MOBILE APP WEB SERVICES SITE WITH SPRING BOOT FRAMEWORK

Report Paper : May 1

Sumeyra Betul Polat - COM-18

Ala-Too International University / Computer Science Department

ACKNOWLEDGEMENT

First of all I would like to thank our lecturer 'Nurlan Shaidullaev' for supporting us learning 'Spring Boot Framework' and for supporting us when we need help. Beside from my lecturer, I like to thank my other students for their effort. We, the students, believe that we will be successful citizens and educators if we receive the necessary education to guarantee our future.In order to accomplish some things, it is necessary to work hard and make efforts. We need an environment where we can work hard to survive the challenges and go on new and bright days, and we need supporters who trust us.But success starts with self-confidence, so first of all thanks to our family and teachers.

INTRODUCTION

The main objective behind developing this Online WebSite project in Java is to provide a system for the new users, inforations where you can find make changes, and available of viewing them. This projected is working with Rest Api where its a different platform of viewing changes. In the proposed Online Products portal project in Java, while going through the application, the user have standars of loging into the system and he/she can view and delete, change the info, which they are looking for.

EXPLANATIONS

- Intellij the LINK:
https://www.jetbrains.com/idea/

- MySQL database LINK:
https://dev.mysql.com/downloads/installer/

- Java Version
https://www.oracle.com/java/technologies/javase-jdk11-downloads.html

- Account to log in to Tomcat (Okta)
https://mvnrepository.com/artifact/com.okta.spring/okta-spring-boot-starter

- Apache Tomcat
https://tomcat.apache.org/download-90.cgi

WHAT IS SPRING BOOT ?

Spring Boot is an open source framework used to create a micro Service.
provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can **just run**.
provides a flexible way to configure Java Beans, XML configurations, and Database Transactions.

What about the advantages ?

- Easy to understand and develop spring applications
- Increases productivity
- Reduces the development time

Execution Procedures
-Sample1-

We have an AddExceptionsHandler class in our exception package. In the code returns the: ResponseEntity<> within (errorMessage, new HttpHeaders(), HttpStatus.INTERNAL_SERVER_ERROR);

```
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

import java.util.Date;

@ControllerAdvice
public class AppExceptionsHandler {

    @ExceptionHandler(value = {UserServiceException.class})
    public ResponseEntity<Object>
handleUserServiceServiceException(UserServiceException ex) {
        ErrorMessage errorMessage = new ErrorMessage(new Date(), ex.getMessage());
        return new ResponseEntity<>(errorMessage, new HttpHeaders(),
HttpStatus.INTERNAL_SERVER_ERROR);
    }

    @ExceptionHandler(value = {Exception.class})
    public ResponseEntity<Object> handleUserServiceServiceException(Exception ex)
{
        ErrorMessage errorMessage = new ErrorMessage(new Date(), ex.getMessage());
        return new ResponseEntity<>(errorMessage, new HttpHeaders(),
```

```
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

-Sample2-

In the second part, there is a class to give the error message, complete with getter and setter method.

```
import java.util.Date;

public class ErrorMessage {

    private Date timestamp;
    private String message;

    public ErrorMessage() {
    }

    public ErrorMessage(Date timestamp, String message) {
        this.timestamp = timestamp;
        this.message = message;
    }

    public Date getTimestamp() {
        return timestamp;
    }

    public void setTimestamp(Date timestamp) {
        this.timestamp = timestamp;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

-Sample3-

And here is the UserServiceException class to get message with the exception

```
public class UserServiceException extends RuntimeException {

    public UserServiceException(String message) {
        super(message);
    }
}
```

-Sample4-

This class is called the user entity in order to use and call in our project

```java
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import java.io.Serializable;

@Entity(name = "users")
public class UserEntity implements Serializable {

    private static final long serialVersionUID = 3599305611574853751L;

    @Id
    @GeneratedValue
    private long id;

    @Column(nullable = false)
    private String userId;

    @Column(nullable = false, length = 50)
    private String firstName;

    @Column(nullable = false, length = 50)
    private String lastName;

    @Column(nullable = false, length = 100, unique = true)
    private String email;

    @Column(nullable = false)
    private String encryptedPassword;

    private String emailVerificationToken;

    @Column(nullable = false)
    private Boolean emailVerificationStatus = false;

    public long getId() {
        return id;
    }

    public String getUserId() {
        return userId;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getEmail() {
        return email;
    }
```

```
    public String getEncryptedPassword() {
        return encryptedPassword;
    }

    public String getEmailVerificationToken() {
        return emailVerificationToken;
    }

    public Boolean getEmailVerificationStatus() {
        return emailVerificationStatus;
    }

    public void setId(long id) {
        this.id = id;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public void setEncryptedPassword(String encryptedPassword) {
        this.encryptedPassword = encryptedPassword;
    }

    public void setEmailVerificationToken(String emailVerificationToken) {
        this.emailVerificationToken = emailVerificationToken;
    }

    public void setEmailVerificationStatus(Boolean emailVerificationStatus) {
        this.emailVerificationStatus = emailVerificationStatus;
    }
}
```

