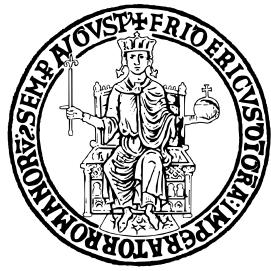


UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II



SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN INFORMATICA

OBJECT ORIENTATION

UNINAFOODLAB APPLICATION

Anno Accademico 2024–2025

# Indice

<b>1 Descrizione del progetto</b>	<b>2</b>
1.1 Descrizione sintetica del problema . . . . .	2
<b>2 Diagramma delle Classi</b>	<b>3</b>
2.1 UML . . . . .	3
2.2 Architettura . . . . .	3
<b>3 Descrizione Package e classi</b>	<b>5</b>
3.1 Descrizione del package Boundary . . . . .	5
3.2 Package <i>Control</i> . . . . .	5
3.2.1 Classe Controller . . . . .	5
3.3 Package <i>Entity</i> . . . . .	5
3.4 Package <i>DaoInterface</i> . . . . .	5
3.5 Package <i>DAO</i> . . . . .	5
3.5.1 Classe ChefDAO . . . . .	5
3.5.2 Classe CorsoDAO . . . . .	6
3.5.3 Classe NotificaDAO . . . . .	6
3.5.4 Classe PreparaDAO . . . . .	6
3.5.5 Classe RicettaDAO . . . . .	6
3.5.6 Classe SessioneOnlineDAO . . . . .	6
3.5.7 Classe SessioneInPresenzaDAO . . . . .	6
3.5.8 Classe TopicDAO . . . . .	6
3.6 Package <i>DbConnection</i> . . . . .	6
3.6.1 Classe DbConnection . . . . .	7
<b>4 Repository GitHub</b>	<b>8</b>
4.1 Cos'è una Repository . . . . .	8
4.2 Struttura del Progetto . . . . .	8
4.3 Accesso al Codice . . . . .	8

# Capitolo 1

## Descrizione del progetto

### 1.1 Descrizione sintetica del problema

Si sviluppi un applicativo Java con interfaccia grafica per la gestione dei corsi tematici offerti dalla piattaforma UninaFoodLab. Il sistema dovrà essere collegato a un database relazionale prepopolato contenente informazioni su chef, ricette e ingredienti. Il sistema deve permettere l'autenticazione degli chef tramite credenziali (username e password). Una volta autenticato, lo chef può aggiungere un nuovo corso, specificando le seguenti informazioni: categoria, data di inizio, frequenza delle sessioni, numero di sessioni. Per ciascuna sessione, deve essere indicata la modalità di svolgimento, ovvero se si tratta di una sessione online o in presenza. Lo chef avrà inoltre la possibilità di visualizzare i corsi esistenti, applicando filtri per categoria. Dopo aver selezionato un corso, lo chef può associare a ciascuna sessione pratica una o più ricette da realizzare. Infine, il sistema deve fornire un report mensile, che permette allo chef di visualizzare: il numero di corsi totali tenuti, il numero di sessioni online e pratiche, e di quest'ultime il numero medio, massimo e minimo di ricette realizzate. Il report deve fornire una rappresentazione grafica dei dati. Il sistema deve permettere allo chef di inserire delle notifiche relative ai propri corsi, in caso di modifiche, come cambio di data o ora di una sessione oppure la sua cancellazione. Durante la creazione della notifica, lo chef specifica se è destinata ad un singolo corso oppure se è risolta a tutti. Tutte le notifiche sono consultabili in un'apposita sezione dell'interfaccia.

