16        "name": "String",
17        "initial": "String",
18        "city": {
19            "id": "Integer",
20            "name": "String",
21            "state": {
22                "id": "Integer",
23                "name": "String",
24                "abbreviation": "String",
25                "country": {
26                    "id": "Integer",
27                    "name": "String",
28                    "abbreviation": "String"
29                }
30            }
31        }
32    }
33 }
```

## Atualizar um estudante

*Request*

```
1 PUT /rest/students
```

```
1 {
2     "user": {
3         "id": "Integer"
4     },
5     "city": {
6         "id": "Integer"
7     }
8 }
```

*Response*

```
1 {
2     "id": "Integer",
3     "user": {
4         "id": "Integer",
5         "username": "String",
6         "firstname": "String",
7         "lastname": "String",
8         "email": "String",
9         "password": "String",
10        "isactive": "Boolean",
11        "lastlogin": "Date",
12        "datejoined": "Date"
13    },
14    "institution": {
15        "id": "Integer",
16        "name": "String",
17        "initial": "String",
18        "city": {
19            "id": "Integer",
20            "name": "String",
21            "state": {
22                "id": "Integer",
23                "name": "String",
24                "abbreviation": "String",
25                "country": {
26                    "id": "Integer",
27                    "name": "String",
28                    "abbreviation": "String"
29                }
30            }
31        }
32    }
33 }
```

## Deletar um estudante

*Request*

```
1 DELETE /rest/students/{id}
```

## API de Permissões de Usuários

API de permissões de usuários.

### Buscar todas as permissões de um usuário

*Request*

```
1 GET /rest/userpermission/filter-by-user-id/{user-id}
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "permission": {
5       "id": "Integer"
6     },
7     "portaluser": {
8       "id": "Integer"
9     }
10   }
11 ]
```

### Adicionar permissões para um usuário

*Request*

```
1 POST /rest/userpermission/{user-id}
```

```
1 [
2   {
3     "permission": {
4       "id": 1
5     },
6     "portaluser": {
7       "id": 5
8     }
9   }
10 ]
```

## API de Grupos de Estudantes

API de grupos de estudantes.

### Buscar todos os grupos de um estudante

*Request*

```
1 GET /rest/studentgroups/filter-by-list-student-id/{student-id}
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "student": {
5       "id": "Integer"
6     },
7     "group": {
8       "id": "Integer"
9     }
10   }
11 ]
```

### Adiciona grupos para um estudante

*Request*

```
1 POST /rest/studentgroups/{student-id}
```

```
1 [
2   {
3     "student": {
4       "id": "Integer"
5     },
6     "group": {
7       "id": "Integer"
8     }
9   }
10 ]
```

## API de Listas de Problemas de um Estudante

API de listas de problemas de um estudante

### Buscar todas as listas de problemas de um estudante

*Request*

```
1 GET /rest/studentproblemList/filter-by-list-student-id/{student-id}
```

```
1 [
2   {
3     "id": "Integer",
4     "listActivity": {
5       "id": "Integer"
6     },
7     "student": {
8       "id": "Integer"
9     }
10   }
11 ]
```

### Adicionar listas de problemas para um estudante

*Request*

```
1 POST /rest/studentproblemList/{student-id}
```

```
1 [
2   {
3     "listActivity": {
4       "id": "Integer"
5     },
6     "student": {
7       "id": "Integer"
8     }
9   }
10 ]
```

## API de Grupo de Administradores

API de grupo de administradores.

### Buscar todos os grupos de administradores

*Request*

```
1 GET /rest/groupadmins
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "name": "String"
5   }
6 ]
```

### Detalhes de um grupo de administradores

*Request*

```
1 GET /rest/groupadmins/{id}
```

*Response*

```
1 {
2   "id": "Integer",
3   "name": "String"
4 }
```

### Criar um novo grupo de administradores

*Request*

```
1 POST /rest/groupadmins
```

```
1 {
2   "name": "String"
3 }
```

*Response*

```
1 {  
2     "id": "Integer",  
3     "name": "String"  
4 }
```

## Atualizar um grupo de administradores

*Request*

```
1 PUT /rest/groupadmins
```

```
1 {  
2     "id": "Integer",  
3     "name": "String"  
4 }
```

*Response*

```
1 {  
2     "id": "Integer",  
3     "name": "String"  
4 }
```

## Deletar um grupo de administradores

*Request*

```
1 DELETE /rest/groupadmins/{id}
```

## API de Permissões de um Grupo de Administradores

API de permissões de um grupo de administradores

### Buscar todas as permissões de um grupo de administradores

*Request*

```
1 GET /rest/groupadminpermission/filter-by-groupadmin-id/{group-admin-id}
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "permission": {
5       "id": "Integer"
6     },
7     "groupadmin": {
8       "id": "Integer"
9     }
10   }
11 ]
```

### Adicionar permissões para um grupo de administradores

*Request*

```
1 POST /rest/groupadminpermission/{group-admin-id}
```

```
1 [
2   {
3     "permission": {
4       "id": "Integer"
5     },
6     "groupadmin": {
7       "id": "Integer"
8     }
9   }
10 ]
```

## API de Usuários de um Grupo de Administradores

API de usuários de um grupo de administradores

### Buscar todas os usuários de um grupo de administradores

*Request*

```
1 GET /rest/groupadminuser/filter-by-groupadmin-id/{group-admin-id}
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "user": {
5       "id": "Integer"
6     },
7     "groupadmin": {
8       "id": "Integer"
9     }
10  }
11 ]
```

## Adicionar usuários para um grupo de administradores

*Request*

```
1 POST /rest/groupadminuser/{group-admin-id}
```

```
1 [
2   {
3     "user": {
4       "id": "Integer"
5     },
6     "groupadmin": {
7       "id": "Integer"
8     }
9   }
10 ]
```

## API de Grupos

API de grupos.

### Buscar todos os grupos

*Request*

```
1 GET /rest/groups
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "name": "String",
5     "description": "String",
6     "teacher": {
7       "id": "Integer"
8     }
9   }
10 ]
```

### Detalhes de um grupo

*Request*

```
1 GET /rest/groups/{id}
```

*Response*

```
1 {
2   "id": "Integer",
3   "name": "String",
4   "description": "String",
5   "teacher": {
6     "id": "Integer"
7   }
8 }
```

## Criar um novo grupo

*Request*

```
1 POST /rest/groups
```

```
1 {
2   "name": "String",
3   "description": "String",
4   "teacher": {
5     "id": "Integer"
6   }
7 }
```

*Response*

```
1 {
2   "id": "Integer",
3   "name": "String",
4   "description": "String",
5   "teacher": {
6     "id": "Integer"
7   }
8 }
```

## Atualizar um grupo

*Request*

```
1 PUT /rest/groups
```

```
1 {
2   "id": "Integer",
3   "name": "String",
4   "description": "String",
5   "teacher": {
6     "id": "Integer"
7   }
8 }
```

*Response*

```
1 {
2   "id": "Integer",
3   "name": "String",
4   "description": "String",
5   "teacher": {
6     "id": "Integer"
7   }
8 }
```

## Deletar um grupo

*Request*

```
1 DELETE /rest/groups/{id}
```

## API de Listas de Problemas de Grupos

API de listas de problemas de grupos.

### Buscar todas as listas de problemas de um grupo

*Request*

```
1 GET /rest/groupproblemfilter/by-group-id/{group-id}
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "listActivity": {
5       "id": "Integer"
6     },
7     "group": {
8       "id": "Integer"
9     }
10   }
11 ]
```

### Adicionar listas de problemas para um grupo

*Request*

```
1 POST /rest/groupproblemfilter/{group-id}
```

```
1 [
2   {
3     "listActivity": {
4       "id": "Integer"
5     },
6     "group": {
7       "id": "Integer"
8     }
9   }
10 ]
```

## API de Tipos de Problemas

API de tipos de problemas.

### Buscar todos os tipos de problemas

*Request*

```
1 GET /rest/problems
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "name": "String",
5   }
6 ]
```

### Detalhes de um tipo de problema

*Request*

```
1 GET /rest/problems/{id}
```

*Response*

```
1 {
2   "id": "Integer",
3   "name": "String",
4 }
```

### Criar um novo tipo de problema

*Request*

```
1 POST /rest/problems
```

```
1 {
2   "name": "String",
3 }
```

*Response*

```
1 {  
2     "id": "Integer",  
3     "name": "String",  
4 }
```

## Atualizar um tipo de problema

*Request*

```
1 PUT /rest/problemtypes
```

```
1 {  
2     "id": "Integer",  
3     "name": "String",  
4 }
```

*Response*

```
1 {  
2     "id": "Integer",  
3     "name": "String",  
4 }
```

## Deletar um tipo de problema

```
1 DELETE /rest/problemtypes/{id}
```

## API de Problemas

API de problemas.

### Buscar todos os problemas

*Request*

```
1 GET /rest/problems
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "name": "String",
5     "problemtype": {
6       "id": "Integer",
7       "name": "String"
8     },
9     "description": "String",
10    "tip": "String",
11    "createdat": "Timestamp",
12    "level": "String",
13    "cost": "Integer"
14  }
15 ]
```

### Detalhes de um problema

*Request*

```
1 GET /rest/problems/{id}
```

*Response*

```
1 {
2   "id": "Integer",
3   "name": "String",
4   "problemtype": {
5     "id": "Integer",
6     "name": "String"
7   },
8   "description": "String",
9   "tip": "String",
10  "createdat": "Timestamp",
11  "level": "String",
12  "cost": "Integer"
13 }
```

## Criar um novo problema

*Request*

```
1 POST /rest/problems
```

```
1 {
2     "name": "String",
3     "description": "String",
4     "tip": "String",
5     "level": "String",
6     "cost": "Integer",
7     "problemtype": {
8         "id": "Integer",
9     }
10 }
```

*Response*

```
1 {
2     "id": "Integer",
3     "name": "String",
4     "description": "String",
5     "tip": "String",
6     "createdat": "Timestamp",
7     "level": "String",
8     "cost": "Integer",
9     "problemtype": {
10         "id": "Integer",
11         "name": "String"
12     }
13 }
```

## Atualizar um problema

*Request*

```
1 PUT /rest/problems
```

```
1 {
2     "id": "Integer",
3     "name": "String",
4     "description": "String",
5     "tip": "String",
6     "level": "String",
7     "cost": "Integer",
8     "problemtype": {
9         "id": "Integer",
10    }
11 }
```

*Response*

```

1 {
2   "id": "Integer",
3   "name": "String",
4   "description": "String",
5   "tip": "String",
6   "createdat": "Timestamp",
7   "level": "String",
8   "cost": "Integer",
9   "problemtype": {
10     "id": "Integer",
11     "name": "String"
12   }
13 }
```

## Deletar um problema

*Request*

```
1 DELETE /rest/problems/{id}
```

## API de Dados de Teste de Problema

API de dados de teste de problema.

### Buscar todos os dados de teste de um problema

*Request*

```
1 GET /rest/testdatas/filter-by-problem-id/{problem-id}
```

*Response*

```

1 [
2   {
3     "id": "Integer",
4     "input": "String",
5     "output": "String",
6     "problem": {
7       "id": "Integer"
8     }
9   }
10 ]
```

## Adicionar dados de teste para um problema

*Request*

