
UNIVERSITÀ DEGLI STUDI DI SALERNO

# Warehouse project

# System Design Document

Top Manager:

| NOME |
| --- |
| Prof. De Lucia Andrea |

Partecipanti:

| NOME | MATRICOLA |
| --- | --- |
| Michele Gargiulo | 0512105530 |

## Indice



# 1. Introduzione

## 1.1 Scopo del sistema

Nel mondo odierno i dispositivi elettronici sono sempre più protagonisti, questo è dovuto soprattutto al fenomeno chiamato "e-commerce".

Al contrario del business off-line, un e-commerce abbatte tutte le barriere di tipo geografico raggiungendo nuovi paesi e nuovi mercati. Inoltre, l'e-commerce, conta sul grande vantaggio del traffico generato dai motori di ricerca.

In una società ormai popolata da dispositivi elettronici, bisogna offrire la possibilità al cliente di acquistare il prodotto desiderato in modo facile, veloce, sicuro e al prezzo più basso sul mercato.

La web application garantirà ai clienti l'acquisto online di vari prodotti, la visione della merce disponibile, suddivisa in categorie e accessibile tramite menù di navigazione, e un supporto online in caso di problematiche.

## 1.2 Obiettivi di design

Il sistema Warehouse 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. 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** | Uno degli aspetti principali di Warehouse è il tempo di risposta, che quindi, deve assicurare una risposta rapida alle varie richieste degli utenti. In casi di connessione lenta, si prevede un tempo massimo per gestire l'intera richiesta di 10 secondi, e questo tempo va a diminuire con l'aumento della velocità di connessione |
| **Throughput** | Il sistema sarà in grado di gestire più utenti in modo concorrente. Non è prevedibile ora stimare il carico giornaliero del sistema, ma, possiamo prevedere un carico iniziale di 50 utenti. |
| **Memoria** | Calcolare delle stime per capire le informazioni da salvare all'interno del database del nostro sistema è difficile, ma possiamo prevedere che in media ogni utente del sistema può creare 5 ordini. |

### *1.2.2 Criteri di Affidabilità*

| | |
|---|---|
| **Robustezza** | Warehouse deve gestire eventuali input errati da parte dell'utente, attivando politiche di controllo sia lato Client, tramite Javascript, che lato server, utilizzando le servlet. Il tutto deve avvenire senza interrompere il funzionamento dell'intero sistema. |
| **Disponibilità** | Warehouse è pensato per essere disponibile ovunque, in qualsiasi momento e accessibile da qualsiasi dispositivo. Quindi, il sistema deve essere sempre disponibile. |
| **Tolleranza all'errore** | Warehouse deve tollerare gli errori imprevisti nel sistema, continuando ad operare come se non ci fossero problemi, in modo trasparente all'utente. Ciò viene realizzato, grazie ad un basso grado di accoppiamento tra i vari sottosistemi, così da |

| | |
|---|---|
| | non propagare gli errori in tutto il sistema |
| **Sicurezza** | Warehouse prevede il controllo degli accessi e dell'autenticazione, permettendo l'uso delle varie funzionalità della piattaforma alle categorie di utente specificate nella tabella degli accessi |

### *1.2.3 Criteri di Manutenzione*

| | |
|---|---|
| **Estendibilità** | Warehouse deve essere progettato per accogliere al meglio nuove funzionalità, integrandole al meglio con i moduli già presenti. Quindi è necessario che il codice sia ben strutturato in moduli. |
| **Modificabilità** | Deve essere possibile modificare il codice in qualsiasi momento, per apportare modifiche o aggiustamenti. Questo comporta che il codice deve essere ben strutturato. |
| **Leggibilità** | Il codice risulterà agevole da leggere grazie all'indentazione e sarà accompagnato da relativi commenti per comprendere al meglio cosa offre |

### *1.2.4 Criteri per l'Utente Finale*

| | |
|---|---|
| **Usabilità** | Warehouse offre ogni sua funzionalità in modo semplice ed intuitivo, garantendo una piacevole interazione tra l'utente e il sistema |

## 1.3 Definizioni, acronimi e abbreviazioni

- Warehouse: Nome del sistema che verrà sviluppato.
- RAD: Requirement Analysis Document
- DBMS: Database Management System

## 1.4 Riferimenti

Documento Requirement Analysis Document - Warehouse

## 1.5 Panoramica

Le attività di system design che costituiscono le fondamenta per l'architettura software del sistema:

- Decomposizione del sistema, in cui il sistema viene suddiviso in diversi sottosistemi ognuno dei quali, a sua volta, è caratterizzato da servizi che offre ad altri sottosistemi.
- Mapping Hardware/Software, riguardante la scelta della configurazione hardware del sistema, la comunicazione tra nodi, il come vengano incapsulati i servizi di un sottosistema.

