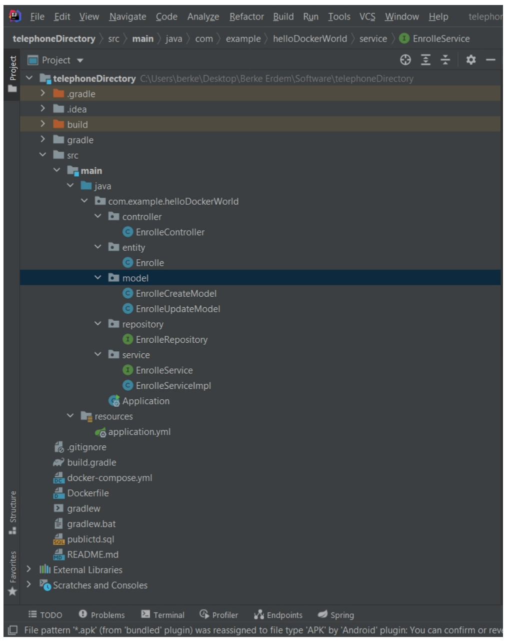
TELEPHONE DIRECTORY APPLICATION

Introduction :

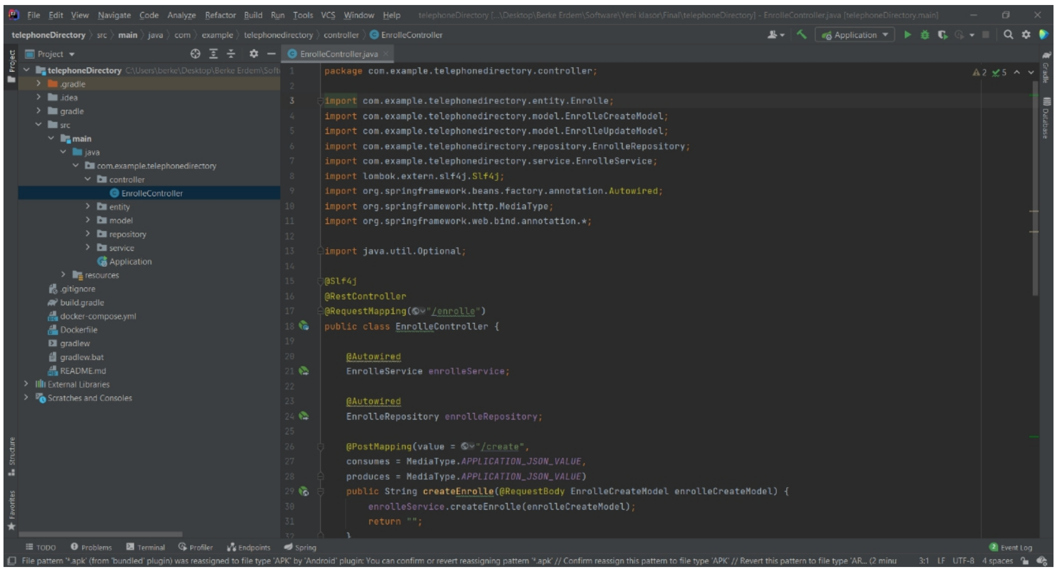
Completed the telephone directory project using Java-Spring Boot. In the project, rest services, entity, models (create and update), repository, service, service implementations and finally controller were written.



```
telephoneDirectory C:\Users\berke\Desktop\Berke Erdem\Software\telephoneDirectory
  .gradle
  .idea
  build
  gradle
  src
    main
      java
        com.example.helloDockerWorld
          controller
            EnrolleController
          entity
            Enrolle
          model
            EnrolleCreateModel
            EnrolleUpdateModel
          repository
            EnrolleRepository
          service
            EnrolleService
            EnrolleServiceImpl
          Application
      resources
        application.yml
  .gitignore
  build.gradle
  docker-compose.yml
  Dockerfile
  gradlew
  gradlew.bat
  publictd.sql
  README.md
External Libraries
Scratches and Consoles
```

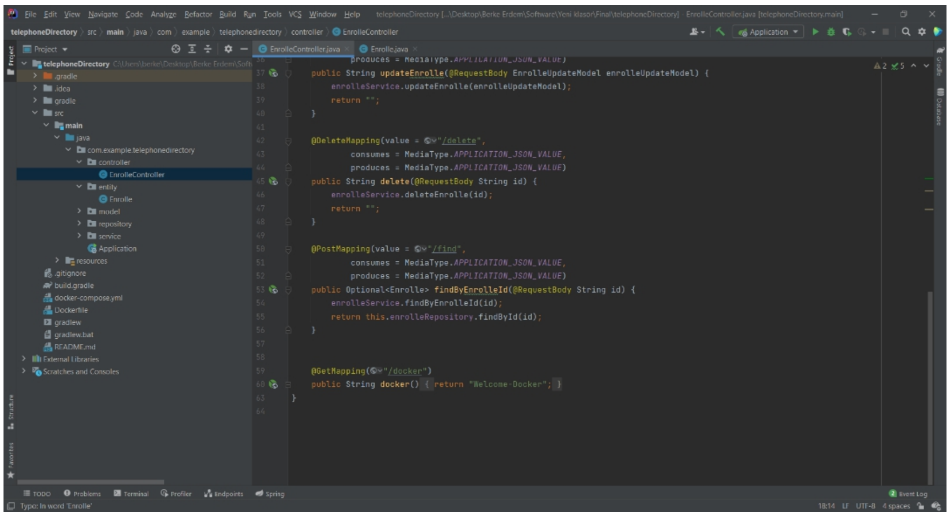
File pattern '*.apk' (from 'bundled' plugin) was reassigned to file type 'APK' by 'Android' plugin: You can confirm or rev