```
1 POST /rest/testdatas/{problem-id}
```

```
1 [
2   {
3     "listActivity": {
4       "id": "Integer"
5     },
6     "group": {
7       "id": "Integer"
8     }
9   }
10 ]
```

## API de Palavras Chaves

API de palavras chaves.

### Buscar todas as palavras chaves

*Request*

```
1 GET /rest/keywords
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "name": "String",
5   }
6 ]
```

### Detalhes de uma palavra chave

*Request*

```
1 GET /rest/keywords/{id}
```

*Response*

```
1 {
2   "id": "Integer",
3   "name": "String",
4 }
```

## Criar uma nova palavra chave

*Request*

```
1 POST /rest/keywords
```

```
1 {
2     "name": "String"
3 }
```

*Response*

```
1 {
2     "id": "Integer",
3     "name": "String",
4 }
```

## Atualizar uma palavra chave

*Request*

```
1 PUT /rest/keywords
```

```
1 {
2     "id": "Integer",
3     "name": "String"
4 }
```

*Response*

```
1 {
2     "id": "Integer",
3     "name": "String"
4 }
```

## Deletar uma palavra chave

*Request*

```
1 DELETE /rest/keywords/{id}
```

## API de Palavras Chaves no Problemas

API de palavras chaves no problemas.

### Buscar todas as palavras chaves de um problema

*Request*

```
1 GET /rest/problemkeywords/filter-by-problem-id/{problem-id}
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "keyword": {
5       "id": "Integer"
6     },
7     "problem": {
8       "id": "Integer"
9     }
10  }
11 ]
```

### Adicionar palavras chaves para um problema

*Request*

```
1 POST /rest/problemkeywords/{problem-id}
```

```
1 [
2   {
3     "keyword": {
4       "id": "Integer"
5     },
6     "problem": {
7       "id": "Integer"
8     }
9   }
10 ]
```

## API de Lista de Atividades

API de lista de atividades.

### Buscar todas as listas de atividades

*Request*

```
1 GET /rest/listactivities
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "name": "String",
5   }
6 ]
```

### Detalhes de uma lista de atividades

*Request*

```
1 GET /rest/listactivities/{id}
```

*Response*

```
1 {
2   "id": "Integer",
3   "name": "String",
4 }
```

### Criar uma nova lista de atividade

*Request*

```
1 POST /rest/listactivities
```

```
1 {
2   "name": "String",
3 }
```

*Response*

```
1 {  
2     "id": "Integer",  
3     "name": "String",  
4 }
```

## Atualizar uma lista de atividade

*Request*

```
1 PUT /rest/listactivities
```

```
1 {  
2     "id": "Integer",  
3     "name": "String",  
4 }
```

*Response*

```
1 {  
2     "id": "Integer",  
3     "name": "String",  
4 }
```

## Deletar uma lista de atividade

*Request*

```
1 DELETE /rest/listactivities/{id}
```

## API de Lista de Problemas de uma Lista de Atividades

API de lista de problemas de uma lista de atividades.

### Buscar todas os problemas de todas lista de atividades

*Request*

```
1 GET /rest/problemlists
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "problem": {
5       "id": "Integer",
6       "name": "String",
7     },
8     "listactivity": {
9       "id": "Integer",
10      "name": "String"
11    }
12  }
13 ]
```

### Buscar todas os problemas de uma lista de atividades

*Request*

```
1 GET /rest/problemlists/filter-by-list-activity-id/{activity-id}
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "problem": {
5       "id": "Integer",
6       "name": "String",
7     },
8     "listactivity": {
9       "id": "Integer",
10      "name": "String"
11    }
12  }
13 ]
```

## Adicionar problemas para uma lista de atividade

*Request*

```
1 POST /rest/problemlists/{activity-id}
```

*Response*

```
1 [
2   {
3     "problem": {
4       "id": "Integer"
5     },
6     "listactivity": {
7       "id": "Integer"
8     }
9   }
10 ]
```

## API de Soluções de Problemas

API de soluções de problemas.

### Buscar todos as soluções de problemas

*Request*

```
1 GET /rest/problemsolutions
```

*Response*

```
1 [
2   {
3     "id": "Integer",
4     "solution": "String",
5     "name": "String",
6     "cost": null,
7     "datesolution": "Timestamp",
8     "result": "String",
9     "user": {
10       "id": "Integer",
11       "username": "String",
12     },
13     "problem": {
14       "id": "Integer",
15       "name": "String",
16     }
17   }
18 ]
```

## Detalhes de uma solução de problema

*Request*

```
1 GET /rest/problemsolutions/{id}
```

*Response*

```
1 {
2     "id": "Integer",
3     "solution": "String",
4     "name": "String",
5     "cost": null,
6     "datesolution": "Timestamp",
7     "result": "String",
8     "user": {
9         "id": "Integer",
10        "username": "String",
11    },
12    "problem": {
13        "id": "Integer",
14        "name": "String",
15    }
16 }
```

## Criar uma nova solução de problema

*Request*

```
1 POST /rest/problemsolutions
```

```
1 {
2     "solution": "String",
3     "name": "String",
4     "user": {
5         "id": "Integer"
6     },
7     "problem": {
8         "id": "Integer"
9     }
10 }
```

*Response*

```

1 {
2     "id": "Integer",
3     "solution": "String",
4     "name": "String",
5     "cost": null,
6     "datesolution": "Timestamp",
7     "result": "String",
8     "user": {
9         "id": "Integer",
10        "username": "String",
11    },
12    "problem": {
13        "id": "Integer",
14        "name": "String",
15    }
16 }
```

**Atualizar uma solução de problema***Request*

```
1 PUT /rest/problemsolutions
```

```

1 {
2     "id": "Integer",
3     "solution": "String",
4     "name": "String",
5     "user": {
6         "id": "Integer"
7     },
8     "problem": {
9         "id": "Integer"
10    }
11 }
```

*Response*

```
1 {
2     "id": "Integer",
3     "solution": "String",
4     "name": "String",
5     "cost": null,
6     "datesolution": "Timestamp",
7     "result": "String",
8     "user": {
9         "id": "Integer",
10        "username": "String",
11    },
12    "problem": {
13        "id": "Integer",
14        "name": "String",
15    }
16 }
```

**Deletar uma solução de problema***Request*

```
1 DELETE /rest/problemsolutions/{id}
```

## APÊNDICE C - SQL DE CRIAÇÃO DO BANCO DE DADOS

*Script* de criação das tabelas do banco de dados.

```
1 --
2 -- PostgreSQL database dump
3 --
4
5 -- Dumped from database version 9.6.8
6 -- Dumped by pg_dump version 9.6.8
7
8 -- Started on 2018-06-12 19:41:20 -03
9
10 SET statement_timeout = 0;
11 SET lock_timeout = 0;
12 SET idle_in_transaction_session_timeout = 0;
13 SET client_encoding = 'UTF8';
14 SET standard_conforming_strings = on;
15 SELECT pg_catalog.set_config('search_path', '', false);
16 SET check_function_bodies = false;
17 SET client_min_messages = warning;
18 SET row_security = off;
19
20 --
21 -- TOC entry 1 (class 3079 OID 13368)
22 -- Name: plpgsql; Type: EXTENSION; Schema: -; Owner:
23 --
24
25 CREATE EXTENSION IF NOT EXISTS plpgsql WITH SCHEMA pg_catalog;
26
27
28 --
29 -- TOC entry 3404 (class 0 OID 0)
30 -- Dependencies: 1
31 -- Name: EXTENSION plpgsql; Type: COMMENT; Schema: -; Owner:
32 --
33
34 COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural language';
35
36
37 SET default_tablespace = '';
38
39 SET default_with_oids = false;
40
41 --
42 -- TOC entry 200 (class 1259 OID 235732)
43 -- Name: admins; Type: TABLE; Schema: public; Owner: postgres
```

```
44 --
45
46 CREATE TABLE public.admins (
47   id integer NOT NULL,
48   user_id integer NOT NULL,
49   groupadmin_id integer NOT NULL
50 );
51
52
53 ALTER TABLE public.admins OWNER TO postgres;
54
55 --
56 -- TOC entry 199 (class 1259 OID 235730)
57 -- Name: admins_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
58 --
59
60 CREATE SEQUENCE public.admins_id_seq
61 START WITH 1
62 INCREMENT BY 1
63 NO MINVALUE
64 NO MAXVALUE
65 CACHE 1;
66
67
68 ALTER TABLE public.admins_id_seq OWNER TO postgres;
69
70 --
71 -- TOC entry 3405 (class 0 OID 0)
72 -- Dependencies: 199
73 -- Name: admins_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
74 --
75
76 ALTER SEQUENCE public.admins_id_seq OWNED BY public.admins.id;
77
78
79 --
80 -- TOC entry 190 (class 1259 OID 235659)
81 -- Name: city; Type: TABLE; Schema: public; Owner: postgres
82 --
83
84 CREATE TABLE public.city (
85   id integer NOT NULL,
86   name character varying(254) NOT NULL,
87   state_id integer NOT NULL
88 );
89
90
91 ALTER TABLE public.city OWNER TO postgres;
92
93 --
94 -- TOC entry 189 (class 1259 OID 235657)
95 -- Name: city_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
96 --
97
98 CREATE SEQUENCE public.city_id_seq
99 START WITH 1
100 INCREMENT BY 1
101 NO MINVALUE
102 NO MAXVALUE
```

```
103 CACHE 1;
104
105
106 ALTER TABLE public.city_id_seq OWNER TO postgres;
107
108 --
109 -- TOC entry 3406 (class 0 OID 0)
110 -- Dependencies: 189
111 -- Name: city_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
112 --
113
114 ALTER SEQUENCE public.city_id_seq OWNED BY public.city.id;
115
116
117 --
118 -- TOC entry 186 (class 1259 OID 235624)
119 -- Name: country; Type: TABLE; Schema: public; Owner: postgres
120 --
121
122 CREATE TABLE public.country (
123 id integer NOT NULL,
124 name character varying(254) NOT NULL,
125 abbreviation character varying(2) NOT NULL
126 );
127
128
129 ALTER TABLE public.country OWNER TO postgres;
130
131 --
132 -- TOC entry 185 (class 1259 OID 235622)
133 -- Name: country_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
134 --
135
136 CREATE SEQUENCE public.country_id_seq
137 START WITH 1
138 INCREMENT BY 1
139 NO MINVALUE
140 NO MAXVALUE
141 CACHE 1;
142
143
144 ALTER TABLE public.country_id_seq OWNER TO postgres;
145
146 --
147 -- TOC entry 3407 (class 0 OID 0)
148 -- Dependencies: 185
149 -- Name: country_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
      postgres
150 --
151
152 ALTER SEQUENCE public.country_id_seq OWNED BY public.country.id;
153
154
155 --
156 -- TOC entry 227 (class 1259 OID 342475)
157 -- Name: group_problemlist_id_seq; Type: SEQUENCE; Schema: public; Owner:
      postgres
158 --
159
```

