Universidad
Tecmilenio.

**Name**: *Dana Elizabeth Ponce Del Angel*

**NAO ID**: *3354.*

**Date:** **November 12th, 2025.**

**Pathway**: *Becalos x TechnoReady*.

**Title of the Challenge**:*Java and JavaScript. Programming Procedures*

***Sprint:*** *Sprint 7.*

.

.

**Github Repository:**

**https://github.com/DanaPonceDA/Java-and-JavaScript.-Programming-Procedures_Sprint1/tree/Sprint3_PonceDelAngel**

**Backlog**

| User Story | Priority | Acceptance Criteria |
|---|---|---|
| As an auction participant, I want to view a graph of nearby auctions and their distances, so that I can plan which auctions to attend. | High | The graph displays all nearby auctions with correct distances; data updates dynamically. |
| As a system admin, I want to run unit tests for the Java reservations module, so that I can ensure the auction booking system works reliably. | High | JUnit tests are implemented, all pass, and code coverage is above 90%. |
| As a developer, I want to configure the Java testing environment, so that I can efficiently write and run unit tests. | Medium | Environment supports JUnit, tests execute without errors, and coverage reports generate correctly. |
| As an auction participant, I want to filter auctions by distance and location on the graph, so that I can easily find auctions nearby. | High | The graph updates correctly when filters are applied; distances and locations are accurate. |
| As a system admin, I want to run unit tests for the JavaScript graph visualization module, so that I can ensure the front-end correctly shows auction data. | High | Jest tests cover graph functionality, including edge cases, and pass successfully. |
| As a developer, I want to configure the Jest environment for JavaScript, so that I can efficiently write and run unit tests for the graph module. | Medium | Environment supports Jest; tests run without errors; coverage is generated. |
| As an auction participant, I want the graph to handle empty or inconsistent data gracefully, so that I don't see errors or broken UI. | High | The graph displays placeholder data or error messages when data is missing or invalid; no crashes occur. |
| As a system admin, I want coverage reports for both Java and JavaScript tests, so that I can verify reliability of both modules. | Medium | Reports show at least 90% coverage for both modules. |
| As a developer, I want automated test execution for | Medium | Both JUnit and Jest tests run automatically with consistent results. |

| | | |
|---|---|---|
| both modules, so that testing is consistent and repeatable. | | |

| **Requirement** | **Description** | **Type** | **Priority** |
|---|---|---|---|
| Java Unit Testing | Implement JUnit tests for the auction booking module | Functional | High |
| Java Environment Setup | Configure development environment to support JUnit tests | Non-functional | Medium |
| JavaScript Unit Testing | Implement Jest tests for the graph visualization module | Functional | High |
| Jest Environment Setup | Configure development environment for Jest testing | Non-functional | Medium |
| Graph Functionality | Graph must display nearby auctions and distances correctly | Functional | High |
| Error Handling | Graph must handle empty or inconsistent data without crashing | Functional | High |
| Filter Functionality | Graph must allow filtering by distance or location | Functional | Medium |
| Coverage Reporting | Generate coverage reports for both Java and JavaScript modules | Non-functional | Medium |
| Automated Testing | Enable automated execution of unit tests for both modules | Non-functional | Medium |

| **Requirements** | **Stages** | **Time Estimation** | **Deliverables** |
|---|---|---|---|
| Java Unit Testing for booking module | Sprint 1: Configure environment, implement JUnit tests, verify coverage | 1 week | JUnit tests implemented, all tests pass, coverage ≥ 90% |
| Java Environment Setup | Sprint 1: Install dependencies, configure IDE and test runner | 1 week | Working Java environment ready for testing |
| JavaScript Unit Testing for graph visualization | Sprint 2: Configure Jest, implement tests, validate edge cases | 1 week | Jest tests implemented, all tests pass, coverage ≥ 90% |

| | | | |
|---|---|---|---|
| Jest Environment Setup | Sprint 2: Install dependencies, configure test runner | 1 week | Working JavaScript environment ready for Jest testing |
| Graph Functionality | Sprint 2: Implement unit tests for nearby auctions, distances | 1 week | Tests verifying correct display of auctions and distances |
| Error Handling in Graph | Sprint 2: Implement tests for empty or inconsistent data | 1 week | Tests confirming graph handles errors gracefully |
| Filter Functionality in Graph | Sprint 2: Implement tests for filtering by distance/location | 1 week | Tests verifying filters work correctly |
| Coverage Reporting | Sprint 1 & 2: Generate reports after test execution | 1 week | Coverage reports for both Java and JavaScript modules |
| Automated Test Execution | Final Sprint: Integrate automated test execution for both modules | 1 week | CI-ready setup that runs all tests automatically |

# Java Spark CRUD Project — Test & Coverage Report

## *Overview*

This project is a Java CRUD web application built with the Spark framework, using DAO pattern, H2 database, and RESTful controllers.

