

UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ADRIANO MARGARIN

**Portal de Algoritmos da Universidade
de Caxias do Sul
Evolução da ferramenta de
gerenciamento do portal**

Alexandre Erasmo Krohn Nascimento
Orientador

Ricardo de Vargas Dornele
Coorientador

Caxias do Sul, Novembro de 2015

Portal de Algoritmos da Universidade de Caxias do Sul

Evolução da ferramenta de gerenciamento do portal

por

Adriano Margarin

Trabalho de Conclusão de Curso submetido ao curso de Bacharelado em Sistemas de Informação do Centro de Ciências Exatas e 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
4 de Dezembro de 2015

Daniel Luís Notari
Coordenador

"Não é preciso fazer coisas extraordinárias para obter resultados extraordinários."

WARREN BUFFET

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 nesse caminho trilhado até aqui.

A todos vocês, minha sincera gratidão.

Adriano Margarin

SUMÁRIO

LISTA DE ACRÔNIMOS	7
LISTA DE FIGURAS	8
LISTA DE TABELAS	11
LISTA DE TRECHOS DE CÓDIGO	12
RESUMO	13
ABSTRACT	14
1 INTRODUÇÃO	15
2 EVOLUÇÃO DE SOFTWARE	18
2.1 Engenharia de Software	19
2.1.1 Reengenharia de Software	20
2.1.2 Engenharia Reversa	21
2.2 Processo de Software	21
2.2.1 Engenharia de requisitos	22
2.3 Casos de uso	22
2.4 Modelos de Domínio	23
2.5 Arquitetura	24
2.6 Usabilidade	24
2.7 Tecnologia	25
2.7.1 JAVA EE	25
2.7.2 REST	25
2.7.3 AngularJS	25
2.7.4 Servidor	26

3 REENGENHARIA DE SOFTWARE	27
3.1 Diagrama de Classe de Domínio	27
3.1.1 User	28
3.1.2 Groups	28
3.1.3 Permission	28
3.1.4 ContentType	29
3.1.5 Aluno	29
3.1.6 Professor	29
3.1.7 Turma	29
3.1.8 Problema	29
3.1.9 Pchave	29
3.1.10 PchaveProb	29
3.1.11 EProblema	29
3.1.12 SProblema	30
3.2 Interfaces Gráficas	30
3.2.1 Cadastro de Aluno	30
3.2.2 Criação de Solução de Problemas	30
3.2.3 Gerenciamento de Alunos	31
3.2.4 Gerenciamento de Problemas	32
3.2.5 Edição de Palavras Chaves	33
3.2.6 Solução de Problemas	34
4 PROPOSTA DE SOLUÇÃO	36
4.1 Diagrama de Classe de Domínio	36
4.2 Requisitos de projeto	37
4.3 Casos de Uso	38
4.3.1 Descrição dos Casos de Uso	40
4.4 Protótipos de Interfaces Gráficas	58
4.4.1 Cadastro de Alunos	58
4.4.2 Autenticação	60
4.4.3 Recuperação de Senha	60
4.4.4 Boas Vindas do Aluno	61
4.4.5 Boas Vindas do Administrador e Professor	62
4.4.6 Lista de Problemas	63
4.4.7 Tipo de Problema	65
4.4.8 Cadastro e Edição Problema	65
4.4.9 Palavras Chaves	66
4.4.10 Dados de Testes	67

4.4.11	Visualização de Solução a partir da listagem de Problemas	68
4.4.12	Lista de Grupos	69
4.4.13	Cadastro e Edição de Grupo	70
4.4.14	Adicionar Participantes	71
4.4.15	Adicionar Lista de Problemas	72
4.5	Diagramas de Robustez	89
5	CONSIDERACOES PARCIAIS	101
	REFERÊNCIAS	102

LISTA DE ACRÔNIMOS

API	<i>Application Programming Interface</i>
UCS	Universidade de Caxias do Sul
NPAPI	<i>Netscape Plugin Application Programming Interface</i>
UML	<i>Unified Modeling Language</i>
ORM	<i>Object-relational mapping</i>
SQL	<i>Structured Query Language</i>
REST	<i>Representational State Transfer</i>
WEB	Rede de Internet
ORM	<i>Object-relational mapping</i>
IHC	Interação humano-computador
HTML	<i>HyperText Markup Language</i>
MVC	<i>Model-View-Controller</i>
JAVA EE	<i>Java Enterprise Edition</i>

LISTA DE FIGURAS

Figura 2.1: Camadas da Engenharia de Software - Fonte (PRESSMAN, 2011)	20
Figura 2.2: Processo de Reengenharia - Fonte (SOMMERVILLE, 2011)	20
Figura 2.3: Casos de uso - Fonte (SOMMERVILLE, 2011)	23
Figura 2.4: Interação entre AngularJS e Arquitetura - Fonte (BRANAS, 2014)	26
Figura 3.1: Diagrama de Domínio do Portal de Algoritmos Atual	28
Figura 3.2: Cadastro de Aluno	30
Figura 3.3: Criação de Solução de Problemas	31
Figura 3.4: Gerência de Alunos	32
Figura 3.5: Gerência de Problemas	33
Figura 3.6: Edição de palavras chaves	34
Figura 3.7: Visualizando solução de um aluno	35
Figura 4.1: Diagrama de Domínio do Portal de Algoritmos Novo	37
Figura 4.2: Casos de Uso que envolvem o ator Administrador	39
Figura 4.3: Casos de Uso que envolvem o ator Professor	40
Figura 4.4: Casos de Uso que envolvem o ator Aluno	40
Figura 4.5: Caso de Uso Manter Usuários	41
Figura 4.6: Caso de Uso Manter Alunos	42
Figura 4.7: Caso de Uso Manter Professores	43
Figura 4.8: Caso de Uso Manter Administradores	44
Figura 4.9: Caso de Uso Manter Tipo de Problema	45
Figura 4.10: Caso de Uso Manter Problemas	46
Figura 4.11: Caso de Uso Manter Palavras Chaves	47
Figura 4.12: Caso de Uso Manter Entrada e Saídas	48
Figura 4.13: Caso de Uso Manter Soluções de Problemas	49
Figura 4.14: Caso de Uso Manter Listas de Problemas	50
Figura 4.15: Caso de Uso Manter Instituições	51
Figura 4.16: Caso de Uso Manter Grupos	52
Figura 4.17: Caso de Uso Manter Permissões	53

Figura 4.18: Caso de Uso Manter Grupos de Administradores	54
Figura 4.19: Caso de Uso Manter Países	55
Figura 4.20: Caso de Uso Manter Estados	56
Figura 4.21: Caso de Uso Manter Cidades	57
Figura 4.22: Caso de Uso Chat Online	57
Figura 4.23: Caso de Uso Aluno Manter suas Informações Pessoais	58
Figura 4.24: Protótipo de Inteface Gráfica de Cadastro de Aluno	59
Figura 4.25: Protótipo de Inteface Gráfica de Autenticação	60
Figura 4.26: Protótipo de Inteface Gráfica de Recuperação de Senha	61
Figura 4.27: Protótipo de Inteface Gráfica de Boas Vindas do Aluno	62
Figura 4.28: Protótipo de Inteface Gráfica de Boas Vindas do Administrador e Professor	63
Figura 4.29: Protótipo de Inteface Gráfica de Problemas	64
Figura 4.30: Protótipo de Inteface Gráfica de Novo Tipo de Problema	65
Figura 4.31: Protótipo de Inteface Gráfica de Novo Problema	66
Figura 4.32: Protótipo de Inteface Gráfica de Palavras Chaves	67
Figura 4.33: Protótipo de Inteface Gráfica de Dados de Testes	68
Figura 4.34: Protótipo de Inteface Gráfica de Visualização de Solução do Pro- blema	69
Figura 4.35: Protótipo de Inteface Gráfica de Grupos	70
Figura 4.36: Protótipo de Inteface Gráfica de Cadastro de Grupo	71
Figura 4.37: Protótipo de Inteface Gráfica de Cadastro de Participantes no Grupo	72
Figura 4.38: Protótipo de Inteface Gráfica de Cadastro de Lista de Problemas no Grupo	73
Figura 4.39: Mockups	74
Figura 4.40: Mockups	75
Figura 4.41: Mockups	76
Figura 4.42: Mockups	76
Figura 4.43: Mockups	77
Figura 4.44: Mockups	78
Figura 4.45: Mockups	78
Figura 4.46: Mockups	79
Figura 4.47: Mockups	80
Figura 4.48: Mockups	80
Figura 4.49: Mockups	81
Figura 4.50: Mockups	82
Figura 4.51: Mockups	82
Figura 4.52: Mockups	83

Figura 4.53: Mockups	84
Figura 4.54: Mockups	84
Figura 4.55: Mockups	85
Figura 4.56: Mockups	86
Figura 4.57: Mockups	86
Figura 4.58: Mockups	87
Figura 4.59: Mockups	88
Figura 4.60: Mockups	88
Figura 4.61: Robustez	89
Figura 4.62: Robustez	90
Figura 4.63: Robustez	91
Figura 4.64: Robustez	92
Figura 4.65: Robustez	93
Figura 4.66: Robustez	93
Figura 4.67: Robustez	94
Figura 4.68: Robustez	95
Figura 4.69: Robustez	96
Figura 4.70: Robustez	97
Figura 4.71: Robustez	97
Figura 4.72: Robustez	98
Figura 4.73: Robustez	98
Figura 4.74: Robustez	99
Figura 4.75: Robustez	100

LISTA DE TABELAS

Tabela 4.1: Requisitos funcionais	38
Tabela 4.2: Requisitos não-funcionais	38

LISTA DE TRECHOS DE CÓDIGO

RESUMO

COLOCAR RESUMO

Palavras-chave: .

COLOCAR EM INGLÊS O TÍTULO DO TCC

ABSTRACT

COLOCAR RESUMO EM INGLÊS

Keywords:

1 INTRODUÇÃO

O portal de algoritmos, desenvolvido no ano de 2009 pelos professores, Ricardo de 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 Centro de Ciências Exatas e 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 portugolem um editor específico. O aluno submete soluções de problemas a fim de validá-las, também 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 chaves (DORNELES; JUNIOR; ADAMI, 2010).

Na criação de um problema, o administrador deve informar um nome, uma descrição onde deve ser explicado o que deve ser solucionado, dicas e palavras chaves, sendo que as últimas informações não são obrigatórias. Também devem ser 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.

Outra funcionalidade do portal é de poder acompanhar as submissões dos alunos. Selecionando um problema pode-se visualizar os alunos que submeteram uma solução para o mesmo e a partir do aluno é possível ver as soluções dele e em seguida visualizar a solução em si. Essa é uma das formas de acompanhar a evolução do aluno. Outra forma é partindo de um determinado aluno e visualizando todas as soluções que ele já submeteu.

Na administração é possível manter todas as informações dos usuários, problemas, palavras chaves, dados de testes, entre outras informações.

Devido a constante evolução tecnologica, o portal de algoritmo 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, *Python* versão 2.6, *Django* versão 1.2, *Plugins NPAPI* (*Netscape Plugin Application Programming Interface*) (*Java Applet*), esas tecnologias citadas, estão desatualizadas ou foram descontinuadas, No caso da tecnologia *NPAPI* foi totalmente desativada (PYTHON, 2015; DJANGO, 2015; GOOGLE, 2015).

Um dos problemas citados acima são a versão do *Python* e *Django*, que possuem problemas de segurança e não possuem mais suporte, fazendo com que o portal esteja vulnerável há acessos não autorizados, possíveis vazamentos de dados sensíveis, como por exemplo, senhas.

Plugins NPAPI deixaram de serem suportados nos navegadores atuais, pelo fato de causarem riscos de segurança para quem esteja utilizando. Desde do 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 (SOMMERVILLE, 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*. O *Python* é uma linguagem interpretada de alto nível, criada por Guido Van Rossum em 1989 e lançada em 1991, atualmente possui o modelo de desenvolvimento comunitário (PYTHON, 2015). Com o auxilio do *framework* *Django*, criado originalmente para gerenciar conteúdos de um jornal da cidade de Lawrence no Kansas, com ele é possível definir a modelagem de dados através de classes *Python* e gerar tabelas do banco de dados para manipulação sem a necessidade de *SQL* (*Structured Query Language*) (DJANGO, 2015).

Com *Python* e *Django* foram desenvolvidas todos os modelos de classes de domínio, que serão detalhadas no capítulo 3, onde através do *ORM* (*Object-relational mapping*) do *Django* é possível criar as tabelas no banco de dados e realizar consultas em através desse *ORM*.

Além do *Python* e *Django*, foram utilizadas tecnologias *NPAPI*, mais precisamente *java applets*. O aplicativo construído com *java applets* faz a interpretação do

código de português estruturado para uma linguagem intermediária, onde essa interpretação é analisada e validada, devolvendo uma resposta para o aluno (GOOGLE, 2015) e (DORNELES; JUNIOR; ADAMI, 2010).

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

Através da engenharia de *software* será produzido um novo portal de algoritmos, desde os estágios iniciais da especificação do sistema até sua manutenção, 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 apresentado a reengenharia de *software* realizada no portal de algoritmo atual.

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

No capítulo 5 são apresentadas as considerações parciais do trabalho.

2 EVOLUÇÃO DE SOFTWARE

Para manter um *software* útil ele deve mudar continuamente, pode ser a partir de uma pressão constante de mudanças que os usuários impõem, para facilitar ou automatizar algumas tarefas do dia-a-dia, entre outros fatores.

Todos os *softwares* passarão pelo processo de envelhecimento, isso é inevitável, algumas causas de problemas podem ser previstas, minimizando os impactos dos danos que são causados por esse fator. Para que um *software* se mantenha útil é preciso que ocorram mudanças, elas 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 são produzidas em cada estágio do processo (SOMMERVILLE, 2011).

O processo de evolução de *software* envolve a compreensão do *software* que tem que ser alterado. Para torna-se possível a evolução é preciso aplicar reengenharia no *software* atual, visando melhorar sua estrutura e inteligibilidade (SOMMERVILLE, 2011).

Para tornar possível a evolução de *software* é preciso seguir alguns processos, esses que serão descritos nas próximas sessões:

- Reengenharia de *Software*
- Engenharia de *Software*
 - Processo de *Software*
 - Engenharia Reversa
 - Engenharia de Requisitos
 - Casos de Uso
 - Modelagem de Sistema
 - Projeto de Arquitetura
 - Implementação
- Usabilidade
- Tecnologias
 - REST (Representational State Transfer)*
 - AngularJS*
 - Servidores

2.1 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 2.1, 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 a liga que mantém as camadas de tecnologia coesa e possibilita o desenvolvimento do *software* (PRESSMAN, 2011).

Figura 2.1: Camadas da Engenharia de Software - Fonte (PRESSMAN, 2011)



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

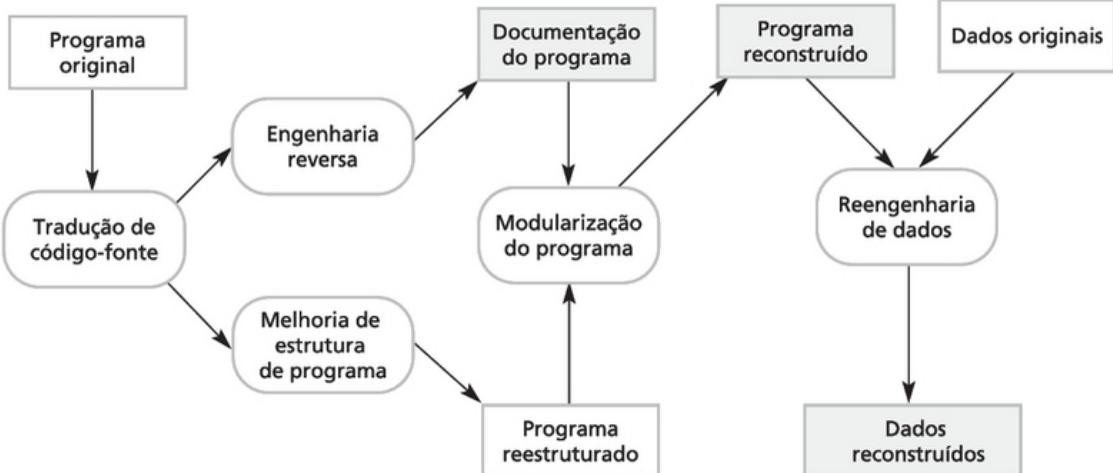
2.1.1 Reengenharia de Software

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

Figura 2.2: 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, está 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 2.2. Pode haver casos que se use 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 sessão será descrito como ela é utilizada no processo de evolução.

2.1.2 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 se faz o uso de códigos fontes, conhecimentos técnicos e experiências dos próprios desenvolvedores.

Na próxima seção são descritos todo o conjunto do processos de *software* e suas atividades, ações e tarefas relacionadas.

2.2 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 pessoas 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 *software*
- Validação 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, teste etc.

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

2.2.1 Engenharia de requisitos

Engenharia de requisitos de sistemas basicamente são as 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 a 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 reair a entradas específicas e de como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem esplicitar o que o sistema não deve fazer (SOMMERVILLE, 2011).

Requisitos não-funcionais são restrições aos serviços ou funções oferecidos 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).

2.3 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

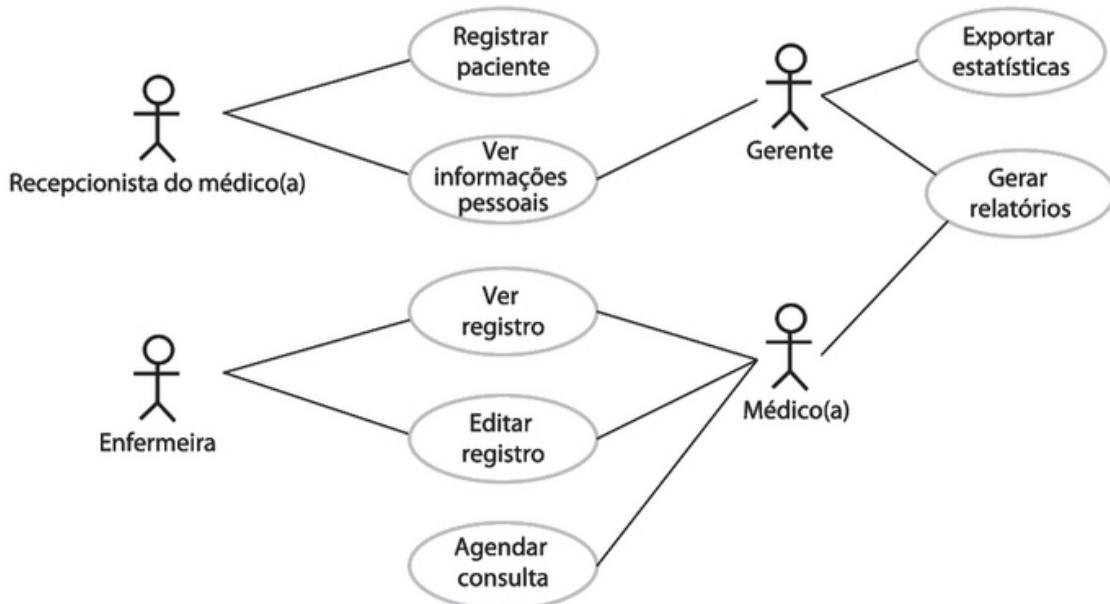
(SOMMERVILLE, 2011) e (PRESSMAN, 2011).

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, eles 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 outro sistema que interagem entre si e cenários é uma sequência específica de ações. Em outros termos casos de uso é uma coleção de cenários relacionados ao sucesso ou fracasso (LARMAN, 2007).

A figura 2.3 ilustra alguns casos de uso para um sistema de informações de pacientes.

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).

2.4 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 conhecido como modelos conceituais.

2.5 Arquitetura

No projeto de arquitetura é a representação da estrutura de dados e seus componentes, ele compreende de como o sistema deve ser organizado a fim de atender as necessidades levantada na engenharia de requisitos (SOMMERVILLE, 2011) e (PRESSMAN, 2011).

Na arquitetura em camadas encontra-se todas as camadas do software, onde é definida a responsabilidade de cada uma. Nesse padrão de arquitetura é uma das maneiras de se conseguir independência entre elas, por exemplo o Modelo-Visão-Controlador MVC, onde é separado a camada de apresentação da interação dos dados do sistema (SOMMERVILLE, 2011) e (PRESSMAN, 2011).

2.6 Usabilidade

Sistemas devem ser fáceis de usar e de aprender, flexíveis e devem despertar nas pessoas uma boa atitude. A usabilidade é a principal busca da *IHC* (Interação humano-computador).

De acordo 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 adequados 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.

2.7 Tecnologia

Nesta sessão serão descritas as tecnologias que vão ser utilizadas na evolução do portal de algoritmo, 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 sérias e de grande porte.

2.7.1 JAVA EE

JAVA EE (Java Enterprise Edition)

2.7.2 REST

2.7.3 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, com ele é possível aumentar sua produtividade no desenvolvimento *WEB* (Rede de Internet) (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 (HyperText Markup Language)* padrão, tornando a vida dos desenvolvedores mais fácil (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 *MVC (Model-View-Controller)* ganhou muita popularidade na 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 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 design* e programadores trabalhem lado a lado, com a ajuda das diretivas, que é um tipo de extensão do vocabulário *HTML*, que traz a capacidade de executar tarefas de linguagem de programação (BRANAS, 2014).

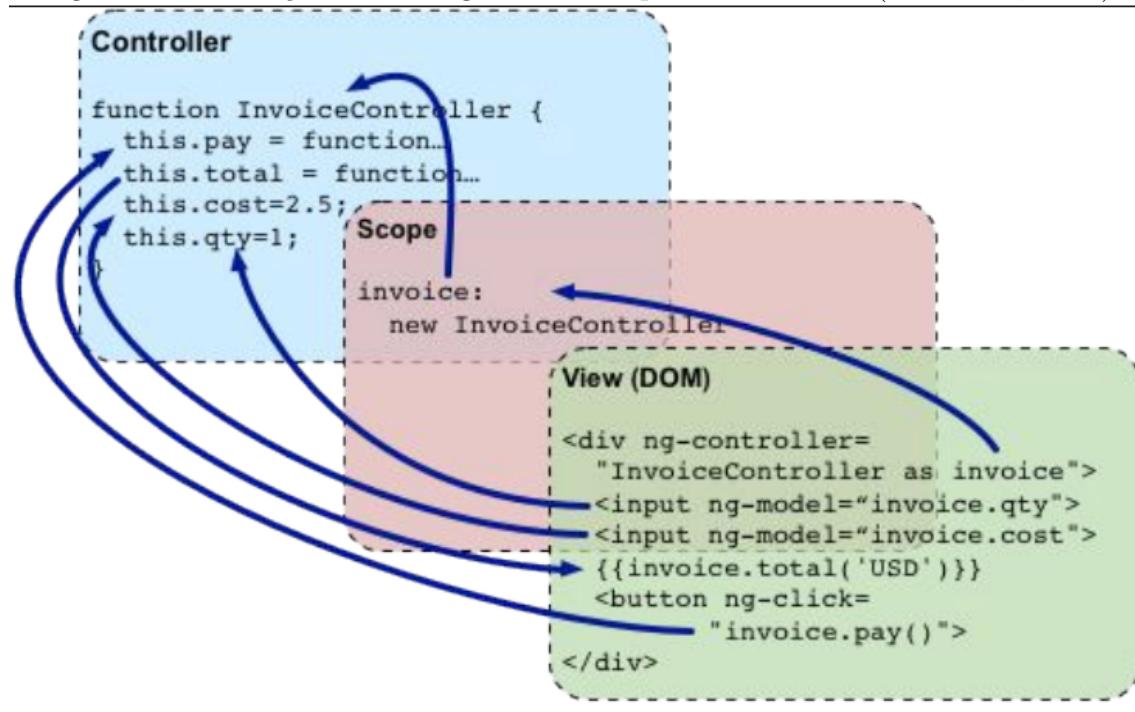
Atrás da *View* existe um *Controller*, nele contém toda a lógica do negócio usado

pela *View*.

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

A figura 2.4 representa a interação entre os componentes do *AngularJS*

Figura 2.4: Interação entre AngularJS e Arquitetura - Fonte (BRANAS, 2014)



2.7.4 Servidor

3 REENGENHARIA DE SOFTWARE

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:

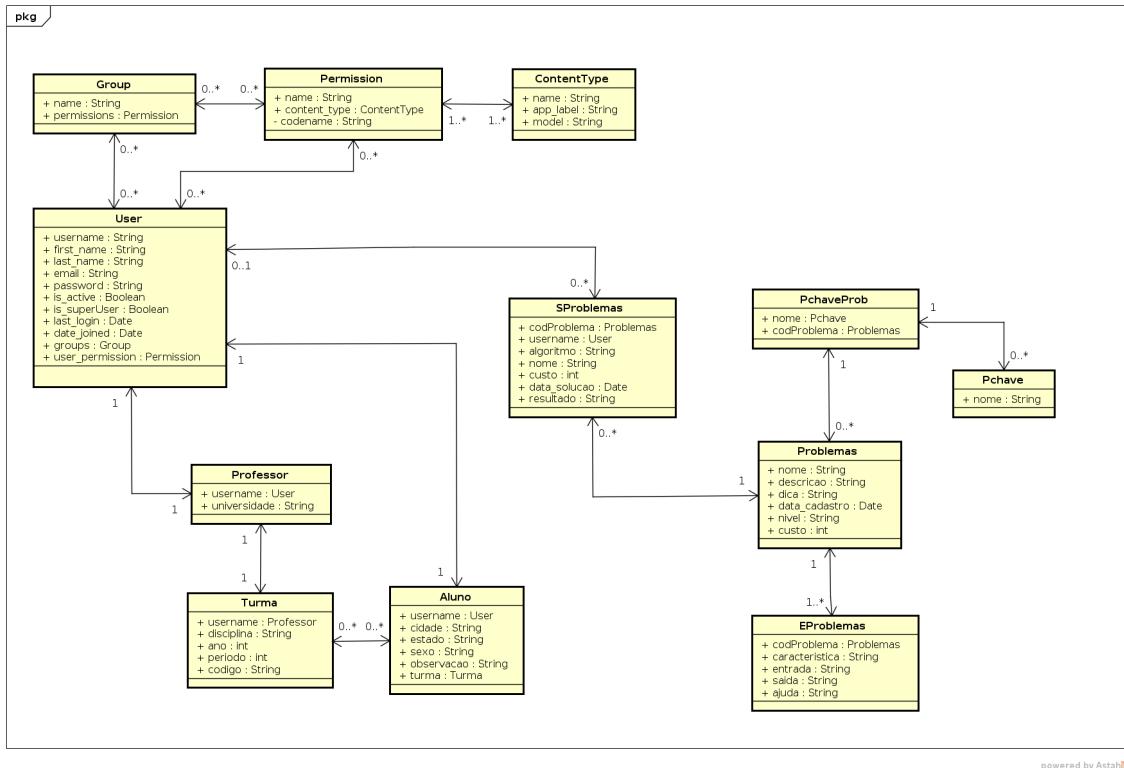
- Os campos não estavam padronizado utilizado *CamelCase*
- Nomes de campos descritos em inglês e em português
- Nomes de classes descritos em inglês e em português
- Nomes de classes não são intuitivas quanto ao seu objetivo

CamelCase é a denominação em inglês para prática de escrever palavras compostas, onde cada palavra é iniciada com minúsculas 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



Abaixo é descrito o que cada classe representa no *software* atual, visando manter todas as funcionalidades após a evolução do mesmo.

3.1.1 User

User representa os usuários. O usuário tem algumas funções importantes no sistema, somente com ele é possível ter acesso a áreas restritas, bem como resolver os exercícios do portal.

