Voer volgende stappen uit om een werkende Continuous Integration & Continuous Delivery pipeline te verkrijgen. De volgorde is uiterst belangrijk om tot een goed resultaat te komen.

B.1 Installatie Ubuntu server

Hier maken we de Ubuntu 18.04 server klaar om gebruik te maken als Jenkins server

1. Typ **ssh root@{ip-adress van server}** van de server in de console om in te loggen in de server
2. Typ nadien het paswoord van de server. Er wordt meteen gevraagd om het te vernieuwen. Kies een nieuw paswoord.
3. Eens ingelogd op de sshd server moeten er enkele zaken aangepast worden aan de ssh configuratie in de file '/etc/ssh/sshd_config'. Open de file (via cat command) en zet de PermitRootLogin op **yes** zodat de root-gebruiker kan inloggen.
4. Zet StrictMode ook op **yes** zodat niemand kan inloggen als de authenticatie documenten leesbaar zijn voor iedereen.
5. Voor het verdere wordt verwacht dat er een SSH is opgezet op de client computer waarmee gewerkt wordt. Indien dit nog niet gebeurd is, neem een kijkje op: https://www.ssh.com/ssh/key/
6. Kopieer de SSH key van de client computer naar de server door op de client computer een console te openen, naar de root folder te gaan (**cd ~**) en **ssh-copy-id root@{ip-adress van server}** te typen
7. Ga terug naar de file '/etc/ssh/sshd_config' op de sshd server
8. Verander PasswordAuthentication en ChallengeResponseAuthentication naar **no** en PubKeyAuthentication naar **yes**. Dit zorgt ervoor dat het niet meer mogelijk is om via een paswoord in te loggen, enkel via ssh.
9. Sla de 'sshd_config' file op
10. Herstart de ssh daemon door **sudo systemctl restart ssh** te typen in de concole van de sshd server
11. Zet nu de UFW firewall op in de sshd server door **ufw app list** te typen.
    Je ziet dat OpenSSH beschikbaar is als applicatie
12. **Ufw allow OpenSSH** zorgt ervoor dat de firewall wordt opgezet
13. Om gebruik te maken van de firewall, typ **ufw enable** en kies voor **yes**
14. Om te controleren of alles daadwerkelijk gelukt is moet je **ufw status** typen

B.2 Source Control

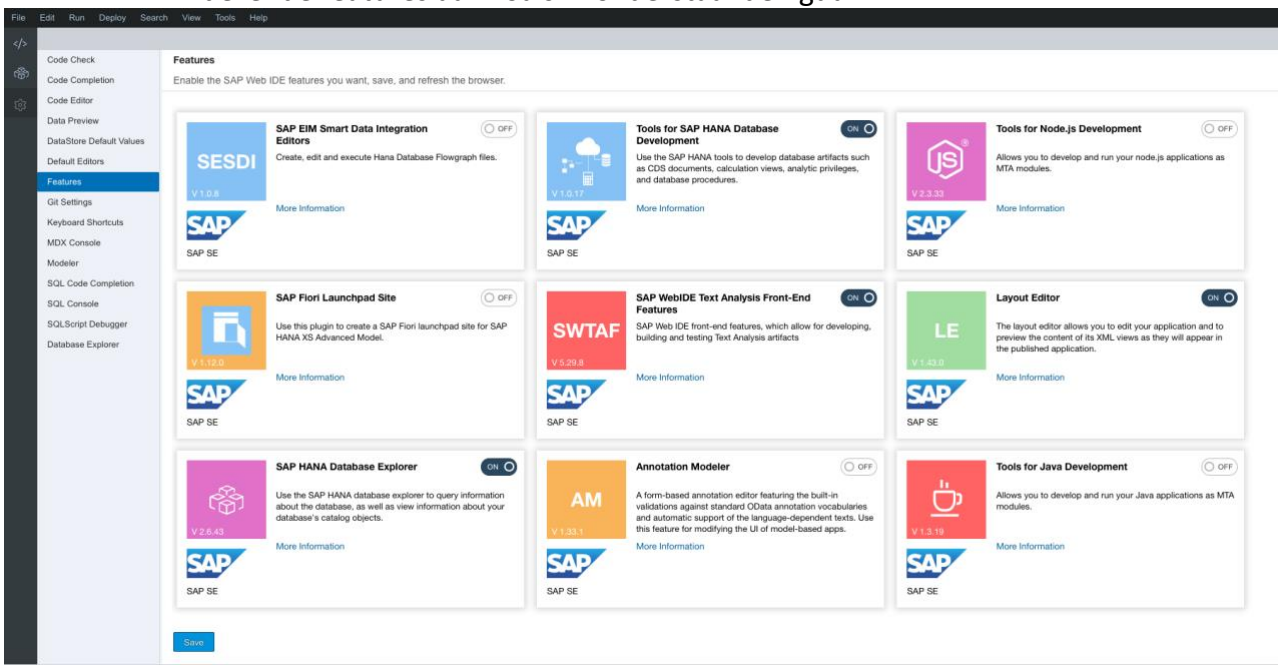
Maak twee nieuwe repositories aan in BitBucket: ééntje voor de HANA 2.0 applicatie en de andere voor de SAPUI5 applicatie:

1. Kies een gepaste naam
2. Geef een gepaste description in
3. Vink ‘This is a private repository’ uit en kies voor Git.
4. Kopieer het adres dat gegeven wordt om een clone te maken van deze repositories

B.3 SAP applicaties

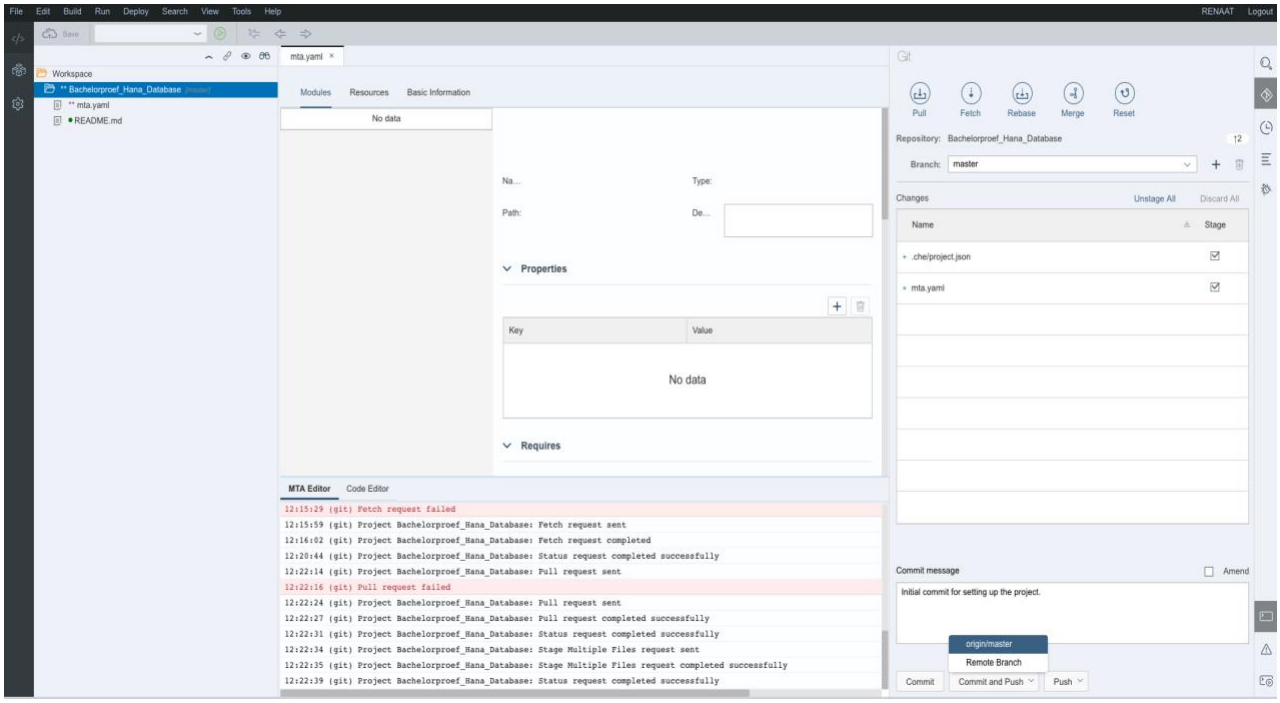
B.3.1 Database (HANA 2.0)

