

**System Design Document**

[Versione 1]

**Sommario**

[**Informazioni sul documento** 2](#_Toc157102514)

[**Generalità** 2](#_Toc157102515)

[**Team Project** 2](#_Toc157102516)

[**Revision History** 3](#_Toc157102517)

[**1. Introduzione** 4](#_Toc157102518)

[**1.1 Obiettivo del sistema** 4](#_Toc157102519)

[**1.2 Obiettivi di design (Design goals)** 5](#_Toc157102520)

[**1.3 Definizioni, acronimi e abbreviazioni** 7](#_Toc157102521)

[**1.4 Riferimenti** 8](#_Toc157102522)

[**1.5 Organizzazione del documento** 8](#_Toc157102523)

[**2. Architettura del sistema software corrente** 9](#_Toc157102524)

[**3. Architettura del sistema software proposto** 10](#_Toc157102525)

[**3.1 Panoramica sulla sezione** 10](#_Toc157102526)

[**3.2 Decomposizione in sottosistemi** 10](#_Toc157102527)

[**3.3 Mapping hardware/software** 13](#_Toc157102528)

**Informazioni sul documento**

**Generalità**

* **Progetto**: TechHeaven
* **Versione**: [Versione 1]
* **Documento**: Documento di design del sistema
* **Data**: [23/01/2024]

**Team Project**

|  |  |  |  |
| --- | --- | --- | --- |
| Nome Membro | Matricola | Ruolo | Contatti |
| Dorotea Serrelli | 0512113740 | Project manager | d.serrelli1@studenti.unisa.it |
| Raffaella Sabatino | 0512115114 | Team member | r.sabatino17@studenti.unisa.it |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 23/01/2024 | 0.1 | Scrittura delle sezioni “Introduzione” e “Architettura del sistema software corrente” | Tutto il team |
| 25/01/2024 | 0.2 | Scrittura dei paragrafi “Decomposizione in sottosistemi” e “Mapping hardware/software” della sezione “Architettura del sistema software proposto” | Tutto il team |
| 31/01/2024 | 0.3 | Scrittura paragrafo “Gestione dei dati persistenti” | Dorotea Serrelli |
|  |  |  |  |

**1. Introduzione**

**1.1 Obiettivo del sistema**

La società TechHeavenSrl è responsabile dall’anno 2000 della gestione del negozio “TechHeaven – Il paradiso digitale”, specializzato nella vendita di prodotti elettronici, elettrodomestici, telefonia.

Tale negozio è, attualmente, un punto vendita di riferimento nella zona per lo smercio, la qualità e il prezzo dei prodotti, al punto che soddisfa un grande bacino di utenza, quasi esclusivamente residente nella provincia.

La società intende espandere i confini della propria attività ed ampliare la clientela, avvalendosi di un sistema software che consenta, sotto il profilo soggettivo, una maggiore conoscibilità della società e dell’affidabilità della stessa; sotto il profilo oggettivo, favorisca l’incremento della vendita dei prodotti.

Il sistema software verrà sviluppato per fornire alla clientela informazioni sulla società, sul punto vendita e sui prodotti trattati.

La piattaforma permetterà, infatti, al cliente di registrarsi, in modo da poter visionare i prodotti in vendita ed acquistarli, tenere traccia dello stato degli ordini effettuati presso il negozio online, creare una lista di prodotti desiderati (wishlist).

La piattaforma, inoltre, consentirà l’accesso ai seguenti dipendenti:

* Gestore degli ordini: responsabile del processo di acquisizione, registrazione ed evasione degli ordini dei clienti, nonché dell’elaborazione di richieste di approvvigionamento di prodotti da inoltrare, poi, all’ufficio acquisti.
* Gestore del catalogo: responsabile della presentazione, organizzazione e gestione del catalogo dei prodotti venduti dal negozio.

La piattaforma, quindi, consentirà al gestore degli ordini di visionare gli ordini commissionati dai clienti al negozio e gli ordini che sono stati spediti, preparare un ordine alla spedizione e fare richiesta di approvvigionamento di prodotti mancanti.

Essa, inoltre, permetterà al gestore del catalogo di visionare il catalogo e di poter inserire, cancellare e modificare un prodotto nel catalogo.

