Università di Salerno

Corso di Ingegneria del Software

ClickEat

System Design Document

Versione 1.0

Progetto: ClickEat	Versione: 1.0
Documento: System design document	Data: 21/12/2018

Partecipanti:

Nome	Matricola
Cupito Andrea [CA]	0512104538
Amoriello Luca [AL]	0512104658
Pasquariello Giovanni [PG]	0512105020
Russo Vincenzo [RV]	0512104130

Scritto da:	Cupito Andrea

Revision History

Data	Versione	Descrizione	Autore
12/10/2018	1.0	Stesura del System design document	Membri del gruppo

Sommario

1 INTRODUZIONE

1.1Scopo del sistema

L'obiettivo del sistema è quello di informatizzare il sistema di ordinazione di un ristorante, sostituendo il metodo cartaceo, con un sistema in grado di aumentare l'efficienza in termini di tempo e di risorse di un'ordinazione prevedendo un tablet al post del classico blocco note e in ogni reparto (cassa compresa) un terminale in sostituzione al "mucchio di carta" per ogni ordinazione. Per efficienza si intende oltre a quella del sistema in sé, anche la riduzione del tempo di consegna della comanda in ogni reparto e la diminuzione del tempo di calcolo della somma del conto.

1.2 Obiettivi di design

Il sistema ClickEat deve essere semplice e intuitivo nell'utilizzo quotidiano da tutti coloro che interagiscono con il sistema. Inoltre, il sistema deve essere efficiente, garantendo tempi di risposta brevi per ogni funzionalità richiesta e tollerante agli errori, grazie a particolari politiche di controllo e prevenzione. In più, il sistema verrà progettato per poter integrare altre funzionalità nel modo più semplice possibile. Il tutto è fondamentale per far sì che il sistema possa essere integrato con semplicità in qualsiasi attività di ristoro.
Tutti questi obiettivi di design possono essere racchiusi in quattro categorie:
Performance, Affidabilità, Manutenzione ed Utente Finale.

1.2.1 Criteri di Performance

Tempo di risposta	Il sistema deve essere in grado di gestire le richieste nel minor tempo possibile. La comunicazione del sistema non è basata sulla connessione ad Internet; quindi non soffre di problematiche riguardanti la velocità di connessione.
Memoria	Il sistema sarà in grado di supportare la quantità di dati da memorizzare; è presumibile che le informazioni relative ai ristoranti non siano tantissime.
Throughput	Il sistema deve essere in grado di gestire più richieste contemporaneamente. Infatti, è prevedibile che più utenti interagiscano concorrentemente a causa dell'affluenza di clienti in un ristorante.

1.2.2 Criteri di Affidabilità

Sicurezza	Il sistema sarà utilizzato solo da utenti opportunamente loggati. Sarà l'amministratore a registrare nuovi utenti nel sistema. ClickEat, prevede il controllo degli accessi e dell'autenticazione, in modo da evitare accessi indesiderati al sistema. Inoltre, il sistema non sarà reso pubblico; sarà possibile utilizzarlo soltanto in locale, ciò non permetterà ad utenti esterni di accedervi.
Tolleranza all'errore	ClickEat deve tollerare gli errori interni al sistema, continuando ad essere operativo come se nulla fosse accaduto; evitando così problemi in sala durante gli orari lavorativi. Ciò verrà implementato tramite un basso accoppiamento tra i sottosistemi, evitando così che un errore in uno dei sopracitati si propaghi in tutto il sistema generando problematiche più gravi. Per quanto concerne problemi hardware dei terminali, si è ideato di installare due terminali: uno alla cassa, uno nella zona di appoggio dei camerieri; in questo modo sarà possibile utilizzare i terminali da due postazioni, e nel caso di malfunzionamento di uno dei due terminali sarà possibile continuare ad usufruire dell'applicazione dal terminale ancora on-line.
Robustezza	Il sistema deve essere in grado di gestire input errati da parte dell'utente, in modo da evitare spiacevoli conseguenze. Ciò verrà implementato mediante appositi controlli sia lato client che lato server.

1.2.3 Criteri di Manutenzione

Estendibilità	Il sistema deve prestarsi a modifiche future, soprattutto se volte ad estendere quest'ultimo. È necessario che il sistema venga strutturato in maniera ordinata, in appositi moduli, per rendere semplice la comprensione anche a terzi, ignari della codifica del sistema.
Leggibilità	Il codice relativo al sistema dovrà essere quanto più leggibile possibile, per permettere una comprensione veloce al fine di modifiche future. Il codice verrà opportunamente formattato (indentazione) e il tutto verrà accompagnato da commenti utile per una maggiore comprensione.

1.2.4 Criteri per l'utente finale

Usabilità	Il sistema sarà semplice nel suo utilizzo, in modo da rendere l'interazione tra utente e sistema piacevole e mai forzata. Ricordiamo che il software deve semplificare la gestione di un ristorante, quindi è opportuno che le funzionalità offerte devono essere sempre chiare e intuibili.

1.3 Definizioni, acronimi e abbreviazioni

ClickEat: Nome del sistema da sviluppare
RAD: Requirement Analysis Document
DBMS: Database Management System

1.4 Riferimenti

Requirement Analysis Document ClickEat

1.5 Panoramica

Forniamo una panoramica sulle attività affrontate nel system design e documentate attraverso il system design document (SDD).
Ciò di cui discuteremo è il fondamento dell'architettura software del prodotto.

