



**UNINABUCO**

ANÁLISE E DESENVOLVIMENTO DE SOFTWARE

TÓPICOS INTEGRADORES II

# Documentação de Projeto de Software

## Versão 3.0

Autor(a): Diego da Silva Oliveira

GitHub:

[https://github.com/diego5310/cadastro\\_lives.git](https://github.com/diego5310/cadastro_lives.git)

Data: 25/05/2020

2020

## Controle de Versão do Documento

Versão	Descrição
3.0	Foram criadas as telas e as classes da camada View, juntamente com o Diagrama de Classes do projeto Lives.

# Sumário

1. Introdução ao Documento .....	4
1.1. Área de negócio do Sistema .....	4
1.2. Principais funcionalidades .....	4
1.3. Método de trabalho.....	4
2. Modelo de Dados .....	4
2.1. Modelo de Visão .....	5
2.2. Modelo Conceitual .....	6
2.3. Modelo Lógico.....	7
2.4. Dicionário de Dados.....	8
2.5. Modelo Físico .....	9
3. Análise e Design .....	14
3.1. Diagrama de Classes .....	14
4. Arquitetura do Software.....	15
4.1. Padrão de projeto.....	15
4.2. Protótipo.....	15

# 1. Introdução ao Documento

## **1.1. Área de negócio do Sistema**

Sistema de cadastro para LiveStreams, podendo atuar como Streamer ou Espectador-Seguidor dependendo da escolha durante o cadastro

## **1.2. Principais funcionalidades**

Cadastro de Streamer ou Espectador-Seguidor, Função de excluir cadastro de um seguidor para a entidade Bot, e métodos de abrir e fechar uma live, e alterar seu título para a entidade Streamer.

## **1.3. Método de trabalho**

Serão utilizadas as seguintes ferramentas:

Banco de dados MySQL

SGBD: MariaDB

Linguagem: Java

IDE: Eclipse/Netbeans

## **2.0. Modelo de Dados**

## 2.1. Modelo de Visão:

Este banco de dados se trata de um sistema para cadastro de Lives, onde terá Streamer, Live, Categoria, Bot e Espectador, com sua especificação de Seguidor (follower) onde seguidor vai herdar de Espectador.

Live pode ter 1 ou vários Bot's e Bot's podem moderar 1 ou várias lives (N para N)

**Streamer** : Realizará as transmissões (lives) e também responsável pela configuração da mesma. Terá: id\_streamer, nome, usuario, senha, dt\_cadastro e inlive

**Bot** : Terá a função de excluir um cadastro de seguidor, caso seja necessário. (para casos de qualquer tipo de preconceito por exemplo) Terá: id\_bot (PK) e nome

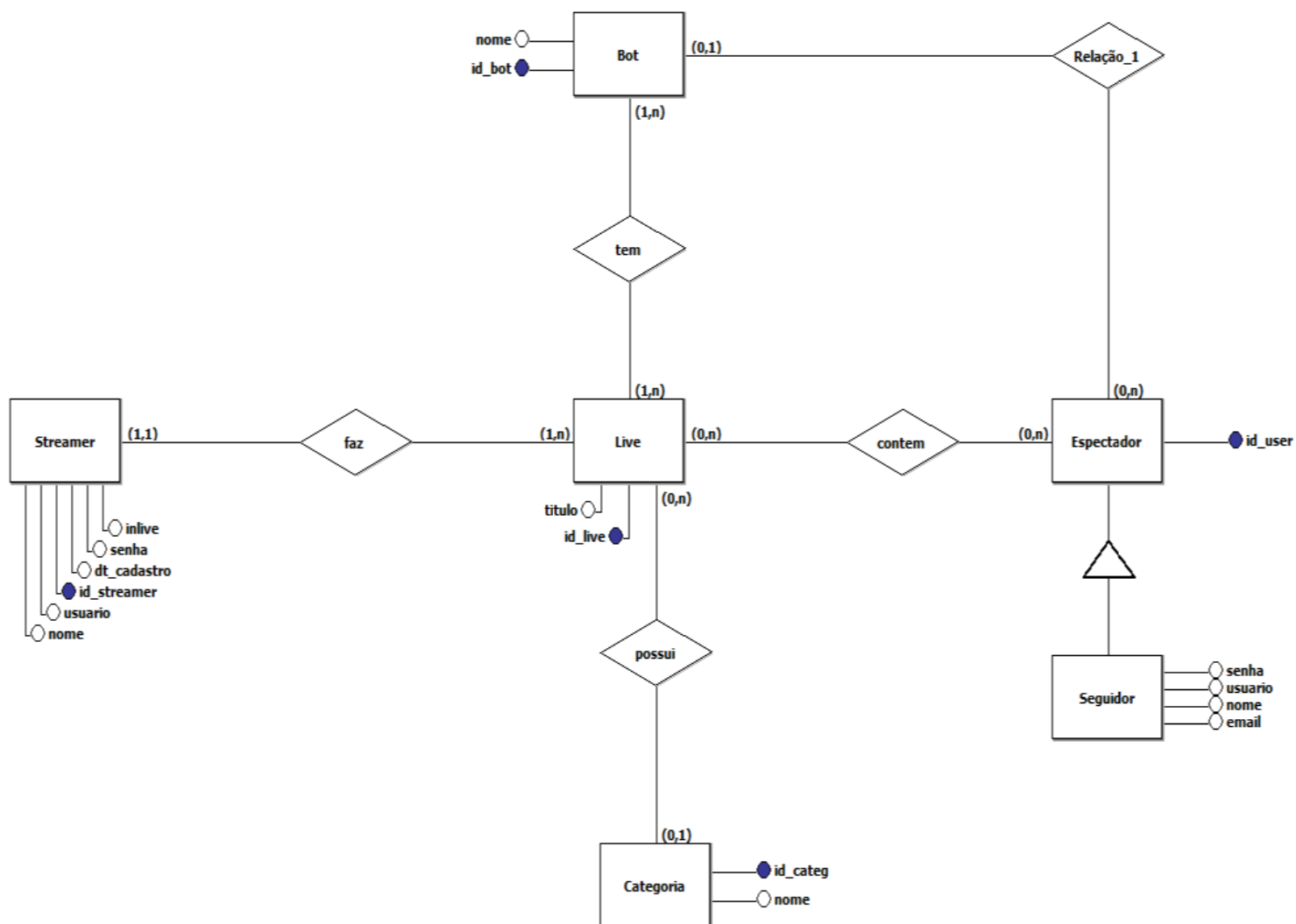
**Espectador** : Poderá assistir as lives e decidir se continuará apenas como espectador ou se tornará seguidor da live, realizando o cadastro. Terá: id\_user(PK)

