Gruppo 13

Progetto:

Paninaro Next

Titolo del documento:

Documento di Architettura

INDICE

Scopo del documento

Il presente documento riporta la definizione dell'architettura del progetto Paninaro Next usando diagrammi delle classi in Unified Modeling Language (UML) e codice in Object Constraint Language (OCL). Nel precedente documento è stato presentato il diagramma degli use case, il diagramma di contesto e quello dei componenti. Ora, tenendo conto

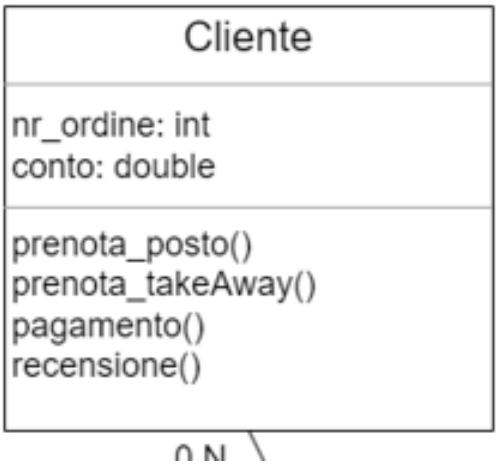
di questa progettazione, viene definita l'architettura del sistema dettagliando da un lato le classi che dovranno essere implementate a livello di codice e dall'altro la logica che regola il comportamento del software. Le classi vengono rappresentate tramite un digramma delle classi in linguaggio UML. La logica viene descritta in OCL perché tali concetti non sono esprimibili in nessun altro modo formale nel contesto di UML.

1. Diagramma delle classi

Nel presente capitolo vengono presentate le classi previste nell'ambito del progetto Paninaro Next. Ogni componente presente nel diagramma dei componenti diventa una o più classi. Tutte le classi individuate sono caratterizzate da un nome, una lista di attributi che identificano i dati gestiti dalla classe e una lista di metodi che definiscono le operazioni previste all'interno della classe. Ogni classe può essere anche associata ad altre classi e, tramite questa associazione, è possibile fornire informazioni su come le classi si relazionano tra loro.

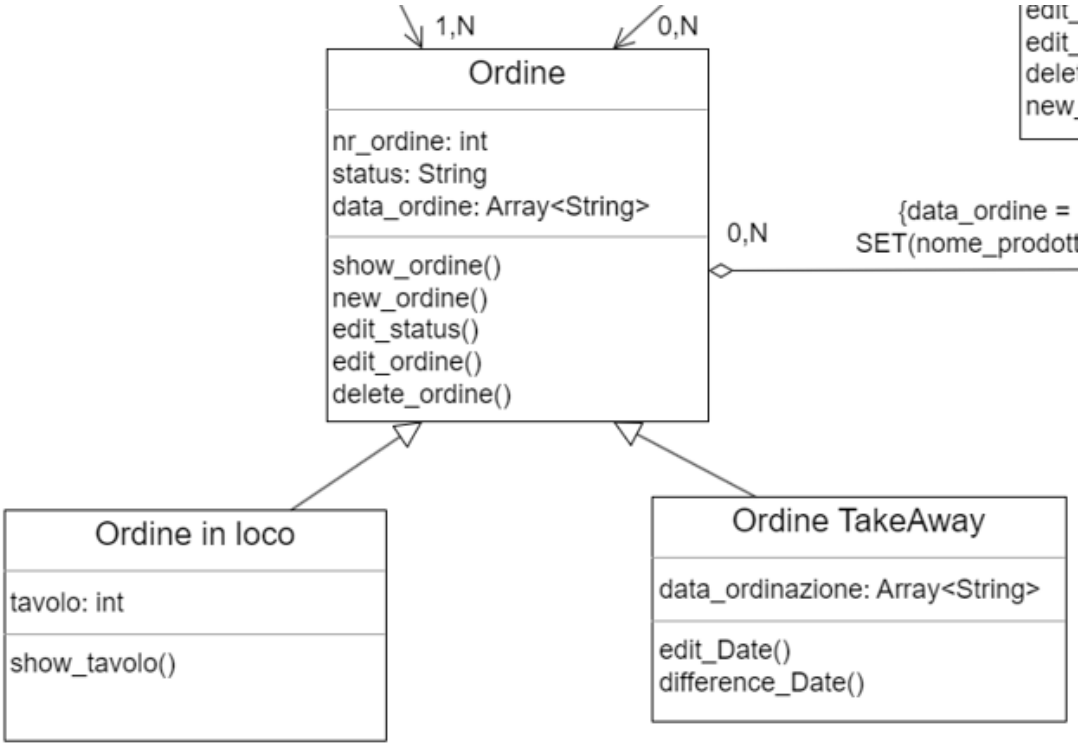
Riportiamo di seguito le classi individuate a partire dai diagrammi di contesto e dei componenti. In questo processo si è proceduto anche nel massimizzare la coesione e minimizzare l'accoppiamento tra classi.