- Decomposizione del sistema, in cui il sistema viene partizionato in diversi sottosistemi i quali, a loro volta, forniscono servizi ad altri sottosistemi. L'insieme dei servizi sarà denominato interfaccia.
- Mapping hardware/software, riguardante la problematica di mappare gli oggetti e le associazioni in apposite componenti reali.
- Politiche di accesso e sicurezza, fondamentale passo del documento, indicante come vengono gestiti gli accessi ai vari sottosistemi da parte degli utenti del sistema.
- Gestione dei dati persistenti, riguarda l'individuazione dei dati persistenti e l'infrastruttura da scegliere per memorizzare tali dati.
- Boundary Conditions, riguardano la determinazione della configurazione, avvio, terminazione del sistema.

2 Architettura software proposta

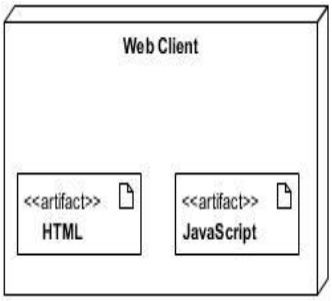
2.1 Panoramica

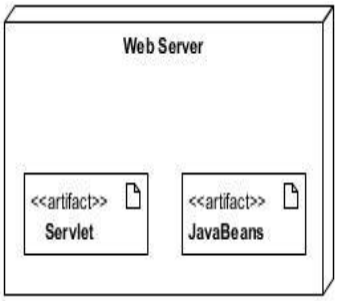
L'architettura del sistema ClickEat è di tipo Client/Server. Il Server riceve la richiesta da parte del client, e risponde con le informazioni richieste, in tempo utile.
I motivi che giustificano tale scelta sono:

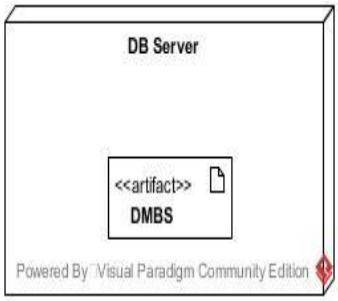
- Portabilità: il server potrà essere installato su una varietà di macchine e sistemi operativi, inclusi dispositivi fissi e mobili.
- Performance: il client dovrà essere in grado di supportare task display-intensive e il server dovrà fornire operazione CPU-intensive.
- Flessibilità: per ogni tipologia di utente connesso il sistema dovrà mostrare l'interfaccia dedicata ad esso, tramite la quale l'utente potrà eseguire le operazioni ad esso concesse.
- Affidabilità: sia il client che il server dovranno garantire la permanenza dei dati in seguito ad eventuali guasti, sarà possibile effettuare il backup dei dati.

Il sistema è strutturato su tre livelli (three-tier):