-Sample5-

Here is our second entity class called AdressEntity this classes values are going to updated, called and used including users.

```java
import javax.persistence.*;
import java.io.Serializable;

@Entity(name = "addresses")
public class AddressEntity implements Serializable {

    private static final long serialVersionUID = -2945997946072035683L;

    @Id
    @GeneratedValue
    private long id;

    @Column(length = 50, nullable = false)
    private String addressId;

    @Column(length = 20, nullable = false)
    private String city;

    @Column(length = 20, nullable = false)
    private String country;

    @Column(length = 100, nullable = false)
    private String streetName;

    @Column(length = 7, nullable = false)
    private String postalCode;

    @Column(length = 15, nullable = false)
    private String type;

    @ManyToOne
    @JoinColumn(name="user_id")
    private UserEntity userDetails;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getAddressId() {
        return addressId;
    }

    public void setAddressId(String addressId) {
        this.addressId = addressId;
    }

    public String getCity() {
        return city;
    }
```

```
    public void setCity(String city) {
        this.city = city;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public String getStreetName() {
        return streetName;
    }

    public void setStreetName(String streetName) {
        this.streetName = streetName;
    }

    public String getPostalCode() {
        return postalCode;
    }

    public void setPostalCode(String postalCode) {
        this.postalCode = postalCode;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public UserEntity getUserDetails() {
        return userDetails;
    }

    public void setUserDetails(UserEntity userDetails) {
        this.userDetails = userDetails;
    }
}
```

-Sample6-

This class is going to read values from property file

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.env.Environment;
import org.springframework.stereotype.Component;

@Component
public class AppProperties {

    @Autowired
    private Environment env;

    public String getTokenSecret() {
        return env.getProperty("tokenSecret");
    }
}
```

-Sample7-

Here is our AuthenticationFilter class, its going to be called only when the code is succesfull

And the message is called only when successfull authentication is conducted, otherwise its going to be called.

Furthermore the Jwts dependecy in pom.xml generates web token, and will send back server response inside. Going to be used in the future requests which requires authentication permissions.

```
import com.dmytroverner.mobileappws.SpringApplicationContext;
import com.dmytroverner.mobileappws.dto.UserDto;
import com.dmytroverner.mobileappws.model.request.UserLoginRequestModel;
import com.dmytroverner.mobileappws.model.response.ErrorMessages;
import com.dmytroverner.mobileappws.service.UserService;
import com.fasterxml.jackson.databind.ObjectMapper;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.userdetails.User;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilt
er;

import javax.servlet.FilterChain;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Date;

public class AuthenticationFilter extends UsernamePasswordAuthenticationFilter {

    private final AuthenticationManager authenticationManager;

    public AuthenticationFilter(AuthenticationManager authenticationManager) {
        this.authenticationManager = authenticationManager;
    }

    @Override
    public Authentication attemptAuthentication(HttpServletRequest request,
                                                HttpServletResponse response)
throws AuthenticationException {
        try {
            UserLoginRequestModel credentials = new ObjectMapper()
```

```
                    .readValue(request.getInputStream(),
UserLoginRequestModel.class);

            return authenticationManager.authenticate(
                    new UsernamePasswordAuthenticationToken(
                            credentials.getEmail(),
                            credentials.getPassword(),
                            new ArrayList<>()
                    )
            );
        } catch (IOException e) {
            throw new
RuntimeException(ErrorMessages.AUTHENTICATION_FAILED.getErrorMessage());
        }
    }

    @Override
    protected void successfulAuthentication(HttpServletRequest request,
                                            HttpServletResponse response,
                                            FilterChain chain,
                                            Authentication authResult) {
        String userName = ((User) authResult.getPrincipal()).getUsername();

