**SPRING BOOT MINI PROJECT**

**//PgApplication Class**

**package** org.statement.com;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public class** PgApplication {

**public** static void main(String[] args) {

SpringApplication.run(PgApplication.class, args);

}

}

**//Owner Class**

**package** org.statement.com.dto;

**import** jakarta.persistence.\*;

**import** lombok.\*;

**import** java.util.List;

@Entity

@Data

@AllArgsConstructor

@NoArgsConstructor

**public class** Owner {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** Long id;

@Column(nullable = **false**)

**private** String name;

@Column(nullable = **false**, unique = **true**)

**private** String phone;

@Column(nullable = **false**, unique = **true**)

**private** String email;

@Column(nullable = **false**)

**private int** age;

@Column(nullable = **false**)

**private** String password;

@OneToMany(mappedBy = "owner", fetch = FetchType.***EAGER***)

**private** List<PGPlace> pgPlaces;

}

**//PGPlace Class**

**package** org.statement.com.dto;

**import** com.fasterxml.jackson.annotation.JsonIgnore;

**import** jakarta.persistence.\*;

**import** lombok.\*;

@Entity

@Data

@AllArgsConstructor

@NoArgsConstructor

**public class** PGPlace {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

**private** Long id;

@Column(nullable = false)

**private** String registrationNumber;

@Column(nullable = false)

**private** double builtUpArea;

@Column(nullable = false)

**private** double rentAmount;

@Column(nullable = false)

**private** String city;

@Column(nullable = false)

**private** String locality;

@Column(nullable = false)

**private** boolean availability;

@ManyToOne

@JoinColumn(name = "owner\_id")

@JsonIgnore

**private** Owner owner;

}

**//Tenant Class**

**package** org.statement.com.dto;

**import** jakarta.persistence.\*;

**import** lombok.\*;

@Entity

@Data

@AllArgsConstructor

@NoArgsConstructor

**public class** Tenant {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

**private** Long id;

@Column(nullable = false)

**private** String name;

@Column(nullable = false, unique = true)

**private** String phone;

@Column(nullable = false)

**private** int age;

@Column(nullable = false, unique = true)

**private** String email;

@Column(nullable = false)

**private** String password;

}

**//OwnerController Class**

**package** org.statement.com.controller;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.\*;

**import** org.statement.com.dto.Owner;

**import** org.statement.com.dto.PGPlace;

**import** org.statement.com.service.OwnerAndPgServiceImpl;

@RestController

@RequestMapping("/owner")

**public class** OwnerController {

@Autowired

**private** OwnerAndPgServiceImpl service;

@PostMapping("/add")

**public** ResponseEntity<Owner> saveOwner(@RequestBody Owner owner) {

**return** service.saveOwner(owner);

}

@PutMapping("/update")

**public** ResponseEntity<Owner> updateOwner(@RequestBody Owner owner) {

**return** service.updateOwner(owner);

}

@PostMapping("places/add/{owner\_id}")

**public** ResponseEntity<PGPlace> addPGPlace(@PathVariable Long owner\_id,@RequestBody PGPlace pgPlace) {

**return** service.addPlace(owner\_id,pgPlace);

}

@PutMapping("/places/edit")

**public** ResponseEntity<PGPlace> updatePGPlace(@RequestBody PGPlace pgPlace) {

**return** service.updatePlace(pgPlace);

}

@DeleteMapping("/places/delete/{id}")

**public** ResponseEntity<String> deletePGPlace(@PathVariable Long id) {

**return** service.deletePlace(id);

}

@GetMapping("/{id}")

**public** ResponseEntity<Owner> getOwnerById(@PathVariable Long id) {

**return** service.findById(id);

}

@PostMapping("/verifyOwner")

**public** ResponseEntity<Owner> verifyOwner(@RequestParam String phone, @RequestParam String password) {

**return** service.verifyOwner(phone, password);

}

@GetMapping("/places/{id}")

**public** ResponseEntity<List<PGPlace>> getOwnerPlacesById(@PathVariable Long id) {

**return** service.getOwnerPlacesById(id);

}

@PutMapping("/{id}/status/{status}")

