**Sistema de Alocação de Sessões Jurídicas (SASJ)**

**Plano de Gerência de Configuração**

Monteiro, 2018

**Revisões**

|  |  |  |
| --- | --- | --- |
| ***Data*** | ***Descrição*** | ***Autor*** |
| *25/04/2018* | Reformulação da estrutura de pastas do repositório | André Luís e Vinícius Gomes |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**CONTEÚDO**

[1](#_1ksv4uv) Introdução 4

[1.1](#_30j0zll) Objetivos 4

[1.1.1](#_1fob9te) Organização do Documento 4

[2](#_3znysh7) Papéis e Responsabilidades 4

[3](#_2et92p0) Plano de Configuração 4

[3.1](#_tyjcwt) Controle de Configuração 4

[3.1.1](#_3dy6vkm) Estrutura do Repositório de Gerência de Configuração 5

[4](#_1t3h5sf) Métodos de Identificação 6

[4.1](#_4d34og8) Documentos 6

[4.2](#_17dp8vu) Versionamento 7

[4.3](#_3rdcrjn) Baselines e Releases 7

[5](#_26in1rg) Ambiente, Ferramentas e Infra-estrutura 7

[5.1](#_lnxbz9) Plano de Software 7

[6](#_35nkun2) Estrutura de Branches do Projeto 8

# **Introdução**

Este documento descreve o Plano de Gerência de Configuração para o projeto de desenvolvimento do sistema SASJ - Sistema de Alocação de Sessões Jurídicas.

## **Objetivos**

O presente documento tem por objetivo apresentar a organização, nomenclatura e regras de versionamento para a gerência de configuração do projeto de desenvolvimento do sistema SASJ.

Este plano é destinado a todos os integrantes da equipe responsável pelo o desenvolvimento do sistema.

### **Organização do Documento**

Nesta seção deve ser descrita a estrutura do documento, conforme exemplo abaixo:

As seções subseqüentes deste documento estão assim organizadas:

* Seção 2, são descrito os papéis e responsabilidades da gerência de configuração;
* Seção 3 é apresenta o plano de configuração onde é definido a estrutura do armazenamento, as configurações bases do projeto, o controle de configuração e as políticas de segurança e acesso aos itens de configuração;

# Papéis **e Responsabilidades**

|  |  |
| --- | --- |
| **Papel** | **Responsabilidade** |
| Gerente de Desenvolvimento (GD) | Juntamente com a CCM receber, analisar e aprovar os PFM. |
| Líder de Projeto (LP) | Planejar as atividades de GC juntamente com o Responsável pela Configuração, designar executante, finalizar SM, autorizar a criação das configurações bases conforme descrito na seção Plano de Configuração. |
| Responsável pela Configuração (RC) | Criar e manter infra-estrutura corporativa (servidores) de GC; Implementar as políticas de Controle de Acesso ao ambiente de GC, Realizar os backups dos repositório de configuração dos projetos |

# **Plano de Configuração**

## **Controle de Configuração**

Para manter o controle de versão durante o desenvolvimento do projeto, será utilizado o *host* de versionamento de projetos Gitlab.

### **Estrutura do Repositório de Gerência de Configuração**

A seguir será apresenta a estrutura definida para armazenamento dos artefatos do projeto no repositório.

REPOSITÓRIO

**+-SISTEMA**

| +-000-REQUISITOS

| +-001-ANÁLISE E PROJETO

**| +-002-IMPLEMENTAÇÃO**

| | +-SASJ-API

| | +-SASJ-UI

| +-003-TESTE

| +-004-GERÊNCIA DE CONFIGURAÇÃO

**+-PROJETO-SASJ**

| +-000-GERÊNCIA DE PROJETO

| +-001-ACOMPANHAMENTO

O repositório é dividido em SISTEMA, para artefatos que pertencem ao sistema independente de quantos projetos irão desenvolvê-lo e PROJETO-SASJ, onde SASJ é o identificador do projeto. Isso é necessário pois vários projetos podem ser executados para o desenvolvimento do mesmo sistema. Os subdiretórios estão organizados pelas disciplinas do processo. A tabela abaixo identifica o que deve ser armazenado em cada subdiretório do projeto.