Fig.1. Project Files



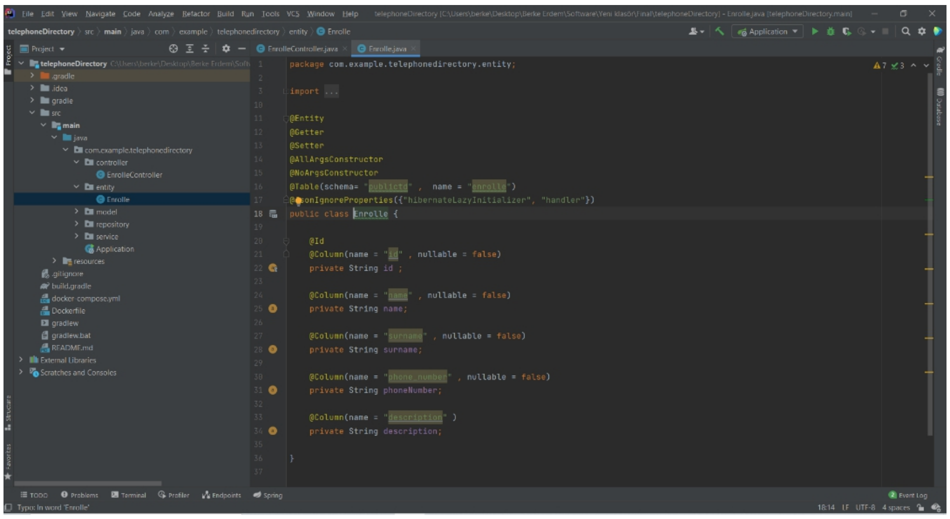
```java
package com.example.telephonedirectory.controller;

import com.example.telephonedirectory.entity.Enrolle;
import com.example.telephonedirectory.model.EnrolleCreateModel;
import com.example.telephonedirectory.model.EnrolleUpdateModel;
import com.example.telephonedirectory.repository.EnrolleRepository;
import com.example.telephonedirectory.service.EnrolleService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.*;

import java.util.Optional;

@Slf4j
@RestController
@RequestMapping("/enrolle")
public class EnrolleController {

    @Autowired
    EnrolleService enrolleService;

    @Autowired
    EnrolleRepository enrolleRepository;

    @PostMapping(value = "/create",
    consumes = MediaType.APPLICATION_JSON_VALUE,
    produces = MediaType.APPLICATION_JSON_VALUE)
    public String createEnrolle(@RequestBody EnrolleCreateModel enrolleCreateModel) {
        enrolleService.createEnrolle(enrolleCreateModel);
        return "";
    }
```

Fig.2. Enrolle Controller Class



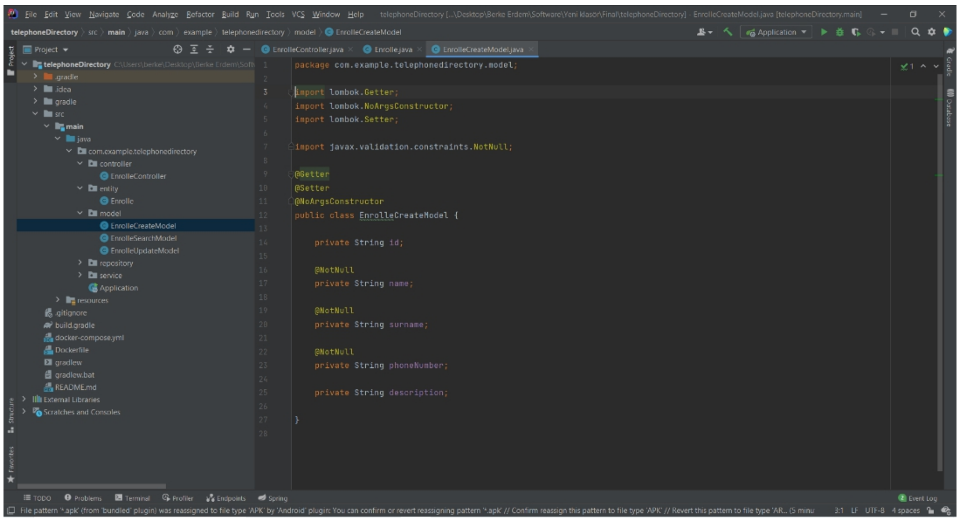
```java
            produces = MediaType.APPLICATION_JSON_VALUE)
    public String updateEnrolle(@RequestBody EnrolleUpdateModel enrolleUpdateModel) {
        enrolleService.updateEnrolle(enrolleUpdateModel);
        return "";
    }

    @DeleteMapping(value = "/delete",
            consumes = MediaType.APPLICATION_JSON_VALUE,
            produces = MediaType.APPLICATION_JSON_VALUE)
    public String delete(@RequestBody String id) {
        enrolleService.deleteEnrolle(id);
        return "";
    }

    @PostMapping(value = "/find",
            consumes = MediaType.APPLICATION_JSON_VALUE,
            produces = MediaType.APPLICATION_JSON_VALUE)
    public Optional<Enrolle> findByEnrolleId(@RequestBody String id) {
        enrolleService.findByEnrolleId(id);
        return this.enrolleRepository.findById(id);
    }


    @GetMapping("/docker")
    public String docker() { return "Welcome-Docker"; }
}
```

