Università degli Studi di Salerno

Corso di Ingegneria Del Software

Eat & Reorder

System Design Document

**Docente:**

Prof. Andrea De Lucia

**Studenti:**

Abate Francesco

Dello Buono Marco

De Martino Vincenzo

Gagliardi Rosario

Sommario

[1. Introduzione 3](#_Toc30427494)

[1.1 Scopo del sistema 3](#_Toc30427495)

[1.2 Design Goals 4](#_Toc30427496)

[1.2.1 DG1 Criteri di Usabilità 4](#_Toc30427497)

[1.2.2 DG2 Criteri di Affidabilità 4](#_Toc30427498)

[1.2.3 DG3 Criteri di Performance 5](#_Toc30427499)

[1.2.4 DG4 Criteri di Modularità 5](#_Toc30427500)

[1.2.5 DG5 Criteri di Implementazione 5](#_Toc30427501)

[1.3 Definizioni, acronimi ed abbreviazioni 5](#_Toc30427502)

[1.4 Riferimenti 5](#_Toc30427503)

[1.5 Panoramica 5](#_Toc30427504)

[2. Architettura del Sistema 6](#_Toc30427505)

[2.1 Panoramica 6](#_Toc30427506)

[2.2 Decomposizione del sistema 6](#_Toc30427507)

[2.3 Gestione dei dati persistenti 7](#_Toc30427508)

[2.4 Matrici di accesso 8](#_Toc30427509)

[2.4.1 Matrice degli accessi – Gestione Utente 9](#_Toc30427510)

[2.4.2 Matrice degli accessi – Gestione Ordini 10](#_Toc30427511)

[2.4.3 Matrice degli accessi – Gestione Mail 10](#_Toc30427512)

[2.4.4 Access Control List 10](#_Toc30427513)

[2.5 Mapping Hardware/Software 11](#_Toc30427514)

[2.6 Boundary Conditions 12](#_Toc30427515)

[2.6.1 First Start-up 12](#_Toc30427516)

[2.6.2 Start-up (a seguito di un fallimento) 12](#_Toc30427517)

[2.6.3 Terminazione 12](#_Toc30427518)

[Servizi dei sottosistemi 13](#_Toc30427519)

[2.7 Servizi della Gestione Utente 13](#_Toc30427520)

[2.8 Servizi della Gestione Ordine 14](#_Toc30427521)

[2.9 Servizi del Gestore Mail 15](#_Toc30427522)

# Introduzione

## Scopo del sistema

Il nostro progetto permette all’utenza di ordinare online piatti dalle aziende della propria città. L’obiettivo principale è quello di agevolare le ordinazioni per il cliente e l’azienda, fornendo dei fattorini che si occuperanno della consegna. La piattaforma sarà estremamente semplice da utilizzare, offrendo un’interfaccia minimale ed intuitiva, in modo tale che qualsiasi utente riesca ad effettuare ogni operazione senza alcun problema. Per rendere possibile ciò, verrà creata una Web App compatibile con qualsiasi dispositivo dotato di browser web, in modo da assicurare un semplice utilizzo a chiunque la provi.

Il nostro sistema fornisce determinate funzioni in base al tipo di utenza.

1. **Funzioni riguardanti l’utenza:**

* **Utente non registrato:**
* Registrazione
* **Cliente, Azienda, Fattorino:**
  + Modifica del profilo
  + Visualizzazione del profilo
  + Login
  + Logout
* **Cliente:**
  + Inserimento di una segnalazione
* **Moderatore:**
  + Gestione di una segnalazione

1. **Funzioni riguardanti l’acquisto:**

* **Cliente, Utente non Registrato**:
  + Ricerca di un’azienda
  + Visualizzazione del listino di un’azienda
  + Visualizzazione di un prodotto
* **Cliente**:
  + Aggiunta di un prodotto al carrello
  + Modifica della quantità di un prodotto al carrello
  + Rimozione di un prodotto dal carrello
  + Visualizzazione del carrello
  + Creazione dell’ordine

