

|  |
| --- |
| System Design Document  Versione 1.0  **ANNO ACCADEMICO 2020/2021** |
|  |
| RISTOMANAGER  Ambrosio Salvatore  Costante Marco  Benitozzi Simone  Nappo Carla Alessia |



PARTECIPANTI

|  |  |
| --- | --- |
| **NOME** | **MATRICOLA** |
| Salvatore Ambrosio | 0512106166 |
| Costante Marco | 0512105772 |
| Benitozzi Simone | 0512105742 |
| Nappo Carla Alessia | 0512105956 |

REVISION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **DATA** | **VERSIONE** | **DESCRIZIONE** | **AUTORE** |
| 03/12/2020 | 1.0 | Prima stesura del template | Costante Marco |
| 04/12/2020 | 1.1 | Aggiunta contenuti | Nappo Carla Alessia |
| 07/12/2020 | 1.2 | Aggiunta contenuti riguardanti l’architettura del software proposto | Benitozzi Simone |
| 10/12/2020 | 1.3 | Aggiunta contenuti | Ambrosio Salvatore |

# INDICE

[INDICE 4](#_Toc57901030)

[1.Introduzione 5](#_Toc57901031)

[1.1 Scopo del sistema 5](#_Toc57901032)

[1.2 Obiettivi di Design 5](#_Toc57901033)

[1.3 Definizioni 6](#_Toc57901034)

[1.4 Acronimi 7](#_Toc57901035)

[1.5 Riferimenti 7](#_Toc57901036)

[1.6 Panoramica 7](#_Toc57901037)

[2. Architettura del software corrente 9](#_Toc57901038)

[3. Architettura del software proposto 10](#_Toc57901039)

[3.1 Panoramica 10](#_Toc57901040)

[3.2 Decomposizione del sistema 12](#_Toc57901041)

[3.3 Mapping hardware e software 14](#_Toc57901042)

[3.4 Gestione dei dati persistenti 15](#_Toc57901043)

[3.5 Controllo degli accessi e sicurezza 16](#_Toc57901044)

[3.6 Controllo del software globale 17](#_Toc57901045)

[3.7 Condizioni di Boundary 18](#_Toc57901046)

[**3.7.1 Avvio del sistema 18**](#_Toc57901047)

[**3.7.2 Terminazione del sistema 18**](#_Toc57901048)

[**3.7.3 Fallimento del sistema 18**](#_Toc57901049)

[4. Servizi dei sottosistemi 19](#_Toc57901050)

[5. Glossario 21](#_Toc57901051)

# 1.Introduzione

## **1.1 Scopo del sistema**

L’obiettivo del sistema è venir incontro all’esigenze dei ristoratori, che a seguito della diffusione del COVID-19 necessitano sempre di più di sistemi sicuri, nonché una rapida gestione di quelle che sono le tipiche esigenze di un’attività ristorativa.

Lo scopo, quindi, è quello di informatizzare la gestione del locale attraverso un sistema che permetta non solo di limitare al massimo i contatti col personale di sala, ma anche di tener traccia della clientela per poi poter avvisare in caso di esposizione al contagio.

Il sistema permette al cliente di accedere al menù, comporre il proprio ordine e mandare la comanda in cucina, il tutto mediante il proprio dispositivo.

RistoManager offre, inoltre, la possibilità di gestire in modo efficiente il menù, il quale soggetto a periodici cambiamenti.

## **1.2 Obiettivi di Design**

1.2.1 Criteri di performance

* Tempo di risposta: RistoManager deve essere reattivo per tutte le operazioni più immediate come l’aggiunta di un prodotto. Per la visualizzazione del menù deve garantire dei tempi di risposta brevi, ma, essendo un sistema web, dipenderà molto dalla qualità di connessione con il quale il dispositivo navigherà.
* Throughput: I picchi di carico devono essere gestiti dal sistema senza rallentamenti, garantendo fluidità.
* Memoria: Il sistema utilizza un database relazionale per memorizzare tutti i dati che non rappresenterà un problema di performance del sistema.

