

UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE CIÊNCIAS EXATAS E DA TECNOLOGIA
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, Junho 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 do Centro de Ciências Exatas e da Tecnologia 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

CCET/UCS

André Luis Martinotto

CCET/UCS

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

André Gustavo Adami
Coordenador

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

AUTOR DESCONHECIDO

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

LISTA DE ACRÔNIMOS	8
LISTA DE FIGURAS	9
LISTA DE TABELAS	12
RESUMO	13
1 INTRODUÇÃO	14
2 EVOLUÇÃO DE SOFTWARE	17
2.1 Reengenharia de Software	18
2.2 Engenharia Reversa	19
2.3 Engenharia de Software	20
2.4 Processo de Software	20
2.5 Engenharia de Requisitos	21
2.6 Casos de Uso	22
2.7 Metodologia ICONIX	23
2.7.1 Diagramas de Classe	23
2.7.2 Diagrama de Sequência	24
2.7.3 Diagrama de Robustez	25
2.7.4 Casos de Usos	25
2.8 Modelos de Domínio	26
2.9 Projeto de Arquitetura	26
2.10 Usabilidade	27
2.11 Tecnologias	27
2.11.1 Python e Django	27
2.11.2 Java EE	28
2.11.3 WildFly	28
2.11.4 DAO	28

2.11.5	REST	28
2.11.6	AngularJS	29
2.12	Ambiente Virtual de Aprendizagem	30
3	AVALIAÇÃO DO SOFTWARE DO PORTAL DE ALGORITMOS DA UCS	32
3.1	Diagrama de Classe de Domínio	32
3.2	Interfaces Gráficas	35
3.2.1	Cadastro de Aluno	35
3.2.2	Criação de Solução de Problemas	35
3.2.3	Gerenciamento de Alunos	37
3.2.4	Gerenciamento de Problemas	38
3.2.5	Edição de Palavra-Chave	39
3.2.6	Solução de Problemas	40
4	PROPOSTA DE SOLUÇÃO	42
4.1	Diagrama de Classe de Domínio	43
4.2	Requisitos de projeto	45
4.2.1	Requisitos Funcionais	45
4.2.2	Requisitos Não-Funcionais	46
4.3	Casos de Uso	46
4.3.1	Descrição dos Casos de Uso	48
4.4	Interfaces Gráficas	68
4.4.1	Cadastro de Aluno	69
4.4.2	Autenticação	70
4.4.3	Boas Vindas ao Administrador e Professor	71
4.4.4	Problemas	72
4.4.5	Tipo de Problema	73
4.4.6	Cadastro e Edição de Problema	74
4.4.7	Palavras-chave	75
4.4.8	Dados de Testes	76
4.4.9	Visualização de Solução de Problema	77
4.4.10	Lista de Grupos	78
4.4.11	Cadastro e Edição de Grupo	79
4.4.12	Adicionar Participantes	80
4.4.13	Adicionar Lista de Problemas	81
4.4.14	Lista de Alunos	82
4.4.15	Cadastro de Alunos	83
4.4.16	Listas de Problemas	84

4.4.17	Cadastro e Edição de Lista de Problemas	85
4.4.18	Lista de Usuários	86
4.4.19	Cadastro e Edição de Usuários	87
4.4.20	Lista de Professores	88
4.4.21	Cadastro e Edição de Professores	89
4.4.22	Lista de Instituições	90
4.4.23	Cadastro e Edição de Instituições	91
4.4.24	Lista de Permissões	92
4.4.25	Cadastro e Edição de Permissões	93
4.4.26	Lista de Grupos de Administradores	94
4.4.27	Cadastro e Edição de Grupos de Administradores	95
4.4.28	Lista de Países, Estados e Cidades	96
4.4.29	Cadastro e Edição de País	97
4.4.30	Cadastro e Edição de Estado	98
4.4.31	Cadastro e Edição de Cidade	99
4.5	Diagramas de Robustez	100
4.5.1	Cadastro de Aluno	100
4.5.2	Autenticação	101
4.5.3	Boas Vindas do Administrador e do Professor	102
4.5.4	Problemas	102
4.5.5	Tipo de Problema	103
4.5.6	Cadastro e Edição de Problema	103
4.5.7	Palavras-chave	104
4.5.8	Dados de Testes	104
4.5.9	Visualização de Solução partindo da listagem de Problemas	105
4.5.10	Lista de Grupos	105
4.5.11	Cadastro e Edição de Grupo	106
4.5.12	Adicionar Participantes	106
4.5.13	Adicionar Lista de Problemas	107
4.5.14	Lista de Alunos	108
4.5.15	Cadastro de Alunos pelo Administrador	109
4.5.16	Lista de Problemas	110
4.5.17	Cadastro e Edição de Lista de Problemas	111
4.5.18	Lista de Usuários	112
4.5.19	Cadastro e Edição de Usuários	112
4.5.20	Lista de Professores	113
4.5.21	Cadastro e Edição de Professores	113
4.5.22	Lista de Instituições	114
4.5.23	Cadastro e Edição de Instituições	114

4.5.24	Lista de Permissões	115
4.5.25	Cadastro e Edição de Permissões	115
4.5.26	Lista de Grupos de Administradores	116
4.5.27	Cadastro e Edição de Grupos de Administradores	116
4.5.28	Lista de Países, Estados e Cidades	117
4.5.29	Cadastro e Edição de País	117
4.5.30	Cadastro e Edição de Estado	118
4.5.31	Cadastro e Edição de Cidade	118
4.6	Diagramas de Sequência	119
4.7	Arquitetura do Software	120
5	SEGURANÇA	122
5.1	Man in the Middle	122
5.2	SQL Injection	122
5.3	HTTPS	122
6	CONSIDERAÇÕES FINAIS	124
REFERÊNCIAS		125
APÊNDICE A - DOCUMENTAÇÃO		127

LISTA DE ACRÔNIMOS

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

LISTA DE FIGURAS

Figura 2.1: Processo de Reengenharia	19
Figura 2.2: Camadas da Engenharia de Software	20
Figura 2.3: Casos de uso	22
Figura 2.4: Diagramas de Classe	23
Figura 2.5: Diagrama de Sequência	24
Figura 2.6: Diagrama e Robustez	25
Figura 2.7: Casos de uso de transferência de dados	26
Figura 2.8: Interação entre AngularJS e Arquitetura	30
Figura 3.1: Diagrama de Domínio do Portal de Algoritmos Atual	33
Figura 3.2: Cadastro de Aluno	35
Figura 3.3: Criação de Solução de Problemas	36
Figura 3.4: Gerência de Alunos	37
Figura 3.5: Gerência de Problemas	38
Figura 3.6: Edição de palavras-chave	39
Figura 3.7: Visualizando solução de um aluno	40
Figura 4.1: Diagrama de Domínio do Portal de Algoritmos Novo	43
Figura 4.2: Casos de Uso que envolvem o ator Administrador	47
Figura 4.3: Casos de Uso que envolvem o ator Professor	47
Figura 4.4: Casos de Uso que envolvem o ator Aluno	48
Figura 4.5: Interface Gráfica de Cadastro de Aluno	69
Figura 4.6: Interface Gráfica de Autenticação	70
Figura 4.7: Interface Gráfica de Boas Vindas do Administrador e Professor .	71
Figura 4.8: Interface Gráfica de Problemas	72
Figura 4.9: Interface Gráfica de Listagem Tipos de Problemas	73
Figura 4.10: Interface Gráfica de Novo Problema	74
Figura 4.11: Interface Gráfica de Palavras Chaves	75
Figura 4.12: Interface Gráfica de Dados de Testes	76
Figura 4.13: Interface Gráfica de Visualização de Solução do Problema	77

Figura 4.14: Interface Gráfica de Grupos	78
Figura 4.15: Interface Gráfica de Cadastro de Grupo	79
Figura 4.16: Interface Gráfica de Cadastro de Participantes no Grupo	80
Figura 4.17: Interface Gráfica de Cadastro de Lista de Problemas no Grupo . .	81
Figura 4.18: Interface Gráfica de Listagem de Alunos	82
Figura 4.19: Interface Gráfica de Cadastro de Alunos	83
Figura 4.20: Interface Gráfica de Lista de Problemas	84
Figura 4.21: Interface Gráfica de Cadastro e Edição de Lista de Problemas . .	85
Figura 4.22: Interface Gráfica de Lista de Usuários	86
Figura 4.23: Interface Gráfica de Cadastro e Edição de Usuários	87
Figura 4.24: Interface Gráfica de Lista de Professores	88
Figura 4.25: Interface Gráfica de Cadastro e Edição de Professores	89
Figura 4.26: Interface Gráfica de Lista de Instituições	90
Figura 4.27: Interface Gráfica de Cadastro e Edição de Instituições	91
Figura 4.28: Interface Gráfica de Lista de Permissões	92
Figura 4.29: Interface Gráfica de Cadastro e Edição de Permissões	93
Figura 4.30: Interface Gráfica de Lista de Grupos de Administradores	94
Figura 4.31: Interface Gráfica de Cadastro e Edição de Grupos de Adminis- tradores	95
Figura 4.32: Interface Gráfica de Listagem de Países, Estados e Cidades	96
Figura 4.33: Interface Gráfica de Cadastro e Edição de País	97
Figura 4.34: Interface Gráfica de Cadastro e Edição de Estado	98
Figura 4.35: Interface Gráfica de Cadastro e Edição de Cidade	99
Figura 4.36: Diagrama de Robustez de Cadastro de Aluno	100
Figura 4.37: Diagrama de Robustez de Autenticação	101
Figura 4.38: Diagrama de Robustez de Boas Vindas Administrador	102
Figura 4.39: Diagrama de Robustez de Problemas	102
Figura 4.40: Diagrama de Robustez de Tipo de Problema	103
Figura 4.41: Diagrama de Robustez de Cadastro e Edição de Problema	103
Figura 4.42: Diagrama de Robustez de Palavras-chave	104
Figura 4.43: Diagrama de Robustez de Dados de Testes	104
Figura 4.44: Diagrama de Robustez de Visualização de Solução partindo da listagem de Problemas	105
Figura 4.45: Diagrama de Robustez de Lista de Grupos	105
Figura 4.46: Diagrama de Robustez de Cadastro e Edição de Grupo	106
Figura 4.47: Diagrama de Robustez de Adicionar Participantes	106
Figura 4.48: Diagrama de Robustez de Adicionar Lista de Problemas	107
Figura 4.49: Diagrama de Robustez de Lista de Alunos	108
Figura 4.50: Diagrama de Robustez de Cadastro de Alunos pelo Administrador	109

Figura 4.51: Diagrama de Robustez de Lista de Problemas	110
Figura 4.52: Diagrama de Robustez de Cadastro e Edição de Lista de Problemas	111
Figura 4.53: Diagrama de Robustez de Lista de Usuários	112
Figura 4.54: Diagrama de Robustez de Cadastro e Edição de Usuários	112
Figura 4.55: Diagrama de Robustez de Lista de Professores	113
Figura 4.56: Diagrama de Robustez de Cadastro e Edição de Professores . . .	113
Figura 4.57: Diagrama de Robustez de Lista de Instituições	114
Figura 4.58: Diagrama de Robustez de Cadastro e Edição de Instituições . . .	114
Figura 4.59: Diagrama de Robustez de Lista de Permissões	115
Figura 4.60: Diagrama de Robustez de Cadastro e Edição de Permissões . . .	115
Figura 4.61: Diagrama de Robustez de Lista de Grupos de Administradores .	116
Figura 4.62: Diagrama de Robustez de Cadastro e Edição de Grupos de Ad- ministradores	116
Figura 4.63: Diagrama de Robustez de Lista de Países, Estados e Cidades . .	117
Figura 4.64: Diagrama de Robustez de Cadastro e Edição de País	117
Figura 4.65: Diagrama de Robustez de Cadastro e Edição de Estado	118
Figura 4.66: Diagrama de Robustez de Cadastro e Edição de Cidade	118
Figura 4.67: Diagrama de Sequência de Cadastro e Edição de Problemas . .	119
Figura 4.68: Arquitetura Lógica do Portal de Algoritmos	120
Figura 4.69: Arquitetura do Software do Portal de Algoritmos	121

LISTA DE TABELAS

Tabela 3.1: Tabelas do Banco de Dados do Portal de Algoritmos Atual	34
Tabela 4.1: Tabelas do Banco de Dados do Portal de Algoritmos Novo	44
Tabela 4.2: Requisitos funcionais	45
Tabela 4.3: Requisitos não-funcionais	46
Tabela 4.4: Caso de Uso Manter Usuários	49
Tabela 4.5: Caso de Uso Manter Alunos	50
Tabela 4.6: Caso de Uso Manter Professores	51
Tabela 4.7: Caso de Uso Manter Administradores	52
Tabela 4.8: Caso de Uso Manter Tipo de Problema	53
Tabela 4.9: Caso de Uso Manter Problemas	54
Tabela 4.10: Caso de Uso Manter Palavras-chave	55
Tabela 4.11: Caso de Uso Manter Entrada e Saídas	56
Tabela 4.12: Caso de Uso Manter Soluções de Problemas	57
Tabela 4.13: Caso de Uso Manter Listas de Problemas	58
Tabela 4.14: Caso de Uso Manter Instituições	59
Tabela 4.15: Caso de Uso Manter Grupos	60
Tabela 4.16: Caso de Uso Manter Permissões	61
Tabela 4.17: Caso de Uso Manter Grupos de Administradores	62
Tabela 4.18: Caso de Uso Manter Países	63
Tabela 4.19: Caso de Uso Manter Estados	64
Tabela 4.20: Caso de Uso Manter Cidades	65
Tabela 4.21: Caso de Uso Chat Online	66
Tabela 4.22: Caso de Uso Aluno Manter suas Informações Pessoais	67

RESUMO

O portal de algoritmos da Universidade de Caxias do Sul encontra-se tecnologicamente defasado, com falhas de segurança e pouca usabilidade.

Utilizando-se de metodologias da engenharia de *software* será realizada a evolução do portal de algoritmos. Levando em consideração a necessidade de facilitar adaptações e manutenções futuras que ocorrem no ciclo de vida de um *software*.

O desenvolvimento da solução contará com a unificação das tecnologias, seguindo os padrões de arquitetura de *software*. Além do portal de algoritmos contar com todas as funcionalidades antigas, serão implementadas diversas novas funcionalidades, que se mostraram necessárias durante o levantamento de requisitos, tanto funcionais como os não-funcionais. Tendo os casos de usos detalhados para certificar o cumprimento dos objetivos.

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

1 INTRODUÇÃO

No presente trabalho será 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 (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 do CCET (Centro de Ciências Exatas e da Tecnologia) 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, 2015; DJANGO, 2015; GOOGLE, 2015).

Um dos problemas citados acima são as versões do *Python* e *Django*, que estão em versões desatualizadas e sem suporte técnico por seus desenvolvedores.

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, 2015). *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, 2015).

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, 2015).

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).

Para atingir o objetivo proposto, o trabalho está organizado da seguinte forma:

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 é apresentado uma proposta de segurança para o servidor, essa segurança deve ser configurado na infra-estrutura.

No Capítulo 6 são apresentadas as considerações parciais do trabalho.

2 EVOLUÇÃO DE SOFTWARE

Para manter um *software* útil ele deve mudar continuamente. Essa mudança pode ser a partir de uma pressão constante de mudanças que os usuários impõem, para facilitar 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

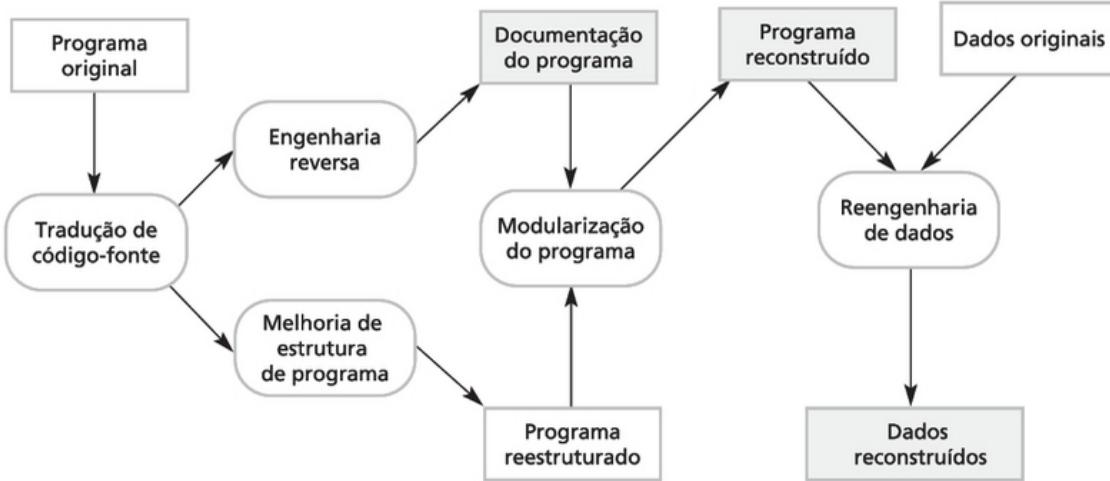
2.1 Reengenharia de Software

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.

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).

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 Casos de Uso

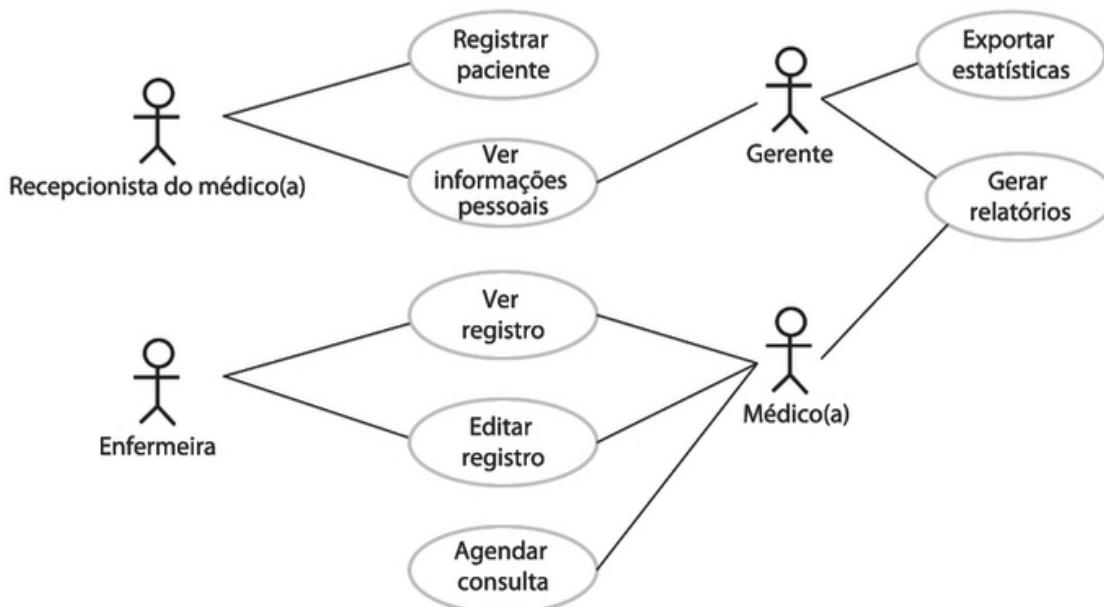
Casos de uso tem por objetivo descrever os requisitos funcionais, delimitação do contexto do sistema documentado e entendimento dos requisitos, onde cada caso de uso deve descrever somente uma funcionalidade ou objetivo do sistema (SOMMERVILLE, 2011) e (PRESSMAN, 2011).

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

Figura 2.3: 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).

2.7 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).

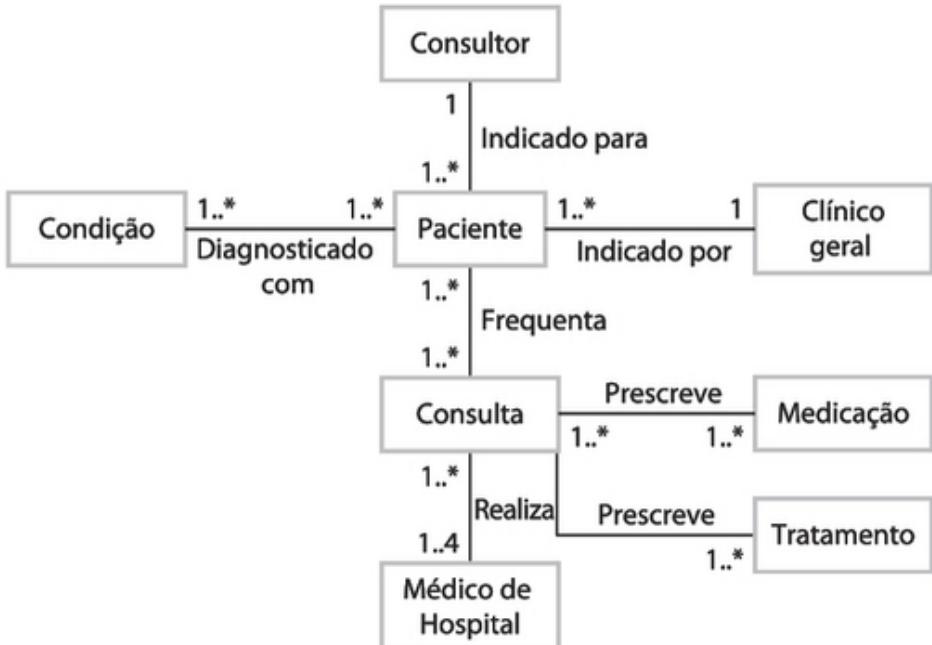
A metodologia ICONIX utiliza-se de um subconjunto da *UML* no qual apenas 4 diagramas são utilizados: diagramas de classe, diagrama de sequência, diagrama de robustez e caso de usos (ROSENBERG et al., 2005).

2.7.1 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 2.4 indica as relações entre os objetos da classe Paciente e objetos de outras classes (SOMMERVILLE, 2011).

