SOFTWARE ARCHITECTURE LAB ASSIGNMENT 1

1. Defining the roles

The roles for the team are the following: Requirements elicitation, Class diagram implementation, Code implementation and Code reviewer.
The first role, **Requirements elicitation**, consists of extracting the functional and non-functional requirements from the provided documentation about the system. It is a fundamental role for all later stages.
The second role, **Class diagram implementation**, starts working right after requirements elicitation and will define an early class diagram for the new system.
The third and fourth roles, **Code implementation and Code reviewer,** are the final tasks in implementing the new system. They will implement the code and review it on Sonarqube so it complies with good practices about clean code.

All roles will be done by Luis Terrados Nieto

2. Functional requirements

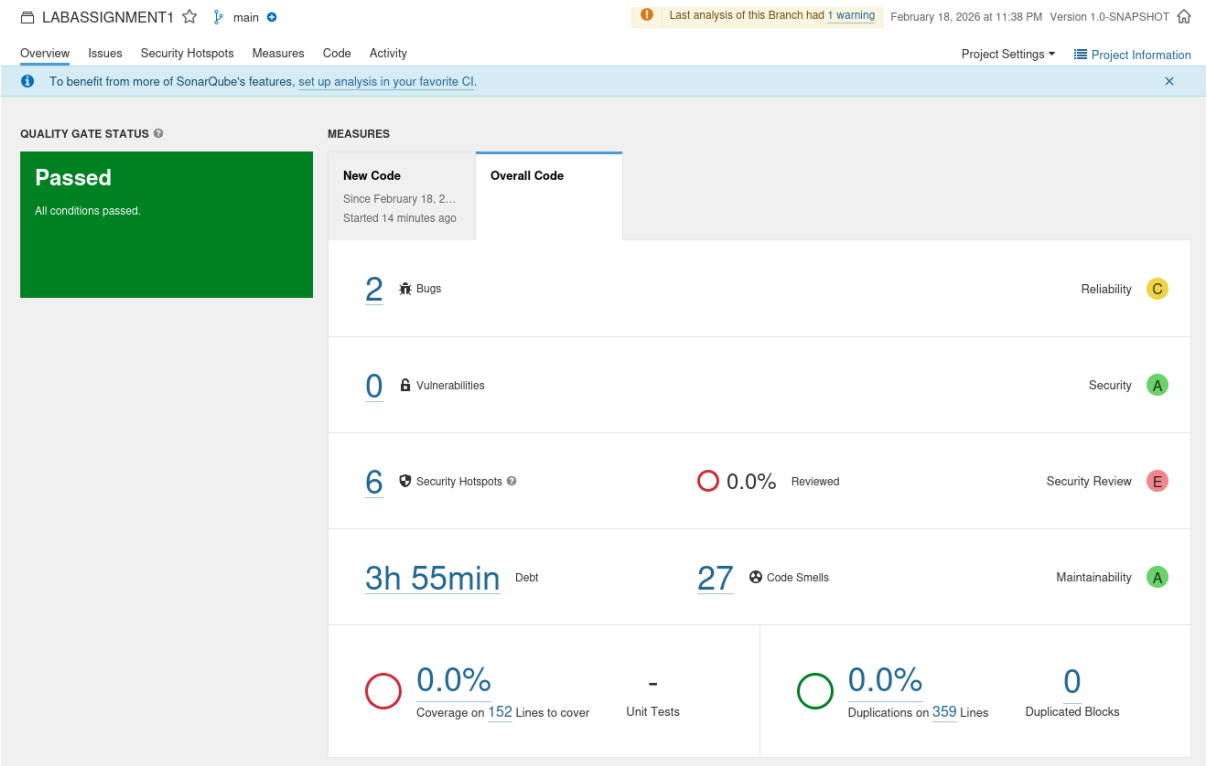
- RF-1: The system must support the following vehicle states: Available, Reserved, InUse, Maintenance, EmergencyLock and Relocating
- RF-2: The SmartMoveController must validate the transition done by a vehicle before changing state
- RF-3: The system must manage the lifecycle of the vehicles using a state machine
- RF-4: The system must only allow a vehicle transition to Maintenance if its telemetry reports a low battery
- RF-5: The system must only allow a vehicle transition to Maintenance if its telemetry reports a fault on the latter.
- RF-6: The system must trigger the EmergencyLock state for a vehicle if its hardware sensors report critical overheating
- RF-7: The system must detect movement of a vehicle without an active rental
- RF-8: The system must activate an alarm if a vehicle moves without an active rental
- RF-9: The system must trigger the EmergencyLock state for a vehicle if the latter moves without an active rental
- RF-10: The system must apply automatically a “Congestion Charge” on the final rental bill if the vehicle operates in London
- RF-11: The system must verify that the user wears the helmet via hardware sensors before unlocking a vehicle of the type moped if the latter operates in Milan

- RF-12: The system must restrict vehicle circulation on forbidden zones via GPS if it operates in Rome
- RF-13: The system must detect the city where a vehicle operates
- RF-14: The system must apply specific local rules depending on the city where the vehicle operates
- RF-15: The system must dynamically evaluate regulation rules of the city where the vehicle operates during execution flow
- RF-16: The system must manage telemetry updates from each vehicle
- RF-17: The location of the vehicle must be provided on each telemetry update via GPS
- RF-18: The battery level of the vehicle must be provided on each telemetry update
- RF-19: The internal temperature of the vehicle must be provided on each telemetry update
- RF-20: The system must continuously monitor the telemetry of the vehicle via background process
- RF-21: The system must intervene by slowing down the vehicle if its internal temperature exceeds 60ºC
- RF-22: The system must intervene by slowing down the vehicle if its battery level is below 5%
- RF-22: The system must intervene by terminating the vehicle rental if its internal temperature exceeds 60ºC
- RF-22: The system must intervene by terminating the vehicle rental if its battery level is below 5%
- RF-23: The system must persist all system data in CSV and JSON files
- RF-24: The system must persist user profiles
- RF-25: The system must persist vehicle state
- RF-26: The system must persist financial transactions
- RF-27: The system must append to a “Global Audit Log” every time a vehicle changes state
- RF-28: The system must append to a “Global Audit Log” every time a payment is processed
- RF-29: Every “Global Audit Log” must include a sequential ID of the vehicle entry
- RF-29: Every “Global Audit Log” must include a checksum of the vehicle entry
- RF-31: The system must detect write fails on the filesystem
- RF-32: The system must perform a manual rollback of the memory state to maintain data consistency if a write fails

3. Non-functional requirements

- NFR-1: The system must guarantee data consistency of the vehicles when concurrent access from rental and telemetry process occurs.
- NFR-2: The system must manage concurrency manually using low-level synchronization mechanisms
- NFR-3: The system must only persist information in CSV and JSON files
- NFR-4: The system must guarantee consistency between memory state and disk data
- NFR-5: The system must maintain a consistent system state if a writing failure occurs
- NFR-6: The system must handle a simulated fleet of 10000 vehicles
- NFR-7: The system must maintain a global audit log that allows complete traceability of states and transitions between them
- NFR-8: The system must immediately react when the telemetry of the vehicle reports critical conditions
- NFR-9: The system must guarantee the consistency of the vehicle when multiple process attempt to simultaneously modify it
- NFR-10: The system shall centralize the validation of business rules and state transitions in a single central controller

SONARQUBE REPORT



MISSING THINGS

As this was a one-man work, the following things are missing:

- At least 40% of code coverage