UNIVERSIDADE
FEDERAL DO CEARÁ

Documento de Arquitetura de *Software*
Interactive *Shelf* (I*Shelf*)

Histórico de Revisões

Data	Versão	Descrição	Autor
22/05/2017	0.1	Elaboração do Documento	Alysson Gomes
04/06/2017	0.2	Edição da visão de pacotes	Alysson Gomes
01/11/2017	0.3	Edição da visão de implementação	Alysson Gomes

1. Introdução

1.1. Finalidade

Este documento apresenta a arquitetura proposta para a solução web de apoio ao desenvolvimento de aplicações na plataforma de computação de alto desempenho HPC-*Shelf*. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.

1.2. Escopo

Este documento define a arquitetura da solução web de apoio ao desenvolvimento de aplicações na plataforma HPC-*Shelf*.

1.3. Definições, Acrônimos e Abreviações

High Performance Computing	HPC
Requisito Arquitetural	RA
Unified Modeling Language	UML
Web services	WS

1.4. Visão Geral

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

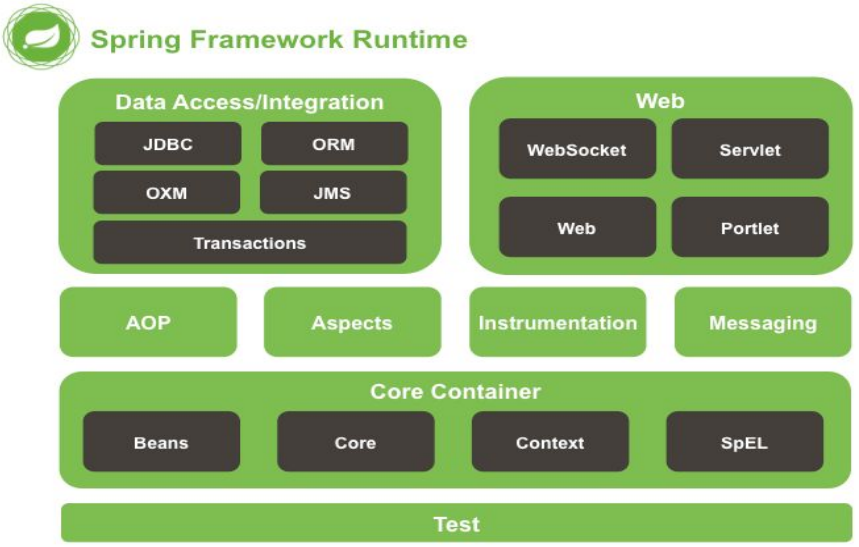
2. Arquitetura da Aplicação

Esse documento apresenta a arquitetura como uma série de visualizações/visões: visão lógica, visão de processos e visão de execução. Essas visualizações são apresentadas utilizando-se UML.

2.1. Representação arquitetural

O sistema está sendo desenvolvido tendo como base a arquitetura MVC adotada pelo Spring Framework. A Figura 1 representa a estrutura do framework.

Figura 1 –Versão básica do Spring Framework



Fonte:http://docs.spring.io/spring/docs/4.0.x/spring-framework-reference/html/overview.html#overviewcore-containe

2.2. Objetivos e Restrições da Arquitetura

A arquitetura proposta tem como objetivo separar os interesses de cada camada da aplicação, disponibilizar um sistema funcional e com potencial de adaptabilidade a mudanças.

2.3. Critérios da Avaliação Arquitetural

Os critérios utilizados para a seleção da solução arquitetural foram:

- Modularidade;
- Manutenibilidade;

3. Metas e Restrições da Arquitetura

Para a execução do sistema será requisitado a instalação do(s) seguinte(s) software(s):

- Mozilla Firefox v. 38 ou superior / Google Chrome v.43 ou superior;

Caso algum software indicado não seja instalado, a aplicação pode não funcionar como esperado, ou não funcionar.

Existem alguns importantes requisitos e restrições do sistema que possuem uma influência significativa na arquitetura: São elas:

- Todas as funções devem estar disponíveis através dos navegadores mais populares;

- O sistema deve possuir diferentes visualizações para diferentes níveis hierárquicos.

