City of
Amsterdam

Version: 1.0
5 January 2023

Amsterdam Municipality

Manual installation

Class: HBO-ICT SE, IS 204
POD: Okechukwu Onwunli
Team: 4
Members: Harmohat Khangura, Arunn LIngeswaren, Ayoub Ez-Zaouia
Jordy Mol, Senne Chin, Ryan Koning

Front-end

Dependencies used

The dependencies can be found in fe/package.json (content root)

Name	**Version**
@headlessui/vue	^1.7.3
@splidejs/vue-splide	^0.6.12
@vue-hero-icons/outline	^1.7.2
axios	^1.1.2
babel-core	^6.26.3
babel-present-env	^1.7.0
bcryptjs	^2.4.3
core-js	^3.8.3
express	^4.18.2
http	^0.0.1-security
http-server	^14.1.1
node-polyfill-webpack-plugin	^2.0.1
vue	^3.2.13
vue-router	^4.1.5
vuex	^4.0.2

Dev dependencies used

The dev dependencies can be found in fe/package.json (content root)

Name	Version
@babel/core	^7.12.16
@babel/eslint-parser	^7.12.16
@vue/cli-plugin-babel	~5.0.0
@vue/cli-plugin-eslint	~5.0.0
@vue/cli-plugin-unit-jest	~5.0.0
@vue/cli-service	^5.0.8
@vue/test-utils	^2.0.0-0
@vue/vue3-jest	^27.0.0-alpha.1
autoprefixer	^10.4.12
babel-jest	^27.5.1
eslint	^7.32.0
eslint-plugin-vue	^8.0.3
jest	^27.0.5
jest-expect-message	^1.1.3
nodemon	^2.0.20
postcss	^8.4.16
tailwindcss	^3.1.8

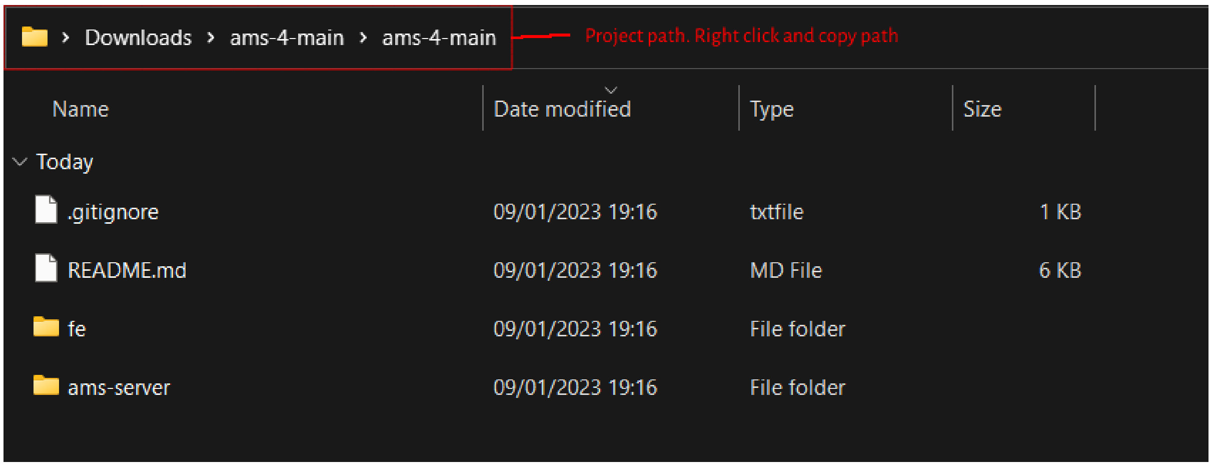
Installation

IntelliJ (https://www.jetbrains.com/idea/download/#section=windows)
MySQL

NOTE: The text after the $ are commands which can be executed in the windows terminal.