**1.2 Obiettivi di design (Design goals)**

In questa sezione sono riportati i Design Goals, ovvero le caratteristiche qualitative che deve possedere il sistema software, la cui formalizzazione esplicita sarà costantemente utilizzata come linea guida per le scelte progettuali che si affronteranno nelle attività successive.

In base alle linee guida delineate dagli autori del libro *Object Oriented Software Engineering* , Bernd Bruegge e Allen H. Dutoit, i design goal sono stati raggruppati nelle seguenti categorie:

* Performance: includono i requisiti di spazio e velocità imposti sul sistema.
* Dependability: determinano quanto sforzo deve essere speso per minimizzare i fallimenti del sistema (crash, falle di sicurezza) e le loro conseguenze.
* Maintenance: determina quanto sforzo è necessario per modificare il sistema dopo il suo rilascio.
* End User: includono qualità che sono desiderabili dal punto di vista dell’utente, ma che non sono state coperte dai criteri di Performance e Dependability.

Nella seguente tabella si riportano le informazioni dei design goals individuati dal RAD : il nome, la descrizione, la categoria di appartenenza e il requisito non funzionale da cui discende.

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Descrizione | Categoria | RNF di origine |
| DG\_1  Quantità dei dati | Il sistema dovrà garantire un’alta capacità di memorizzazione per poter conservare tutte le informazioni necessarie per l’erogazione dei servizi che offre. | Performance | RNF\_11 |
| DG\_2  Gestione permessi sistema | Il sistema deve garantire la separazione netta delle operazioni, sulla base degli utenti che possono effettuarle. | Dependability | RNF\_3 |
| DG\_3  Affidabilità | Il sistema deve garantire che tutte le operazioni avvengano con successo. | Dependability | RNF\_4 |
| DG\_4  Robustezza | Il sistema deve essere in grado di gestire l’inserimento di dati non validi da parte dell’utente, generando messaggi di errore. | Dependability | RNF\_5 |
| DG\_5  Crittografia password utente | Il sistema deve memorizzare la password di accesso di un utente non in chiaro, ossia applicare su di essa un algoritmo crittografico. | Dependability | RNF\_6 |
| DG\_6  Protezione da attacchi SQLInjection | Il sistema deve essere protetto da attacchi di tipo SQLInjection. | Dependability | RNF\_7 |
| DG\_7  Protezione da Cross-site scripting | Il sistema deve difendersi da attacchi di tipo Cross-site scripting. | Dependability | RNF\_8 |
| DG\_8  Protocollo HTTPS | Il sistema deve proteggere l’integrità e la riservatezza dei dati scambiati tra client e server, mediante l’uso di un certificato SSL/TLS. | Dependability | RNF\_9 |
| DG\_9  Portabilità | Il sistema sarà disponibile tramite pagina web da qualsiasi dispositivo (principalmente tablet, smartphone, computer). Ciò rende il sistema altamente portabile in quanto non è necessario adattare il sistema tra diversi sistemi operativi. | Dependability | RNF\_18  RNF\_14 |
| DG\_10  Manutenibilità | Il sistema dovrà essere sviluppato seguendo i principali standards per la buona manutenibilità. | Maintenance | RNF\_12 |
| DG\_11  Estendibilità | Il sistema dovrà essere sviluppato seguendo i principali standards che garantiscono una buona estendibilità delle funzionalità. | Maintenance | RNF\_13 |
| DG\_12  Facilità di utilizzo | Il sistema deve essere facilmente utilizzabile anche da un utente non esperto. | End user | RNF\_1 |
| DG\_13  Interfaccia intuitiva | L’interfaccia utente della piattaforma deve permettere di eseguire azioni in modo chiaro e semplice, rendendo ben esplicita la funzionalità di ogni elemento visuale. | End user | RNF\_2 |
| DG\_14  Sistema responsive | Il sistema dovrà offrire un’interfaccia grafica utente che sia in grado di adattarsi, principalmente, allo schermo di uno dei seguenti dispositivi: smartphone, tablet e computer. | End user | RNF\_10 |