**Seguidor** : Herdará de Espectador, adicionando os campos : email,nome, usuario, senha e dt\_cadastro.

**Live**: A própria transmissão em si, com sua devida categoria servindo como "produto" . Terá: id\_live(PK), id\_streamer (FK), título e id\_categ

**Categoria**: Seria o nome do jogo que estará sendo transmitido. Terá: id\_categ e nome

## 2.2. Modelo Conceitual



## 2.3. Modelo Lógico

**Streamer** (id\_streamer, nome, dt\_cadastro, usuario, senha e inlive)

id\_streamer - Primary key

**Categoria** (id\_categ, nome)

id\_categ – Primary Key

**Live** (id\_live, id\_streamer, id\_categ, titulo)

id\_live - Primary key

id\_streamer referencia Streamer

id\_categ referencia Categoria

**Bot** (id\_bot, nome)

id\_bot - Primary key

**Espectador\_Seguidor** (id\_user, nome, dt\_cadastro, email, usuario, senha, id\_bot)

id\_user - Primary key

id\_bot referencia bot

**Live\_bot** (id\_bot, id\_live)

id\_bot referencia Bot

id\_live referencia Live

id\_live, id\_bot - Primary Key

**Live\_Espectador\_Seguidor** (id\_user, id\_live)

id\_user referencia espectador\_seguidor

id\_live referencia live

id\_user, id\_live - Primary key

## 2.4. Dicionário de Dados

Tabela	Descrição		
Streamer	Realizará as transmissões (Lives)		
Campo	Tipo	Descrição	Observações e Regras
nome	varchar(100)	Nome completo do Streamer	Not Null
id_streamer	int (auto incremento)	id associado ao Streamer	Not Null / PK
email	varchar(50)	E-mail do Streamer	Not Null / Unique
usuario	varchar(20)	Usuario (Login) do Streamer	Not Null / Unique
senha	varchar(20)	Senha utilizada pelo Streamer	Not Null
dt_cadastro	date	Data de cadastro como Streamer	Not Null
inlive	boolean	Determinará se o Streamer está no modo Streaming ou não	Not Null
Tabela	Descrição		
Live	Seria a própria transmissão em si, servindo como "produto"		
Campo	Tipo	Descrição	Observações e Regras
id_live	int (auto incremento)	id associado a transmissão (Live)	Not Null / PK
id_categ	int (auto incremento)	id associado a Categoria	Not Null (FK) (Categoria)
titulo	varchar(50)	titulo da transmissão, escolhido pelo Streamer associado a Live	Not Null
id_streamer	int	id associado ao Streamer	Not Null / FK (Streamer)
Tabela	Descrição		
Bot	Responsável por "moderar" as lives		
Campo	Tipo	Descrição	Observações e Regras
nome	varchar(20)	Nome do Bot	Not Null
id_bot	int (auto incremento)	id associado ao Bot	Not Null / PK
		id associado ao espectador_seguidor	Not Null
Tabela	Descrição		
espectador_seguidor	Assistirá as transmissões		
Campo	Tipo	Descrição	Observações e Regras
nome	varchar(100)	Nome do espectador_seguidor	
id_user	int (auto incremento)	id associado ao espectador_seguidor	Not Null / PK
email	varchar(50)	E-mail atrelado a espectador_seguidor	Unique
usuario	varchar(20)	usuario utilizado pelo espectador_seguidor	Unique
senha	varchar(20)	senha associada a espectador_seguidor	
dt_cadastro	Date	Data de cadastro	
id_bot	int	id referente a tabela Bot	(FK)
Tabela	Descrição		
live_bot	Liga a tabela Bot com Live		
Campo	Tipo	Descrição	Observações e Regras
id_live	int	id associado a tabela Live	Not Null / FK (Live)
id_bot	int	id associado a tabela Bot	Not Null / FK (Bot)
id_live, id_bot	int (auto incremento)	id's associados a tabela live_bot	Not Null / PK (composta)
Tabela	Descrição		
live_espectador_seguidor	garantir que possa haver espectadores diferentes assistindo a mesma live		
Campo	Tipo	Descrição	Observações e Regras
id_user	int	id associado a tabela espectador_seguidor	FK (espectador_seguidor)
id_live	int	id associado a tabela live	FK (live)
id_user, id_live	int	id's associados a live_espectador_seguidor	PK (Composta)
Tabela	Descrição		
Categoria	categoria referente a live		
Campo	Tipo	Descrição	Observações e Regras
id_categ	int (auto incremento)	id associado a Categoria	PK
nome	varchar(50)	nome associado a uma categoria	Not Null



## **2.5. Modelo Físico**

### **2.5.1.**

```
DROP DATABASE IF EXISTS cadastro_lives;
```

```
CREATE DATABASE cadastro_lives;
```

```
use cadastro_lives;
```

```
CREATE TABLE streamer (
```

```
id_streamer int not null primary key auto_increment,
```

```
nome varchar(100) not null,
```

```
usuario varchar(20) not null,
```

```
senha varchar(20) not null,
```

```
dt_cadastro date not null,
```

```
inlive boolean not null
```

```
);
```

```
CREATE TABLE categoria (
```

```
id_categ int not null primary key auto_increment,
```

```
nome varchar(50) not null
```

```
);
```

```
CREATE TABLE live (
```

```
id_live int not null primary key auto_increment,
```

```
id_categ int not null,
```

```
titulo varchar(50) not null,
```

```
id_streamer int not null,
```

```
constraint fk_id_streamer foreign key (id_streamer) references  
streamer(id_streamer),
```

```
constraint fk_id_categ foreign key (id_categ) references categoria(id_categ)
```

```
);
```

```
CREATE TABLE bot (
```

```
id_bot int not null primary key auto_increment,
```

```
nome varchar(20) not null  
);
```

```
CREATE TABLE espectador_seguidor (  
id_user int not null primary key auto_increment,  
email varchar(50),  
usuario varchar(20),  
senha varchar(20),  
nome varchar(100),  
dt_cadastro date,  
id_bot int,  
constraint fk_id_bot foreign key (id_bot) references bot(id_bot)  
);
```

```
CREATE TABLE live_bot (  
id_live int not null,  
id_bot int not null,  
constraint fk_id_live_ foreign key (id_live) references live(id_live),  
constraint fk_id_bot_ foreign key (id_bot) references bot(id_bot),  
constraint pk_live_bot primary key (id_live,id_bot)  
);
```

```
CREATE TABLE live_espectador_seguidor (  
id_user int,  
id_live int,  
constraint fk_id_user foreign key (id_user) references  
espectador_seguidor(id_user),  
constraint fk_id_live foreign key (id_live) references live(id_live),  
constraint pk_live_espectador_seguidor primary key (id_user,id_live)  
);
```

