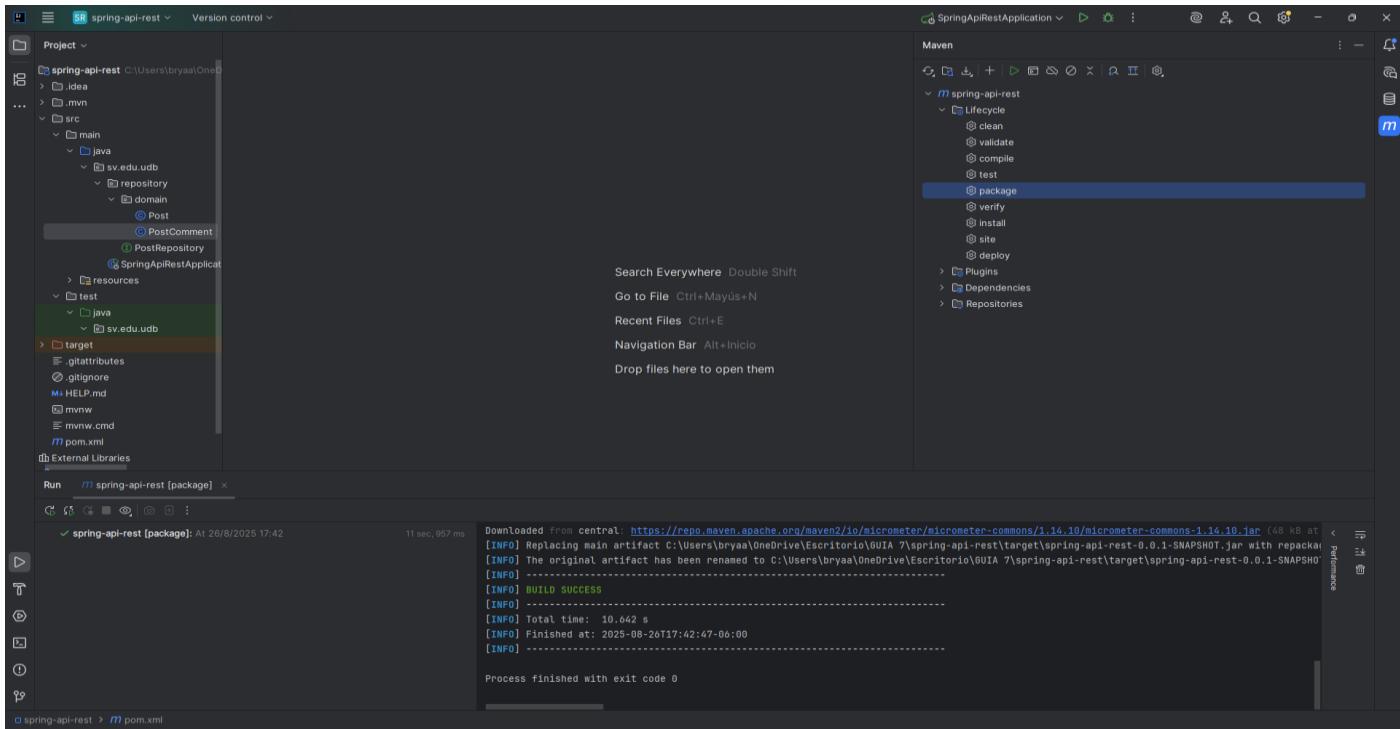
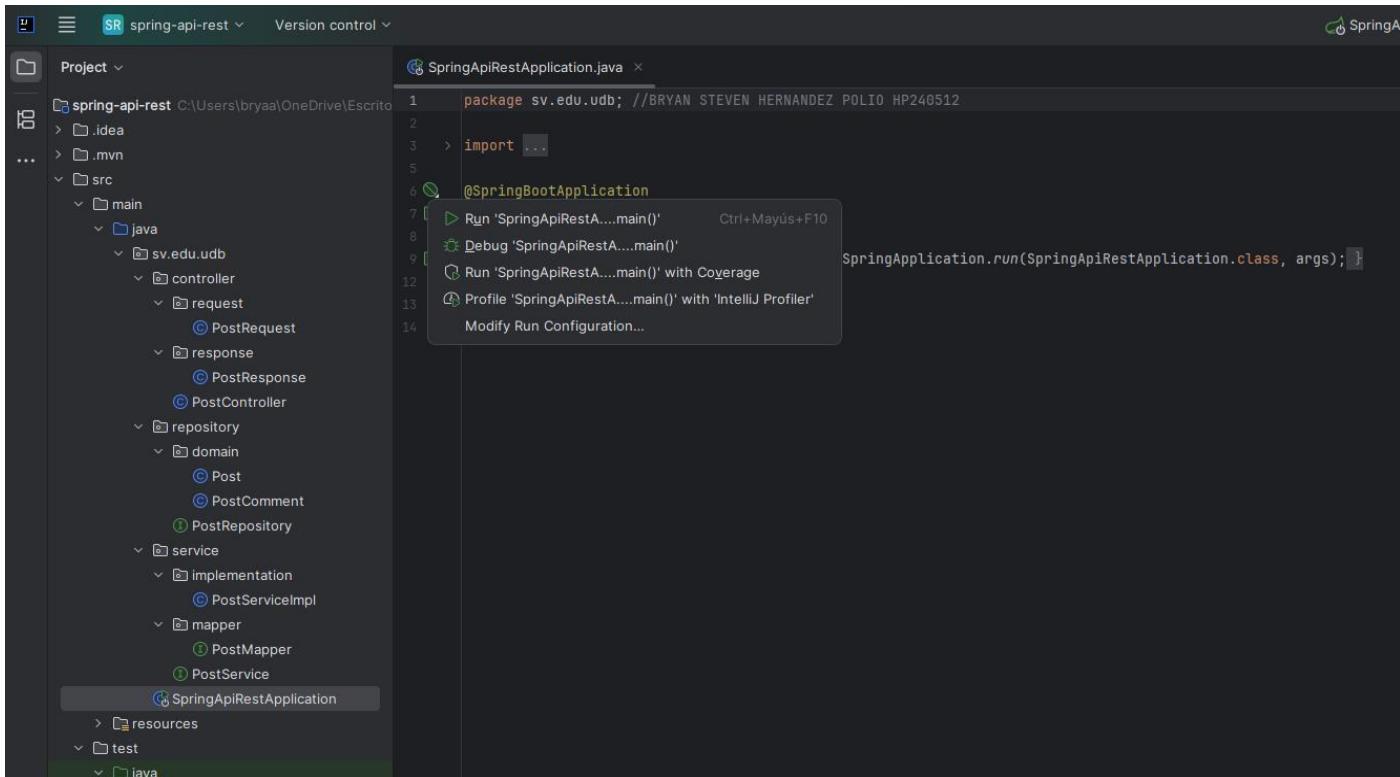


