**Spring Boot Application and**

**Webpage wtih Javascript and Bootstrap**

*I built an application to perform bookings.Then, I built a simple website to interact with the backend application. I used Intellij IDE to develop the application. Also, I used Postman to test the application.*

**1.** Spring boot was developed using the following dependencies:

a. spring-boot-starter-data-jpa: to interact with the database, and store and retrieve data in/from a relational database quickly and easily.

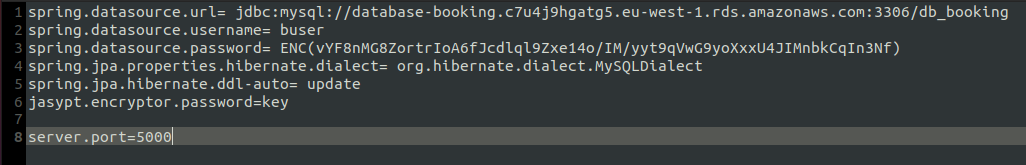
b. spring-boot-starter-web: to build a RESTful application using Spring MVC.

c. mysql-connector-j: to connect to a Mysql database (driver).

d. jasypt-spring-boot-starter (and jasypt-maven-plugin): to encrypt passwords in application.properties file. For example, to encrypt a string we have to wrap the value inside DEC() and then run the following command:

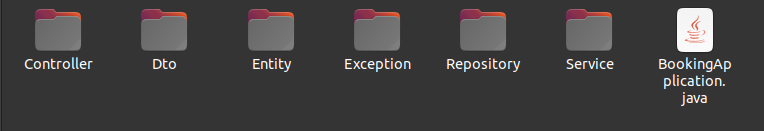
*mvn -X jasypt:encrypt -Djasypt.encryptor.password=key -Djasypt.plugin.path="file:src/main/resources/application.properties"*

**2.** Application.properties file:



Port equal to 5000 because nginx is listening on 5000. This application was deployed on AWS as Elastic Beanstalk. Then, we have the url to the database, user to access to it and its password encrypted using jasypt.

**3.** The application was divided in packages to organize it logically.

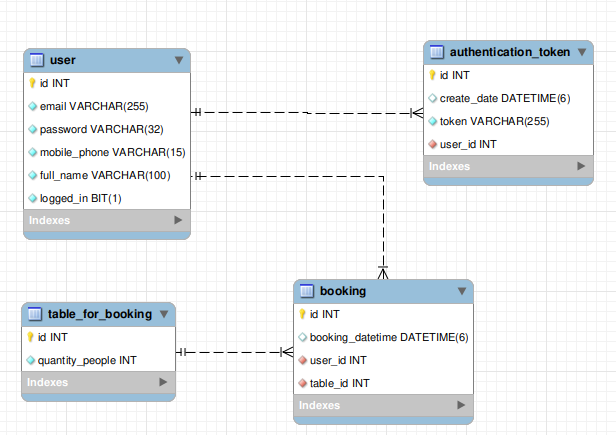


**4.** BookingController.java with two methods.

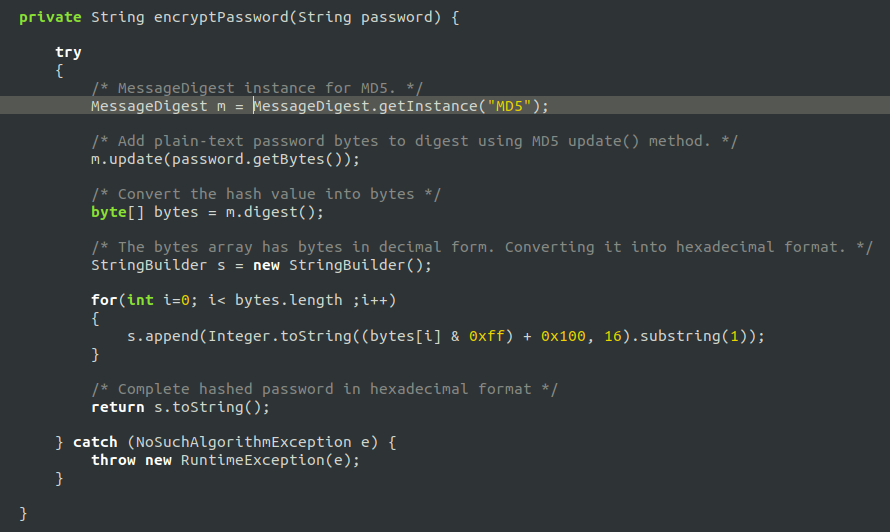


@RestController annotation is used to created RESTful web services using Spring MVC.

