



UNIVERSITA' DEGLI STUDI DI NAPOLI "PARTHENOPE"

FACOLTA' SCIENZE E TECNOLOGIE

CORSO DI LAUREA IN INFORMATICA

PROGETTO PROGRAMMAZIONE 3

LOGISTICAPP

DOCENTE

Angelo Ciaramella
Raffaele Montella

CANDIDATO

Giuseppe Nappo
MATR. 0124001686

ANNO ACCADEMICO 2020/2021

Sommario

1	Descrizione e requisiti del progetto	3
2	Requisiti.....	4
3	Diagramma dei casi d'uso	6
4	Struttura del progetto	7
5	Pattern Utilizzati	9
6	UML	12
7	Problema del Bin Packing	21
8	Algoritmo Next fit	21
9	Database Relazionale.....	22
10	Codice Progetto GitHub	23
11	Utilizzo del programma.....	23

1 Descrizione e requisiti del progetto

Logistica

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 si utilizza un algoritmo approssimato (Next Fit) che risolve il problema del Bin Packing.

Il corriere, inoltre, aggiorna lo stato del collo, il quale deve essere rintracciato dal destinatario mediante il suo codice.

Realizzazione del Progetto

Si sono definiti diversi utenti all'interno del progetto di logistica , come l' Operatore che si occuperà dell'inserimento dei colli all'interno del centro di smistamento . L'operatore ha anche la possibilità di visualizzare tutti i veicoli aziendali disponibili al caricamento.

Un altro utente per cui è stata predisposta l'applicazione è il corriere che all'accesso potrà decidere il veicolo da cui è operativo in quel momento. Potrà iniziare la spedizione , consegnare i colli all'interno del veicolo e potrà decidere di terminare la spedizione. In base allo stato in cui verte il veicolo ad esempio se è impegnato in una spedizione non sarà possibile visualizzare il veicolo da parte degli operatori che si occupano del caricare i veicoli.

Il caricamento dei colli come descritto è stato gestito da un algoritmo di Bin Packing che descriverò più avanti.

Infine l'ultimo Utente sarà il destinatario o chi detiene l'id del collo per visualizzare tutte le informazioni del collo e il suo stato quindi se è stato spedito (inserito nel centro di smistamento), in consegna oppure consegnato.

2 Requisiti

La traccia in questione è stata implementata attraverso le seguenti linee:

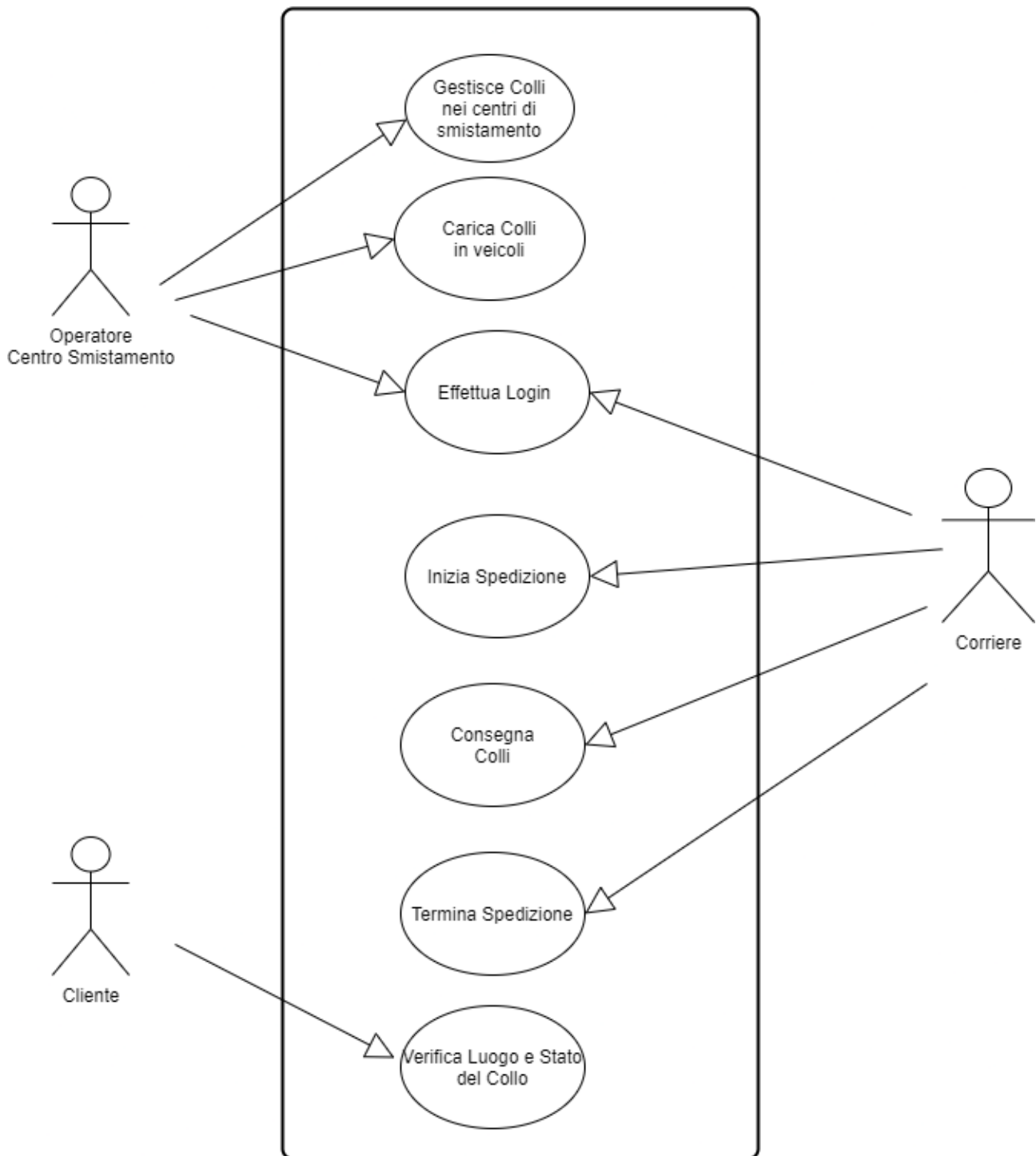
-Rispetto di alcuni dei principi SOLID che ricordiamo essere i seguenti:

- 1) Single Responsibility Principle(SRP)
- 2) Open-Closed Principle(OCP)
- 3) Liskov Substitution Principle(LSP)

- 4) Interface Segregation Principle(ISP)
- 5) Dependency Inversion Principle(DIP)

- Uso dell'interfaccia grafica
- Uso di un database per salvare tutti i dati persistenti
- Commenti in javadoc

3 Diagramma dei casi d'uso



Operatore:

Gestisce i colli : indichiamo che sarà colui che si occuperà di inserire ed eliminare i colli da un centro attraverso degli appositi comandi da interfaccia grafica.

Carica i colli: carica i colli nei veicoli potrà caricare i colli su tutti i veicoli dell'azienda che sono disponibili in quel momento

Effettua Login: accede semplicemente alla sessione di applicazione dedicata agli operatori

Corriere:

Indichiamo colui che guiderà il veicolo

Effettua Login: accede semplicemente alla sessione di applicazione dedicata ai corrieri

Inizia Spedizione: Inizierà la spedizione e quindi cambierà lo stato del veicolo in quel momento il quale non sarà disponibile più nella sessione degli operatori per essere caricato

Consegna Colli: potrà aggiornare lo stato di un collo appena consegnato tramite apposita interfaccia

Termina Spedizione: Terminando la spedizione attraverso apposito Button nell'interfaccia grafica l'operatore si renderà disponibile alla sessione degli operatori dei centri di smistamento

Cliente: Con Cliente indichiamo sia il mittente che il destinatario con il caso d'uso **Verifica luogo e stato del collo** entrambi potranno visualizzare i colli.

NB: In realtà l'applicazione è gestita in modo che chiunque sia in possesso dell'id spedizione (nel nostro caso id del collo) possa risalire allo stato della spedizione così come è gestito anche nella realtà.

4 Struttura del progetto

Prima di mostrare il diagramma UML volevo parlare di come è stato strutturato il progetto

Per rispettare il principio del Open-Closed ho deciso di strutturare il progetto nel seguente modo:

View – Service – Dao -Entity

Nelle view sono andato ad indicare tutto ciò che riguarda le interfacce grafica e quindi le classi che comprendono appunto la composizione dell'interfaccia che utilizzeranno gli utenti.

La creazione di quest'ultime è stata implementata in java swing.

I service sono dei servizi richiamati dalle view dove forniscono un qualsiasi tipo di servizio alla view . Avremo una classe service per ogni caso d'uso . Ad esempio per la gestione dei colli nel centro di smistamento il service fornirà una serie di metodi alla view.

I service usano oggetti di tipo dao.

Il Dao (*Data Access Object*) è un pattern architetturale per la gestione della persistenza , si tratta di un'interfaccia con dei metodi , chiamate crud operation .

come ad esempio: save , get , getall , delete.

Che rappresenta un 'entità tabellare di un RDBMS usato per stratificare e isolare l'accesso ad una tabella tramite query che sono poste all'interno dei metodi della classe creando un maggior livello di astrazione .

Avremo un daoImplement che implementerà tale interfaccia per ogni tipo di Entity.

Le entity vengono utilizzate per stabilire una mappatura tra un oggetto e una tabella nel database. Le entity sono note anche come oggetti di dominio. le entità verranno utilizzate in situazioni in cui è presente una logica di business e come tali contengono informazioni sul sistema (o parte del sistema) che sta modellando.

Quindi perché usare questo modello ? Ho deciso di utilizzare questo modello perché mi interessava capire come strutturare questo progetto e renderlo aperto all'estensione grazie ai service e il pattern dao potremmo cambiare tranquillamente database o implementazione di un daoImplement dato che un service accetta nel suo costruttore un oggetto di tipo Dao oppure un'estensione di quest'ultimo ma non direttamente il daoImplement. E quindi nel nostro caso sarà la view a specificare nel suo costruttore quale DaoImplement dare al service e continuare ad usare quel service anche quando sarà cambiato l'daoimplement che gli fornirà la view.

In questa struttura avremo però una dipendenza tra le view e l'daoimplement dato che la view dovrà usare l'daoimplement da dare al service.

Avrei potuto risolvere inserendo un nuovo strato come il controller il quale sarebbe stato lui a usare i dao e fornire questi a service e poi lui essere usato dalla view, per motivi di tempo però non sono riuscito a rappresentare questa soluzione.

