UNIVERSITÀ DEGLI STUDI DI SALERNO

Corso di Ingegneria del Software

A.A. 2023/2024

SecondLifeTech

SYSTEM DESIGN DOCUMENT

VERSIONE 1.1

09/03/2024

DOCENTE:

PROF. ANDREA DE LUCIA

PRESENTATO DA:

CAPRICANO DAVIDE, 05121 10422

GIARDINETTO GIUSEPPE, 05121 14655

SPERA ALFREDO JESHOUA, 05121 12240 (PROJECT MANAGER)

REVISION HISTORY

Data	Versione	Descrizione
09/02/2024	0.1	Prima stesura
09/02/2024	0.2	Aggiunta grafici
09/02/2024	0.3	Aggiunta matrici di accesso
10/02/2024	0.4	Effettuate modifiche
18/02/2024	1.0	Aggiunte entità
09/03/2024	1.1	Correzioni

Sommario

1. Introduzione

1.1. Finalità del Sistema

Il progetto "SecondLifeTech" ha lo scopo di sviluppare un sito di e-commerce specializzato nella vendita di smartphone e tablet ricondizionati.

Il sito offrirà una piattaforma completa per l'acquisto e la stima monetaria di dispositivi elettronici ricondizionati, promuovendo allo stesso tempo la sostenibilità ambientale attraverso il riutilizzo di dispositivi usati.

Il sistema prodotto faciliterà l'interfacciamento dell'azienda con i propri clienti e possibili tali, questo tramite un'interfaccia intuitiva e facilitata.

1.2. Obbiettivi della Progettazione

1.2.1. Criteri di Usabilità

- *Usabilità*: Il Sistema ha come obbiettivo quello di essere utilizzabile da più clienti possibili, è quindi necessario garantire un'interfaccia semplice e di facile comprensione con una navigazione intuitiva e non complessa

1.2.2. Criteri di Affidabilità

- *Robustezza*: Il Sistema dev'essere in grado di garantire correttamente il servizio in risposta ad eventuali input errati evitando la propagazione degli errori con un'efficiente gestione di questi ultimi
- *Disponibilità*: Il Sistema deve garantire che i dati persistenti non vengano intaccati da eventuali errori
- *Sicurezza*: Il Sistema deve garantire che venga rispettata la privacy di ogni utente attraverso best practice in crittografia e autenticazione

1.2.3. Criteri di Prestazioni

- *Tempo di Risposta*: Il Sistema sarà in grado di fornire il catalogo dei dispositivi all'utente in un tempo inferiore a cinque secondi, garantendo un'esperienza utente rapida e reattiva
- *Spazio Richiesto*: La dimensione complessiva del Sistema non supererà i 10 GB, ottimizzando l'utilizzo delle risorse
- *Concorrenza Dati*: Il Sistema sarà in grado di supportare un elevato numero di utenti contemporanei senza cali di prestazioni

1.2.4. Criteri di Supportabilità

- *Modificabilità*: Il sistema sarà progettato per essere facilmente modificabile e aggiornabile, facilitando l'implementazione di nuove funzionalità e la correzione di eventuali bug
- *Manutenibilità*: L'architettura del sistema sarà modulare e flessibile, consentendo l'inserimento di nuovi tipi di prodotti senza interventi complessi sul codice sorgente

1.2.5. Criteri Legali

- *Privacy*: Il Sistema deve essere conforme alle normative europee sulla privacy online

1.3. Trade-Off Obbiettivi

1.3.1. Tempo di Risposta vs Spazio Richiesto

Data la natura competitiva del mercato del dominio di applicazione, risulta molto più importante riuscire a garantire un servizio con un tempo di risposta che renda la navigazione del sito piacevole. In questo modo l'azienda sarà in grado di competere con altre dello stesso settore.

1.4. Riferimenti

Questo documento si basa sul **Requirement Analysis Document**.

2. Architettura Corrente del Sistema

Nel caso del sistema proposto non è presente alcuna architettura corrente.

3. Architettura Proposta per il Sistema

L'architettura scelta per il Sistema proposto è la **Three-Tier**.

