

# RinoCup

## Sistema para gerenciamento de competição de robótica

Autores do trabalho:

Daniel Keim Almeida	202165021AB
Davi Monken Eckhardt	202265019A
Gabriel Cordeiro Tavares	202265163A

Link para Repositório: <https://github.com/DaviEckhardt/Trabalho-oo>

Login e senha de acesso: "Gleiph", Email("gleiph@gmail.com"), senha "Senha123"

# Introdução

O objetivo deste relatório é apresentar uma análise sucinta sobre o sistema desenvolvido para gerenciar uma competição de robótica, a RinoCup. A motivação para tal escolha foram experiências prévias dos integrantes do grupo, que atualmente são membros ativos da Rinobot, equipe de competição de robótica da UFJF.

As principais classes implementadas no código são: Categoria, Usuário, Equipe e Robô, sendo que todos fazem parte do pacote model. Dentre os outros pacotes desenvolvidos, temos controller, onde são administrados os dados das equipes, usuários e impressões na tela, exception, onde é tratada uma possível exception com o e-mail do usuário, repository, onde são manipuladas as listas de equipes, robôs e usuários, utils, facilitador para acesso e tratamento de arquivos e impressões na tela, interfaces, que manipula os id's (altera ou imprime) e recebe as pesquisas, e view, responsável pelas telas de interação com o usuário.

# Pacotes e Classes

Os pacotes implementados foram: **Controller**, **Exception**, **Interfaces**, **Model**, **Repository**, **Utils** e **View**.

**Controller** administra os dados das equipes, usuários e as impressões na tela. A classe `AtualizaDadosBase` dá às janelas as características como abrir, fechar, fechando, com ícones, sem ícones e/ou ativada. Foi necessário importar a biblioteca `java.awt.Window` para fazer uso dessas funções e o evento `java.awt.event.WindowEvent`. As classes implementadas foram utilizadas para gerenciar suas devidas impressões: `AtualizaDadosCadastro` as impressões na tela de cadastro, `AtualizaDadosListagem` as impressões na tela de listagem das informações e `AtualizaDadosCampeonato` as impressões na tela de chaveamento. A classe `LoginControler` controla o acesso ao sistema, testando se usuário e senha informados por aquele que acessa os sistema estão presentes na base de dados, sendo que ocorre a checagem se o email ou nome foi digitado corretamente (testes feitos em `UsuarioRepository` no package `repository` e `email` e `usuario` no package `model`).

**Exception** trata a exception no email em `EmailException`.

**Model** contém as principais classes do programa: `Categoria`, `Email`, `Equipe`, `Peca` (Peça), `Robo` e `Usuario`. Todas elas implementam a interface de `IentidadeRepository` e tem seus devidos getters e setters implementados. A classe `TipoUsuario` é utilizado na classe `Usuario` para definir o nível de acesso ao sistema que ele terá (`Administrador`, `Capitão` ou `Competidor`), e a `ModoTela` define em qual modo a listagem será chamada, em modo de pesquisa ou listagem.

**Utils** é um facilitador para acesso e tratamento de arquivos e impressões na tela, incluindo tamanho das imagens e posição na tela. A classe `Arquivo` realiza escrita de arquivos, `ImageUtils` redimensiona a imagens e `ScreenUtils` centraliza as janelas de acordo com o tamanho do monitor.