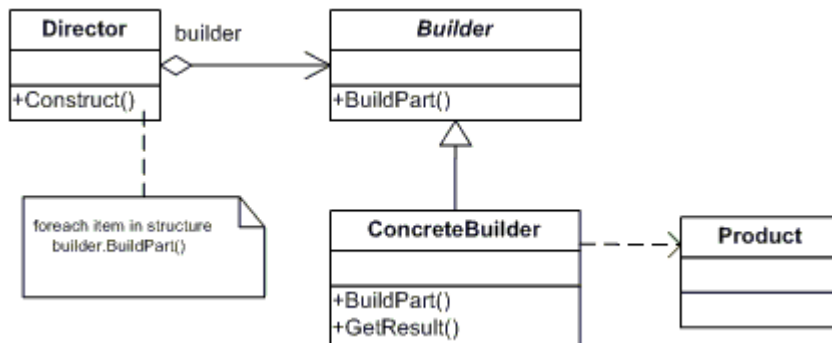
5 Pattern Utilizzati

Oltre al pattern strutturale Dao ho utilizzato.
Builder Pattern
Pattern State.

Ho usato anche il pattern Singleton in svariati contesti come la connessione al Database e l'utilizzo in un Context ossia una classe che forniva informazioni sull'operatore o sul corriere loggati. Rendendo disponibile quell'istanza del context ovunque sarebbe stato necessario

Builder Pattern

Lo Scopo del Builder Pattern è quello di separare la costruzione di un oggetto complesso dalla sua rappresentazione in modo tale che lo stesso processo di costruzione può creare differenti rappresentazioni



Come possiamo vedere dal seguente diagramma abbiamo una classe Director dove al secondo del builder creiamo una tipologia di prodotti.

Io l'ho usato in maniera un po' differente (vista un'implementazione nel libro effettive java) avendo un oggetto complesso come il collo che aveva tanti campi ho deciso di creare una classe builder il quale crea l'oggetto complesso nel suo metodo build e può e ritorna l'istanza dell'oggetto creato in base solo ai valori che abbiamo deciso di settare, in questo modo possiamo evitare Costruttori multipli o campi null appunto nel costruttore dell'oggetto. Sarebbe stato utile rappresentare questa soluzione per tutti gli oggetti del mio sistema ma avrebbe avuto un costo di tempo sull'implementazione

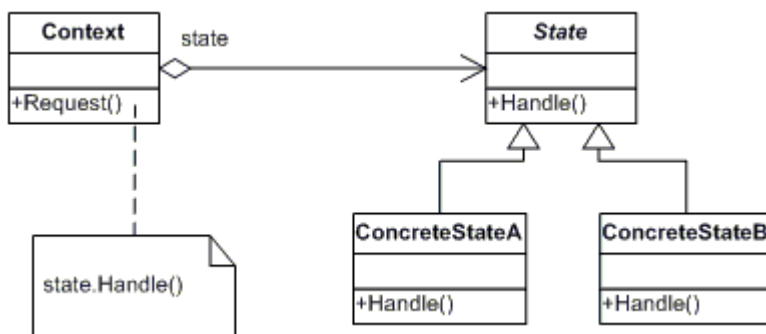
<https://github.com/GiuseppeNappo/LogisticApp/blob/main/src/Entity/Pack/PackBuilder.java>

Link Codice utilizzo codice Pattern builder :

<https://github.com/GiuseppeNappo/LogisticApp/blob/main/src/View/OperatorView/InsertPackFrame.java>

State

Permette ad uno oggetto di cambiare il proprio comportamento quando cambia il suo stato interno. L'oggetto sembrerà cambiare la sua classe.



Ho usato questo pattern andando a rappresentare diversi concretestate come stati del veicolo ognuno con 3 metodi start , read e stop . avendo 3 state come running , ready e onloack questi tre stati indicavano se il veicolo è in movimento e se è disponibile per essere caricato e quindi richiamato dai centri di smistamento e ad ogni cambiamento di stato del veicolo possiamo vedere che cambia anche lo stato dei colli che si trovano su di esso.

Link Codice progetto implementazione Pattern state :

<https://github.com/GiuseppeNappo/LogisticApp/tree/main/src/State>

Link Codice utilizzo codice Pattern state :

<https://github.com/GiuseppeNappo/LogisticApp/blob/main/src/View/CourierView/CourierView.java>

DaoPattern

Il pattern DAO è usato per separare la logica di business dalla logica di accesso ai dati. Infatti, i componenti della logica di business non dovrebbero mai accedere direttamente al database: questo comporterebbe scarsa manutenibilità. Solo gli oggetti previsti dal pattern Dao possono accedervi. Inoltre, se dovessimo modificare il tipo di memoria persistente

utilizzata, o anche passare da Oracle a MySql per esempio, non sarà necessario stravolgere il codice della nostra applicazione, ma basterà modificare i DAO utilizzati.

I concetti principali sono:

- una classe (model) per ogni tabella
- una interfaccia (detta Dao) per ogni tabella contenente tutti i metodi Crud relativi a quella tabella.
- una implementazione per ogni interfaccia Dao

Chiunque voglia poter accedere al DataBase dovrà usare la relativa interfaccia Dao.

Link Codice progetto implementazione PatternDAO:

<https://github.com/GiuseppeNappo/LogisticApp/tree/main/src/Entity/Pack>

Link Codice utilizzo codice PatternDAO:

<https://github.com/GiuseppeNappo/LogisticApp/blob/main/src/Service/ManagePackInCenterService.java>

6 UML

Poiché avendo una grande quantità di classi ho deciso di mostrare i diagrammi delle 4 funzionalità principali in ordine.

1 . Gestione dei colli da parte dell'operatore

In questo diagramma uml rappresentiamo la gestione da parte dell'operatore dei colli nel centro di smistamento come possiamo vedere questo uml fa uso di un servizio chiamato ManagePackInCenterService i quali i suoi metodi forniscono una serie di funzionalità per l'inserimento e l'eliminazione dei colli dal centro di smistamento. Il service fa uso di vari PackDAO che gli

vengono forniti dalla view la quale specifica i daoImplement che il service utilizzerà , il service potrà usare tranquillamente i daoImplement forniti dalla view perché lui conosce le sue interfacce.

2 . Caricamento dei colli sul veicolo

In questo diagramma uml rappresentiamo il caricamento dei colli sui veicoli anche qui vengono usati dei servizi per la manipolazione dei colli e dei veicoli oltre però ai soliti servizi , per il caricamento abbiamo anche il servizio del next-fit che ha la funzionalità di fornire l’algoritmo per la risoluzione del bin Packing . In particolare la funzione di caricamento del veicolo non consiste solo di richiamare l’algoritmo per il caricamento dei colli ma anche quella di recuperare i veicoli “necessari” e con tale termine intendiamo i veicoli disponibili al caricamento

3 . Le Operazioni di consegna/inizio spedizione e termine spedizione del corriere

In questo diagramma uml sono andato a rappresentare tutte le operazione che il corriere potrà azionare dalla sua interfaccia grafica , Come iniziare la spedizione che consiste nel modificare lo stato del veicolo in quel momento attraverso il pattern state e di aggiornare questa informazione sul attraverso i metodi nel manageShippingService e del VehicleOperatioInService che fanno uso dei loro DAO per aggiornare questi dati nel Database , Anche la funzionalità di consegna attraverso il ManagePackInVehicleService e Il ManageShippingService forniscono le funzionalità per cambiare lo stato dei colli (Nel Database)e di eliminare i colli dal veicolo una volta consegnato , Per quanto riguarda la terminazione della spedizione questa sempre i soliti servizi cambierà i valori dei veicoli.

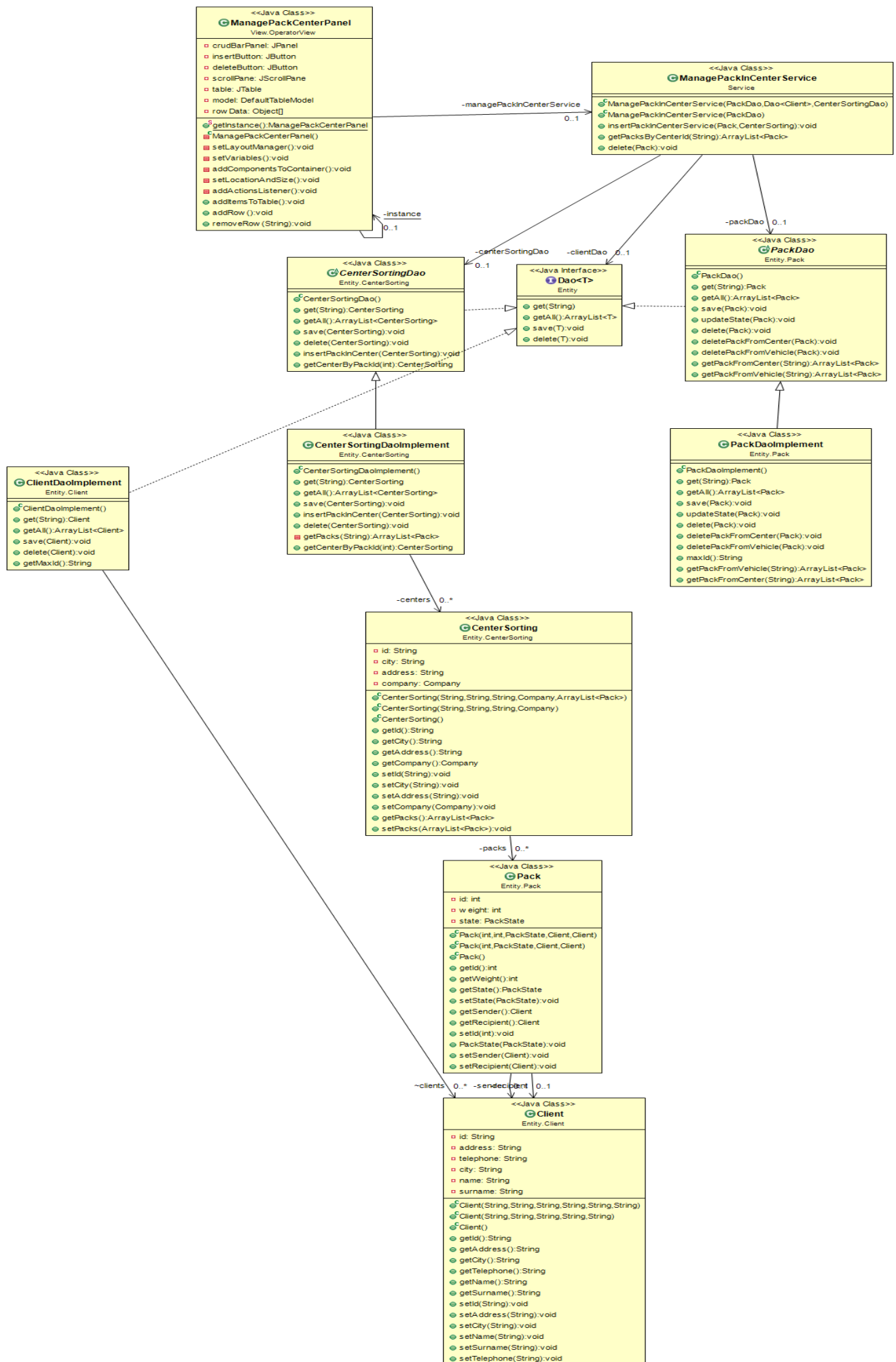
4 . La visualizzazione delle informazioni del collo al cliente

