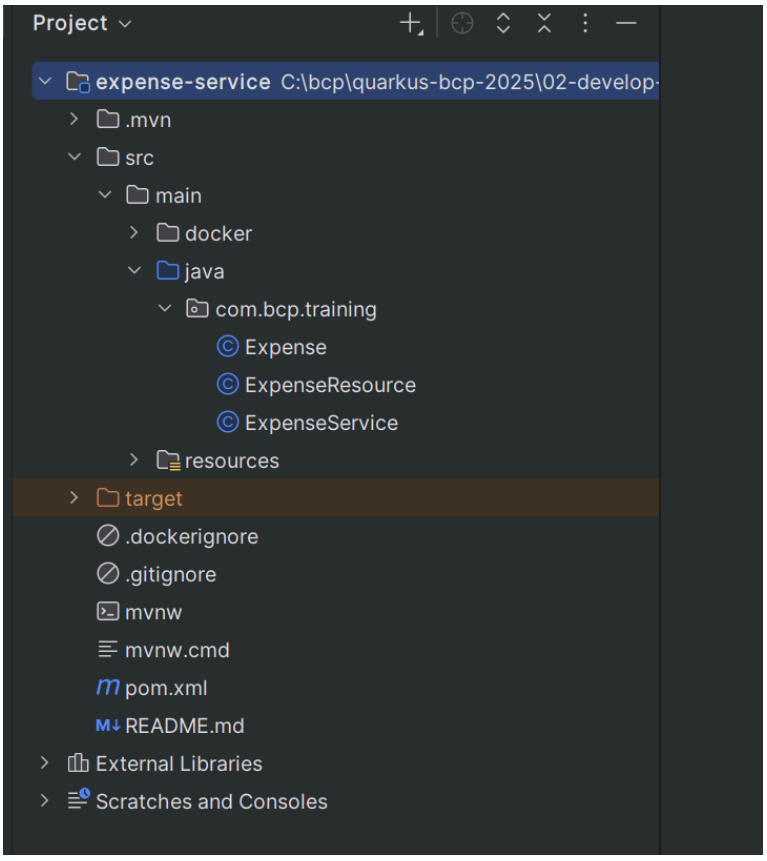
LAB 3: QUARKUS REST

Autor: José Díaz

Github Repo: https://github.com/joedayz/quarkus-bcp-2025.git

1. Abre el proyecto **expense-service**.
   1.1. Abre **tu IDE favorito**, luego haz clic en **Archivo > Abrir carpeta** y navega hasta el directorio:

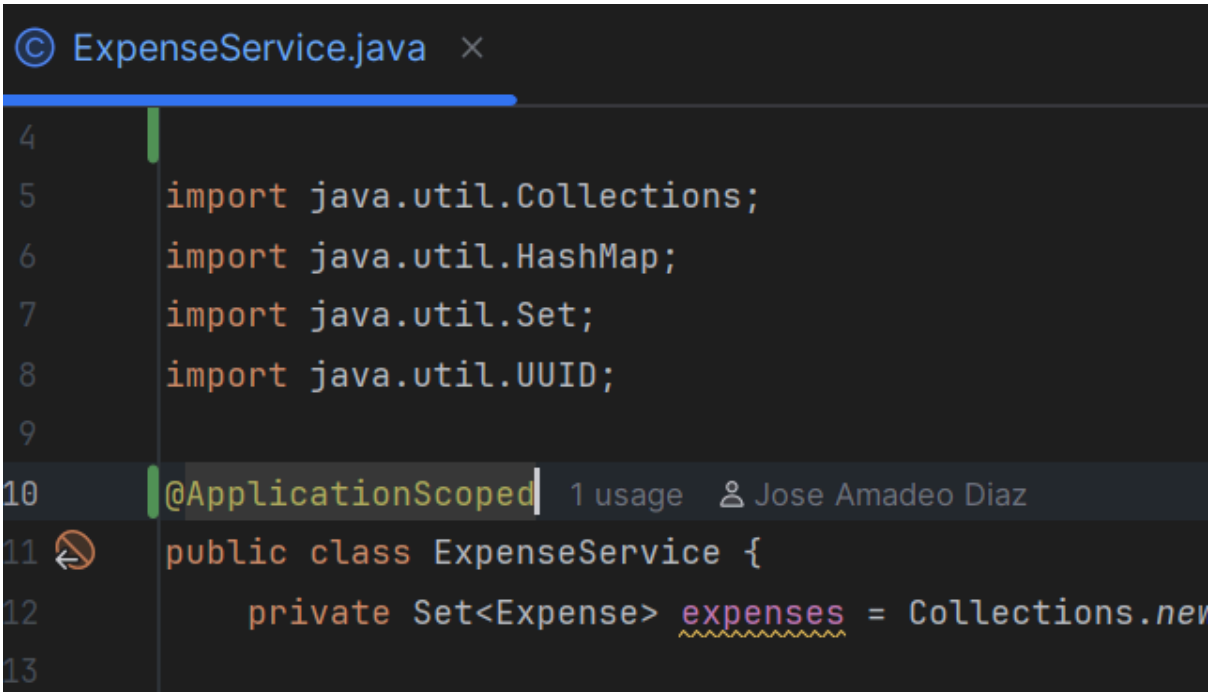


   1.2 Ejecuta **mvn quarkus:dev**

2. Convierte la clase **ExpenseService** en un bean CDI.

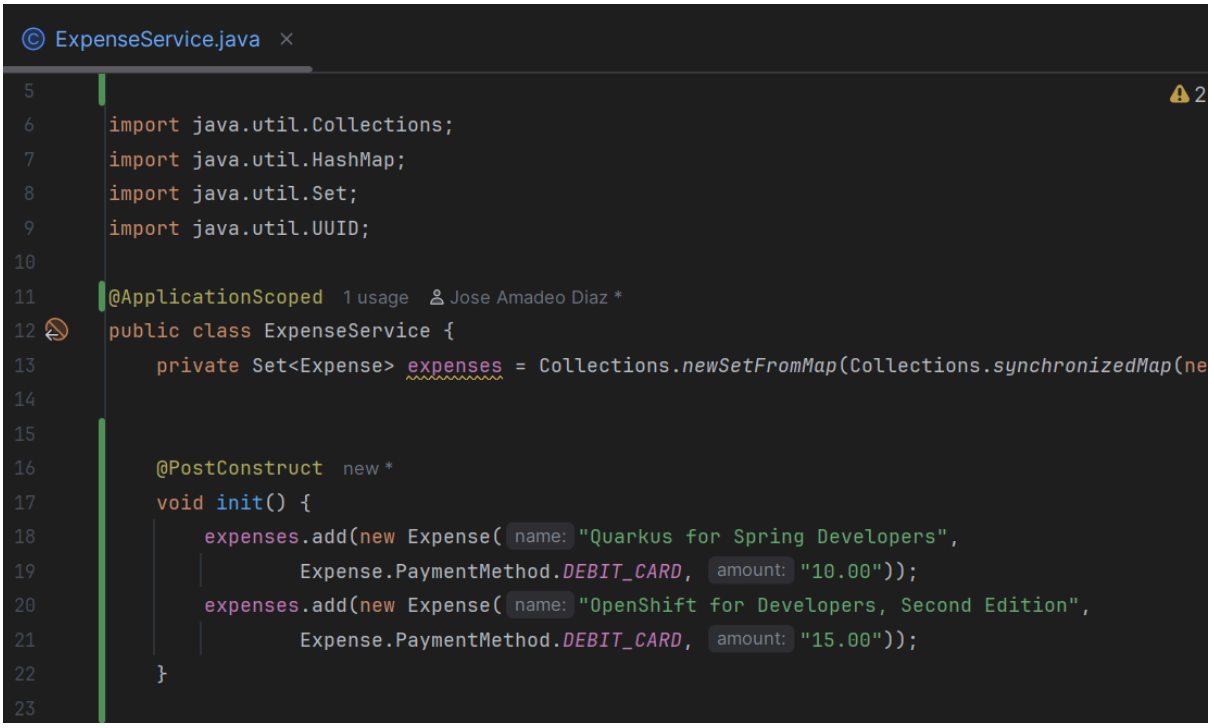
   2.1. Abre el archivo **src/main/java/com/bcp/training/ExpenseService.java**.