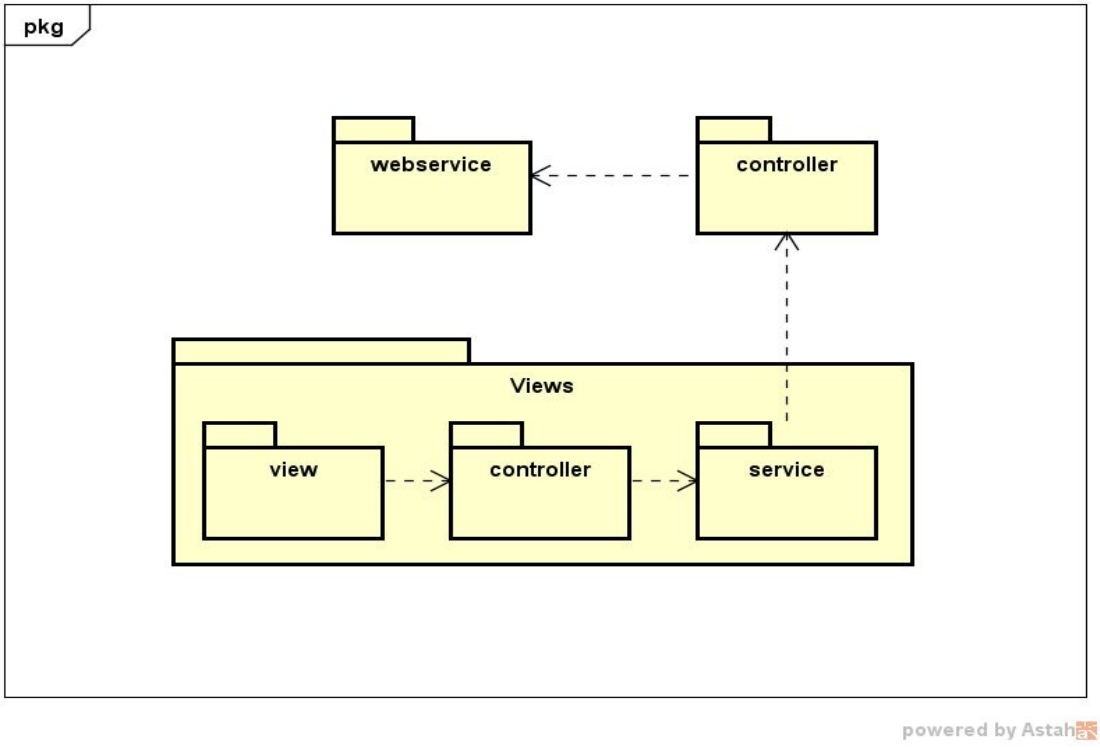
4. Visualização Lógica

Esta seção apresenta diferentes tipos de visões lógicas, com níveis de abstrações distintas, para que se possa obter uma visibilidade de comunicação entre pacotes e classes.

4.1. Visão de pacotes

A visão lógica da solução é composta por 4 diretórios principais:

Figura 2 - Visão de pacotes



Fonte: Próprio Autor

- ***Webservice***

Neste pacote estará contido as classes de mapeamento com o *web service* da plataforma HPC-*Shelf*. Através do arquivo de descrição das funcionalidades do *web service* (arquivo WSDL), será utilizada a ferramenta Jax-WS (KALIN, 2013), contido no pacote de bibliotecas padrão do Java, para criar as classes que farão a comunicação direto com o *web service*.

- ***Controller***

Neste diretório será implementada as classes que que serão nossos controladores, esses são responsáveis por conhecer as urls da aplicação e que lógica aplicar a

cada uma delas, bem como seu retorno. Além disso, são responsáveis por realizar o controle das seções dos usuários.

- ***Views***

Para garantir mais dinamismo na aplicação, parte do processamento da mesma ficará no lado do cliente (*client side*), e consequentemente, parte da lógica de negócio da aplicação nesta parte.

  - ***Service***

Neste pacote conterá os *script* JavaScript que farão a comunicação com o servidor da aplicação. Ex.: *ComponentService*. Todas as requisições a respeito de informações devem ser feitas através deste *script*.

