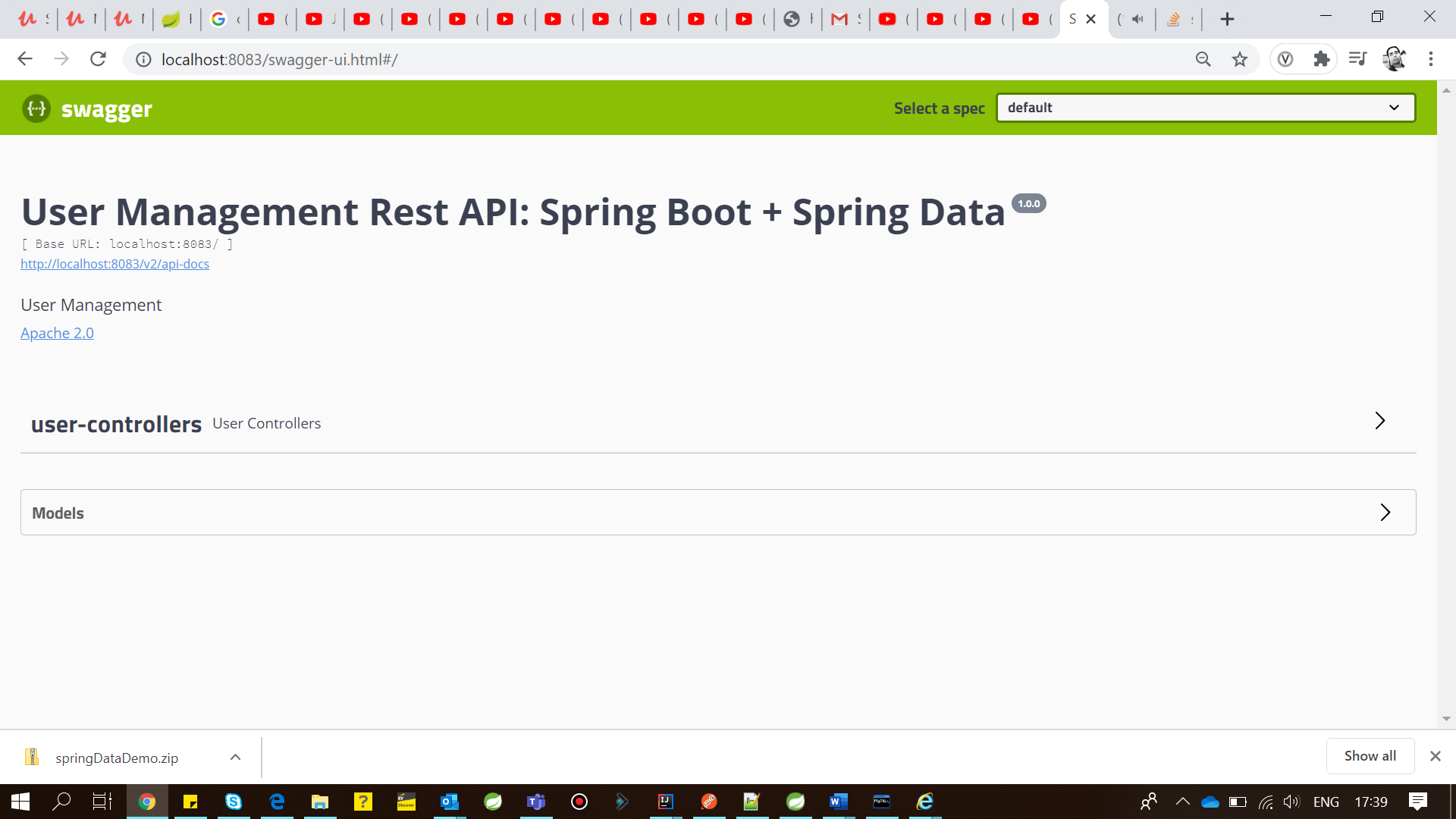
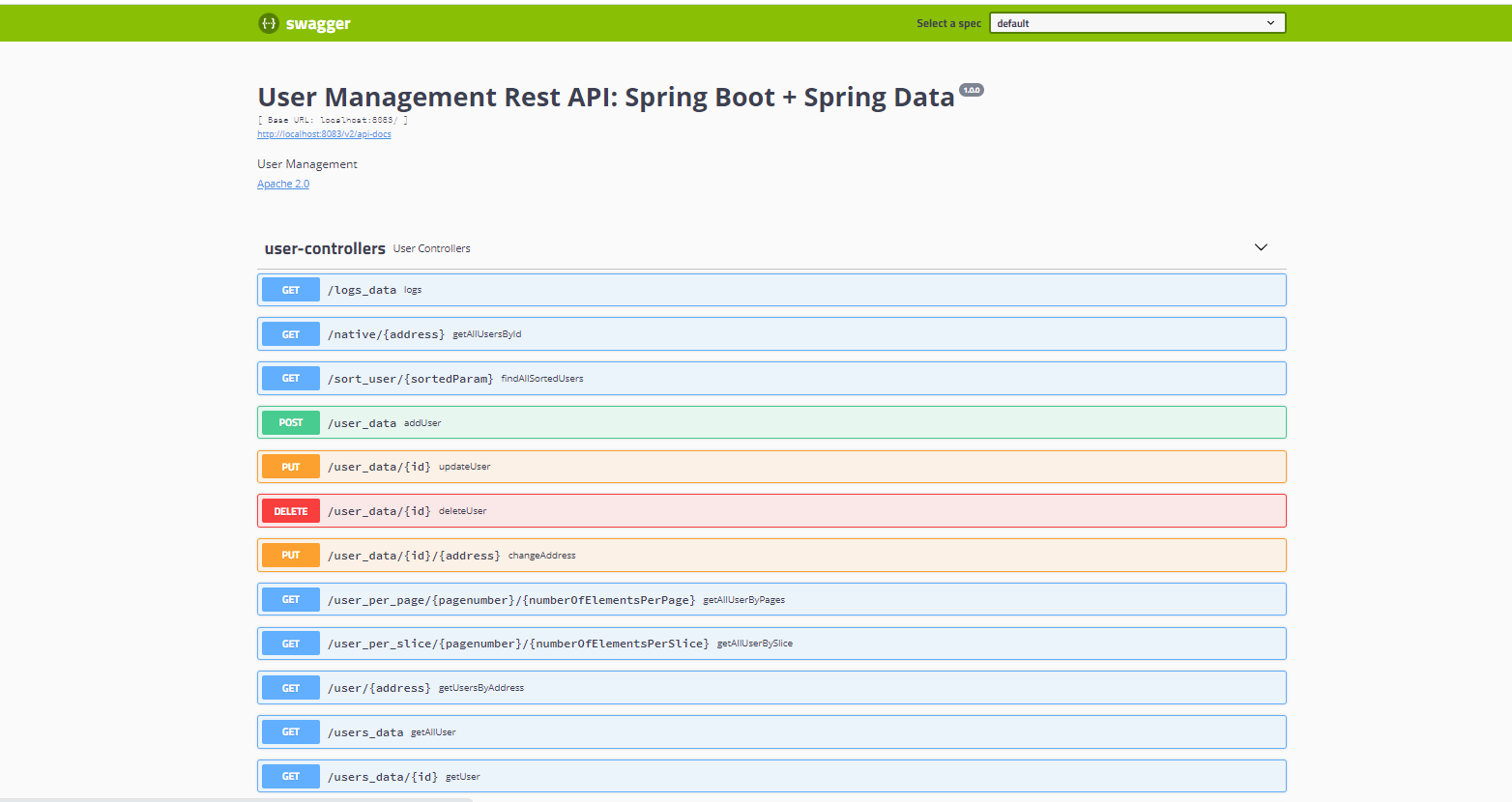
**Spring Data + JPA + Rest API + MySQL (C O D E)**