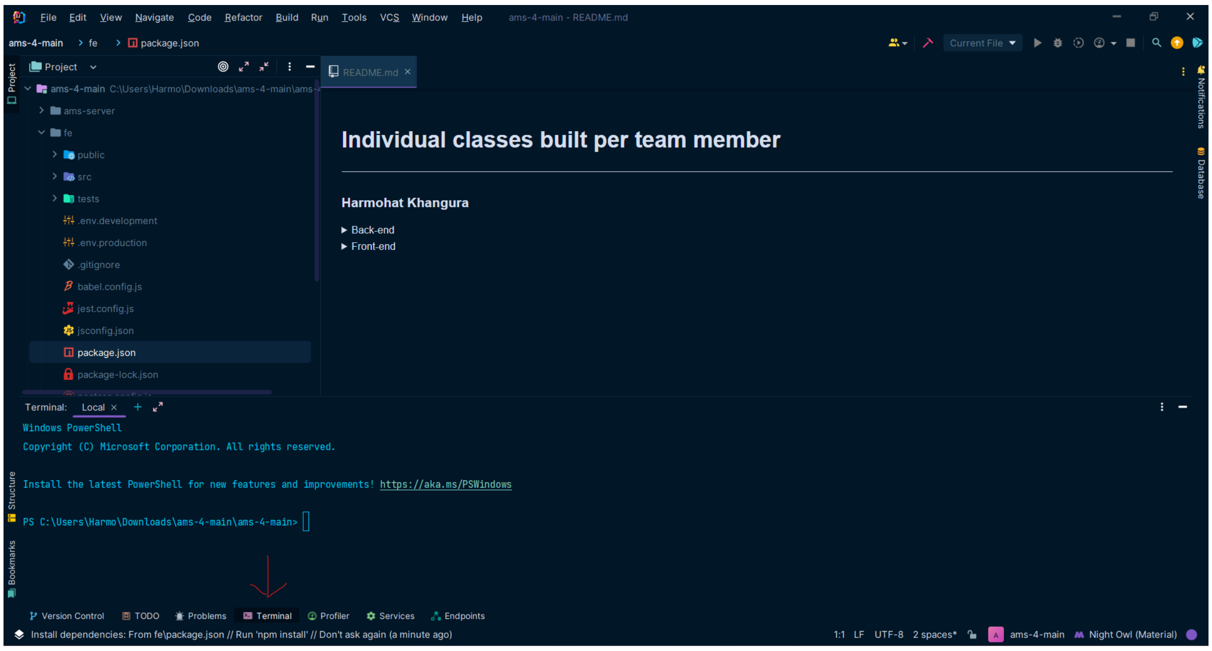
1. Install Node + Node Package Manager (NPM) from https://nodejs.org/en/download/
   ```
   $ node --version
   $ npm --version
   $ npm install -g npm
   $ npm install -g npm@latest
   $ npm install -g npm@6.14.13
   ```
2. Install Vue/CLI (Command Line Interface) https://cli.vuejs.org/guide/installation.html
   ```
   $ npm install -g @vue/cli
   $ npm update -g @vue/cli
   $ vue --version
   ```
3. If you need to upgrade an existing project to the latest version of vue
   ```
   $ npm install vue@latest --save
   $ vue upgrade
   ```

1. Clone or download the project to your computer.
   a. (Extract folder if downloaded as zip)
2. Open IntelliJ and open the extracted project
   a. Open IntelliJ
   b. Click on the menu item `File` top left
   c. Click the option `open`
   d. Enter the project path and click `OK`


Downloads › ams-4-main › ams-4-main
Project path. Right click and copy path

Name	Date modified	Type	Size
Today			
.gitignore	09/01/2023 19:16	txtfile	1 KB
README.md	09/01/2023 19:16	MD File	6 KB
fe	09/01/2023 19:16	File folder	
ams-server	09/01/2023 19:16	File folder	

3. Click on `terminal` on the bottom of your editor


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ams-4-main - README.md
ams-4-main > fe > package.json
Current File
Project
README.md
ams-4-main C:\Users\Harmo\Downloads\ams-4-main\ams-
ams-server
fe
public
src
tests
.env.development
.env.production
.gitignore
babel.config.js
jest.config.js
jsconfig.json
package.json
package-lock.json
Terminal: Local
Version Control TODO Problems Terminal Profiler Services Endpoints
1:1 LF UTF-8 2 spaces* ams-4-main Night Owl (Material)