3.1. Panoramica dell'Architettura

Il sistema verrà suddiviso in tre livelli logici, questi saranno:

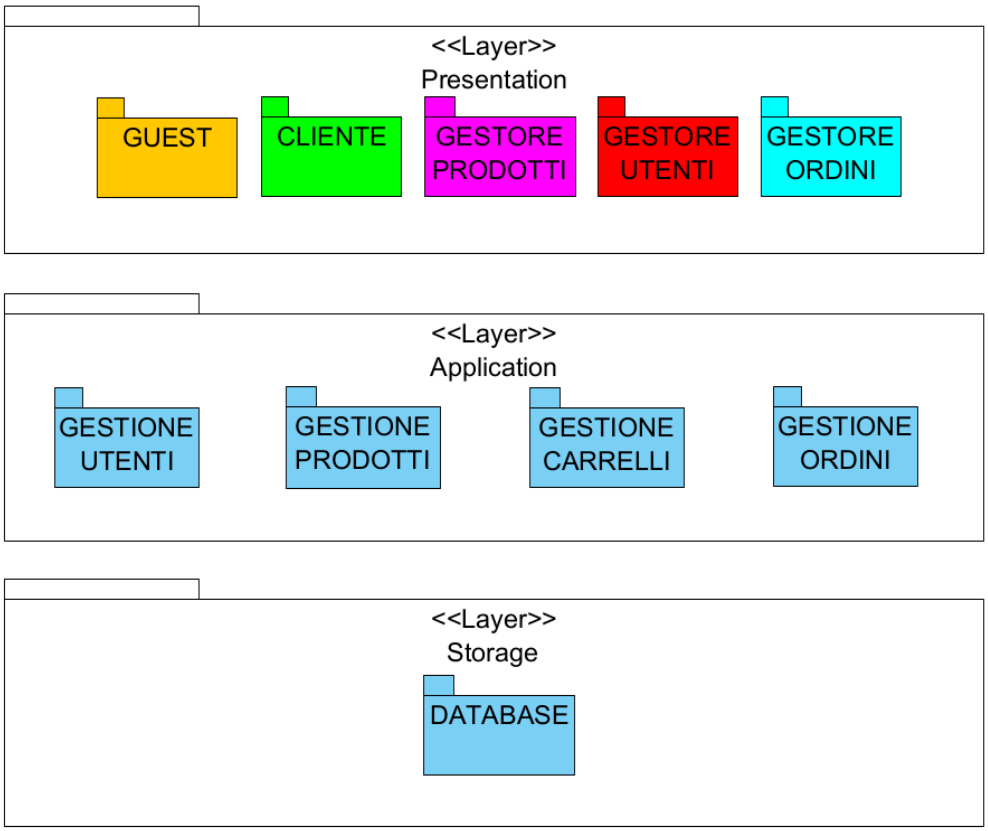
- *Presentazione*: per l'interfacciamento con l'utente tramite grafica
- *Application*: per la logica di business del sistema
- *Data*: per la persistenza dei dati

Il sistema, inoltre, garantirà diversi livelli di *permessi*, suddivisi in base alla categoria di utente che accede al sistema, in questo modo gli utenti *Guest* avranno accesso a ridotte funzionalità rispetto ad un *Cliente*.

Un *Cliente*, poi, avrà delle funzionalità diverse da quelle di un *Gestore Prodotti* che avrà compiti e responsabilità diverse da quelle dei suoi colleghi *Gestore Ordini* e *Gestore Utenti*.

3.2. Decomposizione in Sottosistemi

3.2.1. Sottosistemi e i loro Servizi



3.2.1.1. Gestione Utenti

Questo sottosistema è responsabile delle funzionalità per gestire tutti gli utenti, come:

- Login
- Logout
- Contattare l'azienda
- Operazioni CRUD inerenti agli utenti

3.2.1.2. Gestione Prodotti

Questo sottosistema è responsabile delle funzionalità per gestire tutti i prodotti, come:

- Valutazione usato
- Operazioni CRUD inerenti ai prodotti

3.2.1.3. Gestione Carrelli

Questo sottosistema è responsabile delle funzionalità per gestire tutti i carrelli, come:

- Operazioni per la gestione dei cookie per i carrelli dei guest (senza interagire con il database)
- Operazioni CRUD inerenti ai carrelli