  - ***Controller***

Neste pacote será implementado os *scripts* que conterão a outra parte da lógica de negócio da aplicação.

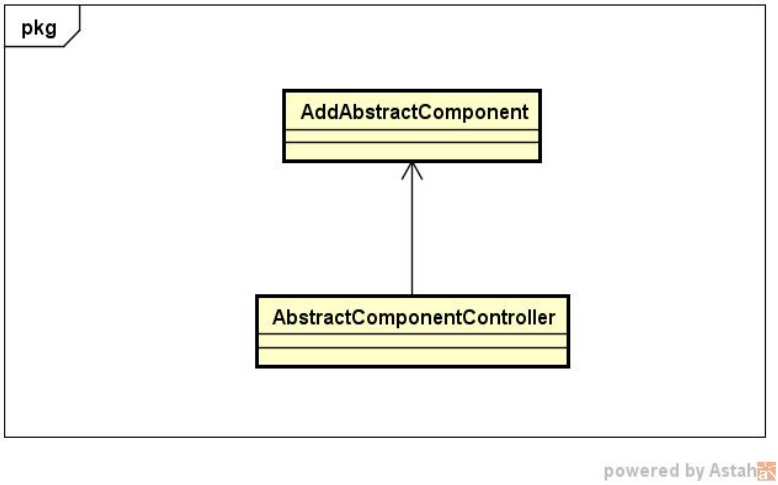
  - ***View***

Por fim, este pacote conterá as interfaces de usuário, onde será exibido todo o conteúdo requisitado pelos services.

4.2. Visão de Classes

Apresenta como as classes se comunicam entre si na aplicação. Neste caso, é abordado o caso de um registro de componente, com respeito aos pacotes ***controller***, ***service*** e ***webservice***.

