



**Docente(s)/Orientador ()**

Diogo Amaral (JDA)

**Unidade Curricular**

Desenvolvimento / Operação de Software

**Grupo G**

Bruno Fernandes [1200349]

Nelson Júnior [1191342]

Paulo Coelho [1050049]

Rui Marques [1191830]

Relatório do Projeto de DOS

Exercício final de DevOps

02/2022

**ÍNDICE**

[1. Introdução 2](#_Toc96296009)

[1. Criação API 3](#_Toc96296010)

[1.1 Estrutura 3](#_Toc96296011)

[2. Projeto 3](#_Toc96296012)

[2.1 Execução do projeto 3](#_Toc96296013)

[ANEXOS i](#_Toc96296014)

[**ANEXO A \_ Imagens** ii](#_Toc96296015)

1. **Introdução**

Este projeto consiste na criação de uma REST API, para área financeira de modo a disponibilizar os dados de transações de uma conta bancária.

Para alcançar este objetivo utilizamos os diversos temas abordados durante o decorrer do semestre, como por exemplo:

* Implementação de solução em .Net Core;
* Controlo de Versões;
* Testes Unitários em Xunit;
* Aplicação de documentação automática (Swagger);
* Utilização de Containers;
* Utilização de ferramentas de Continuous Integration e Continuous Delivery;
* Utilização de ferramentas de provisionamento, gestão de configurações e orquestração;

O funcionamento desta aplicação será exemplificado no decorrer deste documento.

# Criação API

## 1.1 Estrutura

Começamos pela criação de uma solução ASP C# .Net 5.0, onde fizemos toda a estrutura de dados e classes solicitadas no enunciado do projeto permitindo que uma conta bancária será associada unicamente a um cliente, permitindo esse cliente ter várias contas bancárias.

A nossa ideia inicial foi automatizar o processo ao máximo e para isso pensamos em colocar os volumes do jenkins e do sonar no repositório. A solução funcionava no entanto ocupava muito espaço no repositório e criou alguns problemas pelo que decidimos que as configurações inicias que são fáceis e rápidas fossem efetuadas pelo utilizador.

# Projeto

## Execução do projeto

Para executar o projeto na pasta Docker, onde temos um ficheiro Docker-compose.yml abrimos a linha de comandos e corremos o comando Docker-compose up –build. Se tiver problemas com o ElasticSearch por causa da memoria podemos antes usar o comando wsl -d docker-desktop e depois sysctl -w vm.max\_map\_count=262144. Os containers do Jenkins, Sonarqube e postgres serão então criados.

Uma imagem com texto

Descrição gerada automaticamente

Figura 1: Docker composse up --build

Quando tivermos os containers a correr podemos aceder ao sonarqube através do endereço <http://localhost:9000> e ao jenkins <http://localhost:8080>.

Neste caso o jenkins já se encontra configurado com todos os plugins necessários.

Vamos primeiro criar um token no SonarQube para podermos usar no Jenkins.

Para aceder ao sonarqube insira a user: admin e pass:admin.

Uma imagem com texto

Descrição gerada automaticamente

Figura 2 - Sonar my account

Na aba security criamos um token com um nome à nossa escolha.