- Gestione dei dati persistenti, nel quale di individuano gli oggetti che devono essere resi persistenti e quale genere di infrastruttura si deve usare per memorizzare tali oggetti.
- Politiche di accesso e Sicurezza, che ci aiuta a rappresentare tramite delle tabelle le operazioni ed informazioni utilizzabili da ogni singolo attore.
- Controllo del software globale, che ci guida su quali operazioni eseguire ed in che ordine, per garantire il corretto flusso di controllo del sistema.
- Condizioni Boundary, che includono oltre l'avvio e lo shutdown anche la gestione dei fallimenti dovuti all'invecchiamento del sistema, interruzione di corrente o anche a errori di progettazione.

# 2. Architettura software corrente

Attualmente non esiste un sistema software che si occupa di gestire questa problematica, ossia l'amministrazione dei vari documenti che vengono utilizzati dall'università, quindi si tratta di un sistema che rientra nel campo della Greenfield Engineering.

Infatti, in un Greenfield Project lo sviluppo comincia da zero, non esiste nessun sistema a priori e i requisiti sono ottenuti dall'utente finale e dal cliente. Nasce, perciò, a partire dai bisogni dell'utente.

# 3. Architettura software proposta

## 3.1 Panoramica

L'architettura del sistema Warehouse 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, da computer fissi a dispositivi mobili.
● Performance: il client sarà in grado di supportare task interattivi e il server dovrà fornire operazioni CPU-intensive.
● 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 anche in seguito a guasti.

## 3.2 Decomposizione in layer

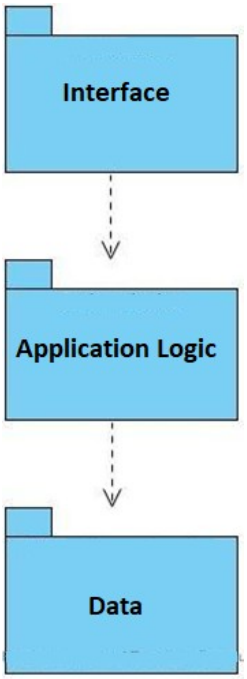
Le funzionalità saranno divise in layer logici in base alle differenti funzionalità utilizzando il modello MVC:

- Interface layer (View)
- Application Logic layer (Controller)
- Data layer (Model)

Interface layer rappresenta l'interfaccia che permette all'utente di interagire con il sistema, ricoprendo il ruolo di client in quanto utilizza un browser per richiedere pagine web al server.

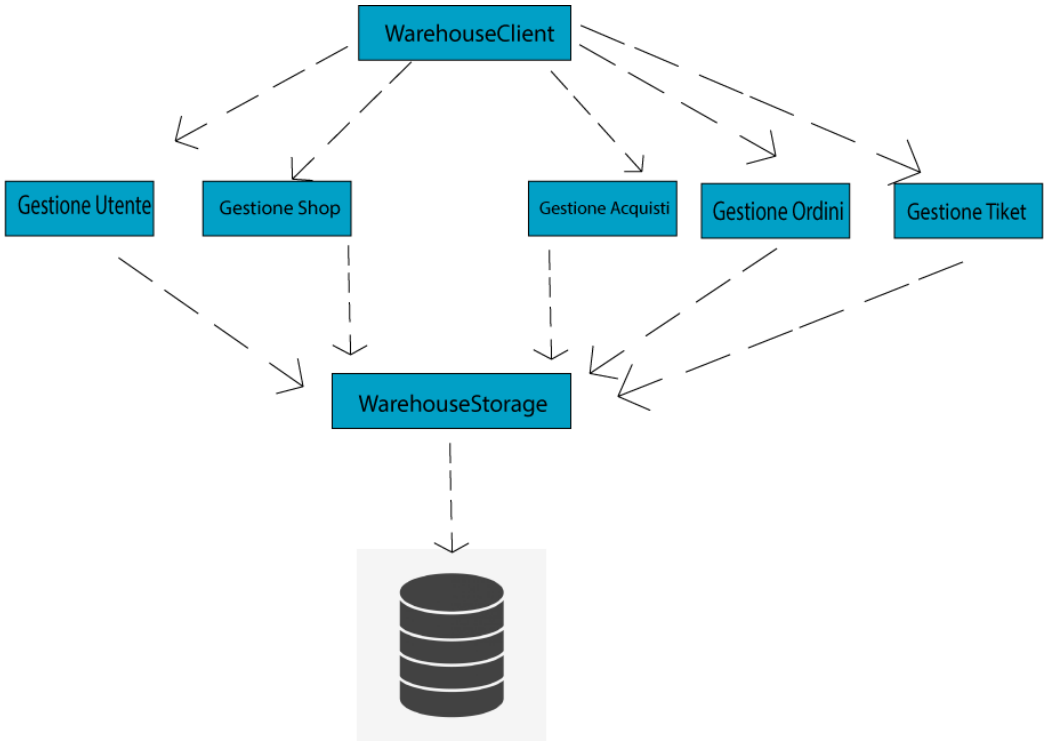
Application Logic layer ha il compito di elaborare i dati da inviare al client. Spesso interroga il database, tramite il Data Layer, per accedere ai dati persistenti.