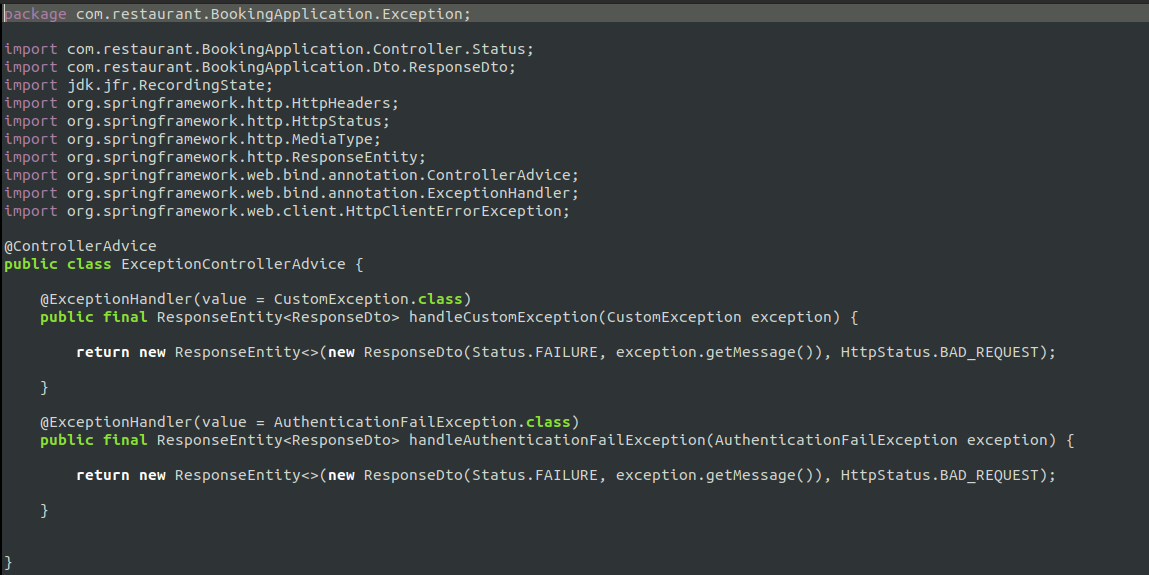
**5.** Simple database was created on Mysql Workbench. Below we can see the technical ERD on Mysql Workbench.



**6.** I used MD5 to generate the password.



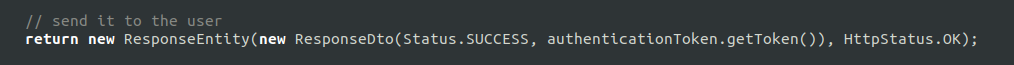
**7.** I used the annotation @ControllerAdvice to handle exceptions across the whole application in one global handling class. It can be viewed as an interceptor of exceptions thrown by methods annotated with @RequestMapping and similar.



**8.** A token is generated everytime a user is registered. Then, when the user is logged in to the application, the server sends the token to the user. The server, in each request, extracts the token from the incoming request. With the token, the server looks up the user details to perform authentication and authorization:

a. If the token is valid, the server accepts the request.

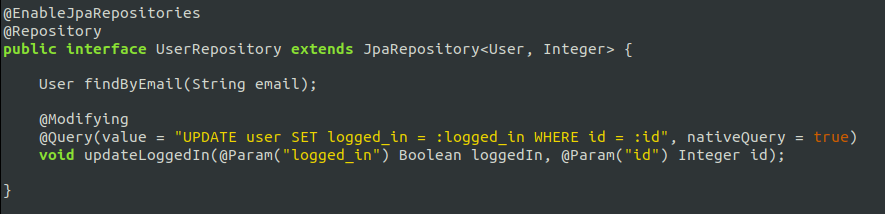
b. If the token is invalid, the server refuses the request.



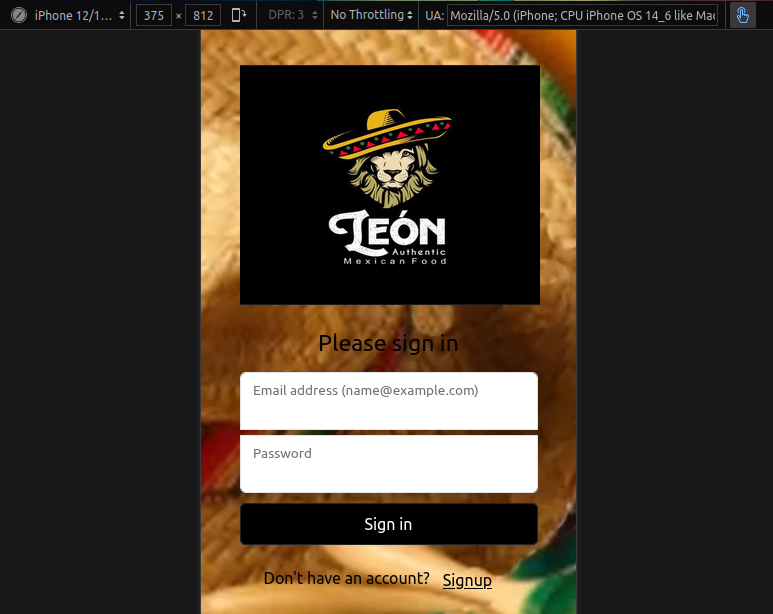
**9.** In logout method, we use the @transactional annotation to wrap a method in a database transaction.

