UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e
Scienza dell'Informazione

Progetto:

Baccalà

Titolo del documento:

Documento di architettura

Scopo del documento

Il presente documento illustra e specifica l'architettura del sistema del progetto baccalà tramite diagrammi delle classi in UML e codice OCL.

Enrico Dalla Croce
Andrea Ervetti
Matteo Possamai

Indice

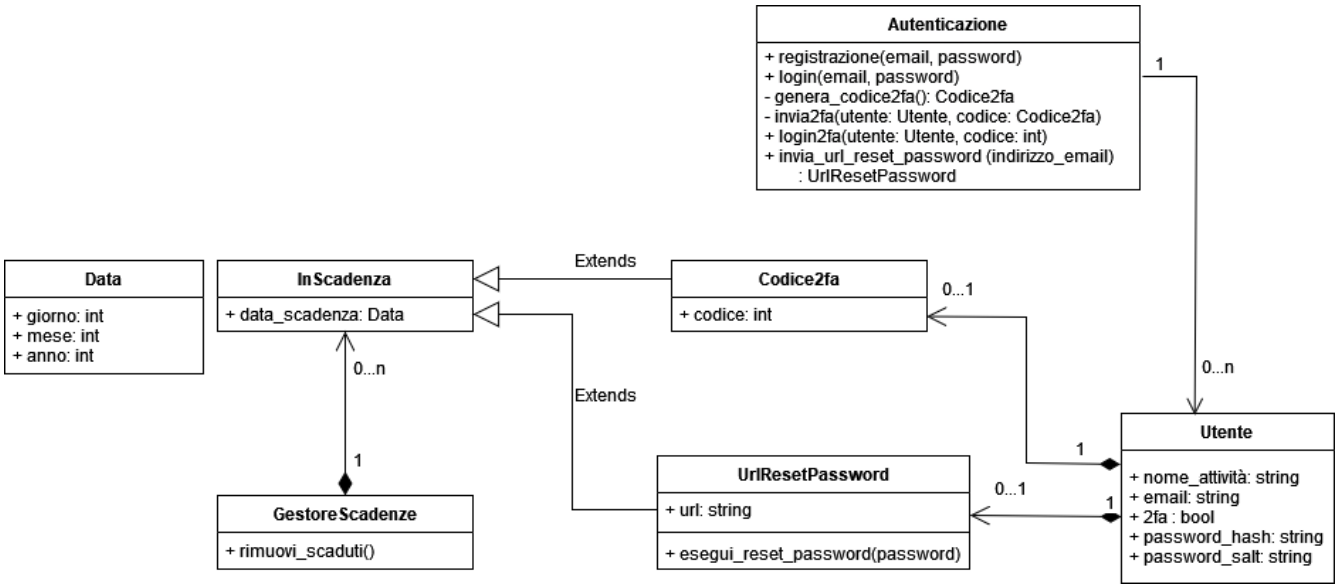
Diagramma delle classi (UML)

Di seguito viene analizzato in piccole porzioni lo schema UML del progetto.
L'obiettivo è quello di approfondire singolarmente parti distinte del software.

Precisazione: la gestione dell'interfaccia multilingua sarà affidata alla libreria i18n. Gli attributi *nome* rappresentano la chiave univoca per identificare un elemento di testo e visualizzarne la versione relativa alla lingua scelta.

Autenticazione

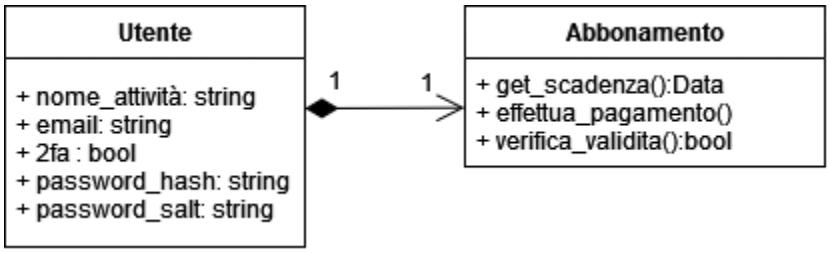
La classe *Autenticazione* implementa le funzionalità di registrazione, login e reset della password. È inoltre possibile abilitare l'autenticazione a due fattori attraverso email.
La classe *UrlResetPassword* contiene il metodo che permette di aggiornare il database con la nuova password inserita dall'utente a seguito del processo del reset della password.
La classe *GestoreScadenze* si occupa di eliminare le istanze delle sottoclassi di *InScadenza*, ovvero *Codice2fa* e *UrlResetPassword*, una volta che sono scadute.
Un'istanza della classe *Utente* contiene i dati di autenticazione di un utente, in particolare contiene l'hash e salt della password. Il salt della password è una stringa pseudo-randomica generata prima di effettuare l'hash, mentre l'hash è generato dall'algoritmo argon2 a partire dall'unione di password e salt.



