

Università degli Studi di Napoli

Parthenope

Progetto di Programmazione III

OrderApp

Relazione

Biancarosa Pasquale (2098)  
Militerno Eugenio (2454)  
Ruotolo Pasquale (2322)

Indice

[**1.1 Requisiti 3**](#_rbt3c5d191uj)

[**1.2 Analisi dei requisiti 3**](#_99udo5qx18g7)

[**1.3 Diagramma del Database 4**](#_1xewie379w7e)

[**1.4 Diagramma delle classi 5**](#_wp9s9oqisv0r)

[**1.5 Pattern utilizzati 7**](#_4mkh6ssgp3qb)

[1.5.1 - Facade 7](#_rid4raq7yjjj)

[1.5.2 - Strategy 8](#_xnvcpkvjs02c)

[1.5.3 - DAO (Data Access Object) 9](#_84ysgfllxn66)

[1.5.3.1 Model 9](#_jkorj9zfsnkt)

[1.5.3.2 DAO 10](#_7l5lw9qaiw4v)

[1.5.3.3 Interfaccia 11](#_1dr6elnj7x39)

[1.5.4 - Factory 12](#_nc3eptgu3kxg)

[1.5.5 - Observer 12](#_h399vj94bccy)

[1.5.6 - Decorator 13](#_jmfgxqwpa0u1)

[**1.6 - Sistema di automazione 14**](#_8tomd5hrj8tb)

**Progetto programmazione III e Laboratorio**

# **1.1 Requisiti**

Si vuole sviluppare un software che deve gestire il sistema di ordini in un ristorante.

L’accesso al software può avvenire in modalità Admin o in modalità Utente, per la gestione e l’aggiunta di prodotti e categorie ordinabili.

Gli ordini possono essere annullati (eliminati), e si può anche modificare un piatto.

Il cliente seduto al tavolo, identificato da un numero univoco, ordina un piatto, che gli verrà consegnato simulando i tempi della cucina. Il cameriere, una volta consegnato l’ordine, segnala che è stato portato al tavolo. Il cliente, una volta consumato l’ordine, procede in cassa dall’admin dove potrà pagare, scegliendo fra carta, contanti e bancomat. Una volta pagato, esce dal ristorante liberando il tavolo.

# **1.2 Analisi dei requisiti**

Il sistema deve prevedere l’accesso sia in modalitá amministratore che in modalitá utente.

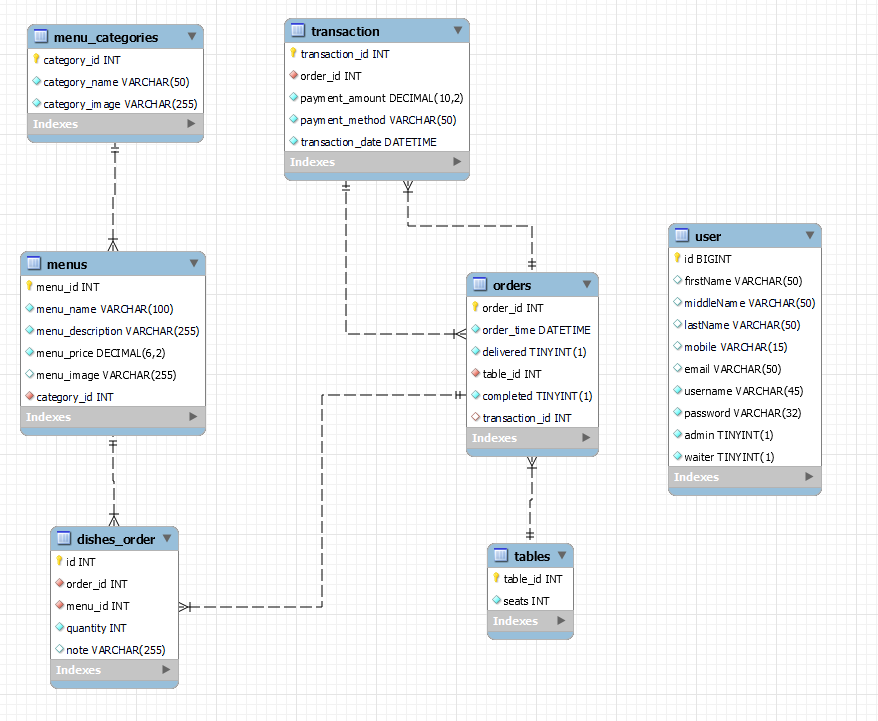
L’amministratore puó effettuare le seguenti operazioni:

* Aggiungere o modificare piatti
* Visualizzare gli utenti registrati
* Visualizzare gli ordini pagati
* Permettere il pagamento degli ordini

Il Cameriere puó effettuare le seguenti operazioni:

* Aggiungere un ordine per un tavolo
* Rimuovere un ordine o un piatto dall’ordine
* Consegnare l’ordine al tavolo

# **1.3 Diagramma del Database**

****

Il database memorizza i dati dell’utente nella tabella User, e dopo il log in ne avvia la sessione.

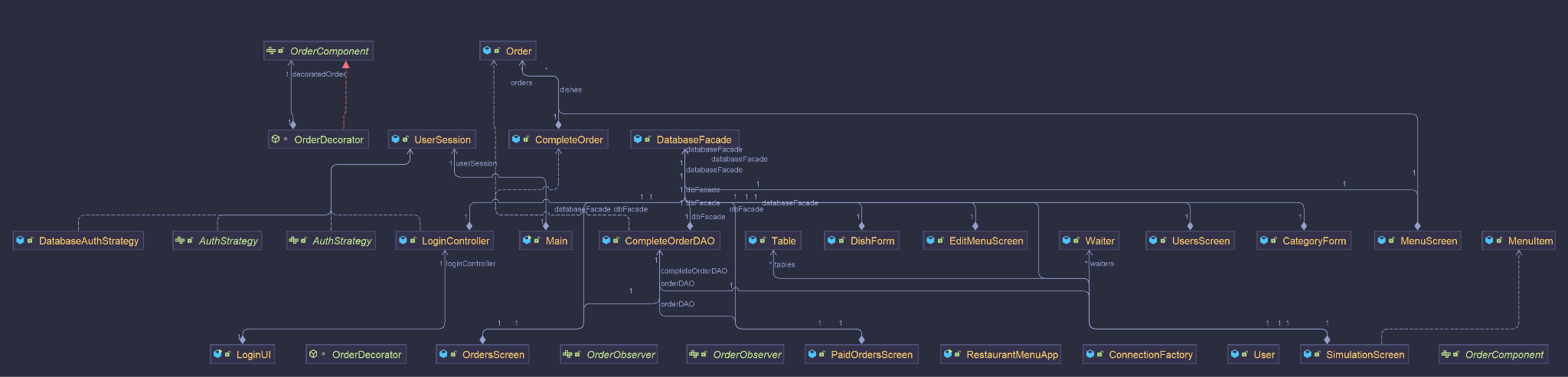
Tramite la flag “admin” e “waiter” è possibile differenziare gli utenti camerieri dall’admin per dargli accesso a funzioni dedicate.

Il database memorizza i menù, suddivisi in categorie, e i vari piatti, tiene traccia degli ordini effettuati e delle transazioni eseguite.

Tabelle:

* Tramite la tabella **User** vengono memorizzati i dati degli utenti del Db. Ne si protegge l’accesso tramite email e password, e grazie alle flag admin e waiter è possibile avviare una sessione con i giusti livelli di permessi.
* Tramite le tabelle **Tables, Orders, Transaction, Dishes\_order** è possibile memorizzare gli ordini associati ai tavoli con il corrispettivo stato e le transazioni associate
* Tramite le tabelle **Menu, Menu\_categories** vengono memorizzate i vari piatti e categorie che poi vengono associate agli ordini effettuati

# **1.4 Diagramma delle classi**

****

Le funzioni delle classi sono le seguenti:

* AuthStrategy : Interfaccia che gestisce la scelta del Login (Waiter o Admin)
* CompleteOrder : Gestisce l'ordine ed il suo completamento (Consegna e Pagamento)
* MenuItem : Record che gestisce il piatto (nome,id,prezzo)
* Order : Gestisce il singolo ordine
* OrderComponent : Interfaccia che permette di aggiungere note e modificarne la quantità
* OrderDecorator : Gestisce la modifica dell'ordine effettiva
* OrderObserver : Interfaccia che permette di notificare i cambiamenti di stato dell'ordine
* Table : Gestisce il tavolo
* User : Gestisce l'utente
* UserSession : Gestisce la sessione d'accesso al DB
* Waiter : Record che gestisce il waiter
* CategoryForm : Gestisce il menù dei piatti, è un form a tendina che può essere modificato
* CompleteOrderDAO : Connette ordini DB con ordini App
* ConnectionFactory : Gestisce la connessione al DB
* DatabaseAuthStrategy : Gestisce la scelta dell'autenticazione al DB (autenticazione Waiter/Admin)
* DatabaseFacade : Gestisce la connessione al DB per eseguire statement ed ottenere risultati
* DishForm : Gestisce il form del singolo piatto (il piatto dentro la categoria)
* EditMenuScreen : Gestisce le modifiche che vengono fatte all'UI del menu, è UI
* LoginController : Gestisce l'implementazione e la logica del Login
* LoginUI : Gestisce l'UI del Login
* Main : Classe principale dell'app
* MenuScreen : Implementa la UI del menù, prezzi di piatti ecc
* OrdersScreen : Gestisce la schermata degli ordini, permette notifiche su cambiamenti di stato degli ordini
* PaidOrdersScreen : Gestisce gli ordini PAGATI (completati)
* RestaurantMenuApp : La classe centrale dell'app
* SimulationScreen : Gestisce la UI della schermata di simulazione
* UserScreen : Gestisce la UI della schermata dell'utente dell'app

# **1.5 Pattern utilizzati**

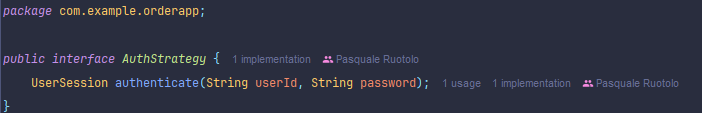
## **1.5.1 - Facade**

Questo pattern fornisce un'interfaccia semplificata per un insieme di interfacce di un sottosistema complesso, rendendo più facile l'utilizzo di quel sottosistema. L'idea principale è di nascondere la complessità del sistema e fornire una singola interfaccia per eseguire operazioni diverse.



## **1.5.2 - Strategy**

Questo pattern consente di scegliere l'algoritmo da utilizzare in fase di esecuzione, senza modificare il contesto in cui l'algoritmo è utilizzato. È utile quando ci sono più varianti di un algoritmo e si vuole evitare un codice condizionale complesso.



Tramite la classe **DatabaseAuthStrategy** una volta inserite le credenziali di accesso la sessione è gestita in tempo reale per garantire l’accesso alle funzioni admin.

## **1.5.3 - DAO (Data Access Object)**

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 MySql ad un altro database per esempio, non sarà necessario stravolgere il codice della nostra applicazione, ma basterà modificare i DAO utilizzati.

