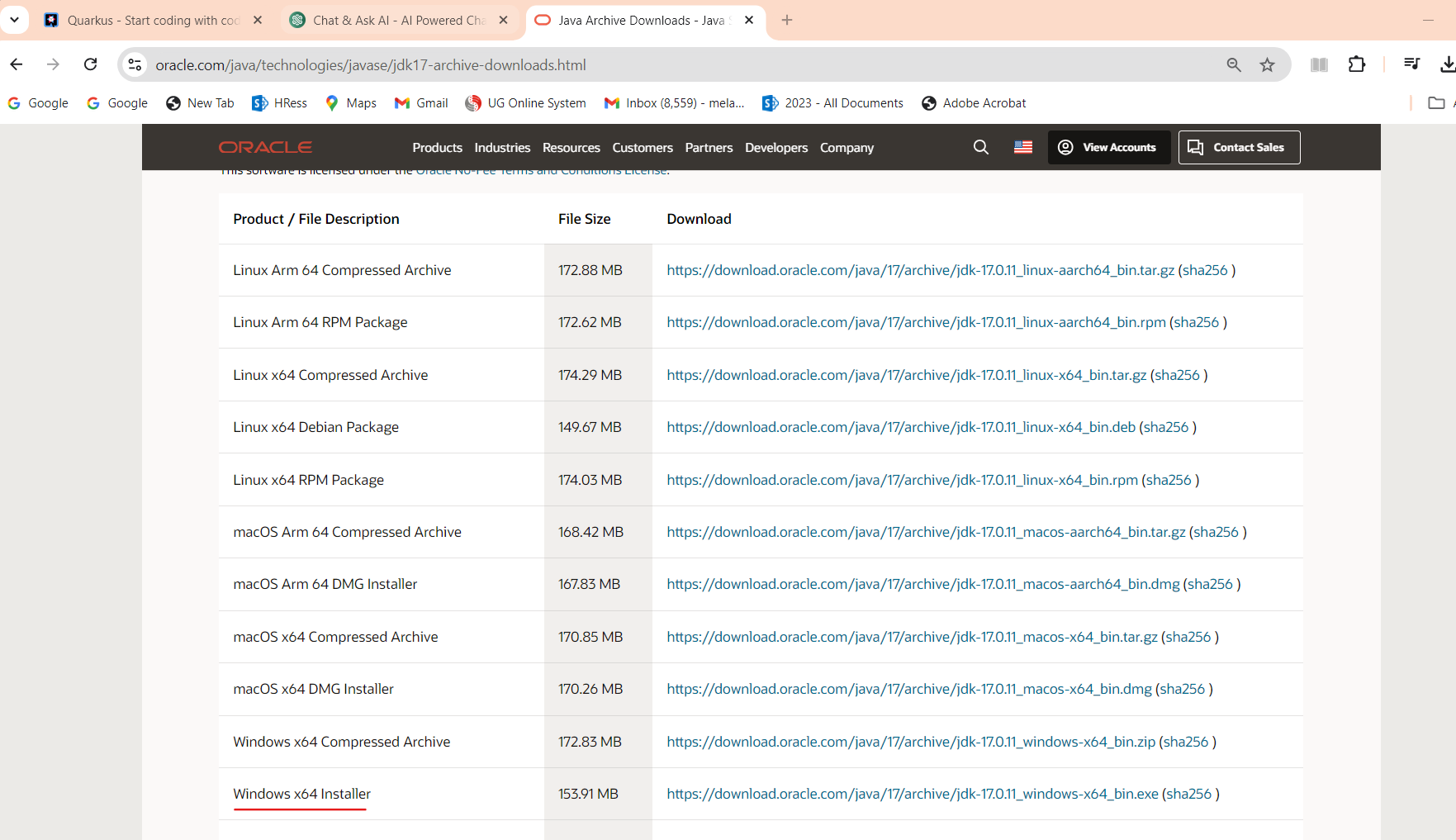
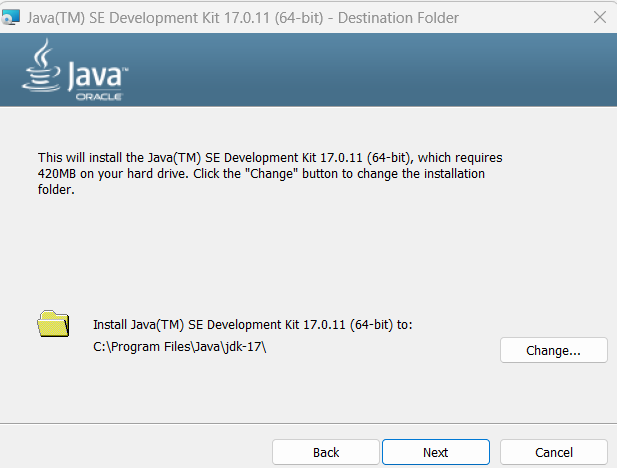
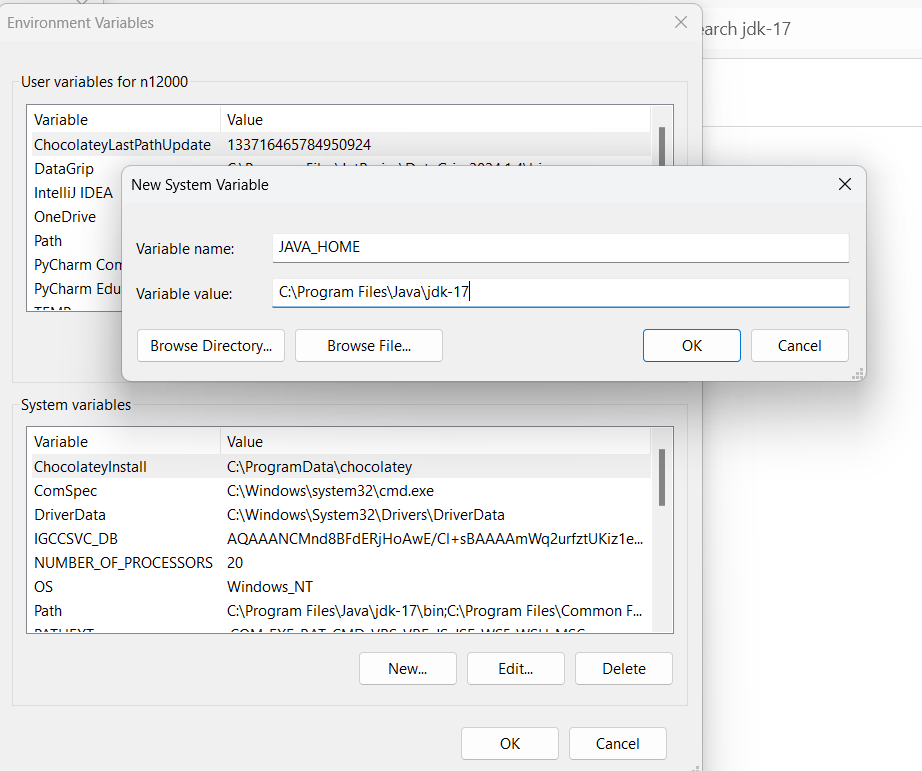
Install JDK-17





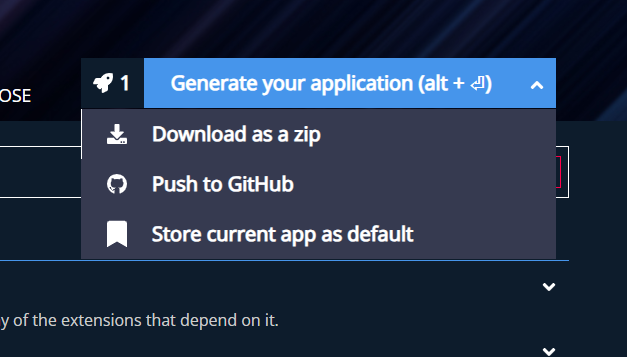
**SET Up Java JAVA\_HOME**



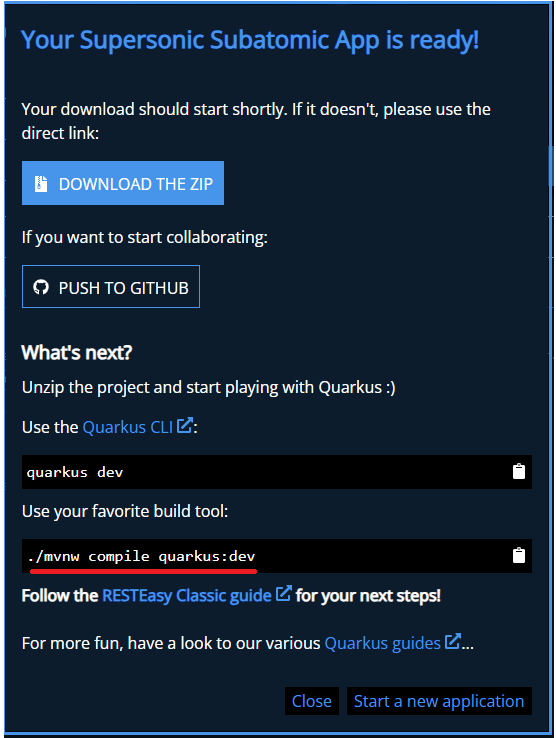
<http://code.quarkus.io>



Choose download as a zip

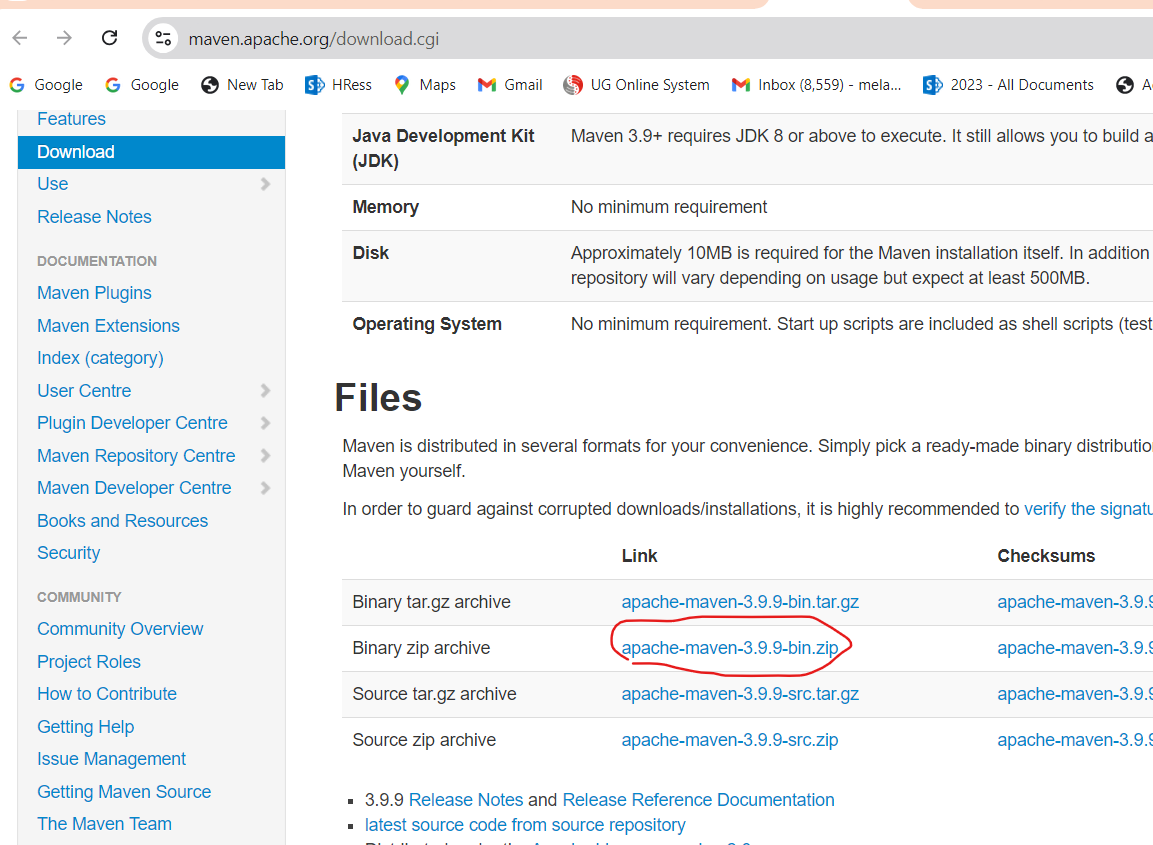


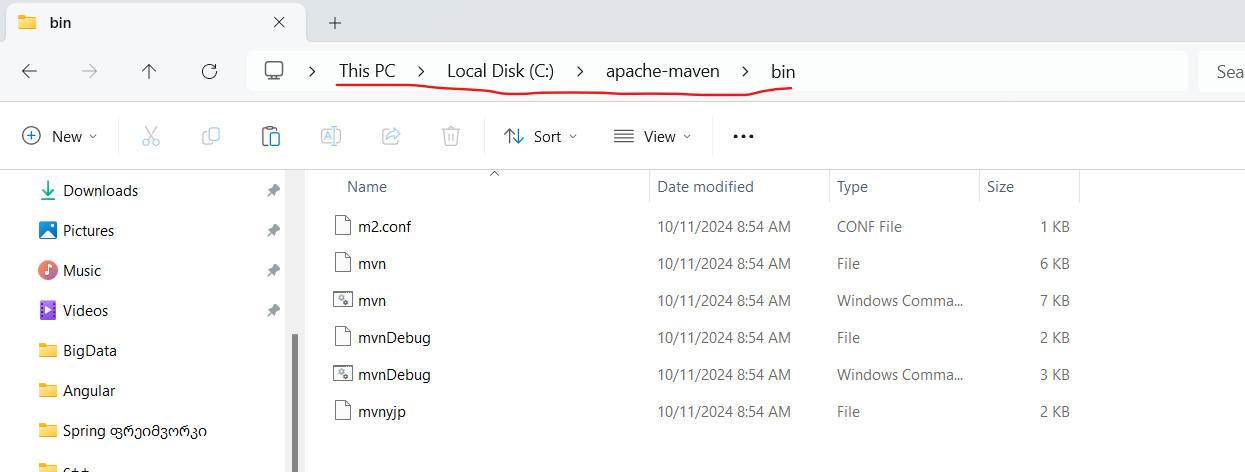
In this window here you see how to run and compile quarkus application:

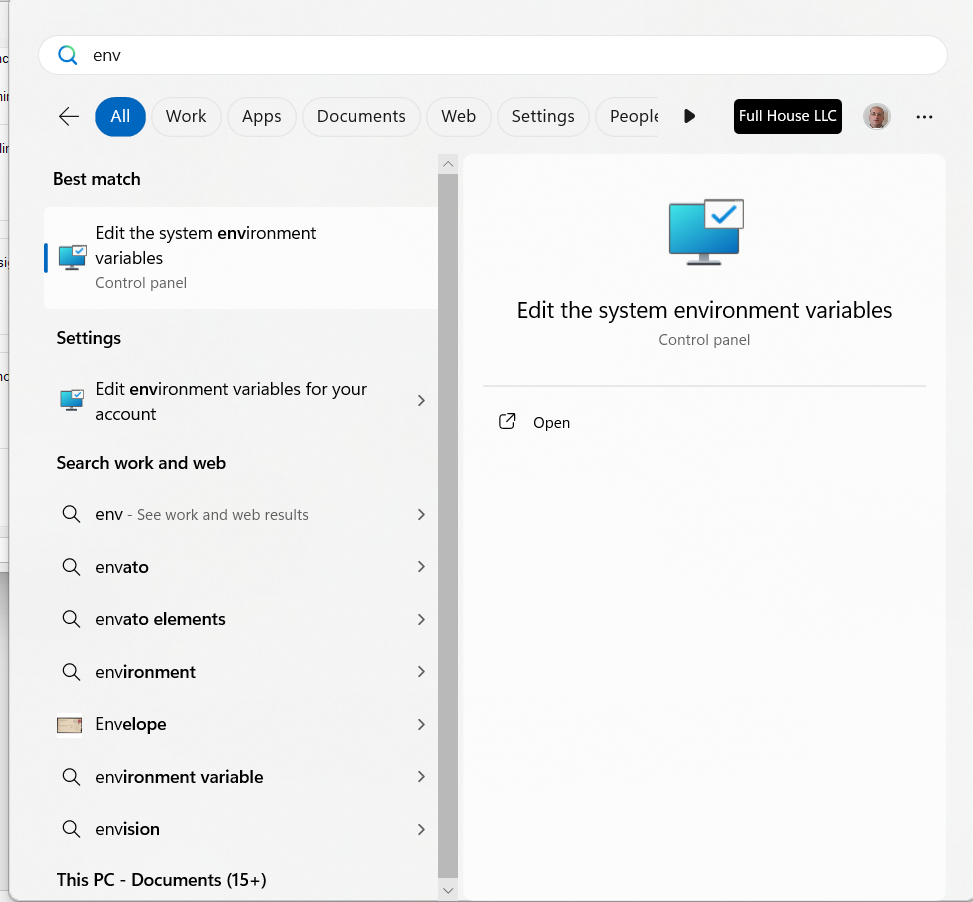


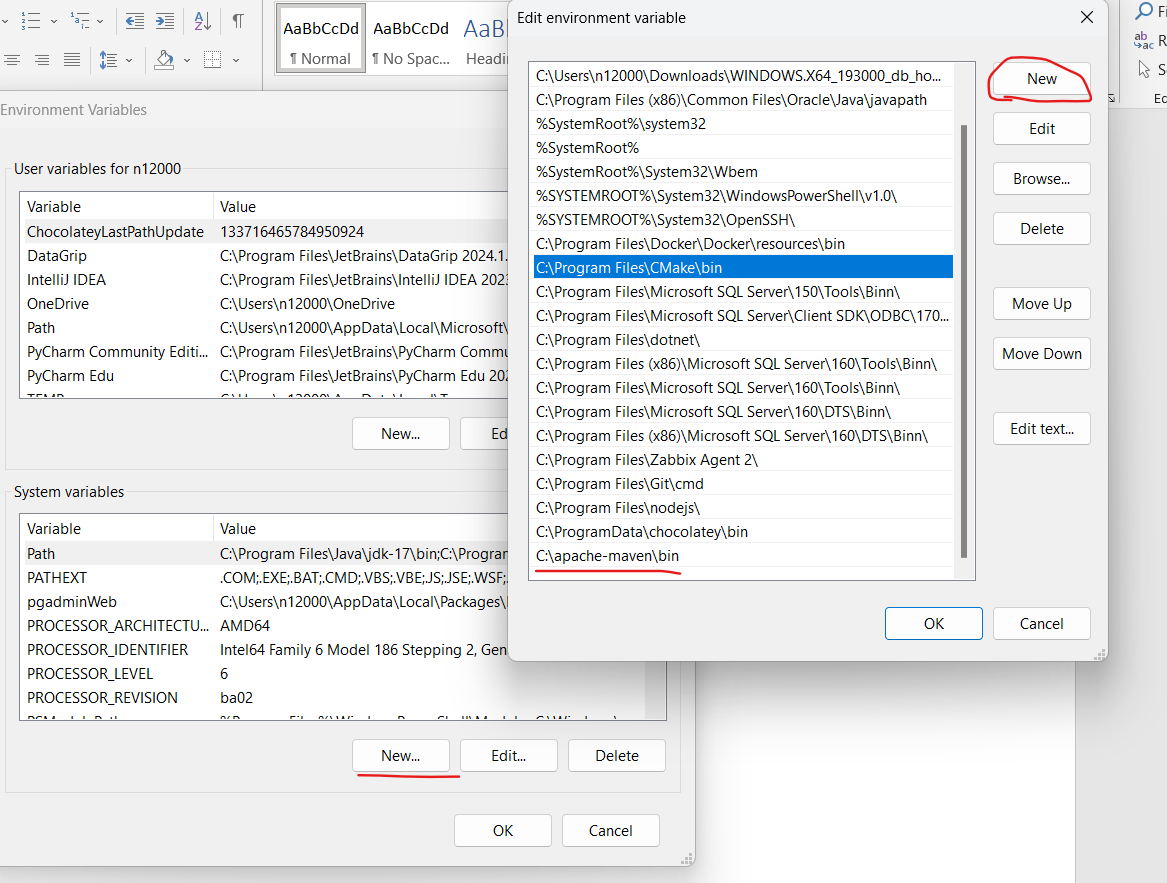
**Download and Install Maven:**

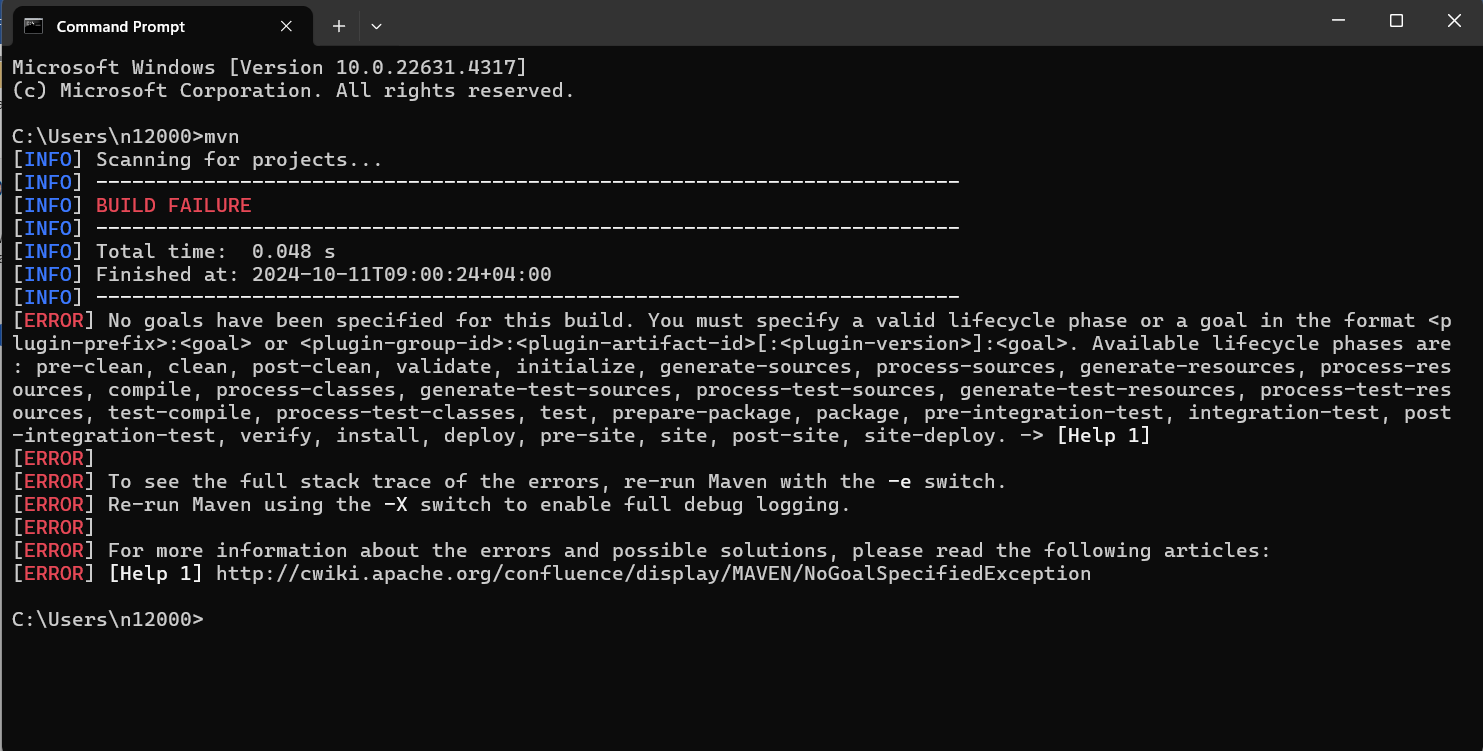
https://maven.apache.org/download.cgi



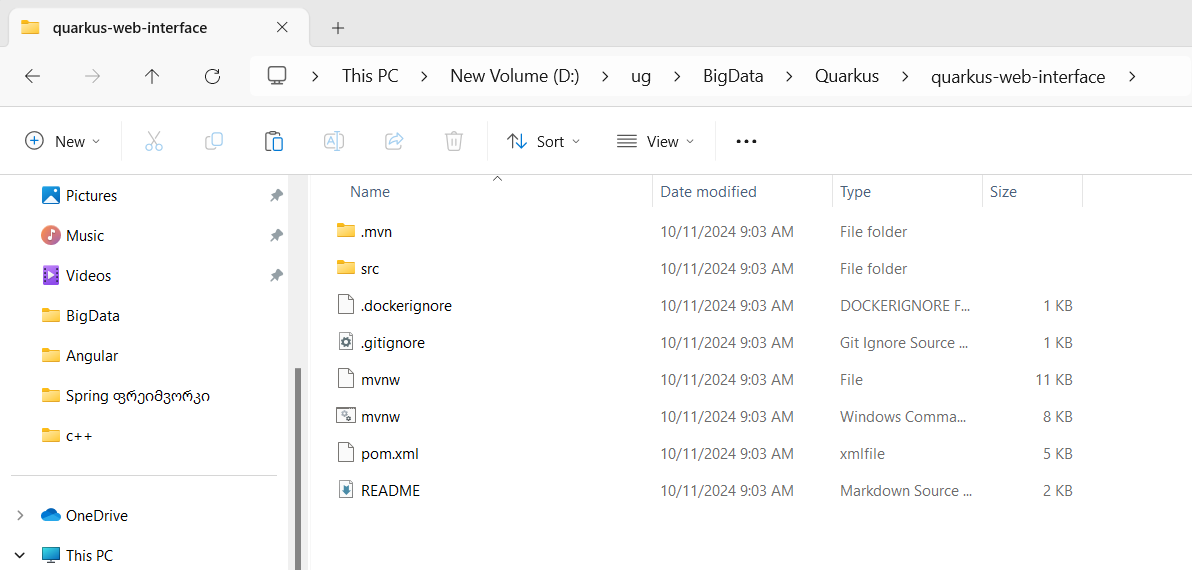








We unzipped downloaded folder:

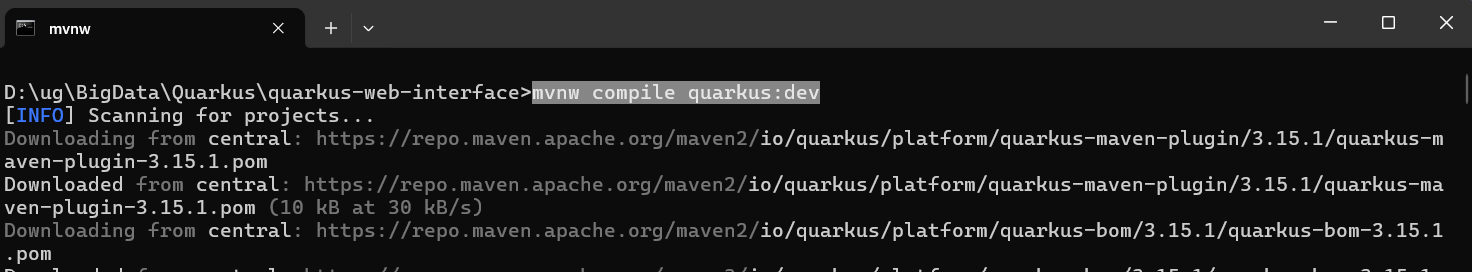


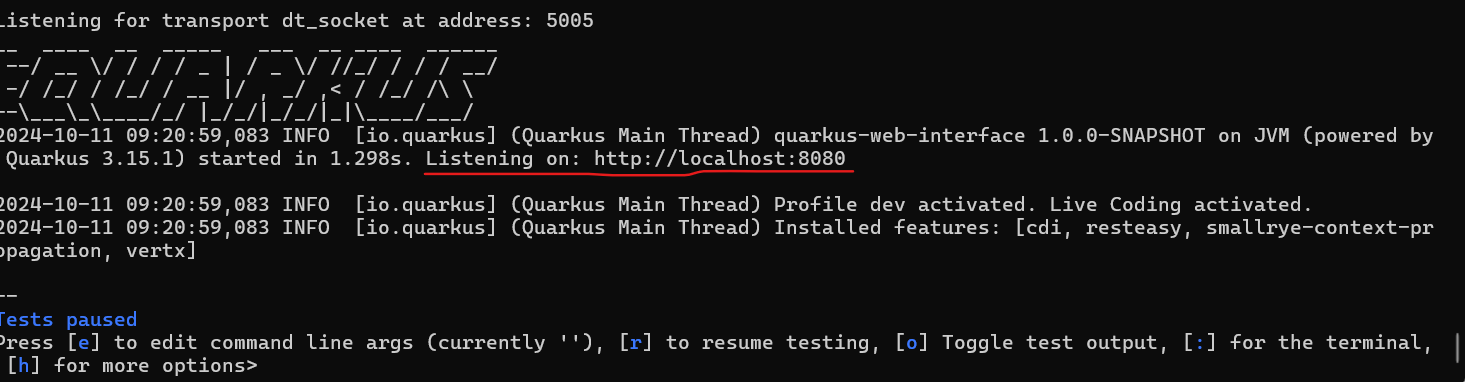


type cmd in the address bar:

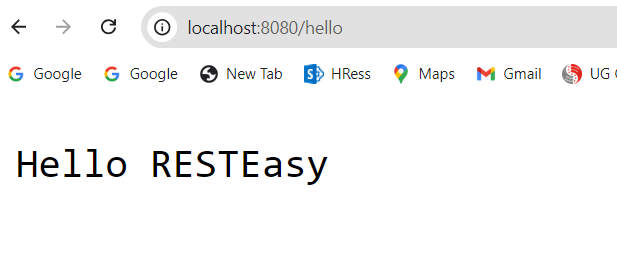
**type:**

**mvnw compile quarkus:dev**

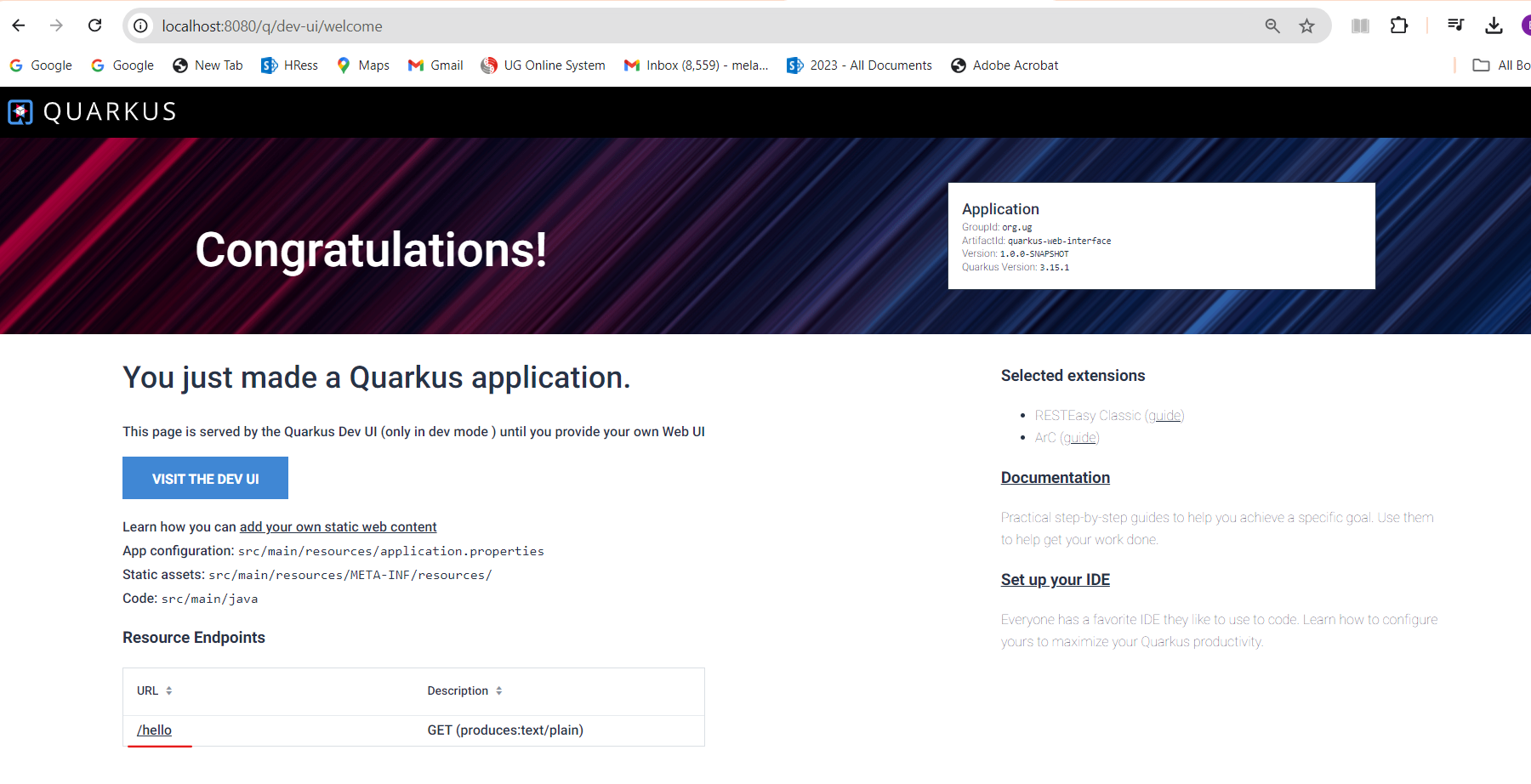




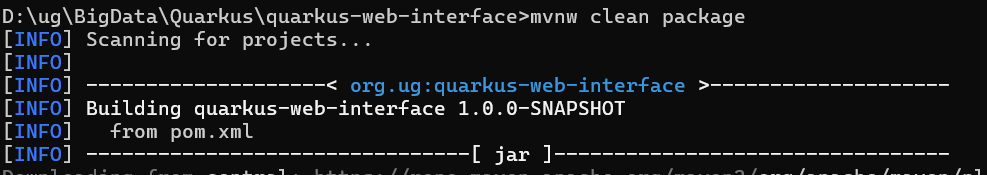
Localhost:8080/hello



Localhost:8080

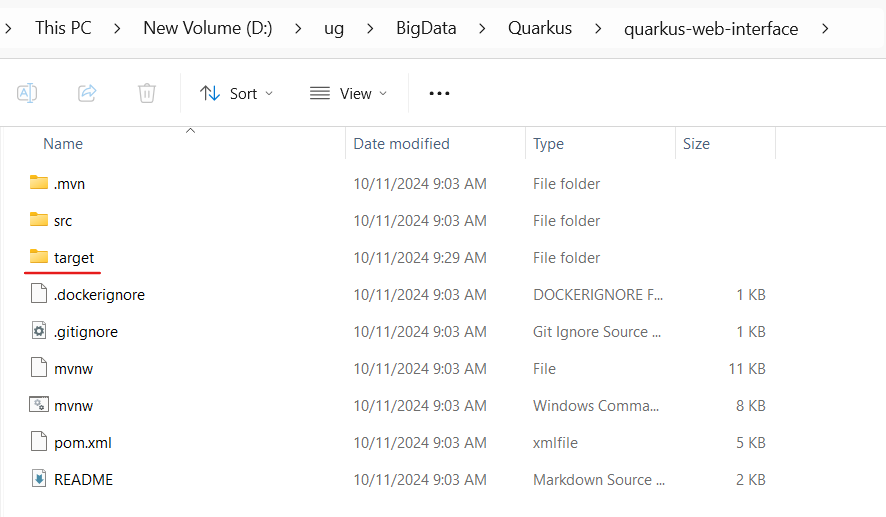


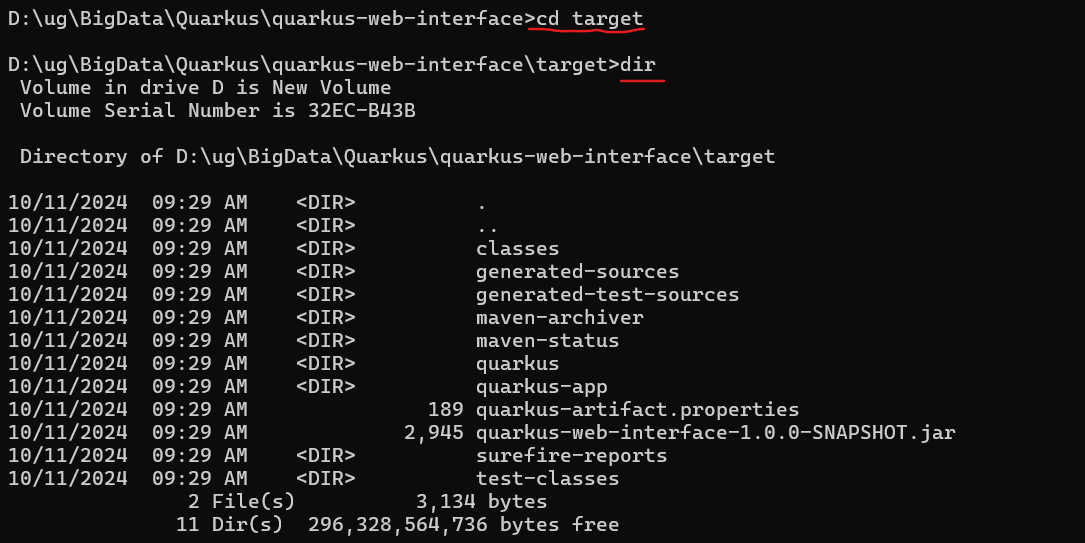
Stop web server and run package our project:

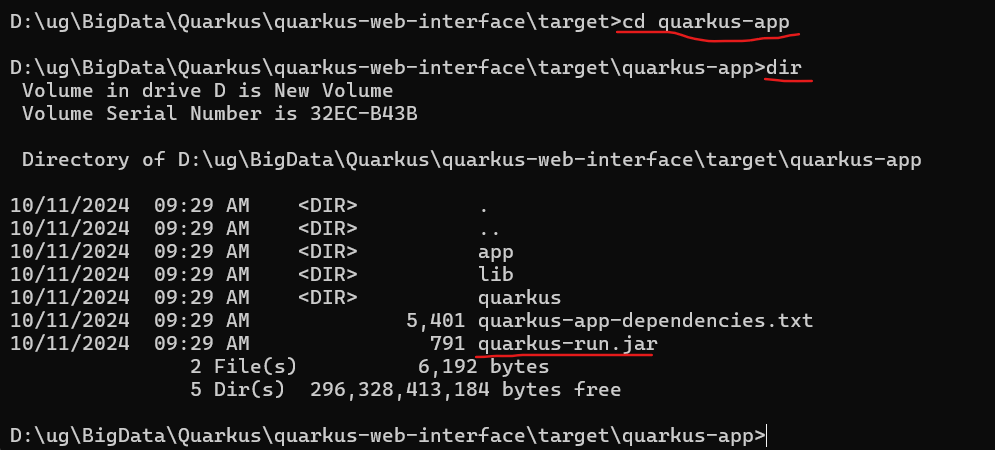
mvnw clean package  




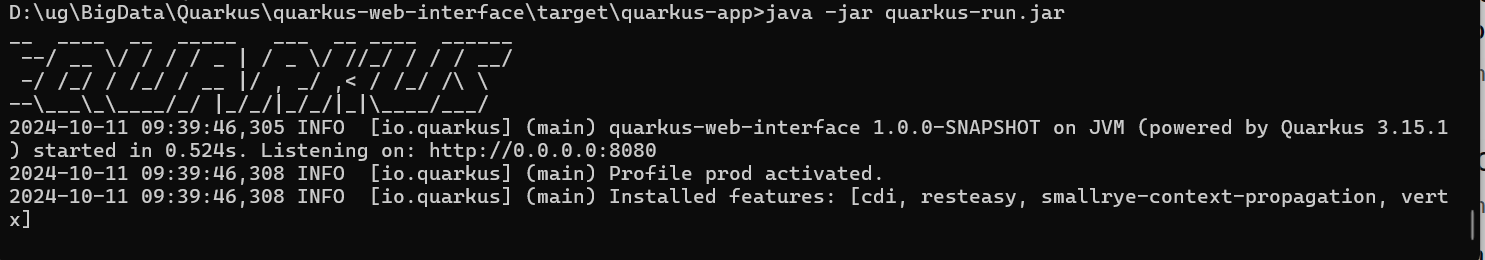
We will see the new folder:

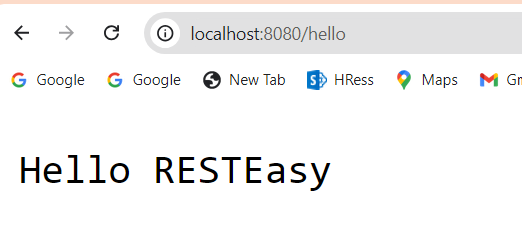




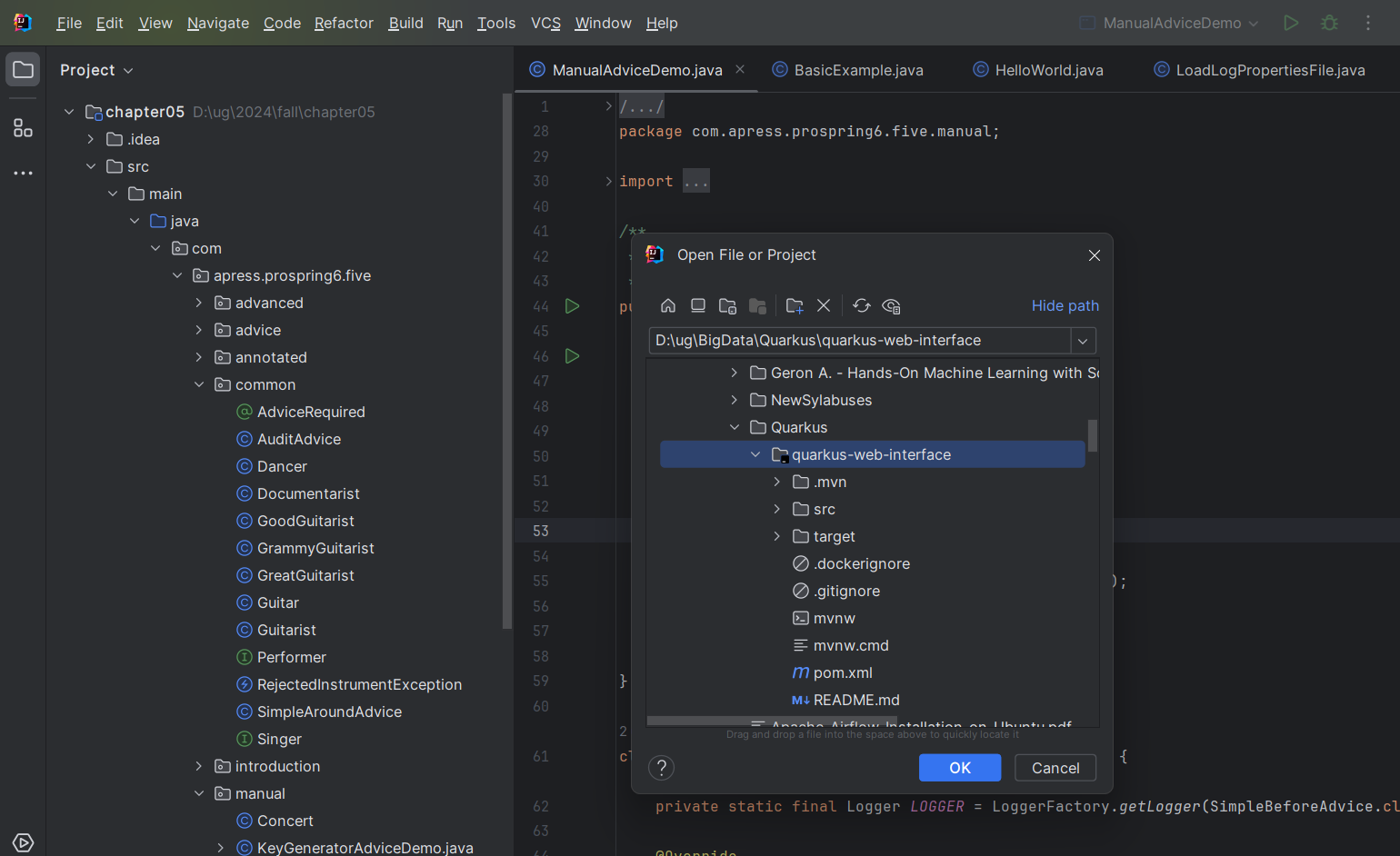


Run application



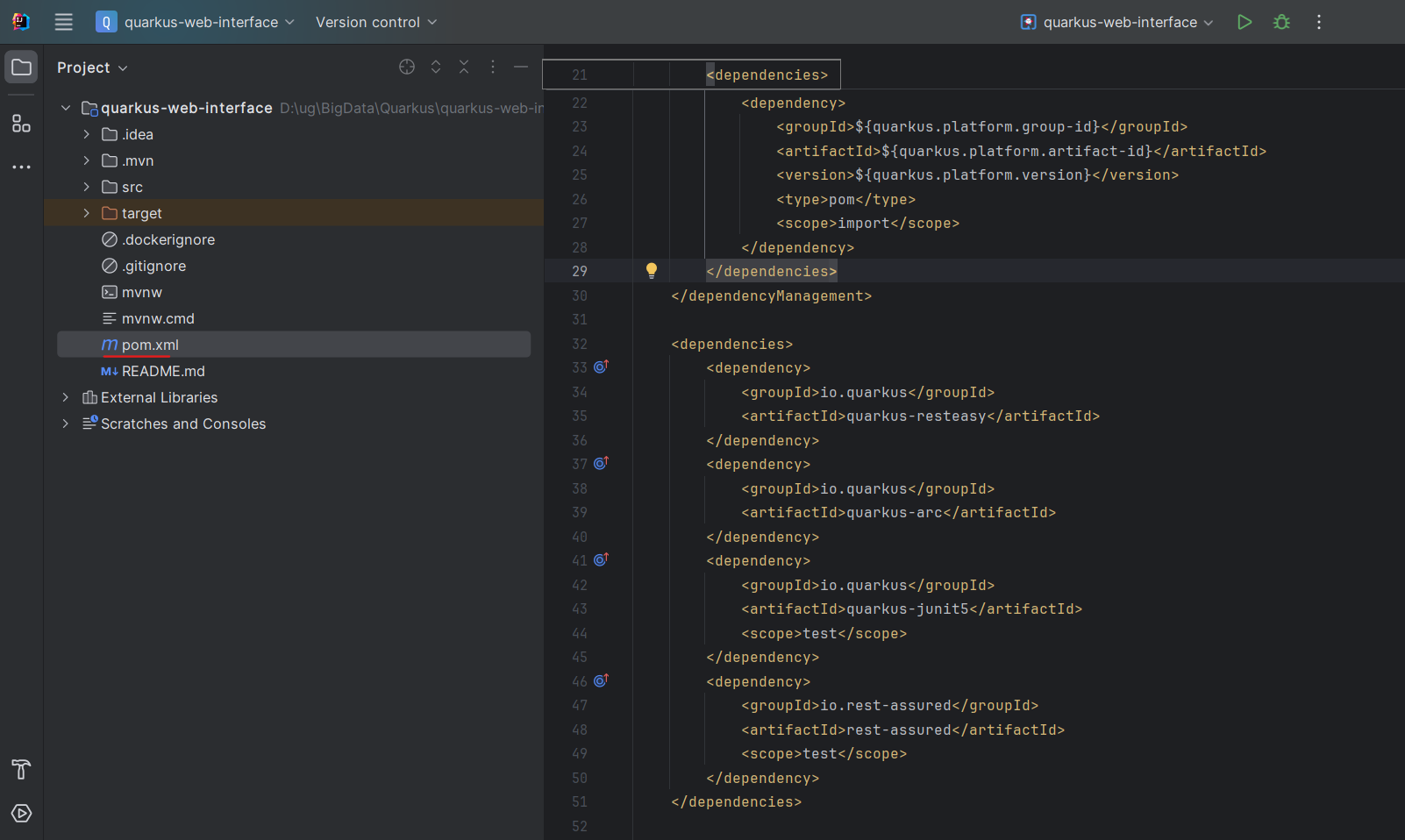


Now Open our Project In Intellij:

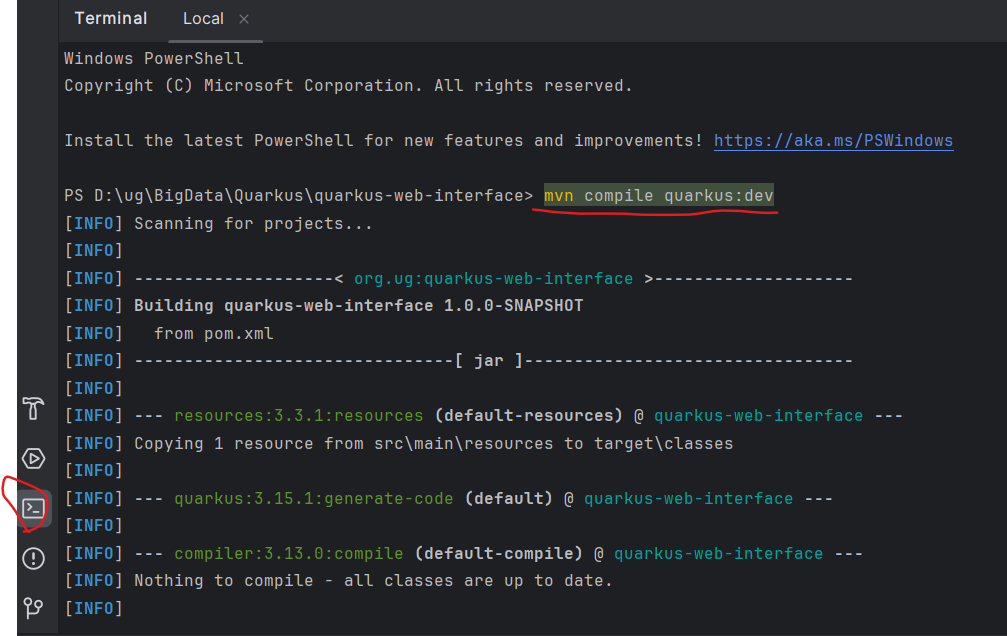


<dependencies>  
 <dependency>  
 <groupId>io.quarkus</groupId>  
 <artifactId>quarkus-resteasy</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>io.quarkus</groupId>  
 <artifactId>quarkus-arc</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>io.quarkus</groupId>  
 <artifactId>quarkus-junit5</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>io.rest-assured</groupId>  
 <artifactId>rest-assured</artifactId>  
 <scope>test</scope>  
 </dependency>  
</dependencies>

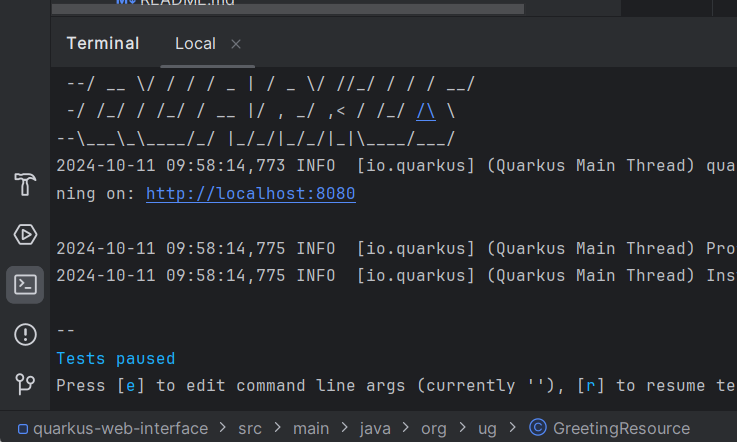
We can see this dependencies in the pom.xml file:

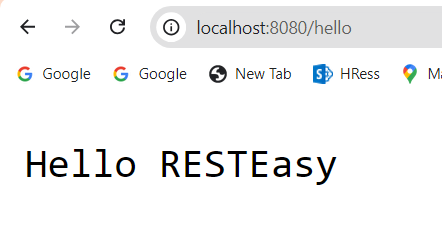


Compile project and run from IntelliJ Terminal Window

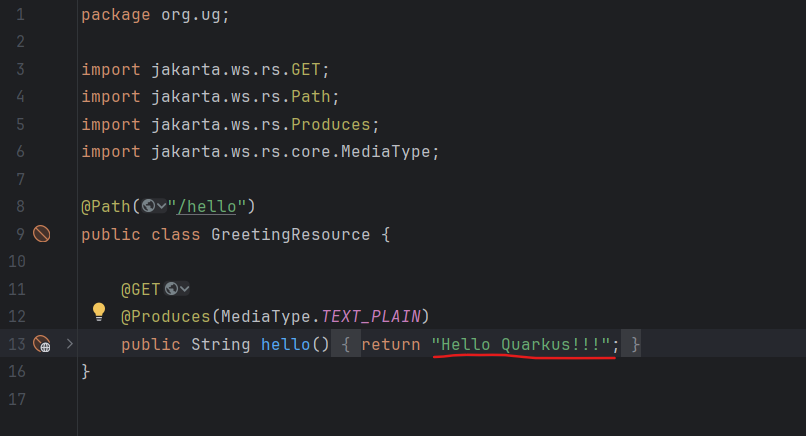


We will see:

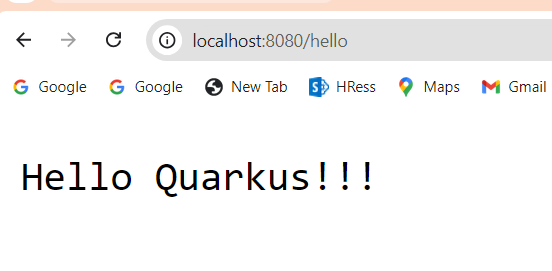


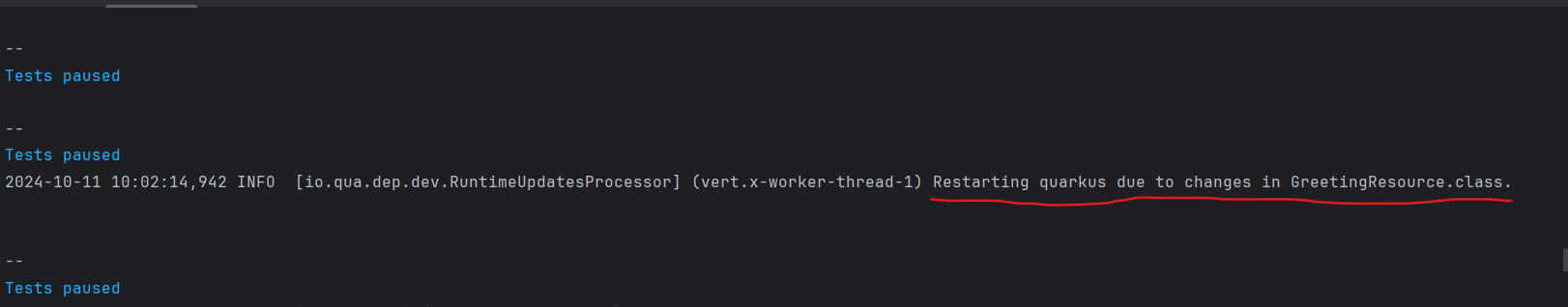


Lets change source

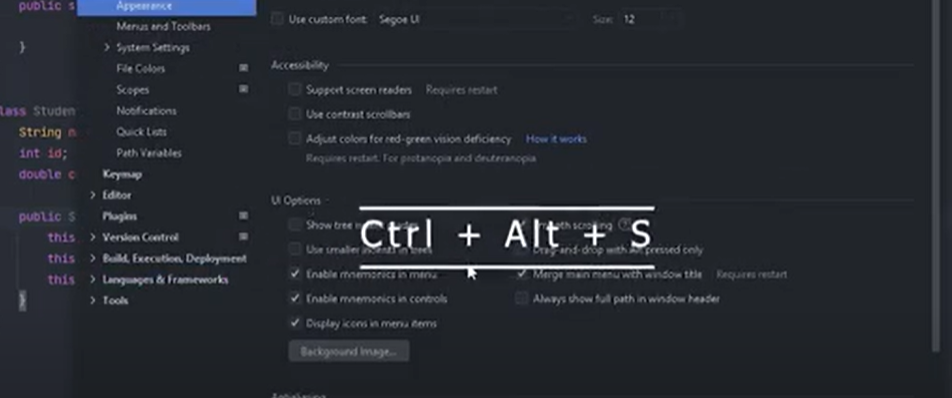


And reload page:

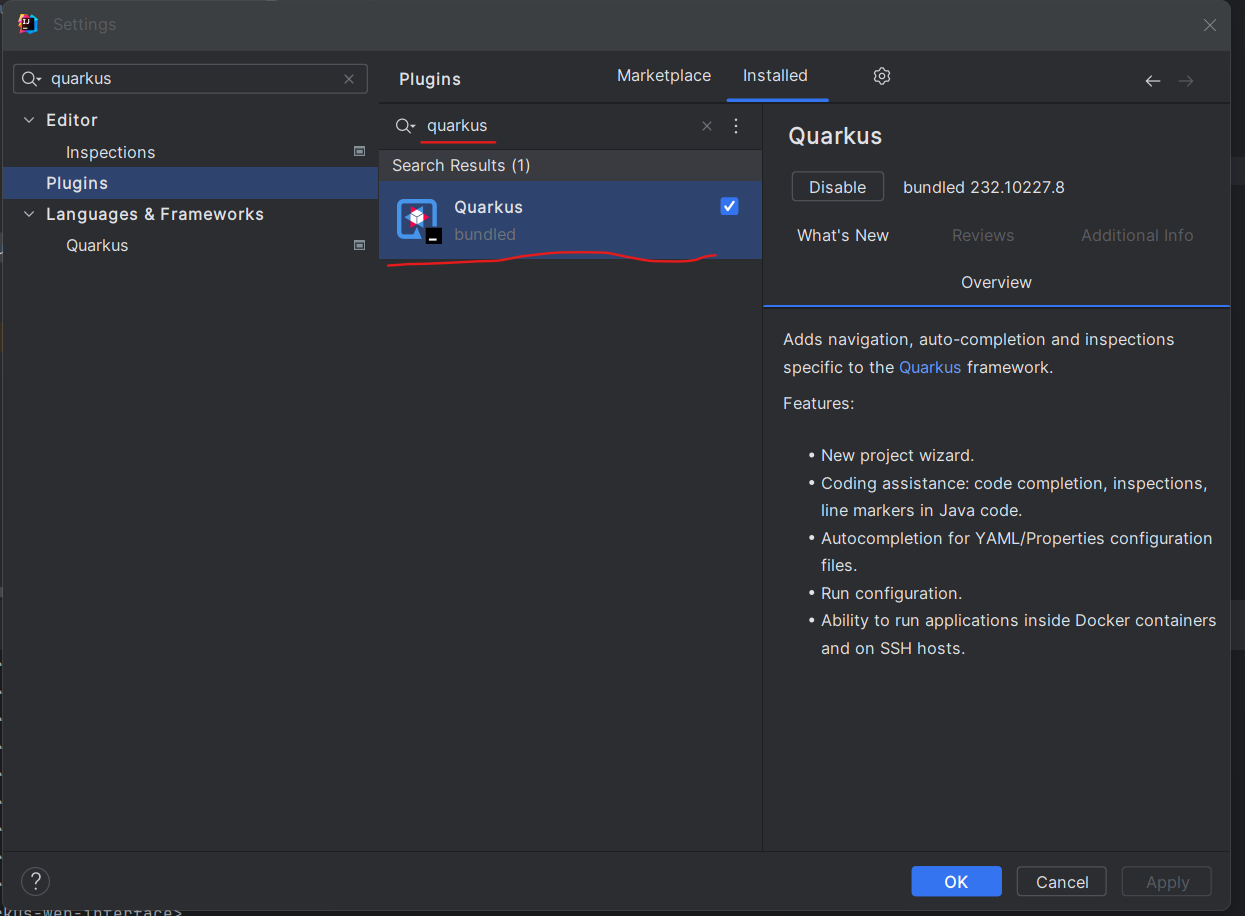




Install Quarkus plugin Inside IntelliJ:

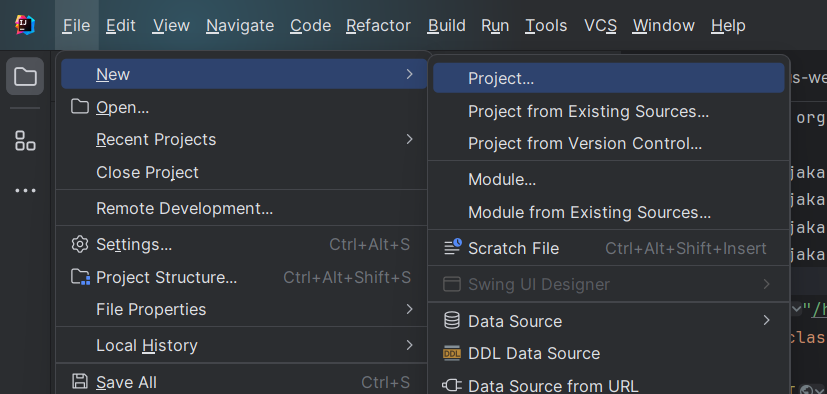


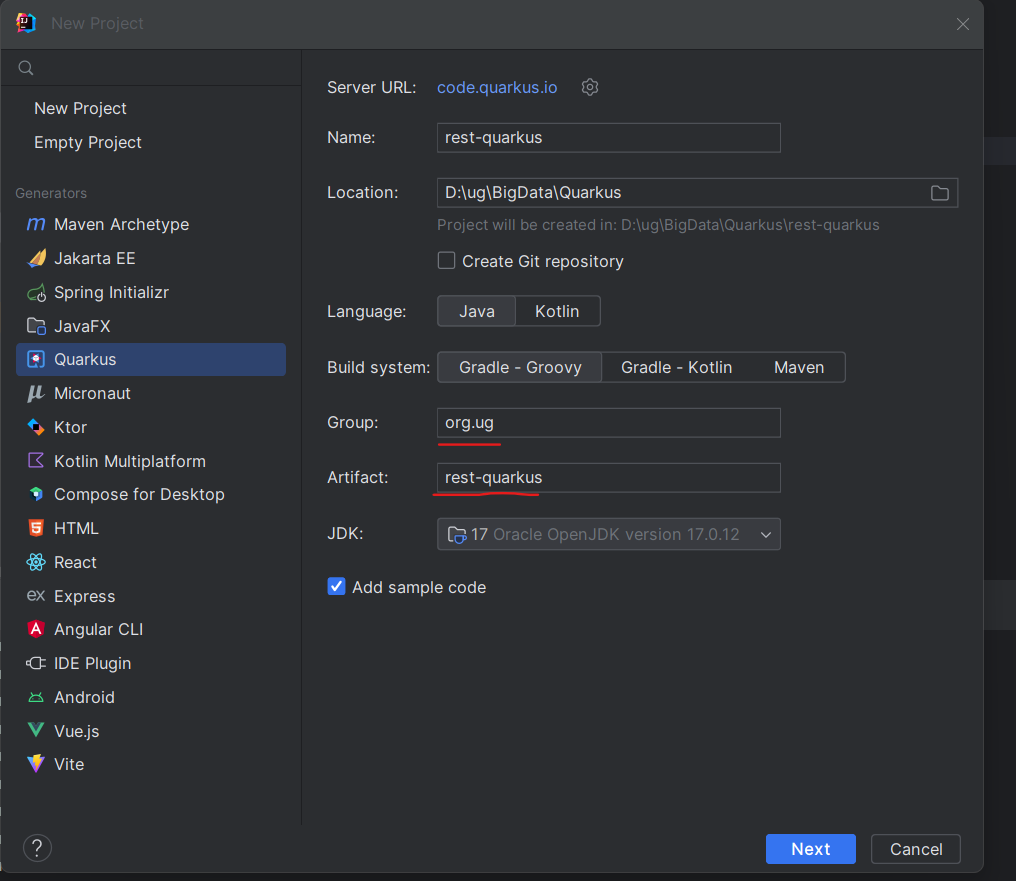
Type Quarkus in Search box:



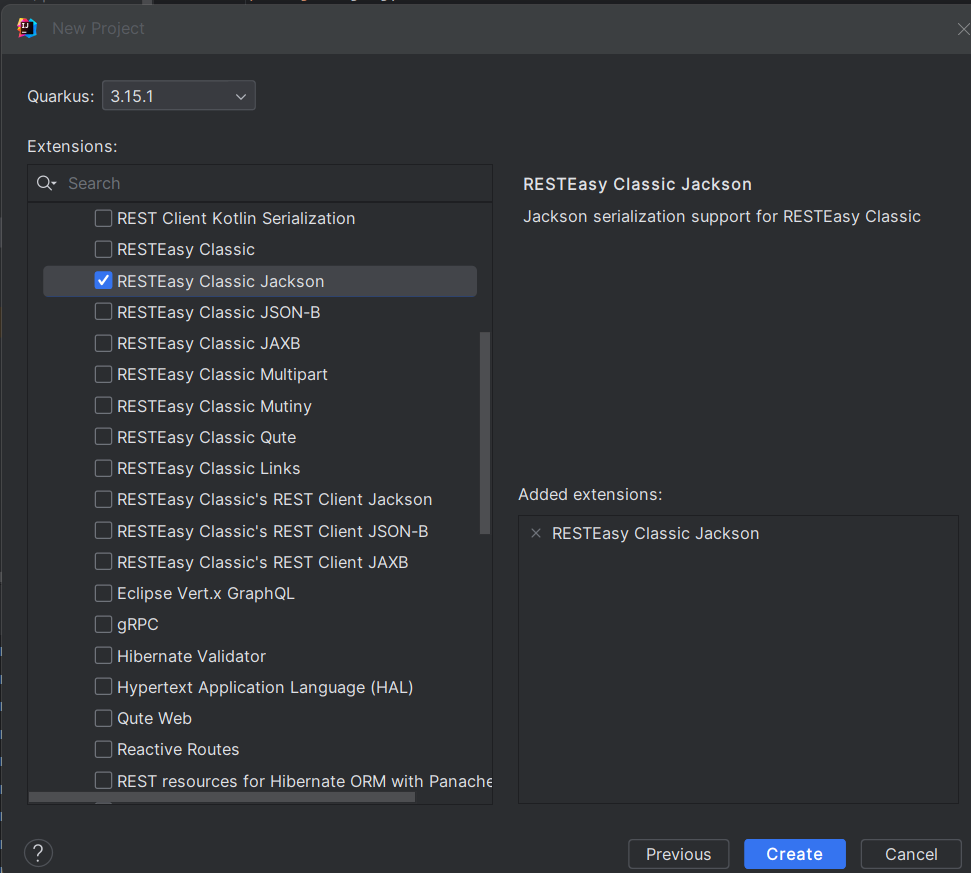


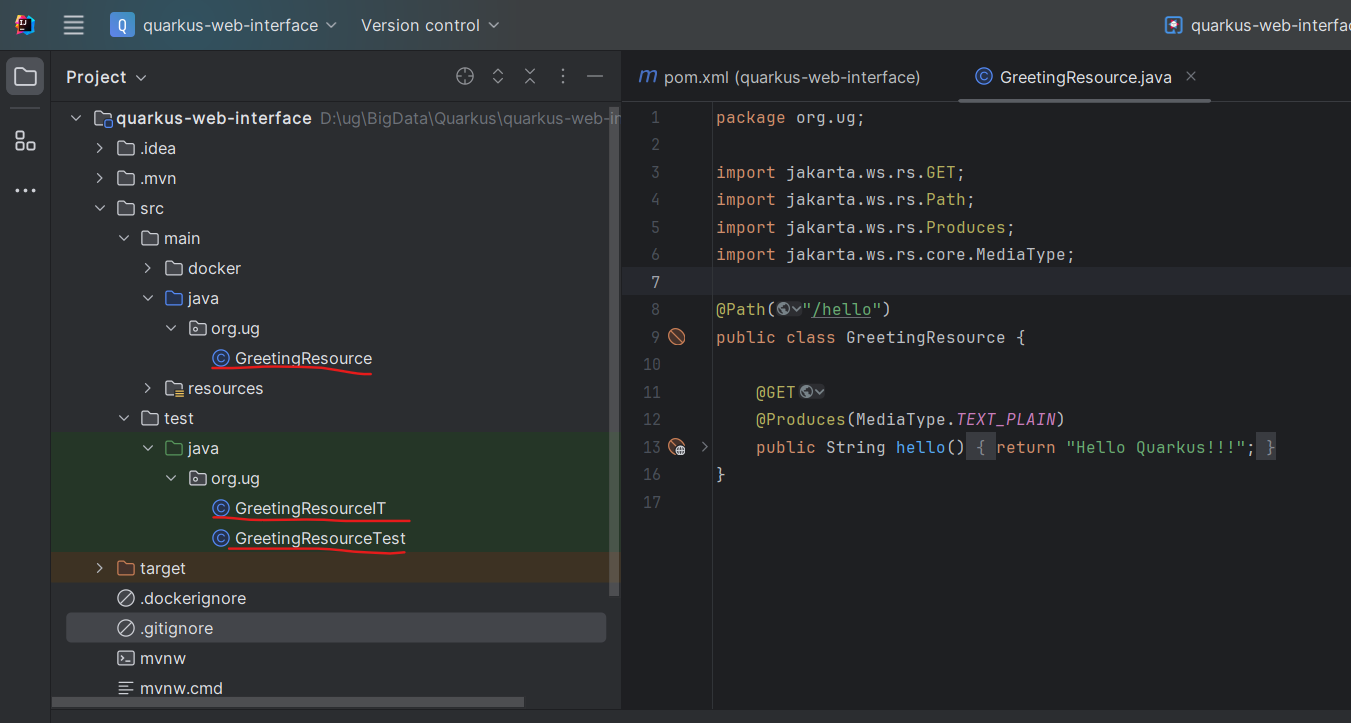
Lets Create new QuarKus project from IntelliJ:

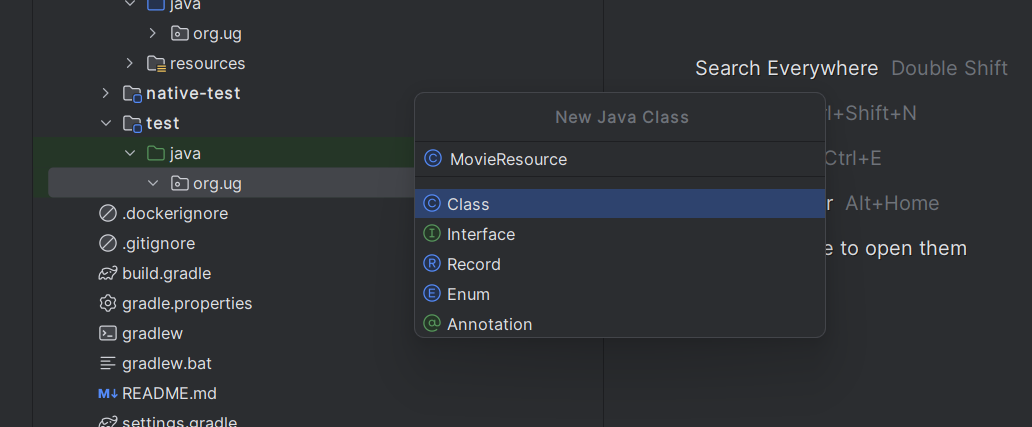


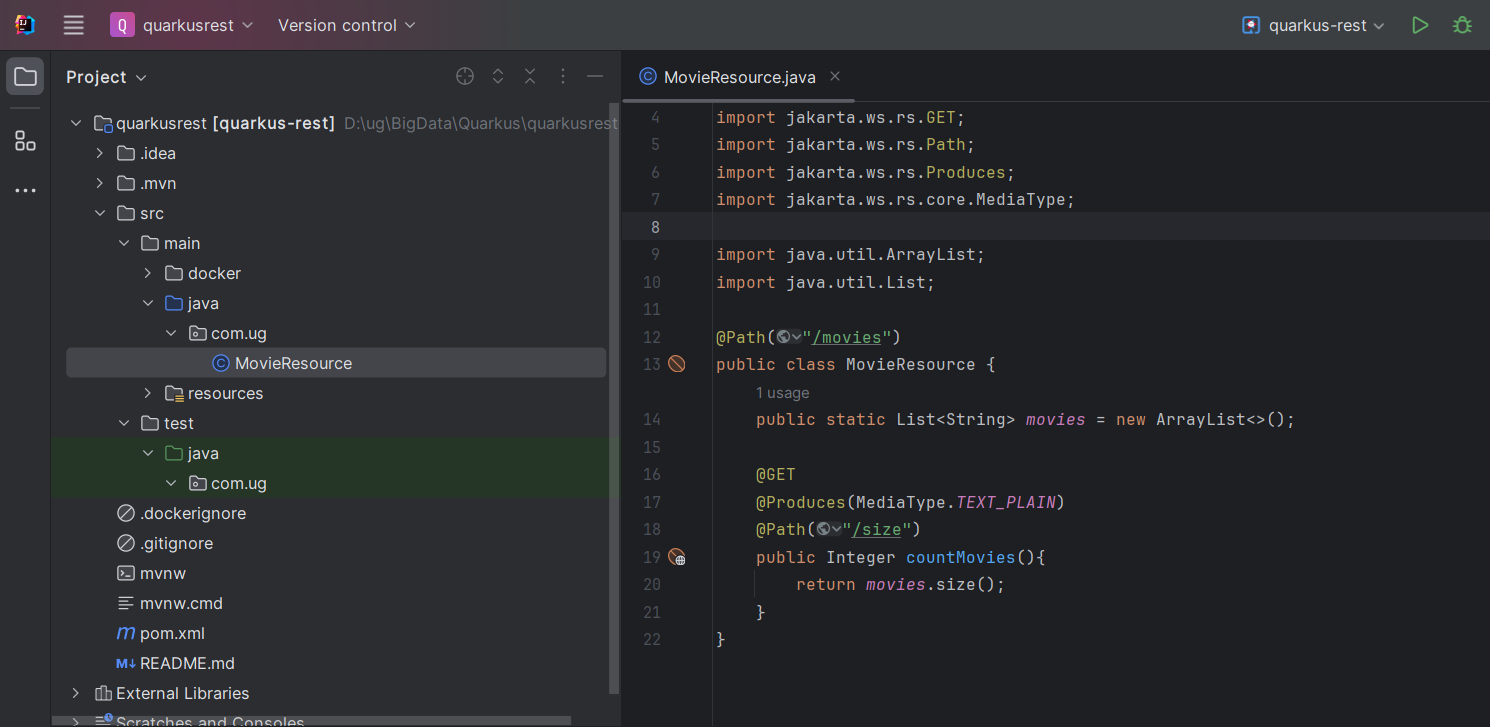


Select the following:

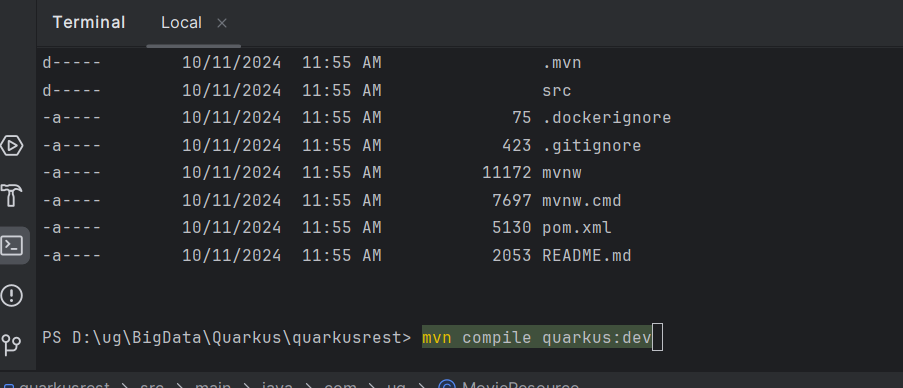


Delete these files:  


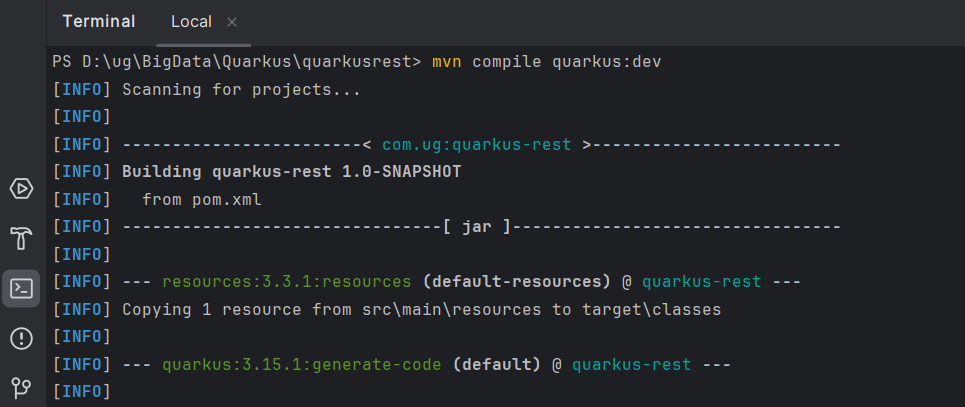
And Create new class: MovieResource:  




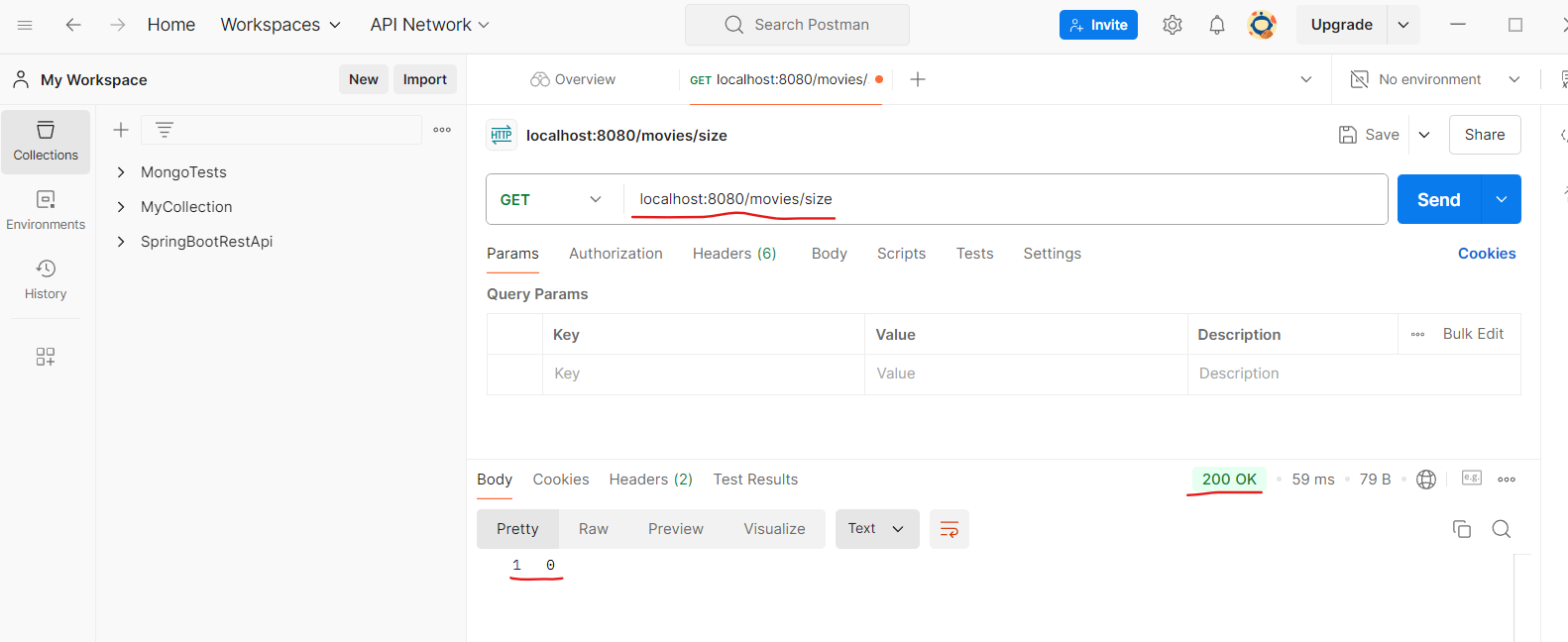
package com.ug;  
  
import com.fasterxml.jackson.databind.ser.std.NumberSerializers;  
import jakarta.ws.rs.GET;  
import jakarta.ws.rs.Path;  
import jakarta.ws.rs.Produces;  
import jakarta.ws.rs.core.MediaType;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Path("/movies")  
public class MovieResource {  
 public static List<String> *movies* = new ArrayList<>();  
  
 @GET  
 @Produces(MediaType.*TEXT\_PLAIN*)  
 @Path("/size")  
 public Integer countMovies(){  
 return *movies*.size();  
 }  
}



The quarkus Application Will start



Use postman and send “Get” **request** to **localhost:8080/movies/size**

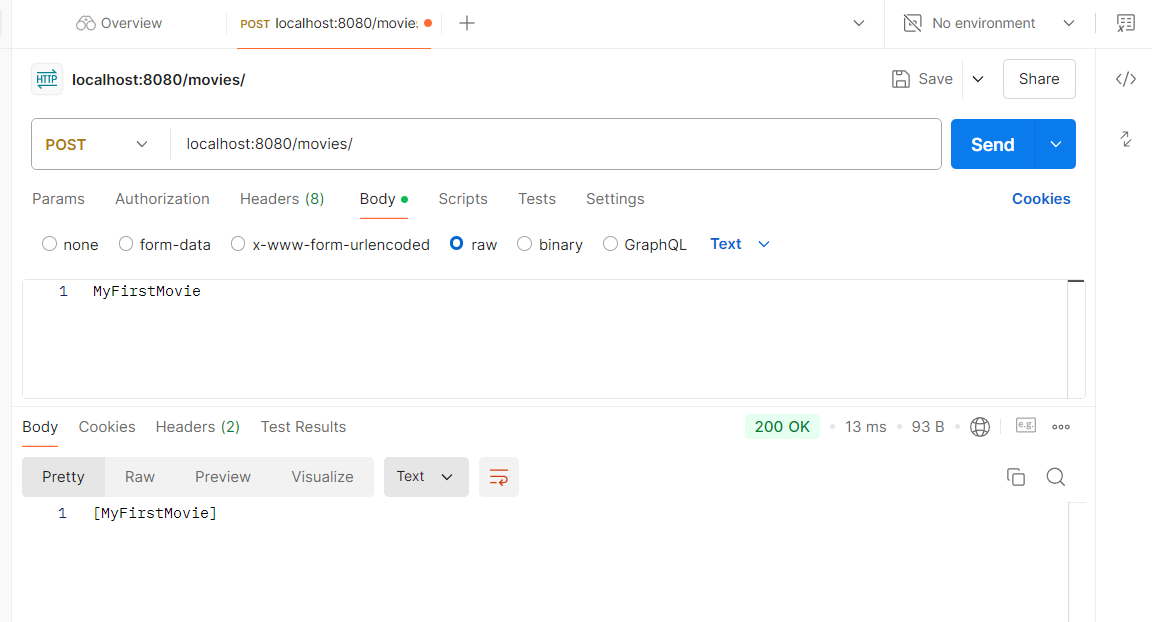


Modify MovieResource class: Add the GET and Post Methods

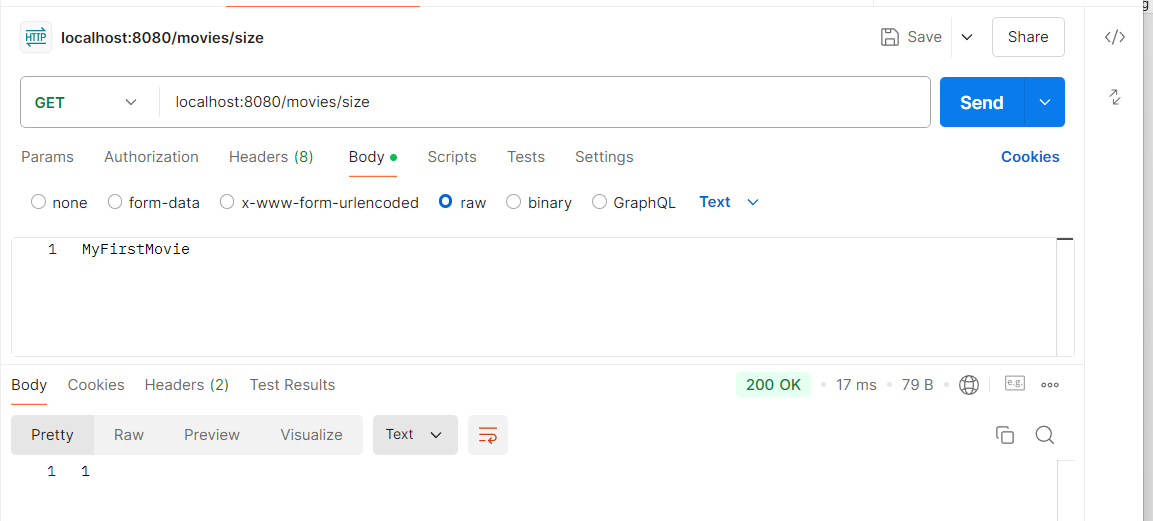
@GET  
@Produces(MediaType.*TEXT\_PLAIN*)  
public Response getMovies(){  
return Response.*ok*(*movies*).build();  
}

@POST  
@Produces(MediaType.*TEXT\_PLAIN*)  
@Consumes(MediaType.*TEXT\_PLAIN*)  
public Response createMovie(String newMovie){  
 *movies*.add(newMovie);  
 return Response.*ok*(*movies*).build();  
}

After modification run post method

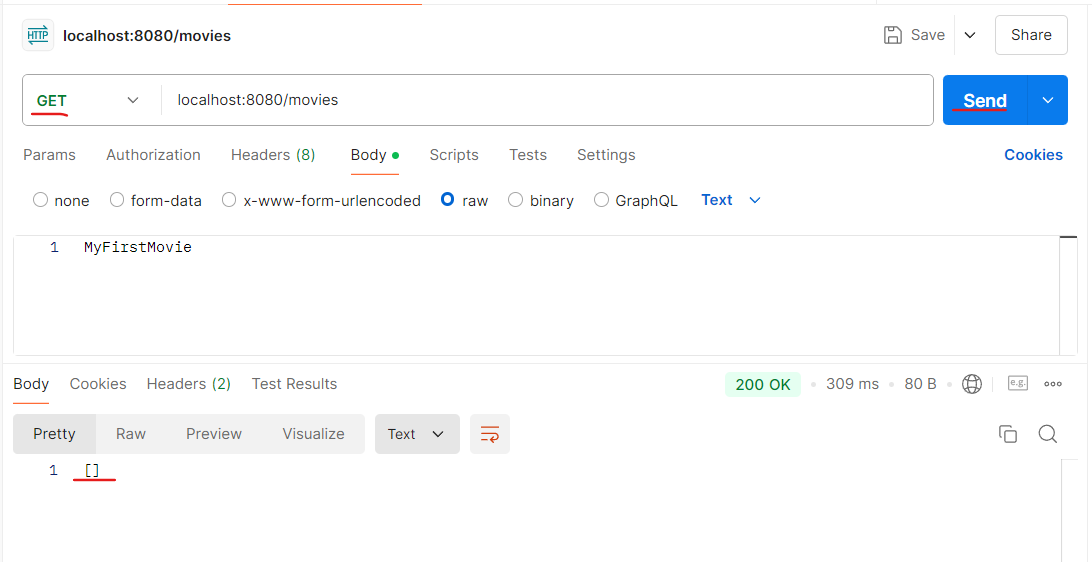


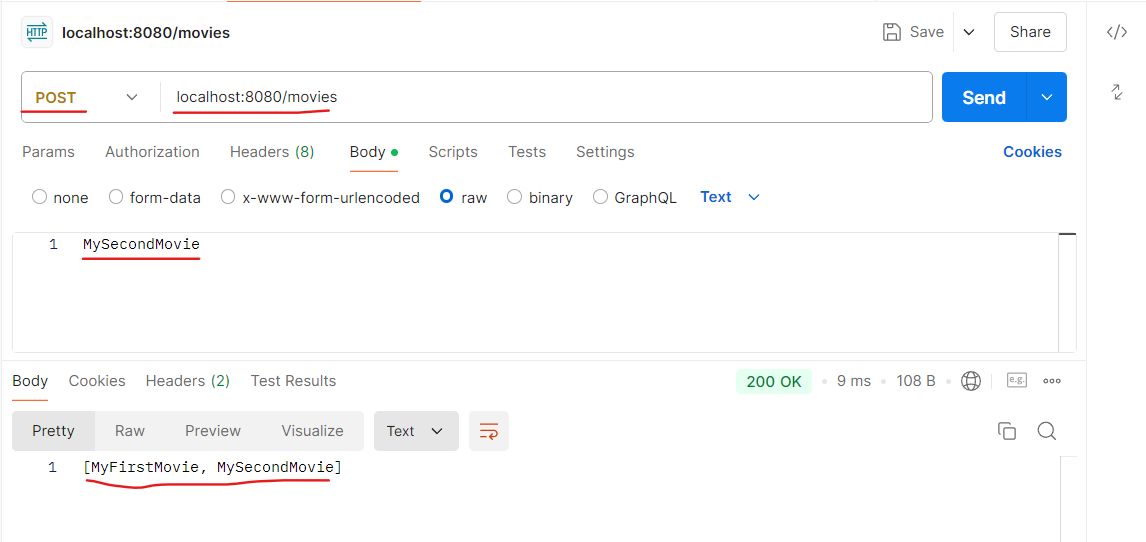
Run once again “Get” **request** to **localhost:8080/movies/size**



