PROGETTAZIONE

CARLOAN

Versione 1.0

Data di rilascio:

INGEGNERIA DEL SOFTWARE A.A. 2014-2015

Realizzato da

Sinisi Francesco 618189 Informatica fra.a7x@gmail.com

Palumbo Vito 624778 Informatica ninenexusx@gmail.com

INDICE

1. ARCHITETTURA

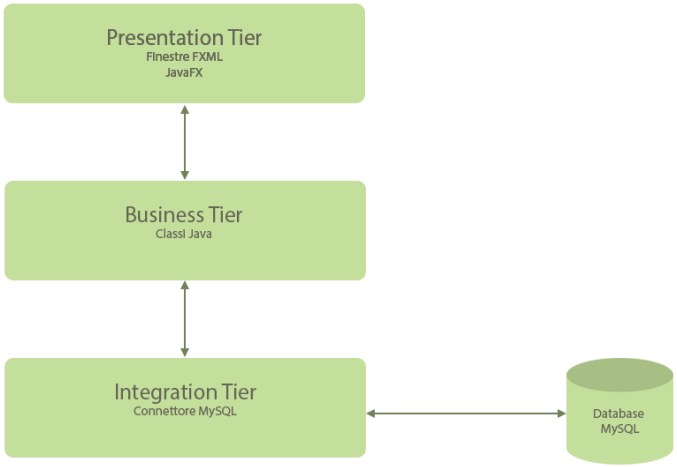
1.1 Livelli Architetturali

Il sistema CarLoan è realizzato sfruttando un'architettura a tre livelli:

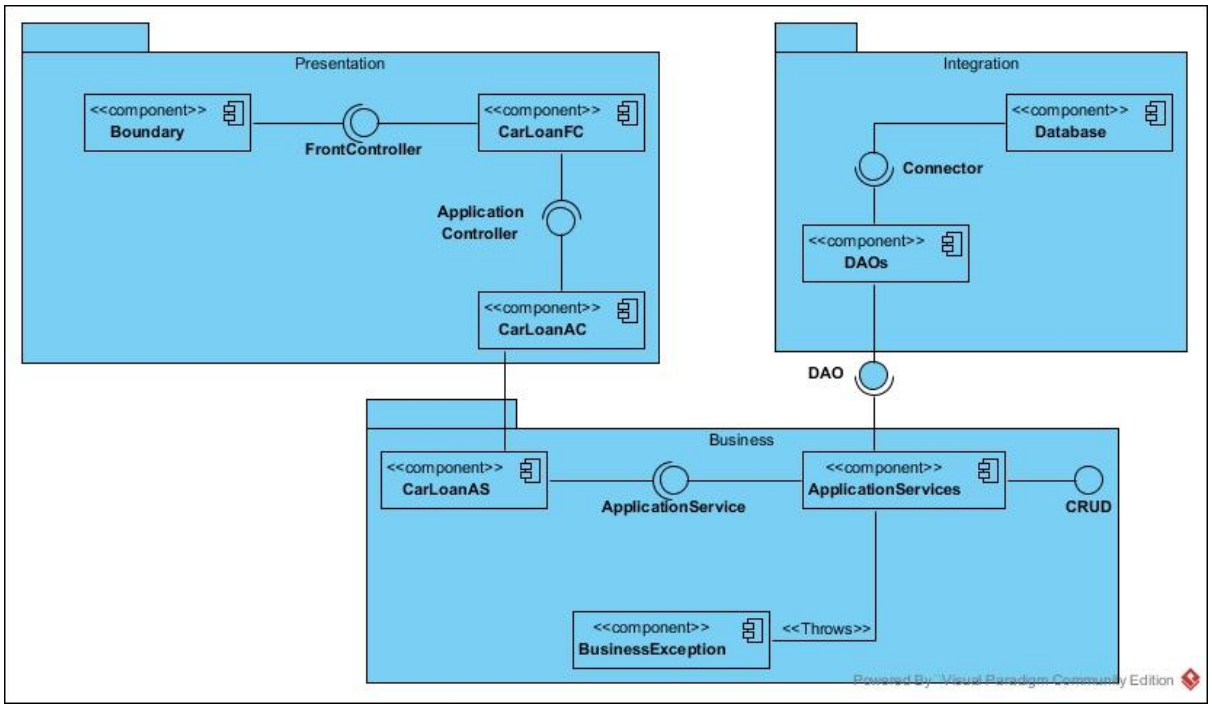
- **Livello di presentazione** in cui sono racchiuse le componenti che si occupano della raccolta delle richieste da parte dell'utente che interagisce con le diverse view del sistema. Gestisce inoltre i dati relativi alla sessione d'esecuzione.

- **Livello di business** che contiene componenti che implementano la logica di business specificata nel documento di analisi.

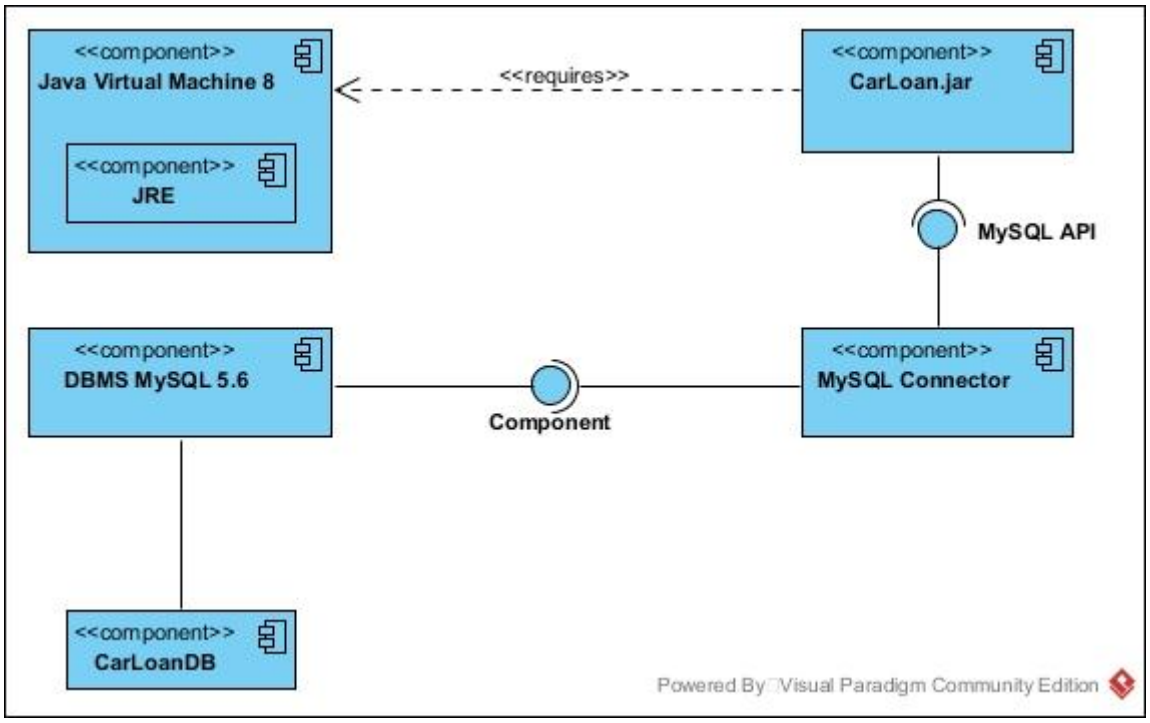
- **Livello di integrazione** le cui componenti si occupano dell'accesso alla sorgente di dati persistenti (database) e forniscono operazioni di base quali creazione, modifica, lettura e cancellazione.



1.2 Diagramma delle Componenti



1.3 Diagramma di Configurazione

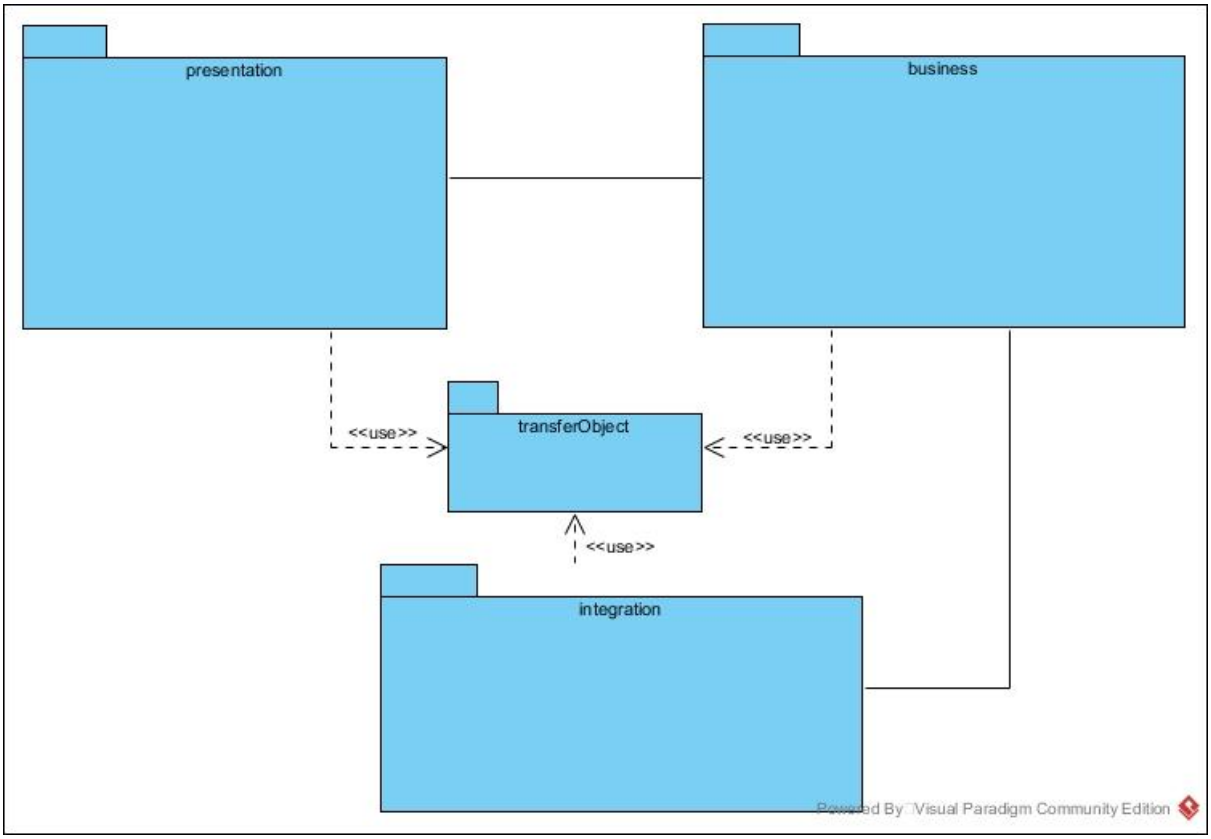


2. PROGETTO DI DETTAGLIO

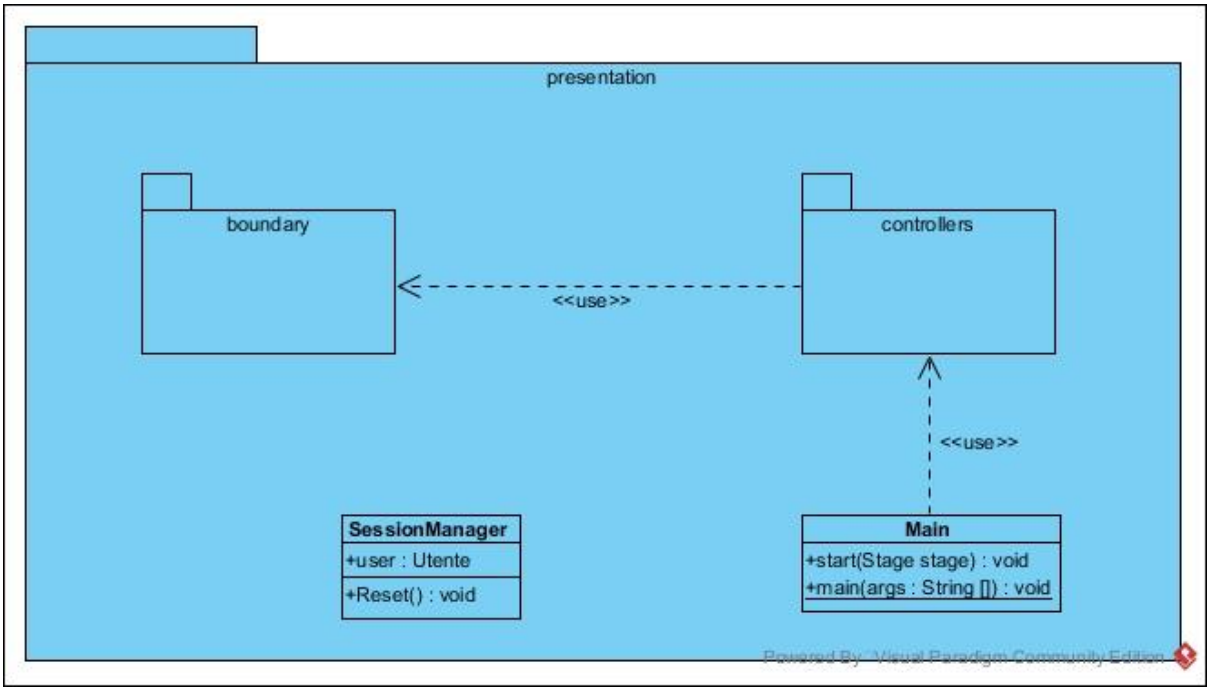
2.1 Diagramma delle Classi

Per una migliore visione dei diagrammi, si consiglia di consultare il file .vpp inerente alla progettazione.