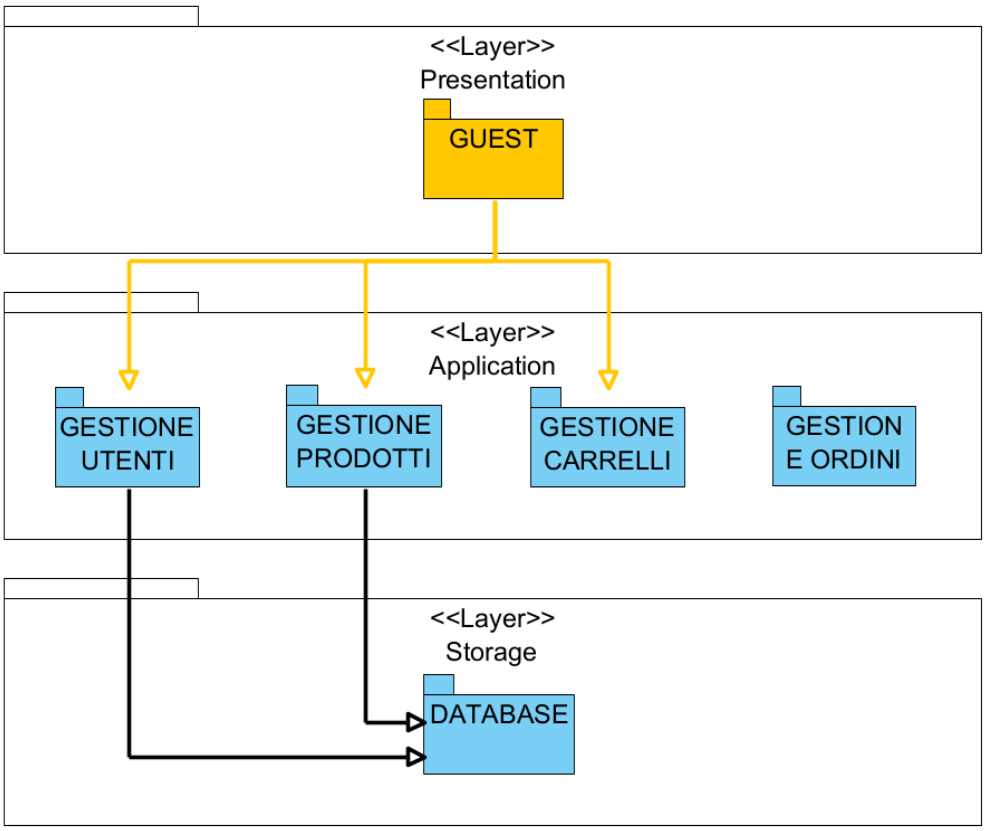
3.2.1.4. Gestione Ordini

Questo sottosistema è responsabile delle funzionalità per gestire tutti i carrelli, come:

- Finalizzazione degli ordini
- Operazioni CRUD inerenti agli ordini

3.2.2. Interazioni degli Utenti con i Sottosistemi

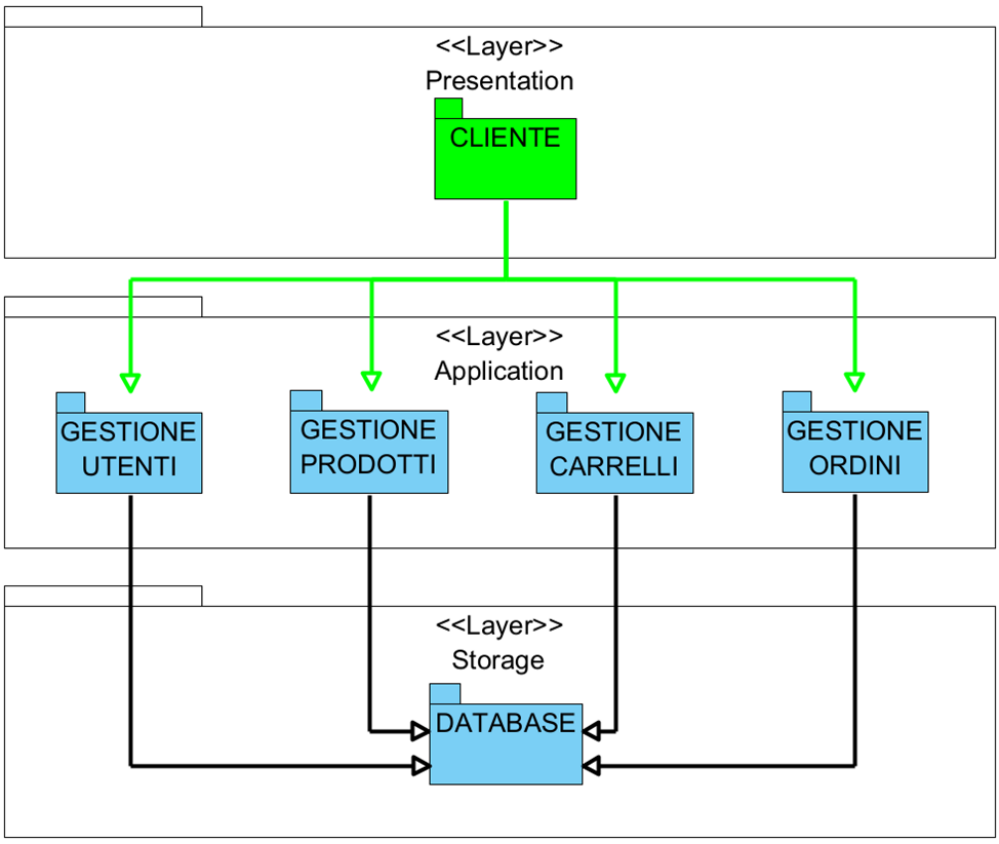
3.2.2.1. Guest



Il *Guest* interagisce con i seguenti sottosistemi:

- **Gestione Utenti**: per fare il login e contattare l'azienda
- **Gestione Prodotti**: per visualizzare i prodotti in vendita e per valutare l'usato
- **Gestione Carrelli**: per modificare il proprio carrello temporaneo attraverso la gestione dei cookie (non interagisce con il database)

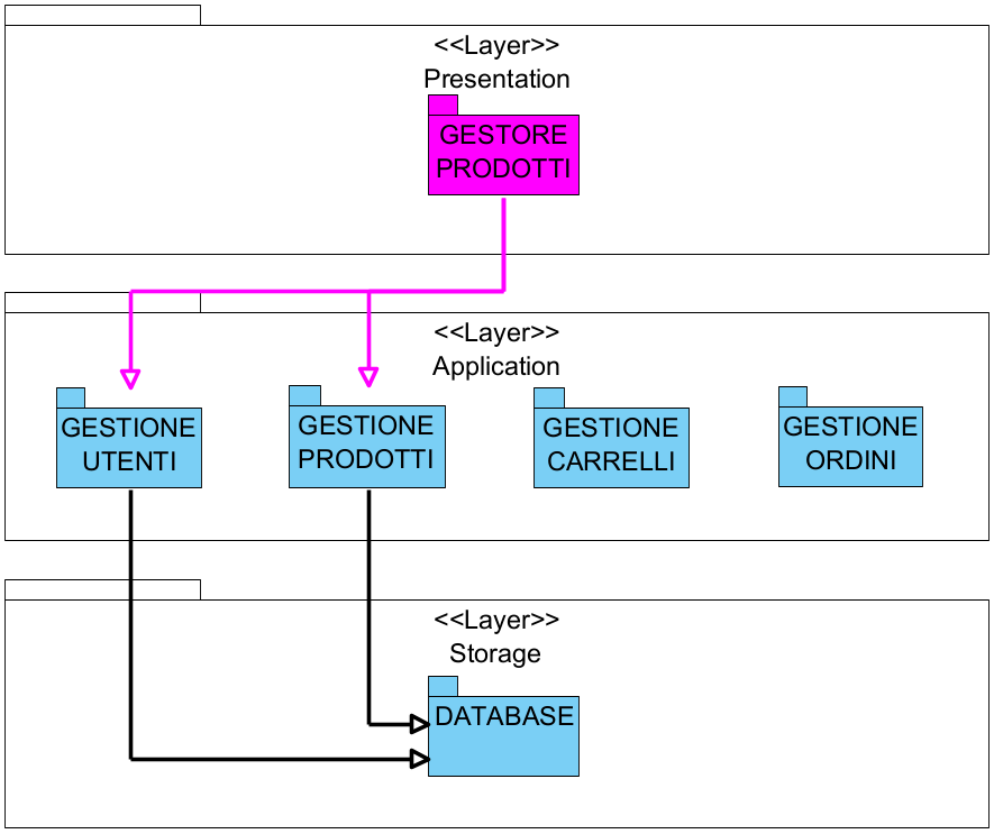
3.2.2.2. Cliente



Il *Cliente* interagisce con i seguenti sottosistemi:

- **Gestione Utenti**: per fare il logout, contattare l'azienda e modificare il proprio profilo
- **Gestione Prodotti**: per visualizzare i prodotti in vendita e valutare il proprio usato
- **Gestione Carrelli**: per modificare ed aggiungere prodotti al proprio carrello
- **Gestione Ordini**: per poter effettuare un ordine e visualizzare gli ordini effettuati

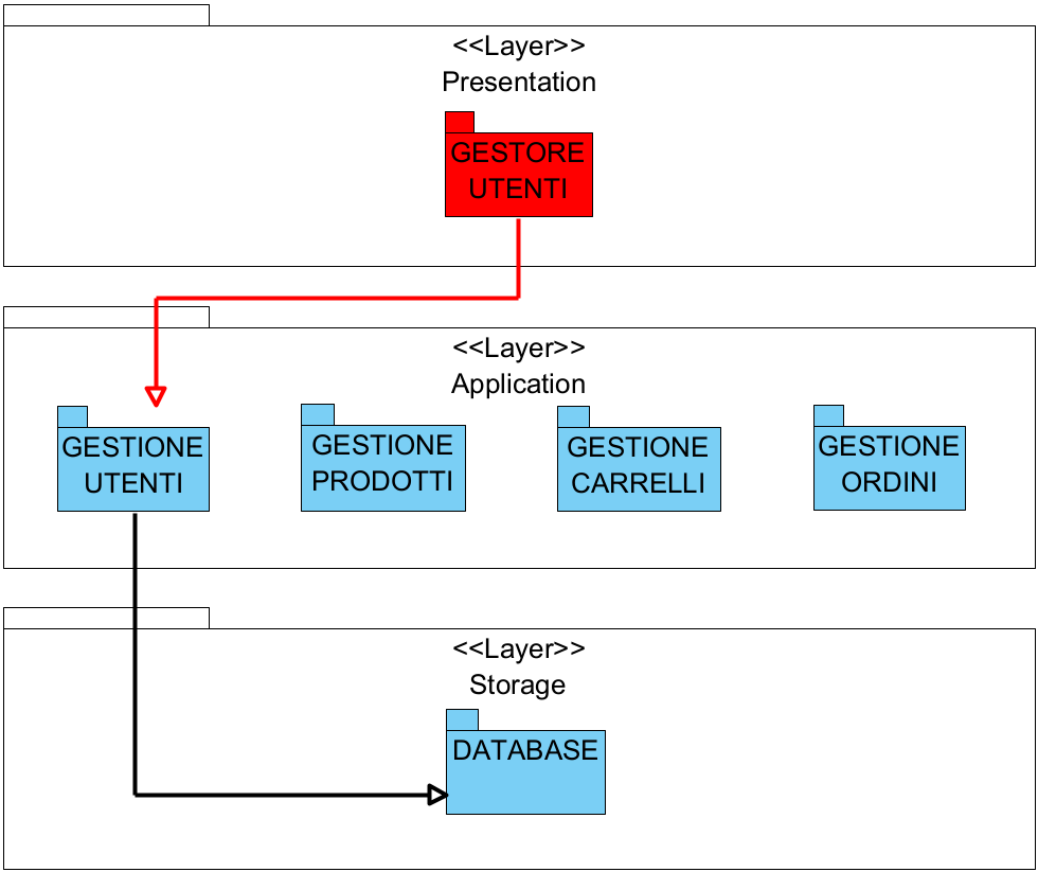
3.2.2.3. Gestore Prodotti



Il *Gestore Prodotti* interagisce con i seguenti sottosistemi:

- **Gestione Utenti**: per fare il logout
- **Gestione Prodotti**: per visualizzare, aggiungere, modificare ed eliminare i prodotti dal catalogo