Fig.3. Enrolle Controller Class



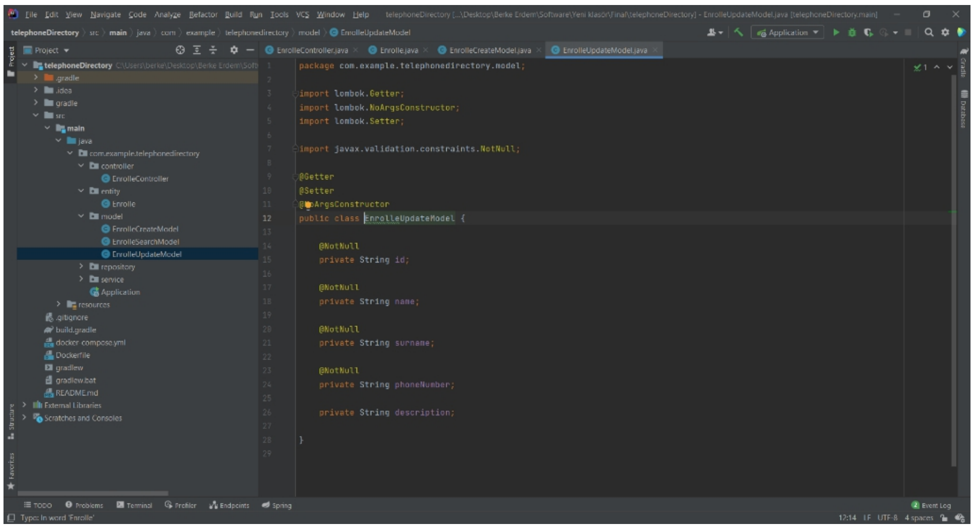
```java
package com.example.telephonedirectory.entity;

import ...

@Entity
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Table(schema= "publictd" ,  name = "enrolle")
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
public class Enrolle {

    @Id
    @Column(name = "id" , nullable = false)
    private String id ;

    @Column(name = "name" , nullable = false)
    private String name;

    @Column(name = "surname" , nullable = false)
    private String surname;

    @Column(name = "phone_number" , nullable = false)
    private String phoneNumber;

    @Column(name = "description" )
    private String description;

}
```

Fig.4. Enrolle Entity Class



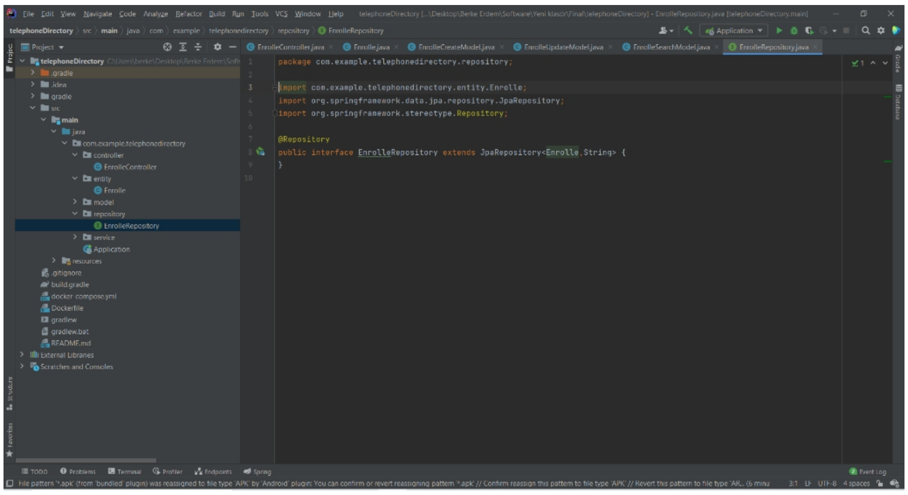
```java
package com.example.telephonedirectory.model;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import javax.validation.constraints.NotNull;

@Getter
@Setter
@NoArgsConstructor
public class EnrolleCreateModel {

    private String id;

    @NotNull
    private String name;

    @NotNull
    private String surname;

    @NotNull
    private String phoneNumber;

    private String description;

}
```

Fig.4. Enrolle Create Model Class



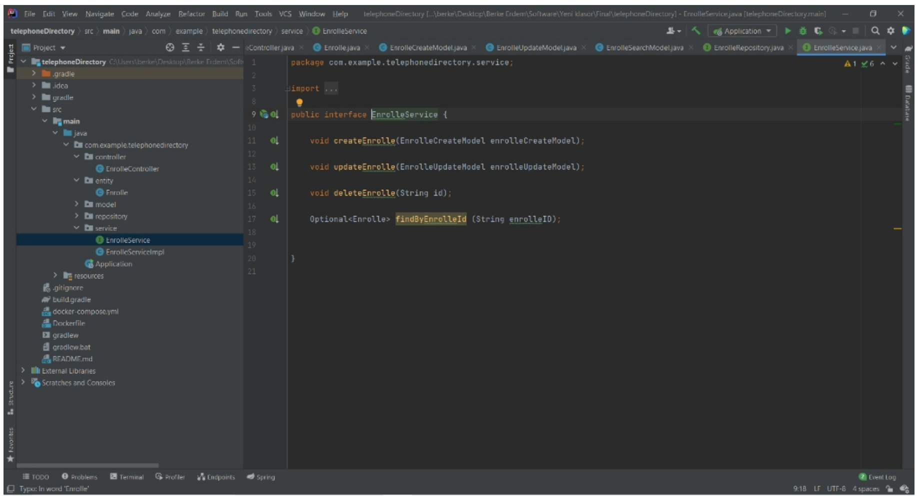
```java
package com.example.telephonedirectory.model;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import javax.validation.constraints.NotNull;

@Getter
@Setter
@NoArgsConstructor
public class EnrolleUpdateModel {

    @NotNull
    private String id;

    @NotNull
    private String name;

    @NotNull
    private String surname;

    @NotNull
    private String phoneNumber;

    private String description;

}
```

