Documento de Especificação de Arquitetura

Sistema: eSaudeBucal

Documento desenvolvido no contexto abaixo:

Disciplina: Arquitetura de Software

Curso: Engenharia de Software

INF - Instituto de Informática

UFG - Universidade Federal de Goiás

1º semestre de 2017

Profª. Fabio Lucena

Grupo: Pablo Henrique Reis Carvalho
Iago Bruno Oliveira Miranda Almeida
Lucas Araujo Da Serra Campos
Bruno Vieira Andrade
Cauã Martins Pessoa
Murilo Eduardo Alves Vaz

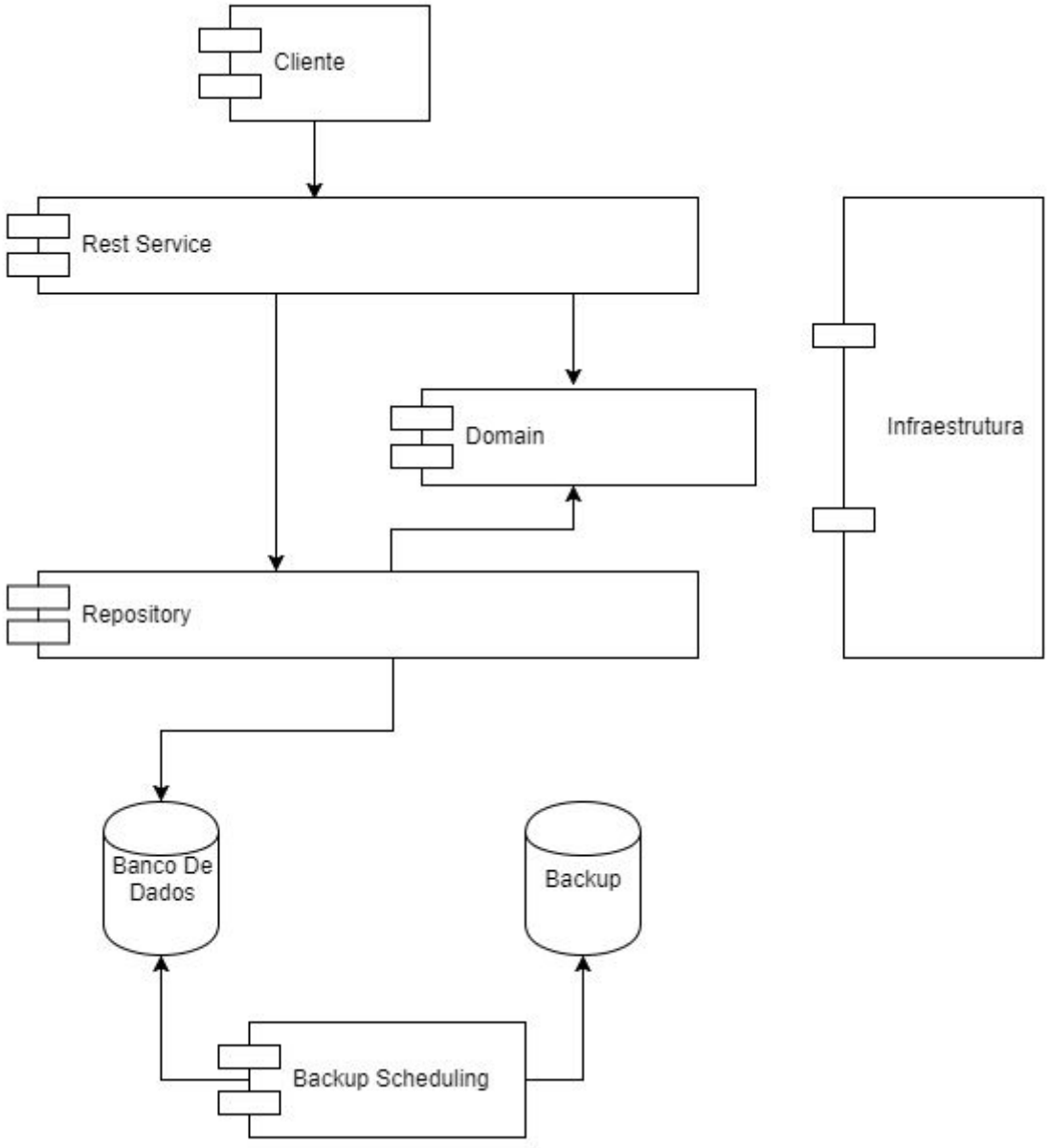
Goiânia-Go, 26 de junho de 2017

1 Intrudução

Esta documentação tem como objetivo detalhar a arquitetura do projeto **eSaudeBucal**, detalhando os requisitos não funcionais, componentes e tecnologias empregadas no escopo do projeto.

2 Visão Geral

O modelo arquitetural que será adotado no projeto é embasado no modelo de cliente servidor, onde a camada do cliente, dependerá do uso dos recursos do navegador Google Chrome (preferencialmente), este por sua vez, irá permitir que o mesmo seja portado para qualquer plataforma que dê suporte a está aplicação ou similares.



1. Arquitetura empregada no Software Vou the Van

3 Requisitos não-funcionais

3.1 Desempenho	3.1.1 As páginas serão ***minificadas,*** reduzindo seu conteúdo e tempo de tráfego pela rede, tornando assim as páginas mais leves, será realizado através do uso do plugin para Maven **YUI Compressor**, configurado através do arquivo **pom.xml**. 3.1.2 Informações de acesso recorrente serão armazenadas em storages do navegador, diminuindo a requisição de informações ao servidor, podendo ser trabalhados através de **services workers**. 3.1.3 Os scripts serão disponibilizados como CDN para diminuir o tráfego, e não realizar download de arquivos desnecessários 3.1.4 Parte do processamento será realizado no cliente utilizando de interfaces ricas com componentes especializados em UX e frameworks JS.
3.2 Portabilidade	3.2.1 Camada visual será desenvolvida de maneira a se adaptar ao tamanho da tela do dispositivo, seja ele tablet, desktop ou smartphone, visando atingir os requisitos de uma **PWA** (Progressive Web Application). 3.2.2 A aplicação será multiplataforma, tendo apenas como dependência a instalação do navegador Google Chrome, que por sua vez, se encontra disponível na maioria das plataformas modernas
3.3 Interoperabilidade	3.3.1 O sistema será desenvolvido em Java, sendo esta uma

	plataforma de desenvolvimento portável entre sistema operacionais que suportam sua máquina virtual, tornando simples a migração de servidores de aplicação; 3.3.2 A comunicação com o banco de dados será realizada através da API de persistência **Java JPA**. Ao implementar esta tecnologia, torna-se fácil portar o banco de dados para outros serviços de armazenamento, pois o acesso aos dados será realizado através desta interface heterogênea, permitindo inclusive, uma migração para outro serviço de **SGBD**.

4 Mecanismos Arquiteturais

Mecanismos de Análise	Mecanismo de Design	Mecanismo de Implementação
Persistência	Banco de Dados Relacional e Não Relacional	**PostgreSQL** e **MongoDB**
Back-End	Camada Lógica responsável pelo implementação do sistema	Java, utilizando como Frameworks Spring, JPA e CDI
Camada de persistência	Classes responsáveis por abstrair os modelos relacionais e realizar a persistência e a comunicação com o banco de dados	JPA utilizando **Hibernate** como provider, implementando o padrão **DAO** (Data Access Object)
Camada Visual	Camada de cliente responsável por estabelecer uma interface de comunicação entre o	Html, Css, JavaScript com a biblioteca de componentes **ReactJs** e a biblioteca **JQuery**

	sistema e o usuário final	
Camada de Teste	Responsável pela validação e qualidade dos artefatos produzidos, com base nos requisitos especificados.	**JUnit** e **Selenium**
Integração contínua	Responsável pela realização dos testes, avaliar a qualidade do código e homologar o deploy da aplicação	**Jenkins** e **SonarQube**
Servidor de Aplicação	Ambiente no qual a aplicação será executada e distribuída	Container **JBoss** em um servidor **Amazon** Linux, **Docker** e **Kubernetes**

5 Fundamentação

O modelo arquitetural a ser empregado no sistema será uma junção entre dois modelos arquiteturais sendo implementados em uma aplicação monolítica e aplicados de acordo com a necessidade do requisito a ser implementado. O modelo de desenvolvimento orientado ao domínio **DDD** (Domain Driven Design) e o **MVC** (Model View Controller), com o intuito de diminuir o acoplamento e responsabilidades de cada camada, facilitando assim o processo de manutenção e desenvolvimento de funcionalidades específicas. Por se tratar de uma aplicação de muitos um alto índice de disponibilidade e requisições, com foco em atender ao público nacional, esse modelo misto seria o mais viável, tendo em vista que a manutenção com baixo grau de acoplamento e com uma boa coesão, facilitaria o processo, tanto para manutenções corretivas, quanto evolutivas, uma vez que a correção de uma camada ou módulo pode ser efetuada independente de seu todo, assim como a implementação de uma nova funcionalidade.

A linguagem adotada será o **Java**, por ser uma linguagem de código aberto e com ótimo suporte da comunidade, além de possuir uma vasta gama de frameworks disponíveis para uso, também de código aberto.

Para o front-end, será adotado o HTML5, com CSS e JavaScript fazendo uso da biblioteca **ReactJs** e **JQuery**, que possibilita uma melhor abordagem referente a renderização de contextos específicos, atualização de estados e elaboração de componentes visuais.

