Università degli Studi di Salerno

Corso di Ingegneria del Software

e-comix

SDD
Versione 1.3

Data: 15/11/2017

Partecipanti:

Nome	Matricola
Gerardo De Rosa	0512103762
Pasquale Nappo	0512103046
Gerardo Ragosta	0512101794

Scritto da:	Gerardo De Rosa, Pasquale Nappo, Gerardo Ragosta

Revision History

Data	Versione	Descrizione	Autore
15/11/2017	1.0	Stesura dell'Introduzione	Gerardo De Rosa Pasquale Nappo
16/11/2017	1.1	Stesura Scopo del Sistema e Design goals	Pasquale Nappo
17/11/2017	1.2	Stesura completa dell'SDD	Pasquale Nappo Gerardo Ragosta
20/11/2017	1.3	Revisione documento	Pasquale Nappo Gerardo Ragosta

Indice

1. Introduzione

1.1 Scopo del Sistema

L'obiettivo del progetto è di creare una piattaforma e-commerce che dia la possibilità a un rivenditore fisico di fumetti e gadget di poter svolgere la propria attività online, dia la possibilità di visualizzare, ricercare, confrontare e acquistare un prodotto presente nella piattaforma.
In particolare la piattaforma e-comix nasce con l'obiettivo di:

- Fornire un catalogo di prodotti, tra cui: fumetti e gadget, organizzando i vari prodotti in modo ordinato, in modo da non rendere la piattaforma dispersiva e di rendere la ricerca dei prodotti immediata.
- Fornire un servizio per l'acquisto di fumetti e gadget mediante l'aggiunta del prodotto al carrello.
- Offrire informazione e dettagli riguardanti i prodotti della piattaforma.

1.2 Design goals

Il sistema fornirà una struttura chiara, completa e semplice. L'utente non dovrà necessariamente effettuare operazioni che richiedano una conoscenza della piattaforma, poiché l'utilizzo del sistema da parte dell'utente sarà guidato dall'interfaccia semplice e intuitiva.

L'interfaccia grafica sarà curata nei minimi dettagli, mediante l'utilizzo di bottoni, finestre di dialogo non invasive, label semplici e icone dettagliate e chiare, offrendo così all'utente un'esperienza di utilizzo rapida e semplificata anche per utenti meno esperti.

1.2.1 Criteri di affidabilità

La piattaforma garantirà il corretto funzionamento, gestendo vari tipi di errori che potranno verificarsi durante l'utilizzo ed eventuali attacchi alla sicurezza. Quindi, e-comix rispetterà i seguenti requisiti di qualità, relativi all'affidabilità:

- **Robustezza (priorità alta):** nel caso in cui l'utente inserisca dati errati nel sistema, quest'ultimo farà visualizzare dei messaggi di errore avvisando l'utente che i dati non sono validi. [Generalizzazione del Requisito Non Funzionale: **RNF_2: Affidabilità** trattato nel paragrafo 3.3.2 del RAD]

- **Affidabilità (priorità alta):** il sistema garantisce il corretto svolgimento delle proprie funzionalità, producendo sempre l'output desiderato. [Requisito non Funzionale: **RNF_2: Affidabilità** trattato nel paragrafo 3.3.2 del RAD]

- **Disponibilità (priorità media):** il sistema sarà sempre funzionante, tranne nei periodi di manutenzione. [Generalizzazione del Requisito non Funzionale: **RNF_2: Affidabilità** trattato nel paragrafo 3.3.2 del RAD]
- **Fault tolerance (priorità alta)**: il sistema garantisce una tolleranza media agli errori, qualora si verificassero esso dovrà essere in grado di gestirli e risolverli nel minor tempo possibile. [Dedotto dal dominio di applicazione]

- **Sicurezza (priorità media):** l'accesso al sistema sarà controllato da un apposito sistema di autenticazione, che permetta ad ogni utente di accedere alla propria area senza modificare le operazioni altrui, garantendo la privacy. In questo modo nessun utente potrà visualizzare i dati sensibili e accedere alle funzionalità degli altri utenti. [Requisito non Funzionale: **RNF_2: Affidabilità** trattato dal paragrafo 3.3.2 del RAD]

1.2.2 Criteri di rendimento

Il sistema garantirà buone performance gestendo adeguatamente tutti gli utenti senza rallentamenti. e-comix, rispetterà i seguenti requisiti di qualità:

- **Tempo di risposta (priorità medio-alta):** il sistema deve garantire un tempo di risposta inferiore ai 3 secondi soprattutto per le funzionalità stimate tra quelle più frequenti come il login. [Requisito non Funzionale: **RNF_3: Prestazioni** trattato nel paragrafo 3.3.3 del RAD]

- **Throughput (priorità media):** il sistema dovrà completare il maggior numero possibile di operazioni nel minor tempo possibile. [Requisito non Funzionale: **RNF_3: Prestazioni** trattato nel paragrafo 3.3.3 del RAD]

1.2.3 Criteri di manutenzione

Il sistema sarà facilmente manutenibile e rispetta i seguenti requisiti:

- **Estensibilità (priorità media):** il sistema sarà realizzato in maniera tale da garantire l'aggiunta di nuove funzionalità in maniera semplice. [Generalizzazione del Requisito non Funzionale: **RNF_4: Manutenibilità** trattato dal paragrafo 3.3.4 del RAD]

- **Modificabilità (priorità alta):** il sistema dovrà essere realizzato in maniera da supportare lo sviluppo di aggiornamenti in modo semplice. [Requisito non Funzionale: **RNF_4: Manutenibilità** trattato nel paragrafo 3.3.4 del RAD]

- **Tracciabilità dei requisiti (priorità alta):** grazie ad una buona e coerente documentazione, sarà facile risalire ai rispettivi requisiti funzionali, use case e altri artefatti, a cui fanno riferimento le varie classi e metodi del sistema. [Documento RAD e matrice di tracciabilità]

1.2.4 Criteri per l'utente finale

Dal punto di vista dell'utente, il sistema dovrà garantire i seguenti di qualità:

- **Utilità (priorità alta):** attraverso l'attività di raccolta dei requisiti, il sistema sarà in grado di soddisfare le esigenze degli utenti. [dedotto dal dominio di applicazione]

- **Usabilità (priorità alta):** il sistema dovrà essere intuitivo e di semplice utilizzo, e sarà progettato tenendo conto di quella che è l'user experience degli studenti. Non sarà necessario l'uso di un manuale utente per compiere le azioni. [Requisito non Funzionale: **RNF_1: Usabilità** trattato nel paragrafo 3.3.1 del RAD]

1.2.5 Trade-offs

- **Spazio vs Velocità:** il sistema lavorerà salvaguardando la memoria minimizzando l'utilizzo di essa senza rallentare eccessivamente il sistema.

- **Tempo di rilascio vs Funzionalità:** Verrà rilasciata una prima versione del sistema che comprenderà non tutte le funzionalità ma solo quelle

con alta priorità. Le altre funzionalità saranno rilasciate nelle successive release.