**springDataDemo**

package com.always.learner.springDataDemo.config;  
import io.swagger.annotations.Contact;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import springfox.documentation.builders.ApiInfoBuilder;  
import springfox.documentation.builders.RequestHandlerSelectors;  
import springfox.documentation.service.ApiInfo;  
import springfox.documentation.spi.DocumentationType;  
import springfox.documentation.spring.web.plugins.Docket;  
import springfox.documentation.swagger2.annotations.EnableSwagger2;  
  
  
@Configuration  
@EnableSwagger2  
public class SwagConfig  
{ @Bean  
 public Docket swaggerConfig()  
 {  
 return new Docket(DocumentationType.*SWAGGER\_2*).select()  
 .apis(RequestHandlerSelectors.*basePackage*("com.always.learner.springDataDemo"))  
 .build()  
 .apiInfo(apiInfo());  
 }  
 ApiInfo apiInfo() {  
 return new ApiInfoBuilder()  
 .title("User Management Rest API: Spring Boot + Spring Data")  
 .description("User Management")  
 .license("Apache 2.0")  
 .licenseUrl("http://www.apache.org/licenses/LICENSE-2.0.html")  
 .termsOfServiceUrl("")  
 .version("1.0.0")  
 // .contact(new Contact("", "", "contact@contact.com.uy"))  
 .build();  
 }  
  
}





package com.always.learner.springDataDemo.controller;  
import com.always.learner.springDataDemo.dto.Users;  
import com.always.learner.springDataDemo.service.UserServices;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.PageRequest;  
import org.springframework.data.domain.Slice;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;  
import java.net.URI;  
import java.util.List;  
  
@RestController  
public class UserControllers  
{  
 Logger logger= LoggerFactory.*getLogger*(UserControllers.class);  
 @Autowired  
 UserServices userService;  
  
 @GetMapping("/users\_data")  
 public List<Users> getAllUser()  
 {  
 logger.info("access all user");  
 return userService.getAllUser();  
 }  
  
 @GetMapping("/users\_data/{id}")  
 public Users getUser(@PathVariable Long id)  
 {  
 logger.info("access user based on id");  
 return userService.getUser(id);  
 }  
  
 @PostMapping("/user\_data")  
 public ResponseEntity<Object> addUser(@RequestBody Users user)  
 {  
 userService.addUser(user);  
 URI path=ServletUriComponentsBuilder.*fromCurrentRequest*()  
 .path("/{id}")  
 .buildAndExpand(user.getId())  
 .toUri();  
 return ResponseEntity.*created*(path).build();  
 }  
  
 @PutMapping("/user\_data/{id}")  
 public void updateUser(@PathVariable Long id,@RequestParam String address)  
 {  
 userService.updateUser(id,address);  
 }  
  
 @DeleteMapping("/user\_data/{id}")  
 public void deleteUser(@PathVariable Long id)  
 {  
 userService.deleteUser(id);  
 }  
  
 @GetMapping("/users/{name}")  
 public List<Users> getAllUsersByNames(@PathVariable String name)  
 {  
 return userService.getAllUsersByNames(name);  
 }  
  
 @GetMapping("/user/{address}")  
 public List<Users> getUsersByAddress(@PathVariable String address)  
 {  
 return userService.getUsersByAddress(address);  
 }  
  
 @PutMapping("/user\_data/{id}/{address}")  
 public void changeAddress(@PathVariable Long id,@PathVariable String address)  
 {  
 userService.changeAddress(id,address);  
 }  
  
 @GetMapping("/sort\_user/{sortedParam}")  
 public List<Users> findAllSortedUsers(@PathVariable String sortedParam)  
 {  
 return userService.findAllSortedUsers(sortedParam);  
 }  
  
 @GetMapping("/user\_per\_page/{pagenumber}/{numberOfElementsPerPage}")  
 public Page<Users> getAllUserByPages(@PathVariable String pagenumber, @PathVariable String numberOfElementsPerPage)  
 {  
 return userService.getAllUserByPages(Integer.*parseInt*(pagenumber),Integer.*parseInt*(numberOfElementsPerPage));  
  