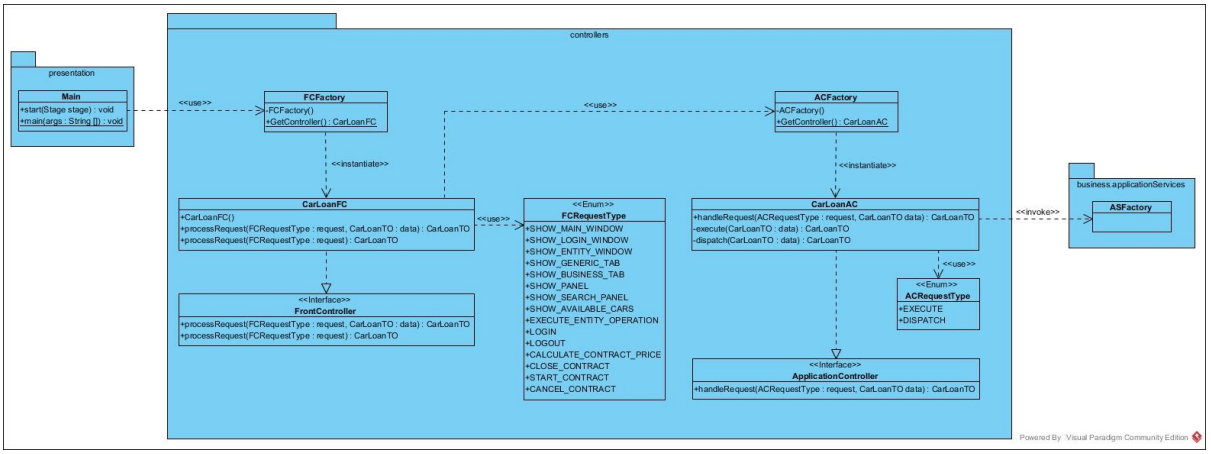
- **CarLoan**



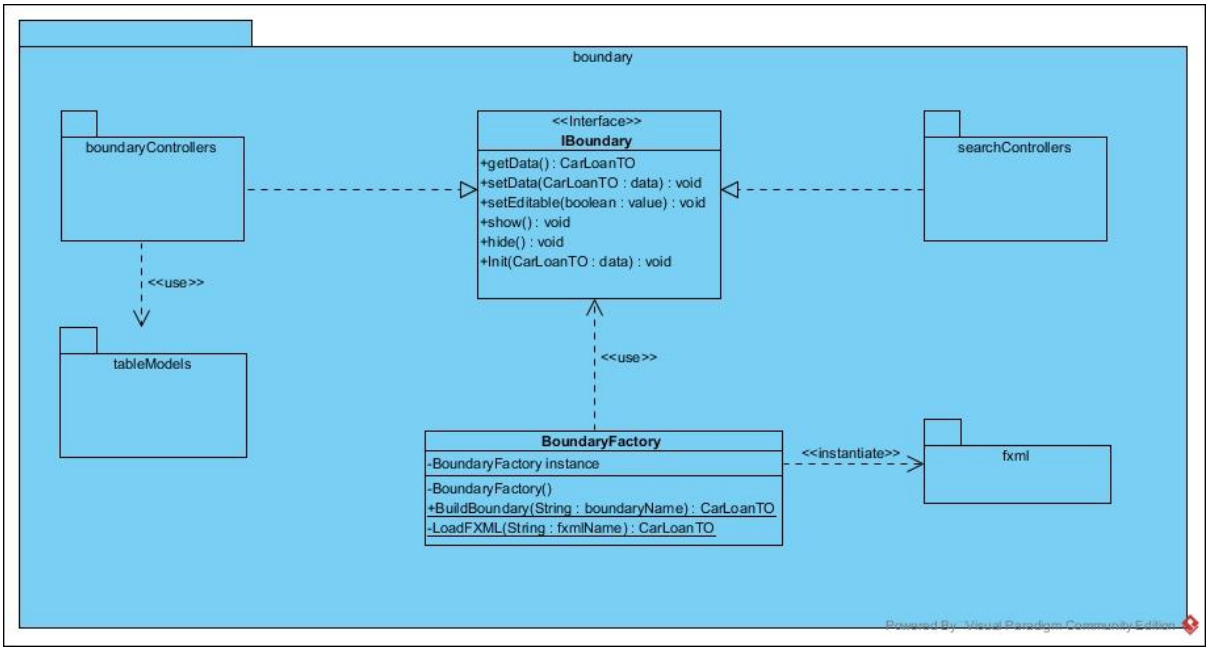
- **Presentation**



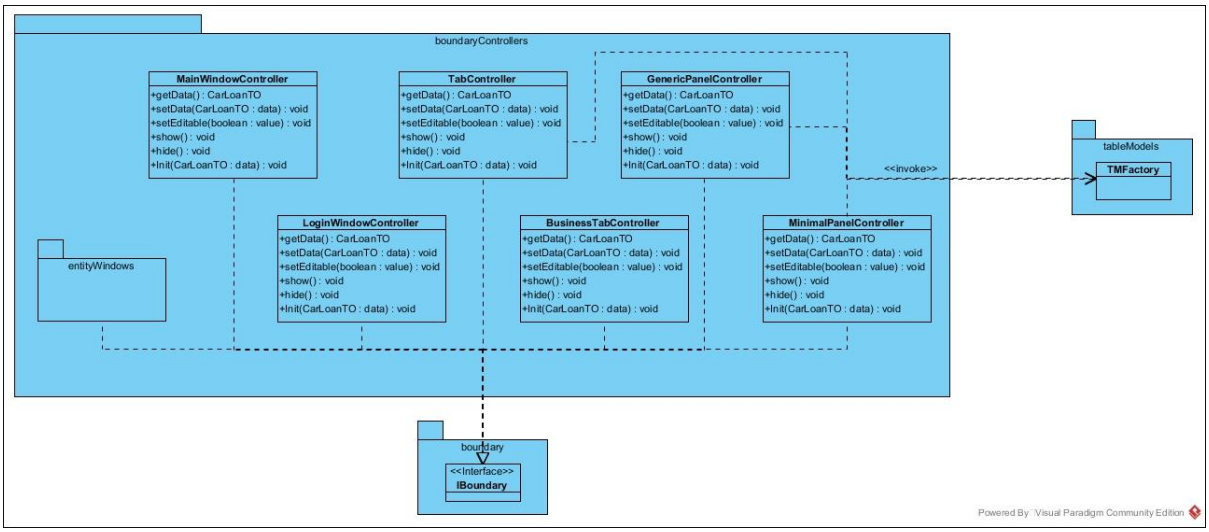
  - **Presentation.Controllers**

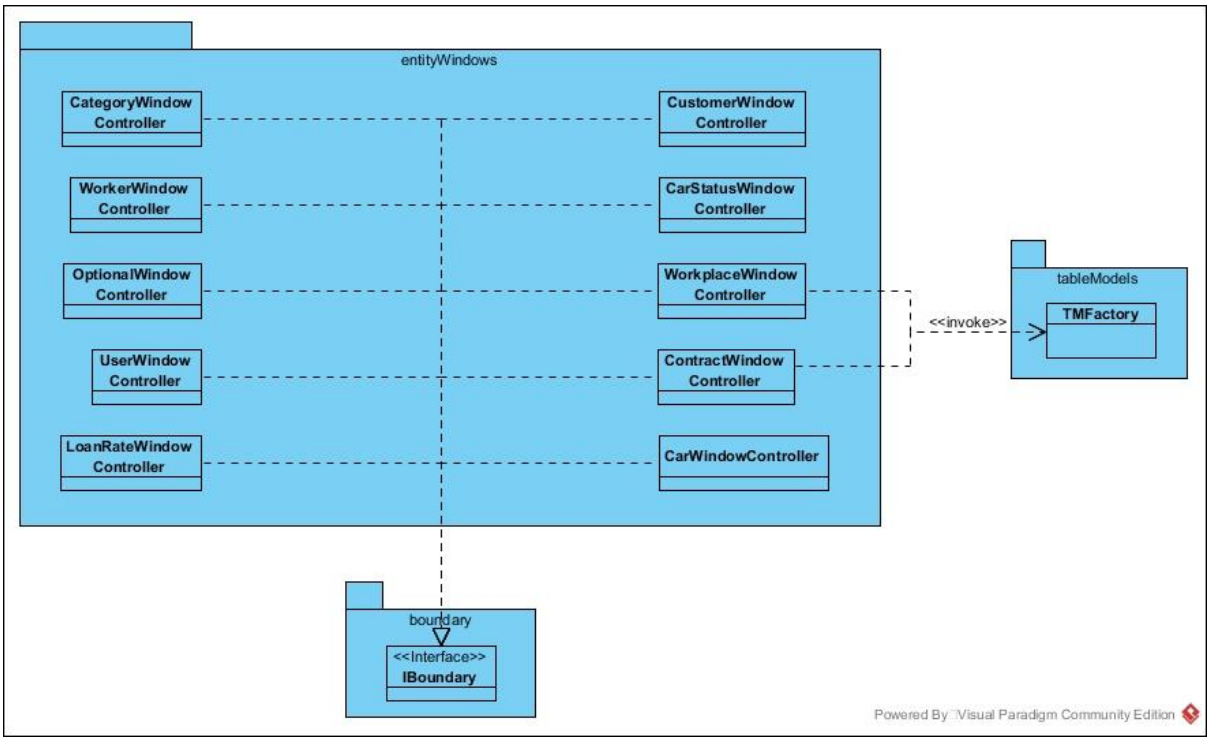


Presentation.Boundary

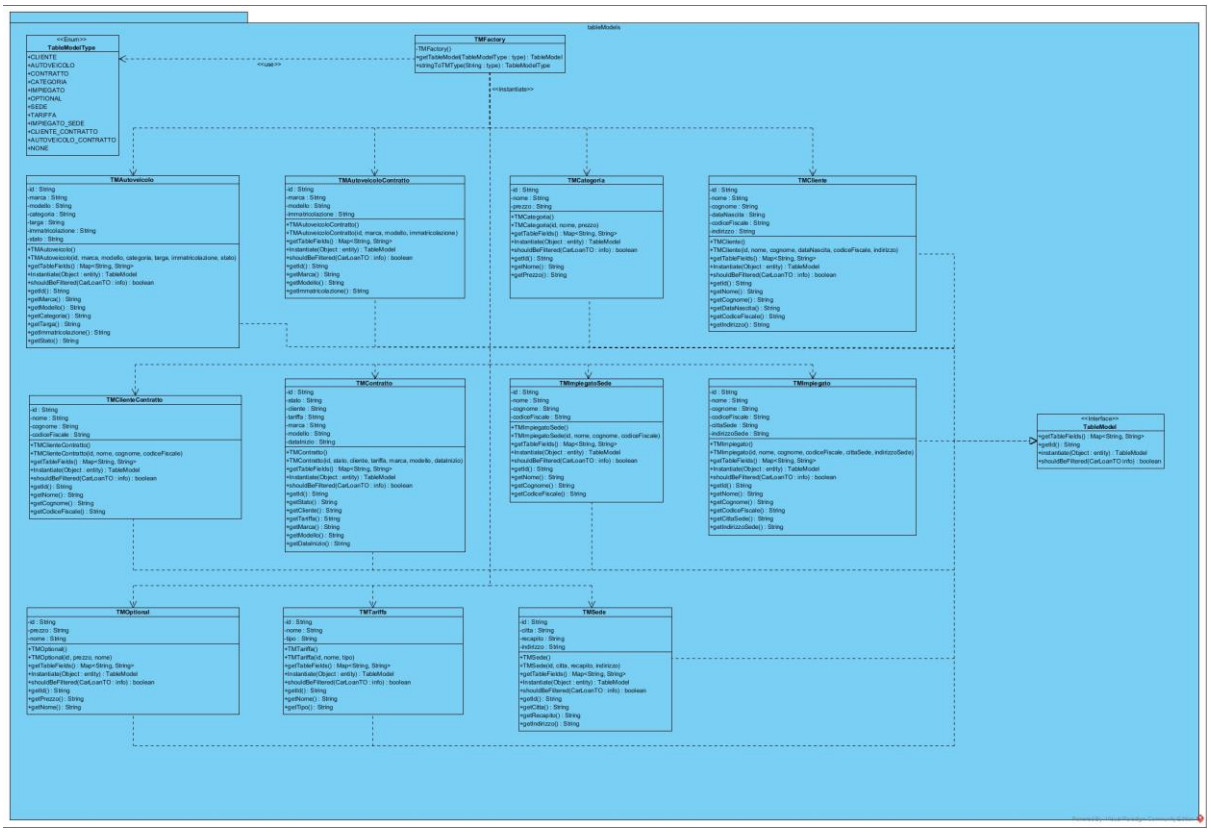


Presentation.Boundary.BoundaryControllers

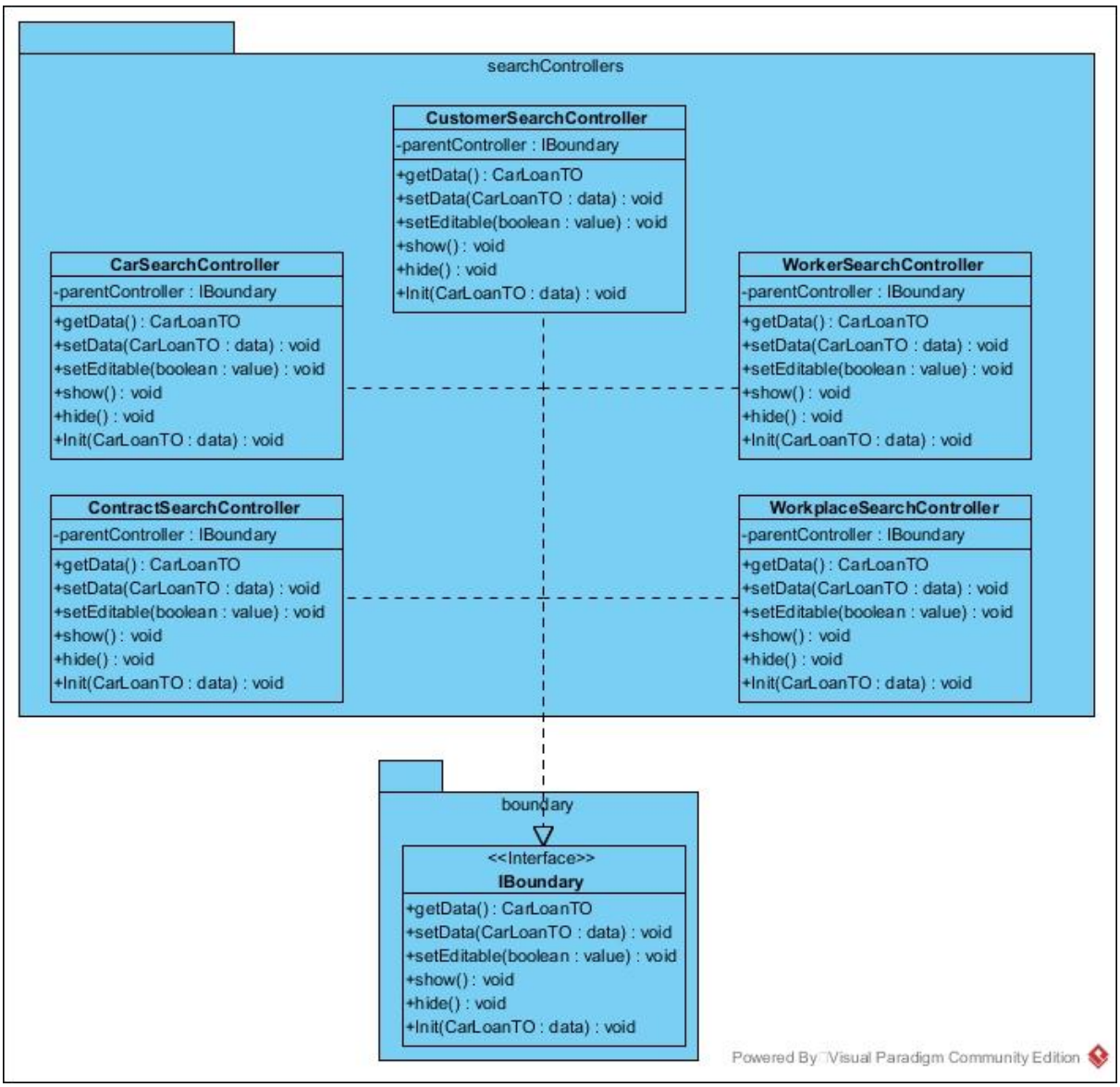