1.3 Definizione, acronimi e abbreviazioni.

RAD: Requirement Analysis Document.
SSD: System Design Document.
HW: Hardware.
SW: Software.
SQL: Structured Query Language.
DBMS: Database Management System.
GUI: Graphical User Interface.

1.4 Riferimenti

- USU_RAD_V_3.00
- Bruegge, Dutoit, Object-Oriented Software Engineering.
- Sommerville, Software Engineering-Addison Wesley.
- Standard StdIEE 830-1998;
- DDI_SDD_2.0 (template offerto dal Top Manager).

1.5 Panoramica

Il seguente documento è strutturato in quattro parti:

- **Introduzione:** riporta una descrizione del sistema specificando le ragioni del suo sviluppo, le caratteristiche del sistema e un accenno sull'utilizzo delle funzioni.

- **Architettura software proposta:** fornisce una panoramica sull'architettura usata per il sistema. I punti riportati per l'architettura del sistema proposto sono: la suddivisione in sottosistemi, il mapping software-hardware, la gestione dei dati persistenti, il controllo degli accessi di sicurezza, flusso di controllo globale e le condizioni limite.

- **Servizio dei sottoinsiemi:** espone una descrizione dei sottosistemi identificati e per ognuno i servizi offerti.

- **Glossario:** è una raccolta di termini di quest'ambito specifico.

2. Architettura software attuale

L’architettura proposta di seguito non andrà a sostituire nessun sistema preesistente.
La progettazione e lo sviluppo del software del software inizierà da zero e seguirà i criteri della Greenfield Engineering.

3. Architettura software proposta

3.1 Panoramica

L’architettura scelta per il sistema da realizzare sarà quella Three-layer. L’utente potrà interagire con l’application layer mediante l’interface layer che offrirà diverse interfacce in base alle necessità dell’utente. L’application layer dovrà poi comunicare con il database per la memorizzazione dei dati persistenti. Sul server, risiede un DBMS che si occupa di recuperare, memorizzare ed interrogare i dati presenti nel database, elaborando, quindi, la richiesta dell’utente. L’aspetto della concorrenza di accessi multipli al database, sarà pertanto gestito dal DBMS stesso che dovrà evitare eventuali colli di bottiglia. Tale architettura conferisce all’intero sistema una maggiore manutenibilità e permette di gestire il problema della concorrenza degli accessi in maniera semplice ed efficace.

3.2 Scomposizione del sottosistema

I tre livelli relativi all’architettura adottata sono:

- InterfaceLayer
- ApplicationLogicLayer
- DataStorageLayer

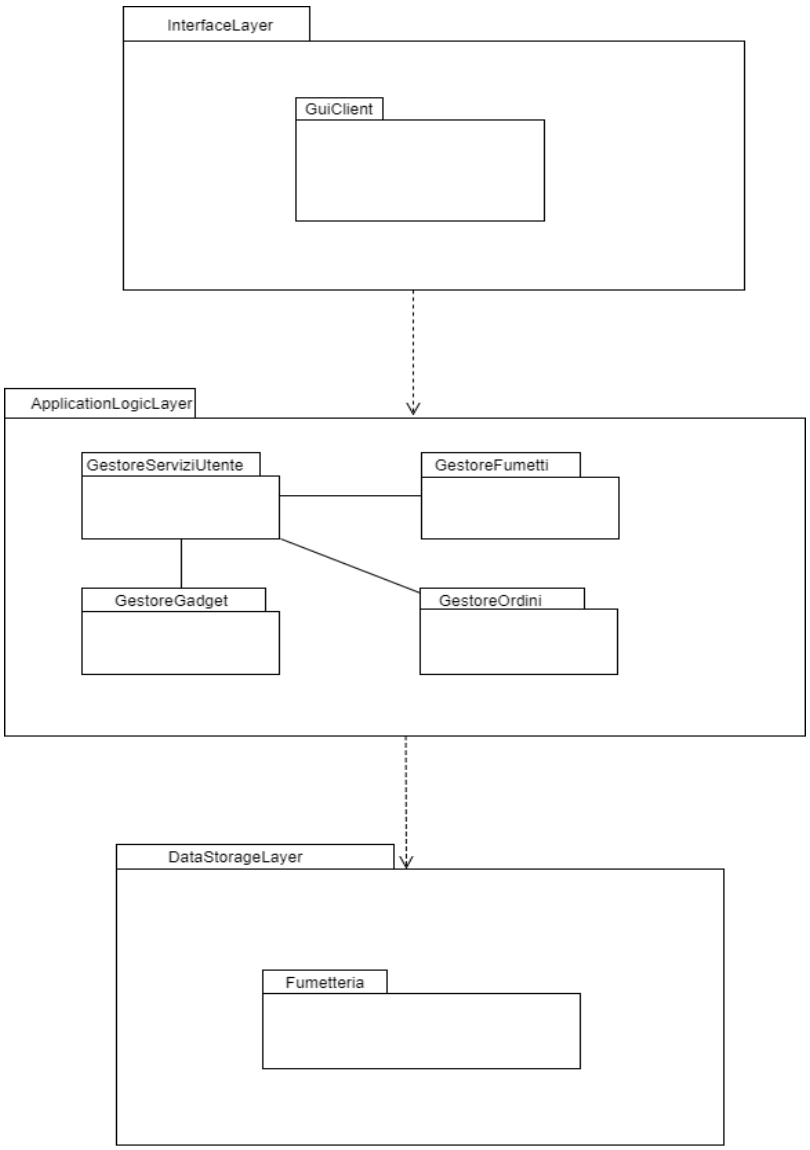
InterfaceLayer: si occupa di gestire l’interfaccia utente; include gli oggetti boundary con cui l’utente interagisce (finestre, form, pagine web, ecc..)

ApplicationLogicLayer: comprende le componenti logiche (oggetti control e entity), responsabili del corretto funzionamento del sistema, e utilizza il database per effettuare operazioni sui dati.

DataStorageLayer: comprende un gestore che si occupa di effettuare operazioni di inserimento, cancellazione e aggiornamento e si occupa di rendere disponibili i dati presenti all'interno del database.

3.2.1 Schema Generale

Di seguito è riportato lo schema generale della suddivisione. Successivamente verranno descritti tutti i sottosistemi più nel dettaglio.