1.1. Utente: cliente



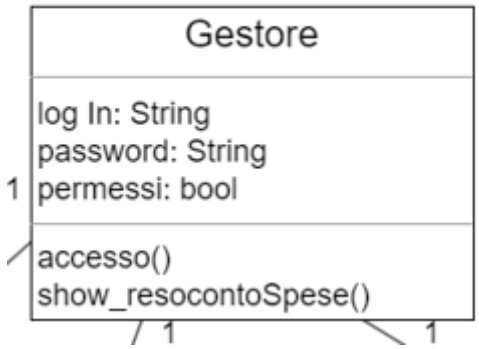
Analizzando il diagramma si nota la classe "Cliente", che è colui che fa un ordine per un paitto al Paninaro Next, al tavolo oppure takeaway.Questa classe è caratterizzata da 2 attributi: nr_ordine (un int che funge da identificatore univoco) e conto (il totale che il cliente deve pagare). Cliente inoltre possiede 4 metodi: uno per prenotare un posto al tavolo se consuma l'ordine in loco, uno per prenotare l'ordine da portare via, uno per pagare e uno per lasciare una recensione (da 0 a 5 stelle).

1.2. Ordini



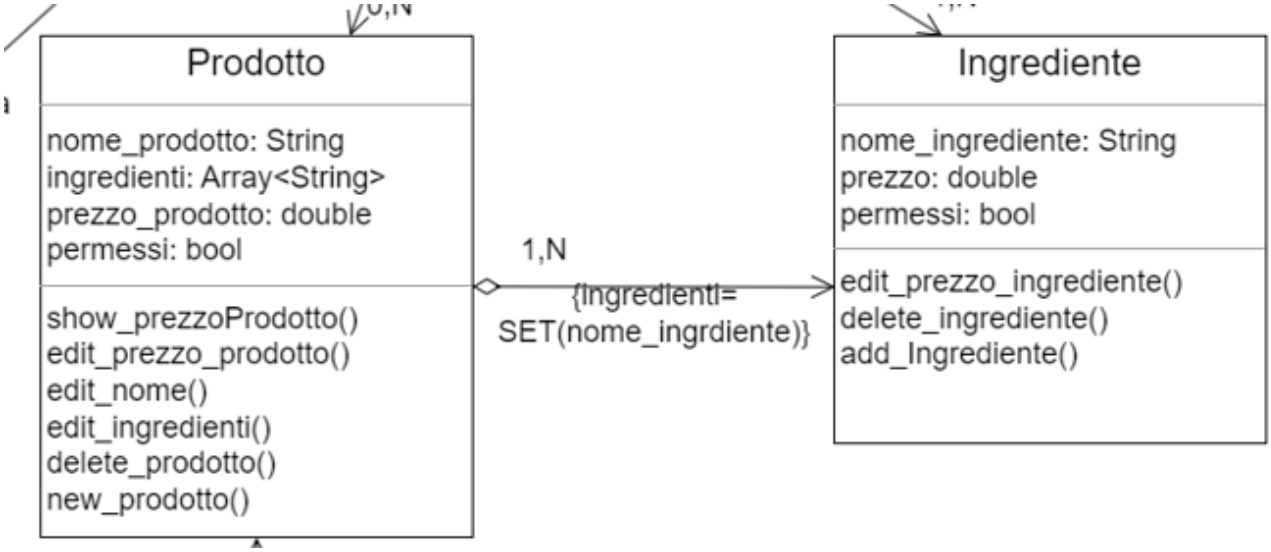
Successivamente troviamo una classe "Ordine", che si distingue in 2 sottoclassi disgiunte: "Ordine in loco" e "Ordine TakeAway". La superclasse ha i seguenti attributi: il numero dell'ordine, il suo status e la data dell'ordinazione. Ha poi questi metodi: uno per mostrare i dati dell'ordine, due per modificarlo e uno per cancellarlo. Alla sottoclasse In Loco viene poi aggiunto un attributo per segnare il tavolo corrispondente all'ordine; invece alla classe TakeAway viene aggiunta la data completa dell'ora del ritiro da parte del cliente dell'ordine, e dei metodi per visualizzarla e modificarla.