Fig.5. Enrolle Update Model Class



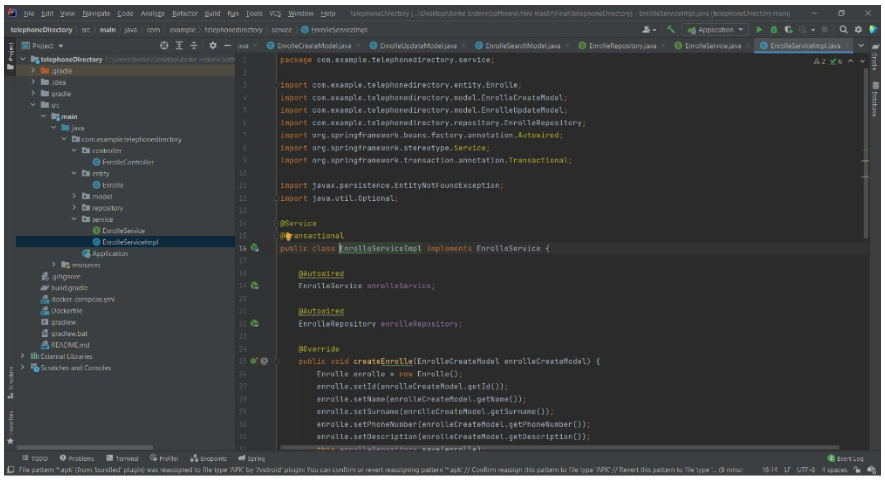
```java
package com.example.telephonedirectory.repository;

import com.example.telephonedirectory.entity.Enrolle;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface EnrolleRepository extends JpaRepository<Enrolle,String> {
}
```

Fig.6. Enrolle Repository Class



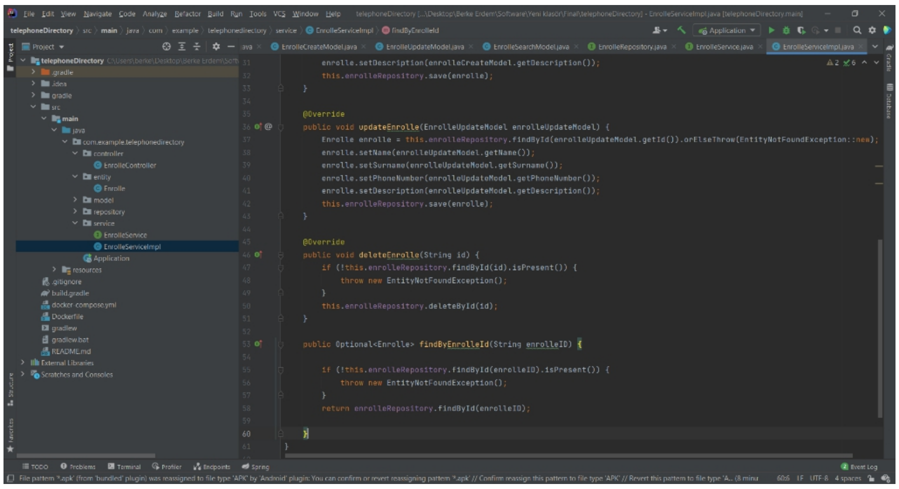
```java
package com.example.telephonedirectory.service;

import ...

public interface EnrolleService {

    void createEnrolle(EnrolleCreateModel enrolleCreateModel);

    void updateEnrolle(EnrolleUpdateModel enrolleUpdateModel);

    void deleteEnrolle(String id);

    Optional<Enrolle> findByEnrolleId (String enrolleID);


}
```

Fig.7. Enrolle Service Class



```java
package com.example.telephonedirectory.service;

import com.example.telephonedirectory.entity.Enrolle;
import com.example.telephonedirectory.model.EnrolleCreateModel;
import com.example.telephonedirectory.model.EnrolleUpdateModel;
import com.example.telephonedirectory.repository.EnrolleRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import javax.persistence.EntityNotFoundException;
import java.util.Optional;

@Service
@Transactional
public class EnrolleServiceImpl implements EnrolleService {

    @Autowired
    EnrolleService enrolleService;

    @Autowired
    EnrolleRepository enrolleRepository;

    @Override
    public void createEnrolle(EnrolleCreateModel enrolleCreateModel) {
        Enrolle enrolle = new Enrolle();
        enrolle.setId(enrolleCreateModel.getId());
        enrolle.setName(enrolleCreateModel.getName());
        enrolle.setSurname(enrolleCreateModel.getSurname());
        enrolle.setPhoneNumber(enrolleCreateModel.getPhoneNumber());
        enrolle.setDescription(enrolleCreateModel.getDescription());
        this.enrolleRepository.save(enrolle);
```

Fig.7. Enrolle Service Implementation Class(1)

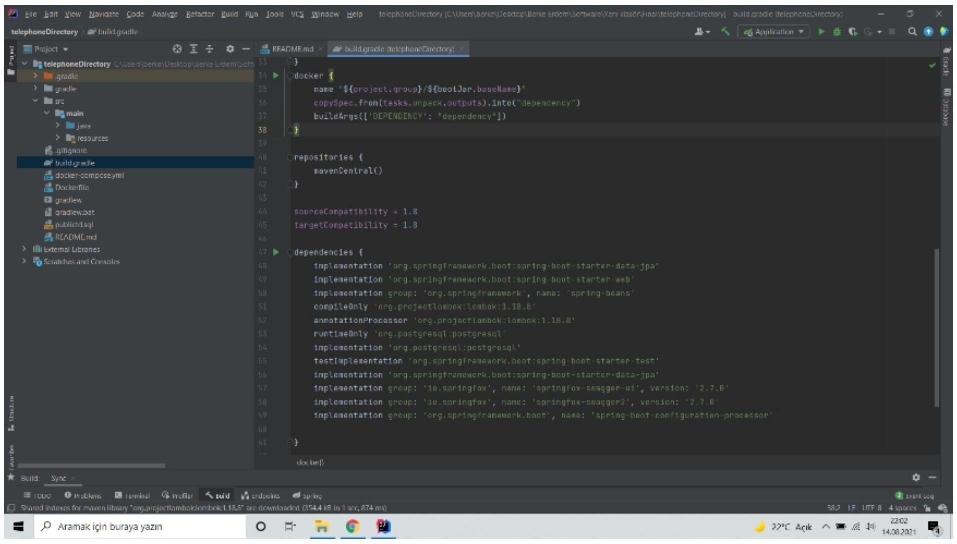


```java
        enrolle.setDescription(enrolleCreateModel.getDescription());
        this.enrolleRepository.save(enrolle);
    }

    @Override
    public void updateEnrolle(EnrolleUpdateModel enrolleUpdateModel) {
        Enrolle enrolle = this.enrolleRepository.findById(enrolleUpdateModel.getId()).orElseThrow(EntityNotFoundException::new);
        enrolle.setName(enrolleUpdateModel.getName());
        enrolle.setSurname(enrolleUpdateModel.getSurname());
        enrolle.setPhoneNumber(enrolleUpdateModel.getPhoneNumber());
        enrolle.setDescription(enrolleUpdateModel.getDescription());
        this.enrolleRepository.save(enrolle);
    }

    @Override
    public void deleteEnrolle(String id) {
        if (!this.enrolleRepository.findById(id).isPresent()) {
            throw new EntityNotFoundException();
        }
        this.enrolleRepository.deleteById(id);
    }

    public Optional<Enrolle> findByEnrolleId(String enrolleID) {

        if (!this.enrolleRepository.findById(enrolleID).isPresent()) {
            throw new EntityNotFoundException();
        }
        return enrolleRepository.findById(enrolleID);

    }
}
```