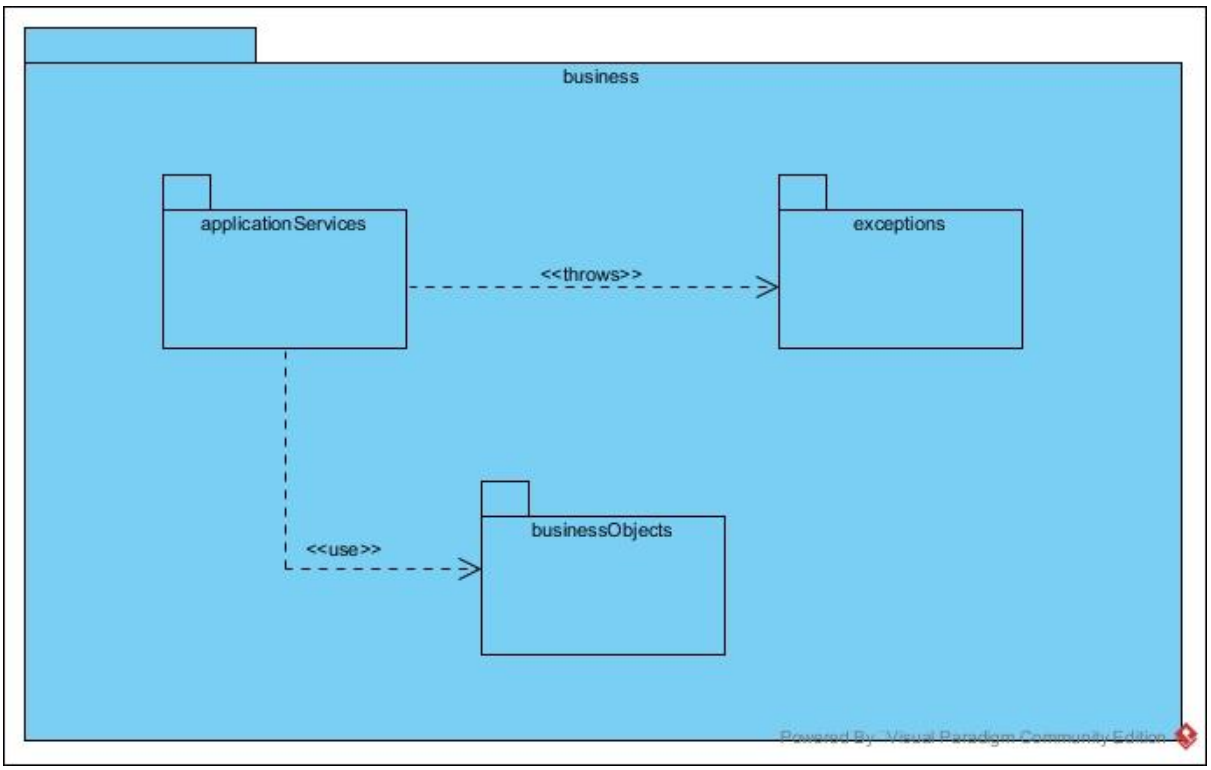
- **Presentation.Boundary.TableModels**



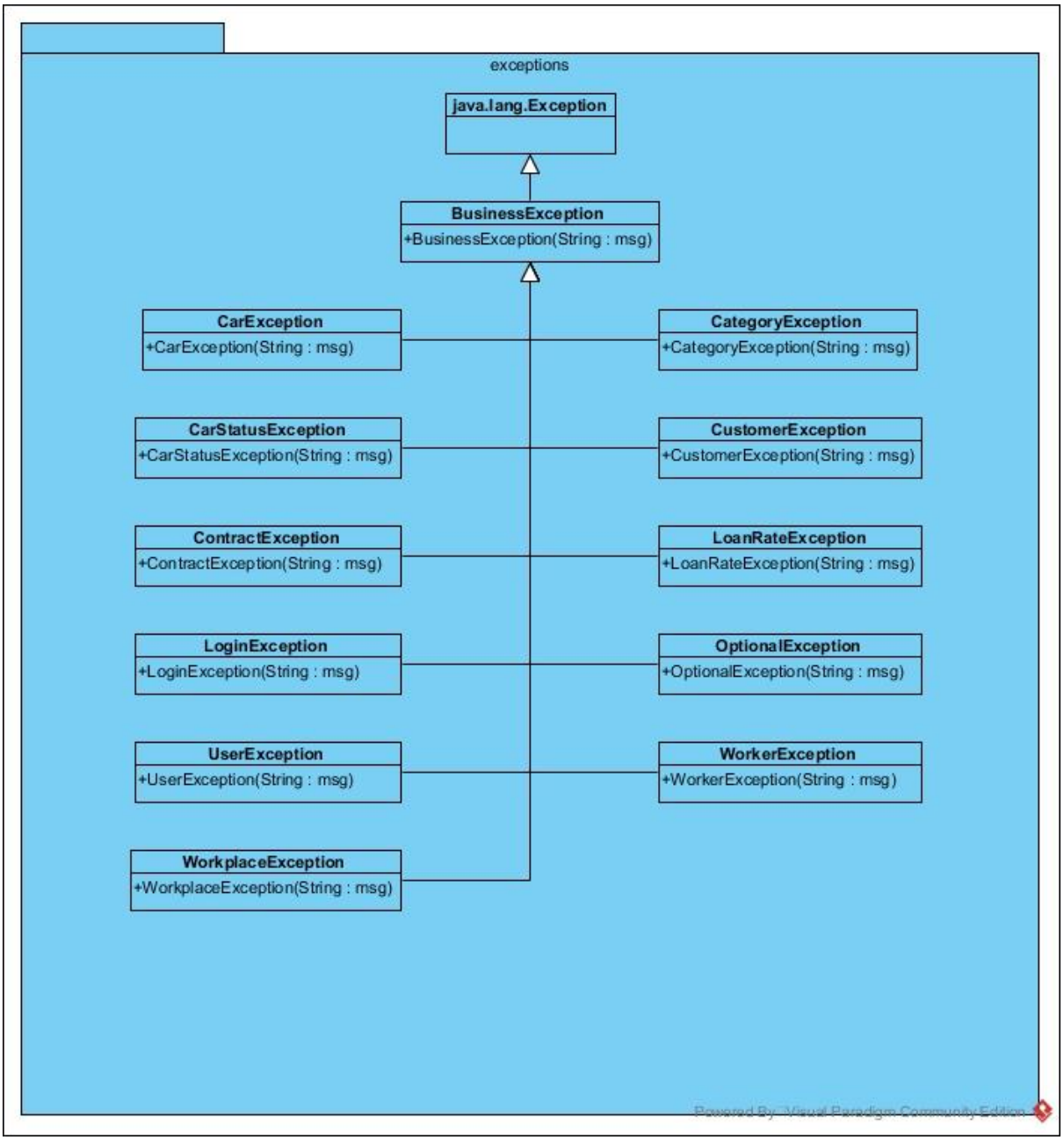
- **Presentation.Boundary.SearchControllers**



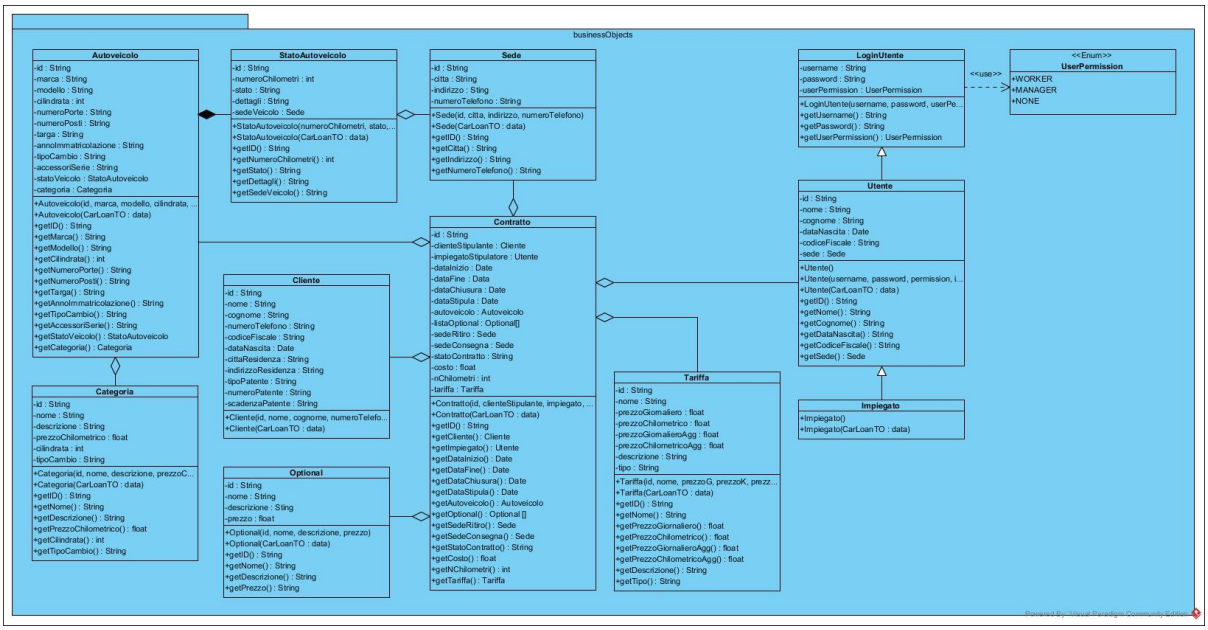
- **Business**


business
applicationServices
exceptions
<<throws>>
businessObjects
<<use>>
Powered By Visual Paradigm Community Edition

