INTRODUZIONE

L’iterazione 3 è dedicata allo sviluppo e all’implementazione dei casi d’uso relativi alla visualizzazione dello stato dei macchinari (UC10) e all’aggiornamento del loro stato di uso (UC11).

ASSUNZIONI FATTE

Per poter conoscere lo stato attuale di un macchinario si è deciso di assumere innanzitutto che su ognuno di essi siano presenti i tre pulsanti “Libera Macchinario”, “Occupa Macchinario” e “Macchinario Guasto” che comunicano al server lo stato corrente quando premuti da un utente.

In vista del caso d’uso legato alla generazione di una scheda fitness aggiornata in base allo stato corrente dei macchinari (UC8), si è scelto di inviare le informazioni relative al nome del macchinario, alla tipologia del macchinario e all’orario al quale si libera il macchinario nel formato String.

L’informazione legata all’orario assume uno specifico valore in una delle seguenti situazioni:

* Premendo “LIBERA MACCHINARIO” viene inviato al server l’orario attuale;
* Premendo “OCCUPA MACCHINARIO” viene inviato al server un orario ottenuto effettuando internamente il calcolo “orario attuale + X minuti”, in cui la stima di X potrebbe essere fatta osservando l’occupazione media di quel macchinario;
* Premendo “MACCHINARIO GUASTO” viene inviato al server l’orario 23:59:59, assicurando la minore priorità possibile.

In questo modo è possibile gestire un sistema di priorità tra i macchinari utilizzando l’orario dal quale sono/saranno liberi.

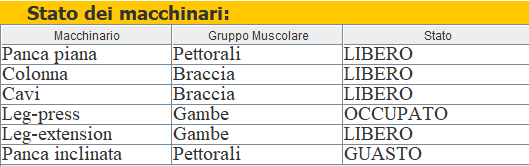
IMPLEMENTAZIONE

Per permettere agli utenti e al trainer di visualizzare lo stato corrente dei macchinari è stato creato un client Dashboard che si occupa di mostrare a schermo in formato tabellare le informazioni sui macchinari aggiornandole ogni 5 secondi.

In fase di implementazione della Dashboard si è usato il paradigma MVC per creare sia una View sia un Controller. Il Controller si occupa di inviare periodicamente richieste all’URL <http://localhost:8080/getStatoMacchinari> (una richiesta ogni 5 secondi) e di processare la stringa JSON ricevuta in risposta attraverso il metodo di parsing “parsingMacchinari(String s)”.

La classe Macchinario è stata implementata in modo tale da possedere un metodo “getStato()”che possa restituisca la stringa “OCCUPATO”, “LIBERO” o “GUASTO” confrontando il timestamp del macchinario con l’orario attuale.

Un possibile output prodotto dalla View della Dashboard è il seguente:



Server-side invece è stato creato il componente “Controller Aggregatore” come REST Controller che contiene un ArrayList di macchinari che aggiorna ogni volta che riceve nuove informazioni da un macchinario. Inoltre, Controller Aggregatore mette a disposizione due API:

* InfoMachineryIF permette alla Dashboard di leggere la lista di macchinari aggiornata:

@GetMapping(“/getStatoMacchinari”)

RisorsaJSON getStatoMacchinari()

* StatusIF permette ad ogni macchinario di aggiornare il proprio stato sulla lista:

GetMapping(“/writeStatoMacchinario”)

writeStatoMacchinario()

CASI DI TEST

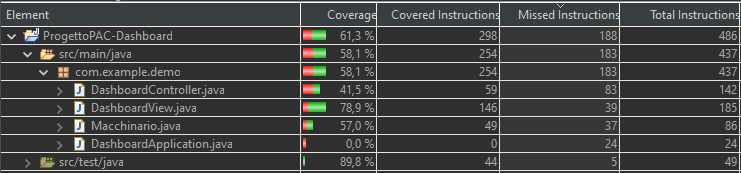
I metodi che richiedono unit-testing sono writeStatoMacchinario() (metodo server-side) e parsingMacchinari() (metodo client-side che si occupa di ritornare una lista di macchinari partendo dalla stringa JSON ricevuta dal Controller Aggregatore).

Per testare il metodo “writeStatoMacchinario()” si è dato in input diversi macchinari, eseguendo quindi il metodo più volte e controllando poi che la lista venga correttamente aggiornata.

Per testare il metodo “parsingMacchinari()” si è dato in input una stringa JSON rappresentante diversi macchinari controllando che la lista ritornata fosse corretta.

COPERTURA DEI TEST

Copertura del test effettuato sul Controller Dashboard:



Copertura del test effettuato sul Controller Aggregatore lato server:

