Documento de Arquitetura de Software Bizu

Histórico de Revisão

Data	Versão	Descrição	Autor	Revisor

Índice

Documento de Arquitetura de Software

1. Introdução

Buscando atingir todos interessados e demais envolvidos com este domínio, este documento fornece um conjunto de visões que resolvem os diversos aspectos de desenvolvimento e implantação que serão adotados por esse sistema. Diante disso, decisões e devolutiva decisórias estão retratadas no decorrer deste.

1.1 Objetivo

Este documento busca apresentar a arquitetura que será adotada para atingir e sanar os requisitos funcionais e não funcionais levantados anteriormente para o sistema.

1.2 Escopo

O escopo deste documento é documentar as partes significativas do ponto de vista da arquitetura, como sua divisão em camadas e pacotes.

1.3 Definições, Acrônimos, e Abreviações

SI – Sistema de Informação

BD – Banco de Dados

1.4 Referências

Documento de arquitetura de software Bizu, nele descreve o framework utilizado e os componentes integrantes. Foi baseado sua estrutura, no Documento de arquitetura de software SEFAZ-GO.

1.5 Visão Geral

Este documento está organizado em tópicos relacionados às diferentes visões arquiteturais.

2. Arquitetura da Aplicação

Esta seção descreve de forma mais clara o sistema e os detalhes da arquitetura que será avaliada, juntamente com o impacto que estar arquitetura terá sob todo o restante do desenvolvimento e produto final.

2.1 Representação da Arquitetura

Os sistemas serão desenvolvidos tendo como base a arquitetura multicamadas aderentes à arquitetura baseada em componentes, utilizando tecnologias respectivas para MODEL, VIEW e CONTROLLER.

2.2 Objetivos e Restrições da Arquitetura

Seguindo a arquitetura baseada em componentes tem como objetivo permitir que os usuários do sistema tenham acesso a uma plataforma que atenda suas expectativas e necessidades, além de cumprir todas as definições de negócio já abordadas.

2.3 Critérios da Avaliação Arquitetural

Os critérios utilizados para a seleção da solução arquitetural foram:

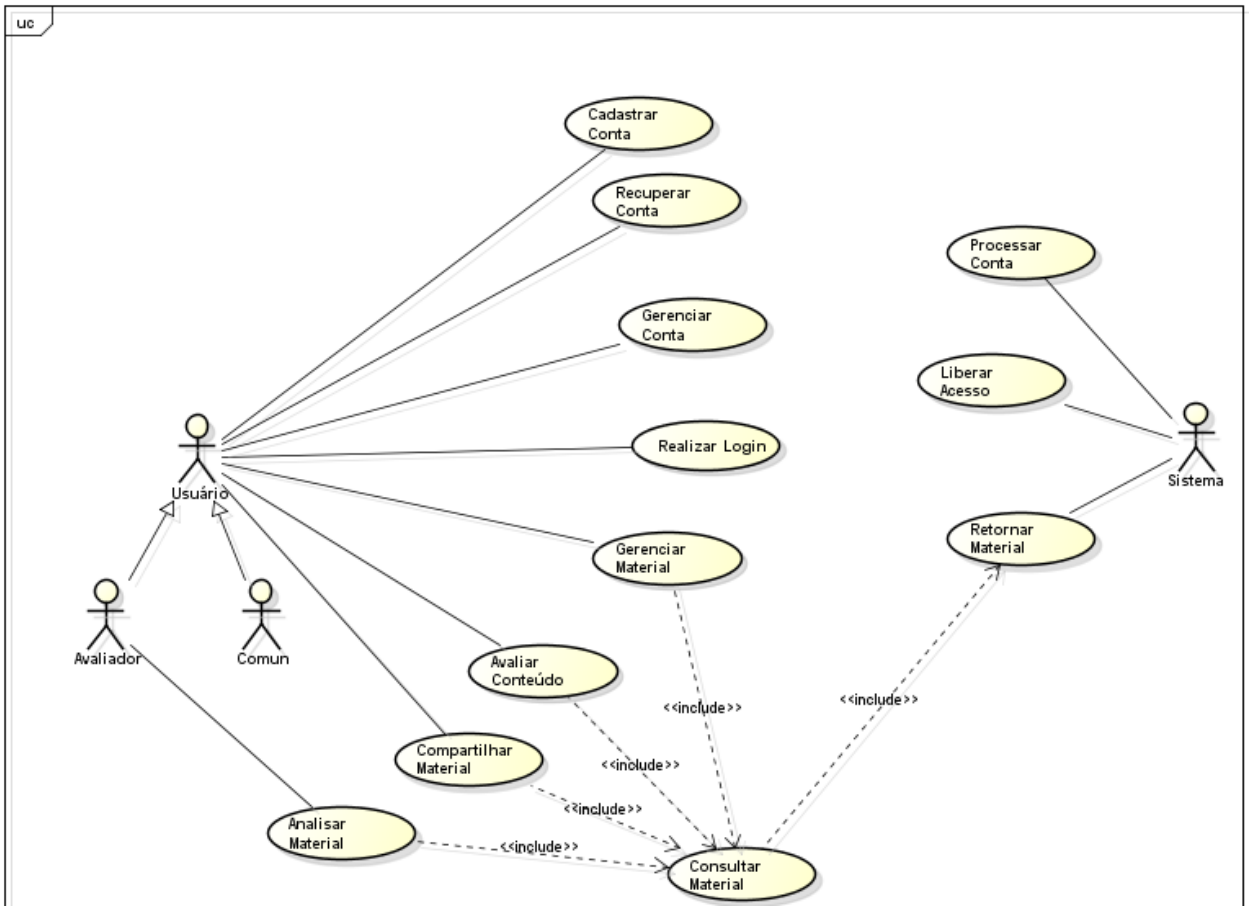
- Manutenibilidade
- Acessibilidade e Usabilidade
- Eficiência
- Disponibilidade

