Object-Oriented Software Design

a.a.2019-2020

AUTOSTRADE

NOME	COGNOME	MATRICOLA
Moretti	Thomas	247286
Piccirilli	Marzia	246972
Rampa	Valentina	246816

1. REQUISITI

L' obiettivo del nostro Sistema è quello di fornire un software capace di calcolare il pedaggio autostradale da un casello di partenza a un casello di destinazione a seconda della rete autostradale, dalla tipologia del percorso e della categoria del veicolo.

L'utente ha la possibilità di visualizzare il percorso effettuato o che vorrà effettuare e l'importo che dovrà saldare semplicemente inserendo la sua targa ed il percorso desiderato.

Il sistema deve, inoltre, fornire un'interfaccia grafica per l'utente e per l'amministratore, il quale dovrà gestire operazioni CRUD riguardanti caselli, autostrade e tariffe.

1.1. Requisiti funzionali

Utente:

- Obbligo di registrazione da parte dell'utente.

- Possibilità di aggiunta di uno o più veicoli a lui intestati. Conoscenza della tariffa complessiva del percorso dal casello di partenza al casello di destinazione secondo la norma vigente selezionata in precedenza dall'amministratore (Unitaria o Ambientale).

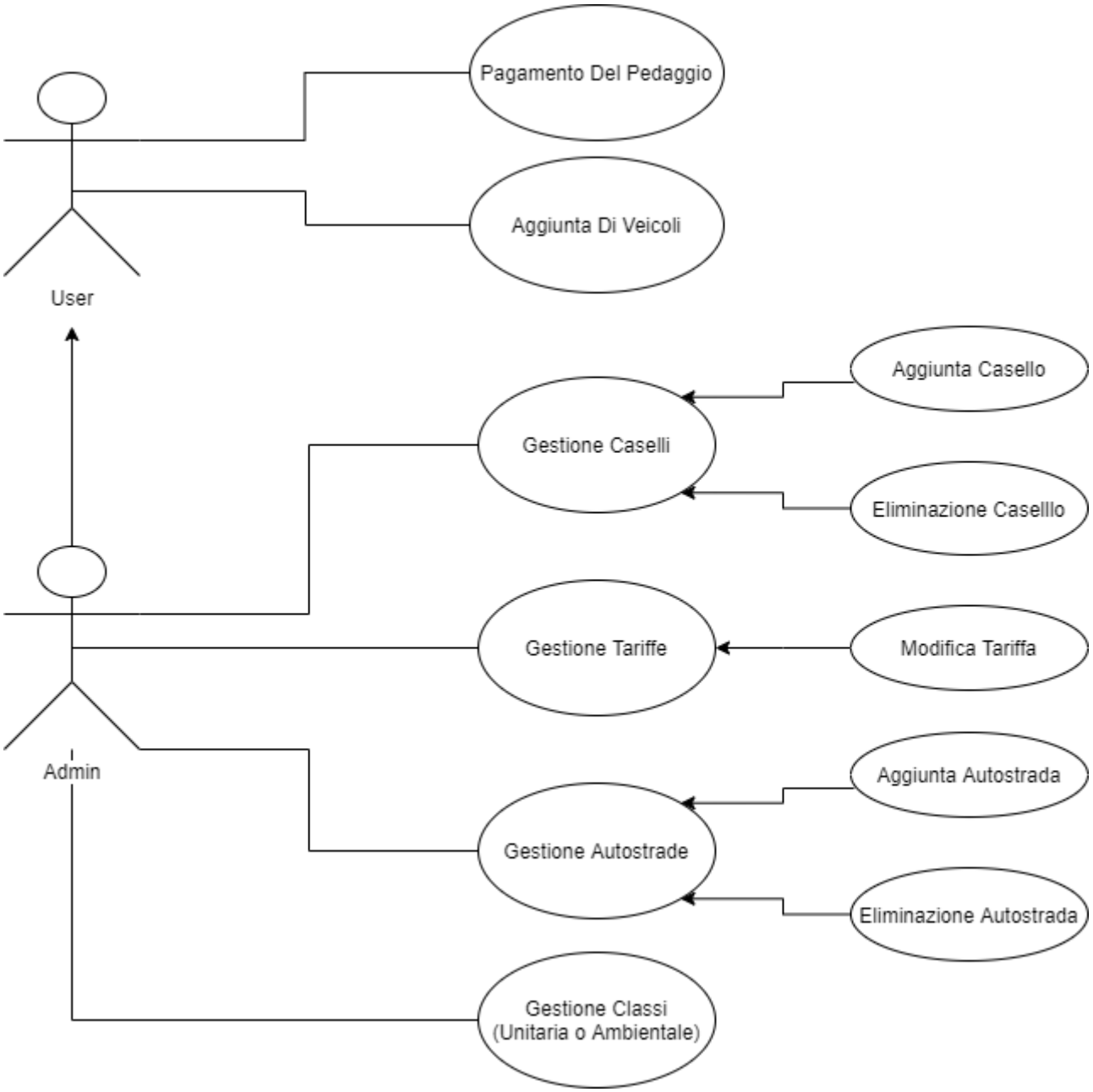
Amministratore:

- Gestione della classe per la scelta del calcolo del pedaggio come normativa vigente (tra Unitaria ed Ambientale)*.
- Gestione dei caselli e delle autostrade (tramite aggiunta ed eliminazione).
- Gestione dell'importo dell'IVA.
- Gestione degli importi delle classi ambientali dei veicoli (attraverso modifica della tariffa per ogni categoria).
- Possibilità di aggiunta di uno o più veicoli a lui intestati.
- Conoscenza della tariffa complessiva del percorso dal casello di partenza al casello di destinazione secondo la norma vigente selezionata in precedenza (Unitaria o Ambientale).
- Selezione della normativa Unitaria o Ambientale.

***** *Per quanto riguarda la classe Unitaria la tariffa complessiva del pedaggio è il risultato calcolato attraverso la distanza tra casello di partenza e casello di destinazione (espressa in chilometri) moltiplicata alla somma tra la tariffa unitaria del veicolo (quindi in base alla classe tariffaria corrispondente) e la tariffa della rete autostradale, applicandone poi l'importo IVA ed il rispettivo arrotondamento.*

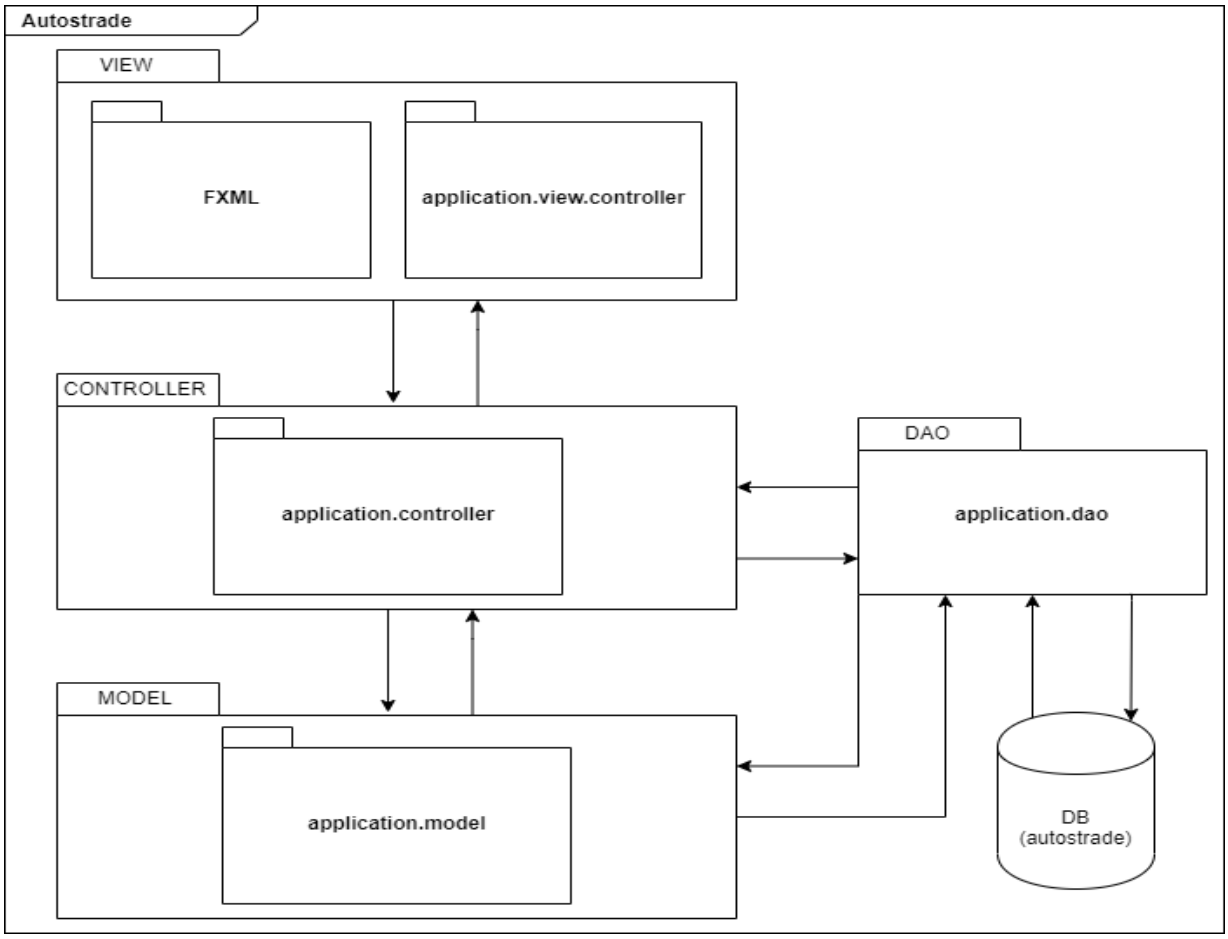
Mentre, per la classe Ambientale la tariffa complessiva del pedaggio è il risultato calcolato attraverso la distanza tra casello di partenza e casello di destinazione (espressa in chilometri) moltiplicata alla somma tra la tariffa unitaria del veicolo (quindi in base alla classe tariffaria corrispondente) e la tariffa ambientale di appartenenza del veicolo (Euro 1-6, che differirà per ogni categoria di veicolo in base all'inquinamento prodotto dallo stesso), applicandone poi l'importo IVA ed il rispettivo arrotondamento.