1.2.2 Criteri di affidabilità

* Robustezza: I componenti devono essere affidabili ed essere in grado di mantenere i propri dati anche in caso di guasti. Inoltre, deve gestire eventuali input non validi da parte degli utenti.
* Disponibilità: RistoManager deve essere disponibile all’uso 24 ore su 24 da parte degli utenti grazie ad un server sempre attivo.
* Tolleranza all’errore: Il sistema deve essere capace di operare durante condizione d’errore.

1.2.3 Criteri di manutenzione

* Estendibilità: La progettazione del sistema sarà condotta in modo da agevolare la facile introduzione di nuove funzionalità.
* Modificabilità: Deve essere possibile intervenire sul codice esistente per correggere eventuali bugs o implementare nuove funzionalità. Bisogna garantire che il codice sia leggibile per rendere agevole la modifica.
* Leggibilità: Il codice sarà ben strutturato per semplificare eventuali interventi sy di esso.
* Tracciabilità dei requisiti: Sarà possibile effettuazione le modifiche necessarie al corretto funzionamento del sistema.

1.2.4 Criteri per l’utente finale

* Usabilità: RistoManager rende ogni funzione di semplice uso garantendo un’ottima esperienza all’utente che dovrà utilizzarlo grazie ad interfacce grafiche intuitive.

## **1.3 Definizioni**

* RistoManager: nome del sistema che verrà sviluppato.
* Cliente abilitato: utente registrato che può effettuazione l’ordine.
* Personale gestione: attore che gestisce il sistema.
* Personale sala: attore che genera codice.
* Personale cucina: attore che accetta e conclude gli ordini.

## **1.4 Acronimi**

# RAD: Requirements Analysis Document

# SDD: System Design Document

# DB: Database

# UC: Use Case

# SEQD: Sequence Diagram

# MU: Mock-Ups

# RF: Functional Requirements

# NRF: Non-Functional Requirements

## **1.5 Riferimenti**

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

## **1.6 Panoramica**

Il documento si compone di cinque parti. In particolare, nella prima parte sono stati introdotti gli obiettivi di design e sono stati forniti i riferimenti ad altri materiali. In seguito, nella sezione sistema software corrente, verrà descritto un sistema software simile. La sezione sistema software proposto documenta il system design del nuovo sistema. In questa sezione vengono descritti i seguenti elementi:

* + Decomposizione in sottosistemi: il sistema viene suddiviso in diversi sottosistemi. Ricordiamo che un sottosistema è formato da un insieme di classi, associazioni, operazioni e vincoli che sono in relazione tra di loro. Ogni sottosistema è caratterizzato dai servizi che offre agli altri sottosistemi.
  + Mapping hardware/software: in questa sezione vengono prese decisioni riguardo le piattaforme hardware su cui il sistema dovrà girare, una volta decise le piattaforme è necessario mappare le componenti su di esse.
  + Gestione dei dati persistenti: descrive i dati persistenti che vengono memorizzarli dal sistema ed il tipo di infrastruttura usata per memorizzarli.
  + Controllo degli Accessi e Sicurezza: descrive il modello degli utenti del sistema in termini di una matrice degli accessi.
  + Controllo Globale del Software: descrive il modo in cui è implementato il controllo globale del software e come si sincronizzano i sottosistemi.
  + Condizioni di boundary: vengono descritte le condizioni limite del sistema come start-up e shutdown e la gestione dei fallimenti del sistema. La sezione servizi dei sottosistemi descrive in termini di operazioni quali sono i servizi forniti da ciascun sottosistema.

L’ultima parte del documento è costituita dal glossario che si occupa di elencare una serie di termini e fornire la relativa spiegazione in maniera tale da fornire supporto a coloro che leggono il documento.