1. Ga naar de Web IDE voor HANA development
2. Maak een nieuw project en kies voor de Multi-Target Application Project template
3. Ga naar de instellingen en kies voor het onderdeel ‘Features’
4. Vink dezelfde features aan zoals in onderstaande figuur



6. Ga naar de Git Settings en voeg daar je gegevens in zodat er verbinding gemaakt kan worden
7. Ga terug naar de Development omgeving, klik met je rechtermuisknop op het project en kies voor Build
8. Om het project aan het account op Bitbucket te linken klikt u op het project met de rechtermuisknop, gaat u naar 'Git' en dan 'Initialize Local Repository'
9. Link de gemaakte repository aan het project door via rechter muisklik op het project op Git en dan Set Remote te klikken
10. Plak de gekopieerde Bitbucket link in het veld voor URL
11. Geef je wachtwoord van Bitbucket in
12. Klik OK wanneer het 'Changes Fetched' window opent

13. Commit de changes naar de repository zoals in onderstaande figuur



B.3.1.1 Database Module

1. Maak een HANA Database Module door op het project met de rechtermuisknop te klikken, bij ‘New’ voor ‘SAP HANA Database Module’ te kiezen
2. Geef de module de naam: **db**
3. Kies **Bachelor_Hana_Database.db** als Namespace, **Voorbeeldapplicatie** als Schema Name en **2.0 SPS 03** als SAP Hana Database Version
4. Klik op de src map in de db module met de rechtermuisknop en kies voor ‘New’ en nadien voor ‘HDB CDS Artifact’
5. Geef **data** in als naam

6. Open de gemaakte file met de Code Editor en voeg volgende data in

```
namespace Bachelorproef_Hana_Database.db.data;

context data {

  /*@@layout{"layoutInfo":{"x":-467,"y":-239.5}}*/
  entity Artiest {
    key Id            : Integer generated by default as identity(start with 1 increment by 1 no
minvalue maxvalue 2999999999 no cache no cycle);
      Naam            : String(256);
      JaarVanOorsprong : Integer;
  }
  technical configuration {
    column store;
  };
};
```

data.hdbcds

7. Maak de database door in de meest linkse tab op de knop 'Database Explorer' te klikken.
8. Klik op het +-teken om een database aan te maken. Kies uit de lijst de optie met jouw naam en geef een realistische naam bij het veld 'How to Show in Display'. Laat de rest van de setting zoals ze zijn.
9. Ga terug naar 'Development' omgeving (in meest linkse tab)
10. Klik met de rechter muisknop op de 'data.hdbcds' file kies de optie 'Build Selected Files'.

11. Er moet natuurlijk ook data in de database zitten om deze te gebruiken in de SAPUI5 applicatie. Een manier om dit te doen is via het uploaden van csv-bestanden waar de data gescheiden is door een komma. Maak een nieuwe file, 'load.hdbtabledata' genaamd, in de data map van de db-module en voeg volgende inhoud toe:

```
{
"format_version": 1,
"imports": [{
  "target_table": "Bachelorproef_Hana_Database.db.data::data.Artiest",
  "source_data": {
    "data_type": "CSV",
    "file_name": "Bachelorproef_Hana_Database.db.data::Artiest.csv",
    "has_header": false,
    "dialect": "HANA",
    "type_config": {
      "delimiter": ","
    }
  },
  "import_settings": {
    "import_columns": ["Id",
    "Naam",
    "JaarVanOorsprong"]
  },
  "column_mappings": {
    "Id": 1,
    "Naam": 2,
    "JaarVanOorsprong": 3
  }
}]
}
```

load.hdbtabledata

12. Maak een csv-bestand aan en voeg volgende data toe

```
1,Nirvana,1989
2,Steak Number Eight,2006
3,Placebo,1995
4,Tool,1992
5,Shame,2016
6,Queens Of The Stone Age,1996
7,Madensuyu,2005
8,Kyuss,1990
```