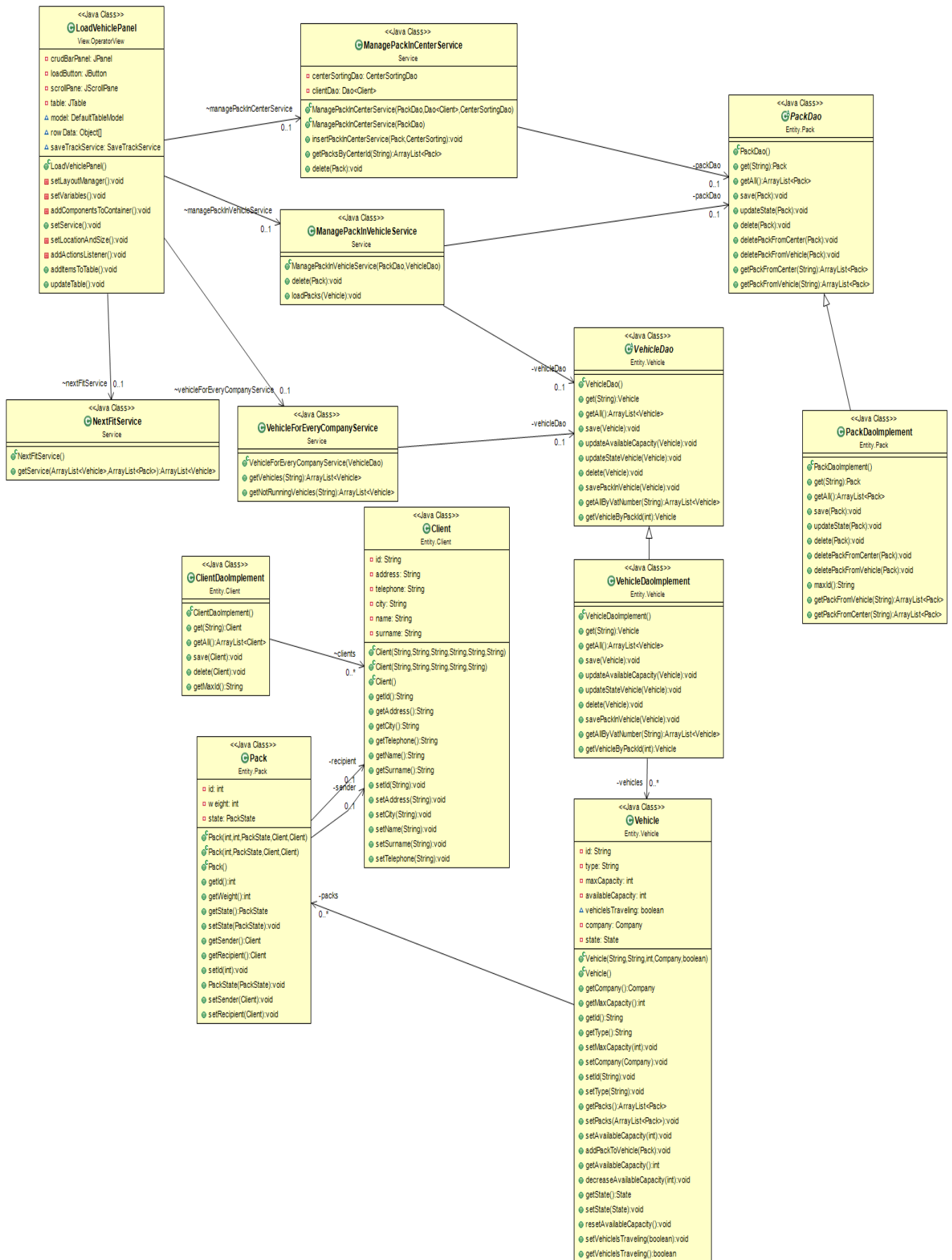
In questo diagramma UML andiamo semplicemente a rappresentare la prelazione delle informazioni riguardanti una traccia che è l’insieme dei dati relativi ad un centro di smistamento , un veicolo e un pacco quindi utile per recuperare le informazioni di stato dei veicoli e i colli e l’ultimo luogo in cui è stato visto il collo

