## **Immagine che contiene testo Descrizione generata automaticamente**

## **RECIPE MANAGER**

Sistem Design Document

Anno Accademico 2019/20

**UNIVERSITA’ DEGLI STUDI DI SALERNO – FACOLTA DI SCIENZE FF. MM. NN.**

**CORSO DI LAUREA TRIENNALE IN INFORMATICA – CORSO DI INGEGNERIA DEL SOFTWARE**



## **Sommario**

[1](#_Toc32069507)

[**RECIPE MANAGER** 1](#_Toc32069508)

[**Sommario** 2](#_Toc32069509)

[**Introduzione** 4](#_Toc32069510)

[**Scopo del sistema** 4](#_Toc32069511)

[**Obiettivi di design** 4](#_Toc32069512)

[**Tabella degli obiettivi di design** 5](#_Toc32069513)

[Throughput 5](#_Toc32069514)

[Memoria 5](#_Toc32069515)

[Robustezza 5](#_Toc32069516)

[Disponibilità 5](#_Toc32069517)

[Sicurezza 5](#_Toc32069518)

[Estensibilità 5](#_Toc32069519)

[**Definizioni, acronimi e abbreviazioni** 6](#_Toc32069520)

[**Riferimenti** 6](#_Toc32069521)

[**Panoramica** 6](#_Toc32069522)

[**Architettura corrente del software** 7](#_Toc32069523)

[**Architettura software proposta** 8](#_Toc32069524)

[**Panoramica** 8](#_Toc32069525)

[**Decomposizione dei sottosistemi** 9](#_Toc32069526)

[**Utente Non Registrato** 11](#_Toc32069527)

[**Utente** 12](#_Toc32069528)

[**Amministratore** 13](#_Toc32069529)

[**Sistema Generale** 14](#_Toc32069530)

[**Gestione Account** 15](#_Toc32069531)

[**Gestione Ricetta** 17](#_Toc32069532)

[**Gestione Amministrazione** 17](#_Toc32069533)

[**Mappatura Hardware/Software** 15](#_Toc32069534)

[**Gestione dati persistenti** 18](#_Toc32069535)

[**Mapping del database** 19](#_Toc32069536)

[**Dettaglio struttura tabelle** 21](#_Toc32069537)

[**Utente** 21](#_Toc32069538)

[Cognome 21](#_Toc32069539)

[**Ricetta** 21](#_Toc32069540)

[**Ingrediente** 21](#_Toc32069541)

[Nome 21](#_Toc32069542)

[**Categoria** 22](#_Toc32069543)

[**Segnalazione** 22](#_Toc32069544)

[**Commento** 22](#_Toc32069545)

[**Ruolo** 22](#_Toc32069546)

[**Controllo accessi e sicurezza** 23](#_Toc32069547)

[**ACL** 23](#_Toc32069548)

[**Controllo del software globale** 24](#_Toc32069549)

[**Condizione dei boundary** 24](#_Toc32069550)

[Avvio del sistema 24](#_Toc32069551)

[**Terminazione del sistema** 24](#_Toc32069552)

[**Fallimento del sistema** 24](#_Toc32069553)

[Servizi dei sottosistemi 25](#_Toc32069554)

[**Sottosistema Gestione Utente** 25](#_Toc32069555)

[Descrizione 25](#_Toc32069556)

[**Servizio** 25](#_Toc32069557)

[Descrizione 25](#_Toc32069558)

[**Sottosistema Gestione Ricetta** 26](#_Toc32069559)

[Crea Ricetta 26](#_Toc32069560)

[Visualizza lista ricette 26](#_Toc32069561)

[Visualizza dettaglio ricetta 26](#_Toc32069562)

[Filtra Ricetta 26](#_Toc32069563)

[Elimina ricetta 26](#_Toc32069564)

[**Sottosistema Gestione Amministrazione** 27](#_Toc32069565)

[Gestione Amministrazione 27](#_Toc32069566)

[Inserisci Categoria 27](#_Toc32069567)

[Modifica Commento 27](#_Toc32069568)

[Elimina Commento 27](#_Toc32069569)

[**Condizioni Baundary** 28](#_Toc32069570)

[**Scenari** 28](#_Toc32069571)

[**Casi d’uso** 30](#_Toc32069572)

[Glossario 31](#_Toc32069573)

## 

## **Introduzione**

### **Scopo del sistema**

Oggigiorno sono tanti i siti in cui è possibile visionare delle ricette, spesso create da chef famosi o da personaggi dello spettacolo e della televisione, ma nessuno di questi prevede ad un utente generico di poter inserire la propria ricetta e allo stesso tempo di esporre un parere sulle ricette altrui.

RecipeManager si prefigge lo scopo di dare un’opportunità a chiunque abbia fantasia e talento necessario, a condividere la propria ricetta online e lasciare che siano gli utenti stessi del sito a giudicarla. Il sistema è pensato anche per coloro che vogliano semplicemente conservare le proprie ricette senza dover necessariamente renderle pubbliche, in quanto sarà provvisto di una sezione personale accessibile solo dall’utente di cui fa parte.

### **Obiettivi di design**

Il sito dovrà essere quanto più semplice e veloce possibile, in modo da dare un’opportunità anche agli utenti meno esperti di inserire e visualizzare le proprie ricette. Inoltre, dovrà rispettare tutti i standard di sicurezza per garantire che i dati dell’utente siano sempre protetti ed essere strutturalmente semplice per permetterne una facile manutenibilità ed estendibilità.