   2.2. Anota la clase con la anotación **@ApplicationScoped**.



2.3 Crea el método **init** que use la anotación **@PostConstruct** para inyectar datos de ejemplo en el bean al inicio de la aplicación.

```
@PostConstruct
void init() {
   expenses.add(new Expense("Quarkus for Spring Developers",
           Expense.PaymentMethod.DEBIT_CARD, "10.00"));
   expenses.add(new Expense("OpenShift for Developers, Second Edition",
           Expense.PaymentMethod.DEBIT_CARD, "15.00"));
}
```



3. Implementar los siguientes REST endpoints

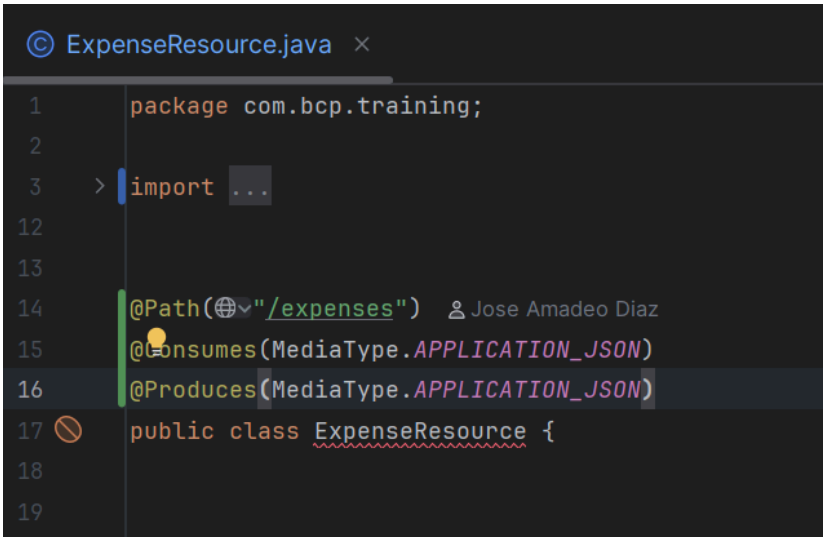
• **GET /expenses**: lista objetos **Expense**.
• **POST /expenses**: crea un objeto **Expense**.

• **PUT /expenses**: actualiza un objeto **Expense**.
• **DELETE /expenses/{UUID}**: elimina el objeto **Expense** especificado.

Todos los endpoints usan el tipo de medio **JSON**.

3.1. Abre el archivo **src/main/java/com/bcp/training/ExpenseResource.java**.
3.2. Configura **ExpenseResource** para atender en la ruta **/expenses**. Luego, configura la clase para usar **JSON** como formato de serialización.

```
@Path("/expenses")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public class ExpenseResource {
```



3.3 Inyecta el expenseService CDI bean

```
@Path("/expenses")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public class ExpenseResource {

    @Inject
    public ExpenseService expenseService;
```

3.4 Implementa los endpoints de la aplicación

```
@Path("/expenses")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public class ExpenseResource {

    @Inject
    public ExpenseService expenseService;


    @GET
    public Set<Expense> list() {
        return expenseService.list();
    }
    @POST
```

```
    public Expense create(Expense expense) {
        return expenseService.create(expense);
    }
    @DELETE
    @Path("{uuid}")
    public Set<Expense> delete(@PathParam("uuid") UUID uuid) {
        if (!expenseService.delete(uuid)) {
            throw new WebApplicationException(Response.Status.NOT_FOUND);
        }
        return expenseService.list();
    }
    @PUT
    public void update(Expense expense) {
        expenseService.update(expense);
    }

}
```

4. Explora el microservicio utilizando **Swagger UI**.

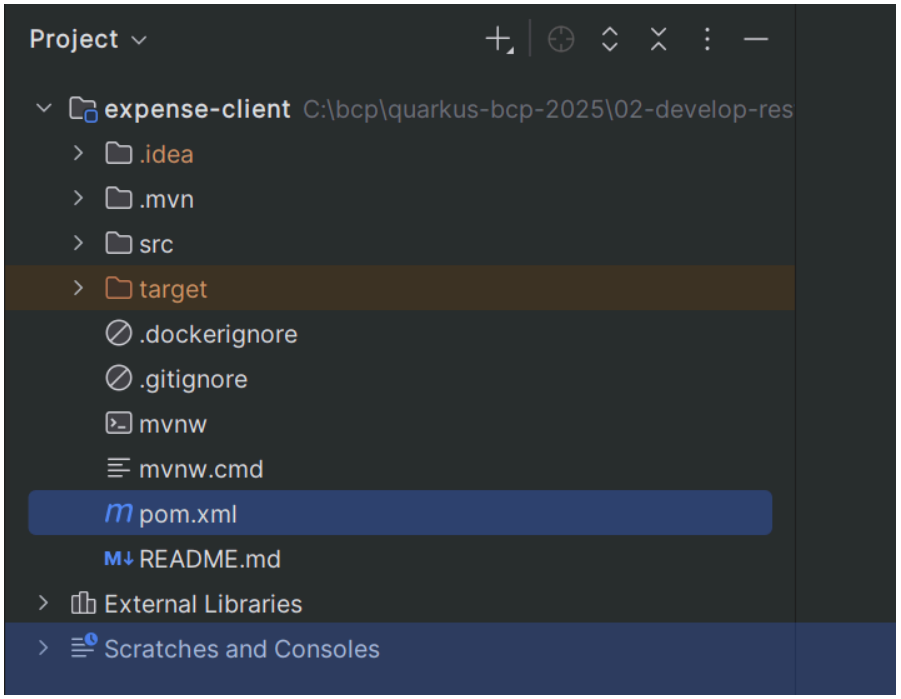
   4.1. En un navegador web, navega a:
   http://localhost:8080/q/swagger-ui

