Installatie Ubuntu 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

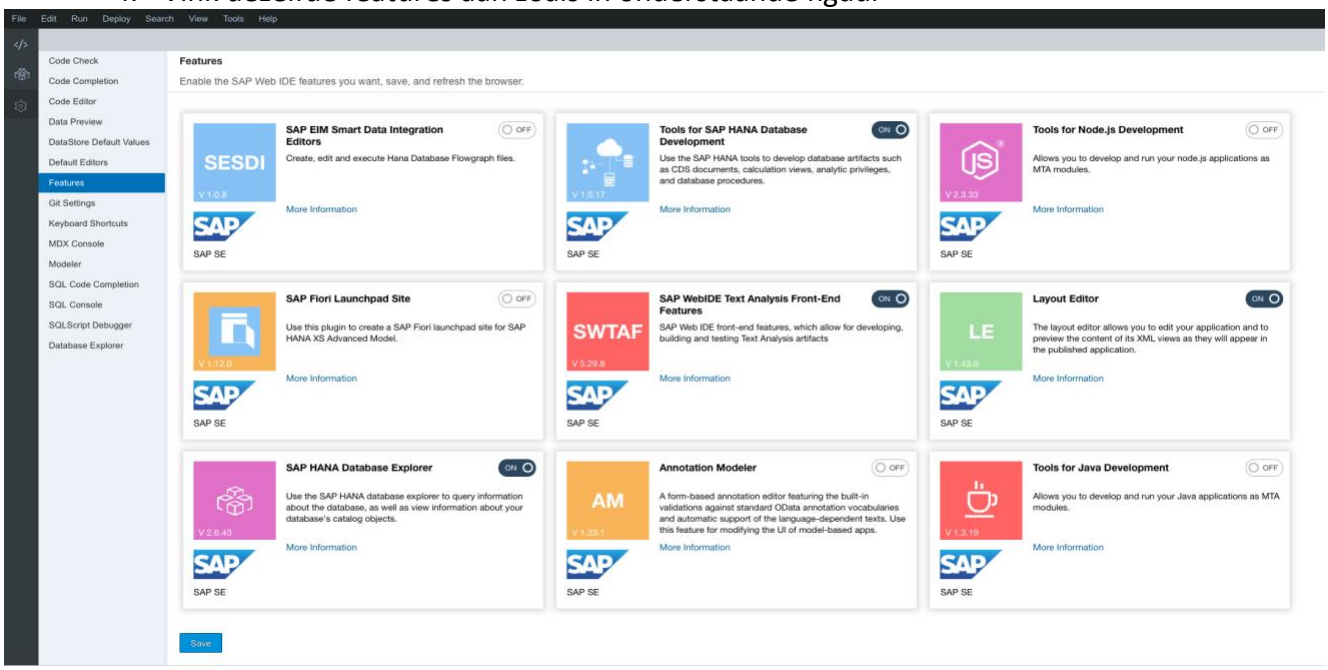
Source Control

Maak een nieuwe private repository in BitBucket:

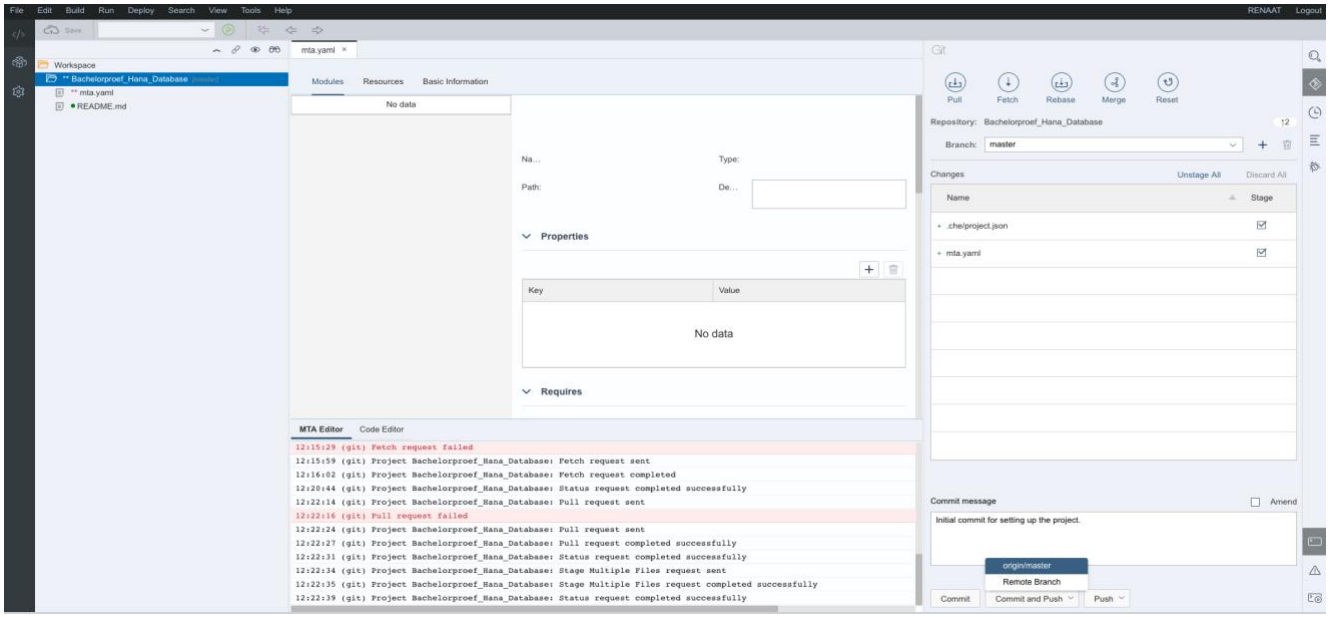
1. Kies een gepaste naam, vink ‘This is a private repository’ aan en kies voor Git.
2. Kopieer het adres dat gegeven wordt om een clone te maken van deze repository

Database

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



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



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;
  };
};
```

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
 }
}]
}
```

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
```

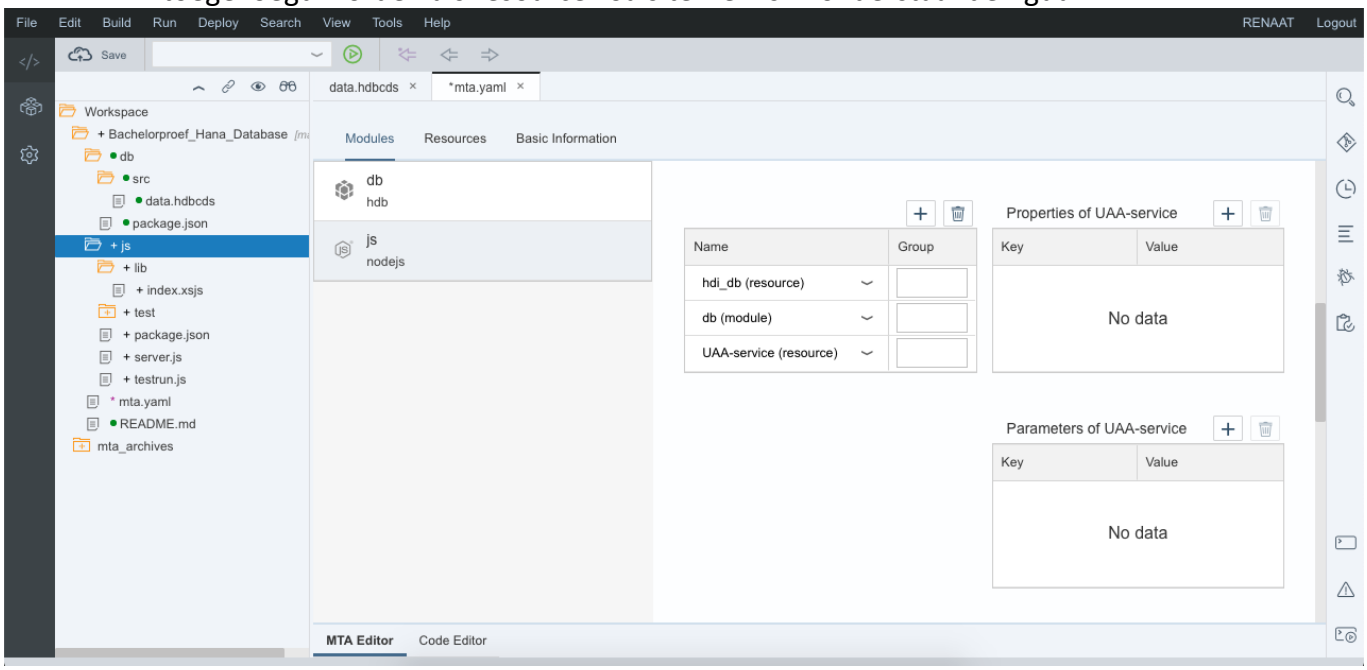
13. Importeer de gemaakt file in de map ‘data’ van de ‘db-module’
14. Build de ‘db-module’ nogmaals
15. Controleer of er effectief data aanwezig is door naar de 'Database Explorer' te gaan
16. Open de map 'tables'
17. Links vanonder verschijnt een venster met de verschillende tabellen. Er rest ons enkel 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

Node.js module

Maak een Node.js module om de data te 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";
}
```