        String token = Jwts.builder()
                .setSubject(userName)
                .setExpiration(new Date(System.currentTimeMillis() +
SecurityConstants.EXPIRATION_TIME))
                .signWith(SignatureAlgorithm.HS512,
SecurityConstants.getTokenSecret())
                .compact();
        UserService userService = (UserService)
SpringApplicationContext.getBean("userServiceImpl");
        UserDto userDto = userService.getUser(userName);

        response.addHeader(SecurityConstants.HEADER_STRING,
SecurityConstants.TOKEN_PREFIX + token);
        response.addHeader("UserID", userDto.getUserId());
    }
}
```

-Sample8-

Gets the value from Header field named ‘Authorization’ inside the request And if the header contains ‘Messager’ Prefix inside its not null

```
import io.jsonwebtoken.Jwts;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import
org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.ArrayList;

public class AuthorizationFilter extends BasicAuthenticationFilter {

    public AuthorizationFilter(AuthenticationManager authenticationManager) {
        super(authenticationManager);
    }

    @Override
    protected void doFilterInternal(HttpServletRequest req,
                                    HttpServletResponse res,
                                    FilterChain chain) throws IOException,
ServletException {
        String header = req.getHeader(SecurityConstants.HEADER_STRING);

        if (header == null || !header.startsWith(SecurityConstants.TOKEN_PREFIX))
{
            chain.doFilter(req, res);
            return;
        }

        UsernamePasswordAuthenticationToken authentication =
getAuthentication(req);
        SecurityContextHolder.getContext().setAuthentication(authentication);
        chain.doFilter(req, res);
    }

    private UsernamePasswordAuthenticationToken
getAuthentication(HttpServletRequest request) {
        String token = request.getHeader(SecurityConstants.HEADER_STRING);

        if (token != null) {
            token = token.replace(SecurityConstants.TOKEN_PREFIX, "");

            String user = Jwts.parser()
                    .setSigningKey(SecurityConstants.getTokenSecret())
```

```
                    .parseClaimsJws(token)
                    .getBody()
                    .getSubject();
            if (user != null) {
                return new UsernamePasswordAuthenticationToken(user, null, new
ArrayList<>());
            }
        }
        return null;
    }
}
```

-Sample9-

Becouse this class is not a bean in order to get accessed to the environment the Token Secret is going the be readed from application.properties in order to get accessed as i called at the beginning

```
import com.dmytroverner.mobileappws.SpringApplicationContext;

public class SecurityConstants {
    public static final long EXPIRATION_TIME = 864000000; // 10 days
    public static final String TOKEN_PREFIX = "Bearer ";
    public static final String HEADER_STRING = "Authorization";
    public static final String SIGN_UP_URL = "/users";

    public static String getTokenSecret() {
        AppProperties appProperties = (AppProperties)
SpringApplicationContext.getBean("appProperties");
        return appProperties.getTokenSecret();
    }
}
```

-Sample10-

To costumize the standart spring login security login routine

And some codes that avoids caching JSON header inside sessions.

```
import com.dmytroverner.mobileappws.service.UserService;
import org.springframework.http.HttpMethod;
import
org.springframework.security.config.annotation.authentication.builders.Authenticat
ionManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity
;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfig
urerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@EnableWebSecurity
public class WebSecurity extends WebSecurityConfigurerAdapter {

    private final UserService userService;
    private final BCryptPasswordEncoder bCryptPasswordEncoder;

    public WebSecurity(UserService userService, BCryptPasswordEncoder
bCryptPasswordEncoder) {
        this.userService = userService;
        this.bCryptPasswordEncoder = bCryptPasswordEncoder;
    }

    @Override
    protected void configure(HttpSecurity httpSecurity) throws Exception {
        httpSecurity.csrf().disable().authorizeRequests()
                .antMatchers(HttpMethod.POST,
SecurityConstants.SIGN_UP_URL).permitAll()
                .anyRequest().authenticated().and()
                .addFilter(getAuthenticationFilter())
                .addFilter(new AuthorizationFilter(authenticationManager()))
                .sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS);
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userService).passwordEncoder(bCryptPasswordEncoder);
    }

    public AuthenticationFilter getAuthenticationFilter() throws Exception {
        final AuthenticationFilter filter = new
AuthenticationFilter(authenticationManager());
        filter.setFilterProcessesUrl("/users/login");
        return filter;
    }
}
```

-Sample11-

Lists the adress informations by userid, if the user is null returns the result

In the address is not null brings the address information

```
import com.dmytroverner.mobileappws.dto.AddressDto;
import com.dmytroverner.mobileappws.entity.AddressEntity;
import com.dmytroverner.mobileappws.entity.UserEntity;
import com.dmytroverner.mobileappws.repository.AddressRepository;
import com.dmytroverner.mobileappws.repository.UserRepository;
import com.dmytroverner.mobileappws.service.AddressService;
import org.modelmapper.ModelMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;

