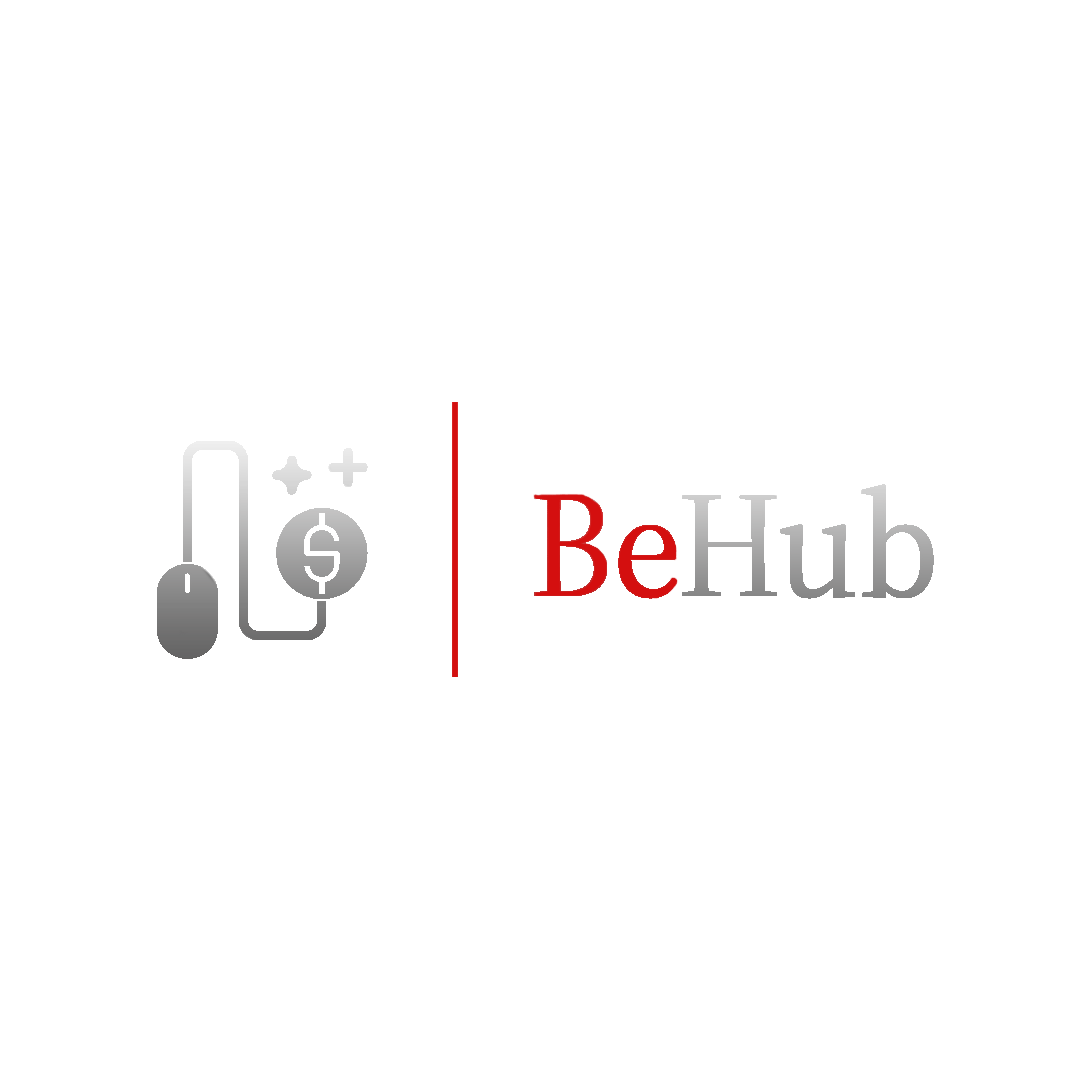


**Corso di Laurea in Informatica**

**Ingegneria del Software**

**System Document Design - BeHub**

Anno Accademico: 2022/23

**Docente: Studenti:**

Prof. Andrea De Lucia Mirko Danilo Pacelli 0512112321

Eljon Hida 0512109978

Carlo Perilli 0512112306

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autori |
| 10/12/2022 | 0.1 | Creazione documento | Tutto il team |
| 14/12/2022 | 0.2 | Decomposizione in sottosistemi e mapping hardware | Mirko Danilo Pacelli |
| 18/12/2022 | 0.3 | Gestione dati persistenti | Tutto il team |