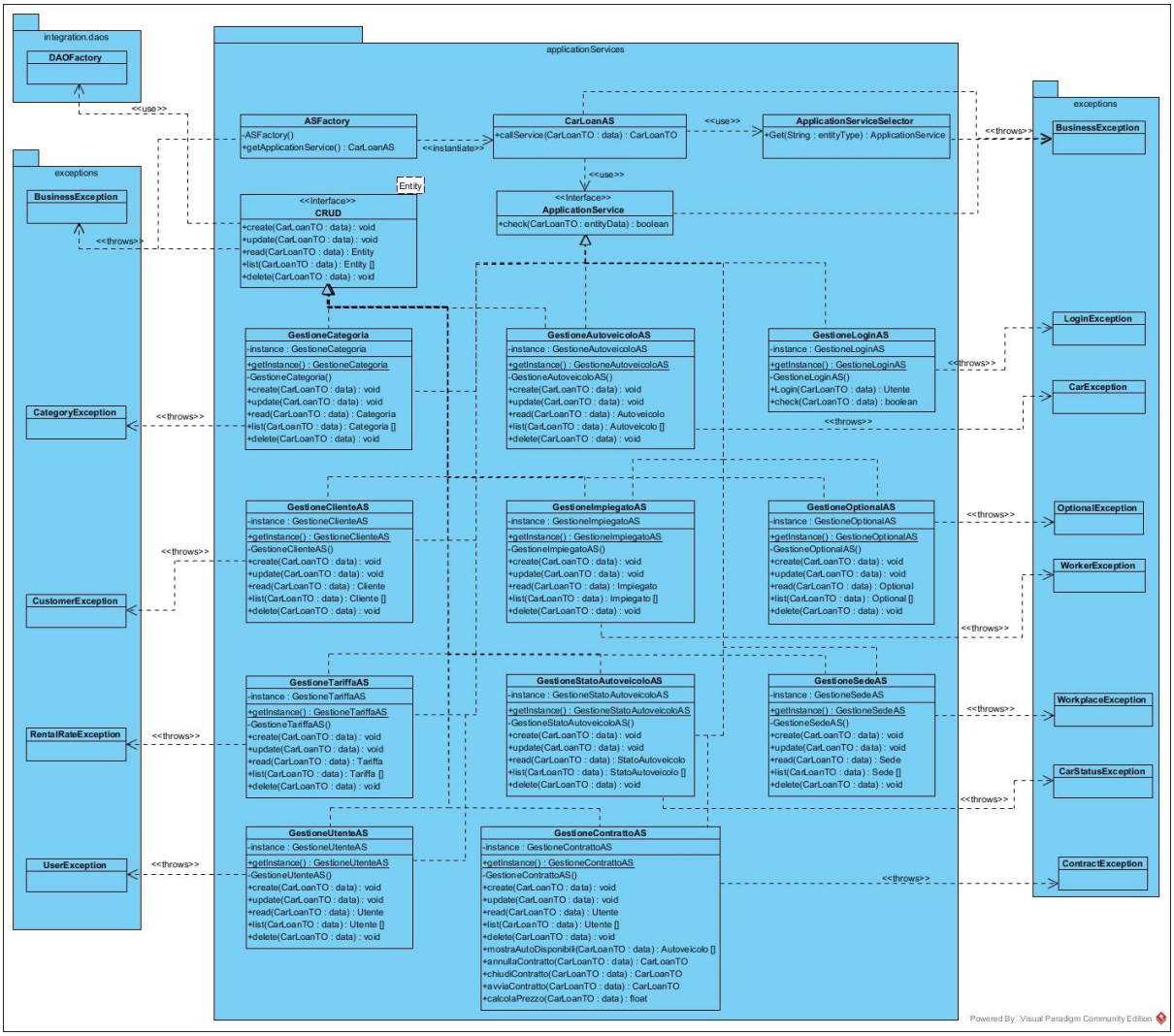
- **Business.Exceptions**



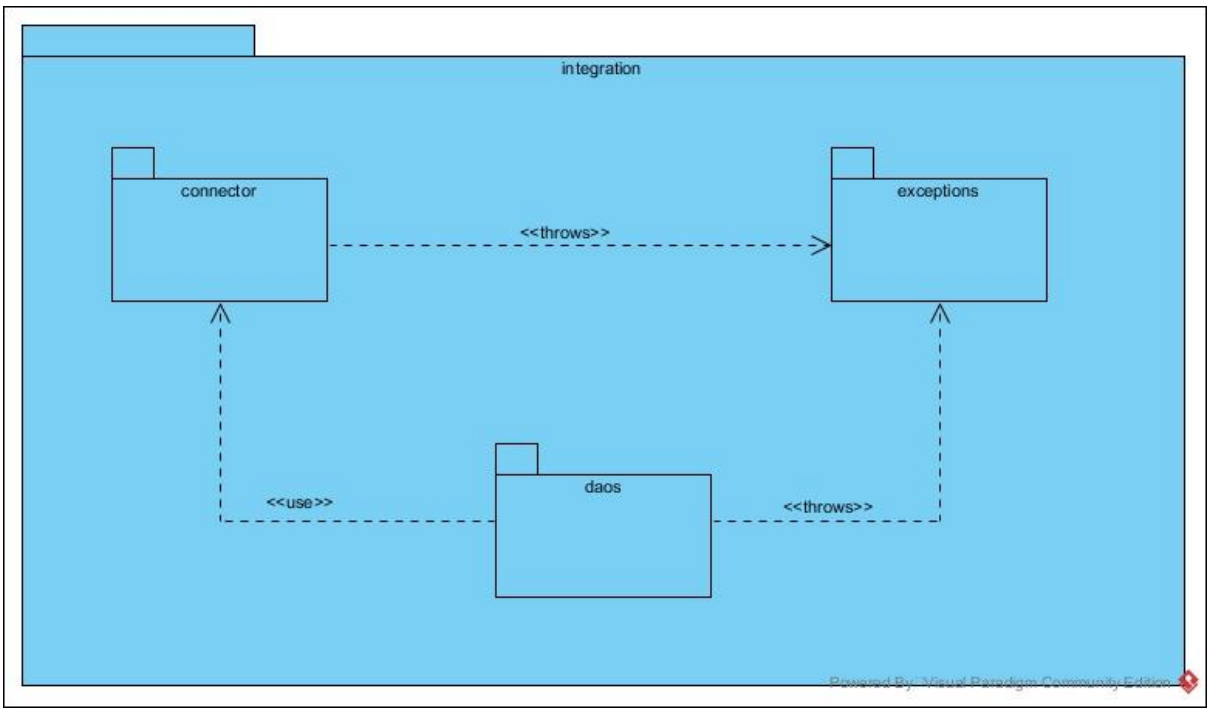
Business.BusinessObjects



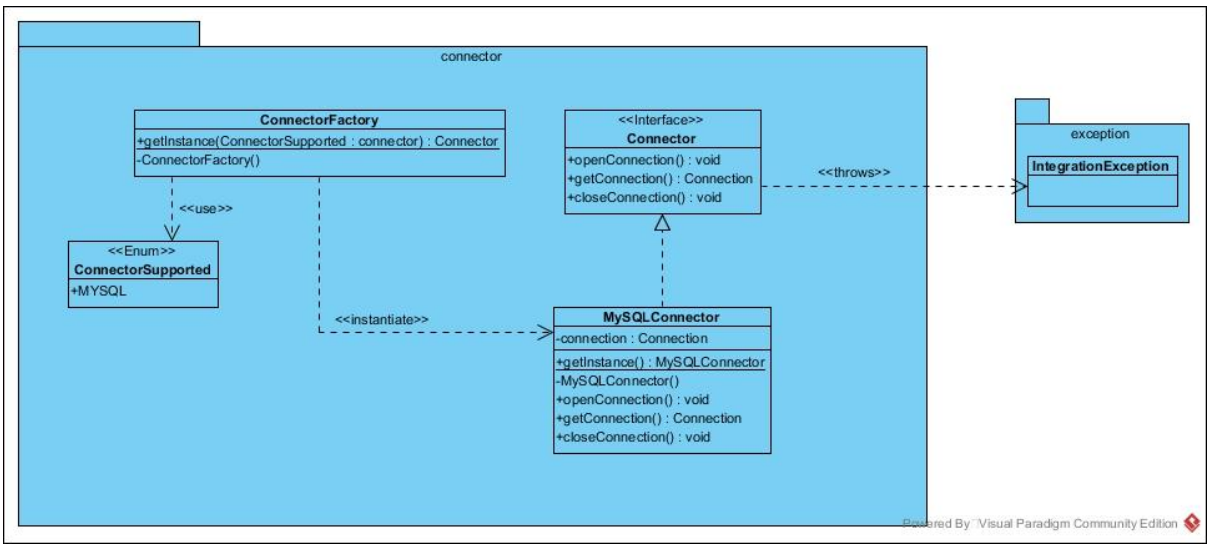
Business.ApplicationServices



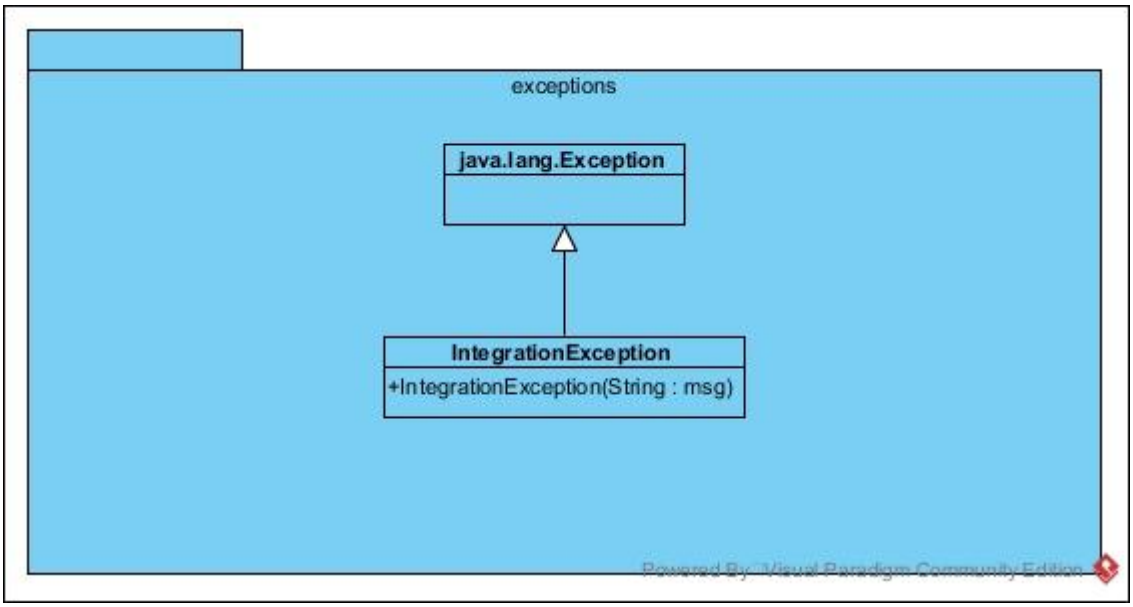
- **Integration**



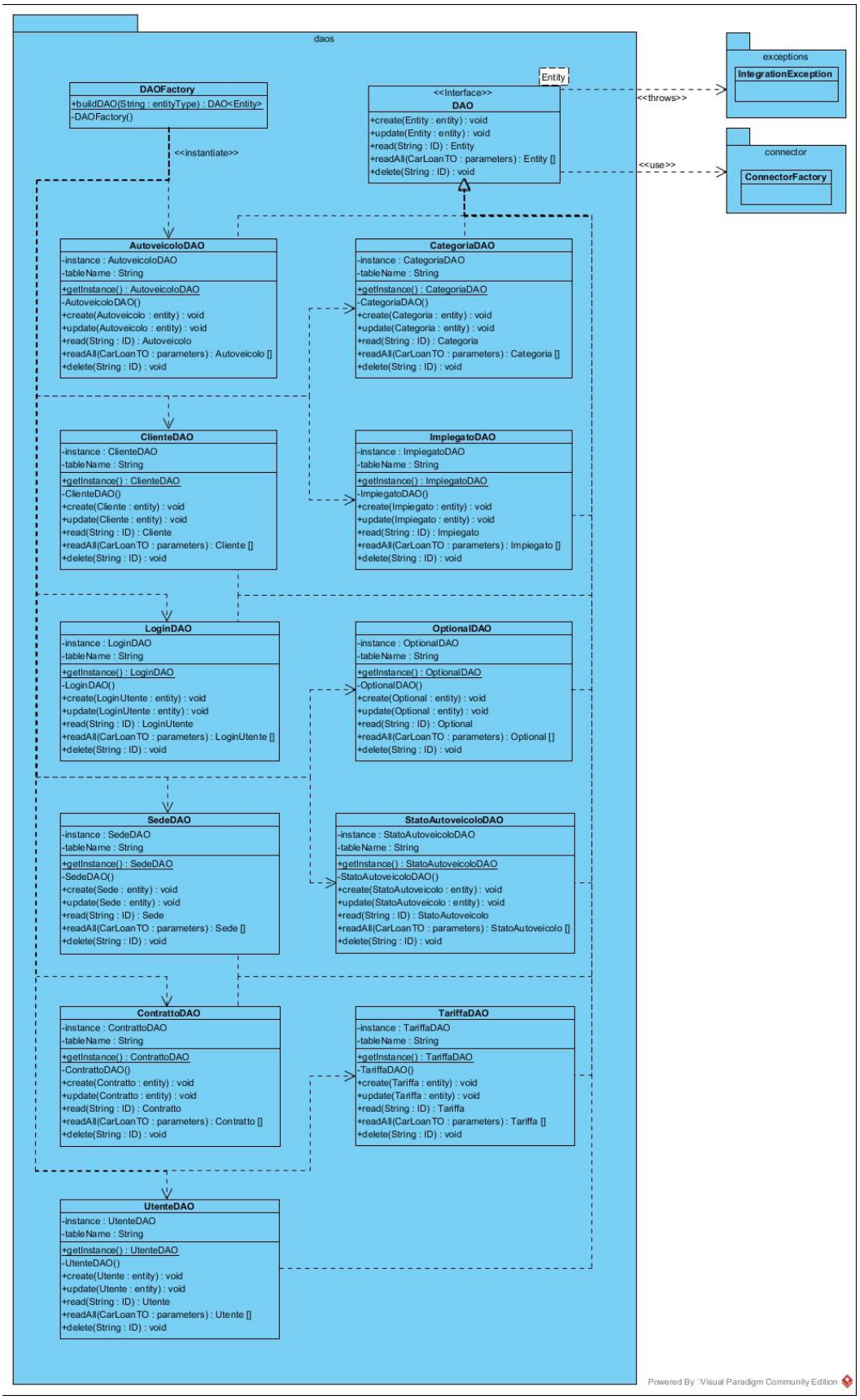
  - **Integration.Connector**



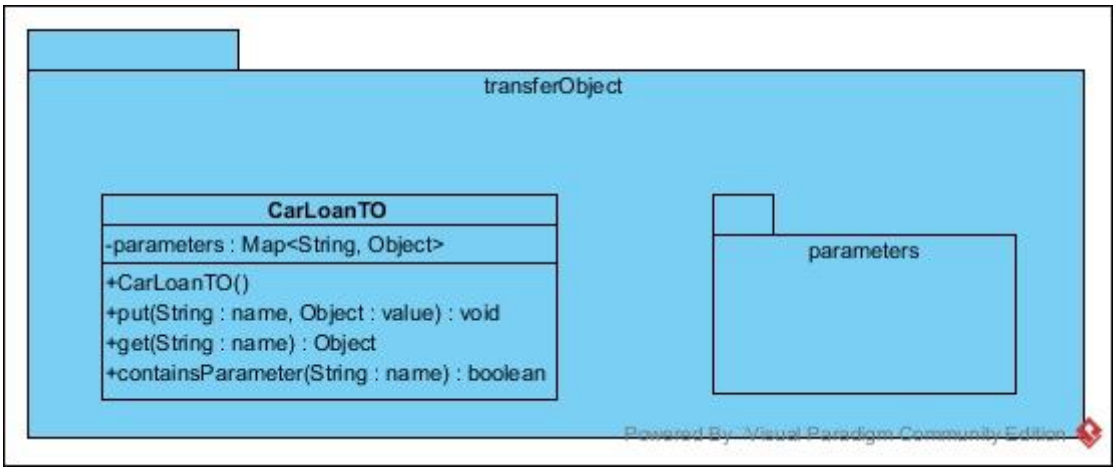
- **Integration.Exceptions**



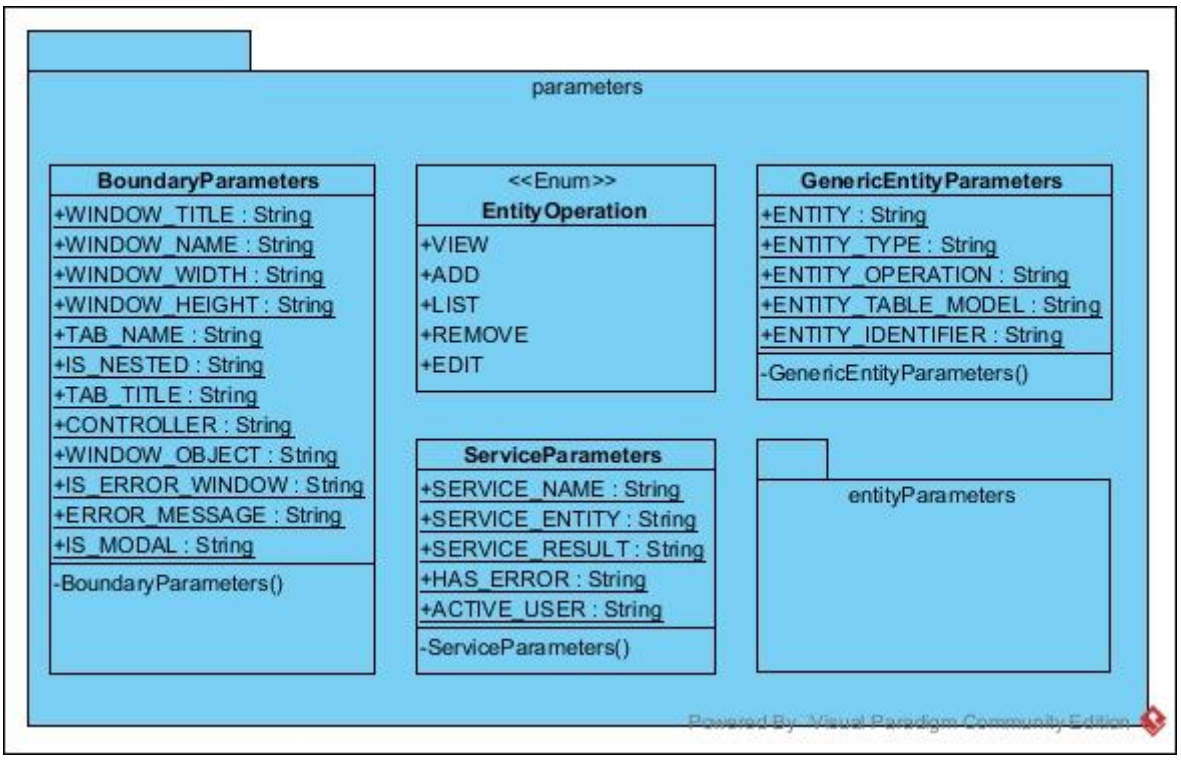
Integration.DAOS

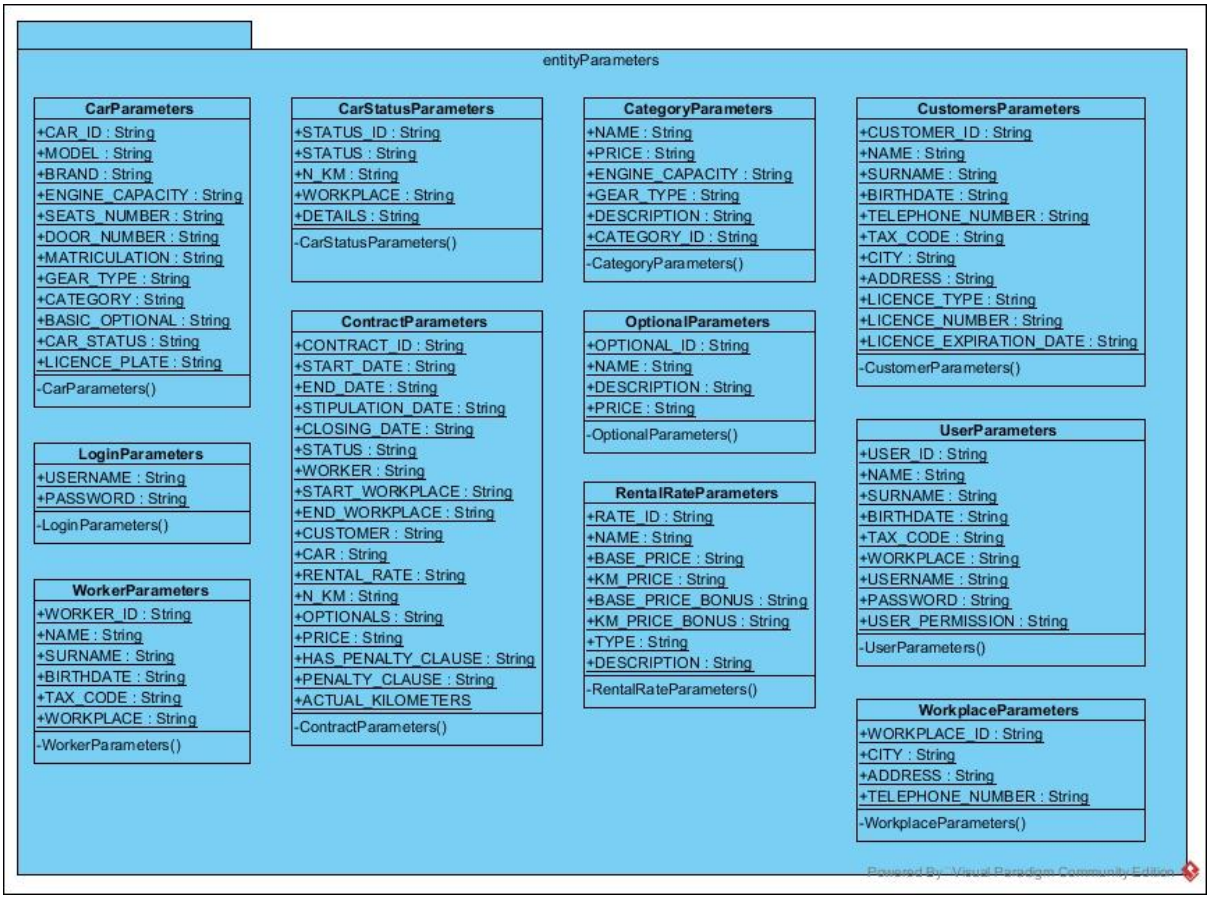


- **TransferObject**



  - **TransferObject.parameters**




entityParameters
CarParameters
+CAR_ID : String
+MODEL : String
+BRAND : String
+ENGINE_CAPACITY : String
+SEATS_NUMBER : String
+DOOR_NUMBER : String
+MATRICULATION : String
+GEAR_TYPE : String
+CATEGORY : String
+BASIC_OPTIONAL : String
+CAR_STATUS : String
+LICENCE_PLATE : String
-CarParameters()
LoginParameters
+USERNAME : String
+PASSWORD : String
-LoginParameters()
WorkerParameters
+WORKER_ID : String
+NAME : String
+SURNAME : String
+BIRTHDATE : String
+TAX_CODE : String
+WORKPLACE : String
-WorkerParameters()
CarStatusParameters
+STATUS_ID : String
+STATUS : String
+N_KM : String
+WORKPLACE : String
+DETAILS : String
-CarStatusParameters()
ContractParameters
+CONTRACT_ID : String
+START_DATE : String
+END_DATE : String
+STIPULATION_DATE : String
+CLOSING_DATE : String
+STATUS : String
+WORKER : String
+START_WORKPLACE : String
+END_WORKPLACE : String
+CUSTOMER : String
+CAR : String
+RENTAL_RATE : String
+N_KM : String
+OPTIONALS : String
+PRICE : String
+HAS_PENALTY_CLAUSE : String
