Informatics Engineering (Integrated Master's), University of Minho - 2019/2020

CLP para teste das Operações do **Serviço de Assinatura com Chave Móvel Digital**

Reverse Engineer da aplicação CMD-SOAP com aplicação dos conceitos de Programação Segura

Grupo 5 - Diogo Araújo A78485, Diogo Nogueira A78957 - **Engenharia de Segurança**

ÍNDICE DE CONTEÚDO

TÓPICOS A ABORDAR AO LONGO DA APRESENTAÇÃO

1 TÉCNICAS DE DESENVOLVIMENTO DE *SOFTWARE*

1.1. Uso de *Third-party Components*
1.2. Gestão dos *Third-party Components*

USO DE *THIRD-PARTY COMPONENTS*

NECESSIDADE/VANTAGENS DE INTRODUZIR *THIRD-PARTY COMPONENTS*

Menor Ciclo de Desenvolvimento de *Software*

Avanço no Projeto Final

Diminuição a Custo Prazo

USO DE *THIRD-PARTY COMPONENTS*

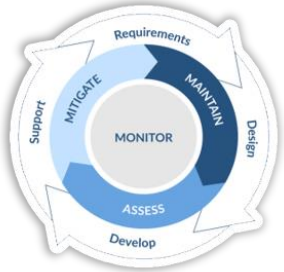
USO DE *EXTERNAL LIBRARIES* DO JAVA

Nome do TPC	Utilidade do *Third-party Component*
JAX WS RI Runtime, Javax JWS API e JAX WS API	Usadas para manipular tudo aquilo que foi criado e compilado pelo ***First-party Component*** **wsimport.** No projeto, este FPC usado para criar todas as classes que permitem estabelecer uma ligação com o SOAP *Server* para a execução das várias operações do serviço de *Signature* CMD.
Apache Commons Lang	Usada para manipular *Strings*. No projeto foi usado para manipular/extrair conteúdo do Certificado e ainda para extrair apenas os 9 dígitos pertencentes ao *User Id* (Número de Telemóvel) do utilizador da aplicação.

GESTÃO DOS *THIRD-PARTY COMPONENTS*

NECESSIDADE ESTABLECER INGREDIENTES-CHAVE PARA UMA BOA GESTÃO DOS TPC's

Estabelecimento das várias Fases do TPC *Life Cycle*



Relação do TPC *Life Cycle* com o *Software Life Cycle*

Produto Final com todos os TPCs supervisionados

GESTÃO DOS *THIRD-PARTY COMPONENTS*

FASE 1 *MAINTAIN* – Manter uma Lista de TPC's

Método	Descrição do Método e sua Utilidade
Criação de uma Bill of Materials (BOM) através de Ferramentas Automáticas	Dado que se programou na linguagem Java, a decisão passou por criar todo o projeto em modo Maven, criando-se desde logo uma BOM automática através do ficheiro pom.xml. **Assim, enumera todas as dependências do software em causa e as suas respetivas versões, permitindo ainda ter todas estas bibliotecas na sua versão mais recente.**
Uso de Identificadores Únicos para cada TPC	Esta prática encontra-se automaticamente em execução ao fazer-se uso do Maven Project Object Model, uma vez que cada dependência apresenta a sua própria identificação.

GESTÃO DOS *THIRD-PARTY COMPONENTS*

FASE 2 *ASSESS* – Avaliar o risco de segurança

Nome do TPC/Java *Library*	Data da Última Atualização	N.º de Artifacts que usam o TPC	Contactos Developers	Vulnerabilidades Recentes
JAX WS RI Runtime	Maio de 2020	243	Sim	Não
Javax JWS API	Junho de 2018	32	Sim	Não
JAX WS API	Outubro de 2018	764	Sim	Não
Apache Commons Lang	Março de 2020	16.809	Sim	Não

GESTÃO DOS *THIRD-PARTY COMPONENTS*

FASE 3 *MITIGATE* – Mitigar ou aceitar o risco

A edição da BOM no ficheiro pom.xml fornece capacidades de mitigar erros conhecidos.

Todas as bibliotecas usadas estão nas suas versões mais recentes sem vulnerabilidades conhecidas.

