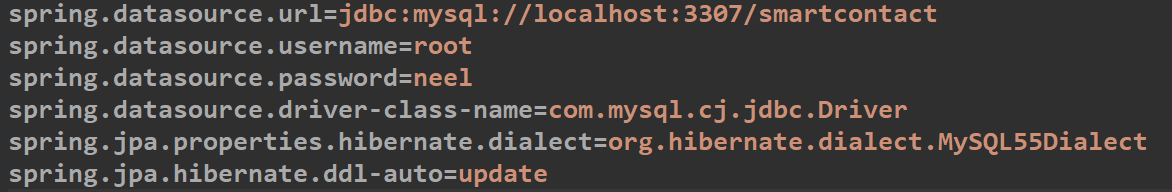
**\*There exists an method for the conversion of the object to the josn directly via httpResponse, Use that method than the primary methods that are used in the many of the method in the documentation**

**Contact Manager Info**

1. In this project we will use the following dependencies of the Spring Boot
   1. Starter-web
   2. Stater-thymeleaf
   3. Stater-data-jpa
   4. Starter-security
   5. Spring- boot-devtools
   6. Mysql-connector-java
   7. Validation-api
   8. Hibernate-validator
2. At first we will not include the dependencies of the security and the validation api and hibernate validator which we will put in the project afterwards
3. Than we will configure the database in the application.properties, Whose configuration is as shown below:

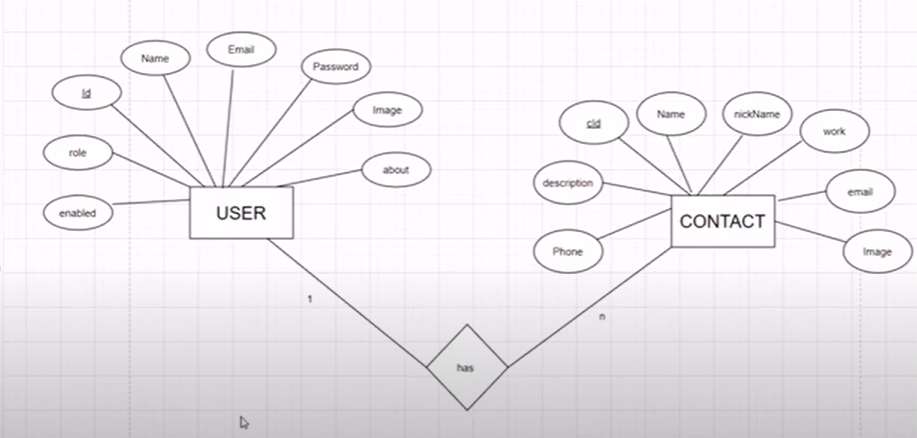


1. Now to make the designing of the database to be simpler, we make the E-R diagram (Entity Relation Diagram) which helps us to plan our project and the er diagram

🡪For making the er-diagram we can take the help of the website named **draw.io**

🡪In er diagram rectangle box indicates the entity and the thing in the oval shape represents the column of the entity and the relations between the two entities are represented by help of the diamond shape, and by connecting the two entities to it by help of the line we define it would have what type of the relationships

🡪As the relation between the user and the contact is of type OneToMany, we write the one on the line connecting the entity and diamond, And as the relation of contact is of many, We write the n there, Example:

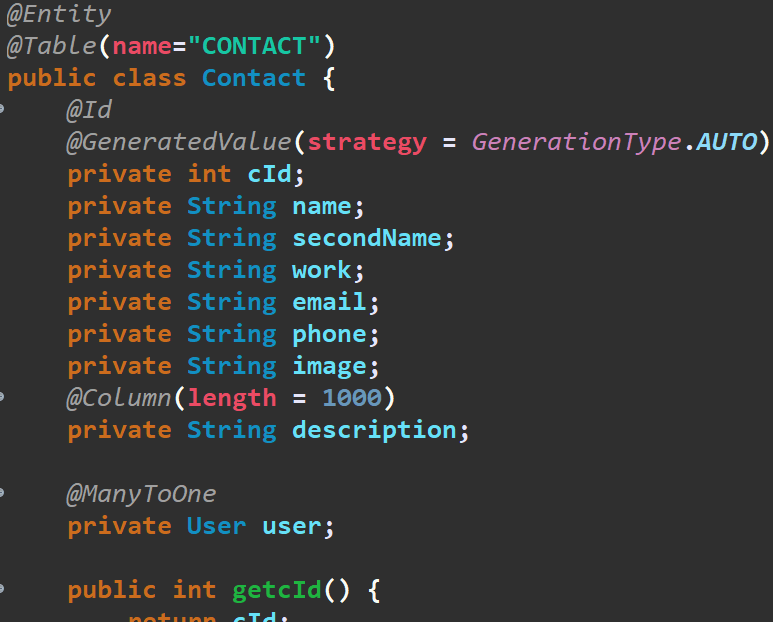


1. Now making the entities named user and the contact in the new entities package, Its example is as followed:

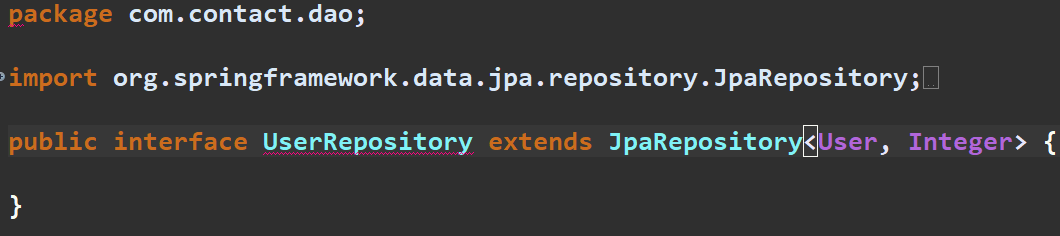
🡺User.js



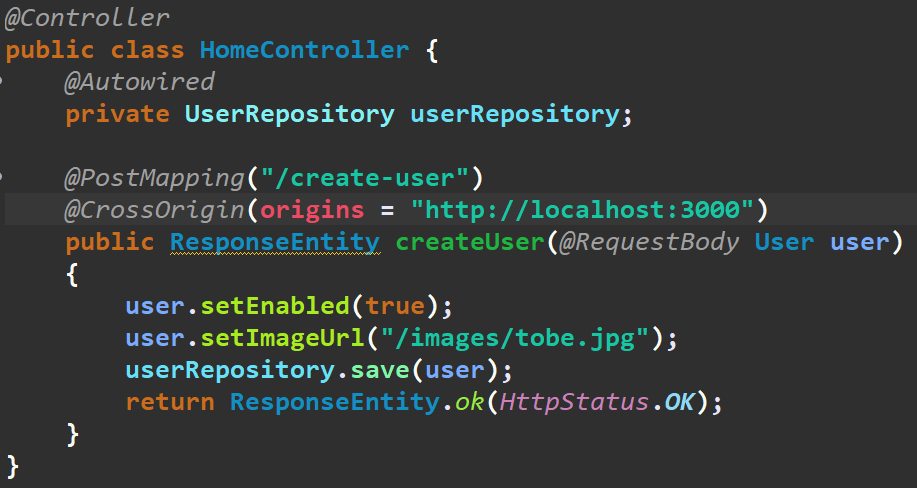
🡺Contact.js



1. Than we will make an repository by help of which we can perform the different functions to the table of the user and for it we will make the userRepository and for now we will use the in-built functions given by the JpaRepository or the CRUDRepository and after wards of we require the another functions than we will make them, Example of userRepository is as follow:



1. Now, Comes the main part of the controller and for it we will currently having the only one file for the controller named HomeController and in it first we will be having the controller of the create user, As shown bellow:



🡪In this we will make the annotation named the controller on the top of the class name

🡪And in this controller we are also using the userRepository, So instead of the importing it we will get it by help of the autowired annotation

🡪And than on the top of the function by which we will control the action in this case the creation of the user, We will use the mapping annotation having the method of the our requirement usage like post, get, etc having the url in it of when to make the action

🡪And than we have the crossOrigin annotation, by help of which we can overcome the crossOrigin error and give the access of the our method to our required server

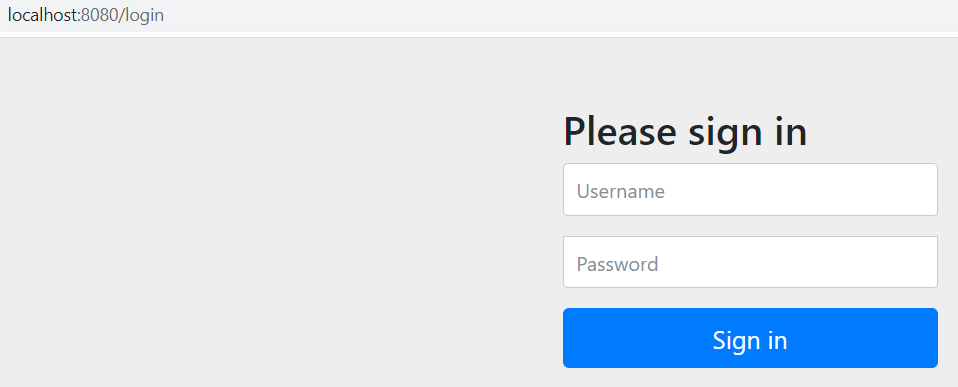
🡪And than to accept the any body of the a type we will use the annotation named RequestBody and than we will give the object name of which type the object that is coming is and than we will make the refrence of that object