3.1.2 Groups

Group 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.

3.1.3 Permission

Permission corresponde as permissões utilizadas no sistema. Tem por objetivo definir permissões de acesso ao sistema, por exemplo se um usuário que possui acesso ao gerenciamento do portal pode cadastrar um aluno, um novo problema.

3.1.4 ContentType

ContentType corresponde os tipos de conteúdo, em tradução livre, é uma classe padrão do *framework django*, ela representa e armazena informações dos modelos utilizados no projeto. Sempre que é criado um modelo novo é criado um tipo de conteúdo automaticamente.

3.1.5 Aluno

Aluno corresponde aos alunos. No portal essa classe é um complemento aos dados do usuário, representando um aluno com suas respectivas informações.

3.1.6 Professor

Professor corresponde aos professores. No portal é um complemento aos dados do usuário, representando um professor com suas respectivas informações.

3.1.7 Turma

Turma corresponde a turma. Essa classe não é utilizada no sistema atual.

3.1.8 Problema

Problema corresponde os problemas. Nessa classe é armazenado todas as informações corresponde ao problema, somente administradores do portal tem permissão para gravar infomações. Essas permissões não são controladas pela classe de domínio *Permission*, esse controle funciona através do campo *is_superuser* da classe de domínio *User*.

3.1.9 Pchave

Pchave corresponde as palavras chaves. Representa uma palavra chave que é utiliza na tabela a seguir. Palavras chaves é um facilitador para a pesquisa de problemas.

3.1.10 PchaveProb

PchaveProb corresponder a relação entre a palavra chave e um problema. Essa classe é utilizada para poder referenciar mais de uma palavra chave para o mesmo problema.

3.1.11 EProblema

EProblema corresponde a tabela de entradas e saídas esperadas de um determinado problema, é possível informar alguma caracteristica que ajude o usuário a resolver um determinado exercício.

3.1.12 SProblema

SProblema corresponde as soluções submetidas pelos usuários em algum exercício resolvido. Possui relação direta com o usuário e um problema.

3.2 Interfaces Gráficas

Nessa sessão 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

Problemas encontrados:

- *Plugin NPAPI* funciona somente no internet explorer após algumas configurações nas excessões do *Java*.
- Pouca Usabilidade.

O cadastro do aluno está misturado com a 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

The screenshot shows the WebAlgo platform interface. On the left, there is a code editor window containing the following pseudocode:

```

1 algoritmo
var n : inteiro
inicio
leia(n)
escreva( n * n )
fimalgoritmo

```

Below the code editor are several buttons: Executar, Passo/Passo, Continuar, Encerrar, Linha a Linha (checked), Leia Aleatório, Dados de teste, and Valida Solução.

On the right, the user is identified as Usuário:amargarin. The interface includes tabs for Manual do Usuário, Gerência de Usuários, Banco de Algoritmos, and Top 5. A dropdown menu for 'Tipo de Problema' shows 'Seqüencial' selected.

The 'Descrição do problema' section contains the following text: "Fazer um algoritmo que leia um valor N, representando o lado de um quadrado, e calcule e escreva a área do quadrado. Dica:A área de um quadrado de lado N é dada por N x N." Below this is an 'Exemplo de [Entrada | Saída] (na ordem)' section showing the input '5' and output '25'.

The 'Soluções de menor custo:' section lists solutions: S00000050, S00000100, S00000150, S00000200, S00000205, S00000210, S00000220, S00000300, S00000350, S00000400, S00000500, S00000600, S00000700, S00000800, S00000900, S00001000, S00001100, S00001200, S00001300, S00001350, S00001400, S00001500, and S00001600. The solution S00000050 is highlighted.

A 'Minhas soluções para o problema: S00000050' section shows three solutions: S001, S002, and S003. Solution S001 is highlighted and has a note: "Solução validada. Custo:16". Buttons for 'Criar Solução' and 'Salvar Solução:[S001_S00000050]' are also present.

Problemas encontrados:

- *Plugin NPAPI* funciona somente no internet explorer após algumas configurações nas excessões do Java.
- Pouca Usabilidade.
 - Diversas confirmações para trocar de problema para ser solucionado
- Para selecionar outro problema tem 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, entende-se por livre qualquer palavra digitada no campos de pesquisa.
- Pesquisa de alunos: É possível realizar uma pesquisa livre, entende-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 realizado 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 realizado uma pesquisa para encontrar as soluções desse aluno.

Figura 3.4: Gerência de Alunos

PortAlgo - Gerencia de Alunos

Supervisor: amargarin

The screenshot shows a web-based application for managing students. At the top, there are two search bars: 'Problemas' and 'Login dos Alunos', each with a 'Filtrar' button. Below these are four main sections:

- Problemas:** A list of problem identifiers, with 'S00000050' highlighted. A scroll bar is visible on the right.
- Descrição do problema:** A text area containing the problem statement: "Faça um algoritmo que leia um valor N, representando o lado de um quadrado, e calcule e escreva a área do quadrado."
- Dica para o problema:** A text area containing the hint: "A área de um quadrado de lado N é dada por N x N."
- Palavras chave o problema:** An empty input field.

On the right side, there are three vertical lists:

- Aluno:** A list of student names, with '(1088566)' highlighted. The list includes: (ABraido), (AGRSilva), (APBalbinot), (Abreu95), (AlissonLF), (Anapmdm), (Anderson), (BMJoaquim), (BSirtori), (BVVieira), (CABFocche), (CPSilva13), (CleberCSO), and (Colcci77).
- Solução:** A list showing solutions for problems S001 and S002.
- Problemas:** An empty list.
- Solução:** An empty list.

At the bottom left, there is a list of student names: lsilveira, 1088566, 11808s8, 12marcola3, 24681012, 27081996, 91319832, and a.martini. Next to it is a section labeled '[...]' with 'Aluno' below it, also containing an empty list.

Problemas encontrados:

- *Plugin NPAPI* funciona somente no internet explorer após algumas configurações nas excessões do *Java*.
- Pouca Usabilidade.

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 chaves
Alterar entradas e saídas

Figura 3.5: Gerência de Problemas

PortAlgo - Gerencia de Problemas Administrador: amargarin

The screenshot shows a web-based application for managing problems. At the top, there are buttons for 'Novo Problema' and 'Cadastrar Problema'. Below this is a sidebar titled 'Problemas Existentes' listing various problem IDs from S00000050 to S00001650. The main area displays problem S00000050. The problem details are as follows:

- [S00000050]**
- Descrição do problema:** Faça um algoritmo que leia um valor N, representando o lado de um quadrado, e calcule e escreva a área do quadrado.
- Dica para o problema:** A área de um quadrado de lado N é dada por N x N.
- Palavras chave o problema:** (empty)
- Exemplo de entrada (na ordem):** 5|3|4|6
- Saida após e/ou durante processamento da entrada:** 25|9|16|36

At the bottom right are buttons for 'Gravar Problema', 'Alterar Descrição/Dicas', 'Alterar Palavras Chave', and 'Alterar Entradas/Saídas'.

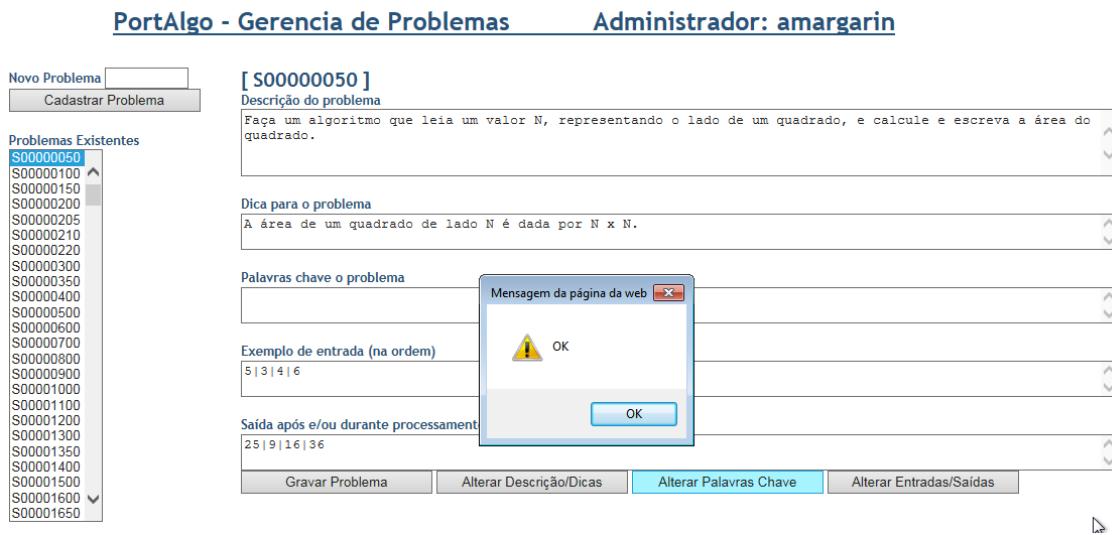
Problemas encontrados:

- *Plugin NPAPI* funciona somente no internet explorer após algumas configurações nas excessões do *Java*.
- 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 Palavras Chaves

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

Figura 3.6: Edição de palavras chaves



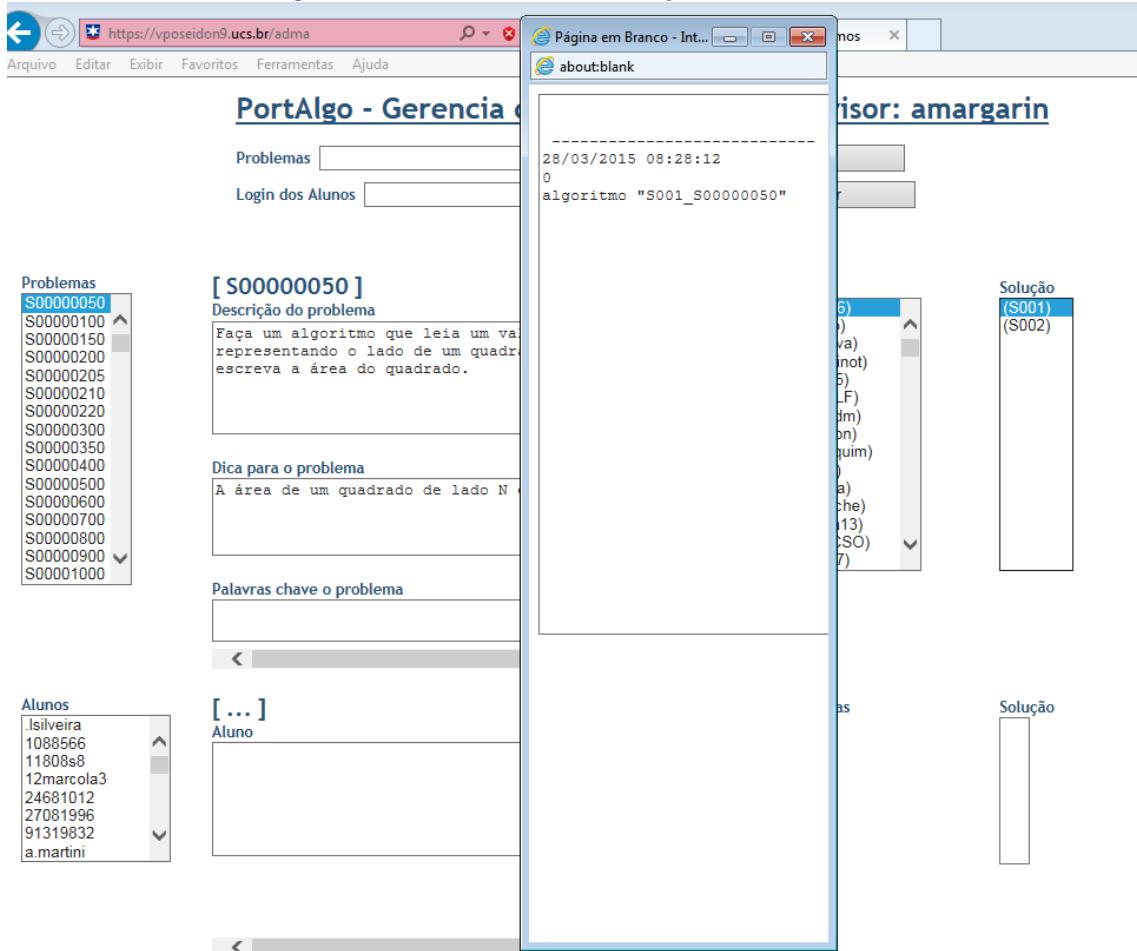
Problemas encontrados:

- *Plugin NPAPI* funciona somente no internet explorer após algumas configurações nas exceções do *Java*.
- 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



Problemas encontrados:

- *Plugin NPAPI* funciona somente no internet explorer após algumas configurações nas excessões do *Java*.
- 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* na versão 9, versões mais atuais desse mesmo navegador já deixaram de suportar o portal, devido ao *plugin NPAPI*.

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

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 sessões seguintes serão apresentados conceitos e artefatos da engenharia de *software*. Utilizando-se de artefatos da metodologia ICONIX será modelado o *software* (ROSENBERG; STEPHENS; COLLINS-COPE, 2005).

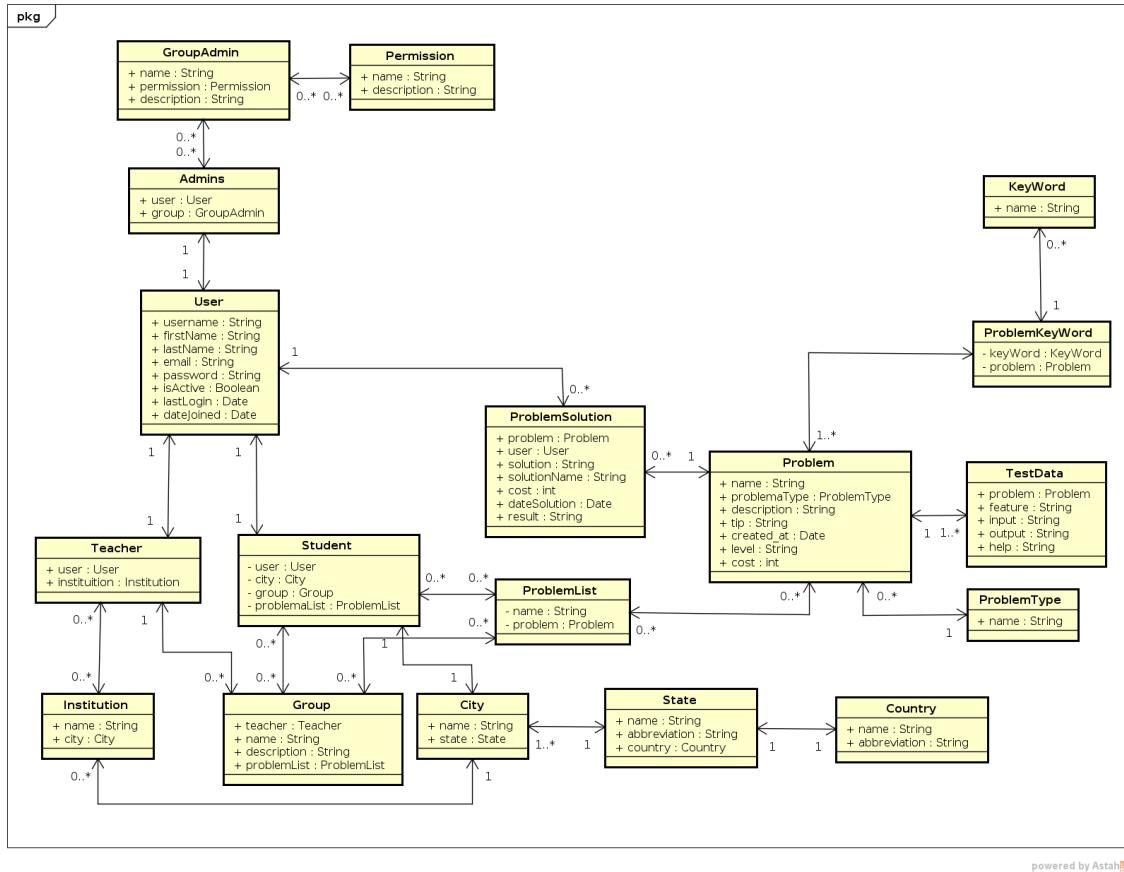
A metodologia ICONIX utiliza-se de um subconjunto da *UML*, apenas 4 diagramas são usados, diagrama de classe, diagrama de sequência, diagrama de robustez e caso de usos (ROSENBERG; STEPHENS; COLLINS-COPE, 2005).

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



A 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. Esse requisitos descrevem o que o sistema deve fazer e suas restrições.

As tabelas 4.1 e 4.2 são respectivamente os requisitos funcionais e não-funcionais do portal de algoritmo a ser desenvolvido.

Tabela 4.1: Requisitos funcionais

Código	Descrição
RF-001	O <i>software</i> deve manter Usuários.
RF-002	O <i>software</i> deve manter Alunos.
RF-003	O <i>software</i> deve manter Professores.
RF-004	O <i>software</i> deve manter Administradores.
RF-005	O <i>software</i> deve manter Tipo de Problemas.
RF-006	O <i>software</i> deve manter Problemas.
RF-007	O <i>software</i> deve manter Palavras Chaves.
RF-008	O <i>software</i> deve manter Entradas e Saídas para os Problemas.
RF-009	O <i>software</i> deve manter Soluções de Problemas.
RF-010	O <i>software</i> deve manter Lista de Problemas.
RF-011	O <i>software</i> deve manter Instituições.
RF-012	O <i>software</i> deve manter Grupos.
RF-013	O <i>software</i> deve manter Grupos de Administradores.
RF-014	O <i>software</i> deve manter Permissões.
RF-015	O <i>software</i> deve manter Países.
RF-016	O <i>software</i> deve manter Estados.
RF-017	O <i>software</i> deve manter Cidades.
RF-018	O <i>software</i> deve possuir um chat para comunicação entre os usuários.
RF-019	O usuário deve manter suas informações pessoais

Entende-se quando se refere em "manter", listar, cadastrar, editar e deletar um objeto.

Tabela 4.2: Requisitos não-funcionais

Código	Descrição
RNF-001	O <i>software</i> deve ser uma aplicação <i>WEB</i> .
RNF-002	O <i>software</i> deve possuir uma <i>API (Application Programming Interface REST)</i> .
RNF-003	O <i>software</i> deve ter uma interface gráfica de fácil operação.
RNF-004	O <i>software</i> deve executar nos principais navegadores de internet.

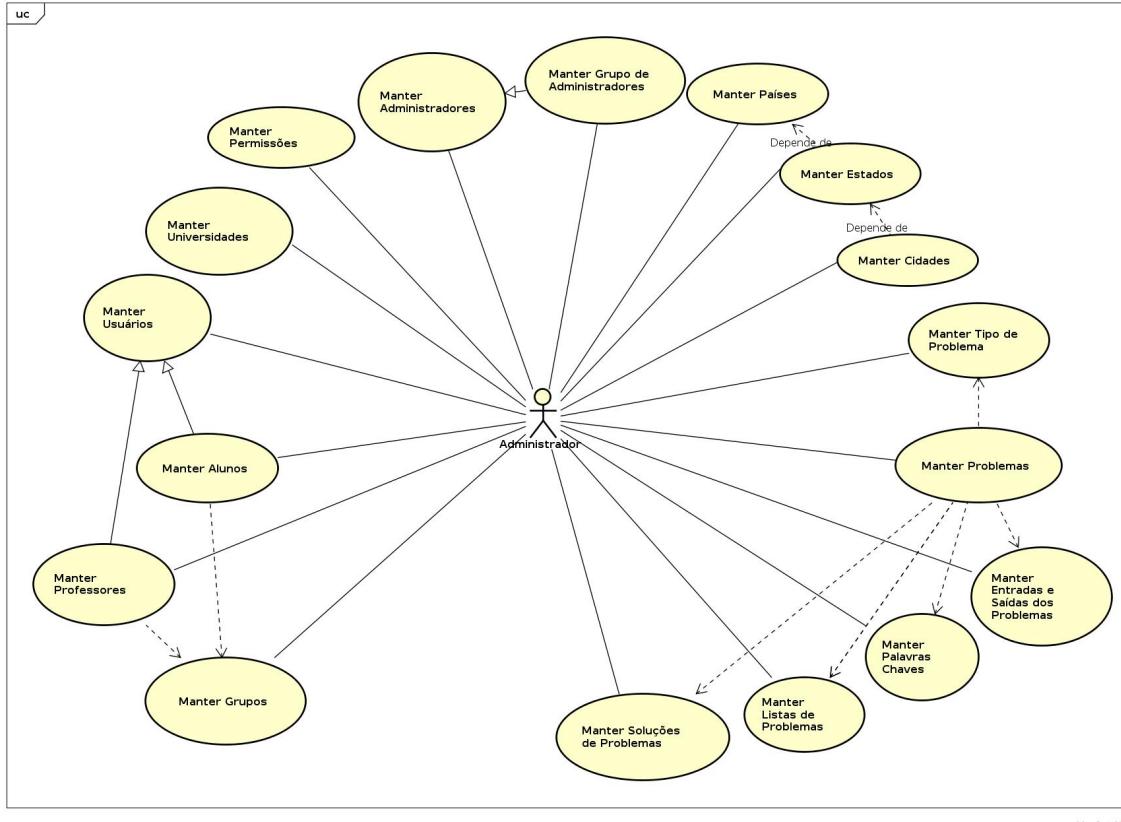
Cada requisito funcional elicitado pode se tornar um caso de uso, esses casos de uso que serão descritos na sessão seguinte.

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*.

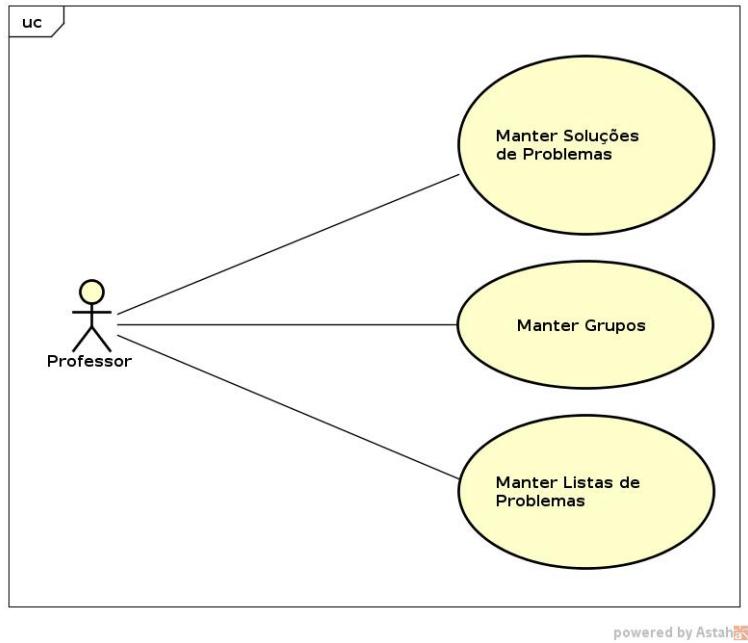
A figura 4.2 estão representados os casos de uso em que o ator ”Administrador” está envolvido para o gerenciamento do portal.

Figura 4.2: Casos de Uso que envolvem o ator Administrador



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

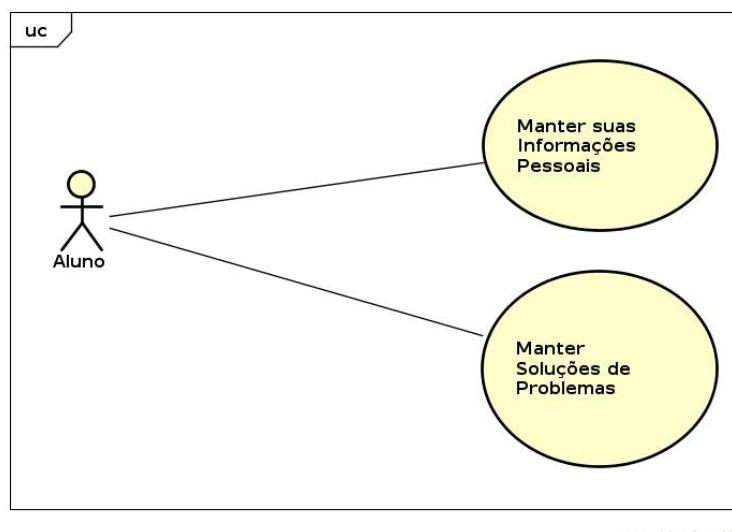
Figura 4.3: Casos de Uso que envolvem o ator Professor



powered by Astah

A 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



powered by Astah

A seguir são apresentados detalhadamente cada casos de uso apresentados nas figuras 4.2, 4.3 e 4.4. Serão apresentados os fluxos de execução principais e alterativos 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 figura 4.5 descreve o caso de uso de manter o cadastro, edição e remoção das informações dos usuários, possui um único ator, o ”Administrador”, a pré-condição é ser um administrador do sistema.

Figura 4.5: Caso de Uso Manter Usuários

CASO DE USO 001 - Manter Usuários	
Descrição	O administrador cadastra usuários para o software, podendo editar e remover.
Autor	Administrador.
Pré-condição	Ser um administrador do portal de algoritmo.
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 obrigatórios. 1. Sistema exibe mensagens de erros.
Pós-condição	Usuário cadastrado

4.3.1.2 Manter Alunos

A figura 4.6 descreve o caso de uso de manter o cadastro, edição e remoção das informações dos alunos, possui um único ator, o ”Administrador”, possui duas pré-condições, ser administrador do sistema e o aluno é preciso estar ligado a um usuário.

Figura 4.6: Caso de Uso Manter Alunos

CASO DE USO 002 - Manter Alunos	
Descrição	O administrador cadastra alunos para o software, podendo editar e remover. Para o cadastro de um aluno é necessário ter um usuário já cadastrado.
Autor	Administrador.
Pré-condição	CASO DE USO 001. Ser um administrador do portal de algoritmo.
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.

4.3.1.3 Manter Professores

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

Figura 4.7: Caso de Uso Manter Professores

CASO DE USO 003 - Manter Professores	
Descrição	O administrador cadastra professores para o software, podendo editar e remover. Para o cadastro de um professor é necessário ter um usuário já cadastrado.
Autor	Administrador.
Pré-condição	CASO DE USO 001. Ser um administrador do portal de algoritmo.
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.

4.3.1.4 Manter Administradores

A figura 4.8 descreve o caso de uso de manter o cadastro, edição e remoção das informações dos administradores do sistema, 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.

Figura 4.8: Caso de Uso Manter Administradores

CASO DE USO 004 - Manter Administradores	
Descrição	O administrador cadastra outros administradores para o software, podendo editar e remover. Para o cadastro de um administrador é necessário ter um usuário já cadastrado.
Autor	Administrador.
Pré-condição	CASO DE USO 001. Ser um administrador do portal de algoritmo.
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.

4.3.1.5 Manter Tipo de Problema

A figura 4.9 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.

Figura 4.9: Caso de Uso Manter Tipo de Problema

CASO DE USO 005 - Manter Tipo de Problema	
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 algoritmo.
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.

4.3.1.6 Manter Problemas

A figura 4.10 descreve o caso de uso de manter o cadastro, edição e remoção das informações de problemas.

Figura 4.10: Caso de Uso Manter Problemas

CASO DE USO 006 – Manter Problemas	
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 algoritmo.
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.

4.3.1.7 Manter Palavras Chaves

A figura 4.12 descreve o caso de uso de manter o cadastro, edição e remoção das informações das palavras chaves de um ou mais problemas.

