UNIVERSITA' DEGLI STUDI DI NAPOLI "PARTHENOPE"

CdL in Informatica

Relazione Progetto di Programmazione e Lab di Programmazione 3

CPU Spedizioni

Docenti
Angelo Ciaramella
Raffaele Montella

Candidato
Antonio Di Marino
Mat: 0124001344

Indice

1 Traccia 3
2 Analisi 4
2.1 Descrizione Progetto 4
2.2 Design Pattern 6
2.2.1 Facade 6
2.2.2 Factory Method 6
2.2.3 State 7
2.2.4 Visitor 7
3 Diagrammi 8
3.1.1 Diagramma pattern Facade 8
3.1.2 Diagramma pattern Factory Method 8
3.1.3 Diagramma pattern State 9
3.1.4 Diagramma pattern Visitor 10

Capitolo 1

Traccia

Si vuole sviluppare un'applicazione relativa alla consegna di merci nel campo della logistica. La logistica è l'insieme delle attività organizzative, gestionali e strategiche che governano i flussi di materiali e delle relative informazioni dalle origini presso i fornitori fino alla consegna dei prodotti finiti ai clienti e al servizio post-vendita.
Si suppone di avere diverse aziende di trasporto per consegnare la merce (corrieri). Ogni azienda ha a disposizione un numero di veicoli identificati da un codice, tipo veicolo e capienza container (numeri di colli che può contenere).
Il collo è identificato da un codice, mittente, destinatario e peso. L'applicazione deve gestire il carico di N colli nei container. Per il riempimento di utilizza un algortimo approssimato (Next Fit) che risolve il problema del Bin Packing (vedi documento allegato).
Il corriere, inoltre, aggiorna lo stato del collo ad ogni centro di smistamento, il quale deve essere rintracciato dal destinatario mediante il suo codice.

Capitolo 2

Analisi

2.1 Descrizione progetto

Una possibile soluzione per questo progetto e` stata ottenuta mediante l'implementazione di una web app.
Sqlite3 e` il database utilizzato per la gestione dei dati che vengono raccolti e/o elaborati dal software.
Un'altra componente della web app e` l'utilizzo dell'applicazione Open Source Apache Tomcat che permette di utilizzare servlet e pagine JSP, dandoci quindi l'opportunita` di sviluppare web app in Java.

Sono stati utilizzati due pattern Architetturali nella realizzazione di questa web app.

1. MVC Pattern
2. DAO Pattern

1. MVC Pattern
Model View Controller ci consente di separare la logica di programmazione dalla logica di business.

- Model: accede ai dati utili per l'applicazione a prescindere dall'interfaccia utente utilizzata.
- View(Pagine JSP): visualizza questi dati e li utilizza nell'interazione con gli utenti
- Controller(Servlet): riceve i comandi dall'utente tramite il View e li esegue.

2. DAO Pattern

Data Access Object ci consente di isolare l'accesso ad una tabella del DB, crea un maggior livello di astrazione.

Si e` deciso di implementare due servizi da offrire all' utente che sono la *Creazione di una Spezione* e il *Tracking*. Un' ulteriore servizio che si e` creato e` la *Sincronizzazione dei Veicoli* che effettuera` un generico Dipendente.
Non e` richiesta la registrazione per poter effettuare una spedizione, di conseguenza un utente puo' effettuare una spedizione compilando l'apposito form contenuto nella pagina JSP dedicata alle spedizioni(*spedizioni.jsp*).
Dopo aver effettuato quest'operazione, la web app elabora l'informazione, prendendo in carico la richiesta dell'utente.
Il Tracking e` un Tracking Approssimativo, in quanto si e` deciso di non utilizzare dei dipendenti per la ricezione dell'orario di consegna ma di approssimarla con calcoli automatici.
La Sincronizzazione dei Veicoli viene utilizzata dal dipendente per determinare in modo automatico quale veicolo deve partire e quale deve tornare dal viaggio.

2.2 Design Pattern

Lo scopo principale di questo progetto e` il corretto utilizzo e implementazione dei Design Pattern, di seguito troverete i pattern utilizzati con le rispettive funzioni.