# Capitolo 2

## Diagramma delle Classi

### 2.1 UML

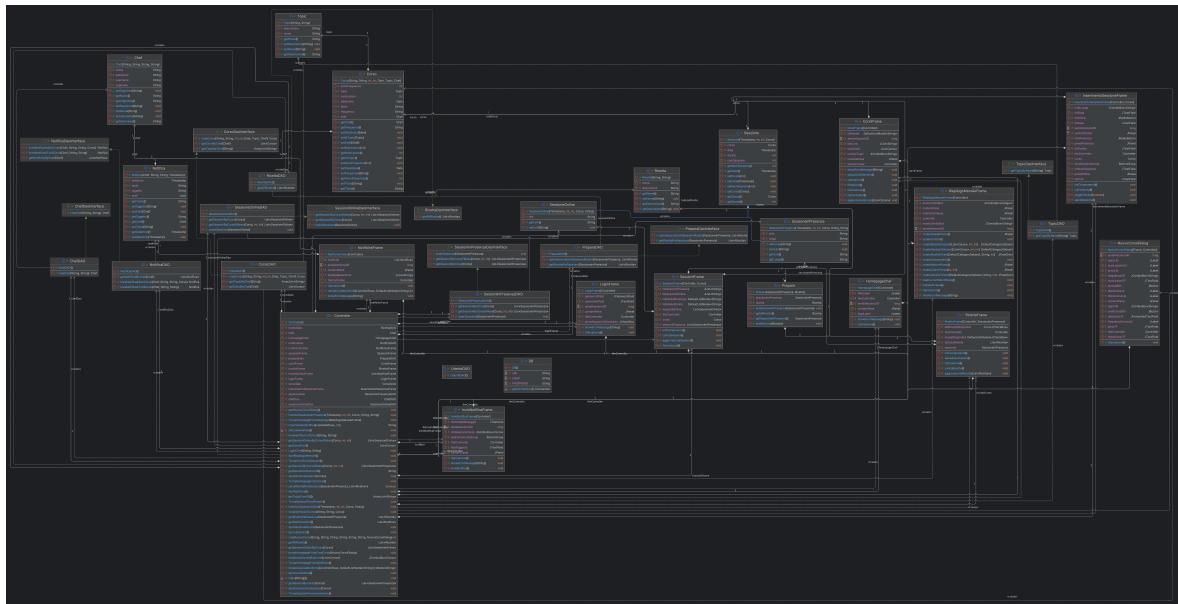


Figura 2.1: Diagramma delle classi UML

### 2.2 Architettura

L'Architettura dell'applicativo è basata sul pattern EBC (Entity Boundary Control). Questo pattern prevede la separazione delle responsabilità tra le diverse componenti del sistema, facilitando la manutenibilità e l'estensibilità dell'applicativo. Come si può vedere dal diagramma UML, le classi sono suddivise in tre categorie principali con ognuna di esse che svolge un ruolo specifico:

- **Entity:** Queste classi rappresentano le entità principali del dominio dell'applicativo, come Chef, Corso, Sessione, Ricetta, Ingrediente e Notifica. Esse contengono gli attributi e i metodi necessari per gestire i dati relativi a queste entità.

- **Boundary:** Le classi di questa categoria sono responsabili dell'interfaccia utente e della gestione delle interazioni con l'utente. Ad esempio, la classe UninaFoodLabApp funge da punto di ingresso per l'applicativo, mentre altre classi come LoginFrame, MainFrame e CourseManagementFrame gestiscono le diverse schermate dell'interfaccia grafica.
- **Control:** Queste classi agiscono come intermediari tra le classi Entity e Boundary. Gestiscono la logica di business dell'applicativo, coordinando le operazioni tra le entità e l'interfaccia utente. Ad esempio, la classe CourseController gestisce le operazioni relative ai corsi, mentre NotificationController si occupa della gestione delle notifiche.

Per l'interfaccia grafica è stato utilizzato il framework Swing di Java, che offre una vasta gamma di componenti per la creazione di interfacce utente ricche e interattive. Le classi del package Boundary estendono JFrame e utilizzano vari componenti Swing come JButton, JTextField, JComboBox e JTable per costruire le diverse schermate dell'applicativo. La connessione al database è gestita tramite JDBC (Java Database Connectivity), che consente di eseguire query SQL e di interagire con il database relazionale.

# Capitolo 3

## Descrizione Package e classi

### 3.1 Descrizione del package Boundary

Il pacchetto contiene le classi della gui

### 3.2 Package *Control*

Il pacchetto contiene la classe controller che gestisce la logica dell'applicativo determinando le interazioni tra le classi e la gestione delle risorse.

#### 3.2.1 Classe Controller

Il controller, seguendo gli schemi dell'architettura BCE, si occupa di gestire le interazioni tra le interfacce del programma e le entità del dominio. Di conseguenza il controller ha il compito di amministrare l'apertura e la chiusura delle finestre della GUI, di interpretare l'interazione tra l'utente e gli elementi dell'interfaccia grafica, di gestire l'allocazione e la liberazione di risorse interagendo con il database mediante l'utilizzo delle classi DAO.