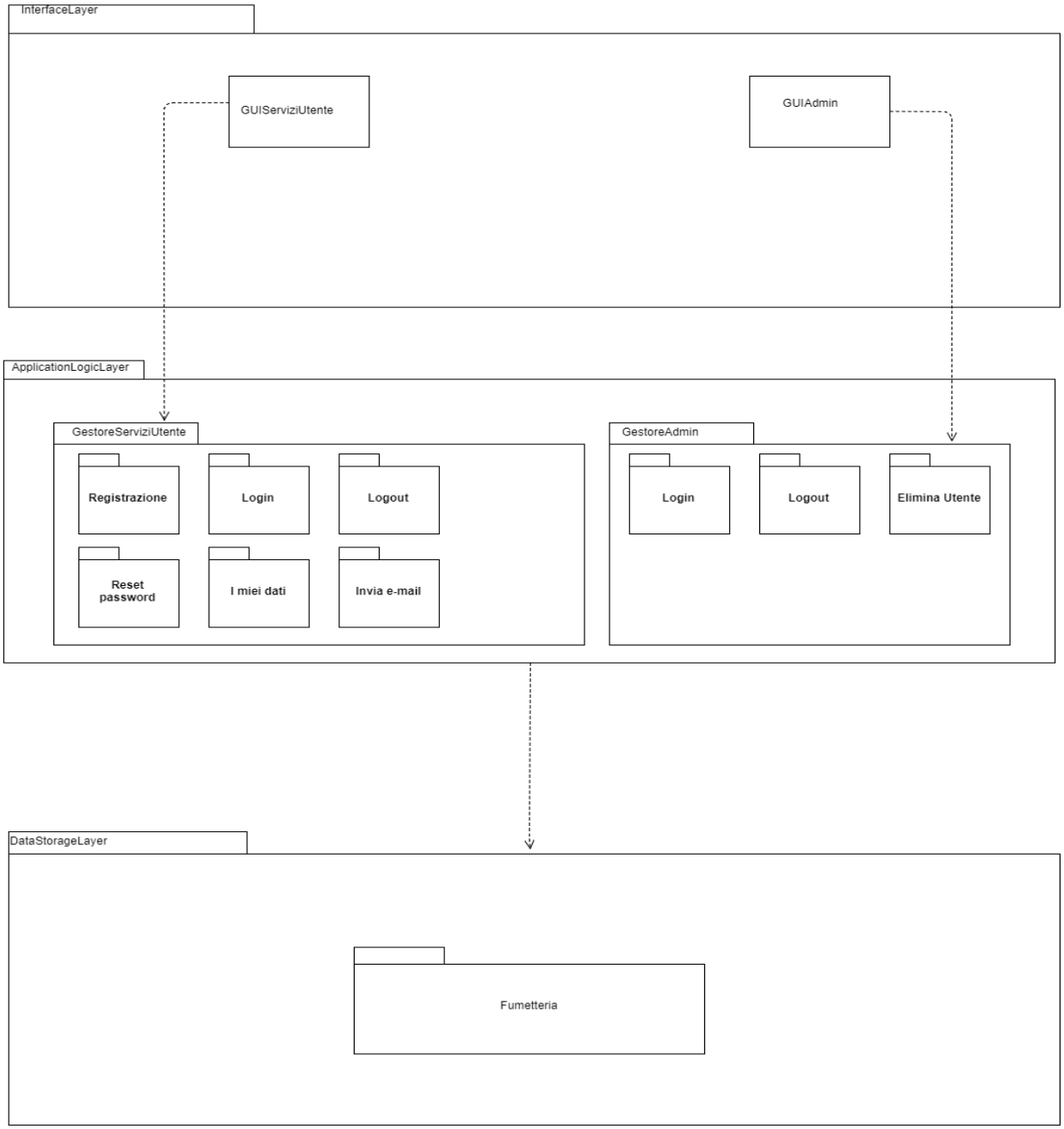


InterfaceLayer: GUIClient indica i sottosistemi che contengono tutti gli oggetti boundary.

ApplicationLogicLayer: contiene i quattro sottosistemi individuati (GestoreServiziUtente, Gestore Fumetti, Gestore Gadget, Gestore Ordini);

DataStorageLayer: sottosistema che ha il compito di effettuare operazioni verso il database.

3.2.2 Gestione Utente



InterfaceLayer:

Include tutte le componenti dell'interfaccia grafica del sistema che offrono le funzionalità per l'accesso a e-comix accessibili da uno o più utenti.

- **GUIServiziUtente:** comprende tutte le interfacce dei servizi offerti dall'utente.
- **GUIAdmin:** tramite questa interfaccia, l'amministratore può accedere alle funzionalità rese solo a lui disponibili.

ApplicationLogicLayer:

Il sottosistema è decomposto ulteriormente in altri due sottosistemi:

- **GestoreServiziUtente:** sottosistemi che contiene le funzionalità per l'accesso al sistema come:

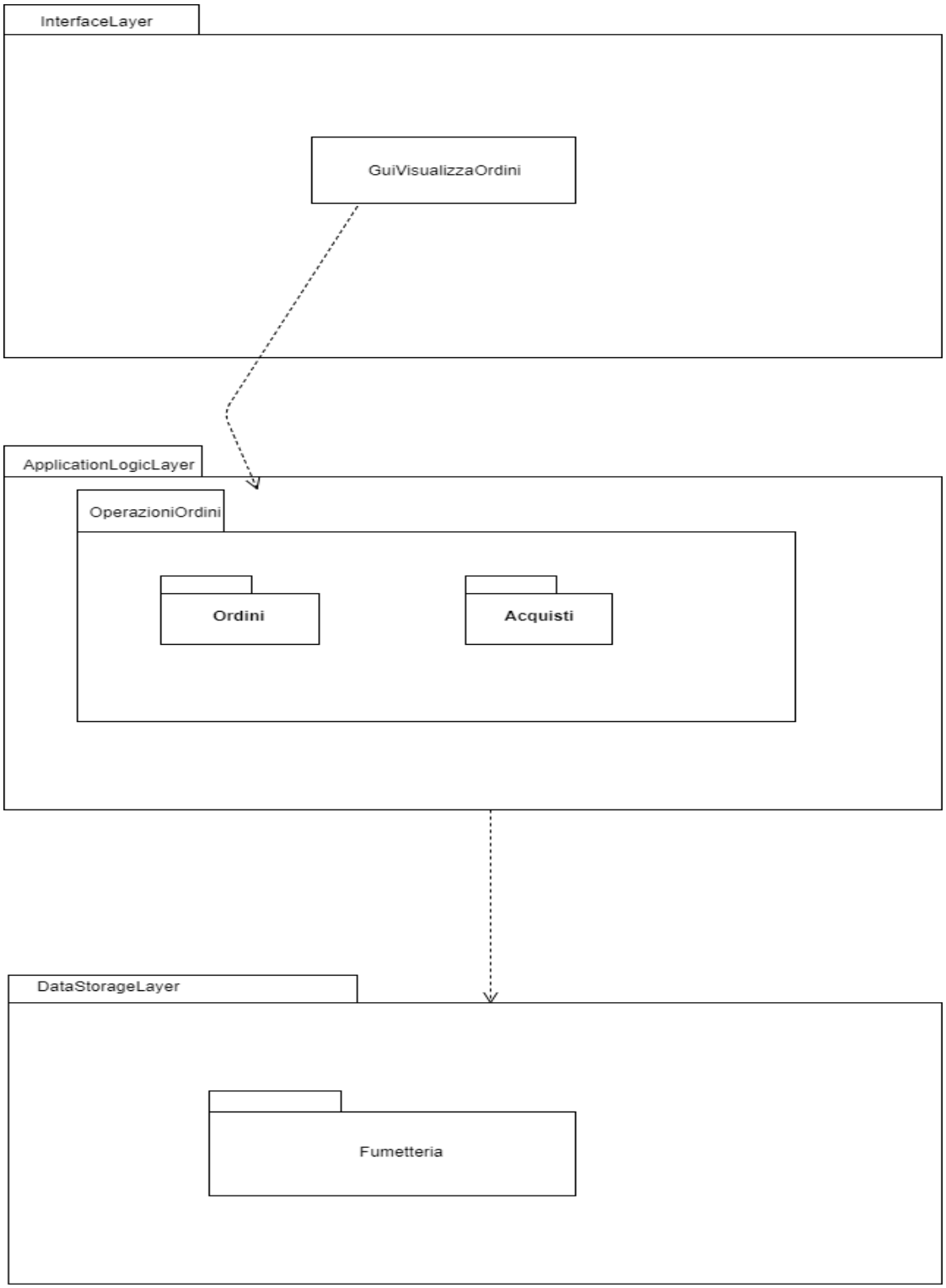
  - **Registrazione:** operazione che permette all'utente di registrarsi
  - **Login:** operazione che permette l'autenticazione al sistema
  - **Logout:** operazione che permette all'utente di scollegarsi dal sistema
  - **Reset Password:** operazione che permette all'utente di resettare la sua password
  - **I miei dati:** operazione che permette all'utente di visualizzare i propri dati
  - **Invia e-mail:** operazione che permette all'utente di inviare un'e-mail

