



Universidad Nacional Autónoma de México
Facultad de Contaduría y Administración



Licenciatura: Informática

Materia: Bases de datos

Unidad 5:
Construcción

Actividad Complementaria 2:
MariaBD 10.6

Nombre: Karen Gómez León

Grupo: 8491

Fecha: 05/11/2021

Unidad 5. Actividad complementaria 2

a) Investiga ¿Qué es un API? y ¿Qué es un microservicio?

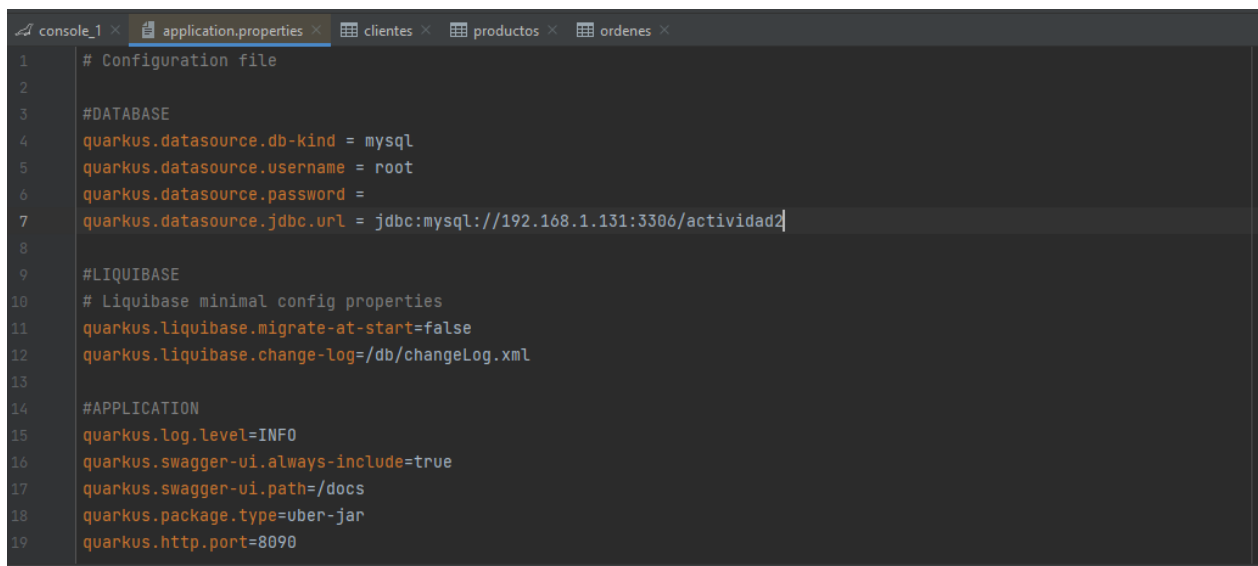
Application Programming Interfaces o interfaz de programación de aplicaciones en español, es una especificación que indica como un módulo se comunica o interactúa con otro de acuerdo a los permisos otorgados y pueden tener una sola función o varias, pero estas no son visibles para el usuario, pueden ser de uso privado, abierto o público, y locales o remotas, lo cual facilita que las aplicaciones no se creen desde cero. Xataka (2019) lo define como un "conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas".

Microservicio: De acuerdo con Red Hat "Los microservicios son tanto un estilo de arquitectura como un modo de programar software... son elementos independientes que funcionan en conjunto para llevar a cabo las mismas tareas". Lo cual permite tratar el software como un conjunto de elementos que conforman a todo un proceso o sistema. Este método de creación de software es considerado cuando se desea mantener compatibilidad entre diferentes plataformas y cuando se desea crear diferentes servicios individuales uno por uno conforme se vaya necesitando, de acuerdo con OpenWebinars HTTP/REST, JSON o Protobuf son los métodos de integración de microservicios más usados los cuales permiten la comunicación de estos pequeños servicios individuales.

b) Investiga que es Swagger. ¿Para qué se utiliza?

Es una especificación OpenAPI de código abierto que ayuda a controlar las diferentes capacidades de las interfaces API, con lo cual es posible que diferentes usuarios interactúen con el proceso de desarrollo a través de la visualización de las interfaces utilizadas y su documentación. De acuerdo con Digital Guide Ionos, Swagger es la tecnología más popular para realizar las documentaciones de API (REST) que utiliza código abierto Linux (Open API initiative) lo que le permite relacionarse ampliamente con GitHub (en donde se puede encontrar generadores de código para la mayoría de los lenguajes de programación) y que además permite realizar pruebas pensadas especialmente para el sistema documentado. Swagger es un conjunto de herramientas usadas para generar documentación ordenada y comprensible lo que hace posible que las interfaces sean usadas por los desarrolladores y la comunidad.

