



Corso di Ingegneria del Software

**Restaurant Management
System Design
Versione 2.0**



Data: 19/01/2018

Partecipanti al progetto

Nome	Matricola
Andrea Cipriano	512104874
Gianmarco Cringoli	512104778
Manuel Flora	512104628

Scritto da:	Andrea Cipriano, Gianmarco Cringoli, Manuel Flora
--------------------	---

Revision History

Data	Versione	Descrizione	Autore
10/12/18	1.1	System decomposition	Andrea Cipriano
10/12/18	1.1	System decomposition	Gianmarco Cringoli
10/12/18	1.1	System decomposition	Manuel Flora
14/12/18	1.2	Subsystem services	Andrea Cipriano
14/12/18	1.2	Persistent data management	Gianmarco Cringoli
14/12/18	1.2	Hardware/software mapping	Manuel Flora
19/01/19	2.0	Miglioramenti generali	Andrea Cipriano

Indice

1. Introduzione	5
1.1. Scopo del sistema	5
1.2. Obiettivi di design	5
1.2.1. Dependability criteria	5
1.2.2. Performance criteria	5
1.2.3. End user criteria	6
1.3. Definizioni, acronimi, e abbreviazioni	6
1.4. Riferimenti	6
1.5. Panoramica	6
2. Architettura sistema corrente	7
3. Architettura sistema proposto	7
3.1. Panoramica	7
3.2. Decomposizione del sottosistema	7
3.2.1. Schema generale	7
3.3. Hardware/Software mapping	8
3.4. Gestione dei dati persistenti	8
3.4.1. Class diagram	9
3.4.2. Modello logico	9
3.4.3. Dizionario dei dati	10
3.5. Controllo degli accessi e sicurezza	10
3.6. Controllo globale del software	11
3.7. Condizioni limite	11
3.7.1. Start-up	11
3.7.2. Terminazione	11
3.7.3. Fallimento	11
3.7.4. Use case Start Server	12
3.7.5. Use case Shutdown Server	12
4. Servizi dei sottosistemi	13
4.1. Gestione utente	13
4.2. Gestione cameriere	13
4.3. Gestione attività	14
4.4. Gestione menù	14

4.5. Gestione prenotazioni	15
4.6. Gestione comanda.....	15
5. Glossario	16

1. Introduzione

Con l'obiettivo di ottimizzare la gestione delle attività ristorative di piccola-media dimensione, si vuole sviluppare un progetto che miri a semplificare le funzioni che il gestore di un ristorante e i camerieri svolgono quotidianamente. Il software, inoltre, è rivolto anche ai clienti dei ristoranti, i quali possono interagire con esso attraverso il web.

1.1. Scopo del sistema

L'obiettivo del sistema "Restaurant Management" è quello di creare una piattaforma per la gestione delle attività di ristorazione di media-piccola dimensione. In particolare "Restaurant Management" nasce dall'idea di:

- fornire un servizio per i clienti del ristorante che gli permette di prenotare un tavolo attraverso il web.
- fornire la possibilità al gestore di creare menù e profili cameriere per migliorare la gestione delle ordinazioni.
- offrire un sistema di gestione comande che permette ai camerieri di comunicare con la cucina.

1.2. Obiettivi di design

Il sistema sarà strutturato in maniera chiara e completa. Le operazioni che l'utente effettuerà non richiedono particolare conoscenza tecnologica poichè l'utilizzo sarà guidato dalla semplicità ed intuitività dell'interfaccia.

L'interfaccia grafica sarà curata nei minimi dettagli, mediante l'utilizzo di bottoni, finestre di dialogo non invasive, label semplici e icone dettagliate che offrono all'utente un'esperienza di utilizzo rapida ed esaustiva.

1.2.1. Dependability criteria

La web application garantirà il corretto funzionamento, gestendo i vari tipi di errori. Quindi Restaurant Management rispetterà i seguenti requisiti di qualità, relativi all'affidabilità:

- **Robustezza (priorità alta):** nel caso i cui l'utente inserisca dati errati nel sistema, quest'ultimo farà visualizzare dei messaggi di errore avvisando l'utente che i dati non sono validi. [Requisito non funzionale: Reability]
- **Affidabilità (priorità alta):** il sistema garantisce il corretto svolgimento delle proprie funzionalità, producendo sempre l'output desiderato [Requisito non funzionale: Reability]
- **Fault tolerance (priorità alta):** il sistema garantisce una tolleranza media agli errori, quando si verificassero esso dovrà essere in grado di gestirli e risolverli nel minor tempo possibile.

