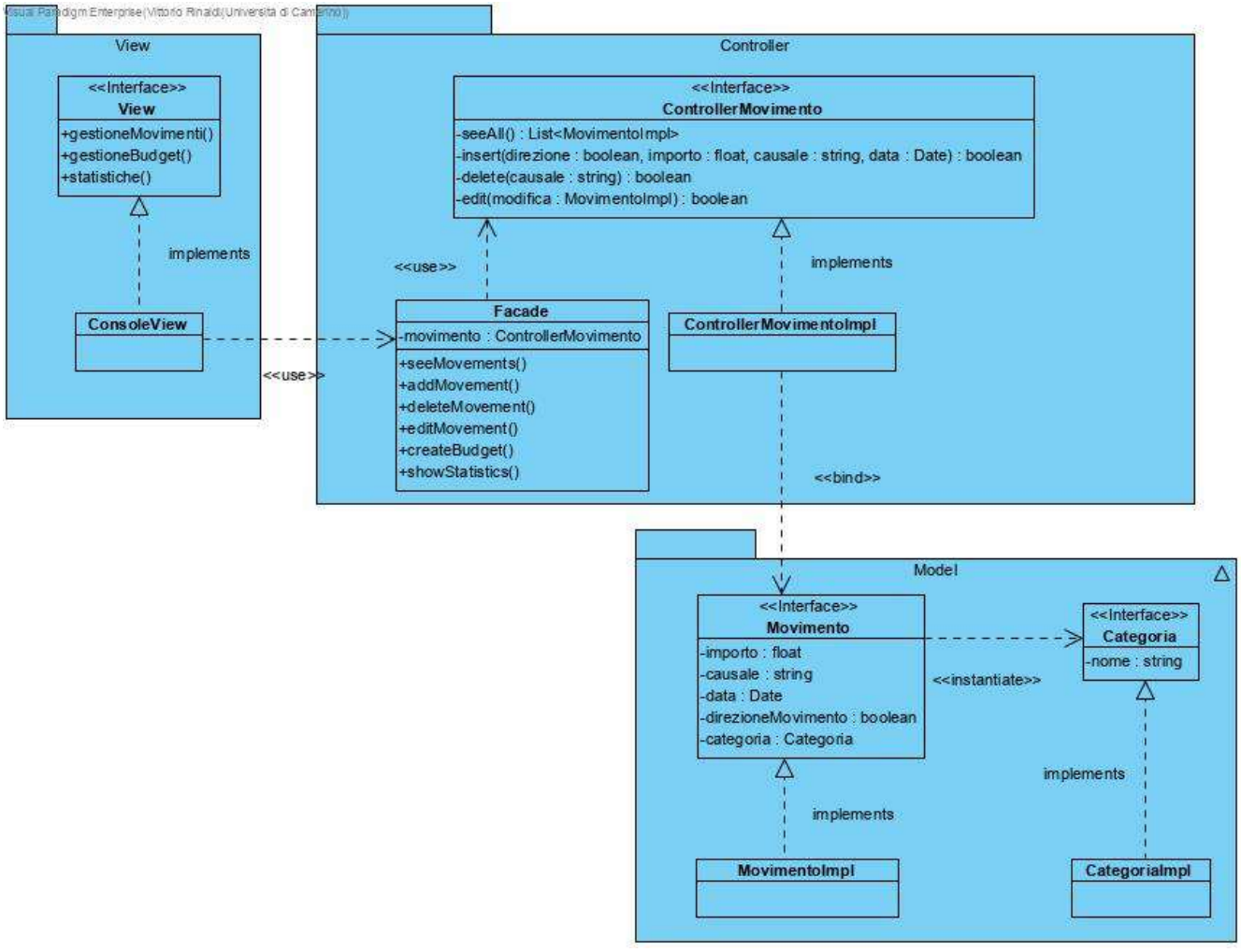
Programmazione Avanzata - Progetto Giugno/Luglio 2020

La progettazione del sistema è stata effettuata seguendo le linee guida dei pattern *MVC* e *FACADE*, l'obiettivo era ottenere classi con una responsabilità bilanciata, un codice pulito ed il rispetto dei principi SOLID della programmazione O/O.

Nonostante la "bontà" del *MVC*, la scelta di utilizzare altresì un pattern come *FACADE* deriva dalla volontà di sottolineare i principi di Responsabilità Singola e di Open-Closed: in questo modo, grazie ai "sotto-controller", permetteremo un'estendibilità più agevolata, dovuta alla stratificazione delle responsabilità di una classe ritenuta troppo protagonista quale era appunto il Controller.

La classe Facade avrà il compito di intercettare le chiamate del front-end (es: REST) e direzionarle ai controller più specifici, i quali conterranno l'implementazione delle suddette e si occuperanno finalmente di manipolare lo Stato del model assegnato, riducendo la responsabilità della prima classe.

Questo apre le porte ad una vasta dose di estendibilità: per esempio, nel caso in cui il cliente avesse bisogno di gestire in futuro i propri asset di investimenti o altri tipi di operazioni finanziarie basterà creare una nuova interfaccia (es: ControllerInvestimenti), collegarla al Facade ed alla corrispondente classe del Model, in questo modo ogni classe avrà una sola responsabilità.



Descrizione delle Classi

- View

Interfaccia che riassume le operazioni fondamentali che l'utente dovrà poter effettuare. Subisce i suoi ordini e li delega alla classe Facade, non curandosi della forma con cui questi verranno implementati.

In questa fase della Modellazione i suoi metodi coincidono con casi d'uso discussi con il cliente.

- ViewImpl

Classe che implementa i metodi dell'interfaccia "View".

- Facade

Anello di congiunzione tra front-end e back-end: intercetterà tutte le chiamate dell'interfaccia grafica indirizzandole al controller più opportuno; anche questa classe opera senza preoccuparsi di come avverrà l'implementazione.

- ControllerMovimento

Interfaccia che descrive il comportamento della classe Movimento con il diritto di manipolarne lo stato. Esegue per ordine del Facade.

Viene utilizzata per frazionare le responsabilità delle classi, in questo modo la modifica di una sola di loro non condizionerà l'intero codice.

- ControllerMovimentoImpl

Classe che implementa i metodi dell'interfaccia "ControllerMovimento".

- Movimento

Gestisce lo stato degli oggetti (transazioni di denaro). Solo il suo controller potrà accedere in scrittura a questa classe mentre la lettura sarà possibile anche lato View, per mostrare l'evoluzione delle informazioni (via metodi accessori).

- Categoria

Classe associata a Movimento: ogni istanza di Movimento sarà correlata ad uno o più oggetti Categoria.