**FÁBRICA DE TECNOLOGIAS TURING**

****



DOCUMENTO DE ARQUITETURA DE SOFTWARE

Gerlabs

VERSÃO: 0.1

Autor:

José Francisco de Oliveira Júnior

Anápolis – GO

2018

**HISTÓRICO DE REVISÃO**

|  |  |  |  |
| --- | --- | --- | --- |
| Versão | Data | Responsável | Descrição |
| 0.1 | 26/04/2018 | José Francisco de Oliveira Júnior | Início do documento de arquitetura de software. |

**SUMÁRIO**

[**1.** **INTRODUÇÃO** 4](#_Toc512526973)

[**1.1** **FINALIDADE** 4](#_Toc512526974)

[**1.2** **ESCOPO** 4](#_Toc512526975)

[**1.3** **DEFINIÇÕES, ACRÔNIMOS E ABREVIAÇÕES** 4](#_Toc512526976)

[**1.4** **REFERÊNCIAS** 5](#_Toc512526977)

[**1.5** **VISÃO GERAL** 5](#_Toc512526978)

[**2.** **REPRESENTAÇÃO DA ARQUITETURA** 6](#_Toc512526979)

[**3.** **METAS E RESTRIÇÕES DA ARQUITETURA** 6](#_Toc512526980)

[**4.** **VISÃO DE CASOS DE USO** 6](#_Toc512526981)

[**5.** **VISÃO LÓGICA** 7](#_Toc512526985)

[**5.1** **VISÃO GERAL** 7](#_Toc512526987)

[**5.2** **CAMADAS** 7](#_Toc512526988)

[**5.3** **PACOTES DE DESIGN SIGNIFICATIVOS DO PONTO DE VISTA DA ARQUITETURA** 8](#_Toc512526989)

[**6.** **VISÃO DE PROCESSOS** 8](#_Toc512526990)

[**7.** **VISÃO IMPLEMENTAÇÃO** 8](#_Toc512526991)

[**8.** **VISÃO DE IMPLANTAÇÃO** 10](#_Toc512526992)

[**9.** **TAMANHO E DESEMPENHO** 11](#_Toc512526996)

[**10.** **QUALIDADE** 11](#_Toc512526997)

## **INTRODUÇÃO**

## **FINALIDADE**

Este documento apresenta uma visão geral abrangente da arquitetura do sistema e utiliza uma série de visões arquiteturais diferentes para ilustrar os diversos aspectos do sistema. Sua intenção é capturar e transmitir as decisões significativas do ponto de vista da arquitetura que foram tomadas em relação ao sistema.

## **ESCOPO**

Este documento fornece uma visão arquitetural abrangente do sistema Gerlabs, usando diversas visões arquiteturais para representar os diferentes aspectos do mesmo.

O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao desenvolvimento do software.

## **DEFINIÇÕES, ACRÔNIMOS E ABREVIAÇÕES**

|  |  |
| --- | --- |
| Termo | Descrição |
| API | *Application Programming Interface*. Definição de um conjunto de funções ou métodos que permitem a utilização de um determinado serviço. |
| *Build* | Uma versão operacional de um sistema ou componente, contendo um subconjunto específico de funcionalidades que serão providos pelo produto final. |
| *Deployment* | Ato de colocar arquivos ou aplicações em ambiente operacional. |
| *Framework* | Uma solução que pode ser reusada total ou parcialmente no desenvolvimento de aplicações de um domínio particular. Pode ser utilizado como base para o desenvolvimento de novos componentes ou aplicações. |
| GRASP | *General Responsibility Assignment Software Patterns*. Conjunto de padrões de projeto que provê um guia para definição de responsabilidades para as classes que fazem parte do sistema. |
| *Hibernate* | Mecanismo de persistência utilizado para a realização de mapeamento objeto-relacional em aplicações Java™. |
| IDE | *Integrated Development Environment*. Ambiente para desenvolvimento integrado de aplicações. |
| Javadoc | Ferramenta de software para geração de documentação de API em HTML a partir de comentários adicionados em formato específico dentro do código fonte Java™. |
|  | Padrão para realização de transformações entre os dados presentes em tabelas relacionais e objetos manipulados por aplicações orientadas a objetos. |
| MVC | *Model-View-Controller*. Padrão de projeto arquitetural no qual o *view* é responsável pela apresentação da aplicação, o *model* representa os dados e regras de negócio da aplicação e o *controller* define as interações entre o *model* e o *view* a partir das ações e dados fornecidos pelo usuário. |
| *Plug-in* | Componente de software que adiciona funcionalidades específicas a um sistema maior. |
| *Scriptlets* | Trechos de código Java™ presentes em páginas JSP. |
| SGBD | Abreviação de Sistema de Gerenciamento de Banco de Dados. |
| *Spring framework* | *Framework* para desenvolvimento de aplicações web. |
| *Tag* | Marcação característica de linguagens com marcadores (HTML e XML, entre outras). |

