SENAI - SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL - CFP132

ELAINE COELHO

JONAS JERÔNIMO CHAVES DE SOUSA

**EVENTVIEWER**

**SISTEMA DE GERENCIAMENTO DE PARTICIPANTES DE EVENTO**

SÃO PAULO

2014

ELAINE COELHO

JONAS JERÔNIMO CHAVES DE SOUSA

**EVENTVIEWER**

**SISTEMA DE GERENCIAMENTO DE PARTICIPANTES DE EVENTO**

**Documentação**

**Versão 1.6**

Trabalho de Conclusão de Curso do Ensino Técnico em Informática do SENAI – Sistema Nacional de Aprendizagem Industrial. Documentação do sistema de gerenciamento de participantes de evento EventViewer.

SÃO PAULO

2014

**Agradecimentos**

* Agradecemos à escola SENAI de informática e seu corpo docente pela experiência proporcionada;
* Agradecemos ao Anderson Rodrigues, caravanista que nos auxiliou nas pesquisas;
* Agradecemos aos colaboradores da empresa AR Sistemas pelo auxílio no desenvolvimento do produto

Histórico de revisões

| Data | Autor | Versão | Modificações Efetuadas |
| --- | --- | --- | --- |
| 05/09/2014 | Jonas e Elaine | 1.0 | Criação do documento |
| 24/10/2014 | Jonas | 1.1 | Alteração das regras de negócio. |
| 15/10/2014 | Elaine | 1.2 | Edição do Modelo Conceitual da base de dados |
| 28/10/2014 | Elaine | 1.3 | Edição do Diagrama de classes |
| 30/10/2014 | Jonas | 1.4 | Edição do diagrama de caso de uso e revisão de todas as narrativas de caso de uso. |
| 06/11/2014 | Elaine e Jonas | 1.5 | Edição dos Protótipos, retirada do Modelo de Domínio e edição do modelo Físico de base de dados |
| 23/11/2014 | Elaine e Jonas | 1.6 | Atualização dos requisitos funcionais, narrativas de uso, diagramas de atividade, diagrama de classe, diagrama de caso de uso, inserção de imagens do modelo conceitual e lógico da base de dados, atualização do modelo físico da base de dados, inserção de imagens do produto final, inserção de referências bibliográficas, atualização do ambiente de desenvolvimento, inserção dos casos de teste. |
| 27/11/2014 | Jonas e Elaine | 1.7 | Adição de regras de negócio, atualização de diagramas de atividade, de classe e de caso de uso, atualização das narrativas de caso de uso. |
| 28/11/2014 | Jonas | 1.8 | Atualização dos modelos lógico e conceitual da base de dados. |

SUMÁRIO

[**1.** **Introdução ao documento** 7](#_Toc405021512)

[**1.1.** **Propósito do documento** 7](#_Toc405021513)

[**1.2.** **Tema** 7](#_Toc405021514)

[**1.3.** **Público alvo** 7](#_Toc405021515)

[**1.4.** **Escopo do produto** 7](#_Toc405021516)

[**1.5.** **Convenções, termos e abreviações.** 7](#_Toc405021517)

[**1.5.1.** **Identificação das siglas** 7](#_Toc405021518)

[**1.5.2.** **Prioridade de requisitos** 8](#_Toc405021519)

[**2.** **Visão geral do produto** 9](#_Toc405021520)

[**2.1.** **Perspectiva do Produto** 9](#_Toc405021521)

[**2.2.** **Principais envolvidos no sistema e suas características** 9](#_Toc405021522)

[**2.3.** **Regras de Negócio** 9](#_Toc405021523)

[**3.** **REQUISITOS DO SISTEMA** 10](#_Toc405021524)

[**3.1.** **Requisitos Funcionais** 10](#_Toc405021525)

[**3.2.** **Requisito não funcional** 11](#_Toc405021526)

[**3.3.** **Premissas, Restrições e Dependências.** 11](#_Toc405021527)

[**4.** **Análise e Design** 12](#_Toc405021528)

[**4.1.** **Diagramas de Casos de Uso** 12](#_Toc405021529)

[**4.2.** **Narrativas de Casos de Uso** 13](#_Toc405021530)

[**4.3.** **Diagramas de atividade** 37](#_Toc405021531)

[**4.4.** **Diagramas de classes** 50](#_Toc405021532)

[**4.5.** **Modelos de dados** 51](#_Toc405021533)

[4.5.1. Modelo Lógico da base de dados 51](#_Toc405021534)

[4.5.2. Modelo Conceitual da base de dados 52](#_Toc405021535)

[4.5.3. Modelo Físico da base de dados 53](#_Toc405021536)

[**4.6.** **Ambiente de desenvolvimento** 68](#_Toc405021537)

[**5.** **Testes** 69](#_Toc405021538)

[**6.** **Protótipos** 73](#_Toc405021539)

[**7.** **Produto Final** 77](#_Toc405021540)

[**8.** **Manual do usuário** 85](#_Toc405021541)

[**9.** **Conclusões e considerações finais** 85](#_Toc405021542)

[**Referências bibliográficas** 85](#_Toc405021543)

[**Glossário** 85](#_Toc405021544)

[**Apêndice A – Plano de testes.** 86](#_Toc405021545)

[**Apêndice B – Manual do usuário.** 87](#_Toc405021546)

# **Introdução ao documento**

## **Propósito do documento**

Este documento tem como objetivo registrar as informações do desenvolvimento do sistema EventViewer, além de dar uma visão geral sobre sua estrutura e funcionamento.

## **Tema**

O sistema será utilizado para auxiliar organizadores de eventos de pequeno, médio ou grande porte, físicos ou virtuais, no gerenciamento dos participantes desses eventos.

## **Público alvo**

Este documento é destinado aos responsáveis pelo desenvolvimento do sistema ou àqueles que tenham interesse em consultar a documentação de um produto de software para fins escolares.

## **Escopo do produto**

O sistema EventViewer englobará o processo de registro de um evento, seu agendamento e registro de informações de seus participantes e convidados.

## **Convenções, termos e abreviações.**

Para uma melhor definição de alguns termos utilizados, o glossário no final nesse documento, deverá ser consultado.

## **Identificação das siglas**

Alguns requisitos e outros itens desse documento são identificados por siglas seguidas de numerações:

RNnº - Regra de negócio número.

RFnº - Requisito funcional número.

RNFnº - Requisito não funcional número.

CDUnº - Caso de uso número.

## **Prioridade de requisitos**

* Essencial: são requisitos imprescindíveis para o correto funcionamento do sistema.
* Importante: é possível utilizar o sistema sem esses requisitos, no entanto, a informação adquirida a partir dele pode ser insuficiente para uma utilização mais detalhista.
* Desejável: com esses requisitos implantados o sistema ficará mais rico em informações, mas sem ele o sistema ainda funcionará corretamente e fornecendo informações suficientes.

# **Visão geral do produto**

## **Perspectiva do Produto**

O sistema será desenvolvido objetivando proporcionar ao usuário final uma experiência completa e intuitiva para auxiliar no registro das informações de:

* Participantes de eventos, além do agrupamento desses participantes;
* Eventos físicos, festas, reuniões, etc, ou virtuais, hangouts, jogos online, etc;
* Convidados dos eventos;
* Locais desses eventos.

## **Principais envolvidos no sistema e suas características**

O programa EventViewer não é destinado a um perfil de usuário específico, portanto pode ser utilizado por qualquer indivíduo que deseje manter o gerenciamento informatizado de participantes de evento, como por exemplo organizadores de excursões, de eventos *onlines*, de palestras.

## **Regras de Negócio**

RN1 - Os participantes serão organizados em grupos para participar de um evento agendado.

RN2 - Todo evento agendado deve ter um local de realização.

RN3 - Todo evento agendado deve ter um grupo de participantes.

RN4 - Todo evento agendado deve ter uma data e horário de realização.

RN5 - O participante de um evento será identificado pelo nome.

RN6 - O grupo de participante será identificado pelo nome.

RN7 - O local de evento deve ter um endereço físico e/ou virtual.

RN8 - O evento será identificado pelo nome.

RN9 - O local de evento será identificado pelo nome.

RN10 - O evento agendado poderá ter convidados.

# **REQUISITOS DO SISTEMA**

## **Requisitos Funcionais**

|  |  |  |
| --- | --- | --- |
| **Função** | **Descrição** | **Casos de uso relacionados** |
| RF1 – Autenticação | O sistema irá utilizar autenticação de acesso mediante usuário e senha. | CDU1 |
| RF2 – Cadastro de usuário | O sistema proporcionará o cadastro de usuários do sistema pelo usuário avançado. | CDU2 |
| RF3 – Cadastro de local de evento | O sistema irá dispor de cadastro de local de evento pelo usuário avançado. | CDU3 |
| RF4 – Cadastro de evento | O sistema possibilitará o cadastro de evento pelo usuário avançado. | CDU4 |
| RF5 – Cadastro de participante | O sistema disponibilizará o cadastro de participantes de evento pelo usuário comum e avançado. | CDU5 |
| RF6 – Cadastro de grupo de participantes | O sistema proporcionará o registro de grupo de participantes pelo usuário comum e avançado. | CDU6 |
| RF7 – Cadastro de convidado | O sistema proporcionará o cadastro de convidado pelo usuário avançado. | CDU7 |
| RF8 – Agendamento de evento | O sistema irá proporcionar o controle de agendamento de eventos pelo usuário comum e avançado | CDU8, CDU11  CDU12, CDU13 |
| RF9 – Consultar evento | O sistema irá possibilitar a pesquisa de um evento específico. | CDU9 |
| RF10 – Consultar grupos | O sistema irá possibilitar a pesquisa de um grupo de participantes específico. | CDU10 |
| RF11 – Manter convidados do evento agendado | O sistema irá possibilitar a inclusão e remoção de um convidado da lista do evento agendado. | CDU11 |
| RF12 – Manter grupos do evento agendado | O sistema irá possibilitar a inclusão e remoção de um grupo da lista do evento agendado. | CDU12 |
| RF13 – Consultar local de evento | O sistema irá dispor de tela para consulta de local de evento. | CDU13 |
| RF14 – Consultar evento agendado | O sistema proporcionará função de consultar eventos agendados. | CDU14 |

## **Requisito não funcional**

RNF1 – O código do sistema será comentado para facilitar a manutenção do mesmo.

## **Premissas, Restrições e Dependências.**

| **Requisitos de *hardware* e *software*** | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Plataforma** | **Componentes** | **Requisito de processamento recomendado** | **Requisito de processamento mínimo** | **Requisito de memória recomendado** | **Requisito de memória mínimo** | **Requisito de espaço em disco** |
| Windows Vista ou superior. | .Net Framework 4.0 ou superior, MYSQL Server 5.6. | Processador Dual Core de 2 GHz ou equivalente. | Processador único, P4 de 1,8 GHz ou equivalente. | 4 GB. | 2 GB. | 1 GB. |

# **Análise e Design**

## **Diagramas de Casos de Uso**

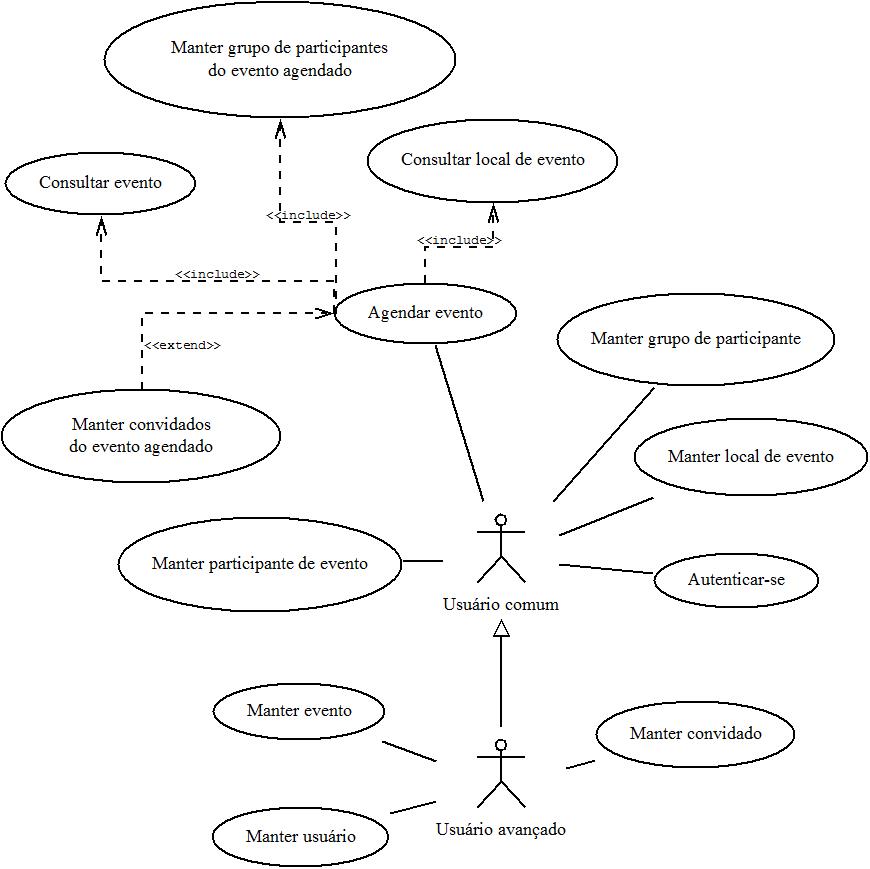


Figura 1 Diagrama de casos de uso do sistema EventViewer.

## **Narrativas de Casos de Uso**

Sigla: **CDU1.**

Nome: autenticar-se.

Objetivo: controlar o acesso de usuários ao sistema.

Atores: usuário comum e usuário avançado.

Prioridade: essencial.

Entradas: usuário e senha.

* Cenário principal

1. O caso de uso inicia quando o ator inicia o sistema.
2. O sistema exibe um formulário para ler o usuário e a senha.
3. O ator digita um usuário e uma senha e pressiona a opção entrar.
4. O sistema valida os dados.
5. O sistema exibe o formulário principal.

* Cenário alternativo

Não há.

* Cenário de exceção

“4.1” Falhas na validação de usuário e/ou senha:

a) Caso o usuário e/ou senha sejam inválidos o sistema exibirá o erro e voltará ao passo 2.

* Pré-condições

Não há.

* Pós-condições

O ator será autenticado e possuirá acesso ao sistema de acordo com o tipo de usuário.

* Saídas

Os dados do usuário autenticado serão informados ao sistema para liberar os acessos de acordo com seu tipo.

Sigla: **CDU2.**

Nome: manter cadastro de usuário.

Objetivo: gerenciar o cadastro de usuários do sistema.

Atores: usuário avançado.

Prioridade: importante.

Entradas: nome, usuário, senha, confirmar senha e status.

* Cenário principal

1. O caso de uso inicia quando o ator acessa o item “usuários” da tela principal do sistema.
2. O sistema exibe um formulário para gerenciar as operações de registrar, atualizar, excluir e consultar usuários do sistema.
3. O ator realiza uma operação.
4. Ao término da operação o sistema retorna ao passo 2.
5. Caso o ator selecione a opção fechar o caso de uso é encerrado e o sistema retorna à tela principal.

* Cenário alternativo

3.1 Registrar usuário

3.1.1 O ator informa os dados do usuário no formulário.

3.1.2 O ator seleciona a opção salvar.

3.1.3 O sistema valida os dados.

3.1.4 O sistema registra as informações na base de dados.

3.2 Consultar usuário

3.2.1 O ator digita o usuário ou o nome no campo buscar.

3.2.2 O ator pressiona a tecla “enter” do teclado.

3.2.3 O sistema realiza uma pesquisa na base de dados.

3.2.4 O sistema exibe a lista de usuários com os resultados da busca

3.2.5 O ator seleciona, com duplo clique, um usuário na lista e o sistema exibe nos campos do formulário seus dados cadastrados.

2.3 Atualizar usuário

3.3.1 Executar Cenário alternativo 3.2

3.3.2 O ator informa os novos dados do usuário no formulário.

3.3.3 O ator seleciona a opção salvar.

3.3.4 O sistema valida os dados.

3.3.5 O sistema atualiza as informações do usuário na base de dados.

3.4 Excluir usuário

3.4.1 Executar Cenário alternativo 3.2.

3.4.2 Com duplo clique, o ator seleciona um usuário na lista.

3.4.3 O sistema exibe nos campos do formulário os dados cadastrados do usuário.

3.4.4 O ator seleciona a opção excluir.

3.4.5 O ator confirma a operação.

3.4.6 O sistema valida a exclusão.

2.4.7 O sistema exclui as informações do usuário da base de dados.

* Cenário de exceção

“3.1.3”, “3.3.4” Falhas na validação de dados.

a) Caso haja algum dado inválido, duplicado ou nulo o sistema exibirá o erro e voltará ao passo anterior.

“3.4.6” Falha na validação da exclusão

a) Caso um usuário comum tenha agendado algum evento no sistema ele não poderá ser excluído.

* Pré-condições

Não há.

* Pós-condições

Caso o usuário cadastrado possua status ativo ele estará apto para autenticação no sistema.

* Saídas

Não há.

Sigla: **CDU3**.

Nome: manter local de evento.

Objetivo: gerenciar o cadastro de locais de evento.