Figura 2.4: Diagramas de Classe

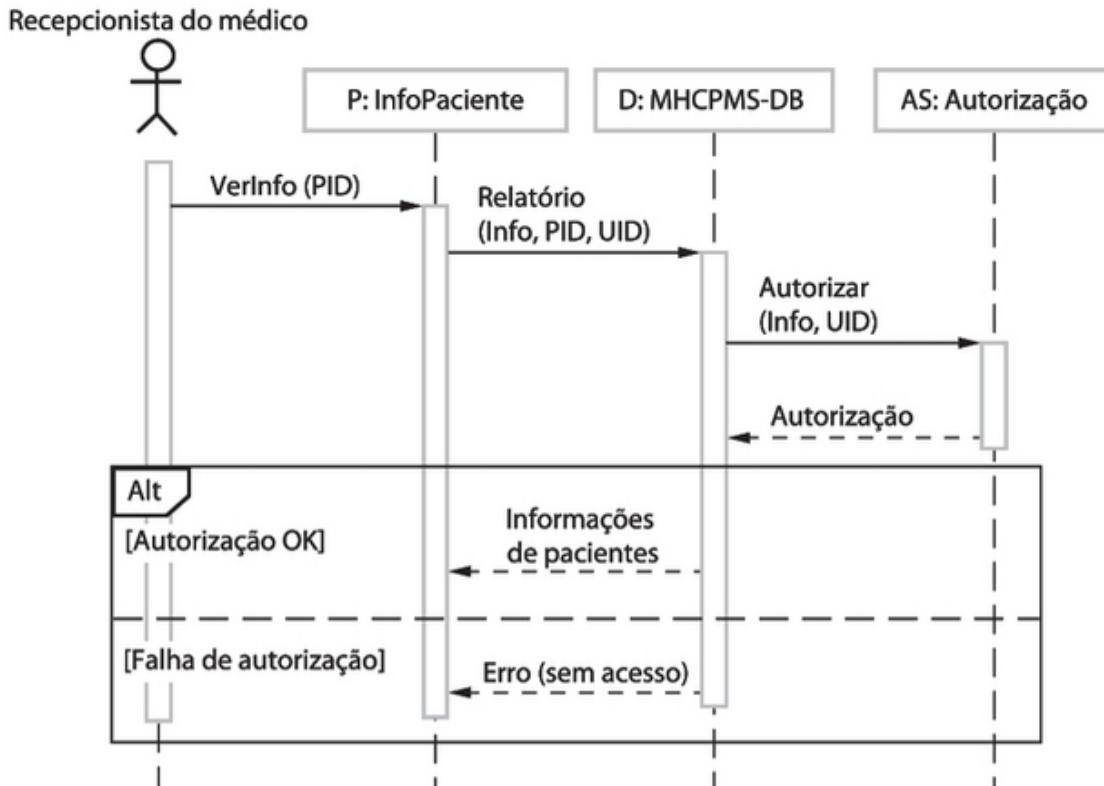


Fonte: (SOMMERVILLE, 2011)

2.7.2 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 2.5: Diagrama de Sequência



Fonte: (SOMMERVILLE, 2011)

A Figura 2.5 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.

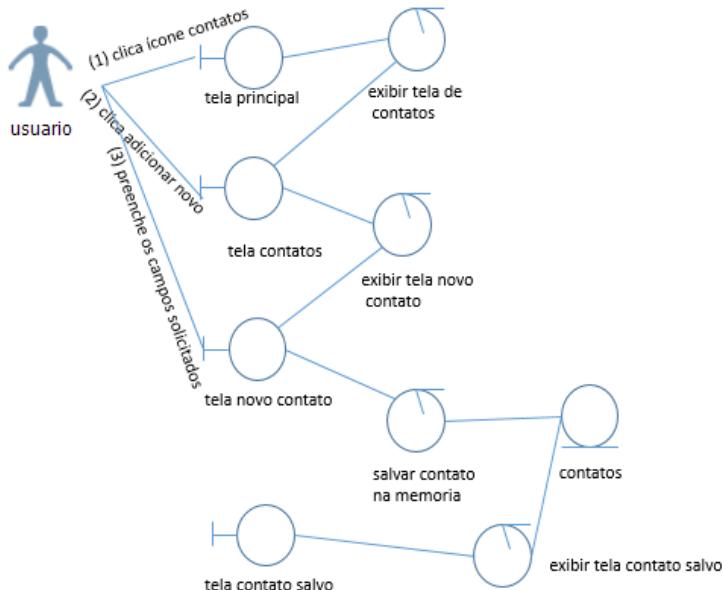
2.7.3 Diagrama de Robustez

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 2.6 representa a interação entre o usuário e as interface de um sistema, bem como todas as interações entre as interfaces.

Como podemos observar na Figura 2.6, 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 para tela de exibição de contatos.

Figura 2.6: Diagrama e Robustez



Fonte: (GALEOTE, 2015)

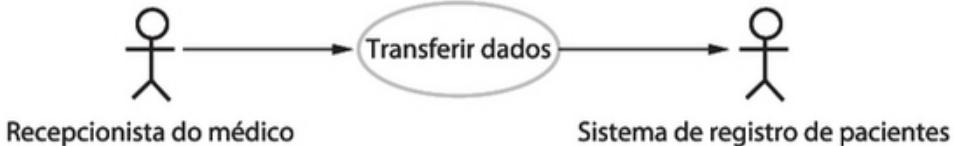
2.7.4 Casos de Usos

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)

2.8 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.

2.9 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).

2.10 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.

O portal de algoritmos atual apresenta algumas telas confusas, que podem ser observadas no Capítulo 4 na Seção 4.4. Esse trabalho pretende melhorá-las.

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.

2.11 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.

2.11.1 Python e Django

O *Python* é uma linguagem interpretada de alto nível, criada por Guido Van Rossum em 1989 e lançada em 1991 e atualmente possui o modelo de desenvolvimento comunitário (PYTHON, 2015). Com o auxílio do *framework Django*, criado origi-

nalmente para gerenciar conteúdos de um jornal da cidade de Lawrence, no Kansas, é possível definir a modelagem de dados através de classes *Python* e gerar tabelas do banco de dados para manipulação sem a necessidade direta de *SQL (Structured Query Language)* (DJANGO, 2015).

2.11.2 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, 2015).

Além disso, ela se destaca com a similaridade de sintaxe da linguagem C/C++, facilidade de internacionalização, simplicidade nas especificações, entre outras (JAVA, 2015).

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, 2015).

2.11.3 WildFly

O servidor de aplicação *WildFly* implementa a mais recente versão do JAVA *EE*, sendo mantido pela *Red Hat* (WILDFLY, 2015).

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, 2015).

2.11.4 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).

2.11.5 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*.

2.11.6 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!** (**WEB!**) (BRANAS, 2014).

O resultado é o desenvolvimento reutilizável e aplicação sustentável de componentes, 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).

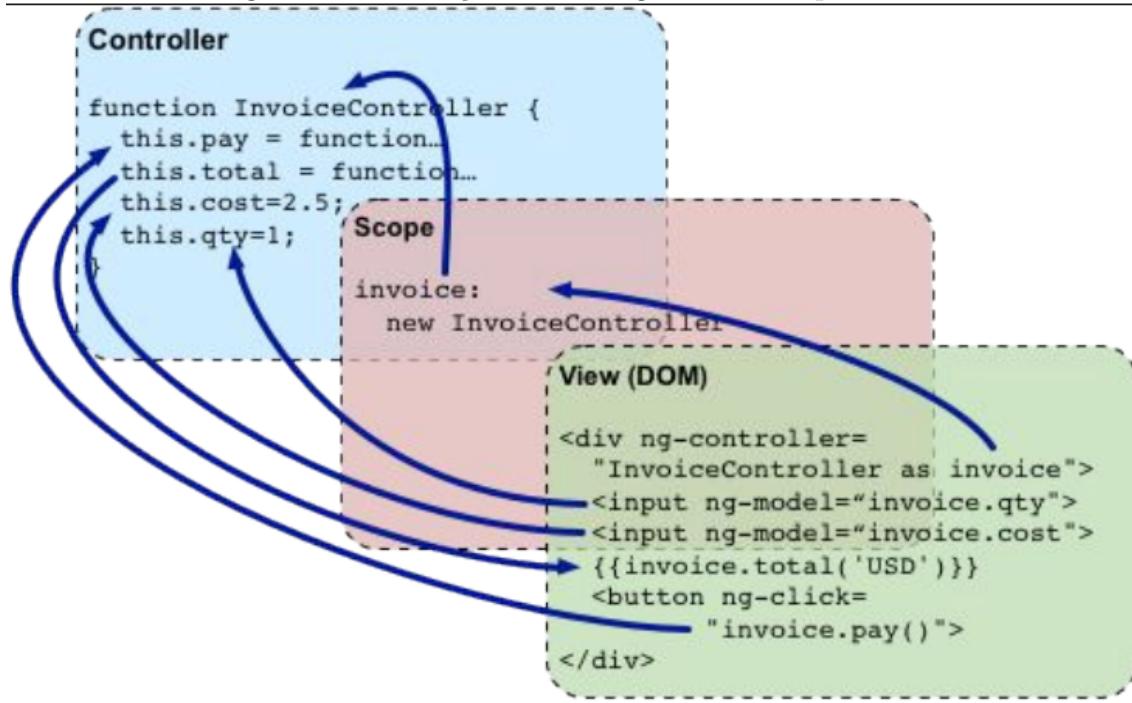
A *View* é escrita em *HTML*, que faz com que *web designers* e programadores trabalhem lado a lado, com a ajuda das diretivas, que são um tipo de extensão do vocabulário *HTML*, que traz a capacidade de executar tarefas de linguagem de programação (BRANAS, 2014).

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)

2.12 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. descrita a modelagem e seus problemas de usabilidade e de tecnologia.

3 AVALIAÇÃO DO SOFTWARE DO PORTAL DE ALGORITMOS DA UCS

Nesse capítulo é descrita a situação atual do sistema, sua modelagem e arquitetura.

3.1 Diagrama de Classe de Domínio

O diagrama de classe de domínio é a representação visual de classes conceituais, ou objetos do mundo real. Também é chamado de modelo conceitual, que significa uma representação de classes conceitos do mundo real, não de objetos de *software* (LARMAN, 2007).

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:

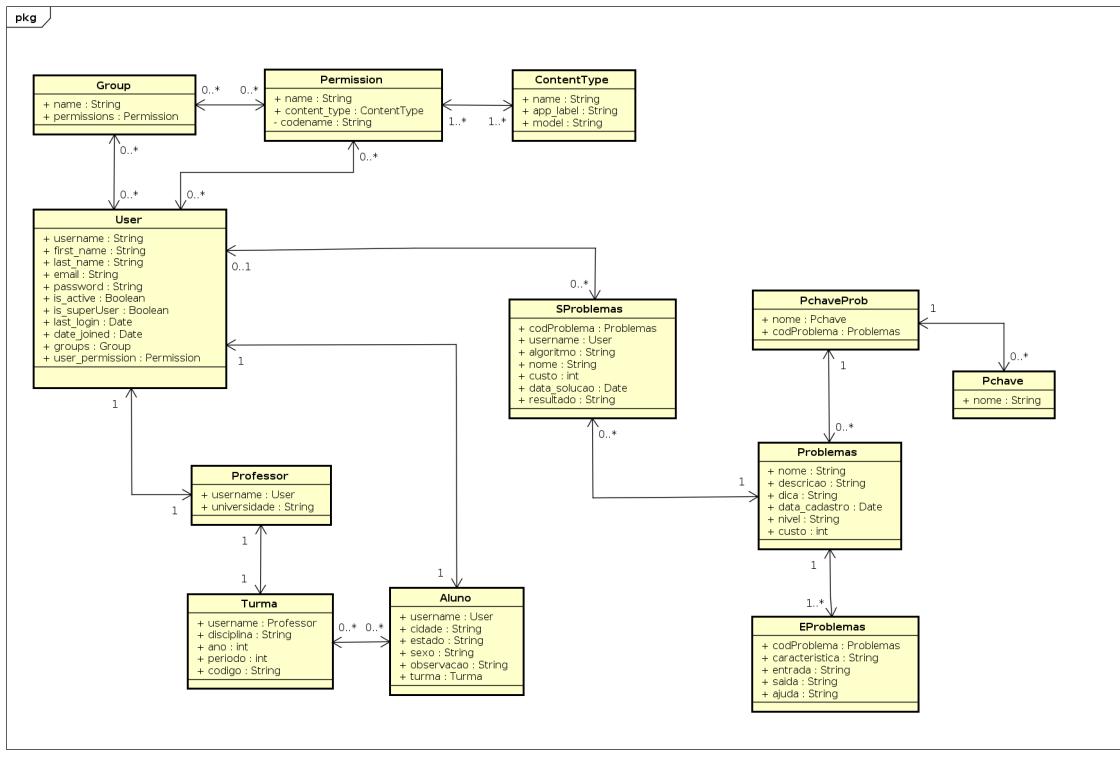
- Os campos não possuem nomes padronizados utilizando *CamelCase*;
- Nomes de campos escritos em inglês e em português;
- Nomes de classes escritos em inglês e em português;
- Nomes de classes não são intuitivos quanto ao seu objetivo;

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. Exemplo:

- lowerCamelCase;
- UpperCamelCase;

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)

A Tabela 3.1 mostra as tabelas do banco de dados e suas descrições referente 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

	Nome da Tabela	Descrição
1	User	Representa os usuários. O usuário tem algumas funções importantes no sistema. Sendo somente com ele 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 que possui acesso ao gerenciamento do portal esse pode 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 em algum exercício resolvido. 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 é exibida as interfaces gráficas.

3.2 Interfaces Gráficas

Nessa seção são descritas as interfaces gráficas atuais do software, 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

The screenshot shows the WebAlgo portal interface. On the left, there's a code editor window titled 'algoritmo' containing some pseudocode. At the top, there are buttons for 'Executar', 'Passo/Passo', 'Continuar', 'Encerrar', and checkboxes for 'Linha a Linha', 'Leia Aleatório', 'Dados de teste', and 'Valida Solução'. Below the code editor are navigation links: 'Dados Aleatórios', 'Exemplo de Entrada/Saída [PARA O PROBLEMA]', 'Sobre', 'Avisos de Compilação', 'Entrada e Saída na Execução', and 'Variáveis / Computação'. A note at the bottom left says 'Interpretador de Algoritmos WebAlgo - Versão Beta - 1.15' and 'Desenvolvido por professores do CCTI - Universidade de Caxias do Sul'. A help link 'AJUDA - Dicas de como usar o portal !!!' is also present. On the right, there's a user login section with fields for 'Usuário' and 'Senha' and a 'Entrar' button. Below it is a registration form with fields for 'Primeiro Nome', 'Último Nome', 'Login (entre 7 e 10)', 'Email', 'Quem é você?', 'Sexo' (with radio buttons for 'Masculino' and 'Feminino'), 'Cidade', 'Estado', and 'Senha'. There are buttons for 'Concluir meu Cadastro' and 'Limpar Campos preenchidos!'. A note at the bottom right says 'Campos com * são obrigatórios!'.

Fonte: (AUTOR, 2018)

Problemas encontrados:

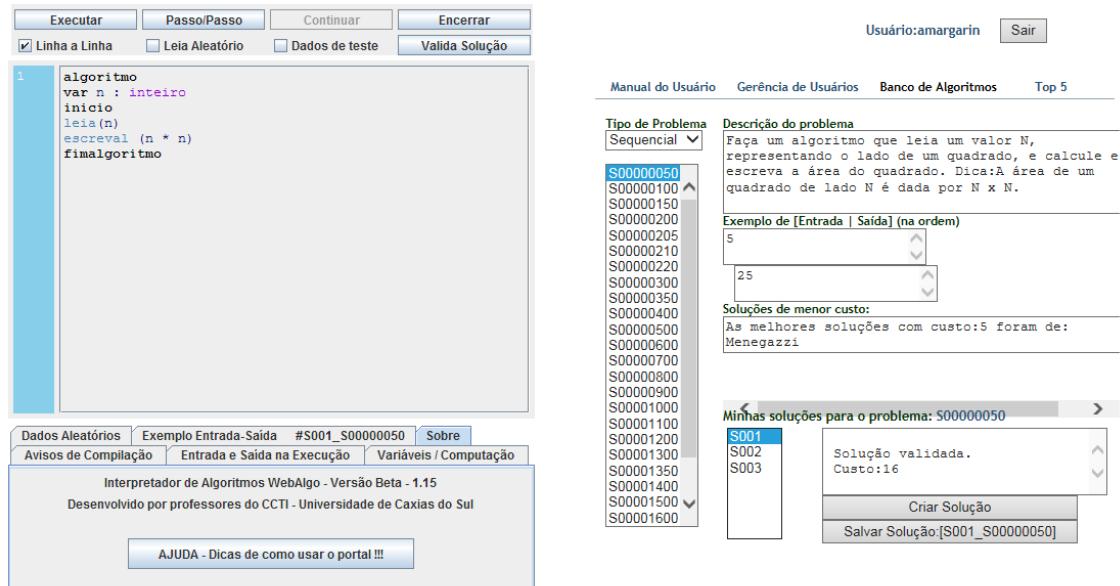
- *Java Applet* não funciona mais em nenhum navegador de internet atual.
- Pouca Usabilidade, uma vez que o cadastro do aluno está na mesma interface da programação de algoritmos.

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.

Figura 3.3: Criação de Solução de Problemas



Fonte: (AUTOR, 2018)

Problemas encontrados:

- *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.

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

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 a user interface for managing problems in a system called PortAlgo. On the left, there's a sidebar with a list of existing problems (S00000050 is selected). The main area displays problem details for S00000050:

- Descrição do problema:** Fazer 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:** A área de um quadrado de lado N é dada por N x N.
- Palavras chave o problema:** (empty field)
- Exemplo de entrada (na ordem):** 5 | 3 | 4 | 6
- Saída após e/ou durante processamento:** 25 | 9 | 16 | 36

At the bottom, there are buttons: Gravar Problema, Alterar Descrição/Dicas, Alterar Palavras Chave (highlighted in blue), and Alterar Entradas/Saídas.

Fonte: (AUTOR, 2018)

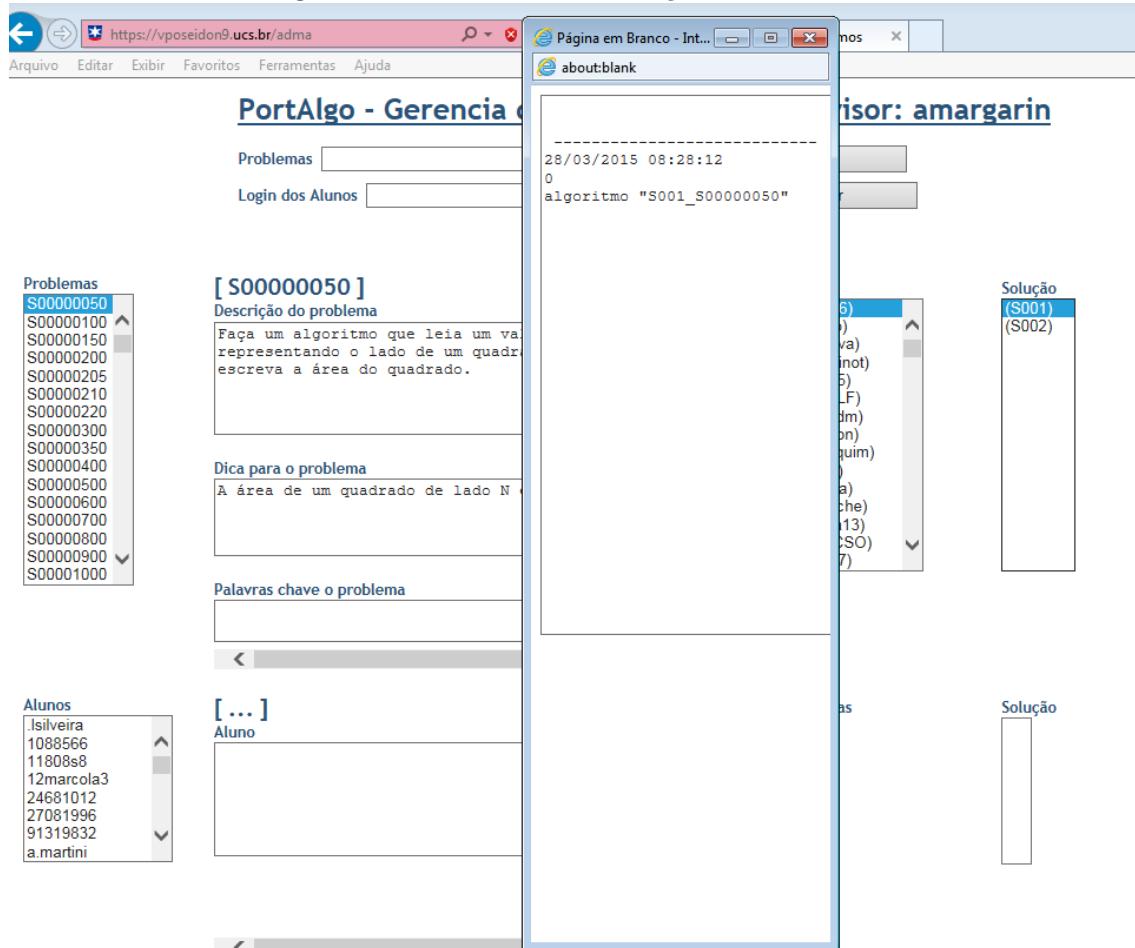
Problemas encontrados:

- 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.

Os problemas encontrados no *software* atual prejudicam a usabilidade do mesmo. Atualmente ele está limitado a apenas um navegador de internet, no caso o *Internet Explorer*.

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 funcio-

nalidades, utilizando-se de tecnologias mais atuais.