@Service
public class AddressServiceImpl implements AddressService {

    private final UserRepository userRepository;

    private final AddressRepository addressRepository;

    private ModelMapper modelMapper;

    @Autowired
    public AddressServiceImpl(UserRepository userRepository, AddressRepository
addressRepository) {
        modelMapper = new ModelMapper();
        this.userRepository = userRepository;
        this.addressRepository = addressRepository;
    }

    @Override
    public List<AddressDto> getAddresses(String userId) {
        List<AddressDto> result = new ArrayList<>();

        UserEntity userEntity = userRepository.findByUserId(userId);
        if (userEntity == null)
            return result;

        List<AddressEntity> addresses =
addressRepository.findAllByUserDetails(userEntity);
        for (AddressEntity addressEntity : addresses) {
            result.add(modelMapper.map(addressEntity, AddressDto.class));
        }

        return result;
    }

    @Override
    public AddressDto getAddress(String addressId) {
        AddressEntity addressEntity =
addressRepository.findByAddressId(addressId);
        if (addressEntity != null) {
            return modelMapper.map(addressEntity, AddressDto.class);
        }
        return null;
```

```
    }
}
```

In this class, we can display the user information with the url address in the application and have the opportunity to make changes.

```
@RestController
@RequestMapping("users")
public class UserController {

    private final UserService userService;
    private final AddressService addressService;

    private ModelMapper modelMapper;

    @Autowired
    public UserController(UserService userService, AddressService addressService)
{
        this.userService = userService;
        this.addressService = addressService;

        modelMapper = new ModelMapper();
    }

    @GetMapping(path="/{userId}",
            consumes = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE },
            produces = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE })
    public UserDetailsResponse getUser(@PathVariable("userId") String userId) {
        UserDto userDto = userService.getUserByUserId(userId);

        return modelMapper.map(userDto, UserDetailsResponse.class);
    }

    @PostMapping(
            consumes = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE },
            produces = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE })
    public UserDetailsResponse createUser(@RequestBody UserDetailsRequestModel
userDetailsRequestModel) {
        UserDto userDto = modelMapper.map(userDetailsRequestModel, UserDto.class);

        UserDto createdUser = userService.createUser(userDto);

        return modelMapper.map(createdUser, UserDetailsResponse.class);
    }

    @PutMapping(path="/{userId}",
            consumes = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE },
            produces = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE })
    public UserDetailsResponse putUser(@PathVariable("userId") String userId,
@RequestBody UserDetailsRequestModel userDetailsRequestModel) {
        UserDto userDto = modelMapper.map(userDetailsRequestModel, UserDto.class);
```

```
        UserDto updatedUser = userService.updateUser(userId, userDto);

        return modelMapper.map(updatedUser, UserDetailsResponse.class);
    }

    @DeleteMapping(path = "/{userId}",
            produces = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE })
    public OperationStatusModel deleteUser(@PathVariable("userId") String userId)
{
        OperationStatusModel operationStatusModel = new OperationStatusModel();
        operationStatusModel.setOperationName(RequestOperationName.DELETE.name());

        userService.delete(userId);


operationStatusModel.setOperationResult(RequestOperationStatus.SUCCESS.name());
        return operationStatusModel;
    }

    @GetMapping(produces = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE })
    public List<UserDetailsResponse> getUsers(@RequestParam(value = "page",
defaultValue = "0") int page,
                                              @RequestParam(value = "limit",
defaultValue = "25") int limit) {
        List<UserDetailsResponse> returnList = new ArrayList<>();

        List<UserDto> users = userService.getUsers(page, limit);
        for (UserDto userDto : users) {
            UserDetailsResponse userDetailsResponse = modelMapper.map(userDto,
UserDetailsResponse.class);
            returnList.add(userDetailsResponse);
        }

        return returnList;
    }

    @GetMapping(path="/{userId}/addresses",
            consumes = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE },
            produces = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE,
                    "application/hal+json"})
    public Resources<AddressResponse> getUserAddresses(@PathVariable("userId")
String userId) {
        List<AddressResponse> addressResponses = new ArrayList<>();

        List<AddressDto> addressesDto = addressService.getAddresses(userId);

        if (addressesDto != null && !addressesDto.isEmpty()) {
            Type listType = new TypeToken<List<AddressResponse>>() {}.getType();
            addressResponses = modelMapper.map(addressesDto, listType);

            addressResponses.forEach(addressResponse -> {
                Link addressLink =
linkTo(methodOn(UserController.class).getUserAddress(userId,
addressResponse.getAddressId())).withSelfRel();
                addressResponse.add(addressLink);
```