BRYAN STEVEN HERNANDEZ POLIO ----- HP240512 ----- DWF G03L

PRIMERA PRUEBA DE CLEAN Y PACKAGE:



Corriendo el Spring Api Rest:



Demostración de su ejecución y el puerto 8080:

The screenshot shows the IntelliJ IDEA interface with the project 'spring-api-rest' open. The code editor displays the main application class, `SpringApiRestApplication.java`, which contains the main method. Below the code editor is the terminal window showing the application's startup logs. The logs indicate the application is running on port 8080, connecting to a HikariDataSource, and starting up various components like JPA, H2, and Tomcat.

```
2025-08-26T18:15:09.312-06:00 INFO 13960 --- [spring-api-rest] [ restartedMain] org.hibernate.orm.connections.pooling : HHH0001005: Database info:  
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-3)']  
Database driver: undefined/unknown  
Database version: 2.5.252  
Autocommit mode: undefined/unknown  
Isolation level: undefined/unknown  
Minimum pool size: undefined/unknown  
Maximum pool size: undefined/unknown  
2025-08-26T18:15:09.349-06:00 INFO 13960 --- [spring-api-rest] [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA support)  
2025-08-26T18:15:09.353-06:00 INFO 13960 --- [spring-api-rest] [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'  
2025-08-26T18:15:09.393-06:00 WARN 13960 --- [spring-api-rest] [ restartedMain] jpabaseconfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during a JSP rendering. If this causes an NPE in your application, add 'jpa.openInView=false' to application.properties to disable it.  
2025-08-26T18:15:09.402-06:00 INFO 13960 --- [spring-api-rest] [ restartedMain] o.s.v.b.OptionalValidatorFactoryBean : Failed to set up a Bean Validation provider: jakarta.validation.NoProviderFoundException: Unable to find a Bean Validation provider implementation.  
2025-08-26T18:15:09.441-06:00 INFO 13960 --- [spring-api-rest] [ restartedMain] o.s.b.a.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:guia-jpa'  
2025-08-26T18:15:09.451-06:00 INFO 13960 --- [spring-api-rest] [ restartedMain] o.s.b.d.a.OptimisticLockingResource : LiveReload server is running on port 35729  
2025-08-26T18:15:09.458-06:00 INFO 13960 --- [spring-api-rest] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'  
2025-08-26T18:15:09.461-06:00 INFO 13960 --- [spring-api-rest] [ restartedMain] sv.edu.udb.SpringApiRestApplication : Started SpringApplication in 0.509 seconds (process running for 18.036)  
2025-08-26T18:15:09.461-06:00 INFO 13960 --- [spring-api-rest] [ restartedMain] .ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged
```