1.2.2. Performance criteria

Il sistema garantirà buone performance gestendo adeguamente tutti gli utenti senza rallentamenti.

1.2.3. End user criteria

Dal punto di vista dell'utente, il sistema dovrà garantire i seguenti requisiti di qualità:

- **Utilità (priorità alta):** attraverso l'attività di raccolta dei requisiti, il sistema sarà in grado di soddisfare le esigenze degli utenti.
- **Usabilità (priorità alta):** il sistema dovrà essere intuitivo e di semplice utilizzo, e sarà progettato tenendo conto di quella che è l'user experience. Non sarà necessario l'uso di un manuale utente per compiere le azioni. [*Requisito non funzionale: Usability*]

1.3. Definizioni, acronimi, e abbreviazioni

Acronimi	Descrizione
RAD	Requirement Analysis Document
SDD	System Design Document
HW	Hardware
SW	Software
SQL	Structure Query Language
DBMS	Database Management System
GUI	Graphical User Interface

1.4. Riferimenti

- Problem Statement v.3.0
- RAD_V_3.0
- B.Brugge, A.H. Dutoit, Object Oriented Software Engineering- Using UML, Patterns and Java, Prentice Hall.
- <https://www.bruegge.in.tum.de>

1.5. Panoramica

Il seguente documento sarà così strutturato:

Introduction: riporta una descrizione del sistema specificando le ragioni del suo sviluppo, le caratteristiche e un accenno sull'utilizzo delle funzionalità.

Architettura del sistema proposto: fornisce una panoramica sull'architettura usata per il sistema. Tratta della suddivisione in sottosistemi, del mapping software-hardware, della gestione dei dati persistenti, del controllo degli accessi in sicurezza, del flusso di controllo globale e delle condizioni limite.

Servizi dei sottosistemi: espone una descrizione dei sottosistemi identificati e, per ognuno, i servizi offerti.

Glossary: è una raccolta di termini nell'ambito specifico

2. Architettura sistema corrente

L'applicazione proposta non andrà a sostituire nessun sistema già esistente. La progettazione e lo sviluppo del software partirà da zero e seguirà i criteri della Greenfield Engineering.