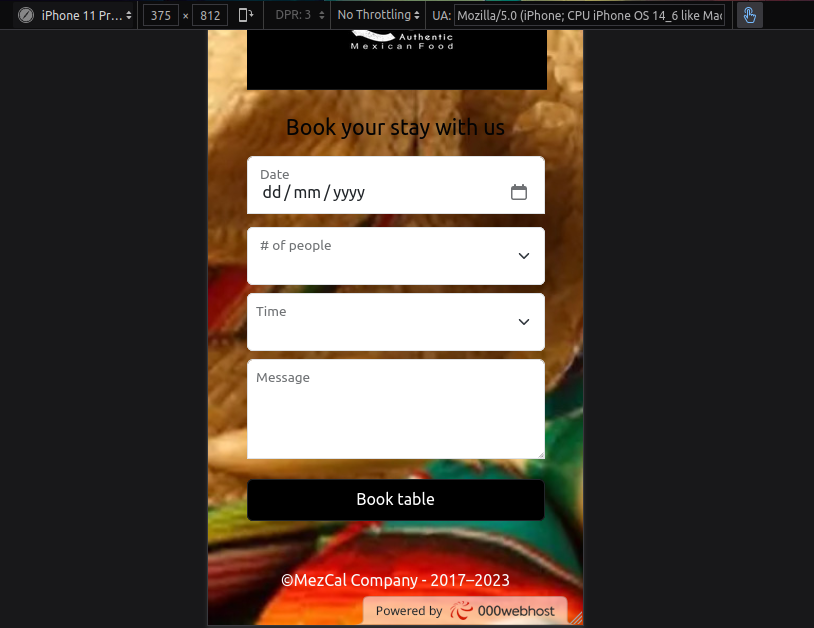


**10.** UserRepository class which shows how to query the database.



**11.** Screenshots of the website on a iPhone 12/13 mini

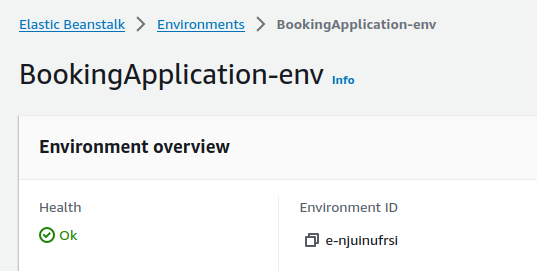
****

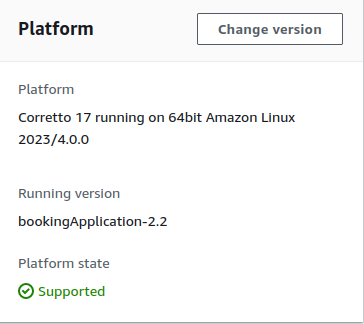


**12.** IDeployed the spring boot application and the website on different servers.

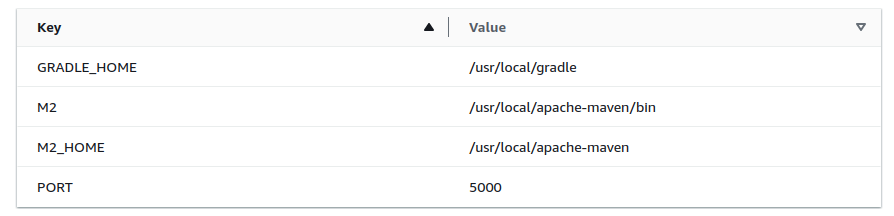
Spring boot deployment:

1. First I created an Elastic Beanstalk. I uploaded and deplyed the jar file which is generated on the terminal with the command mvn clean install.