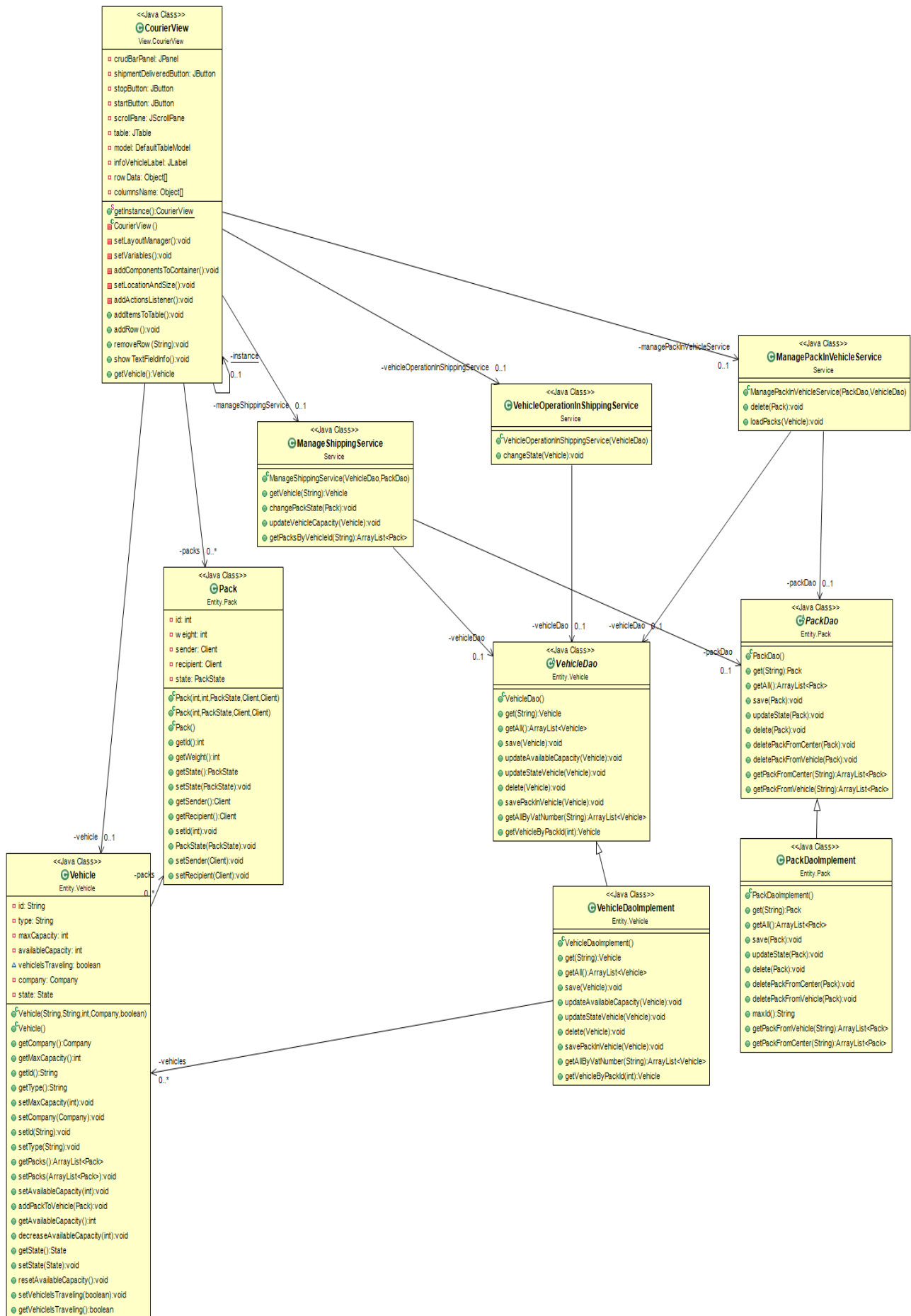
5. DAO Pattern

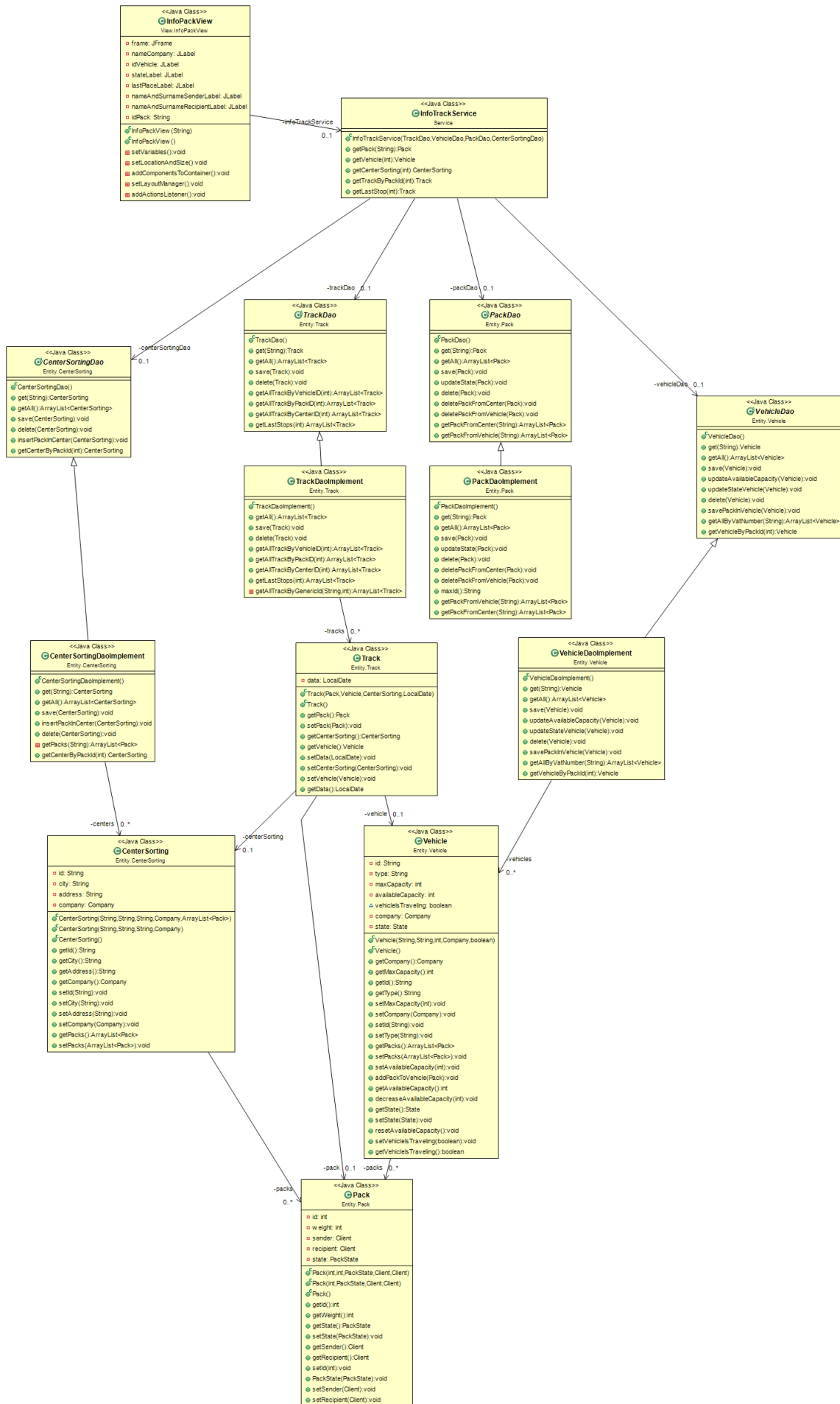
6. STATE Pattern

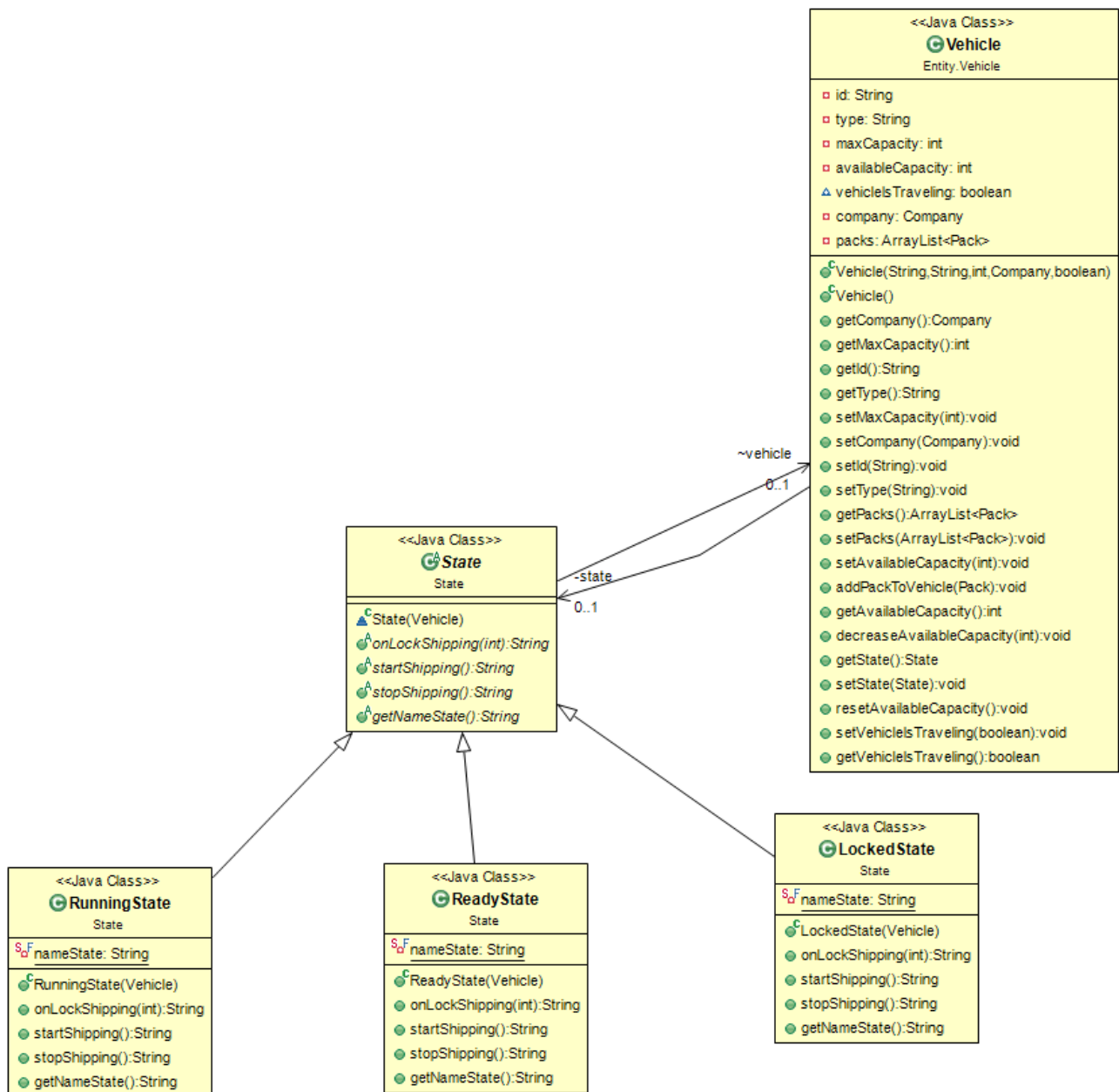
7. Builder Pattern

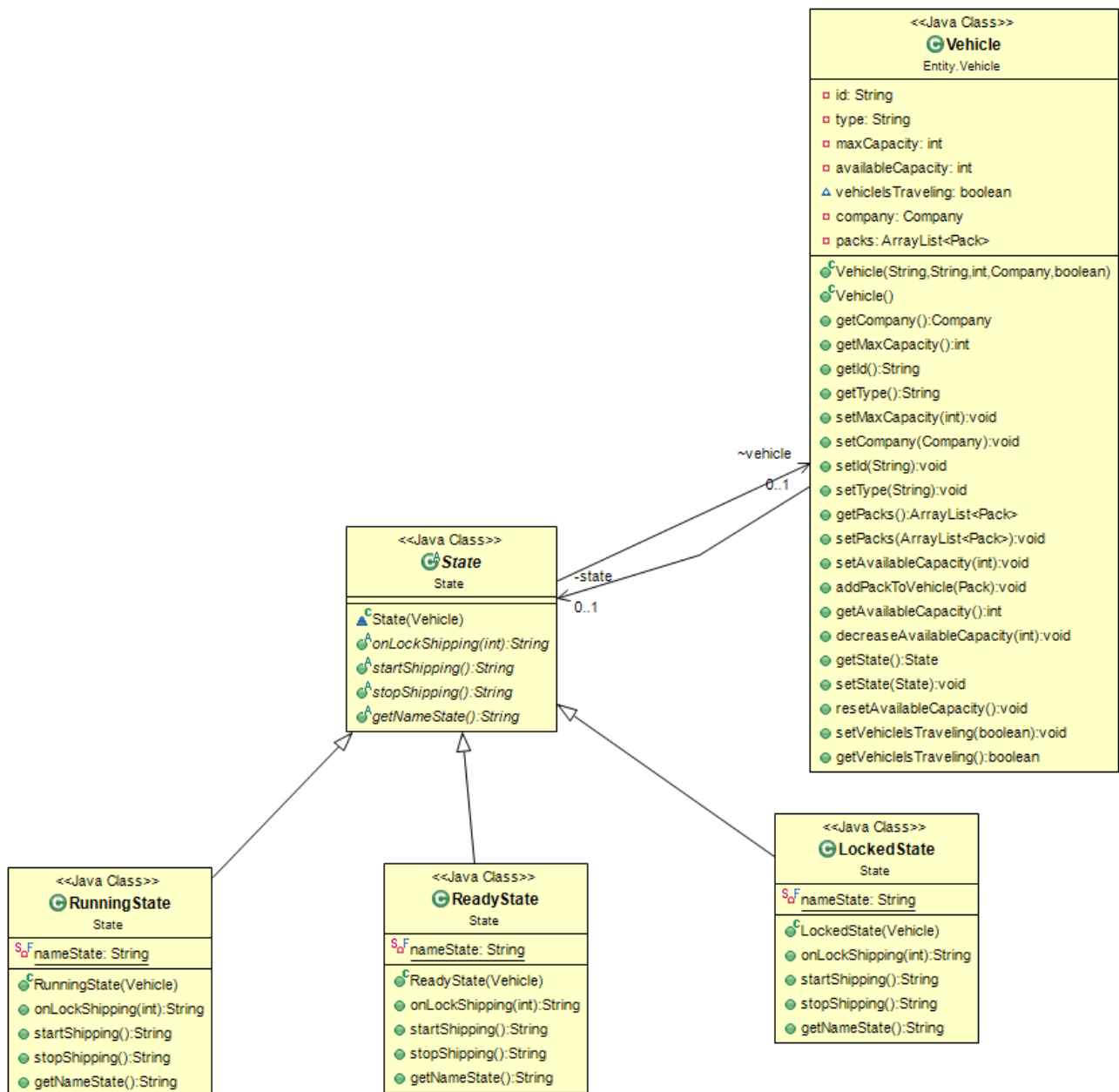


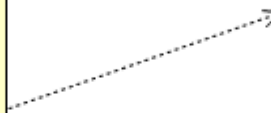
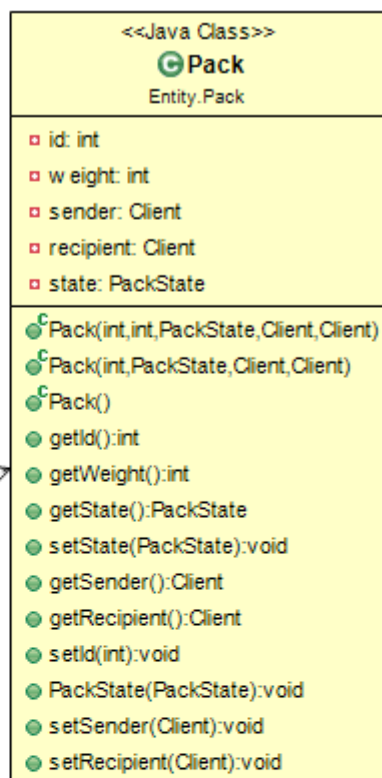
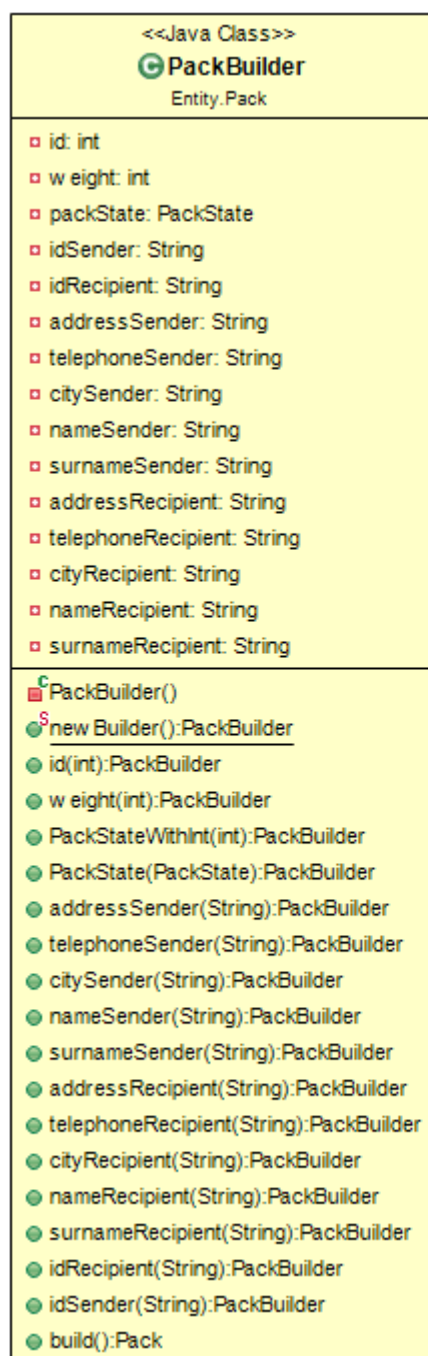












7 Problema del Bin Packing

Nel problema dell'imballaggio dei contenitori, gli articoli di diversi volumi devono essere imballati in un numero finito di contenitori ciascuno di un determinato volume in modo da ridurre al minimo il numero di contenitori utilizzati. Nella teoria della complessità computazionale, è un problema NP-difficile combinatorio. Quando il numero di contenitori è limitato a 1 e ogni articolo è caratterizzato sia da un volume che da un valore, il problema di massimizzare il valore degli articoli che possono entrare nel contenitore è noto come problema dello zaino.

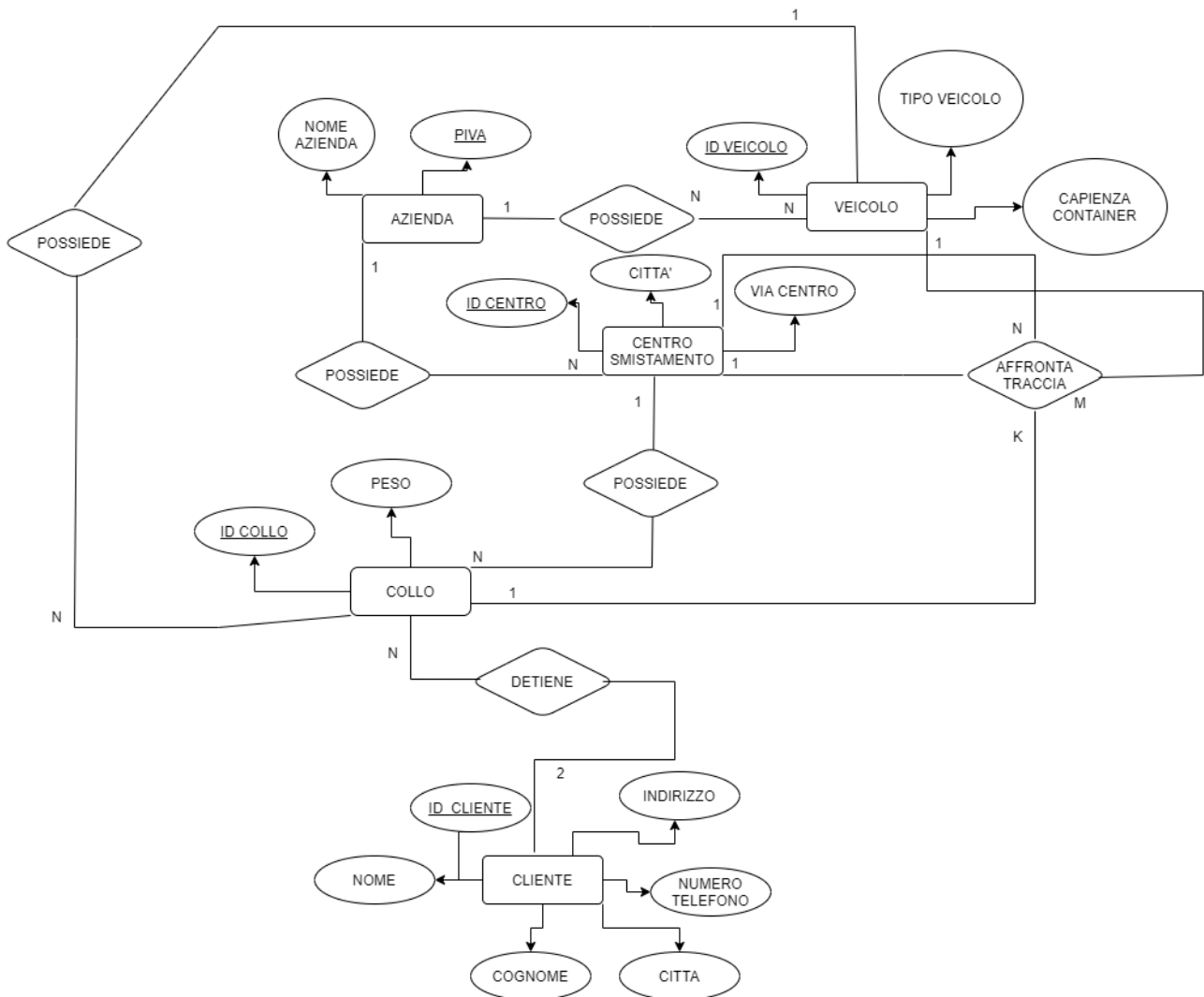
8 Algoritmo Next fit