Data layer ha il compito di memorizzare i dati sensibili del sistema, utilizzando un DBMS,
inoltre, riceve le varie richieste dall' Application Logic layer inoltrandole al DBMS e restituendo i dati richiesti.




## 3.3 Decomposizione in sottosistemi

Lo scopo di questa attività è l'individuazione di sottosistemi a partire dai requisiti funzionali e dal modello di analisi. Per raggiungere questo obiettivo dividiamo il sistema in componenti che possono essere gestite in maniera individuale.




Il sistema si compone di sette sottosistemi che si occupano di gestire aspetti e funzionalità differenti:

| **Interface** | |
|---|---|
| WarehouseClient | questo sottosistema si occupa di gestire le funzionalità del front end del sistema per gli utenti non loggati, gli utenti loggati, l'amministratore e i moderatori. |

| **Application Logic** | |
|---|---|
| GestioneShop | questo sottosistema si occupa della gestione della merce, fra cui la navigazione della sezione Shop, la ricerca della merce secondo dei criteri, l'inserimento dei prodotti e la modifica di quelli esistenti. |
| GestioneUtente | questo sottosistema si occupa della gestione degli utenti, fra cui la registrazione di un utente al sistema e la sua autenticazione, la gestione delle informazioni relative al profilo di un utente, quali la gestione delle informazioni anagrafiche |

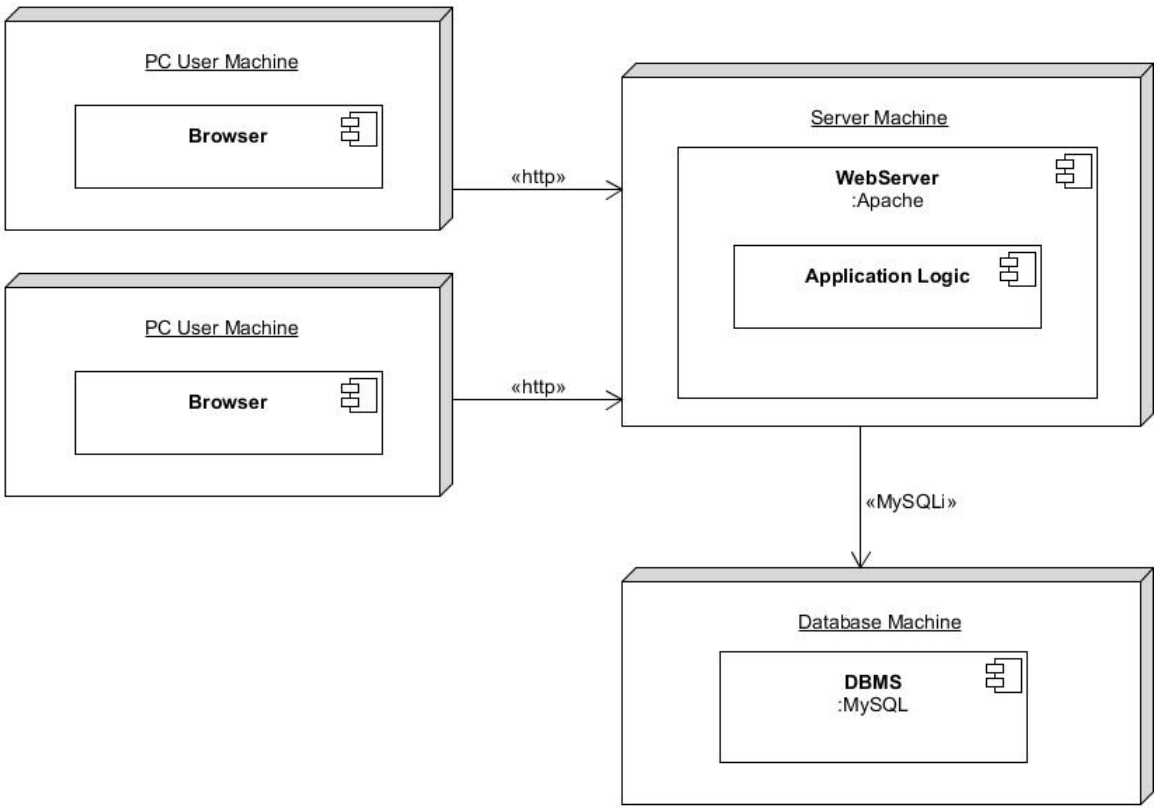
| | |
|---|---|
| | con la possibilità di eventuali modifiche. Si occupa, inoltre, della cancellazione del profilo di un utente da parte dell'amministratore con i relativi "dati sensibili" e la modifica della propria password. |
| GestioneAcquisti | questo sottosistema si occupa della gestione gli acquisti, fra cui l'inserimento di un prodotto all'interno del carrello, la rimozione di tutti o alcuni prodotti dal carrello, la visualizzazione di tutti i prodotti inseriti nel carrello con il prezzo totale e la finalizzazione dell'acquisto tramite un metodo di pagamento. |
| GestioneOrdini | questo sottosistema si occupa della gestione degli ordini, fra cui la visualizzazione degli ordini di un utente, la ricerca e la modifica degli ordini da parte di un operatore e l'annullamento di un ordine da parte di un utente. |
| GestioneTicket | questo sottosistema si occupa della gestione dei ticket, fra cui la creazione di un ticket di per un oggetto, la visualizzazione dei propri ticket da parte di un utente, la visualizzazione dei ticket aperti e chiusi, con la possibilità di modificare lo stato, da parte dell'Assistenza e lo scambio di messaggi tra Utente ed Assistenza. |