2.4 Arquiteturas Descartadas

Por causa da Equipe de desenvolvimento e a bagagem profissional e técnica, foram avaliadas algumas tecnologias para este projeto, baseado em atualidade e capacitação como atributo para escolha das ferramentas de desenvolvimento.

Utilização de Java: foi descartada pela dificuldade e/ou impossibilidade de trabalhar com frameworks JSF ou PrimeFaces, pois no desenvolvimento não obtemos total controle do HTML gerado, não garantindo Eficiência prometida anteriormente.

3. Visão de Casos de Uso



Definição dos casos de uso:

Casos de Uso relacionados a Conta que o usuário necessita para acessar a plataforma do sistema

- Realizar Login
- Recuperar Conta
- Cadastrar Conta

Casos de Uso relacionados aos materiais que o usuário ira manipular dentro do sistema , possibilitando todas as ações necessárias para sanar as expectativas do usuário quanto a funcionalidade.

- Excluir Material
- Cadastrar Material
- Consultar Material
- Avaliar Material
- Compartilhar Material

4. Metas e Restrições de Arquitetura

Softwares Utilizados

Para execução do sistema será necessária a instalação dos seguintes softwares:

- Servidor PHP 8 ativo
- PgAdmin 4
- Sistema operacional Windows 2012 Server;
- Microsoft Visual Studio 2019
- Domínio DNS registrado e redirecionado

Caso não sejam instalados os softwares descritos acima a aplicação não irá funcionar como esperado ou pode nem ser inicializada.

Disponibilidade

Para garantir uma alta disponibilidade os sistemas devem segmentar suas funcionalidades de forma a aplicar uma distribuição para execução das suas aplicações separadas.

- A aplicação deverá ser mantida em um domínio confiável e com confiabilidade de rede, que promova uma conexão estável e reduzindo o mínimo de perda das solicitações dos pacotes.

Acessibilidade & Usabilidade

- Componentes devem tornar o processo de apreensibiliade e compreensibilidade, mas simples possível
- As Views devem ser responsivas e com estrutura que permita uma melhor compreensão das funcionalidades

Manutenibilidade

- Os componentes devem ser construídos de forma a distribuir o processamento do sistema, e descentralizar os componentes.
- Os componentes dos sistemas produzidos deverão ser intuitivos e manter o padrão definido neste documento
- Manter o padrão de classes e componentes por este documento mencionado