Tabela 1 – Definições, Acrônimos e Abreviações

## **REFERÊNCIAS**

* Scrum metodologia ágil para desenvolvimento de software. (<http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>).
* OpenUp processo de desenvolvimento de software. (<http://epf.eclipse.org/wikis/openuppt/>).
* RUP processo proprietário de engenharia de software. (<http://www.wthreex.com/rup/portugues/process/workflow/ana_desi/co_swarch.htm>).
* Philippe Kruchten 1995, “The 4+1 view modelo f architecture”, *IEEE* software. 12(6), novembro de 1995. A origem das visões 4+1 utilizadas para descrição de arquitetura no RUP. (<http://www.wthreex.com/rup/portugues/process/referenc.htm#KRU95>).

## **VISÃO GERAL**

As secções e subsecções a seguir estão organizadas e definidas no decorrer do documento de acordo com o processo definido pelo RUP.

* Representação da Arquitetura;
* Metas e Restrições da Arquitetura;
* Visão de Casos de Uso;
* Realizações de Casos de Uso;
* Visão Lógica;
* Visão Geral;
* Pacotes de Design Significativos do Ponto de Vista da Arquitetura;
* Visão de Processos;
* Visão de Implementação;
* Visão de Implantação;
* Visão de Dados;
* Tamanho e Desempenho;
* Qualidade.

## **REPRESENTAÇÃO DA ARQUITETURA**

A representação adotada para a arquitetura foi o denominado “Modelo de Visão 4 + 1”, proposta por Kruchten, que divide a arquitetura em cinco visões: Visão de Casos de Uso, Visão Lógica, Visão de Implementação, Visão de Processos e Visão de Implantação. Essas visões são apresentadas como Modelos do Astah Community e utilizam a Linguagem de Modelagem Unificada (UML).

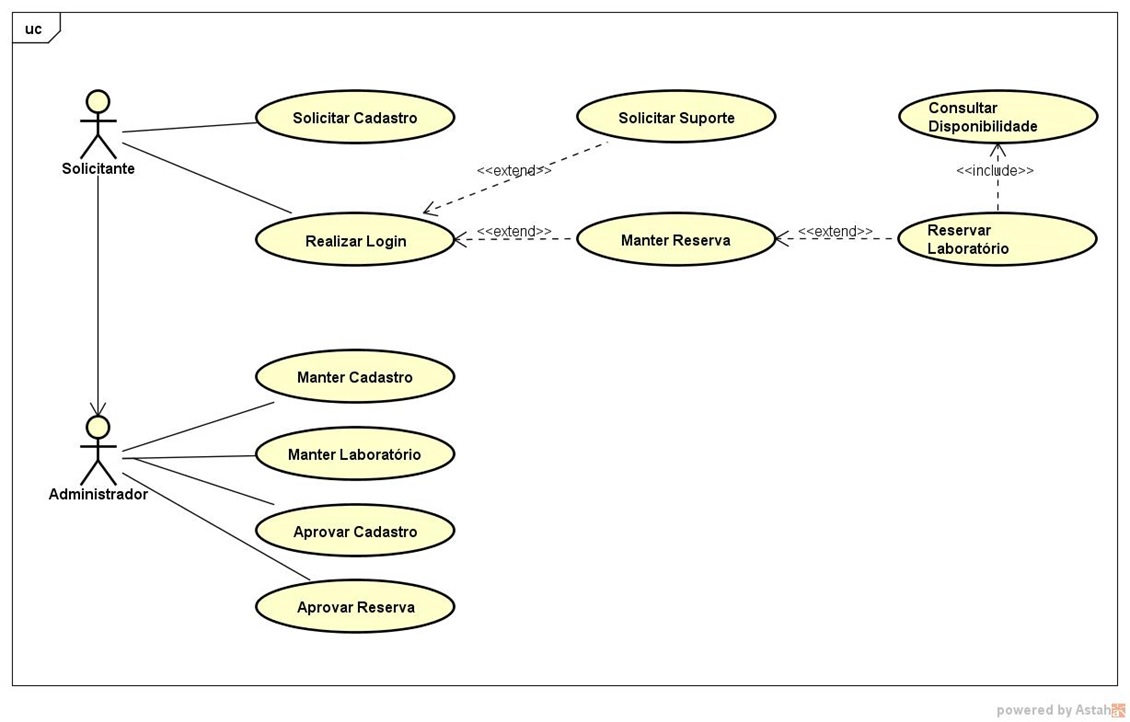
## **METAS E RESTRIÇÕES DA ARQUITETURA**

Para a proposta da arquitetura, foram considerados fatores como finalidade do sistema, tipo de usuários e ambiente de execução. Sendo assim, a arquitetura a ser adotada precisa atender às seguintes características.