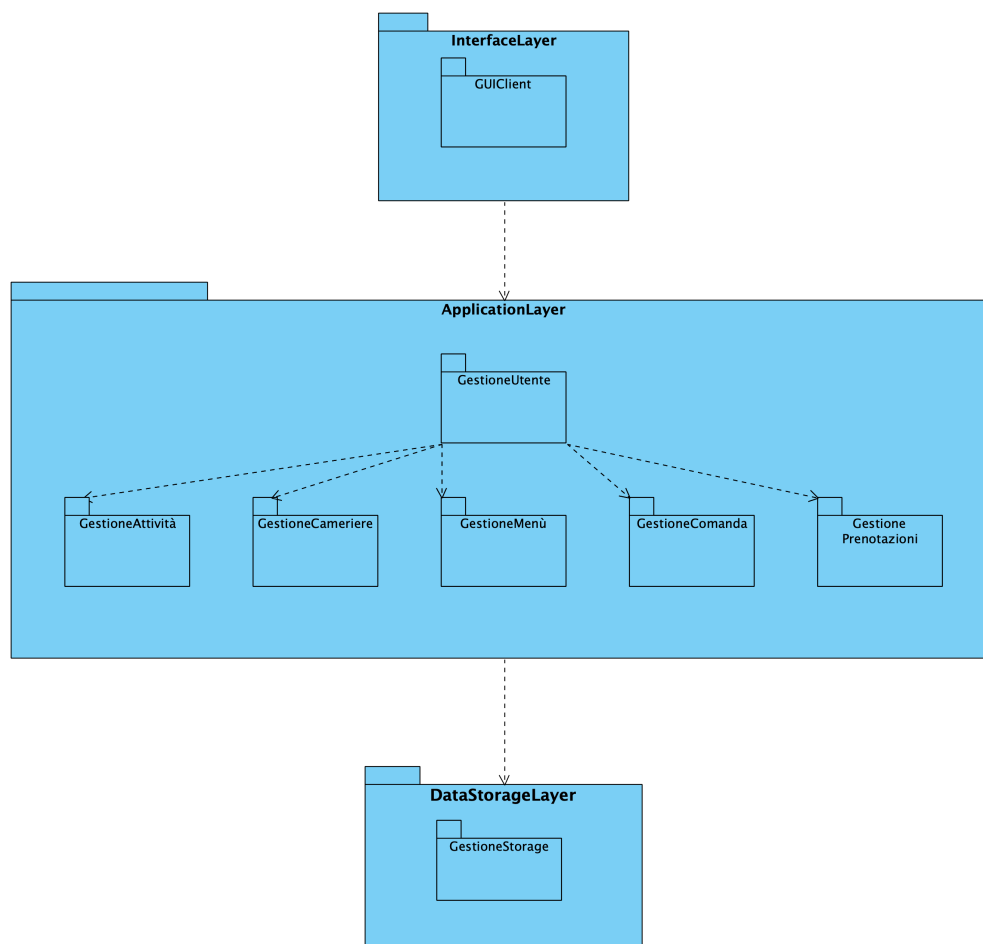
3. Architettura sistema proposto

3.1. Panoramica

L'architettura scelta per il sistema da realizzare sarà quella Three-layer. L'utente potrà interagire con l'application layer mediante l'interface layer che offrirà diverse interfacce di base alle necessità dell'utente. L'application layer dovrà poi comunicare con i database per la memorizzare dei dati persistenti. Sul server, risiede un DBMS che si occupa di recuperare, memorizzare ed interrogare i dati presenti nel database, elaborando, quindi, la richiesta dell'utente. L'aspetto della concorrenza di accessi multipli al database, sarà pertanto gestito dal DBMS stesso che dovrà evitare eventuali colli di bottiglia. Tale architettura conferisce all'intero sistema una maggiore manutenibilità e permette di gestire il problema della concorrenza degli accessi ai dati in maniera semplice ed efficace.

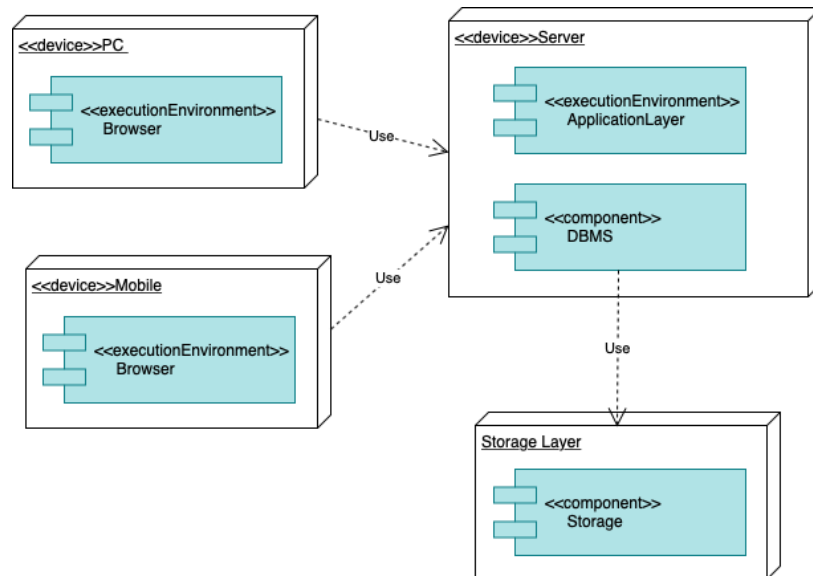
3.2. Decomposizione del sottosistema

3.2.1. Schema generale



- **InterfaceLayer:** GUIClient è l'interfaccia col la quale gli utenti interagiscono con il sistema.
- **ApplicationLogicLayer:** contiene i seguenti sottosistemi: gestione utente, gestione cameriere, gestione attività, gestione menu, gestione prenotazioni
- **GestoreStorage:** sottosistema che ha il compito di effettuare operazioni sul database.

3.3. Hardware/Software mapping



Il sistema sviluppato sarà installato su un solo server e utilizzerà un DBMS MySQL stanziato sullo stesso per la gestione dei dati persistenti. Il sistema sarà diviso in un'architettura client e server.

Il Deployment Diagram mostra le componenti che utilizzerà il nostro sistema; questo diagramma aiuta gli sviluppatori a comprendere le relazioni tra le componenti software e i nodi hardware. Al lato client, l'interfaccia utente verrà mostrata sul browser web ed interagirà con l'application layer, che a sua volta memorizza e interroga i dati presenti nel database.