5. Visão Lógica

5.1 Estrutura de Camadas

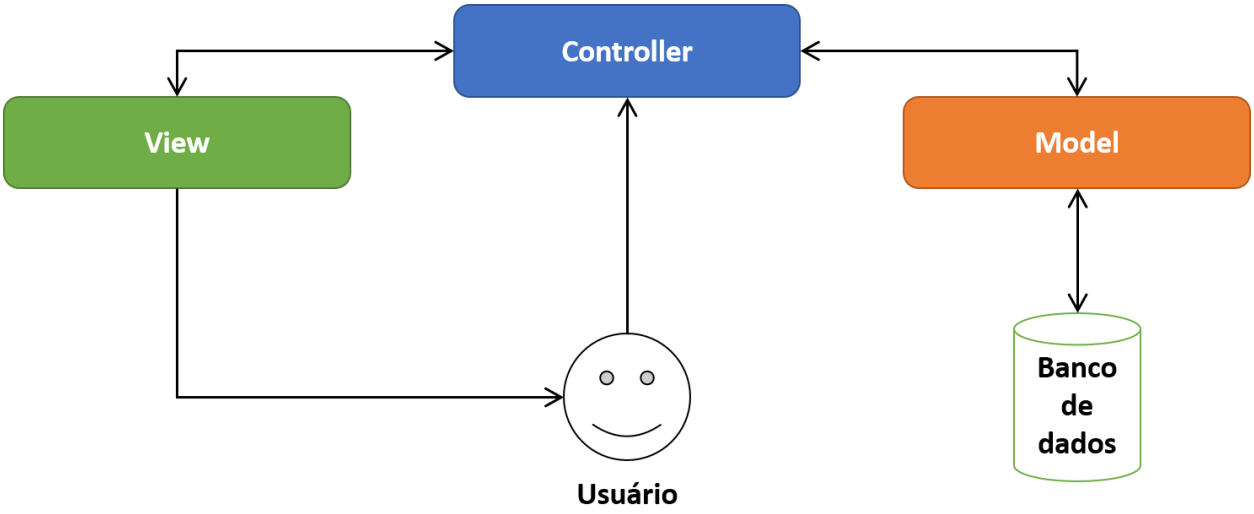
5.1.1 Padrões utilizados

Padrões de Projetos fundamentais na estruturação do framework:

- Observer;
- Facade

5.1.2 Visão das camadas

Demonstração da organização das camadas para aplicações desenvolvidas em acordo com essa arquitetura.



Camada de View

Essa camada conterá todas as interfaces visuais, na qual interagirá diretamente com o usuário do sistema. E estará subdividida em:

- Visual: contém a página Web, e as referencias adequadas para Designer. Utilizando HTML e framework Bootstrap para construção dos componentes e estruturação da view.
- Controle: Contém todos os códigos necessários para comunicação entre a camada de integração ou negocio, com a pagina Web (Ajax).

Camada de Controller

- Controle: Contém todos os códigos necessários para comunicação entre a camada de view e camada Model, utilizando para esta tarefa, Javascript que interage com as páginas HTML e persiste os dados para o BD.

Camada Model

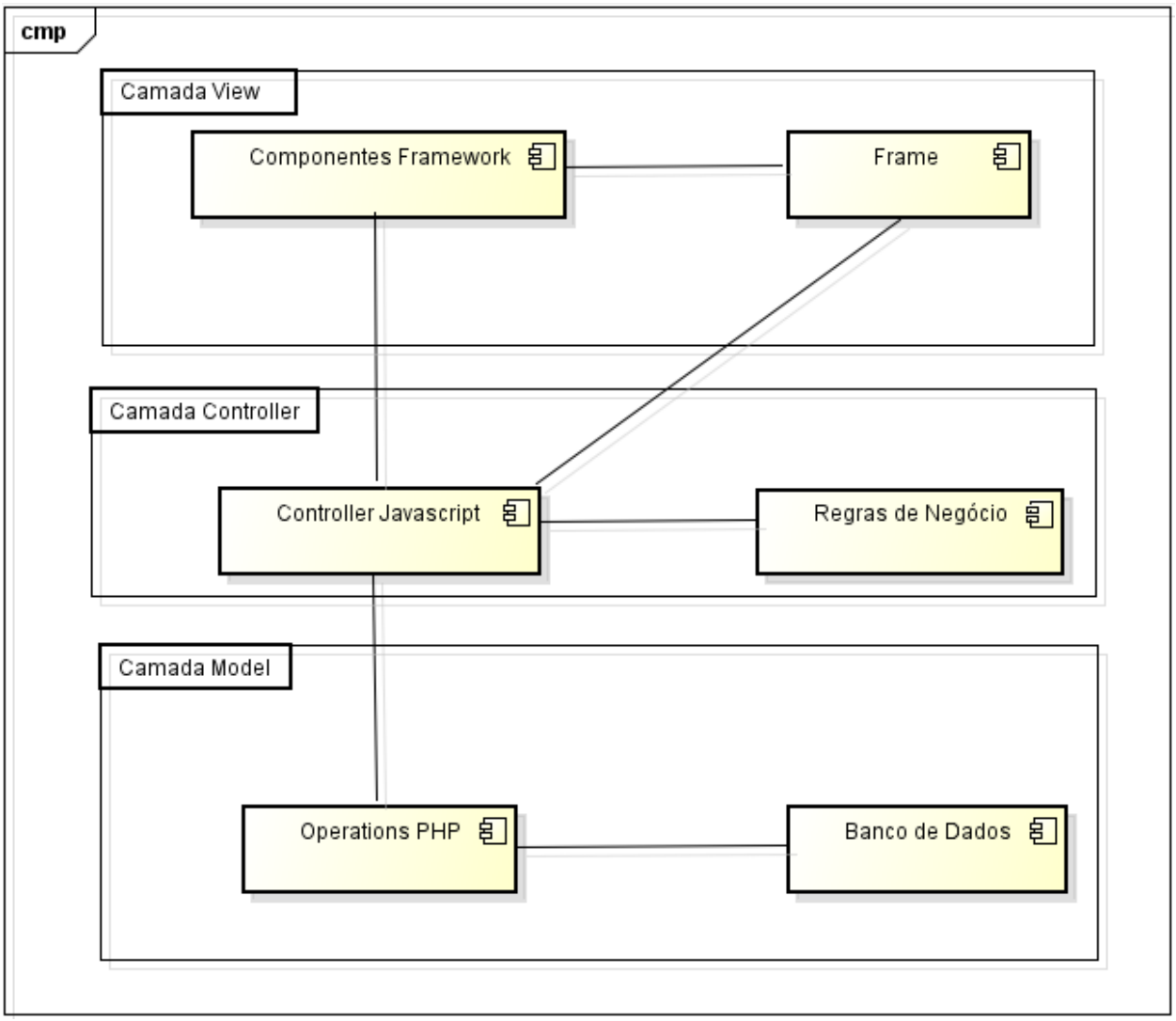
Responsável pela leitura e escrita dos dados, bem como responsável por persistir os dados para