### 2.5.2. Comandos INSERT

/\* Adicionando valores na tabela Streamer \*/

```
INSERT INTO streamer  
(id_streamer,nome,usuario,senha,dt_cadastro,inlive) VALUES (1, 'Evandro  
Mota', 'evandro369', 'Evandro159', '2019-05-01', TRUE);
```

```
INSERT INTO streamer  
(id_streamer,nome,usuario,senha,dt_cadastro,inlive) VALUES (2, 'Enos  
Victor', 'enosvt1', 'Enos5500', '2019-06-02', TRUE);
```

```
INSERT INTO streamer  
(id_streamer,nome,usuario,senha,dt_cadastro,inlive) VALUES (3, 'Maria  
Eduarda', 'dudinha00', 'duda2405', '2020-08-23', TRUE);
```

```
INSERT INTO streamer  
(id_streamer,nome,usuario,senha,dt_cadastro,inlive) VALUES (4, 'Gabriel  
Souza', 'gabriel007', '159753', '2020-09-20', FALSE);
```

```
INSERT INTO streamer  
(id_streamer,nome,usuario,senha,dt_cadastro,inlive) VALUES (5, 'Daniel  
Silva', 'danisil', 'dandan123', '2020-12-28', FALSE);
```

/\* Adicionando valores na tabela Categoria \*/

```
INSERT INTO categoria (id_categ,nome) VALUES (1,'League of Legends');
```

```
INSERT INTO categoria (id_categ,nome) VALUES (2,'Apex Legends');
```

```
INSERT INTO categoria (id_categ,nome) VALUES (3,'CS:GO');
```

```
INSERT INTO categoria (id_categ,nome) VALUES (4,'ARK');
```

```
INSERT INTO categoria (id_categ,nome) VALUES (5,'COD');
```

/\* Adicionando valores na tabela Live \*/

```
INSERT INTO live (titulo,id_streamer,id_categ) VALUES ('live do Evandro',  
1,1);
```

```
INSERT INTO live (titulo,id_streamer,id_categ) VALUES ('live do Victor',  
2,2);
```

```
INSERT INTO live (titulo,id_streamer,id_categ) VALUES ('live da Duda',  
3,3);
```

```
INSERT INTO live (titulo,id_streamer,id_categ) VALUES ('live do Gabriel',  
4,4);
```

```
INSERT INTO live (titulo,id_streamer,id_categ) VALUES ('live do Daniel',  
5,5);
```

```
/* Adicionando valores na tabela Bot */
```

```
INSERT INTO bot (nome) VALUES ('Groovy');
```

```
INSERT INTO bot (nome) VALUES ('Rythm');
```

```
INSERT INTO bot (nome) VALUES ('Element');
```

```
INSERT INTO bot (nome) VALUES ('Holics');
```

```
INSERT INTO bot (nome) VALUES ('Groovy');
```

```
/* Adicionando valores na tabela Espectador_Seguidor */
```

```
INSERT INTO espectador_seguidor  
(email,usuario,senha,nome,dt_cadastro,id_bot) VALUES  
('pauloborges@gmail.com', 'paulobg0', 'paulo#13', 'Paulo Borges', '2019-  
09-13', 3);
```

```
INSERT INTO espectador_seguidor (email,usuario,senha,nome,dt_cadastro)  
VALUES ('rodrigojose@gmail.com', 'jrodrigo1', 'jrod63', 'José Rodrigo',  
'2018-05-12');
```

```
INSERT INTO espectador_seguidor (email,usuario,senha,nome,dt_cadastro)  
VALUES ('carlosalmeida@gmail.com', 'carlosalm36', '1596320', 'Carlos  
Almeida', '2018-07-018');
```

```
INSERT INTO espectador_seguidor (email) VALUES (null);
```

```
INSERT INTO espectador_seguidor (email) VALUES (null);
```

```
/* Adicionando valores na tabela Live_Bot */
```

```
INSERT INTO live_bot (id_live,id_bot) VALUES (1, 1);
```

```
INSERT INTO live_bot (id_live,id_bot) VALUES (2, 1);
```

```
INSERT INTO live_bot (id_live,id_bot) VALUES (3, 3);
```

```
INSERT INTO live_bot (id_live,id_bot) VALUES (4, 4);
```

```
INSERT INTO live_bot (id_live,id_bot) VALUES (5, 5);
```

```
/* Adicionando valores na tabela live_espectador_seguidor */
```

```
INSERT INTO live_espectador_seguidor (id_user,id_live) VALUES (1, 1);
```

```
INSERT INTO live_espectador_seguidor (id_user,id_live) VALUES (2, 2);
```

```
INSERT INTO live_espectador_seguidor (id_user,id_live) VALUES (3, 3);
```

```
INSERT INTO live_espectador_seguidor (id_user,id_live) VALUES (4, 4);
```

```
INSERT INTO live_espectador_seguidor (id_user,id_live) VALUES (5, 5);
```

