Eventgram

Documento de Arquitetura de Software

Versão 1.0

Índice Analítico

Conteúdo

[1. Introdução 3](#_Toc361651315)

[1.1 Finalidade 3](#_Toc361651316)

[1.2 Escopo 3](#_Toc361651317)

[1.3 Definições, Acrônimos e Abreviações 3](#_Toc361651318)

[1.4 Referências 3](#_Toc361651319)

[1.5 Visão Geral 3](#_Toc361651320)

[2. Representação Arquitetural 4](#_Toc361651321)

[3. Metas e Restrições da Arquitetura 4](#_Toc361651322)

[4. Visão de Casos de Uso 4](#_Toc361651323)

[4.1 Realizações de Casos de Uso 5](#_Toc361651324)

[5. Visão Lógica 5](#_Toc361651325)

[5.1 Visão Geral 5](#_Toc361651326)

[5.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura 6](#_Toc361651327)

[5.3 Diagrama de componentes 7](#_Toc361651328)

[6. Visão de Processos 8](#_Toc361651329)

[7. Visão de Implantação 8](#_Toc361651330)

[8. Visão da Implementação 8](#_Toc361651331)

[9. Qualidade 8](#_Toc361651334)

[10. Exceções 9](#_Toc361651335)

Documento de Arquitetura de Software

# Introdução

Esse documento provê uma visão de alto nível dos objetivos da arquitetura, dos estilos arquiteturais e componentes que foram selecionados para poder estruturar as funcionalidades propostas pelos casos de usos levantados do Eventgram.

## Finalidade

Este documento oferece uma visão geral arquitetural abrangente do Eventgram, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao utilizando a linguagem de modelagem unificada (UML – *Unified Modeling Language*).

## Escopo

Este Documento de Arquitetura de Software é aplicado ao Eventgram, que será desenvolvido pelos alunos Thiago Alessandro dos Santos Pereira e Weslley Henrique Santos Campos do curso de Sistemas de Informação da Universidade Federal de Sergipe – Campus Itabaiana, como projeto da disciplina Engenharia de Software II.

## Referências

Os seguintes documentos foram utilizados como referência para a elaboração do documento arquitetural:

* Modelo de Análise
* Modelo de Casos de Uso
* Descrição de Casos de Uso
* Documento de Requisitos Funcionais e Não Funcionais
* *Rational Unified Process*

## Visão Geral

Com o objetivo de contemplar todos os aspectos da arquitetura, estratificamos o documento nas seguintes subseções:

* Subseção 2: Descreve o uso de cada visão;
* Subseção 3: Descreve as restrições arquiteturais do sistema;
* Subseção 4: Descreve os requisitos funcionais que causam significante impacto na arquitetura;
* Subseção 5: Descreve a visão lógica da arquitetura;
* Subseção 6: Descreve a visão de processos;
* Subseção 7: Descreve a visão de implantação;
* Subseção 8: Descreve a visão de implementação;
* Subseção 9: Descreve a visão de dados;
* Subseção 10: Descreve as principais características de dimensionamento do software que têm um impacto na arquitetura;
* Subseção 11: Descreve como a arquitetura do software contribui para todos os recursos.
* Subseção 12: Mostra a hierarquia de exceções.

# Representação Arquitetural

Este documento apresenta a arquitetura como um arranjo das visualizações, mencionadas acima. Essas visões são apresentadas como Modelos do StarUML e utiliza a Linguagem de Modelagem Unificada (UML – *Unified Modeling Language).*

Para representar a arquitetura do software, foram utilizados como base os seguintes estilos arquiteturais:

* Camadas;
* Cliente-Servidor;
* Sub-rotinas;
* Baseado em Eventos;
* Repositório de banco de dados.