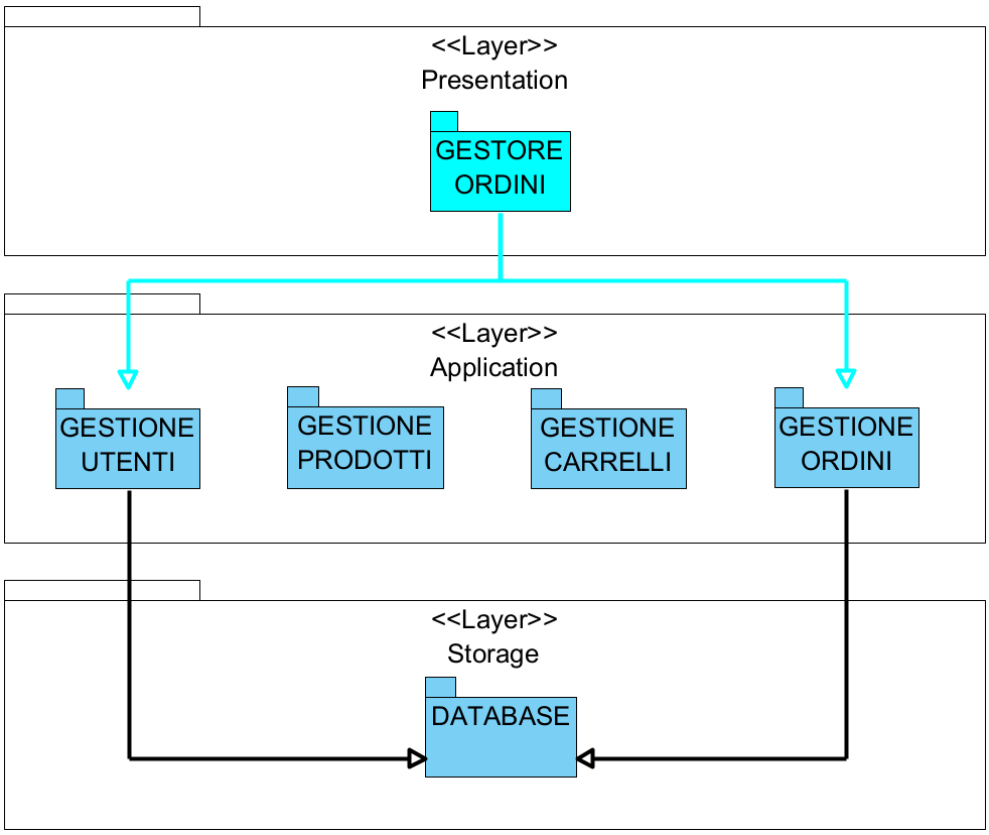
3.2.2.4. Gestore Utenti



Il *Gestore Utenti* interagisce con i seguenti sottosistemi:

- **Gestione Utenti**: per fare il logout, visualizzare ed eliminare gli utenti registrati e aggiungere nuovi utenti

3.2.2.5. Gestore Ordini



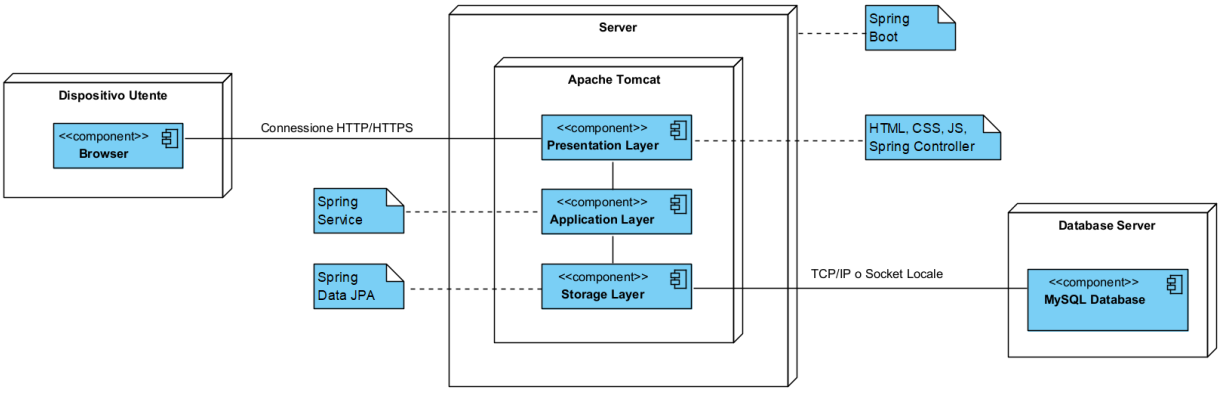
Il *Gestore Utenti* interagisce con i seguenti sottosistemi:

- **Gestione Utenti**: per fare il logout
- **Gestione Ordini**: per visualizzare e cambiare stato agli ordini effettuati nel sistema

3.3. Mapping Hardware/Software

Il sistema utilizzerà Spring con Spring Boot sul Server.

Come mostrato nel seguente grafico, il Dispositivo Utente farà una richiesta al Server, mentre il DB Server conterrà i dati persistenti.