```

160 CREATE SEQUENCE public.group_problemlist_id_seq
161 START WITH 1
162 INCREMENT BY 1
163 NO MINVALUE
164 NO MAXVALUE
165 CACHE 1;
166
167
168 ALTER TABLE public.group_problemlist_id_seq OWNER TO postgres;
169
170 --
171 -- TOC entry 223 (class 1259 OID 342431)
172 -- Name: group_problemlist; Type: TABLE; Schema: public; Owner: postgres
173 --
174
175 CREATE TABLE public.group_problemlist (
176 id integer DEFAULT nextval('public.group_problemlist_id_seq'::regclass) NOT NULL
177 ,
178 group_id integer NOT NULL,
179 problemlist_id integer NOT NULL
179 );
180
181
182 ALTER TABLE public.group_problemlist OWNER TO postgres;
183
184 --
185 -- TOC entry 198 (class 1259 OID 235708)
186 -- Name: groupadmin; Type: TABLE; Schema: public; Owner: postgres
187 --
188
189 CREATE TABLE public.groupadmin (
190 id integer NOT NULL,
191 name character varying(254) NOT NULL
192 );
193
194
195 ALTER TABLE public.groupadmin OWNER TO postgres;
196
197 --
198 -- TOC entry 197 (class 1259 OID 235706)
199 -- Name: groupadmin_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
200 --
201
202 CREATE SEQUENCE public.groupadmin_id_seq
203 START WITH 1
204 INCREMENT BY 1
205 NO MINVALUE
206 NO MAXVALUE
207 CACHE 1;
208
209
210 ALTER TABLE public.groupadmin_id_seq OWNER TO postgres;
211
212 --
213 -- TOC entry 3408 (class 0 OID 0)
214 -- Dependencies: 197
215 -- Name: groupadmin_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
216             postgres
216 --

```

```

217
218 ALTER SEQUENCE public.groupadmin_id_seq OWNED BY public.groupadmin.id;
219
220
221 --
222 -- TOC entry 226 (class 1259 OID 342456)
223 -- Name: groupadmin_permission_id_seq; Type: SEQUENCE; Schema: public; Owner:
      postgres
224 --
225
226 CREATE SEQUENCE public.groupadmin_permission_id_seq
227 START WITH 1
228 INCREMENT BY 1
229 NO MINVALUE
230 NO MAXVALUE
231 CACHE 1;
232
233
234 ALTER TABLE public.groupadmin_permission_id_seq OWNER TO postgres;
235
236 --
237 -- TOC entry 225 (class 1259 OID 342443)
238 -- Name: groupadmin_permission; Type: TABLE; Schema: public; Owner: postgres
239 --
240
241 CREATE TABLE public.groupadmin_permission (
242 id integer DEFAULT nextval('public.groupadmin_permission_id_seq'::regclass) NOT
      NULL,
243 groupadmin_id integer NOT NULL,
244 permission_id integer NOT NULL
245 );
246
247
248 ALTER TABLE public.groupadmin_permission OWNER TO postgres;
249
250 --
251 -- TOC entry 192 (class 1259 OID 235671)
252 -- Name: institution; Type: TABLE; Schema: public; Owner: postgres
253 --
254
255 CREATE TABLE public.institution (
256 id integer NOT NULL,
257 name character varying(254) NOT NULL,
258 city_id integer NOT NULL,
259 initial character varying(10) NOT NULL
260 );
261
262
263 ALTER TABLE public.institution OWNER TO postgres;
264
265 --
266 -- TOC entry 191 (class 1259 OID 235669)
267 -- Name: institution_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
268 --
269
270 CREATE SEQUENCE public.institution_id_seq
271 START WITH 1
272 INCREMENT BY 1
273 NO MINVALUE

```

```
274 NO MAXVALUE
275 CACHE 1;
276
277
278 ALTER TABLE public.institution_id_seq OWNER TO postgres;
279
280 --
281 -- TOC entry 3409 (class 0 OID 0)
282 -- Dependencies: 191
283 -- Name: institution_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
284     postgres
285 --
286 ALTER SEQUENCE public.institution_id_seq OWNED BY public.institution.id;
287
288
289 --
290 -- TOC entry 204 (class 1259 OID 235756)
291 -- Name: keyword; Type: TABLE; Schema: public; Owner: postgres
292 --
293
294 CREATE TABLE public.keyword (
295 id integer NOT NULL,
296 name character varying(254) NOT NULL
297 );
298
299
300 ALTER TABLE public.keyword OWNER TO postgres;
301
302 --
303 -- TOC entry 203 (class 1259 OID 235754)
304 -- Name: keyword_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
305 --
306
307 CREATE SEQUENCE public.keyword_id_seq
308 START WITH 1
309 INCREMENT BY 1
310 NO MINVALUE
311 NO MAXVALUE
312 CACHE 1;
313
314
315 ALTER TABLE public.keyword_id_seq OWNER TO postgres;
316
317 --
318 -- TOC entry 3410 (class 0 OID 0)
319 -- Dependencies: 203
320 -- Name: keyword_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
321     postgres
322
323 ALTER SEQUENCE public.keyword_id_seq OWNED BY public.keyword.id;
324
325
326 --
327 -- TOC entry 222 (class 1259 OID 341024)
328 -- Name: listactivity_id_seq; Type: SEQUENCE; Schema: public; Owner: portalg
329 --
330
```

```

331 CREATE SEQUENCE public.listactivity_id_seq
332 START WITH 1
333 INCREMENT BY 1
334 NO MINVALUE
335 NO MAXVALUE
336 CACHE 1;
337
338
339 ALTER TABLE public.listactivity_id_seq OWNER TO portalg;
340
341 --
342 -- TOC entry 221 (class 1259 OID 341015)
343 -- Name: listactivity; Type: TABLE; Schema: public; Owner: postgres
344 --
345
346 CREATE TABLE public.listactivity (
347 id integer DEFAULT nextval('public.listactivity_id_seq'::regclass) NOT NULL,
348 name character varying(254) NOT NULL
349 );
350
351
352 ALTER TABLE public.listactivity OWNER TO postgres;
353
354 --
355 -- TOC entry 196 (class 1259 OID 235697)
356 -- Name: permission; Type: TABLE; Schema: public; Owner: postgres
357 --
358
359 CREATE TABLE public.permission (
360 id integer NOT NULL,
361 codename character varying(75) NOT NULL,
362 description character varying(254)
363 );
364
365
366 ALTER TABLE public.permission OWNER TO postgres;
367
368 --
369 -- TOC entry 195 (class 1259 OID 235695)
370 -- Name: permission_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
371 --
372
373 CREATE SEQUENCE public.permission_id_seq
374 START WITH 1
375 INCREMENT BY 1
376 NO MINVALUE
377 NO MAXVALUE
378 CACHE 1;
379
380
381 ALTER TABLE public.permission_id_seq OWNER TO postgres;
382
383 --
384 -- TOC entry 3411 (class 0 OID 0)
385 -- Dependencies: 195
386 -- Name: permission_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
            postgres
387 --
388

```

```
389 ALTER SEQUENCE public.permission_id_seq OWNED BY public.permission.id;
390
391
392 --
393 -- TOC entry 216 (class 1259 OID 235826)
394 -- Name: portalgroup; Type: TABLE; Schema: public; Owner: postgres
395 --
396
397 CREATE TABLE public.portalgroup (
398 id integer NOT NULL,
399 name character varying(254) NOT NULL,
400 description character varying(1000),
401 teacher_id integer NOT NULL
402 );
403
404
405 ALTER TABLE public.portalgroup OWNER TO postgres;
406
407 --
408 -- TOC entry 215 (class 1259 OID 235824)
409 -- Name: portalgroup_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
410 --
411
412 CREATE SEQUENCE public.portalgroup_id_seq
413 START WITH 1
414 INCREMENT BY 1
415 NO MINVALUE
416 NO MAXVALUE
417 CACHE 1;
418
419
420 ALTER TABLE public.portalgroup_id_seq OWNER TO postgres;
421
422 --
423 -- TOC entry 3412 (class 0 OID 0)
424 -- Dependencies: 215
425 -- Name: portalgroup_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
        postgres
426 --
427
428 ALTER SEQUENCE public.portalgroup_id_seq OWNED BY public.portalgroup.id;
429
430
431 --
432 -- TOC entry 194 (class 1259 OID 235683)
433 -- Name: portaluser; Type: TABLE; Schema: public; Owner: postgres
434 --
435
436 CREATE TABLE public.portaluser (
437 id integer NOT NULL,
438 username character varying(254) NOT NULL,
439 firstname character varying(254),
440 lastname character varying(254),
441 email character varying(254) NOT NULL,
442 password character varying(254) NOT NULL,
443 isactive boolean,
444 lastlogin timestamp without time zone,
445 datejoined timestamp without time zone
446 );
```

```

447
448
449 ALTER TABLE public.portaluser OWNER TO postgres;
450
451 --
452 -- TOC entry 193 (class 1259 OID 235681)
453 -- Name: portaluser_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
454 --
455
456 CREATE SEQUENCE public.portaluser_id_seq
457 START WITH 1
458 INCREMENT BY 1
459 NO MINVALUE
460 NO MAXVALUE
461 CACHE 1;
462
463
464 ALTER TABLE public.portaluser_id_seq OWNER TO postgres;
465
466 --
467 -- TOC entry 3413 (class 0 OID 0)
468 -- Dependencies: 193
469 -- Name: portaluser_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
        postgres
470 --
471
472 ALTER SEQUENCE public.portaluser_id_seq OWNED BY public.portaluser.id;
473
474 --
475 --
476 -- TOC entry 230 (class 1259 OID 350624)
477 -- Name: portaluser_permission_id_seq; Type: SEQUENCE; Schema: public; Owner:
        postgres
478 --
479
480 CREATE SEQUENCE public.portaluser_permission_id_seq
481 START WITH 1
482 INCREMENT BY 1
483 NO MINVALUE
484 NO MAXVALUE
485 CACHE 1;
486
487
488 ALTER TABLE public.portaluser_permission_id_seq OWNER TO postgres;
489
490 --
491 -- TOC entry 231 (class 1259 OID 350656)
492 -- Name: portaluser_permission; Type: TABLE; Schema: public; Owner: postgres
493 --
494
495 CREATE TABLE public.portaluser_permission (
496 id integer DEFAULT nextval('public.portaluser_permission_id_seq'::regclass) NOT
        NULL,
497 portaluser_id integer NOT NULL,
498 permission_id integer NOT NULL
499 );
500
501
502 ALTER TABLE public.portaluser_permission OWNER TO postgres;

```

```

503
504 --
505 -- TOC entry 208 (class 1259 OID 235774)
506 -- Name: problem; Type: TABLE; Schema: public; Owner: postgres
507 --
508
509 CREATE TABLE public.problem (
510 id integer NOT NULL,
511 name character varying(254) NOT NULL,
512 tip character varying(254),
513 createdat timestamp without time zone,
514 level character varying(75),
515 cost integer,
516 problemtyp_id integer NOT NULL,
517 description text NOT NULL
518 );
519
520
521 ALTER TABLE public.problem OWNER TO postgres;
522
523 --
524 -- TOC entry 207 (class 1259 OID 235772)
525 -- Name: problem_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
526 --
527
528 CREATE SEQUENCE public.problem_id_seq
529 START WITH 1
530 INCREMENT BY 1
531 NO MINVALUE
532 NO MAXVALUE
533 CACHE 1;
534
535
536 ALTER TABLE public.problem_id_seq OWNER TO postgres;
537
538 --
539 -- TOC entry 3414 (class 0 OID 0)
540 -- Dependencies: 207
541 -- Name: problem_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
      postgres
542 --
543
544 ALTER SEQUENCE public.problem_id_seq OWNED BY public.problem.id;
545
546
547 --
548 -- TOC entry 220 (class 1259 OID 235850)
549 -- Name: problemkeyword; Type: TABLE; Schema: public; Owner: postgres
550 --
551
552 CREATE TABLE public.problemkeyword (
553 id integer NOT NULL,
554 keyword_id integer NOT NULL,
555 problem_id integer NOT NULL
556 );
557
558
559 ALTER TABLE public.problemkeyword OWNER TO postgres;
560

```

