# 

# 

# 

**SCM Plan**

Engenharia de Software II

2021/22

# 

# 

**Trabalho prático II**

Realizado por:

Bruno Dylan Pinto Ferreira, nº 8200586

Gonçalo André Fontes Oliveira, nº 8200595

Jorge Miguel Fernandes Correia, nº 8200592

Nuno de Figueiredo Brito e Castro, nº 8200591

# 1. Introdução

## 1.1 Purpose

Este documento possui o planeamento das atividades de ***Software Configuration Management (SCM)*** utilizadas ao longo do desenvolvimento deste trabalho. Podem ser observados todos os detalhes referentes a cada atividade, bem como os seus respetivos responsáveis e em que situações/momentos serão realizadas.

## 1.2 Scope

Tal como referido anteriormente, este documento contém detalhes referentes às atividades, bem como o respetivo responsável pela mesma, por isso, podemos concluir que este documento influenciará basicamente toda a gestão do projeto que, tem de ser realizada consoante o referido neste relatório.

## 1.3 Definitions, Acronyms and Abbreviations

**SCM -** Software Configuration Management

**ECP** - Equivalence Class Partitioning

**BVA** - Boundary Value Analysis

**IDE -** Integrated development environment

## 1.4 Referências

* Conteúdos da disciplina no moodle

# 

# 2. Software Configuration Management

## 2.1 Organização, responsabilidades e interfaces

**Role SCM:** Scrum master

**Responsável:** Bruno, Jorge, Gonçalo, Nuno

**Responsabilidades:**

* Assegurar que o projeto é executado de acordo com as práticas, valores e regras do Scrum e que progridem como planeado.
* Interagir com a equipa de projecto, bem como com o cliente e gestão durante o projecto
* Assegurar que qualquer impedimento é removido e alterado no processo para manter a equipa a trabalhar tão produtivamente quanto possível

**Role SCM:** Product Owner

**Responsável:** Bruno, Jorge, Gonçalo, Nuno

**Responsabilidades:**

* Toma as decisões finais das tarefas relacionadas com o product Backlog
* Supervisionar o projeto
* Estimar esforços de desenvolvimento para os backlog itens
* Aprovar merge requests
* Avaliar o desenvolvimento do projeto
* Gerir o ciclo de vida de desenvolvimento de software.

**Role SCM:** Scrum team

**Responsável:** Bruno, Jorge, Gonçalo, Nuno

**Responsabilidades:**

* Decidir as acções necessárias e organizar-se no sentido de alcançar os objectivos de cada Sprint
* Estimar esforços, criação do Sprint Backlog, revisão do Product Backlog list e sugerir restrições que necessitam de ser removidas do projecto

**Role SCM:** Customer

**Responsável:** Bruno, Jorge, Gonçalo, Nuno

**Responsabilidades:**

* Participar nas tarefas relacionadas com product Backlog itens para o sistema a ser desenvolvido ou melhorado.

**Role SCM:** Management

**Responsável:** Bruno, Jorge, Gonçalo, Nuno

**Responsabilidades:**

* Participa na definição de objectivos e requisitos.

## 2.2 Ferramentas utilizadas

**Java** - A linguagem Java foi a utilizada para este projeto

**IntelliJ** - IDE utilizado para a codificação e execução do código

**GitLab** - A plataforma Gitlab deve ser vista como uma ferramenta SCM que permite a gestão de código fonte, gestão de alterações e gestão de building e releases. Foi o repositório git utilizado para o desenvolvimento deste trabalho.

**Testes de software** - Um teste de software é um software que executa outro software, validando se os resultados são os esperados (teste de estado) ou se executa a sequência de eventos esperado (teste de comportamento).

Resumidamente, os testes de software permitem aos programadores verificar se a lógica do programa desenvolvido está de acordo com os requisitos.

A execução automática de testes permite identificar “bugs” resultantes de mudanças no código fonte.

**ECP** - Técnica destinada a reduzir o número de testes necessários dividindo o domínio de entrada (ou saída) em classes de dados em que os casos de teste podem ser derivados para cada operação, o “tester” deve identificar as classes de equivalência dos argumentos e os estados dos objetos

**BVA** - Técnica baseada na observação de bugs que ocorrem frequentemente em valores fronteira. É focada em testar valores especais (null, 0, etc) e limites do domínio de entrada (ou saída) imediatamente acima e abaixo (além de ou em vez de valores intermédios)

**JUnit** - Framework open-source para realizar testes unitários para código em Java. Permite a execução de testes de forma:

* Fácil
* Regular
* Fiável

Contém várias funcionalidades para testing entre elas a capacidade de testar cada componente de um programa de forma independente do resto do programa

**Gradle** - É uma ferramenta que permite integrar e automatizar várias tarefas relacionadas processo de desenvolvimento de software em várias linguagens de programação.

O Gradle determina quais os componentes do projeto que estão atualizados, evitando a recompilação de todo o projeto.

**Jacoco** - É uma biblioteca gratuita de cobertura de código para Java, que disponibiliza informações, tal como o nome indica, de taxas de cobertura, quer de ramos quer de instruções, identificando ainda quais as instruções/ramos que não foram cobertos.

**Teams** - O Teams foi a ferramenta de comunicação utilizada para realizar discussões e tomar decisões sobre o projeto.

# 3. The configuration Management Program

## 3.1 Configuration Identification

### 3.1.1 Identification Methods

Nesta secção serão definidos os nomes que serão dados ao diversos artefactos que posteriormente serão criados ao longo do desenvolvimento do projeto:

* Para cada branch criado para o desenvolvimento de cada módulo, será dado o nome ***Dev\_NomeModulo.***
* Para os branch’s criados dentro dos Dev’s, será dado o nome ***Feature\_NomeDaFeature.***
* *Para o nome de cada Sprint, será dado o nome* ***Sprint#NumeroDaSprint.***

### 3.1.2 Project Baselines

Ao longo do projeto usando o grupo que criamos no gitlab, fomos estabelecendo marcos temporais para realizar cada módulo.



## 

## 

## 3.2 Configuration and Change Control

### 3.2.1 Change Request Processing and Approval

Ao longo do desenvolvimento do projeto poderá ser necessário realizar algumas alterações.

Sempre que é necessário proceder a alterações é necessário criar uma issue com a label **Change Request: New** associada, onde deverão ser explicadas todas as alterações necessárias e o seu respetivo propósito.

Após isso deve ser notificado no grupo da plataforma Teams que foi criado um change request e que necessita de ser analisado.

Para a aprovação ou negação do change request, existirá um change request board que permitirá alterar o estado da issue para **Change Request: Approved** ou **Change Request: Rejected.**

### 3.2.2 Change Control Board

Tal como referido anteriormente, existirá um board que permitirá gerir os change requests que, poderão ter três estados:

* **Change Request: New →** Change request criado que necessita ser analisado.
* **Change Request: Approved →** Change request aprovado, portanto podem ser realizadas as alterações.
* **Change Request: Rejected →** Change request rejeitado, portanto não podem ser realizadas as alterações.

## 3.3 Configuration Status Accounting

### 3.3.1 Project Media Storage and Release Process

Os dados são guardados online, no repositório do GitLab e, vendo que o GitLab é um sistema de controlo de versões, a recuperação de planos e backups são facilmente acessíveis.