Coordinatore:

|  |
| --- |
| **Professore** |
| Prof. De Lucia Andrea |

Partecipanti:

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| Amoroso Gaetano | 0512105246 |
| Tranquillo Andrea | 0512103486 |

### **Tabella degli obiettivi di design**

|  |  |
| --- | --- |
| **Tempo di Risposta** | Il sito dovrà essere, in ogni momento della giornata, altamente reattivo, mantenendosi con tempi di risposta al di sotto del secondo. Questo permetterà all’utente di poter facilmente trovare la sua ricetta preferita nel minor tempo possibile |
| **Tempo di Sessione** | Il tempo di Sessione di un utente dovrà essere molto lungo, in quanto egli passerà gran parte del tempo a visionare una stessa ricetta. Se cosi non dovesse essere, l’utente si ritroverebbe a dover effettuare l’accesso ogni volta che avrà provato una ricetta disponibile sulla sua pagina personale o nella lista pubblica di ricette |
| Throughput | Avendo necessità di mantenere un lungo tempo di sessione per ogni utente, il server dovrà essere abbastanza potente da poter gestire una elevata quantità di utenti connessi contemporaneamente, facendo in modo che sia reattivo soprattutto negli orari di picco di utenza (10-13 e 17-21) |
| Memoria | Il sistema su cui sarà presente il database dovrà avere abbastanza memoria da conservare tutti i dati degli utenti e tutte le ricette che creeranno |

|  |  |
| --- | --- |
| Robustezza | Il sistema dovrà essere in grado di mantenere integri tutti i dati nel caso in cui ci sia un malfunzionamento hardware/software oppure, essendo un sito web, nel caso in cui ci sia un attacco informatico al sistema stesso |
| Disponibilità | Il sito dovrà essere accessibile in tutte le sue funzionalità 24 ore al giorno |
| Sicurezza | Verranno adottati dei criteri di sicurezza per rendere sicuri tutti i dati degli utenti, tra cui l’adozione di funzioni hash che permettano la criptazione delle password e l’uso di tabelle di accesso per consentire all’utente di effettuare solo operazioni di cui ha ricevuto i permessi |

|  |  |
| --- | --- |
| Estensibilità | Il sistema sarà sviluppato con tutte le nuove tecnologie disponibili per i web server che permetteranno una facile estensione delle funzionalità di tutto il sito |
| **Modificabilità** | Il codice sarà scritto in maniera precisa ed ordinata, mantenendo l’uso dei commenti per identificare in modo più semplice la funzione del sito che conterrà bug o malfunzionamenti |

### **Definizioni, acronimi e abbreviazioni**

* RecipeManager: nome del sistema che si andrà a sviluppare
* Visitatore: utente non registrato a RecipeManager e che potrà solo effettuare operazioni di ricerca e visualizzazione
* Web Server: software che si occupa della gestione di tutte le componenti lato web, ovvero Apache 9.0
* DBMS: sta per “Database Management System” ed è il software che permette agli amministratori del sito, o a procedure automatiche, di effettuare operazioni sul database

### **Riferimenti**

* Bern Bruegge, Allen H. Dutoit, Object-Oriented Software Engineering - Using UML, Patterns, and JAVA, 3rd edition.

### **Panoramica**

Il documento è strutturato in diverse sezioni, in cui sarà possibile trovare:

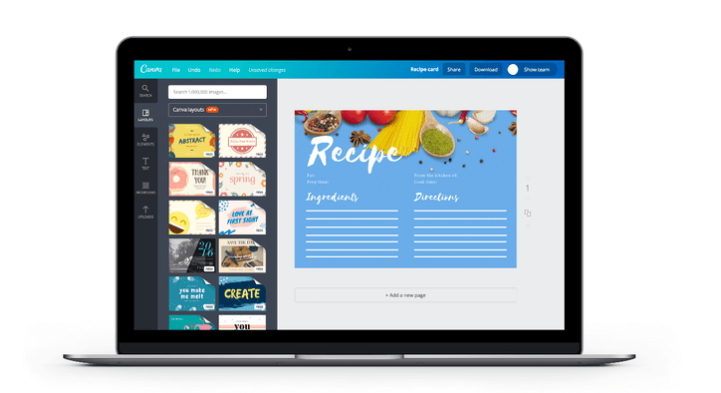
* Architettura corrente del software: sezione in cui si va a descrivere l’architettura di sistemi simili presenti sul web, ed in cui vi si discuterà i punti in comune
* Architettura software proposta: questa sezione è divisa in diverse sottosezioni (elencate di seguito) e che descriveranno più nel dettaglio tutte le scelte di design che sono state fatte per il sistema che si andrà a sviluppare
* Decomposizione in sottosistemi: il sistema sarà diviso in diversi sottosistemi e questa sezione elenca le responsabilità di ognuno di essi
* Mapping hardware/software: descrive in che modo i sottosistemi vengono assegnati ai diversi componenti hardware e software adottati per il sistema generale
* Gestione dati persistenti: descrive quali dati devono essere resi persistenti (quindi conservati) all’interno del database e quali sono le infrastrutture richieste per consentire tutto ciò. Include la descrizione degli schemi di dati e quale database verrà utilizzato
* Controllo accessi e sicurezza: sezione in cui vengono descritti tutti i parametri di sicurezza adottati dal sistema e i diversi ruoli assegnati agli utenti attraverso l’uso di tabelle di accesso
* Controllo del software globale: descrive il modo in cui è implementato il controllo globale del software e come si sincronizzano i sottosistemi.
* Condizione sui boundary: in questo paragrafo vengono descritte le condizioni limite del sistema come start-up e shutdown e la gestione dei fallimenti del sistema
* Servizi dei sottosistemi: qui vengono descritti i servizi offerti agli utenti del sito in base al sottosistema.