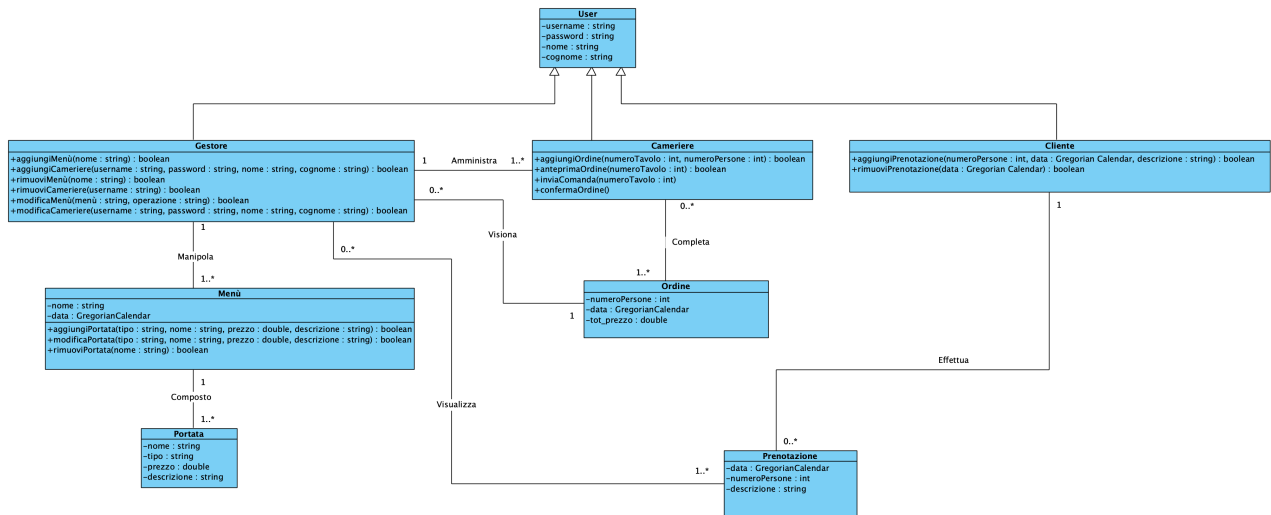
3.4. Gestione dei dati persistenti

Il sistema userà sia un servizio di storage su file, che servizi di storage su Database. Le risorse saranno opportunamente salvate in cartelle sul server.

L'archiviazione su file sarà utilizzata per inserire il ristorante da gestire; riteniamo che creare una tabella nel database per inserire un solo record sia dispendioso e poco utile. La scelta del DBMS è ricaduta su MySQL viste le conoscenze del team. La scelta di un database relazionale, rispetto a un database object-oriented, è motivata dalle alte prestazioni offerte dai primi.

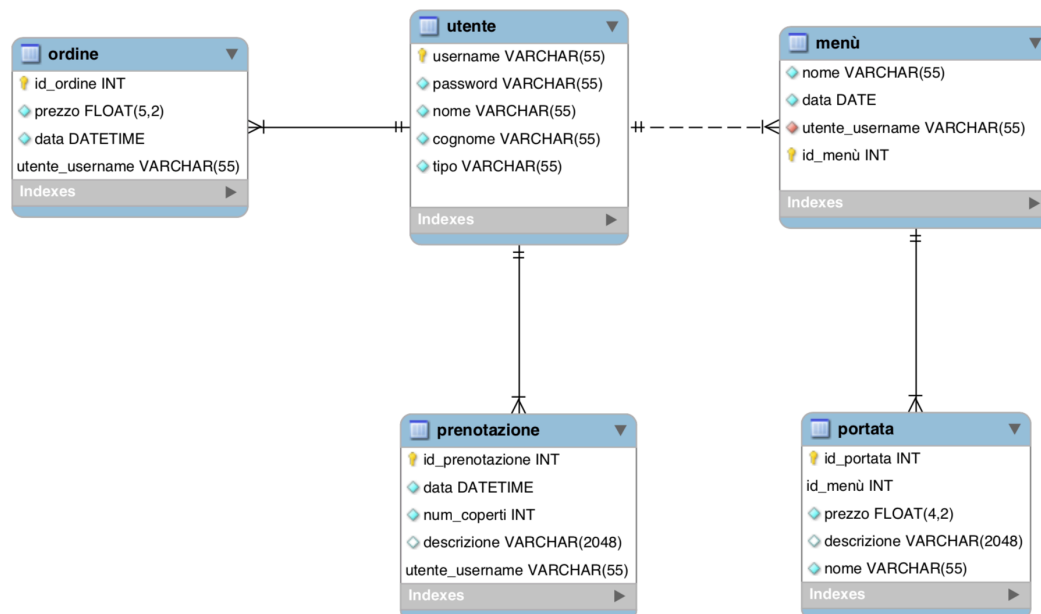
3.4.1. Class diagram

Il class diagram qui mostrato descrive le entità che saranno salvate nel database. A differenza del class diagram presente nel RAD_V3, abbiamo effettuato delle modifiche riguardanti le entità cucina, tavolo e portata comanda, che non necessitano di salvataggio su database e ristorante, che sarà salvato in un file come su detto.