2.2.1 Facade

Quando un cliente vuole spedire un pacco compila un form che viene poi trasmesso alla servlet, la servlet deve poi “contattare” ogni singola componente lato server per poter eseguire le operaizioni necessarie alla creazione di una spedizione. Si e` deciso quindi di delegare il tutto a una classe Facade, il pattern Facade viene appunto utilizzato per nascondere la complessita` del sistema e le dipendenze del Client.
Il client quindi delega la creazione di una Spedizione a una componente ad hoc.
La classe SpedizioneFacade si occupera` quindi di: Inserire il collo nel Database, ricostruire il percorso del pacco per arrivare a destinazione, determinare il veicolo con cui dovra` essere spedito e inviare un email sia al mittente che al destinatario con il codice per il tracking.

2.2.2 Factory Method

Nel momento in cui si crea una veicolo bisogna determinare se questo veicolo e` un autocarro o un furgone di conseguenza si e` scelto di utilizzare il factory method in modo da delegare al Creator la creazione dell’oggetto.

2.2.3 State

Quando il dipendente effettuera` la Sincronizzazione dei Veicoli, automaticamente i

veicoli cambieranno stato, in base a determinate condizioni. Gli stati possibili sono: Pronto e In Viaggio. I veicoli Pronti sono tutti i veicoli presenti nella loro sede di partenza mentre i veicoli in viaggio sono i veicoli impegnati nelle consegne dei colli. Utilizzando il pattern state si sono limitati tutti i controlli di selezione sugli stati dei veicoli. Automatizzando il comportamento degli oggetti in base al loro stato.