| **Storage** | |
|---|---|
| WarehouseStorage | questo sottosistema è responsabile di memorizzare oggetti persistenti. |

## 3.4 Mappatura hardware/software

Il sistema che si vuole sviluppare sarà diviso in client e server, per il DBMS verrà utilizzato MySQL, mentre per l'interfaccia si usufruirà del framework Bootstrap.
**Il client fornirà:** interfacce utente, elaborazione al front-end delle informazioni, gestione dei dati centralizzata, integrità, consistenza e sicurezza dei dati.
**Il server fornirà:** elaborazione centralizzata (archiviazione e interazione col database).

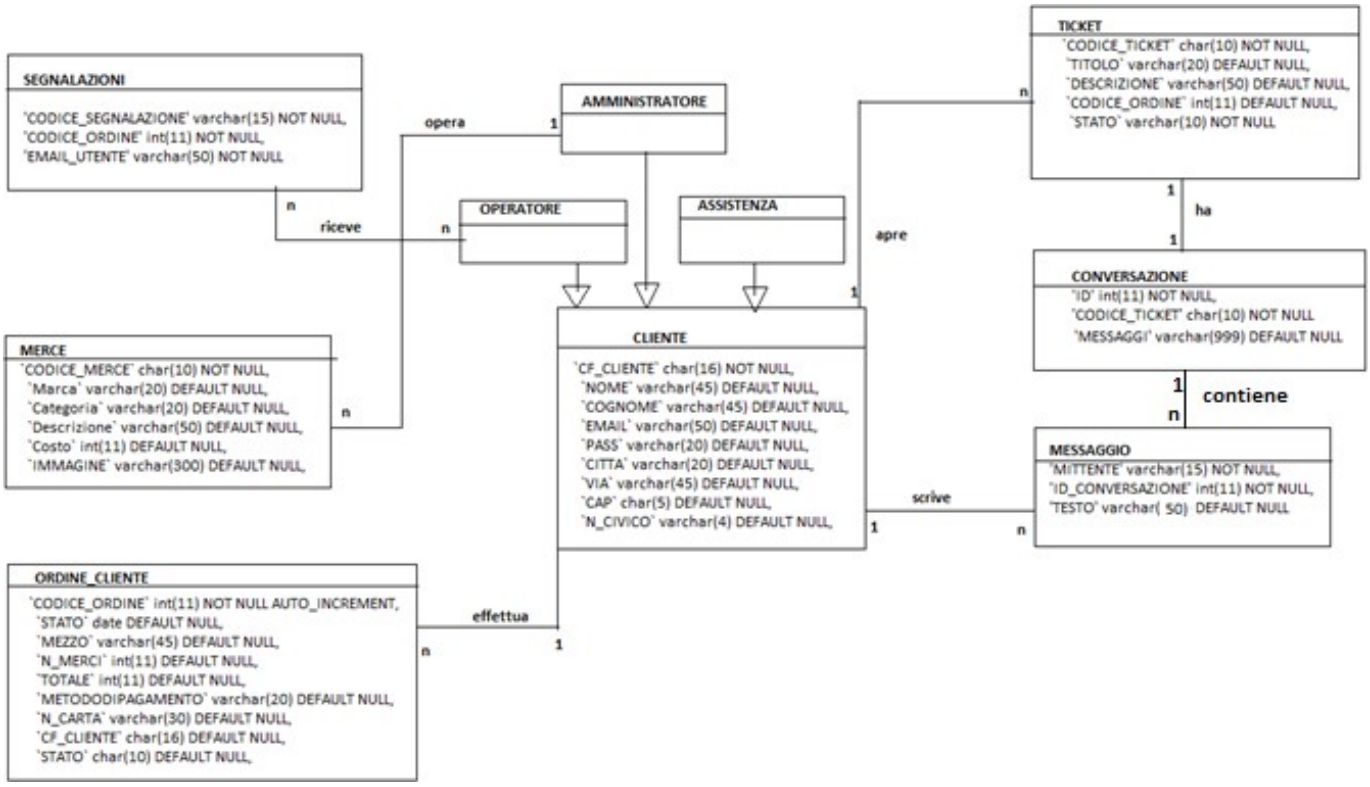
Nell'applicazione proposta i Client sono Web browser, mentre il Web Server contiene le componenti che si occupano della logica applicativa, dell'interfaccia e dei dati. Quando il Web server riceve una richiesta per una pagina, essa viene analizzata dall'interprete del linguaggio, il quale restituisce un file contenente solo il codice HTML e Java Script che deve essere inviato al Web browser. Il Web server invia query al Database MySQL utilizzando il protocollo MySQLi.




## 3.5 Gestione dei dati persistenti

Per la gestione dei dati persistenti si è deciso di utilizzare un database relazionale poiché le informazioni che si andranno a memorizzare richiedono un accesso multiplo opportunamente gestito da questo tipo di memorizzazione.