L'algoritmo di next fit consiste nell'inserire i colli che arrivano in qualsiasi ordine in un container se l'i-esimo collo è più grande del j-esimo container passiamo al prossimo altrimenti se l'i-esimo collo entra nel j-esimo container anche l'i-esimo collo più proverà ad inserirlo nel j-esimo container proprio perché questo algoritmo ogni volta che deve inserire un collo non parte dal primo container ma dall'ultimo container allocato

Link codice algoritmo next fit :

<https://github.com/GiuseppeNappo/LogisticApp/blob/main/src/NextFit/NextFit.java>

9 Database Relazionale



Questo è il modello relazione su cui si è basato il database , come già spiegato prima , per ogni tabella del database si sono andate a creare delle entità i DAO rappresentano essenzialmente il modo di accesso ai dati di questa entità sul database chiunque vogli accedere ai dati relativi un'entità porta accedere solamente attraverso l'interfaccia del DAO o la classe che estende il DAO nel caso in cui alcune entità abbiano bisogno di informazioni più specifiche.

Per rappresentare la rappresentazione di questo database ho utilizzato SQLite

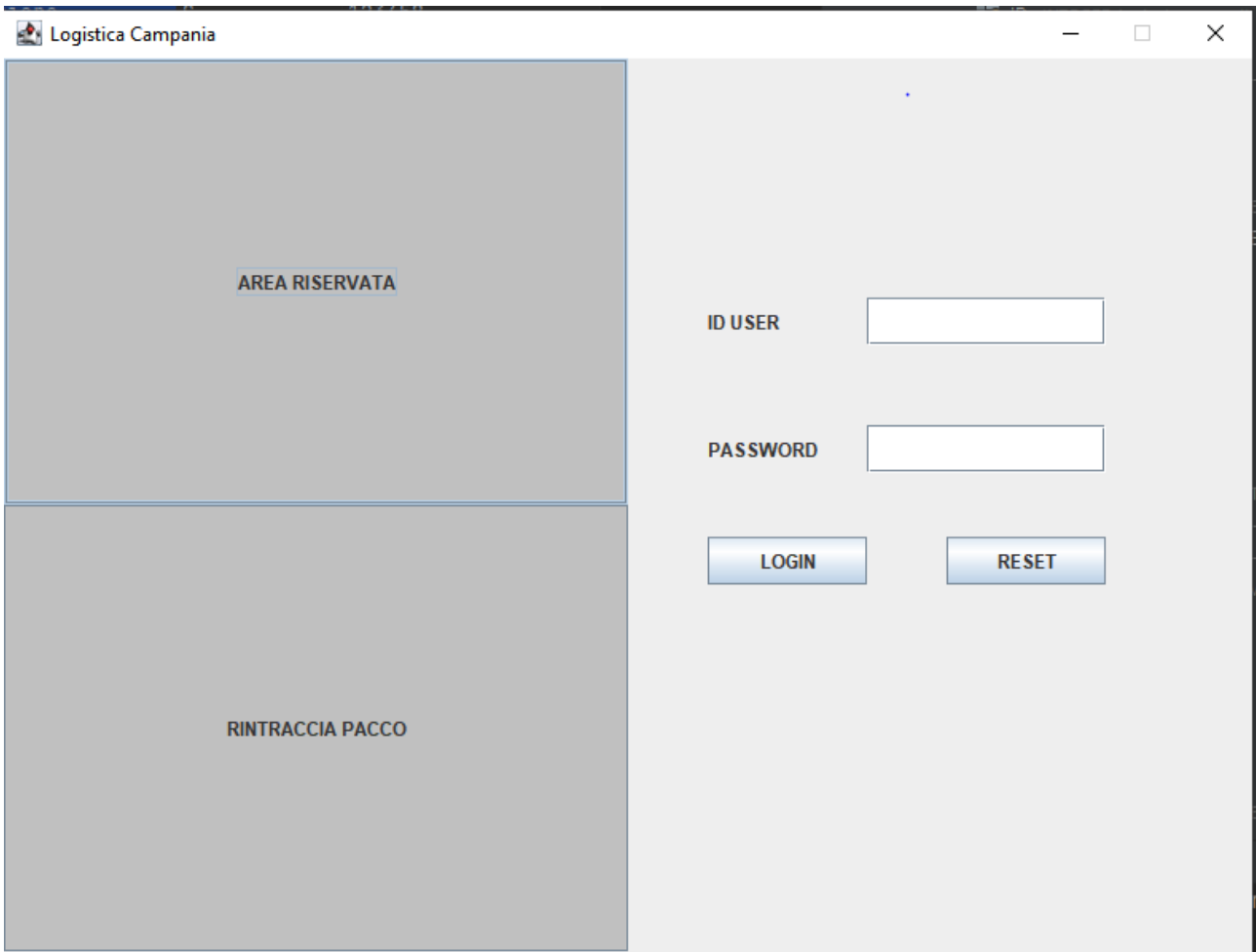
10 Codice Progetto GitHub

Qui il link del codice GitHub del progetto :