Atores: usuário avançado.

Prioridade: essencial.

Entradas: nome do local, endereço físico (cep, país, estado, cidade, bairro, logradouro, número da residência, complemento), endereço virtual e status.

* Cenário principal

1. O caso de uso inicia quando o ator acessa o item “locais de eventos” na tela principal do sistema.
2. O sistema exibe um formulário para gerenciar as operações de registrar, atualizar, excluir e um botão “consultar locais de eventos” que abrirá um novo formulário somente com a finalidade de consulta.
3. O ator realiza uma operação.
4. Ao término da operação o sistema retorna ao passo 2.
5. Caso o ator selecione a opção fechar o caso de uso é encerrado e o sistema retorna à tela inicial.

* Cenário alternativo

3.1 Registrar local de evento

3.1.1 O ator informa os dados do local de evento no formulário.

3.1.2 O ator seleciona a opção salvar.

3.1.3 O sistema valida os dados.

3.1.4 O sistema registra as informações na base de dados.

3.2 Consultar local de evento

3.2.1 No formulário de locais de eventos, o ator pressiona o botão consultar locais de eventos.

3.2.2 O sistema executa consultar local de evento [CDU13].

3.3 Atualizar local de evento

3.3.1 Executar cenário alternativo 2.2.

3.3.2 O ator informa os dados atualizados do local de evento no formulário.

3.3.3 O ator seleciona a opção salvar.

3.3.4 O sistema valida os dados.

2.3.5 O sistema atualiza as informações do local de evento na base de dados.

3.4 Excluir local de evento

3.4.1 Executar cenário alternativo 2.2.

3.4.2 O sistema exibe nos campos do formulário os dados cadastrados do local de evento.

3.4.3 O ator seleciona a opção excluir.

3.4.4 O ator confirma a operação.

3.4.5 O sistema valida a exclusão.

3.4.6 O sistema exclui as informações do local de evento da base de dados.

* Cenário de exceção

“3.1.3” e “3.3.4” Falhas na validação de dados.

a) Caso haja algum dado inválido, duplicado ou nulo o sistema exibirá o erro e voltará ao passo anterior.

“3.4.5” Falha na exclusão do local de evento

a) Caso o local de evento esteja vinculado à um evento agendado não será possível excluí-lo.

* Pré-condições

Não há.

* Pós-condições

Caso o local de evento cadastrado possua status ativo poderá ser agendado eventos nesse local.

* Saídas

Não há.

Sigla: **CDU4**.

Nome: manter evento.

Objetivo: gerenciar o cadastro de eventos.

Atores: usuário avançado.

Prioridade: essencial.

Entradas: nome, logotipo, descrição e status.

* Cenário principal

1. O caso de uso inicia quando o ator acessa o item “eventos” na tela principal do sistema.
2. O sistema exibe um formulário para gerenciar as operações de registrar, atualizar, excluir e um botão “consultar eventos” que abrirá um novo formulário somente com a finalidade de consulta.
3. O ator realiza uma operação.
4. Ao término da operação o sistema retorna ao passo 2.
5. Selecionando a opção fechar o caso de uso é encerrado e o sistema retorna à tela inicial.

* Cenário alternativo

3.1 Registrar evento

3.1.1 O ator informa os dados do evento no formulário.

3.1.2 O ator seleciona a opção salvar.

3.1.3 O sistema valida os dados.

3.1.4 O sistema registra as informações na base de dados.

3.2 Consultar evento

3.2.1 No formulário de eventos, o ator pressiona o botão consultar eventos.

3.2.2 Executar consultar evento [CDU9].

3.3 Atualizar evento

3.3.1 Executar cenário alternativo 3.2.

3.3.2 O ator informa os dados atualizados do evento no formulário.

3.3.3 O ator seleciona a opção salvar.

3.3.4 O sistema valida os dados.

3.3.5 O sistema atualiza as informações do evento na base de dados.

3.4 Excluir evento

3.4.1 Executar cenário alternativo 3.2.

3.4.2 O ator seleciona a opção excluir.

3.4.3 O ator confirma a operação.

3.4.4 O sistema valida a exclusão.

3.4.5 O sistema exclui as informações do evento da base de dados.

* Cenário de exceção

“3.1.3” e “3.3.4” Falha na validação de dados.

a) Caso haja algum dado inválido, duplicado ou nulo o sistema exibirá o erro e voltará ao passo anterior.

“3.4.4” Falha na validação de exclusão de evento

a) Caso haja algum agendamento feito para o evento, não será possível excluí-lo.

* Pré-condições

Não há.

* Pós-condições

Caso o evento cadastrado possua status ativo poderá ser agendado no sistema.

* Saídas

Não há.

Sigla: **CDU5**.

Nome: manter participante.

Objetivo: gerenciar o cadastro de participantes de evento.

Atores: usuário avançado e usuário comum.

Prioridade: essencial.

Entradas: nome, identidade, cpf, rg, telefone residencial, celular, *e-mail*, foto, cep, país, estado, município, bairro, logradouro, número da residência, complemento e status.

* Cenário principal

1. O caso de uso inicia quando o ator acessa o item “participantes” na tela principal do sistema.
2. O sistema exibe um formulário para gerenciar as operações de registrar, atualizar, excluir e consultar participante de evento.
3. O ator realiza uma operação.
4. Ao término da operação o sistema retorna ao passo 2.
5. Caso o ator selecione a opção fechar o caso de uso é encerrado e o sistema retorna à tela inicial.

* Cenário alternativo

3.1 Registrar participantes

3.1.1 O ator informa os dados do participante de evento no formulário.

3.1.2 O ator seleciona a opção salvar.

3.1.3 O sistema valida os dados.

3.1.4 O sistema registra as informações na base de dados.

3.2 Consultar participantes

3.2.1 O ator digita o nome no campo buscar.

3.2.2 O ator pressiona a tecla “enter” do teclado.

3.2.3 O sistema realiza uma pesquisa na base de dados.

3.2.4 O sistema exibe a lista de participantes com os resultados da pesquisa

3.2.5 O ator seleciona um participante de evento na lista.

3.2.6 Com duplo clique no participante selecionado, o sistema exibe seus dados cadastrados nos campos do formulário.

3.3 Atualizar participantes

3.3.1 Executar cenário alternativo 3.2.

3.3.2 O ator informa os dados atualizados do participante no formulário.

3.3.3 O ator seleciona a opção salvar.

3.3.4 O sistema valida os dados.

3.3.5 O sistema atualiza as informações do participante de evento na base de dados.

3.4 Excluir participante de evento

3.4.1 Executar cenário alternativo 3.2.

3.4.2 O ator seleciona a opção excluir.

3.4.3 O sistema valida a exclusão.

3.4.4 O sistema exclui as informações do participante de evento da base de dados.

* Cenário de exceção

“3.1.3” e “3.3.4” Falha na validação de dados.

a) Caso haja algum dado inválido, duplicado ou nulo o sistema exibirá o erro e voltará ao passo anterior.

“3.4.3” Falha na validação de exclusão.

a) Caso o participante de evento esteja em um grupo que está vinculado a um agendamento de evento não será possível excluí-lo.

* Pré-condições

Não há.

* Pós-condições

Caso o participante de evento cadastrado possua status ativo poderá ser adicionado a grupos de participantes no sistema.

* Saídas

Não há.

Sigla: **CDU6**.

Nome: manter grupo

Objetivo: gerenciar o cadastro de grupos.

Atores: usuário avançado e usuário comum.

Prioridade: essencial.

Entradas: nome do grupo, participantes, descrição, status e criar grupo.

* Cenário principal

1. O caso de uso inicia quando o ator acessa o item “grupos” na tela principal do sistema.

2. O sistema exibe um formulário para gerenciar as operações de registrar, atualizar, deixar inativo e um botão “consultar eventos” que abrirá um novo formulário somente com a finalidade de consulta.

1. O ator realiza uma operação.
2. Ao término da operação o sistema retorna ao passo 2.
3. Caso o ator selecione a opção fechar o caso de uso é encerrado e o sistema retorna à tela inicial.

* Cenário alternativo

3.1 Registrar grupo de participante

3.1.1 O ator seleciona um participante que deseja inserir no novo grupo na lista da direita e os insere na lista à esquerda do formulário.

3.1.2 O ator clica no botão salvar grupo.

3.1.3 O sistema valida os dados.

3.1.4 O sistema registra as informações na base de dados.

3.2 Consultar grupos

3.2.1 O ator clica no botão “consultar grupos”.

3.2.2 Executar consultar grupos [CDU10].

3.3 Atualizar grupos

3.3.1 Executar cenário alternativo 3.2.

3.3.2 O ator informa os dados atualizados do grupo no formulário.

3.3.3 O ator clica no botão “salvar grupo”.

3.3.4 O sistema valida os dados.

3.3.5 O sistema atualiza as informações do participante de evento na base de dados.

3.4 Inativar grupo

3.4.1 O ator seleciona um grupo na lista.

3.4.2 Com um duplo clique, o sistema exibe nos campos do formulário “grupos” os dados cadastrados do grupo selecionado.

3.4.3 O ator altera o status de ativo para inativo, nos dados do grupo no formulário.

3.4.4 O ator clica no botão “salvar grupo”.

3.4.5 O sistema valida os dados.

3.4.6 O sistema atualiza as informações do grupo na base de dados.

* Cenário de exceção

“3.1.3” e “3.3.4” Falha na validação de dados.

a) Caso haja algum dado inválido, duplicado ou nulo o sistema exibirá o erro e voltará ao passo anterior.

* Pré-condições

O tipo de grupo e os participantes deverão estar com o cadastro ativo no sistema para serem adicionados a um grupo.

* Pós-condições

Caso o grupo de participantes cadastrado possua status ativo poderá ser adicionado a um agendamento de evento.

* Saídas

Não há.

Sigla: **CDU7**.

Nome: manter o convidado.

Objetivo: gerenciar o cadastro de convidados do sistema.

Atores: usuário comum e avançado.

Prioridade: desejável.

Entradas: nome, descrição e status.

* Cenário principal

1. O caso de uso inicia quando o ator acessa o item “convidados” da tela principal do sistema.
2. O sistema exibe um formulário para gerenciar as operações de registrar, atualizar, excluir e consultar os convidados do sistema.
3. O ator realiza uma operação.
4. Ao término da operação o sistema retorna ao passo 2.
5. Caso o ator selecione a opção fechar o caso de uso é encerrado e o sistema retorna à tela inicial.

* Cenário alternativo

3.1 Registrar o convidado

3.1.1 O ator informa os dados do convidado no formulário.

3.1.2 O ator seleciona a opção salvar.

3.1.3 O sistema valida os dados.

3.1.4 O sistema registra as informações na base de dados.

3.2 Consultar o convidado

3.2.1 O ator digita o nome do convidado no campo buscar.

3.2.2 O ator pressiona o botão “enter” do teclado.

3.2.3 O sistema realiza uma pesquisa na base de dados.

3.2.4 O sistema exibe a lista de convidados com os resultados da busca.

3.2.5 O ator seleciona um convidado na lista.

3.2.6 Com duplo clique, o sistema exibe nos campos do formulário os dados cadastrados.

3.3 Atualizar o convidado

3.3.1 Executar cenário alternativo 3.2.

3.3.2 O ator informa os dados atualizados do convidado no formulário.

3.3.3 O ator clica no botão salvar.

3.3.4 O sistema valida os dados.

3.3.5 O sistema atualiza as informações do convidado na base de dados.

3.4 Excluir o convidado

3.4.1 Executar cenário alternativo 3.2.

3.4.2 O ator seleciona a opção excluir.

3.4.3 O ator confirma a operação.

3.4.4 O sistema valida a exclusão.

3.4.5 O sistema exclui as informações do convidado da base de dados.

* Cenário de exceção

“3.1.3” e “3.3.4” Falhas na validação de dados.

a) Caso haja algum dado inválido, duplicado ou nulo o sistema exibirá o erro e voltará ao passo anterior.

“3.4.4” Falha na exclusão do convidado

a) Caso o convidado esteja registrado em um agendamento de evento, não poderá ser excluído.

* Pré-condições

Não há.

* Pós-condições

Caso o convidado cadastrado possua status ativo ele estará apto para ser utilizado no agendamento de evento.

* Saídas

Não há.

Sigla: **CDU8**.

Nome: agendar evento.

Objetivo: gerenciar o cadastro de eventos agendados no sistema.

Atores: usuário comum e avançado.

Prioridade: importante.

Entradas: evento, local de evento, capacidade de participantes, grupos de participante, convidados, descrição, data do evento, horário de encontro, horário de término, e status.

* Cenário principal

1. O caso de uso inicia quando o ator acessa o item “agendar eventos” da tela principal do sistema.
2. O sistema exibe um formulário para gerenciar as operações de registrar, atualizar, excluir os agendamentos de eventos do sistema.
3. O ator realiza uma operação.
4. Ao término da operação o sistema retorna ao passo 2.
5. Caso o ator selecione a opção fechar o caso de uso é encerrado e o sistema retorna à tela inicial.

* Cenário alternativo

3.1 Registrar o agendamento de evento

3.1.1 O ator pressiona o botão consultar evento.

3.1.2 Executar consultar evento (CDU9).

3.1.3 O ator pressiona o botão consultar local de evento.

3.1.4 Executar consultar local de evento (CDU13).

3.1.5 O ator pressiona o botão consultar grupos.

3.1.6 Executar manter grupos do evento agendado (CDU12).

3.1.7 Caso necessário, o ator pressiona botão consultar convidados e executar manter convidados do evento agendado (CDU11).

3.1.8 O ator informa os dados para agendamento de evento no formulário.

3.1.9 O ator seleciona a opção agendar evento.

3.1.10 O sistema valida os dados.

3.1.11 O sistema registra as informações na base de dados.

3.3 Atualizar o agendamento de evento

3.3.1 Executar consultar eventos agendados [CDU14].

3.3.2 O ator informa os dados atualizados do agendamento do evento no formulário.

3.3.3 O ator clica no botão “salvar alteração”.

3.3.4 O sistema valida os dados.

3.3.5 O sistema atualiza as informações do evento agendado na base de dados.

3.4 Encerrar o evento agendado

3.4.1 Executar consultar eventos agendados [CDU14].

2.4.3 O ator seleciona o status “Encerrado”.

2.4.4 O sistema atualiza o status na base de dados.

* Cenário de exceção

“3.1.10” e “3.3.4” Falhas na validação de dados.

a) Caso haja algum dado inválido, duplicado ou nulo o sistema exibirá o erro e voltará ao passo anterior.

* Pré-condições

O evento, os grupos de participantes, os convidados e o local do evento deverão estar cadastrados no sistema com o status ativo para serem utilizados no agendamento.

* Pós-condições

O evento agendado será listado na agenda de eventos.

* Saídas

Não há.

Sigla: **CDU9**.

Nome: consultar evento.

Objetivo: consultar os eventos cadastrados no sistema.

Atores: usuário comum e avançado.

Prioridade: importante.

Entradas: nome do evento.

* Cenário principal

1. O caso de uso inicia quando o ator seleciona a opção “consultar evento” da tela de “eventos” ou da tela de “agendar eventos”.
2. O sistema exibe um formulário para digitar o nome do evento a ser consultado na barra de busca.
3. O ator aperta a tecla “enter do teclado”.
4. O sistema exibe os resultados da busca, na lista de eventos cadastrados.
5. O ator seleciona o evento desejado na lista.
6. Com duplo clique, o sistema abre e exibe os dados do evento selecionado nos campos da tela de início da consulta, “eventos” ou “agendar eventos”.
7. Caso o ator selecione a opção fechar a tela utilizada na consulta, o sistema retorna à tela principal.

* Cenário alternativo

Não há.

* Cenário de exceção

Não há.

* Pré-condições

Não há.

* Pós-condições

Não há.

* Saídas

Informações do evento selecionado.

Sigla: **CDU10**.

Nome: consultar grupo.

Objetivo: consultar os grupos cadastrados no sistema.

Atores: usuário comum e avançado.

Prioridade: essencial.

Entradas: nome do grupo de participantes.

* Cenário principal

1. O caso de uso inicia quando o ator seleciona a opção “consultar grupo de participantes” da tela de “grupos”.
2. O sistema exibe um formulário para digitar o nome do grupo a ser consultado na barra de busca.
3. O ator aperta a tecla “enter do teclado”.
4. O sistema exibe os resultados da busca, na lista de grupos cadastrados.
5. O ator seleciona o grupo desejado na lista.
6. Com duplo clique, o sistema abre e exibe os dados do grupo selecionado nos campos da tela de início da consulta, “grupos” ou “agendar eventos”.
7. Caso o ator selecione a opção fechar a tela utilizada na consulta, o sistema retorna à tela principal.

* Cenário alternativo

Não há.

* Cenário de exceção

Não há.

* Pré-condições

Não há.

* Pós-condições

Não há.

* Saídas

Informações dos grupos de participantes selecionados.

Sigla: **CDU11**.

Nome: manter convidados do evento agendado.

Objetivo: manter os convidados do evento agendado no sistema.

Atores: usuário comum e avançado.

Prioridade: importante.

Entradas: nome do convidado.