* Modularidade: Faz com que o sistema possua partes não acopladas, facilitando a possível substituição de componentes do mesmo. Essa característica se mostra muito importante, por se tratar de um aplicativo voltado para Web, onde frequentemente surgem novas funcionalidades.
* Manutenibilidade: O sistema deve está no ar 24 horas por dia, o que exige uma flexibilidade a permitir a manutenção fácil e rápida do mesmo.
* Reusabilidade: O reuso é considerado hoje como um dos mais importantes fatores durante o desenvolvimento de um sistema. Com isto é possível à utilização de classes e componentes em outros projetos, favorecendo o tempo de produção e a qualidade do produto desenvolvido.

## **VISÃO DE CASOS DE USO**

Figura 1 - Diagrama de Casos de Uso - Gerlabs









## **VISÃO LÓGICA**



## **VISÃO GERAL**

Esta subseção descreve toda a decomposição do modelo de design em termos de camadas e de hierarquia de pacotes.

## **CAMADAS**

A visão em camadas descreve a organização dos elementos de projeto em grupos, independentemente de seu empacotamento físico ou deployment. As camadas da arquitetura podem ser vistas na figura 8 – Camadas da arquitetura, e possuem os seguintes papéis:

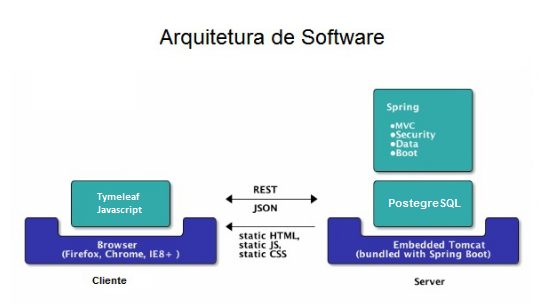


Figura 4 – Camadas do Gerlabs

## **PACOTES DE DESIGN SIGNIFICATIVOS DO PONTO DE VISTA DA ARQUITETURA**

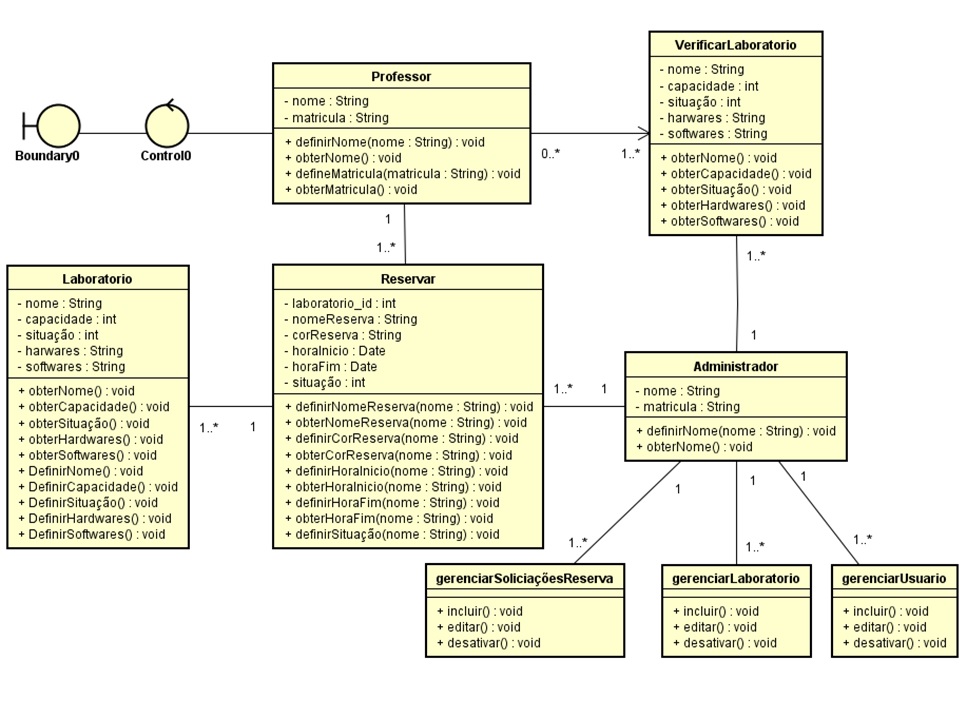


Figura 6 – Diagrama de Classes

## **VISÃO DE PROCESSOS**

O sistema é gerenciado por meio de processos, esses processos podem ser divididos com base na sua capacidade de influência para o sistema como um todo, podendo ser classificados em dois tipos:

**Processos leves**: São processos de baixa importância dentro do sistema, tais como threads de baixa prioridade criadas para processamento paralelo.