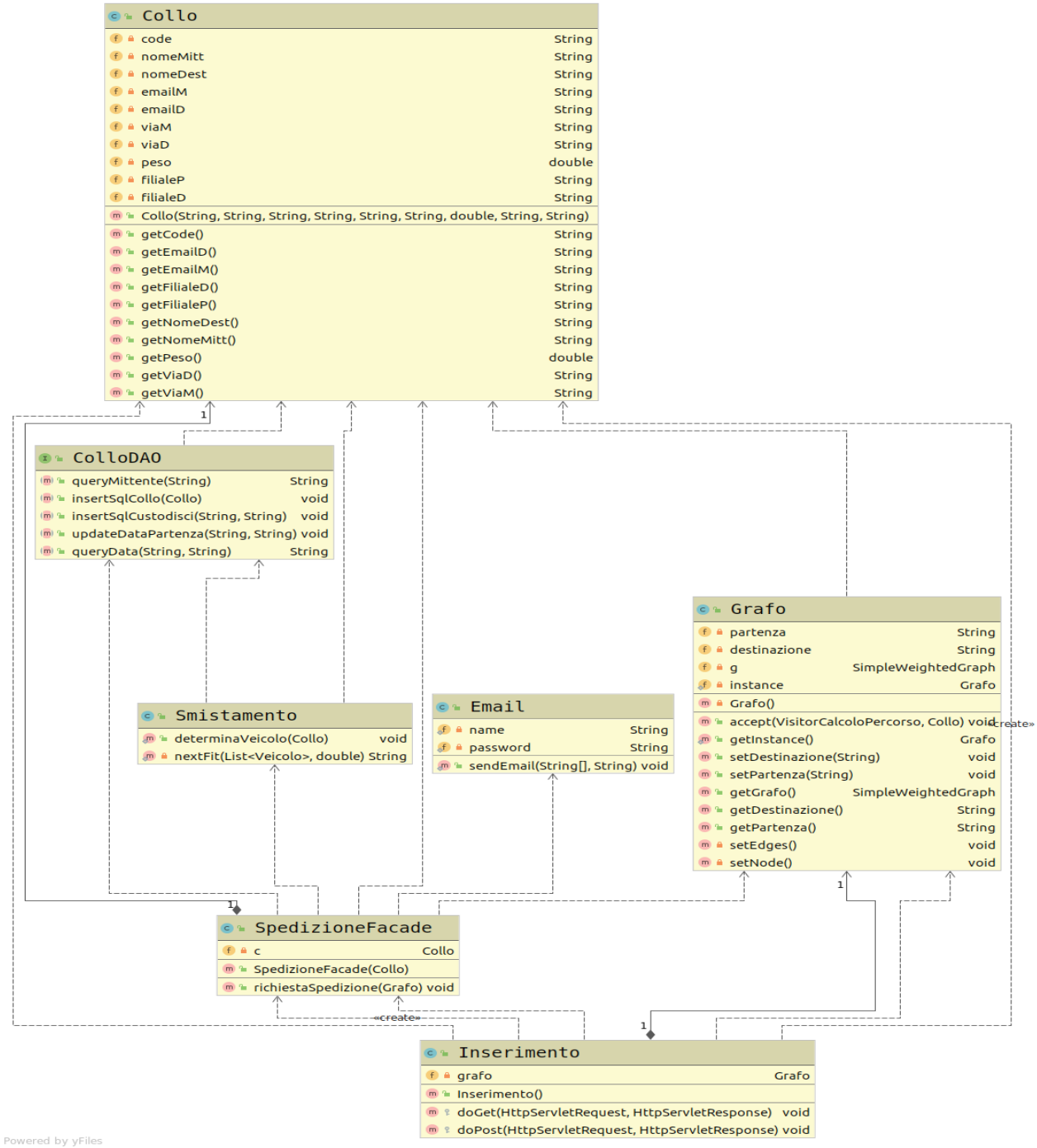
2.2.4 Visitor

Per la determinazione del percorso migliore da intraprendere per la consegna di un collo viene utilizzato l'algoritmo di Dijkstra. Considerando che la classe Grafo aderisce al principio solid di singola responsabilita` per l'implementazione del calcolo del percorso si e` deciso di "arricchire" il codice della classe grafo implementando il pattern visitor.