# 2. Architettura del software corrente

CHIEDERE AL TUTOR!!

# 3. Architettura del software proposto

## **3.1 Panoramica**

L’architettura del sistema è di tipo client/server. Il server riceve le richieste da parte del client, e risponde in tempo utile. I motivi di questa scelta sono:

* Portabilità: il sistema potrà essere utilizzato su una varietà di macchine e sistemi operativi, e dovrà essere ottimizzato per l’utilizzo da dispositivi mobili, dai quali si prevede una maggiore affluenza da parte dei clienti;
* Scalabilità: il server sarà in grado di gestire un numero variabile di client contemporaneamente, garantendo prestazioni ottimali anche in situazioni di maggiore stress.
* Flessibilità: per ogni tipologia di utente che effettua l’accesso al sistema, vi sarà un’interfaccia grafica apposita, tramite la quale ogni attore potrà eseguire le operazioni ad esso riservate.
* Affidabilità: entrambi i componenti client e server devono essere affidabili ed essere in grado di mantenere i propri dati, nel rispetto della privacy degli interessati, anche in seguito a guasti: deve quindi essere possibile effettuare dei backup periodici al database.

Nello sviluppo del sistema verrà utilizzata un’architettura di tipo Model-View-Controller, che prevede che il software venga diviso in tre parti, ciascuna delle quali con un compito diverso. In particolare:

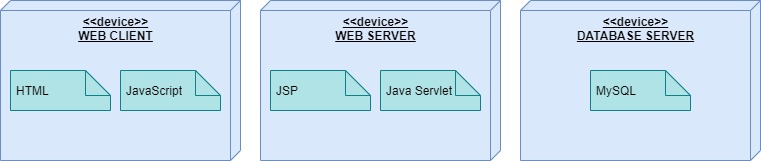
Il model si occupa della di gestire i dati e quindi sarà responsabile dell’interazione con il database.

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.

Dal punto di vista implementativo il model verrà realizzato utilizzando classi Java, la parte di view verrà implementata utilizzando pagine HTML con l’integrazione di funzioni JavaScript, mentre i control saranno realizzati tramite Java Servlet.



## **3.2 Decomposizione del sistema**

Per realizzare il sistema è stata usata un’architettura three-tier, in cui la logica dell’applicazione viene suddivisa in tre layer:

* Presentation layer: costituito dalle interfacce grafiche con cui l’utente interagisce.
* Application layer: costituito dagli oggetti che si occupano della gestione del controllo, dell’elaborazione dati e di notificare i cambiamenti al presentation layer. Interagisce con il database attraverso lo storage layer.
* Storage layer: si occupa della memorizzazione dei dati persistenti e del loro recupero dal database.

Il Presentation Layer è stato suddiviso in cinque sottosistemi:

* + Sottosistema Ospite. Costituito dall’interfaccia riservata ad un utente che ha appena effettuato l’accesso alla piattaforma, da cui ha la possibilità di prenotarsi oppure ordinare.
  + Sottosistema Cliente Abilitato. Include tutte le interfacce grafiche a cui il cliente a cui è già stato assegnato un tavolo può accedere, comprensive di visualizzazione del menù, attraverso criteri opzionali, opzioni per effettuare un nuovo ordine e fare richieste specifiche alla cucina.
  + Sottosistema Personale di Gestione. Include tutte le interfacce grafiche a cui un membro dello staff che si occupa degli aspetti di gestione può accedere. In particolare, ha accesso ad operazioni di gestione del menù, quali aggiunta, modifica e rimozione di prodotti, e degli utenti registrati sulla piattaforma, dei quali può visualizzare dati, filtrati per un lasso di tempo specificato, e può inoltre rimuovere account appartenenti agli altri membri dello staff.
  + Sottosistema Personale di Sala. Include tutte le interfacce a cui un membro dello staff di sala può accedere.