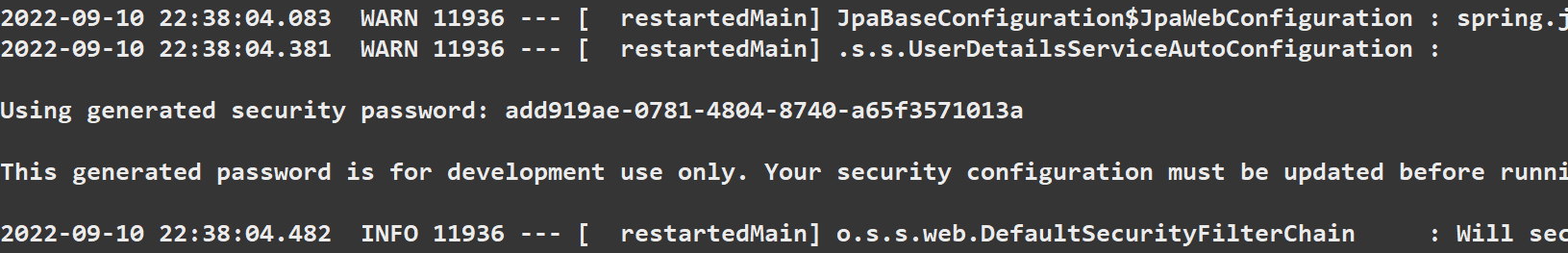
🡪And than we can use the different methods of the user class to make the addition in the object that has commed as shown in the example

🡪Than by help of the save method from the userRepository we will save the object in the database that had commed from the url, And we can also use the any other method from the repository for making the different database related function

🡪And than we can send the response by help of the ResponseEntity class, And instead of the ok we can use the different things also, And we can also send the status in the response by help of the httpResponse and there instead of the .OK, we can use the another method also

1. Now we can also make the password to be encoded, decoded, securing the specific url by help of the spring security as shown in below points:
   1. For getting it we would have to include the **spring boot starter security** dependency from the mvn repository
   2. And after including this dependency all the link of the projects will be blocked by the spring security and that can only be accessed when we make login in the form that is available at the **<port-number>/login** link
   3. And if we logged in one time than we can use the all the link of the project
   4. And if we want to log out than we can use the **<port-number>/logout**
   5. And for log in we can use the user as the username and the password will be coming in the terminal of the spring boot, Example is as shown in the figure:

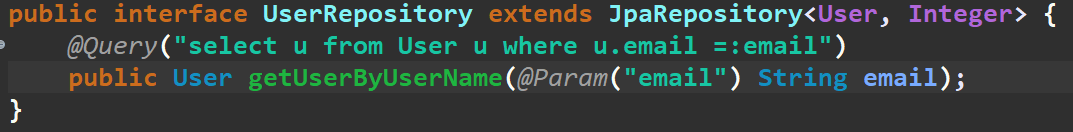




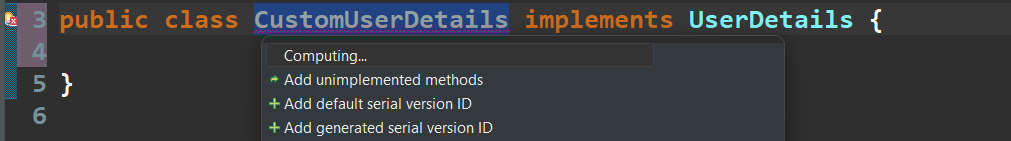
* 1. And now we have to change the system that is by default given by the spring security and we can change the password, username to email and also the html page as per our requirement
  2. And for making the configuration of spring security, And how to use the database to verify the user and password we have to make an role based authorization system with the help of spring security
  3. And for making the role-based authorization system we have to follow the following three steps(Overall main) System:
     1. Provide the implementation of **UserDetails** in an class named **CustomUserDetail** (To be made) and init we have to give the body of the methods of the implemented interface
     2. Provide the implementation of **UserDetailService** in an class named **UserDetialServiceImpl** (To be made) and init we have to give the body of the methods of the implemented interface
     3. Make security configuration class with all configuration, A simple class that extends the **WebSecurityConfigurationAdapter** and by doing that we get the methods to customize the settings according to our requirement

🡺Following the above steps and some extra one in detail one by one:

1. Now we will go to the repository interface which we have made by name of UserRepository in this project, And in it we will make an method by name getUserByUserName and it can also have the any other name as per our project
2. Than on it we will write the **@Query** annotation and in it we will write the query, And in the query we have to give the parameter by help of the special operator and than we can put that parameter’s value by help of the @Param annotation while calling it, Example of it is shown below:

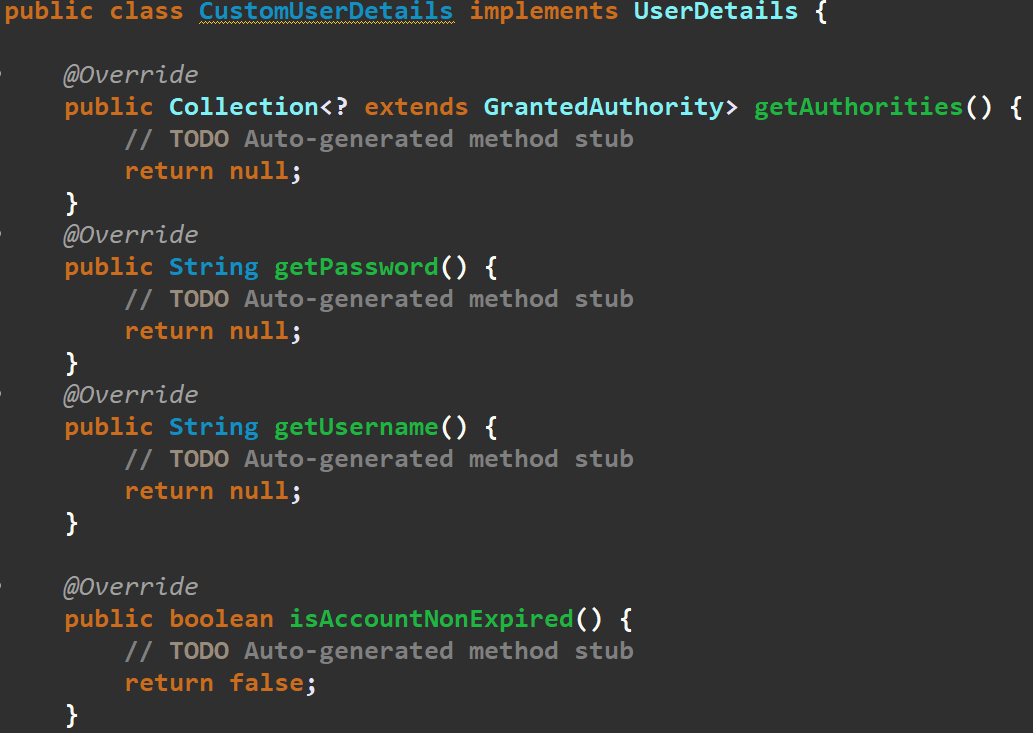


1. Now lets come to the first step that is to make the CustomUserDetail class, And for it we will make an new package having the .config name
2. And in this class that we have made we have to implement the interface named UserDetails
3. And to add the un-implemented method of the interface that we have just implemented we will click on the light bulb that is been showed by the ide and in it we will select the **Add unimplemented methods**, As shown below:

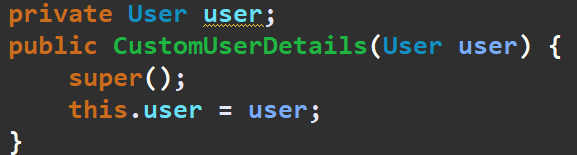


🡪Make sure you make the double click on add unimplemented methods otherwise the methods will not be implemented

1. And due to it the unimplemented methods will be implmeneted, Example of it is as shown below:



1. And than as we want to get the details of the user and work with the user we will declare an private variable of it and will also make the constructor for it, Hence we can get the details of that particular user and can use the details of it, Example:



1. And now we can get the details like password, username etc from thr user and can make them return back as per required, So we will make the some change in the override methods that we have inherited, Example:

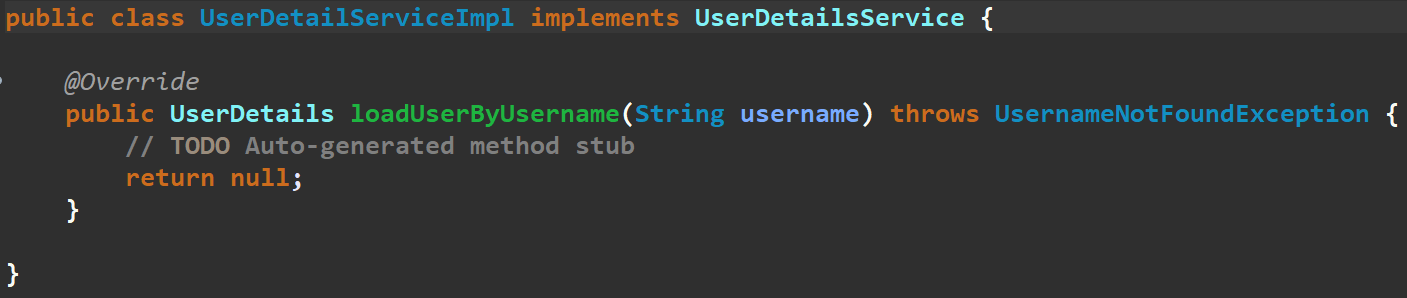
🡪Here we have given the user.password() in return in the getPassword method

🡪And than in the username we have given the email as we have it as Quique one in our table

🡪And in some of the methods like isEnabled and many more we have make them true, And if any changes would be required than we will manage them later