```

561 --
562 -- TOC entry 219 (class 1259 OID 235848)
563 -- Name: problemkeyword_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
564 --
565
566 CREATE SEQUENCE public.problemkeyword_id_seq
567 START WITH 1
568 INCREMENT BY 1
569 NO MINVALUE
570 NO MAXVALUE
571 CACHE 1;
572
573
574 ALTER TABLE public.problemkeyword_id_seq OWNER TO postgres;
575
576 --
577 -- TOC entry 3415 (class 0 OID 0)
578 -- Dependencies: 219
579 -- Name: problemkeyword_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
      postgres
580 --
581
582 ALTER SEQUENCE public.problemkeyword_id_seq OWNED BY public.problemkeyword.id;
583
584
585 --
586 -- TOC entry 214 (class 1259 OID 235814)
587 -- Name: problemlist; Type: TABLE; Schema: public; Owner: postgres
588 --
589
590 CREATE TABLE public.problemlist (
591 id integer NOT NULL,
592 problem_id integer NOT NULL,
593 listactivity_id integer NOT NULL
594 );
595
596
597 ALTER TABLE public.problemlist OWNER TO postgres;
598
599 --
600 -- TOC entry 213 (class 1259 OID 235812)
601 -- Name: problemlist_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
602 --
603
604 CREATE SEQUENCE public.problemlist_id_seq
605 START WITH 1
606 INCREMENT BY 1
607 NO MINVALUE
608 NO MAXVALUE
609 CACHE 1;
610
611
612 ALTER TABLE public.problemlist_id_seq OWNER TO postgres;
613
614 --
615 -- TOC entry 3416 (class 0 OID 0)
616 -- Dependencies: 213
617 -- Name: problemlist_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
      postgres

```

```
618 --
619
620 ALTER SEQUENCE public.problemlist_id_seq OWNED BY public.problemlist.id;
621
622
623 --
624 -- TOC entry 212 (class 1259 OID 235799)
625 -- Name: problemsolution; Type: TABLE; Schema: public; Owner: postgres
626 --
627
628 CREATE TABLE public.problemsolution (
629 id integer NOT NULL,
630 solution character varying NOT NULL,
631 name character varying(254) NOT NULL,
632 cost integer,
633 datesolution timestamp without time zone,
634 result character varying(254),
635 user_id integer NOT NULL,
636 problem_id integer NOT NULL
637 );
638
639
640 ALTER TABLE public.problemsolution OWNER TO postgres;
641
642 --
643 -- TOC entry 211 (class 1259 OID 235797)
644 -- Name: problemsolution_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
645 --
646
647 CREATE SEQUENCE public.problemsolution_id_seq
648 START WITH 1
649 INCREMENT BY 1
650 NO MINVALUE
651 NO MAXVALUE
652 CACHE 1;
653
654
655 ALTER TABLE public.problemsolution_id_seq OWNER TO postgres;
656
657 --
658 -- TOC entry 3417 (class 0 OID 0)
659 -- Dependencies: 211
660 -- Name: problemsolution_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
       postgres
661 --
662
663 ALTER SEQUENCE public.problemsolution_id_seq OWNED BY public.problemsolution.id;
664
665
666 --
667 -- TOC entry 206 (class 1259 OID 235765)
668 -- Name: problemtyp; Type: TABLE; Schema: public; Owner: postgres
669 --
670
671 CREATE TABLE public.problemtyp (
672 id integer NOT NULL,
673 name character varying(254) NOT NULL
674 );
675
```

```
676
677 ALTER TABLE public.problemtype OWNER TO postgres;
678
679 --
680 -- TOC entry 205 (class 1259 OID 235763)
681 -- Name: problemtype_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
682 --
683
684 CREATE SEQUENCE public.problemtype_id_seq
685 START WITH 1
686 INCREMENT BY 1
687 NO MINVALUE
688 NO MAXVALUE
689 CACHE 1;
690
691
692 ALTER TABLE public.problemtype_id_seq OWNER TO postgres;
693
694 --
695 -- TOC entry 3418 (class 0 OID 0)
696 -- Dependencies: 205
697 -- Name: problemtype_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
       postgres
698 --
699
700 ALTER SEQUENCE public.problemtype_id_seq OWNED BY public.problemtype.id;
701
702
703 --
704 -- TOC entry 188 (class 1259 OID 235635)
705 -- Name: state; Type: TABLE; Schema: public; Owner: postgres
706 --
707
708 CREATE TABLE public.state (
709 id integer NOT NULL,
710 name character varying(254) NOT NULL,
711 abbreviation character varying(3) NOT NULL,
712 country_id integer NOT NULL
713 );
714
715
716 ALTER TABLE public.state OWNER TO postgres;
717
718 --
719 -- TOC entry 187 (class 1259 OID 235633)
720 -- Name: state_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
721 --
722
723 CREATE SEQUENCE public.state_id_seq
724 START WITH 1
725 INCREMENT BY 1
726 NO MINVALUE
727 NO MAXVALUE
728 CACHE 1;
729
730
731 ALTER TABLE public.state_id_seq OWNER TO postgres;
732
733 --
```

```

734 -- TOC entry 3419 (class 0 OID 0)
735 -- Dependencies: 187
736 -- Name: state_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
737 --
738
739 ALTER SEQUENCE public.state_id_seq OWNED BY public.state.id;
740
741
742 --
743 -- TOC entry 218 (class 1259 OID 235838)
744 -- Name: student; Type: TABLE; Schema: public; Owner: postgres
745 --
746
747 CREATE TABLE public.student (
748 id integer NOT NULL,
749 user_id integer NOT NULL,
750 city_id integer NOT NULL
751 );
752
753
754 ALTER TABLE public.student OWNER TO postgres;
755
756 --
757 -- TOC entry 228 (class 1259 OID 342494)
758 -- Name: student_group_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
759 --
760
761 CREATE SEQUENCE public.student_group_id_seq
762 START WITH 1
763 INCREMENT BY 1
764 NO MINVALUE
765 NO MAXVALUE
766 CACHE 1;
767
768
769 ALTER TABLE public.student_group_id_seq OWNER TO postgres;
770
771 --
772 -- TOC entry 232 (class 1259 OID 350723)
773 -- Name: student_group; Type: TABLE; Schema: public; Owner: postgres
774 --
775
776 CREATE TABLE public.student_group (
777 id integer DEFAULT nextval('public.student_group_id_seq'::regclass) NOT NULL,
778 group_id integer NOT NULL,
779 student_id integer NOT NULL
780 );
781
782
783 ALTER TABLE public.student_group OWNER TO postgres;
784
785 --
786 -- TOC entry 217 (class 1259 OID 235836)
787 -- Name: student_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
788 --
789
790 CREATE SEQUENCE public.student_id_seq
791 START WITH 1
792 INCREMENT BY 1

```

```

793 NO MINVALUE
794 NO MAXVALUE
795 CACHE 1;
796
797
798 ALTER TABLE public.student_id_seq OWNER TO postgres;
799
800 --
801 -- TOC entry 3420 (class 0 OID 0)
802 -- Dependencies: 217
803 -- Name: student_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
     postgres
804 --
805
806 ALTER SEQUENCE public.student_id_seq OWNED BY public.student.id;
807
808
809 --
810 -- TOC entry 229 (class 1259 OID 342513)
811 -- Name: student_problemlist_id_seq; Type: SEQUENCE; Schema: public; Owner:
     postgres
812 --
813
814 CREATE SEQUENCE public.student_problemlist_id_seq
815 START WITH 1
816 INCREMENT BY 1
817 NO MINVALUE
818 NO MAXVALUE
819 CACHE 1;
820
821
822 ALTER TABLE public.student_problemlist_id_seq OWNER TO postgres;
823
824 --
825 -- TOC entry 224 (class 1259 OID 342434)
826 -- Name: student_problemlist; Type: TABLE; Schema: public; Owner: postgres
827 --
828
829 CREATE TABLE public.student_problemlist (
830 id integer DEFAULT nextval('public.student_problemlist_id_seq'::regclass) NOT
     NULL,
831 problemlist_id integer NOT NULL,
832 student_id integer NOT NULL
833 );
834
835
836 ALTER TABLE public.student_problemlist OWNER TO postgres;
837
838 --
839 -- TOC entry 202 (class 1259 OID 235744)
840 -- Name: teacher; Type: TABLE; Schema: public; Owner: postgres
841 --
842
843 CREATE TABLE public.teacher (
844 id integer NOT NULL,
845 user_id integer NOT NULL,
846 institution_id integer NOT NULL
847 );
848

```

```
849
850 ALTER TABLE public.teacher OWNER TO postgres;
851
852 --
853 -- TOC entry 201 (class 1259 OID 235742)
854 -- Name: teacher_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
855 --
856
857 CREATE SEQUENCE public.teacher_id_seq
858 START WITH 1
859 INCREMENT BY 1
860 NO MINVALUE
861 NO MAXVALUE
862 CACHE 1;
863
864
865 ALTER TABLE public.teacher_id_seq OWNER TO postgres;
866
867 --
868 -- TOC entry 3421 (class 0 OID 0)
869 -- Dependencies: 201
870 -- Name: teacher_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
     postgres
871 --
872
873 ALTER SEQUENCE public.teacher_id_seq OWNED BY public.teacher.id;
874
875
876 --
877 -- TOC entry 210 (class 1259 OID 235786)
878 -- Name: testdata; Type: TABLE; Schema: public; Owner: postgres
879 --
880
881 CREATE TABLE publictestdata (
882 id integer NOT NULL,
883 input character varying(254),
884 output character varying(254) NOT NULL,
885 problem_id integer NOT NULL
886 );
887
888
889 ALTER TABLE publictestdata OWNER TO postgres;
890
891 --
892 -- TOC entry 209 (class 1259 OID 235784)
893 -- Name: testdata_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
894 --
895
896 CREATE SEQUENCE publictestdata_id_seq
897 START WITH 1
898 INCREMENT BY 1
899 NO MINVALUE
900 NO MAXVALUE
901 CACHE 1;
902
903
904 ALTER TABLE publictestdata_id_seq OWNER TO postgres;
905
906 --
```