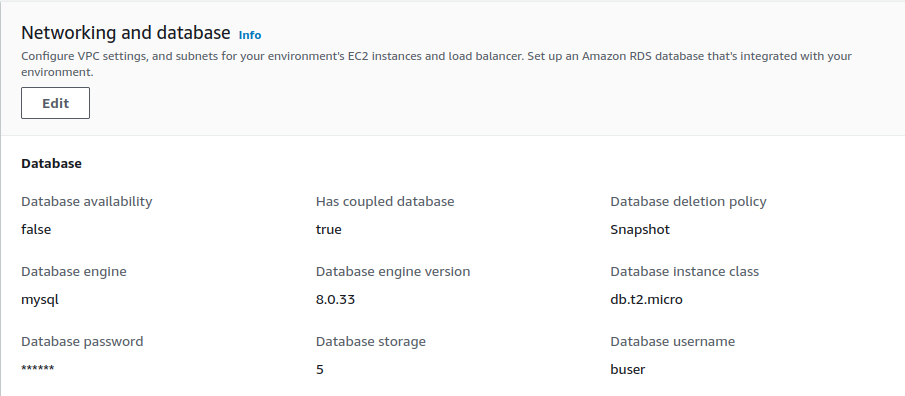




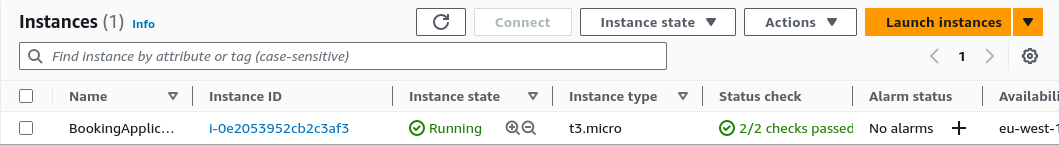
We can see the port 5000 as a environment property.



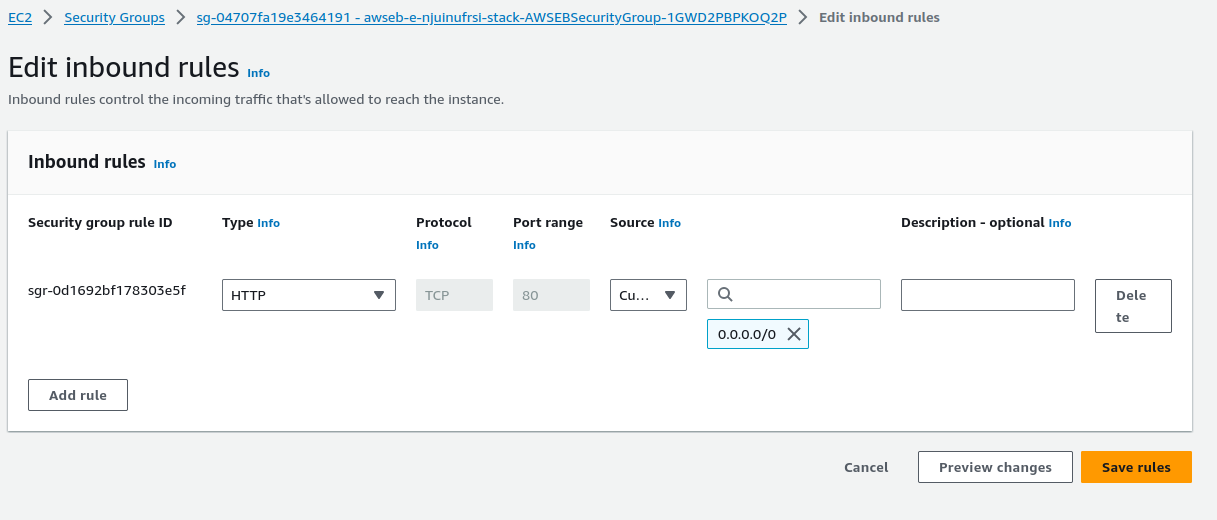
I defined a database (I am going to create the database later on this document):



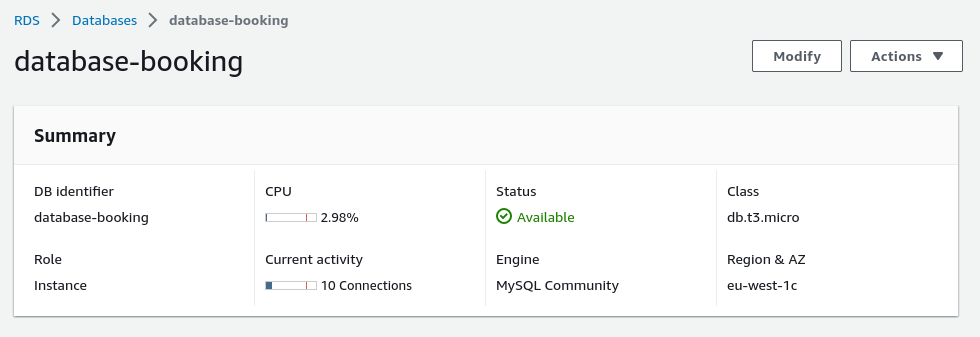
An instance is created automatically.