   4.2. Haz clic en **GET /expenses**. Luego haz clic en **Try it out** y después en **Execute**. Verifica que la sección **Server response** contenga los elementos que configuraste en el método **com.bcp.training.ExpenseService#init**.

   4.3. En la terminal de **tu IDE**, donde está corriendo la aplicación, presiona **q** para detener el microservicio. ✅

5. Implementa el microservicio cliente.

   5.1. Abre **tu IDE**, luego haz clic en **File > Open Folder**, selecciona el directorio **expense-client** y después haz clic en **Open**.



5.2. Abre la interfaz **src/main/java/com/bcp/training/client/ExpenseServiceClient.java**. Luego, configura **/expenses** como la ruta en la que atiende el microservicio **expense-service**. Usa la anotación **@RegisterRestClient** para habilitar que Quarkus instancie esta interfaz.

```java
@Path("/expenses")
@RegisterRestClient
public interface ExpenseServiceClient {

    @GET
    Set<Expense> getAll();

    @POST
    Expense create(Expense expense);
}
```

5.3 En el archivo **src/main/resources/application.properties**, configura el endpoint al que el cliente enviará las solicitudes.

```
quarkus.http.port=8090
quarkus.rest-client."com.bcp.training.client.ExpenseServiceClient".url=htt
p://localhost:8080
```

5.4 En el archivo **src/main/resources/application.properties**, configura el endpoint al que el cliente envía las solicitudes.

```
@Path("/expenses")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public class ClientResource {


   @Inject
   @RestClient
   ExpenseServiceClient service;
```

application.properties × ClientResource.java ×

```java
package com.bcp.training.service;

import ...

@Path("/expenses")  Jose Amadeo Diaz *
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public class ClientResource {


    @Inject  2 usages
    @RestClient
    ExpenseServiceClient service;

    @GET  Jose Amadeo Diaz
    public Set<Expense> getAll() { return service.getAll(); }

    @POST  Jose Amadeo Diaz
    public Expense create(Expense expense) { return service.create(expense); }
}

```

6. Empaqueta los microservicios como imágenes de contenedor.

   6.1. En el microservicio **expense-client**, agrega la extensión de Quarkus **container-image-jib**.

   En **tu IDE**, haz clic en **Terminal > New Terminal** y ejecuta el siguiente comando:

```
mvn quarkus:add-extension -Dextensions=container-image-jib
```

```
Terminal   Local
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\bcp\quarkus-bcp-2025\02-develop-rest-start\expense-client> mvn quarkus:add-extension -Dextensions=container-image-jib
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< com.bcp.training:expense-client >-------------------
[INFO] Building expense-client 1.0.0-SNAPSHOT
[INFO]   from pom.xml
```

6.2 En el archivo **src/main/resources/application.properties**, configura el nombre de la imagen.

```
quarkus.container-image.build=true
quarkus.container-image.group=quay.io
quarkus.container-image.name=expense-client
```

```
application.properties
1  quarkus.http.port=8090
2  quarkus.rest-client."com.bcp.training.client.ExpenseServiceClient".url=http://localhost:8080
3
4  quarkus.container-image.build=true
5  quarkus.container-image.group=quay.io
6  quarkus.container-image.name=expense-client
```