Comprende funzionalità che gli permettono di generare un codice per un nuovo cliente, e di effettuare, su richiesta del cliente, ordinazioni

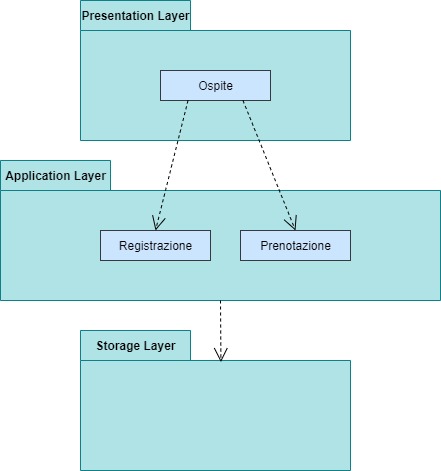
* + Sottosistema Personale di Cucina. Include le interfacce accessibili da un membro dello staff di cucina, dalle quali può effettuare operazioni di visualizzazione, accettazione e notifica del completamento di un ordine.

L’ Application Layer è stato diviso invece in quattro sottosistemi:

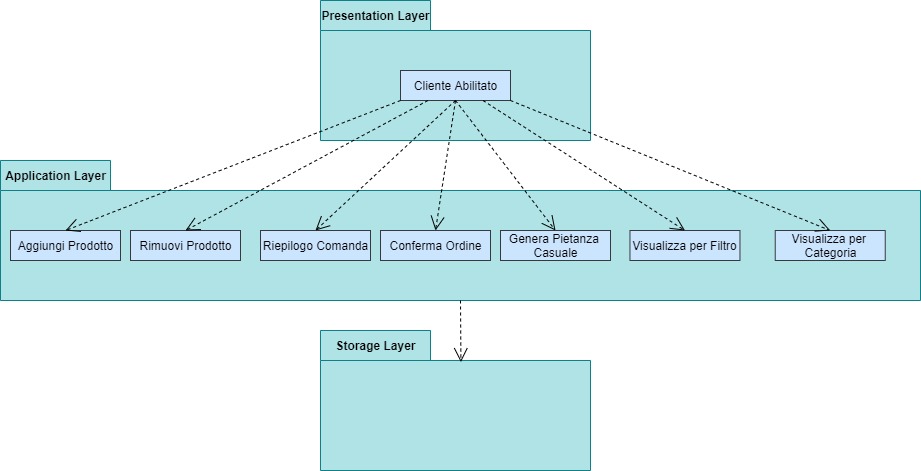
* + Sottosistema Gestione utenti. Include le operazioni che permettono di prenotare un tavolo oppure ordinare, nel caso di Ospite, effettuare login e logout, per tutti i membri dello staff, eliminare utenti e visualizzare dati statistici, nel caso del Personale di Gestione e generare un codice per un nuovo cliente, nel caso del Personale di Sala.
  + Sottosistema Gestione Comande. Include le operazioni che permettono al cliente, e su richiesta al personale di sala, di ordinare nuove portate, rimuovere prodotti aggiunti in precedenza, visualizzare un riepilogo totale dell’ordine e confermare l’ordinazione da spedire in cucina.
  + Sottosistema Gestione Cucina. Include le operazioni che permettono al personale di cucina di visualizzare, accettare e notificare il completamento per ordini ricevuti dai clienti.
  + Sottosistema Gestione Menù. Include le operazioni che permettono al personale di gestione di aggiungere, modificare e rimuovere prodotti dal menu.

