AC – Food back end:

1. Requirement analysis:
   1. Requirement functions:

- Menu: save menu’s image into database, get list all menu, get menu by id, create, edit, and deleted menu from database.

- Store: get list all store, get store information by id, create, edit and deleted store from database.

- User: get list all user, get user information by id, create, edit and deleted user from database.

- Orders: get list all orders, get list orders by user’s id, get list order from date to date, create, edit and deleted order from database.

- Users have roles : ADMIN and USER.

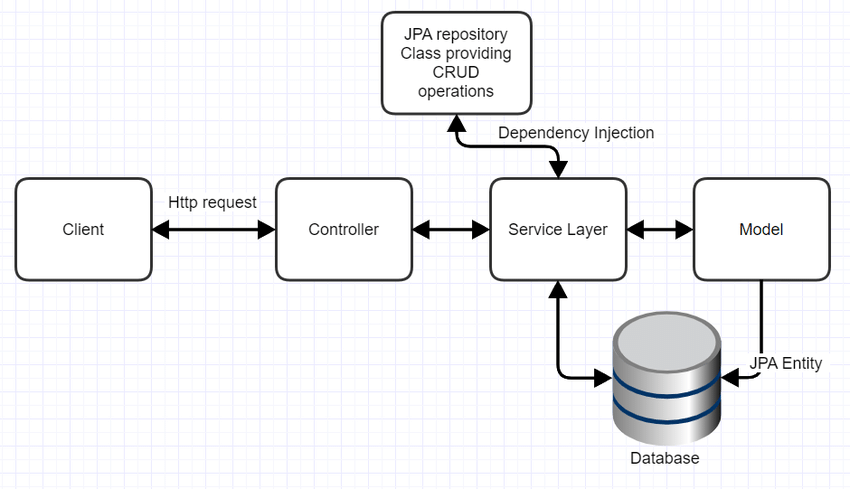
With role ADMIN: user can add, edit, deleted that user, store and menu. Get List all orders, by user’s id and list order from date to date.

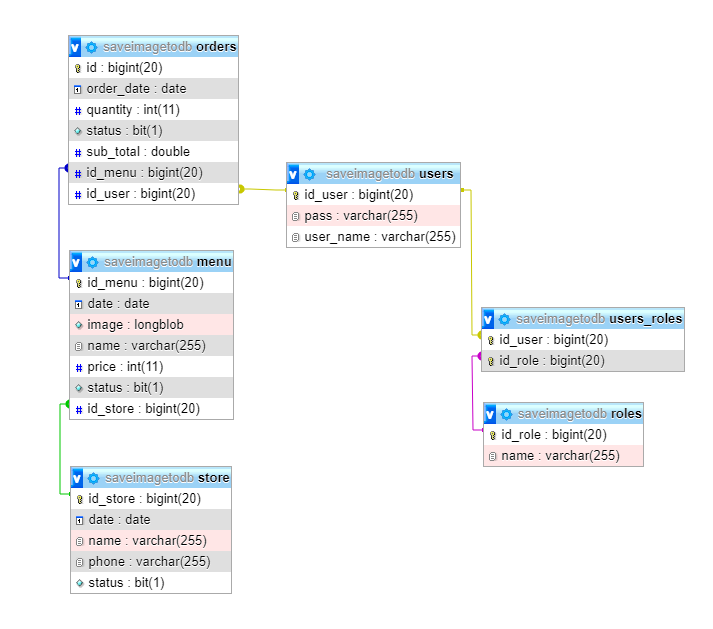
With role USER: user can create and edit orders.

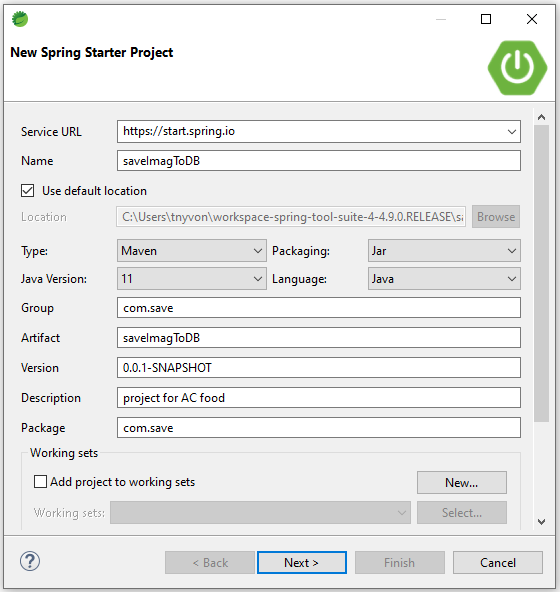
All user have to be authentication and authorization before sending request.

- Customize error.

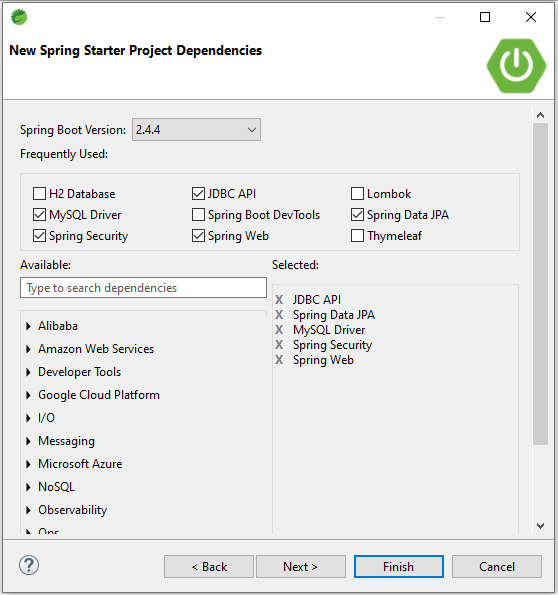
* 1. Technologies used: Spring boot version 2.4.3, JDK.
  2. Tool: build tool Maven, Spring Tool Suite, Xampp, Postman.
  3. Project structure:

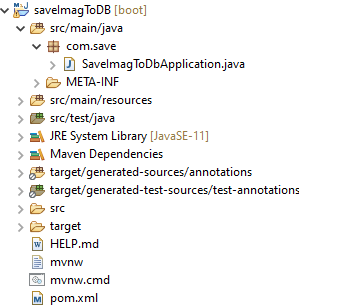


1. Database analysis:
2. Set up environment:
   1. Downloading and configuration JDK and Maven.
   2. Downloading and setup Spring Tool Suite, XAMPP, Postman.
3. Setup Project:
   1. Create a new Spring Starter Project name SaveImageToDB.



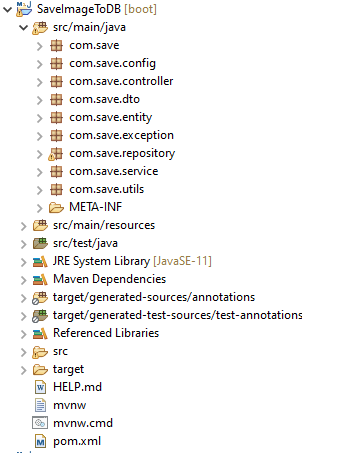
* 1. Adding dependences: Spring Data JPA, Spring Web, Spring Security, MySQL Driver.

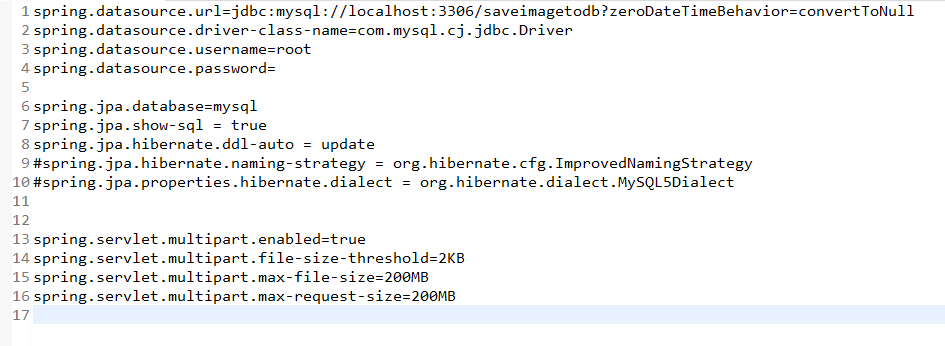


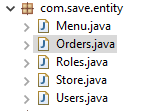
Click finish. Your project’s structure will be like this:

* 1. Create packages:

Include: com.save.config, com.save.controller, com.save.dto, com.save.entity, com.save.exception, com.save.repository, com.save.service, com.save.util .