Foram apresentados nesse capítulo os 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, é 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*. 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 um 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:

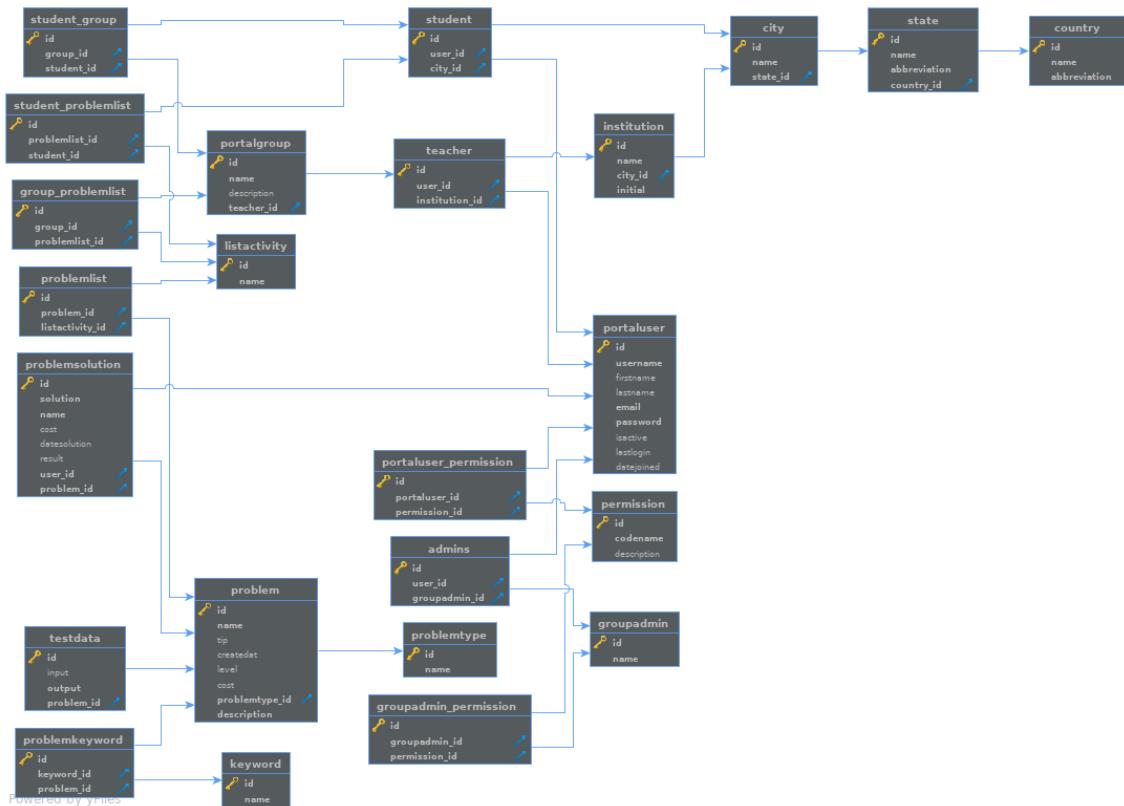
- Manter Administradores.
- Manter Grupos de Administradores.
- Manter Usuários, Alunos e Professores.
- Manter Grupos de Alunos.
- Manter Instituições.
- Manter Permissões.
- Manter Países, Estados e Cidades.
- Manter Tipos de Problemas.
 - Manter Problemas.
 - Manter Entradas e Saídas
 - Manter Palavras-chave
 - Manter Listas de Problemas
 - Manter Soluções de Problemas

- Chat para comunicação entre usuários
- Relatórios.
 - Relatórios de acesso: esse relatório demonstra todas as vezes que um usuário acessou o portal, podendo ser filtrado por data.
 - Relatório de problemas solucionados: esse relatório demonstra todas as soluções submetidas pelos usuários, podendo ser filtrado por data.

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 novo. Nesse novo diagrama foram melhoradas as descrições dos campos, descrevendo-os em inglês e utilizando *CamelCase*.

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

	Nome da Tabela	Descrição
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 do portal de algoritmos, classe nova que surgiu da necessidade de padronizar nomes de cidades.
3	country	Representa os países do portal de algoritmos, 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 do portal de algoritmos, 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 padronizadas descrições de campos e descrições de classes para o padrão *CamelCase*.

Na seção seguinte são descritos os requisitos de projeto que são os requisitos funcionais e não-funcionais.

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 software deve possuir um chat para comunicação entre os usuários.
RF-019	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 ter uma interface gráfica de fácil operação.
RFN-004	O software deve executar nos principais navegadores de internet.

Fonte: (AUTOR, 2018)

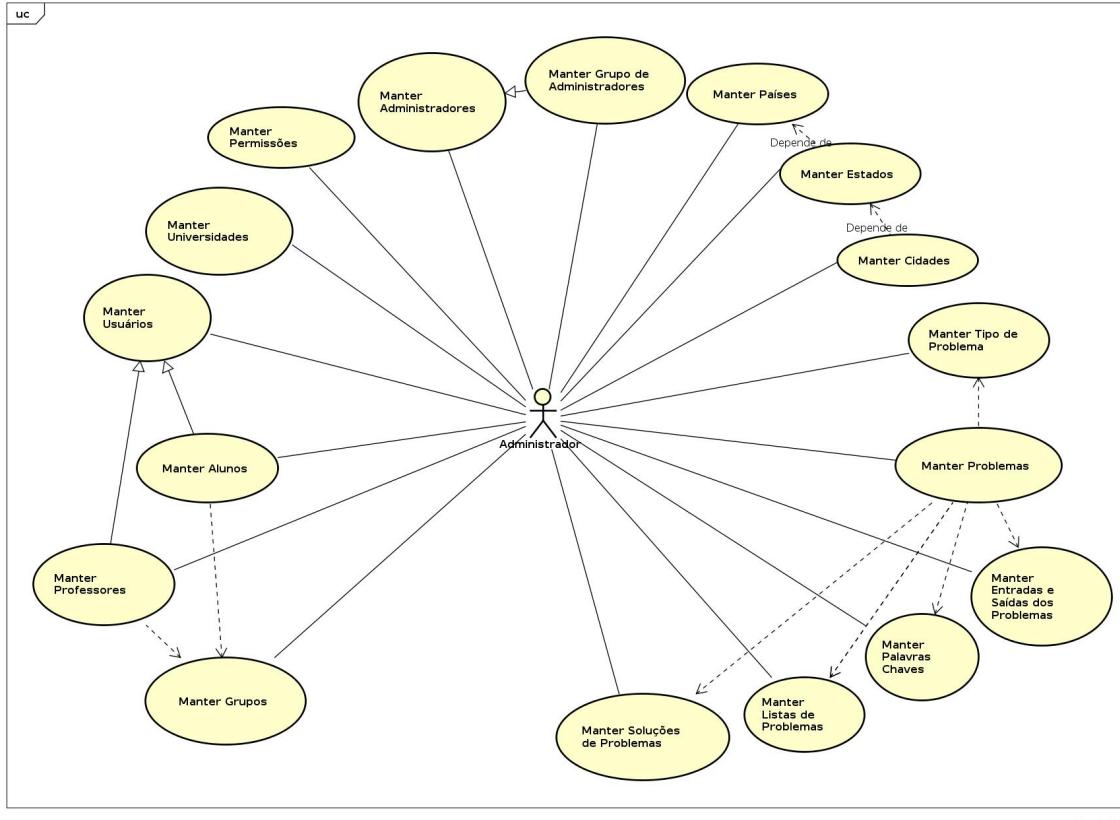
Cada requisito funcional elicitado pode se tornar um caso de uso. Esses casos de uso serão descritos na seção seguinte.

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.

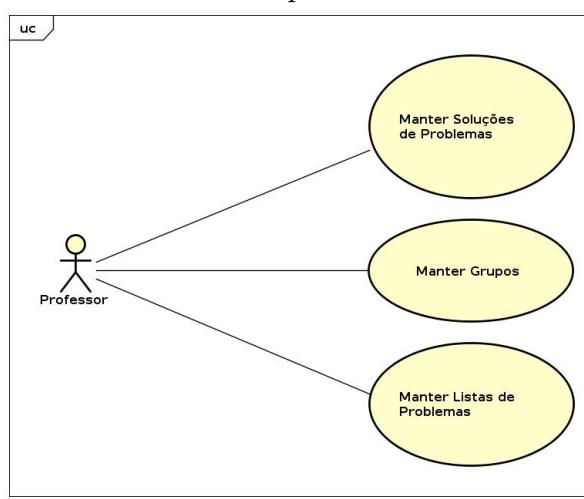
Figura 4.2: Casos de Uso que envolvem o ator Administrador



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.

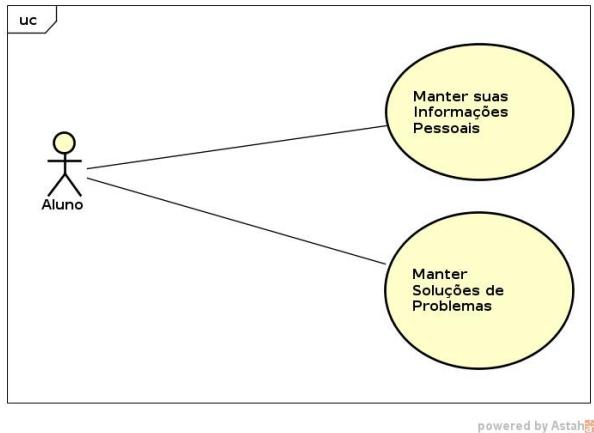
Figura 4.3: Casos de Uso que envolvem o ator Professor



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Fonte: (AUTOR, 2018)

4.3.1.18 Chat Online

A Tabela 4.21 descreve o caso de uso do *chat online*, ele servirá para comunicação interna em tempo real dentro do portal de algoritmos.

Tabela 4.21: Caso de Uso Chat Online

CASO DE USO 018 – Sistema deve possuir um chat para comunicação entre os usuários (RF-018)	
Descrição	O chat servirá para comunicação por troca mensagens de texto privadas, em grupo ou públicas.
Autor	Administrador, Professor e Aluno.
Pré-condição	CASO DE USO 001. Usuário autenticado no sistema
Fluxo principal	<ol style="list-style-type: none"> 1. Usuário acessa o sistema 2. Sistema exibe lista de usuários autenticados 3. Usuário seleciona um usuário para trocar mensagens 4. Sistema abre nova janela para a troca de mensagem 5. Usuários trocam mensagens
Fluxo alternativo e exceções	
Pós-condição	Troca de mensagens entre usuários

Fonte: (AUTOR, 2018)

4.3.1.19 Manter suas Informações Pessoais

A Tabela 4.22 descreve o caso de uso de cadastro, edição e remoção das informações pessoais do aluno.

Tabela 4.22: Caso de Uso Aluno Manter suas Informações Pessoais

CASO DE USO 019 – Manter suas informações pessoais (RF-019)	
Descrição	O aluno pode se cadastrar, editar e remover suas informações pessoais.
Autor	Aluno.
Pré-condição	Não possui.
Fluxo principal	<ol style="list-style-type: none"> 1. Usuário acessa cadastro público do portal. 2. Sistema exibe formulário para cadastro. 3. Usuário preenche todos os campos obrigatórios. 4. Sistema retorna para formulário de login.
Fluxo alternativo e exceções	Item 3: <ol style="list-style-type: none"> 1. Usuário não preenche todos os campos obrigatórios. 1. Sistema exibe mensagens de erros.
Pós-condição	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 the title 'Portal de Algoritmos'. The address bar indicates the URL is 'localhost:4200/registro-aluno'. The main content is a registration form titled 'Quero me cadastrar'. It has two columns of input fields: 'Nome' and 'Sobrenome', 'E-mail' and 'Usuário', 'Senha' and 'Repita a senha'. Below these are dropdown menus for 'Instituição' and 'Cidade'. At the bottom are 'Limpar' and 'Salvar' buttons. The footer of the page displays the logo 'UCS' and the text 'Campus Sede: Rua Francisco Getúlio Vargas, 1130 - CEP 95070-560 - Caxias do Sul' and 'Fone: +55 (54) 3218-2100 - Universidade de Caxias do Sul'.

Fonte: (AUTOR, 2018)

Nessa interface gráfica o aluno preenche todos os campos abaixo:

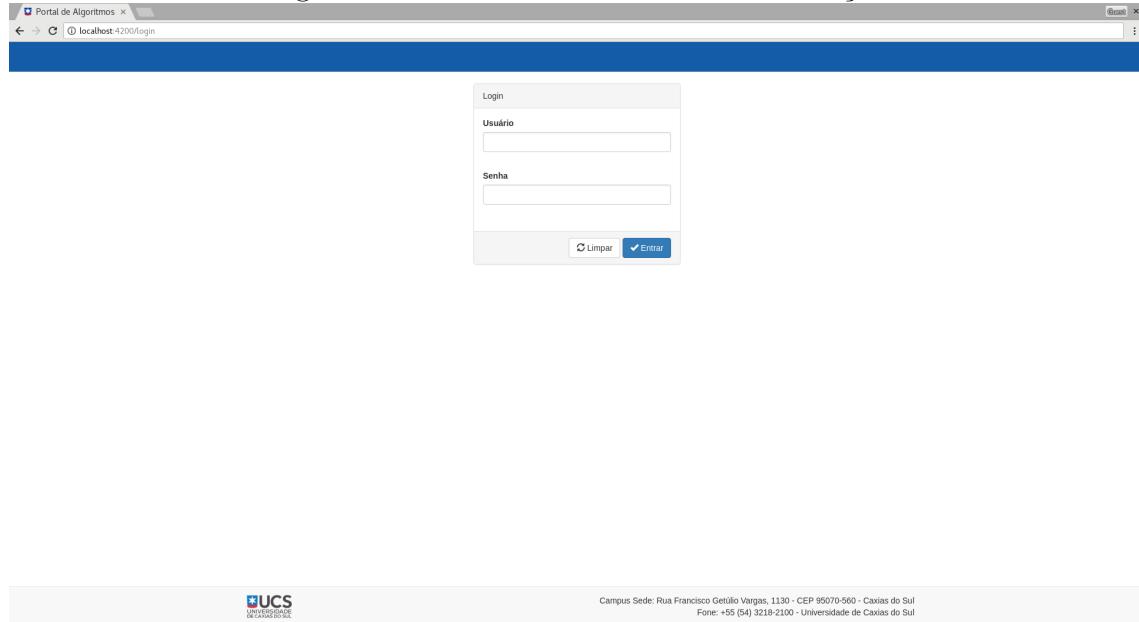
- Nome.
- Sobrenome.
- E-mail.
- Login: o login é seu *username* que será utilizado para acessar o portal de algoritmos.
- Senha.
- Repita a senha: nesse campo faz-se a verificação se a senha digitada anteriormente corresponde a essa.
- Instituição: exibe todas instituições cadastradas pelo administrador.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

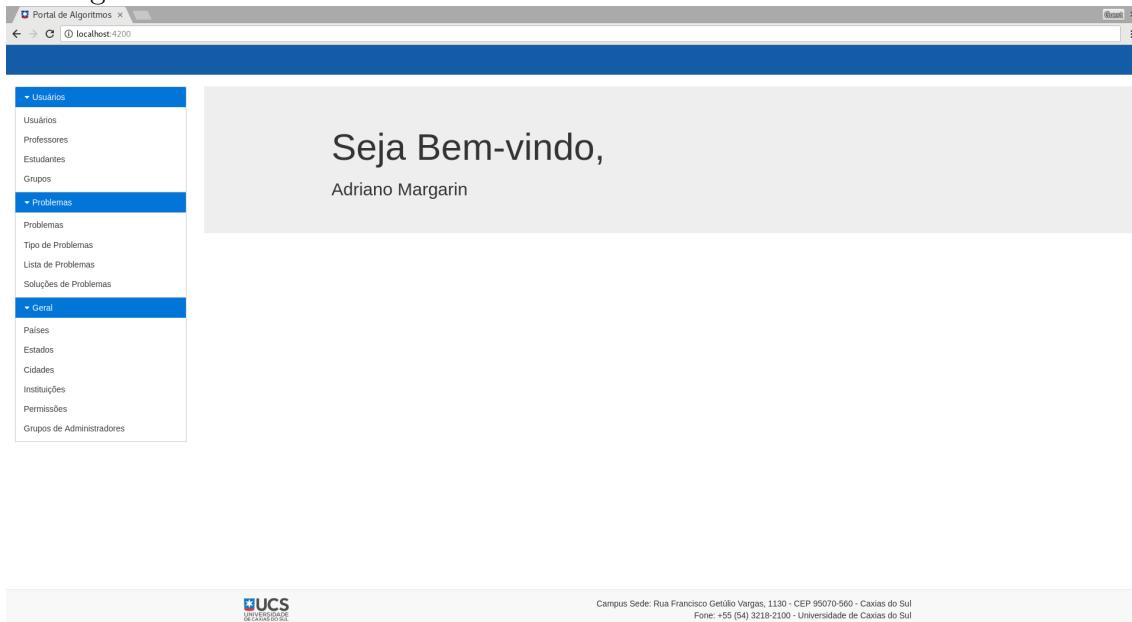
Nessa interface gráfica o aluno, professor ou administrador realiza a autenticação no portal de algoritmos. A autenticação somente é autorizada para usuários ainda ativos no sistema.

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 administrador 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

Sidebar (Left):

- Usuários
 - Usuários
 - Professores
 - Estudantes
 - Grupos
- Problemas
 - Problemas
 - Lista de Problemas
 - Soluções de Problemas
- Geral
 - Países
 - Estados
 - Cidades
 - Instituições
 - Permissões
 - Grupos de Administradores

Footer:

Campus Sede: Rua Francisco Getúlio Vargas, 1130 - CEP 95070-560 - Caxias do Sul
Fone: +55 (54) 3218-2100 - Universidade de Caxias do Sul

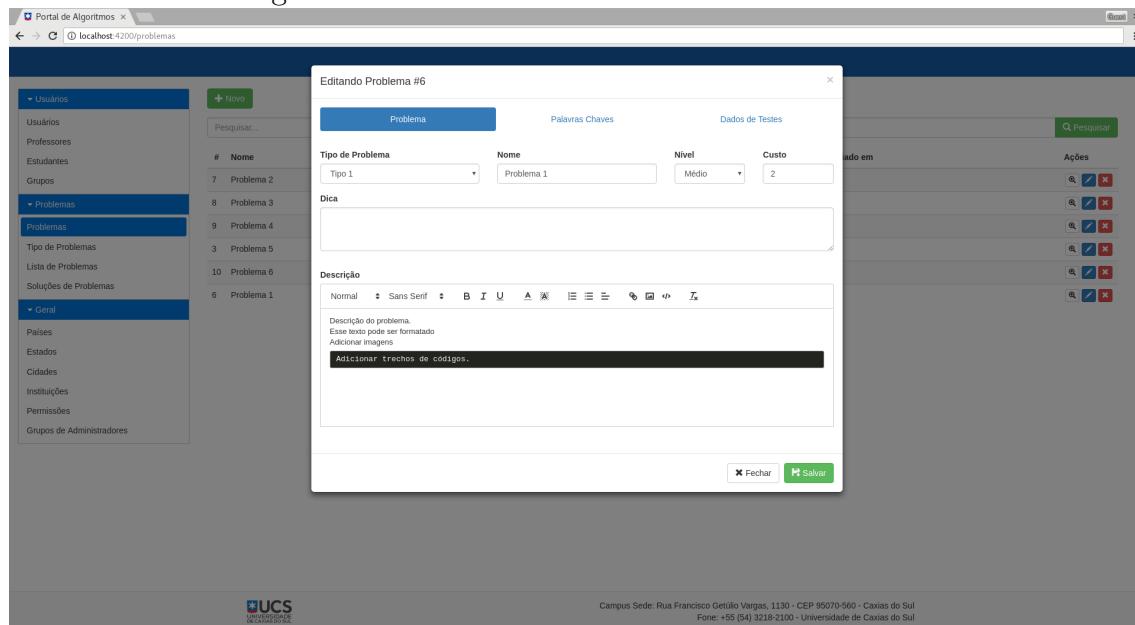
Fonte: (AUTOR, 2018)

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



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.