1.3. Utente: gestore



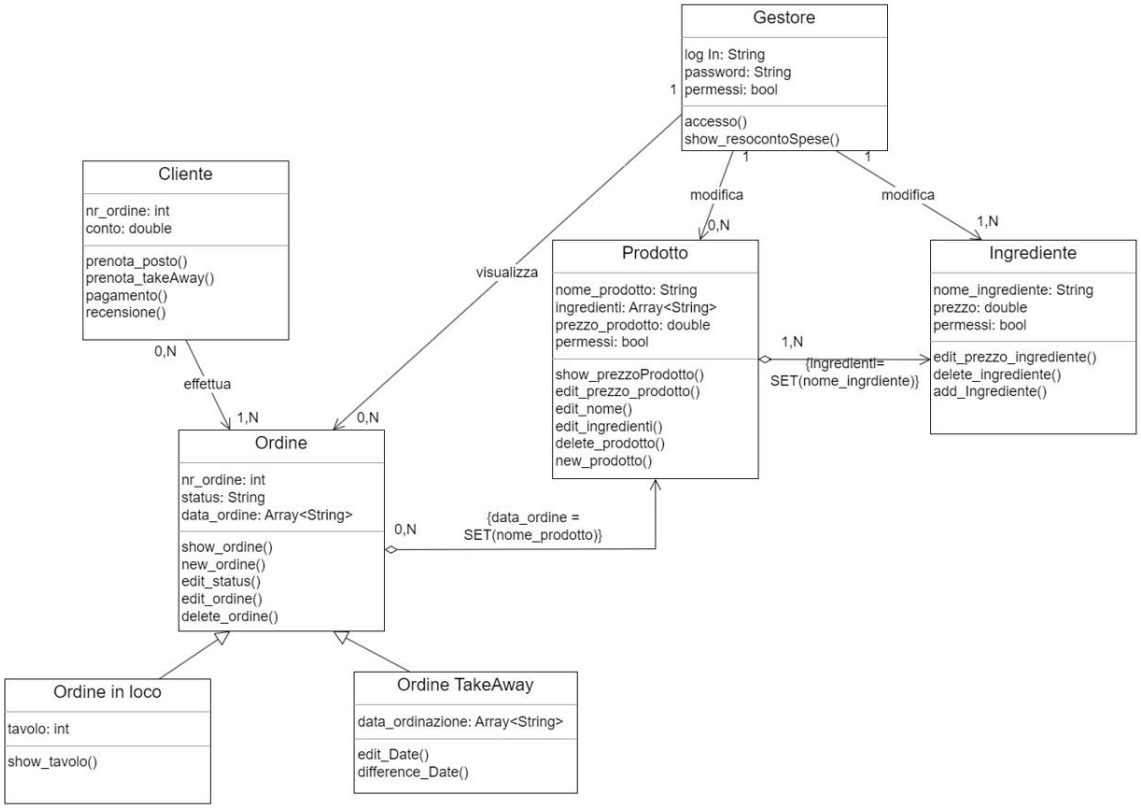
La classe Gestore è quella dell'amministratore del sistema di Paninaro Next, e comprende degli attributi per il login (username e password), e dei metodi per accedere e visualizzare il resoconto della sua attività.

1.4. Prodotti e Ingredienti



Le due classi prodotti e ingredienti sono legate tra loro, in quanto ogni prodotto è composto da degli ingredienti. In Ingrediente è presente il nome e il prezzo, con dei metodi per modificare, eliminare e aggiungere gli ingredienti. In Prodotto è invece presente la lista degli ingredienti da cui è composto, un metodo per calcolarne il prezzo dinamicamente e altri per modificare, eliminare e aggingere prodotti. Infine, per accedere a queste classi sono necessari i permessi da amministratore.