Individual classes built per team member

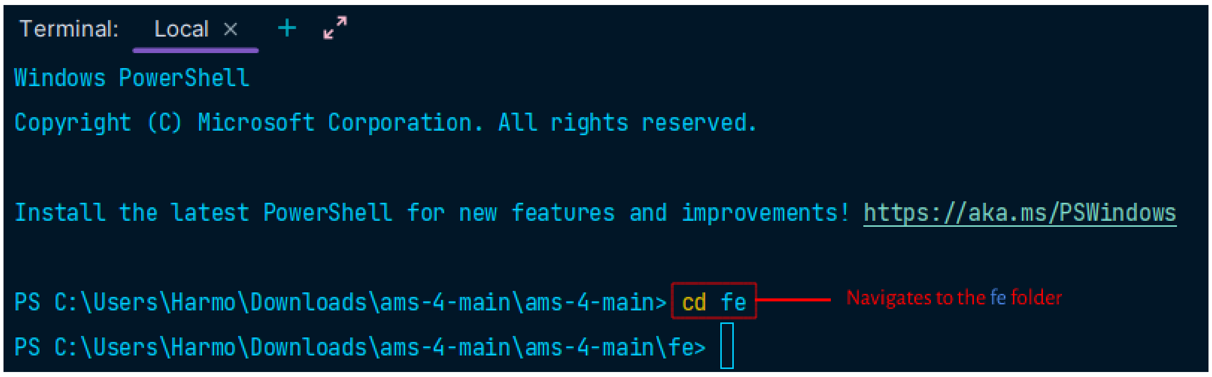
Harmohat Khangura

- Back-end
- Front-end

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Harmo\Downloads\ams-4-main\ams-4-main>
```

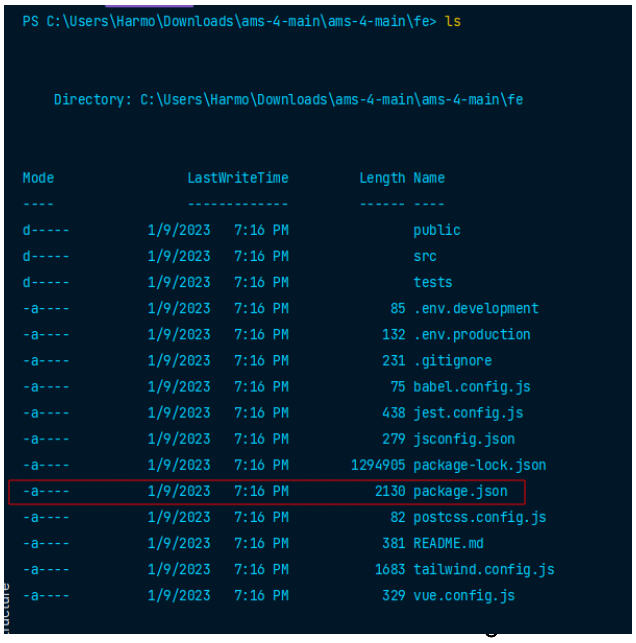
Install dependencies: From fe\package.json // Run 'npm install' // Don't ask again (a minute ago)

4. In the terminal, navigate to the *fe* folder.
    a. Command: `cd fe` (and click enter)



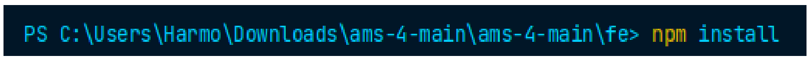
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Harmo\Downloads\ams-4-main\ams-4-main> cd fe
PS C:\Users\Harmo\Downloads\ams-4-main\ams-4-main\fe>
```

5. For the the frontend we need to install all the dependencies which are described in `package.json`
    a. In the terminal you must be inside the *fe* directory.
    b. To show the content of the *fe* directory you can run the following command:
        i. Command: `ls` (and click enter)



```
PS C:\Users\Harmo\Downloads\ams-4-main\ams-4-main\fe> ls


    Directory: C:\Users\Harmo\Downloads\ams-4-main\ams-4-main\fe


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          1/9/2023   7:16 PM                public
d-----          1/9/2023   7:16 PM                src
d-----          1/9/2023   7:16 PM                tests
-a----          1/9/2023   7:16 PM             85 .env.development
-a----          1/9/2023   7:16 PM            132 .env.production
-a----          1/9/2023   7:16 PM            231 .gitignore
-a----          1/9/2023   7:16 PM             75 babel.config.js
-a----          1/9/2023   7:16 PM            438 jest.config.js
-a----          1/9/2023   7:16 PM            279 jsconfig.json
-a----          1/9/2023   7:16 PM        1294905 package-lock.json
-a----          1/9/2023   7:16 PM           2130 package.json
-a----          1/9/2023   7:16 PM             82 postcss.config.js
-a----          1/9/2023   7:16 PM            381 README.md
-a----          1/9/2023   7:16 PM           1683 tailwind.config.js
-a----          1/9/2023   7:16 PM            329 vue.config.js
```

    c. To install the dependencies we must run the following command inside the *fe* directory.
    Command: `npm install`
    d. When everything was installed you should see a new folder in the *fe* folder called: *node_modules*

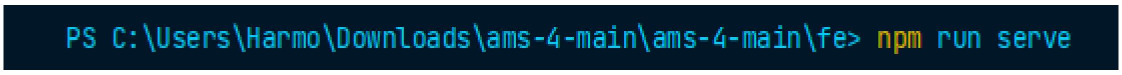


```
PS C:\Users\Harmo\Downloads\ams-4-main\ams-4-main\fe> npm install
```

Commands

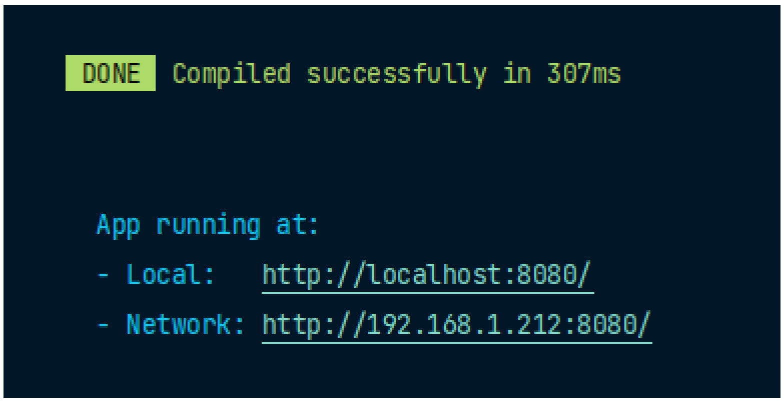
The following commands are meant to be runned in the local/development environment. These commands are being runned in the terminal in the *fe* directory.