1. **Funzioni riguardanti la gestione di un ristorante:**

* **Azienda**:
  + Visualizzazione dei prodotti
  + Inserimento di un prodotto nel listino
  + Modifica di un prodotto nel listino
  + Rimozione di un prodotto dal listino
  + Visualizzazione degli ordini commissionati
  + Visualizzazione dei dettagli di un ordine

1. **Funzioni riguardanti la consegna**:

* **Fattorino**:
  + Visualizzazione delle consegne
  + Visualizzazione dei dettagli di una consegna
  + Conferma del ritiro di un ordine
  + Conferma della consegna di un ordine

## Design Goals

La Web App Eat&Reorder punterà all’essere il più lineare ed intuitiva possibile, in modo da offrire un’interfaccia grafica lineare ed intuitiva. Il sistema sottolinea, quindi, tramite la sua semplice interfaccia grafica, tutte le funzioni disponibili all’utente, il quale difficilmente troverà difficoltà nell’utilizzo di esse, evitando problemi ed inserimenti di dati errati.

Il sistema proposto rispetterà i seguenti **Criteri di Design**:

### DG1 Criteri di Usabilità

* + **DG1.1** – Semplicità: il sistema permetterà all’utente di effettuare qualsiasi operazione ad egli inerente senza riscontrare problemi e con estrema semplicità.
  + **DG1.2** – Interfaccia grafica: il sistema offrirà un’interfaccia grafica semplice e minimale, onde evitare confusione nell’utilizzo della piattaforma.

### DG2 Criteri di Affidabilità

* + **DG2.1** – Consistenza dei dati: il sistema registra dati e transazioni, garantendo il loro salvataggio in un database e la loro usabilità nell’applicazione.
  + **DG2.2** – Il sistema deve garantire che i dati inseriti dall’utente siano corretti, in modo da non immettere dati errati all’interno del database.
  + **DG2.3** – Correttezza: il sistema produrrà sempre l’output atteso in base all’operazione svolta dall’utente, evitando errori indesiderati.
  + **DG2.4** – Disponibilità: il sistema è disponibile agli utenti 24 ore su 24 e 7 giorni su 7 con alcune limitazioni. I clienti potranno effettuare acquisti prezzo l’azienda scelta solamente durante i giorni e gli orari indicati da quest’ultima.
  + **DG2.5** – Sicurezza: il sistema garantisce all’utenza lo svolgimento delle operazioni inerenti solamente alla tipologia di utente che lo utilizza.

### DG3 Criteri di Performance

* + **DG3.1** – Tempi di risposta: il sistema deve garantire tempi di risposta relativamente brevi per ogni funzionalità. Mediamente una richiesta dovrà esser soddisfatta in meno di 5 secondi. Ovviamente tale tempo può dipendere dalla velocità della connessione.
  + **DG3.2** – Throughput: il sistema sarà capace di gestire una media di 100 utenti, consentendo loro di effettuare operazioni senza alcun rallentamento.
  + **DG3.3** – Dati: i dati salvati dal sistema verranno ricavati nel minor tempo possibile.

### DG4 Criteri di Modularità

* + **DG4.1** – Modificabilità: il sistema permette di apportare modifiche alle funzionalità già implementate senza la necessità di modificare i sottosistemi.

### DG5 Criteri di Implementazione

* + **DG5.1** – Implementazione: il sistema, implementerà tutti i requisiti funzionali definiti in fase di analisi. In future versioni potranno essere incrementate o rivalutate le funzionalità specificate.

## Definizioni, acronimi ed abbreviazioni

* **DG**: Design Goal
* **RAD**: Requirements Analysis Document
* **MVC**: Model View Pattern
* **DBD**: Database Document

## Riferimenti

Eat&Reorder RAD

Object Oriented Software Engineering Using UML

## Panoramica

Il documento si compone di **3 parti fondamentali**:

1. Introduzione: vengono illustrati gli obiettivi del sistema proposto e vengono descritte le varie funzionalità messe a disposizione per i diversi attori.
2. Architettura del sistema proposto: viene illustrata la decomposizione del sistema.
3. Servizi dei sottosistemi e glossario: viene illustrata la descrizione dei servizi forniti dal sistema.

