




UNIVERSITÀ DEGLI STUDI DI NAPOLI  
**FEDERICO II**

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

CORSO DI LAUREA IN INFORMATICA  
INSEGNAMENTO DI OBJECT ORIENTATION

*Anno accademico 2023/2024*

Progettazione e sviluppo di un sistema O-O 

**Unina Delivery**

**Autori:**

Nome	Matricola	E-Mail
Gianluca Fiorentino	N86/4650	<a href="mailto:gianlu.fiorentino@studenti.unina.it">gianlu.fiorentino@studenti.unina.it</a>
Luigi Dota	N86/4718	<a href="mailto:lu.dota@studenti.unina.it">lu.dota@studenti.unina.it</a>

**Docente:**

Sergio Di Martino

# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Descrizione del Problema . . . . .	1
1.2	Analisi del Problema . . . . .	1
<b>2</b>	<b>Class Diagram</b>	<b>2</b>
2.1	Diagramma delle classi UML . . . . .	2
<b>3</b>	<b>Sequence Diagram</b>	<b>4</b>
3.1	Funzionalità di login . . . . .	4
3.2	Funzionalità di report . . . . .	5
<b>4</b>	<b>GitHub</b>	<b>6</b>

# 1 Introduzione

---

## 1.1 | Descrizione del Problema

Unina Delivery è un sistema per la gestione della logistica delle spedizioni di merci. Sulla base degli ordini dei clienti, l'operatore può, attraverso l'uso del sistema, pianificare le spedizioni, tenendo conto di fattori come la disponibilità della merce, il suo peso, la presenza di mezzi di trasporto e corrieri disponibili. Inoltre, può aggiornare la disponibilità dei prodotti in magazzino, filtrandoli opportunamente per nome e quantità. Infine, ha la possibilità di visualizzare un report statistico mensile selezionando il mese e l'anno di riferimento.

## 1.2 | Analisi del Problema

Nell'ambito del corso di Basi di Dati, è già stata progettata e sviluppata una base di dati dedicata alla descrizione, memorizzazione e gestione del problema.

Ora l'obiettivo è concentrarsi sulle funzionalità dell'applicativo e sulle modalità di sviluppo. Nello specifico, si è scelto di utilizzare il linguaggio **FXML** per la creazione delle interfacce grafiche, supportate da file di script **CSS** per la gestione dello stile degli elementi visibili a schermo. Per quanto riguarda il back-end, viene impiegato **JavaFX** con un'architettura **BCE** (boundary-control-entity) oriented.

Diamo adesso uno sguardo alla modellazione del problema con il *Class Diagram* e due *Sequence Diagram*.

## 2 Class Diagram

---

### 2.1 Diagramma delle classi UML

Al fine di una lettura quantomeno apprezzabile del class diagram, abbiamo preferito riportarlo alla pagina seguente, in modo da poter dedicargli quanto più spazio possibile, senza la necessità di doverlo spezzettare in più pagine. Inoltre, gli si è dato un minimo di colore in modo tale da distinguere a colpo d'occhio i vari *packages*, o pacchetti che dir si voglia (una soluzione simile verrà presa anche per i sequence diagram). Si ricorda, inoltre, che, per quanto la soluzione non sia printer-friendly, è possibile zoomare quanto si vuole e la qualità dell'immagine resterà la stessa, potendo dunque leggere senza alcun problema gli attributi e i metodi di ogni classe.

**Nota:** si fa presente che, per semplici questioni di leggibilità, sono state evitate le frecce <uses> dalle classi Control alle classi DAO. Si chiede al lettore, dunque, di immaginare che per ogni freccia Control -> DTO, ci sia una freccia corrispondente da Control a DAOimp.