Figura 3 - Visão do relacionamento entre classes

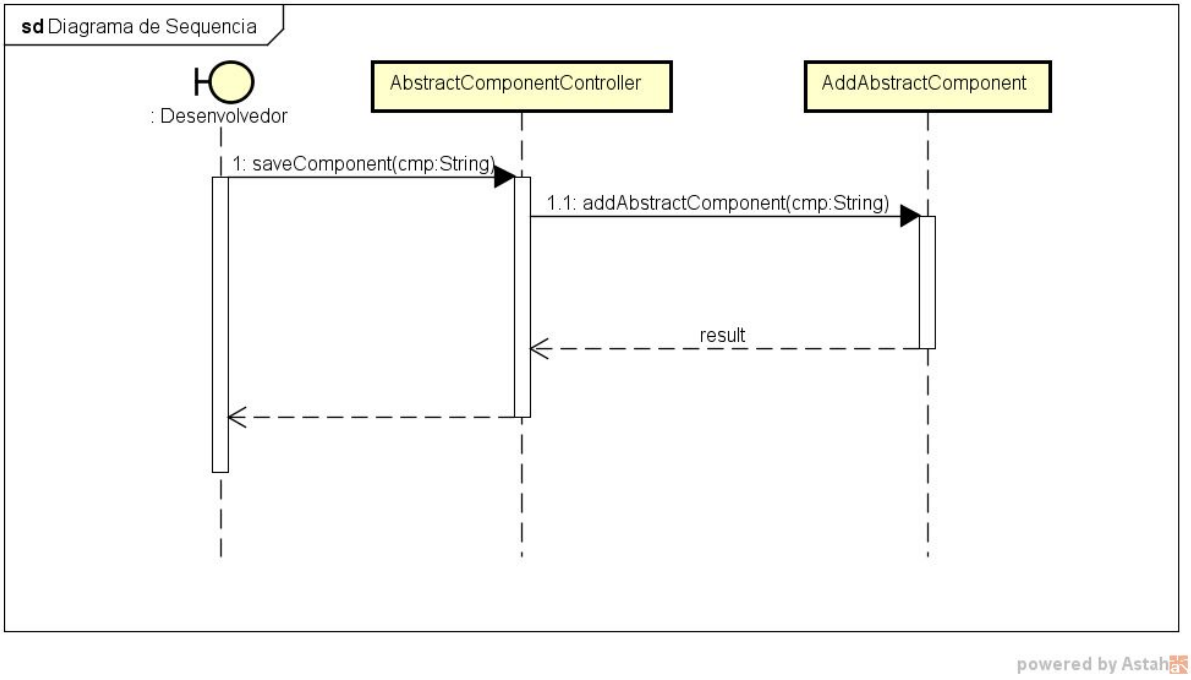


Fonte: Próprio Autor

5. Visão de Processos

Esta seção descreve a decomposição do sistema em processos, sendo evidenciado até então, o processo de registro de um novo componente no sistema. No qual o desenvolvedor logado no sistema faz uma requisição para criar um novo componente, o fluxo é representado na Figura 4.