Figura 4.11: Caso de Uso Manter Palavras Chaves

CASO DE USO 007 – Manter Palavras Chaves	
Descrição	O administrador cadastra palavras chaves para um problema, podendo editar e remover.
Autor	Administrador.
Pré-condição	CASO DE USO 006. Ser um administrador do portal de algoritmo.
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 chaves. 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.

4.3.1.8 Manter Entrada e Saídas

A figura 4.12 descreve o caso de uso de manter o cadastro, edição e remoção das informações das entradas e saídas de um determinador problema.

Figura 4.12: Caso de Uso Manter Entrada e Saídas

CASO DE USO 008 – Manter Entradas e Saídas	
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 algoritmo.
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-chaves. 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 administradores. 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.

4.3.1.9 Manter Soluções de Problemas

A figura 4.13 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.

Figura 4.13: Caso de Uso Manter Soluções de Problemas

CASO DE USO 009 - Manter Soluções de Problemas	
Descrição	O administrador, aluno e/ou professor pode cadastrar uma solução para um problema, podendo editar e remover.
Autor	Administrador, Aluno, Professor.
Pré-condição	CASO DE USO 006. Ser um administrador do portal de algoritmo.
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.

4.3.1.10 Manter Listas de Problemas

A figura 4.14 descreve o caso de uso de manter o cadastro, edição e remoção das listas de problemas, elas agrupam problemas para serem aplicados a um grupo e/ou aluno.

Figura 4.14: Caso de Uso Manter Listas de Problemas

CASO DE USO 010 - Manter Listas de Problemas.	
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 algoritmo. 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.

4.3.1.11 Manter Instituições

A figura 4.15 descreve o caso de uso de manter o cadastro, edição e remoção das instituições que podem ser utilizadas no cadastro do professor ou aluno.

Figura 4.15: Caso de Uso Manter Instituições

CASO DE USO 011 - Manter Instituições.	
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 algoritmo.
Fluxo principal	<ol style="list-style-type: none"> 1. Administrador acessa listagem de instituições. 2. Sistema lista todas universidades 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 universidades. 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.

4.3.1.12 Manter Grupos

A figura 4.16 descreve o caso de uso de manter o cadastro, edição e remoção de grupos, eles servem para agrupar alunos e professores, onde os professores e administradores podem criar listas de problemas específicos para os grupos ou alunos.

Figura 4.16: Caso de Uso Manter Grupos

CASO DE USO 012 – Manter Grupos.	
Descrição	O administrador ou professor cadastra 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 algoritmo. 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.

4.3.1.13 Manter Permissões

A figura 4.17 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.

Figura 4.17: Caso de Uso Manter Permissões

CASO DE USO 013 – Manter Permissões.	
Descrição	O administrador cadastra permissões para o software, podendo editar ou remover.
Autor	Administrador.
Pré-condição	Ser um administrador do portal de algoritmo.
Fluxo principal	<ol style="list-style-type: none"> 1. Administrador acessa listagem de permissões. 2. Sistema lista todas universidades já cadastrados. 3. Administrador acessa cadastro de gpermissões . 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.

4.3.1.14 Manter Grupos de Administradores

A figura 4.18 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.

Figura 4.18: Caso de Uso Manter Grupos de Administradores

CASO DE USO 014 – Manter Grupos de Administradores.	
Descrição	O administrador cadastra grupos de administradores, podendo editar ou remover. No cadastro pode ser associado as permissões já cadastradas.
Autor	Administrador.
Pré-condição	CASO DE USO 013. Ser um administrador do portal de algoritmo.
Fluxo principal	<ol style="list-style-type: none"> 1. Administrador acessa listagem de grupos de administradores. 2. Sistema lista todas universidades 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.

4.3.1.15 Manter Países

A figura 4.19 descreve o caso de uso de manter o cadastro, edição e remoção de países.

Figura 4.19: Caso de Uso Manter Países

CASO DE USO 015 – Manter Países	
Descrição	O administrador cadastrá países, podendo editar ou remover.
Autor	Administrador.
Pré-condição	Ser um administrador do portal de algoritmo.
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.

4.3.1.16 Manter Estados

A figura 4.20 descreve o caso de uso de manter o cadastro, edição e remoção de estados.

Figura 4.20: Caso de Uso Manter Estados

CASO DE USO 016 - Manter Estados.	
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 algoritmo.
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.

4.3.1.17 Manter Cidades

A figura 4.21 descreve o caso de uso de manter o cadastro, edição e remoção de cidades.

Figura 4.21: Caso de Uso Manter Cidades

CASO DE USO 017 - Manter Cidades.	
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 algoritmo.
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.

4.3.1.18 Chat Online

A figura 4.22 descreve o caso de uso do chat online, ele servirá para comunicação interna em tempo real dentro do portal de algoritmos.

Figura 4.22: Caso de Uso Chat Online

CASO DE USO 018 – Sistema deve possuir um chat para comunicação entre os usuários.	
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

4.3.1.19 Manter suas Informações Pessoais

A figura 4.23 descreve o caso de uso de cadastro, edição e remoção das informações pessoais do aluno.

Figura 4.23: Caso de Uso Aluno Manter suas Informações Pessoais

CASO DE USO 019 – Manter suas informações pessoais.	
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.

Foram apresentados os casos de uso que serão utilizados para a evolução do portal de algoritmos.

A seguir são apresentados os protótipos de interface do portal de algoritmos.

4.4 Protótipos de Interfaces Gráficas

Os protótipos a seguir procuram seguir os princípios de usabilidade.

4.4.1 Cadastro de Alunos

A figura 4.24 é o protótipo de interface gráfica de cadastro de um novo aluno.

Figura 4.24: Protótipo de Interface Gráfica de Cadastro de Aluno

O formulário de cadastro de aluno contém os seguintes campos:

- Nome**: Campo para inserir o nome.
- Sobrenome**: Campo para inserir o sobrenome.
- E-mail**: Campo com o valor "seu.email@email.com".
- Usuário**: Campo para inserir o usuário.
- Senha**: Campo com o valor "*****".
- Repita a senha**: Campo com o valor "*****".
- Quem é você?**: Campo com o valor "Escolha uma das instituições".
- País**: Botão com o valor "Opções".
- Estado**: Botão com o valor "Opções".
- Cidade**: Botão com o valor "Escolha uma das opções".
- Limpar**: Botão vermelho.
- Registrar**: Botão verde.

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.

- Quem é você?

Escolhe uma das opções disponíveis, como por exemplo: Alunos da UCS (Universidade de Caxias do Sul).

- País.
- Estado.

Somente exibirá as opção depois que selecionar um País.

- Cidade

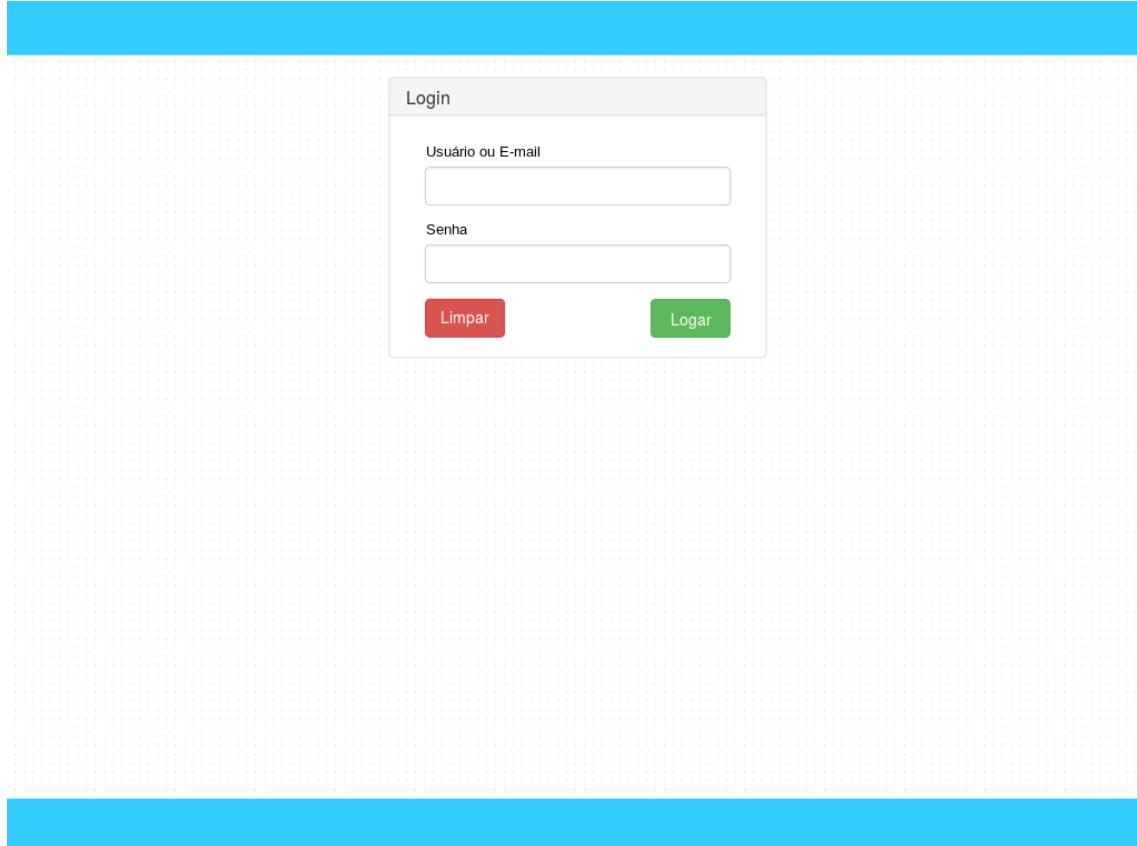
Somente exibirá as opção depois que selecionar um Estado.

Após o preenchimento de todos os campos, o alunos clica em "Registrar" e seu cadastro estará realizado.

4.4.2 Autenticação

A figura 4.25 é o protótipo de interface gráfica de autenticação.

Figura 4.25: Protótipo de Inteface Gráfica de Autenticação



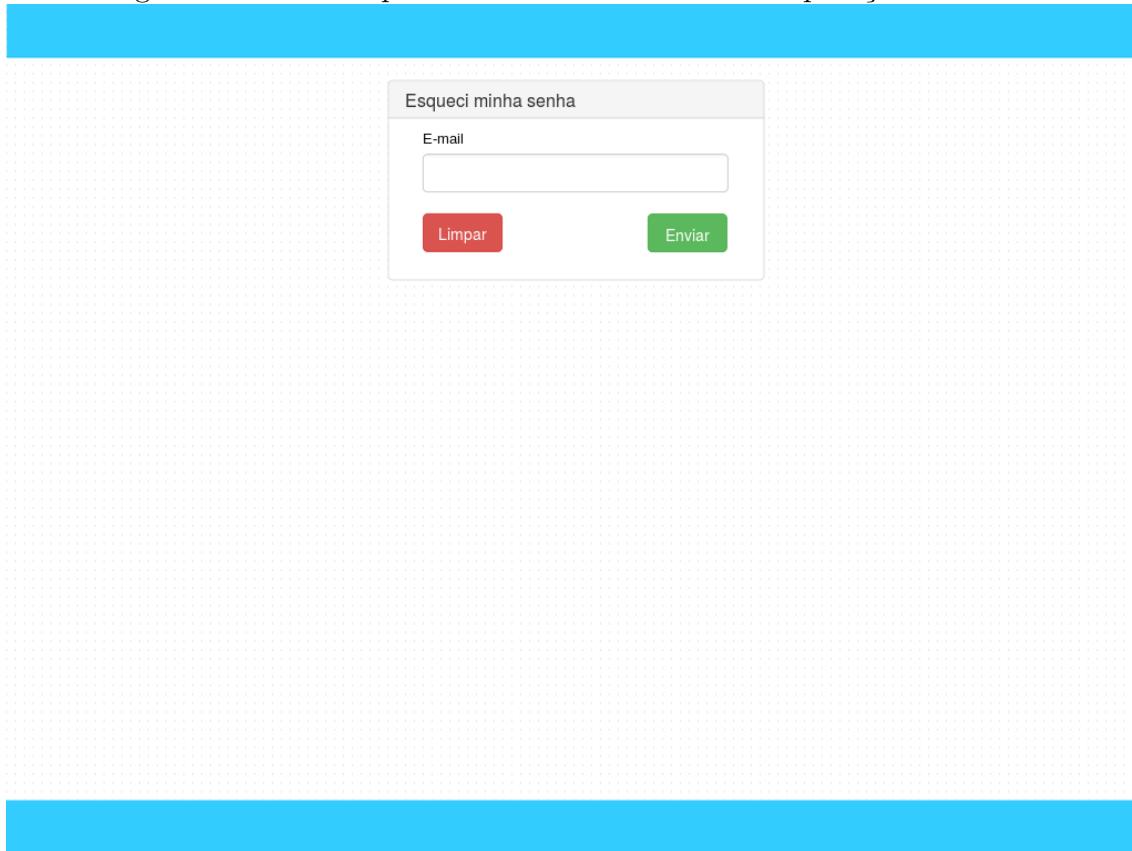
Nessa interface gráfica o aluno, professor ou administrador realizam a autenticação no portal de algoritmos. A autenticação somente é autorizada para usuários ainda ativos no sistema.

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 Recuperação de Senha

A figura 4.26 é o protótipo de interface gráfica de recuperação de senha.

Figura 4.26: Protótipo de Inteface Gráfica de Recuperação de Senha



Nessa interface o usuário é capaz de solicitar uma nova senha somente preenchendo seu e-mail cadastrado no portal de algoritmos.

Após o preenchimento do e-mail o usuário clica em "Enviar", fazendo com que o sistema dispare um e-mail com a senha nova e anulando a senha antiga.

4.4.4 Boas Vindas do Aluno

A figura 4.27 é o protótipo de interface gráfica de Boas Vindas do Aluno.

Figura 4.27: Protótipo de Inteface Gráfica de Boas Vindas do Aluno

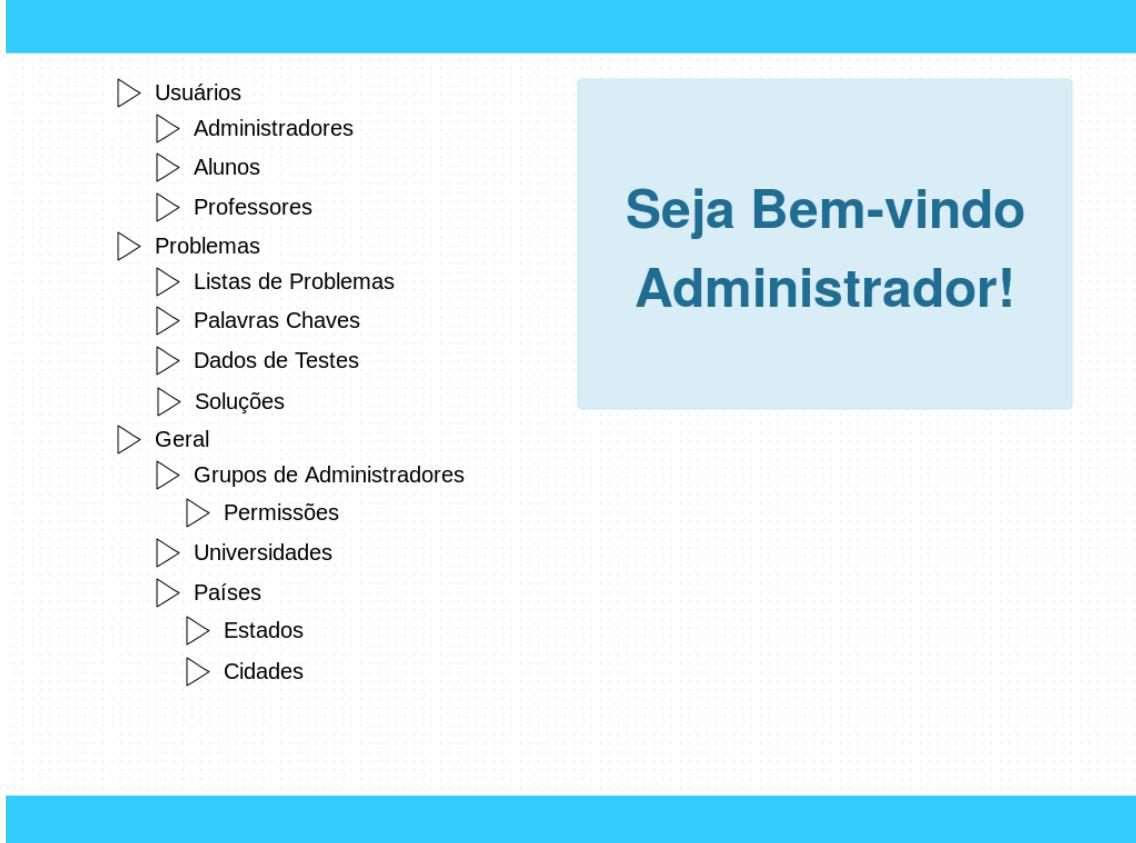


Essa interface gráfica é exibida após a autenticação com sucesso do Aluno. É exibido um vídeo explicativo de utilização do novo portal de algoritmos.

4.4.5 Boas Vindas do Administrador e Professor

A figura 4.28 é o protótipo de interface gráfica de Boas Vindas do Administrador e Professor.

Figura 4.28: Protótipo de Inteface Gráfica de Boas Vindas do Administrador e Professor

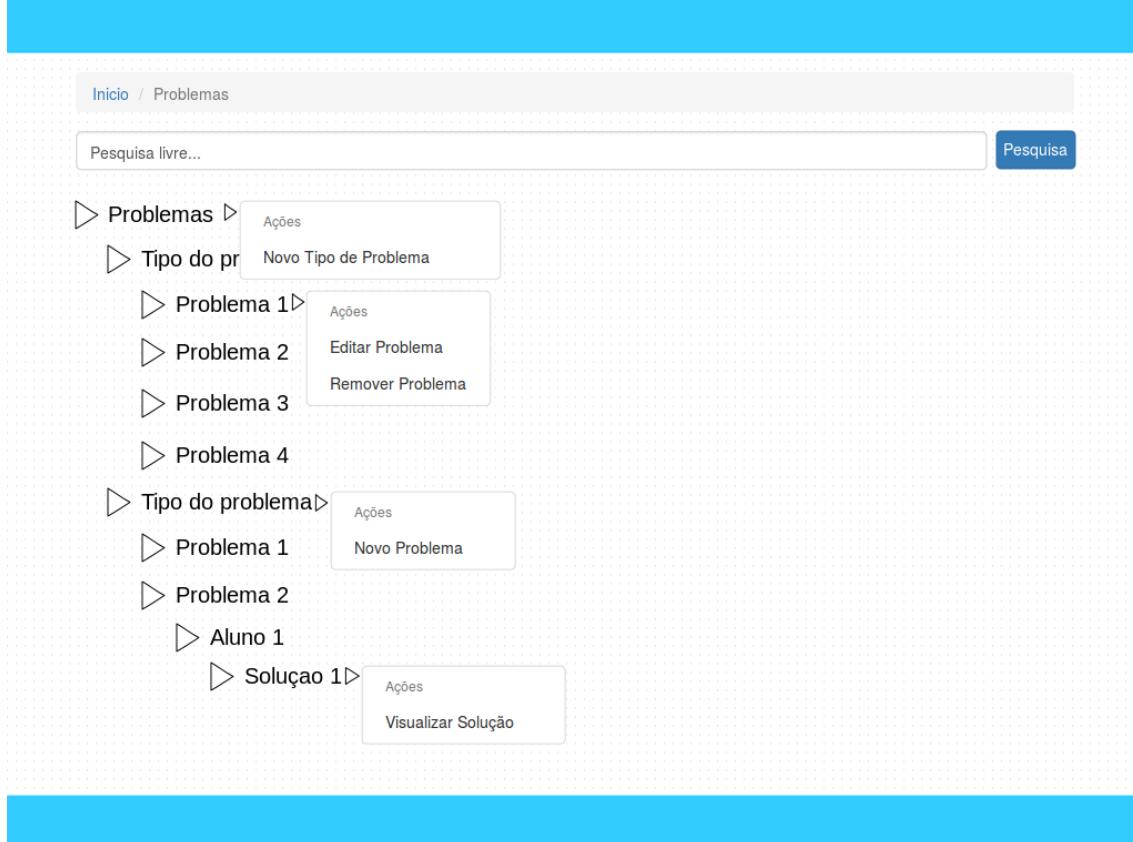


Essa interface gráfica é exibida após a autenticação com sucesso do Administrador ou Professor. Nessa interface é exibido uma árvore com links para as demais interfaces gráficas, agrupadas por contextos.

4.4.6 Lista de Problemas

A figura 4.29 é o protótipo de interface gráfica da listagem de problemas. Os problemas são agrupados por tipo de problemas.

Figura 4.29: Protótipo de Inteface Gráfica de Problemas



Nessa interface gráfica possuímo as seguintes ações:

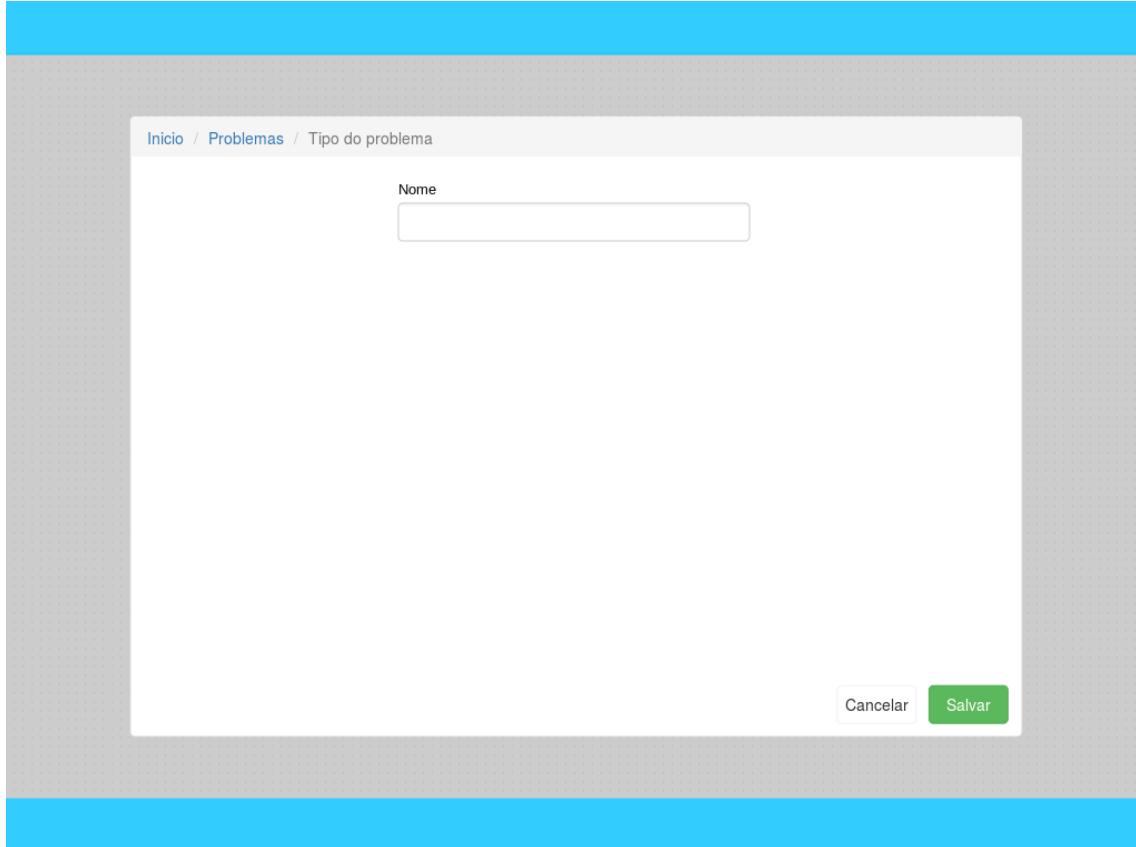
- Pesquisa livre.
O administrador pode realizar a filtragem por qualquer palavra.
- Novo Tipo de Problema.
Selecionado essa ação, o administrador é redirecionado para a interface gráfica da figura 4.30.
- Novo Problema.
Selecionando essa ação, o administrador é redirecionado para a interface gráfica da figura 4.31.
- Editar Problema.
Selecionando essa ação, o administrador é redirecionado para a interface gráfica da figura 4.31.
- Remover Problema.
Selecionando essa ação, o administrador remove o problema que está sofrendo a ação.
- Visualizar Solução.
Selecionando essa ação, o administrador é redirecionado para a interface

gráfica da figura 4.34.

4.4.7 Tipo de Problema

A figura 4.30 é o protótipo de interface gráfica de cadastro de novo tipo de problema.

Figura 4.30: Protótipo de Inteface Gráfica de Novo Tipo de Problema

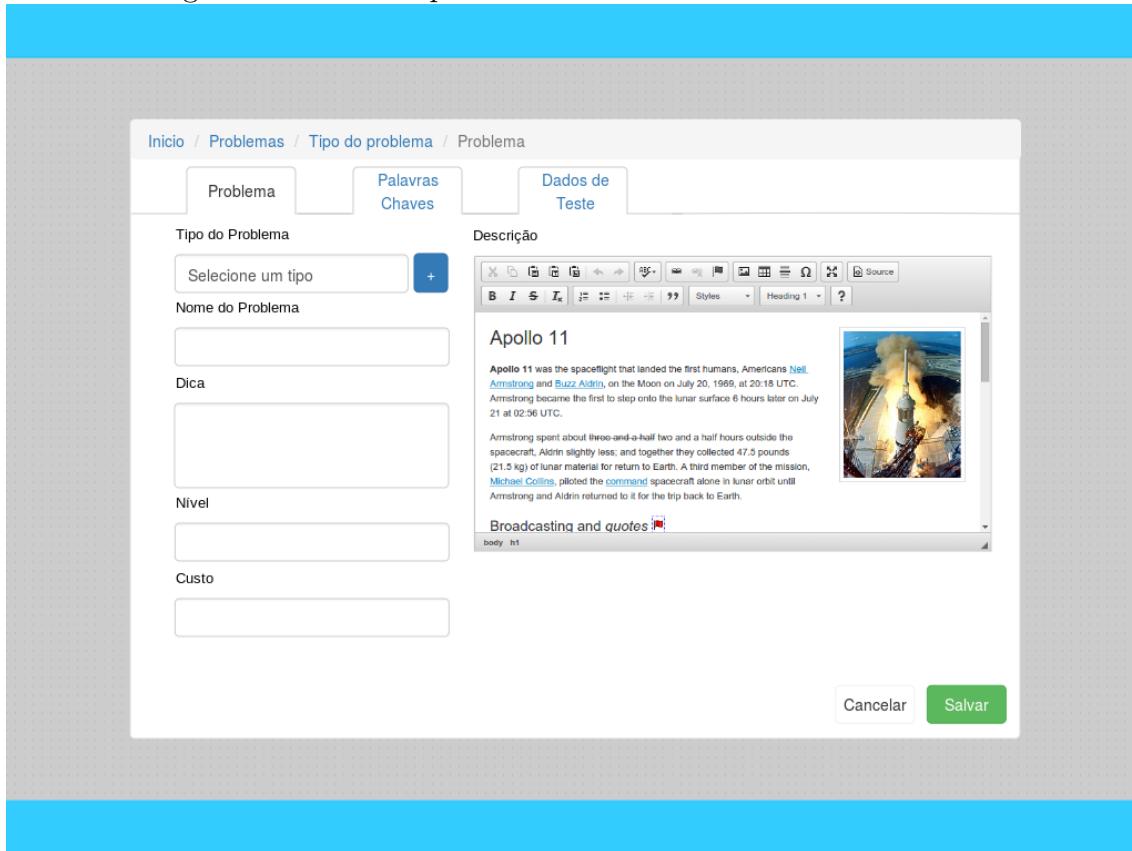


Nessa interface gráfica o administrador preenche somente um "Nome" para o tipo de problema. Após o preenchimento o administrador clica em "Salvar" e é redirecionado para listagem de problemas que foi descrito na seção 4.4.6.