**public** ResponseEntity<PGPlace> changeStatus(@PathVariable Long id, @PathVariable boolean status) {

**return** service.changeStatus(id, status);

}

}

**//PGPlaceController Class**

**package** org.statement.com.controller;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.\*;

**import** org.statement.com.dto.Owner;

**import** org.statement.com.dto.PGPlace;

**import** org.statement.com.service.OwnerAndPgServiceImpl;

@RestController

@RequestMapping("/pg")

**public class** PGPlaceController {

@Autowired

**private** OwnerAndPgServiceImpl service;

@GetMapping("/details/{id}")

**public** ResponseEntity<PGPlace> getPGPlaceDetails(@PathVariable Long id) {

**return** service.getPlaceById(id);

}

@GetMapping("/owner/{id}")

**public** ResponseEntity<Owner> getPGOwnerDetails(@PathVariable Long id) {

**return** service.getPGOwner(id);

}

//Important

@GetMapping("/city/{city}")

**public** ResponseEntity<List<PGPlace>> getPGPlacesByCity(@PathVariable String city) {

**return** service.getPlacesByCity(city);

}

@GetMapping("/locality/{locality}")

**public** ResponseEntity<List<PGPlace>> getPGPlacesByLocality(@PathVariable String locality) {

**return** service.getPlacesByLocality(locality);

}

}

**//TenantController Class**

**package** org.statement.com.controller;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.\*;

**import** org.statement.com.dto.Tenant;

**import** org.statement.com.service.OwnerAndPgServiceImpl;

@RestController

@RequestMapping("/tenant")

**public class** TenantController {

@Autowired

**private** OwnerAndPgServiceImpl service;

@PostMapping

**public** ResponseEntity<Tenant> saveTenant(@RequestBody Tenant tenant) {

**return** service.saveTenant(tenant);

}

@GetMapping("/{id}")

**public** ResponseEntity<Tenant> getTenantById(@PathVariable Long id) {

**return** service.findTenantById(id);

}

@DeleteMapping("/{id}")

**public** ResponseEntity<String> deleteTenantById(@PathVariable Long id) {

**return** service.deleteTenant(id);

}

@PostMapping("/verifyTenant")

**public** ResponseEntity<Tenant> verifyTenant(@RequestParam String email, @RequestParam String password) {

**return** service.verifyTenant(email, password);

}

}

**//PGPlaceService Interface**

**package** org.statement.com.service;

**import** java.util.List;

**import** org.springframework.http.ResponseEntity;

**import** org.statement.com.dto.Owner;

**import** org.statement.com.dto.PGPlace;

**public** interface PGPlaceService {

ResponseEntity<List<PGPlace>> getPlacesByCity(String city);

ResponseEntity<List<PGPlace>> getPlacesByLocality(String locality);

ResponseEntity<PGPlace> getPlaceById(Long id);

ResponseEntity<PGPlace> addPlace(Long id,PGPlace place);

ResponseEntity<PGPlace> updatePlace(PGPlace place);

ResponseEntity<String> deletePlace(Long id);

ResponseEntity<PGPlace> changeStatus(Long id, boolean status);

ResponseEntity<List<PGPlace>> getOwnerPlacesById(Long id);

ResponseEntity<Owner> getPGOwner(Long id);

}

**//OwnerAndPgServiceImpl Class**

**package** org.statement.com.service;

**import** java.util.List;

**import** java.util.Optional;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.\*;

**import** org.springframework.stereotype.Service;

**import** org.statement.com.dto.\*;

**import** org.statement.com.repository.\*;

@Service

**public class** OwnerAndPgServiceImpl implements PGPlaceService {

@Autowired

**private** PGPlaceRepository pgPlaceRepository;

@Autowired

**private** OwnerRepository ownerRepository;

@Autowired

**private** TenantRepository tenantRepository;

// ADD OWNER

**public** ResponseEntity<Owner> saveOwner(Owner owner) {

if(owner.getAge()>=18)

