**Installation  
Manual**

Ziggo team 4

|  |  |
| --- | --- |
| **Version** | 1.0 |
| **Date** | 9th May 2019 |
| **Team** | Ziggo team 4 |
| **Authors** | 500763721 – Donovan Schaafsma 500764941 – José Niemel 500775494 – Youssef Ben Youb  500780453 – Matthijs Snijders  500778162 – Emre Efe |
| **Client** | Ziggo |

# 

# Table of contents

[Table of contents 1](#_heading=h.1fob9te)

[Introduction 2](#_heading=h.2et92p0)

[Requirements 3](#_heading=h.3dy6vkm)

[Server side 3](#_heading=h.1t3h5sf)

[Web platform 3](#_heading=h.4d34og8)

[Database 4](#_heading=h.2s8eyo1)

[Setup 4](#_heading=h.17dp8vu)

[Jax RS Server 5](#_heading=h.3rdcrjn)

[Setup 5](#_heading=h.26in1rg)

[Deployment 5](#_heading=h.lnxbz9)

[Web application 8](#_heading=h.35nkun2)

[Setup 8](#_heading=h.1ksv4uv)

[Deployment 8](#_heading=h.44sinio)

# Introduction

In this document, we will explain how to install and deploy all the necessities to get the ziggo platform up and running.

First off, we will have to set up a database that requires MYSQL. After that we have to set up a Jax RS server so that we can communicate with the MYSQL database locally. Finally, we will have to set up a web platform where we can run the web application. When this is all up and running, the application will be deployed.

# 

# Requirements

Here we outline all the required software and dependencies that are needed to get the web platform up and running correctly.

## Server side

The required software and the dependencies that are needed to get the server up and running are the following:

* Our git project: ONZE GIT LINK HIER
* JDK1.8: <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
* Maven Java.
* Hibernate (Maven).
* Jax RS (Maven).
* Glassfish or any other Java server that you would like to use.
* Database running on MYSQL.

## Web platform

The required software and dependencies that are needed to get the server up and running are the following:

* Our git project: ONZE GIT LINK HIER
* Github Desktop or any other FTP of your choice.
* Node Package Manager (NPM): <https://node.js>
* Angular CLI: <https://angular.io/guide/quickstart>

# Database

To get the platform fully functional, we will need to have a stable database so that we can save all the data we gather and have for this application on one safe place. The database is configured with MYSQL and should be able to communicate with the local Java server. For our project. We used Jax RS.

## Setup

Firstly, go to: **GIT LINK HIER** and then download the git project. This can be done easily by clicking on the cloud icon with a downward facing arrow in it. Once this icon is pressed you can choose to download the project.

After the first step you will need to start your database server through whatever server you would like to use. Open it with a SQL editor of your choice. We used MYSQL Workbench. Now you will need to import a SQL file from our repository. Containing in this file is the code that is needed to automatically create the correct database with all the correct information in it for you. This file can be found in the **src/main/resources/sql/data.sql** folder.

After you have successfully imported the SQL file, your database will be set up and ready to go.

# Jax RS Server

To get the platform up and running, we will need a server that can safely communicate with our database and our application. For this, we used a Jax RS server that runs on Java. The server works as a REST API that can communicate through JSON payloads.

## Setup

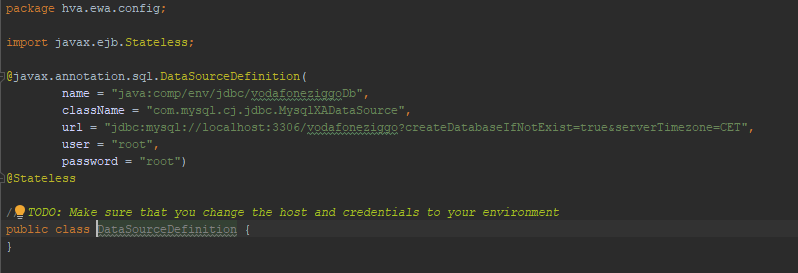
First of all, make sure that all the steps above have been done.