```
907 -- TOC entry 3422 (class 0 OID 0)
908 -- Dependencies: 209
909 -- Name: testdata_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
      postgres
910 --
911
912 ALTER SEQUENCE publictestdata_id_seq OWNED BY publictestdata.id;
913
914
915 --
916 -- TOC entry 3140 (class 2604 OID 235735)
917 -- Name: admins id; Type: DEFAULT; Schema: public; Owner: postgres
918 --
919
920 ALTER TABLE ONLY public.admins ALTER COLUMN id SET DEFAULT nextval('public.
      admins_id_seq'::regclass);
921
922
923 --
924 -- TOC entry 3135 (class 2604 OID 235662)
925 -- Name: city id; Type: DEFAULT; Schema: public; Owner: postgres
926 --
927
928 ALTER TABLE ONLY public.city ALTER COLUMN id SET DEFAULT nextval('public.
      city_id_seq'::regclass);
929
930
931 --
932 -- TOC entry 3133 (class 2604 OID 235627)
933 -- Name: country id; Type: DEFAULT; Schema: public; Owner: postgres
934 --
935
936 ALTER TABLE ONLY public.country ALTER COLUMN id SET DEFAULT nextval('public.
      country_id_seq'::regclass);
937
938
939 --
940 -- TOC entry 3139 (class 2604 OID 235711)
941 -- Name: groupadmin id; Type: DEFAULT; Schema: public; Owner: postgres
942 --
943
944 ALTER TABLE ONLY public.groupadmin ALTER COLUMN id SET DEFAULT nextval('public.
      groupadmin_id_seq'::regclass);
945
946
947 --
948 -- TOC entry 3136 (class 2604 OID 235674)
949 -- Name: institution id; Type: DEFAULT; Schema: public; Owner: postgres
950 --
951
952 ALTER TABLE ONLY public.institution ALTER COLUMN id SET DEFAULT nextval('public.
      institution_id_seq'::regclass);
953
954
955 --
956 -- TOC entry 3142 (class 2604 OID 235759)
957 -- Name: keyword id; Type: DEFAULT; Schema: public; Owner: postgres
958 --
959
```

```
960 ALTER TABLE ONLY public.keyword ALTER COLUMN id SET DEFAULT nextval('public.
    keyword_id_seq'::regclass);
961
962
963 --
964 -- TOC entry 3138 (class 2604 OID 235700)
965 -- Name: permission id; Type: DEFAULT; Schema: public; Owner: postgres
966 --
967
968 ALTER TABLE ONLY public.permission ALTER COLUMN id SET DEFAULT nextval('public.
    permission_id_seq'::regclass);
969
970
971 --
972 -- TOC entry 3148 (class 2604 OID 350698)
973 -- Name: portalgroup id; Type: DEFAULT; Schema: public; Owner: postgres
974 --
975
976 ALTER TABLE ONLY public.portalgroup ALTER COLUMN id SET DEFAULT nextval('public.
    portalgroup_id_seq'::regclass);
977
978
979 --
980 -- TOC entry 3137 (class 2604 OID 235686)
981 -- Name: portaluser id; Type: DEFAULT; Schema: public; Owner: postgres
982 --
983
984 ALTER TABLE ONLY public.portaluser ALTER COLUMN id SET DEFAULT nextval('public.
    portaluser_id_seq'::regclass);
985
986
987 --
988 -- TOC entry 3144 (class 2604 OID 235777)
989 -- Name: problem id; Type: DEFAULT; Schema: public; Owner: postgres
990 --
991
992 ALTER TABLE ONLY public.problem ALTER COLUMN id SET DEFAULT nextval('public.
    problem_id_seq'::regclass);
993
994
995 --
996 -- TOC entry 3150 (class 2604 OID 235853)
997 -- Name: problemkeyword id; Type: DEFAULT; Schema: public; Owner: postgres
998 --
999
1000 ALTER TABLE ONLY public.problemkeyword ALTER COLUMN id SET DEFAULT nextval('
    public.problemkeyword_id_seq'::regclass);
1001
1002
1003 --
1004 -- TOC entry 3147 (class 2604 OID 235817)
1005 -- Name: problemlist id; Type: DEFAULT; Schema: public; Owner: postgres
1006 --
1007
1008 ALTER TABLE ONLY public.problemlist ALTER COLUMN id SET DEFAULT nextval('public.
    problemlist_id_seq'::regclass);
1009
1010
1011 --
```

```

1012 -- TOC entry 3146 (class 2604 OID 235802)
1013 -- Name: problemsolution id; Type: DEFAULT; Schema: public; Owner: postgres
1014 --
1015
1016 ALTER TABLE ONLY public.problemsolution ALTER COLUMN id SET DEFAULT nextval(
    public.problemsolution_id_seq'::regclass);
1017
1018
1019 --
1020 -- TOC entry 3143 (class 2604 OID 235768)
1021 -- Name: problemtyp id; Type: DEFAULT; Schema: public; Owner: postgres
1022 --
1023
1024 ALTER TABLE ONLY public.problemtyp ALTER COLUMN id SET DEFAULT nextval('public.
    problemtyp_id_seq'::regclass);
1025
1026
1027 --
1028 -- TOC entry 3134 (class 2604 OID 235638)
1029 -- Name: state id; Type: DEFAULT; Schema: public; Owner: postgres
1030 --
1031
1032 ALTER TABLE ONLY public.state ALTER COLUMN id SET DEFAULT nextval('public.
    state_id_seq'::regclass);
1033
1034
1035 --
1036 -- TOC entry 3149 (class 2604 OID 235841)
1037 -- Name: student id; Type: DEFAULT; Schema: public; Owner: postgres
1038 --
1039
1040 ALTER TABLE ONLY public.student ALTER COLUMN id SET DEFAULT nextval('public.
    student_id_seq'::regclass);
1041
1042
1043 --
1044 -- TOC entry 3141 (class 2604 OID 235747)
1045 -- Name: teacher id; Type: DEFAULT; Schema: public; Owner: postgres
1046 --
1047
1048 ALTER TABLE ONLY public.teacher ALTER COLUMN id SET DEFAULT nextval('public.
    teacher_id_seq'::regclass);
1049
1050
1051 --
1052 -- TOC entry 3145 (class 2604 OID 235789)
1053 -- Name: testdata id; Type: DEFAULT; Schema: public; Owner: postgres
1054 --
1055
1056 ALTER TABLE ONLY publictestdata ALTER COLUMN id SET DEFAULT nextval('public.
    testdata_id_seq'::regclass);
1057
1058
1059 --
1060 -- TOC entry 3185 (class 2606 OID 235740)
1061 -- Name: admins admins_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
1062 --
1063
1064 ALTER TABLE ONLY public.admins

```

```

1065 ADD CONSTRAINT admins_pkey PRIMARY KEY (id);
1066
1067
1068 --
1069 -- TOC entry 3167 (class 2606 OID 235667)
1070 -- Name: city city_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
1071 --
1072
1073 ALTER TABLE ONLY public.city
1074 ADD CONSTRAINT city_pkey PRIMARY KEY (id);
1075
1076
1077 --
1078 -- TOC entry 3161 (class 2606 OID 235629)
1079 -- Name: country country_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
1080 --
1081
1082 ALTER TABLE ONLY public.country
1083 ADD CONSTRAINT country_pkey PRIMARY KEY (id);
1084
1085
1086 --
1087 -- TOC entry 3229 (class 2606 OID 342482)
1088 -- Name: group_problemlist group_problemlist_pkey; Type: CONSTRAINT; Schema:
     public; Owner: postgres
1089 --
1090
1091 ALTER TABLE ONLY public.group_problemlist
1092 ADD CONSTRAINT group_problemlist_pkey PRIMARY KEY (id);
1093
1094
1095 --
1096 -- TOC entry 3240 (class 2606 OID 342463)
1097 -- Name: groupadmin_permission groupadmin_permission_pkey; Type: CONSTRAINT;
     Schema: public; Owner: postgres
1098 --
1099
1100 ALTER TABLE ONLY public.groupadmin_permission
1101 ADD CONSTRAINT groupadmin_permission_pkey PRIMARY KEY (id);
1102
1103
1104 --
1105 -- TOC entry 3182 (class 2606 OID 235716)
1106 -- Name: groupadmin groupadmin_pkey; Type: CONSTRAINT; Schema: public; Owner:
     postgres
1107 --
1108
1109 ALTER TABLE ONLY public.groupadmin
1110 ADD CONSTRAINT groupadmin_pkey PRIMARY KEY (id);
1111
1112
1113 --
1114 -- TOC entry 3170 (class 2606 OID 235679)
1115 -- Name: institution institution_pkey; Type: CONSTRAINT; Schema: public; Owner:
     postgres
1116 --
1117
1118 ALTER TABLE ONLY public.institution
1119 ADD CONSTRAINT institution_pkey PRIMARY KEY (id);

```

```

1120
1121
1122 --
1123 -- TOC entry 3193 (class 2606 OID 235761)
1124 -- Name: keyword_keyword_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
1125 --
1126
1127 ALTER TABLE ONLY public.keyword
1128 ADD CONSTRAINT keyword_pkey PRIMARY KEY (id);
1129
1130
1131 --
1132 -- TOC entry 3227 (class 2606 OID 341022)
1133 -- Name: listactivity_list_pkey; Type: CONSTRAINT; Schema: public; Owner:
      postgres
1134 --
1135
1136 ALTER TABLE ONLY public.listactivity
1137 ADD CONSTRAINT list_pkey PRIMARY KEY (id);
1138
1139
1140 --
1141 -- TOC entry 3179 (class 2606 OID 235702)
1142 -- Name: permission_permission_pkey; Type: CONSTRAINT; Schema: public; Owner:
      postgres
1143 --
1144
1145 ALTER TABLE ONLY public.permission
1146 ADD CONSTRAINT permission_pkey PRIMARY KEY (id);
1147
1148
1149 --
1150 -- TOC entry 3216 (class 2606 OID 235834)
1151 -- Name: portalgroup_portalgroup_pkey; Type: CONSTRAINT; Schema: public; Owner:
      postgres
1152 --
1153
1154 ALTER TABLE ONLY public.portalgroup
1155 ADD CONSTRAINT portalgroup_pkey PRIMARY KEY (id);
1156
1157
1158 --
1159 -- TOC entry 3245 (class 2606 OID 350661)
1160 -- Name: portaluser_permission_portaluser_permission_pkey; Type: CONSTRAINT;
      Schema: public; Owner: postgres
1161 --
1162
1163 ALTER TABLE ONLY public.portaluser_permission
1164 ADD CONSTRAINT portaluser_permission_pkey PRIMARY KEY (id);
1165
1166
1167 --
1168 -- TOC entry 3174 (class 2606 OID 235691)
1169 -- Name: portaluser_portaluser_pkey; Type: CONSTRAINT; Schema: public; Owner:
      postgres
1170 --
1171
1172 ALTER TABLE ONLY public.portaluser
1173 ADD CONSTRAINT portaluser_pkey PRIMARY KEY (id);

```

```

1174
1175
1176 --
1177 -- TOC entry 3202 (class 2606 OID 235782)
1178 -- Name: problem_problem_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
1179 --
1180
1181 ALTER TABLE ONLY public.problem
1182 ADD CONSTRAINT problem_pkey PRIMARY KEY (id);
1183
1184 --
1185 --
1186 -- TOC entry 3222 (class 2606 OID 235858)
1187 -- Name: problemkeyword_problemkeyword_pkey; Type: CONSTRAINT; Schema: public;
1188 --           Owner: postgres
1189 --
1190 ALTER TABLE ONLY public.problemkeyword
1191 ADD CONSTRAINT problemkeyword_pkey PRIMARY KEY (id);
1192
1193
1194 --
1195 -- TOC entry 3211 (class 2606 OID 235822)
1196 -- Name: problemlist_problemlist_pkey; Type: CONSTRAINT; Schema: public; Owner:
1197 --           postgres
1198 --
1199 ALTER TABLE ONLY public.problemlist
1200 ADD CONSTRAINT problemlist_pkey PRIMARY KEY (id);
1201
1202
1203 --
1204 -- TOC entry 3208 (class 2606 OID 235807)
1205 -- Name: problemsolution_problemsolution_pkey; Type: CONSTRAINT; Schema: public;
1206 --           Owner: postgres
1207 --
1208 ALTER TABLE ONLY public.problemsolution
1209 ADD CONSTRAINT problemsolution_pkey PRIMARY KEY (id);
1210
1211
1212 --
1213 -- TOC entry 3196 (class 2606 OID 340995)
1214 -- Name: problemtypename_uindex; Type: CONSTRAINT; Schema: public;
1215 --           Owner: postgres
1216 --
1217 ALTER TABLE ONLY public.problemtype
1218 ADD CONSTRAINT problemtypename_uindex UNIQUE (name);
1219
1220
1221 --
1222 -- TOC entry 3198 (class 2606 OID 235770)
1223 -- Name: problemtypename_pkey; Type: CONSTRAINT; Schema: public; Owner:
1224 --           postgres
1225
1226 ALTER TABLE ONLY public.problemtype
1227 ADD CONSTRAINT problemtypename_pkey PRIMARY KEY (id);

```