**return new**  ResponseEntity<>(ownerRepository.save(owner), HttpStatus.CREATED);

**return new**  ResponseEntity<>(HttpStatus.BAD\_REQUEST);

}

// UPDATE OWNER

**public** ResponseEntity<Owner> updateOwner(Owner owner) {

**if** (ownerRepository.existsById(owner.getId()) && owner.getAge()>=18) {

Owner existingOwner = ownerRepository.findById(owner.getId()).or**Else**(**null**);

**if** (existingOwner != **null**) {

List<PGPlace> existingPGPlaces = existingOwner.getPgPlaces();

Owner updatedOwner = ownerRepository.save(owner);

**for** (PGPlace pgPlace : existingPGPlaces) {

pgPlace.setOwner(updatedOwner);

pgPlaceRepository.save(pgPlace);

}

**return new**  ResponseEntity<>(updatedOwner, HttpStatus.OK);

}

}

**return new**  ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

// FIND BY ID OWNER

**public** ResponseEntity<Owner> findById(Long id) {

**return** ownerRepository.findById(id).map(owner -> **new**  ResponseEntity<>(owner, HttpStatus.OK))

.or**Else**Get(() -> **new**  ResponseEntity<>(HttpStatus.NOT\_FOUND));

}

// VERIFY BY PHONE AND PASSWORD

**public** ResponseEntity<Owner> verifyOwner(String phone, String password) {

Owner owner = ownerRepository.findByPhoneAndPassword(phone, password);

**if** (owner != **null**) {

**return new**  ResponseEntity<>(owner, HttpStatus.OK);

}

**return new**  ResponseEntity<>(HttpStatus.UNAUTHORIZED);

}

// GET PG PLACE BY CITY

@Override

**public** ResponseEntity<List<PGPlace>> getPlacesByCity(String city) {

List<PGPlace> pgPlaces = pgPlaceRepository.findByCity(city);

**if** (!pgPlaces.isEmpty()) {

**return new**  ResponseEntity<>(pgPlaces, HttpStatus.OK);

}

**return new**  ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

// GET PG PLACE BY CITY

@Override

**public** ResponseEntity<List<PGPlace>> getPlacesByLocality(String locality) {

List<PGPlace> pgPlaces = pgPlaceRepository.findByLocality(locality);

**if** (!pgPlaces.isEmpty()) {

**return new**  ResponseEntity<>(pgPlaces, HttpStatus.OK);

}

**return new**  ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

// GET PG PLACE BY ID

@Override

**public** ResponseEntity<PGPlace> getPlaceById(Long id) {

**return** pgPlaceRepository.findById(id).map(pgPlace -> **new**  ResponseEntity<>(pgPlace, HttpStatus.OK))

.or**Else**Get(() -> **new**  ResponseEntity<>(HttpStatus.NOT\_FOUND));

}

// ADD PG PLACE

@Override

**public** ResponseEntity<PGPlace> addPlace(Long id, PGPlace place) {

Optional<Owner> recOwner = ownerRepository.findById(id);

**if** (recOwner.isPresent()) {

Owner o = recOwner.get();

o.getPgPlaces().add(place);

place.setOwner(o);

ownerRepository.save(o);

**return new**  ResponseEntity<>(pgPlaceRepository.save(place), HttpStatus.CREATED);

}

**return new**  ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

// UPDATE PG PLACE

@Override

**public** ResponseEntity<PGPlace> updatePlace(PGPlace pgPlace) {

**if** (pgPlaceRepository.existsById(pgPlace.getId())) {

pgPlace.setId(pgPlace.getId());

**return new**  ResponseEntity<>(pgPlaceRepository.save(pgPlace), HttpStatus.OK);

}

**return new**  ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

// DELETE PG PLACE BY ID

@Override

**public** ResponseEntity<String> deletePlace(Long id) {

**if** (pgPlaceRepository.existsById(id)) {

pgPlaceRepository.deleteById(id);

**return new**  ResponseEntity<>("Deleted", HttpStatus.OK);

}

**return new**  ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

// UPDATE CHANGE STATUS

@Override

**public** ResponseEntity<PGPlace> changeStatus(Long id, boolean status) {

**return** pgPlaceRepository.findById(id).map(pgPlace -> {

pgPlace.setAvailability(status);