+PENALTY_CLAUSE : String
+ACTUAL_KILOMETERS
-ContractParameters()
CategoryParameters
+NAME : String
+PRICE : String
+ENGINE_CAPACITY : String
+GEAR_TYPE : String
+DESCRIPTION : String
+CATEGORY_ID : String
-CategoryParameters()
OptionalParameters
+OPTIONAL_ID : String
+NAME : String
+DESCRIPTION : String
+PRICE : String
-OptionalParameters()
RentalRateParameters
+RATE_ID : String
+NAME : String
+BASE_PRICE : String
+KM_PRICE : String
+BASE_PRICE_BONUS : String
+KM_PRICE_BONUS : String
+TYPE : String
+DESCRIPTION : String
-RentalRateParameters()
CustomersParameters
+CUSTOMER_ID : String
+NAME : String
+SURNAME : String
+BIRTHDATE : String
+TELEPHONE_NUMBER : String
+TAX_CODE : String
+CITY : String
+ADDRESS : String
+LICENCE_TYPE : String
+LICENCE_NUMBER : String
+LICENCE_EXPIRATION_DATE : String
-CustomerParameters()
UserParameters
+USER_ID : String
+NAME : String
+SURNAME : String
+BIRTHDATE : String
+TAX_CODE : String
+WORKPLACE : String
+USERNAME : String
+PASSWORD : String
+USER_PERMISSION : String
-UserParameters()
WorkplaceParameters
+WORKPLACE_ID : String
+CITY : String
+ADDRESS : String
+TELEPHONE_NUMBER : String
-WorkplaceParameters()
Powered By: Visual Paradigm Community Edition

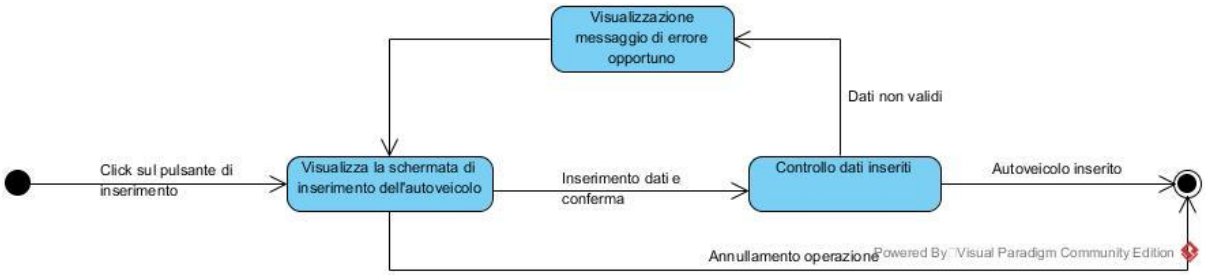
2.2 Specifiche delle Classi

Per favorire la leggibilità del documento, è stato deciso di dividere la specifica delle classi per livelli architetturali di appartenenza. I link per i vari documenti sono elencati di seguito:

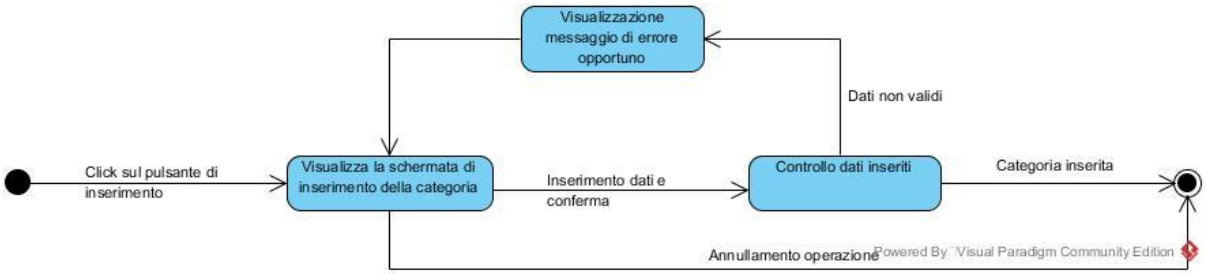
- Presentation
- Business
- Integration
- Transfer Object