```

1228
1229
1230 --
1231 -- TOC entry 3164 (class 2606 OID 235643)
1232 -- Name: state state_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
1233 --
1234
1235 ALTER TABLE ONLY public.state
1236 ADD CONSTRAINT state_pkey PRIMARY KEY (id);
1237
1238
1239 --
1240 -- TOC entry 3249 (class 2606 OID 350728)
1241 -- Name: student_group student_group_pkey; Type: CONSTRAINT; Schema: public;
      Owner: postgres
1242 --
1243
1244 ALTER TABLE ONLY public.student_group
1245 ADD CONSTRAINT student_group_pkey PRIMARY KEY (id);
1246
1247
1248 --
1249 -- TOC entry 3219 (class 2606 OID 235846)
1250 -- Name: student student_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
1251 --
1252
1253 ALTER TABLE ONLY public.student
1254 ADD CONSTRAINT student_pkey PRIMARY KEY (id);
1255
1256
1257 --
1258 -- TOC entry 3234 (class 2606 OID 342520)
1259 -- Name: student_problemlist student_problemlist_pkey; Type: CONSTRAINT; Schema:
      public; Owner: postgres
1260 --
1261
1262 ALTER TABLE ONLY public.student_problemlist
1263 ADD CONSTRAINT student_problemlist_pkey PRIMARY KEY (id);
1264
1265
1266 --
1267 -- TOC entry 3190 (class 2606 OID 235752)
1268 -- Name: teacher teacher_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
1269 --
1270
1271 ALTER TABLE ONLY public.teacher
1272 ADD CONSTRAINT teacher_pkey PRIMARY KEY (id);
1273
1274
1275 --
1276 -- TOC entry 3205 (class 2606 OID 235794)
1277 -- Name: testdata testdata_pkey; Type: CONSTRAINT; Schema: public; Owner:
      postgres
1278 --
1279
1280 ALTER TABLE ONLY publictestdata
1281 ADD CONSTRAINT testdata_pkey PRIMARY KEY (id);
1282
1283

```

```

1284 --
1285 -- TOC entry 3251 (class 2606 OID 407796)
1286 -- Name: student_group_unique_together_group_id_with_student_id; Type:
           CONSTRAINT; Schema: public; Owner: postgres
1287 --
1288
1289 ALTER TABLE ONLY public.student_group
1290 ADD CONSTRAINT unique_together_group_id_with_student_id UNIQUE (group_id,
           student_id);
1291
1292
1293 --
1294 -- TOC entry 3242 (class 2606 OID 407792)
1295 -- Name: groupadmin_permission_unique_together_groupadmin_id_with_permission_id;
           Type: CONSTRAINT; Schema: public; Owner: postgres
1296 --
1297
1298 ALTER TABLE ONLY public.groupadmin_permission
1299 ADD CONSTRAINT unique_together_groupadmin_id_with_permission_id UNIQUE (
           groupadmin_id, permission_id);
1300
1301
1302 --
1303 -- TOC entry 3213 (class 2606 OID 341044)
1304 -- Name: problemlist_unique_together_listactivity_id_with_problem_id; Type:
           CONSTRAINT; Schema: public; Owner: postgres
1305 --
1306
1307 ALTER TABLE ONLY public.problemlist
1308 ADD CONSTRAINT unique_together_listactivity_id_with_problem_id UNIQUE (
           problem_id, listactivity_id);
1309
1310
1311 --
1312 -- TOC entry 3247 (class 2606 OID 407794)
1313 -- Name: portaluser_permission_unique_together_portaluser_id_with_permission_id;
           Type: CONSTRAINT; Schema: public; Owner: postgres
1314 --
1315
1316 ALTER TABLE ONLY public.portaluser_permission
1317 ADD CONSTRAINT unique_together_portaluser_id_with_permission_id UNIQUE (
           portaluser_id, permission_id);
1318
1319
1320 --
1321 -- TOC entry 3224 (class 2606 OID 341007)
1322 -- Name: problemkeyword_unique_together_problem_id_with_keyword_id; Type:
           CONSTRAINT; Schema: public; Owner: postgres
1323 --
1324
1325 ALTER TABLE ONLY public.problemkeyword
1326 ADD CONSTRAINT unique_together_problem_id_with_keyword_id UNIQUE (keyword_id,
           problem_id);
1327
1328
1329 --
1330 -- TOC entry 3232 (class 2606 OID 407790)
1331 -- Name: group_problemlist_unique_together_problemlist_id_with_group_id; Type:
           CONSTRAINT; Schema: public; Owner: postgres

```

```

1332 --
1333
1334 ALTER TABLE ONLY public.group_problemlist
1335 ADD CONSTRAINT unique_together_problemlist_id_with_group_id UNIQUE (group_id,
    problemlist_id);
1336
1337
1338 --
1339 -- TOC entry 3237 (class 2606 OID 407798)
1340 -- Name: student_problemlist_unique_together_problemlist_id_with_student_id;
    Type: CONSTRAINT; Schema: public; Owner: postgres
1341 --
1342
1343 ALTER TABLE ONLY public.student_problemlist
1344 ADD CONSTRAINT unique_together_problemlist_id_with_student_id UNIQUE (
    problemlist_id, student_id);
1345
1346
1347 --
1348 -- TOC entry 3187 (class 2606 OID 407788)
1349 -- Name: admins_unique_together_user_id_with_groupadmin_id; Type: CONSTRAINT;
    Schema: public; Owner: postgres
1350 --
1351
1352 ALTER TABLE ONLY public.admins
1353 ADD CONSTRAINT unique_together_user_id_with_groupadmin_id UNIQUE (user_id,
    groupadmin_id);
1354
1355
1356 --
1357 -- TOC entry 3183 (class 1259 OID 235741)
1358 -- Name: admins_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1359 --
1360
1361 CREATE UNIQUE INDEX admins_id_uindex ON public.admins USING btree (id);
1362
1363
1364 --
1365 -- TOC entry 3165 (class 1259 OID 235668)
1366 -- Name: city_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1367 --
1368
1369 CREATE UNIQUE INDEX city_id_uindex ON public.city USING btree (id);
1370
1371
1372 --
1373 -- TOC entry 3157 (class 1259 OID 235632)
1374 -- Name: country_abbreviation_uindex; Type: INDEX; Schema: public; Owner:
    postgres
1375 --
1376
1377 CREATE UNIQUE INDEX country_abbreviation_uindex ON public.country USING btree (
    abbreviation);
1378
1379
1380 --
1381 -- TOC entry 3158 (class 1259 OID 235630)
1382 -- Name: country_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1383 --

```

```
1384
1385 CREATE UNIQUE INDEX country_id_uindex ON public.country USING btree (id);
1386
1387
1388 --
1389 -- TOC entry 3159 (class 1259 OID 235631)
1390 -- Name: country_name_uindex; Type: INDEX; Schema: public; Owner: postgres
1391 --
1392
1393 CREATE UNIQUE INDEX country_name_uindex ON public.country USING btree (name);
1394
1395
1396 --
1397 -- TOC entry 3199 (class 1259 OID 331882)
1398 -- Name: fki_problemtyp_id_fk; Type: INDEX; Schema: public; Owner: postgres
1399 --
1400
1401 CREATE INDEX fki_problemtyp_id_fk ON public.problem USING btree (problemtyp_id
   );
1402
1403
1404 --
1405 -- TOC entry 3230 (class 1259 OID 342493)
1406 -- Name: group_problemlist_uindex; Type: INDEX; Schema: public; Owner: postgres
1407 --
1408
1409 CREATE UNIQUE INDEX group_problemlist_uindex ON public.group_problemlist USING
   btree (id);
1410
1411
1412 --
1413 -- TOC entry 3180 (class 1259 OID 235717)
1414 -- Name: groupadmin_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1415 --
1416
1417 CREATE UNIQUE INDEX groupadmin_id_uindex ON public.groupadmin USING btree (id);
1418
1419
1420 --
1421 -- TOC entry 3238 (class 1259 OID 342474)
1422 -- Name: groupadmin_permission_id_uindex; Type: INDEX; Schema: public; Owner:
   postgres
1423 --
1424
1425 CREATE UNIQUE INDEX groupadmin_permission_id_uindex ON public.
   groupadmin_permission USING btree (id);
1426
1427
1428 --
1429 -- TOC entry 3168 (class 1259 OID 235680)
1430 -- Name: institution_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1431 --
1432
1433 CREATE UNIQUE INDEX institution_id_uindex ON public.institution USING btree (id
   );
1434
1435
1436 --
1437 -- TOC entry 3191 (class 1259 OID 235762)
```

```
1438 -- Name: keyword_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1439 --
1440
1441 CREATE UNIQUE INDEX keyword_id_uindex ON public.keyword USING btree (id);
1442
1443
1444 --
1445 -- TOC entry 3225 (class 1259 OID 341023)
1446 -- Name: list_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1447 --
1448
1449 CREATE INDEX list_id_uindex ON public.listactivity USING btree (id);
1450
1451
1452 --
1453 -- TOC entry 3176 (class 1259 OID 235704)
1454 -- Name: permission_codename_uindex; Type: INDEX; Schema: public; Owner:
      postgres
1455 --
1456
1457 CREATE UNIQUE INDEX permission_codename_uindex ON public.permission USING btree
      (codename);
1458
1459
1460 --
1461 -- TOC entry 3177 (class 1259 OID 235703)
1462 -- Name: permission_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1463 --
1464
1465 CREATE UNIQUE INDEX permission_id_uindex ON public.permission USING btree (id);
1466
1467
1468 --
1469 -- TOC entry 3214 (class 1259 OID 235835)
1470 -- Name: portalgroup_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1471 --
1472
1473 CREATE UNIQUE INDEX portalgroup_id_uindex ON public.portalgroup USING btree (id)
      ;
1474
1475
1476 --
1477 -- TOC entry 3171 (class 1259 OID 235694)
1478 -- Name: portaluser_email_uindex; Type: INDEX; Schema: public; Owner: postgres
1479 --
1480
1481 CREATE UNIQUE INDEX portaluser_email_uindex ON public.portaluser USING btree (
      email);
1482
1483
1484 --
1485 -- TOC entry 3172 (class 1259 OID 235692)
1486 -- Name: portaluser_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1487 --
1488
1489 CREATE UNIQUE INDEX portaluser_id_uindex ON public.portaluser USING btree (id);
1490
1491
1492 --
```