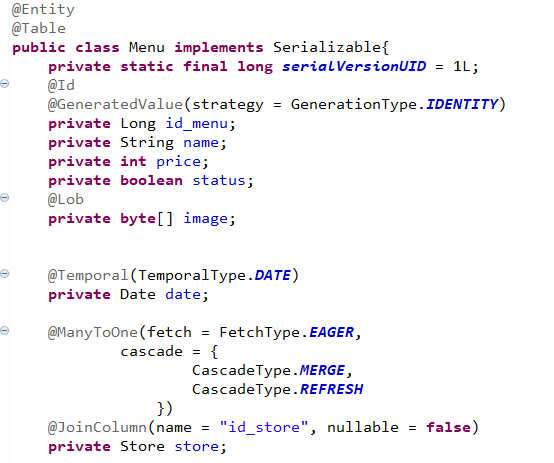
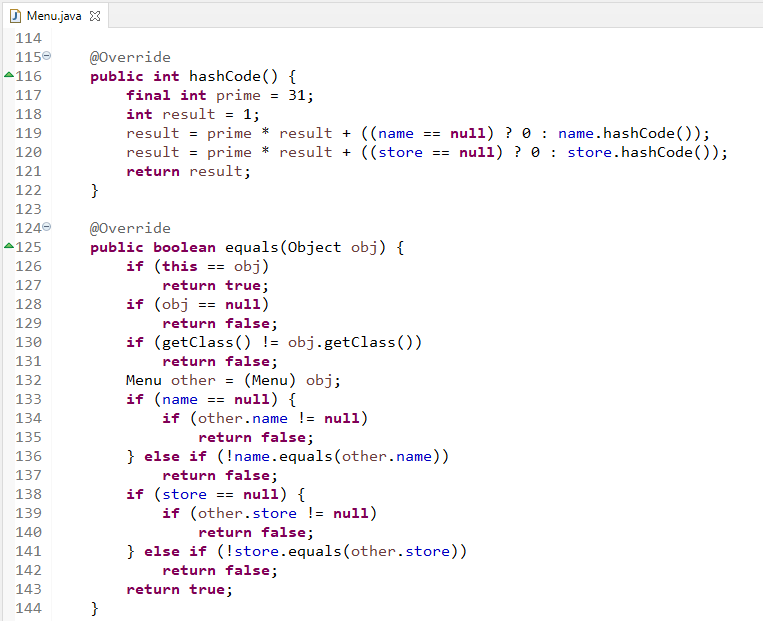
* 1. Configure src/main/resources/application.properties to connect to database and download file.

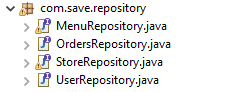
1. Before implement functions:
   1. Create entity classes in com.save.entity package:

Create entity classes with attribute similar to the database and generate getter, setter, constructor method for them. But Menu, Store and User we must add hashCode() and equals() methods which use for avoiding duplicate data.

Notice about the annotations: if you don’t declare these annotation. There won’t be automation create table in database.

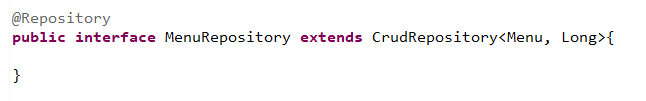
For example: Menu class

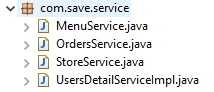
 

* 1. Declaring interfaces in com.save.repository package:

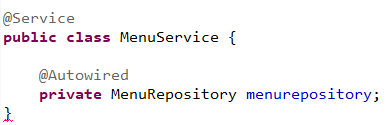
Notice to declare @Repository annotation and extends CrubRepository for each interface.

For example: MenuRepository interface

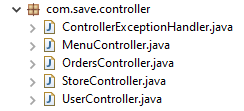


* 1. Declaring services in com.save.service package:

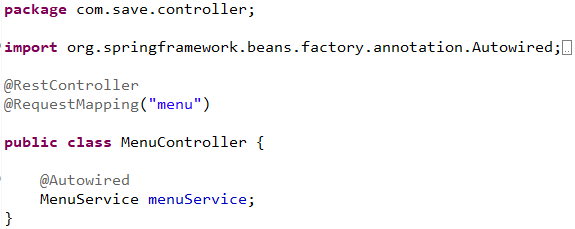
Notice to declare @Service annotation and @Autowired annotation to inject repository class corresponding to service class.

For example: MenuService class

* 1. Declaring Rest controller in com.save.controller package:

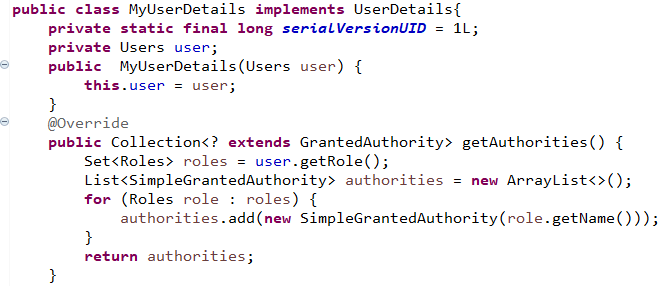


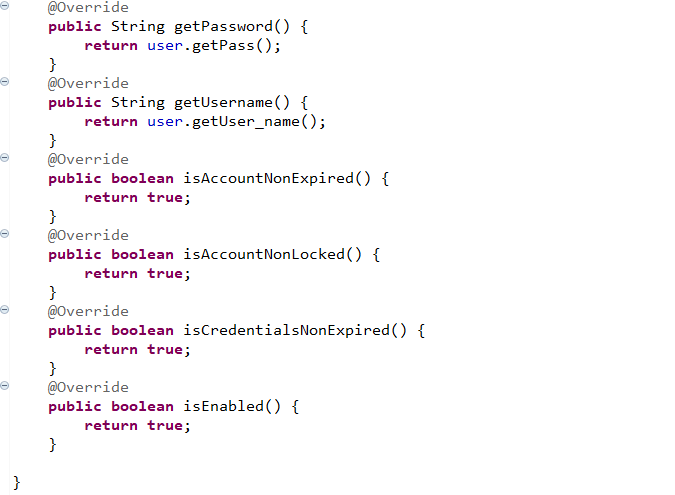
Notice to declare @RestController annotation and @Autowired annotation to inject service class corresponding to rest controller class.

For example: MenuController class

1. Implement functions:
   1. About authentication and authorization:

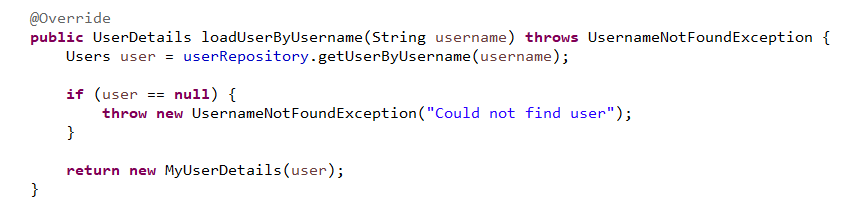
Create MyUserDetail class to implements interface UserDetails of [org](eclipse-javadoc:%E2%98%82=SaveImageToDB/C:%5C/Users%5C/tnyvon%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/spring-security-core%5C/5.4.5%5C/spring-security-core-5.4.5.jar=/maven.pomderived=/true=/=/org.eclipse.jst.component.nondependency=/=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.security=/=/maven.artifactId=/spring-security-core=/=/maven.version=/5.4.5=/=/maven.scope=/compile=/%3Corg).[springframework](eclipse-javadoc:%E2%98%82=SaveImageToDB/C:%5C/Users%5C/tnyvon%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/spring-security-core%5C/5.4.5%5C/spring-security-core-5.4.5.jar=/maven.pomderived=/true=/=/org.eclipse.jst.component.nondependency=/=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.security=/=/maven.artifactId=/spring-security-core=/=/maven.version=/5.4.5=/=/maven.scope=/compile=/%3Corg.springframework).[security](eclipse-javadoc:%E2%98%82=SaveImageToDB/C:%5C/Users%5C/tnyvon%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/spring-security-core%5C/5.4.5%5C/spring-security-core-5.4.5.jar=/maven.pomderived=/true=/=/org.eclipse.jst.component.nondependency=/=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.security=/=/maven.artifactId=/spring-security-core=/=/maven.version=/5.4.5=/=/maven.scope=/compile=/%3Corg.springframework.security).[core](eclipse-javadoc:%E2%98%82=SaveImageToDB/C:%5C/Users%5C/tnyvon%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/spring-security-core%5C/5.4.5%5C/spring-security-core-5.4.5.jar=/maven.pomderived=/true=/=/org.eclipse.jst.component.nondependency=/=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.security=/=/maven.artifactId=/spring-security-core=/=/maven.version=/5.4.5=/=/maven.scope=/compile=/%3Corg.springframework.security.core).[userdetails](eclipse-javadoc:%E2%98%82=SaveImageToDB/C:%5C/Users%5C/tnyvon%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/spring-security-core%5C/5.4.5%5C/spring-security-core-5.4.5.jar=/maven.pomderived=/true=/=/org.eclipse.jst.component.nondependency=/=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.security=/=/maven.artifactId=/spring-security-core=/=/maven.version=/5.4.5=/=/maven.scope=/compile=/%3Corg.springframework.security.core.userdetails).UserDetails as below:





Declare UsersDetailServiceImp class implements UserDetailService and add method loadUserByUsername(String username), in order to check login information.

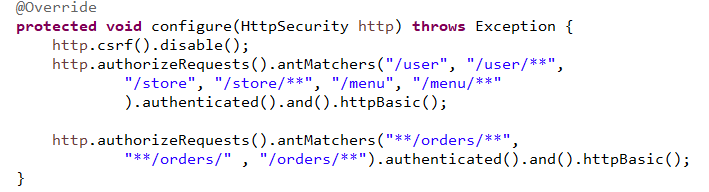
As below: UsersDetailServiceImp class



WebSecurityConfig class used to configure security for the application, it is annotated with @Configuration annotation, which tells Spring that it is a configuration class, and will be parsed by Spring at the time the application is run.