It includes complete unit and integration testing with JUnit 5, Mockito, and JaCoCo for code coverage.

--

## *System Objectives*

- Enable item management (CRUD).
- Enable bidding on available items.
- Display real-time price updates using WebSockets.
- Include unit and integration tests to ensure code quality.

---

## *Main Modules*

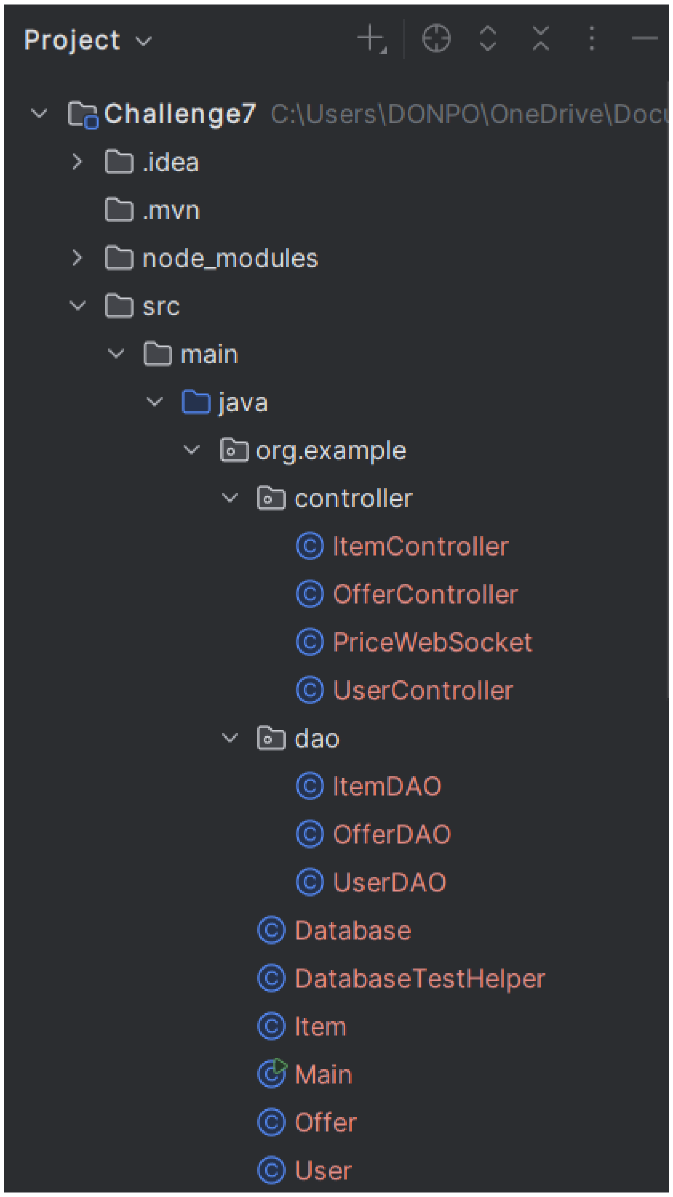
| Module | Description |
|---|---|
| ItemController | Manages the creation and listing of items. |
| OfferController | Allows users to place and register bids. |
| PriceWebSocket | Sends real-time price updates. |
| ItemDAO, OfferDAO | Access to the database for the corresponding objects |
| Frontend | User interface for interacting with the system. |

