



# >Three Way Milkshake\_

---

## Verbale esterno 6

### 23 Marzo 2021

---

#### Three Way Milkshake - Progetto "PORTACS"

threewaymilkshake@gmail.com

<b>Versione</b>	0.1.0
<b>Stato</b>	Non approvato
<b>Uso</b>	Esterno
<b>Approvazione</b>	–
<b>Redazione</b>	De Renzis Simone
<b>Verifica</b>	Greggio Nicolò
<b>Destinatari</b>	Three Way Milkshake Prof. Vardanega Tullio Prof. Cardin Riccardo

#### **Descrizione**

Verbale del meeting del 2021-03-23 del gruppo Three Way Milkshake con il  
proponente Sanmarco Informatica.



>Three Way  
Milkshake\_

Verbale esterno 6

## Registro delle modifiche

Vers.	Descrizione	Redazione	Data red.	Verifica	Data ver.
0.1.0	Stesura del verbale	De Renzis Simone	2021-03-26	Greggio Nicolò	——



## Indice

<b>1</b>	<b>Informazioni generali</b>	<b>3</b>
1.1	Dettagli sull'incontro . . . . .	3
1.2	Ordine del giorno . . . . .	3
<b>2</b>	<b>Verbale della riunione</b>	<b>4</b>
2.1	Organizzazione dell'architettura del software . . . . .	4
2.1.1	Server . . . . .	4
2.1.2	Client . . . . .	4
2.2	Analisi dei design pattern da adottare . . . . .	4
2.3	Guida manuale . . . . .	5
<b>3</b>	<b>Tracciamento temi affrontati</b>	<b>6</b>



# 1 Informazioni generali

## 1.1 Dettagli sull'incontro

- **Luogo:** Incontro telematico tramite piattaforma Google Meet;
- **Data:** 2021-03-23;
- **Ora di inizio:** 17:00;
- **Ora di fine:** 18:00;
- **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:

- organizzazione dell'architettura del software;
- analisi dei design pattern da adottare;
- guida manuale.



## 2 Verbale della riunione

### 2.1 Organizzazione dell'architettura del software

È stata intavolato un confronto con il proponente per definire alcuni dettagli riguardanti l'architettura del software.

#### 2.1.1 Server

Per quanto riguarda la componente server, sono stati definiti 3 livelli su cui si può articolare l'architettura:

1. **layer di comunicazione:** gestisce la comunicazione tra il server e i client, e si può specializzare in due sezioni: la prima può essere gestita tramite Socket, e riguarderà l'invio dei dati per il monitor real-time (di visualizzazione del magazzino con i muletti che si muovono al suo interno); la seconda, gestibile tramite API di tipo REST, regola l'interazione dei due attori "Amministratore" e "Responsabile" con l'interfaccia grafica. Per approcciarsi a questa pratica, il proponente suggerisce di usufruire della libreria "Jersey" di Java, indicando come possibile fonte di studio una guida dedicata sul sito [html.it](http://html.it);
2. **layer di business:** il motore di calcolo del sistema;
3. **layer di persistenza:** per la gestione della persistenza, nel caso del nostro applicativo per il momento è previsto il salvataggio in file *.json*. È stato evidenziato come sia importante assicurare, a questo livello, la maggior indipendenza possibile dagli altri layer per consentire, in un futuro, l'estensione ad altri tipi di database.

#### 2.1.2 Client

I client si differenziano a seconda della tipologia:

1. i muletti prevedono una componente da realizzare in Node.js per interfacciarsi con le Socket con cui comunica il server;
2. per "Amministratore" e "Responsabile" è sufficiente la realizzazione dell'interfaccia in Angular, in quanto essa è in grado di comunicare tramite le API REST esposte dal server.

### 2.2 Analisi dei design pattern da adottare

Sono stati evidenziate alcune componenti del server che potrebbero essere modellate nel design pattern **Singleton**:

1. path finder: che incapsula l'algoritmo per la ricerca del percorso migliore;
2. collision detection: per la gestione delle collisioni;
3. warehouse map: la rappresentazione interna della mappa del magazzino.

Per queste classi è necessario assicurare la presenza di un'unica istanza condivisa, motivo per cui il pattern Singleton può essere impiegato efficacemente.

Per quanto riguarda ConnectionAcceptor (classe dedicata ad accettare le connessioni entranti) è stata pensata la possibilità di introdurre un pattern di tipo **Factory**.



## 2.3 Guida manuale

È stata discussa l'implementazione della guida manuale per i muletti: in particolare è stato evidenziato come non debba esserci alcuna attesa di feedback da parte del server nel movimento azionato da guida manuale. Tuttavia è comunque necessario, per le unità guidate manualmente, la comunicazione con il server della propria posizione, necessarie per il calcolo (ed eventualmente, i ricalcoli) del percorso migliore per raggiungere la destinazione. Una possibilità implementativa per assicurare queste caratteristiche è l'introduzione di un timer interno al client che regoli lo spostamento dell'unità sulla base degli input utente ricevuti.



### 3 Tracciamento temi affrontati

Codice	Decisione
VE_6.1	Architettura del server a 3 layer
VE_6.2	Comunicazione con i Muletti avvengono tramite Socket (richiedono componente in Node.js)
VE_6.3	Comunicazione con Amministratore e Responsabile avvengono tramite REST API
VE_6.4	Path finder, collision detection e warehouse map possono beneficiare del design pattern Singleton.
VE_6.5	ConnectionAcceptor può beneficiare del design pattern Factory.
VE_6.6	Possibilità di introdurre un timer interno ai client per temporizzare gli spostamenti in guida manuale.