- **GestoreAdmin:** sottosistema che comprende le funzionalità dell'amministratore del sistema:

  - **Login:** operazione che permette l'autenticazione al sistema
  - **Logout:** operazione che permette all'utente di scollegarsi dal sistema
  - **EliminaUtente:** operazione che permette all'amministratore di eliminare un utente dal sistema

DataStorageLayer: Comprende **Fumetteria** che permette di effettuare operazioni sul database.

3.2.1 Gestione Gestore Ordini



InterfaceLayer:

Include tutte le componenti dell'interfaccia grafica del sistema che offrono le funzionalità per la gestione dei prodotti:

- **GUIVisualizzaOrdini:** interfaccia che permette la visualizzazione da parte dell'amministratore degli ordini effettuati dall'utente

ApplicationLayer:

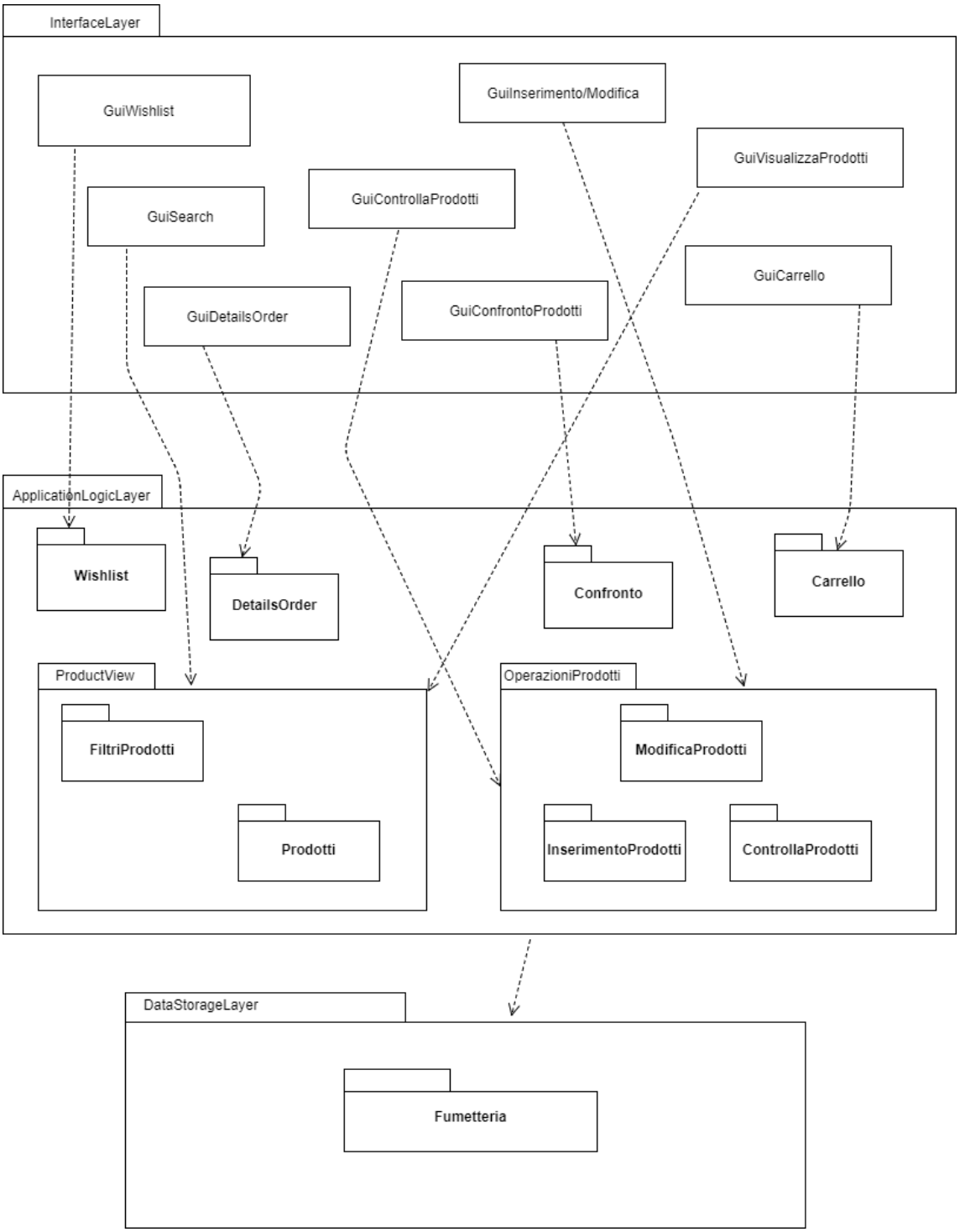
Comprende tutte le componenti logiche associate alla gestione di un ordine:

- **Ordini:** operazione che permette all'amministratore di visualizzare gli ordini degli utenti
- **Acquisti:** operazione che permette agli utenti di visualizzare i propri ordini

DataStorageLayer:

Comprende Fumetteria che permette di effettuare operazioni sul database

3.2.4 Gestore Prodotti



InterfaceLayer:

Include tutte le componenti dell'interfaccia grafica del sistema che offrono le funzionalità sulla gestione dei prodotti:

- **GUIWishList**: interfaccia che permette la visualizzazione dei prodotti aggiunti alla lista dei desideri
- **GUISearch:** interfaccia che permette la visualizzazione dei prodotti ricercati
- **GUIControllaProdotti:** interfaccia che permette di gestire i prodotti del sistema
- **GUIInserimento/Modifica:** interfaccia che permette di inserire o modificare i prodotti
- **GUIVisualizzaProdotti:** interfaccia che permette al gestore di visualizzare tutti i prodotti presenti all'interno della piattaforma
- **GUIDetailsOrder:** interfaccia che permette di visualizzare i dettagli degli ordini
- **GUIConfrontoProdotti:** interfaccia che permette di visualizzare i prodotti aggiunti al confronto
- **GUICarrello:** interfaccia che permette di visualizzare tutti i prodotti aggiunti al carrello