**View** é responsável pelas telas de interação com o usuário, utilizando as bibliotecas `awt.BorderLayout`, `awt.event.ActionEvent`, `swing.JButton`, `swing.JFrame` e `swing.JPanel`. A classe `Chaveamento` mostra a classificação das equipes na competição, além de ter um sistema de decisão aleatório dos jogos (caso o administrador não venha a interferir). `CadastroBase` implementa os componentes da janela que será impressa, como o desenho da tela e do rodapé, o botão para salvar e o de fechar. `ListagemBase` determina o tamanho da tela, os campos que poderão ser editados, desenha a divisão de linhas e colunas, o rodapé e os botões, contém funções como `remove`, `escolher`, `carregar`, `editar` e `mostrar` e é utilizada como base para `ListagemEquipe`, `ListagemPeca`, `ListagemRobo` e `ListagemUsuario`. Há polimorfismo onde essas classes modificam as funções de cadastro, edição e acesso à lista de acordo com sua necessidade e permissão (um pré-filtro foi implementado para os casos de remoção). `EquipeCadastro` herda funções de `CadastroBase` e é um polimorfismo de `IPesquisa`. Nessa tela os campos `Nome` e `Cidade` deverão ser preenchidos para cadastrar uma equipe na competição, além de selecionar um capitão para a mesma. As características da equipe podem ser editadas. A classe `JroboField` permite a manipulação das informações dos robôs.

**Repository** é o pacote onde são manipuladas e armazenadas as equipes, robôs, peças e usuários fazendo uso de listas (`import java.util.List`), importando a interface `java.lang.reflect.Type`, a qual permite obter informações sobre os tipos dos objetos e manipulá-los dinamicamente e a classe `com.google.gson.reflect.TypeToken` para conversão de objetos Java em JSON e vice-versa, permitindo representar tipos genéricos em tempo de execução. A classe `Repository` é uma implementação abstrata da interface `IRepository`, como uma lista do tipo `Type` (da interface `Type`). Todas as classes desse package são um polimorfismo, cada uma implementa a classe `Repository.java` com suas propriedades. `EquipeRepository` confere se a equipe está inscrita (se existe na competição), e faz uso da

classe `TypeToken`, `PecaRepository` faz uso da classe `TypeToken` e acessa as peças inscritas pela equipe, `RoboRepository` confere se o robô está inscrito (se existe na competição), e faz uso da classe `TypeToken`, e `UsuarioRepository` permite fazer busca do usuario através de login e senha, confere se o email existe (dentro da lista de login) e confere se o usuário é capitão da equipe.

## Perfis de acesso

Quanto aos perfis de acesso, pretendemos implementar 3: usuário, o qual terá permissão para acessar resultados da competição e quais as equipes que estão participando, capitão, que além de acessar resultados, poderá inscrever seus competidores, robôs, peças e modificar dados de sua própria equipe, e administrador, o qual terá todas as permissões citadas anteriormente, e também poderá alterar resultados de partidas, desclassificar equipes, entre outros.