- The `npm run serve` command compiles and starts the development server.



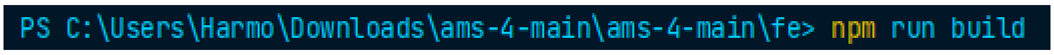
```
PS C:\Users\Harmo\Downloads\ams-4-main\ams-4-main\fe> npm run serve
```

- After running the serve command you will get this (see picture underneath) result. You can open the links to see the website.



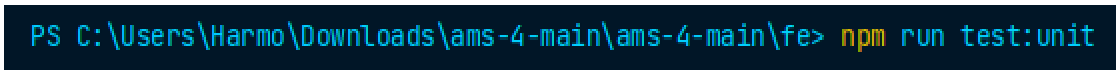
```
 DONE  Compiled successfully in 307ms


  App running at:
  - Local:   http://localhost:8080/
  - Network: http://192.168.1.212:8080/
```

- `npm run build`: .vue or .jsx files would need to be compiled into Javascript in order for the browser to deal with it. `npm run build` creates a build folder with compiled js files that you can upload to the server.



```
PS C:\Users\Harmo\Downloads\ams-4-main\ams-4-main\fe> npm run build
```

- `npm run test:unit:` runs all the created tests



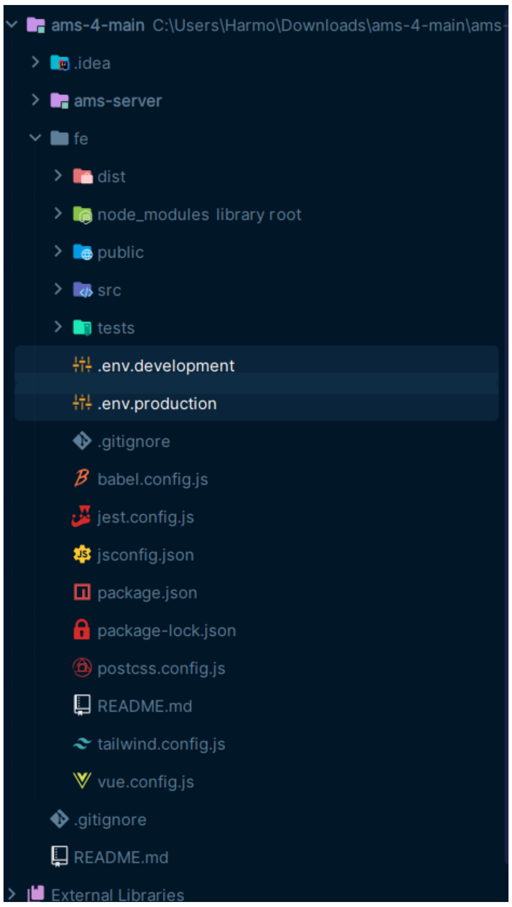
```
PS C:\Users\Harmo\Downloads\ams-4-main\ams-4-main\fe> npm run test:unit
```

The front-end has been set up now. To get the application 100% up and running we need to set up the back end.

.env files

The application uses an API, this API is made in the back-end. Our front-end and back-end are running on different ports and URLs. In the .env files we can store this API url. In the *fe* directory there are 2 .env files.

1. *.env.development* : for local development
2. *.env.production* : for production



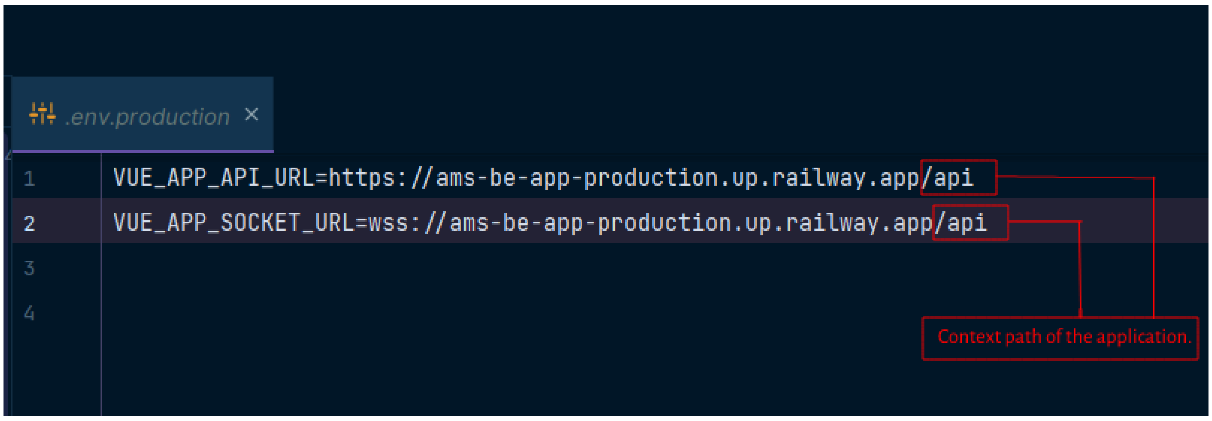
```
ams-4-main  C:\Users\Harmo\Downloads\ams-4-main\ams-
  .idea
  ams-server
  fe
    dist
    node_modules  library root
    public
    src
    tests
    .env.development
    .env.production
    .gitignore
    babel.config.js
    jest.config.js
    jsconfig.json
    package.json
    package-lock.json
    postcss.config.js
    README.md
    tailwind.config.js
    vue.config.js
  .gitignore
  README.md
External Libraries
```