Em https://mvnrepository.com listam-se as várias bibliotecas em uso e verificar a versão mais recente para cada uma delas.

GESTÃO DOS *THIRD-PARTY COMPONENTS*

FASE 4 *MONITOR* – Monitorizar mudanças aos TPC's

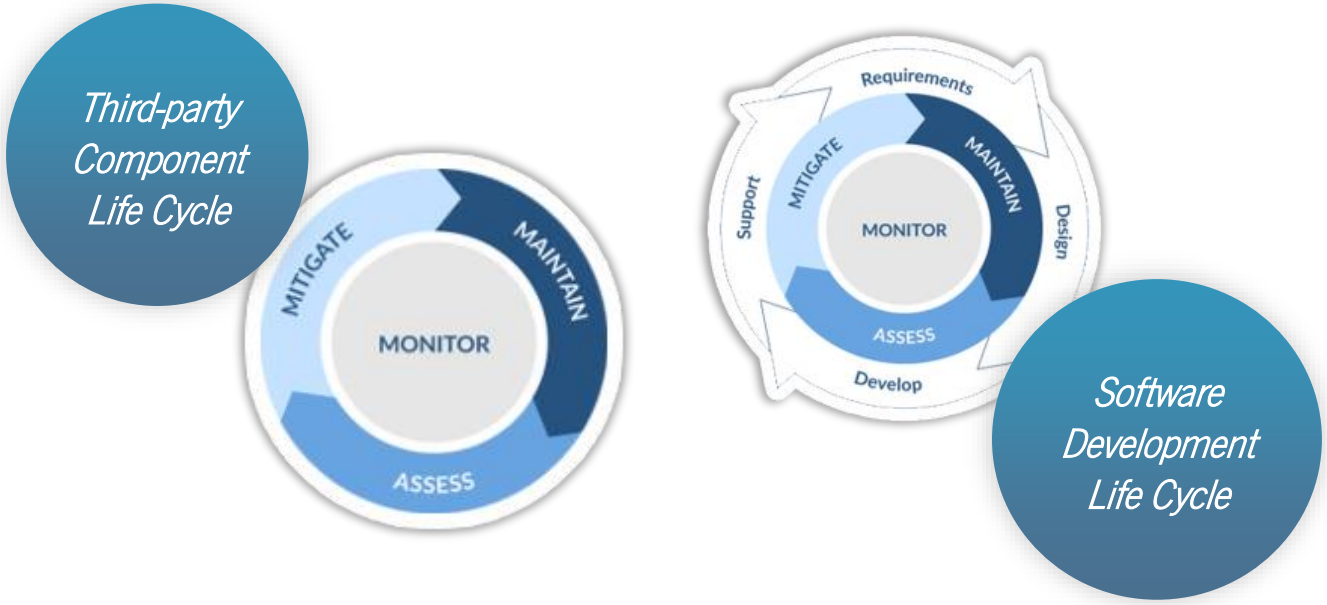
- Verificar a possível *End of Life* (EOL) dos TPC's, analisando se o componente em si foi descontinuado ou apenas a sua versão atual. O próprio IntelliJ IDEA consegue fazer esta verificação através dos vários *warnings* que são mostrados ao longo do código.

- Atentar a possíveis vulnerabilidades que possam surgir para os componentes usados no projeto.

- Manter um perfil de risco no geral de forma a se poder comunicar futuras vulnerabilidades.

CICLO DE VIDA DE UM TPC VS SDLC

ETAPAS DE AMBOS OS CICLOS



2 FERRAMENTAS E INDICADORES DE QUALIDADE E/OU TESTES DE *SOFTWARE*

2.1. Características de Qualidade do *Software*
2.2. Métricas de Qualidade do *Software*

CARACTERÍSTICAS DE QUALIDADE DE *SOFTWARE*

ISO/IEC 25010 E AS OITO CARACTERÍSTICAS DE QUALIDADE DO *SOFTWARE*

Característica de Qualidade de *Software*	Utilização no Projeto
Functional Suitability	O projeto tem como base a assinatura de documentos e facilmente se adequa a esse propósito.
Performance Efficiency	O projeto tem uma *footprint* mínima de recursos computacionais, efetuando ligação ao servidor apenas nos momentos fulcrais.

