

Nama: KEVIN OCTAVIANUS HALIM

## Inisilasi Project:

Java SpringBoot

The screenshot shows the Spring Initializr web application interface. The browser address bar displays 'https://start.spring.io'. The page features the 'spring initializr' logo at the top left. The configuration is divided into two main sections: 'Project' and 'Dependencies'.

**Project Configuration:**

- Project:** ☐ Gradle - Groovy, ☐ Gradle - Kotlin, ☒ **Maven**
- Language:** ☒ **Java**, ☐ Kotlin, ☐ Groovy
- Spring Boot:** ☐ 3.3.0 (SNAPSHOT), ☐ 3.3.0 (RC1), ☐ 3.2.6 (SNAPSHOT), ☒ **3.2.5**, ☐ 3.1.12 (SNAPSHOT), ☐ 3.1.11
- Project Metadata:**
  - Group:
  - Artifact:
  - Name:
  - Description:
  - Package name:
  - Packaging: ☒ **Jar**, ☐ War
  - Java: ☐ 22, ☒ **21**, ☐ 17

**Dependencies Configuration:**

- Spring Data JPA** (SQL): Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
- Validation** (I/O): Bean Validation with Hibernate validator.
- Lombok** (DEVELOPER TOOLS): Java annotation library which helps to reduce boilerplate code.
- Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
- MySQL Driver** (SQL): MySQL JDBC driver.

An 'ADD DEPENDENCIES... CTRL + B' button is located at the top right of the Dependencies section.

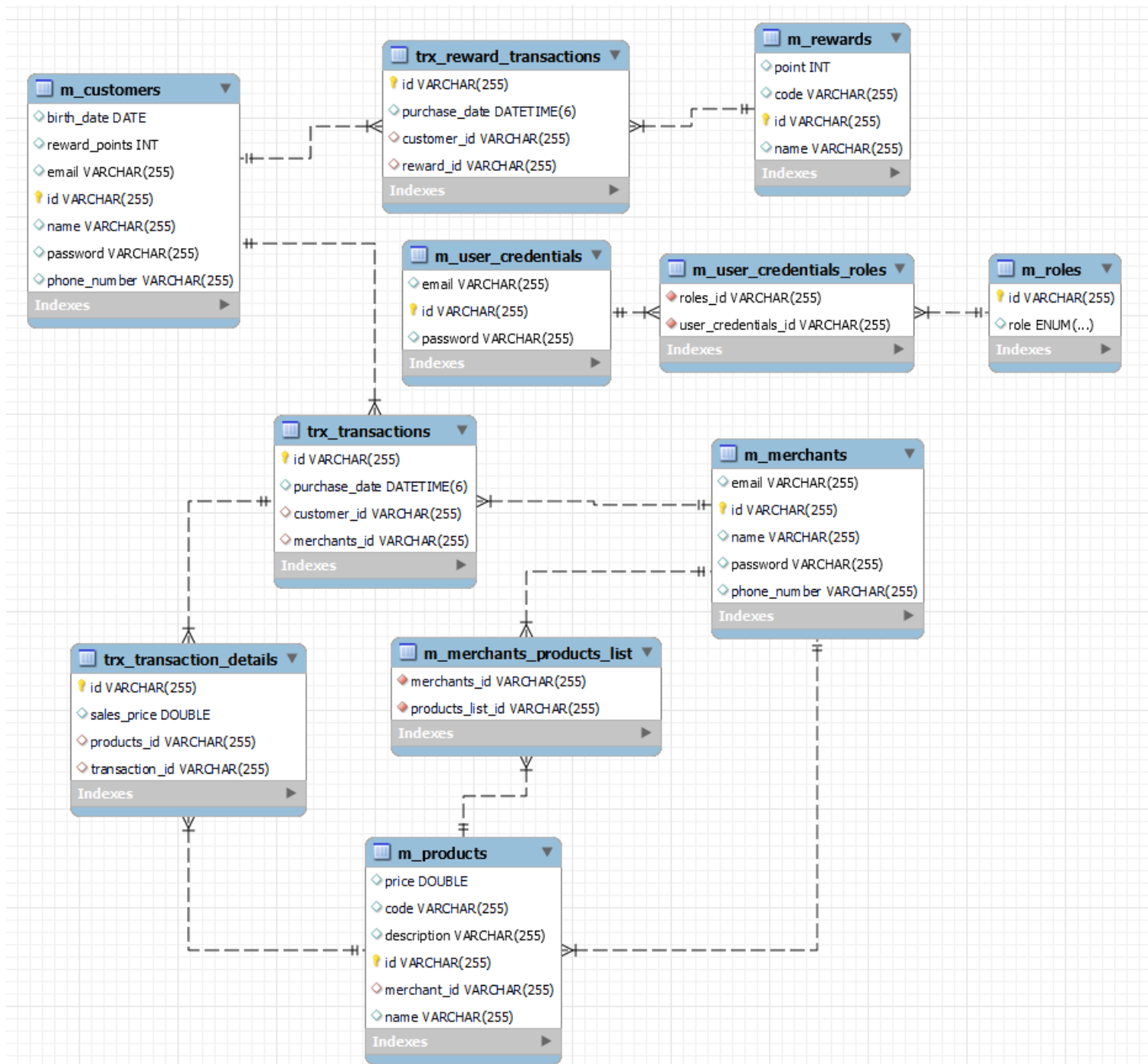
Versi Java yang Digunakan Adalah 21 karena PC saya menggunakan Java versi 21.