Fig.8. Enrolle Service Implementation Class(2)

Create method saves the data entered in JSON format to the database, with the Delete method, the data is deleted in the database according to the information of the data registered in the database, The data registered in the update method database is updated with JSON format and The FindbyId method, on the other hand, presents the information available in the database according to the id in JSON format.

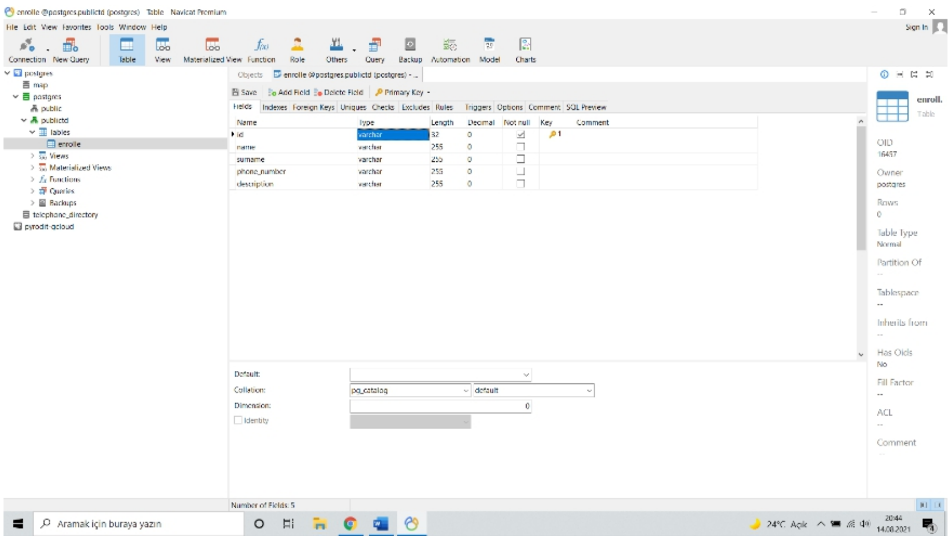
Gradle was used in the project. It was the first experience.



```groovy
}
docker {
    name "${project.group}/${bootJar.baseName}"
    copySpec.from(tasks.unpack.outputs).into("dependency")
    buildArgs(['DEPENDENCY': "dependency"])
}

repositories {
    mavenCentral()
}

sourceCompatibility = 1.8
targetCompatibility = 1.8

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation group: 'org.springframework', name: 'spring-beans'
    compileOnly 'org.projectlombok:lombok:1.18.8'
    annotationProcessor 'org.projectlombok:lombok:1.18.8'
    runtimeOnly 'org.postgresql:postgresql'
    implementation 'org.postgresql:postgresql'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation group: 'io.springfox', name: 'springfox-swagger-ui', version: '2.7.0'
    implementation group: 'io.springfox', name: 'springfox-swagger2', version: '2.7.0'
    implementation group: 'org.springframework.boot', name: 'spring-boot-configuration-processor'

}
```

Fig.9. Build Gradle File

The data was kept in the database. PostgreSQL is used. In the Navicat app.Database interaction is provided using Spring-Data.

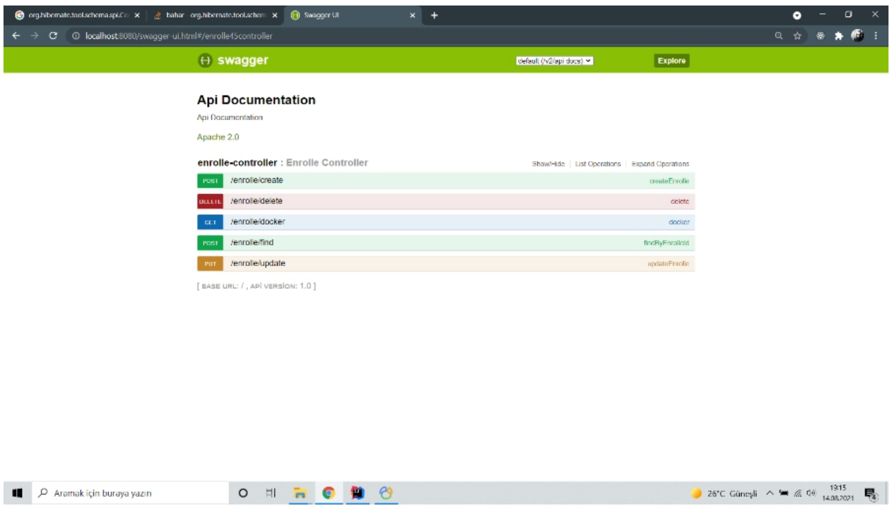


enrolle @postgres.publictd (postgres) - ...