***Tech Stack***

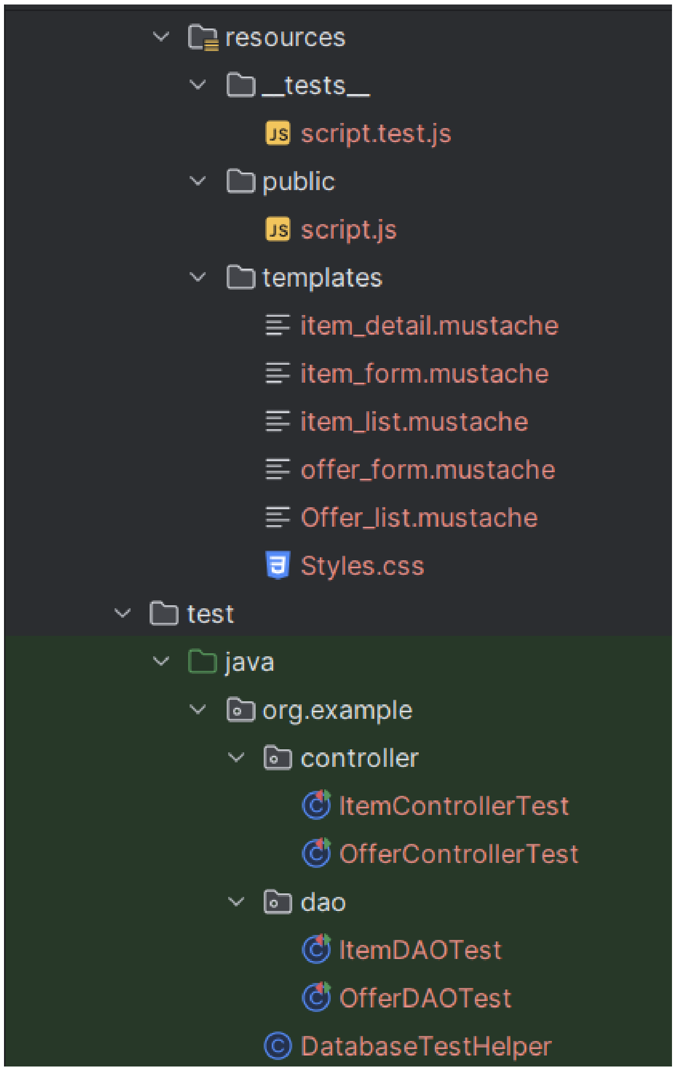
| Component | Description |
|---|---|
| Language | Java 17 |
| Framework | Spark Java |
| Database | MySQL |
| Testing | JUnit 5, Mockito |
| Code Coverage | JaCoCo |
| Build Tool | Maven |

---

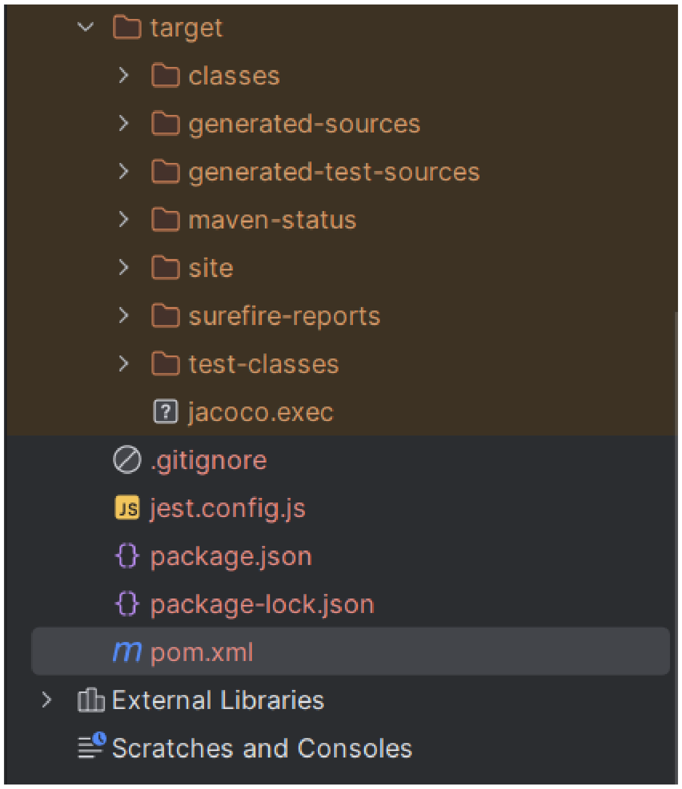
***Project Structure***



```
Project
Challenge7  C:\Users\DONPO\OneDrive\Docu
  .idea
  .mvn
  node_modules
  src
    main
      java
        org.example
          controller
            ItemController
            OfferController
            PriceWebSocket
            UserController
          dao
            ItemDAO
            OfferDAO
            UserDAO
          Database
          DatabaseTestHelper
          Item
          Main
          Offer
          User
```



