



>Three Way Milkshake_

Verbale esterno 2

22 Gennaio 2020

Three Way Milkshake - Progetto "PORTACS"

threewaymilkshake@gmail.com

Versione	0.1.0
Stato	Approvato
Uso	Esterno
Approvazione	
Redazione	Greggio Nicolò
Verifica	
Destinatari	Three Way Milkshake Prof. Vardanega Tullio Prof. Cardin Riccardo

Descrizione

Verbale del meeting del 2021-01-22 del gruppo Three Way Milkshake con il proponente Sanmarco Informatica.



>Three Way
Milkshake_

Verbale esterno 2

Registro delle modifiche

Versione	Descrizione	Data	Nominativo	Ruolo
0.1.0	Stesura del verbale	2021-01-22	Greggio Nicolò	Redattore



Indice

1	Informazioni generali	3
1.1	Dettagli sull'incontro	3
1.2	Ordine del giorno	3
2	Verbale della riunione	4
2.1	Sensori ed unità	4
2.2	Tecnologie e formazione	4
2.2.1	Java	4
2.2.2	Node.js	4
2.2.3	Frontend	4
2.2.4	Design pattern	4
2.2.5	Salvataggio dati	5
2.2.6	Approccio al multithreading	5
2.2.7	Simulazione muletti	5
2.2.8	Controllo diagrammi attività server	5
2.2.9	Possibili requisiti qualitativi	5
2.2.10	Strategie di progettazione	6
3	Tracciamento temi affrontati	7



1 Informazioni generali

1.1 Dettagli sull'incontro

- **Luogo:** Incontro telematico tramite piattaforma Google Meet;
- **Data:** 2021-01-22;
- **Ora di inizio:** 16:00;
- **Ora di fine:** 16:45;
- **Partecipanti interni:** (6/6)
 - Chiarello Sofia;
 - Crivellari Alberto;
 - De Renzis Simone;
 - Greggio Nicolò;
 - Tessari Andrea;
 - Zuccolo Giada.
- **Partecipanti esterni:** (1)
 - Beggiato Alex (Sanmarco Informatica).

1.2 Ordine del giorno

La riunione prevede la discussione con il proponente dei seguenti punti:

- sensori per unità;
- tecnologie possibili e fonti di formazione;
- salvataggio di dati;
- approccio al multithreading;
- simulazione muletti;
- controllo diagrammi attività algoritmo server;
- possibili requisiti qualitativi;
- strategie di progettazione.



2 Verbale della riunione

2.1 Sensori ed unità

- Nel contesto reale si rende naturalmente necessaria una parte di sensoristica
 - per gli scopi di questo progetto si può tralasciare;
- il sistema ha tutte le informazioni di cui ha bisogno
 - il server centrale si occupa di tutto;
- non è necessario simulare i sensori.

2.2 Tecnologie e formazione

Sono state confermate le tecnologie discusse in confronti precedenti con il proponente, sono emerse diverse fonti da sfruttare.

2.2.1 Java

- Corso html.it;
- si possono approfondire alcune novità delle versioni > 11
 - tuttavia versione 8 va bene.

2.2.2 Node.js

- Documentazione ufficiale;
- corsi su html.it.

2.2.3 Frontend

- Angular o angular js
- guide e documentazioni ufficiali su corrispondenti siti;
- typescript vs javascript
 - dipende dal tempo che ci vogliamo dedicare.

2.2.4 Design pattern

- Nessun vincolo su design pattern;
- potrebbe essere comodo il concetto di factory
 - eventualmente bypassato da librerie;
 - ma in generale come pattern più pulito è più pratico;
 - adatto per sistemi composti da tanti piccoli componenti.



2.2.5 Salvataggio dati

- Non è per forza necessaria una base di dati;
- la configurazione iniziale può essere ricevuta da un file di testo;
- consigliate strutture noSQL comunque;
- semplici JSON possono essere sufficienti;
- non è richiesto il tracciamento delle operazioni.

2.2.6 Approccio al multithreading

- L'introduzione di framework in questo ambito può portare ad una elevata complessità;
- preferire come approccio l'uso di thread puri
 - timer task, runnable...;
- solo in caso di difficoltà valutare e discutere l'introduzione di un framework per il multithreading;
- limitare l'uso in generale di librerie di terze parti.

2.2.7 Simulazione muletti

- Carta bianca, si può usare tutto ciò che può dare una mano:
- in questo contesto anche diverse librerie, anche se ne sfruttiamo solo una piccola percentuale.

2.2.8 Controllo diagrammi attività server

- Bene in generale;
- quando si introdurrà il concetto di tempo e timer task utilizzare apposita icona start con orologio.

2.2.9 Possibili requisiti qualitativi

- In genere 2 famiglie:
 - bontà software
 - * indicatori comuni sul sw;
 - * non serve stare dentro certi limiti, basta misurarli;
 - * editor moderni fanno quasi tutto da soli;
 - applicativi
 - * requisito sensato può essere: $\frac{\text{tempo risposta}}{\text{numero dispositivi}}$;
 - * e.g.: entro 1 sec fino a 50, entro 2 fino a 100...;
 - * sistema qualità basato su tempi di risposta e numero di fail, questi ultimi in ogni caso non devono mai bloccare l'applicativo (input received, no output producecd);



- decisioni spettano al gruppo
 - i muletti vanno ad una certa velocità;
 - considerando tempi di reazione e velocità conseguirà uno spazio di spostamento;
 - stabilire in quanto spazio massimo si vuole l'arresto;
 - ricavare tempi di risposta che si dovranno attendere dal sistema;
 - discorso analogo per il numero di fail.

2.2.10 Strategie di progettazione

- inizialmente si consiglia di procedere in parallelo:
 - 3 progettazione;
 - 3 studio tecnologie ed approccio codice;
- questo da vantaggi;
- valutando eventuali scostamenti, riequilibrare le partizioni.



3 Tracciamento temi affrontati

Codice	Domanda	Risposta
VE_2.1	Simulazione sensori	Non serve
VE_2.2	Fonti per studio tecnologie	Principalmente siti e documentazioni ufficiali e html.it
VE_2.3	Design pattern	Nessuna richiesta particolare, factory può essere utile
VE_2.4	Salvataggio dati	Non serve una base di dati, basta un file JSON con le configurazioni necessarie
VE_2.5	Approccio multithreading	Thread e costrutti puri di java
VE_2.6	Simulazione muletti	Carta bianca
VE_2.7	Requisiti qualitativi	Bontà software e applicativi, decisioni specifiche spettano al gruppo