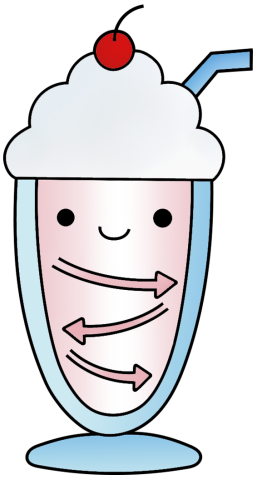

>Three Way Milkshake_

Verbale esterno 7
9 Aprile 2021

Three Way Milkshake - Progetto "PORTACS"

threewaymilkshake@gmail.com

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

Descrizione

Verbale del meeting del 2021-04-09 del gruppo Three Way Milkshake con il proponente Sanmarco Informatica.

Registro delle modifiche

Vers.	Descrizione	Redazione	Data red.	Verifica	Data ver.
0.1.0	Stesura del verbale	De Renzis Simone	2021-04-15	Greggio Nicolò	———

Indice

1 Informazioni generali

1.1 Dettagli sull'incontro

- **Luogo**: Incontro telematico tramite piattaforma Google Meet;
- **Data**: 2021-04-09;
- **Ora di inizio**: 14:30;
- **Ora di fine**: 16:15;
- **Partecipanti interni**: (6/6)
  - Chiarello Sofia;
  - Crivellari Alberto;
  - De Renzis Simone;
  - Greggio Nicolò;
  - Tessari Andrea;
  - Zuccolo Giada.
- **Partecipanti esterni**: (2)
  - Beggiato Alex (Sanmarco Informatica);
  - Piva Alessandra (Sanmarco Informatica);

1.2 Ordine del giorno

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

- esposizione dell'architettura del software;
- breve dimostrazione dell'eseguibile fin'ora prodotto;
- chiarimenti su Docker;
- modalità di parcheggio dei muletti;
- chiusura del server.

2 Verbale della riunione

2.1 Esposizione dell'architettura del software

Il meeting è iniziato con l'esposizione dell'archiettura del software modellata dal gruppo nelle ultime settimane. Sono stati quindi mostrati i diagrammi UML relativi alle componenti **client**: l'esposizione è stata motivo di conferma sulla bontà dell'applicazione del pattern architetturale *MVC* nel contesto dei client. É stato spiegato il funzionamento di ognuno dei moduli.

Il proponente ha inoltre chiarito come sia preferibile predisporre un modulo unico (lato client) dedicato al login, piuttosto che averne due separati (uno per l'Amministratore e uno per il Responsabile).

Si è proceduto poi a mostrare il diagramma UML del **server** e a spiegarne velocemente il funzionamento. La discussione si è focalizzata sui design pattern applicati (soprattutto Singleton): il proponente condivide le scelte fatte, e chiede dei chiarimenti in merito al modulo di gestione delle *Task*. L'esposizione non può raggiungere una profondità troppo elevata, anche a causa dei limiti temporali della riunione.

2.2 Breve dimostrazione dell'eseguibile fin'ora prodotto

Approfittando della presenza concordata di Alessandra Piva, Responsabile della logistica presso l'azienda proponente Sanmarco Informatica, il gruppo ha predisposto una breve dimostrazione dell'eseguibile fin'ora realizzato. Sono stati mostrati il monitor di visualizzazione real-time dei muletti, testando i movimenti di 3 muletti al raggiungimento delle task assegnate. É stato messo alla prova il sistema di gestione delle collisioni e di calcolo del percorso; per poi passare alla guida manuale di un muletto.

L'esposizione ha avuto riscontro positivo da parte del proponente, che ha evidenziato come il "core" dell'applicazione sia stato realizzato coerentemente ai requisiti prefissati. É stato suggerito di distinguere i muletti in circolazione colorando in modo diverso (e coerente) le icone dei muletti.

2.3 Chiarimenti su Docker

- dockerfile vanno customizzati/aggiunte configurazioni manualmente per ogni forklift aggiunto (va bene vero?)
  - creare cartella condivisa tra file system e container dove metto un file config.cfg
  - file con stesso nome
  - si può anche gestire configurazione porta con file o passare parametro ambiente
  - in ogni caso PASSARE TUTTO ALLO STESSO MODO (per gli utenti manager/admin questo probabilmente non dev'essere vero)
- dockerfile da far girare su una macchina che faccia da server, l'ip di questa dev'essere conosciuto da tutti
- dockerfile per unità da far girare su dispositivo (associato ai muletto) che abbia broswer e possibilità di eseguire tutta la roba che serve (necessaria configurazione manuale per ogni nuova, ma dopotutto il numero è statico e deciso dall'admin quindi non dovrebbe essere problema)

- dockerfile da far girare su dispositivi dei responsabili/admin, che siano sempre in grado di far girare il tutto e un browser per usare l'interfaccia
- ogni unità (che sia forklift/admin/manager) fa girare al suo interno un mini server node che da una parte di connette per gestire la comunicazione al suo esterno con il server java che tutti conoscono, e all'interno tramite un altro server http fa appoggiare socket.io per parlare con l'interfaccia ng e fare cose

2.4 Modalità di parcheggio dei muletti

É stata chiesta conferma al proponente sulla modalità di gestione dei muletti quando essi dovessero rimanere in uno stato di "pausa" perchè non presenti nuove *Task* assegnabili: essi possono dirigersi in una località esterna alla mappa, tramite un passaggio nella stessa, nella quale sia previsto il "parcheggio" delle unità. Esse rimangono a disposizione per nuovi compiti da svolgersi qual ora questi venissero aggiunti dal Responsabile.

2.5 Proposta di file *.json* per la persistenza

É stata presentata una proposta di file *.json* da utilizzarsi per la persistenza dei dati di autenticazione dei muletti e degli utenti. La struttura si è rivelata coerente con quanto atteso, il proponente ha segnalato come non sia necessario conservare un token di password per i muletti.

2.6 Chiusura del server

L'applicativo non dispone di un modo per interrompere la componente server: per ora è prevista la chiusura tramite interruzione del processo (*ctrl-c* da terminale). Il proponente non pone vincoli su questo aspetto, ma pone l'attenzione sull'importanza di mantenere uno stato coerente dopo la chiusura per quanto riguarda i dati salvati. Siccome la persistenza viene mantenuta in memoria durante l'esecuzione, e le scritture su file avvengono ad ogni modifica, l'interruzione dell'applicazione server non dovrebbe mettere a rischio la corretta conservazione dei dati.

3 Tracciamento delle decisioni

Codice	Decisione
VE_7.1	Unico modulo per l'autenticazione (lato client).
VE_7.2	Architettura dei client secondo il pattern MVC è corretta.
VE_7.3	Architettura del server risulta sostanzialmente corretta.
VE_7.4	Il core dell'applicazione è realizzato coerentemente con i requisiti prefissati.
VE_7.5	Il parcheggio delle unità può avvenire in un'area esterna alla mappa accessibile tramite un passaggio presente sulla stessa.
VE_7.6	Docker
VE_7.7	Docker
VE_7.8	I muletti non necessitano di password per l'autenticazione.
VE_7.9	Importante mantenere coerente il salvataggio dei dati in caso di chiusura del server.