**return new**  ResponseEntity<>(pgPlaceRepository.save(pgPlace), HttpStatus.OK);

}).or**Else**Get(() -> **new**  ResponseEntity<>(HttpStatus.NOT\_FOUND));

}

@Override

**public** ResponseEntity<List<PGPlace>> getOwnerPlacesById(Long id) {

List<PGPlace> pgPlaces = ownerRepository.findPlacesById(id);

**if** (!pgPlaces.isEmpty()) {

**return new**  ResponseEntity<>(pgPlaces, HttpStatus.OK);

}

**return new**  ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@Override

**public** ResponseEntity<Owner> getPGOwner(Long id) {

Owner owner = pgPlaceRepository.findOwnerId(id);

**if** (owner != **null**) {

**return new**  ResponseEntity<>(owner, HttpStatus.OK);

} **else** {

**return new**  ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

**public** ResponseEntity<Tenant> saveTenant(Tenant tenant) {

if(tenant.getAge()>=18)

**return new**  ResponseEntity<>(tenantRepository.save(tenant), HttpStatus.CREATED);

**return new**  ResponseEntity<>(HttpStatus.BAD\_REQUEST);

}

**public** ResponseEntity<String> deleteTenant(Long id) {

**if** (tenantRepository.existsById(id)) {

tenantRepository.deleteById(id);

**return new**  ResponseEntity<>("Deleted", HttpStatus.OK);

}

**return new**  ResponseEntity<>(HttpStatus.NOT\_FOUND); }

**public** ResponseEntity<Tenant> findTenantById(Long id) {

**return** tenantRepository.findById(id).map(tenant -> **new**  ResponseEntity<>(tenant, HttpStatus.OK))

.or**Else**Get(() -> **new**  ResponseEntity<>(HttpStatus.NOT\_FOUND));

}

**public** ResponseEntity<Tenant> verifyTenant(String email, String password) {

Tenant tenant = tenantRepository.findByEmailAndPassword(email, password);

**if** (tenant != **null**) {

**return new**  ResponseEntity<>(tenant, HttpStatus.OK);

}

**return new**  ResponseEntity<>(HttpStatus.UNAUTHORIZED);

}

}

**//OwnerRepository Interface**

**package** org.statement.com.repository;

**import** java.util.List;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.data.jpa.repository.Query;

**import** org.statement.com.dto.Owner;

**import** org.statement.com.dto.PGPlace;

**public** interface OwnerRepository extends JpaRepository<Owner, Long> {

**public** Owner findByPhoneAndPassword(String phone, String password);

@Query("select o.pgPlaces from Owner o where o.id=?1")

**public** List<PGPlace> findPlacesById(Long id);

}

**//PGPlaceRepository** **Interface**

**package** org.statement.com.repository;

**import** java.util.List;

**import** java.util.Optional;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.data.jpa.repository.Query;

**import** org.springframework.data.repository.query.Param;

**import** org.springframework.http.ResponseEntity;

**import** org.statement.com.dto.Owner;

**import** org.statement.com.dto.PGPlace;

**public** interface PGPlaceRepository extends JpaRepository<PGPlace, Long> {

**public** List<PGPlace> findByCity(String city);

**public** List<PGPlace> findByLocality(String locality);

@Query("SELECT p.owner FROM PGPlace p WHERE p.id = :id")

**public** Owner findOwnerId(@Param("id") Long id);

}

**//TenantRepository Interface**

**package** org.statement.com.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.data.jpa.repository.Query;

**import** org.statement.com.dto.Owner;

**import** org.statement.com.dto.Tenant;

**public interface** TenantRepository **extends** JpaRepository<Tenant, Long> {

@Query("SELECT t FROM Tenant t WHERE t.email = :email AND t.password = :password")

**public** Tenant findByEmailAndPassword(String email, String password);

}

**//application.properties**

spring.application.name=PG

server.port=8085

spring.datasource.url= jdbc:mysql://localhost:3306/pg\_api\_app?createDatabaseIfNotExist=true

spring.datasource.username=root

spring.datasource.password=J2EE@java

spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.format\_sql=true

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL8Dialect

**//pom.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>3.3.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>org.software</groupId>

<artifactId>PG</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>PG</name>

<description>PGproject forSpring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>