# Metas e Restrições da Arquitetura

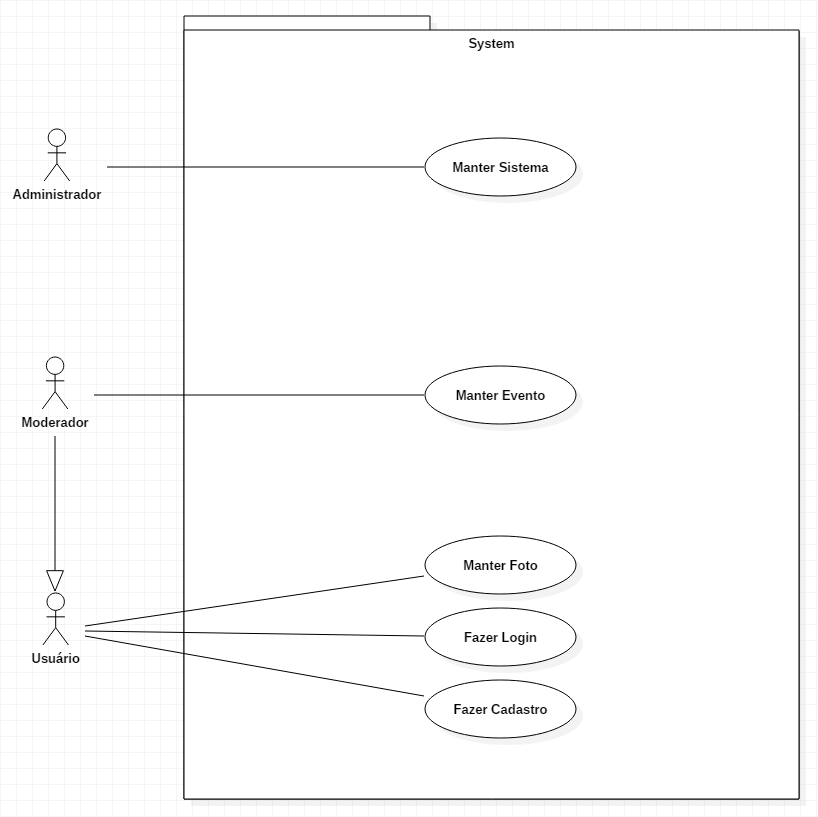
Para que o software tenha um comportamento esperado por nossos *stakeholders*, definimos as seguintes restrições:

* O Sistema deverá ser *Cross-Browser*, ou seja, as funcionalidades por ele oferecidas deverão funcionar plenamente nos mais diversos browsers disponíveis no mercado, a exemplo, do *Mozilla Firefox*, *Google Chrome, Opera, Safari* em suas versões mais recentes;
* O sistema será implementado utilizando os recursos inicialmente disponibilizados pelo *.NET Core* e ASP.NET Core MVC, tendo como linguagem de programação principal o C#;
* O sistema terá os seus dados persistidos no banco de dados MongoDB;
* Apenas usuários autenticados podem ser acessar o sistema;
* O Sistema será multiplataforma, funcionado principalmente nos sistemas operacionais das plataformas Windows, Linux e MAC OS, em suas versões mais recentes, principalmente;
* O Sistema terá como *Design Pattern* o modelo MVC.

# Visão de Casos de Uso

Uma descrição da visão de casos de uso da arquitetura de software. A Visão de Casos de Uso é uma entrada importante para a seleção do conjunto de cenários e/ou casos de uso que são o foco de uma iteração. Ela descreve o conjunto de cenários e/ou os casos de uso que representam alguma funcionalidade central e significativa. Também descreve o conjunto de cenários e/ou casos de uso que possuem cobertura arquitetural substancial (que experimenta vários elementos de arquitetura) ou que enfatizam ou ilustram um determinado ponto complexo da arquitetura.

* CS001 - Cadastro;
* CS002 - Editar Cadastro;
* CS003 - Excluir Cadastro;
* CS004 – Login;
* CS005 – Logout;
* CS006 - Criar Evento;
* CS007 - Deletar Evento;
* CS008 - Adicionar postagem;
* CS009 - Deletar postagem (Usuário & proprietário do evento);
* CS010 - Curtir Postagem;
* CS011 - Descurtir Postagem;
* CS012 - Comentar Postagem;
* CS013 - Excluir Comentário;
* CS014 - Compartilhar Postagem;
* CS015 - Seguir evento;
* CS016 - Unfollow evento;
* CS017 - Seguir usuário;
* CS018 - Unfollow Usuário;
* CS019 - Bloquear usuário;
* CS020 - Selecionar tipo de perfil público ou privado;
* CS021 - Editar postagem;
* CS022 - Editar Evento;