**Processos pesados**: São processos de alto impacto dentro do sistema como um todo, em que o mau gerenciamento pode comprometer outras áreas do sistema, tais como threads criadas para a interação com o usuário.

## **VISÃO IMPLEMENTAÇÃO**

A arquitetura proposta pelo sistema Gerlabs possui diversos componentes de negócio que utilizam as seguintes tecnologias.

|  |  |
| --- | --- |
| **Tecnologias** | **Descrição** |
| Java 8 | Linguagem de programação OO. |
| Spring Boot | Esse Framework é um recurso de automatização que serve para ajudar a preparar o ambiente de configuração do Spring. |
| Spring JPA | O Spring JPA é útil para desenvolvedores que fazem uso de persistência de dados em suas aplicações e procuram uma forma mais rápida, simples e eficaz de implementá-la. |
| Spring MVC | Esse framework implementa um grande número de funções, como injeção de dependência, persistência de dados e uma implementação para o padrão MVC para a criação de aplicações WEB. |
| Spring Security | Esse framework centraliza a configuração em um único XML, dispensando configurações do container e tornando a aplicação web um arquivo WAR auto contido. |
| Hibernate | É um framework para realizar o mapeamento objeto relacional (ORM) escrito na linguagem Java, onde seu principal objetivo é diminuir a complexidade envolvida no desenvolvimento de aplicações que necessitam trabalhar com banco de dados relacional, onde ele realiza a intermediação entre o banco de dados e sua aplicação, poupando o desenvolvedor de ter que se preocupar com instruções SQL para recuperar ou persistir os dados do seu software. |
| Banco de Dados PostgreSQL | PostgreSQL é um sistema gerenciador de banco de dados objeto relacional (SGBDOR). |
| Tymeleaf | Frameworks JavaScript, ele adota uma abordagem mais ligada à sintaxe HTML, funcionando como uma espécie de extensão da linguagem. |
| Bootstrap | Framework front-end. |
| Flyway DB | Ferramenta de migração de banco de dados Integra-se com Maven, Ant, Gradle, etc. Funciona com os scripts SQL existentes Compatível com vários BDs, possui Java API. |
| Tomcat | Servlet cointainer que é utilizado como Implementação de Referência para as tecnologias Java Servlets e Java Server Pages. Será utilizado para a realização de deployments e testes da aplicação durante seu desenvolvimento. |

Tabela 2 – Tecnologias

**6.1 PERSISTÊNCIA**

A persistência dos dados manipulados pela aplicação se dará através da utilização do PostgreSQL como SGBD. Como o PostgreSQL é um SGBD relacional, será utilizado o Hibernate, mecanismo de persistência para o mapeamento objeto relacional, que será a responsável por realizar a conversão dos dados relacionais persistidos junto ao PostgreSQL para objetos da linguagem Java, e vice-versa. Com isso, SQLs de manipulação de dados não precisam ser implementadas pelos desenvolvedores, pois são geradas pela própria ferramenta. Essa característica pode ser vista como um facilitador do desenvolvimento, pois, através da camada de abstração fornecida pelo Hibernate, os desenvolvedores lidam apenas com objetos. Contudo, é importante reforçar que a modelagem dos dados da aplicação deverá ser realizada de acordo com as formas normais.

## **VISÃO DE IMPLANTAÇÃO**

Esta secção apresenta como o sistema deve ser mapeado para o hardware.