4.4.8 Cadastro e Edição Problema

A figura 4.31 é o protótipo de interface gráfica de cadastro e/ou edição de um novo problema.

Figura 4.31: Protótipo de Inteface Gráfica de Novo Problema



A interface está dividida em três abas, Problema, Palavras Chaves e Dados de Testes, cada uma dessas abas são ações, que serão descritas abaixo:

- Cadastro do 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.

- Cadastro de Palavras Chaves.

O cadastro de palavras chaves é descrito na seção 4.4.9

- Cadastro de Dados de Testes.

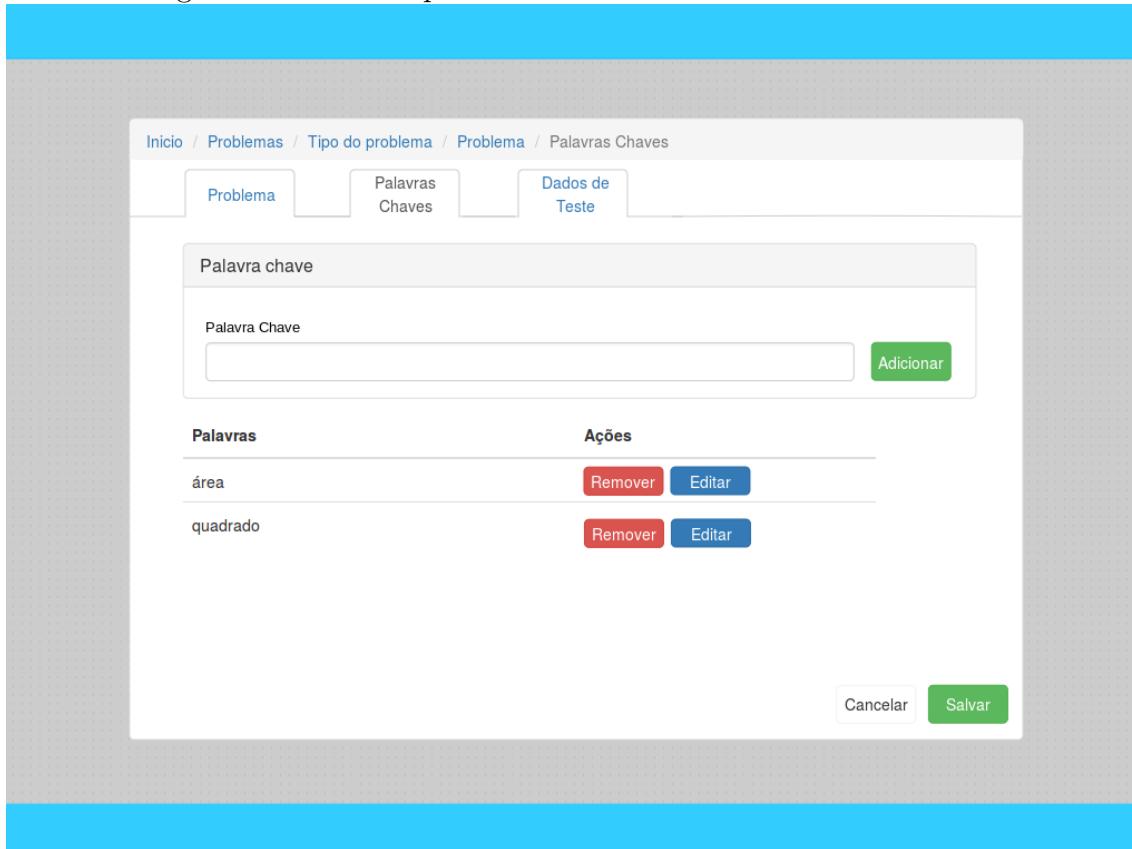
O cadastro de dados de tests é descrito na seção 4.4.10

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

4.4.9 Palavras Chaves

A figura 4.32 é o protótipo de interface gráfica de cadastro e/ou edição de palavras chaves de um problema.

Figura 4.32: Protótipo de Inteface Gráfica de Palavras Chaves

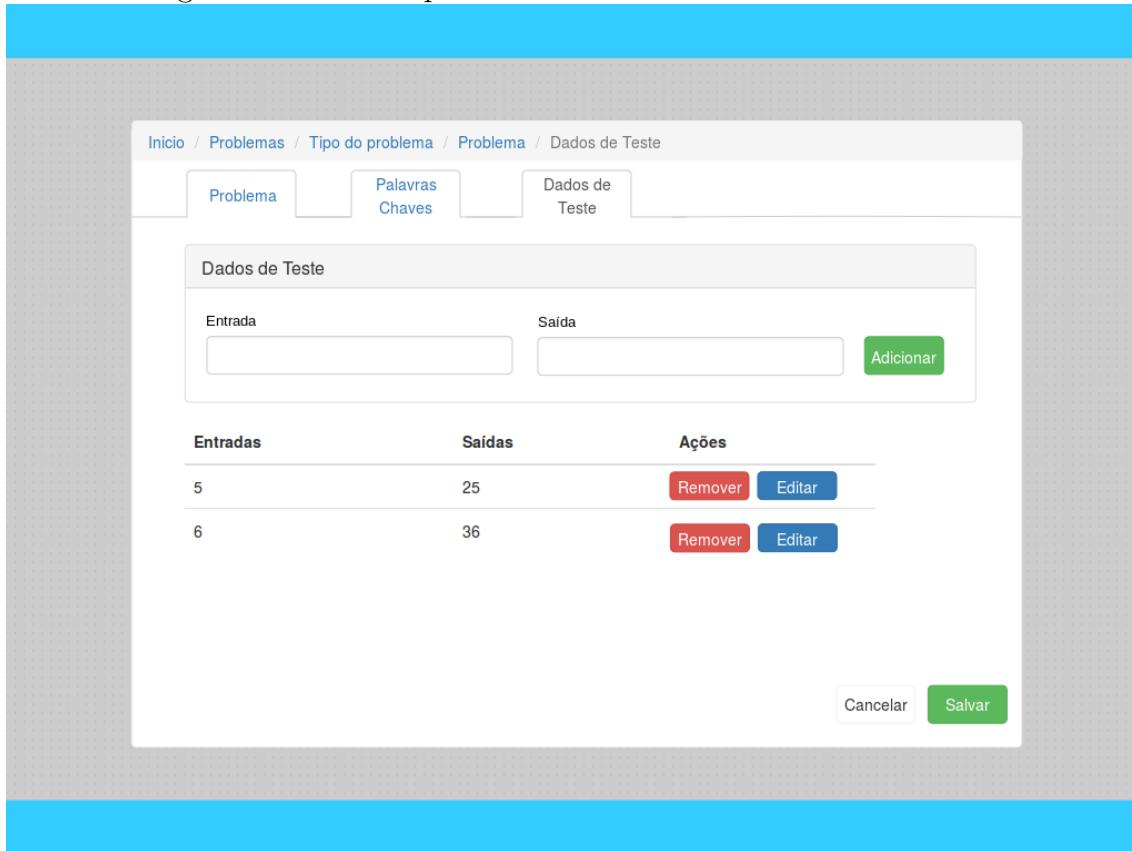


Nessa interface o administrador preenche o campo "Palavra Chave" e clica em "Adicionar", fazendo essa ação ele associa a nova palavra chave ao problema que ele está cadastrando ou editando. O administrador pode também editar ou remover uma palavra chave já cadastrada, para isso é preciso selecionar um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

4.4.10 Dados de Testes

A figura 4.33 é o protótipo de interface gráfica de cadastro e/ou edição de dados de testes de um problema.

Figura 4.33: Protótipo de Inteface Gráfica de Dados de Testes

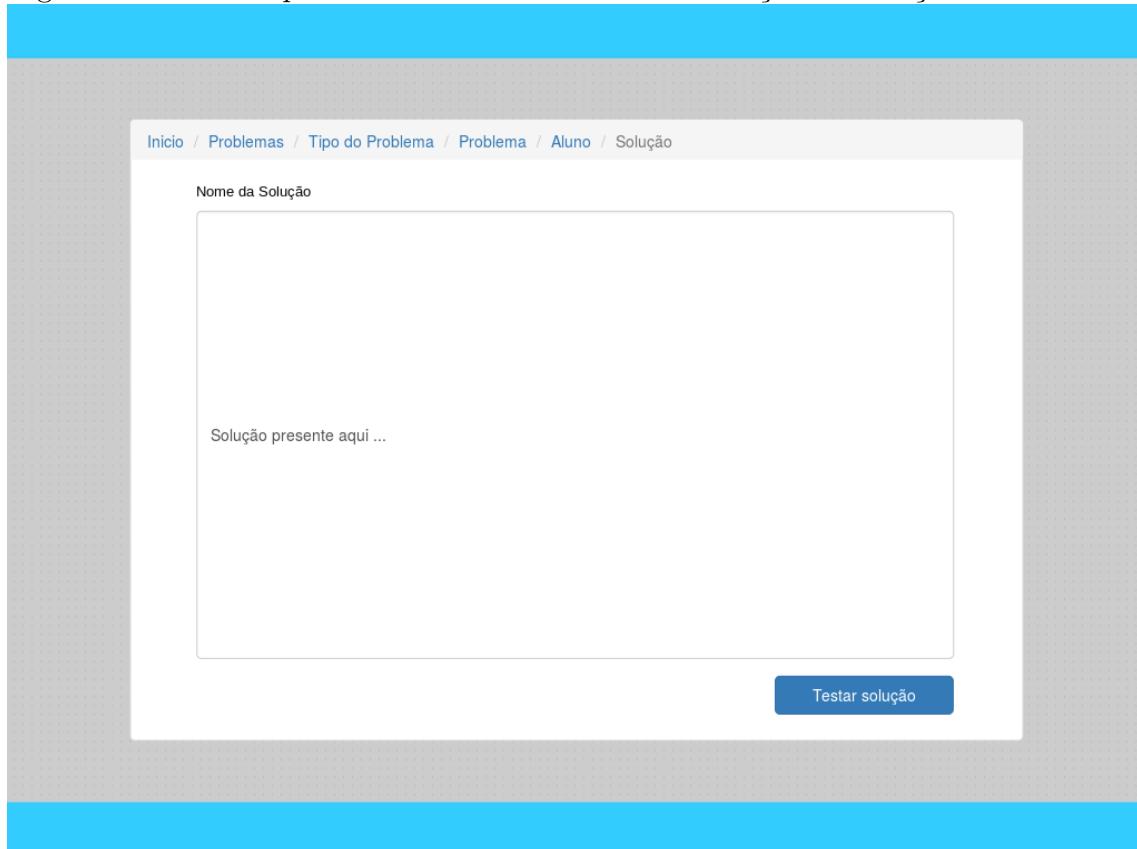


Nessa interface o administrador preenche os campos ”Entradas” e ”Saída” e clica em ”Adicionar”, fazendo essa ação ele associa o novo dado de teste ao problema que ele está cadastrando ou editando. O administrador pode também editar ou remover um dado de teste já cadastrada, para isso é preciso selecionar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

4.4.11 Visualização de Solução a partir da listagem de Problemas

A figura 4.34 é o protótipo de interface gráfica de visualização de uma solução de um determinado problema. Como podemos ver no topo da interface existe o caminho percorrido para chegar até essa solução do problema. Em outras interface que serão descritas no decorrer deste trabalho, esse caminho pode variar de acordo do contexto do gerenciamento do portal de algoritmos.

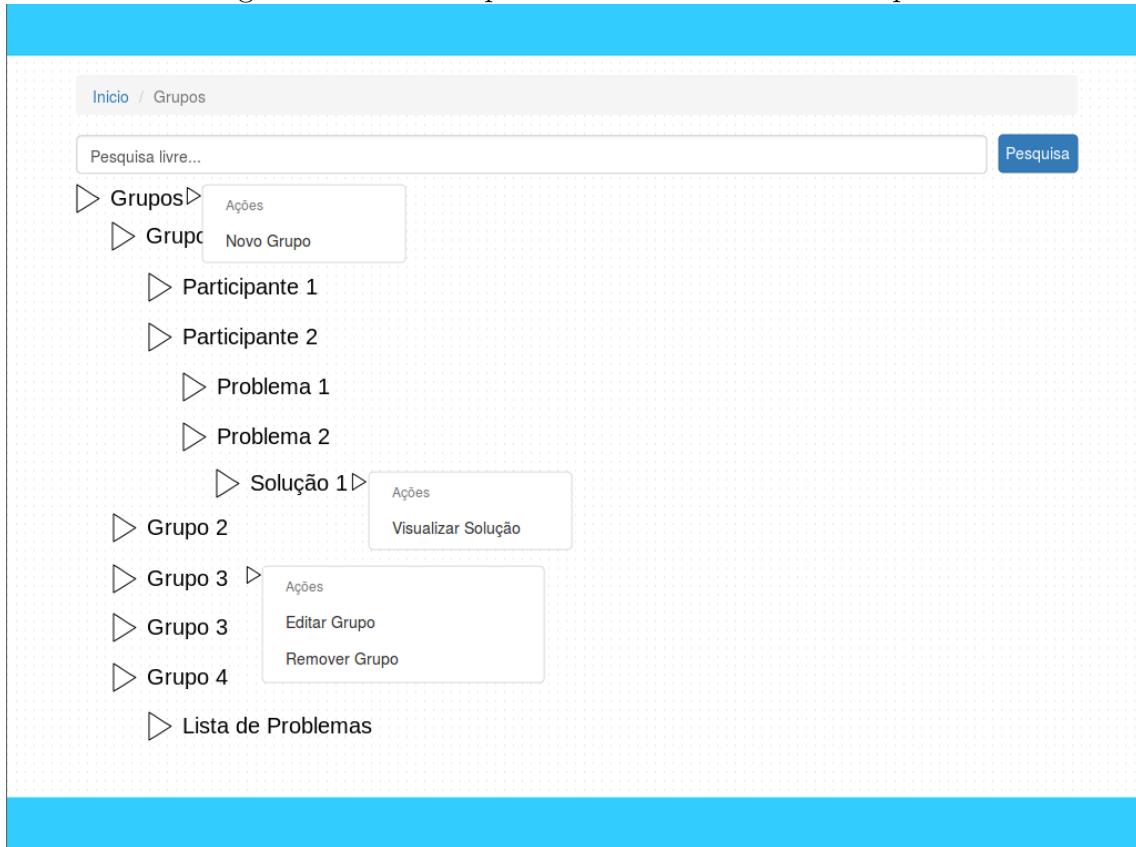
Figura 4.34: Protótipo de Inteface Gráfica de Visualização de Solução do Problema



4.4.12 Lista de Grupos

A figura 4.35 é o protótipo de interface gráfica de listagem dos grupos já cadastrados. Essa listagem corresponde a todos os grupos, caso seja um administrador com total acesso, ou apenas os grupos cadastrados por um professor, sendo que o professor somente terá na listagem os grupos que ele cadastrou.

Figura 4.35: Protótipo de Interface Gráfica de Grupos



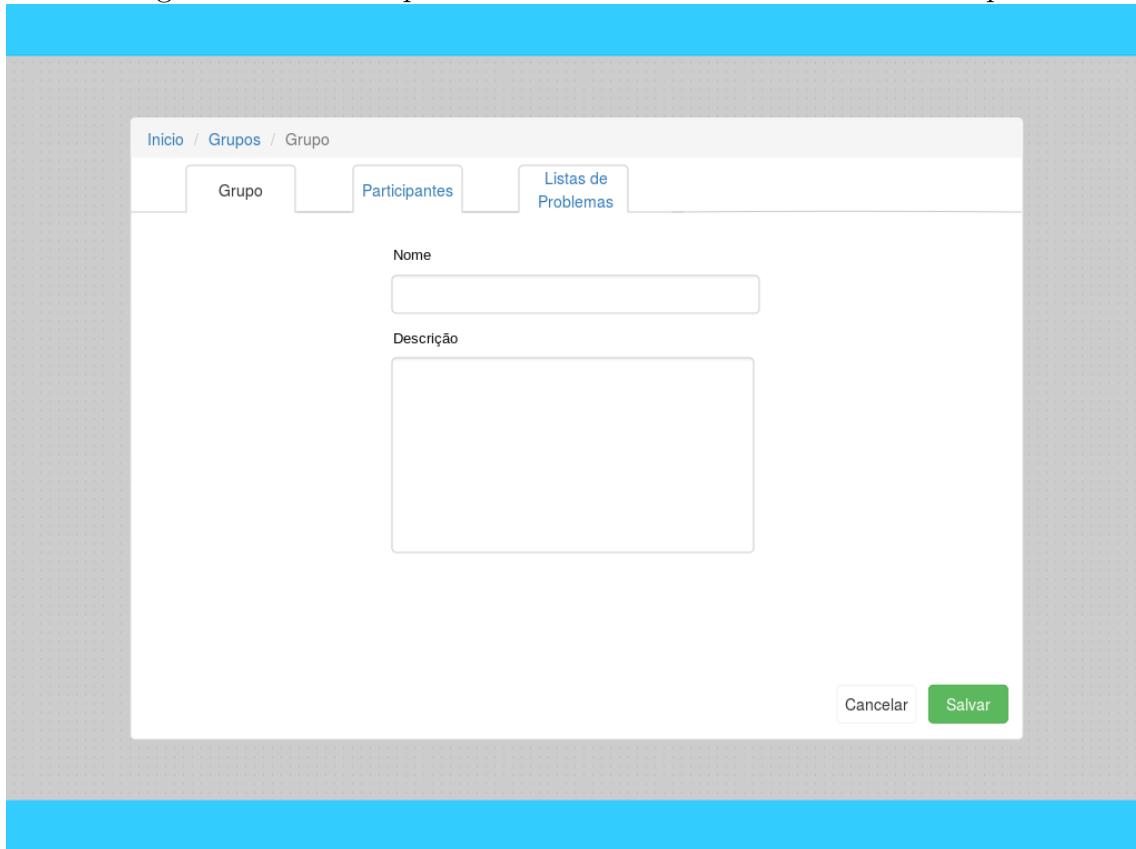
Nessa interface gráfica possuímos as seguintes ações:

- Novo Grupo
- Editar Grupo
- Remover Grupo
- Remover

4.4.13 Cadastro e Edição de Grupo

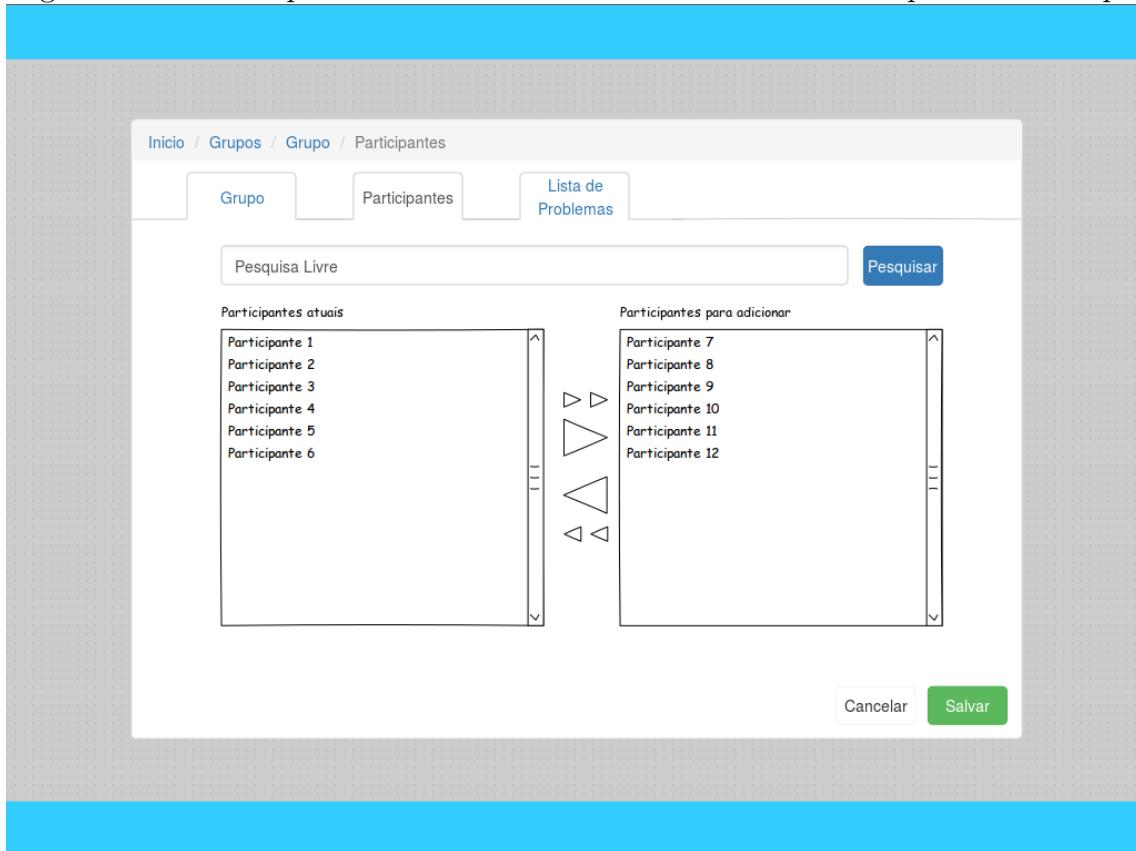
A figura 4.36 é o protótipo de interface gráfica de

Figura 4.36: Protótipo de Inteface Gráfica de Cadastro de Grupo



4.4.14 Adicionar Participantes

Figura 4.37: Protótipo de Inteface Gráfica de Cadastro de Participantes no Grupo



4.4.15 Adicionar Lista de Problemas

Figura 4.38: Protótipo de Inteface Gráfica de Cadastro de Lista de Problemas no Grupo