1. **Introduzione**

***1.1 Scopo del sistema***

Il sistema che si vuole realizzare ha l’obiettivo di creare una piattaforma, o meglio “hub”, in cui gli utenti possano vendere i propri prodotti e acquistare prodotti venduti da altri utenti. Attraverso la piattaforma online il venditore potrà mettere in vendita i suoi prodotti, nuovi o usati, appartenente a diverse categorie e specificandone le caratteristiche fondamentali. Gli acquirenti, invece, potranno acquistare i prodotti direttamente dalla loro abitazione e tenere facilmente traccia dei loro ordini.

La piattaforma sarà estremamente semplice da utilizzare per utenti che hanno già utilizzato piattaforme di questo tipo e risulterà abbastanza semplice invece per utenti che non hanno mai acquistato online. Interfaccia intuitiva, minimalismo nella bara di navigazione e colori tenui all’occhio permettono all’utente di effettuare ogni operazione senza alcun problema.

Verrà infatti creata una Web App compatibile con la maggior parte dei dispositivi dotati di broser web.

***1.2 Design Goals e Trade-off***

La Web App BeHub è ideata come piattaforma con un alto livello di usabilità, in modo tale che possa essere utilizzata anche da utenti poco esperti, ed è necessario quindi implementare interfacce grafiche semplici ed intuitive. È inoltre prioritario garantire un’alta affidabilità dei dati, criptando nel database i dati sensibili degli utenti e scartando eventuali dati errati inseriti. La piattaforma infine deve avere bassi costi di manutenibilità, deve essere responsive per poter avere lo stesso impatto visivo da dispositivi avendo dimensione diverse (PC, smartphone, …), deve gestire correttamente l’accesso simultaneo di un numero discreto di utenti e deve permettere in maniera semplice di poter aggiungere nuove funzionalità per migliorarla.

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

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Descrizione e priorità | Criterio | Priorità |
| Grafica | La Web App BeHub deve avere un’interfaccia grafica semplice e di facile utilizzo in modo tale che gli utenti, indipendente dalle loro conoscenze di piattaforme di e-commerce, possano navigare nella piattaforma. Verranno usati per tale obiettivo fogli di stile CSS ed il framework BootStrap. | Usabilità | Alta |
| Icone | Il sistema deve utilizzare icone facilmente riconoscibili per permettere agli utenti di poter accedere a determinate funzioni in maniera intuitiva. Oltre ad icone “standard” verranno utilizzate icone più coinvolgenti. | Usabilità | Alta |
| Responsività | Le interfacce grafiche del sistema devono essere responsive per poter permettere la coerenza grafica delle pagine visualizzate da diversi dispositivi quali PC, Smartphone e Tablet. | Usabilità | Alta |
| Notifica Errori | Il sistema deve notificare all’utente eventuali errori nella compilazione di form con un testo, fornendo suggerimenti sulle modalità di compilazione corrette. | Usabilità | Alta |
| Gestione Dati Errati | Il sistema deve riconoscere e scartare eventuali errori nei dati non memorizzandoli nel database, rispondendo in maniera adeguata. Le form eviteranno la maggior parte di questi problemi con dei controlli sul formato dei dati. | Affidabilità | Alta |
| Sicurezza dei Dati | I dati sensibili come la password e il CVV devono essere criptati all’interno del database in SHA-256 e manipolati attraverso il codice. | Affidabilità | Alta |
| Inserimento Prodotti | Il sistema deve permettere l’inserimento di nuovi prodotti da parte del venditore senza modificare il sistema corrente. | Estensibilità | Ata |
| Nomi Variabili | Le variabili all’interno del codice devono avere nomi adatte a descriverle per permettere una maggiore facilità nella manutenzione del codice. | Manutenibilità | Alta |
| Commenti | Il codice deve avere commenti adeguati a descriverne il funzionamento per permetterlo rendere più leggibile e, di conseguenza, più semplice da manutenere. | Manutenibilità | Media |
| Tempo di Risposta | Il sistema deve essere reattivo per tutte le operazioni mantenendosi, nel 90% dei casi, sotto i 3 secondi di tempo di risposta. Verranno perciò utilizzate strutture dati efficienti e ridotti al minimo i cicli nel codice. | Prestazioni | Bassa |