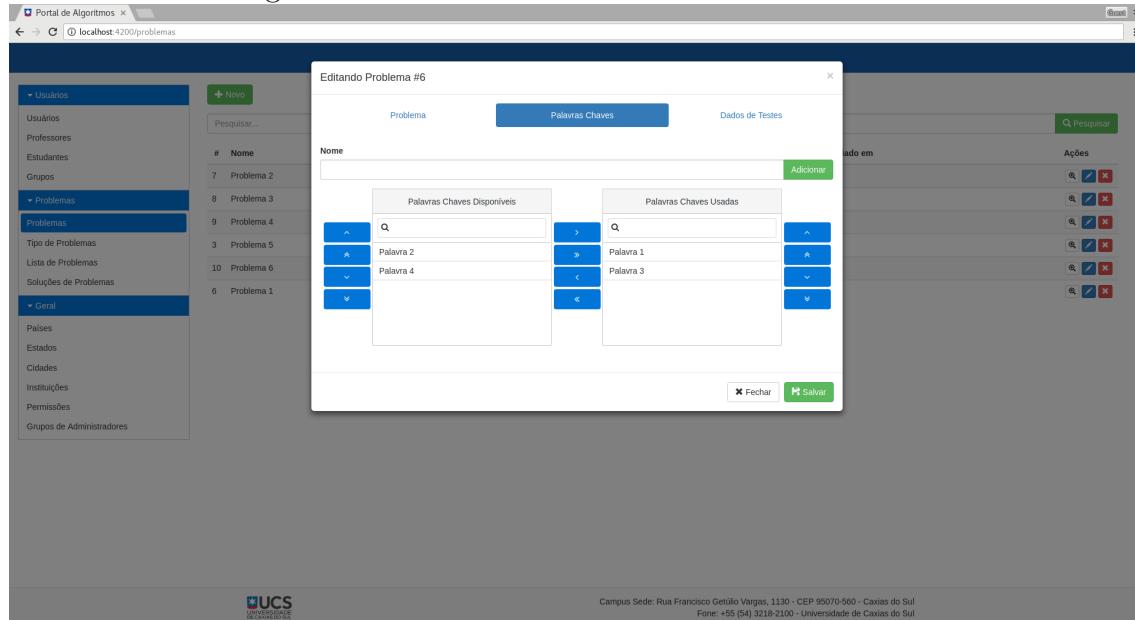
O cadastro de dados de testes é descrito na Seção 4.4.8

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



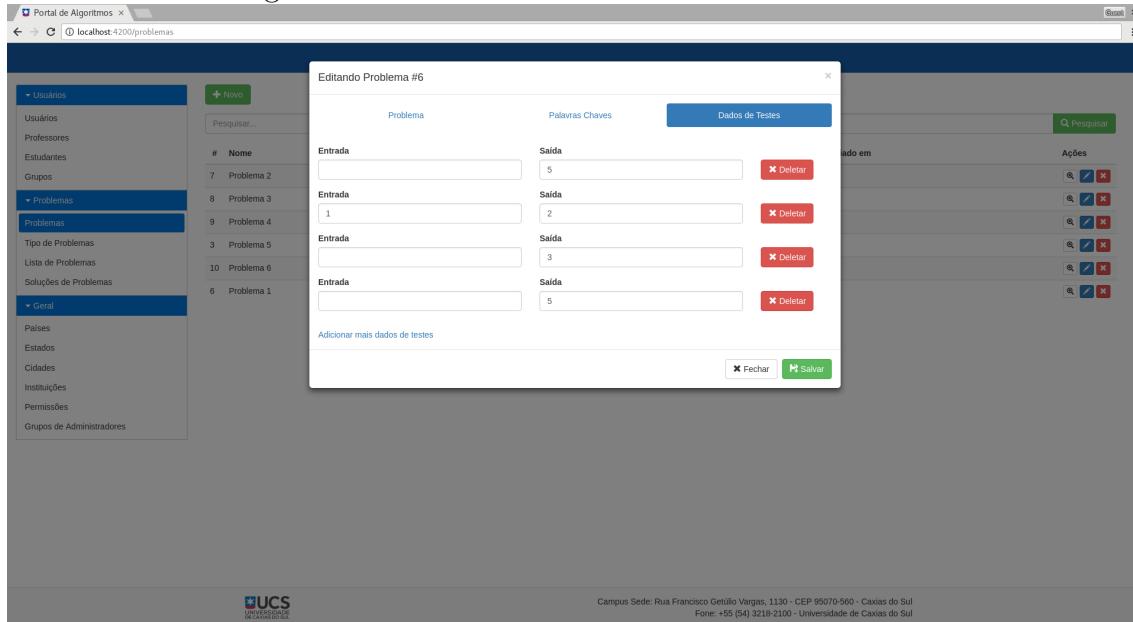
Fonte: (AUTOR, 2018)

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



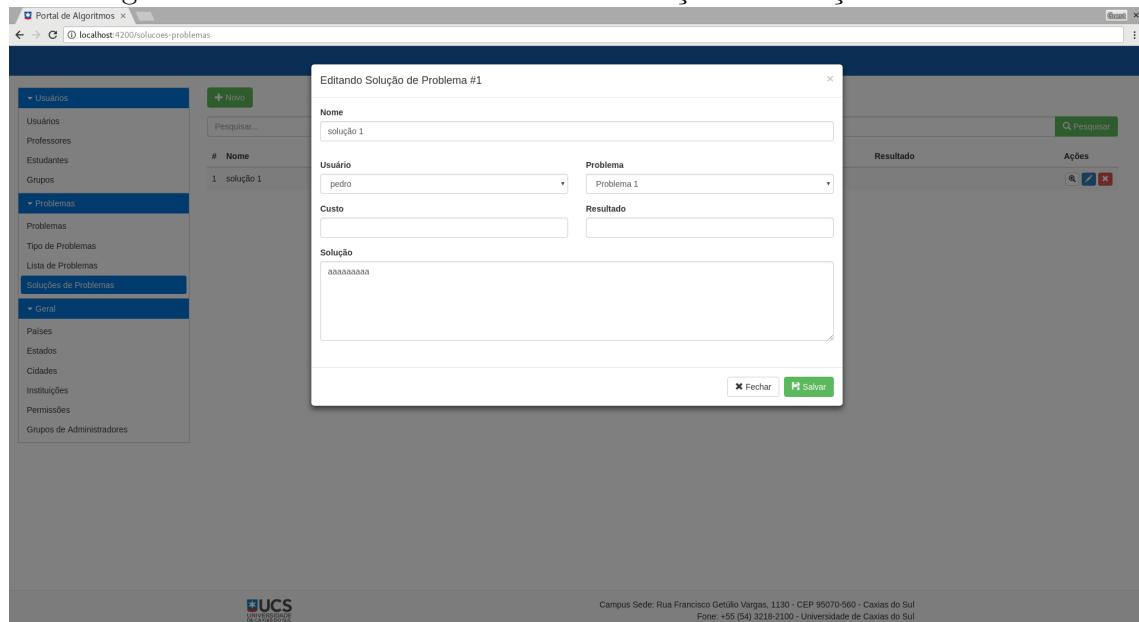
Fonte: (AUTOR, 2018)

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

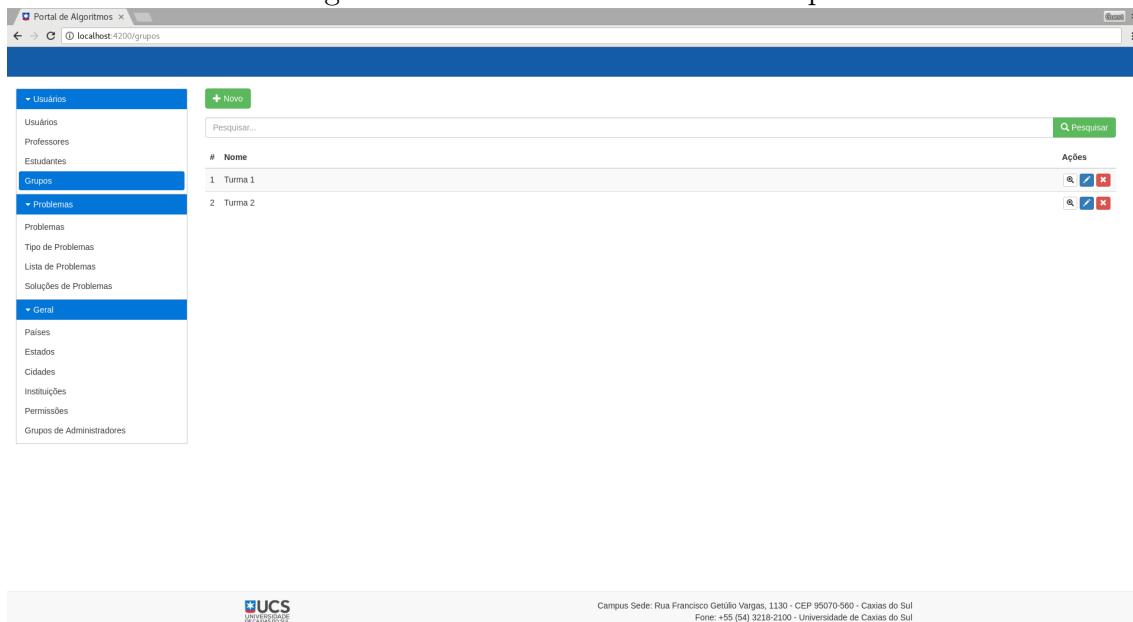


Fonte: (AUTOR, 2018)

4.4.10 Lista de Grupos

A Figura 4.14 é a interface gráfica de listagem dos grupos ja 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

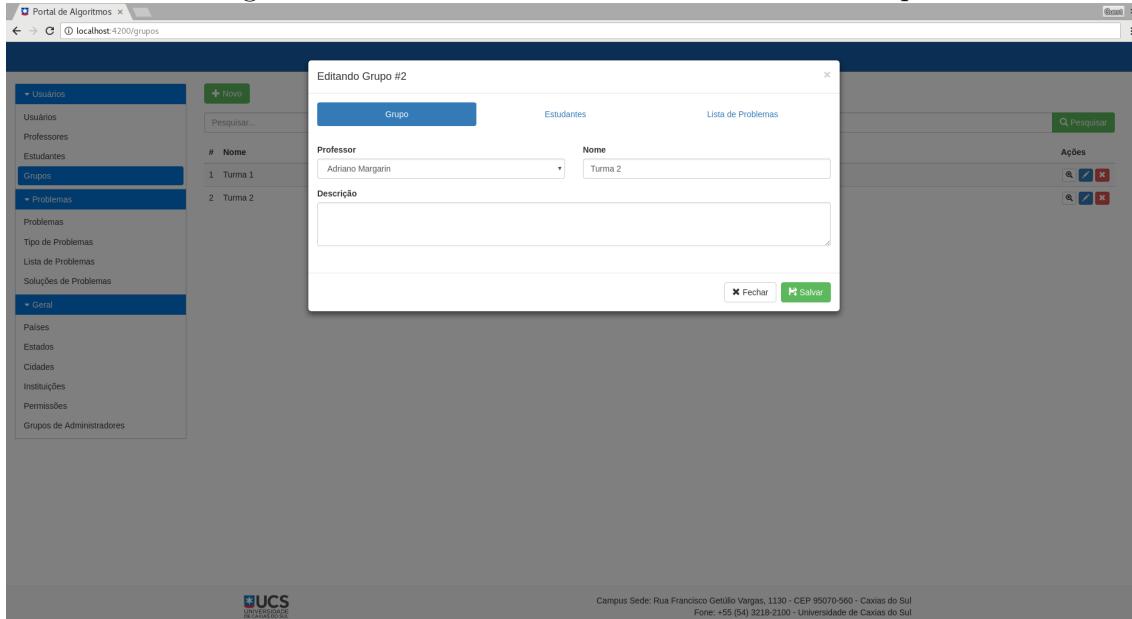
- Remover Grupo.

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



Fonte: (AUTOR, 2018)

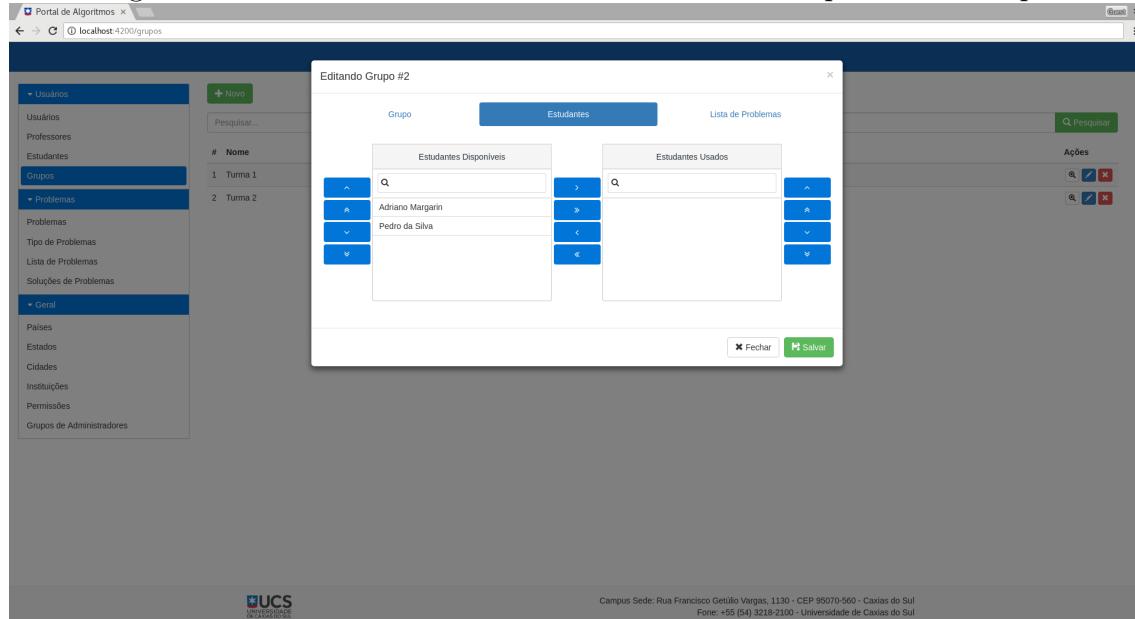
A interface gráfica está dividida em três abas, Grupo, Participantes, Lista de Problemas sendo que cada uma dessas abas estão descritas abaixo:

- Grupo.
Para o cadastro de um novo grupo, basta preencher os campos, Nome e Descrição.
- Participantes.
O cadastro de participantes é descrito na Figura 4.16
- Lista de Problemas.
O cadastro de lista de problemas é descrito na Figura 4.17

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



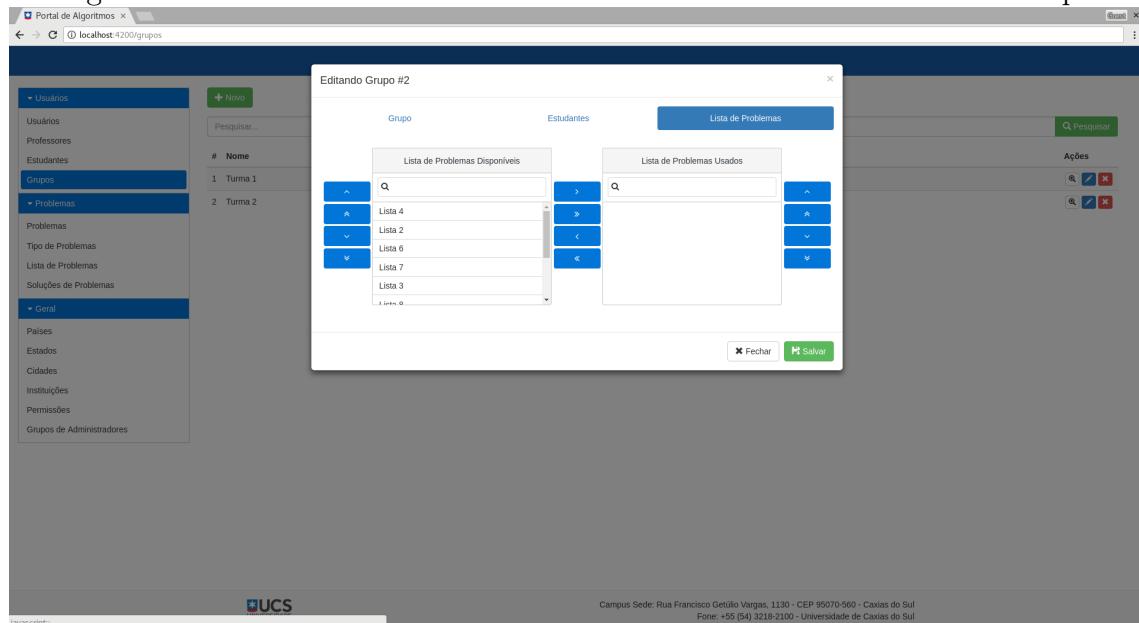
Fonte: (AUTOR, 2018)

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



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)

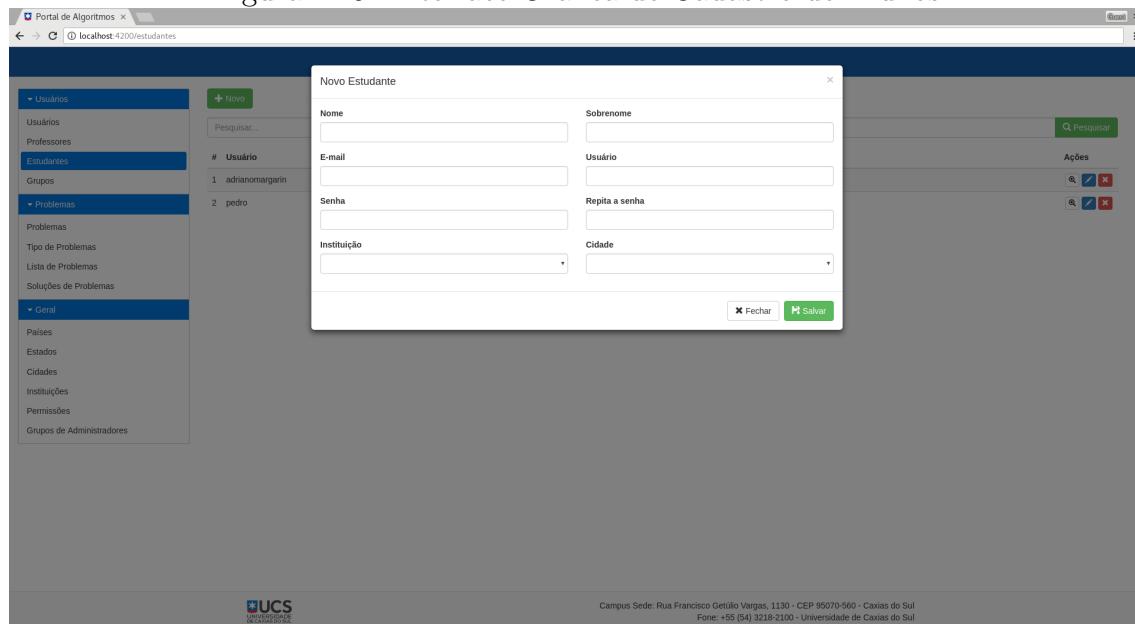
Nessa interface gráfica encontramos as seguintes ações:

- 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



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)

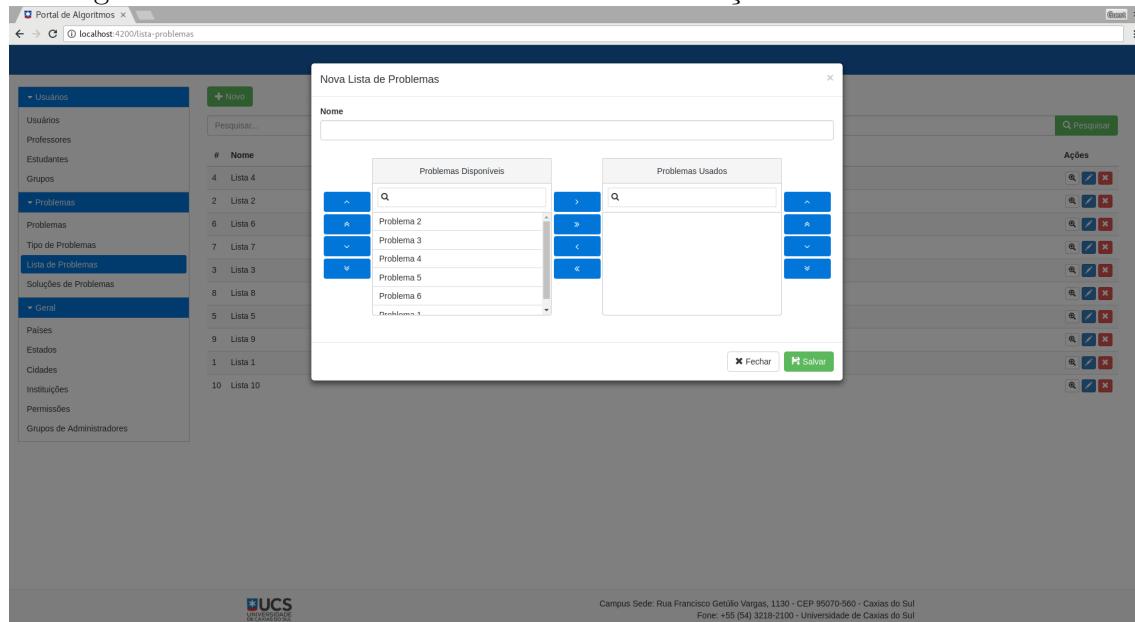
Nessa interface gráfica temos as seguintes ações:

- 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.
- Visualizar Solução: selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura ??