Name	Type	Length	Decimal	Not null	Key	Comment
id	varchar	32	0	✓	1	
name	varchar	255	0			
surname	varchar	255	0			
phone_number	varchar	255	0			
description	varchar	255	0			

Fig.10. Navicat Enrolle Table

Swagger was added to the project and the tests were done using swagger. (Additional checks were made in Postman.)



localhost:8080/swagger-ui.html#/enrolle45controller

Api Documentation

Api Documentation

Apache 2.0

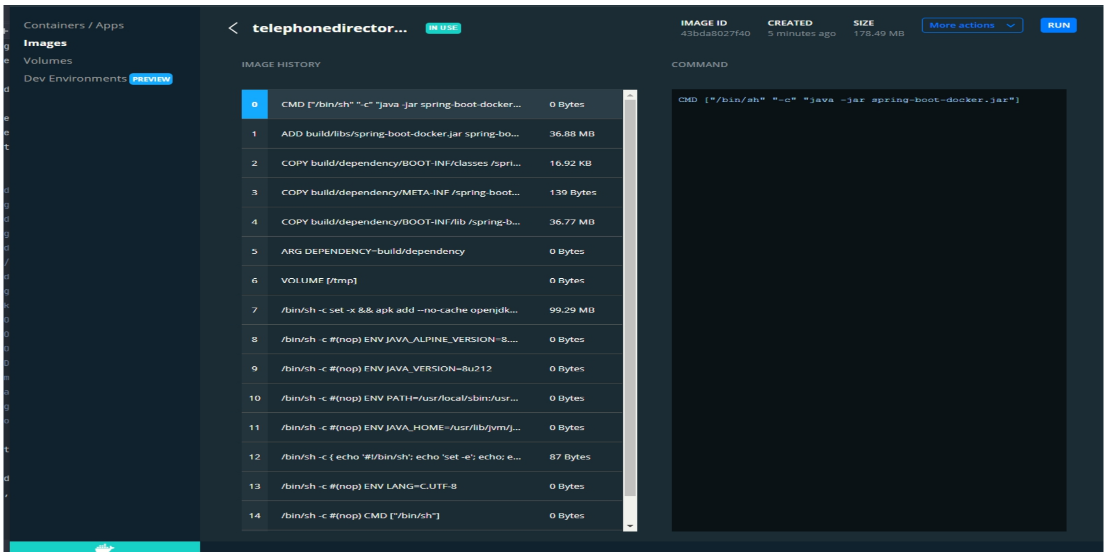
enrolle-controller : Enrolle Controller

```
POST    /enrolle/create    createEnrolle
DELETE  /enrolle/delete    delete
GET     /enrolle/docker    docker
POST    /enrolle/find      findByEnrolleId
PUT     /enrolle/update    updateEnrolle
```

[BASE URL: / , API VERSION: 1.0]

Fig.11. Swagger Dependency

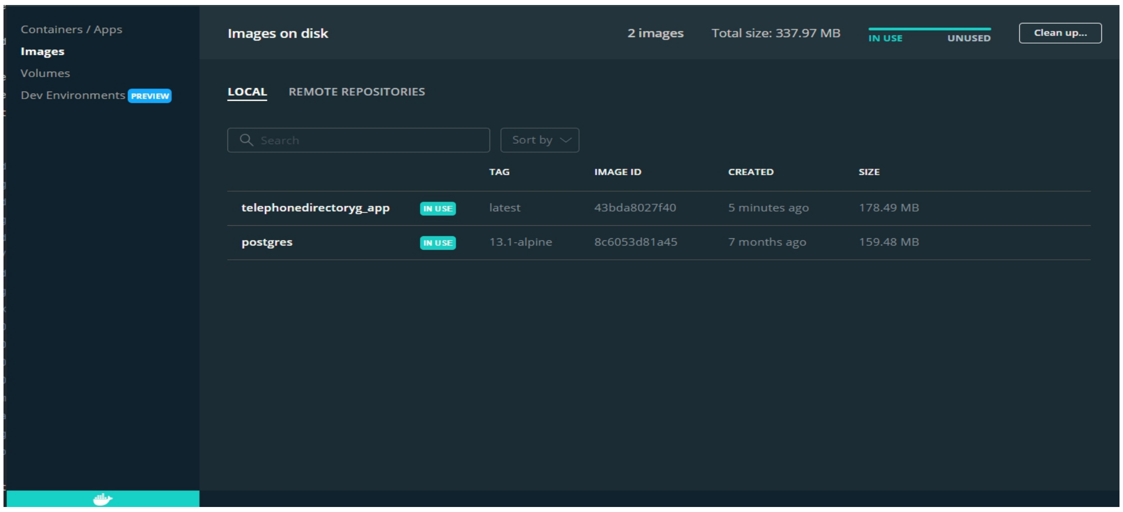
There were difficulties in the Docker part of the project, but the project was supported by using medium articles, udemy courses and web applications such as youtube and the Docker part was completed.