ApplicationLogicLayer:

Comprende tutte le componenti logiche associate alla gestione di un prodotto:

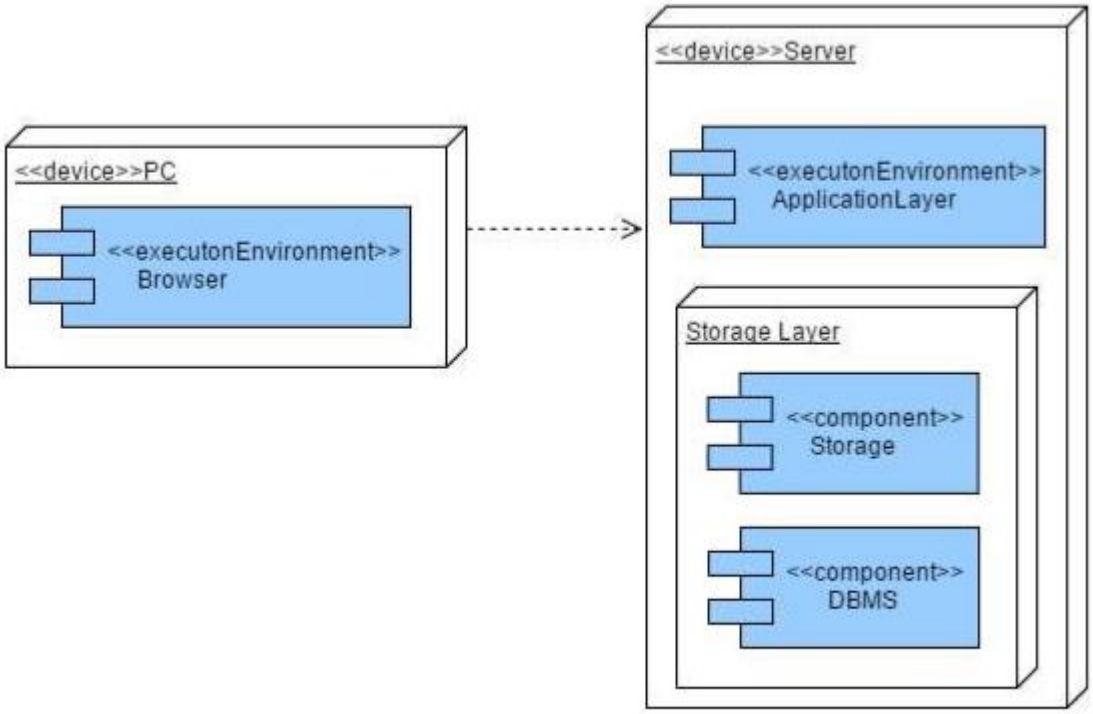
- **Wishlist:** operazione che inserisce un prodotto alla lista dei desideri
- **DetailsOrder:** operazione che fa visualizzare i dettagli di un prodotto
- **Confronto:** operazione che inserisce un prodotto al confronto
- **Carrello:** operazione che inserisce un prodotto all'interno del carrello
- **Prodotti:** operazione che fa visualizzare i prodotti ricercati
- **FiltriProdotti:** operazione che applica filtri alla ricerca di un prodotto
- **InserimentoProdotti:** operazione che inserisce prodotti all'interno della piattaforma
- **ControllaProdotti:** operazione che fa visualizzare tutti i prodotti del sistema al gestore
- **ModificaProdotti:** operazione che modifica i prodotti all'interno della piattaforma

DataStorageLayer:

Comprende Fumetteria che permette di effettuare operazioni sul database

3.3 Hardware/Software mapping

Il sistema sviluppato sarà installato su un solo server e utilizzerà un DBMS MySQL stanziato sullo stesso per la gestione dei dati persistenti.
Il sistema sarà diviso in un'architettura client e server.



Il Deployment Diagram mostra le componenti che utilizzerà il nostro sistema; questo diagramma aiuta gli sviluppatori a comprendere le relazioni tra le componenti software e i nodi hardware.
Al lato client, l'interfaccia utente verrà mostrata sul browser web interagirà con l'application layer, che a sua volta memorizza i dati presenti sul database.