# Architettura del Sistema

## Panoramica

Il sistema prevederà l’utilizzo di un’architettura ibrida tra **Client-Server** ed **MVC**: il lato Client si occuperà delle pagine di presentazione del sistema, quindi delle view, mentre il lato Server si occuperà della logica applicativa e di controllo, quindi del model e del controller. Per il server sarà utilizzato il WebServer Apache Tomcat 9, per la gestione dei dati sarà utilizzato il DBMS MySQL.

## Decomposizione del sistema

Per semplificare la progettazione e lo sviluppo dell’applicazione, i sottosistemi saranno decomposti secondo lo schema previsto dalla seguente architettura: Presentation, Application e Storage.

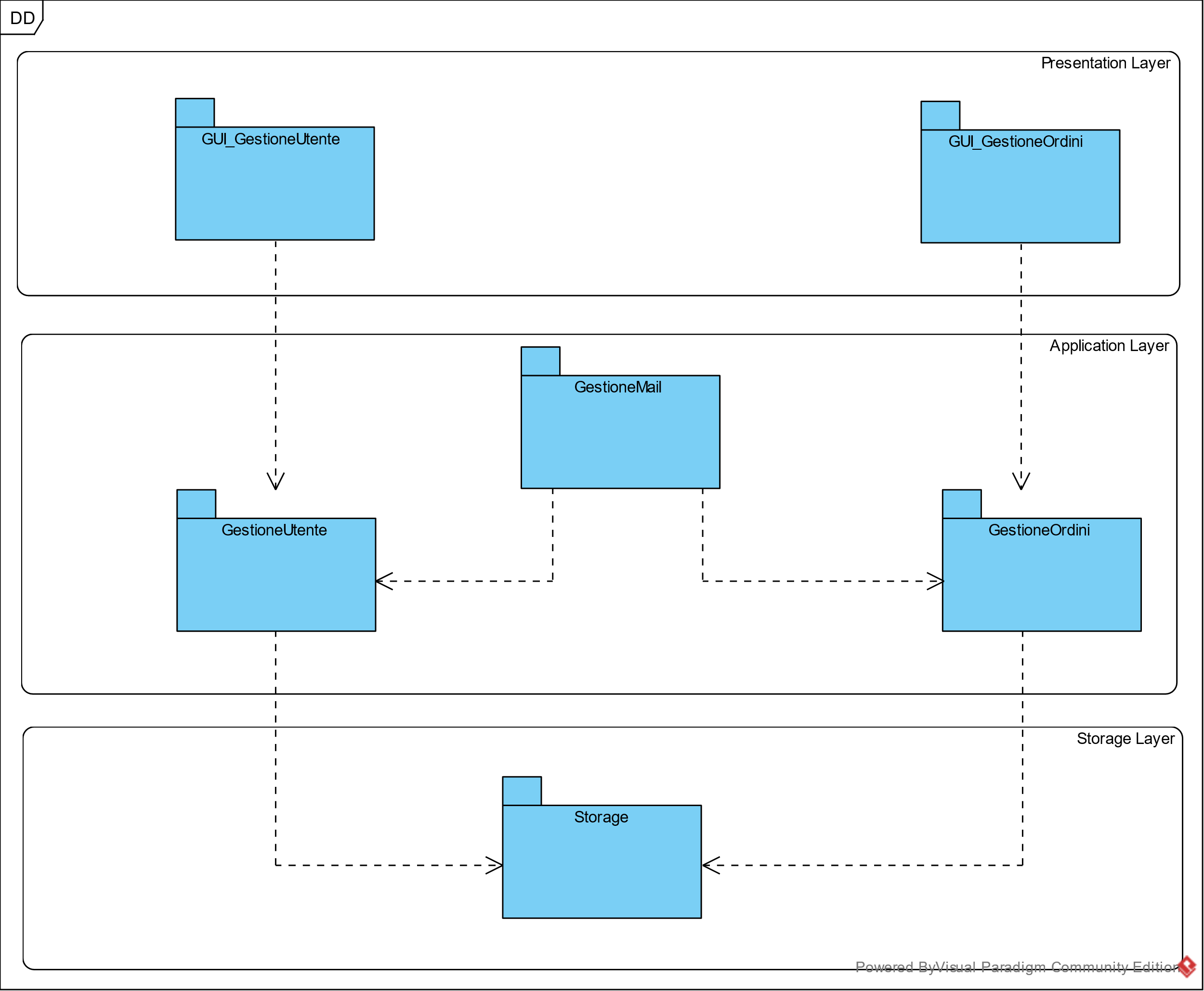
Il **livello Presentation** si occupa di gestire l’interfaccia grafica del sottosistema e contiene tutti gli elementi che interagiscono con l’utente.