# Interface

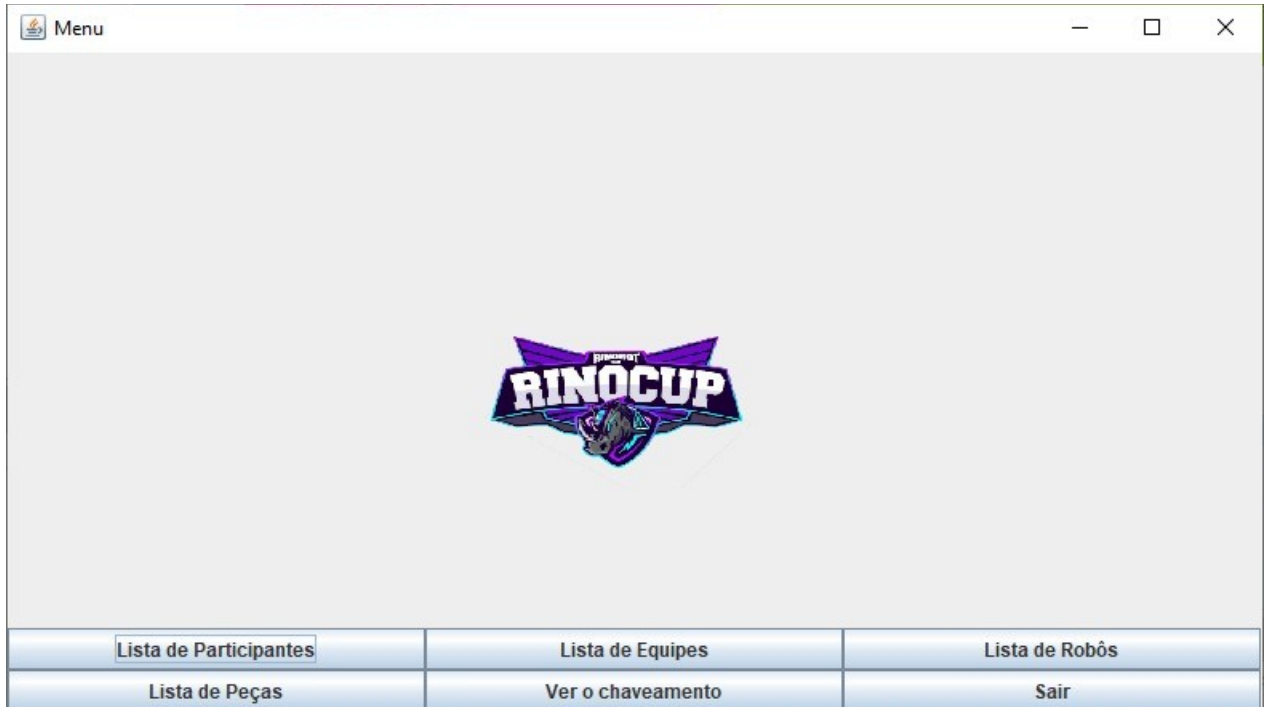


Login

Email ou login Senha

Entrar Cancelar

Login

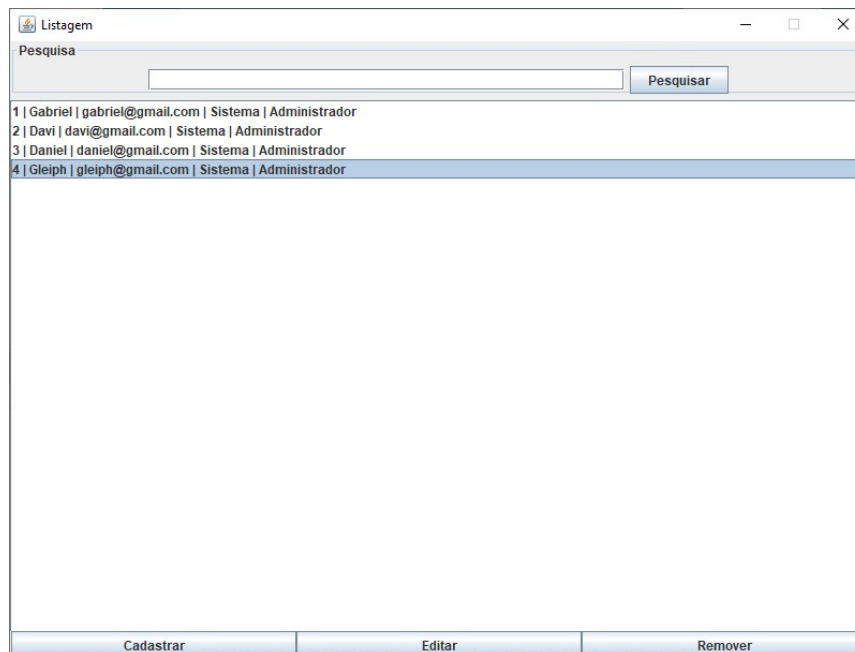


Menu

**RINOCUP**

Lista de Participantes	Lista de Equipes	Lista de Robôs
Lista de Peças	Ver o chaveamento	Sair

Boas vindas



Listagem

Pesquisa

Pesquisar

1   Gabriel   gabriel@gmail.com   Sistema   Administrador
2   Davi   davi@gmail.com   Sistema   Administrador
3   Daniel   daniel@gmail.com   Sistema   Administrador
4   Gleiph   gleiph@gmail.com   Sistema   Administrador

Cadastrar Editar Remover

Lista usuários

Cadastro...