- Interface Layer
- Application Logic Layer
- Data Layer





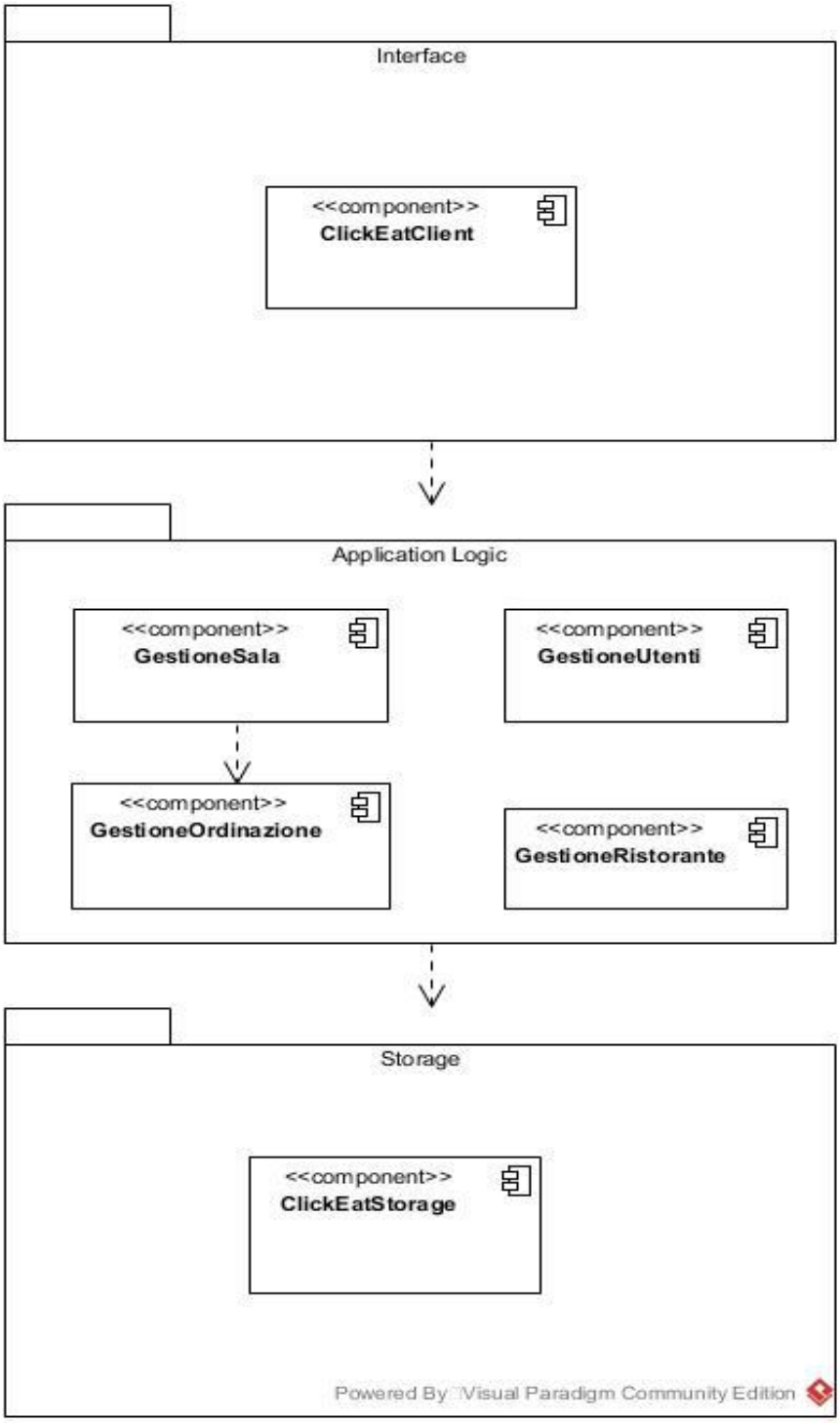


Interface Layer rappresenta l'interfaccia che permette all'utente di interagire con i sistemi sottostanti; ricopre il ruolo di client utilizzando il browser per richiedere pagine web al server centrale.

Application Logic Layer ha il compito di verificare la correttezza delle operazioni e inviare i dati al client; spesso interroga il Database per accedere ai dati persistenti del sistema.

Data Layer ha la funzione di custodire e memorizzare i dati persistenti, tramite l'utilizzo di un DBMS. Inoltre, è responsabile della comunicazione con l'Application Logic Layer per fornire a quest'ultimo i dati richiesti, tramite l'interazione con il Database.

2.2 Decomposizione in sottosistemi



Lo scopo di questa attività è quella di decomporre il sistema in diversi sottosistemi individuali, a partire dai requisiti funzionali e dal modello di analisi.
Il sistema presenta uno stile architetturale three-tier in cui il sottosistema ClickEatClient si occupa del front end per gli utenti, con i relativi oggetti utili a soddisfare i casi d'uso. ClickEatServer è il sottosistema responsabile del controllo degli accessi e delega ai sottostanti sottosistemi la logica applicativa. L'ultimo tier è composto dal sottosistema ClickEatStorage che si occupa della memorizzazione dei dati persistenti.

Interface	
ClickEatClient	Tale sottosistema di occupa del front end del sistema per il cameriere, l'addetto alla casse e l'amministratore. (Trattasi delle diverse tipologie di utente registrato)

Application Logic	
Gestione Ordinazione	Tale sottosistema si occupa della gestione delle ordinazioni effettuate dai clienti presenti nel ristorante. Nello specifico permette di: creare un'ordinazione, aggiungere e/o rimuovere piatti dalle ordinazioni esistenti, aggiungere e/o eliminare gli ingredienti per i vari piatti presenti, visualizzare un riepilogo relativo ad un'ordinazione selezionata, inviare le ordinazioni ad altri terminali del sistema.
Gestione Utente	Tale sottosistema si occupa della gestione degli utenti, fra cui: controllo delle credenziali utente per il login, logout di un utente, registrazione di un nuovo utente al sistema, rimozione di un utente registrato.
Gestione della Sala	Tale sottosistema si occupa della gestione della sala, fra cui: visualizzare lo stato della sala in base allo stato dei tavoli presenti, richiesta del conto per un determinato tavolo, stampa del conto del tavolo con conseguente liberazione del tavolo sopracitato.
Gestione Ristorante	Tale sottosistema si occupa della gestione del ristorante, fornendo servizi per: visualizzare lo storico degli ordini (giornaliero, settimanale, mensile, annuale), aggiungere un nuovo piatto al menu, rimuovere un piatto dal menu, modificare un piatto presente nel menu, aggiungere e rimuovere un Tavolo da sistema.

Storage	
ClickEat Storage	Tale sottosistema si occupa della memorizzazione dei dati persistenti del sistema.

2.3 Mappatura Hardware/Software

La nostra applicazione rispecchia il pattern MVC (Model, View, Controller) definiti nel seguente modo:

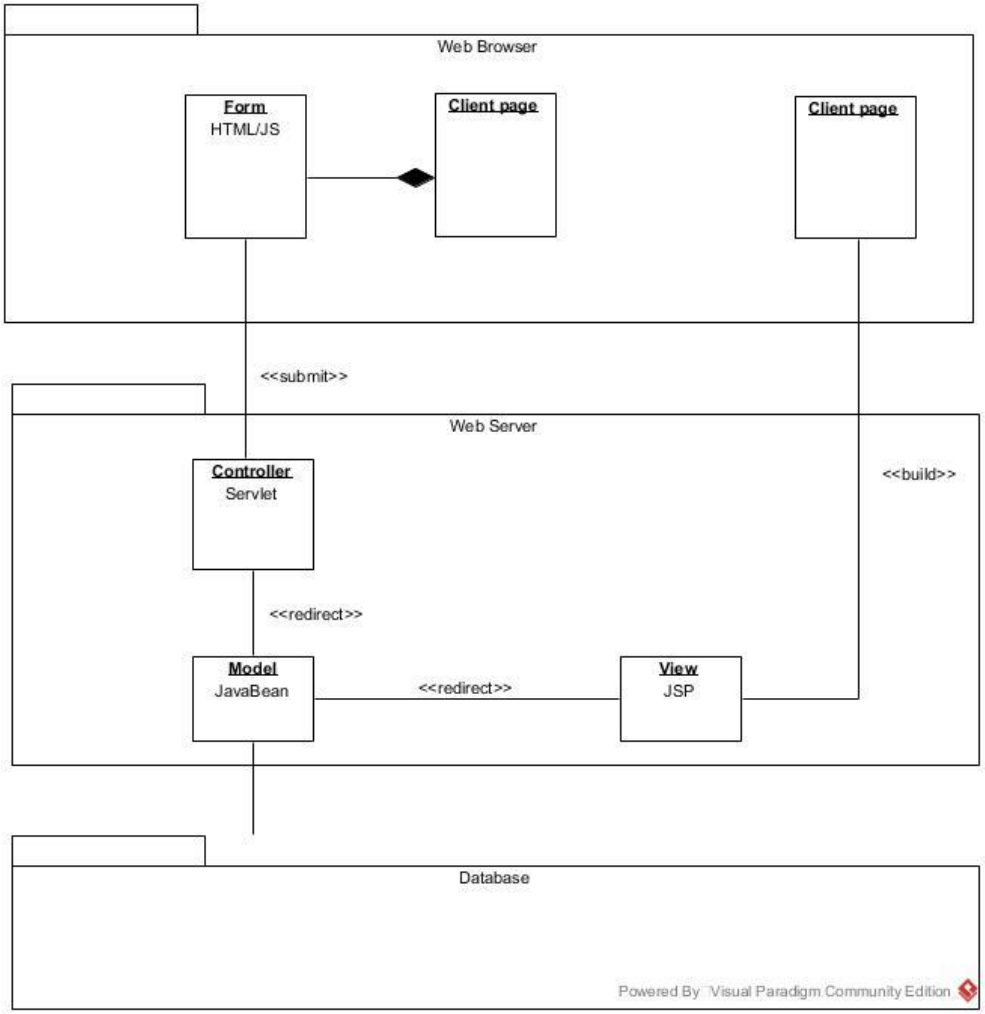
- Model: sono le componenti atte a rappresentare i dati e la logica del problema.
- View: è l'interfaccia con cui l'utente opera.
- Controller: rappresenta la logica per la gestione del flusso di controllo e le operazioni sul sistema.

Una buona progettazione del sistema deve interessare ciascuno dei tre livelli citati. Tuttavia, il semplice UML non ci permette di rappresentare un'applicazione web in modo astratto (e completo); per ovviare a tale problematica utilizzeremo le Web Application Extension (WAE), definite per la prima volta da Conallen, le quali ci permetteranno di rappresentare:

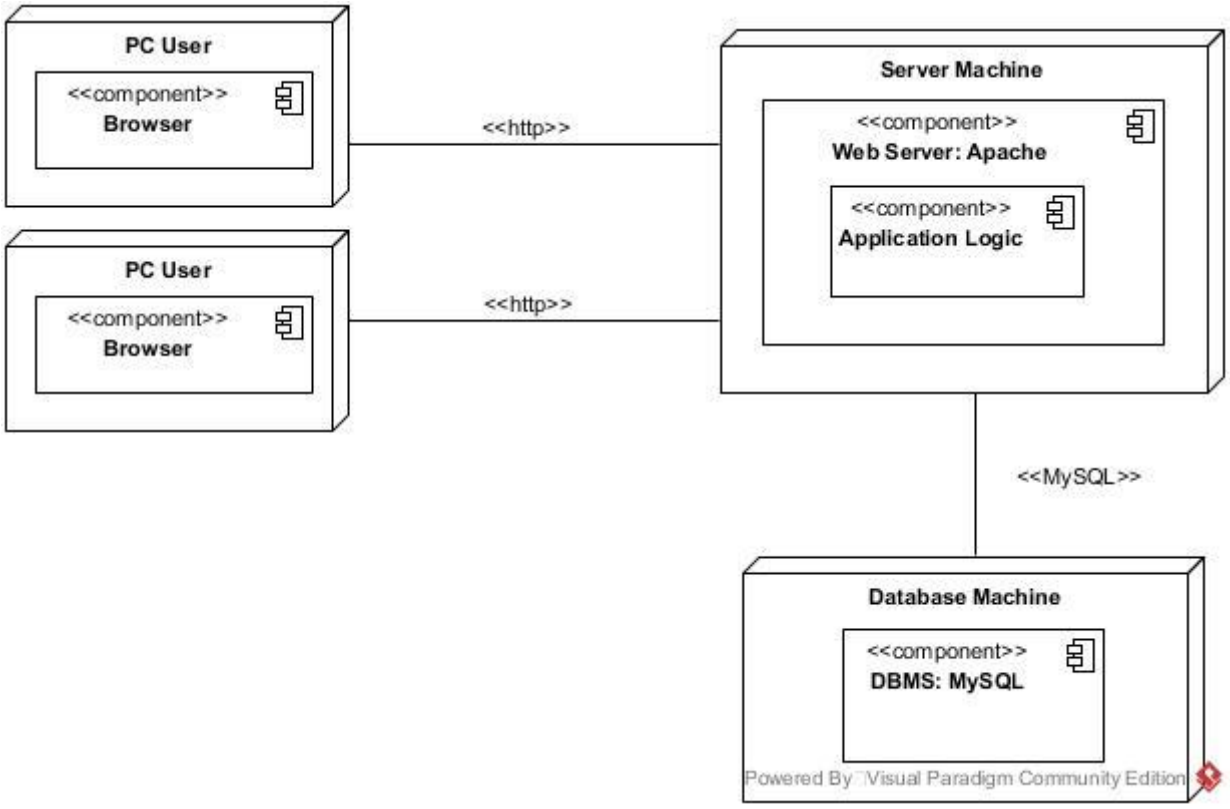
- Pagine Client
- Pagine Server
- Form
- Le associazioni tra le varie pagine (build, submit, ecc.)