1.2. Use Case Diagram



2. ARCHITETTURA DEL SISTEMA

2.1 Modello Dell'Architettura Software (Package Diagram)



2.2. Descrizione Dell' Architettura

La nostra architettura prevede tre tipologie di livelli:

- Il primo livello 'Application', in cui vi è la classe "Main" della nostra applicazione;
- Il secondo livello, in cui vi sono .controller, .view , .dao e .model;
- Il .view prevede un terzo livello dove vi sono riportate le view in .fxml e i relativi .controller.

2.3. Strategie Adottate Per La Costruzione Del Sistema

Le strategia da noi utilizzate per la costruzione del sistema sono:

- Per le viste messe a disposizione per l'utente, collocate nel package "application.view.fxml" abbiamo utilizzato la tecnologia JavaFX attraverso il programma Scene Builder;

- Per la parte statica del sistema e quelli che sono l'input dell'utente abbiamo ritenuto utile adottare il pattern MVC (basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali model, view e controller);

- Per la persistenza dei dati abbiamo ritenuto consono l'utilizzo di una Database MYSQL.

2.4. Design Patterns

I principale Design Patterns utilizzati sono:

***MVC*:** Model-View-Controller (MVC) è un pattern utilizzato in programmazione per dividere il codice in blocchi dalle funzionalità ben distinte. Questo pattern architetturale è in grado di separare la logica di presentazione dei dati dalla logica di business (ovvero la logica di elaborazione che rende operativa l'applicazione).
Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali: Model, View, Controller.

- Il Model fornisce i metodi per accedere ai dati utili all'applicazione, ovvero si occupa solo dello stato e della logica (business logic) dell'applicazione.
- Il View visualizza i dati contenuti nel Model e si occupa dell'interazione tra l'infrastruttura del sistema e gli utenti.
- Il Controller riceve i comandi dell'utente (in genere attraverso il View) e li attua modificando lo stato degli altri due componenti.

DAO (Data Access Object): è un pattern architetturale per la gestione della persistenza.

Si tratta fondamentalmente di una classe con relativi metodi che rappresenta un' entità tabellare di un RDBMS, per stratificare e isolare l'accesso ad una tabella tramite query (poste all'interno dei metodi della classe) ovvero al data layer da parte della business logic creando un maggiore livello di astrazione ed una più facile manutenibilità.