* Cenário principal

1. O caso de uso inicia quando o ator seleciona a opção “consultar convidados” da tela de agendamento de eventos.
2. O sistema exibe um formulário para gerenciar as operações de adicionar e remover os convidados do evento agendado.
3. O ator realiza uma operação.
4. Ao término da operação o sistema retorna ao passo 2
5. Caso o ator selecione a opção “OK” o caso de uso é encerrado e o sistema retorna à tela de “agendar eventos”.

* Cenário alternativo
  1. Adicionar convidado ao evento agendado
     1. O ator digita todo ou parte do nome do convidado na barra de busca.

3.1.2 O ator aperta a tecla “enter do teclado”

3.1.3 O sistema exibe os resultados da busca, na lista de convidados cadastrados.

* + 1. O ator seleciona um convidado na lista de resultados da busca.

3.1.5 Com duplo clique, o sistema adiciona o convidado à lista de convidados do evento agendado.

3.1.6 Caso o ator selecione a opção fechar a tela utilizada na consulta, o sistema retorna à tela principal.

* 1. Remover convidado do evento agendado
     1. O ator seleciona um convidado na lista de convidados do evento agendado.
     2. Com duplo clique, o sistema remove o convidado da lista de convidados do evento agendado.
     3. Caso o ator selecione a opção fechar a tela utilizada na consulta, o sistema retorna à tela principal.
* Cenário de exceção

“3.1.5” Falha na adição de convidado.

1. Caso o convidado já esteja na lista do evento agendado não será possível adicioná-lo novamente.

* Pré-condições

Não há.

* Pós-condições

Não há.

* Saídas

Informações dos convidados selecionados.

Sigla: **CDU12**.

Nome: manter grupos do evento agendado.

Objetivo: manter os grupos do evento agendado no sistema.

Atores: usuário comum e avançado.

Prioridade: importante.

Entradas: nome do grupo de participante.

* Cenário principal

1. O caso de uso inicia quando o ator seleciona a opção “consultar grupos de participante” da tela de agendamento de eventos.
2. O sistema exibe um formulário para gerenciar as operações de adicionar e remover os grupos de participante do evento agendado.
3. O ator realiza uma operação.
4. Ao término da operação o sistema retorna ao passo 2.
5. Caso o ator selecione a opção “OK” o caso de uso é encerrado e o sistema retorna à tela de “agendar eventos”.

* Cenário alternativo

3.1 Adicionar grupo de participante ao evento agendado

* + 1. O ator digita todo ou parte do nome do grupo na barra de busca.
    2. O ator aperta a tecla “enter do teclado”
    3. O sistema exibe os resultados da busca, na lista de grupos cadastrados.
    4. O ator seleciona um grupo na lista de resultados da busca.

3.1.5 Com duplo clique, o sistema adiciona o grupo à lista de grupos do evento agendado.

3.1.6 Caso o ator selecione a opção fechar a tela utilizada na consulta, o sistema retorna à tela principal.

* 1. Remover grupo do evento agendado
     1. O ator seleciona um grupo na lista de grupos do evento agendado.
     2. Com duplo clique, o sistema remove o grupo da lista de grupos do evento agendado.
     3. Caso o ator selecione a opção fechar a tela utilizada na consulta, o sistema retorna à tela principal.
* Cenário de exceção

“3.1.5” Falha na adição de grupo.

1. Caso o grupo já tenha sido adicionado, não será possível adicioná-lo novamente.

“3.2.2” Falha na remoção de grupo.

1. Caso seja o único grupo da lista, não será possível removê-lo.

* Pré-condições

Não há.

* Pós-condições

Não há.

* Saídas

Informações dos grupos selecionados.

Sigla: **CDU13**.

Nome: consultar local de evento.

Objetivo: consultar os locais de evento cadastrados no sistema.

Atores: usuário comum e avançado.

Prioridade: importante.

Entradas: nome do local de evento.

* Cenário principal

1. O caso de uso inicia quando o ator seleciona a opção “consultar local de evento” da tela de “cadastro de local de evento” ou da tela de “agendar eventos”.
2. O sistema exibe um formulário para digitar o nome do local de evento a ser consultado na barra de busca.
3. O ator aperta a tecla “enter do teclado”.
4. O sistema exibe os resultados da busca, na lista de locais de evento cadastrados.
5. O ator seleciona o local de evento desejado na lista.
6. Com duplo clique, o sistema abre e exibe os dados local de evento selecionado nos campos da tela de início da consulta, seja “local de evento” ou “agendar eventos”.
7. Caso o ator selecione a opção fechar a tela utilizada na consulta, o sistema retorna à tela principal.

* Cenário alternativo

Não há.

* Cenário de exceção

Não há.

* Pré-condições

Não há.

* Pós-condições

Não há.

* Saídas

Informações do local de evento selecionado.

Sigla: **CDU14**.

Nome: consultar eventos agendados.

Objetivo: consultar os locais de eventos agendados no sistema.

Atores: usuário comum e avançado.

Prioridade: essencial.

Entradas: data inicial, data final.

* Cenário principal

1. O caso de uso inicia quando o ator seleciona a opção “consultar eventos agendados” da tela de inicial.
2. O ator informa os parâmetros de busca.
3. O ator aperta a tecla “buscar eventos agendados”.
4. O sistema exibe os resultados da busca dos eventos agendados.
5. O ator seleciona o evento agendado desejado na lista.
6. Com o clique sobre uma informação do evento agendado, o sistema abre e exibe os dados do item selecionado.
7. Caso o ator selecione a opção fechar, o sistema é encerrado.

* Cenário alternativo

6.1 Exibir cadastro do evento

6.1.1 Com o clique sobre o logotipo do evento, o sistema executa manter evento (CDU4).

6.1.2 O sistema retorna ao passo 4.

6.2 Exibir cadastro do agendamento de evento

6.2.1 Com o clique sobre a data do agendamento ou horário ou sobre o status da capacidade, o sistema executa agendar evento (CDU8).

6.2.2 O sistema retorna ao passo 4.

6.3 Exibir cadastro de local do evento

6.3.1 Com o clique sobre o local do evento, o sistema executa manter local de evento (CDU13).

6.3.2 O sistema retorna ao passo 4.

* Cenário de exceção

Não há.

* Pré-condições

Não há.

* Pós-condições

Não há.

* Saídas

Informações do evento agendado selecionado.

## **Diagramas de atividade**

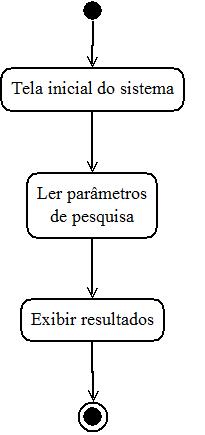


Figura 2 - Diagrama de atividade de consulta de eventos agendados no sistema.

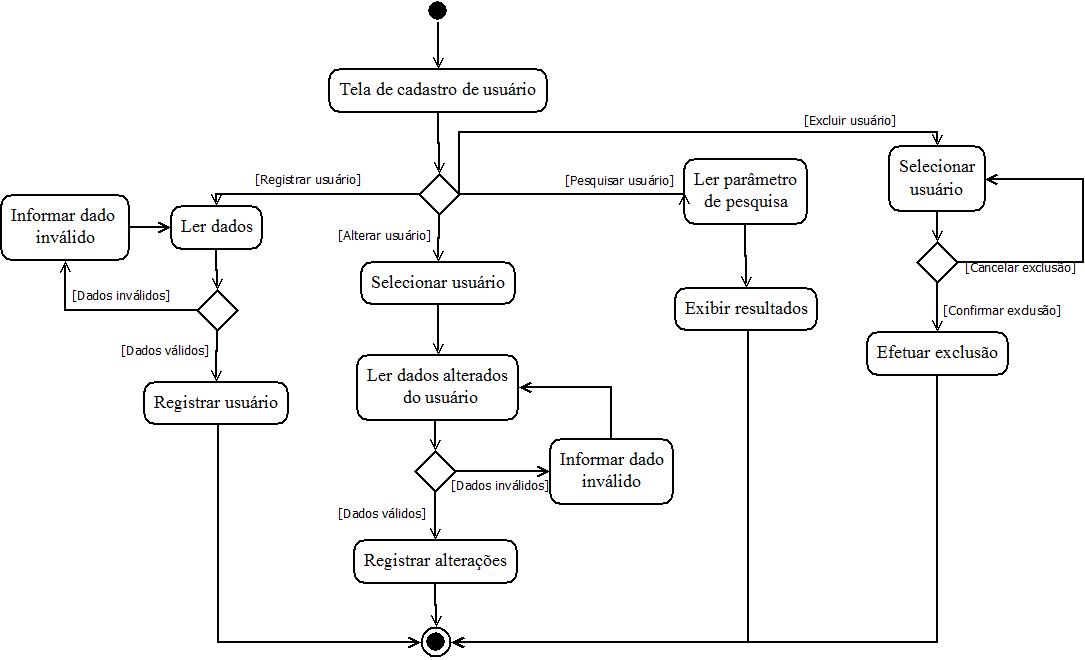


Figura 3 - Diagrama de Atividade para Manter Usuário no sistema.

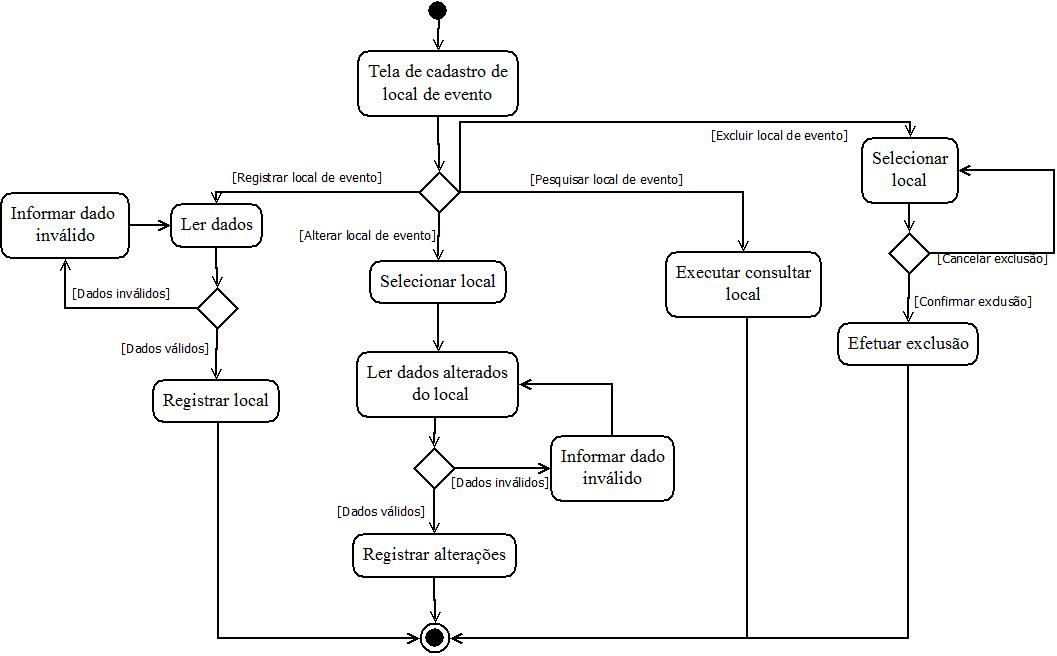


Figura 4 - Diagrama de Atividade para Manter Local de Evento no sistema.

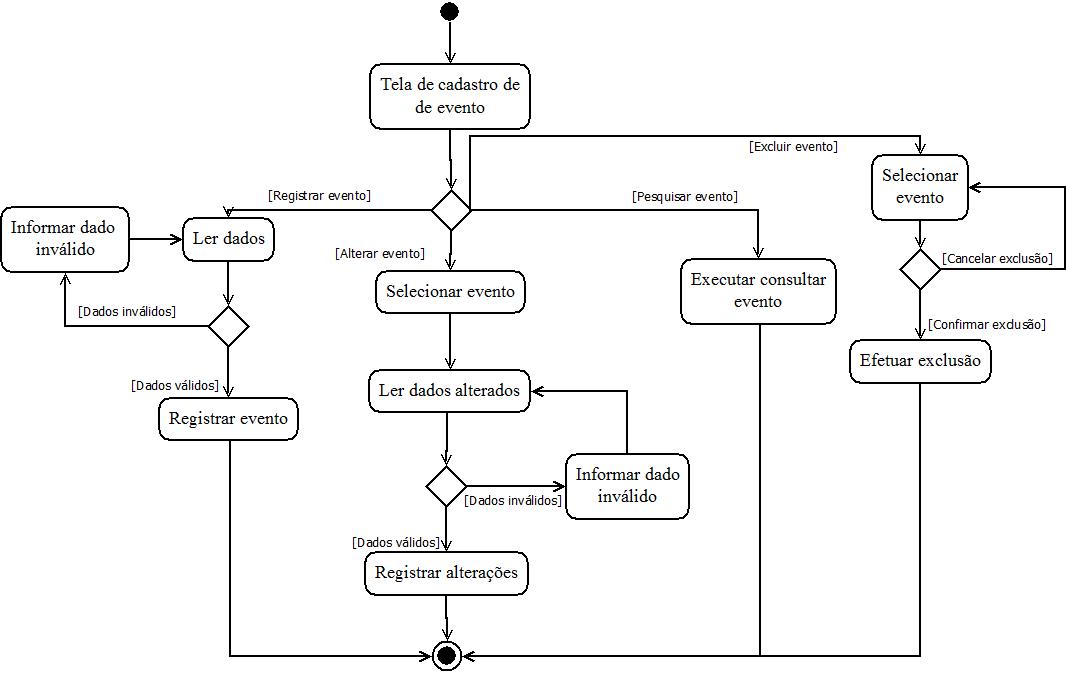


Figura 5 Diagrama de Atividade para Manter Evento no sistema.

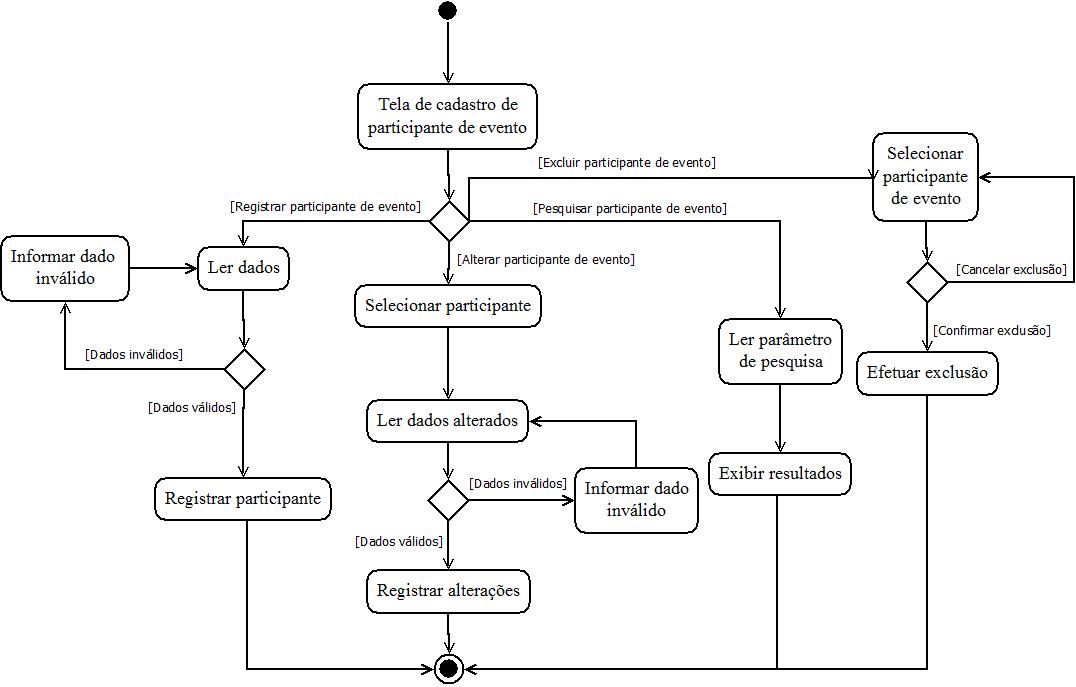


Figura 6 Diagrama de Atividade para Manter Participante de Evento no sistema

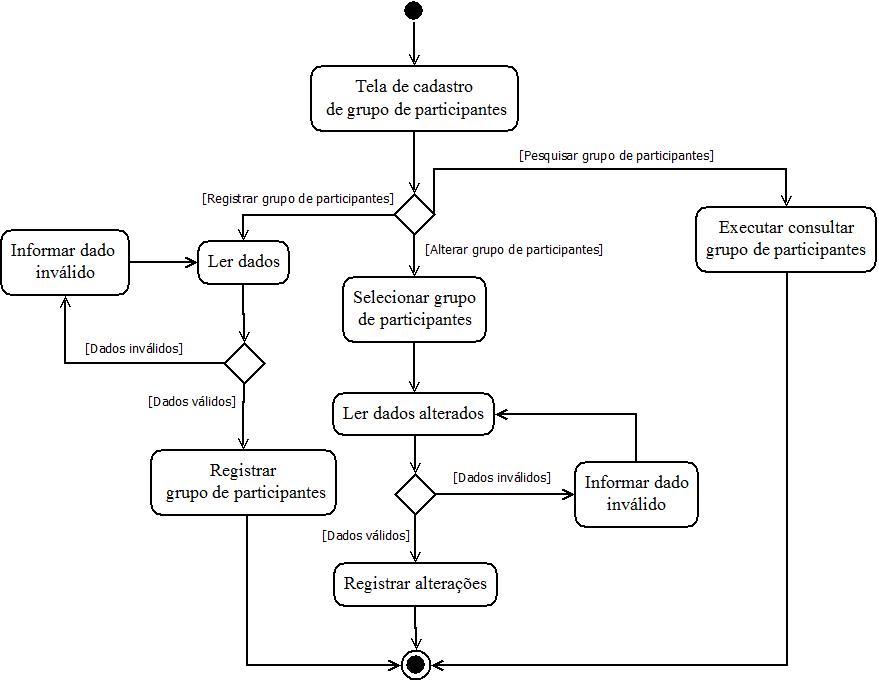


Figura 7 Diagrama de Atividade para Manter Grupo de Participante no sistema.