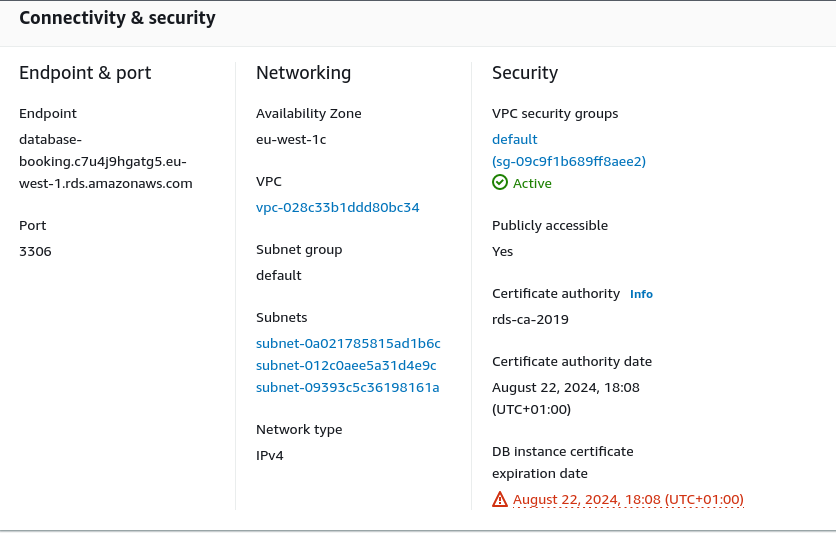


Security Group with the following inbound rule:



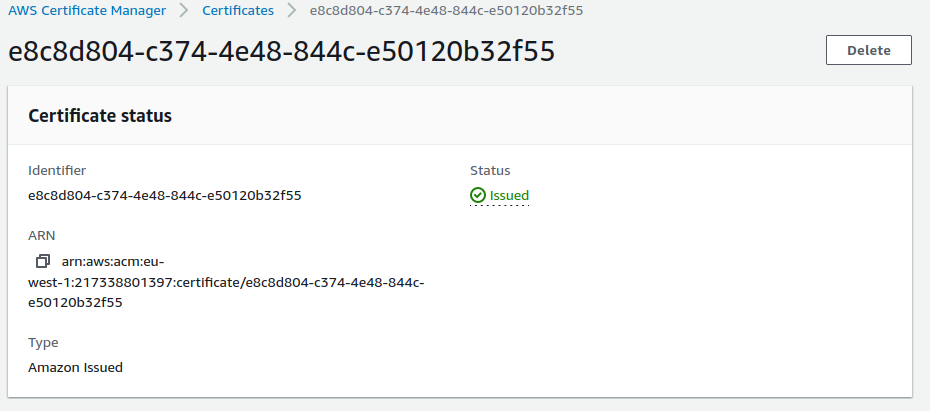
I created the following database:

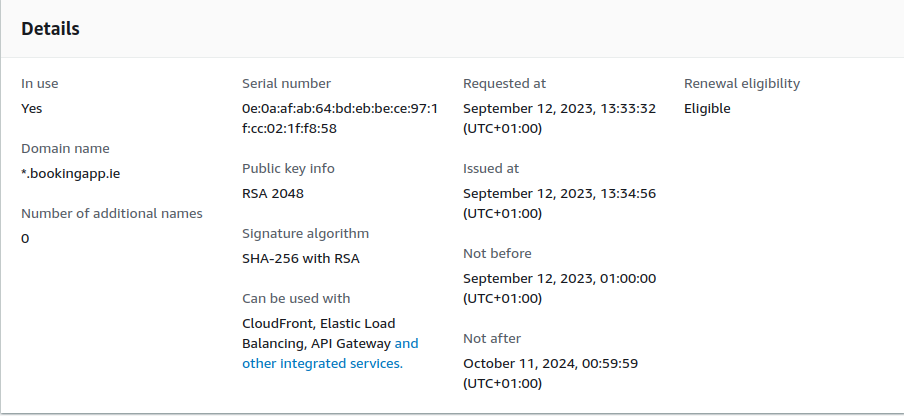




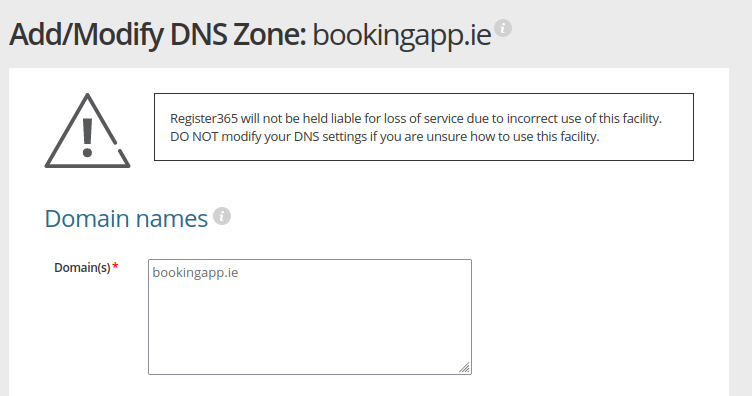
I can access to this database using Workbench Mysql.