Il vantaggio relativo all'uso del DAO è quindi il mantenimento di una rigida separazione tra le componenti di un'applicazione, le quali potrebbero essere il "Modello" e il "Controllo" in un'applicazione basata sul paradigma MVC

2.5. Obiettivi Di Design

Il software 'Autostrade' ha come principale obiettivo quello di garantire scalabilità, quindi la gestione di modifiche del numero di autostrade, di utenti e di amministratori.

Abbiamo scelto di impiegare un numero di dati sufficiente, per il funzionamento del nostro sistema, che prevede l'utilizzo di due normative (quali Unitaria e Ambientale), con la possibilità, da parte dell'amministratore, di un eventuale modifica.

3. PROGETTAZIONI DI CLASSI E INTERFACCE

3.1 Descrizione di Classi, interfacce e membri (Entity Relationship Diagram)

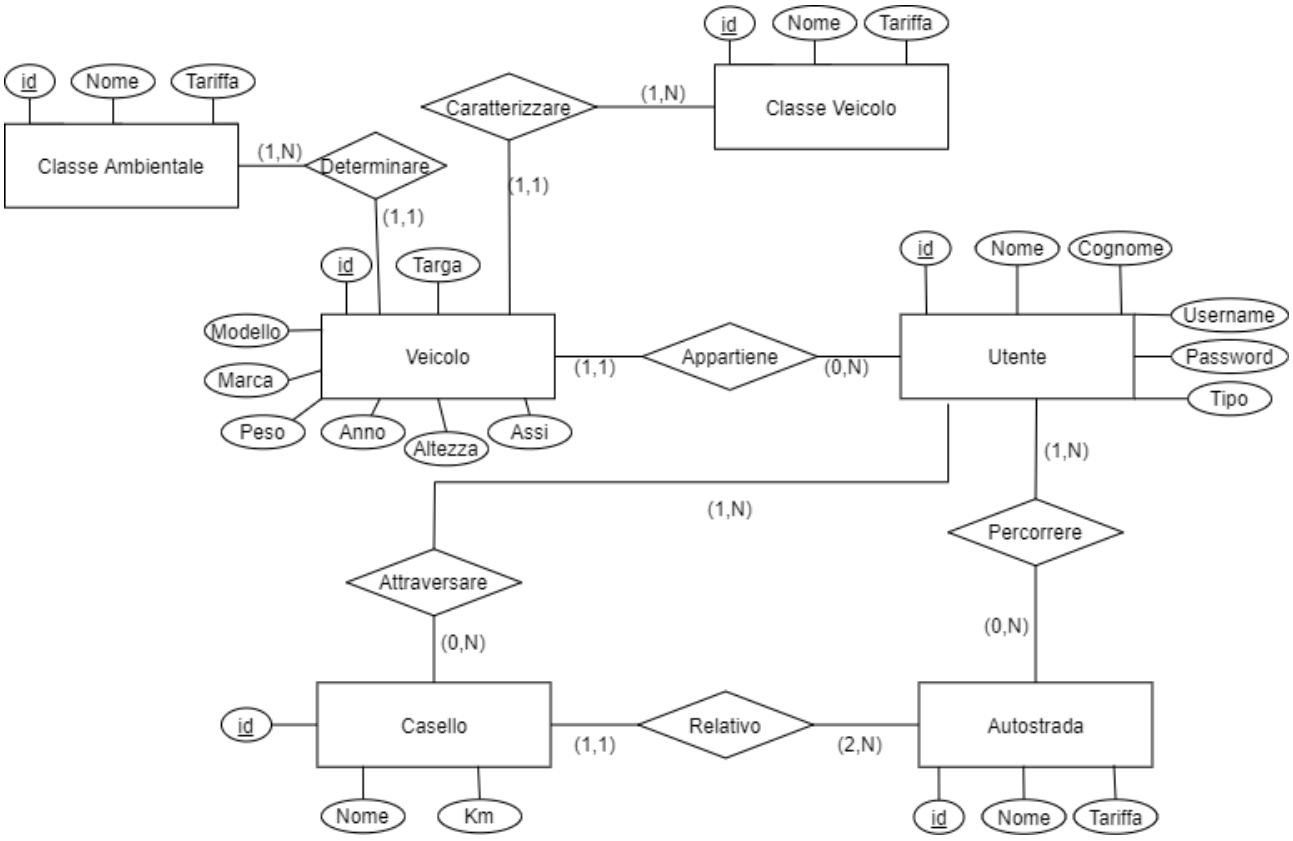


Tabelle:

- VEICOLO = {ID, ID_UTENTE, ID_CLASSE_VEICOLO, ID_CLASSE_AMBIENTALE, modello, marca, targa, peso, anno, altezza, assi}.
- UTENTE = {ID, nome, cognome, username, password, tipo}
- CASELLO = {ID, ID_AUTOSTRADA, nome, km}
- AUTOSTRADA = {ID, nome, tariffa}
- CLASSE VEICOLO = {ID, nome, tariffa}
- CLASSE AMBIENTALE = {ID, nome, tariffa}
- ATTRAVERSARE = {ID, ID_UTENTE, ID_CASELLO_PARTENZA, ID_CASELLO_ARRIVO}
- PERCORRERE = {ID, ID_UTENTE, ID_AUTOSTRADA}

Ogni **veicolo** è relativo ad una determinata **classe veicolo** (categoria che caratterizza la sua tariffa unitaria) e una **classe ambientale** (categoria che determina la sua classe di inquinamento). Esso, inoltre, deve appartenere ad uno specifico **utente**.

Ogni **utente** può percorrere più **autostrade** e attraversare diversi **caselli**.

Ogni **autostrada** ha almeno due **caselli** (uno di entrata e uno di uscita).

Le tabelle "**Attraversare**" e "**Percorrere**" sono due entità che si generano automaticamente dalle relazioni (1,n) e (1,n) delle rispettive entità "Utente"-"Casello" e "Utente"-"Autostrada".

Rappresentano uno storico del database degli utenti che attraversano determinati caselli e percorrono determinate autostrade.

Nella base di dati sarà presente un campo **gestione** che si occuperà di impostare la **classe** (il suo attributo nome) attraverso la quale calcolare il pedaggio:

Se nel database il suo nome è settato a "Classe Unitaria", allora il pedaggio sarà calcolato tenendo conto della tariffa unitaria del veicolo e, **ugualmente** per tutti i veicoli, della tariffa di quell'autostrada (attributo "tariffa" entità Autostrada).

Se invece, il suo nome è settato a "Classe Ambientale" allora il pedaggio sarà calcolato tenendo conto della tariffa unitaria del veicolo e della tariffa corrispondente alla sua classe ambientale **(in questo modo non viene applicata una tariffa autostradale uguale per tutti i veicoli, ma essa differirà in base al livello di inquinamento del veicolo**).

Nella gestione sarà possibile modificare anche l'**IVA** da applicare per il calcolo del pedaggio.

VEICOLO:

ATTRIBUTI	DESCRIZIONE
ID	Chiave primaria dell'entità
ID_UTENTE	Chiave esterna
ID_CLASSE_VEICOLO	Chiave esterna
ID_CLASSE_AMBIENTALE	Chiave esterna
MODELLO	Modello del veicolo
MARCA	Marca del veicolo
TARGA	Targa del veicolo (anche chiave)
PESO	Peso del veicolo
ANNO	Anno di immatricolazione del veicolo
ALTEZZA	Altezza del veicolo
ASSI	Numero assi del veicolo (1, 2, 3, 4, 5 o più)

CASELLO:

ATTRIBUTI	DESCRIZIONE
ID	Chiave primaria dell'entità
ID_AUTOSTRADA	Chiave esterna
NOME	Nome del casello
KM	Km di altezza del casello

ATTRAVERSARE:

ATTRIBUTI	DESCRIZIONE
ID	Chiave primaria dell'entità
ID_UTENTE	Chiave esterna
ID_CASELLO_PARTENZA	Chiave esterna
ID_CASELLO_ARRIVO	Chiave esterna

UTENTE:

ATTRIBUTI	DESCRIZIONE
ID	Chiave primaria dell'entità
NOME	Nome dell'utente
COGNOME	Cognome dell'utente
USERNAME	Username dell'utente (anche chiave)
TIPO	Amministratore o utente

AUTOSTRADA:

ATTRIBUTI	DESCRIZIONE
ID	Chiave primaria dell'entità
NOME	Nome dell'autostrada (anche chiave)
TARIFFA	Tariffa applicata all'autostrada

CLASSE AMBIENTALE:

ATTRIBUTI	DESCRIZIONE
ID	Chiave primaria dell'entità
NOME	Nome della classe ambientale
TARIFFA	Tariffa classe ambientale

PERCORRERE:

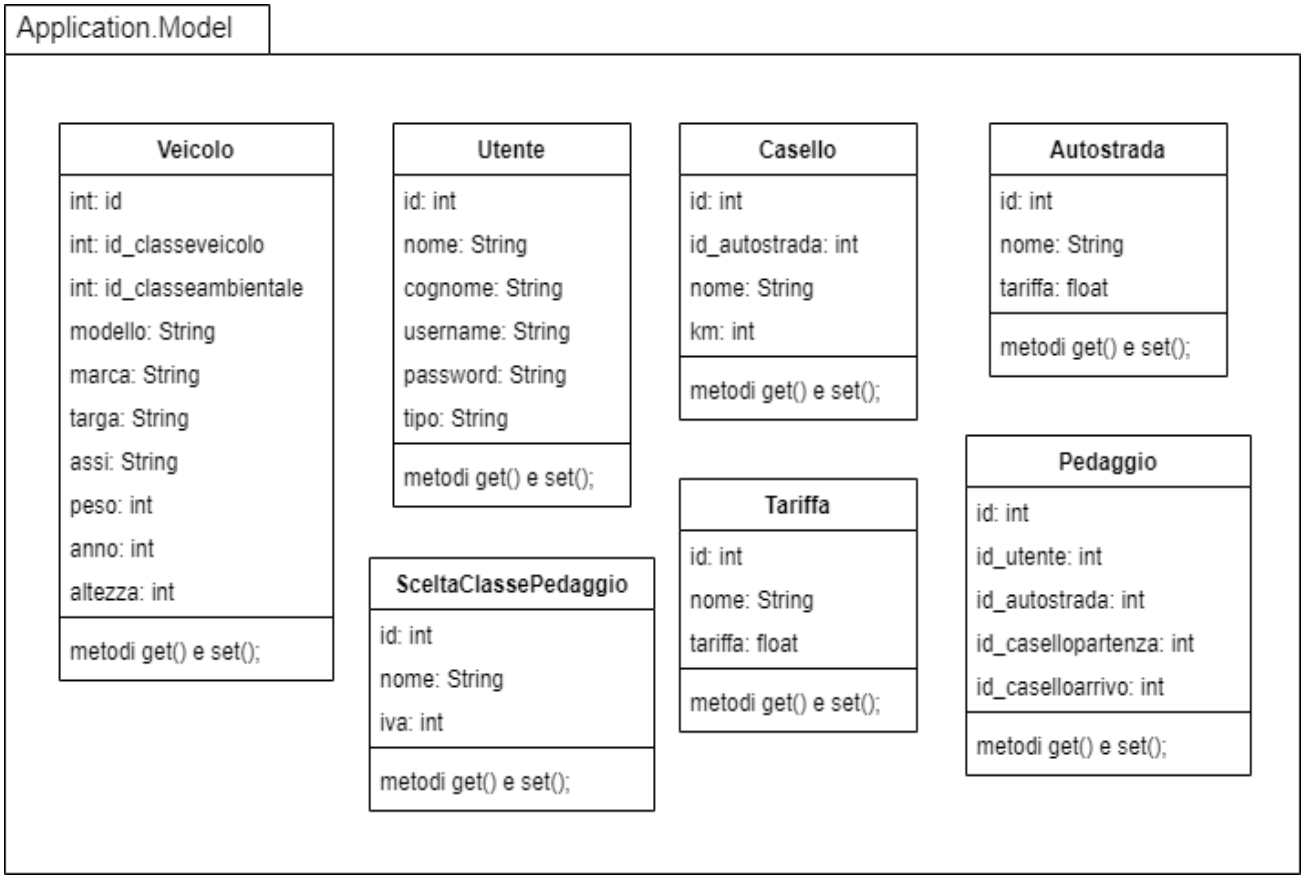
ATTRIBUTI	DESCRIZIONE
ID	Chiave primaria dell'entità
ID_UTENTE	Chiave esterna
ID_AUTOSTRADA	Chiave esterna

CLASSE VEICOLO:

ATTRIBUTI	DESCRIZIONE
ID	Chiave primaria dell'entità
NOME	Nome della classe del veicolo
TARIFFA	Tariffa unitaria