CARACTERÍSTICAS DE QUALIDADE DE *SOFTWARE*

ISO/IEC 25010 E AS OITO CARACTERÍSTICAS DE QUALIDADE DO *SOFTWARE*

Característica de Qualidade de *Software*	Utilização no Projeto
Compatibility	O projeto foi desenvolvido em Java, fornecendo uma compatibilidade de funcionar em milhares de milhões de dispositivos.
Usability	O menu feito de maneira acessível, com escolhas simples de entender e no máximo necessitando *input* fulcral, funcionando autonomamente nos outros casos.

CARACTERÍSTICAS DE QUALIDADE DE *SOFTWARE*

ISO/IEC 25010 E AS OITO CARACTERÍSTICAS DE QUALIDADE DO *SOFTWARE*

Característica de Qualidade de *Software*	Utilização no Projeto
Reliability	O programa oferece uma panóplia de *handlers* e exceções de forma a ter uma tolerância à falha correspondente à complexidade do mesmo.
Security	O projeto utiliza as *KeyStores* da API Java que fornecem uma segurança complexa de forma a proteger as informações dos certificados e das assinaturas correspondentes.

CARACTERÍSTICAS DE QUALIDADE DE *SOFTWARE*

ISO/IEC 25010 E AS OITO CARACTERÍSTICAS DE QUALIDADE DO *SOFTWARE*

Característica de Qualidade de *Software*	Utilização no Projeto
Maintainability	O projeto contém vários módulos e classes para interligar com o *web service* da AMA e também as várias etapas do projeto.
Portabilty	O projeto pode e deve ser atualizado com *updates* às dependências e atualizações aos módulos de *web servicing.*

MÉTRICAS DE QUALIDADE DE *SOFTWARE*

AS OITO MÉTRICAS DE QUALIDADE DE CÓDIGO DO *SOFTWARE*

De forma a se obter uma maneira mais sistemática de medir e qualificar as características faladas até então, mitigando toda a abordagem feita pela ISO 25000, surgem oito métricas de qualidade de código de *software:*

1. *Code coverage*
2. *Abstract interpretation*
3. *Cyclomatic complexity*
4. *Compiler warnings*
5. *Coding standards*
6. *Code duplication*
7. *Fan out*
8. *Security*

MÉTRICAS DE QUALIDADE DE *SOFTWARE*

AS OITO MÉTRICAS DE QUALIDADE DE CÓDIGO DO *SOFTWARE*

Explicação da Implementação das Oito Métricas feita aquando da demonstração do *Command Line Program*

3 COMMAND LINE PROGRAM

3.1. Estrutura do Programa
3.2. Inicialização/Utilização do Programa
3.3. Funcionamento do Programa

ESTRUTURA DO PROGRAMA

PASTAS E FICHEIROS DO *COMMAND LINE PROGRAM*

- **Diretoria doc** que possui todos os Ficheiros e Diretorias necessárias ao uso do JavaDoc gerado para o projeto
- **Diretoria src** que contém a diretoria **java** que possui toda a parte do código Java e ainda a diretoria **resources** que possui todos os ficheiros extra ao programa
  - **Diretoria java**
    - **Diretoria code**
      - Ficheiro CmdConfig.java que é usado para definir o *Application Id* fornecido pela AMA
      - Ficheiro CmdSoapMsg.java que contém todas as funções que preparam e executam os vários comandos SOAP da Assinatura com Chave Móvel Digital
      - Ficheiro TestCmdWsdl.java que contém a *main* do programa que permite que os testes dos vários comandos sejam executados
    - **Diretoria wsdlservice**
      - Ficheiros *.java que criam todo o *Web Service* necessário para executar o teste das várias operações da Assinatura com Chave Móvel Digital
  - **Diretoria resources**
    - Ficheiro ama.wsdl que define como descreve como todo o *Web Service* SOAP funciona
    - Ficheiro(s) XXXXXXXXX.pem que correspondem a todos os ficheiros PEM dos vários utilizadores do programa, usando-se como nome do ficheiro o seu *User Id*
- Ficheiro pom.xml que consiste na *Bill of Materials* do *software* desenvolvido