 }  
  
 @GetMapping("/user\_per\_slice/{pagenumber}/{numberOfElementsPerSlice}")  
 public Slice<Users> getAllUserBySlice(@PathVariable String pagenumber, @PathVariable String numberOfElementsPerSlice) {  
 return userService.getAllUserBySlice(Integer.*parseInt*(pagenumber),Integer.*parseInt*(numberOfElementsPerSlice));  
 }  
  
 @GetMapping("/native/{address}")  
 public List<Users> getAllUsersById(@PathVariable String address)  
 {  
 return userService.getAllUsersByAddress(address);  
 }  
  
 @GetMapping("/logs\_data")  
 public String logs()  
 {  
 logger.trace("A TRACE MESSAGE");  
 logger.info("An INFO Message");  
 logger.debug("A DEBUG Message");  
 logger.error("An ERROR Message");  
 logger.warn("A WARN Message");  
 return "Hey buddy see the console for different level of logs";  
 }  
}

package com.always.learner.springDataDemo.dto;  
  
import javax.persistence.\*;  
  
@Entity  
@Table(name = "users")  
public class Users  
{  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 private String name;  
 private String address;  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
}

package com.always.learner.springDataDemo.repository;  
  
import com.always.learner.springDataDemo.dto.Users;  
import org.springframework.data.repository.PagingAndSortingRepository;  
  
public interface UserRepository extends PagingAndSortingRepository<Users,Long>  
{  
  
}

package com.always.learner.springDataDemo.repository;  
import com.always.learner.springDataDemo.dto.Users;  
import org.springframework.data.jpa.repository.Lock;  
import org.springframework.data.jpa.repository.Modifying;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.CrudRepository;  
import org.springframework.data.repository.query.Param;  
import org.springframework.stereotype.Repository;  
  
import javax.persistence.LockModeType;  
import javax.transaction.Transactional;  
import java.util.List;  
  
@Repository  
public interface UserRepositorys extends CrudRepository<Users, Long> {  
  
 @Transactional  
 @Modifying  
 @Query("update Users set address=:address where id=:id")  
 public void updateAddress(@Param("id") Long id, @Param("address") String address);  
  
  
 //Indexed Query  
 @Query("select u from Users u where u.name=?1")  
 List<Users> getAllUsersByNames(String name);  
  
 //Native Query  
 @Query(value = "select u from users u where u.address=:address",nativeQuery=true)  
 List<Users> getAllUsersByAddress(@Param("address") String address);  
  
 //Named Query  
 @Query("select u from Users u where u.address=:address")  
 List<Users> getUsersByAddress(@Param("address") String address);  
  
 @Transactional  
 @Modifying  
 @Query("update Users set address=?2 where id=?1")  
 @Lock(LockModeType.*NONE*)  
 public void changeAddress(Long id, String address);  
}

package com.always.learner.springDataDemo.service;  
import com.always.learner.springDataDemo.dto.Users;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.Slice;  
  
import java.util.List;  
  
public interface UserServices  
{  
 public List<Users> getAllUser();  
  
 public Users getUser(Long id);  
  
 public void addUser(Users user);  
  
 public String updateUser(Long id, String address);  
  
 public void deleteUser(Long id);  
  
 public List<Users> getAllUsersByNames(String name);  
  
 public List<Users> getUsersByAddress(String address);  
  
 public String changeAddress(Long id, String address);  
  
 public List<Users> findAllSortedUsers(String sortedParam);  
  
 public Page<Users> getAllUserByPages(int pagenumber, int numberOfElementsPerPage);  
  
 public Slice<Users> getAllUserBySlice(int pagenumber, int numberOfElementsPerPage);  
  
 public List<Users> getAllUsersByAddress(String address);  
  
  
  
  
}

package com.always.learner.springDataDemo.service;  
import com.always.learner.springDataDemo.dto.Users;  
import com.always.learner.springDataDemo.repository.UserRepository;  
import com.always.learner.springDataDemo.repository.UserRepositorys;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.PageRequest;  
import org.springframework.data.domain.Slice;  
import org.springframework.data.domain.Sort;  
import org.springframework.data.repository.query.Param;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class UserServiceImpls implements UserServices {  
  