3.4. Gestione dei Dati Persistenti

Per la persistenza dei dati, la scelta è stata orientata verso un DBMS relazionale, in quanto, oltre al vantaggio della sua semplicità di implementazione e diffusione nei sistemi moderni e storici, un DBMS permette la persistenza e l'accesso ai dati concorrente con estrema trasparenza allo sviluppatore e al cliente. Pertanto, è stato scelto MySQL per la persistenza dei dati.

I dati persistenti nel sistema sono stati individuati nel RAD (4.3.1. Dizionario delle Entità). Sono state effettuate le seguenti modifiche:

- Aggiunte le entità "IndirizzoSpedizione" e "MetodoPagamento": queste saranno relazionate agli utenti registrati permettendo ad un utente di possedere più indirizzi di spedizione e più metodi di pagamento.
- Divisa l'entità "Prodotto" in "ProdottoModello" e "ProdottoVariazione": questa divisione si è resa necessaria poiché le informazioni relative a un modello venivano ripetute su ogni riga di variazione. In questo modo viene risparmiato spazio sul database e vengono velocizzate le query.
- Aggiunta l'entità "ImmagineProdotto": serve per memorizzare un'immagine corrispondente a più variazioni di modello.
- Aggiunta l'entità "ProdottoInCarrello": serve per memorizzare un prodotto all'interno di un certo carrello con la quantità inserita.
- Aggiunta l'entità "ProdottoInOrdine": serve per memorizzare un prodotto all'interno di un certo ordine per preservarlo in caso di eliminazione di un prodotto o di un utente.

Nome Entità	Attributi	Descrizione
UtenteRegistrato	• Nome • Cognome • E-Mail • Password • Ruolo • Indirizzi di Spedizione • Metodi di Pagamento • Numero Telefono	Un'entità che rappresenta un *Utente Registrato* all'interno del sistema Può essere specializzato in *Gestore* o *Cliente*
Carrello	• Prodotti nel Carrello • Totale • Utente	Un'entità che rappresenta il carrello di un *UtenteRegistrato*
ProdottoInCarrello	• Carrello • Prodotto • Quantità • Subtotale	Un'entità che rappresenta un Prodotto inserito in un carrello Contiene informazioni inerenti alla quantità inserita nel carrello
ProdottoModello	• Nome • Brand • Immagine • Categoria	Un'entità che rappresenta un modello di un prodotto

ProdottoVariazione	• RAM • Display • Spazio Interno • Prezzo • Quantità • Colore • Anno • Condizione	Un'entità che rappresenta la variazione di un prodotto. Ogni variazione è collegata ad un modello
Ordine	• E-mail • Prodotti nell'Ordine • Data • Totale • Indirizzo di spedizione • Stato spedizione	Un'entità che rappresenta un ordine nel sistema
ProdottoInOrdine	• Variazione Prodotto (se esiste) • Informazioni prodotto (se la VP non esiste) • Sub-totale • Quantità	Un'entità che permette di conservare i prodotti degli ordini, anche se l'utente e/o il prodotto vengono eliminati
IndirizzoSpedizione	• Via • Città • Nazione • Codice Postale	Un'entità che rappresenta un indirizzo di spedizione
MetodoPagamento	• Numero carta • Nome proprietario • Data scadenza • CVV	Un'entità che rappresenta un metodo di pagamento
ImmagineProdotto	• Nome • Tipo • Dati	Un'entità che rappresenta un'immagine

3.5. Controllo degli Accessi e Sicurezza

3.5.1. Gestione Utenti

	Login	Registrazione	Logout	CreaUtente	RimuoviUtente	VisualizzaUtenti	ModificaProfilo	VisualizzaProfilo	EliminaProfilo	Contatta
Guest	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓
Cliente	✗	✗	✓	✗	✗	✗	✓	✓	✓	✓
Gestore Prodotti	✗	✗	✓	✗	✗	✗	✓	✓	✗	✗
Gestore Utenti	✗	✗	✓	✓	✓	✓	✓	✓	✗	✗
Gestore Ordini	✗	✗	✓	✗	✗	✗	✓	✓	✗	✗