To set up the Jax RS server, you will first need to download the correct version of Java. The version of Java that you need is Java 1.8. You can download this from the official Java website.

After this is done. You need some software that can actually run a Java server. At the start of this document we put a few server that we have been using to run our servers with. One thing that you should keep remember is that the server should be able to run Maven plugins, if that isn’t the case. The persistence layer that communicates with the database will not be able to work.

After this step navigate to: **EWAZiggo-backend\src\main\java\hva\ewa\config\DataSourceDefinition.java** file. In the file you will find all the relevant database information.

These properties need to be changed:



In the first field, you will need to put the name of your database.

In the second field, you will need to put th class name.

In the third field you need to put the url of the database.

In the fourth field, you will need to put your username you use to connect to the database. In our case it is “root”.

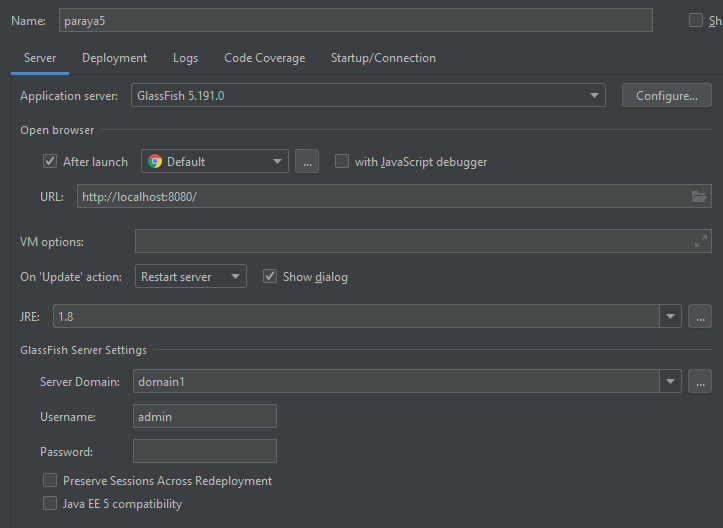
In the fifth and last field, you will need to put your password you use to connect to the database. This password can be a custom one, but most passwords are quite standard. For instance, “root” or “admin”. Be careful whilst entering the credentials. Because it can cause some security issues.

After setting up your configuration, you will now be ready for deployment.

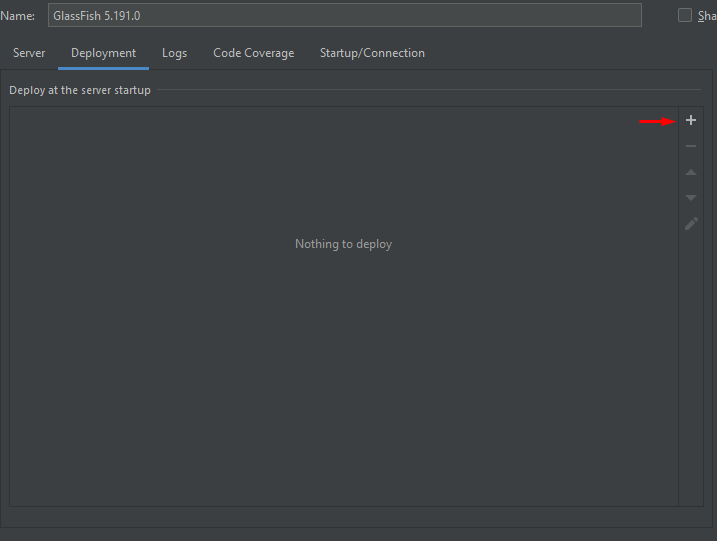
## Deployment

To deploy the Jax RS server, run the Java code via one of the servers from the requirements.