banco de dados SQL, logo será utilizado o PHP 8 para realizar essa manipulação.

5.2 Divisão de componentes

5.2.1 Visão dos componentes

Os componentes estão organizados da seguinte forma:



Camada de View

Componentes Framework – Representa o conjunto de componentes que estrutura o Frame do sistema, contem botões, formulários, menus e outros componentes necessários para cumprir os requisitos não funcionais e funcionais.

Frame – Representa a tela contendo os componentes do sistema.

Camada de Controller

Controller JavaScript – Representa o controle de ações feito pelo sistema, controlando as iterações com os frames e o fluxo de dados para a camada Model.

Regras de Negócio – Responsável por auxiliar no controle do fluxo de dados, regulando os dados segundo as normas definidas anteriormente pelas Regras de Negócio.

Camada de Model

Operations PHP – Representa a ferramenta responsável pela persistência de dados para o Banco de Dados.

Banco de Dados – Responsável pelo gerenciamento dos dados persistidos.

5.2.2 Tecnologias adotadas

Camada de Apresentação: BootsTrap
Camada de Controller: JavaScript
Camada de Model: PHP 8 e PgAdmin 4

5.2.3 Componentes básicos

Para um desenvolvimento mais eficiente, o Bootstrap fornecerá por meio de um template, componentes e frames pré definidos que serão utilizados para estruturação do sistema.

Elaboração e desenvolvimento de uma interface padrão capaz de executar os seguintes itens:

- Nomes de "variáveis" e "cabeçalho" do grid mais significativos;
- Máscaras nos campos "cpf", "endereço" e outros necessários;
- Facilidade para manter "regras" de filtro incluídas pelo usuário;
- Estudo de um grid funcional e customizável, com bastante recursos extras para o usuário, como: pesquisas, filtros, ocultação de colunas, várias opções de paginação, múltipla seleção, duplo click, grande quantidade de registros e resposta rápida.

A interface web criada automaticamente a partir de um contexto em formato XML, terá as seguintes áreas:

• Parâmetros de pesquisa:

- Variável – este controle conterá as colunas que poderão ser usadas como filtros de pesquisa. O usuário poderá criar regra de filtros obrigatórios;
- Operador – este controle conterá operadores lógicos (>, <, =, etc) para compor a regra de pesquisa;

- Nome Material – este controle contém o valor a ser pesquisado, podendo inclusive ser inserido uma lista de valores ou selecionar a partir de uma outra tabela;
- Conector – este controle permite conectar mais de uma regra de pesquisa.