3.5.2. Gestione Prodotti

	VisualizzaCatalogo	RicercaProdotto	VisualizzaDettagliProdotto	StimaUsato	EliminaProdotto	AggiungiProdotto	ModificaProdotto	ModificaBanner
Guest	✓	✓	✓	✓	✗	✗	✗	✗
Cliente	✓	✓	✓	✓	✗	✗	✗	✗
Gestore Prodotti	✓	✓	✓	✗	✓	✓	✓	✓
Gestore Utenti	✗	✗	✗	✗	✗	✗	✗	✗
Gestore Ordini	✗	✗	✗	✗	✗	✗	✗	✗

3.5.3. Gestione Carrelli

	VisualizzaCarrello	AggiungiAlCarrello	EliminaDalCarrello	ModificaQuantitàInCarrello
Guest	✓	✓	✓	✓
Cliente	✓	✓	✓	✓
Gestore Prodotti	✗	✗	✗	✗
Gestore Utenti	✗	✗	✗	✗
Gestore Ordini	✗	✗	✗	✗

3.5.4. Gestione Ordini

	FinalizzaOrdine	VisualizzaOrdiniEffettuati	VisualizzaTuttiGliOrdini	ModificaStatoOrdine
Guest	✗	✗	✗	✗
Cliente	✓	✓	✗	✗
Gestore Prodotti	✗	✗	✗	✗
Gestore Utenti	✗	✗	✗	✗
Gestore Ordini	✗	✗	✓	✓

3.6. Controllo del Flusso del Sistema

Il flusso è di tipo Event-Driven. Infatti, ogni azione dell'utente è associata ad un evento a cui il sistema reagisce in risposta:

- Gli eventi saranno attivati attraverso pulsanti o altri componenti presenti sulle pagine web
- Questi eventi saranno gestiti dai corrispondenti event handler, che inoltreranno la richiesta ai controller
- I controller saranno responsabili di chiamare i servizi dall'Application Layer, per poi ricevere e gestire la loro risposta

3.7. Condizioni al Boundary

3.7.1. Avvio del Server

- Controllo dei prerequisiti:
  - Verifica della versione di Java installata
  - Verifica della presenza di memoria RAM sufficiente
  - Verifica dello spazio libero su disco
- Download delle dipendenze:
  - Maven viene automaticamente utilizzato per scaricare le dipendenze del progetto dal Maven Central Repository prima della compilazione
- Compilazione del progetto:
  - Il codice sorgente Java viene compilato in bytecode utilizzando il compilatore `javac`
- Esecuzione del server:
  - Il comando `mvn spring-boot:run` viene utilizzato per avviare il server Spring Boot
  - Esegue il download delle dipendenze e compila il progetto
  - Il server è in ascolto sulla porta 8080 per impostazione predefinita

3.7.2. Spegnimento del Server

- Arresto normale:
  - Inviare un segnale `SIGTERM` al processo del server
  - Il server Spring Boot avvierà la procedura di arresto ordinato, completando tutte le richieste in sospeso prima di terminare
- Arresto forzato:
  - Inviare un segnale `SIGKILL` al processo del server
  - Il server Spring Boot terminerà immediatamente, perdendo tutte le richieste in sospeso

3.7.3. Gestione degli Errori

- Log degli errori:
  - Tutti gli errori vengono registrati in un file di log
  - Il file di log viene utilizzato per identificare la causa dell'errore e per risolverlo
- Ripristino automatico:
  - Il sistema è in grado di ripristinarsi automaticamente da alcuni errori, come ad esempio un errore di connessione al database
  - Il sistema tenta di riconnettersi al database e ripristinare il servizio
  - Se, dopo alcuni tentativi (oppure dopo un time-out), non riesce a ripristinare il servizio, mostra una pagina di errore 500 all'utente

4. Glossario

Termine	Definizione
Guest	Ospite
Browser	Software per la navigazione in Internet
Prodotto	Merce in vendita sul sito
Dispositivo	Sinonimo di “Prodotto”, può essere uno Smartphone o un Tablet
Modello	Particolare famiglia di prodotti
Variazione	Specializzazione di un modello
Second Hand	Usato
DB	Database
Dominio di Applicazione	Il settore in cui il sistema opera
Logica di Business	Le regole, procedure e le operazioni che definiscono il comportamento del sistema
Cookie	Memorizzazione dati sul browser dell’utente
Operazioni CRUD	Operazioni principali per interagire con le entità persistenti: Create (Crea), Read (Lettura), Update (Modifica), Delete (Elimina)
Web Server	Server dove viene eseguito il codice
VP	Variazione Prodotto