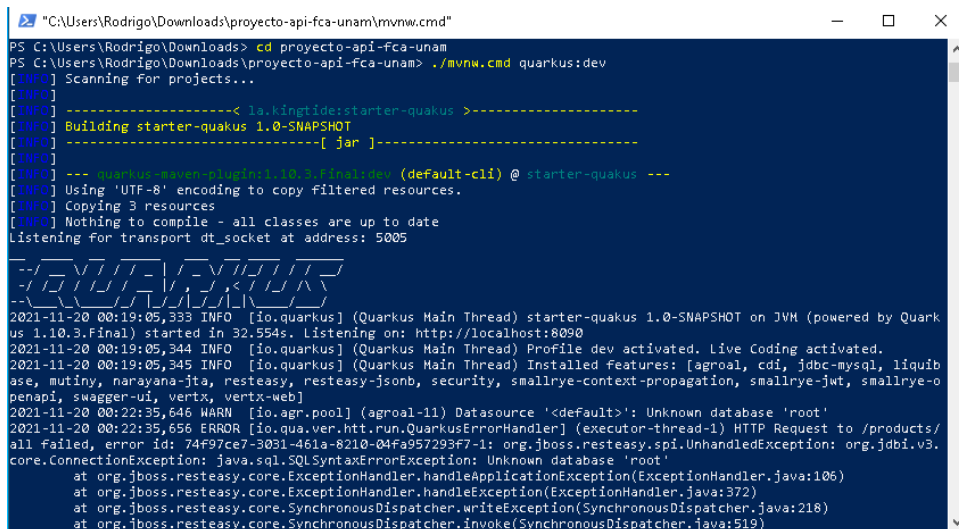
c) Descarga el proyecto del API que se encuentra en Github. En el foro se indicará la dirección de Github.



```
1 # Configuration file
2
3 #DATABASE
4 quarkus.datasource.db-kind = mysql
5 quarkus.datasource.username = root
6 quarkus.datasource.password =
7 quarkus.datasource.jdbc.url = jdbc:mysql://192.168.1.131:3306/actividad2
8
9 #LIQUIBASE
10 # Liquibase minimal config properties
11 quarkus.liquibase.migrate-at-start=false
12 quarkus.liquibase.change-log=db/changeLog.xml
13
14 #APPLICATION
15 quarkus.log.level=INFO
16 quarkus.swagger-ui.always-include=true
17 quarkus.swagger-ui.path=/docs
18 quarkus.package.type=uber-jar
19 quarkus.http.port=8090
```

Imagen 1. Modificación de las credenciales en el archivo "application.properties". Para poder lograr la pantalla mostrada, fue necesario la instalación de Java Y Maven, la creación de las variables de ambiente en "Editar la variable de usuario" en el entorno, se modificaron las variables path y las rutas en "Editar variable de entorno" de Windows

(de acuerdo a las instrucciones mostradas en el repositorio <https://github.com/tiempor3al/proyecto-api>), además se verificó la instalación en consola PowerShell con los comandos “java -version” y “mvn --version”. A continuación, se descargó el proyecto contenido en el repositorio y se modificaron las credenciales de la base de datos (actividad2), el usuario ingresado y contraseña corresponden a los creados para la base de datos de MariaDB (root y sin contraseña), mientras que la dirección ip (IPv4) se encontró utilizando el comando “ipconfig” en consola y el puerto 3306 es el estándar para MariaDB/MySQL. Seguido de la compilación del proyecto como se muestra en la siguiente pantalla.



```
"C:\Users\Rodrigo\Downloads\proyecto-api-fca-unam\mvnw.cmd"
PS C:\Users\Rodrigo\Downloads> cd proyecto-api-fca-unam
PS C:\Users\Rodrigo\Downloads\proyecto-api-fca-unam> ./mvnw.cmd quarkus:dev
[INFO] Scanning for projects...
[INFO]
[INFO] -----< 1a.kingtide:starter-quakus >-----
[INFO] Building starter-quakus 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- quarkus:dev:1.0.3.Final:dev (default-cli) @ starter-quakus ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 3 resources
[INFO] Nothing to compile - all classes are up to date
[INFO] Listening for transport dt_socket at address: 5005

2021-11-20 00:19:05,333 INFO [io.quarkus] (Quarkus Main Thread) starter-quakus 1.0-SNAPSHOT on JVM (powered by Quarkus 1.0.3.Final) started in 32.554s. Listening on: http://localhost:8090
2021-11-20 00:19:05,344 INFO [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding activated.
2021-11-20 00:19:05,345 INFO [io.quarkus] (Quarkus Main Thread) Installed features: [agroal, cdi, jdbc-mysql, liquibase, mutiny, narayana-jta, resteasy, resteasy-jsonb, security, smallrye-context-propagation, smallrye-jwt, smallrye-openapi, swagger-ui, vertx, vertx-web]
2021-11-20 00:22:35,646 WARN [io.agroal.pool] (agroal-11) DataSource '<default>': Unknown database 'root'
2021-11-20 00:22:35,656 ERROR [io.qua.ver.htt.run.QuarkusExceptionHandler] (executor-thread-1) HTTP Request to /products/all failed, error id: 74f97ce7-3031-461a-8210-04fa957293f7-1: org.jboss.resteasy.spi.UnhandledException: org.jboss.core.ConnectionException: java.sql.SQLException: Unknown database 'root'
    at org.jboss.resteasy.core.ExceptionHandler.handleApplicationException(ExceptionHandler.java:106)
    at org.jboss.resteasy.core.ExceptionHandler.handleException(ExceptionHandler.java:372)
    at org.jboss.resteasy.core.SynchronousDispatcher.writeException(SynchronousDispatcher.java:218)
    at org.jboss.resteasy.core.SynchronousDispatcher.invoke(SynchronousDispatcher.java:519)
```

Imagen 2. Compilación del proyecto descargado en el directorio “Downloads”, para la descarga de las librerías requeridas para el proyecto, ejecutándose la aplicación web en modo desarrollo en el puerto 8090. De acuerdo con las instrucciones del repositorio este modo permite la modificación de archivos y re-compilación durante ejecución.