```

1493 -- TOC entry 3243 (class 1259 OID 350672)
1494 -- Name: portaluser_permission_id_uindex; Type: INDEX; Schema: public; Owner:
      postgres
1495 --
1496
1497 CREATE INDEX portaluser_permission_id_uindex ON public.portaluser_permission
      USING btree (id);
1498
1499
1500 --
1501 -- TOC entry 3175 (class 1259 OID 235693)
1502 -- Name: portaluser_username_uindex; Type: INDEX; Schema: public; Owner:
      postgres
1503 --
1504
1505 CREATE UNIQUE INDEX portaluser_username_uindex ON public.portaluser USING btree
      (username);
1506
1507
1508 --
1509 -- TOC entry 3200 (class 1259 OID 235783)
1510 -- Name: problem_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1511 --
1512
1513 CREATE UNIQUE INDEX problem_id_uindex ON public.problem USING btree (id);
1514
1515
1516 --
1517 -- TOC entry 3220 (class 1259 OID 235859)
1518 -- Name: problemkeyword_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1519 --
1520
1521 CREATE UNIQUE INDEX problemkeyword_id_uindex ON public.problemkeyword USING
      btree (id);
1522
1523
1524 --
1525 -- TOC entry 3209 (class 1259 OID 235823)
1526 -- Name: problemlist_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1527 --
1528
1529 CREATE UNIQUE INDEX problemlist_id_uindex ON public.problemlist USING btree (id)
      ;
1530
1531
1532 --
1533 -- TOC entry 3206 (class 1259 OID 235808)
1534 -- Name: problemsolution_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1535 --
1536
1537 CREATE UNIQUE INDEX problemsolution_id_uindex ON public.problemsolution USING
      btree (id);
1538
1539
1540 --
1541 -- TOC entry 3194 (class 1259 OID 235771)
1542 -- Name: problemtyp_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1543 --
1544

```

```

1545 CREATE UNIQUE INDEX problemtyp_id_uindex ON public.problemtyp USING btree (id)
      ;
1546
1547
1548 --
1549 -- TOC entry 3162 (class 1259 OID 235644)
1550 -- Name: state_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1551 --
1552
1553 CREATE UNIQUE INDEX state_id_uindex ON public.state USING btree (id);
1554
1555
1556 --
1557 -- TOC entry 3217 (class 1259 OID 235847)
1558 -- Name: student_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1559 --
1560
1561 CREATE UNIQUE INDEX student_id_uindex ON public.student USING btree (id);
1562
1563
1564 --
1565 -- TOC entry 3235 (class 1259 OID 342531)
1566 -- Name: student_problemlist_uindex; Type: INDEX; Schema: public; Owner:
      postgres
1567 --
1568
1569 CREATE INDEX student_problemlist_uindex ON public.student_problemlist USING
      btree (id);
1570
1571
1572 --
1573 -- TOC entry 3188 (class 1259 OID 235753)
1574 -- Name: teacher_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1575 --
1576
1577 CREATE UNIQUE INDEX teacher_id_uindex ON public.teacher USING btree (id);
1578
1579
1580 --
1581 -- TOC entry 3203 (class 1259 OID 235795)
1582 -- Name: testdata_id_uindex; Type: INDEX; Schema: public; Owner: postgres
1583 --
1584
1585 CREATE UNIQUE INDEX testdata_id_uindex ON public.testdata USING btree (id);
1586
1587
1588 --
1589 -- TOC entry 3254 (class 2606 OID 278023)
1590 -- Name: institution_city_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:
      postgres
1591 --
1592
1593 ALTER TABLE ONLY public.institution
1594 ADD CONSTRAINT city_id_fk FOREIGN KEY (city_id) REFERENCES public.city(id);
1595
1596
1597 --
1598 -- TOC entry 3267 (class 2606 OID 350678)
1599 -- Name: student_city_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:

```

```

      postgres
1600 --
1601
1602 ALTER TABLE ONLY public.student
1603 ADD CONSTRAINT city_id_fk FOREIGN KEY (city_id) REFERENCES public.city(id);
1604
1605
1606 --
1607 -- TOC entry 3252 (class 2606 OID 278013)
1608 -- Name: state_country_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:
      postgres
1609 --
1610
1611 ALTER TABLE ONLY public.state
1612 ADD CONSTRAINT country_id_fk FOREIGN KEY (country_id) REFERENCES public.country(
      id);
1613
1614
1615 --
1616 -- TOC entry 3270 (class 2606 OID 342483)
1617 -- Name: group_problemlist_group_id_fk; Type: FK CONSTRAINT; Schema: public;
      Owner: postgres
1618 --
1619
1620 ALTER TABLE ONLY public.group_problemlist
1621 ADD CONSTRAINT group_id_fk FOREIGN KEY (group_id) REFERENCES public.portalgroup(
      id);
1622
1623
1624 --
1625 -- TOC entry 3279 (class 2606 OID 350734)
1626 -- Name: student_group_group_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:
      postgres
1627 --
1628
1629 ALTER TABLE ONLY public.student_group
1630 ADD CONSTRAINT group_id_fk FOREIGN KEY (group_id) REFERENCES public.portalgroup(
      id);
1631
1632
1633 --
1634 -- TOC entry 3256 (class 2606 OID 342451)
1635 -- Name: admins_groupadmin_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:
      postgres
1636 --
1637
1638 ALTER TABLE ONLY public.admins
1639 ADD CONSTRAINT groupadmin_id_fk FOREIGN KEY (groupadmin_id) REFERENCES public.
      groupadmin(id);
1640
1641
1642 --
1643 -- TOC entry 3274 (class 2606 OID 342464)
1644 -- Name: groupadmin_permission_groupadmin_id_fk; Type: FK CONSTRAINT; Schema:
      public; Owner: postgres
1645 --
1646
1647 ALTER TABLE ONLY public.groupadmin_permission
1648 ADD CONSTRAINT groupadmin_id_fk FOREIGN KEY (groupadmin_id) REFERENCES public.

```

```

        groupadmin(id);

1649
1650
1651 --
1652 -- TOC entry 3258 (class 2606 OID 332807)
1653 -- Name: teacher_institution_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:
      postgres
1654 --
1655
1656 ALTER TABLE ONLY public.teacher
1657 ADD CONSTRAINT institution_id_fk FOREIGN KEY (institution_id) REFERENCES public.
      institution(id);
1658
1659
1660 --
1661 -- TOC entry 3269 (class 2606 OID 341001)
1662 -- Name: problemkeyword_keyword_id_fk; Type: FK CONSTRAINT; Schema: public;
      Owner: postgres
1663 --
1664
1665 ALTER TABLE ONLY public.problemkeyword
1666 ADD CONSTRAINT keyword_id_fk FOREIGN KEY (keyword_id) REFERENCES public.keyword(
      id);
1667
1668
1669 --
1670 -- TOC entry 3264 (class 2606 OID 341038)
1671 -- Name: problemlist_listactivity_id_pk; Type: FK CONSTRAINT; Schema: public;
      Owner: postgres
1672 --
1673
1674 ALTER TABLE ONLY public.problemlist
1675 ADD CONSTRAINT listactivity_id_pk FOREIGN KEY (listactivity_id) REFERENCES
      public.listactivity(id);
1676
1677
1678 --
1679 -- TOC entry 3275 (class 2606 OID 342469)
1680 -- Name: groupadmin_permission_permission_id_fk; Type: FK CONSTRAINT; Schema:
      public; Owner: postgres
1681 --
1682
1683 ALTER TABLE ONLY public.groupadmin_permission
1684 ADD CONSTRAINT permission_id_fk FOREIGN KEY (permission_id) REFERENCES public.
      permission(id);
1685
1686
1687 --
1688 -- TOC entry 3277 (class 2606 OID 350667)
1689 -- Name: portaluser_permission_permission_id_fk; Type: FK CONSTRAINT; Schema:
      public; Owner: postgres
1690 --
1691
1692 ALTER TABLE ONLY public.portaluser_permission
1693 ADD CONSTRAINT permission_id_fk FOREIGN KEY (permission_id) REFERENCES public.
      permission(id);
1694
1695
1696 --

```

```

1697 -- TOC entry 3276 (class 2606 OID 350662)
1698 -- Name: portaluser_permission_portaluser_permission_id_fk; Type: FK CONSTRAINT;
      Schema: public; Owner: postgres
1699 --
1700
1701 ALTER TABLE ONLY public.portaluser_permission
1702 ADD CONSTRAINT portaluser_permission_id_fk FOREIGN KEY (portaluser_id)
      REFERENCES public.portaluser(id);
1703
1704
1705 --
1706 -- TOC entry 3268 (class 2606 OID 340996)
1707 -- Name: problemkeyword_problem_id_fk; Type: FK CONSTRAINT; Schema: public;
      Owner: postgres
1708 --
1709
1710 ALTER TABLE ONLY public.problemkeyword
1711 ADD CONSTRAINT problem_id_fk FOREIGN KEY (problem_id) REFERENCES public.problem(
      id);
1712
1713
1714 --
1715 -- TOC entry 3260 (class 2606 OID 341008)
1716 -- Name: testdata_problem_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:
      postgres
1717 --
1718
1719 ALTER TABLE ONLY publictestdata
1720 ADD CONSTRAINT problem_id_fk FOREIGN KEY (problem_id) REFERENCES public.problem(
      id);
1721
1722
1723 --
1724 -- TOC entry 3263 (class 2606 OID 341027)
1725 -- Name: problemlist_problem_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:
      postgres
1726 --
1727
1728 ALTER TABLE ONLY public.problemlist
1729 ADD CONSTRAINT problem_id_fk FOREIGN KEY (problem_id) REFERENCES public.problem(
      id);
1730
1731
1732 --
1733 -- TOC entry 3262 (class 2606 OID 350744)
1734 -- Name: problemsolution_problem_id_fk; Type: FK CONSTRAINT; Schema: public;
      Owner: postgres
1735 --
1736
1737 ALTER TABLE ONLY public.problemsolution
1738 ADD CONSTRAINT problem_id_fk FOREIGN KEY (problem_id) REFERENCES public.problem(
      id);
1739
1740
1741 --
1742 -- TOC entry 3273 (class 2606 OID 350683)
1743 -- Name: student_problemlist_problemlist_id_fk; Type: FK CONSTRAINT; Schema:
      public; Owner: postgres
1744 --

```