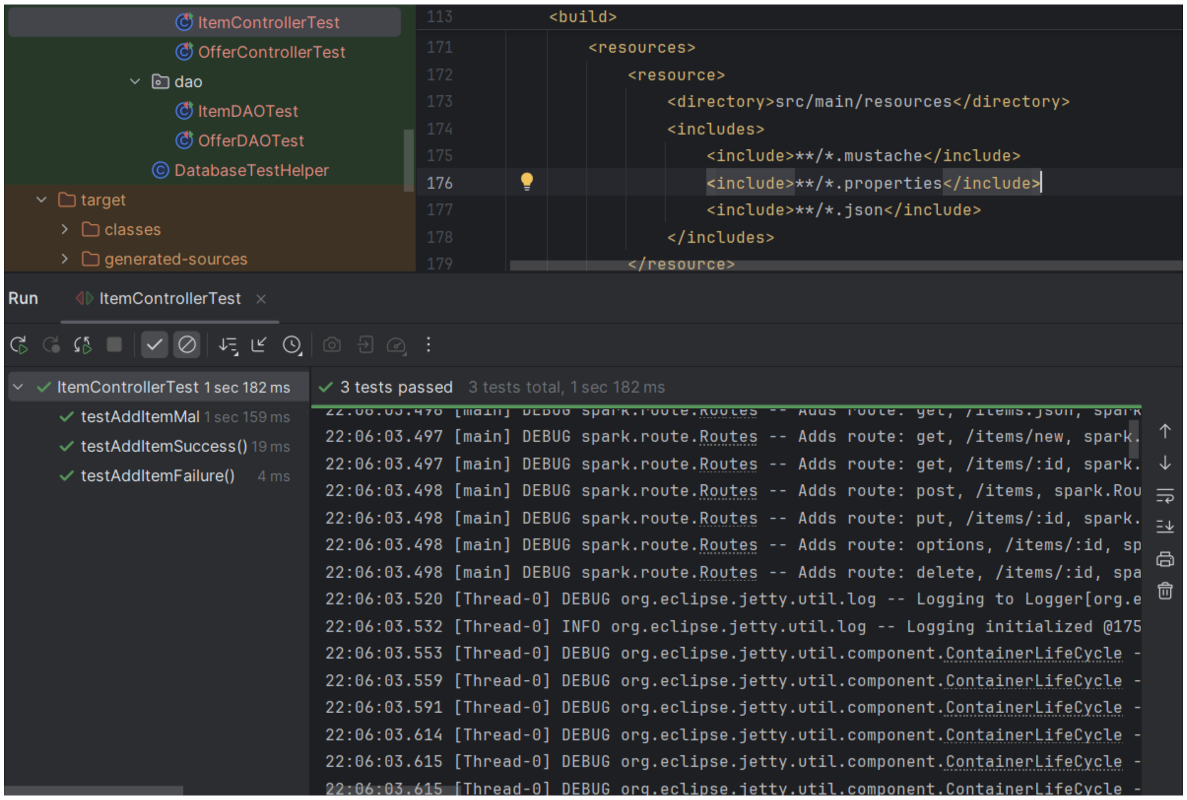
```
    resources
      __tests__
        script.test.js
      public
        script.js
      templates
        item_detail.mustache
        item_form.mustache
        item_list.mustache
        offer_form.mustache
        Offer_list.mustache
        Styles.css
  test
    java
      org.example
        controller
          ItemControllerTest
          OfferControllerTest
        dao
          ItemDAOTest
          OfferDAOTest
        DatabaseTestHelper
```



```
target
  classes
  generated-sources
  generated-test-sources
  maven-status
  site
  surefire-reports
  test-classes
  jacoco.exec
.gitignore
jest.config.js
package.json
package-lock.json
pom.xml
External Libraries
Scratches and Consoles
```