Di seguito vi è una rappresentazione tra le varie componenti del sistema a Compile Time attraverso un Component Diagram. Successivamente sarà presentato un Deployment Diagram, utile per mostrare la creazione dinamica delle pagine HTML e della Logica Applicativa, rispettando ovviamente il pattern MVC.

Component Diagram:



Deployment Diagram:
Nel sistema proposto i Client sono Web Browser, mentre il Web Server contiene tutte le componenti che si occupano della logica applicativa, dell'interfaccia e dei dati. Per realizzare le componenti del Client vengono utilizzare apposite JSP.
Il Web Server invia query al Database MySQL utilizzando il protocollo MySQL.

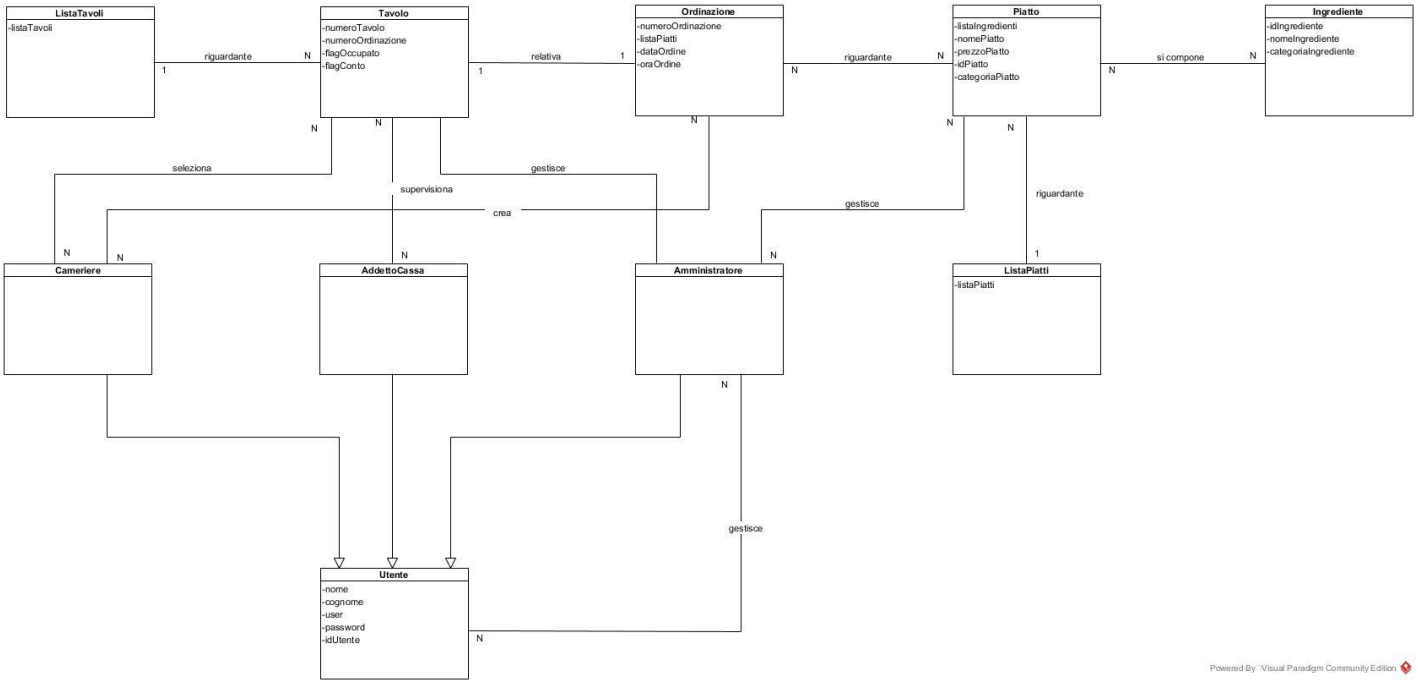


2.4 Gestione dei dati persistenti

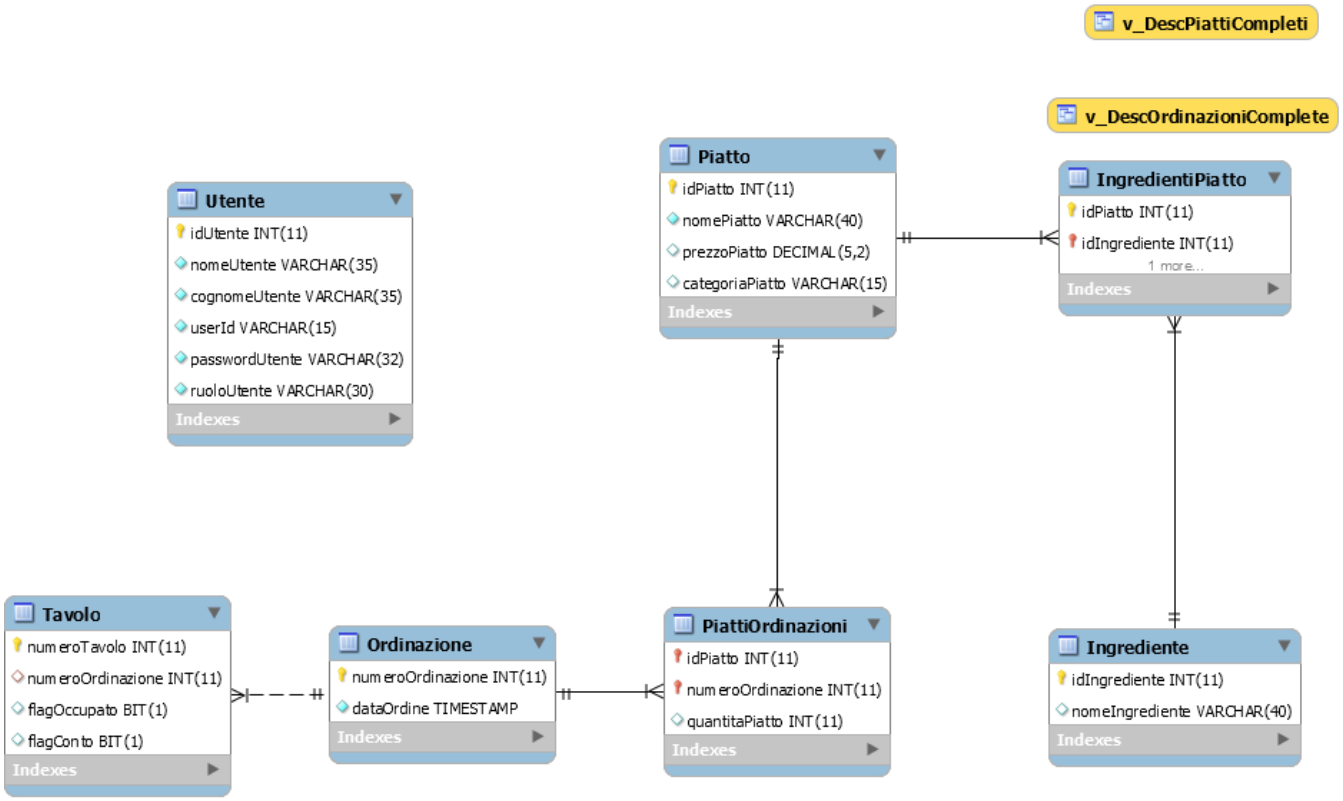
Per la memorizzazione dei dati si è scelto di utilizzare un Database relazionale che tramite il concetto di tabelle e relazioni permette di gestire in modo ottimale l'archiviazione delle informazioni di varie entità del sistema e soprattutto permette una facile interrogazione per poter ottenere le informazioni.
Le tecnologie utilizzate sono, per quanto riguarda il linguaggio di interrogazione, MySql nella versione 8.0, mentre è stato scelto il Database Management System PhpMyAdmin alla versione 4.8.5.
Si è scelto di utilizzare un DBMS poiché, grazie ad un'interfaccia utente semplice da utilizzare, è utilizzabile anche da amministratori non molto pratici nella gestione dei dati.
Un DBMS inoltre garantisce:

- Sicurezza: grazie ad un sistema di autorizzazioni che per ogni utente stabilisce permessi di lettura, scrittura e modifica.
- Integrità: ovvero la gestione di più utenti contemporaneamente, impedendo, ad esempio, la modifica contemporanea di un dato.
- Supporto alle interrogazioni: permette di usare linguaggi di interrogazione mediante query per poter prelevare, aggiornare o inserire i dati.