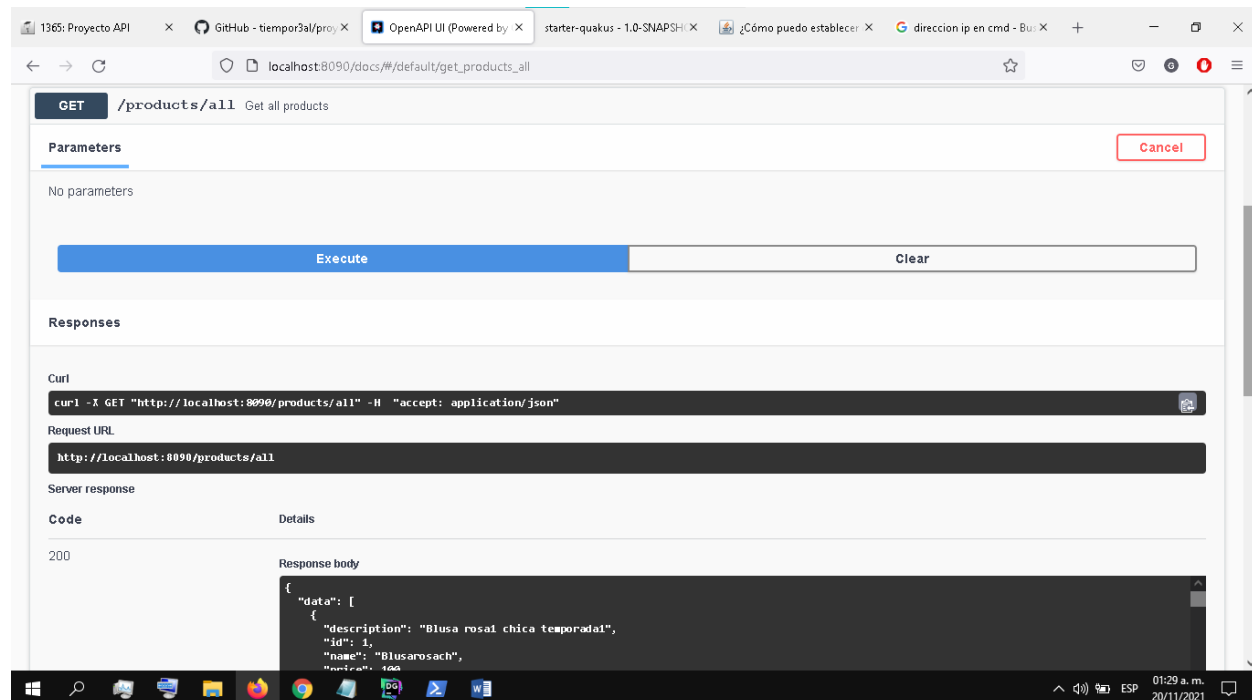


Imagen. Ejecución del endpoint GET en Quarkus ProjectAPI. Para la ejecución de actividades fue necesario ejecutar y navegar en <http://localhost:8090> y en Quarkus (Swagger) <http://localhost:8090/docs> .

