SISLAR

Documento de Arquitetura de Software

Versão 1.0

Histórico da Revisão

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 15/09/2017 | 1.0 | Descrição da arquitetura do sistema | Renato Aguiar |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Índice

1. Introdução 3

1.1 Objetivo 3

1.2 Escopo 3

1.3 Definições, Acrônimos e Abreviações 3

1.4 Referências 3

1.5 Visão Geral 3

2. Representação Arquitetural 3

3. Restrições e Metas Arquiteturais 3

4. Visão de Casos de Uso 3

5. Visão Lógica 3

5.1 Visão Geral 3

6. Visão da Implementação 3

7. Visualização da Implantação 3

8. Tamanho e Desempenho 3

9. Qualidade 3

Documento de Arquitetura de Software

# Introdução

Através deste documento serão abordados os aspectos da Arquitetura do sistema SISLAR, mostrando seu objetivo, qual o escopo de abrangência, além de mostrar a representação arquitetural do sistema, junto com casos de usos, tipos de restrições, dentre outros.

## Objetivo

Este documento fornece uma visão arquitetural abrangente do sistema, usando diversas visões de arquitetura para representar diferentes aspectos do sistema. Ele pretende capturar e transmitir as decisões arquiteturas significativas que foram tomadas em relação ao sistema.

## Escopo

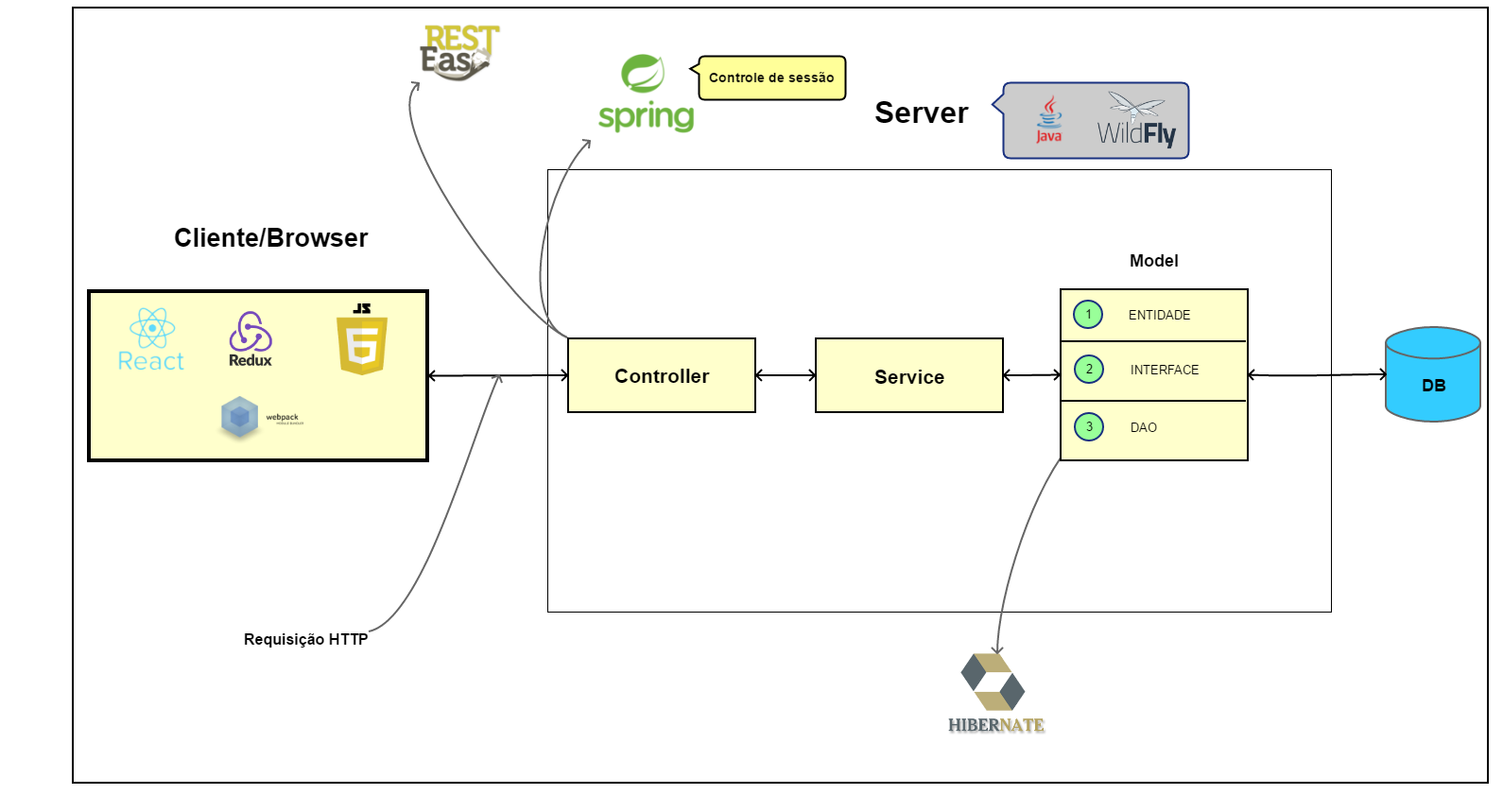
Este documento tem como escopo apresentar uma visão/representação da arquitetura do sistema.