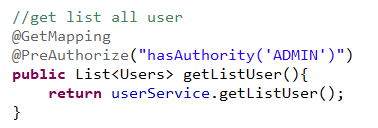


Paying attention to configure(HttpSecurity http) method.

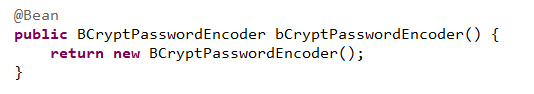


There is the place for declare which request need to be authorized and authenticated before user access. It will combine with @PreAuthorize annotation for any request need to implement at rest controller class.

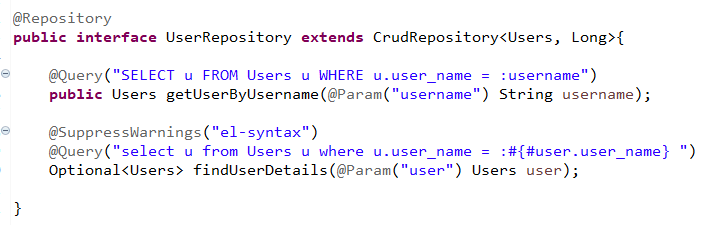
For example: UserController class

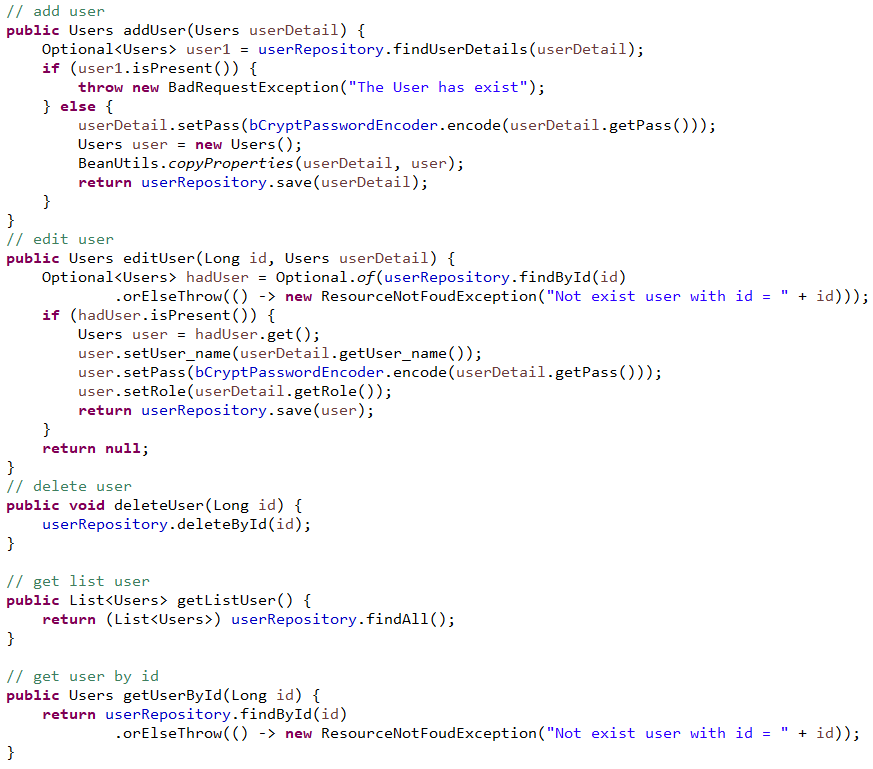
To get list all user, there must be authenticate and authorize with user’s role ADMIN.

After that, we declare a BCryptPasswordEncoder bean in SaveImageToDbApplication class to encode user’s password when a user is stored into database.

As below: SaveImageToDbApplication class

* 1. About user’s functions:

UserRepository interface:

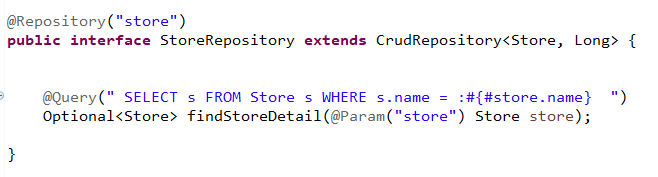
UsersDetailServiceImp class:

UserController class:

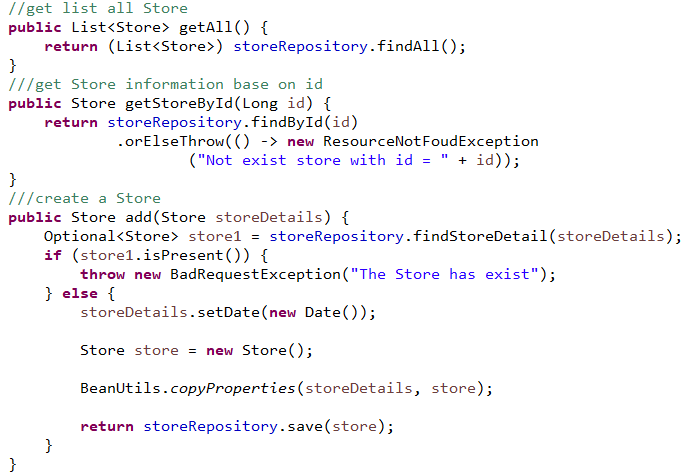


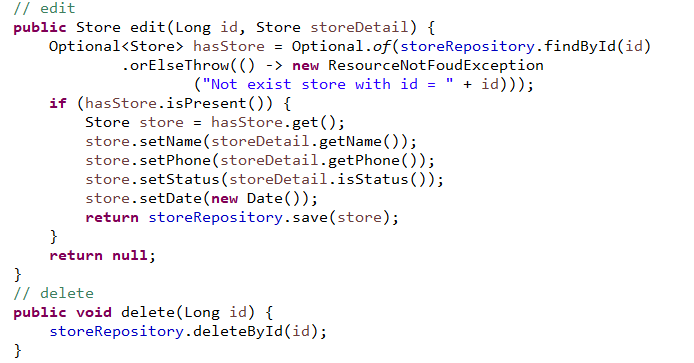
* 1. About Store’s functions:

StoreRepository interface:

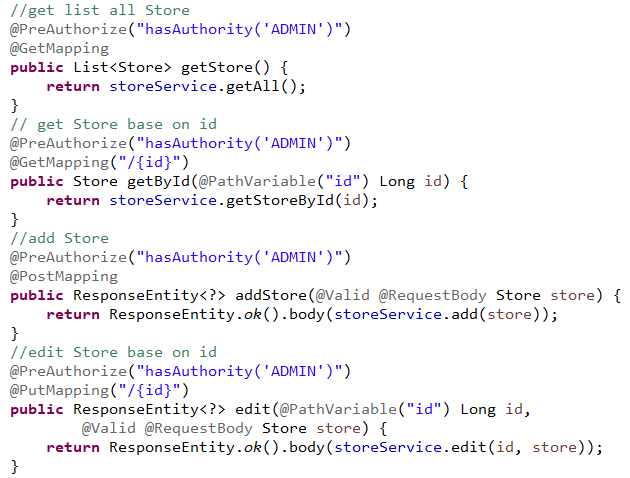


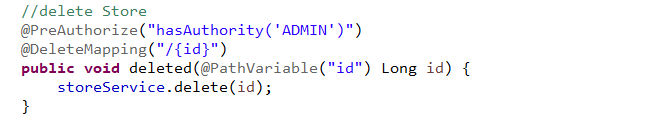
StoreService class:





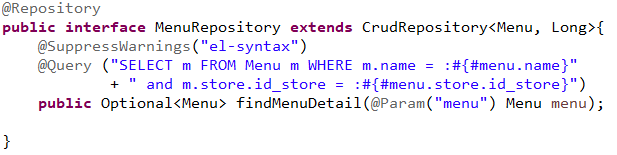
StoreController class:

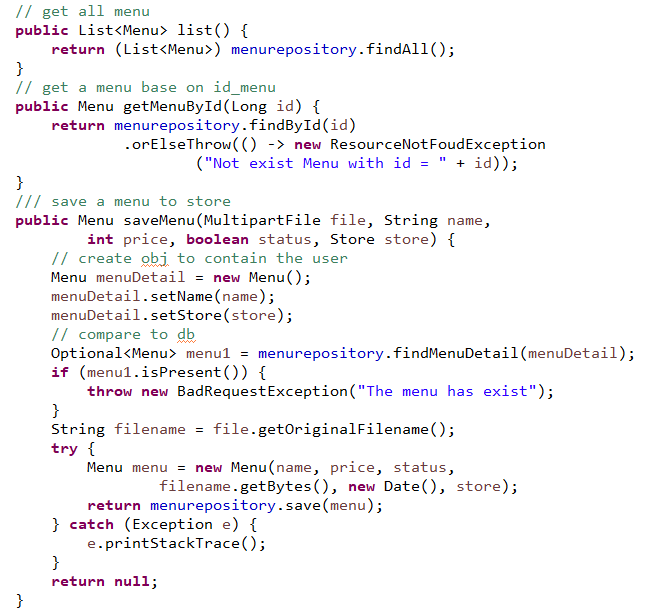


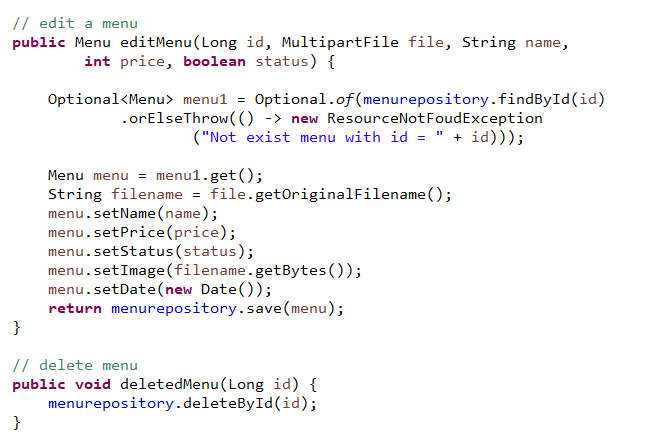


* 1. About functions of menu:

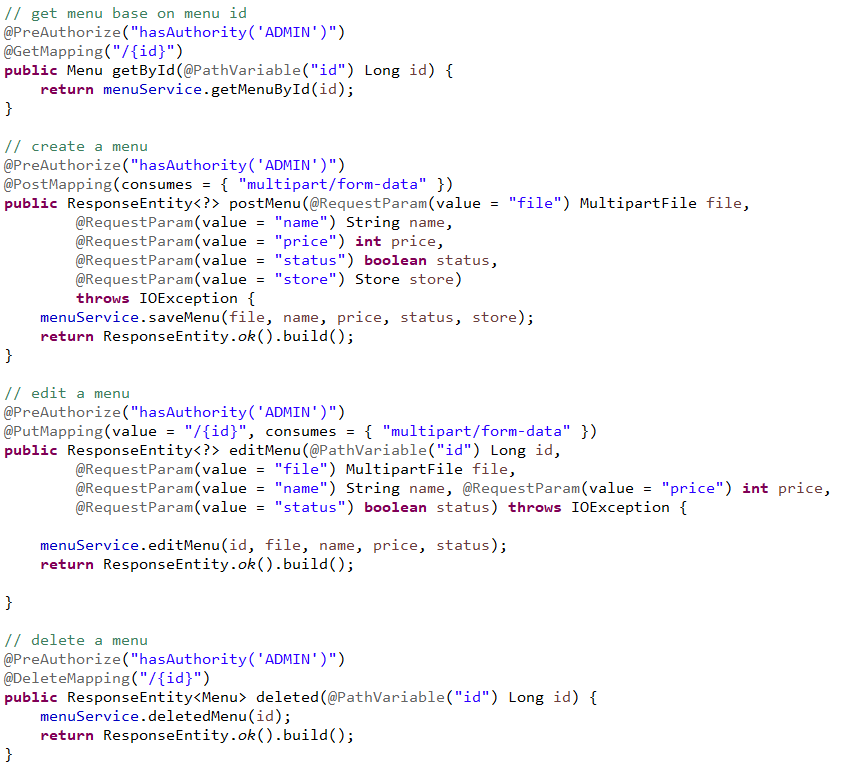
MenuRepository interface:



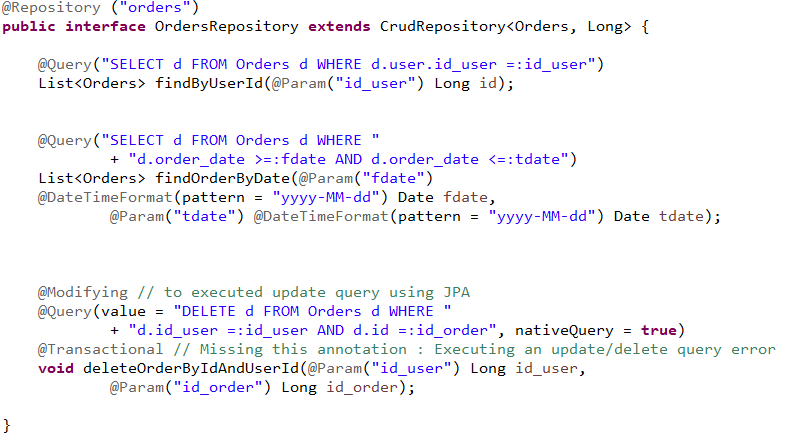
MenuService class:

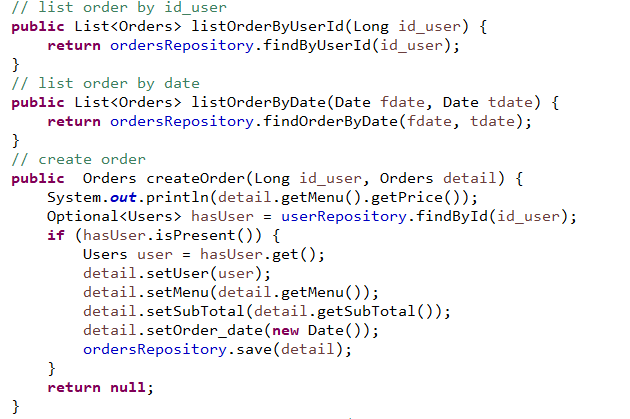


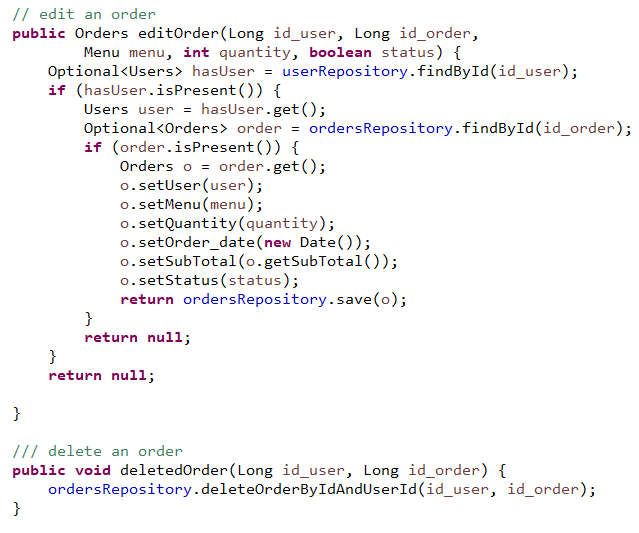
MenuController class:

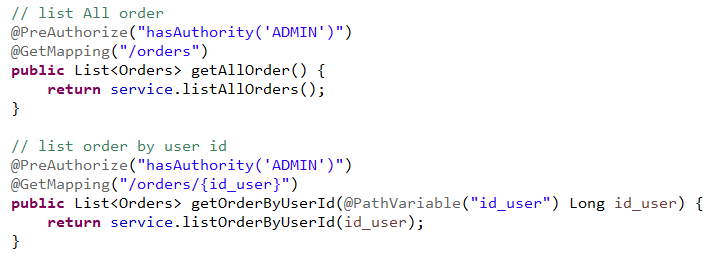


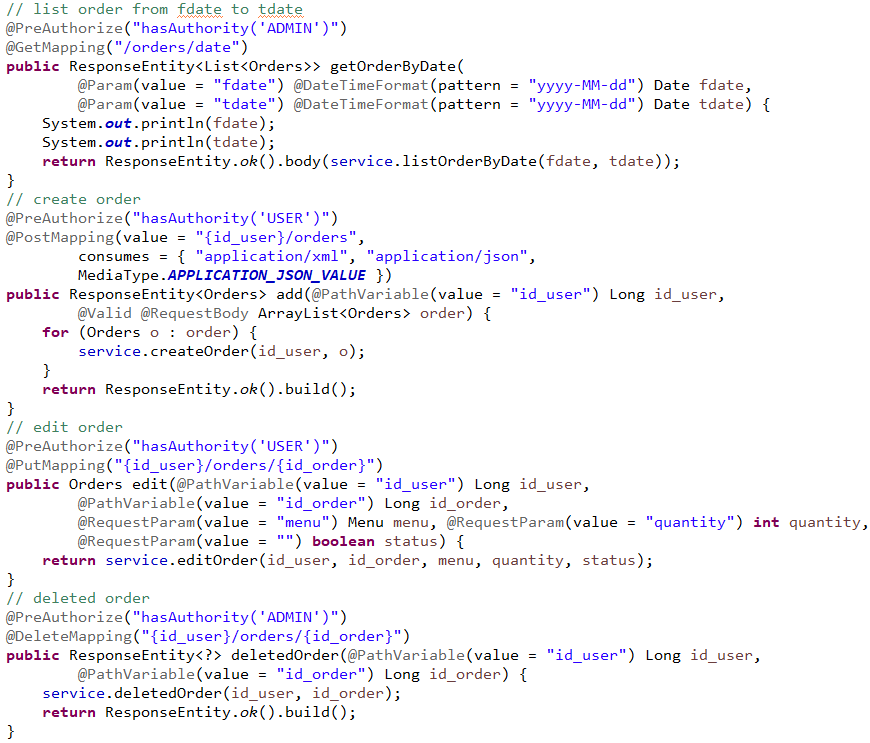
* 1. About functions of Orders entity:

OrdersRepository interface:

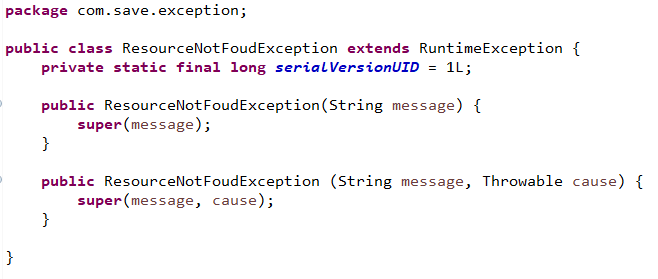
OrdersService class:

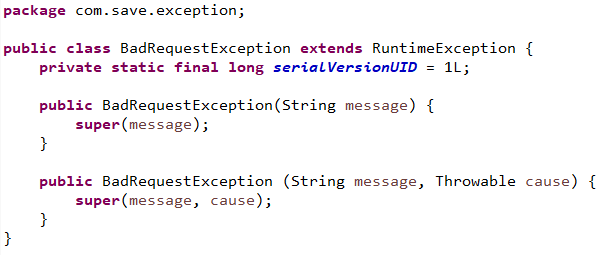


OrdersController class:

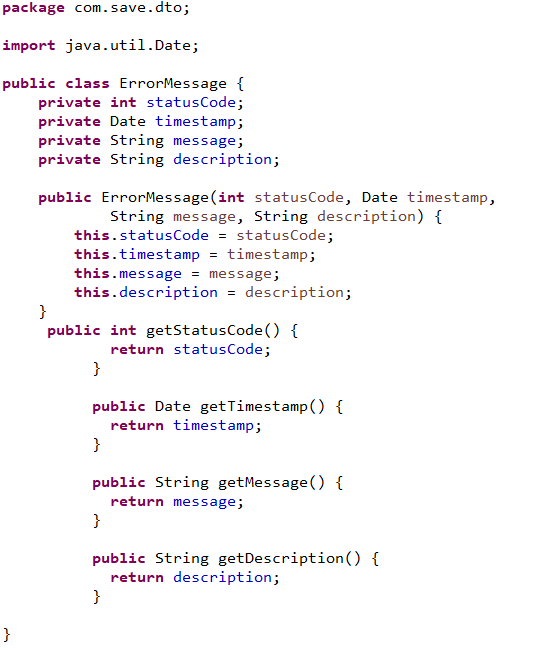


* 1. About error customize:

Define error: ResourceNotFoundException and BadRequestException in com.save.exception package as below.



Define error message in com.save.dto package:



Define a rest controller advice class to handler error in com.save.controller packge.

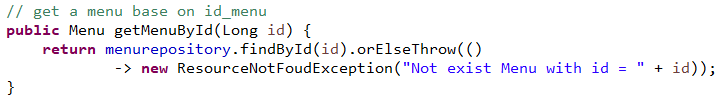
As below: ControllerExceptionHandler class

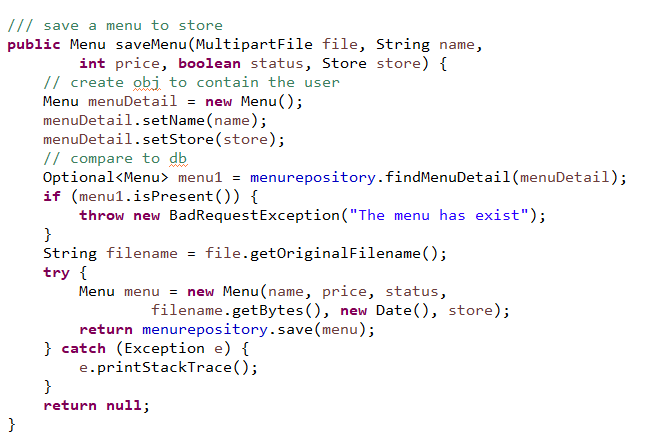


Now, an exception is thrown will display with figures are defined in ErrorMessage class.

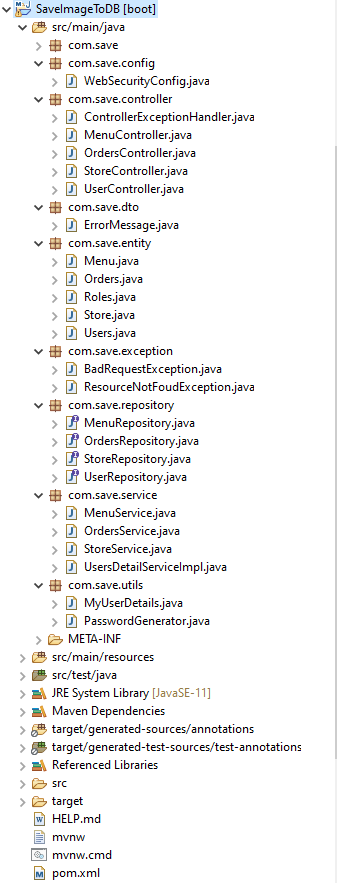
Then, all we need is throw the exception in logical layer.

For example: MenuService class

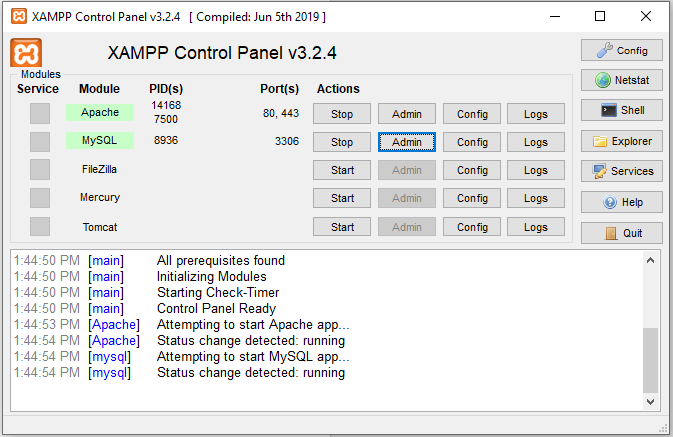


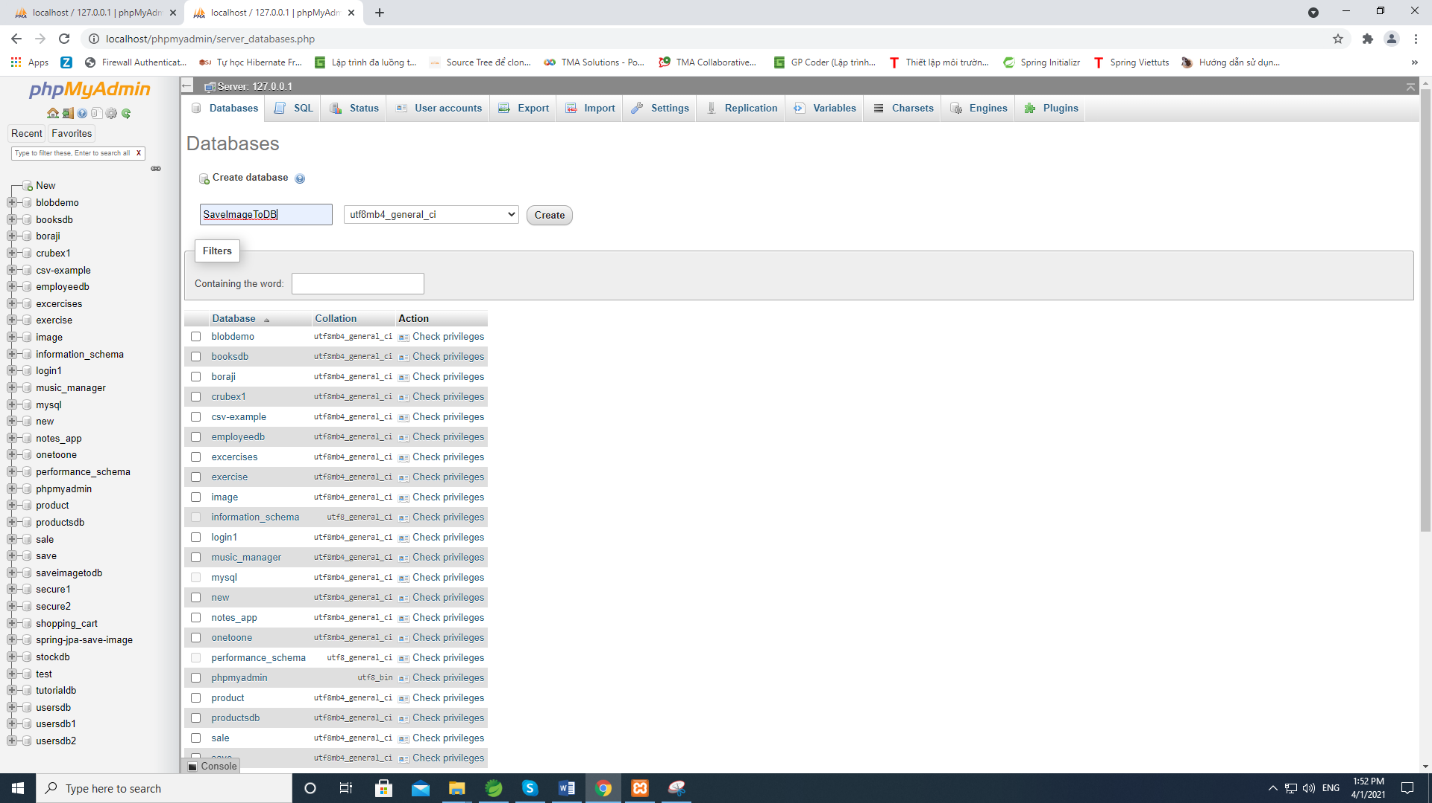


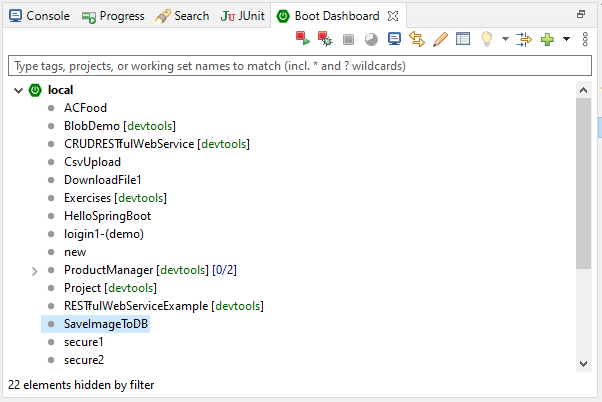
Finally, the structure of project will similar as below:



1. Running:

Open Xampp to start Apache and MySQL

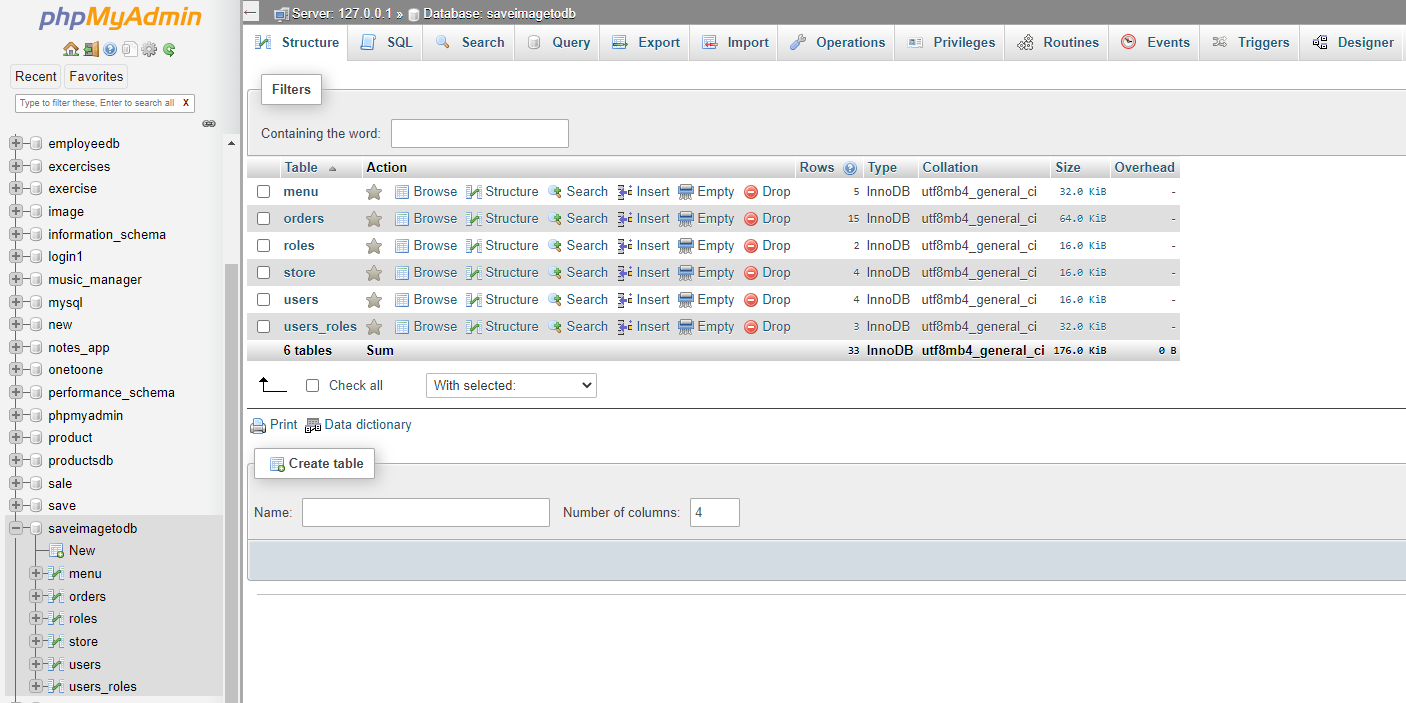
Create a database saveimagetodb in phpMyAdmin (this site will appear when click admin (MySQL) at xampp)

At Spring Tool Suit: open boot dashboard

Right click in SaveImageToDB -> choose Restart

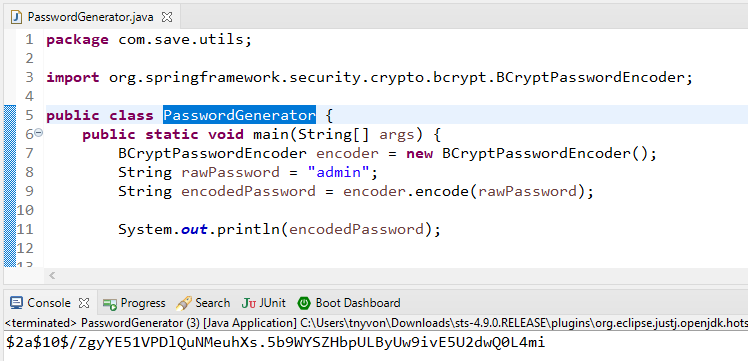
See results on Spring Tool Suite console and phpMyAdmin





Before testing, we have to insert a user’s information into table user. Running PasswordGenerator class as java application to encoder the password before store it into database.

For example: PasswordGenerator class

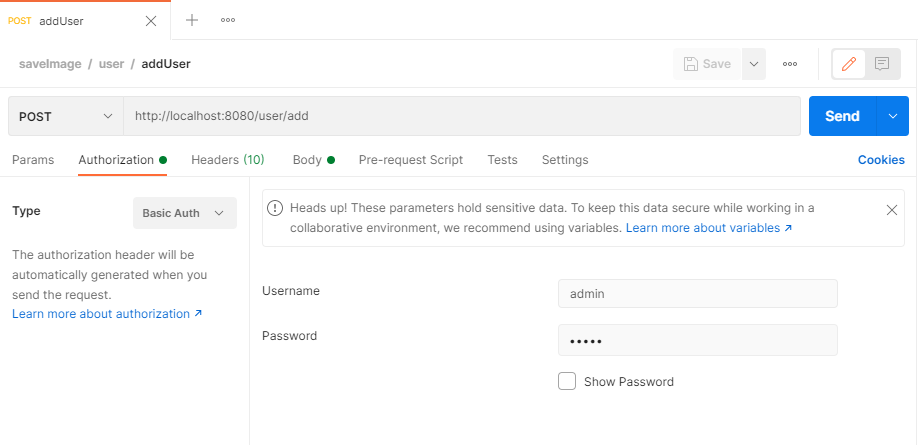


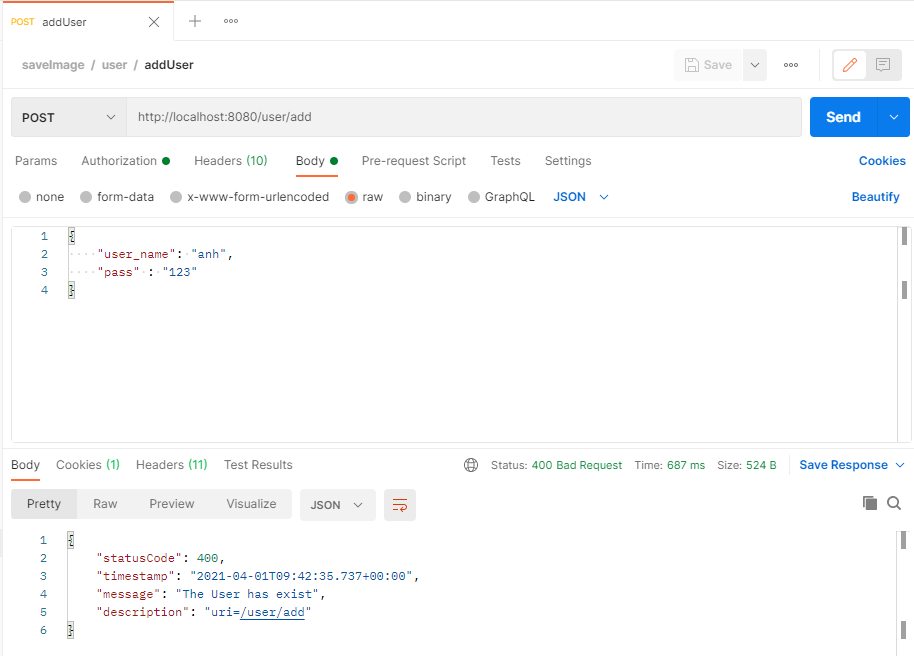
1. Testing:

Using Postman to test rest API have been written

***Attention***: each request have to be authenticate before accessing, and it will base on user’s role to decide which request have authorization to access.