|  |  |  |  |
| --- | --- | --- | --- |
| Diretório Principal | Subdiretório | Descrição | |
| SISTEMA | 000-REQUISITOS | Armazenamento de Documentos de requisitos, documentos de casos de uso, Product backlog, User stories, etc. | |
| 001-ANÁLISE E PROJETO | Documentos referentes à análise e projeto do sistema como documentos de arquitetura, modelos UML, modelo de dados, etc. | |
| 002-IMPLEMENTAÇÃO | SASJ-API | Códigos relativos a API (*back-end*) e a interface gráfica (*front-end*) |
| SASJ-UI |
| 003-TESTE | Documento relativo aos cenários de teste para as User Stories. | |
| 004-GERÊNCIA DE CONFIGURAÇÃO | Armazenamento do plano do plano de gerência de configuração. | |
| PROJETO | 000-GERÊNCIA DE PROJETO | Armazenamento do plano de projeto. | |
| 001-ACOMPANHAMENTO | Deve possuir um subdiretório para cada iteração do projeto contendo o plano da iteração, reuniões de acompanhamento da iteração, Reunião de finalização da iteração, Burndown da iteração e timeline da iteração. | |

# 

# **Métodos de Identificação**

## **Documentos**

Todos os documentos disponibilizados no repositório devem ser identificados baseados na seguinte nomenclatura:

<ID\_ARTEFATO>-<NOME\_ARTEFATO>

Onde:

* <ID\_ARTEFATO> é a sigla de identificação do artefato conforme Tabela 6.
* <NOME\_ARTEFATO> é nome de identificação do artefato conforme Tabela 6.

|  |  |
| --- | --- |
| ID\_ARTEFATO | NOME\_ARTEFATO |
| PPJ | Plano de Projeto |
| REQ | Documento de Requisitos |
| UCS | Documento de Casos de Uso |
| PBL | Product Backlog |
| TEA | Testes de Aceitação |
| PGC | Plano de Gerência de Configuração |
| PIT | Plano de Iteração |
| DRI | Documento de Riscos |
| ATA | Ata de Reunião |
| RAP | Relatório de Acompanhamento do Projeto |
| BRD | Burndown |

Tabela 6 - Identificadores e Nomes dos Artefatos

## **Versionamento**

Para gerenciamento do versionamento do software, será adotado a maior parte das definições da especificação do Versionamento Semântico 2.0.0 - SemVer (<http://semver.org/lang/pt-BR/)> e prevê a seguinte estrutura:

**Versão: MAJOR.MINOR.PATCH**

1. “MAJOR” representa a versão principal do software. Ela será incrementada a cada nova release. Neste caso, “MINOR” e “PATCH” devem ser decrementadas para o valor zero (0).
2. “MINOR” representa a versão secundário. Ela será incrementada quando adicionar

funcionalidades compatíveis com as versões “MAJOR”, ou seja, que não afetam a

versão atual do Software. Neste caso, apenas o “PATCH” deve ser decrementada

para o valor zero (0).

1. “PATCH” representa a versão para possíveis correções na versão atual do software. Ela será incrementada quando houver correção de falhas que não afetam a versão atual do software.

## **Baselines e Releases**

Sempre que tivermos uma versão estável do software para testes ou aprovada da documentação deve ser gerada uma baseline com o objetivo de recuperarmos o estado em que ficou o código e a documentação.

Abaixo segue como serão nomeadas as baselines e releases. Ao final de uma iteração, uma baseline é gerada. Caso uma baseline possua erro uma nova baseline deverá ser gerada para uma nova rodada de testes. Assim, as baselines devem seguir a seguinte nomenclatura:

V-<ID\_VERSAO>IT<NUMERO\_ITERACAO>-BASELINE-<NUMERO\_DA\_BASELINE>

Onde:

* ID\_VERSAO: O identificador da versão que está sendo construída.
* NUMERO\_ITERACAO: O número da iteração em que foi gerada a baseline.
* NUMERO\_BASELINE: O número sequencial da baseline dentro da iteração.