I criteri di usabilità ed affidabilità sono prioritari nello sviluppo della piattaforma. I primi non comportano trade-off con gli altri obiettivi, mentre i criteri di affidabilità hanno priorità rispetto al tempo di risposta, che in caso di controlli su molti input potrebbe necessitare di un tempo di risposta di poco superiore ai 3 secondi, anche se il criterio di prestazione sarà comunque rispettato nella maggioranza dei casi.

***1.3 Definizioni, acronimi e abbreviazioni***

* **RAD**: Documento di Analisi dei Requisiti;
* **e-commerce**: Negozio online;
* **Design Goals**: Obiettivi di design progettati per il sistema proposto;
* **Trade-off**: Scelte e compromessi tra design goals;
* **BeHub**: Nome del sistema proposto;
* **Database**: Architettura esterna che si occupa della memorizzazione e gestione dei dati persistenti;
* **CSS**: Fogli di stile;
* **BootStrap**: Librerie per la parte grafica;
* **SHA-256**: Sistema per criptare i dati;
* **CVV**: Codice di tre cifre del metodo di pagamento.
* **SDD**: System Design Document.

***1.4 Riferimenti***

* RAD BeHub;
* Problem Statement BeHub;
* Libro Object-Oriented Software Engineering (Using UML, Patterns, and Java) Third Edition Autori: -- Bernd Bruegge & Allen H. Dutoit

***1.5 Panoramica del documento***

Il presente SDD è formato da quattro sezioni:

**Introduzione**: Viene descritto lo scopo, i design goals ed i trade-off proposti per il sistema.

**Architettura software proposta**: Viene descritto come il sistema verrà partizionato in sottosistemi, il mapping Hardware/Software, la gestione dei dati persistenti, controllo degli accessi e sicurezza, controllo flusso globale del sistema e boundary conditions.

**Servizi dei sottosistemi**: Vengono descritti i servizi che ogni sottosistema offre.

**Glossario**: Illustra il significato dei termini utilizzati nel documento.

1. **Architettura del Sistema Proposto**

***2.1 Panoramica***

La piattaforma BeHub è un e-commerce che interagisce con gli utenti tramite un’interfaccia web e gestiste i dati persistenti attraverso un database relazionale.

Il sistema proposto utilizza un’architettura Client/Server che permette di separare la logica di business dalla logica di presentazione, in modo tale da permettere di lavorare su componenti diverse simultaneamente e migliorando la manutenzione e la leggibilità.

Nello sviluppo del sistema utilizzeremo un’architettura di tipo Model-View-Controller [MVC]. L’architettura MVC si basa su tre componenti principali:

* Model [M] = specifica le procedure di estrazione da DB ed elaborazione dei dati.
* View [V] = specifica la modalità e la forma di presentazione dei dati all’utente.
* Controller [C] = specifica il flusso dell’applicazione e controlla l’interazione tra gli altri livelli.

L’utilizzo del modello MVC comporta numerosi vantaggi, quali tra i più importanti, la possibilità di suddividere il lavoro nel caso ci debbano lavorare più persone (o gruppi di persone), contemporaneamente, con competenze diverse, agevolazione a un eventuale lavoro di manutenzione dato l’utilizzo di un modello e di regole standard che ne accentuano la comprensione. Per natura dell’architettura MVC si ha basso accoppiamento tra model, view e controller poiché una modifica ad un sottosistema non influenza nessun altro sottosistema. Inoltre, l’architettura MVC favorisce la cosiddetta **alta coesione** poiché consente il raggruppamento logico di azioni correlate su una stessa componente.