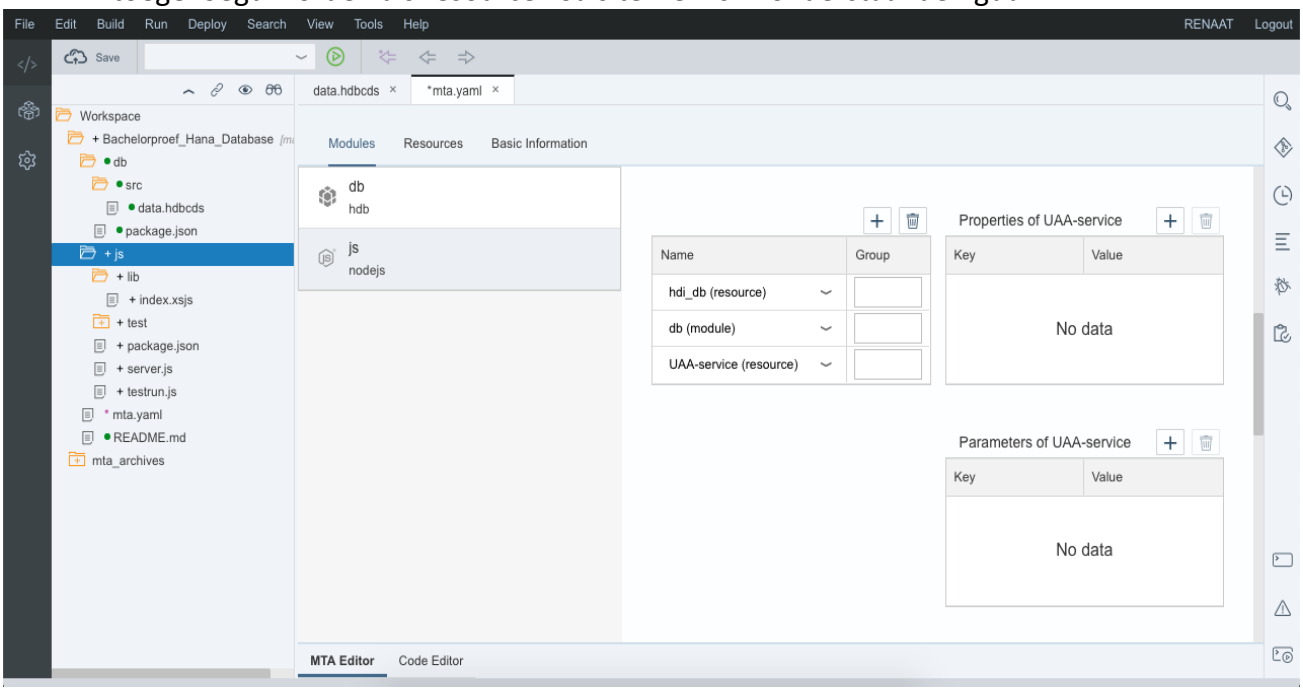
Artiest.csv

13. Importeer de gemaakt file in de map ‘data’ van de ‘db-module’
14. Build de ‘db-module’ nogmaals
15. Controleer of er effectief data aanwezig is door naar de 'Database Explorer' te gaan
16. Open de map 'tables'
17. Links vanonder verschijnt een venster met de verschillende tabellen. Er rest ons enkel nog op de juiste tabel te klikken met de rechter muisknop en kies dan 'Generate SELECT statement'.
18. Klik op de groene play knop om het statement uit te voeren
19. Deze veranderingen kunnen best gepusht worden naar Bitbucket

B.3.1.2 Node.js module

Maak een Node.js module om de data te kunnen exporteren als een OData service

1. Ga terug naar de instellingen van de Features en zorg ervoor dat de feature ‘Tools For Node.js Development’ beschikbaar is
2. Save de instellingen en laad het project opnieuw
3. De volgende stap is de module maken door op het project met de rechtermuisknop te klikken, new en dan Node.js Module kiezen. Geef het een goede naam (js is good practice) en zorg ervoor dat Enable XSJX support aangevinkt is alvorens op Finish te klikken
4. Databeveiliging is zeer belangrijk. Binnen HANA wordt er gebruik gemaakt van UAA (User Account en Authentication) om de Node.js module te beveiligen. Deze service moet samen met de database module en de HDI container, achter de db module, toegevoegd worden als resource zoals te zien is in onderstaande figuur



5. Maak in de ‘js-module’ in de ‘lib’ folder een nieuwe file en geef het volgende naam: **xsodata/services.xsodata**
6. Kopieer volgende code in deze file

```
service {
        "Bachelorproef_Hana_Database.db.data::data.Artiest" as "Artiesten";
}
```

services.xsodata

7. Maak een xsjs service door in de ‘lib’ folder een nieuwe file aan te maken met volgende naam: **xsjs/hdb.xsjs**

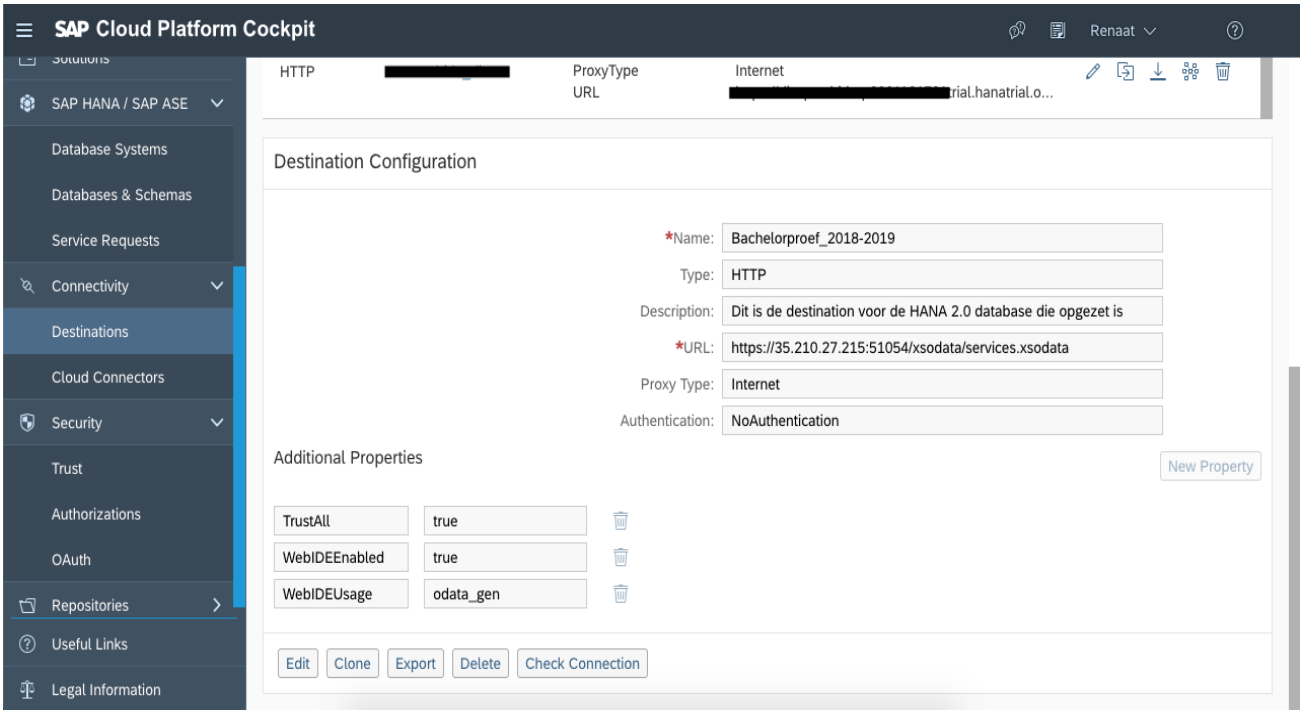
8. Kopieer de code die u hier ziet in deze file

```
/*eslint no-console: 0, no-unused-vars: 0, dot-notation: 0*/
/*eslint-env node, es6 */
"use strict";

var conn = $.hdb.getConnection();
var query = "SELECT FROM Bachelorproef_Hana_Database.db.data::data.Artiest { " +
        " Id as \"ArtiestId\", " +
        " Naam as \"ArtiestenNaam\", " +
        " JaarVanOorsprong as \"Jaar\" " +
        " } ";
var rs = conn.executeQuery(query);

var body = "";
for (var item of rs) {
        body += item.ArtiestenNaam + "\t" + item.Jaar + "\n";
}

$.response.setBody(body);
$.response.contentType = "application/vnd.ms-excel; charset=utf-16le";
$.response.headers.set("Content-Disposition",
        "attachment; filename=Excel.xls");
$.response.status = $.net.http.OK;
```

hdb.xsjs

9. Build de nieuwe Node.js module door op de module met de rechtermuisknop te klikken, Run, Run As en dan Node.js Application te kiezen
10. Klik onderaan het scherm in het nieuwe venster op de link
11. De index.html wordt geopend in de browser. Het is mogelijk om de metadata te bekijken door in de url de '/index.html' te vervangen door '/xsodata/services.xsodata/$metadata'
12. Om de OData van de entiteit in de database te bekijken moet '/$metadata' vervangen worden door '/Artiesten'. Op deze manier verschijnt de OData van de tabel Artiest
13. Als voorgaande stappen gelukt zijn, kan men de OData-service gebruiken als een destination in SAP Cloud Platform. Om dit te doen zijn er enkele stappen nodig: ga naar SAP Cloud Platform