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			<input checked="" type="checkbox"/>	
5	adrianomargarin	amargarin@ucs.br	Adriano	Margarin			<input checked="" type="checkbox"/>	

Fonte: (AUTOR, 2018)

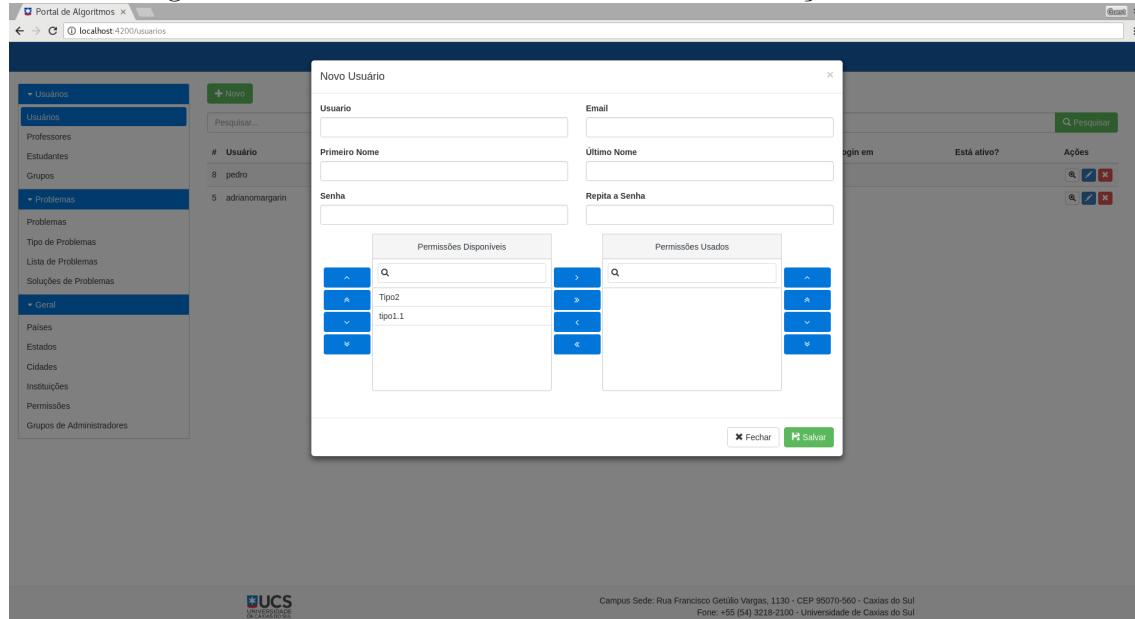
Nessa interface gráfica temos as seguintes ações:

- 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



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)

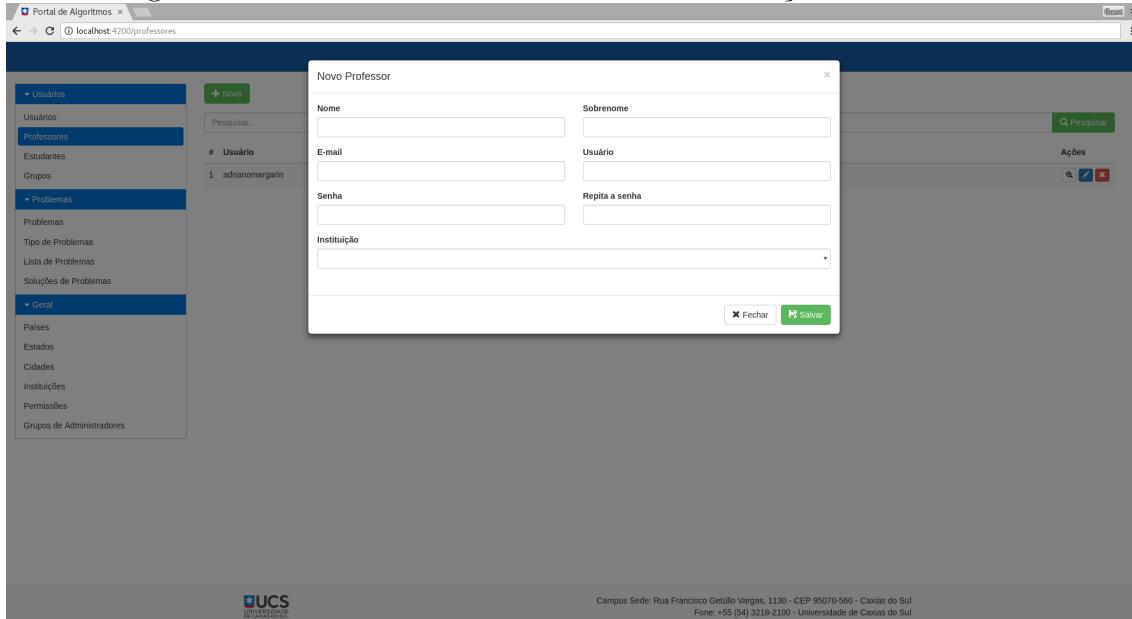
Nessa interface gráfica temos as seguintes ações:

- 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



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/RS	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)

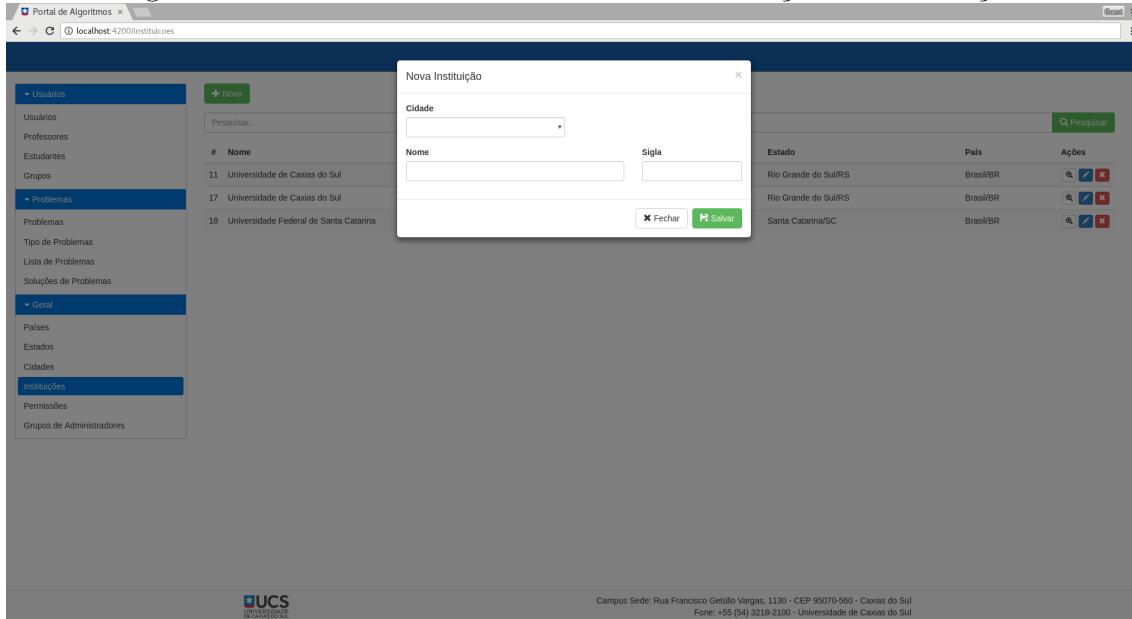
Nessa interface gráfica temos as seguintes ações:

- 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



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.

Selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.29.

- Editar Permissões.

Selecionando essa ação, o administrador é direcionado para a interface gráfica da Figura 4.29.

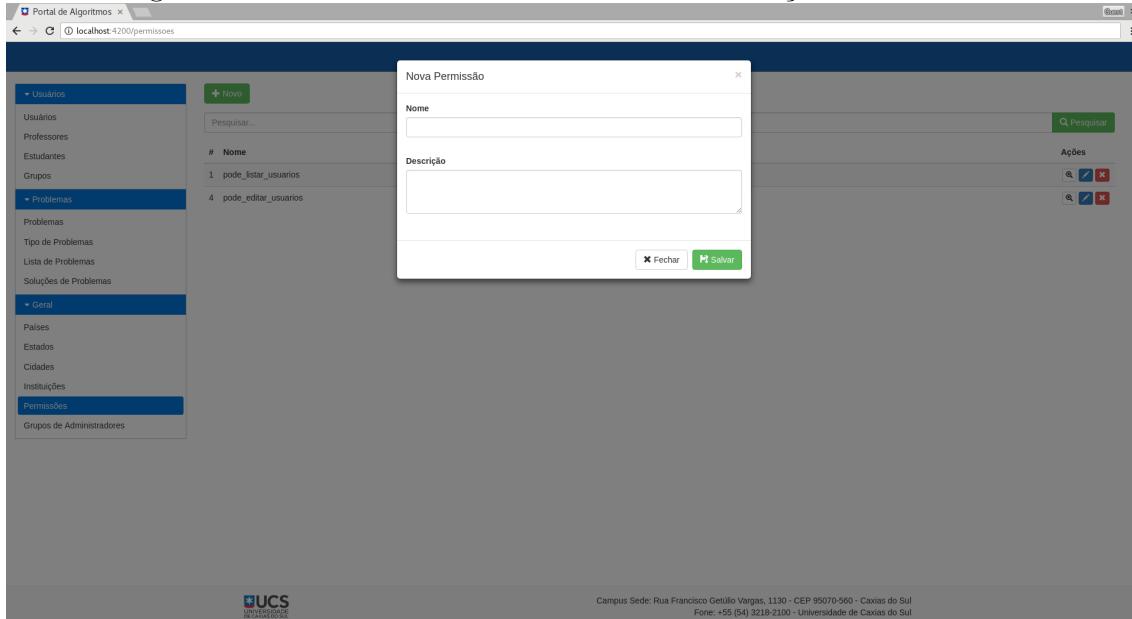
- Remover Permissões.

Selecionando 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



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)

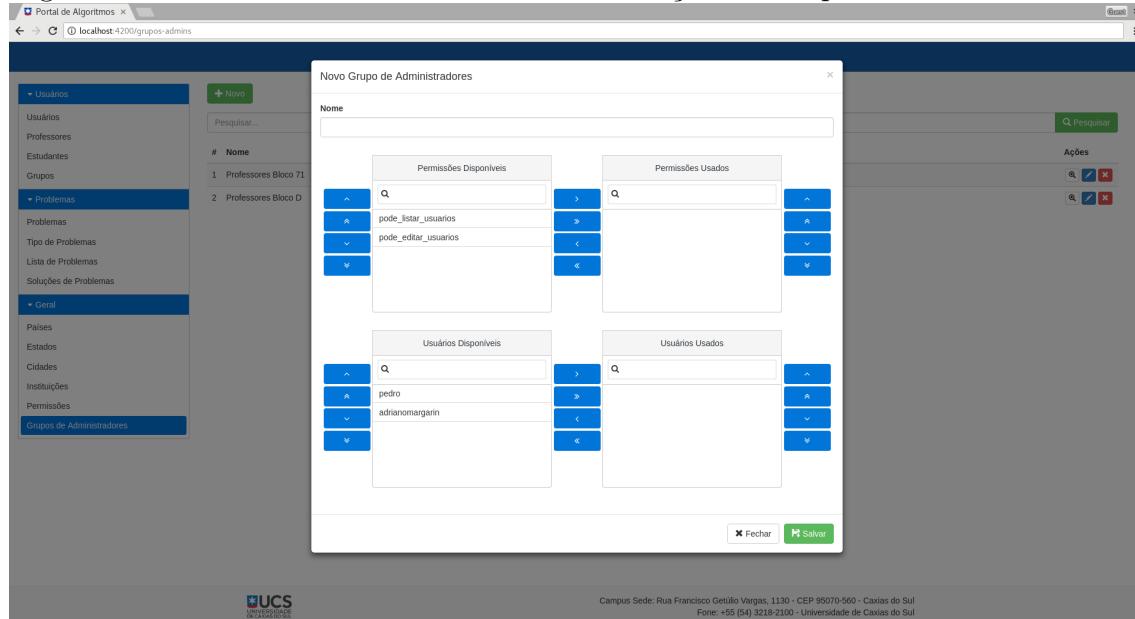
Nessa interface gráfica temos as seguintes ações:

- 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



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)

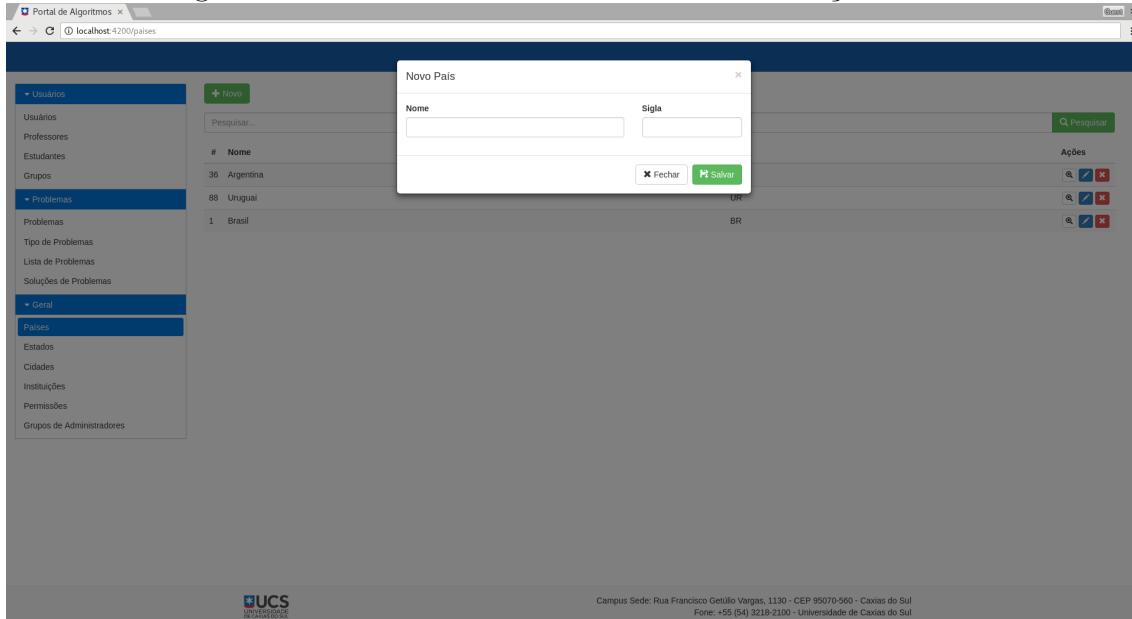
Nessa interface encontramos as seguintes ações:

- 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



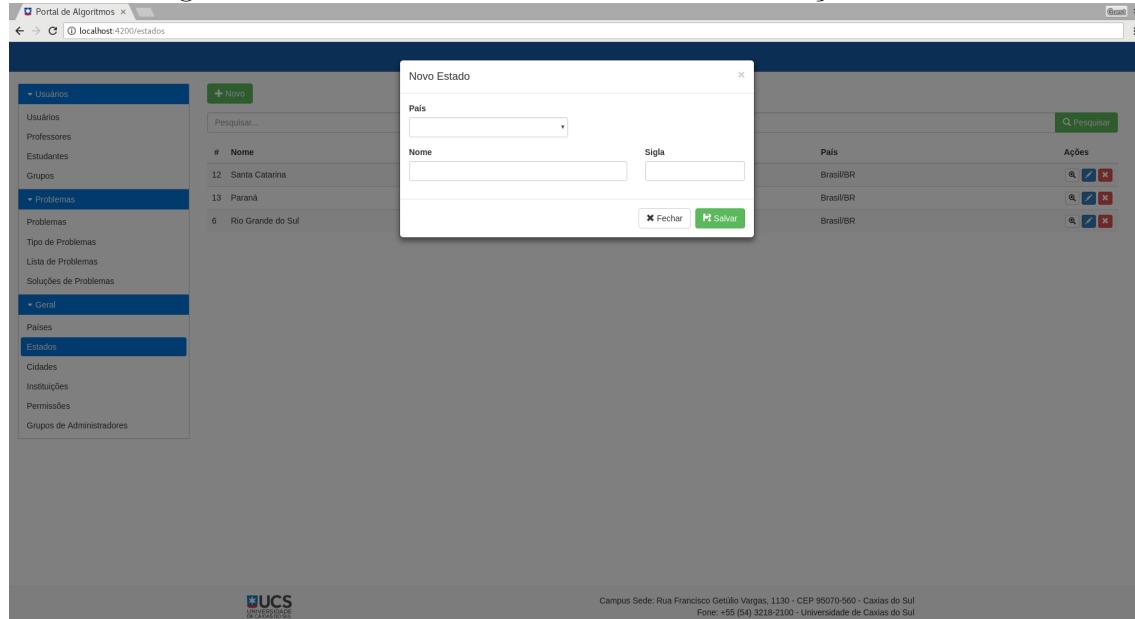
Fonte: (AUTOR, 2018)

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



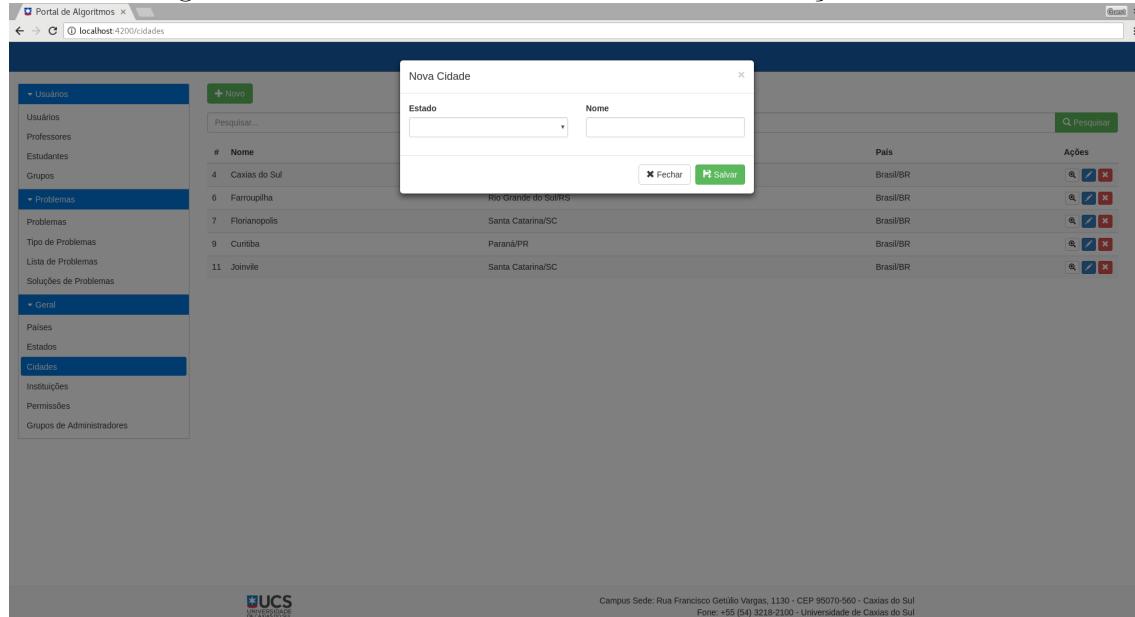
Fonte: (AUTOR, 2018)

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



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.

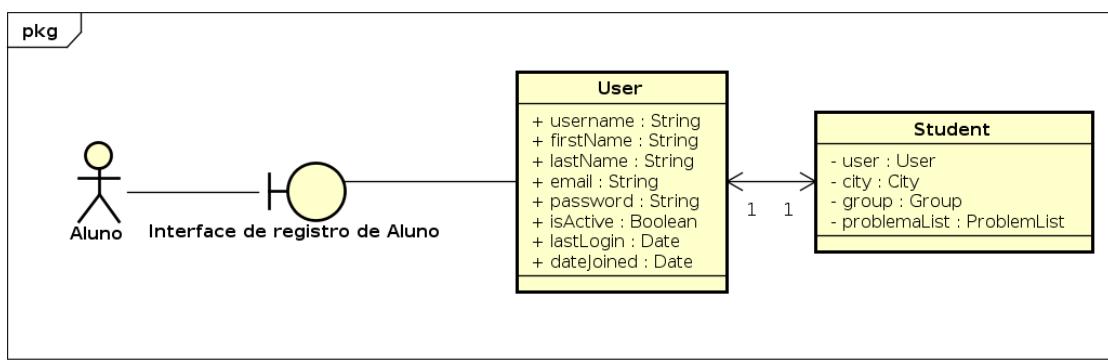
4.5 Diagramas de Robustez

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 da Figura 4.5.

Figura 4.36: Diagrama de Robustez de Cadastro de Aluno

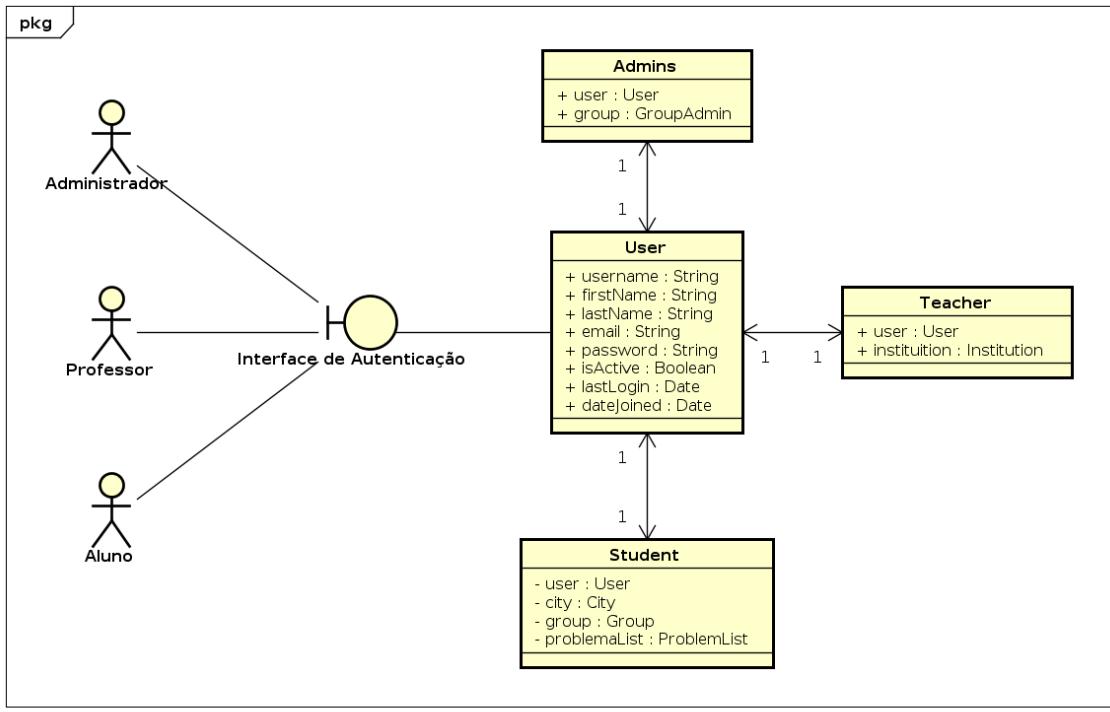


Fonte: (AUTOR, 2018)

4.5.2 Autenticação

A Figura 4.37 demonstra as associações das classes de domínio na interface gráfica da Figura 4.6.

Figura 4.37: Diagrama de Robustez de Autenticação



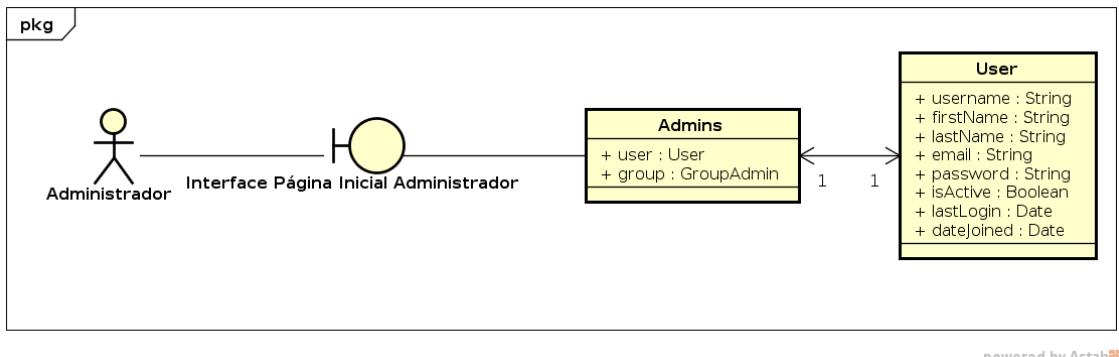
powered by Astah

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 da Figura 4.7.