**Figura 1 – Diagrama de Casos de Uso**

## Realizações de Casos de Uso

A seguir temos uma breve descrição de cada caso de uso, utilizado na descrição das principais funcionalidades do sistema. São eles:

* CSU01 – [Cadastrar Usuário] – [O usuário através da tela principal pode cadastrar];
* CSU04 – [Login] – [O usuário através da tela principal poderá fazer login];
* CSU06 – [Criar Evento] – [O usuário já conectado ao sistema, poderá criar um evento];
* CSU08 – [Adicionar Postagem] – [O usuário poderá adicionar uma postagem no seu evento];
* CSU010 – [Curtir Postagem] – [O usuário poderá curtir uma postagem];
* CSU014 – [Compartilhar Postagem] – [O usuário poderá compartilhar uma postagem onde tem permissão];
* CSU015 – [Seguir Evento] – [Um usuário poderá seguir um evento];
* CSU017 – [Seguir Usuário] – [Um usuário poderá seguir um outro usuário].

# Visão Lógica

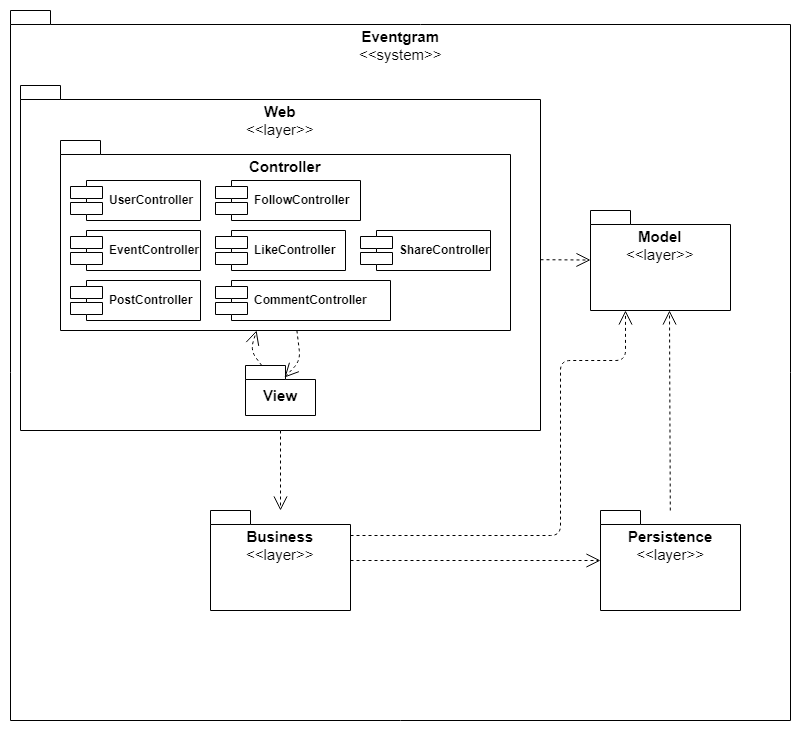
## Visão Geral

A visão lógica do Eventgram é composta principalmente por 3 pacotes:

* Model: Esse pacote representa a implementação da parte lógica do domínio da aplicação. Aqui serão armazenadas classes que representam as entidades da aplicação e classes responsáveis pela persistência dos dados;
* Views: Nesse pacote são armazenados os componentes responsáveis por prover a interface gráfica da aplicação;
* Controller: Nesse pacote são armazenados os componentes do tipo Controller. Tais componentes são responsáveis por relacionar os Models às Views, de acordo com a interação do usuário.