14. Ga naar destinations en voeg een nieuwe toe met de gegevens zoals in onderstaande figuur

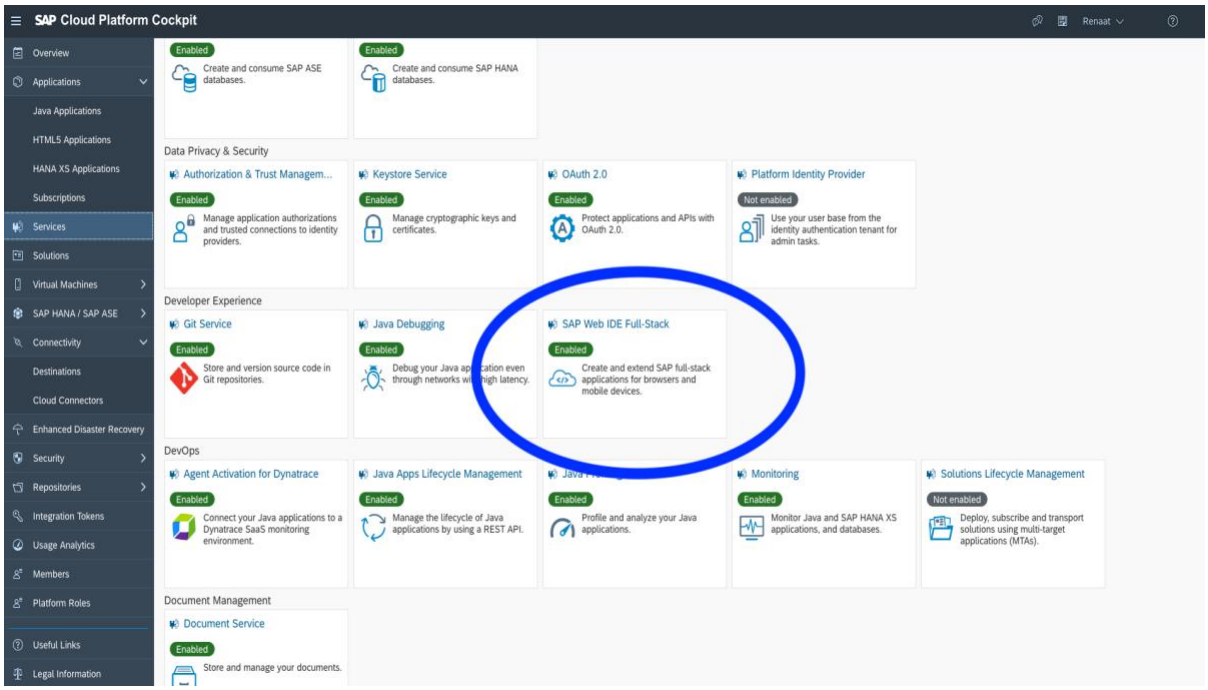


Om de code uit deze handleiding te bekijken kan je volgende repository clonen op je lokale machine:
https://RenaatHaleydtAmista@bitbucket.org/RenaatHaleydtAmista/bachelorproef_hana_database.git

B.3.2 UI5 applicatie

Maak een nieuwe applicatie, startend vanaf een template:

1. Ga terug naar de SAP Cloud Platform cockpit
2. Ga bij Services (te vinden in de uiterst linkse kolom) en zoek voor SAP Web IDE Full-Stack



3. Kies onderaan voor Go to Service
4. Kies in de uiterst linkse kolom voor Settings en ga naar de optie Git Committer
5. Geef je gegevens in om met BitBucket te configureren
6. Ga terug naar de Development omgeving door in de linkse kolom op de knop te drukken met </> als teken
7. Eens het venster geopend wordt klik je op de folder Workspace
8. Ga dan naar de top menubalk en kies voor File -> Git -> Clone Repository
9. Geef de juiste url in van de repository die je zonet gemaakt hebt in BitBucket voor de SAPUI5 applicatie
10. Geef het paswoord in van BitBucket wanneer hier om gevraagd wordt
11. Klik op de nieuwe folder die aangemaakt is en merk op dat deze dezelfde naam heeft als de Repository op BitBucket
12. Ga naar File -> New -> Project from Template
13. Kies SAPUI5 Application en geef het een correcte naam (Bachelorproef_2018-2019)
14. Kies XML als formaat voor de eerste view en geef het een juiste naam (App bijvoorbeeld)
15. Klik op finish
16. Push deze gegevens naar BitBucket als initiele commit

17. In de neo-app.json file, voeg volgende code toe aan de routes array:

```
{
        "path": "/xsodata/services.csodata ",
        "target": {
              "type": "destination",
              "name": "Bachelorproef_2018-2019"
        },
        "description": "Bachelorproef_2018-2019"
    }
```

neo-app.json

18. Voeg volgende code toe aan het 'sap.app' object in manifest.json

```
"dataSources": {
                    "mainService": {
                          "uri": "/xsodata/services.xsodata/",
                          "type": "OData",
                          "settings":{
                                "odataVersion": "2.0"
                          }
                    }
              }
```

manifest.json

19. In de manifest.json moet ook nog het default model toegevoegd worden aan het object 'models'

```
"" : { "dataSource": "mainService" }
```

manifest.json

20. Verander de code in App.view.xml door volgende code

```
<mvc:View  controllerName="Bachelorproef_2018-2019.Bachelorproef_2018-
2019.controller.App"
        xmlns:mvc="sap.ui.core.mvc" displayBlock="true” xmlns="sap.m"
        xmlns:core="sap.ui.core" xmlns:l="sap.ui.layout" xmlns:f="sap.ui.layout.form">
        <Shell id="shell">
        <App id="app">
                <pages>
                        <Page id="page" title="{i18n>title}">
                                <content>
                                        <IconTabBar id="idTopLevelIconTabBar"
                                        class="sapUiResponsiveContentPadding">
                                        <items>
                                        <IconTabFilter  text="Data Binding" key="db">
                                                <List headerText="Artiesten"
                                                width="auto" items="{/Artiesten}"
                                                mode='SingleSelectMaster'
                                                selectionChange="onItemSelected">
                                                <items>
                                                        <ObjectListItem title="{Naam}"
                                                        intro="{JaarVanOorsprong}">
                                                        </ObjectListItem>
                                                </items>
                                                </List>
                                        </IconTabFilter>
                                        <IconTabFilter>
                                                <l:VerticalLayout
                                                class="sapUiContentPadding" width="100%">
                                                <l:content>
                                                        <MessageStrip text="Als voorbeeld"
                                                        showIcon="true"
                                                        showCloseButton="true"
                                                        class="sapUiMediumMarginBottom">
                                                        </MessageStrip>
                                                </l:content>
                                                </l:VerticalLayout>
                                        </IconTabFilter>
                                        </items>
                                </IconTabBar>
                                </content>
                        </Page>
                </pages>
        </App>
        </Shell>
</mvc:View>
```

