DOCUMENTO DE ARQUITETURA DE SOFTWARE - DAS

Solução: SGE

Este documento tem como objetivo descrever a arquitetura lógica e física do projeto Sistema de Gestão de Eventos.

Documento de Arquitetura de Software

|  |  |
| --- | --- |
| **SGE – Sistema de Gestão de Eventos** | |
| **Gestor do Projeto** | **Arquiteto de Software** |
| Luan Augusto David de Alencar Lima | Luan Augusto David de Alencar Lima |
| luanalencar134@gmail.com | luanalencar134@gmail.com |
| (83) 99938-2839 | (83) 99938-2839 |

|  |
| --- |
| **Objetivo deste Documento** |
| Este documento tem como objetivo descrever as principais decisões de projeto tomadas pela equipe de desenvolvimento e os critérios considerados durante a tomada destas decisões. Suas informações incluem aparte de *hardware* e *software* do sistema. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Histórico de Revisão* | | | | |
| Data | Demanda | Autor | Descrição | Versão |
| 09/11/2021 | [XX000000] | *Luan Augusto David de Alencar Lima* | Criação do documento | 1.0 |
|  |  |  |  |  |
|  |  |  |  |  |

Sumário

[1. INTRODUÇÃO 3](#_Toc452479025)

[1.1 Finalidade 3](#_Toc452479026)

[1.2 Escopo 3](#_Toc452479027)

[1.3 Definições, Acrônimos e Abreviações 3](#_Toc452479028)

[1.4 Referências 4](#_Toc452479029)

[2. REPRESENTAÇÃO ARQUITETURAL 4](#_Toc452479030)

[3. REQUISITOS E RESTRIÇÕES ARQUITETURAIS 5](#_Toc452479031)

[4. VISÃO DE CASOS DE USO 5](#_Toc452479032)

[4.1 Casos de Uso significantes para a arquitetura 5](#_Toc452479033)

[5. VISÃO LÓGICA 7](#_Toc452479034)

[5.1 Visão Geral – pacotes e camadas 7](#_Toc452479035)

[6. VISÃO DE IMPLEMENTAÇÃO 9](#_Toc452479036)

[6.1 Caso de Uso [00X] 9](#_Toc452479037)

[6.1.1 Diagrama de Classes 9](#_Toc452479038)

[6.1.2 Diagrama de Sequência 10](#_Toc452479039)

[7. VISÃO DE IMPLANTAÇÃO 11](#_Toc452479040)

[8. DIMENSIONAMENTO E PERFORMANCE 11](#_Toc452479041)

[8.1 Volume 11](#_Toc452479042)

[8.2 Performance 11](#_Toc452479043)

[9. QUALIDADE 11](#_Toc452479044)

# INTRODUÇÃO

## Finalidade

Este documento fornece uma visão arquitetural abrangente do Sistema de Gestão de Eventos a fim de descrever o software da forma mais apurada possível, usando diversas visões de arquitetura para **representar** diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.

O documento irá adotar uma estrutura baseada na visão “4+1” de modelo de arquitetura [KRU41] inicialmente por Philippe Kruchten’s.



Figura 1 – Arquitetura 4+1

## Escopo

Este Documento de Arquitetura de Software descreve os aspectos de projeto do Sistema de Gestão de Eventos que são considerados significativamente relevantes na visão de arquitetura, ou sejam os elementos e comportamentos que são fundamentais para guiar e estabelecer diretrizes para a construção do sistema. Stakeholders que exigirem um entendimento técnico do Sistema de Gestão de Eventos são convidados para tomarem ciência deste documento e posteriormente, participar do processo de revisão dos artefatos complementares à este projeto como Documentos de Casos de Uso, Diagramas de Processos de Negócio e demais artefatos que contribuem para o enriquecimento da documentação do produto final a ser entregue. Por fim, o código-fonte poderá ser revisado após a entrega.