Nello specifico:

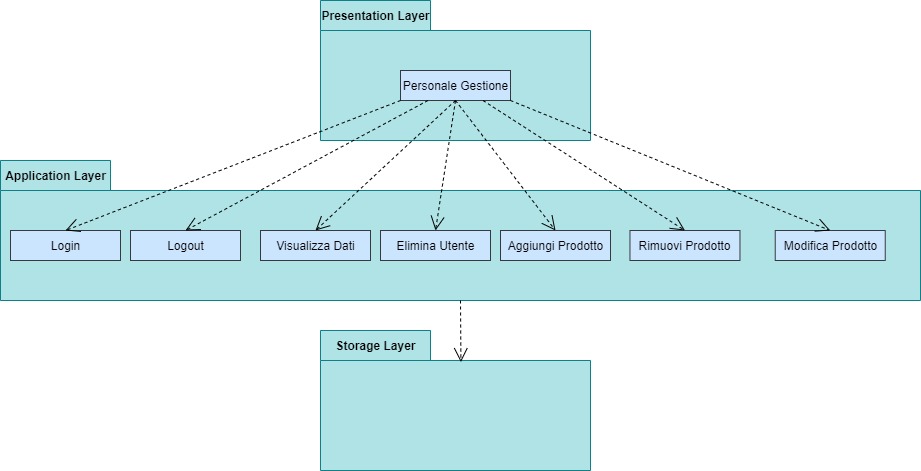
3.2.1 Ospite



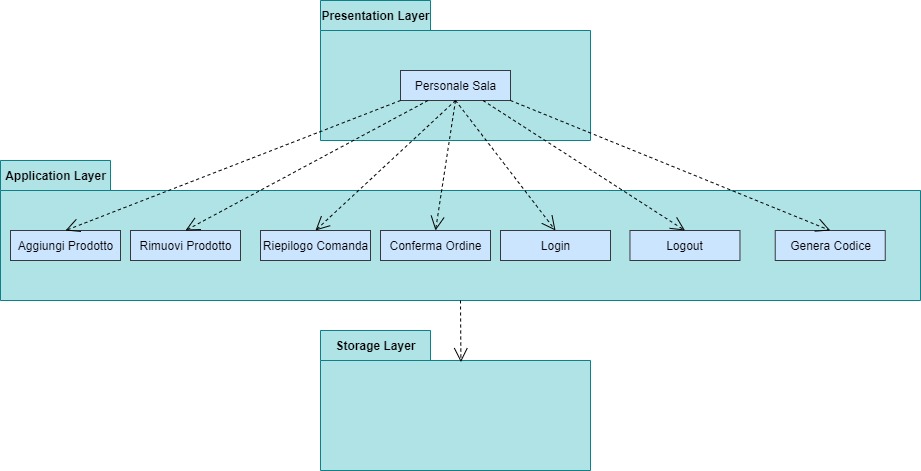
3.2.2 Cliente Abilitato



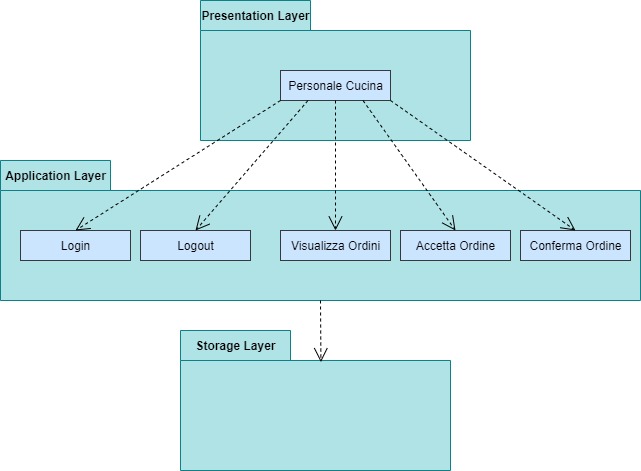
3.2.3 Personale di Gestione



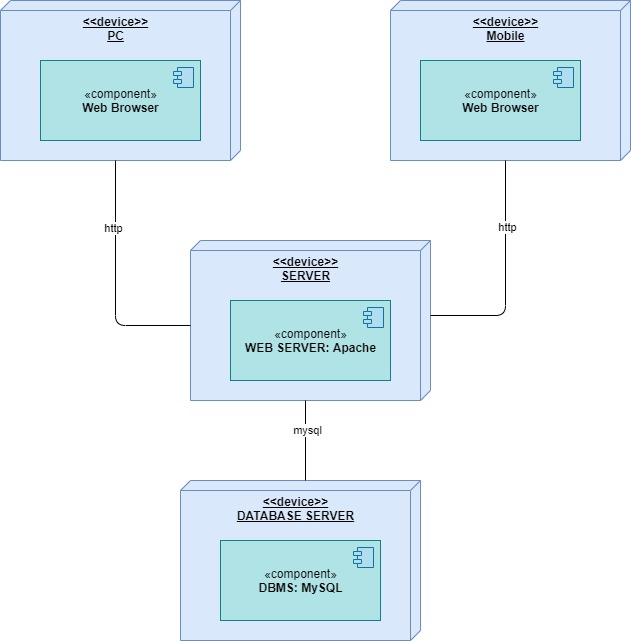
3.2.4 Personale di Sala



3.2.4 Personale di Cucina



## **3.3 Mapping hardware e software**



Web Server

Il server utilizzato è Apache Tomcat.

Interface layer

L’utente utilizza il sistema mediante un Web Browser installato all’interno del suo dispositivo.

Application Logic layer

Le funzionalità del sistema sono implementate in linguaggio Java Servlet. Il codice in Java Servlet verrà tradotto in linguaggio HTML e JSP e interpretato dal browser dell’utente.

La comunicazione con il Database sarà implementata attraverso JDBC.

Database Server

Il Database Management System utilizzato è MySQL.

## **3.4 Gestione dei dati persistenti**

## **3.5 Controllo degli accessi e sicurezza**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Ospite** | **Cliente Abilitato** | **Personale Gestione** | **Personale Sala** | **Personale Cucina** |
| **Gestione Utenti** | ✓prenotazione  ✓registrazione |  | ✓visualizza dati  ✓elimina utente  ✓login  ✓logout | ✓login  ✓logout  ✓genera codice | ✓login  ✓logout |
| **Gestione Comande** |  | ✓aggiungi  ✓rimuovi  ✓riepilogo  ✓conferma |  | ✓aggiungi  ✓rimuovi  ✓riepilogo  ✓conferma |  |
| **Gestione Cucina** |  |  |  |  | ✓visualizza ordine  ✓accetta ordine  ✓conferma ordine |