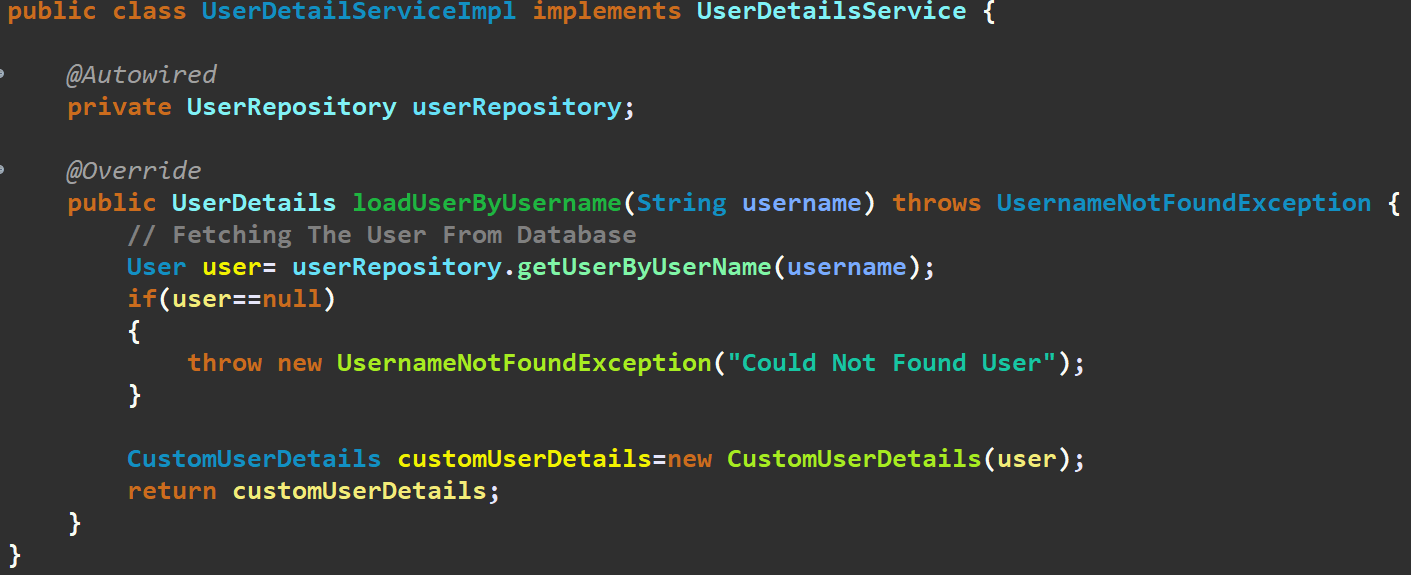
🡪And in the getAuthorites we have passed an list of simpleGrantedAuthority by help of the SimpleGrantedAuthority and in list we can also pass more things also

1. Than comes the second step in which we have to provide the implementation details of the user, And for it in the .config package we will make an new class having the name UserDetailServiceImpl instead of it we can also give any other name as per our wish
2. And init we have to implement interface named UserDetalsService, And than on clicking on the bulb as we have done in the upper case we have to add all the unimplemented methods of the interface, Example:



1. And now in this methods we have to get the user from the database and than return the user detail from this function and for fetching the user from the database we will take the help of the userRepository

🡪And than we have to return the userDetail, We will make an object of the CustomuserDetail and than we will pass the user as the argument in it, Example:

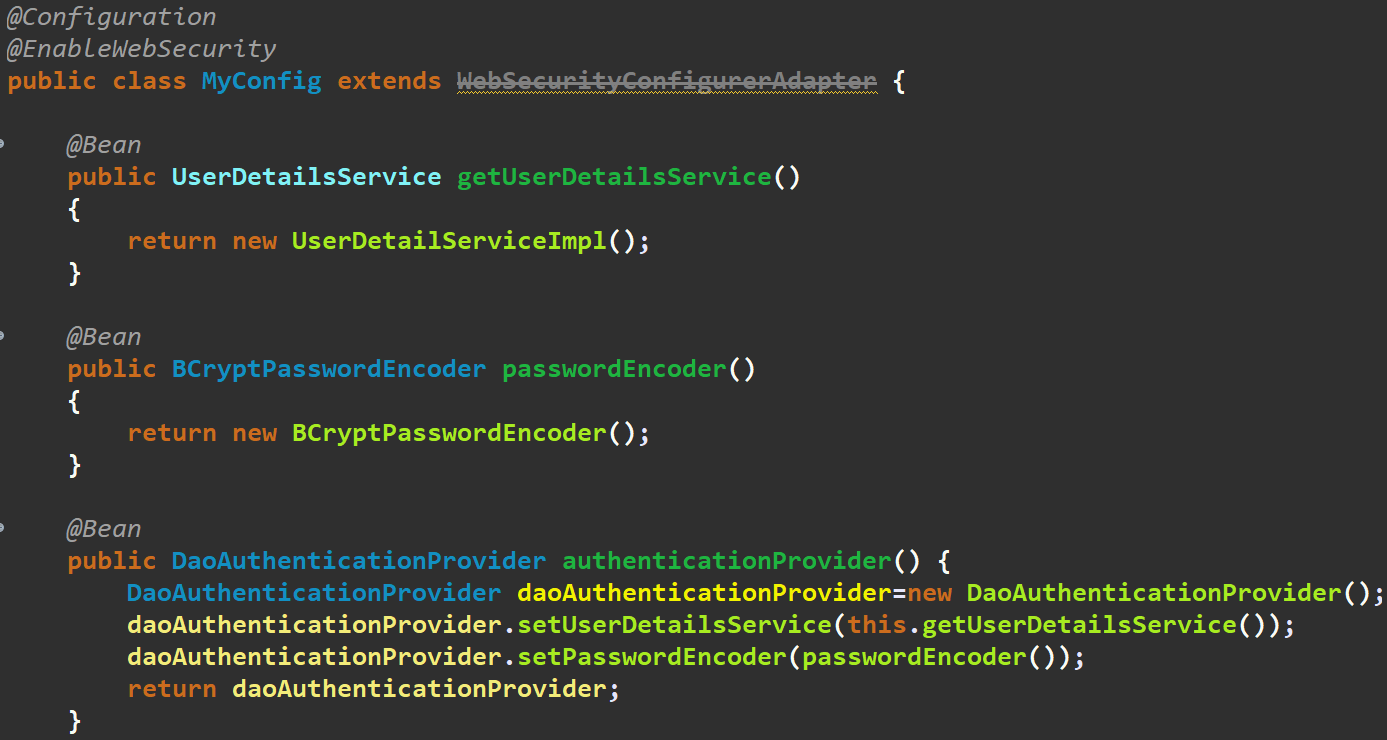


1. Now comes the third and the last step, That is to make an configuration class having the name MyConfig and in it we will put on the configuration annotation and than make it extends **WebSecurityConfigurerAdapter** and by help of it we will get the methods that we want

🡪Than with the @configuration we will also use the @EnableWebSecurity by which our security configuration will become an-able

🡪And before declaring the methods, we will declare some of the beans which are important

🡪And on that three beans we will write the @Bean annotation, And that three beans that are to be declared are:



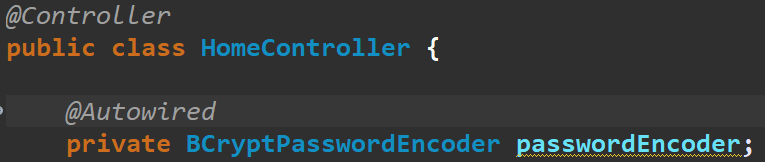
🡪And than we will write the configure method, As shown below, Which is our main method and for it we will go to the source🡪Override/Implement Method and first click on the configure(AuthenticationManagerBuilder) and than add some text to it as shown below and than we will also include the configure(Security) by same way from the source, Example is as shown below:



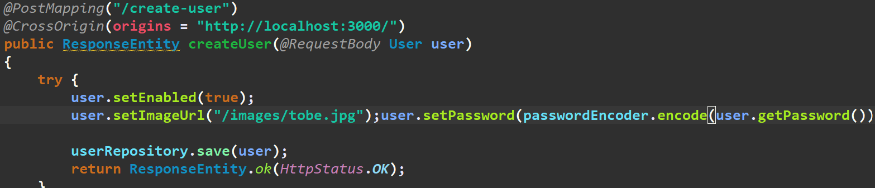
1. And after configuring the spring security, We will make the change for saving the password that we give while siging or registering, Such that it will save in the hashcode form:

🡪And for it we will go to the controller which handles the saving of the data in the databae, i.e. In this project HomeContoller

🡪And in it we will write the following statement on the top of the contoller class with the annotation of autowired, As shown below:

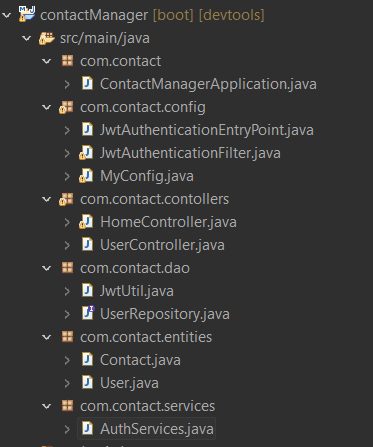


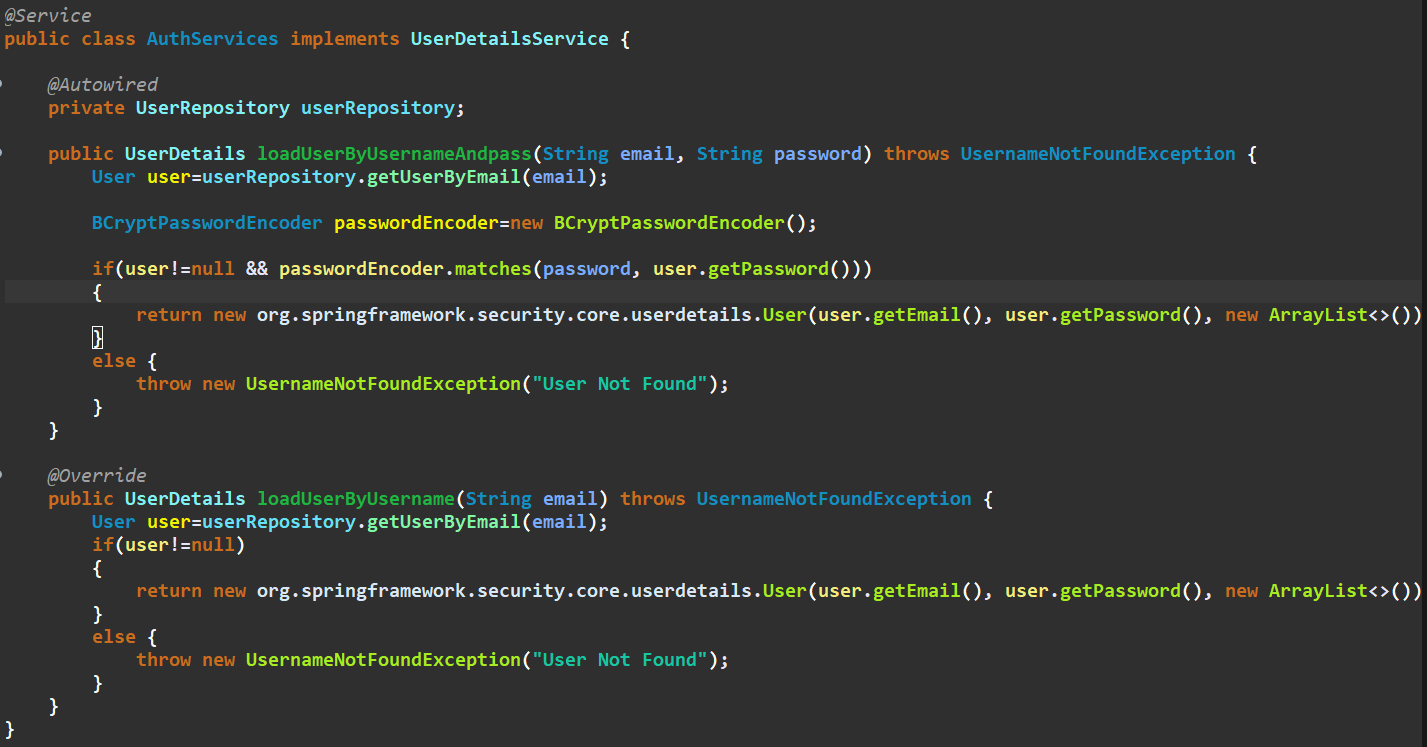
🡪 And than we will change the password that is been made saved in the database by adding the following statement in the block of code, The example of block of code before and after the implementation of password saving format in the hashcode is as shown below:



1. We have made some of the new classes and also make some changes in the existing classes, The some of them are as follows:

🡺Project Lineup

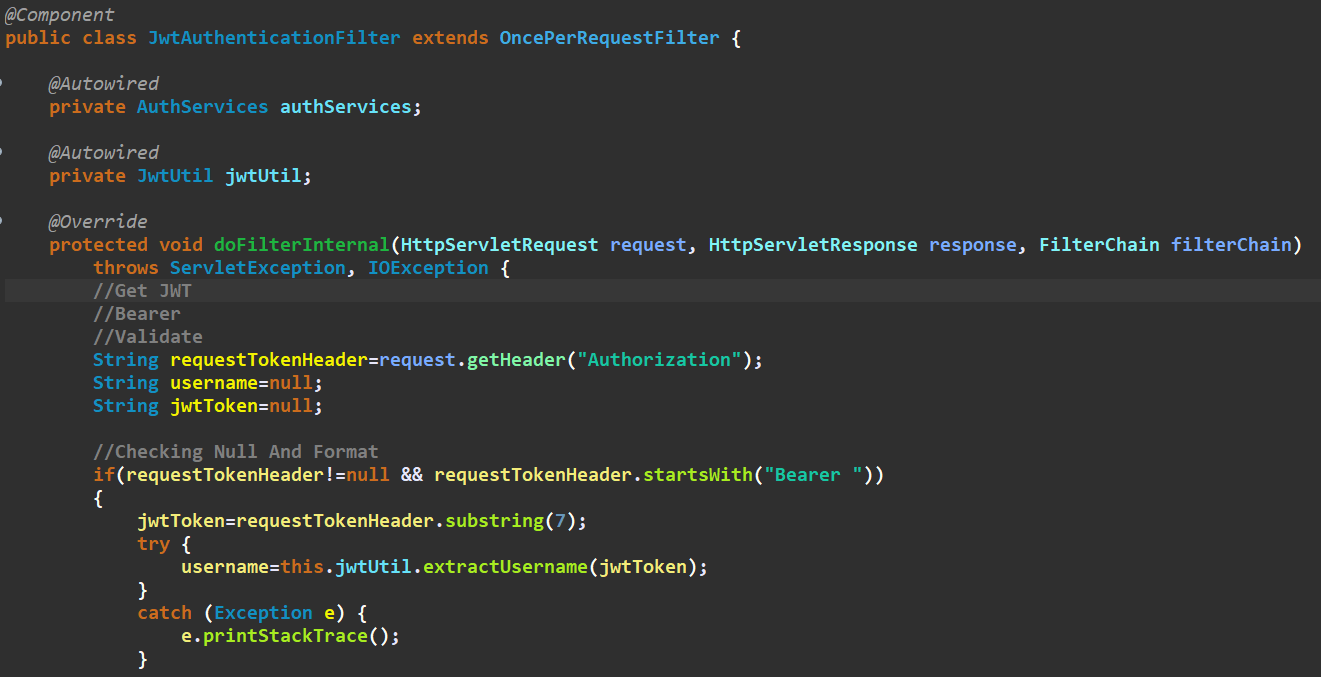
  
🡺AuthServices.java

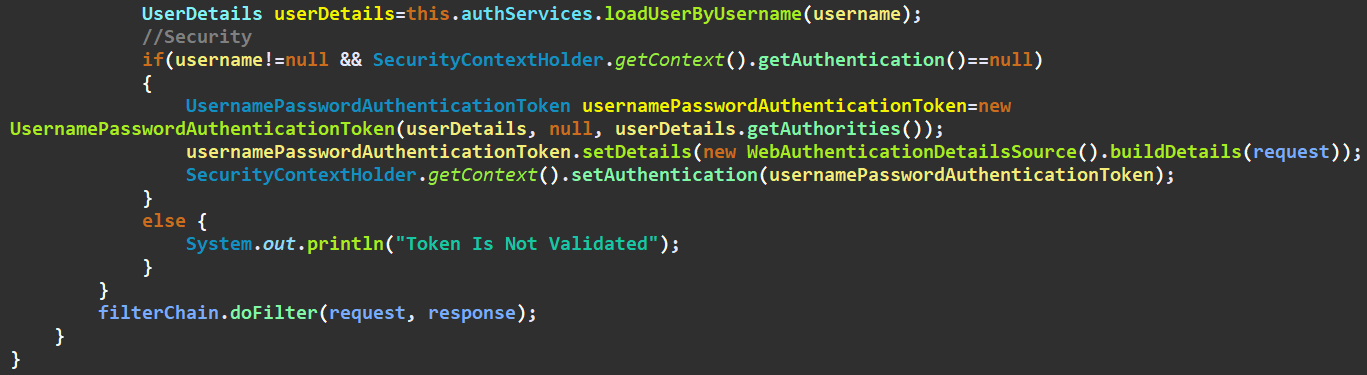


🡪For the encreption related work we have used the BCryptPasswordEncoder class and for mathing the password in the database and the password given from the user we have used the .mathces method

🡪First method is for login and the second for receving data after logined

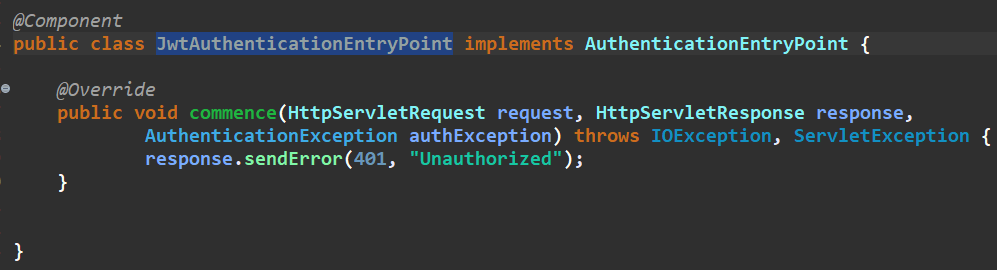
🡺JwtAuthenticationFilter.java





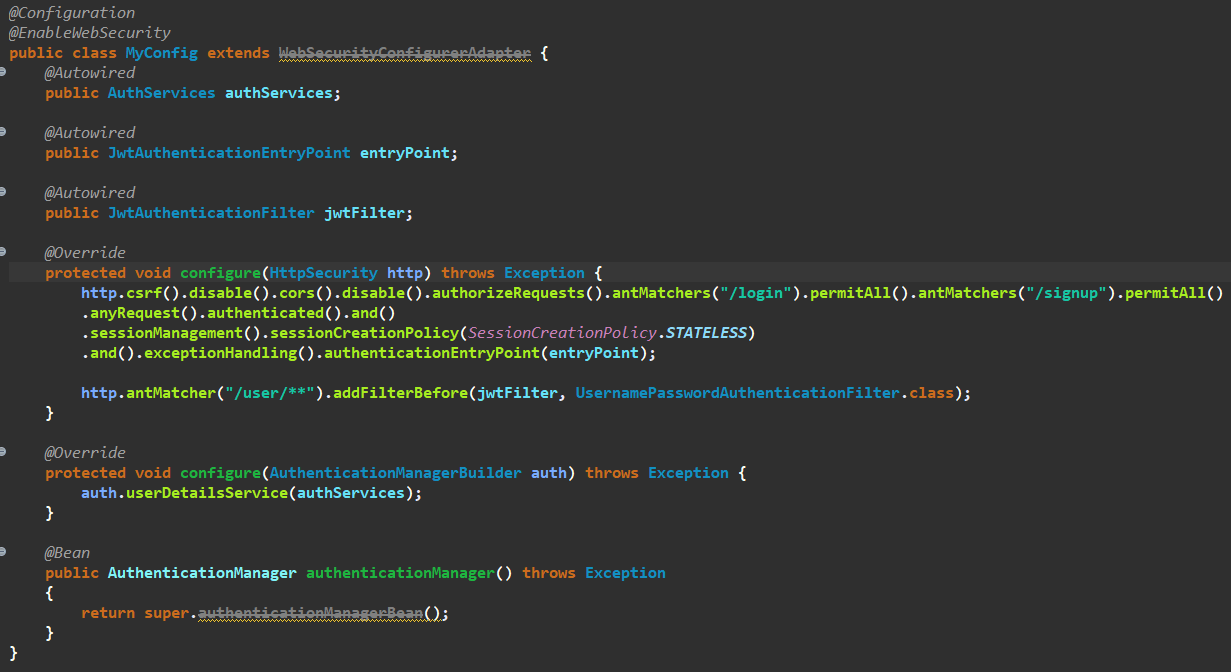
🡪As we have set in the MyConfig file such that if there is any url with the /uesr in it than it will make the render of this filter first and will check if the user is been authenticated or not, And if authentication is proved by above methods than it will diretcly show the required page

