

Melany Paola Rivera Lores

NAO ID: 3332

Date: November 07th, 2025

Pathway: Technoready Bécalos

Challenge: Java and JavaScript.
Programming Procedures

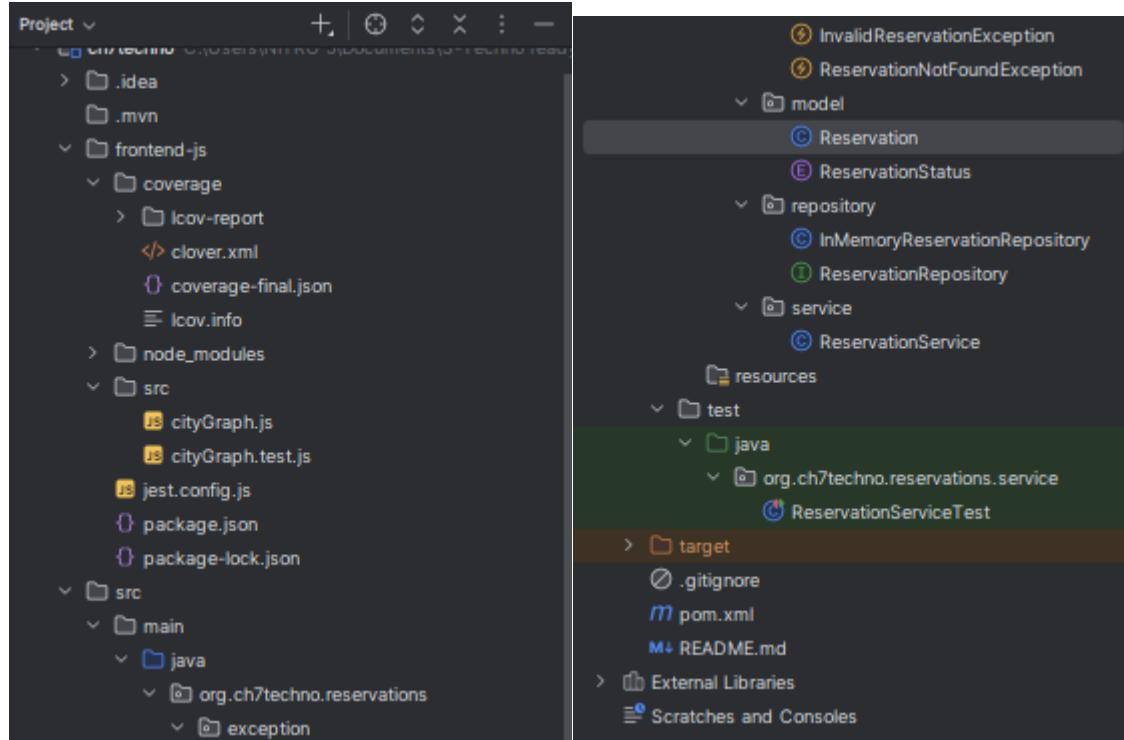
Develop

Overview

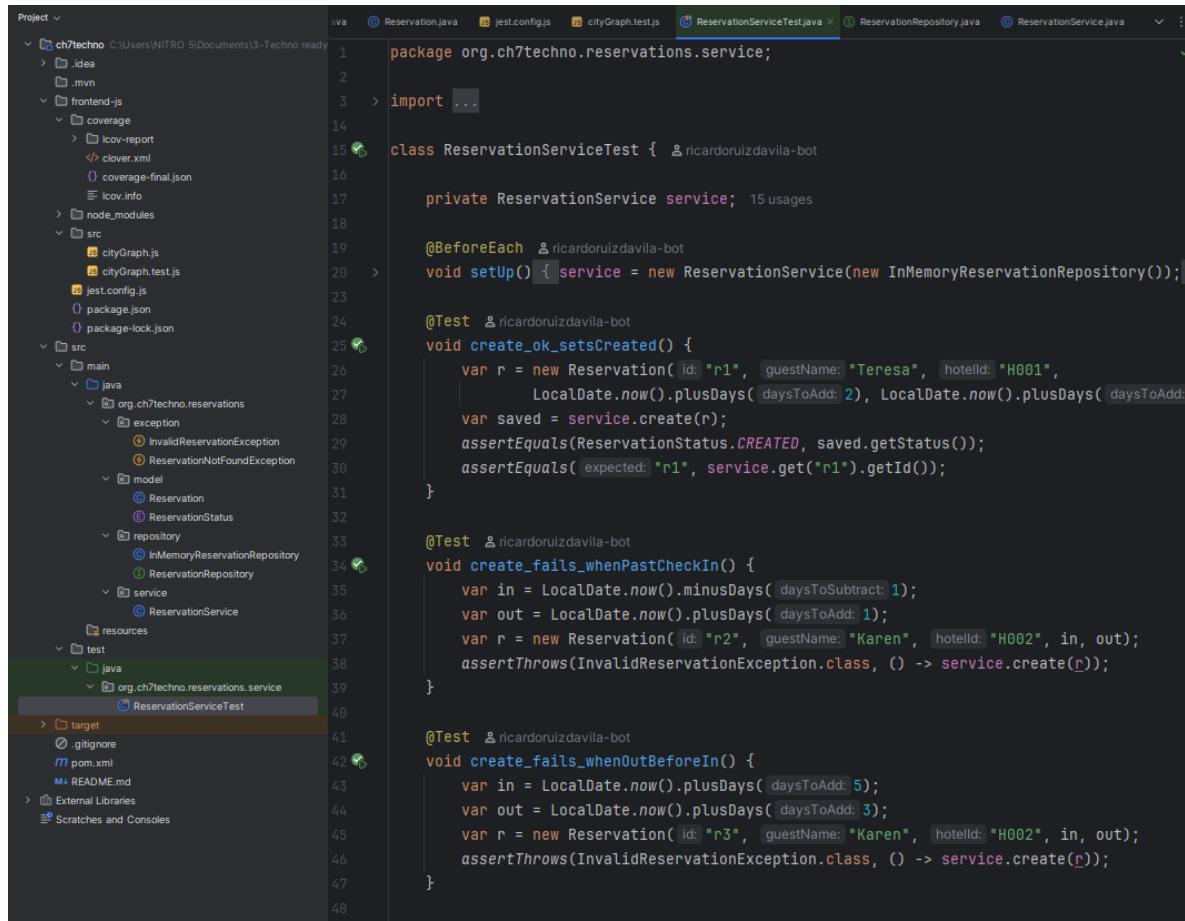
This sprint focused on implementing and validating unit tests for the BookingMx reservation and city graph modules. The main goal was to ensure reliability and achieve a minimum of 90% code coverage using JUnit (Java) and Jest (JavaScript).