**1.3 Definizioni, acronimi e abbreviazioni**

In questa sezione si riportano di seguito l’insieme delle definizioni, degli acronimi e delle abbreviazioni che verranno utilizzati in questo documento:

* Design goals: le caratteristiche qualitative che il sistema software deve possedere, ottenute dal dominio applicativo e dal cliente/dagli utenti finali;
* Sottosistema: una parte sostituibile del sistema software che offre un sottoinsieme di servizi che soddisfano una o più funzionalità definite nel RAD;
* Dati persistenti: i dati che sopravvivono alla singola esecuzione del sistema e che, dunque, devono essere memorizzati;
* Mapping hardware/software: lo studio della connessione tra le componenri logiche e le compnenti fisiche che costituiscono il sistema;
* Boundary conditions: le modalità di avvio, spegnimento e configurazione del sistema;
* RAD: Requirements Analysis Document;
* SDD: System Design Document.

**1.4 Riferimenti**

Di seguito si elencano i documenti utili alla comprensione del documento:

* Problem statement;
* RAD;
* …

**1.5 Organizzazione del documento**

Il presente documento di System Design consta di quattro sezioni:

1. **Introduzione**: si descrivono lo scopo del sistema, gli obiettivi di design su cui il sistema deve essere svluppato, le definizioni e gli acronimi utilizzati nel documento ed i riferimenti agli altri documenti rilasciati.
2. **Architettura del sistema software corrente**: si delinea lo stato attuale dell’architettura del software già presente.
3. **Architettura del sistema software proposto**: si illustra come il sistema verrà decomposto nei sottosistemi, il mapping Hardware/Software dei sottosistemi, la gestione dei dati persistenti, le politiche di sicurezza e di gestione degli accessi adottate e il flusso di controllo del sistema; inoltre, verranno presentate la struttura dei singoli sottosistemi e le boundary conditions riguardanti l’intero sistema.
4. **Glossario:** riporta la lista dei termini usati nel documento con annessa spiegazione.

**2. Architettura del sistema software corrente**

La società, al momento, non presenta un sistema informatico in grado di raccogliere gli ordini dei clienti in modo automatico, bensì prende in carico solo gli ordini effettuati da clienti che risiedono nella regione in cui è localizzato il negozio.

Attualmente, gli ordini richiesti dai clienti vengono inviati al negozio tramite telefono o fax.

Come si evince dal contesto appena descritto, questa modalità di gestione degli ordini comporta perdita di tempo per la raccolta dei dati ed oneri aggiuntivi per la relativa gestione, a discapito della produttività dell’azienda.

**3. Architettura del sistema software proposto**

**3.1 Panoramica sulla sezione**

In questa sezione si descrivono le scelte progettuali adottate e le modalità con cui sono state condotte le seguenti attività di progettazione del sistema:

* Decomposizione del sistema;
* Mapping hardware/software;
* Gestione dei dati persistenti;
* Gestione della sicurezza e del controllo degli accessi;
* Flusso di controllo globale del sistema;
* Boundary conditions.

Sono stati considerati diversi stili architetturali per il sistema software proposto (client-server, peer-to-peer, three-tier, repository, …) e, tra questi, si è scelto di optare per lo stile Three-tier combinato con il design pattern Observer.

Il motivo di tale scelta risiede nel fatto che lo stile architetturale Three-tier è adatto per i sistemi interattivi, in cui è richiesto l’intervento dell’utente e si devono gestire interazioni più complesse tra i componenti del sistema, e, poi, la separazione della logica di presentazione da quella relativa all’elaborazione ed archiviazione dei dati offre una serie di vantaggi:

* Leggibilità;
* Manutenzione;
* Riuso.

Si è pensato di adottare il design pattern Observer per poter consentire ai componenti che si trovano nel livello di presentazione di poter reagire ai cambiamenti degli elementi nel livello di Storage.

Nello sviluppo del sistema si utilizzeranno per la presentazione dei dati e l'interfaccia utente HTML, CSS e Javascript, mentre per la logica applicativa e, quindi, il back-end si utilizzerà Java e JSP (JavaServer Pages).