## Definições, Acrônimos e Abreviações

QoS – Quality of Service, ou qualidade de serviço. Termo utilizado para descrever um conjunto de qualidades que descrevem as requisitos não-funcionais de um sistema, como performance, disponibilidade e escalabilidade [QOS].

|  |  |
| --- | --- |
| ***Abreviação*** | ***Descrição*** |
| **JPA** | Java Persistence API |
| **DTO** | Objetos de transferência de dados |
| **MAPPER** | Ferramenta que representa um tipo de interface referente ao framework de mapeamento de DTOs para classes de negócio, o MapStruct, que vai gerar uma implementação baseada em uma definição de interface que será feita. |

Tabela 1-Tabela de abreviações

|  |  |
| --- | --- |
| ***Termo*** | ***Definição*** |
| **Gateway** | O gateway permite gerenciar o tráfego de entrada e saída da rede de microsserviços através de serviços de balanceamento de carga, roteamento e proxy. |
| **Feign** | É um comunicador de API que se seja necessário o sistema com outra API através de uma busca HTTP terá que fazer uma chamada na API externa. Então, o Feign será a ferramenta que fará isto. |
| **Docker** | Com o Docker, é possível lidar com os containers como se fossem máquinas virtuais modulares e extremamente leves. Além disso, os containers oferecem maior flexibilidade para você criar, implantar, copiar e migrar um container de um ambiente para outro otimizando assim aplicações. |
| **Elasticsearch** | Banco de dados não relacional com mecanismo de busca e análise de dados distribuído. Através de uma arquitetura baseada em REST podemos torna o aplicativo mais performático em velocidade e escalabilidade. |

Tabela 2-Tabela de abreviações

## Referências

[KRU41]: The “4+1” view model of software architecture, Philippe Kruchten, November 1995, <http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf>

[QOS] <https://docs.oracle.com/cd/E19636-01/819-2326/6n4kfe7dj/index.html>

JPA <https://www.devmedia.com.br/introducao-a-jpa-java-persistence-api/28173>

DTO [https://www.lambda3.com.br/2021/04/dto-vs-viewmodel/l](https://docs.oracle.com/cd/E19636-01/819-2326/6n4kfe7dj/index.html)

Mapper <https://medium.com/dev-cave/mapstruct-mapeando-seus-dtos-para-model-8bc362b628fe>

Gateway <https://docs.microsoft.com/pt-br/azure/architecture/microservices/design/gateway>

Feign <https://emmanuelneri.com.br/2018/06/05/clients-dinamicos-com-feign-e-eureka/#:~:text=O%20Feign%20%C3%A9%20uma%20boa,em%20Java%20de%20forma%20f%C3%A1cil.&text=Feign.,-builder()%3A%20Inst%C3%A2ncia>

Docker <https://www.redhat.com/pt-br/topics/containers/what-is-docker>

Elasticsearch <https://www.elastic.co/pt/what-is/elasticsearch>

# REPRESENTAÇÃO ARQUITETURAL

Este documento irá detalhar as visões baseado no modelo “4+1” [KRU41], utilizando como referência os modelos definidos na MDS. As visões utilizadas no documento serão:

|  |  |  |  |
| --- | --- | --- | --- |
| Visão | Público | Área | Modelo da MDS |
| Lógica | Analistas | Realização dos Casos de Uso | Contém as definições do sistema em visão de diagramas de classes e objetos que descrevem os serviços que o sistema irá prover ao usuário final. |
| Processo | Integradores | Performance, Escalabilidade, Concorrência | São os processos que irão dar o suporte de entendimento dos mecanismos do sistema, representados em forma de diagramas de comunicação, sequência e atividades. |
| Implementação | Programadores | Componentes de Software | Define as especificações do sistema como interfaces e componentes utilizados, detalhando como estes componentes são dispostos em camadas e subsistemas. |
| Implantação | Gerência de Configuração | Nodos físicos | Descreve como os nós de hardware serão dispostos para trazer vida ao sistema, bem como configurações físicas necessárias para instalação, empacotamento e distribuição. |
| Caso de Uso | Todos | Requisitos funcionais | Define os principais direcionadores do sistema, que são compostos pelos requisitos funcionais do sistema. |
| Dados | Especialistas em dados  Administradores de dados | Persistência de dados | Descreve em uma visão de dados, a descrição das principais tabelas utilizadas e diagramas de entidade-relacionamento. |

Tabela 1 – Visões, Público, Área e Artefatos da MDS

# REQUISITOS E RESTRIÇÕES ARQUITETURAIS

Esta seção descrever os requisitos de software e restrições que tem um impacto significante na arquitetura.

|  |  |
| --- | --- |
| Requisito | Solução |
| Linguagem | Java 8 |
| Plataforma | Toda a parte de backend deverá ser implementada pela equipe na IDE Intellij para um melhor desenvolvimento e evolução do sistema. |
| Segurança | Todo usuário deverá ter sua senha criptografada no BD e um usuário não poderá se cadastrar caso já tenha seu e-mail cadastrado na plataforma. |
| Persistência | Para a persistência dos dados será utilizado JPA e PostgreSQL. |
| Internacionalização (i18n) | Plataforma com alcance nacional. |

Tabela 2 – Exemplo de requisitos e restrições

# VISÃO DE CASOS DE USO

Esta seção lista as especificações centrais e significantes para a arquitetura do sistema.