Figura 4.38: Diagrama de Robustez de Boas Vindas Administrador

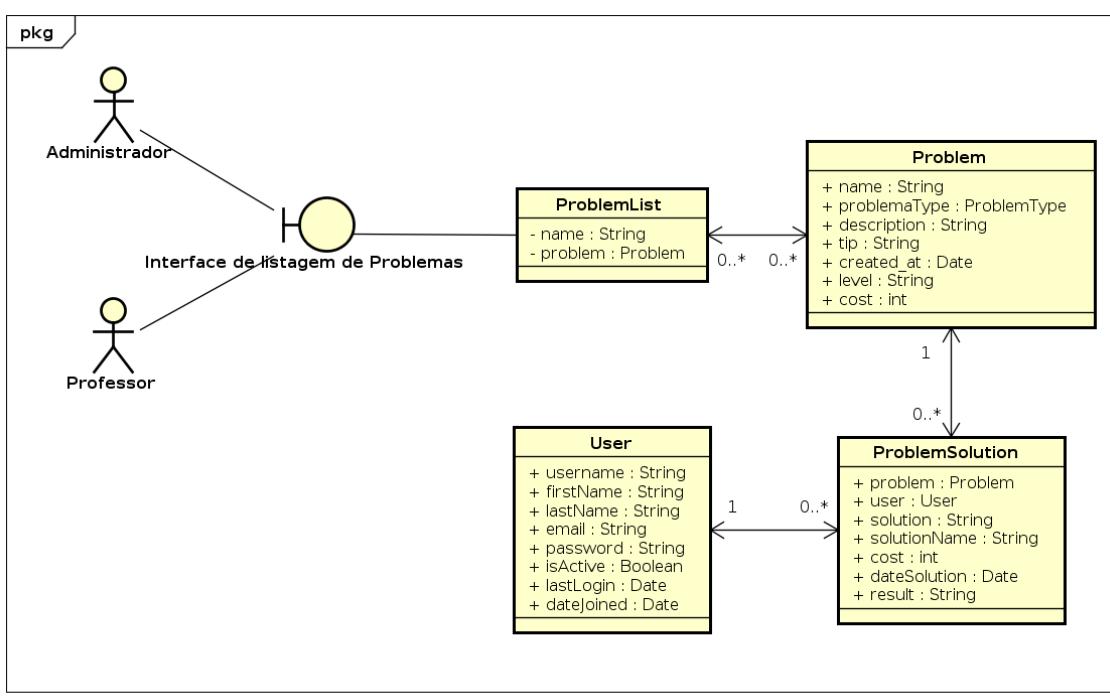


Fonte: (AUTOR, 2018)

4.5.4 Problemas

A Figura 4.39 demonstra as associações das classes de domínio na interface gráfica da Figura 4.8.

Figura 4.39: Diagrama de Robustez de Problemas

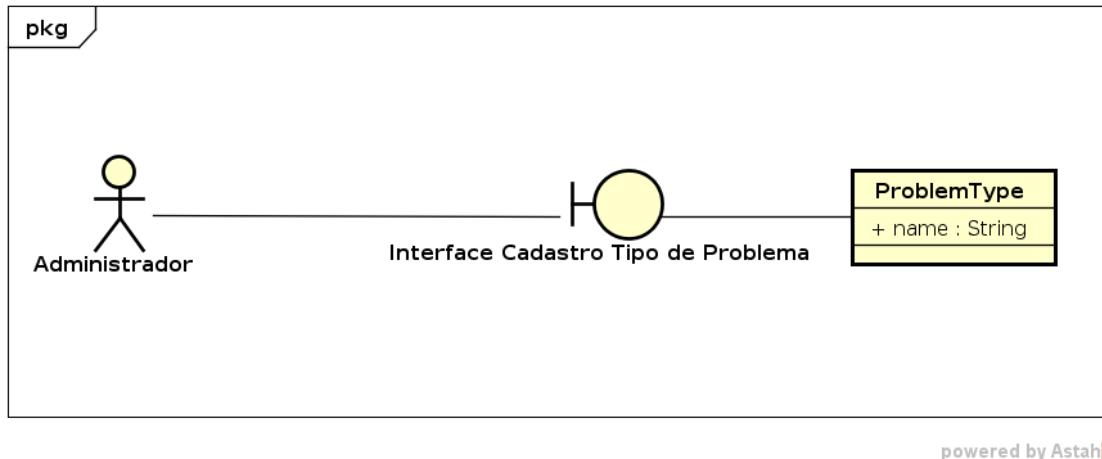


Fonte: (AUTOR, 2018)

4.5.5 Tipo de Problema

A Figura 4.40 demonstra as associações das classes de domínio na interface gráfica da Figura 4.9.

Figura 4.40: Diagrama de Robustez de Tipo de Problema

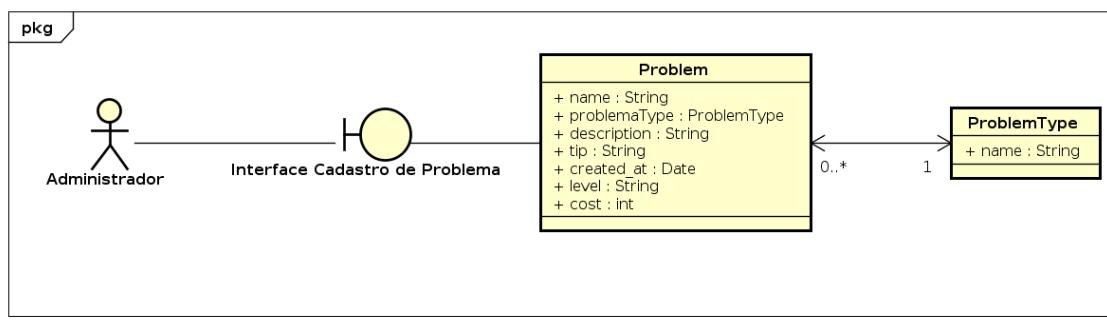


Fonte: (AUTOR, 2018)

4.5.6 Cadastro e Edição de Problema

A Figura 4.41 demonstra as associações das classes de domínio na interface gráfica da Figura 4.10.

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

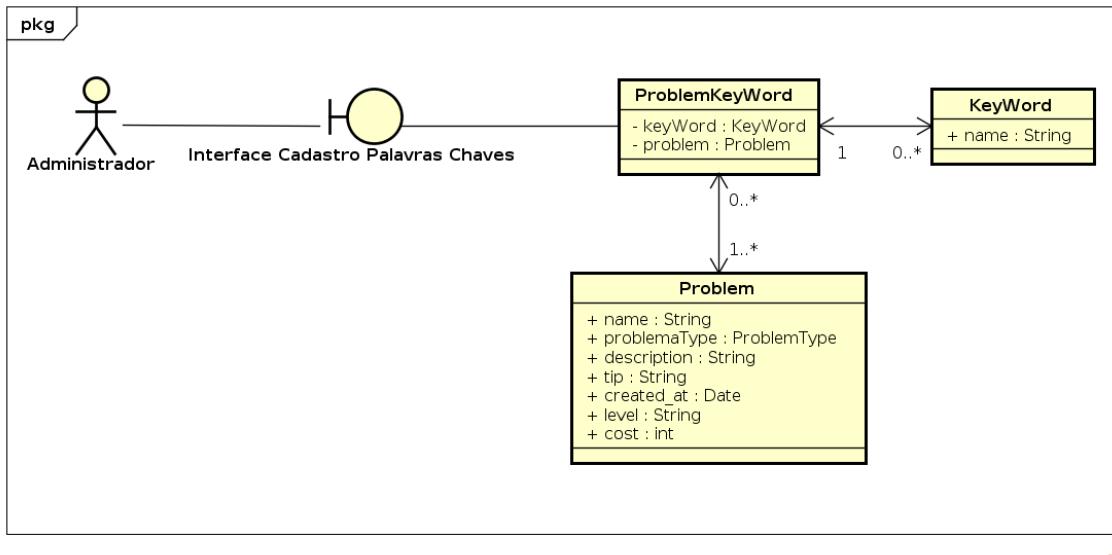


Fonte: (AUTOR, 2018)

4.5.7 Palavras-chave

A Figura 4.42 demonstra as associações das classes de domínio na interface gráfica da Figura 4.11.

Figura 4.42: Diagrama de Robustez de Palavras-chave



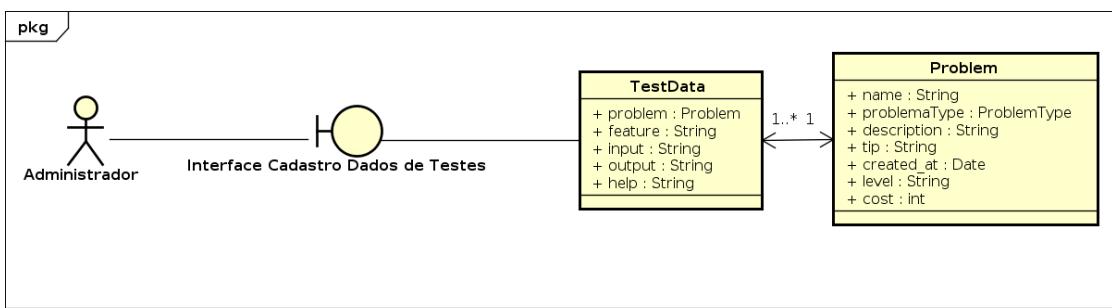
powered by Astah

Fonte: (AUTOR, 2018)

4.5.8 Dados de Testes

A Figura 4.43 demonstra as associações das classes de domínio na interface gráfica da Figura 4.12.

Figura 4.43: Diagrama de Robustez de Dados de Testes



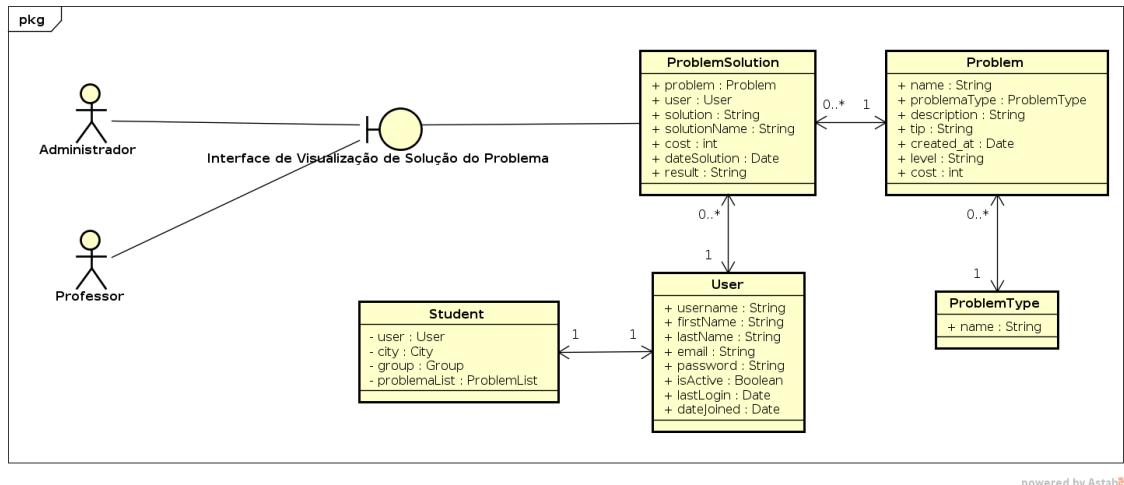
powered by Astah

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 da Figura 4.13.

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

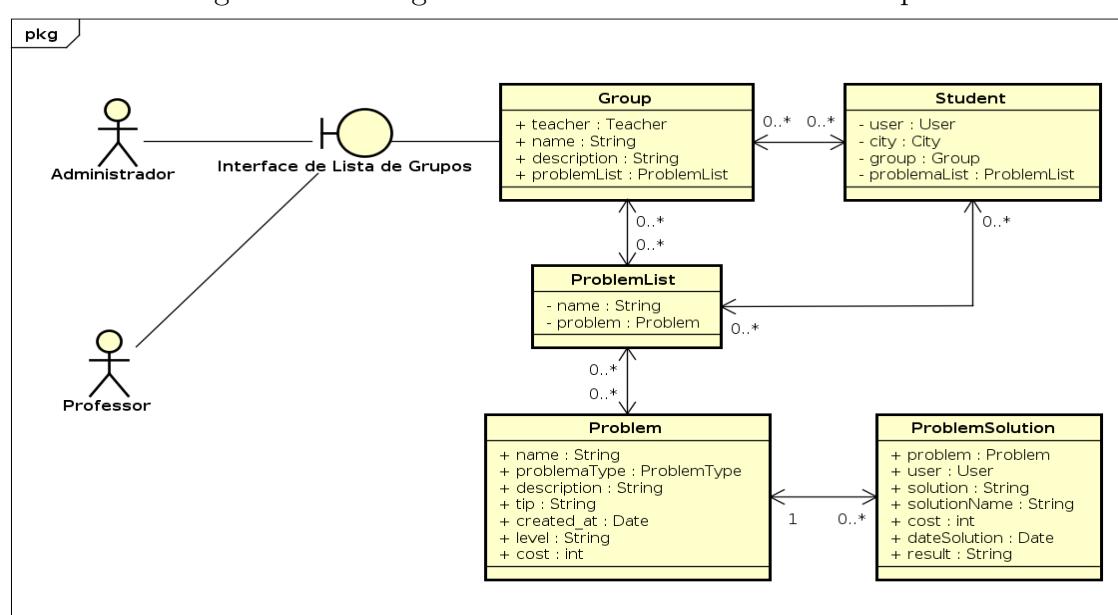


Fonte: (AUTOR, 2018)

4.5.10 Lista de Grupos

A Figura 4.45 demonstra as associações das classes de domínio na interface gráfica da Figura 4.14.

Figura 4.45: Diagrama de Robustez de Lista de Grupos

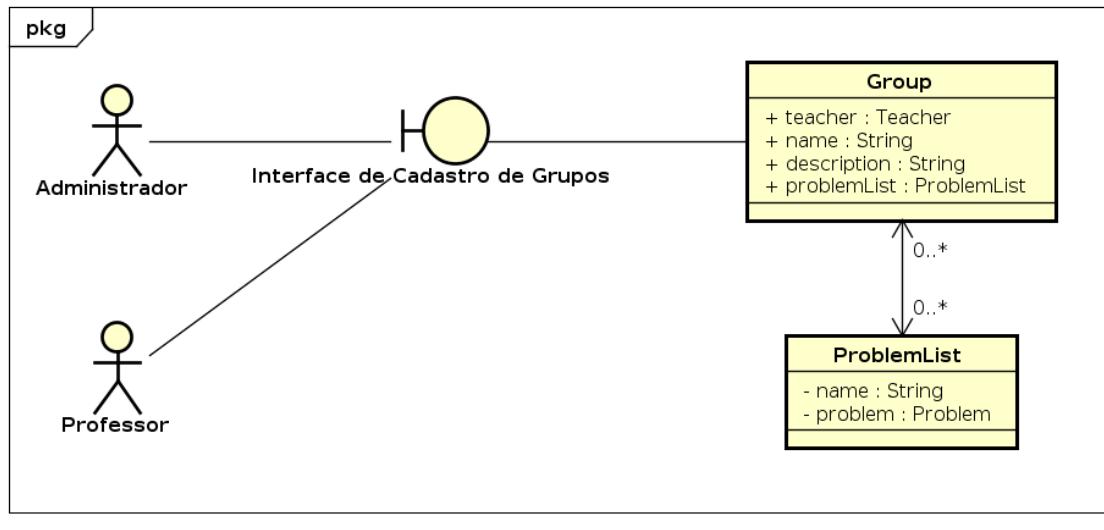


Fonte: (AUTOR, 2018)

4.5.11 Cadastro e Edição de Grupo

A Figura 4.46 demonstra as associações das classes de domínio na interface gráfica da Figura 4.15.

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



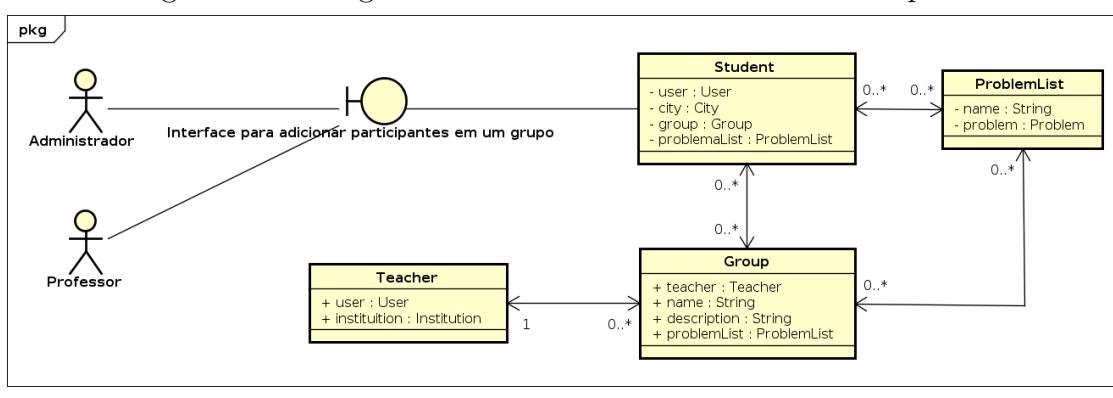
powered by Astah

Fonte: (AUTOR, 2018)

4.5.12 Adicionar Participantes

A Figura 4.47 demonstra as associações das classes de domínio na interface gráfica da Figura 4.16.

Figura 4.47: Diagrama de Robustez de Adicionar Participantes



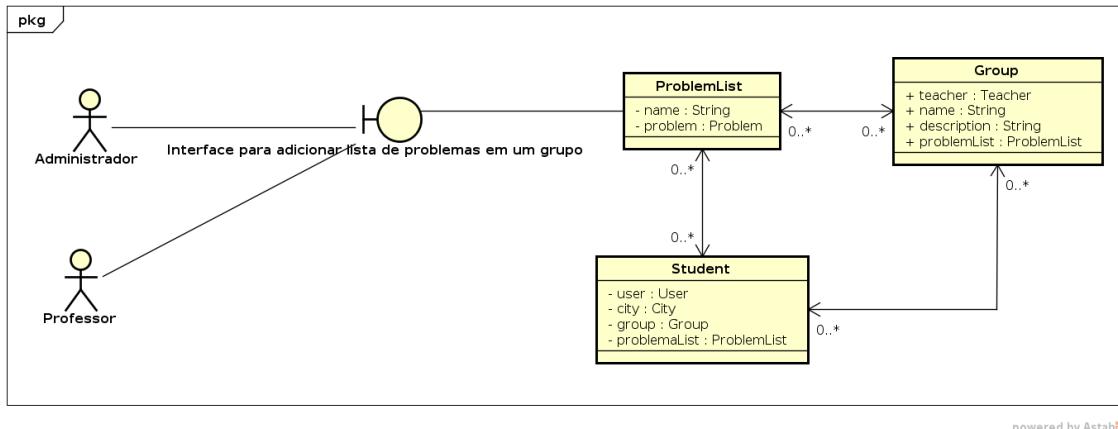
powered by Astah

Fonte: (AUTOR, 2018)

4.5.13 Adicionar Lista de Problemas

A Figura 4.48 demonstra as associações das classes de domínio na interface gráfica da Figura 4.17.

Figura 4.48: Diagrama de Robustez de Adicionar Lista de Problemas



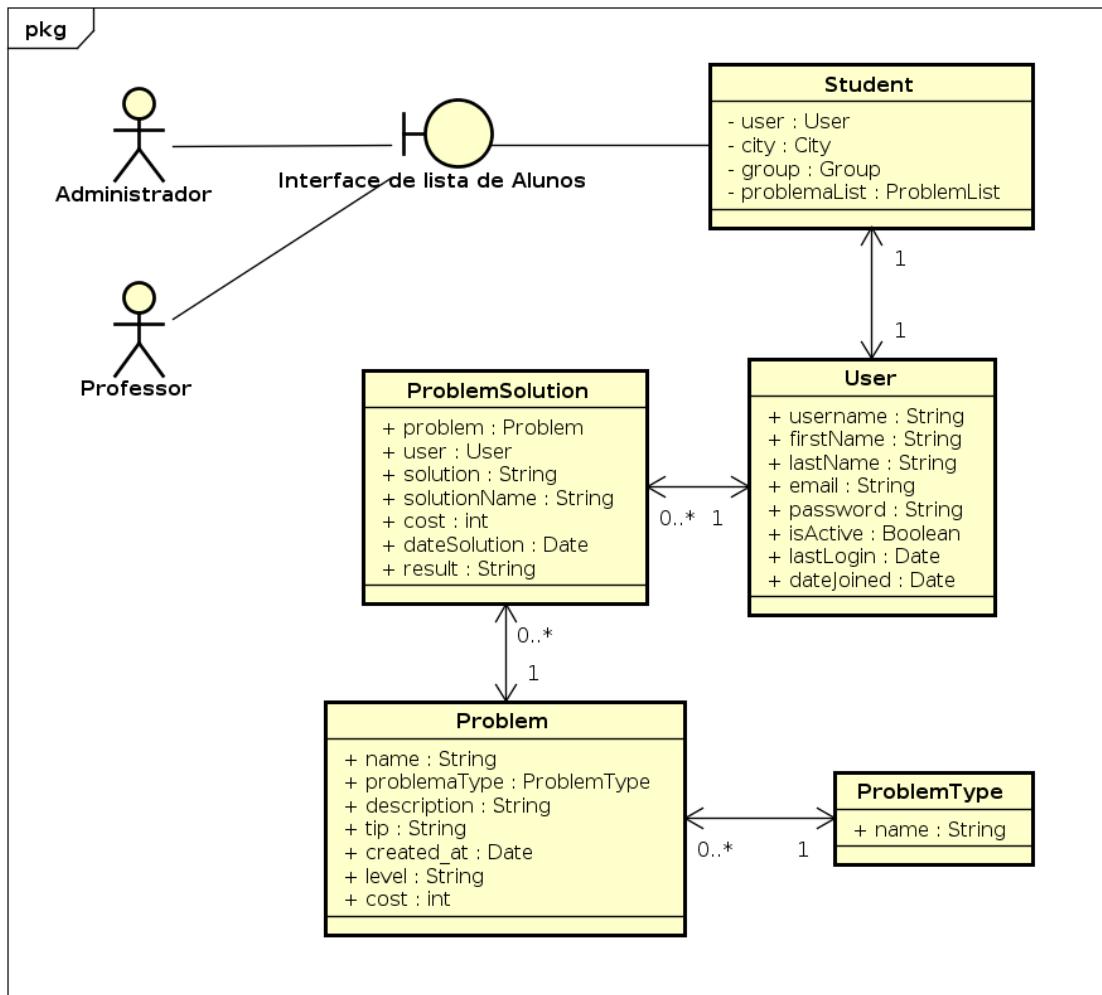
powered by Astah

Fonte: (AUTOR, 2018)

4.5.14 Lista de Alunos

A Figura 4.49 demonstra as associações das classes de domínio na interface gráfica da Figura 4.18.