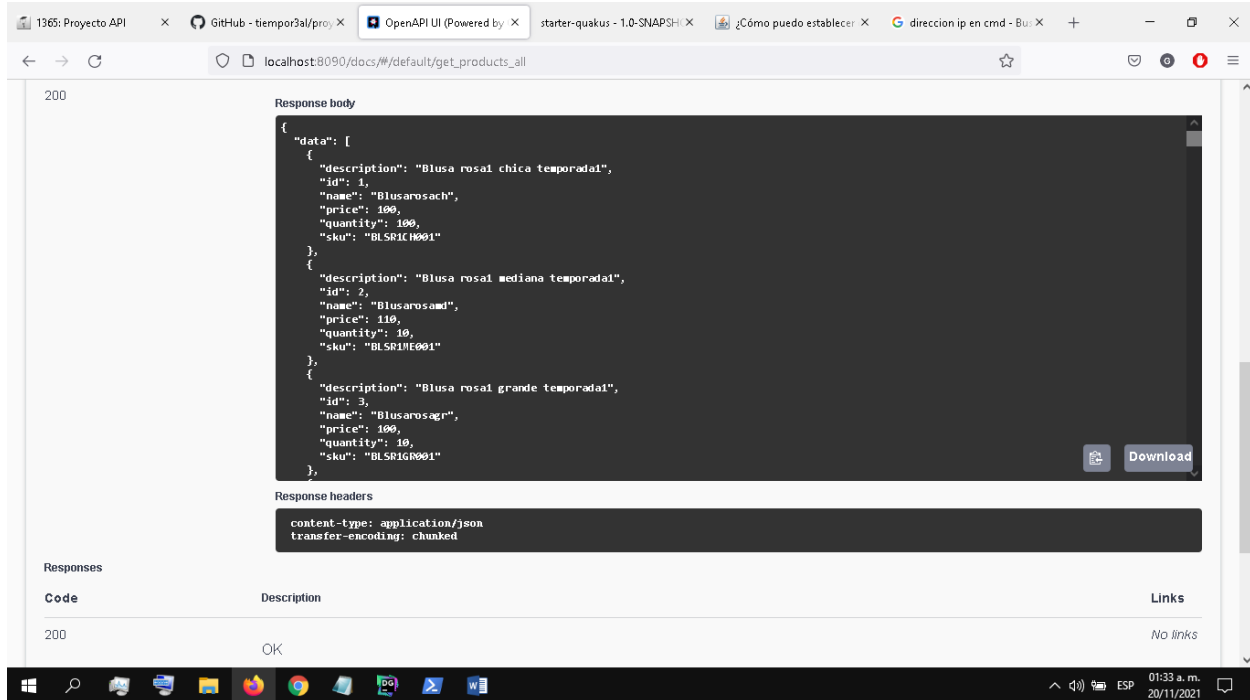


Imagen. Respuestas mostradas por el servidor, en donde se puede ver que es devuelta la información contenida en la tabla “productos” de la base de datos “actividad2”.

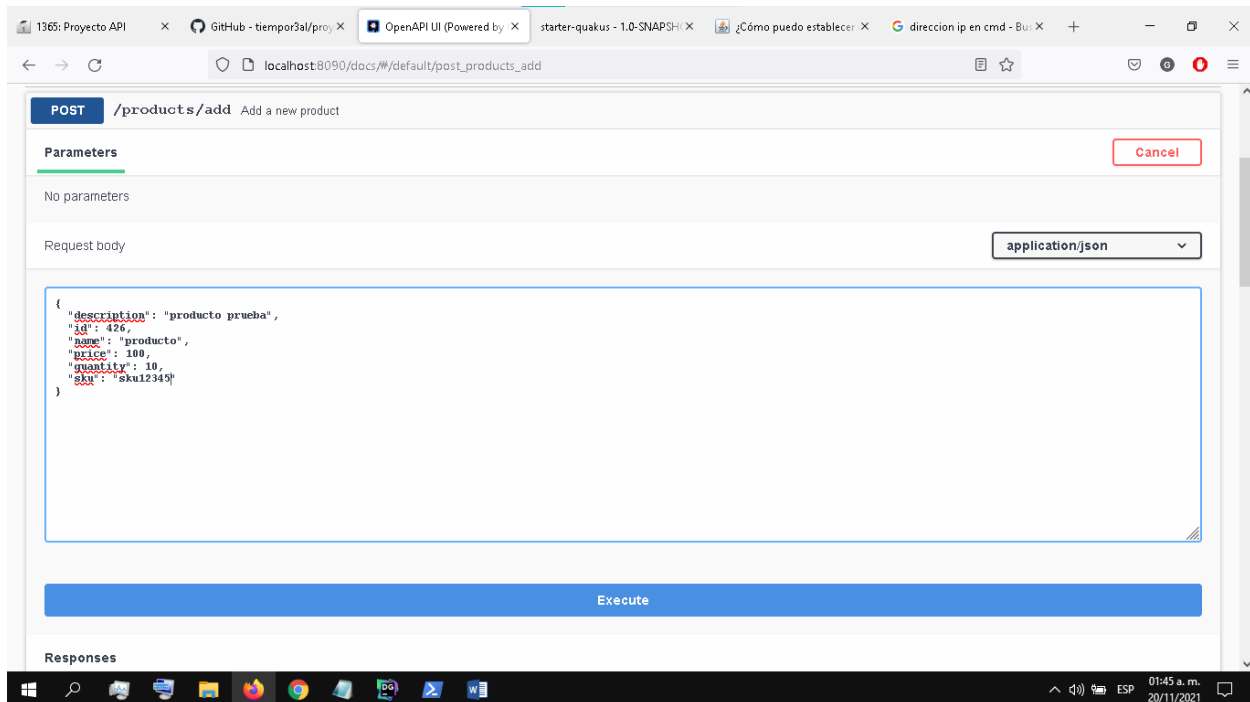
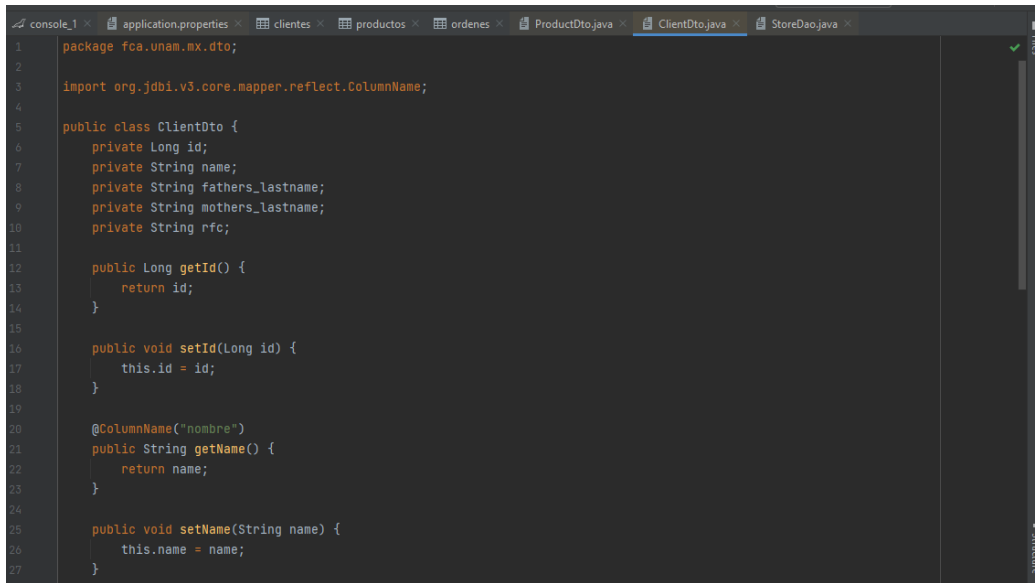