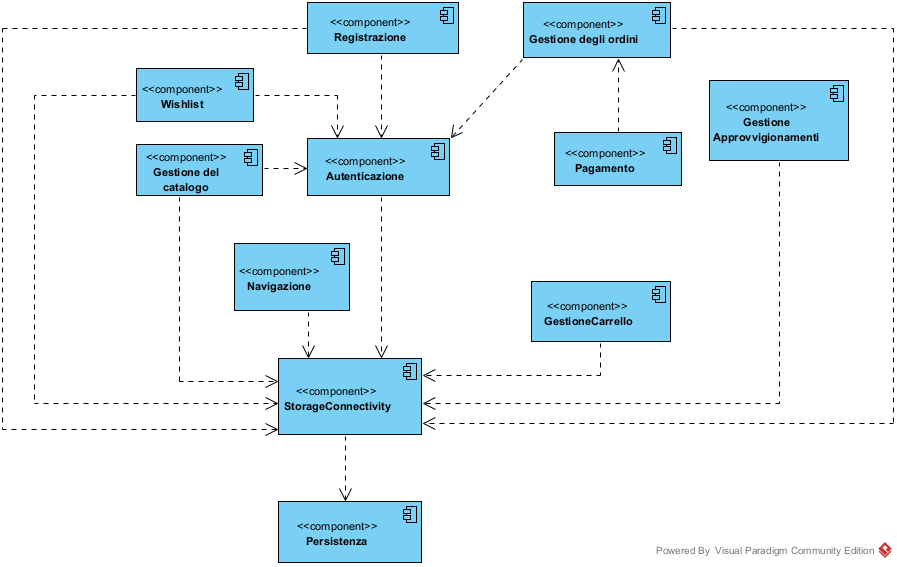
Per quanto riguarda la gestione dei dati persistenti, si è scelto di utilizzare un database relazionale con DBMS MySQL, che utilizza il linguaggio SQL per gestire e interrogare i dati, poiché il database è scalabile e fornisce servizi per la gestione della concorrenza, il controllo degli accessi, il recupero da arresto.

**3.2 Decomposizione in sottosistemi**

Analizzando il modello dei casi d’uso ed il modello dinamico presenti nel RAD, si sono individuati i seguenti sottosistemi:

* Registrazione : si occupa di gestire la registrazione dei nuovi clienti del negozio;
* Autenticazione: è responsabile delle funzionalità Login, Logout, accesso all’area riservata, modifica dei dati personali e reimpostazione della password associati al cliente;
* Wishlist : si occupa della visualizzazione della lista dei desideri del cliente, dell’aggiunta di prodotti e della rimozione degli stessi dalla wishlist;
* Gestione carrello: è responsabile della visualizzazione del carrello virtuale e delle operazioni di aggiunta, rimozione e variazione delle quantità di un prodotto all’interno del carrello;
* Navigazione: si occupa della ricerca di un prodotto nel catalogo per barra di ricerca o per menù di navigazione, della visualizzazione della pagina di risultati della ricerca e della visualizzazione delle referenze di un prodotto;
* Gestione degli ordini: offre servizi relativi alla creazione (check-out del carrello) e preparazione di un ordine del cliente alla spedizione, della visualizzazione degli ordini da spedire e degli ordini evasi dal gestore degli ordini;
* Gestione approvvigionamenti: si occupa della visualizzazione ed elaborazione di richieste di approvvigionamento di prodotti fatte dal gestore degli ordini;
* Gestione del catalogo: è responsabile della gestione del catalogo detenuta dal gestore del catalogo (visualizzazione del catalogo, aggiunta di un prodoto, rimozione di un prodotto, modifica delle specifiche di un prodotto);
* Persistenza: si occupa di gestire la persistenza dei dati con un database;
* Storage Connectivity: si interpone tra i vari sottosistemi e il sottosistema “Persistenza”.