Figura 4.49: Diagrama de Robustez de Lista de Alunos



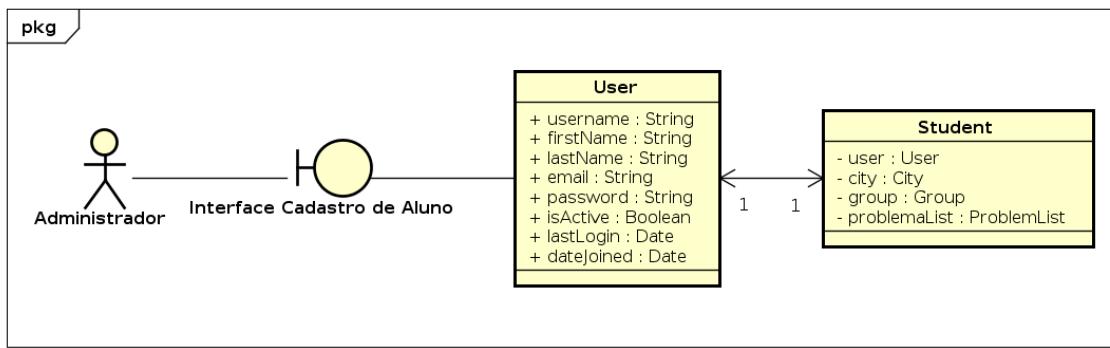
powered by Astah

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 da Figura 4.19.

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



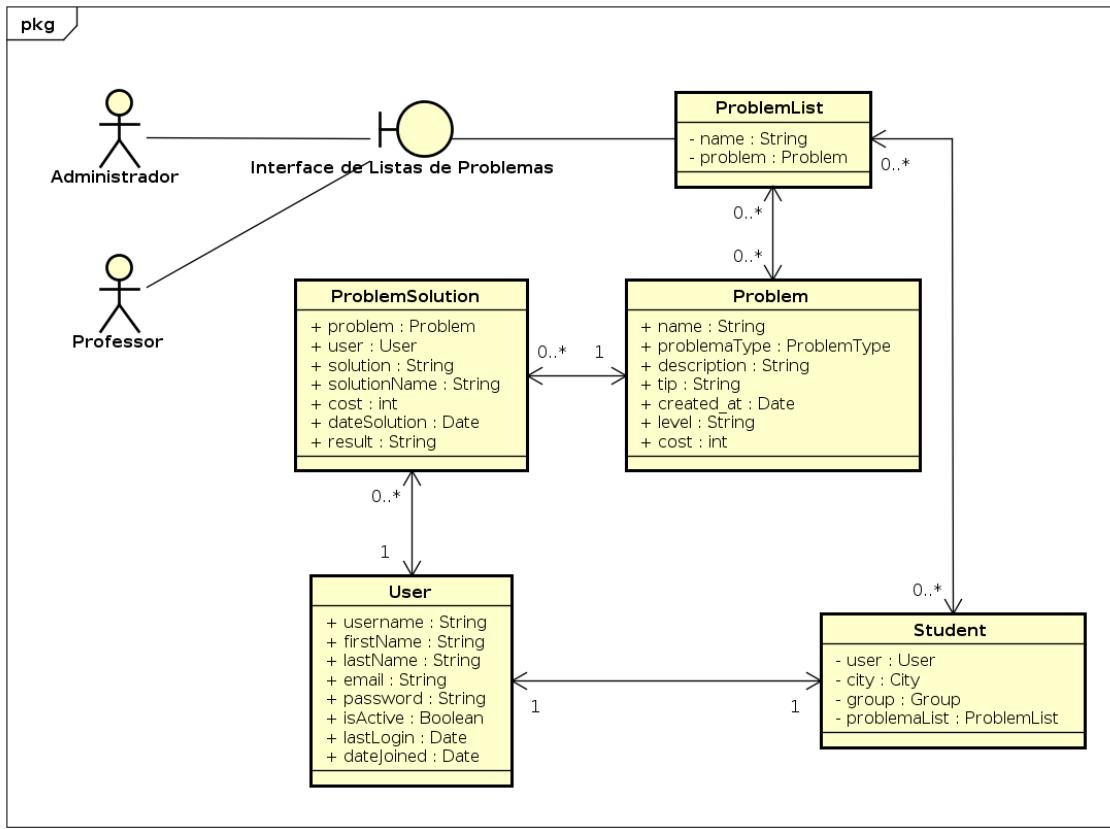
powered by Astah

Fonte: (AUTOR, 2018)

4.5.16 Lista de Problemas

A Figura 4.51 demonstra as associações das classes de domínio na interface gráfica da Figura 4.20.

Figura 4.51: Diagrama de Robustez de Lista de Problemas



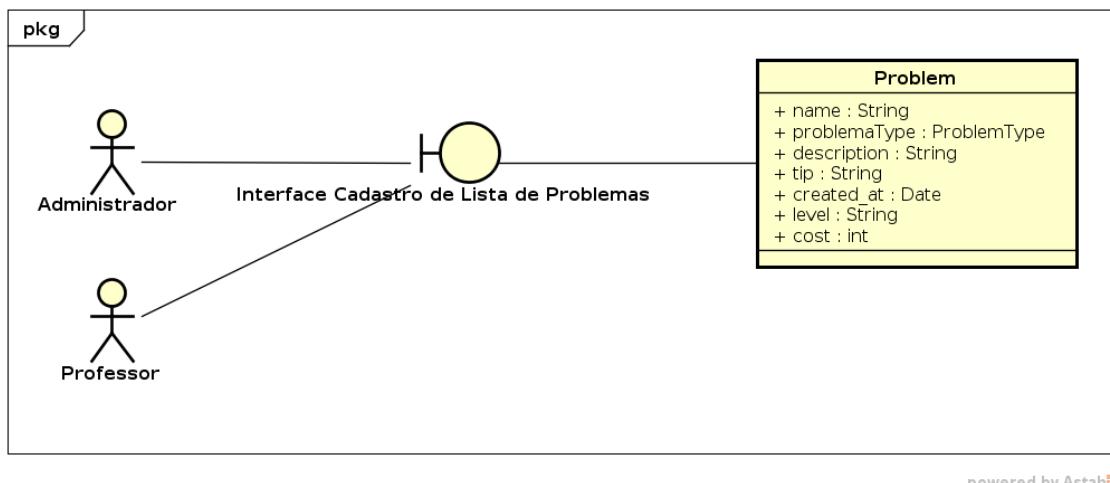
powered by Astah

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 da Figura 4.21.

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



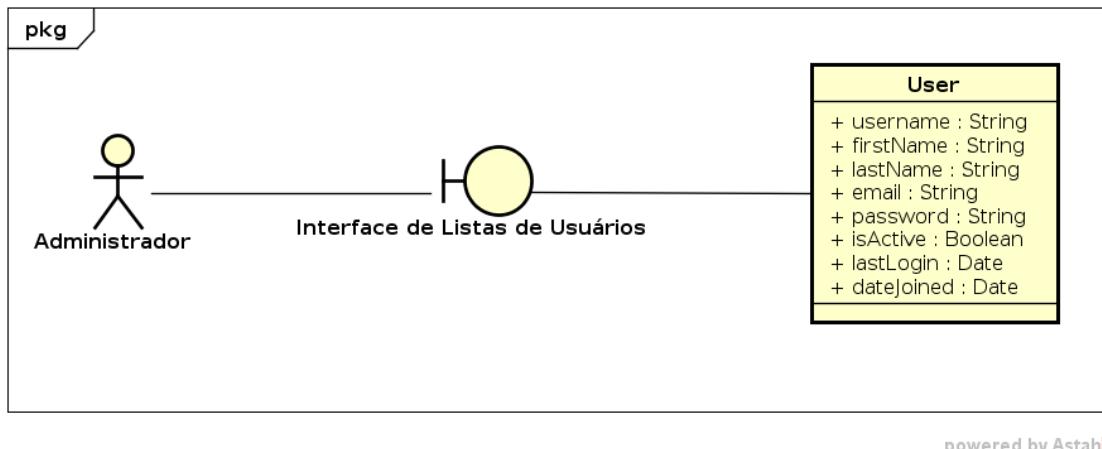
powered by Astah

Fonte: (AUTOR, 2018)

4.5.18 Lista de Usuários

A Figura 4.53 demonstra as associações das classes de domínio na interface gráfica da Figura 4.22.

Figura 4.53: Diagrama de Robustez de Lista de Usuários

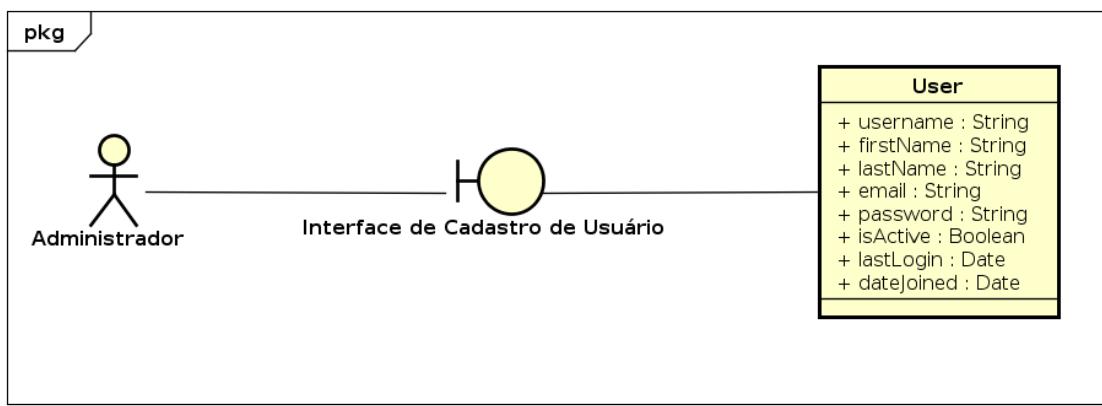


Fonte: (AUTOR, 2018)

4.5.19 Cadastro e Edição de Usuários

A Figura 4.54 demonstra as associações das classes de domínio na interface gráfica da Figura 4.23.

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

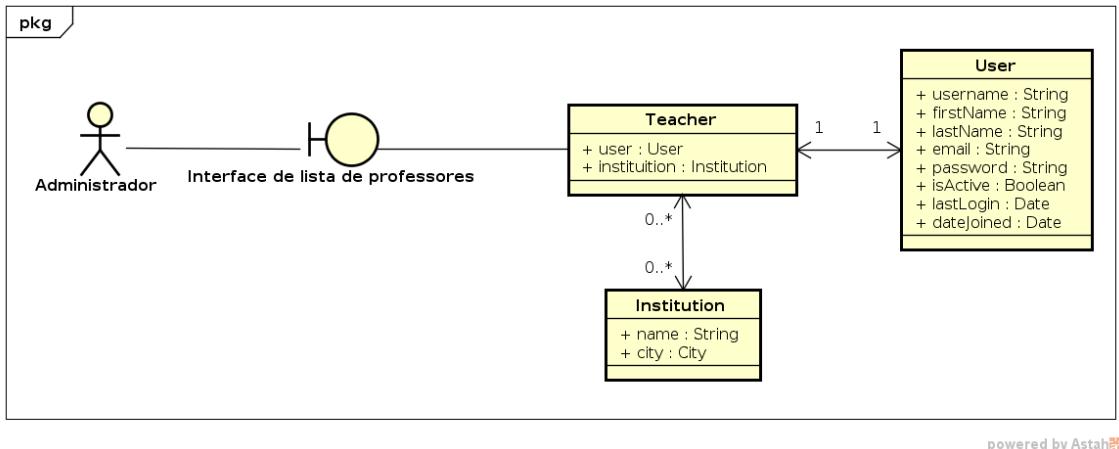


Fonte: (AUTOR, 2018)

4.5.20 Lista de Professores

A Figura 4.55 demonstra as associações das classes de domínio na interface gráfica da Figura 4.24.

Figura 4.55: Diagrama de Robustez de Lista de Professores



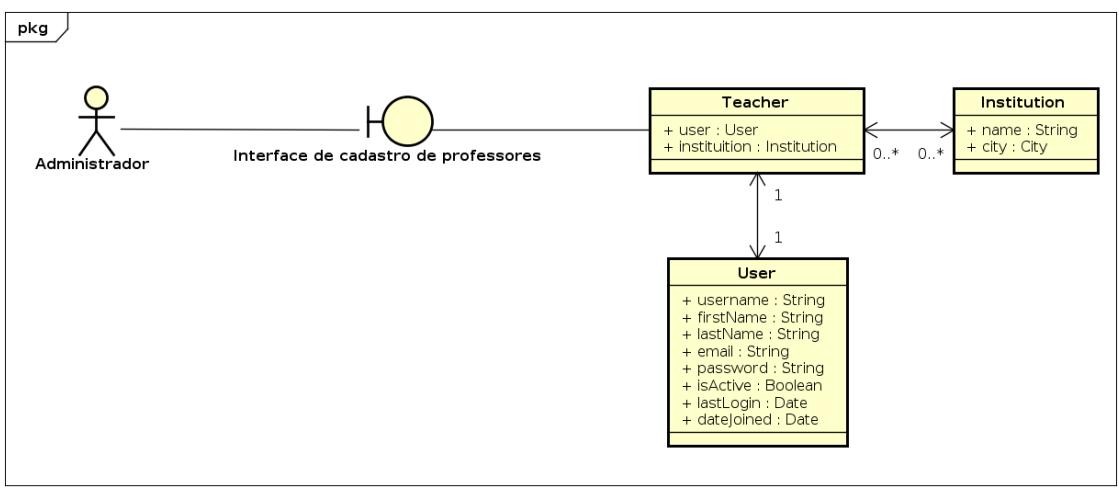
powered by Astah

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 da Figura 4.25.

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



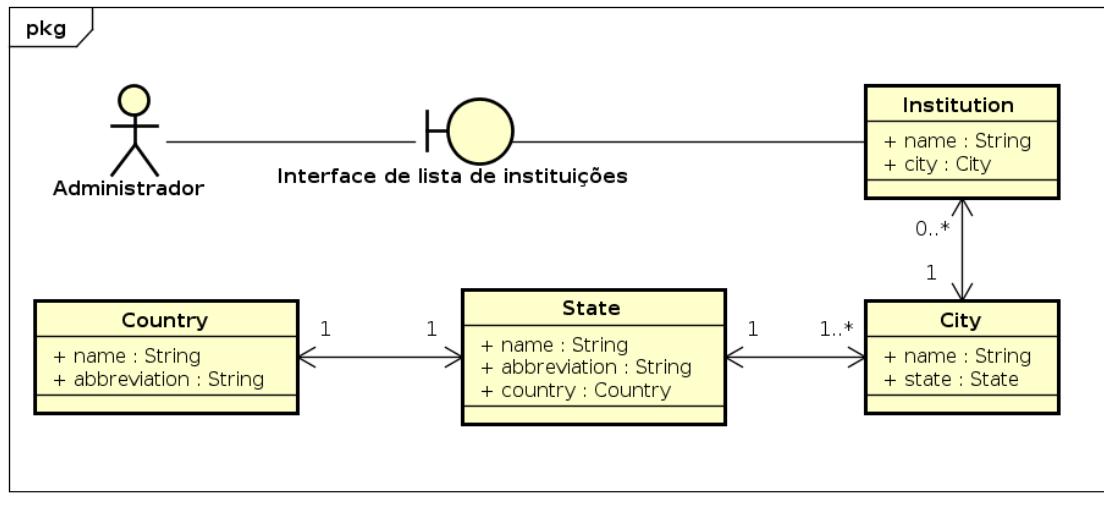
powered by Astah

Fonte: (AUTOR, 2018)

4.5.22 Lista de Instituições

A Figura 4.57 demonstra as associações das classes de domínio na interface gráfica da Figura 4.26.

Figura 4.57: Diagrama de Robustez de Lista de Instituições



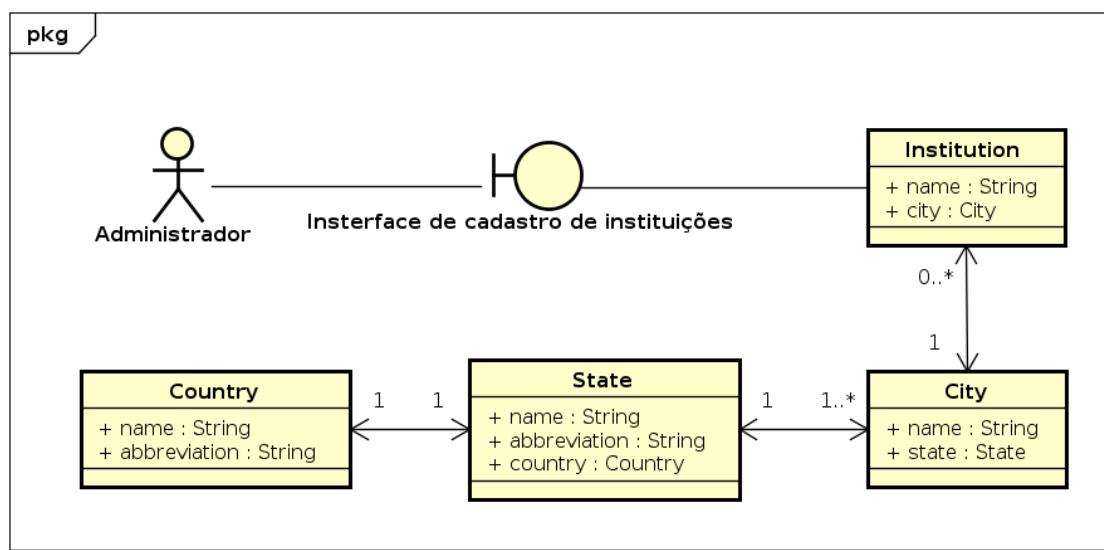
powered by Astah

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 da Figura 4.27.

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



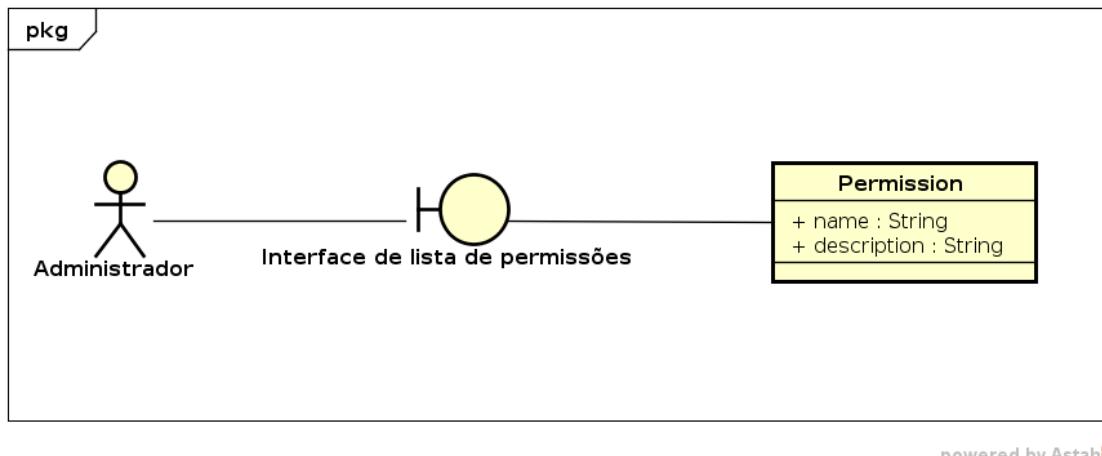
powered by Astah

Fonte: (AUTOR, 2018)

4.5.24 Lista de Permissões

A Figura 4.59 demonstra as associações das classes de domínio na interface gráfica da Figura 4.28.

Figura 4.59: Diagrama de Robustez de Lista de Permissões

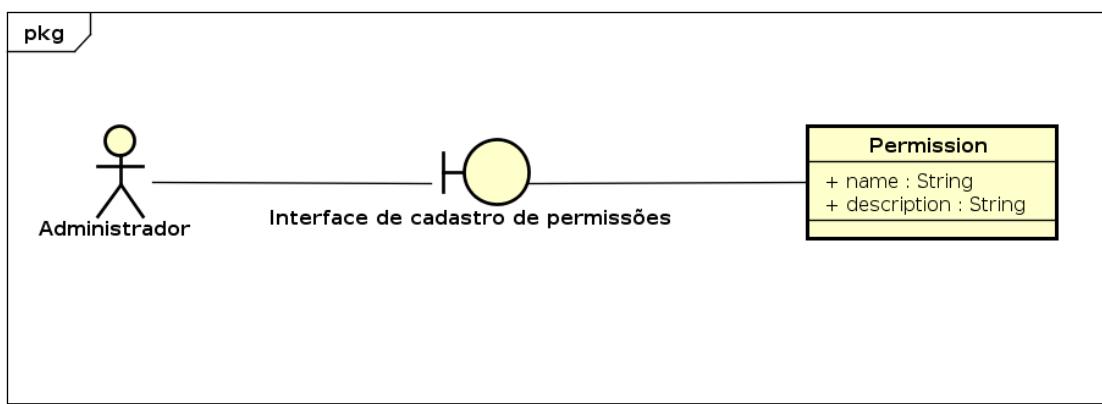


Fonte: (AUTOR, 2018)

4.5.25 Cadastro e Edição de Permissões

A Figura 4.60 demonstra as associações das classes de domínio na interface gráfica da Figura 4.29.