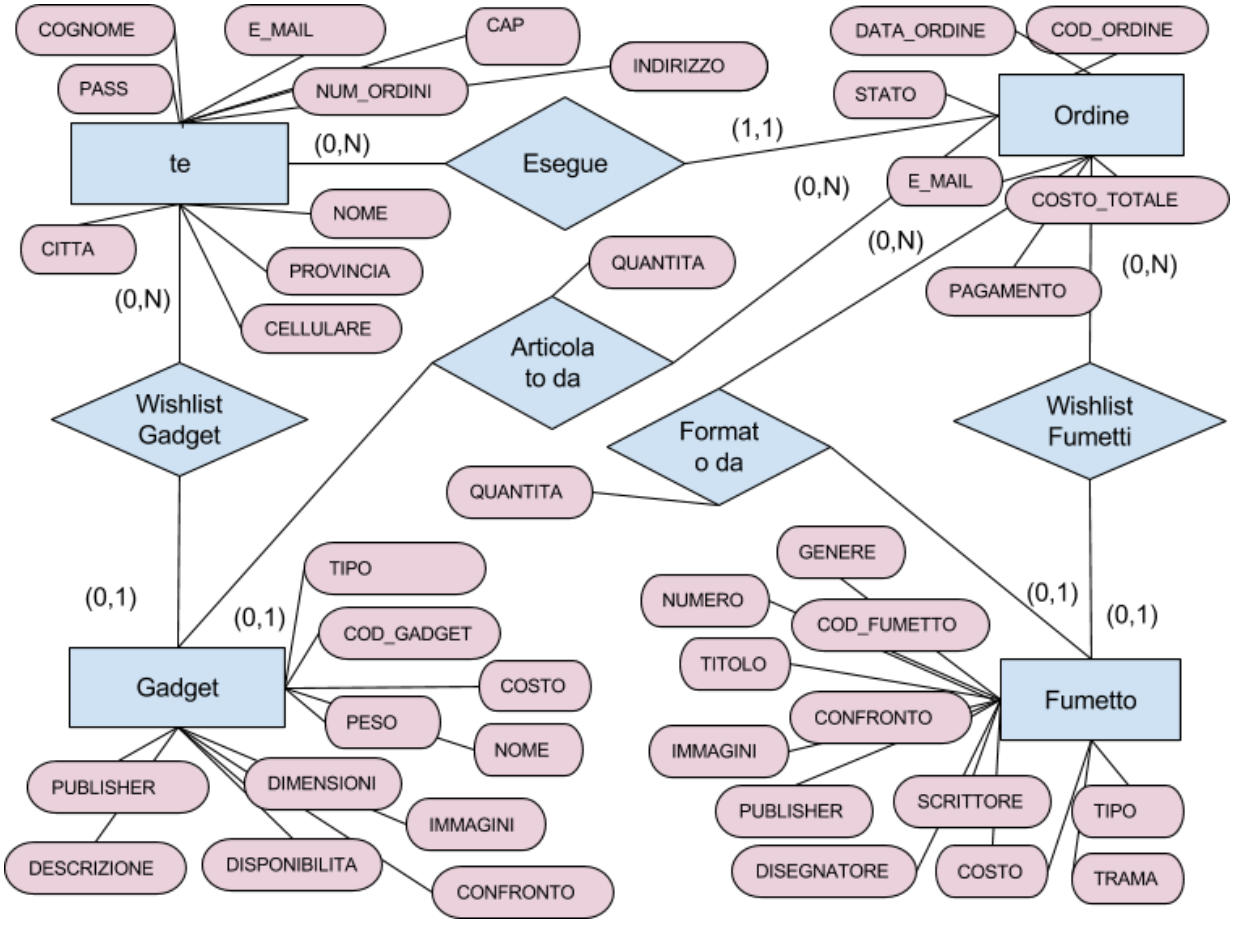
3.4 Gestione dei dati persistenti

Il sistema userà sia un servizio di storage su file, che servizi su Database. Le risorse saranno opportunamente salvate in cartelle su server.

La scelta del DBMS è ricaduta su MySQL viste le conoscenze dei membri del team. La scelta di un database relazionale, rispetto a un database object-oriented, è motivata dalle alte prestazioni offerte dai primi.

Nome Indicatore	Dati
Utente	E_mail, Nome, Cognome, Num_Ordini, Pass, Cap, Cellulare, Citta, Indirizzo, Provincia.
Ordine	Cod_Ordine, Data_Ordine, Costo_Totale, Stato, Pagamento, E_mail⇧.
Gadget	Cod_Gadget, Tipo, Peso, Costo, Dimensioni, Nome, Publisher, Descrizione, Immagini, Confronto, Disponibilità.
Fumetti	Cod_Fumetto, Genere, Costo, Scrittore, Disegnatore, Publisher, Tipo, Trama, Titolo, Numero, Immagini, Confronto, Disponibilità.

3.4.1 Scherma EER



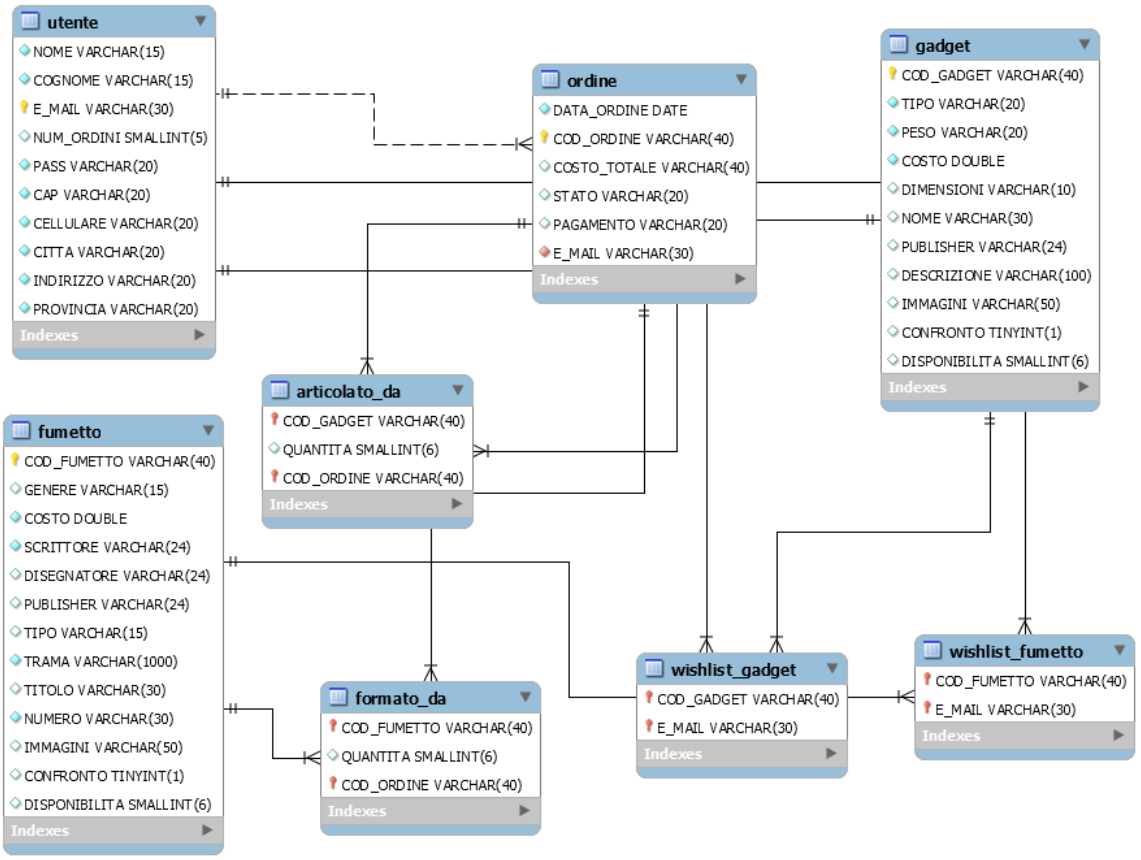
3.4.2 Dizionario dei Dati

Entità	Descrizione	Attributi	Identificatore
Utente	Elenco delle persone che utilizzano il sistema.	Nome, Cognome, Num_Ordini, Pass, Cap, Cellulare, Citta, Indirizzo, Provincia	E_mail
Ordine	Elenco degli ordini effettuati dai clienti del sistema.	Data_Ordine, Costo_Totale, Stato, Pagamento, E_mail.	Cod_Ordine
Gadget	Elenco dei Gadget presenti nel sistema.	Tipo, Peso, Costo, Dimensioni, Nome, Publisher, Descrizione, Immagini, Confronto, Disponibilità.	Cod_Gadget
Fumetto	Elenco dei Fumetti presenti nel sistema.	Genere, Costo, Scrittore, Disegnatore, Publisher, Tipo, Trama, Titolo, Numero, Immagini, Confronto, Disponibilità.	Cod_Fumetto

3.4.3 Tavola dei volumi

Concetto	Costrutto	Volume
Utente	Entità	85
Gadget	Entità	500
Fumetto	Entità	1500
Ordine	Entità	100
Esegue	Relazione	250
Articolato da	Relazione	150
Formato da	Relazione	150
Wishlist Gadget	Relazione	500
Wishlist Fumetti	Relazione	1500

3.4.4 Modello Logico



3.5 Controllo degli accessi e sicurezza

Il controllo degli accessi sarà alla base della sicurezza del nostro sistema, permettendo ad ogni utente o gestore di collegarsi al sistema tramite l'utilizzo di username e password, che verranno richieste ad ogni singolo accesso. La sessione termina quando l'utente effettua il logout.