3.4.2. Modello logico

Il modello logico rappresenta la struttura del database.



3.4.3.Dizionario dei dati

Tabella	
	Attributi
Utente	<ul style="list-style-type: none"> • <i>username</i>: chiave primaria utente • <i>password</i>: attributo • <i>nome</i>: attributo per il nome dell'utente • <i>cognome</i>: attributo per il cognome dell'utente • <i>tipo</i>: attributo per distinguere i vari utenti (gestore, cameriere, cliente)
Tabella	
	Attributi
Menù	<ul style="list-style-type: none"> • <i>id_menù</i>: chiave primaria menù • <i>nome</i>: attributo per il nome del menù • <i>data</i>: attributo per la data di aggiunta
Tabella	
	Attributi
Portata	<ul style="list-style-type: none"> • <i>id_portata</i>: chiave primaria portata • <i>nome</i>: attributo per il nome della portata • <i>prezzo</i>: attributo per il prezzo della portata • <i>descrizione</i>: attributo per la descrizione della portata
Tabella	
	Attributi
Prenotazione	<ul style="list-style-type: none"> • <i>id_prenotazione</i>: chiave primaria prenotazione • <i>data</i>: attributo per la data della prenotazione • <i>numeroPersone</i>: attributo per il numero delle persone • <i>descrizione</i>: attributo per la descrizione di una prenotazione • <i>utente_username</i>: vincolo di integrità referenziale
Tabella	
	Attributi
Ordine	<ul style="list-style-type: none"> • <i>id_ordine</i>: chiave primaria ordine • <i>totale</i>: attributo per l'ammontare dell'ordine • <i>data</i>: attributo per la data dell'ordine • <i>utente_username</i>: vincolo di integrità referenziale con l'utente (cameriere)

3.5. Controllo degli accessi e sicurezza

Il controllo degli accessi sarà alla base della sicurezza del nostro sistema, permettendo ad ogni utente di collegarsi al sistema tramite l'utilizzo di username e password, che verranno richieste ad ogni singolo accesso. La sessione termina quando l'utente effettua il logout.

Nel caso l'accesso al sistema non abbia successo, verrà inviata una notifica di fallimento indicando che è avvenuto un errore nell'inserimento di username o password e consentendo all'utente di effettuare un nuovo tentativo.

L'utente "cliente" potrà usufruire dei servizi che non utilizzano la registrazione, ma nel caso cui lui vorrebbe effettuare una prenotazione verrà reindirizzato alla pagina di login o registrazione.

Il sistema fornirà più visite (interfacce grafiche) a seconda dell'attore che ci interagirà, in modo che ognuno possa accedere solo alle rispettive funzionalità previste.

3.6. Controllo globale del software

Per quanto riguarda il flusso di controllo esterno fra sottoinsiemi, il server sarà sempre in funzione in attesa di eventuali richieste da parte dell'utenza; il sistema non avrà problemi a gestire più utenti contemporaneamente.

3.7. Condizioni limite

Il server sarà sempre attivo permettendo l'utilizzo del servizio in qualsiasi orario e in presenza di un malfunzionamento, il sistema mostrerà una pagina di manutenzione.

3.7.1. Start-up

Per il primo start-up del sistema "RestaurantManagement", è necessario l'avvio di un web server che fornisca il servizio di un Database MySQL per la gestione dei dati persistenti e l'interpretazione ed esecuzione del codice lato server, assumendo che, prima dello start-up iniziale, il database ha un gestore che, una volta loggato, può gestire il proprio ristorante. In seguito, tramite la homepage, qualsiasi utente potrà usufruire di alcuni servizi, ma solo dopo essersi registrato ed effettuato il login può accedere a tutte le funzionalità a lui dedicate.