## Atores

|  |  |
| --- | --- |
| ***Abreviação*** | ***Descrição*** |
| **Administrador** | Responsável por realizar, manter e aprovar inscrições de candidatos |
| **Candidato** | Usuário final da aplicação. |

Figura 2 – Exemplo de Diagrama com os casos de uso significativos e atores  
Tabela 4-Tabela de atores do sistema

# 4.2 Descrição dos principais casos de uso ou história de usuário

|  |  |
| --- | --- |
| ***Ator*** | ***Descrição*** |
| **Evento** | Um Evento possui um título, período (início e fim), horário, descrição, quantidade de vagas, tipo do evento, tipo de inscrição, valor, local do evento e perguntas para realizar a inscrição no evento |
| **Tipo de Evento** | Workshop, Palestra, Mini Curso, Treinamento, Musica |
| **Pergunta** | Uma pergunta possui apenas um título e obrigatoriedade. |
| **Usuário** | Um usuário possui nome, CPF, e-mail, telefone, data de nascimento e uma chave única de inscrição aos eventos. |
| **Inscrição** | Uma inscrição é feita por um candidato para um Evento. Para se inscrever, o candidato deve responder perguntas solicitadas para o evento. A pré-inscrição possui uma situação e um código do candidato. |
| **Tipo de Situação** | Aguardando, Aprovação, Aprovada, Recusada, Cancelada, etc. |

# VISÃO LÓGICA

Descrever uma visão lógica da arquitetura. Descrever as classes mais importantes, sua organização em pacotes de serviços e subsistemas, e a organização desses subsistemas em camadas. Também descreve as realizações dos casos de uso mais importantes, por exemplo, aspectos dinâmicos da arquitetura. Diagramas de classes e sequência devem ser incluídos para ilustrar os relacionamentos entre as classes significativas na arquitetura, subsistemas, pacotes e camadas.

## Visão Geral – pacotes e camadas

D:\Trabalho\Atividades\documentacao_arquitetura_sistemas\graficos\Diagrama_camadas.emf

