	Plano de Gerenciamento de Configuração
Data: 06/11/2018	**Objetivo Estratégico:** Gerenciamento dos dados administrativos do Instituto Ada Lovelace.
ID: 1008	**Nome do Projeto:** sigAda - Sistema de Gerenciamento do Ada
CC: Paulo Pires	**Cliente:** Instituto Ada Lovelace
Patrocinador: UFRJ	**Gerente de Projeto:** Rafael Cardoso

1. Histórico de Versão

Data	Versão	Descrição	Autor(es)
06/11/2018	1.0.0	Abertura do plano de controle de versões	Eduardo, Rafael e Tainá
15/11/2018	1.1.0	Adição de novos exemplos no tópico 4.1.2 na arte da nomenclatura dos artefatos	Rafael, Eduardo, Tainá e Ricardo
16/11/2018	1.2.0	Adição dos tópicas de estratégia de branch e merge. Adição do tópico Check-in e Check-out do desenvolvedor. Correção da numeração dos tópicos.	Rafael e Tainá
17/11/2018	1.3.0	Adição de convenção de nomenclatura referente aos códigos	Tainá
18/11/2018	1.3.1	Correção da tabela do tópico 3.2, em que a ferramenta Google Drive foi adicionada como controle de versão da documentação.	Rafael
10/12/2018	1.4.0	Alteração da tabela no tópico 4.1.6 e	Eduardo, Rafael e

		correções no texto.	Tainá
11/12/2018	1.5.0	Adição da nomenclatura utilizada para a criação de uma nova branch.	Rafael

2. Introdução

2.1. Finalidade

Este documento tem como finalidade apresentar as ferramentas e metodologias utilizadas para o gerenciamento de configuração e controle de versionamento.

2.2. Escopo

Este documento especificará padrões de organização, nomenclatura, funcionamento e modo de utilização de todas as ferramentas utilizadas para controle de versionamento da documentação e dos códigos implementados.

2.3. Definições, Acrônimos e Abreviações

Termos	Significados
IDE	Ambiente de Desenvolvimento Integrado (Integrated Development Environment)
GC	Gerente de configuração
CS	Caso de uso

2.4. Visão Geral

Os próximos tópicos deste documento descrevem como o projeto será estruturado em termos de nomenclaturas de artefatos, organização de código fonte e documentação. Além disso, é definido como se dá a gerência da configuração do projeto, qual(is) pessoa(s) estão no cargo e quais são suas responsabilidades. Ao final, é apresentado alguns recursos utilizados para o estudo das ferramentas selecionadas para o versionamento dos artefatos, como tutoriais em vídeo e documentação.

2.5 Referências

- Documento de Arquitetura (ID - 1004)

3. Gerenciamento de Configuração de Software

3.1. Organização, Responsabilidades e Interfaces

O gerente de configuração tem um papel importante no projeto, uma vez que assegura que os artefatos gerados estejam corretos e coerentes de acordo com as especificações apresentadas neste documento. Também é ele que é responsável por controlar as mudanças realizadas no projeto.

Gerente de configuração (GC): Rafael Cardoso.

3.2. Ferramentas, Ambiente e Infraestrutura

A gerência da configuração se dará através do sistema operacional Windows e Linux, utilizando as ferramentas abaixo, precisando, portanto, de uma conexão com a internet de qualidade.

Tipo	Ferramenta	Versão	Função
Controle de Versão	Git e GitHub	-	Repositório para os códigos e documentação e o mesmo será responsável pelo versionamento
Controle de Versão	Google Drive	-	Repositório da documentação e o mesmo será responsável pelo versionamento
IDE	Eclipse e/ou Netbeans	-	Utilizado para a implementação do software, tendo integração com o GitHub, para que se possa atualizar o código armazenado nos repositórios
Editor de Texto	Google Docs	-	Utilizado para a criação

			dos artefatos gerados no projeto

4. O Programa de Gerenciamento de Configuração

4.1 Identificação da Configuração

4.1.1 Nomenclatura

Documentação

Os artefatos gerados pelo projeto, referente à documentação, devem seguir o seguinte padrão de nomenclatura para o título do arquivo:

<TIPO_ART>_InstitutoAdaLovalace.<TIPO_EXT>

Identificador	Significado
<TIPO_ART>	Tipo de artefato, onde as iniciais de cada palavra estão em maiúsculo e não há espaço entre as palavras. Ex.: PlanoDeProjeto, PlanoDeIteração
<TIPO_EXT>	Tipo de extensão de arquivo. Ex.: pdf, docx, odt

Para as imagens, como os diagramas e esquemas, será utilizado o seguinte padrão:

<SIGLA_TIPO_IMG>_<DESCRIÇÃO_IMG>_InstitutoAdaLovelace.<TIPO_EXT>

Identificador	Significado
<SIGLA_TIPO_IMG>	Sigla que diz ao que a imagem se refere. Para criar a sigla, usa-se até três caracteres para diferenciar as siglas entre elas, caso seja necessário. Ex.: DA - Diagrama de Atividade

	DS - Diagrama de Sequência DC - Diagrama de Classe MER - Modelo Entidade-Relacionamento
<DESCRIÇÃO_IMG>	Descrição de poucas palavras do conteúdo da imagem. Tal descrição não contém espaço e as iniciais das palavras estão em maiúsculo. Ex.: CS01FazerLogin CS18
<TIPO_EXT>	Tipo da extensão do arquivo. Ex.: jpg, png, gif

Código

Os artefatos gerados pelo projeto, referente ao código fonte, deve seguir as regras de nomenclatura mencionadas no Documento de Arquitetura, cujo link está disponível no início deste documento.

4.1.2 Versionamento dos artefatos

Nomenclatura

O número de versão dos artefatos é dada pela lógica abaixo:

<primeiro>.<segundo>.<terceiro>

Onde, o acréscimo de 1 é feito caso:

- <primeiro> : Uma alteração que modifica completamente o artefato é aprovada pelo cliente.
- <segundo> : Uma alteração é feita no artefato, porém suas características principais são mantidas.
- <terceiro> : É feita uma correção de detalhe no artefato, como uma correção de *bug*, ortográfica, formatação, entre outras.

Comentários de release

A plataforma GitHub disponibiliza que comentários possam ser adicionados a cada alteração nos códigos armazenados nos repositórios. Utilizaremos deste benefício para fazer um melhor controle do que está ocorrendo naquele artefato em específico, seguindo a estrutura a seguir:

<SIMBOLO> <COMENTARIO>

Identificador	Significado
<SIMBOLO>	Representa a ação que foi realizada naquele artefato. Os símbolos pode ser: ● "+" : Adição de algum elemento. ● "-" : Eliminação de algum elemento. ● "#": Alteração/Correção de algum elemento.
<COMENTARIO>	Descrição da localização que sofreu a ação dita pelo símbolo, além disso, pode ser escrito o motivo daquela ação. O texto deve ser claro e sucinto. Caso se tenha feito a ação em mais de um lugar, eles devem ser listados, utilizando como separador o ";"

4.1.3 Check-in e Check-out do desenvolvedor

Os indivíduos responsáveis pelo desenvolvimento do sistema devem realizar as técnicas de check-in e check-out.
Desse modo, o desenvolvedor ao iniciar as suas atividades deve, previamente, realizar um check-out de código do sistema de gerenciamento de versões para a sua máquina local, a fim de sempre trabalhar com as versões atualizadas do sistema.
O mesmo se aplica ao desenvolvedor que está implementando ou corrigindo algo do sistema. É necessário que ao término dessa atividade, ou quando for solicitado, realize check-in de código para o sistema de gerenciamento de versões, a fim de que este sempre esteja com a versão mais atualizada do sistema. Assim, outros desenvolvedores, ao realizarem o check-out, terão a versão mais atualizada em suas máquinas.