<https://github.com/GiuseppeNappo/LogisticApp/tree/main/src>

11 Utilizzo del programma

Questa è la schermata principale nell'area riservata possiamo loggarci come operatori o corrieri
Il software capirà automaticamente chi siamo a seconda delle nostre credenziali



Logistica Campania

AREA RISERVATA

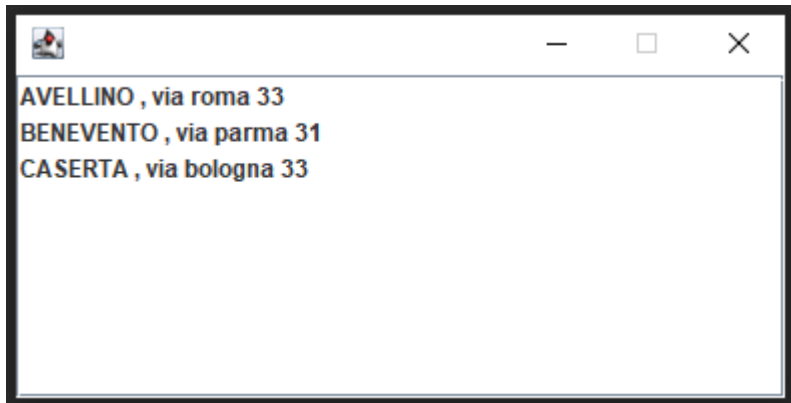
RINTRACCIA PACCO

ID USER

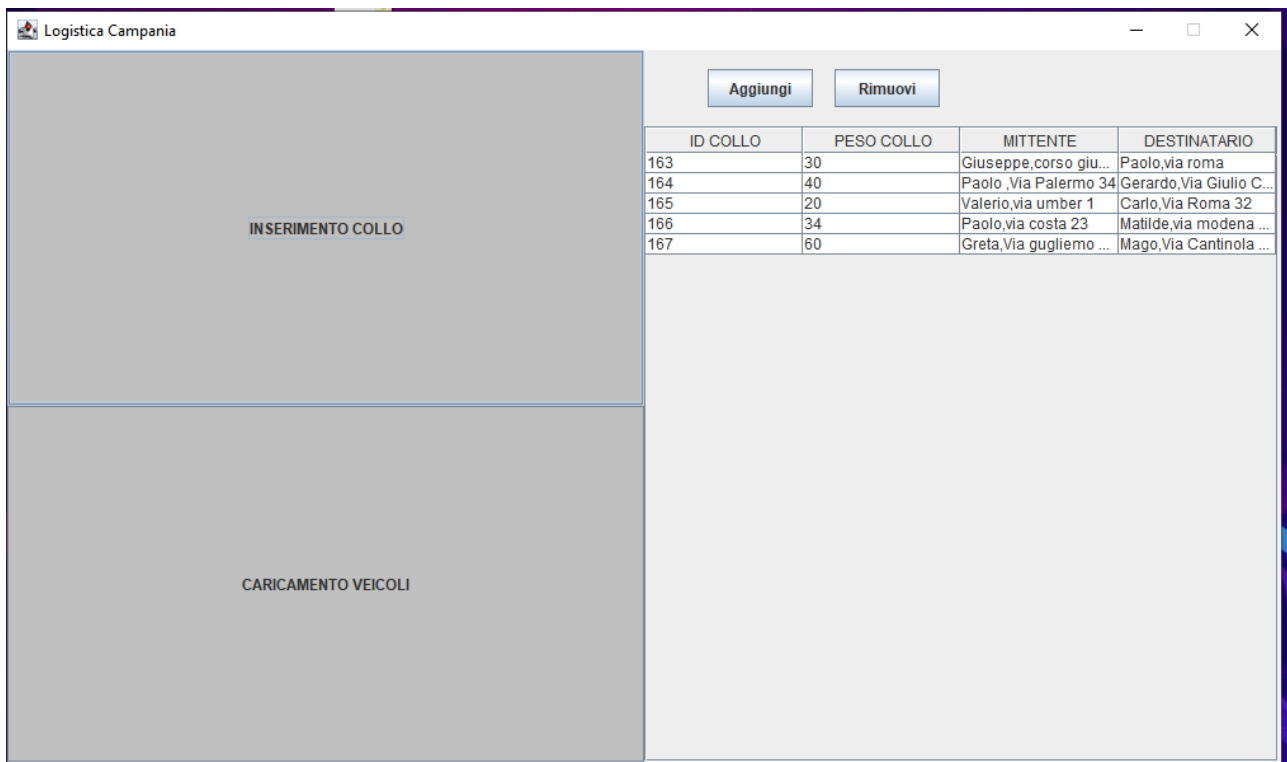
PASSWORD

LOGIN RESET

Se decidessimo di loggare come operatore di un centro smistamento ci comparirà la scelta di tutti i centri di smistamento a cui possiamo accedere . Accediamo alla schermata successiva scegliendo uno dei centri sottostanti.



Questa è la schermata dopo aver scelto un centro di smistamento , ogni centro di smistamento vede i colli che ha già al suo interno e può aggiungerne o rimuoverne.



In questa schermata possiamo vedere tutti i veicoli del centro dell'azienda e non dello specifico centro poiché i veicoli sono disponibili ad essere caricati da tutti i centri di smistamento aziendale finché il corriere che guida il veicolo decide di iniziare la spedizione e quindi non essere più disponibile alla visione dai centri di smistamento

Logistica Campania

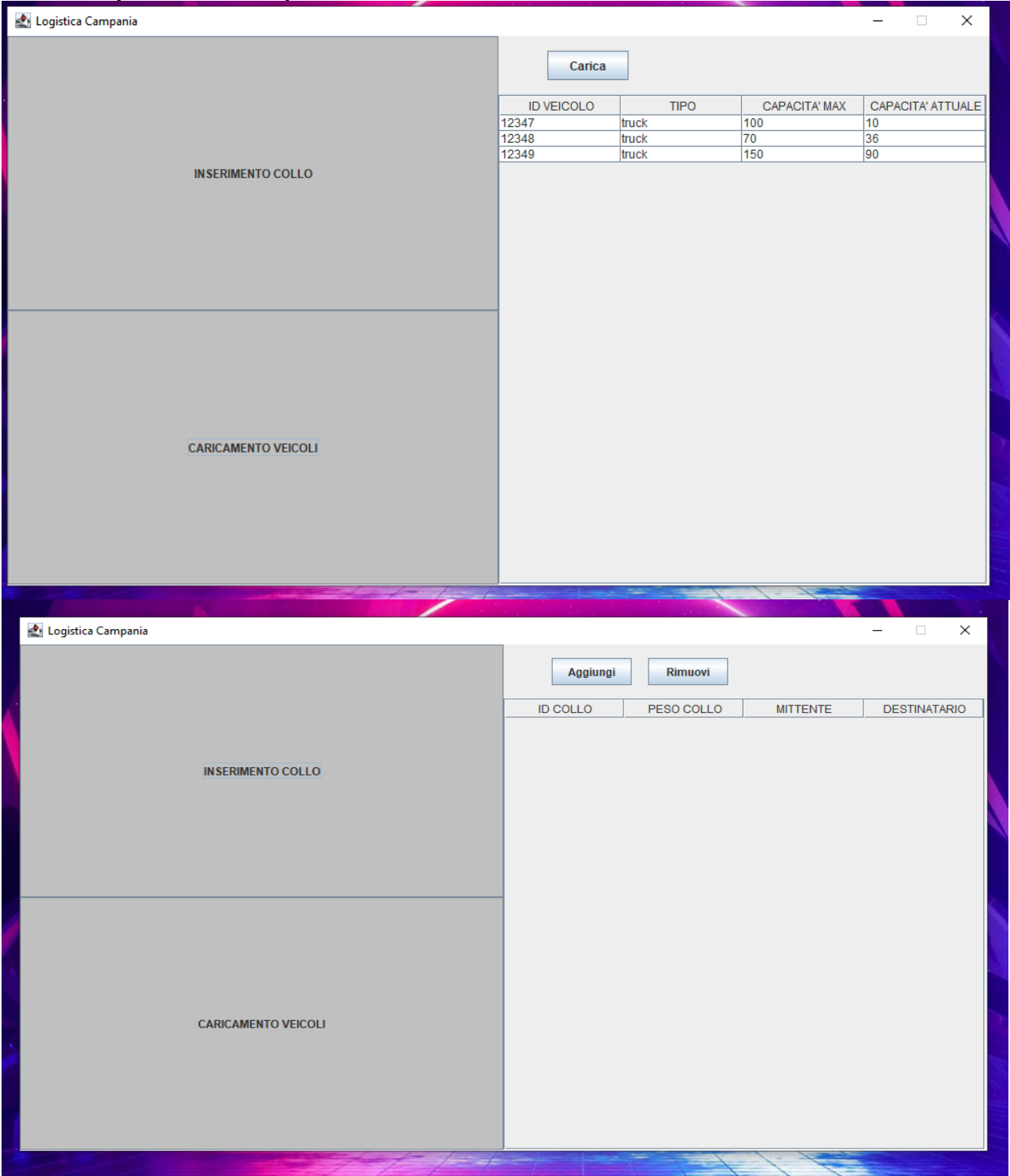
INSERIMENTO COLLO

CARICAMENTO VEICOLI

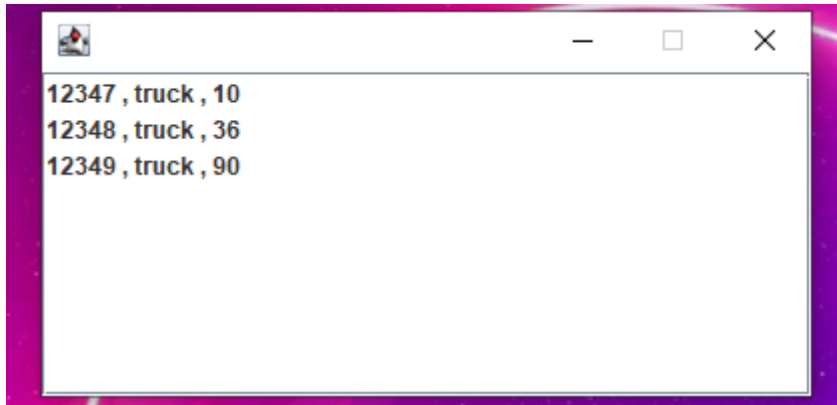
Carica

ID VEICOLO	TIPO	CAPACITA' MAX	CAPACITA' ATTUALE
12347	truck	100	100
12348	truck	70	70
12349	truck	150	150