3.2. Descrizione Dettagli Di Design

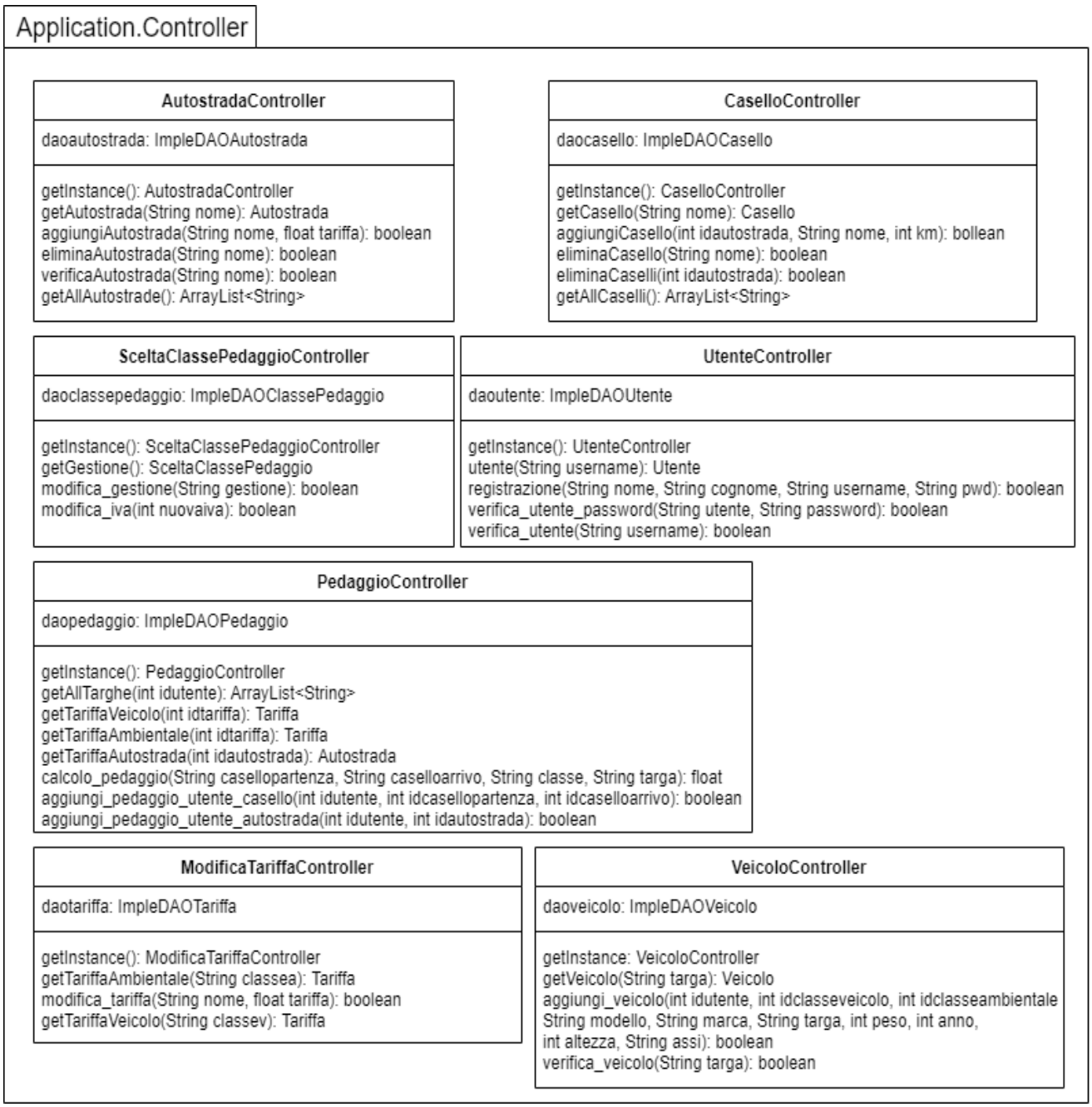
- Model Package



Rappresenta il Model nel pattern MVC.