```
                Link userLink =
linkTo(methodOn(UserController.class).getUser(userId)).withRel("user");
                addressResponse.add(userLink);
            });
        }
        return new Resources<>(addressResponses);
    }

    @GetMapping(path="/{userId}/addresses/{addressId}",
            consumes = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE },
            produces = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE,
                        "application/hal+json"})
    public Resource<AddressResponse> getUserAddress(@PathVariable("userId") String
userId,
                                                    @PathVariable("addressId")
String addressId) {
        AddressDto address = addressService.getAddress(addressId);

        Link addressLink =
linkTo(methodOn(UserController.class).getUserAddress(userId,
addressId)).withSelfRel();
        Link userLink =
linkTo(UserController.class).slash(userId).withRel("user");
        Link addressesLink =
linkTo(methodOn(UserController.class).getUserAddresses(userId)).withRel("addresses
");

        AddressResponse addressResponse = new AddressResponse();
        if (address != null) {
            addressResponse = modelMapper.map(address, AddressResponse.class);
        }
        addressResponse.add(addressLink);
        addressResponse.add(userLink);
        addressResponse.add(addressesLink);

        return new Resource<>(addressResponse);
    }
}
```

-Sample13-

Here we have the variables with getter and setter methods includes AddressDto and UserDto

```
public class AddressDto {

    private long id;
    private String addressId;
    private String city;
    private String country;
    private String streetName;
    private String postalCode;
    private String type;
    private UserDto userDetails;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getAddressId() {
        return addressId;
    }

    public void setAddressId(String addressId) {
        this.addressId = addressId;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public String getStreetName() {
        return streetName;
    }

    public void setStreetName(String streetName) {
        this.streetName = streetName;
    }
```

```
    public String getPostalCode() {
        return postalCode;
    }

    public void setPostalCode(String postalCode) {
        this.postalCode = postalCode;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public UserDto getUserDetails() {
        return userDetails;
    }

    public void setUserDetails(UserDto userDetails) {
        this.userDetails = userDetails;
    }
}
```

-Sample14-

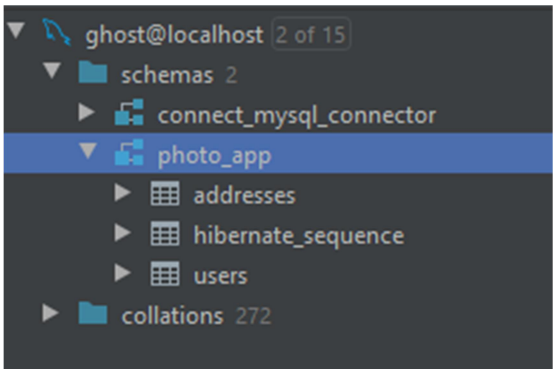
UserDto, with getEmailVerification

```
public class UserDto implements Serializable {
    private static final long serialVersionUID = -3245997946072035681L;
    private long id;
    private String userId;
    private String firstName;
    private String lastName;
    private String email;
    private String password;
    private String encryptedPassword;
    private String emailVerificationToken;
    private Boolean getEmailVerificationStatus = false;
    private List<AddressDto> addresses;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getUserId() {
        return userId;
    }
```

```
    public void setUserId(String userId) {
        this.userId = userId;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getEncryptedPassword() {
        return encryptedPassword;
    }

    public void setEncryptedPassword(String encryptedPassword) {
        this.encryptedPassword = encryptedPassword;
    }

    public String getEmailVerificationToken() {
        return emailVerificationToken;
    }

    public void setEmailVerificationToken(String emailVerificationToken) {
        this.emailVerificationToken = emailVerificationToken;
    }

    public Boolean getGetEmailVerificationStatus() {
        return getEmailVerificationStatus;
    }

    public void setGetEmailVerificationStatus(Boolean getEmailVerificationStatus)
{
        this.getEmailVerificationStatus = getEmailVerificationStatus;
```