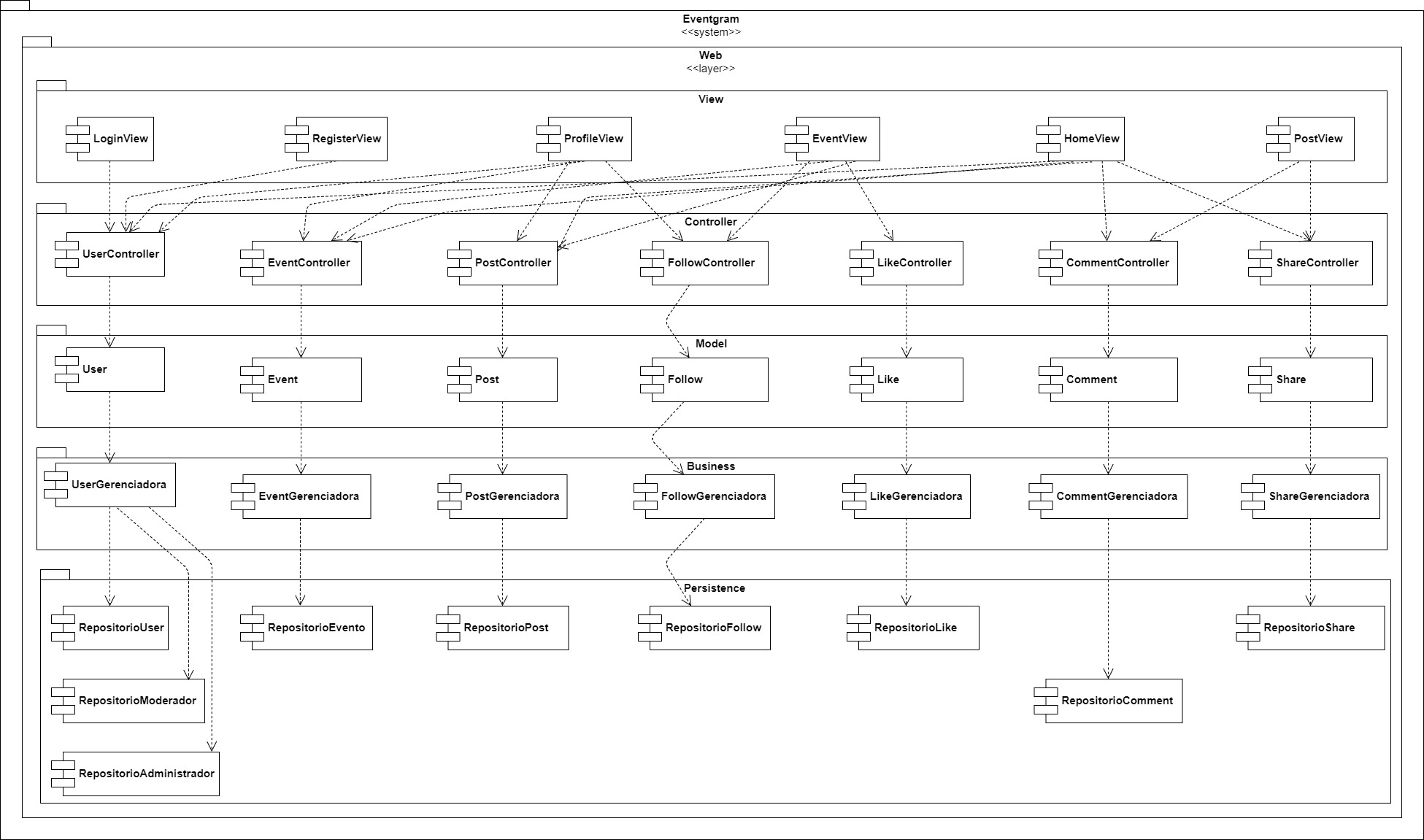
## Pacotes de Design Significativos do Ponto de Vista da Arquitetura