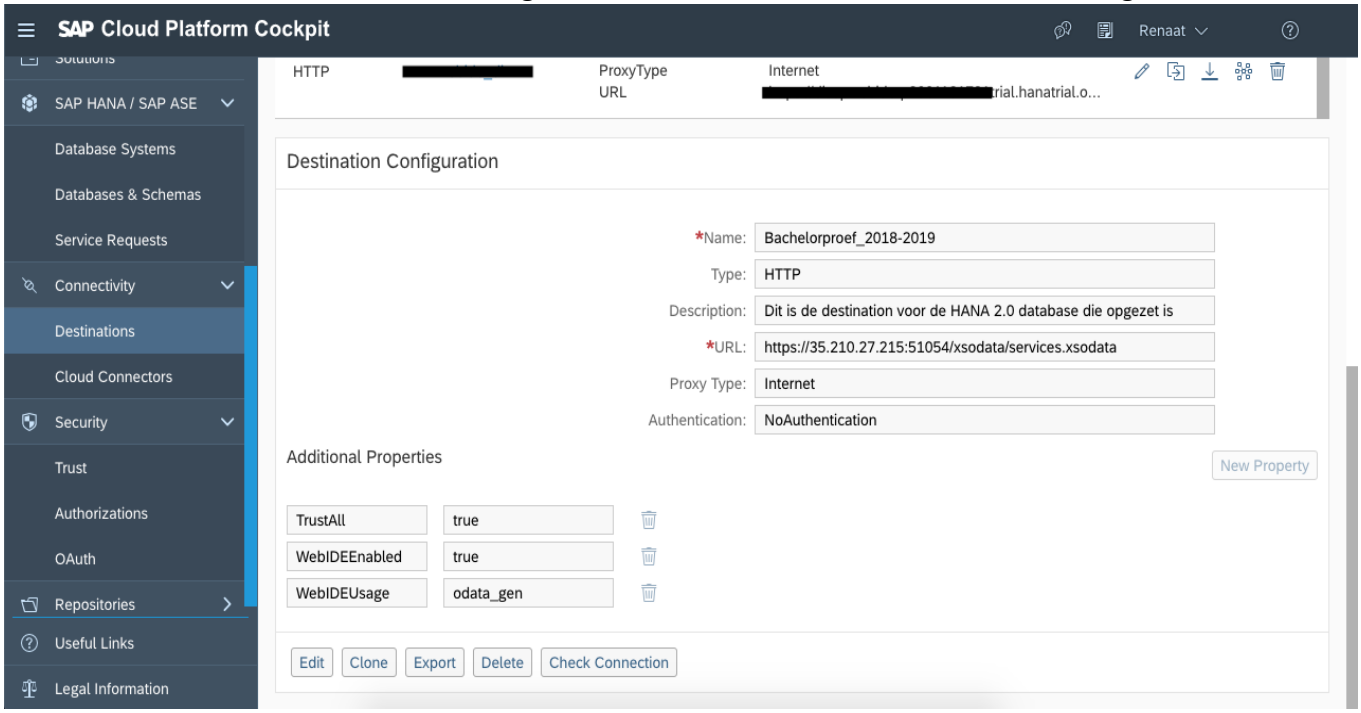
7. Maak een xsjs service door in de ‘lib’ folder een nieuwe file aan te maken met volgende naam: **xsjs/hdb.xsjs**

