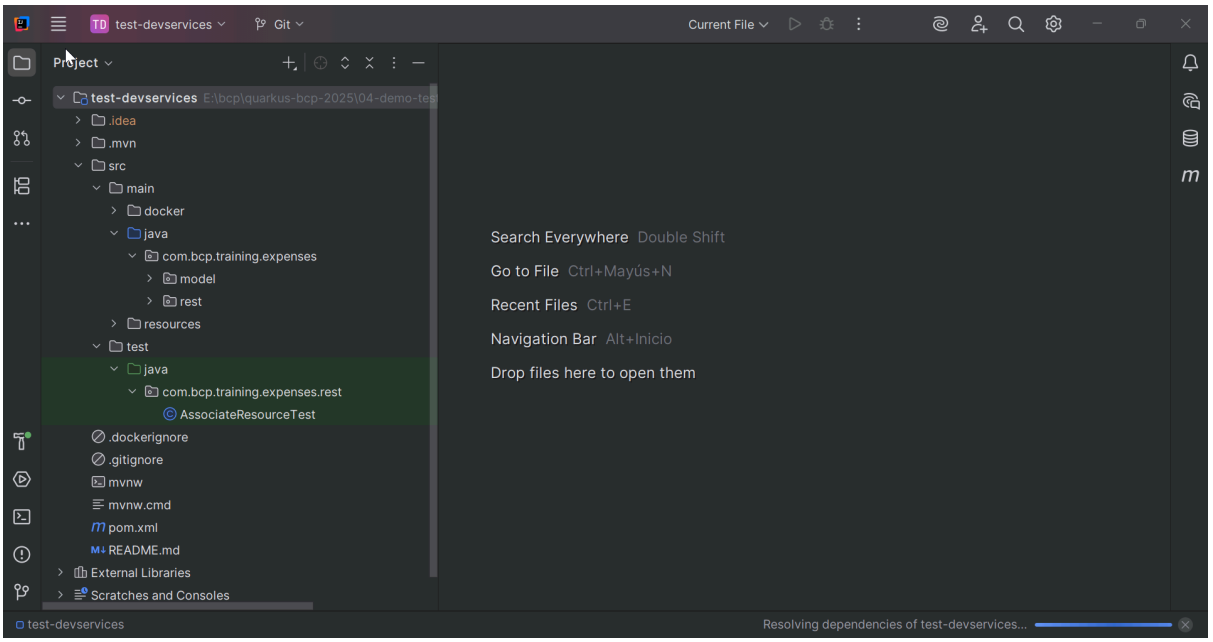
LAB 8: QUARKUS DEV SERVICES

Autor: José Díaz
Apoyo: Juan Ramirez

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

1. Abre el proyecto **test-devservices**.



2. Crea una anotación personalizada para ejecutar una base de datos PostgreSQL.
   Crea el archivo src/test/java/com/bcp/training/rest/WithPostgresDB.java.
   La interfaz de anotación debe tener tres parámetros de tipo String: name, username y password.
   El objetivo de esta anotación es cualquier tipo de Java, y debe tener retención en tiempo de ejecución.

   La anotación debe lucir como el siguiente código:

```
import io.quarkus.test.common.QuarkusTestResource;
```

```java
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface WithPostgresDB {
    String username() default "";
    String password() default "";
    String name() default "";
}
```

3. Crea la clase de recurso de prueba para PostgreSQL.
    1. Crea el archivo `src/test/java/com/redhat/training/rest/PostgresDBTestResource.java`.

    2. La clase de recurso de prueba debe implementar la interfaz `QuarkusTestResourceConfigurableLifecycleManager`.

    3. Debe usar la anotación `WithPostgresDB` como parámetro genérico para esta interfaz

```
import
io.quarkus.test.common.QuarkusTestResourceConfigurableLifecycleManager;
public class PostgresDBTestResource implements
QuarkusTestResourceConfigurableLifecycleManager<WithPostgresDB> {
}
```

4. Agrega el código para capturar los parámetros de la anotación y establecerlos en los campos de la clase.

```
    private String name;
    private String username;
    private String password;


    @Override
    public void init(WithPostgresDB params) {
        username = params.username();
        password = params.password();
        name = params.name();
    }
```

5. Agrega el campo del contenedor de base de datos PostgreSQL de Testcontainers a la clase de recurso.

```
public class PostgresDBTestResource implements
QuarkusTestResourceConfigurableLifecycleManager<WithPostgresDB> {

    private static final DockerImageName imageName =
DockerImageName.parse("postgres:14.1")
            .asCompatibleSubstituteFor("postgres");
    private static final PostgreSQLContainer<?> DATABASE = new
PostgreSQLContainer<>(imageName);

    private String name;
    private String username;
    private String password;


    @Override
    public void init(WithPostgresDB params) {
        username = params.username();
```

```
        password = params.password();
        name = params.name();
    }
```

6. Agrega el código que inicia el contenedor de base de datos y establece las propiedades del *datasource* en el evento **start** del recurso.

```
    @Override
    public Map<String, String> start() {

DATABASE.withDatabaseName(name).withUsername(username).withPassword(password
).start();

        return Map.of("quarkus.datasource.username", username,
                "quarkus.datasource.password", password,
                "quarkus.datasource.jdbc.url", DATABASE.getJdbcUrl());
    }
```

7. Agrega el código que detenga el contenedor de la base de datos en el evento **stop** del recurso.

```
    @Override
    public void stop() {
        DATABASE.stop();
    }
```

8. Anota la interfaz de anotación personalizada `WithPostgresDB` con `@QuarkusTestResource`.

Pasa la clase recién creada `PostgresDBTestResource` como el parámetro por defecto y configura el parámetro `restrictToAnnotatedClass` en `true`.

```
@QuarkusTestResource(value = PostgresDBTestResource.class,
restrictToAnnotatedClass = true)
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface WithPostgresDB {
    String username() default "";
    String password() default "";
    String name() default "";
}
```

9. Anota la clase `AssociateResourceTest` con la anotación `@WithPostgresDB`.
Los parámetros para la anotación son:

- `tc-test` → nombre de la base de datos
- `tc-user` → usuario
- `tc-pass` → contraseña

```
```

```java
@QuarkusTest
@TestHTTPEndpoint( AssociateResource.class )
@WithPostgresDB(name = "tc-test", username = "tc-user", password =
"tc-pass")
public class AssociateResourceTest {
```

10. Ejecuta las pruebas para verificar que la base de datos está siendo desplegada por **Testcontainers**.

```
[student@workstation test-devservices]$ mvn test
...output omitted...

[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.redhat.training.rest.AssociateResourceTest

...output omitted...

... [# [postgres:15]] (pool-4-thread-1) Creating container for image: postgres:15
... [# [postgres:15]] (pool-4-thread-1) Container postgres:15 is starting:
 7a073...
... [# [postgres:15]] (pool-4-thread-1) Container postgres:15 started in
 PT1.482541461S

...output omitted...

[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
...output omitted...
```

enjoy!

Jose