GitHub

README do *Command Line Program* disponível no Repositório Grupo em https://bit.ly/3hhTCQL

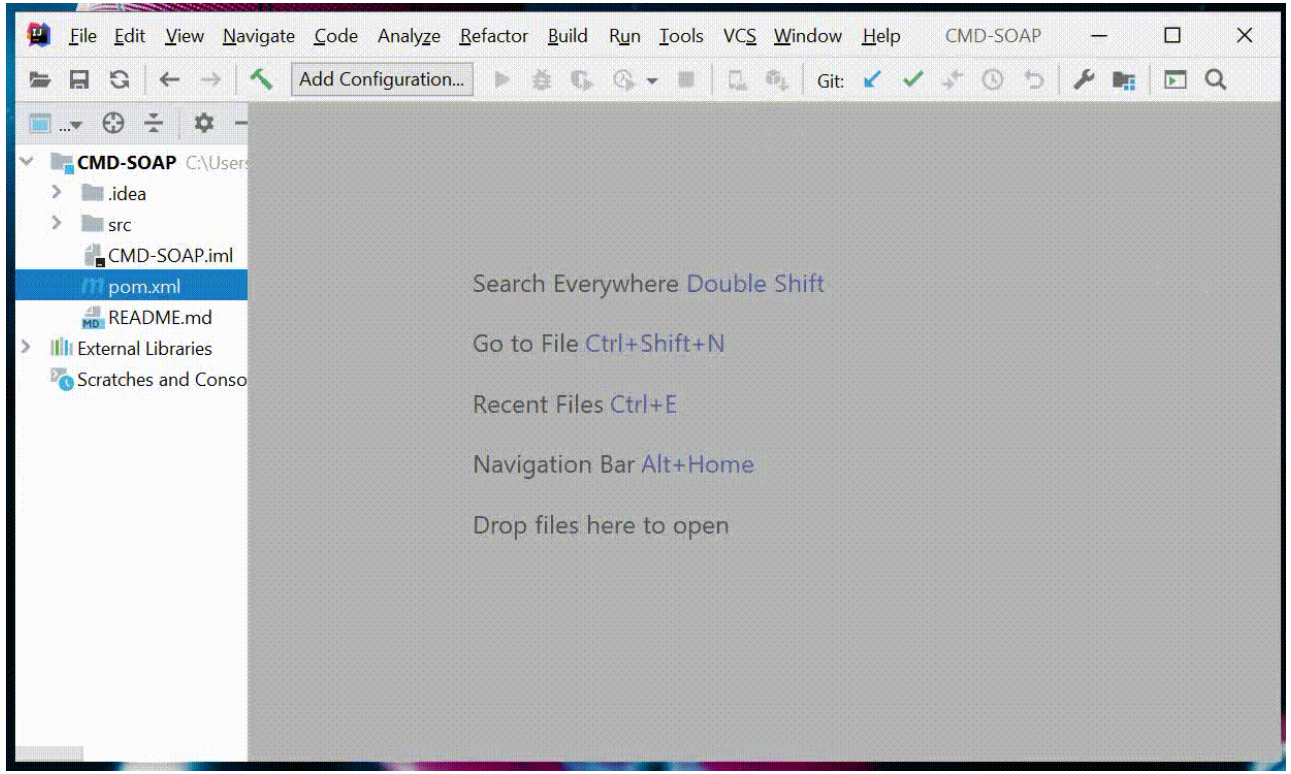
INICIALIZAÇÃO DO PROGRAMA

MODO DE INICIALIZAR O PROGRAMA ATRAVÉS DO **INTELIJ IDEA**

1 PASSO

Abrir o ficheiro pom.xml como projeto

O próprio InteliJ IDEA assume esta BOM como sendo um *project file*



2/3 PASSO

Fazer a *build* da *Bill of Materials*

As dependências essenciais ao programa serão instaladas

Fazer o *run* da Main do programa

FUNCIONAMENTO DO PROGRAMA

MODO DE UTILIZAÇÃO DO *COMMAND LINE PROGRAM*

Demonstração do Funcionamento do Programa no *Closed Source Software*