### *3.5.1* Class Diagram




**Cliente:** questa classe contiene tutte le informazioni relative ad un utente registrato. La tipologia dell'utente può variare in Operatore, Assistenza e Amministratore.

**Merce:** questa classe contiene le informazioni necessarie per la memorizzazione di un prodotto. La creazione di quest'ultimo viene effettuata dall'Amministratore.

**Ordine_Cliente:** questa classe contiene le informazioni relative ad un acquisto effettuato, da parte di un cliente

**Messaggio:** la classe Messaggio mantiene le informazioni relative ad un Messaggio inviato da un Utente. Questo viene collegato a Conversazione in quanto tutti i messaggi sono racchiusi all'interno di una conversazione.

**Conversazione:** la classe Conversazione mantiene le informazioni relative alle conversazioni di un cliente con l'Assistenza, per un determinato ordine. Ogni conversazione è collegata ad un ticket aperto da un cliente.

**Ticket:** la classe Ticket mantiene le informazioni relative alle controversie aperta da un cliente in merito ad un ordine effettuato. Ogni ticket contiene una conversazione unica tra cliente ed Assistenza.

**Segnalazioni:** la classe Segnalazioni mantiene le informazioni relative alle segnalazioni dell'assistenza, mosse verso l'operatore, al fine di aggiornare lo stato dell'ordine per il quale è stato aperto un ticket.