Abbonamento

La classe *Abbonamento* si interfaccia al servizio Stripe e implementa le seguenti funzionalità:

- fornire al sistema le informazioni sulla validità dell'abbonamento di ogni utente;
- redirigere l'utente alla pagina di pagamento;
- mostrare all'utente la data di scadenza del proprio abbonamento.

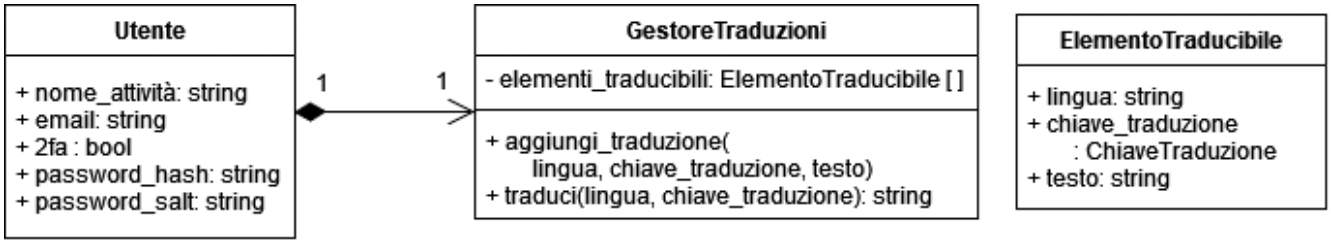


Traduzioni

La classe *GestoreTraduzioni* si occupa della traduzione degli elementi testuali del menù. Contiene un insieme di istanze della classe *ElementoTraducibile*, ovvero corrispondenze chiave-valore dove la chiave è composta dagli elementi *lingua* e *chiave_traduzione* e il valore è l'elemento *testo*. I suoi metodi permettono le seguenti funzioni:

- aggiungere un'istanza di *ElementoTraducibile*;
- trovare il testo corrispondente ad una coppia chiave_traduzione-lingua cercando nell'insieme *elementi_traducibili*.

Ogni utente gestore, quindi, interagisce con il proprio dizionario aggiungendo traduzioni che verranno successivamente mostrate nel menù in base alla lingua selezionata dall'utente consumatore.

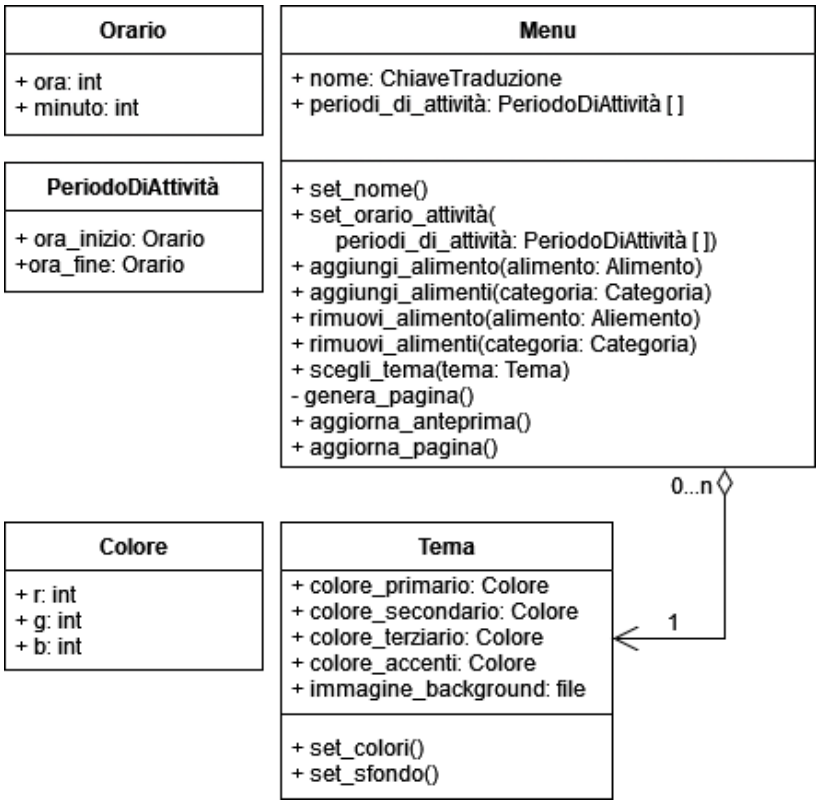


Menu

Un'istanza della classe *Menu* gestisce tutto quello che riguarda un singolo menù:

- aggiungere e rimuovere alimenti;
- cambiare il periodo di attività del menù;
- scegliere il tema tra quelli presenti;
- generare la pagina da mostrare all'utente consumatore.