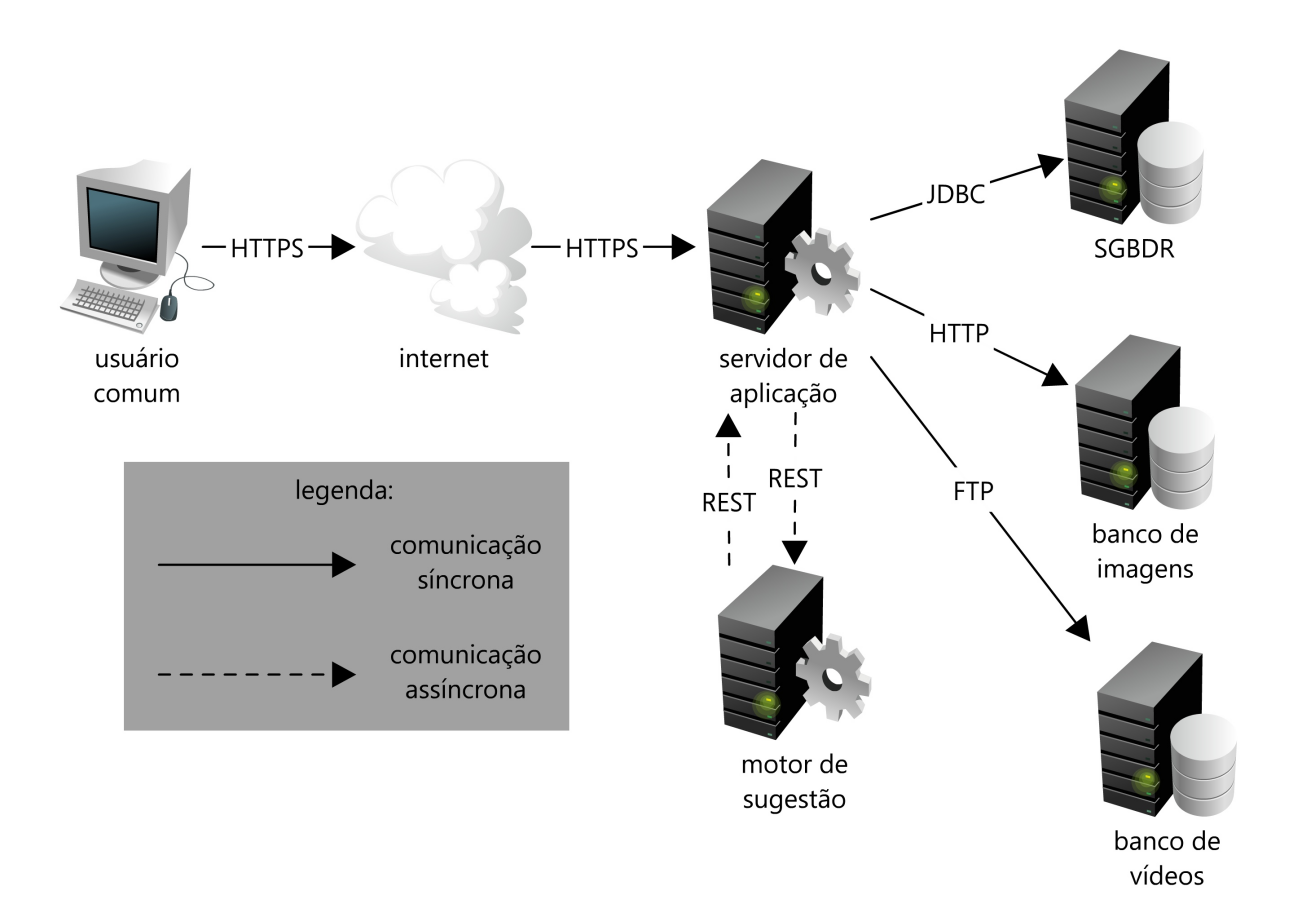


Figura 8 – Visão de Implantação



## **TAMANHO E DESEMPENHO**

Esta secção descreve as características volumétricas e de capacidade de resposta do sistema para definição da arquitetura.

|  |  |
| --- | --- |
| **Ação** | **Sistema** |
| Usuários | O sistema deverá ser capaz de suportar o acesso simultâneo de 15.000 usuários. |

Tabela 3 – Tamanho e Desempenho

## **QUALIDADE**

|  |  |
| --- | --- |
| **Requisito** | **Descrição** |
| Confiabilidade | Tolerância a falhas: O sistema deverá continuar no ar mesmo quando ocorrer falhas. |
| Funcionalidade | Adequabilidade: O sistema deverá possuir formulários de fácil preenchimento de acordo com às necessidades do usuário. |
| Manutenibilidade | Testabilidade: O sistema deverá ser capaz de sofrer modificações e ser validado após a realização das modificações. |
| Usabilidade | Operabilidade: O sistema deverá ser de fácil manuseio para o usuário.  Apreensibilidade: O sistema deverá possuir uma interface amigável para ser de fácil uso para o usuário.  Atratividade: O sistema deverá possuir uma interface atrativa, capaz de prender a atenção do usuário. |
| Portabilidade | Adaptabilidade: O sistema deverá ser adaptável em todos os tipos de dispositivos móveis. |

Tabela 4 – Requisitos de Qualidade