• Regras:

- Esta área contém as regras de pesquisa inseridas pelo usuário;
- Para garantir maior agilidade no processo de pesquisa, o usuário poderá alterar valores dos campos de filtro de consulta, baseado nas regras de negócio.

• Responsabilidades do Frame:

- Permite visualização dos diversos materiais na Tela Principal;
- Permite acessar os menus e selecionar as ações possíveis;
- Permite visualizar as preferências de usuário;

5.3 Estratégia de Reuso

Utilizando componentes predefinidos do Bootstrap, o reuso se torna mais eficiente e pois os componentes já estão construídos e podem ser facilmente alterados novamente para diversas outras utilizações.

6. Visão de Implantação

Os sistemas são distribuídos da seguinte forma:

- Servidores Web

Estrutura Lógica do funcionamento dos hardwares

Compreensibilidade

A aplicação deverá ser organizada de forma a melhorar a capacidade do usuário de aprender e compreender as funcionalidades.

Escalabilidade

A aplicação deverá estar preparada de forma a utilizar múltiplos processamentos, possibilitando a distribuição em máquinas distintas para realizar o processamento.

Manutenibilidade

A aplicação deverá estar nas melhores práticas de implementação para que as manutenções sejam feitas de forma breve, e também se adequando e suportando novos ambientes. Visão de Implementação.

5.1. Visão Geral

Fluxo de comunicação entre os modelos de classes criadas.

Conexão com o Banco de dados

O PHP com suas diversas funcionalidades e controles com o Pegadinha, será responsável pela conexão e manipulação dos dados que serão inseridos, buscados ou excluídos dentro do Banco.

Conexão com o PHP

O JavaScript será responsável por enviar os dados para o PHP, utilizando as ferramentas disponíveis dentro do Ajax e captura dos dados via JSON.

5.3. Estrutura de Componentes do Framework

Componentes utilizados na estruturação das funcionalidades:

Botões – Responsável pelos submit de formulários e gatilho para envio de informações.

Formulário – Responsável por receber as informações inseridas vinculados ao JavaScript por meio das Labels do Input, a fim de capturar as informações.

SideBar – Menu lateral responsável por manter as opções disponíveis para o usuário interagir nas diversas funcionalidades do sistema.

Mavbar – Menu superior responsável por fornecer dados importantes e preferencias de usuário, bem como estilizar as páginas com ícone do Bizu e uma caixa de pesquisa.

7. Visão de Implementação

Será utilizado o Visual Studio Code para desenvolvimento, organizado em pastas para cada tipo da camada MVC.

View

HTML

Frames.html

Cotroller

JavaScript

Funcionalidades.js

Model

PHP

Operaçao.php

Baseado na linguagem HTML, o contexto é configurado através de tag's. São elas:

- <head> - utilizada para abranger os includes;
- <body> - corpo do frame html utilizando bootstrap
- <forms> - formulário utilizando as classes Booststrap;
- <button> - botões utilizando as classes Booststrap;
- <nav> - barra de navegação utilizando as classes Booststrap;
- <input> - campo de inserção de dados, utilizando classes Booststrap;
- <label> - Descrição de um campo ou funcionalidade, utilizando classes Booststrap;
- <table> - tabela que será utilizada a base do Booststrap;

- OBS: Todos os inputs podem ter um atributo chamado "mascara". As possíveis máscaras são: "data", "cpf", "endereço";

6.1. Visão Geral

Fluxo de comunicação entre os modelos de classes criadas.