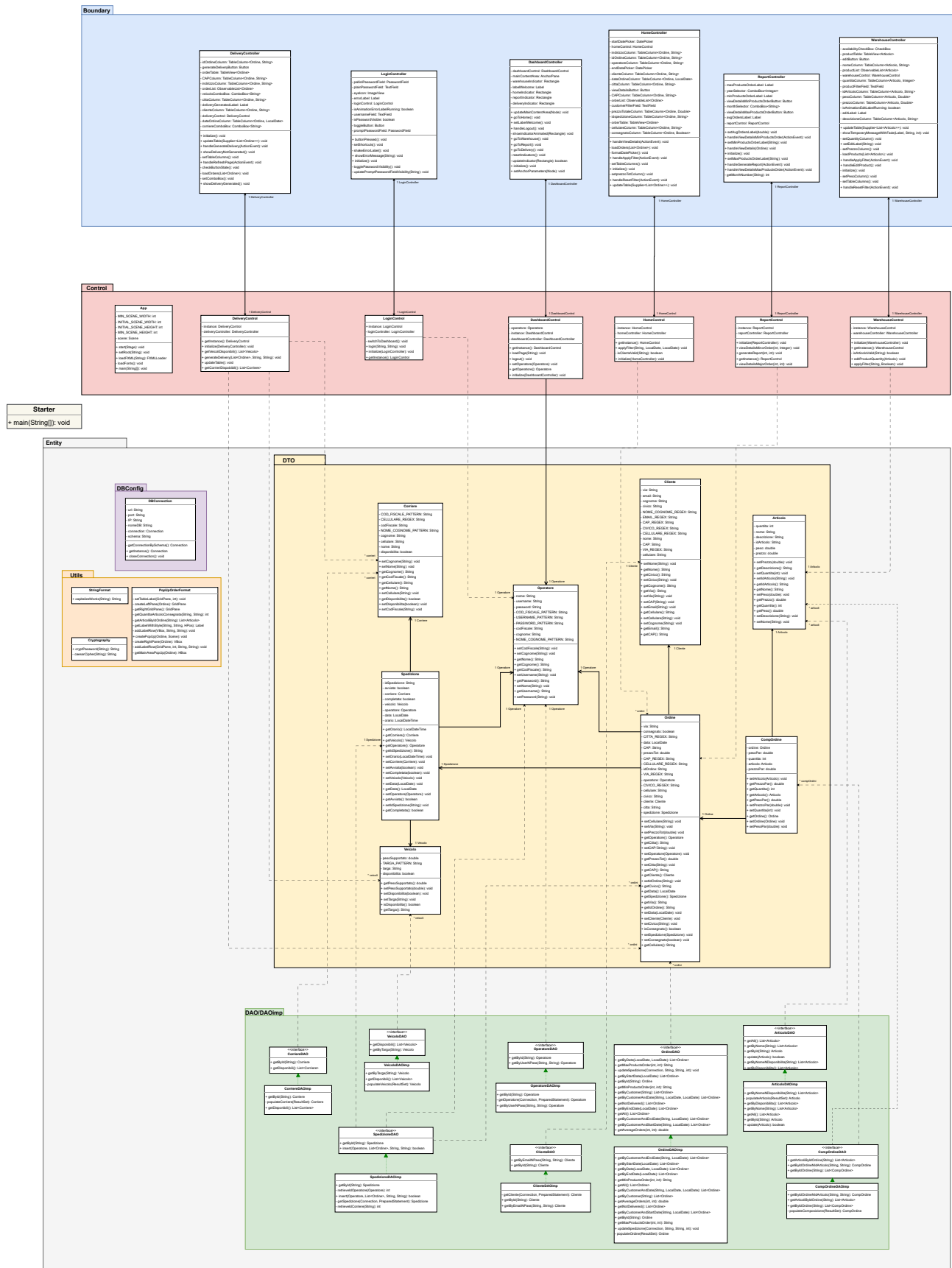


Figura 1: UML Class Diagram

## 3 Sequence Diagram

### 3.1 Funzionalità di login

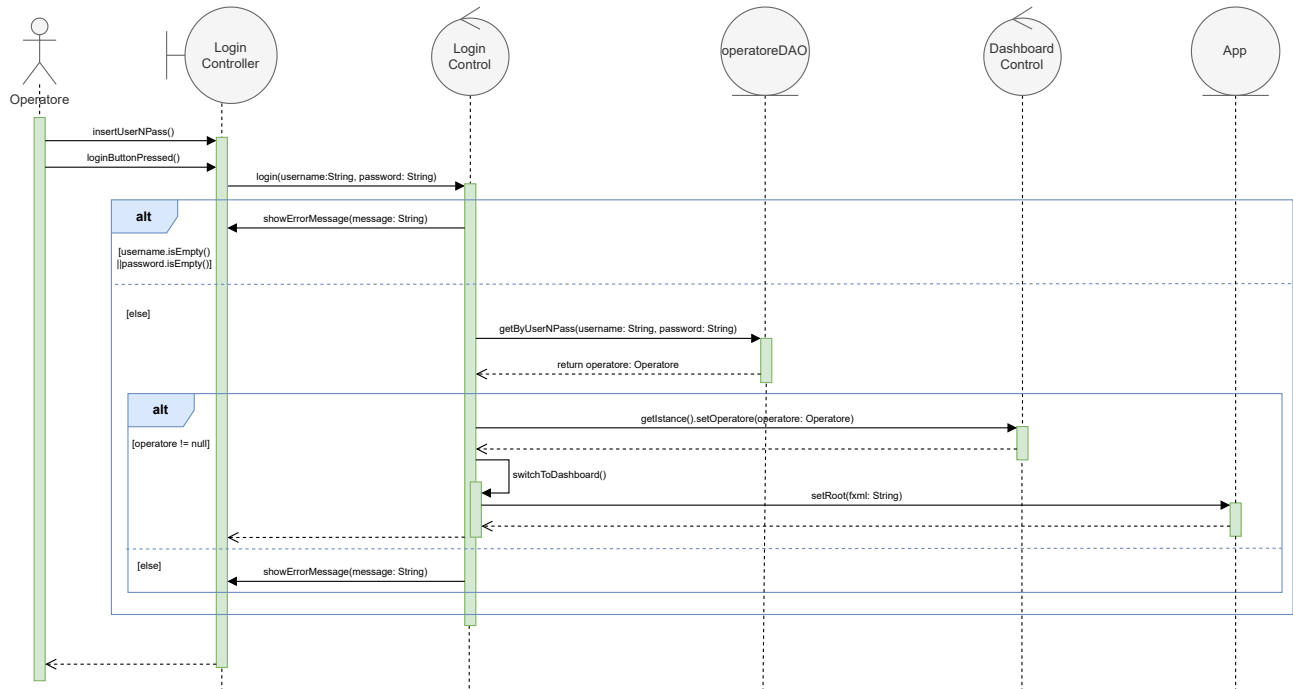


Figura 2: Login Sequence Diagram

### 3.2 Funzionalità di report

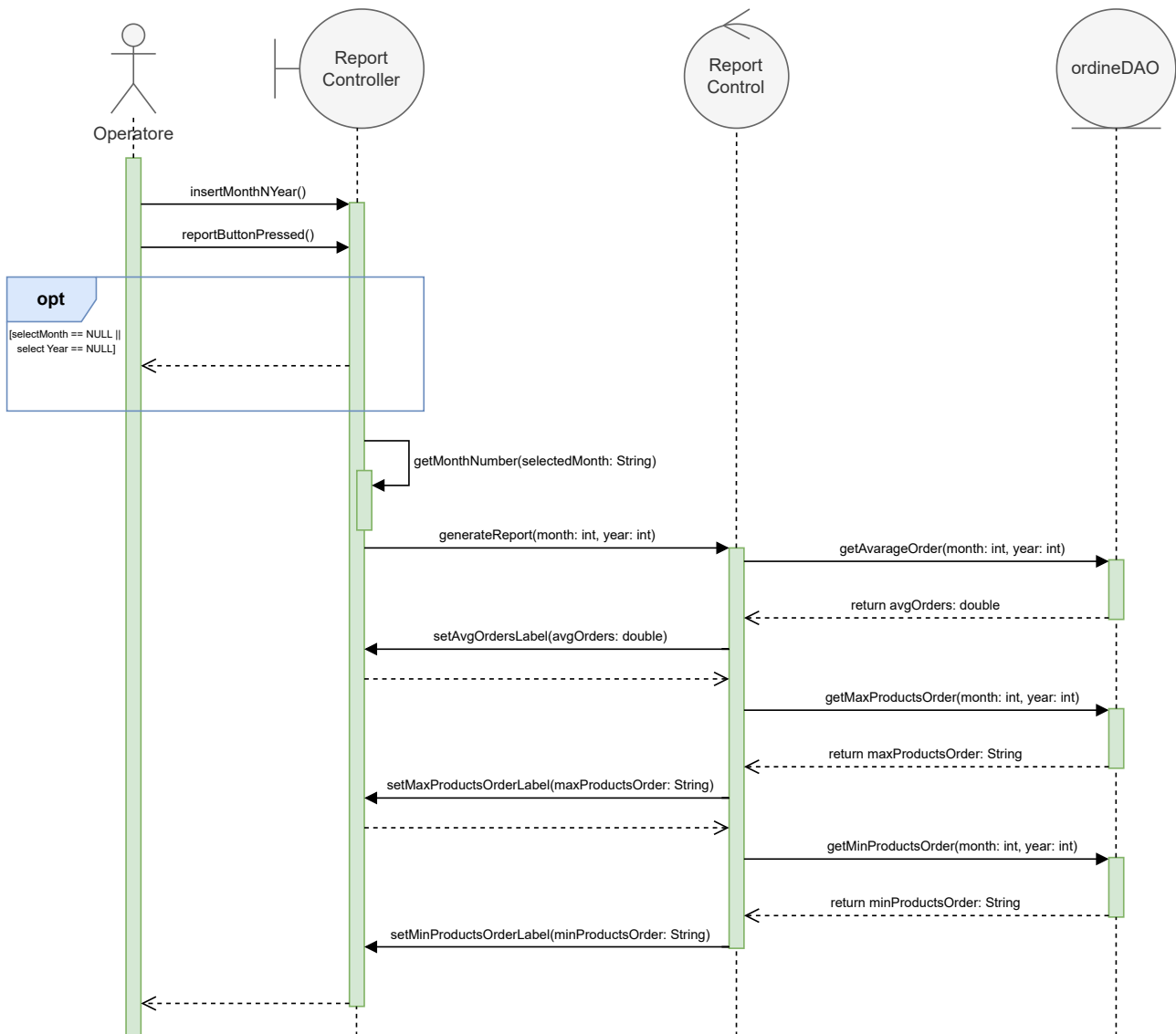


Figura 3: Report Sequence Diagram

## 4 GitHub

---

Per qualsiasi ulteriore riferimento al progetto e alla sua creazione e fase evolutiva, è possibile cliccare [qui](#) per essere indirizzati alla relativa repository GitHub, o in alternativa digitare: `https://github.com/Gianluca-F/OOBD_Project` (si presti attenzione all'underscore).