🡺JwtAuthenticationEntryPoint.java



🡪The class will be showed when there is authentication not done, And some thing is wrong

🡺MyConfig.java (Security Configuration)



🡪In this class there is all the configuration related to the security

🡺JwtUtil.java

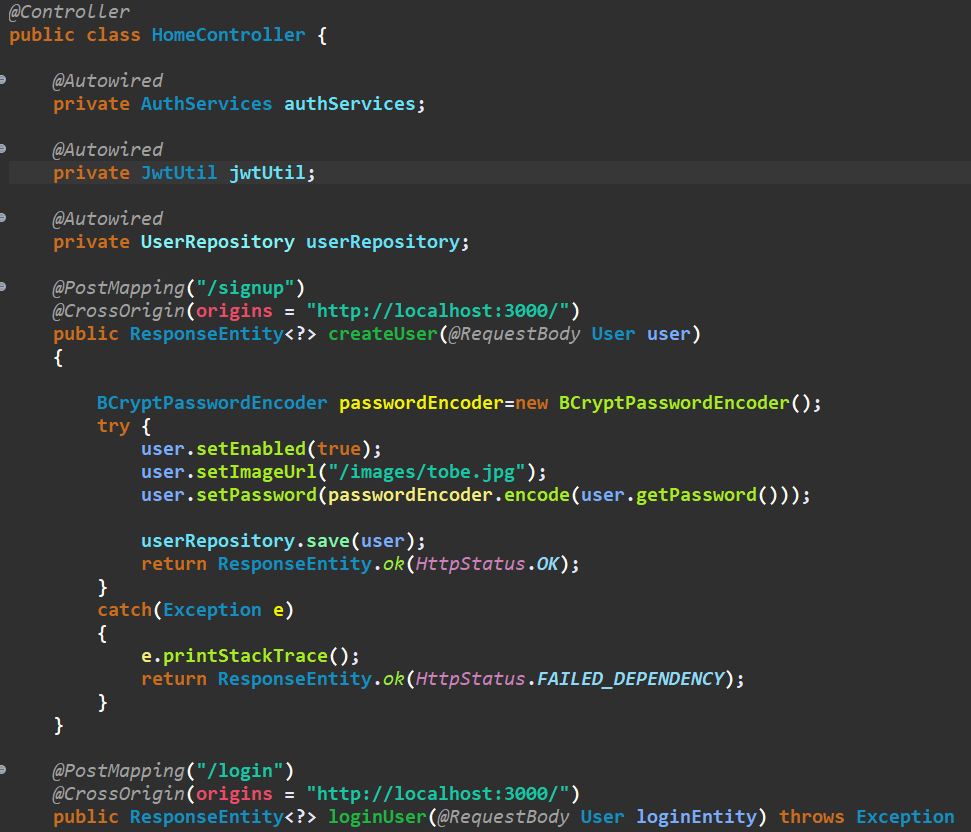


🡪This is the utility class which helps us to work with the jwt, Some of the changes are been done according to the requirement of the project, The original one is available on the github, And would easily get by writing the JwtUtil class on google

🡪The classes and etc can be changed, And also new methods can also be made as per project requirements

🡪In the claims which is an map we can add the new data with its name and value in the generate token method

🡺HomeController.java





🡺UserController

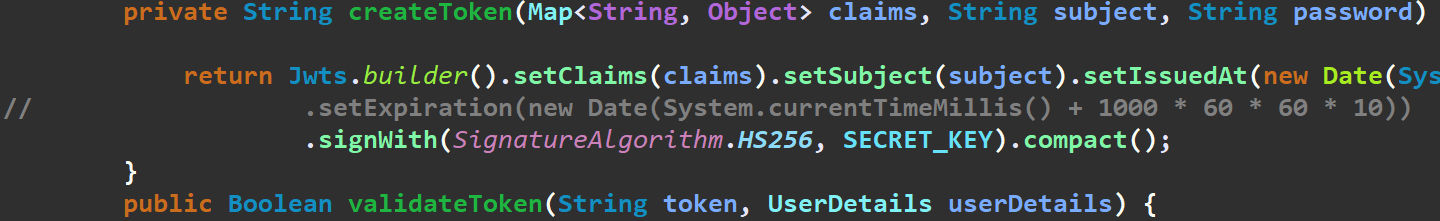


1. Now as there is cors error while accessing the secured url i.e. url with the /user in the link, Than we have to add the .cors() function to the configuration of that specific url, Example:



🡪Than only we can access it by other server

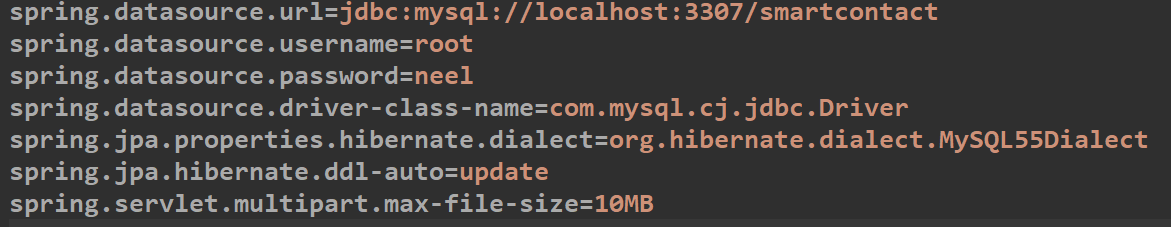
1. We can also make the time limit of the jwt token to infinite by making some change in the JwtUtil.js class, And in it we will make change in the createToken method:



1. Now we will make an controller for the adding of the new contacts, Which is as shown below:

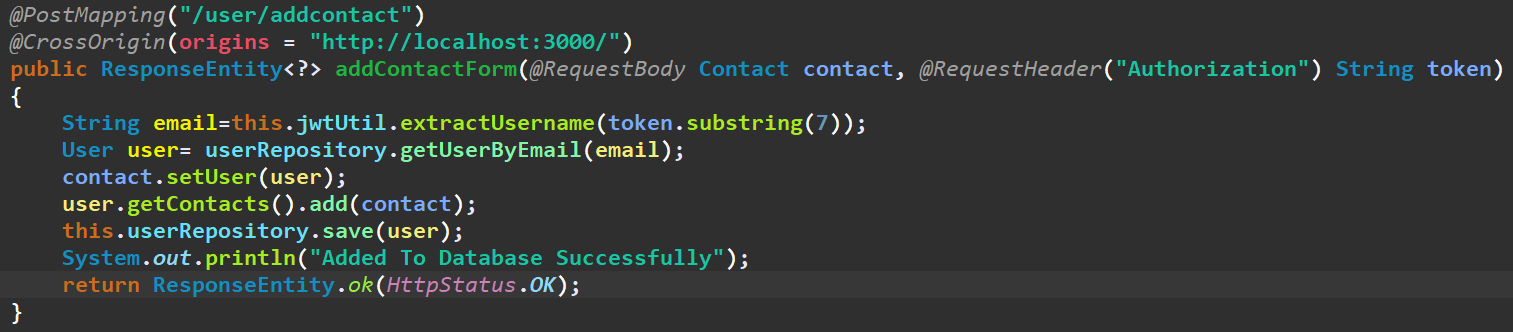
Edw

1. And to set the max file size of the uploading file we have to add the following statment in the application.properties



🡪If we want to set the max-file size to be infinite than we have to use the -1 instead of the any value

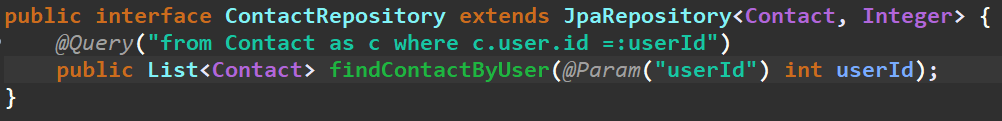
1. Now we will make an controller in the UserController file by name of the addContactForm and in it we will write the code for adding the contact to an user’s profile, Code of this controller is as shown below:



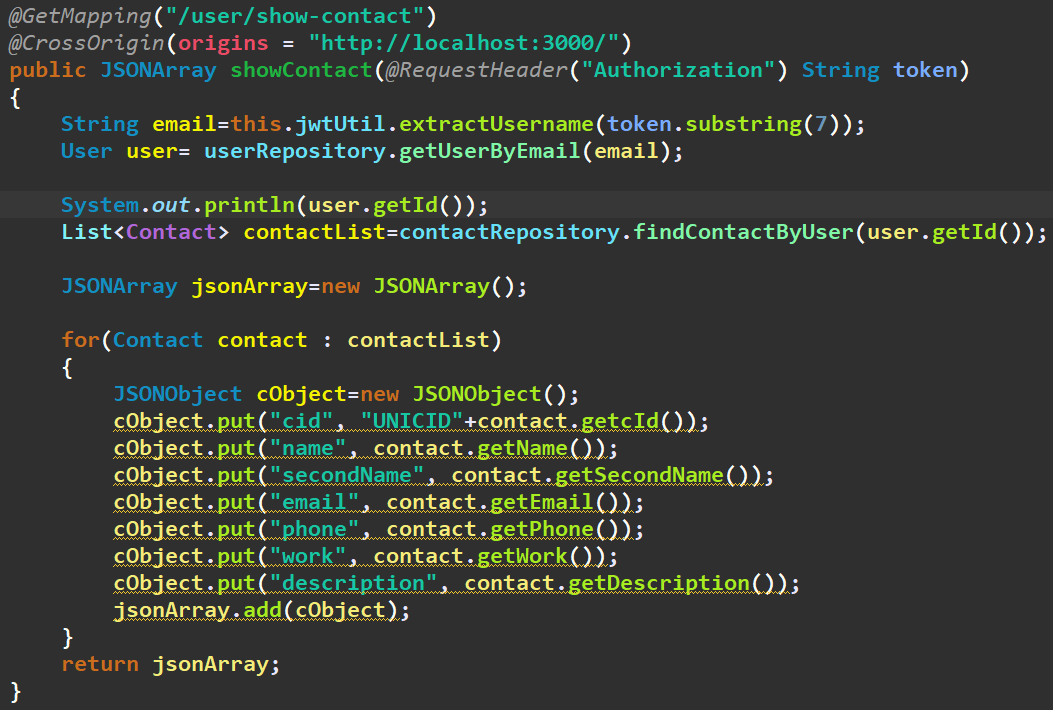
🡺Now lets see what is the logic behind the code:

* + - 1. First we will take the body and the header given in the request by help of the @RequestBadt and the @RequestHeader annotations and than will give the type of the input coming and at last we will assign the name to it
      2. Than by help of the extractUsername method we will extract the email id from the jwt
      3. And make sure you give the string after thr 7th index, As before it there would be text of bearer and that is to be not send tp the extractUsername method
      4. Than by help of the getUserByEmail method of the userRepository we will get the user containing the particular email from database
      5. Than from now comes some complex buy easy code
      6. As we have done the mapping of the ManyToOne from the contact to user, We have to also put the user in the contact’s entity, So to make it by help of the setUser mehtod of contact, We will add the user in the contact
      7. And than as we do not have any special method for updating the data, So we will add this contact to the user’s contacts that we got by email and getContacts method of the user, And than we will add the our new contact to that data by help of the add method
      8. And thus our work is done, And at last by help of the save method from the userRepository we will update the user that exist in the database
      9. **FACT**: If the user already exist with the same Primary Id and we try to save it another time than instead of making the new coloumn, It will update the existing one

1. Now before we work on the api of fetching the contacts of the user, We will first make an contactRepository, Example of it is as follow:



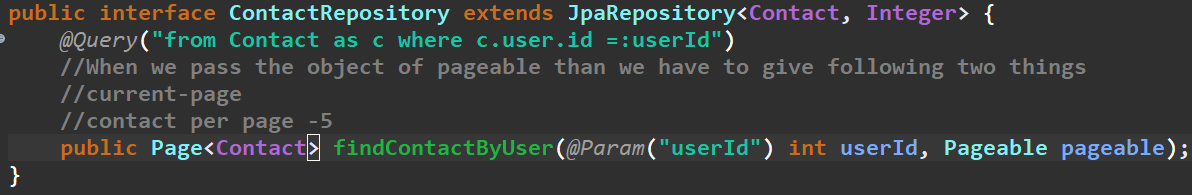
🡪And the controller is also as shown below:



1. Now we will work on the pagination system on the frontend, To apply that pagination system we have to also make some changes in the backend also
   1. In pagination we divide our data in the specific number of the data, By which we will make the load on the server to be less and the required and maintainable amount of data is showed on each page
   2. Example of the pagination is as shown: 
   3. Now to make the pagination to be done, we will have to first make the planning that how many page is to be shown, on each of the page
   4. And to make the things to be implemented, we will have to maintain the two variables the one to define the page on which we are, And the other is for the total pages of the data that can be made
   5. And to give the data as per requirment we will give the current page number in link of the viewcontacts
   6. And by help of the pathVariable annotation, we will get the current page number of the page, And will change the link if we want to change the data on the page, For now we will handle the change of the page by changing the url manually, And at the end of implementation we will make it to be done by help of the buttons in the frontened
   7. And for implementing this system on the backend we will take the help of the page interface

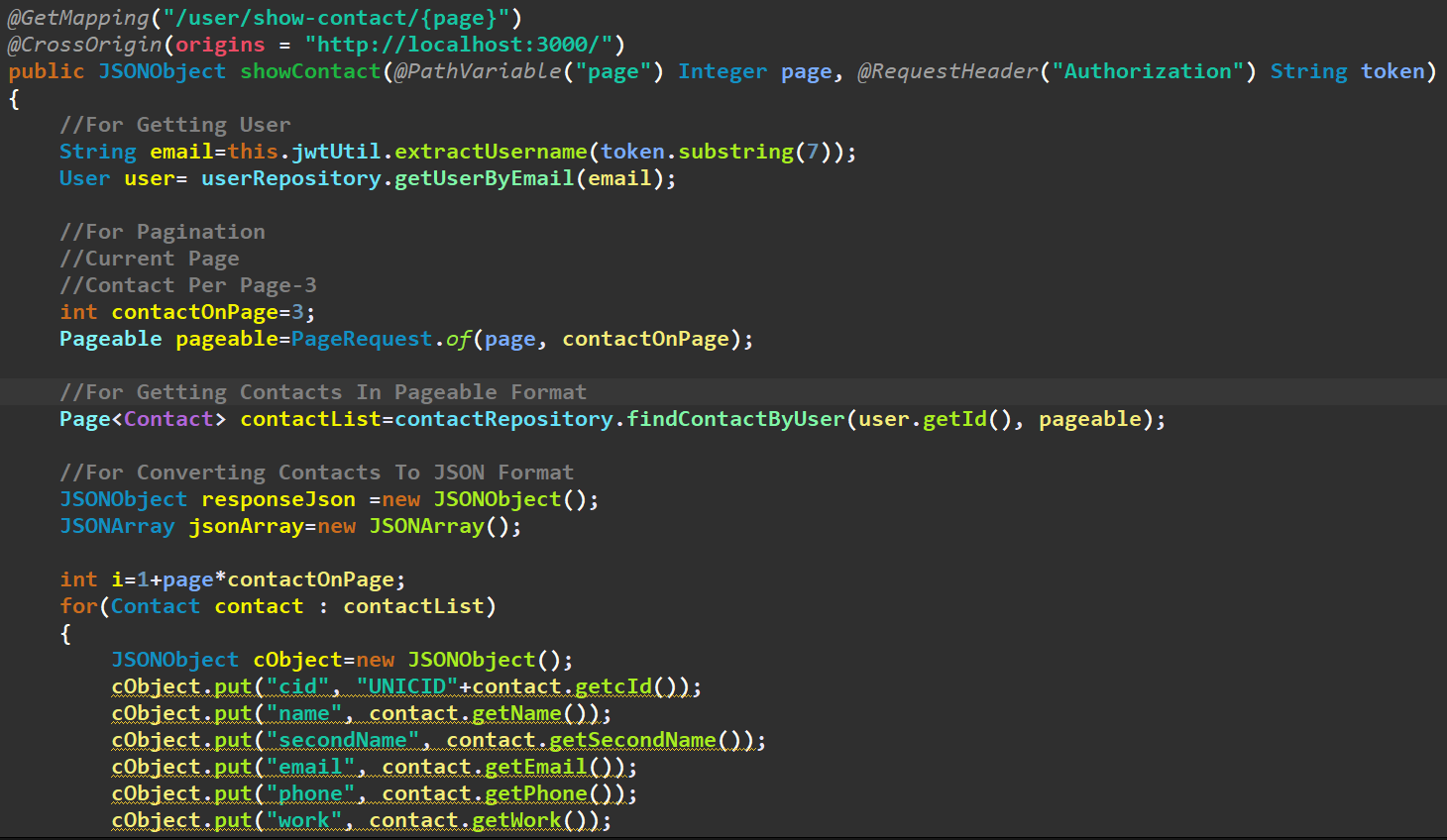
🡪**Page is sublist of the list of the object, it allows gain the information about the position of it in the containing entire list**

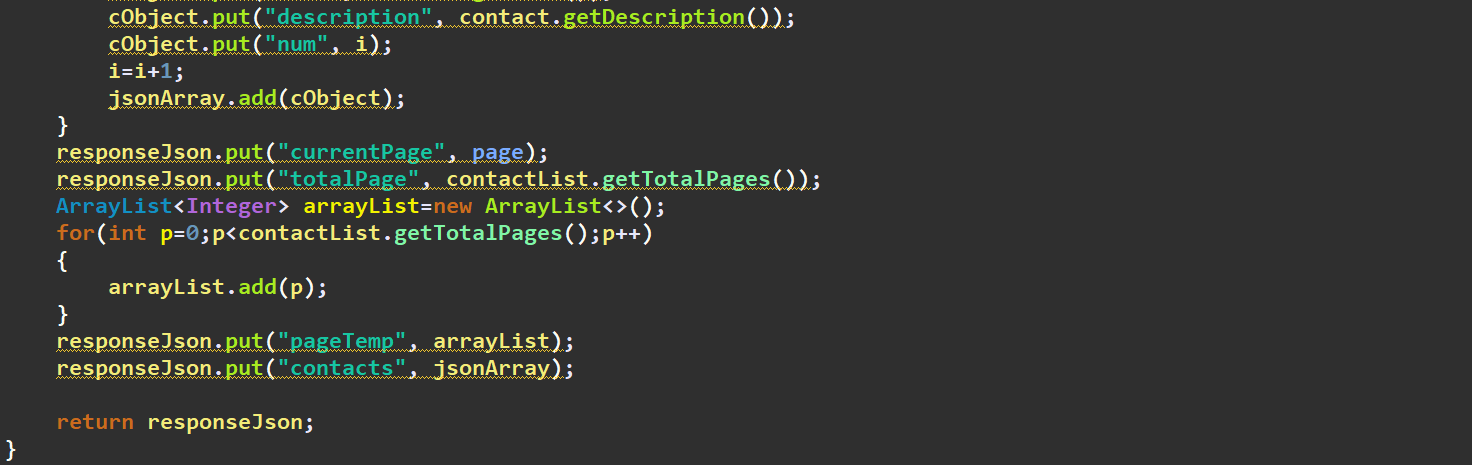
* 1. And to get the data acccoring to the page given in the url, We have to make the changes in the contactRepository, The changes made is as shown below:



🡪Make sure that you import the pagable from the org.springframework.data.domain.Pageable;

* 1. And the code of the UserController’s showContent function is as shown below, Which is responsible for the backend support of the data of contacts which is suportable for applying pagination, Code :





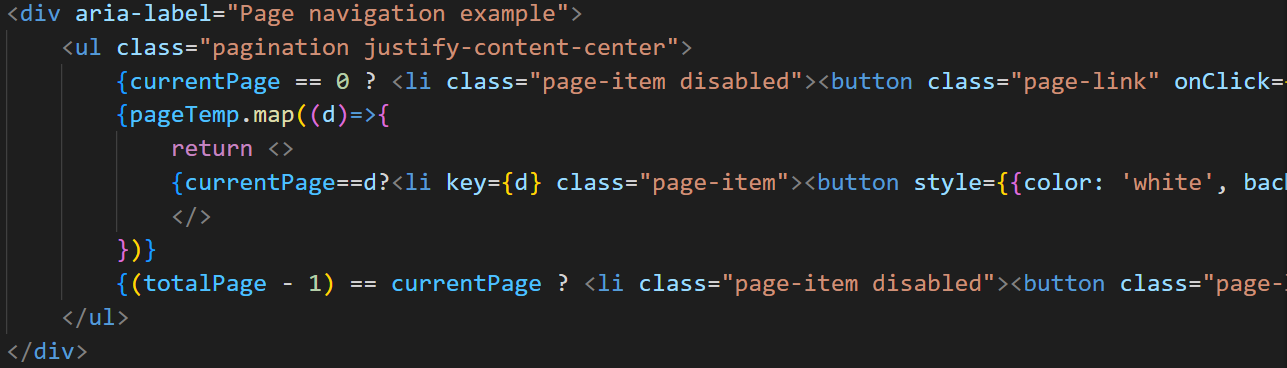
🡪We have to change the contactOnPage variable’s value to how much contact or data do we want to keep on the one page, And the whole system would be runned accordingly

🡪And the variable page used in the pageRequest method is the current page number, Of which data we want and its value is comming from path variable

🡪int i is used for giving the unique number to the each of the contact according to their page and for it an special code is also wriiten before starting of the for loop, Which calculates from where the number of the contacts starts for the current page

🡪There also exists an getTotalPages method for getting total number of the pages of the data

* 1. The pagination system implemented in the **frontend** is as shown, But the whole code is very huge some of it is shown here, It is recommended to view this file whose name is ViewContacts.js in the userModule/components/src folder, Example code of it is as shown below(Not-whole):



🡪Also refer the different methods made for the different button functions

🡪Directly pagination in the bootstrap is not found, Search in google than only it is found

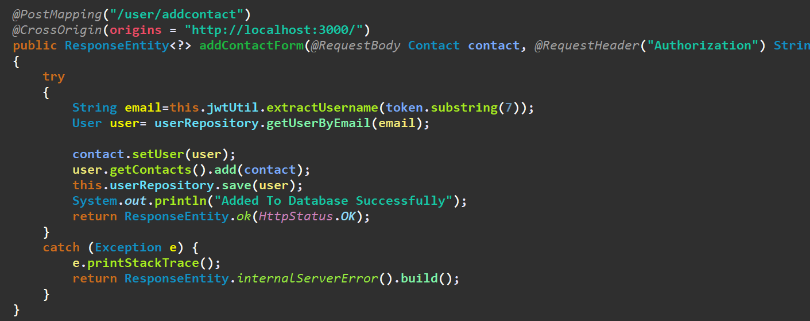
1. Now the code for the getting the user from the userId, Is as followed:



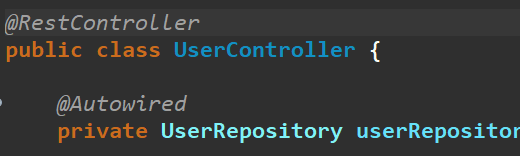
1. Now we can also make the some configuration things to be stuffed good in the code by writing the common necessary url for using the methods of any functions, And giving the cors information on the starting of the controller instead of the starting of the each of the method(If all methods have same cors server and url)

🡪For example before doing the cleaning of the code the code looks like:

* 1. Of Any Function/Method:



* 1. Of Controller:

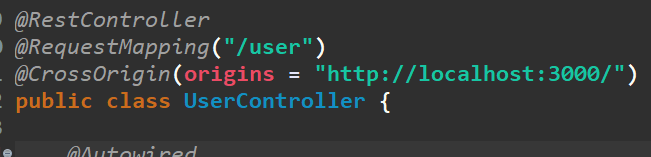


🡪After cleaning and doing some advancment code looks like:

1. Of any Function/Method:



1. Of controller:



🡪Hence, by doing this the extra common url from the each of the function’s mapping is removed, and the same annotations from each of the methods is removed, And common things are directly written on the top of the class only, **By doing this the readibility of the code increases**

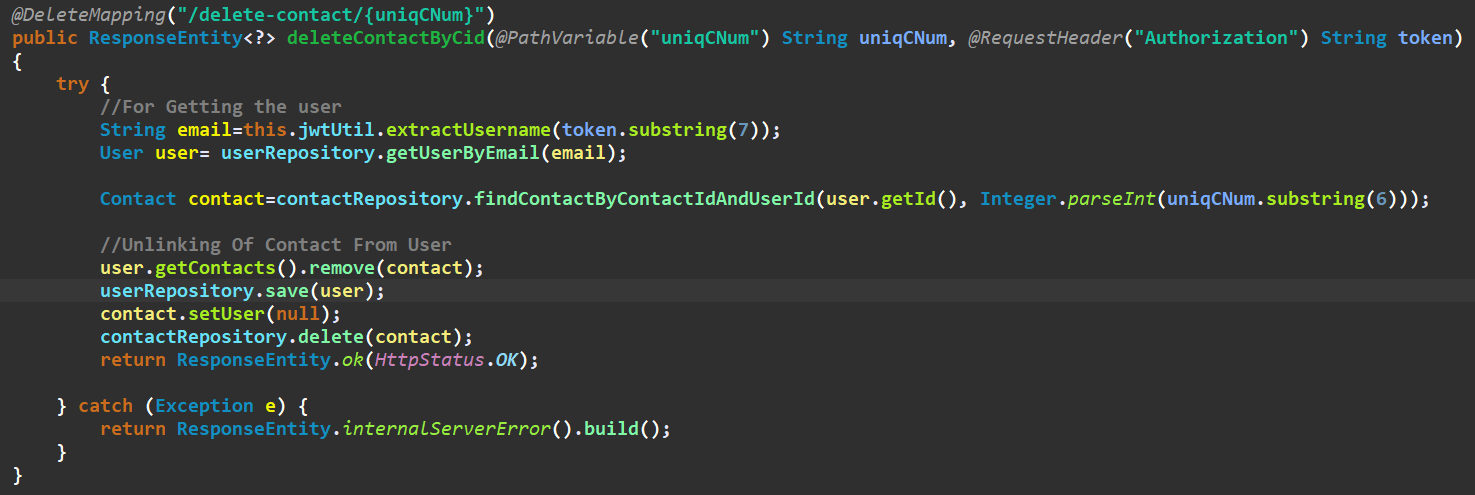
1. (I.M.P.) Now we will make an method by which we can delete the specific contact from the contacts by help of id of the contact, Code is as shown below:

🡪Now while deleting the contact, There will a error, From the databse and we will be unable to delete the contact that we have selected

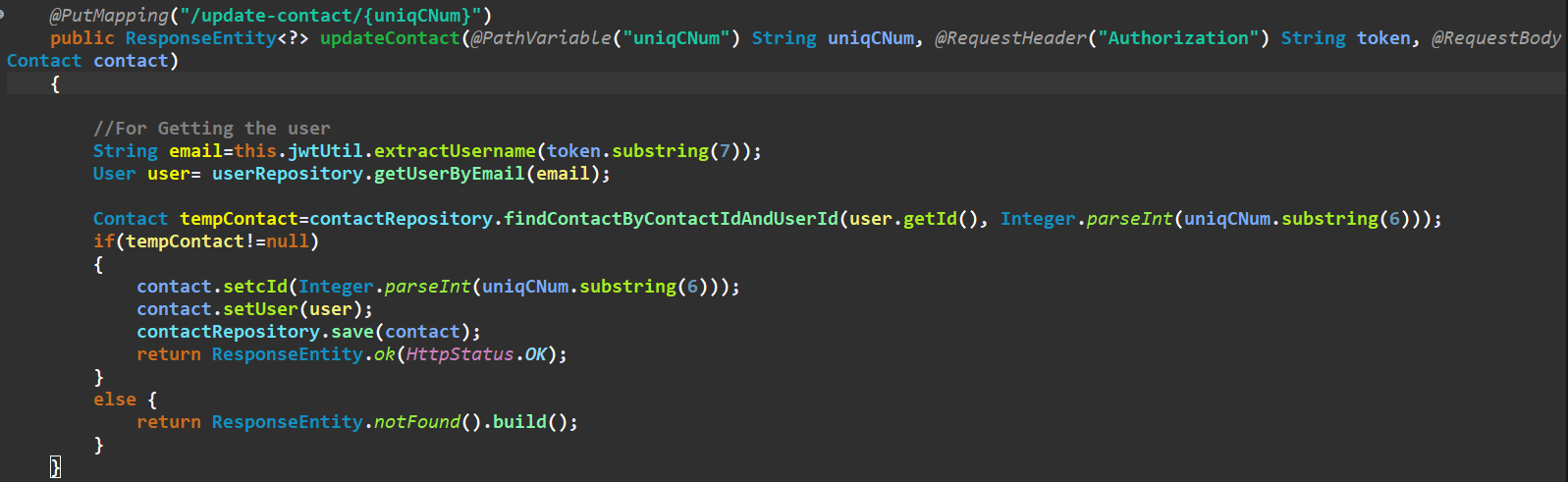
**🡪The error would be coming as we have done the mapping of the user and the contact from both sides, Now to delete this contact we have to un-link/un-mapp this mapping from both the side or entity, And after that only we can delete the contact**