Di seguito si riporta in UML il component diagram relativo alla decomposizione del sistema software:

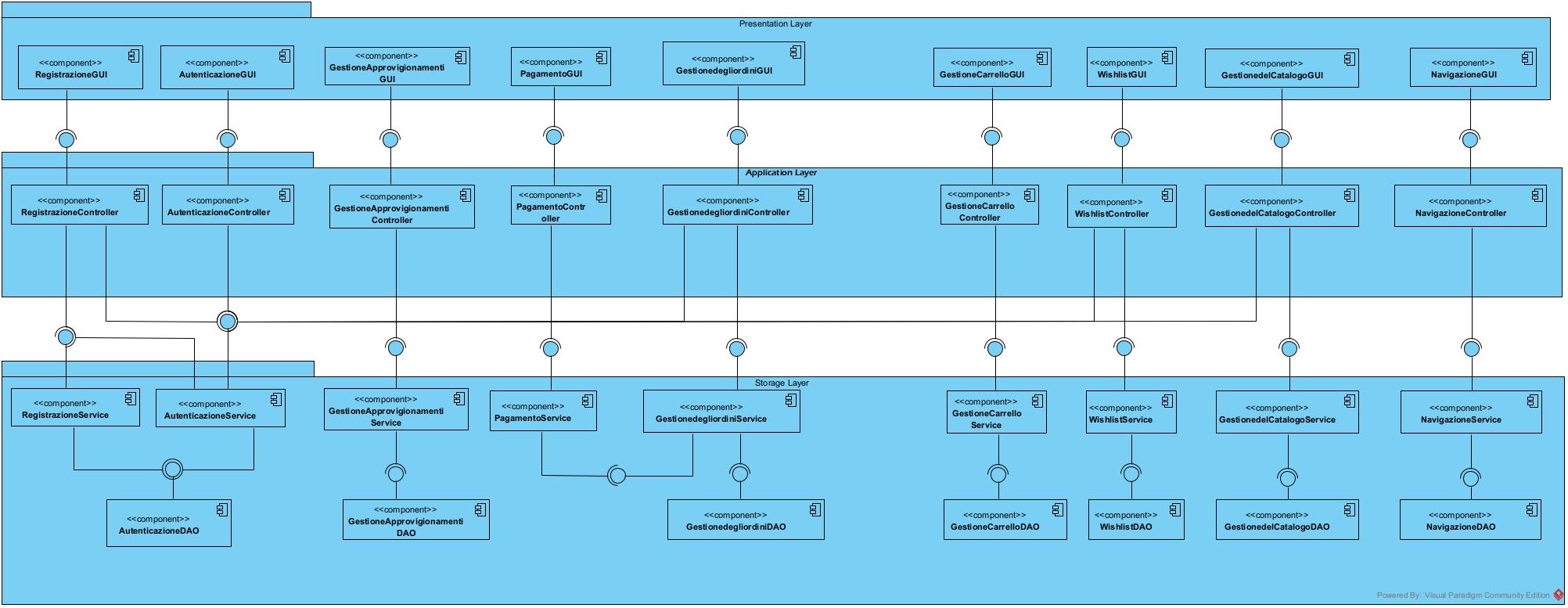


Alcuni sottosistemi saranno gestiti dalle seguenti componenti COTS (Commercial off-the-shelf):

* Persistenza sarà gestita attraverso un DBMS relazionale su sistema MySQL.

Si riporta, inoltre, il diagramma architetturale del sistema software, offrendo una una vista dettagliata di ciascun sottosistema, evidenziando le componenti principali:

* GUI (Graphic User Interface): contiene le varie view che saranno renderizzate per creare le pagine web da mostrare al cliente.
* Controller: si occupa della logica per il controllo del sistema.
* Service: si occupa della logica di business.
* DAO: Data Access Object, che si occupa di fornire accesso ai dati persistenti.