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

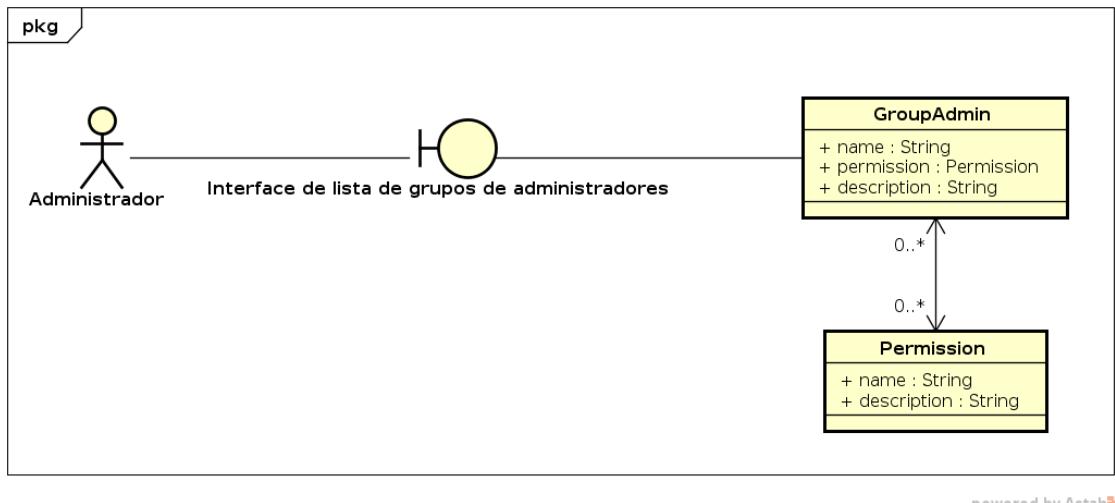


Fonte: (AUTOR, 2018)

4.5.26 Lista de Grupos de Administradores

A Figura 4.61 demonstra as associações das classes de domínio na interface gráfica da Figura 4.30.

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



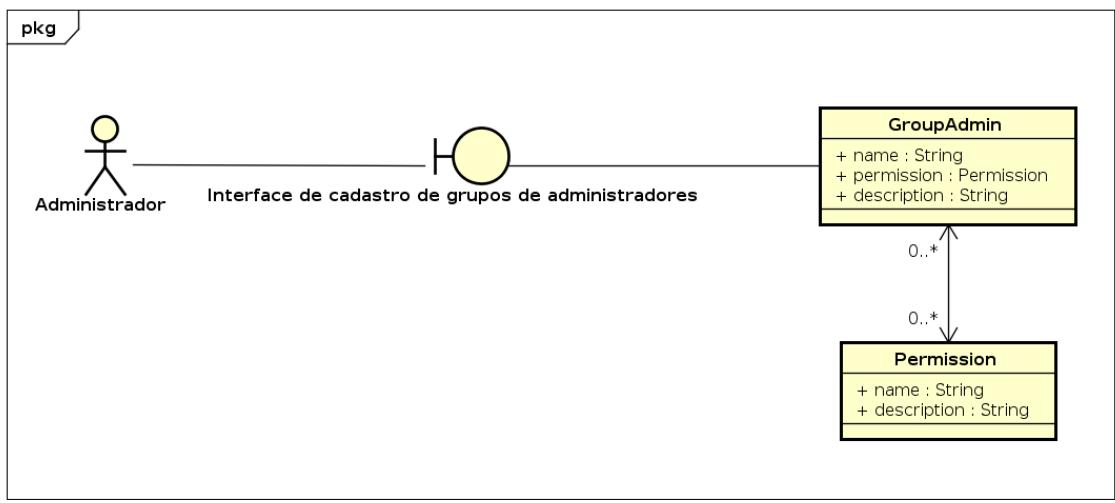
powered by Astah

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 da Figura 4.31.

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



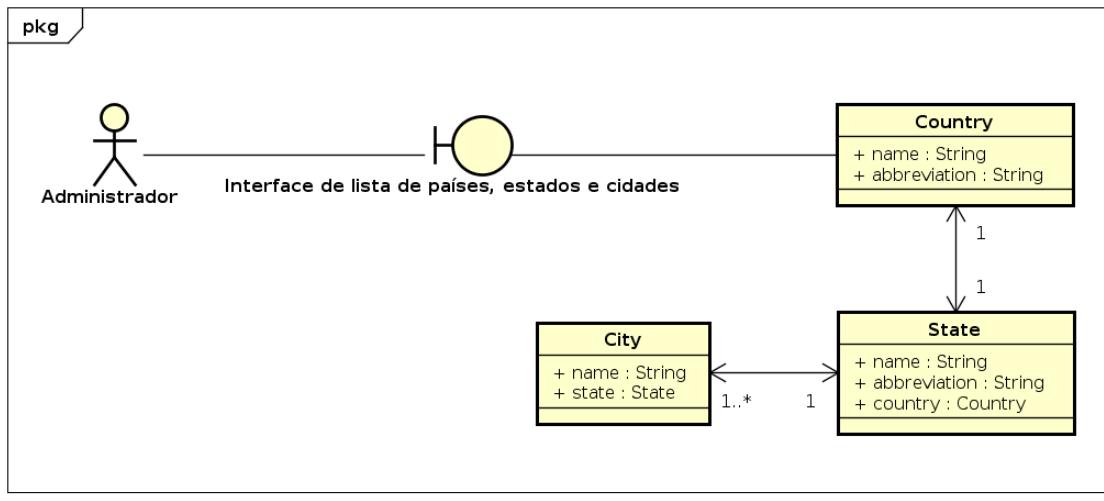
powered by Astah

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 da Figura 4.32.

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



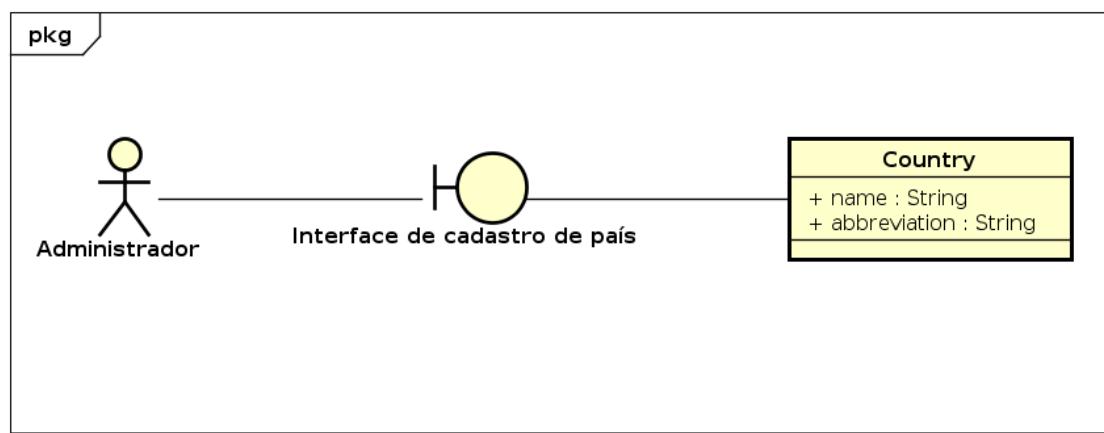
powered by Astah

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 da Figura 4.33.

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



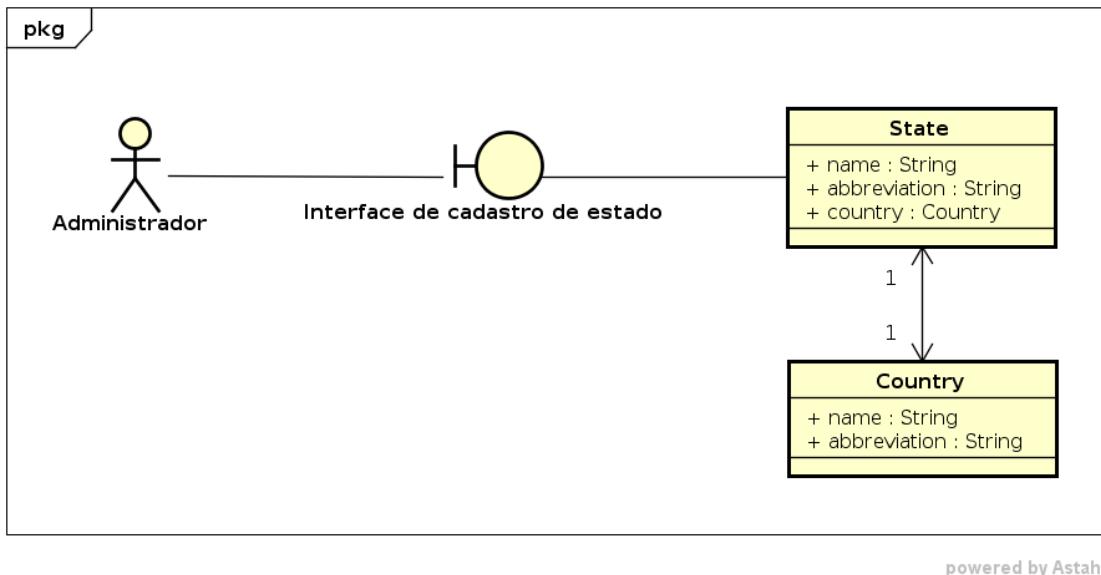
powered by Astah

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 da Figura 4.34.

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

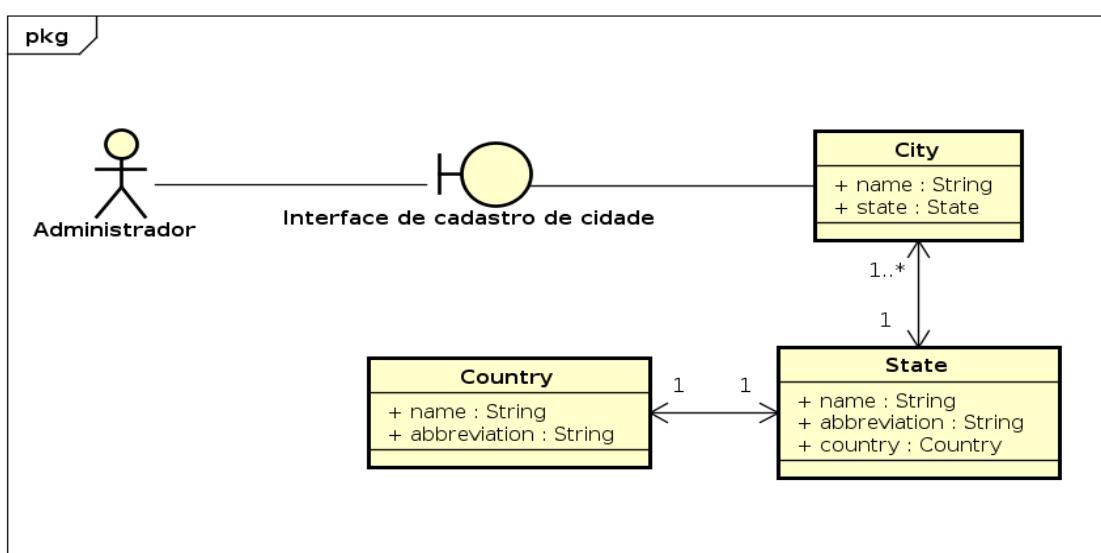


Fonte: (AUTOR, 2018)

4.5.31 Cadastro e Edição de Cidade

A Figura 4.66 demonstra as associações das classes de domínio na interface gráfica da Figura 4.35.

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

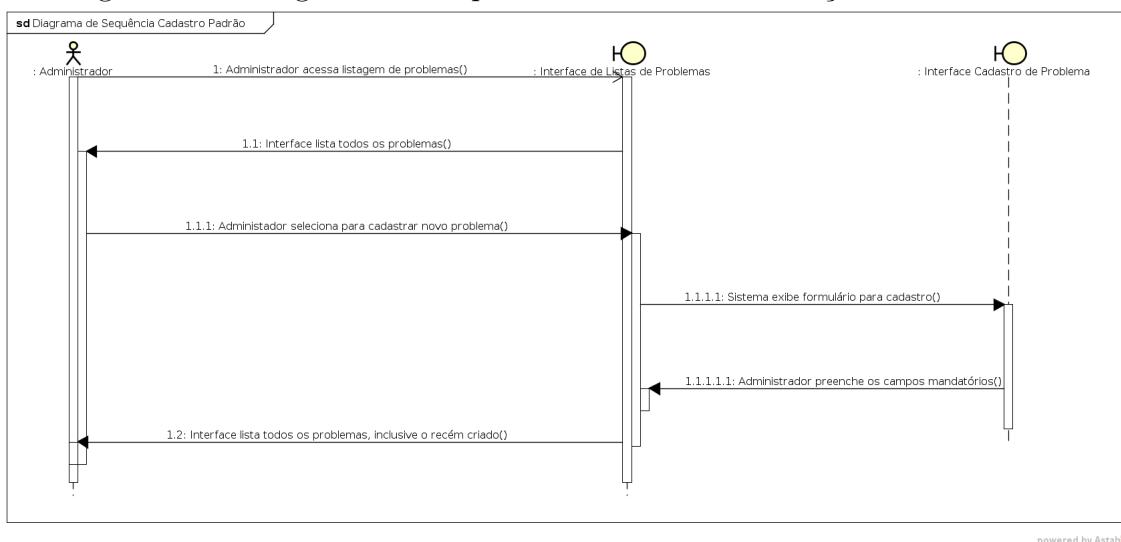


Fonte: (AUTOR, 2018)

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



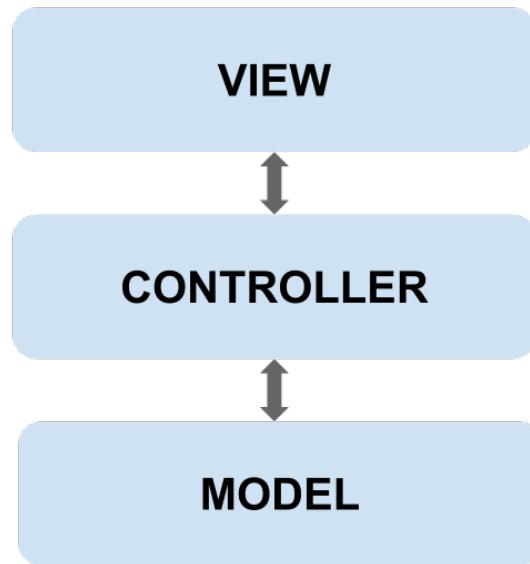
Fonte: (AUTOR, 2018)

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.

Figura 4.68: Arquitetura Lógica do Portal de Algoritmos



Fonte: (AUTOR, 2018)

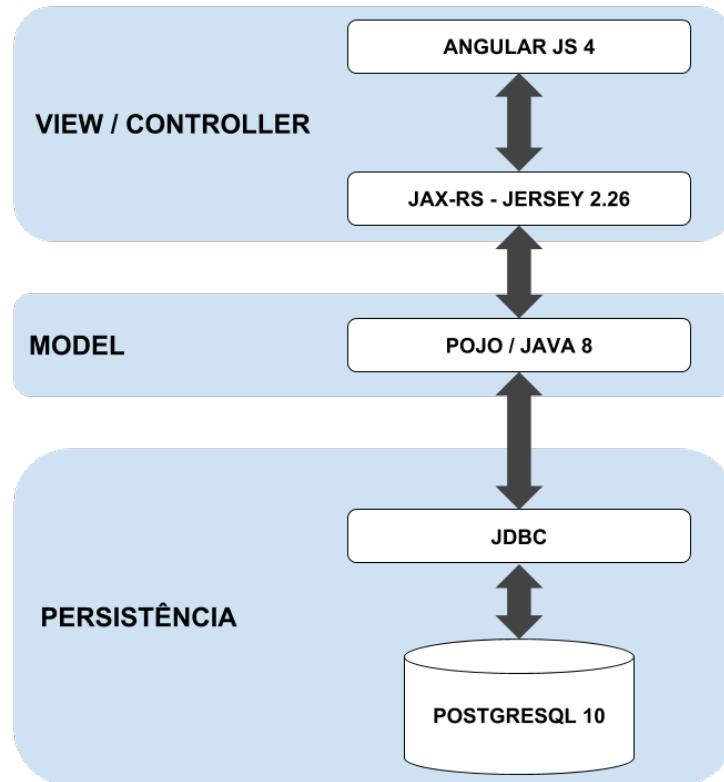
Na camada de *Model* é feita a manipulação de dados, leitura, escrita de dados.

Na camada de *Controller* é responsável por receber todas as requisições do usuário, nessa camada possui métodos de ações.

Na camada de *View* temos a interação com o usuário, essa camada faz apenas a exibição dos dados.

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

Figura 4.69: Arquitetura do Software do Portal de Algoritmos



Fonte: (AUTOR, 2018)

5 SEGURANÇA

O Portal de Algoritmos antes da atualização apresenta algumas falhas de segurança, abaixo são citadas apenas algumas como sendo graves. Além das falhas citadas é proposto uma solução utilizando *HTTPS* (*Hyper Text Transfer Protocol Secure*).

5.1 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, 2015)

5.2 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 dadosm 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, 2016)

5.3 HTTPS

O *HTTPS* 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)

Let's Encrypt possui os seguintes principios: ser gratuito, automático, seguro, transparente, aberto e cooperativo.

6 CONSIDERAÇÕES FINAIS

Problema e solução

O novo *software* foi desenvolvido utilizando tecnologias de ponta, como *Java* na última versão estável, *AngulaJS* na versão mais recente estável e um *framework* de *CSS* (*Cascading Style Sheets*) fácil de utilizar.

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

A interface gráfica ficou mais amigável e intuitiva para o usuário, utilizando padrões de telas, botões, listagem e cadastros em geral.

Com um banco de dados atual o *software* está mais confiável no armazenamento das informações cadastrais, com essa atualização estamos garantindo integridade e segurança nas informações armazenadas nele.

Com a implementação do novo gerenciador do portal de algoritmos concluímos que foi construido um *software* robusto, organizado, com alta qualidade de código e de fácil entendimento sobre o que ele foi proposto.

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: 2 de Novembro de 2015.

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.

ENCRYPT, 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>. Acesso em: 2 de Dezembro de 2015.

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: 2 de Novembro de 2015.

JAVA. **Java.** <Disponível em: <https://docs.oracle.com/javase/7/tutorial/>>. Acesso em: 25 de Novembro de 2015.

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>. Acesso em: 3 de Junho de 2018.

OWASP. **SQL Injection.** <Disponível em: 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: 2 de Novembro de 2015.

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: 2 de Dezembro de 2015.

APÊNDICE A - DOCUMENTAÇÃO

API de Países, endpoint base *countries*

```
1   GET /countries
```

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

```
1   GET /countries/:id
```

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

```
1   POST /countries
```

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

```
1   PUT /countries
```

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

```
1   DELETE /countries/:id
```

```
1      GET /states
```

```
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 ]
```

```
1      GET /states/:id
```

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

```
1      POST /states
```

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

```
1      PUT /states
```

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

```
1      DELETE /states/:id
```

```
1     GET /cities
```

```
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 ]
```

```
1     GET /cities/:id
```

```
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 }
```

```
1     POST /cities
```

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

```
1     PUT /cities
```

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

```
5         "id": "Integer",
6     }
7 }
```

```
1     DELETE /cities/:id
```

```
1     GET /institutions
```

```
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 ]
```

```
1     GET /institutions/:id
```

```
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 }
```

```
1     POST /institutions
```

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

```
6      }
7 }
```

```
1 PUT /institutions
```

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

```
1 DELETE /institutions/:id
```

```
1     GET /permissions
```

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

```
1     GET /permissions/:id
```

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

```
1     POST /permissions
```

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

```
1     PUT /permissions
```

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

```
1     DELETE /permissions/:id
```

```
1     GET /users
```

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

```
1     GET /users/:id
```

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

```
1     POST /users
```

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

```
1     PUT /users
```

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

```
1     DELETE /users/:id
```

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

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

```
1     POST userpermission/{user-id} # arrumar portaluser para user
```

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

1	/teachers
---	-----------

1	/students
---	-----------

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

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

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

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

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

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

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

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

```
1     GET /groupadmins
```

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

```
1     GET /groupadmins/:id
```

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

```
1     POST /groupadmins
```

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

```
1     PUT /groupadmins
```

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

```
1     DELETE /groupadmins/:id
```

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

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

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

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

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

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

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

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

```
1     GET /groups
```

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

```
1     GET /groups/:id
```

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

```
1     POST /groups
```

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

```
1     PUT /groups
```

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

```
1     DELETE /groups/:id
```

```
1     GET /groupproblemlist/filter-by-group-id/{group-id}
```

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

```
1     POST /groupproblemlist/{group-id}
```

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

```
1     GET /problemtypes
```

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

```
1     GET /problemtypes/:id
```

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

```
1     POST /problemtypes
```

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

```
1     PUT /problemtypes
```

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

```
1     DELETE /problemtypes/:id
```

```
1     GET /problems
```

```
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 ]
```

```
1     GET /problems/:id
```

```
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 }
```

```
1     POST /problems
```

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

```
1     PUT /problems
```

```
1 {
2   "id": "Integer",
```

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

```
1     DELETE /problems/:id
```

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

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

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

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

```
1     GET /keywords
```

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

```
1     GET /keywords/:id
```

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

```
1     POST /keywords
```

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

```
1     PUT /keywords
```

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

```
1     DELETE /keywords/:id
```

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

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

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

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

```
1     GET /listactivities
```

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

```
1     GET /listactivities/:id
```

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

```
1     POST /listactivities
```

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

```
1     PUT /listactivities
```

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

```
1     DELETE /listactivities/:id
```

```
1     GET /problemlists
```

```
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 ]
```

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

```
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 ]
```

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

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

```
1     GET /problemsolutions
```

```
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 ]
```

```
1     GET /problemsolutions/:id
```

```
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 }
```

```
1     POST /problemsolutions
```

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

```
1 PUT /problemsolutions
```

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

```
1 DELETE /problemsolutions/:id
```