Nel caso l'accesso al sistema non abbia successo, verrà inviata una notifica di fallimento indicando che è avvenuto un errore nell'inserimento di username o password e consentendo all'utente di effettuare un nuovo tentativo.

Il visitatore potrà usufruire dei servizi che non utilizzano la registrazione visualizzando un invito a registrarsi o loggarsi per i servizi che la richiedono.

Il sistema fornirà più viste (interfacce grafiche) a seconda dell'attore che ci interagirà, in modo che ognuno possa accedere solo alle rispettive funzionalità previste.

Sottosistema ---------- Attore	Gestione			
	Gestione Utente	Gestione Ordini	Gestione Fumetti	Gestione Gadget
Gestore	+doSave(ClientBean Client) +Login() +Logout() +getClient(String email) +doDelete(String code) +doRetrieveAll()	+doRetrieveAllAdmin()	+doRetrieveByKey(String code) +doRetrieveAll() +doAddConfrontComics(String code) +doAddCartComics(String code) +doRetrieveAllSearchComics(String search) +doRetrieveAll(String genere, String publisher, String tipo) +doRetrieveAllCartComics() +doDeleteCartComics(String code) +doDeleteConfrontComics(String code) +doSaveC(ComicsBean Comic) +doUpdateC(ComicsBean Comic, String comic) +doDeleteC(ComicsBean Comic, String comic) +QuantityComics(String code) +doRetrieveAllOrderComics(String code)	+doRetrieveByKey(String code) +doRetrieveAll(String tipo, String publisher) +doAddConfrontGadget(String code) +doAddCartGadget(String code) +doDeleteCartGadget(String code) +RetriveAllCart(String code) +doRetrieveAllSearchGadget() +doDeleteConfrontGadget(String code) +doRetrieveAllConfrontGadget() +doSaveG(GadgetBean Gadget) +doUpdateG(GadgetBean Gadget, String codgadget) +doDeleteGadget(String code) +doRetrieveAll() +QuantityGadget(String code) +doRetrieveAllOrderGadget(String code)
Utente	+send(String to, String sub, String msg, final String user, final String pass) +doUpdate(ClientBean Client, String email) +Login() +Logout() +getClient(String email)	+doRetrieveAll(String email)	+doRetrieveByKey(String code) +doRetrieveAll() +doAddConfrontComics(String code) +doAddCartComics(String code) +doRetrieveAllSearchComics(String search) +doRetrieveAll(String genere, String publisher, String tipo) +doRetrieveAllCartComics() +doDeleteCartComics(String code) +doDeleteConfrontComics(String code) +doDeleteWishlistComics(String code, String email) +doRetrieveAllWishlistC(String code) +doAddWishlistComics(String code, String email) +QuantityComics(String code) +doRetrieveAllOrderComics(String code) +Shopcomics(String comic, String order, int quantity, int availability)	+doRetrieveByKey(String code) +doRetrieveAll() +doRetrieveAll(String tipo, String publisher) +doAddConfrontGadget(String code) +doAddCartGadget(String code) +doDeleteCartGadget(String code) +doAddWishlistGadget(String code, String email) +RetriveAllCart(String code) +doRetrieveAllSearchGadget() +doDeleteConfrontGadget(String code) +doRetrieveAllConfrontGadget() +doDeleteWishlistGadget(String code, String email) +doRetrieveAllWishlistG(String code) +QuantityGadget(String code) +doRetrieveAllOrderGadget(String code) +Shopgadget(String gadget, String order, int quantity, int availability)
Visitatore	+doSave(ClientBean Client) +send(String to, String sub, String msg, final String user, final String pass) +checkEmailRegistration()		+doRetrieveByKey(String code) +doRetrieveAll() +doAddConfrontComics(String code) +doAddCartComics(String code) +doRetrieveAllSearchComics(String search) +doRetrieveAll(String genere, String publisher, String tipo) +doRetrieveAllCartComics() +doDeleteCartComics(String code) +doDeleteConfrontComics(String code)	+doRetrieveByKey(String code) +doRetrieveAll() +doRetrieveAll(String tipo, String publisher) +doAddConfrontGadget(String code) +doAddCartGadget(String code) +doDeleteCartGadget(String code) +RetriveAllCartGadget() +doRetrieveAllSearchGadget() +doDeleteConfrontGadget(String code) +doRetrieveAllConfrontGadget()

3.6 Controllo globale del software

Per quanto riguarda il flusso di controllo esterno tra sottosistemi, il server sarà sempre in funzione (a meno di manutenzioni) in attesa di eventuali richieste di servizi da parte dell'utenza; il sistema non avrà problemi a gestire più utenti contemporaneamente, ma nel caso di molte richieste contemporanee, il server potrebbe ovviare al sovraccarico utilizzando la tecnica FIFO.

3.7 Boundary conditions

Il server sarà sempre attivo permettendo l'utilizzo del servizio in qualsiasi orario, e in presenza di un malfunzionamento, il sistema mostrerà la pagina di manutenzione, non permettendo le normali operazioni previste dallo stesso. Nel caso in cui si verifichi un crash del sistema dovuto ad un errore, si tenterà un ripristino del backup più recente.

3.7.1 Start-up

Per il primo start-up del sistema "E-comix", è necessario l'avvio di un web server che fornisca il servizio di un Database MySQL per la gestione dei dati persistenti e l'interpretazione ed esecuzione del codice lato server, assumendo che prima dello start-up iniziale il database ha almeno un gestore che una volta loggato può accedere a tutte le funzionalità del sistema, alcuni utente registrati con ordini effettuati ed alcuni prodotti già inseriti.

3.7.2 Terminazione

Al momento della chiusura della web page, quindi senza effettuare il logout, la sessione rimarrà attiva.