Imagen. Ejecución del endpoint POST, en donde se puede ver que se está agregando un nuevo producto a la tabla productos “id” 426, en formato JSON (content-type: application/json).

- d) Completa los endpoints que hacen falta. Ejecuta el proyecto. Crea el endpoint /clients/all que muestre todos los clientes. Deberás tomar como referencia el endpoint GET /products/all. Para ello deberás:
1. Crear el archivo ClientDto.java en la carpeta dto.



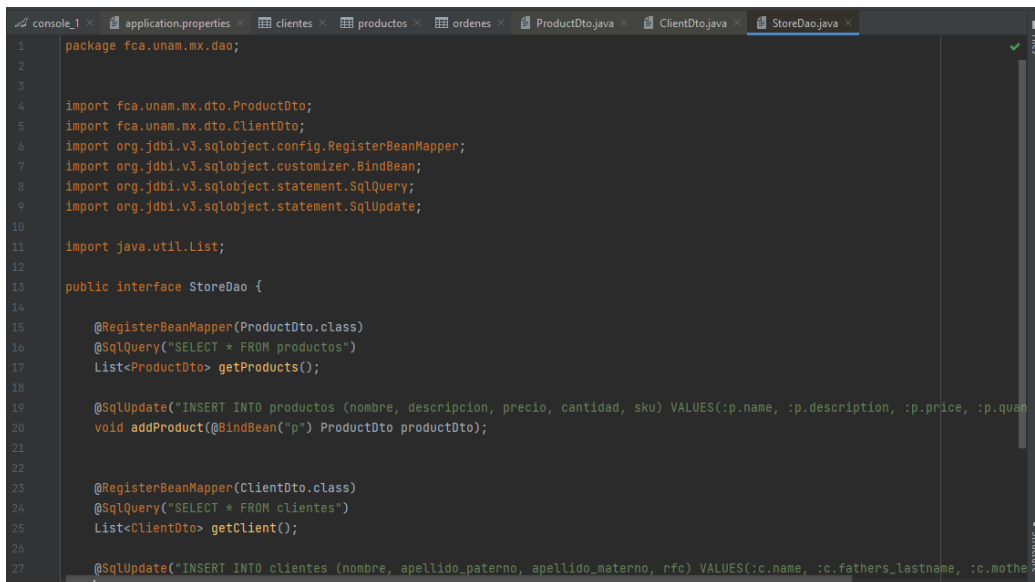
```
1 package fca.unam.mx.dto;
2
3 import org.jdbi.v3.core.mapper.reflect.ColumnName;
4
5 public class ClientDto {
6     private Long id;
7     private String name;
8     private String fathers_lastname;
9     private String mothers_lastname;
10    private String rfc;
11
12    public Long getId() {
13        return id;
14    }
15
16    public void setId(Long id) {
17        this.id = id;
18    }
19
20    @ColumnName("nombre")
21    public String getName() {
22        return name;
23    }
24
25    public void setName(String name) {
26        this.name = name;
27    }
28 }
```

Imagen. Creación del archivo ClientDto de acuerdo a la tabla clientes, creado de manera similar a ProductDto.