Figura 2 – Exemplo de Diagrama de Camadas da Aplicação

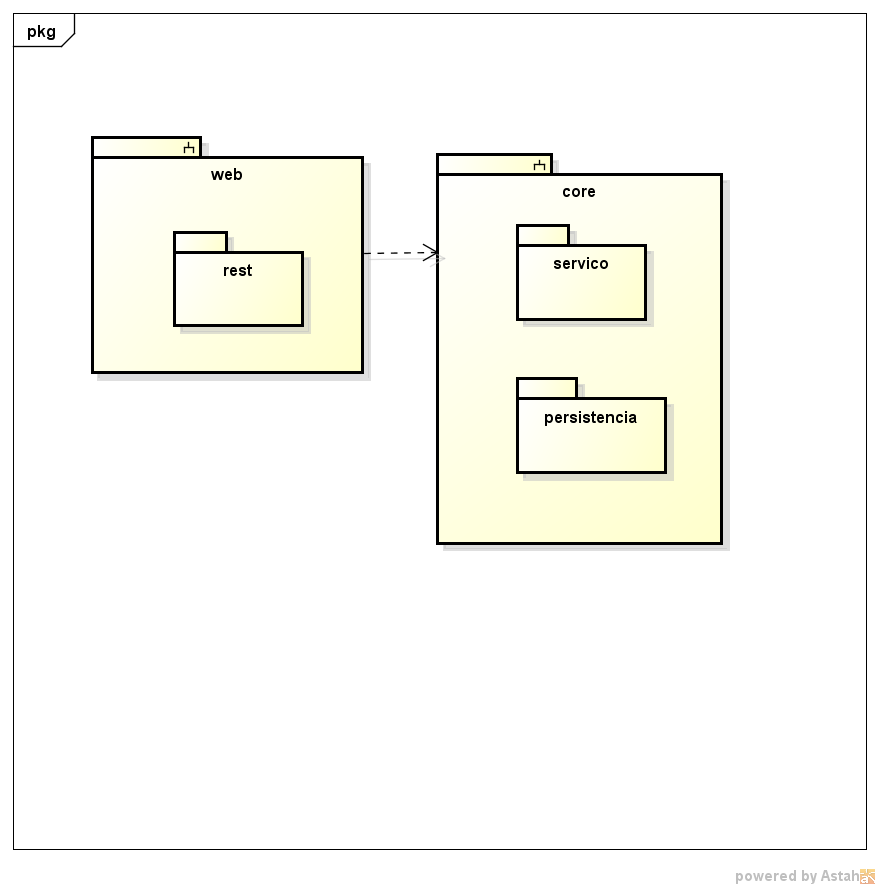


Figura 3 – Exemplo de Diagrama de Pacotes da Aplicação

# VISÃO DE IMPLEMENTAÇÃO

### Diagrama de Classes

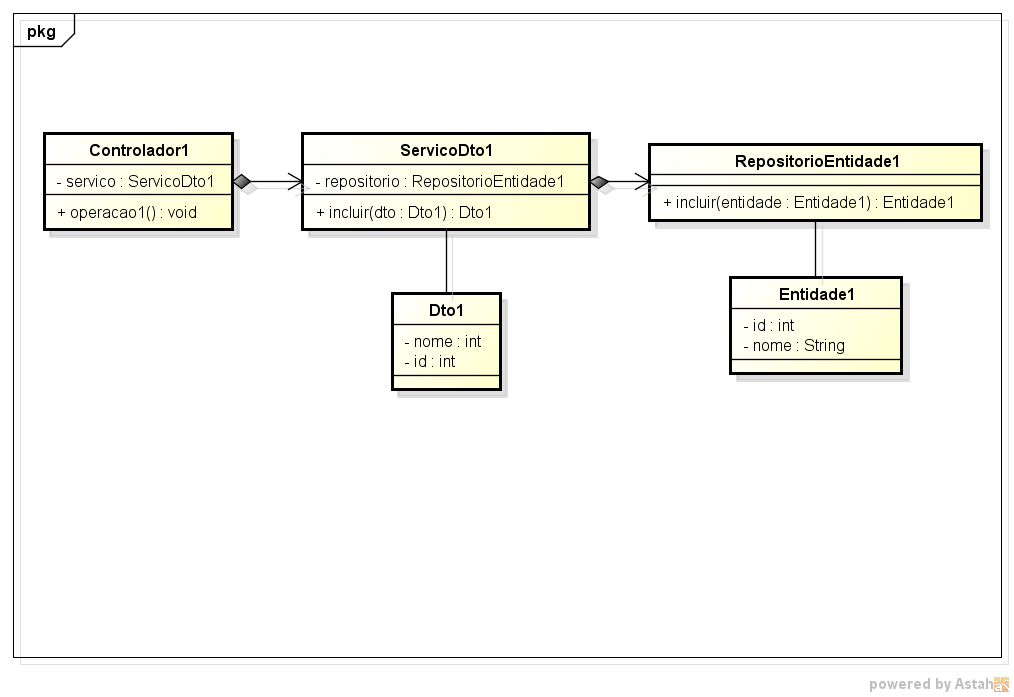


Figura 20 – Exemplo de Diagrama de Classes

### Diagrama de Sequência

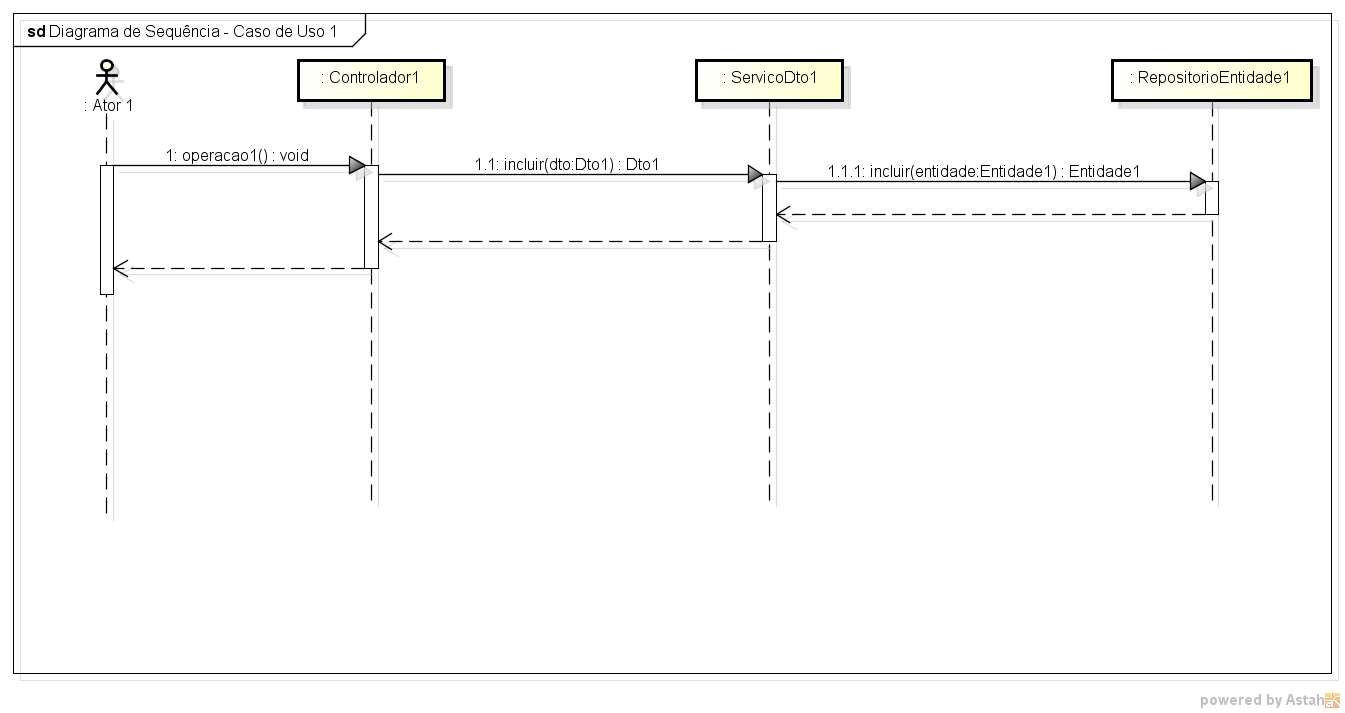


Figura 20 – Exemplo de Diagrama de Sequência

# VISÃO DE IMPLANTAÇÃO

Descrever os nodos físicos, as configurações e os artefatos que serão implantados.

### Chamadas entre serviços

Em u­­­ma arquitetura de microsserviço, temos eu nos integrar a diversos serviços para que uma funcionalidade ou sistemas fiquem completos. Com o intuito de facilitar a integração