3.7.3 Fallimento

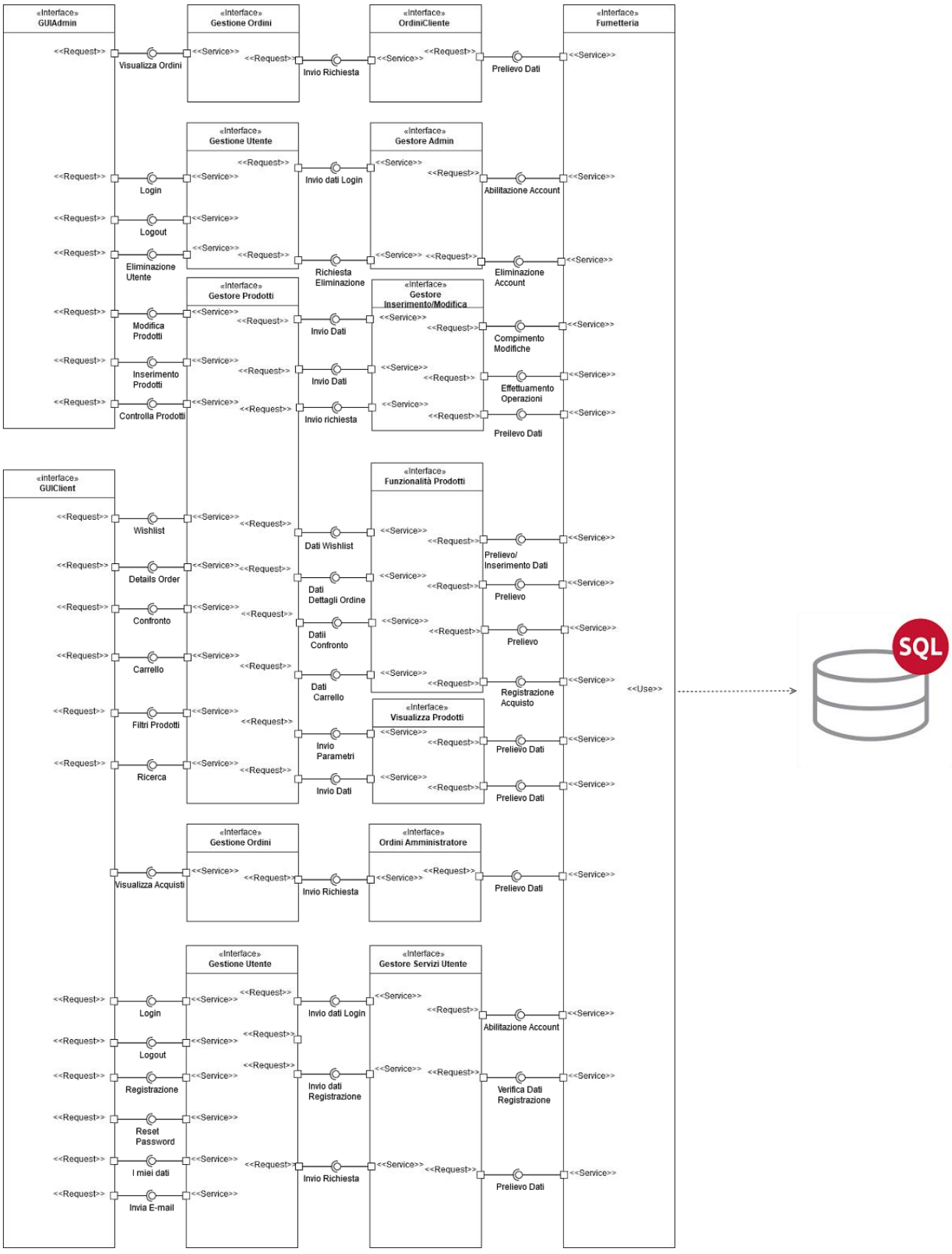
Alcuni esempi di fallimento generato dal sistema:

1. Link ad una pagina inesistente: il sistema mostrerà all’utente una pagina che lo avvisa dell’errore, cliccando sulla freccia sinistra, si potrà tornare alla pagina precedente.

Alcuni esempi di fallimenti possibili generati dalla parte Client:

1. Durante la registrazione di un nuovo utente se questo è già registrato apparirà un messaggio di errore.
2. Durante la registrazione di un nuovo utente se questo inserisce dati errati o non completi, la registrazione fallirà mostrando un messaggio di errore.
3. Nell'inserimento di un nuovo prodotto, se i campi non verranno compilati tutti o in modo errato, verrà mostrato un messaggio di errore.
4. Se il gestore od un utente non registrato cercherà di aggiungere prodotti alla lista desideri verrà generato un messaggio di errore.
5. Se il gestore tenterà di aggiungere prodotti al carrello verrà mostrato un messaggio di errore.

4. Servizi sottosistema



GUIAdmin: interfaccia che gestisce l'interazione dell'amministrazione con il sistema.

GUIClient: Interfaccia che gestisce l'interazione dell'utente con il sistema.

Gestione Utente: offre 3 servizi all'interfaccia Admin:

- Login
- Logout
- Eliminazione Utente

E 6 servizi all'interfaccia Client:

- Login
- Logout
- Registrazione
- Reset Password
- I miei Dati
- Invia e-mail

Gestione Ordini: offre un servizio all'interfaccia Admin:

- Visualizza Ordini

E offre un servizio all'interfaccia Client:

- Visualizza Acquisti

Gestione Prodotti: offre 3 servizi alla GuiAdmin:

- Modifica Prodotti
- Inserimento Prodotti
- Controlla Prodotti

E offre 6 servizi alla GuiClient:

- Wishlist
- Details Order
- Confronto
- Carrello
- Filtri Prodotti
- Ricerca

Ordini Cliente: offre un servizio a Gestione Ordini:

- Invio Richiesta

Gestore Admin: offre due servizi a Gestore Utente:
- Invio dati Login
- Richiesta Eliminazione

Gestore Inserimento Modifica: offre tre servizi a Gestore Prodotti:
- Invio Dati
- Invio Dati
- Invio Richiesta

Funzionalità Prodotti: offre 4 servizi a Gestore Prodotti:
- Dati Wishlist
- Dati Dettagli Ordine
- Dati Confronto
- Dati Carrello

Visualizza Prodotti: offre 2 servizi a Gestore Prodotti:
- Invio Parametri
- Invio Dati

Ordini Amministratore: offre un servizio a Gestione Ordini:
- Invio Richiesta

Gestore Servizi Utente: offre tre servizi a Gestore Utente:
- Invio Dati Login
- Invio Dati Registrazione
- Invio Richiesta

Fumetteria: offre i servizi a (Ordini Cliente, Ordini Amministratore, Gestore Admin, Gestore Inserimento/Modifica, Funzionalità prodotti, Visualizza Prodotti, Gestore Servizi Utente):
- Abilitazione Account
- Prelievo Dati
- Verifica Dati Registrazione
- Registrazione Acquisto
- Prelievo
- Prelievo/Inserimento Dati
- Effettuazione Operazioni
- Compimento Modifiche
- Eliminazione Account

5. Glossario

Design goals: obiettivi qualitativi del sistema, identificano le qualità su cui deve essere focalizzato il sistema.

Criteri di affidabilità: quantificazione dello sforzo che deve essere speso per minimizzare i crash del sistema e delle loro conseguenze.

Criteri di rendimento: requisiti imposti sul sistema in termini di spazio e velocità.

Query: interrogazione di un database per estrarre o aggiornare i dati che soddisfano un certo criterio di ricerca.

Criteri di manutenzione: requisiti imposti sul sistema in termini di manutenibilità.

Criteri per l'utente finale: qualità non incluse nei criteri di performance e affidabilità che sono desiderabili dal punto di vista dell'utente.

Trade-offs: compromesso.

Schema ER: un modello per la rappresentazione concettuale dei dati ad un alto livello di astrazione.

Dizionario dei Dati: permette di arricchire lo schema ER con descrizioni in linguaggio naturale di entità, relazioni ed attributi.

Tavola dei volumi: Specifica il numero stimato di istanze per ogni entità e associazione dello schema.

Modello logico: il modello logico discende dal modello concettuale e disegna l'architettura che tiene conto delle strutture proprie di quel particolare tipo di database.

Boundary conditions: soluzioni a problemi fisici.