| **CLIENTE** | |
|---|---|
| **Campo** | **Vincolo** |

| | |
|---|---|
| CF_CLIENTE | Primary key, char(16), not null |
| nome | Varchar(45), default null |
| cognome | Varchar(45), default null |
| e-mail | Varchar(50), default null, UNIQUE KEY |
| pass | Varchar(20), not null |
| città | Varchar(20), default null |
| via | Varchar(45), default null |
| cap | Char(5), default null |
| N_civico | varchar(4), default null |
| Rank | Varchar(15), default null |

| **MERCE** | |
|---|---|
| **Campo** | **Vincolo** |
| CODICE_MERCE | Char(10), not null, primary key |
| Marca | Varchar(20), default null |
| Categoria | Varchar(20), default null |
| Descrizione | Varchar(50), default null |
| Costo | Int(11), default null |
| Immagine | Varchar(300), default null |

| **ORDINE_CLIENTE** |
|---|

| Campo | Vincolo |
|---|---|
| CODICE_ORDINE | Int(11), auto increment not null |
| DataOrdine | Date, default null |
| Mezzo | Varchar(45), default null |
| N_merci | Int(11), default null |
| Totale | Int(11), default null |
| MetodoDiPagamento | Varchar(20),default null |
| N_Carta | Varchar(30), default null |
| CF_CLIENTE | Char(16), default null, foreign key{CLIENTE} |
| Stato | Char(10), default null |

| **MESSAGGIO** | |
|---|---|
| **Campo** | **Vincolo** |
| MITTENTE | Vachar(20), not null, primary key |
| ID_CONVERSAZIONE | Int(11), not null, foreign key{CONVERSAZIONE} |
| Testo | Varchar(50), default null |

| **CONVERSAZIONE** | |
|---|---|
| **Campo** | **Vincolo** |
| id | Int(11) not nul, primary key |
| CODICE_TICKET | Char(10) not null, foreign key{TICKET} |
| Messaggi | Varchar(999), default null |

| **TICKET** |
|---|

| Campo | Vincolo |
|---|---|
| CODICE_TICKET | Char(10) not null, primary key |
| Titolo | Varchar(999), default null |
| Descrizione | Varchar(50), default null |
| CODICE_ORDINE | Int(11), default null, foreign key{ORDINE_CLIENTE} |
| Stato | Varchar(10), not null |

| SEGNALAZIONE | |
|---|---|
| **Campo** | **Vincolo** |
| CODICE_SEGNALAZIONE | Varchar(15), not null, primary key |
| CODICE_ORDINE | Int(11), not null, foreign key{ORDINE_CLIENTE} |
| Email_Assistenza | Varchar(50), not null, foreign key{CLIENTE} |

## 3.6 Controllo degli accessi e sicurezza

Warehouse è un sistema multiutente, quindi attori differenti hanno il permesso di eseguire diverse operazioni su vari insiemi di oggetti.

Il controllo degli accessi è garantito tramite l'utilizzo di username e password che verranno richieste per ogni singolo accesso a Warehouse.
Per schematizzare al meglio il controllo degli accessi abbiamo suddiviso per tipologia di utente le azioni consentite, al fine di ottenere una visione più compatta e dettagliata grazie ad una matrice degli accessi riportata di seguito:

| | **Utente frontend** | **Utente loggato** | **Operatore** | **Assistenza** | **Amministratore** |
|---|---|---|---|---|---|
| **Gestione Utente** | Crea account | Visualizza dati profilo, modifica dati profilo | Visualizza dati profilo, modifica dati profilo | Visualizza dati profilo, modifica dati profilo | Visualizza dati utente, modifica dati utente, rank-Up utente, cancella utente |
| **Gestione Shop** | Visualizza merce, ricerca merce | Visualizza merce, ricerca merce | Visualizza merce, ricerca merce | Visualizza merce, ricerca merce | Visualizza merce, ricerca merce, modifica merce, cancella merce |
| **Gestione Acquisti** | Aggiungi prodotto al carrello, rimuovi prodotto dal carrello | Aggiungi prodotto al carrello, rimuovi prodotto dal carrello, inserimento ordine | Nessuna operazione | Nessuna operazione | Nessuna operazione |
| **Gestione Ordini** | Nessuna operazione | Visualizza ordini, annulla un ordine | Visualizza ordini utenti, Modifica stato ordini | Nessuna operazione | Nessuna operazione |
| **Gestione Ticket** | Nessuna operazione | Apri ticket, visualizza ticket aperti e chiusi, visualizza conversazione ticket, invia messaggio, visualizza messaggio | Visualizza segnalazione assistenza. | Visualizza ticket utenti, chiudi ticket, visualizza conversazioni ticket, invia messaggio, ricevi messaggio, invia segnalazione | Nessuna operazione |

## 3.7 Controllo globale del software

Il sistema EP fornisce funzionalità che richiedono una continua interazione da parte dell'utente, per questo motivo abbiamo scelto come meccanismo di controllo del flusso globale l'event-driven control.

## 3.8 Boundary conditions

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.