Il **livello Application** si occupa di elaborare determinati input e di fornire risultati che verranno mostrati graficamente tramite il livello Presentation.

Il **livello Storage** si occupa di conservare i dati degli utenti che interagiscono con il sistema.

Il sistema è diviso in due sottosistemi: **GestioneUtente** e **GestioneOrdini**.

Il sistema prevede inoltre un ulteriore sottosistema **GestioneMail** che sarà presente solo a livello di logica applicativa.



Il sottosistema “**gestione utente**” si occupa dell’ottenimento, della modifica e dell’inserimento degli utenti: sostanzialmente, quindi, effettua tutte le operazioni che riguardino l’utenza.

Il sottosistema “**gestione ordini**” si occupa dell’ottenimento, della modifica e dell’inserimento degli ordini e delle consegne: sostanzialmente, quindi, effettua tutte le operazioni che riguardino ordini e consegne.

Il sottosistema “**gestione mail**” si occupa dell’invio delle email: sostanzialmente, quindi, effettua tutte le operazioni che riguardino l’invio di e-mail.

## Gestione dei dati persistenti

Riguardo la persistenza dei dati, si farà riferimento al **DBD: Database Document.**

In tale documento viene specificato il diagramma Entity Relationship e le tabelle del database, le quali specificano a loro volta i tipi di dati che compongono ogni tabella e le associazioni tra loro.

Inoltre, vengono definite tutte le query eseguite nel sistema.

## Matrici di accesso

Essendo Eat&Reorder un sistema con differenti tipi di utenza, essi possono accedere a diverse funzionalità a seconda del tipo di utenza che si assume in quel momento.

Per documentare le funzionalità che ogni utente possiede, si rappresenta una matrice che suddivide le funzionalità in base all’utente.

Le matrici di accesso saranno rappresentate nelle prossime pagine.

### Matrice degli accessi – Gestione Utente

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attori  Oggetti | **Cliente** | **Azienda** | **Fattorino** | **Moderatore** | **UtenteNonRegistrato** |
| GestoreUtente | aggiornaCliente()  creaSegnalazione()  visualizzaProfilo()  ricercaAziende()  visualizzaListino()  visualizzaDettagliProdotto()  login()  logout() | aggiornaAzienda()  visualizzaProfilo()  inserisciProdotto()  modificaProdotto()  rimuoviProdotto()  visualizzaMenu()  login()  logout() | aggiornaFattorino()  visualizzaProfilo()  login()  logout() | gestisciSegnalazione()  login()  logout() | registraCliente()  registraAzienda()  registraFattorino()  ricercaAziende()  visualizzaListino()  visualizzaDettagliProdotto() |

### Matrice degli accessi – Gestione Ordini

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attori  Oggetti | **Cliente** | **Azienda** | **Fattorino** | **UtenteNonRegistrato** |
| Carrello | aggiungiAlCarrello()  modificaQuantitaCarrello()  rimuoviDalCarrello()  visualizzaCarrello() |  |  |  |
| GestoreOrdini | ordina() | visualizzaOrdini()  visualizzaDettagliOrdine() | visualizzaConsegne()  confermaRitiro()  confermaConsegna() |  |