2. Modificar el archivo StoreDao.java para incluir el nuevo query getClients().

Deberás mapear ClienteDto de manera similar a:

```
@RegisterBeanMapper(ProductDto.class)
@SqlQuery("SELECT * FROM productos")
List<ProductDto> getProducts();
```



```
1 package fca.unam.mx.dao;
2
3
4 import fca.unam.mx.dto.ProductDto;
5 import fca.unam.mx.dto.ClientDto;
6 import org.jdbi.v3.sqlobject.config.RegisterBeanMapper;
7 import org.jdbi.v3.sqlobject.customizer.BindBean;
8 import org.jdbi.v3.sqlobject.statement.SqlQuery;
9 import org.jdbi.v3.sqlobject.statement.SqlUpdate;
10
11 import java.util.List;
12
13 public interface StoreDao {
14
15     @RegisterBeanMapper(ProductDto.class)
16     @SqlQuery("SELECT * FROM productos")
17     List<ProductDto> getProducts();
18
19     @SqlUpdate("INSERT INTO productos (nombre, descripcion, precio, cantidad, sku) VALUES(:p.name, :p.description, :p.price, :p.quantity, :p.sku)")
20     void addProduct(@BindBean("p") ProductDto productDto);
21
22
23     @RegisterBeanMapper(ClientDto.class)
24     @SqlQuery("SELECT * FROM clientes")
25     List<ClientDto> getClients();
26
27     @SqlUpdate("INSERT INTO clientes (nombre, apellido_paterno, apellido_materno, rfc) VALUES(:c.name, :c.fathers_lastname, :c.mothers_lastname, :c.rfc)")
28     void insertClient(@BindBean("c") ClientDto clientDto);
29 }
```

Imagen. Modificación del archivo StoreDao en donde se puede ver que el query para la tabla de clientes se agregó de acuerdo a los datos propios de la tabla de clientes.

3. ¿Para qué sirve la directiva @RegisterBeanMapper (https://jdbi.org/#_beanmapper)?

Rastreo o recuperación de registros de una tabla a partir de un identificador (id) recuperando los campos o columnas de la tabla indicada.

4. ¿Qué deberá devolver el método getClients?

De manera semejante a getProduct, mostrará los registros de la tabla "clientes" en formato JSON o en otras palabras una lista de los objetos o beans de la tabla de clientes agrupados de acuerdo a los campos contenidos en la tabla.

5. Deberás modificar el archivo StoreDal.java y crear el método getClients() de manera similar a:

Do public ResponseDto<List<ProductDto>>getProducts() {

```
ResponseDto responseDto = new ResponseDto<List<ProductDto>>();
responseDto.setSuccess(true);
Jdbi jdbi = jdbiService.getInstance();
var products = jdbi.withExtension(StoreDao.class, dao -> dao.getProducts());
responseDto.setData(products);
return responseDto;
}
```

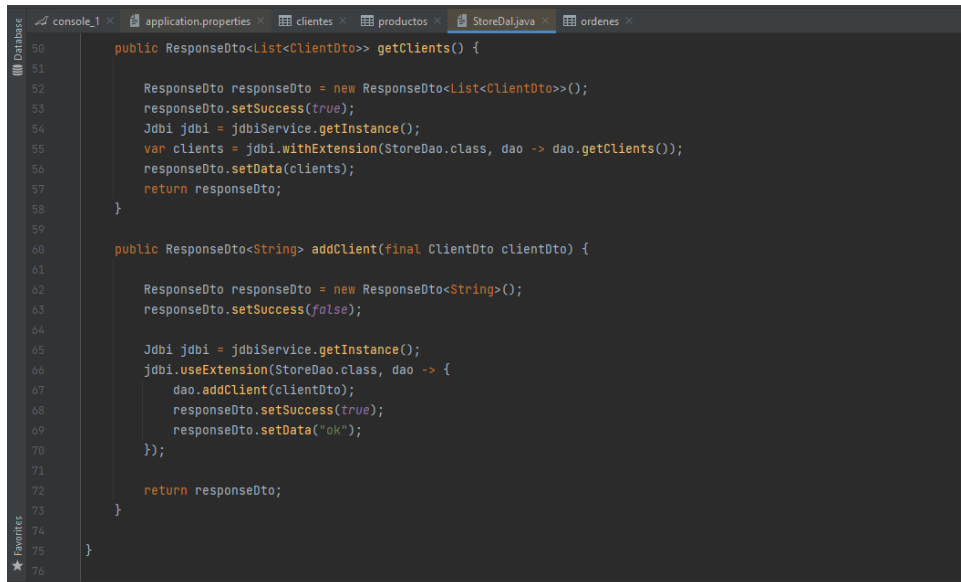


Imagen. Creación de los métodos get y add en la tabla clientes de manera similar a los métodos mostrados en el archivo para la tabla productos.

Deberás crear el archivo ClientsResource.java en la carpeta src/main/java/resources junto a ProductsResources.java. El archivo ClientsResource.java deberá contener un método getClients() similar a:

```
@GET
@Path("/all")
@Produces(MediaType.APPLICATION_JSON)
@Operation(summary = "Get all products")
@APIResponses(value = {@APIResponse(responseCode = "200", content = @Content(mediaType =
MediaType.APPLICATION_JSON))})
public Response getProducts() {
    var responseDto = storeDal.getProducts();
    return Response.ok(responseDto).build();
}
```

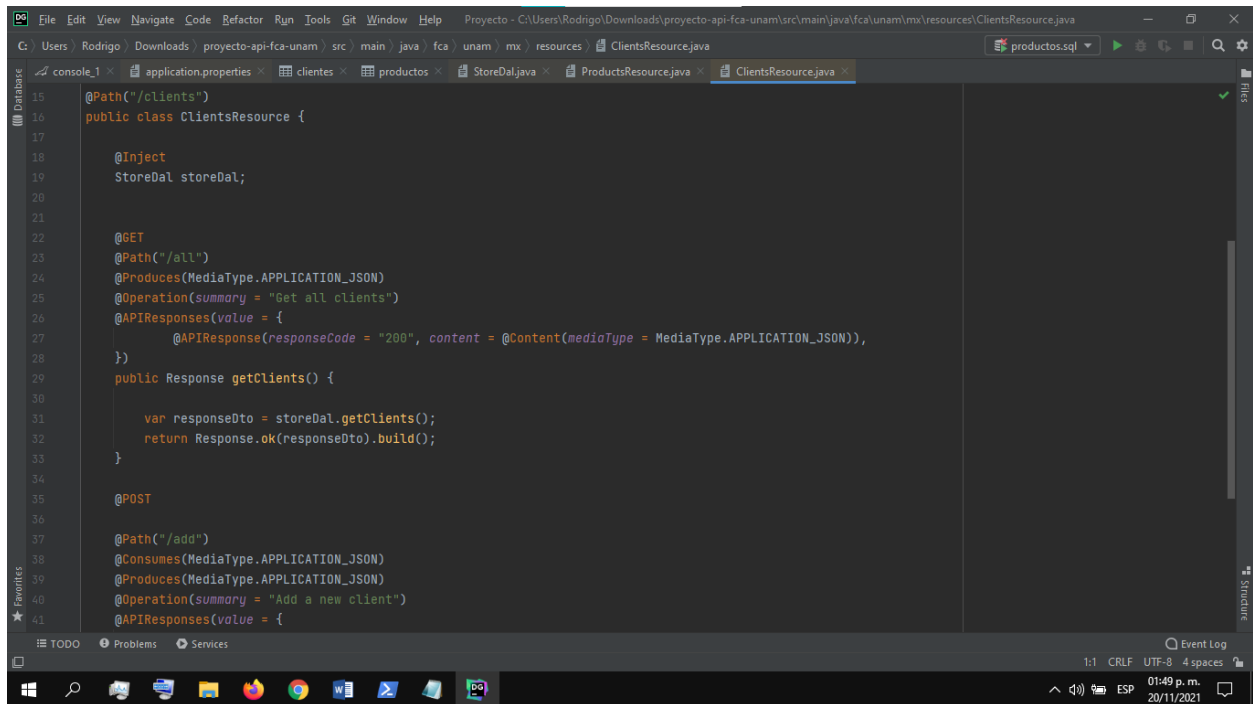


Imagen. Creación del archivo ClientsResource.java, creado modificando el archivo ProductsResources.java.

6. Compila y ejecución del proyecto.

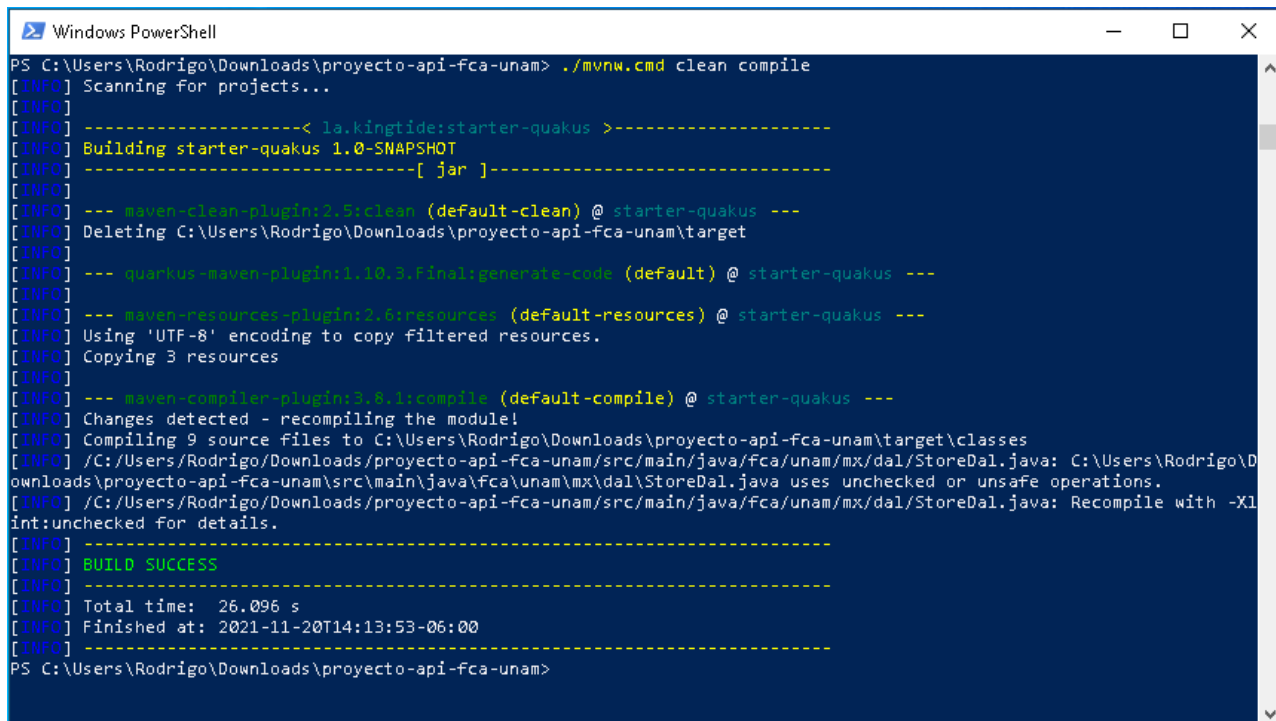


Imagen. Compilación del comando: ./mvnw.cmd clean compile en PowerShell en donde se puede ver que la compilación fue exitosa.