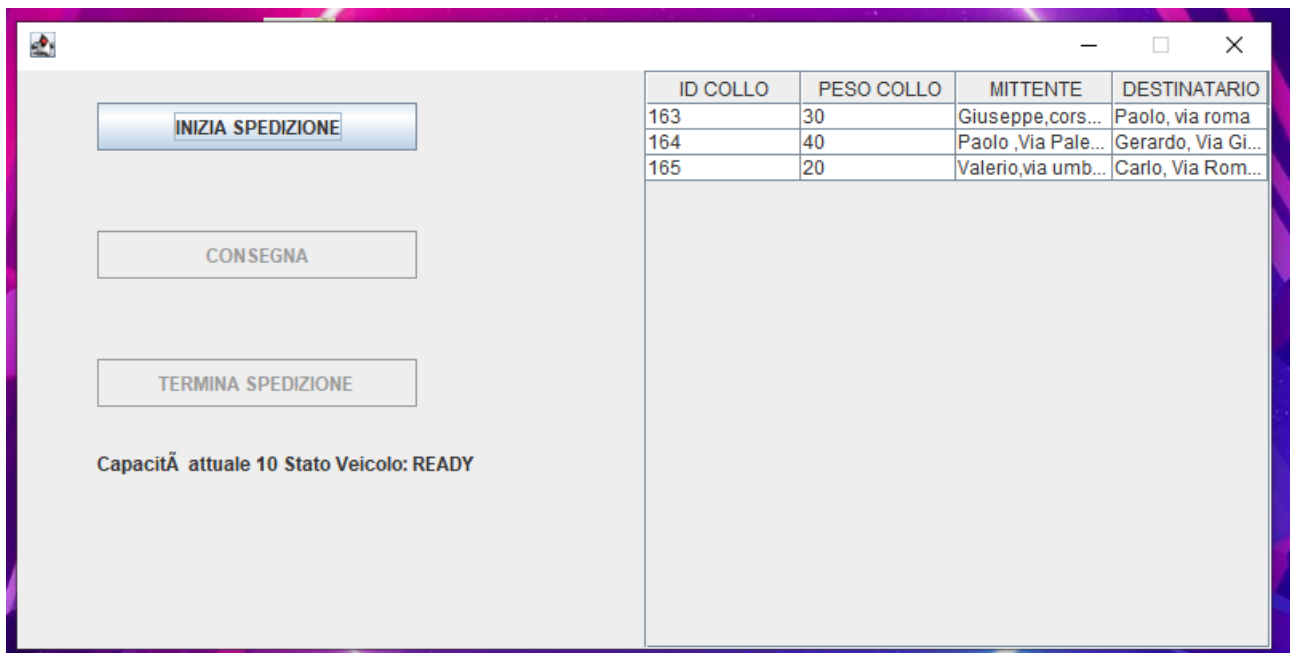
Dopo aver premuto il pulsante carica possiamo notare da queste due immagini come la capacità attuale dei veicoli sia diminuita in base ai colli che erano nel centro di smistamento precedentemente , con il pulsante carica appunto carichiamo attraverso l’algoritmo next-fit tutti i colli a disposizione su tutti i veicoli disponibili. Come vediamo nella seconda schermata che corrisponde a quella dell’inserimento i colli non sono più nel centro perché sono stati caricati.



Se al momento del login invece decidessimo di loggare come un corriere dovremmo scegliere tra i veicoli disponibili in quella azienda. In questa tipologia viene mostrato l'id del veicolo a cui si vuole accedere il tipo del veicolo e la capacità attuale di quest'ultimo

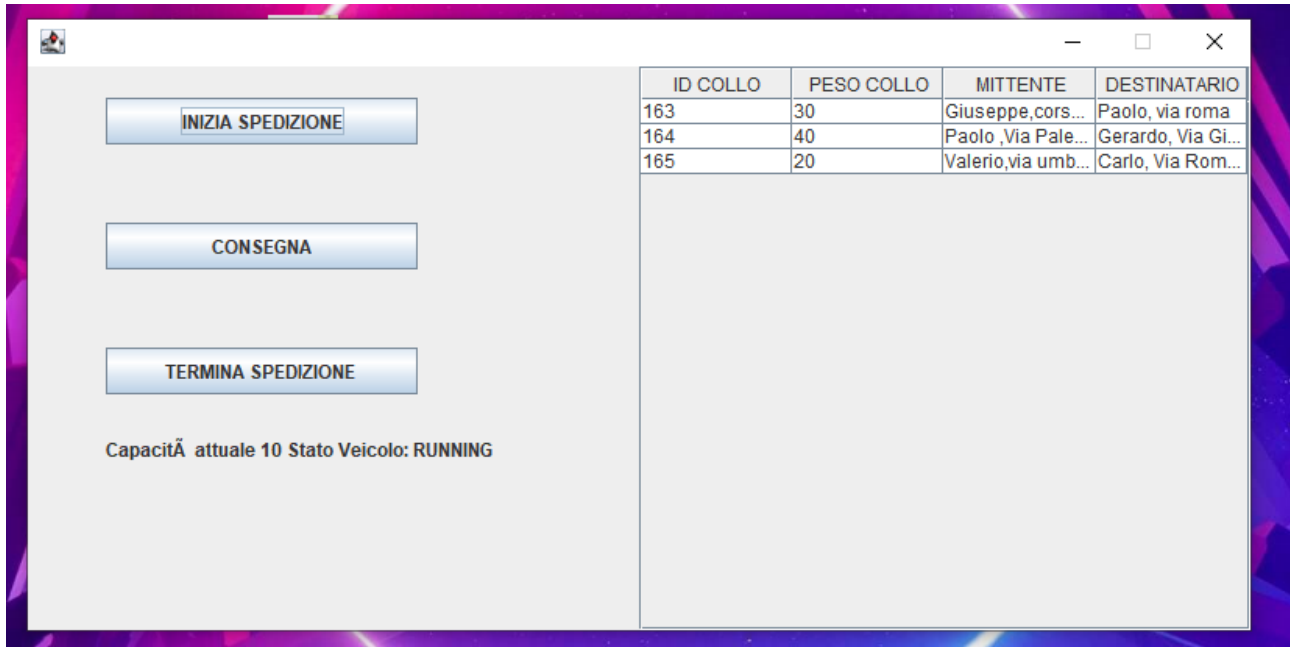


Questa è la schermata del veicolo una volta scelto , possiamo vedere i colli all'interno del veicolo e abbiamo la possibilità di iniziare la spedizione se ancora non iniziata.

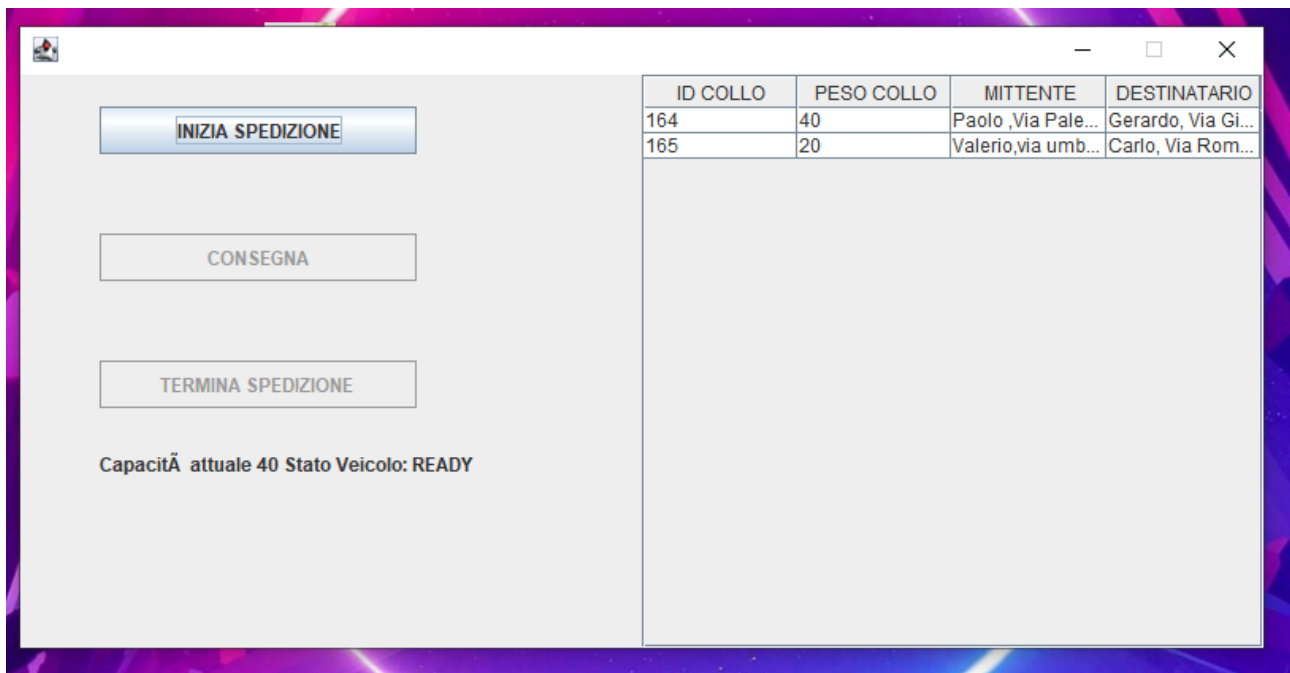


Nel caso in cui dovessimo scegliere di iniziare la spedizione come in questo caso vediamo che lo stato del veicolo cambia da ready a running in modo da non essere più disponibili ai centri di smistamento per il caricamento.