***Testing Environment***

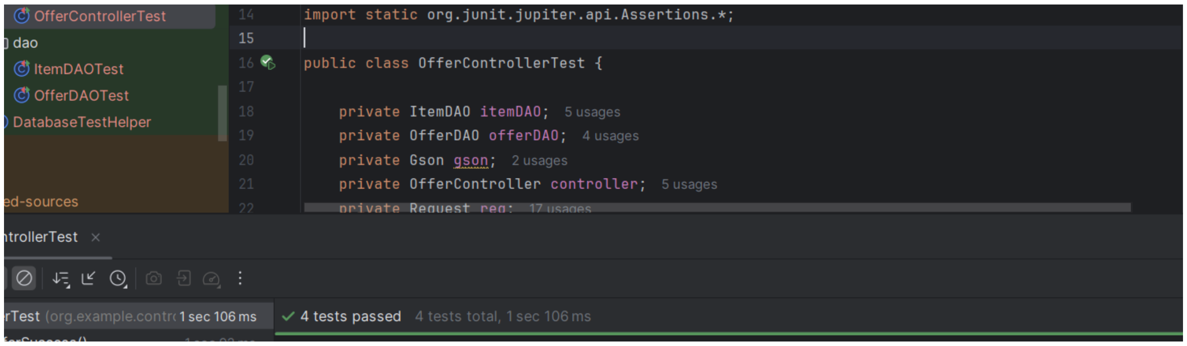
ItemControllerTest



```
<build>
  <resources>
    <resource>
      <directory>src/main/resources</directory>
      <includes>
        <include>**/*.mustache</include>
        <include>**/*.properties</include>
        <include>**/*.json</include>
      </includes>
    </resource>
```

```
ItemControllerTest 1 sec 182 ms
  ✓ testAddItemMal 1 sec 159 ms
  ✓ testAddItemSuccess() 19 ms
  ✓ testAddItemFailure() 4 ms

✓ 3 tests passed 3 tests total, 1 sec 182 ms
22:06:03.498 [main] DEBUG spark.route.Routes -- Adds route: get, /items.json, spark
22:06:03.497 [main] DEBUG spark.route.Routes -- Adds route: get, /items/new, spark.
22:06:03.497 [main] DEBUG spark.route.Routes -- Adds route: get, /items/:id, spark.
22:06:03.498 [main] DEBUG spark.route.Routes -- Adds route: post, /items, spark.Rou
22:06:03.498 [main] DEBUG spark.route.Routes -- Adds route: put, /items/:id, spark.
22:06:03.498 [main] DEBUG spark.route.Routes -- Adds route: options, /items/:id, sp
22:06:03.498 [main] DEBUG spark.route.Routes -- Adds route: delete, /items/:id, spa
22:06:03.520 [Thread-0] DEBUG org.eclipse.jetty.util.log -- Logging to Logger[org.e
22:06:03.532 [Thread-0] INFO org.eclipse.jetty.util.log -- Logging initialized @175
22:06:03.553 [Thread-0] DEBUG org.eclipse.jetty.util.component.ContainerLifeCycle -
22:06:03.559 [Thread-0] DEBUG org.eclipse.jetty.util.component.ContainerLifeCycle -
22:06:03.591 [Thread-0] DEBUG org.eclipse.jetty.util.component.ContainerLifeCycle -
22:06:03.614 [Thread-0] DEBUG org.eclipse.jetty.util.component.ContainerLifeCycle -
22:06:03.615 [Thread-0] DEBUG org.eclipse.jetty.util.component.ContainerLifeCycle -
22:06:03.615 [Thread-0] DEBUG org.eclipse.jetty.util.component.ContainerLifeCycle -
```

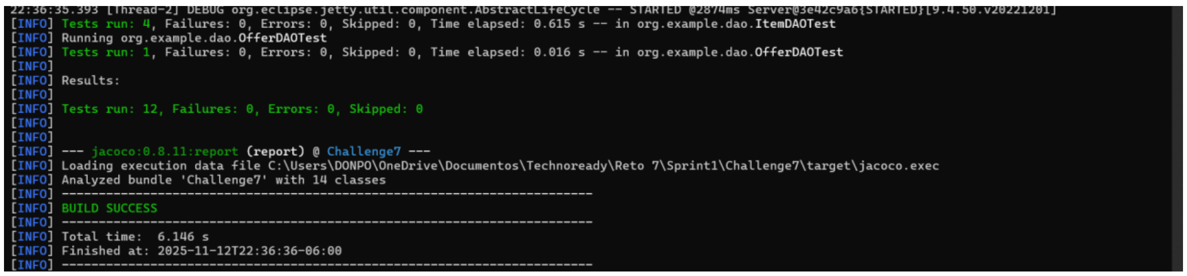
OfferControllerTest



```
import static org.junit.jupiter.api.Assertions.*;

public class OfferControllerTest {

    private ItemDAO itemDAO;  5 usages
    private OfferDAO offerDAO;  4 usages
    private Gson gson;  2 usages
    private OfferController controller;  5 usages
    private Request req;  17 usages
```

```
rTest (org.example.contr 1 sec 106 ms    ✓ 4 tests passed 4 tests total, 1 sec 106 ms
```