### *3.8.1 Start-up*

Per il primo start-up del sistema Warehouse è necessario l'avvio di un web server che fornisca il servizio di un Database per la gestione dei dati persistenti e l'interpretazione ed esecuzione del codice lato server. In seguito, tramite l'interfaccia di Login, sarà possibile autenticarsi tramite opportune credenziali (username e password) come utente con accesso alle funzionalità del sistema. Una volta effettuato l'accesso, Warehouse presenterà la home relativa al tipo di utente loggato, dalla quale si potranno effettuare tutte le operazioni che il sistema gli fornisce.

### *3.8.2 Terminazione*

Al momento della chiusura dell'applicativo si ha la terminazione del sistema con un regolare Logout dal sistema. Viene assicurata la consistenza dei dati, annullando eventuali operazioni che erano in esecuzione.

### *3.8.3 Fallimento*

Possono verificarsi diversi casi di fallimento del sistema:

- Nel caso di guasti dovuti al sovraccarico del database con successivo fallimento dello stesso, è prevista come procedura preventiva il salvataggio periodico dei dati sotto forma di codice SQL per la successiva rigenerazione del database.
- Nel caso in cui si verifichi un'interruzione inaspettata dell'alimentazione, non sono previsti metodi che ripristinano lo stato del sistema a prima dello spegnimento inaspettato.
- Un altro caso di fallimento potrebbe derivare dal software stesso che causa una chiusura inaspettata dovuta ad errori commessi durante la fase di implementazione, non sono previste politiche correttive, l'unico processo che potrà essere eseguito è la chiusura del sistema e il suo successivo riavvio.
- Un altro caso di fallimento potrebbe essere dovuto ad un errore critico nell'hardware, non è prevista alcuna misura correttiva.

### *3.8.4 Casi d'uso*

| | |
|---|---|
| **ID** | UC_Startup |
| **Nome Caso Uso** | Startup |
| **Attori**<br><br>**Partecipanti** | Amministratore |
| **Condizione di Entrata** | L'amministratore accede al web server |
| **Flusso di Eventi** | Attore<br><br>1. L’amministratore accede al sistema e clicca sul pulsante “Avvia”.<br><br>Sistema<br><br>2. Warehouse accende il server e attiva i servizi in remoto rendendosi disponibile per le richieste notificando il successo dell’operazione all’utente. |
| **Condizione di Uscita** | Il server è attivo e i relativi servizi sono disponibili. |

| | |
|---|---|
| **ID** | UC_Shutdown |
| **Nome Caso Uso** | Shutdown |
| **Attori**<br><br>**Partecipanti** | Amministratore |
| **Condizione di Entrata** | L'amministratore accede al web server |
| **Flusso di Eventi** | Attore<br><br>1. L’amministratore accede al sistema e clicca sul pulsante “Spegni”.<br><br>Sistema<br><br>2. Warehouse effettua una scansione per verificare se ci sono eventuali richieste in sospeso, porta a termine le richieste e avvia le procedure di arresto. Il sistema notifica il successo dell’operazione all’utente. |
| **Condizione di Uscita** | Il server si è spento correttamente. |

# 4. Servizi dei sottosistemi

## 4.1 Gestione Utente

| Sottosistema | Descrizione |
|---|---|
| Gestione Utente | Sottosistema che gestisce la registrazione di un utente, la sua autenticazione e le operazioni necessarie alla sua gestione. |
| **Servizio** | **Descrizione** |
| insertUtente() | Permette di inserire un nuovo utente all'interno del database. |
| inoltraLogin() | Permette ad un utente di poter effettuare l'accesso al sistema. |
| inoltraLogout() | Permette ad un utente di uscire dal sistema. |
| deleteUser() | Permette all'amministratore di cancellare un utente |
| cambiaPassword() | Permette ad un utente di cambiare la propria password. |
| setRankUtente() | Permette all'amministratore di modificare il rank di un utente |
| modificaDati() | Permette ad un utente di modificare i propri dati. |
| selectUser() | Permette di cercare un utente all'interno del database. |
| getUsers() | Permette di visualizzare la lista di tutti gli utenti. |