The most important file to look at is the *.env.production* file. For the key `VUE_APP_API_URL` you have to give the URL (host) of the back-end as value with the /api after the URL.

<u>NOTE:</u> Port number is not required for the production.

Example:
Back-end URL = https://ams-be-app-production.up.railway.app

- `VUE_APP_API_URL=https://ams-be-app-production.up.railway.app/api`
- `VUE_APP_SOCKET_URL=wss://ams-be-app-production.up.railway.app/api`



```
VUE_APP_API_URL=https://ams-be-app-production.up.railway.app/api
VUE_APP_SOCKET_URL=wss://ams-be-app-production.up.railway.app/api
```

Back-end

Dependencies used

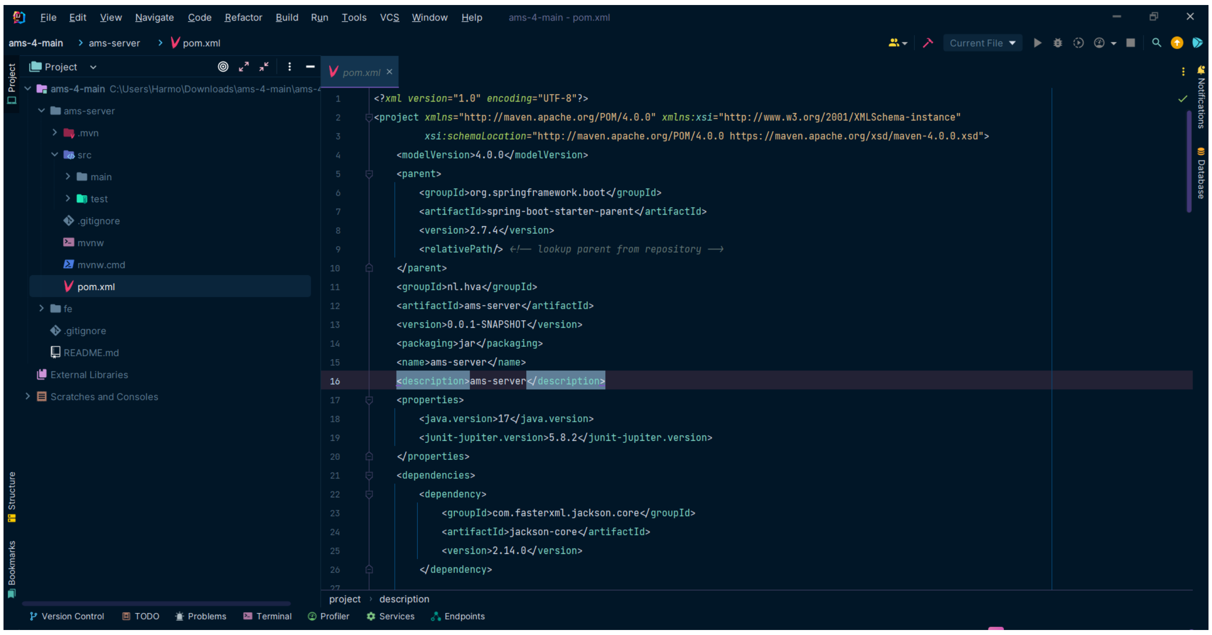
The dependencies can be found in ams-server\pom.xml

Group ID	**Artifactid**	**Version**
com.fasterxml.jackson.core	jackson-core	2.14.0
org.springframework.boot	spring-boot-starter-web	2.7.4
org.springframework.boot	spring-boot-devtools	2.7.4 (runtime)
org.junit.jupiter	junit-jupiter	5.8.2 (test)
org.springframework.boot	spring-boot-starter-test	2.7.4 (test)
com.h2database	h2	2.1.214 (runtime)
org.springframework.boot	spring-boot-starter-websocket	2.7.4
org.springframework.boot	spring-boot-starter-data-jpa	2.7.4
mysql	mysql-connector-java	8.0.30 (runtime)
com.google.code.gson	gson	2.9.1
com.vladmihalcea	hibernate-types-52	2.20.0
io.jsonwebtoken	jjwt-api	0.11.2
io.jsonwebtoken	jjwt-impl	0.11.2
io.jsonwebtoken	jjwt-jackson	0.11.2 (runtime)