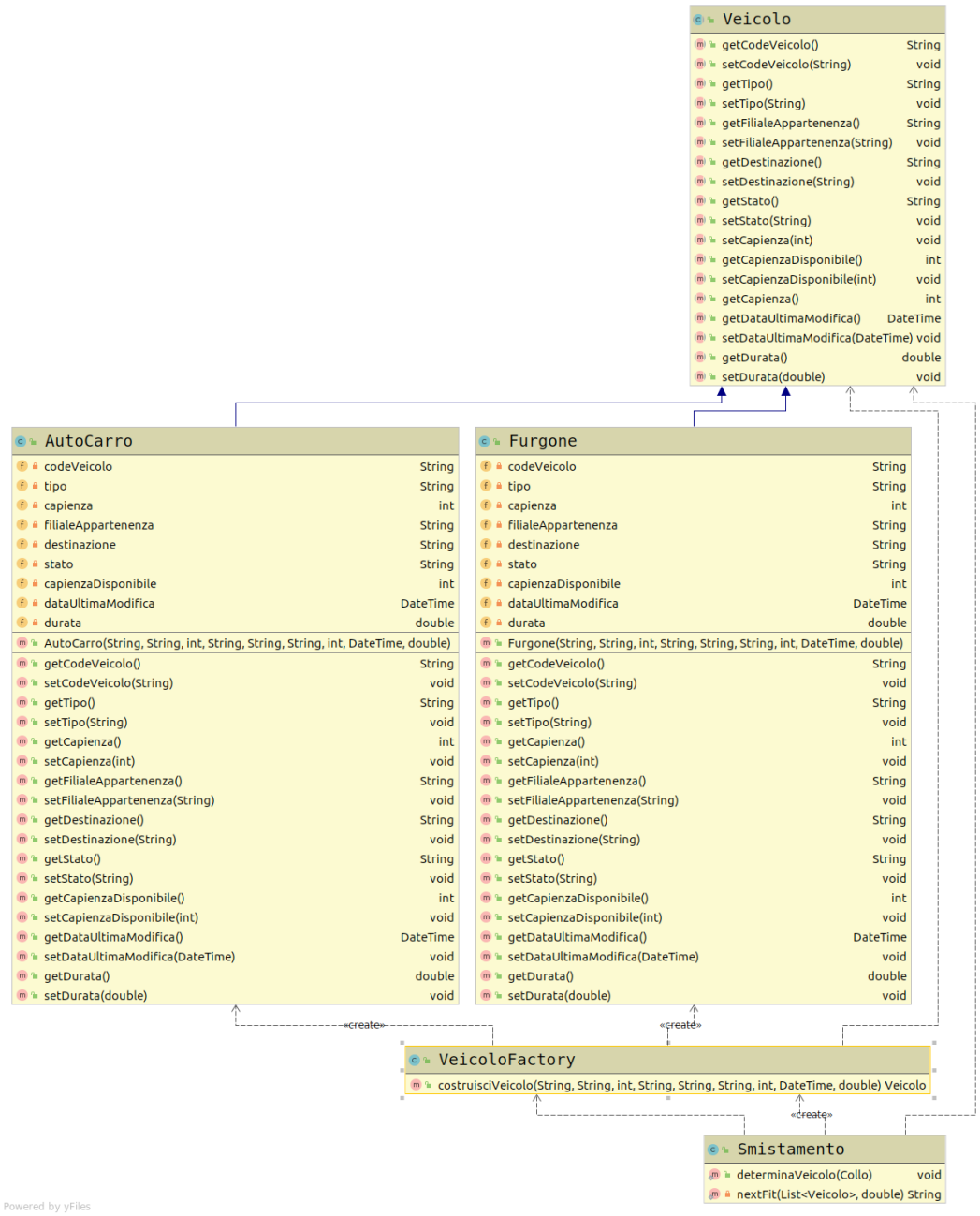
Capitolo 3

Diagrammi

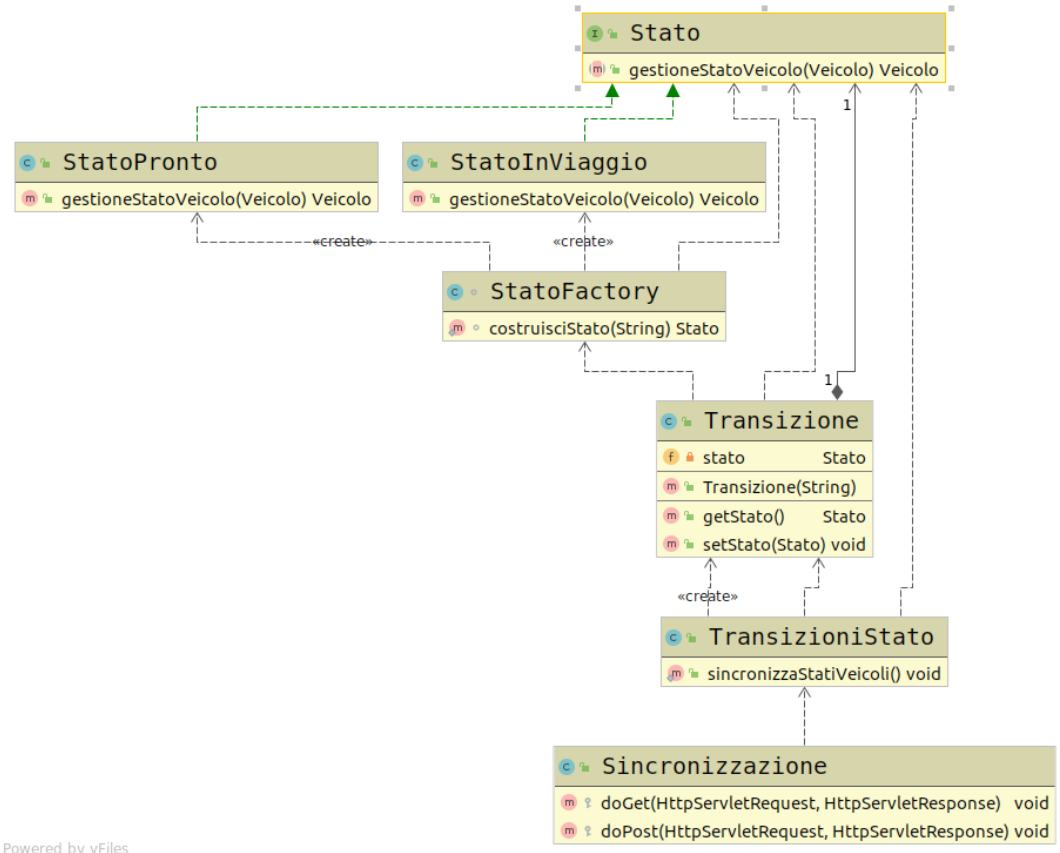
3.1.1 Diagramma pattern Facade



3.1.2 Diagramma pattern Factory Method



3.1.3 Diagramma pattern State



3.1.4 Diagramma pattern Visitor