LAB 21: QUARKUS MONITORING LOGGING

Autor: José Díaz

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

Abre el proyecto **18-monitor-logging-start**

1. Abre el proyecto con tu editor favorito.
2. Ejecuta la aplicación en modo desarrollo: **mvn quarkus:dev**

```
[student@workstation monitor-logging]$ mvn quarkus:dev
...output omitted...
INFO  [io.quarkus] (Quarkus Main Thread) expenses ... Listening on: http://
localhost:8080
...output omitted...
```

3. Loguea un mensaje de error cuando un request al endpoint **GET /expenses/{name}** trata de obtener un expense que no existe.

   a. Abre la clase **ExpenseResource** y modifica el método **getByName** para loguear mensajes de error **ExpenseNotFoundException**.

```
@GET
@Path( "/{name}" )
public Expense getByName( @PathParam( "name" )
String name ) {

    try {
        return expenses.getByName( name );
    } catch( ExpenseNotFoundException e ) {
        var message = e.getMessage();
        Log.error( message );
        throw new NotFoundException( message );
    }
}
```

   b. Abre una nueva terminal y haz un requests para obtener el expense llamado **none**. Este expense no existe.

curl -v http://localhost:8080/expenses/none

```
[student@workstation monitor-logging]$ curl -v \
http://localhost:8080/expenses/none
...output omitted...
>
< HTTP/1.1 404 Not Found
< Content-Type: application/json
< content-length: 0
<
...output omitted...
```

c. Retorna a la terminal donde la aplicación esta ejecutandose y verifica que la salida muestra el siguiente error:

```
...output omitted...
ERROR [com.red.tra.exp.ExpensesResource] (executor-thread-0) Expense not found:
 none
```

4. Loguea un mensaje de debug y ajusta el log level de la aplicación a **DEBUG**.

a. En la clase **ExpensesResource**, modifica el método **getByName** para loguear el mensaje debug **Getting expense {name}.**

```
@GET
@Path( "/{name}" )
public Expense getByName( @PathParam( "name" )
String name ) {
    Log.debug( "Getting expense " + name );
    try {
        return expenses.getByName( name );
    } catch( ExpenseNotFoundException e ) {
        var message = e.getMessage();
        Log.error( message );
        throw new NotFoundException( message );
    }
}
```

b. Haz un request para obtener el expense llamado **joel-2**.

curl -s http://localhost:8080/expenses/joel-2 | jq

```
[student@workstation monitor-logging]$ curl -s \
http://localhost:8080/expenses/joel-2 | jq
{
  "uuid": "6df0b95d-afec-4171-a980-7c915a309f69",
  "name": "joel-2",
  "creationDate": "2023-01-23T10:22:32.005245452",
  "paymentMethod": "CASH",
  "amount": 10.0,
  "username": "joel@example.com"
}
```

c. Retorna a la terminal donde la aplicación esta ejecutandose. Verifica que el mensaje de debug no es mostrado. La consola no muestra el mensaje de debvug porque el default log leve es **INFO**.
d. Agrega la siguiente linea a **src/main/resources/application.properties**:

```
quarkus.log.level = DEBUG
```

e. Reejecuta el request al mismo endpoint:

curl -s http://localhost:8080/expenses/joel-2 | jq

```
[student@workstation monitor-logging]$ curl -s \
http://localhost:8080/expenses/joel-2 | jq
...output omitted...
```

f. Verifica que el mensaje de DEBUG es mostrado en la consola de la aplicación.

```
...output omitted...
DEBUG [io.qua.arc.run.BeanContainerImpl] (Quarkus Main Thread) No matching
 bean ...
DEBUG [io.qua.arc.run.BeanContainerImpl] (Quarkus Main Thread) No matching
 bean ...
...output omitted...
DEBUG [com.red.tra.exp.ExpensesResource] (executor-thread-0) Getting expense
 joel-2
```

Observa que la aplicación podría mostrar mensajes de debug que no son relevantes para este ejercicio.

```
...output omitted...
DEBUG [io.qua.arc.run.BeanContainerImpl] (Quarkus Main Thread) ...
DEBUG [io.qua.arc.run.BeanContainerImpl] (Quarkus Main Thread) ...
...output omitted...
DEBUG [com.red.tra.exp.ExpensesResource] (executor-thread-0) Getting expense
 joel-2
```

5. Configurar el log level DEBUG solo para el paquete **com.bcp.training.expense**.

a. En el archivo application.properties cambia el root log level de DEBUG a INFO:

```
quarkus.log.level = INFO
```

b. En el mismo archivo, agrega la siguiente linea para establecer el log level de la categoría **com.bcp.training.expense** a **DEBUG**.

```
quarkus.log.category."com.bcp.training.expense".level = DEBUG
```

c. Reejecuta el request al mismo endpoint.

curl -s http://localhost:8080/expenses/joel-2 | jq

```
[student@workstation monitor-logging]$ curl -s \
http://localhost:8080/expenses/joel-2 | jq
...output omitted...
```

d. Verifica que los logs de la aplicación muestran solo mensajes de debug generados en el **paquete com.bcp.training.expense**.

```
...output omitted...
INFO  [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding
 activated.
INFO  [io.quarkus] (Quarkus Main Thread) Installed features: [...]
INFO  [io.qua.dep.dev.RuntimeUpdatesProcessor] (vert.x-worker-thread-0) Live
 reload total time ...
DEBUG [com.red.tra.exp.ExpensesResource] (executor-thread-0) Getting expense
 joel-2
```

6. Personalizar el logging en modo de desarrollo. Envía los logs al archivo, define un formato específico de logging, y desactiva la rotación de log cuando la aplicación reinicia.

a. Agrega las siguientes lineas al archivo application.properties.

```
%dev.quarkus.log.file.enabled=true
%dev.quarkus.log.file.path=C:\\Users\\josed\\DO378
\\monitor-logging\\dev.logs
%dev.quarkus.log.file.format=%d %-5p [%F] %m%n
%dev.quarkus.log.file.rotation.rotate-on-
boot=false
```

b. Reejecuta los requests en el mismo endpoint.

curl -s http://locahost:8080/expenses/joel-2 | jq

```
[student@workstation monitor-logging]$ curl -s \
http://localhost:8080/expenses/joel-2 | jq
...output omitted...
```

c. Verifica que el **/home/student/DO378/monitor-logging/dev.logs** contiene los logs en el formato específico.

```
...output omitted...
2023-01-23 09:11:39,451 DEBUG [ExpensesResource.java] Getting expense joel-2
```

d. Retorna a la terminal donde la aplicación esta corriendo y tipea **q** para detener la aplicación.

Fin del laboratorio.

Enjoy!

José