## Depedencies:

- Spring Data JPA  
Menggunakan Library Hibernate & Spring Data yang berguna sebagai ORM.
- Validation  
Library Validation yang nantinya dapat membantu penerapan validasi sederhana.
- Lombok  
Untuk mengurangi boilerplate codes seperti Getter & Setter, Constructor.
- Spring Web  
Untuk menggunakan library untuk membantu pembuatan project REST API.
- MySQL Driver  
Driver untuk MySQL yang nantinya akan digunakan sebagai database untuk project ini.

Security Authentication dengan JWT nantinya akan ditambahkan.

## Database Structures



## Pom.xml

### Swagger / OPENAPI

```
<!--      OPENAPI / SWAGGER DEPEDENCIES -->
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
  <version>2.5.0</version>
</dependency>
```

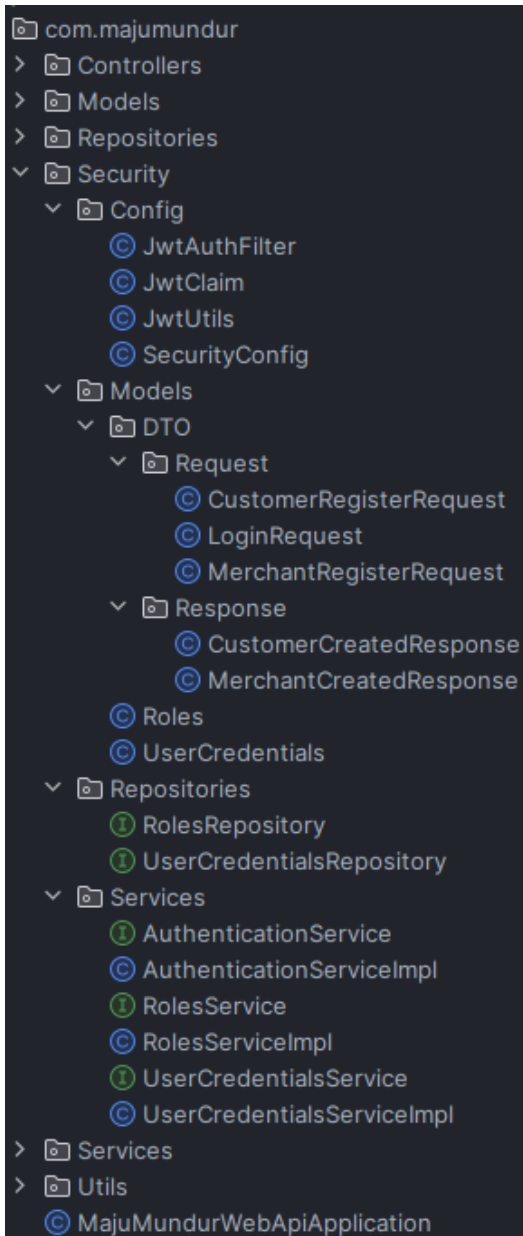
### Security Dependencies

```
<!--      Security Auth Depedencies -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>

<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-core</artifactId>
  <version>6.2.2</version>
</dependency>

<dependency>
  <groupId>com.auth0</groupId>
  <artifactId>java-jwt</artifactId>
  <version>4.4.0</version>
</dependency>
```

## Code Structures



```
com.majumundur
├── Controllers
├── Models
├── Repositories
├── Security
│   ├── Config
│   │   ├── JwtAuthFilter
│   │   ├── JwtClaim
│   │   ├── JwtUtils
│   │   └── SecurityConfig
│   ├── Models
│   │   ├── DTO
│   │   │   ├── Request
│   │   │   │   ├── CustomerRegisterRequest
│   │   │   │   ├── LoginRequest
│   │   │   │   └── MerchantRegisterRequest
│   │   │   └── Response
│   │   │       ├── CustomerCreatedResponse
│   │   │       └── MerchantCreatedResponse
│   │   ├── Roles
│   │   └── UserCredentials
│   ├── Repositories
│   │   ├── RolesRepository
│   │   └── UserCredentialsRepository
│   └── Services
│       ├── AuthenticationService
│       ├── AuthenticationServiceImpl
│       ├── RolesService
│       ├── RolesServiceImpl
│       ├── UserCredentialsService
│       └── UserCredentialsServiceImpl
├── Services
├── Utils
└── MajuMundurWebApiApplication
```