Class Diagram:



- **Utente:** Questa classe contiene tutte le informazioni relative ad un utente registrato. La generalizzazione chiarisce il ruolo dell'utente che può variare: Cameriere, AddettoCassa, Amministratore.
- **Tavolo:** Questa classe contiene tutte le informazioni relative ad un tavolo presente nel sistema. L'aggiunta di quest'ultimo è effettuata da un amministratore del sistema, legato tramite la relazione "gestisce". Attraverso la relazione "seleziona", un tavolo viene collegato ad un Cameriere, il quale può selezionarlo quando necessita di focalizzarsi su un Tavolo specifico. Un tavolo si compone anche di un apposito "flag", utile per segnalare lo stato di quest'ultimo in:
    - **True:** il tavolo è occupato.
    - **False:** il tavolo è libero.
- **Ordinazione:** Questa classe contiene tutte le informazioni relative ad un'ordinazione presente nel sistema. La creazione di quest'ultima è affidata all'utente cameriere, esplicitata mediante la relazione "crea".
- **Piatto:** Questa classe mantiene tutte le informazioni relative ad un Piatto presente nel sistema. Ogni Piatto ha una relazione con gli ingredienti ("si compone") utile per comprendere l'insieme di ingredienti che compongono il singolo piatto.
- **Ingrediente:** Questa classe mantiene tutte le informazioni relative ad un Ingrediente presente nel sistema.
- **ListaTavoli:** Questa classe mantiene un elenco di tutti i tavoli presenti nel sistema, per rendere più agevole le operazioni in cui è utile avere una panoramica di quest'ultimi. Tale classe è soltanto indicativa, essa non verrà mappata all'interno del DataBase.
- **ListaPiatti:** Questa classe mantiene un elenco di tutti i piatti presenti nel sistema, per rendere più agevole le operazioni in cui è utile avere una panoramica di quest'ultimi. Tale classe è soltanto indicativa, essa non verrà mappata all'interno del DataBase.