Then, I created a certificate to enable HTTPS.

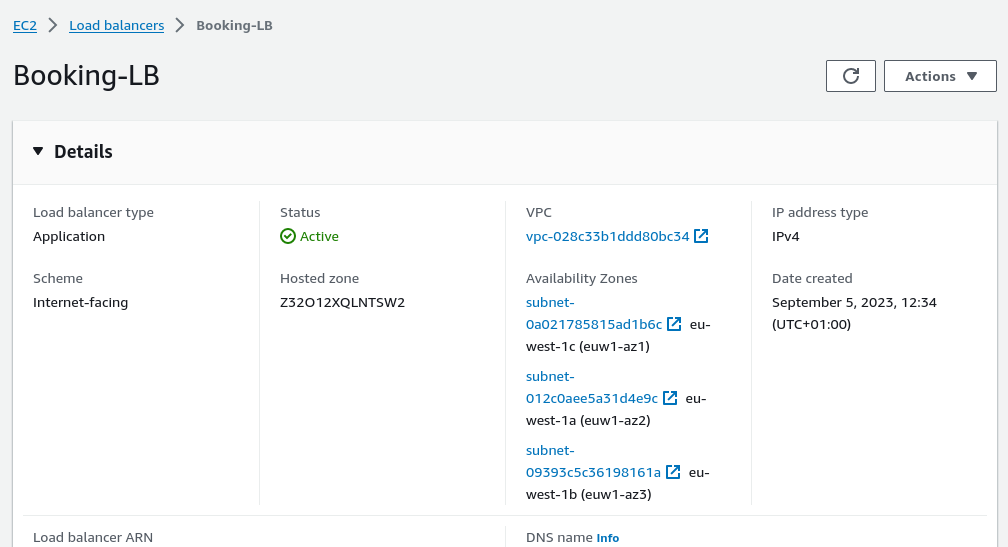




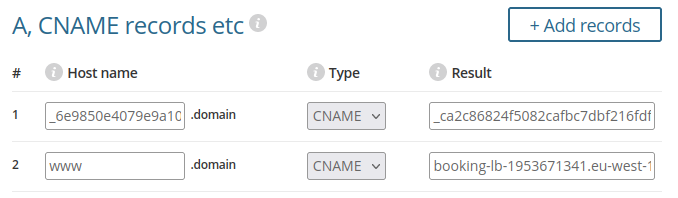
I have to create a CNAME record in the DNS Settings in the control panel where the domain was created. In this case, I created a domain on register365.com. The domain is www.bookingapp.ie.