## **Architettura corrente del software**

Il sistema che andremo a creare non rimpiazzerà nessun altro sistema già esistente, pertanto verrà fatto un sondaggio su architetture simili già esistenti e gli obiettivi che i due sistemi hanno in comune. Uno dei siti online più conosciuti sullo sviluppo di ricette personali è senza ombra di dubbio Canva.

Canva è un sito che permette di creare ricette tramite modelli prefabbricati, in cui l’utente dovrà solo inserire tutti i campi necessari per completarla, quali il titolo, gli ingredienti e la modalità di preparazione. Una volta creata la ricetta l’utente potrà scegliere se conservarla nel suo archivio personale o condividerla sui social ai parenti o amici.

****



Gli obiettivi dei due sistemi sono pressoché simili. Infatti entrambi avranno un’interfaccia semplice ed intuitiva, avranno un modello di ricetta da far compilare all’utente ed entrambi dovranno gestire la sua condivisione. L’aspetto che differenzia in particolar modo i due siti è la modalità di condivisione. Canva permette di condividere la ricetta solo sui social, mentre il RecipeManager sarà dotato di una bacheca pubblica che ogni utente, registrato o non, potrà visualizzare.

## **Architettura software proposta**

### **Panoramica**

Nello sviluppo del nostro sistema utilizzeremo un’architettura di tipo MVC. IL pattern Model-View-Controller prevede che il software venga diviso in tre parti ognuna delle quali ha un compito diverso. In particolare:

* il model si occupa di gestire i dati e quindi sarà responsabile dell’interazione con il database sottostante
* la view si occupa di curare l’interazione con l’utente e quindi avrà il compito di gestire la formattazione dei dati che verranno visualizzati.
* Il controller dopo aver ricevuto i comandi forniti dall’utente si occuperà di elaborare i dati, passarli al model se necessario e inviare la risposta al view appropriato.

L’utilizzo del modello MVC comporta numerosi vantaggi tra i quali ricordiamo la possibilità di suddividere il lavoro più facilmente tra i vari componenti del team e la maggiore agilità negli interventi di manutenzione. Infatti, l’utilizzo di un modello e di regole standard facilita la comprensione anche chi non ha inizialmente lavorato su quel sottosistema.

All’interno del nostro sistema il model verrà realizzato utilizzando classi Java appropriate, la parte di view verrà implementata utilizzando pagine HTML e JavaScript, i control saranno realizzati tramite Servlet.

## **Decomposizione dei sottosistemi**

Per realizzare il sistema è stata usata un’architettura three-tier. Questo è un caso particolare di un’architettura multi tier in cui la logica dell’applicazione viene suddivisa in tre layer:

• Presentation Layer: composto dalle interfacce grafiche e in particolare dai boundary object come le form che vengono compilate dall’utente.

• Application Layer: composto dagli oggetti che di occupano della gestione del controllo, dell’elaborazione dati e di notificare i cambiamenti al presentation layer. Questo strato interagisce con il database sottostante tramite lo storage layer.

• Storage Layer: si occupa della memorizzazione dei dati persistenti e del loro recupero dal database ad esempio eseguendo delle query.

Durante la fase di analisi dei requisiti effettuata nel RAD, abbiamo individuato un insieme di oggetti, e quindi di classi, che componevano il sistema. Ora però bisogna raggruppare queste classi in diversi sottosistemi, ognuno con il suo scopo.

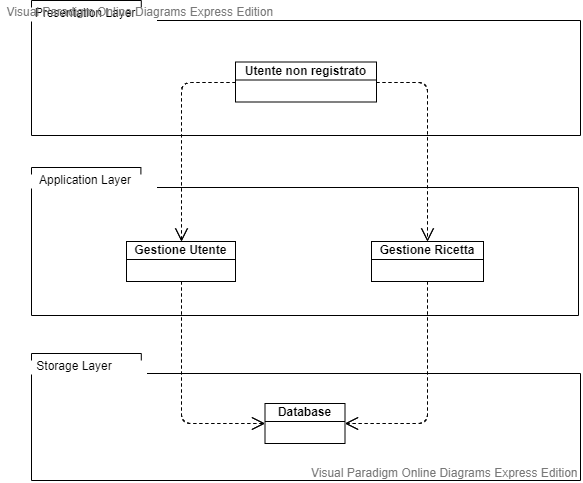
Nel Presentation Layer sono presenti 3 sottosistemi

* Sottosistema Utente: include tutte le interfacce grafiche che gli permettono di effettuare l’accesso, visualizzare la pagina personale e commentare le ricette. Nella sua area personale può, inoltre, creare, modificare o eliminare una ricetta personale
* Sottosistema Amministratore: sono presenti tutte le interfacce grafiche utili alla gestione e alla moderazione degli aspetti del sistema. Nella sua area personale può aggiungere una categoria o ruolo, mentre nelle pagine delle ricette può moderare i commenti.
* Sottosistema Utente non Registrato: include tutte le interfacce grafiche che permettono ad un utente non registrato di registrarsi o di visionare la lista completa delle ricette pubbliche.