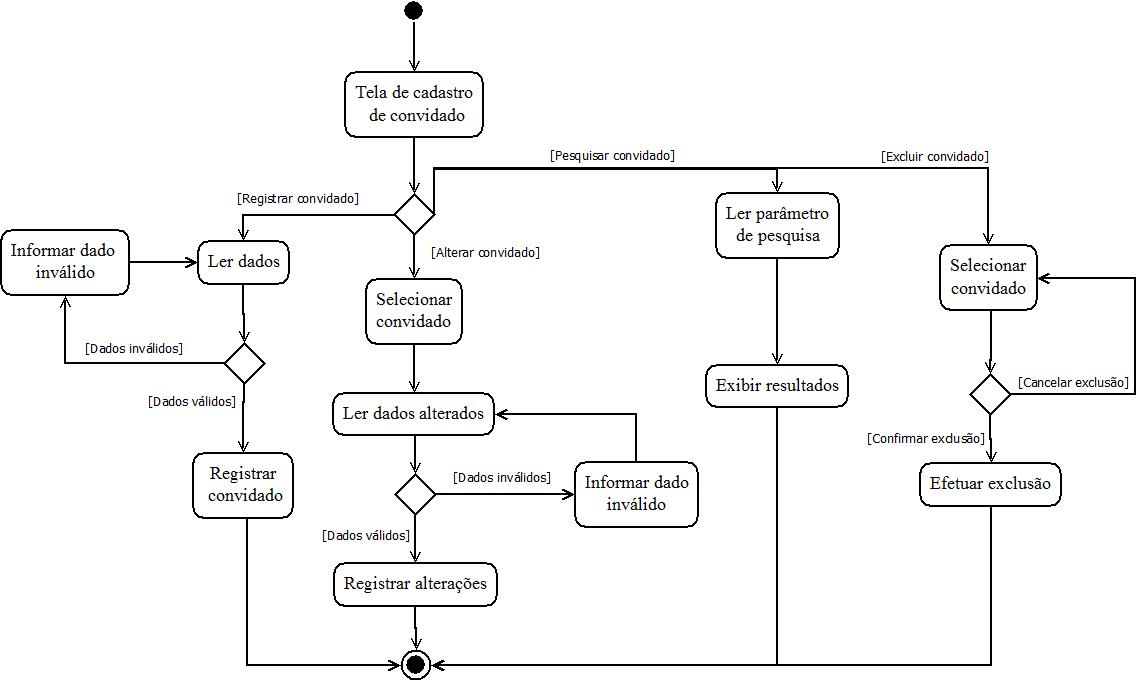


Figura 8 Diagrama de Atividade para Manter Convidado no sistema.

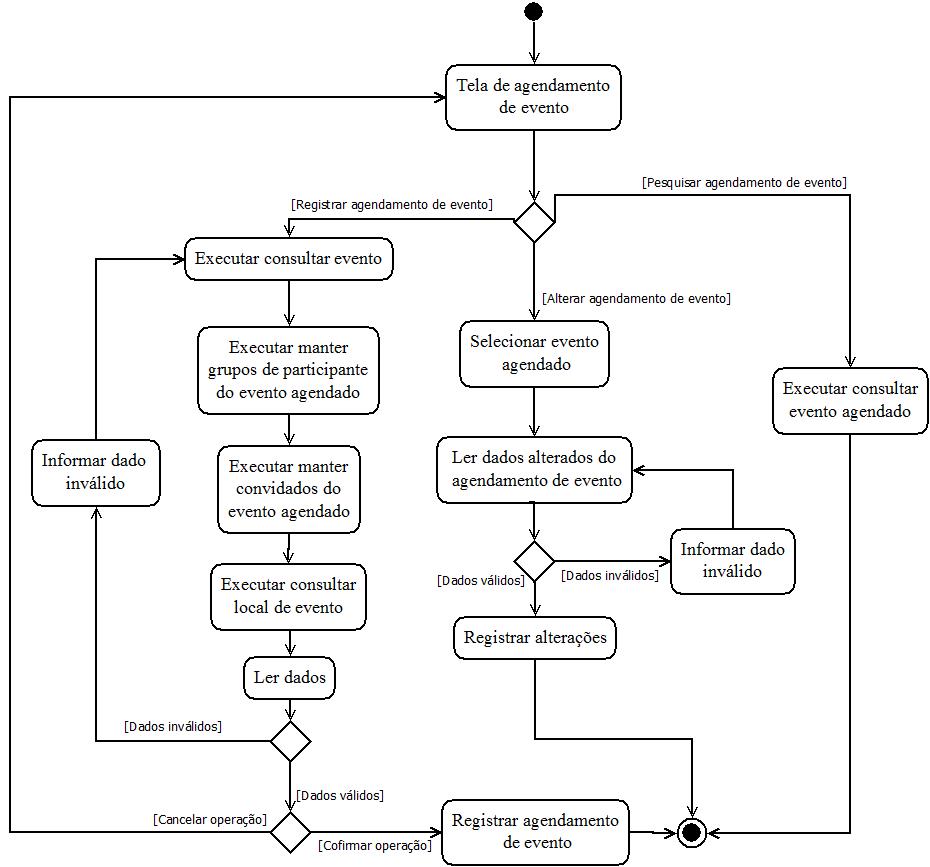


Figura 9 Diagrama de Atividade para Agendar evento no sistema

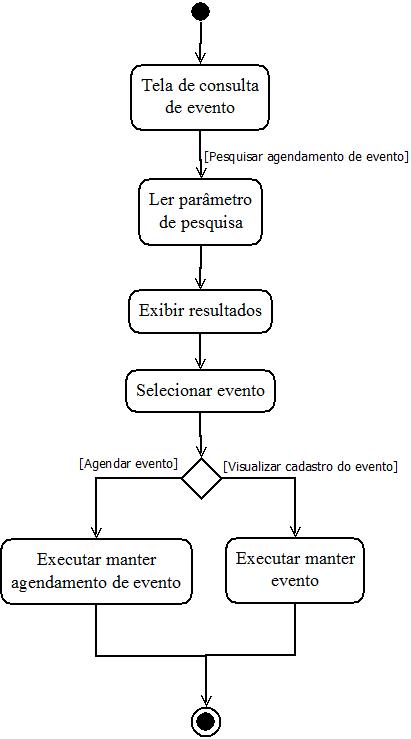


Figura 10 Diagrama de Atividade para Consultar Evento no sistema

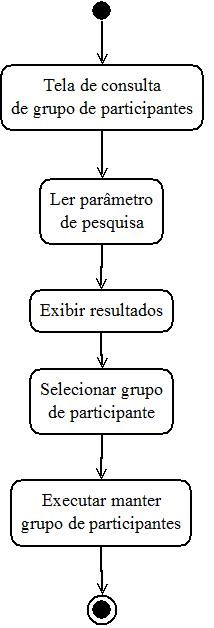


Figura 11 Diagrama de Atividade para Consultar Grupo de Participante no sistema.

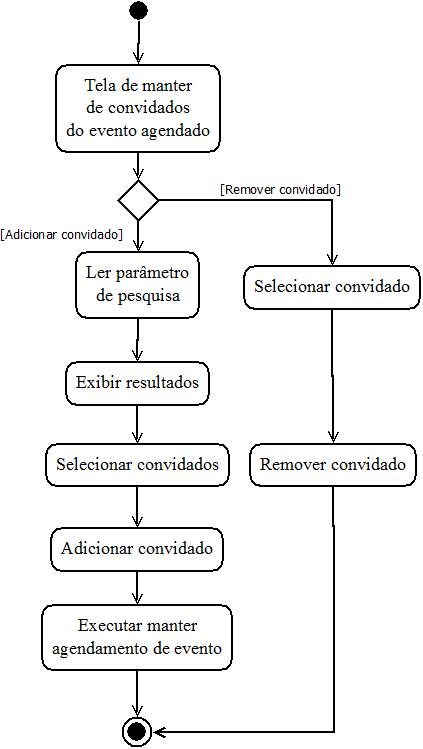


Figura 12 Diagrama de Atividade para manter Convidados do evento agendado no sistema.



Figura 13 Diagrama de Atividade para Manter Grupos do evento agendado no sistema.

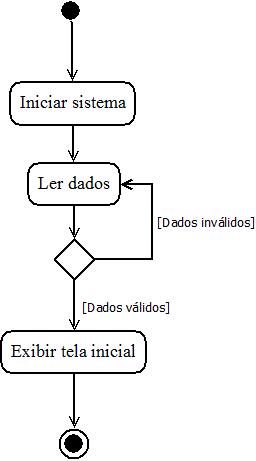


Figura 14 Diagrama de Atividade para autenticação no sistema.

## **Diagramas de classes**

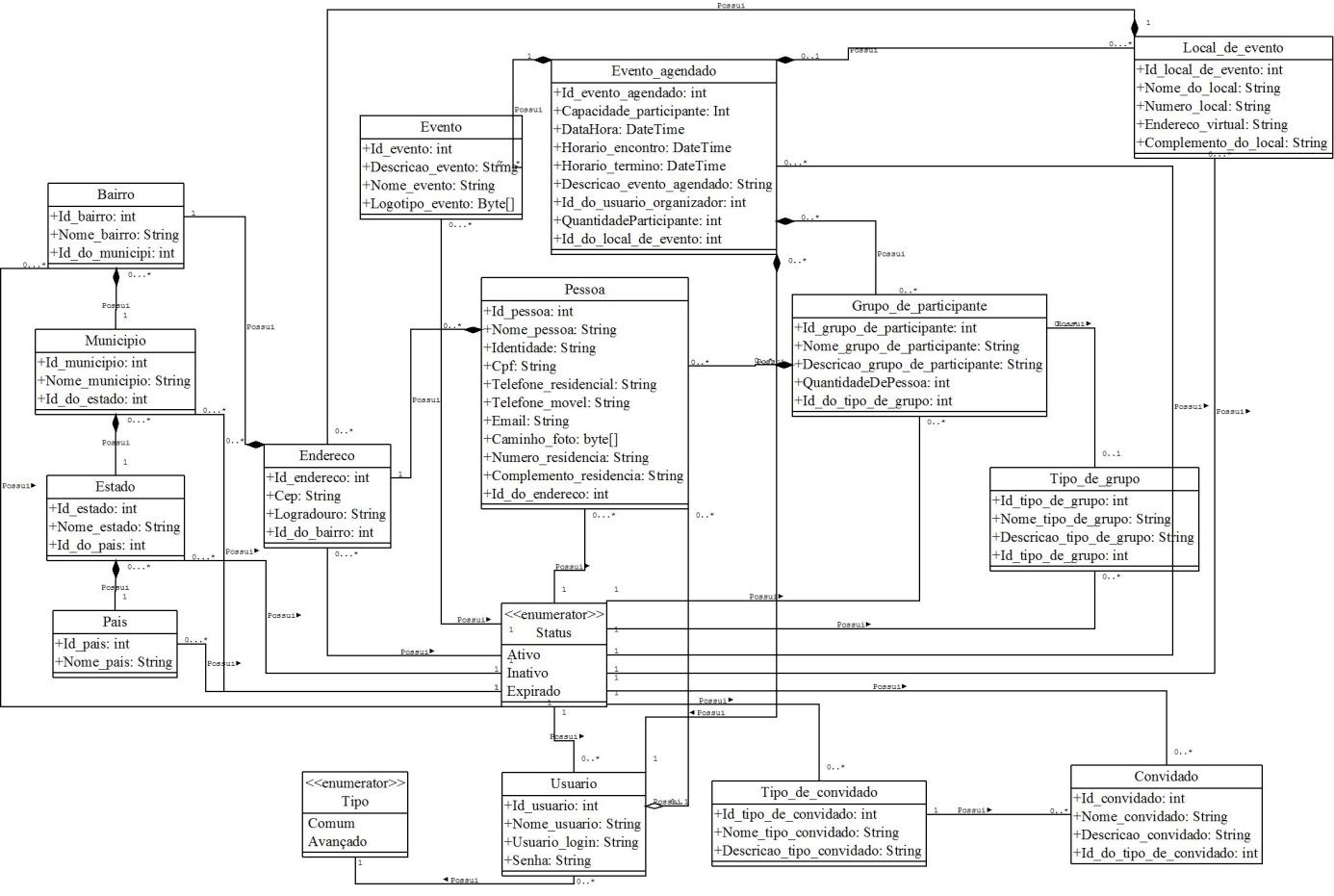


Figura 15 Diagrama das Classes do sistema.

## **Modelos de dados**

## Modelo Lógico da base de dados

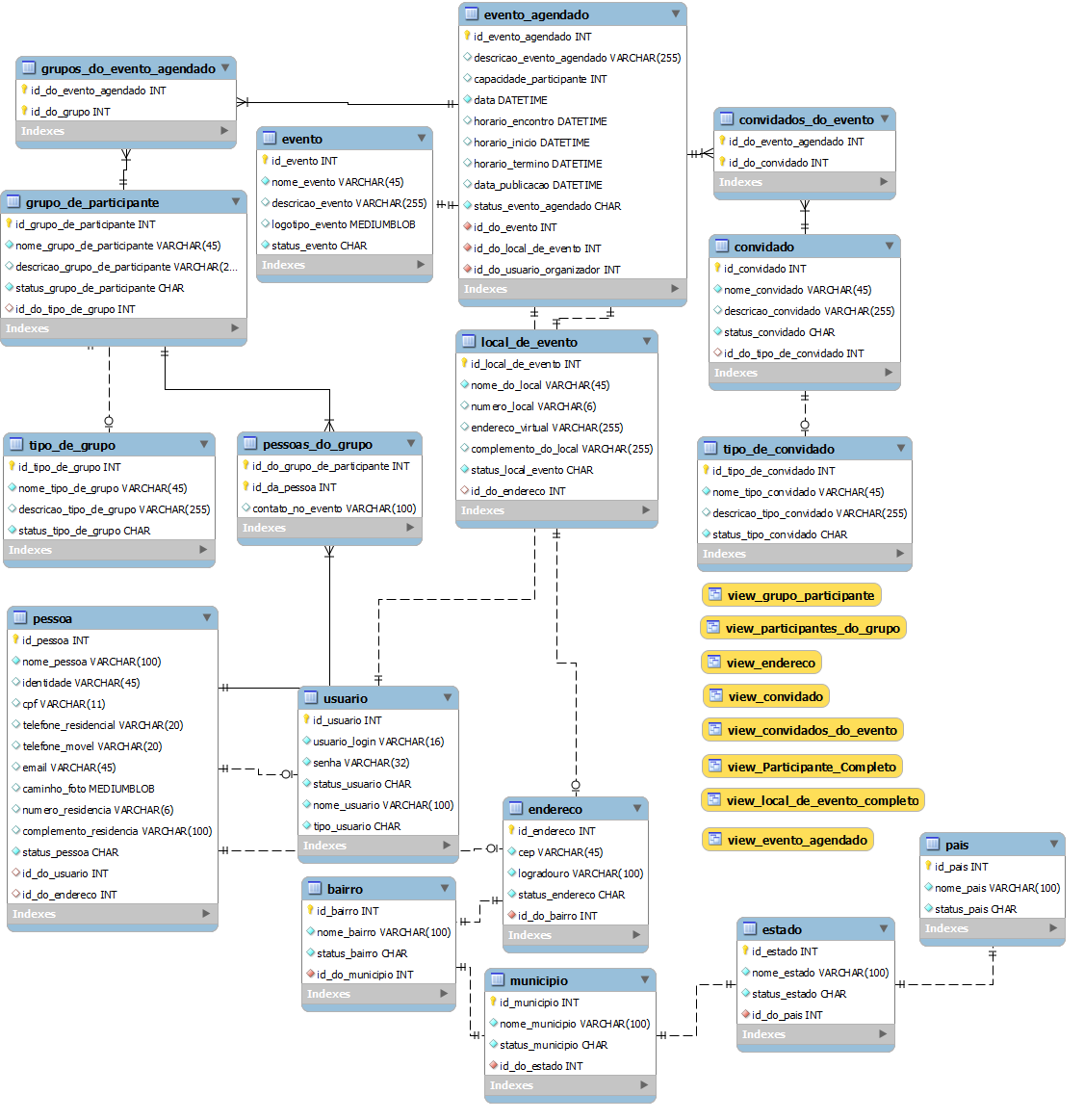


Figura 16 Modelo lógico da base de dados do sistema.