 Logger logger= LoggerFactory.*getLogger*(UserServiceImpls.class);  
 @Autowired  
 UserRepositorys userRepository;  
  
 @Autowired  
 UserRepository userRepo;  
  
 @Override  
 public List<Users> getAllUser() {  
 List<Users> userList= (List<Users>) userRepository.findAll();  
 logger.info("UserList from the DataBase:- {}",userList);  
 return userList;  
 }  
  
 @Override  
 public Users getUser(Long id) {  
 Optional<Users> userOptional=userRepository.findById(id);  
 Users user=userOptional.get();  
 logger.info("User - {} find by Id - {}",user,id);  
 return user;  
 }  
  
 @Override  
 public void addUser(Users user) {  
 userRepository.save(user);  
 logger.info("User added- {}",user);  
 }  
  
 @Override  
 public String updateUser(Long id,String address) {  
 userRepository.updateAddress(id,address);  
 logger.info("New Address for user-id {} is - {}",id,address);  
 return "New Address for user-id"+id+" is - "+address;  
 }  
  
 @Override  
 public void deleteUser(Long id) {  
 userRepository.deleteById(id);  
 logger.info("Delete {} of User",id);  
 }  
  
 @Override  
 public List<Users> getAllUsersByNames(String name)  
 {  
 List<Users> usersList=userRepository.getAllUsersByNames(name);  
 logger.info("get All Users By Names - {}",usersList);  
 return usersList;  
 }  
  
 @Override  
 public List<Users> getUsersByAddress(String address)  
 {  
 return userRepository.getUsersByAddress(address);  
 }  
  
 @Override  
 public String changeAddress(Long id,String address) {  
 userRepository.changeAddress(id,address);  
 logger.info("New Address for user-id {} is - {}",id,address);  
 return "New Address for user-id"+id+" is - "+address;  
 }  
  
 @Override  
 public List<Users> findAllSortedUsers(String sortedParam)  
 {  
 return (List<Users>) userRepo.findAll(Sort.*by*(sortedParam));  
 }  
  
 @Override  
 public Page<Users> getAllUserByPages(int pagenumber, int numberOfElementsPerPage) {  
 return userRepo.findAll(PageRequest.*of*(pagenumber,numberOfElementsPerPage));  
 }  
  
 @Override  
 public Slice<Users> getAllUserBySlice(int pagenumber, int numberOfElementsPerPage) {  
 return userRepo.findAll(PageRequest.*of*(pagenumber,numberOfElementsPerPage));  
 }  
  
 @Override  
 public List<Users> getAllUsersByAddress(String address) {  
 return userRepository.getAllUsersByAddress(address);  
 }  
  
}

package com.always.learner.springDataDemo;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class SpringDataDemoApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringDataDemoApplication.class, args);  
 }  
  
}

server:  
 port: 8083  
  
spring:  
 application:  
 name: spring-data-demo  
 datasource:  
 driver-class-name: com.mysql.cj.jdbc.Driver  
 password: Guidewire@2019  
 url: jdbc:mysql://localhost:3306/alwayslearning  
 username: root

plugins {  
 id 'org.springframework.boot' version '2.3.1.RELEASE'  
 id 'io.spring.dependency-management' version '1.0.9.RELEASE'  
 id 'java'  
}  
  
group = 'com.always.learner'  
version = '0.0.1-SNAPSHOT'  
sourceCompatibility = '1.8'  
  
repositories {  
 mavenCentral()  
}  
  
dependencies {  
 testImplementation('org.springframework.boot:spring-boot-starter-test')  
 implementation 'org.springframework.boot:spring-boot-starter-web'  
  
 //MySQL  
 compile group: 'mysql', name: 'mysql-connector-java', version: '8.0.17'  
  
 //SpringData  
 implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
  
 //Add Swagger dependencies  
 compile('io.springfox:springfox-swagger2:2.9.2')  
 compile('io.springfox:springfox-swagger-ui:2.9.2')  
 compile group: 'io.swagger.core.v3', name: 'swagger-maven-plugin', version: '2.1.3'  
  
}  
  
test {  
 useJUnitPlatform()  
}