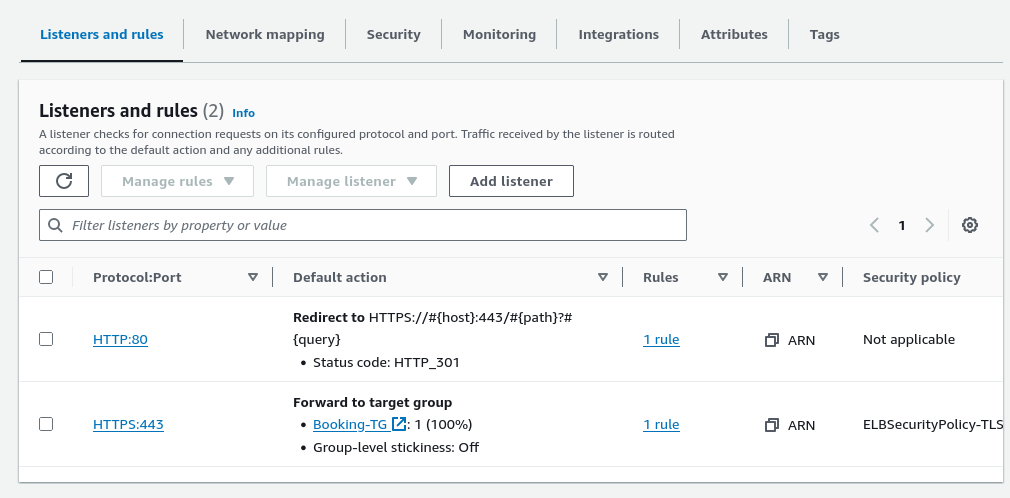


Also, I created a load balancer



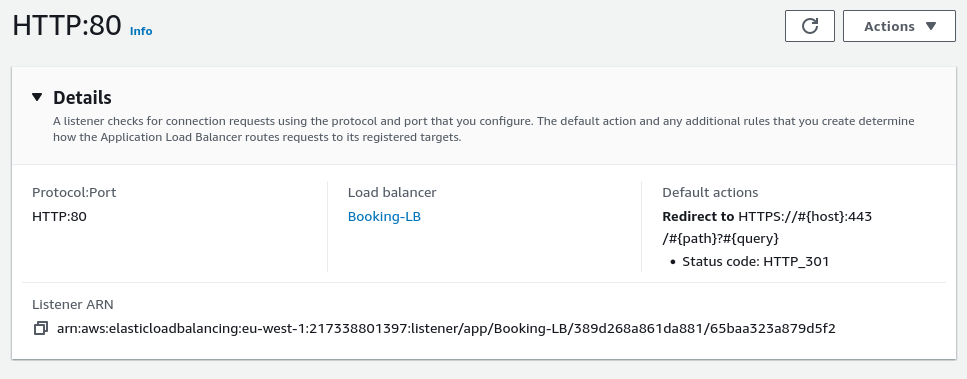
I had to create a CNAME record in the DNS settings in register365.com and add the DNS name of this load balancer.





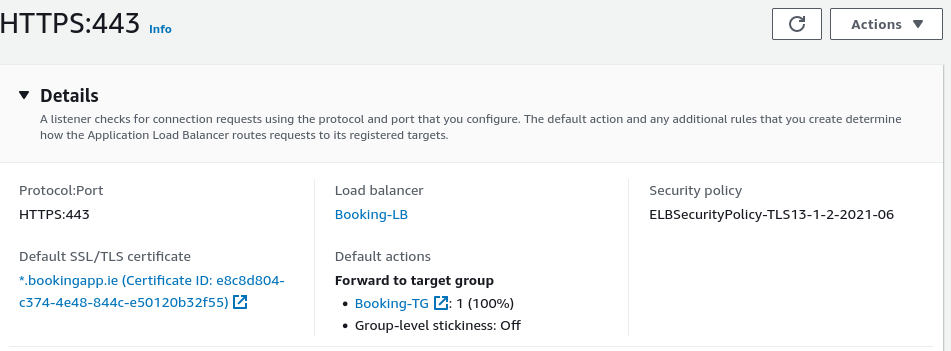
HTTP:80 Rule

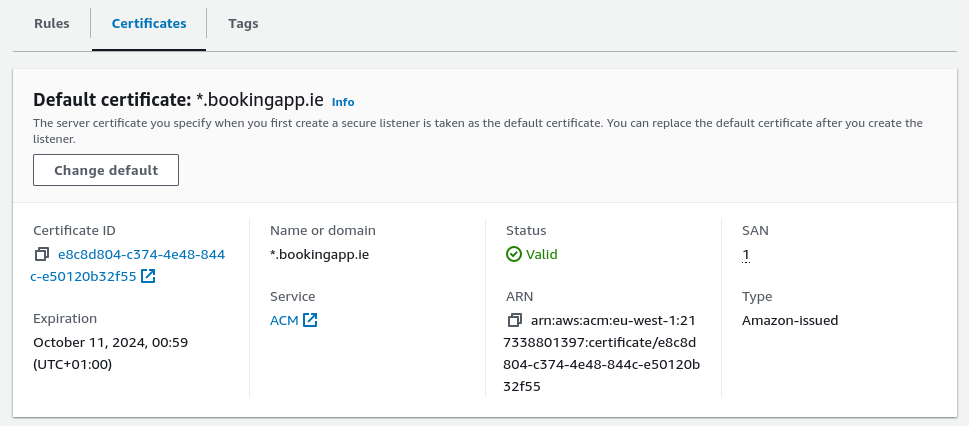
I created a rule with an action to redirect to HTTPS.