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;
```

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 zoals in onderstaande figuur



UI5 applicatie

Maak een nieuwe applicatie, startend vanaf een template:

1. Kies SAPUI5 Application en geef het een correcte naam (Bachelorproef_2018-2019)
2. Kies XML als formaat voor de eerste view en geef het een juiste naam (App bijvoorbeeld)
3. Klik op finish
4. 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"
    }
```

5. Voeg volgende code toe aan het ‘sap.app’ object in manifest.json

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

6. In de manifest.json moet ook nog het default model toegevoegd worden aan het object ‘models’

```
“” :  { “dataSource”: “mainService” }
```

```
<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>
```

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

8. 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);
		}
```

9. U kan nu de app runnen en testen

Testen

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);
	});

});
```

2. In de test folder moet volgende file aangemaakt worden: QUnit.js . In de file moet dezelfde content komen als in 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 dit:

```
<!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.

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. OPA testen toevoegen

Proof-of-concept

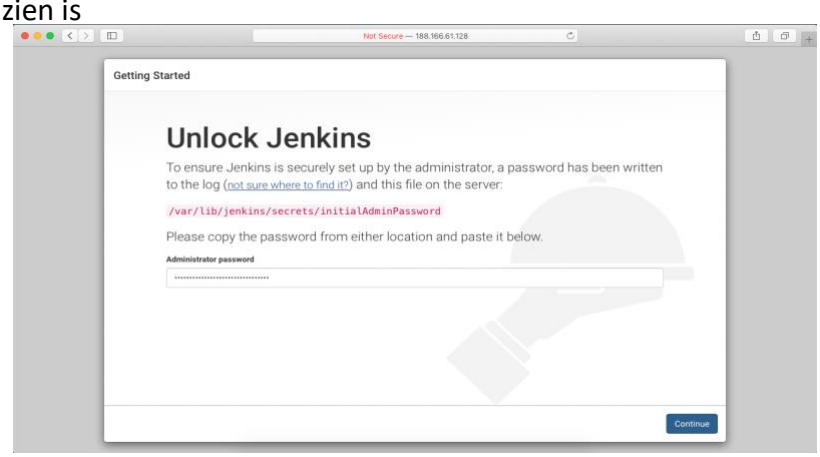
Jenkins

Installatie van Jenkins op de 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 kunnen we overgaan tot de installatie van Jenkins op de server. We maken gebruik van de packages die Jenkins onderhoud om de software te installeren op Ubuntu door **wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -** in te typen
5. Eens dit gebeurd is moeten we de Debian package repository toevoegen 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 moeten we die kunnen aanspreken vanuit een browser, daarom moeten we de regels van de firewall wat aanpassen. 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.
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 gecontrolleerd hebt op zijn juistheid
16. Nadien kies je voor 'Start using Jenkins'
17. Het dashboard van Jenkins wordt geopend

Configureren van Jenkins in samenwerking 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 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 build en kies voor Freestyle project
14. Bij Source Code Managements kies je voor Git en voer je de url naar de BitBucket repository in
15. Bij Credentials kies je om een nieuwe credential toe te voegen
16. Vul bij Username en Password de gegevens van je BitBucket account in
17. Zorg dat bij Branches to build */master is ingevuld
18. Kies bij Repository browser bitbucketweb uit de lijst
19. Bij Build Triggers kies je voor de optie: Build when a change is pushed to BitBucket
20. Druk op Save om alles op te slaan

Node.js en npm installeren 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

Nodige packages installeren op Ubuntu 18.04

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 gebruiken we 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**

Configureer het UI5 project

1. In de console op je lokale machine moet je de repository downloaden.
   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 voeg /node_modules toe aan de lijst

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