4.1.4 Estratégia de branch

A ramificação (*branch*) será feita quando for necessário desenvolver ou alterar um recurso isolado do ramo principal *master*. Desse modo, sempre que for necessário acrescentar uma nova *feature* ao sistema ou realizar correções de "bugs" críticos será feito uma nova ramificação.
Caso seja necessário, ramificações de ramificação podem ocorrer ao longo do desenvolvimento do sistema. Um exemplo seria fazer uma ramificação da "master" para acrescentar um novo

recurso e, ao terminá-la, fazer uma nova ramificação para a correção de bugs do novo recurso desenvolvido.
É de suma importância que os nomes das ramificações criadas sejam intuitivos, de modo a facilitar o acompanhamento e busca dessa ramificação. Assim, uma possível nomenclatura adotada foi de colocar o nome do pacote ou utilizar o próprio nome do arquivo e, em seguida, colocar o nome da pessoa que está realizando a modificação. Um exemplo de branch seria DAORafael. Se alguma modificação em algum arquivo for necessária, a branch deve começar com Hotfix. Um exemplo de branch de correção seria HotfixManterAlunoFormRafael. Esse padrão deve ser utilizado para facilitar a compreensão do que está sendo trabalhado em cada branch.

4.1.5 Estratégia de merge

Ao término de uma nova feature e/ou correção do sistema, será necessário realizar o "merge" com o ramo principal "master". Como o sistema desenvolvido só será utilizado para atender o Instituto Ada Lovelace, não há a necessidade de criar ramificações que não se fundam com o ramo "master" para criar funcionalidades distintas a fim de atender a necessidade especiais de outro instituto.
Desse modo, para realizar o merge será preciso testar se a nova funcionalidade desenvolvida não apresenta nenhum problema e se, ao fundir com o ramo "master", as outras funcionalidades desenvolvidas continuarão funcionando corretamente. Caso, alguma funcionalidade deixe de funcionar após o merge, uma nova ramificação será criada a fim de arrumar esse problema. Se ocorrer algum conflito de merge, em que alterações foram feitas em linhas de código já existentes tanto nesse ramo quanto no ramo "master", será preciso decidir quais as linhas de código que serão mantidas ou que serão deletadas. Em casos extremos, o merge pode ser desfeito e, também, é possível voltar para um "commit" específico a fim de restaurar o sistema para um estado em que se encontrava funcional.

4.1.6 Estrutura do repositório de versões

O repositório principal é destinado ao armazenamento da documentação, dos códigos implementado e para a apresentação do projeto.

Diretório	Conteúdo
.	README do repositório, contendo algumas informações básicas.

./settings	Arquivos criados pela IDE.
./Documentos	Esse diretório contém todos documentos gerado durante o desenvolvimento. Tais documentos estão em PDF.
./sigAdaForms	Diretório que contém as imagens utilizadas pelo software nos formatos jpg e png.
./bin	Arquivos criados pela IDE.
./src	Esse diretório contém todos os códigos gerados.
./Apresentacao	Diretório voltado para as apresentações periódicas do projeto.

4.2 Processo de Armazenamento de Mídia

O repositório do projeto deve ser *clonado* pelos integrantes da equipe, bem como armazenado em alguma ferramenta de armazenamento na nuvem, como o Google Drive. Isso é feito para, caso ocorra algum problema com o repositório original, existem cópia dos artefatos guardadas.

5. Treinamento e Recursos

- O canal no YouTube, Felipe Classroom Tutorials, possui um tutorial sobre o Google Docs. Disponível em: <https://www.youtube.com/playlist?list=PLwXXOxvDboeagQBUlNBfC_-t9vA9c3bul>
- O GitHub também disponibiliza um tutorial interativo para aprender a como usá-lo. O tutorial pode ser encontrado em: <https://try.github.io>
- Diversos conteúdos a respeito do controle de versão e gerência de configuração de forma completa, mas didática pode ser encontrado em: <http://git-scm.com/doc>