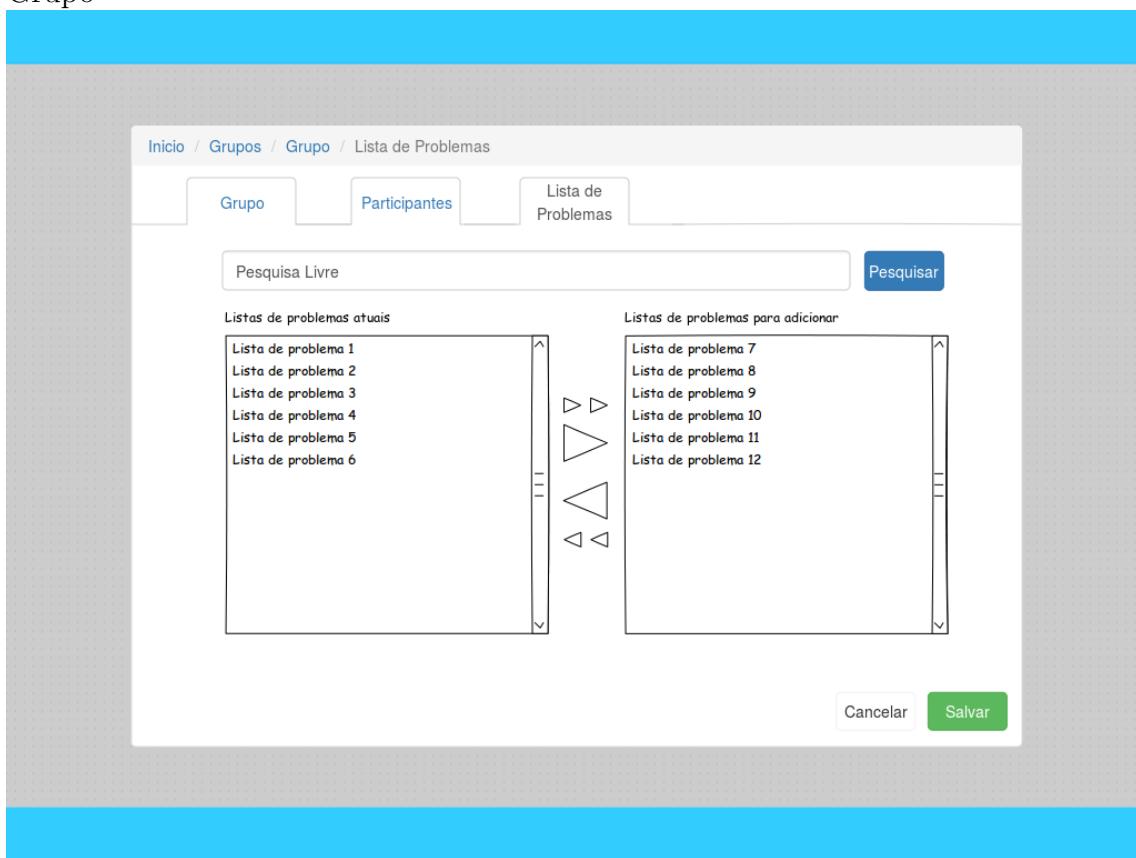


Figura 4.39: Mockups

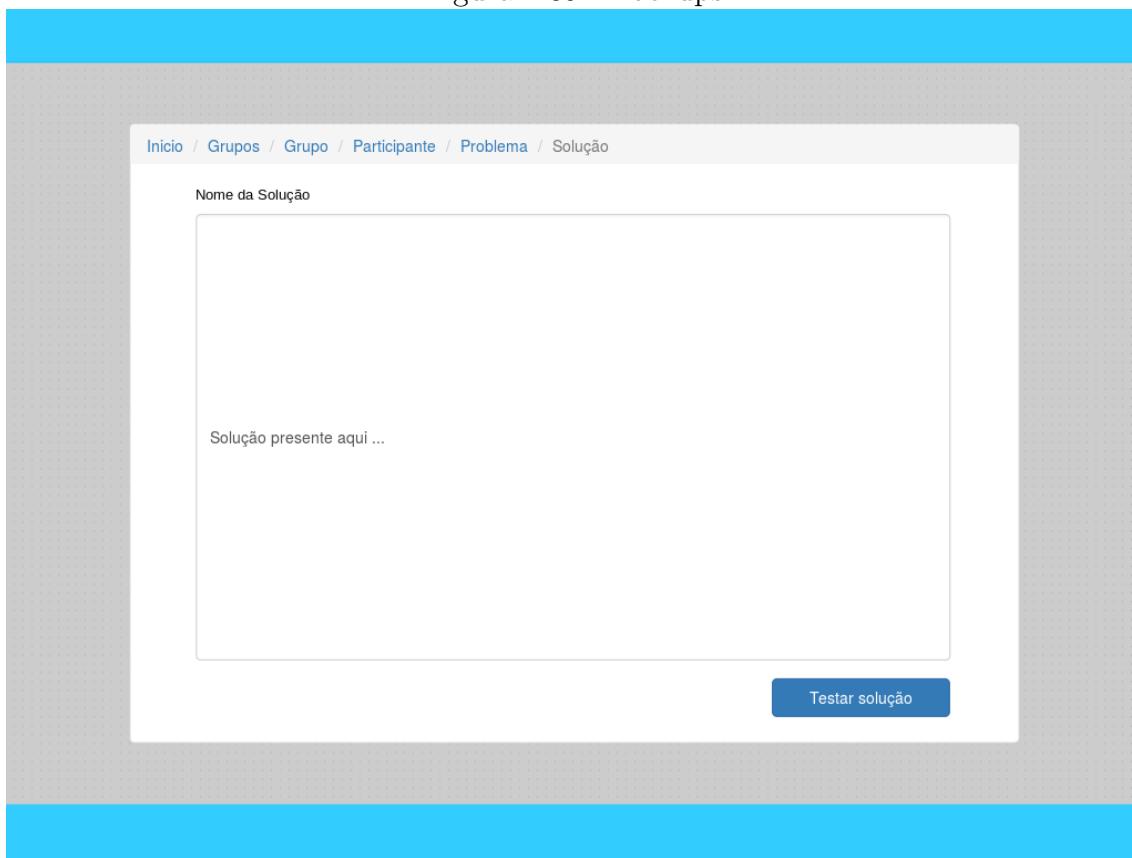


Figura 4.40: Mockups

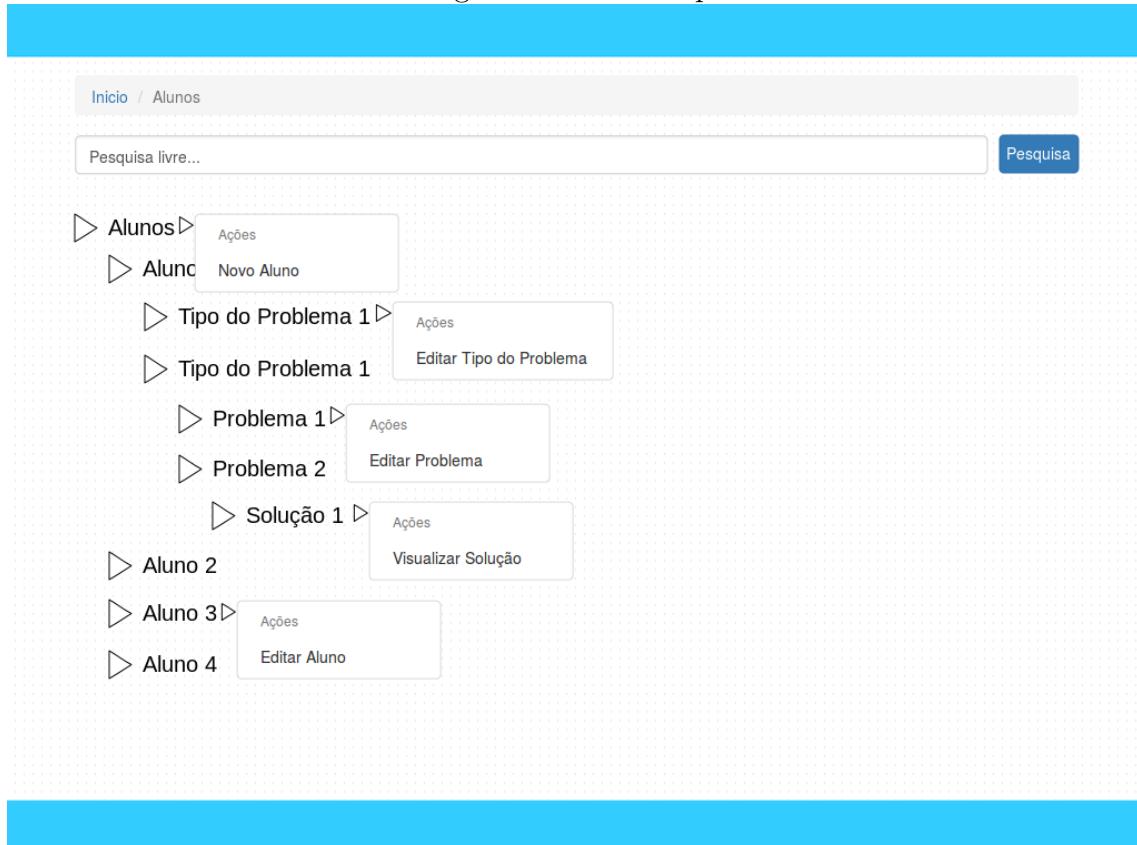


Figura 4.41: Mockups

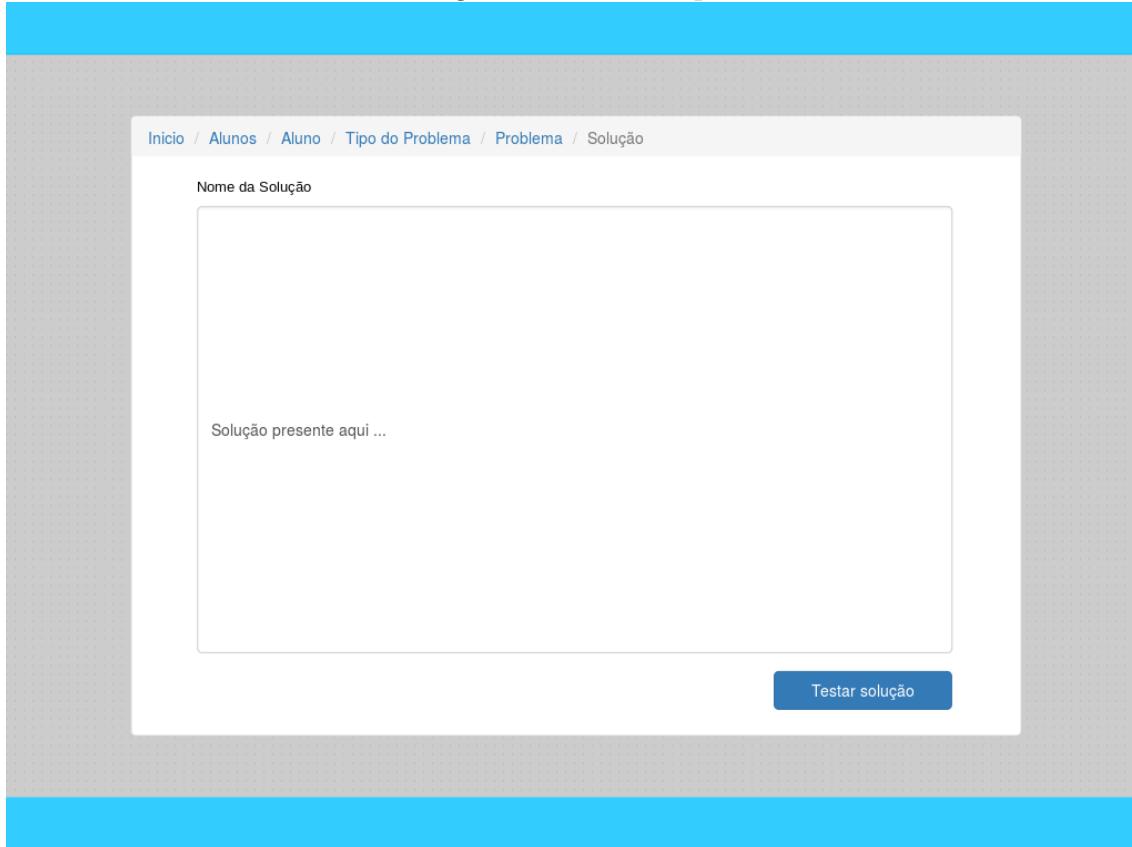


Figura 4.42: Mockups

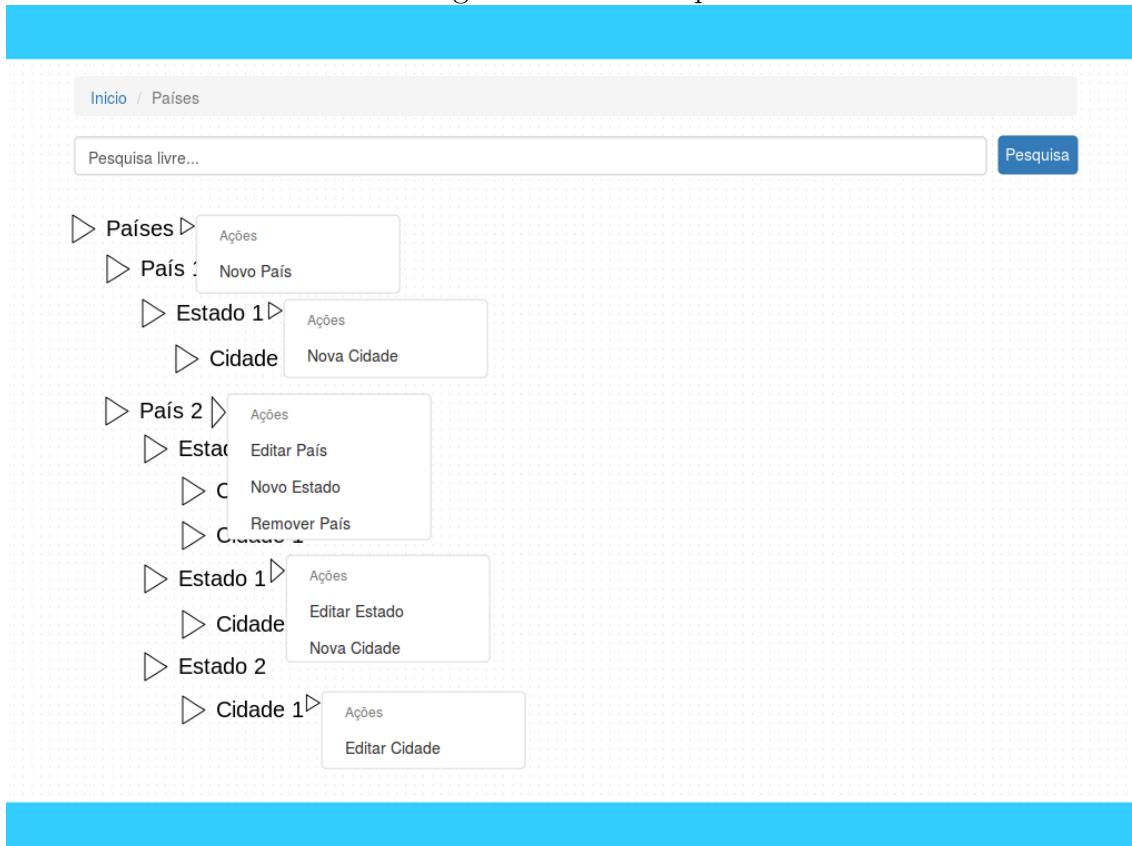


Figura 4.43: Mockups

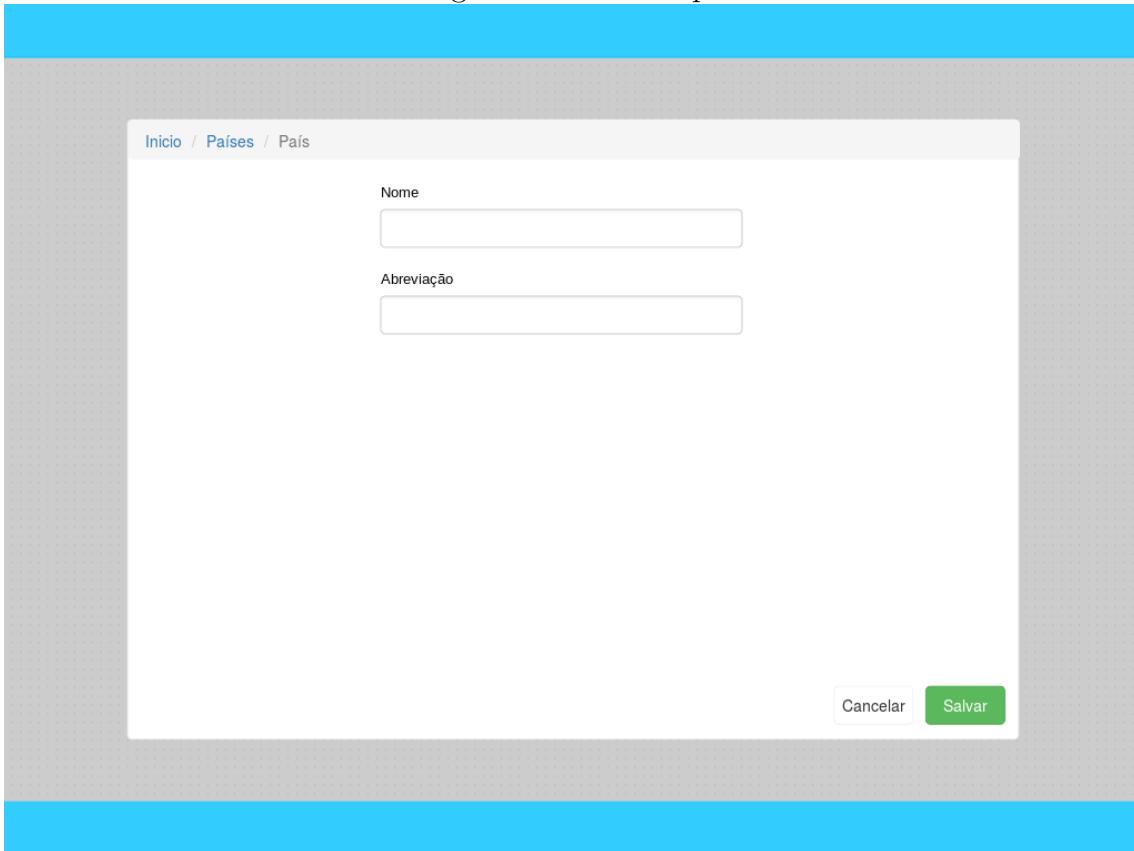


Figura 4.44: Mockups

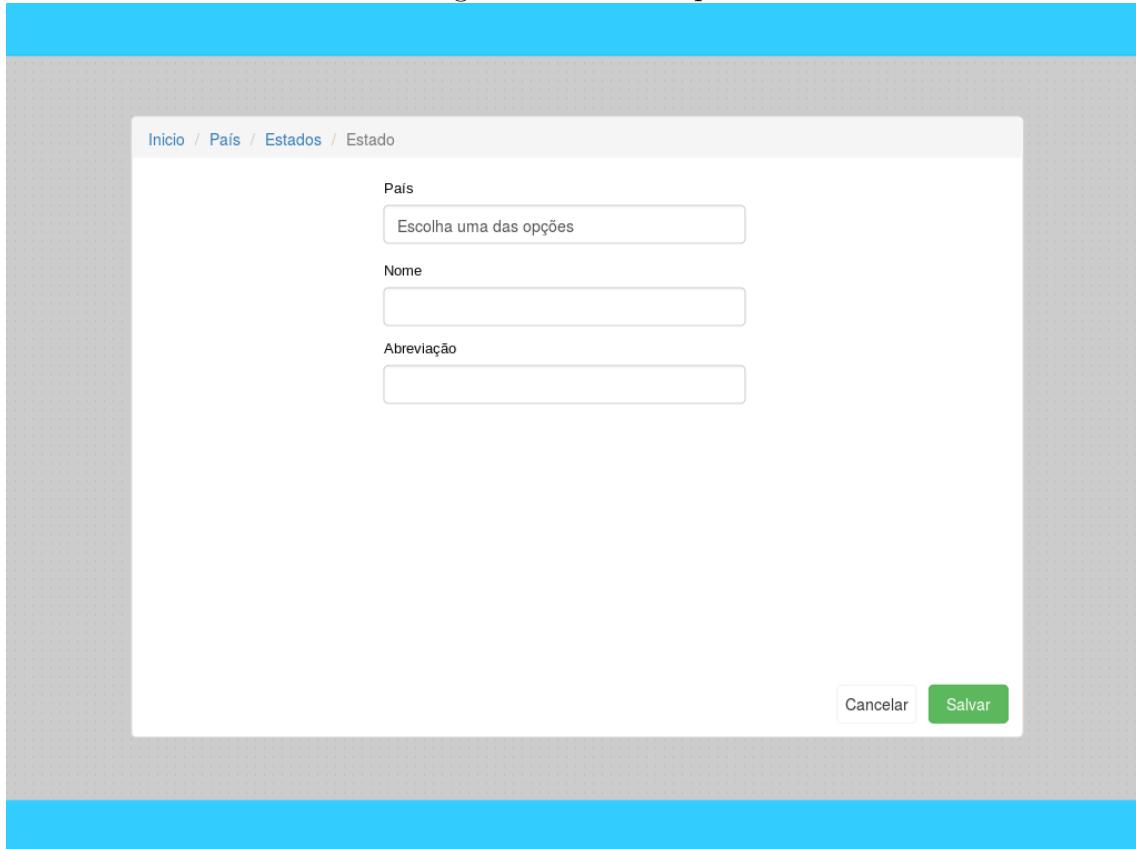


Figura 4.45: Mockups

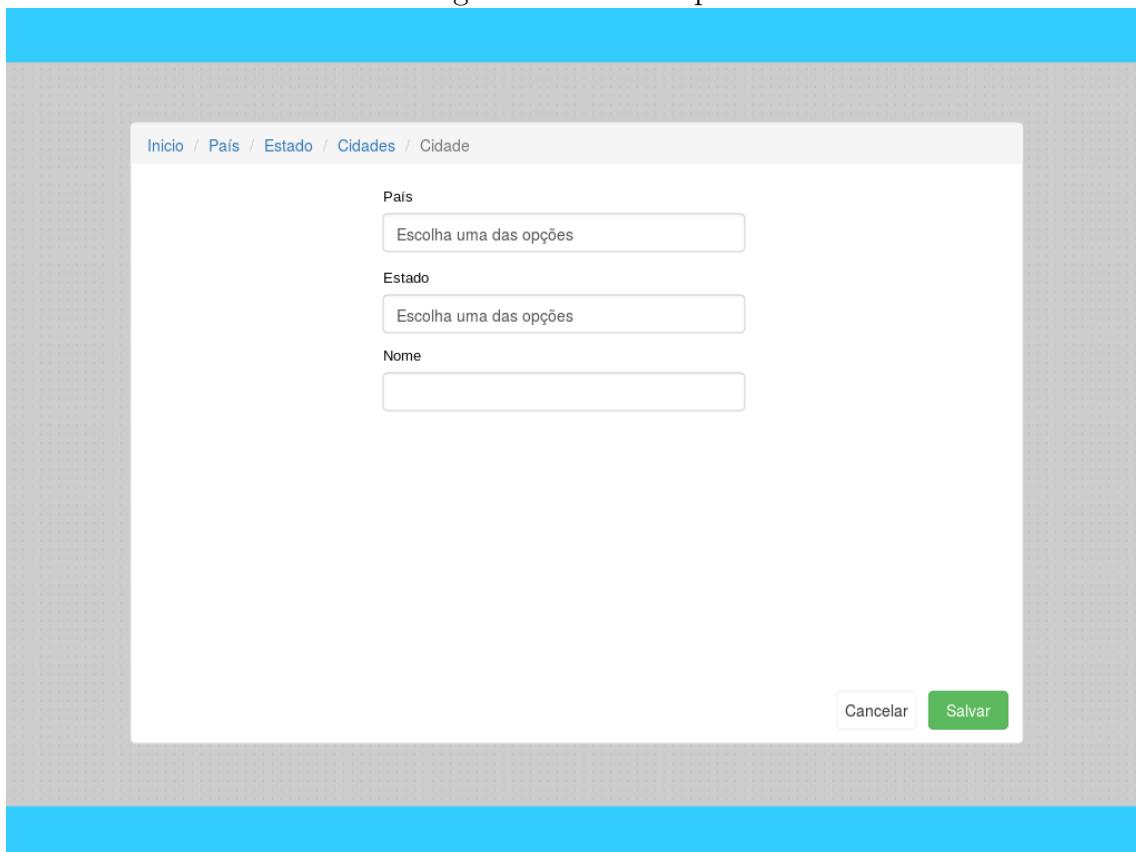


Figura 4.46: Mockups

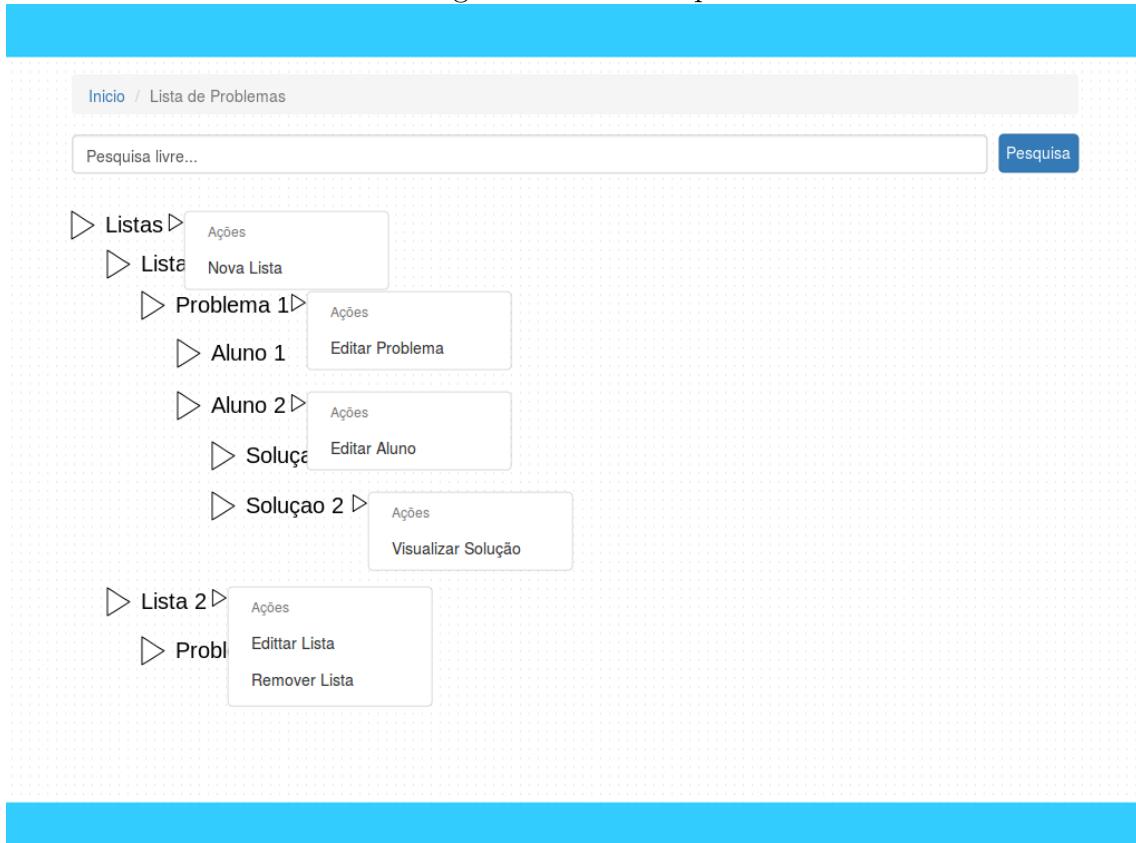


Figura 4.47: Mockups

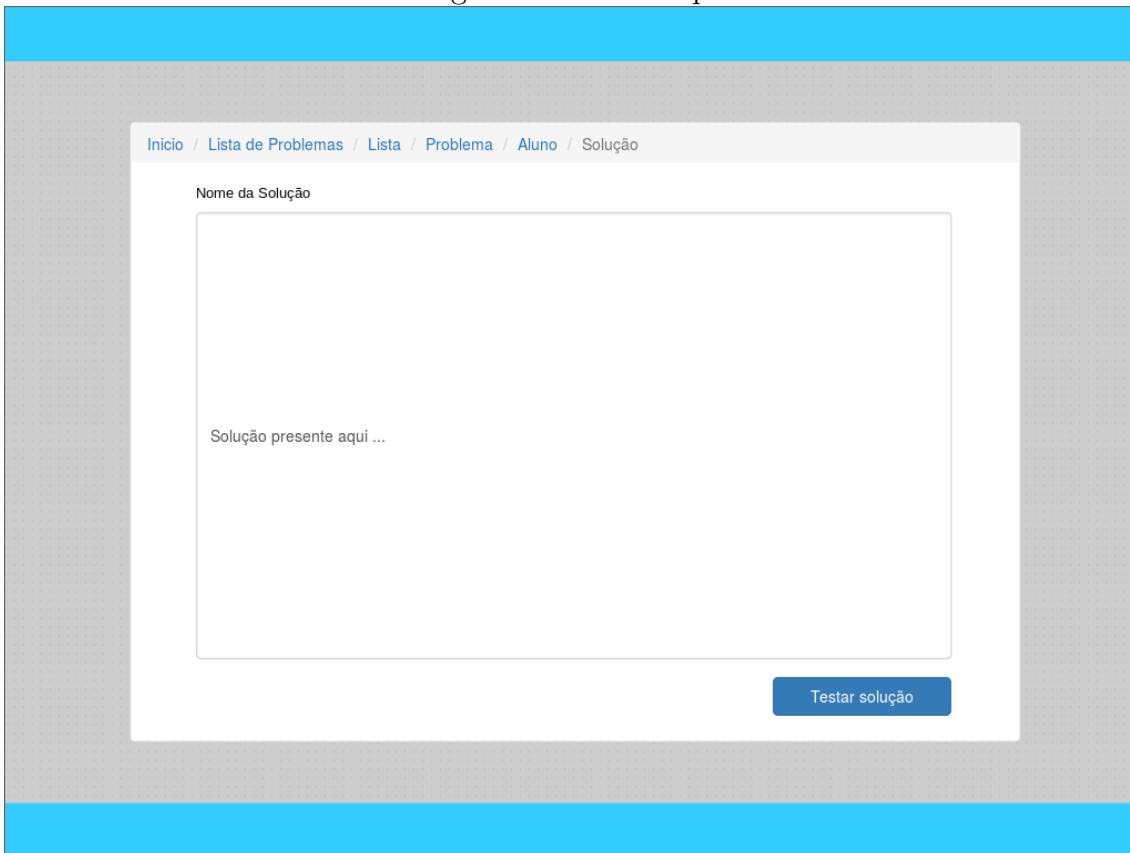


Figura 4.48: Mockups

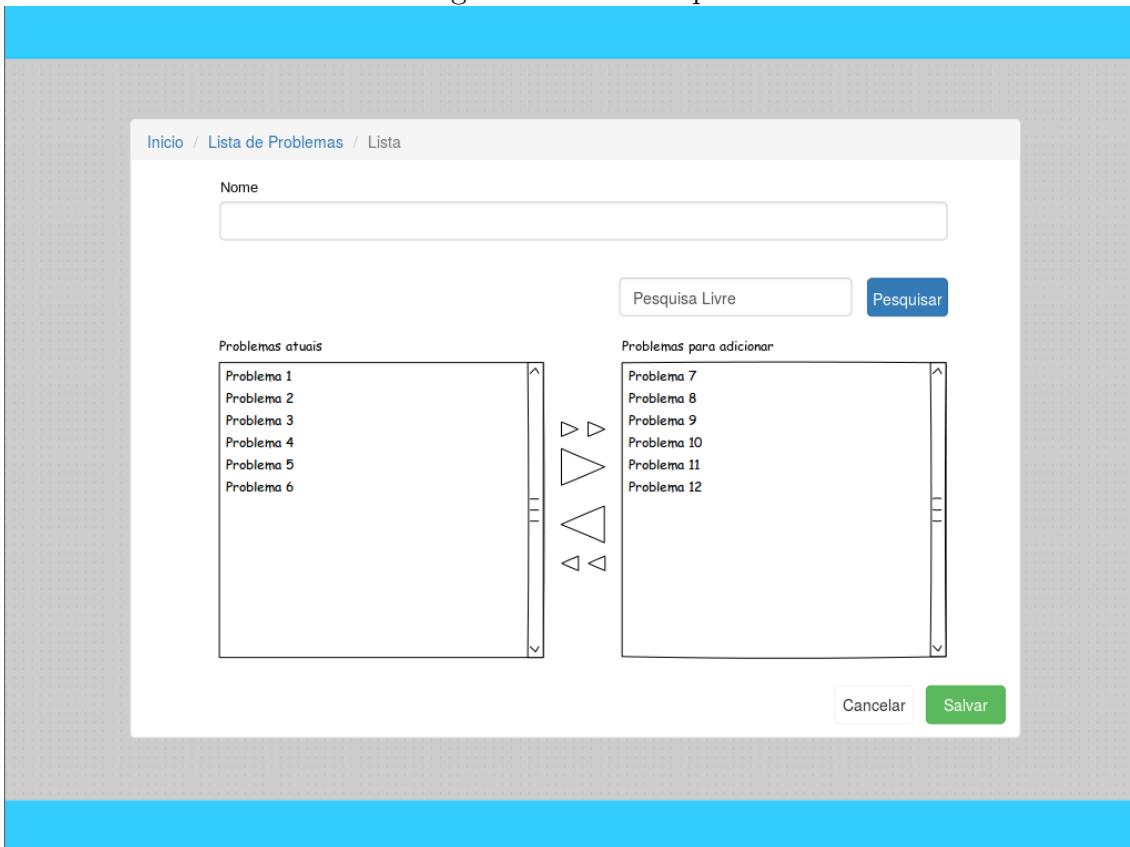


Figura 4.49: Mockups