| **Gestione Menù** |  | ✓genera pietanza casuale  ✓visualizza per filtro  ✓visualizza per categoria | ✓aggiungi prodotto  ✓rimuovi prodotto  ✓modifica prodotto |  |  |

## **3.6 Controllo del software globale**

Il controllo del flusso software viene gestito da classi Java Servlet 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.

## **3.6.1 Problemi di concorrenza e sincronizzazione**

????

## **3.7 Condizioni di Boundary**

Le condizioni limite riguardano l’accensione e lo spegnimento del sistema per quanto riguarda il lato Server.

### 3.7.1 Avvio del sistema

Per lo start-up del sistema è necessario avviare il web server, dopodiché il sistema tenterà di recuperare i dati dal database se presenti.

A questo punto l’utente avrà la possibilità di collegarsi dal proprio dispositivo attraverso web browser. Il sistema mostrerà la pagina iniziale, nella quale l’utente avrà la possibilità di prenotare e ordinare.

### 3.7.2 Terminazione del sistema

Per terminare la propria attività, un cliente collegato alla piattaforma dovrà semplicemente chiudere la pagina del browser in cui l’applicazione è in esecuzione, mentre, se si tratta di un membro dello staff, per la regolare terminazione sarà necessario effettuare anche l’operazione di logout.

Per quanto riguarda invece la terminazione del server, sarà necessario l’intervento dell’amministratore del sito che dovrà effettuare la procedura di terminazione dopo la quale non sarà temporaneamente possibile per nessun client connettersi al sistema, fino ad un successivo riavvio.