### 3.3 Package Entity

### 3.4 Package *DaoInterface*

Il package contiene le interfacce che definiscono i metodi per l'accesso ai dati nel database. Inoltre nel caso in cui si volesse cambiare il database, basterebbe creare una nuova classe DAO che implementa queste interfacce.

### 3.5 Package *DAO*

Il package contiene le classi che si occupano della connessione al database e dell'esecuzione delle query SQL.

#### 3.5.1 Classe ChefDAO

La classe ChefDAO si occupa della gestione delle operazioni relative alla tabella Chef del database. Tramite il metodo creaChef di questa classe, è possibile istanziare un oggetto Chef a partire dai dati presenti nel database ed effettuare il login nell'applicativo.

### **3.5.2 Classe CorsoDAO**

La classe CorsoDAO si occupa della gestione delle operazioni relative alla tabella Corso del database. Tramite i metodi di questa classe, è possibile aggiungere un nuovo corso, recuperare i corsi esistenti e applicare filtri per categoria.

### **3.5.3 Classe NotificaDAO**

La classe NotificaDAO si occupa della gestione delle operazioni relative alla tabella Notifica del database.

Tramite i metodi di questa classe, è possibile aggiungere una nuova notifica, recuperare le notifiche esistenti e visualizzarle nell'apposita sezione dell'interfaccia.

### **3.5.4 Classe PreparaDAO**

La classe PreparaDAO si occupa della gestione delle operazioni relative alla tabella Prepara del database.

Tramite i metodi di questa classe, è possibile associare una o più ricette a ciascuna sessione pratica di un corso.

### **3.5.5 Classe RicettaDAO**

La classe RicettaDAO si occupa della gestione delle operazioni relative alla tabella Ricetta del database. Tramite i metodi di questa classe, è possibile recuperare le ricette esistenti e visualizzarle nell'apposita sezione dell'interfaccia.

### **3.5.6 Classe SessioneOnlineDAO**

La classe SessioneOnlineDAO si occupa della gestione delle operazioni relative alla tabella SessioneOnline del database.

Tramite i metodi di questa classe, è possibile aggiungere una nuova sessione online o recuperare le sessioni online esistenti e visualizzarle nell'apposita sezione dell'interfaccia.

### **3.5.7 Classe SessioneInPresenzaDAO**

La classe SessioneInPresenzaDAO si occupa della gestione delle operazioni relative alla tabella SessioneInPresenza del database.

Tramite i metodi di questa classe, è possibile aggiungere una nuova sessione in presenza o recuperare le sessioni in presenza esistenti e visualizzarle nell'apposita sezione dell'interfaccia.

### **3.5.8 Classe TopicDAO**

La classe TopicDAO si occupa della gestione delle operazioni relative alla tabella Topic del database. Tramite i metodi di questa classe, è possibile recuperare i topic esistenti e visualizzarle nell'apposita sezione dell'interfaccia.

## **3.6 Package *DbConnection***

Il package contiene la classe che si occupa della connessione al database.

### **3.6.1 Classe DbConnection**

La classe DbConnection si occupa di gestire la connessione al database. Fornisce un metodo statico `getConnection()` che restituisce un oggetto `Connection` per interagire con il database.

# Capitolo 4

## Repository GitHub

### 4.1 Cos'è una Repository

Una repository è uno spazio di archiviazione digitale dove vengono conservati i file di un progetto software insieme alla loro cronologia completa delle modifiche. GitHub è una piattaforma web che ospita repository Git, offrendo strumenti collaborativi per lo sviluppo software.

### 4.2 Struttura del Progetto

Il progetto UninaFoodLab è organizzato in una struttura chiara che facilita la navigazione e la comprensione del codice:

- **Codice sorgente:** Organizzato in package secondo il pattern architettonale EBC
- **Documentazione:** Include diagrammi UML, specifiche tecniche e guide utente
- **Database:** Script SQL per la creazione e il popolamento del database
- **Risorse:** Immagini, icone e altri file multimediali utilizzati dall'applicazione

### 4.3 Accesso al Codice

Il codice sorgente del progetto è disponibile al seguente indirizzo:

<https://github.com/xFireFox27/UninaFoodLab.git>

La repository contiene l'intero progetto con la cronologia completa delle modifiche, permettendo di seguire l'evoluzione del software durante lo sviluppo.