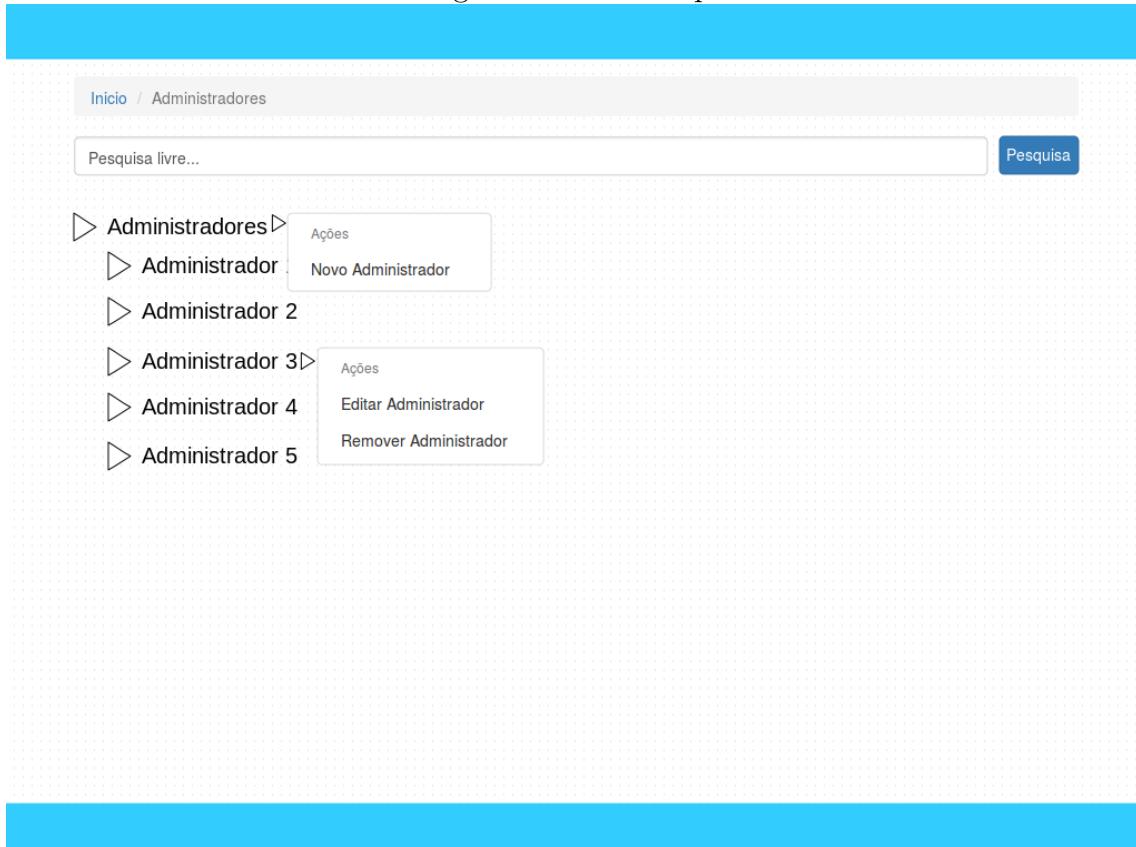


Figura 4.50: Mockups

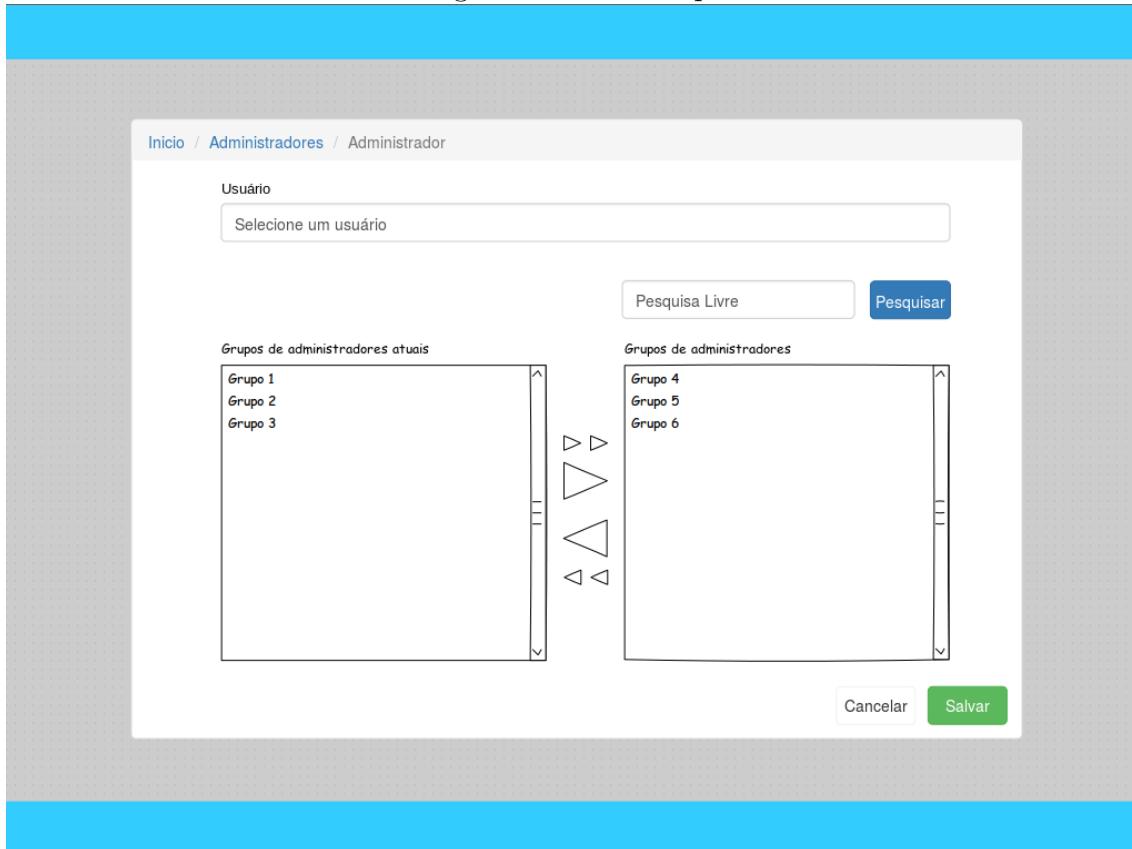


Figura 4.51: Mockups

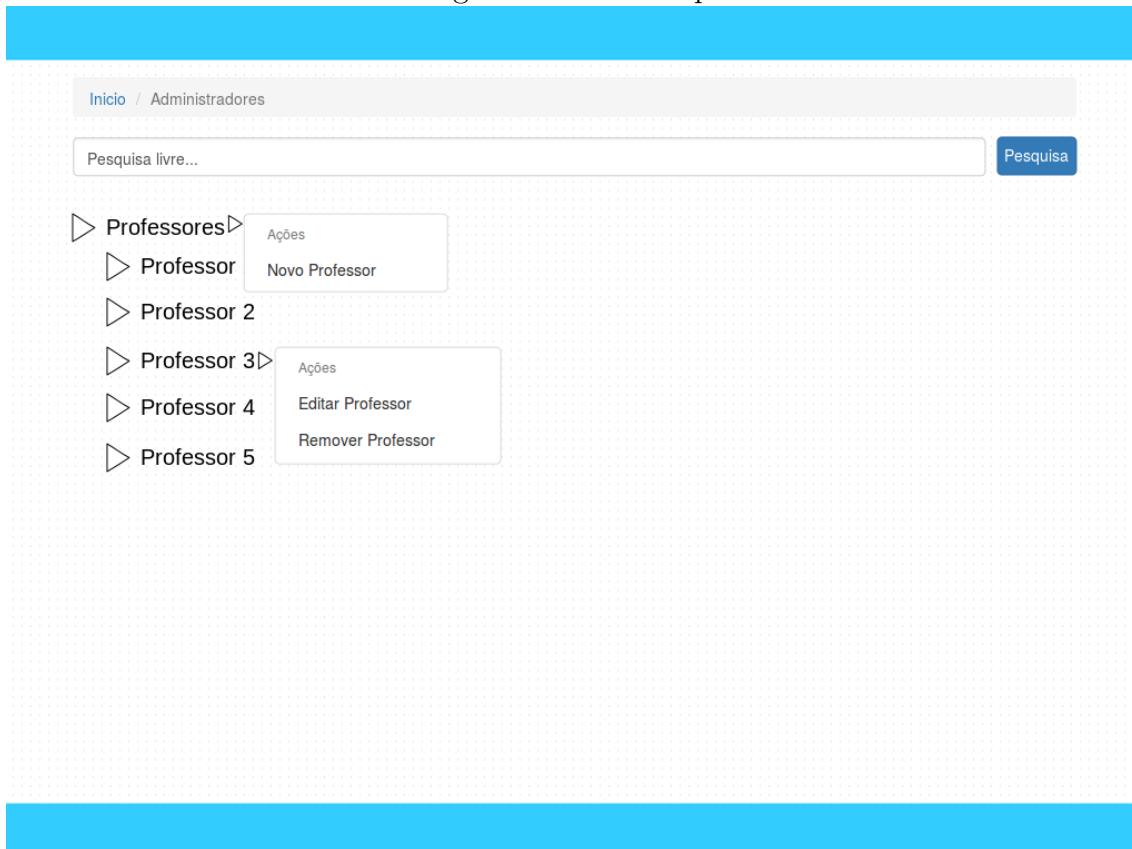


Figura 4.52: Mockups

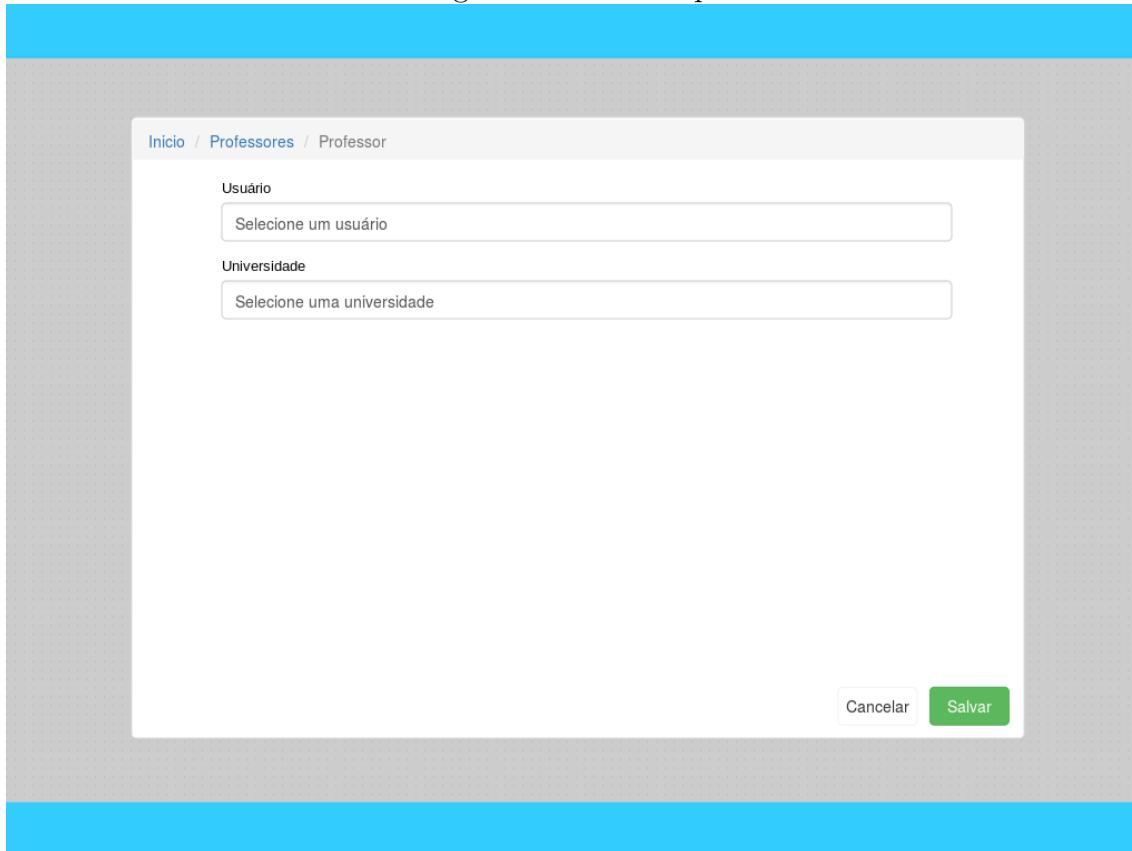


Figura 4.53: Mockups

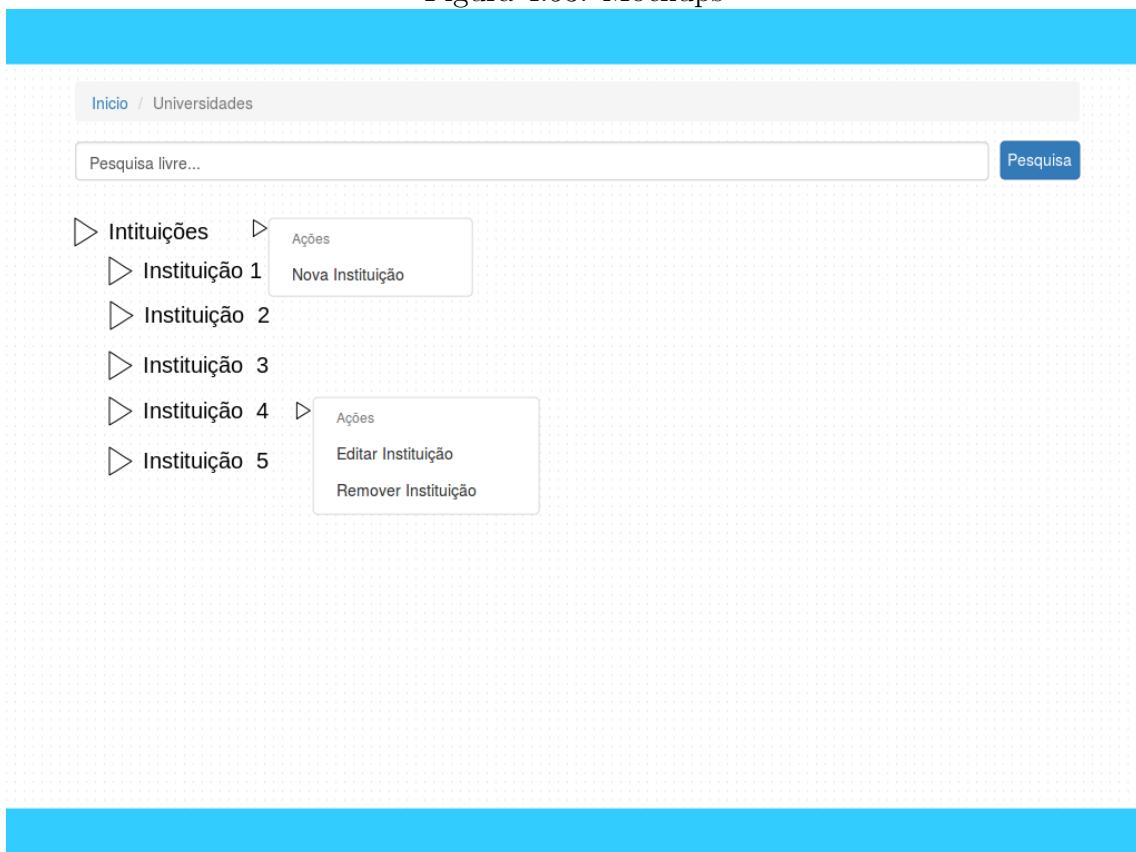


Figura 4.54: Mockups

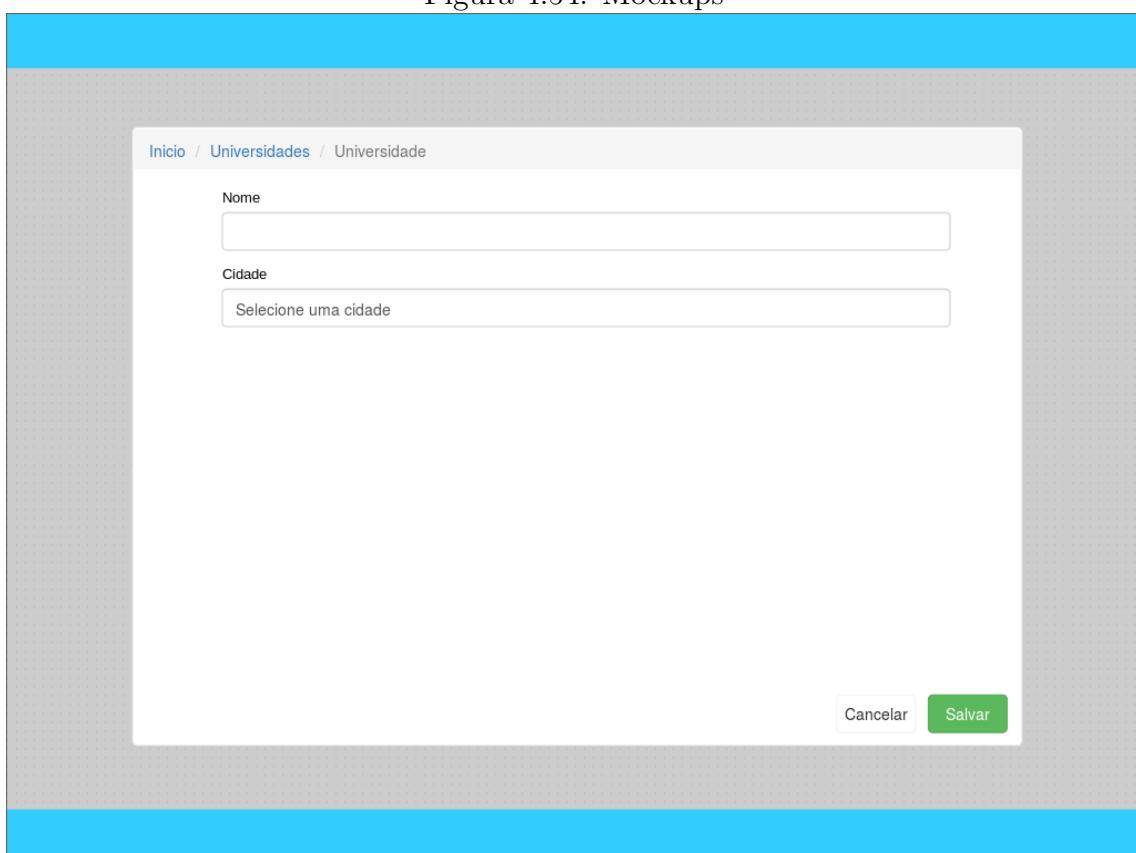


Figura 4.55: Mockups

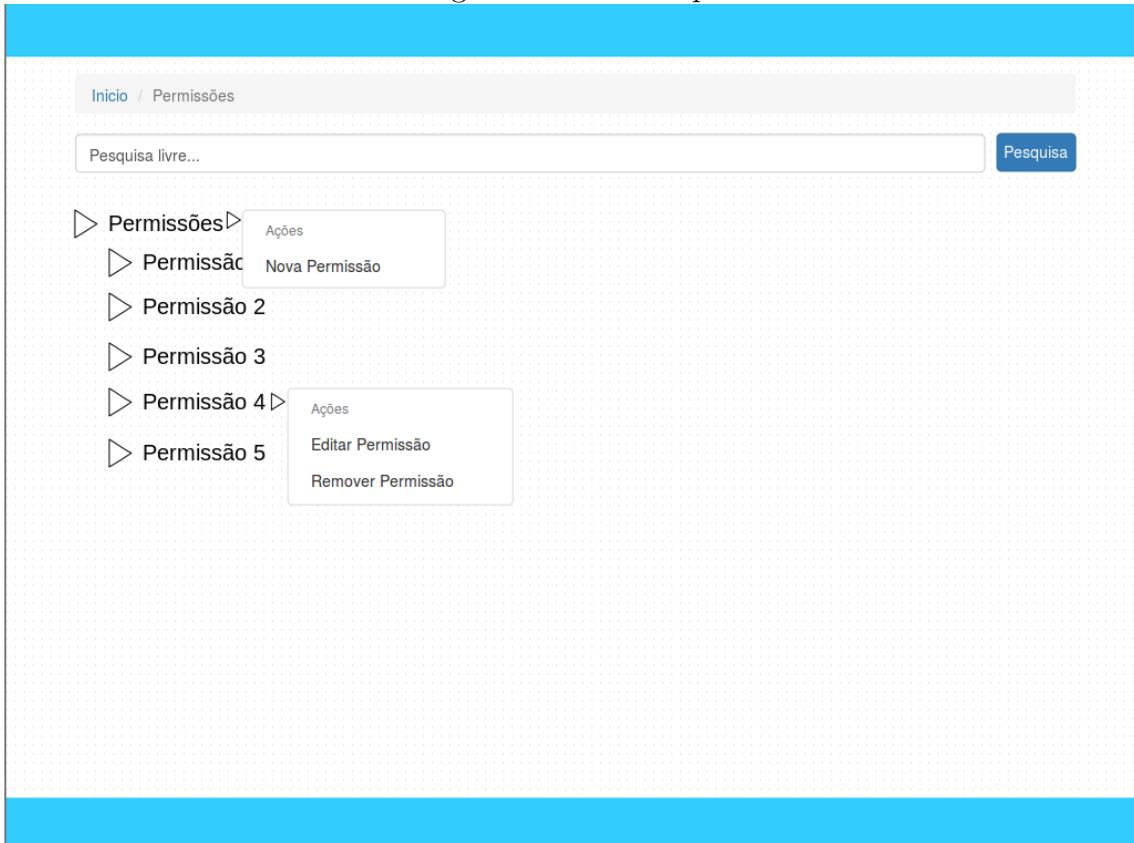


Figura 4.56: Mockups

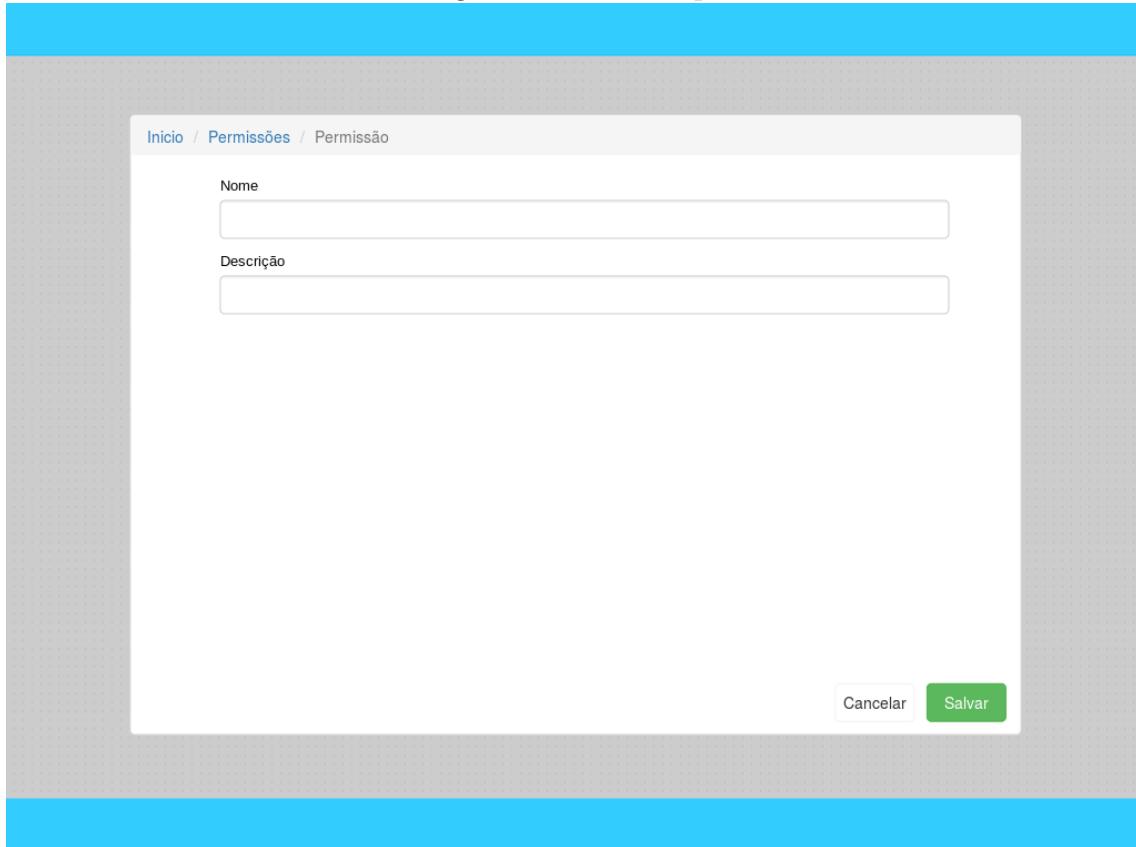


Figura 4.57: Mockups

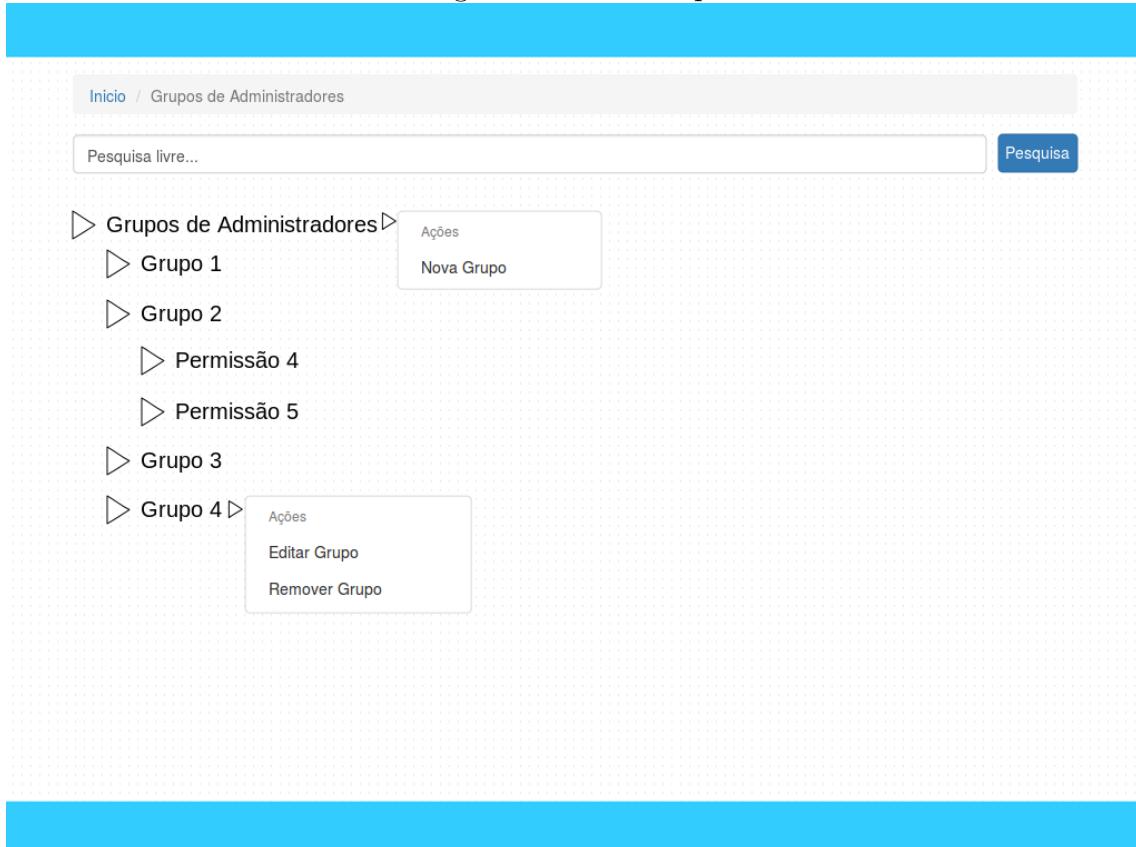


Figura 4.58: Mockups

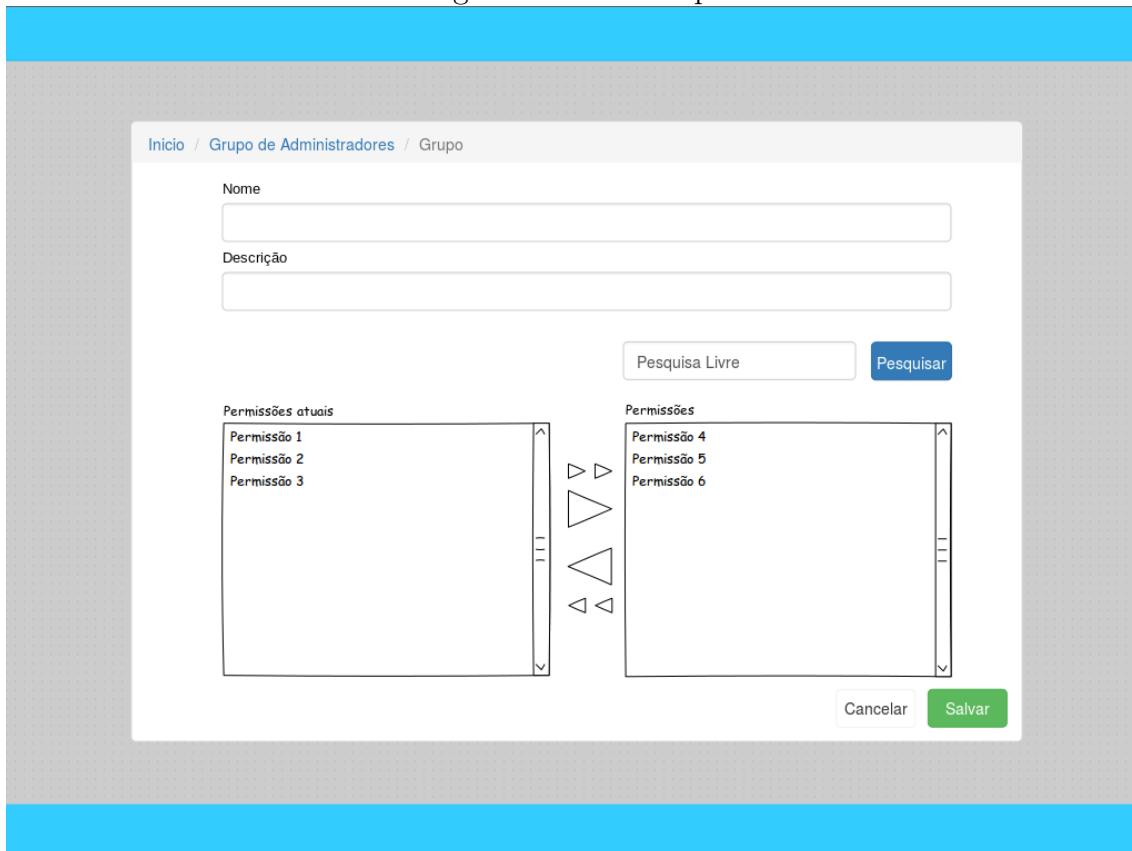


Figura 4.59: Mockups

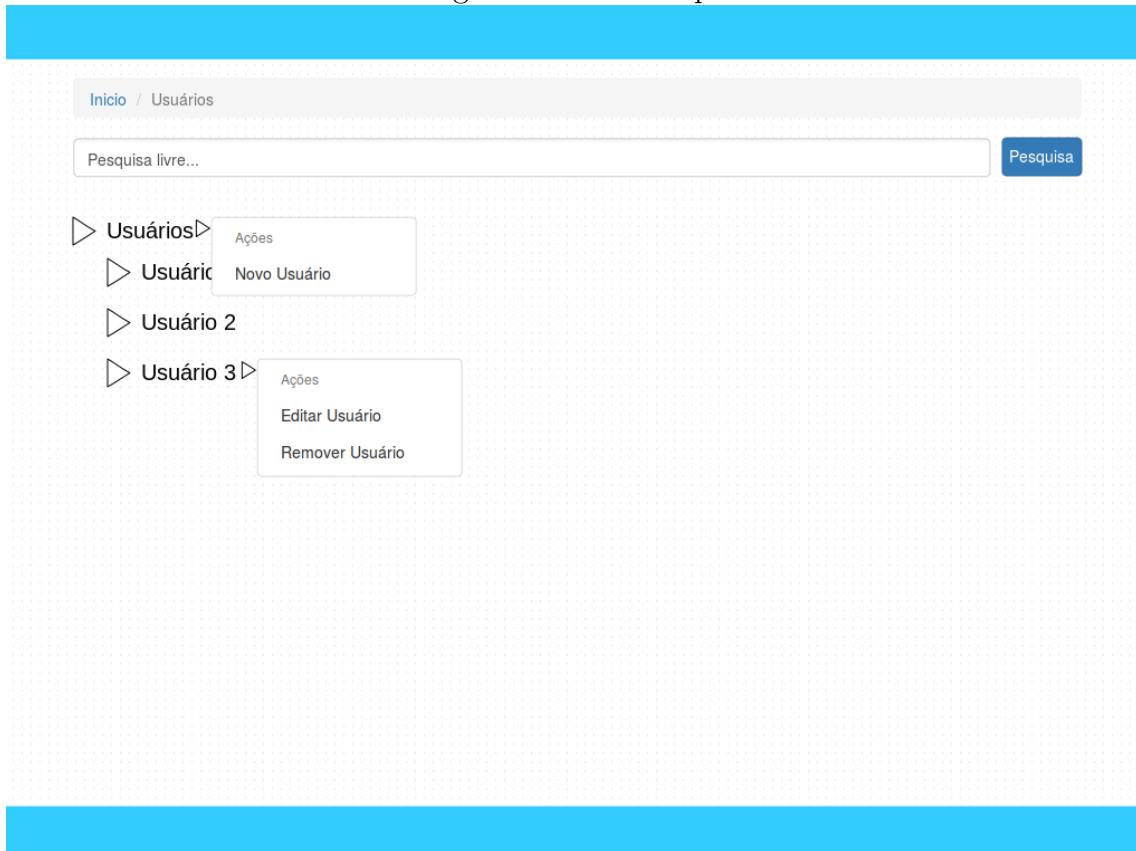
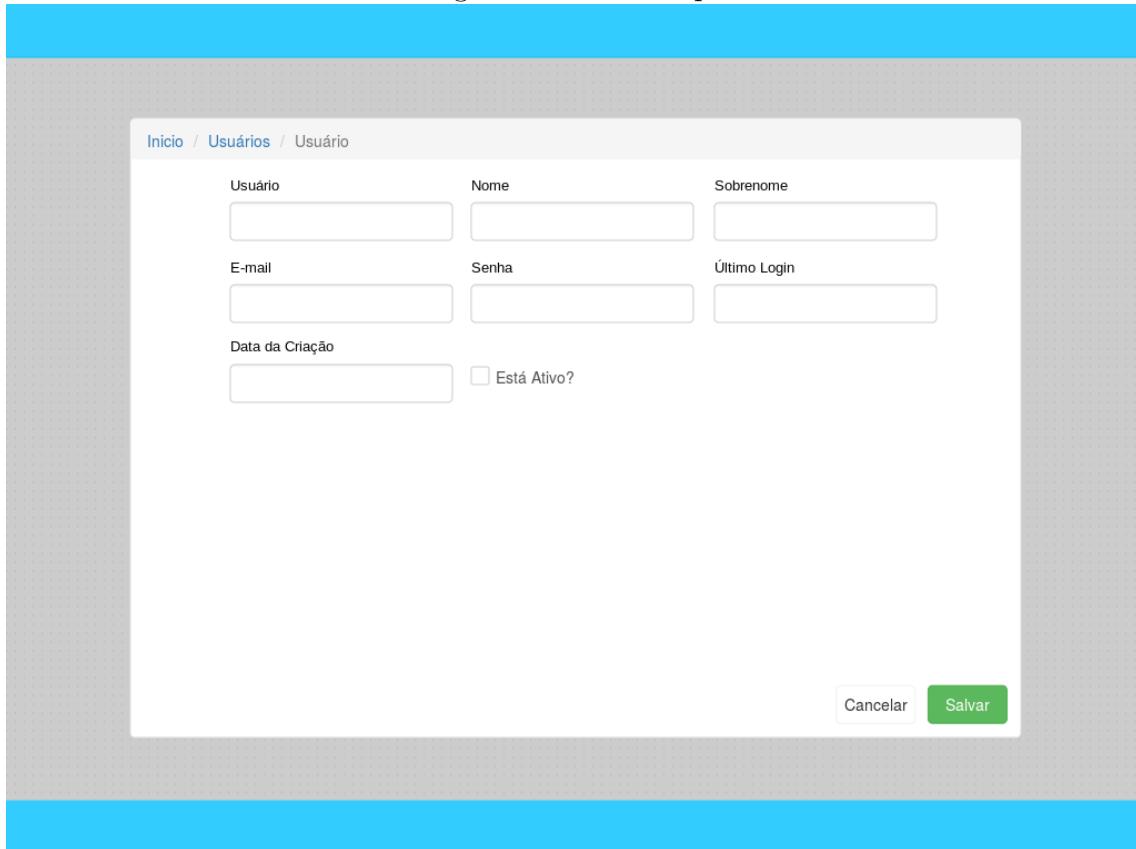