La classe *Tema* ha lo scopo di permettere all'utente di personalizzare l'aspetto del proprio menù modificando i colori e lo sfondo.



Alimento

Un'istanza della classe *Alimento* gestisce le informazioni relative a un alimento, in particolare la lista degli ingredienti.

Un'istanza della classe *Categoria* rappresenta un insieme di alimenti che può essere usato dall'utente consumatore come filtro durante la navigazione di un menù.

Un ingrediente può comparire in diversi alimenti senza creare duplicati per lo stesso ingrediente; questo vale anche per gli alimenti che compaiono in diversi menù o diverse categorie.

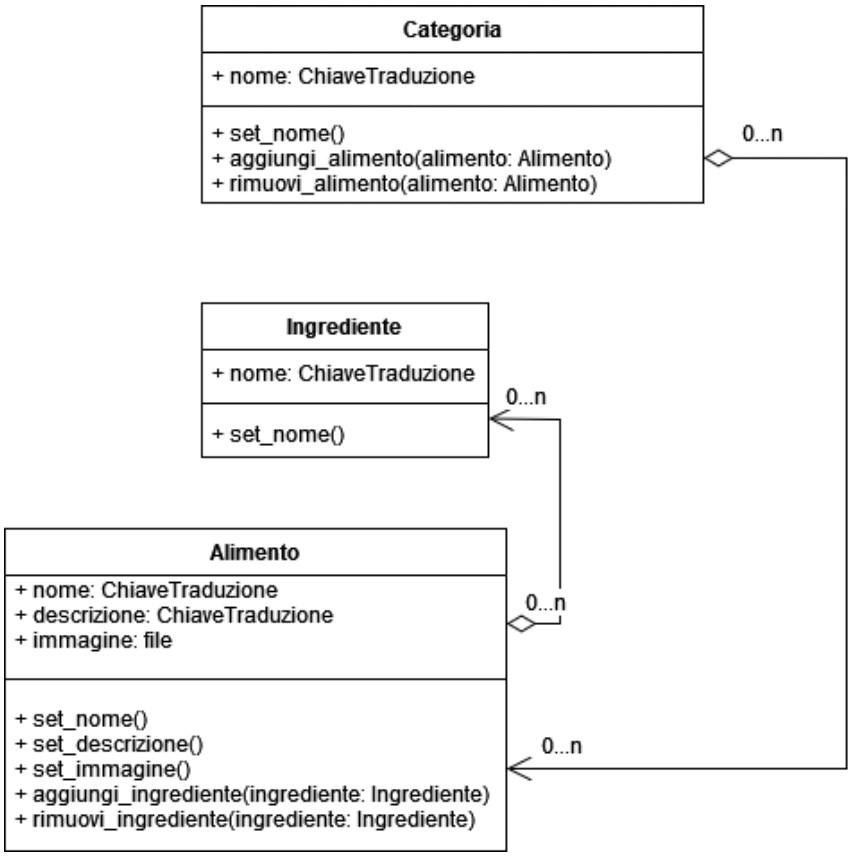
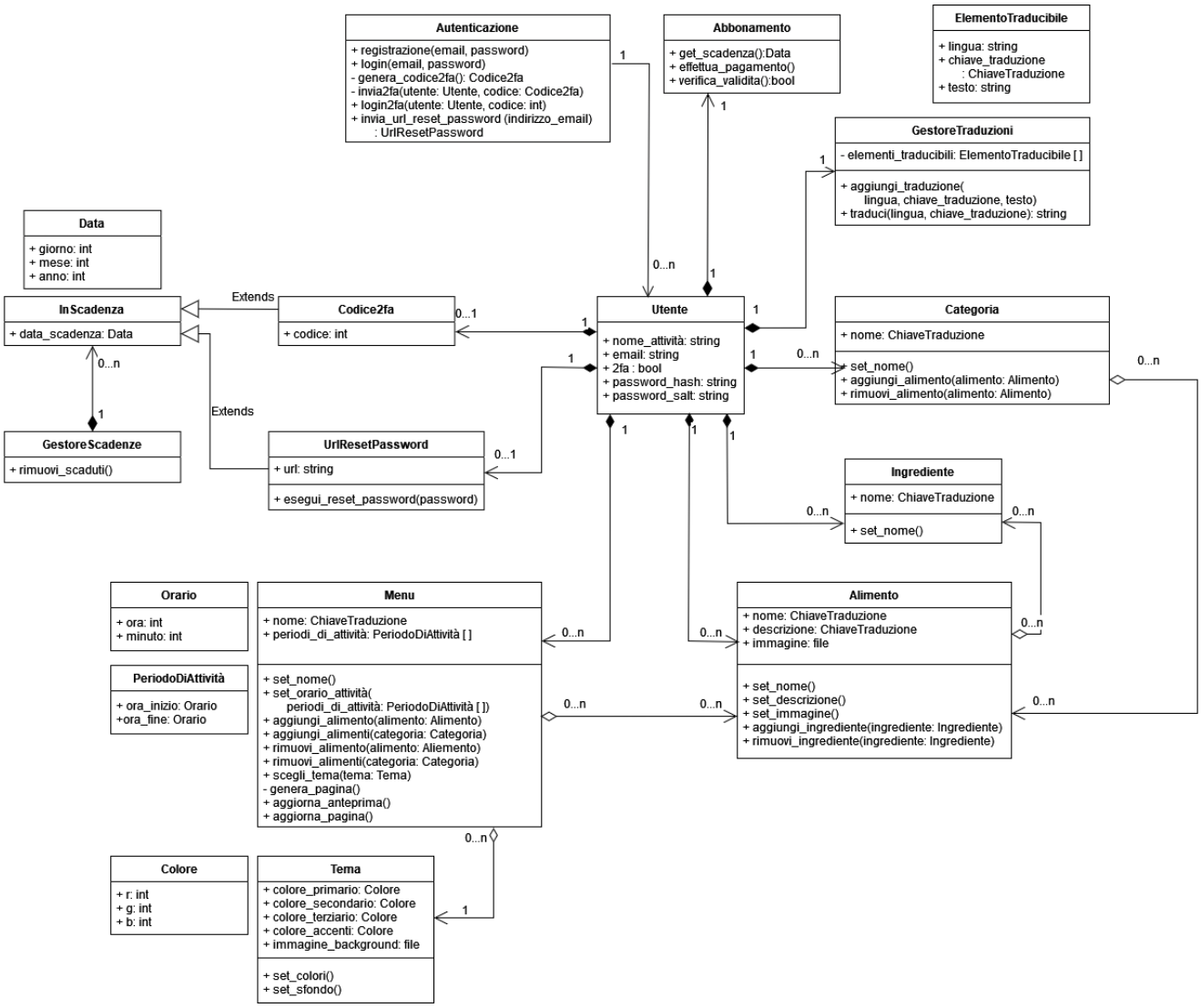