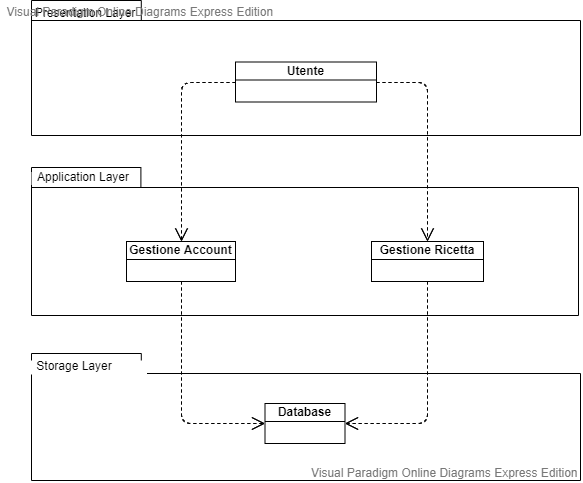
I tre principali sottosistemi individuati nell’Application Layer sono:

* Sottosistema Ricetta: qui sono presenti tutte le classi che servono a comporre, nella maniera più completa possibile, una ricetta. Sono comprese infatti le classi “ricetta”, “categorie” e “ingredienti”, in quanto ogni ricetta dovrà necessariamente appartenere ad una categoria e dovrà avere una serie di ingredienti per consentirne la preparazione.
* Sottosistema Utente: questo sottosistema ha la responsabilità di creare e gestire tutti gli utenti del sito ed è caratterizzato dalle classi “utente” e “ruolo”. La classe ruolo è fondamentale, in quanto consente al sistema di permettere l’accesso a quell’utente a determinate sezioni del sito.
* Sottosistema Amministrazione: nel sottosistema amministrazione sono –aggiungere nuovi ruoli o nuove categorie e moderare i commenti creati dagli utenti, modificandoli o eliminandoli