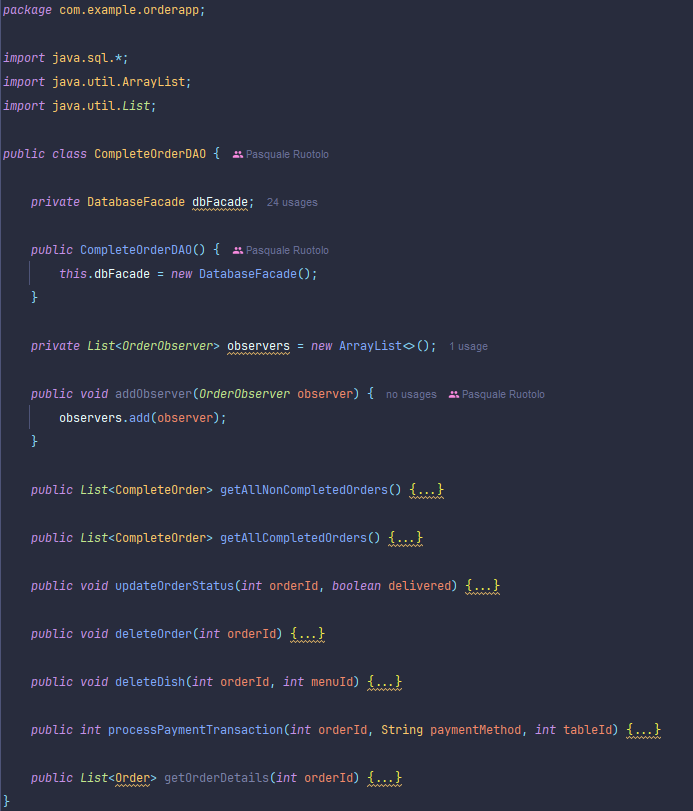
I concetti principali sono:

* la classe CompleteOrder ovvero il modello per gli ordini completati
* un’interfaccia (detta DAO) per ogni tabella contenente tutti i metodi CRUD relativi a quella tabella.
* l’interfaccia

### **1.5.3.1 Model**



### **1.5.3.2 DAO**

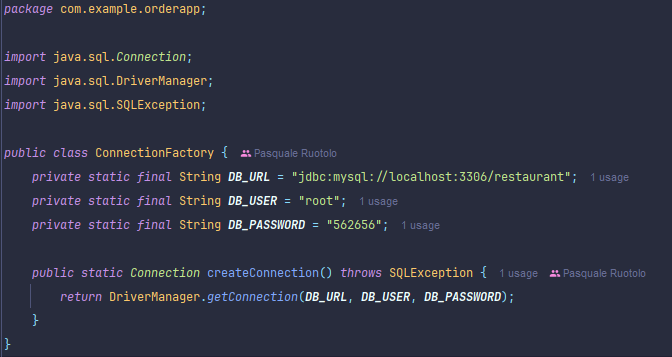
****

### **1.5.3.3 Interfaccia**



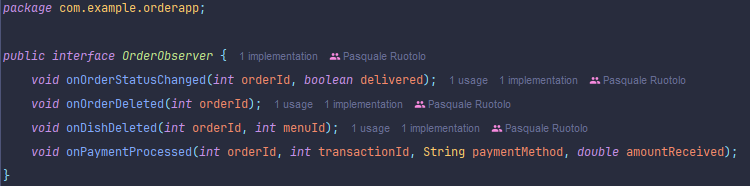
## **1.5.4 - Factory**

Il factory method pattern descrive un approccio di programmazione con il quale poter **creare oggetti senza bisogno di dover specificare la loro classe**. Questo permette di cambiare comodamente e in maniera flessibile l’oggetto creato. Lo sviluppatore sceglie se specificare il factory method in un’interfaccia e quindi implementarlo come classe figlio o come classe base ed eventualmente sovrascrivere dalle classi derivate. Questo metodo **opera a livello della classe costruttore standard** per separare la costruzione degli oggetti dagli oggetti stessi e permettere così l’utilizzo dei principi SOLID.



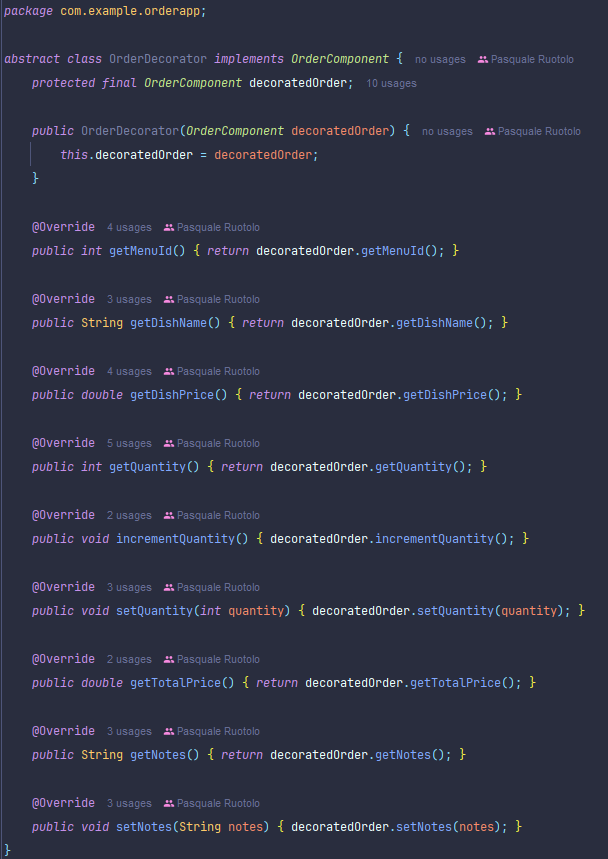
## **1.5.5 - Observer**

Questo pattern è utilizzato per implementare un meccanismo di notifica che consente agli oggetti di osservare e reagire ai cambiamenti di stato di un altro oggetto.

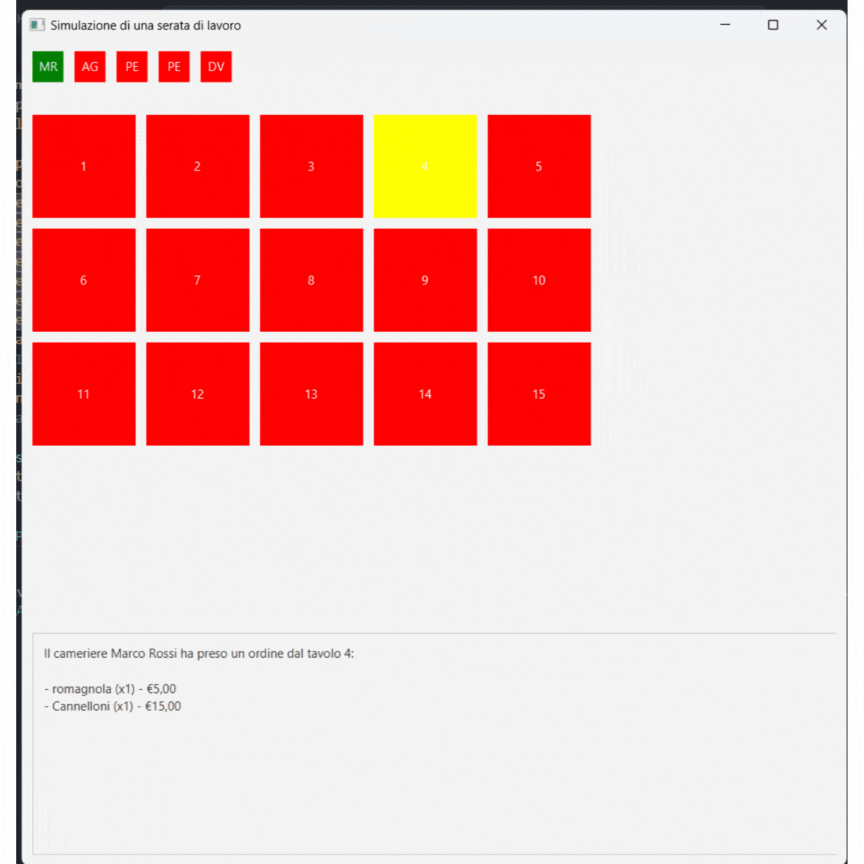
****

## **1.5.6 - Decorator**

Questo pattern è utile per aderire al principio di responsabilità singola, evitando la proliferazione di sottoclassi attraverso l'uso della composizione anziché dell'ereditarietà. (serve per inserire note negli ordini)



# **1.6 - Sistema di automazione**

****

Grazie a questa simulazione possiamo testare il corretto funzionamente del software

* I tavoli rossi sono quelli liberi
* I tavoli gialli sono quelli che hanno ordinato e stanno aspettando l’ordine
* I tavoli verdi sono quelli che hanno consumato l’ordine, e stanno pagando. Dopo di ciò, il tavolo tornerà rosso.