3.7.2. Terminazione

La chiusura della web page non comporta un logout dal sistema. Per effettuare il logout bisogna cliccare sul pulsante specifico

3.7.3. Fallimento

Un esempio di fallimento possibile generato dalla parte del sistema:

- malfunzionamento del server dovuto ad un improvviso guasto al disco. La priorità del sistema nell'affrontare questo tipo di problema è quella di perdere meno dati possibili. A questo scopo potrebbe essere usata questa strategia:
 - salvare su GitHub, e sul personal computer dell'amministratore, la logica di business dell'applicazione;

Un esempio di fallimento possibile generato dalla parte Client:

- nell'inserimento dei dati, per accedere al sistema oppure in fase di creazione (di una qualsiasi cosa, menù, cameriere ecc.), o modifica, se i campi non verranno completati
- correttamente e/o non avranno riscontro dal database, l'accesso, l'inserimento o modifica fallirà, mostrando un messaggio d'errore del mancato accesso/inserimento/modifica.

3.7.4. Use case Start Server

Nome caso d'uso: **StartServer**



Attori partecipanti:	Amministratore.
Flusso d'eventi:	<ol style="list-style-type: none">1. L'amministratore avvia il server.2. Il server segnala che l'avvio è avvenuto con successo e i suoi servizi vengono messi a disposizione degli utenti.
Entry condition:	<ul style="list-style-type: none">• Il server è spento e funzionante.
Exit condition:	<ul style="list-style-type: none">• Il server viene avviato con successo.
Exceptions:	Il sistema segnala che non è possibile avviare il server e che, quindi, non viene avviato con successo.

3.7.5. Use case Shutdown Server

Nome caso d'uso: **ShutdownServer**



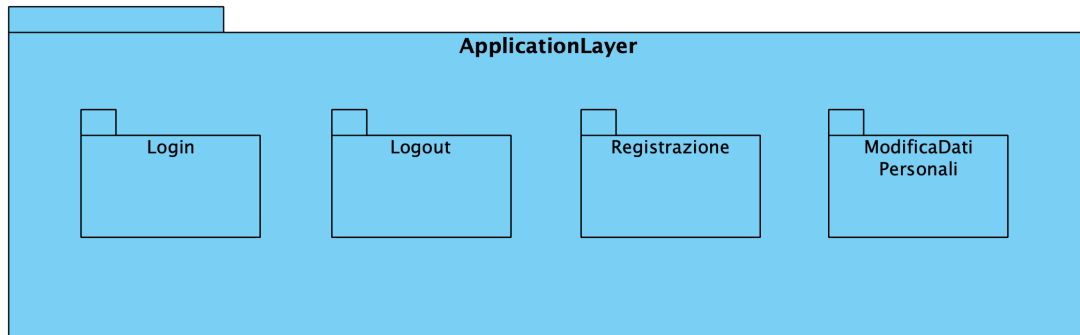
Attori partecipanti:	Amministratore.
Flusso d'eventi:	<ol style="list-style-type: none">1. L'amministratore spegne il server..2. Il server si spegne
Entry condition:	<ul style="list-style-type: none">• Il server è acceso.

Exit condition:

- Il server viene arrestato.

4. Servizi dei sottosistemi

4.1. Gestione utente



ApplicationLogicLayer

Comprende tutte le componenti logiche associate all'accesso al sistema

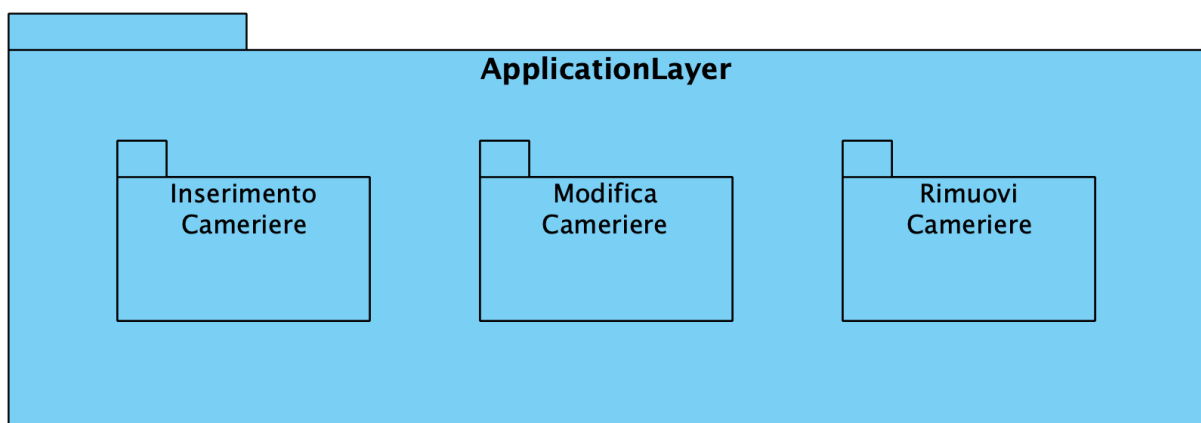
Registrazione: operazione che permette al cliente di registrarsi al sistema

Login: permette agli utenti di autenticarsi al sistema.