## Visão Geral

Em uma visão geral esse documento vai fornecer informações arquiteturais do sistema, quais são as restrições e metas a serem alcançadas, exemplificação de casos de usos, qual a visão lógica para o sistema.

Abordando pontos como qualidade, visão e implementação.

# Representação Arquitetural



# Restrições e Metas Arquiteturais

|  |  |
| --- | --- |
| Linguagens | Java e React |
| Servidor de aplicações | Wildfly 10.0 |
| Segurança | Spring Security |
| Persistência | Hibernate e JPA |

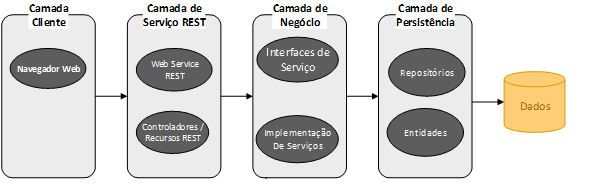
# Visão de Casos de Uso

* Efetuar Login;
* Cadastrar Funcionário;
* Editar Funcionário;
* Apagar Funcionário;
* Escala de Funcionários;
* Check-list enfermeira;
* Anexar documentos;
* Cadastrar Idoso;
* Editar Idoso;
* Apagar Idoso;
* Adicionar Relatório Diário Idoso;
* Editar Relatório Diário Idoso;
* Consultar Relatório Diário Idoso;

# Visão Lógica

## As principais classes do sistema serão aquelas que irão prover a comunicação entre o cliente e servidor, através de filters e controller, o cliente enviará a requisição para o servidor, o filter avaliará se é um usuário e dado válido, sendo aprovado a requisição procede e chega no serviço rest, que neste caso será um conjunto de controllers, a função chamada no controller será responsável por chamar o servisse correto para executar aquela ação, caminhando assim até a camada de persistência e persistindo ou recuperando dados.

## Visão Geral



# Visão da Implementação

O software será dividido em duas camadas, camada de usuário/funcionário e camada de cliente.

* Camada de usuário terá funcionalidades como Efetuar Login, Cadastrar Funcionário, Editar Funcionário, Apagar Funcionário, Escala de Funcionários, Check-list enfermeira, Anexar documentos e entre outras implícitas nessas já citadas.
* Camada de cliente terá funcionalidade como Cadastrar Idoso, Editar Idoso, Apagar Idoso, Adicionar Relatório Diário Idoso, Editar Relatório Diário Idoso. Consultar Relatório Diário Idoso e entre outras implícitas nessas já citadas.

# Visualização da Implantação

A aplicação usará S.O ubuntu 16.04, servidor wildfly hospedado pela digitalOcan com respectivas configurações, 512mb de memória ram, 1 VCPU, 20gb SSD e 1TB de transferência máxima. Esquema de cliente servidor.

# Tamanho e Desempenho

Volume:

1. Número estimado de usuários: Inicialmente 25, podendo crescer exponencialmente;
2. Número de acesso diários: Aproximadamente 400 logins efetuados para 25 usuários;
3. Tempo de sessão de um usuário: 30 minutos;

Performance:

1. Disponibilidade do sistema: O sistema deve estar disponível 24x7;
2. O tempo de resposta das requisições não devem ultrapassar o tempo definido por cada uma no documento de requisitos;

# Qualidade

* Escalabilidade: O sistema estará preparado para a escalabilidade de acessos que possam crescer com o tempo e com o aumento de usuários acessando;
* Disponibilidade: O sistema ficará disponível 24x7, com exceções de pequenas paradas para correções ou upgrades;
* Segurança: Usuário só acessará o sistema se estiver autenticado, o controle de sessão será feito pelo Spring security;
* Confiabilidade: Os dados serão protegidos e também possuirá backups para o caso de eventual perda de dados.