### **2.5.3. Os Relatórios e os comandos Selects**

#### **Relatórios:**

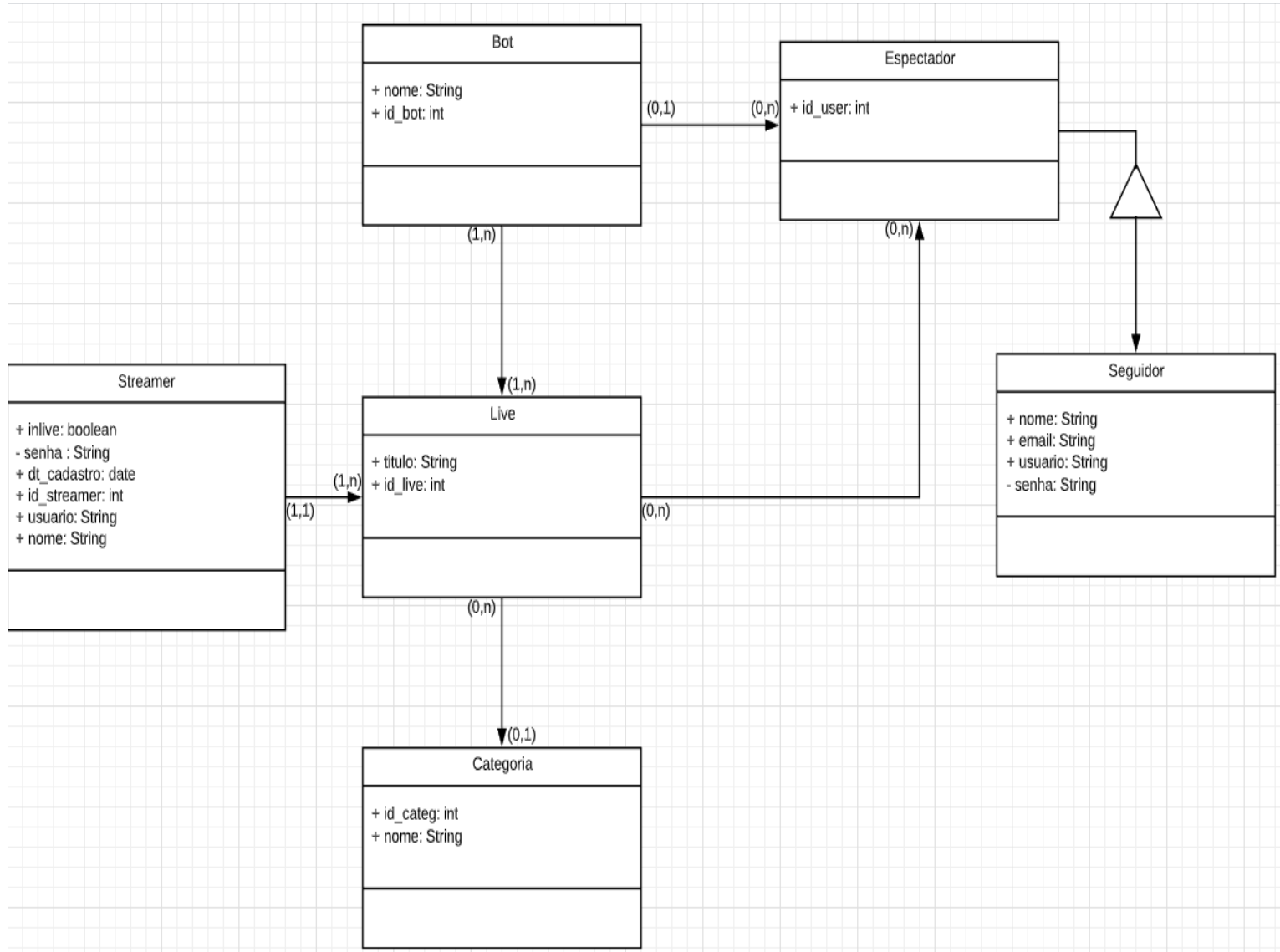
- 1 - Listar a quantidade de lives por Bot
- 2 - listar os nomes e emails dos seguidores da live de código 3.
- 3 - Listar a quantidade de cadastro de streamers, por mês e no ano de 2019.
- 4 - Listar os bots cadastrados
- 5 - listar as lives, ordenando pela categoria.

#### **Selects:**

- 1) Select count(liv.id\_live), bot.nome as nome\_bot from live liv, live\_bot lbo, bot where liv.id\_live = lbo.id\_live and lbo.id\_bot = bot.id\_bot group by bot.nome;
- 2) Select ese.nome, ese.email, liv.id\_live from live liv, espectador\_seguidor ese, live\_espectador\_seguidor les where liv.id\_live = 3 and liv.id\_live = les.id\_live and les.id\_user = ese.id\_user;
- 3) Select id\_streamer, dt\_cadastro from streamer where YEAR(streamer.dt\_cadastro) = 2019 group by MONTH(streamer.dt\_cadastro);
- 4) Select id\_bot, nome FROM bot;
- 5) select live.id\_live, live.titulo, categoria.nome from live, categoria where live.id\_categ = categoria.id\_categ order by categoria.nome;

## 3.0. Análise e Design

### 3.1. Diagrama de Classes



## 4.0. Arquitetura do Software

### 4.1. Padrão de projeto

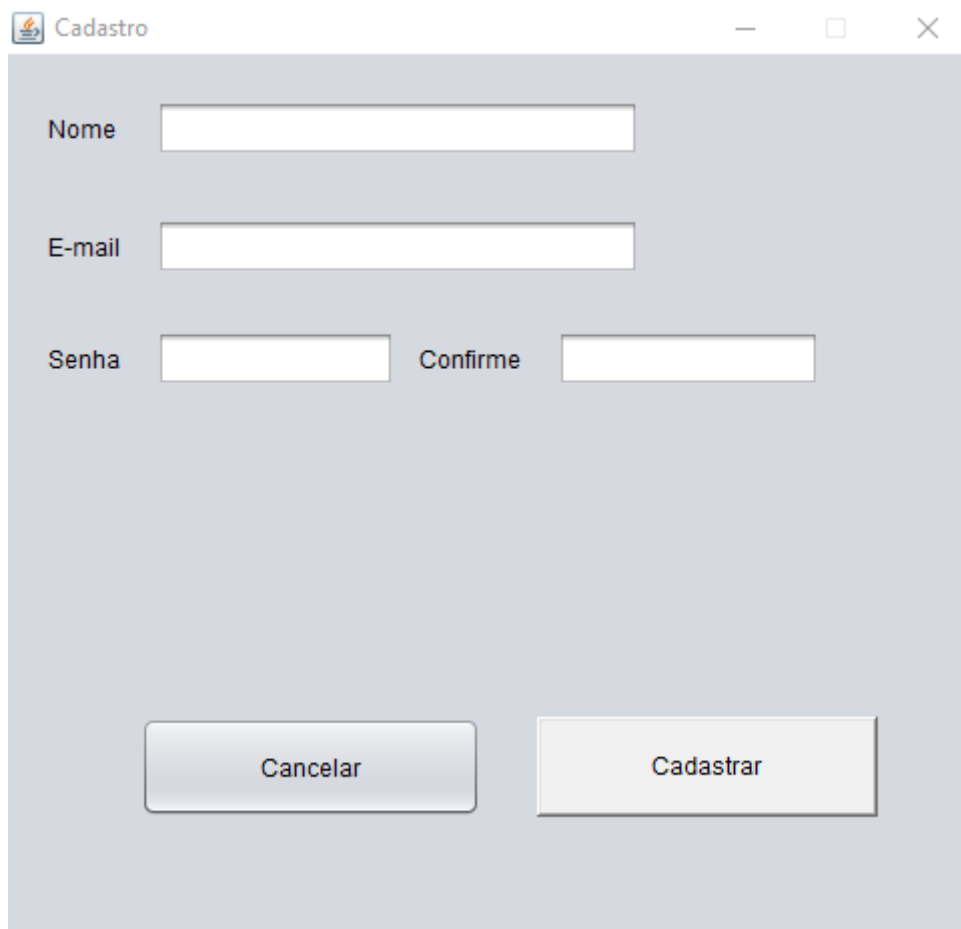
Deverá ser apresentada uma breve descrição sobre o padrão MVC e descrever as classes que serão implementadas em cada uma das camadas.