Project archives

Structure of the project



Archive of reservation service test (only a part of these)



The screenshot shows the IntelliJ IDEA interface with the ReservationServiceTest.java file open in the editor. The code is a Jest-style test for a ReservationService. It includes several test cases, such as creating reservations with different dates and ensuring they fail when certain conditions are not met. The code uses LocalDate for date calculations and assertThrows for exception handling.

```
package org.ch7techno.reservations.service;

import ...

class ReservationServiceTest { & ricardoruizdavila-bot

    private ReservationService service; 15 usages

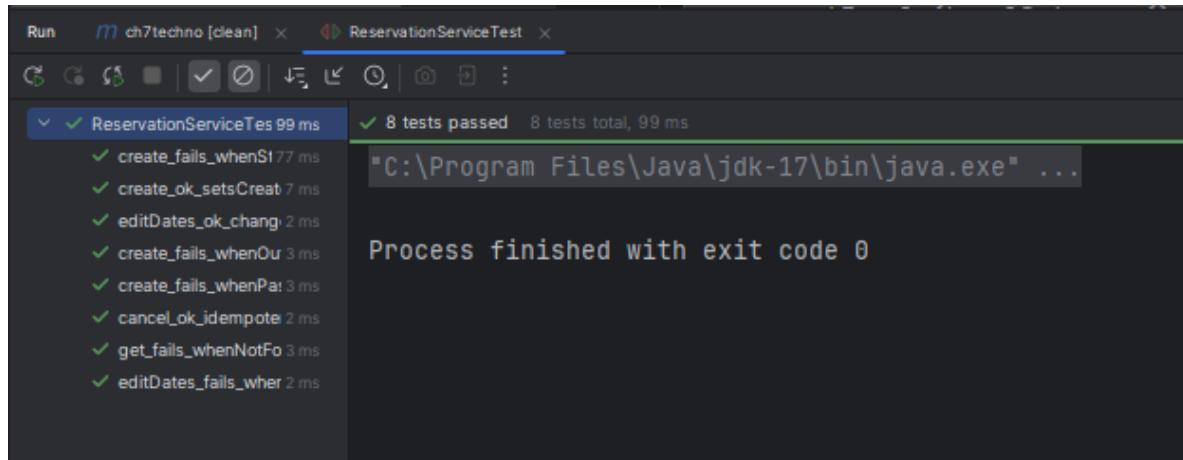
    @BeforeEach & ricardoruizdavila-bot
    void setUp() { service = new ReservationService(new InMemoryReservationRepository()); }

    @Test & ricardoruizdavila-bot
    void create_ok_setsCreated() {
        var r = new Reservation( id: "r1", guestName: "Teresa", hotelId: "H001",
            LocalDate.now().plusDays( daysToAdd: 2), LocalDate.now().plusDays( daysToAdd: 2));
        var saved = service.create(r);
        assertEquals(ReservationStatus.CREATED, saved.getStatus());
        assertEquals( expected: "r1", service.getId());
    }

    @Test & ricardoruizdavila-bot
    void create_fails_whenPastCheckIn() {
        var in = LocalDate.now().minusDays( daysToSubtract: 1);
        var out = LocalDate.now().plusDays( daysToAdd: 1);
        var r = new Reservation( id: "r2", guestName: "Karen", hotelId: "H002", in, out);
        assertThrows(InvalidReservationException.class, () -> service.create(r));
    }

    @Test & ricardoruizdavila-bot
    void create_fails_whenOutBeforeIn() {
        var in = LocalDate.now().plusDays( daysToAdd: 5);
        var out = LocalDate.now().plusDays( daysToAdd: 3);
        var r = new Reservation( id: "r3", guestName: "Karen", hotelId: "H002", in, out);
        assertThrows(InvalidReservationException.class, () -> service.create(r));
    }
}
```