1.5. Diagramma delle classi complessivo



2. Codice in Object Constraint Language

In questo capitolo è descritta in modo formale la logica prevista nell'ambito di alcune operazioni di alcune classi. Tale logica viene descritta in Object Constraint Language (OCL) perché tali concetti non sono esprimibili in nessun altro modo formale nel contesto di UML.

1. Cliente e ordine

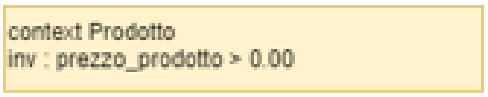
   Quando un cliente fa un ordine questo deve avere lo stesso numero;

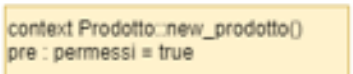
2. Ordine TakeAway

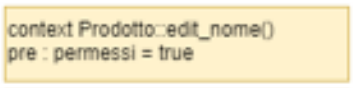
   La data di ordinazione deve essere minore o uguale alla data di ritiro dell'ordine nella classe TakeAway;

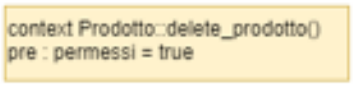
3. Prodotto

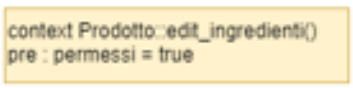
   Il prezzo di ogni prodotto deve essere strettamente maggiore di 0. E per modificare, aggiungere o cancellare un prodotto servono gli adeguati permessi;

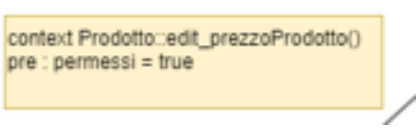












4. Ingredienti

Per aggiungere, modificare o cancellare un prodotto sono necessari i permessi di amministratore. In aggiunta, il prezzo di ogni ingrediente deve essere strettamente maggiore di 0;

```
context Ingrediente::add_ingrediente()
pre : permessi = true

context Ingrediente::edit_prezzo_ingrediente()
pre : permessi = true

context Ingrediente::delete_ingrediente()
pre : permessi = true

context Ingrediente
inv : prezzo > 0.00
```