## Modelo Conceitual da base de dados

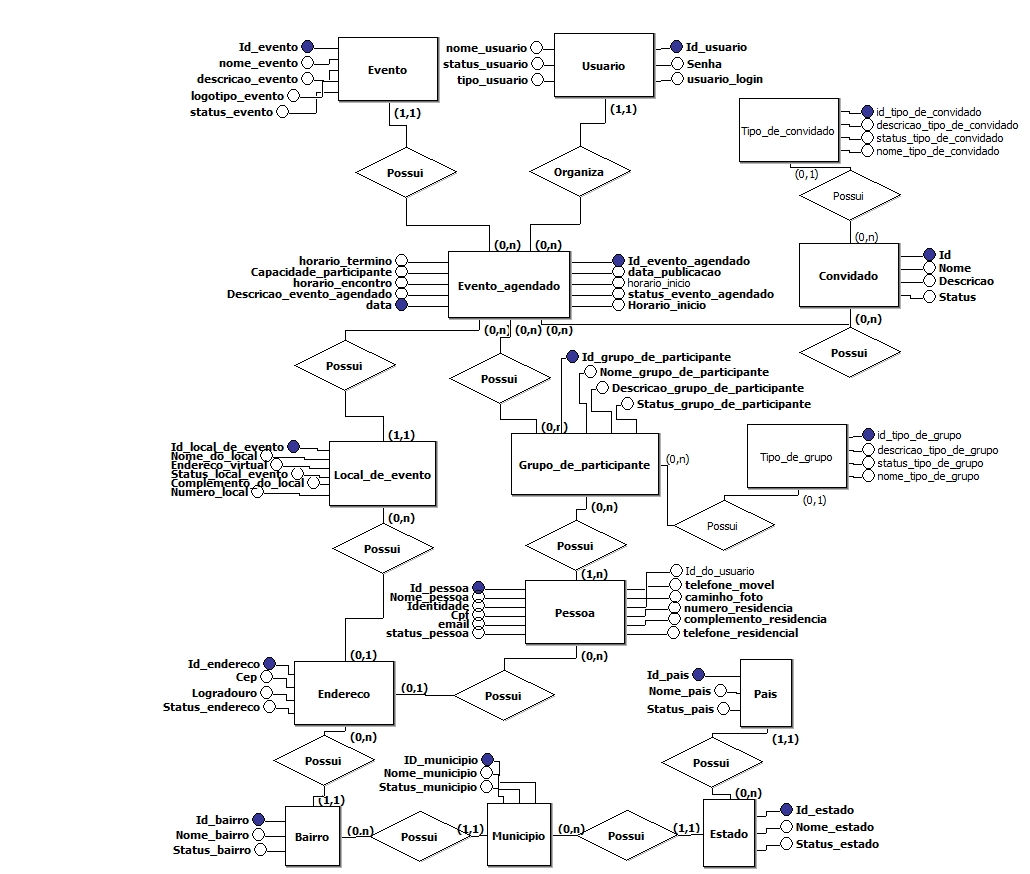


Figura 17 Modelo Conceitual da base de dados

## Modelo Físico da base de dados

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

CREATE SCHEMA IF NOT EXISTS `Event\_Viewer` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci ;

USE `Event\_Viewer` ;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`evento`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`evento` (

`id\_evento` INT NOT NULL AUTO\_INCREMENT,

`nome\_evento` VARCHAR(45) NOT NULL,

`descricao\_evento` VARCHAR(255) NULL,

`logotipo\_evento` MEDIUMBLOB NULL,

`status\_evento` CHAR NOT NULL DEFAULT '1',

PRIMARY KEY (`id\_evento`),

UNIQUE INDEX `nome\_UNIQUE` (`nome\_evento` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`pais`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`pais` (

`id\_pais` INT NOT NULL AUTO\_INCREMENT,

`nome\_pais` VARCHAR(100) NOT NULL,

`status\_pais` CHAR NOT NULL DEFAULT '1',

PRIMARY KEY (`id\_pais`),

UNIQUE INDEX `nome\_pais\_UNIQUE` (`nome\_pais` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`estado`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`estado` (

`id\_estado` INT NOT NULL AUTO\_INCREMENT,

`nome\_estado` VARCHAR(100) NOT NULL,

`status\_estado` CHAR NOT NULL DEFAULT '1',

`id\_do\_pais` INT NOT NULL,

PRIMARY KEY (`id\_estado`),

INDEX `fk\_estado\_pais1\_idx` (`id\_do\_pais` ASC),

UNIQUE INDEX `nome\_estado\_UNIQUE` (`nome\_estado` ASC),

CONSTRAINT `fk\_estado\_pais1`

FOREIGN KEY (`id\_do\_pais`)

REFERENCES `Event\_Viewer`.`pais` (`id\_pais`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`municipio`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`municipio` (

`id\_municipio` INT NOT NULL AUTO\_INCREMENT,

`nome\_municipio` VARCHAR(100) NOT NULL,

`status\_municipio` CHAR NOT NULL DEFAULT '1',

`id\_do\_estado` INT NOT NULL,

PRIMARY KEY (`id\_municipio`),

INDEX `fk\_municipio\_estado1\_idx` (`id\_do\_estado` ASC),

CONSTRAINT `fk\_municipio\_estado1`

FOREIGN KEY (`id\_do\_estado`)

REFERENCES `Event\_Viewer`.`estado` (`id\_estado`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`bairro`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`bairro` (

`id\_bairro` INT NOT NULL AUTO\_INCREMENT,

`nome\_bairro` VARCHAR(100) NOT NULL,

`status\_bairro` CHAR NOT NULL DEFAULT '1',

`id\_do\_municipio` INT NOT NULL,

PRIMARY KEY (`id\_bairro`),

INDEX `fk\_bairro\_municipio1\_idx` (`id\_do\_municipio` ASC),

CONSTRAINT `fk\_bairro\_municipio1`

FOREIGN KEY (`id\_do\_municipio`)

REFERENCES `Event\_Viewer`.`municipio` (`id\_municipio`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`endereco`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`endereco` (

`id\_endereco` INT NOT NULL AUTO\_INCREMENT,

`cep` VARCHAR(45) NOT NULL,

`logradouro` VARCHAR(100) NOT NULL,

`status\_endereco` CHAR NOT NULL DEFAULT '1',

`id\_do\_bairro` INT NOT NULL,

PRIMARY KEY (`id\_endereco`),

UNIQUE INDEX `cep\_UNIQUE` (`cep` ASC),

INDEX `fk\_endereco\_bairro1\_idx` (`id\_do\_bairro` ASC),

CONSTRAINT `fk\_endereco\_bairro1`

FOREIGN KEY (`id\_do\_bairro`)

REFERENCES `Event\_Viewer`.`bairro` (`id\_bairro`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`local\_de\_evento`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`local\_de\_evento` (

`id\_local\_de\_evento` INT NOT NULL AUTO\_INCREMENT,

`nome\_do\_local` VARCHAR(45) NOT NULL,

`numero\_local` VARCHAR(6) NULL,

`endereco\_virtual` VARCHAR(255) NULL,

`complemento\_do\_local` VARCHAR(255) NULL,

`status\_local\_evento` CHAR NOT NULL DEFAULT '1',

`id\_do\_endereco` INT NULL,

PRIMARY KEY (`id\_local\_de\_evento`),

INDEX `fk\_local\_de\_evento\_endereco1\_idx` (`id\_do\_endereco` ASC),

CONSTRAINT `fk\_local\_de\_evento\_endereco1`

FOREIGN KEY (`id\_do\_endereco`)

REFERENCES `Event\_Viewer`.`endereco` (`id\_endereco`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`usuario`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`usuario` (

`id\_usuario` INT NOT NULL AUTO\_INCREMENT,

`usuario\_login` VARCHAR(16) NOT NULL,

`senha` VARCHAR(32) NOT NULL,

`status\_usuario` CHAR NOT NULL,

`nome\_usuario` VARCHAR(100) NOT NULL,

`tipo\_usuario` CHAR NOT NULL,

PRIMARY KEY (`id\_usuario`),

UNIQUE INDEX `usuario\_UNIQUE` (`usuario\_login` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`evento\_agendado`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`evento\_agendado` (

`id\_evento\_agendado` INT NOT NULL AUTO\_INCREMENT,

`descricao\_evento\_agendado` VARCHAR(255) NULL,

`capacidade\_participante` INT NULL,

`data` DATETIME NOT NULL,

`horario\_encontro` DATETIME NULL,

`horario\_inicio` DATETIME NULL,

`horario\_termino` DATETIME NULL,

`data\_publicacao` DATETIME NULL,

`status\_evento\_agendado` CHAR NOT NULL DEFAULT '1',

`id\_do\_evento` INT NOT NULL,

`id\_do\_local\_de\_evento` INT NOT NULL,

`id\_do\_usuario\_organizador` INT NOT NULL,

PRIMARY KEY (`id\_evento\_agendado`),

INDEX `fk\_evento\_agendado\_evento1\_idx` (`id\_do\_evento` ASC),

INDEX `fk\_evento\_agendado\_local\_de\_evento1\_idx` (`id\_do\_local\_de\_evento` ASC),

INDEX `fk\_evento\_agendado\_usuario1\_idx` (`id\_do\_usuario\_organizador` ASC),

CONSTRAINT `fk\_evento\_agendado\_evento1`

FOREIGN KEY (`id\_do\_evento`)

REFERENCES `Event\_Viewer`.`evento` (`id\_evento`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_evento\_agendado\_local\_de\_evento1`

FOREIGN KEY (`id\_do\_local\_de\_evento`)

REFERENCES `Event\_Viewer`.`local\_de\_evento` (`id\_local\_de\_evento`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_evento\_agendado\_usuario1`

FOREIGN KEY (`id\_do\_usuario\_organizador`)

REFERENCES `Event\_Viewer`.`usuario` (`id\_usuario`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`pessoa`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`pessoa` (

`id\_pessoa` INT NOT NULL AUTO\_INCREMENT,

`nome\_pessoa` VARCHAR(100) NOT NULL,

`identidade` VARCHAR(45) NULL,

`cpf` VARCHAR(11) NULL,

`telefone\_residencial` VARCHAR(20) NULL,

`telefone\_movel` VARCHAR(20) NULL,

`email` VARCHAR(45) NULL,

`caminho\_foto` MEDIUMBLOB NULL,

`numero\_residencia` VARCHAR(6) NULL,

`complemento\_residencia` VARCHAR(100) NULL,

`status\_pessoa` CHAR NOT NULL DEFAULT '1',

`id\_do\_usuario` INT NULL,

`id\_do\_endereco` INT NULL,

PRIMARY KEY (`id\_pessoa`),

INDEX `fk\_pessoa\_usuario1\_idx` (`id\_do\_usuario` ASC),

INDEX `fk\_pessoa\_endereco1\_idx` (`id\_do\_endereco` ASC),

CONSTRAINT `fk\_pessoa\_usuario1`

FOREIGN KEY (`id\_do\_usuario`)

REFERENCES `Event\_Viewer`.`usuario` (`id\_usuario`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_pessoa\_endereco1`

FOREIGN KEY (`id\_do\_endereco`)

REFERENCES `Event\_Viewer`.`endereco` (`id\_endereco`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`tipo\_de\_grupo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`tipo\_de\_grupo` (

`id\_tipo\_de\_grupo` INT NOT NULL AUTO\_INCREMENT,

`nome\_tipo\_de\_grupo` VARCHAR(45) NOT NULL,

`descricao\_tipo\_de\_grupo` VARCHAR(255) NULL,

`status\_tipo\_de\_grupo` CHAR NOT NULL DEFAULT '1',

PRIMARY KEY (`id\_tipo\_de\_grupo`),

UNIQUE INDEX `nome\_UNIQUE` (`nome\_tipo\_de\_grupo` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`grupo\_de\_participante`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`grupo\_de\_participante` (

`id\_grupo\_de\_participante` INT NOT NULL AUTO\_INCREMENT,

`nome\_grupo\_de\_participante` VARCHAR(45) NOT NULL,

`descricao\_grupo\_de\_participante` VARCHAR(255) NULL,

`status\_grupo\_de\_participante` CHAR NOT NULL DEFAULT '1',

`id\_do\_tipo\_de\_grupo` INT NULL,

PRIMARY KEY (`id\_grupo\_de\_participante`),

INDEX `fk\_grupo\_de\_participante\_tipo\_de\_grupo1\_idx` (`id\_do\_tipo\_de\_grupo` ASC),

CONSTRAINT `fk\_grupo\_de\_participante\_tipo\_de\_grupo1`

FOREIGN KEY (`id\_do\_tipo\_de\_grupo`)

REFERENCES `Event\_Viewer`.`tipo\_de\_grupo` (`id\_tipo\_de\_grupo`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`tipo\_de\_convidado`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`tipo\_de\_convidado` (

`id\_tipo\_de\_convidado` INT NOT NULL AUTO\_INCREMENT,

`nome\_tipo\_convidado` VARCHAR(45) NOT NULL,

`descricao\_tipo\_convidado` VARCHAR(255) NULL,

`status\_tipo\_convidado` CHAR NOT NULL DEFAULT '1',

PRIMARY KEY (`id\_tipo\_de\_convidado`),

UNIQUE INDEX `nome\_UNIQUE` (`nome\_tipo\_convidado` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`convidado`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`convidado` (

`id\_convidado` INT NOT NULL AUTO\_INCREMENT,

`nome\_convidado` VARCHAR(45) NOT NULL,

`descricao\_convidado` VARCHAR(255) NULL,

`status\_convidado` CHAR NOT NULL DEFAULT '1',

`id\_do\_tipo\_de\_convidado` INT NULL,

PRIMARY KEY (`id\_convidado`),

INDEX `fk\_convidado\_tipo\_de\_convidado1\_idx` (`id\_do\_tipo\_de\_convidado` ASC),

CONSTRAINT `fk\_convidado\_tipo\_de\_convidado1`

FOREIGN KEY (`id\_do\_tipo\_de\_convidado`)

REFERENCES `Event\_Viewer`.`tipo\_de\_convidado` (`id\_tipo\_de\_convidado`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`grupos\_do\_evento\_agendado`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`grupos\_do\_evento\_agendado` (

`id\_do\_evento\_agendado` INT NOT NULL,

`id\_do\_grupo` INT NOT NULL,

PRIMARY KEY (`id\_do\_evento\_agendado`, `id\_do\_grupo`),

INDEX `fk\_evento\_agendado\_has\_grupo\_de\_participante\_grupo\_de\_parti\_idx` (`id\_do\_grupo` ASC),

INDEX `fk\_evento\_agendado\_has\_grupo\_de\_participante\_evento\_agendad\_idx` (`id\_do\_evento\_agendado` ASC),

CONSTRAINT `fk\_evento\_agendado\_has\_grupo\_de\_participante\_evento\_agendado`

FOREIGN KEY (`id\_do\_evento\_agendado`)

REFERENCES `Event\_Viewer`.`evento\_agendado` (`id\_evento\_agendado`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_evento\_agendado\_has\_grupo\_de\_participante\_grupo\_de\_partici1`

FOREIGN KEY (`id\_do\_grupo`)

REFERENCES `Event\_Viewer`.`grupo\_de\_participante` (`id\_grupo\_de\_participante`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`convidados\_do\_evento`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`convidados\_do\_evento` (

`id\_do\_evento\_agendado` INT NOT NULL,

`id\_do\_convidado` INT NOT NULL,

PRIMARY KEY (`id\_do\_evento\_agendado`, `id\_do\_convidado`),

INDEX `fk\_evento\_agendado\_has\_convidado\_convidado1\_idx` (`id\_do\_convidado` ASC),

INDEX `fk\_evento\_agendado\_has\_convidado\_evento\_agendado1\_idx` (`id\_do\_evento\_agendado` ASC),

CONSTRAINT `fk\_evento\_agendado\_has\_convidado\_evento\_agendado1`

FOREIGN KEY (`id\_do\_evento\_agendado`)

REFERENCES `Event\_Viewer`.`evento\_agendado` (`id\_evento\_agendado`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_evento\_agendado\_has\_convidado\_convidado1`

FOREIGN KEY (`id\_do\_convidado`)

REFERENCES `Event\_Viewer`.`convidado` (`id\_convidado`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Event\_Viewer`.`pessoas\_do\_grupo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`pessoas\_do\_grupo` (

`id\_do\_grupo\_de\_participante` INT NOT NULL,

`id\_da\_pessoa` INT NOT NULL,

`contato\_no\_evento` VARCHAR(100) NULL,

PRIMARY KEY (`id\_do\_grupo\_de\_participante`, `id\_da\_pessoa`),

INDEX `fk\_grupo\_de\_participante\_has\_pessoa\_pessoa1\_idx` (`id\_da\_pessoa` ASC),

INDEX `fk\_grupo\_de\_participante\_has\_pessoa\_grupo\_de\_participante1\_idx` (`id\_do\_grupo\_de\_participante` ASC),

CONSTRAINT `fk\_grupo\_de\_participante\_has\_pessoa\_grupo\_de\_participante1`

FOREIGN KEY (`id\_do\_grupo\_de\_participante`)

REFERENCES `Event\_Viewer`.`grupo\_de\_participante` (`id\_grupo\_de\_participante`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_grupo\_de\_participante\_has\_pessoa\_pessoa1`