App.view.xml

21. Wanneer op een item geklikt wordt in de lijst verschijnt er een Message Toast met de leeftijd van de Artiest. Deze wordt berekend in de AppController zoals u hier kan zien.

```
sap.ui.define([
	'sap/m/MessageToast',
	"sap/ui/core/mvc/Controller"
], function (MessageToast, Controller) {
	"use strict";

	return Controller.extend("Bachelorproef_2018-2019.Bachelorproef_2018-
2019.controller.App", {
		onInit: function() {
		},
		onItemSelected: function(oEvent) {
			var oSelectedItem = oEvent.getParameter("listItem");

			var msg = oSelectedItem.getTitle() + " is " +
this.calculateYearFromNow(oSelectedItem.getIntro()) + " jaar oud!";
			MessageToast.show(msg);
		},

		calculateYearFromNow: function(jaar) {
			return (new Date().getFullYear() - jaar);
		}
```

App.controller.js

22. U kan nu de app runnen en testen
23. Eens u zeker bent dat de app voldoende werkt moeten de wijzigingen worden overgebracht naar BitBucket

B.3.2.1 Testen

B.3.2.1.1 QUnit

1. Kopieer volgende code in de App.controller.js in de unit folder dat zich in de test folder bevindt

```
/*global QUnit*/

sap.ui.define([
	"Bachelorproef_2018-2019/Bachelorproef_2018-2019/controller/App.controller"
], function (Controller) {
	"use strict";

	QUnit.module("App Controller");

	QUnit.test("I should test the App controller", function (assert) {
		var oAppController = new Controller();
		oAppController.onInit();
		assert.ok(oAppController);
	});

	QUnit.test("Test of calculate the year method", function(assert) {
		// arrangement
		var oController = new Controller();

		// action

		// assertions
		assert.equal(oController.calculateYearFromNow(2017), 2);
	});

});
```

App.controller.js (in test folder)

2. In de test folder zelf moet volgende file aangemaakt worden: QUnit.js . Plak in de nieuwe file de inhoud uit volgende link: https://docs.google.com/document/d/1h6VsETEXU76xO1c_0ybMannID0W3X5xfJBhqhK64WIs/edit?usp=sharing. Dit bestand is aangemaakt omdat het te groot zou zijn om in deze proof-of-concept weer te geven.
3. Terug in de unit folder moet de inhoud van unitTests.qunit.html vervangen worden door deze:

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="utf-8">
	<title>Unit tests for Bachelorproef_2018-2019.Bachelorproef_2018-
2019</title>

	<script id="sap-ui-bootstrap"
		src="https://sapui5.hana.ondemand.com/resources/sap-ui-core.js"
		data-sap-ui-resourceroots='{
			"Bachelorproef_2018-2019/Bachelorproef_2018-2019":
"../../"
		}'>
	</script>

	<link rel="stylesheet" type="text/css"
href="../../../resources/sap/ui/thirdparty/qunit-2.css">

	<script src="../../../resources/sap/ui/thirdparty/qunit-2.js"></script>
	<script src="../../../resources/sap/ui/qunit/qunit-junit.js"></script>
	<script src="../../../resources/sap/ui/qunit/qunit-coverage.js"></script>

	<script src="../QUnit.js"></script>
	<script src="unitTests.qunit.js"></script>
</head>
<body>
	<div id="qunit"></div>
	<div id="qunit-fixture"></div>
</body>
</html>
```

unitTests.qunit.html

4. Klik dan in de unit folder op unitTest.qunit.html met de rechtermuisknop en kies voor: Run As -> Run As Unit Test
5. In het nieuwe tabblad dat automatisch opent zou je de twee testen moeten zien die groen kleuren.

B3.2.1.2 OPA5

1. In de folder integration zie je de file opaTests.qunit.html, de inhoud moet vervangen worden door het volgende:

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="utf-8">
	<title>Integration tests for Basic Template</title>

	<script id="sap-ui-bootstrap"
		src="https://sapui5.hana.ondemand.com/resources/sap-ui-
core.js"
		data-sap-ui-theme='sap_belize'
		data-sap-ui-resourceroots='{
			"Bachelorproef_2018-2019.Bachelorproef_2018-2019":
"../../"
		}'
		data-sap-ui-animation="false"
		data-sap-ui-compatVersion="edge"
		data-sap-ui-async="true">
	</script>

	<link rel="stylesheet" type="text/css"
href="../../../resources/sap/ui/thirdparty/qunit-2.css">

	<script src="../../../resources/sap/ui/thirdparty/qunit-2.js"></script>
	<script src="../../../resources/sap/ui/qunit/qunit-junit.js"></script>

	<script src="../QUnit.js"></script>
	<script src="opaTests.qunit.js"></script>
</head>
<body>
	<div id="qunit"></div>
	<div id="qunit-fixture"></div>
</body>
</html>
```

opaTests.qunit.html

2. We maken hier gebruik van de default OPA5 test, enkel om de werking met Karma te tonen
3. Om de OPA5 test te runnen is het voldoende om op de opaTests.qunit.html met de rechtermuisknop te klikken en te kiezen voor Run -> Run as Web Application

B.4 Proof-of-concept

B.4.1 Jenkins installeren en front-end aanspreken

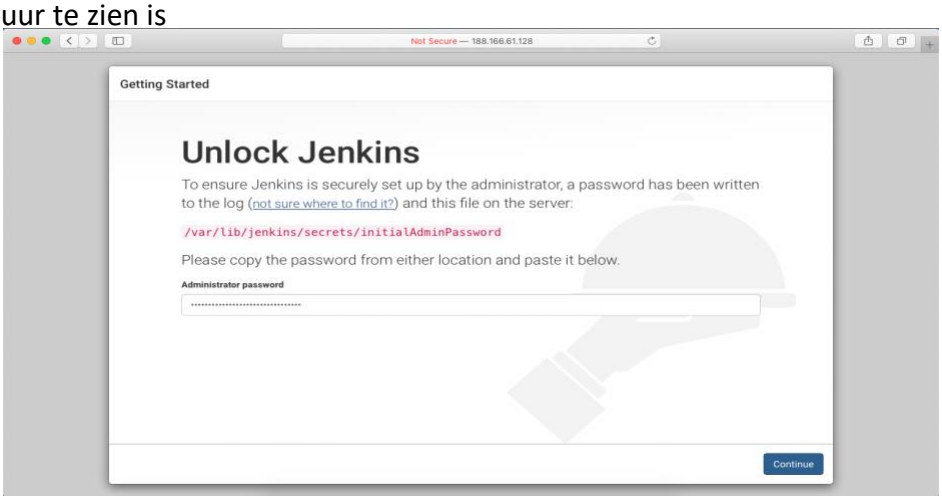
Momenteel hebben we draaiende applicaties en is het de beurt om Jenkins te installeren en te runnen op onze server.

1. Omdat Jenkins op Java draait is het nodig om Java 8 te installeren. Dit is dan ook de eerste stap in het proces.
2. Log in de server door **ssh root@188.166.61.128** in de console op jouw computer te typen
3. Voer het commando **sudo apt install openjdk-8-jdk** intypen om JAVA 8 te installeren. De server vraagt bevestiging om te downloaden en te installeren, er moet enkel maar **y** worden getypt en op return gedrukt worden om te bevestigen
4. Nu kan worden overgegaan tot de installatie van Jenkins op de server. De packages die Jenkins onderhoud om de software te installeren op Ubuntu worden gebruikt door **wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -** in te typen
5. Eens dit gebeurd is wordt de Debian package repository toegevoegd aan de server's sources.list door **sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'** te typen
6. Typ **sudo apt update**
7. Installeer Jenkins door **sudo apt install Jenkins** in te typen

Maak gebruik van Jenkins

8. Typ **sudo systemctl start jenkins** in
9. Wanneer na het invoeren van commando **sudo systemctl status jenkins** active verschijnt wil dit zeggen dat Jenkins runt op de Ubuntu server
10. Eens Jenkins gestart is moet deze kunnen worden aangesproken vanuit een browser, daarom moeten de regels van de firewall wat worden aangepast. Standaard draait Jenkins op poort 8080. Deze moet geopend worden op de server door het commando **sudo ufw allow 8080** in te voeren
11. Om Jenkins in de browser te configureren moet je naar http://188.166.61.128:8080 surfen. De cijfers voor de :8080 staan voor het ip-adres van de Ubuntu server, de :8080 is de poort waar je naartoe wil gaan. Hier draait - zoals in de vorige stappen geconfigureerd - Jenkins op. Als we naar dit ip-adress surfen spreken we de Jenkins front-end aan

12. Het paswoord dat gevraagd wordt is terug te vinden in de 'initialAdminPassword' file dat zich in de map '/var/lib/jenkins/secrets/' bevindt op de server. Het is voldoende om op de server volgend commando in te voeren: **sudo cat /var/lib/jenkins/secrets/initialAdminPassword**. Het paswoord dat als uitvoer komt moet gekopieerd worden en geplakt worden in het invoerveld zoals in onderstaande figuur te zien is



13. Klik op ‘Install suggested plugins’ om de plugins te installeren
14. Klik op ‘Continue as admin’ om door te gaan als Admin user
15. Klik op ‘Save and Finish' wanneer je de url gecontroleerd hebt op zijn juistheid
16. Nadien kies je voor 'Start using Jenkins'
17. Het dashboard van Jenkins wordt geopend

B.4.2 Maken van pipeline 2.0 en configuratie met BitBucket

1. Log in op de webpagina van Jenkins die opgestart is op de server
2. Ga naar Manage Jenkins in de linkse kolom
3. Klik op Manage Plugins
4. Klik op Available
5. Zoek naar Bitbucket
6. Installeer ‘Bitbucket Plugin’, ‘Bitbucket Pipeline for Blue Ocean’, ‘GitHub Branch Source Plugin’
7. Ga naar de console op de server en typ **sudo systemctl restart jenkins**
8. Ga naar de BitBucket repository van de UI5 applicatie
9. Klik op Settings en kies bij WorkFlow voor Webhooks
10. Klik op de knop Add webhook en vul onderstaande gegevens in:
    a. Title: Jenkins
    b. URL: {JenkinsURL}/bitbucket-hook/ (in dit geval is dat: http://188.166.61.128:8080/bitbucket-hook/)
11. Kies voor Repository push als Trigger en sla de settings op
12. Ga terug naar Jenkins en kies ervoor om een nieuw item toe te voegen
13. Voeg een gepaste titel in, bijvoorbeeld bachelorProef_pipeline en kies voor Pipeline
14. Bij Build Triggers vink je de optie: Build when a change is pushed to BitBucket aan
15. Onder Pipeline, bij Definition kies je voor de optie Pipeline script from SCM
16. Bij SCM kies je voor Git
17. Bij het veld Repository URL voer je de url naar de SAPUI5 BitBucket repository in
18. Bij Credentials kies je om een nieuwe credential toe te voegen
19. Vul bij Username en Password de gegevens van je BitBucket account in

20. Zorg dat bij Branches to build */master is ingevuld
21. Kies bij Repository browser bitbucketweb uit de lijst
22. Bij Build Triggers kies je voor de optie: Build when a change is pushed to BitBucket. Je hoeft geen gegevens in te vullen bij de URL
23. Bij Script Path moet er Jenkinsfile komen te staan. Dit is default ingevuld, maar best te controleren
24. Druk op Apply en nadien op Save om alles op te slaan

B.4.3 Installeren van Node.js, npm en nodige packages op Ubuntu 18.04

1. Eens ingelogd op de server typ je **sudo apt update**
2. Nadien **sudo apt install nodejs**
3. Om npm te installeren is het voldoende om **sudo apt install npm** te typen in de console
4. Na het installeren van Node.js en npm is het nodig om alle gebruikte packages te installeren voor gebruik van Karma
1. Voor we gebruik kunnen maken van UI5 op onze server moet dit uiteraard eerst geïnstalleerd worden. Dit doe je door **npm install –global @ui5/cli** te typen
2. Installeer Karma door **npm install karma –save-dev** in de console van de server te typen
3. **npm install karma-chrome-launcher karma-ui5 –save-dev** moet ook getypt worden
4. Het is makkelijker om de Karma command line te installeren, deze kan gebruikt worden in de karma.conf.js. Dit doe je door **npm install –g karma-cli** in te typen. Nu is het mogelijk om van overal op de server Karma te runnen door simpelweg karma in te typen.
5. Het kan zijn dat je gebruik moet maken om de Chrome driver te installeren. Deze wordt gebruikt om Karma te kunnen runnen. Typ **sudo nano /etc/apt/sources.list.d/google-chrome.list** in de console
6. In de geopende file moet je volgende content plakken:

```
deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main
```

/etc/apt/sources.list.d/google-chrome.list

7. Druk ctrl+o en nadien op Enter om te bevestigen. Daarna moet je de combinatie ctrl+x indrukken om af te sluiten
8. Download Google's signing key door volgend commando in te typen: **wget https://dl.google.com/linux/linux_signing_key.pub**
9. Nu moet je de package manager aanspreken om de key toe te voegen door **sudo apt-key add linux_signing_key.pub** te typen
10. Update pachame manager: **sudo apt update**
11. Installeer de laatste stabiele versie van Google Chrome: **sudo apt install google-chrom-stable**

B.4.4 Configureer het UI5 project en Jenkins voor een Jenkins 2.0 pipeline

We maken in deze handleiding gebruik van de Jenkins 2.0 pipeline. Omdat het tal van voordelen met zich meebrengt en vrij makkelijk is om op te zetten. Verdere uitleg kan u terugvinden in de bachelorproef. In onderstaande stappen maken we ons project klaar voor configuratie met Jenkins 2.0 en gebruik van de SAP/jenkins-library.
De eerste stages die we toepassen is het aanspreken van Karma om de Qunit en OPA5 testen automatisch te runnen.