Figura 4 - Visão de chamadas em sequência



Fonte: Próprio Autor

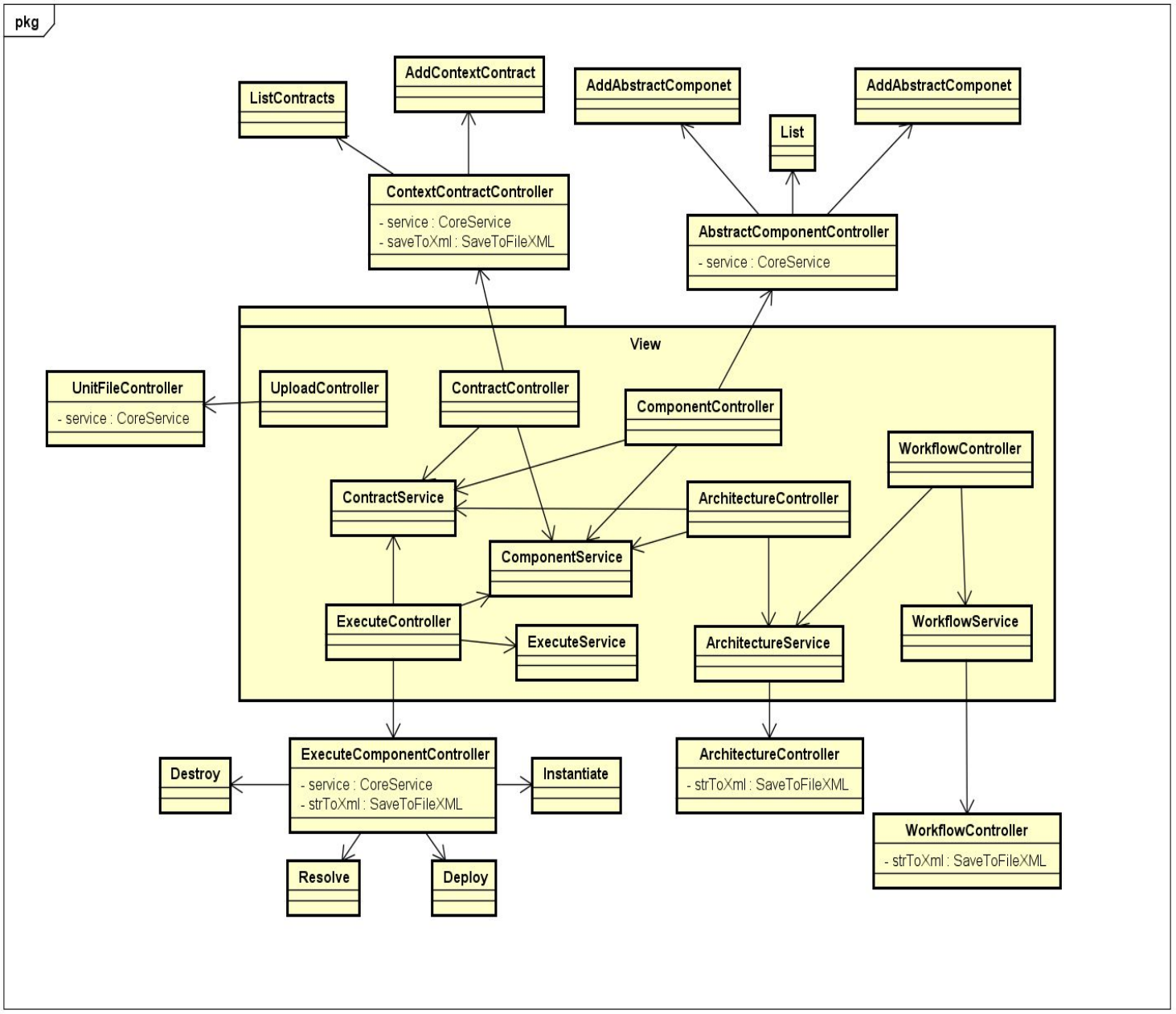
6. Visão de Implementação

Esta seção descreve a estrutura geral do modelo de implementação, as subdivisões arquiteturais e os componentes mais significativos da arquitetura.

6.2. Visão de Registro de Componentes Abstratos

Na figura 5 estão representadas todas as entidades dos controladores do projeto, com seu respectivos atributos e relacionamentos.

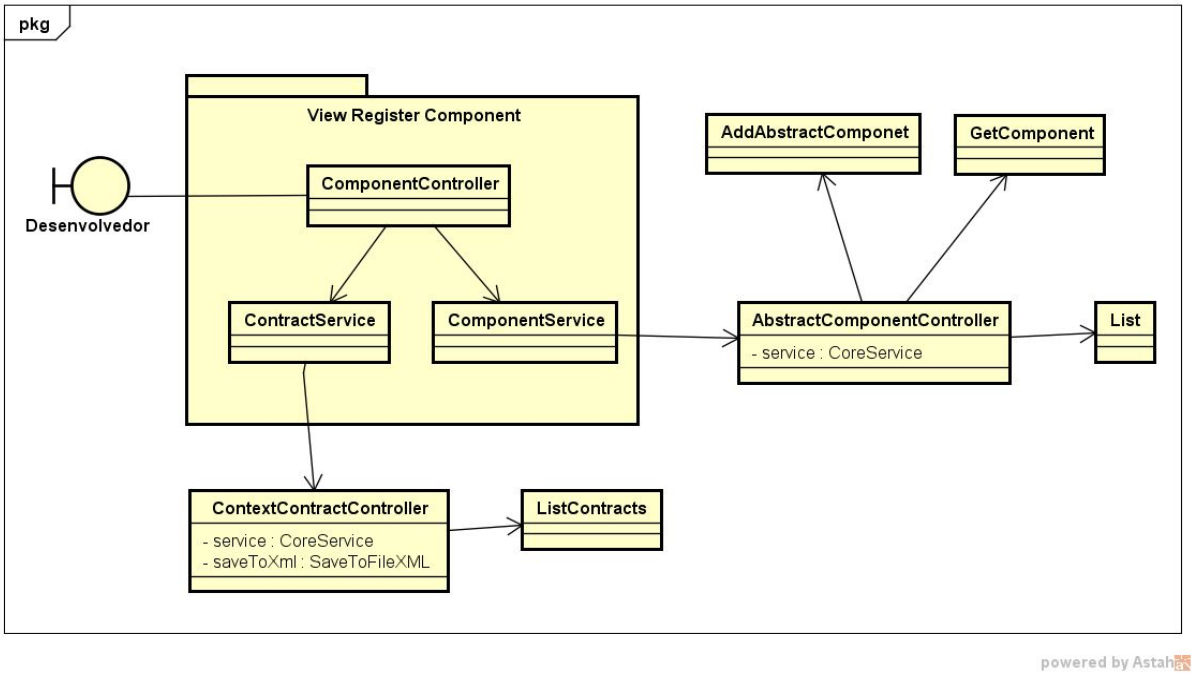
Figura 5 - Visão Geral de Classes



6.3. Visão de Registro de Componente

Na figura 6 são apresentadas as entidades que fazem parte do processo de criação de componentes abstratos.