For example: test add user request (<http://localhost:8080/user/add>)

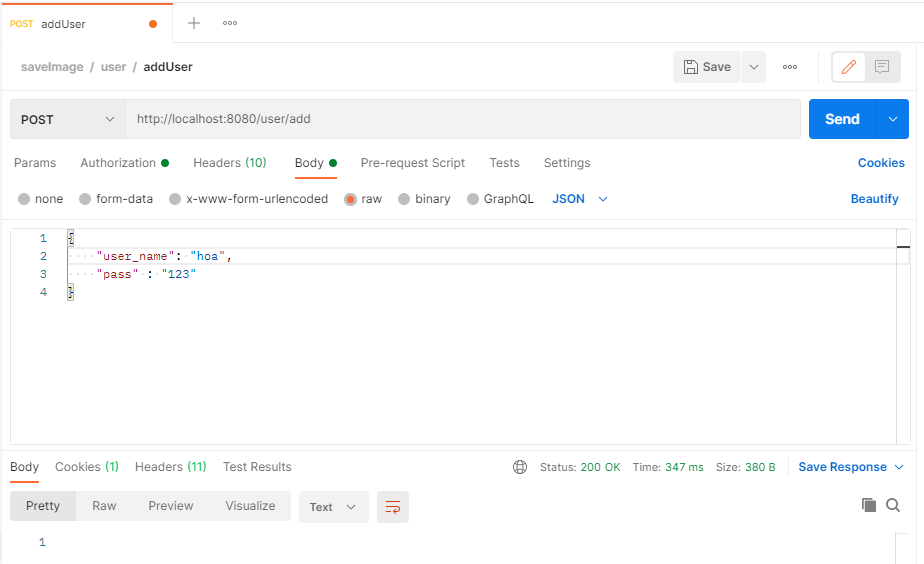




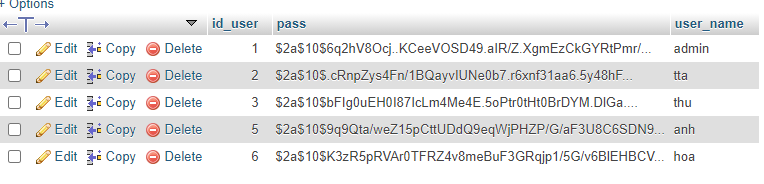
Starting to test some request now:

1. User’s functions:

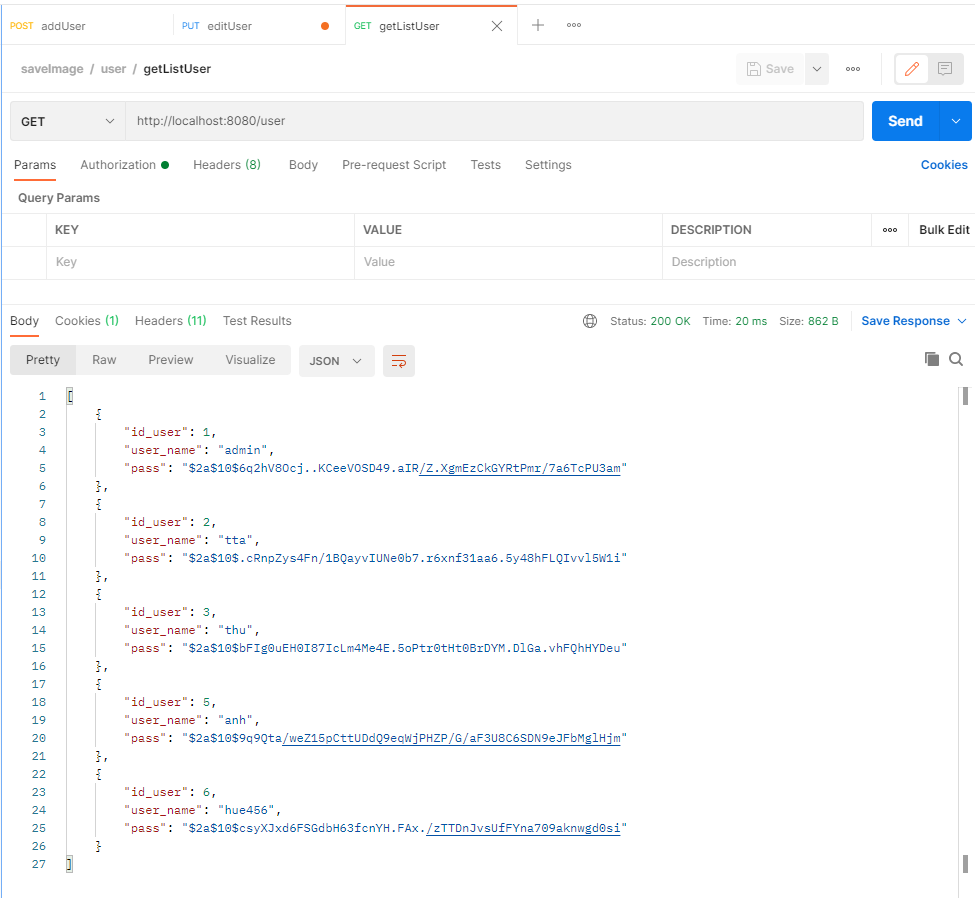
* Add user:



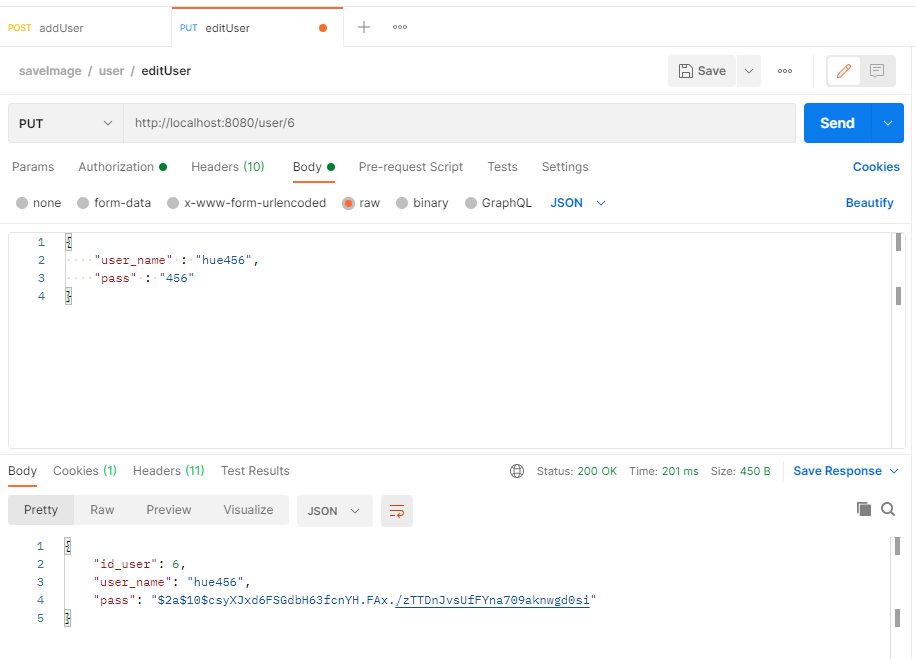
Look at user with id\_user = 6, it is the user just have added.



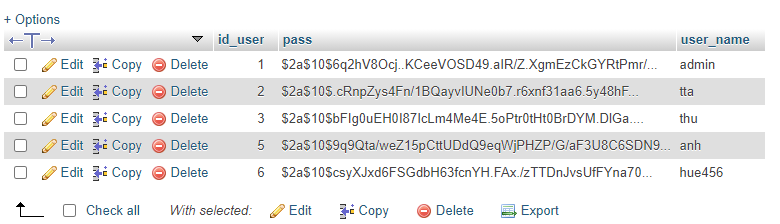
* Get list user:



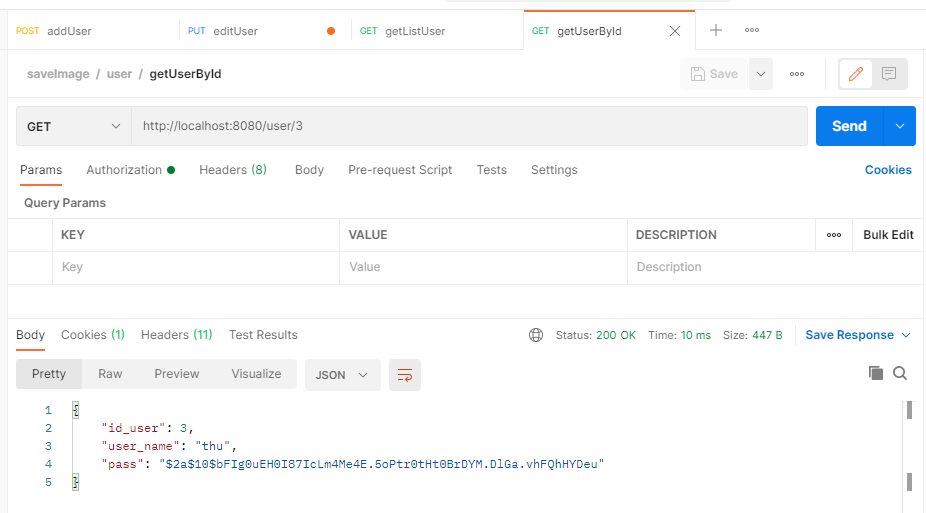
- Edit user with id\_user = 6:



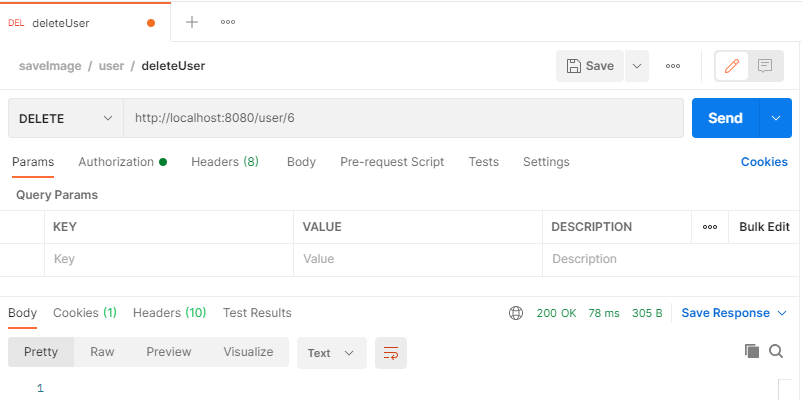
Look at user with id\_user = 6, it is resaved with new information.



