>Three Way Milkshake_

Manuale Manutentore

Three Way Milkshake - Progetto "PORTACS"

threewaymilkshake@gmail.com

Versione	0.0.1
Stato	Non approvato
Uso	Esterno
Approvazione	———
Redazione	———
Verifica	———
Destinatari	Sanmarco Informatica Prof. Vardanega Tullio Prof. Cardin Riccardo Three Way Milkshake

Descrizione

Manuale di supporto allo sviluppo e manutenzione del software G PORTACS

Registro delle modifiche

Vers.	Descrizione	Redazione	Data red.	Verifica	Data ver.
0.2.1	Incremento § 5.2	Zuccolo Giada	2021-04-22	———	———
0.2.0	Stesura § 5.2	Tessari Andrea	2021-04-22	———	———
0.1.0	Stesura § 1	Tessari Andrea	2021-04-22	———	———
0.0.1	Impaginazione	De Renzis Simone	2021-04-15	———	———

Indice

1 Introduzione **6**
1.1 Scopo del documento . . . 6
1.2 Scopo del prodotto . . . 6
1.3 Riferimenti . . . 6
1.3.1 Normativi . . . 6
1.3.2 Informativi . . . 7

2 Tecnologie e librerie **8**
2.1 Server . . . 8
2.1.1 Tecnologie . . . 8
2.1.2 Librerie e Framework . . . 8
2.2 Client . . . 8
2.2.1 Tecnologie . . . 8
2.2.2 Librerie e Framework . . . 8
2.3 Versionamento e Continuous Integration . . . 8

3 Setup **9**
3.1 Requisiti di sistema . . . 9
3.1.1 Requisiti Hardware . . . 9
3.1.2 Requisiti Software . . . 9
3.2 Installazione . . . 9

4 Testing **10**
4.1 JUnit . . . 10
4.2 Libreria test frontend . . . 10

5 Architettura del sistema **11**
5.1 Server . . . 11
5.1.1 Diagramma delle classi . . . 11
5.1.1.1 Persistance layer . . . 11
5.1.1.2 Business layer . . . 11
5.1.1.2.1 Mappa . . . 11
5.1.1.2.2 Clients . . . 11
5.1.1.2.3 Tasks . . . 11
5.1.1.2.4 Collisioni . . . 11
5.1.1.3 Communication layer . . . 11
5.2 Client . . . 12
5.2.1 Diagramma di classe per l'unità . . . 12
5.2.2 Diagramma di classe per l'admin-manager . . . 13
5.3 Comunicazione . . . 14
5.3.1 Diagrammi di sequenza . . . 14
5.3.2 Protocollo di comunicazione . . . 14

6 Estendere PORTACS **15**
6.1 Algoritmo alternativo per il path finding . . . 15
6.2 Introdurre nuove tipologie di utenti . . . 15
6.2.1 Lato server . . . 15
6.2.2 Lato client . . . 15

6.3 Implementare tipi di persistenza alternativi 15
6.4 Modificare handler nell'algoritmo di gestione delle collisioni 15

Elenco delle figure

5.2.1 Diagramma UML delle classi per l'unità 12
5.2.2 Diagramma UML delle classi per gli admin e i manager 13

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è presentare tutte le informazioni necessarie al mantenimento e all'estensione del software PORTACS, mostrando nel dettaglio l'architettura del sistema e l'organizzazione del codice sorgente.
In questo documento saranno presentate le varie tecnologie usate, sia lato front end che back end, come anche le varie librerie e framework. Verrà inoltre mostrato il sistema di versionamento utilizzato e la Continuous Integration applicata.

1.2 Scopo del prodotto

Il capitolato$_G$ C5 propone un progetto$_G$ in cui viene richiesto lo sviluppo di un software per il monitoraggio in tempo reale di unità che si muovono in uno spazio definito. All'interno di questo spazio, creato dall'utente per riprodurre le caratteristiche di un ambiente reale, le unità dovranno essere in grado di circolare in autonomia, o sotto il controllo dell'utente, per raggiungere dei punti di interesse posti nella mappa. La circolazione è sottoposta a vincoli di viabilità e ad ostacoli propri della topologia dell'ambiente, deve evitare le collisioni con le altre unità e prevedere la gestione di situazioni critiche nel traffico.

1.3 Riferimenti

1.3.1 Normativi

- Norme di progetto$_G$ v3.0.0 : per qualsiasi convenzione sulla nomenclatura degli elementi presenti all'interno del documento;
- Regolamento progetto$_G$ didattico:
  https://www.math.unipd.it/~tullio/IS-1/2020/Dispense/P1.pdf;
- Model-View Patterns:
  https://www.math.unipd.it/~rcardin/sweb/2020/L02.pdf;
- SOLID Principles:
  https://www.math.unipd.it/~rcardin/sweb/2020/L04.pdf;
- Diagrammi delle classi:
  https://www.math.unipd.it/~rcardin/swea/2021/DiagrammidelleClassi_4x4.pdf;
- Diagrammi dei package:
  https://www.math.unipd.it/~rcardin/swea/2021/DiagrammideiPackage_4x4.pdf;
- Diagrammi di sequenza:
  https://www.math.unipd.it/~rcardin/swea/2021/DiagrammidiSequenza_4x4.pdf;
- Design Pattern Creazionali:
  https://www.math.unipd.it/~rcardin/swea/2021/DesignPatternCreazionali_4x4.pdf;
- Design Pattern Strutturali:
  https://www.math.unipd.it/~rcardin/swea/2021/DesignPatternStrutturali_4x4.pdf;

- Design Pattern Comportamentali:
https://www.math.unipd.it/~rcardin/swea/2021/DesignPatternComportamentali_4x4.pdf.