```
{
  "name": "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-ui5": "^1.0.0"
  },
  "scripts": {
    "karma": "karma start"
  }
}
```

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. Ga terug naar de build job in Jenkins en klik op Configure in de linkse kolom
8. Scroll helemaal naar onder tot de sectie Build en voeg daar een build step toe
9. Kies voor Execute shell
10. Typ het volgende in de Command invoer

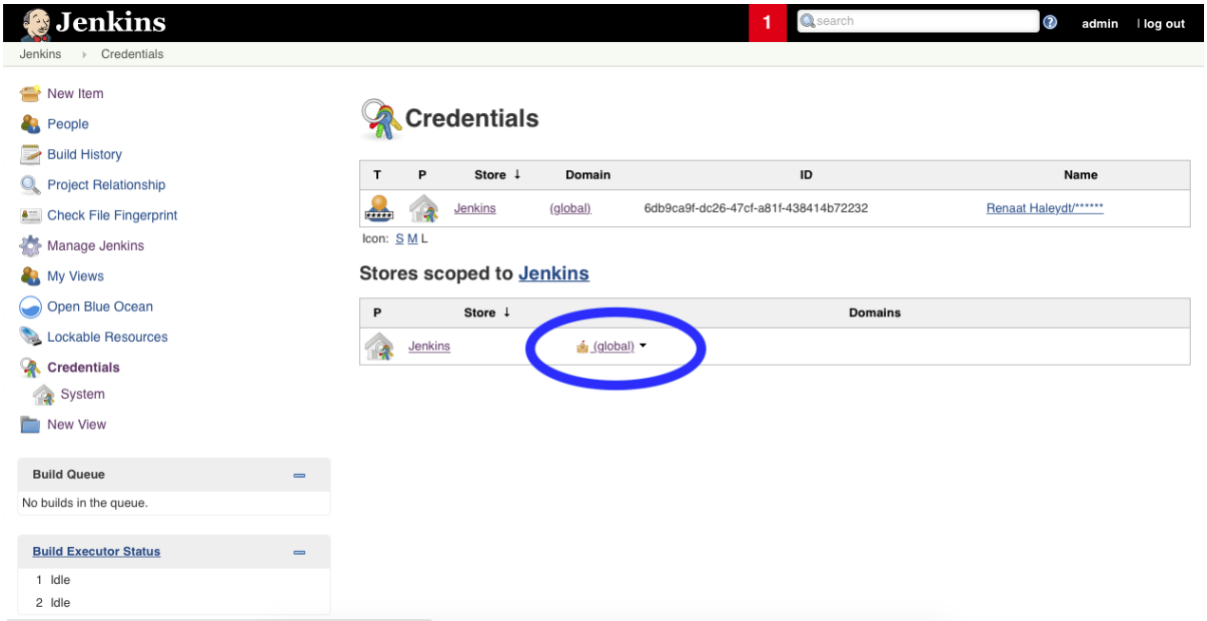
```
npm install
npm run karma
```

11. Druk op Apply en nadien op Save
12. Ga nu terug naar de gekloonde repository in de console op jouw machine en push de wijzigingen naar de source control manager
13. Automatisch wordt de build opgestart in Jenkins en Karma wordt uitgevoerd
14. Alle Unit en integration testen zouden moeten slagen zoals op de figuur te zien is TODO: FOTO slagen testen (in console) toevoegen

Deploy Stappen

Jenkins

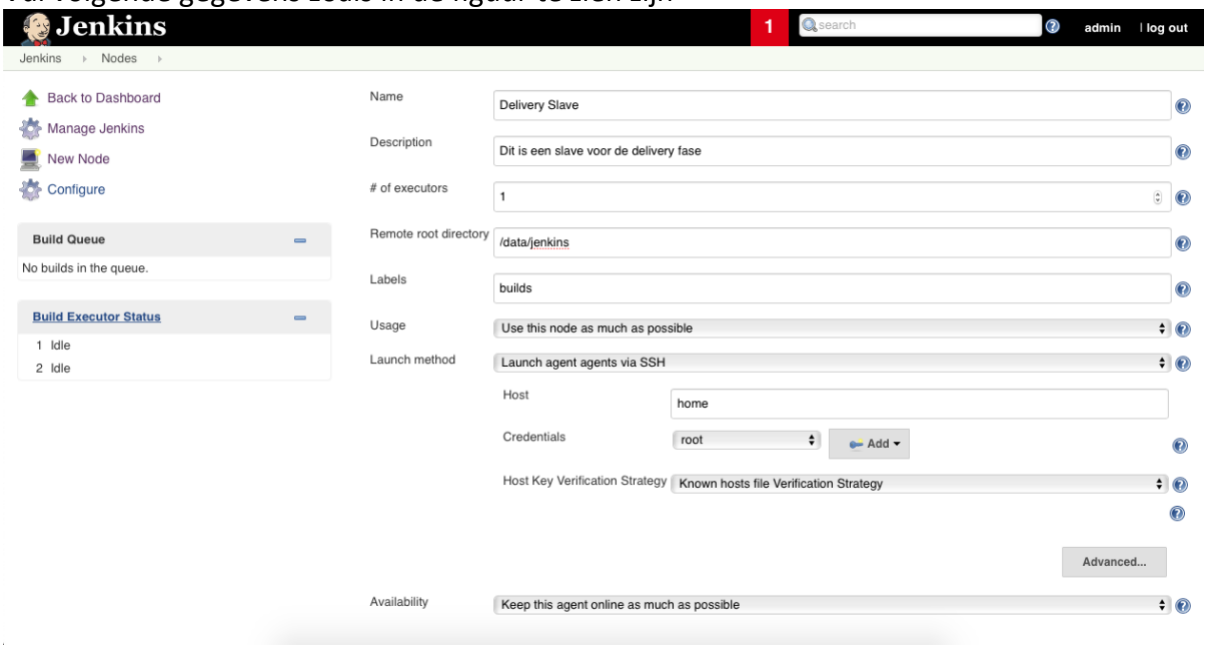
1. Ga terug naar de front-end van Jenkins en download volgende plugins:
    a. Build Pipeline
    b. ConditionalBuild
    c. Copy Artifact
2. Klik op Download now and install after restart
3. Herstart Jenkins op de server
4. Op de server moet er een ssh-key gemaakt worden door **ssh-keygen** te typen
5. Druk 3x op enter om de default waarden te krijgen
6. Ga nu terug naar de front-end van Jenkins en kies in de linkse kolom voor Credentials en klik op de knop zoals op de figuur



7. Links ziet u dan een knop Add Credentials verschijnen. Eens hierop geklikt, verschijnt er een nieuw venster waar je volgende gegevens invoert:
    a. Kind: SSH Username with private key

b. Scope: Global
c. Username: Uw accountnaam van de lokale machine waar je jenkins runt
d. Private Key vink je Enter directly aan en plak je je private key van je server die zonet gemaakt is