6.3 En una terminal, construye la imagen de contenedor de **expense-client**.

```
Terminal   Local
[WARNING] [io.quarkus.resteasy.reactive.server.deployment.QuarkusServerEndpointIndexer] Quarkus detected the use of JSON in JAX-RS method 'com.bcp.training.service.ClientResource#create' but no JSON extension has been added. Consider adding 'quarkus-rest-jackson' (recommended) or 'quarkus-rest-jsonb'.
 a specific image digest - build may not be reproducible
[INFO] [io.quarkus.container.image.jib.deployment.JibProcessor] Using base image with digest: sha256:360822c35c5741f542ab78fe123e6c4d9b68e0113a88d6e0250bb1f377b17f29
[INFO] [io.quarkus.container.image.jib.deployment.JibProcessor] Container entrypoint set to [/opt/jboss/container/java/run/run-java.sh]
[INFO] [io.quarkus.container.image.jib.deployment.JibProcessor] Created container image quay.io/expense-client:1.0.0-SNAPSHOT (sha256:1366fcbc2cd55de79b65c27da01564c27f4f9387c0813f40c1048f103e89f68d)

[INFO] [io.quarkus.deployment.QuarkusAugmentor] Quarkus augmentation completed in 12367ms
[INFO]
[INFO] --- failsafe:3.5.3:integration-test (default) @ expense-client ---
[INFO] Tests are skipped.
[INFO]
```

6.4 Cambia al proyecto **expense-service**. Haz clic en **File > Open Recent** y selecciona el proyecto **expense-service**.

6.5 Verifica que el microservicio **expense-service** ya use la extensión **container-image-jib** examinando el archivo **pom.xml**.

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-container-image-jib</artifactId>
</dependency>
```

6.6 Verifica el nombre de la imagen del contenedor examinando el archivo **src/main/resources/application.properties**.

```
quarkus.container-image.build=true
quarkus.container-image.group=quay.io
quarkus.container-image.name=expense-service
```

6.7 En una nueva ventana de terminal, construye la imagen de contenedor de **expense-service**

mvn clean install

```
Terminal  Local
0250bb1f377b17f29
[INFO] [io.quarkus.container.image.jib.deployment.JibProcessor] Container entrypoint set to [/opt/jboss/container/java/run/run-java.sh]
[INFO] [io.quarkus.container.image.jib.deployment.JibProcessor] Created container image quay.io/expense-service:1.0.0-SNAPSHOT (sha256:1ac0297a10000
4f2906bed04838689424065d829beed08f68319bb03aaa58613)

[INFO] [io.quarkus.deployment.QuarkusAugmentor] Quarkus augmentation completed in 11578ms
[INFO]
[INFO] --- failsafe:3.5.3:integration-test (default) @ expense-service ---
[INFO] Tests are skipped.
[INFO]
```

6.8 Verifica que ambas imágenes existan en tu estación de trabajo.

Linux o mac:

```
podman images | grep expense
```

Windows:

```
podman images | Select-String "expense"
```

```
PS C:\bcp\quarkus-bcp-2025\02-develop-rest-start\expense-service> podman images | Select-String "expense"

quay.io/expense-service                          1.0.0-SNAPSHOT  a4ca0228cafe  2 minutes ago  410 MB
quay.io/expense-client                           1.0.0-SNAPSHOT  37fc39830ac9  7 minutes ago  409 MB
localhost/expense-app                            latest          d798d7cda892  4 days ago     501 MB
```

7. Prueba los contenedores de las aplicaciones

7.1. En una nueva terminal, crea una red de **Podman** para tu aplicación.

```
podman network create expense-net
```

7.2. Inicia el contenedor **expense-service** en la red **expense-net** como un proceso en segundo plano.

```
podman run --rm --name expense-service -d --network expense-net quay.io/expense-service:1.0.0-SNAPSHOT
```

7.3. Inicia el contenedor **expense-client** con los siguientes parámetros:

- **Red**: expense-net
- **Mapeo de puertos**: exponer el puerto 8090
- **Variables de entorno**:
  –
  `QUARKUS_REST_CLIENT__COM_BCP_TRAINING_CLIENT_EXPENSESERVICECLIENT__URL="http://expense-service:8080"`

```
podman run --rm --name expense-client -d -e QUARKUS_REST_CLIENT__COM_BCP_TRAINING_CLIENT_EXPENSESERVICECLIENT__URL="http://expense-service:8080" -p 8090:8090 --network expense-net quay.io/expense-client:1.0.0-SNAPSHOT
```

Debes proporcionar la variable de entorno, porque el microservicio **expense-client** contiene la siguiente entrada en el archivo **application.properties**:

```
quarkus.rest-client."com.bcp.training.client.ExpenseServiceClient".url=http://
localhost:8080
```

7.4. En un navegador web, navega a **http://localhost:8090** y prueba la aplicación **expense**.

8. Para y elimina los contenedores

```
podman stop -a
```

Podman elimina ambos contenedores debido a la opción **--rm**.

enjoy!

Jose