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

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

[Always address Sections 2 through 6 of this template. Other sections are recommended, depending on the amount of novel architecture, the amount of expected maintenance, the skills of the development team, and the importance of other architectural concerns.]

# Metas Arquiteturais e Filosofia

[Describe the philosophy of the architecture. Identify issues that will drive the philosophy, such as: Will the system be driven by complex deployment concerns, adapting to legacy systems, or performance issues? Does it need to be robust for long-term maintenance?

Formulate a set of goals that the architecture needs to meet in its structure and behavior. Identify critical issues that must be addressed by the architecture, such as: Are there hardware dependencies that should be isolated from the rest of the system? Does the system need to function efficiently under unusual conditions?]

# Premissas e Dependências

[List the assumptions and dependencies that drive architectural decisions. This could include sensitive or critical areas, dependencies on legacy interfaces, the skill and experience of the team, the availability of important resources, and so forth]

# Requisitos Críticos da Arquitetura

[Insert a reference or link to the requirements that must be implemented to realize the architecture.]

# Decisões, Restrições e Justificativas

[List the decisions that have been made regarding architectural approaches and the constraints being placed on the way that the developers build the system. These will serve as guidelines for defining architecturally significant parts of the system. Justify each decision or constraint so that developers understand the importance of building the system according to the context created by those decisions and constraints. This may include a list of DOs and DON’Ts to guide the developers in building the system.]

* Decision or constraint and justification
* Decision or constraint and justification

# Mecanismos Arquiteturais

[List the architectural mechanisms and describe the current state of each one. Initially, each mechanism may be only name and a brief description. They will evolve until the mechanism is a collaboration or pattern that can be directly applied to some aspect of the design.]

## Mecanismo Arquitetural 1

[Describe the purpose, attributes, and function of the architectural mechanism.]

## Mecanismo Arquitetural 2

[Describe the purpose, attributes, and function of the architectural mechanism.]

# Principais Abstrações

[List and briefly describe the key abstractions of the system. This should be a relatively short list of the critical concepts that define the system. The key abstractions will usually translate to the initial analysis classes and important patterns.]

# 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

[Describe the architectural views that you will use to describe the software architecture. This illustrates the different perspectives that you will make available to review and to document architectural decisions.]

## Recommended views

* **Logical:** Describes the structure and behavior of architecturally significant portions of the system. This might include the package structure, critical interfaces, important classes and subsystems, and the relationships between these elements. It also includes physical and logical views of persistent data, if persistence will be built into the system. This is a documented subset of the design.
* **Operational:** Describes the physical nodes of the system and the processes, threads, and components that run on those physical nodes. This view isn’t necessary if the system runs in a single process and thread.
* **Use case:** A list or diagram of the use cases that contain architecturally significant requirements.