Logout: permette agli utenti di scollegarsi dal sistema.

ModificaDatiPersonali: permette di modificare i dati personali

4.2. Gestione cameriere



ApplicationLogicLayer

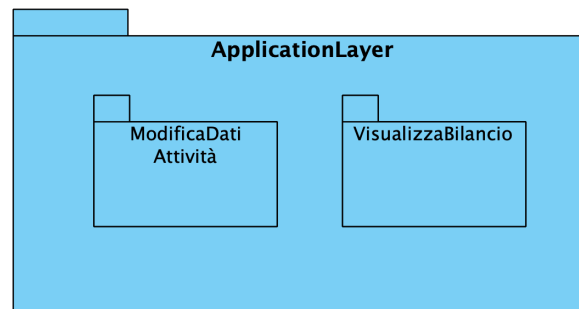
Comprende tutte le componenti logiche associate alla gestione di un profilo "cameriere"

InserimentoCameriere: permette di inserire un account "cameriere" attraverso il quale i camerieri possono accedere al sistema.

ModificaCameriere: permette di apportare modifiche ad un account "cameriere" creato in precedenza.

RimuoviCameriere: permette di eliminare un account "cameriere" creato in precedenza.

4.3. Gestione attività



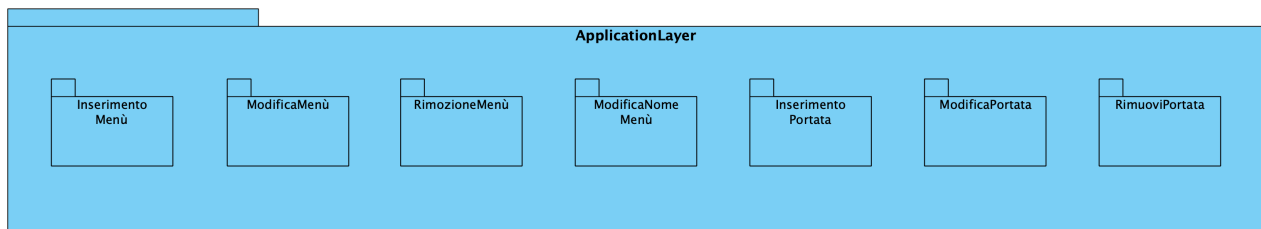
ApplicationLogicLayer

Comprende tutte le componenti logiche associate alla gestione delle informazioni relative al ristorante.

Modifica dati attività: permette di modificare i dati dell'attività ristorativa inseriti al primo accesso al sistema

Visualizza bilancio: permette di visualizzare una panoramica sul bilancio dell'attività

4.4. Gestione menù



ApplicationLogicLayer

Comprende tutte le componenti logiche associate alla gestione dei menu e delle relative portate

InserimentoMenu: permette di inserire un nuovo menu.

ModificaMenu: permette di modificare uno dei menu inseriti.

RimozioneMenu: permette di eliminare definitivamente un menu, con le rispettive portate.

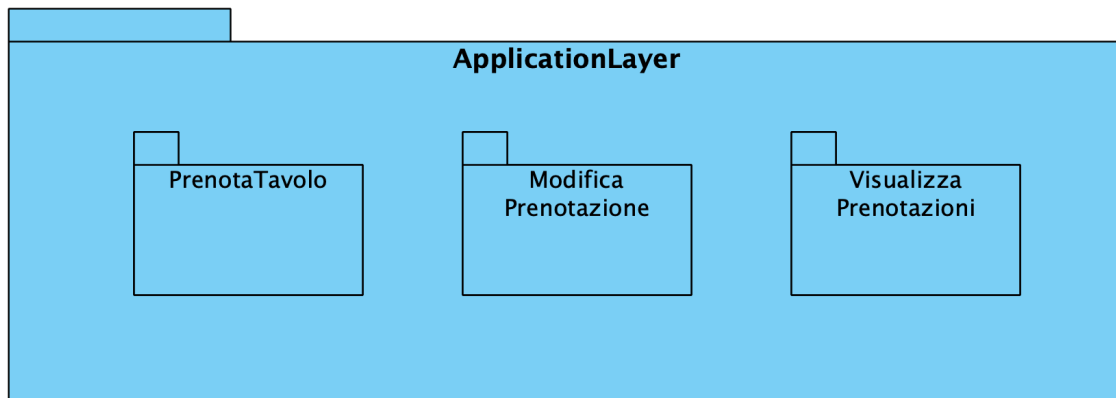
ModificaNomeMenu: permette di modificare il nome di un menu inserito.

