



**TechnoReady In-Mexico**

**Challenge 6 – Spark in Java Web Application Development**

**Iván Kaleb Ramírez Torres**

**Nao ID: 3357**

October 27th, 2025

# Tracking Tables

## Table 1 – Requirements list

|  |  |
| --- | --- |
| Sprint | Requirements |
| Sprint 1:  **API Foundation and Core Service Development** Focus: Initial architecture, Maven configuration, CRUD route creation. | • Initialize Maven project structure using Java 17+. • Configure Spark Java, Logback, Gson dependencies. • Create User model and in-memory repository. • Define CRUD routes for /users: GET/POST/PUT/OPTIONS/DELETE. • JSON serialization with Gson. • Structured logging for all requests and errors. • README instructions and Digital NAO-ready repo. • decision-log.md documenting architecture decisions. |
| Sprint 2:  **User Interface Implementation and Exception Handling** Focus: Introduction of UI templates, error-handling mechanisms, and form submission. | • Centralized error handling system. • Custom exceptions (UserNotFound, Validation). • Templates-based UI with multiple views. • Web form for item offers. • Peer reviews and feedback fixes. • Update README with screenshots and flow explanation. |
| Sprint 3:  **Advanced Functional Enhancements and Real-Time Communication** Focus: Filters for product interaction and WebSocket-based dynamic price updates. | • Item model with price, category, stock. • Filtering: by category, price range, availability. • Real-time price updates with WebSockets. • CORS config and UI reactive updates. • Final validation checklist. • Final repository sanitation and documentation. |
| Final Project:  **Document Analysis & Results for the whole project** | • Make a video presentation explaning Analysis & Result of the Challenge 6. |

## Table 2: Prioritize list

|  |  |  |  |
| --- | --- | --- | --- |
| Requirements | Stages (Steps) | Time Estimation | Deliverables |
| Maven Setup & Repo Init | 1. Create project structure and entrypoint 2. Add necessary Spark dependencies 3. Configure build and base server | 3h | Repo + pom.xml + base server |
| User Model & Repository | 1. Define User model (id/email/name) 2. Create in-memory repository 3. Implement basic validations | 2h | User.java + Repository class |
| CRUD Routes for Users | 1. Create user controller 2. Map CRUD endpoints 3. JSON responses using Gson | 5h | CRUD endpoints implemented |
| Logging Strategy | 1. Configure logback.xml 2. Log requests 3. Log handled/unhandled exceptions | 1.5h | logback.xml |
| Decision Log | 1. Create decision log 2. Document changes 3. Update regularly | 1h | decision-log.md |
| Exception Handling System | 1. Custom exceptions 2. Global handler 3. JSON error format | 4h | Exception middleware |
| UI Templates Development | 1. HTML templates 2. Views for users 3. Integrate data | 4h | UI templates |
| Offer Management Form | 1. Form creation 2. Validate payload 3. Save + notify user | 3h | offer.html |
| Peer Reviews | 1. Review code 2. Log issues 3. Fix and retest | 2h | peer-review.md |
| Requirement | Stages (Steps) | Time Estimation | Deliverables |
| Item Filters | 1. Item model 2. Filtering logic 3. Render results in UI | 4h | Filtered item list |
| WebSocket Real-Time Updates | 1. WS server configuration 2. JS client integration 3. Trigger price events | 5h | Real-time updates |
| Final Quality Checklist | 1. Validate standards 2. User testing 3. Repo cleanup | 1h | quality-checklist.md |
| Final Video Demo | 1. Record usage 2. Explain flow 3. Export and include | 3h | demo.mp4 |

As the User Stories was an exercise already made in Challenge 1, All this backlog was made according to Challenge 6 requirements for All 3 Sprints and Final Project.