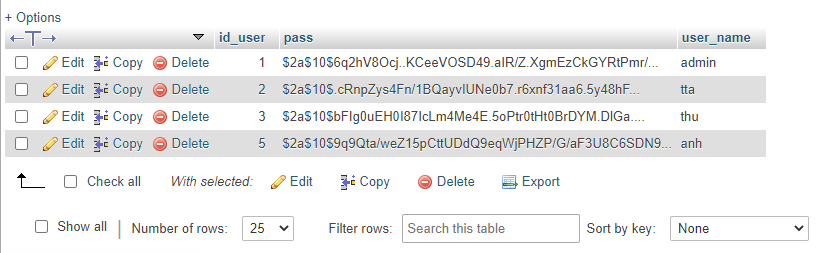
* Get user’s information base on id\_user:



* Delete user’s information base on id\_user.

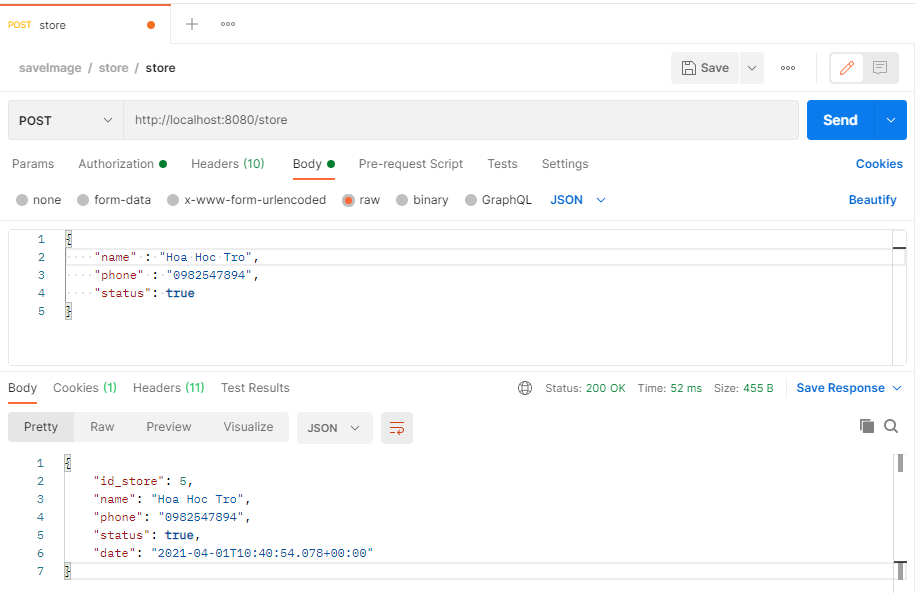


Look at users table, user with id\_user=6 have been removed.

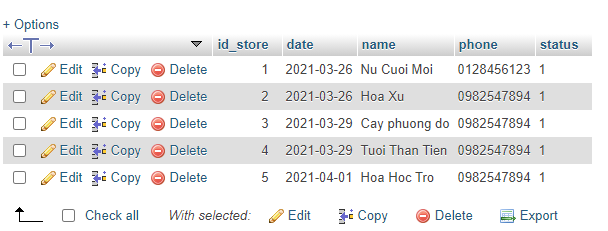


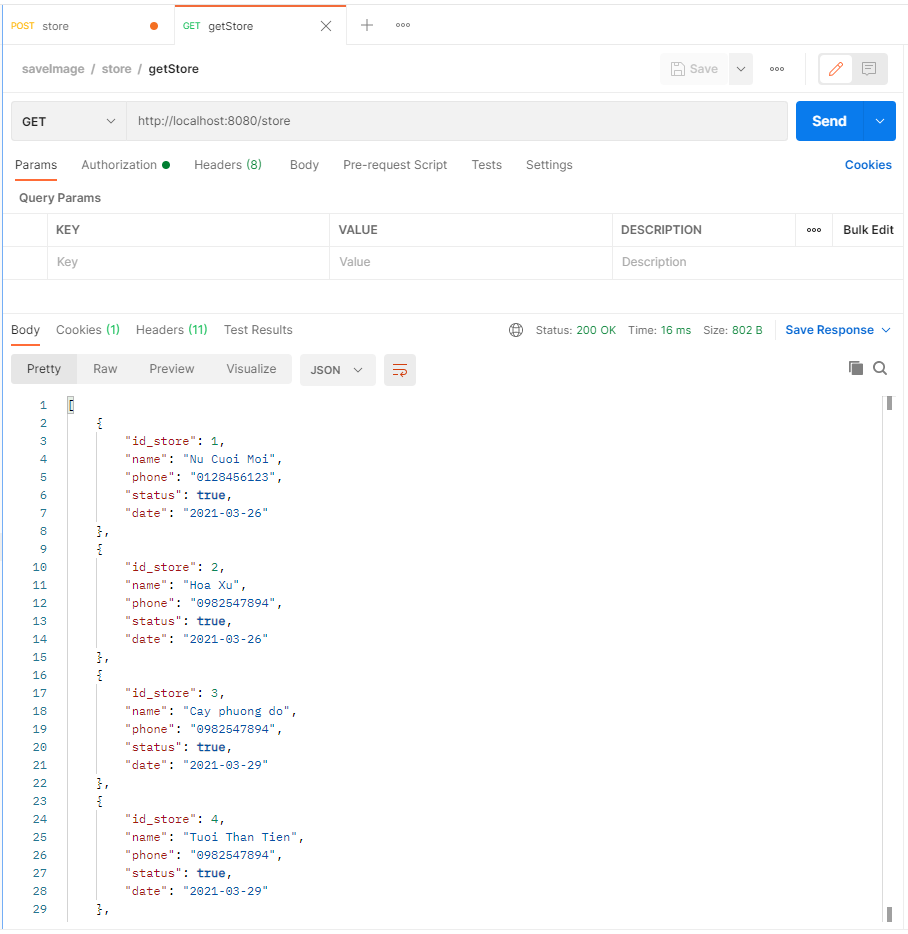
1. Store’s functions:

* Add Store:



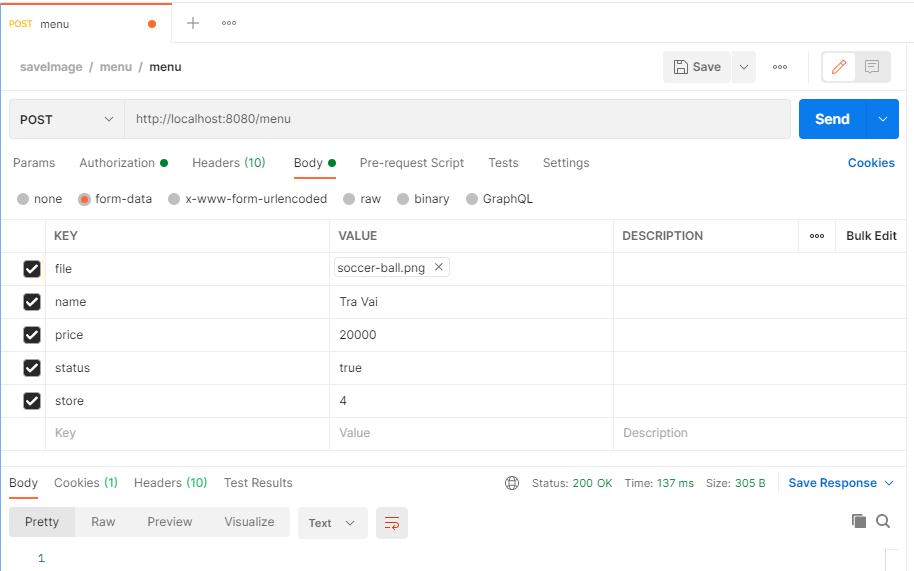
Look at the store with id\_store= 5, it is the user just have added.



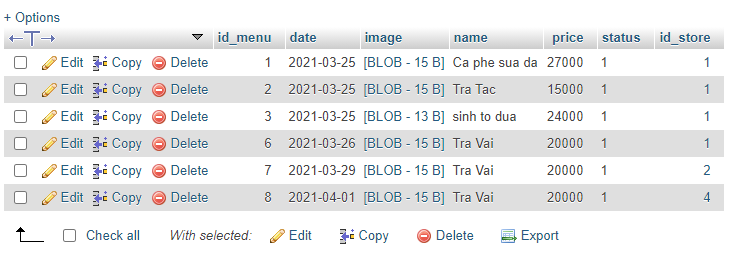
* Get list all Store:
* Delete, edit, get store by id\_store, will be test similar to delete, edit, get user by id\_user above.

1. Menu’s functions:

* Add menu:



Look at Menu table with id\_menu = 8, it is the menu has just added.



* Delete, edit, get store by id\_store, will be test similar to delete, edit, get user by id\_user above