The image shows a hierarchical tree structure of a code project. The root is 'com.majumundur'. It has several sub-packages: 'Controllers', 'Models', 'Repositories', 'Security', 'Services', 'Utils', and 'MajuMundurWebApiApplication'. The 'Security' package is expanded, showing its internal structure: 'Config' (containing 'JwtAuthFilter', 'JwtClaim', 'JwtUtils', and 'SecurityConfig'), 'Models' (containing 'DTO', 'Roles', and 'UserCredentials'), 'Repositories' (containing 'RolesRepository' and 'UserCredentialsRepository'), and 'Services' (containing 'AuthenticationService', 'AuthenticationServiceImpl', 'RolesService', 'RolesServiceImpl', 'UserCredentialsService', and 'UserCredentialsServiceImpl'). The 'DTO' package is further expanded to show 'Request' (with 'CustomerRegisterRequest', 'LoginRequest', and 'MerchantRegisterRequest') and 'Response' (with 'CustomerCreatedResponse' and 'MerchantCreatedResponse').

Package / Folder Security berisikan class yang terkait Authentication seperti Register, Login, Repository untuk UserCredentials, dan lain sebagainya.

- ▼ com.majumundur
  - ▼ Controllers
    - Ⓢ AuthenticationController
    - Ⓢ CustomerController
    - Ⓢ MerchantsController
    - Ⓢ RewardsController
    - Ⓢ TransactionsController
  - ▼ Models
    - > DTO
      - Ⓢ Customers
      - Ⓢ Merchants
      - Ⓢ Products
      - Ⓢ Rewards
      - Ⓢ RewardTransactions
      - Ⓢ TransactionDetails
      - Ⓢ Transactions
  - ▼ Repositories
    - Ⓢ CustomersRepository
    - Ⓢ MerchantsRepository
    - Ⓢ ProductsRepository
    - Ⓢ RewardsRepository
    - Ⓢ RewardTransactionsRepository
    - Ⓢ TransactionDetailsRepository
    - Ⓢ TransactionsRepository
  - > Security
  - ▼ Services
    - Ⓢ CustomersService
    - Ⓢ CustomersServiceImpl
    - Ⓢ MerchantsService
    - Ⓢ MerchantsServiceImpl
    - Ⓢ ProductsService
    - Ⓢ ProductsServiceImpl
    - Ⓢ RewardsService
    - Ⓢ RewardsServiceImpl
    - Ⓢ RewardTransactionsService
    - Ⓢ RewardTransactionsServiceImpl
    - Ⓢ TransactionDetailsService
    - Ⓢ TransactionDetailsServiceImpl
    - Ⓢ TransactionsService
    - Ⓢ TransactionsServiceImpl
    - Ⓢ ValidationService
  - ▼ Utils
    - Ⓢ RoleEnum
    - Ⓢ SwaggerConfig
    - Ⓢ MajuMundurWebApiApplication

## application.properties

```
application.properties x
1  #POSISI TAMPILAN SWAGGER UI
2  springdoc.swagger-ui.path=/api/swagger/swagger-ui.html
3
4  spring.application.name=MajuMundurWebAPI
5
6  spring.datasource.username=${DB_USER:root}
7  spring.datasource.password=${DB_PASS:root}
8  spring.datasource.url=jdbc:mysql://${DB_HOST:localhost}:${DB_PORT:3306}/${DB_NAME:majumundurdb}
9  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
10
11 spring.jpa.hibernate.ddl-auto=update
12
13 jwt.secret.key = ${JWT_KEY:key}
14 jwt.expiration = ${JWT_EXPIRATION:1200}
15 jwt.app.name = ${JWT_APP_NAME:majumundurAPI}
```

untuk mengakses Swagger UI bisa membuka link

<http://localhost:8080/api/swagger/swagger-ui/index.html>

“spring.jpa.hibernate.ddl-auto=update” merupakan command untuk hibernate agar melakukan update pada DDL jika ada perubahan pada struktur models / table.

Alternatifnya bisa gunakan “spring.jpa.hibernate.ddl-auto=create” agar table selalu dibuat baru sehingga data pada database akan dihapus setiap kali menjalankan aplikasi.