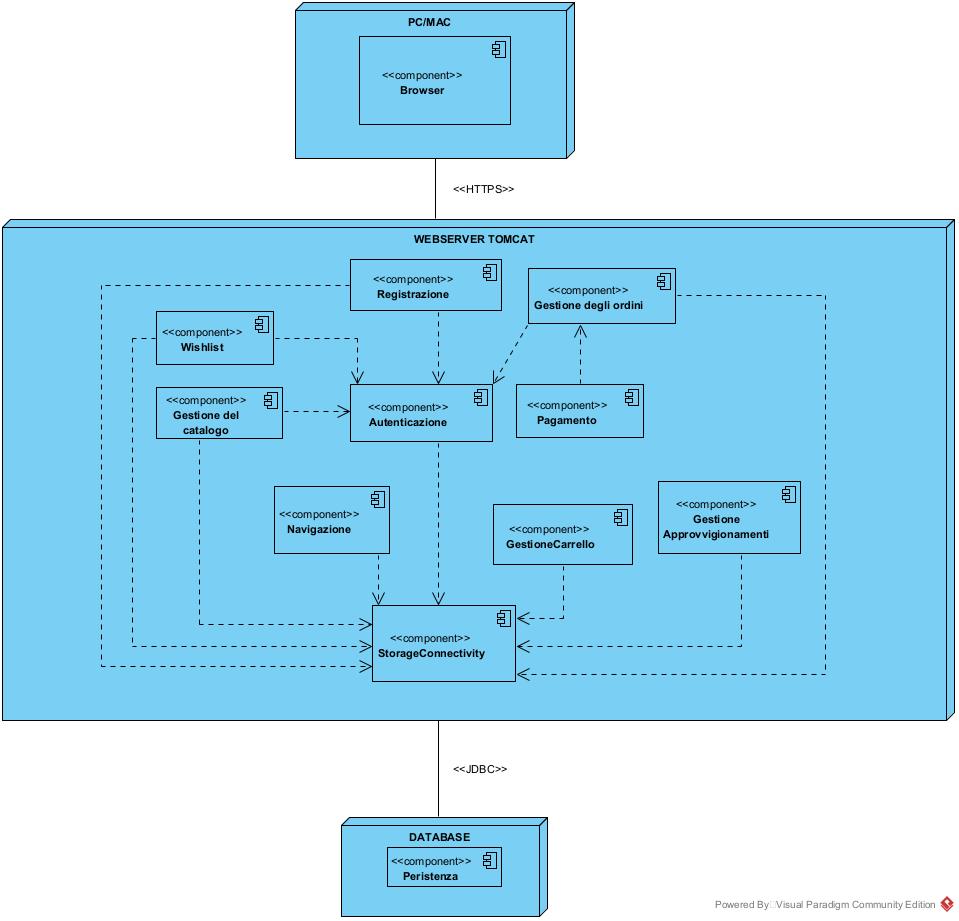
Descrizione sottosistemi ….

**3.3 Mapping hardware/software**

L’ applicazione web che verrà sviluppata si basa su una piattaforma hardware costituta da un server che risponde alle richieste effettuate dai clienti; tali clienti effettueranno le richieste da una qualsiasi macchina con un browser ed una connessione ad Internet.

Visto che il nostro sistema è una applicazione Web-based, residente su un web server basata su un’architettura non distribuita, risiede su un solo nodo.

Si illustra di seguito un UML deployment diagram che descrive il mapping hardware/software.