0	CMD ["/bin/sh" "-c" "java -jar spring-boot-docker...	0 Bytes
1	ADD build/libs/spring-boot-docker.jar spring-bo...	36.88 MB
2	COPY build/dependency/BOOT-INF/classes /spri...	16.92 KB
3	COPY build/dependency/META-INF /spring-boot...	139 Bytes
4	COPY build/dependency/BOOT-INF/lib /spring-b...	36.77 MB
5	ARG DEPENDENCY=build/dependency	0 Bytes
6	VOLUME [/tmp]	0 Bytes
7	/bin/sh -c set -x && apk add --no-cache openjdk...	99.29 MB
8	/bin/sh -c #(nop) ENV JAVA_ALPINE_VERSION=8....	0 Bytes
9	/bin/sh -c #(nop) ENV JAVA_VERSION=8u212	0 Bytes
10	/bin/sh -c #(nop) ENV PATH=/usr/local/sbin:/usr...	0 Bytes
11	/bin/sh -c #(nop) ENV JAVA_HOME=/usr/lib/jvm/j...	0 Bytes
12	/bin/sh -c { echo '#!/bin/sh'; echo 'set -e'; echo; e...	87 Bytes
13	/bin/sh -c #(nop) ENV LANG=C.UTF-8	0 Bytes
14	/bin/sh -c #(nop) CMD ["/bin/sh"]	0 Bytes

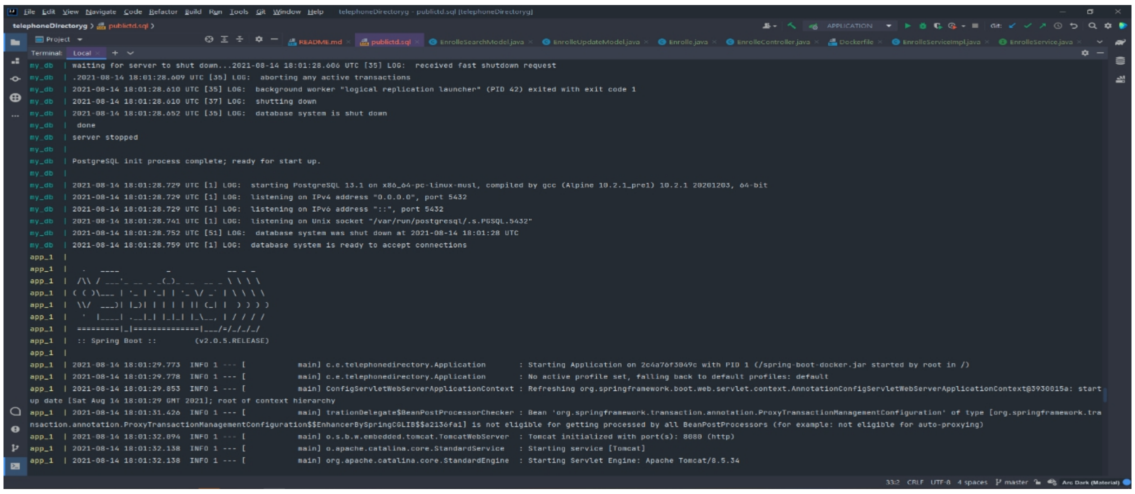
```
CMD ["/bin/sh" "-c" "java -jar spring-boot-docker.jar"]
```

Fig.12. Image History



	TAG	IMAGE ID	CREATED	SIZE
telephonedirectoryg_app IN USE	latest	43bda8027f40	5 minutes ago	178.49 MB
postgres IN USE	13.1-alpine	8c6053d81a45	7 months ago	159.48 MB

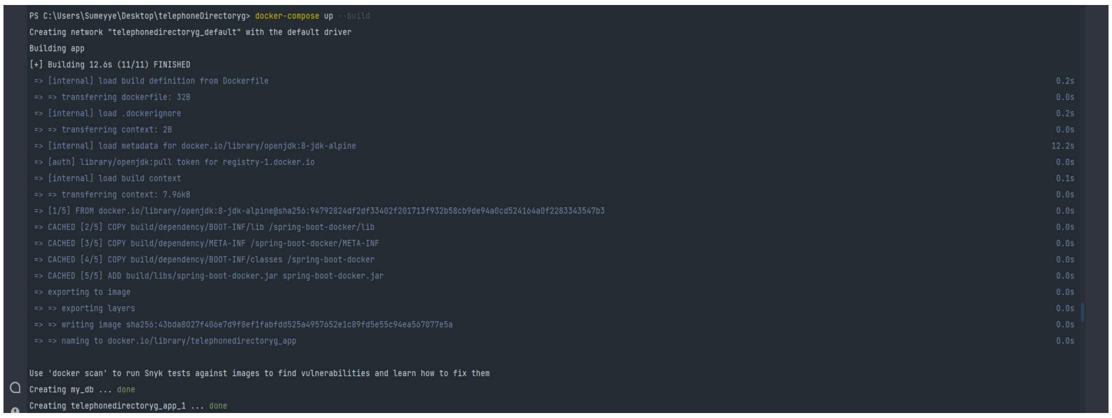
Fig.13. Images on Disc



File Edit View Navigate Code Refactor Build Run Tools Git Window Help telephoneDirectoryg - publictd.sql [telephoneDirectoryg]
telephoneDirectoryg › publictd.sql ›
Project
README.md
publictd.sql
EnrolleSearchModel.java
EnrolleUpdateModel.java
Enrolle.java
EnrolleController.java
Dockerfile
EnrolleServiceImpl.java
EnrolleService.java
APPLICATION
Terminal: Local
33:2 CRLF UTF-8 4 spaces master Arc Dark (Material)