de serviços, o Spring Clound fornece o projeto Feign, eu seria uma maneira declarativa de criar web services clientes.

O Feign conecta seu código a APIs HTTP utilizando anotações em um modelo de template REST.

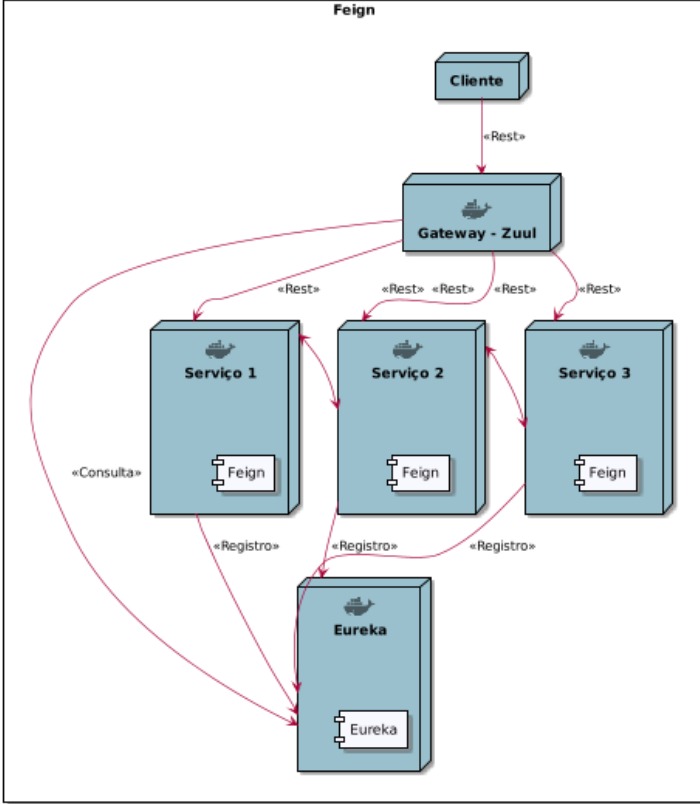


Figura 20 – Exemplo de Diagrama de Implantação Java

# DIMENSIONAMENTO E PERFORMANCE

## Volume

Enumerar de 1 a 5 os itens relativos ao volume de acesso aos recursos da aplicação:

* Número de estimado usuários: 5
* Número estimado de acessos diários: 5
* Número estimado de acessos por período: 2
* Tempo de sessão de um usuário: 3

## Performance

Enumerar os itens referentes à resposta esperada do sistema:

* Tempo máximo para a execução de determinada transação: 5