Dopo aver illustrato la struttura dello storage mediante un modello ad entità, procediamo con la descrizione delle singole entità partecipanti:

NOTA: Le entità "IngredientiPiatto" e "PiattiOrdinazioni" sono utili per mappare la relazione molti a molti tra "Ingrediente" e "Piatto" (nel primo caso) e "Ordinazione" e "Piatto" (nel secondo caso). Inoltre, l'entità "Utente" presenta l'attributo "ruoloUtente" per specificare il preciso ruolo all'interno del sistema; tale scelta (detta MAPPING VERTICALE) è stata utilizzata per mappare la generalizzazione degli utenti (vedere Class Diagram).

UTENTE	
Campo	**Vincolo**
idUtente	Int auto_increment, primary key
nomeUtente	Lunghezza massima: 35 caratteri, not null
cognomeUtente	Lunghezza massima: 35 caratteri, not null
passwordUtente	Lunghezza massima: 32 caratteri, not null
ruoloUtente	Lunghezza massima: 30 caratteri, not null
userId	Lunghezza massima: 15 caratteri, unique, not null

PIATTO	
Campo	**Vincolo**
idPiatto	Int auto_increment, primary key
nomePiatto	Lunghezza massima: 40 caratteri, not null
prezzoPiatto	Lunghezza massima: 5 decimali a sinistra della virgola e 2 decimali a destra della virgola, not null
categoriaPiatto	Lunghezza massima: 15 caratteri, not null

INGREDIENTE	
Campo	**Vincolo**
idIngrediente	Int auto_increment, primary key
nomeIngrediente	Lunghezza massima: 40 caratteri, not null

TAVOLO	
Campo	**Vincolo**
numeroTavolo	Int auto_increment, primary key
numeroOrdinazione	Int, foreign key {ORDINAZIONE}
flagOccupato	Bit
flagConto	Bit

ORDINAZIONE	
Campo	**Vincolo**
numeroOrdinazione	Int auto_increment, primary key
dataOrdine	TIMESTAMP, not null

PIATTIORDINAZIONI	
Campo	**Vincolo**
idPiatto	Int auto_increment, foreign key {PIATTO}, primary key
numeroOrdinazione	Int auto_increment, foreign key {ORDINAZIONE}, primary key
quantitàPiatto	Int, default = 1

INGREDIENTIPIATTO	
Campo	**Vincolo**
idPiatto	Int auto_increment, foreign key {PIATTO}, primary key
idIngrediente	Int auto_increment, foreign key {INGREDIENTE}, primary key

Per alleggerire il carico delle query, abbiamo creato delle viste che ci permettessero di ottenere i dati richiesti senza dover interrogare entità multiple e, quindi, aumentare la complessità delle interazioni; precisiamo che tali viste non verranno utilizzate per effettuare operazioni di UPDATE. Procediamo con la loro descrizione:

- V_DescPiattiCompleti: vista utile per ottenere in maniera compatta i piatti con i relativi ingredienti.

V_DescPiattiCompleti	
Campo	**Vincolo**
idPiatto	Int auto_increment
nomePiatto	Lunghezza massima: 40 caratteri
prezzoPiatto	Lunghezza massima: 5 decimali a sinistra della virgola e 2 decimali a destra della virgola
categoriaPiatto	Lunghezza massima: 15 caratteri
Lista_ingredienti	Text

- V_DescOrdinazioniComplete: vista utile per ottenere in maniera compatta il riepilogo completo di un ordinazione.

V_DescOrdinazioniComplete	
Campo	**Vincolo**
dataOrdine	TIMESTAMP
numeroTavolo	Int auto_increment
numeroOrdinazione	Int auto_increment
idPiatto	Int auto_increment
nomePiatto	Lunghezza massima: 40 caratteri
Lista_ingredienti	Text
prezzoPiatto	Lunghezza massima: 5 decimali a sinistra della virgola e 2 decimali a destra della virgola
CategoriaPiatto	Lunghezza massima: 15 caratteri
quantitàPiatto	Int

2.5 Controllo degli accessi e sicurezza

ClickEat è un sistema multiutente, quindi attori differenti hanno accesso a servizi differenti. Per schematizzare al meglio il concetto di "accesso" alle operazioni abbiamo utilizzato una matrice degli accessi, la quale mostra le operazioni disponibile in base all'utente selezionato.

ATTORI OGGETTI	CAMERIERE	ADDETTO ALLA CASSA	AMMINISTRATORE
UTENTE	Login, Logout.	Login, Logout.	Login, Logout, Crea Account, Cancella Account, Modifica Account.
ORDINAZIONE	Crea Ordinazione, Modifica Ordinazione, Cancella Ordinazione.	Visualizza, Stampa conto.	Visualizza.
PIATTO	Aggiunta piatto ad Ordinazione, Rimozione piatto da Ordinazione, Modifica piatto presente in un Ordinazione.	Visualizza.	Aggiunta piatto al sistema, Rimozione piatto dal sistema, Modifica Piatto dal sistema.
TAVOLO	Seleziona tavolo.	Modifica stato.	Aggiungi al sistema, Rimuovi dal sistema.
INGREDIENTE	Aggiunta di ingredienti in un piatto presente in un Ordinazione, Rimozione di ingredienti da un piatto presente in un Ordinazione	Visualizza.	Aggiunta di ingredienti nel sistema, Rimozione ingrediente dal sistema.