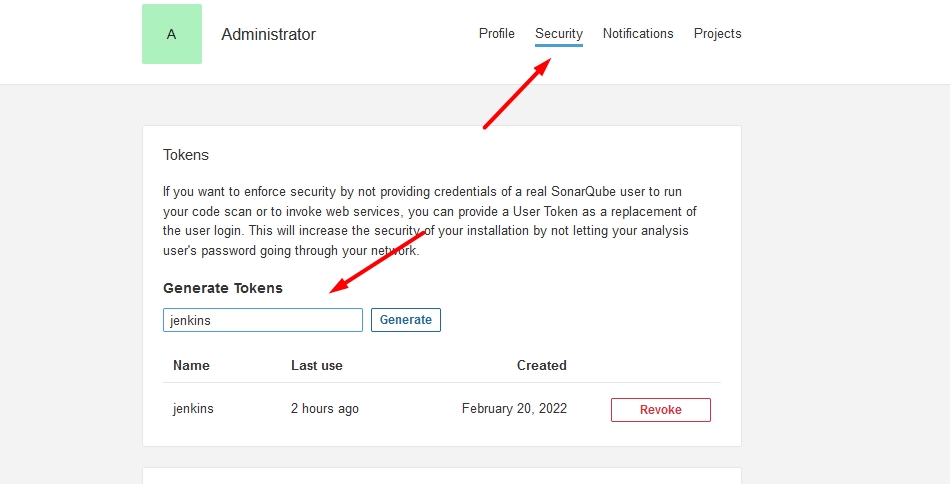


Figura 3 - Sonar token

Podemos ainda adicionar um webhook para podermos usar com o quality gate do jenkins.

Administration > configuration > webhooks e clique em create.

Uma imagem com texto

Descrição gerada automaticamente

O nome podemos escolher e o URL é o nome do nosso container de Jenkins.

Uma imagem com texto

Descrição gerada automaticamente

Agora vamos ao Jenkins, aqui precisamos de adicionar as credenciais para o Git e para o Sonar.

Adicionamos credenciais em gerir jenkins > manage credentials. Escolhemos secret texto e adicionamos o token gerado no Sonar, o ID e a descrição convem ser descritivo. Para o git usamos Username e password com a nossa identificação.

Uma imagem com texto

Descrição gerada automaticamente

Figura 4 - Add credentials

De seguida configuramos o servidor de scanner em gerir jenkins > configurar sistema > SonarQube Servers . O nome é sonar, que é usado no nosso Jenkinsfile e o URL é <http://sonar:9000> e a credencial que criamos para o sonar.Por fim vamos configurar o scanner MS em global Tool Configuration em SonarScanner for MSBuild, o nome pode ser ms e o MSBUILD\_SQ\_SCANNER\_HOME = /root/.dotnet/tools/ se tiver para instalar automaticamente tiramos pois já se encontra instalado.

Uma imagem com texto

Descrição gerada automaticamente

Figura 5 - Jenkins Conf

Podemos agora criar uma pipeline em novo item > pipeline e um nome à nossa escolha. Aqui podemos ir no fim da página e copiar o script do projeto “Script pipeline” ou utilizar o jenkins file. Para utilizar o jenkins file devemos escolher a opcao pipeline script from SCM, escolher git com a nossa autenticação e escolher o caminho do Jenkinsfile no repositório. Por fim é só dar build.

Uma imagem com texto

Descrição gerada automaticamente

Figura 6: Pipeline

Para subir a nossa aplicação na pasta PROD dentro da pasta Docker, podemos correr o comando Docker-compose up –build, que vai levantar a nossa aplicação e o sql server. A instância de sql irá criar automaticamente a base de dados banco e se pretendido podemos no visual studio -> ferramentas -> Gerenciador de pacotes nuget-> console e correr o comando Update-database para inserir conteúdo na base de dados (correr as migrations).

De momento o Docker está instalado na nossa máquina do Jenkins, e podemos entrar na máquina com este comando docker exec -it jenkins bash e podemos ver a versão com o comando docker-compose –version. No entanto tentamos colocar na pipeline o seguinte stage, mas o Docker composse não era encontrado.

Uma imagem com texto

Descrição gerada automaticamente

Figura 7 - stage docker

# **ANEXOS**

**ANEXOS**

## **ANEXO A \_ Imagens**

[Figura 1: Docker composse up --build 3](#_Toc96296001)

[Figura 2 - Sonar my account 4](#_Toc96296002)

[Figura 3 - Sonar token 4](#_Toc96296003)

[Figura 4 - Add credentials 5](#_Toc96296004)

[Figura 5 - Jenkins Conf 6](#_Toc96296005)

[Figura 6: Pipeline 6](#_Toc96296006)

[Figura 7 - stage docker 7](#_Toc96296007)