Nome:

Email:

Senha:

Categoria:

Sistema

Escolher Equipe

Salvar

## Cadastro Usuário

Listagem

Pesquisa

Pesquisar

Cadastro de Equipe

Nome:

Cidade:

Selecionar Capitão

Salvar

Cadastrar

Editar

Remover

## Cadastro Equipe



Listagem

Pesquisa

Pesquisar

Cadastro...

Nome:

Categoria:

VSSS

Escolher Equipe

Salvar

Cadastrar

Editar

Remover

## Cadastro Membro

Listagem

Pesquisa

Pesquisar

Cadastro...

Nome:

Valor:

0

Quantidade:

0

Escolher Robô

Salvar

Cadastrar

Editar

Remover

## Cadastro Peças

**Painel de Categorias**

Selecione uma categoria:

VSSS
Mini Sumo
Sumo Lego
SPL
Seguidor de Linha
Perseguidor de Linha
Combate

Painel de Categorias

**Simulação de campeonato**

Diagrama de chaveamento (torneio) mostrando os participantes (W.O.) e as etapas de confronto:

```
graph LR; W1[W.O.] --- W2[W.O.]; W3[W.O.] --- W4[W.O.]; W5[W.O.] --- W6[W.O.]; W7[W.O.] --- W8[W.O.]; W9[W.O.] --- W10[W.O.]; W11[W.O.] --- W12[W.O.]; W13[W.O.] --- W14[W.O.]; W15[W.O.] --- W16[W.O.]; W17[W.O.] --- W18[W.O.]; W19[W.O.] --- W20[W.O.]; W21[W.O.] --- W22[W.O.]; W23[W.O.] --- W24[W.O.]; W25[W.O.] --- W26[W.O.]; W27[W.O.] --- W28[W.O.]; W29[W.O.] --- W30[W.O.]; W31[W.O.] --- W32[W.O.]; W33[W.O.] --- W34[W.O.]; W35[W.O.] --- W36[W.O.]; W37[W.O.] --- W38[W.O.]; W39[W.O.] --- W40[W.O.]; W41[W.O.] --- W42[W.O.]; W43[W.O.] --- W44[W.O.]; W45[W.O.] --- W46[W.O.]; W47[W.O.] --- W48[W.O.]; W49[W.O.] --- W50[W.O.]; W51[W.O.] --- W52[W.O.]; W53[W.O.] --- W54[W.O.]; W55[W.O.] --- W56[W.O.]; W57[W.O.] --- W58[W.O.]; W59[W.O.] --- W60[W.O.]; W61[W.O.] --- W62[W.O.]; W63[W.O.] --- W64[W.O.]; W65[W.O.] --- W66[W.O.]; W67[W.O.] --- W68[W.O.]; W69[W.O.] --- W70[W.O.]; W71[W.O.] --- W72[W.O.]; W73[W.O.] --- W74[W.O.]; W75[W.O.] --- W76[W.O.]; W77[W.O.] --- W78[W.O.]; W79[W.O.] --- W80[W.O.]; W81[W.O.] --- W82[W.O.]; W83[W.O.] --- W84[W.O.]; W85[W.O.] --- W86[W.O.]; W87[W.O.] --- W88[W.O.]; W89[W.O.] --- W90[W.O.]; W91[W.O.] --- W92[W.O.]; W93[W.O.] --- W94[W.O.]; W95[W.O.] --- W96[W.O.]; W97[W.O.] --- W98[W.O.]; W99[W.O.] --- W100[W.O.];
```

Chaveamento

# Instrução para compilar e executar o projeto

Primeiramente é necessário executar o comando “mvn clean install” ou “mvn install”, o qual gerará a pasta target, a qual possui os arquivos e pastas que são resultados do processo de compilação e empacotamento. Dentre eles estará o arquivo “RinoCup-1.0-SNAPSHOT-jar.with.dependencies.jar”, o qual será utilizado para executar o projeto. Para executar o projeto, basta abrir a pasta que está com o arquivo pom.xml pelo terminal e digitar o comando “java -jar target/RinoCup-1.0-SNAPSHOT-jar.with.dependencies.jar”.