School Drive

Projeto Arquitetural

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Objetivo

A arquitetura utilizada no sistema baseiam-se na arquitetura em camadas MVC.

Objetivos:

* Separação de componentes por responsabilidades em comum.
* Utilizado para estruturar e definir melhor as funcionalidades de um sistema.
* Separar o código de acesso aos dados do código de negócio e interface com o usuário (Modularidade)

# Metas Arquiteturais e Filosofia

[Descreve a filosofia da arquitetura. Identifica assuntos que vão levar a filosofia, como: O sistema será

conduzido por interesses de distribuição complexa, adaptando-se aos sistemas legados, ou questões de

execução? Ele precisa ser robusto para manutenções de longo prazo?

Formula um conjunto de objetivos que a arquitetura necessita satisfazer em sua estrutura e comportamento.

Identifica questões críticas que devem ser endereçadas pela arquitetura, como: Há dependências de hardware

que devem ser isoladas do resto do sistema? O sistema precisa funcionar eficientemente sob condições

incomuns?]

# Premissas e Dependências

[Lista as hipóteses e dependências que levam as decisões arquiteturais. Isto pode incluir áreas criticas ou

sensíveis, dependências de interfaces de sistemas legados, a habilidade e experiência da equipe, a

disponibilidade de recursos, e assim por diante]

# Requisitos Críticos da Arquitetura

Segue link da documentação do Spring Framework e seus demais módulos utilizados

* http://projects.spring.io/spring-framework/
* <http://projects.spring.io/spring-data/>
* <http://projects.spring.io/spring-security/>

# Decisões, Restrições e Justificativas

# Mecanismos Arquiteturais

A arquitetura desse projeto é baseada nos fundamento do Spring Framework com IoC e injeção de dependências, logo para cada camada existe um jeito diferente de se fazer nessa seção ira descrever cada para se utilizar o esse framework.

Para esse projeto um padrão de codificação será utilizado nas interfaces em uma determinada camada do software declaramos a interface no pacote principal e definimos um pacote filho chamado "impl" onde serão armazenadas as classes de implementação.

**DAO**: Encontram-se as classes que irão fazer a conexão com os bancos de dados utilizados pela aplicação, bem como definir os comandos de consulta, inclusão, remoção e alteração desses dados.

**Controller**: Encontram-se as classes que irão trabalhar as requisições do usuário na interface, e a parte de lógica de negócios.

**View**: Contém as classes que farão a interface com o usuário com as funcionalidades do sistema.

**Entidade**: Onde são codificadas as classes que farão parte do domínio do projeto.

**Exception**: Onde são codificadas as classes responsáveis pelo tratamento de condições anormais ocorridas na execução compilação do sistema.

**Util**: Encontram-se as classes que serão compartilhadas por projeto objetivando a Reusabilidade.

**Message:** Encontra-se as classes que serão utilizadas para geração de mensagem.

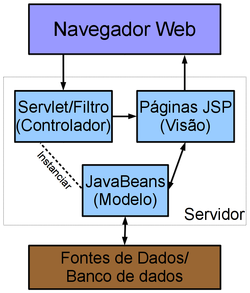
Para pasta WEB-INF terá o arquivo de necessário para a aplicação de visualização e de configuração do Spring

Pasta:

* Resources: pasta contendo todos os arquivo separados em pasta CSS,JS, Imagem do sistema.
* Pages: pasta contendo todos os arquivo para view do sistema separada por função.
* Arquivo do Spring: Responsável por configurar o contêiner do Spring
  + Spring-mvc.xml : Tem a função de configurar o a parte de view do sistema como a pasta que vai procurar as views que serão utilizada .
  + Spring-data.xml: Tem a função de configurar a conexão com o banco de dados como seja por JDBC ou JPA.
  + Spring-security.xml: Tem a função de configurar a permissão de que tem acesso ao sistema e realizar a autenticação do usuário com seu papel(ROLE) no sistema.
  + Spring-context.xml: Responsável por iniciar as configurações supracitadas além de ser esse aquivo que é referenciado pelo web.xml.
* Web.xml: Responsável por carregar e configurar a aplicação do contêiner do Spring

Sobre a parte de view nesse projeto será utilizado JavaServer Pages (JSP) é uma tecnologia que ajuda os desenvolvedores de software a criarem páginas web geradas dinamicamente baseadas em HTML, XML ou outros tipos de documentos.