Per la logica di presentazione verranno utilizzati:

* **HTML5;**
* **CSS;**
* **BootStrap;**

Per la logica di business verranno utilizzati:

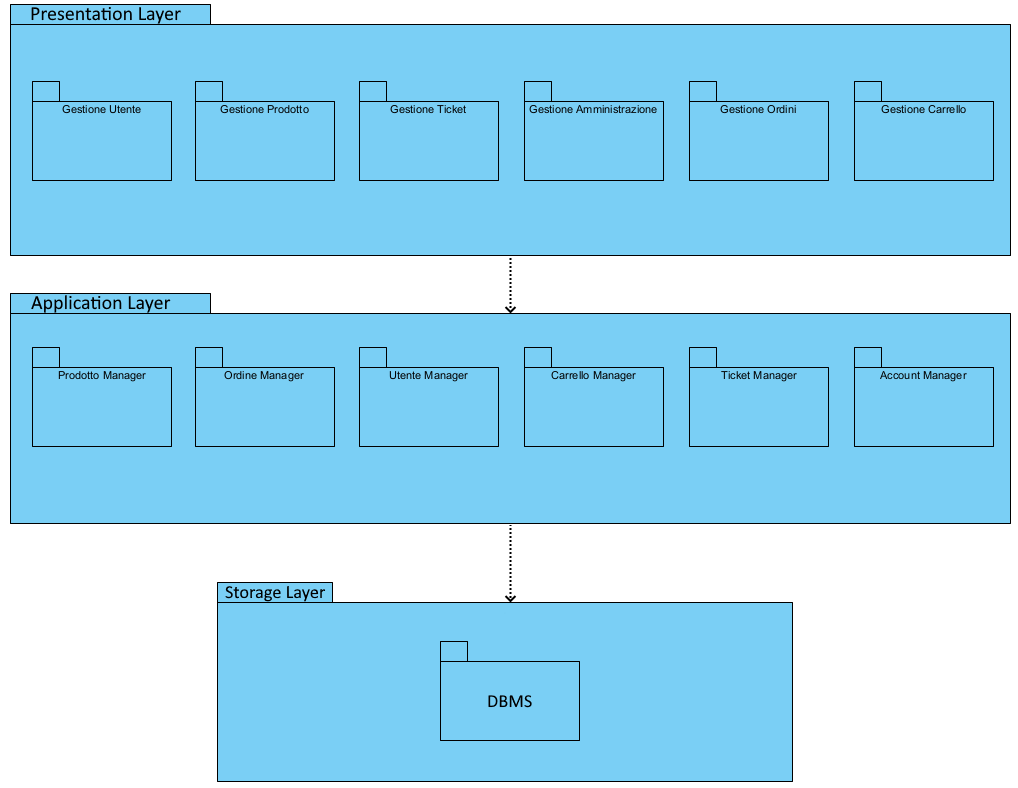
* **Java 17.X**
* **JavaScript**

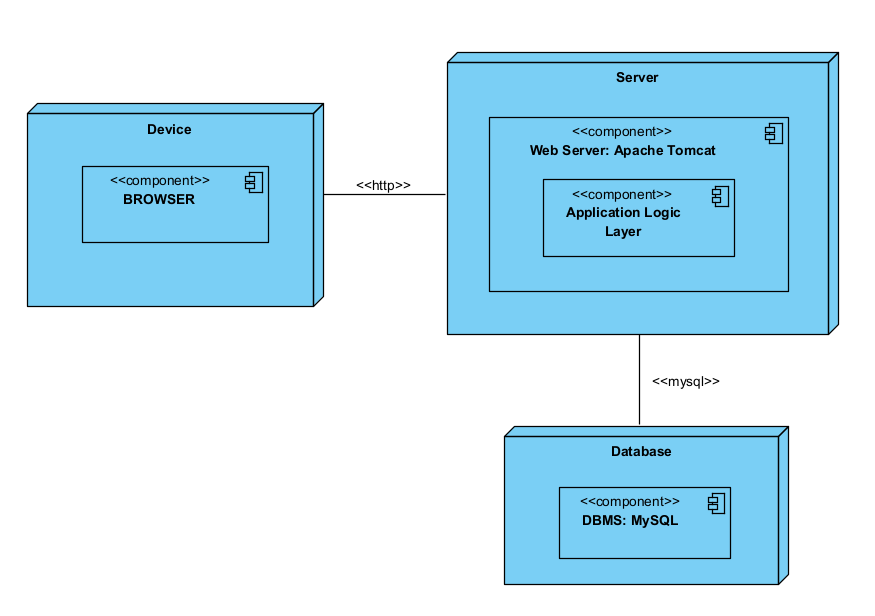
Per la gestione dei dati persistenti verrà invece utilizzato il **DBMS MySQL**