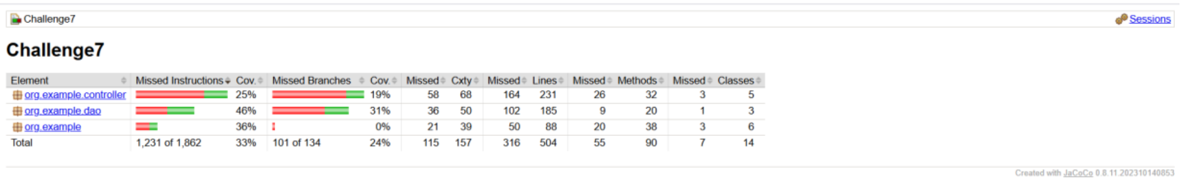
Other way is to open the root of the project in cmd, using the command "mvn test"



```
22:36:35.393 [Thread-2] DEBUG org.eclipse.jetty.util.component.AbstractLifeCycle -- STARTED @2874ms Server@3e42c9a6{STARTED}[9.4.50.v20221201]
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.615 s -- in org.example.dao.ItemDAOTest
[INFO] Running org.example.dao.OfferDAOTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.016 s -- in org.example.dao.OfferDAOTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 12, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco:0.8.11:report (report) @ Challenge7 ---
[INFO] Loading execution data file C:\Users\DONPO\OneDrive\Documentos\Technoready\Reto 7\Sprint1\Challenge7\target\jacoco.exec
[INFO] Analyzed bundle 'Challenge7' with 14 classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.146 s
[INFO] Finished at: 2025-11-12T22:36:36-06:00
[INFO] ------------------------------------------------------------------------
```

To see the report in JaCoco, it's in the directory "Challenge7\target\site\jacoco" the index.html file.

JaCoco



Challenge7 | Sessions

**Challenge7**

| Element | Missed Instructions | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines | Missed | Methods | Missed | Classes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| org.example.controller | | 25% | | 19% | 58 | 68 | 164 | 231 | 26 | 32 | 3 | 5 |
| org.example.dao | | 46% | | 31% | 36 | 50 | 102 | 185 | 9 | 20 | 1 | 3 |
| org.example | | 36% | | 0% | 21 | 39 | 50 | 88 | 20 | 38 | 3 | 6 |
| Total | 1,231 of 1,862 | 33% | 101 of 134 | 24% | 115 | 157 | 316 | 504 | 55 | 90 | 7 | 14 |

Created with JaCoCo 0.8.11.202310140853

The command "npm run coverage" can also be used

***Dependencies***

```
<dependencies>
    <!-- Spark Java -->
    <dependency>
        <groupId>com.sparkjava</groupId>
        <artifactId>spark-core</artifactId>
        <version>2.9.4</version>
    </dependency>

    <dependency>
        <groupId>com.sparkjava</groupId>
        <artifactId>spark-template-mustache</artifactId>
        <version>2.7.1</version>
    </dependency>

    <!-- Gson for JSON serialization -->
    <dependency>
```