Para nosso caso o Spring framework especificamente o modulo SpringMVC quem faz essa parte do servlet/filtro é o dispatcher/filtroSecurity configurado do Spring-mvc e o Spring-security que por sua vez é inicializado no web.xml.



# Principais Abstrações

[Lista e descreve brevemente as abstrações chave do sistema. Pode ser uma lista relativamente curta dos

conceitos críticos que definem o sistema. As abstrações chave geralmente irão traduzir para classes de analise

inicial e padrões importantes.]

# Camadas do Framework da Arquitetura

O Spring Framework é composto por recursos organizados em cerca de 20 módulos. Estes módulos são agrupados em Núcleo Container, Data Access / Integração, Web, AOP (Aspect Oriented Programming), Instrumentação, Mensagens e teste, como mostrado no diagrama a seguir.



Os blocos de construção descritos anteriormente tornar Primavera uma escolha lógica em muitos cenários, a partir de applets para aplicações empresariais de pleno direito que usam a funcionalidade de gerenciamento de transações e framework web integração do Spring.

Nesse projeto serão usados:

* Os **Spring-core** e **spring-beans** módulos fornecer as peças fundamentais dos blocos a cima, incluindo as características de IoC e injeção dependência. O BeanFactory é uma aplicação sofisticada do padrão de factory. Ele elimina a necessidade de singleton programáticas e permite dissociar a configuração e especificação de dependências de sua lógica do programa real.
* A **Spring-SpEL** módulo fornece uma poderosa linguagem de expressão para consultar e manipular um objeto gráfico em tempo de execução. É uma extensão da linguagem de expressão unificada (unified EL), conforme especificado na especificação JSP 2.1.
* O **Spring-jdbc** módulo fornece um JDBC camada abstrata que elimina a necessidade de fazer a codificação JDBC tedioso e análise de códigos de erro específico de banco de dados de fornecedores.
* O **Spring-ORM** módulo fornece camadas de integração para os populares de mapeamento objeto-relacional APIs, incluindo JPA , JDO e Hibernate e o próprio Spring Data JPA . Usando a primavera-ORM módulo que você pode usar todos esses enquadramentos O / R de mapeamento em combinação com todas as outras características ofertas Spring, como o recurso de gerenciamento de transação declarativa simples mencionado anteriormente.
* A **Spring-web** módulo oferece recursos de integração básicas orientadas para a web como a funcionalidade de upload de arquivo multipart e a inicialização do contêiner IoC usando ouvintes Servlet e um contexto de aplicação orientada a web. Ele também contém as peças relacionadas com a web de apoio a comunicação remota do Spring.
* A **Spring-webMVC** módulo (também conhecido como o Web-Servlet módulo) contém model-view-controller (do Spring MVC ) de implementação para aplicações web. Framework MVC do Spring fornece uma separação clara entre o código e modelos de formulários web domínio e integra-se com todos os outros recursos do Spring Framework.

# Visões Arquiteturais

[Descreve as visões arquiteturais que você utilizará para descrever a arquitetura do software. Ilustra as

diferentes perspectivas que você disponibilizará para revisar e documentos de decisões arquiteturais.]

Visões recomendadas

• Lógica: Descreve a estrutura e comportamento das porções arquiteturalmente

significantes do sistema. Isto pode incluir pacotes de estrutura, interfaces críticas,

importantes classes e subsistemas, e as relações entre estes elementos. Pode

também incluir visões físicas e lógicas dos dados persistente, se a persistência for

usada no sistema. Também é um subsistema do design

• Operacional: Descreve os nós físicos do sistema e os processos, processos de

execução, e componentes que rodam nestes nós físicos. Esta visão não é necessária

se o sistema roda em um único processo e processo de execução.

• Caso de uso: Uma lista ou diagrama os casos de uso que contém requisitos

arquiteturalmente significantes