***2.2 Decomposizione del sistema***

Il sistema viene suddiviso nei seguenti sottosistemi:

* **Gestione Utente:** si occupa di gestire le funzioni di login, logout, modifica e inserimento delle informazioni dell’account;
* **Gestione Prodotto:** è responsabile dell’inserimento, modifica ed eliminazione delle informazioni del prodotto, ed include l’eliminazione di un prodotto da parte dell’addetto al catalogo;
* **Gestione Ticket:** si occupa dell’inserimento di ticket e della loro visualizzazione;
* **Gestione Carrello:** si occupa della visualizzazione del carrello con inserimento ed eliminazione di un prodotto, e modifica della quantità dei singoli prodotti;
* **Gestione Ordine:** è responsabile della visualizzazione degli ordini e dell’inserimento di un ordine nel database;
* **Gestione Amministrazione:** si occupa delle funzioni dedicate agli addetti al supporto e catalogo, come la visualizzazione degli ordini di un utente e l’inserimento di una categoria;
* **Persistenza:** si occupa della gestione dei dati persistenti tramite il DBMS MySQL.

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

***2.3 Mapping Hardware***

**Device:** l’utente utilizza il sistema tramite un browser qualsiasi nel proprio device;

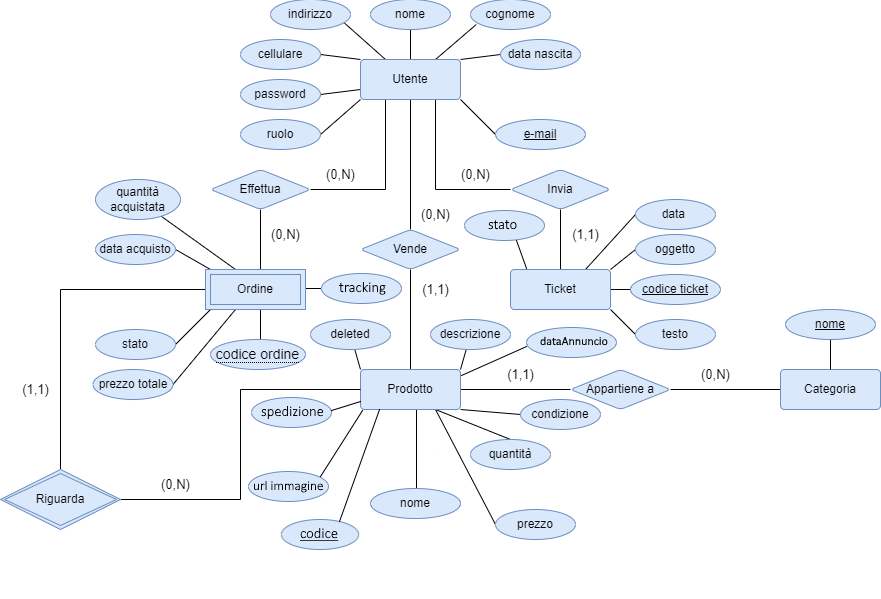
**Web Server:** il server utilizzato è Apache Tomcat versione 10.X;

**Application Logic Layer:** la parte back-end è sviluppata tramite utilizzo di JavaServlet, mentre la parte front-end è sviluppata mediante l’uso di HTML5, JSP, CSS e framework BootStrap;

**Database:** il sistema utilizza il DBMS MySQL per la gestione dei dati persistenti.

***2.4 Gestione Dati Persistenti***

Lo schema EER (Enhanced Entity Relationship) modellato per il sistema è il seguente:



Per il sistema proposto è stato deciso l’utilizzo di un database relazione quale MySQL che permette in maniera efficiente di poter gestire tutti i dati persistenti richiesti. Allo stesso tempo non si ha la necessità di sfruttare l’astrazione fornita da un database Object-oriented e si preferiscono le maggiori performance offerte da un database relazionale.

Di seguito è riportata in maniera dettagliata la lista delle entità persistenti con i loro attributi, dopo la fase di mapping logico:

***Utente***

|  |  |  |  |
| --- | --- | --- | --- |
| TABLE | TIPO | VINCOLI | KEY |
| email | Varchar(50) | NOT NULL | PRIMARY KEY |
| passwordUser | Varchar(50) | NOT NULL |  |
| nome | Varchar(50) | NOT NULL |  |
| cognome | Varchar(50) | NOT NULL |  |
| indirizzo | Varchar(50) | NOT NULL |  |
| telefono | Varchar(50) | NOT NULL |  |
| numeroCarta | Char(50) |  |  |
| dataNascita | Date | NOT NULL |  |
| intestatario | Varchar(50) |  |  |
| Ruolo | ENUM('AC','AS','RU') | NOT NULL, DEFAULT ‘RU’ |  |

***Categoria***

|  |  |  |  |
| --- | --- | --- | --- |
| TABLE | TIPO | VINCOLI | KEY |
| Nome | ENUM('Abbigliamento','Calzature','Elettronica','Libri','Giocattoli') | NOT NULL | PRIMARY KEY |

***Prodotto***

|  |  |  |  |
| --- | --- | --- | --- |
| TABLE | TIPO | VINCOLI | KEY |
| Codice | int | NOT NULL, AUTO\_INCREMENT | PRIMARY KEY |
| Nome | Varchar(50) | NOT NULL |  |
| Descrizione | Text | NOT NULL |  |
| Deleted | Bool | NOT NULL |  |
| Prezzo | Double | NOT NULL |  |
| speseSpedizione | Double | NOT NULL |  |
| emailVenditore | Varchar | NOT NULL | FOREIGN KEY REFERENCES Utente(email) |
| urlImmagine | Varchar | NOT NULL |  |
| nomeCategoria | ENUM('Abbigliamento','Calzature','Elettronica','Libri','Giocattoli') | NOT NULL | FOREIGN KEY REFERENCES Categoria(nome) |
| Condizione | ENUM(“Nuovo” , “Usato”) | NOT NULL |  |
| Quantity | Int | NOT NULL |  |
| dataAnnuncio | Date | NOT NULL |  |

***Ordine***

|  |  |  |  |
| --- | --- | --- | --- |
| TABLE | TIPO | VINCOLI | KEY |
| codiceOrdine | Int | NOT NULL, AUTO\_INCREMENT | PRIMARY KEY |
| codiceProdotto | Int | NOT NULL | PRIMARY KEY, FOREIGN KEY REFERENCES Prodotto(codice) |
| emailCliente | Varchar(50) | NOT NULL | FOREIGN KEY REFERENCES Utente(email |
| prezzoTotale | Double(10,2) | NOT NULL |  |
| Quantity | Int | NOT NULL |  |
| dataAcquisto | Date | NOT NULL |  |
| tracking | Varchar | NOT NULL |  |
| stato | ENUM('In lavorazione', 'Spedito', 'Consegnato') | NOT NULL, default 'In lavorazione' |  |

***Ticket***

|  |  |  |  |
| --- | --- | --- | --- |
| TABLE | TIPO | VINCOLI | KEY |
| codice | int | NOT NULL | PRIMARY KEY |
| Testo | Text | NOT NULL |  |
| Oggetto | Varchar(70) | NOT NULL |  |
| dataInvio | Date | NOT NULL |  |
| EmailUtente | Varchar(50) | NOT NULL |  |
| Stato | ENUM('Aperto', 'Chiuso') | NOT NULL |  |