Installation

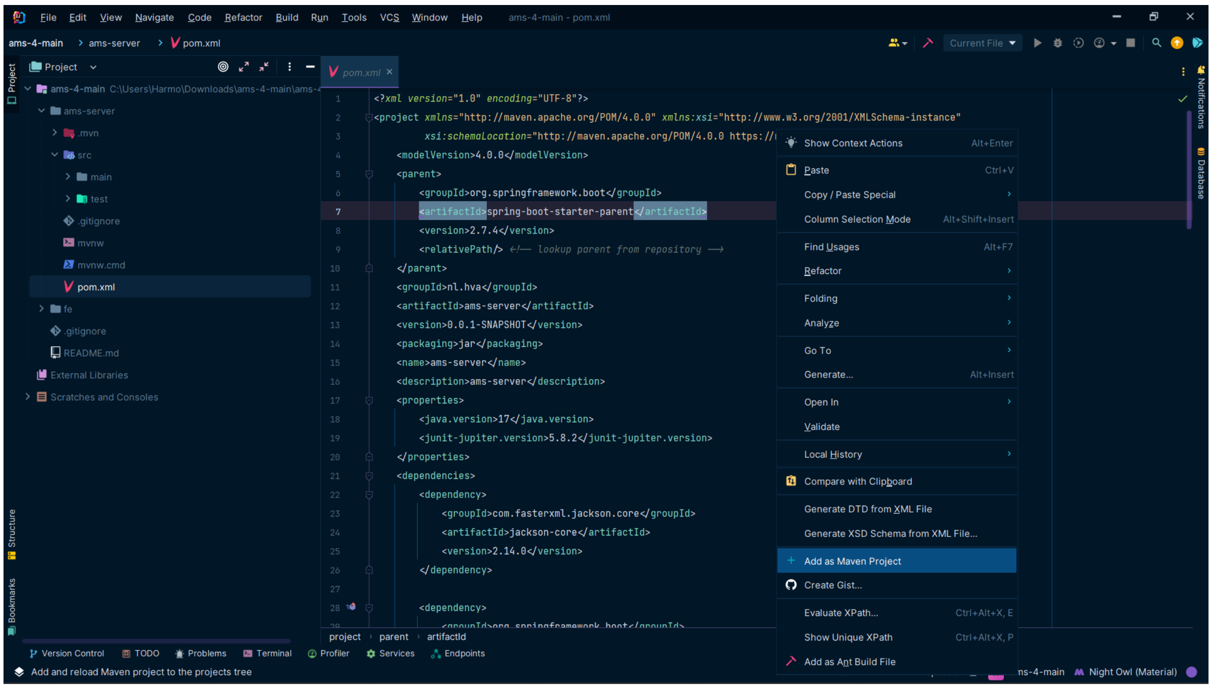
All the back-end related code is inside the *ams-server* directory.