Figura 4.60: Mockups



4.5 Diagramas de Robustez

Figura 4.61: Robustez

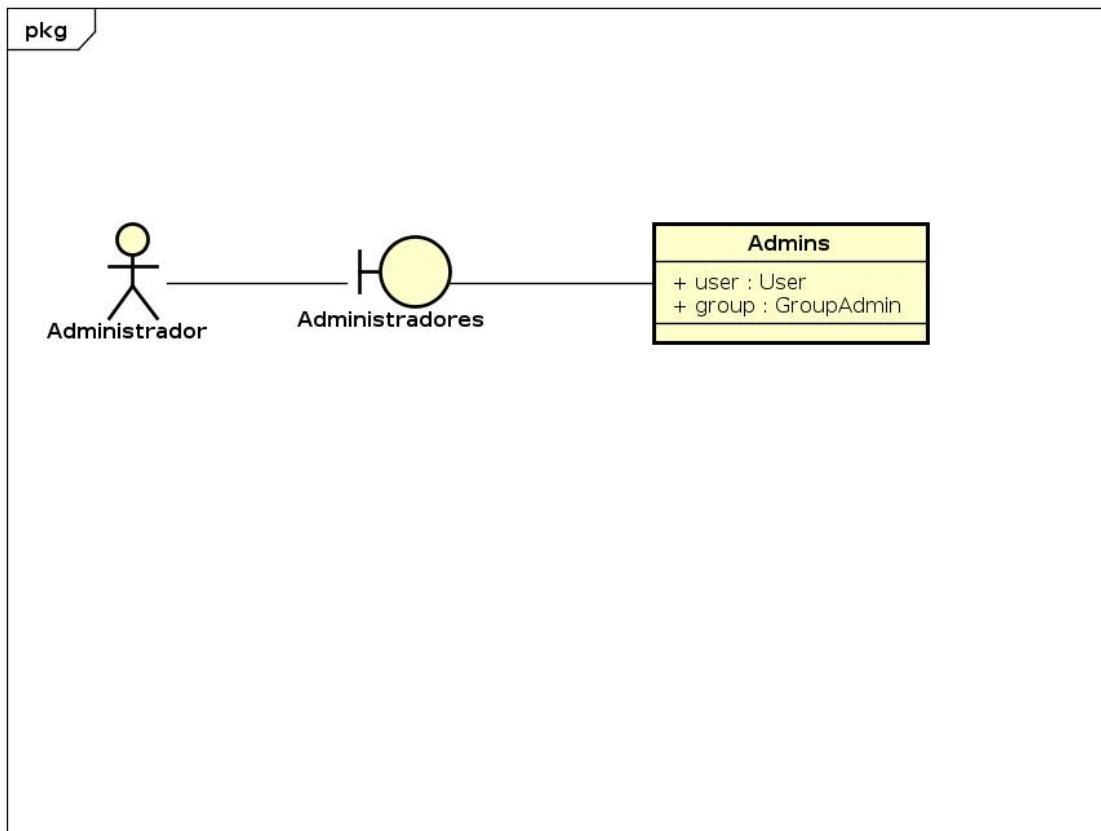


Figura 4.62: Robustez

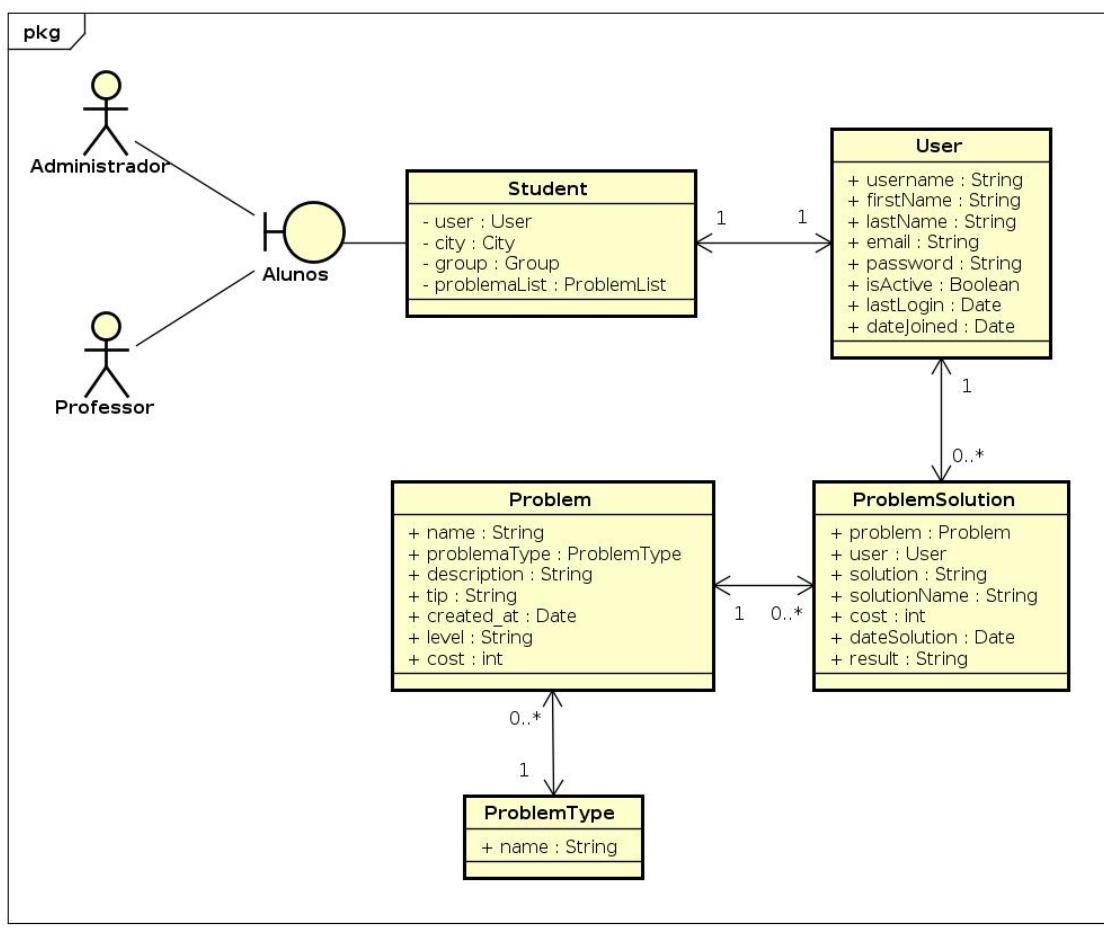


Figura 4.63: Robustez

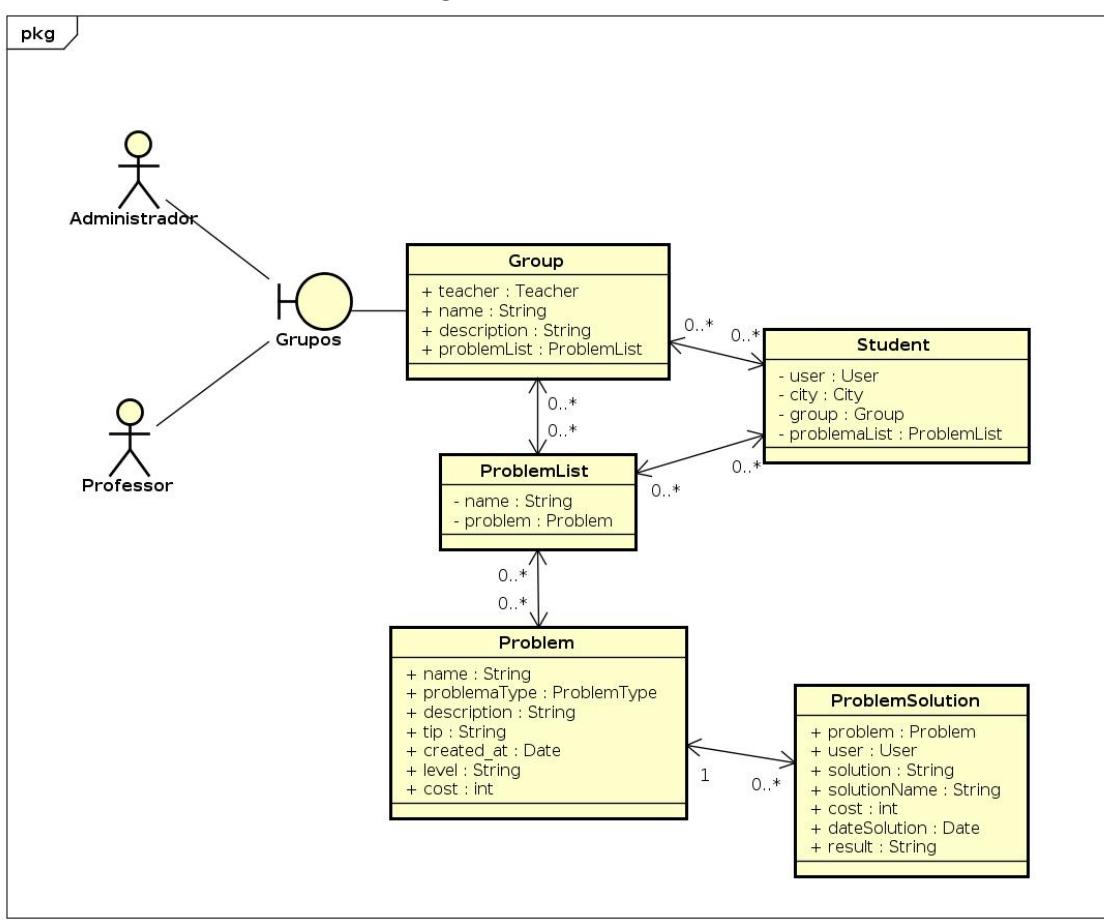


Figura 4.64: Robustez

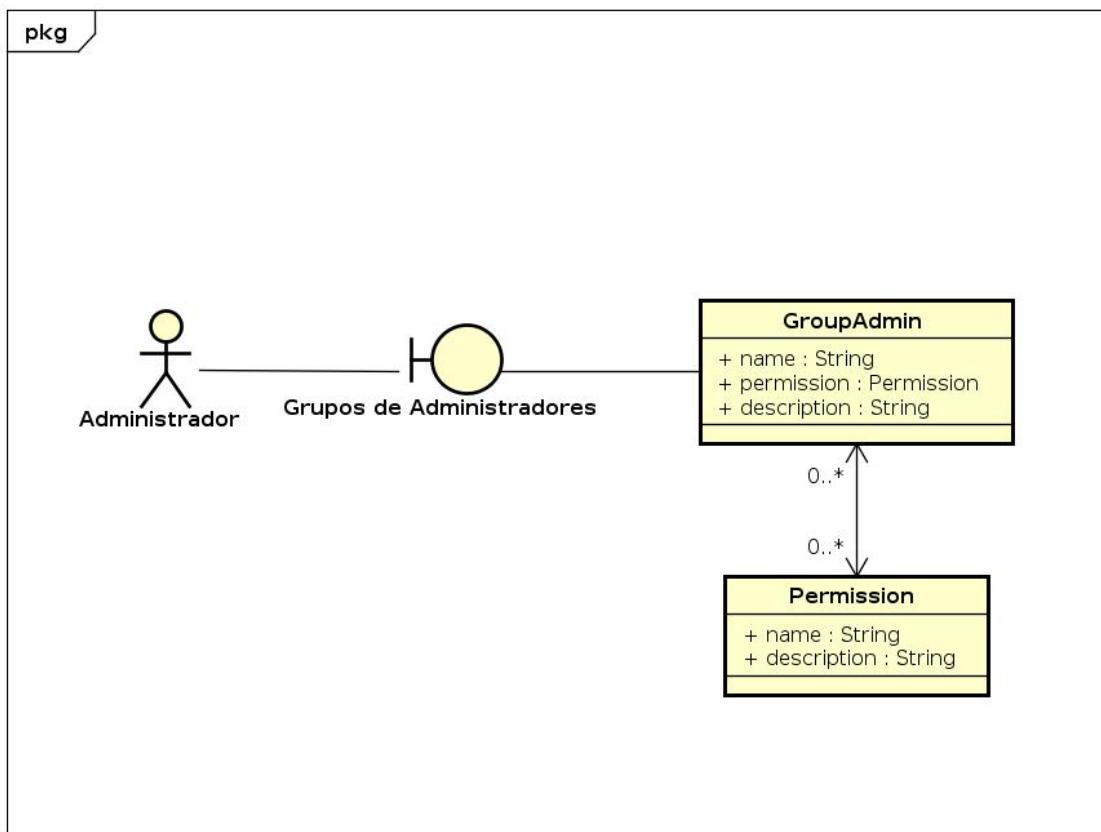


Figura 4.65: Robustez

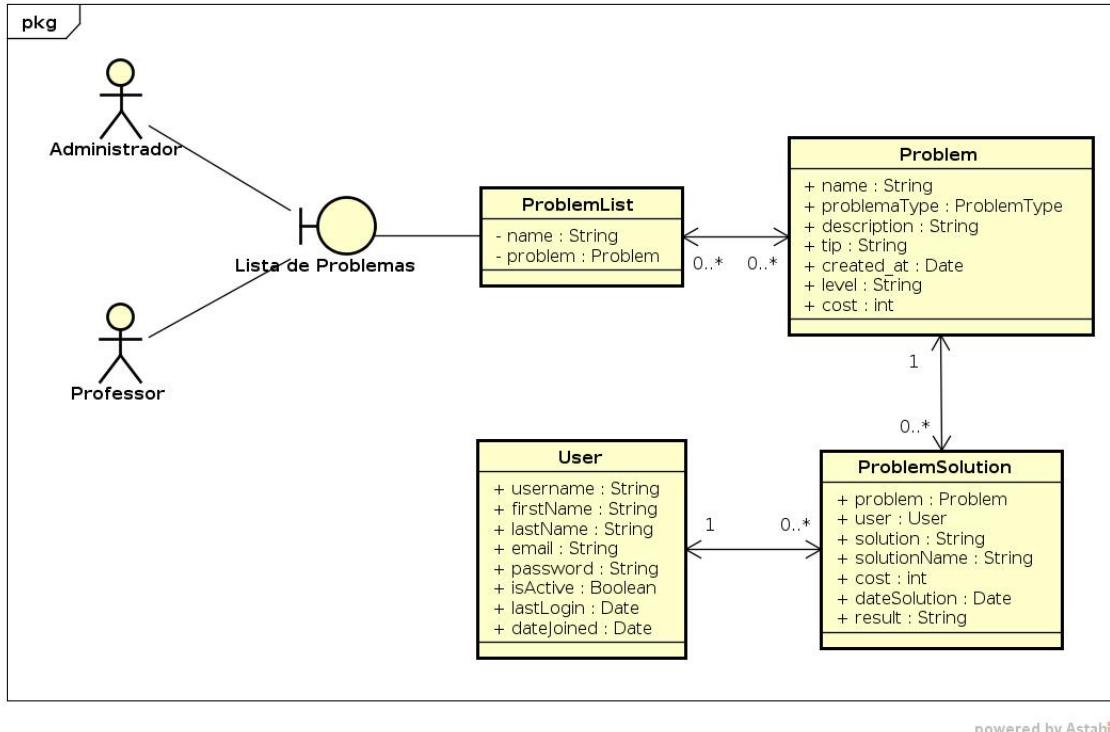


Figura 4.66: Robustez

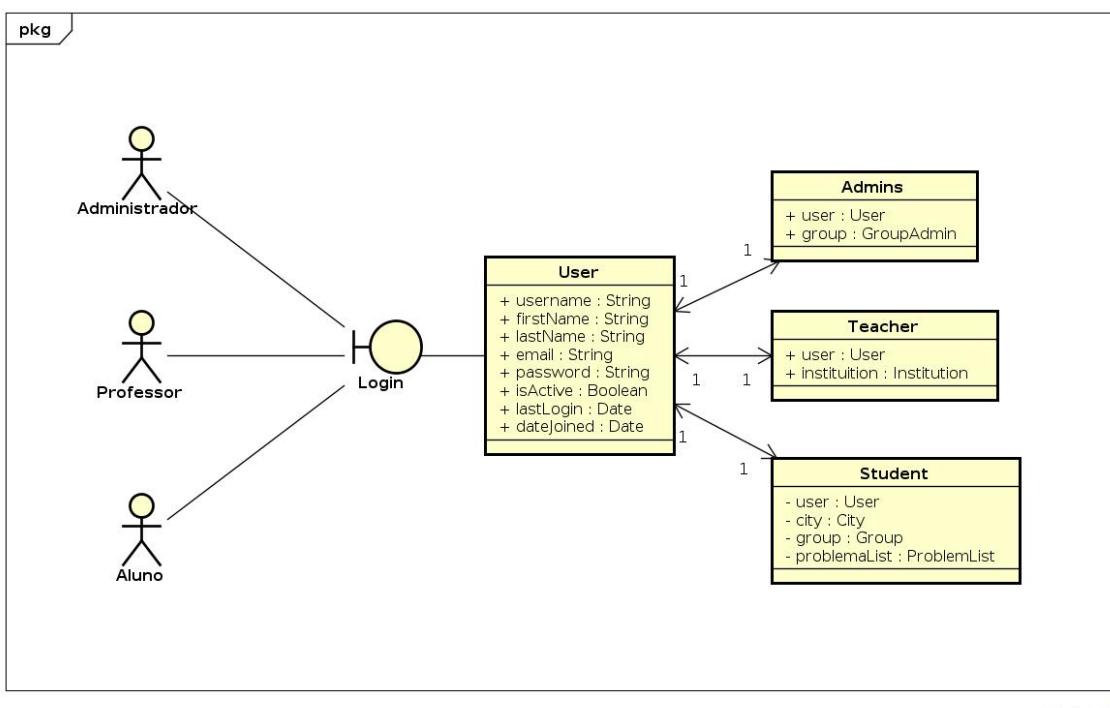


Figura 4.67: Robustez

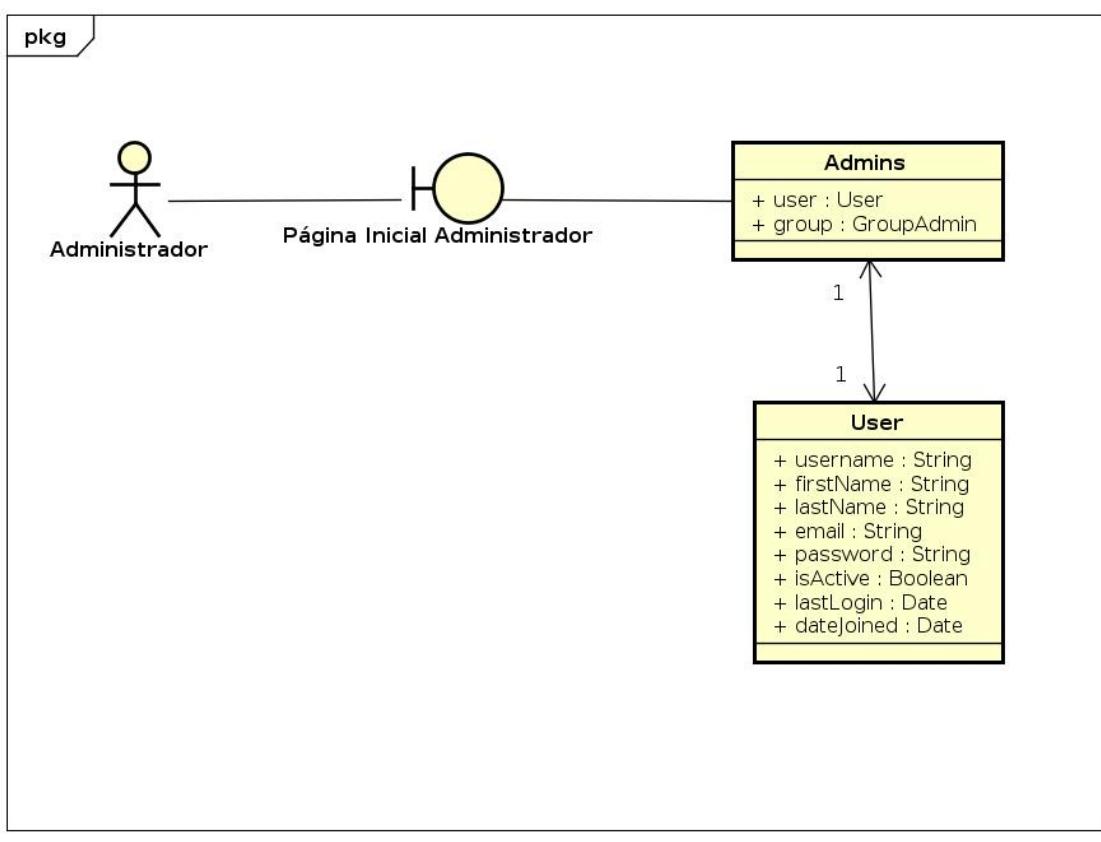


Figura 4.68: Robustez

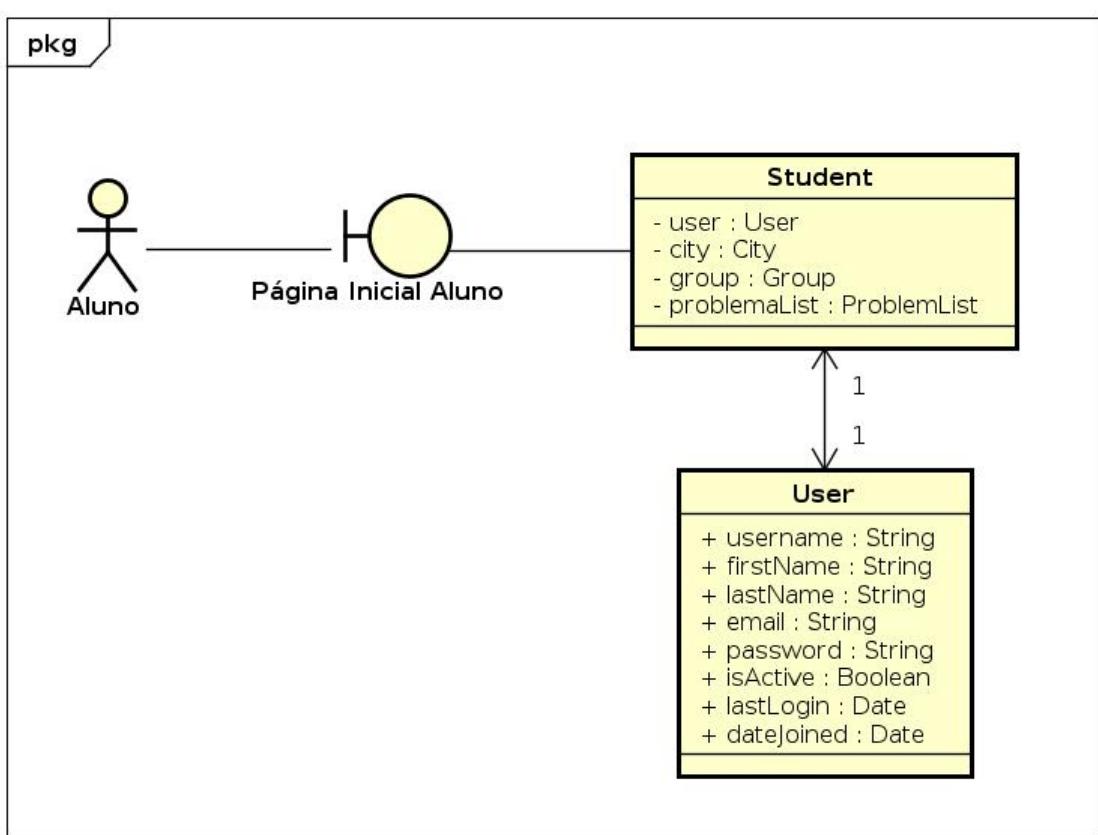
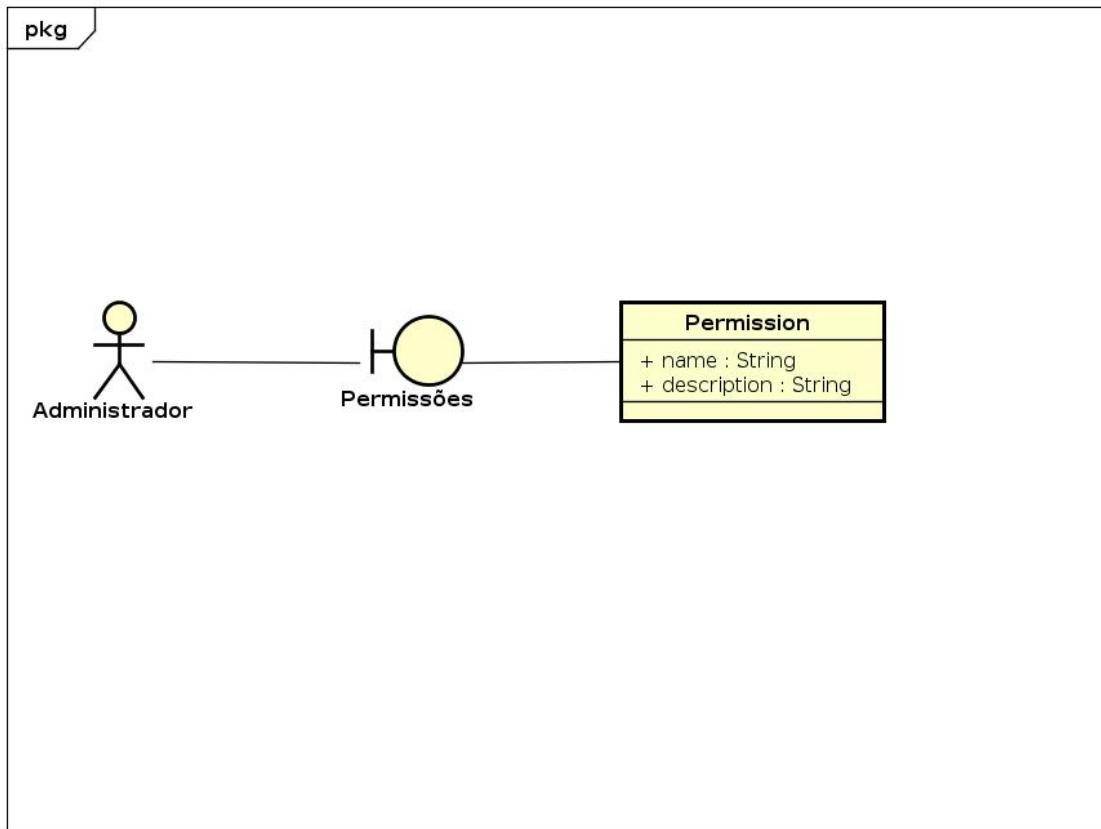
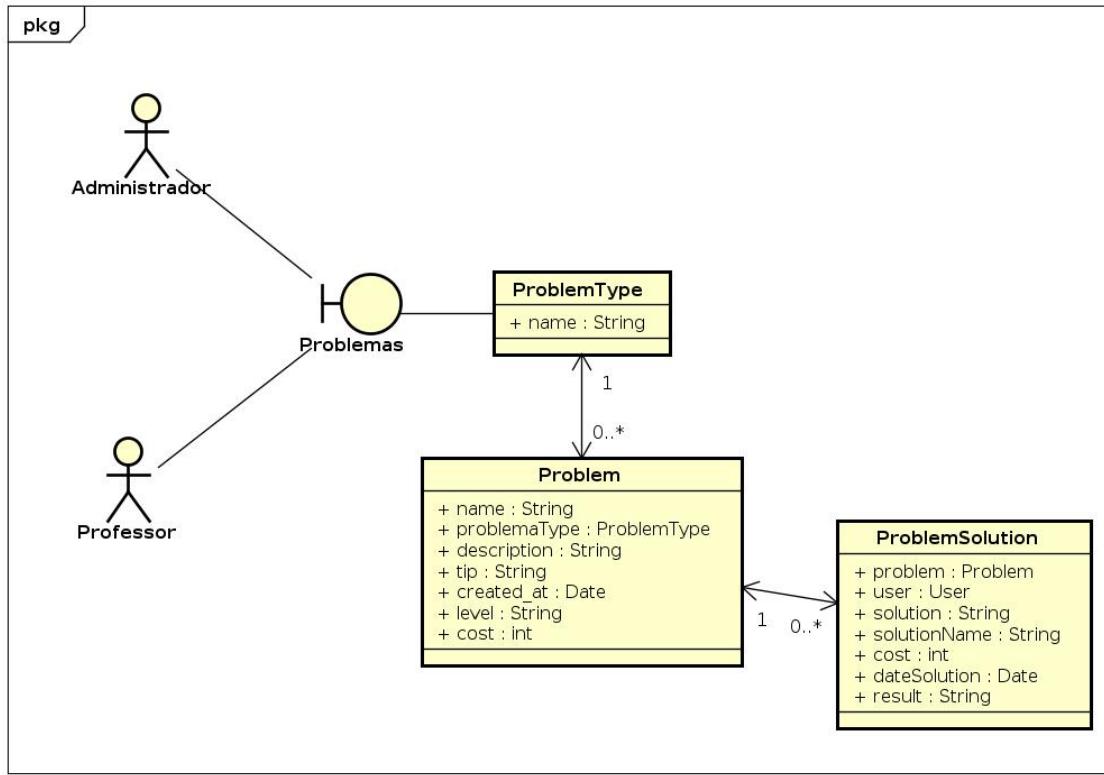


Figura 4.69: Robustez



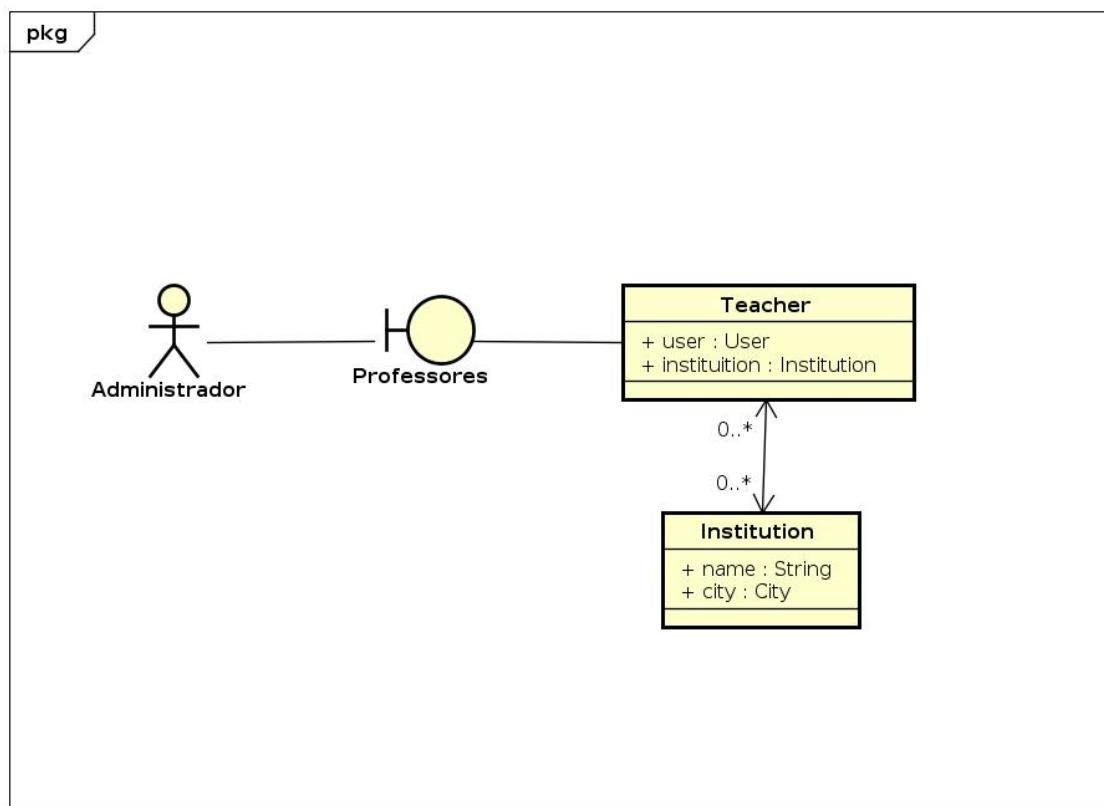
powered by Astah

Figura 4.70: Robustez



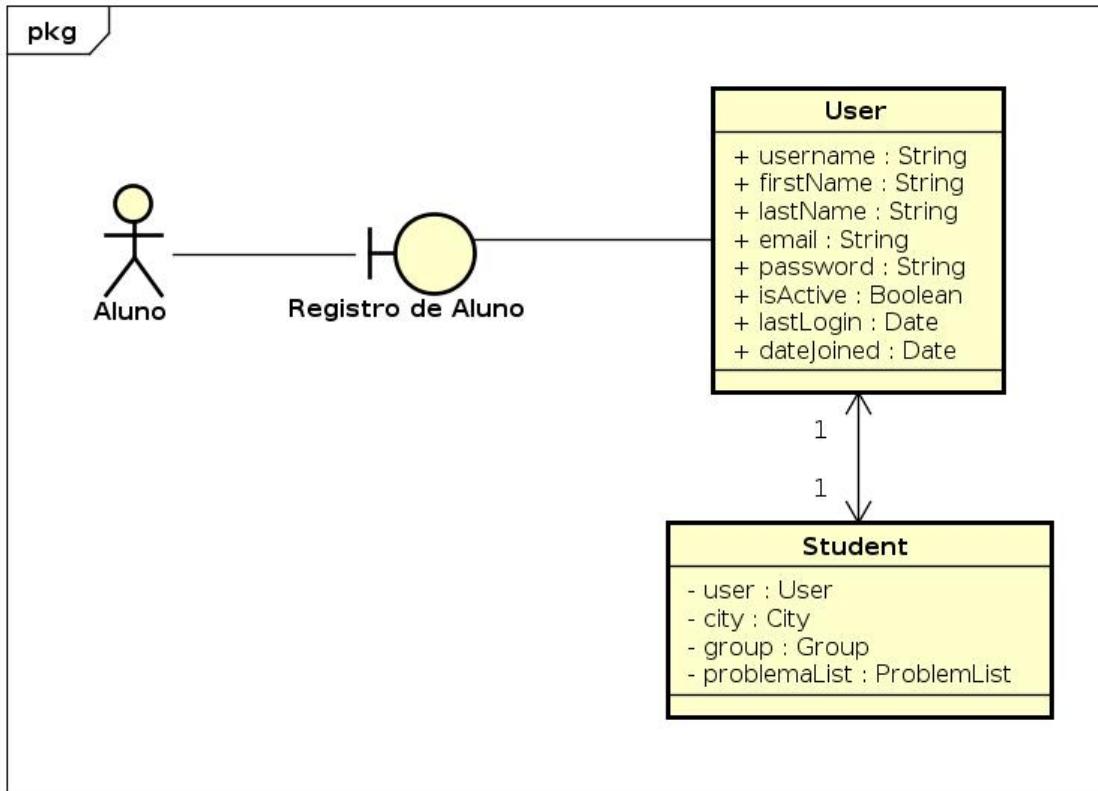
powered by Astah

Figura 4.71: Robustez



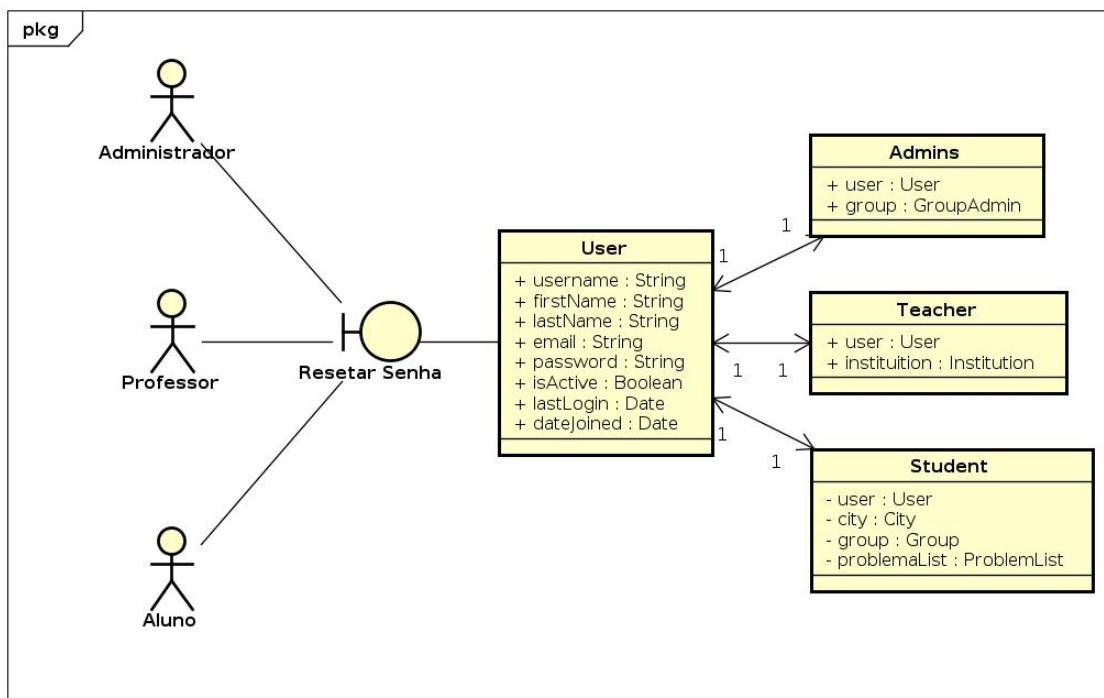
powered by Astah

Figura 4.72: Robustez



powered by Astah

Figura 4.73: Robustez



powered by Astah

Figura 4.74: Robustez

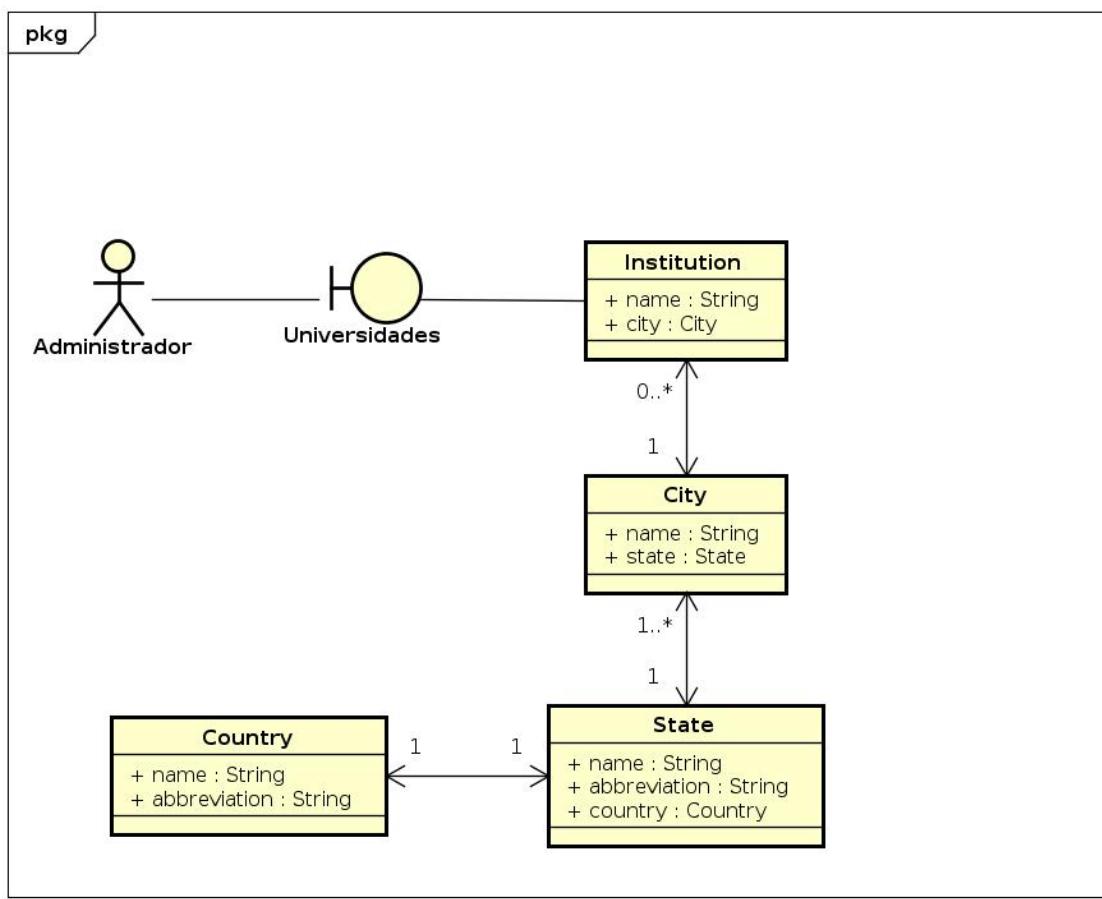