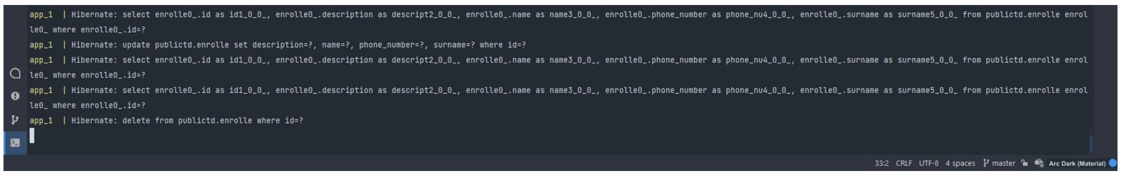
```
my_db  | waiting for server to shut down...2021-08-14 18:01:28.606 UTC [35] LOG:  received fast shutdown request
my_db  | .2021-08-14 18:01:28.609 UTC [35] LOG:  aborting any active transactions
my_db  | 2021-08-14 18:01:28.610 UTC [35] LOG:  background worker "logical replication launcher" (PID 42) exited with exit code 1
my_db  | 2021-08-14 18:01:28.610 UTC [37] LOG:  shutting down
my_db  | 2021-08-14 18:01:28.652 UTC [35] LOG:  database system is shut down
my_db  |  done
my_db  | server stopped
my_db  |
my_db  | PostgreSQL init process complete; ready for start up.
my_db  |
my_db  | 2021-08-14 18:01:28.729 UTC [1] LOG:  starting PostgreSQL 13.1 on x86_64-pc-linux-musl, compiled by gcc (Alpine 10.2.1_pre1) 10.2.1 20201203, 64-bit
my_db  | 2021-08-14 18:01:28.729 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
my_db  | 2021-08-14 18:01:28.729 UTC [1] LOG:  listening on IPv6 address "::", port 5432
my_db  | 2021-08-14 18:01:28.741 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
my_db  | 2021-08-14 18:01:28.752 UTC [51] LOG:  database system was shut down at 2021-08-14 18:01:28 UTC
my_db  | 2021-08-14 18:01:28.759 UTC [1] LOG:  database system is ready to accept connections
app_1  |
app_1  |   .   ____          _            __ _ _
app_1  |  /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
app_1  | ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
app_1  |  \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
app_1  |   '  |____| .__|_| |_|_| |_\__, | / / / /
app_1  |  =========|_|==============|___/=/_/_/_/
app_1  |  :: Spring Boot ::        (v2.0.5.RELEASE)
app_1  |
app_1  | 2021-08-14 18:01:29.773  INFO 1 --- [           main] c.e.telephonedirectory.Application       : Starting Application on 2c4a76f3049c with PID 1 (/spring-boot-docker.jar started by root in /)
app_1  | 2021-08-14 18:01:29.778  INFO 1 --- [           main] c.e.telephonedirectory.Application       : No active profile set, falling back to default profiles: default
app_1  | 2021-08-14 18:01:29.853  INFO 1 --- [           main] ConfigServletWebServerApplicationContext : Refreshing org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@3930015a: start
up date [Sat Aug 14 18:01:29 GMT 2021]; root of context hierarchy
app_1  | 2021-08-14 18:01:31.426  INFO 1 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.transaction.annotation.ProxyTransactionManagementConfiguration' of type [org.springframework.tra
nsaction.annotation.ProxyTransactionManagementConfiguration$$EnhancerBySpringCGLIB$$a2136fa1] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
app_1  | 2021-08-14 18:01:32.094  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
app_1  | 2021-08-14 18:01:32.138  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
app_1  | 2021-08-14 18:01:32.138  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache Tomcat/8.5.34
```

Fig.14. IntelliJ Terminal



```
PS C:\Users\Sumeyye\Desktop\telephoneDirectoryg> docker-compose up --build
Creating network "telephonedirectoryg_default" with the default driver
Building app
[+] Building 12.6s (11/11) FINISHED
 => [internal] load build definition from Dockerfile                                                              0.2s
 => => transferring dockerfile: 32B                                                                               0.0s
 => [internal] load .dockerignore                                                                                 0.2s
 => => transferring context: 2B                                                                                   0.0s
 => [internal] load metadata for docker.io/library/openjdk:8-jdk-alpine                                          12.2s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                                    0.0s
 => [internal] load build context                                                                                 0.1s
 => => transferring context: 7.96kB                                                                               0.0s
 => [1/5] FROM docker.io/library/openjdk:8-jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd524164a0f2283343547b3   0.0s
 => CACHED [2/5] COPY build/dependency/BOOT-INF/lib /spring-boot-docker/lib                                       0.0s
 => CACHED [3/5] COPY build/dependency/META-INF /spring-boot-docker/META-INF                                      0.0s
 => CACHED [4/5] COPY build/dependency/BOOT-INF/classes /spring-boot-docker                                       0.0s
 => CACHED [5/5] ADD build/libs/spring-boot-docker.jar spring-boot-docker.jar                                     0.0s
 => exporting to image                                                                                            0.0s
 => => exporting layers                                                                                           0.0s
 => => writing image sha256:43bda8027f406e7d9f8ef1fabfdd525a4957652e1c89fd5e55c94ea567077e5a                      0.0s
 => => naming to docker.io/library/telephonedirectoryg_app                                                        0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
Creating my_db ... done
Creating telephonedirectoryg_app_1 ... done
```

Fig.15. IntelliJ Terminal



```
app_1  | Hibernate: select enrolle0_.id as id1_0_0_, enrolle0_.description as descript2_0_0_, enrolle0_.name as name3_0_0_, enrolle0_.phone_number as phone_nu4_0_0_, enrolle0_.surname as surname5_0_0_ from publictd.enrolle enrol
le0_ where enrolle0_.id=?
app_1  | Hibernate: update publictd.enrolle set description=?, name=?, phone_number=?, surname=? where id=?
app_1  | Hibernate: select enrolle0_.id as id1_0_0_, enrolle0_.description as descript2_0_0_, enrolle0_.name as name3_0_0_, enrolle0_.phone_number as phone_nu4_0_0_, enrolle0_.surname as surname5_0_0_ from publictd.enrolle enrol
le0_ where enrolle0_.id=?
app_1  | Hibernate: select enrolle0_.id as id1_0_0_, enrolle0_.description as descript2_0_0_, enrolle0_.name as name3_0_0_, enrolle0_.phone_number as phone_nu4_0_0_, enrolle0_.surname as surname5_0_0_ from publictd.enrolle enrol
le0_ where enrolle0_.id=?
app_1  | Hibernate: delete from publictd.enrolle where id=?
```

Fig.16. IntelliJ Terminal

Berke ERDEM