Il package Application.Model si occupa di fare da tramite tra l'applicazione e il database sottostante, ovvero della parte logica del sistema; ma è importante precisare che il Model non contiene direttamente i dati**.**

- Controller Package

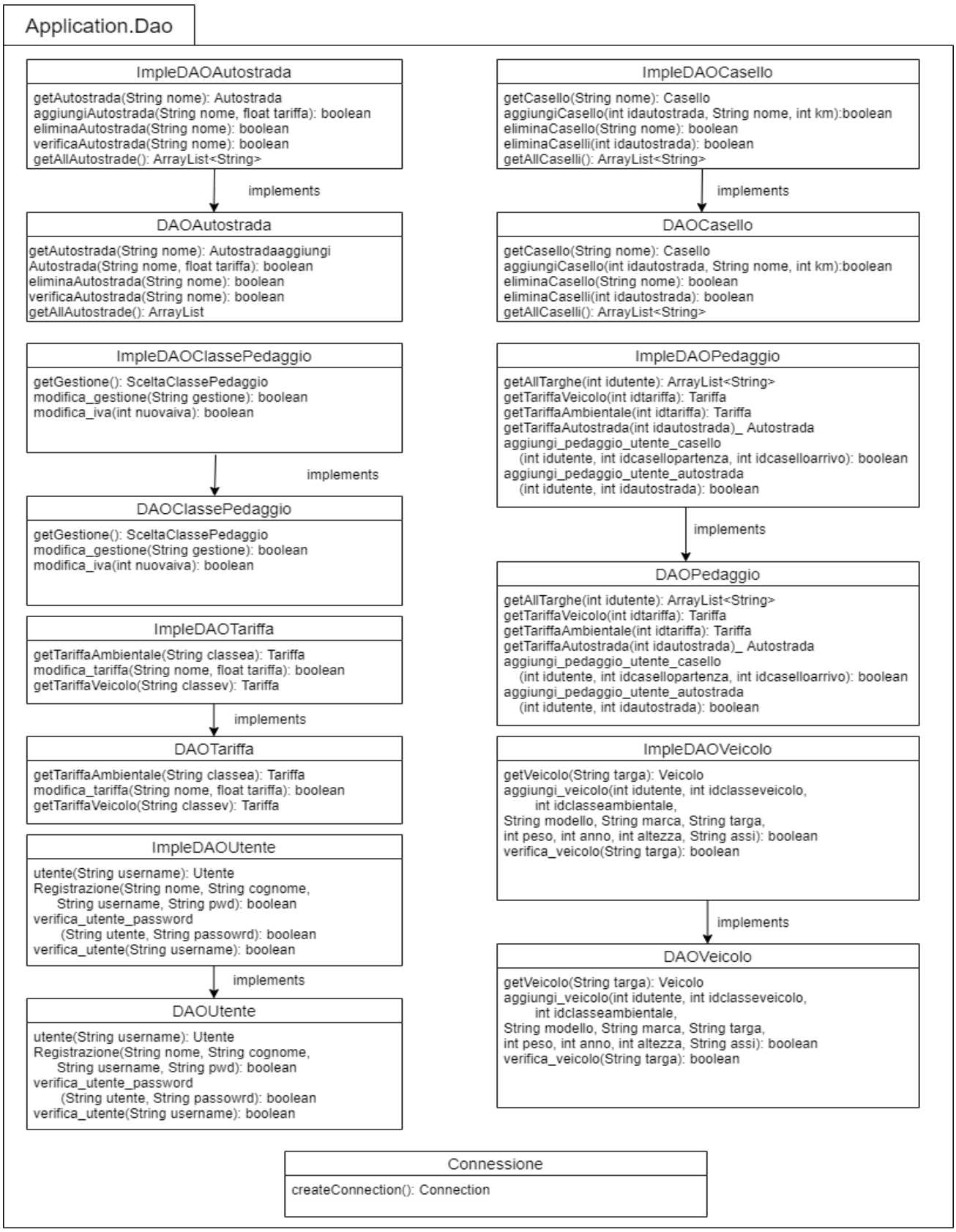


Rappresenta il Controller nel pattern MVC.

Il package Application.Controller ha la responsabilità di trasformare le interazioni dell'utente della View in azioni eseguite dal Model. Il Controller implementa la logica di controllo dell'applicazione.

Il package application.view.controller e application.view.fxml gestiscono le View di questo sistema, l'uno attraverso l'implementazione e l'altro attraverso la grafica dell'interfaccia (gestita in JavaFx).

- DAO Package



Rappresenta il pattern architetturale DAO (Data Access Object) utilizzato per isolare il package relativo alla persistenza dei dati.

Il vantaggio relativo all'uso del DAO è il mantenimento di una rigida separazione tra le componenti, le quali potrebbero essere il "Modello" e il "Controllo" in un'applicazione basata su MVC.