Primera prueba Get de Postman en localhost:

The screenshot shows the Postman application interface. A new collection named 'API Network' is selected. A POST request is defined for the endpoint 'localhost:8080/posts'. The 'Params' tab is active, showing a single parameter 'Key' with value 'Value'. The 'Body' tab shows a JSON response with an empty array. The status bar at the bottom indicates a successful 200 OK response with 346 ms duration and 166 B size.

PRUEBAS DE POST MAN:

The screenshot shows the Postman interface with a dark theme. A project named "spring-api-rest" is open, and the file "SpringApiRestApplication.java" is displayed in the code editor. In the main workspace, a POST request is being made to "localhost:8080/posts". The request body is set to raw JSON:

```
1 {  
2   "title": "Hello wordl",  
3   "postDate": "25/09/2024"  
4 }
```

The response status is 201 Created, with a response time of 29 ms and a size of 216 B. The response body is identical to the request body.

The screenshot shows the Postman interface with a dark theme. A project named "spring-api-rest" is open, and the file "SpringApiRestApplication.java" is displayed in the code editor. In the main workspace, a GET request is being made to "localhost:8080/posts". The request body is set to raw JSON:

```
1 {  
2   "title": "Hello wordl",  
3   "postDate": "25/09/2024"  
4 }
```

The response status is 200 OK, with a response time of 7 ms and a size of 213 B. The response body is a JSON array containing one element, which matches the request body.

POST SERVICE TEST

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "spring-api-rest". The "src/test/java" directory contains a package "sv.edu.udb" with sub-packages "controller", "repository", and "service". A file named "PostServiceTest.java" is selected.
- Code Editor:** The code for "PostServiceTest.java" is displayed. It imports various Mockito and JUnit assertions. The class "PostServiceTest" contains a constructor annotated with "@InjectMocks" and a private field "private PostServiceImpl postService;". A comment explains the purpose of the @Mock annotations on the repository fields. The "Run" tab shows the test results.
- Run Tab:** The "PostServiceTest" run configuration is selected. The output shows:
 - 6 tests passed (6 tests total, 1sec 63ms)
 - Find a no existed id then throws an exception (1sec 32ms)
 - Find all the post when we have data (11ms)
 - Delete Post when post id exist (4ms)
 - Should update Post When Post Request and id (8ms)
 - Save Post when the Post request is valid (4ms)
 - Find Post by id (4ms)- A warning message from OpenJDK 64-Bit Server VM: "Sharing is only supported for boot loader classes because bootstrap classpath has been appended".
- The process finished with exit code 0.

POST CONTROLLER TEST FINAL:

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Tree:** Shows the project structure with packages like `Post`, `PostComment`, `PostRepository`, `service`, `implementation`, `mapper`, and `PostService`. Below these are `SpringApiRestApplication` and test files like `PostControllerTest.java` and `PostServiceTest.java`.
- Code Editor:** The `PostControllerTest.java` file is open, containing Java code for testing a REST controller. It includes annotations like `@Test` and `@DisplayName("Get all post")`. The code uses Mockito to mock a service and verify its behavior.
- Run Output:** The bottom pane shows the execution results for the `PostControllerTest` class. It indicates 1 test passed in 170ms. The log output shows the test name, duration, and the Spring Boot environment information. The log ends with the message "Completed initialization in 2 ms".