2.3 Diagramma degli Stati

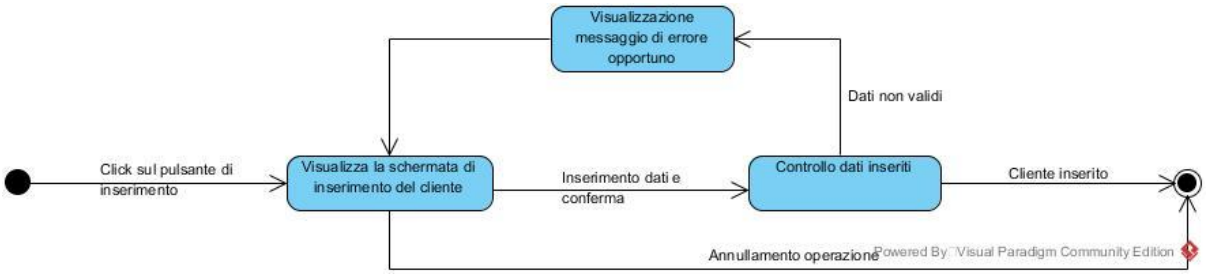
- Aggiungi Autoveicolo



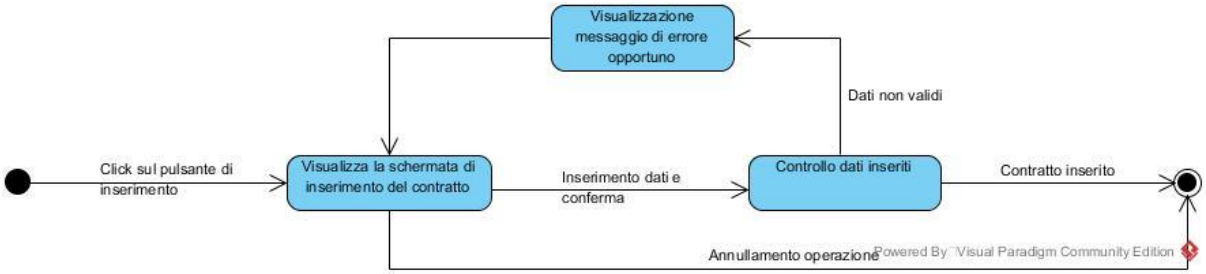
- Aggiungi Categoria



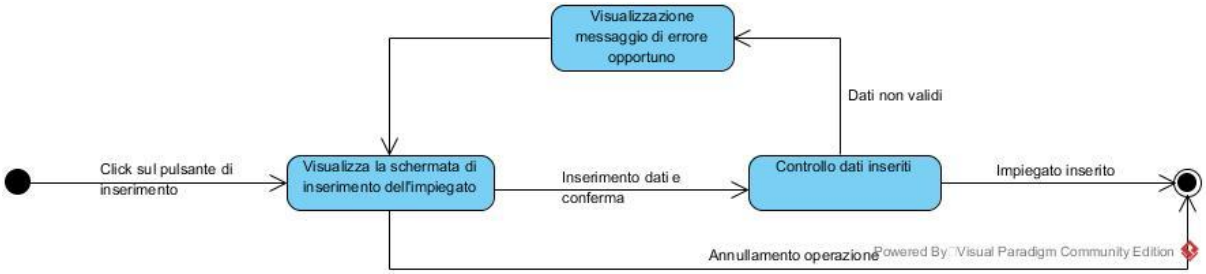
- Aggiungi Cliente



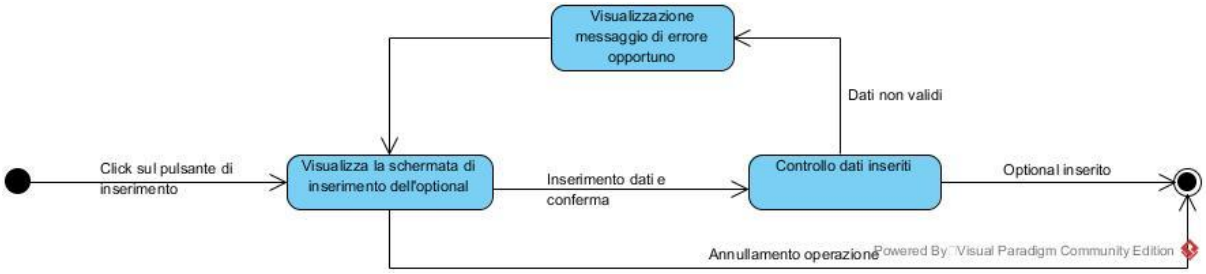
- Aggiungi Contratto



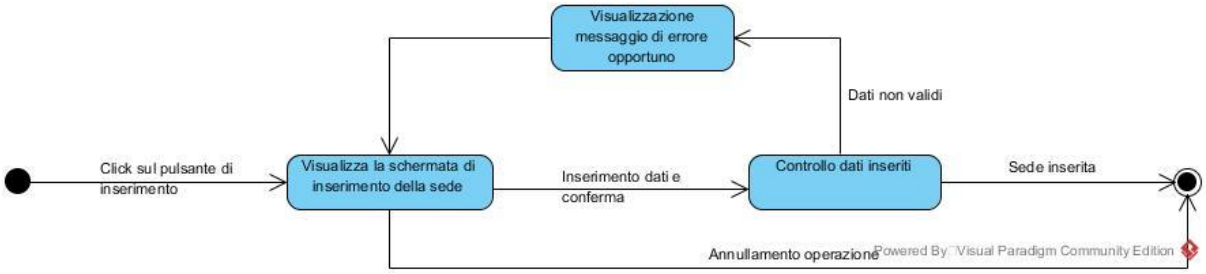
- Aggiungi Impiegato



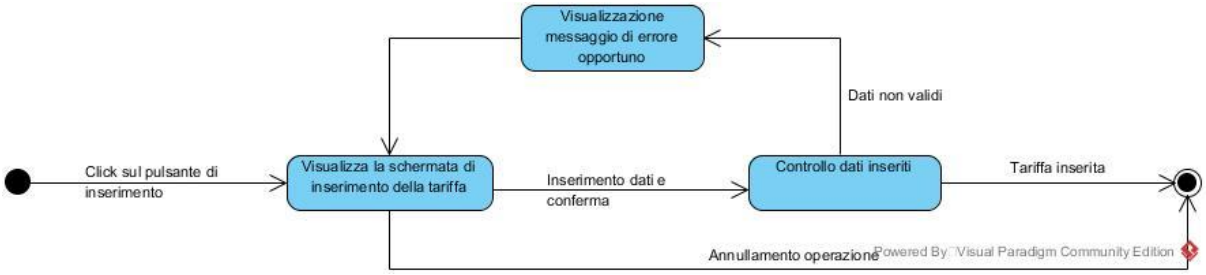
- Aggiungi Optional



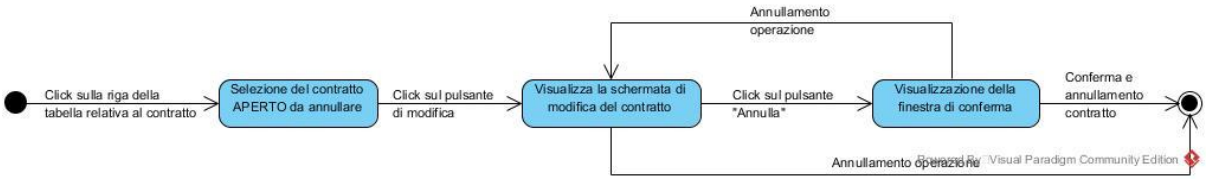
- Aggiungi Sede



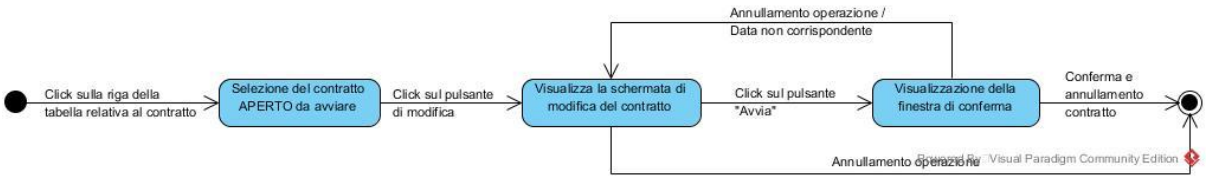
- Aggiungi Tariffa



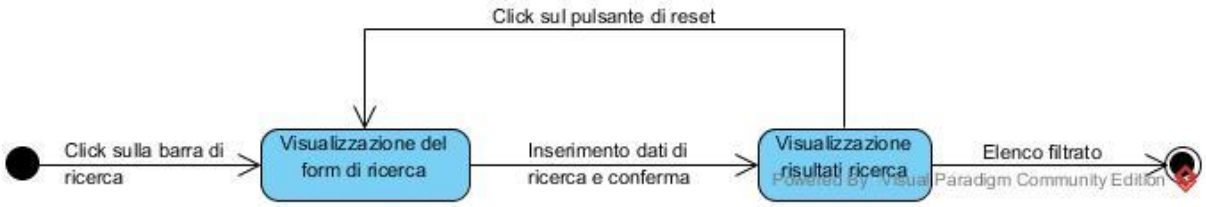
- Annulla Contratto



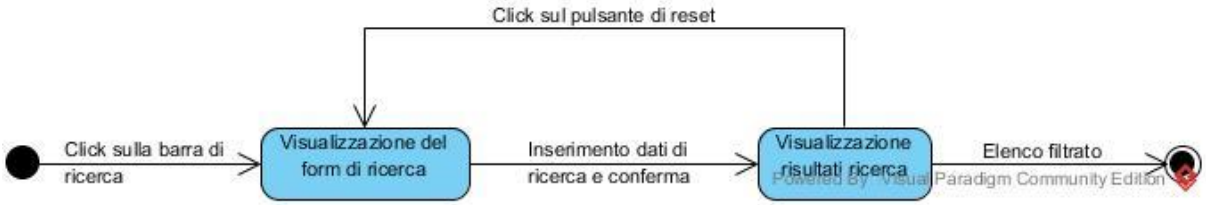
- Avvia Contratto



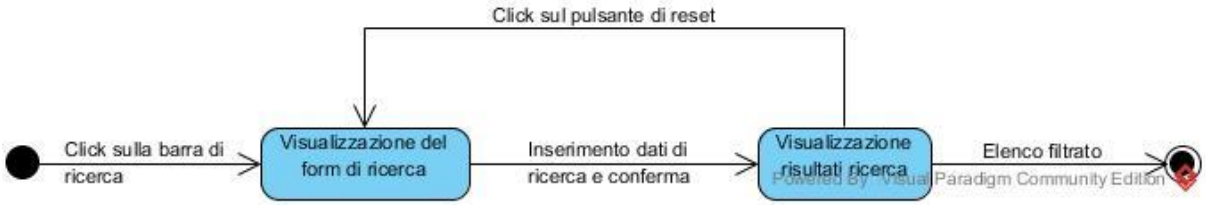
- Cerca Autoveicolo



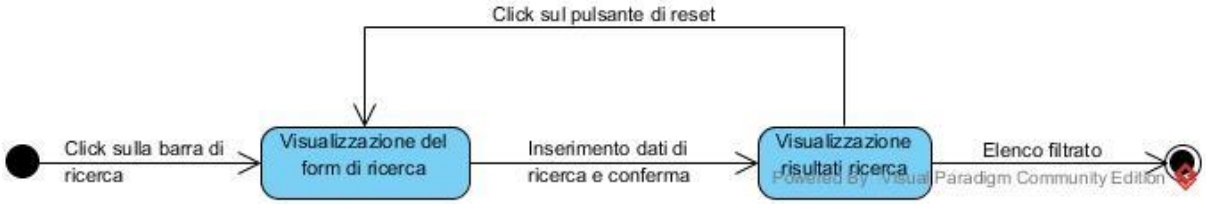
- Cerca Cliente



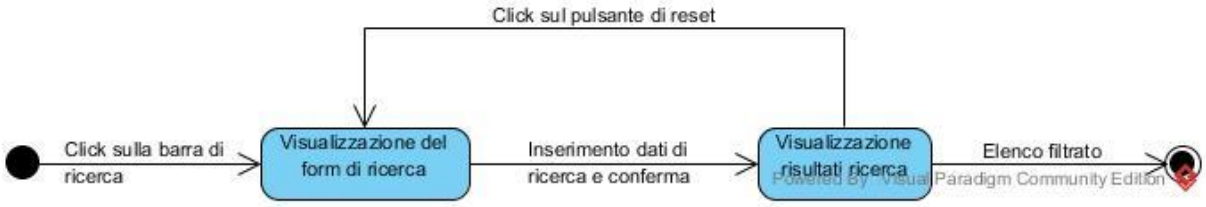
- Cerca Contratto



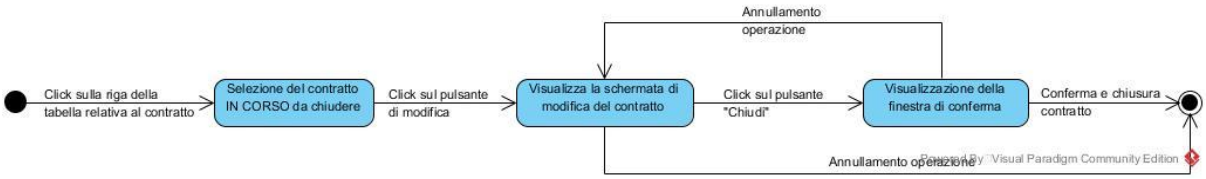
- Cerca Impiegato



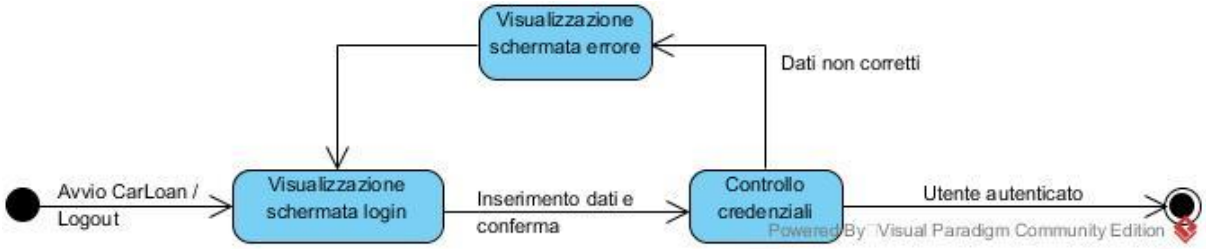
- Cerca Sede



- Chiudi Contratto

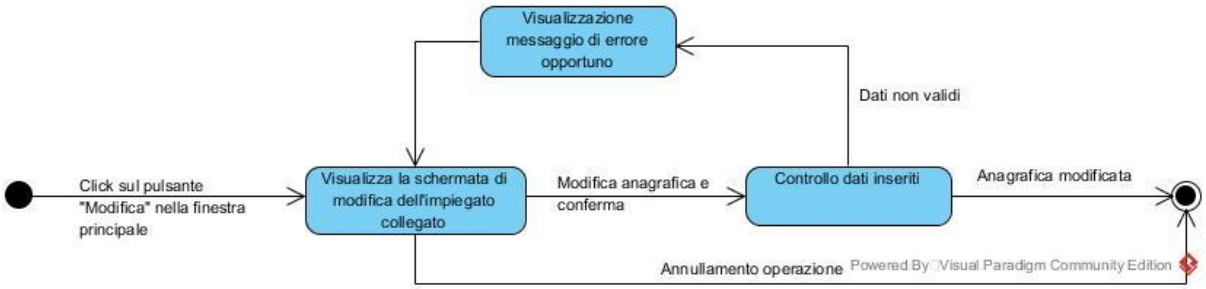


- Login

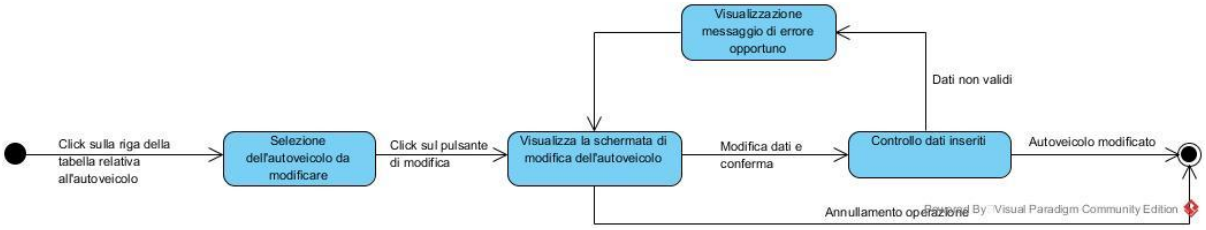


- Logout

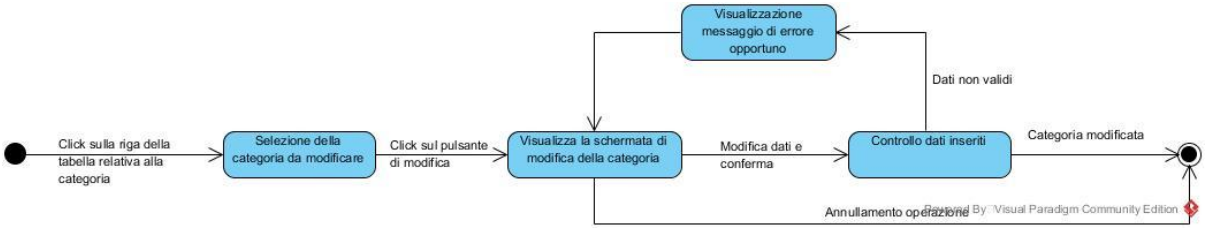
- Modifica Anagrafica



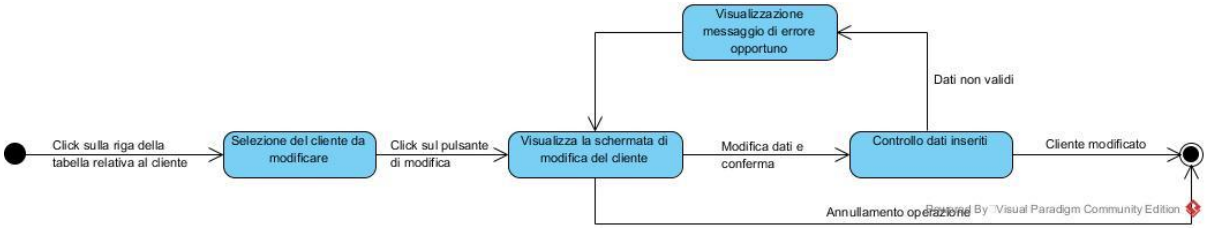
- Modifica Autoveicolo



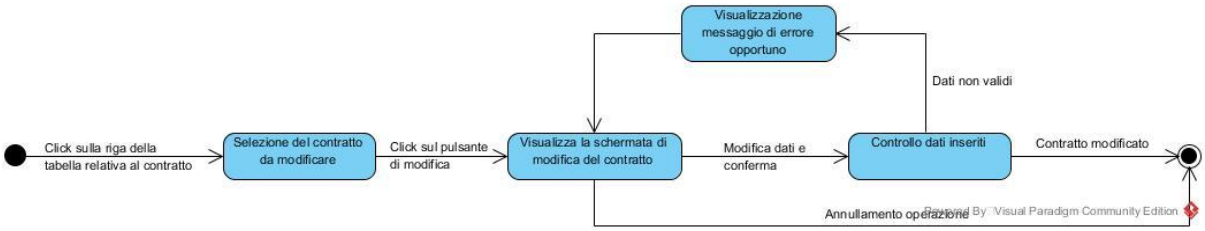
- Modifica Categoria



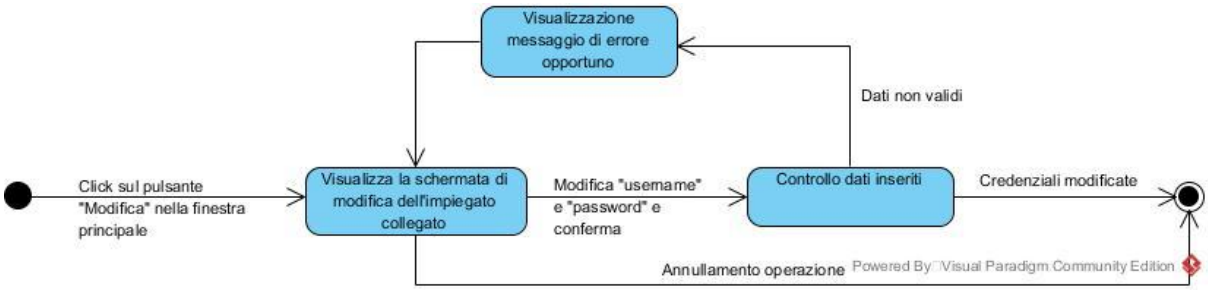
- Modifica Cliente



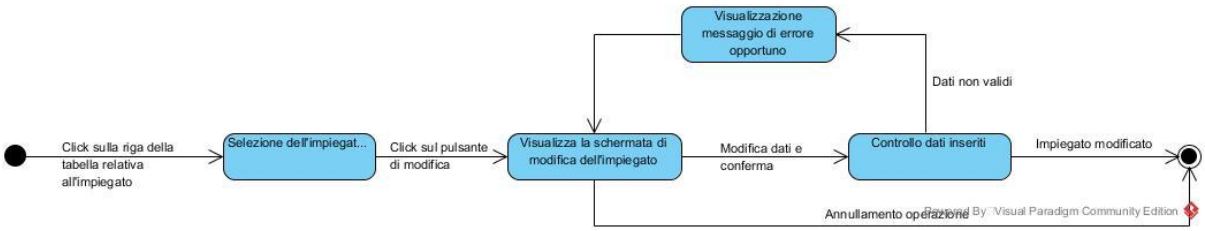
- Modifica Contratto



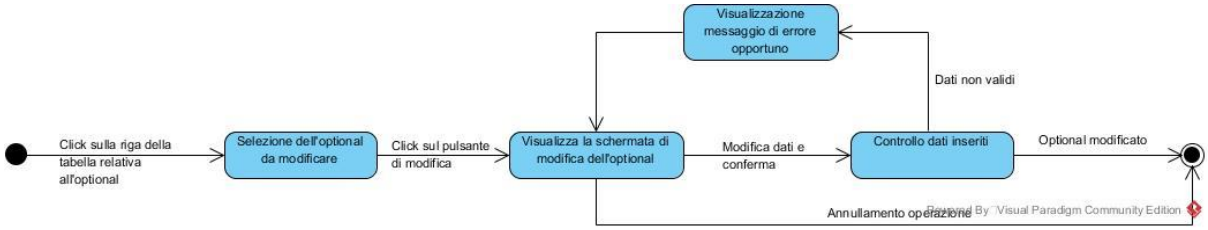
- Modifica Credenziali



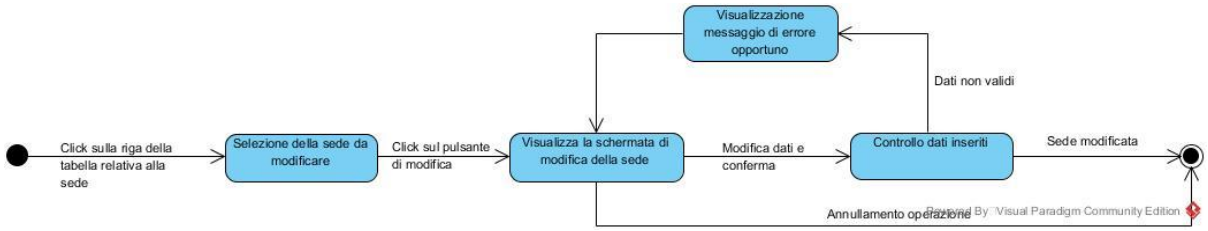
- Modifica Impiegato



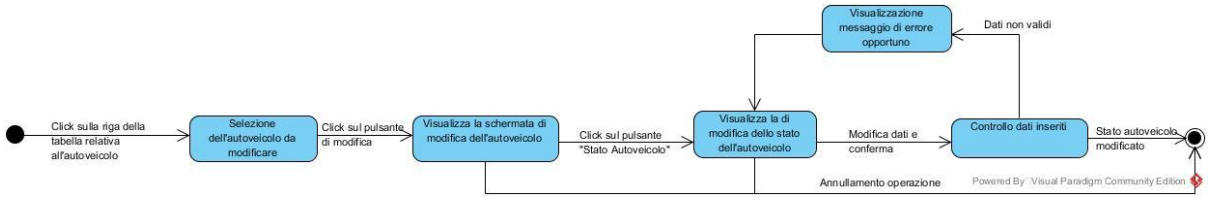
- Modifica Optional



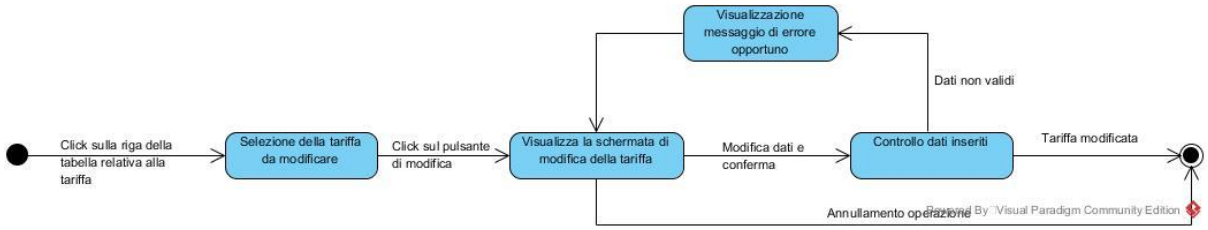
- Modifica Sede



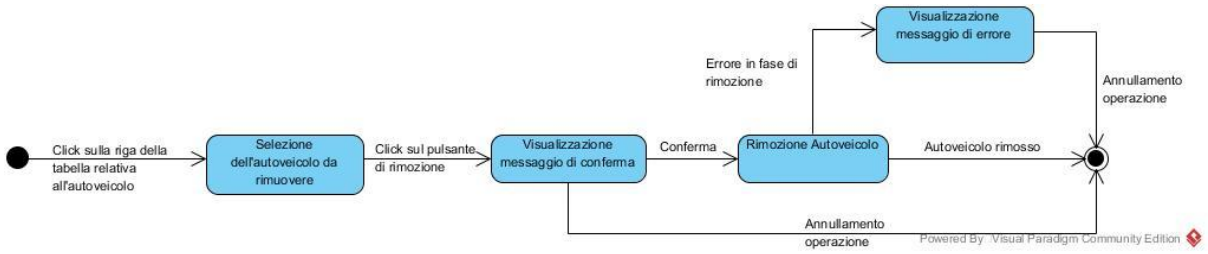
- Modifica Stato Autoveicolo



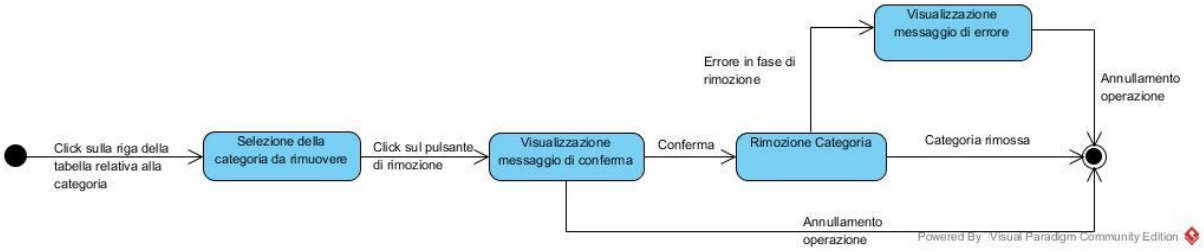
- Modifica Tariffa



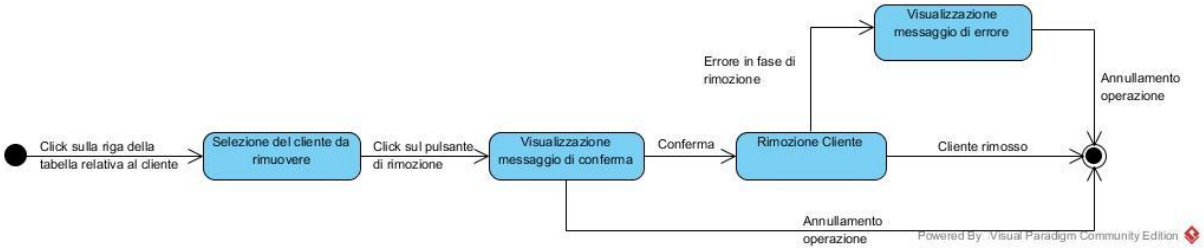
- Rimuovi Autoveicolo



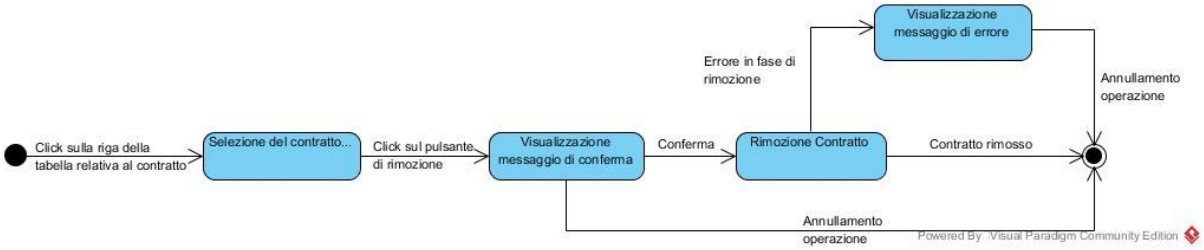
- Rimuovi Categoria



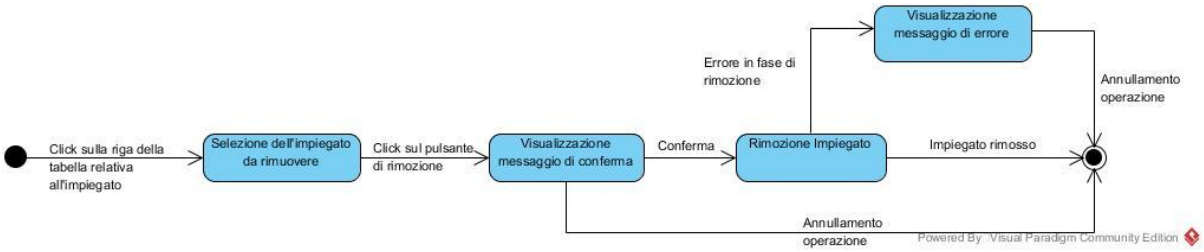
- Rimuovi Cliente



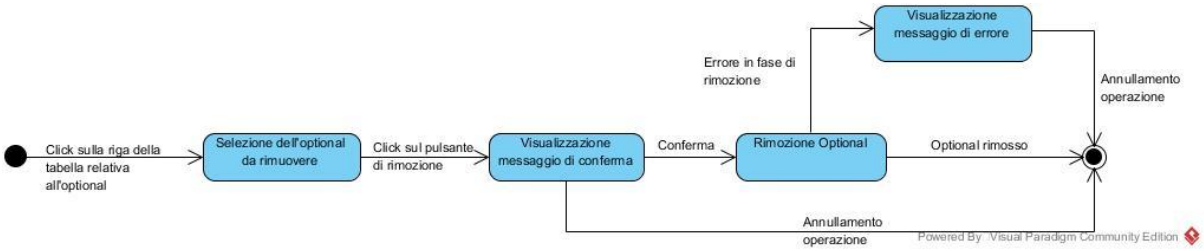
- Rimuovi Contratto



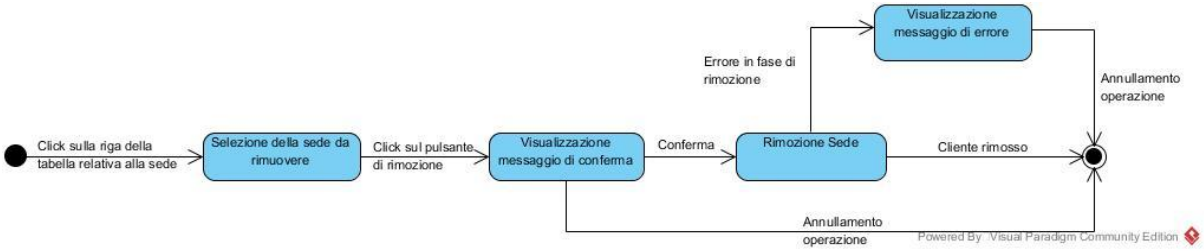
- Rimuovi Impiegato



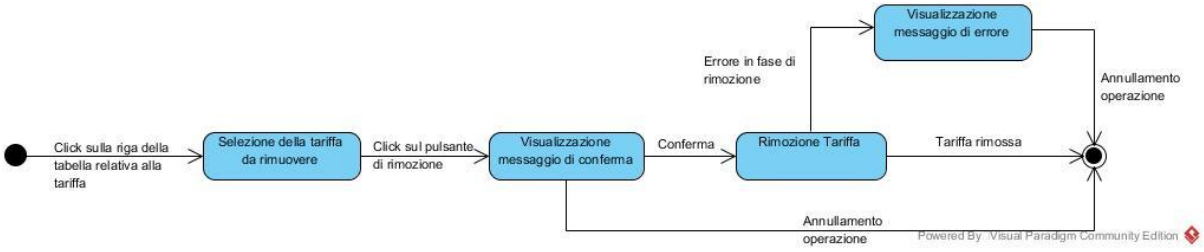
- Rimuovi Optional



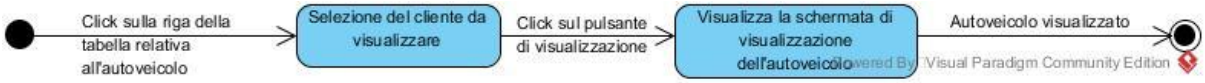