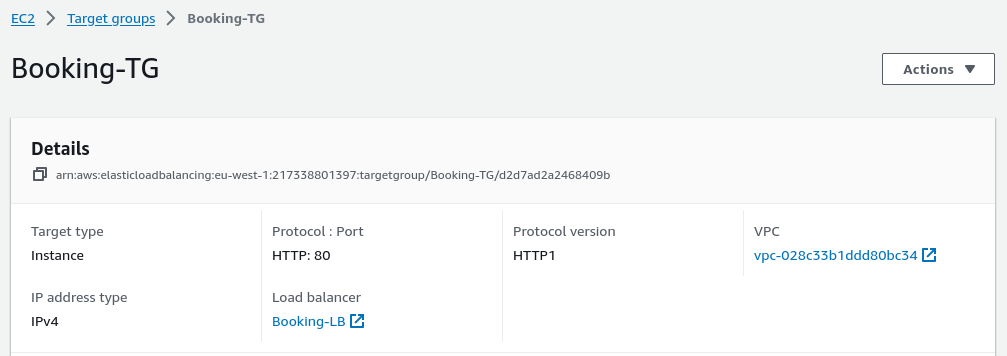
HTTPS:443

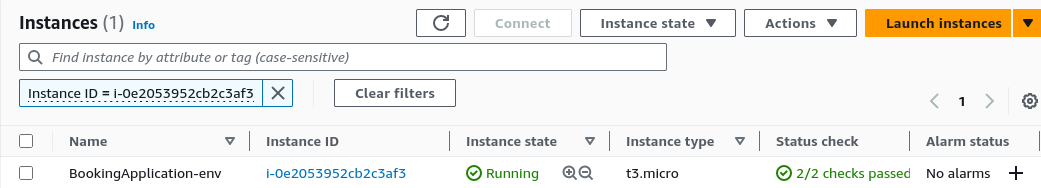
Pointing to a Target group and also a SSL/TSL certificate was assigned.



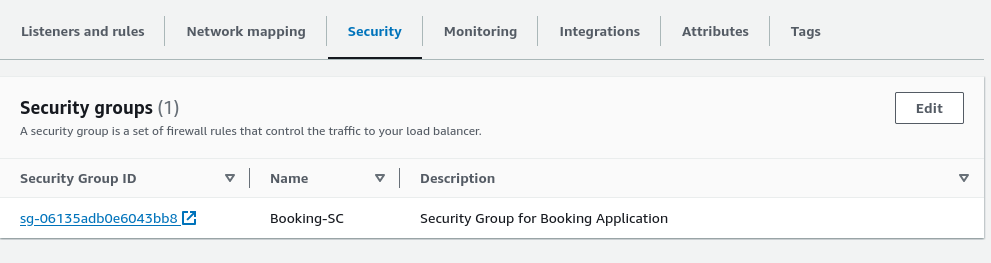


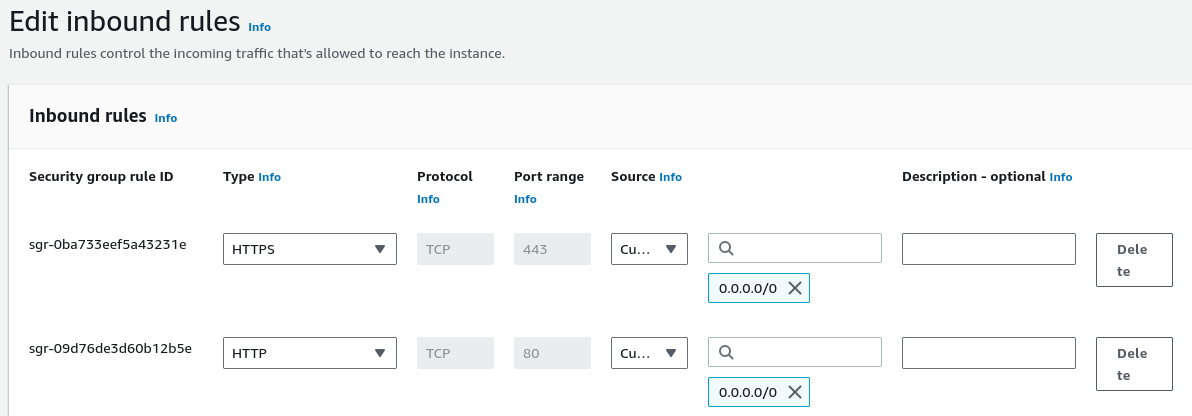
Target group which points to the spring boot application (instance).



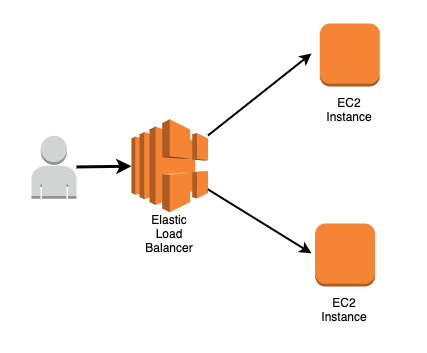


I also created a security group for the load balancer with two inbound rules.





So general map is the following:



Regarding website, it was deployed on

<https://www.000webhost.com/>.Link to access to the webpage is https://leonfoodauthentic.000webhostapp.com/

**13.** Enhancements:

In general, we use PUT when you want to modify a single resource that is already a part of the resources collection. PUT overwrites the resource in its entirety. We use PATCH if the request updates part of the resource. Use POST when you want to add a child resource under resources collection. Therefore, I could use to log in to the application the method PATCH instead of POST because when a user is logged into the application, a part of the record in the user table is updated (field logged\_in). I could change this to be aligned with the concepts.