1.3.2 Informativi

- Glossario: per la definizione dei termini (pedice G) e degli acronimi (pedice A) evidenziati nel documento;
- Capitolato d'appalto C5-PORTACS:
https://www.math.unipd.it/~tullio/IS-1/2020/Progetto/C5.pdf
- Software Engineering - Iam Sommerville - 10^{th} Edition.
- Angular:
https://angular.io/;
- Node.js:
https://nodejs.org/en/;
- PrimeNG:
https://www.primefaces.org/primeng/;
- Java:
https://www.java.com/it/;
- Spring:
https://spring.io/;
- Docker:
https://www.docker.com/.

2 Tecnologie e librerie

2.1 Server

2.1.1 Tecnologie

- **Java**
- **Json**
- **Docker**
- **Gradle**

2.1.2 Librerie e Framework

- **Spring**
- **Gson**
- **Junit**
- **Mockito**

2.2 Client

2.2.1 Tecnologie

- **Node.js**
- **HTML**
- **CSS**
- **Typescript**

2.2.2 Librerie e Framework

- **Angular**
- **PrimeNG**
- **Libreria di test1**
- **Libreria di test2**

2.3 Versionamento e Continuous Integration

- **GitHub**
- **GitHub Action**

3 Setup

3.1 Requisiti di sistema

3.1.1 Requisiti Hardware

3.1.2 Requisiti Software

3.2 Installazione

4 Testing

4.1 JUnit

4.2 Libreria test frontend

5 Architettura del sistema

Qui si potrebbe mettere uno schema simile a quello della slide iniziale per evidenziare l'architettura client-server

5.1 Server

Dire che è 3 layer architecture

Qui potrebbe esserci il diagramma minimale complessivo

5.1.1 Diagramma delle classi

5.1.1.1 Persistance layer

5.1.1.2 Business layer

5.1.1.2.1 Mappa

5.1.1.2.2 Clients

5.1.1.2.3 Tasks

5.1.1.2.4 Collisioni

5.1.1.3 Communication layer

5.2 Client

5.2.1 Diagramma di classe per l'unità

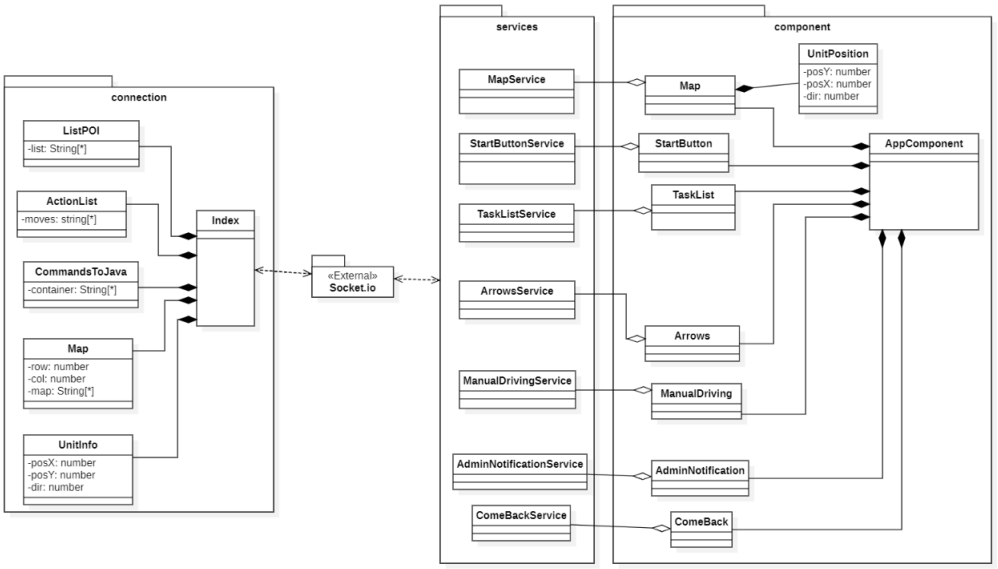

Figura 5.2.1: Diagramma UML delle classi per l'unità

Qui utilizziamo due tecnologie: Node per quanto riguarda il package connection e Angular per il resto. Queste due parti del front end comunicano attraverso il package esterno Socket.io, necessario gestire un flusso di dati attraverso i socket.
Il package services, come descritto anche nella documentazione di Angular, fa da intermediario tra il package connection e il package component, utilizzando degli Observer in ascolto di uno specifico socket e instradando l'informazione verso l'opportuno component.
Il package component permette di visualizzare sullo schermo le informazioni richieste grazie anche ai template di Angular. Ogni classe di questo package serve ad una specifica funzionalità:

- Map -> visualizza la mappa del magazzino con la posizione in real time dell'unità;
- StartButton -> mostra un bottone che serve a far partire l'unità;
- TaskList -> mostra la lista di task che l'operatore dovrà compiere;
- Arrows -> visualizza le azioni che compie l'unità in real time;
- ManualDrive -> permette di cambiare guida da manuale ad automatica e viceversa, facendo visualizzare anche i pulsanti da premere per far muovere l'unità manualmente in caso;
- AdminNotification -> visulizza un pulsante che, se premuto, notifica all'admin un evento eccezionale;
- ComeBack -> mostra un pulsante alla fine del turno dell'operatore che, se premuto, guida automaticamente il muletto verso la propria base.

Il package connection, attraverso le classi Index e CommandsToJava, instaura inoltre una comunicazione TCP Socket con java, permettendo di creare una connessione tra frontend e backend.

5.2.2 Diagramma di classe per l'admin-manager

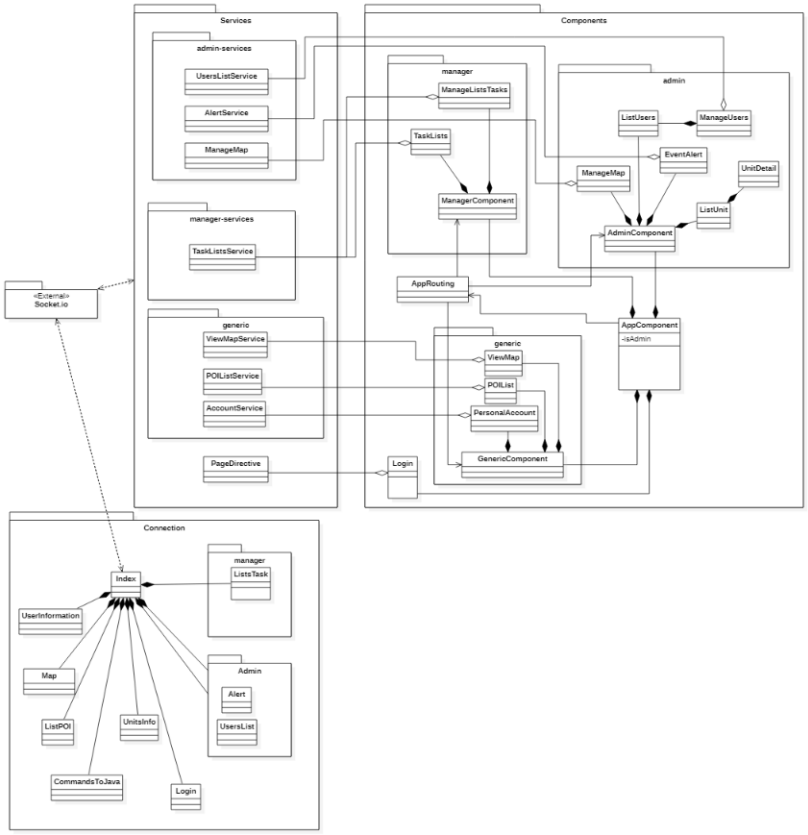

Figura 5.2.2: Diagramma UML delle classi per gli admin e i manager

Il diagramma di classe dell'admin-responsabile è molto simile a quello dell'unità: si avvale delle tecnologie Node, con il package connection, ed Angular, con tutti gli altri package.
Il contesto dei package è lo stesso dell'unità, di cui però si fa una differenziazione tra le funzionalità del'admin e quelle del manager grazie alla classe Login e al routing di AppRouting presente nel package component: permette all'utente di effettuare il login, "attivando" solamente le funzionalità riferite al tipo di utente loggato.
Il package generic contiene classi di funzionalità condivise tra amnagere e admin.
Le classi presenti in component permettono di attivare certe classi riferite al package (e quindi alle funzionalità di uno specifico tipo di utente):

- admin:
  - ListUsers -> aggiunta o rimozione di manager;

  - EventAlert -> visualizzazione eventi eccezionali;
  - ManageMap -> modifica la planimetria e le caratteristiche della mappa;
- manager:
  - TaskLists -> visualizzazione delle liste di task;
  - MangeListsTask -> aggiunta, modifica e rimozione di liste di task;
- generic (sia per admin che per manager):
  - ViewMap e POIList -> visualizzazione in real time di tutte le unità nel magazzino.

5.3 Comunicazione

5.3.1 Diagrammi di sequenza

5.3.2 Protocollo di comunicazione

6 Estendere PORTACS

6.1 Algoritmo alternativo per il path finding

6.2 Introdurre nuove tipologie di utenti

6.2.1 Lato server

6.2.2 Lato client

6.3 Implementare tipi di persistenza alternativi

6.4 Modificare handler nell'algoritmo di gestione delle collisioni