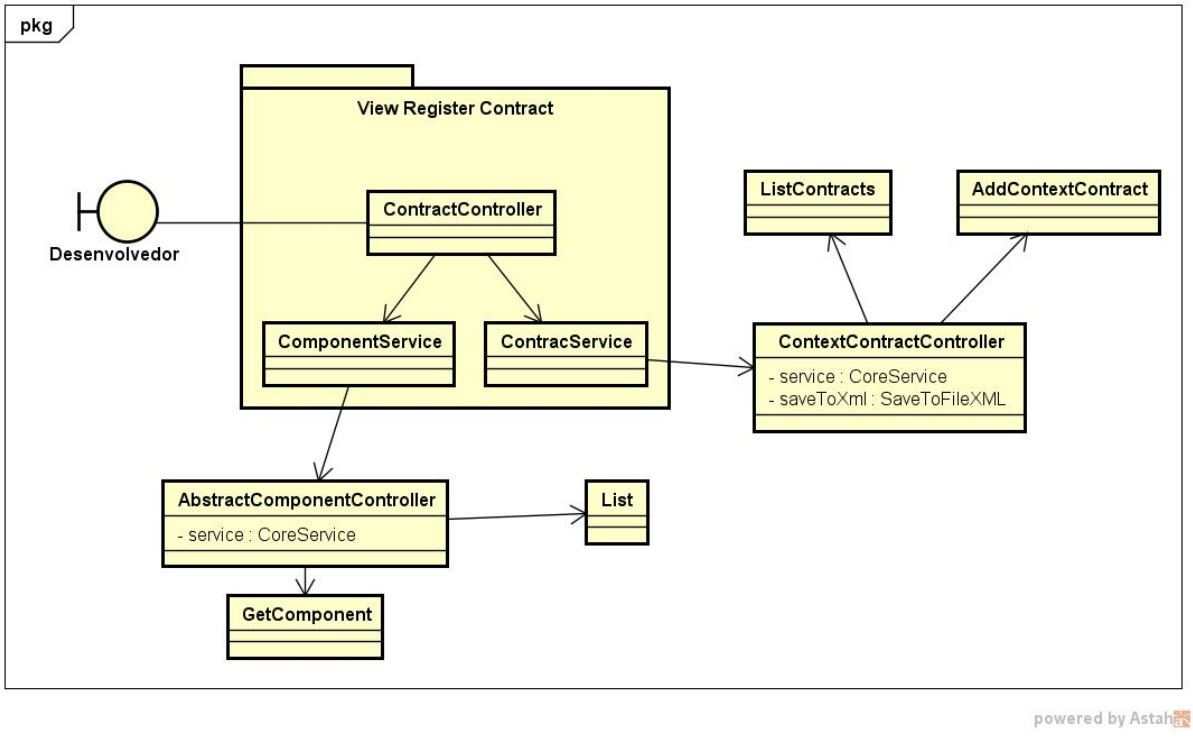
Figura 6 - Classes do processo de registro de componentes



6.4. Visão de Registro de Contrato Contextual

Na figura 7 são apresentadas as entidades que fazem parte do processo de criação de contratos contextuais.

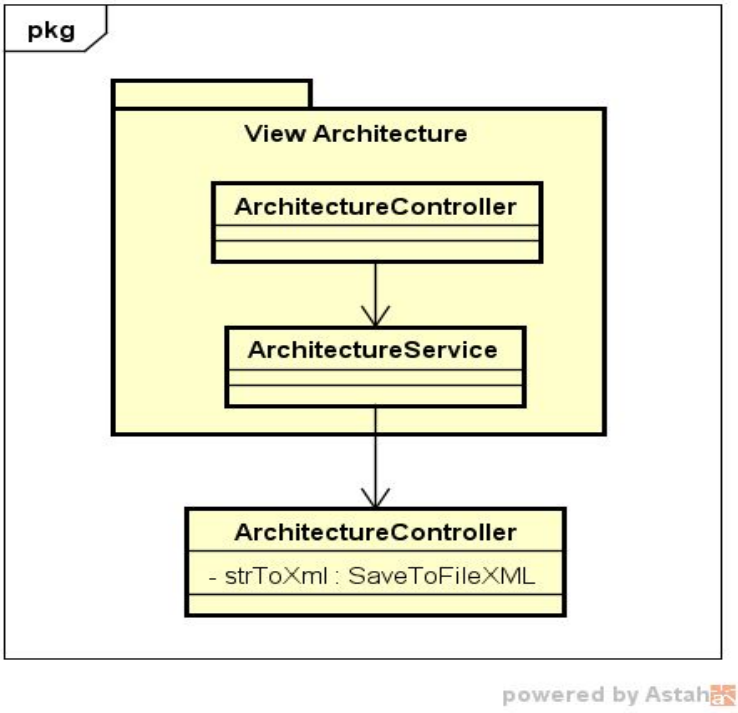
Figura 7 -Classes do processo de registro de contrato