### 4.2. Protótipo

#### 4.2.1. Telas e Classes da camada View, Requisitos Funcionais e os Códigos das Classes



Na **Tela Inicial**, temos 5 botões, cada um com suas respectivas funções, onde o próprio nome do botão já indica sua funcionalidade, ao clicar em um dos botões, o usuário será direcionado a respectiva tela responsável pela função que os botões indicam.

A screenshot of a software window titled "Cadastro". The window has a light gray background and a standard title bar with a minimize button, a maximize button (disabled), and a close button. Inside the window, there are three input fields: "Nome" (Name), "E-mail", and "Senha" (Password). The "Senha" field is split into two parts: "Senha" and "Confirme" (Confirm), each with its own input field. At the bottom of the window, there are two buttons: "Cancelar" (Cancel) and "Cadastrar" (Register).

Cadastro


Nome

E-mail

Senha  Confirme

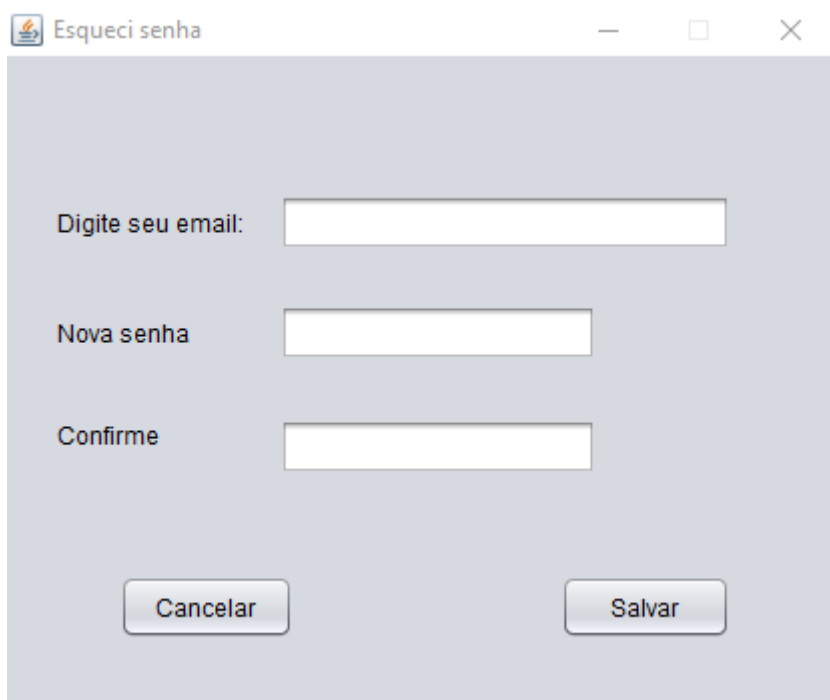
A **Tela de Cadastro** será usada para todos os tipos de cadastro, como Streamer ou Seguidor, preenchendo os campos de textos e os de senha e clicando em 'Cadastrar' para finalizar e concluir o cadastro ou em 'cancelar' para não realizar nenhum registro.





The image shows a software window titled "Login". At the top center is a black television icon with a green screen and two antennae. Below the icon, the word "Login" is displayed above a white text input field. Further down, the word "Senha" is displayed above another white text input field. At the bottom left, there is a button labeled "Esqueci Senha". At the bottom right, there is a button labeled "Entrar". The window has a standard title bar with a minimize button, a maximize button, and a close button.

A **Tela de Login**, será mostrada caso o usuário escolha a opção de 'acessar' e tenha realizado cadastro, seja como Streamer ou Espectador para, e nesta tela também há uma funcionalidade para os usuários cadastrados que esqueceram suas senhas, fazendo com que ao clicar, sejam redirecionados para outra tela (TelaEsqueci) onde poderá estar criando uma nova senha.



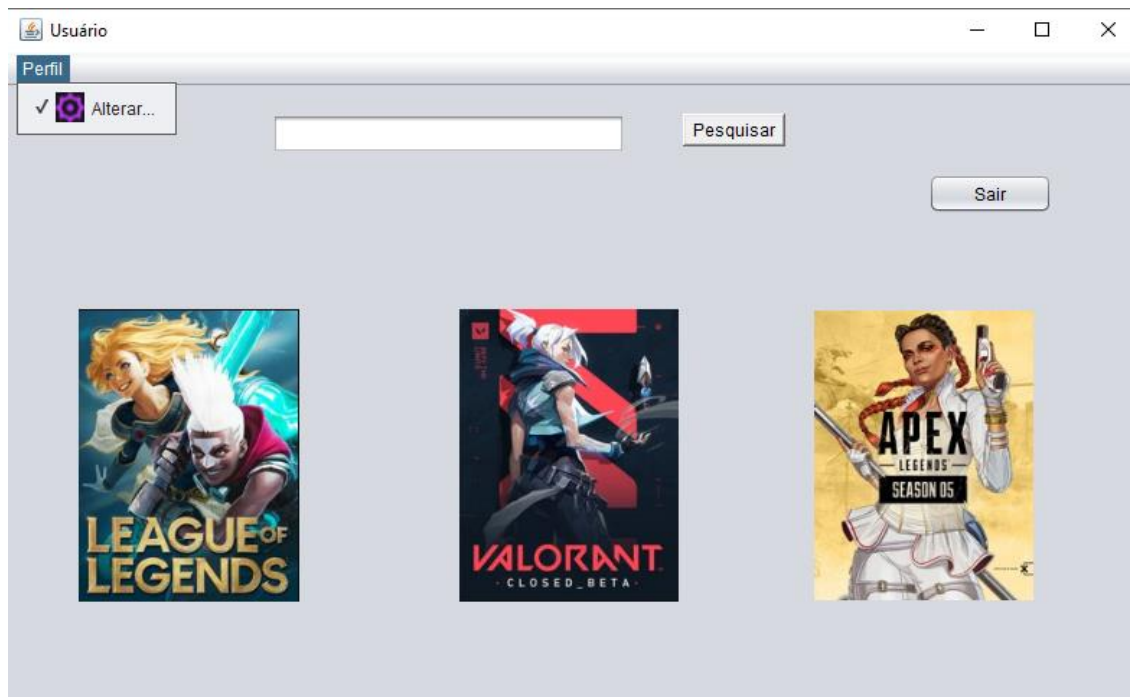
Esqueci senha

Digite seu email:

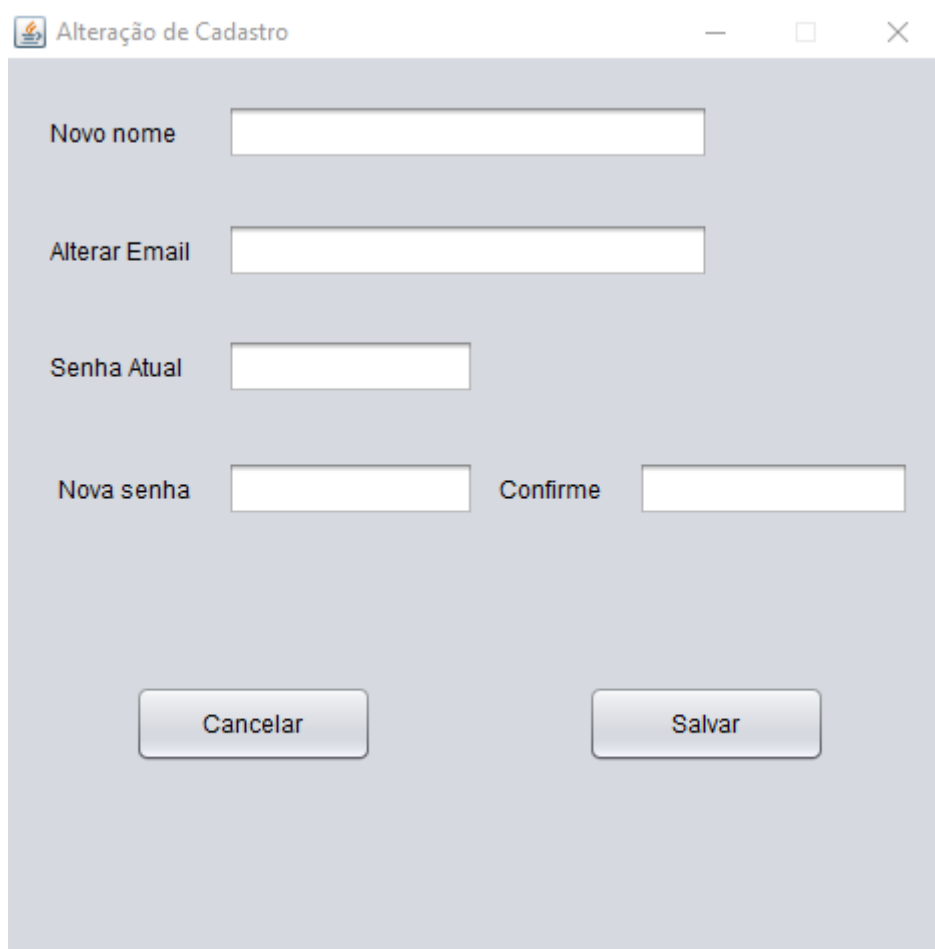
Nova senha

Confirme

A **Tela Esqueci** é exibida após um usuário cadastrado clicar em 'esqueci senha', funcionalidade da **Tela Login**, para que o usuário possa criar sua nova senha, preenchendo os demais campos (Texto e Senha) e clicando em 'Salvar' para que a operação seja realizada, ou em 'Cancelar' para abortar a alteração, caso ele lembre da senha durante a etapa.



A **Tela de Seguidor** será exibida após o usuário que anteriormente, realizou um cadastro como seguidor, fizer o Login, tendo um botão 'Sair' para fechar a tela, um menu 'Perfil' com um item de menu chamado 'Alterar...' que ao ser clicado, irá direcionar o usuário a **Tela de Alteração de Cadastro**, onde poderá alterar seu nome, email e senha, preenchendo os campos solicitados e salvando as alterações, clicando no botão 'Salvar'.