### Matrice degli accessi – Gestione Mail

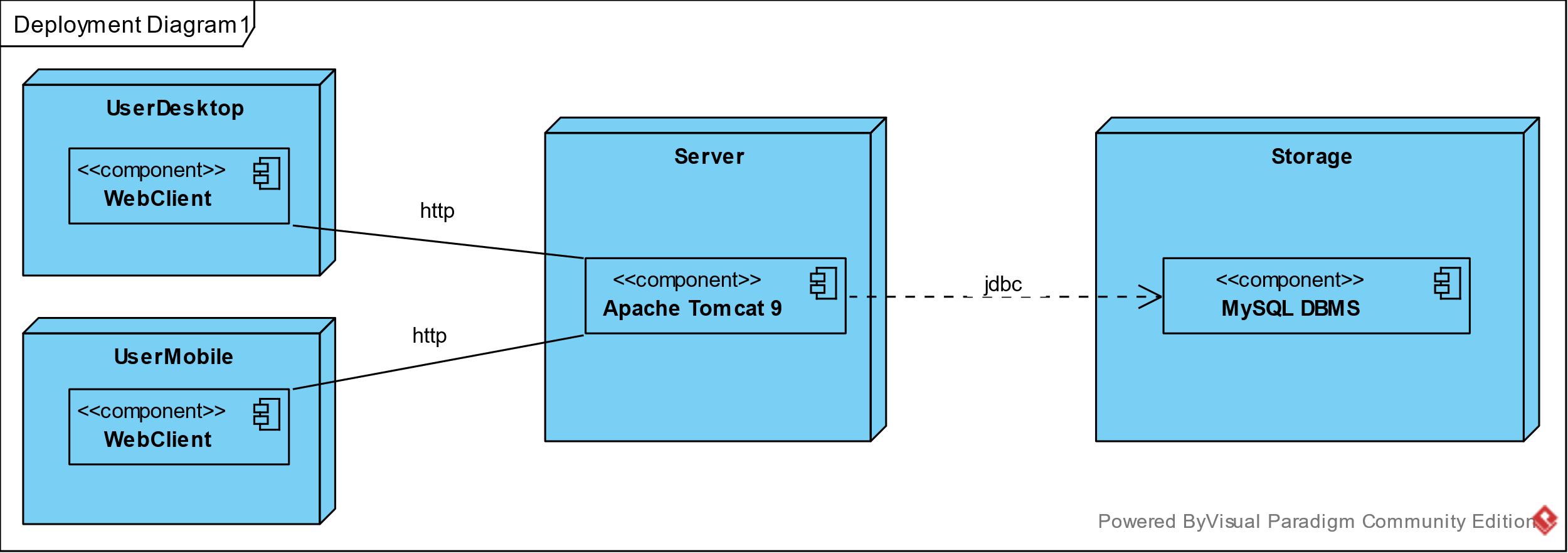
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attori  Oggetti | **Cliente** | **Azienda** | **Fattorino** | **Moderatore** | **UtenteNonRegistrato** |
| GestoreMail | inviaMailOrdine()  inviaMailModeratore() |  |  | inviaConfermaBan() |  |

### Access Control List

Nel sistema non sarà implementata una access control list.

Tuttavia il sistema di controllo degli accessi alle varie funzionalità sarà assicurato tramite il layer di controllo che si occuperà di gestire gli accessi degli utenti alle varie funzionalità.

## Mapping Hardware/Software



Il sistema sarà strutturato su una architettura a tre livelli usato tipicamente da numerose web application. La struttura a tre livelli include un presentation layer, un layer di logica di business e un data layer.

Il client consiste di un personal computer o di un dispositivo mobile(smartphone,tablet) che fornisce all’utente l’interfaccia grafica del sistema. Genera inoltre le richieste al server tramite il protocollo HTTP e visualizza le pagine HTML nel browser.

Il server consiste di un web server che riceve le richieste dal cliente e le processa. Per fare ciò il web server fa richieste tramite jdbc al livello di storage, gestisce i dati e fornisce i risultati al client.

Il data layer consiste di un database SQL relazionale gestito tramite il DBMS MySQL che gestisce la persistenza dei dati degli utenti e degli ordini.

La scelta di questo tipo di architettura consente di diminuire i tempi di manutenzione e aumenta la flessibilità del sistema software.