```

1745
1746 ALTER TABLE ONLY public.student_problemlist
1747 ADD CONSTRAINT problemlist_id_fk FOREIGN KEY (problemlist_id) REFERENCES public.
    listactivity(id);
1748
1749
1750 --
1751 -- TOC entry 3271 (class 2606 OID 350699)
1752 -- Name: group_problemlist problemlist_id_fk; Type: FK CONSTRAINT; Schema:
    public; Owner: postgres
1753 --
1754
1755 ALTER TABLE ONLY public.group_problemlist
1756 ADD CONSTRAINT problemlist_id_fk FOREIGN KEY (problemlist_id) REFERENCES public.
    listactivity(id);
1757
1758
1759 --
1760 -- TOC entry 3259 (class 2606 OID 331877)
1761 -- Name: problem problemtyp_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:
    postgres
1762 --
1763
1764 ALTER TABLE ONLY public.problem
1765 ADD CONSTRAINT problemtyp_id_fk FOREIGN KEY (problemtyp_id) REFERENCES public.
    problemtyp(id);
1766
1767
1768 --
1769 -- TOC entry 3253 (class 2606 OID 278018)
1770 -- Name: city state_id_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres
1771 --
1772
1773 ALTER TABLE ONLY public.city
1774 ADD CONSTRAINT state_id_fk FOREIGN KEY (state_id) REFERENCES public.state(id);
1775
1776
1777 --
1778 -- TOC entry 3272 (class 2606 OID 342521)
1779 -- Name: student_problemlist student_id_fk; Type: FK CONSTRAINT; Schema: public;
    Owner: postgres
1780 --
1781
1782 ALTER TABLE ONLY public.student_problemlist
1783 ADD CONSTRAINT student_id_fk FOREIGN KEY (student_id) REFERENCES public.student(
    id);
1784
1785
1786 --
1787 -- TOC entry 3278 (class 2606 OID 350729)
1788 -- Name: student_group student_id_fk; Type: FK CONSTRAINT; Schema: public; Owner
    : postgres
1789 --
1790
1791 ALTER TABLE ONLY public.student_group
1792 ADD CONSTRAINT student_id_fk FOREIGN KEY (student_id) REFERENCES public.student(
    id);
1793
1794

```

```

1795 --
1796 -- TOC entry 3265 (class 2606 OID 350693)
1797 -- Name: portalgroup_teacher_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:
      postgres
1798 --
1799
1800 ALTER TABLE ONLY public.portalgroup
1801 ADD CONSTRAINT teacher_id_fk FOREIGN KEY (teacher_id) REFERENCES public.teacher(
      id);
1802
1803
1804 --
1805 -- TOC entry 3257 (class 2606 OID 332802)
1806 -- Name: teacher_user_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:
      postgres
1807 --
1808
1809 ALTER TABLE ONLY public.teacher
1810 ADD CONSTRAINT user_id_fk FOREIGN KEY (user_id) REFERENCES public.portaluser(id)
      ;
1811
1812
1813 --
1814 -- TOC entry 3255 (class 2606 OID 342446)
1815 -- Name: admins_user_id_fk; Type: FK CONSTRAINT; Schema: public; Owner: postgres
1816 --
1817
1818 ALTER TABLE ONLY public.admins
1819 ADD CONSTRAINT user_id_fk FOREIGN KEY (user_id) REFERENCES public.portaluser(id)
      ;
1820
1821
1822 --
1823 -- TOC entry 3266 (class 2606 OID 350673)
1824 -- Name: student_user_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:
      postgres
1825 --
1826
1827 ALTER TABLE ONLY public.student
1828 ADD CONSTRAINT user_id_fk FOREIGN KEY (user_id) REFERENCES public.portaluser(id)
      ;
1829
1830
1831 --
1832 -- TOC entry 3261 (class 2606 OID 350739)
1833 -- Name: problemsolution_user_id_fk; Type: FK CONSTRAINT; Schema: public; Owner:
      postgres
1834 --
1835
1836 ALTER TABLE ONLY public.problemsolution
1837 ADD CONSTRAINT user_id_fk FOREIGN KEY (user_id) REFERENCES public.portaluser(id)
      ;
1838
1839
1840 -- Completed on 2018-06-12 19:41:20 -03
1841
1842 --
1843 -- PostgreSQL database dump complete
1844 --

```



## APÊNDICE D - MIGRAÇÃO BANCO DE DADOS

*Script* de migração de dados.

```

1 SELECT id, username, first_name, last_name, email, is_active, password FROM
    auth_user INTO OUTFILE 'auth_user.csv' FIELDS TERMINATED BY ',' enclosed ''
    LINES TERMINATED BY '\n';
2
3 SELECT id, username_id FROM portal_aluno INTO OUTFILE 'portal_aluno.csv' FIELDS
    TERMINATED BY ',', enclosed '' LINES TERMINATED BY '\n';
4
5 SELECT id, nome, descricao, dica, nivel, custo FROM portal_probemas INTO OUTFILE
    'portal_probemas.csv' FIELDS TERMINATED BY ',', enclosed '' LINES
    TERMINATED BY '\n';
6
7 SELECT id, codProblema_id, entrada, saida FROM portal_eprobemas INTO OUTFILE
    'portal_eprobemas.csv' FIELDS TERMINATED BY ',', enclosed '' LINES
    TERMINATED BY '\n';
8
9 SELECT id, nome FROM portal_pchave INTO OUTFILE 'portal_pchave.csv' FIELDS
    TERMINATED BY ',', enclosed '' LINES TERMINATED BY '\n';
10
11 SELECT id, nome_id, codProblema_id FROM portal_pchaveprob INTO OUTFILE
    'portal_pchaveprob.csv' FIELDS TERMINATED BY ',', enclosed '' LINES
    TERMINATED BY '\n';
12
13 SELECT id, codProblema_id, nomeSolucao, custo, username_id, data_solucao,
    resultado FROM portal_sprobemas INTO OUTFILE 'portal_sprobemas.csv' FIELDS
    TERMINATED BY ',', enclosed '' LINES terminated by '\n';

```

```

1 import psycopg2
2 connection = psycopg2.connect(dbname='portalg', user='postgres', password='',
    host='localhost')
3 cursor = connection.cursor()
4
5 file_auth_user = open('auth_user.csv', 'r')
6 for row in file_auth_user.readlines():
7     row = row.split(';')
8
9     sql = "INSERT INTO portaluser (id, username, firstname, lastname, email,
    isactive, password) "
10    sql += "VALUES ({}, '{}', '{}', '{}', '{}', {}, '{}')".format(row[0], row[1],
        row[2], row[3], row[4], row[5], row[6])
11    cursor.execute(sql)
12    connection.commit()
13

```

```

14 cursor.execute("SELECT setval('portaluser_id_seq', (SELECT MAX(id) FROM
15     portaluser)+1);")
16 connection.commit()
17 cursor.execute("INSERT INTO country (id, name, abbreviation) VALUES (1, 'Brasil
18     ', 'BR')")
19 cursor.execute("INSERT INTO state (id, name, abbreviation, country_id) VALUES
19     (1, 'Rio Grande do Sul', 'RS', 1)")
20 cursor.execute("INSERT INTO city (id, name, abbreviation, state_id) VALUES (1,
21     'Caxias do Sul', 'RS', 1)")
22 connection.commit()
23
24 cursor.execute("SELECT setval('country_id_seq', (SELECT MAX(id) FROM country)+1)
25     ;")
26 cursor.execute("SELECT setval('state_id_seq', (SELECT MAX(id) FROM state)+1);")
27 cursor.execute("SELECT setval('city_id_seq', (SELECT MAX(id) FROM city)+1);")
28 connection.commit()
29
30
31 file_portal_aluno = open('portal_aluno.csv', 'r')
32 for row in file_portal_aluno.readlines():
33     row = row.split(';')
34
35     sql = "INSERT INTO student (id, user_id, city_id) VALUES ({}, {}, {})".format(
36         row[0], row[1], 1)
37     cursor.execute(sql)
38     connection.commit()
39
40 cursor.execute("SELECT setval('portaluser_id_seq', (SELECT MAX(id) FROM
41     portaluser)+1);")
42 connection.commit()
43
44
45 file_portal_probemas = open('portal_probemas.csv', 'r')
46 for row in file_portal_probemas.readlines():
47     row = row.split(';')
48
49     sql = "INSERT INTO problem (id, name, description, tip, level, cost,
50         problemtyp_id) "
51     sql += "VALUES ({}, {}, {})".format(row[0], row[1], row[2], row[3], row[4],
52         row[5], 1)
53     cursor.execute(sql)
54     connection.commit()
55
56 cursor.execute("SELECT setval('problem_id_seq', (SELECT MAX(id) FROM problem)+1)
57     ;")
58 connection.commit()
59
60 file_portal_eprobemas = open('portal_eprobemas.csv', 'r')
61 for row in file_portal_eprobemas.readlines():
62     row = row.split(';')

```

```

62
63 sql = "INSERT INTO testdata (id, problemtyp_id, input, output) "
64 sql += "VALUES ({} , {} , {} )".format(row[0] , row[1] , row[2] , row[3])
65 cursor.execute(sql)
66 connection.commit()
67
68 cursor.execute("SELECT setval('testdata_id_seq', (SELECT MAX(id) FROM testdata)
+1);")
69 connection.commit()
70
71
72 file_portal_pchave = open('portal_pchave.csv' , 'r')
73 for row in file_portal_pchave.readlines():
74 row = row.split(';')
75
76 sql = "INSERT INTO keyword (id, name) "
77 sql += "VALUES ({} , {} , {} )".format(row[0] , row[1])
78 cursor.execute(sql)
79 connection.commit()
80
81 cursor.execute("SELECT setval('keyword_id_seq', (SELECT MAX(id) FROM keyword)+1)
;")
82 connection.commit()
83
84
85 file_portal_pchaveprob = open('portal_pchaveprob.csv' , 'r')
86 for row in file_portal_pchaveprob.readlines():
87 row = row.split(';')
88
89 sql = "INSERT INTO problemkeyword (id, keyword_id, problem_id) "
90 sql += "VALUES ({} , {} , {} )".format(row[0] , row[1] , row[2])
91 cursor.execute(sql)
92 connection.commit()
93
94 cursor.execute("SELECT setval('problemkeyword_id_seq', (SELECT MAX(id) FROM
problemkeyword)+1);")
95 connection.commit()
96
97
98 file_portal_sprobemas = open('portal_sprobemas.csv' , 'r')
99 for row in file_portal_sprobemas.readlines():
100 row = row.split(';')
101
102 sql = "INSERT INTO problemsolution (id, problem_id, name, cost, user_id,
datesolution, result) "
103 sql += "VALUES ({} , {} , {} , {} , {} , {} , {} )".format(row[0] , row[1] , row[2] , row
[3] , row[4] , row[5] , row[6])
104 cursor.execute(sql)
105 connection.commit()
106
107 cursor.execute("SELECT setval('problemsolution_id_seq', (SELECT MAX(id) FROM
problemsolution)+1);")
108 connection.commit()
109
110
111 cursor.close()
112 connection.close()

```

## APÊNDICE E - PROJETOS

O desenvolvimento do Portal de Algoritmos foi dividido em dois códigos bases, *backend* e *frontend* e foi feito o versionamento de código utilizando o protocolo *git* e repositório de código *bitbucket*.

### Backend

O *Backend* foi desenvolvido em Java.

Para começar a desenvolver é preciso clonar o projeto do *bitbucket* e importar o mesmo na sua *IDE* (*Integrated Development Environment*) de preferência. As dependências do projeto são mantidas no arquivo pom.xml e gerenciadas pelo *maven*, então basta *buildar* o projeto com ele.

O servidor de aplicação utilizado foi o *Wildfly*, é preciso então baixar e configurar localmente ou no servidor de aplicação.

Link do projeto: <https://bitbucket.org/adrianomargarin/portalg>

### Frontend

O *Frontend* foi desenvolvido com *Angular 4*, *HTML* e *CSS*. As dependências do projeto são mantidas no arquivo package.json e para instalar é utilizado o comando a seguir.

```
1 npm install
```

Para executar o projeto é preciso levantar o servidor de aplicação com o comando a seguir.

```
1 ng serve
```

Link do projeto: <https://bitbucket.org/adrianomargarin/portalg-frontend>