Para releases a nomenclatura é parecida. Todo o fim de iteração deve ser gerada uma release com a versão a ser validada pelo cliente. Assim, as releases de iteração devem seguir a seguinte nomenclatura.

R-<ID\_VERSAO>-IT-<NUMERO\_ITERACAO>

Onde:

* ID\_VERSAO: O identificador da versão que está sendo construída.
* NUMERO\_ITERACAO: O número da iteração em que foi gerada a release. >>

# **Ambiente, Ferramentas e Infra-estrutura**

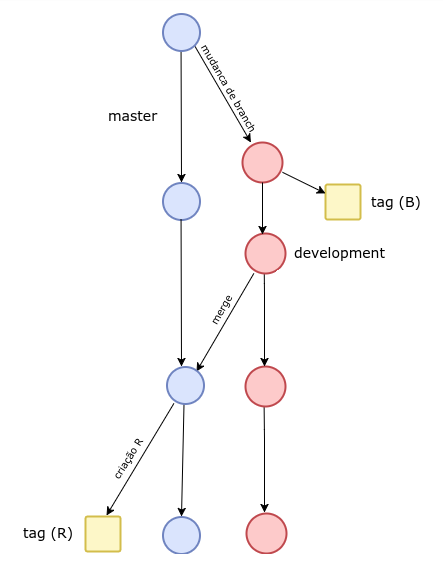
## **Plano de Software**

|  |  |  |  |
| --- | --- | --- | --- |
| **Software** | **Propósito** | **Ambiente** | **Release/Versão** |
| Git | Controle de Repositório | Desenvolvimento | 2.17.0 |
| Google Docs | Documentos de gerenciamento do projeto | Todos | -online- |
| Gitlab | Acesso ao repositório | Desenvolvimento | -online- |
| Pipefy | Edição do cronograma | Desenvolvimento | -online- |
| Java JDK | Kit básico para o desenvolvimento em Java | Desenvolvimento | 1.8 |
| Spring Framework | Framework MVC para o desenvolvimento de aplicações Web. | Desenvolvimento | 5.0 |
| Spring Boot | Ferramenta de configuração de projetos Spring | Desenvolvimento | 1.5.10 |
| Angular | Framework front-end para desenvolvimento de GUI responsiva e amigável | Desenvolvimento | 4.0 |
| Angular CLI | Ferramenta de configuração de projetos Angular | Desenvolvimento | 1.7.3 |
| Node.js | Plataforma para construir aplicações web escaláveis JavaScript | Desenvolvimento | 8.10 |
| Hibernate | Framework de persistência. | Desenvolvimento | 5.0 |
| Visual Studio Code | IDE de desenvolvimento. | Desenvolvimento | 1.2 |
| STS | IDE de desenvolvimento. | Desenvolvimento | 3.6.2 |
| MySQL | Banco de dados da aplicação. | Desenvolvimento | 6.5 |
| Flyway | Ferramenta de automação de migrações contendo inserções no banco e modelagens no mesmo. | Desenvolvimento | 4.0 |
| Heroku | Servidor de hospedagem | Produção | -- online -- |
| Heroku CLI | Ferramenta de configuração do heroku | Desenvolvimento | 7.0.12 |
| MySQL Workbench | Gerenciador de banco de dados MySQL. | Desenvolvimento | 6.3 |

***Tabela 7 - Plano de Software***

# **Estrutura de Branches do Projeto**

Nesta seção deve ser descrita a estratégia de branches para execução do projeto. Para alguns exemplos, consulte os links abaixo.



* A *branch master* deverá conter os artefatos da release mais recente. Após a conclusão de cada iteração é recomendado a realização de um merge da *branch development* para *branch master* dos artefatos produzidos.
* A *branch development* será o local de produção dos artefatos do projeto, artefatos como, desenvolvimento de novas funcionalidades, testes, atualização de documentação e etc.
* Uma *tag* deve ser criada sempre que existir uma versão do software usável. Assim que o software estiver com as funcionalidades prometidas implementadas, testadas e aprovadas, deve-se gerar uma *tag* e uma *release* com esta versão (observando-se a quantidade de *releases* planejadas e quais funcionalidades ela deve conter).