2.6 Controllo globale del software

Il controllo del flusso software verrà gestito da apposite Servlet che interagendo con il client svolgeranno le varie operazioni. Il server smisterà ogni nuova richiesta alla Servlet apposita, inoltrando poi la risposta al client.

2.7 Boundary Conditions

In questa fase determiniamo l'accensione e lo spegnimento del sistema per quanto riguarda il lato Server.

Scenari:

Nome Scenario	Startup Server
Attori	Amministratore: Marco
Flusso di Eventi	1. Marco 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 l'avvenuto avvio con una notifica.

Nome Scenario	Shutdown Server
Attori	Amministratore: Marco
Flusso di Eventi	1. Marco decide di voler arrestare il sistema e quindi clicca sul pulsante "Arresta". 2. Il sistema effettua una scansione per controllare se ci sono richieste in esecuzione. 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 l'avvenuto spegnimento con una notifica.

Casi d'uso:

ID	UC_Startup	
Nome caso d'uso	Startup Server	
Attori	Amministratore: Marco	
Entry Condition	L'amministratore accede al sistema	
Flusso di Eventi	Utente Marco accede al Sistema e clicca sul pulsante "Avvia".	Sistema ClickEat accende il server e attiva i servizi in remoto rendendosi disponibile per le richieste notificando il successo dell'operazione all'utente.
Exit Condition	Il Server è attivo e i relativi servizi sono attivi.	

ID	UC_Shutdown	
Nome caso d'uso	Shutdown Server	
Attori	Amministratore: Marco	
Entry Condition	L'amministratore accede al sistema	
Flusso di Eventi	Utente Marco accede al Sistema e clicca sul pulsante "Arresta".	Sistema ClickEat effettua una scansione per verificare se ci sono eventuali richieste in esecuzione, porta a termine quest'ultime e avvia le procedure di arresto. Il sistema notifica il successo dell'operazione all'utente.
Exit Condition	Il Server si è arrestato correttamente.	

3. Servizi dei sottosistemi

3.1. Gestione Sala

Sottosistema	Descrizione
Gestione Sala	Sottosistema che si occupa della visualizzazione e della modifica dello stato attuale della sala.
Servizio	**Descrizione**
freeTavolo()	Permette di settare lo stato del tavolo a “non occupato”.
isOccupato()	Permette di verificare se un tavolo è occupato o meno.
isContoPresente()	Permette di verificare se è stato richiesto il conto per un tavolo.
setOccupato()	Permette di settare lo stato del tavolo a “occupato”.
getTavolo()	Permette di selezionare un singolo Tavolo.

3.2. Gestione Ristorante

Sottosistema	Descrizione
Gestione Ristorante	Tale sottosistema si occupa della gestione del ristorante; fornendo metodi per gestire i dati persistenti del sistema (Piatti, Tavoli, Ingredienti).
Servizio	**Descrizione**
ottieniTavoli()	Permette di ottenere la lista dei tavoli presenti nel sistema.
creaTavolo()	Permette di aggiungere un nuovo Tavolo al sistema,
eliminaTavolo()	Permette di eliminare un Tavolo dal sistema.
ottieniListaPiatti()	Permette di ottenere la lista dei Piatti presenti nel sistema.
ottieniPiatto()	Permette di ottenere un singolo Piatto presente nel sistema.
ingredientiNelPiatto()	Permette di ottenere la lista di ingredienti che compongono un Piatto.
creaPiatto()	Permette di inserire un Piatto nel sistema.
eliminaPiatto()	Permette di eliminare un Piatto presente nel sistema.
creaIngrediente()	Permette di aggiungere un nuovo Ingrediente nel sistema.
eliminaIngrediente()	Permette di eliminare un Ingrediente presente nel sistema.
ottieniListaIngredienti()	Permette di ottenere la lista degli Ingredienti presenti nel sistema.

3.3. Gestione Utenti

Sottosistema	Descrizione
Gestione Utenti	Tale sottosistema si occupa della gestione della gestione degli Utenti; fornendo metodi per aggiungere ed eliminare quest'ultimi.
Servizio	**Descrizione**
ottieniUtenti()	Permette di ottenere la lista degli Utenti presenti nel sistema.
creaUtente()	Permette di aggiungere un nuovo Utente al sistema,
eliminaUtente()	Permette di eliminare un Utente dal sistema.

3.4. Gestione Ordinazione

Sottosistema	Descrizione
Gestione Ristorante	Tale sottosistema si occupa della gestione del ristorante; fornendo metodi per gestire i dati persistenti del sistema (Piatti, Tavoli, Ingredienti).
Servizio	**Descrizione**
creaOrdinazione()	Permette di creare una nuova Ordinazione e inserirla nel sistema.
ottieniOrdinazione()	Permette di ottenere un'Ordinazione presente nel sistema.
inserisciPiattoIntoOrdinazione()	Permette di aggiungere un Piatto ad un Ordinazione.
impostaIdEDataOrdine()	Permette di settare l'identificativo e l'orario ad un'ordinazione
eliminaPiattoIntoOrdinazione()	Permette di eliminare un Piatto presente in un'ordinazione esistente
updateQtaIntoOrdinazione()	Permette di modificare le quantità dei Piatti ordinati