- Rimuovi Sede



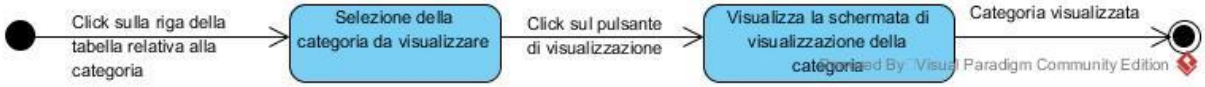
- Rimuovi Tariffa



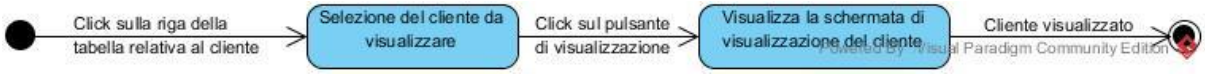
- Visualizza Autoveicolo



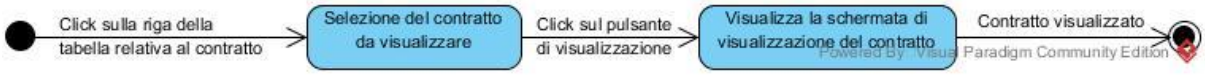
- Visualizza Categoria



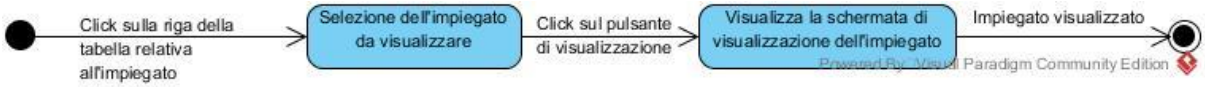
- Visualizza Cliente



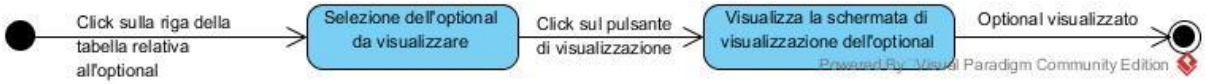
- Visualizza Contratto



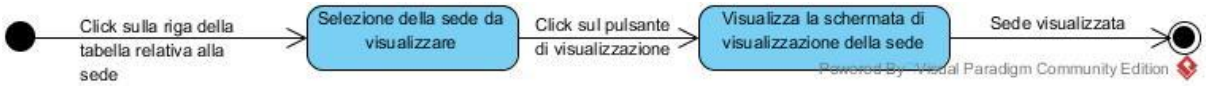
- Visualizza Impiegato



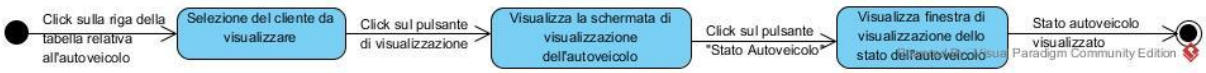
- Visualizza Optional



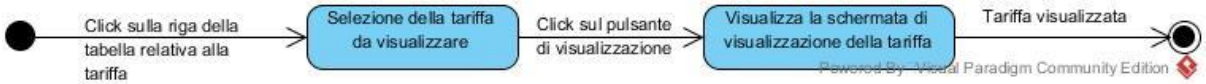
- Visualizza Sede



- Visualizza Stato Autoveicolo



- Visualizza Tariffa



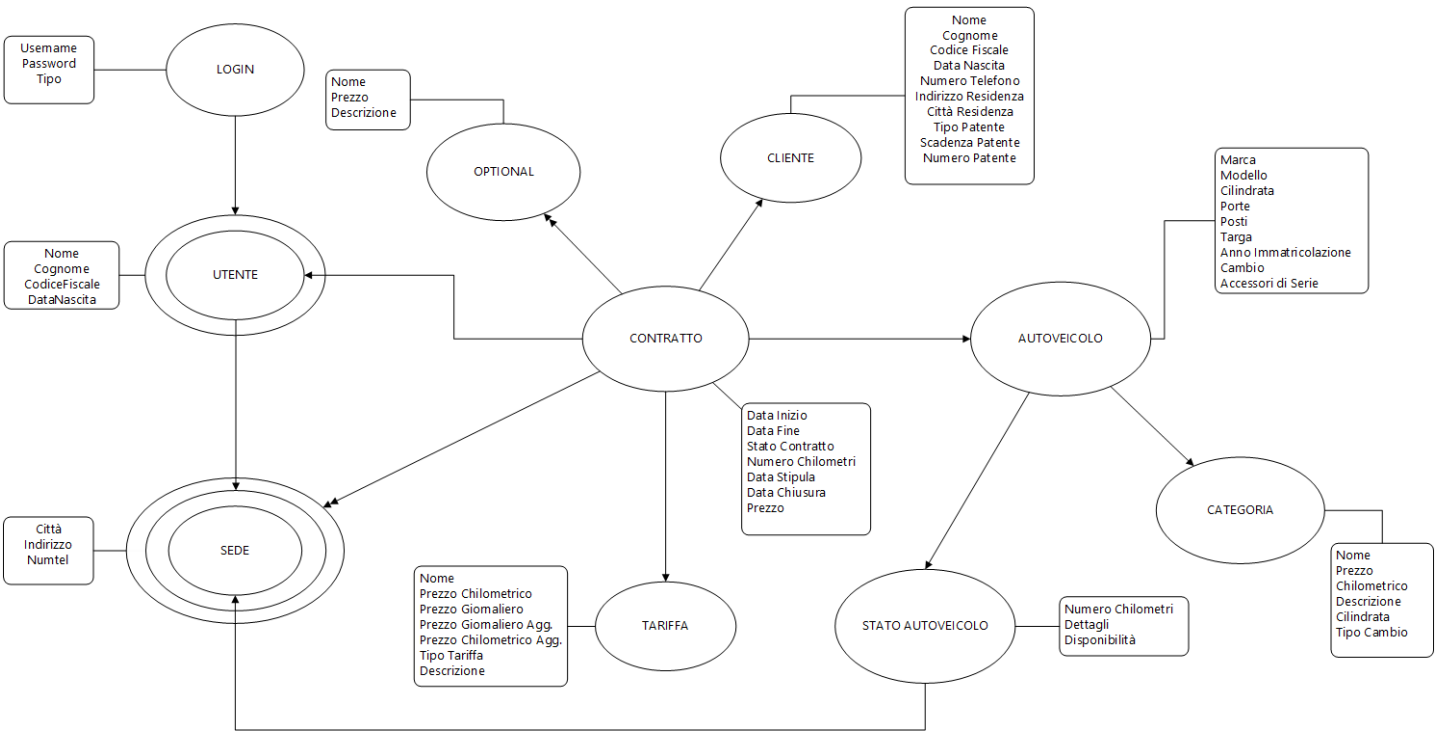
2.4 Diagrammi di Sequenza

Per favorire la leggibilità del documento, i diagrammi di sequenza sono inseriti in un documento separato da questo raggiungibile da qui.
In ogni caso, è consigliabile consultare il file .vpp inerente alla progettazione per una visione più chiara.

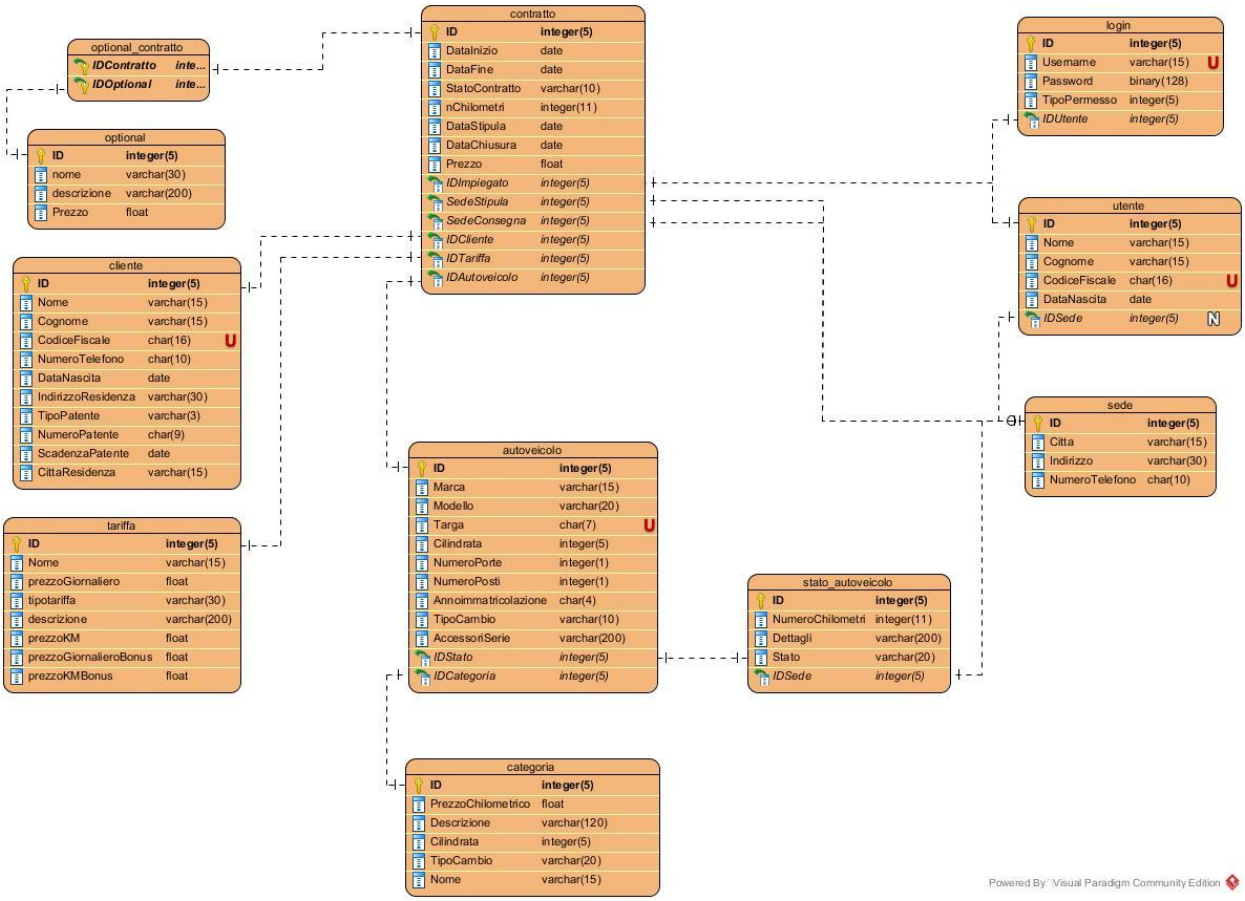
3. PROGETTO DEI DATI

3.1 Database

3.1.1 Diagramma delle Dipendenze dei Dati



3.1.2 Modello del Database



3.1.3 Dettaglio dei Dati

Per favorire la leggibilità di questo documento, si è deciso di fornire il dettaglio dei dati in un documento separato, raggiungibile da QUI.

4. APPENDICE

4.1 Pattern utilizzati

- **Pattern Architetturali**

Front Controller

Permette di centralizzare la gestione delle richieste provenienti dalle finestre, fornendo un unico punto d'accesso al livello di business e tenendo le view all'oscuro sulla logica di navigazione tra le stesse.
L'utilizzo del front controller rende il sistema più manutenibile dato che assicura una netta separazione tra livello di presentazione e di business evitando inoltre la duplicazione di codice.

Application Controller

L'utilizzo dell'application controller permette di centralizzare la logica di esecuzione dei servizi e la logica di dispatch delle GUI del programma.
La logica di esecuzione dei servizi si rifà all'application service del sistema mentre la logica di dispatch delle GUI sfrutta un BoundaryFactory che istanzierà finestre fxml su richiesta.

Application Service

Gli Application Service forniscono un accesso ai servizi forniti dal livello di business che incapsulano al loro interno la logica associata alla gestione di ogni oggetto di business.