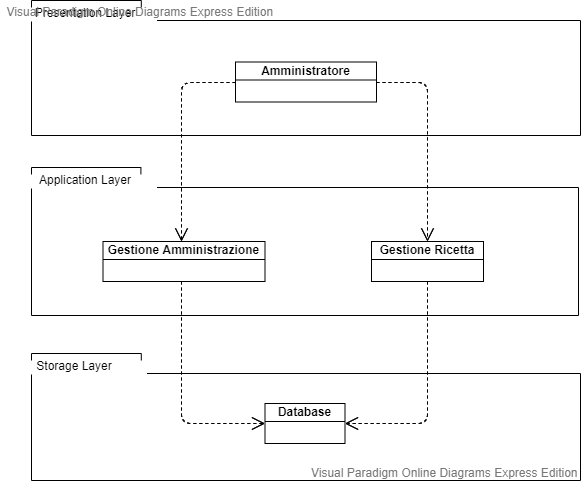
### **Utente Non Registrato**



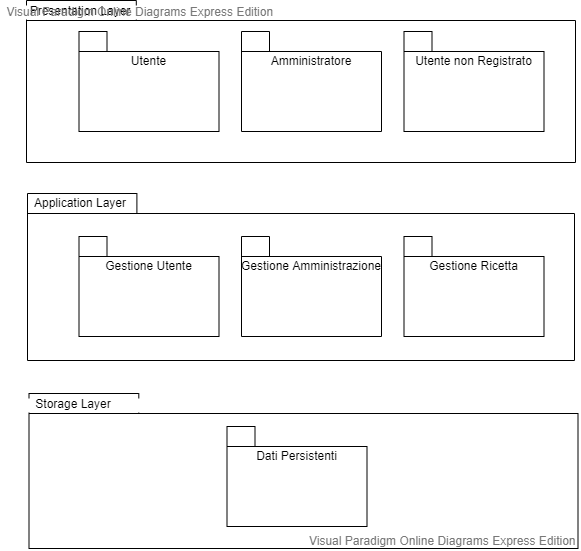
### **Utente**



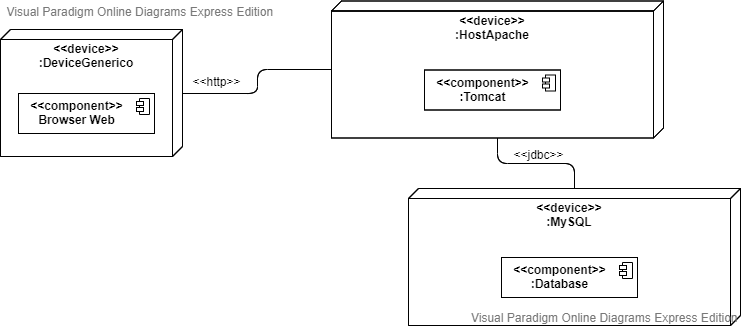
### **Amministratore**



### **Sistema Generale**

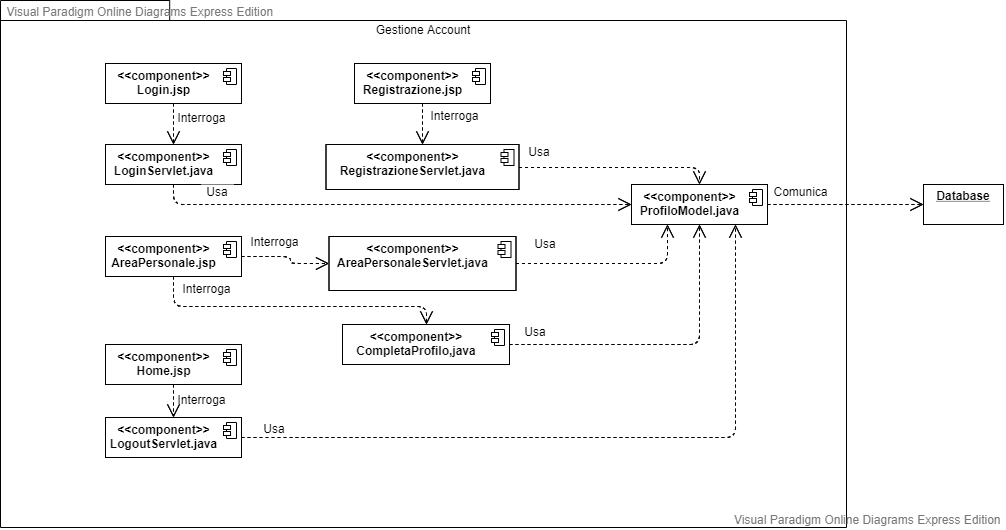


## **Mappatura Hardware/Software**

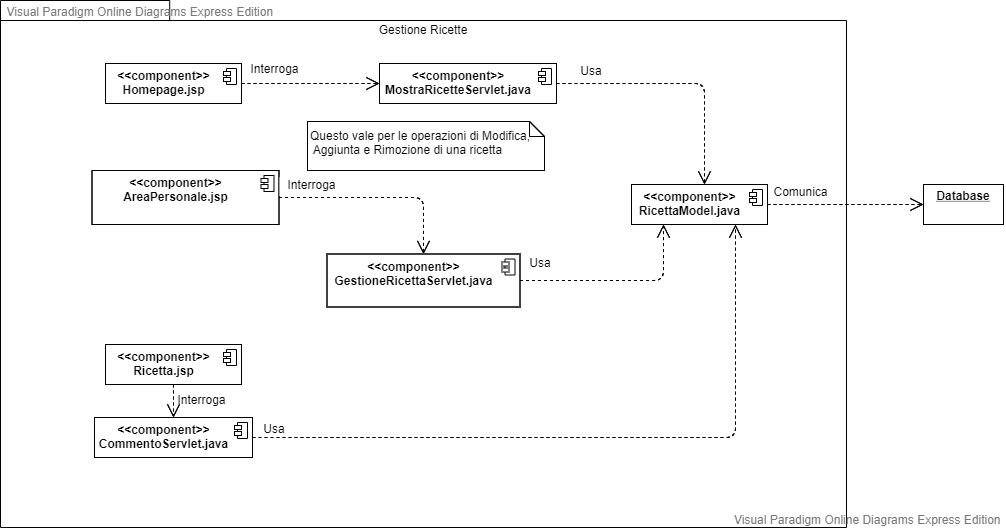


* Nel nostro sistema utilizzeremo un server Apache Tomcat versione 9.0.
* L’utente utilizza il sistema mediante un Browser installato all’interno del suo dispositivo preferito (ad es. Opera, Firefox, Chrome).
* Le funzionalità del sistema, ovvero le Servlet, saranno integrate tramite l’uso del linguaggio Java. Le Servlet poi effettueranno i dovuti reindirizzamenti alle pagine JSP presenti sul server e che saranno visualizzate dai client.
* Il DBMS usato è MySQL, il quale mette a disposizione degli sviluppatori delle API che permettono l’interazione tra sistema e database.

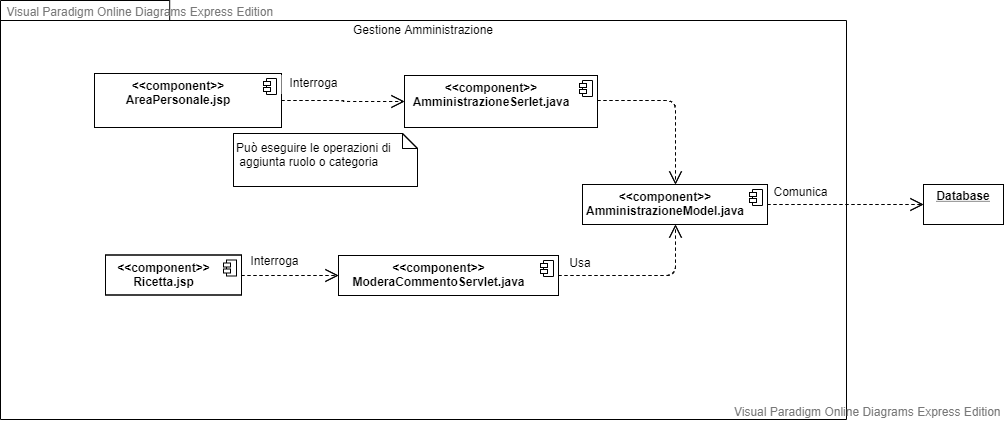
### **Gestione Account**



### **Gestione Ricetta**



### **Gestione Amministrazione**



## **Gestione dati persistenti**

Durante la fase di analisi, abbiamo individuato una serie di entità che devono necessariamente essere rese persistenti per garantire all’utente finale tutte le informazioni di cui ha bisogno per usufruire al meglio del sito web. Le entità candidate per tale operazioni sono:

* Utenti: qui vengono depositate tutte le informazioni riguardanti l’utente, quali i suoi dati d’accesso (email, password) ed i suoi dati personali (username, nome, cognome, ecc.)
* Ricette: la ricetta è la parte fondamentale di tutto il sistema. Essa dovrà contenere tutti i dati utili alla ricerca, visualizzazione, modifica e rimozione della stessa.
* Commenti: ogni ricetta potrà avere un voto e/o un commento, che conterrà il nome dell’autore in modo da poter essere facilmente moderato dall’amministratore
* Categorie: è importante conservare le categorie di ricette per consentire facilitare la creazione di una ricetta all’utente e per consentire la ricerca delle ricette tramite filtro
* Ruolo: ha la responsabilità di conservare tutti i ruoli potenzialmente assumibili da un utente
* Ingredienti: si occupa di rendere persistenti gli ingredienti di cui una ricetta può essere composta. Questa sarà utile al fine di rendere più semplice per l’utente e sicuro per il sistema, che la ricetta non abbia problemi di integrità

Il sito dovrà gestire una elevata quantità di dati, insieme a query complesse per l’ottenimento dei vari attributi. Proprio per questo, useremo un database di tipo relazionale, in modo da sfruttare la sua struttura ad alto livello di astrazione tramite **jdbc** per rendere facile e veloce la gestione di tutti i dati del sito. Dovendo gestire un sistema object-oriented, il DBMS, realizzato in **mysq.**

### **Mapping del database**

Di seguito verrà riportato lo schema logico del database, in cui verranno mostrati gli attributi, i vincoli ed il tipo di chiave per ogni tabella. I valori con \* sono chiavi esterne

**Utente**(idUtente, email, password, nome, cognome, dataDiNascita, telefono, dataDiRegistrazione, ruolo\*)

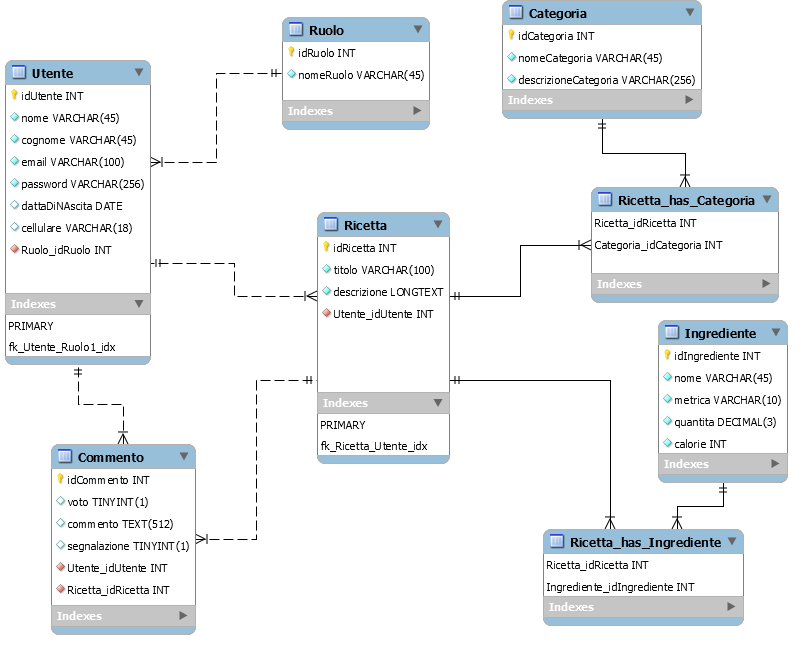
**Ricetta**(IdRicetta, titolo, descrizione, ingredienti\*, pareri\*, autore\*, categorie\*)

**Ingrediente**(IdIngrediente, nome, metrica, quantità, calorie)

**Categoria**(IdCategoria, nome, ricetta\*, descrizione)

**Commento**(IdCommento, voto, commento, segnalazione, autore\*, ricetta\*)

**Ruolo**(IdRuolo, nome, utente\*)



### **Dettaglio struttura tabelle**

#### **Utente**

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Vincoli | Chiave |
| IdUtente | int | NOT NULL | Chiave Primaria |
| Email | Varchar(100) | NOT NULL |  |
| Password | Varchar(256) | NOT NULL |  |
| Nome | Varchar(45) | NOT NULL |  |
| Cognome | Varchar(45) | NOT NULL |  |
| DataDiNascita | Date | NOT NULL |  |
| Telefono | Varchar(18) | NOT NULL |  |
| DataDiRegistrazione | Date | NOT NULL |  |
| Ruolo | int | NOT NULL | Chiave Esterna |

#### **Ricetta**

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Vincoli | Chiave |
| IdRicetta | int | NOT NULL | Chiave Primaria |
| Titolo | Varchar(100) | NOT NULL |  |
| Descrizione | Longtext | NOT NULL |  |
| Ingredienti | int | NOT NULL | Chiave Esterna |
| Pareri | int |  | Chiave Esterna |
| Autore | int | NOT NULL | Chiave Esterna |
| Categorie | int | NOT NULL | Chiave Esterna |