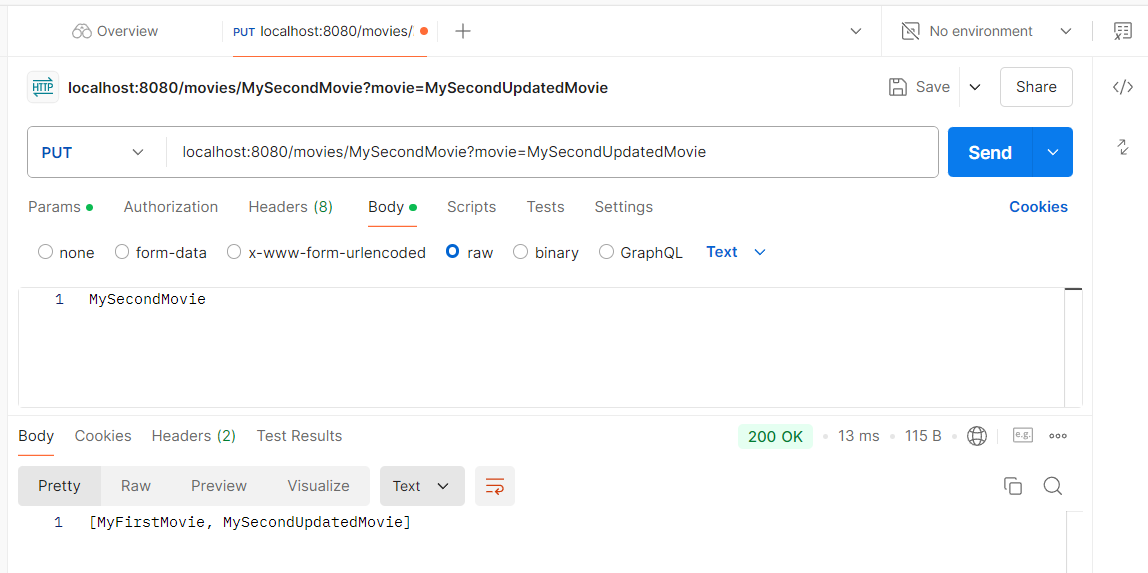
Now add Put Method:

@PUT  
@Path("{movieToUpdate}")  
@Produces(MediaType.*TEXT\_PLAIN*)  
@Consumes(MediaType.*TEXT\_PLAIN*)  
public Response updateMovie(  
 @PathParam("movieToUpdate") String movieToUpdate,  
 @QueryParam("movie") String updateMovie) {  
  
 *movies* = *movies*.stream().map(movie -> {  
 if (movie.equals(movieToUpdate)) {  
 return updateMovie; // Update the movie if it matches  
 }  
 return movie; // Return the original movie if it doesn't match  
 }).collect(Collectors.*toList*());  
  
 return Response.*ok*(*movies*).build();  
}

First send the get method:  


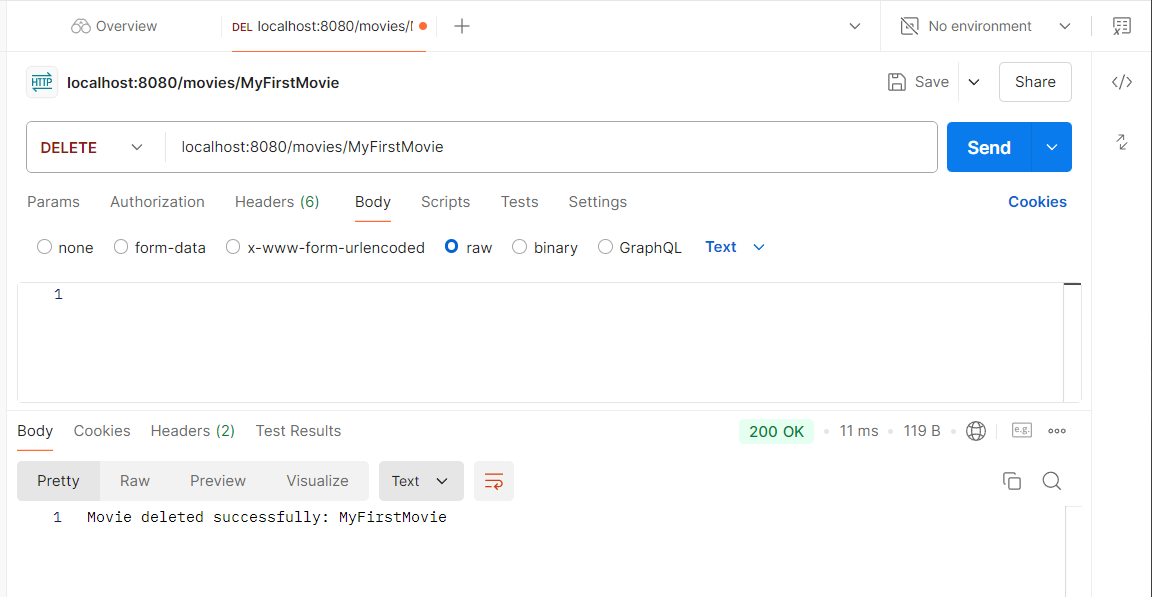
Next lets send two post methods:  


Now send put method



Delete Request:

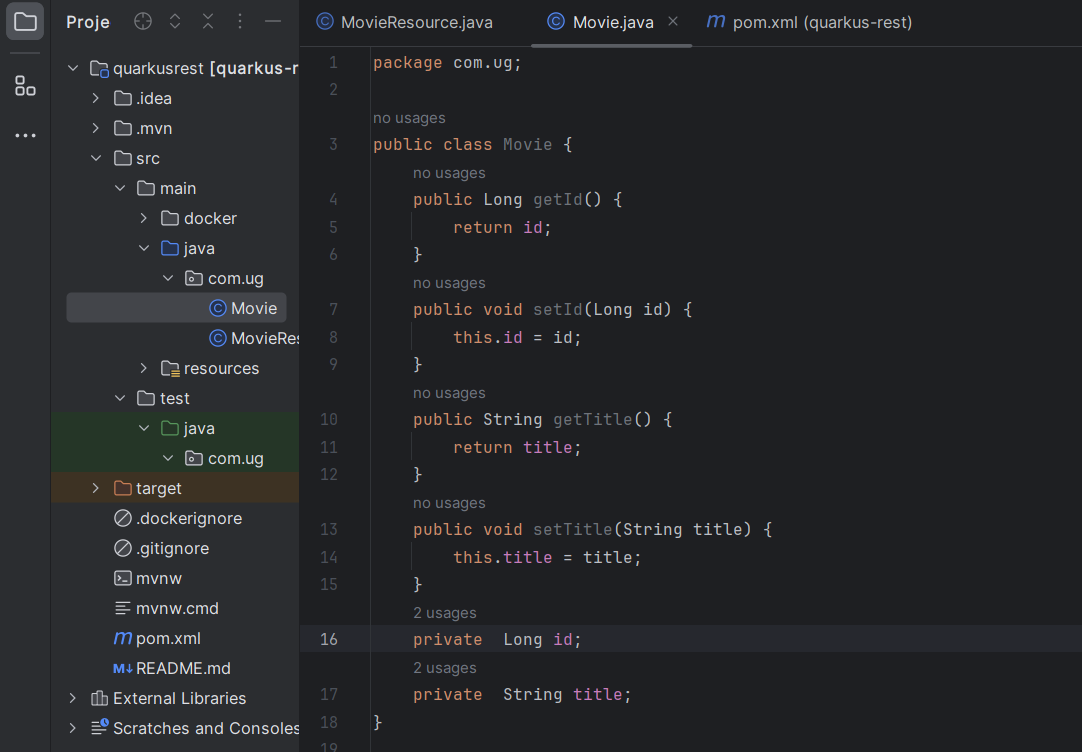
@DELETE  
@Path("{movieToDelete}")  
@Produces(MediaType.*TEXT\_PLAIN*)  
public Response deleteMovie(@PathParam("movieToDelete") String movieToDelete) {  
 boolean removed = *movies*.removeIf(movie -> movie.equals(movieToDelete));  
  
 if (removed) {  
 return Response.*ok*("Movie deleted successfully: " + movieToDelete).build();  
 } else {  
 return Response.*status*(Response.Status.*NOT\_FOUND*)  
 .entity("Movie not found: " + movieToDelete)  
 .build();  
 }  
}



So Our Final MovieResource class looks as follows:

package com.ug;  
  
import com.fasterxml.jackson.databind.ser.std.NumberSerializers;  
import jakarta.ws.rs.\*;  
import jakarta.ws.rs.core.MediaType;  
import jakarta.ws.rs.core.Response;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
@Path("/movies")  
public class MovieResource {  
 public static List<String> *movies* = new ArrayList<>();  
  
 @GET  
 @Produces(MediaType.*TEXT\_PLAIN*)  
 public Response getMovies(){  
 return Response.*ok*(*movies*).build();  
 }  
  
 @GET  
 @Produces(MediaType.*TEXT\_PLAIN*)  
 @Path("/size")  
 public Integer countMovies(){  
 return *movies*.size();  
 }  
  
 @POST  
 @Produces(MediaType.*TEXT\_PLAIN*)  
 @Consumes(MediaType.*TEXT\_PLAIN*)  
 public Response createMovie(String newMovie){  
 *movies*.add(newMovie);  
 return Response.*ok*(*movies*).build();  
 }  
  
 @PUT  
 @Path("{movieToUpdate}")  
 @Produces(MediaType.*TEXT\_PLAIN*)  
 @Consumes(MediaType.*TEXT\_PLAIN*)  
 public Response updateMovie(  
 @PathParam("movieToUpdate") String movieToUpdate,  
 @QueryParam("movie") String updateMovie) {  
  
 *movies* = *movies*.stream().map(movie -> {  
 if (movie.equals(movieToUpdate)) {  
 return updateMovie; // Update the movie if it matches  
 }  
 return movie; // Return the original movie if it doesn't match  
 }).collect(Collectors.*toList*());  
  
 return Response.*ok*(*movies*).build();  
 }  
  
 @DELETE  
 @Path("{movieToDelete}")  
 @Produces(MediaType.*TEXT\_PLAIN*)  
 public Response deleteMovie(@PathParam("movieToDelete") String movieToDelete) {  
 boolean removed = *movies*.removeIf(movie -> movie.equals(movieToDelete));  
  
 if (removed) {  
 return Response.*ok*("Movie deleted successfully: " + movieToDelete).build();  
 } else {  
 return Response.*status*(Response.Status.*NOT\_FOUND*)  
 .entity("Movie not found: " + movieToDelete)  
 .build();  
 }  
 }  
  
}

**Now create class Movie:**

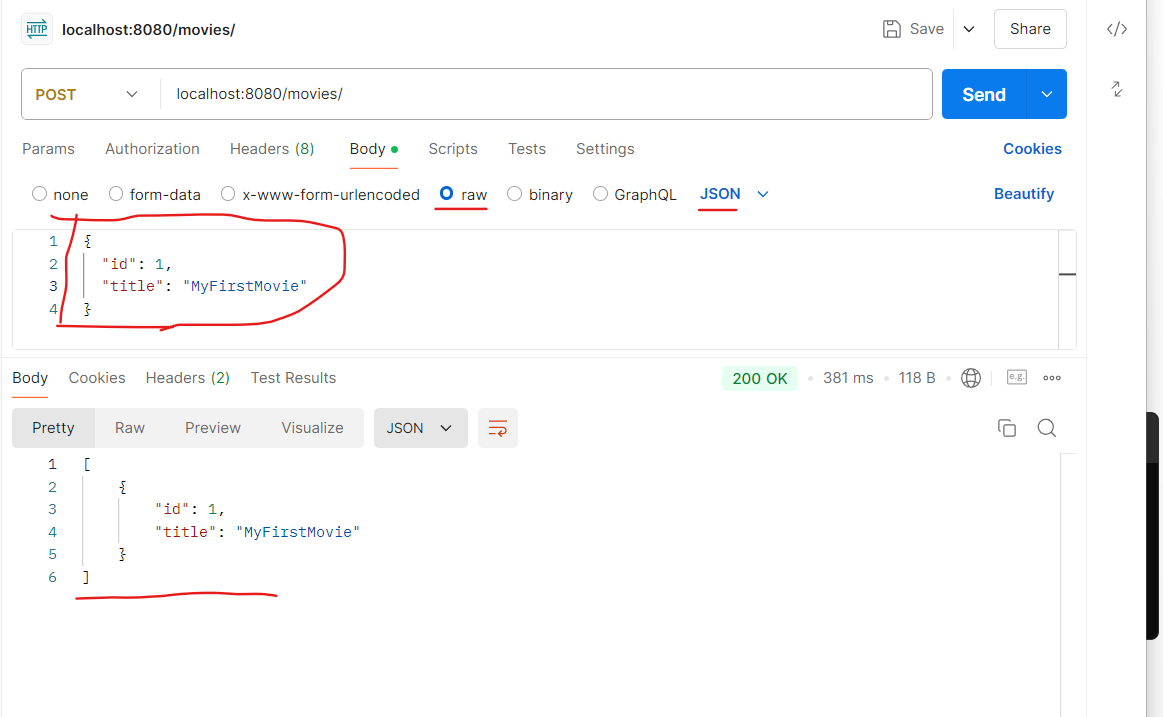


package com.ug;  
  
public class Movie {  
 private Long id;  
 private String title;  
 public Long getId() {  
 return id;  
 }  
 public void setId(Long id) {  
 this.id = id;  
 }  
 public String getTitle() {  
 return title;  
 }  
 public void setTitle(String title) {  
 this.title = title;  
 }  
}