Data Access Object

L'uso del Data Access Object permette di separare la logica di business dalla logica di persistenza dati.
È stato realizzato mediante l'uso di un'interfaccia comune realizzata poi da un Data Access Object per ogni oggetto di business: in questo modo si garantisce un'interfaccia di accesso standard alle operazioni e si applica il principio di segregazione delle interfacce in quanto ogni oggetto di business dipende da una sola interfaccia DAO specifica per quell'oggetto.

Transfer Object

Si è scelto di utilizzare il Transfer Object per diminuire il grado di accoppiamento tra i vari livelli, quindi "impacchettando" all'interno del transfer object (realizzato tramite un hashmap nel sistema) i parametri delle richieste o i dati da trasferire tra i vari livelli. L'uso del Transfer Object consente anche di trasportare molti dati con un basso numero di chiamate di funzione.

- **Design Pattern**

Singleton

Si è pensato di realizzare diverse classi seguendo il Singleton Design Pattern, in particolare tra le classi DAO e ApplicationService perché si è ritenuto fosse superfluo e poco efficente mantenere in memoria più istanze della stessa classe.

Abstract Factory

Si è ricorso all'uso dell'Abstract Factory Design Pattern in più occasioni: per la creazione dei controller (FrontController e ApplicationController), per l'istanziazione dei differenti DAO e per il caricamento delle finestre fxml.
L'uso di tale Design Pattern permette di nascondere i dettagli di istanziazione di un oggetto favorendo la flessibilità del sistema rimuovendo dipendenze dei tipi permettendo ai client di operare sugli oggetti sfruttando le interfacce messe a disposizione.

Factory Method

Si è ritenuto opportuno ricorrere all'utilizzo del Factory Method nella realizzazione dei table model relativi alle GUI. Si è deciso di rilegare ad ogni table model, l'istanziazione di oggetti table model specifici basandosi su un'entità. Questo è stato realizzato definendo un metodo "Instantiate(Object)" nell'interfaccia "TableModel" in modo tale da "obbligare" ogni table model specifico a indicare il modo in cui i propri oggetti vengono creati.