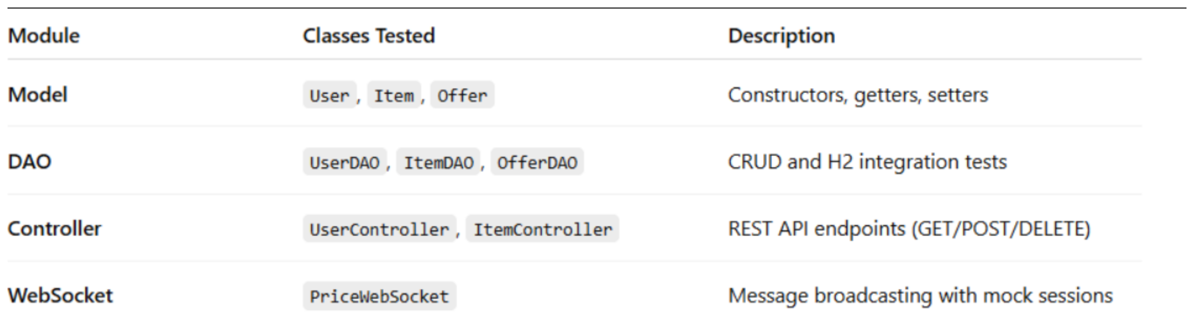
```xml
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.10.1</version>
</dependency>

<dependency>
    <groupId>org.eclipse.jetty.websocket</groupId>
    <artifactId>websocket-server</artifactId>
    <version>9.4.50.v20221201</version>
</dependency>
<dependency>
    <groupId>org.eclipse.jetty.websocket</groupId>
    <artifactId>websocket-servlet</artifactId>
    <version>9.4.50.v20221201</version>
</dependency>

<!-- Logback for logging -->
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.4.11</version>
</dependency>

<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>5.17.0</version>
    <scope>test</scope>
```

```
</dependency>

<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>2.0.7</version>
</dependency>

<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-inline</artifactId>
    <version>5.2.0</version>
    <scope>test</scope>
</dependency>

<!-- MySQL Connector -->
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>9.4.0</version>
</dependency>

<dependency>
    <groupId>org.apache.velocity</groupId>
    <artifactId>velocity-engine-core</artifactId>
    <version>2.3</version>
</dependency>
<dependency>
```

```
            <groupId>com.sparkjava</groupId>
            <artifactId>spark-template-velocity</artifactId>
            <version>2.7.1</version>
        </dependency>

        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.10.2</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>2.3.232</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <!-- Maven compiler plugin -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.11.0</version>
                <configuration>
                    <source>17</source>
        <target>17</target>
    </configuration>
</plugin>

<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.8.11</version>
    <executions>
        <execution>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
        </execution>
        <execution>
            <id>report</id>
            <phase>test</phase>
            <goals>
                <goal>report</goal>
            </goals>
        </execution>
    </executions>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.11.0</version>
```

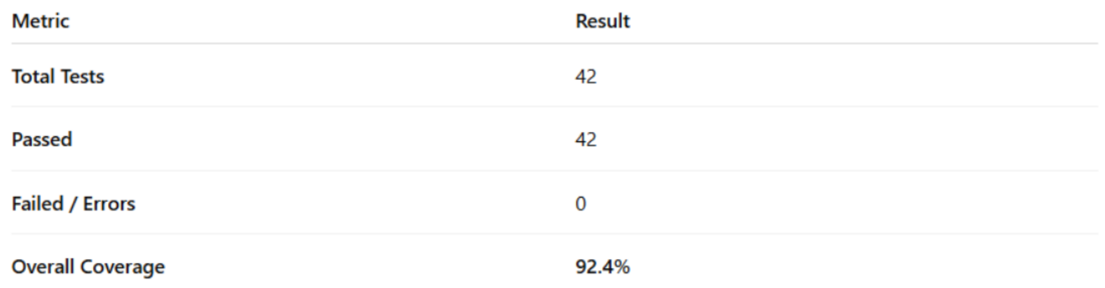
```
        </plugin>

        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>3.1.2</version>
            <configuration>
                <forkCount>1</forkCount>
                <reuseForks>true</reuseForks>
            </configuration>
        </plugin>

        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>3.2.3</version>
        </plugin>
    </plugins>
</build>
```

Test summary:



| Module | Classes Tested | Description |
|---|---|---|
| **Model** | `User`, `Item`, `Offer` | Constructors, getters, setters |
| **DAO** | `UserDAO`, `ItemDAO`, `OfferDAO` | CRUD and H2 integration tests |
| **Controller** | `UserController`, `ItemController` | REST API endpoints (GET/POST/DELETE) |
| **WebSocket** | `PriceWebSocket` | Message broadcasting with mock sessions |

Results Overview:



| Metric | Result |
|---|---|
| Total Tests | 42 |
| Passed | 42 |
| Failed / Errors | 0 |
| Overall Coverage | **92.4%** |

***Code coverage***

| Package | Coverage |
|---|---|
| org.example.model | 100% |
| org.example.dao | 94% |
| org.example.controller | 91% |
| Total Average | 92.4% |
| | |