**Figura 2 – Diagrama de Pacotes**

****

## Diagrama de componentes

**Figura 3 – Diagrama de Componentes**

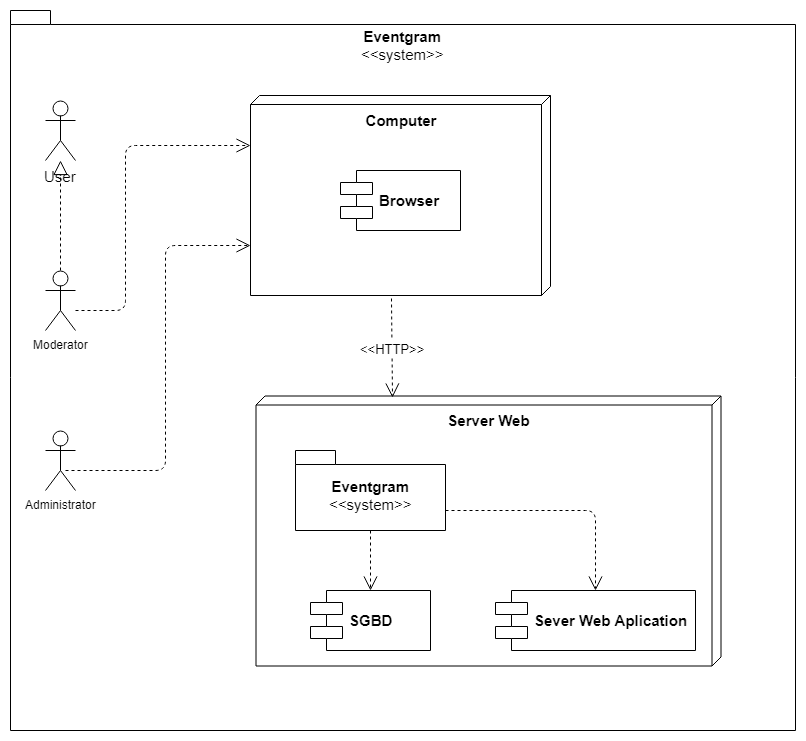
****

# Visão de Processos

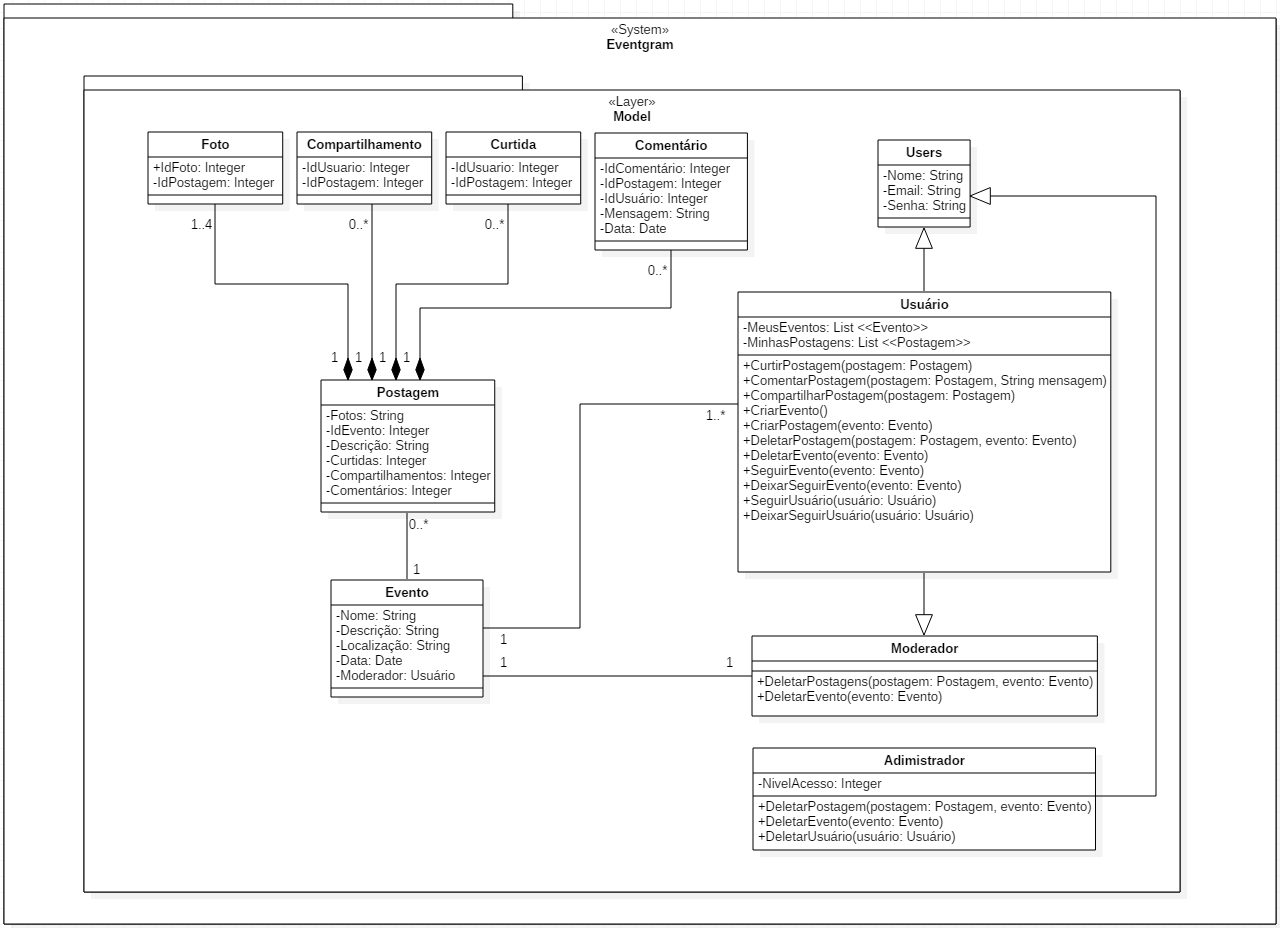
Ainda não foi definida

# Visão de Implantação

**Figura 4 – Diagrama de Implantação**

****

# Visão da Implementação

**Figura 5 – Diagrama de Classes**

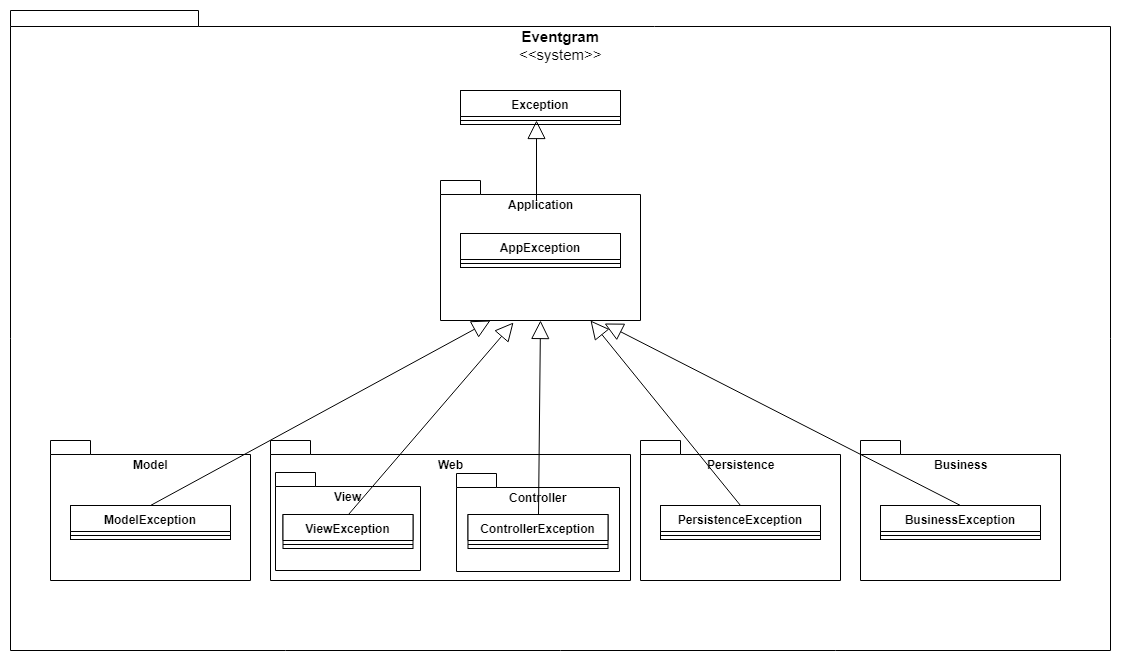
# Qualidade

Uma descrição de como a arquitetura do software contribui para todos os recursos (exceto a funcionalidade) do sistema: extensibilidade, confiabilidade, portabilidade e assim por diante. Se essas características possuírem significado especial, como implicações de segurança, garantia ou privacidade, elas deverão ser delineadas claramente.

Especificar os requisitos não funcionais.

# Exceções

O tratamento de exceções se dará preferencialmente em camadas. O sistema possuirá tratamento de exceções específicas definidas pela *Application.Exception* que utilizará os tratamentos de exceções genéricos definidos em *System.Exception.*

**Figura 6 – Diagrama da Arquitetura de Exceções**