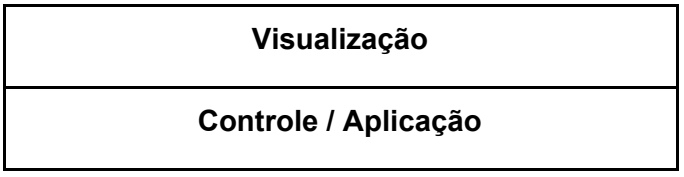
Tendo como base a linguagem Java, os frameworks adotados para o desenvolvimento desta aplicação foram: **JPA** para persistência, utilizando como ***provider*** a biblioteca ***Hibernate***, devido a extensa documentação e maior adoção da comunidade. Para camada visual, foi adotado JavaScript com o uso de **ReactJs**, por ter uma melhor resposta de interação para mudanças de contexto, ideal para o desenvolvimento de componentes para chats e **service workers**, assim como o desenvolvimento independente do back-end.

O back-end será implementado conforme o framework **Spring MVC,** possibilitando a construção dos serviços e controles responsáveis pela comunicação entre a camada de **visualização** e a camada de **persistência**.

O controle de permissão, será implementado através do framework **Spring Security**, lidando com o acesso às requisições e autenticação de usuários, assim como o controle de restrição para os perfis de usuários (**Paciente** e **Corpo Clínico**).

6 Definição Arquitetural

Os sistema conforme especificado será dividido em uma arquitetura mista que pode ser representada pelo **MVC padrão** e o um **MVC integrado** ao **DDD**, visando diminuir o grau de acoplamento e modelando a arquitetura de acordo com a complexidade do problema, sendo separadas em **camada de persistência**, **camada de controle ou aplicação, camada de visualização, camada de domínio quando for necessário e camada de infraestrutura**. As camadas devem ser imaginadas como estruturas independentes, em que cada uma tem uma responsabilidade e quando necessário, uma camada ou módulo poderá acessar uma outra camada, sem interferir no seu funcionamento normal. Dentre essa divisão, somente a camada de domínio deve ser isolada e não depender de nenhuma camada adicional. A estrutura hierárquica pode ser descrita conforme a figura abaixo.



Infraestrutura
Domínio
Persistência

6.1 Camada de persistência

Camada responsável por efetuar a comunicação com o banco e mapear as estruturas das tabelas presentes no banco de dados em modelos relacionais de objetos. As operações de inclusão, consulta, atualização e deleção de dados, são realizadas através do módulo DAO (Data Access Object), utilizando para tal o framework de persistência JPA.

Os modelos presente nessa camada, são abstrações das estruturas de dados presente no banco de dados, facilitando a manipulação deste através da aplicação e a alteração da estrutura de dados, tornando flexível a mudanças futuras.

O **DAO** será implementado em duas categorias, o referente ao banco de dados relacional e ao banco de dados não relacional (**NoSQL**).

6.2 Camada de controle ou aplicação

Camada de software com atribuições referente ao controle de fluxo da aplicação implementação de operações especificadas para front service e conversão de dados para acesso às demais camadas, integrando a camada de visualização com o camada de domínio se necessário ou a camada de dados do sistema. Nesta camada se encontram as chamadas de operações que visam atender os requisitos funcionais e fornecer os serviços acessados pela camada de visualização ou cliente.

Para cada página do sistema (Camada de Visualização) será implementado um controle, que será responsável por implementar as funcionalidades da respectiva página. Cada controle terá o mesmo nome da sua camada visual, acrescida de um sufixo **Controller**.

6.3 Camada de Visualização

Camada que provê a interação entre o usuário e o sistema, responsável por realizar as chamadas de execução da camada de controle e exibir informações e respostas vindas desta. A camada de controle no sistema ESaudeBucal será composta pelo conjunto de páginas responsável por compor a interface de comunicação com o usuário final. Por definição do modelo arquitetural implementado, cada página deve ser controlada por uma classe de controle. Essa mudança veio para facilitar e resolver alguns problemas no desenvolvimento desta camada. Assim surgindo diversos frameworks que facilitam esse desenvolvimento. Como já citado será utilizado o react para essa implementação.

6.4 Camada de Domínio

Camada responsável por implementar o Business da aplicação, nesta camada será implementado os fluxo de dados abstraído das funcionalidades descritas a serem implementadas, mantendo uma linguagem ubíqua com mínimo de acoplamento.

6.5 Infraestrutura

Fornece recursos tecnico que dão suporte as demais camadas da aplicação. Como integrações com outras aplicações ou outros sistemas como é o caso da emissão de nota fiscal da **SEFAZ**, e integração com **Cielo** e **PagSeguro**.