ica

5. Gestore

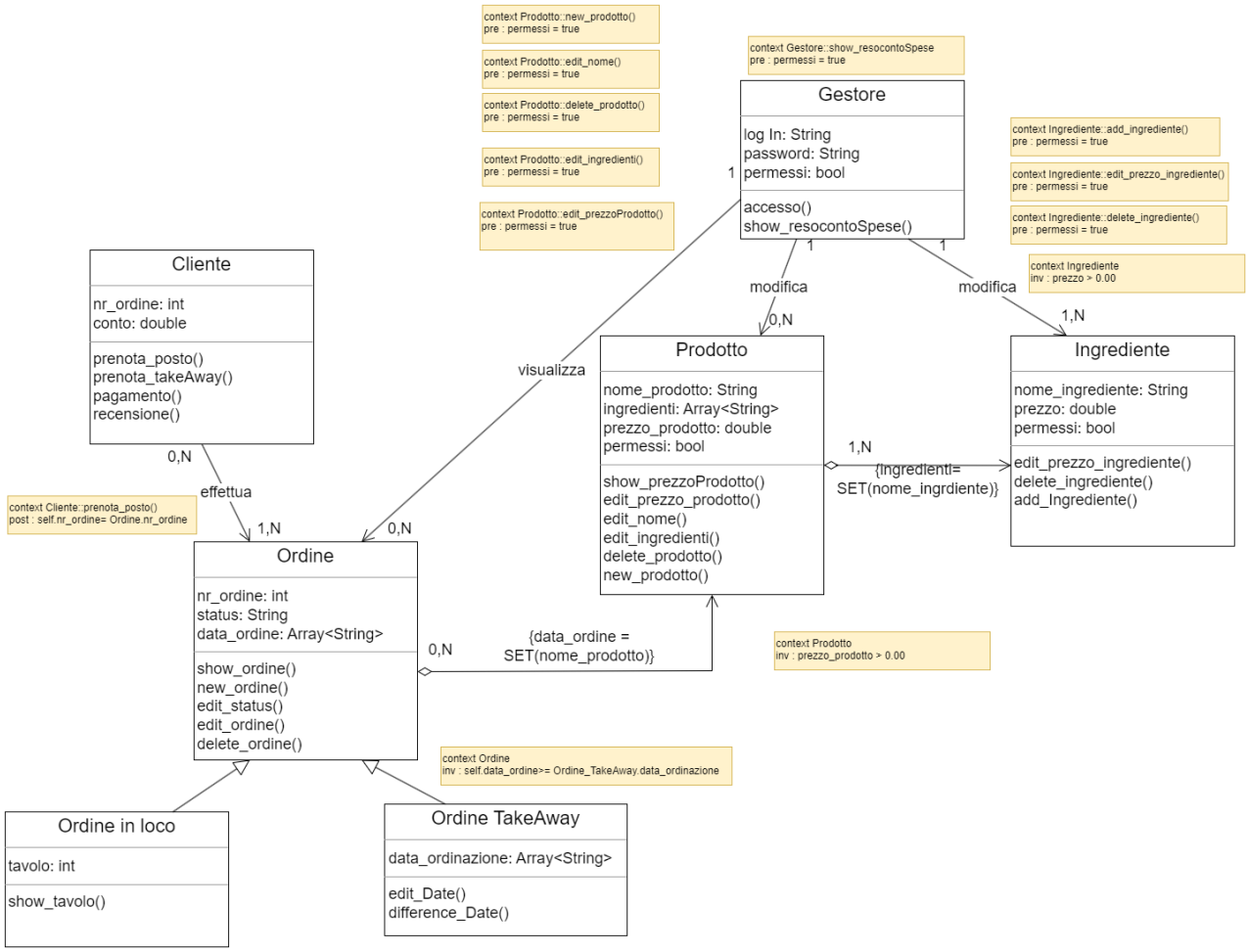
   E' possibile vedere il resoconto dell'attività solo se si possiedono i permessi.

```
context Gestore::show_resocontoSpese
pre : permessi = true
```

3. Diagramma delle classi con codice OCL

Riportiamo infine il diagramma delle classi con tutte le classi fino ad ora presentate ed il codice OCL individuato.

Gruppo 13


context Prodotto::new_prodotto()
pre : permessi = true
context Prodotto::edit_nome()
pre : permessi = true
context Prodotto::delete_prodotto()
pre : permessi = true
context Prodotto::edit_ingredienti()
pre : permessi = true
context Prodotto::edit_prezzoProdotto()
pre : permessi = true
context Gestore::show_resocontoSpese
pre : permessi = true
Gestore
log In: String
password: String
permessi: bool
accesso()
show_resocontoSpese()
context Ingrediente::add_ingrediente()
pre : permessi = true
context Ingrediente::edit_prezzo_ingrediente()
pre : permessi = true
context Ingrediente::delete_ingrediente()
pre : permessi = true
context Ingrediente
inv : prezzo > 0.00
Cliente
nr_ordine: int
conto: double
prenota_posto()
prenota_takeAway()
pagamento()
recensione()
modifica
modifica
visualizza
effettua
context Cliente::prenota_posto()
post : self.nr_ordine= Ordine.nr_ordine
Prodotto
nome_prodotto: String
ingredienti: Array<String>
prezzo_prodotto: double
permessi: bool
show_prezzoProdotto()
edit_prezzo_prodotto()
edit_nome()
edit_ingredienti()
delete_prodotto()
new_prodotto()
{Ingredienti=
SET(nome_ingrdiente)}
Ingrediente
nome_ingrediente: String
prezzo: double
permessi: bool
edit_prezzo_ingrediente()
delete_ingrediente()
add_Ingrediente()
Ordine
nr_ordine: int
status: String
data_ordine: Array<String>
show_ordine()
new_ordine()
edit_status()
edit_ordine()
delete_ordine()
{data_ordine =
SET(nome_prodotto)}
context Prodotto
inv : prezzo_prodotto > 0.00
context Ordine
inv : self.data_ordine>= Ordine_TakeAway.data_ordinazione
Ordine in loco
tavolo: int
show_tavolo()
Ordine TakeAway
data_ordinazione: Array<String>
edit_Date()
difference_Date()