FOREIGN KEY (`id\_da\_pessoa`)

REFERENCES `Event\_Viewer`.`pessoa` (`id\_pessoa`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

USE `Event\_Viewer` ;

-- -----------------------------------------------------

-- Placeholder table for view `Event\_Viewer`.`view\_endereco`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`view\_endereco` (`id\_endereco` INT, `cep` INT, `logradouro` INT, `status\_endereco` INT, `id\_do\_bairro` INT, `id\_bairro` INT, `nome\_bairro` INT, `status\_bairro` INT, `id\_do\_municipio` INT, `id\_municipio` INT, `nome\_municipio` INT, `status\_municipio` INT, `id\_do\_estado` INT, `id\_estado` INT, `nome\_estado` INT, `status\_estado` INT, `id\_do\_pais` INT, `id\_pais` INT, `nome\_pais` INT, `status\_pais` INT);

-- -----------------------------------------------------

-- Placeholder table for view `Event\_Viewer`.`view\_grupo\_participante`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`view\_grupo\_participante` (`id\_grupo\_de\_participante` INT, `nome\_grupo\_de\_participante` INT, `descricao\_grupo\_de\_participante` INT, `status\_grupo\_de\_participante` INT, `id\_do\_tipo\_de\_grupo` INT, `id\_tipo\_de\_grupo` INT, `nome\_tipo\_de\_grupo` INT, `descricao\_tipo\_de\_grupo` INT, `status\_tipo\_de\_grupo` INT);

-- -----------------------------------------------------

-- Placeholder table for view `Event\_Viewer`.`view\_participantes\_do\_grupo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`view\_participantes\_do\_grupo` (`id\_do\_grupo\_de\_participante` INT, `id\_da\_pessoa` INT, `contato\_no\_evento` INT, `id\_pessoa` INT, `nome\_pessoa` INT, `identidade` INT, `cpf` INT, `telefone\_residencial` INT, `telefone\_movel` INT, `email` INT, `caminho\_foto` INT, `numero\_residencia` INT, `complemento\_residencia` INT, `status\_pessoa` INT, `id\_do\_usuario` INT, `id\_do\_endereco` INT);

-- -----------------------------------------------------

-- Placeholder table for view `Event\_Viewer`.`view\_convidado`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`view\_convidado` (`id\_convidado` INT, `nome\_convidado` INT, `descricao\_convidado` INT, `status\_convidado` INT, `id\_do\_tipo\_de\_convidado` INT, `id\_tipo\_de\_convidado` INT, `nome\_tipo\_convidado` INT, `descricao\_tipo\_convidado` INT, `status\_tipo\_convidado` INT);

-- -----------------------------------------------------

-- Placeholder table for view `Event\_Viewer`.`view\_convidados\_do\_evento`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`view\_convidados\_do\_evento` (`id\_do\_evento\_agendado` INT, `id\_do\_convidado` INT);

-- -----------------------------------------------------

-- Placeholder table for view `Event\_Viewer`.`view\_Participante\_Completo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`view\_Participante\_Completo` (`id\_pessoa` INT, `nome\_pessoa` INT, `identidade` INT, `cpf` INT, `telefone\_residencial` INT, `telefone\_movel` INT, `email` INT, `caminho\_foto` INT, `numero\_residencia` INT, `complemento\_residencia` INT, `status\_pessoa` INT, `id\_do\_usuario` INT, `id\_do\_endereco` INT, `id\_endereco` INT, `cep` INT, `logradouro` INT, `status\_endereco` INT, `id\_do\_bairro` INT, `id\_bairro` INT, `nome\_bairro` INT, `status\_bairro` INT, `id\_do\_municipio` INT, `id\_municipio` INT, `nome\_municipio` INT, `status\_municipio` INT, `id\_do\_estado` INT, `id\_estado` INT, `nome\_estado` INT, `status\_estado` INT, `id\_do\_pais` INT, `id\_pais` INT, `nome\_pais` INT, `status\_pais` INT);

-- -----------------------------------------------------

-- Placeholder table for view `Event\_Viewer`.`view\_local\_de\_evento\_completo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`view\_local\_de\_evento\_completo` (`id\_local\_de\_evento` INT, `nome\_do\_local` INT, `numero\_local` INT, `endereco\_virtual` INT, `complemento\_do\_local` INT, `status\_local\_evento` INT, `id\_do\_endereco` INT, `id\_endereco` INT, `cep` INT, `logradouro` INT, `status\_endereco` INT, `id\_do\_bairro` INT, `id\_bairro` INT, `nome\_bairro` INT, `status\_bairro` INT, `id\_do\_municipio` INT, `id\_municipio` INT, `nome\_municipio` INT, `status\_municipio` INT, `id\_do\_estado` INT, `id\_estado` INT, `nome\_estado` INT, `status\_estado` INT, `id\_do\_pais` INT, `id\_pais` INT, `nome\_pais` INT, `status\_pais` INT);

-- -----------------------------------------------------

-- Placeholder table for view `Event\_Viewer`.`view\_evento\_agendado`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Event\_Viewer`.`view\_evento\_agendado` (`id\_evento\_agendado` INT, `descricao\_evento\_agendado` INT, `capacidade\_participante` INT, `data` INT, `horario\_encontro` INT, `horario\_inicio` INT, `horario\_termino` INT, `data\_publicacao` INT, `status\_evento\_agendado` INT, `id\_do\_evento` INT, `id\_do\_local\_de\_evento` INT, `id\_do\_usuario\_organizador` INT, `id\_evento` INT, `nome\_evento` INT, `descricao\_evento` INT, `logotipo\_evento` INT, `status\_evento` INT, `id\_local\_de\_evento` INT, `nome\_do\_local` INT, `numero\_local` INT, `endereco\_virtual` INT, `complemento\_do\_local` INT, `status\_local\_evento` INT, `id\_do\_endereco` INT, `id\_endereco` INT, `cep` INT, `logradouro` INT, `status\_endereco` INT, `id\_do\_bairro` INT, `id\_bairro` INT, `nome\_bairro` INT, `status\_bairro` INT, `id\_do\_municipio` INT, `id\_municipio` INT, `nome\_municipio` INT, `status\_municipio` INT, `id\_do\_estado` INT, `id\_estado` INT, `nome\_estado` INT, `status\_estado` INT, `id\_do\_pais` INT, `id\_pais` INT, `nome\_pais` INT, `status\_pais` INT, `id\_usuario` INT, `usuario\_login` INT, `senha` INT, `status\_usuario` INT, `nome\_usuario` INT, `tipo\_usuario` INT, `id\_do\_evento\_agendado` INT, `id\_do\_grupo` INT, `id\_grupo\_de\_participante` INT, `nome\_grupo\_de\_participante` INT, `descricao\_grupo\_de\_participante` INT, `status\_grupo\_de\_participante` INT, `id\_do\_tipo\_de\_grupo` INT, `id\_convidado` INT, `nome\_convidado` INT, `descricao\_convidado` INT, `status\_convidado` INT, `id\_do\_tipo\_de\_convidado` INT);

-- -----------------------------------------------------

-- View `Event\_Viewer`.`view\_endereco`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Event\_Viewer`.`view\_endereco`;

USE `Event\_Viewer`;

CREATE OR REPLACE VIEW `Event\_Viewer`.`view\_endereco` AS

select

\*

from

endereco

join

bairro ON endereco.id\_do\_bairro = bairro.id\_bairro

join

municipio on bairro.id\_do\_municipio = municipio.id\_municipio

join

estado on municipio.id\_do\_estado = estado.id\_estado

join

pais on estado.id\_do\_pais = pais.id\_pais

;

-- -----------------------------------------------------

-- View `Event\_Viewer`.`view\_grupo\_participante`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Event\_Viewer`.`view\_grupo\_participante`;

USE `Event\_Viewer`;

CREATE OR REPLACE VIEW `Event\_Viewer`.`view\_grupo\_participante` AS

select \* from grupo\_de\_participante gp

left join

tipo\_de\_grupo tg

on

gp.id\_do\_tipo\_de\_grupo = tg.id\_tipo\_de\_grupo

;

-- -----------------------------------------------------

-- View `Event\_Viewer`.`view\_participantes\_do\_grupo`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Event\_Viewer`.`view\_participantes\_do\_grupo`;

USE `Event\_Viewer`;

CREATE OR REPLACE VIEW `Event\_Viewer`.`view\_participantes\_do\_grupo` AS

select \* from pessoas\_do\_grupo pg

left join

pessoa p

on

pg.id\_da\_pessoa = p.id\_pessoa;

-- -----------------------------------------------------

-- View `Event\_Viewer`.`view\_convidado`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Event\_Viewer`.`view\_convidado`;

USE `Event\_Viewer`;

CREATE OR REPLACE VIEW `Event\_Viewer`.`view\_convidado` AS

select

\*

from

convidado c

left join

tipo\_de\_convidado tc ON c.id\_do\_tipo\_de\_convidado = tc.id\_tipo\_de\_convidado;

-- -----------------------------------------------------

-- View `Event\_Viewer`.`view\_convidados\_do\_evento`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Event\_Viewer`.`view\_convidados\_do\_evento`;

USE `Event\_Viewer`;

CREATE OR REPLACE VIEW `Event\_Viewer`.`view\_convidados\_do\_evento` AS

select \* from convidados\_do\_evento ce

join

view\_convidado vc

on

ce.id\_do\_convidado = vc.id\_convidado;

-- -----------------------------------------------------

-- View `Event\_Viewer`.`view\_Participante\_Completo`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Event\_Viewer`.`view\_Participante\_Completo`;

USE `Event\_Viewer`;

CREATE OR REPLACE VIEW `Event\_Viewer`.`view\_Participante\_Completo` AS

select

\*

from

pessoa

left join

endereco ON pessoa.id\_do\_endereco = endereco.id\_endereco

left join

bairro ON endereco.id\_do\_bairro = bairro.id\_bairro

left join

municipio on bairro.id\_do\_municipio = municipio.id\_municipio

left join

estado on municipio.id\_do\_estado = estado.id\_estado

left join

pais on estado.id\_do\_pais = pais.id\_pais

;

-- -----------------------------------------------------

-- View `Event\_Viewer`.`view\_local\_de\_evento\_completo`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Event\_Viewer`.`view\_local\_de\_evento\_completo`;

USE `Event\_Viewer`;

CREATE OR REPLACE VIEW `view\_local\_de\_evento\_completo` AS

select

\*

from

local\_de\_evento

left join

endereco ON local\_de\_evento.id\_do\_endereco = endereco.id\_endereco

left join

bairro ON endereco.id\_do\_bairro = bairro.id\_bairro

left join

municipio on bairro.id\_do\_municipio = municipio.id\_municipio

left join

estado on municipio.id\_do\_estado = estado.id\_estado

left join

pais on estado.id\_do\_pais = pais.id\_pais

;

-- -----------------------------------------------------

-- View `Event\_Viewer`.`view\_evento\_agendado`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Event\_Viewer`.`view\_evento\_agendado`;

USE `Event\_Viewer`;

CREATE OR REPLACE VIEW `Event\_Viewer`.`view\_evento\_agendado` AS

SELECT

`evento\_agendado`.`id\_evento\_agendado` AS `id\_evento\_agendado`,

`evento\_agendado`.`descricao\_evento\_agendado` AS `descricao\_evento\_agendado`,

`evento\_agendado`.`capacidade\_participante` AS `capacidade\_participante`,

`evento\_agendado`.`data` AS `data`,

`evento\_agendado`.`horario\_encontro` AS `horario\_encontro`,

`evento\_agendado`.`horario\_inicio` AS `horario\_inicio`,

`evento\_agendado`.`horario\_termino` AS `horario\_termino`,

`evento\_agendado`.`data\_publicacao` AS `data\_publicacao`,

`evento\_agendado`.`status\_evento\_agendado` AS `status\_evento\_agendado`,

`evento\_agendado`.`id\_do\_evento` AS `id\_do\_evento`,

`evento\_agendado`.`id\_do\_local\_de\_evento` AS `id\_do\_local\_de\_evento`,

`evento\_agendado`.`id\_do\_usuario\_organizador` AS `id\_do\_usuario\_organizador`,

`evento`.`id\_evento` AS `id\_evento`,

`evento`.`nome\_evento` AS `nome\_evento`,

`evento`.`descricao\_evento` AS `descricao\_evento`,

`evento`.`logotipo\_evento` AS `logotipo\_evento`,

`evento`.`status\_evento` AS `status\_evento`,

`local\_de\_evento`.`id\_local\_de\_evento` AS `id\_local\_de\_evento`,

`local\_de\_evento`.`nome\_do\_local` AS `nome\_do\_local`,

`local\_de\_evento`.`numero\_local` AS `numero\_local`,

`local\_de\_evento`.`endereco\_virtual` AS `endereco\_virtual`,

`local\_de\_evento`.`complemento\_do\_local` AS `complemento\_do\_local`,

`local\_de\_evento`.`status\_local\_evento` AS `status\_local\_evento`,

`local\_de\_evento`.`id\_do\_endereco` AS `id\_do\_endereco`,

`endereco`.`id\_endereco` AS `id\_endereco`,

`endereco`.`cep` AS `cep`,

`endereco`.`logradouro` AS `logradouro`,

`endereco`.`status\_endereco` AS `status\_endereco`,

`endereco`.`id\_do\_bairro` AS `id\_do\_bairro`,

`bairro`.`id\_bairro` AS `id\_bairro`,

`bairro`.`nome\_bairro` AS `nome\_bairro`,

`bairro`.`status\_bairro` AS `status\_bairro`,

`bairro`.`id\_do\_municipio` AS `id\_do\_municipio`,

`municipio`.`id\_municipio` AS `id\_municipio`,

`municipio`.`nome\_municipio` AS `nome\_municipio`,

`municipio`.`status\_municipio` AS `status\_municipio`,

`municipio`.`id\_do\_estado` AS `id\_do\_estado`,

`estado`.`id\_estado` AS `id\_estado`,

`estado`.`nome\_estado` AS `nome\_estado`,

`estado`.`status\_estado` AS `status\_estado`,

`estado`.`id\_do\_pais` AS `id\_do\_pais`,

`pais`.`id\_pais` AS `id\_pais`,

`pais`.`nome\_pais` AS `nome\_pais`,

`pais`.`status\_pais` AS `status\_pais`,

`usuario`.`id\_usuario` AS `id\_usuario`,

`usuario`.`usuario\_login` AS `usuario\_login`,

`usuario`.`senha` AS `senha`,

`usuario`.`status\_usuario` AS `status\_usuario`,

`usuario`.`nome\_usuario` AS `nome\_usuario`,

`usuario`.`tipo\_usuario` AS `tipo\_usuario`,

`grupos\_do\_evento\_agendado`.`id\_do\_evento\_agendado` AS `id\_do\_evento\_agendado`,

`grupos\_do\_evento\_agendado`.`id\_do\_grupo` AS `id\_do\_grupo`,

`grupo\_de\_participante`.`id\_grupo\_de\_participante` AS `id\_grupo\_de\_participante`,

`grupo\_de\_participante`.`nome\_grupo\_de\_participante` AS `nome\_grupo\_de\_participante`,

`grupo\_de\_participante`.`descricao\_grupo\_de\_participante` AS `descricao\_grupo\_de\_participante`,

`grupo\_de\_participante`.`status\_grupo\_de\_participante` AS `status\_grupo\_de\_participante`,

`grupo\_de\_participante`.`id\_do\_tipo\_de\_grupo` AS `id\_do\_tipo\_de\_grupo`,

`convidado`.`id\_convidado` AS `id\_convidado`,

`convidado`.`nome\_convidado` AS `nome\_convidado`,

`convidado`.`descricao\_convidado` AS `descricao\_convidado`,

`convidado`.`status\_convidado` AS `status\_convidado`,

`convidado`.`id\_do\_tipo\_de\_convidado` AS `id\_do\_tipo\_de\_convidado`

FROM

((((((((((((`evento\_agendado`

JOIN `evento` ON ((`evento\_agendado`.`id\_do\_evento` = `evento`.`id\_evento`)))

JOIN `local\_de\_evento` ON ((`evento\_agendado`.`id\_do\_local\_de\_evento` = `local\_de\_evento`.`id\_local\_de\_evento`)))

LEFT JOIN `endereco` ON ((`local\_de\_evento`.`id\_do\_endereco` = `endereco`.`id\_endereco`)))

LEFT JOIN `bairro` ON ((`endereco`.`id\_do\_bairro` = `bairro`.`id\_bairro`)))

LEFT JOIN `municipio` ON ((`bairro`.`id\_do\_municipio` = `municipio`.`id\_municipio`)))

LEFT JOIN `estado` ON ((`municipio`.`id\_do\_estado` = `estado`.`id\_estado`)))

LEFT JOIN `pais` ON ((`estado`.`id\_do\_pais` = `pais`.`id\_pais`)))

LEFT JOIN `usuario` ON ((`evento\_agendado`.`id\_do\_usuario\_organizador` = `usuario`.`id\_usuario`)))

LEFT JOIN `grupos\_do\_evento\_agendado` ON ((`grupos\_do\_evento\_agendado`.`id\_do\_evento\_agendado` = `evento\_agendado`.`id\_evento\_agendado`)))