## 4.2 Gestione Shop

| Sottosistema | Descrizione |
|---|---|
| Gestione Shop | Sottosistema che si occupa della gestione della merce, fra cui la navigazione della sezione Shop, la ricerca della merce secondo dei criteri, l'inserimento dei prodotti e la modifica/rimozione di quelli esistenti. |
| **Servizio** | **Descrizione** |
| getAllMerce() | Permette di visualizzare la lista di tutta la merce disponibile nel sistema |

| | |
|---|---|
| getMerceByCategoria() | Permette di visualizzare la lista della merce di una determinata categoria |
| getMerceByMarca() | Permette di visualizzare la lista della merce di una determinata marca |
| getMerceByName() | Permette di visualizzare un determinato prodotto |
| insertMerce() | Permette all'amministratore di inserire un nuovo prodotto nel sistema |
| modificaMerce() | Permette all'amministratore di modificare un prodotto esistente |
| deleteMerce() | Permette all'amministratore di cancellare un prodotto dal sistema |

## 4.3 Gestione Acquisti

| **Sottosistema** | **Descrizione** |
|---|---|
| Gestione Acquisti | Sottosistema che si occupa della gestione gli acquisti, fra cui l'inserimento di un prodotto all'interno del carrello, la rimozione di prodotti dal carrello, la visualizzazione di tutti i prodotti inseriti nel carrello con il prezzo totale e l'acquisto dei prodotti nel carrello. |
| **Servizio** | **Descrizione** |
| add() | Permette ad un cliente di inserire un oggetto all'interno del carrello. |
| remove() | Permette ad un cliente di rimuovere un oggetto dal carrello. |
| getAll() | Permette ad un cliente di visualizzare la lista degli oggetti nel carrello. |
| removeAll() | Permette ad un cliente di rimuovere tutti gli articoli dal carrello. |
| insert() | Permette ad un cliente di effettuare il pagamento ed inserire l'ordine nel sistema. |

## 4.4 Gestione Ordini

| Sottosistema | Descrizione |
|---|---|
| Gestione Ordini | Sottosistema si occupa della gestione degli ordini, fra cui la visualizzazione e ricerca degli ordini, la modifica degli ordini da parte di un operatore e l'annullamento di un ordine da parte di un utente. |
| **Servizio** | **Descrizione** |
| getAll() | Permette di visualizzare la lista di tutti gli ordini nel sistema |
| getOrdersByUser() | Permette di visualizzare la lista di tutti gli ordini di un utente nel sistema |
| getOrderByID() | Permette di visualizzare un ordine con un determinato ID di riferimento. |
| setOrder() | Permette di modificare un ordine da parte di un operatore. |
| cancel() | Permette all'utente di annullare un ordine effettuato. |

## 4.5 Gestione Ticket

| Sottosistema | Descrizione |
|---|---|
| Gestione Ticket | Sottosistema si occupa della gestione dei ticket, fra cui la creazione di un ticket di per un oggetto, la visualizzazione dei propri ticket da parte di un utente, la visualizzazione dei ticket aperti e chiusi, la modifica dello stato di un ticket da parte dell'Assistenza e lo scambio di messaggi tra Utente ed Assistenza. |
| **Servizio** | **Descrizione** |
| getAll() | Permette di visualizzare la lista di tutti i ticket nel sistema |
| getTicketByUser() | Permette di visualizzare la lista dei ticket di un utente |
| getTicketByID() | Permette di visualizzare un ticket con un determinato ID di riferimento. |
| getTicketByStatus() | Permette di visualizzare la lista dei ticket aperti o chiusi. |
| insert() | Permette all'utente di creare un nuovo ticket nel sistema. |
| setTicket() | Permette all'Assistenza di modificare lo stato del ticket. |
| getConversation() | Permette all'utente di visualizzare la conversazione di un ticket |
| getMessages() | Permette di visualizzare tutti i messaggi all'interno della conversazione |

| | |
|---|---|
| getMessagesByRank() | Permette di visualizzare tutti i messaggi di un utente o dell'Assistenza, all'interno della conversazione. |
| insertConversation() | Permette di creare una conversazione in un ticket aperto. |

# 5. Glossario

- **Warehouse:** nome del sistema che si sta sviluppando
- **Amministratore**: il termine identifica la persona che gestisce la piattaforma Warehouse
- **Operatore:** il termine identifica le persone che hanno la funzione di gestire gli ordini dei clienti.
- **Utente:** il termine identifica le persone che sono correttamente loggate al sistema
- **Utente frontend:** il termine identifica le persone che non sono loggate al sistema, che quindi, avranno un insieme di funzionalità disponibili più ristretto rispetto agli utenti loggati.
- **Cliente:** il termine identifica gli utenti che sono abilitati ad effettuare acquisti sulla piattaforma.
- **RAD:** documento di Analisi dei Requisiti
- **DBMS:** sistema di gestione di basi di dati
- **Database:** insieme organizzato di dati persistenti