Diagramma Completo

Questo è il diagramma complessivo del nostro applicativo dove si possono vedere tutte le cardinalità e le connessioni tra gli elementi.



Object Constraint Language (OCL)

Di seguito vengono analizzate le logiche degli elementi non esprimibili attraverso lo schema UML.

Periodo di attività del menù

La seguente invariante è necessaria per evitare che l'utente inserisca fasce orarie invalide.

```
context PeriodoDiAttività inv:
ora_inizio < ora_fine
```

Invio del codice per l'autenticazione a due fattori (2FA)

Prima di inviare e generare il codice per l'autenticazione a due fattori si verifica che l'utente abbia attivato tale funzione.

```
context Autenticazione::invia2fa(utente: Utente, codice: Codice2fa) pre:
utente.2fa = true
```

Login verificando la validità del codice inserito (2FA)

Un utente che non ha attivato l'autenticazione a due fattori non dovrebbe poter inserire un codice per completare l'accesso.
Lasciare attivo questo servizio anche se inutilizzato non è una buona pratica perché rappresenterebbe una superficie di attacco.

```
context Autenticazione::login2fa(utente: Utente, codice: Codice2fa) pre:
utente.2fa = true
```

Diagramma delle classi (UML) completato con codice OCL

Il seguente diagramma rappresenta la struttura completa del nostro applicativo integrata con il codice OCL.

Questo è lo schema integrale, utilizzabile come documentazione, su cui si basa l'implementazione