#### **Ingrediente**

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Vincoli | Chiave |
| ID | int | NOT NULL | Chiave Primaria |
| Nome | Varchar(45) | NOT NULL |  |
| Metrica | Varchar(10) | NOT NULL |  |
| Quantità | Decimal(3) | NOT NULL |  |
| Calorie | int | NOT NULL |  |

#### **Categoria**

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Vincoli | Chiave |
| ID | int | NOT NULL | Chiave Primaria |
| Nome | Varchar(45) | NOT NULL |  |
| Ricetta | int | NOT NULL | Chiave Esterna |
| Descrizione | Varchar(256) | NOT NULL |  |

#### **Segnalazione**

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Vincoli | Chiave |
| ID | int | NOT NULL | Chiave Primaria |
| Testo | Varchar(200) | NOT NULL |  |
| Origine | int | NOT NULL | Chiave Esterna |

#### **Commento**

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Vincoli | Chiave |
| IDCOMMENTO | int | NOT NULL | Chiave Primaria |
| commento | Text05121) |  |  |
| voto | tinuInt(1) |  |  |
| segnalazione | tiniInt(1) |  |  |
| Autore | int | NOT NULL | Chiave Esterna |
| Ricetta | int | NOT NULL | Chiave Esterna |

#### **Ruolo**

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Vincoli | Chiave |
| ID | int | NOT NULL | Chiave Primaria |
| Nome | Varchar(45) | NOT NULL |  |
| Utente | int | NOT NULL | Chiave Esterna |

## **Controllo accessi e sicurezza**

Nel sistema RecipeManager sono presenti diversi attori, ma non tutti possono eseguire le stesse operazioni. A questo proposito, si è reso necessario descrivere in una tabella degli accessi tutte le operazioni che ogni attore può eseguire:

### **ACL**

|  |  |  |  |
| --- | --- | --- | --- |
| Oggetti\Attori | Utente Registrato | Visitatore | Manager |
| Utente | * Login * Logout * Completa Profilo * Visualizza Profilo | * Registrazione | * Login * Logout * Visualizza Profilo |
| Ricetta | * Nuova Ricetta * Visualizza lista Ricette * Visualizza Dettaglio Ricetta | * Visualizza lista Ricette * Visualizza Dettaglio Ricetta | * Visualizza lista Ricette * Visualizza Dettaglio Ricetta * Elimina Ricetta |
| Commento | * Nuovo Parere * Visualizza Commenti | * Visualizza Commenti | * Visualizza Commenti * Cancella Commenti * Modifica Commenti |

## **Controllo del software globale**

Il controllo del flusso software viene gestito da classi JavaServlet che interagendo con il client, il quale si interfaccia tramite un web browser, svolgono le varie operazioni. Il server smista ogni nuova richiesta alle classi JavaServlet, inoltrando poi la risposta al client

### **Condizione dei boundary**

Le condizioni limite riguardano l’accensione e lo spegnimento del sistema per quanto riguarda il lato Server. Dal lato Client si riferiscono agli errori di connessione al server.

#### Avvio del sistema

Per lo start-up del sistema è necessario avviare il web server. In seguito, tramite l'interfaccia di Login, sarà possibile autenticarsi tramite opportune credenziali (useraname e passowrd) come Amministratore, Operatore e Cliente.

#### **Terminazione del sistema**

Al momento della corretta chiusura dell’applicazione, si ha la terminazione del sistema con un regolare Logout. Per terminare il server è necessario l’intervento dell’amministratore. Questo dovrà effettuare la procedura di terminazione dopo la quale non sarà possibile per nessun client connettersi al sistema.

#### **Fallimento del sistema**

Nel caso si verifichi un errore dovuto all’hardware o al software si cercherà di ripristinare una configurazione del sistema precedente allo stato d’errore. Visto che i dati sono gestiti dal DBMS non c’è rischio di perderli. Tuttavia, se si verifica un guasto al supporto di memorizzazione si può avere una perdita di dati.

## Servizi dei sottosistemi

In questa sezione vengono descritti tutti i servizi che ogni sottosistema può offrire. Nella sezione 3.2 abbiamo individuato tre sottosistemi molto importanti: utente, ricetta e parere. Vediamoli nel dettaglio:

### **Sottosistema Gestione Utente**

|  |  |  |
| --- | --- | --- |
| **Sottosistema** | Gestione Utente | |
| Descrizione | Questo sottosistema si occupa di gestire tutti servizi legati all’interazione con l’utente | |
| Servizi offerti | |  |
| **Servizio** | Descrizione | |
| Login | Servizio che permette all’utente di effettuare il login al suo profilo personale | |
| Logout | Servizio che permette all’utente di disconnettere il proprio profilo dal sistema | |
| Registrazione | Servizio che permette all’utente di registrarsi al sistema | |
| **Completa** **Profilo** | Servizio che permette all’utente di scrivere informazioni aggiuntive al proprio profilo | |
| **Modifica** **Profilo** | Servizio che permette all’utente di modificare i dati del suo profilo | |