I dati vengono periodicamente salvati nel database, in maniera tale che al successivo riavvio del sistema vengano recuperati nella loro interezza.

### 3.7.3 Fallimento del sistema

Nel caso si verifichi un errore che comprometta il corretto funzionamento dell’applicazione, si cercherà di ripristinare una configurazione del sistema precedente allo stato d’errore. Essendo i dati gestiti da un DBMS non dovrebbero esserci rischi di perdita degli stessi.

Nel caso in cui si dovesse invece verificare un guasto al supporto di memorizzazione potrebbe verificarsi una perdita di dati.

# 4. Servizi dei sottosistemi

## **4.1 Gestione Utenti**

|  |  |  |
| --- | --- | --- |
| DESCRZIONE: | Sottosistema che gestisce e raccoglie tutte le funzionalità di cui possono usufruire gli utenti del sistema. | |
| SERVIZI OFFERTI | | |
| SERVIZIO | | DESCRZIONE |
| Registrazione | | Permette al cliente di inserire le proprie generalità e il codice del tavolo per poter accedere al menù ed effettuare un ordine |
| Prenotazione tavolo | | Permette al cliente di prenotare un tavolo scegliendo data, ora e numero di posti |
| Login | | Permette allo staff di poter accedere alla propria area del sistema |
| Logout | | Permette allo staff di disconnettersi dal sistema |
| Visualizzazione dati | | Permette di visualizzare la clientela di un determinato intervallo temporale |
| Eliminazione utente | | Permette di rimuovere un membro dello staff dal sistema |
| Generazione codice | | Permette di generare un codice univoco per il tavolo |

## **4.2 Gestione Comande**

|  |  |  |
| --- | --- | --- |
| DESCRZIONE: | Sottosistema che gestisce e raccoglie le funzionalità che riguardano la gestione delle comande inviate in cucina. | |
| SERVIZI OFFERTI | | |
| SERVIZIO | | DESCRZIONE |
| Aggiungi prodotto | | Permette al cliente di aggiungere portate alla propria comanda |
| Rimuovi prodotto | | Permette al cliente di rimuovere portate dalla comanda |
| Riepilogo comanda | | Permette al cliente di visualizzare la comanda prima di inviarla ed effettuare modifiche |
| Conferma ordine | | Permette al cliente di inviare la comanda in cucina |

## **4.3 Gestione Cucina**

|  |  |  |
| --- | --- | --- |
| DESCRZIONE: | Sottosistema che gestisce e raccoglie le funzionalità che riguardano la gestione, l’esecuzione e il lavoro della cucina | |
| SERVIZI OFFERTI | | |
| SERVIZIO | | DESCRZIONE |
| Visualizzazione comanda | | Permette di visualizzare in tempo reale tutte le comande che arrivano in cucina |
| Accettazione comanda | | Permette di prendere in carico una comanda |
| Conferma comanda | | Permette di notificare il completamento della comanda |

## **4.4 Gestione Prodotti/Menù**

|  |  |  |
| --- | --- | --- |
| DESCRZIONE: | Sottosistema che gestisce e raccoglie le funzionalità che riguardano la gestione, l’aggiornamento e la consultazione del menù. | |
| SERVIZI OFFERTI | | |
| SERVIZIO | | DESCRZIONE |
| Aggiungi prodotto | | Permette al gestore di aggiungere un prodotto al menù |
| Modifica prodotto | | Permette al gestore di modificare le caratteristiche di un prodotto sul menù |
| Rimuovi prodotto | | Permette al gestore di rimuovere un prodotto dal menù |
| Genera pietanza casuale | | Permette al cliente di generare una portata in modo casuale |
| Visualizza per filtro | | Permette al cliente di visualizzare le portate secondo un determinato filtro |
| Visualizza per categoria | | Permette al cliente di visualizzare i prodotti secondo una determinata categoria |

# 5. 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.

# 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.