**🡪And for unlinking the the contact from the user we will take the help of the getContacts and remove function, And for unlinking user from contact we will use getContacts and remove function, And after that only we can make the contact to be deleted**

🡪The code of this process and the method for deletion is as shown below:

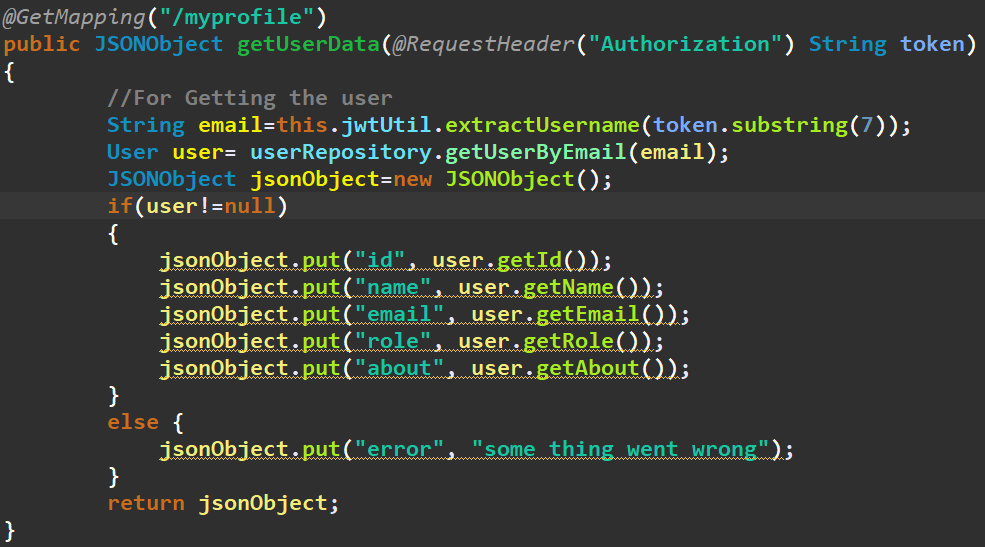


1. Code used for updation of contact used in the controller is as shown below:



🡪As there does not exists any method for the updation, So that we can use the save method for the updation, But make sure the primaryid of the new object is same as the primary id already stored in the database, Than only the updation will take place

1. Code used for fetching my profile page’s data in controller is as shown below:



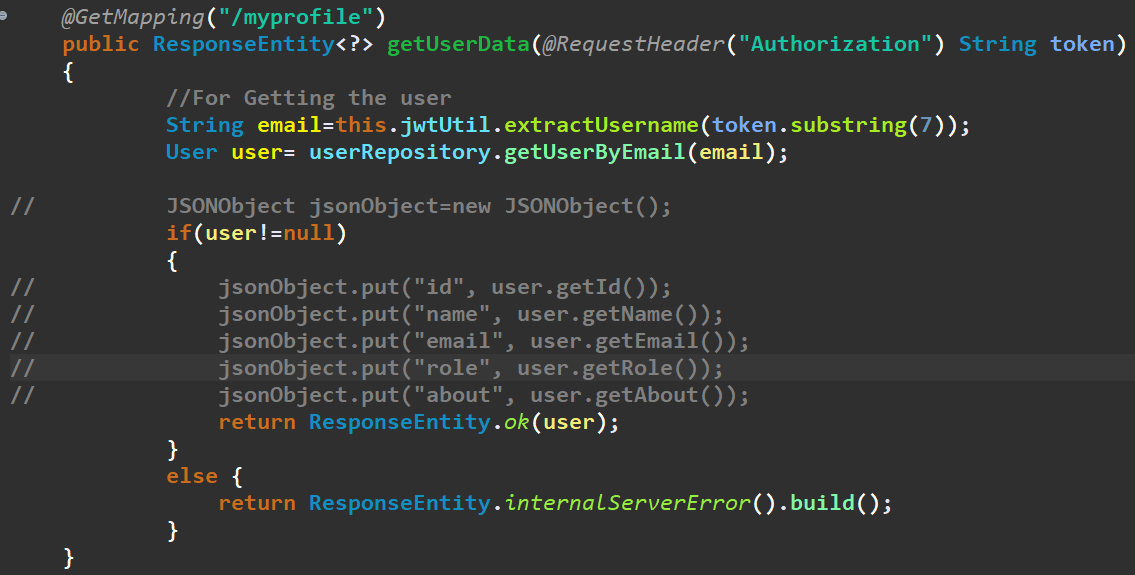
1. Instead of **converting the data to the json by help** of the jsonObject method, we can directly change the given data in to the object by directly sending the data in the http response

🡪Example of the code that converts the object to json manually and the other method that does it automaticaaly is as shown in the figure:

* 1. Manual One:

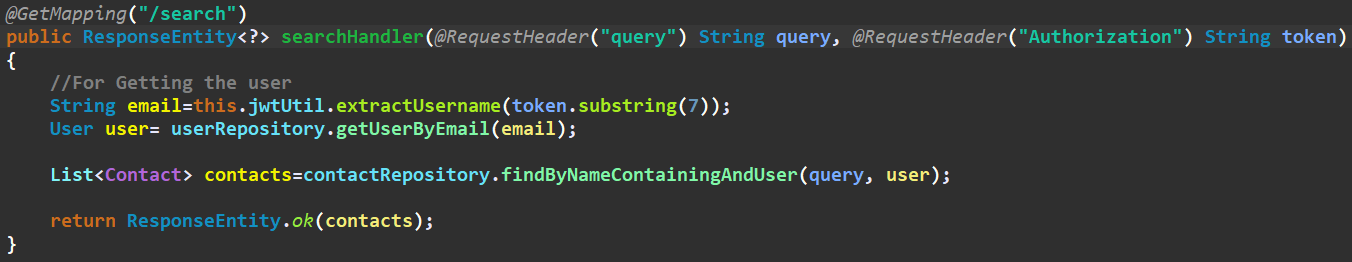


* 1. Direct One:



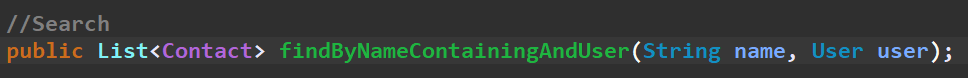
**🡪Till now this method was not done, But this is the effective one user it only, But when you want to add the other details to the json than you have to use the primary method only….**

1. Now we will make an controller for the **search functionality**, The code for it is as shown below:



🡪Here we have used the method of directly sending the json file, As told in the above method

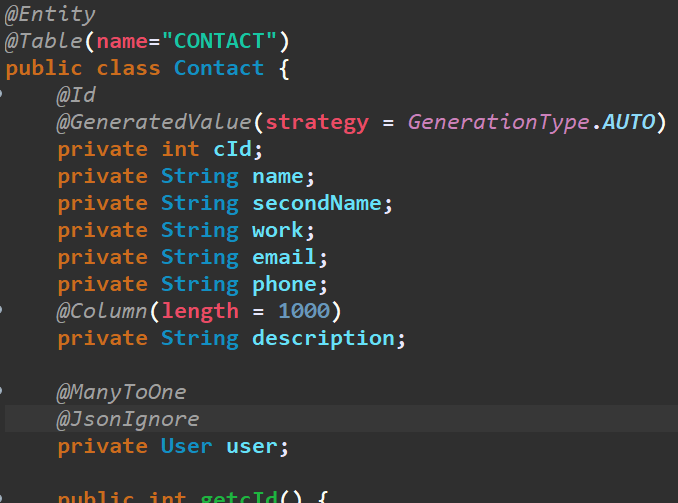
🡺And the method that we have made in the contactRepository is as shown below:



\*If **@JsonIgnore** annotation is not used in both there would be also problem in the searching of the repository method

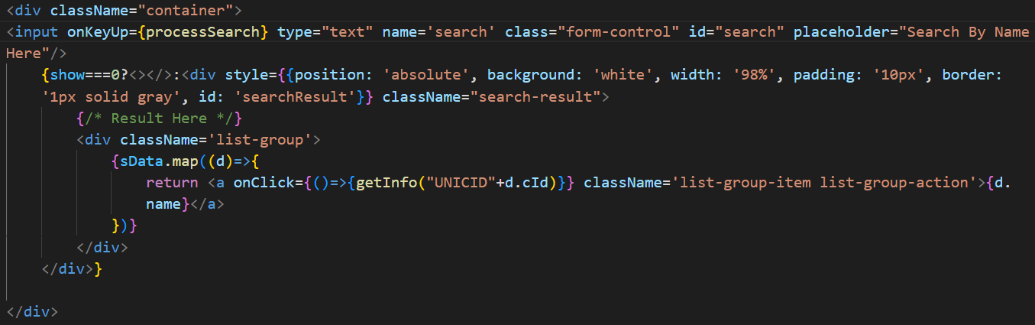
🡺The main change that we have made is in the entities(i.e. in the user and the contact), In which we know that we have done the mapping, So that when we convert the entity to json than the round cycle will be formed, To overcome this problem we have used the **@JsonIgnore annotation** as shown below on the user and contact(i.e. on vice-versa), Example is as shown below of both:





🡺Requirment from the frontend side:

* 1. An input field, And the a link method, As shown below:



* 1. Js code such that when the one’s key is made up the search function is made to be activated and the the data is to be made set
  2. And the method used for the getting the data is also as shown below:

