



>Three Way Milkshake_

Verbale esterno 5

16 Marzo 2021

Three Way Milkshake - Progetto "PORTACS"

threewaymilkshake@gmail.com

Versione	1.0.0
Stato	—
Uso	Esterno
Approvazione	—
Redazione	Zuccolo Giada
Verifica	Chiarello Sofia
Destinatari	Three Way Milkshake Prof. Vardanega Tullio Prof. Cardin Riccardo

Descrizione

Verbale del meeting del 2021-03-16 del gruppo Three Way Milkshake con il
proponente Sanmarco Informatica.



>Three Way
Milkshake_

Verbale esterno 5

Registro delle modifiche

Vers.	Descrizione	Redazione	Data red.	Verifica	Data ver.
0.1.0	Stesura del verbale	Zuccolo Giada	2021-03-16	Chiarello Sofia	2021-03-19



Indice

1	Informazioni generali	3
1.1	Dettagli sull'incontro	3
1.2	Ordine del giorno	3
2	Verbale della riunione	4
2.1	Discussione su architettura	4
2.2	Docker	4
2.3	Sicurezza	4
2.4	Confronto sull'algoritmo per la rilevazione delle collisioni	4
2.5	PoC	4
3	Tracciamento temi affrontati	5



1 Informazioni generali

1.1 Dettagli sull'incontro

- **Luogo:** Incontro telematico tramite piattaforma Google Meet;
- **Data:** 2021-03-16;
- **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:

- discussione su architettura;
- Docker;
- sicurezza;
- confronto sull'algoritmo per la rilevazione delle collisioni;
- Proof of Concept.



2 Verbale della riunione

2.1 Discussione su architettura

Vengono discussi alcuni modelli di design patter da poter utilizzare. E' emerso quanto segue:

- Observer
in quanto il segnale deve essere gestito a seconda della sua tipologia;
- Layer:
Il proponente consiglia di non usare più di 5 layer.
Più precisamente vengono indicati dal proponente:
 - DAO (Data Access Object);
 - Business Logic;
 - Strato servizi che riceve segnali HTTP;
 - Layer di accesso al DB sono poco utilizzati;
- il proponente ha consigliato al gruppo di visualizzare il Singleton Pattern;
- il proponente ha consigliato al gruppo di visualizzare il Factory Pattern;
- no microservizi poichè si aggiunge tempo di latenza, quindi sono inutili ed è più difficile da creare. Il proponente consiglia un sistema più simile a un monolite.

2.2 Docker

Secondo il proponente il client va fatto containerizzando istanze di Node e Angular.

2.3 Sicurezza

Discussione su come fare per la comunicazione sicura.

- Si consiglia Java socket in HTTPS;
- procedura:
lato java → collegarsi ad un'api rest in http → fatto in automatico → serve il certificato da chi viene invocato → scambio chiavi → canale crittografato;
- utilizzare Secure Socket Layer;
- il proponente consiglia:
<https://docs.oracle.com/javase/10/security/sample-code-illustrating-secure-socket-connection-client-and-server.htm#JSSEC-GUID-AA1C27A1-2CA8-4309-B281-D6199F60E666>;

2.4 Confronto sull'algoritmo per la rilevazione delle collisioni

Mostrato al proponente la proposta di algoritmo prodotta per il rilevamento delle collisioni. Il proponente ha appoggiato la proposta e ha consigliato di fare autonomamente l'algoritmo senza cercare soluzioni accademiche ritenute troppo complicate.

2.5 PoC

E' stato mostrato al proponente il PoC e il suo funzionamento.

3 Tracciamento temi affrontati

Codice	Domanda	Risposta
VE_5.1	Design Pattern	Discussione su varie architetture possibili <ul style="list-style-type: none">• Observer;• Singleton Pattern;• Factory Pattern;• no microservizi;• sistema più simile a monolite; Discussione sui Layer
VE_5.2	Docker	Client fatto containerizzando istanze di Node e Angular.
VE_5.3	Sicurezza	<ul style="list-style-type: none">• Java socket in HTTPS;• usare SSL;• Link.
VE_5.4	Algoritmo rilevazione collisioni	<ul style="list-style-type: none">• proposta corretta;• non cercare soluzioni accademiche.