Alteração de Cadastro

Novo nome

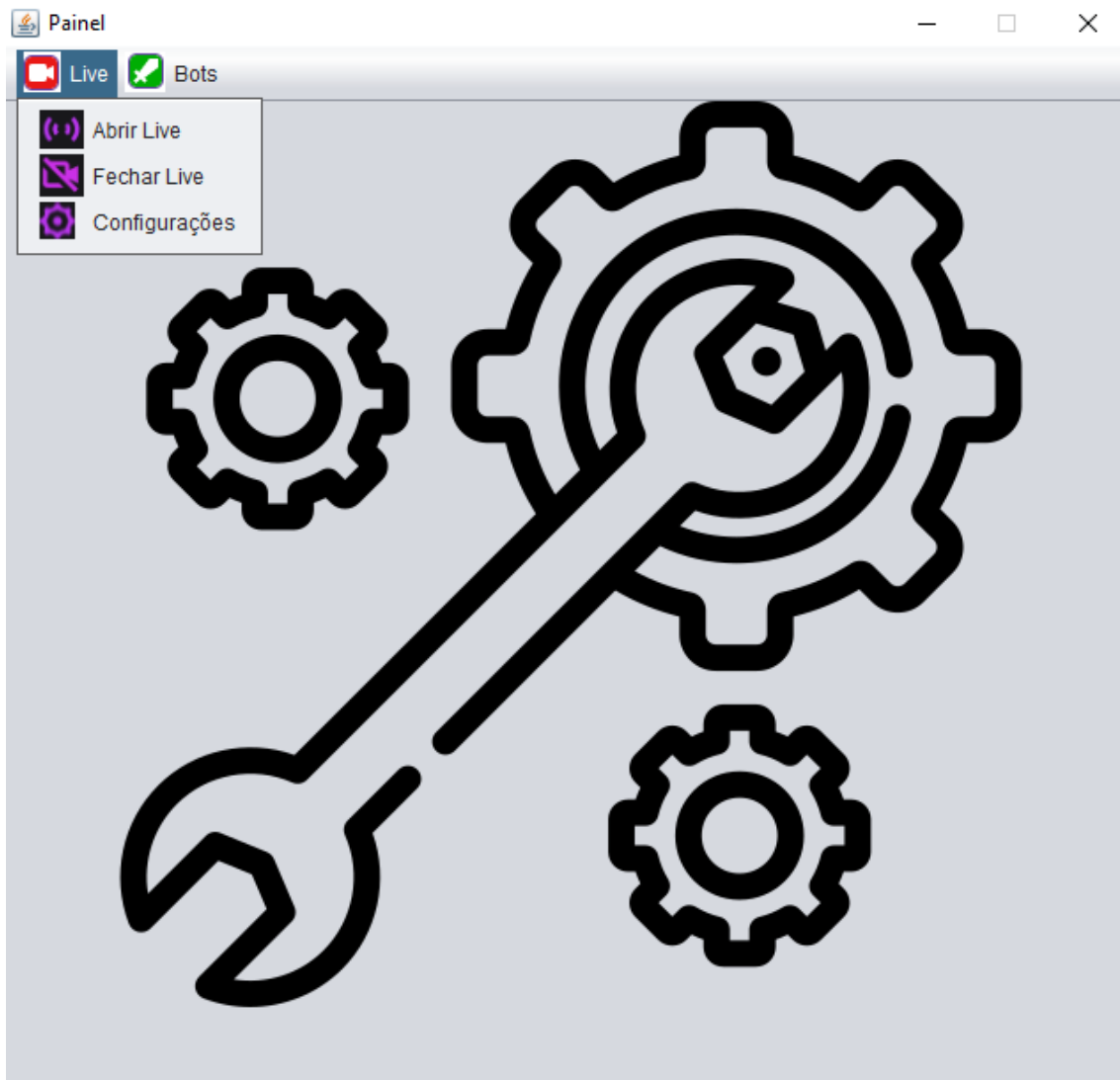
Alterar Email

Senha Atual

Nova senha  Confirme

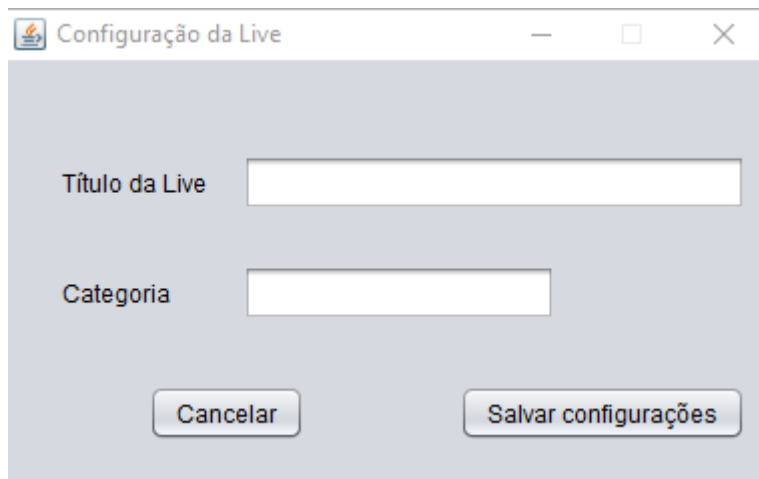
Cancelar Salvar

A **Tela de Alteração de Cadastro**, será exibida após o usuário Seguidor clicar em 'Alterar', funcionalidade esta que se encontra na **Tela de Seguidor**, onde ao preencher os campos de texto e senha e clicando no botão de 'Salvar' irá realizar as alterações nos registros de seu cadastro, ou clicando no botão 'Cancelar' para desistir de efetuar as alterações.



A **Tela Streamer** será exibida após o usuário cadastrado como Streamer realizar o Login, tendo itens de Menu como 'Abrir Live' que ao ser clicado irá exibir a mensagem "Transmissão Iniciada" e 'Fechar Live' trará a mensagem "Transmissão Encerrada", já o item Configurações irá abrir a **Tela de Configuração da Live**, para que o Streamer preencha os campos de Título e Categoria referentes a Live, tendo como opção um botão de 'Salvar Configurações' ou 'Cancelar' para abortar operação.

Nesta mesma **Tela de Streamer** em 'Bots' haverá um item de menu 'Configurar' que irá trazer a **Tela de Bots**.



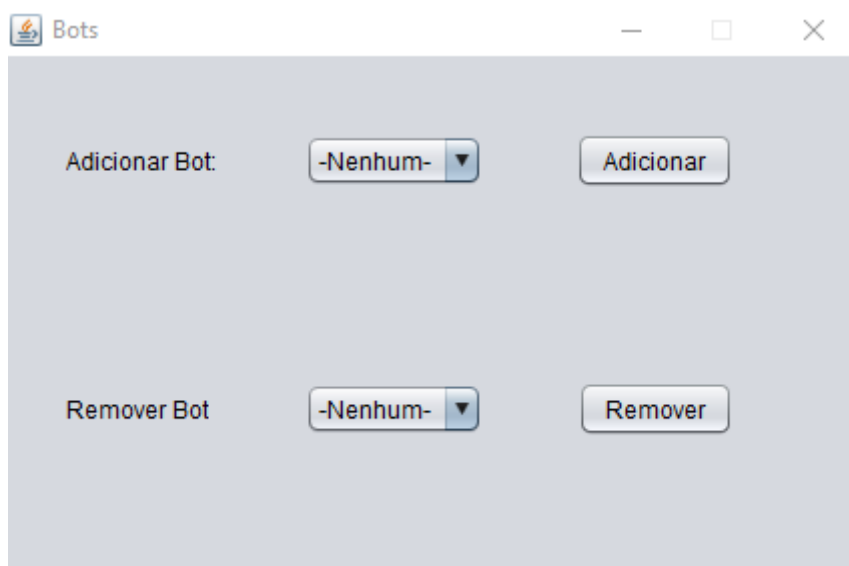
Configuração da Live

Título da Live

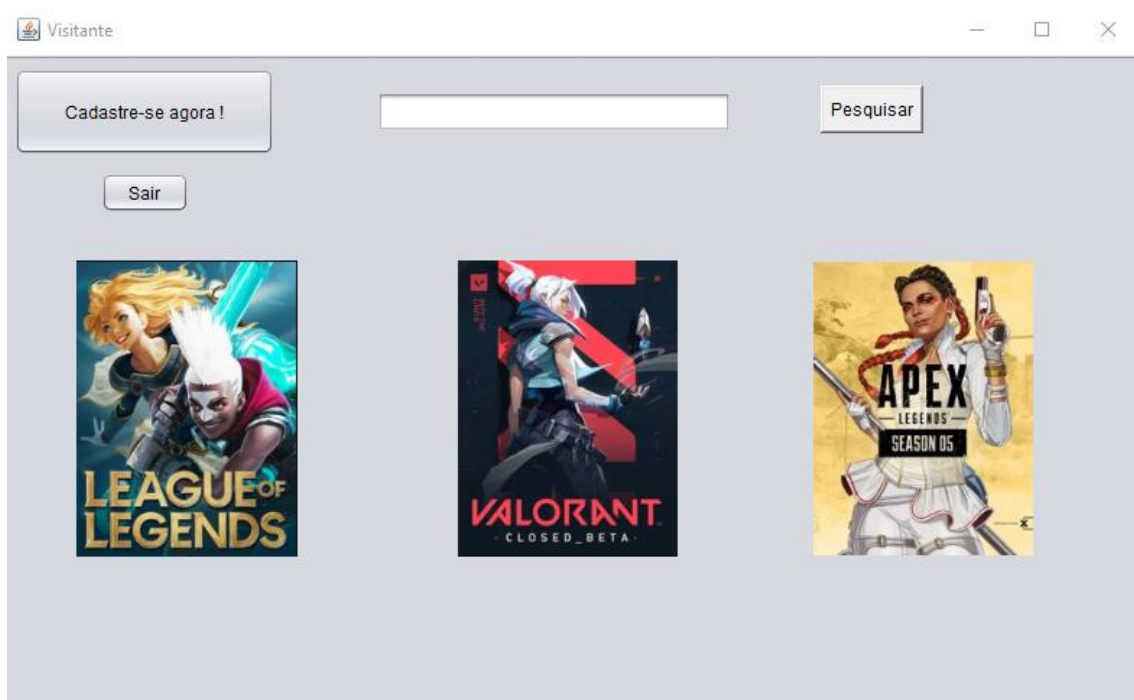
Categoria

Cancelar Salvar configurações

A **Tela de Configuração da Live** será exibida caso o Streamer clique em 'Configurações', ao preencher os campos, clicando em 'Salvar configurações' irá finalizar e concluir a operação, e 'Cancelar' irá fechar a tela sem realizar alguma alteração.



A **Tela Bots**, exibida após o Streamer clicar em 'Bots' e em 'Configurar' funcionalidades que pertencem a **Tela Streamer**, para que o Streamer decida qual Bot ele quer adicionar ou remover, selecionando por nome nas caixas de seleção e finalizando a operação ao clicar nos botões 'Adicionar' ou 'Remover'.



A **Tela Visitante** será exibida para o usuário que ainda não é cadastrado em nenhuma das opções, ou que simplesmente decidam acessar como visitante, funcionalidade disponível na **Tela de Login**, ao clicar no botão 'Acesar', terá botão de 'Sair' para fechar a tela e um botão 'Cadastre-se agora!' caso o visitante tenha interesse em se cadastrar, ao clicar no botão de 'Cadastre-se agora' o visitante será direcionado para a **Tela de Cadastro**.



# Códigos das Classes View

## Tela Inicial:

```
private void CadEspecActionPerformed(java.awt.event.ActionEvent evt) {  
    TelaCadastro telCad = new TelaCadastro();  
    telCad.setVisible(true);  
}
```

```
private void CadStrActionPerformed(java.awt.event.ActionEvent evt) {  
    TelaCadastro telCad = new TelaCadastro();  
    telCad.setVisible(true);  
}
```

```
private void AcVisitActionPerformed(java.awt.event.ActionEvent evt) {  
    TelaVisitante telVisit = new TelaVisitante();  
    telVisit.setVisible(true);  
}
```

```
private void AcLoginActionPerformed(java.awt.event.ActionEvent evt) {  
    TelaLogin telLog = new TelaLogin();  
    telLog.setVisible(true);  
}
```

```
private void SairInicialActionPerformed(java.awt.event.ActionEvent evt) {  
    this.dispose();  
}
```

## **Tela Cadastro:**

```
private void BotaoCadActionPerformed(java.awt.event.ActionEvent evt) {  
    JOptionPane.showMessageDialog(null,"Cadastrado com sucesso");  
}
```

```
private void CancelarCadActionPerformed(java.awt.event.ActionEvent  
evt) {  
    this.dispose();  
}
```

### Tela Login:

```
private void EntrarLoginActionPerformed(java.awt.event.ActionEvent evt) {
```

```
if(txtLogin.getText().equals("paulobg0")&&txtSenha.getText().equals("paulo
#13")){
```

```
JOptionPane.showMessageDialog(null, "Bem Vindo");
```

```
TelaSeguidor telSeg = new TelaSeguidor();
```

```
telSeg.setVisible(true);
```

```
}else{
```

```
if(txtLogin.getText().equals("evandro369")&&txtSenha.getText().equals("Evandro159")){
```

```
JOptionPane.showMessageDialog(null, "Bem Vindo");
```

```
TelaStreamer telStr = new TelaStreamer();
```

```
telStr.setVisible(true);
```

```
}else{
```

```
JOptionPane.showMessageDialog(null, "Acesso Negado!");
```

}

}

}

```
private void EsqueciSenhaActionPerformed(java.awt.event.ActionEvent
evt) {
```

```
TelaEsqueci telEsq = new TelaEsqueci();
```

```
telEsq.setVisible(true);
```

}

### **Tela Esqueci:**

```
private void SalvarSenhaNovaActionPerformed(java.awt.event.ActionEvent  
evt) {
```

```
    JOptionPane.showMessageDialog(null,"Senha alterada");  
}
```

```
private void  
CancelarSenhaNovaActionPerformed(java.awt.event.ActionEvent evt) {  
    this.dispose();  
}
```

## **Tela Seguidor:**

```
private void
AlterarPerfilSeguidorActionPerformed(java.awt.event.ActionEvent evt) {

TelaAlterar telAlt = new TelaAlterar();
telAlt.setVisible(true);
    }

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt)
    {
        // TODO add your handling code here:
    }

    private void SairSeguidorActionPerformed(java.awt.event.ActionEvent
evt) {
        this.dispose();
    }
```

### **Tela Alteração de Cadastro:**

```
private void SalvarAltActionPerformed(java.awt.event.ActionEvent evt) {  
    JOptionPane.showMessageDialog(null,"As alterações foram  
    realizadas");  
}  
  
private void CancelarAltActionPerformed(java.awt.event.ActionEvent evt)  
{  
    this.dispose();  
}
```

## **Tela Streamer:**

```
private void ConfigLiveActionPerformed(java.awt.event.ActionEvent evt) {  
    LiveConfig livConfig = new LiveConfig();  
    livConfig.setVisible(true);  
}  
  
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent  
evt) {  
    JOptionPane.showMessageDialog(null,"Transmissão iniciada");  
}  
  
private void FecharLiveActionPerformed(java.awt.event.ActionEvent evt)  
{  
    JOptionPane.showMessageDialog(null,"Transmissão encerrada");  
}  
  
private void ConfigBotActionPerformed(java.awt.event.ActionEvent evt) {  
    TelaBots telBot = new TelaBots();  
    telBot.setVisible(true);  
}
```

### **Tela Configuração da Live:**

```
private void SalvarConfigActionPerformed(java.awt.event.ActionEvent evt) {  
    JOptionPane.showMessageDialog(null,"Configurações aplicadas");  
}
```

```
private void CancelarConfigActionPerformed(java.awt.event.ActionEvent  
evt) {  
    this.dispose();  
}
```



## **Tela Bots:**

```
private void AddBotActionPerformed(java.awt.event.ActionEvent evt) {  
    JOptionPane.showMessageDialog(null,"Bot adicionado!");  
}
```

```
private void DelBotActionPerformed(java.awt.event.ActionEvent evt) {  
    JOptionPane.showMessageDialog(null,"Bot removido!");  
}
```

**Tela Visitante:**

```
private void CadVisitActionPerformed(java.awt.event.ActionEvent evt) {  
TelaCadastro telCad = new TelaCadastro();  
telCad.setVisible(true);
```

```
}
```

```
private void SairVisitActionPerformed(java.awt.event.ActionEvent evt) {  
this.dispose();  
}
```