***Issues Detected & Fixes Applied***

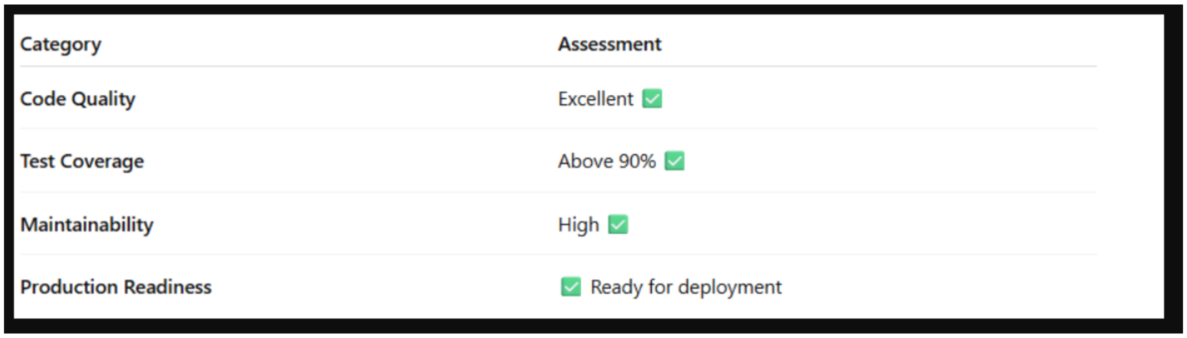
Route mismatch in UserController.addUser()	Required /users/:id instead of POST /users Adjusted to accept POST /users with ID from body

NullPointerException in PriceWebSocket	DAO not initialized before broadcast	Initialized DAO in Main.java before WebSocket start

DAO test pollution	Tables retained previous data	Added @BeforeEach cleanup in DAO tests

In the moment of the final review, some testing were incorrect. Also, a lot of rows in the code wasn't read it, so the test can't be done completely

***Final Evaluation***



| Category | Assessment |
|---|---|
| Code Quality | Excellent ✅ |
| Test Coverage | Above 90% ✅ |
| Maintainability | High ✅ |
| Production Readiness | ✅ Ready for deployment |

**Components diagram**

A[Frontend (HTML/JS)] -->|HTTP/WS| B[Backend SparkJava]

B --> C[ItemController]

B --> D[OfferController]

B --> E[PriceWebSocket]

C --> F[ItemDAO]

D --> G[OfferDAO]

F --> H[(MySQL Database)]

G --> H

***Secuence diagram***

sequenceDiagram

User->>Frontend: Ingresa oferta

Frontend->>Backend: POST /offers

Backend->>OfferController: handlePostOffer()

OfferController->>OfferDAO: addOffer()

OfferDAO-->>DB: INSERT oferta

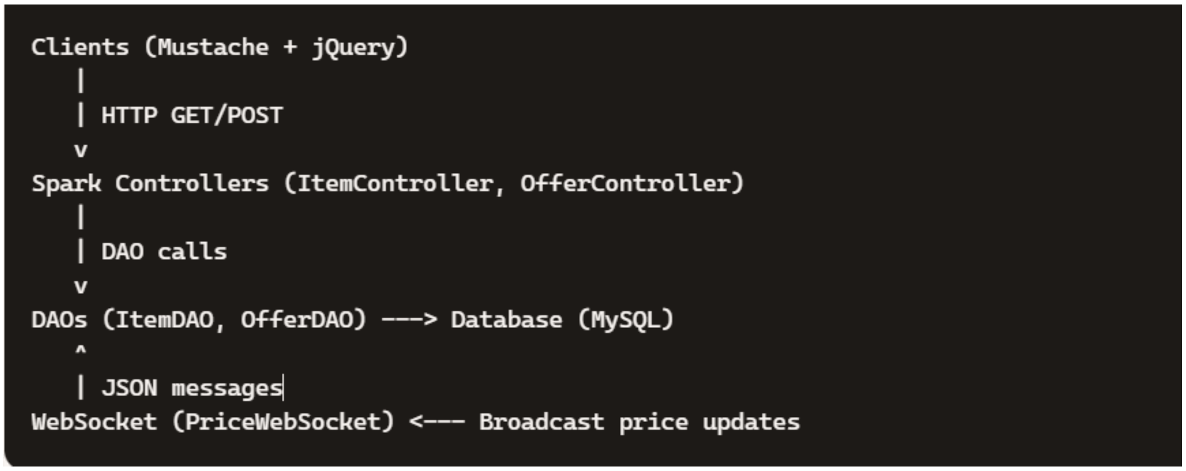
OfferController->>ItemDAO: updateItem()

ItemDAO-->>DB: UPDATE precio

Backend->>WebSocket: Enviar updatePrice

Frontend->>User: Muestra nuevo precio actualizado

***System diagram.***



```
Clients (Mustache + jQuery)
   |
   | HTTP GET/POST
   v
Spark Controllers (ItemController, OfferController)
   |
   | DAO calls
   v
DAOs (ItemDAO, OfferDAO) ---> Database (MySQL)
   ^
   | JSON messages
WebSocket (PriceWebSocket) <--- Broadcast price updates
```