InserimentoPortata: permette di aggiungere una portata ad un menu.

ModificaPortata: permette di modificare tutti gli attributi relativi ad una portata

RimozionePortata: permette di rimuovere una portata da un menu.

4.5. Gestione prenotazioni



ApplicationLogicLayer

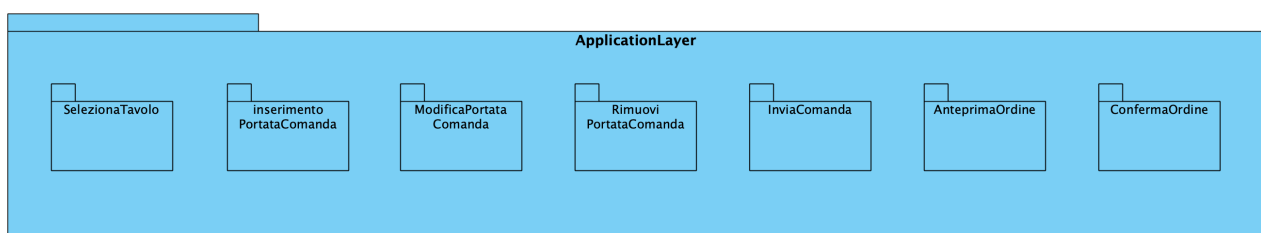
Comprende tutte le componenti logiche associate alla gestione delle prenotazioni

PrenotaTavolo: permette di prenotare un tavolo presso il ristorante.

VisualizzaPrenotazioni: permette di visualizzare lo storico delle prenotazioni effettuate

RimozionePrenotazione: permette, laddove è possibile, di disdire una prenotazione

4.6. Gestione comanda



ApplicationLogicLayer

Comprende tutte le componenti logiche associate alla gestione dei menu e delle relative portate

SelezionaTavolo: interfaccia attraverso la quale si può inserire un nuovo menu.

InserimentoPortataComanda: interfaccia attraverso la quale si può modificare un menu.

ModificaPortataComanda: interfaccia attraverso la quale si può eliminare un menu.

RimuoviPortataComanda: interfaccia attraverso la quale si può modificare il nome di un menu.

InviaComanda: interfaccia attraverso la quale si può inserire una portata al menu.

AnteprimaOrdine: interfaccia attraverso la quale si può modificare una portata del menu.

ConfermaOrdine: interfaccia attraverso la quale si può rimuovere una portata al menu.

5. Glossario

Design goals: obiettivi qualitativi del sistema, identificano le qualità su cui deve essere focalizzato il sistema.

Dependability Criteria: quantificazione dello sforzo che deve essere speso per minimizzare i crash del sistema e delle loro conseguenze.

Perfomance Criteria: requisiti imposti sul sistema in termini di spazio e velocità.

End User Criteria: qualità non incluse nei criteri di performance e affidabilità che sono desiderabili dal punto di vista dell'utente.

Interface layer: si occupa di gestire l'interfaccia utente. Include gli oggetti boundary con cui l'utente interagisce (finestre, form, pagine web, ecc...)

Application logic layer: comprende le componenti logiche (oggetti entity e control), responsabili del corretto funzionamento del sistema. Inoltre, utilizza il database per effettuare operazioni sui dati.

Data storage: gestore che si occupa di effettuare operazioni di inserimento, aggiornamento e cancellazione. Rende disponibili i dati persistenti all'interno del database.

Persistent data management: gestore dei dati che sono memorizzati nel database

Modello logico: il modello logico discende dal modello concettuale e disegna un'architettura che tiene conto delle strutture proprie di quel particolare tipo di database.

Query: interrogazione di un database per estrarre o aggiornare i dati che soddisfano un certo criterio di ricerca.

Dizionario dei dati: permette di arricchire il modello logico con descrizioni in linguaggio naturale.

Boundary conditions: soluzioni a problemi fisici.