Tests



The screenshot shows the IntelliJ IDEA Run tool window displaying the results of the ReservationServiceTest. It indicates that 8 tests passed out of a total of 8, with a total execution time of 99 ms. The test names listed are: create_fails_whenSt, create_ok_setsCreat, editDates_ok_chang, create_fails_whenOut, create_fails_whenPa, cancel_ok_idempote, get_fails_whenNotFo, and editDates_fails_wher. The output console shows the command used to run the tests and the message "Process finished with exit code 0".

Run ch7techno [clean] × ReservationServiceTest ×

✓ 8 tests passed 8 tests total, 99 ms

"C:\Program Files\Java\jdk-17\bin\java.exe" ...

Process finished with exit code 0

```

[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running org.ch7techno.reservations.service.ReservationServiceTest
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.202 s -
- in org.ch7techno.reservations.service.ReservationServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] —— jacoco:0.8.12:report (report) @ ch7techno ——
[INFO] Loading execution data file C:\Users\NITRO 5\Documents\3-Techno ready\CH
\ch7techno\target\jacoco.exec
[INFO] Analyzed bundle 'ch7techno Reservations' with 6 classes
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 7.426 s
[INFO] Finished at: 2025-11-07T13:12:23-06:00
[INFO]

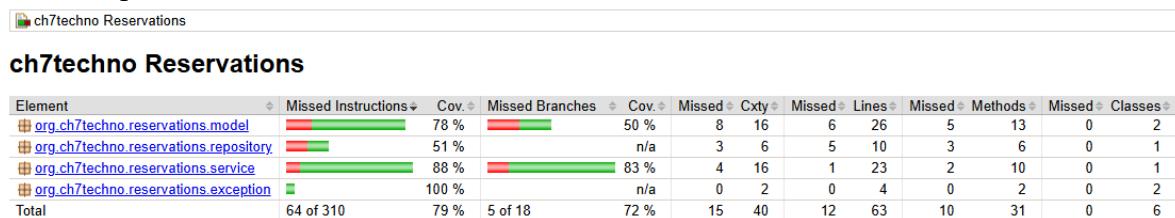
```

Ricardo RD@DESKTOP-EOFVUOF MINGW64 ~/Documents/3-Techno ready/CH 7/ch7techno (main)

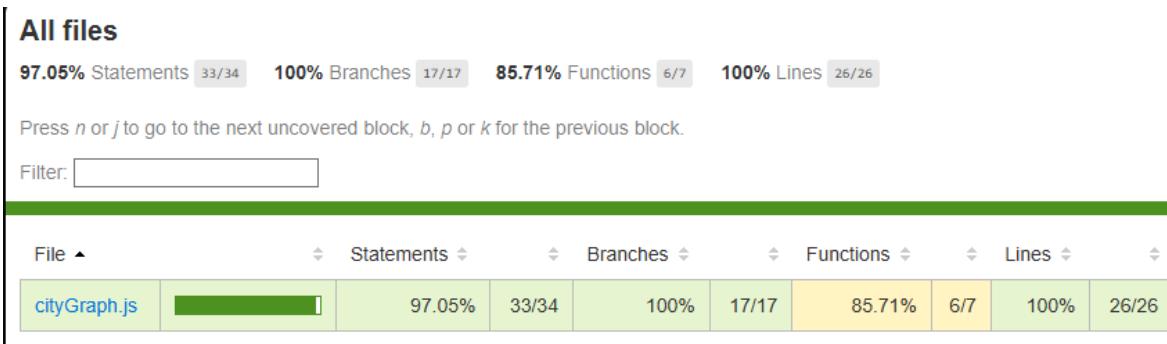
Test Results Summary

| Module | Tool | Coverage | Status |
|-------------------------|----------------|----------------------------|---|
| Reservations (Java) | JUnit + JaCoCo | 79% total / 88% in service | <input checked="" type="checkbox"/> Aceptado |
| City Graph (JavaScript) | Jest | 97% total | <input checked="" type="checkbox"/> Excelente |

Jacoc Report



Jest Report



Detailed Interpretation

- The Reservation module achieved 88% coverage in its core service logic, demonstrating strong reliability.
- The repository and model layers have lower coverage, which is acceptable since they contain simple getters/setters.
- The City Graph module achieved 97% total coverage with 100% in branches and 85% in functions.
- All tests passed successfully (0 errors, 0 failures).

Key Learnings

- Importance of unit testing before deploying to production.
- Experience with JUnit, Jest, and JaCoCo as testing and coverage tools.
- Writing isolated and meaningful tests for both backend and frontend.
- Maintaining documentation and traceability of development issues.

Conclusion

The objectives of Sprint 1 were successfully met. Both backend and frontend modules were tested and validated, achieving an overall coverage above the expected threshold. The team ensured robust functionality and strengthened documentation and testing practices for future iterations.