B.4.4.1 Jenkins library in front-end

1. We willen gebruik maken van de jenkins-library die SAP aanbiedt. Ga naar de front-end van Jenkins (http://188.166.61.128:8080/)
2. Kies links in de menu voor Manage Jenkins -> Configure System
3. In de sectie Global Pipeline Libraries klik je op de Add button
4. Naast Library Name vul je piper-lib-os in
5. Bij Default Version vul je master in. Dit zorgt ervoor dat je telkens de laatste versie van deze library gebruik. Dit is de veiligste optie.
6. Kies Modern SCM bij Retrieval Method
7. Kies Git bij Source Code Management en plak https://github.com/SAP/jenkins-library in het veld naast Project Repository
8. Nu kunnen we gebruik maken in de Jenkinsfile van deze library

B.4.4.2 Configureer het SAPUI5 project met Jenkins 2.0

1. Download de SAPUI5 applictatie repository op je lokale machine.
   Ga eerst naar een map waar je de repository in wil bewaren. Kloon de repository in deze map.
   In mijn geval volstaat het om **git clone https://RenaatHaleydtAmista@bitbucket.org/RenaatHaleydtAmista/bachelorproef_sapui5.git** in te typen
2. Stap in de map van de repo en voer het commando **ui5 init** uit. Dit maakt een ui5.yaml file aan in de repo waar de gegevens om het project te deployen instaan
3. In dezelfde map voer je het commando **git add .gitignore** uit. Dit maakt een gitIgnore-file aan om ervoor te zorgen dat de node_modules niet op Git belanden
4. Open de gitignore file en vervang de content door volgende code:

```
fioriHtmlRunner.html
.*/fioriHtmlRunner.html
visual_ext_index.html
/webapp/visual_ext_index.html
extended_runnable_file.html
.*/extended_runnable_file.html
sap-ui-cachebuster-info.json
mock_preview_sapui5.html
.*/mock_preview_sapui5.html
UIAdaptation_index.html
changes_preview.js
AppVariant_index.html
AppVariantPreviewPayload.zip
mergedManifestDescriptor.json
APIExternalProducer.js
.*/APIExternalProducer.js
/mta_archives/
/node_modules/
.mta
*.mtar
dist
```

.gitignore

5. Open de file package.json en verwijder de huidige inhoud. Plak volgende code in de lege file:

```
{
 "name": "Bachelorproef_2018-2019.Bachelorproef_2018-2019",
 "version": "0.0.1",
 "description": "",
 "private": true,
 "devDependencies": {
  "@ui5/cli": "^1.2.0", "eslint": "^4.19.1",
  "karma": "^4.0.1",
  "karma-chrome-launcher": "^2.2.0", "karma-coverage": "^1.1.2",
  "karma-phantomjs-launcher": "^1.0.4", "karma-ui5": "^1.0.0",
  "grunt": "1.0.1",
  "@sap/grunt-sapui5-bestpractice-build": "^1.3.17"
 },
 "scripts": {
  "karma": "karma start",
  "karma-ci": "karma start karma.conf.js",
  "watch": "npm run karma"
 }
}
```

package.json

6. Maak een nieuw bestand aan in de hoofdmap van de repository en noem deze karma.conf.js . Dit bestand bevat het script waarmee Karma opgestart wordt. Plak volgende content in het bestand:

```
module.exports = function(config) {
	config.set({
		frameworks: ["ui5"],
		ui5: {
			url: "https:" + "//openui5.hana.ondemand.com",
			mode: "html",
			testpage: [
				"webapp/test/testsuite.qunit.html"
			]
		},
		browsers: ['ChromeHeadlessNoSandbox'],
		customLaunchers: {
			ChromeHeadlessNoSandbox: {
			base: 'ChromeHeadless',
			flags: ['--no-sandbox']
			}
		},
		singleRun: true
	 });
};
```

karma.conf.js

7. Er moet nog één bestand aangemaakt worden: .npmrc. Doe dit door in de root directory van het project **touch .npmrc** uit te voeren
8. Vul de file met volgende inhoud:

```
# The public npm registry from where to fetch e.g. Grunt
registry=https://registry.npmjs.org

# The SAP npm registry from where to fetch SAP specific Grunt modules
@sap:registry=https://npm.sap.com
```

.npmrc

9. In de console ben je nog altijd in de map van de SAPUI5 applictie repository. Voer nu een npm install uit

10. Controleer of de Web IDE een Gruntfile.js aangemaakt heeft in de root folder van het project. Indien dit niet het geval is maak je deze aan en plak je volgende code erin

```
module.exports = function (grunt) {
	"use strict";
	grunt.loadNpmTasks("@sap/grunt-sapui5-bestpractice-build");

	grunt.registerTask("default", [
		"lint",
		"clean",
		"build"
	]);
};
```

Gruntfile.js

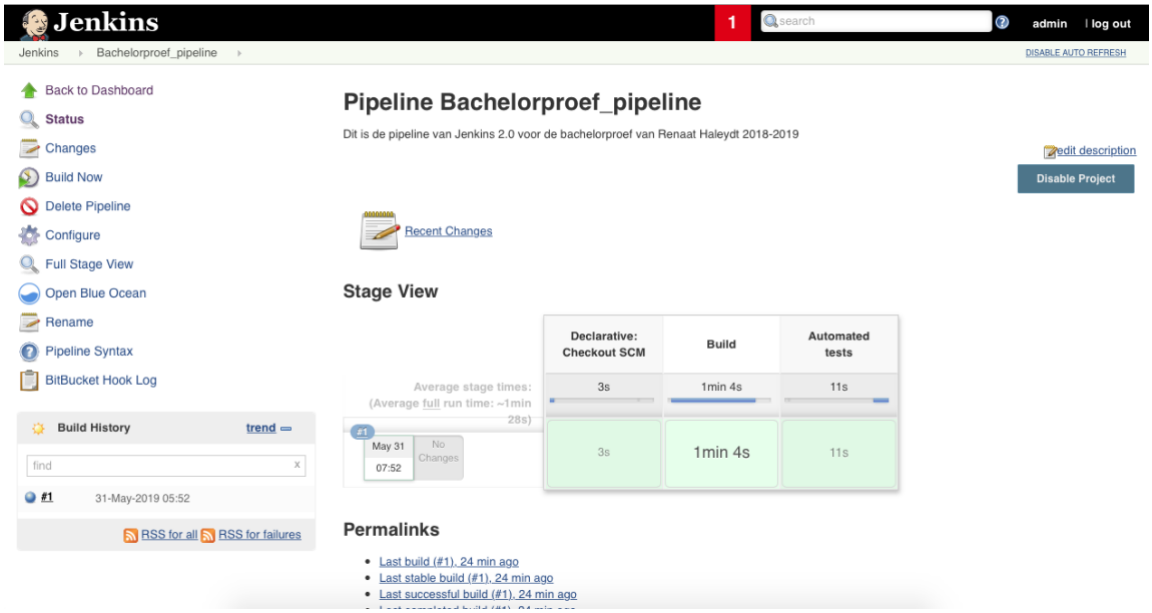
11. Er rest ons nu enkel de Jenkinsfile toe te voegen
12. In de console typ je volgend commando wanneer je je in de root folder van het project bevindt: **touch Jenkinsfile**
13. Je plakt volgende inhoud in deze nieuwe file

```
#!groovy

@Library('piper-lib-os') _

pipeline {
    agent any

    stages {
      stage('Build') {
         steps {
            sh 'npm install'
             }
      }
      stage('Automated tests') {
         steps {
          sh 'npm run karma'
         }
      }
   }
}
```

Jenkinsfile

14. Push de wijzigingen naar BitBucket
15. Automatisch wordt de build opgestart in Jenkins en wordt Karma uitgevoerd

16. Na deze stappen zou het scherm in de Bachelorproef_pipeline in de front-end er zo moeten uitzien



17. Wanneer alles groen kleurt is met success een Continuous Integration omgeving voorzien in Jenkins
18. Alle files uit de SAPUI5 applicatie zijn ook terug te vinden op deze BitBucket repository: https://RenaatHaleydtAmista@bitbucket.org/RenaatHaleydtAmista/bachelorproef_sapui5.git

B.4.5 Delivery fase

Hier moeten we de mtar file aanmaken die aan de hand van de Multitarget Application Archive Builder wordt gemaakt.