## Boundary Conditions

### First Start-up

Per il primo start-up del sistema è necessario l'avvio del DMBS che fornisca il servizio di un Database MySQL per la gestione dei dati persistenti. L’amministratore dovrà dunque inserire all’interno del database almeno un account Moderatore fornendo una mail e una password in quanto tale account non può essere creato dall’interfaccia del sistema. Tale account sarà poi fornito all’utente moderatore responsabile della piattaforma. L’amministratore inoltre dovrà avviare la macchina necessaria all’esecuzione del web-server.

### Start-up (a seguito di un fallimento)

Nel caso in cui si verifichi un'interruzione inaspettata dell'alimentazione, non sono previsti metodi che ripristinino lo stato del sistema a un stato antecedente allo spegnimento inaspettato.

Un altro caso di fallimento potrebbe derivare dal software stesso che causa un crash inaspettato dovuto ad errori commessi durante la fase di implementazione, e non sono previste politiche correttive, l’unico processo che potrà essere eseguito è la chiusura del sistema e il suo successivo riavvio.

### Terminazione

Al momento della chiusura dell’applicativo basterà effettuare lo spegnimento del DBMS e del web server. Viene assicurata la consistenza dei dati, annullando eventuali operazioni ancora in esecuzione.

# Servizi dei sottosistemi

## Servizi della Gestione Utente

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| registraCliente() | Consente all’utente di registrarsi come Cliente |
| registraAzienda() | Consente all’utente di registrarsi come Azienda |
| egistraFattorino() | Consente all’utente di registrarsi come Fattorino |
| login() | Consente ad un utente registrato di autenticarsi al sistema |
| logout() | Consente ad un utente autenticato di uscire dal sistema |
| aggiornaCliente() | Consente al cliente di modificare il profilo personale |
| aggiornaAzienda() | Consente all’azienda di modificare il proprio profilo personale |
| aggiornaFattorino() | Consente al fattorino di modificare il profilo personale |
| creaSegnalazione() | Consente al cliente di creare una segnalazione verso un azienda |
| gestisciSegnalazione() | Consente al moderatore di bannare una azienda |
| visualizzaProfilo() | Consente ad un utente registrato di visualizzare il proprio profilo |
| inserisciProdotto() | Consente all’azienda di aggiungere prodotti al proprio menu |
| modificaProdotto() | Consente all’azienda di modificare un prodotto del proprio menu |
| rimuoviProdotto() | Consente all’azienda di eliminare un prodotto dal proprio menu |
| visualizzaMenu() | Consente all’azienda di visualizzare il proprio menu |
| ricercaAziende() | Consente al cliente o all’utente non registrato di ricercare le aziende in base alla città specificata |
| visualizzaListino() | Consente al cliente o all’utente non registrato di visualizzare il listino di una azienda cercata |
| visualizzaDettagliProdotto() | Consente al cliente o all’utente non registrato di visualizzare i dettagli di un prodotto presente nel listino di una azienda cercata |

## Servizi della Gestione Ordine

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| aggiungiAlCarrello() | Consente al cliente di aggiungere un prodotto al carrello |
| modificaQuantitàCarrello() | Consente al cliente di modificare la quantità di un prodotto presente nel carrello |
| rimuoviDalCarrello() | Consente al cliente di eliminare un prodotto dal carrello |
| visualizzaCarrello() | Consente al cliente di visualizzare il carrello |
| ordina() | Consente al cliente di effettuare un ordine con i prodotti del carrello |
| visualizzaOrdini() | Consente all’azienda di visualizzare la lista degli ordini da preparare |
| visualizzaDettagliOrdine() | Consente all’azienda di visualizzare il dettaglio di un ordine da preparare e al Fattorino di visualizzare il dettaglio di un ordine da consegnare |
| visualizzaConsegne() | Consente al Fattorino di visualizzare gli ordini da consegnare |
| confermaRitiro() | Consente al Fattorino di confermare il ritiro di un ordine |
| confermaConsegna() | Consente al Fattorino di confermare la consegna di un ordine |

## Servizi del Gestore Mail

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| inviaMailModeratore() | Consente di inviare una mail al moderatore |
| inviaMailOrdine() | Consente di inviare una mail con l’ordine al Cliente |
| inviaConfermaBan() | Consente di inviare una mail di avvenuto ban dell’Azienda |