LEFT JOIN `grupo\_de\_participante` ON ((`grupos\_do\_evento\_agendado`.`id\_do\_grupo` = `grupo\_de\_participante`.`id\_grupo\_de\_participante`)))

LEFT JOIN `convidados\_do\_evento` ON ((`convidados\_do\_evento`.`id\_do\_evento\_agendado` = `evento\_agendado`.`id\_evento\_agendado`)))

LEFT JOIN `convidado` ON ((`convidados\_do\_evento`.`id\_do\_convidado` = `convidado`.`id\_convidado`)));

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

-- -----------------------------------------------------

-- Data for table `Event\_Viewer`.`usuario`

-- -----------------------------------------------------

START TRANSACTION;

USE `Event\_Viewer`;

INSERT INTO `Event\_Viewer`.`usuario` (`id\_usuario`, `usuario\_login`, `senha`, `status\_usuario`, `nome\_usuario`, `tipo\_usuario`) VALUES (1, 'ADMIN', 'ADMIN', '1', 'ADMIN', '1');

COMMIT;

## **Ambiente de desenvolvimento**

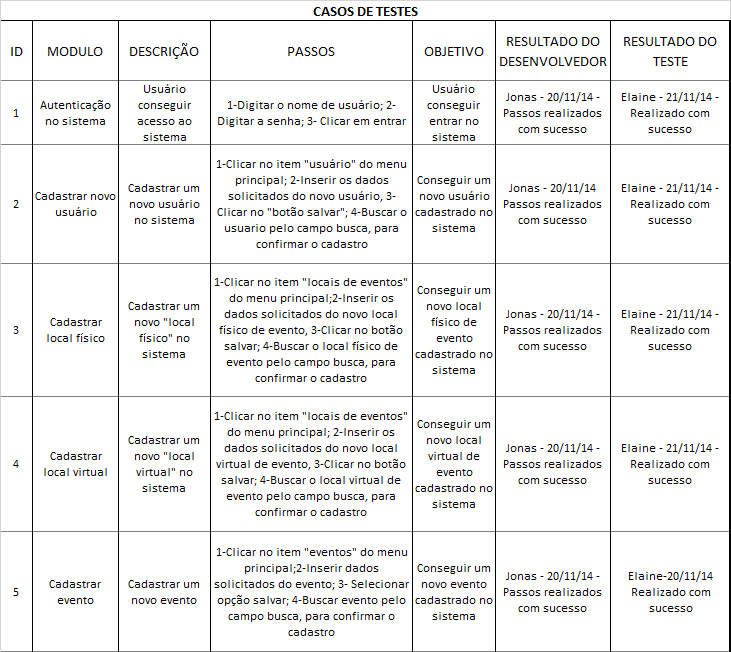
O sistema será desenvolvido utilizando a linguagem C# no Visual Studio 2010 utilizando a tecnologia de interface de usuário WPF (*Windows Foundation Presentation*).Durante o desenvolvimento do sistema foi utilizado também as seguintes ferramentas:

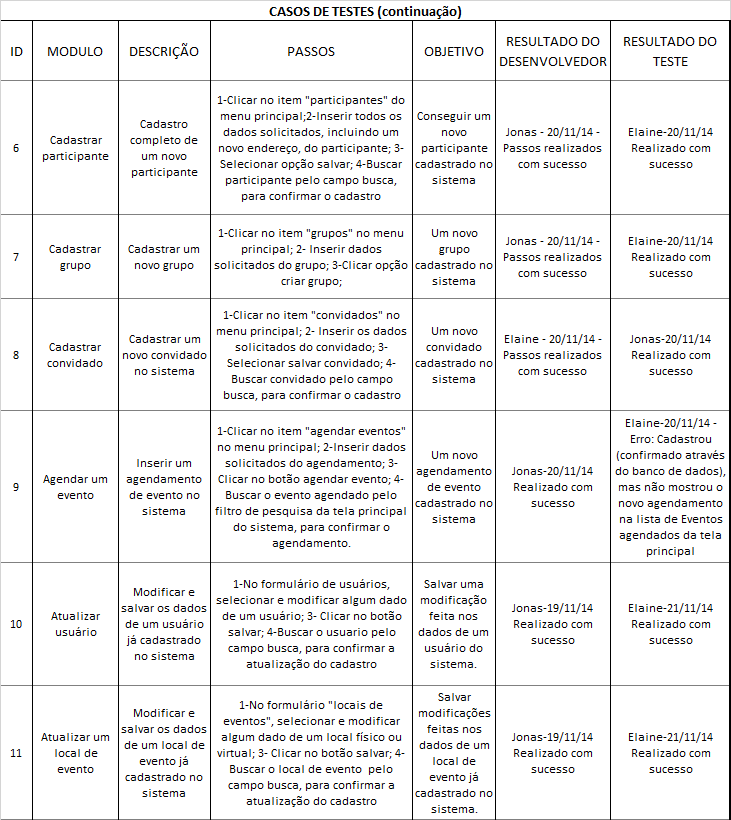
* Extended WPF Toolkit™ Community Edition;
* Violet;
* Dia Portable;
* MySql Workbench;
* Fireworks;
* Excel 2010;
* Word 2010;
* BrModelo 2.0.

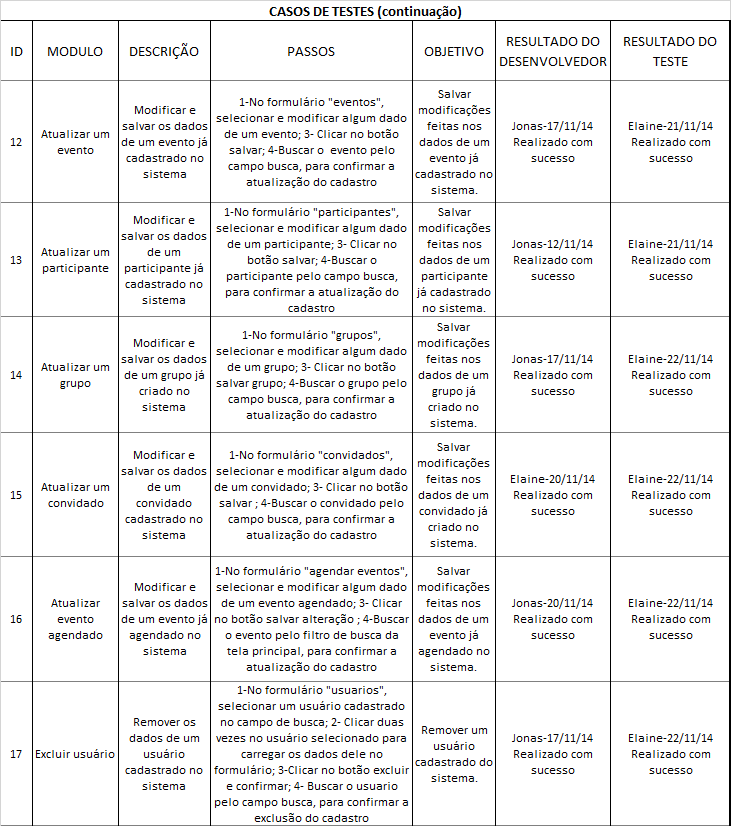
# **Testes**

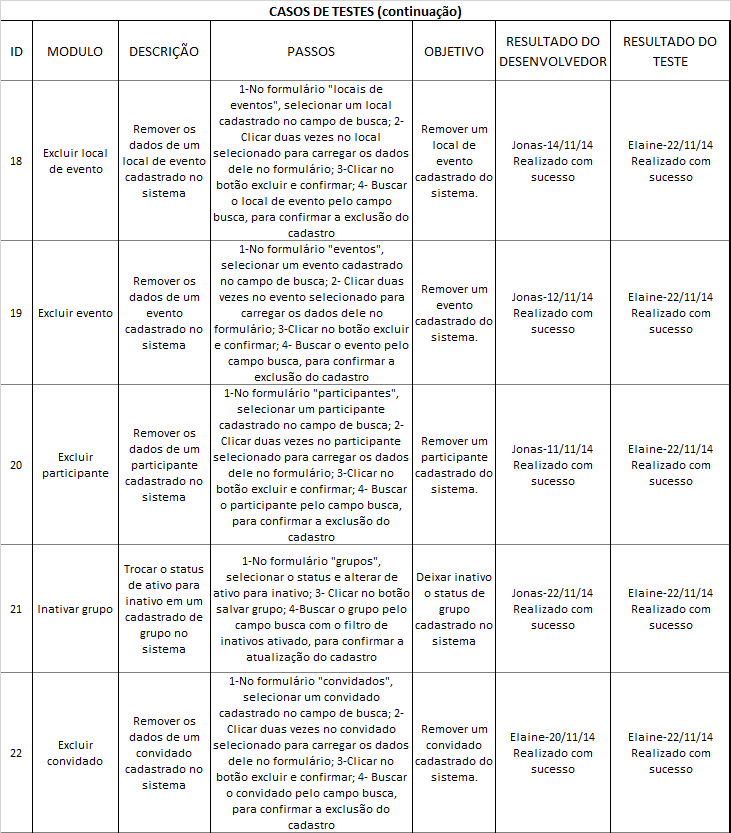
Apêndice A – Plano de testes.

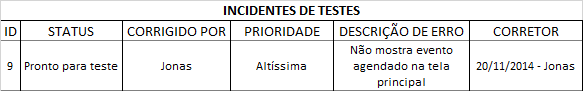
SE TIVER APENDICE NÂO DEVE TER NO DOCUMENTO











# **Protótipos**

No início do desenvolvimento desse projeto, foram criados alguns protótipos das interfaces com a finalidade de gerar, para o cliente, uma melhor compreensão do funcionamento do sistema.

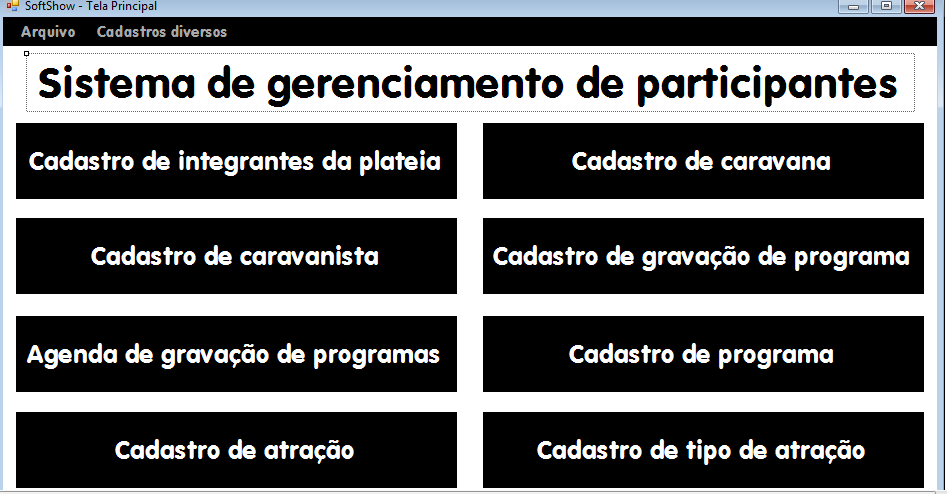


Figura 17 – 1º Protótipo da Interface Principal do Sistema



Figura 18 – 2º Protótipo da Interface Principal do Sistema

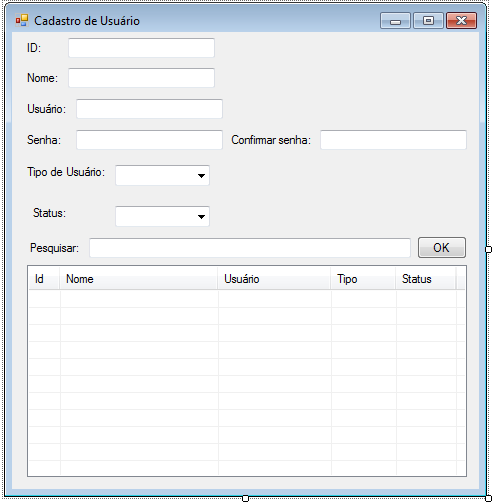


Figura 19 – Protótipo da Interface de Cadastro do Usuário

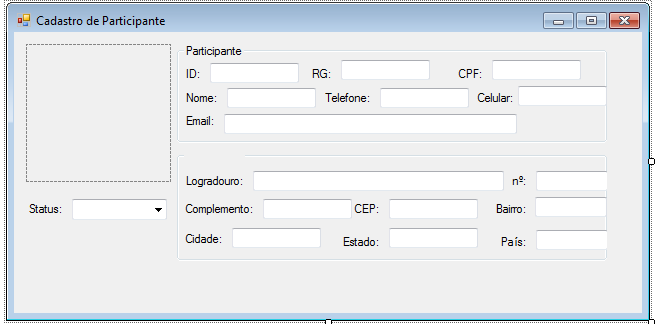


Figura 20 – Protótipo da Interface de Cadastro dos Participantes

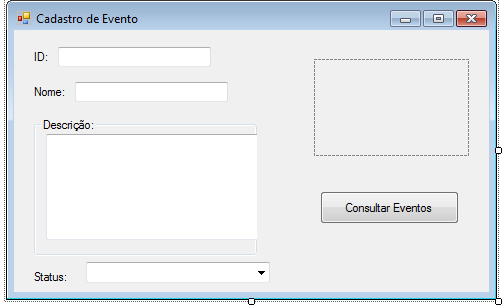


Figura 21 – Protótipo da Interface de Cadastro dos Eventos

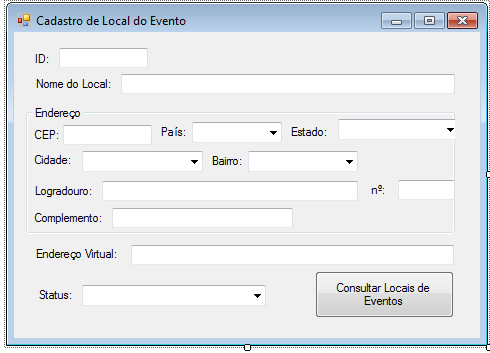


Figura 22 – Protótipo da Interface de Cadastro dos Locais dos Eventos

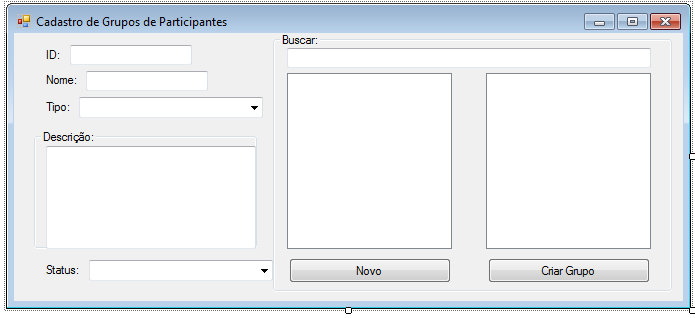


Figura 23 – Protótipo da Interface de Cadastro dos Grupos de Participantes

# **Produto Final**

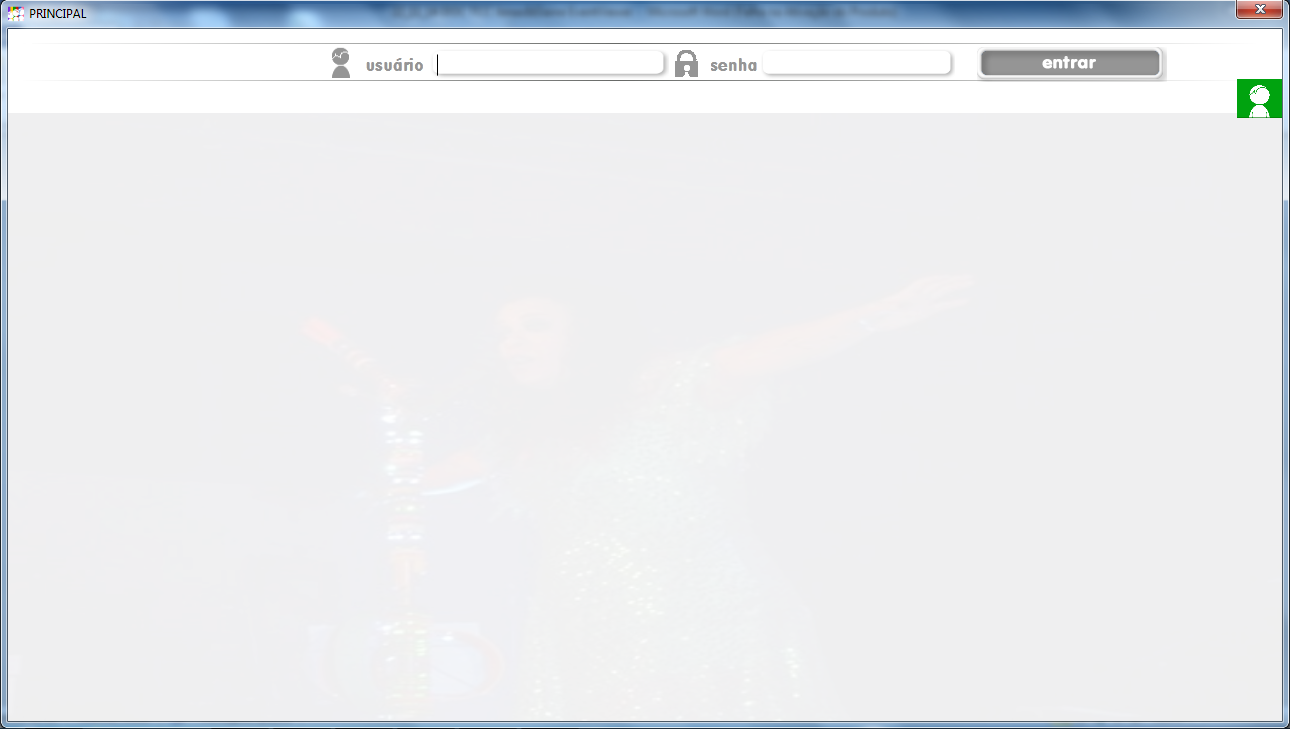


Figura 24 - Tela de autenticação do sistema.