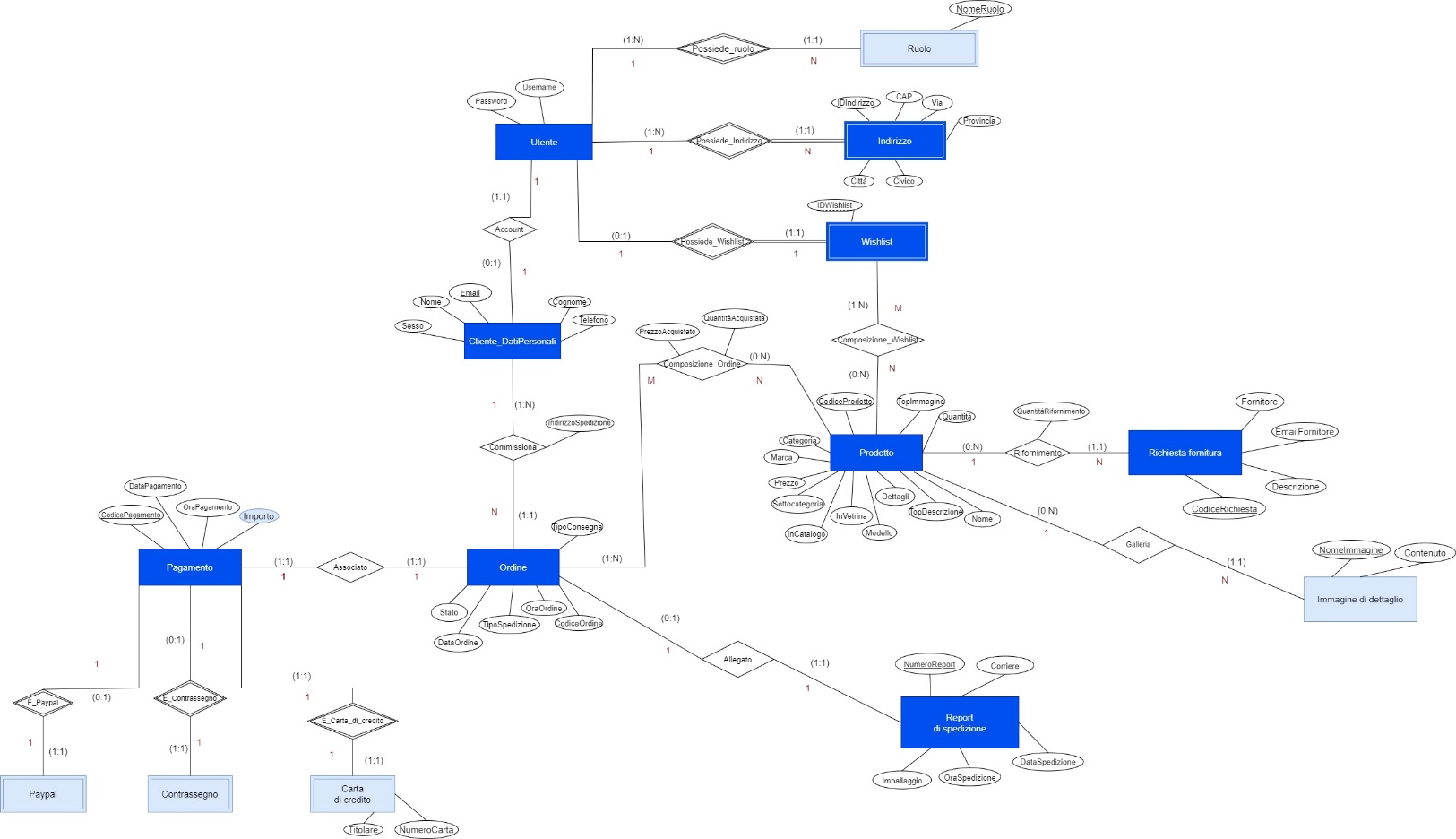
**3.4 Gestione dei dati persistenti**

Per la gestione della memorizzazione dei dati persistenti del sistema si è deciso di utilizzare un database relazionale, al fine di gestire agevolmente l’accesso concorrente ai dati e, allo stesso tempo, garantire la consistenza dei dati tramite l’utilizzo di un DBMS.

La scelta di utilizzare un DBMS è guidata dalla volontà di rimanere coerenti con i design goals stabiliti, potendo contare su:

* **Imposizioni di vincoli di integrità sui dati:** un DBMS permette di specificare diversi tipi di vincoli per garantire l’integrità dei dati e, poi, ad ogni cambiamento di stato del database, controlla che tali vincoli siano soddisfatti.
* **Privatezza dei dati**: un DBMS permette un accesso protetto ai dati. Utenti diversi (nel nostro caso il cliente, il gestore degli ordini e il gestore del catalogo) possono avere accesso ad apposite porzioni del database e sono abilitati ad eseguire determinate operazioni su di esse.
* **Affidabilità dei dati**: un DBMS offre dei metodi per salvare copie dei dati e per ripristinare lo stato della base di dati in caso di guasti software e hardware.
* **Atomicità delle operazioni**: un DBMS permette di effettuare sequenze di operazioni in modo atomico, ovvero o l’intera sequenza di operazioni viene eseguita con successo oppure nessuna di queste operazioni viene effettuata. L’atomicità delle transazioni permette di mantenere uno stato del database consistente con la realtà modellata.

Si riporta lo schema EER ristrutturato per la progettazione concettuale del database:



E lo schema relazionale per la progettazione concettuale del database.

Per maggiori dettagli si consiglia la visione del documento di design del database presente nei documenti Deliverables rilasciati al cliente.