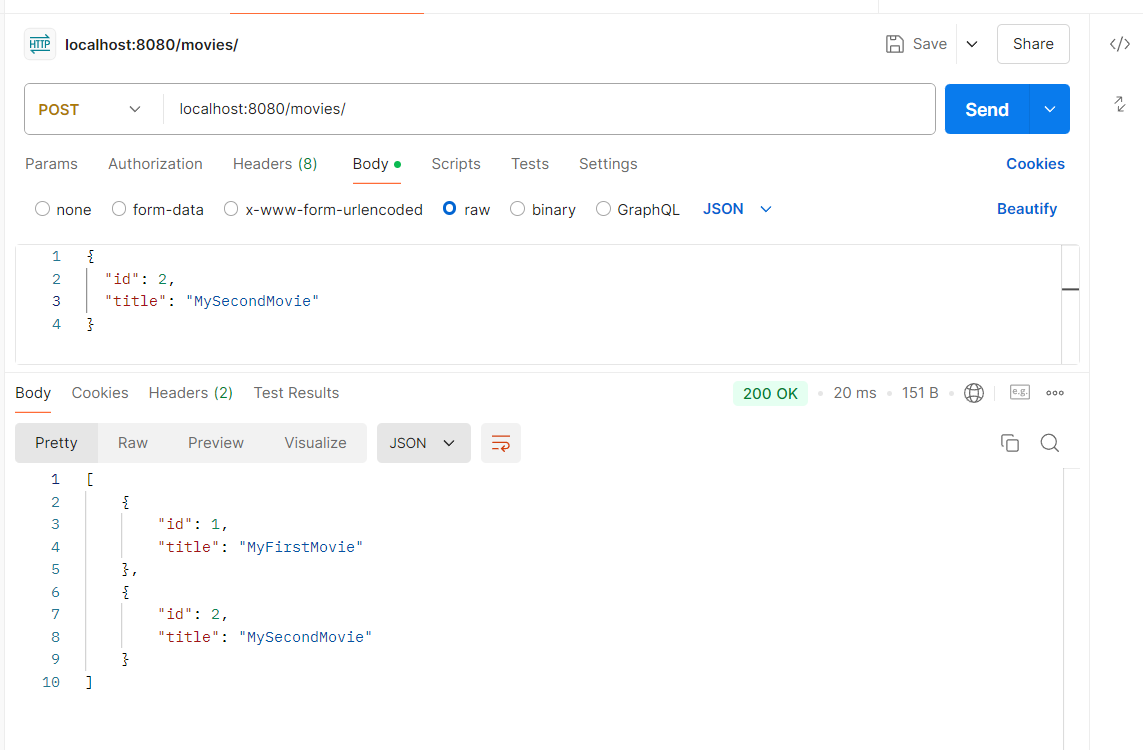
**An also Modify MovieResource Class:**

package com.ug;  
  
import jakarta.ws.rs.\*;  
import jakarta.ws.rs.core.MediaType;  
import jakarta.ws.rs.core.Response;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Optional;  
import java.util.stream.Collectors;  
  
@Path("/movies")  
public class MovieResource {  
 public static List<Movie> *movies* = new ArrayList<>();  
  
 @GET  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Response getMovies() {  
 return Response.*ok*(*movies*).build();  
 }  
  
 @GET  
 @Path("/size")  
 @Produces(MediaType.*TEXT\_PLAIN*)  
 public Response countMovies() {  
 return Response.*ok*(*movies*.size()).build();  
 }  
  
 @POST  
 @Consumes(MediaType.*APPLICATION\_JSON*)  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Response createMovie(Movie newMovie) {  
 *movies*.add(newMovie);  
 return Response.*ok*(*movies*).build();  
 }  
  
 @PUT  
 @Path("{id}")  
 @Consumes(MediaType.*APPLICATION\_JSON*)  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Response updateMovie(@PathParam("id") Long id, Movie updateMovie) {  
 *movies* = *movies*.stream().map(movie -> {  
 if (movie.getId().equals(id)) {  
 movie.setTitle(updateMovie.getTitle());  
 }  
 return movie;  
 }).collect(Collectors.*toList*());  
  
 return Response.*ok*(*movies*).build();  
 }  
  
 @DELETE  
 @Path("{id}")  
 @Produces(MediaType.*TEXT\_PLAIN*)  
 public Response deleteMovie(@PathParam("id") Long id) {  
 boolean removed = *movies*.removeIf(movie -> movie.getId().equals(id));  
  
 if (removed) {  
 return Response.*ok*("Movie deleted successfully: " + id).build();  
 } else {  
 return Response.*status*(Response.Status.*NOT\_FOUND*)  
 .entity("Movie not found with id: " + id)  
 .build();  
 }  
 }  
}

Post:

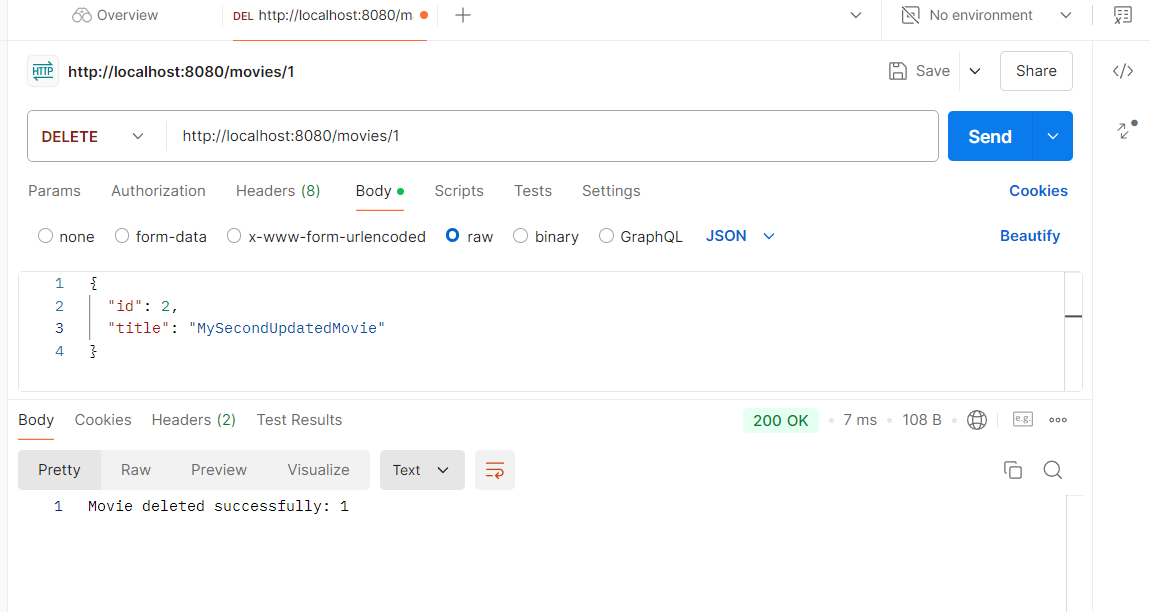


Post:



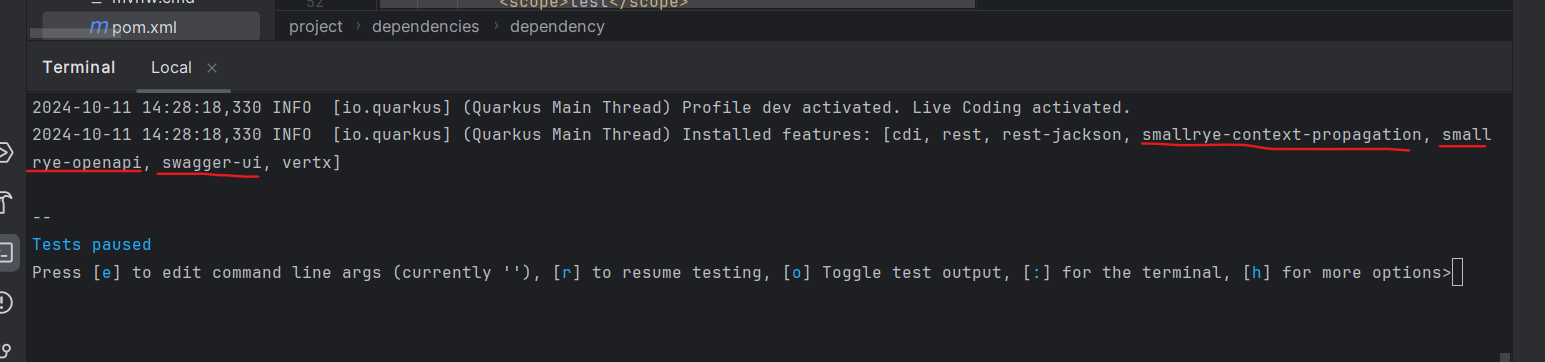
PUT:



**Delete:**  


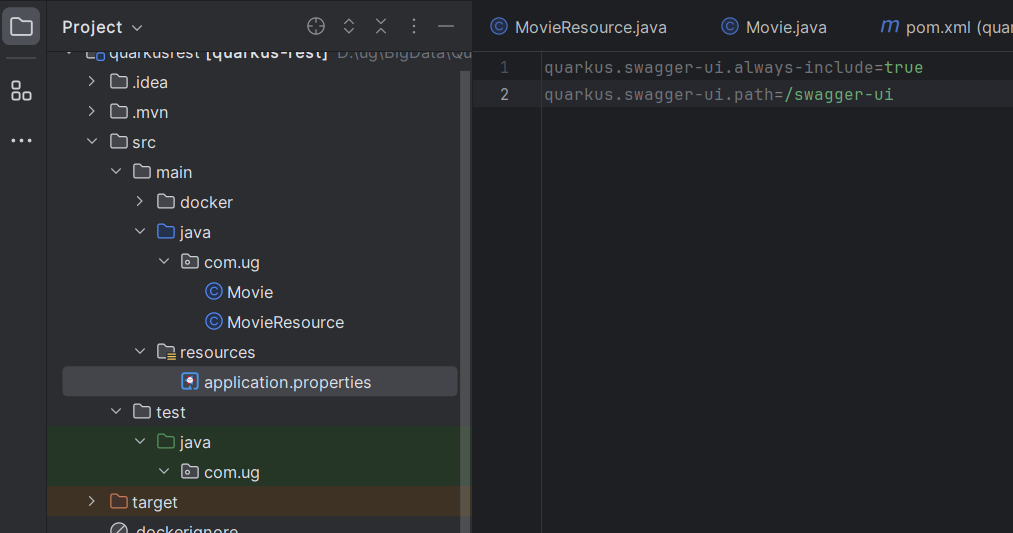
**Add the following Dependency:**

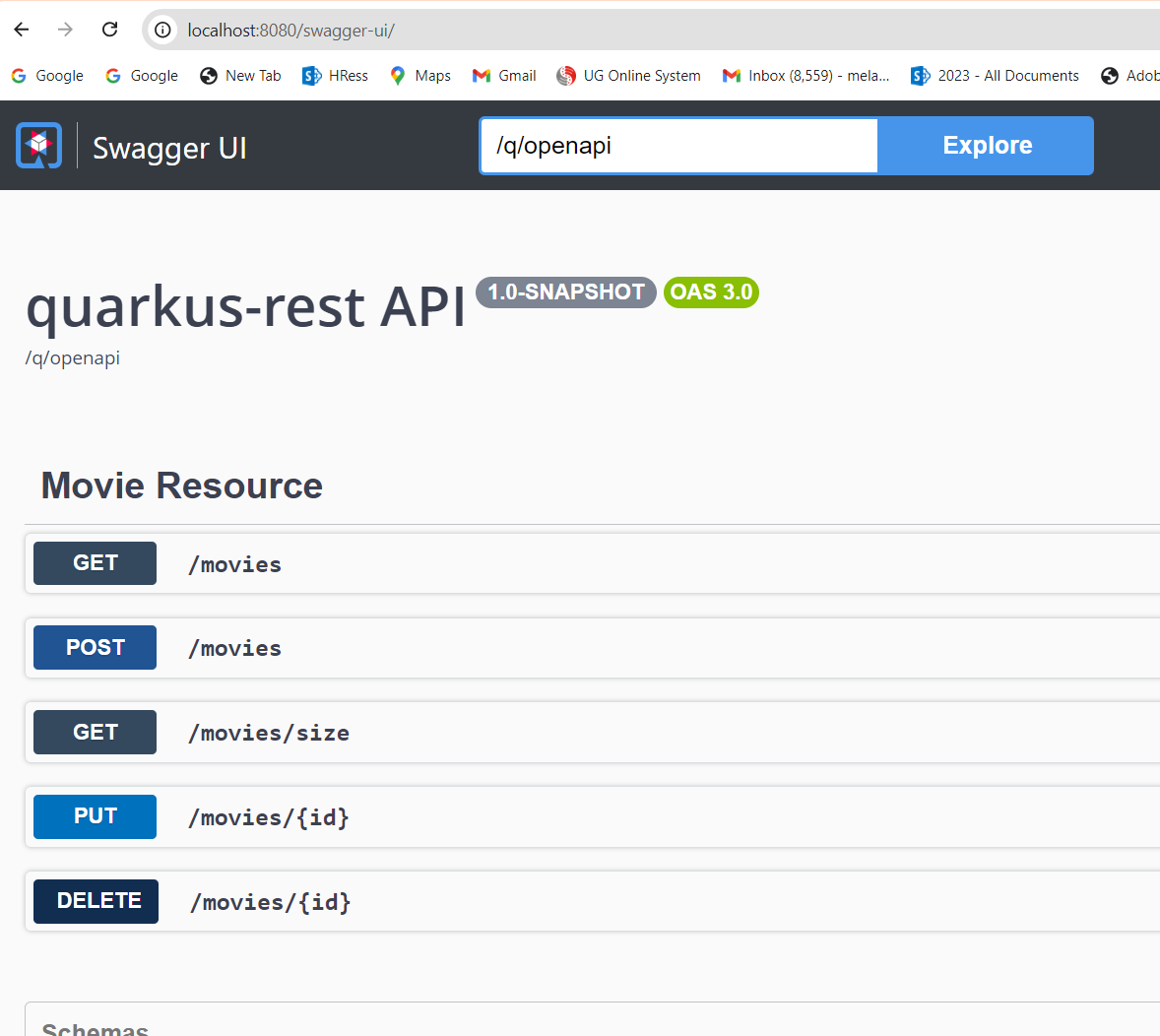
<dependency>  
 <groupId>io.quarkus</groupId>  
 <artifactId>quarkus-smallrye-openapi</artifactId>  
</dependency>

**And look in the terminal:  
**

**SWAGGER-UI**

**Modify application.properties File:**

****

****