1. Open the *pom.xml* file in IntelliJ.



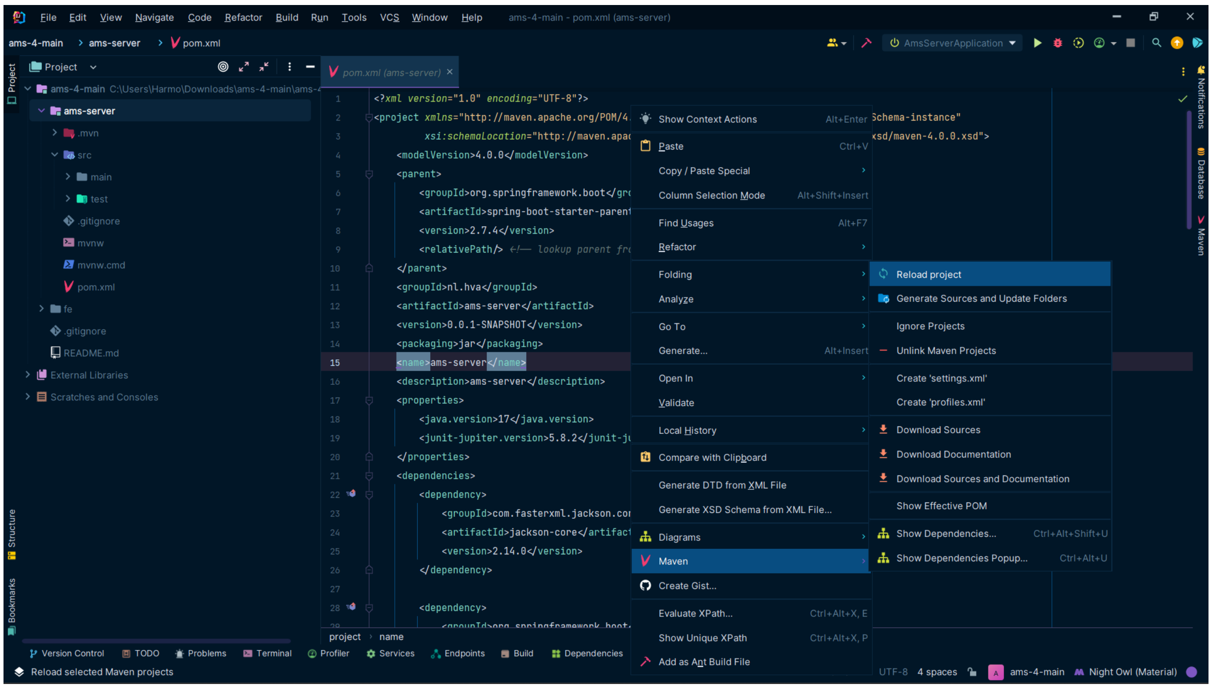
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.4</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>nl.hva</groupId>
    <artifactId>ams-server</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>ams-server</name>
    <description>ams-server</description>
    <properties>
        <java.version>17</java.version>
        <junit-jupiter.version>5.8.2</junit-jupiter.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-core</artifactId>
            <version>2.14.0</version>
        </dependency>
```

2. Inside the *pom.xml* file right-click and in the options click for `Add as Maven Project`



Show Context Actions Alt+Enter
Paste Ctrl+V
Copy / Paste Special
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Refactor
Folding
Analyze
Go To
Generate... Alt+Insert
Open In
Validate
Local History
Compare with Clipboard
Generate DTD from XML File
Generate XSD Schema from XML File...
Add as Maven Project
Create Gist...
Evaluate XPath... Ctrl+Alt+X, E
Show Unique XPath Ctrl+Alt+X, P
Add as Ant Build File
project › parent › artifactId
Add and reload Maven project to the projects tree
Night Owl (Material)

3. Inside the *pom.xml* file right-click and choose `Maven > Reload project`



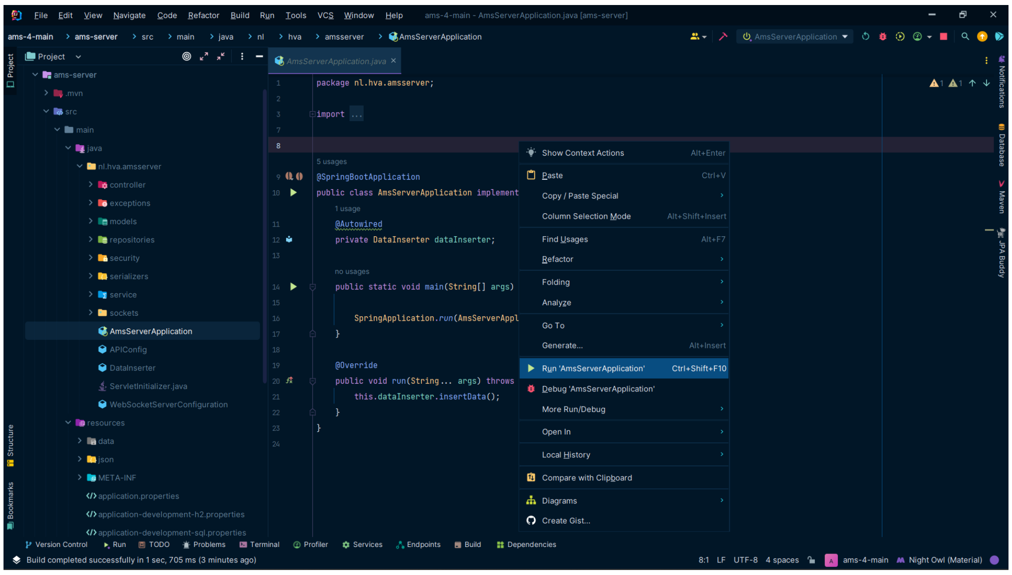
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.                    Schema-instance"
         xsi:schemaLocation="http://maven.apac                    xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</gro
        <artifactId>spring-boot-starter-parent
        <version>2.7.4</version>
        <relativePath/> <!-- lookup parent fro
    </parent>
    <groupId>nl.hva</groupId>
    <artifactId>ams-server</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>ams-server</name>
    <description>ams-server</description>
    <properties>
        <java.version>17</java.version>
        <junit-jupiter.version>5.8.2</junit-ju
    </properties>
    <dependencies>
        <dependency>
            <groupId>com.fasterxml.jackson.cor
            <artifactId>jackson-core</artifact
            <version>2.14.0</version>
        </dependency>

        <dependency>
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ams-4-main - pom.xml (ams-server)
ams-4-main > ams-server > pom.xml
AmsServerApplication
Project
ams-4-main C:\Users\Harmo\Downloads\ams-4-main\ams-
ams-server
.mvn
src
main
test
.gitignore
mvnw
mvnw.cmd
pom.xml
fe
.gitignore
README.md
External Libraries
Scratches and Consoles
pom.xml (ams-server)
Show Context Actions Alt+Enter
Paste Ctrl+V
Copy / Paste Special
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Refactor
Folding
Analyze
Go To
Generate... Alt+Insert
Open In
Validate
Local History
Compare with Clipboard
Generate DTD from XML File
Generate XSD Schema from XML File...
Diagrams
Maven
Create Gist...
Evaluate XPath... Ctrl+Alt+X, E
Show Unique XPath Ctrl+Alt+X, P
Add as Ant Build File
Reload project
Generate Sources and Update Folders
Ignore Projects
Unlink Maven Projects
Create 'settings.xml'
Create 'profiles.xml'
Download Sources
Download Documentation
Download Sources and Documentation
Show Effective POM
Show Dependencies... Ctrl+Alt+Shift+U
Show Dependencies Popup... Ctrl+Alt+U
project > name
Version Control TODO Problems Terminal Profiler Services Endpoints Build Dependencies
Reload selected Maven projects
UTF-8 4 spaces ams-4-main Night Owl (Material)

4. Navigate to *ams-server\src\main\resources* in IntelliJ.
    a. Right-click on resources > New > File
        i. Filename: *application.properties*
        ii. Add the following data in application.properties:

```
spring.profiles.active=development-h2
#spring.profiles.active=development-sql
#spring.profiles.active=production

jwt.passphrase=This is very secret information for my private encryption key. However,
this story still is too short for truly secure 512 bit encryption.
jwt.duration-of-validity=1200
jwt.issuer=EWA
```

    b. The lines which starts with `#` are disabled. You only can have **one** active *spring.profiles.active.*
5. Navigate to *ams-server\src\main\java\nl\hva\amsserver* in IntelliJ.
    a. Open the file called *AmsServerApplication*
        i. Right-click in the file.
        ii. Click the option `Run AmsServerApplication`
        iii. The back-end is now running at `http://localhost:8083/`



```
package nl.hva.amsserver;

import ...


5 usages
@SpringBootApplication
public class AmsServerApplication implement
    1 usage
    @Autowired
    private DataInserter dataInserter;

    no usages
    public static void main(String[] args)

        SpringApplication.run(AmsServerAppl
    }

    @Override
    public void run(String... args) throws
        this.dataInserter.insertData();
    }
}
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ams-4-main - AmsServerApplication.java [ams-server]
ams-4-main > ams-server > src > main > java > nl > hva > amsserver > AmsServerApplication
AmsServerApplication.java
Project
ams-server
.mvn
src
main
java
nl.hva.amsserver
controller
exceptions
models
repositories
security
serializers
service
sockets
AmsServerApplication
APIConfig
DataInserter
ServletInitializer.java
WebSocketServerConfiguration
resources
data
json
META-INF
application.properties
application-development-h2.properties
application-development-sql.properties
Show Context Actions Alt+Enter
Paste Ctrl+V
Copy / Paste Special
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Refactor
Folding
Analyze
Go To
Generate... Alt+Insert
Run 'AmsServerApplication' Ctrl+Shift+F10
Debug 'AmsServerApplication'
More Run/Debug
Open In
Local History
Compare with Clipboard
Diagrams
Create Gist...
Version Control Run TODO Problems Terminal Profiler Services Endpoints Build Dependencies
Build completed successfully in 1 sec, 705 ms (3 minutes ago)
8:1 LF UTF-8 4 spaces ams-4-main Night Owl (Material)

6. If the the active profile in *application.properties* is `development-h2`, you can access the database with following URL: http://localhost:8083/api/h2-console/
    a. The JDBC URL should have the value: jdbc:h2:mem:testdb
    b. Click connect

.properties files

The .properties files are used to store the configurable parameters of the application. In the application there are 4 .properties files, these are:

1. *application.properties*
    a. Decides which active profile to use (h2, sql or production)
    b. Stores the JWT key values
2. *application-development-h2.properties*
    a. Stores the server configuration
    b. Configuration for the h2 in memory database
3. *application-development-sql.properties*
    a. Stores the server configuration
    b. Configuration for the MySQL database
4. *application-production.properties*
    a. Stores the server configuration
    b. Configuration for the MySQL database
        i. DB_URL = The database URL. Depending on your host environment this can be stored as a variable.
        ii. DB_USER = The username to login for the database. Depending on your host environment this can be stored as a variable.
        iii. DB_PASSWORD = The password to login for the database. Depending on your host environment this can be stored as a variable.
        iv.

NOTE: When opening the database you can open the table `account`. You will see there are some initial accounts. The password of these accounts are all the same: 12345678