6.5. Visão de Montagem da Arquitetura

Na figura 8 são apresentadas as entidades que fazem parte do processo de montagem de arquiteturas de aplicações.

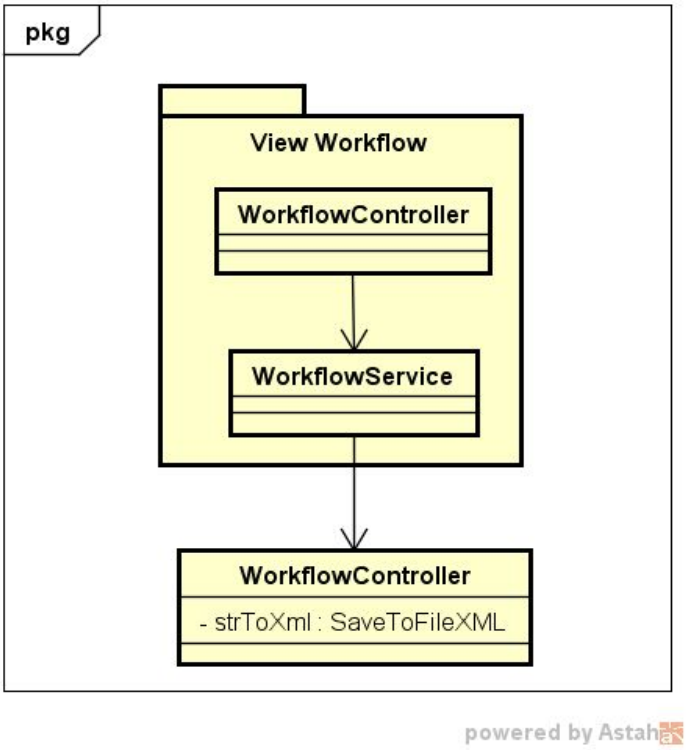
Figura 8 - Classes do processo de montagem de arquitetura



6.5. Visão de Montagem da *Workflow*

Na figura 9 são apresentadas as entidades que fazem parte do processo de criação de *workflow*.

Figura 9 - Classes do processo de montagem de arquitetura



7. Qualidade

- **Modularidade:**
  - Como Meyer (1988) define, modularidade pode ser vista como a habilidade de conceber um sistema a partir de "peças" encaixáveis através de uma interface coerente, melhorando assim a reusabilidade e a extensibilidade. Por isso, os componentes estão separados por seus estereótipos, cada classe pertencente ao sistema tem um pacote específico que agrupa classes que possuem o mesmo estereótipo.

- **Manutenibilidade:**
  - Segundo a IEEE (1990), manutenibilidade é a facilidade com que um sistema ou componente de software pode ser modificado para se corrigir falhas, melhorar desempenho (ou outros atributos), ou ser adaptado a mudanças no ambiente. Por isso, através da separação de interesses no momento de se criar um componente ou classe, seguindo padrões de nomenclatura, buscando manter a coesão entre pacotes, facilita-se assim a manutenibilidade da arquitetura.

REFERÊNCIAS

WEISSMANN, Henrique Lobo. **Vire o jogo com Spring Framework**. São Paulo, SP: Casa do Código, 2012. p. 133.

KALIN, Martin. **Java Web Services: Up and Running: A Quick, Practical, and Thorough Introduction**. " O'Reilly Media, Inc.", 2013.

IEEE Std. 610.12-1990. **IEEE Standards Collection: Software Engineering**. IEEE, 1993.

MEYER, Bertrand. **Object-oriented software construction**. New York: Prentice hall, 1988.