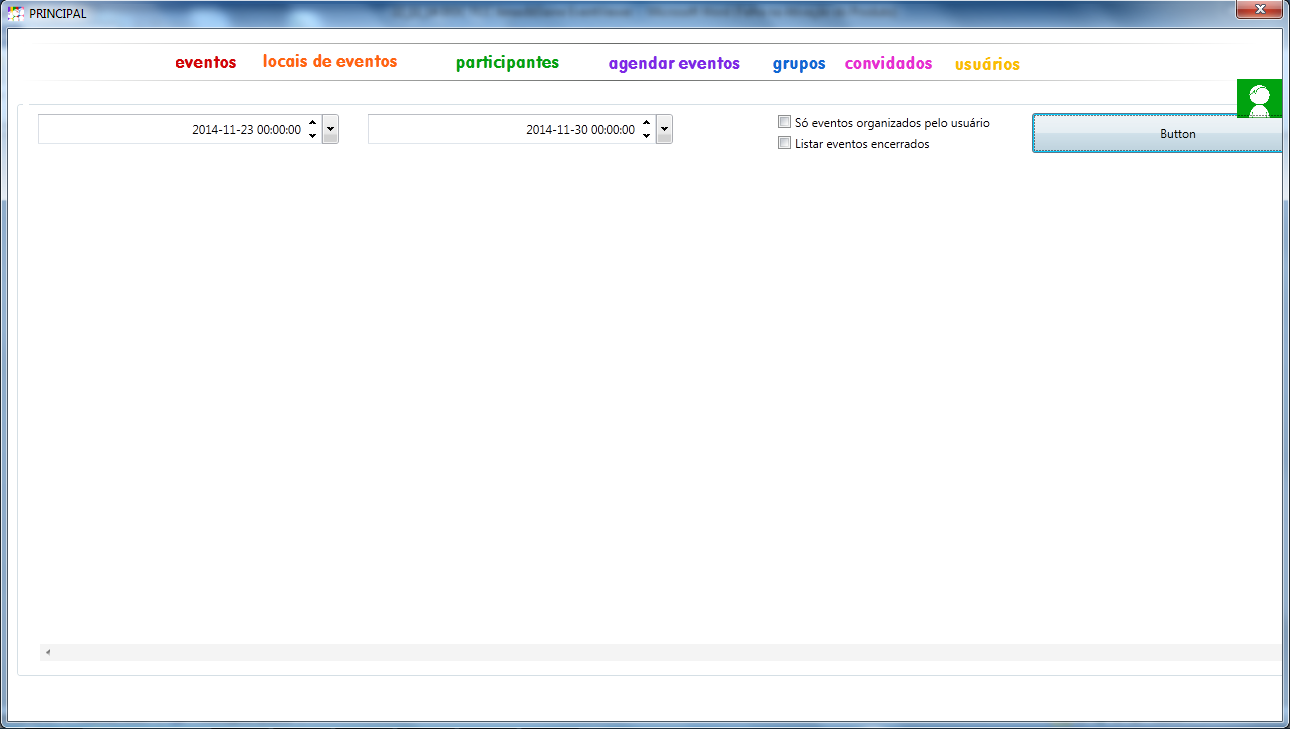


Figura 25 - Tela inicial do sistema.



Figura 26 - Tela de cadastro de evento.

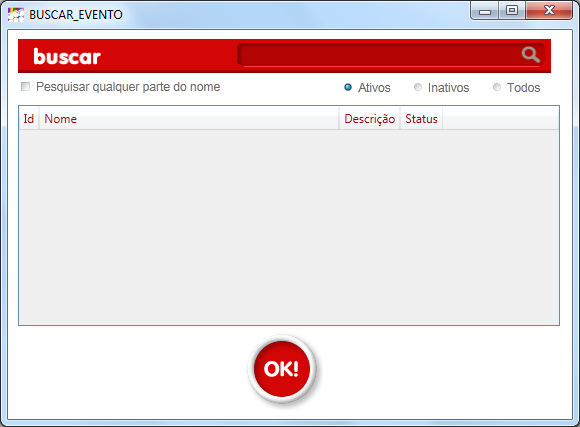


Figura 27 - Tela de consulta de evento.



Figura 28 - Tela de cadastro de local de evento.

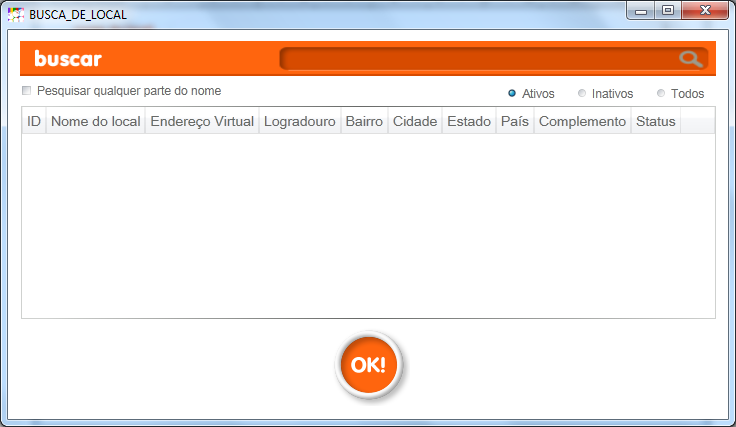


Figura 29 - Tela de consulta de local de evento.

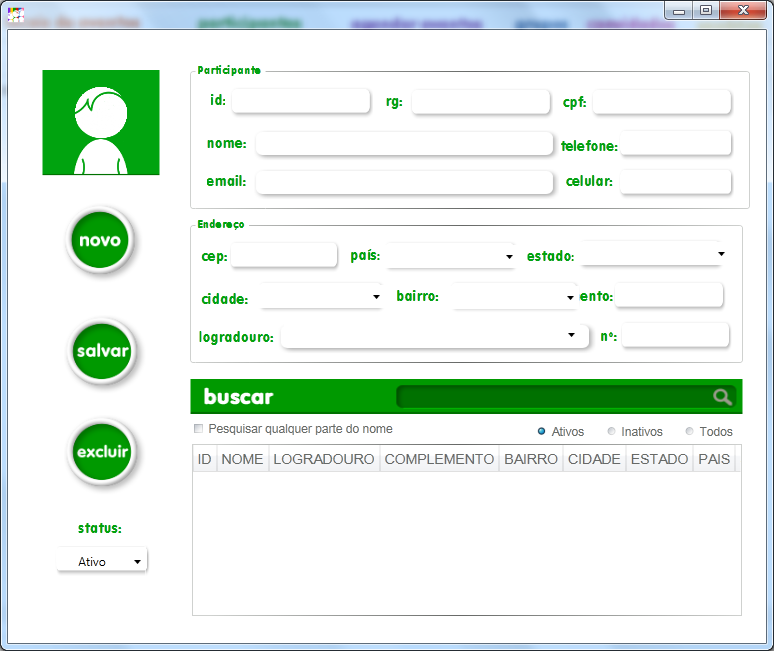


Figura 30 - Tela de cadastro de participante de evento do sistema.



Figura 31 - Tela de cadastro de grupo.

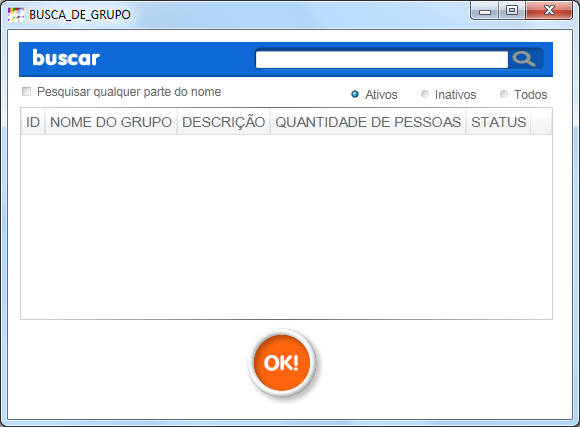


Figura 32 - Tela de consulta de grupos de participantes de evento do sistema.



Figura 33 - Tela de cadastro de convidado do sistema.

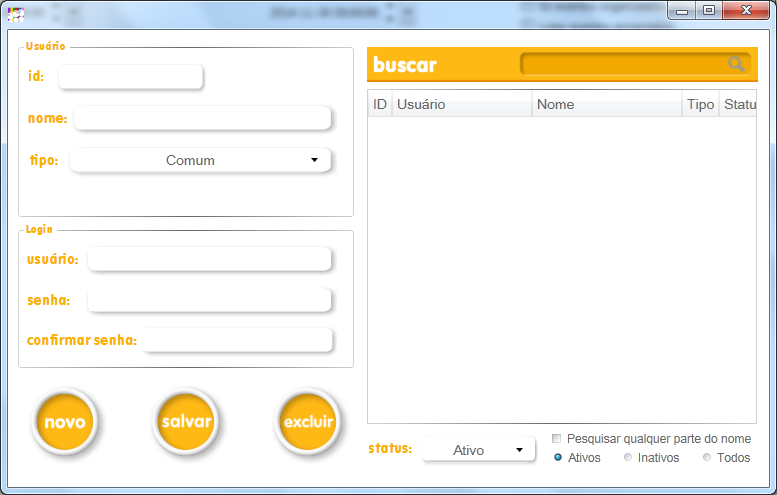


Figura 34 - Tela de cadastro de usuário do sistema.

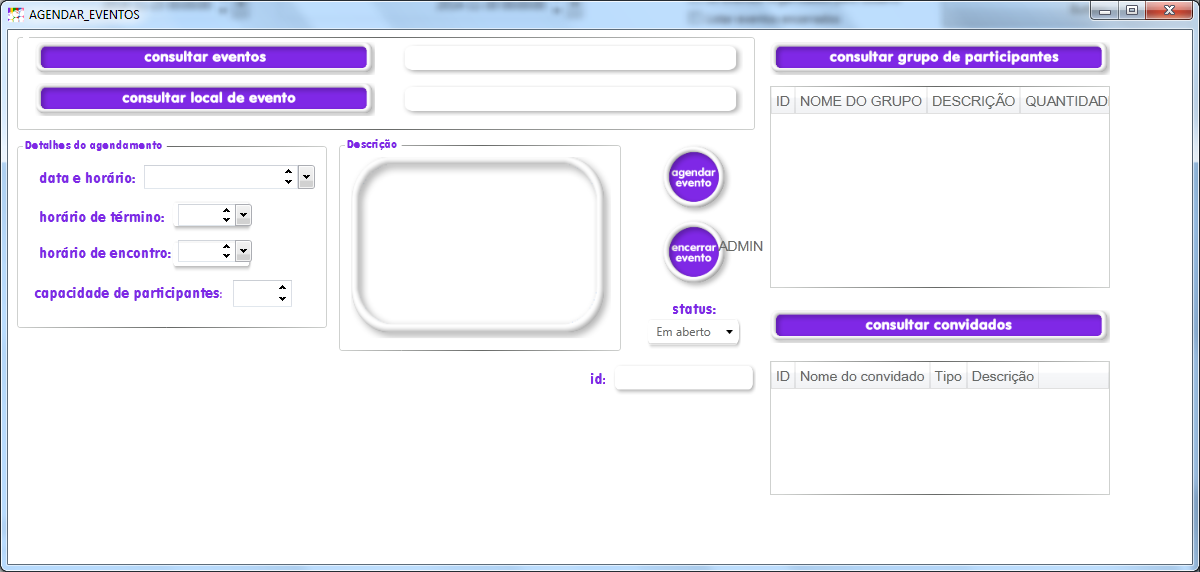


Figura 35 – Tela de agendamento de eventos do sistema.

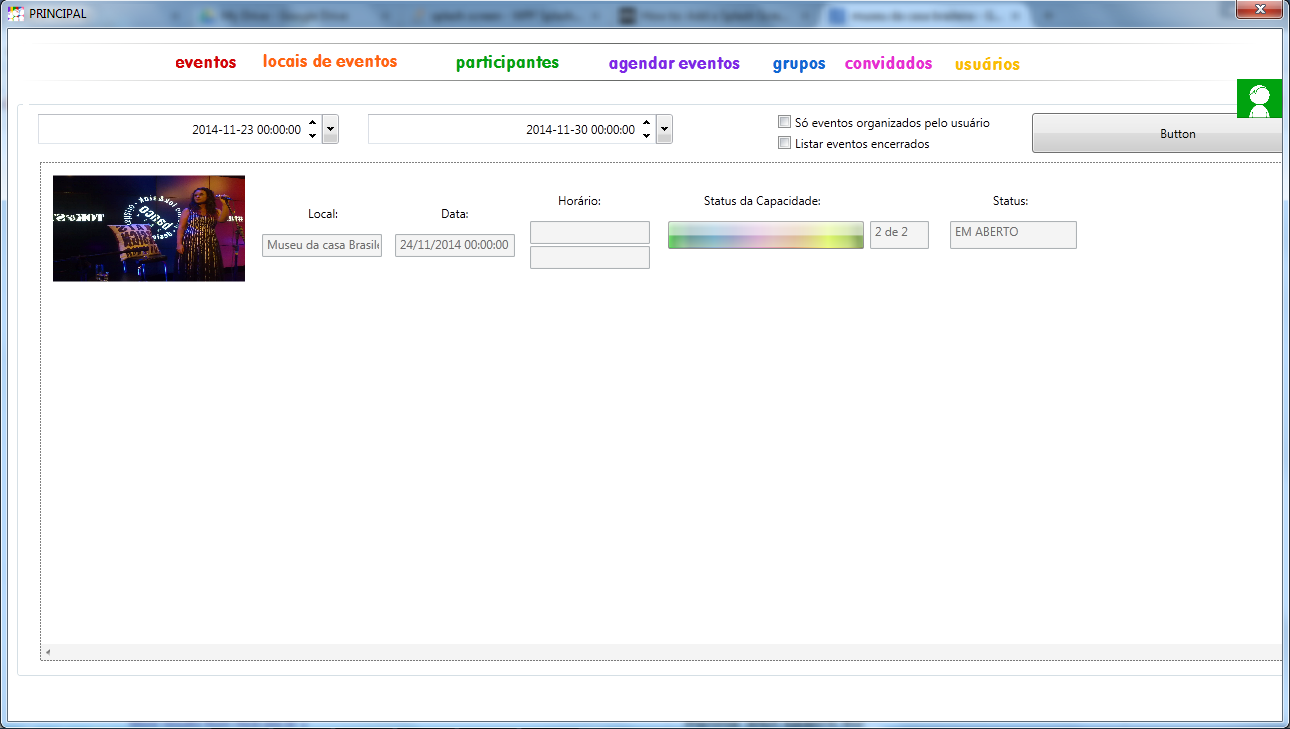


Figura 36 - Tela de consulta de eventos agendados.

# **Manual do usuário**

Apêndice B – Manual do usuário.

# **Conclusões e considerações finais**

Com este projeto conclui-se que o desenvolvimento de um sistema de computador envolve etapas complexas de planejamento, pesquisa de mercado e de tecnologias, registro de atividades e revisões do produto para que o objetivo seja alcançado. É importante ressaltar que durante o desenvolvimento do software a manutenção preventiva tem importante papel na qualidade do produto final e diminui os custos da manutenção corretiva, possibilitando espaço para adição de novos recursos no sistema.

# **Referências bibliográficas**

Elmasri, Ramez. **Sistemas de banco de dados**. São Paulo: Pearson Addison, 2005.

Documentação do MySQL. Disponível em: <http://dev.mysql.com/doc/refman/5.6/en/>. Acesso em: 23 nov. 2014.

Documentação *Windows Presentation Foundation*. Disponível em <http://msdn.microsoft.com/en-us/library/ms754130%28v=vs.110%29.aspx > . Acesso em 23 nov. 2014.

Documentação *Extended WPF Toolkit*. Disponível em <http://wpftoolkit.codeplex.com/documentation >. Acesso em 23 nov. 2014.

# **Glossário**

*Host* – termo utilizado para anfitrião ou organizador de algo.

*Log* – termo utilizado para identificar o registro de ação dentro de um sistema.

*Hardware* – termo utilizado para identificar a parte física de um computador.

*Software* – termo utilizado para identificar a parte lógica de um computador.

# **Apêndice A – Plano de testes.**

Na pagina seguinte será inserido o documento impresso do plano de testes.

Lembrar de contar páginas do manual como paginas do documento.

# **Apêndice B – Manual do usuário.**

Na pagina seguinte será inserido o documento impresso do manual.

Lembrar de contar páginas do manual como paginas do documento.