```
    }

    public List<AddressDto> getAddresses() {
        return addresses;
    }

    public void setAddresses(List<AddressDto> addresses) {
        this.addresses = addresses;
    }
}
```

MySQL Connectin Informations

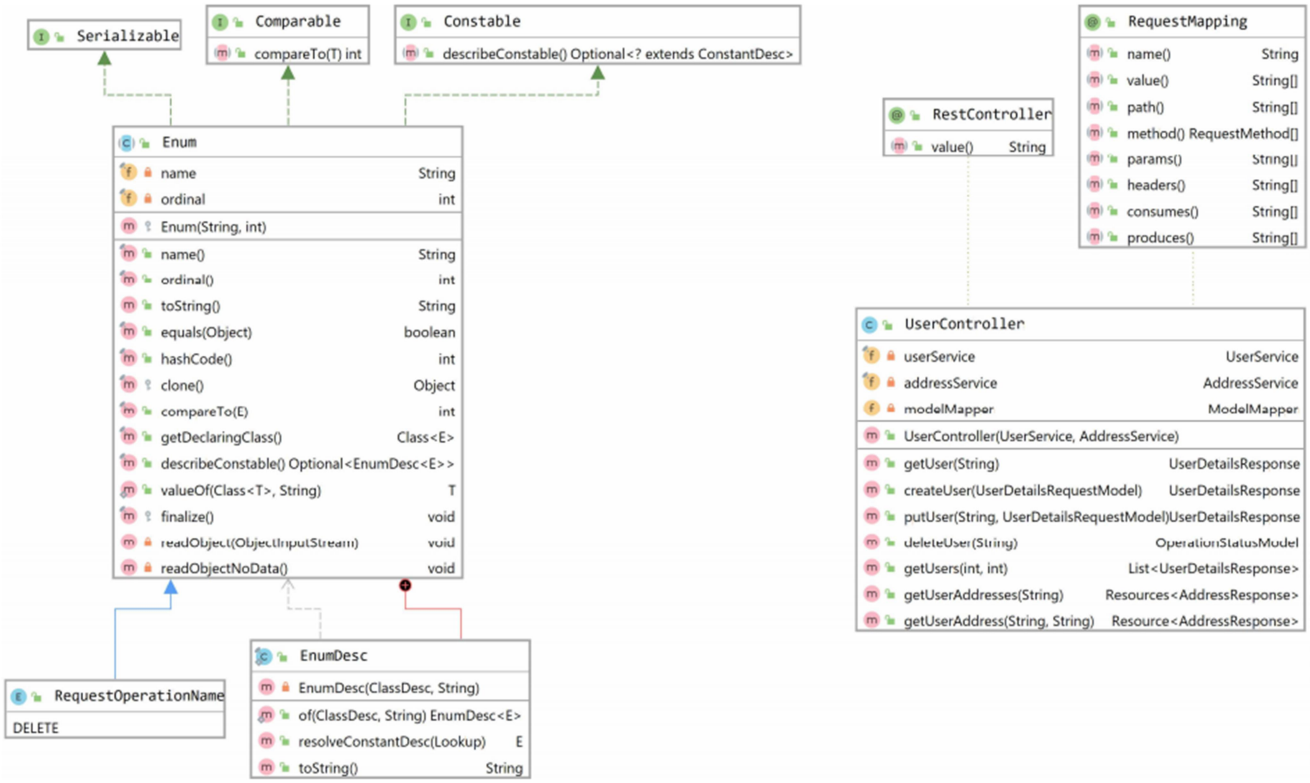


ADDITIONAL FEATURES

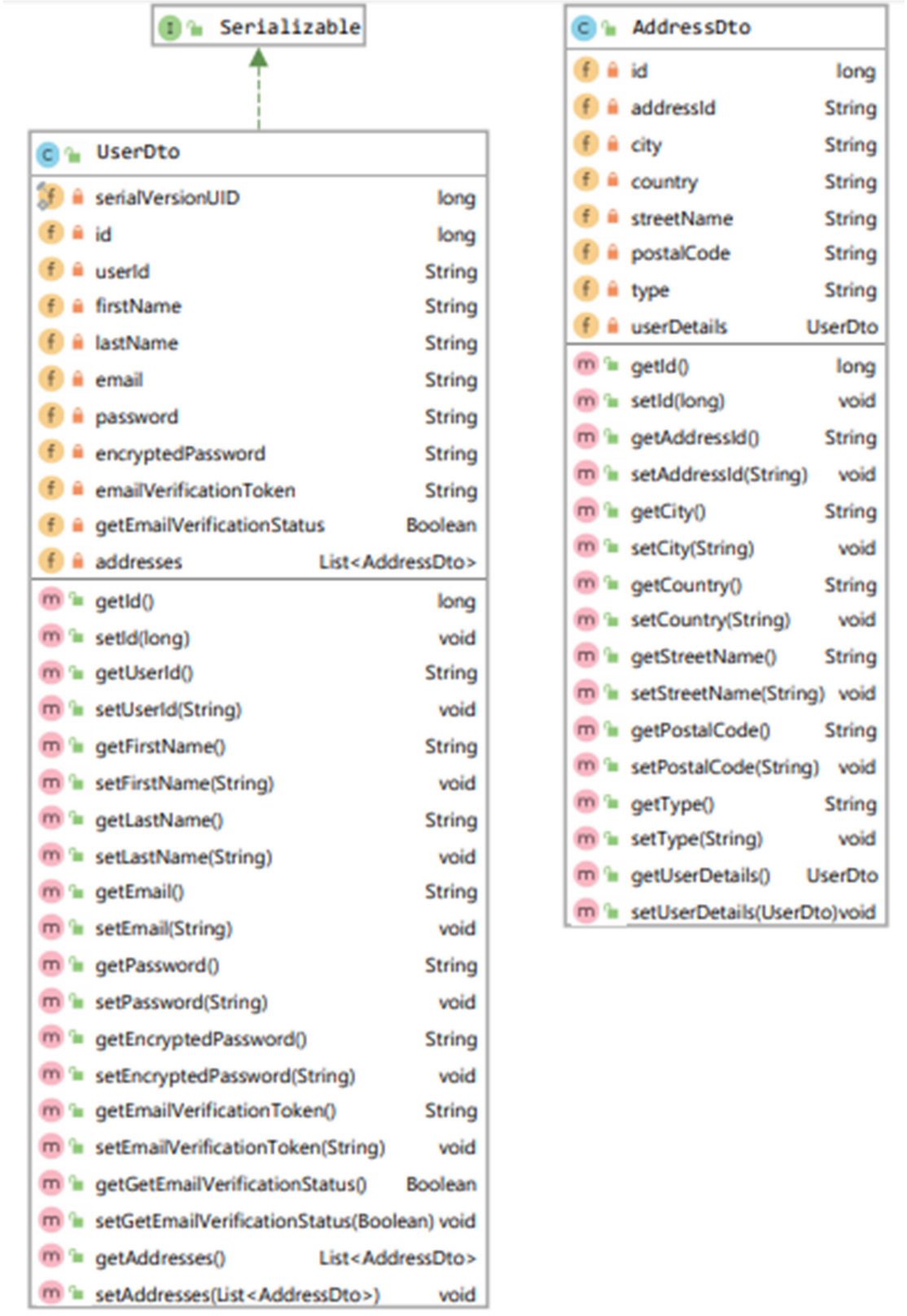
In this we added some additional features for validation purposes.
Below is on details:

Feature 1:

CONTROLLER UML DIAGRAM

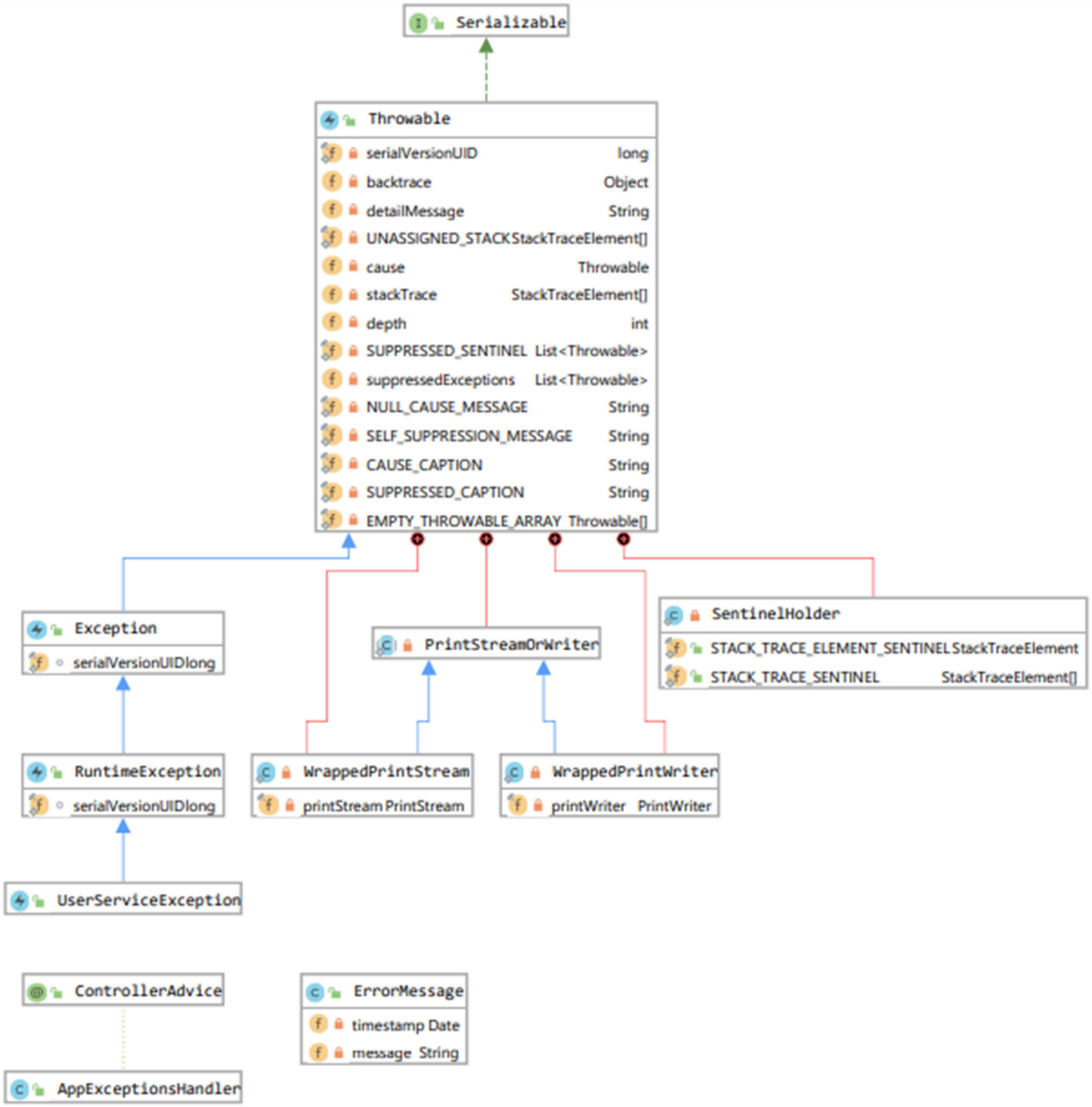


DTO CLASS UML DIAGRAM


Serializable
UserDto
serialVersionUID long
id long
userId String
firstName String
lastName String
email String
password String
encryptedPassword String
emailVerificationToken String
getEmailVerificationStatus Boolean
addresses List<AddressDto>
getId() long
setId(long) void
getUserId() String
setUserId(String) void
getFirstName() String