Conexão com o Banco de dados

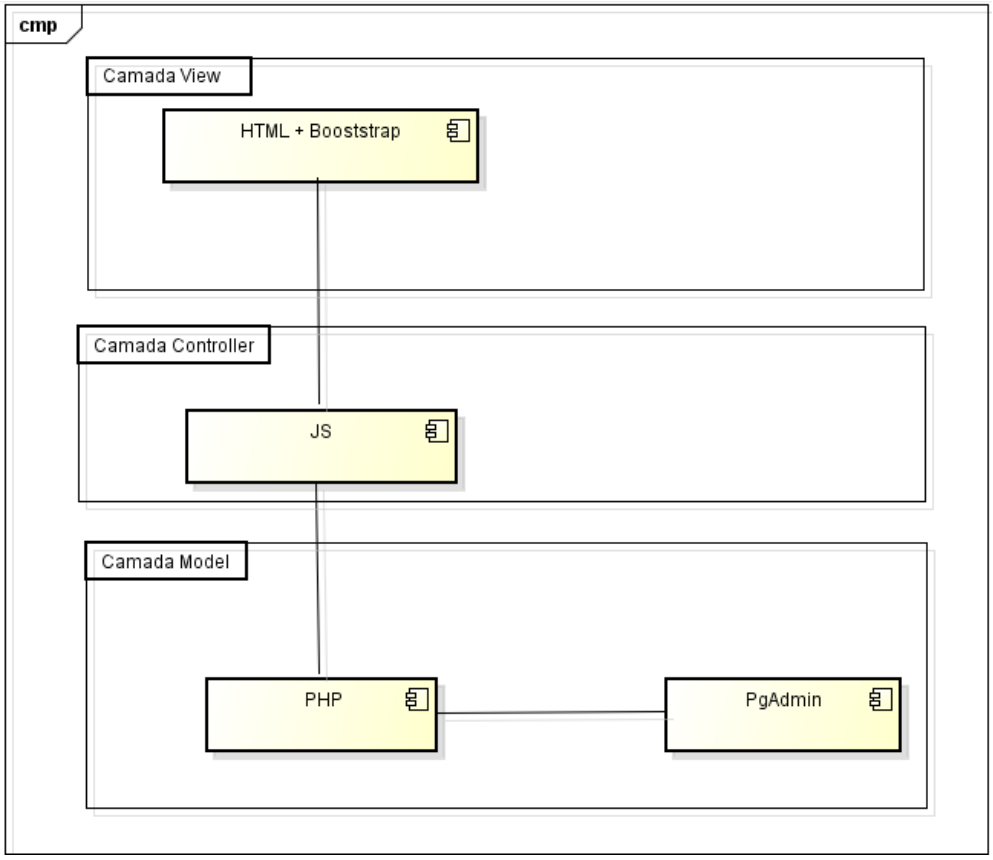
O PHP com suas diversas funcionalidades e controles com o Pegadinha, será responsável pela conexão e manipulação dos dados que serão inseridos, buscados ou excluídos dentro do Banco.

Conexão com o PHP

O JavaScript será responsável por enviar os dados para o PHP, utilizando as ferramentas disponíveis dentro do Ajax e captura dos dados via JSON.

6.2. Implementação das Camadas

De forma mais clara e objetiva a implementação do sistema, baseado no estilo arquitetural utilizado em camadas, será realizado segundo o seguinte diagrama:



Todas a parte de visualização para o usuário na cama view será construída a partir do HTML juntamente com o framework adotado, além de ter como suporte de controle o Javascript, controlando as requisições e a dinâmica das informações e componentes.

Por fim, a manipulação das informações e dados serão realizados pelo PHP, através de scripts com cada tipo de consulta ou operação a ser realizada com os dados dentro do Banco.

8. Visão de Dados

Os dados serão organizados segundo as estruturas abaixo:

- Tabela – Os dos serão organizados nessa estrutura quando consultados dentro dos requisitos solicitados
- Lista – alguns campos dentro do HTML e mesmo dentro do Javascript utilizaram as Listas como forma de organizar e ordenar ou ate mesmo controlar o que será exibido.

Os tipos dos dados, ou seja, as variáveis utilizadas nos campos e informações:

- Int – Valores inteiros
- String – Conjunto de caracteres

- Date – Tipo da variaevel Data, para registrar data ou gerar horário atual dentro do JS.
- Json – Retorno obtido com as consultas o Ajax, dentro do JS
- Double – Variavel de ponto flutuante (valores decimais)

9. Tamanho e Performance

A arquitetura escolhida e as visões fornecidas devem garantir que um conjunto de 200.000 mil usuários possam utilizar o SI com o nível de qualidade garantido, conseguindo suprir os requisitos funcionais em performance eficaz, além de manter os atributos de qualidade já definido nos documentos de requisitos, acordado pelos stakeholders.

Como meta, o tempo de resposta das requisições não será superior a 2 s, sendo ainda necessário que em caso de queda do sistema/servidor, o mesmo tenha restabelecimento imediato.

10. Qualidade

O nível de qualidade que será aferido do SI, será baseado nos atributos de qualidade definidos no documento de requisitos, bem como nos atributos de arquitetura definidos anteriormente nesse documento. Ficando assim como responsabilidade da Etapa de deste, a capacidade de inferir essas informações do produto final de software , entretanto sobre o documento de arquitetura e seu processo devem aplicar o método de verificação segundo a ISO 12207 para identificações de anomalias dentro desta etapa.