### **Sottosistema Gestione Ricetta**

|  |  |  |
| --- | --- | --- |
| **Sottosistema** | **Gestione Ricetta** | |
| **Descrizione** | Questo sottosistema si occupa di gestire tutti servizi che si occupano della creazione, visualizzazione e rimozione di una ricetta | |
| **Servizi offerti** | |  |
| **Servizio** | **Descrizione** | |
| Crea Ricetta | Servizio che permette all’utente registrato di creare una nuova ricetta, facendogli inserire titolo, ingredienti e modalità di preparazione | |
| Visualizza lista ricette | Servizio che permette all’utente di visualizzare la lista completa di tutte le ricette pubbliche | |
| Visualizza dettaglio ricetta | Servizio che permette all’utente di visualizzare la pagina completa di una ricetta | |
| Filtra Ricetta | Servizio che permette all’utente di filtrare le ricette in base a parametri specifici | |
| Elimina ricetta | Servizio che permette all’utente di eliminare una propria ricetta personale | |

### **Sottosistema Gestione Amministrazione**

|  |  |  |
| --- | --- | --- |
| **Sottosistema** | Gestione Amministrazione | |
| **Descrizione** | Questo sottosistema si occupa di gestire tutti gli aspetti amministrativi, quali la modifica o l’eliminazione dei commenti, e l’aggiunta di nuovi ruoli o categorie | |
| **Servizi offerti** | |  |
| **Servizio** | **Descrizione** | |
| **Inserisci Ruolo** | Servizio che permette all’utente di assegnare un voto ad una ricetta | |
| Inserisci Categoria | Servizio che permette all’utente di inserire un commento in una ricetta | |
| Modifica Commento | Servizio che permette all’amministratore di modificare un commento per renderlo più rispettoso nei confronti dell’autore della ricetta | |
| Elimina Commento | Servizio che permette all’amministratore di eliminare un commento nel caso non voglia più renderlo pubblico | |
|  |  | |

## **Condizioni Baundary**

### **Scenari**

|  |  |
| --- | --- |
| **Nome Scenario** | Startup Sistema |
| **Istanze di Attori Partecipanti** | Paolo: Amministratore |
| **Flusso di Eventi** | 1. Paolo decide di voler avviare il sistema e quindi clicca sul pulsante “Avvia”. 2. Il sistema, con le opportune procedure di avvio, attiva i server e i relativi servizi in remoto rendendosi disponibile ad eventuali richieste. 3. Il sistema notifica il successo della procedura. |

|  |  |
| --- | --- |
| **Nome Scenario** | Shut-down Sistema |
| **Istanze di Attori Partecipanti** | Paolo: Amministratore |
| **Flusso di Eventi** | 1. Paolo decide di voler arrestare il sistema e quindi accede alla pagina dedicata e clicca sul pulsante “Arresta”. 2. Il sistema effettua una scansione per verificare se ci sono ancora richieste in sospeso. 3. Il sistema porta a termine le eventuali richieste in sospeso 4. Tramite le opportune procedure di arresto il sistema disattiva i servizi in remoto e il server. 5. Il sistema notifica il successo della procedura. |

### **Casi d’uso**

|  |  |
| --- | --- |
| **ID** | Startup |
| **Nome Caso Uso** | Startup Server |
| **Istanze di Attori Partecipanti** | Amministratore |
| **Condizione di Entrata** | L'amministratore visualizza la console per effettuare l’avvio del sistema. |
| **Flusso di Eventi** | |  |  | | --- | --- | | Utente | Sistema | | L’amministratore clicca sul pulsante “Avvia” |  | |  | Il sistema attiva i propri servizi rendendosi disponibile in tutte le sue funzionalità. | |

## Glossario

**Client**: componente che accede ai servizi o alle risorse di un'altra componente, detta server.

**Deployment Diagram**: Schema che descrive la struttura dinamica del sistema

**DBMS**: programma informatico (o, più frequentemente, un insieme di programmi) progettato per gestire un database, ovvero un insieme di numerosi dati strutturati. Le operazioni, normalmente, sono richieste da un gran numero di utenti.

**Form**: finestra di dialogo incorporata in una pagina Web che consente all'utente di inserire informazioni destinate ad un server. Generalmente richiede un programma sul server che si occupi di esaminare le informazioni inviate. E’ composto da spazi (campi) predefiniti, ad esempio menù a tendina, elenchi puntati o caselle di testo libero.

**JDBC**: API per il linguaggio di programmazione Java che serve ai client per connettersi a un database. Fornisce metodi per interrogare e modificare i dati. È orientata ai database relazionali.

**Login**: Procedura attraverso la quale ci si collega con un qualsiasi servizio in linea. All'utente viene assegnato un nome di login ed una password che vengono richiesti dal sistema ogni volta che ci si collega.

**Layer**: E’ un insieme di classi con funzionalità simile (tipicamente raggruppati in un unico

package).

**Logout**: Operazione attraverso la quale si termina un collegamento con un sistema al quale

si ha accesso attraverso un nome utente e una password (vedi login).

**MySQL**: Database management system relazionale, composto da un client con interfaccia a caratteri e un server, disponibile su molte piattaforme.

**Password**: È un metodo di sicurezza che, mediante una stringa di caratteri, permette di identificare un utente specifico. Generalmente le password sono formate da una sequenza di lettere e numeri; digitando correttamente questi caratteri, si può avere accesso al computer o alla rete.

**Package**: Un Package rappresenta una collezione di classi ed interfacce che possono essere raggruppate in base alla funzione comune da esse svolta

**Server**: Programma di gestione di un servizio che invia informazioni in un particolare formato ricevuto e interpretato da un programma Client dal lato ricevente.

**Shut-down**: fase di terminazione del sistema.

**Start-up**: processo di accensione e di avvio di un computer, di un dispositivo o di un sistema.