B.4.5.1 Jenkins server

Eerst downloaden we de Multitarget Application Archive Builder op onze server.

1. Ga naar de console van de Jenkins server
2. Ga naar de root directory door **cd /** in te typen
3. Maak een map MTA aan in deze directory: **mkdir MTA**
4. Download de MTA Builder door volgend commando in te typen: **wget -nv --output-document=/MTA/mta.jar --no-cookies --header "Cookie: eula_3_1_agreed=tools.hana.ondemand.com/developer-license-3_1.txt" https://tools.hana.ondemand.com/additional/mta_archive_builder-1.1.19.jar**

   Let wel op dat je de juiste versie download. De laatst mogelijke versie van de MTA Builder is hier terug te vinden: https://tools.hana.ondemand.com/#cloud

B.4.5.2 SAPUI5 project

Er moeten twee zaken gebeuren alvorens de mtar aangemaakt kan worden: er moet een stage worden toegevoegd aan de Jenkinsfile. Jenkins weet dat er een extra fase bijkomt en voert deze automatisch uit.
Ook moet er een nieuwe file mta.yaml aangemaakt worden die de MTA Builder zal gebruiken om de mtar file aan te maken.

1. Voeg volgende stage toe aan de Jenkinsfile

```
stage('Ready to deploy') {
        steps {
          sh 'export PATH=${WORKSPACE}/node_modules/grunt/bin:$PATH'
          sh 'java -jar /MTA/mta.jar --mtar bachelorproef.mtar --build-target=NEO
build'
        }
    }
```

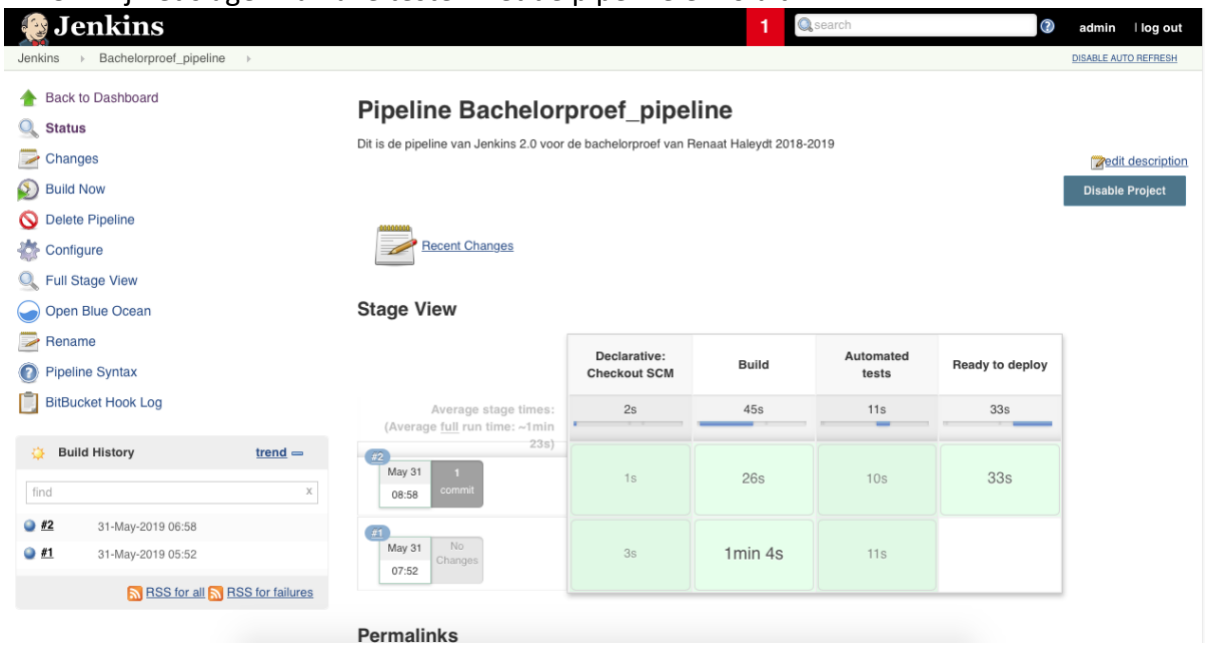
Jenkinsfile

2. Maak de nieuwe mta.yaml file aan door in de root directory van het project volgend commando uit te voeren: **touch mta.yaml**
3. Open de file en plak volgende content erin

```
_schema-version: "2.0.0"
ID: "Bachelorproef_2018-2019.Bachelorproef_2018-2019"
version: 0.0.1

parameters:
 hcp-deployer-version: "1.0.0"

modules:
 - name: "Bachelorproef_2018-2019"
   type: html5
   path: .
   parameters:
     version: 1.0.0-${timestamp}
   build-parameters:
    builder: grunt
    build-result: dist
```

Mta.yaml

4. Push de wijzigingen naar BitBucket en open de Bachelorproef_pipeline in de front-end van Jenkins. Automatisch wordt de pipeline gestart.
5. Bij het slagen van alle testen ziet de pipeline er zo uit



6. Op dit moment is er een .mtar file in de root folder aangemaakt door de MTA Builder. Deze kan worden gebruikt om te deployen naar SAP Cloud Platform wat de Delivery fase voltooid. Deze is terug te vinden op de Jenkins server in de directory met path: /var/lib/jenkins/workspace/Bachelorproef_pipeline.

B.4.6 Blue Ocean pipeline

Jenkins biedt plugins aan om de pipeline beter te visualiseren, Blue Ocean genaamd.
Om hier gebruik van te maken moet je volgende stappen uitvoeren.

1. Ga naar de front-end van Jenkins en klik bovenaan links op Jenkins
2. Kies voor Manage Jenkins -> Manage Plugins
3. Ga naar de tab Available en zoek voor Blue Ocean
4. Klik op de Plugin genaamd Blue Ocean en druk nadien op de knop Download now and install after restart
5. Ga naar de console van de jenkins server
6. Voer het commando: **sudo systemctl restart jenkins** uit
7. Ga terug naar de front-end van Jenkins
8. Klik op de Bachelorproef_pipeline en klik nadien in de linkse op Open Blue Ocean
9. Klik op de knop Run en klik op de rij van de build die bezig is. Een nieuw venster wordt geopend
10. Als alles goed gaat heb je een mooie visualisatie van het verloop van de pipeline zoals hier te zien is