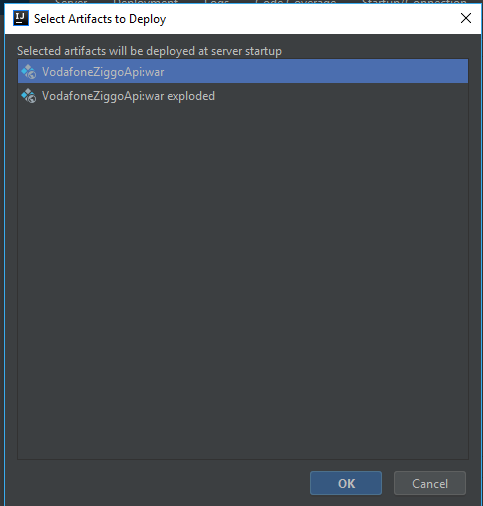
Open your IDE of choice and go to the edit configurations tab.   
The tab should look something like this:



In this window, click on “Deployment” and you will see this tab.

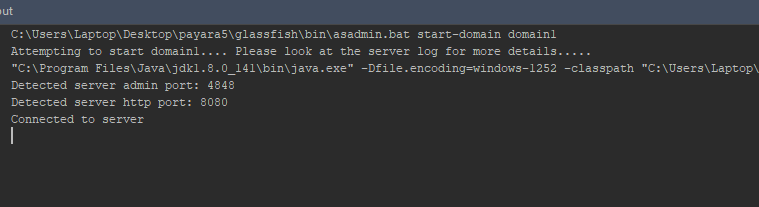


Click on the “+” button and click on “Artifact”



Select “**VodafoneZiggoZpi:war**”.

If your server is successfully running, you should see a output identical to this:



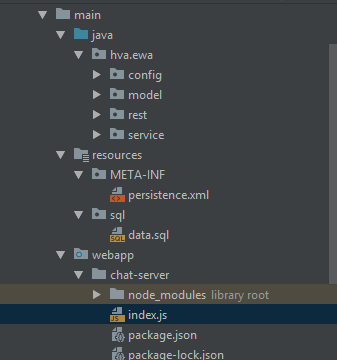
# Chat Server

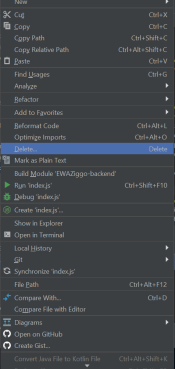
The chat server is used in our application to communicate. The chat server allows users to communicate to employees and it also allows employees to chat to other employees. So the chat server is an essential part of our application

## Setup

The set up for the chat server is quite simple. But you have to make sure you have everything set up regarding the application before setting up the chat server, otherwise it won't work.

## Deployment

If everything works as intended, you will now be able to deploy the chat server. To deploy the chat server you need to do a few short steps in order the actually deploy it.  
  


right click on the “index.js”.  
  


And then click on “Run ‘index.js’”. Once you have run this file. The chat server will be up and running and you will be able to type in the chat.

# Web application

After we have successfully set up our database and server, we would want to see what the application looks like whilst running. To run our application we need a web server that is able to host multiple files and that is able to run TypeScript.

## Setup

Make sure that the git project is downloaded as described in the previous two chapters above.

We would want to use the Angular CLI to build the existing source files into production so we can add this into our web server through a FTP. To do this, we need to go back to the requirements section and download all necessities that are needed to be able to run our project, such as Node Package Manager (NPM) and Angular CLI.

After setting up your NPM to work with Angular CLI, you can start using the CLI.

To see if the application works you must write down the following in the CLI console:***ng serve –o*** After building the webapp, it will automatically open in your system default browser like google chrome, or microsoft edge.

If the application does not have any data to show it will most likely means that your application isn’t properly connected to the Jax RS server. The most likely reason for this is the following: the Jax RS server is not on the same host or the server is not online. If this is the case, you will have to put your Jax RS server on a different host or you should change the url inside all of the .service.ts files. This can be done with the “Find and Replace” functionality inside your IDE.  
Try to find “localhost:8080” and replace this localhost with your hosts ip address and port. You should do this for all occurrences inside the project.

## Deployment

If everything else works as intended, you can now start building the project for production.

If all else works as intended, you can start building the project for production.

This is done with the command: ***ng build -prod****.*

After the angular CLI finishes successfully creating the production build, you can navigate towards the */app* folder inside the project. In here you can find all files that make up the web application.