setFirstName(String) void
getLastName() String
setLastName(String) void
getEmail() String
setEmail(String) void
getPassword() String
setPassword(String) void
getEncryptedPassword() String
setEncryptedPassword(String) void
getEmailVerificationToken() String
setEmailVerificationToken(String) void
getGetEmailVerificationStatus() Boolean
setGetEmailVerificationStatus(Boolean) void
getAddresses() List<AddressDto>
setAddresses(List<AddressDto>) void
AddressDto
id long
addressId String
city String
country String
streetName String
postalCode String
type String
userDetails UserDto
getId() long
setId(long) void
getAddressId() String
setAddressId(String) void
getCity() String
setCity(String) void
getCountry() String
setCountry(String) void
getStreetName() String
setStreetName(String) void
getPostalCode() String
setPostalCode(String) void
getType() String
setType(String) void
getUserDetails() UserDto
setUserDetails(UserDto) void

ENTITY UML DIAGRAM

EXCEPTIONS


Serializable
Throwable
serialVersionUID long
backtrace Object
detailMessage String
UNASSIGNED_STACK StackTraceElement[]
cause Throwable
stackTrace StackTraceElement[]
depth int
SUPPRESSED_SENTINEL List<Throwable>
suppressedExceptions List<Throwable>
NULL_CAUSE_MESSAGE String
SELF_SUPPRESSION_MESSAGE String
CAUSE_CAPTION String
SUPPRESSED_CAPTION String
EMPTY_THROWABLE_ARRAY Throwable[]
Exception
serialVersionUID long
PrintStreamOrWriter
SentinelHolder
STACK_TRACE_ELEMENT_SENTINEL StackTraceElement
STACK_TRACE_SENTINEL StackTraceElement[]
RuntimeException
serialVersionUID long
WrappedPrintStream
printStream PrintStream
WrappedPrintWriter
printWriter PrintWriter
UserServiceException
ControllerAdvice
ErrorMessage
timestamp Date
message String
AppExceptionsHandler

REPOSITORY UML DIAGRAM