Ad ogni cambio dello stato del veicolo anche i colli subiranno un cambiamento di stato



Qui mostriamo i colli all'interno del veicolo dopo aver consegnato il collo con id 164



Qui mostriamo che un veicolo in stato di running non è visualizzabile ai centri smistamento

Logistica Campania

INSERIMENTO COLLO

CARICAMENTO VEICOLI

Carica

ID VEICOLO	TIPO	CAPACITA' MAX	CAPACITA' ATTUALE
12348	truck	70	36
12349	truck	150	90

Questa è la schermata di rintracciamento del collo dove ci basterà inserire l'id del collo per conoscere tutte le informazioni

Logistica Campania

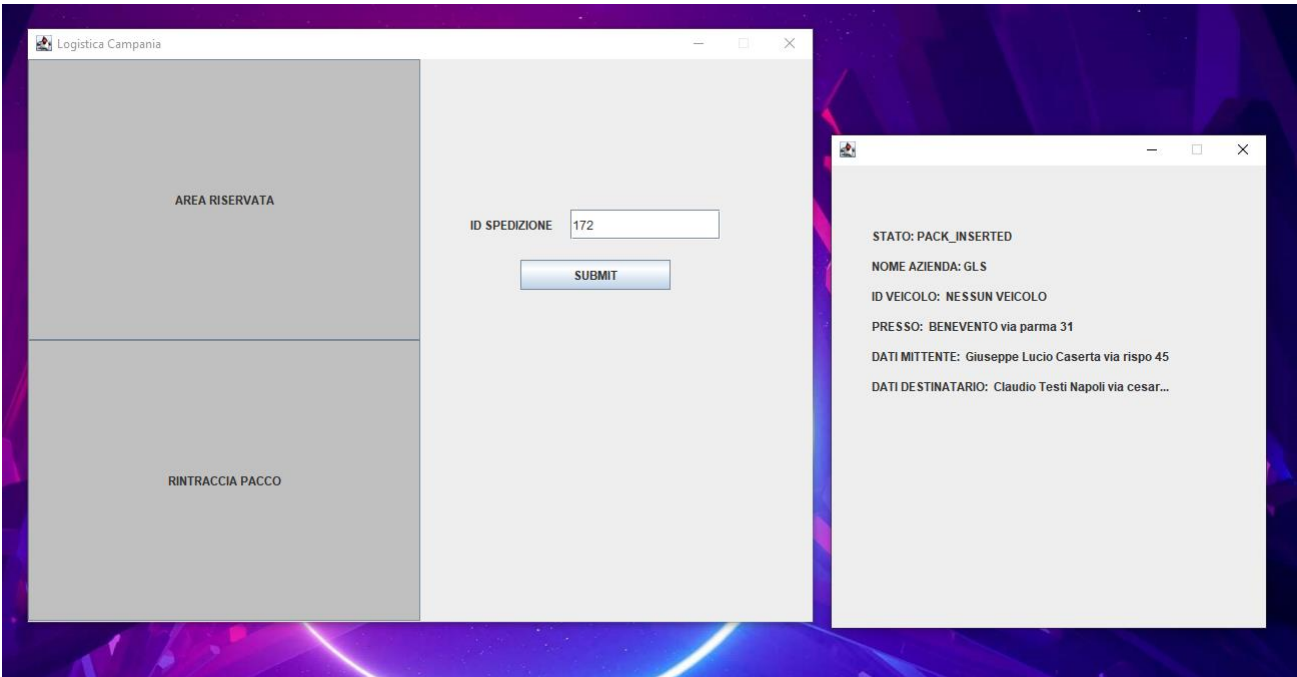
AREA RISERVATA

RINTRACCIA PACCO

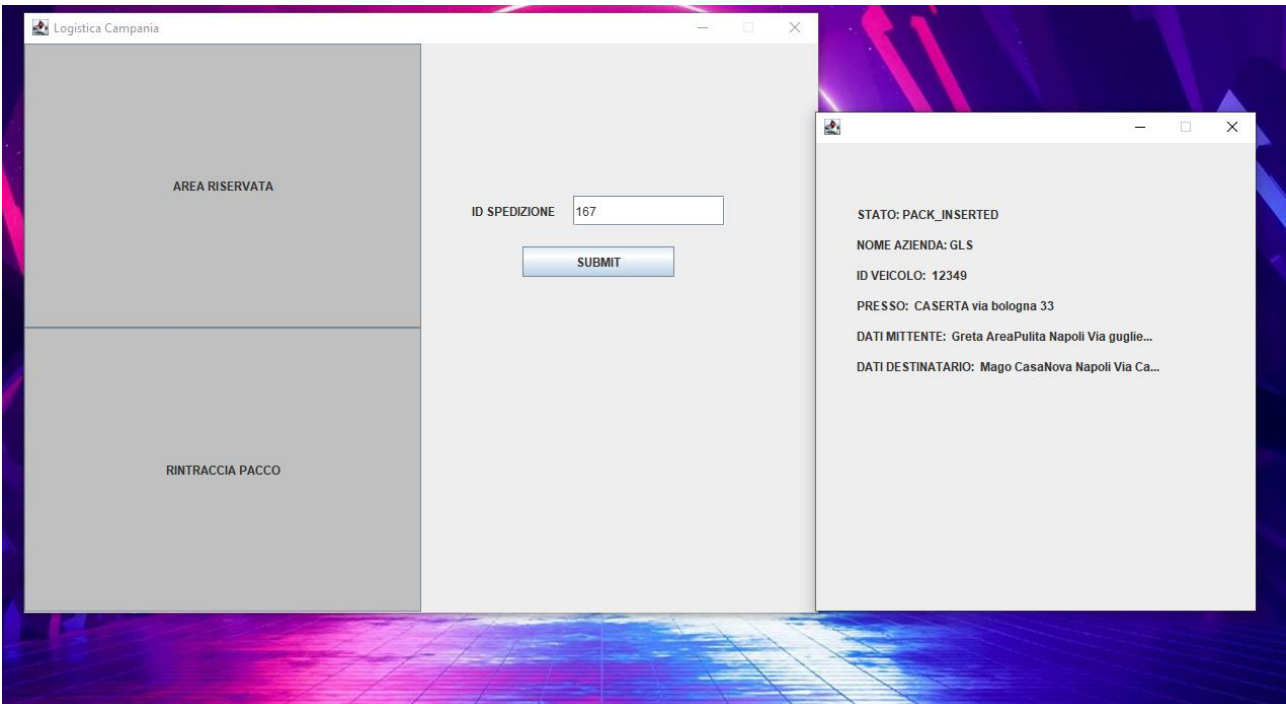
ID SPEDIZIONE

SUBMIT

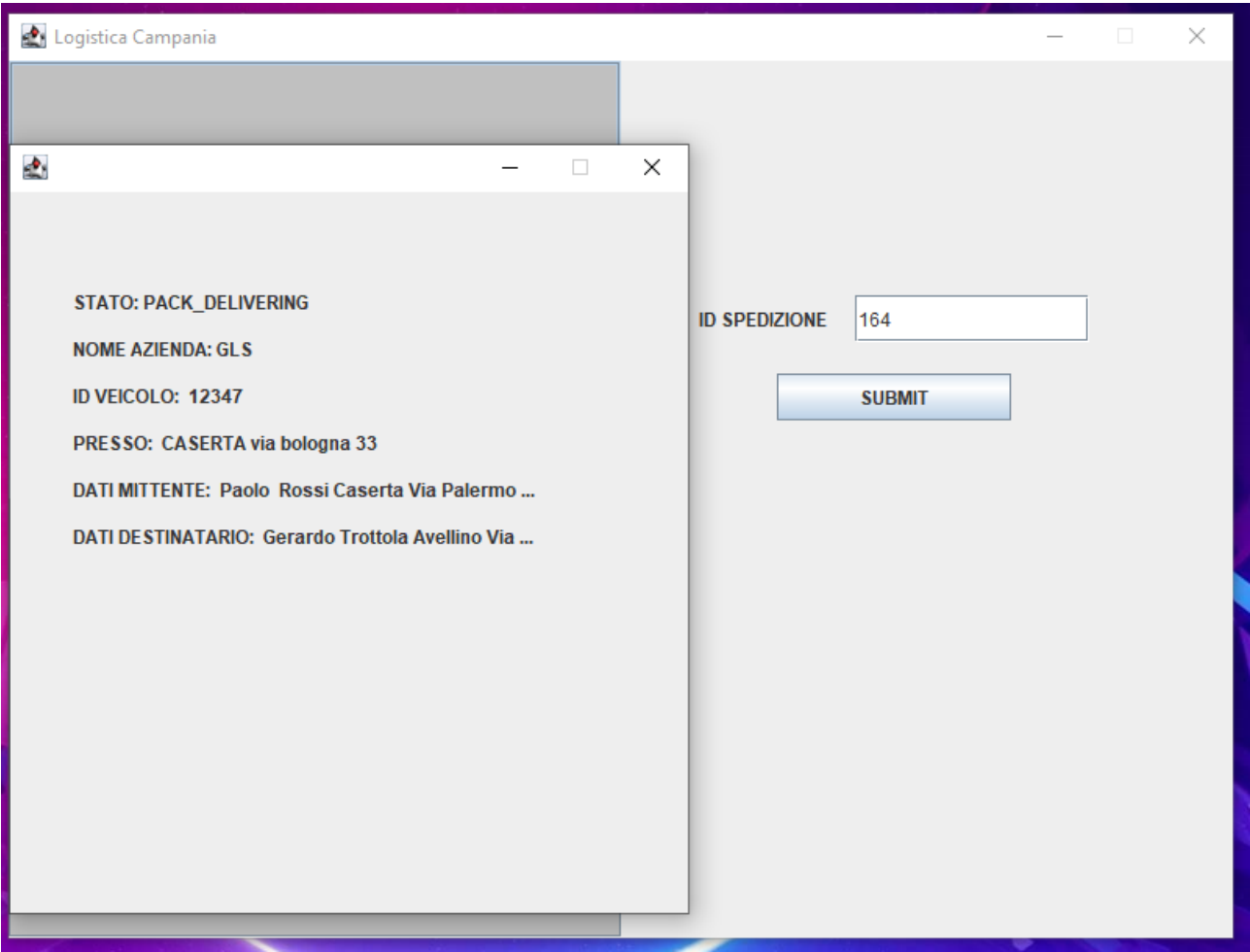
Questo è lo stato di un collo che è stato inserito ma il quale non è stato ancora caricato in nessun veicolo



Questo è lo stato di un collo che è stato inserito e caricato ma il veicolo in cui è posto è fermo



Questo è lo stato di un collo in consegna



Questo è lo stato di un collo consegnato con successo