***2.5 Controllo Accessi e Sicurezza***

L’accesso alla piattaforma per tutte le tipologie di utenti comprese le figure amministrative è effettuato tramite l’inserimento delle apposite credenziali (EMAIL,PASSWORD) che verranno richieste allo scadere della sessione in cui si opera. BeHub è un sistema con differenti tipi di utenza, essi possono accedere a diverse funzionalità a seconda del tipo di utenza che assume in quel momento.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Objects  Actors | Account | Prodotto | Carrello | Ordine | Ticket |
| Cliente | modificaProfilo eliminaAccount log-out | viewProducts addToCart  justBuy | check-out removeFrom increaseQuantity decreaseQuantity modifyQuantity | requestOrder | writeTicket  openTicket checkStatus |
| Utente non registrato | sign-up  log-in | viewProducts |  |  |  |
| Addetto al supporto | Log-out |  |  | viewOrders | manageTickets  writeTicket |
| Addetto al catalogo | Log-out | addProduct removeProduct |  |  |  |

***2.6***

***2.7 Controllo del Software Globale***

All’interno di BeHub il controllo del software globale viene effettuato dal Web Server, che si occupa di smistare le varie richieste alle Java Servlet appropriate. Le Java Servlet gestiscono la richiesta generando una risposta che viene successivamente inclusa in una pagina JSP per essere visualizzata dall’utente come pagina html.

***2.8 Boundary Conditions***

***2.8.1 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 “addetto al supporto” e almeno un account “addetto al catalogo” fornendo una mail e una password in quanto tale account non può essere creato dall’interfaccia del sistema. Tale account sarà poi fornito agli utenti responsabili della piattaforma. L’amministratore inoltre dovrà avviare la macchina necessaria all’esecuzione del web-server.

***2.8.2 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 ad uno 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.

***2.8.3 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.

***3 Servizi dei sottosistemi***

|  |  |
| --- | --- |
| Gestione Account | |
| Registrazione | Consente ad un utente che sta visitando la Web App di registrarsi al sistema. |
| Login | Consente ad un utente registrato di effettuare l’accesso al sistema. |
| Logout | Consente ad un utente che ha effettuato l’accesso di uscire dal sistema. |
| Visualizza Profilo | Consente ad un utente registrato di visualizzare le proprie informazioni personali. |

|  |  |
| --- | --- |
| Gestione Assistenza | |
| Apertura Ticket | Consente ad un cliente di scrivere e inviare un ticket verso l’addetto al supporto. |
| Lista Ticket | Consente ad un utente o all’addetto al supporto di visualizzare la lista dei ticket. |
| Lettura Ticket | Consente ad un utente o all’addetto al supporto di visualizzare in dettaglio il ticket. |

|  |  |
| --- | --- |
| Gestione Utente | |
| Visualizza Carrello | Consente ad un utente di visualizzare il proprio carrello. |
| Aggiungi Prodotto al Carrello | Consente ad un utente di aggiungere i prodotti al carrello. |
| Rimuovi Prodotto dal Carrello | Consente ad un utente di rimuovere i prodotti dal carrello. |
| Modifica quantità Prodotto | Consente ad un utente di modificare la quantità di un prodotto che si vuole acquistare. |
| Visualizza Dettagli Prodotto | Consente ad un utente di visualizzare le informazioni complete di un prodotto. |

|  |  |
| --- | --- |
| Gestione Cliente | |
| Visualizza Checkout | Consente ad un utente di visualizzare il ckeck-out. |
| Acquisto Prodotto | Consente ad un utente di acquistare un prodotto. |

|  |  |
| --- | --- |
| Gestione Prodotto | |
| Aggiungi Prodotto | Consente ad un utente di caricare un prodotto. |
| Rimuovi Prodotto | Consente ad un utente o all’addetto al catalogo di eliminare un prodotto. |
| Modifica Prodotto | Consente ad un utente di apportare modifiche ad un prodotto già caricato. |

|  |  |
| --- | --- |
| Controllo Ordini | |
| Visualizza Ordini ricevuti | Consente ad un utente di visualizzare gli ordini da spedire. |
| Visualizza Storico Ordini | Consente ad un utente di visualizzare la lista degli ordini spediti. |
| Aggiorna Tracking | Consente ad un utente di aggiornare il numero di Tracking di un ordine. |