8. Op je lokale machine die je als Jenkins slave gaat gebruiken moet je naar de pagina: https://tools.hana.ondemand.com/#cloud gaan om de SAP JVM te downloaden
9. Zorg er ook voor dat Git is geïnstalleerd op je Jenkins Slave
10. Op je Jenkins slave ga je naar de directory ~/.ssh en kopieer je de public ssh key van de Jenkins master (de server) in een nieuwe file authorized_keys genaamd
11. Op je jenkins server moet je een directory maken waar de build job workspace gelokaliseerd is en noem deze /data/jenkins
12. Keer terug naar de front-end van Jenkins en kies in de linker kolom voor Manage Jenkins
13. Kies voor Manage Nodes en nadien voor New Node (in linkse kolom)
14. Geef een correcte naam voor de nieuwe slave (Delivery Slave bijvoorbeeld) en vink de optie permanent aan
15. Vul volgende gegevens zoals in de figuur te zien zijn



16. Druk op Save

Nexus

1. Op je lokale machine (in dit geval een Macbook Pro 2012) maak je in de home directory een map aan, nexus genaamd
2.

1. Maak een nieuwe user op de Jenkins server door **adduser nexus** te typen
2. Geef de user een nieuw paswoord en bevestig het paswoord
3. De gegevens over de Full Name tot en met Other mogen gewoon blanco gelaten worden
4. Geef de nieuwe user sudo access door **usermod –aG sudo nexus** te typen als root

5. Maak in de root folder een nieuwe folder aan die data heet en die nog een extra folder heeft: nexus
6. Log in als nexus door **login nexus** te typen
7. Geef het juiste paswoord in
8. Verander de permissies van de /data/nexus door **chmod u=rw,o=r /data/nexus** te typen
9. Ga in de /data/nexus folder en typ volgend commando als nexus: **sudo wget http://www.sonatype.org/downloads/nexus-latest-bundle.zip**
10. Om te unzippen moet je de unzipper downloaden door **sudo apt install unzip** in te typen en het sudo paswoord in te geven
11. In de /data/nexus folder typ je **sudo unzip nexus-latest-bundle.zip** waardoor de download unzipt
12.