Subtask 1

Download Docker from docker.com
- https://www.docker.com/get-started

Pull latest postgres image
- docker pull postgres

Run image with correct parameters
- docker run -d -p 5432:5432 --name 223postgres -e POSTGRES_PASSWORD=pwd postgres

Check if Docker container is running
- docker ps

Open JetBrains DataGrip and connect to postgres
- jdbc:postgresql://localhost:5432/postgres

Subtask 2

Now it is time to prepare Spring Boot to be able to connect with the container created above and generate the needed relations and relationships based on our java classes. In order to do so we have to enrich our business objects with the relevant hibernate annotations and add the repositories. To see what that looks like with products elaborate on the code below.
HINT: Add Spring Data and PostgreSQL Drivers as dependencies to the file build.gradle.

Before we dedicate ourselves to Product.java we want to create a class ProductRepository.java. Make sure you configure the implemented interface right as shown below. The second parameter Integer declares the type of the primary key being used in the business object Product.java.

```java
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ProductRepository extends JpaRepository<Product, Integer> {
}
```

To make sure that Spring Boot can successfully link the ProductRepository.java with the business object Product.java with have to add the annotation @Entity. @Id labels the attribute id as primary key.

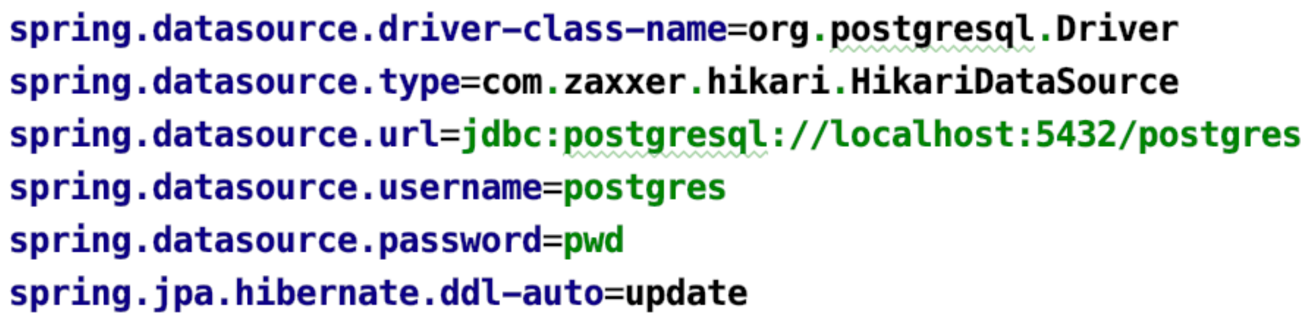
```java
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Product {

    @Id
    private int id;
```

Subtask 3

Before we can let everything run and see the magic happen we have to add the necessary lines to the file application.properties located in the resources folder.



```
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.type=com.zaxxer.hikari.HikariDataSource
spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
spring.datasource.username=postgres
spring.datasource.password=pwd
spring.jpa.hibernate.ddl-auto=update
```

HINT: Verify that you use the same credentials as you initially passed to docker run.

If everything runs the way it's supposed to, DataGrid should now list you a relation called product together with all attributes.

Subtask 4

Last but not least we add the Class user.java to the project and see how Spring Boot can help us when it comes to assigning roles and authorities. This is done in plenum, however.