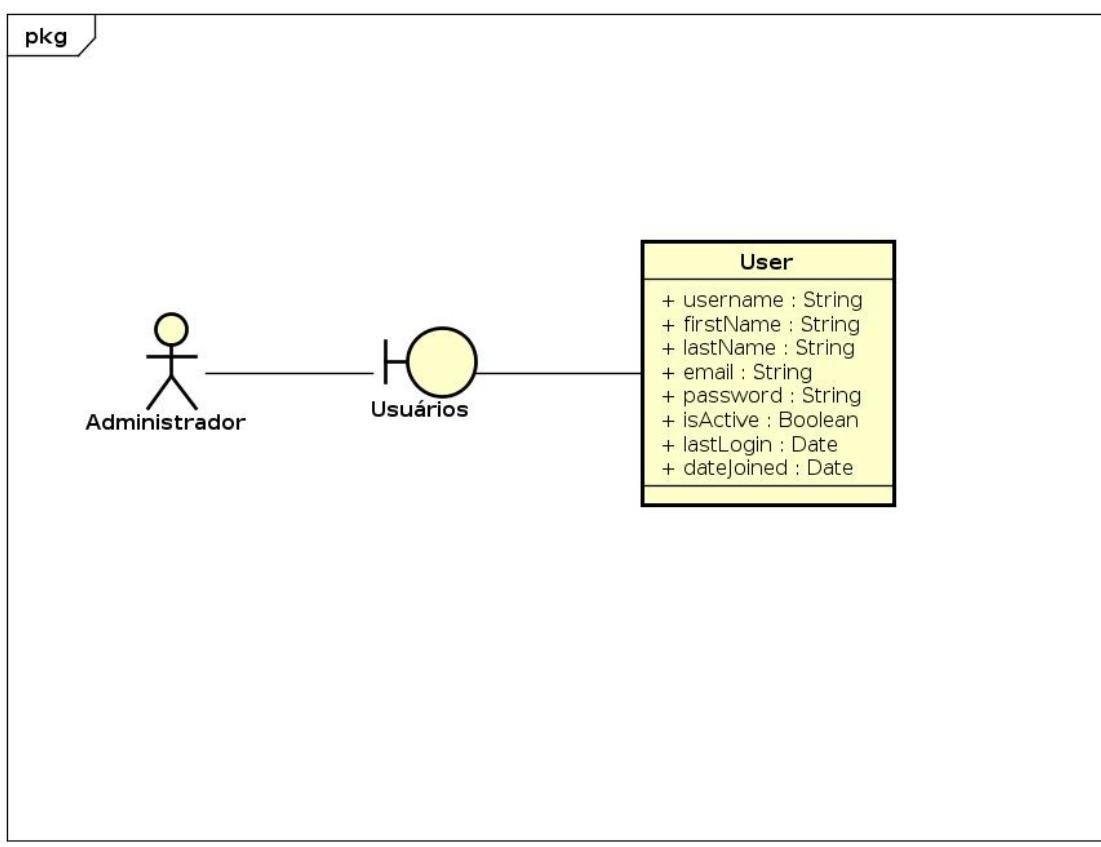


Figura 4.75: Robustez



5 CONSIDERACOES PARCIAIS

REFERÊNCIAS

BENYON, D. **Interação Humano-Computador**. São Paulo - SP: Person, 2011. 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. ISBN 978-1-78398-008-6.

DJANGO. **Django**. [S.l.]: Django Software Foundation, 2015. <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. Caxias do Sul - RS: Soursse, 2010.

GOOGLE. **Conteúdo baseado em plug-in não funciona no Google Chrome**. [S.l.]: Google Support, 2015. <Disponível em: <https://support.google.com/chrome/answer/6213033?hl=pt-BR>>. Acesso em: 2 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.

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**. [S.l.]: Python Software Foundation, 2015. <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.

ROSENBERG, D.; STEPHENS, M.; COLLINS-COPE, M. **Agile development with ICONIX process : people, process, and pragmatism**. New York: Apress, 2005. ISBN 1-59059-464-9.

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