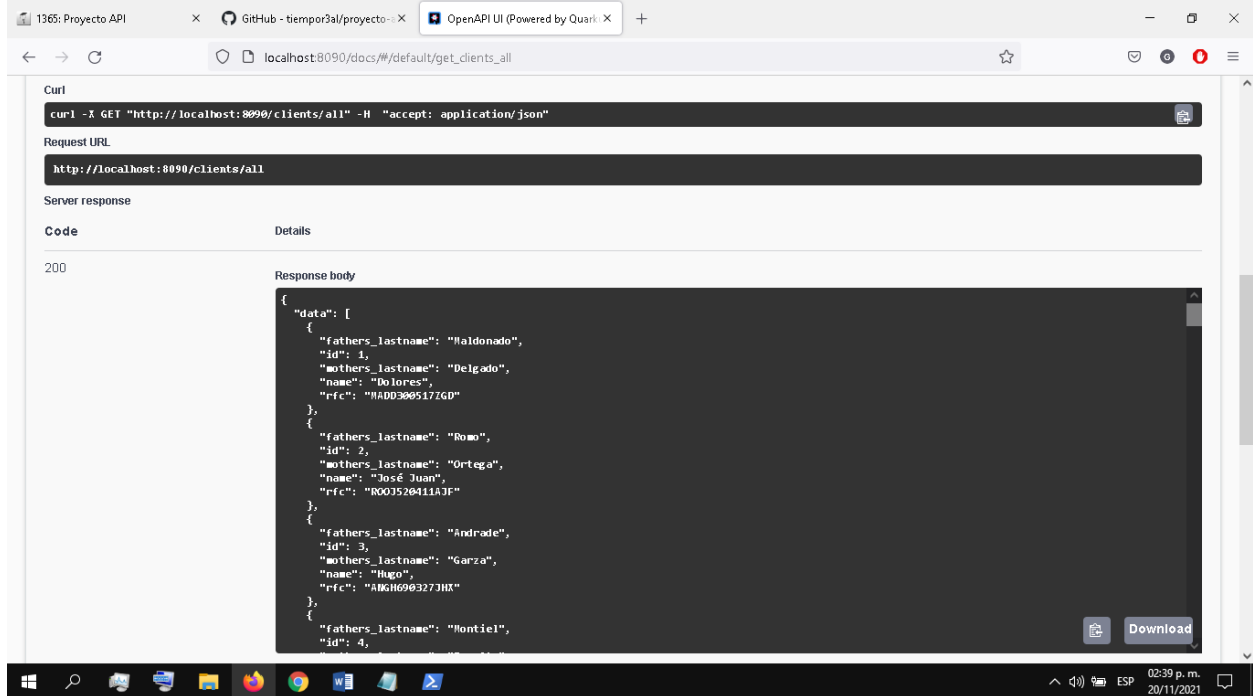


Imagen. Ejecución exitosa del Endpoint GET/clients/all en formato JSON.

- e) Crea un repositorio en Github y sube tus cambios.

Se creó una cuenta en GitHub y repositorio "actividad_endpoints" con visibilidad Publica, la dirección es: https://github.com/golekaren/actividad_endpoints

- f) En un documento en PDF, anexa las capturas de pantalla del proyecto y la liga a tu repositorio. Sube
- g) Anota tus respuestas y las capturas de pantalla en un documento y sube el archivo en formato PDF a la plataforma

Referencias

Abril, E. (19 de abril de 2016). *Qué son Microservicios y ejemplos reales de uso*. OpenWebinars. <https://openwebinars.net/blog/microservicios-que-son/>

Crear un repositorio. (s.f.). GitHub Docs. <https://docs.github.com/es/get-started/quickstart/create-a-repo#create-a-repository>

FCA (2017). *Bases de Datos. Apunte electrónico*. Facultad de Contaduría y Administración. Universidad Nacional Autónoma de México.

Fernández, Y. (23 de agosto de 2019). *API: qué es y para qué sirve*. XATAKA. <https://www.xataka.com/basics/api-que-sirve>

¿Qué son y para qué sirven los microservicios? (s.f.). RedHat. <https://www.redhat.com/es/topics/microservices>

Swagger: más comodidad en el desarrollo de API. (17 de febrero de 2021). Digital Guide IONOS. <https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/que-es-swagger/>

Table of contents. BeanMapper. (s.f.). JDBI. https://jdbi.org/#_beanmapper